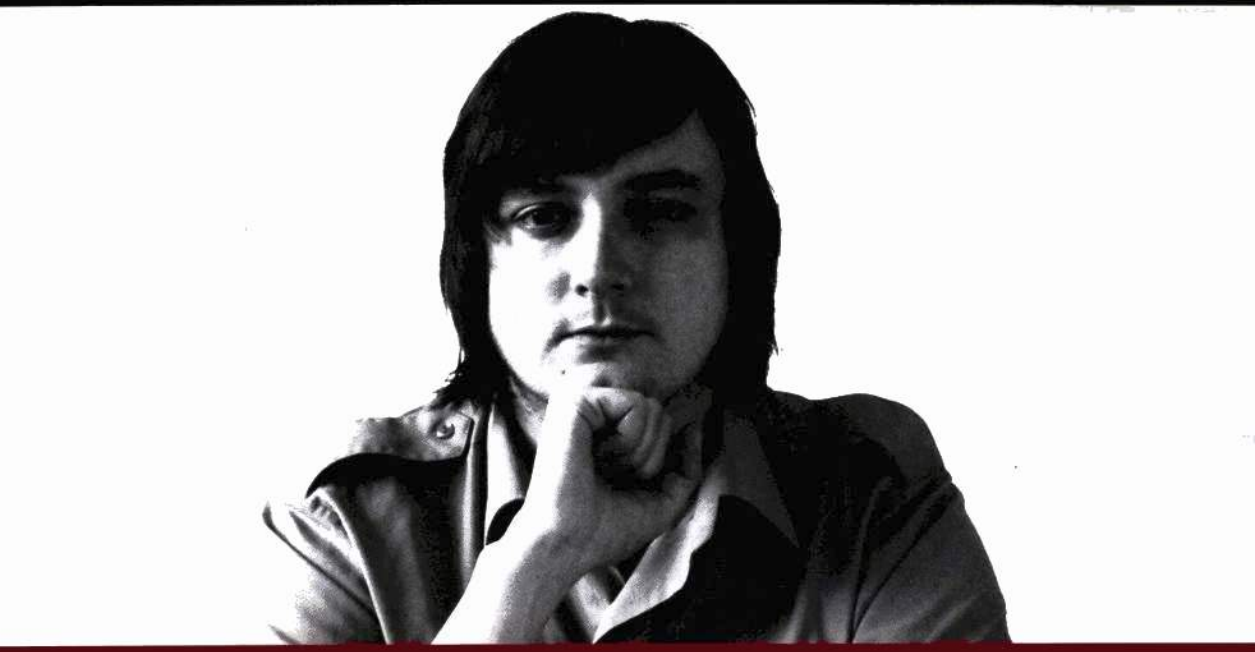
PROGRAMMER TO PROGRAMMER™



Beginning Web Programming with HTML, XHTML, and CSS, 2nd Edition

Web编程入门经典

——HTML、XHTML和CSS（第2版）

(美) Jon Duckett 著
杜静 敖富江 译

wrox™
清华大学出版社

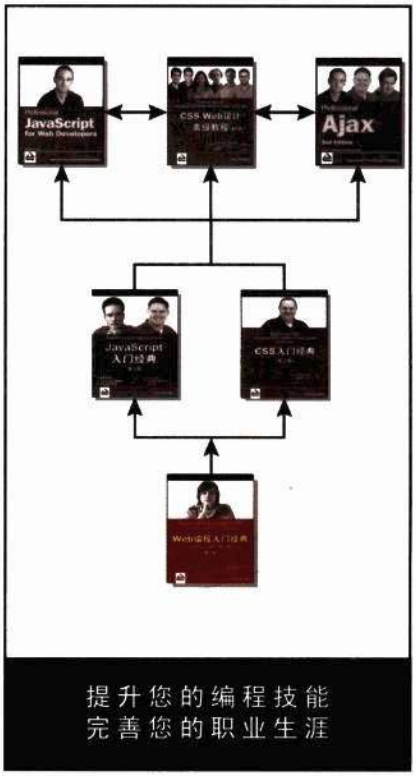
PROGRAMMER TO PROGRAMMER™

Web编程入门经典——HTML、XHTML和CSS（第2版）

Beginning Web Programming with HTML，XHTML，and CSS，2nd Edition

本书为这个大家熟悉的主题，提供了一种新的学习方法：如何创建当今存在的Web页面——以及在可预见的未来如何创建它们。仅使用HTML代码编写Web页面的年代已经一去不复返。随着Web技术的发展，为了创建有效、吸引人的Web页面，开发人员需要学习很多技术。本书回顾了HTML，也介绍了如何使用XHTML构造Web页面和用于控制页面外观的层叠样式表(CSS)。

本书通过一些典型示例探索了Web浏览器的演化，以及其如何反映Web页面的开发方式。读者将学习到如何利用浏览器的最新功能，以及如何确保所创建的页面能够工作于较老但仍在使用的浏览器中。另外，读者将了解如何为许多能够访问Web页面的设备编写页面。通过结合可用性和可访问性，开发人员将能够编写外观优美、代码专业并采用了最新技术的Web页面。



提升您的编程技能
完善您的职业生涯

本书主要内容

- 组成HTML和XHTML的不同元素和属性，以及如何使用它们编写Web页面
- 利用CSS使页面更吸引人并更容易操作
- JavaScript的基础知识，以便开发人员在Web页面中添加交互
- 如何将站点发布到Internet中，如何为站点找到访问者，以及如何让搜索引擎识别站点

本书读者对象

本书适用于想学习如何创建Web页面的读者，也适用于想提高Web设计技能水平的读者。本书的读者不需要具有编程或者Web页面编写知识。

本书源代码下载及技术支持

http://www.wrox.com

http://www.tupwk.com.cn/downpage

Wrox Beginning guides are crafted to make learning programming languages and technologies easier than you think, providing a structured, tutorial format that will guide you through all the techniques involved.

ISBN 978-7-302-21597-4

定价：79.80元

wrox™
An Imprint of WILEY

图书上架分类建议	Web编程 HTML、XHTML、CSS

读者信箱：wkservice@vip.163.com
投稿信箱：bookservice@263.net

Web 编程入门经典

——HTML、XHTML 和 CSS

(第 2 版)

(美) Jon Duckett 著
杜静 敖富江 译

清华大学出版社
北 京

Jon Duckett
Beginning Web Programming with HTML, XHTML, and CSS，2nd Edition
EISBN: 978-0-470-25931-3

图书在版编目(CIP)数据

Web 编程入门经典——HTML、XHTML 和 CSS(第 2 版)/ (美) 达科特(Duckett，J.)著；杜静，敖富江 译. —北京：清华大学出版社，2010.1

书名原文：Beginning Web Programming with HTML，XHTML，and CSS，2nd Edition
ISBN 978-7-302-21597-4
Ⅰ. ①W…　Ⅱ. ①达…　②杜…　③敖…　Ⅲ. ①超文本标记语言，HTML、XHTML—主页制作—程序设计　②主页制作—软件工具，CSS　Ⅳ. ①TP312②TP393.092

中国版本图书馆 CIP 数据核字(2009)第 226811 号

责任编辑：王　军　于　平
装帧设计：孔祥丰
责任校对：成凤进
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　　总　　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：40.75　**字　数**：992 千字
版　　次：2010 年 1 月第 1 版　　**印　　次**：2010 年 1 月第1 次印刷
印　　数：1～4000
定　　价：79.80 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：030084-01

前　言

目前关于设计和构建 Web 页面的书籍很多，因此首先感谢您选择阅读本书。本书与其他书籍的不同点是什么呢？Web 概念已经出现了十多年，在这期间引入了大量用于创建 Web 页面的技术，其中一些技术目前还在使用，而其他技术已经消失。许多介绍如何编写 Web 页面的书籍是相同书籍早期版本的修订，因此仍然采用与以前版本相同的介绍方法。而本书的目的是介绍如何为当今以及将来的 Web 创建页面。因此，读者阅读完本书之后，仍可将其作为一本有用的参考书放在身边，在需要时随时翻阅。

曾经有一段时间，编写 Web 页面的程序员仅需要掌握一种编程语言，即 HTML 语言。但是随着 Web 技术的发展，为了创建有效并吸引人的 Web 页面，程序员需要学习更多的技术，需要掌握多种不同语言，主要包括：

- HTML 和 XHTML：HTML 和 XHTML 用于解释 Web 页面的结构。它们用于指明哪些文本作为题头，段落的起始位置和结束位置在何处，哪些图像应当出现在文档中，以及指定不同页面之间的链接。不应当将 HTML 和 XHTML 看作是两种独立的语言。相反，可以将 XHTML 看作是与 HTML 的最新版本非常相似的语言。
- CSS：CSS 用于控制文档的外观。例如，可以使用 CSS 指定字型应该是较大的、粗体的 Arial 字型，或者指定页面的背景应该是亮绿色。另外，还可以使用 CSS 控制不同项在页面上的位置。例如，利用 CSS 将文本放置在同一个页面的两列中。
- JavaScript：利用 JavaScript 可以在创建的 Web 页面上添加交互性，并且可以操作显示 Web 页面的浏览器。

尽管事实上需要掌握多种语言(而不只是 HTML)，但是可以将 HTML 看作熟悉 Web 的好机会，因为 HTML 中用于创建 Web 页面的许多技术拥有成熟的、有利的方法；或者将其看作“最佳实践”，因为可以利用它创建完整的 Web 站点。

本书简介

本书将介绍如何利用 HTML 和 XHTML 控制 Web 页面的结构，如何利用 CSS 赋予 Web 页面样式，如何利用 JavaScript 添加交互性。但是，只是学习最新的技术并不能确保编写出优秀的 Web 页面。由于用于编写 Web 页面的技术不断改进，因此浏览器(用于访问 Web 的程序和设备)也在改进。浏览器反映——并且有时甚至是通知——用于创建 Web 页面的语言的发展方向。问题在于并不是每个人都在其计算机上安装了最新软件，因此人们不仅希望编写出的 Web 页面能够充分利用浏览器的最新功能，而且希望确保 Web 页面能够在当今仍然流行的较老浏览器上正确显示。

因为 Web 页面的构建方式不断改变，并且存在多种不同版本的 Web 浏览器，所以本书中列举的某些功能被标记为“逐渐淘汰”，这意味着虽然该功能仍然能够工作于现代的浏

览器中，但是不再建议使用它，因为软件可能不会始终支持它。

在编写 Web 页面时需要了解的另一个问题是，能够访问 Web 的设备正在不断增加，例如移动电话、PDA(个人数字助理)和电视机顶盒。这些设备通常利用相同的语言，本书中将介绍这些语言——通过学习结合使用 XHTML 和 CSS，相比于仅采用旧的 HTML 编写的 Web 站点，您将能够创建出生命力更持久的 Web 站点。

最近几年来，Web 技术改变的另外一个领域是越来越强调可用性和可访问性。可用性是指使 Web 站点易于被用户浏览，并使用户获得他们到达您的站点所希望查找的内容；而可访问性是指使尽可能多的用户能够访问 Web 站点，特别是一些具有行为障碍的人(他们可能具有受损的视力或者无法使用鼠标)。在世界上的许多国家，如果站点不能满足严格的访问指导原则，政府不会同意构建该 Web 站点。在构建 Web 站点之前仔细思考，将意味着视力受损的人们能够查看具有较大文本的站点，或者通过使用屏幕阅读器阅读站点。有些书籍专门讨论可用性和可访问性主题，它们的目标是需要学习如何使代码具有更好的可访问性和可用性的 Web 开发人员，但本书的目标是使开发人员从一开始就牢记这些原则。

阅读完本书之后，读者编写的 Web 页面将不仅能够使用最新的技术，而且仍然能够通过较老的浏览器查看。外观优美的页面仍然能够被视力受损或身体受损的人们访问。这些页面不仅能够满足当今访问者的需求，也能够工作于正在出现的技术。并且，读者掌握的技巧将会在相当长的一段时间内有效。

本书读者对象

本书适用于希望学习如何创建 Web 页面的读者；也适用于已经从事过 Web 页面的编写(或许使用过某种 Web 页面制作工具)，但是希望真正理解 Web 编程语言以便创建更优秀页面的读者。

有经验的 Web 开发人员也能够从本书受益，因为本书介绍了某些最新的技术，例如 XHTML，并且鼓励读者接受某些 Web 标准，这些标准不仅满足能够访问 Web 的新设备的需求，而且能够使 Web 站点获得更多的访问者。

本书的读者不需要具有任何编程经验，这是一本编程的入门书籍。无论读者是编程爱好者，还是希望从事 Web 编程职业，本书都将教授 Web 编程的基础知识。当然，术语“程序员”通常与“低层次”联系在一起，但是阅读完本书之后，即使希望被称作为 Web 设计人员，也需要知道如何编写代码以便创建出完善的 Web 站点。

本书主要内容

阅读完本书之后，读者将能够创建出具有专业外观并且代码优美的 Web 页面。

读者不仅将学习如何编写由标记语言(例如 XHTML)组成的代码，而且将看到如何应用这些代码创建具有复杂布局的页面、定位文本和图像在页面中的出现位置，以及获得所需的颜色和字体。在这个过程中，您将看到如何使页面能够易于使用，并且获得尽可能多的访问者。本书还将介绍一些实用技术，例如如何使 Web 站点能够在 Internet 上被访问，如

何使搜索引擎识别 Web 站点。

本书介绍的主要技术包括 HTML、XHTML 和 CSS，并且介绍关于 JavaScript 的基础知识，这些知识足以用于在一些示例页面中添加交互性，并使读者能够处理一些基本脚本。在这个过程中，本书还向读者介绍将来可能希望学习的一些其他技术。

本书鼓励读者编写的代码遵循 Web 标准：HTML、XHTML 和 CSS 均由 World Wide Web 联盟创建和维护，World Wide Web 联盟也简称为 W3C(网址为 www.w3.org/)，这是一个专门致力于创建 Web 规范的组织。本书也介绍某些非标准的功能，当读者遇到这样的标记并需要知道它的作用时，了解这些功能是非常有帮助的(遇到这种情况时，将专门指明这些功能不是标准的组成部分)。

使用本书的要求

使用本书时，读者需要的全部条件包括：一台安装了 Web 浏览器(最好是 Firefox 2 或更高版本、Safari 2 或更高版本、Internet Explorer 6 或更高版本)的计算机以及一个简单的文本编辑器(例如 Windows 的 Notepad 或 Mac 的 TextEdit)。

如果读者具有一个 Web 页面编辑器程序，例如 Macromedia Dreamweaver 或 Microsoft FrontPage，也可以使用该程序，但本书不会讲解如何使用这些程序。这些程序各有不同，并且已经有介绍它们的专门书籍。即使具有这样的程序，但在真正理解这些程序生成的代码之后，则能够编写出更好的站点，因为经常需要手动编写和编辑这些代码。

本书组织结构

本书第 1 章将介绍如下内容：创建 Web 站点的主要任务是利用其构成元素和属性标记出现在站点上的文本。元素和属性描述了文档的结构(哪些是题头，哪些是文本的段落，哪些是链接等)。

本书前 6 章描述了组成 HTML 和 XHTML 的不同元素和属性，以及如何使用它们编写 Web 页面。以任务相关的方式将这几章组织成不同的领域，例如将一个文档结构化为题头和段落，创建页面之间的链接，添加颜色和图像，显示表等。在介绍每一个任务和主题时，本书都将首先给出示例，向读者演示可以实现的功能；然后更详细地介绍使用的元素和属性。

需要提醒的是，读者在第一次阅读本书时不需要阅读元素每个方面的详细解释——只需要了解该元素可以实现的功能即可。出于完整性(并将相关信息保持在相同位置)的原因，本书中介绍了一些可能会很少用到的功能。当发现需要使用某些比较难以理解的功能时，可以返回来重新仔细阅读它们。因此，如果希望更快地阅读本书，只需要记住标识的要点，然后继续下去。

每一章的结束部分给出了一些练习，以方便读者温习所学的概念。如果需要返回并重新阅读该章的内容以完成练习，不要为此感到担心。本书的目的是为读者在未来的几年中提供有用的参考，因此不要感觉需要用心学习所有内容。在阅读本书的过程中，读者将看到每个元素能够被哪些浏览器支持，并将学习到大量有用的提示和技巧以及创建专业 Web

页面的技术。

学习如何利用 HTML 和 XHTML 创建和结构化文档之后，读者接下来将学习如何使用层叠样式表(Cascading Style Sheet，CSS)创建更具吸引力的 Web 页面，并且学习如何改变所使用字体的字型和大小、文本的颜色、各项的背景和周围的边框，以及如何将对象对齐在页面的中间、左边或右边等知识。

学习完这两章(第 8 章和第 9 章)之后，利用本书中的示例将能够编写非常复杂的 Web 页面。这些章节的内容可以作为有用的参考，在将来可以重新查阅它们，其中的示例可以作为构建自己的 Web 站点的工具箱。

第 9 章和第 10 章着重介绍 Web 页面设计问题。读者将看到一些流行的 Web 页面布局示例以及构造它们的方式，并且学习如何创建优秀的导航栏，以便用户在您的站点找到所需的页面；您将发现使表单有效的各个方面；并且将学习如何使 Web 站点能够被尽可能多的人访问。这两章实际上基于本书前半部分中介绍的理论，有助于创建具备专业外观的页面，以便吸引用户并使站点更易于使用。

第 11 章和第 12 章介绍 JavaScript，这是一种称为脚本语言的编程语言，可将其用于 Web 页面中。虽然完整的 JavaScript 语言非常庞大，无法在这两章中全面介绍它，但是读者应当能够了解它的工作方式，并了解如何将脚本集成到页面中。

第 13 章(也就是最后一章)将介绍如何为将站点发布到 Internet 上做准备，并介绍 Web 主机托管、FTP 和代码验证方面的知识。最后，该章给出阅读完本书后可以继续执行什么操作的概念；您可能希望在 Web 站点上添加大量其他内容，或者希望通过学习进一步提高 Web 编程技能，本章将给出其他可以实现的功能以及需要学习的相关知识。

本书包含了几个有用的附录，包括 XHTML 元素和 CSS 特性的参考。其中一个附录解释了 XHTML 和 CSS 如何指定颜色。其他附录分别介绍可利用的字符编码、语言代码以及一些能够用于 HTML、XHTML、CSS 和 JavaScript 的转义字符。最后一个附录列出了一些较老的标记，实际上不应当再使用这些标记，这里列举它们只是为了防止读者在创建站点时遇到一些较老的 Web 技术。

源代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 http://www.wrox.com/ 或 www.tupwk.com.cn/downpage 上下载。登录到站点 http://www.wrox.com/，使用 Search 工具或使用书名列表可以找到本书。接着单击本书细目页面上的 Download Code 链接，可以获得所有的源代码。

注释：

由于许多图书的标题都很类似，所以按 ISBN 搜索是最简单的，本书英文版的 ISBN 是 978-0-470-25931-3。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以

进入 http://www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是错误总是难免的，如果您在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

请给 wkservice@vip.163.com 发电子邮件，我们会检查您的反馈信息，如果是正确的，我们将在本书的后续版本中采用。

要在网站上找到本书英文版的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/booklist.shtml。

P2P.WROX.COM

要与作者和同行讨论，请加入 p2p.wrox.com 上的 P2P 论坛。这个论坛是一个基于 Web 的系统，便于您张贴与 Wrox 图书相关的消息和相关技术，与其他读者和技术用户交流心得。该论坛提供了订阅功能，当论坛上有新的消息时，它可以给您传送感兴趣的论题。Wrox 作者、编辑和其他业界专家和读者都会到这个论坛上来探讨问题。

在 http://p2p.wrox.com 上，有许多不同的论坛，它们不仅有助于阅读本书，还有助于开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读使用协议，并单击 Agree 按钮。

(3) 填写加入该论坛所需要的信息和自己希望提供的其他信息，单击 Submit 按钮。

(4) 您会收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

注释：

不加入 P2P 也可以阅读论坛上的消息，但要张贴自己的消息，则必须加入该论坛。

加入论坛后，可以张贴新消息，响应其他用户张贴的消息。可以随时在 Web 上阅读消息。如果要让该网站给自己发送特定论坛中的消息，可以单击论坛列表中该论坛名旁边的 Subscribe to this Forum 图标。

关于使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作情况以及 P2P 和 Wrox 图书的许多常见问题。要阅读 FAQ，可以在任意 P2P 页面上单击 FAQ 链接。

目　录

第1章

创建结构化文档

本章介绍编写 Web 页面所需要学习的首要技术：HTML 和 XHTML。事实上，您将学习的是 XHTML——后面会解释 HTML 和 XHTML 之间的区别。(可以将 XHTML 简单看作为 HTML 的最新版本)。

本章的主要目的是演示 XHTML 如何描述文档的结构。

在本章中将介绍以下内容：

- 学习标签、元素和属性之间的区别
- 了解 Web 页面如何使用标记描述结构化页面的方式
- 介绍用于标记题头和段落等文本的元素
- 学习许多其他元素，这些元素能够在文档中添加表示信息和语义
- 了解如何在文档中添加项目列表和编号列表
- 介绍一些用于区分 XHTML 中元素类型的核心概念

阅读完本章后，读者能够了解如何使用 XHTML 构造页面，并能够编写自己的第一个 Web 页面。

1.1 结构化文档组成的 Web

在日常生活中，人们会接触各种类型的打印文档——报纸、列车时刻表、保险单。Web 由大量链接在一起的文档所组成，这些文档类似于日常生活中的文档。因此，首先回顾日常生活中看到的一些文档的结构，并将它们与 Web 页面作比较。

每天早上我习惯阅读报纸。报纸由一些故事和文章(很可能还有一些广告)组成。每个故事具有一个标题、一些段落，或许还有一个副标题以及其他更多的段落，还可能包含一幅或两幅图片。

我一般不买日报，因为常常在线观看新闻，但 Web 站点中的文章的结构与报纸中文章的结构非常相似。每篇文章也是由题头、文本段落和图片组成。两者非常相似，仅有的区别是，Web 站点中的每个故事具有它自己的页面，访问该页面仅需要单击它的标题行或者简短概述，这些标题行和简短概述可能位于站点的主页面或者站点的某个分部(例如政治、体育或者娱乐部分)的主页面。

举一个例子：假设乘火车去看望一个朋友，因此需要查看列车时刻表以了解到他那里

的列车车次。列车时刻表的主要部分是一张表，表中列举了列车在各个站的到达和离开时间。如同许多文档具有标题和段落一样，许多其他的文档也使用表；从报纸财经增刊中的股票信息页面到电视节目播出列表，人们每天都会遇到一些包含信息的表——并且这些表通常也被创建在 Web 中。

另外一种常见的文档是表单。例如，桌面上有一张保险公司的表单。在这张表单中有一些需要填写的字段，包括姓名、地址、保险金额，也有一些方框，需要在这些方框中进行选择以指出房子中房间的数量和大门门锁的类型。事实上，Web 页面中存在大量表单，从询问查找内容的简单搜索框到在线订购书籍或 CD 之前要求填写的注册表单。

根据上面的介绍可以发现，日常生活中打印文档的结构和 Web 中页面的结构非常类似。因此，在开始编写 Web 页面时，您会很容易地理解代码将告诉 Web 浏览器需要显示信息的结构——哪些文本被放置在标题、段落或者表中等——以便浏览器能够正确将它们显示给用户。

为了告诉 Web 浏览器文档的结构——如何建立标题、段落、表等——您需要学习 HTML 和 XHTML。

1.2 XHTML 简介

XHTML(可扩展超文本标记语言)与它的前身 HTML 是 Web 上最广泛使用的语言。从名称中可以看出，XHTML 是一种标记语言，它看上去可能很复杂，但如果意识到每天都会遇到各种标记，这种观点就会改变。

当在文字处理软件中创建文档时，可以对文本添加样式以解释文档的结构。例如，可以使用题头样式(通常具有较大的字体)区分标题和文本主体，使用 Enter(或 Return)键开始一个新的段落，在文档中插入表以放置数据，或者为一系列相关的要点创建项目列表等。这些样式会影响文档的表现形式，但是这种标记的关键目的是提供一种结构，以使文档更容易理解。

当标记 Web 文档时，实际上执行的是一种非常相似的过程，不同的只是需要在 Web 文本上添加标签。对于 XHTML，关键是要记住添加标签以指明文档的结构，即文档的哪一部分是题头，哪些部分是段落，哪些内容属于表等。浏览器(例如 Internet Explorer、Firefox 和 Safari 等)将使用这种标记以用户熟悉的方式显示文本，它的行为类似于文字处理软件(标题的字体大于正文，段落间存在一些空格，项目列表的每一项前面有一个圆点)。然而，这些内容的表现方式由浏览器决定；XHTML 规范没有说明应当使用哪些字体或者该字体的尺寸。

注意：

早期版本的 HTML 允许控制文档的表现形式——例如文档应当使用哪些字型和颜色——但是 XHTML 标记不应该用于样式化文档，这些工作由 CSS 完成，第 7 章中将介绍 CSS。

首先查看一个非常简单的 Web 页面。前言中曾提到，编写 Web 页面不需要任何特殊的程序——可以只使用一个文本编辑器，例如 Windows 中的 Notepad 或 Mac 中的 TextEdit。

编辑完成之后，以.html 文件扩展名的形式存储文件。可以从 Wrox 的 Web 站点 www.wrox.com 处下载本书中的所有代码，其中就有这个示例，它位于 Chapter 1 文件夹中，文件名为 ch01_eg01.html。

```
<html>
  <head>
     <title>Popular Websites: Google</title>
  </head>
  <body>
    <h1>About Google</h1>
    <p>Google is best known for its search engine, although
       Google now offers a number of other  services.</p>
    <p>Google's mission is to organize the world's
       information and make it universally accessible and
       useful.</p>
    <p>Its founders Larry Page and Sergey Brin started
       Google at Stanford University.</p>
  </body>
</html>
```

初看上去，这些代码不容易理解，但下面会逐一解释它们。在这段代码中有几组尖括号，其中包括一些单词或字母，例如<html>、<head>、</title>和</body>。尖括号和其中的单词一起称为标签，这些就是前面讨论过的标记。图 1-1 所示为该页面在浏览器中的外观。

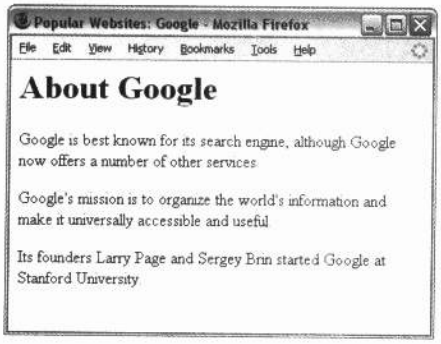


图 1-1

这个文档中包含了题头“About Google”和一个介绍 Google 公司的文本段落。另外注意，在该浏览器窗口的左上角处显示了“Popular Websites: Google”，它称为该页面的标题。

为了理解第一个示例中的标记，需要查看尖括号中编写的内容，并将它们与图中所见的内容比较，具体请阅读接下来的小节。

1. 标签和元素

查看上面示例中的第一行和最后一行代码，可以看到包括字母 html 的成对的尖括号。两个成对的尖括号及其中的所有字符称为标签，该示例中存在许多标签。本示例中的所有标签都是成对出现；在某对标签中，前面的标签称为起始标签，后面的标签称为结束标签。

两者的不同点是结束标签在第一个尖括号后面有一个正斜杠。

某对标签和其中包含的内容统称为元素。图 1-2 中给出了上面示例中页面的题头。

图 1-2

起始标签表明“这是题头的开始”，结束标签表明“这是题头的结束”。类似于 XHTML 中的大多数标签，尖括号内部的文本解释了该标签的目的——这里的 h1 表明它是第一级题头(或顶级题头)。后面将会看到，也有表示副标题的标签(<h2>、<h3>、<h4>、<h5>和<h6>)。如果不标记“About Google”，则它将只是另一段文本；无法表明它是题头。

下面查看关于公司介绍的 3 段文本，每一段位于起始标签 和结束标签 之间，其中 p 表示段落。

注意：

由于以下两个基本概念非常重要，因此这里再次给出它们的定义：标签由尖括号和它们中间的字母和数字组成，而元素由标签以及起始标签和结束标签之间的文本组成。

这个示例中的标记实际上描述了标签之间的内容，而标签提供的额外意义是它描述了文档的结构。在起始标签 和结束标签 之间是段落，而在<h1>标签和</h1>标签之间是题头。事实上，整个文档被包含在起始标签<html>和结束标签</html>之间。

人们经常采用家族树中的术语描述元素间的关系。例如，包含其他元素的元素称为父元素，而父元素的起始标签和结束标签之间的元素称为该元素的子元素。因此，<title>元素是<head>元素的子元素，而<head>元素是<title>元素的父元素。此外，<title>元素可以被认为是<html>元素的孙元素。

注意：

XHTML 标签只能以小写字母编写。

2. 头和主体

当采用 XHTML 编写 Web 页面时，整个页面包含在起始标签<html>和结束标签</html>之间，如同前面的示例所示。在<html>元素中，存在两个主要的页面部分：

- <head>元素：通常称为页面的头，它包含了关于整个页面的信息(但不是页面的主要内容)，具体的信息包括页面的标题和描述或者搜索引擎用于索引页面用的关键字。<head>元素由起始标签<head>和结束标签</head>以及两者之间的所有文本组成。

- <body>元素：通常称为页面的主体，它包含人们在浏览器主窗口中实际看到的信息。<body>元素由起始标签<body>和结束标签</ body >以及两者之间的所有文本组成。

在第一个示例页面的<head>元素中，可以看到<title>元素，如下所示：

```
<head>
   <title>Popular Websites: Google</title>
</head>
```

在起始标签<title>和结束标签</title>之间是“Popular Websites: Google”，它是这个 Web 页面的标题。图 1-1 是这个 Web 页面的屏幕截图，前面曾提醒读者注意浏览器窗口顶部右边的单词。这是浏览器(例如 IE、Firefox 和 Safari)显示文档标题的位置，它也是在收藏夹中保存页面时使用到的名字。

页面的实际内容位于<body>元素中，这是希望用户阅读的内容，并且显示在浏览器主窗口中。

注意：

head 元素中包含了关于文档的信息，这些信息不显示在主页面中。body 元素存储页面的实际内容，可以在浏览器中查看到这些内容。

可能已经注意到，示例中的标签以对称顺序出现。如果希望某个元素位于另外一个元素之中，则该元素的起始标签和结束标签都必须位于包含其的元素之中。例如，下面的语句是正确的：

```
<p> This paragraph contains some <em>emphasized text.</em></p>
```

而下面的语句是错误的，因为结束标签没有在段落元素之中：

```
<p> This paragraph contains some <em>emphasized text. </p></em>
```

换句话说，如果一个元素包含另外一个元素，则它必须完全包含该元素这称为正确地嵌套元素。

3. 元素的属性

Web 文档和普通文档的区别在于链接(或超链接)，利用链接可以从一个 Web 页面转向另外一个页面。下面查看链接的示例，该示例在前面的示例中添加一个链接。创建链接的方法是使用<a>元素，其中 a 是 anchor 的缩写。

这里将添加从该页面到 Google 的链接，添加位置在文档末尾的新段落中，新的示例(代码为 ch01_eg02.html)中仅需要添加一个新行，并且突出显示该行：

```
<html>
  <head>
     <title>Popular Websites: Google</title>
  </head>
  <body>
    <h1>About Google</h1>
```

```
    <p>Google is best known for its search engine, although Google now offers a
       number of other services.</p>
    <p>Google's mission is to organize the world's information and make it
       universally accessible and useful.</p>
    <p>Its founders Larry Page and Sergey Brin started Google at Stanford
University.</p>
    <p><a href="http://www.Google.com/">Click here to visit Google's Web
    site.</a></p>
  </body>
</html>
```

在这个新的段落中，<a>元素创建了链接。在起始标签<a>和结束标签之间是用户可以单击的文本，本示例中的文本是“Click here to visit Google's Web site.”。图 1-3 中所示为这个页面在浏览器中的外观。

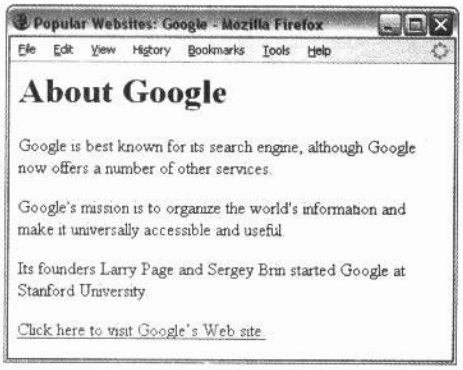


图 1-3

如果仔细观察这个链接的起始标签，会发现它附带一个属性。在这个示例中，属性是 href；属性的后面紧跟着一个等号，然后是到 Google 的 Web 站点的 URL，该 URL 包含在引号中。在这个示例中，href 属性指明链接的目标 URL。下一章中将详细介绍链接，而这里的示例说明了属性的作用。

属性用于补充说明附带它们的元素，它们通常出现在附带它们的元素的起始标签中。属性由两部分组成：名称和值：

- 名称是想要设置的元素的特性。在本示例中，<a>元素附带属性的名称是 href，该属性用于指明链接的目标 URL。
- 值是希望该特性具有的值。在本示例中，值是链接的目标 URL，因此 href 属性的值是“http://www.Google.com”。

属性的值必须放置在双引号中，并且通过等号与名称隔开。如果希望在新窗口中打开链接，可以在起始标签<a>中添加一个 target 属性，并将其值设置为_blank：

```
<a href="http://www.Google.com" target="_blank">
```

该示例说明一个元素可以附带多个属性，但是一个元素不能有两个同名的属性。

注意：

所有的属性都由两部分组成，即属性的名称和属性的值，它们之间通过等号隔开。值应该放置在双引号中，而所有 XHTML 属性的名称都应当以小写字母编写。

4. XML 声明

有时在 XHTML 文档的起始部分存在 XML 声明。XHTML 语言事实上是使用另一种称为 XML(Extansible Markup Language，可扩展标记语言，通常用于创建标记语言)的语言编写的，并且任何 XML 文档都可以从可选的 XML 声明开始：

```
<?xml version="1.0" encoding="UTF-8" ?>
```

如果在文档中添加 XML 声明，则它必须位于文档的起始位置，并且其前面不能有任何内容，甚至不能有空格。encoding 属性指明文档中使用的编码方式。

说明：

encoding(character encoding 的缩写)表示程序或操作系统如何存储用户想要显示的字符。不同的语言具有不同的字符，由于某些程序比其他程序支持更多的字符，所以存在多种不同的编码方式。

5. 文档类型声明

前面提及，XHTML 是 HTML 的后继——尽管可以将其认为是 HTML 的最新版本。XHTML 比它的前身 HTML 遵循更严格的语法。例如，在 XHTML 中元素和属性的名称必须小写(早期的 HTML 版本不区分大小写)，每个具有内容的元素必须有对应的结束元素，某些元素和属性可能被标记为逐渐淘汰——这意味着它们可能在将来的 XHTML 版本中被逐渐淘汰。

因此，每个 XHTML 页面应当以一个 DOCTYPE 声明开始，以告诉浏览器(或任何其他程序)该页面中使用的 HTML 或 XHTML 版本。

虽然这里将 XHTML 作为一种语言讨论，但实际上已经发布了 3 种版本的 XHTML——发布这些版本是为了帮助现有 Web 开发者从 HTML 向 XHTML 转变：

- Transitional XHTML 1.0，该版本仍然允许开发人员使用 HTML4.1 中的不赞成使用标记，但是要求程序设计人员使用新的更严格的语法。
- Strict XHTML 1.0，用于告诉 XHTML 的向前路径，它不需要逐渐淘汰标记，但是遵从新的更严格的语法。
- Frameset XHTML 1.0，该版本用于创建使用框架技术的 Web 页面，第 6 章中将介绍框架。

不要因为 HTML 和 XHTML 具有多个版本而感到不安，本书中将主要介绍 Transitional XHTML 1.0。在介绍过程中，将了解哪些元素和属性已经被标记为逐渐淘汰。如果可以避免使用逐渐淘汰的元素和属性，则编写的代码遵循 Strict XHTML 1.0。

DOCTYPE 声明在文档中的位置位于起始标签<html>之前，如果在文档中使用了可选的 XML 声明，则 DOCTYPE 声明位于 XML 声明之后。

如果编写的代码遵循 Transitional XHTML 1.0(并且在文档中包含样式标记)，则 DOCTYPE 声明应该如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

如果编写的代码遵循 Strict XHTML 1.0，则 DOCTYPE 声明如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

对于框架集文档(第 6 章中将介绍这种文档)，DOCTYPE 声明如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

注意：

Strict XHTML 文档必须在根元素之前包含 DOCTYPE 声明；但是，如果创建的是 Transitional 或 Frameset 类型的文档，则不需要包含 DOCTYPE 声明。

学习完 Transitional XHTML 1.0 之后，应当能够理解较老版本的 HTML，并且通过这些知识可以确信，编写的 XHTML 代码将能够工作于当前 Web 上使用的大多数浏览器中。

1.3 核心元素和属性

现在已经介绍了如何使用描述文档结构的元素标记 Web 页面的内容，下一步将介绍一些元素，利用这些元素可以描述可能希望显示在 Web 上的各种类型文档的结构。本章剩余部分和接下来的几章将描述所有这些元素。

在介绍每个元素时，本书将给出它们的使用方法和可能采用的属性。这将使本书可以作为一本在学会如何编写 Web 页面之后供您查阅的完整参考手册。但是，当您第一次阅读本书时，如果觉得已经理解了元素的意义，则可以忽略某些内容，在需要时再重新阅读它们。

本节首先介绍组成文档基本结构的 4 个主要元素：<html>、<head>、<title>和<body>。这 4 个元素会出现在您所编写的每一个 XHTML 文档中，它们可以被看作文档的框架。

1.3.1 <html>元素

<html>元素是整个 XHTML 文档的包含元素。在可选的 XML 声明和必需的 DOCTYPE 声明之后，每个 XHTML 文档应当具有一个起始标签<html>，并且应当以一个结束标签</html>作为结束。

如果编写的代码遵循 Strict XHTML 1.0，则起始标签必须也包含名称空间标识符(namespace identifier；用于表明文档中的标记属于 XHTML 1.0 名称空间)。因此，起始标签应当如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

虽然在 Transitional XHTML 文档中并不严格要求包含<html>元素，但最好在所有的 XHTML 文档中都包含它。

仅有两个元素是<html>元素的直接子元素：<head>和<body>(尽管<head>和<body>元素通常包含更多的元素)。

<html>元素还可以附带如下属性，本章后面的“属性组”一节将介绍这些属性：

```
id dir lang xml:lang
```

说明：

在 HTML 4.1 和更早的版本中包含 version 属性，以表明文档所使用的 HTML 版本，但该属性现在已经被淘汰。XHTML 文档应该使用 DOCTYPE 声明和 xmlns 属性来指示使用的 XHTML 版本。

1.3.2　<head>元素

<head>元素是所有其他头元素的容器，它应该紧跟在起始标签<html>之后。

每个<head>元素应当包含一个<title>元素以指示文档的标题，它也可以以任意顺序包含如下元素的任意组合：

- <base>，第 2 章中将介绍该元素。
- <object>，用于包含图像、JavaScript 对象、Flash 动画、MP3 文件、QuickTime 电影以及页面的其他组件，第 3 章中将介绍该元素。
- <link>，用于链接到外部文件，例如样式表或 JavaScript 文件，第 7 章中将介绍这些内容。
- <style>，用于在文档中包含 CSS 规则，第 7 章中将介绍该元素。
- <script>，用于在文档中包含脚本，第 11 章中将介绍这些内容。
- <meta>，用于包含与文档相关的信息，例如关键字和文档的描述，它们特别有助于搜索应用程序，第 13 章中将介绍这些内容。

说明：

profile 属性实际上还没有使用，尽管 XHTML 规范中包含该属性，但只是为了在将来使用它指定配置文件的 URL，配置文件描述文档的内容。其他的属性将在本章后面的“属性组”一节中介绍。

起始标签<head>能够附带如下属性：

```
id dir lang xml:lang profile
```

1.3.3　<title>元素

编写每个页面时，应该给其指定一个标题。标题位于<title>元素中(在本章前面的部分中已经见到过该元素的使用方法，它是<head>元素的子元素)。<title>元素的使用方式包括以下几种：

- 位于浏览器窗口的顶部(如第一个示例和图 1-1 所示)
- 作为浏览器(IE、Firefox 和 Safari)中书签的默认名称
- 通过使用其内容的搜索引擎以帮助索引页面

因此，使用实际描述站点内容的标题是非常重要的。例如，站点的主页面不应当只使用“Home Page”标注；而是应当采用能够描述站点内容的语句。例如，对于 Wrox 的主页面，不是简单地采用“Wrox Home Page”，而是采用如下方式：

```
<title>Wrox: Books for programmers written by programmers</title>
```

对于优秀的页面标题，访问者在阅读它之后就应当能够了解该页面的内容，而不需要查看页面的实际内容。

<title>元素中只能包含关于页面标题的文本，而不能包含其他任何元素。<title>元素可以附带如下属性，在本章后面的“属性组”一节中将介绍这些属性：

```
id dir lang xml:lang
```

1.3.4 <body>元素

<body>元素出现在<head>元素之后，它包含用户能够在浏览器主窗口中看到的部分 Web 页面内容，这部分内容有时称为主体内容。该元素包含的内容广泛，从位于标题下的一些段落到包含表单和表的复杂布局，因此它是 XHTML 文档的主要组成部分。本章和后面 4 章中介绍的大部分内容均位于起始标签<body>和结束标签</body>之间。

<body>元素中可以附带“属性组”一节中介绍的所有属性。如果使用的是 Transitional XHTML 和 HTML 4.1，则可以在<body>元素中使用下面几个逐渐淘汰的属性(参见附录 I)：

```
background bgcolor alink link vlink text
```

另外，<body>元素中还能够使用一些浏览器专用属性，附录 I 中也介绍了这些属性：

```
language, topmargin, bottommargin, leftmargin, rightmargin, scroll,
bgproperties, marginheight, marginwidth
```

1.4 属性组

本章前面已经介绍过，属性位于元素的起始标签中，并为附带它们的元素提供额外的信息。所有的属性均由名称和值组成；名称反映了属性描述的元素的特性，值是该特性的值。例如，xml:lang 属性描述了元素中使用的语言；值为 EN-US 时，表明元素中使用的语言是美国英语。XHTML 中的大多数元素可以附带本节中介绍的部分或所有属性。

存在 3 个属性组，大多数 XHTML 元素可以附带它们(<html>、<head>、<title>和<body>元素共享其中一些属性)。如果此时感觉它们比较抽象，请不要担心；随着更深入地阅读本书，一切问题将迎刃而解。由于这些属性能够被大多数元素使用，这里将对它们进行分组，以避免每次遇到它们时重复描述。此时如果不能完全理解它们，可以记住此处描述它们的位置，在后面需要时返回重新阅读。3 个属性组分别是：

- 核心属性：class、id 和 title 属性
- 国际化属性：dir、lang 和 xml:lang 属性
- UI 事件：与如下事件关联的属性：onclick、ondoubleclick、onmousedown、onmouseup、onmouseover、onmousemove、onmouseout、onkeypress、onkeydown 和 onkeyup(第 11 章中将更详细地介绍它们)。

说明：
核心属性和国际化属性统称为通用属性。

1.4.1　核心属性

核心属性共有 4 个，它们能够被大多数(但不是所有)XHTML 元素使用：

```
id title class style
```

对于不同于此处描述的其他元素，这些属性有时具有专门的含义，但是在下面的小节中将给出这些属性的普通用法。

1. id 属性

id 属性可用于唯一标识页面内的任何元素。用户通常希望唯一标识元素，以便能够链接到文档中的特定部分，或者便于指定元素，从而将 CSS 样式或 JavaScript 与文档中某个元素的内容关联。

id 属性的语法如下所示(其中 string 是为该属性选择的值)：

```
id="string"
```

例如，id 属性可用于区分两个段落元素，如下所示：

```
    <p id="accounts">This paragraph explains the role of the accounts
department.</p>
    <p id="sales">This paragraph explains the role of the sales department.</p>
```

注意，对于 id 属性的值存在一些特殊的规则：

- 必须以字母(A-Z 或 a-z)开头，然后可以跟上任意数量的字母、数字(0-9)、连字符、下划线、冒号和句号(不能以数字、连字符、下划线、冒号或句号开头)。
- 在某个文档中保持唯一；在同一个 XHTML 文档中，任意两个 id 属性不能具有相同的值。

在引入 id 属性之前，name 属性在 HTML 文档中发挥着相同的作用，但在 HTML 4.01 中将该属性标识为逐渐淘汰，因此现在编写 XHTML 文档时应当使用 id 属性。name 属性仅可在 Transitional XHTML 中使用，但不能在 Strict XHTML 中使用(如果使用的是较老的、引入 id 属性之前的浏览器，则需要使用 name 属性)。

2. class 属性

id 属性用于唯一标识特定的元素，而 class 属性用于指定某个元素属于特定的元素类。

class 属性通常用于 CSS，第 7 章中介绍 CSS 时将更详细地讨论 class 属性的使用。class 属性的语法如下所示：

```
class="className"
```

属性值也可以是以空格隔开的类名称的列表。例如：

```
class="className1 className2 className3"
```

3. title 属性

title 属性给出元素的建议标题，它的语法如下所示：

```
title="string"
```

这个属性的行为依赖于使用它的元素，尽管它通常显示为加载元素时的工具提示。

并不是每一个能够使用 title 属性的元素都需要该属性，因此在后面遇到特别需要使用这个属性的元素时再详细解释它用于该元素时的行为。

4. style 属性(逐渐淘汰)

利用 style 属性能够指定元素中的 CSS 规则。第 7 章中将介绍 CSS，但是下面的示例给出了 style 属性的使用方式：

```
<p style="font-family:arial; color:#FF0000;">Some text</p>
```

但是，最好尽量不要使用这个属性。在 XHTML 1.0 中这个属性被标识为逐渐淘汰(这意味着它在将来的 XHTML 版本中可能被移除)。如果需要使用 CSS 规则来指定元素的显示方式，最好单独使用一个样式表，在第 7 章中介绍 CSS 时将讨论相应的技术。

1.4.2 国际化属性

存在 3 种国际化属性，用户可以使用它们编写具有不同语言和字符集的页面。大多数(尽管不是全部)XHTML 元素都能够使用这些国际化属性(对于支持多种语言的文档来说，它们是非常重要的属性)。

```
dir lang xml:lang
```

即使是最新浏览器也仍然不能很好地支持这些属性，因此最好不要指定根据所需的方向创建文本的字符集，尽管 xml:lang 属性能够被其他支持 XML 的应用程序使用。

下面是包含了有用的 W3C 文档的 Web 地址，这些文档详细地描述了国际化问题，但是下面将简单地介绍这些国际化属性：

```
http://www.w3.org/TR/i18n-html-tech/
```

注意：

国际化属性有时称为 i18n 属性，这个奇怪的名称来源于 draft-ietf-html-i18n 规范，该规范中第一次定义了这个属性。

1. dir 属性

利用 dir 属性可以指定文本在浏览器中的显示方向。当需要指定整个文档(或文档的大部分)的方向性时，应当在<html>元素中使用 dir 属性，而不是在<body>元素中使用该属性，原因有两点：<html>元素能够被浏览器更好地支持，并且随后该属性应用于头元素以及主体中的元素。如果希望改变文档一小部分内容的方向，也可以将 dir 属性用于文档主体的元素中。

dir 属性可以使用表 1-1 所示两个值中的一个值。

表 1-1

值	意　义
ltr	从左到右(默认值)
rtl	从右到左(某些语言，例如希伯来语或阿拉伯语，从右向左阅读)

2. lang 属性

利用 lang 属性可以指示文档中使用的主要语言，但在 XHTML 中保留这个属性只是为了与早期的 HTML 版本向后兼容。在新的 XHTML 文档中，lang 属性已经被 xml:lang 属性代替(下一节中将介绍 xml:lang 属性)。但是，XHTML 规范建议在 XHTML 1.0 文档的<html>元素中同时使用 lang 属性和 xml:lang 属性(以在不同的浏览器之间获得最大的兼容性)。

设计 lang 属性是为了向用户提供语言特有的显示，尽管它对主要的浏览器具有较小的影响。使用 lang 属性后，真正受益的是搜索引擎(搜索引擎利用它能够告诉用户采用哪一种语言编写文档)、屏幕阅读器(屏幕阅读器利用它能够以不同的方式发音不同的语言)以及一些应用程序(应用程序能够在它们不支持所提供的语言或者该语言与它们的默认语言不同时向用户发出警报)。当 lang 属性用于<html>元素中时，它将作用于整个文档；而在用于其他元素中时，它将仅作用于这些元素的内容。

lang 属性的值是 ISO-639 标准两字符语言代码。如果希望指定某种语言的方言，可以在语言代码后面紧跟一个破折号和一个子代码名称。表 1-2 中给出一些示例。

表 1-2

值	意　义
ar	阿拉伯语
en	英语
en-us	美国英语
zh	中文

在附录 G 中列出了当今使用的大多数主流语言的语言代码。

3. xml:lang 属性

在 XHTML 中，xml:lang 属性是 lang 属性的替代属性。以 XML 编写的所有语言中都

可以使用该属性。本章前面曾提到，XHTML 是采用 XML 编写的，因此它具有字符前缀“xml:”。xml:lang 属性的值应该是 ISO-639 国家代码，例如前面小节中列出的几种国家代码；附录 G 中包含了 ISO-639 国代码的完整列表。

虽然 xml:lang 属性对主要的浏览器没有影响，但其他支持 XML 的应用程序和搜索引擎可以使用这些信息，因此在文档中包含 xml:lang 属性是一种良好的习惯。当用于<html>元素中时，该属性将作用于整个文档；而在用于其他元素中时，它仅作用于这些元素的内容。

1.4.3 UI 事件

利用 UI 事件属性可以将事件(例如按键事件或者将鼠标指针移动到某个元素上)与脚本(一段编程代码，当事件发生时运行该代码)相关联。例如，当某个人将鼠标指针移动到某个特定元素的内容上时，可能使用一个脚本改变内容的颜色。

第 14 章中将详细介绍 UI 事件。UI 事件的名称能够非常清晰地表明它们所关联的事件；例如，当用户单击元素的内容时激活 onclick 事件，当鼠标移动时激活 onmousemove 事件，当用户将鼠标指针移出某个特定元素的内容时激活 onmouseout 事件。

下面 10 个事件统称为公共事件：

```
onclick, ondoubleclick, onmousedown, onmouseup, onmouseover, onmousemove,
onmouseout, onkeypress, onkeydown, onkeyup
```

<body>和<frameset>元素也有如下两个事件，分别在页面打开和关闭时激活：

```
onload  onunload
```

最后，还存在大量只作用于表单的事件(第 5 章中将提及这些事件，并且在第 11 章中将再次介绍它们)：

```
onfocus, onblur, onsubmit, onreset, onselect, onchange
```

到目前为止，本书已经介绍了一些预备知识，并且讨论了一些组成 XHTML 文档的程序框架的元素。接下来讨论如何标记出现在 Web 页面上的文本。

1.5 基本文本格式

本章前面已经介绍了 XHTML 文档的程序框架结构和一些核心属性，因此现在可以了解如何标记文本以便描述它的结构。因为创建的几乎每一个文档都将包含一些形式的文本，所以下面将介绍的元素是大多数页面的基本构件。

在阅读本部分之前，请记住如下要点：某种浏览器可能以一种特定方式显示每种元素，而另一种浏览器可能以完全不同的方式显示它们；不同的浏览器显示的字体大小可能不同(因此同一块文本占用的空间也不同)，显示的字型也可能不同。本书直到第 7 章才会介绍如何控制文本显示的外观(字型、颜色、字体大小)。

本节中将介绍如何使用如下的基本文本格式元素：

```
h1, h2, h3, h4, h5, h6

p, br, pre
```

如果希望人们阅读您所编写的内容，则在 Web 上良好地结构化文本甚至比编写文本更重要。在 Web 站点上，人们无法很好地阅读长的、宽的文本段落，除非正确地拆分这些段落(第 9 章中将介绍这方面的内容)。因此，在开始 Web 开发职业生涯时就保持良好的习惯，将有助于确保您创建的页面更具吸引力。

在开始学习用于标记文本的元素之前，了解文本默认的显示方式会非常有帮助(如果希望以不同的方式显示文本，则必须告诉浏览器以何种方式显示文本)。

1.5.1　空格和流

在开始标记文本之前，最好先理解当 XHTML 遇到空格时的处理方式，以及浏览器如何处理长句子和文本段落。

您可能会认为，如果在两个单词中间放置几个连续的空格，则在屏幕上这些空格将出现在这两个单词之间，但是实际情况并非如此；默认情况下只会显示一个空格。这种情况称为空格折叠。同样，如果在源文档中开始一个新行，或者放置多个连续的空行，则这些新行将被忽略并被处理为一个空格。对制表符的处理也是如此。例如，查看如下段落(该段落源自于代码示例中的 ch01_eg03.html)：

```
<p>This  paragraph shows how  multiple spaces    between    words are
treated as a single space. This is known as white space collapsing, and
the big spaces between   some of the   words will not appear  in the
browser.

It also demonstrates how the browser will treat multiple carriage returns
(new lines) as a single space, too.</p>
```

图 1-4 所示，浏览器将多个空格和几个回车(用于将文本换行)处理为一个空格。

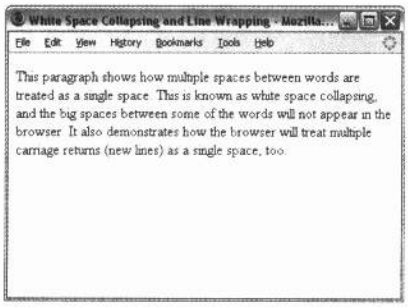


图 1-4

图 1-4 也表明，当浏览器显示文本时，如果到达某行的末尾，则会自动在新行中继续显示文本。请再次查看这个示例的代码，并查看浏览器中新行的起始位置，结果表明这些文本在屏幕上的显示位置和在代码中的位置是不同的。可以自己实验这种情况，因为本书

的下载代码中提供了所有示例；可以只改变浏览器窗口的大小(缩小或放大)，并注意文本如何在屏幕上的新位置中换行。

这种处理空格的方式非常有帮助，因为开发人员可以在代码中添加多个空格，而这些空格不会在实际的文档中显示，但是利用它们能够缩进代码，使其更易于阅读。本章前两个示例演示了缩进代码，其中子元素向左缩进，从而与它们的父元素区分。本书中的所有代码都采用这种缩进方式，以便具有更好的可读性(如果希望在文档中保留空格，需要使用<pre>元素，本章后面的部分将介绍这些元素；或者使用 实体参考，附录 F 中将介绍这方面的内容)。

因此，学习如何使用一些元素控制文本的显示方式非常重要，本章后面的部分中将介绍这些元素。

1.5.2 使用 hn 元素创建题头

无论创建的是何种类型的文档，它们都会具有某些形式的题头。报纸具有大字标题，表单上的题头给出表单的目的，体育运动结果表上的标题给出球队所属的联盟或分区等。

对于较长的文本，题头有助于描述文档的结构。如果查看本书的内容表，可以看到如何安排不同级别的题头以添加本书的结构，并且在主题头下具有副标题。

XHTML 提供了 6 级题头，分别使用元素<h1>、<h2>、<h3>、<h4>、<h5>和<h6>实现。虽然浏览器支持以不同的方式显示题头，但是它们通常将<h1>元素显示为最大的题头，而将<h6>元素显示为最小的题头，并且可以利用 CSS 重写这 6 个元素的字体大小和样式。不同级别题头的显示效果如图 1-5 所示(ch01_eg04.html)。

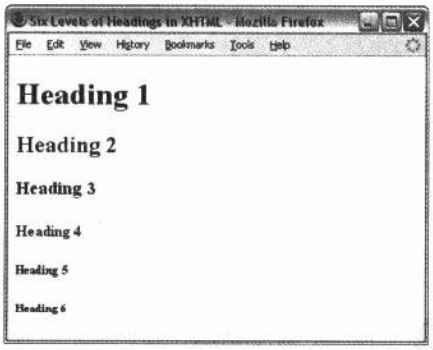


图 1-5

说明：

默认情况下，大多数浏览器显示的<h1>、<h2>和<h3>元素内容大于文本在文档中的默认尺寸。<h4>元素的内容与默认文本的大小相同，而<h5>和<h6>元素的内容较小一些。

下面是利用题头描述文档结构的另一个示例(ch01_eg05.html)，其中<h2>元素是<h1>元素的副标题(这段代码实际上模仿了本章中当前一节的结构)。

```
<h1>Basic Text Formatting</h1>
<p> This section is going to address the way in which you mark up text.
```

```
Almost every document you create will contain some form of text, so this
will be a very important section. </p>
<h2>Whitespace and Flow</h2>
<p> Before you start to mark up your text, it is best to understand what
XHTML does when it comes across spaces and how browsers treat long sentences
and paragraphs of text.</p>
<h2>Creating Headings Using hn Elements</h2>
<p> No matter what sort of document you are creating, most documents have
headings in some form or other...</p>
```

图 1-6 所示为这段代码的显示效果。

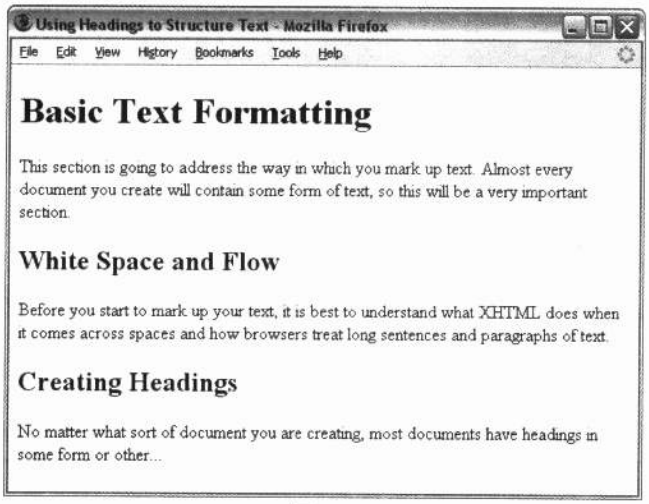


图 1-6

这 6 个题头元素可以附带所有的通用属性以及逐渐淘汰的 align 属性，如下所示：

```
align class id style title dir lang xml:lang
```

align 属性(逐渐淘汰)

逐渐淘汰的 align 属性指示题头是出现在页面的左边、中间或右边(默认是左边)，它可以采用表 1-3 所示的 3 个值中的一个值。

表 1-3

值	意　义
left	题头显示在浏览器窗口的左边(如果题头嵌套在另一个元素中，则显示在包含元素的左边)。如果没有使用 align 属性，则 left 是默认值
center	题头显示在浏览器窗口的中间(如果题头嵌套在另一个元素中，则显示在包含元素的中间)
right	题头显示在浏览器窗口的右边(如果题头嵌套在另一个元素中，则显示在包含元素的右边)

在此处提到 align 属性是因为后面很多元素都会用到它。该属性被标记为逐渐淘汰，这是因为它并不帮助描述文档的结构——而是用于影响页面的外观，现在应该使用 CSS 实现它的功能。下面是使用 align 属性的示例(ch01_eg06.html)：

```
<h1 align="left">Left-Aligned Heading</h1>
   <p>This heading uses the align attribute with a value of left.</p>
<h1 align="center">Centered Heading</h1>
   <p>This heading uses the align attribute with a value of center.</p>
<h1 align="right">Right-Aligned Heading</h1>
   <p>This heading uses the align attribute with a value of right.</p>
```

图 1-7 给出了 align 属性在浏览器中的显示效果。

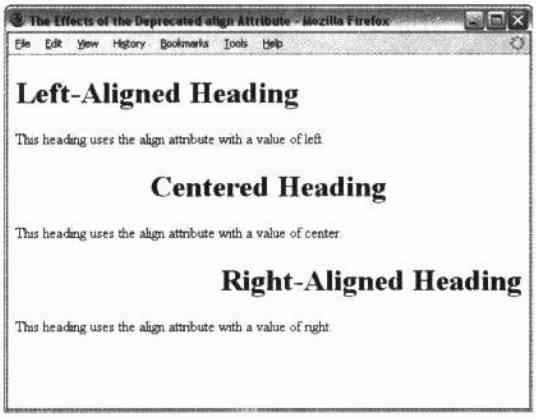


图 1-7

align 属性已经被 CSS 中的 text-align 属性取代。附录 I 中给出了关于 align 属性的更详细介绍。

1.5.3 使用 元素创建段落

元素提供了结构化文本的另一种方式。每一个文本段落都应当包含在起始标签 和结束标签 之间，如下面的示例所示(ch01_eg07.html)：

```
<p>Here is a paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
```

当浏览器显示段落时，它通常在下一个段落之前插入一个新行，并且添加一小段额外的垂直空间，如图 1-8 所示。

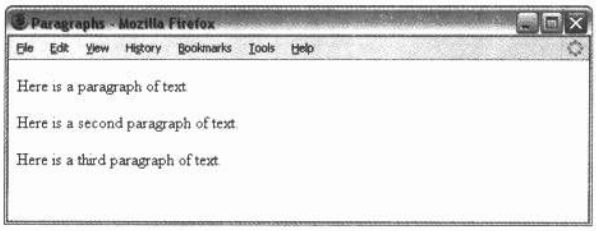


图 1-8

<p>元素能够附带所有的通用属性和逐渐淘汰的 align 属性：

```
align class id style title dir lang xml:lang
```

1.5.4　使用
元素创建换行

当使用
元素时，它后面跟随的内容将从下一行开始显示。
元素是一种空元素，它不需要起始标签和结束标签，因为没有内容填充在这些标签之间。

注意：

*
元素在字符 br 和正斜杠之间存在一个空格。如果省略这个空格，则较老的浏览器无法正确显示换行，而如果遗漏正斜杠，只使用
，则它将是无效的 XHTML。*

大多数浏览器允许使用多个
元素以将文本向下推几行。另外，许多设计人员喜欢在文本段落之间使用两个换行而不是使用<p>元素结构化文本，如下所示：

```
Paragraph one<br /><br />
Paragraph two<br /><br />
Paragraph three<br /><br />
```

虽然这样能够产生与使用段落元素类似的效果，但是如果没有使用<p>元素本身描述每个段落，则文档不会再描述每一个段落的起始部分和结束部分。此外，在 Strict XHTML 中，
元素仅能够用于块级元素中。对于块级元素，例如<p>元素，通常按照这些元素的前面和后面具有一个换行一样进行处理。在本章的结束部分将更详细地介绍块级元素。

注意：

*避免仅使用
元素决定文本的位置；这样可能产生预料之外的结果，因为这样做所创建的空间量取决于字体的大小。相反，应当使用 CSS，第 7 章中将介绍 CSS。*

下面是在段落中使用
元素的示例(ch01_eg08.html)：

```
<p>When you want to start a new line you can use the &lt;br /&gt; element.
So, the next<br />word will appear on a new line.</p>
```

图 1-9 中给出了单词“next”和“do”之后的换行。

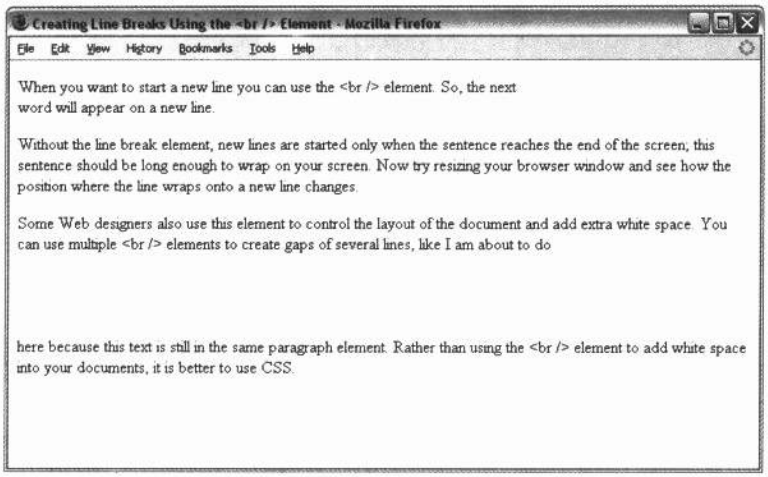


图 1-9

元素可以附带核心属性和 clear 属性，clear 属性用于图像，附录 I 将介绍该属性：

```
clear class id style title
```

1.5.5 使用<pre>元素创建预先格式化的文本

有时希望文本的显示格式与它在 XHTML 文档中编写的格式相同。例如，在文本到达浏览器的边界时，不希望它换行；不希望忽略多个空格；而希望文本在放置换行符的位置换行。

起始标签<pre>和结束标签</pre>之间的文本将保留源文档中的格式。然而需要知道的是，大多数浏览器在默认情况下将以等宽字体显示这些文本(Courier 是等宽字体的示例，因为字母表中的每个字母占用相同的宽度。在非等宽字体中，*i* 通常比 *m* 窄)。

<pre>元素的两种最常见的使用方法是：显示列表数据而不使用表(此时必须使用等宽字体，否则列无法正确对齐)；表示计算机源代码。例如，下面的代码将一段 JavaScript 源代码放到<pre>元素中(ch01_eg09.html)。

```
<pre>
function testFunction(strText){
   alert (strText)
}
</pre>
```

图 1-10 中给出了<pre>元素中的内容以等宽字体显示的结果。更重要的是，图中保留了<pre>元素中所示的格式——空格被保留。

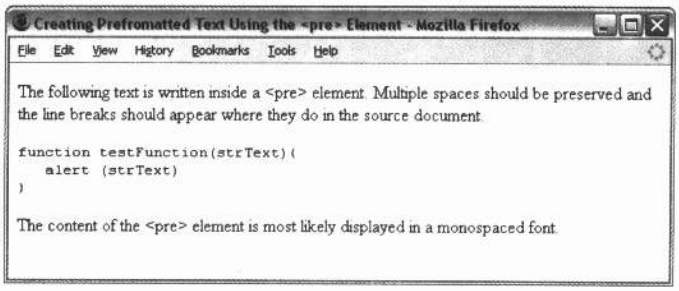


图 1-10

注意：

虽然制表符字符在<pre>元素中有效，并且一个制表符应该表示 8 个空格，但是不同的浏览器中制表符的实现方式不同，因此最好使用空格代替。

在本章的“短语元素”一节中将介绍更多可用于表示代码的元素，例如<code>、<kbd>和<var>元素。

注意：

Firefox、IE 和 Safari 支持 XHTML 规范中用于阻止换行的扩展元素：<nobr>元素(该元素保留其包含元素的普通样式，并且不会导致文本以等宽字体显示)。由于它是扩展元素，因此不是有效的 XHTML。附录 I 中将介绍<nobr>元素。

试一试　基本文本格式

现在已经介绍了一些用于格式化文本(题头和段落)的基本元素，接下来有必要练习它们。

在本示例中需要创建一个关于爵士乐传奇人物的新页面，并且这个页面介绍 Miles Davis。因此，请准备好文本编辑器或 Web 页面制作工具，并且完成以下步骤：

(1) 要求创建的文档遵循 Strict XHTML，因此添加 XML 声明和 DOCTYPE 声明，以表明编写的是 Strict XHTML 文档：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

(2) 添加文档的程序框架：<html>、<head>、<title>和<body>元素。<html>根元素附带 xmlns 属性，以表明标记属于 XHTML 名称空间。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
    <title>Jazz Legends - Miles Davis</title>
  </head>
  <body>
```

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

(3) 页面将拥有一个主题头和一些二级题头，这些题头给出了人们即将看到的页面结构：

```
<body>
  <h1>Jazz Legends - Miles Davis</h1>
  <h2>Styles of Miles</h2>
  <h2>Davis the Painter</h2>
</body>
```

(4) 现在可以在页面的题头下填充一些段落：

```
<body>
  <h1>Jazz Legends - Miles Davis</h1>
    <p>Miles Davis is known to many as one of the world's finest jazz musicians
    and an outstanding trumpet player. He also earned great respect in the
    world of music as an innovative bandleader and composer.</p>
  <h2>Styles of Miles</h2>
    <p>Miles Davis played and wrote in a variety of styles throughout his
    career, from tunes that have become jazz standards to his more
    experimental improvisational work. </p>
    <p>In the 1950s Miles was known for a warm, rich, wispy sound and was able
    to vary the color of his sound, pitch. He was also adept in using a Harmon
    mute. In the 1960s Miles began to play more in the upper register. In 1969
    he even incorporated the use of electronic instruments in his music.</p>
  <h2>Davis the Painter</h2>
    <p>Miles' love was not only for music; he is also considered a fine
    painter. Inspired by a Milan-based design movement known as Memphis,
    Miles painted a series of abstract paintings in 1988.</p>
  </body>
</html>
```

(5) 将文件保存为 miles.html 并在 Web 浏览器中打开它，结果如图 1-11 所示。

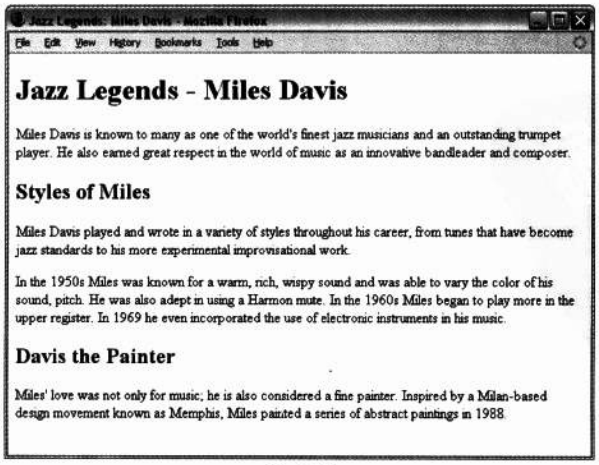


图 1-11

工作原理

这个页面的起始行是可选的 XML 声明。因为这是一个 Strict XHTML 文档(因此也是 XML 文档)，所以需要包含 XML 声明。下一行是 DOCTYPE 声明，Strict XHTML 文档中中要求具有该声明。DOCTYPE 声明表明文档遵循的 XHTML 版本。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

然后整个页面包含在<html>根元素中。起始标签<html>中附带了名称空间标识符，这是表明文档所包含的标记是 XHTML 的另一种方式。<html>元素也附带了 lang 属性，该属性指示文档所采用的语言。此处的 Web 页面是以英语编写的，因此使用两个字母的 ISO 代码表示英语(附录 G 中给出了所有国家代码的列表)。虽然当前 lang 属性几乎没有什么实际意义，但是它将对面向未来的文档起到帮助作用。

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

<html>元素可以仅包含两个子元素：<head>元素和<body>元素。<head>元素包含页面的标题，根据页面的标题能够了解页面所包含信息的类型。

```
<head>
   <title>Jazz Legends: Miles Davis</title>
</head>
```

另一方面，<body>元素包含了 Web 页面的主要部分——用户在 Web 浏览器的主要部分中实际看到的内容。另外注意，这个页面中包含了一些题头以结构化页面中的信息，如同字处理文档中的题头一样。

文档中存在不同级别的题头以帮助增强它的结构。在这个示例中，有一个介绍 Miles Davis 的主题头——这个页面的主题——然后是两个副标题，分别介绍 Miles Davis 的音乐和其他兴趣方面的信息。

不要忘记在末尾放置结束标签</html>——必须正确地结束每一个元素。

1.6　表现元素

如果使用过字处理程序，则会熟悉将文本设置为粗体、斜体或添加下划线；对于 HTML 和 XHTML 来说，这些只是可用于指示文本外观的 10 个选项里面的 3 个。这 10 个选项分别是：粗体、斜体、等宽字体、添加下划线、添加删除线、电传打印字体、较大字体、较小字体、上标和下标。

从技术上来说，这些元素仅影响文档的表现，但是在 Transitional XHTML 1.0 和 Strict XHTML 1.0 中都保留了它们。在本章后面的部分中将介绍一些专门用于指示强调一段文本的元素，这些元素将导致相似的信息表现。

下面讨论的表现元素都能够附带本章前面介绍的通用属性和 UI 事件属性。

注意：

另外需要注意的是，可以使用 CSS 获得相似的结果，第 7 章中将介绍 CSS。

1.6.1 <b>元素

出现在<b>元素中的文本将加粗显示，如下所示：

```
The following word uses a <b>bold</b> typeface.
```

这并不一定意味着浏览器必须使用加粗版本的字体。有些浏览器使用算法加粗组成字体的线(使其外观为粗体)，而有些浏览器可能突出显示文本或在文本下方添加下划线(如果它们不能找到加粗版本的字体)。

注意：

<b>元素与元素具有相同的效果。本章后面将介绍元素，利用该元素可以指示着重强调它的内容。

1.6.2 <i>元素

<i>元素的内容显示为斜体文本，如同下面的单词 italic 所示：

```
The following word uses an <i>italic</i> typeface.
```

这并不一定意味着浏览器必须使用斜体版本的字体。大多数浏览器使用算法倾斜文本以模仿斜体字体。

注意：

<i>元素与元素具有相同的效果。本章后面将介绍元素，利用该文本可以指示强调它的内容。

1.6.3 <u>元素(逐渐淘汰)

<u>元素的内容被添加一条下划线：

```
The following word would be <u>underlined</u>
```

在 HTML 4 和 XHTML 1.0 中，<u>元素被标记为逐渐淘汰，尽管它仍然被当前的浏览器支持。首选的方法是使用 CSS 实现这种效果，第 7 章中将对此进行介绍。

1.6.4 <s>元素和<strike>元素(逐渐淘汰)

<s>元素或<strike>元素的内容上方显示一条删除线，删除线是一条通过文本的细线(<s>是<strike>的缩写形式)。

```
The following word would have a <s>strikethrough</s>.
```

在 HTML 4.1 和 Transitional XHTML 1.0 中，<s>元素和<strike>元素都被标记为逐渐淘汰的，而在 Strict XHTML 1.0 中它们已经被删除，尽管它们仍然被当前的浏览器支持。首选的方法是使用 CSS 实现这种效果，第 7 章中将对此进行介绍。

1.6.5　<tt>元素

<tt>元素的内容以等宽字体显示。

```
The following word will appear in a <tt>monospaced</tt> font.
```

图 1-12 中给出了<b>、<i>、<u>、<s>和<tt>元素的使用效果(ch01_eg10.html)。

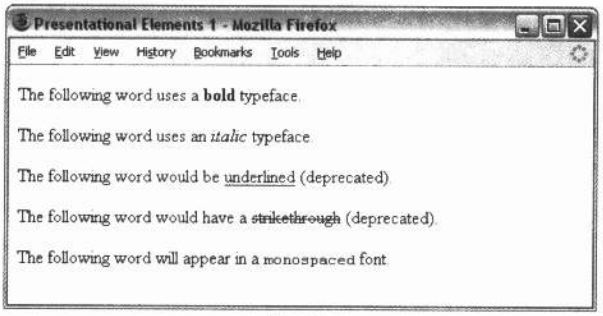


图 1-12

1.6.6　<sup>元素

<sup>元素的内容以上标的形式显示；该元素使用的字体大小与周围的字符相同，但是它所显示的字符高度只有其他字符高度的一半。

```
Written on the 31<sup>st</sup> February.
```

<sup>元素可用于在等式中添加指数值，或者在数值(例如日期)后面添加 st、nd、rd 和 th 后缀。然而，在某些浏览器中，<sup>元素将使上标文本所在的行与其上方的行之间产生较大间距。

1.6.7　<sub>元素

<sub>元素的内容以下标的形式显示；该元素使用的字体大小与周围的字符相同，但它所显示的字符高度只有其他字符高度的一半。

```
The EPR paradox<sub>2</sub> was devised by Einstein, Podolsky, and Rosen.
```

<sub>元素能够与<a>元素(下一章中将介绍该元素)结合使用以创建脚注。

1.6.8　<big>元素

<big>元素的内容在显示时比周围的文本大一个字体尺寸。但是如果字体已经是最大字

体，则该元素将无效。可以将多个<big>元素嵌套在一起使用，每一个元素中的内容将比其外围元素的内容大一个字体尺寸。

```
The following word should be <big>bigger</big> than those around it.
```

通常应当使用 CSS 而非<big>元素进行格式化。

1.6.9 <small>元素

<small>元素的内容在显示时比其周围的文本小一个字体尺寸。但是，如果字体已经是最小字体，则该元素将无效。可以将多个<small>元素嵌套在一起使用，每一个元素中的内容将比其外围元素的内容小一个字体尺寸。

```
The following word should be <small>smaller</small> than those around it.
```

通常应当使用 CSS 而非< small >元素进行格式化。

1.6.10 <hr />元素

<hr />元素用于在页面上创建一条水平线。它是一个空元素，非常类似于
元素：

```
<hr />
```

通常使用该元素将页面的不同部分(这些部分之间不适合使用新的题头)隔开。
图 1-13 给出了使用<sup>、<sub>、<big>、<small>和<hr />元素的效果(ch01_eg11.html)。

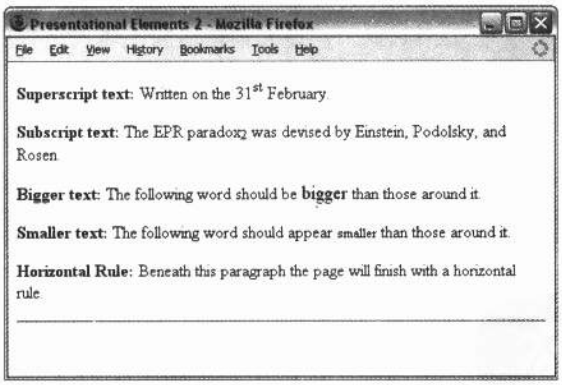


图 1-13

1.7 短语元素

以下元素的使用范围没有前面介绍的元素广泛。如同元素名称所指示，设计这些元素是为了描述它们的内容：

- 和元素用于强调

- <blockquote>、<cite>和<q>元素用于引用和引证
- <abbr>、<acronym>和<dfn>元素用于缩写词、首字母缩写词和关键术语
- <code>、<kbd>、<var>和<samp>元素用于计算机代码和信息
- <address>元素用于地址

虽然其中一些短语元素的显示方式与前面介绍的<b>、<i>、<pre>和<tt>元素相似，但设计它们是为了特殊的目的。例如，和元素分别用于强调和着重强调文本，并且存在多个用于标记引用的元素。

通常趋向于忽略这些元素，而仅使用前面小节中介绍的表现元素创建相同的可视效果，但是开发人员至少应当了解这些元素，并且最好习惯于在适当的时候使用它们。例如，当希望在一个句子中的某个单词上添加强调时，应当使用或元素而不是本章前面介绍的表现元素，原因包括以下几个方面：

- 一些应用程序(例如屏幕阅读器，用于向视力受损的用户阅读 Web 页面)能够在阅读声音中添加合适的音调，以便视力受损的用户可以听到强调所在的位置。
- 可以编写一些自动化的程序，用于找到具有强调的单词，并且将这样的单词作为文档中的关键字，或者专门索引这些单词，以便用户能够找到文档中的重要术语。

适当地使用这些元素能够在文档中添加更多的信息(例如哪些单词应当具有强调，哪些部分是程序代码，哪些部分是地址等)，而不仅是表明如何形象地表现文档。

下面介绍的所有短语元素能够附带本章前面给出的通用属性和 UI 事件属性。

1.7.1 元素添加强调

元素的内容是文档中的强调点，该内容通常以斜体显示。对一些单词使用这种强调方式，例如下面句子中的单词“must”：

```
<p>You <em>must</em> remember to close elements in XHTML.</p>
```

使用这个元素的原则是仅当需要对单词添加强调时才使用它，而不能是因为希望斜体显示文本而使用它。如果只是出于样式化原因而希望斜体显示文本，可以使用<i>元素或者 CSS。

1.7.2 元素添加着重强调

元素用于着重强调它的内容——强调程度强于元素。与元素一样，应当仅在希望对文档的一部分添加着重强调时才使用元素。大多数可视化浏览器没有以斜体显示着重强调的文本，而是以粗体显示该文本。

```
<p><em>Always</em> look at burning magnesium through protective colored
glass as it <strong>can cause blindness</strong>.</p>
```

图 1-14 给出了元素和元素在 Firefox 浏览器中的显示效果(ch01_eg12.html)。

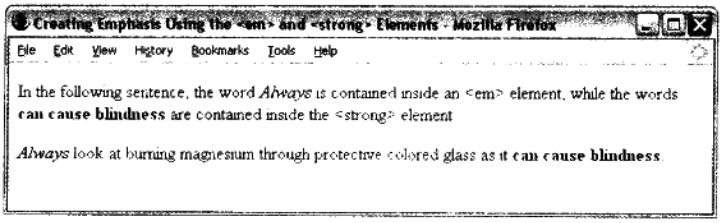


图 1-14

需要记住如何以在很大程度上无关的方式(粗体或斜体)展现这些元素。应该使用这些元素在短语上添加强调，从而使文档更有意义，而不是为了控制它们的可视化外观。在第 7 章中将会介绍，利用 CSS 改变元素的可视化表现非常简单——例如，使用黄色背景突出显示元素中的任何单词，并且以粗体(而非斜体)显示它们。

1.7.3 用于缩写词的<abbr>元素

通过将缩写词放置在起始标签<abbr>和结束标签</abbr>之间，可以指示何时使用缩写形式。

尽可能考虑使用 title 属性，将其值设置为缩写词的完全版本。如果缩写的是外语单词，也可以使用 XHTML 中的 xml:lang 属性(或者 HTML 中的 lang 属性)。

例如，如果希望将 Bev 指示为 Beverly 的缩写词，可以使用如下所示的<abbr>元素：

```
I have a friend called <abbr title="Beverly">Bev</abbr>.
```

1.7.4 用于首字母缩写词的<acronym>元素

利用<acronym>元素，可以指示起始标签<acronym>和结束标签</acronym>之间的文本是首字母缩写词。

尽可能使用 title 属性，并且将其值设置为<acronym>元素中首字母缩写词的完全版本。如果首字母缩写词采用一种不同的语言，可以在 XHTML 文档中包含 xml:lang 属性(或者在 HTML 文档中包含 lang 属性)。

例如，如果希望指示 XHTML 为一个首字母缩写词，可以使用如下所示的<acronym>元素(ch01_eg13.html)：

```
This chapter covers marking up text in <acronym title="Extensible Hypertext
Markup Language">XHTML</acronym>.
```

图 1-15 中给出了 Firefox 浏览器中<abbr>元素和<acronym>元素的显示效果(添加下划线的单词)，并且将鼠标悬停在这些单词上时，title 属性中的值将显示为工具提示。Internet Explorer 7 不改变这些元素的外观，但是它也将 title 属性的值显示为工具提示。

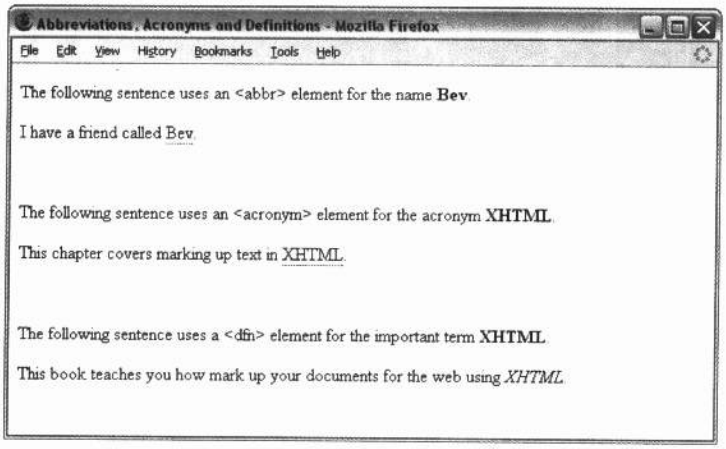


图 1-15

1.7.5　用于特殊术语的<dfn>元素

使用<dfn>元素可以指定正在引入特殊术语。

通常仅在第一次引入关键术语时使用<dfn>元素。当前的大多数浏览器以斜体字体显示<dfn>元素的内容。

例如，可以采用如下方式表明术语“XHTML”在句子中的重要性，并且应该标记：

```
This book teaches you how mark up your documents for the Web using
<dfn>XHTML</dfn>.
```

图 1-15 中给出了使用<dfn>元素之后的效果(ch01_eg13.html)。

1.7.6　用于引用文本的<blockquote>元素

如果希望引用另一个位置中的一段文本，应该使用<blockquote>元素。注意，当引用较短时，可以使用单独的<q>元素，下一节中将介绍该元素。下面是 ch01_eg14.html 中的示例：

```
<p>The following description of XHTML is taken from the W3C Web site:</p>
<blockquote> XHTML 1.0 is the W3C's first Recommendation for XHTML,
following on from earlier work on HTML 4.01, HTML 4.0, HTML 3.2 and HTML
2.0. </blockquote>
```

<blockquote>元素中的文本通常相对于周围的文本左右缩进显示，有时使用斜体字体显示(但是该元素应当仅用作为引用；如果只是希望斜体显示文本段落，则应该使用 CSS)。图 1-16 中给出了该元素的显示效果。

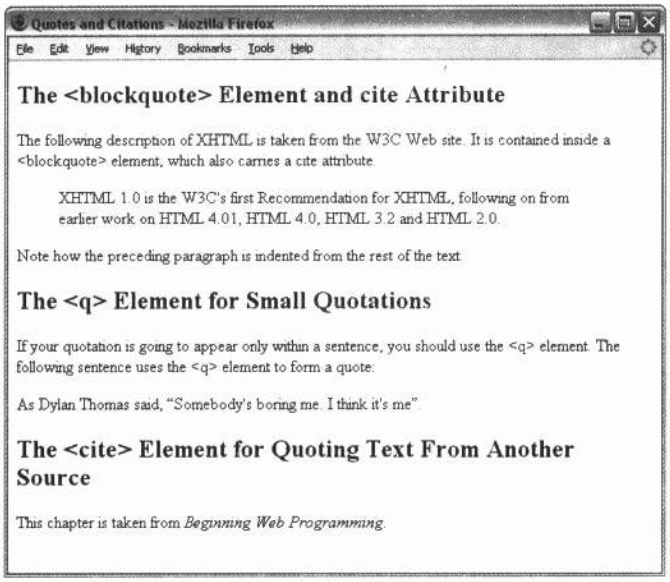


图 1-16

在<blockquote>元素中使用 cite 属性

可以在<blockquote>元素中使用 cite 属性以指明引用的来源，该属性的值应当是一个指向在线文档的 URL，并且尽可能给出文档的精确位置。实际上浏览器不对这个属性执行任何操作，但是它表示引用的来源在何处，将来可能需要用到它——该属性也可以被其他处理应用程序使用(ch01_eg14.html)。

```
<blockquote cite="http://www.w3.org/markup/"> XHTML 1.0 is the W3C's first
Recommendation for XHTML, following on from earlier work on HTML 4.01, HTML
4.0, HTML 3.2 and HTML 2.0.</blockquote>
```

注意：

在编写本书时，有些验证器不支持 cite 属性。例如 W3C 验证器，它不会识别在<blockquote>元素中出现的 cite 属性。

1.7.7 用于短引用的<q>元素

当希望在句子内添加引用而不是将引用作为独立的缩进块时，可以使用<q>元素(ch01_eg14.html)：

```
<p>As Dylan Thomas said, <q>Somebody's boring me. I think it's me</q>.</p>
```

HTML 规范和 XHTML 规范中都规定，<q>元素中的文本在显示时需要包含在双引号中。Firefox 浏览器会为用户插入这些引用标记，而 IE7 浏览器不会这样操作。因此，如果希望使用引用标记包括引用，则需要注意的是，在文档中插入它们将导致 Firefox 浏览器中显示两组引用。Firefox 浏览器和 IE 浏览器都不会以任何其他方式改变这种元素的外观。

<q>元素也能附带 cite 属性，该属性的值应该设置为指向引用来源的 URL。

1.7.8　用于引证的<cite>元素

当引用文本时，可以通过将其放置到起始标签<cite>和结束标签</cite>之间来指示来源。与打印的出版物一样，默认情况下<cite>元素的内容以斜体文本显示(ch01_eg12.html)。

```
This chapter is taken from <cite>Beginning Web Development</cite>.
```

如果引用的是在线资源，则应当将<cite>元素放置在一个<a>元素中以创建到相关文档的链接，第 2 章中将介绍<a>元素。

一些应用程序可以潜在地利用<cite>元素。例如，搜索应用程序可以使用<cite>标签找到参考某些著作的文档，浏览器能够通过收集<cite>元素的内容来生成给定文档的参考文档目录。但是，目前该元素还没有广泛用于已有的功能。

图 1-16 中给出了<blockquote>、<q>和<cite>元素的显示效果。

1.7.9　用于代码的<code>元素

如果页面中包含任何编程代码(在 Web 中并不常见)，则可以使用下面 4 种元素。出现在 Web 页面中的代码应当放置在<code>元素中。通常<code>元素中的内容以等宽字体显示，如同大多数编程书籍中的代码一样(包括本书)。

说明：

如果希望表示 XHTML 标记，不能仅在这些元素内使用起始尖括号和结束尖括号。浏览器会错误地将尖括号中的内容当作实际的标记。相反，使用<代替左尖括号<，使用>代替右尖括号>。附录 F 中给出了所有这些字符实体的列表。

下面的代码给出了在 HTML 中使用<code>元素表示<h1>元素和它的内容的示例(ch01_eg15.html)：

```
<p><code>&lt;h1&gt;This is a primary heading&lt;/h1&gt;</code></p>
```

图 1-17 中演示了这段代码在浏览器中的显示。

从理论上来说，使用<code>元素有助于搜索应用程序查看<code>元素中的内容，以便寻找特定的代码段。<code>元素通常与<pre>元素一起使用，以便保留代码的格式。

1.7.10　用于通过键盘输入的文本的<kbd>元素

当谈论计算机时，如果希望告诉读者输入一些文本，可以使用<kbd>元素表明应该输入的内容，下面是使用该元素的示例(ch01_eg15.html)：

```
<p>Type in the following: <kbd>This is the kbd element</kbd>.</p>
```

<kbd>元素中的内容通常以等宽字体显示，非常类似于<code>元素的内容。图 1-17 中

给出了该元素在浏览器中的显示。

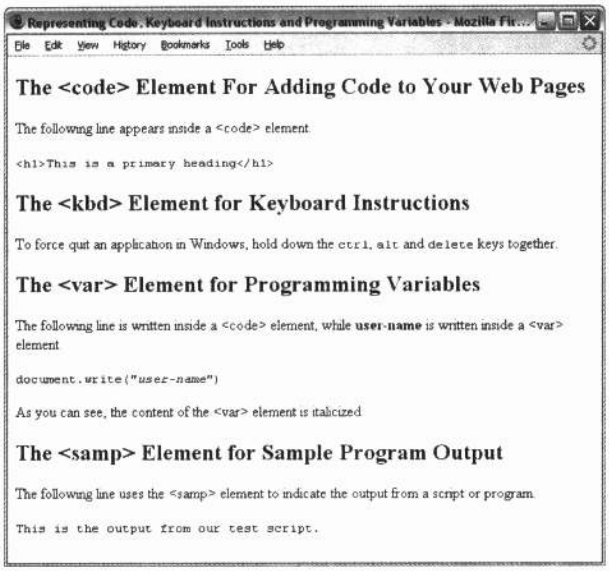


图 1-17

1.7.11 用于编程变量的<var>元素

<var>元素是对程序员非常有帮助的另一个元素。该元素通常与<pre>元素和<code>元素一起使用，以表明它的内容是可以由用户提供的一个变量(ch01_eg15.html)。

```
<p><code>document.write("<var>user-name</var>")</code></p>
```

<var>元素中的内容通常以斜体显示，如图 1-17 中所示。

如果不熟悉变量的概念，可以参考本书中的第 11 章。

1.7.12 用于程序输出的<samp>元素

<samp>元素用于指示程序、脚本等的样本输出，它主要用于为编程概念编制文档时。例如(ch01_eg15.html)：

```
<p>If everything worked you should see the result <samp>Test completed
OK</samp>.</p>
```

<samp>元素中的内容通常以等宽字体显示，如图 1-17 所示。

1.7.13 用于地址的<address>元素

许多文档中需要包含邮局寄送邮件的地址，并且存在一个用于包含地址的特殊<address>元素。例如，下面的<address>元素中包含了 Wrox 的地址(ch01_eg16.html)。

```
<address>Wrox Press, 10475 Crosspoint Blvd, Indianapolis, IN 46256</address>
```

浏览器中显示地址的方式与其周围的文档不同，IE、Firefox 和 Safari 浏览器以斜体显示地址，如图 1-18 所示(但是可以使用 CSS 重写该样式)。

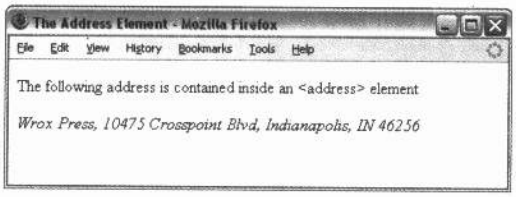


图 1-18

<address>元素通常位于文档的末尾，以表明谁编写了文档以及谁负责文档的可信性等。该元素也有助于自动化应用程序从文档中读取地址。

到此为止已经介绍了各种短语元素，但是还有很多与文本相关的元素需要介绍。

1.8　列表

在 Web 页面上添加列表存在多种原因，例如在主页上放置 5 本喜爱的相册，包含一组编号的指令供访问者按顺序执行(类似本书“试一试”部分中的步骤)。

在 XHTML 中，可以创建 3 种类型的列表：

- 无序列表，例如项目符号列表
- 有序列表，利用一系列数值或字母而非项目符号
- 定义列表，用于指定术语和它的定义

我可以确信您能够想到关于列表的更多用法并开始使用它们。

1.8.1　利用<ul>元素创建无序列表

如果希望在页面上放置项目符号列表，可以利用
元素(代表无序列表)编写该列表。希望编写的每个项目符号或每一行都必须包含在起始标签
和结束标签
之间(li 代表列表项(list item))。

元素必须始终以
标签结束。尽管有些 HTML 页面中可能省略结束标签，但这是一种不良的习惯，开发人员应当尽量避免。

如果希望创建项目符号列表，可以按照如下示例操作(ch01_eg17.html)：

```
<ul>
    <li>Bullet point number one</li>
    <li>Bullet point number two</li>
    <li>Bullet point number three</li>
</ul>
```

该列表在浏览器中的外观如图 1-19 所示。

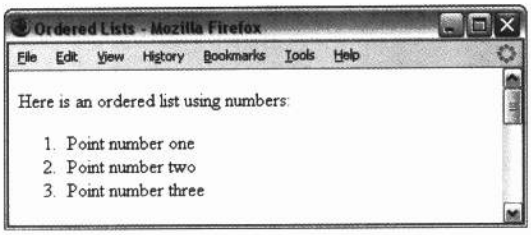


图 1-19

和<li>元素可以附带所有的通用属性和 UI 事件属性。

在 HTML 4.1 中，<ul>元素也可以附带 compact 属性。Transitional XHTML 1.0 也支持该属性，但是 Strict XHTML 1.0 不支持该属性。compact 属性能够使项目符号的垂直距离更近，它的值应该也是 compact，如下所示：

```
<ul compact="compact">
    <li>Item one</li>
    <li>Item two</li>
    <li>Item three</li>
</ul>
```

1.8.2 有序列表

有时人们希望列表是有序的。在有序列表中，列表中每一项的前缀不是项目符号，而是数值(1、2、3)、字母(A、B、C)或罗马数字(i 、 ii 、iii)。

利用
元素可以编写有序列表。有序列表中的每一项嵌入在
元素中，并且包含在起始标签
和标签结束
之间(ch01_eg18.html)。

```
<ol>
    <li>Point number one</li>
    <li>Point number two</li>
    <li>Point number three</li>
</ol>
```

结果如图 1-20 所示。

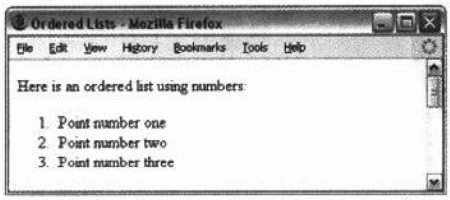


图 1-20

如果希望在列表项的前面使用字母或罗马数字，则需要在<ol>元素中使用现在逐渐淘汰的 type 属性。

1. 使用 type 属性(逐渐淘汰)选择有序列表中的数值、字母或罗马数字

在<ol>元素中利用 type 属性可以将列表项的排序方式由默认使用的数值改为表 1-4 中列出的选项，方式是将 type 属性的值设置为相应的字符。

表 1-4

type 属性的值	描　述	示　例
1	阿拉伯数字(默认值)	1、2、3、4、5
A	大写字母	A、B、C、D、E
a	小写字母	a、b、c、d、e
I	大写罗马数字	Ⅰ、Ⅱ、Ⅲ、Ⅳ、Ⅴ
i	小写罗马数字	ⅰ、ⅱ、ⅲ、ⅳ、ⅴ

例如，下面示例中的有序列表使用小写罗马数字(ch01_eg13.html)：

```
<ol type="i">
    <li>This is the first point</li>
    <li>This is the second point</li>
    <li>This is the third point</li>
</ol>
```

该示例的显示结果如图 1-21 所示。

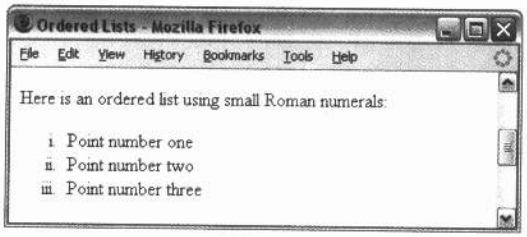


图 1-21

在 HTML 4.1 中，type 属性被标记为逐渐淘汰的类型，这是因为已经具有它的替代物——CSS 中的 list-style-type 属性；因此，type 属性仅能够用于 Transitional XHTML 中，而不能用于 Strict XHTML 1.0 中。另外需要注意，CSS 中的 list-style-type 属性只能工作于 IE 4 和 Netscape 4 以上版本的浏览器中。

可以在<li>元素中使用 type 属性，这样将重写<ol>元素中该属性的值，但在 HTML 4.1 中这样的用法已经逐渐淘汰，因此应当避免这样使用。所有的通用属性和 UI 事件属性都可用于<ol>元素中。另外，<ol>元素还能够附带 start 属性，该属性用于控制列表中的起始数值。

2. 使用 start 属性(逐渐淘汰)改变有序列表中的起始数值

如果希望指定编号列表的起始数值，可以在<ol>元素中使用 start 属性。start 属性的值

必须是列表中该项的数值表示，因此在使用大写字母排序的列表中，使用值“4”表示字母“D”(ch01_eg18.html)。

```
<ol type="i" start="4">
    <li>Point number one</li>
    <li>Point number two</li>
    <li>Point number three</li>
</ol>
```

这段代码的结果如图 1-22 所示。

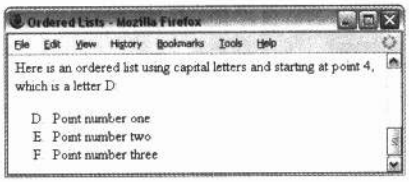


图 1-22

在 HTML 4.1 中，start 属性被标记为逐渐淘汰，因此仅能够用于 Transitional XHTML 1.0 中，而不能用于 Strict XHTML 1.0 中。

1.8.3 定义列表

定义列表是一种特殊类型的列表，这种列表用于提供术语，并且术语后面跟上简短的文本定义或描述。定义列表包含在<dl>元素中，<dl>元素中包含交替出现的<dt>和<dd>元素。<dt>元素的内容是即将定义的术语，<dd>元素中包含前面<dt>元素内容的定义。例如，下面是一个定义列表，它描述了 XHTML 中不同类型的列表(ch01_eg19.html)。

```
<dl>
  <dt>Unordered List</dt>
     <dd>A list of bullet points.</dd>
  <dt>Ordered List</dt>
     <dd>An ordered list of points, such as a numbered set of steps.</dd>
  <dt>Definition List</dt>
     <dd>A list of terms and definitions.</dd>
</dl>
```

这段代码在浏览器中的显示如图 1-23 所示(ch01_eg19.html)。

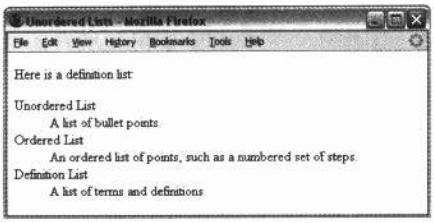


图 1-23

本节中介绍的 3 个元素都能够附带通用属性和 UI 事件属性。

1.8.4　列表的嵌套

列表中能够嵌套其他列表。例如，可能希望一个编号列表包含多个单独的列表，每个列表对应于编号列表中的一项。每个列表单独编号，除非使用 start 属性专门指定。每一个新的嵌套列表必须放置在<li>元素中(ch01_eg20.html)。

```
<ol type="I">
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
  <li>Item four
    <ol type="i">
      <li>Item 4.1</li>
      <li>Item 4.2</li>
      <li>Item 4.3</li>
    </ol>
  </li>
  <li>Item Five</li>
</ol>
```

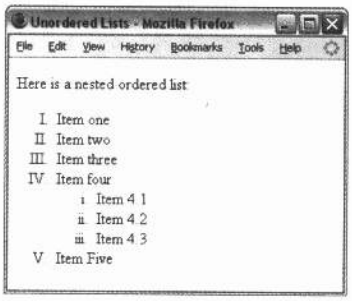


图 1-24

这段代码在浏览器中的显示如图 1-24 所示。

试一试　使用文本标记

现在已经介绍了一些可用于标记文本的不同元素和属性，接下来有必要练习使用它们。在这个示例中，使用多种文本标记创建显示食谱的页面。因此，打开文本编辑器或 Web 页面制作工具并完成以下步骤：

(1) 要求创建的文档遵循 Transitional XHTML 1.0，因此首先添加可选的 XML 声明和 DOCTYPE 声明：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

(2) 添加文档的程序框架元素：<html>、<head>、<title>和<body>元素。不要忘记在根元素处放置名称空间标识符以及指示文档语言的属性：

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
     <title>Wrox Recipes - World's Best Scrambled Eggs</title>
  </head>
  <body>
  </body>
</html>
```

(3) 在文档主体中添加一些适当的题头元素：

```
<body>
  <h1>Wrox Recipes - World's Best Scrambled Eggs</h1>
  <h2>Ingredients</h2>
  <h2>Instructions</h2>
</body>
```

(4) 在<h1>元素之后给出该食谱的解释(以及它是世界上最佳食谱的原因)。在下面的两个段落中使用了本章前面介绍的多个元素。

```
<h1>Wrox Recipes - World's Best Scrambled Eggs</h1>
  <p>I adapted this recipe from a book called
    <cite cite=" http://www.amazon.com/exec/obidos/tg/detail/-
    /0864119917/">Sydney Food</cite> by Bill Grainger. Ever since tasting
    these eggs on my 1<sup>st</sup> visit to Bill's restaurant in Kings
    Cross, Sydney, I have been after the recipe. I have since transformed
    it into what I really believe are the <em>best</em> scrambled eggs
    I have ever tasted.</p>
  <p>This recipe is what I call a <q>very special breakfast</q>; just look at
    the ingredients to see why. It has to be tasted to be believed.</p>
```

(5) 在第一个<h2>元素之后，以无序列表的方式列举食谱的配料。

```
<h2>Ingredients</h2>
  <p>The following ingredients make one serving:</p>
    <ul>
      <li>2 eggs</li>
      <li>1 tablespoon of butter (10g)</li>
      <li>1/3 cup of cream <i>(2 3/4 fl ounces)</i></li>
      <li>A pinch of salt</li>
      <li>Freshly milled black pepper</li>
      <li>3 fresh chives (chopped)</li>
    </ul>
```

(6) 在第二个<h2>元素之后添加一些指令；这些指令将以编号列表的形式显示：

```
<h2>Instructions</h2>
    <ol>
      <li>Whisk eggs, cream, and salt in a bowl.</li>
      <li>Melt the butter in a non-stick pan over a high heat <i>(taking care
      not to burn the butter)</i>.</li>
      <li>Pour egg mixture into pan and wait until it starts setting around
          the edge of the pan (around 20 seconds).</li>
      <li>Using a wooden spatula, bring the mixture into the center as if it
          were an omelet, and let it cook for another 20 seconds.</li>
      <li>Fold contents in again, leave for 20 seconds, and repeat until
          the eggs are only just done.</li>
      <li>Grind a light sprinkling of freshly milled pepper over the eggs
          and blend in some chopped fresh chives.</li>
    </ol>
  <p>You should only make a <strong>maximum</strong> of two servings per
    frying pan.</p>
```

(7) 将这个示例保存为 eggs.html。在浏览器中打开它时，可以看到如图 1-25 所示的页面。

图 1-25

工作原理

对于 XML 声明和文档的程序框架，前面已经多次出现，此处不再详细讨论。现在主要关注用于标记文本的一些新元素。

在文档的主题头(包含在<hl>元素中)之后是两个文本段落。首先查看第一个段落。

在第一个句子中，使用<cite>元素指示一个参考书籍，页面中介绍的食谱来源于此书籍。下一个句子中使用了<sup>元素以写出“1st”，并且使用了上标文本——这使得第一行和第二行之间的文本距离大于第二行和第三行之间的文本距离(因为上标字母伸出该行上方)。在最后一个句子中，对单词“best”进行了强调，因为这是我曾经尝过的最美味的炒鸡蛋。

```
<h1>Wrox Recipes- World's Best Scrambled Eggs</h1>
  <p>I adapted this recipe from a book called
    <cite cite="http://www.bills.com.au">Sydney Food</cite> by Bill Grainger.
    Ever since tasting these eggs on my 1<sup>st</sup> visit to Bill's
    restaurant in Kings Cross, Sydney, I have been after the recipe. I have
    since transformed it into what I really believe are the <em>best</em>
```

```
    scrambled eggs I have ever tasted. </p>
```

在第二个元素中有另一个新元素：用于表示句子中的短引用的<q>元素。

```
<p>Although this recipe may be what I call a <q>very special breakfast</q>,
  just look at the ingredients to see why, it has to be tasted to be
  believed.</p>
```

食谱的配料(列在<h2>元素下方)以无序列表的形式显示，其中以斜体显示必须准备的奶油量的备选度量：

```
<ul>
  <li>2 eggs</li>
  <li>10g butter</li>
  <li>1/3 cup of cream <i>(2 3/4 fl ounces)</i></li>
  <li>a pinch of salt</li>
  <li>freshly milled black pepper</li>
  <li>3 fresh chives (chopped)</li>
</ul>
```

煎鸡蛋的指令(列在第二个<h2>元素下方)包含在一个编号列表和两个额外的段落中。您可能已经注意到，编号列表中包含一个斜体的注释，该注释提醒不要烧糊黄油；最后一个段落中包含一个着重强调，提醒不应该在一个平底锅中煎蛋两次以上。

```
<h2>Instructions</h2>
  <p>The following ingredients make one serving.</p>
    <ol>
      <li>Whisk eggs, cream, and salt in a bowl.</li>
      <li>Melt the butter in a non-stick pan over a high heat <i>(taking care
          not to burn the butter)</i>.</li>
      <li>Pour egg mixture into pan, and wait until it starts setting
          around the edge of the pan (around twenty seconds).</li>
      <li>Using a wooden spatula, bring the mixture into the center as
          if it was an omelet, and let it cook for another 20 seconds.</li>
      <li>Fold contents in again, leave for 20 seconds, and repeat until
          the eggs are only just done.</li>
      <li>Grind a light sprinkling of freshly milled pepper over the eggs
          and blend in some chopped fresh chives.</li>
    </ol>
  <p>You should only make a <strong>maximum</strong> of two servings per
    frying pan.</p>
```

与往常一样，该 Web 页面以结束标签</body>和结束标签</html>作为结尾。希望您能够喜欢这些煎鸡蛋——现在请您亲自练习这个示例。

1.9 编辑文本

当与其他人一起编辑文档时，如果能够看到其他人执行的改动，则会非常有帮助。即

使是处理自己的文档，跟踪自己执行的改动也会非常有帮助。为此特别设计了两个元素，分别用于修订和编辑文本：

- <ins>元素用于添加文本
- <del>元素用于删除一些文本

从下面的代码中可以看到对 XHTML 文档执行的改动(ch01_eg21.html)：

```
<h1>How to Spot a Wrox Book</h1>
<p>Wrox-spotting is a popular pastime in bookshops. Programmers like to find
the distinctive <del>blue</del><ins>red</ins> spines because they know that
Wrox books are written by <del>1000 monkeys</del><ins>Programmers</ins> for
Programmers.</p>
<ins><p>Both readers and authors, however, have reservations about the use
of photos on the covers.</p></ins>
```

该示例在浏览器中的显示如图 1-26 所示。

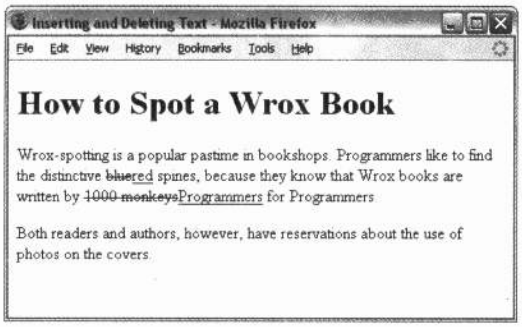


图 1-26

在某些编辑工具中，这两种功能也非常有用，可以注释不同的作者执行的改动和修改。

注意：

如果熟悉微软的 Word 软件，则可以知道<ins>元素和<del>元素与该软件中的“修订”功能非常相似(可以在“工具”菜单下找到该功能)。“修订”功能以下划线的形式显示新添加的文本，并在删除的文本上绘制删除线。

在使用<ins>元素和<del>元素时必须非常小心，确保不以位于内联元素(例如<b>元素和<i>元素)中的块级元素(例如<p>元素和<h2>元素)作为结尾。本章的末尾部分中将详细介绍块级元素和内联元素。

1.9.1　使用<ins>元素指示新添加的文本

<ins>元素中新添加到文档的文本将以下划线的形式显示，以表明它是新的文本(如图 1-26 所示)。

```
    <ins><p>This paragraph is contained inside an &lt;ins&gt;
element.</p></ins>
```

可以在<ins>元素和<del>元素中使用 cite 属性，以指示改动的来源或原因，但是 cite 属性具有严格的限制(它的值必须是 URL)。

也可以使用 title 属性以表明谁添加了<ins>元素或<del>元素以及添加或删除文本的原因；在大多数浏览器中，title 属性中的信息将作为工具提示提供给用户。

<ins>元素和<del>元素也可以附带 datetime 属性，datetime 属性的值是日期和时间，格式如下所示：

YYYY-MM-DDThh:mm:ssTZD

该格式的详细解释如下：

- YYYY 表示年
- MM 表示月
- DD 表示该月中的日期
- T 是日期和时间之间的分隔符
- hh 是小时
- mm 是分钟数
- ss 是秒数
- TZD 是时区标志符

例如，2004-04-16T20:30-05:00 表示美国东部时间 2004 年 4 月 16 日 20 点 30 分

注意：

通常 datetime 属性的值仅由程序或制作工具输入，因为它的格式比较长，无法手动输入。

1.9.2 使用<del>元素指示删除的文本

如果希望删除文档中的一些文本，则可以将它们放置在<del>元素中，以表明它们被标记为删除。显示<del>元素中的文本时，它们的上方将出现一条删除线(如图 1-26 所示)。

```
<del><p>This paragraph is contained inside a &lt;del&gt; element.</p></del>
```

与<ins>元素一样，<del>元素能够附带 cite、datetime 和 title 属性。

注意：

学习如何使用 CSS 之后可以了解到：通过使用 CSS，能够根据需要显示和隐藏插入的和删除的内容。

1.10 利用字符实体表示特殊字符

在文档中可以使用大多数字母数字字符，并且浏览器能够正确显示它们。但是，在 XHTML 中有些字符具有特殊意义，并且某些字符无法使用键盘直接输入。例如，当希望在浏览器中显示尖括号时，不能直接在文档中输入尖括号，否则浏览器会将其后面的字母错误地当作为标记。但是，可以使用一组不同的字符来表示这些特殊字符，这组字符称为字符实体。有时字符实体也称为转义字符。

所有的特殊字符都具有相应的数字实体，可以使用这些数字实体将它们添加到文档中。另外，某些特殊字符还具有命名实体，如表 1-5 所示。

表 1-5

字　符	数字实体	命名实体
"	&#034;	&quot;
&	&#038;	&amp;
<	&#060;	&lt;
>	&#062;	&gt;

附录 F 中给出了所有字符实体(或特殊字符)的列表。

1.11　注释

可以将注释放置在 XHTML 文档中的任何标签之间。注释的语法如下所示：

```
<!-- comment goes here -->
```

“<!--”和“-->”之间的任何内容都不会显示在浏览器中，但是可以在文档的源代码中看到它们。

经常注释代码是一种良好的实践，特别是在复杂的文档中，可以向查看代码的任何人指示文档的各个部分以及其他任何注意事项。注释也帮助您和其他人理解您的代码。

有时甚至可以注释整段代码。例如，对于下面的代码片段，在浏览器中不会看到<h2>元素中的内容。也有些注释用于指示文档的特定部分、谁添加了该部分以及什么时候添加。

```
<!-- Start of Footnotes Section added 04-24-04 by Bob Stewart -->
  <!-- <h2>Character Entities</h2> -->
  <p><strong>Character entities</strong> can be used to escape special
  characters that the browser might otherwise think have special meaning.</p>
<!-- End of Footnotes section -->
```

1.12　<font>元素(逐渐淘汰)

HTML 3.2 中引入了<font>元素，用户可以使用它更多地控制文本的外观。在 HTML 4.0 中该元素被标记为逐渐淘汰，并且已经从 XHTML 中移除。但是在该元素短暂的一生中，它获得了大量的使用，如果查看其他人编写的代码，可以看到在很多位置都使用了该元素。附录 I 中给出了关于<font>元素的更多内容。下面的示例中给出了<font>元素的用法：

```
<h3>Using the &lt;font&gt; element</h3>
<font face="arial, verdana, sans-serif" size="2" color="#666666">The
&lt;font&gt; element has been deprecated since HTML 4.0. You should now use
CSS to indicate how text should be styled. </font>
```

1.13 理解块级元素和内联元素

到目前为止已经介绍了很多可用于标记文本的元素。现在有必要集中查看位于<body>元素中的所有元素，它们可以划分为以下两类元素：

- 块级元素
- 内联元素

这种划分只是在概念上具有不同，但是对于 XHTML 的其他功能(本书后面将介绍其中一些功能)来说，这种划分具有重要影响。

在屏幕上显示时，块级元素的前面和后面都将换行，如同存在回车或换行符一样。例如，<p>、<h1>、<h2>、<h3>、<h4>、<h5>、<h6>、<ul>、<ol>、<dl>、<pre>、<hr />、<blockquote>和<address>都是块级元素。在显示这些元素时都将从新行中开始显示，它们后面的内容也将在新行中显示。

另一方面，内联元素一般出现在句子中，在浏览器中显示时不会换行。以下元素都是内联元素：<b>、<i>、<u>、、、<sup>、<sub>、<big>、<small>、<li>、<ins>、<del>、<code>、<cite>、<dfn>、<kbd>和<var>。

例如，请查看下面的题头和段落。这些元素在浏览器中显示时从新行开始显示，并且它们后面的文本也从新行开始显示。同时，段落中的内联元素不会显示在新行中。代码如下所示(ch01_eg22.html)：

```
<h1>Block-Level Elements</h1>
  <p><strong>Block-level elements</strong> always start on a new line. The
  <code>&lt;h1&gt;</code> and <code>&lt;p&gt;</code> elements will not sit
  on the same line, whereas the inline elements flow with the rest of the
  text.</p>
```

该代码在浏览器中的显示如图 1-27 所示。

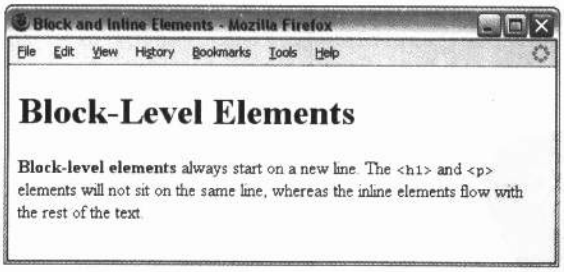


图 1-27

在 Strict XHTML 中，块级元素能够包含其他块级元素和内联元素。但是，内联元素只能出现在块级元素中，它们不能包含块级元素(因此不能将<b>元素放置在块级元素外部)。

1.14　利用 元素和元素分组元素

利用 元素或元素可以分组多个元素，以创建页面的某些部分或者子部分。这些元素自身不会影响页面的外观，但是它们通常与 CSS 一起使用，以允许用户添加样式到页面的某个部分(详情参考第 7 章)。例如，可以将页面中的所有脚注放置到一个 元素中，以表明 元素中的所有元素与该脚注相关。然后可以附加一个样式到这个 元素，从而使用一组特殊的样式规则来显示脚法。

如下代码演示了用于分组块级元素的 元素：

```
<div class="footnotes">
  <h2>Footnotes</h2>
  <p><b>1</b> The World Wide Web was invented by Tim Berners-Lee</p>
  <p><b>2</b> The W3C is the World Wide Web Consortium who maintain many Web
     standards</p>
</div>
```

另一方面，可以使用元素仅分组内联元素。因此，如果句子或段落的某个部分需要分组，可以使用元素。下面的代码中添加了一个元素，以表明哪些内容与某位发明家相关。该元素中包含了一个粗体元素和一些文本：

```
<div class="footnotes">
  <h2>Footnotes</h2>
  <p><span class="inventor"><b>1</b> The World Wide Web was invented by Tim
    Berners Lee</span></p>
  <p><b>2</b> The W3C is the World Wide Web Consortium who maintain many Web
     standards</p>
</div>
```

该元素自身对文档在浏览器中的显示外观没有任何影响，但是它对标记添加了额外的意义，即将相关元素分组在一起。这种分组也可以供其他处理应用程序使用，或者用于使用 CSS 规则将特殊的样式赋给组中的元素(参见第 7 章)。

元素和元素可以附带所有的通用属性和 UI 事件属性，甚至是逐渐淘汰的 align 属性(Strict XHTML 1.0 中已经移除该属性)。

1.15　本章小结

本章主要介绍如何使用 XHTML 结构化文档中的文本。

本章介绍了如何使用描述文档的结构元素标记 Web 页面中的内容。这些元素由一个起始标签、一个结束标签以及这些标签之间的内容组成。为了修改元素的某些特性，可以在起始标签中附带属性，属性由名称和值组成。XHTML 可以看作为 HTML 的最新版本，并且存在 3 种不同类型的 XHTML——为了告知浏览器使用的是哪一种类型，可以使用 DOCTYPE 声明。

本章还介绍了许多新元素以及它们能够附带的属性。每个 XHTML 文档必须至少包含

<html>、<head>、<title>和<body>元素，并且<html>元素应当附带一个名称空间标识符。

然后本章介绍了一些属性：核心属性(class、id 和 title)、国际化属性(dir、lang 和 xml:lang)和 UI 事件属性，本书中经常会使用这些属性，因为大多数元素都支持它们。

最后，本章讨论了一些用于描述文本结构的元素：

- 6 个级别的题头元素：<h1>、<h2>、<h3>、<h4>、<h5>和<h6>
- 段落元素<p>、预先格式化某些部分的元素<pre>、换行元素
和地址元素<address>
- 表现元素，例如<b>、<i>、<u>、<s>、<tt>、<sup>、<sub>、<strike>、<big>、<small>和<hr />
- 短语元素，例如、、<abbr>、<acronym>、<dfn>、<blockquote>、<q>、<cite>、<code>、<kbd>、<var>、<samp>和<address>
- 列表元素，例如使用<ul>元素和<li>元素创建无序列表，使用<ol>元素和<li>元素创建有序列表，使用<dl>、<dt>和<dd>元素创建定义列表。
- 编辑元素，例如<ins>和<del>
- 分组元素，例如 和

其中对某些元素的使用多于其他元素，但是如果某个元素适用于尝试标记的内容(从段落到地址)，则应该使用它。适当地结构化文本有助于其比仅使用换行和表现元素格式化更持久地存在。

从第 2 章开始，本书给出的示例中将会多次使用这些元素，在这一章中将介绍如下重要主题：在文档之间链接(以及链接到文档的特定部分)。

最后需要提醒的是，为了更好地学习，可以查看其他人编写的 Web 页面。可以观察 Web 上的 HTML 或 XHTML 页面，方式是使用浏览器上的“查看”菜单或“工具”菜单并选择“源代码”选项(有时是“查看源代码”或“页面源代码”选项)(采用这种方式可能会学习到很多不良的习惯——因此仍然需要继续阅读本书，以避免这些不良的习惯)。

1.16 练习

所有练习的答案都在附录 A 中给出。

1. 利用相关的表示元素标记下面的语句。

```
The 1st time the bold man wrote in italics, he underlined several key words.
```

2. 标记下面的列表，其中具有插入和删除的内容：

- 1 ~~1/2~~ 3/4 cups ricotta
- 3/4 cup milk
- 4 eggs
- 1 cup plain white flour
- 1 teaspoon baking powder
- ~~75g~~ 50g butter
- pinch of salt

链接和导航

Web 与其他媒体的实质区别是 Web 页面可以包含了链接，利用链接可以直接转向其他页面(甚至给定页面的特定部分)。链接通常称为超链接，Web 的非凡成就与其息息相关。通过超链接，访问者可以在各个 Web 站点之间导航，方式是单击页面上的单词、短语和图像等。

普通的 Web 站点由一组页面组成，用户通过超文本链接在各个页面之间导航。这些页面通常包含指向其他 Web 站点的链接，以及指向所在站点的其他页面的链接。

本章将介绍如何创建站点中不同页面之间的链接，如何创建指向站点页面中特定位置的链接，以及如何创建指向其他站点(称为外部站点)的链接。

如同第 1 章中介绍的标记描述了文档的结构一样，链接描述了文档的哪些部分可以链接到其他文档的相应部分。因此，链接形成了不同文档之间的相互关系。

在学习链接的时候，了解一些关键概念也是非常重要的，包括如何将站点结构化到不同的文件夹中(文件夹也称为目录)，如何使用相对 URL 链接站点内的页面。

在本章中将介绍以下内容：

- 如何构造文件夹用于存放 Web 站点的页面
- 如何链接站点内的不同页面
- 如何链接到站点中页面的特定部分
- 如何链接到其他站点

本章仅介绍如何链接 Web 页面，而不会介绍链接或嵌入其他文件机制，特别是<link>元素(第 7 章中讨论样式表时将介绍该元素)或<img>元素和<object>元素(第 3 章中将介绍这些元素)。

2.1 基本链接

为了介绍链接，首先查看几个简单的例子。一旦了解链接的基本概念，您将发现仍有许多知识需要学习。下面将逐步介绍这些知识。

利用<a>元素可以指定链接。在起始标签<a>和结束标签之间的文本组成链接的内容，用户可以在浏览器中单击它们。下面部分将介绍如何链接到其他文档和电子邮件地址。

2.1.1 链接到其他文档

为了链接到其他文档，起始标签<a>需要附带属性 href，href 属性的值是链接页面的地址。

下面的示例是 ch02_eg01.html 文档的<body>元素。该页面包含指向第二个页面 index.html 的链接：

```
<body>
   Return to the <a href="index.html">index page</a>.
</body>
```

只要index.html文件与ch02_eg01.html文件处于相同的文件夹中，则在单击“index page”时，就会在相同的窗口中加载index.html页面，并且替换当前的ch02_eg01.html页面。如图2-1所示，<a>元素的内容形成了链接。

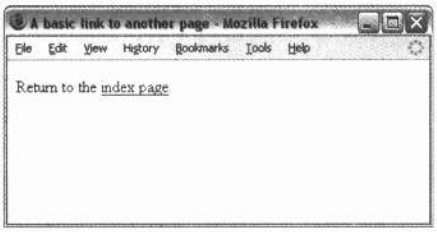


图 2-1

注意：

这就是本章下载代码中的链接的工作方式。记住，可以单击浏览器中的 View 菜单，然后选择 View Source 选项来查看 HTML 或 XHTML 页面的源代码。

虽然可以将各种类型的元素放置在<a>元素中，但最好尽量使链接简明，并且使<a>元素中的内容实际地描述链接的目的页面。由于<a>元素中的内容比其周围的文本醒目(通常以不同的颜色显示)，所以许多人在需要进入下一个页面时会简单浏览页面以获得相应的链接，而不会完整地阅读当前页面。因此，如果页面中所有链接的内容只是“单击此处(click here)”，则用户很可能不会长久地停留在该站点中，因为这些链接无法清楚并快速地告知他们目的页面的信息。

注意：

许多 Web 设计人员也在<a>元素中使用图像(下一章中将介绍相关内容)，但是在使用图像时，必须确保图像能够清楚地指示链接的目的页面。

如果希望链接到不同的站点，则可以使用下面的语法。其中指定了链接的目的页面的完整 URL(Uniform Resource Locator，统一资源定位符)，而不仅仅是文件名 cho2_ego2.html(本章后面将详细介绍 URL)：

```
<body>
   Why not visit the <a href="http://www.wrox.com/">Wrox Web site</a>?
</body>
```

这个链接指向 Wrox 的 Web 站点。href 属性的值与用户访问 Wrox 的 Web 站点时在浏览器中输入的内容相同。这种 URL 称为限定 URL，因为它包含了该 Web 站点的域名。

当希望链接到相同 Web 站点中的某个页面时，可以使用称为“相对 URL”的简写形式；本章的第一个示例使用的就是这种 URL，其中 href 属性的值不是以域名起始的。这样不仅不需要输入完整的 URL，而且具有其他优点。

在链接中使用 title 属性也是良好的编程习惯，因为在大多数浏览器中，当用户将光标悬停在链接上时，使用 title 属性将显示一个工具提示(在屏幕上显示说明标题的气泡状图标)。另外，对于语音浏览器——通常由视力受损人员使用——将会大声读出标题)。

title 属性的值应当能够描述链接的目的页面。当使用图像作为链接时，这一点尤其重要。例如，下面给出了到 Google 主页的链接(ch02_eg03.html)：

```
<a href="http://www.Google.com/" title="Search the Web with
Google">Google</a>is a very popular search engine.
```

图 2-2 显示了 title 属性，当用户将光标悬停在链接上时，该属性给出了链接的更多相关信息。

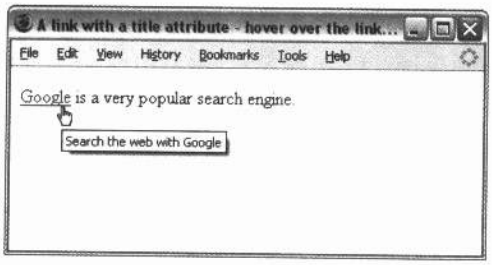


图 2-2

需要注意的是，<a>元素中的所有内容都将显示为链接，包括文本或图像周围的空格，因此最好避免直接在起始标签<a>和结束标签之间使用空格。例如，下面的链接中具有起始和结尾空格(ch02_eg03.html)：

```
Why not visit the<a href="http://www.wrox.com/"> Wrox Web site </a>?
```

如图 2-3 所示，链接中的空格也将带有下划线。

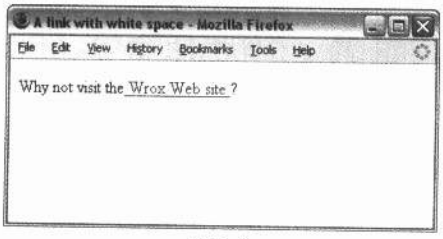


图 2-3

因此，最好在起始标签<a>和结束标签的外部使用空格，如下所示：

```
Why not visit the<a href="http://www.wrox.com/">Wrox Web site</a>?
```

当然，在<a>元素内的单词之间仍然需要使用空格，只要不在链接的开头和末尾使用它们即可。

2.1.2 链接到 e-mail 地址

您很可能已经看到，许多站点中的链接显示 e-mail 地址。当单击这样的链接时，将在默认的 e-mail 程序中打开一封新的电子邮件，并且准备好发送电子邮件到链接指向的地址。

为了创建指向 e-mail 地址的链接，需要使用<a>元素的如下形式的语法：

```
<a href="mailto:name@example.com">name@example.com</a>
```

其中，href 属性的值以关键字 mailto 开头，后面紧跟一个冒号，然后是 e-mail 地址。与其他链接一样，<a>元素中的内容将是链接在浏览器中的可见部分，因此可以选择使用如下语句：

```
<a href="mailto:name@example.com">E-mail us</a>.
```

或者，如果想让用户在单击 e-mail 地址之前看到该地址，可以采用如下语句：

```
For sales enquiries e-mail <a href="mailto:name@example.com">
sales@example .com</a>.
```

但是，使用这种技术也存在如下的缺陷：一些恶意的 Web 用户会使用小程序自动搜索 Web 站点中的 e-mail 地址。搜索到 e-mail 地址之后，他们就会向这些地址发送垃圾邮件。

在创建到 e-mail 地址的链接时，还有其他一些主要的备选方法：

- 使用 e-mail 表单，从而访问者在 Web 站点上填写该表单以发送 e-mail。接收到邮件后，可以正常回复它，因为自动程序无法根据联系人表单获得 e-mail 地址。e-mail 表单的应用需要 CGI 脚本或者服务器端的脚本语言，例如 ASP.net、JSP、PHP、Cold Fusion 和 Ruby。第 5 章中将给出关于 e-mail 表单的示例。
- 利用 JavaScript 将 e-mail 地址写进 Web 页面(第 12 章将介绍这方面的内容)。这种技术背后的主要思想是，在 Web 上搜索 e-mail 地址的程序无法读取地址的 JavaScript 版本。

前面已经介绍如何创建最基本的链接，下面将研究关于链接的更具深度的主题。为了深入了解页面之间的链接，您需要阅读后面的几页内容，其中将详细解释如何将 Web 站点中的文件组织到文件夹中，并且详细地分析 URL(URL 是一种地址，用于标识 Web 站点上的页面和其他资源)。

2.2 理解目录和目录结构

目录是 Web 站点中文件夹的名称(如同硬盘中包含不同的文件夹一样，Web 站点包含多个目录)。通常一个 Web 站点将包含多个目录，其中每个目录包含 Web 站点的不同部分。例如，一个具有多个分部的大型 Web 站点将为每个分部准备一个独立的目录，并且不同类型的文件(例如图像和样式表)通常保存在各自的特定目录中。

如同可能将硬盘中的文件组织到单独的文件夹中一样，将 Web 站点中的文件组织到目录中非常重要，这样就能够更加方便地找到需要的文件，并且能够控制所有文件。可以设想一下，如果将 Web 站点中使用的所有文件放置到同一个文件夹中，对这些文件的操作将会很快变得非常复杂。

图 2-4 中给出了一个新站点的目录结构的示例，站点的每一部分具有一个独立的文件夹，而不同类型的文件也具有独立的文件夹。在主文件夹中有专门用于存放图像、脚本和样式表的文件夹。另外注意，Music 部分分别为 Features、MP3 和 Reviews 准备了单独的文件夹。

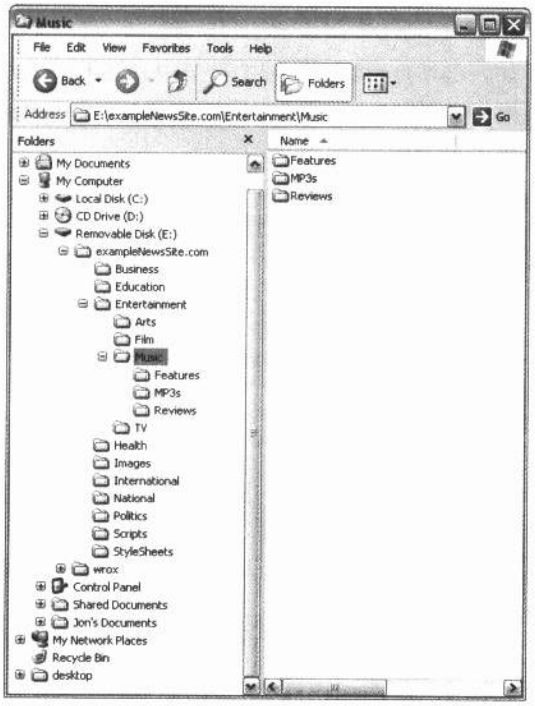


图 2-4

另外，类似于图 2-2 所示的目录结构能够帮助用户导航站点，而不用知道确切的文件名；并且用户可以选取他们想要的部分进行浏览，例如从 http://www.ExampleNewsSite.com/Business/中获得商业新闻，从 http://www.ExampleNewsSite.com/Entertainment/Music/中获得音乐页面。

使 Web 站点具有良好的组织结构是非常重要的。一个小型的 Web 站点可能会以令人惊讶的速度快速成长，并且包含的文件远远超出最初设想的数量。因此，在开始构建任何类型的 Web 站点时，都应当首先创建能够适应这种快速成长的优秀目录结构。

在学习链接相关内容的过程中了解一些用于描述目录结构和目录间关系的术语是非常重要的，因此重新查看图 2-4 中的目录结构示例：

- 保存整个 Web 站点的主目录称为 Web 站点的根目录；在本示例中，根目录是 exampleNewSite.com。
- 位于其他目录中的目录称为子目录。这里，Film 是 Entertainment 的子目录。
- 包含其他目录的目录称为父目录。这里，Entertainment 是 Arts、Film、Music 和 TV 的父目录。

2.2.1 链接的目标位置

在本章开始部分的示例中，创建了到同一个目录中不同页面的链接和到不同 Web 站点中页面的链接。

现在您已经了解了如何将站点组织到单独的目录中，接下来需要学习如何链接到当前站点不同文件夹中的页面——例如，如何从主页面链接到 Entertainment 部分中的一个页面，该部分位于 Entertainment 文件夹中。

可以在站点的每一个链接中使用完整的 URL(完整的 URL 也称为绝对 URL，在 Web 浏览器的地址栏中输入的就是这种 URL)，尽管最好使用相对 URL(相对 URL 是一种简写形式，用于链接站点不同文件夹中的文件)。相对 URL 指定某个文件相对于当前文件的位置。

在了解如何创建相对 URL 以及相对 URL 与绝对 URL 的区别之前，您需要掌握 URL 的结构。

2.2.2 URL 的组成

URL 由几个部分组成，每一部分向 Web 浏览器提供的信息有助于用户找到想要的页面。首先通过查看最常见的 URL 来帮助您理解 URL 的各个部分。图 2-5 中所示的 URL 有 3 个关键部分：模式、主机地址和文件路径，下面的小节中将依次讨论它们。

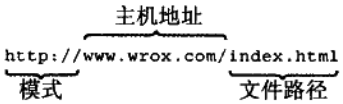


图 2-5

1. 模式

模式标识链接到的 URL 的类型，因而也标识了检索资源的方式。例如，大多数 Web 页面使用超文本传输协议(Hypertext Transfer Protocol，HTTP)传递信息，这就是大多数 Web 页面以 http://开头的原因，但是在使用网上银行或者下载大型文件时，您可能也注意到了其他前缀。

表 2-1 列出了最常见的模式。

表 2-1

模　式	描　述
http://	超文本传输协议(HTTP)用于向 Web 服务器请求页面，并将它们从 Web 服务器发送回浏览器
https://	安全超文本传输协议(Secure Hypertext Transfer Protocol，HTTPS)使用数字证书加密在浏览器和 Web 服务器之间发送的数据
ftp://	文件传输协议是在 Web 上传输文件的另一种方法。HTTP 由于与浏览器集成而经常用于浏览 Web 站点，而 FTP 经常用于在 Web 中传输大型文件以及向 Web 服务器上传源文件
file://	用于指明文件位于本地硬盘中或者局域网上的一个共享目录中

2. 主机地址

主机地址是 Web 站点的地址，利用它可以找到 Web 站点。主机地址的形式可以是 IP 地址(4 组 0 到 255 之间的数字，例如 192.0.110.255)，也可以是站点的域名(更为常见的形式)，例如 www.wrox.com。

注意：

可以利用 IP 地址找到任何连接到 Internet 的计算机；但是，域名比 IP 地址更容易记住，因此更为常用。但是，实际上所有的域名在后台都将被转换为保存 Web 站点的计算机的 IP 地址，转换的方法是咨询域名服务器(Domain Name Server，DNS)，DNS 中包含了运行 Web 站点的计算机的域名和 IP 地址的目录。

注意，www 实际上不是域名的组成部分，尽管它通常用于主机地址中——它与所使用的 HTTP 协议无关。

3. 文件路径

文件路径始终以正斜杠(/)开始，并且可能包含一个或多个目录名(目录是Web服务器上的文件夹的别名)；每个目录名以正斜杠字符(/)隔开，并且文件路径可能以文件名结束。下面的Overview.html就是一个文件名：

```
/books/newReleases/BeginningWebDevelopment/Overview.html
```

如果文件路径中不包含文件名，则 Web 服务器通常执行以下 3 种操作中的一种(取决于它的配置)：

- 返回一个默认文件(对于以 HTML 编写的 Web 站点，默认文件是 index.html 或 default.html)
- 提供所在目录中的文件列表
- 显示一条消息，表明无法找到所访问的页面或者无法浏览文件夹中的文件

4. URL 的其他部分

URL 还可以包含许多不常见的其他部分。

证书用于为站点中的密码保护部分指定用户名和密码。证书位于主机地址的前面，通过@符号与主机地址隔开。注意，用户名和密码以冒号隔开。下面的URL中显示了用户名administrator和密码letmein：

```
http://administrator:letmein@www.wrox.com/administration/index.html
```

端口类似于 Web 服务器的门。一个 Web 服务器通常具有几个服务器程序，这些服务器程序运行于同一台机器中，每个程序使用不同的端口进行通信。例如，http://和 https://默认情况下使用不同的端口(标准 http://通常使用端口 80，https://通常使用端口 443)。

端口的使用机会很多，但是如果指定端口，则应该将其放在域名的后面，并且以冒号隔开。例如，如下地址指定了运行在 8080 端口上的 Web 服务器：

```
http://www.wrox.com:8080/index.html
```

段标识符位于文件名的后面，用于指示页面的特定部分，浏览器应该直接转到该部分。它们通常用于长页面中，以方便用户快速到达页面的特定部分，而不需要滚动整个页面以查找该位置。

段标识符以英镑符号(#)与文件名隔开，如下所示：

```
http://www.wrox.com/newTitles/index.html#HTML
```

本章后面的部分中将详细介绍段标识符。

路径参数用于向服务器程序传递额外的信息。它们以问号(?)与 URL 隔开，并且以名/值对的形式出现，名和值之间以等号隔开(如果没有问号，路径参数就相当类似于属性)。路径参数通常用于从访问者处收集信息，并向服务器上的程序传递信息，以便向访问者返回所需要的页面。这种参数通常称为查询字符串。

当使用 Web 页面上的表单时，例如搜索表单或在线订购表单，浏览器能够将用户提供的信息附加在 URL 中，以向服务器传递信息——此时用户不需要将路径参数输入到 URL 中。

在下面的例子中，将路径参数 searchTerm=HTML 添加到 URL 中，用于表明用户搜索术语 HTML：

```
http://www.wrox.com/search.aspx?searchTerm=HTML
```

2.2.3 绝对 URL 和相对 URL

URL 用于定位 Internet 中的资源。每一个 Web 页面和图像——事实上是 Internet 中的每一个文件——都具有唯一的 URL，即能够用于找到特定文件的地址。Internet 中任意两个文件都不会具有相同的 URL。

如果想访问某个 Web 站点中的特定页面，需要在浏览器的地址栏中输入该页面的 URL。例如，为了获得本章前面虚构的新闻站点中的电影页面，可以输入如下 URL：

```
http://www.exampleNewsSite.com/Entertainment/Film/index.html
```

这是一个绝对 URL，其中包含了用于唯一标识 Internet 中某个文件所需的所有信息。

绝对 URL 可能很快变得相当长，并且 Web 站点的每一个页面可以包含很多链接。因此，有必要了解指向 Web 站点中文件的 URL 的简写形式：相对 URL。

相对 URL 用于指示资源相对于当前页面的位置。例如，假设您正在查看如下虚拟新闻站点的娱乐部分的索引页面：

```
http://www.exampleNewsSite.com/Entertainment/index.html
```

然后您希望添加到每个分部的链接：Film、TV、Arts和Music。没有使用每个页面的完整URL，而是可以使用相对URL。例如：

```
Film/index.html
TV/index.html
Arts/index.html
Music/index.html
```

可以确定的是，比起输入下面的地址，使用相对 URL 会快很多：

```
http://www.exampleNewsSite.com/Entertainment/Film/index.html
http://www.exampleNewsSite.com/Entertainment/TV/index.html
http://www.exampleNewsSite.com/Entertainment/Arts/index.html
http://www.exampleNewsSite.com/Entertainment/Music/index.html
```

您可能有兴趣知道的是，Web 浏览器仍然请求完整 URL 而不是缩短的相对 URL。但是，浏览器会将相对 URL 转换为完整的绝对 URL。

在站点中使用相对 URL 的另一种关键优点是，用户可以方便地改变站点的域名或者将站点的某个分布复制到新站点中，而不需要改变所有的链接，因为每一个链接都相对于相同站点中的其他页面。

注意：

相对 ULR 只适用于相同 Web 站点的相同目录结构中的链接；不能使用它们链接到其他服务器上的页面。

下面的小节中将概述可以使用的不同类型的相对 URL。

1. 相同目录

如果希望链接到或者包含相同目录中的资源，可以只使用该文件的名称。例如，为了从主页面(index.html)链接到“联系我们”页面(contactUs.html)，可以使用如下形式的相对 URL：

```
contactUs.html
```

该文件位于相同的文件夹中，因此不需要指定其他任何内容。

2. 子目录

图 2-4 中的 Film、TV、Arts 和 Music 目录都是 Entertainment 目录的子目录。如果正在编写 Entertainment 目录中的一个页面，可以采用如下形式的相对 URL 创建到子目录的索引页面的链接：

```
Film/index.html
TV/index.html
Arts/index.html
Music/index.html
```

该种相对 URL 包含了子目录名，后面紧跟一个正斜杠(/)以及希望链接到的页面的名称。

对于每一个额外的子目录，可以只添加该目录的名称，并紧跟一个正斜杠(/)。因此，如果从站点根文件夹中的页面(例如站点的主页面)创建链接，则可以使用如下所示的相对 URL 到达相同的页面：

```
Entertainment/Film/index.html
Entertainment/TV/index.html
Entertainment/Arts/index.html
Entertainment/Music/index.html
```

3. 父目录

如果想创建从某个目录到它的父目录(包含该目录的目录)的链接，则可以使用“../”符号(两个句点或圆点，后跟一个正斜杠)。例如，为了从 Music 目录中的一个页面链接到 Entertainment 目录中的某个页面，可以使用如下的相对 URL：

```
../index.html
```

如果想创建从 Music 目录到根目录的链接，则可以重复使用“../”符号：

```
../../index.html
```

每一次重复使用“../”符号，就会向上到达另一个目录。

4. 从根目录

也可以指示文件相对于站点的根文件夹。因此，如果想创建从站点中的任意页面到 contactUs.html页面的链接，可以使用该页面的路径，并在路径前面放置一个正斜杠。例如，如果contactUs.html页面位于根文件夹中，只需要输入：

```
/contactUs.html
```

或者，可以使用如下形式的相对 ORL 创建从站点的任意位置到 Music 部分的索引页面的链接：

```
/Entertainment/Music/index.html
```

起始部分的正斜杠表示根目录，后面的部分是从根目录开始的路径。

5. 默认文件

您可能已经注意到，在许多站点中不需要实际地指定希望浏览的确切页面。例如，只需要输入域名或者域名加目录，如下所示：

```
http://www.exampleNewsSite.com/
```

或者

```
http://www.exampleNewsSite.com/Entertainment/
```

这是因为许多 Web 服务器允许它们的所有者在访问者只指定目录时向其发送默认文件。因此，如果输入 http://www.exampleNewsSite.com/Entertainment/，则服务器将返回 Entertainment 目录的默认文件，如果输入 http://www.exampleNewSite.com/，则服务器返回 Web 站点根文件夹的默认文件(请记住，正斜杠字符可以用于指示相对于根目录)。

大多数服务器使用 index.html 或者 default.html 作为默认的 HTML 文件名(如果使用的是服务器端的语言，例如 ASP.net 或 PHP，则文件名可能有所不同)。

您可能已经注意到，这些 URL 都以正斜杠字符结束。如果没有在请求 URL 的末尾包含结尾的正斜杠，则 URL 可能如下：

```
http://www.exampleNewsSite.com
```

或者

```
http://www.exampleNewsSite.com/Entertainment
```

在这些情形下，服务器倾向于通过这些名称查找文件，而不是查找一个页面，并告诉浏览器请求在末尾带有正斜杠的相同页面。例如，如果在浏览器中输入不包含结尾斜杠字符的 URL：

```
http://www.wrox.com
```

大多数浏览器将返回主页面：

```
http://www.wrox.com/
```

因此，如果创建到 Web 站点中的文件夹的链接(而不是特定的页面)，最好在 URL 的末尾添加一条正斜杠。

2.2.4　<base>元素

前面已经提到，当浏览器遇到相对 URL 时，它实际上是将相对 URL 转换为完整的绝对 URL。<base>元素允许用户指定页面的基 URL，这样当浏览器遇到相对 URL 时，将在它们的前面添加基 URL。

可以指定基 URL 为 href 属性的值。例如，可以以如下形式指定 http://www.exampleSite2.com/ 的基 URL：

```
<base href="http://www.exampleSite2.com/" />
```

在这种情形下，如下所示的相对 URL：

```
Entertainment/Arts/index.html
```

最终导致浏览器请求如下的页面：

```
http://www.exampleSite2.com/Entertainment/Arts/index.html
```

当某个页面被移动到新的服务器，但是仍然想让该页面上所有的链接指向初始的站点，或者当页面本身没有 URL 时(例如 HTML 的 e-mail)，基 URL 就非常有用。

除了 href 属性之外，<base>元素能够附带的另一个唯一的属性是 id 属性。

2.3 利用<a>元素创建链接

在本章起始部分已经给出了一些利用<a>元素的示例。但是，它们只是用于表明利用<a>元素能够做哪些事情。现在您已经对目录结构有了一定的了解，因此有必要进一步了解链接。

Web 上的所有超文本链接能够让访问者从 Web 的一个部分到达另外一个部分。前面已经介绍了让访问者从一个页面到达另外一个页面的链接(本部分将更深入地介绍这方面的内容)。您可能已经遇到过能够将访问者带到页面特定部分的链接(相同页面的特定部分或者不同页面的特定部分)。

类似于所有的旅程，这些链接都有称为“源”的起点和称为“目的地”的终点，它们都称为“锚点”

页面上可以单击的每个链接实际上是源锚点，可以使用<a>元素创建源锚点。

2.3.1 利用 href 属性创建源锚点

当讨论 Web 上的链接时，大多数人所能想到的是源锚点——无论链接包含的是文本还是图像。用户可以单击源锚点并期望到达其他某个位置。

本章前面已经介绍过，用户能够单击的链接的文本包含在起始标签<a>和结束标签之间，并且由 href 属性的值指定用户应该转到的 URL。

例如，当用户单击“Wrox Press website”(位于<a>元素内)时，链接将用户转到 http://www.wrox.com/：

```
Why not visit the <a href="http://www.wrox.com/">Wrox Press website</a> to
find out about some of our other books?
```

而位于虚构新闻站点主页面的如下链接将用户转到 Film 的主页面(注意，该链接使用了相对 URL)：

```
You can see more films in the <a href="Entertainment/Film/index.html">film
section</a>.
```

默认情况下，链接的外观如图 2-6 所示，即带有下划线的蓝色文本。

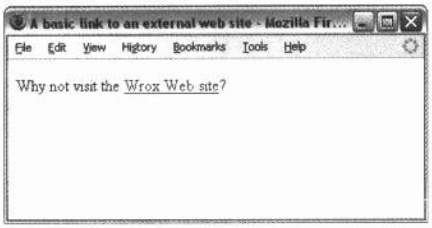


图 2-6

只有在希望链接到页面的特定部分时，才需要指定一个目的地锚点，下一部分中将介绍这方面的内容。

2.3.2　利用 name 和 id 属性创建目的地锚点(链接到页面的特定部分)

如果 Web 页面很长，则您可能想链接到该页面的特定部分。当页面无法完全显示在浏览器窗口中，并且用户必须滚动以查找页面的相关部分时，链接到页面的特定部分就非常有用。

目的地锚点允许页面开发人员标记页面中源链接可以指向的特定位置。

Web 页面上到页面特定部分的常见链接示例包括：

- 长页面底部的“返回顶部”链接
- 页面内容的列表，该列表可将用户转到页面的相关部分
- 到脚注或定义的链接

可以使用<a>元素创建目的地锚点，但当用于创建目的地锚点时，它必须附带一个 id 属性(如果创建可以通过早期的浏览器查看的页面，例如 IE3 和 Netscape 3，还需要附带一个 name 属性)，因为 id 属性在 HTML 4 中才被引用。

注意：

第 1 章中介绍过，name 属性和 id 属性是两个通用属性，大多数元素都能够附带它们。

借助一个示例进行说明，假设某个页面很长并具有一个主题头和几个副标题。屏幕中无法一次性显示整个页面，用户必须滚动以便浏览它。因此，需要在文档的起始部分添加到每个主题头的链接。

在创建到页面各个部分的链接之前(使用源锚点)，必须首先添加目的地锚点。可以在下面的代码中看到：每一个页面的副标题包含一个<a>元素，并且<a>元素具有一个 id 属性，该属性的值唯一标识了该部分：

```
<h1>Linking and Navigation</h1>
<h2><a id="URL">URLs</a></h2>
<h2><a id="SourceAnchors">Source Anchors</a></h2>
<h2><a id="DestinationAnchors">Destination Anchors</a></h2>
<h2><a id="Examples">Examples</a></h2>
```

添加目的地锚点之后，可以添加源锚点以链接到这些部分，如下所示：

```
<p>This page covers the following topics:
  <ul>
    <li><a href="#URL">URLs</a></li>
    <li><a href="#SourceAnchors">Source Anchors</a></li>
    <li><a href="#DestinationAnchors">Destination Anchors</a></li>
    <li><a href="#Examples">Examples</a></li>
  </ul>
</p>
```

源锚点中 href 属性的值是英镑符号(#)后跟 id 属性的值。

图 2-7a 中的示例页面具有几个到页面不同部分的链接；图 2-7b 演示了用户单击第二

个链接时会直接转到页面的相应部分。可以从 Wrox.com 处获得本章下载代码中这个示例的完整代码并尝试运行它；示例的文件名为 ch02_eg06.html。

注意，目的地锚点中始终具有一些内容，这一点很重要；否则某些浏览器可能找不到目的地。例如，不应当使用下面的代码指示页面的顶部：

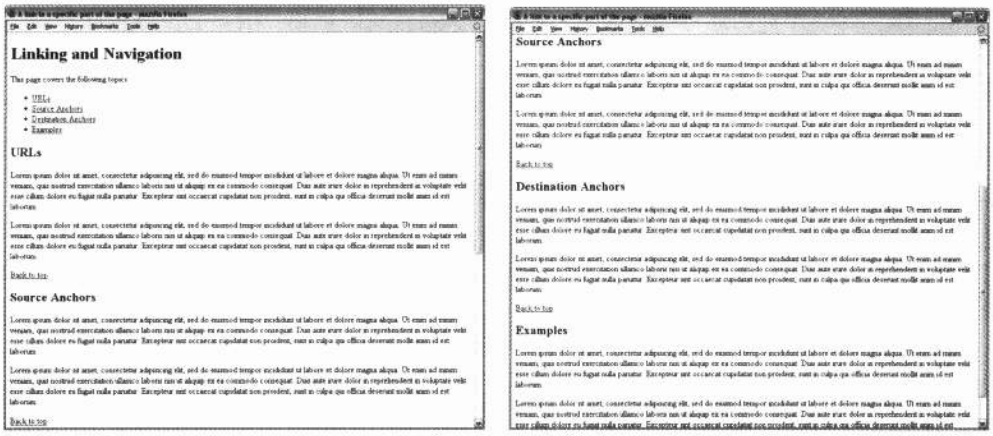
```
<a id="top"></a>
```

相反，应该在主题头或其他一些内容周围放置目的地锚点，如下所示：

```
<h1><a id="top">Linking and Navigation</a></h1>
```

如果某个人想从不同的 Web 站点处链接到这个页面的特定部分，应当添加这个页面的完整 URL，后面紧跟英镑符号#，然后是 id 属性的值，如下所示：

```
http://www.example.com/HTML/links.html#SourceAnchors
```



(a) (b)

图 2-7

注意：

在一个页面中，name 和 id 属性的值必须唯一，并且源锚点必须匹配相应的目的地锚点。

2.3.3 <a>元素的其他属性

<a>元素支持所有的通用属性、UI 事件属性以及下面的一些属性：

```
accesskey charset cords href hreflang rel rev shape style tabindex target
type
```

1. accesskey 属性

accesskey 属性提供可用于激活链接的键盘快捷键。例如，可以使用 T 键作为访问键，当用户在按下 T 键同时按下 Alt 键或 Ctrl 键(取决于操作系统)时，将激活相应的链接。这

意味着浏览器将立即跟踪该链接，或者意味着浏览器将高亮显示该链接，然后用户必须按下 Enter 或 Return 键以跟踪该链接。

应该在源锚点中指定 accesskey 属性。例如，如果想在用户按下 T 键(同时按下 Alt 键或 Ctrl 键)时跟踪指向页面顶部的链接，可以以如下方式使用 accesskey 属性：

```
<a id="bottom" accesskey="t">Back to top</a>
```

注意，按键不区分大小写。在第 5 章中讨论表单时将更详细地介绍 accesskey 属性(并提供一些示例)。

2. charset 属性

charset 属性指示 URL 所指向的文档的字符集编码，它的值必须是标识字符集的字符串，例如 UTF-8 或者 ISO-8859-1(关于字符集列表可以查看附录 F 确认)。

charset 属性通常用于源锚点中，并且仅当包含该链接的主文档的语言与当前文档的语言不同时才使用它。例如：

```
<a href="http://www.wrox.com/" charset="UTF-8">Wrox Web Site</a>
```

该属性在链接到外语站点并且这些站点以一些特定的编码编写时非常有用，用户可能不能够理解这些编码，甚至不能够查看它们(例如，并不是所有的美国计算机都安装了查看中文文本所需的字符)。

3. coords 属性

coords 属性用于源锚点中包含图像时。使用它可以创建图像映射，在图像映射中，图像的不同部分链接到不同的文档或相同文档的不同部分。coords 属性的值是 x 和 y 坐标，用于指示图像的哪一部分应该跟踪该链接。

第 3 章中将介绍如何使用图像作为链接。

4. hreflang 属性

hreflang 属性指定源链接所指向的文档的语言，并且仅当给定 href 属性的值时才能使用它。例如：

```
<a href="http://www.wrox.com/" hreflang="en-US">Wrox Web Site</a>
```

附录 G 中列举了该属性可能具有的值。

5. rel 属性

rel 属性用于源锚点中，指示当前文档与 href 属性所指定的资源之间的关系。当前的主要浏览器都不使用该属性，但是有些自动化应用程序可能使用它。例如，下面的链接使用 rel 属性表明它的目的地是文档中使用的词汇表：

```
For more information, please read the  <a href="#glossary"
rel="glossary">glossary</a>.
```

表 2-2 中给出了 rel 属性的可能值。

表 2-2

值	描　述
toc(或者 contents)	作为当前文档的内容表的文档
index	作为当前文档的索引的文档
glossary	包含与当前文档相关的词汇表的文档
copyright	包含当前文档的版权声明的文档
start	一系列有序文档中的第一个文档
next	一系列有序文档中的下一个文档
prev(或者 previous)	一系列有序文档中的前一个文档
help	帮助用户理解或者浏览页面和/或站点的文档
chapter	作为文档集合内某一章的文档
section	作为文档集合内某个部分的文档
subsection	作为文档集合内某个分部的文档
appendix	作为文档集合内某个附录的文档

6. rev 属性

rev 属性的作用与 rel 属性相同，但用于目的地锚点中以描述目的地和源的关系。当前的主要浏览器都不支持该属性。

7. shape 属性

如果想创建图像映射，可以使用 shape 属性指示作为可单击热点的区域的形状。第 3 章中将更详细地介绍 shape 属性，其中将介绍如何创建图像映射。

8. tabindex 属性

为了理解 tabindex 属性，需要先了解元素获得焦点的意义；用户能够交互的任何元素都能够获得焦点。当页面加载后，如果用户按下 Tab 键，则浏览器将在页面中能够交互的部分之间移动焦点。页面上能够获得焦点的部分包括链接、表单的某些部分(例如允许输入文本的方框)。当一个链接获得焦点之后，用户如果按下 Enter 键，则该链接将被激活。可以看到焦点作用于 Google Web 站点；如果重复按下 Tab 键，则焦点将依次通过在页面上的每个链接。焦点依次通过每一个链接之后，将进入到用于输入搜索术语的文本框中，然后通过站点的按钮，并且通常最终返回到用于输入 URL 的位置。当用户再次重复按下 Tab 键之后，焦点将在相同的元素上循环地传递。

tabindex 属性允许用户指定在按下 Tab 键时链接(或窗体控件)获得焦点的顺序。这样，当用户按下 Tab 键时，焦点可能跳到页面上用户想交互的关键项。

tabindex 属性的值是 0 到 32 767 之间的数字。tabindex 属性值为 1 的链接获得焦点的

顺序在 tabindex 属性值为 20 的链接之前(如果使用 tabindex 值 0，则按照在文档中出现的顺序获得焦点)。第 5 章中将详细地介绍 tabindex 属性。

9. target 属性

target 属性用于指示包含在链接中的文档应当在哪一个窗口或框架中打开。第 6 章中将更详细地介绍框架方面的知识。该属性的语法形式如下：

```
<a href="Page2.html" target="main">Page 2</a>
```

当用户单击 Page 2 链接时，在称为 main 的窗口或框架中加载文档 Page2.html。如果想在一个新窗口中打开链接，可以将 target 属性的值设置为_blank。

10. title 属性

本章开始部分提到，title 属性对于图像链接来说是非常重要的，它也能够在大多数浏览器以可视化文本工具提示的形式向访问者提供额外的信息，或者在语音浏览器中向具有听觉障碍的人提供听觉提示。本章开始部分的图 2-2 中演示了当用户当光标悬停在链接上时，title 属性在 Firefox 浏览器中的显示效果。

11. type 属性

type 属性指定链接的 MIME 类型。附录 H 中列举了一些 MIME 类型。HTML 页面的 MIME 类型是 text/html，而 JPEG 图像的 MIME 类型是 img/jpeg。下面关于 type 属性的示例用于指示链接所指向的文档是 HTML 文档：

```
<a href="index.html" type="text/html">Index</a>
```

理论上，浏览器能够使用 type 属性中的信息以不同的方式显示页面，或者向用户指示目的地页面的格式，尽管目前还没有用到该功能。

试一试　在页面中创建链接

下面尝试制作一个长页面，该页面具有一些到页面不同部分的链接。在这个示例中，需要创建一个餐馆菜单的页面。因此，打开文本编辑器或 Web 页面制作工具并遵循以下步骤：

(1) 开始部分是 XML 声明、DOCTYPE 声明和一些作为文档框架的元素：<html>、<head>、<title>和<body>。记住赋予文档一个标题，并在<html>根元素中添加名称空间标识符：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">

<head>
   <title>A menu example</title>
```

```
    </head>
    <body>

    </body>
    </html>
```

(2) 在<body>元素中添加页面的题头。每一个题头应该具有一个目的地锚点，以便直接链接到页面的该部分。主题头用作“返回到顶部”链接，而菜单中的每一道菜都有一个用于描述它的 id 属性：

```
<body>
  <h1><a id="top">Wrox Cafe Menu</a></h1>
  <h2><a id="starters">Starters</a></h2>
  <h2><a id="mains">Main Courses</a></h2>
  <h2><a id="desserts">Desserts</a></h2>
</body>
```

(3) 在标题和起始部分之间，不仅要有一个介绍性的段落，而且要有一个链接到每一道菜的菜单。为了符合 Strict XHTML 规范，顶部的链接位于一个块级元素 中：

```
<h1><a id="top">Wrox Cafe Menu</a></h1>
<div id="nav"><a href="#starters">Starters</a> | <a href="#mains">Main
Courses</a> | <a href="#desserts">Desserts</a></div>

<p>Welcome to the Wrox Cafe, where we pride ourselves on good, honest home
cooked food at good, honest prices.</p>
<h2><a id="starters">Starters</a></h2>
```

(4) 在页面的底部有一个关于素菜的描述。到素食项的链接将指向该描述，因此它需要具有一个目的地锚点。

```
<p><a id="vege">Items marked with a (v) are suitable for vegetarians.</a></p>
```

(5) 最后，以项目列表的形式添加菜单中的每一项。注意，素食项具有一个链接到素菜描述的链接。不要忘记添加一个“返回到顶部”链接。

```
<h2><a id="starters">Starters</a></h2>
<ul>
   <li>Chestnut and Mushroom Goujons (<a href="#vege">v</a>)</li>
   <li>Goat Cheese Salad (<a href="#vege">v</a>)</li>
   <li>Honey Soy Chicken Kebabs</li>
   <li>Seafood Salad</li>
</ul>
<p><small><a href="#top">Back to top</a></small></p>
<h2><a id="mains">Main courses</a></h2>
<ul>
   <li>Spinach and Ricotta Roulade (<a href="#vege">v</a>)</li>
   <li>Beef Tournados with Mustard and Dill Sauce</li>
   <li>Roast Chicken Salad</li>
   <li>Icelandic Cod with Parsley Sauce</li>
```

```
    <li>Mushroom Wellington (<a href="#vege">v</a>)</li>
</ul>
<p><small><a href="#top">Back to top</a></small></p>
<h2><a id="desserts">Desserts</a></h2>
<ul>
    <li>Lemon Sorbet (<a href="#vege">v</a>)</li>
    <li>Chocolate Mud Pie (<a href="#vege">v</a>)</li>
    <li>Pecan Pie (<a href="#vege">v</a>)</li>
    <li>Selection of Fine Cheeses from Around the World</li>
</ul>
<p><small><a href="#top">Back to top</a></small></p>
```

(6) 将示例存储为 menu.html，并在浏览器中查看它。最终结果将如图 2-8 所示。

图 2-8

工作原理

您已经看到了页面的框架部分(以及它前面的一些声明)，因此下面重点讨论链接。

在形成简单导航栏的第一个题头之下有 3 个源锚点。当单击时，这些源锚点将把用户带到页面相应的部分。这些项保存在一个 元素之中，因为在 Strict XHTML 1.0 中，<a>元素应当位于一个块级元素中——尽管一些早期的版本不考虑这些规范。

```
    <div id="nav"><a href="#starters">Starters</a> | <a href="#mains">Main
courses
    </a> | <a href="#esserts">Desserts</a></div>
```

元素中的 id 属性用于标识这个块级分组元素的目的。这个元素不像其他一些元素(例如<p>或者<h2>)一样具有特定的目的，因此添加 id 属性的作用是提示该元素分组的内容。

在菜单的每一部分之下存在一个额外的源锚点，用于将用户带回页面的顶部。

```
<p><small><a href="#top">Back to top</a></small></p>
```

最后，带有文本 v 的源锚点表明该项是素食，并且用于将用户带到页面底部解释 v 的意义的解答部分。

```
<li>Mushroom wellington (<a href="#vege">v</a>)</li>
```

目的地锚点使用 id 属性指明链接的潜在目标。每一个题头都包含一个目的地锚点。主菜单题头要求有一个锚点，以便“返回到顶部”链接能够将用户带到页面的顶部；而副标题的锚点便于顶部的导航菜单将用户带到页面相应的部分。

记住，目的地锚点必须具有一些内容——它们不能为空，否则浏览器无法识别它们，这也是它们被放置在题头元素中并采用实际的题头名的原因：

```
<h1><a id="top">Wrox Cafe Menu</a></h1>
<h2><a id="starters">Starters</a></h2>
<h2><a id="mains">Main courses</a></h2>
<h2><a id="desserts">Desserts</a></h2>
```

同样，底部用于指明 v 符号意义的段落包含一个目的地锚点，这一点类似于题头。

```
<p><a id="vege">Items marked with a (v) are suitable for vegetarians.</a></p>
```

2.4 高级 e-mail 链接

本章的开始部分介绍过，可以让链接自动打开用户的默认 e-mail 编辑器并向相应的地址(或提供的其他任何 e-mail 地址)发送一封 e-mail。完成该操作的语句如下所示：

```
<a href="mailto:info@example.org">info@example.org</a>
```

用户也可以指定其他信息，例如邮件的主题、主体以及应当抄送或者密送到的人。

为了在 e-mail 中添加一个主题，在 e-mail 地址的后面跟上一个问号，以将 e-mail 地址和额外的值隔开。然后使用名/值对指定希望控制邮件的额外属性，名和值之间以等号隔开。

例如，为了将主题设置为 Enquiry，需要添加 subject 属性名和主题的内容，如下所示：

```
<a href="mailto:info@example.org?subject=Enquiry">
```

可以使用多个名/值对指定多个属性，每个名/值对之间以&符号隔开。下面的示例中添加了主题和抄送地址：

```
<a href="mailto:info@example.org?subject=XHTML&cc=sales@example.org"></a>
```

表 2-3 列出了能够添加的所有属性。

表 2-3

属　性	目　的
subject	在 e-mail 中添加主题行；使用它可以添加 e-mail 的主题行，以便用户更容易识别邮件的来源
body	在 e-mail 主体中添加消息，但是需要知道的是，用户能够更改该消息
cc	向抄送地址发送一份邮件的副本；该值必须是有效的 e-mail 地址。如果想提供多个地址，只需重复该属性，并使用&符号将它们隔开
bcc	秘密地向密送地址发送一份邮件副本，并且任何收件人都不能看到其他的收件人；该值必须是有效的 e-mail 地址。如果想提供多个地址，只需重复该属性，并使用&符号将它们隔开

如果想在主题行中任意单词之间添加空格，应当在单词间添加%20，而不是直接添加空格。如果想在 e-mail 的消息主体中换行，应当添加%0D%0A。

人们经常只在 e-mail 链接中添加 e-mail 地址。如果想添加主题行或者消息主体，最好创建一个 e-mail 表单，第 5 章中将对此进行介绍。

2.5　本章小结

本章中主要介绍了链接方面的知识，链接是 XHTML 的一部分，它为超文本带来特殊的效果。链接使得站点的访问者能够在页面之间切换，甚至在页面的不同部分之间切换(从而不用滚动就可以找到所需要的位置)。

您已经看到可以使用<a>元素创建源锚点，它是大多数人在提到 Web 上的链接时所能够想到的对象。源锚点的内容是用户能够单击的内容——它应该是关于目标的包含丰富信息的精确描述(而不只是类似于“单击此处”的文本)，或者可以是一幅图像(第 3 章中将介绍这方面的内容)。

也可以使用<a>元素创建目的地锚点。目的地锚点有点类似于索引点或者特殊标记，因为它们允许创建直接转到页面某个部分的链接。目的地锚点应该始终具有一些内容，并且在 HTML 中引入的用于目的地锚点的原有 name 属性在 Strict XHTML 中被 id 属性取代(尽管 id 属性只能用于 3＋1 版本的浏览器中)。

在介绍链接的过程中，本章也介绍了关于 URL 的更多内容，特别是绝对 URL(类似于出现在浏览器地址栏中的内容)和相对 URL(它描述了资源相对于包含它的文档的位置)间的区别。对于学习下一章和学习如何在文档中添加图像以及其他对象来说，了解使用相对 URL 的不同方法将非常有帮助。

2.6　练习

所有练习的答案都在附录 A 中给出。

1. 回顾本章“试一试”中的示例，其中创建了一个菜单和一个新页面，该页面中具有一些指向菜单中每一道菜的链接(类似于菜单页面顶部的链接)。然后添加一个到 Wrox Press Web 站点(www.wrox.com)的链接。页面应当如图 2-9 所示。

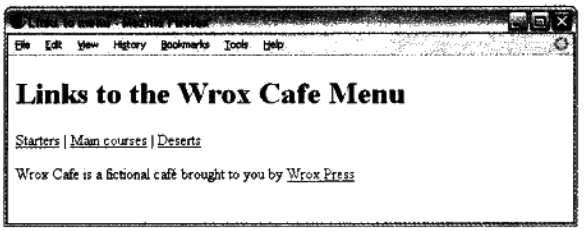


图 2-9

2. 采用如下语句，在应当具有链接的部分周围放置<a>元素。

```
<p>To find out why advertising on our site works, visit the testimonials
page.</p>
```

3. 下面<a>元素放置的位置存在什么错误？

```
<p>You can read the full article <a>here</a>.</p>
```

第3章

图像和对象

在本章中，您将开始学习一些能够给 Web 页面注入生机的 Web 设计技巧。首先将学习如何利用<img>元素向文档中添加图像。您将了解用于 Web 上的图像的主要格式之间的区别，并了解如何准备用于 Web 的图像。本章还将介绍如何将图像制作为链接，甚至是如何将图像划分为多个部分，以便不同的部分链接到不同的页面——这被称为图像映射。

本章然后将介绍<object>元素，可以利用它向页面中插入各种对象，从 MP3、Flash 电影到 ActiveX 控件，甚至是图像。

3.1 在站点中添加图像

图像和图形能够为站点带来生机。在本节中不仅将介绍如何向页面中插入图像和图形，而且将介绍在 Web 中能够使用的不同的图像格式(例如 GIF、JPEG 和 PNG)。您将了解到什么时候应当选择何种图像格式。

在 Web 中使用图像时必须非常小心，因为如果没有正确地准备它们，将降低页面的加载速度——速度慢的站点将无法吸引到访问者。另外，您很可能是在桌面计算机或笔记本计算机上编写您的第一个站点，因而无法意识到页面加载的时间长短，直到它真正地加载到 Web 服务器上。因此，选择正确格式的图像并正确地保存它们，将有助于使得站点的加载速度更快，并使得访问者更乐意访问该站点。

注意：

为了练习目的，可以从其他站点下载一些图像，方式是右击图像(或者 Ctrl+单击)并选择“下载图像到磁盘”或者“图像另存为”选项。但是请记住，图像是有版权的，如果在站点中使用了其他人的图像，将会负法律责任。

掌握如何向页面中插入正确类型的图像之后，您将学习如何将它们转化为链接，甚至是如何编写用于划分它们的代码，以便当用户单击图像的不同部分时会被转到不同的 Web 页面。

3.1.1 图像格式的类型

首先有必要了解计算机如何存储和绘制图片。计算机创建图形的方式主要包括两种：

- 位图图形：位图图形将图片划分为由像素组成的网格，并指定每一个像素的颜色，类似于计算机告诉显示屏幕每一个像素的颜色。广义地说，位图是照片以及色调与颜色的复杂渐变的理想格式。存在几种不同的位图格式；常见的包括 JPEG、GIF、TIFF、PNG，这些格式被相当混淆地统一命名为位图或 BMP。在本章后面的部分中将更详细地介绍 JPEG、GIF 和 PNG。
- 矢量图形：矢量图形将图像划分为线和形状(类似于线框制图)，并以坐标的形式存储线，然后利用颜色填充线之间的空间。矢量图形通常用于艺术线条、图解和动画，它们通常适用于具有大面积单调颜色(相对于纹理、多种色调的颜色和摄影风格)的图像。

在早期，位图是 Web 中使用的主要图像格式。但是，最近有些格式(例如 Flash 和 SVG)也采用了矢量图形。

3.1.2 位图图像

Web 上的大多数静态图像是位图图像。本章前面已经提到，位图被划分为由像素组成的网格。如果近距离观察计算机屏幕，则能够发现组成屏幕的像素。请查看图 3-1，该示例中的位图图像的一部分已经被修改，以演示像素如何组成图像。

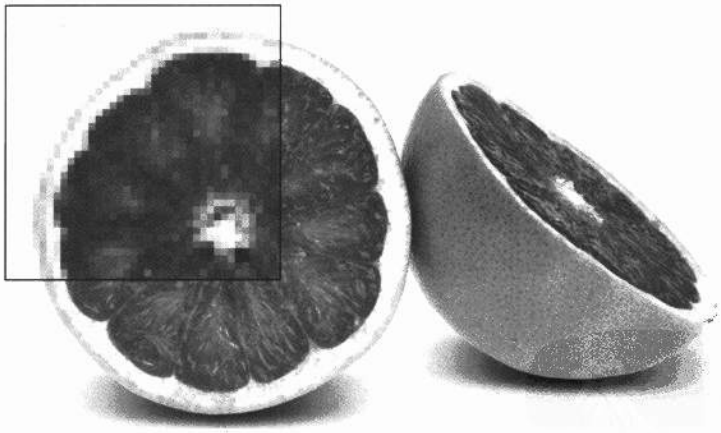


图 3-1

屏幕的每平方英寸中像素的数量称为图像的分辨率。Web 中的图像最大能够每英寸显示 72 个像素；用于打印的图像通常具有更高的分辨率，提供给打印机的图像的分辨率通常是每英寸 300 点(注意，屏幕所采用的分辨率单位是“像素/英寸”，而打印机所采用的单位是“点/英寸”)。图像每英寸包含的像素或点的数目越多，则文件的尺寸将越大。因而，Web 上使用的任意图像所采用的分辨率是每英寸 72 点。如果以更高的分辨率保存 Web 上的图像，则将创建不必要的大型文件，从而导致更长的下载时间。

注意：

虽然可以很容易将 300 点/英寸的图像存储为 72 像素/英寸以用于 Web，但是不能简单地将 72 像素/英寸的图像增加为 300 点/英寸，因为无法知道每平方英寸中另外 228 个像素的颜色。如果尝试增加图像的分辨率，则图像将可能呈现为颗粒状。因此，如果拥有 300 点/英寸的高分辨率图片，则以该尺寸保存它是有意义的，以便将来以较大尺寸或较高分辨率显示它。

浏览器通常支持 3 种常用的位图图形格式，大多数图形程序以这 3 种格式保存图像：

- GIF：Graphics Interchange Format(图形交换格式，发音为“gif”或者“jif”)
- JPEG：Joint Photographic Experts Group Format(联合照相专家组格式，发音为“jay peg”)
- PNG：Portable Network Graphics(可移植网络图形，发音为“ping”或者“pee en gee”)

下面简单介绍这些格式，因为理解这些格式的工作原理将有助于选择如何保存图像。

1. GIF 图像

在 Web 出现的早期，GIF(或图形交换格式)是所有 Web 图形的标准。使用多达 256 色的调色板创建 GIF 图像，其中图像的每一个像素是 256 种颜色的一种。每一幅不同的 GIF 图像具有不同的 256 色调色板，而这 256 种颜色是从超过 1600 万种颜色中选出来的。保存图像的程序将选择能够最好表示图像的调色板。

GIF 文件将调色板存储在查找表中，每一个像素参考查找表中的颜色信息，而不是为每个像素指定单独的颜色信息。因此，如果许多像素使用相同的颜色，则图像不用重复相同的颜色信息，并最终产生较小的图像文件。这种存储图像的方式称为索引颜色格式。图 3-2 中给出了由 Adobe Photoshop 软件创建的一个 GIF 文件。可以在图的右边看到用于该图像的颜色调色板，即图像右边下半部分由一组方格组成的部分。

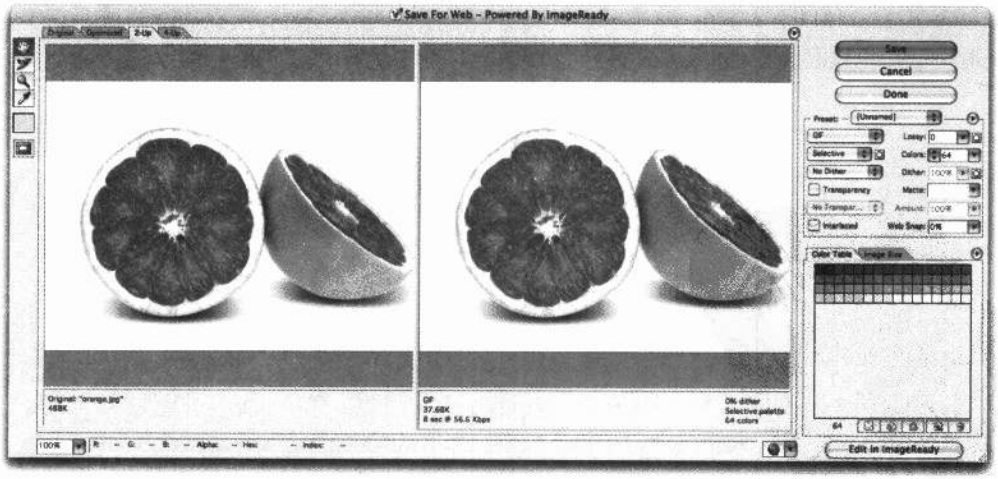


图 3-2

因为 GIF 图像保存颜色信息的方式是采用查找表，所以它们特别适合于具有大面积单调颜色区域的图像。单调颜色区域是指只有一种色调的区域；例如，只具有一种绿色的矩形就是单调颜色，而一幅草地图片则具有多种不同的绿色。图像所使用的颜色越少，则它的 GIF 文件将越小。

如果一幅 GIF 图像中的颜色少于 16 种(这种情况下也可以称为 4 位 GIF)，则它的文件尺寸将只有使用 256 种颜色的 GIF 文件(称为 8 位 GIF)的一半。因此，如果创建的图像使用少于 16 种颜色，则有必要检查程序是否自动地将图像保存为 4 位 GIF，因为这将导致产生更小的文件，并且下载该文件的速度比下载 8 位 GIF 文件的速度更快。

注意：

如果文本或线本来具有两种颜色(例如黑色和白色)，但使用了平滑边以使它们看上去更平滑，则图像所包含的颜色将超过两种，因为边使用了多种其他颜色以便使图像看上去更平滑。

如果 GIF 文件需要使用 256 种以上的颜色，则大多数图形程序在保存 GIF 文件时使用一种称为抖动的技术，以更好地表示额外的颜色。因此，它们在相邻像素中使用两种或更多的颜色以创建第三种颜色的效果。抖动具有下面两种缺陷：

- 它会在颜色中产生一些条带。这种情况通常发生在图像的某些地方初始时看上去颜色很单调，但实际上具有很多种不同的色调。例如，当一种颜色和另外一种颜色之间具有平滑的过渡(称为渐变)时，抖动将使用许多种不同的颜色以创建平滑效果。此时，颜色之间的改变将变得更加明显。
- 如果抖动颜色旁边具有单调色，则将能够看到颜色改变发生的位置(因为抖动颜色实际上由多种颜色组成)。

图 3-3 演示了保存为 GIF 格式时，即使一个简单的渐变也可以产生条带，原因是图像中包含了 256 种以上的颜色——如果仔细地观察图像，您会发现梯度具有垂直线，而不是平滑地从黑色过渡为白色。

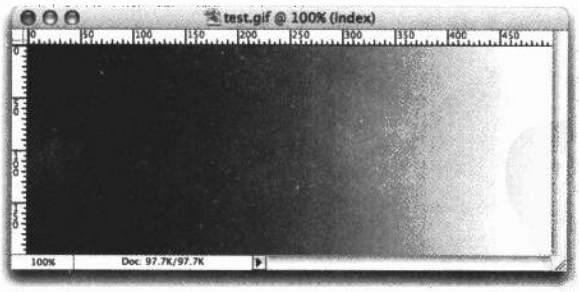


图 3-3

因为 GIF 格式仅支持 256 种颜色，不得不使用抖动技术以获得更多的颜色，所以它们不适合包含 256 种以上的颜色的详细照片。如果照片、渐变或任意图像中具有相同颜色的相似色调，最好是使用 JPEG 格式，它能够支持无限种颜色；或者使用 PNG 格式——接下来将介绍这两种格式。

GIF 格式具有另外一种方便的功能——可以指定 GIF 中一种或多种颜色了解以表示透明背景——在图像的指定颜色的部分中，背景将透视出来。但需要了解的是，每个像素都是打开的或者关闭的，即不透明的或者透明的——不存在透明度，因为存在 alpha 颜色透明度格式。因此，如果将透明背景用于弯曲的转角，转角将出现像素化的效果。为了克服这个问题，需要使透明颜色尽可能地接近背景颜色(或者如果使用的是 Photoshop 软件，则可以使用磨砂功能)。

图 3-4 演示了创建 GIF 图像时不使用合适的背景所出现的像素化效果(注意图中特别显示的转角)。

图 3-4

为了使得 GIF 文件更小，可以采用一种称为 LZW 的压缩技术压缩它们。该技术扫描图像的所有行，查找具有相同颜色的连续像素。当遇到这些像素时，该技术就会指示从某个点开始向后 x 个像素具有相同的颜色。

LZW 压缩是一种无损压缩技术，因为不丢失任何数据，所以不影响图像质量(相反的一种技术称为有损压缩技术，在压缩时某些数据被抛弃，因此无法通过压缩文件恢复)。但是，当具有相同颜色的连续像素的数目较少时，该技术的压缩率将较低。因此，该技术不能很好地压缩相片图像，因为在相片中相邻像素看上去可能相同，但实际上具有非常轻微的差别。此外，如果图片使用复杂的抖动技术以获得细微差别的着色效果，则找到具有相同颜色的连续像素的机会较少，因此文件不能被很好地压缩。

有些程序能够将文件保存为交错图像。交错意味着图像中线的存储顺序与它们在图像中的出现顺序不同，浏览器将依次显示每个第 8 条线，然后填充它们之间的线。交错图像的思想是，如果拥有一个大文件并且网络速度较慢，则用户将提前看到某些内容，并且图像将逐渐变得清晰。但是，随着 Web 的连接速度的提升，交错 GIF 逐渐退出了历史舞台。

2. 动画 GIF

GIF 图像能够在一个文件中存储多个帧（或者多个图像副本），这使得 GIF 图像能够在不同版本/帧之间循环，并创建简单的动画效果。它的原理与翻书动画类似。在翻书动画中，书的每一页的绘图与前一页具有轻微的差别，因此当用户翻书时，图像看上去好像在移动。

如果动画图像中包含大面积的单调颜色，则这种技术非常适用。这种技术的压缩非常高效，因为仅需要存储每一帧中改变的像素以及这些像素的位置。但是，对于照片这种技术就不太适合，因为这样存储的图像文件会非常大。

注意：

在使用动画 GIF 时需要非常谨慎。许多站点都提供动画 GIF，从做有趣事情的卡通人物到跳跃或燃烧的项目符号。尽管第一次看到这样的页面可能会留下深刻印象，但不久就会厌倦，因为它们会降低网站的访问速度，并且让用户无法专注于实际的内容。因此，虽然动画 GIF 能够使得个人主页看上去有趣，但在大公司的站点中很少能够看到它们。如果尝试创建外观看上去专业的网站，最好是仅当动画 GIF 能够向用户提供额外的信息时才使用它们。

3. JPEG 图像

开发 JPEG 图像格式的目的是用于存储和压缩具有很多种颜色的图像，例如照片。在保存 JPEG 时，通常可以指定图像的压缩程度——具体依赖于想要得到的图像质量。压缩 JPEG 图像的过程包括抛弃人们通常无法感知的颜色数据，例如微小的颜色改变。但是当图像被压缩时，因为该图像格式抛弃了某些数据，所以这些数据将丢失，从而无法从压缩的版本恢复到最初版本——因此该压缩技术是一种有损压缩技术。

不同图像应用的压缩量是不同的，并且仅能在压缩 JPEG 之后才能判断它的压缩程度。因此压缩后文件的大小随着图像的压缩程度而改变。当保存 JPEG 图像时，用户经常会被询问使用的质量百分比；100%表明根本不压缩图片；对于照片，通常可以将质量比设置为 60%左右(但通常不要更低)。某些程序使用单词“极好”、“非常好”、“好”等取代百分比来描述图像的质量。

优秀的图像编辑程序能够在用户选择压缩的程度时，并排比较初始图像和压缩后的图像。图 3-5 中给出了当用户准备保存 JPEG 图像以用于 Web 时，Adobe Photoshop 软件在相邻的位置比较两个版本图像的情形。其中，左边的是初始图像，右边的是即将保存以用于 Web 的图像版本。

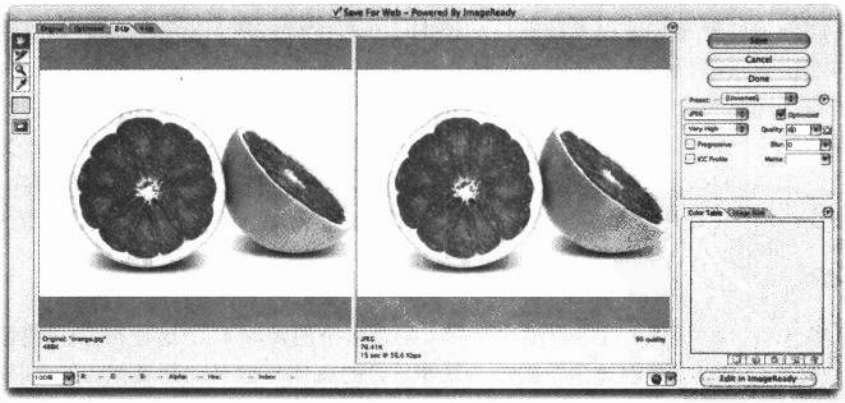


图 3-5

由于 JPEG 格式用于操作类似于照片的现实图像，因此不适合于存储具有大量单调颜色或者高对比度硬边线(例如雕刻字和素描画)的图像。随着 JPEG 图像压缩比的增加，也可能在颜色相似的位置看到条带。

当使用 Progressive JPEG 时，JPEG 也支持交错存储，从而可以首先下载具有斑驳画面的图像，然后随着图像剩余部分的加载，填充更详细的画面。这种技术最重要的用途是，它可以让用户了解正在下载的图像的大小，以及大概的下载完成程度。但是，Web 中逐渐不再使用 JPEG 图像，因为它们包含太多的细节，在实际看到想要的画面之前往往需要浏览图像的大量细节。

4. PNG 图像

可移植网络图形(Portable Network Graphics)格式是最近才出现的格式，它开发于 20 世纪 90 年代末期，原因是当时拥有 GIF 专利的公司(Unisys)决定向开发能够创建和查看 GIF 的软件公司收取使用该技术的许可费。虽然 Web 设计人员和 Web 浏览人员不受这种收费情况的影响，但制作 GIF 相关软件的公司却需要付费。

PNG 格式的用途和 GIF 图像类似，但设计人员创建它是为了解决 GIF 格式存在的一些缺陷。8 位的 PNG 和 8 位的 GIF 具有相同的限制——仅有 256 种颜色，并且每个像素要么是透明的，要么是不透明的。于是产生了增强的 PNG-24，这是 PNG 的 24 位版本，它具有如下优点：

- 图像中所使用的颜色数量不受限制，因此可以包含任何颜色而不丢失任何数据。
- 使用映射(类似于 GIF 中用于指示每个像素的颜色的查找表)为每一个像素提供不同层次的透明度，这样可以创建柔软的、平滑的边。
- 每八条线中采样一条的方法被替换为一种二维采样方法，该方法显示图像的速度比 GIF 快 8 倍。
- PNG 24 位文件可以包含灰度校正信息，这使得不同的显示器或平台中显示的颜色差别细微。

另外，所有 PNG 的压缩比例好于对应的 GIF。但是，PNG 格式的缺陷是并不是所有的浏览器都支持它。虽然在早期的浏览器版本中提供了基本支持，但更高级的功能还需要进一步实现。例如，Internet Explorer 直到版本 6 时才能正确地处理透明度。

5. 保持小的文件尺寸

在为站点保存图像时，人们通常希望以能够最大程度压缩图像的格式保存图像，以便获得较小的文件尺寸。这不仅使得页面能够更快地加载，也节省了驻留站点的费用。

通常某一种或另一种格式是显而易见的选择，经验法则如下：

- 使用 JPEG 格式存储具有许多细节的如照片般逼真的图片或具有细微色调差别的图片。
- 使用 GIF 格式存储具有单调颜色(而不是纹理颜色)和硬边线的图像，例如图表、文本或徽标。

注意：

如果不需要一些高级的功能(例如透明度)，或者如果知道网站的主要访问者使用的是最近发布的浏览器，可以考虑使用 PNG 格式。

请查看下面的两幅图像(如图 3-6 所示)——第一幅是秋天的叶子的照片，第二幅是虚构公司 Wheels 的徽标，该徽标仅使用了两种颜色——在表 3-1 中可以看到每一幅图像分别保存为 GIF 和 JPEG 时的文件尺寸(其中 JPEG 存储为 60%的图像质量)。

图 3-6

表 3-1

图　像	JPEG	GIF
叶子	54.81k	116.3
Wheels	8.26	6.063k

Wheels 的徽标具有大面积的单调颜色，而森林的照片使用许多不同的色调。因此，徽标更适合使用 GIF 或者 PNG 格式保存，而森林的照片更适合使用 JPEG 格式保存。

注意：

如果在站点中使用了大量的图像，则优秀的图像编辑软件会非常有帮助。Adobe Photoshop 是专业人员使用的最流行的图像编辑软件，尽管它的价格不菲。但是，该软件存在一个具有有限功能的版本，称为 Photoshop Elements，它包括了许多常用的功能——包括保存到 Web 中的选项。另外两种流行的图像编辑程序是 JASC 软件公司的 Paint Shop Pro 和免费的 Gimp，其中 Gimp 可以从 www.GIMP.org 下载。

如果不得不在站点中包含许多大的、复杂的照相图像，最好在页面初次加载时向用户提供图像的较小版本，然后添加一个到较大版本的链接。这些较小的图像通常称为缩略图，在图库或者包含概述信息的页面(例如新闻站点的主页和列举一些产品的页面，利用它们可以链接到一个具有更详细内容和较大图像的页面)中经常会看到它们。

注意：

当创建较小图像版本时，可以利用图像编辑程序按比例缩小图像。不要只是简单地修改<img />或<object>元素的 width 属性和 height 属性，因为用户仍然需要下载图像的完整尺寸，即使他们看到的是较小版本(完整尺寸的图像将花费更长的时间下载)。通过创建所使用的较小图像的缩略图，页面的加载速度将显著提高。

3.1.3　矢量图像

插图软件和动画软件更趋向于使用矢量格式保存图像，Web 领域中最流行的矢量图形格式是 Flash(可以在很多站点中看到 Flash 图像)。

矢量格式根据所绘制线之间的坐标存储信息，然后在线内可以指定彩色填充。由于矢量格式基于标记线上的点的坐标，因此它更容易缩放到不同的尺寸，方式是简单地增加或减少根据其标出坐标的每个点之间的距离。

默认情况下，浏览器和 XHTML 不支持任何矢量图形格式，但是主要的浏览器现在附带了查看 Flash 文件所需的 Flash 播放器。从而，Flash 成为当前在 Web 上部署矢量图形和动画的最流行的方式。虽然可以免费下载 Flash 播放器，并且浏览器也支持它，但需要了解的是，Adobe 公司向创建 Flash 文件的软件收费，并且学习如何使用该软件是全新的技能(这超出了本书的介绍范围)。

作为一种备选的矢量图形格式，W3C 开发了可伸缩矢量图形(Scalable Vector Graphics，SVG)，采用 XML 编写该格式(类似于 XHTML)，因此能够很容易地将其集成到 XHTML 中(另外，它是一种开放的标准，并不是与 Flash 一样由私人公司创建)。许多工具支持 SVG，但在编写本书时，这种格式的使用还不是非常广泛。

微软也开发了一种使用矢量图形的称为 Silverlight 的技术，以此来与 Flash 竞争。该技术也吸引了众多开发人员的兴趣，但在编写本书时，Silverlight 所需要的播放器并没有被广泛支持。

人们趋向于利用最新的<object>元素将 Flash、Silverlight 和 SVG 文件包含在页面中(或者利用 JavaScript 将它们写进页面中)。事实上，W3C 很愿意看到最终使用该元素将所有的图像包含进页面，但是目前通常使用<img>元素添加图像。

3.1.4　使用<img>元素添加图像

通常使用<img>元素将图像添加到站点中。该元素必须附带 src 属性以指明图像的来源，并且必须附带 alt 属性，alt 属性的值是图像的可选描述，在图像没有加载或者用户视力受损时使用。

例如，下面的代码行将图像 wrox_logo.gif 添加到页面中(该图像位于 images 文件夹中，images 目录位于与 XHTML 文件相同的目录中)。可以在 ch03_eg01.html 文件中找到这行代码。

```
<img src="logo.gif" alt="Wrox logo" />
```

图 3-7 给出了这幅图像在浏览器中的外观。

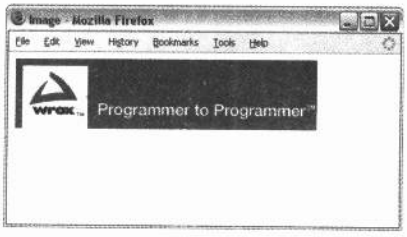


图 3-7

除了能够附带所有的通用属性和 UI 事件属性外，<img>元素还可以附带如下属性：

```
src alt align border height width hspace vspace ismap usemap longdesc name
```

1. src 属性

src 属性用于指定加载图像的 URL。

```
src="url"
```

该 URL 可以是绝对 URL 或者相对 URL，类似于在第 2 章中讨论页面的链接时所使用的 URL。该 URL 也可以使用与用于链接 XHTML 页面相同的简写符号，以指明图像位于哪个文件夹中。

在 Web 站点中为图像创建独立的目录(或文件夹)是一种不错的想法。如果站点非常庞大，甚至需要为不同类型的图像创建单独的文件夹(例如，使用一个文件夹存储用在接口中的图像，而对站点的每一个分部分别使用一个文件夹)。

注意：

通常来说，站点的图像应当始终驻留在服务器上。将图像链接到其他站点并不是优秀的实践，因为如果其他站点的所有者决定移动该图像，则您的站点的访问者将不再能够看到该图像。

2. alt 属性

alt 属性用于指定图像的备选文本，以防止用户无法看到图像(因为各种原因)。例如：

```
alt="Wrox logo"
```

它通常称为 alt 文本，这个属性的值必须能够实际地描述图像。用户无法看到图像的最常见的两个原因是：

- 因为浏览器无法正确下载文件；不能找到文件
- 因为用户视力受损，从而无法看到图像

有时图像不传达任何信息，仅用于增强页面的布局效果(例如，页面中可能具有一幅仅作为设计元素的图像，但该图像不会添加任何信息到页面中)。此时仍然需要使用 alt 属性，但是不附带任何值，如下所示：

```
alt=" "
```

3. align 属性

align 属性用于对齐页面中的图像或包含图像的元素(例如表单元格)。

```
align="right"
```

该属性可以附带表 3-2 中的某个值。

表 3-2

值	目　的
top	图像的顶部与当前文本行的顶部对齐
middle	图像的中部与当前文本的基线对齐
bottom	图像的底部与当前文本行的基线对齐(默认值)，它通常导致图像位于文本之上
left	图像与包含它的窗口或元素以及任何环绕它的文本的左边对齐
right	图像与包含它的窗口或元素以及任何环绕它的文本的右边对齐

您可能遇到过 absbottom、texttop、absmiddle 和 baseline 值，但这些值不是标准的扩展，它们可能会产生不一致的结果。

4. border 属性(逐渐淘汰)

border 属性指定图像周围的边框的宽度，单位是像素：

```
border="2"
```

如果不使用这个属性，将不会出现边框，除非将图像用作链接，此时可以指定 border="0"(具体内容可参考本章后面的“使用图像作为链接”一节)。这个属性已经被 CSS 的 border 特性所替代。

5. height 属性和 width 属性

height 属性和 width 属性分别用于指定图像的高度和宽度：

```
height="120" width="180"
```

这两个属性值的单位几乎总是像素。

从技术上来说，这两个属性的值也可以是页面或包含元素的百分比(此时数值后面需要紧跟一个百分比符号)，但这种情况很少出现，以非创建图像时的任意尺寸来显示图像通常会扭曲图像或者使图像模糊。

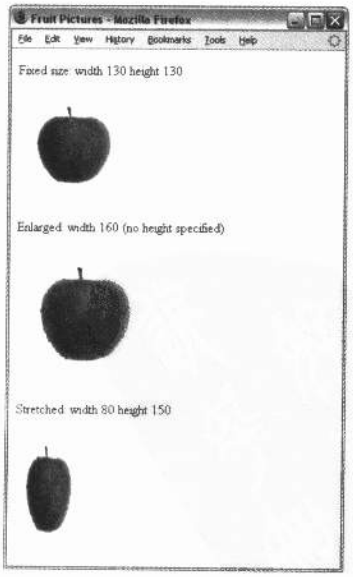


图 3-8

指定图像的大小有助于浏览器更快、更平滑地布局页面，因为这样浏览器就能够为图像分配正确的空间量，并且在图像加载完成之前生成页面剩余的部分。

如果确实希望以比存储在 Web 服务器中的图像尺寸小的尺寸显示图像，则可以只提供 height 或 width 属性中的一个值，而忽略另一个属性，此时浏览器将为图像维护正确的高宽比(宽度相对于高度)。但是图像看上去可能并不是很清晰。如果提供的宽度和高度比与图像本身的宽度和高

度比不同，则甚至可能会扭曲图像。

图 3-8 中分别给出实际尺寸的图像(最上方：130×130 像素)、放大的图像(中间：width 属性被赋值 160 像素)以及被扭曲的图像(最下方：width 属性被赋值 80 像素，height 属性被赋值 150 属性)。

下面是这个示例的代码(ch03_eg02.html)：

```
<p>Fixed size: width 130 height 130</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="130"
height="130" />
<p>Enlarged: width 160 (no height specified)</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="160" />
<p>Stretched: width 80 height 150</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="80"
height="150" />
```

如果想以比初始版本小很多的尺寸显示图像，则不能只是指定相同图像的较小尺寸，而是应该在图像操作程度中重新赋给图像不同的尺寸，以创建用于 Web 站点的较小版本。如果使用 width 属性和 height 属性缩小图像的尺寸，则用户仍然将不得不下载完整尺寸的图像，这将比加载较小的版本花费更多的时间，并且占用更多的带宽。

6. hspace 属性和 vspace 属性(逐渐淘汰)

hspace 属性和 vspace 属性用于控制图像周围的空白数量。

```
hspace="10"
vspace="14"
```

这些属性值的单位是像素，用于指定图像周围应该预留的空白，空白类似于白色边框。在 CSS 出现之前，hspace 属性和 vspace 属性特别有用，因为文本可能环绕在图像周围，除非文本和图像之间存在距离，否则难以阅读文本，并且页面看上去也不专业。图 3-9 演示了这种思想(ch03_eg03.html)。

图 3-9

这两个属性已经逐渐被淘汰，在 CSS 中可以采用 border 特性或 margin 特性实现相同的效果。

7. ismap 属性和 usemap 属性

ismap 属性和 usemap 属性用于图像映射，关于图像映射方面的内容将在本章后面的“图像映射”一节中介绍。

8. longdesc 属性

longdesc 属性用于指示某个文档(或该文档的某个部分)的 URL，该文档中包含了对图像的详细描述。

```
longdesc="../accessibility/profit_graphs.txt"
```

设计该属性是帮助无法看到图像的用户了解图像，以及提供图像中无法看到的一些额外信息。该属性的优秀使用示例是为曲线图或者图表提供文本解释。

遗憾的是，longdesc 属性没有被广泛支持。因而，另一种经常使用的可选方式是在图像旁边放置一个链接，该链接指向图像的详细描述(通常是一个到页面底部的链接)。在起始标签<a>和结束标签之间是字母 D(代表描述信息)。在图 3-10 中可以看到该方式的一个示例(ch03_eg04.html)。

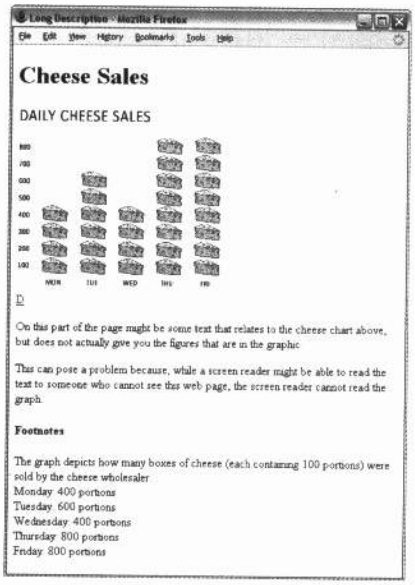


图 3-10

9. name 属性(逐渐淘汰)

name 属性用于为图像指定一个名称，以便通过脚本代码引用该图像。该属性是 id 属性的前身，已经被 id 属性替代。

```
name="image_name"
```

试一试　向文档中添加图像

在这个示例中将在一个文档中添加一些图像；这些图像是一些关于食物的明亮色彩图像，并且每一幅图像都带有相关的描述。打开文本编辑器或 Web 页面制作工具，并完成以下步骤：

(1) 首先添加 XML 声明和 DOCTYPE 声明，并添加 XHTML 文档的程序框架，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Fruit Pictures</title>
</head>
<body>
</body>
</html>
```

(2) 在页面的主体中添加如下代码，特别注意其中的<img>元素：

```
<h1>The Fruit Pictures Page</h1>
<p>The first image is an image of an apple.</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="130" height="130" />
<p>The second image is an image of an orange cut in half.</p>
<img src="images/orange.jpg" alt="Photo of orange" width="130" height="130" />
<p>The third image shows a group of bananas.</p>
<img src="images/banana.jpg" alt="Photo of bananas" width="130" height="130" />
```

(3) 将文件保存为 fruit.html，并在浏览器中打开它。最终得到的结果将类似于图 3-11 所示。

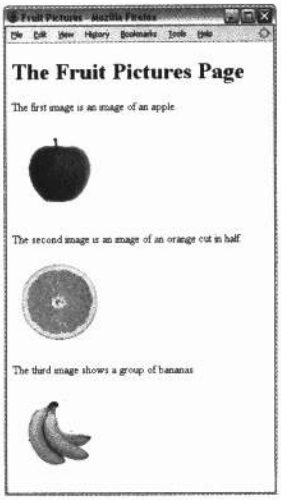


图 3-11

工作原理

您已经多次遇到这段代码中的大部分内容。但是，这段代码主要关注于<img />元素，每一个<img />元素添加一幅新图像。在这个示例中存在 3 个<img />元素。

```
<img src="images/apple.jpg" alt="Photo of red apple" width="130"
height="130" />
```

其中 src 属性指示图像的 URL。这个示例中的 URL 都是相对于 images 目录的 URL，images 目录包含在与示例页面相同的目录中。在第 2 章中提到，组织文件结构非常重要——在这里可以了解这种重要性的原因(可以清晰地表明图像应当位于站点结构中)。

编写的每一个<img />元素中都应当使用 alt 属性。如果浏览器无法加载图像，则将显示该属性的内容，并且如果访问者视力受损，该属性值也能告诉他们图像的内容。

width 属性和 height 属性用于告诉浏览器图像的显示大小。通过包含这些属性，浏览器能够更快地布局页面，因为它能够在不等待图像下载完成的情况下显示页面中的其他项。虽然可以使用这两个属性放大或缩小图像，但最好让图像具有希望使用的尺寸。如果让图像更小，应该保存它的一个新版本，而不只是使用这两个属性，这样能够节省访问者的时间和带宽。

3.2　利用<object>元素添加其他对象

W3C 在 HTML 4 中引入了<object>元素，目的是利用它在文档中嵌入所有类型的媒体，除了图像之外，还包括 MP3 文件、Flash 电影、QuickTime 电影、JavaScript 对象、Java 小程序等。W3C 甚至希望最终可以利用<object>元素在文档中包含图像。

虽然我们习惯于浏览器支持 GIF、JPEG 图像甚至最近出现的 PNG 图像，但并不习惯于其支持 MP3 声音文件、Flash 电影、QuickTime 电影或者 Java 应用程序。当然，对于后面的几种媒体，通常可以使用<object>元素在页面中包含其他一些类型的软件，以便播放或加载相应的媒体文件。例如：

- 利用 Flash 播放器播放 Flash 电影；
- 利用 Windows Media Player 播放 Windows 媒体文件；
- 可以在多种播放器中播放 MP3，包括 Flash 播放器、Windows Media 播放器、QuickTime Player。

因此，当需要在 Web 页面中嵌入声音、视频或 Java/JavaScript 应用程序时，不仅需要具有相应的文件，而且需要选择一种应用程序嵌入到页面中，用于播放/运行该文件。

在页面中包含正确的工具是一项复杂的操作，因为并不是所有的计算机都安装希望使用的应用程序。另外，每一种播放器都具有很多不同的版本。在编写本书时，在 Web 页面中嵌入移动图形和视频的最常用方式是使用 Flash(Flash 用于服务站点中的大多数视频和音频文件，例如 YouTube 站点和 MySpace 站点)。但是，虽然 Flash 通常被认为已经在世界上超过 95%的计算机上安装，但 Flash 播放器仅在最新的版本中才支持播放音频和视频。

在<object>元素引入之前，大量的元素用于在页面中插入多媒体对象，例如<applet>、

<embed>和<bgsound>元素，但是这些元素都逐渐被淘汰(附录 I 中包含对这些元素的介绍)。

<object>元素最初由微软引入以支持其 Active X 技术；但是，不久它就用于在 Web 页面中嵌入所有类型的对象。为了在页面中嵌入一个对象，需要指定以下内容：

- 用于显示或播放对象的代码的位置(有时称为对象的实现)
- 显示或播放的实际数据(例如电影、音频文件或程序)
- 对象在运行时所需要的一些额外值

前面两点内容在使用<object>元素时添加，而额外的值将在<param>元素中提供，该元素可以是<object>元素的子元素。

虽然<object>元素可以包含子元素<param>，但<object>元素的其他内容应当仅在浏览器不能生成对象时才会显示：

```
<object>
Your browser does not appear to support the format used in this film clip,
for more details please look <a href="../help/video.htm">here</a>
</object>
```

可以按照偏爱的查看顺序嵌入多个<object>元素，因此可以在首选的一个<object>元素中放置可选的对象格式。如果这些方式都不被支持，浏览器将显示文本内容。为了支持较老的或者不同版本的浏览器，可以添加一些较老的代码，例如在<object>元素中添加已经逐渐淘汰的<embed>元素和<applet>元素。

顺便提及的是，当在页面中添加视频或音频时，最好提供一个用于关闭音乐的按钮。

3.2.1 <object>元素的属性

<object>元素可以附带所有通用属性、UI 事件属性和下面的一些属性：

```
archive border classid codebase codetype data declare height width hspace
vspace name standby tabindex usemap
```

下面分别介绍这些属性，其中最常用的是 classid 属性(后面将介绍该属性)、type 属性和 id 属性(第 1 章中已经讨论过该属性)。

1. archive 属性

archive 属性通常用于基于 Java 的应用程序。它允许预先加载档案文件中的一些类或者对象集合——例如，当一个类依赖于其他类时——并且通常用于加快程序执行速度。该属性的值是一个或多个指向资源(如果存在多个资源，则资源之间以空格隔开)的 URL。

2. border 属性(逐渐淘汰)

border 属性指定显示在对象周围的边框的宽度；该属性值的单位是像素。但是，该属性将逐渐被淘汰，应该使用 CSS 中的 border 特性作为代替。

3. classid 属性

classid 属性用于指定对象的实现。当需要在页面中包含需要加载的 Flash 或 QuickTime 文件以及相应插件时，该属性的值指示播放或运行这些文件所需要的应用程序。当工作于 Java 环境中时，这个属性的值很可能是希望包含的 Java 类。

下面是 classid 属性的示例，该示例在页面中嵌入播放器以播放 QuickTime 电影：

```
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
```

4. codebase 属性

codebase 属性为<object>元素中的相对 URL 提供一个可选的基 URL。如果没有指定这个属性，则将使用页面所在的文件夹。例如，如果工作于 Java 环境中，则指定该属性如下：

```
codebase="http://www.example.org/javaclasses/"
```

但是，当遇到 QuickTime 电影或 Flash 文件时，IE 使用该属性指定播放或运行这些文件所需要的程序的位置。例如，可以从如下位置下载 QuickTime ActiveX 控件(播放 QuickTime 电影所需要的控件)：

```
codebase="http://www.apple.com/qtactivex/qtplugin.cab"
```

该属性也可以标识需要下载的文件的版本。如果对象没有安装在加载页面的机器中，浏览器将进入 URL 指定的位置获取它(然而在开始下载之前将对用户显示一个警告)。

5. codetype 属性

codetype 属性指定浏览器所期望的 MIME 类型。仅在已经指定 classid 属性后才会使用该属性。例如，如果工作于 Java 环境中，则指定该属性如下：

```
codetype="application/java"
```

如果希望在页面中嵌入 QuickTime 电影，可以使用如下值：

```
codetype="video/quicktime"
```

浏览器可以使用 codetype 属性跳过它不支持的媒体类型，而不必下载一些不必要的对象。附录 H 中介绍了一些 MIME 类型。

6. declare 属性

declare 属性用于声明一个对象，但是不实例化它。该属性可用于对象的向前引用，因此仅当用于创建到其他对象的交叉引用或者用作其他对象的参数时才加载这些对象。

declare 属性是一个布尔类型的属性，虽然在 HTML 中它不需要具有值，但在 XHTML 中所有的属性都需要具有值，因此使用方式如下：

```
declare="declare"
```

7. data 属性

如果对象具有一个文件需要处理或播放，则 data 属性用于指定该文件的 URL。例如，下面是到一个 MP3 文件的 URL：

```
data="http://www.example.com/mp3s/newsong.mp3"
```

该属性的值可以是一个相对 URL，并且是相对于 codebase 属性中所提供的值(如果指定该属性)；否则它将是相对于页面本身所在的位置。

8. height 属性和 width 属性

height 属性和 width 属性分别用于指定对象的高度和宽度，属性值的单位是像素或者包含元素的百分比。可以将这两个属性视为类似于<img>元素的 height 属性和 width 属性。使用这两个属性将使得页面的加载速度更快，因为浏览器可以在不完全加载对象的情况下开始布局页面剩余的部分。

9. hspace 属性和 vspace 属性(逐渐淘汰)

hspace 属性和 vspace 属性用于指定显示在对象周围的空白数量，类似于<img>元素的相应属性。它们已经被 CSS 中的 margin 特性和 border 特性所替代。

10. name 属性(逐渐淘汰)

name 属性提供一个名称，可以使用该名称访问对象，特别是用于脚本中。在 XHTML 中，它已经被 id 属性所替代。

11. standby 属性

standby 属性指定一个文本字符串，该字符串在对象加载时使用：

```
standby="Trailer for Harry Potter 27 is loading"
```

它的值应当是正在加载对象的有意义的描述。

12. tabindex 属性

tabindex 属性用于指定页面中对象的焦点移动顺序。第 5 章中将讨论焦点移动顺序。

13. usemap 属性

usemap 属性用于指明对象是一个图像映射，该图像映射中包含了一些作为超链接的已定义区域。它的值是用于对象的映射文件。该属性可以是一个到外部文件的完整 URL，或者是一个到内联<mapElement>元素的 mapName 属性值的引用。具体内容请查看本章后面的“图像映射”一节。

3.2.2　<param>元素

<param>元素用于向对象传递参数。对象所需要的参数类型取决于对象完成的任务，如果对象的任务是加载 MP3 播放器到页面中，则需要指定 MP3 文件的位置。另外，如果需要添加一个视频到页面中，则对象用于表明是否在页面加载时自动播放视频，或者是否等待用户单击播放按钮后再开始播放。

除了通用属性和基本事件属性之外，<param>元素可以附带如下属性：

```
name type value valuetype
```

1. name 属性和 value 属性

name 属性和 value 属性充当一个“名/值”对(相当类似于属性本身的含义)。name 属性提供传递给应用程序的参数的名称，而 value 属性给出该参数的值。

下面是取自于一个 QuickTime 电影的两个示例。第一个参数指示需要加载以播放的文件的来源，第二个参数指示电影应该在加载时自动开始播放(不需要用户亲自启动它)：

```
<param name="src" value="movieTrailer.mov" />
<param name="autoplay" value="true" />
```

如果处理的是一个 Java 小程序，可使用 name 和 value 属性传递值到方法中。

2. valuetype 属性

如果对象接受参数，则 valuetype 属性指示参数的类型，例如文件、URL 或另一个对象。表 3-3 给出了该属性的可能值。

表 3-3

值	目　的
data	参数值是一个简单的字符串——这是默认值
Ref	参数值是一个 URL
object	参数值是另一个对象

3. type 属性

如果传递给对象的参数是一个字符串，则不需要指定 type 属性。但是，如果传递一个 URL 或对象，则应当使用 type 属性。该属性的作用是告诉对象即将传递的参数的 MIME 类型。

例如，假设希望指定正在传递一个 Java 对象作为参数，则 type 属性如下所示：

```
type="application/java"
```

3.2.3　在页面中添加 Flash 电影

下面查看一个示例，该示例使用<object>元素将一个 Flash 电影添加到页面中。Adobe

的 Flash 软件(可以用于创建 Flash 动画)具有一项非常有用的功能，它可以创建<object>代码，从而方便用户将电影添加到页面中。当完成“发布”文件的过程时(这基本意味着为 Web 准备好了文件)，该软件创建一个示例文件，可使用这个示例文件在浏览器中显示文件。

在下面的示例中，<object>元素用于在 Web 页面中包含 Flash 播放器——classid 属性指定在页面中需要包含 Flash 播放器，而 width 属性和 height 属性指定 Flash 文件的尺寸。然后，在<object>元素中包含几个<param>元素，它们给出了关于 Flash 播放器的进一步信息(ch03_eg05.html)。

```
    <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="300"
    height="200"
    codebabse="http://download.macromedia.com/pub/shockwave/cabs/flash/swfl
ash.cab">
      <param name="movie" value="motion/flash_sample.swf">
      <param name="play" value="true">
      <param name="loop" value="false">
      <embed src="motion/flash_sample.swf" width="300" height="200" play="true"
    loop="false" QUALITY="best" menu="false" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_
Prod_Version
    =ShockwaveFlash">
       </embed>
    </object>
```

<param>元素影响对象(这里是 Flash 播放器)的行为方式，因为<object>元素用于包含不同类型的对象；<param>元素特定于每一个对象。

在这个示例中，带有指定值的参数具有如下意义：

- movie 指定了将加载到 Flash 播放器中的 Flash 电影的位置。
- play 指定了页面加载时是否默认播放电影。
- loop 指定在电影播放结束时是否循环播放。

在这个示例中还有<embed>元素，在附录 I 中包含对该元素的详细介绍，该附录中给出了逐渐淘汰的非标准代码。<embed>元素在浏览器的早期版本中被引入，用于插入各种插件(插件是一些小程序，它们不是浏览器的组成部分)，但是最近的大多数浏览器只能识别<object>元素，而忽略<embed>元素。<enmed>元素包含在图 3-12 中只是因为仍然会经常看到同时使用这两个元素的情况。

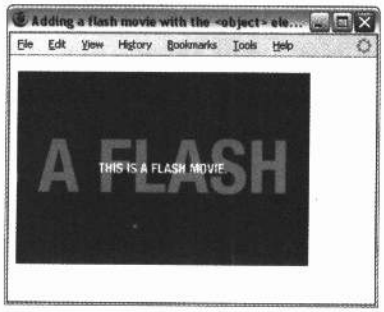


图 3-12

虽然<object>元素经常用于在页面中添加 Flash 电影，但您将看到许多站点通过使用 JavaScript 的 SWFObject 向页面中添加 Flash 电影——这种技术不仅检查浏览器中是否安装了所需要的 Flash 版本，而且如果用户不具有所需要的 Flash 版本，它能够显示提示文本。可以从 http://blog.deconcept.com/swfobject/中了解关于 SWFObject 的更多知识并下载相应的 JavaScript 文件。

3.3　使用图像作为链接

将图像转换为链接是很容易的操作；类似于上一章中介绍的将文本放置在起始标签<a>和结束标签之间，可以将图像放置在这两个标签中。图像通常用于创建图形按钮或到其他页面的链接，如下所示(ch03_eg06.html)。

```
    <a href="../index.html" title="Click here to return to the home page">
    <img src="images/banana.jpg" width="130" height="130" alt="Banana"
border="0"
    /></a>
```

注意其中逐渐淘汰的 border 属性的使用。当在<a>元素中使用图像时，在 Windows 操作系统的 IE 浏览器中，图像将获得一个边框，如图 3-13 所示。

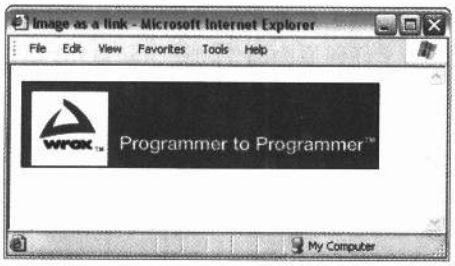


图 3-13

这个边框可能非常难看，因此可以指定 border 属性为 0 像素宽，或者将 CSS 的<img />元素的 border 特性设置为 0(第 7 章中将介绍这种方式)。

同样需要注意，这个示例中的图像并不是一个关于链接的非常好的示例，因为它没有表明链接将转向何处。如果使用图像作为链接，应当阐明用户单击链接时将会发生的情况。

注意：

当希望使用<object>元素包含图像时，将图像放置在<a>元素中的技术也将有效，但当使用<object>元素包含其他对象(例如 Flash 电影、QuickTime 播放器、Windows 媒体播放器)时，该技术则不一定有效。

3.4 图像映射

图像映射允许用户指定对应单个图像的不同区域的多个链接，从而当用户单击图像的不同部分时，可以到达不同的页面。存在如下两种类型的图像映射：

- 服务器端图像映射
- 客户端图像映射

两者的不同点在于决定链接目的页面的代码的执行位置不同。对于客户端图像映射，浏览器根据用户单击的位置指示链接的目的页面；而对于服务器端图像映射，浏览器将用户单击位置的坐标发送给服务器，然后由服务器端的脚本文件处理这些坐标以决定用户应该转向哪个页面。

图 3-14 给出一幅 GIF 图像，它将被转换为图像映射。当用户单击圆时，将看到图库里面的内容；当用户单击花园时，将看到关于雕塑花园的页面；当他们单击画室时，将看到关于画室的页面。其中每一部分称为一个可单击的热点。

当图像需要被划分为不规则的形状时(例如地图)，图像映射就非常有帮助。但是，如果图像被划分为网格，最好手动将图像划分为多个块，并将它们放置到一个表中(下一章中将介绍表)。

热点的面积不能太小；否则用户很难正确选择想要的区域。如果发生这种情况，他们将很快感到挫折，会离开您的站点。对于定位电动控制装置困难的用户来说，图像映射也是非常不便的。因此，如果出于任何原因而使用图像映射作为站点的主要导航方法，应当在页面底部提供相应的文本链接(在 alt 文本中指示)。

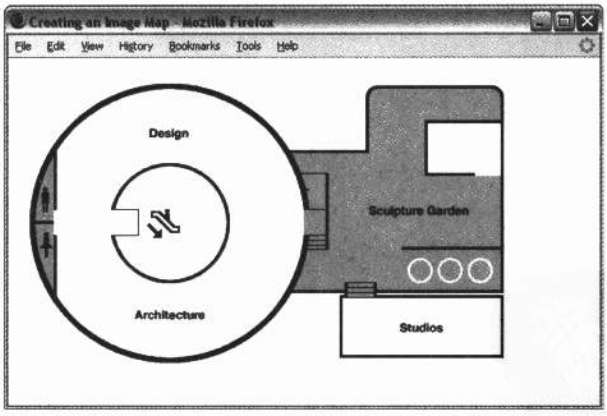


图 3-14

3.4.1 服务器端图像映射

对于服务器端图像，<img>元素(位于一个<a>元素中)附带一个特殊的 ismap 属性，ismap 属性告诉浏览器在用户单击图像映射时向服务器发送表示鼠标指针所在位置的 x，y 坐标。然后使用服务器中的一个脚本基于传递给它的坐标确定用户应当跳转到的页面。

例如，请查看下面的链接，其中<img>元素附带了 ismap 属性，该属性的值为 ismap(在 HTML 中 ismap 属性不需要具有值；但是，在 XHTML 中所有属性都必须有值，因此在 XHTML 中采用该属性的名称作为值以使其有效)：

```
<a href="../location/map.aspx"><img src="../images/states.gif" alt="map
of US States" border="0" ismap="ismap" /></a>
```

现在，如果用户单击图像中从<0,0>到<50,75>像素之间的位置，浏览器将向服务器发送如下 URL 信息：

```
http://www.example.org/location/map.aspx?50,75
```

可以看到附加在 URL 末尾的坐标在<a>元素中指定。

对于服务器端图像映射，在服务器中需要有一个脚本、映射文件或者应用程序来处理坐标，以便知道用户应当跳转到哪个页面。图像映射的实现方法具体依赖于运行在其上的服务器的类型。

由于服务器端图像映射由服务器处理，因此在 HTML 或 XHTML 中没有它们的实现。遗憾的是，本书中没有空间为每一种不同的平台介绍它们的实现方法。如果想了解服务器端图像映射，应当购买一本介绍服务器端脚本的书籍，例如关于 ASP.Net、PHP、CGI 或 JSP 的书籍。可以在 Wrox.com 中查看关于这个主题的书籍列表。

3.4.2　客户端图像映射

由于服务器端图像映射依赖于服务器技术，因而引入了另一种作用于浏览器的技术，即客户端图像映射。客户端图像映射使用 XHTML 页面中的代码来指示图像的哪一部分应当链接到哪个页面。因为划分图像各个部分的代码位于浏览器中，所以浏览器可以向用户提供额外的信息，例如在状态栏中显示一个 URL，或者当鼠标悬浮在图像上时显示一个工具提示。

存在两种创建客户端图像映射的方法：在<img>元素中使用<map>元素和<area>元素，以及在<object>元素中使用<map>元素，其中后者是最近经常使用的方法。

1. 采用<map>元素和<area>元素的客户端图像映射

这是较早的创建图像映射的方法，浏览器已经支持该方法很长时间，可以追溯到 Netscape 4 和 IE 4 出现的年代。

通常使用<img />元素将组成映射的图像插入到页面中，不同之处是该元素附带了一个称为 usemap 的额外属性。usemap 属性的值是<map>元素中 name 属性的值（下面将介绍 name 属性），前面添加一个英镑符号(#)。

<map>元素为图像实际地创建映射，并且它通常紧跟在<img />元素之后。它是<area>元素的容器，<area>元素实际地定义可单击的热点。<map>元素仅附带一个属性，即 name 属性，该属性用于标识映射的名称。这就是<img />元素知道使用哪个<map>元素的方式。

<area>元素指定定义每一个可单击热点的边界的形状和坐标。下面是用于图 3-14 中图像的图像映射示例(ch03_eg06.html)。

```
<img src="gallery_map.gif" alt="Gallery Map" width="500" height="300"
border="0" usemap="#gallery" />
<map name="gallery">
  <area shape="circle" coords="154,150,59" href="foyer.html" target="_self"
      alt="Foyer" >
  <area shape="poly"
coords="272,79,351,79,351,15,486,15,486,218,272,218,292,
      166,292,136,270,76" href="sculpture_garden.html" target="_self"
      alt="Sculpture garden" />
  <area shape="rect" coords="325,224,488,286" href="workshop.html"
      target="_self" alt="Artists workshops" />
</map>
```

<img />元素中 usemap 属性的值是#gallery，并且该值也用于<map>元素中。然后<area>元素实际定义了图像中可单击的每一部分。

如果存在两个区域互相重叠，则代码中首先出现的区域将具有较高的优先级。

<area>元素能够附带的属性与<a>元素类似。下面将介绍一些与图像映射相关的属性；其他属性请参考本章前面的“使用<img>元素添加图像”一节。

```
accesskey alt shape coords href nohref target tabindex taborder notab
```

shape 属性

shape 属性的值实际上影响了浏览器使用坐标(由 coords 属性指定)的方式，因此它是必需的属性。如果不指定 shape 属性，则 IE 通常假设该区域是矩形。

表 3-4 列举了 shape 属性的可能值。

表 3-4

值	创建的形状
default	整个图像都没有被定义在一个区域中(应该最后指定该值)
rectangle 或 rect	矩形
polygon 或 poly	多边形
circle 或 circ	圆形

最好使用值的缩写形式，因为它们能够被较老的浏览器更好地支持。如果希望表明图像中没有被<area>元素指示的任何部分，应当在最后使用值 default 指定——它类似于针对图像剩余部分的捕获所有代码块。

coords 属性

coords 属性用于指定可单击热点的区域。指定的坐标的数量依赖于所创建的形状(已经在 shape 属性中指定形状)。

- 一个矩形包含 4 个坐标，其中前两个坐标表示矩形左上角，后两个坐标表示右下角。
- 一个圆形包含 3 个坐标，前两个坐标是圆形的中心，而第三个坐标是半径，单位是像素。

- 多边形的每一个顶点由两个坐标值表示，因此一个三角形将包含 6 个坐标值，一个五角形将包含 10 个坐标值，以此类推。不需要在最后再次指定第一个坐标，因为形状将自动闭合。

有些 Web 制作和图像编辑程序能够帮助用户计算出图像映射的坐标；这些程序提供一种工具，利用这种工具可以选择希望转变为映射的区域，并使用这些形状创建坐标。图 3-15 给出了 Dreamweaver 的图像映射工具——因为每一种程序都是不同的，所以应当查看所使用程序的帮助文件以了解如何创建图像映射。

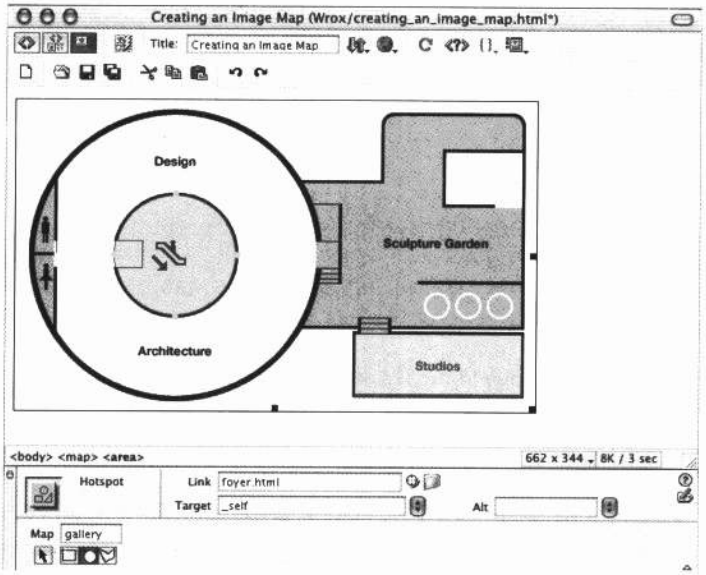


图 3-15

href 属性和 nohref 属性

href 属性的作用类似于<a>元素中的 href 属性；它的值是用户单击图像某个区域时希望加载的页面的 URL。

如果没有 href 属性，则必须使用 nohref 属性以表明单击该区域不会到达任何页面，它的值为 nohref。

alt 属性

alt 属性给出图像某个区域的备选文本，它的作用类似于<img />元素中的 alt 属性。当用户使用鼠标在该区域上移动时，该属性将实际地重写为图像指定的 alt 文本。

target 属性

target 属性指定页面将被加载到哪一个框架或窗口中，它的可能值与<a>元素的 target 属性相同。

tabindex 属性

tabindex 属性用于指定用户单击 Tab 键时页面上各个项的切换顺序，它的值是从 1 到

32 767 之间的数值。在第 5 章中将详细讨论该属性。

2. 使用<object>元素创建客户端图像映射

从 HTML 4 开始提倡使用<object>元素在文档中添加图像映射，而不是使用<map>元素(但在 Strict XHTML 1.0 中仍然可以使用<map>元素)。<object>元素采用不同的方式创建图像映射。

<object>元素可以附带 usemap 属性(该属性的值是<map>元素中 name 属性的值，前面添加一个英镑符号)。在<object>元素内部使用的是熟悉的<map>元素，该元素附带 name 属性。但是，在<map>元素内部是标准的<a>元素。

此处使用<a>元素有助于解释为什么它能够附带 shape 和 coords 这样的属性。

```
<object data="gallery_map.gif" type="image/gif" alt="Gallery Map" width="500"
   height="300" border="0" usemap="#gallery" />
<map name="gallery">
  <a shape="circle" coords="154,150,59" href="foyer.html"
  target="_self">Foyer</a>
  <a shape="poly" coords="272,79,351,79,351,15,486,15,486,218,272,218,292,
  166,292,136,270,76" href="sculpture_garden.html" target="_self">Sculpture
garden</a>
  <a shape="rect" coords="325,224,488,286" href="workshop.html"
     target="_self">
     Artists workshops</a>
</map>
```

应当将 alt 文本(或者链接的描述)放置在<a>元素中，而不是使用 alt 属性。

但是，对这种图像映射创建方式的支持非常有限。因此，当前最好坚持使用传统的方法。

3.5 本章小结

本章中介绍了如何在页面中添加图像和其他多媒体对象，从而使页面看上去更加生动。

本章介绍了用于 Web 的不同类型的图像。虽然图像能够使页面更加生动，但要注意它们的尺寸。如果页面具有太多的图像或者图像太大，都将使页面的加载速度显著变慢。因此，需要选择好图像的格式，一方面能够很好地压缩图像，另一方面保持它的质量。对于具有大面积单调颜色的图像，GIF 格式是一种较好的选择，而 JPEG 格式是照片图像和具有相同颜色渐变的图形的优秀选择。如果需要在页面中使用大量图像，则购买优秀的图像编辑软件有助于以适当的格式保存图像。

虽然<img />元素是当今在文档中包含图像的最常用方法，但是本章也介绍了<object>元素在将来会更多地使用该元素。<object>元素已经广泛用于在页面中嵌入其他类型的文件和代码，从 Flash 或 QuickTime 电影到 Java 小程序和 JavaScript 对象。

最后，本章介绍了如何将图像划分为可单击的热点，从而将图像的不同部分转换为独立的链接。在一幅图像中创建独立链接的另外一种方式是手动将其分成多个部分，并将每

一部分放置到表的不同单元格中；在第 4 章中将更详细地介绍表。

3.6　练习

所有练习的答案都在附录 A 中给出。

1. 在下面的示例中，添加一些图标图像，它们分别表示一本日记、一部照相机和一张报纸。在第 3 章下载代码的 images 文件夹中提供了所有这些图像。

```
<h1>Icons</h1>
<p>Here is an icon used to represent a diary.</p>
<img src="images/diary.gif" alt="Diary" width="150" height="120" /><br />

<p>Here is an icon used to represent a picture.</p>
Camera image goes here<br />

<p>Here is an icon used to represent a news item.</p>
Newspaper image goes here<br />
```

完成的页面应当类似于图 3-16。

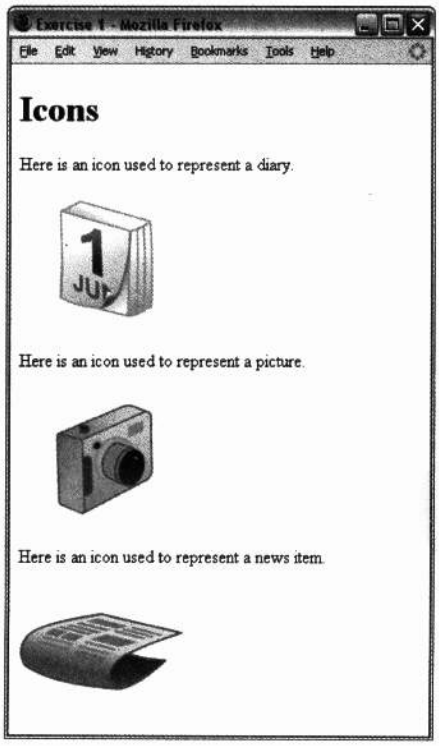


图 3-16

2. 查看图 3-17 和图 3-18 中的图像，确定如果以 JPEG 或 GIF 格式保存它们，是否获得较小的文件尺寸以及较好的图像质量。

图 3-17

图 3-18

第4章

表

表通常用于显示各种样式的数据，例如时间表、财政报告和体育运动结果。因此，当需要以行和列的形式显示信息时，可以使用本章中介绍的标记来创建表。

本章首先介绍用于创建各种表的基本元素。然后介绍表的一些高级功能，例如表标题、题头以及一些较复杂的表布局。本章也将介绍一些用于控制表外观的标记，这些标记已经被逐渐淘汰。尽管更提倡使用 CSS 控制页面的外观，但是有时也需要使用较老的标记，以便较老的浏览器能够按照您的意图正确显示页面。本章最后讨论与表相关的一些可访问性问题，因为它们能够带来明显的效果，特别是针对视力受损的用户。

4.1 表简介

为了操作表，需要首先将其想像为网格。类似于电子数据表，表由行和列组成，如图 4-1 所示。

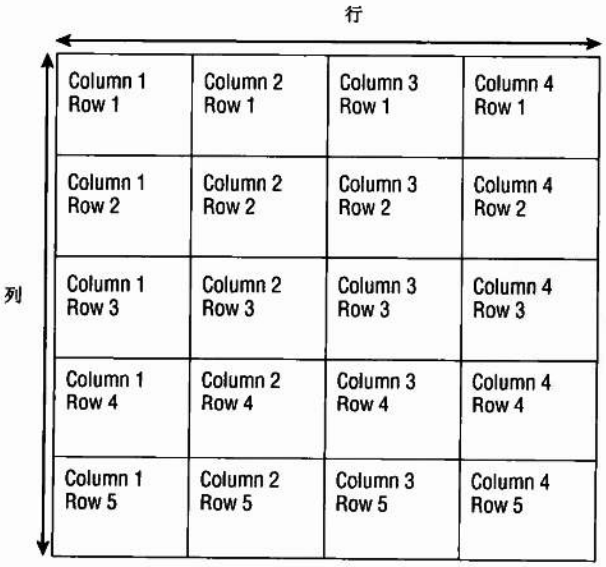


图 4-1

在图 4-1 中可以看到由一些矩形组成的网格，每一个矩形称为一个单元格。一行由从左到右相同行中的单元格集合组成，而一列由从上到下的一行单元格集合组成。

到目前为止您已经了解到，XHTML 中元素的名称趋向于表示它们所包含的标记的类型。因此在 XHTML 中使用<table>元素来创建表是常见的情况。

在<table>元素内，通过逐行的方式创建表。一行包含在一个<tr>元素内——该元素代表表的行。然后在每个行元素内使用<td>元素编写每个单元格——<td>元素代表表的数据。

下面是一个非常基本的表示例(ch04_eg01.html)。

```
<table border="1">
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
```

本书始终仔细地缩进表代码，以便读者更容易看出表的结构，并且在新的代码行中开始编写每一行和单元格。虽然这只是一种个人偏好，但是在表中如果遗漏一个结束标签或一个尖括号，都会导致整个表无法正确地显示。当查看代码时(特别是当查看本章后面介绍的复杂嵌套表时)，缩进有助于跟踪所处的位置。

在 Web 浏览器中，该示例显示非常基本的表，但是它给出了表的组成方式的思想。可以在图 4-2 中看到该示例的结果。

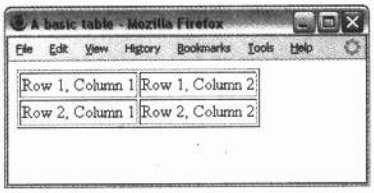


图 4-2

所有的表都将遵循这个基本结构，但是存在一些其他元素和属性可以用于控制表的外观。如果一行或一列需要包含题头，可以使用<th>元素代替表数据或<td>元素。默认情况下，大多数浏览器以粗体文本呈现<th>元素内的内容。

说明：

每个单元格必须由一个<td>元素或者<th>元素来表示，以便即使该元素为空时也能够正确显示表。

下面是一个稍微复杂一些的表示例，该表包含了题头(ch04_eg02.html)。

```
<table border="1">
  <tr>
    <th></th>
    <th>Outgoings ({$})</th>
    <th>Receipts ({$})</th>
    <th>Profit ({$})</th>
  </tr>
  <tr>
    <th>Quarter 1 (Jan-Mar)</th>
    <td>11200.00</td>
    <td>21800.00</td>
    <td><b>10600.00</b></td>
  </tr>
  <tr>
    <th>Quarter 2 (Apr-Jun)</th>
    <td>11700.00</td>
    <td>22500.00</td>
    <td><b>10800.00</b></td>
  </tr>
  <tr>
    <th>Quarter 3 (Jul - Sep)</th>
    <td>11650.00</td>
    <td>22100.00</td>
    <td><b>10450.00</b></td>
  </tr>
  <tr>
    <th>Quarter 4 (Oct - Dec)</th>
    <td>11850.00</td>
    <td>22900.00</td>
    <td><b>11050.00</b></td>
  </tr>
</table>
```

注意：

表可能占用大量的空间，使得文档变长，但是具备清晰格式的表将使得代码更容易被理解。在编写代码时无论如何熟悉它，如果以良好的结构编写代码，那么在一年之后重新阅读它时，您将会感到很高兴。

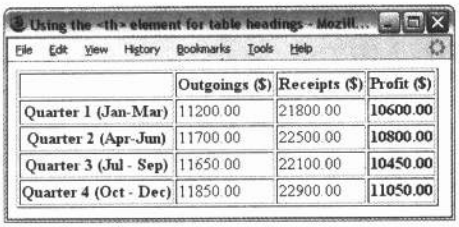
这个示例中的表给出了一个小型公司的财政总结。从顶部的第一行开始，可以看到表示收入、支出和利润的 3 个题头。第一个单元格实际上是空的，但是仍然必须在代码中为其添加一个<td>元素或<th>元素；否则第一行比其他行将少一个单元格，并且列的排列也不协调。

在每一行中，第一个表单元格也是表题头单元格(<th>)，用于指示是哪一个季度的结果。然后每一行剩余 3 个单元格包含了表数据，因此它们包含在<td>元素中。

表明利润的数字包含在<b>元素中，从而以粗体字型显示利润数字。事实上，这个示例演示了任何单元格都可以包含所有样式的标记。在表中放置标记的唯一约束是它必须嵌套在表单元格元素(<td>元素或<th>元素)中。不能将元素的起始标签放置在一个单元格内，

而将其结束标签放置在该单元格的外部——反之亦然。

图 4-3 中给出了这个表在 Web 浏览器中的外观。



	Outgoings ($)	Receipts ($)	Profit ($)
Quarter 1 (Jan-Mar)	11200.00	21800.00	10600.00
Quarter 2 (Apr-Jun)	11700.00	22500.00	10800.00
Quarter 3 (Jul - Sep)	11650.00	22100.00	10450.00
Quarter 4 (Oct - Dec)	11850.00	22900.00	11050.00

图 4-3

注意：

很多人在创建表时实际上不愿意使用<th>元素，而是对每一个单元格(包括题头)都使用<td>元素。但是，<th>元素有助于改进表的可访问性，在本章末尾将会对此进行介绍。如果出于某种原因已经存在该元素，那么使用它也是不错的想法。当使用 CSS 赋予表样式时，<th>元素也有助于以不同的方式显示单元格。

4.2 基本表元素和属性

现在您已经了解如何创建基本的表，本节将更详细地描述上面提到的几种元素，并且介绍它们能够附带的属性。利用这些属性，可以创建更复杂的表布局。

4.2.1 创建表的<table>元素

<table>元素是所有表的包含元素，它可以附带如下属性：

- 所有的通用属性
- 用于脚本的基本事件属性

<table>元素可以附带如下逐渐淘汰的属性。尽管这些属性逐渐被淘汰，但目前仍然会看到大量使用它们：

```
align bgcolor border cellpadding cellspacing dir frame rules summary width
```

1. align 属性(逐渐淘汰)

尽管 align 属性逐渐被淘汰，但它仍然频繁用于表中。当用于<table>元素时，它表明表是否与页面的左边(默认情况)、右边或中间对齐(当用于单元格时，它用于对齐单元格的内容，后面将对此进行介绍)。该属性的语法如下所示：

```
align="center"
```

如果表包含在另外一个元素中，则 align 属性将表明表是否应当与该元素的左边、右边或中间对齐。

如果表对齐，则文本应当紧跟其后并环绕它。例如，下面是一个左对齐表，后面紧跟着一些文本(ch04_eg03.html)：

```
<table border="1" align="left">
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit...
```

文本应该紧跟表并环绕它，如图 4-4 中的第一个表所示。

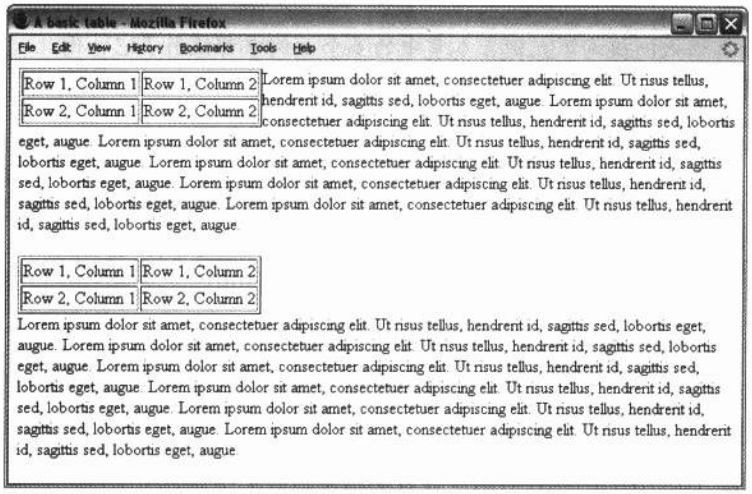


图 4-4

为了防止这种情况发生，可以在表后面放置一个换行符，并添加 clear 属性(<br clear="left" />)，结果如图 4-4 中的第二个表所示。

```
</table>
<br clear="left" />
Lorem ipsum dolor sit amet, consectetuer adipiscing elit...
```

clear 属性用于指示浏览器如何显示换行符后面的文本。如果该属性的值是 left，则表明仅当浏览器窗口的左边页边空白没有内容时，文本才能开始显示(如果文本包含在一个元素中，则需要该元素的左边页边空白没有内容时才能显示文本)。clear 属性能够采用的值包括 all、left、right、none；附录 I 中更详细地介绍 clear 属性，虽然 clear 属性已被 CSS 中的 clear 特性所替代，两者完成相同的工作，并且后者是更好的选择。

2. bgcolor 属性(逐渐淘汰)

bgcolor 属性用于设置表的背景颜色。这个属性的值应该是一个 6 位数字的代码(称为十六进制代码)或一种颜色名。附录 D 中详细介绍了在 XHTML 和 CSS 中指定颜色的方式。该属性的语法如下：

```
bgcolor="#rrggbb"
```

3. border 属性(逐渐淘汰)

如果使用 border 属性，则在表的周围和每一个单元格的周围创建边框。这个属性的值是表的外部边框的宽度，单位为像素。如果将该属性的值设置为 0，或者没有使用这个属性，则表的周围和任何单元格的周围将没有边框。

```
border="0"
```

说明：

虽然这个属性逐渐被淘汰，但在本章中的多个示例中都使用了它，因此可以清楚地看到每一个表单元格的边。

4. cellpadding 属性(逐渐淘汰)

cellpadding 属性用于创建单元格的边和它的内容之间的间隔。这个属性的值可以是单元格的每条边中的空间量或补白，单位是像素或者百分比值(表宽度的百分比)。

可以想像的是，如果两个单元格都具有文字，而单元格的边和文字之间没有间隔，则内容将难于阅读。

```
cellpadding="5" 或 cellpadding="2%"
```

5. cellspacing 属性(逐渐淘汰)

cellspacing 属性用于在每个单元格的边框之间创建空格。这个属性的值可以是想要创建的单元格之间的空间量，单位为像素或者百分比值(表宽度的百分比)。

```
cellspacing="6" 或 cellspacing="2%"
```

6. dir 属性

dir 属性用于指定表中文本的方向。该属性的可能值包括 ltr(文本从左到右)和 rtl(文本从右到左，用于希伯来语和阿拉伯语这样的语言)：

```
dir="rtl"
```

如果在<table>元素中使用了一个具有 rtl 值的 dir 属性，则单元格将从右边开始出现，每一个后续的单元格放置在该单元格的左边。

7. frame 属性(逐渐淘汰)

frame 属性用于控制整个表最外部边框(称为该表的框架)的外观，它的控制程度高于

border 属性。如果同时使用 frame 属性和 border 属性，则 frame 属性具有更高的优先级。该属性的语法如下：

```
frame="frameType"
```

表 4-1 中给出了 frameType 的可能值。

表 4-1

值	目 的
void	没有外部边框(默认值)
above	仅具有边框
below	仅底部具有边框
hsides	顶部和底部具有边框
lhs	表左边具有边框
rhs	表右边具有边框
vsides	表的左边和右边具有边框
box	所有的边都具有边框
border	所有的边都具有边框

常用的浏览器对 frame 属性的支持并不是很好，因此最好使用 CSS 中的相应特性。

8. rules 属性(逐渐淘汰)

rules 属性用于指示表的哪些内边框将显示，例如行和列。该属性的语法如下所示；默认值是 none。

```
rules="ruleType"
```

表 4-2 中给出了 ruleType 的可能值。

表 4-2

值	目 的
none	没有内部边框(默认值)
groups	在所有的表组之间显示内部边框(由<thead>、<tbody>、<tfoot>和<colgroup>元素创建组)
rows	在每一行之间显示水平边框
cols	在每一列之间显示垂直边框
all	在每一行和每一列之间分别显示水平边框和垂直边框

同样，常用的浏览器对该属性的支持并不是很好，因此最好使用 CSS 中的相应特性。

9. summary 属性

summary 属性用于为非可视化浏览器(例如语音浏览器或盲文浏览器)提供表的目的和结构的概述。这个属性的值不会在 IE 或 Firefox 浏览器中显示，但为了可访问性目的，最好在页面中包含它：

```
summary="Table shows the operating profit for the last four quarters. The
first column indicates the quarter, the second indicates outgoings, the
third indicates receipts, and the fourth indicates profit."
```

10. width 属性(逐渐淘汰)

width 属性用于指定表的宽度，单位是像素或者是可用空间的百分比。当表没有被嵌套在其他元素中时，可用空间就是屏幕的宽度；否则可用空间是包含它的元素的宽度。

```
width="500" 或 width="90%"
```

4.2.2 包含表行的<tr>元素

<tr>元素用于包含表中的每一行。出现在同一个<tr>元素中的内容应该显示在相同行中。它可以附带 5 个属性，其中 4 个逐渐被淘汰，最好使用 CSS 中的相应特性替代它们。

1. align 属性(逐渐淘汰)

align 属性指定某行中所有单元格的内容的位置。

```
align="alignment"
```

表 4-3 列举了 align 属性的所有可能值。

表 4-3

值	目　的
left	内容是左对齐的
right	内容是右对齐的
center	内容在单元格中水平居中
justify	两端对齐单元格中的文本以填充整个单元格
char	单元格的内容与指定字符的第一个实例水平对齐(例如，数字可以与小数点的第一个实例对齐)

默认情况下，<td>单元格通常是左对齐，而<th>单元格通常是居中对齐。遗憾的是，仅有 Firefox 2 和 Netscape 6+浏览器支持文本的两端对齐，并且 IE 和 Firefox 浏览器都不支持 char 值。

2. bgcolor 属性(逐渐淘汰)

bgcolor 属性设置行的背景色。这个属性的值应当是附录 D 中讨论的十六进制代码或者颜色值。

```
bgcolor="#rrggbb"
```

bgcolor 属性通常用在<tr>元素中，以便使得表的行交替显示不同的背景色，从而具有更好的可读性。

3. char 属性

char 属性用于指定行中每一个单元格的内容将与特殊字符(该字符称为轴字符)的第一个实例对齐。在 HTML 中，这个属性的默认字符是小数位，主要思想是小数数字将与小数位对齐，如下所示：

```
13412.22
  232.147
 2449.6331
    2.12
```

该属性的语法如下所示：

```
char="."
```

遗憾的是，在编写本书时，这个潜在非常有用的属性并没有被支持，浏览器没有支持它的需求。

4. charoff 属性

charoff 属性的名称是其作用的缩写，该属性用于指定一个字符偏移量。它指示使用 char 属性对齐的字符应该根据作为偏移量的字符数量或者文本长度的百分比进行定位。如果这个属性被省略，默认行为是使偏移量与“char 属性中指定字符前面所出现的文本内容的最大长度”等值，

```
charoff="5"
```

遗憾的是，在编写本书时，这个属性并没有被支持，浏览器没有支持它的需求。

5. valign 属性(逐渐淘汰)

valign 属性指定行中每一个单元格的内容的垂直对齐方式，语法如下所示：

```
valign="verticalPosition"
```

表 4-4 给出了 verticalPosition 的可能值。

表 4-4

值	目　的
top	内容与单元格的顶部对齐
middle	内容与单元格的中心垂直对齐
bottom	内容与单元格的底部对齐
baseline	对齐内容，从而每一个单元格中的第一行文本从相同的水平线开始

4.2.3 表示表单元格的<td>元素和<th>元素

表中的每一个单元格将由一个<td>元素或<th>元素表示，其中前者用于包含表数据的单元格，后者用于包含表题头的单元格。

默认情况下，<th>元素的内容通常以粗体显示，与单元格的中心水平对齐。而<td>元素通常以左对齐和非粗体的形式显示(除非通过 CSS 或者其他元素指示)。

<td>元素和<th>元素能够附带的属性集相同，这些属性仅应用于它们所在的单元格。这些属性的效果将覆盖表或者任何包含该单元格的元素(例如行)的相应设置。

除了通用属性和基本事件属性之外，<td>元素和<th>元素还能够附带如下属性：

```
abbr    align   axis      bgcolor  char    charoff   colspan  headers
height  nowrap  rowspan   scope    valign  widthThe  abbr     Attribute
```

1. abbr 属性

abbr 属性用于提供单元格内容的缩写版本。如果在小屏幕中使用浏览器查看页面，将显示这个属性中的内容，而不是单元格的完整内容。

```
abbr="description of services"
```

虽然主要的浏览器当前不支持这个属性，但它的应用将会越来越广泛，因为会有越来越多具有小屏幕的设备访问 Internet。

2. align 属性(逐渐淘汰)

align 属性设置单元格内容的水平对齐方式。

```
align="alignment"
```

align 属性的可能值包括 left、right、center、justify 和 char，在本章前面的“align 属性”一节中已经描述过这些值。

3. axis 属性

利用 axis 属性可以向单元格添加概念上的分类，从而表示 *n* 维数据。这个属性的值将是单元格所属的分类的名称列表，名称之间以逗号隔开。

```
axis="heavy, old, valuable"
```

这个属性并不会带来可视的格式化效果，而是用于保存数据，然后可以以编程方式使用这些数据，例如查询所有属于某个特定分类的单元格。

4. bgcolor 属性(逐渐淘汰)

bgcolor 属性设置单元格的背景颜色。这个属性的值可以是十六进制代码，也可以是一种颜色名——附录 D 中有关于十六进制代码和颜色名的介绍。

```
bgcolor="#rrggbb"
```

5. char 属性

char 属性用于指定一个字符，该字符的第一个实例将用于水平对齐单元格的内容(具体内容可查看本章前面“包含表行的<tr>元素”一节的“char 属性”子节中的描述)。

6. charoff 属性

charoff 属性用于指定偏移字符的数量，这些字符可以显示在 char 属性的值指定的字符之前(具体内容可查看本章前面“包含表行的<tr>元素”一节的“charoff 属性”子节中的描述)。

7. colspan 属性

colspan 属性用于指定一个单元格将跨越的表列的数量。colspan 属性的值是单元格伸展的列的数量(具体请查看本章后面“使用 colspan 属性跨越多列”一节)。

```
colspan="2"
```

8. headers 属性

headers 属性用于指示哪一个题头对应该单元格。这个属性的值是题头单元格的 id 属性值的列表，属性值之间以空格隔开：

```
headers="income q1"
```

这个属性的主要目的是用于支持语音浏览器。当该浏览器向用户朗读表时，通常很难确定当前正在查看的是哪一行和哪一列；因此，header 属性用于提醒用户当前单元格的数据属于哪一行和哪一列。

9. height 属性(逐渐淘汰)

height 属性用于指定单元格的高度，单位为像素或者与可用空间的百分比：

```
height="20" 或 height="10%"
```

10. nowrap 属性(逐渐淘汰)

nowrap 属性用于阻止单元格中的文本换行。仅当换行会使文本变得无意义时(例如一

行代码如果变成两行将无法工作)才应该使用 nowrap 属性。在 HTML 中，使用不带属性值的该属性。但是在 Transitional XHTML 中，这是不被允许的用法，而是应当使用如下形式：

```
nowrap="nowrap"
```

11. rowspan 属性

rowspan 属性用于指定单元格跨越的表行数量，该属性的值是单元格伸展的行数量 (具体内容请查看本章后面的“使用 rowspan 属性跨越多行”一节)。

```
rowspan="2"
```

12. scope 属性

scope 属性用于指示当前的题头向哪些单元格提供标签或者题头信息。在基本表中，可使用它替代 headers 属性，但是对这种功能的支持并不是很广泛：

```
scope="range"
```

表 4-5 给出了该属性可能的值。

表 4-5

值	目　的
row	单元格包含行的题头信息
col	单元格包含列的题头信息
rowgroup	单元格包含行组的题头信息(使用<thead>、 <tbody>或<tfoot>元素创建行中的单元格组)
colgroup	单元格包含列组的题头信息(使用<col> 或<colgroup>元素创建列组，本章后面将讨论这两个元素)

13. valign 属性(逐渐淘汰)

valign 属性用于指定单元格内容的垂直对齐方式。可能的值包括 top、middle、bottom 和 baseline，本章前面“包含表行的<tr>元素”一节的“valign 属性”子节中详细讨论这些属性值。

14. width 属性(逐渐淘汰)

width 属性用于指定单元格的宽度，单位是像素或者是可用空间的百分比。

```
width="150" 或 width="30%"
```

仅需要为表的第一行中的单元格指定 width 属性，剩余行将遵循第一行中的单元格宽度。

如果为<table>元素指定了 width 属性，并且后面单个单元格的宽度总体上大于 width 属性指定的宽度，则大多数浏览器将缩小这些单元格的宽度以匹配表的宽度。

也可以添加一个特殊的值“*”，这意味着这个单元格将占用表中剩余的可用空间。因此，如果表的宽度是 300 像素，一行中前两个单元格分别被指定为 50 像素宽，并且第三个

单元格具有值“*”，则它将占用200像素——即表剩余的宽度。如果没有指定表宽度，则第三列将占用浏览器窗口的剩余宽度。

需要注意的是，不能为不同表行中的对应列指定不同宽度的<td>元素。因此，如果表的第一行中具有3个100像素宽的<td>元素，则第二行中不能具有一个200像素宽和两个50像素宽的<td>元素。

试一试 易于访问的时间表

在这个示例中将创建一个时间表，这个时间表专门为视觉受损的人设计。因为在现实世界中很可能已经遇到过这种时间表，所以这个示例中将包含一些逐渐淘汰的属性。

(1) 由于这个示例中包含一些逐渐淘汰的属性，所以需要建立可以处理Transitional XHTML 1.0文档的程序框架：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
   <title>An accessible timetable</title>
</head>
<body>
</body>
</html>
```

(2) 接着添加创建具有3行和3列的表所需要的主要元素，最左边的列和最上方的行将包含题头。在创建表的过程中，为表添加一些内容。该时间表将通过XHTML给出虚构的周末课程表，包括周六和周日的早上和下午的课程：

```
<body>
<table>
  <tr>
    <th></th>
    <th>Saturday</th>
    <th>Sunday</th>
  </tr>
  <tr>
    <th>Morning</th>
    <td>The structure of a document and how to mark up text.</td>
    <td>Adding tables and forms to pages. Splitting pages up into windows
       called frames.</td>
  </tr>
  <tr>
    <th>Afternoon</th>
    <td>Linking between pages and adding color images and objects to
       your pages.</td>
    <td>Using CSS to style your documents and make them look attractive.</td>
  </tr>
</table>
</body>
```

(3) 下一步是向包含内容的<th>元素中添加 id 属性，向<td>元素中添加 header 属性。header 属性的值必须与 id 属性的值对应，以表明哪一个题头对应于哪一个单元格：

```
<table>
  <tr>
    <th></th>
    <th id="Saturday">Saturday</th>
    <th id="Sunday">Sunday</th>
  </tr>
  <tr>
    <th id="Morning">Morning</th>
    <td headers="Saturday Morning" abbr="Structure and markup">The
        structure of a document and how to markup text.</td>
    <td headers="Sunday Morning" abbr="Tables, forms and frames">Adding
        tables
        and forms to pages. Splitting pages up into windows called
frames</td>
  </tr>
  <tr>
    <th id="Afternoon">Afternoon</th>
    <td headers="Saturday Afternoon" abbr="Links, color, images,
        objects">Linking between pages, and adding color images and
        objects to your pages.</td>
    <td headers="Sunday Afternoon" abbr="CSS">Using CSS to style your
        documents
        and make them look attractive.</td>
    </tr>
</table>
```

(4) 将文件保存为 table.html。图 4-5 中的示例包含一些 CSS 样式规则，第 8 章中将更多地介绍 CSS 方面的知识。

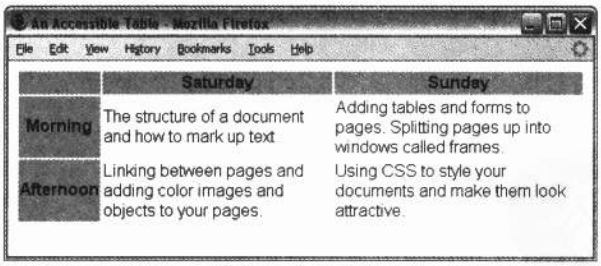


图 4-5

工作原理

将这个表包含在<table>元素中，然后一次编写一行内容。从最顶端的行开始，该行具有 3 个表题头元素。第一个元素是空的，因为表的左上角单元格为空。后两个元素包含关于日期的题头。注意，个别的表单元格将使用 id 属性，以便能够指示哪些题头对应这些单元格。

```
<table>
  <tr>
    <th></th>
    <th id="Saturday">Saturday</th>
    <th id="Sunday">Sunday</th>
  </tr>
  ...
</table>
```

在表的下一行中，第一个单元格是该行的题头，该题头表明该行给出早上课程的时间。后两个单元格是表的数据。**headers** 属性包含它们的对应题头元素中的 **id** 属性值，而 **abbr** 属性包含表单元格内容的缩写：

```
<tr>
  <th id="Morning">Morning</th>
  <td headers="Saturday Morning" abbr="Structure and markup">The structure of
     a document and how to markup text.</td>
  <td headers="Sunday Morning" abbr="Tables, forms and frames">Adding tables
     and forms to pages. Splitting pages up into windows called frames</td>
</tr>
```

最后一行的结构与第二行相同：

```
<tr>
  <th id="Afternoon">Afternoon</th>
  <td headers="Saturday Afternoon" abbr="Links, color, images,
     objects">Linking between pages, and adding color images and
     objects to your pages.</td>
  <td headers="Sunday Afternoon" abbr="CSS">Using CSS to style your
  documents
     and make them look attractive.</td>
</tr>
```

只要接受依次编写每一行的思想，则可以没有任何问题地创建非常复杂的表。

坦率地说，这个示例比后面将遇到的大多数表都复杂。很多人都不习惯在<table>元素中使用 **id** 属性和 **header** 属性，但添加这两个属性将使得视觉受损的人更容易使用表，特别是当表具有大量的列和行时。此外，您不会经常看到在表单元格中使用 **abbr** 属性。事实上，如果现在查看 Web 中其他人编写的代码，将很可能会看到大量逐渐淘汰的属性，而不是使用本示例中强调的属性。

注意：

在页面中包含上面的属性，将使您有别于其他没有学习如何使表更具可访问性的程序员。此外，许多工作岗位对可访问性问题的重视逐渐增加，因此您应当学习如何使用这些属性。

4.3 高级表

现在您已经了解了关于创建表的基本知识，接下来可以开始学习一些更高级的内容，例如：

- 将表划分为 3 个部分：表头、表主体和表尾
- 给表添加标题
- 利用 rowspan 属性和 colspan 属性使单元格跨越多行或多列
- 使用<colgroup>元素分组列
- 使用<col>元素在不相关的列之间共享属性

4.3.1 将表划分为表头、表主体和表尾

表可以划分为 3 个部分：表头、表主体和表尾。表头和表尾与字处理文档中的页眉和页脚相似，它们在每一页中保持相同的内容，而表主体是表的主要内容。

表各部分的划分使得浏览器可以显示更丰富的表格式。例如当打印表时，如果表超过一页，则浏览器可以在每一页中打印表头和表尾。听觉浏览器(它能够为用户阅读页面)可以让用户在内容和具有额外信息的表头或表尾之间方便地定位。

如果表太大而无法在一页中显示，可以使表头和表尾始终可见，而表主体则具有一个滚动条以方便浏览。但是，主要的浏览器不支持这种方式。

将表划分为表头、表主体和表尾的 3 个元素是：

- 用于创建独立的表头的<thead>元素
- 用于指示表主体的<tbody>元素
- 用于创建独立的表尾的<tfoot>元素

一个表可能包含多个<tbody>元素，以指示不同的“页面”或者数据组。

说明：

在源文档中，<tfoot>元素必须出现在<tbody>元素之前。

下面的表示例使用了这些元素(ch04_eg04.html)：

```
<table>
  <thead>
    <tr>
       <td colspan="4">This is the head of the table</td>
    </tr>
  </thead>
  <tfoot>
    <tr>
       <td colspan="4">This is the foot of the table</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
```

```
        <td>Cell 1</td>
        <td>Cell 2</td>
        <td>Cell 3</td>
        <td>Cell 4</td>
      </tr>
      <tr>
        ...more rows here containing four cells...
      </tr>
    </tbody>
    <tbody>
      <tr>
        <td>Cell 1</td>
        <td>Cell 2</td>
        <td>Cell 3</td>
        <td>Cell 4</td>
      </tr>
      <tr>
        ...more rows here containing four cells...
      </tr>
    </tbody>
  </table>
```

图 4-6 给出了这个示例在 Firefox 浏览器中的外观，Firefox 浏览器支持 thead、tbody 和 tfoot 元素。注意，这个示例利用 CSS 赋予表头和表尾背景阴影，并且表头和表尾元素中使用的字体较大；另外，每一个<td>元素的高度被设置为 100 像素，以便获得较大的表。

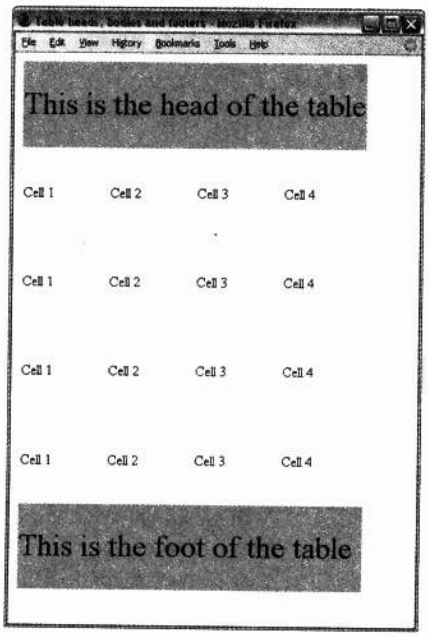


图 4-6

这 3 个元素附带的属性相同。除了通用属性之外，它们还能够附带如下属性：

```
align char charoff valign
```

1. align 属性(逐渐淘汰)

align 属性用于指定文本和被包含元素的水平位置。align 属性的可能值包括 left、right、center、justify 和 char，在本章前面“包含表行的<tr>元素”一节的“align 属性”子节中已经描述过这些值。

2. char 属性

char 属性指定一个字符，该字符的第一个实例用于水平对齐列组中每个单元格的内容(详细描述可查看本章前面“包含表行的<tr>元素”一节中的“char 属性”子节)。

3. charoff 属性

charoff 属性用于指定偏移字符的数量，这些字符可显示在 char 属性值指定的字符之前(详细描述可查看本章前面“包含表行的<tr>元素”一节中的“charoff 属性”子节)。

4. valign 属性(逐渐淘汰)

valign 属性用于指定每一个元素内的单元格内容的垂直对齐方式。可能的值包括 top、middle、bottom 和 baseline，本章前面“包含表行的<tr>元素”一节的“valign 属性”子节中讨论过这些属性值。

4.3.2 在表中添加<caption>

为了添加表标题，可以在起始标签<table>后面和第一行或表头之前使用<caption>元素：

```
<table>
<caption>Spanning columns using the colspan attribute</caption>
   <tr>
```

默认情况下，大多数浏览器将在表上方居中显示这个属性的内容，如下一节中的图 4-7 所示。

4.3.3 使用 colspan 属性跨越多列

在前面介绍<td>元素和<th>元素时提及，它们能够附带一种属性，这个属性允许表单元格扩展到多列。

记住，当您操作表时，需要将其想象为网格。colspan 属性允许单元格跨越多列，这意味着它能够水平跨越网格中的多个矩形。请查看下面的示例，该示例使用逐渐淘汰的 border、width、height 和 bgcolor 属性可视化地演示了这一点(ch04_eg05.html)：

```
<table border="1">
  <caption>Spanning columns using the colspan attribute</caption>
  <tr>
    <td bgcolor="#efefef" width="100" height="100"> </td>
    <td bgcolor="#999999" width="100" height="100"> </td>
    <td bgcolor="#000000" width="100" height="100"> </td>
  </tr>
  <tr>
    <td bgcolor="#efefef" width="100" height="100"> </td>
    <td colspan="2" bgcolor="#999999"> </td>
  </tr>
  <tr>
    <td colspan="3" bgcolor="#efefef" height="100"> </td>
  </tr>
</table>
```

从中可以看到，对于单元格跨越的每一个额外列，不需要为该行再添加一个单元格。因此，如果一个表具有 3 列，其中一个单元格跨越了两列，则该行将仅具有两个<td>元素。

可能还会注意到，在单元格中使用了不换行间隔字符( )，包含该字符是为了使单元格具有内容；如果表的单元格没有内容，则某些浏览器将不显示背景色(无论该背景色是由 CSS 还是逐渐被淘汰的 bgcolor 属性指定)。

图 4-7 给出了该示例在浏览器中的外观。

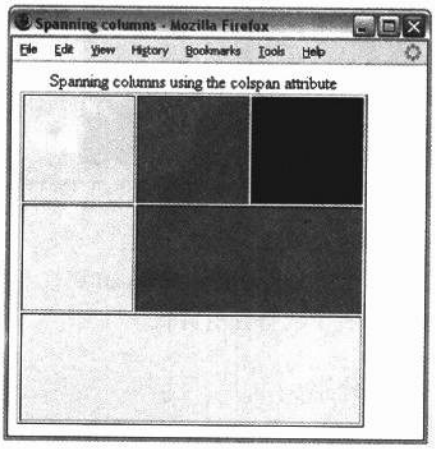


图 4-7

4.3.4 使用 rowspan 属性跨越多行

rowspan 属性起到的作用与 colspan 属性类似，但其作用的方向与 colspan 属性相反；该属性允许单元格垂直跨越多个单元格。

当使用 rowspan 属性时，该行下方对应的单元格必须保留出来：

```
<table border="1">
  <caption>Spanning rows using the colspan attribute</caption>
```

```
  <tr>
    <td bgcolor="#efefef" width="100" height="100"> </td>
    <td bgcolor="#999999" width="100" height="100"> </td>
    <td rowspan="3" bgcolor="#000000" width="100" height="100"> </td>
  </tr>
  <tr>
    <td bgcolor="#efefef" height="100"> </td>
    <td rowspan="2" bgcolor="#999999"> </td>
  </tr>
  <tr>
    <td bgcolor="#efefef" height="100"> </td>
  </tr>
</table>
```

rowspan 属性的效果如图 4-8 所示。

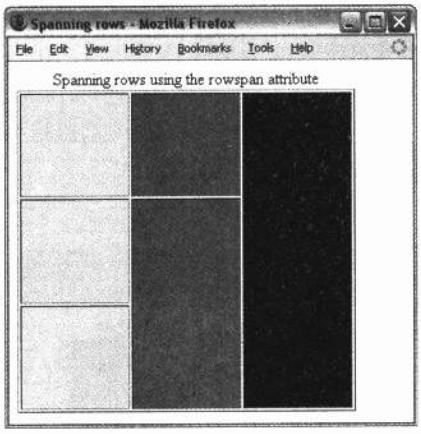


图 4-8

对于利用表控制页面布局的设计人员来说，rowspan 属性和 colspan 属性特别有用；但是这种控制布局的技术已经被 CSS 中的特性所替代。

4.3.5 使用<colgroup>元素分组列

如果正在创建复杂的表，可以使用<colgroup>元素将一个或多个相邻列分组在一起。当两个或多个相邻列包含相似类型的信息时，进行分组就非常有用。这样可以将格式应用于列组，而不是单独地赋予每一列样式。在第 7 章中学习 CSS 时，将看到如何使用 class 属性将特定样式与列组关联在一起。

例如，下面的表具有 12 列。前 8 列位于第一个列组，接下来的两列位于第二个列组，最后两列位于第三个列组：

```
<table>
  <colgroup span="8" width="75" class="mainColumns" />
  <colgroup span="2" width="50" class="subTotalColumns" />
```

```
  <colgroup span="2" width="80" class="totalColumns" />
  <tr>
    <td></td>
    ...
    <td></td>
  </tr>
</table>
```

当使用<colgroup>元素时，它紧接在起始标签<table>之后。span 属性用于指示列组中包含多少列，width 属性设置组中每一列的宽度(但在 XHTML 中应当使用 CSS 替代该属性)，class 属性可以用于附加更多的 CSS 样式。

除了通用属性之外，<colgroup>元素还能附带如下属性：

```
align char charoff span valign width
```

虽然使用该元素可以执行一些基本的格式化，例如背景色修改，但浏览器对它的支持有限。

1. align 属性(逐渐淘汰)

align 属性用于指定<colgroup>元素所包含单元格中的文本的水平位置。align 属性可能的值包括 left、right、center、justify 和 char，本章前面“包含表行的<tr>元素”一节中的“align 属性”子节中描述过这些属性值。

2. char 属性

char 属性用于指定一个字符，该字符的第一个实例用于水平对齐列组中每一个单元格的内容(具体内容可查看本章前面“包含表行的<tr>元素”一节的“char 属性”子节中的描述)。

3. charoff 属性

charoff 属性用于指定偏移字符的数量，这些字符可显示在 char 属性值指定的字符之前(具体内容可查看本章前面“包含表行的<tr>元素”一节的“charoff 属性”子节中的描述)。

4. span 属性

span 属性用于指定<colgroup>元素所跨越的列的数量，例如：

```
span="5"
```

5. valign 属性(逐渐淘汰)

valign 属性用于指定单元格内容的垂直对齐方式。可能的值包括 top、middle、bottom 和 baseline，关于这些属性值的描述可查看本章前面“包含表行的<tr>元素”一节中的“valign 属性”子节。

6. width 属性

width 属性用于指定列中每个单元格的宽度，单位为像素或者可用空间的百分比。width 属性也可以采用特殊的值“0*”，该值指定列的宽度为显示该列中所有内容所需要的最小宽度。

4.3.6 利用<col>元素让列共享样式

使用<col>元素可以获得与<colgroup>元素类似的效果，但是没有实际地指定结构化的列组。它也可以用于指示某一列与该组中剩余的列具有不同的格式。

<col>元素始终是空元素，因此仅用于附带属性，而不附带内容。

例如，下面的表具有 10 列，前面的 9 列虽然不处于一个组中，但它们的格式可以与最后一列不同，因为它们属于独立的集合。

```
<table>
  <colgroup span="10">
    <col span="9" width="100" id="mainColumns" />
    <col span="1" width="200" id="totalColumn" />
  </colgroup>
  <tr>
    <td></td>
    ...
    <td></td>
  </tr>
</table>
```

<col>元素所能附带的属性与<colgroup>元素相同。

遗憾的是，浏览器当前对分组列的支持是有限的。

4.4 表的可访问性问题

因为表能够用于创建网格，很多设计人员过去经常使用它们控制整个文档的布局，并且整个 Web 页面都建立在一个表中。在考虑如何使用表控制文档的布局之前，重点是要理解它们如何被非可视化用户代理(例如语音浏览器)处理；否则，视觉受损的人可能无法访问您的页面。为了理解如何使表具有可访问性，首先需要了解表如何线性化页面。

4.4.1 表的线性化

为了理解屏幕阅读器如何阅读表，考虑下面的简单表：

```
<table border="1">
  <tr>
    <td>Column 1, Row 1</td>
    <td>Column 2 Row 1</td>
  </tr>
```

```
  <tr>
    <td>Column 1, Row 2</td>
    <td>Column 2, Row 2</td>
  </tr>
</table>
```

图 4-9 给出了这个简单表在浏览器中的外观。

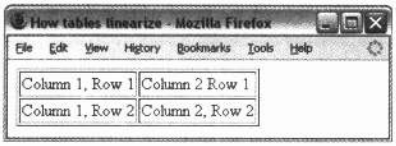


图 4-9

屏幕阅读器在为访问者阅读表时，趋向于对表执行线性化。这意味着它们从第一行开始，从左到右一个接一个地阅读该行中的单元格，然后移动到下一行中，直到阅读完表中的每一行。图 4-9 中的单元格的阅读顺序如下所示：

- 第 1 行第 1 列
- 第 1 行第 2 列
- 第 2 行第 1 列
- 第 2 行第 2 列

4.4.2 用于布局的表线性化

因为表可以用于控制元素在页面中的显示位置，所以 Web 开发者经常习惯于使用表安排文本或图像在页面中显示的位置。因为设计人员可以控制表的一些特性，例如每个单元格的宽度，所以可以创建具有多个文本列的布局，并确定每列的宽度。通常整个 Web 页面的主体包含于一个表中。

注意：

虽然 W3C 的意图是让表单独用于存放数据，而让 CSS 作为定位页面中的元素的首选机制，但实现该意图的前提是浏览器需要改进对 CSS 定位功能的支持(第 9 章中将介绍这方面的内容)，并且更多的设计人员需要了解如何使用 CSS 进行定位。因此，在很长一段时间内，表很可能仍将用于控制 Web 页面的布局。

第 9 章和第 10 章中将更多地介绍如何使用表和 CSS 来控制元素在页面中的位置，但是现在应当考虑如何在为屏幕阅读器用户线性化的表中编写页面，并且应当在确定表能够正确线性化之后才使用其进行布局。

本章前面提及，可以在表单元格内包含标记，只要整个元素包含在该单元格中。这意味着可以在表单元格中放置另外一个表，创建所谓的嵌套表。

如果使用嵌套表，当屏幕阅读器遇到包含另一个表的单元格时，它必须对嵌套表线性化之后，才能移动到下一个单元格。例如，图 4-10 中给出了一种常用的页面布局。

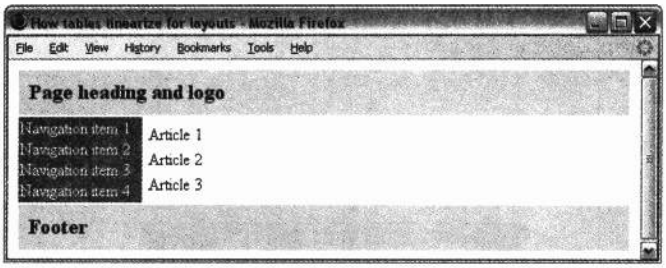


图 4-10

图 4-10 中的布局由一个具有三行两列的表创建。

- 在第一行中，题头和徽标位于一个跨越两列的单元格中。
- 在第二行中，第一个单元格包含了导航栏，第二个单元格包含了一个嵌套表，该嵌套表具有 3 行 1 列。
- 在第三行中，单元格跨越了两列，类似于第一行。

下面是这个页面的代码(注意，在该源文档中也具有一些 CSS 规则，用于赋予该表样式，ch04_eg10.html)：

```
<table>
  <tr>
    <td colspan="2" id="heading">Page heading and logo</td>
  </tr>
  <tr>
    <td id="navigation">Navigation item 1 <br />
        Navigation item 2 <br />
        Navigation item 3 <br />
        Navigation item 4 <br />
    </td>
    <td>
      <table>
        <tr>
           <td>Article 1</td>
        </tr>
        <tr>
           <td>Article 2</td>
        </tr>
        <tr>
           <td>Article 3</td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
     <td colspan="2" class="footer">Footer</td>
  </tr>
</table>
```

注意：

这个示例本来可以对第一行使用<thead>元素，对最后一行使用<tfoot>元素。但是因为表的内容实际上不是表列数据，所以<table>元素的使用并不是它的真正意图；此处将仅依赖于一些基本元素。

在这个示例中，页面中元素的阅读顺序如下：

- Page heading and logo
- Navigation item 1
- Navigation item 2
- Navigation item 3
- Navigation item 4
- Article 1
- Article 2
- Article 3
- Footer

实际上最好使用语音浏览器测试表，例如 www.w3.org/WAI/References/Browsing#2 中列举的某个语音浏览器。

记住，如果使用表控制页面布局，应当使用样式表而不是标记来控制表中文本的外观(例如，不要只是使用<th>元素来获得文本的居中和加粗效果，<th>元素仅用于题头。不要使用元素获得斜体文本，因为屏幕阅读器可能在语音中添加音调变化以强调该文本)。

4.4.3 用于数据的表线性化

如果使用表表示数据，应当尽量使表简单，单元格不要跨越多行和多列，因为它们会使表变得非常复杂。下面是创建用于存放数据的表所使用的一些一般性指导原则：

- 尽量使用<th>元素指示表题头。如果不喜欢它们的可视化表示，可以使用 CSS 样式覆盖该元素。
- 如果不能使用<th>元素指示表题头，可以对带有题头的单元格使用具有 row 或 col 值的 scope 属性。
- 始终将题头放置在表的第一行第一列中。
- 如果表比较复杂，并且其中包含一些跨越多个单元格的单元格，则对这些单元格使用 headers 属性，并且清晰地指示哪一个题头应用于线性化过程中的下一个单元格。

4.5 本章小结

在本章中您已经看到，对于 web 开发人员来说，表是一种非常强大的工具。表不仅可以用于布局表列数据，也经常用于控制页面的布局。

本章介绍所有的表如何基于一种网格模式以及如何使用 4 种基本元素：<table>，用于包含每个表；<tr>，用于包含表行；<td>，用于包含表数据的单元格；<th>，用于表示包

含题头的单元格。

本章也介绍了如何为表添加表头、表尾和标题。当表的长度大于浏览器窗口或者一页打印纸张时，最好在表中添加<thead>元素和<tfoot>元素，它们有助于读者通过表头和表尾中的信息了解表中的内容。

单元格也可以跨越多行或多列，但对于包含数据的表来说应当尽量避免这样操作，因为这将使听觉浏览器很难向用户阅读表中的内容。也可以将列分组到一起，以便能够保持一致的结构，从而让它们共享相同的样式和属性。

最后，本章给出了关于表使用的一些可访问性问题。清楚了解表的线性化过程是非常重要的(屏幕阅读器在为用户阅读表之前执行线性化过程)，以便视觉受损的用户能够更好地访问网站。本章最后给出了一些能够使表对所有访问者具有更好的可访问性的指导原则。

下一章中将介绍如何使用表单从访问者处收集信息。

4.6 练习

所有练习题的答案都在附录 A 中给出。

1. 用于表的<caption>元素应当放置在文档的什么位置，默认情况下在何处显示标题？
2. 屏幕阅读器以什么样的顺序阅读图 4-11 中的表单元格？

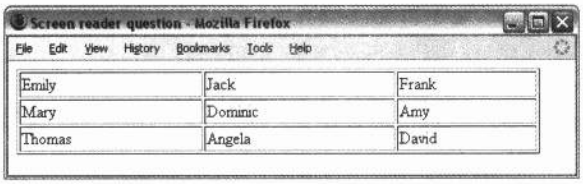


图 4-11

3. 创建一个表来存放图 4-12 中的数据。提示如下：文档必须遵循 Transitional XHTML 1.0 规范，因为 width 属性用于表第一行中的单元格。在本章中已经给出了一些如何生成边框的示例，即使用另一个逐渐淘汰的属性，但是这个属性应当应用于<table>元素而不是单元格。

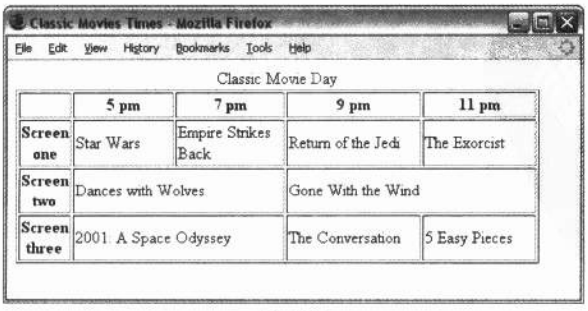


图 4-12

第5章

表　单

当希望从访问站点的用户处收集信息时，通常需要使用表单。您很可能已经在不同的Web站点上使用多种不同类型的表单，从简单的搜索框(它允许用户输入关键字以找到所需要的内容)到复杂的表单(例如在线订购食品或预订度假的表单)。

Web上的表单与需要填写的纸张表单非常类似。纸张上具有输入文本的区域、选中的方框、一些可供选择的选项等。在Web上，可以通过组合一些表单控件(例如用于输入文本的文本框、用于选中的复选框、选项框和单选按钮等)来创建表单。在本章中将介绍如何将每一种类型的控件组合到表单中。

本章将介绍以下内容：

- 使用<form>元素创建表单的方式
- 可以用于建立表单的不同类型表单控件——例如文本输入框、单选按钮、选项框和提交按钮
- 处理用户输入的数据的方式
- 使表单具有更好的可访问性的方式
- 结构化表单的内容的方式

学习完本章之后，您将能够创建所有类型的表单，以便从站点访问者处收集信息。

注意：

对所收集数据的处理方式依赖于Web站点所驻留的服务器。XHTML只用于向用户提供表单；它不允许用户说明如何处理所收集的数据。为了更好地了解如何处理通过表单收集的数据，请查看关于服务器端语言(例如ASP.net、PHP或JSP)的书籍。可在Wrox.com站点列出的书籍中查看关于这些主题的书籍。

5.1　表单简介

创建的任何表单都将位于<form>元素中。在起始标签<form>和结束标签</form>之间，存在一些表单控件(文本输入框、下拉框、复选框、提交按钮等)。类似于页面的其他元素，<form>元素中也可以包含其他XHTML标记。

用户向表单中输入数据之后，通常需要单击提交按钮(但是按钮上的实际文本可能是其他内容，例如Search、Send或Proceed——并且通常按下Enter键会具有类似于单击该按钮

的相同效果)。这表明用户已经填写完表单，并且通常将表单数据发送给 Web 服务器。

当访问者所输入的数据到达服务器之后，某个脚本或者其他程序通常会处理它们，并向访问者返回一个新 Web 页面。返回的页面通常响应访问者所提出的请求，或者确认访问者执行的操作。

作为一个示例，您可能希望在页面中添加一个如图 5-1 所示的搜索表单。

图 5-1

这个表单包含了一个文本框，用户可以在其中输入一些关键字以搜索希望找到的内容；另外包含一个提交按钮，该按钮上具有单词“Search”。当用户单击 Search 按钮时，信息被发送到服务器。然后服务器处理数据，并为用户生成一个新页面，该页面告诉用户哪些页面满足搜索标准(如图 5-2 所示)。

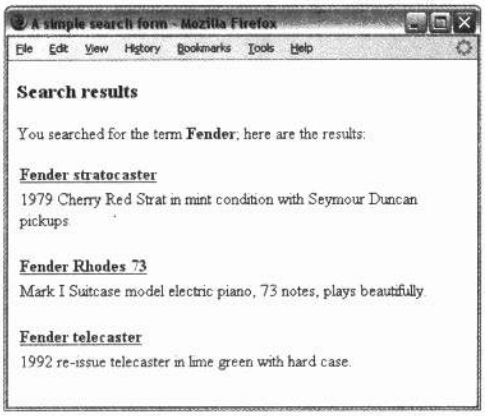


图 5-2

当用户填好一个表单后，数据以“名/值”对的形式发送给服务器，名字对应于表单控件的名称，值是用户输入的内容(如果用户可以输入一个答案)或者选择的选项值(如果表单上存在一个选项列表)。

每一项都必须具有一个名称和一个值，因为如果在一个表单上有 5 个文本框，则需要知道哪一个数据对应哪一个文本框。然后处理应用程序才能分别处理每一个表单控件中的信息。

下面是图 5-1 所示的简单搜索表单的代码：

```
<form action="http://www.example.org/search.aspx" method="get">
  <h3>Search the site</h3>
  <input type="text" name="txtSearchItem" />
```

```
    <input type="submit" value="Search" />
</form>
```

其中<form>元素附带了一个称为 action 的属性，它的值是 Web 服务器中处理搜索请求的页面的 URL。同时，method 属性指明服务器使用哪一个 HTTP 方法将表单数据发送给服务器(在本章后面，您将了解到可以使用两个方法 get 和 post)。

在本章后面将介绍一些更高级的表单，但是现在首先详细地查看组成表单的元素。

5.2 利用<form>元素创建表单

表单通常位于一个称为<form>的元素中。<form>元素还能包含其他标记，例如段落、题头等。但是，<form>元素绝对不能包含其他<form>元素。

假设页面中的<form>元素相互独立(任何<form>元素都不包含其他<form>元素)，则页面中可以包含任意数量的表单。例如，可以在同一个页面中包含登录表单、搜索表单、用于订阅时事通讯的表单等。如果一个页面中有多个表单，则用户一次只能向服务器发送一个表单中的数据。

每一个<form>元素应该至少附带两个属性：

```
action method
```

<form>元素也可以附带所有的通用属性、UI 事件属性以及下面的属性：

```
enctype accept accept-charset onsubmit onreset
```

5.2.1 action 属性

action 属性表明提交表单时将如何处理数据。通常 action 属性的值是 Web 服务器中的一个页面或者一个程序，当用户单击提交按钮后，该页面或程序将接收表单中的信息。

例如，如果具有一个由“用户名”和“密码”组成的登录表单，用户所输入的内容将被传递给 Web 服务器中以 ASP.net 语言编写的 login.aspx 页面，此时 action 属性将如下所示：

```
<form action="http://www.example.org/membership/login.aspx">
```

大多数浏览器将只接受作为 action 属性值的以 http://开头的 URL。

5.2.2 method 属性

将表单数据发送给服务器的方式有两种，每一种对应于一个 HTTP 方法：

- get 方法，它将数据作为 URL 的一部分进行发送
- post 方法，它将数据隐藏在 HTTP 头中

在本章的后面将详细介绍这两个方法，其中您将了解到它们的意义以及何时应当使用哪一个方法。

5.2.3 id 属性

id 属性允许用户唯一标识页面中的<form>元素，如同可以使用该属性唯一标识页面中的任何元素一样。

为每一个<form>元素提供一个 id 属性是优秀的实践，因为许多表单利用样式表和脚本，它们需要使用 id 属性来标识表单。

注意：

在第 12 章中将介绍，如果自动将浏览器的光标放置在表单的第一个文本框中，有时会对用户有帮助作用。为此，需要添加一个 id 或 name 属性以标识表单。

在一个文档中，id 属性的值应该是唯一的，它也应该遵循第 1 章中提到的一些其他规则。有些人以字符 frm 作为表单的 id 属性或 name 属性值的起始字符，然后使用该值的剩余部分描述表单所收集数据的类型，例如 frmLogin 或 frmSearch。

5.2.4 name 属性(逐渐淘汰)

类似于在其他元素中的用法，name 属性是 id 属性的前身。

与 id 属性一样，name 属性的值在文档中应该是唯一的。另外，该属性的值通常以字符 frm 作为开头，后面跟上表单的目的。

5.2.5 onsubmit 属性

您很可能遇到过这样的情况：在 Web 站点中填写完表单，然后单击发送表单数据的按钮(甚至在将页面发送给服务器之前就执行该操作)，此时将会显示一条消息告诉您没有输入某些数据或者输入错误的数据。当这种情况发生时，您很可能是遇到了使用 onsubmit 属性的表单，该属性在浏览器中运行一段脚本，在表单被发送给服务器之前检查所输入数据的正确性。

当用户单击提交按钮时，将会激活一个事件。这类似于浏览器举起它的手并说“嘿，我正向服务器发送这个表单的数据。”隐藏在这些事件背后的思想是，可以在将数据发送给服务器之前运行一个脚本(例如 JavaScript 脚本)，以确保提交的数据的质量和准确性。onsubmit 属性的值应该是一个脚本函数，当事件激活时将使用该函数。

因此，<form>元素中的 onsubmit 属性可能类似于如下：

```
onsubmit="validateFormDetails();"
```

在文档中必须已经定义了 validateFormDetails()函数(很可能在<head>元素中定义它)。因此，当用户单击提交按钮时，将调用并运行这个函数。

在表单数据发送给服务器之前对其进行一些检查具有如下两个关键优点：

- 如果页面中的数据存在错误，用户不需要花费额外的时间等待页面被发送给服务器然后返回。
- 服务器不需要处理大量的错误检查，因为已经在浏览器端执行检查。

这两个优点都将节省服务器的工作，对于非常繁忙的站点来说，这一点非常重要。

5.2.6 onreset 属性

某些表单包含一个 reset 按钮，它能够清空表单中的所有内容，有时该按钮上显示的可能是类似于 Clear Form(清除表单)的文本；当按下这个按钮时，将激活 onreset 事件，并且运行相应的脚本。

当使用 onreset 属性时，它的值是一个脚本(与 onsubmit 属性一样)，当用户单击调用该脚本的按钮时执行它。

注意：

onreset 事件和属性的使用频率远低于 onsubmit。然而，如果在页面中提供一个 Clear Form 按钮，最好在用户计划清除表单之前执行确认操作，以防止用户无意中按下该按钮。

5.2.7 enctype 属性

如果使用 HTTP post 方法向服务器发送数据，则在向服务器发送数据之前可以使用 enctype 属性指定浏览器编码数据的方式(以确保数据安全到达)。浏览器支持两种类型的编码方式：

- application/x-www-form-urlencoded，这是大多数表单使用的标准方法。使用该编码方式的原因是某些字符(例如空格、加号和某些其他非字母数字字符)不能发送给 Web 服务器。相反，它们被用于表示它们的其他字符所替代。
- multipart/form-data，这种方式允许将数据以多个部分的方式发送，每个连续的部分对应于一个表单控件，发送顺序按照它们在表单中的出现顺序。每个部分可以具有一个可选的“内容-类型”头，以指明该表单控件的数据类型。

如果没有使用这个属性，则浏览器将使用第一种编码方式。因此，如果表单允许用户向服务器上传文件(例如一幅图像)，或者需要向服务器发送非 ASCII 字符，则必须使用这个属性，此时该属性被赋予第二个值：

```
enctype="multipart/form-data"
```

5.2.8 accept-charset 属性

accept-charset 属性的思想是，它指定一个字符编码列表(用户可以输入该列表)，然后服务器可以根据该列表进行处理。但是，IE 7 和 Firefox 2 浏览器不支持这个属性。该属性的值应该是以空格或者逗号隔开的字符集列表(附录 E 中介绍了各种字符集)。

例如，下面的代码指示服务器接受 UTF-8 编码：

```
accept-charset="utf-8"
```

当前主要的浏览器允许输入任何字符集。

5.2.9 accept 属性

accept 属性类似于 accept-charset 属性，不同之处是它采用一个以逗号隔开的内容类型(或文件类型)列表，服务器可以利用该列表处理表单。但是，IE 7 和 Firefox 2 浏览器不支持这个属性。

该属性的思想是用户不能上传与列表中所列举的内容类型不同的文件。下面的代码表明用户仅能上传 GIF 或 JPEG 类型的图像：

```
accept="image/gif, image/jpg"
```

但是，当前主要的浏览器仍然允许用户上传任何类型的文件。附录 H 中列举了一些 MIME 类型。

5.2.10 target 属性

target 属性通常用于<a>元素中，以指明新的页面将被加载到哪一个框架或者浏览器窗口中。它也操作生成新页面的表单，并且允许用户指明当提交表单时生成的页面被加载到哪一个框架或者浏览器窗口中。

5.2.11 空白和<form>元素

另外需要注意的是，当浏览器遇到<form>元素时，它通常会在该元素周围创建一些额外的空白。这将影响您的设计，特别是希望将表单放置于较小的区域中时，例如将搜索表单放置在菜单栏中。如果利用 CSS 无法在目标浏览器中解决这个问题，则避免该问题的唯一方法是仔细放置<form>元素。

为了避免创建额外的空白，可以尝试将<form>元素放置在文档的起始部分或者末尾附近；如果在 Transitional XHTML 1.0 文档中使用表进行布局，则可以将<form>元素放置在<table>元素和<tr>元素之间(需要清楚的是，后一种方法是一种欺骗方式，因此如果对页面进行验证，该方法可能会造成错误。但是，大多数浏览器仍将以期望的方式显示表和表单)。

5.3 表单控件

本节将介绍一些不同类型的表单控件，可以利用它们从站点访问者处收集数据。本节主要介绍以下控件：

- 文本输入控件
- 按钮
- 复选框和单选按钮
- 选项框(有时称为下拉菜单)和列表框
- 文件选择框
- 隐藏控件

5.3.1 文本输入

您必定在许多 Web 页面中见到过文本输入框。最著名的文本输入框可能是 Google 主页中间靠右的文本输入框，它允许用户输入想要搜索的关键字。

在一个打印表单中，文本输入框的等价物是一个方框或者下划线，用户可以在其中或其上填写相应的内容。

实际上，表单中可以使用 3 种类型的文本输入框：

- 单行文本输入控件：用于仅需要一行用户输入的项，例如搜索框或者 e-mail 地址。利用<input>元素可以创建单行文本输入控件。
- 密码输入控件：这种控件类似于单行文本输入控件，不同之处是它能够掩藏用户输入的字符，因而无法在屏幕中看到用户输入的字符。这种控件趋向于以星号或者圆点代替用户输入的每一个字符，从而其他人无法通过简单地查看屏幕来了解用户的输入。密码输入控件主要用于在登录表单中输入密码或者某些敏感内容，例如信用卡号。密码输入控件的创建方式也是使用<input>元素。
- 多行文本输入控件：当要求用户输入的文本长度大于单个句子时，可以使用该控件。利用<textarea>元素创建多行文本输入控件。

1. 单行文本输入控件

利用<input>元素可以创建单行文本输入控件，此时该元素的 type 属性的值设置为 text。下面是单行文本输入控件用作搜索框的简单示例(ch05_eg02.html)：

```
    <form action="http://www.example.com/search.aspx" method="get"
name="frmSearch">
      Search:
      <input type="text" name="txtSearch" value="Search for" size="20"
             maxlength="64" />
      <input type="submit" value="Submit" />
    </form>
```

图 5-3 给出了这个表单在浏览器中的外观。

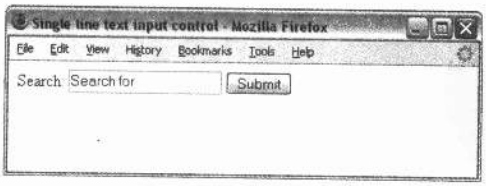


图 5-3

注意：

类似于一些人习惯以 frm 作为表单名称的起始字符，也经常使用 txt 作为文本框名称的起始字符，以指明该表单控件是一个文本框。当在服务器中处理数据时，这种方式特别方便，因为它有助于用户了解什么类型的表单控件发送该数据。

表 5-1 列举了<input>元素用于创建文本输入控件时可以附带的属性。注意，这个元素的 name 属性的目的非常特别，与前面遇到的其他元素不同。

表 5-1

属　性	目　的
type	指示希望创建的输入控件的类型。当希望创建单行文本输入控件时，该属性的值是 text。这个属性是必须的，因为<input>元素也用于创建其他类型的表单控件，例如单选按钮和复选框
name	用于提供“名/值”对(“名/值”对发送给服务器，表示每个表单控件和用户输入的值中的名称。每一个控件需要一个名称，以便与其相关联的值(值由用户提供或者选择)能够在另一端单独获取。
value	为文本输入控件提供一个初始值，当表单加载时将在页面中看到这个值。仅在用户希望页面加载时就在文本输入控件中写入一些内容(例如提示用户应该输入什么数据)时才使用这个属性；更多的时候，很可能将其设置为空
size	利用该属性可以指定文本输入控件的宽度，单位为字符。在前面的示例中，搜索框的宽度是 20 个字符。size 属性不影响用户能够输入的字符数(在前面的示例中，用户可以输入 40 个字符)；它只是指示输入控件具有多少个字符宽。如果用户输入的字符数大于输入控件的宽度，通过使用箭头键可以向右或向左滚动字符
maxlength	通过该属性可以指定能够输入到文本框中的最大字符数。通常输入最大字符数之后，即使用户按下更多的键盘键，也不会添加新的字符

当<input>元素的 type 属性的值为 text 时，它也可以附带下面的属性：

- 所有的通用属性
- disabled、readonly、tabindex 和 accesskey，本章后面将介绍它们

2. 密码输入控件

如果希望收集一些敏感数据，例如密码和信用卡信息，应当使用密码输入控件。密码输入控件掩藏了用户在屏幕上输入的字符，方式是使用圆点或者星号替换这些字符。

密码输入控件的创建方式与单行文本输入控件基本相同，不同之处是它的<input> 元素的 type 属性值被设置为 password。

下面的示例是一个登录表单，它结合了一个单行文本输入控件和一个密码输入控件(ch05_eg03.html)：

```
<form action="http://www.example.com/login.aspx" method="post">
  Username:
  <input type="text" name="txtUsername" value="" size="20" maxlength="20" />
  <br />
  Password:
  <input type="password" name="pwdPassword" value="" size="20"
maxlength="20" />
```

```
  <input type="submit" value="Submit" />
</form>
```

注意：

密码输入控件的名称的起始字符通常是 pwd，从而在服务器端处理数据时，可以知道所关联的值来自于一个密码输入框。

图 5-4 给出了这个登录表单在浏览器中的外观，此时用户准备开始在该表单中输入内容。

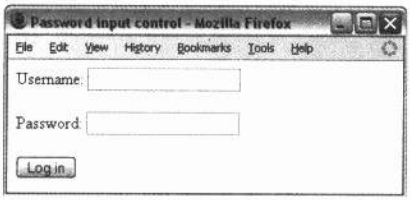


图 5-4

注意：

虽然密码在屏幕上是隐藏的，但它们仍然作为纯文本通过 Internet 发送。为了确保它们的安全，应当在客户端和服务器之间建立一条 SSL 连接。

3. 多行文本输入控件

如果希望允许站点访问者输入多行文本，应该使用<textarea>元素创建一个多行文本输入控件。

下面是一个多行文本输入控件的示例，它用于从站点访问者处收集反馈信息(ch05_eg04.html)：

```
<form action="http://www.example.org/feedback.asp" method="post">
  Please tell us what you think of the site and then click submit:<br />
    <textarea name="txtFeedback" rows="20" cols="50">
Enter your feedback here.
    </textarea>
  <br />
  <input type="submit" value="Submit" />
</form>
```

注意，<textarea>元素内的文本没有缩进。起始标签<textarea>和结束标签</textarea>之间的任何内容都被视为如同在一个<pre>元素内编写，并且源文档的任何格式都将被保留。如果单词“Enter your feedback here”在代码中被缩进，则它们在浏览器中显示的多行文本输入控件内也被缩进。

图 5-5 给出了这个表单在浏览器中的外观。

在图 5-5 中，当页面加载时，起始标签<textarea>和结束标签</textarea>之间的文本显示在文本区域中。用户在添加自己的文本之前可以删除该文本，如果用户没有从文本框中删除该文本，则当提交表单时，它们将被发送给服务器。用户通常只在<textarea>元素中已经写入的文本后输入文本，因此最好避免在该元素内添加内容，但仍然应该具有起始标签

<textarea>和结束标签</textarea>，否则一些较老的浏览器可能无法正确显示该元素。

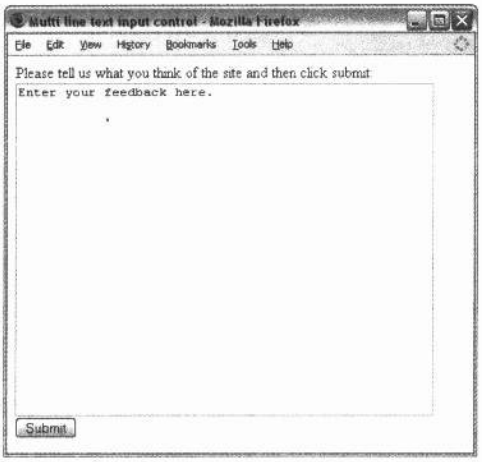


图 5-5

<textarea>元素可以附带表 5-2 中列举的属性。

表 5-2

属　性	目　的
name	控件的名称，它用于发送给服务器的“名/值”对中
rows	用于指定<textarea>的大小，它指示<textarea>元素应该具有的文本行数，从而对应于它的高度
cols	用于指定<textarea>的大小；此处该属性指定文本框的宽度并表示列数，一列等于一个字符的平均宽度

<textarea>元素还可以附带如下属性：

- 所有通用属性
- disabled、readonly、tabindex 和 accesskey，本章后面将介绍它们
- UI 事件属性

默认情况下，当用户输入的文本超出<textarea>的宽度时，文本将换行到下一行(这意味着重新开始下一行文本，类似于文字处理软件中对文本的操作)，但是服务器接收文本时，仍然认为它们都在一行中。由于某些用户期望句子能够在屏幕中应该换行的位置进行换行，所以一些主要的浏览器也支持称为 wrap 的额外属性，该属性允许用户指示文本换行的方式。该属性可能的值包括如下：

- off(默认值)，该值意味着如果用户输入文本占用的空间超出了允许的宽度，则在文本框中添加滚动栏，并且用户必须通过滚动才能看到输入的内容。
- virtual，意味着当文本换行时，用户将在新行中看到它。但是当文本传送到服务器时，所有的文本将在同一行中，除非用户按下 Enter 键，此时将该键处理为一个换行。
- physical，意味着当用户看到文本从新行开始时，服务器接收的文本也将是如此。

但是，wrap 属性并不是 XHTML 规范的一部分。

5.3.2 按钮

按钮通常用于提交表单，但是有时用于清除或者复位表单，甚至是用于触发客户端脚本(例如，在页面内的基本贷款计算器表单中，可以使用按钮触发一个脚本，该脚本能够在不将数据发送给服务器的情况下计算应该偿还的款项)。可以采用 3 种方式创建按钮：

- 使用<input>元素，将其 type 属性值设置为 submit、reset 或 button
- 使用<input>元素，将其 type 属性值设置为 image
- 使用<button>元素

对于每一种不同的方法，按钮的外观将稍有不同。

1. 使用<input>元素创建按钮

当使用<input>元素创建按钮时，可以通过使用 type 属性来指定按钮的类型。type 属性可以采用如下值：

- submit，创建能够自动提交表单的按钮
- reset，创建能够自动将表单控件复位为初始值的按钮
- button，创建用于在用户单击时触发客户端脚本的按钮

下面的示例演示了这 3 种类型的按钮(ch05_eg05.html)：

```
<input type="submit" name="btnVoteRed" value="Vote for reds" />
<input type="submit" name="btnVoteBlue" value="Vote for blues" />
<br /><br />
<input type="reset" value="Clear form" /> <br /><br />
<input type="button" value="calculate" onclick="calculate()" />
```

图 5-6 给出了这些按钮在 PC 机器(Mac 机器中显示的按钮将是标准 Mac 样式)上的 Firefox 浏览器中的外观。

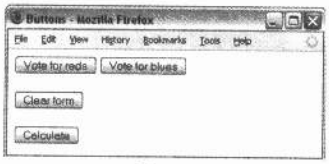


图 5-6

表 5-3 给出了按钮使用的属性。

表 5-3

属 性	目 的
type	用于指定按钮的类型，可采用如下值之一：submit、reset 或 button
name	用于设置按钮的名称。如果同一个表单中具有多个按钮，则仅需要向按钮添加 name 属性(在这种情况下，该属性用于指出哪一个按钮被单击)。但是，使用该属性以任何方式指明按钮执行的操作则是优秀的实践

(续表)

属　性	目　的
value	用于指定按钮上显示的文本。如果给定一个 name 属性，则 value 属性的值作为该表单控件的“名/值”对的一部分发送给服务器。如果没有给定值，则该按钮没有“名/值”对发送
size	用于指定按钮的宽度，单位为像素，但是 Firefox 2 和 IE7 浏览器不支持这个属性
onclick	用于当用户单击按钮时触发一个脚本：这个属性的值是应该运行的脚本

当按钮获得或丢失焦点时，利用 onfocus 或 onblur 事件属性可以触发一个脚本，其方式与在用户单击按钮时触发脚本相同。

当<input>元素的 type 属性值为 submit、reset 或 button 时，它也可以采用如下属性：

- 所有的通用属性
- disabled、readonly、tabindex 和 accesskey，本章后面将介绍它们
- UI 事件属性

如果提交按钮不具有 value 属性，则浏览器中所显示的文本可能不适合于表单的目的——例如，IE 浏览器中显示的文本为 Send Query，这并不是登录按钮表单的理想文本。

2. 使用图像作为按钮

可以使用图像作为按钮，而不是使用浏览器生成的标准按钮。创建图像按钮的方式与创建其他按钮的方式非常相似，不同之处是 type 属性的值为 image：

```
<input type="image" src="submit.jpg" alt="Submit" name="btnImageMap" />
```

注意：

按钮的 name 属性值可以以字符 btn 开头，以便遵循前面提到的命名约定(当在其他代码中引用表单控件的名称时，使用这种前缀有助于让用户迅速获知作为信息来源的表单控件的类型)。

因为创建的按钮具有一幅图像，所以需要具有两个额外的属性，如表 5-4 所示。

表 5-4

属　性	目　的
src	指定图像文件的来源
alt	为图像提供可选的文本。当无法找到图像时显示该文本，该文本也对语音浏览器有所帮助(在 IE 5 和 Netscape 6 浏览器中才开始支持该属性)

如果图像按钮具有 name 属性，当单击该按钮时，浏览器将向服务器发送一个“名/值”对。名称是用户所提供的 name 属性；值是一对 x 和 y 坐标，它代表用户所单击的按钮上的位置(类似于第 3 章中处理服务器端图像映射时执行的操作)。

图 5-7 中给出了一个图形提交按钮。当用户将鼠标指针悬停在该按钮上时，Firefox 和

IE 浏览器将改变光标的外观以作为该按钮可用性的提示。

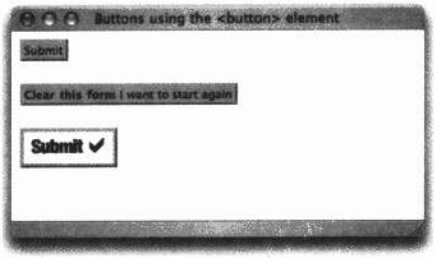


图 5-7

3. 使用<button>元素创建按钮

<button>元素是最近引入的元素，它允许用户在起始标签<button>和结束标签</button>之间指定按钮上显示的文本。因此，可以在这两个标签之间包含文本标记或者图像元素。

在 IE 4 和 Netscape 6 浏览器中才开始支持该元素，但是支持该元素的浏览器也提供一种浮雕(或 3D)效果的按钮，当单击按钮时将出现向上或向下移动的效果。

下面是使用<button>元素的一些示例(ch06_eg06.html)：

```
<button type="submit">Submit</button>
<br /><br />
<button type="reset"><b>Clear this form</b> I want to start again</button>
<br /><br />
<button type="button"><img src="submit.gif" alt="submit" /></button>
```

其中第一个提交按钮只包含文本，第二个复位按钮包含文本和其他标记(以<b>元素的形式)，第三个提交按钮包含一个<img>元素。

图 5-8 给出了这些按钮在浏览器中的外观。

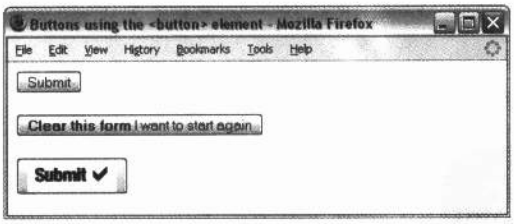


图 5-8

5.3.3 复选框

复选框类似于纸张表单上必须选中的小方框。与电灯开关一样，复选框可以是开状态或者关状态。当复选框被选中时处于开状态，用户可以通过单击复选框来切换开和关状态。

复选框可以单独出现，每一个复选框具有自己的名称；多个复选框可以以组的形式出现，它们共享同一个控件名称并允许用户为相同的特性选择多个值。

当需要为用户提供如下功能时，复选框是理想的表单控件：

- 利用一个控件提供简单的是或否响应(例如接受一些条款和条件或者订阅 e-mail 列表)
- 从可能的选项的列表中选择其中的几项(例如让用户从给定列表中选择他们具有的所有技能)

可以使用<input>元素创建复选框，将该元素的 type 属性值设置为 checkbox。

下面是几个使用相同控件名称的复选框的示例(ch05_eg07.html)：

```
<form action="http://www.example.com/cv.aspx" method="get" name="frmCV">
Which of the following skills do you possess? Select all that apply.
  <input type="checkbox" name="chkSkills" value="html" />HTML <br />
  <input type="checkbox" name="chkSkills" value="xhtml" />XHTML <br />
  <input type="checkbox" name="chkSkills" value="CSS" />CSS<br />
  <input type="checkbox" name="chkSkills" value="JavaScript"
/>JavaScript<br />
  <input type="checkbox" name="chkSkills" value="aspnet" />ASP.Net<br />
  <input type="checkbox" name="chkSkills" value="php" />PHP
</form>
```

为了在本章中保持一致的表单元素命名约定，可以将复选框名称的起始字符设置为 chk。图 5-9 给出了这个表单在浏览器中的外观。注意，在每一个复选框后面有一个换行，这样每一个复选框将单独占用一行(如果并排放置复选框，则用户很可能不知道哪一个标签对应于哪一个复选框)。

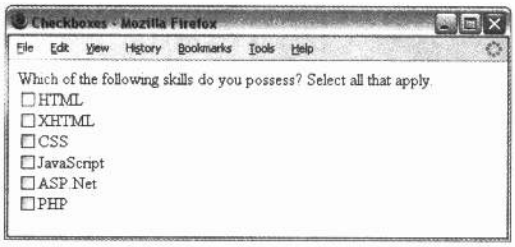


图 5-9

因为所有选中的技能都将以“名/值”对的形式发送给处理应用程序，所以如果有人选择了多个技能，将有多个“名/值”对发送给服务器，并且它们共享相同的名称。

注意：

处理具有相同名称的多个复选框的方式取决于向服务器发送数据的方法。如果使用 HTTP get 发送数据，则选中的复选框将作为查询字符串中 URL 的一部分发送。然而，如果使用 HTTP post 方法发送数据，则将获得一个表示已选中选项的可遍历数组。

作为对比，下面是单个复选框，它提供简单的是或否选项：

```
<form action="http://www.example.org/accept.aspx" name="frmTandC"
method="get">
  <input type="checkbox" name="chkAcceptTerms" checked="checked" />
```

```
    I accept the <a href="terms.htm">terms and conditions</a>.<br />
  <input type="submit" />
</form>
```

注意，创建这个复选框的<input>元素没有附带 value 属性。没有附带 value 属性时，值是 on。在这个示例中，也有一个称为 checked 的属性，它的值是 checked，用于指明当页面加载时该复选框已被选中。

注意：

在 HTML4.1 之前，可以提供不带有值的 checked 属性。这种过程称为属性最小化，附带不具有值的 checked 属性的元素被认为是开状态。在 XHTML 中，所有的属性必须具有值，因此重复这些属性的名称作为它们的值。较老的浏览器可能会忽略该值，但是它们仍然会确认该属性的存在。

表 5-5 给出了 type 属性值为 checkbox 的<input>元素能够附带的属性。

表 5-5

属 性	目 的
type	指明希望创建的是复选框
name	赋予控件名称。多个复选框可以共享相同的名称，但前提是希望让用户从同一个列表中选择多项——此时，这些复选框应当依次放置在表单中
value	如果复选框被选中，则该值将被发送给服务器
checked	指明当页面加载时复选框应该被选中
size	指明复选框的大小，单位为像素(但在 IE 7 或 Firefox 2 浏览器中不支持该属性)

复选框也可以附带如下属性：

- 所有通用属性
- disabled、readonly、tabindex 和 accesskey，本章后面将介绍它们
- UI 事件属性

5.3.4 单选按钮

单选按钮与复选框的相似之处是其状态也是开或关，但是主要区别在于以下两点：

- 当存在一组共享相同名称的单选按钮时，仅可以从其中选择一个单选按钮。当选择某个单选按钮之后，如果用户单击另一个选项，则新的选项将被选中，原有的选项将被取消选中。
- 不能将单选按钮用作单个表单控件来指明开或关状态，因为此时一旦选中该单选按钮，将无法取消选择它(除非编写脚本)。

因此，如果希望为用户提供多个选项，并且一次只能选择其中一个选项，则单选按钮是理想的选择。在这种情况下，一种替代方法是使用下拉选项框，下拉选项框允许用户从多个选项中仅选择一项。是使用一个下拉选项框还是使用一组单选按钮，依赖于下面 3 个方面：

- 用户的期望：如果表单模仿纸张表单，该纸张表单提供多个复选框，并且用户只能从中选择一个复选框，则应当使用一组单选按钮。
- 查看所有选项的需要：如果用户预先看到所有的选项能够有助于他们进行选择，应当使用一组单选按钮。
- 空间的需要：如果空间较小，则下拉选项框将比一组单选按钮节省空间。

注意：

术语“单选按钮”来自于老式收音机。在某些老式收音机中，用户一次仅能按下一个按钮来选择一个事先设置好的广播台，而不能同时按下收音机中的两个按钮，按下其中一个按钮将弹起另一个按钮。

仍然需要使用<input>元素来创建单选按钮，此时该元素的 type 属性值将被设置为 radio。例如，下面的单选按钮用于允许用户选择他们想要采用的旅行方式(ch05_eg08.html)：

```
<form action="http://www.example.com/flights.aspx" name="frmFlightBooking"
      method="get">
    Please select which class of travel you wish to fly: <br />
  <input type="radio" name="radClass" value="First" />First class <br />
  <input type="radio" name="radClass" value="Business" />Business class <br />
  <input type="radio" name="radClass" value="Economy" />Economy class <br />
</form>
```

其中只允许用户选择 3 个选项中的一个，因此单选按钮是理想的控件。本书将单选按钮名称的起始字符设置为 rad。图 5-10 中给出了这段代码在浏览器中的外观。

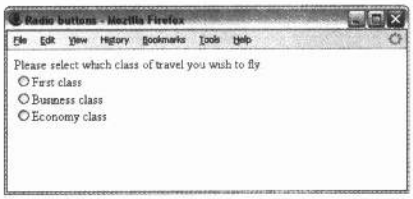


图 5-10

表 5-6 给出了 type 属性值为 radio 时<input>元素所能附带的属性。

表 5-6

属　性	目　的
type	指明创建的是单选按钮表单控件
name	赋予表单控件的名称
value	用于指示选择该选项时将发送给服务器的值
checked	指明当页面加载时应该默认选中该选项。记住，不要使用单个单选按钮，因为此时用户无法取消选择该选项。如果使用这个属性，则对于 XHTML 兼容的属性来说，也应当选中值
size	该属性指明单选按钮的大小，单位为像素，但是在 IE 7 或 Firefox 2 浏览器中不支持该属性

单选按钮也可以附带如下属性：

- 所有的通用属性
- 所有 UI 事件属性
- disabled、tabindex 和 accesskey，本章后面将介绍它们

注意：

如果具有一组共享相同名称的单选按钮，当页面加载时某些浏览器将自动选择第一个选项——尽管在 HTML 规范中没有要求这一点。因此，如果单选按钮表示一组值——例如用于投票应用程序——则最好选择一个中间选项作为默认值，以防止某些用户忘记选择其中一项时，浏览器选择的结果不会存在什么偏见。为此，需要使用 checked 属性。

5.3.5 选项框

下拉选项框允许用户从下拉菜单中选择一项。下拉选项框比一组单选按钮占用的空间要少得多。

对于单行文本输入控件，可以采用下拉选项框代替，此时可以限制用户输入的选项。例如，可以使用选项框允许用户指示他们居住的国家或州(优点是所有来自美国的用户将具有相同的值，而不是可能会填写 U.S.A.、U.S.、United States、America 或 North America，然后不得不处理同一个国家的多种答案)。

下拉选项框包含在一个<select>元素中，选项框中的每一个选项则包含在一个<option>元素中。例如，下面的表单为用户创建的下拉选项框用于选择一种颜色(ch05_eg09.html)：

```
<select name="selColor">
  <option selected="selected" value="">Select color</option>
  <option value="red">Red</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
</select>
```

其中起始标签<option>和结束标签</option>之间的文本用于向用户显示选项，如果某个选项被选中，则它的值(在 value 属性中提供)将被发送给服务器。第一个<option>元素没有值，而它的内容是 Select color；这表明用户必须选择一种颜色。最后再次注意，这里使用字符 sel 作为选项框名称的起始字符。

图 5-11 给出了这段代码在浏览器中的外观。

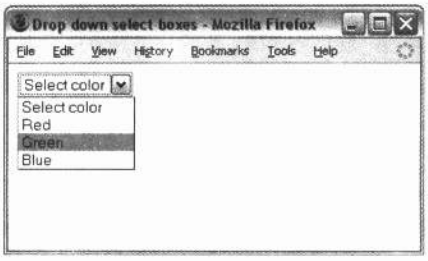


图 5-11

注意，选项框的宽度将是显示给用户的最长选项的宽度；在这个示例中，选项框的宽度是文本 Select color 的宽度。

1. <select>元素

<select>元素是下拉列表框的包含元素，它可以采用表 5-7 所示的属性。

表 5-7

属　性	目　的
name	控件的名称
size	可用于表示一个滚动列表框，后面将介绍滚动列表框。它的值是列表中同时可见的行数
multiple	允许用户从菜单中选择多个项。如果没有该属性，则用户将只能选择其中一项。在早期的 HTML 版本中，这个属性不具有值。但是，在有效的 XHTML 中，该属性必须被赋予值 multiple(例如<select multiple="multiple">)。注意，使用这个属性将改变选项框的外观，在本章后面的“利用 multiple 属性选择多个选项”一节中将对此进行介绍

HTML 和 XHTML 规范中指明，<select>元素必须至少包含一个<option>元素，但实际上它通常应该包含多个<option>元素。毕竟，如果下拉列表框只包含一个选项，将会使用户感到困惑。

2. <option>元素

在任何<select>元素中至少具有一个<option>元素。起始标签<option>和结束标签</option>之间的文本将显示为该选项的标签。<option>元素可以采用表 5-8 所示的属性。

表 5-8

属　性	目　的
value	选项的值。如果选项被选中，它的值将被发送给服务器
selected	指定某个选项作为页面加载时初始选中的选项。这个属性也可以用于多个<option>元素，即使<select>元素没有附带 multiple 属性。在早期的 XHTML 版本中，这个属性不需要具有值。但为了在 XHTML 中有效，需要将这个属性的值设置为 selected
label	赋予选项标签的一种替代方式，即通过一种属性而不是元素内容赋予选项标签。当使用<optgroup>元素时，这个属性非常有用，本章后面将介绍<optgroup>元素

3. 创建滚动选项框

前面已经提到，还可以创建滚动菜单。在滚动菜单中，用户一次能够在选项框中看到多个选项。为了实现这个功能，只需要在<select>元素中添加 size 属性。size 属性的值是用户一次能够看到的选项数量。

虽然滚动选项框菜单很少使用，但是它们能够向用户指示存在非常多的选项，并且允许用户一次看到其中的几个选项。例如，下面的滚动选项框允许用户选择一周中的某一天(ch05_eg10.html)：

```
    <form action="http://www.example.org/days.aspx" name="frmDays"
method="get">
      <select size="4" name="selDay">
        <option value="Mon">Monday</option>
        <option value="Tue">Tuesday</option>
        <option value="Wed">Wednesday</option>
        <option value="Thu">Thursday</option>
        <option value="Fri">Friday</option>
        <option value="Sat">Saturday</option>
        <option value="Sun">Sunday</option>
      </select>
    <br /><br /><input type="submit" value="Submit" />
    </form>
```

从图 5-12 中可以看到，这种方式为用户提供了多个选项，并且通过一次仅为用户提供少量选项限制了选项框的空间占用量。

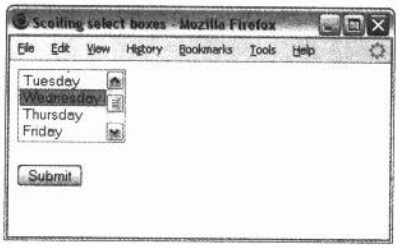


图 5-12

注意，后面小节中讨论的 multiple 属性不用于这个元素。

4. 利用 multiple 属性选择多个选项

multiple 属性允许用户从选项框中一次选择多个项。multiple 属性的值必须是单词 multiple，以便使得编写的 XHTML 代码有效(但是早期的 HTML 版本不需要该属性具有值)。

添加这个属性后，将自动使选项框看上去类似于滚动选项框。下面是一个多项选项框的示例，它允许用户选择一周中的多天(ch05_eg11.html)：

```
    <form action="http://www.example.org/days.aspx" method="get"
name="frmDays">
      Please select more than one day of the week:<br />
        <select name="selDays" multiple="multiple">
        <option value="Mon">Monday</option>
        <option value="Tue">Tuesday</option>
        <option value="Wed">Wednesday</option>
        <option value="Thu">Thursday</option>
```

```
    <option value="Fri">Friday</option>
    <option value="Sat">Saturday</option>
    <option value="Sun">Sunday</option>
  </select>
<br /><br /><input type="submit" value="Submit">
</form>
```

结果如图 5-13 所示。在没有添加 size 属性的情况下，该选项框仍然是一个滚动选项框。

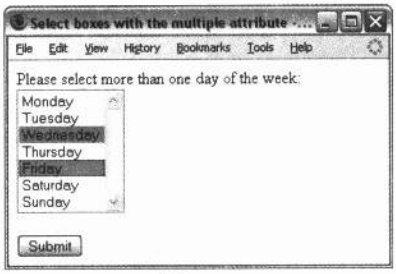


图 5-13

5. 利用<optgroup>元素分组选项

如果选项框中的选项非常多，可以使用<optgroup>元素将它们分组到一起，该元素类似于一个容器元素，容器元素用于将多个元素包含在一个组中。

<optgroup>元素可以附带一个 label 属性，它的值是选项组的标签。在下面的示例中，按照设备的类型将不同选项分组(ch05_eg12.html)：

```
    <form action="http://www.example.org/info.aspx" method="get"
name="frmInfo">
      Please select the product you are interested in:<br />
      <select name="selInformation">
        <optgroup label="Hardware">
            <option value="Desktop">Desktop computers</option>
            <option value="Laptop">Laptop computers</option>
        </optgroup>
        <optgroup label="Software">
            <option value="OfficeSoftware">Office software</option>
            <option value="Games">Games</option>
        </optgroup>
        <optgroup label="Peripherals">
            <option value="Monitors">Monitors</option>
            <option value="InputDevices">Input Devices</option>
            <option value="Storage">Storage</option>
        </optgroup>
      </select>
      <br /><br /><input type="submit" value="Submit" />
      </form>
```

不同的浏览器通常以不同的方式显示<optgroup>元素。图 5-14 给出了该示例在 Mac 机器上的 Safari 浏览器中的外观，而图 5-15 给出了该示例在 PC 机器上的 Firefox 浏览器中

的外观。

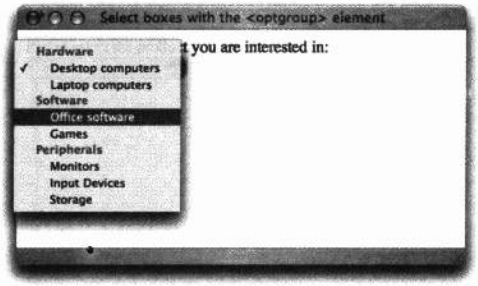


图 5-14

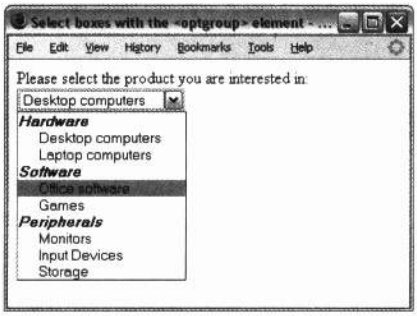


图 5-15

分组元素的另外一种替代方式是利用附带 disabled 属性的<option>元素，本章后面将介绍该属性(ch05_eg13.html)：

```
    <form action="http://www.example.org/info.aspx" method="get"
name="frmInfo">
      Please select the product you are interested in:<br />
      <select name="selInformation">
        <option disabled="disabled" value=""> -- Hardware -- </option>
          <option value="Desktop">Desktop computers</option>
          <option value="Laptop">Laptop computers</option>
        <option disabled="disabled" value=""> -- Software -- </option>
          <option value="OfficeSoftware">Office software</option>
          <option value="Games">Games</option>
        <option disabled="disabled" value=""> -- Peripherals -- </option>
          <option value="Monitors">Monitors</option>
          <option value="InputDevices">Input Devices</option>
          <option value="Storage">Storage</option>
      </select>
      <br /><br /><input type="submit" value="Submit" />
      </form>
```

在本章后面将看到，使用 disabled 属性可阻止用户选择附带该属性的选项。通过使用

一对破折号，选项组将变得更清晰，如图 5-16 所示。

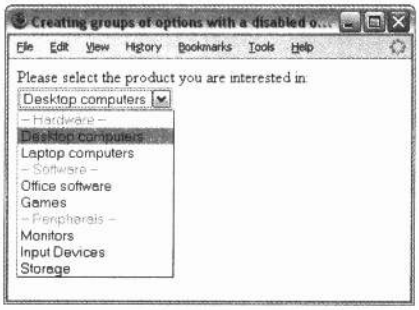


图 5-16

注意：

如果决定使用选项框作为导航的一部分，例如为了允许用户快速跳转到站点的某个部分，则也应当在页面中包含一个提交按钮或执行按钮。但是，在用户选择了某个选项之后，避免使用 JavaScript 自动将他们带到相关页面。在这种情况下，使用 JavaScript 提交表单通常被认为是可用性极差的示例。主要原因之一是用户可能无意中选择了错误的部分；例如，如果用户使用向上或向下方向键选择选项，脚本将在用户遇到第一个选项时被触发。另外，某些浏览器可能不支持脚本，这就需要全面检查不同的平台和不同的浏览器。

6. 选项框的属性

为了实现介绍的完整性，下面是<select>元素能够附带的属性的完整列表。

- name、size 和 multiple，前面已经介绍过它们
- disabled 和 tabindex，本章后面将介绍它们
- 所有的通用属性
- UI 事件属性

同时，<option>元素可以附带如下属性：

- label，前面已经介绍过该属性
- disabled，本章后面将介绍该属性
- 所有的通用属性
- UI 事件属性

5.3.6 文件选项框

如果希望允许用户从他的计算机上传文件到 Web 站点，则需要使用文件上传框，也称为文件选项框。可以(再次)使用<input>元素创建文件选项框，该元素的 type 属性值需要被设置为 file(ch05_eg14)：

```
<form action="http://www.example.com/imageUpload.aspx" method="post"
      name="fromImageUpload" enctype="multipart/form-data">
   <input type="file" name="fileUpload" accept="image/*" />
<br /><br /><input type="submit" value="Submit" />
</form>
```

注意：

当使用文件上传框时，<form>元素的 method 属性值必须是 post。

在这个示例中有一些属性，在本章的开头部分提到过它们。

- <form>元素中添加了 enctype 属性，该属性具有值 multipart/form-data，从而每个表单控件将被独立地发送给服务器。使用文件上传框的表单要求具有该属性。
- <input>元素中添加了 accept 属性，用于指示能够选择上传的文件的 MIME 类型。在这个示例中，任何图像格式都可以上传，因为在 MIME 类型的 image/部分之后使用了通配符(星号)。遗憾的是，Firefox 2 和 IE 7 浏览器不支持该属性。

在图 5-17 中，当单击 Browse 按钮时将打开一个文件对话框，利用它能够浏览文件并选择一个文件上传。

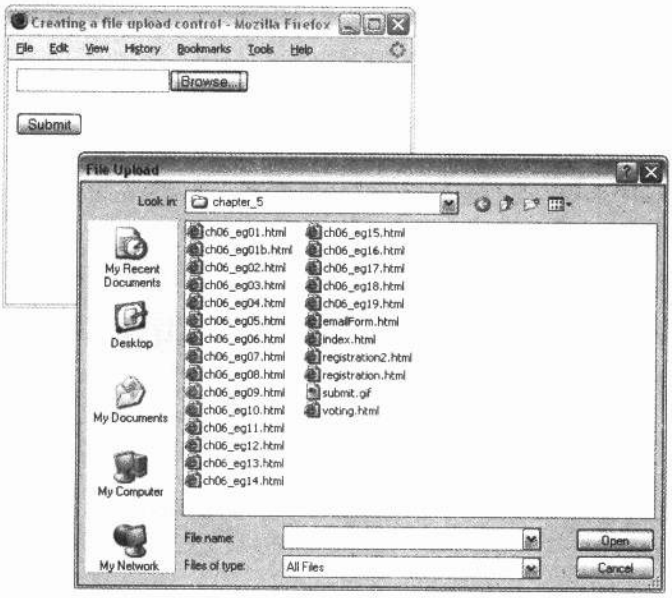


图 5-17

type 属性值为 file 的<input>元素能够采用如下属性：

- name、value 和 accept，前面已经介绍过它们
- tabindex、accesskey、disabled 和 readonly，本章后面将介绍它们
- 所有的通用属性
- UI 事件属性

5.3.7 隐藏控件

有时您会希望在页面间传递信息并且不让用户看到它们；此时可以使用隐藏表单控件。然而，重点需要注意的是，虽然用户不能在浏览器中显示的 Web 页面上看到隐藏表单控件，但如果用户查看页面的源代码，则他们将能够看到源代码中的值。因此，隐藏控件

不能用于不希望让用户看到的敏感信息。

注意：

您可能在 Web 上看到过一些多页的表单。较长的表单通常会带来混淆；将它们划分为多个部分将有助于用户识别，这意味着用户需要填写多个表单。此时，Web 站点程序员通常希望在第一个表单(位于第一个页面中)和后续页面中的表单之间传递用户输入的值。隐藏元素是程序员可用于在页面之间传递值的方法之一。

创建隐藏控件的方法是使用<input>元素，该元素的 type 属性值需要设置为 hidden。例如，下面的表单包含一个隐藏表单控件，它指明用户在站点的哪一部分中填写了表单(ch05_eg15.html)：

```
    <form action="http://www.example.com/vote.aspx" method="get"
name="fromVote">
        <input type="hidden" name="hidPageSentFrom" value="home page" />
        <input type="submit" value="Click if this is your favorite page of our
        site." />
    </form>
```

隐藏表单控件的名称和值仍然能够发送给服务器，此时隐藏表单控件必须附带 name 属性和 value 属性。

图 5-18 表明隐藏表单控件没有显示在页面上，但可以在页面的源代码中看到它。

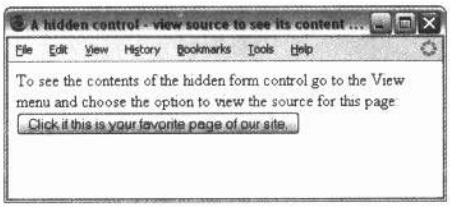


图 5-18

注意：

在第 8 章中您将看到，也可以利用 CSS 的 display 属性和 visibility 特性隐藏表单控件。

5.3.8 对象控件

HTML 4.0 规范中引入了使用对象作为表单一部分的能力——嵌入<object>元素。为了成为有效的表单控件，<object>元素必须位于<form>元素中。例如，您可能希望使用一个具有某种图形交互功能的对象，然后存储对象的名称和它的值。但是在编写本书时，主要的浏览器都没有实现这个功能。

试一试　创建一个注册表单

在这个示例中，需要结合几个表单控件以组成一个站点注册表单。

(1) 创建一个新的 Transitional XHTML 1.0 文档，并且添加它的程序框架。然后添加题头和表明用户应该执行的操作的简介：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Registration</title>
</head>
<body>
  <h2>User Registration</h2>
  <p>Please complete the following form to register with our site:</p>
</body>
</html>
```

(2) 表单放置在一个具有两列的表中，指令位于左边的一列，表单控件在右边一列中对齐(如果不这样做，表单控件在页面中将显得不均匀)。这是编写表单的常用技巧。

在前两行中可以添加文本输入框，分别用于输入用户名和密码，然后留一行空行用于隔开页面：

```
<table>
  <tr>
    <td>User Name:</td>
    <td><input type="text" name="txtUserName" size="20" /></td>
  </tr>
  <tr>
    <td>Password:</td>
    <td><input type="password" name="pwdPassword" size="20" /></td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
  </tr>
</table>
```

(3) 在用户名和密码之后添加两个单选按钮，以便用户选择他或她的性别：

```
<tr>
  <td>Gender:</td>
  <td><input type="radio" name="radSex" value="male" />Male</td>
</tr>
<tr>
  <td></td>
  <td><input type="radio" name="radSex" value="female" />Female</td>
</tr>
<tr><td> </td><td> </td></tr>
```

(4) 接下来添加一个选项框，让用户选择如何知道这个 Web 站点：

```
<tr>
  <td>How did you hear about us?:</td>
  <td>
    <select name="selReferrer">
      <option selected="selected" value="">Select answer</option>
      <option value="website">Another website</option>
      <option value="printAd">Magazine ad</option>
      <option value="friend">From a friend</option>
      <option value="other">Other</option>
    </select>
  </td>
</tr>
<tr><td> </td><td> </td></tr>
```

(5) 最后一个选项是让用户选择他们是否订阅站点的时事通讯，可以使用一个复选框完成该功能。另外，表单中也需要有一个提交按钮：

```
<tr>
  <td>Please select this box if you wish<br /> to be added to our mailing list
      <br /><small>We will not pass on your details to any third
      party.</small></td>
  <td><input type="checkbox" name="chkMailingList" /></td>
</tr>
<tr>
  <td></td>
  <td><input type="submit" value="Register now" /></td>
</tr>
</table>
```

(6) 将文件存储为 registration.html 并在浏览器中打开它；结果如图 5-19 所示。

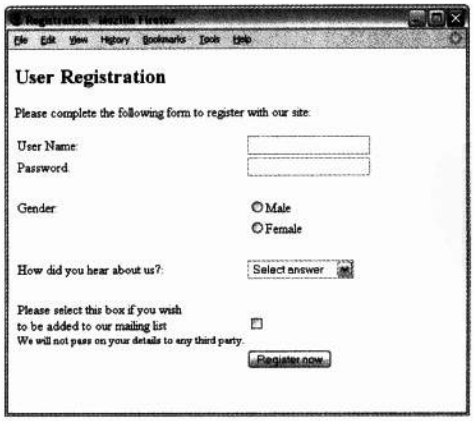


图 5-19

工作原理

这是一个使用多个表单控件的表单示例，这里仅需要注意其中的控件。

首先，存在两个文本输入框，分别用于输入用户名和密码。因为密码是敏感信息，这里使用了一个密码类型的文本输入框，它能够防止站在用户身后的任何人看到用户的输入。记住，其中的 size 属性控制输入框的宽度。

```
<tr>
  <td>User Name:</td>
  <td><input type="text" name="txtUserName" size="20" /></td>
</tr>
<tr>
  <td>Password:</td>
  <td><input type="password" name="pwdPassword" size="20" /></td>
</tr>
<tr>
```

接下来，两个单选按钮用于指示用户的性别。两个单选按钮具有相同的名称，以进行互斥的选择——用户仅可以选择两个单选按钮中的一个。当用户选择某个选项时，该元素的 value 属性值将被发送给服务器。

```
<tr>
  <td>Gender:</td>
  <td><input type="radio" name="radSex" value="male" />Male</td>
</tr>
<tr>
  <td></td>
  <td><input type="radio" name="radSex" value="female" />Female</td>
</tr>
```

在选项框中，用户必须选择他们如何知道这个站点。第一个选项被自动选中，这是因为它附带 selected 属性，这也暗示用户必须选择其中的一个选项。所选择项的值将与名称 selReferrer 一起被发送给服务器。

```
<select name="selReferrer">
  <option selected="selected" value="">Select answer</option>
  <option value="website">Another website</option>
  <option value="printAd">Magazine ad</option>
  <option value="friend">From a friend</option>
  <option value="other">Other</option>
</select>
```

最后使用一个复选框让用户选择是否订阅站点的时事通讯。

```
<tr>
  <td>Please select this box if you wish<br /> to be added to our mailing list
     <br /><small>We will not pass on your details to any third
     party.</small></td>
  <td><input type="checkbox" name="chkMailingList" /></td>
</tr>
```

为了发送表单，用户必须单击 Register now 按钮。此处显示单词 Register now 是因为它们作为 value 属性的值提供：

```
<tr>
  <td></td>
  <td><input type="submit" value="Register now" /></td>
</tr>
```

现在您已经了解表单的基础知识，可以进一步掌握一些关于表单的高级功能，以用于增强您的表单。

5.4 利用<label>元素为控件创建标签

在创建表单时，非常重要的一点是为用户提供优秀的标签，以便他们知道在何处应该输入什么样的数据。

填写表单时，人们通常无从下手；例如，保险公司表单或税务表单都难以填写。因此，直到访问者完全确定他们应当在表单中填写什么信息之后，他们才会倾向于填写表单。

某些表单控件，例如按钮，已经具有标签。但是，对于大多数表单控件来说，需要程序员提供标签。

对于没有标签的控件，应该使用<label>元素赋予它们标签。这个元素仅用于告诉用户应当输入什么信息，而不会以任何方式影响表单(ch05_eg16.html)。

注意：

这个表单放置在一个表中；这就确保即使标签具有不同的长度，文本输入框也会在它们自己的列中对齐。如果多个文本输入框的缩进程度不同，则使用起来非常不便。

```
    <form action="http://www.example.org/login.aspx" method="post"
name="frmLogin">
      <table>
        <tr>
          <td><label for="Uname">User name</label></td>
          <td><input type="text" id="Uname" name="txtUserName" /></td>
        </tr>
        <tr>
          <td><label for="Pwd">Password</label></td>
          <td><input type="password" id="Pwd" name="pwdPassword" /></td>
        </tr>
      </table>
    </form>
```

其中，<label>元素附带称为 for 的属性，该属性指明哪一个表单控件与该标签相关联。for 属性的值必须与对应表单控件的 id 属性的值相同。例如，对于文本框表单控件(用户需要在该表单控件中输入用户名)，它的 id 属性值是 Uname，因此该文本框的对应标签的 for 属性值也应当是 Uname。

图 5-20 给出该登录页面在浏览器中的外观。

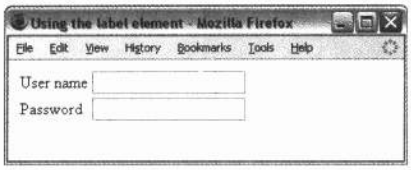


图 5-20

标签的位置可以在控件之前，也可以在控件之后。对于文本框，通常将标签放置在它的左边；而对于复选框和单选按钮，如果将标签放置在右边，则通常更容易将其与正确的表单控件相关联。

注意：

对每个表单控件应当使用一个新的<label>元素。

<label>元素的另外一种使用方式是作为包含元素，这种类型的标签有时称为隐式标签。例如：

```
    <form action="http://www.example.org/login.asp" method="post"
name="frmLogin">
        <label for="Uname"><input type="text" id="Uname" name="txtUserName"
/></label>
        <label for="Pwd"><input type="password" id="Pwd" name="pwdPassword"
/></label>
    </form>
```

这种方式的不足之处在于，用户无法控制标签相对于表单控件的显示位置，并且无法让标签与表单控件位于不同的表单元格，因为标记无法正确嵌套。

注意：

在本章后面的“焦点”一节中将介绍如何赋予表单元素焦点。当标签获得焦点时，该焦点将被传递给关联的控件。

5.5 利用<fieldset>元素和<legend>元素结构化表单

大型表单会给用户带来混淆，因此最好将相关的表单控件进行分组。使用<fieldset>元素和<legend>元素实现该功能——它们帮助分组控件。

这两个元素都在 IE4 和 Netscape 6 浏览器中引入；然而，较老的浏览器仅是忽略它们，因此可以安全地在所有的表单中包含它们。

- <fieldset>元素在表单控件组周围创建边框，以表明这些表单控件是相关的。
- <legend>元素允许用户为<fieldset>元素指定一个标题，该标题作为表单控件组的名称。使用<legend>元素时，它必须是<fieldset>元素的第一个子元素。

在下面的示例中，将一个表单划分为 4 个部分：联系人信息、竞赛问题、加时赛问题以及输入竞赛(ch05_eg17.html)：

```
    <form action="http://www.example.org/competition.asp" method="post"
name="frmComp">
      <fieldset>
        <legend><em>Contact Information</em></legend>
          <label>First name: <input type="text" name="txtFName" size="20"
/></label><br />
          <label>Last name: <input type="text" name="txtLName" size="20"
/></label><br />
          <label>E-mail: <input type="text" name="txtEmail" size="20"
/></label><br />
        </fieldset>
        <fieldset>
        <legend><em>Competition Question</em></legend>
          How tall is the Eiffel Tower in Paris, France? <br />
          <label><input type="radio" name="radAnswer" value="584" />
             584ft</label><br />
          <label><input type="radio" name="radAnswer" value="784" />
             784ft</label><br />
          <label><input type="radio" name="radAnswer" value="984" />
             984ft</label><br />
          <label><input type="radio" name="radAnswer" value="1184" />
             1184ft</label><br />
        </fieldset>
        <fieldset>
          <legend><em>Tiebreaker Question</em></legend>
          <label>In 25 words or less, say why you would like to win $10,000:
             <textarea name="txtTiebreaker" rows="10" cols="40"></textarea>
          </label>
      </fieldset>
      <fieldset>
        <legend><em>Enter competition</em></legend>
          <input type="submit" value="Enter Competition" />
      </fieldset>
    </form>
```

其中<fieldset>元素在表单控件组的周围创建边框，<legend>元素赋予表单控件组名称。需要强调的是，使用的<legend>元素必须是<fieldset>元素的第一个子元素。结果如图 5-21 所示。

<fieldset>元素可以附带如下属性：

- 所有通用属性
- 基本事件属性

注意：

如果使用表来格式化表单，则<table>元素必须在<fieldset>元素之内。如果<fieldset>元素在用于格式化页面的表之内，则整个字段集必须驻留在一个单元格内。

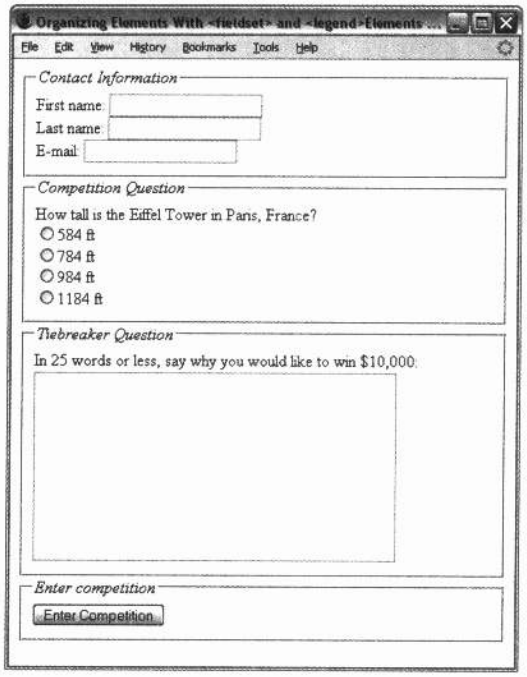


图 5-21

<legend>元素可以附带如下属性：

- accesskey，下一节中将介绍该属性
- align(该属性将逐渐淘汰——应当使用 CSS 的定位功能)
- 所有通用属性
- UI 事件属性

5.6 焦点

当 Web 页面中具有多个链接或者多个表单控件时，您可能已经注意到，可以使用 Tab 键在这些元素间进行移动(或者使用 Shift+Tab 以相反的顺序移动)。当在元素之间进行移动时，Web 浏览器趋向于在元素上添加一些形式的边框或以高亮显示元素(假设该元素是链接或表单控件)。这种方式称为接收焦点。

根据您对 XHTML 已经了解的知识，并不是文档中的每一个元素都接收焦点。事实上，仅是用户能够交互的元素才可以接收焦点，例如链接和表单控件。事实上，如果用户期望与一个元素交互，则该元素必须能够接收焦点。

元素获得焦点的方式有 3 种：

- 使用定点设备(例如鼠标或轨迹球)选择元素。

- 使用键盘在元素之间进行切换——通常使用 Tab 键(或者使用 Shift+Tab 键在元素间反向切换)。某些文档中的元素可以被赋予固定的焦点移动顺序，这样能够指示元素获得焦点的顺序。
- 可以使用类似键盘快捷键的系统(称为访问键)选择某个特定元素。例如，在 PC 计算机上可以按下 Alt 键加上一个访问键(例如 Alt+E)，而在 Mac 计算机上可以按下 Control 键加上一个访问键(例如 Control+E)。

5.6.1 焦点移动顺序

如果希望控制元素获得焦点的顺序，可以使用 tabindex 属性赋予元素一个从 0 到 32 767 之间的数值，从而组成焦点移动顺序的一部分。每次用户按下 Tab 键时，焦点将移动到下一个具有最高焦点移动顺序的元素上(再次提醒，按下 Shift+Tab 键将以相反顺序移动焦点)。

下面的元素可以附带 tabindex 属性：

```
<a> <area> <button> <input> <object> <select> <textarea>
```

Netscape 6 和 IE 4 浏览器中最早开始支持 tabindex 属性，但是较老的浏览器仅是忽略这个属性，因此可以安全地在所有的文档中使用该属性。

在用户通过按下 Tab 键在文档中所有能够获得焦点的元素之间依次切换之后，焦点将返回给浏览器的某个功能(通常是地址栏)。

为了演示焦点移动顺序的原理，下面的示例赋予复选框不同的焦点移动顺序(ch05_eg18.html)：

```
    <form action="http://www.example.com/tabbing.asp" method="get"
      name="frmTabExample">
      <input type="checkbox" name="chkNumber" value="1" tabindex="3" /> One<br />
      <input type="checkbox" name="chkNumber" value="2" tabindex="7" /> Two<br />
      <input type="checkbox" name="chkNumber" value="3" tabindex="4" />
Three<br />
      <input type="checkbox" name="chkNumber" value="4" tabindex="1" /> Four<br />
      <input type="checkbox" name="chkNumber" value="5" tabindex="9" /> Five<br />
      <input type="checkbox" name="chkNumber" value="6" tabindex="6" /> Six<br />
      <input type="checkbox" name="chkNumber" value="7" tabindex="10" />Seven
<br />
      <input type="checkbox" name="chkNumber" value="8" tabindex="2" />
Eight<br />
      <input type="checkbox" name="chkNumber" value="9" tabindex="8" /> Nine<br />
      <input type="checkbox" name="chkNumber" value="10" tabindex="5" /> Ten<br />
    <input type="submit" value="Submit" />
    </form>
```

在这个示例中，复选框接收焦点的顺序为：

```
4, 8, 1, 3, 10, 6, 2, 9, 5, 7
```

图 5-22 给出了 PC 计算机上 Firefox 2 浏览器中元素获得焦点后的样式，默认情况下，

该浏览器赋予获得焦点的元素黄色框线(其他的浏览器可能显示不同样式的框线——Internet Explorer 浏览器使用蓝色线)。图 5-22 中放大获得焦点的元素，以便能够更清晰地查看它。

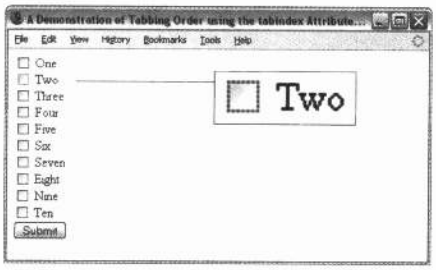


图 5-22

tabindex 值的起始数值必须是 1 或更大的值，而不能是 0，因为能够获得焦点但没有 tabindex 属性的元素会被提供值 0，在遍历具有 tabindex 属性的元素之后，会将焦点移动到这些元素，并且焦点移动的顺序为这些元素的显示顺序。如果两个元素具有相同的 tabindex 属性值，它们获得焦点的顺序将是它们在文档中显示的顺序。

注意：

如果一个元素被禁用，则它不能获得焦点，并且不参与焦点移动顺序。

5.6.2 访问键

访问键类似于键盘快捷键。访问键是文档字符集(出现在用户键盘上的字符)中的单个字符，当这个键与其他键(例如 Windows 中的 Alt 键和 Apple 中的 Control 键)一起使用时，浏览器将自动转向对应部分(具体哪一个键与访问键一起使用取决于所使用的操作系统和浏览器)。

可以利用 accesskey 属性定义访问键。这个属性的值是希望让用户按下的字符和键盘上的键(接合其他键，具体取决于所使用的操作系统和浏览器)。

下面的元素可以附带 accesskey 属性：

```
<a> <area> <button> <input> <label> <legend> <textarea>
```

accesskey 属性在 Netscape 6 和 IE 4 浏览器中获得支持，但是较老的浏览器仅是忽略这个属性，因此可以安全地在所有的文档中使用该属性。

为了了解访问键的工作原理，回顾竞赛表单示例(ch05_eg17.html)，参考本章前面的“利用<fieldset>元素和<legend>元素结构化表单”一节。现在可以在<legend>元素中添加 accesskey 属性：

```
<legend accesskey="c"><u>C</u>ontact Information (ALT + C)</legend>
<legend>Competition Question</legend>
<legend accesskey="t"><u>T</u>iebreaker Question (ALT + T)</legend>
<legend>Enter competition</legend>
```

这个文件的新版本位于下载代码的 ch05_eg19.html 文件中(在文档中添加了额外的
元素，它指出当使用访问键时屏幕如何滚动到对应的部分)。为了提示用户可以使用访问键作为快捷键，可以以两种方式将相应的信息添加到<legend>元素的信息中：

- 在标题后的方括号中
- 通过在访问键上添加下划线

图 5-23 给出了这个更新后的示例在浏览器中的外观。

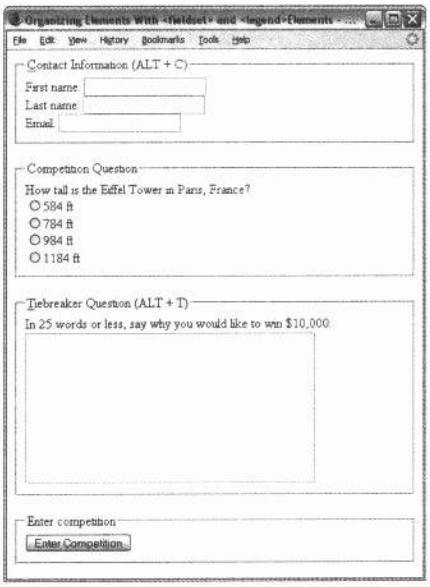


图 5-23

访问键的使用效果取决于与它一起使用的元素。对于<legend>元素，像前面介绍的一样，浏览器将自动滚动到对应的页面部分，并将焦点赋予给该部分中的第一个表单控件。当与表单控件一起使用时，这些元素将获得焦点。当元素获得焦点时，用户可以与它进行交互(例如在文本控件中输入文本，或对其他的表单控件按下 Enter 键或 Return 键)。

当使用字母 a-z 时，指定大写或小写的访问键都可行，尽管严格来说应当使用小写形式。

5.7 禁用的或只读的控件

在本章中，您已经看到几个元素能够附带称为 disabled 或 readonly 的属性：

- readonly 属性阻止用户改变表单控件本身的值，尽管可以使用脚本修改它。readonly 控件的名/值对将被发送给服务器，它的值应该是 readonly。
- disabled 属性将禁用表单控件，从而用户不能够修改该控件。可以使用脚本重新启用该表单控件。但是，除非重新启用该表单控件，否则名/值对将不会发送给服务器。该属性的值必须是 disabled。

Netscape 6 和 IE 5 浏览器中最早实现了 readonly 属性和 disabled 属性，但是较老的浏

览器会忽略它们，因此可以在所有的文档中添加这两个属性。但需要注意的是，因为较老的浏览器忽略这两个属性，所以使用较老浏览器的用户仍然能够与具有 readonly 属性或 disabled 属性的表单控件交互。

如果希望阻止访问者改变表单的某个部分，或者是因为它无法改变(例如一些条款和条件)，或者是因为希望提醒用户他们已经说过的内容，此时 readonly 控件就非常有用。在用户协议或者 e-mail 表单(e-mail 表单允许用户将一个 Web 页面邮寄给朋友)的主体中经常会使用 readonly 类型的控件。

如果希望阻止用户与某个控件交互直到他们执行某些操作，此时 disabled 属性就特别有用。例如，可以使用一个脚本禁用提交按钮，直到所有的表单字段都包含值。

表 5-9 列举了可以具有 readonly 属性和 disabled 属性的表单控件。

表 5-9

元 素	readonly	disabled
<textarea>	是	是
<input type="text" />	是	是
<input type="checkbox" />	否	是
<input type="radio" />	否	是
<input type="submit" />	否	是
<input type="reset" />	否	是
<input type="button" />	否	是
<select>	否	是
<option>	否	是
<button>	否	是

表 5-10 给出了 readonly 属性和 disabled 属性之间的主要区别。

表 5-10

属 性	readonly	disabled
可以修改	可以被脚本而不是用户修改	当禁用时不能修改
将发送到服务器	是	当禁用时不能发送
将接收焦点	是	否
包含在焦点移动顺序中	是	否

5.8 向服务器发送表单数据

本章前面介绍了提交按钮，用户可以使用它向服务器发送表单数据，但是本书还没有介绍 HTTP get 方法和 HTTP post 方法之间的区别。您或许记得，可以通过添加 method 属

性到<form>元素来指定使用哪一个方法——类似于本章中的所有示例执行的操作。

method 属性可以附带的值包括 get 和 post，只能选择其中之一，它们对应于用于发送表单数据的 HTTP 方法。如果<form>元素不附带 method 属性，则默认情况下将使用 get 方法。如果正在使用一个文件上传表单控件，则必须选择 post 方法(并且必须将 enctype 属性的值设置为 multipart/form-data)。

5.8.1 HTTP get

当使用 HTTP get 方法向服务器发送表单数据时，表单数据将附加在<form>元素的 action 属性指定的 URL 中。

表单数据与 URL 之间以问号隔开。在问号之后是每一个表单控件的名/值对，每一个名/值对之间由一个&符号隔开。

例如，考虑下面的登录表单(在介绍密码表单控件时引入了该表单)：

```
<form action="http://www.example.com/login.aspx" method="get">
  Username:
  <input type="text" name="txtUsername" value="" size="20"
maxlength="20"><br />
  Password:
  <input type="password" name="pwdPassword" value="" size="20"
maxlength="20">
  <input type="submit" />
</form>
```

当单击提交按钮时，用户名和密码附加在 http://www.example.com/login.aspx 中，形成查询字符串：

```
http://www.example.com/login.aspx?txtUsername=Bob&pwdPassword=LetMeIn
```

注意，当浏览器请求一个具有空格或不安全字符(例如/、\、=、&和+，它们在 URL 中具有特殊的含义)的 URL 时，它们将被一个十六进制编码所替代。由浏览器自动完成这种称为 URL 编码的操作。当数据到达服务器时，服务器将自动解码特殊字符。

在 URL 中传递表单数据的最大优点之一是它可以被添加书签。如果查看主要的搜索引擎(例如 Google)的搜索执行方式，则可以发现它们使用了 get 方法，以便页面可以被添加书签。

但是，get 方法也具有一些缺点。事实上，当发送敏感数据时，例如密码或信用卡卡号，则不应该使用 get 方法，因为作为 URL 一部分的敏感数据能够被所有人看到(并且可以被添加书签)。

在以下情况下不应该使用 HTTP get 方法：

- 更新数据源，例如数据库或者电子数据表(因为某些人可以建立能够更改数据源的 URL)。
- 处理敏感信息，例如密码或信用卡卡号(因为作为 URL 一部分的敏感表单数据将可见)。
- 具有大量数据(因为较老的浏览器不允许 URL 超出 1024 个字符——但是主要浏览器的最新版本没有这种限制)。

- 表单包含一个文件上传控件(因为上传的文件不能在 URL 中传递)。
- 用户可能输入非 ASCII 字符，例如 Hebrew 字符或 Cyrillic 字符。

在这些情况下，应当使用 HTTP post 方法。

5.8.2 HTTP post

当使用 HTTP post 方法向服务器发送表单数据时，表单数据将在 HTTP 头中被透明地发送给服务器。虽然无法看到这些头，但它们以纯文本的方式发送，并且无法保证安全性(除非在安全套接字层(SSL)中发送数据)。

如果使用 post 方法发送前面介绍的登录表单，则它在 HTTP 头中将类似如下：

```
User-agent: MSIE 5.5
Content-Type: application/x-www-form-urlencoded
Content-length: 35
...other headers go here...
txtUserName=Bob&pwdPassword=LetMeIn
```

注意，最后一行是表单数据，它的格式与 get 方法中问号之后的数据相同——也是 URL——如果它包含空格或者任何其他 URL 中预留使用的字符，则会被编码。

对于包含查询字符串的页面来说，也可以使用 post 方法向其发送表单数据。例如，某个页面可能需要处理用户是否订阅时事通讯，因此您可能选择在查询字符串中指示用户是否订阅时事通讯。同时，您可能希望使用 post 方法发送表单中用户的实际联系细节，因为正在更新数据源。此时，可以使用下面的<form>元素：

```
<form action="http://www.example.com/newsletter.asp?action=subscribe"
     method="post">
```

使用 HTTP post 方法的唯一问题是，用户在表单中输入的信息无法像其包含在 URL 中时一样被添加书签。因此，不能使用它检索页面(使用指定的表单数据生成该页面)，而当为由大多数搜索引擎生成的页面添加书签时可以执行该操作，但这样对安全有好处。

试一试 回顾注册表单

现在回顾一下本章前面的“试一试”中介绍的注册表单，这一次在其中添加一些其他字段，以使其具有更好的可用性。

(1) 打开前面建立的 registration.html 文件，将其保存为 registration2.html 文件，以便具有一个不同的副本并处理该副本。

(2) 为所有的表单控件创建一个<label>元素，这包括将该控件的指令放入一个<label>元素中。该元素应该附带 for 属性，for 属性的值是对应表单控件中 id 属性的值，如下所示：

```
    <tr>
        <td><label for="userName">User name:</label></td>
        <td><input type="text" name="txtUserName" size="20" id="username"
/></td>
    </tr>
```

(3) 必须分别赋予两个单选按钮不同的标签：

```
<tr>
    <td>Gender:</td>
    <td><input type="radio" name="radSex" value="male" id="male" />
    <label for ="male">Male</label></td>
</tr>
<tr>
    <td></td>
    <td><input type="radio" name="radSex" value="female" id="female" />
    <label for="female">Female</label></td>
</tr>
```

注意：

如果您记得上一章中关于屏幕阅读器的表线性化的讨论，则对于大多数用户来说这种方式应该工作良好。但是，如果右边的另外一列具有一些不相关信息(例如广告)，这种方式将使读者产生混淆，因此用于表单控件的表当仅保存控件和它们的标签。

(4) 在用户名和密码之后添加 4 个新文本框。第一个文本框用于确认密码，然后将有一个空行。后面跟着两个文本输入框：一个用于输入用户的名，另一个用于输入用户的姓。然后是另外一个空行，接下来的文本输入框用于输入用户的 e-mail 地址：

```
<tr>
  <td><label for="confPwd">Confirm Password:</label></td>
  <td><input type="password" name="pwdPasswordConf" size="20"
     id="confPassword" /></td>
</tr>
<tr><td> </td><td> </td></tr>
<tr>
  <td><label for="firstName">First name:</label></td>
  <td><input type="text" name="txtFirstName" size="20" id="firstName"
/></td>
</tr>
<tr>
  <td><label for="lastName">Last name:</label></td>
  <td><input type="text" name="txtLastName" size="20" id="lastName" /></td>
</tr>
<tr><td> </td><td> </td></tr>
<tr>
  <td><label for="email">Email address:</label></td>
  <td><input type="text" name="txtEmail" size="20" id="email" /></td>
</tr>
```

(5) 使用<fieldset>元素将表单拆分为两个部分。第一部分将指示关于用户的信息(包含用户名、密码、姓名、e-mail 地址和性别等细节)。第二部分用作关于公司的信息(用户如何找到该站点以及用户是否希望添加到邮件列表中)。

两个<fieldset>元素都将附带访问键。下面是作为表单第二部分的<fieldset>元素：

```
<fieldset>
    <legend accesskey="u">About <u>U</u>s (ALT + U)</legend>
  <table>
    <tr>
      <td><label for="referrer">How did you hear about us?</label>:</td>
      <td>
        <select name="selReferrer" id="referrer">
          <option selected="selected" value="">Select answer</option>
          <option value="website">Another website</option>
          <option value="printAd">Magazine ad</option>
          <option value="friend">From a friend</option>
          <option value="other">Other</option>
        </select>
      </td>
    </tr>
    <tr><td> </td><td> </td></tr>
    <tr>
      <td><label for="mailList">Please select this box if you wish<br /> to be
          added to our mailing list
        <br /><small>We will not pass on your details to any third
        party.</small></label></td>
      <td><input type="checkbox" name="chkMailingList" id="mailList" /></td>
    </tr>
  </table>
</fieldset>
```

这个扩展后的注册表单现在具有更好的可用性。如果再次保存该文件并在浏览器中打开它，结果将类似于图 5-24 所示。

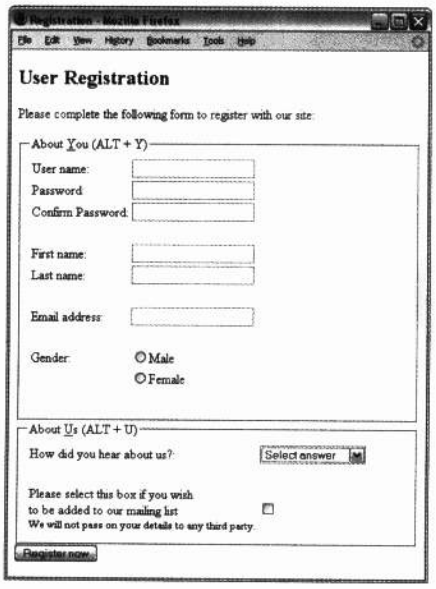


图 5-24

工作原理

您应该已经对这些代码很熟悉，但有必要说明如下一些关键点。

- 使用<fieldset>元素将表单划分为两个部分。这种额外的结构可使表单更易于使用，因为用户能够清楚地知道他们正处于哪个部分中。
- 如果创建的是长表单，accesskey 属性将非常有用，因为它提供了键盘快捷键，从而用户能够立即转向相应的部分。事实上，相对于用户很少使用的表单来说，在创建用户频繁访问的站点时更可能使用 accesskey 属性。只有在用户已经熟悉表单并希望在不同的部分之间快速切换时，才可能会使用快捷键。
- 在第 10 章中将会看到，如果创建特别长的表单，最好将它划分为多个页面。
- 对于使用屏幕阅读器的用户来说，<label>元素非常有帮助。它确保用户知道需要在哪些表单控件中输入什么内容。
- 当使用<fieldset>元素拆分页面时，确保元素正确嵌套。不能将<fieldset>元素跨越表的多行。

5.9 本章小结

本章介绍了如何创建在线表单，它们是很多站点的非常重要的部分。大多数情况下，当希望或需要从访问站点的访问者处收集信息时，就需要使用表单。在本章中已经介绍了多个不同的表单示例。

从简单的搜索框和登录页面到复杂的在线订购表单和注册过程，表单是 Web 设计的重要组成部分。

本章介绍了表单如何驻留在<form>元素中，以及在一个表单中可能具有多个表单控件。您已经看到如何使用<input>元素创建多种类型的表单控件，包括单行文本输入控件、复选框、单选按钮、文件上传框、按钮和隐藏表单控件。本章还介绍了用于创建多行文本输入框的<textarea>元素以及用于创建选项框的<select>元素和<option>元素。

一旦创建了表单和其中的表单控件，需要确保表单中的每一个元素都被正确地赋予标签，以便用户知道他们应当在什么位置填写什么信息或者选择什么选项。也可以使用<fieldset>元素和<label>元素来组织较大的表单，并利用 tabindex 属性和 accesskey 属性辅助导航。

最后，本章介绍了什么时候应当使用 HTTP get 方法或 post 方法向服务器发送表单数据。

本书接下来将介绍关于框架方面的内容，它是本书中核心 XHTML 章节的最后一章。在第 12 章中将介绍更多关于表单设计方面的内容，其中包括一些设计问题，这些问题将使得表单更易于理解。

5.10 练习

所有练习的答案都在附录 A 中给出。

1. 创建一个类似于图 5-25 所示的 e-mail 反馈表单。

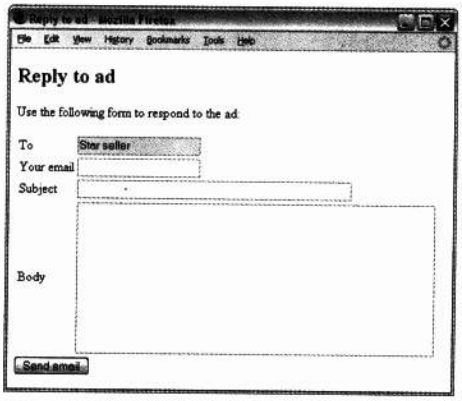


图 5-25

注意，第一个文本框是一个 readonly 类型的文本框，从而用户不能改变 e-mail 接收人的姓名。

2. 创建一个类似于图 5-26 所示的投票或排名表单。

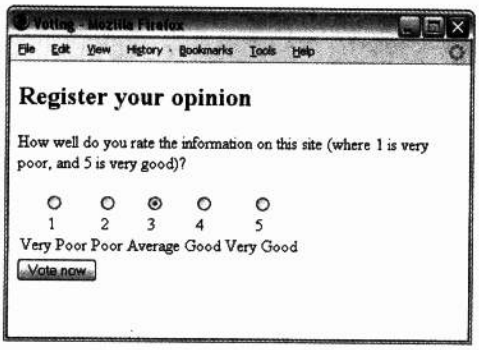


图 5-26

```
<head>
```

注意，下面的<style>元素添加到文档的<head>部分中，以确保表的每一列具有相同的固定宽度，并且文本居中对齐(第 7 章中将更详细地介绍这方面的知识)。

```
  <title>Voting</title>
  <style type="text/css">td {width:100; text-align:center;}</style>
</head>
```

第6章

框　架

框架能够将浏览器窗口划分为多个独立的部分或窗格，每一个窗格包含一个独立的XHTML页面。框架提供的主要优点之一是，用户利用它能够加载或者重新加载单个窗格，而不需要重新加载浏览器窗口的全部内容。浏览器窗口中的框架集合称为框架集。

窗口被划分为框架，类似于表被组织为行和列(尽管它们通常是相对基本的结构)。最简单的框架集可能只是将屏幕划分为两行，而复杂的框架集可能具有多行和多列。

本章将介绍以下内容：

- 创建具有多个框架的框架集文档的方式
- 创建内联框架(或者 ifra...e)的方式，内联框架是位于另一个页面中的单个窗口
- 当用户浏览器不能使用框架时的处理方式

注意：

需要预先提醒的是，实际上我很少主张使用框架，尽管这可能只是一种偏好，在本章的第二部分中给出了其中的原因——第二部分中的一个简单示例将帮助您理解框架的本质。

6.1 框架集简介

为了帮助您理解框架，图 6-1 中给出了浏览器中的一个框架集文档。这个框架集将页面划分为 3 个部分，页面的每一个部分是一个独立的 XHTML 文档。

在第 1 章中曾提到，当编写一个框架集文档时，需要使用不同的 DOCTYPE 声明。这是因为框架集文档使用一些元素的方式与其他的 XHTML 文档不同。

为了创建框架集文档，首先需要使用<frameset>元素，它用于替代<body>元素。框架集定义了划分页面的行和列。然后，通过一个<frame>元素表示每一个框架。

也需要掌握<noframes>元素。当浏览器不支持框架时，该元素向用户提供一条消息。

为了更好地理解框架的工作原理，下面给出了图 6-1 中所示框架集的代码(ch06_eg01.html)：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head>
  <title>Frames example</title>
```

```
</head>
<frameset rows="150, *, 100">
   <frame src="top_frame.html" />
   <frame src="main_frame.html" />
   <frame src="bottom_frame.html" />
   <noframes><body>
     This site uses a technology called frames. Unfortunately, your
     browser does not support this technology. Please Please upgrade
     your browser and visit us again!
   </body></noframes>
</frameset>
</html>
```

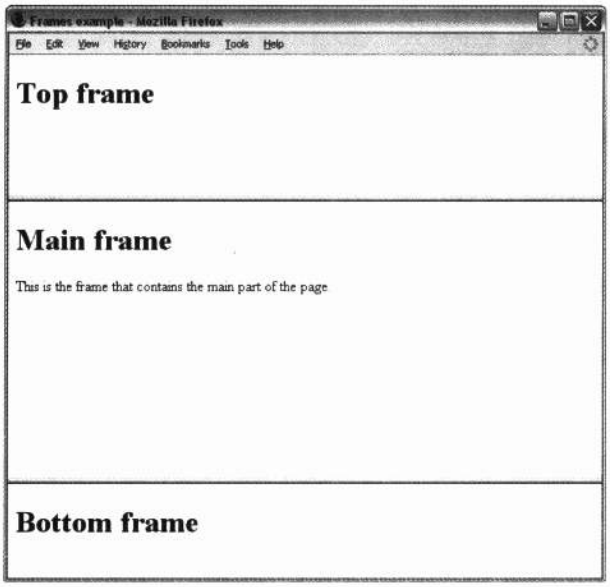


图 6-1

实际上，新的 DOCTYPE 声明所起的作用仅是允许用户使用这些框架相关的元素。

本章前面已经提到，文档中不存在<body>元素，因为该元素被<frameset>元素所替代；另外，在结束标签</head>和起始标签<frameset>之间不能有其他标记，除非是希望包含的注释。

在本章后面您将看到，<frameset>元素必须附带两个属性：rows 和 cols，它们指定了组成框架集的行和列的数量。在上面的示例中仅有 3 行，第一行有 150 像素高，第三行只有 100 像素高，而第二行占用了页面的剩余部分(星号用于表明该部分将占用页面剩余的部分)。

```
<frameset rows="150, *, 100">
```

在<frameset>元素之内是空的<frames />元素。<frames />元素指示将加载到该框架中的文档的 URL；采用 src 属性指定 URL(类似于在<img />元素中指定图像文件的方式)。另外也存在一个<noframes>元素，如果用户的浏览器不支持框架，就会显示该元素的内容。

在这个示例中，在一个浏览器窗口中将显示如下 3 个独立的文档：

- top_frame.html
- main_frame.html
- bottom_frame.html

根据图 6-1 所示，能够很容易地发现这些页面分别对应于窗口的哪一部分。

为了展示框架工作原理的另外一种思想，请查看图 6-2，该图中的页面同时使用了水平框架和垂直框架(这表明框架的作用类似于有时使用简单的表划分页面)。

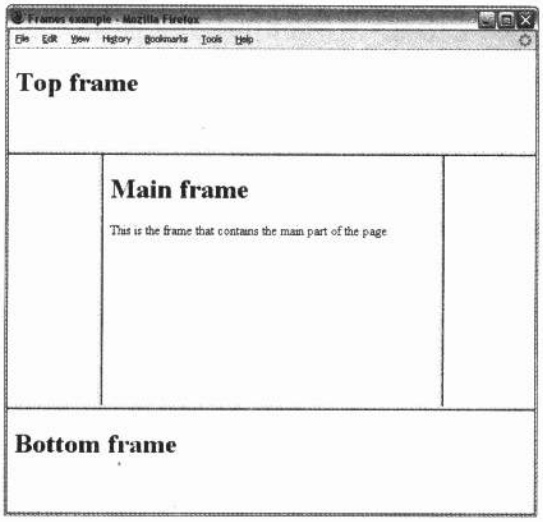


图 6-2

注意：

尽管 Netscape 浏览器从版本 2 开始支持框架，IE 浏览器在版本 3 中开始引入框架，但是直到 4.0 版本的 HTML 中框架才被 HTML 所接受。

现在您已经了解框架集文档的外观，在更详细地查看语法之前，首先了解什么时候应当使用框架。

6.2 使用框架的时机

实际上，当前很少使用框架。我个人认为，建议在页面中使用框架的情形非常少。我认为需要使用框架的情形包括：

- 当希望在单个页面中显示大量的内容并且不能将文档拆分为多个独立的页面时，则可以使用框架创建一个导航栏，该导航栏链接到长文档的各个子部分。

- 当页面的一部分具有很多数据并且在页面的其他部分改变时不希望重新加载该部分时，可以考虑使用框架。例如，对于一个摄影站点，可以在一个框架中包含很多缩略图，而在另一个框架中包含主图片。如果访问者希望查看新的主图片，则可以仅加载该主图片，而不需要每次都重新加载缩略图。

根据第一个示例可以发现，对于布局中的每一个框架，都需要一个文件作为它的内容(实际上每一个框架是自己的 Web 页面)，因此站点中文件的数量可能会增长得很快。因而需要特别注意文件结构，以便不会迷失在文件的海洋中。

对于框架，需要注意的其他一些缺点包括：

- 搜索引擎通常会链接到独立的框架中的内容，而不是用户看到的框架集(或框架组)中的内容(并且如果访问者登录到某个独立的框架中，将不得不在每一个框架中使用 JavaScript 重新加载整个框架集)。
- 某些浏览器无法很好地打印整个框架集(可能一次仅打印其中一个框架)。
- 浏览器的“后退”按钮可能无法像用户期望的那样工作。
- 某些较小的设备无法处理框架，通常是因为它们的屏幕不够大，以至于无法被划分。
- 可能无法获得漂亮的布局，因为比设计人员具有更低分辨率显示器的用户可能会仅看到框架的一部分，而比设计人员具有更高分辨率显示器的用户看到的框架周围可能具有大量空白。
- 如果具有一个导航框架(该框架将不同的页面加载到一个“主框架”中)，则很难创建导航栏来告诉用户他们正处于哪个页面中(因为其他框架加载新页面时并不通知导航栏)。

还需要提醒的是，当 Web 开发人员希望创建一个页面并需要仅刷新该页面的某个部分(而不是整个页面)时，通常会使用称为 AJAX(Asynchronous JavaScript and XML，异步 JavaScript 和 XML)的技术。

根据我对框架的观点，如果您认为使用框架的优点多于缺点，则应该使用它们。因此，下面将更详细地介绍一些关于框架的语法。

6.3 <frameset>元素

在框架集文档中，<frameset>元素取代了<body>元素。<frameset>元素的属性指定了浏览器窗口将如何被划分为多个行和列，这些属性包括：

- cols 指定框架集中列的数量。
- rows 指定框架集中行的数量。

<frameset>元素包含用于文档中每一个框架(或者<frameset>元素构造的网格的每一个单元格)的<frame>元素，以及一个<noframes>元素，该元素指明当用户浏览器不加载框架时将显示的内容。

除了 rows 属性和 cols 属性，框架集元素也可以附带如下属性：

```
class id onload onunload rows style title
```

大多数浏览器也支持下面一些经常使用的属性(其中一些属性包含在此处是因为它们的使用很广泛)。但是，它们不是 W3C 规范的组成部分。

```
onblur onfocus border bordercolor frameborder framespacing
```

6.3.1 cols 属性

cols 属性指定了框架集中包含的列的数量以及每一列的大小。开发人员必须为框架集中的每一列指定一个宽度值，并且提供的值的数量(不同的值之间使用逗号隔开)表明了文档中列的数量。例如，文档中具有 3 列：第一列占用浏览器窗口 20%的宽度，第二列占用 60%的宽度，第三列占用剩余的 20%宽度，则该属性如下所示：

```
cols="20%, 60%, 20%"
```

因为具有 3 个值，所以浏览器窗口知道应当有 3 列。

可以采用如下 4 种方式之一指定每一列的宽度：

- 以像素为单位的绝对值
- 浏览器窗口(如果某个框架集位于另外一个框架集中，也称为框架的嵌套，则是父框架)的百分比
- 使用通配符符号
- 浏览器窗口(或者父框架)的相对宽度

指定列的宽度时，可以混合使用这些方式，但是注意它们的优先级(在介绍 4 种方法后将对此进行讨论)。

如果没有指定 cols 属性，则默认值将是 100%。因此如果没有指定 cols 属性，则将存在一列，并且该列占用浏览器 100%的宽度。

1. 以像素为单位的绝对值

为了指定以像素为单位的列宽度，仅需要使用一个数值(不需要在数值之后使用 px 或者其他任何字符)。例如，下面的代码具有 3 列：第一列占用 100 像素，第二列占用 500 像素，第三列占用页面剩余的空间(因为使用了通配符*)。

```
cols="100, 500, *"
```

如果仅使用绝对值，并且窗口的宽度小于或大于指定的值，则浏览器将按照相对于浏览器窗口宽度的比例来调整每一列的宽度。因此，如果希望 3 列都具有 100 像素的宽度，可以采用如下方式指定：

```
cols="100, 100, 100"
```

但是，如果浏览器窗口是 600 像素宽，则最终每一列将为 200 像素宽。因此，如果确实希望指定不会增长的固定绝对宽度，则可以在第三列之后使用一个通配符，并且为第四列指定空内容或者不为其包含一个<frame />元素：

```
cols="100, 100, 100, *"
```

有趣的是，如果具有200像素宽的4列，并且浏览器窗口的宽度只有600像素，则每一列将被按比例压缩到150像素宽；窗口不会使用滚动栏以支持800像素宽的页面。

2. 浏览器窗口或者父框架的百分比

为了将列的宽度指定为窗口的百分比(或者如果使用嵌套框架，则是父框架的百分比，本章后面将介绍嵌套框架)，可以使用一个数值后跟一个百分比符号。例如，下面的属性值指定了两列，其中一列占用40%的浏览器窗口，另一列占用60%的浏览器窗口：

```
cols="40%, 60%"
```

如果以百分比指定宽度，但它们的总和小于或大于100%，则浏览器将按比例调整宽度。

3. 通配符

当使用绝对值或百分比指定宽度时，星号或通配符指示“窗口的剩余宽度”。在下面的示例中，第一列具有400像素宽，第二个框架占用剩余的浏览器窗口宽度：

```
cols="400, *"
```

如果两行或两列被赋予通配符，则剩余的宽度将被它们均分。

4. 列的相对宽度

作为百分比方法的一种替代方式，可以使用浏览器窗口的相对宽度，利用一个示例可以很好地演示该思想。下面的窗口被划分为6份：第一列占用窗口的1/2，第二列占用1/3，第三列占用1/6：

```
cols="3*, 2*, 1*"
```

其中，通过累加相对宽度的值将窗口划分为6份。

5. 值的优先级和调整窗口大小

绝对值宽度的优先级总是比相对宽度高。考虑下面具有3列的示例：

```
cols="250, *, 250"
```

如果窗口仅有510像素宽，则中间的框架将仅有10个像素宽。该示例清楚地说明为什么在设计框架的时候需要非常小心，以便用户能够看到您计划让他们看到的内容。

此外，如果用户调整窗口大小，使得窗口小于500像素宽，则浏览器将会尽可能地显示使用绝对宽度定义的列，而忽略使用相对宽度定义的列。

当用户调整窗口大小时，相对宽度和百分比将被重新计算，而绝对宽度将保持不变。

如果指定的列数量多于所需框架数量，则最右边的列将是空白；如果指定太多的<frame />元素，则多余的元素将被忽略。

6.3.2 rows 属性

rows 属性的工作原理类似于 cols 属性，并且可以附带相同的值，但它用于指定框架集中行的数量。例如，下面的 rows 属性将指定 3 行：最顶端的行具有 100 像素高，第二行具有屏幕的 80%高，最下面的一行占用屏幕剩余的空间(如果还剩下空间的话)：

```
rows="100, 80%, *"
```

rows 属性的默认值是 100%，因此如果没有指定 rows 属性，则一行将占用 100%的浏览器高度。

6.3.3 针对 <frameset>元素的浏览器专用扩展

大多数常用的浏览器(例如 IE、Firefox 和 Safari)支持针对<frameset>元素的一些非常重要的扩展，这里有必要介绍它们。在第一个示例中您可能已经注意到，默认情况下框架将创建边框，并且您很可能希望控制该边框的外观。虽然现在可以使用 CSS 来控制这些特性，但是如果查看以前的代码，您很可能会遇到一些能够控制外观的属性。

1. border 属性

border 属性指定了每个框架的边框的宽度，单位为像素。Netscape 3 和 IE 4 浏览器中开始引入该属性。

```
border="10"
```

图 6-3 给出了具有 10 个像素边框宽度的第一个示例的外观。如果将该图与图 6-2 相比较，将能够看到每一个框架之间具有更高的灰线(ch06_eg02.html)：

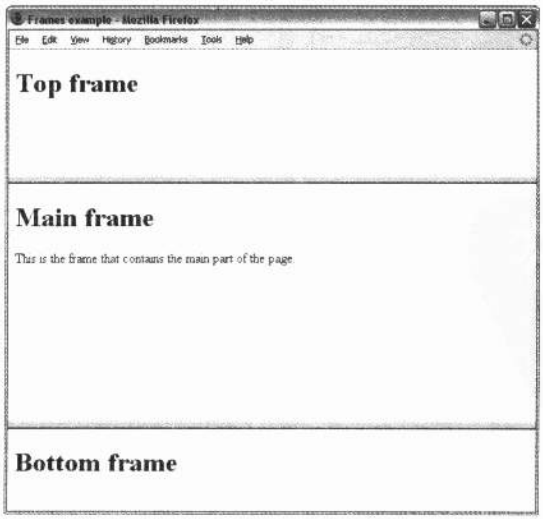


图 6-3

如果不需要边框，可以将该属性的值设置为 0。

当第一次创建框架集文档时，最好将这个属性的值设置为 1，即使是不需要边框，因为在构建站点时，这样操作将使得框架清晰；可以通过修改<frameset>元素中的该属性来方便地删除边框。

2. frameborder 属性

frameborder 属性指定是否应该在框架之间显示三维边框。下面的代码表明框架之间不应该有边框(类似于 border 属性值被设置为 0)：

```
frameborder="0"
```

表 6-1 给出了 frameborder 属性的可能值。

表 6-1

值	目 的
1 或 yes	表明应当显示边框，它是默认值(yes 值不是 HTML 4 的一部分，但是仍然被一些常用的浏览器所支持)
0 或 no	表明不应当显示边框(no 值不是 HTML 4 的一部分，但是仍然被一些常用的浏览器所支持)

图 6-4 给出了没有边框的框架的外观——无法看到一个框架的起始位置和另一个框架的结束位置——除非框架中的页面具有不同的图像或者背景色(ch06_eg04.html)。

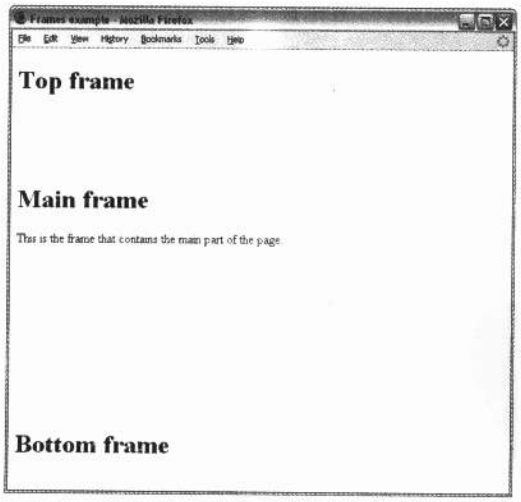


图 6-4

3. framespacing 属性

framespacing 属性指定了框架集中框架之间的空间量。该值的单位是像素，如果没有指定的话，默认值是 2。

```
framespacing="25"
```

图 6-5 给出了本章第一个示例(如图 6-1 所示)使用 framespacing 属性之后的外观，其中框架之间具有 25 个像素的空间(ch06_eg05.html)。

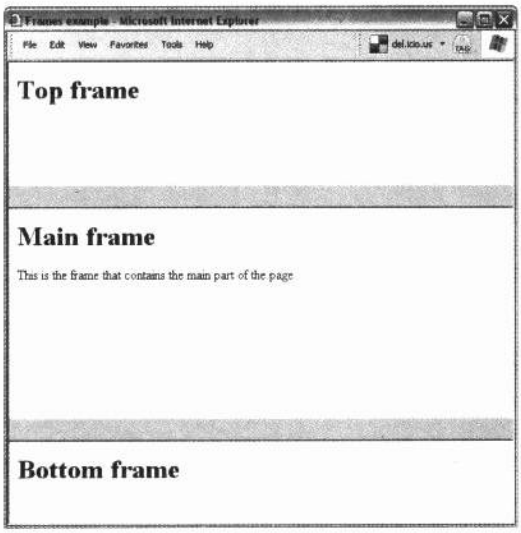


图 6-5

如果需要确保在较老的浏览器中显示的框架不具有边框，则应当结合使用 border、frameborder 和 framespacing 属性，并且将每一个属性值都设置为 0(这将确保 Netscape 3 和 IE 3 之后发布的浏览器中显示的框架不具有边框)。

```
border="0" frameborder="0" framespacing="0"
```

附录 I 中还介绍了一些其他的浏览器专用属性

6.4 <frame>元素

<frame>元素用于指示框架集中每一个框架的内容。<frame>元素始终是空元素，因此不应该包含任何内容，但是每一个<frame>元素都必须附带一个 src 属性，用于指示哪个页面应该表示该框架。

<frame>元素可以附带任何通用属性以及下面的属性：

```
frameborder marginwidth marginheight noresize scrolling longdesc src name
```

注意，不存在与<frame>元素相关的 CSS 样式。

6.4.1 src 属性

src 属性指明应该用于框架中的文件。

```
src="main_page.html"
```

src 属性的值是一个普通的 XHTML 页面，因此对于每一个<frame />元素，必须存在一个相应的页面。

虽然这个属性的值通常是服务器上的一个文件，但它的值也可以是其他任何 URL，因此可以使用 src 属性指定任何其他站点。

您可能已经发现，Internet 上的某些搜索引擎(例如 Google 中的图像搜索)将在页面的顶部创建一个框架集，用于保存搜索到的站点，而在页面底部将是用户所请求的页面。

如果使用这样的一个框架，则最好提供一个能够关闭顶部框架的链接，以便允许访问者仅查看主框架中的内容(类似于 Google)。

6.4.2 name 属性

name 属性用于为框架指定名称；它用于指示文档应当加载到哪一个框架中。当希望在第一个框架中创建一些链接，而将页面加载到第二个框架中时，该属性非常有用。此时第二个框架需要一个名称来将其自己标识为链接的目标。在本章后面您将看到关于在框架之间建立链接方面的内容。

```
name="main_frame"
```

需要注意的是，这里 name 属性没有被 id 属性所替换(当引入 XHTML 作为 HTML 的继任者时，某些其他 HTML 元素中的 name 属性被 id 属性所替换)。

6.4.3 frameborder 属性

frameborder 属性指定是否显示框架的边框；它重写<frameset>元素中 frameborder 属性的值(前提是给定该属性)，并且两者可能的值是相同的。表 6-2 给出了 frameborder 属性的可能值。

表 6-2

值	目 的
1 或 yes	指明显示边框，是默认值(yes 值不是 HTML 4 的一部分，但是仍被 IE 和 Netscape 浏览器所支持)
0 或 no	指明不显示边框(no 值不是 HTML 4 的一部分，但是仍被 IE 和 Netscape 浏览器所支持)

6.4.4 marginwidth 属性和 marginheight 属性

页边空白是框架的三维边框与其内容之间的空间。

marginwidth 属性用于指定框架边框的左边和右边与框架内容之间的空间的宽度。该属性值的单位为像素。

marginheight 属性指定框架边框的顶部和底部与框架内容之间的空间的高度。该属性值的单位为像素。

```
marginheight="10" marginwidth="10"
```

6.4.5 noresize 属性

通过单击和拖动框架的边框，通常能够调整框架的大小。如果用户无法阅读框架中的内容，则该功能非常有用，但是它会使设计人员更难于控制页面的布局。

noresize 属性能够阻止用户调整框架的大小。在 HTML 4 中，该属性是一个不具有值的最小化属性，但是现在它必须附带值 noresize：

```
noresize="noresize"
```

注意：

对于显示器分辨率低于您的显示器分辨率的用户来说，可能无法完全看到您在高分辨率监视器中看到的框架内容。如果使用 noresize 属性，不能看到框架完整内容的用户将无法调整框架的大小以看到缺少的内容。

6.4.6 scrolling 属性

如果框架的内容太多，以至于分配给它的空间不够，则浏览器将很可能为用户提供一个滚动栏，以便他们能够阅读该框架中剩余的内容。

可以使用 scrolling 属性来控制出现在框架上的滚动栏的外观：

```
scrolling="yes"
```

该属性可以采用表 6-3 中的 3 个可能值之一。

表 6-3

值	目　的
yes	指示框架必须总是包含一组滚动栏，而不论是否需要它们，尽管 IE 浏览器仅显示一个垂直的滚动栏，Firefox 浏览器中的显示方式则如同该属性被设置为 auto
no	指示框架必须不包含一组滚动栏，即使框架无法容纳这些内容
auto	指示当框架无法容纳内容时浏览器应该包含滚动栏，否则不显示它们

6.4.7 longdesc 属性

longdesc 属性用于提供一个到其他页面的链接，该页面包含关于框架内容的较长描述。这个属性的值应当是指向描述信息所在位置的 URL。

```
longdesc="framedescription.html"
```

注意：

W3C 指示这个 URL 的值不能是相同页面内的一个锚点。

6.5 <noframes>元素

如果用户的浏览器不支持框架(现在这种情形非常少见)，则<noframes>元素中的内容将显示给用户。

在 XHTML 中，必须在<noframes>元素中放置一个<body>元素，因为<frameset>元素替代了<body>元素。但是，如果浏览器无法理解<frameset>元素，它应该忽略这些元素和<noframes>元素，并理解<noframes>元素中包含的<body>元素的内容。

在表达该元素中的内容时应当非常仔细。不应当按照如下所示编写该元素：

```
<noframes><body>This site requires frames.</body></noframes>
```

该内容对普通用户来说毫无意义——毕竟他们的浏览器很可能非常古老，并且他们很可能没有学习过 HTML 或 XHTML，不知道什么是框架。相反，应当提供类似于如下所示的详细描述内容：

```
    <noframes><body>This site makes uses of a technology called frames.
    Unfortunately the browser you are using does not support this technology. We
    recommend that you update your browser. We apologize for any inconvenience
this
    causes.</body></noframes>
```

如果希望更漂亮地展示消息，可以在<noframes>元素内使用其他 XHTML 标记。

注意：

虽然您以为浏览器不支持框架的用户提供一个无框架版本的站点，但这需要大量的工作。因此，可选的方法是提供到一些页面(这些页面是框架的组成部分)的链接，以便用户在浏览器不支持框架的情况下仍然能够看到站点的内容。

6.6 创建框架之间的链接

框架最流行的用法之一是在一个框架中放置导航栏，然后将具有内容的页面加载到另一个单独的框架中。在如下 3 种情形下，这种用法非常有用：

- 当导航栏尺寸非常大时(例如画廊摄影图片的缩略图)。通过使用框架，用户不需要在查看新页面时重新加载导航栏。
- 当主文档非常长时，导航栏提供到主文档不同位置处的快捷方式(作用类似于始终可见的内容表)。
- 当不希望重新加载整个页面时。

每一个<frame>元素可以附带一个 name 属性，以便赋予该框架一个名称。这个名称将用于链接中，以指示新页面应该加载到哪一个框架中。考虑下面的简单示例：

```
<frameset cols="200, *"
  <frame src="frames/linksNav.html" />
```

```
    <frame src="frames/linksMain.html" name="main_page" />
</frameset>
```

这个示例具有两列，并且只有一行。第一列具有 200 像素宽，它将包含导航栏。第二列或框架将包含页面的主要部分。左边导航条中的链接将加载页面到右边的主页面中。

linksNav.html 文件中的链接如下所示：

```
<a href="http://www.wrox.com" target="main_page">Wrox Press</a><br /><br />
<a href="http://www.google.com" target="main_page">Google</a><br /><br />
<a href="http://www.microsoft.com" target="main_page">Microsoft</a><br
/><br />
<a href="http://news.bbc.co.uk/" target="main_page">BBC News</a><br /><br />
```

注意：

用于创建导航栏的技术很难指明用户正处于哪一个页面中，因为每次加载一个新页面时都需要使用 JavaScript 或者其他方法将信息从主页面传递到导航栏中。

图 6-6 给出了这个示例在浏览器中的外观。

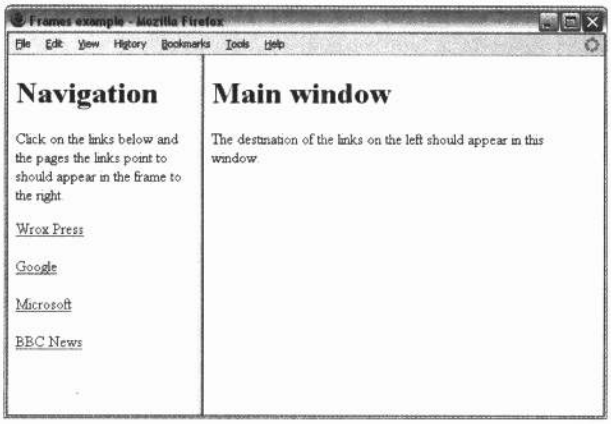


图 6-6

对于每一个需要加载新页面的框架来说，都需要使用 name 属性——这是将新内容加载到框架中的关键。

target 属性也可以采用表 6-4 中列举的属性值。

表 6-4

值	目　的
_self	将页面加载到当前框架中
_blank	将页面加载到新的浏览器窗口中——打开一个新的窗口(类似于使用不存在的目标)
_parent	将页面加载到父窗口中。对于单个框架集，父窗口是指主浏览器窗口(此时页面将替换所有的框架)；对于嵌套框架，页面将替换内层框架集所在的框架
_top	将页面加载到浏览器窗口中，替换任何当前框架

如果创建到外部页面的链接，通常需要为 target 属性使用_top 值，以便外部站点替换当前的整个站点；毕竟，用户可能不希望在您的站点的框架中查看外部的页面。其他的选项将在新的窗口中打开外部站点。

说明：

对于初学者来说，当创建使用框架的 Web 站点时如果出现问题，原因通常是忘记在<frame>元素中添加 name 属性以及在<a>元素中添加 target 属性。无论是忘记添加哪一个属性，浏览器都将只加载该框架中的链接。

利用<base>元素设置默认目标框架

可以使用<base>元素在页面(该页面包含将在另一个框架中打开的链接)中设置一个默认目标框架。<base>元素必须附带称为 target 的属性，它的值是希望将内容加载到其中的框架的名称。因此，可以将下面的代码添加到 linksNav.html 中，以指定一个默认的框架目标：

```
<head>
  <base target="main_page" />
</head>
```

6.7 框架集的嵌套

单个框架集提供一种类似于表的由行和列组成的固定网格结构。如果希望创建更复杂的设计，可以选择使用嵌套框架集。

可以在<frame>元素所在的位置使用新的<frameset>元素作为替换，以此创建嵌套框架集。查看如下示例(ch06_eg07.html)：

```
<frameset rows="*, 300, *">
  <frame src="frames/top_frame.html" />
  <frameset cols="*, 400, *">
    <frame src="frames/blank.html" />
    <frame src="frames/main_frame.html" />
    <frame src="frames/blank.html" />
  </frameset>
  <frame src="frames/bottom_frame.html" />
</frameset>
```

这个示例创建了一个具有 3 行的框架集。中间一行是具有 3 列的嵌套框架集，其中两边的列实际上共享了相同的文件。图 6-7 给出了这个示例在浏览器中的外观。

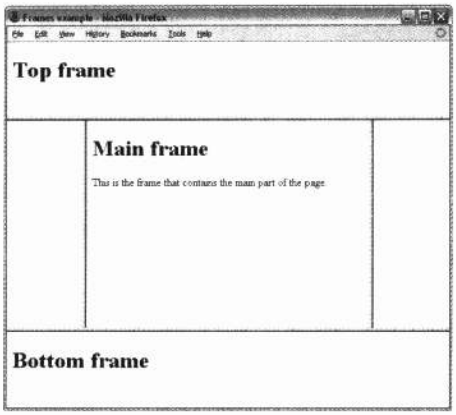


图 6-7

试一试　基于框架的图书阅读器

在这个示例中，您将创建一个基于框架的阅读器，用于预览关于图书的详细描述。

该阅读器的思想是，其中一个页面包含关于新书发布的所有细节。然后在该页面的顶部有一个包含导航条的框架；这个框架不会改变大小或者滚动。另外也有第三个框架，它只包含版权和隐私政策，类似于许多 Web 站点底部所给出的内容。

在开始构建这个示例之前，首先了解需要创建的内容会很有帮助。图 6-8 中给出了这个阅读器的外观。

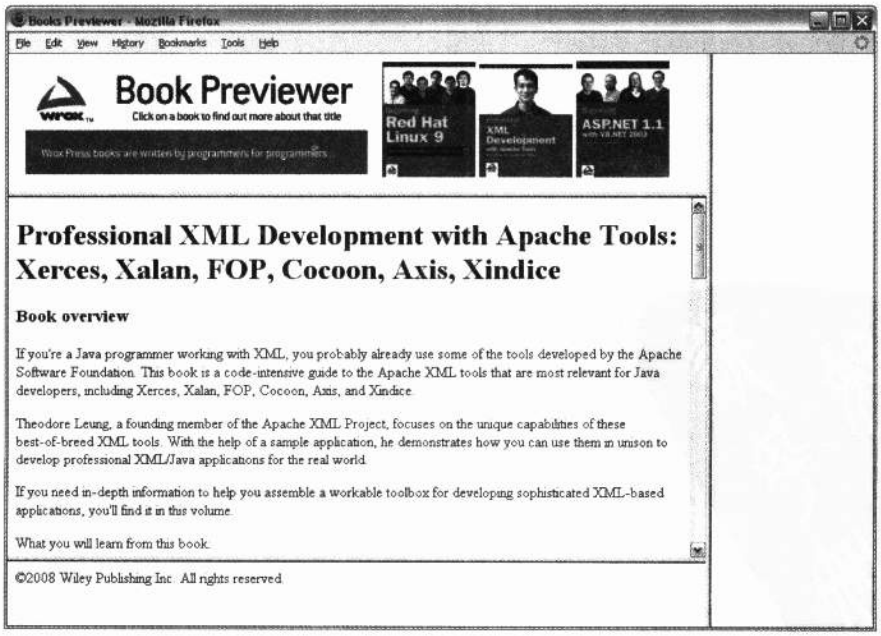


图 6-8

这个示例实际上由 4 个文件组成：

- books.html，它包含了整个文档使用的框架集
- nav.html，它是顶部的框架
- newBooks.html，它是具有所有书籍详细描述的页面
- footer.html，它是包含了页脚图像的页面

需要按照如下顺序完成这些页面的创建：

(1) 准备好文本编辑器或者 Web 页面编辑器，创建框架集文档的程序框架。注意，这个文档将与到目前为止所创建的文档稍有不同。books.html 文件的代码如下所示：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Books Previewer</title>
  </head>
</html>
```

(2) 将页面划分为几个相关的框架。虽然屏幕截图上看上去好像只有 3 个框架，但如果仔细观察，会发现中间框架的边缘没有接触浏览器窗口的右边。此处希望使页面具有固定的宽度。

这个示例包含两个<frameset>元素。第一个元素将页面划分为两列：第一列为 750 像素宽，第二列占用页面的剩余空间(前提是窗口的宽度大于 750 像素)。

在第一列中可以看到第二个<frameset>元素，该元素保存页面的实际内容。在这个嵌套的框架集中有 3 个<frame>元素，在图 6-7 中可以清楚地看到它们。

```
<frameset cols="750, *">
  <frameset rows="150, *, 70" frameborder="1" noresize="noresize" >
     <frame src="frames/nav.html" scrolling="no" />
     <frame src="frames/newBooks.html" name="main_page" />
     <frame src="frames/footer.html" scrolling="no" />
  </frameset>
</frameset>
  <noframes><body>
     This site uses a technology called frames. Unfortunately, your
     browser does not support this technology. Please upgrade your
     browser and visit us again!
<body></noframes>
```

可以看到 3 个<frame />元素，每一个都指向自己的文件。

(3) 创建称为 nav.html 的新文件，用于组成窗口顶部导航框架的内容。这只是一个普通的 XHTML 文档，因此像平常一样首先给出页面的程序框架。另外还需要在<head>元素中添加一个<style>元素，以防止(作为链接的)图像的周围出现边框(在第 9 章中将更多地介绍这方面的内容，现在仅需要复制相应代码)。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Navigation</title>
  <style type="text/css">img {border-style:none; border-width:0px;}</style>
</head>
<body>
</body>
</html>
```

(4) 在 nav.html 页面中，导航栏的左边有一副图像，它是该页面的标题。然后有 3 幅图像，它们是指向主框架中页面的不同部分的链接：

```
<img src="../images/nav_left.gif" width="390" height="125" alt="Books
Preview" />
<a href="../frames/newBooks.html#linux" target="main_page">
   <img src="../images/543784.jpg" width="100" height="125" /></a>
<a href="../frames/newBooks.html#xml" target="main_page">
   <img src="../images/543555.jpg" width="100" height="123" /></a>
<a href="../frames/newBooks.html#asp" target="main_page">
   <img src="../images/557076.jpg" width="100" height="126" /></a>
```

其中<a>元素具有 href 属性和 target 属性。请记住，target 属性用于指明页面应当加载到哪一个框架中。href 属性不仅用于指向应当加载的页面，在这个例子中，它还用于指向该页面的特定位置。

(5) 创建 newBooks.html 文件。该页面的程序框架和基本的文本标记都非常简单；如果需要的话，可以从本章的代码下载中找到它们。在这个普通的 XHTML 页面中，需要重点关注的是，<a>元素用作目标锚点，以便导航窗格中的链接向用户显示页面的适当部分。

<a>元素位于<h1>元素中，用于包含题头(注意，如果目标锚点元素是空的，某些浏览器将无法识别它们)。在下面的代码中可以看到这些题头元素：

```
<h1><a name="xml">Professional XML Development with Apache Tools: Xerces,
      Xalan, FOP, Cocoon, Axis, Xindice</a></h1>
<h1><a name="linux">Beginning Red Hat Linux 9</a></h1>
<h1><a name="asp">Beginning ASP.NET 1.1 with VB .NET 2003</a></h1>
```

(6) 创建 footer.html 文件，它是非常简单的纯 XHTML 页面，在该页面中具有一副图像：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Footer</title>
</head>
<body>
  <img src="../images/footer.gif" alt="Copyright and privacy statement"
```

```
        width="700" height="40" />
</body>
</html>
```

这个框架集在浏览器中的外观如图 6-8 所示，因此应当再次仅关注一些关键点。

工作原理

第一眼看上去，这个示例仅包含 3 个框架。通过密切观察可得知，为了获得固定宽度，保存内容的 3 个框架位于另一个框架集中。包含框架集的一列具有 750 像素的固定宽度，该列中具有一个<frameset>元素而不是相应的<frame>元素。因为第二列是空的，所以它也不需要有<frame>元素。

```
<frameset cols="750, *">
  <frameset rows="150, *, 70" frameborder="1" noresize="noresize" >
     <frame src="frames/nav.html" scrolling="no" />
     <frame src="frames/newBooks.html" name="main_page" />
     <frame src="frames/footer.html" scrolling="no" />
  </frameset>
</frameset>
```

其中嵌套的框架集将页面明显地划分为 3 行，每一行具有一个对应的<frame>元素。嵌套的<frameset>元素附带了 noresize 属性，以阻止用户调整组成每一行的不同框架的大小。

另外注意，第一个和最后一个<frame>元素附带了 scrolling 属性，并且该属性的值是 no，以阻止这些框架具有滚动栏。

这个示例演示了如何利用框架允许用户对一些非常长的文档进行导航(可以想像，在主页面中可能具有很多书籍)。它也演示了对于单个 Web 页面，最终可能会获得非常多的文件；以及必须仔细地跟踪这些页面和记住哪一个页面出现在哪一个框架中。可以看到，导航框架中的<a>元素非常复杂，它指明了哪一个框架是目标框架，并且必须提供文档源。

```
<a href="../frames/newBooks.html#linux" target="main_page">
   <img src="../images/543784.jpg" width="100" height="125" /></a>
<a href="../frames/newBooks.html#xml" target="main_page">
   <img src="../images/543555.jpg" width="100" height="123" /></a>
<a href="../frames/newBooks.html#asp" target="main_page">
   <img src="../images/557076.jpg" width="100" height="126" /></a>
```

虽然这些链接并不是很复杂，但是如果具有 3 个框架，每一个框架都有指向其他框架的链接，则情况将变得相当复杂。然而，这个示例仅用于演示框架集的工作原理。

6.8 利用<iframe>元素创建浮动框架或内联框架

另一种特殊的框架通常称为 iframe(但有时称为内联框架或者浮动框架，因为它可以出现在 HTML 或者 XHTML 页面内的任何位置)；这种框架不需要出现在<frameset>元素内或者使用框架集文档类型声明的文档内。

可以使用<iframe>元素创建浮动框架，与图像类似的是，内联框架能够具有环绕其的文字，并且可以设置围绕浮动框架的边框和页边空白。

下面是一个关于浮动框架的简单示例(ch06_eg08.html)：

```
<body>
<h1>Floating frame</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut risus
tellus, hendrerit id, sagittis sed, lobortis eget, augue.
  <iframe src="frames/iframe.html">
    Error! You should be seeing our news in this window.
    This site uses a technology called frames which is not supported by older
    browsers. If you are using a version of Internet Explorer older than
    version 3 or a version of Netscape older than version 6 you might need
    to upgrade yourbrowser.
  </iframe>
 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut risus tellus,
hendrerit id, sagittis sed, lobortis eget, augue.</p>
</body>
```

注意，在起始标签<iframe>和结束标签</iframe>之间是当浏览器不支持<iframe>元素时显示的消息。

即使不在起始标签<iframe>和结束标签</iframe>之间添加消息，也不能将<iframe>作为一个空元素；否则在某些浏览器中将不会正确显示它。因此，仍然需要添加起始标签<iframe>和结束标签</iframe>，但不包含内容。

这个页面在浏览器中的外观如图 6-9 所示。

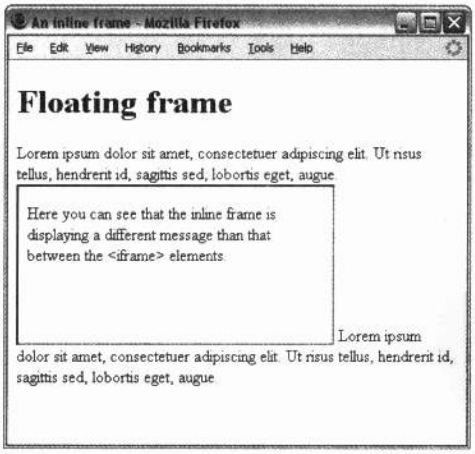


图 6-9

注意：

虽然直到 HTML 4.0 才引入<iframe>元素，但它首先出现在 IE 3 和 Netscape 6 浏览器中。

<iframe>元素

<iframe>元素位于一个普通 XHTML 页面的中部，用于创建内联框架。它必须附带的唯一属性是src(该属性的值是文档中包含<iframe>元素的页面的URL)，但是最好添加height属性和 width 属性以控制它的尺寸。请记住，这个元素不需要是框架集文档类型的一部分。

除了通用属性之外，<iframe>元素还可以附带如下属性：

```
align height width frameborder longdesc marginwidth marginheight name
scrolling src
```

注意，不存在特定于<iframe>元素的 CSS 样式或者事件。

1. src 属性

<iframe>元素中必须具有 src 属性，因为 src 属性指明了浏览器能够在何处找到该框架的内容所在的文件，类似于其在<frame>元素中的作用。

2. align 属性(逐渐淘汰)

align属性用于指明浮动框架外部的文本的显示方式，它可以附带表6-5中列举的某个值。

表 6-5

值	目 的
left	框架将与页面的左页边空白对齐，允许它周围的文本出现在右边
right	框架将与页面的页边空白空对齐，允许它周围的文本出现在左边
top	框架的顶部将与环绕它的文本内联
middle	框架的中部将与环绕它的文本内联
bottom	框架的底部将与环绕它的文本内联(是默认设置，如图 6-9 所示)

3. height 属性和 width 属性

height 属性和 width 属性分别用于指定框架的高度和宽度，与图像中的对应属性作用相同。

```
height="250" width="500"
```

height属性和width属性值的单位可以是像素(如同上面的代码行所示)或者浏览器窗口或父元素(如果该元素包含在其他元素中)的百分比(如同下面的代码行所示)。

```
height="20%" width="40%"
```

但需要注意的是，具有不同屏幕分辨率的用户将看到不同的屏幕大小。如果不指定高度或宽度，浏览器的大小将基于屏幕的完整尺寸。

4. frameborder 属性

frameborder 属性用于指定是否显示框架的边框；它的值应该是边框的像素数量。值为 0 表示没有边框。

```
frameborder="0"
```

5. longdesc 属性

longdesc 属性可用于指定一个到其他页面的链接，该页面中具有框架内容的文本描述。如果在框架中放置图像、图表或图形，该属性非常有用，因为文本描述能够使站点对视力受损的用户具有更好的可访问性。如果用户无法正确加载框架，该元素也很有用。

```
longdesc="../textDescriptions/iframe1.html"
```

6. marginheight 属性和 marginwidth 属性

marginheight 属性和 marginwidth 属性用于指定框架边框和框架内容之间的距离，单位为像素。

```
marginwidth="8" marginheight="8"
```

marginwidth 属性用于指定框架的左边框和右边框与框架内容之间的距离，而 marginheight 属性用于指定框架的顶部边框和底部边框与框架内容之间的距离。

7. scrolling 属性

scrolling 属性指定框架是否应该具有滚动栏(类似于<frame>元素中的 scrolling 属性的作用)。

试一试　使用内联框架

在这个示例中，需要为儿童创建一个简单页面以学习水果。该页面允许儿童单击一个链接，然后将对应的图像加载到内联框架中，而不改变页面的剩余部分。

(1) 创建标准 Transitional XHTML 文档的程序框架，如下所示：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Iframe example</title>
</head>
<body>
</body>
</html>
```

(2) 添加一个题头，然后添加<iframe>元素，其中<iframe>元素必须具有一个 name 属性。在这个示例中，需要使用 scrolling 属性来阻止框架具有滚动栏。另外也需要设置框架

的大小——使其宽和高均为 150 个像素：

```
<h1>Learn your fruit</h1>
  <iframe name="iframe" height="150" width="150" scrolling="no">Pictures of
     fruit will appear here</iframe>
```

(3) 添加一些链接，通过这些链接可以将一些图像加载到内联框架中。与其他类型的框架一样，如果希望让链接加载页面到另一个框架中，则链接必须附带 target 属性，该属性的值是内联框架的名称。

```
<p>Click on the name of the fruit to see an image of it appear.</p>
<p>The <a href="images/orange.jpg" target="iframe">orange</a> is an
   orange colored fruit.</p>
<p>This <a href="images/apple.jpg" target="iframe">apple</a> is red
   and crunchy.</p>
<p>The <a href="images/banana.jpg" target="iframe">banana</a> is
   long and yellow.</p>
```

这个示例在浏览器中的外观如图 6-10 所示。

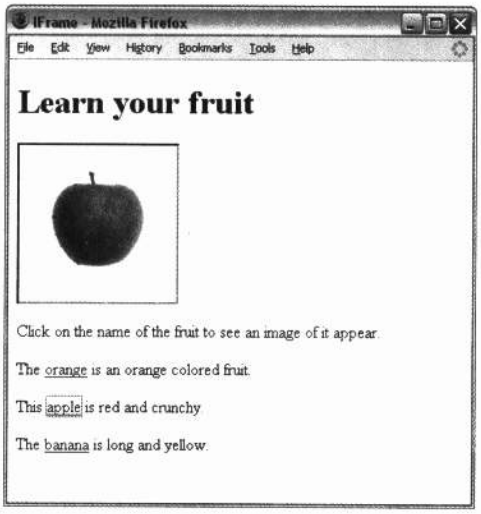


图 6-10

工作原理

<iframe>元素非常有用，因为它允许仅刷新页面的一部分，而不用重新加载整个页面，该元素也能够是一个普通 XHTML 文档的一部分。

一旦在页面中添加了<iframe>元素，可以仅在页面中创建一些链接，这些链接将新的内容加载到内联框架中。在这个示例中，当页面加载时内联框架中没有初始内容，尽管可以使用 src 属性来指示一个默认文件，以便页面加载时显示该文件。

记住，<iframe>元素不能是一个空元素，在起始标签和结束标签之间是当框架无法加载时希望用户看到的内容。

6.9 本章小结

本章介绍了框架，它能够将浏览器划分为多个独立的窗格。每一个窗格包含一个独立的 XHTML 文档，可以独立于其他框架加载或重新加载该文档。

如果页面的部分内容保持相同而主体改变，框架就非常有用——例如，当主体非常长(并且希望导航始终可见)或者加载导航的时间非常长(并且不希望为每一个页面重新加载它)时。

本章介绍了两种类型的框架：

- 传统的框架集文档，它使用<frameset>元素将屏幕划分为多行和多列。然后，<frameset>元素为窗口的每一部分包含一个<frame>元素。这些框架属于框架集文档类型，并且需要与其他 XHTML 文档不同的 DOCTYPE 声明，因为<frameset>元素替代了<body>元素。
- 最近出现的内联框架或者浮动框架，它位于一个普通的 XHTML 页面中，并允许仅重新加载该框架的内容。内联框架可以出现在文档内的任何位置。

本书前面已经提到，可以使用 AJAX 代替框架来重新加载页面的部分内容。在本书后面的内容中您将看到，可以使用表或者 CSS 来控制展示的内容，因此当今最常用的框架是内联框架。

6.10 练习

所有练习的答案都在附录 A 中给出。

1. 重新创建图 6-11 所示的框架集文档(在该文档中，单击一种水果将会加载一个新页面到主窗口中)。当页面加载时，它将附带对应水果的详细描述。

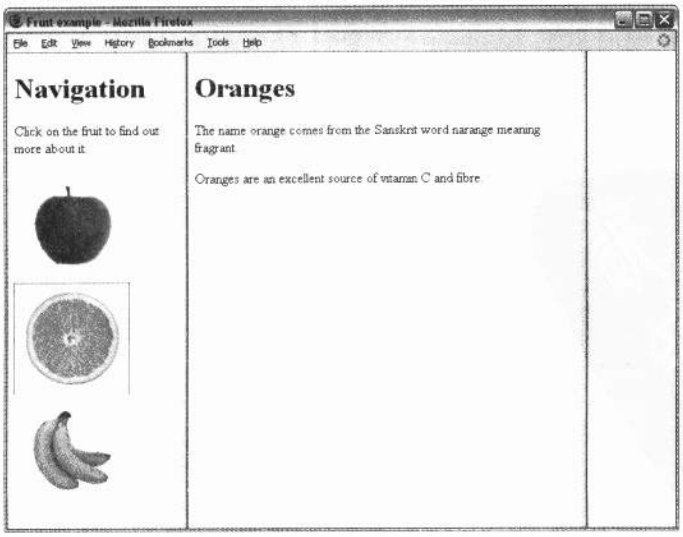


图 6-11

2. 重新创建图 6-12 所示的<iframe>元素。

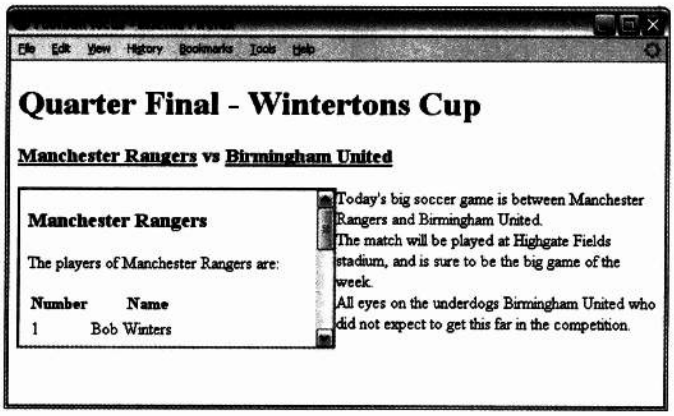


图 6-12

第 7 章

层叠样式表

本书前面已经介绍了如何使用 XHTML 的各种元素和属性来构造文档的内容，接下来将学习如何使页面看上去更生动。

本章将介绍如何使用层叠样式表(简写为 CSS)控制页面的样式，包括字体的颜色和大小、线的宽度和颜色、页面中各项之间的空白量。层叠样式表规范的工作原理在于允许用户指定一些规则，这些规则描述了文档中元素内容的表现形式。事实上，可以通过设置不同的规则来控制页面中每一个元素的外观，以便页面看上去更令人感兴趣。

本书前面已经提到，早期版本的 HTML 使用 Web 页面标记中的元素和属性来控制文档的外观(本书前面已经使用过这样的方式)。但是，W3C(负责 Web 技术开发的组织)做了一个相当令人震惊的决定，即 HTML 和 XHTML 语言不再包含指示文档外观的指令——而是使用 CSS 来控制 Web 页面的外观。

W3C 已经实际地发布了两个版本的 CSS。本章中介绍的特性和功能来源于 CSS1 和 CSS2(CSS2 根据 CSS1 扩展)。目前 W3C 正致力于再一次更新 CSS，更新后的结果将称为 CSS3；CSS3 方面的开发工作将很快完成；但是，本书后面的一些地方将会提醒您，浏览器已经准备开始实现 CSS3 某些方面的内容。您也将看到，当前的浏览器仍然无法支持某些特性。CSS 非常有意义的一个方面是，即使浏览器没有实现某些 CSS 特性，用户仍然能够阅读文档——只是页面的外观看上去与用户计划的差距很大。

本章将介绍以下内容：

- CSS 规则的组成
- 如何在文档中放置 CSS 规则，以及如何链接到外部 CSS 文档
- 如何使用特性和值控制文档中不同元素的外观
- 如何使用 CSS 控制文本的外观
- CSS 如何基于框模型，如何为这些框设置不同的特性(例如边框的宽度和样式)

学习完本章后，您应当能够熟练编写 CSS 样式表，应当掌握很多特性，通过在样式表中使用这些特性来影响文档的外观。

在第 8 章中将介绍 CSS1 和 CSS2 中的一些更高级的特性，以及如何使用 CSS 定位页面中元素的内容。

注意：

自从引入 Web 之后，人们期望能够像打印设计人员控制打印页面那样来控制 Web 页面。但是，作为媒介的 Internet 与打印媒介有一些固有的区别。例如，一本书中的打印页面在该书的每一个副本中都具有相同的大小；读者不需要拥有一种字体以查看打印页面(对于 Web 页面，则需要有相应的字体)，也不需要亲自选择打印页面。这些都是第 9 章和第 10 章中将要解决的问题，这两章主要介绍关于页面布局和设计方面的问题。

7.1 CSS 简介

CSS 的工作原理是允许设计人员将规则与文档中出现的元素相关联。这些规则管理元素的内容的显示方式。图 7-1 给出了关于 CSS 规则的一个示例，它由两部分组成：

- 选择器，它指定声明应用于哪个或哪些元素(如果应用于多个元素，则不同元素之间使用逗号隔开)
- 声明，它用于设置元素的样式

图 7-1

图 7-1 中的规则将应用于所有<h1>元素，并且指示这些元素的字型是 Arial。

声明也由以冒号隔开的两部分组成：

- 特性，它是希望影响的所选元素的特性，这个例子中的特性是 font-family。
- 值，它是特性的声明，这个例子中的值是 Arial 字型。

这与 HTML 中元素附带属性的方式非常相似，其中属性能够控制元素的特性，属性值将是该特性的设置。但是，通过利用 CSS，不需要对<h1>元素的每一个实例都指定属性，选择器将指示该规则应用于文档中的所有<h1>元素。

下面是一条 CSS 规则应用于多个不同元素的示例(这个示例中包括 3 个元素：<h1>、<h2>和<h3>)。应用该规则的每一个元素的名称之间以逗号隔开。该规则还为这些元素指定了多个特性，其中每个特性-值对之间以分号隔开。注意，所有的特性都位于大括号中：

```
h1, h2, h3 {
    font-weight:bold;
    font-family:arial, verdana, sans-serif;
    color:#000000;
    background-color:#FFFFFF;}
```

即使您以前没有看到过 CSS 规则，现在也应当已经了解它们的作用。在上面的例子中，选择器(<h1>、<h2>和 <h3>)中指定的每一个题头元素的内容将以 Arial 粗体字显示(除非计算机上没有安装 Arial 字体。如果没有安装该字体，则将查找是否安装 Verdana 字体。如果

找不到 Verdana 字体，则使用默认的 sans-serif 字体)，并且以白色背景黑色字的形式显示。

注意：

如果声明中只有一个特性-值对，则它的末尾不需要分号。但是，由于一个声明可以由多个特性-值对组成，并且同一个规则中的每一个特性-值对必须由分号隔开，所以最好在编写每一个规则之后都添加一个分号，以便后面添加另外一条规则；如果忘记添加分号，则后面添加的特性-值对将被忽略。

7.1.1 一个基本的示例

下面的示例使用了大量的 CSS 规则，其中大多数规则的作用都可以通过它们的名称清晰地了解。根据这个示例，您将了解 CSS 的其他不同方面，例如如何控制文本、表、空白和背景等。

在开始创建该示例之前，首先查看没有添加 CSS 规则的 XHTML 文档，下面将逐步处理这个文档。图 7-2 给出了该文档不具有样式时在浏览器中的外观。

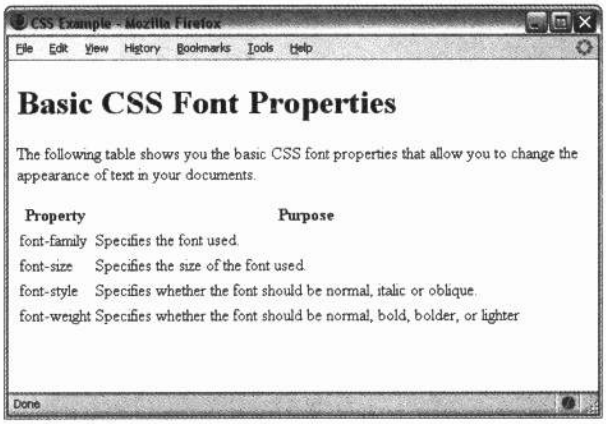


图 7-2

下面是图 7-2 中所示文档的代码(ch07_eg01.html)，它包含了一个题头、一个段落和一个表。注意<head>元素内<link>元素的使用，它告诉浏览器这个文档应当被<link>元素附带的 href 属性值中指定的样式表赋予样式。另外注意<td>元素附带的 class 属性，它的值是 code；使用该属性值区分包含代码的<td>元素与文档中的其他文本。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>CSS Example</title>
  <link rel="stylesheet" type="text/css" href="ch07_eg01.css" />
</head>
<body>
```

```
    <h1>Basic CSS Font Properties</h1>
    <p>The following table shows you the basic CSS font properties that allow
you
    to change the appearance of text in your documents.</p>
    <table>
      <tr>
        <th>Property</th>
        <th>Purpose</th>
      </tr>
      <tr>
        <td class="code">font-family</td>
        <td>Specifies the font used.</td>
      </tr>
      <tr>
        <td class="code">font-size</td>
        <td>Specifies the size of the font used.</td>
      </tr>
      <tr>
        <td class="code">font-style</td>
        <td>Specifies whether the font should be normal, italic or oblique.</td>
      </tr>
      <tr>
        <td class="code">font-weight</td>
        <td>Specifies whether the font should be normal, bold, bolder,
        or lighter</td>
      </tr>
      </table>
    </body>
    </html>
```

图 7-3 给出了添加样式表之后的文档外观。

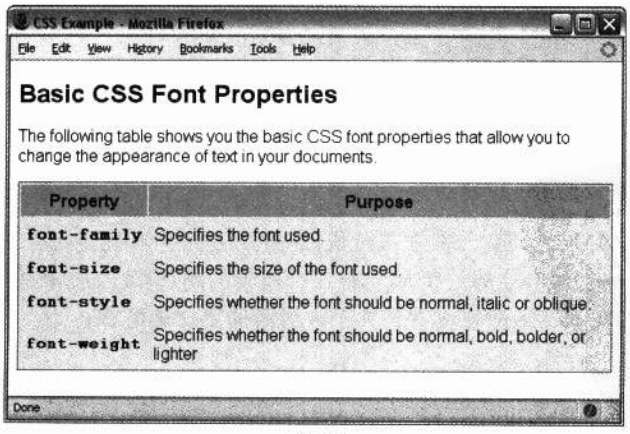


图 7-3

注意，查看这个文档中使用的样式表。所有的 CSS 样式表都存储为以.css 为扩展名的文件，这个示例的样式表文件为 ch07_eg01.css。

可以在与用于创建 XHTML 页面的相同编辑器中创建 CSS 样式表，因为 CSS 文件只是简单的文本文件(类似于 XHTML 文件)，也可以在 Windows 的 Notepad 软件或者 Mac 的 TextEdit 软件中创建它们。

下面依次查看样式表中的规则，以便了解每一条规则的作用。该文档由多条独立的规则组成，除了第一行之外——该行不是规则，而是一条注释。起始符号“/*”和结束符号“*/”之间的内容将被浏览器忽略，因此不会显示它们：

```
/* Style sheet for ch07_eg01.html */
```

第一条规则应用于<body>元素。它指定页面中文本和线的默认颜色是黑色，页面的背景是白色，并且整个文档中使用的字体将是 Arial。如果系统中没有安装 Arial 字体，则将使用 Verdana 字体；如果没有找到 Verdana 字体，则将使用任何一种 sans-serif 字体。

```
body {
    color:#000000;
    background-color:#ffffff;
    font-family:arial, verdana, sans-serif; }
```

注意：

一般为文档主体指定 background-color 特性，因为某些人可能会改变他们计算机的默认背景色(使得默认背景色不是亮白色)；如果不设置这个特性，则用户浏览器的背景色将是他们已经选择的颜色。

接下来的两条规则分别指定<h1> 元素和 <p>元素内的内容的大小：

```
h1 {font-size:18pt;}
p {font-size:12pt;}
```

然后，添加一些设置来控制表的外观——首先赋予它亮灰色背景，然后在表的周围绘制一个像素宽的黑灰色边框：

```
table {
    background-color:#efefef;
    border-style:solid;
    border-width:1px;
    border-color:#999999;}
```

在表内部，题头应当具有一种中灰背景色(稍微比表主体的颜色黑一点)，文本应当以粗体显示，在单元格边缘和文本之间应当有 5 个像素的内边距(内边距是方框边缘和其内部内容之间的距离，在本章后面您将看到关于内边距的更详细解释)。

```
th {
    background-color:#cccccc;
    font-weight:bold;
    padding:5px;}
```

每一个表数据单元格具有 5 个像素的内边距。这样能够使文本具有更好的可读性，否则某一列中的文本可能会与相邻列中的文本紧密相连。

```
td {padding:5px;}
```

最后，在图 7-3 中您可能已经注意到，CSS 特性涉及的表单元格中的字体为 Courier。这是因为在该 XHTML 文档中对应的表单元格附带了 class 属性，并且该属性的值为 code。class 属性无法单独改变文档的外观(如图 7-2 所示)。但是，class 属性能够将 CSS 规则与 class 属性具有某个指定值的元素相关联。因此，下面的规则仅应用于附带的 class 属性值为 code 的<td>元素，而不是所有的<td>元素：

```
td.code {
    font-family:courier, courier-new, serif;
    font-weight:bold;}
```

到此为止就完成了第一个示例，可以在本书的下载代码中找到这个示例。如果希望从一个 Web 站点处查看类似于这样的样式表，可以在浏览器中输入该样式表的 URL，然后将在浏览器中看到它的文本，或者它将下载到本地计算机上。可以使用下载的代码试验该样式表，以查看样式表在计算机上的显示；为了完成该试验，当在浏览器中查看上面的示例时，将文件名 ch07_eg01.html 替换为 ch07_eg01.css，则将看到 CSS 规则在浏览器中的效果。

7.1.2　继承

CSS 的一项强大功能是，应用于某个元素的许多特性将会被它的子元素(即包含在对其声明规则的元素之内的元素)所继承。例如在上面的示例中，一旦为<body>元素声明了 font-family 特性，它将应用于<body>元素内的所有元素(<body>元素的所有子元素)。

如果后面指定了更加特定的规则，则更加特定的规则将重写与<body>元素(或者其他任何包含元素)相关联的任何特性。在上面的示例中，大多数文本的字体是 Arial，由与<body>元素相关联的规则指定。有些表单元格使用 Courier 字体，这些不同的表单元格具有一个 class 属性，并且该属性的值为 code：

```
<td class="code">font-size</td>
```

下面是与这些元素关联的规则：

```
td.code {
    font-family:courier, courier-new, serif;
    font-weight:bold;}
```

这条规则的优先级高于与<body>元素关联的规则，因为关于所应用元素的选择器更加特定。

特性的继承性将节省为每一个元素编写规则和所有特性-值对方面的工作，从而使样式表更加紧凑。附录 C 中包含关于 CSS 特性的便利参考手册，其中给出了哪些特性具有继承性，哪些特性不具有继承性。

7.2 添加 CSS 规则的位置

本章起始处的示例使用了一种独立的样式表(或外部样式表)来包含 CSS 规则。这涉及在 XHTML 文档头中使用<link />元素来指明哪个样式表应当用于控制文档的外观。

CSS 规则还可以出现在 XHTML 文档内的两个位置处：

- 在<head>元素内，通过<style>元素包含特性
- 作为元素的 style 属性的值，前提是该元素附带 style 属性

当样式表规则保存在文档头中的<style>元素内时，它们称为内部样式表。

```
<head>
   <title>Internal Style sheet</title>
   <style type="text/css">
   body {
     color:#000000;
     background-color:#ffffff;
     font-family:arial, verdana, sans-serif; }
   h1 {font-size:18pt;}
   p {font-size:12pt;}
</style>
</head>
```

当 style 属性用于 XHTML 元素时，样式表规则称为内联样式规则。例如：

```
<td style="font-family:courier; padding:5px; border-style:solid;
border-width:1px; border-color:#000000;">
```

这些特性添加为 style 属性的值。这里不需要选择器(因为样式自动地应用于附带 style 属性的元素)，并且不需要大括号。但是，仍然需要利用冒号将特性和它的值隔开，并且每一个特性-值对之间以分号隔开。

在 Transitional XHTML 中 style 属性被逐渐淘汰，而在 Strict XHTML 1.0 中则不允许使用该属性，因为它引入了样式标记，而这些 XHTML 规范要求文档只应该包含用于解释文档的语义和结构的标记。

7.2.1 <link>元素

<link />元素可用于创建到 CSS 样式表的链接。<link />元素始终是空元素，用于描述两个文档之间的关系。可以通过多种方式使用该元素，而不只是样式表。例如，该元素可用于描述到一个 RSS 提要(RSS 提要对应于一个页面)的链接，但当用于样式表时，该关系表明包含<link />元素的文档将使用指定 CSS 文档中包含的规则来展现。

当<link />元素用于将样式表附加到文档时，相对于使用<a>元素创建的链接种类，它创建一种非常不同的链接种类。因为样式表自动地关联到文档，所以用户不用单击任何对象来激活该链接。

注意：

<link />元素最初也能够用于创建一系列有序页面之间的导航。但是，当用于这种情形时，主要的浏览器都将忽略<link />元素。

当用于样式表时，<link />元素必须附带 3 种属性：type、rel 和 href。下面的示例是一个称为 interface.css 的 CSS 文件中的<link />元素，该文件位于 stylesheets 文件夹中：

```
    <link rel="stylesheet" type="text/css"
href="../stylesheets/interface.css" />
```

除了一些核心属性，<link />元素也能够附带如下属性：

```
    charset dir href hreflang media rel rev style target type
```

本书前面已经介绍了其中的一些属性，因此下面的章节中讨论其中一些比较重要的属性以及一些较少使用的属性。

1. rel 属性

rel 属性是必需的属性，用于指明包含链接的文档和链接到的文档之间的关系。

用于操作样式表的关键值是 stylesheet。

```
    rel="stylesheet"
```

第 1 章中介绍过该元素的其他可能值。

2. type 属性

type 属性指明将链接到的文档的 MIME 类型；这里由于处理的是 CSS 样式表，因此 MIME 类型是 text/css：

```
    type="text/css"
```

附录 H 中列举了其他 MIME 类型。

3. href 属性

href 属性指定将链接到的文档的 URL。

```
    href="../stylesheets/interface.css"
```

这个属性的值可以是绝对或者相对 URL。

4. hreflang 属性

hreflang 属性指定了编写指定资源的语言，它的值必须是附录 G 中指定的语言代码之一。

```
    hreflang="en-US"
```

5. media 属性

media 属性指定了计划用于文档的输出设备：

```
media="screen"
```

这个属性正变得越来越重要，因为人们访问 Internet 的方式多样化并使用不同的设备，表 7-1 中给出了该属性的可能值。

表 7-1

值	用　途
screen	非分页的计算机屏幕
tty	具有固定间距字符网格的媒体，例如电传打字设备、终端或者具有有限显示能力的移动设备
tv	具有低分辨率、彩色屏幕和有限滚动页面能力的 TV 设备
print	打印文档，有时称为分页媒体(以及屏幕上以打印预览模式显示的文档)
projection	投影仪
handheld	便携式设备，特点是屏幕小、位图图形和有限的带宽
braille	Braille 触觉反馈设备
embossed	Braille 分页打印机
aural	语音合成器
all	适合于所有设备

7.2.2 <style>元素

<style>元素用于<head>元素内部，作用是在文档中包含样式表规则，而不是链接到一个外部文档。当文档中需要包含少量额外规则，并且这些规则不会应用于其他共享相同样式表的文档时，有时也可以使用该元素。

例如，下面示例中的 XHTML 文档使用<link />元素包含一个样式表，并且利用一个<style>元素包含一条额外的规则：

```
<head>
   <title>
   <link rel="stylesheet" type="text/css" href="../styles/mySite.css" />
   <style type="text/css">
     h1 {color:#FF0000;}
   </style>
</head>
```

<style>元素可以附带如下属性：

```
dir lang media title type
```

有些浏览器也支持 id 属性和 src 属性，尽管这两个属性不是 W3C 规范的内容。

注意：

许多文档设计人员在<style>元素内部添加注释符，从而所有的 CSS 规则在作为内部样式表时出现在<!--和-->标记之间。主要思想是，有些较老的浏览器不能理解 CSS，这样做能够向这些较老的浏览器隐藏这些规则的代码。这种技术的缺陷是，许多较新的浏览器允许分离 XHTML 注释的内容，并且不处理它们(但现在流行的浏览器都不具有这种功能)。这样，将来的某些浏览器能够忽略所有的样式规则。实际上，服务器也能够分离注释，不将它们发送到客户端。事实上，很可能访问您的站点并且因为不使用注释包含样式规则而导致浏览问题的浏览器数量非常少，从而可以如前面所示完全忽略他们。

7.2.3　外部 CSS 样式表的优点

如果两个或者更多的文档需要使用同一个样式表，则应该始终使用一个外部样式表(但有时也需要使用一个内部样式表以重写外部样式表中的规则)。

比起内部样式表或者内联的样式规则，外部样式表具有很多优点，主要包括以下一些方面：

- 相同的样式表可以被站点中的所有 Web 页面重用，这将节省为每一个独立的文档都包含样式标记的工作。
- 由于仅需要编写一次样式规则，比起在每一个元素或者每一个文档中都编写它们，这样得到的源文档将小得多。这意味着，一旦 CSS 样式表被使用它的第一个文档下载，随后的文档将能够更快速地下载它(因为浏览器保留了该 CSS 样式表的一份副本，并且不需要为每一个页面都下载这些规则)。这也减少了服务器(服务器是将 Web 页面发送给访问站点的人的计算机)的工作量，因为它发送较少的页面。
- 通过仅改变一个样式表可以改变多个页面的外观，而不需要单独改变每一个页面；当希望改变公司的颜色方案或者用于特定类型元素(该元素出现在整个站点中)的字体时，这种方法也是非常有用的。
- 样式表可以作为样式模板，用于帮助不同的作者获得相同的文档样式，而不需要学习所有单个的样式设置。
- 因为源文档不包含样式规则，所以不同的样式表可以被附加到同一个文档中。因此对于同一个 XHTML 文档，当访问者使用的是桌面计算机时可以使用一个样式表，当访问者使用的是便携式设备时可以使用另外一个样式表，而当页面正在打印或者在 TV 上查看页面时，则可以使用其他类型的样式表等。对于同一个文档，可以为不同的访问者使用不同的样式表。
- 一个样式表可以导入或使用其他样式表中的样式，从而实现模块化开发以及更好的重用性。
- 如果删除样式表，则将使站点对于视力受损的人来说具有更好的可访问性，因为不再控制字体和颜色方案。

因此，公平地说，当编写整个站点时应当使用一个外部样式表来控制显示效果，但在下一章中您将看到可以为站点的不同方面使用多种不同的样式表。

7.3 CSS 特性

现在您已经了解 CSS 的背景、如何编写 CSS 规则以及可以在何处放置这些规则等，本章剩余的部分将介绍一些特性，可以使用它们影响文档的显示效果。特别地，本章将介绍 font、text、border、padding 和 margin 等特性。

表 7-2 给出了 CSS1 和 CSS2 中的主要特性，在本章或第 8 章中将会介绍所有这些特性。

表 7-2

字体	边框	内边距	表
font	border	padding	border-collapse
font-family	border-bottom	padding-bottom	border-spacing
font-size	border-bottom-color	padding-left	caption-side
font-size-adjust	border-bottom-style	padding-right	empty-cells
font-stretch	border-bottom-width	padding-top	table-layout
font-style	border-color	**容积**	**列表和标记符**
font-variant	border-left	height	list-style
font-weight	border-left-color	line-height	list-style-image
文本	border-left-style	max-height	list-style-position
color	border-left-width	max-width	list-style-type
direction	border-right	min-height	marker-offset
letter-spacing	border-right-color	min-width	**生成的内容**
text-align	border-right-style	width	content
text-decoration	border-right-width	**定位**	counter-increment
text-indent	border-style	bottom	counter-reset
text-shadow	border-top	clip	quotes
text-transform	border-top-color	left	**分类**
unicode-bidi	border-top-style	overflow	clear
white-space	border-top-width	right	cursor
word-spacing	border-width	top	display
背景	**页边空白**	vertical-align	float
background	margin	z-index	position
background-attachment	margin-bottom	**外边框**	visibility
background-color	margin-left	outline	
background-image	margin-right	outline-color	
background-position	margin-top	outline-style	
background-repeat		outline-width	

本书中将不会介绍某些特定的特性，或者是因为它们很少使用，或者是因为对它们的支持较少(例如，本书将避免介绍一些听力方面的样式表，因为支持它们的听力浏览器较少)。可以在下面的 Web 站点中找到关于这些特性的更详细介绍(或者可以查阅一些专门介绍 CSS 的书籍)：

- www.w3.org/style/css/
- www.devguru.com/Technologies/css/quickref/css_index.html
- www.w3schools.com/css/css_reference.asp

7.4　控制字体

可以采用多种特性来控制文档中文本的外观，可以将这些特性划分为如下两组：

- 直接影响字体和其外观的特性
- 对文本具有其他格式影响的特性

表 7-3 列举了能够直接影响字体的特性。

表 7-3

特　性	目　的
font	允许将下面的多个特性组合成一个特性
font-family	指定所使用字体的字体族(用户必须已经在计算机上安装该字体)
font-size	指定字体的大小
font-weight	指定字体是否是正常的、加粗的或者比包含元素更粗
font-style	指定字体是否是正常的、斜体或者倾斜的(倾斜的字体是正常字体的倾斜，而不是字体的独立斜体版本)
font-stretch	可以用于控制字体中实际字母的宽度(而不是它们之间的空间)
font-variant	指定字体是否是正常的或者小型大写字母
font-size-adjust	可以用于修改字体的字符大小宽高比

在开始学习字体之前，首先需要理解一些问题。或许最重要的问题是，字体与字型是不同的概念：

- 字型是字体族，例如 Arial 字体族。
- 字体是某个字体族的一个特定成员，例如 Arial 12-point bold。

可以发现这两个术语经常交换使用，但了解它们的区别还是非常有意义的。

字型趋向于属于以下两组中的一组：serif 字体和 sans-serif 字体。使用 serif 字体的字母具有额外的卷曲。例如，图 7-4 中的第一个单词是 serif 字体，其字母顶部和底部都向后倾斜，而 sans-serif 字体的字母末尾笔直。第三种常用字型是等宽 serif 字体。等宽字体中的每一个字母都具有相同的宽度，而非等宽字体的不同字母具有不同宽度(在 serif 字体和 sans-serif 字体中，*l* 比 *m* 窄)。如图 7-4 中的示例所示。

图 7-4

在一般的打印理论中，对于长段的文本，使用 serif 字体更容易阅读。但是，在 Internet 中这种理论不一定正确；许多人发现 serif 字体很难在屏幕上阅读，很可能是因为屏幕的分辨率不像打印单词那么好。这使得 sans-serif 字体更容易在屏幕上阅读，因为它们没有那么复杂。

为了研究影响字体的特性，大多数示例将遵循一种使用文本段落的类似结构：每一个<p>元素附带具有不同值的 class 属性：

```
<p class="one">Here is some text.</p>
<p class="two">Here is some text.</p>
<p class="three">Here is some text.</p>
```

利用 class 属性可以将不同的样式添加到不同的元素，这些元素共享相同的名称。

7.4.1 font-family 特性

font-family 特性可用于指定应该使用的字型。这个特性最大的缺陷在于，查看页面的人员必须在其计算机上安装所需要的字体，否则不能以该字体看到页面上的文本。但是，可以指定多种字体，如果用户不具有第一种字体，则浏览器将查找列表中的下一种字体(ch07_eg02.css)。

```
p.one {font-family:arial, verdana, sans-serif;}
p.two {font-family:times, "times new roman", serif;}
p.three {font-family:courier, "courier new", serif;}
```

注意：

如果一种字体名包含空格，例如 times new roman 或 courier new，则应当将字体名放置在双引号中。

图 7-5 中给出了这个示例在浏览器中的外观；在图 7-5 中，每一个段落应用不同类型的字体(ch07_eg02.html)。

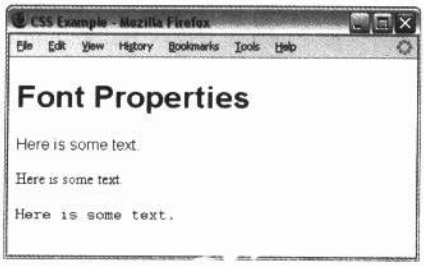


图 7-5

以逗号隔开的可用字体列表必须以 5 种通用字体名之一结尾，以便如果计算机无法找到指定的任意一种字型，则可以使用它的默认通用字体。这些通用字体名如表 7-4 所示。

表 7-4

通用字体名	字 体 类 型	示　例
serif	具有衬线的字体	Times
sans-serif	不具有衬线的字体	Arial
monospace	固定宽度字体	Courier
cursive	仿效手写的字体	Comic Sans
fantasy	用于标题等的修饰性字体	Impact

在选择字体时需要记住的一件事情是，每一种字体可能具有不同的高度或宽度。因此，您很可能希望选择一种相似大小的字体作为首选字体的备选。例如，Courier New 字体比 Impact 字体(Impact 字体非常高和窄)短和宽。如果以某一种字体设计页面，则页面的布局对于所选择的第二种具有不同大小的字体将稍微有些不同。

当设计人员希望使用一种特定的字型但大多数用户计算机上很可能没有该字型时，可以在 GIF 图像中放置文本。对于大段的文本，使用图像的方式通常不合适；但对于徽标或者标题以及其他少量的文本来说，这种解决方法非常不错。如果采用了这种解决方案，请记住必须将图像中看到的文本作为 alt 属性的值提供。

注意：

对于用户计算机上很可能没有安装的字体并且下载该字体也存在问题的情况，为了使用该字体，需要执行一些额外的操作；然而，大多数字体都是有版权的——与软件一样——不能由购买者简单地分发。另外，许多用户对从 Web 站点上下载文件非常谨慎，因此这不能成为获得想要的页面外观的依赖性技术。如果确实希望为少量的文本使用非标准字体，除了图像之外，另外一种方式是在 SIFR 中结合使用 Flash 和 JavaScript，从而能够创建出令人感兴趣的效果(http://novemberborn.net/sifr)。

7.4.2　font-size 特性

font-size 特性用于指定字体的大小，可以以下几种方式指定这个特性的值：

- 绝对大小
- 相对大小
- 长度
- 百分比(相对于父元素)

下面的值是绝对大小：

```
xx-small x-small small medium large x-large xx-large
```

下面的两个值是相对大小：

```
smaller larger
```

长度可以采用下面的一种长度单位来表达：

```
px em ex pt in cm pc mm
```

在本章后面的“长度”一节中，您将看到这些不同单位的含义(因为它们通常与多种特性结合使用，而不仅仅是字体)。其中最常用的一种单位是 px，它代表像素。

百分比计算为相对于包含该文本的元素的比例：

```
2% 10% 25% 50% 100%
```

例如：

```
p.one {font-size:xx-small;}
p.twelve {font-size:12px;}
p.thirteen {font-size:3pc;}
p.fourteen {font-size:10%;}
```

图 7-6 给出了这些不同字体大小在浏览器中的外观(ch07_eg03.html 和 ch07_eg03.css 包含了几种指定字体大小方式的示例，以及它们外观的比较)。

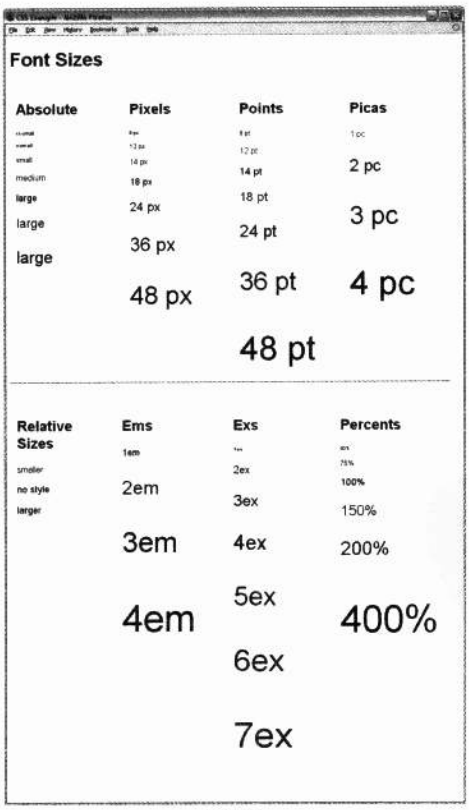


图 7-6

7.4.3　font-weight 特性

大多数字体具有不同的变体，例如加粗和斜体。虽然很多建立完善的字体对于粗体文本中的每一个字符都具有完全不同的版本，但是浏览器将使用一种算法来计算并添加粗体字符的厚度。由于浏览器使用算法，这意味着还可以创建字体的较浅版本。这就是 font-weight 特性的作用。

font-weight 特性的可能值包括：

```
normal bold bolder lighter 100 200 300 400 500 600 700 800 900
```

因此可以采用如下方式赋予文本粗体(ch07_eg04.css)：

```
p.one {font-weight:normal;}
p.two {font-weight:bold;}
p.three {font-weight:bolder;}
p.four {font-weight:lighter;}
p.five {font-weight:100;}
p.six {font-weight:200;}
```

图 7-7 给出了这些值在浏览器中的效果(ch07_eg04.html)。

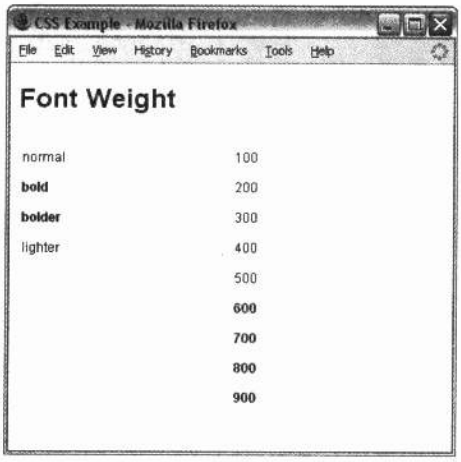


图 7-7

其中 bold 是最常用的值，但是可能也会遇到使用 normal 的情形(特别是如果大量的文本已经是粗体，并且必须创建不同效果的文本)。

7.4.4　font-style 特性

font-style 特性可以用于指定字体是 normal、italic 或 oblique 等，它们是该特性的值；例如：

```
p.one {font-style:normal;}
p.two {font-style:italic;}
```

```
p.three {font-style:oblique;}
```

图 7-8 给出了这些值在浏览器中的效果(ch07_eg05.css)。

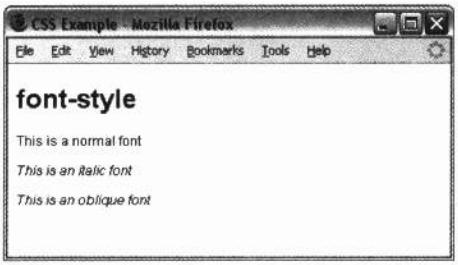


图 7-8

7.4.5 font-variant 特性

font-variant 特性具有两种可能的值：normal 和 small-caps。small-caps 字体看上去类似于大写字母集的较小版本。

例如，请查看下面的段落，其中包含了一个元素，该元素具有一个 class 属性(ch07_eg06.html)：

```
<p>This is a normal font, but then <span class="smallcaps">there
are some small caps</span> in the middle.</p>
```

现在请查看下面的样式表(ch07_eg06.css)：

```
p {font-variant:normal;}
span.smallcaps {font-variant:small-caps;}
```

如图 7-9 所示，与元素关联的规则指明它的内容应当以 small-caps 字体显示。

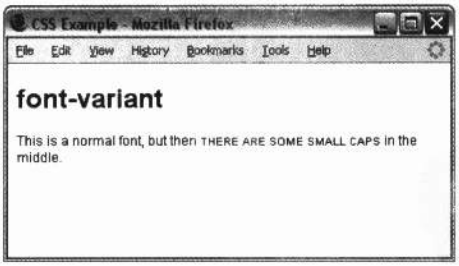


图 7-9

7.4.6 font-stretch 特性

font-stretch 特性设置字体中实际字母的宽度(而不是它们之间的空间)，它的值可以是相对的或者固定的。相对值如下所示。

```
normal wider narrower
```

固定值如下所示：

```
ultra-condensed extra-condensed condensed semi-condensed semi-expanded
expanded extra-expanded ultra-expanded
```

例如，可以使用如下语法创建压缩的 Arial 字体：

```
p {font-family:arial; font-stretch:condensed;}
```

但遗憾的是，这个特性不被 IE 7 或 Firefox 2 浏览器所支持。

7.4.7　font-size-adjust 特性

本章前面已经提到，字体的高度和宽度可以不同。字体的方位值是字体中小写字母 *x* 的高度和字体高度之比。font-size-adjust 特性可用于修改字体的方位值。

例如，Verdana 字体的方位值是 0.58(这意味着当字体的大小是 100px 时，它的 x 高度是 58 像素)。Times New Roman 字体的方位值是 0.46(这意味着当字体的大小是 100px 时，它的 x 高度是 46 像素)。因而当字体较小时，Verdana 字体比 Times New Roman 字体具有更好的可阅读性。因此，通过修改字体的方位值，可以改变它的高度。

遗憾的是，这个特性不被 IE 7 或 Firefox 2 浏览器所支持。

7.5　文本格式化

除了一些字体特性之外，可以使用其他多种特性影响文本的外观或格式化，这些特性如表 7-5 所示。

表 7-5

特　性	目　的
color	指定文本的颜色
text-align	指定包含元素内的文本的对齐方式
vertical-align	指定包含元素内和与包含元素相关的文本的垂直对齐方式
text-decoration	指定文本是否具有下划线、上划线、删除线或闪烁
text-indent	指定文本相对于左边框的缩进距离
text-transform	指定元素的内容为全部大写、全部小写或者首字母大写
text-shadow	指定文本具有阴影
letter-spacing	控制字母间的宽度(对于打印设计人员来说，这称为字距调整)
word-spacing	控制每个单词间的空间量
white-space	指定空白是否应当被折叠、保留或者不能换行
direction	指定文本的方向(类似于 dir 属性)
unicode-bidi	用于创建双向文本

7.5.1 color 特性

color 特性可用于指定文本的颜色。这个特性的值可以颜色的是十六进制编码或者颜色名(附录 D 中详细讨论了 Web 中颜色的指定方式)。

例如，下面的规则将使段落元素的内容为红色(ch07_eg07.html)：

```
p {color:#ff0000;}
```

7.5.2 text-align 特性

text-align 特性的作用类似于逐渐淘汰的 align 属性，它用于指定包含元素或者浏览器窗口内的文本的对齐方式。表 7-6 中给出了该属性的可能值。

表 7-6

值	目　的
left	将文本与包含它的元素的左边框对齐
right	将文本与包含它的元素的右边框对齐
center	将文本居中放置在包含它的元素的中间
justify	将文本的宽度扩展为包含它的元素的整个宽度

例如，下面的示例作用于具有 500 像素宽的表。每一行的规则如下所示(ch07_eg08.css)：

```
td.leftAlign {text-align:left;}
td.rightAlign {text-align:right;}
td.center {text-align:center;}
td.justify {text-align:justify;}
```

图 7-10 中给出了该示例在浏览器中的显示效果。

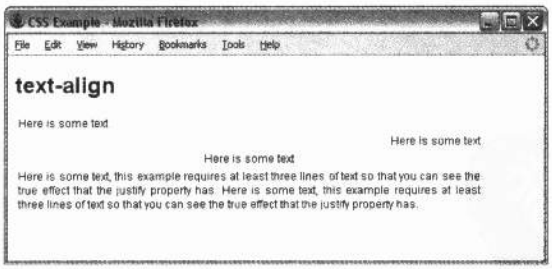


图 7-10

7.5.3 vertical-align 特性

当操作内联元素(特别是图像和文本的某些部分)时，vertical-align 特性非常有用。可使用该属性控制内联元素在包含元素内的垂直位置。

```
span.footnote {vertical-align:sub;}
```

该特性可以采用多种值，如表 7-7 所示。

表 7-7

值	目　的
baseline	所有内容都与父元素的基线对齐(该值为默认设置)
sub	使元素成为下标。对于图像，它的顶部应当与基线对齐。对于文本，字体主体的顶部应当与基线对齐
super	使元素成为上标。对于图像，它的底部应当与字体的顶部平齐。对于文本，下行字母(位于文本行下方的 *g* 或 *p* 等字母的部分)的底部应当与字体主体的顶部对齐
top	文本的顶部和图像的顶部应当与该行中最高元素的顶部对齐
text-top	文本的顶部和图像的顶部应当与该行中最高文本的顶部对齐
middle	元素的垂直中点应当与父元素的垂直中点对齐
bottom	文本的底部和图像的底部应当与该行中最低元素的底部对齐
text-bottom	文本的底部和图像的底部应当与该行中最低文本的底部对齐

该特性也可以接受长度或百分比值。

可以在浏览器中使用 ch07_eg09.html 文件试验所有这些值的效果。

图 7-11 给出了其中某些值的效果。

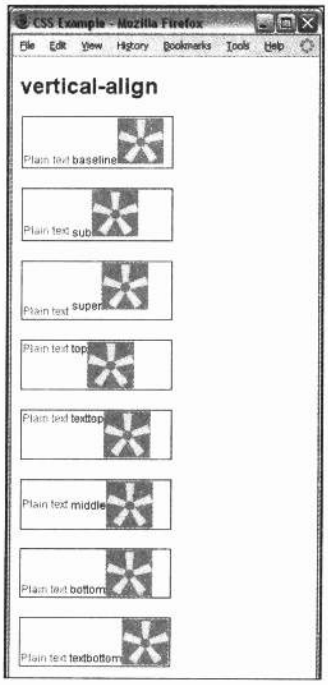


图 7-11

7.5.4 text-decoration 特性

text-decoration 特性可具有表 7-8 所示的值。

表 7-8

值	目　的
underline	在内容下方添加一条线
overline	在内容顶部添加一条线
line-through	类似于删除线文本，使用一条线穿越文本的中间。通常，该值仅用于标识文本，以表明这些文本被删除
blink	创建闪烁的文本(这通常被认为是不赞成使用或令人讨厌的方式)

例如，在下面的例子中，这些特性分别用于不同的段落：

```
p.underline {text-decoration:underline;}
p.overline {text-decoration:overline;}
p.line-through {text-decoration:line-through;}
p.blink {text-decoration:blink;}
```

图 7-12 给出了 ch07_eg10.html 文件中演示的特性的效果。注意，blink 特性仅被 Netscape 浏览器和 Firefox 浏览器支持。

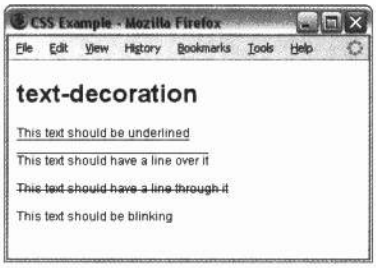


图 7-12

7.5.5 text-indent 特性

text-indent 特性可用于缩进元素内文本的第一行。例如，下面示例中第二段的第一行被缩进。下面是 ch08_eg11.html 文件中的 XHTML 代码：

```
    <p>This paragraph should be aligned with the left-hand side of the browser.
</p>
    <p class="indent">The content of this paragraph should be indented by 3 em.
</p>
```

下面是缩进第二段的规则(ch08_eg11.css)：

```
.indent {text-indent:3em;}
```

图 7-13 给出了这些代码在浏览器中的显示效果。

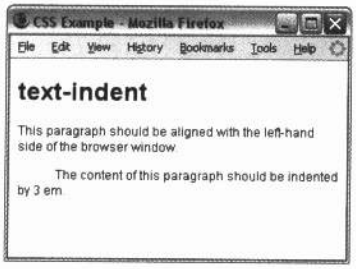


图 7-13

7.5.6　text-shadow 特性

text-shadow 特性用于创建阴影，阴影是单词的黑色版本，它位于单词的下方，并且稍微有些偏移。阴影通常用于打印媒体，它的流行性使其在 CSS2 中获得了专用的 CSS 特性。这个特性的值非常复杂，因为它采用 3 个长度，并且后跟一种颜色，但颜色是可选项：

```
.dropShadow { text-shadow: 0.3em 0.3em 0.5em black}
```

前两个长度指定阴影偏移量的 X 和 Y 坐标，而第三个长度指定模糊效果。然后紧跟一种颜色，颜色可以是颜色名称或者十六进制编码。

遗憾的是，IE 7 浏览器或 Firefox 2 浏览器不支持这个特性，但是在下载代码中的 ch07_eg12.html 文件和 ch07_eg12.css 文件中提供了一个示例。

7.5.7　text-transform 特性

text-transform 特性用于指定元素内容的大小写，它的可能值如表 7-9 所示。

表 7-9

值	目　的
none	不发生任何改变
capitalize	每个单词的第一个字母大写
uppercase	元素的整个内容都大写
lowercase	元素的整个内容都小写

请查看下面的 4 个段落，每一个段落的形式如下所示(但是 class 属性具有不同的值)：

```
    <p class="none"><i>The Catcher in the Rye</i> was written by J.D.
Salinger</p>
```

下面是 text-transform 特性的 4 种不同值的用法(ch07_eg13.css)：

```
    p.none {text-transform:none;}
    p.Capitalize {text-transform:Capitalize;}
```

```
p.UPPERCASE {text-transform:UPPERCASE;}
p.lowercase {text-transform:lowercase;}
```

图 7-14 给出了这些段落应用样式之后在浏览器中的显示效果。

图 7-14

7.5.8 letter-spacing 特性

letter-spacing 特性用于控制打印设计人员所谓的字间距：字母间的距离。松散的字间距表明字母间具有大量空间，而紧密的字间距表明字母挤压在一起。无字间距是指该字体的字母之间具有正常的距离。

该特性的可能值是 normal 或者一种长度单位(长度单位将是下一个介绍的主题)。例如(ch07_eg14.html 文件所使用的样式表 ch07_eg14.css)：

```
span.wider {letter-spacing:10px;}
```

图 7-15 给出了这个示例在浏览器中的显示效果。

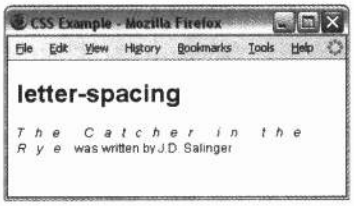


图 7-15

7.5.9 word-spacing 特性

word-spacing 特性用于设置单词间的距离，它的值应该是一种长度单位。例如(ch07_eg15.html 文件所使用的样式表 ch07_eg15.css)：

```
span.wider {word-spacing:20px;}
```

图 7-16 给出了这个示例在浏览器中的显示效果。

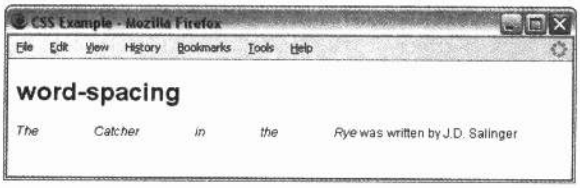


图 7-16

7.5.10　white-space 特性

white-space 特性控制是否在块级元素内或者之间保留空白。默认情况下，浏览器将两个或多个相邻空格改为单个空格，并且将任何按下的 Enter 键也改为单个空格。white-space 特性提供的效果与 XHTML 的<pre>元素和 nowrap 属性相同。该特性的可能值如表 7-10 所示。

表 7-10

值	意　义
normal	采用普通的空白折叠规则
pre	空白被保留，类似于 XHTML 的<pre>元素，但格式是为该元素指示的格式，而不只是一种等宽字体
nowrap	仅在显式地提供 元素后，文本才会换行，否则不会换行

例如，可以按照如下方式使用 white-space 特性(ch07_eg16.css)：

```
.pre {white-space:pre;}
.nowrap {white-space:nowrap;}
```

但是，IE 7 浏览器不支持 pre 值，但是 Netscape 4/Firefox 1 浏览器或更新的浏览器支持它。IE 6 浏览器和 Netscape 4/Firefox 1 浏览器以及更新的浏览器支持 nowrap 特性，图 7-17 给出特性的效果。

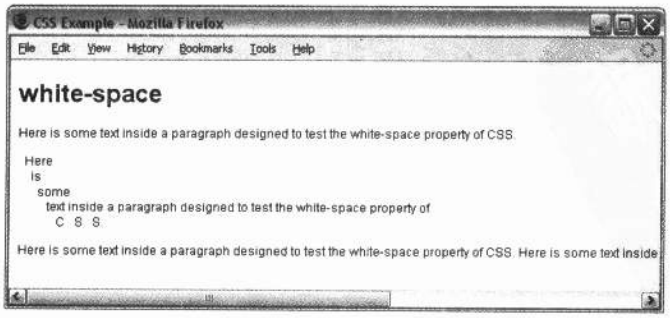


图 7-17

7.5.11 direction 特性

direction 特性非常类似于 dir 属性，它用于指定文本流动的方向。表 7-11 给出了该属性的可能值。

表 7-11

值	意　义
Ltr	文本的方向从左到右
rtl	文本的方向从右到左
inherit	文本的方向与它的父元素相同

例如，下面的规则分别作用于两个不同的段落，指示不同的文本方向(ch07_eg17.html 文件所使用的样式表 ch07_eg17.css)：

```
p.ltr {direction:ltr;}
p.rtl {direction:rtl;}
```

实际上，在 IE 浏览器和 Firefox 浏览器中使用这个特性之后，得到的效果类似于使用 align 属性。值 rtl 将简单地右对齐文本，如图 7-18 所示。但是需要注意，对于段落中方向为从右到左的句子，句点将显示在该句子的左边。

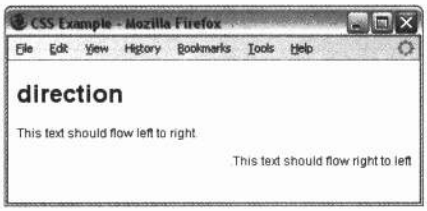


图 7-18

注意：

不要使用这个特性取代 text-align 特性，因为新的浏览器完全支持这个特性还需要一段时间。

7.5.12 unicode-bidi 特性

设计 unicode-bidi 特性是为了国际化目的；名称中的 bidi 是 bi-directional(双向)的缩写。该特性允许单词的显示方向参考 Unicode 标准，并且作者能够将元素内容的方向改为与 Unicode 标准相反。表 7-12 给出了该特性的可能值。

表 7-12

值	目　的
Normal	不允许任何方向的嵌入
embed	元素启用额外的嵌入级别，并且将遵循计划的 Unicode 方向

(续表)

值	目　的
bidi-override	重写内联元素的默认方向值，以便 direction 特性能够用于设置该元素内的方向(重写 Unicode 设置)

当需要内联元素的方向与包含元素的剩余部分不同时，该特性特别有用——例如，可以编写具有不同方向的单词，因为 embed 值允许文本遵循与包含元素剩余部分相反的方向。如果希望阻止这种情况发生，可以使用 bidi-override 值。

7.6　文本伪类

在学习文本相关操作时，存在两种非常有用的伪类，它们可以帮助处理文本。这两种伪类可用于将元素的第一个字母或第一行以与该元素剩余部分不同的方式进行显示。通常在布局文本时使用这些伪类。

7.6.1　first-letter 伪类

first-letter 伪类可用于仅为元素的第一个字母指定规则。该伪类通常用于杂志文章或书籍中的新页面的第一个字符。

在下面的示例中，first-letter 伪类应用于一个<p>元素，该元素附带 class 属性，并且该属性的值为 pageOne(ch07_eg18.html 使用的样式表 ch07_eg18.css)：

```
p.pageOne:first-letter {font-size:42px;}
```

在图 7-19 中可以看到这个 first-letter 伪类的效果(在该图中也给出了下一个即将介绍的伪类示例)。

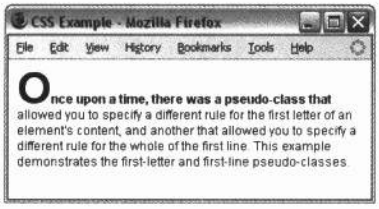


图 7-19

7.6.2　first-line 伪类

first-line 伪类能够将段落的第一行以与该段落剩余部分不同的方式进行显示。通常第一行以粗体字体显示，以便读者能够清晰看到(文章的)简介或者(诗歌的)第一行。

伪类的名称与它所作用的元素之间以冒号隔开：

```
p:first-line {font-weight:bold;}
```

但是需要注意，如果窗口大小发生变化，使第一行具有较少的文本，则浏览器中仅有第一行将被赋予这种新样式。可以在图 7-19 中看到 first-line 伪类的效果，该图中也演示了 first-letter 伪类的效果。

试一试　字体测试页面

现在您已经了解如何使用 CSS 来格式化文本，接下来将所学到的内容用于实践，因此创建一个字体测试页面。可以利用这个页面测试浏览器是否支持某种字体。

(1) 创建一个新的 XHTML 文档，该文档具有到目前为止习惯于创建的程序框架：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Font test</title>
</head>
<body>
</body>
</html>
```

(2) 在外部样式表中添加一个<link />元素。样式表的名称为 font-test.css。

```
<head>
   <title>Font text</title>
   <link rel="stylesheet" type="text/css" href="font-test.css" />
</head>
```

(3) 在文档主体中添加 4 个 元素，每一个元素都包含文本行“The quick brown fox jumped over the lazy dog.”。

对于每一个元素，赋予其一个 class 属性，该属性的值为不同字型的名称。并且每一个句子以字型的名称开头，如下所示：

```
<div class="arial">Arial: The quick brown fox jumped over the lazy dog.</div>

<div class="helvetica">Helvetica: The quick brown fox jumped over the lazy
dog.
</div>

<div class="TimesNewRoman">Times New Roman: The quick brown fox jumped over
the
lazy dog.</div>

<div class="MrsEaves">Mrs Eaves: The quick brown fox jumped over the lazy
dog.
</div>
```

(4) 将这个文件保存为 font-test.html。

(5) 在使用的编辑器中创建一个新的文档，并将其保存为 font-test.css 文件。

(6) 为添加到 XHTML 文档中的每一个 元素添加一个选择器：

```
div.arial
div.helvetica
div.TimesNewRoman
div.MrsEaves
```

(7) 为每个选择器添加一个 font-family 特性，并且对该特性赋予指定的字型值：

```
div.arial {font-family:arial;}
div.helvetica {font-family:Helvetica;}
div.TimesNewRoman {font-family:"Times New Roman";}
div.MrsEaves {font-family:"Mrs Eaves";}
```

(8) 在希望查看的字型后面添加另外一种字型，并使用逗号将两者隔开。注意，第二种字型应当与希望查看的字型差别较大。这里使用 Courier(一种等宽字体)作为第二种选择，从而能够清晰地了解浏览器是否支持前面指定的第一种字体。

```
div.arial {font-family:arial, courier;}
div.helvetica {font-family:Helvetica, courier;}
div.TimesNewRoman {font-family:"Times New Roman", courier;}
div.MrsEaves {font-family:"Mrs Eaves", courier;}
```

(9) 添加下面的规则，以确保每一行之间有充足的空格，从而方便查看这些字体：

```
div {line-height:28px;}
```

(10) 保存这个 CSS 文件，并在浏览器中打开 XHTML 页面。该示例的效果将如图 7-20 所示。

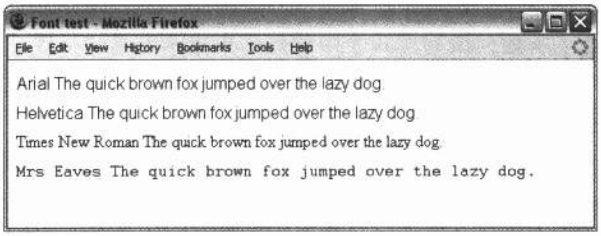


图 7-20

Mrs Eaves 字型是一种 serif 字体，类似于 Times。根据图 7-20 所示，显示该屏幕截图的计算机没有安装 Mrs Eaves 字型，因为浏览器中显示的是 Courier 字体——一种等宽字体。

本章末尾的一个练习题将扩展这个示例。

工作原理

关于这个示例，需要注意的第一个方面是 XHTML 源文档中使用的<link />元素，它指明该文件将被 font-test.css 样式表赋予样式。

```
<link rel="stylesheet" type="text/css" href="font-test.css" />
```

该行代码展示了 3 个属性，对于指示包含链接的文档和链接到的文档之间的关系以定位样式表，它们都是必须具有的属性。

现在浏览器将使用 font-test.css 文件中指定的样式表来布局这个示例。该 XHTML 文档中的每一个 元素附带一个 class 属性，CSS 使用它来识别特定元素的内容，并赋予其与其他 元素不同的样式。class 属性的值是将被检查的字型。

CSS 规则中的选择器确定每一条规则所应用的元素。此示例中，在样式表中使用类选择器独立标识每一个 元素，以便不同的规则可以应用到每一个元素。例如，采用如下方式标识以 Arial 字型显示的文本：

```
div.arial
```

然后在选择器后面的大括号中添加一些特性。font-family 特性用于为所选择的元素(以及它们的子元素——因为这个特性将被子元素所继承)指定希望使用的字型。然后指定第二种在外观上差别较大的字体，以便浏览器在无法找到第一种字体时使用它；这将能够清楚地表明浏览器是否支持该字体。这里使用 Courier 字体，因为它是一种等宽字体，能够明显地与其他字体区分。

```
div.arial {font-family:arial,courier;}
```

最后，line-height 特性在每一行文本之间添加额外的高度，以便使该属性示例具有更好的可读性。在额外的用于每一个 元素的选择器中指定该特性，而不是在每一个 元素中都重复指定它。

7.7 选择器

现在您应当已经了解如何编写样式表中的规则来指示元素的外观，在查看更多可用于影响文档布局的特性之前，需要掌握更多的一些基础知识，首先学习一些用于确定将某条规则应用到哪个元素或哪些元素的不同方式。

可以使用多种方式选择元素，而不只是使用本章前面元素名的方式(该方式称为简单选择器)或者使用样式表所赋予样式的文档中的类属性值。可以创建一些更特殊的选择器。除了提供元素的名称作为选择器之外，可以使用下面的一些选择器。

7.7.1 通用选择器

通用选择器是一个星号；它类似于通配符，并且匹配文档中的所有元素类型。

```
*{}
```

如果希望让一条规则应用于所有的元素，可以使用这种选择器。有时，该选择器用于一些默认特性，例如 font-family 特性和 font-size 特性，这些特性将作用于整个文档(除非其他更特定的选择器指示元素的这些特性应当使用不同值)。

通用选择器与将默认样式应用于<body>元素稍微有些区别，它作用于所有元素，并且不依赖于从应用于<body>元素的规则继承而来的特性。

7.7.2　类型选择器

类型选择器匹配以逗号隔开的元素列表中的所有元素，利用它可以将一些相同的规则作用于多个元素。例如，下面的代码将匹配所有的 h1、h2 和 p 元素。

```
h1, h2, p {}
```

如果将一些相同的规则应用于多个元素，则这种技术将使得样式表的规模较小，从而节省服务器(服务器是将 Web 页面发送到请求者处的计算机)的带宽和负载。

7.7.3　类选择器

类选择器可用于将一条规则与附带 class 属性的元素匹配，其中该 class 属性的值为类选择器中指定的值。例如，假设存在一个具有 class 属性的<p>元素，其中 class 属性的值为 BackgroundNote，如下所示：

```
<p class="BackgroundNote">This paragraph contains an aside.</p>
```

可以以如下两种方式之一使用类选择器：首先，可以简单地将一条规则赋予任何具有 class 属性并且属性值为 BackgroundNote 的元素，如下所示，只需要在 class 属性的值之前放置一个句点：

```
.BackgroundNote {}
```

或者可以采用如下方式：创建一个仅选择附带 class 属性的<p>元素的选择器，其中 class 属性的值是 BackgroundNote：

```
p.BackgroundNote {}
```

如果具有多个能够附带 class 属性且属性值相同的元素(例如一个<p>元素和一个 元素都可以使用具有相同值的 class 属性)，并且希望这些元素的内容以相同的方式显示，则可以使用前一种方式。如果希望样式仅作用于<p>元素，则应当使用后一种方式。

7.7.4　id 选择器

id 选择器的作用方式类似于类选择器，但它仅作用于 id 属性的值。但是，在 id 属性值之前不是句点，而是使用英镑符号(#)。因此，id 属性值为 abstract 的<p>元素可以通过如下选择器识别：

```
p#abstract
```

因为在一个文档内 id 属性的值应该是唯一的，所以这个选择器将仅应用于一个元素的内容。

7.7.5 子选择器

子选择器匹配的元素是另外一个元素的直接子元素。下面的选择器匹配任何作为<td>元素的直接子元素的<b>元素：

```
td>b {}
```

该选择器可用于为任何是<td>元素的直接子元素的<b>元素指定不同的样式，而不是为文档中其他位置出现的<b>元素指定不同的样式。

请注意，这个选择器仅作用于作为该父元素的直接子元素的<b>元素。下面的选择器没有什么实际意义，因为<b>元素不可能是<table>元素的直接子元素(相反，<tr>元素更可能是<table>元素的直接子元素)：

```
table>b {}
```

其中小于号(>)称为连结符。

遗憾的是，IE7 浏览器是第一种支持子选择器的 Internet Explorer 版本，因此在您的站点中使用这个选择器之前，需要先调查有多少站点访问者使用 IE7 浏览器(第 13 章中将介绍如何完成该任务)。如果大量访问者仍然使用 IE6 浏览器，则应当在 IE6 中测试 Web 站点，以确保显示的外观是想要的效果。

7.7.6 后继选择器

后继选择器所匹配的元素类型是另一个指定元素的后继，该元素可位于任何嵌套级别，而不只是指定元素的直接子元素。虽然小于号是子选择器的连结符，但对于后继选择器来说，连结符是空格。查看如下示例：

```
table b {}
```

在这个示例中，选择器匹配任何作为<table>元素的子元素的<b>元素，这意味着它将应用于<td>元素和<th>元素中的<b>元素。

后继选择器与子选择器的最大区别在于，它应用于<table>元素的所有子元素，而不只是直接子元素。

7.7.7 相邻兄弟选择器

相邻兄弟选择器匹配指定元素的相邻兄弟元素。例如，如果希望为任何 1 级题头后的第一个段落指定不同的样式，可以使用相邻兄弟选择器，如下所示：

```
h1+p {}
```

其中两个元素必须具有相同的元素，并且这将仅作用于题头之后的直接<p>元素。

遗憾的是，IE7 浏览器是第一种支持相邻兄弟选择器的 Internet Explorer 版本，因此在您的站点中使用这个选择器之前，需要调查有多少站点访问者使用 IE7 浏览器(第 13 章中将介绍如何完成该任务)。如果大量访问者仍然使用 IE6 浏览器，则应当在 IE6 中测试 Web

站点，以确保显示的外观是想要的效果。

7.7.8　利用子选择器和相邻兄弟选择器降低标记中类的相关性

子选择器和相邻兄弟选择器都非常重要，因为它们能够减少需要添加到 XHTML 文档中的 class 属性的数量。

因为各种可能性，很容易在文档中添加一些 class 属性。例如，可能希望让<h1>元素之后的第一个段落以粗体显示，并且可能已经在每一个<h1>元素之后的第一个<p>元素中添加一个 class 属性。此时的代码可以正常运行，但是在您意识到问题之前，您的标记中可能已经充斥各种类，这些类仅是为了更容易地控制页面的外观。

此外，如果接下来希望让每一个<h1>元素之后的前两个<p>元素以粗体显示，则将不得不返回并且为每一个<h1>元素之后的第二个<p>元素添加一个新的 class 属性。因此，子选择器和相邻兄弟选择器将为赋予文档样式带来很多灵活性，并且使标记看上去更加清晰。

请查看下面的 XHTML 内容(ch07_eg19.html)：

```
<p>Here is an example of some adjacent sibling and child selectors.</p>
<div>
  <p>One</p>
  <p>Two</p>
  <p>Three</p>
  <p>Four</p>
  <p>Five</p>
</div>
```

通过仅使用相邻兄弟选择器和子选择器，创建的页面如图 7-21 所示。

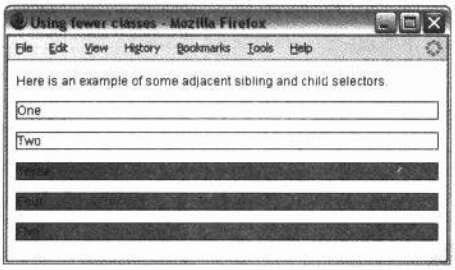


图 7-21

该示例中存在 3 种不同的段落样式：

- 第一个段落没有边框或背景色。
- 元素中的所有段落都有边框。
- 最后 3 个段落具有灰色背景。

这里没有使用 3 种不同的类来指定不同的段落样式，而是使用一条规则来控制所有段落中使用的字体：

```
p {font-family:arial, verdana, sans-serif;}
```

下面是第二条规则，它应用于任何是 元素的子元素的段落(因为第一个段落不在 元素内，所以该规则不应用于第一个段落)。

```
div>p {border:1px solid #000000;}
```

第三条规则匹配任何第三个连续<p>元素的段落(因为第四个和第五个<p>元素前面有两个<p>元素，所以这条规则应用于它们)。

```
p+p+p {background-color:#999999;}
```

请记住，这个示例在 IE6 或更早期版本的 IE 浏览器中无法正确工作，因为在 IE7 浏览器中首次引入这些选择器。

7.7.9 属性选择器

属性选择器能够将元素附带的属性用于选择器中。可以以多种方式使用属性选择器，如表 7-13 所示，但仅有一些最新版本的浏览器才支持它们。

表 7-13

名　称	示　例	匹　配
存在选择器	p[id]	任何附带 id 属性的<p>元素
相等选择器	p[id="summary"]	任何附带 id 属性的<p>元素，其中 id 属性的值是 summary
空格选择器	p[class~="XHTML"]	任何附带 class 属性的<p>元素，其中 class 属性的值为以空格隔开的单词列表，XHTML 也是单词列表中的一个单词
连字符选择器	p[language|="en"]	任何附带 language 属性的<p>元素，其中 language 属性的值以 en 开头，后面紧跟一个连字符(连字符用于语言属性)
前缀选择器(CSS3)	p[attr^"b"]	附带值以 b 开头的任何属性的<p>元素(CSS3)
子串选择器(CSS3)	p[attr*"on"]	附带值包含字符 on 的任何属性的<p>元素(CSS3)
后缀选择器(CSS3)	p[attr$"x"]	附带值以字母 x 结尾的任何属性的<p>元素(CSS3)

Internet Explorer 在 IE7 浏览器中实现了这些属性选择器，为了使这些属性选择器能够正常工作，XHTML 文档必须具有严格的!DOCTYPE 声明。

Firefox 2 浏览器仅支持前 4 个属性选择器。

注意：

另外还可以在选择器中使用正则表达式。但是，当前的主流浏览器不支持在选择器中使用正则表达式。此外，正则表达式是复杂的主题，在考虑学习它之前，最好使用这里给出的选择器。

下面查看这些属性选择器的用法。下面的代码中有 7 个不同的段落元素，每一个附带不同的属性/属性值(ch07_eg20.html)：

```
    <p id="introduction">Here's paragraph one, each have different
attributes.</p>
    <p id="summary">Here's paragraph two, each have different attributes.</p>
    <p class="important XHTML">Here's paragraph three, each have different
    attributes.</p>
    <p language="en-us">Here's paragraph four, each have different
attributes.</p>
    <p class="begins">Here's paragraph five, each have different
attributes.</p>
    <p class="contains">Here's paragraph six, each have different
attributes.</p>
    <p class="suffix">Here's paragraph seven, each have different
attributes.</p>
```

下面查看一个 CSS 样式表，它使用一些属性选择器将不同的样式规则与上面的每一个元素相关联(ch07_eg20.css)：

```
p[id] {border:1px solid #000000;}
p[id="summary"] {background-color:#999999;}
p[class~="XHTML"] {border:3px solid #000000;}
p[language|="en"] {color:#ffffff; background-color:#000000;}
p[attr^"b"]{border:3px solid #333333;}
p[attr*"on"] {color:#ffffff; background-color:#333333;}
p[attr$"x"] {border:1px solid #333333;}
```

在图 7-22 中给出了该示例在 Firefox 2.0 浏览器中的结果。其中这个版本的 Firefox 浏览器仅支持前 4 个属性选择器，而不支持后 3 个(后 3 个是 CSS3 中新添加的属性选择器)。

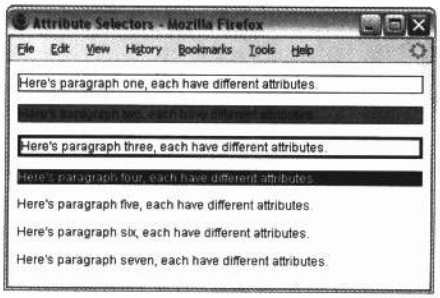


图 7-22

因为 XHTML 是区分大小写的，所以所有的选择器都应该匹配相应元素名的大小写。

7.8 长度

您已经看到某些特性的值以长度的形式给出(字体大小、文本行的高度)，并且将遇到很多 CSS 特性都需要指定长度作为值。现在介绍这方面的内容，因为下一节中介绍的几个特性需要使用长度作为值。

在 CSS 中，长度可以通过以下 3 种方式之一度量：

- 绝对单位
- 相对单位
- 百分比

7.8.1 绝对单位

表 7-14 给出了一些可以在 CSS 中使用的绝对单位。

表 7-14

单　位	全　名
pt	一点
pc	十二点
in	一英寸
cm	一厘米
mm	一毫米

对于英寸、厘米、毫米，不需要额外说明它们，但是另外两个单位非常有意思。一点是一英寸的 1/72(等同于大多数计算机屏幕分辨率中的一个像素)，十二点是一英寸的 1/12(即等于 12 点)。印刷工人趋向于使用点来度量字体大小和行距(文本行之间的距离)，而使用十二点度量文本行的长度。

7.8.2 相对单位

相对单位和百分比非常有用，但是它们也带来一些问题，需要了解这些问题，原因包括两个方面：

- 随着显示文档所使用的媒体种类的变化，它们可以调整大小。
- 在 Web 浏览器中，用户可以增加或减少字体的大小，并且页面的剩余部分将会缩放以适合当前字体大小。

1. px

像素是屏幕上最小的分辨率单位，并且是 CSS 中指定字体大小和长度的最常用方式。

从技术上来说，使用像素作为度量单位的布局的大小可以取决于正在使用的媒介(继续学习后面的内容，以理解这里为什么使用“可以”一词)。

大多数计算机屏幕的分辨率为 72 点每英寸(dpi)，但是可以发现最先进的激光和喷墨打印机都具有更高的分辨率——我当前使用的打印机的分辨率是 300dpi。相反，移动电话和 PDA 具有比计算机屏幕更低的分辨率。

因此，一个 500 像素宽的表在 72dpi 屏幕上的宽度是 9.9444 英寸，在 300dpi 屏幕上的宽度是 1.666 英寸，在 32dpi 屏幕上的宽度是 13.888 英寸(具有如此低分辨率的屏幕非常有可能达不到该宽度)。

但是，当从 IE 浏览器或 Firefox 浏览器中打印一个 Web 页面时，浏览器将会调整像素以给出可读的文档版本。事实上，CSS 推荐用户代理重新调整像素单位，从而在距离一臂长度阅读时，1 像素对应于大约 0.28 毫米或 1/90 英寸。但是从技术上来说，这就使得像素不再是一种相对单位，而是一种绝对单位。

注意：

大多数功能强大的编程语言都具有一个函数，程序员利用该函数可以根据屏幕分辨率调整图像，但在 CSS 中不能实现该操作。

2. em

em 单位直接对应于参考元素的字体大小，参考元素可以是该元素或包含它的元素。

术语 em 通常被认为来源于小写字母 m 的宽度，但现在通常认为是字体的高度(注意，en 是 em 的一半)。

3. ex

ex 是小写字母 *x* 的高度。因为不同的字体具有不同的比例，所以 ex 相对于字体的大小以及字体的类型。在图 7-23 中，可以看到 Courier 字型中的 *x* 小于 Impact 字型中的 *x*。

图 7-23

7.8.3　百分比

百分比给出相对于另外一个值的某个值(具体的值取决于正在讨论的特性)。注意，当一个百分比值被继承时，它是由继承的百分比(而不是百分比本身)设置的值。

7.9　框模型简介

现在您已经掌握如何指定特性，学习了很多选择器，并且了解一些基本长度单位。接下来，本章将介绍更多特性集，可以使用这些特性控制元素内容的表示。但是在此之前，

需要理解 CSS 如何基于框模型。

在 CSS 中，每一个元素被视为一个框。请记住，这将有助于理解如何利用 CSS 创建有吸引力的布局。

如表 7-15 所示，每一个框具有 3 个特性，必须了解这些特性。

表 7-15

特　性	说　明
border	每一个框都具有一个边框，即使不能看到该边框。这使得框的边缘与其他框分开
margin	页边空白是一个框的边缘与和相邻框之间的距离
padding	内边距是框内容和它的边框之间的空间

图 7-24 有助于更好地理解这些特性，该图给出了框的各个部分(其中黑线是边框)。

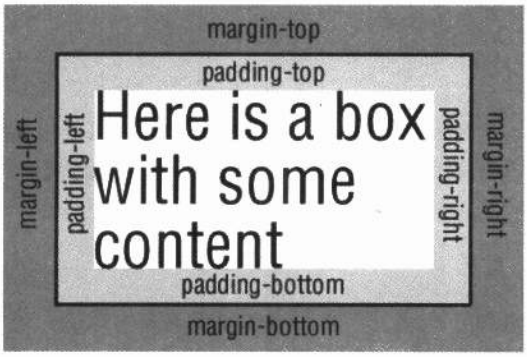


图 7-24

可以使用 CSS 分别控制每个框的顶部、底部、左边和右边边框以及页边空白和内边距；并且可以为框的每一边指定不同的宽度和颜色。

在设计中创建空白时 padding 特性和 margin 特性非常重要，空白是页面各部分之间的空格。例如，如果在一个具有边框的框中放置文本，则会希望存在一些内边距，以便文本的边缘不接触边框(如果文本实际地接触相同颜色的边框，则阅读起来非常困难)。

同时，如果存在两个具有边框的框，并且在它们之间没有页边空白，则这两个框将接触在一起，并且框之间接触的线看上去将较粗。

但是，对于页边空白存在一种有趣的现象：当一个元素的底部页边空白接触另一个元素的顶部页边空白，只会显示其中范围较大的页边空白(如果它们的大小相等，则最终的页边空白将等于其中一个页边空白的大小)。图 7-25 给出了两个相邻框的垂直页边空白的折叠效果。

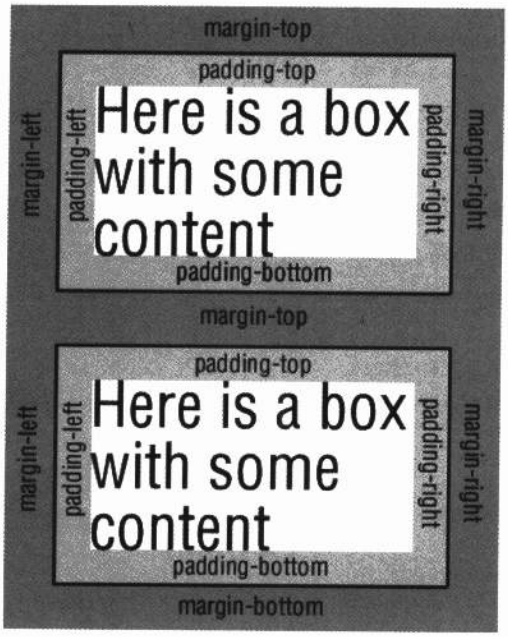


图 7-25

为了真正地理解框模型如何作用于元素，请查看下一节中的示例。

7.9.1　演示框模型的示例

本节利用真实页面演示框模型，该页面看上去如同<body>元素创建包含整个页面的框，然后每个题头、段落、图像或者链接创建页面内的其他框。

每个框具有一些不同的特性，它们影响框中内容的外观。请查看下面的 XHTML 文档 (ch09_eg21.html)：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Understanding the Box Model</title>
  <link rel="stylesheet" type="text/css" href="ch07_eg19.css" />
</head>
<body>
  <h1>Thinking Inside the Box</h1>
  <p class="description">When you are styling a web page with CSS page you
  must start to think in terms of <b>boxes</b>.</p>
  <p>Each element is treated as if it generates a new box. Each box can have
  new rules associated with it.</p>
  <img src="images/boxmodel.gif" alt="How CSS treats a box" />
  <p>As you can see from the diagram above, each box has a <b>border</b>.
```

```
  Between the content and the border you can have <b>padding</b>, and
  outside of the border you can have a <b>margin</b> to separate this box
  from any neighboring boxes.</p>
</body>
</html>
```

文档主体中包含的每一个元素——<body>、<h2>、<p>、<img>和<b>——都视为如同位于独立的框中。通过创建一些 CSS 规则，使用本章后面将会介绍的一些新特性在每一个元素的周围添加边框，以此来表明每一个元素位于独立的框中(ch07_eg21.css)：

```
body {
   color:#000000;
   background-color:#ffffff;
   font-family:arial, verdana, sans-serif;
   font-size:12px;
   line-height:24px;}
body, h1, p, img, b {
   border-style:solid;
   border-width:2px;
   border-color:#000000;
   padding:2px;}
h1, b {background-color:#cccccc;}
```

这个示例将帮助您更好地理解 CSS 样式如何选择一个元素，然后如何利用适当的值设置不同的特性。

图 7-26 给出了这个页面在浏览器中的显示效果。虽然该页面不够吸引人，但它演示了如何为每一个元素创建框。图中的线实际上是为元素创建的框的边框。除了每一个元素都具有边框之外，<h1>元素和 <bold>元素还具有灰色背景以便区分它们。

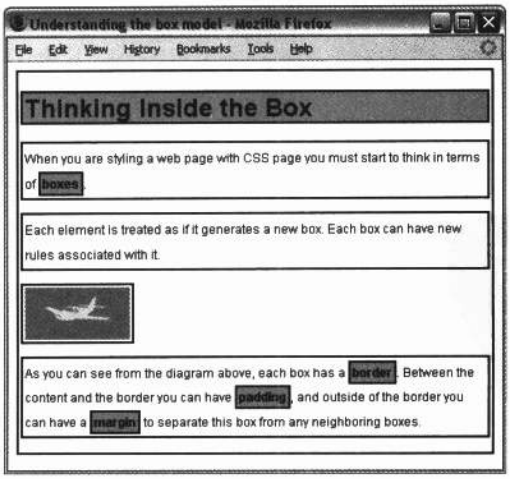


图 7-26

第 1 章中提及，在块级元素和内联元素之间存在区别；在操作 CSS 时，这种区别非常重要，因为它确定每个框的处理方式。这个示例也很好地演示了这一点：如果仔细观察<h1>元素，会发现它的框占用整个浏览器宽度，相反<b>元素周围的框位于段落剩余部分的中间，而不是占用整行。

<h1>元素是块级元素，而<body>元素和<p>元素也是块级元素。<hl>元素被视为如同它自己创建一个独立的块，并且出现在自己的新行中。<b>元素是一个内联元素，跟在包含它的元素之后显示，而不会出现在自己的新行中。默认情况下，块级元素也将占用页面(或者包含它的元素)的整个宽度，而内联元素将仅占用它需要的空间。

<img />元素看上去类似于块级元素，但它实际上是内联元素。这是因为，虽然该元素看上去好像位于自己的行中，但它的边框仅占用图像的宽度。如果它是块级元素，则边框将占用整个浏览器宽度。该图像位于自己的行中，仅是因为它两边的元素都是块级元素(因而周围的元素出现在自己的行中)。

在 Strict XHTML 中，这种图像元素必须放置在块级元素中，因为<body>元素内肯定有作为其子元素的块级元素。但是，在 Transitional XHTML 中这并不重要，可以通过将元素放置在 元素内来简单地修正这个问题(您可能记得 元素是分组元素)。

注意：

如果是在 Internet Explorer 6 或更早的 IE 浏览器版本中检查站点，可能会发现框的大小与期望的不同。在与标准兼容的浏览器中，框的总宽度为：

```
width = margin-left + border-left + padding-left + width + padding-right
+ border-right + margin-right
```

在较老的 IE 浏览器版本中，内边距和页边空白都不包含在计算之内，因此公式为：

```
width = margin-left + width + margin-right
```

为了克服 IE6 浏览器中的这个问题，可以确保使其运行在标准兼容模式下，方式是在 XHTML 页面中包含一个!DOCTYPE 声明，在第 1 章中创建的 XHTML 页面中就具有该声明。

7.9.2　Border 特性

border 特性可用于指定代表元素的框的边框外观。可以改变边框的 3 种特性：

- border-color，指示边框的颜色
- border-style，指示边框应该是实线、虚线或双线，或者是其他的可能值之一。
- border-width，指示边框的宽度

1. border-color 特性

border-color 特性可用于改变框周围的边框的颜色。例如：

```
p {border-color:#ff0000;}
```

该特性的值可以是十六进制的颜色编码或者颜色名(附录 D 中详细地讨论了颜色方面的内容)，也可以表达为红、绿、蓝的值(0 到 255 之间的值)或者红、绿、蓝的百分比。表 7-16 中给出了一些示例。

表 7-16

颜 色 名	十六进制编码	RGB 值	RGB 百分比
red	#ff0000	rgb (255, 0, 0)	rgb (100%, 0, 0)
green	#00ff00	rgb (0, 255, 0)	rgb (0, 100%, 0)
blue	#0000ff	rgb (0, 0, 255)	rgb (0, 0, 100%)

可以使用下面的特性单独改变框的边框底部、左侧、顶部和右侧的颜色。

- border-bottom-color
- border-right-color
- border-top-color
- border-left-color

2. border-style 特性

border-style 特性可用于指定边框的样式：

```
p {border-style:solid;}
```

这个特性的默认值是 none，因此自动不显示边框。表 7-17 给出了该特性的一些可能值。

表 7-17

值	描 述
none	没有边框(等价于 border-width:0;)
solid	边框由单条实线组成
dotted	边框由一系列的点组成
dashed	边框由一系列的短线组成
double	边框由两条实线组成；border-width 特性的值创建两条线的总宽度以及它们之间的空间
groove	边框看上去如同雕刻进页面中
ridge	边框的外观与值为 groove 时相反
inset	边框使框看上去如同嵌入页面
outset	边框使框看上去如同位于画布之外
hidden	与 none 相同，但是对于表元素存在边框冲突解决方法

图 7-27 给出了这些值的示例在浏览器中的显示效果(来源于 ch07_eg22.html)。注意，尽管图 7-27 中的最后 4 个示例看上去非常相似，但是实际上它们并不相同，您可以亲自利用这个示例的下载代码测试它们。

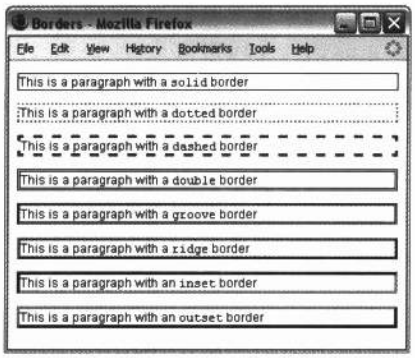


图 7-27

可以使用下面的特性单独改变框的边框底部、左侧、顶部和右侧的样式：

- border-bottom-style
- border-right-style
- border-top-style
- border-left-style

3. border-width 特性

border-width 特性可用于设置边框的宽度。

```
p {border-style:solid;}
   border-width:4px;}
```

border-width 特性的值不能是百分比，而必须是一个长度(本章前面的“长度”一节中介绍了长度)或者下面的值之一：

- thin
- medium
- thick

CSS 规范中并没有指定 thin、medium 和 thick 值的具体宽度(以像素为单位)。因此对应于这些关键字的实际宽度取决于特定的浏览器。

可以使用下面的特性单独改变框的边框底部、左侧、顶部和右侧的宽度：

- border-bottom-width
- border-right-width
- border-top-width
- border-left-width

4. 利用简写表达边框的特性

border 特性允许在一个特性中同时指定线的颜色、样式和宽度：

```
p {border: 4px solid red;}
```

如果使用了这种简写，则值之间不能有任何内容(除了空格之外)。也可以使用下面的特性单独为框中每一边的线指定颜色、样式和宽度：

- border-bottom
- border-top
- border-left
- border-right

7.9.3 padding 特性

padding 特性可用于指定元素的内容和它的边框之间的空间量：

```
td {padding:5px;}
```

这个属性的值可以是长度、百分比或者单词 inherit。如果值是 inherit，则元素的内边距与其父元素相同。

如果使用的是百分比，则百分比相对于包含该元素的框。因此，如果规则指示<body>元素上的内边距是 10%、浏览器窗口宽度的 5%将作为<body>元素内容内的每一边的内边距。也就是说，如果规则指示一个位于 100 像素正方形的单元格内的<td>元素具有 10%的内边距，则边框内的正方形的每一边周围将具有 5 个像素的内边距。

元素的内边距不会继承，因此如果<body>元素具有值为 50 像素的 padding 特性，则它不会自动应用于<body>元素内的所有其他元素。

可以使用下面的特性为框的每一边指定不同的内边距量：

- padding-bottom
- padding-top
- padding-left
- padding-right

在创建元素的内容和它的边框之间的空白时，padding 属性非常有用(即使边框不可见，内边距也可以阻止两个相邻框的内容相互接触)。查看图 7-28 中的两个段落。

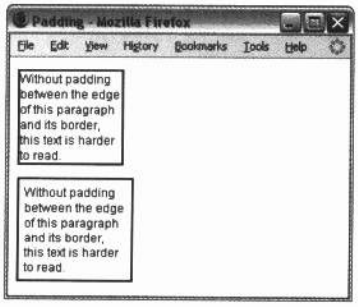


图 7-28

如果查看用于这两个段落元素的 CSS 规则，可以发现默认情况下第一个段落没有内边距；如果希望两个段落之间存在间隙，则必须指定内边距(ch07_eg23.css)。

```
.a, .b {border-style:solid;
  border-color:#000000;
  border-width:2px;
  width:100px;}
.b {padding:5px;}
```

当一个表具有大量的相邻单元格时，padding 特性就非常有用。

7.9.4　margin 特性

margin 特性是框之间的距离，它的值可以是长度、百分比或者 inherit，这些值的意义与 padding 特性中的值相同。

```
p {margin:20px;}
```

与 padding 特性一样，margin 特性的值不会被子元素继承。但是请记住，两个框的垂直页边空白接触时，将导致相互折叠，因此块之间的距离不是两个页边空白的总和，而是其中的较大者(如果两个页边空白相等，则可以是其中的任何一个页边空白)。

也可以使用下面的特性为框中每一边上的页边空白设置不同的值：

- margin-bottom
- margin-top
- margin-left
- margin-right

如果查看下面的示例(如图 7-29 所示，该示例来源于 ch07_eg24.html)，可以看到 3 个段落，它们看上去好像以相等的距离隔开。但是，这些段落的顶部页边空白比底部页边空白高，因此当两个框接触时，底部页边空白将被忽略(页边空白被折叠)。该示例还演示了如何设置内联元素边上的左右页边空白——如图中高亮显示的单词所示。再次提醒，这不是最吸引人的示例，但它演示了块和内联框的页边空白用法。

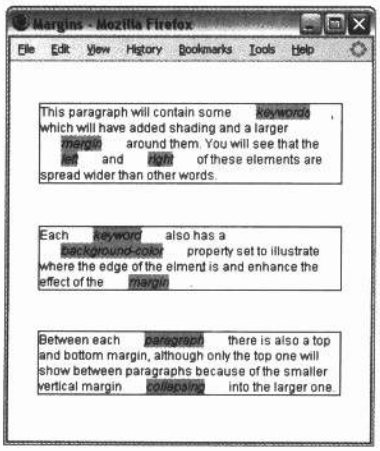


图 7-29

段落中使用元素强调的单词具有 margin-left 特性和 margin-right 特性设置。因为这些元素还具有背景色设置，所以可以实际地看到左右两边的页边空白如何将该单词与周围的单词隔开。

下面是 ch07_eg24.css 中的规则：

```
body {
    color:#000000;
    background-color:#ffffff;
    font-family:arial, verdana, sans-serif;
    font-size:12px;}
p {
    margin-top:40px;
    margin-bottom:30px;
    margin-left:20px;
    margin-right:20px;
    border-style:solid;
    border-width:1px;
    border-color:#000000;}
em {
    background-color:#cccccc;
    margin-left:20px;
    margin-right:20px;}
```

7.9.5 面积

现在您已经看到在每个框的周围有边框，在每个框的内部有内边距，而环绕这些对象的则是页边空白。下面将介绍如何改变框的面积。

表 7-18 中给出的特性可用于控制框的面积。

表 7-18

特　性	目　的
height	设置框的高度
width	设置框的宽度
line-height	设置文本行的高度(类似于布局程序中的行距设置)
max-height	设置框的最大高度
min-height	设置框的最小高度
max-width	设置框的最大宽度
min-width	设置框的最小宽度

1. height 特性和 width 特性

height 特性和 width 特性可用于设置框的高度和宽度，它们能够采用的值包括长度、百分比或者关键字 auto(默认值是 auto)。

下面可以看到两个段落元素的 CSS 规则。第一条规则具有一个 class 属性，它的值是 one，第二条规则的 class 属性的值是 two(ch07_eg25.css)：

```
p.one {
  width:200px; height:100px;
  padding:5px; margin:10px;
  border-style:solid; border-color:#000000; border-width:2px;}
p.two {
  width:300px; height:100px;
  padding:5px; margin:10px;
  border-style:solid; border-color:#000000; border-width:2px;}
```

在图 7-30 中可以看到，第一个段落将是 200 像素宽和 100 像素高，而第二个段落将是 300 像素宽和 100 像素高。

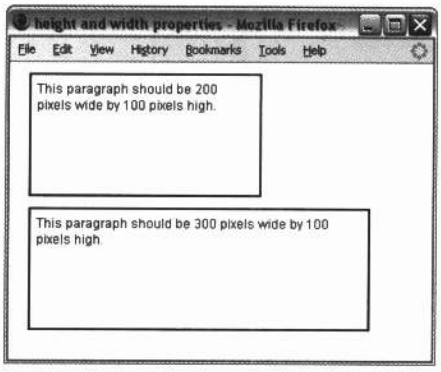


图 7-30

2. line-height 特性

当布局文本时，line-height 特性是最重要的一个特性，它可用于增加文本行之间的空间(打印设计人员将其称为行距)。

line-height 特性的值可以是数值、长度或者百分比。较好的方法是为这个特性指定与文本大小相同的度量单位。

下面的代码具有两条设置不同 line-height 特性的规则(ch07_eg26.css)：

```
p.one {
  line-height:16px;}
p.two {
  line-height:28px;}
```

在图 7-31 中可以看到，第一个段落没有 line-height 属性，而第二个和第三个段落对应于前面的规则。在每一个文本行之间添加额外的高度可使它们具有更好的可读性，特别是对于较长的文章。

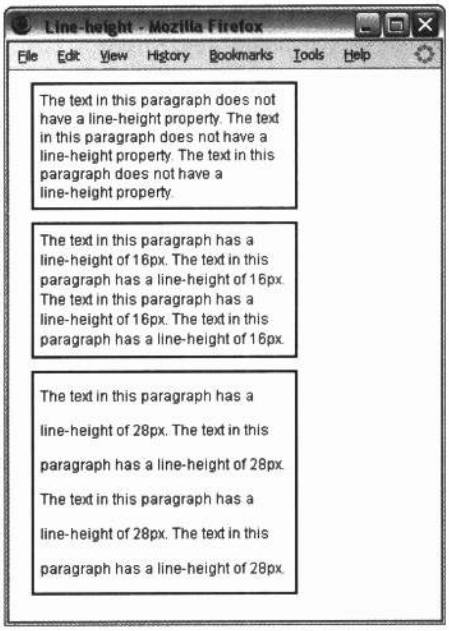


图 7-31

3. max-width 特性和 min-width 特性

max-width 特性和 min-width 特性分别用于指定框的最大和最小宽度。当希望创建能够伸缩的页面部分以适应用户的屏幕大小时，这两个特性非常有用。max-width 特性能够防止框太宽以至于不方便阅读(文本行太长，在屏幕上阅读它们较困难)，而 min-width 特性将有助于防止框太窄以至于不可阅读。但是，需要重点注意的是，IE7 浏览器是第一种支持这两个特性的 Internet Explorer 浏览器版本。

这两个特性的值可以是数值、长度或百分比，并且不允许负值。例如，下面的规则指定 元素不能小于 200 像素宽，并且不能大于 500 像素宽(ch07_eg27.css)：

```
div {min-width:200px;
   max-width:500px;
   padding:5px;
   border:1px solid #000000;}
```

图 7-32 给出了这个示例在浏览器中的外观，该图中给出了两个浏览器窗口，可以利用下载代码中的 ch07_eg27.html 文件亲自试验该 CSS 规则。第一个窗口的打开宽度超过 500 像素，并且框的宽度没有伸展超过 500 像素；第二个窗口的宽度小于 200 像素，这使得浏览器显示一个水平滚动栏，因为用户无法看到所有的框。

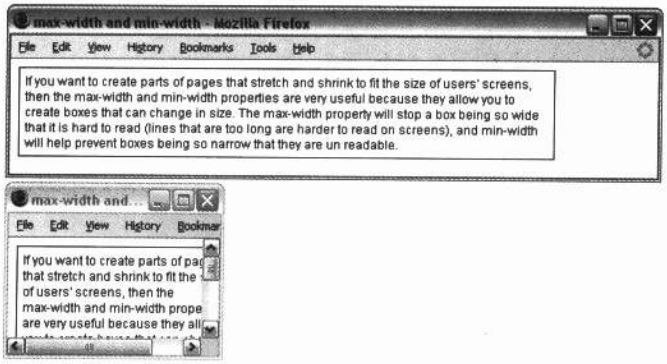


图 7-32

4. min-height 特性和 max-height 特性

min-height 特性和 max-height 特性分别对应于 min-width 特性和 max-width 特性，但是它们分别指定框的最小高度和最大高度。再次强调，IE7 浏览器是第一种支持这两个特性的 Internet Explorer 浏览器版本。

这两个特性的值可以是数值、长度或百分比，并且不允许负值。查看下面的示例(ch07_eg28.css)：

```
div {min-height:50px;
   max-height:200px;
   padding:5px;
   border:1px solid #000000;}
```

同样，这两个特性在创建能够根据用户浏览器窗口的大小而调整尺寸的布局时非常有用。但是，在图 7-33 中您可以看到令人感兴趣的现象：如果框中内容占用的空间大于允许的空间(因为具有这些规则)，则内容会超出框的范围(下一节中将介绍如何处理这种情况)。

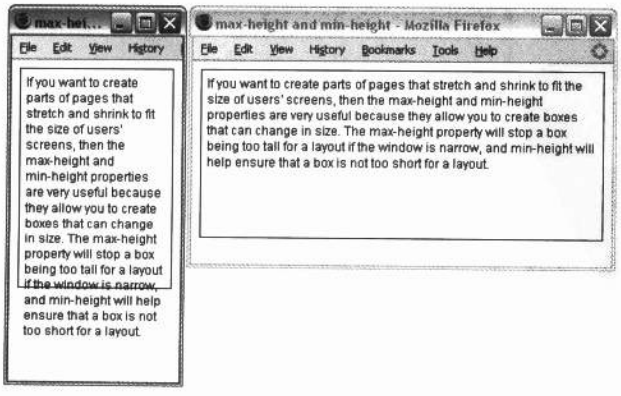


图 7-33

5. overflow 特性

在图 7-33 中能够看到，当控制框的大小时，希望框中容纳的内容需要的空间可能多于允许该框具有的空间。这种情况不一定只是由 min-height 和 max-height 或 min-width 和 max-width 特性引起的，还可能有许多其他原因，例如简单地为框设置固定宽度和高度或者赋予框负的页边空白。

overflow 特性设计用于处理这些情况，它能够采用表 7-19 中的某个值。

表 7-19

值	目　的
hidden	超出框范围的内容被隐藏
scroll	框被赋予滚动栏，以便用户滚动查看内容

查看下面的示例，其中两个 元素的高度和宽度通过 max-height 特性和 max-width 特性控制，使得 元素的内容超出框的范围。第一个元素的 overflow 特性值被设置为 hidden，第二个元素的 overflow 特性值被设置为 scroll(ch07_eg29.css)。

```
div {max-height:75px;
   max-width:250px;
   padding:5px;
   margin:10px;
   border:1px solid #000000;}
div.one {overflow:hidden;}
div.two {overflow:scroll;}
```

下面查看图 7-34，该图中显示了 ch07_eg29.html 文件的内容。可以看到这两个特性的效果——在第一个框中，当文本超出框的空间时被简单地切除；在第二个框中创建一个滚动栏，允许用户滚动查看内容。

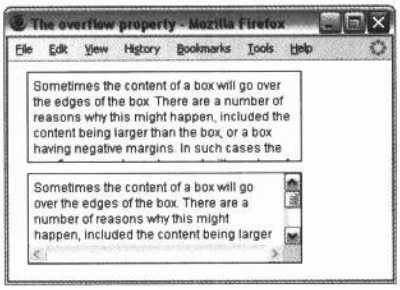


图 7-34

试一试　用于代码的样式表

我经常发现需要联机显示代码，因此编写了下面的样式表，以便能够定义类似于本书前面介绍的样式，用于显示 Web 中的代码。在下一章中您将看到，可以根据需要将这些代码包含到其他样式表中，这就表明它是可重用的样式表。

该样式表具有多个样式，分别用于块级元素和内联元素。表 7-20 给出了需要创建的样式。

表 7-20

样 式 名	内联或块级	用　途
codeInText	内联	对于句子中间的编写少量代码，使用等宽字体显示
codeForeground	块级	为了显示示例，以等宽字体高亮显示代码
codeBackground	块级	类似于 codeForeground 样式，但不高亮显示，因为前面已经看到该内容，或者它不是示例的关键点
keystroke	内联	用户应当使用键盘输入的键，以斜体字区分它们
importantWords	内联	关键术语的第一次使用：有助于用户浏览文档，因为它们以粗体字体显示
boxText	块级	创建一个重要的或者关键的注释块，其中注释位于框中并具有阴影背景
background	块级	创建一块斜体文本，该块具有旁白或者令人感兴趣的注释

(1) 需要做的第一件事情是为每一个样式创建类选择器。这里的几个样式没有使用元素名，因为样式可以应用于不同的元素(例如，框文本可以在<p>元素或者分组其他元素的 组中)。使用元素的选择器是代表代码的选择器。

```
code.codeInText{}
code.codeForeground {}
code.codeBackground {}
.keystroke {}
.importantWords {}
.boxText {}
.background {}
```

(2) 现在开始在每一个选择器的大括号内添加一些声明。首先是 codeInText 样式，该样式用于表示代码的句子或段落中间出现的单词。类似于关于编程的大多数书面文件中的传统，这些代码将以等宽字体显示。首选字型——使用 font-family 特性指定——是 Courier 字型，如果无法找到该字型，浏览器将尝试查找 Courier New 字型，如果仍然无法找到，则使用其默认等宽字体(但是大多数计算机确实已经安装 Courier 或 Courier New 字型)。

为了使代码更容易阅读，它们的字体将以粗体文本显示，使用 font-weight 特性指示这一点。

```
.codeInText {font-family:courier, "courier new", monospace;
font-weight:bold;}
```

(3) 第二种样式是 codeForeground 样式，该样式使用与 codeInText 样式中相同的字体类型。

下面是一些需要注意的事项：

- codeForeground 样式必须始终显示为块级元素，为了防止该类错误地用于内联元素，display 特性的值为 block，以确保它显示为一个块(在第 8 章中将更详细地介绍 display 特性)。
- letter-spacing 特性的值被设置为负值，因为等宽字体趋向于占用太多的页面宽度。因此，为了在一行中获得尽可能多的字符，该特性的值被赋予 - 0.1em(或者字体高度的 10%)。
- codeForeground 样式的背景色是灰色。这有助于区分代码，使其具有更好的可读性。框内部添加 1.5em 大小的内边距，以便文本不向右接触背景色的边缘——这也使代码更容易阅读。
- 页边空白确保了框不会接触任何其他框或者段落。底部页边空白比顶部页边空白小，并且这个样式表中的所有样式都使用这样的 margin 特性。

```
.codeForeground {
 font-family:courier, "courier new", monosapce; font-weight:bold;
 letter-spacing:-0.1em;
 display:block;
 background-color:#cccccc;
 padding:0.5em;
 margin-bottom:1em; margin-top:1.5em;}
```

(4) codeBackground 样式与 codeForeground 样式一致，除了它的 background-color 特性是白色之外：

```
.codeBackground {
 font-family:courier, "courier new", monosapce; font-weight:bold;
 letter-spacing:-0.1em;
 display:block;
 background-color:#ffffff;
 padding:0.5em;
 margin-bottom:1em; margin-top:1em;}
```

(5) keystroke 样式的字型是 Times，如果无法获得 Times 字型，将使用 Times New Roman 字型，如果这两种字体都无法找到，则浏览器使用默认的 serif 字型。keystroke 样式应该采用斜体，如下所示：

```
.keyStroke {
 font-family:times, "Times New Roman", serif;
 font-style:italic;}
```

(6) importantWords 样式仅为粗体：

```
.importantWords {font-weight:bold; }
```

(7) boxText 样式是一种粗体字体样式，并且具有非常亮的灰色背景；它的真正不同之处是起具有边框。类似于 codeForeground 样式，boxText 样式具有内边距，以便文本不接触边框——使其更容易阅读——并且该样式的左边和右边以及垂直方向具有页边空白，将其与其他元素隔开。注意，底部页边空白比顶部页边空白稍微小一点。

```
.boxText {
  font-weight:bold;
  background-color:#efefef;
  width:90%;
  padding:1em;
  margin-left:3em; margin-right:3em; margin-bottom:1em; margin-top:1.5em;
  border-style:solid; border-width:1px; border-color:#000000;}
```

(8) 最后一个样式是 background 样式。这个样式采用斜体，与 boxText 样式具有相同的内边距和页边空白量。

```
.background {
  font-style:italic;
  width:90%;
  padding:1em;
  margin-left:3em; margin-right:3em; margin-bottom:1em; margin-top:1em;}
```

(9) 对于这个示例，为<p>元素和<body>元素分别包含了一条规则(但它们都不是用于代码样式的标准 CSS 的一部分)：

```
body {
  color:#000000;
  background-color:#ffffff;
  font-family:arial, verdana, sans-serif;
  font-size:12px;}
p {margin-bottom:1em; margin-top:1.5em;}
```

(10) 将这个文件保存为 codeStyles.css。然后查看下面的 XHTML 文件，该文件使用这个样式表。<link />元素指示这个示例所使用的样式表。然后可以看到一些通过 class 属性与这些样式关联的元素：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>CSS Example</title>
  <link rel="stylesheet" type="text/css" href="codeStyles.css" />
</head>
<body>
<p>You are about to see some <code class="codeInText">codeInText</code>
followed by some <span class="importantWords">importantWords</span>, and the
font for a <span class="keystroke">keystroke</span>.</p>
<p>Next you will see some foreground code:</p>
<code class="codeForeground">p {font-family:arial, sans-serif;
font-weight:bold;}</code>
<p>Next you will see some background code:</p>
<code class="codeBackground">p {font-family:arial, sans-serif;
font-weight:bold;}</code>
<p class="boxText">This is some boxed text for important statements.</p>
```

```
    <p class="background">Here is a background comment or aside.</p>
</body>
</html>
```

如果在浏览器中查看这个示例，它的外观将如图 7-35 所示。

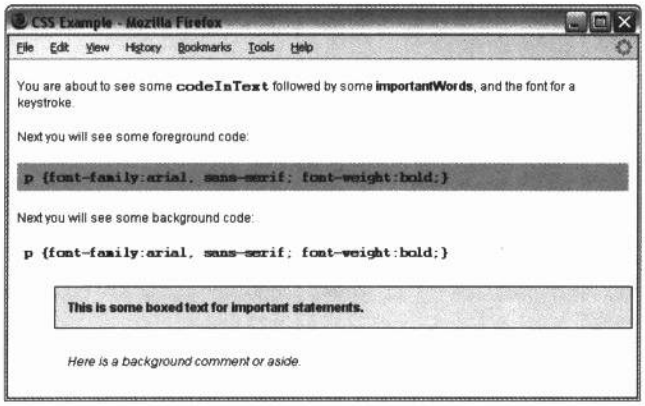


图 7-35

工作原理

您已经通篇阅读了本章，应该能够很好地理解这个示例的工作原理，但是有必要回顾其中的一些关键点。

在示例 XHTML 文档头中使用<link />元素将样式表链接到该文档。

```
<link rel="stylesheet" type="text/css" href="codeStyles.css" />
```

这个样式表的特点是它能够用于多个文档，这就是样式规则没有放置在 XHTML 文档的<style>元素内的原因。

注意，其中几个样式使用 margin 特性，并且顶部的页边空白大于底部的页边空白。我倾向于底部页边空白稍微比顶部页边空白小一些，以便当垂直方向相邻的页边空白折叠时，能够知道哪一个页边空白更可能显示出来。这一点非常有用，因为实际上看不到页边空白的边缘，而使用 border 特性可以查看框的边缘。

```
.codeBackground {
  font-family:courier, "courier new", monospace; font-weight:bold;
  letter-spacing:-0.1em;
  display:block;
  background-color:#ffffff;
  padding:0.5em;
  margin-bottom:1em; margin-top:1em;}
```

在这些示例中，左边和右边的页边空白较大，从而框被缩进。如果使用 text-indent 特性，则仅有第一行被缩进。

在代码的块框中存在两个特性，在样式表的其他任何位置都没有使用这两个特性。

letter-spacing 特性用于将尽可能多的字母放置在同一行中显示。但是，不能将这些字母间距设置太窄，否则用户将无法阅读单词(这里 em 的 1/10 是最大值)。另一个特性是 display，它的值是 block，以确保 codeForeground 样式和 codeBackground 样式被视为为块级元素。

由于左右边空的出现，boxText 样式和 background 样式被缩进，以便它们能够清晰地与围绕它们的文本分离，并且能够突出显示。

您可能已经注意到，样式表中指定的所有长度都以 em 为单位，以便它们与文档中文本的默认大小相关。如果其中某些元素的值以绝对大小给出，则它们可能会突然显示为比周围的文本大很多或者小很多，因为它们的长度不是相对长度。

7.10　本章小结

本章介绍了如何编写 CSS 样式表。CSS 样式表由一些规则组成，这些规则首先选择它们将应用的元素，然后包含一些特性-值对，特性-值对用于指定元素内容的外观。

本章也介绍了如何改变字体和文本的外观。

CSS 将每一个元素视为如同其是一个独立的框，然后使用特性控制每一个框的外观，以此来设法显示文档。本章介绍了如何设置每一个框的面积、边框、内边距和页边空白。

在下一章中将介绍更多的一些特性，并且介绍如何使用 CSS 来定位元素，它们能够为页面创建更有吸引力的布局。您将看到如何通过样式表将内容插入到文档中、如何处理项目列表、如何创建计数器等内容。

7.11　练习

1. 返回到本章中的第一个“试一试”示例，在其中添加一些样式，以显示每一种字体的粗体和斜体版本。最终得到的页面将如图 7-36 所示。

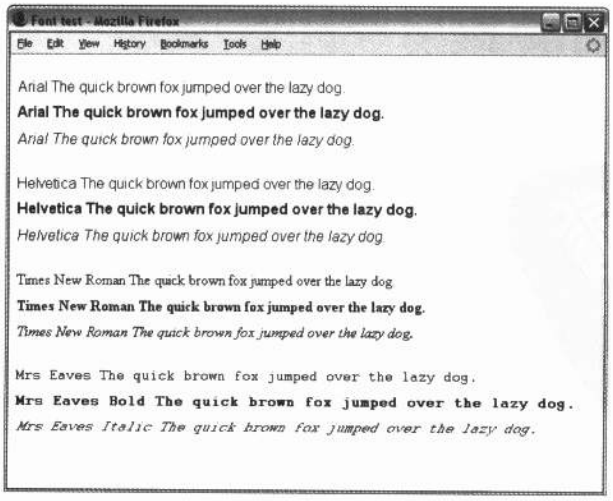


图 7-36

只允许在源文档中使用元素和
元素以及在样式表中使用类选择器。在 元素的内容上需要添加一个顶部页边空白，以将它们互相分离。

2. 查看下面的 XHTML 页面：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Font test</title>
  <link rel="stylesheet" type="text/css" href="tableStyles.css" />
</head>
<body>
  <table>
    <tr>
      <th>Quantity</th>
      <th>Ingredient</th>
    </tr>
    <tr class="odd">
      <td>3</td>
      <td>Eggs</td>
    </tr>
    <tr>
      <td>100ml</td>
      <td>Milk</td>
    </tr>
    <tr class="odd">
      <td>200g</td>
      <td>Spinach</td>
    </tr>
    <tr>
      <td>1 pinch</td>
      <td>Cinnamon</td>
    </tr>
  </table>
</body>
</html>
```

现在创建 tableStyles.css 样式表，使这个示例的外观如图 3-37 所示。

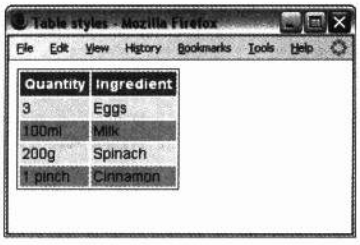


图 7-37

不要担心获得的尺寸与图 7-37 中的屏幕截图不同，但需要确保在单元格中添加内边距，并且在外部添加边框。IE 浏览器在默认情况下将创建白色边框，在第 8 章中将介绍如何删除边框。

更多层叠样式表

本章将介绍更多关于 CSS 方面的知识，首先介绍 CSS 规范的前几章未涉及其他的许多特性，使用这些特性可控制链接、背景、列表样式、表样式和框周围的外边框(与边框不同的最后一种特性)的表示。然后介绍:before 伪类和:after 伪类，使用它们可以在即将赋予样式的源文档中的指定元素之前或之后添加内容，并且这些内容不属于源文档。最后，本章将介绍如何使用 CSS 定位页面中的框——从而介绍如何使用它们布局页面，而不是使用表进行布局。

阅读完本章之后，您将了解如何使用 CSS 来控制如下的方面：

- 链接的表示
- 文档的背景
- 项目列表和编号列表的样式
- 表的外观
- 框周围的外边框
- 能够获得焦点或者激活的框
- 在 XHTML 文档中的某个元素之前或之后添加内容
- 3 种定位方案，可用于确定框将出现在页面上的哪个位置——这是为利用 CSS 创建布局所做的准备

注意：

本章中介绍的某些功能并未被浏览器广泛支持。但其依然值得学习，以便您了解 CSS 的发展方向。

8.1 链接

前面已经介绍过，color 特性可改变任何元素中的文本的颜色，Web 设计人员通常在应用于<a>元素的规则中使用这个特性来改变链接的颜色。但是如果这样做的话，链接将总是保持一种颜色——即使已经访问使用鼠标指针悬停或者单击。

稍微改变已访问的链接颜色的能力将有助于用户浏览站点，当用户将鼠标指针悬停在链接上时改变链接的颜色，有助于鼓励用户单击该链接。因此，当创建一个能够改变链接颜色的规则时，表 8-1 中列举的伪类能够帮助将不同的样式关联到处于不同状态的链接。

表 8-1

伪　类	目　的
link	用于一般状态的链接的样式
visited	用于已经访问的链接的样式
active	用于当前激活(单击)的链接的样式
hover	当鼠标指针悬停在链接上时的链接样式

当使用这些伪类时，很可能需要同时使用下面的特性：

- color：通常用于改变链接的颜色。本章前面已经提到，如果能够稍微区分已经访问的链接和未访问的链接，将会非常有帮助，因为这样有助于用户了解自己已经访问过哪些页面。此外，当用户将鼠标指针悬停在链接上时，稍微改变链接的颜色有助于鼓励用户单击该链接。
- text-decoration：通常用于控制链接是否具有下划线。在 Web 中，链接的下方通常具有下划线，但从 20 世纪 90 年代末开始更趋向于不在链接下方使用下划线。使用 text-decoration 特性，可以指定链接下没有下划线，或者仅当用户将鼠标指针悬停在链接上或选中它时才将其设置为具有下划线。
- background-color：高亮显示链接，如同使用荧光记号笔绘制。该属性主要用于用户将鼠标指针悬停在链接上时提供轻微的颜色改变。

在下面的示例中，当用户与链接交互时，将改变链接的样式(ch08_eg01.css)：

```
body {background-color:#ffffff;}
a {
  font-family: arial, verdana, sans-serif;
  font-size:12px;
  font-weight:bold;}
a:link {
  color:#0000ff;
  text-decoration:none;}
a:visited {
  color:#333399;
  text-decoration:none;}
a:active {
  color:#0033ff;
  text-decoration:underline;}
a:link:hover {
  background-color:#e9e9e9;
  text-decoration:underline;}
```

图 8-1 中给出了这个样式表在浏览器中的外观(该样式表用于 ch08_eg01.html)，但是在这个打印图中很难看到完全的效果，即无法看到用户鼠标指针悬停在链接上和访问过站点之后链接改变颜色。因此，请读者利用本章的下载代码测试这个示例。

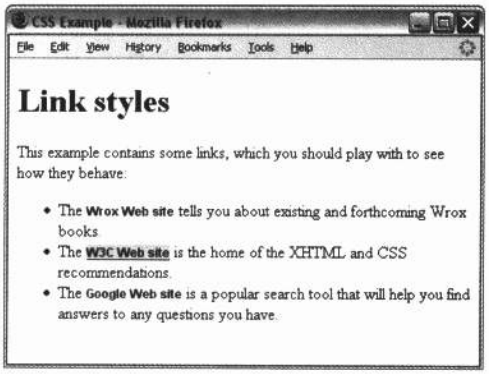


图 8-1

还有两个伪类:focus 和:active，它们分别用于当元素获得焦点或者激活时改变元素的样式。本章后面将介绍这两个伪类。

8.2 背景

表 8-2 中列举了 CSS 中的 6 个特性，它们可用于指定整个浏览器窗口或者单个框中的背景的显示效果。

表 8-2

特　性	目　的
background-color	指定一种颜色作为页面或者框的背景色
background-image	将一幅图像设置为页面或者框的背景
background-repeat	指示背景图像是否应该在页面或者框中重复显示
background-attachment	指示背景图像应该固定在页面的某个位置，并且指示当用户向下滚动页面时图像是否应当停留在该位置
background-position	指示图像应当定位在浏览器窗口或者包含它的框中的特定位置处
background	一种简写形式，可用于指定上面所有的特性

需要注意的是，在较老的浏览器版本中，简写形式 background 特性比其他单独的特性获得了更好的支持，但是在使用简写形式的 background 特性之前，需要了解这些特性能够采用的值。

8.2.1 background-color 特性

background-color 特性可为页面或 CSS 创建的框内部指定一种单色背景。

这个特性的值可以是十六进制编码、颜色名或者 RGB 值(附录 D 中更深入地介绍了颜色方面的内容)，例如(ch08_eg02.css)：

```
body {background-color:#cccccc; color:#000000;}
b {background-color:#FF0000; color:#FFFFFF;}
p {background-color: rgb(255,255,255);}
```

当为<body>元素设置 background-color 特性时，该特性将影响整个文档；当该特性用于其他任何元素时，它将在为该元素创建的框的边框内使用指定的颜色。图 8-2 显示了前面的样式用于 ch08_eg02.html 时的效果。

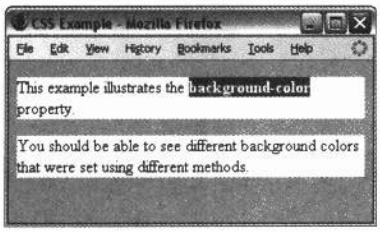


图 8-2

注意：

编者本人几乎在编写的所有样式表中都为<body>元素添加一条规则，用于设置 background-color 特性，原因很简单：某些人可能为他们的计算机设置一种非纯白色的背景(通常是因为这样可以减轻眼睛的疲劳)。当操作系统的背景色改变后，Web 浏览器中的背景色通常为该背景色(类似 Word 字处理软件这样的应用程序也是如此)。如果不指定这个特性，则无法保证站点访问者具有与您相同的背景色。

8.2.2　background-image 特性

background-image 特性可用于在 CSS 中将一幅图像设置为任何框的背景，并且它的效果非常强大。该特性能够附带的值如下：起始字母为 url，然后在小括号和双引号内包括图像的 URL：

```
body {background-image: url("images/background.gif);" }
```

background-image 特性重写 background-color 特性。但是，最好在使用背景图像时同时提供 background-color 特性，并且将其值设置为与图像主要颜色类似的颜色，因为当正在加载页面或者因为各种原因无法加载图像时，页面可以使用这种颜色。

下面的示例使用了一幅 200 像素宽、150 像素高的单一背景图像。默认情况下，这幅图像多次重复，以布满页面(ch08_eg03.css)。background-color 特性被设置为与背景图像相同的颜色(以防止图像无法加载)：

```
body {
  background-image: url("images/background.gif");
  background-color: #cccccc;}
```

图 8-3 给出了这个示例在浏览器中的外观(ch08_eg03.html)。

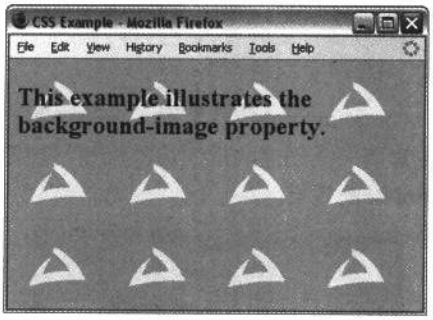


图 8-3

这不是一个关于背景图像的优秀示例，但是它演示了背景图像的工作原理。该示例中存在的问题是，背景图像中使用的颜色和它上方显示的文本之间的颜色对比度太低，以至于难以阅读文本。

必须确保背景图像和它上方显示的文本之间具有足够的对比度；否则用户阅读文本时将存在困难。此外，低对比度的图像(由相似颜色组成的图像)通常是较好的背景图像，因为寻找一种在高对比度图像之上具有很好可读性的颜色较为困难。

图 8-4 给出了该背景图像的改进示例，其中的文本位于单色上方，这使得文本能够更容易阅读。这一次还使用了一幅更大的图像(ch08_eg03b.html)。

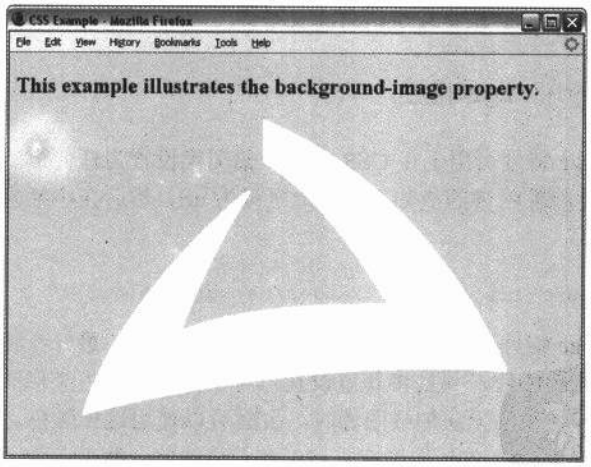


图 8-4

您可能已经注意到，没有方法可用于表达背景图像的预期宽度和高度，并且 background-image 特性不具有 alt 属性(由于各种原因无法看到图像时显示的备选文本)；因此，背景图像不能用于传送任何重要信息，因为对于看不到背景图像的访问者来说，图像上的信息是不可访问的。

在使用大图像文件作为背景图像时需要非常谨慎，因为加载大文件的速度可能非常慢。图像的文件越大(即字节数越大)，则加载和显示它的时间越长。

background-image 特性可以很好地作用于大多数块级元素，但是某些较老的浏览器在显示表中的背景图像时会存在问题。

8.2.3　background-repeat 特性

默认情况下，background-image 特性会多次重复以布满整个页面，创建一种称为墙纸的效果。墙纸由一幅多次重复的图像组成，并且看不到它的边缘(如果图像经过良好设计)。因此，重要的是任何模式都应当是可镶嵌的或者能够很好地结合在一起。墙纸通常由一些纹理组成，例如纸张、大理石花纹或者抽象的表面，而不是照片或者徽标。

如果不希望图像在页面背景中多次重复，可以使用 background-repeat 特性，该特性有 4 个可选用的值，如表 8-3 所示。

表 8-3

值	目　的
repeat	图像多次重复以覆盖整个页面。
repeat-x	图像将在页面水平方向多次重复(但在垂直方向不重复)
repeat-y	图像将在页面垂直方向多次重复(但在水平方向不重复)
no-repeat	图像将仅显示一次

具有不同值的特性能够带来各种令人感兴趣的效果，因此值得依次查看。前面已经给出 repeat 值的效果，下面查看 repeat-x 值的效果，该值沿着浏览器的 x 轴创建一条水平栏(ch08_eg04.css)：

```
body {
  background-image: url("images/background_small.gif");
  background-repeat: repeat-x;
  background-color: #ffffff;}
```

在图 8-5 中可以看到使用这个特性后的效果。

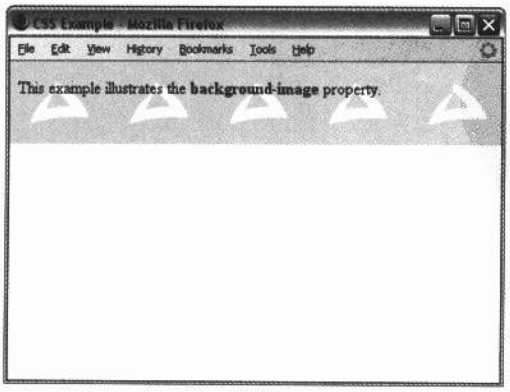


图 8-5

repeat-y 值的工作原理与 repeat-x 值相同，但作用方向不同：它沿着浏览器 y 轴方向创建垂直栏(ch08_eg05.css)：

```
body {
  background-image: url("images/background_small.gif");
  background-repeat: repeat-y;
  background-color: #ffffff;}
```

在图 8-6 中可以看到结果，该结果由左边向下的边栏组成。

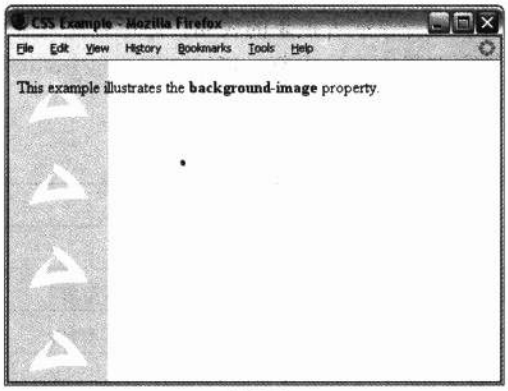


图 8-6

最后一个值是 no-repeat，默认情况下它仅在浏览器窗口的左上角显示图像的一个实例(ch08_eg06.css)：

```
body {
  background-image: url("images/background_small.gif");
  background-repeat: no-repeat;
  background-color: #eaeaea;}
```

图 8-7 中给出了结果；注意，页面的背景色被设置为与图像中所使用的颜色相同。

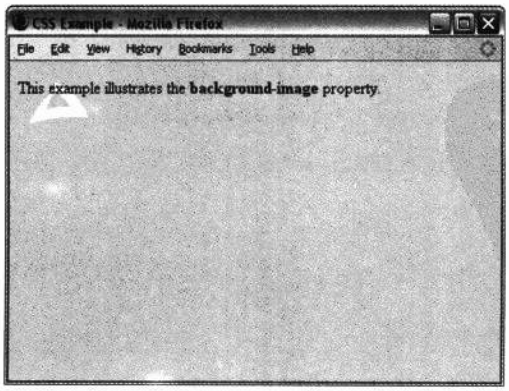


图 8-7

8.2.4　background-position 特性(用于固定背景的位置)

当 background-color 特性的值与图像的背景色相同时，无法看到图像的边缘，如图 8-7 所示。但是，您可能希望修改这幅图像的位置，此时可以使用 background-position 特性，该特性可采用表 8-4 中所示的值。

表 8-4

值	意　义
x%　y%	沿着 x 轴(水平)和 y 轴(垂直)的百分比
xy	沿着 x 轴(水平)和 y 轴(垂直)的绝对长度
left	在页面或者包含元素的左边显示
center	在页面或者包含元素的中间显示
right	在页面或者包含元素的右边显示
top	在页面或者包含元素的顶部显示
bottom	在页面或者包含元素的底部显示

下面的示例固定图像的位置，如图 8-8 所示(ch08_eg07.css)：

```
body {
  background-image: url("images/background_small.gif");
  background-position: 50% 20%;
  background-repeat: no-repeat;
  background-color: #eaeaea; }
```

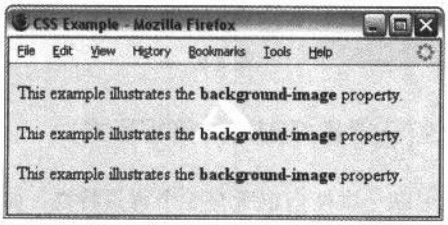


图 8-8

这幅图像将水平方向居中(因为它被设置为距离页面左侧 50%的屏幕宽度)，并且垂直方向距离屏幕顶端向下 1/5 处(因为它被定位为距离屏幕顶部 20%的窗口高度)。

8.2.5　background-attachment 特性(用于水印)

background-attachment 特性可用于指定称为水印的图像。这种设置的关键区别是，即使用户向上或向下滚动页面或者滚动页面中的所有其他元素，背景图像都将停留在相同的位置。background-attachment 特性可以采用两个值，如表 8-5 所示。

表 8-5

值	目　的
fixed	用户向上或向下滚动页面时图像不会移动
scroll	图像停留在页面背景上的相同位置处。如果用户向上或向下滚动页面，则图像也随之移动

在下面的示例中，即使用户滚动页面，图像也停留在页面的中心位置(ch08_eg08.css)：

```
body {
  background-image: url("images/background_small.gif");
  background-attachment: fixed;
  background-position: center;
  background-repeat: no-repeat;
  background-color: #eaeaea; }
```

在图 8-9 中，用户已经向下滚动页面，但是图像仍然停留在窗口的中心位置。

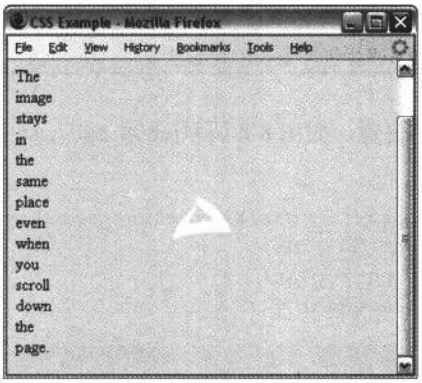


图 8-9

8.2.6 background 特性(获得良好支持的简写形式)

利用 background 特性可以一次性指定所有 5 个背景特性。如果不提供任何值，则将使用默认值。可以以任意顺序提供值：

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

例如，可以编写如下代码：

```
body {background: #cc66ff; url(images/background_small.gif) fixed
no-repeat center;}
```

该代码所产生的效果与图 8-9 中的示例相同。

8.3　列表

第 1 章中已经介绍了列表，列表在传达一组编号的或者采用项目符号的要点时非常有用。可以简单地使用<ul>和<li>元素创建无序列表，或者使用<ol>和<li>元素创建有序列表，但是使用 CSS 可以在很大程度上控制列表的外观。

注意，项目符号或者编号列表情况下的数值称为标记符。

在本节中将介绍表 8-6 中列举的列表特性。

表 8-6

特　性	目　的
list-style-type	可用于控制标记符(项目符号或者数值)的形状或外观
list-style-position	指定列表中的较长项是否占用多行文本，如果占用多行，则指定换到的第二行是应该与第一行的文本对齐还是从标记符起始位置的下方开始
list-style-image	指定一幅图像作为标记符，而不是项目符号或者数值
list-style	作为上述特性的简写形式
marker-offset	指定列表中标记符和文本之间的距离

8.3.1　list-style-type 特性

list-style-type 特性可用于控制无序列表中项目符号(也称为标记符)的形状或样式或者有序列表中编号字符的样式。表 8-7 中给出了用于无序列表的一些标准样式。

表 8-7

值	标 记 符
none	没有标记符
disc (默认值)	填充的圆
circle	空心圆
square	填充的正方形

表 8-8 中列举了用于有序列表的广泛支持的值。

表 8-8

值	意　义	示　例
decimal	数值	1、2、3、4、5
decimal-leading-zero	在前面添加 0 的数值	01、02、03、04、05
lower-alpha	小写字母数字字符	a、b、c、d、e

(续表)

值	意　义	示　例
upper-alpha	大写字母数字字符	A、B、C、D、E
lower-roman	小写罗马数字	i、ii、iii、iv、v
upper-roman	大写罗马数字	I、II、III、IV、V

list-style 特性可以用于<ul>元素和<ol>元素，也可以用于<li>元素。下面的示例演示所有这些样式(ch08_eg09.html)：

```
li.a {list-style:none;}
li.b {list-style:disc;}
li.c {list-style:circle;}
li.d {list-style:square;}
li.e {list-style:decimal;}
li.f {list-style:lower-alpha;}
li.g {list-style:upper-alpha;}
li.h {list-style:lower-roman;}
li.i {list-style:upper-roman;}
```

图 8-10 给出了该示例的结果，其中显示了每一种项目符号的效果。

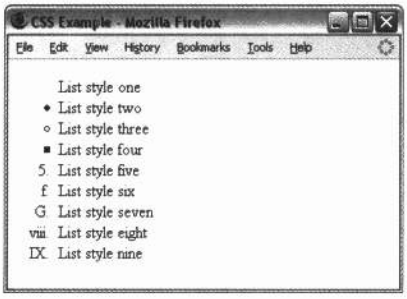


图 8-10

8.3.2 list-style-position 特性

list-style-position 特性指示标记符应当出现在包含项目列表的框的内部还是外部。

当项目列表中某一项的文本超过一行时才会显现具体的区别。因为这个特性设置换行的文本是出现在项目符号的下方还是与第一行文本位置对齐。这个特性具有两个值，如表 8-9 所示。

表 8-9

值	目　的
inside	如果文本超出一行，则换行的文本将位于标记符的下方。如果列表具有值 outside，则换行的文本将会缩进显示
outside	如果文本超出一行，则换行的文本将与第一行文本的起始位置(项目符号的右边)对齐

下面是这个特性的编写方式：在该示例中，该特性位于<ul>元素或<ol>元素内(ch08_eg10.css)：

```
ul {list-style-position:outside; }
ol {list-style-position:inside; }
```

图 8-11 给出了该示例在浏览器中的显示效果。

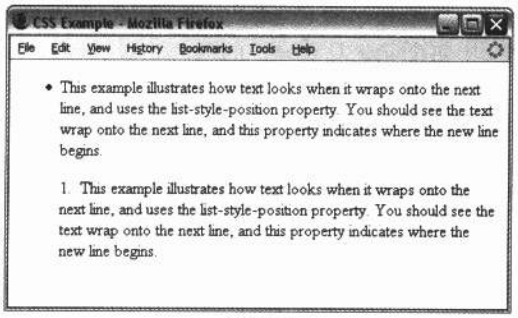


图 8-11

在图 8-11 中，值为 outside 的 list-style-position 特性在文本的左边创建项目符号。在值为 inside 的 list-style-position 特性创建的列表中，列表项的起始位置从值为 outside 时的列表项的内容处开始，并且添加的标记符位于文本中，而不是单独位于一边。

8.3.3　list-style-image 特性

list-style-image 特性可用于指定一幅图像，从而可以使用自己的项目符号样式。该特性的语法如下所示，类似于 background-image 特性，特性的值以字母 url 开头，后面紧跟图像的 URL，并且 URL 位于小括号和双引号中(ch08_eg11.css)：

```
li {list-style-image: url("images/bulletpoint.gif");}
```

该示例在浏览器中的显示效果如图 8-12 所示，其中具有一些三角形状的项目符号。

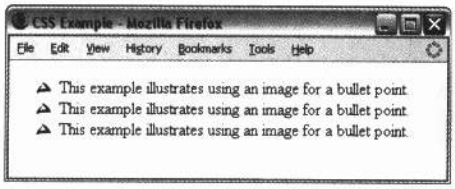


图 8-12

如果图像无法显示，浏览器将只显示一个圆点，而不是破碎的图像符号。

注意：

如果使用的是嵌套列表，则该特性的值将继承它的父元素。如果不希望继承，可以赋予该特性 none 值。

8.3.4 list-style 特性(简写形式)

list-style 特性可以一次表达其他 3 种特性，并且这 3 种特性可以按照任意顺序出现。其语法如下所示：

```
ul {list-style: inside circle;}
```

请记住，也可以为<ul>、<ol>、<li>、<dl>、<dt>和<dd>元素设置 border、padding 和 margin 特性，因为在 CSS 中每个元素具有自己的框。

8.3.5 marker-offset 特性

marker-offset 特性可用于指定标记符和与该标记符相关的文本之间的距离，它的值必须是一个长度，如下所示：

```
li {marker-offset:2em;}
```

遗憾的是，IE7 浏览器和 Firefox 2 浏览器不支持这个特性。

8.4 表

在上一章中，很多示例对表使用了 CSS。通常用于<table>、<td>和<th>元素的特性包括以下一些特性：

- padding，用于设置表单元格的边框和它的内容之间的空间量——这个特性对于表的可读性来说非常重要。
- border，用于设置表边框的特性。
- text 和 font，用于改变表单元格中编写的任何内容的外观。
- text-align，用于设置表单元格中内容的水平对齐方式，包括左、右或居中 3 种对齐方式。
- vertical-align，用于设置表单元格中内容的垂直对齐方式，包括顶部、中部或底部 3 种对齐方式。
- width，用于设置表或表单元格的宽度。
- height，用于设置单元格的高度(通常也用于行)。
- background-color，用于改变表或单元格的背景色。
- background-image，用于在表或单元格中添加一幅背景图像。

需要注意的是，除了 background-color 特性和 height 特性之外，最好避免将这些特性用于<tr>元素，因为当这些特性用于行时，浏览器对它们的支持并没有像它们用于单个单元格时那么好。

查看图 8-13 中的表，可能您会觉得熟悉该表，因为上一章的开头已经给出该表，但是这一次添加了一个<caption>元素(ch08_eg12.html)。

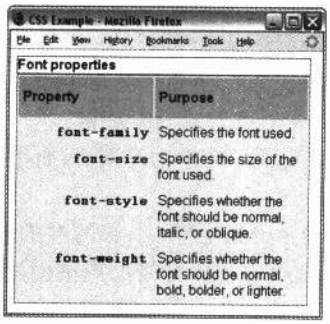


图 8-13

下面查看用于这个表的样式表(ch08_eg12.css)：

```
body {color:#000000; background-color:#ffffff;}
h1 {font-size:18pt;}
p {font-size:12pt;}
table {
  background-color:#efefef;
  width:350px;
  border-style:solid;
  border-width:1px;
  border-color:#999999;
  font-family:arial, verdana, sans-serif;}
caption {
  font-weight:bold;
  text-align:left;
  border-style:solid; border-width:1px; border-color:#666666;
  color:#666666;}
th {
  height:50px;
  font-weight:bold;
  text-align:left;
  background-color:#cccccc;}
td, th {padding:5px;}
td.code {
  width:150px;
  font-family:courier, courier-new, serif;
  font-weight:bold;
  text-align:right;
  vertical-align:top;}
```

下面是这个示例中需要注意的一些关键点，可以使用本章后面即将介绍的一些新的表特性来修改其中一些关键点：

- <table>元素具有一个 width 特性，用于将表的宽度固定为 350 像素；否则表将根据需要占用尽可能多的屏幕空间，以在一行中显示尽可能多的文本。

- <table>元素还有一个 border 特性设置，该特性在表的周围创建一个像素宽的边框。但是，表中的所有单元格不继承这个特性。
- <caption>元素具有 font-weight、border 和 text-align 特性设置。默认情况下，文本是普通(非粗体)类型、居中对齐并且没有边框。
- <th>元素具有 50 像素的高度，文本为左对齐(而不是居中对齐，居中对齐是默认设置)。
- <th>和<td>元素都有一个 padding 特性，该特性的值为 5 px (5 像素)，以便单元格的内容不接触单元格的边框。在单元格的周围创建空间非常重要，这将使表具有更好的可读性。
- class 属性值为 code 的<td>元素具有一个 width 特性，该特性的值为 150 px (150 像素)。该特性设置确保了整列的内容位于一行中。但是，无法将样式赋予列，但是对于 width 特性，一旦对某个元素设置该特性，则不需要对该列中的所有其他元素设置该元素。

注意：

不同的浏览器对利用 CSS 赋予表样式的支持并不完整。例如，虽然可以为标题设置 border 特性，但无法为其设置 height 特性，所以应当在尽可能多的浏览器中试验示例。

应该注意两列之间的间隔(间隔在表头与单元格之间非常明显)。默认情况下，在表中每一个单元格之间都会创建一个边框，以便在每个单元格之间创建少量空间，从而防止无法指定规则来创建这种必要的间隔。但是，可以使用 border-spacing 特性来删除间隔，下一节将介绍该特性。

8.4.1 表的特性

很多特性仅与表相关，表 8-10 中列举了这些特性。其中，border-style 特性可以附带一些特殊的值。当学习边框方面的知识时，了解如何通过使用 border-collapse 特性控制的两种模型之一来显示边框会非常有帮助。

表 8-10

特　性	目　的
border-collapse	该特性指示浏览器是应当控制相互接触的相邻边框的外观，还是让每个单元格维持自己的样式
border-spacing	指定表的单元格之间应该具有的宽度
caption-side	指定标题应当显示在表的哪一边
empty-cells	指定单元格为空时是否应当显示边框
table-layout	该特性允许浏览器加速表的布局，方式是浏览器使用遇到的第一个宽度特性作为列的剩余部分的宽度(而不是必须加载整个表后才显示它)

8.4.2　border-collapse 特性

border-collapse 特性指定浏览器是应当显示每一个边框——即使两个相邻单元格具有不同的边框特性——还是应当基于一组内置的复杂规则来自动决定显示哪一个边框。表 8-11 给出了 border-collapse 特性的两个可选值。

表 8-11

值	目　的
collapse	水平边框将折叠，垂直边框则互相邻接(关于 CSS 规范中不同边框规则的冲突解决方法存在一些复杂的规则，但是应当试验它们以了解其工作原理)
separate	遵守独立的规则，并且可使用一些不同的特性来进一步控制外观

下面的示例中存在两个表，第一个表的 border-collapse 特性的值为 collapse，第二个表的 border-collapse 特性的值为 separate，并且这两个表都包含分别具有虚线和实线边框的相邻单元格：

```
table.one {border-collapse:collapse;}
table.two {border-collapse:separate;}
td.a {border-style:dotted; border-width:3px; border-color:#000000;
  padding:
10px;}
td.b {border-style:solid; border-width:3px; border-color:#333333;
  padding:
10px;}
```

图 8-14 给出了该示例的显示效果，对于 collapse 值，浏览器显示的边框互相拆叠，从而实线边框的优先级高于虚线边框。当然，如果边框都是实线，显示的效果就不会很奇怪，但是该示例很好地演示了 collapse 值所带来的效果(事实上，您很可能不希望让内部线比外部线粗——因此这可能正是您所需的效果)。

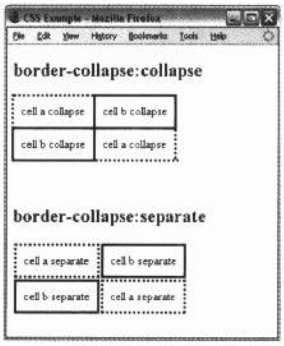


图 8-14

在本节的开始部分给出了一个表，其中表题头与单元格之间具有浅灰色的间隔。利用 border-collapse 特性可以去除该间隔。图 8-15 给出了本章开始部分示例中的边框折叠后的显示效果。

图 8-15

如果对该特性使用 separate 值，则可以使用另外两个特性进一步控制边框的表示：

- border-spacing
- empty-cells

下面的小节中将讨论这两个特性。

8.4.3 border-spacing 特性

border-spacing 特性指定相邻单元格的边框之间的距离，它可以采用一个或两个值这些值应当是长度单位。

如果提供一个值，则该值将同时应用于垂直和水平边框：

```
td {border-spacing:15px;}
```

也可以指定两个值，此时第一个值用于水平间隔，第二个值用于垂直间隔：

```
td {border-spacing:2px; 4px;}
```

图 8-16 中给出了该特性的显示效果(ch08_eg15.html，使用的样式表为 ch08_eg15.css)：

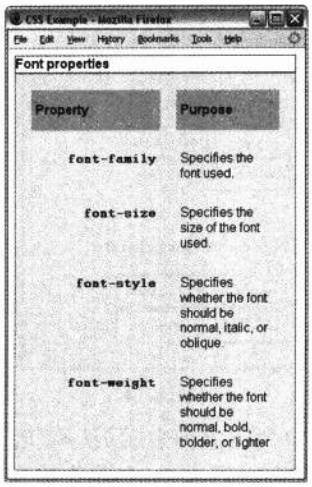


图 8-16

注意，Internet Explorer 浏览器直到 IE7 版本才支持这个特性。

8.4.4　caption-side 特性

caption-side 特性用于指定<caption>元素的内容相对于表的放置位置，表 8-12 中给出该特性的可选值。

表 8-12

值	目　的
top	标题出现在表的上方(默认值)
right	标题出现在表的右边
bottom	标题出现在表的下方
left	标题出现在表的左边

例如，下面的代码将标题设置为放置在表的下方(ch08_eg16.css)：

```
caption {caption-side:bottom}
```

遗憾的是，IE 浏览器直到 IE7 版本才开始支持这个特性。图 8-17 给出了 caption-side 特性的效果；其中这个表的标题已经移动到表的下方(而不是上方)。

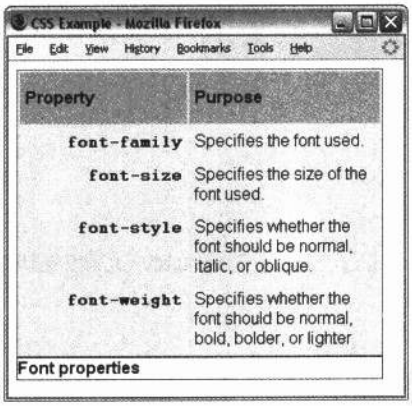


图 8-17

8.4.5　empty-cells 特性

empty-cells 特性指示当单元格没有内容时是否显示它的边框。该特性可以选用表 8-13 中的 3 个值之一。

表 8-13

值	目　的
show	即使单元格为空，也会显示边框(在 Firefox 浏览器中该值是默认值)。
hide	如果单元格为空，将隐藏边框(在 IE 浏览器中该值是默认值)。
inherit	边框将服从包含它的表的规则(仅用于嵌套表)。

注意：

如果希望显式地隐藏或显示边框，则应当使用这个特性，因为 IE 浏览器和 Netscape 浏览器处理空单元格的方式不同。

下面的表具有两个空单元格：一个空<th>元素和一个空<td>元素(ch08_eg17.html)：

```
<table>
  <tr>
    <th></th>
    <th>Title one</th>
    <th>Title two</th>
  </tr>
  <tr>
    <th>Row Title</th>
    <td>value</td>
    <td>value</td>
  </tr>
  <tr>
    <th>Row Title</th>
    <td>value</td>
    <td></td>
  </tr>
</table>
```

在下面的代码中，empty-cells 特性用于隐藏<table>元素中空单元格的边框(ch08_eg17.css)：

```
table {
  background-color:#efefef;
  width:350px;
  border-collapse:separate;
  empty-cells:hide;}
td {padding:5px;
  border-style:solid;
  border-width:1px;
  border-color:#999999;}
```

图 8-18 给出了这个表的显示效果，其中空单元格没有边框。

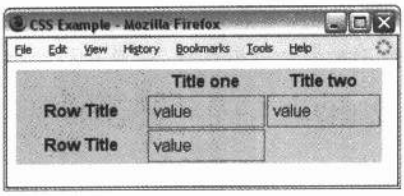


图 8-18

是否使用这个特性是个人偏好问题，如果指定没有边框，当然也不会有任何问题。

8.4.6　table-layout 特性

table-layout 特性用于帮助控制浏览器如何显示或者布局一张表(尽管浏览器对该特性的支持有限)。这个特性可以采用表 8-14 中的 3 种可能值。

表 8-14

值	目　的
fixed	浏览器将利用为某列指定的第一个宽度来计算布局(前提是给出了该宽度)，并使用该宽度计算该列中所有其他单元格的宽度。如果表很庞大并且设计人员为第一行指定了宽度，则这样做将加速表的显示
auto	浏览器在显示表之前查看每一个单元格，然后基于所有单元格的设置计算表的大小。这种显示方式较慢，但是如果不知道每一列的确切大小，则这种方式就非常有用。这是默认值
inherit	将服从包含它的表的规则(仅用于嵌套表)

除非表非常庞大或者包含很多会降低加载速度的图像，否则应当避免使用这个特性。

一些其他的特性(本章中没有讨论它们)可用于为元素组指定规则，但是浏览器对这些特性的支持并不完整。这些特性包括：

- IE5 和之后的浏览器版本支持的 table-header-group 特性和 table-footer-group 特性。
- Netscape 6 和 Firefox 浏览器支持的 inline-table、table-row、table-column-group、table-column、table-row 和 table-cell 等特性。

8.5　外边框

外边框与上一章中介绍的边框相似，但是有两点关键的区别：

- 外边框不占用空间。
- 外边框不必是矩形。

外边框特性的思想是，页面开发人员可能希望向用户强调页面的某些方面；该特性将允许开发人员实现该功能，而不影响页面的流程(在何处定位元素)，因为外边框不像物理边框那样占用空间。外边框样式几乎是在显示页面之后才出现在页面之上。

注意：

Internet Explorer 7 浏览器不支持外边框特性，但虚线框特性能够工作于 Firefox 浏览器。

表 8-15 中列举了 4 个外边框特性。

表 8-15

特　性	目　的
outline-width	指定外边框的宽度
outline-style	指定外边框的线样式
outline-color	指定外边框的颜色
outline	以上特性的简写形式

注意，外边框的所有边都是相同的，不能为该元素的不同边指定不同的值。

8.5.1 outline-width 特性

outline-width 特性指定将添加到框内的外边框的宽度，它的值必须是长度值或者 thin、medium、thick 中的某个值——这一点类似于 border-width 属性。其语法如下：

```
input {border-width:2px;}
```

8.5.2 outline-style 特性

outline-style 特性指定外边框线的样式(solid、dotted 或 dashed)，这些线将环绕着框。它的值必须是第 7 章中介绍的 border-style 特性所使用的某个值。例如：

```
input {outline-style:solid;}
```

8.5.3 outline-color 特性

outline-color 特性用于指定外边框的颜色，它的值必须是颜色名、十六进制颜色或者 RGB 值，与第 7 章中介绍的 color 特性和 border-color 特性相同。例如：

```
input {outline-color:#ffoooo;}
```

8.5.4 outline 特性(简写形式)

outline 特性是一种简写形式，它可用于指定前面讨论的 3 种特性中任意一种特性的值，并且可以采用任意顺序。本章的下载代码中包含了以下示例(ch08_eg18.css)：

```
input {outline: #ff0000 thick solid;}
```

该示例的显示效果如图 8-19 所示。

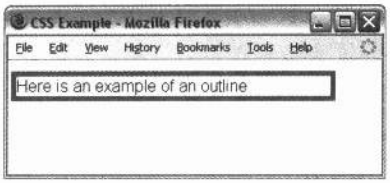


图 8-19

上面讨论的几种外边框特性可以用于:focus 伪类和:active 伪类(下一节中将介绍这两个伪类)，以指示哪个元素当前被激活或者具有焦点。

8.6　:focus 伪类和:active 伪类

本书在第 5 章中讨论了焦点。如果用户将要与某个元素交互，则该元素需要能够获得焦点；一般来说，这种元素是表单控件和链接。

当一个元素获得焦点时，它是在外观上趋向于与其他元素稍有不同。:focus 伪类可用于在元素获得焦点时将一些额外的规则与该元素相关联，而:active 伪类可用于当元素被激活(例如用户单击链接)时将更多的样式与这些元素相关联。

在下面的示例中，当任何<input>元素获得焦点之后，规则将会在<input>元素周围添加红色边框(ch08_eg19.css)：

```
input:focus {outline:#ff0000 thick solid;}
```

但是，:focus 伪类不被 IE7 浏览器(或者 Mac 计算机中的 Firefox 2 浏览器)所支持。图 8-20 给出了在 PC 计算机的 Firefox 浏览器中，当具有该样式的文本输入框获得焦点时的外观。

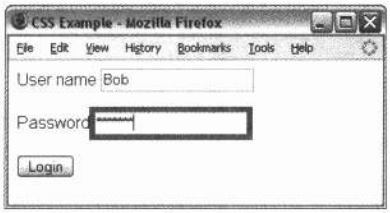


图 8-20

这些伪类将使用户在填写表单时能够知道自己当前正在填写哪一项。

8.7　生成的内容

CSS2 中引入了向 XHTML 文档中添加内容的强大方式——这些内容不是被赋予样式的初始 XHTML 文档的一部分。这些内容仅能够出现在某个选择器指定的、元素之前或者之后，并且该元素带有:before 或:after 伪元素。然后，将 content 特性用于这些伪元素，以指定哪些内容应该插入到文档中。

Netscape 6 浏览器和以后的版本以及 IE7 浏览器对:before 和:after 伪元素的支持有限，但是 Firefox 浏览器很好地支持这些伪元素。

8.7.1 :before 和:after 伪元素

使用:before 和:after 伪元素能够将文本添加到某个选择器中定义的单个或多个元素的每一个实例之前或之后。例如，下面的 CSS 规则将单词“You need to register to read the full article”添加到附带 class 属性并且该属性值为 abstract 的<p>元素的每一个实例之前(ch08_eg20.css)：

```
p.abstract:after {content: "You need to register to read the full
  article.";
  color:#ff0000;}
```

在其中可以看到，:after 伪元素用于一个选择器之后，该选择器标识:after 伪元素应用于哪一个元素。然后在声明内可以看到 content 特性；双引号内的文本将添加到指定元素的末尾。content 特性可以向文档中添加多种类型的内容，而不只是文本，下一节中将对此进行介绍。

如果不在规则中添加其他声明，则采用父元素的默认样式，但是在这个示例中添加的内容以红色编写。可以在图 8-21 中看到这个伪元素的使用效果。

图 8-21

IE7 浏览器支持:before 和:after 伪元素的第一个 Internet Explorer 版本。

注意：

默认情况下，使用这些伪类创建的元素将是内联的，除非使用具有 block 值的 display 特性，但是如果选择器中标识的元素是一个内联元素，则不能使用具有 block 值的 display 特性。

8.7.2 content 特性

当 content 特性用于:before 和:after 伪元素中时，它指示哪些内容应当添加到文档中。表 8-16 中列举了该特性能够采用的值，每一种值将不同类型的内容添加到被赋予样式的 XHTML 文档中。

表 8-16

值	目　的
字符串	为了插入纯文本，其中可能不包含双引号，因此不能包含附带属性的 XHTML 标记(术语“字符串”指一系列的字母数字字符，而不是 CSS 特性)
URL	该 URL 能够指向将包含在此处的图像、文本文件或者 HTML 文件
计数器	用于页面中元素的编号计数器(下一节中将讨论)
atrr(x)	该元素附带的 x 属性的值(这对于非 XHTML 的语言来说非常有用)
open-quote	插入适当的左双引号(请查看本章后面的“双引号”一节)
close-quote	插入适当的右双引号(请查看本章后面的“双引号”一节)
no-open-quote	不使用任何左双引号
no-close-quote	不使用右双引号(通常用于乏味的讲话中某个人说了很长时间的话时，并且该样式指示仅在最后一个段落中使用右双引号)

1. 计数器

本书前面已经介绍如何在页面中添加编号列表，因此自动编号并不是一种新概念。但是，counter()函数设计用于创建计数器，每次浏览器遇到任何指定的元素(不只是<li>元素)时，计数器都会自动递增。

如果希望创建文档的自动编号分部，并且该分部不会作为排序列表(使用<ol>或 元素创建)的一部分出现，则这种思想就非常有用。

为了解 counter()函数的用法，查看下面的示例 XHTML 文本段落(ch08_eg21.html)：

```
<body>
<h1> Introducing Web Technologies</h1>
  <h2>Introducing HTML</h2>
  <h2>Introducing CSS</h2>
  <h2>Introducing XHTML</h2>
<h1> Structure of Documents</h1>
  <h2>Text</h2>
  <h2>Lists</h2>
  <h2>Tables</h2>
  <h2>Forms</h2>
</body>
```

这个示例包含两个计数器：一个称为 chapter，另一个称为 section。每次遇到一个<h1>元素时，chapter 计数器将递增 1；每次遇到一个<h2>元素时，section 计数器将递增 1。

此外，每次浏览器遇到一个<h1>元素时，它将在<h1>元素的内容之前插入单词 Chapter 和计数器中的数值。同时，每次浏览器遇到一个<h2>元素时，它将显示 chapter 计数器的数值，后面跟上一个句点以及 section 计数器的值。

其运行结果将如图 8-22 所示。

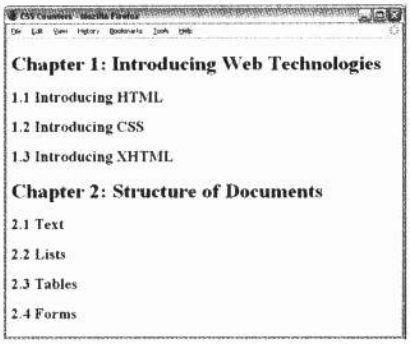


图 8-22

现在查看这个示例的工作原理。首先需要注意的是，在开始操作之前，在<body>元素中使用 counter-reset 特性将 chapter 和 section 计数器设置为 0.

然后是一些使用:before 伪类的 CSS 规则，它们用于自动插入章节的自动编号。

最后，counter-increment 特性在每次遇到指定的元素时将计数器递增 1(ch08_eg21.css)：

```
body {counter-reset: chapter; counter-reset: section;}
h1:before {content: "Chapter " counter(chapter) ": ";}
h1 {counter-increment: chapter; counter-reset: section;}
h2:before { content: counter(chapter) "." counter
(section) " "; }
h2 {counter-increment: section; }
```

2. 双引号

值 open-quote 和 close-quote 可以用于 content 特性，以便在出现的指定元素之前或者之后添加双引号。

遗憾的是，这些功能不被 IE7 浏览器所支持，但是可以查看 Firefox 浏览器中的示例。首先，下面是示例的 XHTML 文本段落(ch08_eg22.html)：

```
<h1>Generated quotes</h1>
<p>Here are some quotes from Oscar Wilde:</p>
<blockquote>Consistency is the last refuge of the
unimaginative.</blockquote>
<blockquote>If you want to tell people the truth, make them laugh, otherwise
they'll kill you.</blockquote>
<blockquote>It is a very sad thing that nowadays there is so little useless
information.</blockquote>
```

现在使用下面的 CSS 语句在<blockquotes>元素之前和之后添加双引号(ch08_ eg22.css)：

```
blockquote:before {content: open-quote;}
blockquote:after {content: close-quote;}
```

图 8-23 给出了该示例的结果。

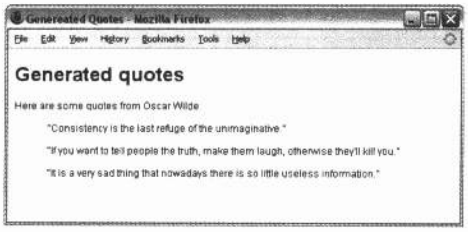


图 8-23

8.8　其他特性

应当了解一些其他特性，下面的章节中将介绍这些特性：

- cursor 特性；
- display 特性；
- visibility 特性。

8.8.1　cursor 特性

cursor 特性用于指定显示应该给用户的鼠标光标类型。当使用图像作为表单上的提交按钮时通常实现这个特性。默认情况下，当用户将鼠标指针悬停在一个链接上时，光标将从指针形状改变为手状形状。但是，对于表单上的提交按钮不会发生这种情况。因此，可以使用 cursor 特性，在用户将鼠标指针悬停在作为提交按钮的图像上时将光标的形状改为手状。这为用户提供一种可视化的暗示，表明他们可以单击该图像。

表 8-17 给出了 cursor 特性的可能值。

表 8-17

值	说　明
auto	光标的形状取决于它所悬停的上下文区域(遇到文本时显示 I 形状，遇到链接时显示手状形状等)
crosshair	显示十字准线或者加号
default	通常显示箭头
pointer	显示指示方向的手(在 IE4 浏览器中，这个值是 hand)
move	显示握着的手(表明正在拖放 DHTML)
e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize	表明可以移动某条边。例如，如果利用鼠标伸展图像的框，则 se-resize 光标用于指示从框的东南角开始移动

(续表)

值	说　明
text	显示竖线 I
wait	显示沙漏
help	显示问号或者气球，表明鼠标指针悬停在帮助按钮上
<url>	光标图像文件的来源

注意：

必须是在能够为用户添加有用的信息时才使用这些值，并且是在用户期望看到光标的添加这些信息。例如，将鼠标指针悬停在链接上时显示十字准线，则会混淆访问者。

8.8.2 display 特性

display 特性将元素(或者框)强制为与用户期望类型不同的框。您可能已经注意到，在上一章中使用该特性使内联框看上去类似于块框。

```
display:block;
```

该特性也可以采用值 inline，以使传统的块级框成为内联框。对于这个特性，除了这个值之外，可能希望使用的其他唯一值是 none，该值用于表明不显示框(并且框对页面的布局不起任何效果——如同其完全不在标记中)。

除了这些用法之外，可能没有其他原因需要使用这个特性。这个特性能够采用的其他值用于非 XHTML 的语言。

8.8.3 visibility 特性

visibility 特性可用于隐藏框，使其不可见，但是框仍然会影响页面的布局(即使看不到框的内容)。可以选择使用 visibility 特性隐藏错误信息以便仅在用户需要看到这些内容时才显示，或者隐藏测试的答案直到用户选择了某个选项。

visibility 特性可以采用表 8-18 中列举的值。

表 8-18

值	目　的
visible	向用户显示框和它的内容
hidden	框和它的内容不可见，但是这些内容仍然影响页面的布局
collapse	该值仅用于动态的表列和行效果，本章不介绍这方面的内容，因为超出了本书的介绍范围，并且当前对它们的支持有限

例如，下面的 4 个文本段落(ch08_eg23.html)：

```
<body>
  <p>Here is a paragraph of text.</p>
  <p>Here is a paragraph of text.</p>
  <p class="invisible">This paragraph of text should be invisible.</p>
  <p>Here is a paragraph of text.</p>
</body>
```

注意，第三段具有一个 class 属性，它的值表明其是不可见类的一部分。下面查看这个类的规则(ch08_eg23.css)：

```
p.invisible {visibility:hidden;}
```

在图 8-24 中可以看到，不可见段落仍然占用空间，但是它对用户不可见。

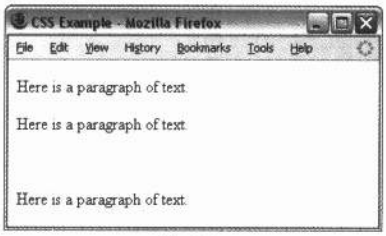


图 8-24

注意，源代码仍然包含不可见段落中的内容，因此不能使用这种方法隐藏敏感信息，例如信用卡详情或者密码。

8.9　额外的规则

在开始查看如何使用 CSS 定位页面中的元素之前，考虑如下 3 个规则：

- @import 在当前的样式表中导入另外一个样式表。
- @charset 指示样式表使用的字符集。
- !important 指示用户自定义的规则比作者的样式表具有更高的优先级。

8.9.1　@import 规则：模块化的样式表

@import 规则可用于从另外一个样式表中导入样式，它应当出现在样式表的起始位置以及任何规则之前，并且它的值是一个 URL。可以通过以下两种方式之一编写该规则：

```
@import "mystyle.css";
@import url("mystyle.css");
```

每一种方式都能够工作良好。@import 规则的重要性在于，它能够使开发人员以模块化的方法开发样式表。开发人员可以为站点的不同方面创建独立的样式表。在上一章中为代码样式创建样式表时引入了这一概念。现在如果希望在编写的任何其他样式表中包含这些样式，而不是重复地编写它们，则只需要使用@import 规则将这些规则导入正在编写的

样式表。

下面是一个相关的示例，该示例中的样式表导入了上一章中的 codeStyles.css 样式表(为了方便起见，该文件被复制到存放本章下载代码的文件夹中)。这个示例是 ch08_eg24.css：

```
@import "codeStyles.css"
body {
   background-color:#ffffff;
   font-family:arial, verdana, helvetica, sans-serif;}
h1 {font-size:24pt;}
```

该样式表本身不包含很多规则；代码样式全部来源于导入的样式表中。图 8-25 给出了使用这个样式表(该样式表主要包含关于代码的样式)的页面(ch08_eg24.html)。

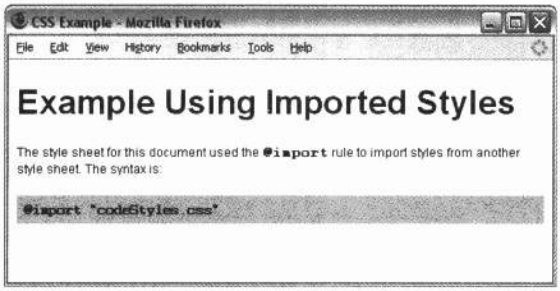


图 8-25

您可能也考虑过开发一个表单样式表，用于任何使用很多表单的页面或者站点。采用以上介绍的方式能够为表单控件创建更漂亮的外观。

8.9.2 @charset 规则

如果编写的文档使用的字符集不是 ASCII 或 ISO 8859-1，则可能希望在样式表的顶端设置@charset 规则，以指示样式表所使用的字符集。

@charset 规则必须出现在样式表的起始位置，在它的前面甚至不能出现空格。它的值位于双引号中，并且必须是本书后面附录 C 中指定的一种语言编码。

```
@charset "iso-8859-1"
```

8.9.3 !important 规则

CSS 以及样式与内容分离的部分目标是使文档对视力受损的人员具有更好的可访问性。因此，在花费宝贵的时间学习 CSS 以及如何编写样式表以使站点更具有吸引力之后，必须说明的是用户也能够创建自己的样式表！

事实上，很少有用户在浏览页面时创建自己的 CSS 样式表，以便以他们想要的方式查看页面，但是该功能确实存在，主要为浏览页面不便的人员设计。默认情况下，您的样式表不一定是用户希望看到的；但是，用户的样式表能够包含!important 规则，它表明“重写站点中关于该特性的样式表。”例如，用户可以使用如下规则：

```
p {font-size:18pt !important;
  font-weight:bold !important;}
```

您无法强迫用户使用您的样式表，并且事实上只有很少(如果有的话)的访问者将创建自己的样式表，因此不需要担心这个规则——这里之所以介绍它，是为了帮助理解该规则是什么以及可能使用它的原因。

注意：

在 CSS1 中，!important 规则允许作者否决用户的样式表，但是在第二版 CSS 中取消了该功能。

8.10　CSS 的定位功能

到目前为止，您已经学习在 CSS 中如何利用框表示每个元素的内容，并且了解到可以使用很多特性影响框和它的内容的外观。现在该了解如何控制元素的内容出现在页面中的位置，方式是指定将框定位在页面中的哪个位置。

在 CSS 出现之前，通常使用表精确控制页面中内容的位置，并且内容以它们在 XHTML 文档中的出现顺序显示。但是，通过使用 CSS 进行定位，可以不使用表布局页面，并且显示信息的顺序可以与它们在 XHTML 文档中的出现顺序不同。

在 CSS2 中，存在 3 种类型的定位方式，可利用它们控制页面的布局：普通定位、浮动定位和绝对定位。在下面的章节中，将介绍如何使用每一种定位方案指示元素内容在页面中出现的位置。

注意：

遗憾的是，您仍然将频繁地看到使用表定位元素在页面中的位置——第 9 章中将更深入地介绍页面布局方面的内容。但是，当前存在使用 CSS 进行定位的强烈趋势，它能够使页面的内容具有更好的可重用性。一旦页面的布局依赖于使用表，则通常页面将仅限于显示在最初为其设计的媒体上。随着更多具有不同功能的设备访问 Internet，您很可能会更多地看到使用 CSS 进行定位，它允许布局随着屏幕进行缩放，并允许不同的样式表用于同一个文档。

8.10.1　普通流

默认情况下，通过使用称为普通流的技术在页面中布局元素。在普通流中，页面中的块级元素从上到下排列(记住，每一个块级元素将出现在一个新行中)，而内联元素将从左到右排列(因为它们不需要从新行开始)。

例如，每一个题头和段落应该出现在一个不同的行中，而<b>、和等元素的内容将位于一个段落或者其他块级元素中；它们不从新行开始。

图 8-26 中的 3 个段落演示了这一点，每一个段落是一个块级元素，从上到下依次排列。在每一个段落内部是内联元素的示例，在该示例中是<b>元素(ch08_eg25.html)。

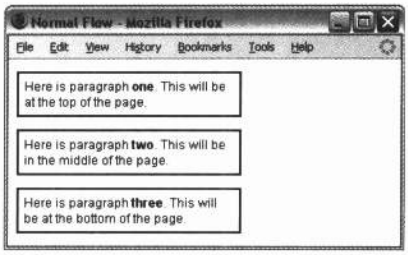


图 8-26

如果希望让元素内容的位置与在普通流中出现的位置不同，可以使用其他两个特性：position 和 float。

8.10.2 position 特性

position 特性可用于为框指定一个位置，它可以采用表 8-19 中列举的 4 个值。

表 8-19

值	意　义
static	该值的功能与普通流相同，并且是默认值，因此您将很少看到指定该值
relative	框的位置可以与它在普通流中的位置有所偏移
absolute	框被精确地定位在包含它的元素的位置中，方式是使用从包含元素的左上角开始的 x 和 y 坐标
fixed	位置从一个固定点开始计算：对于浏览器，这个点是浏览器窗口的左上角，如果用户滚动窗口，不会改变该位置

在后续的章节中将依次介绍这些值的用法。

8.10.3 框偏移特性

本章后面将会介绍，当框具有一个 position 特性并且该特性的值是 relative、absolute 或 fixed 时，则将同时使用框偏移特性来指示这些框的位置。表 8-20 中列举了一些框偏移特性。

表 8-20

特　性	意　义
top	偏移位置从包含它的元素的顶部开始
right	偏移位置从包含它的元素的右边开始
bottom	偏移位置从包含它的元素的底部开始
left	偏移位置从包含它的元素的左边开始

每一个特性都可以采用长度、百分比或者 auto 值。相对单位(包括百分比)通常以相对于包含它的框的面积或者特性进行计算。

8.10.4　相对定位

相对定位将元素相对于它在普通流中的位置进行定位。元素与它在普通流中的位置的偏移量由框偏移特性指定。

下面回顾前面关于普通定位的章节中遇到的示例，但是这一次使用相对定位重新定位第二个段落，如图 8-27 所示。

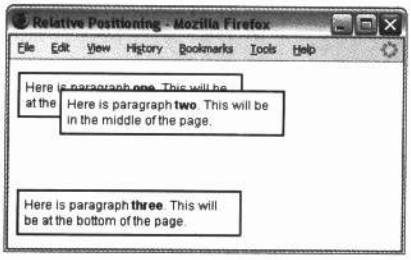


图 8-27

在这个示例中，第二个段落与它在普通流中的位置(即在上一个示例中的位置)有所偏移，左边偏移 40 个像素并且顶部偏移 40 个像素——注意下面代码中的负号，它将段落在普通流中的位置向上提升(ch08_eg26.css)。

```
p {border-style:solid;
  border-color:#000000;
  border-width:2px;
  padding:5px;
  background-color:#FFFFFF;}
p.two {
  position:relative;
  left: 40px;
  top: -40px;}
```

框偏移特性(在这个示例中是 top 和 left)的值可以是长度、百分比或者 auto。如果值是绝对长度，则可以是负值。

注意：

在左偏移和右偏移之间只能选择一种偏移，在顶部偏移和底部偏移之间也只能选择一种偏移。如果同时指定了左偏移和右偏移，或者同时指定了顶部偏移和底部偏移，则其中一个值必须是另外一个值的负值(例如，top:3px; bottom:-3px;)。如果同时具有顶部偏移和底部偏移，或者同时具有左偏移和右偏移，并且其中一个值不是另外一个值的负值，则右偏移或者底部偏移将被忽略。

当使用相对定位时，某些框最终可能会相互重叠，如同前面的示例所示。因为将框相对于普通流中的位置偏移，所以如果偏移量足够大，则其中一个框最终将位于另外一个框之上。这可能会创建期望的效果；但是，应当注意其中的一些缺陷：

- 除非为框设置背景(背景颜色或者图像)，否则默认情况下它将是透明的，这将使重叠文本不容易被识别。在前面的示例中使用 background-color 特性将段落的背景色设置为白色，从而防止发生这种情况。
- 当相对定位的元素互相重叠时，CSS 规范没有表明哪个元素应当出现在上方，因此不同浏览器中的处理方式不同(但是可以使用 z-index 特性控制这种情况，本章后面将介绍该特性)。

8.10.5 绝对定位

绝对定位将元素的内容从普通流中完全移除，并且可以固定该元素的位置。

可以指定一个元素内容的定位方式为绝对定位，方式是赋予它一个 position 特性，并且该特性的值为 absolute；然后使用框偏移特性将其定位到所需的位置。

框偏移特性将元素的位置固定在相对于包含块的位置——包含块是一个元素，其 position 特性被设置为 relative 或者 fixed。

查看下面的样式表。这个样式表再次用于 3 个段落，但是这一次这些段落包含在一个 元素中，该元素也使用了绝对定位(ch20_eg27.css)：

```
div.page {
  position:absolute;
  left:50px;
  top: 100px;
  border-style:solid; border-width:2px; border-color:#000000;}
p {
  background-color:#FFFFFF;
  width:200px;
  padding:5px;
  border-style:solid; border-color:#000000; border-width:2px;}
p.two {
  position:absolute;
  left:50px;
  top: -25px;}
```

图 8-28 给出该示例在浏览器中的外观；从该图中可以清晰地看到，第二段不再位于页面的中间。第二段的元素已经被排除在普通流之外，因为第三段现在放置在第二段原来在普通流中的位置(如果第二段参与普通流的话)。此外，第二段甚至出现在第一段之前，并且偏向于页面的右边！

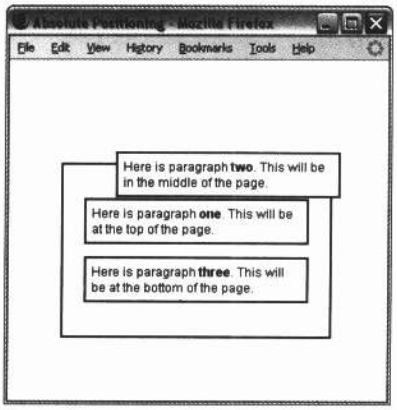


图 8-28

这里使用<div class"page">元素表明该段落相对于包含块进行定位——包含块为绝对定位元素 。

绝对定位元素总是出现在相对定位元素之上，如图 8-28 所示，除非使用了 z-index 特性(本章后面将介绍 z-index 特性)。

需要注意的是，因为绝对定位框被排除在普通流之外，即使两个垂直页边空白相遇，它们也不会产生折叠。

8.10.6　固定定位

对于 positioning 特性，最后一个需要注意的值是 fixed。这个值指定元素的内容不仅从普通流中完全移除，而且当用户向下滚动页面时框不移动。

Firefox 浏览器和 Safari 浏览器已经提供了对固定定位的支持，Internet Explorer 浏览器从 IE7 版本开始支持该定位方式。

下面的 XHTML 文档示例演示了固定定位(ch08_eg28.html)。这个示例具有多个连续的段落，以便当页面向下滚动时可以看到 元素的内容固定在页面的顶部：

```
<div class="header">Beginning Web Development</div>
<p class="one">This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls.</p>
```

下面是这个示例的样式表(ch08_eg28.css)。header 元素中有一个值为 fixed 的 position 特性，它被定位在浏览器窗口的左上角位置：

```
div.header {
  position:fixed;
  top: 0px;
  left:0px;
  width:100%;
  padding:20px;
  font-size:28px;
```

```
    color:#ffffff; background-color:#666666;
    border-style:solid; border-width:2px; border-color:#000000;}
  p {
    width:300px;
    padding:5px;
    color:#000000; background-color:#FFFFFF;
    border-style:solid; border-color:#000000; border-width:2px;}
  p.one {margin-top:100px; }
```

最后一条规则使第一段从页面的顶部下降，以便能够看到它，但是，在当前的上下文中该规则不起作用。

图 8-29 给出这个固定的题头(header)元素的显示效果，即使用户向下滚动页面，该元素仍然显示。

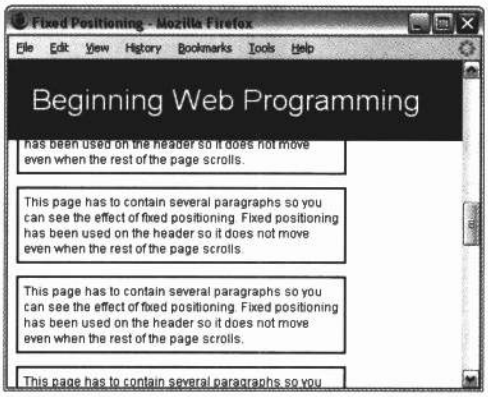


图 8-29

8.10.7 z-index 特性

绝对定位元素具有重叠其他元素的趋势。当发生重叠时，默认的处理方式是第一个元素位于后面的元素下方。这称为堆叠上下文。如果具有一些绝对定位或者相对定位的框，可以使用 z-index 特性修改堆叠上下文来控制哪些框出现在上方。如果熟悉图形设计程序包，则堆叠上下文类似于使用“带到顶部”和“发送到底部”功能。

z-index 特性的值是一个数值，数值越大，则元素的显示就会越接近顶部。

为了更好地理解 z-index 特性，查看另外一个绝对定位示例——这一次仅有 3 个段落：

```
    <p class="one">Here is paragraph <b>one</b>. This will be at the top of the
    page.</p>
    <p class="two">Here is paragraph <b>two</b>. This will be underneath the
other elements.</p>
    <p class="three">Here is paragraph <b>three</b>. This will be at the bottom
of the page.</p>
```

这些段落共享公共的 width、background-color、padding 和 border 特性。然后使用绝对

定位方式单独定位每个段落。因为这些段落都是重叠的，这里添加了一个 z-index 特性来控制哪个段落出现在顶部；该特性的值越大，则相应的段落越接近于顶部(ch08_eg29.css)：

```
p {
  width:200px;
  background-color:#ffffff;
  padding:5px; margin:10px;
  border-style:solid; border-color:#000000; border-width:2px;}
p.one {
    z-index:3;
    position:absolute;
    left:0px; top:0px;}
p.two {
  z-index:1;
  position:absolute;
  left:150px; top: 25px;}
p.three {
    z-index:2;
    position:absolute;
    left:40px; top:35px;}
```

该实例的显示效果如图 8-30 所示，第二段现在出现在第一段和第三段的下方，第一段仍然位于顶部。

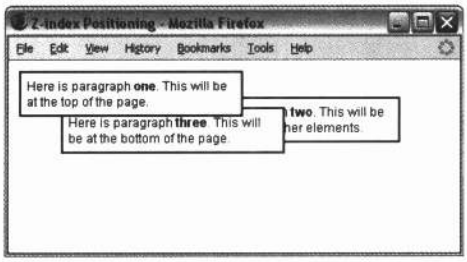


图 8-30

8.10.8　使用 float 特性浮动

float 特性可将元素排除在普通流之外，并且将它尽可能地放置在包含框的左边或者右边，但是位于包含框的内边距之内。该元素的垂直页边空白不会与上方或下方元素的页边空白发生折叠，如同普通流中的块框一样(因为它已经被排除在普通流之外)；浮动框将与包含框的顶部对齐。

为了指示框浮动在包含框的左边或者右边，可以设置 float 特性，该特性可以采用表 8-21 中列举的某个值。

表 8-21

值	目　的
left	框浮动在包含元素的左边，而包含元素的内容将流动到它的右边
right	框浮动在包含元素的右边，而包含元素的内容将流动到它的左边

(续表)

值	目　的
none	框不浮动，将位于其在普通流中的位置处
inherit	框将采用与它的包含元素相同的特性

指定 float 特性之后，也必须设置 width 特性，用于指示浮动框占用包含框的宽度；否则，浮动框将自动占用 100%的包含框宽度，不为环绕它的其他内容留下任何空间，从而使其类似于块级元素。

查看下面的 XHTML 文档(ch08_eg30.html)，并且注意第一段起始位置中的元素：

```
<body>
  <h1>Heading</h1>
  <p><span class="pullQuote">Here is the pullquote. It will be removed from
  normal flow and appear on the right of the page.</span>
  Here is paragraph <b>one</b>. This will be at the top of the page. Here is
  paragraph <b>one</b>. This will be at the top of the page. Here is paragraph
  <b>one</b>. This will be at the top of the page. Here is paragraph
  <b>one</b>. This will be at the top of the page. Here is paragraph <b>one
  </b>. This will be at the top of the page.</p>
  <p>Here is paragraph <b>two</b>. This will be at the bottom of the page.</p>
</body>
```

甚至内联的元素也能够浮动，使其脱离它的包含元素(不只是块级框能够浮动)。该元素将被排除在普通流之外，并被放置到包含元素<p>的右边，因为这里使用具有 right 值的 float 特性(ch08_eg30.css)。

```
p {
  border-style:solid;
  border-color:#000000;
  border-width:2px;
  padding:5px;
  background-color:#FFFFFF;
  width:500px;}
.pullQuote {
  float:right;
  width:150px;
  padding:5px;
  margin:5px;
  border-style:solid;
  border-width:1px; }
```

在图 8-31 中可以看到，class 属性值为 pullQuote 的元素的内容位于页面的右边，而剩余的段落流动到左边，然后位于该元素的下方。

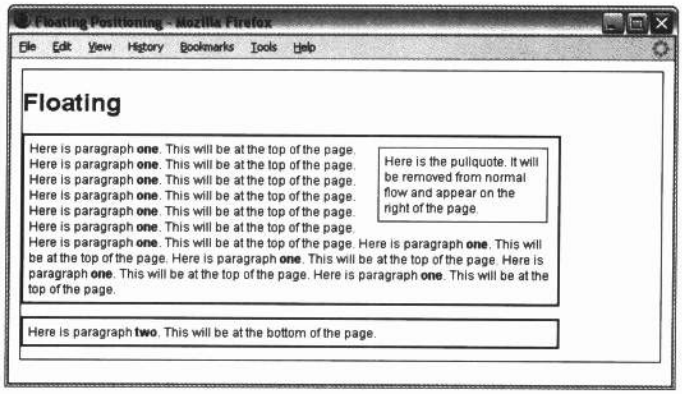


图 8-31

在下一章中讨论页面布局时将介绍更多 float 特性工作原理的示例。

8.10.9　clear 特性

当使用浮动框时，clear 特性特别有用。如图 8-31 所示，内容可以围绕浮动元素进行布局；但是，开发人员可能不希望发生这种情况——他们可能宁愿浮动元素的左边没有任何内容，并且将包围内容推到浮动元素的下方。clear 特性可以完成该功能，表 8-22 中给出了这个特性能够采用的值。

表 8-22

值	目　的
left	具有 clear 特性的元素的内容将从浮动元素的左边清除(它不会出现在浮动元素的左边)
right	具有 clear 特性的元素的内容将从浮动元素的右边清除(它不会出现在浮动元素的右边)
both	具有 clear 特性的元素的内容将从浮动元素的两边清除(它不会出现在浮动元素的任意一边)
none	允许在任意一边浮动

查看下面的示例(ch08_eg31.html)，它与上面的示例稍有不同，因为这一次 PullQuote 位于 元素而不是元素中，并且它不包含在段落中：

```
<h1>Floating</h1>
  <div class="pullQuote">Here is the pullquote. It will be removed from
    normal flow and appear on the right of the page.</div>
  <p>Here is paragraph <b>one</b>. This paragraph will get pushed underneath
the floating element. </p>
```

下面的样式表将把 PullQuote 元素浮动到右边，但是段落元素使用 clear 特性阻止任何浮动内容出现在它的右边(ch08_eg31.css)：

```
p {
  clear:right;
  background-color:#FFFFFF; }
div.pullQuote {
  float:right;
  padding:5px;
  margin:5px;
  width:150px;
  border-style:solid; border-width:1px; }
```

图 8-32 显示了这个示例中 clear 特性的工作原理。

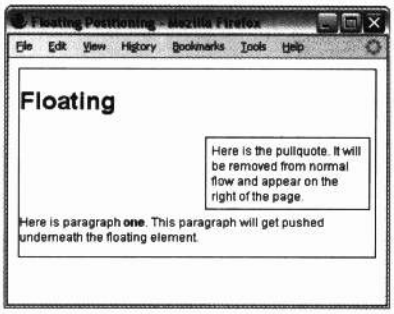


图 8-32

试一试　样本布局

在这个示例中将创建一个样本页面布局，该布局结合使用本章中介绍的一些技术来显示一篇文章，其中使用 CSS 而不是表来布局页面。

本示例中将要操作的无样式表的页面如图 8-33 所示。

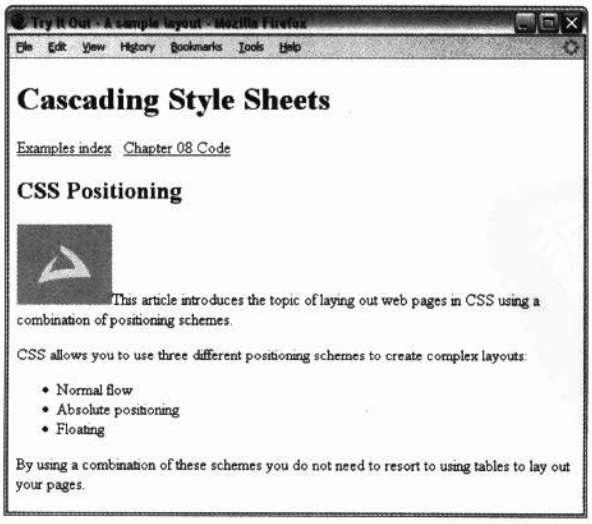


图 8-33

```
<body>
<h1>Cascading Style Sheets</h1>
  <div class="nav"><a href="../index.htm">Examples index</a>
      <a href="">Chapter 8 Code</a></div>
<h2>CSS Positioning</h2>
  <p class="abstract"><img class="floatLeft" src="images/background.gif"
     alt="wrox logo" />This article introduces the topic of laying out
     web pages in CSS using a combination of positioning schemes.</p>
  <p>CSS allows you to use three different positioning schemes to create
    complex layouts:</p>
  <ul>
    <li>Normal flow</li>
    <li>Absolute positioning</li>
    <li>Floating</li>
  </ul>
  <p>By using a combination of these schemes you do not need to resort to
     using tables to lay out your pages.</p>
</body>
```

这个示例说明了 CSS 中需要注意的一些问题——特别重要的是，演示了虽然存在一些非常有用的特性，但是某些特性仅被最新的浏览器所支持。虽然可以通过设计使站点能够工作于大多数浏览器，但可能无法获得一些技术让站点工作于所有需要的浏览器——因此需要全面地测试站点。

在 Firefox 浏览器中，这个“试一试”示例的显示效果如图 8-34 所示；在 IE7 浏览器中将获得类似的结果。

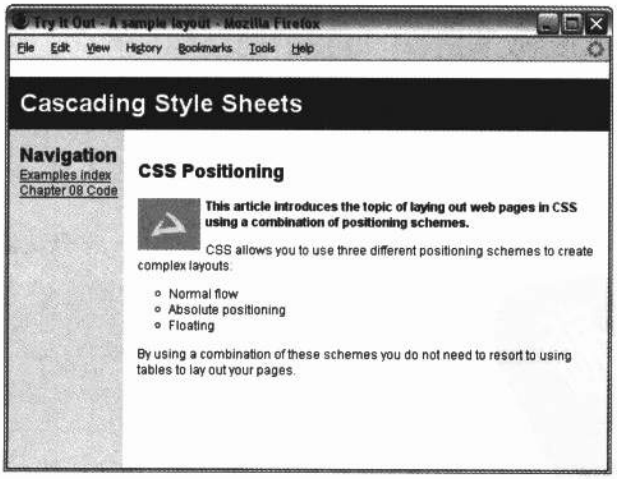


图 8-34

图 8-35 中给出了这个页面在 IE6 浏览器中的显示效果——因此如果仍然有访问者使用 IE6 浏览器访问您的站点，则需要考虑哪些功能可以起作用，哪些功能无效。

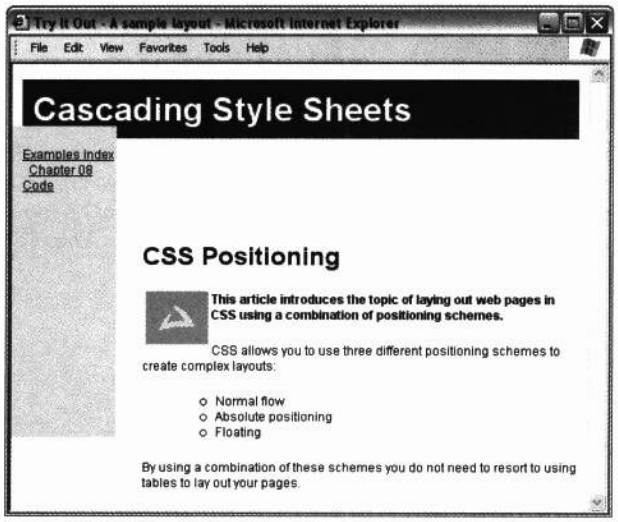


图 8-35

为了开始编写这个页面的 CSS 文件，启动 Web 页面编辑软件，逐步完成以下步骤：

(1) 创建称为 samplePage.css 的文件，添加 XHTML 页面中的元素，并且使用标识每一种类型元素的适当选择器。最终获得类似于如下的列表；然后将能够依次查看每一个元素的规则。

```
body {}
h1 {}
div.nav {}
h2 {}
p {}
p.abstract {}
img {}
ul {}
```

(2) 首先添加<body>元素的规则，该规则仅是设置页面的一些默认值：

```
body {
    color:#000000;
    background-color:#ffffff;
    font-family:arial, verdana, sans-serif;
    font-size:12px;}
```

(3) 接下来是站点的题头，它使用固定定位，即使用户滚动页面，它也将停留在页面的顶部。它也有一个 z-index 特性，用于确保这个题头始终位于导航的上方。

```
h1 {
  position:fixed;
  top:0px; left:0px;
  width:100%;
  color:#ffffff; background-color:#666666;
```

```
  padding:10px;
  z-index:2;}
```

(4) 从普通流中移除导航，因为导航使用的是绝对定位。它被定位在顶部向下 60 个像素处，以便当页面首次加载时链接不会消失在页面题头的下方。导航被放置在一个 100 像素宽、300 像素高并且具有亮灰色背景的框中，它的 z-index 值是 1，这就确保它位于页面题头的下方(题头的 z-index 值是 2)。

```
div.nav {
  z-index:1;
  position:absolute;
  top:60px;
  left:0px;
  width:100px;
  height:300px;
  padding-left:10px; padding-top:20px; padding-bottom:10px;
  background-color:#efefef;}
```

(5) 导航栏包含单词"Navigation"，该单词不在初始 HTML 文档中。这个样式表使用 CSS :before 伪类添加该单词。可以看到，该伪类也有其他关联的样式。

```
div.nav:before {
  content: "Navigation ";
  font-size:18px;
  font-weight:bold;}
```

(6) 接下来是<h2>元素的规则，它需要从左边缩进，因为导航占用其左边的 110 个像素。在该元素的顶部也具有内边距，以便它的文本位于题头下方。

```
h2 {
  padding-top:80px;
  padding-left:115px;}
```

(7) 然后是段落的两条规则，第一条规则用于所有的段落，第二条规则确保文章的摘要以粗体显示。类似于<h2>元素，所有的段落都需要从左边缩进。

```
p {padding-left:115px;}
p.abstract{font-weight:bold;}
```

(8) 位于第一个段落中的图像浮动到文本的左边。该段落中的文本围绕图像布局。图像的右边也有 5 像素宽的内边距。

```
img {
  float:left;
  width:60px;
  padding-right:5px;}
```

(9) 最后是用于无序列表元素的规则，它需要相对于段落或者二级题头进行缩进。该规则也利用 list-style 特性指定了使用的项目符号的样式。

```
ul {
  clear:left;
  list-style:circle;
  padding-left:145px;}
```

(10) 保存该样式表，并在浏览器中加载使用该样式表的 samplePage.html 文件。

工作原理

本章前面已经提到，需要重点注意的是，并不是这个示例中的所有规则都能工作于所有浏览器。但是，在 Firefox 浏览器或 IE7 浏览器中可以很好地展示这个示例，因此下面的讨论集中于在图 8-34 中显示的结果。

从固定的题头开始，<h1>元素被排除在普通流之外，并且固定在浏览器窗口的顶部(注意，题头上方的空白实际上不会显示)。绝对定位元素总是位于顶部，因此如果页面上的其他内容(即导航和段落)不使用内边距来防止它们重叠，题头就会覆盖这些元素。

导航栏不仅使用 padding-top 特性使其位于题头的下方，而且使用了 z-index 特性。z-index 特性确保如果导航和标题重叠，则标题将出现在上方。这里不能依赖 CSS 的像素精确定位，并且如果导航栏出现在上方，则看上去有些奇怪。

题头、段落和组成文章主体的无序列表都必须具有内边距，以便它们离浏览器左边有一段距离，否则它们将与导航重叠(导航已经被排除在普通流之外，因为它采用的是绝对定位)。因此，这些段落和其他位于普通流中的元素必须远离绝对定位元素。

第一个段落内的图像在块级段落容器中浮动，并且具有值为 5 像素的 padding-right 特性，以便文本不与它的右边缘接触。

最后，比起段落或者题头，无序列表必须更加远离左边的页边空白一段距离，因为它的文本开始位置的左边具有项目符号。如果无序列表的 padding-left 特性值与题头或者段落的相同特性值一样是 115，则列表的文本将从左边 115 像素处开始，但是标记符(项目符号)将距离左边更近。因此，将这个特性设置为 145，以便列表比文本具有进一步的缩进。

8.11 本章小结

本章介绍了可用于控制列表、链接、表、外边框和背景的一些 CSS 特性，然后介绍了如何利用 CSS 从样式表向文档添加内容。:before 伪类和:after 伪类可用于在选择器指定的元素之前或者之后添加内容，添加的内容包括文本、图像或者文件中的内容。甚至可以利用 counter()函数自动编号或计数任何元素，并且可以管理复杂的引号集(但是，并不是所有的浏览器都支持这些功能)。

还介绍了如何使用@import 规则将其他样式表中的规则包含到当前样式表中，创建模块化的样式表，以及重用来自于站点的不同部分的规则。另外，@charset 规则指示样式表使用的字符集。

最后，介绍了 CSS 的 3 种主要定位方案：普通流(和它的衍生物相对定位)、绝对定位(和它的衍生物固定定位)、浮动定位。这些是控制文档内容显示位置的强大工具；它们完成了将样式独立于内容的功能，因为不需要使用表来控制文档的布局(下一章中将详细介绍布局

方面的内容)。

如本章中的示例(特别是本章最后的较长示例)所示，即使是在最新的浏览器中，对 CSS 的支持仍然是不完全的。考虑到 CSS2 规范于 1998 年完成，而浏览器制造商仍然没有竭力尝试很好地实现它，这不得不说是一种悲哀。

需要注意的是，虽然可以创建利用 CSS 布局的页面，但是较老的浏览器并不都支持这些布局。因此，某些设计人员结合使用较老的页面布局技术和用于赋予一些样式的 CSS。

8.12　练习

所有练习的答案都在附录 A 中给出。

1. 在这个练习中，需要创建内容的链接表，它位于一个长文档的上方，以有序列表的方式出现并链接到文档主要部分中的题头。

本书的下载代码中提供了 XHTML 文件 exercise1.html，请为该文件创建样式表。该样式表应该执行以下操作：

- 设置所有链接的样式，包括激活的和访问过的链接
- 列表的内容设置为粗体
- 列表的背景设置为亮灰色，并使用内边距确保项目编号能够显示
- 链接框的宽度设置为 250 像素宽
- 将题头项目符号的样式改为空心圆
- 将链接项目符号的样式改为正方形

创建的页面应当如图 8-36 所示。

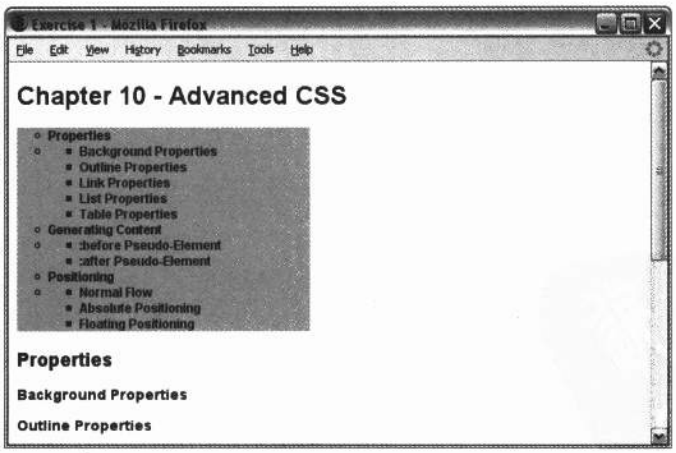


图 8-36

2. 在这个练习中，测试您的 CSS 定位技能。您需要创建一个页面，该页面以不同的方式表示指向本章不同小节处的链接。每一小节将在不同的块中显示，每一个块将被绝对定位在一个从左上到右下方向的对角线上。中间的框必须出现在最上方，如图 8-37 所示。

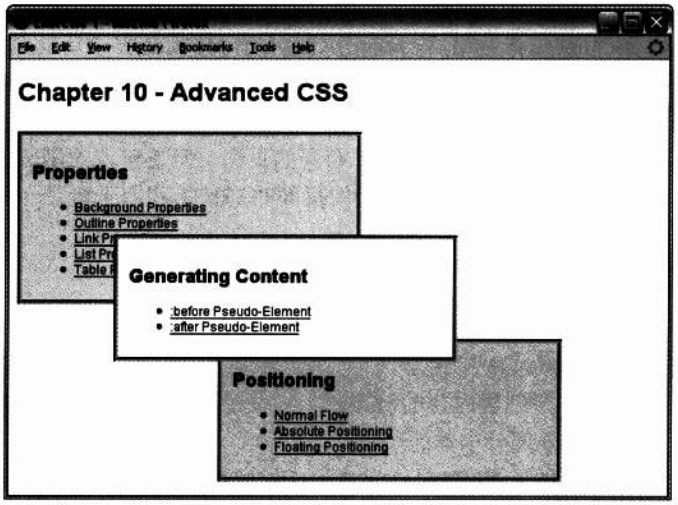


图 8-37

可以在本章的下载代码中找到 XHTML 源文件 exercise2.html。

第 9 章

页 面 布 局

本书使用两章的篇幅介绍设计问题，本章是其中的第一章，介绍如何设计和构造 Web 页面的布局。尽管没有专门介绍如何设计页面的权威书籍，但是在设计页面外观时需要考虑一些重要的因素，本章中将介绍这些方面的知识。

没有人能够告诉您如何使页面更具有吸引力——这是个人偏好问题。在本章中将讨论站点的目标、页面的大小、应当出现在页面上的内容以及每一个项应当位于页面上的什么位置等。您也需要知道如何通过代码实现这些设计。本章大致地分为 4 部分，分别反映如下这些主题：

- 正确理解站点的目标
- 将页面作为整体看待并解决关于如何确定页面大小的问题
- 了解组成每一个页面的元素，例如徽标、题头、链接和可能存在的广告
- 在页面中定位各种元素

学习完本章中页面的整体布局之后，请查看第 10 章，学习关于页面内设计的一些更具体的问题，例如菜单、表单、链接和文本的设计。

9.1 理解站点

无论是打算为自己创建 Web 站点还是希望为客户创建 Web 站点，在开始设计之前必须很好地理解希望创建的站点。首先需要询问一些基础问题，以确保理解站点拥有者的目标以及站点访问者期望在该站点上找到什么内容；特别是，需要确保知道：

- 期望什么样的人访问该站点？
- 认为站点访问者期望在该站点上找到什么类型的信息？
- 期望访问者的实际访问频率是多少？
- 希望站点为访问者做些什么？

如果无法清楚这些问题，您将不能设计出具有吸引力的站点。其中前 3 个问题是关于访问者以及他们希望从站点获得什么，而不是您希望从访问者处获得什么。例如，我可能希望访问者每天都访问站点——但重要的是考虑这是否是符合实际的期望目标。毕竟，如果访问者可能仅是偶尔访问站点，这将对设计(以及页面不同部分的更新频率)带来很大的影响。

下面的内容将在您开始设计站点之前，回答这些问题。

9.1.1 理解站点的目标

在开始设计站点之前，必须确保已经清晰地定义了站点的目标。一个站点可能具有多个目标，但是通过在开始阶段调查一些问题，您将能够正确认识到希望实现的目标。

例如，如果是为一个公司创建站点，则可以询问自己以下问题：

- 正在处理的是产品或服务吗？
- 人们对该产品或服务的需求频率是多少？
- 一旦用户订购了一个产品或者服务，他们有可能返回这个站点并再次订购或者查找更多的信息吗？
- 正在尝试向没有听说过该产品或者服务的人进行促销，还是尝试向知道正在追求的是什么的人更详细地解释该特定产品或者服务？
- 提供的产品或者服务与竞争者的主要区别是什么？
- 是从站点直销还是让访问者与销售人员联系(创建销售向导)，或者告诉人们从哪里能够获得该产品或者服务？

或许正在创建的站点不是为了推出一个产品或服务；您可能正在尝试教授一些新技能；创建一个社区；提供一份简历；创建一个作品夹；发布关于业余爱好或者兴趣的信息；提供打印出版、广播节目、电视节目或者其他形式的媒体；或者发布关于某个主题的意见。每一种类型的站点都具有它们自己的问题集，但重要的是考虑站点真正想要实现的是什么目标，以及为了满足这些目标所需要具有的内容。

如果是为公司创建 Web 站点，则可能会遇到这样的一些客户：他们认为需要一个 Web 站点(或者需要更新他们已有的 Web 站点)，但是无法确定将在站点中放置哪些内容。特别是为小型或中等规模的公司构建站点，在开始创建站点之前通常需要帮助客户了解 Web 站点能够为他们的业务做些什么。

考虑下面一些例子：

摄影师可能需要一个站点作为其工作的作品夹，并且向其他需要摄影服务的人员提供详细的联系方式。

本地运营的小型馅饼公司的主要目标可能是生成销售向导，以便向餐饮行业提供美味的馅饼。

旅馆可能希望展示房间的一些照片，提供关于住宿的详细信息，显示具有康乐信息的位置的地图，以及能够在线预订房间。

科研公司可能希望展示他们当前正在进行的研究的投资方信息、过去的成果和研究团队。

Web 站点的各种目标几乎是无止尽的。但是在开始设计每个新站点时，您需要尝试列举所有的目标；完成该操作之后，可以开始研究如何构造这些信息。

说明：

如果正在为客户设计站点，最好在定义站点的目标时征得客户的同意。在站点开发期间，许多客户可能会希望添加一些额外的功能，因此在开始设计时确定好站点目标是非常

重要的。如果客户在后面希望扩展目标，则可以和他们重新谈判以争取额外的利益(例如额外的开发时间或者额外的费用)。

9.1.2 期望的站点访问者

在整个设计过程中，必须牢记一件事情：需要为站点的目标观众设计页面——而不只是客户。无论是选择在站点的哪个位置放置元素，还是确定是否使用某项技术或者功能(例如声音或动画)，决策的主要依据必须是目标访问者想要的是什么。因此，理解目标访问者自然就非常重要。

遗憾的是，某些公司要求设计人员推出公司领导认为是最有趣的或最重要的消息，而不是将他们自己放在访问者的角度。例如，公司的 Web 站点上很少会出现投资方的信息(例如季度报告或关于董事会的信息)占用页面头版的主要空间。否则，这就是在向访问者暗示该站点的目标是公司的投资方而不是顾客，并且该站点更倾向于赚钱，而不是关注顾客的利益。作为一个顾客，在交出自己辛苦获得的金钱之前不会希望知道这个公司的利润率！相反，站点应当具有到站点某个部分的链接，该部分的目标是站点的少量投资方，而在有价值的头版空间中放置一些顾客感兴趣的链接。

因此需要思考以下问题：

- 谁将访问站点？访问者将是潜在的顾客(公众或其他公司的成员)、投资方、业余爱好人员、报社和媒体或者学生和研究人员吗？
- 访问者来到这个站点的原因是什么？他们是希望购买产品或者服务吗？或者他们查找公司的位置、营业时间或者联系电话/e-mail 吗？他们学习一门新技能吗？他们是查找关于公司、服务或者感兴趣领域的更多信息吗？他们是来确定是否值得投资吗？
- 访问者来到这个站点的潜在动机是什么？他们是为了娱乐(并因此很可能正在浏览网页)，还是为了执行某些操作，例如进行订购或者找出某些信息(此时他们可能希望快速获得结果)？根据这一点和上一点，可以确定期望 50%的访问者是为了一种原因而来，而另外 50%的访问者是为了另外一种原因而来。
- 对于这些访问者，您了解哪些信息？需要统计哪些人对所提供的产品、服务或介绍的主题感兴趣。访问者的年龄、性别和技能等因素都会影响一些设计决策。

9.1.3 新内容

在任何站点的开发周期早期，需要解决的另外一个重要问题是在站点正式运营后拥有者是否愿意花费时间开发和维护站点的新内容。询问这种问题的原因很简单：如果站点的内容不改变，如何期望访问者多次返回到该站点呢？

某些站点，例如包含有用参考信息的站点，可能会被相同的人员访问多次，但是对于大多数小公司的站点——它们仅介绍公司的产品或者服务——将很少会产生大量重新访问站点的人员，除非人们是为了再次订购相同的产品或者服务而返回该站点，或者公司定期发布新产品。

某些站点不需要经常改变；例如，如果您是修缮屋顶的工人，则访问者一旦修理过屋顶之后，他们很可能不会很快返回您的站点(他们起初可能是为了查看您的工作的成功案例，但是一旦修理好屋顶，他们通常不再有兴趣定期访问您的站点)。但是，如果正在为新书或者音乐的发行而创建站点，则很可能会定期有新信息需要宣布，这或许能够吸引访问者每隔几星期都来访问该站点。或者您的期望可能是介于两种情况之间，例如服装站点每年两次展示新的服饰；而对于参考站点，访问者可能仅是偶然才返回该站点。

因此，需要调查期望相同的人重新访问站点的频率。如果希望他们定期返回您的站点，则必须为他们提供有吸引力的内容。

保持内容新鲜的问题在于需要花费很多时间，并且需要某个人负责站点的定期更新。

9.1.4 定义站点的内容

现在您对于站点的目标、站点针对的访问者以及内容的更新频率都有了很好的理解，下面进一步了解站点上出现的实际内容是什么。

当生成站点的潜在内容时，需要像集体讨论会那样进行处理——不要退缩！记住，站点必须是为了满足访问者的需求和访问者期望从站点获得的信息，而不是为了希望他们看到什么内容。

内容列表可以包括：公司提供的产品和服务的相关信息、工作的照片或示例；获得这些产品和服务的方式；联系细节；以及关于公司的信息(对于在 Internet 上进行贸易的小型或中型公司，在交易之前，顾客通常希望知道一些关于它们的背景；这有助于让他们放心交出手中的金钱)。不要忘记进一步地深度探讨；例如，需要包含关于产品和服务的一些什么信息？产品可以包含照片、描述、尺寸、关于它在哪里和如何生产的信息以及它的典型应用等。服务可能需要包含所涉及工作的描述、完成该工作需要的时间、执行服务需要的其他条件、执行服务的人员以及如何评价服务的执行。

注意：

如果是在销售商品，则应当给出它的价格——如果价格是变化的(例如，屋顶修缮人员的收费是根据屋顶的类型和尺寸确定的)，则为产品或服务提供价格向导，比起没有提供价格的站点，这会带来更多的生意机会。

也应该查看解决相似问题的其他站点——竞争对手——了解他们在做什么、哪些方面没有做好以及这些站点是否满足您期望的站点访问者的需求。这里需要考虑的一个关键点是，如何做得不同或者更好——即考虑能够使您的站点比竞争对手更好的方面。

记住，您将希望在大多数页面中添加徽标或者商标以及搜索表单或可能出现的广告等。也需要记住一些令人厌烦但仍然必要的功能，例如版权声明、条款条件、隐私政策(如果正在从用户处收集信息或者使用称为 cookies 的技术存储用户计算机上的信息，则隐私政策将非常重要)。

一旦为顾客准备好他们可能希望从站点处获得的所有可能内容，则可以将您想法的转向实际希望为这个 Web 站点执行的操作。注意，一些未使用的想法可能会在站点将来的更新中使用。

9.1.5 分组和分类

现在可以开始将关于希望介绍的内容的想法分组。如果站点是为了推广多个产品或服务，则可以将这些产品或服务放在产品或服务的相关组中，一个组可以划分为多个子组。例如：

- 可以将公司的组成、公司的历史以及公司的现状等信息组合在一起，放置在“关于我们”部分中。在该部分中，也可以包含公司人员的简介。
- 人们能够与公司联系的不同方式(电话、e-mail、传真、营业时间、地图等)，并且最好有一个联系表单，可以将所有联系方式放置在“联系我们”组中。
- 如果公司具有外部投资方并且是上市公司，您可能希望为投资方创建一个部分，在该部分中放置公司的报告、董事会信息等。

对于大多数站点，创建的部分不要超过 6 个或 7 个，这些部分将组成站点的主要或全局导航项。例如，可以具有如下一些部分：产品、购买地点、贸易询价、关于我们和联系我们。也需要有一个主页(它不包含在 6 个或 7 个主要组中)。这种站点分组方法将使站点更容易导航和理解。

某些部分将很可能包含一些子部分，这些子部分可以具有自己的多个页面。并且，每种类别中可以具有多于 7 个的子部分。例如，出版商在书籍部分中可以具有多于 7 种体裁的书籍，烹饪站点可以根据食物的类型或成分组织食谱部分。这些分部组成次要或者类别导航。

记住，分组应当能够反映期望站点访问者希望执行的操作，并且能够让顾客理解您的产品、服务或主题。例如，如果顾客正在您的站点上寻找一类产品，他们是在制造商列表中还是产品类型列表中寻找该产品？

这些类别和子类别类似于关于内容的表，它们将组成站点导航的基础——每一部分将占用主菜单中的一项，而分部将组成它们自己的子菜单。在 Web 站点上这种组织方式非常重要，因为它们不像书籍一样具有一种线性顺序；用户访问 Web 站点的路线极有可能相当不同。站点组织得越好，用户将越可能有机会找到他们想要的内容。

9.1.6 创建站点地图

现在您应当已经很好地理解组成站点的部分和页面，因此可以开始绘制站点地图，它看上去类似于族树或者 Windows 资源管理器上的文件夹列表。站点地图应当从站点的主页开始，所有的主要类别位于树的顶端。

如果某个类别包含子类别或者多个页面，则这些页面应当显示为第一个页面的子页面。例如，如果其中一个主要类别是“产品”，则可以将其划分为多个子部分，每个子部分都有关于该类别某一项的页面，或者在该页面上列出两种或 3 种产品——并且每一种产品可以具有自己的页面，位于该产品的孙子页面位置。

图 9-1 中给出了一个站点地图的示例；可以垂直(如同该示例中所做的一样)或水平(更加类似于族树)地绘制该地图。

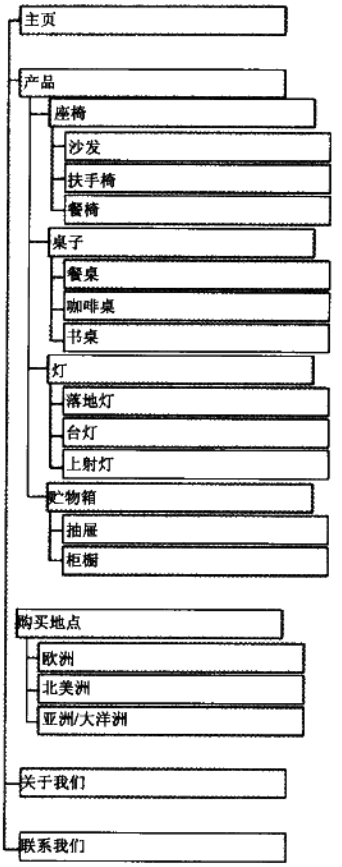


图 9-1

创建站点地图之后，您将了解以下一些方面：

- 站点中具有多少页面
- 每个页面上将显示什么信息
- 每个页面上将显示什么链接(以及这些链接指向何处)

创建站点地图之后，最好尝试查看初始期望用户到达站点后看到的内容，并且查看用户如何通过站点地图进行导航，逐步获得您认为他们将需要的信息。例如，一个制造木匠专用工具的公司可以将访问者定义为两组：

- 贸易采购员希望：
  - 浏览他们能够存储在其 DIY/硬件仓库中的产品列表
  - 查找顾客请求的特定产品
  - 销售团队的联系详情
- 公众人员希望：
  - 浏览他们能够在 DIY/硬件仓库中找到的产品列表

- 查找他们需要的特定产品
- 了解如何联系公司

在编程术语中，组成个人与计算机系统间交互的步骤(用以完成特定任务)通常称为用例。

9.1.7 标识每个页面的关键元素

在开始实际设计页面之前，还需要执行最后一个步骤，即标识每个页面上将出现的关键项或者元素。这些元素将包括：商标或徽标、主要导航、类别导航(位于相关的类别页面上)、题头、页面的主要内容、搜索框、用于自我宣传的空间或者用于为其他公司的产品或服务做广告的空间。

说明：

开始考虑在何处放置页面中的关键元素之前，应当创建一个关于这些关键元素的列表，但是如果能够掌握每个元素将占用多少空间，则会非常有帮助。

这些元素将反映站点的目标。但需要注意的是：许多客户可能希望在每一个页面上放置每一个元素。必须向他们表明，您所完成的组织和规划将导致良好的设计和简单的导航，能够避免在每个页面上放置每一个元素的需求(第 10 章中将更详细地介绍导航)。容易导航并且组织有序的站点比起在每个页面上放置每一个元素的站点要好得多，因为如果页面上的信息太多，则更难以找到想要的内容。

9.2 页面大小(和屏幕分辨率)

现在您已经知道在每个页面上将有些什么内容，因此可以开始设计每一个页面。如同艺术家在开始绘画之前需要确定画布的大小一样，您必须首先确定站点中使用的页面的大小。

遗憾的是，不同的站点访问者可能具有不同大小的显示器，并且显示器可能具有不同的分辨率。因此，对于访问站点的每一个用户，他们看到的页面可能不同；无法设计出在您的显示器上看上去很好并期望它在其他人的计算机上看上去也很好的页面。影响您的“画布”大小的因素有很多。考虑以下因素：

- 不同的计算机具有不同的屏幕分辨率(800×600 和 1024×768 像素是目前最流行的分辨率)。
- 不同的用户具有不同大小的显示器(15、17、19 或 21 以上英寸的显示器)。
- 人们通常不会利用整个屏幕显示区域进行浏览——可能会有工具栏或其他应用程序占用部分空间。

如果在开始设计和构建站点时不使用明智的页面尺寸，则当某个客户回到家中并注意到在工作计算机上看上去显示良好的内容却不能在家中的显示器上正常显示时，您最终可能需要重新构建整个页面。

屏幕分辨率是指显示器屏幕上组成画面的像素数量。800×600 分辨率的屏幕将具有 800 像素宽和 600 像素高，1024×768 分辨率的屏幕将具有 1024 像素宽和 768 像素高，

表 9-1 中给出了 theCounter.com 站点的屏幕分辨率统计数据。统计数据来源于 8 年中的每年一月份对该站点访问者的记录，该表给出了具有不同屏幕分辨率的访问者的百分比。可以在 www.theCounter.com/stats/处查看这些不断更新的统计数据，该站点还具有大量其他有用的统计数据，包括使用不同浏览器版本的访问者的百分比。

表 9-1 (%)

月/年	640×480	800×600	1024×768	1152×864	1280×1024
Jan-08	0	8	48	3	28
Jan-07	0	13	53	3	22
Jan-06	0	21	58	3	12
Jan-05	0	28	54	3	10
Jan-04	1	37	49	3	6
Jan-03	2	46	40	3	4
Jan-02	4	52	34	2	3
Jan-01	7	54	30	2	2
Jan-00	11	56	25	2	2

在 2008 年 1 月仅有 8%的用户的屏幕分辨率是 800×600，90%的用户具有 1024×768 或更高的屏幕分辨率。此时页面宽度通常在 980 像素左右。

但是，当确定页面的宽度时，重要需要记住应该为访问者设计站点。即使您或客户使用的是 21 英寸、1280×1024 分辨率的显示器，也需要确保设计可用于 15 英寸、800×600 分辨率的显示器上(在本章的后面您将看到，Web 页面上的内容经常划分为多列，许多站点将页面右边的空间用于对于正常使用站点不太重要的信息)。

注意：

大多数操作系统允许用户改变显示器的分辨率，因此可以尝试更改分辨率，以了解不同的用户所看到的不同外观。在 PC 计算机上，可以在 Windows Control Panel 中的 Display 选项下更改分辨率；在 Mac 计算机上，则可以在 System Preferences 中的 Displays 选项下更改分辨率。

在垂直方面需要考虑的是，许多用户将具有一个菜单或者任务栏(例如 Windows 操作系统中的任务栏或者 Mac OS X 操作系统中的快捷工具栏)，它们会占用屏幕一部分的垂直高度。还需要考虑出现在浏览器窗口中的各种工具栏。因此，应当确保页面的关键点出现在浏览器窗口顶部的 550 像素内；有时这些空间称为明显位置，即在用户开始滚动页面之前看到的屏幕空间。

说明：

虽然通常应该避免期望用户水平滚动页面，但是可以很安全地期望他们垂直滚动页面。但是，访问者应当能够在不滚动页面的情况下了解页面的主要内容，因此需要确保在

页面加载时它的主要部分可见。通常来说，至少应当能够看到公司的徽标或者商标、任何页面的主要题头以及主要导航的前面几项。

固定宽度设计与流体设计

尽管本章前面提到，应当使内容位于980像素宽的页面内，并且用户应当能够根据从屏幕顶部开始550像素内的内容理解页面，但是您可能已经注意到某些设计伸展以填满整个页面。这种设计称为流体设计。相反，强制页面为特定的宽度或高度的设计称为固定宽度设计。

注意：

值得注意的是，具有较高分辨率显示器的用户倾向于在他们所浏览窗口的边缘留有较大的空余空间，显示更多的桌面或者其他正在运行的应用程序。因此，当用户具有更高分辨率的屏幕时，他们的浏览器窗口将很少占用整个屏幕。

当为客户设计 Web 站点时，最常见的一种误解是，客户经常认为每一个 Web 页面都占用整个屏幕。实情并非如此！

当然，某些站点会伸展以占满整个浏览器窗口。此时，页面的一些部分会根据浏览器串口的大小伸展或收缩，如果用户调整浏览器窗口的大小，页面通常会随着窗口改变大小。图 9-2 给出了一个虚构的新闻站点，该站点使用流体设计技术以使页面占用整个屏幕空间(注意，图 9-2 和图 9-3 中的浏览器窗口具有相同的宽度)。

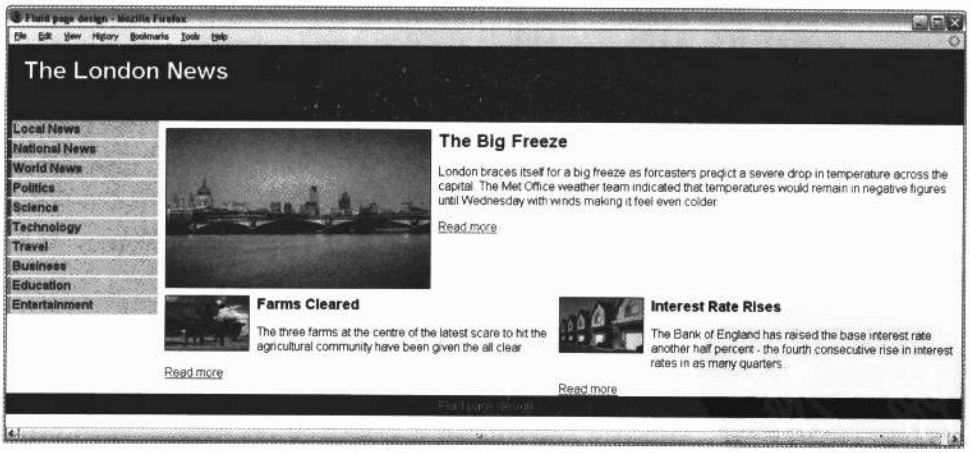


图 9-2

事实上，图 9-2 采用了流体设计和固定宽度设计的混合技术，因为页面左边的导航占用了相同的宽度，而页面的主要部分会伸展以填满浏览器窗口。

但是，许多站点具有完全固定的宽度，同时也能够与页面的左边或者中间对齐。这些站点应当考虑浏览器窗口的有限宽度(如同本章前面介绍的一样)。这种方法(相对于流体设计)的主要区别是，设计人员更多地控制页面的布局，因为他们知道页面到底有多大。这也

意味着设计人员可以限制某些方面的宽度，例如文本列宽度。控制页面的大小在 Web 中也非常有用，因为用户通常会发现很难阅读太宽的段落；当用户到达某行文本的末端时，其视线将很难返回到正确的下一行。

图 9-3 中给出了一个固定宽度设计的示例。当用户增加浏览器窗口的大小时，页面将保持相同的大小，但是页面的右边产生空白(图 9-3 中的浏览器窗口与图 9-2 中的浏览器窗口具有相同的宽度)。

图 9-3

现在已经介绍了流体设计和固定宽度设计，下面的部分中将介绍如何在代码中创建它们。

1. 流体设计

流体设计能够伸展以填满整个页面。为了实现该功能，需要利用百分比值指定页面的比例。例如，可以让页面占用 95%的浏览器窗口宽度，以便在页面周围始终具有少量空白。图 9-4 给出了一个占用浏览器窗口 95%的页面。如果用户增加浏览器窗口的大小，则页面将增加尺寸，但是保留外部的边框。

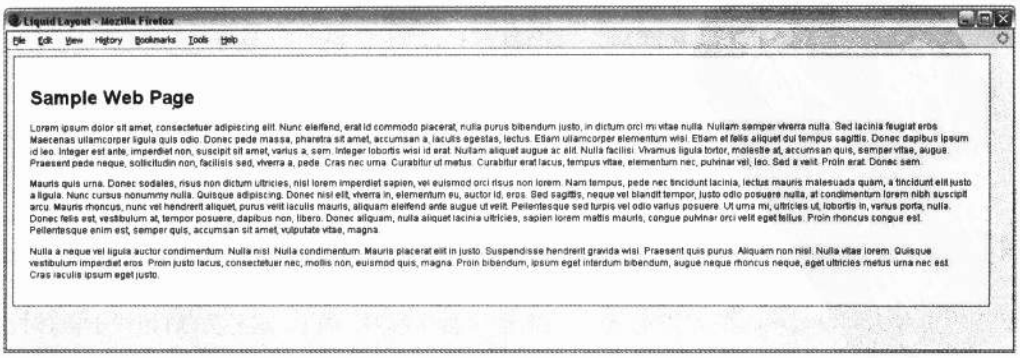


图 9-4

下面是利用 元素创建这种效果的代码(ch09_eg01.html)：

```
<body>
  <div id="page">
    <!-- CONTENT OF PAGE GOES HERE -->
  </div>
</body>
```

相应的样式表包含一条用于 元素的规则，该规则将 width 特性的值设置为 95%。该规则也存在一些其他特性设置以显示包含框，使该示例看起来更具有吸引力：

```
div.page {
  width:95%;
  background-color:#ffffff;
  border:1px solid #666666;
  padding:20px;
  font-size:12px;}
```

在 CSS 出现之前，Web 设计人员通常使用表控制页面中内容的位置，特别是当他们希望在布局中放置多列时。

无论在何处，都应当尽量避免使用<table>元素控制布局，除非是展示表列数据。例如，列车时刻表是真实的表列数据，它包含许多行和许多列，因此应当放置在一个表中。但是，新闻页面实际上不是由表列数据组成，即使新闻被划分为两列组成的布局。

事实上，这种规则的唯一例外情况是在面对较老的浏览器时，例如 Internet Explorer 5 浏览器或者 Netscape 5 浏览器(当前很少有用户使用它们)。

注意：

如果仔细观察，您可能会注意到在 IE 浏览器和 Firefox 浏览器中页面右边的空间通常稍微多于左边的空间。但是，这种现象很少会引人注意。

流体布局方法同时存在一些优点和缺点。其优点如下：

- 页面会伸展以填满浏览器窗口，因此当窗口很大时，在页面周围也不会留下很大的空间。
- 如果用户打开很小的浏览器窗口，则页面会收缩以适合窗口，而不需要用户滚动页面。
- 这种设计可以接受用户设置的字体大于设计人员预期的字体，因为页面布局可以伸展。

缺点如下：

- 如果设计人员不控制页面某些部分的宽度，则页面看上去可能会与设计人员期望的外观区别很大，在某些元素周围可能会有难看的空白，或者某些项会挤压在一起。
- 如果用户具有非常宽的窗口，则文本行将变得非常长，使文本难以阅读。
- 如果用户具有非常窄的窗口，则单词可能会被挤压得过小，每一行中可能最终只有一到两个单词。

2. 固定宽度设计

固定宽度设计使用长度值指示页面的尺寸，例如像素、em(即“全身”，表示欧美文字活字宽度的大小，以一个 12 点的“M”为全身——编者注)和厘米(cm)等长度值。固定宽度设计允许设计人员更多地控制页面的外观，因为设计人员知道画布的大小；当用户调整浏览器窗口的大小时，页面不会伸展或伸缩。即使是在不同分辨率的显示器上，这种设计看上去也只是具有稍微不同的大小，但是页面中元素的比例能够保持相同。图 9-5 中给出了一种固定宽度页面的示例。后面将给出这个页面的代码(ch9_eg02.html)。

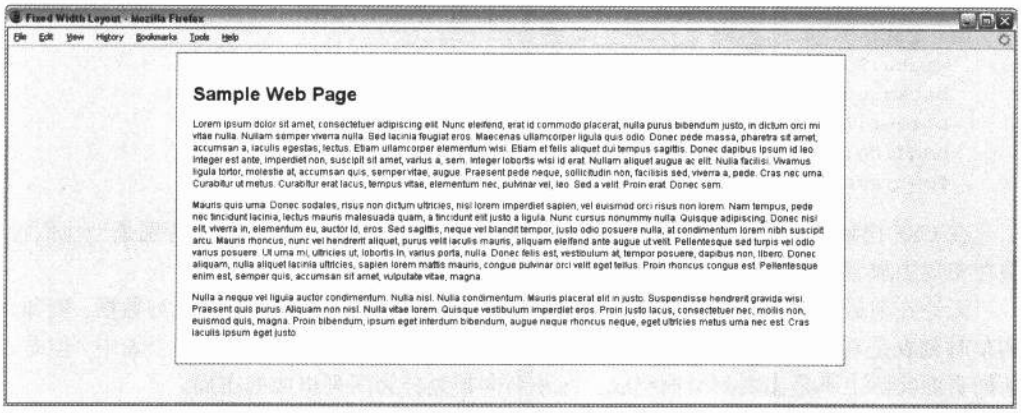


图 9-5

虽然图 9-5 看上去类似于图 9-4，但如果对随本书剩余部分的下载代码(可从 www.wrox.com 处下载)一起提供的对应代码进行测试，将会发现这个示例不会伸展以占用更多的浏览器窗口，它不同于前面流体布局的示例。

当 width 特性用于块级元素时，无论用户的浏览器窗口有多大，该元素(以及页面的布局)都会保持相同大小。如果用户的浏览器窗口窄于布局指定的宽度，则将出现水平滚动栏；而如果用户的浏览器窗口宽于布局指定的宽度，则页面的右边将出现更多的空间；或者如果包含页面的框是居中对齐的，则两边都将出现更多的空间。

在固定宽度设计中，width 特性的值通常以像素的形式给出。在下面的示例中可以看到，保存页面的元素附带一个 id 属性，该属性的值为 page(ch9_eg02.html)：

```
<body>
  <div id="page">
     <!-- CONTENT OF PAGE GOES HERE -->
  </div>
</body>
```

下面查看对应于这个元素的 CSS 规则(ch9_eg02.css)：

```
div#page{
  width:800px;
  margin-left:auto;
  margin-right:auto;
```

```
background-color:#ffffff;
border:1px solid #666666;
padding:20px;
font-size:12px;}
```

width 特性的值将页面的宽度固定为 800 像素宽，而无论浏览器窗口的宽度是否多于或少于 800 像素。您可能也注意到，margin-left 特性和 margin-right 特性的值为 auto，它们确保了页面在浏览器窗口中居中。

注意：

在更高分辨率的显示器中，固定宽度设计使用的像素看上去较小，因为当相同大小的显示器被设置为更高分辨率时，在屏幕中将具有更多的可见像素。

当使用的页面尺寸(以像素为单位)与上面的示例相似时(宽度在 700～1000 像素之间)，则限制为只能通过桌面计算机(或笔记本计算机)中的浏览器访问该站点，而不能通过具有较小屏幕的设备访问，例如 PDA 或者移动电话。而且，该页面尺寸对于电视机顶盒来说也会太大，因为电视的分辨率低于计算机屏幕(在美国，电视的分辨率是 320×240)。

与流体设计一样，固定宽度设计也同时存在优点和缺点。

优点如下：

- 像素值在控制元素的宽度和位置方面是精确的。
- 设计人员能够更多地控制页面中各项的外观和位置。
- 图像的大小将总是相对于页面保持相同。
- 设计人员可以控制文本行的长度，而不需要考虑用户浏览器窗口的大小。

缺点如下：

- 如果用户将字体大小设置为较大的值，则文本可能无法按照计划容纳在分配的空间中。
- 如果用户显示器的分辨率高于为页面所设计的分辨率，则页面在用户的屏幕上看上去将较小，从而难以阅读。
- 这种设计的页面仅适用于屏幕大小和分辨率类似于桌面计算机的设备(例如，页面很可能无法被移动电话或者 PDA 所使用)。
- 编写的代码可能最终会存在很多容器元素，即仅用于控制页面布局的元素。这不仅使页面更加混乱，而且使其更容易断开。
- 页面可能位于浏览器窗口的中间，并且周围存在很多的空白。

现在已经介绍如何控制页面的大小，下面将介绍如何设计页面的内容。

9.3 设计页面

现在您应当已经知道存在多少页面，哪些页面链接到其他页面，每个页面具有哪些主要元素(这里的元素是指页面上的项，例如导航、商标、文章/产品等，而不是标签和它们的内容)以及页面是具有固定大小还是可以伸展。因此，可以开始考虑如何将内容填充到页面中，哪些元素应当分组在一起以及它们应当出现在页面的什么位置。在开始构建页面之

前，应当准备好所有这些方面。

注意：

在开始设计站点之前，询问客户他们最喜欢的 Web 站点和他们喜欢这些站点的哪些方面将会非常有帮助。这将使设计人员能够了解客户的爱好和吸引他们的地方。

9.3.1 规划元素的位置

现在可以开始感受信息在页面上的展现方式，并且规划每一个元素应当出现在页面上的位置。此时设计人员应当仅使用文本和线来规划每个元素(例如题头或者文本的主体)在页面中的位置以及它将获得多少空间；而暂时不用考虑颜色、字体、背景、图像或者其他的一些设计问题。

在这个阶段由于不添加可视化的表示，虽然看上去可能很奇怪(并且最初难以理解)，但非常重要的是可以只关注确保在页面中包含用户能够交互的每一项并赋予它们必须的空间。这个过程有时称为线框绘制。图 9-6 给出了一个 Web 站点的线框示例：

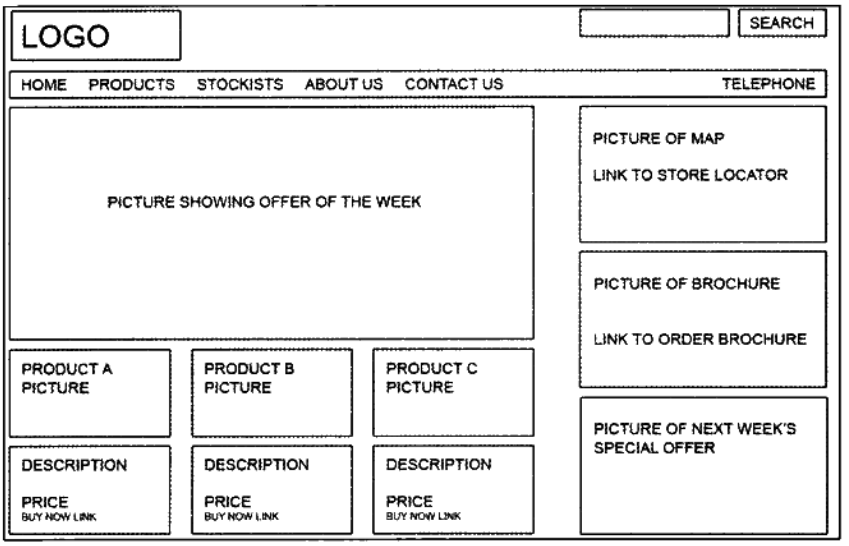


图 9-6

一旦创建好线框模型，可以重新查看期望访问站点的目标访问者列表，以确保他们能够很容易找到您认为他们访问这个站点所需要查找的内容。根据这个简单的模型可以看出链接的目的地，并且能够理解站点的功能，而不会被页面的外观问题转移注意力。这一点非常重要，原因包括两个方面：

- 当向用户和客户展示详尽设计的站点的原型时，他们趋向于关注可见的元素而不是建议的功能。因此，框线模型能够确保客户关注内容的功能和结构而不是它的外观。
- 如果确实需要执行一些改动，可以在设计或编程工作开始之前进行改动，从而避免后面重新编写和/或重新设计站点的大部分内容。

在设计页面的过程中，该阶段的重点是决定哪些元素最重要并且应当出现在页面的顶部。并不是所有的页面内容都将同时显示在屏幕上——或者至少不是在所有的分辨率情况下都可以全部显示。

注意：

假设存在较长的页面，用户不得不垂直滚动以查看整个页面，则需要确保最重要的元素出现在页面的顶部，并且当页面打开时用户能够根据看到的内容了解页面的主旨。

一般的规则是，站点中最常使用的功能应当总是具有较高的优先级，而不是优先显示市场营销部门本周或本月希望推出的内容。请记住，设计站点的目的是为了访问者。如果不能使访问者感到满意，则站点就不成功。

通常来说，当页面加载时下面的项应当首先可见：

- 商标；
- 全局导航(到主页和站点主要部分的链接——注意，主页应该始终是导航的第一项)；
- 子部分导航(如果正位于站点的某个分部，则分部导航应当包含到该分部内的各个部分的链接)；
- 页面的题头或者标题(除了主页之外，主页可能不需要题头或者标题)；
- 足够的内容，用户能够根据这些内容了解页面的主旨；
- 用于搜索站点的选项；
- 促销/广告(自己的或者其他人的)。

当页面加载时，不需要出现在页面可见部分的项包括如下：

- 页面剩余部分的细节(例如，如果具有一篇新闻文章，则仅需要能够看到头条新闻和概述即可；整篇文章不需要出现在页面的顶部)；
- 一些到相关信息或者其他站点的链接(这些内容与使用这个页面不相关)；
- 广告；
- 页脚导航——版权声明、条款和条件、隐私政策(通常这些都是必需的内容，但是很少使用，因此应当出现在页面的底部)。

另外需要牢记的重点是，Web 页面的访问者趋向于浏览页面以了解是否是他们想要的页面，因此需要帮助他们挑选关注的元素——他们几乎从不阅读页面上的所有文本。根据有名的 Web 可用性研究人员和作者 Jacob Nielsen 的研究，人们浏览页面的顺序通常类似于大写字母“F”，先从两个水平条开始，然后是一个垂直条(F 的形状和大小取决于设计、页面的类型以及页面中的信息量)。

因此，将主要导航放在页面的右边并不合适，较好的方式是将其放在徽标下方，从左到右排列。

同样重要的是，文本应当很容易浏览，这意味着应当提供清晰、简洁的题头和副标题，并且尽可能使题头中开始的几个单词能够表明下面段落的主题。

如果是为很可能希望定期改变站点主要功能的公司设计站点，则可能希望分配页面的某个部分给公司控制。可以将主页(或者子类别的主页)的一部分留给他们用于定期改变功能。例如，商店可能在每次市场商机到来时改变页面的主要部分，例如宗教节日、新年、情人节、母亲节、父亲节、学期的开始等。

9.3.2 引入样式

现在您已经了解站点在文本和简单线方面的外观，可以开始向页面中添加样式或字符——采用一些颜色、字体、背景、图像——以创建外观吸引人的页面。

注意，由于知道每个元素在页面中的位置，这将是一项较为容易的任务。

注意：

在被告知所有的元素都已经放置在页面上并且这些元素将占用大量的空间时，一些设计人员可能会遇到一些问题。设计人员可能会觉得他正在被要求为一幅图片着色而不是绘制它。因此，您可能希望要求设计人员参与线框的设计。某些设计人员也可能发现设计非常困难，因为这种工作所需要的方法与他们常用的方法区别很大。

页面中元素的大小和位置是设计过程的一个有效部分(不仅是可视化外观，而且是接口或交互的设计——站点处理的方式)。但是，为设计创建线框的过程有助于用户或客户将注意力放在站点的实际功能上，并有助于开发人员在开始设计页面之前确定功能。开发人员可以选择地告诉客户：线框中元素的精确位置可以改变，这里只是指示将出现在这些页面上的内容。

注意：

如果使用设计人员开发 Web 页面，则将经常不得不权衡，并且有时需要允许设计人员完全重新定位每一页上的元素，以及改变每一项所需要的空间量。只需要确保在页面加载时一些必要的项出现在页面的可见部分中。当开发一些站点之后，您很可能会找到自己的平衡点。

1. 已经完成的工作

类似于任何市场营销形式，创建一种容易识别的样式将有助于强调一种商标。如果公司具有徽标，则应当使用该徽标。如果公司具有特定的颜色，这应当将这些颜色作为站点的颜色方案。但是除此之外，通常需要设计一些具有吸引力的内容。

说明：

除非客户专门要求重新设计公司的徽标或者改变他们的颜色，否则应当避免执行这些操作，因为这是他们已经构建的商标的一部分，并且很可能出现在公司的信纸和广告牌上。

应当询问客户您是否能够拥有徽标的一份数字副本(而不是通过手册扫描输入)。如果广告或传单是打印版本，则客户或者设计它的人员应当具有一份电子版本，可以使用该电子版本。

注意：

我过去曾经有几个客户的徽标令人恐怖，会降低站点的美观程度，但他们没有兴趣更改徽标。如果您不幸地遇到这样的徽标，最好保持相对较小的徽标尺寸；然后可以依赖于公司的颜色来维持一致性，有时甚至可以在徽标附近添加公司的名称并采用较大的简单字体。

您也应当要求公司提供一些材料，例如产品的照片或者为以前的顾客做的工作，以及能够提供的任何文本。如果客户能够为站点提供不错的照片，则会使站点看上去更专业。

2. 常见的页面元素

在大多数情况下，一个站点中的所有页面应当维持一定程度的一致性。与任何形式的商标一样，这种方式有助于访问者根据站点的外观识别它。因此，设计页面的第一步应当是查找将出现在每一个页面中的元素。这通常意味着从商标和主要导航开始。

商标和主要导航在每一个页面中都应当位于相同的位置。例如，如果确定将主要导航放置在徽标的下方并从左到右伸展，则在每一个页面中都应当如此操作。然后可以选择将子导航放置在页面的不同位置，例如页面的左边。但是，当元素出现在多个页面中时，它们应当在每一个页面中都位于相同的位置，从而用户能够更快地了解如何使用站点。

类似地，如果站点为提供的每一个产品或服务专门给出一个页面，或者为出版的每一篇文章或故事专门给出一个页面，则每一个页面都应当遵循一致的设计。例如，如果创建一个在线商铺，则会希望以类似的方式布局每一个产品的信息，以便用户更容易找到这些信息(例如产品的大小或价格)。类似地，如果正在创建基于文章/新闻的站点，则文章的布局很可能是类似的。

如果页面的底部包含一些到其他页面(例如版权声明、隐私政策以及条款和条件等)的链接，则所有页面的底部看上去都应当相同。

您将经常听到与页面相关的术语“页眉”和“页脚”。术语“页眉”通常用于描述站点中任意页面的题头，它们在整个站点中趋向于一致，通常包含徽标和主要导航。同理，“页脚”是出现在每个页面底部的任意内容。页头和页脚之间是页面的内容或主体。

3. 图像对设计的影响

图像的使用经常对站点访问者的感觉具有强烈影响。优秀的徽标、图形和照片是低水平的站点与吸引人的站点之间的主要区别。主页面上的拙劣徽标或者难看的照片将阻止用户通览站点，无论站点的内容有多么优秀。

设计人员越来越多地为站点添加专门为该公司拍摄的高质量照片，并且这些照片通常不只是产品的照片——它们是代表一种生活方式的图像或者是公司竭力希望与商标相关联的图像。这些图像可以与公司的其他市场营销工作紧密关联(或者来自于这些市场营销工作)。

照片的质量通常取决于站点的预算。如果客户具有一些用于市场营销的专业照片，则应该考虑使用它们。或者如果客户的预算足够多，则可以雇佣一个摄影师进行适当的拍照。实际上，跨国公司的站点和流行商标中都具有令人印象深刻的图形。

注意：

虽然在 Web 上具有大量的免费剪贴画，但是它们将使站点看上去很业余。这些免费的剪贴画对于业余爱好者站点来说较为合适，但是用于公司的 Web 站点则不理想。

也可以从大量的库存照片站点(包含大量可供授权使用的照片)处获得图像，而不是雇佣一个摄影师。但需要注意的是，使用这些图像通常需要付费，费用可能会相对便宜，也可能会非常昂贵。廉价销售图像的站点包括 www.istockphoto.com 和 www.sxc.hu/，图像销售价格较贵的站点包括 www.gettyimages.com 和 www.corbis.com。

注意：

在使用一幅图像之前，应当始终确保具有必要的版权许可。如果没有版权许可，则可能会引起官司和昂贵的罚金，或者至少是被告知删除该图像，这就需要您重新设计站点并向客户解释错误。

4. 分组元素

可以使用下面的方法使用户了解页面中的某些元素是相关的。例如，可能希望将新产品的链接、注册表单中的元素或产品的概述详情分组在一起。

- 位置：确保设计的类似元素的位置接近
- 颜色：使用前景色和背景色块可以清晰地表明哪些项是相关的。
- 边框和内边距：在页面中的一组元素和其他项之间创建间隔，以表明哪些元素分组在一起
- 样式：例如对于导航项使用类似的按钮

图 9-7 给出了 Apple 商店中的一个页面，其中演示了以上每一种方法的示例。这个页面演示了分组相关元素的多个示例(主要导航、商铺中销售的 Mac 计算机的类型、分组在一起的不同型号)。颜色用于区分主要导航，边框用于分组页面的不同部分，并且选择按钮具有清晰的样式。

图 9-7

9.3.3 导航

当开始任何设计时，最先确定的很可能是导航的位置。

如何设计导航很大程度取决于在该菜单上需要具有多少项。开发人员应当已经很好了解了站点的结构，并且应当已经创建了页面的层次结构，在创建站点地图时，该层次结构图将决定导航的内容——请记住，站点最多只能分为 7 个部分。在一个页面上应当尽可能地避免具有超过 10 个主要链接。

在站点的分部中，副标题和链接的数量可以超过 7 个，并且子导航或次要导航有时在页面中的出现位置与主要导航不同。

1. 顶部导航

将导航直接放置在页眉的下方是非常流行的方法，通常导航与页面的左边对齐或者居中对齐。图 9-7 中的页面是一个顶部导航示例，该页面显示了 Apple.com Web 站点。

2. 左导航

将导航栏放置在左边是一种较为少见的选择，但是某些站点确实将它们的主要导航放置在页面左边。图 9-8 给出了 BBC 的新闻站点，它的主要导航位于页面左边。

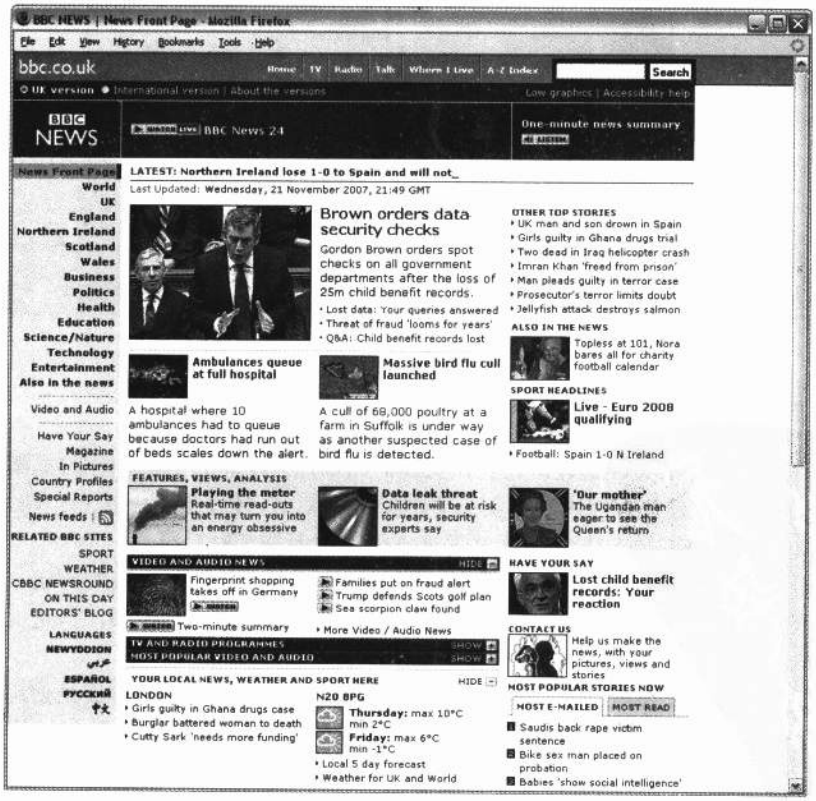


图 9-8

3. 顶部和左导航

许多站点结合使用顶部导航和左导航，其中顶部导航用于主要导航，左导航用于次要导航或者子导航。图 9-9 给出了这样的示例，该示例来源于 eBay.com，它的顶部实际上具有两个导航集，左边的子导航取决于用户所在的部分。

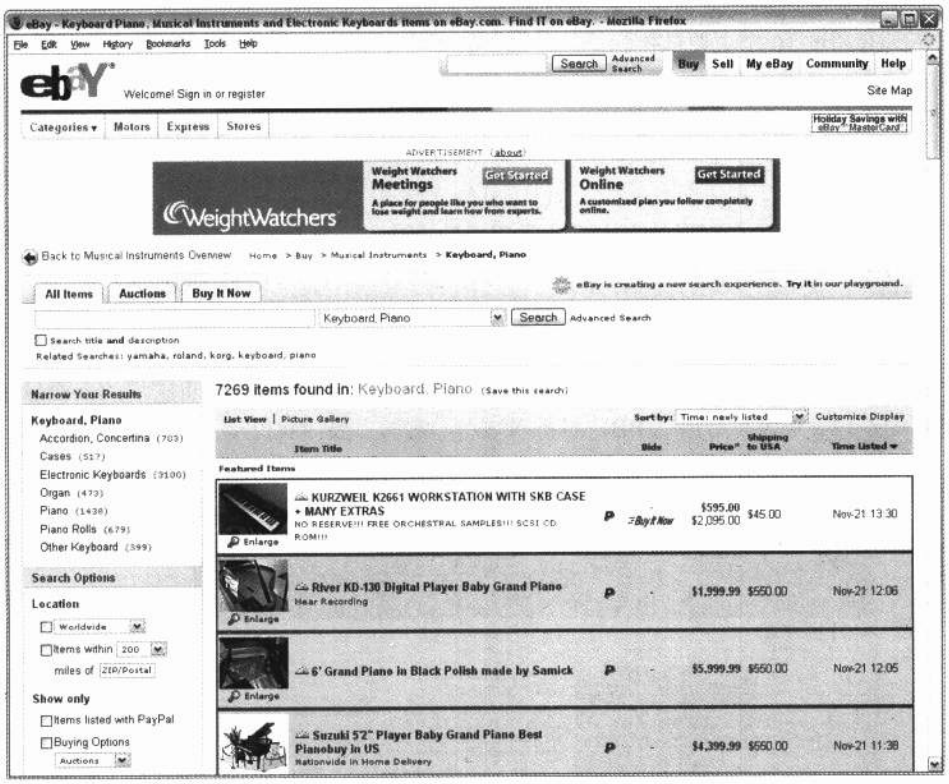


图 9-9

4. 底部导航

通常应当避免将主要导航链接放置在页面的底部，因为如果不使用框架(框架将使页面的编写变得复杂——建议仅在内容确实需要框架时才使用它们)或者 CSS 定位特性(CSS 定位特性仅被最新的浏览器支持)，则无法保证它们在页面加载时可见。

但是，通常将重要性较低的链接放置在页面的底部——例如，到版权声明、隐私政策以及条款和条件的链接。这些是您可能希望在每一个页面上都包含的项，但是又不希望它们占用页面初次加载时的宝贵屏幕空间。

您也将发现某些站点在页面的底部放置仅由文本组成的导航栏，前提是使用复杂的图像创建主导航栏。这将有助于使用屏幕阅读器的人员更快速、更方便地访问链接。

5. 右导航

您很少会发现采用右导航栏的站点，但并不是没有听说过这样的站点。将主要链接放置在页面右边的缺点是，当页面加载时，如果用户的浏览器窗口较窄，则可能无法看到链接；它不像左导航栏或顶部导航那样能够立即显示出来。另外，用户对具有右导航的站点的使用经验较少，因此通常不期望能够在右边找到导航。

通常右边的列用于额外的内容或到站点其他部分的链接，这些对于用户的导航来说并不重要。例如，电子商务商铺可能会在右边的列中包含一系列到相关产品或者添加到商铺的最新项的链接；这些都是为站点用户添加的额外功能，对于导航来说不是必须的功能。

9.3.4 主页面

第一印象非常关键，因此主页面非常重要。除非是为公司或者家喻户晓的主题创建站点，否则让访问者能够从主页面获得站点的主要目标是非常重要的——可以考虑使用公司名称中的一句标志性文字或徽标辅助完成该任务。

然后需要强调用户到达这个站点时最有可能希望执行的任务——以便尽快地帮助大多数人找到他们希望看到的信息。

说明：

本章前面已经提到，需要重点记住的是：主页面不应当仅介绍公司市场营销部门希望介绍的每周或每月营销成绩。主页面不应当只是他们的广告栏，而必须也能够解决站点主要访问者的需求。例如，市场营销部门可能希望推出一款新产品，而访问站点的大多数顾客希望找到一种较早的、更可靠的产品。如果这些用户无法在站点中找到他们想要的信息，则市场营销部门无法最大化推广他们的产品。在用户的需求和公司的需求之间进行平衡非常重要——并且用户应当具有更高的优先级。

因为访问者趋向于浏览页面而不是阅读所有内容，所以所有题头和链接名都应当以重要的关键字开头，这样有助于用户理解该部分或链接的作用。

注意：

补充说明一下，曾经多次有客户要求我创建 flash 动画，用于在用户到达站点的主页面之前播放。但是，这些所谓的醒目页面通常不被赞成使用，无论动画多么令人印象深刻，因为它们阻碍用户查看希望看到的内容。

9.3.5 内容页面

内容页面是大多数站点的主要部分；在新闻站点中，内容页面可以包含一些文章；在电子商务站点中，内容页面可以包含每一件产品的细节。内容页面显示内容的方式应当使用户容易阅读信息。

本章前面已经提到，如果具有多个产品或服务，则为每一项所提供的信息应当一致。如果处理的是衣服，访问者应当能够快速并方便地了解这款衣服具有的不同颜色和大小。

可以通过将类似的信息放置在每个页面的相同位置来实现该操作。根据这种思想，实际上仅需要设计一个文章页面或者产品页面，所有其他页面应当遵循该模板。

不能使页面太拥挤；清晰的表示能够使用户将注意力集中于内容。即使有大量的信息需要填充在页面上，也需要确保不同元素之间存在足够的空间。

图像应当与产品、服务或正在讨论的主题相关，如果它们是左对齐或者右对齐，并且文本环绕在它们周围，则通常看上去更加美观。在图像和它们周围的文本之间也应当存在间隔(可以使用 CSS 中的 padding 特性或 margin 特性进行设置)。

注意：

应当避免在任何商业站点上使用剪贴画或 GIF 动画，但是如果需要的话，可以在个人站点上使用它们。虽然跳舞的猫第一眼看上去非常可爱，但是如果它与尝试向用户展现的主题无关，则将会转移站点的实际目标。

如果处理的是公司出售的产品，则这些页面需要面向操作——它们必须允许顾客定位或选择某项产品，查找所需的相关信息，然后最有可能的是购买该产品。

当需要展现大量文本时，确保页面中的文本行不要太宽。本书前面已经提到，很多人在计算机屏幕上阅读宽文本行时会存在困难。此外，确保所有文本被划分为适当的副标题，并且尽可能使用非常短的段落编写专门用于 Web 的版本。

注意：

许多公司认为，简单地在 Web 上给他们的产品或服务做广告将带来新的业务，顾客如果需要报价将会联系他们。他们似乎认为，向用户说出价格将更容易实现交易。根据我的经验，如果在站点中添加报价，则将会获得更多的询问(如果提供的每件产品/服务的价格不同，则最好提供已经完成的工作的价格示例)。记住，通过 e-mail 询问价格不会使交易更容易实现(除非用户提供了电话联系方式)，因为顾客对于您来说仍然是匿名的，并且他们可能忽略您发送的 e-mail。

9.4 构造页面

前面介绍了如何控制页面大小，并且讨论了 Web 页面设计应当考虑的一些常见问题。如果您正在设计一个站点，则应当已经了解如何将线框转换为可用的并且吸引人的站点。现在必须将这些设计转换为代码。

如果希望让元素在页面中的位置不是它们在普通流中应当出现的位置(在普通流中，每个元素只按照编写它们的顺序出现)，则需要使用表或者 CSS 来确定元素在页面中的位置。设计人员希望创建的一种最常见的效果是具有多列的布局——例如，页面左边的一个窄列用于导航，紧跟的第二列包含页面的主要内容(并且有时右边也存在多列)。

专业设计人员趋向于使用网格来布局页面——网格由一系列的行和列组成，它们定义了页面的形状和内容的位置。例如，图 9-10 给出了一个页面如何划分为多行和多列(使用一些粗黑线隔开每一行和每一列)。

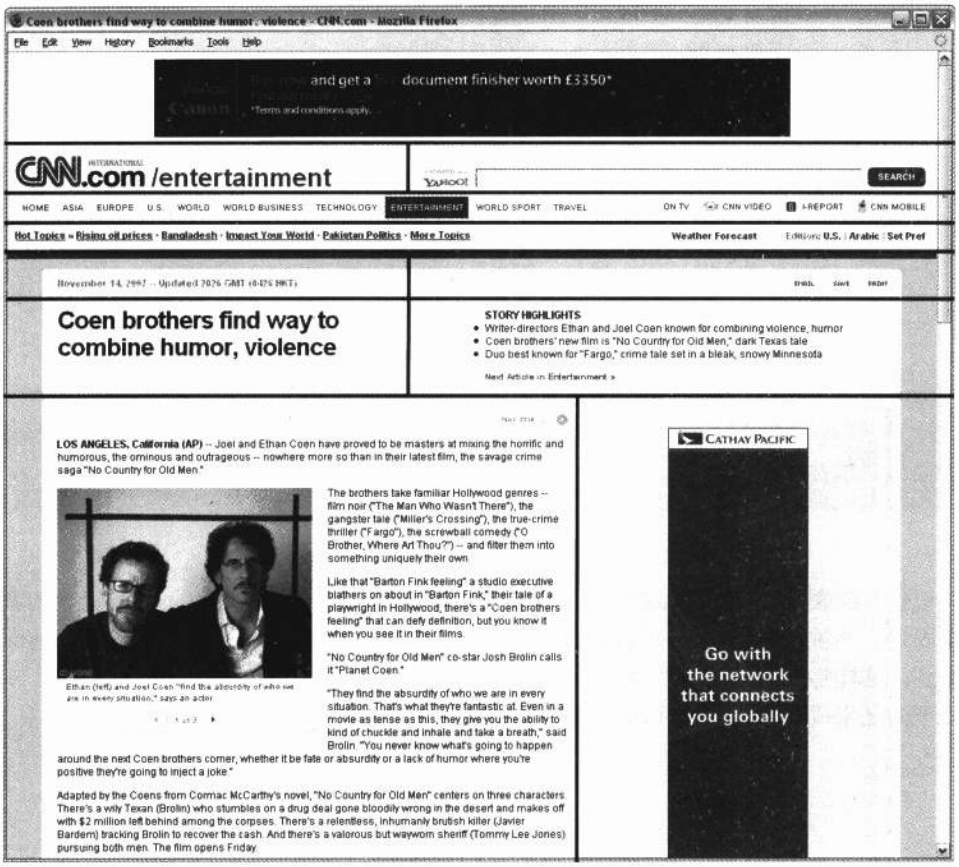


图 9-10

这些行和列可以具有不同的高度和宽度，但毫无疑问地是存在一个网格。页眉包含多个行(广告、标题和搜索栏、导航以及热点主题/版本)。然后在页面主体的开始部分存在另外两行(第一行显示日期，第二行是标题/故事提要)。在此之后，页面主体具有两列的布局；主列用于放置文章，第二列用于放置广告。

在下面的小节中将介绍如何创建单列布局和多列布局。

9.4.1 单列布局

单列布局是最容易创建的布局方式，通常用于较小的站点。单列布局的站点通常至少具有 3 到 4 行。

如果查看图 9-11，会发现这个示例站点基于固定宽度的页面，该页面具有 1 列和 3 行。依次查看每一行，其中第一行包含公司名、徽标或商标，第二行包含导航，第三行包含页面的内容。每一行都具有不同的背景色，以便区分它们(还可以存在第四行，用于包含版权声明、到各种条款和条件或隐私政策的链接以及少数访问者将使用的一些其他必要的链接)。

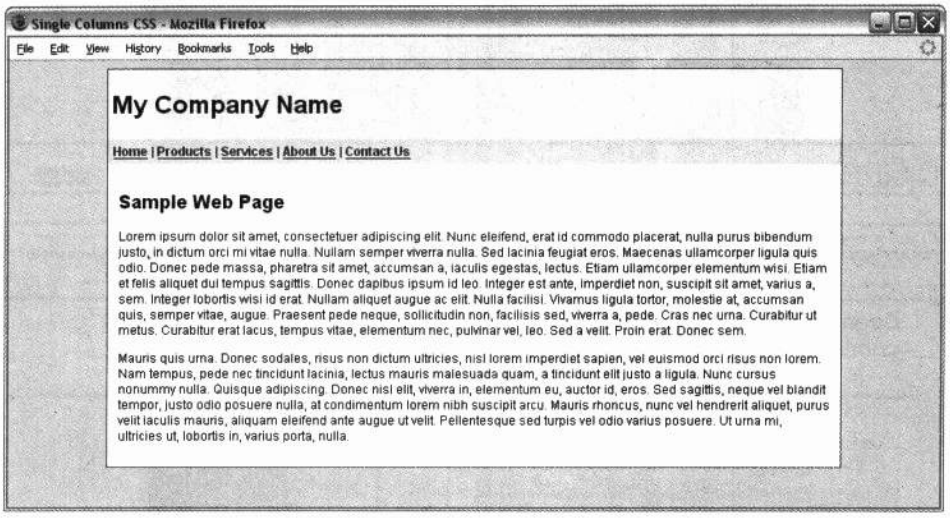


图 9-11

记住，在这种类型的站点中，必须控制文本的宽度。如果包含页面的 元素的宽度没有固定，并被允许伸展到浏览器窗口的整个宽度，则最终文本行可能会非常长，前面已经提到，这样将使页面难以阅读。

下面是生成这种结构的代码(ch9_eg03.html)：

```
<body>
  <div class="page">
    <div class="header"><h1>My Company Name</h1></div>
    <div class="nav">
      <!--NAVIGATION GOES HERE -->
    </div>
    <div class="content">
    <!-- MAIN PAGE CONTENT GOES HERE -->
    </div>
  </div>
</body>
```

现在查看这个页面的 CSS。这里需要注意的关键点是，page 类的 width 特性设置为创建固定宽度页面，margin-left 特性和 margin-right 特性被设置为 auto，以使页面在屏幕中居中显示(另外还设置了一些其他的背景和边框样式，以便更容易查看示例)。

```
body {
  background-color:#d6d6d6;
  font-family:arial, verdana, sans-serif;}
.page {
  width:700px;
  margin-left:auto;
  margin-right:auto;
  font-size:12px;
```

```
  background-color:#ffffff;
  border-style:solid; border-width:1px; border-color:#666666;}
.header {background-color:#f3f3f3; padding:3px;}
.nav {font-weight:bold; background-color:#e3e3e3; padding:5px;}
.content {padding:10px;}
```

根据 CSS 规范，任何具有相等的左边和右边页边空白的块级元素(在这个示例中是 元素)应当显示在页面的中间。您可能会猜测为什么此处使用了“应当”一词，因为该规则不被一些较老的浏览器所支持，它仅在 IE6 浏览器和 Netscape 6 浏览器中才开始获得支持(并且 IE 仅在文档具有 Strict HTML 4.0 声明或 XHTML DOCTYPE 声明时才会支持它)。图 9-12 给出了这个示例在 IE6 浏览器中的外观。

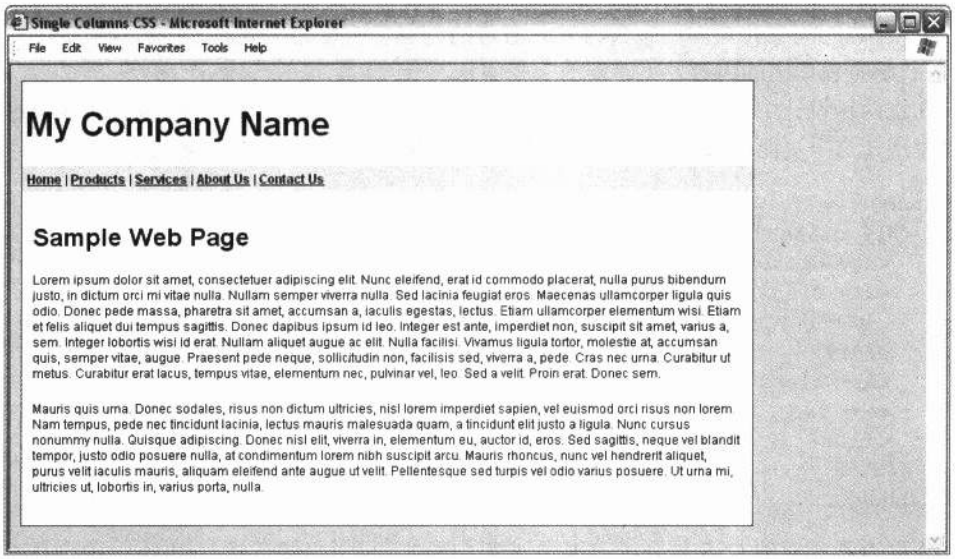


图 9-12

但是，使用一种简单的技巧可以克服这个问题，方式是添加两个 text-align 特性。将第一个 text-align 特性添加到包含元素中(在这个示例中是<body>元素)并赋予值 center，以居中显示它内部的元素；将第二个 text-align 特性添加到 元素中，该元素的 class 属性的值是 left，以阻止该元素内部的文本居中显示。结果如下所示(ch11_eg03.css)：

```
body {background-color:#d6d6d6;
  font-family:arial, verdana, sans-serif;
  text-align:center;}
.page {
  margin-left:auto;
  margin-right:auto;
  text-align:left;
  width:700px;
  font-size:12px;
  background-color:#ffffff;
```

```
  border-style:solid; border-width:1px; border-color:#666666;}
.header {background-color:#f3f3f3; padding:3px;}
.nav {font-weight:bold; background-color:#e3e3e3; padding:5px;}
.content {padding:10px;}
```

事实上，为了使页面正确显示，不需要像这个例子那样添加一些额外的特性，这就降低了采用 CSS 作为表示格式的意义。

9.4.2 双列布局

双列布局趋向于用于以下两种情形之一：

- 左边的列包含左导航，另一列包含页面的主要内容。
- 左边的列包含页面的主要内容，右边的列包含一些相关内容、广告或其他对于理解主窗口中的内容并不重要的一些信息。

在这两种设计中，页面通常仍然具有一个题头，并且题头跨越两列。从下面的代码中可以发现，这个页面的 XHTML 与上一个示例的 XHTML 相同(ch11_eg04.html)：

```
<body>
  <div class="page">
    <div class="header"><h1>My Company Name</h1></div>
    <div class="nav">
      <!--NAVIGATION GOES HERE -->
    </div>
    <div class="content">
    <!-- MAIN PAGE CONTENT GOES HERE -->
    </div>
  </div>
</body>
```

下面查看相应的 CSS 规则，该规则创建了两列。可以仅尝试修改 元素的 CSS 规则，该元素的 class 属性具有值 nav：

```
.nav {
  float:left;
  width:100px;
  font-weight:bold;
  background-color:#e3e3e3;
  padding:5px;}
```

这个元素需要具有值为 left 的 float 特性。必须在该元素中指定 float 特性，因为它也影响下面元素的内容的生成方式。另外，当指定 float 特性时，应该也为该块指定宽度。如果没有指定宽度，则它将占用其包含块的全部宽度。

图 9-13 给出了该示例的显示效果。

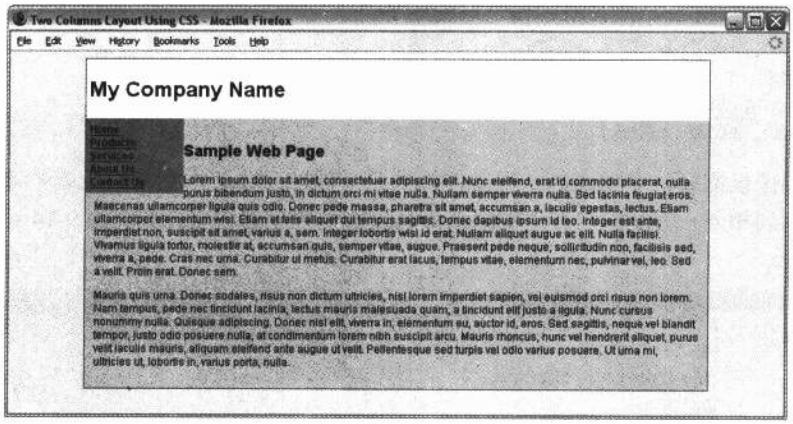


图 9-13

这里产生了一个非常严重的问题——左边的列没有占用页面的全部高度，并且内容的主体出现在它的下方。由于页面具有背景色，这个问题被突出显示。假设正在为一个新闻站点创建模板；每篇文章可能具有不同的长度，并且希望左边的列不仅伸展到页面的整个高度，而且具有相似颜色的背景。但是，因为文章的高度不同，所以不能设置固定的高度，并且简单地赋予 height 特性 100%的值也不会其作用。相反，需要修改样式表中的其他两个规则：

- 在页面的主内容处添加一个左页边空白。该页边空白的大小与导航的宽度相同(这是下面的代码中第二段突出显示的部分)。
- 不使用背景颜色控制列的背景色，而是在包含元素(元素)中使用一幅背景图像。

因此，修改后的样式表如下所示(ch09_eg04.css)：

```
body {background-color:#efefef;
  font-family:arial, verdana, sans-serif;
  text-align:center;}
.page {
  margin-left:auto;
  margin-right:auto;
  text-align:left;
  width:700px;
  font-size:12px;
  background-image:url(images/2columnbackground.gif);
  background-repeat:repeat-y;
  border:1px solid #666666;}

.header {
  padding:3px;
  background-color:#ffffff;}
.nav {
  font-weight:bold;
  padding:5px;
```

```
    float:left;
    width:100px;}
  .content {
    padding:10px;
    margin-left:100px;}
```

背景图像仅需要一个像素高，因此图像的尺寸将非常小。因为包含元素的高度将与 2 列中的较高列相同，所以将在占用页面全部高度的两列中都添加阴影。图 9-14 给出了这个页面的外观。

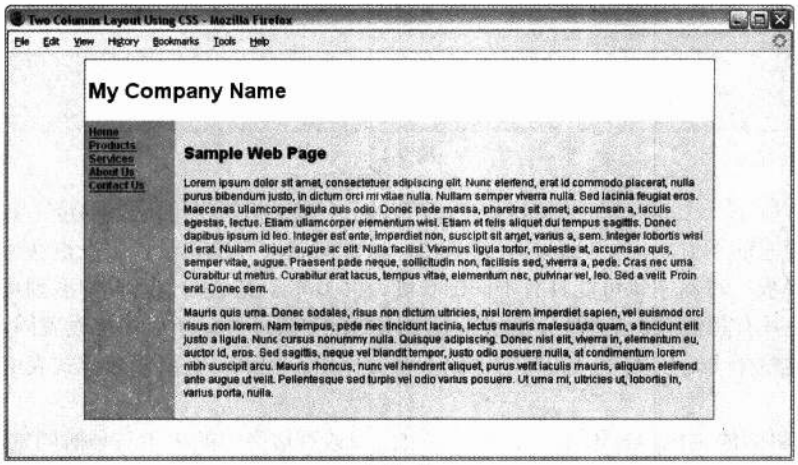


图 9-14

9.4.3 3 列布局

在 3 列布局中，通常导航位于左边列，主要内容位于中间列，然后在右边列中包含一些额外内容，例如相关链接、一些感兴趣的项、广告等。图 9-15 给出了一个 3 列布局的示例。

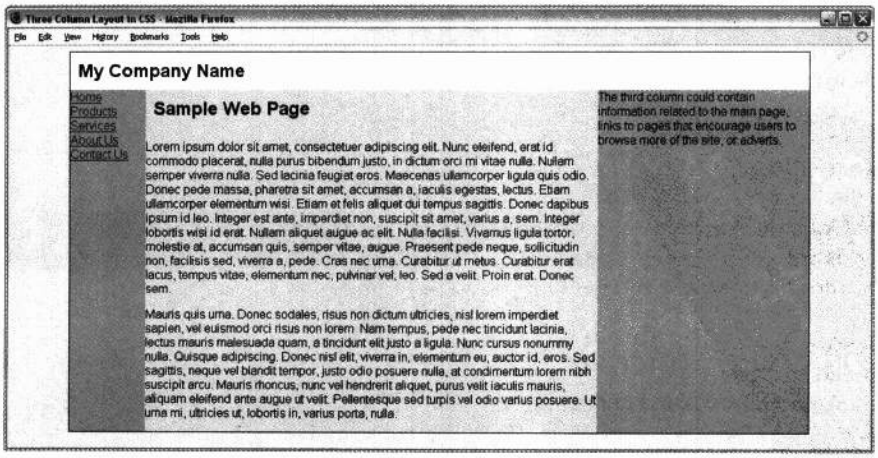


图 9-15

下面的示例演示了如何在 XHTML 中创建这种布局，该示例使用具有 3 列的表，每一个表列都有固定宽度(ch9_eg05.html)：

```
<body>
  <div class="page">
    <div class="header">
      <!-- heading goes here -->
    </div>
    <div class="nav">
      <!-- navigation goes here -->
    </div>
    <div class="content">
      <!-- content goes here -->
    </div>
    <div class="right">
      <!-- right hand column content goes here -->
    </div>
  </div>
</body>
```

对于这个示例，可以采用与上面示例相似的方法；但是，这里将使用 float 特性定位所有 3 列。

```
body {
  color:#000000;
  background-color:#efefef;
  font-family:arial, verdana, sans-serif;}
.page {
  width:980px;
  border:1px solid #000000;
  background-image:url(images/3columnbackground.gif);
  background-repeat:repeat-y;
  margin-left:auto;
  margin-right:auto;
  margin-top:0px;
  padding:0px;}
.header {background-color:#ffffff;}
.nav {
  float:left;
  width:100px;}
.content {
  float:left;
  width:600px;}
.right {
  float:left;
  width:278px;}
.clear{clear:both;}
```

从中可以看到， 元素(该元素具有值为 page 的 class 属性)是整个页面的容器，并且具有一幅背景图像，以确保 3 列都具有一致的阴影。

此处值得注意的一点是，虽然整个页面是 980 像素宽，但如果将每一列的 width 属性值加在一起，则结果仅是 978 像素。这是因为 Internet Explorer 6 浏览器(以及早期的版本)将边框和内边距都计算为框宽度的一部分。需要牢记的是，IE 浏览器将页面两边的单个像素宽的边框也作为包含框宽度的一部分进行计算(如果在 IE 浏览器中所有列的总宽度达到 980 像素，则右边的列将被挤压到其他两列之下)。

这个错误称为框模型错误，在 Web 中经常产生这种错误。根据 CSS 规范，块级元素的宽度应当仅表示框中的内容的宽度，不包含内边距、边框和页边空白的宽度，而 IE 5 和 IE 6 浏览器将边框和内边距都作为框宽度的一部分计算。为了在 IE 6 浏览器中避免产生这种错误，可以在文档中包含 Strict XHTML !DOCTYPE 声明。

还需要提醒的是，因为内边距将作为框宽度的一部分添加，为了确保 3 列中的任何内容不接触框的边缘，需要创建较窄的列并添加 padding 特性，或者在用于创建每一列的 元素的所有直接子元素中添加页边空白或者内边距。

9.4.4 牺牲列

有时设计中的第三列(右边的列)也称为牺牲列，类似于上面示例中的第三列。具体思想是，虽然牺牲列的内容可能会帮助用户更好地理解站点并改进他们的体验，但它不是站点日常使用必须具有的对象。因此，即使用户的屏幕分辨率低于整个页面的宽度，仍然能够使用站点。因此，如果用户的屏幕分辨率是 600×800(600 像素高和 800 像素宽)，则用户仍然能够看到左边的列和中间列；但是必须滚动页面才能看到右边的列。

显示器分辨率为 800×600 的用户可以通过滚动页面看到牺牲列，但是这种页面设计方法使得用户能够看到页面的所有主要内容。

如果用户的浏览器在窗口宽度范围内无法显示牺牲列，则用户应当不会丢失页面的任何含义，它的主要内容应当不会难以理解。但是，用户可能会丢失屏幕之外的一些额外信息或者广告。

图 9-16 所示的设计示例中使用了一个牺牲列。浏览器被设置为 760 像素宽，用户可以清晰地看到该报纸的主要文章和主要导航。虽然在页面的右边存在一些额外信息和广告，但它们对于页面的使用并不重要。

图 9-16

9.4.5 利用 CSS 的高级布局

因为 IE 6 浏览器在浏览器市场中占有重要份额，所以利用 CSS 控制 Web 页面的布局变得相当常见。而且这种方式很快被认为是布局 Web 页面的正确方式，并且很多站点提供了使用 CSS 创建吸引人的布局的技巧。下面的站点可以帮助 Web 开发人员使用 CSS 创建吸引人的布局：

- www.thenoodleincident.com/tutorials/box_lesson/boxes.html
- www.bluerobot.com/web/layouts/
- www.glish.com/css/
- www.alistapart.com/topics/code/css/
- www.meyerweb.com/eric/css/edge/
- www.positioniseverything.net/(这个站点对于处理浏览器错误非常有帮助，所谓的浏览器错误是指编写的 CSS 代码的显示效果与期望的效果不同)。

在 CSS 出现之前，通常使用表控制元素在页面中出现的位置。如果查看足够的 Web 页面源代码，将会发现大量的站点仍然使用表控制页面的布局；在阅读完关于布局的一章之前，您应当可能已经查看一个这样的示例。

9.4.6 利用嵌套表创建布局

第 4 章介绍表时曾提到，表可用于控制页面中元素的位置。支持 CSS 布局的浏览器在大多数 Web 用户中变得流行之前，使用表控制布局是一种常见的技术。查看图 9-17 中的页面。

图 9-17

整个页面都位于一个表中：页眉位于表的一行中，页面的主要内容位于第二行中，页脚位于第三行中。在每一行中都是嵌套表，这些嵌套表包含第一行中的页眉、第二行中的主要内容、第三行中的页脚。

嵌套表的使用使页面看上去如同跨越整个浏览器窗口，但是同时允许控制内容本身的宽度。页面顶部和底部的背景色能够伸展整个浏览器宽度，而文本和图像位于嵌套表中，并且这些嵌套表具有固定宽度。

在这个设计中，最外层的表伸展整个浏览器窗口宽度和高度。在该表中，第一行包含页眉，第二行是主要内容，第三行是页脚。在这些行内是具有固定宽度的嵌套表，它们确保页面的内容保持相同的宽度。

下面查看这个页面的 XHTML 代码(ch9_eg06.html)。在这个示例中集成了一些较老样式的技术，从而如果遇到像这样编写的页面，就能够理解它的工作原理。第一个表是最外层表，它包含整个页面。确保该表伸展整个浏览器窗口的高度和宽度(其中 height 属性是 XHTML 规范的扩展，它仅被 IE 4 或更新版本、Netscape 3 或更新版本的浏览器所支持)：

```
    <body>
    <table width="100%" height="100%" border="0" cellpadding="0"
cellspacing="0">
      <tr valign="top"><td valign="top">
```

在第一行之内，另外一个表包含页面的题头。第一行中的这个嵌套表与其他两行的嵌套表(它们分别包含主要内容和页脚)稍微有所不同，因为页眉实际上包含具有 3 种不同阴影的 3 行，这 3 行组成了页眉：

- 第一行是黑色阴影，并且右边存在两个单词“Latin Example”；
- 第二行是中灰色阴影，包含了一个徽标和一个横幅广告；
- 第三行是轻灰色阴影，包含了导航栏。

因此，页眉位于一个嵌套表中，该表也包含了 3 行，并且伸展整个浏览器宽度。每一行具有一个不同的类，该类对应于一条 CSS 规则，用于指示相应的背景色。但是，这些行的内容需要具有相同的 700 像素宽。为了获得该宽度，内容放置在另外一个具有固定宽度的嵌套表中。

```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr class="topBar">
    <td height="20">
       <table width="700" border="0" align="center" cellpadding="0"
              cellspacing="0">
         <tr>
           <td class="TM">Latin Example</td>
         </tr>
       </table>
     </td>
   </tr>
   <tr>
     <td height="100" class="masthead">
       <table width="700" border="0" align="center" cellpadding="0"
              cellspacing="0">
         <tr>
           <td><!--LOGO IMAGE GOES HERE --></td>
           <td><!-- BANNER AD GOES HERE --></td>
         </tr>
       </table>
     </td>
   </tr>
   <tr>
     <td height="20" class="nav">
       <table width="700" border="0" align="center" cellpadding="0"
              cellspacing="0" class="nav">
         <tr>
           <td><a href="">Home</a> | <a href="">Products</a> |
               <a href="">Services</a> | <a href="">About Us</a> |
               <a href="">Contact Us</a> </td>
         </tr>
       </table>
```

```
      </td>
    </tr>
  </table>
</td></tr>
```

这是页眉的末尾，也是包含整个页面的表的第一行末尾。

下一行是另外一个嵌套表，它保存页面的主要内容。

```
<tr valign="middle"><td valign="middle">
  <table width="700" border="0" align="center" cellpadding="0"
         cellspacing="10">
    <tr>
      <td valign="top" width="100%">
        <h2>Sample Web Page</h2>
        <p><!-- LATIN TEXT GOES HERE --></p>
      </td>
    </tr>
  </table>
</td></tr>
```

最后一行是页面的页脚，它看上去如同伸展整个页面宽度，这一点类似于页眉。这种效果的创建方式是，赋予整个行相同的背景色，然后将具有固定宽度的嵌套表放置在该行中，该嵌套表包含整个页脚：

```
<tr valign="bottom" class="footer">
  <td valign="bottom">
    <table width="700" border="0" align="center" cellpadding="0"
    cellspacing="0">
      <tr>
        <td height="20" class="footer">&copy; 2008 Latin Example</td>
      </tr>
    </table>
  </td>
</tr>
</table>
</body>
</html>
```

在操作表时必须记住的是，一个表单元格必须完整地包含其他任何元素——不允许元素跨越两个表单元格。

9.5 本章小结

本章介绍了关于页面布局的基础知识。所有设计人员在创建更多的站点并尝试使用一些较为复杂的技术之后将学到一些新技巧。本章主要介绍一些基础技能，利用它们可以将设计思路转换为可用的布局。

本章不仅介绍与改变页面的物理外观有关的内容，而且介绍了如何设计站点。在开始规划任何设计之前，需要确保了解站点的目标、期望访问站点的人群、期望访问者希望从

您的站点中获得的内容。然后确定需要将什么类型的信息放置在页面上并将这些信息组织成不同的部分。这些部分应当能够反映站点采用的结构，并且开发人员还能够创建一个站点地图，用于表明所有的页面如何组织在一起，以及用户如何导航站点以实现您认为他们希望完成的任务。

第一种真正的设计决策应当是选择使用固定宽度布局还是流体布局——即页面应当总是保持相同的大小还是随着浏览器窗口的大小扩展和收缩。

了解站点的整体结构、页面的大小、出现在每个页面上的元素后，可以开始创建站点的线框设计，线框仅由线和文本组成——不包含任何样式，直到客户理解并同意站点中内容的类型以及站点访问者如何获得他们想要的内容之后，才开始考虑样式设计。

接下来就可以开始在页面上添加样式——字体、颜色、图像、元素的定位等。确定页面的外观之后，可以使用 CSS 将元素定位在页面中。

希望本章提供的实用建议能够帮助开发人员较容易地设计 Web 站点，并且能够帮助开发人员与他们的客户打交道。在下一章中，将介绍如何设计 Web 站点的更特殊方面。

9.6 练习

所有练习的答案都在附录 A 中给出。

1. 再一次查看本章开始部分中给出的页面；图 9-18 再次显示了该页面。请列举出页面中在设计阶段应当列出的所有不同元素，并将它们放置在相关的分组或者类别中。

例如，对于搜索框可以列举以下元素：

```
Title
Navigation
Main news article
```

图 9-18

2. 尝试重新创建图 9-18 中看到的页面。提示：该页面是一种固定宽度的设计，因此可以使用关于创建固定宽度布局一节中的样本代码作为起点，并在其中添加一些内容。

第10章

设计问题

本章介绍影响页面特定部分的设计问题——文本、菜单、表和表单。本章的每一节解决不同的设计问题，并且每一节都包含一些有用的提示，它们将使开发人员制作的页面不仅看上去更吸引人，而且更容易被更多访问者使用。

本章首先介绍文本以及如何对齐它们，如何将它们间隔开来以及如何控制文本列的宽度，然后介绍如何选择字体和字体大小，并且您将了解背景图像如何影响文本的可读性。

本章接下来介绍导航，这个主题覆盖了3个方面：菜单、链接和搜索功能。几乎每个站点都具有菜单，它帮助用户在站点的各个部分之间导航。如果菜单不清晰，人们将无法浏览您的站点，他们不会愿意查看更多的内容。因此，需要掌握关于菜单设计的一些关键点，以便所设计的菜单更容易被用户理解和使用。大多数站点也将链接放置在页面的其他部分而不只是菜单中，并且开发人员需要让用户清晰地看到站点中具有哪些链接，以便他们知道在哪里单击。可以使用一些特殊的技术告知用户哪里有链接，例如颜色、下划线以及改变光标图标的形状以指示用户链接在哪里。最后，搜索选项能够帮助用户找到他们想要的内容，而不需要使用导航；或者当用户尝试使用导航但仍然无法找到想要的内容时，可以使用搜索选项。

本章将快速介绍如何在表中添加阴影以帮助用户更容易地查看数据。然后将介绍如何创建可用的表单。表单是从用户处收集信息的最常用方式。但是，大多数人都不愿意填写表单，因此设计良好的表单可以极大地增加用户填写表单的机会，并且增加用户填写正确信息的机会。

虽然本章不能教授您如何成为优秀的Web页面设计人员——这需要创造力、优秀的眼力和天资——但是本章将介绍如何实现一些能够带来优秀设计的效果，并且介绍一些在设计站点时可以使用的指导原则，利用它们可以同时改进站点的外观和可用性。

注意：

在本章的整个介绍过程中，您将看到称为屏幕阅读器的软件。屏幕阅读器能够为用户朗读页面。虽然屏幕阅读器通常由视力受损的人员使用，但它们很可能会流行于其他基于Web的环境中，例如某些人可能希望访问一些信息，但是他们正在驾车或者做一些其他的事情，从而无法阅读屏幕，这些人可以考虑使用屏幕阅读器。

10.1 文本

在本节中将介绍一些与在页面中定位文本相关的问题。首先介绍与文本的位置和行距

相关的问题，然后介绍与字体相关的问题。本节主要包括以下内容：

- 添加空白以制作更吸引人的页面。
- 仔细对齐文本以使其更易于阅读。
- 宽文本列通常难以阅读(最好不要用于概述)。
- 背景图像能够使文本难以阅读(因为对比度低)。
- 必须仔细选择字体。
- 固定大小的字体在不同分辨率的屏幕上可能具有不同的大小。

10.1.1 空白有助于制作更吸引人的页面

文本之间应当始终具有足够的空间。太过拥挤的页面不仅难于阅读，而且使页面的吸引力降低。导航、文本、图像和其他项之间存在空间的页面将更容易被阅读，并且更具吸引力。页面中元素之间的空间通常被设计人员称为空白。

可以使用两种方式在页面中创建空白：内边距和页页边空白白。此处需要注意的是，空白不一定是白色；它仅指页面中元素之间的空间，这些元素包括文本、图像、表单控件、表边框和链接等(空白通常与页面的背景具有相同的颜色)。

也可以改变页面中某个框的颜色，以将该设计元素与页面的其他部分分离(例如，主要导航通常放置在彩色的背景上)。通过赋予页面某一部分稍微不同的背景阴影，可以帮助该部分相对于周围的其他项突出显示。再次提醒，具有背景色的元素也应该使用内边距和页页边空白白来创建边缘周围的间隔。

请查看图 10-1，其中页面上的导航、文本、图像和表之间存在少量间隔或不存在间隔(ch10_eg01.html)。

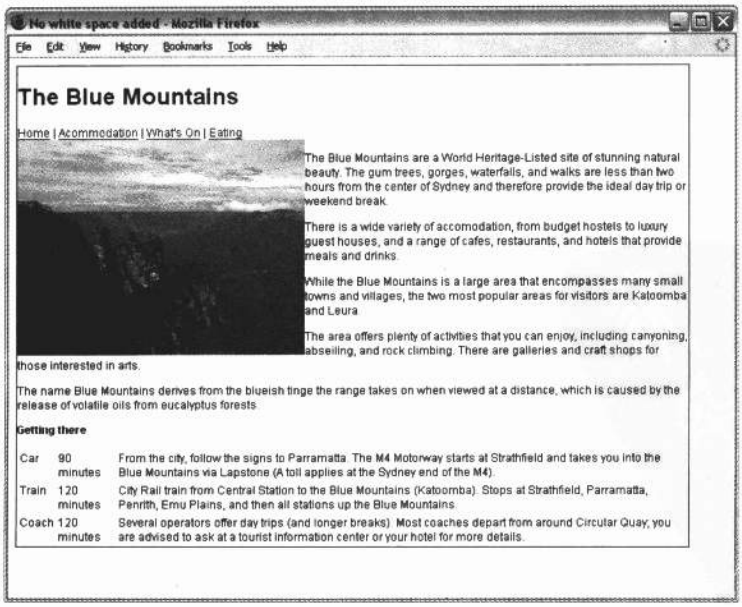


图 10-1

现在将图 10-1 与图 10-2 进行比较。页面中元素之间添加的空间使得它具有更好的可读性并且更吸引人，而背景色有助于分组相关的项(例如导航和旅游指南)，并且将它们与周围的项区分开来(ch10_eg02.html)。

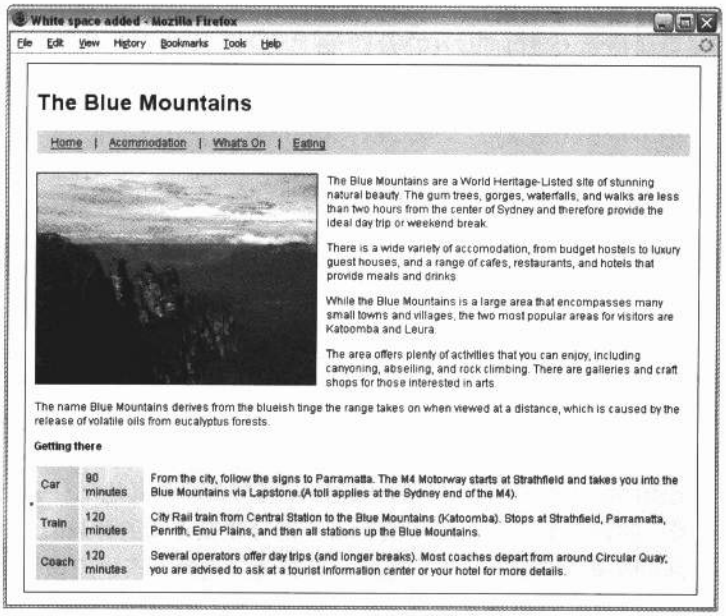


图 10-2

显而易见的是，包含页面的框内具有空间；导航中的项不是非常接近图片，它们具有背景色，以将导航分组在一起并将题头与页面的剩余部分分离，并且它们周围存在内边距和页边空白，这使得它们更易于阅读。另外，在主图片和它周围的文本间也添加了空间，方式是赋予图片页边空白。最后，位于底部的表的单元格中使用背景色和内边距的组合，以在单元格之间添加空间。

使用 CSS 向页面添加额外空间。第 7 章中已经介绍了 CSS 如何作用于框模型，其中 XHTML 文档中的每一个元素都表示为一个框。每个框具有一个边框，而边框外部存在页边空白，页边空白将这个框与相邻的框分离。框中的内边距将内容与边框分离。使用 border 特性和 margin 特性向这个页面添加空白。

更改这个示例外观的第一条 CSS 规则应用于 元素，该元素包含整个页面。包含页面的粗黑线外部存在 10 个像素宽的页边空白以添加空间；然后在围绕页面的粗黑线的内部存在 10 个像素宽的内边距，从而创建线和页面内容之间的空间(使文本不再接触这些线)。

```
.page {
  width:700px;
  margin:10px;
  padding:10px;
  border:1px solid #000000;}
```

在导航上添加背景色有助于将导航分组在一起，并且帮助将页面的题头与页面内容的剩余部分分离。这个框的底部页边空白有助于将导航与页面分离，而这个灰色框内的内边距在链接周围创建空间。

在第二条规则中，每个链接左边和右边的页边空白在导航中的各项之间添加空间，使这些导航项相互独立，从而更易于阅读。

```
.navigation {
  background-color:#d6d6d6;
  padding:5px;
  margin-bottom:20px;}
.navigation a {margin:0px 10px 0px 10px;}
```

<img>元素的右边也被赋予页边空白，以增加该元素和旁边文本之间的距离。从图 10-1 中可以看出，当文本与图像的边缘接触时会变得较难阅读：

```
img {
  margin-right:10px;
  border:1px solid #000000;}
```

最后，页面底部表中的不同单元格具有不同的背景阴影，它们与单元格之间的间隔一起使内容具有更好的可读性。页面中每个表单元格的内边距也在文本和这些单元格的边框之间添加间隔，再次增加了内容的可读性。

```
td {padding:5px;}
.transport {background-color:#d6d6d6;}
.duration {background-color:#e6e6e6;}
.description {background-color:#f2f2f2;}
```

10.1.2 仔细对齐文本以使其更具可读性

对齐文本的方式决定了文本的可读性。通常对文本采用左对齐方式(这是默认对齐方式)。但是，文本也可以是居中对齐、右对齐或两端对齐，另外它还能够被页面中的其他元素挤出所在的行。

通常来说，如果文本段落是左对齐，则它将具有更好的可读性。虽然您可能认为文本段落居中对齐或两端对齐看上去较为美观，但这将使文本更难以阅读。居中对齐最好用于题头(偶尔也用于非常短的段落)。

注意：

如果喜欢使用两端对齐的文本，则需要确保文本列足够宽以支持这种对齐方式，而不是像此处的一些单词之间存在较大的间隔——在窄段落中，两端对齐方式看上去非常奇怪。

图 10-3 中的示例具有 3 段文本：第一段居中对齐，第二段左对齐，第三段两端对齐(ch10_eg03.html)。

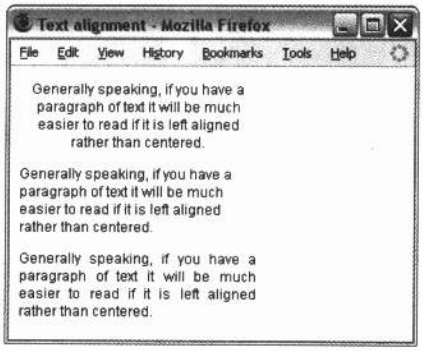


图 10-3

10.1.3 调整文本行高度以使文本更具可读性

当处理大量文本时，增加文本行之间的垂直空间量通常非常有帮助——这种空间量被图形设计人员和印刷工人称为行距。可以使用 CSS 的 line-height 特性增加或减少文本行之间的空间量。

在 Web 页面中增加这种垂直空间量是非常有用的，原因有以下两点：

- 如果文本行之间存在更多的空间，则当到达一行的末尾后，更容易发现下一行的起点。
- 人们通常浏览 Web 页面而不是阅读它，当浏览页面中的文本时，通常关注于字母的顶部而不是底部，因此这种额外的空间使访问者能够更容易地快速阅读页面。

图 10-4 给出的示例具有两个段落，第二个段落增加了文本行之间的空间。

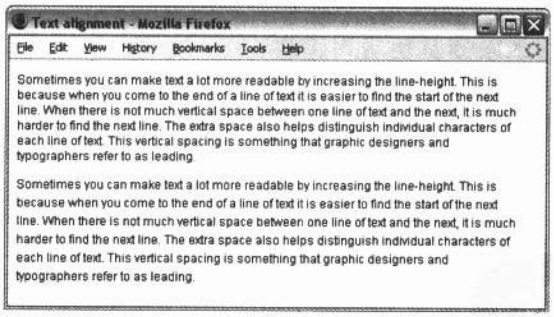


图 10-4

10.1.4 宽列的文本更难以阅读

大多数人都发现计算机屏幕中的长文本行难以阅读。因为文本行越宽，用户越难以找到正确的行——特别是在用户浏览页面而不是仔细阅读它的情况下，这种问题变得尤其严重。因此，限制文本的列宽度非常有帮助。

文本列的宽度取决于它们的内容。如果是为整篇文章创建页面，则列的宽度很可能宽于为多个文章片段(用于许多用户的快速浏览)创建的页面。毕竟，当用户决定希望阅读一

些内容时，相对于浏览主页面，他们将在查找正确的文本上花费更多的时间。

对于流体设计来说，这是特别重要的问题，因为流体设计的页面将伸展以填满整个浏览器窗口。此时具有高分辨率显示器的用户将最终得到非常宽的页面，从而难以找到下一行文本的起点。图 10-5 中给出了一个这样的示例(ch10_eg05.html)。

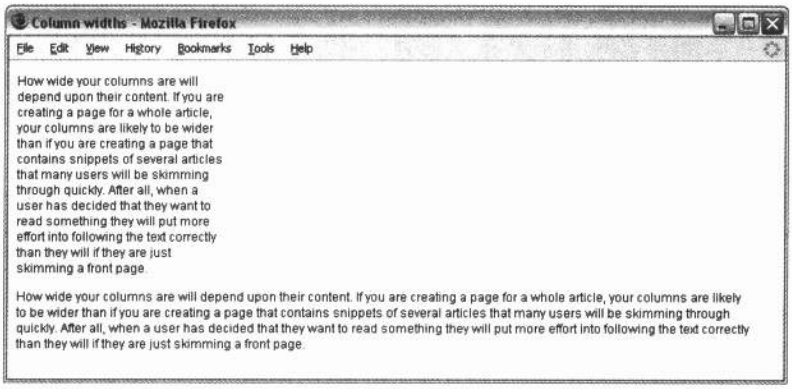


图 10-5

可以对块级元素使用 CSS 的 width 特性来控制文本的宽度——在这个示例中，该特性用于<p>元素。

您也将发现，为联机格式编写的段落通常短于为打印版本编写的段落，因为较短的段落更易于阅读。如果不得不编写非常长的内容，则最好将其划分到多个不同的页面中，而不是将其放置在一个长页面中，使得用户不得不滚动页面才能完整地阅读它。

10.1.5 背景图像会使文本难以阅读

如果希望在任何文本后面使用背景图像，则必须确保文本能够突出显示在背景之上。在许多 Web 站点上——特别是使用背景图像作为整个页面的壁纸的 Web 站点——选择背景图像的原因是用户喜欢图像，而不是这些图像是很好的背景。

使用照片作为背景图像通常并不合适，因为照片的各个部分具有不同的对比度，从而使文本较难阅读。如果确实希望使用图像作为背景，则需要：

- 确保图像具有低对比度——例如使用纹理——而不是使用照片。
- 赋予任何包含文本的框背景色。

请记住，背景图像必须是尺寸较小的文件，否则它们将花费较长的加载时间。指定背景色与照片的主要颜色相似也是一种优秀的实践，因为当页面加载时通常将显示该背景色。

10.1.6 仔细选择字体

对于长时间阅读来说，人们通常认为 serif 字体更容易阅读。有证据表明，为了理解句子的含义，并不需要阅读 serif 字体中的每一个字符(相比较 sans-serif 字体而言)。事实上，当阅读以熟悉的语言编写的书籍和长篇文章时，优秀的读者不需要仔细查看每个字符，仅需要关注于字符的上半部分或者熟悉单词的大概形状。

但是，在 Web 上该证据并不充分。许多人发现在 Web 上更容易阅读 sans-serif 字体，因为字体上的衬线(顶部、尾部和卷曲)实际上降低了字符清晰度(主要是因为屏幕的分辨率低于打印页面的分辨率)。事实上，无论是使用 serif 字体还是 sans-serif 字体，只要它们足够大，则都易于阅读。但是，为较长的文本块使用 serif 字体的论据无法从打印版本照搬到 Web 页面中。

当选择哪些字体(或字型)用于 Web 页面时，应该确保您认为用户很可能已经在他们的计算机上安装这些字体；如果用户没有在他们的计算机上安装所需的字体，则您的设计就不会以该字体显示。遗憾的是，能够期望用户拥有的字体是非常有限的。可以安全地假设大多数用户已经安装 Arial、Verdana、Impact、Courier 或 Courier New、Georgia、Times 或 Times New Roman 等字型，但是不能假设除此之外的其他字型(特别是当访问者使用不同的操作系统时)。

可以提供一种候选字体，以防止用户不具有首选的字体。如果计算机上没有安装首选的字体，则浏览器将查找候选列表中的下一种字体，以此类推(如果最终没有找到一种字体，则计算机将使用该字体族的默认字体)。

在 CSS 中，当以偏好的顺序提供字体列表时，可以采用类似于如下的方式使用 font-family 字体：

```
font-family: Geneva, Arial, Helvetica, sans-serif;
```

下面是一些可以应用的常见字体列表：

- Sans-serif：Helvetica、Arial、sans-serif
- Sans-serif：Arial、Verdana、sans-serif
- Serif：Times New Roman、Times、serif
- Monospace：Courier New、Courier、等宽字体

事实上，大多数用户会安装多种字型；只是开发人员无法确保用户具有与他们相同的字体。例如，很少有用户会安装 ITC New Baskerville 字体，但是该字体的外观与 Times 字体类似，因此可以将 ITC New Baskerville 字体作为首选的字体，然后如果用户没有安装该字体，则可以依赖于 Times 字体。

注意：

这种方法的缺点是，ITC New Baskerville 字体是一种较宽的字体，因此它将在布局中占用比 Times 字体更多的空间——当设计页面时需要了解字体的尺寸区别，否则这种区别可能导致页面外观与预期的效果差距较大。

虽然存在一些技术能够用于将字体下载到用户计算机上，但是如果用户没有安装该字体，则这些技术也具有一些缺陷：

- 它们需要具有分发该字体的许可权(与 CD 一样，字体通常也是有版权的)。
- 它们不能够在所有的浏览器中都起作用。
- 将花费更多的时间绘制页面。
- 如果只是为了查看一个页面，则人们不喜欢下载文件。

如果需要使用一种特殊的字体，例如用于徽标的字体，可以考虑使用包含文本的 GIF 图像。但是，如果将图像用于大量的文本，则被认为是糟糕的实践，并且应当将文本作为 alt 属性的值，以便无法看到图像的人了解图像。

另外还要确保选择的字型适合于希望绘制的图像。例如，如果正在尝试绘制专业的图像，则应当避免使用类似于 Comic Sans MS 的字体，该字体是一种“娱乐性”的字体。

注意：

如果希望更详细地了解这个主题，在 http://psychology.wichita.edu/surl/usabilitynews/3S/font.htm 中包含了关于各种字体有用性的令人感兴趣的信息。

10.1.7 固定大小的字体受屏幕分辨率影响

应当了解的是，如果使用固定字体大小，例如像素或点，则它们出现在屏幕上的大小将取决于用户的显示器。例如，12 像素高的字体在 1 280×1 024 的显示器上看上去要比在 800×600 的显示器上小很多(两个显示器具有相同物理尺寸)，因为前者在屏幕上的像素数量比后者多 40%。

使用固定大小的字体也使用户难以改变字体的大小，即使他们阅读文本时存在困难。

10.2 导航

关于导航最有趣的一件事情是，无论如何规划站点，不同的用户仍然会以他们自己的方式使用它。例如，即使期望人们从站点的主页开始访问站点，但是站点投入运行之后，您会发现其他站点将链接到该站点的不同页面，并且许多访问者通过另一个页面访问该站点。

因此，在设计站点时，您的工作是帮助人们尽快和尽可能方便地找到他们想要的信息。

用户将通过以下 3 种常用方式之一导航站点：

- 使用站点提供的菜单
- 通过文本中间和页面其他部分中提供的链接浏览
- 搜索相关的信息项

在本节中将介绍如何让用户使用这 3 种方法更方便浏览站点。

10.2.1 菜单

菜单是任何 Web 站点的关键部分，它可能会出现在多个页面中。用户可以使用菜单快速而方便地查看站点的各个部分，并且快速到达他们想去的位置。

第 9 章中介绍过，一个站点可以具有多个菜单；可以将主要导航配置在一个菜单中，将次要导航配置在子菜单或独立的菜单中。通常菜单出现在站点的顶部并从左到右排列(在徽标的上方或下方)，或者自上而下出现在页面的左边。

菜单趋向于成为用户在站点的各个部分之间进行导航的主要方式，优秀的菜单设计在

以下方面具有巨大的影响：

- 当用户到达站点时，他们是否能够获得想要的内容
- 他们将在站点上花费多长时间

在本节中将介绍以下 8 种指导性规则：

- 菜单必须关注用户希望获得的内容。
- 菜单必须简洁。
- 如果使用图标，确保它们易于理解并且也需要添加文本链接。
- 菜单项的分组必须合理。
- 菜单必须被快速、方便地阅读。
- 菜单项必须易于选择。
- 菜单的加载速度必须快速。
- 菜单在整个站点中必须一致。

1. 菜单必须关注用户希望获得的内容

对于绝大多数的 Web 站点来说，关于菜单的首要考虑事项是满足用户的需求——特别是商业站点。您需要询问以下问题：

- 用户来到站点希望查找什么内容？他们是希望发现谁建立了公司以及谁是董事会成员，还是希望查找提供的产品或服务方面的信息？
- 如何以用户能够理解的简洁术语最好地描述这些信息？重要的是，菜单项不能太冗长，否则它们将难以浏览。另外，不能使用访问者可能无法理解的专业术语(否则他们将不知道单击什么内容)。
- 菜单项中最重要的是什么？每个链接的突出地位应当反映请求该信息的站点访问者数量。例如，如果正在创建一个音像商店站点，该站点主要销售吉他而不是鼓，则第一个菜单项(位于主页按钮之后)应当是并于吉他而不是鼓的链接(即使您希望建立关于鼓的业务)。

通过这些问题可以较好地理解菜单中菜单项的顺序以及每个菜单项的意义。记住，也可以使用一个页脚导航，用于类似关于公司背景信息或者潜在广告客户的项。

2. 菜单必须清晰地独立于内容

当设计页面时，菜单必须能够被立即识别出，以便用于站点的导航。可以使用大量的技术实现该目标：

- 菜单的字体大小可以与页面主要内容的字体大小不同(通常菜单的文本应当大于页面中的主要文本)。
- 可以在菜单周围添加额外的空间(如同本章前面介绍空白时的示例所示)。
- 可以将菜单放置在具有不同背景色的框中(如同本章前面的示例所示)，或者使用一条线将其与主要内容分离。

虽然使用图像可以使菜单与内容具有很大区别，但必须注意的是图像不能太大，否则将降低站点的加载速度。图 10-6 给出了在独立的框内使用图像作为导航的示例。

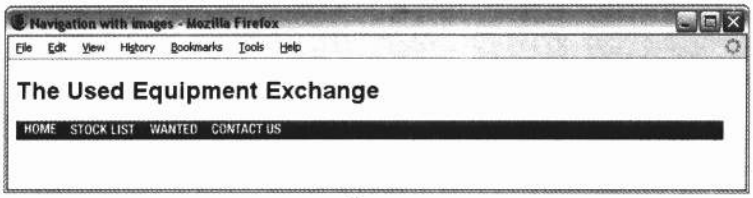


图 10-6

下面仔细查看这个菜单的创建方式：在一个 元素内创建该菜单；该元素的 CSS 规则指定它应该具有背景图像，并且背景图像从左到右重复。背景仅需要一个像素宽，以使该图像的尺寸较小，从而节省页面的加载时间。背景图像的高度与用于单击的图像相同。

在 元素内部是链接到其他页面的图像。在本章后面查看这个示例的 CSS 规则时，请注意其中一个规则如何指定链接内的图像不应该具有边框——这是因为默认情况下 IE 会在这样的图像周围添加蓝色框。

在每幅图像之间是一幅间隔图像，该图像是一条较黑的线，用于分离相互对接的链接。

下面是该示例的 XHTML 代码(ch10_eg06.html)：

```
    <div class="page">
      <h1>The Used Equipment Exchange</h1>

      <div class="navigation">
        <img src="images/navigation_divider.gif" alt="" width="2" height="16" />
        <a href="/" title="Home page">
          <img src="images/navigation_home.gif" alt="Home"
                width="38" height="16"
        </a>
        <img src="images/navigation_divider.gif" alt="" width="2" height="16" />
        <a href="stocklList.aspx" title="Stock List">
          <img src="images/navigation_stock_list.gif" alt="Stock List"
width="70" height="16" />
        </a>
        <img src="images/navigation_divider.gif" alt="" width="2" height="16" />
        <a href="equipmentWanted.aspx" title="Equipment Wanted">
          <img src="images/navigation_wanted.gif" alt="Wanted" width="54"
               height="16" />
        </a>
        <img src="images/navigation_divider.gif" alt="" width="2" height="16" />
        <a href="contactUs.aspx" title="Contact Us">
          <img src="images/navigation_contact_us.gif" alt="Contact Us" width="75"
               height="16" />
        </a>
        <img src="images/navigation_divider.gif" alt="" width="2" height="16" />
        </div>
      </div>
    </body>
```

下面是这个示例的 CSS 规则(ch10_eg06.css)：

```
body {
  background-color:#ffffff;
  font-family:arial, verdana, sans-serif; font-size:12px;}

.page {width:700px;}
.navigation {
  background-image:url(images/backdrop.gif);
  background-repeat:repeat-x;}

a img {border:none;}
```

因为所有这些图像的尺寸都非常小(使用 Adobe Photoshop 软件中的 Save for Web 选项保存用于 Web 的图像)，所以当下载页面时它们不会增加太多的下载时间。

3. 使用图标表示链接时确保每个人都能理解它们

许多 Web 站点使用称为图标的图像作为链接。这些图标都是图像，例如用于指示搜索功能的放大镜。如果打算使用图标，则需要确保目标观众能够理解这些图标的意义，否则他们不会单击这些图标。

许多用户熟悉下面的图标：

- 用于指示主页的房子
- 用于指示搜索功能的放大镜
- 用于指示 e-mail 地址或链接的信封
- 用于指示帮助文件的问号

如果使用不常见的图标，则最好同时以一些相关单词和图像的形式添加链接(不要期望用户会将光标悬停在链接上以查看该链接的工具提示)。

4. 菜单必须能够被快速、容易地阅读

本章前面已经提到，当浏览 Web 页面时，大多数访问者实际上不会阅读它们——而是浏览页面。如果菜单与页面的主要部分区别较大(可以对页面主体内的链接使用粗体、不同颜色或具有下划线的文本)，则将有助于用户更容易地浏览并注意到导航项。

菜单中使用的单词或图像必须足够大以方便阅读(特别是对于屏幕分辨率较高的用户来说，他们显示器上的文本将较小)，并且文本必须与背景之间具有较高的对比度。链接也必须较短和简洁。例如，文本为“Home”的链接比文本为“Front door”的链接更明显，更容易阅读和理解。几个简单的单词始终好于一个专业术语单词。

5. 菜单项的分组必须合理

如果有很多页面，则最好创建一些子菜单。如果创建子菜单，则非常重要的是将菜单项分组，以便访问者了解在何处查找他们想要的内容，而不需要搜索多个部分或类别。

如果使用子菜单，则应该确保它们能够与主菜单清晰地区分，并且确保能够清晰地表明哪一项属于哪个部分。子菜单通常使用不同的背景色、更小的字体、缩进的位置，或者使用一种备选的颜色来表明它们区别于主菜单。

例如，如果正在创建一个计算机商店站点，则可以创建一个具有3个主要部分的分组，每一个主要部分包含自己的一些子部分：

- 计算机：桌面计算机、笔记本计算机
- 软件：商业软件、游戏软件
- 外围设备：打印机、扫描仪

这种方式比起以字母顺序排序所有部分的菜单更容易导航。

6. 菜单项必须容易选择

如果菜单项太小或者菜单项之间没有足够的空间，则某些用户可能难以选择正确的菜单项。如果用户具有存在问题的鼠标、视力太差或者电动控制困难，则很难点中较小的或紧密的目标，即使是能够控制指向设备(在屏幕上四处移动光标的装置)的用户也会发现更容易点中较大的目标。此外，大多数用户都会发现移动的目标不容易点中——因此在大多数设计中尽量避免使用它们。

在创建菜单时，需要确保它能够工作于所有的主要浏览器中。随着Web的发展，许多菜单(特别是使用JavaScript语言编写的下拉菜单)甚至不能工作于一些常用的浏览器中。

有两种方法可以避免发生这种问题：

- 在许多不同类型的浏览器中测试编写的菜单(特别是在较老的浏览器版本中测试)。
- 避免在菜单中使用复杂的代码。

下拉菜单或者弹出式菜单(当用户将鼠标悬停在题头上时将出现的一些新的菜单项)特别容易出问题，原因包括两个方面：

- 它们经常以JavaScript语言编写，该语言的实现方式在不同的浏览器中稍有不同——特别是在较老的浏览器中。因此，虽然菜单可以在设计人员的浏览器中表现为正常工作，但一些其他访问者可能无法导航该站点。
- 它们可能太敏感或移动太快，以至于用户无法选择他们需要的项。

注意：

多年以来，我遇到很多尝试实现下拉菜单的站点无法工作于我的浏览器中的情况。因此，为了可用性原因，现在我完全避免使用这种菜单。

某些设计人员甚至采用一些处于试验阶段的菜单类型(特别是采用Flash技术的菜单)，它们通常需要非常灵巧的控制——当用户将鼠标悬停在某个菜单项上并移动到左边或右边时，这些菜单通常在菜单项之间移动和滑动。虽然那些能够很好控制指向设备的菜单在试验性站点上具有良好的外观，但它们会使无法很好地控制指向设备的用户不能正常浏览站点。因此，这些菜单最好只应用于一些试验性站点而不是正规的商业站点。

7. 菜单必须能够快速加载

当创建菜单时，不能期望站点的每个访问者都具有快速的Internet连接——某些访问者可能仍然使用拨号连接或者办公场所中很多用户共享的同一个连接。尽管连接速度不断提升，但是菜单也应当能够在几秒内加载。如果菜单的加载时间超过8秒，很多用户会认为该页面不能加载或者浏览器已经冻结——他们会尝试重新加载该页面，或者甚至更糟糕

地是，单击“返回”按钮或进入另一个页面。

对于在菜单中使用图形或 Flash 的设计人员来说，加载速度特别重要(纯文本的菜单的加载速度非常快)。如果希望在用户将鼠标指针悬停在图像上时改变该图像以使其突出显示，则加载时间将是正常时间的两倍(因为这将需要加载两幅图像以便切换)。

注意，某些浏览器在显示表之前需要完全加载该表的内容，因此如果在表中放置图像，则加载页面时用户可能需要等待相当长的时间。

8. 菜单在整个站点中必须一致

站点中包含的页面越多，则需要的导航项越多。一旦引入子菜单，则导航将变得非常复杂，并且不同页面中的导航将会改变。非常重要的是，主要导航在所有页面中需要保持一致。

站点中每个部分的子菜单应当位于每个页面的相同位置处，并且具有类似的外观，以方便用户确切地知道在何处导航站点。

10.2.2 链接

除了访问者用于导航站点的菜单之外，许多页面还在组成文档主体的文本中包含一些其他超链接。本节介绍关于不是主菜单一部分的链接的两方面主题：

- 文本链接
- 作为链接的图像

1. 文本链接

默认情况下，文本链接通常是蓝色并具有下划线。某些可用性方面的专家建议所有的链接应当保持它们的默认外观。但是，根据以往 Web 开发经验，对链接使用清晰区别于主文本的颜色将可以更容易看出哪些文本组成链接。

如第 7 章中所示，当用户将鼠标悬停在链接上或者用户已经访问过链接时，都可以改变链接的外观。下面的代码能够快速帮助您回忆如何使用 CSS 规则改变链接的外观(ch10_eg07.css)：

```
a {font-weight:bold; color:#ff0000; text-decoration:none;}
a:hover {color:#FF9900; text-decoration:underline; background-color:#f9f0f0;}
a:visited {color:#990000;}
```

当用户将鼠标悬停在这些链接上时，它们将具有下划线和改变的颜色，并且获得一种背景色。访问过的链接将具有不同的阴影，用于提醒用户他们已经到过哪些页面。如果运行本章下载代码中的示例，可以看到这种功能。

注意：

通常应该避免当用户将鼠标悬停在链接上时使用一种不同大小的文本，因为这会改变字体的宽度，从而使其难以阅读并改变文本行的宽度。

2. 作为链接的图像

图像经常用于菜单、广告、可单击的照片、图形图标等对象中的链接。当使用图像作为链接时，应该在图像中使用两种属性：

- alt = "description of image or text on image"：使用这个属性告诉不能看到图像的用户图像是什么或图像说明什么。
- title="where the link will take the user"：使用这个属性为用户提供一个工具提示，告诉用户链接将把其带到何处；屏幕阅读器也使用该属性。

如果不使用 CSS 控制作为链接的图像的边框(将它们设置为“无边框”)，则应该也添加 border 属性：

- border="0"：如果不使用 CSS 或者这个属性来控制边框，则在许多浏览器中图像的周围将存在边框，看上去不悦目。

在本章的前面存在一个使用图像作为链接的示例(ch10_eg06.html)。在第 12 章中，您将看到一个使用 JavaScript 创建称为翻转图像的示例，即当用户将鼠标悬停在图像上时图像会改变。

10.2.3　站点搜索功能

用户导航站点的第三种方式是使用搜索功能。搜索功能允许用户立即查找一些与他们希望站点中找到的信息相关的关键字(或单词)。搜索功能能够节省用户了解站点导航方案的时间，并且如果他们在查找所需信息时存在困难，则搜索功能提供了查找信息的另一种方式。

1. 搜索功能使站点更可用

随着站点的不断发展，搜索功能变得日益重要。如果站点仅具有少数页面，则能够很容易理解菜单。但是，对于较大的站点，它们可能会具有一些子菜单，此时并不是所有的选项都出现在每一个页面中，因此较大的站点能够真正地从搜索功能中获益。

可以通过多种方法在站点上实现搜索功能。虽然某些方法需要相当高级的编程经验，但是也可以应用一些方法添加相当简单的搜索功能。

对于内容存储在数据库中的大型商业站点来说，可以使用称为查询的编程命令来询问数据库哪些页面中包含了用户希望搜索的项，或者可以使用一种特殊的索引程序来索引站点，从而方便地实现搜索功能。

对于不使用数据库或者索引工具的站点，添加搜索功能的最简便方法是使用一种第三方搜索实用程序来索引站点。这些搜索服务也提供用于创建搜索框的代码，搜索框会将查询发送给服务的站点。当站点访问者使用搜索框时，他们的查询被发送给提供搜索服务的公司的服务器，服务器然后将答案以站点的名义返回给用户。

提供这种服务的最著名公司是 Google，并且到编写本书时为止免费提供此种服务(Google 的收入来源于与搜索结果一起提供的广告——但是从图 10-7 中可以看出，这些广告并不是很烦人：它们仅出现在结果页面的右边，如同用户向 Google.com 站点发送查询一样)。

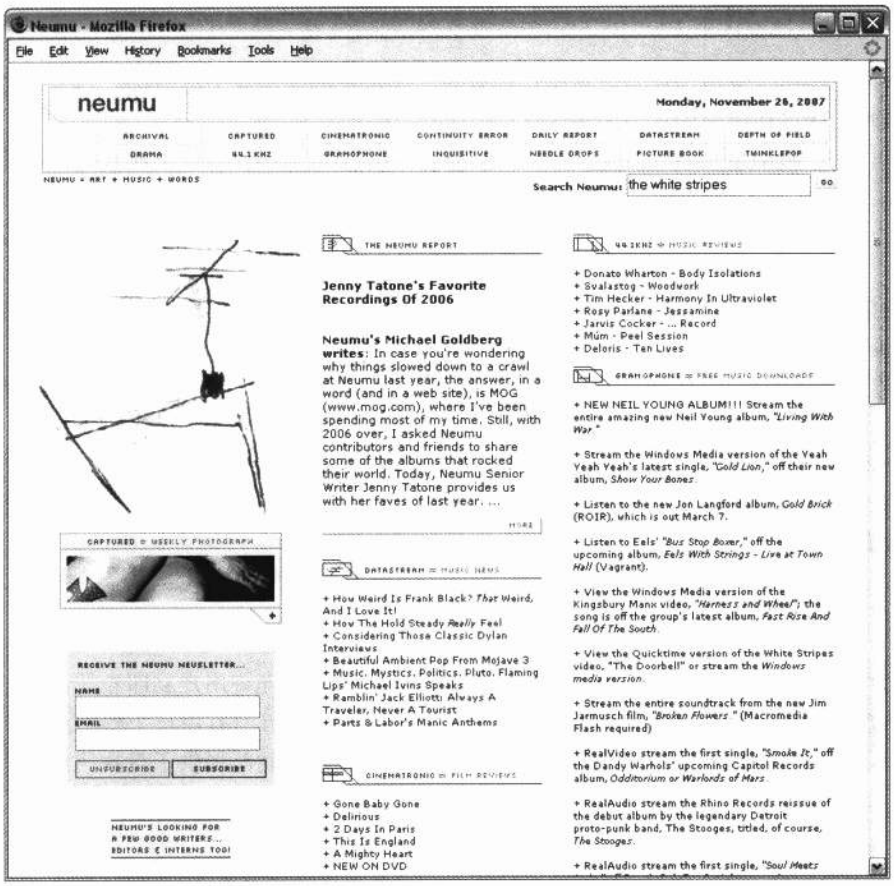


图 10-7

2. 在站点中添加 Google 搜索

Google 是当前 Internet 上最广泛使用的搜索引擎，它提供非常强大和灵活的服务，可以使用它的搜索引擎在自己的站点中提供搜索功能。在编写本书时，用户在使用这种服务时必须注册。但是，相关的指令和站点上的设置过程非常简单，并且该服务是免费的。

图 10-7 显示了称为 Neumu.net 的艺术和音乐站点如何在导航栏下方添加一个小搜索框。

当这个站点的访问者搜索 Neumu 站点时，请求被发送给 Google。Google 然后生成一个页面(该页面中包含 Neumu 站点中具有搜索关键字的所有项)，并且将该页面发送给访问者。这些结果指向 Neumu 站点，如图 10-8 所示。

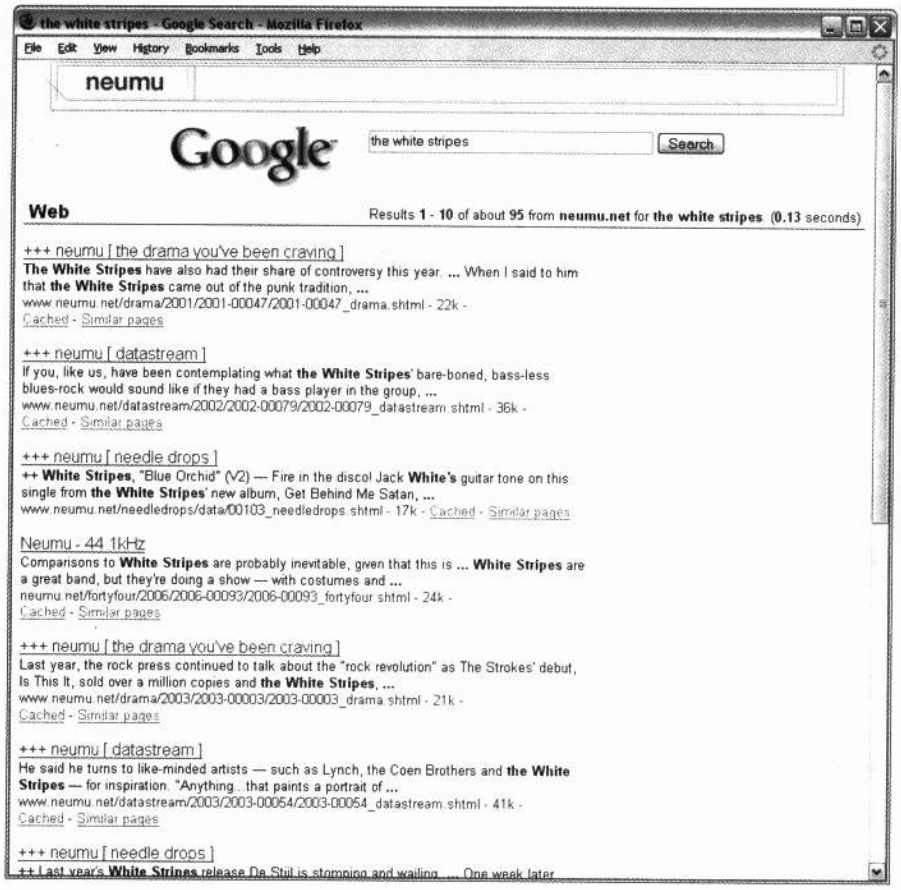


图 10-8

通过选择浏览器菜单中的 View Source 选项，可以查看该搜索框的代码。

说明：

为了获得搜索功能，不能只复制这些代码，还需要亲自注册该服务。

另一个称为 www.Atomz.com 的站点提供有限页面数量的免费搜索服务，但是对一些较大的站点收费。

10.3 在表的多行中添加阴影

当处理包含多行信息的表时，如果轻微变换行的背景色，则能够让访问者更容易识别表的每一行。图 10-9 给出了一个表示例，该表中的每一行具有交替的颜色。

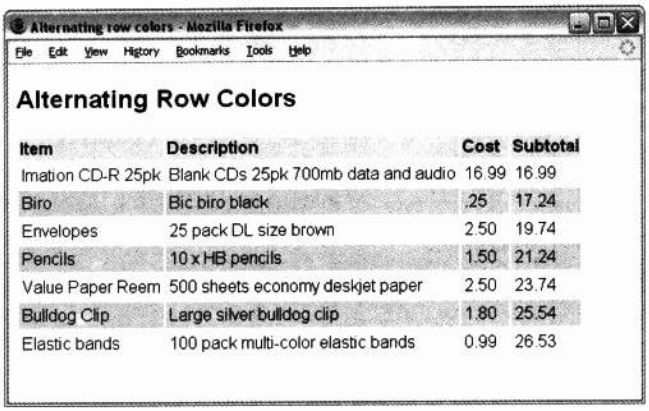


图 10-9

通过对表行使用 odd 类和 even 类实现这种效果，如下所示(ch10_eg08.html)：

```
<table>
  <tr>
    <th>Item</th>
    <th>Description</th>
    <th>Cost</th>
    <th>Subtotal</th>
  </tr>
  <tr class="even">
    <td>Imation CD-R 25pk</td>
    <td>Blank CDs 25pk 700mb data and audio</td>
    <td>16.99</td>
    <td>16.99</td>
  </tr>
  <tr class="odd">
    <td>Biro</td>
    <td>Bic biro black</td>
    <td>.25</td>
    <td>17.24</td>
  </tr>
  <tr class="even">
    <td>Envelopes</td>
    <td>25 pack DL size brown</td>
    <td>2.50</td>
    <td>19.74</td>
  </tr>
</table>
```

下面是这个示例的 CSS 代码：

```
body{
  color:#000000; background-color:#ffffff;
  font-family:arial, verdana, sans-serif; font-size:12pt;}
```

```
th {font-weight:bold; text-align:left; background-color:#fff336;}
.odd {background-color:#d6d6d6;}
.even {background-color:#ffffff;}
```

记住，无论如何使用背景色，背景和文本之间都必须具有较强的对比度，以便用户能够方便地阅读文本。本示例中非常亮的灰色是一种很好的颜色示例，该颜色不会严重影响表本身的可读性。

10.4 表单

我从来没有遇到过喜欢填写表单的人——特别是在 Web 上。因此，如果站点必须包含一个表单，则优秀的设计是必须的要素，否则人们不会愿意填写表单(例如，对于在线商城站点，表单是核心业务的必须部分)。

本节将介绍以下内容：

- 在设计表单之前执行的工作
- 如何设计表单，如何选择正确的表单控件，如何将表单控件正确地分组，如何标记表单控件
- 如何最合适地布局表单

为了介绍本节中的关键点，将给出一个包含基本表单的示例，用户在注册联机服务之前必须填写该表单。

10.4.1 设计表单之前的工作

在处理表单的外观之前，需要做一些准备工作——如同在开始设计站点之前需要进行准备一样，但是这个过程花费的时间较少。

1. 首先列举必需的信息

当设计一个表单时，应当首先创建需要从用户处获得的信息的完整列表。可以从常规列表开始，该列表中包括注册详情、姓名、邮寄地址和 e-mail 地址等项，但是需要确保知道该列表中组成这些需求的每一项。例如，是否需要独立地获得用户的名和姓？如果需要，则在表单中将具有独立的对应项。组成地址的项有哪些：门牌号/名称、街道的名称、市区的名称、邮政编码等？哪一项需要独立于其他项？

下面是当前注册表单中需要的信息列表：

- 注册信息
- 用户的姓名
- 用户的地址
- 用户的联系详情

确定该列表中的信息之后，需要的准确信息如下所示：

- 注册信息：用户名和密码
- 姓名：名和姓

- 地址：街道地址、城市的邮政编码
- 联系信息：e-mail 地址、区号、电话号码

当创建表单时，仅需要询问完成某项工作所必须的信息。当从访问者处收集信息时，应该避免向访问者询问过多问题；表单越长，用户将越不会喜欢填写它。

如果希望收集许多不重要的信息(例如，统计站点访问者的组成状况)，则可以考虑为注册/购买产品后参与和回答问题的用户提供奖励，例如一个抽奖的入口。

说明：

当收集和存储顾客的信息时，必须确保满足所在国家的数据保护法。

2. 分组相关的信息

知道希望从站点访问者处收集哪些信息之后，需要查看是否存在这些信息的逻辑分组，以便帮助访问者更好地理解该表单。

如果找到相关信息的分组，则需要确保该信息组中的项在表单中位置接近。在本节的示例中，需要 3 组信息：

- 姓名和 e-mail 地址
- 注册详情
- 其他联系详情

在这个示例中，分组与所需信息的初始列表相同，但是有时分组可能与初始列表区别较大。

3. 模仿用户熟悉的纸质表单

如果创建的联机应用程序中的表单是用户前面已经填写过的熟悉的纸质表单，则需要确保联机表单反映纸质表单(注意，如果用户不熟悉该表单，则没有必要模仿纸质表单)。如果应用程序的目标是将现有的软件联机，则也可以对该软件模仿。

将表单模仿为用户熟悉的对象的原因是显而易见的；这将使用户更容易填写表单。这并不是说表单的布局必须与纸质表单完全相同(纸质表单中的许多问题通常拥挤在一起)，而是以类似的顺序和分组询问相似的问题。

4. 用户是否每次提供相同的信息

每次用户访问站点时必须提供相同的信息吗？是否需要将一些数据存储在数据库中(或其他应用程序中)，并且在用户再次登录时检索这些数据？例如，如果处理的是在线商店，在用户登录之后，应用程序将记住用户的姓名、地址和联系方式吗？

注意：

如果将要存储用户的信息——特别是他们的信用卡详情——则必须确保遵守所在国家存储这些信息的相关法律。

另外还需要考虑表单的处理方式。如果由人来处理表单，则这些人需要能够解释用户输入的数据，相反如果信息直接存入数据库，则用户必须确保他们输入的信息的精确性。

这会影响收集信息所需要的表单控件的选择。

5. 是否有其他需要出现在表单上的信息

有些表单包含一些额外信息，例如运输信息、价格列表、注意的法律事项等。在开始设计表单之前，应当清楚了解放置在它上面的所有信息，而不仅是了解表单控件本身。

10.4.2　设计表单

现在您已经知道表单需要捕获什么信息，可以开始设计它。首先可以选择一些适当的控件类型，然后将它们分组在一起并标记它们。最后考虑表单的布局以控制它的外观。

1. 选择表单控件的类型

在第 5 章中介绍了可以使用的不同类型表单控件。选择正确类型的表单控件以收集信息非常重要。确定为每部分信息使用哪些表单控件之后，您将会了解表单的可能长度和布局。

输入文本：

- 如果只有一行文本，则使用<input>元素，并且该元素的 type 属性的值为 text。
- 如果希望用户输入多行的文本，可以使用<textarea>元素。
- 如果信息是敏感的(例如信用卡或密码)，则使用<input>元素，并且该元素的 type 属性的值为 password。

为用户提供有限的选择项：

- 如果用户仅能够(从几个选项中)选择一个选项，则使用一组单选按钮(这些单选按钮具有相同的名称)或者一个下拉选择框。
- 如果用户可以选择多项，则使用复选框或者多项选择框。

也需要考虑访问者习惯于使用哪种方式给出这种类型的信息。例如，通常使用一系列的文本输入框作为地址的每一行，而不是使用文本输入框输入街道名，而使用选择框(作为地址的第一行)表明街道是一条街、路或者林荫道。

记住，每一个表单控件应当使用一个名称来描述它的内容。不是随意命名为 input1 或 input2，您经常会看到一些表单控件名具有某个前缀，用于描述它们是哪种类型的控件：

- txtName 用于文本框和文本区域
- radName 用于单选按钮
- chkName 用于复选框
- selName 用于选择框

单选按钮和复选框

尽管单选按钮和复选框占用的空间多于选择框，但是它们能够更容易被访问者使用，因为用户可以一次看到所有的选项(前提是选项列表不能太长，例如世界上所有国家的列表，此时可以使用下拉选择框代替)。

如果仅有 3 个或 4 个选项，并且仅允许用户选择其中一个选项，则相比于选择框，单选按钮通常是较好的选择，因为它的所有选项都是可见的。这种规则的例外情况是当设计中包含多个选择框时(此时设计的一致性更为重要)。

如果仅有 3 个或 4 个选项，并且允许用户选择多个选项，则对多个选项使用复选框通常好于使用一个多项选择框，无论这些复选框将占用多少空间——这不仅是因为复选框更常用，而且因为如果使用一个多项选择框，则通常必须告诉用户他们能够选择多项以及如何选择。

如果用户不得不表明他们赞同或者已经阅读了什么内容(例如一些条款和条件)，则复选框也是理想的选择。在这种情况下，使用复选框而不是单选按钮非常重要。当选择一个单选按钮时，可以通过选择其他的单选按钮来改变选择，但是无法取消选择一个组中的所有单选按钮(而对于复选框来说，可以通过单击同一个复选框来取消对它的选择)。

注意：

一定不要使用编程语言(例如 JavaScript)来更改单选按钮或复选框的意图。换句话说，不应该使复选框互斥(像单选按钮一样)，并且不应该允许用户从一组单选按钮中选择多个单选按钮，因为这将使期望单选按钮或复选框遵循它们的普通默认行为的用户感到迷惑。另外，不应该在同一个表单中重复地混合使用单选按钮和选择框，否则用户可能会感到迷惑。

相比于选择框来说，单选按钮和复选框能够向用户提供更多的信息。单选按钮或复选框旁边可以有较长的文本描述，而如果在选择框中使用较长的描述，则整个选择框将变得非常宽。在图 10-10 中可以看到较长的下拉列表框(超出屏幕)和一组单选按钮的示例(ch10_eg09.html)。

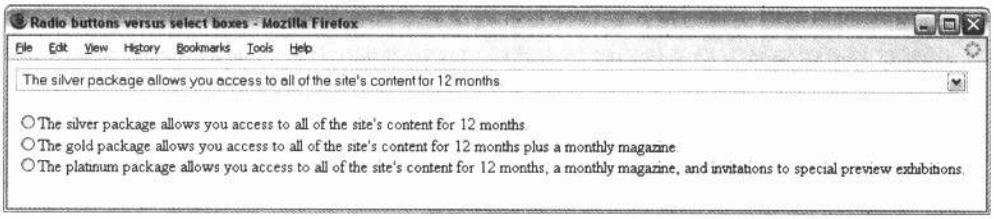


图 10-10

如果单选按钮表示一个可选的问题，则不能自动地选择一项作为默认值。不能像对待复选框那样，通过单击取消选择所有的单选按钮；只可以选择不同的单选按钮。如果用户无法选择给出的任意一个选项，则为用户提供一个“其他”选项通常非常有帮助。

选择框

选择框也称为下拉列表框，它能够节省屏幕空间，特别是当存在很多选项时，但如图 10-10 所示，如果每个选项的描述太长，则选择框看上去不是很美观。事实上，选择框的宽度等于其中最宽选项的宽度。

需要注意的是，当为用户提供选择框时，必须包含用户需要的所有选项。例如，如果使用一个下拉列表框表示美国的各个州，并且存在来自于美国之外的访问者，则必须至少存在一个选项供不在美国的人使用，即使该选项只是“Outside U.S.”。

选择框中各项的顺序应当反映用户的体验：例如，如果使用月的名称，则按照时间顺序排序；而如果使用州或国家的名称，则按照字母表顺序排序列表更易于使用。

如果较长列表中的一个(或多个)选项更流行或更可能被选择，则应当将这些选项放置在选择框的顶部，以便用户最先看到它们。

文本框

文本框是用户提供大多数信息的最自然方式。通常来说，文本区域应当足够大，以便用户在不显示滚动栏的情况下输入他们所需的文本(除非文本非常长，例如一封 e-mail 或者一篇文章)。

注意，用户通常将文本框的大小作为他们应当提供的文本长度的指示。这对于日期一类的文本来说非常有用，如图 10-11 所示，其中希望用户输入用于表示年份的 4 个数字。

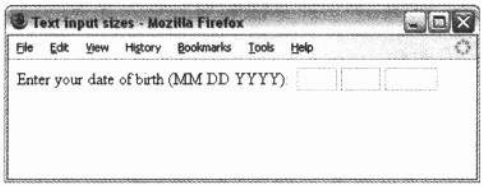


图 10-11

2. 控件的分组

确定使用哪些表单控件之后，可以开始将它们放置到页面中。本章前面已经提到，应该将这些控件分组为相关的信息项——并且这些组应当反映用户对主题的理解。

可以采用如下方式分组表单元素：

- 字段集
- 标签
- 将表单拆分为多个页面

也可以使用本章第一节中介绍的内边距或阴影，该节主要介绍空白。

使用<fieldset>元素

在第 5 章中已经介绍了<fieldset>元素，使用该元素可以在起始标签<fieldset>和结束标签</fieldset>之间分组表单的各个部分。表单也可以附带一个<legend>元素，用于指明该框的标题。

例如，下面的表单用于输入用户的注册详情(ch10_eg11.html)：

```
<form name="frmLogin" action="login.aspx" method="post">
  <fieldset>
    <legend>Login</legend>
      User name: <input type="text" size="12" name="txtUserName" /><br />
      Password: <input type="password" size="12" name="txtPassword" /><br />
```

```
    Confirm password: <input type="password" size="12"
            name="txtPasswordConfirmed" /><br />
    <input type="submit" value="Log in" />
  </fieldset>
</form>
```

字段集在 IE4 和 Netscape 6 浏览器中引入。较老的浏览器如果不能理解<fieldset>和<legend>按钮，则会忽略它们，因此可以安全地将这些元素添加到所有表单中。这个示例的外观如图 10-12 所示。

图 10-12

也可以选择使用字段集的替代对象分组表单的各个部分，例如换行符、背景色或与某些样式规则相关联的表，但字段集是专门为分组表单元素而引入的，并且可以将样式与<fieldset>元素关联，如同本示例中所示(ch10_eg11.css)：

```
fieldset {
  width:250px;
  padding:10px;
  font-size:12px;
  text-align:right;}
```

注意样式表中 width 特性的设置方式。这对于添加到<fieldset>元素非常有帮助，因为如果不这样设置，它们将伸展为整个浏览器窗口(或包含元素)的宽度。

将表单拆分为多个独立的页面

较长的表单不仅使用户厌烦，而且使用户难以填写。如果正在编写验证和错误处理机制(例如错误消息，用于告知某个表单字段没有填写或者包含了错误类型的信息)，则随着表单变长，代码将变得非常复杂。因此，如果具有较长的表单，可以将其拆分为多个页面。执行该操作的原因包括以下方面：

- 表单越小越容易被人接受。
- 将相关信息放置在相同的页面上可以较为容易地理解这些信息。

作为通用的指导原则，表单的大小不应该超过整个屏幕(1024×768 分辨率)，从而用户不需要频繁滚动页面。

如果将表单拆分为多个独立的页面，则应当向用户指明他们当前正处于表单的哪个部分。图 10-13 中包含拆分为 4 个页面的一个表单以及一个确认页面。

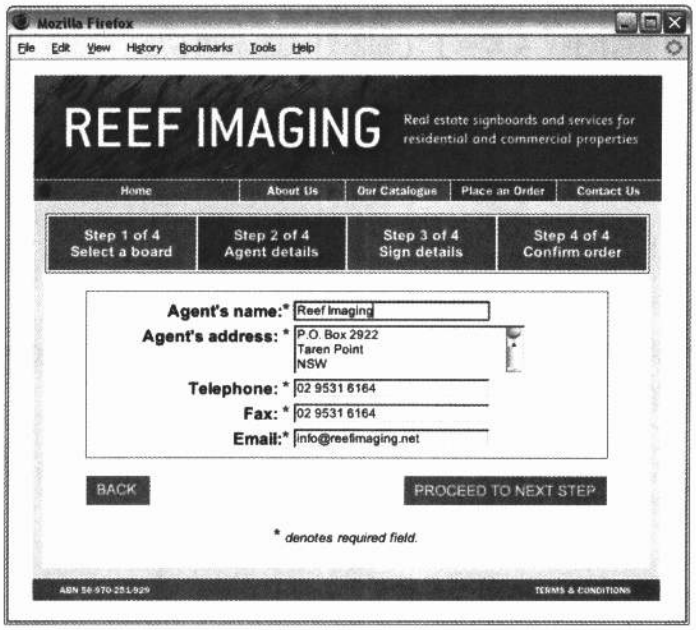


图 10-13

将表单拆分为多个页面会对编程工作带来新的复杂性，因为程序必须记住用户在每个表单之间已经输入的内容；但是，存在好几种方式可以完成该功能，并且只需要少量的额外工作。通常可能希望用户依次按步骤填写表单，而不是允许他们随意地选择页面，因此避免使用允许他们在任意页面之间随意跳转的链接。

为问题编号

如果具有很多问题，如同应用程序表单或在线测试中一样，则应当将这些问题编号，以便用户知道问题的开始和结束位置。如果希望指示用户应当跳转到表单的另一部分，则这种做法也非常有用，因为这样能够显式地指示用户应当跳转到哪个编号问题。

3. 表单的布局

理想情况下，表单的布局应当能够反映用户处理该数据时期望看到的内容。布局与用户对纸质表单或软件中表单的体验相关。甚至可以考虑日常生活中的体验，例如用户填写地址的方式(通常将地址写在几个独立的行中，而不是使用下拉列表框)。

标记控件

关于表单布局的首要问题是标记控件。每一个控件具有清晰的标签非常重要，从而用户可以知道他们应当填写什么样的信息以及在何处添加该信息。存在两种类型的标签：

- 隐式标签，控件附近的普通文本和标记
- 显式标签，利用<label>元素赋予的标签

下面的指导原则指出了元素的标签通常应当出现的位置：

- 文本输入字段：标签位于输入框的左边或正上方
- 复选框和单选按钮：标签位于复选框或单选按钮的右边
- 按钮：标签位于按钮自身上——即按钮的值

隐式标签是标记控件的最简单方式。为了添加隐式标签，可以简单地在当前讨论的控件旁边添加文本。例如(ch10_eg12.html)：

```
First name: <input type="text" name="txtFirstName" size="12" /> <br />
Last name: <input type="text" name="txtLastName" size="12" /> <br />
E-mail address: <input type="text" name="txtEmail" size="12" /> <br />
<input type="submit" value="subscribe" />
```

这种方法的缺点是它的外观不是非常吸引人——特别是在表单较长时——因为表单控件相互之间没有很好地对齐，如图 10-14 所示。

图 10-14

<label>元素需要少量的额外编程工作，但是经常使用该元素是一种很好的习惯。在第 5 章中介绍过，<label>元素必须包含表单控件，或者使用 for 属性，并且该属性的值是表单控件的 id 属性值：

```
<label for="firstName">First name: </label>
<input type="text" name="txtFirstName" size="12" id="firstName" />
<label for="lastName">Last name: </label>
<input type="text" name="txtLastName" size="12" id="lastName" />
<label for="email">E-mail address: </label>
<input type="text" name="txtEmail" size="12" id="email" />
```

遗憾的是，这个示例的显示效果类似于图 10-14 中的示例，但是<label>元素确实有其优点：

- 更易于屏幕阅读器将控件与它的标签关联在一起。特别是当标签与相关的控件不相邻时，该元素能够将它们关联在一起——例如，在一个表中，标签可能出现在表单控件上方的行中。
- 标签能够增加单选按钮或复选框的可单击区域。对于这些用户难以准确单击的控件来说，这种功能非常有用，因为用户可以单击标签。

注意：

只有在 IE4 浏览器和 Netscape 6 浏览器以及更新的版本中才支持标签；但是，一些较老的浏览器会忽略<label>元素并显示它们的内容，因此可以安全地在任何表单中使用它们。

下面示例中的表单允许用户表明他们如何知道某个公司。当用户单击标签时，与其关联的单选按钮将被选中(ch10_eg13.html)：

```
<form name="frmExample" action="" method="post">
  <fieldset>
    <legend>How did you hear about us?</legend>
    <input type="radio" id="referrer1" name="radReferrer" value="Mouth" />
    <label for="referrer1" >Word of Mouth</label><br />
    <input type="radio" id="referrer2" name="radReferrer" value="Google" />
    <label for="referrer2" >Google Search</label><br />
    <input type="radio" id="referrer3" name="radReferrer"
    value="Magazine Ad" />
    <label for="referrer3" >Magazine Ad</label><br />
    <input type="radio" id="referrer4" name="radReferrer" value="Other" />
    <label for="referrer4" >Other</label>  
    <input type="text" value="txtOther" size="12" /><br />
    <input type="submit" value="Submit" />
  </fieldset>
</form>
```

这个表单的外观如图 10-15 所示。

图 10-15

注意：

当为表单选择提示或标签时，必须选择对用户确实有意义的单词。有些对于您来说显而易见的对象，对于不熟悉该主题的访问者来说可能并不清晰——例如，productId 编号可能是产品的唯一标识号，但是顾客可能不知道该标识号或者在何处查找它。

使用表进行布局

许多设计人员和程序员使用表进行布局并将表单元素定位在他们的 Web 页面中。但是，必须记住表在屏幕阅读器中的线性化方式(第 4 章中讨论过该主题)。

为了提醒您屏幕阅读器如何线性化简单的表，请查看表 10-1。

表 10-1

Row 1 Column1	Row 1 Column 2
Row 2 Column 1	Row 2 Column 2
Row 3 Column 1	Row 3 Column 3

这个表中的单元格通常按照如下顺序逐行读取：

```
Row 1 Column 1, Row 1 Column 2, Row 2 Column 1, Row 2 Column 2,
Row 3 Column 1, Row 3 Column 2.
```

因此，在表中布局前面示例的正确方式如下所示(ch10_eg14.html)。注意，这个示例没有使用<label>元素，从而可以理解在不使用<label>元素的情况下读取元素的顺序：

```
<table>
  <tr>
    <td class="formPrompt">First name: </td>
    <td><input type="text" name="txtFirstName" size="12" /></td>
  </tr>
  <tr>
    <td class="formPrompt">Last name: </td>
    <td><input type="text" name="txtLastName" size="12" /></td>
  </tr>
  <tr>
    <td class="formPrompt">E-mail address: </td>
    <td><input type="text" name="txtEmail" size="12" /></td>
  </tr>
</table>
```

这将正确地排序元素，并且使用屏幕阅读器的用户将理解该表单。注意，用作标签的<td>元素上的 class="formPrompt "与一个 CSS 样式表规则(该样式表规则指明文本在表中应该右对齐)关联。这就使页面上的显示更整洁，并且防止标签和它关联的控件之间出现较大的间距。结果如图 10-16 所示。

图 10-16

对于更为复杂的表，则需要考虑许多事项。例如，查看如图 10-17 所示的表单。

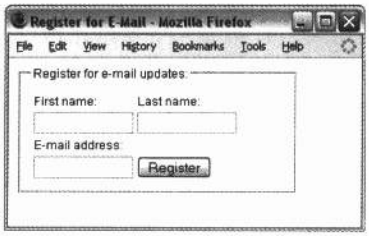


图 10-17

图 10-17 中具有两列表单控件，并且标签位于元素的上方。这种设计必须使用<label>元素；否则屏幕阅读器将首先读取第一行中的两个标签，然后读取第二行中的两个表单控件(ch10_eg15.html)：

```
<table>
  <tr>
    <td><label for="fname">First name:</label></td>
    <td><label for="lname">Last name:</label></td>
  </tr>
  <tr>
    <td><input type="text" name="txtFirstName" id="fname" size="12" /></td>
    <td><input type="text" name="txtLastName" id="lname" size="12" /></td>
  </tr>
  <tr>
    <td><label for="email">E-mail address:</label></td>
    <td></td>
  </tr>
  <tr>
    <td><input type="text" name="txtEmail" id="email" size="12" /></td>
    <td><input type="submit" value="Register" /></td>
  </tr>
</table>
```

但是，通常更好的方式是坚持在一列中放置表单控件——并且最好使用 CSS 而不是表来控制表单的布局。虽然纸质的表单通常在多列中显示问题，但在 Web 使用多列表单控件并不合适，原因如下所示：

- 无法知道用户屏幕的大小，并且用户可能无法看到第二列(特别是少数屏幕分辨率仍然是 800×600 的用户)。
- 用户很有可能遗漏填写表单中的一些项。
- 开发人员不得不使用复杂的布局，这将可能会混淆使用屏幕阅读器的人群。

将相关信息放置在表单控件的左右相邻处或上方

现在您已经了解到优秀的标签对于用户的理解至关重要，因此下面的示例给出一些需要额外注意的标签位置。查看图 10-18 中所示的示例，该表单用于输入电话号码。

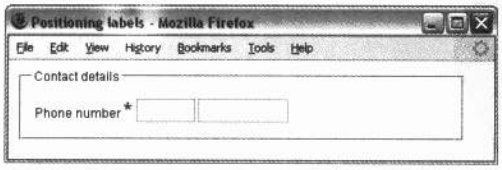


图 10-18

此处没有指示每一个独立输入框的用途。虽然可以猜测其中一个输入框用于输入区号，另一个输入框用于输入电话号码的主要部分，但是使用屏幕阅读器的用户很可能会对第二个输入框的用途感到迷惑，因为他们仅能够听表单，而不能查看表单。某些用户，特别是性急的用户，可能会将整个电话号码输入在一个文本框中。

对于这个示例，更好的方法是分别指示区号和电话号码的标签，如图 10-19 所示。

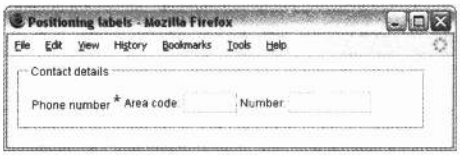


图 10-19

此时该表单显得非常清晰，它的代码如下所示(ch10_eg16.html)：

```
<table>
  <tr>
    <td class="label">Phone number <span class="important">*</span></td>
    <td>Area code<input type="text" name="txtTelAreaCode" size="5" />
       Number<input type="text" name="txtTelNo" size="10" /></td>
  </tr>
</table>
```

当使用多个单选按钮或多选按钮表达一个选项或等级时，适当地添加标签也非常重要。图 10-20 中的示例存在一些问题。

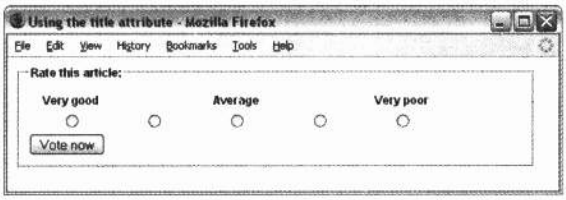


图 10-20

这个示例的代码将单选按钮和标签放置在一个表中。这个示例存在的问题是标签与正确的单选按钮不相关——在下面的代码中使用<label>元素并将其与正确的表单元格关联。问题是使用屏幕阅读器的用户将仅能够听到 3 个选项的标签，而实际上存在 5 个选项以供选择。应当分别为每一个选项提供一个标签。

```
<table>
  <tr>
    <td><label for="VeryGood">Very good</label></td>
    <td></td>
    <td><label for="Average">Average</label></td>
    <td></td>
    <td><label for="VeryPoor">Very poor</label></td>
  </tr>
  <tr>
    <td><input type="radio" name="radRating" value="5" id="VeryGood"
    /></td>
    <td><input type="radio" name="radRating" value="4" id="Good" /></td>
    <td><input type="radio" name="radRating" value="3" id="Average" /></td>
    <td><input type="radio" name="radRating" value="2" id="Poor" /></td>
    <td><input type="radio" name="radRating" value="1" id="VeryPoor" /></td>
```

```
  </tr>
</table>
```

如果确实不希望为每一项提供一个文本替代项，则可以使用相当极端的替代项，即将一个单像素、透明的 GIF 图像用于<label>元素中的 alt 文本，该图像不会在向使用屏幕阅读器的人员解释每个选项的浏览器中显示(ch10_eg17.html)，如下所示：

```
<table>
  <tr>
    <td><label for="VeryGood">Very good</label></td>
    <td><label for="Good"><img src="images/1px.gif" alt="This option
        has no label its value is good" /></td>
    <td><label for="Average">Average</label></td>
    <td><label for="Poor"><img src="images/1px.gif" alt= "This option
        has no label its value is poor" /></td>
    <td><label for="VeryPoor">Very poor</label></td>
  </tr>
  <tr>
    <td><input type="radio" name="radRating" value="5" id="VeryGood" /></td>
    <td><input type="radio" name="radRating" value="4" id="Good" /></td>
    <td><input type="radio" name="radRating" value="3" id="Average" /></td>
    <td><input type="radio" name="radRating" value="2" id="Poor" /></td>
    <td><input type="radio" name="radRating" value="1" id="VeryPoor" /></td>
  </tr>
</table>
```

无法实际地看到这个示例与前一个示例的区别，但是如果不能查看页面并依赖于屏幕阅读器，则将能够听出其中的区别。

必须填写的信息

表单中通常包含一些用户必须回答以正确处理的问题。如果某个表单控件必须填写，则应当告诉用户这一点。通常使用星号(*)指示用户某些字段必须填写，当然也会在页面中包含一条注释以表明星号的意义。此外，通常将星号设置为与其周围主要文本不同的颜色(例如红色)，从而用户能够了解它的重要性。

```
First name <span class="required">*</span>:
<input type="text" name="txtFirstName" size="12" />
```

required 类可用于如下所示的 CSS 规则中：

```
span.required {
  font-weight:bold;
  font-size:20px;
  color:#ff0000;}
```

在图 10-18 和图 10-19 中可以看到相应的示例，即电话号码示例的屏幕截图。在完整的表单中，还应当包含一条提示信息，用于以文本方式表明星号的意义。

仔细放置按钮

在页面中放置按钮时应该非常谨慎。按钮应当与表单的相关部分接近；例如，在一个

在线商铺中，用于购买某件产品的按钮应当接近该产品，如图 10-21 所示。

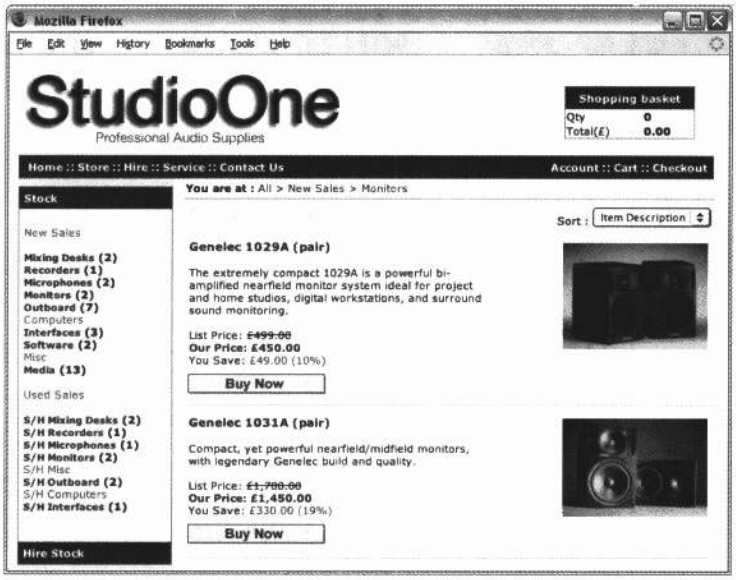


图 10-21

如果在表单中使用 Next、Proceed 或 Submit 按钮——例如，为了链接表单的不同页面以指示用户应该继续执行下一步——这些按钮应当出现在页面的右边；Back 按钮应当出现在左边。这对应于用户使用浏览器窗口中的 Back 和 Forward 按钮的体验(Back 是左边第一个按钮，Forward 位于它的右边)。这也遵循访问者阅读采用从左到右排列的语言编写的文本的方向(因此当用户到达表单的末尾时，他们应当从左到右重新阅读)。图 10-21 给出了一个这样的示例。

注意：

如果使用 Reset 或 Clear 表单按钮，它们应当位于 Submit 或 Next 按钮的左边，因为这反映了 Web 用户以前的体验。

在表单控件中使用 title 属性

为用户添加额外信息的一种方式是在表单控件中使用 title 属性。当用户将光标放置在表单控件上时，title 属性的值将作为工具提示出现。这有助于阐明用户必须输入的信息的类型。

例如，下面的文本输入框中要求用户输入一个授权码。title 属性说明授权码来自于何处(ch10_eg18.html)：

```
<form name="frmExample" action="" method="post">
  <fieldset>
    <legend>Enter your authorization code</legend>
    Code: 
    <input type="text" name="txtAuthCode" title="Enter the authorization
    code that was e-mailed when you registered." /></td>
```

```
  </fieldset>
</form>
```

结果如图 10-22 所示，当用户将鼠标悬停在文本输入框上时将显示工具提示。

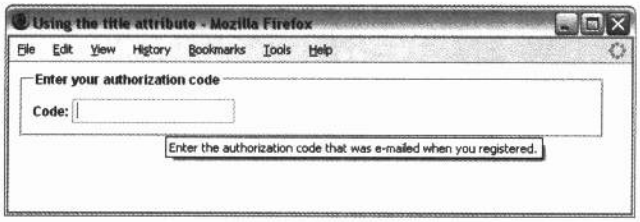


图 10-22

制表符索引

创建表单之后，应当检查表单元素的焦点移动顺序。用户应当能够使用 Tab 键在表单控件之间移动。如果表单控件获得焦点的顺序不是期望用户填写表单的顺序，则应当使用 tabindex 属性，该属性的值为 0 到 32 767 之间的整数(第 5 章中详细介绍过这个属性)。

tabindex 属性可用于下面的元素中：

```
<a> <area> <button> <input> <object> <select> <textarea>
```

在第 12 章的一个示例中将会看到，当页面加载时可以自动将焦点赋给一个表单元素。并且，您也将看到如何影响当前具有焦点的表单控件的外观。

不要依赖颜色来传递信息

虽然颜色在帮助理解表单方面是非常强大的工具，但是不应该单独依赖颜色来传递信息，并且必须确保颜色之间存在足够的对比度以使区别清晰。

例如，不能只使用颜色指明表单中哪些字段必须填写。在图 10-23 中，表单使用颜色指明哪些字段必须填写，但是因为本书的打印色是黑色和白色并且表单使用颜色传递信息，因此无法方便地看出哪些项必须填写。

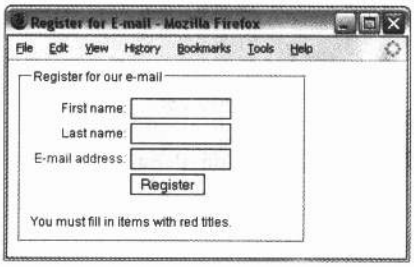


图 10-23

注意：

这是一个重要的问题，因为相当一部分的人群是某些形式的色盲。

但是，可以方便地解决这个问题。如图 10-24 所示，使用星号和颜色同时标注必须填写的字段。

图 10-24

注意：

一种测试是否过多依赖颜色的优秀方法是将显示器的颜色设置改为灰度级(在 Windows 操作系统中使用 Control Panel 窗口中的 Display Properties 选项，而在 Mac 操作系统中使用 System Preferences 窗口中的 Display 设置)。当页面处于灰度级时，如果信息丢失，则表明过多依赖颜色。

对表单元素使用 CSS

使用 CSS 来控制表单元素已经变得越来越流行，特别是控制文本输入框、文本区域和 Submit 按钮的边框与背景色，以便创建更加样式化的表单。虽然在 PC 计算机上使用 CSS 非常先进，但是 Mac 平台对在表单元素中使用 CSS 的支持并不是很好。

图 10-25 给出了一个表单，它的文本输入框具有黑色实线边框和亮灰色背景。

图 10-25

下面是与<input>元素关联的 CSS 样式(ch10_eg20.html)：

```
input {
  border-style:solid;
  border-color:#000000;
  border-width:1px;
  background-color:#d6d6d6;}
```

如果对表单元素使用样式，则必须确保不能因为添加一些不必要的样式而使表单更加难以填写。与任何类型的文本一样，如果文本控件之间没有很好的对比度，则它们将难以阅读，并且用户可能输入不正确的信息。

测试表单

布局完表单之后，需要对其进行测试。在第 13 章将更多地介绍测试站点方面的内容。但是，当设计完表单之后，如果能够观察使用该表单的人如何与其进行交互，则将非常有帮助。

执行这项工作的最重要的(也是必须记住的)方面是，如果看到用户即将犯错误，也不要打断他们；观察用户执行的操作，因为这将表明用户对表单的期望工作方式。

试一试　站点注册表单

在这个示例中，将为 Web 站点创建一个简单的注册表单。需要使用表 10-2 中列举的表单控件收集相应的信息。

表 10-2

信　息	表 单 控 件	是否必须填写
名	文本输入框	是
姓	文本输入框	是
e-mail 地址	文本输入框	是
访问站点的密码	密码文本输入框	是
密码确认	密码文本输入框	是
注册	Submit 按钮	N/A

图 10-26 给出了完成后的表单外观。

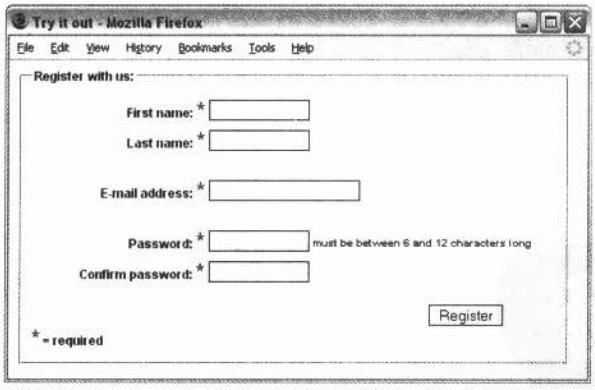


图 10-26

(1) 建立文档的程序框架，现在您可能已经非常熟悉如何执行该操作。不要忘记链接 CSS 样式表 registration.css。也可以添加一个<form>元素：

```
<html>
  <head>
    <title>Try it out</title>
    <link rel="stylesheet" type="text/css" href="registration.css" />
```

```
  </head>
  <body>
  <form name="frmExample" action="register.aspx" method="post">
  </form>
  </body>
</html>
```

(2) 该表单仅包含 5 个文本输入框，因此不需要将其拆分为多个独立的组，但是如果将所有文本输入框放置在一个具有<legend>元素的<fieldset>元素中，也是一种不错的想法。

```
<form name="frmExample" action="" method="post">
<fieldset>
  <legend>Register with us:</legend>
</fieldset>
</form>
```

(3) 表单剩余的部分位于<fieldset>元素内的表中。该表中具有两列，一列用于标签，另一列用于表单控件，并且每一列中的单元格需要具有独立的 class 属性。

包含标签的单元格的 class 属性值将是 label，而包含表单控件的单元格的 class 属性值将是 form。使用<label>元素指定每个表单元素的标签。每个表单控件都必须填写，因此可以在标签旁边添加一个必须填写的标志。

第一行将包含一个文本输入框，用于输入用户的名。可以将其添加在<legend>元素的后面：

```
<table>
  <tr>
    <td class="label">
      <label for="fname">First name: <span class="required">*</span></label>
    </td>
    <td class="form">
      <input type="text" name="txtFirstName" id="fname" size="12" />
    </td>
  </tr>
</table>
```

(4) 添加用于输入姓和 e-mail 地址的文本输入框；它们均位于<tr>元素中，这些元素紧跟在上一步中添加的行后面。可以使 e-mail 地址文本输入框的尺寸略大于其他文本输入框，因为其中的文本可能较长：

```
<tr>
  <td class="label">
    <label for="lname">Last name:
    <span class="required">*</span></label>
  </td>
  <td class="form">
    <input type="text" name="txtLastName" id="lname" size="12" />
  </td>
</tr>
<tr><td> </td><td> </td></tr>
```

```
<tr>
  <td class="label">
    <label for="email">E-mail address:
    <span class="required">*</span></label>
  </td>
  <td class="form">
    <input type="password" name="txtEmail" id="email" size="20" />
  </td>
</tr>
<tr><td> </td><td> </td></tr>
```

e-mail 地址行的上方和下方添加了一个空表行，以稍微间隔表单。如果表单控件的数量很多，则比起对行进行拆分，表单看上去可能会更长、更复杂，但是在这个示例中，这些行不需要拆分为不同的部分。

(5) 用户需要填写的最后两个控件是为站点提供密码并确认该密码的控件。两个<input>元素都具有 type 属性并且属性值都为 password。为了解释密码的长度必须在 6 到 12 字符之间，在第一个密码框右边的列中添加一条消息。将注释添加在密码输入框右边的原因是，如果将其放置在左边，则所有标签将无法对齐。

```
<tr>
  <td class="label">
     <label for="pwd">Password: <span class="required">*</span></label>
  </td>
  <td class="form">
    <input type="password" name="txtPassword" id="pwd" size="12" />
    <span class="small"> must be between 6 and 12 characters long</span>
  </td>
</tr>
<tr>
  <td class="label">
    <label for="pwdConf">Confirm password:
    <span class="required">*</span></label>
  </td>
  <td class="form">
    <input type="password" name="txtPasswordConf" id="pwdConf" size="12" />
  </td>
</tr>
```

(6) 在表单的末尾处添加一个 Submit 按钮。将该按钮放置在一个 元素中，以便将其定位在表单的右边。按钮的后面紧跟着对星号作用的解释。

```
<div class="submit"><input type="submit" value="Register" /></div>
<span class="required">*</span> = required
```

(7) 将这个表单存储为 registration.html，当在浏览器中打开这个页面时，所看到的页面将如前面的图 10-25 所示。

下面是这个示例使用的 CSS 样式表(registration.css)：

```
body{color:#000000; background-color:#ffffff;
  font-family:arial, verdana, sans-serif; font-size:12pt;}
fieldset {font-size:12px; font-weight:bold; padding:10px;
  width:500px;}
td {font-size:12px;}
td.label {text-align:right;
  width:175px;}
td.form {width:350px;}
div.submit {width:450px; text-align:right; padding-top:15px;}
span.small {font-size:10px;}
span.required {font-weight:bold; font-size:20px; color:#ff0000;}
input {border-style:solid; border-color:#000000; border-width:1px;
  background-color:#f2f2f2;}
```

工作原理

这个示例相当直观，下面是需要注意的一些方面：

- 使用一个两列的表和 CSS 对齐表单标签和表单控件。相比于表单元素不对齐的表，这会使表单更简洁并更易于阅读。
- 使用<label>元素标记每个表单控件。
- 根据表线性化内容的方式，屏幕阅读器应该能够方便地向用户朗读其中的信息。
- 表单包含在一个<fieldset>元素中以显示表单的界限和尺寸，并且添加一个<legend>元素以描述表单的作用。
- 必须填写的信息由红色星号指示，这里使用颜色和符号来指示额外的含义(请记住，不应该单独依赖于颜色来传递信息)。
- e-mail 地址输入框的上方和下方都添加了空白，以便使布局更吸引人，并且会使用户不那么畏惧填写表单。
- 利用 元素将 Submit 按钮放置在表单的右边。Submit 按钮位于表单右边遵循了眼睛浏览页面的方式，并且表明向前移动而不是向后移动——这符合用户关于浏览器菜单中 Forward 按钮位于 Back 按钮右边的体验。
- 文本输入框和 Submit 按钮都被赋予 CSS 样式，以向它们提供外边框和背景，这是比纯 XHTML 表单控件更加样式化的表示方式。

10.5 本章小结

本章介绍了关于 Web 页面布局的更多知识。第 9 章中介绍了页面的普通布局或结构，在本章中则介绍了关于页面特定部分的问题：文本、导航、表和表单——即填充页面结构的各种元素。从在页面中的元素(例如文本或图像)之间添加空白，到对齐表内表单的各个部分，本章介绍了很多有用的提示，它们有助于您设计出更优秀的页面。

但是，必须记住的是，不存在一些固定的规则能够使您成为优秀的设计人员；您是否具有 Web 设计的眼光取决于您的艺术细胞和创新性。该工作可以被认为是大量职业的组合。它有点类似于成为一个艺术家，因为需要技能、耐心和实践。它也有点类似于穿衣考

究，因为穿衣方面的得体很难解释。或者，它类似于成为一个室内设计人员，因为必须知道如何搭配各种设计元素。它也类似于建筑师，因为需要确保每个人都能够访问站点。它同样类似于品牌顾问，因为需要选择合适的颜色和字体，以实现站点的目标和价值。

本章介绍的知识是关于如何使描绘在纸面上的设计成为实际的页面布局，以及一些如何实现这些设计的提示和技巧。

必须遵循的唯一真正的限制是，当设计 Web 页面时，需要让目标观众觉得很受吸引并易于使用。请记住，如果希望站点吸引很多访问者，不要只为自己设计它，也不要只为客户设计它，而是为了站点的预期访问者设计它。

10.6　练习

所有练习的答案都在附录 A 中给出。

1. 在这个练习中，需要对本章末尾的“试一试”部分中的表单(registration.html)添加第二个页面。表 10-3 给出了需要在该表单中添加的新项。

另外也需要添加以下内容：

- 在页面顶部添加一条指示，告诉用户已经完成了表单的多少内容。
- 在页面底部添加一个 Back 按钮和一个 Proceed 按钮(而不是 Submit 按钮)。

表 10-3

信　息	表 单 控 件	是否必须填写
地址 1	文本输入框	是
地址 2	文本输入框	否
城镇/郊区	文本输入框	否
城市/州	文本输入框	是
邮政编码	文本输入框	是

完成上述操作之后，页面应当如图 10-27 所示(registration2.html)。

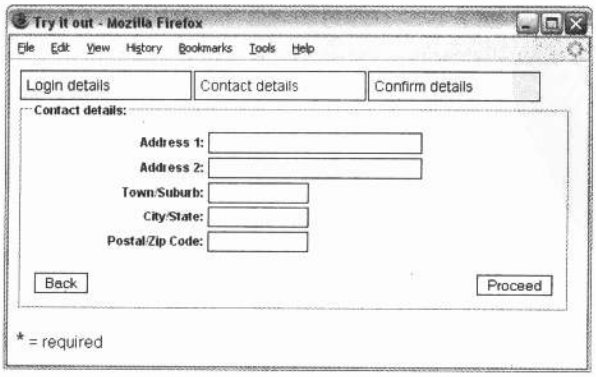


图 10-27

第11章

学习 JavaScript

在前面的章节中主要介绍了一种标记语言(XHTML)和一种样式表语言(CSS)。虽然可以将对这些语言的学习作为“编程的入门”，但是大多数资深程序员认为标记文档和真正的“编程”之间存在区别，他们认为在真正的编程中，主要是对数据执行计算，并且基于程序接收到的输入以编程方式制定决策。在本章中，您将真正地开始学习编程；本章介绍 JavaScript 编程语言的一些基本知识。JavaScript 是一种轻量级的编程语言，通常称为脚本语言，但是在学习该语言的过程中，您将了解很多编程的基础概念。

不可能在一到两章中介绍关于 JavaScript 的所有方面，但是本章和下一章中介绍的内容将足够帮助您理解从 Web 上免费获得的数以千计的脚本，并且可以将它们集成到您的 Web 页面中。甚至可以自定义这些脚本，并基于在这两章中所介绍的内容编写自己的脚本。另外，您还将了解什么是真正的编程。

因此，本章将介绍 JavaScript 的基本知识；然后在第 12 章中将介绍很多示例，这些示例可以作为有用的脚本库，可以在自己的页面中使用它们，并且能够实践在本章中学习到的基本概念。

JavaScript 赋予 Web 开发人员一种用于 Web 页面中的编程语言，并且允许他们执行以下任务：

- 读取文档中的元素，并且将新元素和文本写入文档
- 操作或移动文本
- 创建弹出式窗口
- 对数据执行数学计算
- 响应事件，例如用户将鼠标悬停在图像上或单击按钮
- 获取用户计算机上的当前日期和时间或文档上一次修改的时间
- 确定用户屏幕大小、浏览器版本或屏幕分辨率
- 基于一些条件执行操作，例如如果用户在表单中输入了错误的信息或者按下特定的按钮，则提醒用户。

JavaScript 最早被 Netscape 2.0 浏览器引入，但是当时称为 LiveScript。该语言的具体思想是在 Web 上的文档中添加交互功能，此前这些文档都是静态的。在引入 JavaScript 之前，用户仅能在浏览器中输入 URL 或单击链接、阅读页面和查看图像。JavaScript 允许 Web 页面制作人员访问并操作文档的功能和内容以及用于查看文档的浏览器。

说明：
JavaScript 不等于 Java，Java 是一种较为庞大的编程语言(尽管它们存在一些相似性)。

可能需要多次仔细阅读本章以很好地掌握使用 JavaScript 能够执行的操作；阅读本章中的示例之后，您应当能够很好地了解该语言的强大功能。有很多内容需要学习，但是这两章能够帮助您做好准备。

11.1　编程的定义

在本章中您将看到，编程在很大程度上是对数据执行计算。能够执行的任务示例包括：

- 针对数值的数学计算，例如加法、减法、乘法和除法。
- 检查一个值是否匹配另一个值(用户是否输入某些特定的文本或数值)。
- 查找文本的某个子部分，例如一个单词的第三或第四个字母，或者一条语句的第一个或最后一个单词。
- 检查一段文本的长度，或者找出一段文本内字母 t 第一次出现的位置。
- 检查两个值是否不同，或者一个值是否长于或短于另一个值。
- 基于是否满足一个条件(或多个条件中的一个)执行不同的操作。例如，如果用户输入了一个小于 10 的数值，程序将执行一种操作；否则它将执行另一种操作。
- 重复执行某个操作固定次数，或者直到满足条件才停止执行(例如用户按下一个按钮)。

这些操作看上去可能非常简单，但是它们可以结合起来，从而变得非常复杂和功能强大。不同的操作集可以在不同的情形下执行不同的次数。

为了能够表示任何内容，编程语言首先需要一种工作环境。您将看到 JavaScript 用于浏览器中的 Web 文档(从技术上来说，JavaScript 也可以驻留在其他应用程序或 Web 服务器中，但是此处主要关注它在浏览器中的用法)。因此用于执行计算的值将主要来源于加载到浏览器中的 Web 文档。例如，可能需要检查用户是否在某个表单字段中输入密码。如果用户没有输入任何内容，则可以要求他或她首先提供一个密码，然后再将表单提交给服务器；否则可以正常地提交表单。

由于不同的编程语言通常需要处理相同的应用程序或相同类型的文档，因此存在应用编程接口(Application Programming Interface，API)，它类似于关于使用编程语言能够要求什么内容以及如何格式化响应的手册。

Web 文档的 API 称为 DOM(文档对象模型)。例如，DOM 定义了可以获取或设置的 Web 文档特性以及可以执行的方法。特性是关于文档或它的内容的信息——例如，任何图像的 height 属性的值、任何链接的 href 属性值或者在表单文本框中输入的密码长度。同时，方法允许执行一些操作，例如对表单执行的 reset()方法或 submit()方法，它们允许复位或提交表单。

理解特性和方法之间的区别非常重要。特性告诉您一些信息(例如，汽车的特性可以是它的颜色或引擎大小)，而方法实际地执行某些操作(因此方法可以用于加速或改变某个齿轮)。

现在查看关于 DOM 的示例：如果希望创建一幅图像，并且当用户将鼠标悬停在其上方时改变该图像(称为翻转图像)，则需要创建两幅图像——第一幅用于在普通默认状态时

显示，第二幅用于当用户将鼠标悬停在图像上时显示。因此，DOM 可以用于访问该图像的 src 特性，当用户将鼠标悬停在图像上时使用脚本加载另外一幅图像。当用户的鼠标指针不在图像上时，改回初始图像。为了响应用户的鼠标在图像上的移动，也需要了解事件。

为了让人们能够与程序交互，编程语言需要能够响应事件，例如用户在某个元素上移动鼠标，单击鼠标按键，按下键盘上的一个键，或者单击表单上的提交按钮。当任何一种操作发生时，将激活一个事件。当事件激活时，它能够用于触发一个脚本的特定部分(例如前面讨论的图像翻转脚本)。

由事件触发的脚本的另外一个常见示例是在用户提交表单时触发的脚本。通常，浏览器中的脚本将检查用户是否在表单中输入合适的数据；如果值不满足脚本中设置的条件，将产生一条错误消息，提醒用户必须输入正确的数据。

因此，DOM 指定如何使用 JavaScript(或其他编程语言)访问关于文档的信息。然后可以利用 JavaScript 基于从文档中获取的值执行计算或决策。脚本甚至可以调用方法和改变文档的某些特性。也可以使用事件触发特定的脚本。现在您已经了解脚本能够执行的操作，接下来需要学习如何在 Web 页面中添加脚本。

11.2 在页面中添加脚本的方式

类似于 CSS 规则，JavaScript 既可以嵌入到页面中，也可以放置在一个外部脚本文件中。但是为了能够在浏览器中工作，浏览器必须支持 JavaScript，并且必须启用该功能(很多浏览器允许用户禁止 JavaScript 功能)。需要牢记的是，用户可能没有启用他们浏览器中的 JavaScript 功能，因此应当仅使用 JavaScript 增强使用页面的体验；不能将其作为使用或查看页面的必备条件。

脚本添加在页面的<script>元素内。起始标签<script>中的 type 属性指示该元素内的脚本语言类型。存在多种其他类型的脚本语言(例如 VBScript 或 Perl)，但是 JavaScript 是目前浏览器中使用的最流行的脚本语言。下面是一个非常简单的脚本，它将在页面中写出"My first JavaScript"语句(ch11_eg01.html)：

```
<html>
<body>
  <p>
    <script type="text/javascript">
      document.write("My first JavaScript")
    </script>
  </p>
</body>
</html>
```

JavaScript 使用 write()方法将文本写入文档(请记住，方法用于执行操作或计算)。在页面中脚本所在的位置输出该文本。图 11-1 给出了这个简单页面的显示效果。

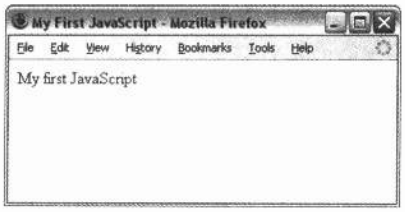


图 11-1

JavaScript 代码在页面中的放置位置非常重要。如果将其放置在页面主体中——如同这个示例所示——则它将在页面加载时运行(或执行)。但是，有时希望脚本仅在某个事件触发时运行；事件可以是类似于按下键盘键或单击提交按钮等。这将通常导致调用函数。函数放置在位于页面的<head>中的<script>元素内，以确保它们在页面显示之前加载，因而在页面加载时能够立即使用。使用函数可以在页面不同部分中重用相同的脚本。

也可以在外部文档中编写 JavaScript，该外部文档的扩展名为.js。如果脚本将被多个页面使用，则这是一种非常好的方法——因为不需要在每一个使用该脚本的页面中重复它，并且如果希望更新脚本，则仅需要在一个位置更改它。当将 JavaScript 放置在一个外部文件中时，需要使用<script>元素的 src 属性；src 属性的值应当是指向包含 JavaScript 的文件的绝对或相对 URL。例如：

```
<script type="JavaScript" src="scripts/validation.js" />
```

因此存在 3 个位置可以用于放置 JavaScript 代码——并且一个 XHTML 文档可以同时使用这 3 种方式，因为一个文档能够包含的脚本数量是没有限制的。这 3 个位置如下所示：

- 位于页面的<head>部分中：这些脚本在事件触发它们时被调用。
- 位于页面的<body>部分中：这些脚本将在页面加载时运行。
- 位于一个外部文件中：如果链接放置在<head>元素中，则脚本的处理方式与将其放置在文档的头部并等待事件触发相同，而如果链接放置在<body>元素中，则它的行为类似于脚本放置在文档主体中并在页面加载时执行。

某些早期的浏览器不支持 JavaScript；因此，有时将看到在一个 HTML 或 XHTML 注释中编写 JavaScript，以便较老的浏览器可以忽略该脚本，否则它可能会造成错误，如下所示。较新的浏览器将仅会忽略<script>元素中的这些注释：

```
<script type="text/javascript">
  <!--
    document.write("My first JavaScript")
  //-->
</script>
```

注意这个 XHTML 文档最后几个字符之前的两个正斜杠(//)。这实际上是一个 JavaScript 注释，用于阻止 JavaScript 编译器处理“-->”字符。

为了创建格式美观的 XHTML 文档，在文档中包含脚本时必须非常仔细，因为 JavaScript 包含一些不能用于 Strict XHTML 中的 CDATA 部分外部的字符(例如尖括号<和>)。CDATA 部分向用于处理文档的任意程序表明，该部分的代码不包含标记(因此不应该

像标记一样处理)。这就可以有效地使用不能以其他方式出现在文档中的字符。

遗憾的是,在 CDATA 部分内的包含脚本——类似于 XHTML——会对不支持 XML 的早期浏览器造成问题。但是,可以将 JavaScript 注释与 CDATA 部分组合来实现向后兼容性,如下所示:

```
<script type="text/javascript">
//<![CDATA[
...
]]>
</script>
```

如果担心浏览器不能支持脚本,备选方法是使用外部脚本,因为如果浏览器不能处理<script>元素,它甚至不会尝试加载包含脚本的文档。

11.2.1 JavaScript 中的注释

可以在 JavaScript 代码中以两种方式添加注释。第一种方式前面已经使用过,它允许注释该行中注释符之后的任意内容。在下面的代码中,与两个正斜杠在同一行并位于两个正斜杠之后的语句都被处理为注释:

```
<script type="text/javascript">
document.write("My first JavaScript") // comment goes here
</script>
```

也可以使用如下语法注释多行,将注释包含在起始字符对“/*”和结束字符对“*/”之间:

```
/* This whole section is commented
out so it is not treated as a part of
the script. */
```

这与 CSS 中的注释相似。

注意:

与所有代码一样,清晰地注释代码是优秀的实践,即使您是可能使用该代码的唯一人员,因为对于在编写脚本时看上去很清晰的代码,在后面重新阅读它时也可能难以理解。添加变量名的描述以及函数和其参数的解释,这些都是注释代码以使其更易于阅读的优秀示例。

11.2.2 <noscript>元素

当用户浏览器不支持 JavaScript 或已经禁用它时,<noscript>元素为用户提供备选的内容。它可以包含文档创建人员希望用户看到的任意 XHTML 内容,这些内容在用户没有启用浏览器的 JavaScript 功能时显示。

注意：

严格来说，W3C 规范仅要求该元素的内容在浏览器不支持所需要的脚本语言时才显示；但是，浏览器制造商却规定该元素在脚本语言被关闭时也起作用。仅有的例外是 Netscape 2 浏览器，它即使在支持脚本时也显示该内容——但是安装这种浏览器的用户非常少，因此不需要担心它。

试一试　创建外部 JavaScript

本章前面已经给出一个向页面中写入文本的简单 JavaScript 示例。在这个示例中，将把代码移动到一个外部文件中，这个外部文件将用于向页面中写入一些文本。

(1) 打开编辑器并输入如下代码：

```
document.write("Here is some text from an external file.");
```

(2) 将这个文件保存为 external.js。

(3) 在编辑器中创建一个新页面，并且添加如下代码。注意此时<script>元素为空，但是附带一个 src 属性，并且该属性的值是一个 JavaScript 文件：

```
<html>
<body>
  <script src="external.js" type="text/JavaScript">
  </script>
  <noscript>This only shows if the browser has JavaScript turned off.
  </noscript>
</body>
</html>
```

(4) 将这个示例保存为 ch11_eg02.html 并在浏览器中打开它，结果将如图 11-2 所示。

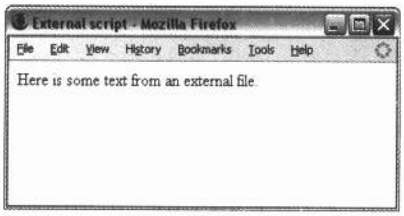


图 11-2

工作原理

当 XHTML 文档中的<script>元素附带 src 属性时，该元素用于加载一个外部 JavaScript 文件。这个 src 属性指示 JavaScript 文件的源——即在何处能够找到它。该属性的值可以是一个相对或完整 URL。

可以使用这种方法在文档的<head>或<body>部分中包含外部 JavaScript。如果将它们放置在文档主体中，则它们将在页面加载时执行——如同脚本实际地位于页面中的该位置一样——如这个示例所示。如果将它们放置在头部，则它们将被一个事件触发。

对于大多数功能，趋向于使用外部 JavaScript 文件，并且将<script>元素放置在文档的头部。这就能够在开发的不同站点中重用这些脚本，并且确保 XHTML 文档关注于内容而不是杂乱的脚本。

11.3 文档对象模型

本章开始部分中提到，JavaScript 本身仅执行一些计算或者处理一些基本的字符串。为了使文档的交互性更强，脚本需要能够访问文档的内容，并且需要知道用户什么时候与其交互。脚本通过与浏览器交互完成该功能，当交互时使用在称为文档对象模型的应用编程接口中规划的特性和方法。

在本节中将着重介绍文档对象模型中的 3 种对象：文档对象、表单集合(和它的子对象)以及图像对象。在本章的末尾部分还将介绍其他对象。

11.3.1 文档对象模型简介

文档对象模型解释脚本能够获取什么样的文档特性以及能够修改哪些特性；它也定义了一些方法，可以调用这些方法以对文档执行操作。您将在本章中看到，文档的特性通常对应于文档中 XHTML 元素附带的属性，而方法执行一些任务。

例如，文档对象模型指定如何获取用户输入在表单中的值。获取这些值之后，可以使用 JavaScript 确保用户为该表单控件输入了合适的值。JavaScript 是执行计算的编程语言——在此处用于检查用户输入的值是否合理——而文档对象模型(DOM)解释如何访问文档。

图 11-3 演示了第 0 级的 HTML 文档对象模型(本章后面将会介绍，存在不同级别的 DOM)。应当注意到，该结构类似于家族树。

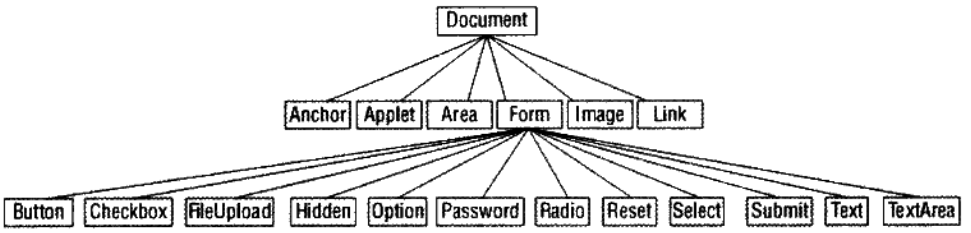


图 11-3

注意：

DOM 不是 JavaScript 的一部分；它只是解释所有的编程语言应当如何访问文档的特性。其他语言能够以相同的方式访问文档的特性。浏览器(或其他应用程序)实现 DOM 的方式也是开放的——例如，Firefox 浏览器和 Internet Explorer 浏览器将使用不同的代码获取或设置特性以及执行方法，但是它们的效果应该是相同的。

图 11-3 演示了页面中的元素如何作为可脚本化的对象用于脚本中。文档对象代表整个

文档，然后每个子对象代表该文档中类似标记的集合：

- forms 集合包含文档中的所有<form>标记。
- image 集合代表文档中的所有图像。
- link 集合代表页面内的所有超链接。
- anchor 集合代表文档中的所有锚点(具有 name 属性或 id 属性(而非 href 属性)的<a>元素。
- area 集合代表文档中所有使用<area>元素的图像映射。
- applet 集合代表文档内所有的小应用程序。

forms 集合还具有一些子对象，它们分别代表显示在表单上的不同类型的表单控件：按钮、复选框、文件上传框、隐藏输入框、选项、密码输入框、单选按钮、复位按钮、选择框、提交按钮、文本框和文本区域。

为了更好地理解如何使用 DOM 访问文档，查看下面的简单文档，它包含一个表单和两个链接：

```
<h1>User Registration</h1>
<form name="frmLogin" action="login.aspx" method="post">
  Username <input type="text" name="txtUsername" size="12" /> <br />
  Password <input type="password" name="pwdPassword" size="12" /> <br />
  <input type="submit" value="Log In" />
</form>
<p>If you are a new user <a href="register.aspx">Register here</a> |
  If you have lost your password you can <a href="lostPassword.aspx">retrieve
  your password here</a>.</p>
```

DOM 将表单的内容作为 forms 集合的一部分在脚本中使用，将链接作为 link 集合的一部分在脚本中使用。

注意：

表单的操作属性中引用的页面都是假设的页面；下载代码中没有这些页面。本章后面给出的具有 JavaScript 代码的 XHTML 页面独立于具有.aspx 扩展名的页面，其中后者位于服务器中。

存在两种方式可用于访问这个文档中的值——它们都涉及指示文档中您所感兴趣的指定部分，方式是使用点标记。这涉及在对象模型中的每个对象之间使用一个句点字符。解释该方法的最佳方式是使用一个示例。

为了访问文档中的第一个链接，可以使用如下方式：

```
document.links[0].href
```

这条语句具有 4 个部分，其中 3 个部分以句点隔开以到达第一个链接：

- 单词 document 指示正在访问的是文档对象。
- 单词 links 对应于 link 集合(毕竟这个示例获取的是文档中第一个链接的值)。

- [0]指示希望获取的是文档中的第一个链接。集合中项的编号从 0 开始，而不是从 1 开始，这意味着 link 集合中的第二个链接使用[1]表示，第三个链接使用[2]表示，依此类推。
- 此处已经指出希望获取这个链接的 href 特性。

每个对象具有对应于该元素类型的不同特性；例如，链接具有一些特性，其中 href 特性用于访问这个<a>元素中 href 属性的值。类似地，<textarea>对象具有 cols、disabled、readOnly 和 rows 特性，它们分别对应于该元素的属性。

另外一种可选方法是使用元素名导航文档。例如，下面的链接请求密码输入框的值：

```
document.frmLogin.pwdPassword.value
```

这条语句也有 4 个部分：

- 最前面的 document 再次表明这是顶层对象。
- 表单的名称是 frmLogin。
- 后面紧跟表单控件的名称 pwdPassword。
- 最后的特性是密码输入框的值，这个特性称为 value。

这两种方法都允许导航文档，选择感兴趣的元素和这些元素的特性。然后能够获取这些值，对它们执行计算，以及提供备选的值。

也存在第二种类型的对象模型，即浏览器对象模型，它使程序员可以使用浏览器的功能，例如窗口对象可用于创建新的弹出式窗口。本章稍后将介绍窗口对象。

注意：

出于学习 JavaScript 的目的，在本章中处理 DOM Level 0，因为它能够被大多数浏览器所支持。它的语法的创建时间在 W3C 创建它的 DOM Level 1、2 和 3 规范之前(这 3 种规范更为复杂，并且它们在不同浏览器中具有不同的支持程度)。熟悉这些基础知识之后，可以根据需要了解一些更多的细节。

11.3.2 对象、方法和特性

如图 11-3 中所示，一个对象模型(例如文档对象模型)由代表文档不同部分的多个对象组成。每个对象可以具有一些特性和方法：

- 特性表明关于对象的信息。
- 方法执行相应的操作。

理解如何利用某个对象之后，利用所有类型的对象都会更为容易——并且当开始编程时，您将会遇到许多不同类型的对象。下面的小节中将介绍文档对象的一些特性和方法。

1. 文档对象的特性

在表 11-1 中给出了一些文档对象特性，其中几个特性对应于<body>元素附带的属性，<body>元素包含文档的内容。

可以设置和读取许多特性。如果能够设置一个特性，则该特性称为读/写特性(因为可以读取它，也可以写入它)，相反只能够读取的特性称为只读特性。在表 11-1 的最后一列

中可以看出哪些特性可以读取，哪些特性可以写入。

表 11-1

性 质 名	目　　的	读/写
alinkColor	指定链接的颜色(类似于<body>元素中逐渐淘汰的 alink 属性)	读/写
bgcolor	指定背景色(类似于<body>元素中逐渐淘汰的 bgcolor 属性)	读/写
fgcolor	前景/文本的颜色(类似于<body>元素中逐渐淘汰的 text 属性)	读/写
lastModified	文档最近一次修改的日期(Web 服务器通常将该信息在 HTTP 头中发送， HTTP 头无法直接查看)	只读
linkColor	指定链接的颜色(类似于<body>元素中逐渐淘汰的 link 属性)	读/写
referrer	用户来自于的 XHTML 页面的 URL(如果用户单击某个链接)。如果没有访问来源，则该特性为空	只读
title	<title>元素中页面的标题	只读(直到 IE 5 浏览器和 Netscape 6 浏览器以及后面的版本才支持该特性)
vlinkColor	<body>元素中逐渐淘汰的 vlink 属性	读/写

注意：

逐渐被淘汰的特性已经被删除，取而代之的是采用 CSS 赋予文本、链接和背景样式。

例如，可以通过如下方式访问文档的标题：

```
document.title
```

或者可以通过如下方式查找文档最近一次修改的日期：

```
document.lastModified
```

注意，如果服务器不支持 lastModified 特性，IE 浏览器将显示当前日期，而其他浏览器通常显示“1 January 1970”(这是大多数计算机根据其计算所有日期的日期)。

2. 文档对象的方法

方法执行一些操作，并且通常后面紧跟一对括号。在某些方法的括号内，有时能够看到形参或实参，它们能够影响方法执行的操作。

例如，在下面的表 11-2 中存在两个方法，它们采用字符串作为实参；这两个方法都向页面中写入字符串(字符串是字符的序列，字符可以包括字母、数值、空格和标点符号)。

表 11-2

方 法 名	目　　的
write(string)	可用于向文档中添加文本或元素
writeln(string)	功能与 write()类似，但在输出的末尾添加新行(类似于完成输入之后按下 Enter 键)。

在 ch11_eg01.html 中已经使用文档对象的 write()方法，其中显示了如何使用 write()方法将内容写入到文档中：

```
document.write('This is a document');
```

write()方法可以采用一个字符串作为参数。在这个示例中，字符串是“This is a document”。

也可以采用表达式作为 write()方法的参数。例如，下面的代码将写出文本字符串“Page last modified on”，并且后面紧跟文档的最近修改日期。

```
document.write('Page last modified on ' + document.lastModified);
```

在本章后面将更详细地介绍表达式，但是在这个示例中，表达式求值为(或导致产生)一个字符串。例如，您可能会看到类似于“Page last modified on 12th December 2007”的输出。

现在您已经了解文档对象的特性和方法，这将有助于学习其他一些对象的特性和方法。

11.3.3 forms 集合

forms 集合保存对应于页面中每个<form>元素的引用。这听起来可能有些复杂，但是可以设想一个具有多个表单的 Web 页面——一个登录表单、一个用于新用户注册的注册表单以及一个位于相同框中的搜索框。在这种情况下，需要能够区分页面中的不同表单。

因此，如果登录表单是文档中的第一个表单，并且希望访问登录表单的 action 特性(登录表单将位于 XHTML 文档的<form>元素中)，则可以使用下面的索引号选择对应的表单并访问它的特性和方法(请记住，索引号从 0 开始，0 用于第一个表单，1 用于第二个表单，2 用于第三个表单，依此类推)：

```
document.forms[0].action
```

另外，可以使用表单名直接访问表单对象：

```
document.frmLogin.action
```

选择的表单具有自己的对象，并且这些对象具有一些特性(主要对应于<form>元素的属性)和方法。介绍表单的特性和方法之后，接下来介绍对应于不同表单控件的对象、特性和方法。

1. 表单对象的特性

表 11-3 中列举了表单对象的特性。

表 11-3

性 质 名	目　　的	读/写
action	<form>元素的 action 属性	读/写
length	给出表单中表单控件的数量	只读
method	<form>元素的 method 属性	读/写
name	<form>元素的 name 属性	只读
target	<form>元素的 target 属性	读/写

2. 表单对象的方法

表 11-4 列举了表单对象的方法。

表 11-4

方 法 名	目　　的
reset()	将所有表单元素复位为它们的默认值
submit()	提交表单

注意：

在本章后面将介绍事件和事件处理程序。但是应当注意，如果使用表单对象的 submit() 方法，则该<form>元素上的任何 onsubmit 事件处理程序都将被忽略。

11.3.4　表单元素

当访问一个表单时，通常希望访问它的一个或多个元素。每个<form>元素具有一个 elements[]集合对象作为其特性，该集合对象代表表单中的所有元素，其工作方式与 forms[] 集合类似；它允许利用索引访问所需的元素(该索引是对应于元素在文档中的顺序的数值，该数值从 0 开始)。另外，也可以使用元素的名称。

下面是可能希望对表单中的元素执行的操作：

- 文本字段：读取用户输入的数据，或者将新文本写入这些元素中。
- 复选框和单选按钮：测试它们是否被选中，以及选中或取消选中它们。
- 按钮：禁用它们直到用户选择一个选项。
- 选择框：选择一个选项或者查看用户选择哪个选项。

1. 表单元素的特性

表 11-5 列举了表单元素的特性。

表 11-5

性　质	应用的元素	目　的	读/写
checked	复选框和单选按钮	当选中时返回 true，当没有选中时返回 false	读/写
disabled	除了隐藏元素之外的所有元素	当被禁用并且用户无法与其交互时返回 true(仅被 IE 4 浏览器和 Netscape 6 浏览器以及后面的版本支持)	读/写
form	所有元素	返回元素所在的表单的引用	只读
length	选择框	<select>元素中选项的数量	只读
name	所有元素	访问元素的 name 属性	只读
selectedIndex	选择框	返回当前选中的项的索引号	读/写
type	所有元素	返回表单控件的类型	只读
value	所有元素	访问元素的 value 属性或文本输入框的内容	读/写

注意：

如果希望某个表单控件直到某人执行相应的操作——例如，禁用 Submit 按钮直到用户同意一些条款和条件——则应当在页面加载时在脚本中禁用该表单控件，而不是利用 XHTML 在表单控件自身中禁用它；在第 12 章中将介绍关于这个主题的更多内容。

2. 表单元素的方法

表 11-6 列举了表单元素的方法。

表 11-6

性 质 名	应用的元素	读/写
blur()	除了隐藏元素之外的所有元素	将焦点从当前活动元素上移动到焦点移动顺序中的下一个元素
click()	除了文本之外的所有元素	模仿用户在元素上单击鼠标
focus()	除了隐藏元素之外的所有元素	将焦点赋予该元素
select()	除了隐藏元素之外的文本元素	选择该元素中的文本

试一试　收集表单数据

在这个示例中将获取一个文本框的值，并将其写入到 JavaScript 警告框。这个示例的主要目的是显示如何获取表单中的值，但是它也将介绍事件和 JavaScript 警告框。

这个简单的表单只包含一个文本输入框和一个提交按钮。当用户在文本框中输入内容并单击提交按钮之后，在文本框中输入的值将会出现在警告框中。图 11-4 给出了用户单击提交按钮之后的页面。

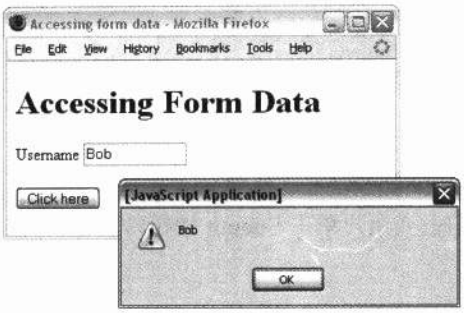


图 11-4

单击 OK 按钮之后，警告框将消失。

(1) 创建 Transitional XHTML 页面的程序框架文档，并且添加一个题头，用于解释该示例演示的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Accessing form data</title>
</head>
<body>
  <h1>Accessing Form Data</h1>
</body>
</html>
```

(2) 在文档主体中添加一个<form>元素。该表单应当包含一个用于输入用户名的文本输入框和一个提交按钮，如下所示：

```
<body>
<form name="frmLogin">
  Username <input type="text" name="txtUsername" size="12" /> <br />
  <input type="submit" value="Click here" />
</form>
</body>
```

(3) 在<form>元素中添加 onsubmit 属性，并赋予它如下的值：

```
   <form name="frmLogin"
onsubmit="alert(document.frmLogin.txtUsername.value)">
     Username <input type="text" name="txtUsername" size="12" /> <br />
     <input type="submit" value="Click here" />
   </form>
```

将文件保存为 ch11_eg3.html 并在浏览器中打开它。当在文本输入框中输入内容并单击 Submit 按钮时，将会看到如图 11-4 所示的警告框，该警告框中显示了输入在文本框中的值。

工作原理

创建警告框并确保它显示用户输入在文本框中的内容的代码行是 onsubmit 事件处理程序属性的值：

```
<form name="frmLogin" onsubmit="alert(document.frmLogin.txtUsername.value)">
```

当 onsubmit 事件激活时(当用户单击 Submit 按钮时发生)，运行这个简单的脚本行。此时调用 alert()方法：

```
alert(document.frmLogin.txtUsername.value)
```

alert(string)方法用于将字符串写入到文本框中。与前面介绍的文档对象的 write()方法一样，该字符串不需要是希望显示的实际文本。在这个示例中，并不是每次脚本运行时都将相同的字符串写入到警告框中，而是将用户输入到文本框中的文本写入警告框。

可以看到，在 alert()内已经选择文本输入框和它的 value 特性。因此，将文本输入框的值写入警告框中(可能也会注意到，在这个示例中没有使用双引号，在第一个示例中，当将字符串写入页面时必须使用双引号)。

当用户单击 Submit 按钮时，onsubmit 事件激活，它创建包含文本输入框的值的警告框。

11.3.5 images 集合

images 集合提供图像对象的引用，图像对象代表文档中的每一幅图像。同样可以通过使用名称或集合中的索引号来引用这些图像对象。因此，可以采用如下方式查找第一幅图像的 src 属性：

```
document.images[0].src
```

或者可以使用名称直接访问对应于一幅图像的图像对象。例如，如果图像具有 name 属性，该属性的值为 imgHome，可以使用如下方式访问它：

```
document.imgHome.src
```

可能希望更改的主要特性是 src 特性，特别是在创建翻转图像时。

图像对象没有方法，但是它具有多个特性，这些特性仅被 Netscape 3 以上版本的浏览器和 IE4 以上版本的浏览器支持。

1. 图像对象的特性

表 11-7 列举了图像对象的特性。

表 11-7

性　质	目　的	读/写
border	<img>元素的 border 属性	读/写
complete	指明图像是否已经成功加载	只读

(续表)

性　质	目　的	读/写
height	<img>元素的 height 属性	读/写
hspace	<img>元素的 hspace 属性	读/写
lowsrc	<img>元素的 lowsrc 属性(指示图像的一较低分辨率的版本)	读/写
name	<img>元素的 name 属性	读/写
src	<img>元素的 src 属性	读/写
vspace	<img>元素的 vspace 属性	读/写
width	<img>元素的 width 属性	读/写

试一试　简单的图像翻转

在这个示例中将介绍当用户将鼠标悬停在图像上时，如何使用一幅图像替换另一幅图像。这种类型的图像通常用于导航项中，以表明用户能够单击它们。

虽然为了能够实现翻转，需要加载两幅图像而不是一幅图像，但是它们非常有效，并且如果仔细地选择图像(确保图像文件不会太大)，则为每次翻转加载另外一幅图像的额外系统开销不是什么问题。

在这个示例中，您将看到两幅简单的图像，它们均具有单词“click here”。当页面加载时，图像将是绿色底白色字，但是当用户将鼠标悬停在图像上时，它将变为红色底白色字。

(1) 创建 Transitional XHTML 文档的程序框架：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Image Rollover</title>
</head>
<body>
</body>
</html>
```

(2) 在文档主体中添加下面的链接和图像：

```
<p>Hover over the image with your mouse to see the simple rollover effect.
<br/>
<a href=""
  <img src="images/click_green.gif" width="100" height="50" border="0"
       alt="Example button" name="button" />
</a>
</p>
```

(3) 现在在<a>元素中添加下面的 onmouseover 和 onmouseout 事件处理程序属性，这些属性具有指定的值：

```
<a href=""
   onmouseover="document.images.button.src='images/click_red.gif';"
   onmouseout="document.images.button.src='images/click_green.gif'">
```

(4) 将这个示例保存为 ch11_eg4.html 并在浏览器中打开它，然后将鼠标悬停在图像上(但是不单击它)。您将看到如图 11-5 所示的效果，其中鼠标位于图像上方。

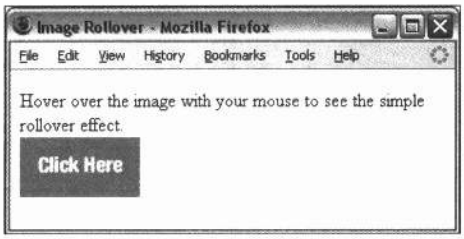


图 11-5

工作原理

当用户将鼠标悬停在图像上时，onmouseover 事件激活；当用户将鼠标从图像上移开时，onmouseout 事件激活。这是存在独立的属性对应于每一个事件的原因，当其中一个事件激活时，对应属性的值中包含的脚本被执行。

onmouseover 和 onmouseout 事件处理程序属性中的脚本告诉浏览器改变图像的 src 属性，从而向用户显示一幅不同的图像。

第一个事件(onmouseover)指明当鼠标放置在图像上时发生的情况；第二个事件(onmouseout)指明当鼠标从图像中移开时应当执行的操作。

在 ch11_eg04.html 的代码中可以看出，当用户将鼠标放置在图像上时，链接内图像的 src 特性——使用符号 document.images.link 指定——将改变。

```
<a href=""
  onmouseover="document.images.button.src='click_red.gif';"
  onmouseout="document.images.button.src='click_green.gif'">
  <img src="click_green.gif" width="100" height="50" border="0"
name="button">
</a>
```

<img />元素必须具有一个 name 属性，以便能够以名称在链接中引用图像(否则将不得不使用图像在 images 集合中的索引号)。在这种情形下最好使用名称而不是 images 集合中的索引号，因为如果后面在文档中的这幅图像之前添加另外一幅图像，则整个脚本将需要改变。

注意，如果没有事件指明当鼠标离开图像时应当发生的情况，则图像将保持红色而不是变回绿色。图像翻转脚本是关于改变或设置特性而不是仅读取特性的优秀示例。

在第 12 章中将介绍更复杂版本的图像翻转示例，该示例显示如何创建一个函数来改变同一个文档内的多幅图像；如果在导航栏中使用翻转，这种方式就特别有用。

11.3.6　不同类型的对象

在 JavaScript 中存在多种类型的对象，每一种对象负责一组相关的功能。例如，文档对象具有一些与文档相关的方法和特性；forms 集合是文档对象的一部分，它处理与表单相关的信息等。本章后面将会介绍，存在多种不同的对象，每一种对象分别处理一组不同的功能和特性。

下面是一些可能遇到的对象类型：

- W3C DOM 对象：这些对象类似于本章已经介绍的一些对象，但是在最新的浏览器中存在更多的对象，可以使用它们更全面地控制文档。W3C 发布的不同级别的 DOM 中也存在一些额外的对象。
- 内置对象：有些对象是 JavaScript 语言本身的一部分，主要包括日期对象(用于处理日期和时间)、数学对象(提供一些数学函数)。在本章的后面将更详细地介绍这些内置对象。
- 自定义对象：如果开始编写高级的 JavaScript 代码，则甚至可以开始创建自己的 JavaScript 对象，这些对象包含一些相关的功能；例如，可以编写一个验证对象，用于验证自己的表单。

虽然本章不可能介绍自定义对象的创建方式，但在本章的后面将介绍一些内置对象。

11.4　开始利用 JavaScript 编程

学习 DOM 之后，您已经了解它如何用于访问 Web 浏览器中的文档。但是，真正引入编程概念的是 JavaScript。DOM 允许获取或设置特性，并且该方法可用于引发一些操作，例如向页面中写入新内容。现在可以开始了解如何在脚本中使用这些值和特性创建更强大的文档。

本章前面已经提到，编程语言主要是执行一些计算。下面是一些需要掌握的关键概念，以便执行不同类型的计算：

- 用于存储信息的变量；变量类似于少量内存，在其中可以存储数值、字符串(由一系列的字符组成)或对象的引用。然后可以在代码中执行计算以修改保存在变量中的数据。
- 允许对变量或引用执行操作的运算符。存在一些不同类型的运算符，例如：
  - 算术运算符，可用于对数值执行加法(+)、减法(−)。
  - 比较运算符，可用于比较两个字符串以查看它们是否相同或不同(例如，x 是否等于 y)。
- 由包含一些规则(规则用于执行操作)的一些相关代码组成的函数。例如，可以编写一个函数计算需要偿还的贷款数量，在计算时向其传递一些变量，分别指明借款的数量、借贷的年数以及借贷的利率(函数与方法非常相似，但是在 JavaScript 中，方法属于对象，而函数由程序员编写)。

  条件语句允许使用变量和运算符指定条件。例如，条件可以是变量 varTimeNow(它

包含了当前时间)的值是否大于 12。如果该条件满足并且当前的时间值大于 12，则基于这个条件执行相应的操作——例如文档给出“Good afternoon”。否则，如果时间在中午之前，则文档给出“Good morning”。

- 循环，可以通过设置它运行代码块指定的次数，或者运行代码块直到某个条件满足。例如，可以使用循环让文档写出您的姓名 100 次。
- 另外存在一些内置的 JavaScript 对象，它们具有一些特定用途的方法。例如，与 DOM 的文档对象具有允许向文档中写入文本的方法类似，使用内置的 JavaScript 日期对象可以获得日期、时间或一周中的星期几。

下面的小节中将更详细地解释这些关键概念。

11.5 变量

变量用于存储数据。为了在变量中存储信息，可以赋予变量一个名称，并在该变量名称和希望使其具有的值之间放置一个等号。例如，下面是包含用户名的变量：

```
userName = "Bob Stewart"
```

这个变量称为 userName，它的值是"Bob Stewart"。如果没有赋值，则该变量的值是 undefined(注意，如果在代码中写出变量的值，则该值必须放置在引号中)。

当第一次使用一个变量时，就会创建该变量。创建变量的过程称为声明变量。可以利用 var 语句声明变量，如下所示：

```
var userName = "Bob Stewart"
```

注意：

仅当在一个函数内创建变量并且该变量的名称与一个全局变量相同时，才需要使用 var 关键字——为了理解这一点，需要理解函数、全局变量和局部变量，后面将分别介绍它们。

变量的值可以被脚本检索或更改，此时可以使用变量的名称。

关于 JavaScript 中的变量必须记住以下规则：

- 变量名区分大小写。
- 变量名必须以字母或下划线字符作为起始字符。
- 在同一个文档中避免赋予两个变量相同的名称，因为其中一个变量可能会重写另外一个变量的值，从而产生错误。
- 尽量对变量使用具有描述性的名称，这将使代码更易于理解(并且如果存在问题，这将有助于调试代码)。

11.5.1 为变量赋值

当希望为变量赋值时，首先放置变量名，然后放置一个等号，最后放置希望赋给变量的值。本章前面已经给出在声明变量时对其进行赋值的示例。在下面的示例中，对一个变量赋值，然后更改该值：

```
var userName = "Bob Stewart"
userName = "Robert Stewart"
```

userName 变量的值现在等于"Robert Stewart"。

11.5.2　变量的生命周期

当在一个函数中声明变量时，仅能够在该函数内访问该变量(本书后面将介绍函数方面的知识)。函数运行结束后，则无法再次调用该变量。函数中的变量称为局部变量。

因为局部变量仅在函数内起作用，所以不同的函数可以包含具有相同名称的变量(每一个变量仅被包含其的函数识别)。

如果在函数外部声明变量，则页面中的所有函数都能访问它。这些变量的生命周期从声明它们时开始，直到页面关闭时结束。

局部变量占用的内存和资源比页面级变量少，因为它们仅在函数运行期间占用内存，而不是在整个页面的生命周期中都必须创建并记住它们。

11.6　运算符

运算符本身是一个关键字或符号，当用于一个表达式中时，它将对值执行一些操作。例如，算术运算符“＋”将两个值加在一起。

该符号用于具有一个或两个值的表达式中，并且对这些值执行计算以生成结果。例如，下面是一个使用“x”运算符的表达式：

```
area = (width x height)
```

表达式类似于一个数学表达式。值称为操作数。仅需要一个操作数(或值)的运算符有时称为一元运算符，而需要两个值的运算符有时称为二元运算符。

在本节中将介绍的不同类型的运算符包括：

- 算术运算符
- 赋值运算符
- 比较运算符
- 逻辑运算符
- 字符串运算符

在本章后面和下一章中将给出很多实际使用运算符的示例。但是，首先需要学习每一种类型的运算符。

11.6.1　算术运算符

算术运算符对操作数执行算术操作(注意到在表 11-8 中的示例中，x = 10)。

表 11-8

符　号	描　述	示例 (x = 10)	结　果
+	加法	x+5	15
−	减法	x−2	8
*	乘法	x*3	30
/	除法	x/2	15
%	取模操作(除法的余数)	x%3	1
++	递增(将变量递增 1——这种技术经常用于计数器中)	x + +	11
−−	递减(将变量递减 1)	x - -	9

11.6.2 赋值运算符

基本的赋值运算符是等号，但是不要认为它检查两个值是否相等。相反，它用于将一个值赋给等号左边的变量，在前面的小节中介绍变量时给出了相应的示例。

赋值运算符可以组合其他一些运算符，以便同时执行某种操作并将值赋给变量。例如，对于算术运算符，赋值运算符可用于创建简写版本的运算符，如下所示：

```
total = total - profit
```

该语句可以简写为如下语句：

```
total -= profit
```

虽然该语句看上去可能不是非常直观，但是如果需要执行许多类似于这样的计算，这种类型的简写方式能够节省很多代码。表 11-9 中列举了一些简写方式。

表 11-9

符　号	使用简写方式的示例	不使用简写方式的相等语句
+=	x+=y	x=x+y
−=	x−=y	x=x−y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

11.6.3 比较运算符

如表 11-10 中所示，比较运算符比较两个操作数，然后根据比较的结果返回 true 或 false。注意，用于检查两个操作数是否相等的比较运算符是两个等号(一个等号则是赋值运

算符)。

表 11-10

操 作 符	说　明	示　例
==	等于	1==2 返回 false 3==3 返回 true
!=	不等于	1!=2 返回 true 3!=3 返回 false
>	大于	1>2 返回 false 3>3 返回 false 3>2 返回 true
<	小于	1<2 返回 true 3<3 返回 false 3<1 返回 false
>=	大于等于	1>=2 返回 false 3>=2 返回 true 3>=3 返回 true
<=	小于等于	1<=2 返回 true 3<=3 返回 true 3<=4 返回 false

11.6.4　逻辑或布尔运算符

逻辑或布尔运算符返回 true 或 false。该运算符非常有用，因为它们可用于一次对多个表达式求值。表 11-11 列出了各种逻辑或布尔运算法符。

表 11-11

操 作 符	名　称	说　明	示例(其中 x=1 并且 y=2)
&&	与	检查两个条件是否全部满足	(x < 2 && y > 1) 返回 true(因为两个条件都满足)
??	或	检查两个条件中的一个条件是否满足	(x < 2 ??y < 2) 返回 true(因为第一个条件满足)
!	非	检查某个条件是否都不满足	! (x > y) 返回 true(因为 x 不大于 y)

逻辑或布尔运算符中的两个操作数求值为 true 或 false。例如，如果 x=1 并且 y=2，则 x<2 为 true，y>1 为 true。因此，下面的表达式返回 true，因为两个操作数都求值为 true：

```
(x<2 && y>1)
```

11.6.5 字符串运算符

也可以使用“+”运算符将文本添加到字符串中。例如，下面的“+”运算符将两个字符串变量加在一起：

```
firstName = "Bob"
lastName = "Stewart"
name = firstName + lastName
```

现在 name 变量的值将是 Bob Stewart。将两个字符串加在一起的过程称为字符串连接。

可以使用刚才介绍的比较运算符比较字符串。例如，可以检查用户是否将某个特定的值输入到文本框中(在本章后面介绍条件语句时将会给出关于这个主题的更多内容)。

11.7 函数

现在开始介绍函数，本章前面已经多次提及函数。函数由一些代码组成，当某个事件激活或者调用该函数时执行这些代码。通常一个函数包含多行代码。函数可以在<head>元素内编写，并且可以在页面内的多个位置重写，或者可以位于一个外部文件中，该外部文件在<head>元素中链接。

11.7.1 定义函数的方式

创建或定义函数需要 3 个部分：

- 定义函数的名称。
- 指示哪些值需要作为参数。
- 添加语句。

例如，如果希望创建一个函数计算矩形的面积，可以将函数命名为 calculateArea()(请记住，函数名后应当紧跟圆括号)。然后为了计算面积，需要知道矩形的宽和高，因此将以参数的形式传递它们(参数是函数完成工作需要的信息)。在函数内的大括号之间是语句，它们指明面积等于宽乘以高(宽和高都已经传递给函数)。然后返回面积：

```
function calculateArea(width, height) {
  area = width * height
  return area
}
```

如果函数没有参数，它的名称后面仍然必须具有圆括号，例如 logOut()。

11.7.2 调用函数的方式

如果仅位于文档头中，calculateArea()函数自身不会执行任何操作；它必须被调用。在

这个示例中，可以在一个使用 onclick 事件的简单表单中调用该函数，以便在用户单击 Submit 按钮时计算面积。

下面的表单中包含两个文本输入框，分别用于输入宽度和高度，并且宽度和高度将作为参数传递给函数(ch11_eg05.html)：

```
<form name="frmArea" action="">
Enter the width and height of your rectangle to calculate the size:<br />
Width: <input type="text" name="txtWidth" size="5" /><br />
Height: <input type="text" name="txtHeight" size="5" /><br />
<input type="button" value="Calculate area"
 onclick="alert(calculateArea(document.frmArea.txtWidth.value,
 document.frmArea.txtHeight.value))" />
</form>
```

当 onclick 事件激活时，仔细查看发生的情况。首先调用一个 JavaScript 警告，然后在警告中调用 calculateArea()函数，从而将面积的值写入到警告框中。在圆括号内调用 calculateArea()函数，传递的两个参数分别是宽度文本框的值和高度文本框的值，方式是使用前面关于 DOM 的一节中介绍的点标记。

注意，即使函数没有参数，在调用函数时仍然需要在函数名的后面使用圆括号；例如，下面的函数可以在没有作为参数传递的任何额外信息的情况下运行：

```
<input type="submit" onClick="exampleFunction()" />
```

11.7.3　return 语句

返回结果的函数必须使用 return 语句。这条语句指定某个值将返回给调用函数的位置。例如，calculateArea()函数返回矩形的面积：

```
function calculateArea(width, height) {
  area = width * height
  return area
}
```

某些函数只是返回 true 或 false 值。在本章后面介绍事件时，将会给出返回 false 的某个函数如何阻止某个操作发生。例如，如果与表单中的 onsubmit 事件关联的函数返回 false，则表单不会提交给服务器。

11.8　条件语句

使用条件语句可以根据不同的语句执行不同的操作。存在 3 种类型的条件语句：

- if 语句，如果需要在某个条件为真时执行脚本，则使用该语句。
- if...else 语句，如果希望在条件为真时执行一组代码，而在条件为假时执行另外一组代码，则使用该语句。

- switch 语句，当希望根据一个条件从多个代码块中选择某个代码块执行时，可以使用该语句。

11.8.1 if 语句

if 语句允许在指定的条件为真时执行代码；如果条件为真，则执行大括号内的代码。下面是 if 语句的语法：

```
if (condition)
{
   code to be executed if condition is true
}
```

例如，如果时间是早上，您可能希望在主页上显示文本“Good Morning”。可以使用如下脚本实现该功能(ch11_eg06.html)：

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
</script>
```

如果仅执行一条语句(如此处所示)，则可以省略大括号，因此下面的代码将完成相同的工作(但是最好如前面所示包含大括号)。

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();
  if (time < 12)
    document.write('Good Morning');
</script>
```

这个示例首先创建一个日期对象(本章后面将介绍日期对象)，然后调用日期对象的 getHours()方法来获得时间中的小时数(使用 24 小时计时方法)。如果时间小于 12 小时，则脚本在页面中写入“Good Morning”(如果时间大于 12 小时，则将看到空白页面，因为脚本不会在页面中写入任何内容)。

11.8.2 if...else 语句

如果具有两种可能的情况，并且希望在每种情况下执行不同的操作，则可以使用一条 if...else 语句。这意味着：“如果指定的条件满足，则运行第一块代码；否则运行第二块代码”。语法如下所示：

```
if (condition)
{
```

```
  code to be executed if condition is true
}
else
{
  code to be executed if condition is false
}
```

返回到前面的示例，如果时间在中午之前，则可以写出“Good Morning”；如果时间在中午之后，则写出“Good Afternoon”(ch11_eg07.html)：

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
  else {
    document.write('Good Afternoon');
  }
</script>
```

使用条件语句可以存在很多可能性。事实上，在第 12 章介绍的一个示例中包含多条这样的语句，以创建一些功能强大的复杂示例。

11.8.3　switch 语句

switch 语句允许您处理一个条件的多种结果。该语句具有一个表达式，并且该表达式通常是一个变量。立刻对该表达式求值，然后将表达式的值与结构中每一条 case 语句的值进行比较。如果存在匹配，则执行相应的代码块。

下面是 switch 语句的语法：

```
switch (expression)
{
case option1:
   code to be executed if expression is what is written in option1
   break;
case option2:
   code to be executed if expression is what is written in option2
   break;
case option3:
   code to be executed if expression is what is written in option3
   break;
default:
   code to be executed if expression is different from option1, option2,
   and option3
}
```

这里使用 break 阻止代码自动运行到下一条 case 语句中。例如，可以检查用户在文本框中输入了什么类型的动物，并且根据文本框中输入的动物类型在屏幕上写出不同的内容。

下面是页面中出现的表单。当用户输入了一个动物名并单击按钮时，调用包含在文档头中的 checkAnimal()函数(ch11_eg08.html)：

```
<p>Enter the name of your favorite type of animal that stars in a cartoon:</p>
<form name="frmAnimal">
    <input type="text" name="txtAnimal" /><br />
    <input type="button" value="Check animal" onclick="checkAnimal()" />
</form>
```

下面是包含 switch 语句的函数：

```
function checkAnimal() {
 switch (document.frmAnimal.txtAnimal.value){
   case "rabbit":
     alert("Watch out, it's Elmer Fudd!")
     break;
   case "coyote":
     alert("No match for the road runner - meep meep!")
     break;
   case "mouse":
     alert("Watch out Jerry, here comes Tom!")
     break;
   default : alert("Are you sure you picked an animal from a cartoon?");
  }
}
```

最后一个选项——也是默认选项——在没有 case 语句满足条件时显示。在图 11-6 中可以看到当用户输入 rabbit 之后的显示效果。

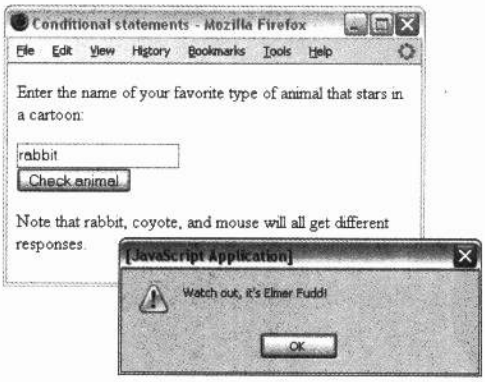


图 11-6

注意，如果用户输入的文本的大小写不同，则它将不会匹配 switch 语句中的选项。因为 JavaScript 区分大小写，如果字母的大小写不匹配 switch 语句中值的大小写，则最终两者不匹配。可以在执行检查之前使用 toLowerCase()方法将所有文本转换为小写形式，以此来解决这个问题。该方法属于内置的 JavaScript 字符串对象，本章后面将介绍该方法。

11.8.4　条件(或三元)运算符

条件运算符(也称为三元运算符)基于一个条件将值赋给变量：

```
variablename=(condition)?value1:value2
```

例如，存在两个变量 instruction 和 color，如果 color 的值是 red，则希望变量 instruction 的值为 STOP，否则希望变量 instruction 的值为 CONTINUE。可以采用如下方式：

```
instruction=(color=="red")?"STOP":"CONTINUE"
```

11.9　循环

循环语句用于执行相同的代码块指定次数：

- while 循环运行相同的代码块直到一个条件为真。
- do while 循环在检查条件之前先运行一次。如果条件为真，则继续运行该循环直到条件为假(while 和 do while 循环之间的区别是，无论条件是否满足，do while 循环都会运行一次)。
- for 循环运行相同的代码块指定次数。

11.9.1　while 循环

在 while 循环内，如果条件为真则执行代码块，并且只要条件为真，代码块将持续执行。语法如下所示：

```
while (condition)
{
   code to be executed
}
```

在下面的示例中，可以看到一个 while 循环显示数字 3 的乘法表。它的工作原理是基于一个计数器 i；每次 while 脚本循环时，计数器递增 1(使用++算术运算符实现递增，如下面包含 i++的代码行所示)。因此，脚本第一次运行时计数器是 1，循环写出 1×3 =3；下一次循环时，计数器的值是 2，因此循环写出 2×3 =6。该循环持续运行直到条件——i 不再小于 11——为真(ch11_eg09.html)：

```
<script type="text/JavaScript">
i = 1
while (i < 11) {
  document.write(i + " x 3 = " + (i * 3) + "<br />" );
    i ++
}
</script>
```

这个示例的结果如图 11-7 所示。

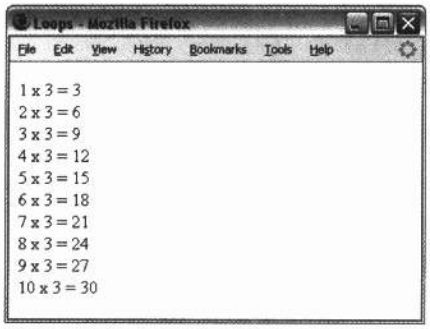


图 11-7

11.9.2 do...while 循环

do...while 循环执行代码块一次，然后检查条件。只要条件为真，它将继续循环。因此，无论条件是什么，循环至少运行一次(因为条件位于指令之后)。下面是 do…while 循环的语法：

```
do
{
  code to be executed
}
while (condition)
```

例如，下面再次是关于 3 的乘法表的示例——计数器的初始值设置为 12，它大于条件中所需的值，因此能够看到 12×3 = 36，但是后面没有任何内容(因为遇到条件时，条件不满足)：

```
<script type="text/JavaScript">
i = 12
do {
  document.write(i + " x 3 = " + (i * 3) + "<br />" );
    i ++
}
while (i < 11)
</script>
```

现在，如果将计数器的初始值改为 1，则结果将与前面的示例相同，直到计数器的值为 11 时循环才结束。

11.9.3 for 循环

for 循环执行代码块指定的次数；当知道需要代码块执行多少次时(而不是运行到某个特定的条件为真或假时，可以使用该循环)。首先，下面是 for 循环的语法：

```
for (a; b; c)
{
  code to be executed
}
```

现在需要了解 a、b、c 分别代表什么：

- a 在循环运行之前求值，并且仅求值一次。它的理想用法是将一个值赋给一个变量；例如可以使用它将计数器设置为 0，方式是 i=0。
- b 应当是一个条件，指明循环是否应当再次运行；如果它返回 true，则循环将再次运行。例如，可以使用它检查计数器是否小于 11。
- c 在循环运行后求值，它可以包含一些乘法表达式，表达式之间以逗号隔开(例如 i++, j++;)。例如，可以使用它递增计数器。

因此，如果再次回到前面 3 的乘法表示例，可以采用如下形式编写它：

```
for (i=0; i<11; i++) {
  document.write(i + " x 3 = " + (i * 3) + "<br />" );
}
```

其中 a 是将计数器赋值 0 的位置；b 是声明条件的位置，即如果计数器的值小于 11，循环将持续运行；c 是每次循环运行后将计数器递增 1 的位置。计数器变量和条件的赋值以及计数器递增都出现在关键字 for 后面的圆括号内。

在对应于字母 a 的部分中可以一次赋值多个变量，方式是使用逗号将它们隔开。例如：

```
i=0, j=5;
```

11.9.4　无限循环和 break 语句

注意，如果循环中的表达式总是求值为真，则该循环称为无限循环。这种循环可能耗尽系统资源，甚至使计算机崩溃，但是某些浏览器会检测无限循环并通常终止它们。

但是，可以添加一个 break 语句来终止无限循环；在下面的循环中，循环次数被设置为 100(ch11_eg12.html)：

```
for (i=0; /* no condition here */ ; i++) {
  document.write(i + " x 3 = " + (i * 3) + "<br />" );
    if (i == 100) {
      break;
    }
}
```

当脚本执行到 break 语句时，它将停止运行。这就能够有效地防止循环运行太多的次数。

11.10　事件

本章前面介绍过事件处理程序用作为 XHTML 元素的属性——例如 onclick 和 onsubmit 事件处理程序。当某些事情发生时事件将产生。有两种类型的事件可用于触发脚本：

- window 事件，当窗口中发生了某些事情时，该事件将产生。例如，页面加载或卸载(被另一个页面替换或关闭)，或者焦点移动到或远离另一个窗口或框架。
- 用户事件，当用户使用鼠标(或其他指向设备)或键盘与页面中的元素交互时，该事件将产生。

存在一组称为固有事件的事件，所有的浏览器都支持这些事件。固有事件与一个元素或元素集合关联，并且它们在标记中用于元素时类似于属性。属性的值是脚本，当事件在该元素上发生时执行该脚本(这可能调用文档的<head>部分中的一个函数)。

例如，onmouseover 事件和 onmouseout 事件可用于改变图像的源属性，从而创建一个简单的图像翻转，如本章前面所示：

```
<a href=""
  onmouseover="document.images.link.src='images/click_red.gif';"
  onmouseout="document.images.link.src='images/click_green.gif'">
  <img src="images/click_green.gif" width="100" height="50" border="0"
  name="link">
</a>
```

表 11-12 提供了大多数常用事件的描述，您很可能会遇到这些事件。

表 11-12

事　件	目　的	应用的元素
onload	文档已经完成加载(如果用于一个框架集中，则所有的框架已经完成加载)	<body> <frameset>
onunload	文档从一个窗口或框架集中卸载或删除	<body> <frameset>
onclick	在元素上单击鼠标(或其他指向设备)上的按键	大多数元素
ondblclick	在元素上双击鼠标(或其他指向设备)上的按键	大多数元素
onmousedown	在元素上按下(但不释放)鼠标(或其他指向设备)上的按键	大多数元素
onmouseup	在元素上释放鼠标(或其他指向设备)上的按键	大多数元素
onmouseover	鼠标(或其他指向设备)移动到元素上	大多数元素
onmousemove	移动位于元素上的鼠标(或其他指向设备)	大多数元素
onmouseout	鼠标(或其他指向设备)移出元素	大多数元素
onkeypress	在元素上按下或释放键	大多数元素
onkeydown	在元素上按住某个键	大多数元素
onkeyup	在元素上释放键	大多数元素
onfocus	元素接收到焦点，方式包括使用鼠标(或其他指向设备)单击该元素、按照焦点移动顺序将焦点移动到该元素上或者使用代码赋予该元素焦点	<a><area><button><input><label><select><textarea>
onblur	元素丢失焦点	<a><area><button><input><label><select><textarea>

(续表)

事　件	目　的	应用的元素
onsubmit	提交表单	<form>
onreset	复位表单	<form>
onselect	用户选择了文本字段中的某些文本	<input> <textarea>
onchange	控件丢失输入焦点，它的值自从获得焦点后已经改变	<input> <select> <textarea>

在本章和下一章中将给出关于使用这些事件的示例。也可以检查从第 1 章到第 6 章中介绍的元素支持哪些方法；几乎每一个元素都能够与一个事件关联。

11.11　内置对象

本章开始部分中介绍了文档对象，现在是时候了解一些内置的 JavaScript 对象。本节将介绍这些内置对象的方法和特性，其中方法可以对数据执行操作，而特性给出关于数据的信息。

说明：

除非额外说明，否则本节中介绍的所有特性和方法都被 Netscape 2 浏览器和 IE3 浏览器以及更高版本的浏览器所支持。

11.11.1　字符串对象

字符串对象可用于处理文本字符串。在开始使用内置对象之前，需要创建该对象的一个实例。可以采用将变量赋予字符串对象的方式创建字符串对象的实例，如下所示：

```
myString = new String('Here is some big text')
```

现在字符串对象包含单词“Here is some big text”。具有这个对象的变量之后，可以将字符串写入文档或者对它执行某些操作。例如，下面的方法写入字符串，如同该字符串位于<big>元素中一样：

```
document.write(myString.big())
```

注意：

如果查看这个元素的源代码，实际上它的内部没有<big>元素；没有启用 JavaScript 的用户完全看不到这些单词。

可以采用如下方式检查这个特性的长度：

```
alert(myString.length)
```

在开始使用字符串对象之前，首先必须创建它并对其赋值。

1. 特性

表 11-13 给出了字符串对象的主要特性和它的目的。

表 11-13

性　质	目　的
length	返回字符串中字符的数量

2. 方法

表 11-14 列举了字符串对象的方法和它们的目的。

表 11-14

方　法	目　的
anchor(name)	创建一个锚点元素(具有 name 或 id 属性而不是 href 属性的<a>元素)
big()	如同文本位于<big>元素中一样显示它
bold()	如同文本位于<bold>元素中一样显示它
charAt(index)	返回指定位置的字符(例如，如果具有一个字符串“banana”，方法调用为 charAt(2)，则将得到字母 n——注意，索引从 0 开始)
fixed()	如同文本位于<tt>元素中一样显示它
fontcolor(color)	如同文本位于具有 color 属性的<font>元素中一样显示它
fontsize(fontsize)	如同文本位于具有 size 属性的<font>元素中一样显示它
indexOf(searchValue, [fromindex])	返回指定字符串 searchValue 在另一个字符串中第一次出现的位置。例如，如果字符串为“banana”，并且希望找到字母 n 第一次出现的位置，则可以使用 indexOf(n)。 如果使用 fromindex 参数，则搜索将从该索引处开始。例如，可能希望从第 4 个字符处开始搜索。 如果搜索的字符串没有出现，则该方法返回－1
italics()	如同文本位于<i>元素中一样显示它
lastIndexOf(searchValue, [fromIndex])	类似于方法 indexOf()，但是从右到左运行
link(targetURL)	在文档中创建一个链接
small()	如同文本位于<small>元素中一样显示它
strike()	如同文本位于<strike>元素中一样显示它
sub()	如同文本位于<sub>元素中一样显示它
substr(start),[length])	返回指定的字符。“14,7”将返回 7 个字符，并且从第 14 个字符开始返回(起始点是 0)。 注意，该方法仅被 IE4 浏览器和 Netscape 4 浏览器以及以后的版本所支持

(续表)

方　法	目　的
substring(startPosition, endPosition)	返回起始索引点和结束索引点之间的指定字符。“7,14”将返回从第 7 个字符到第 14 个字符之间的所有字符，但是不包括第 14 个字符(起始点是 0)
sup()	如同文本位于<sup>元素中一样显示它
toLowerCase()	将字符串转换为小写形式
toUpperCase()	将字符串转换为大写形式

试一试　使用字符串对象

在这个示例中将收集字符串的一部分，并且全部转换为大写字母。这个示例将从文本“Learning about Built-in Objects is easy”中收集单词“Built-in objects”，并将它们转换为大写字母。

(1) 创建 XHTML 文档的程序框架，如下所示：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:land="en">
<head>
  <title>String Object</title>
</head>
<body>
</body>
</html>
```

(2) 因为这个示例中的代码仅在一个位置处执行，所以脚本可以添加到文档主体内。因此添加<script>元素，并且在其内部编写如下代码：

```
<script type="text/JavaScript">
  myString = new String('Learning about Built-in Objects is easy')
  myString = myString.substring(15, 31)
  myString = myString.toUpperCase()
  document.write(myString)
</script>
```

(3) 将这个文件保存为 ch11_eg14.html，当在浏览器中打开它时，将看到如图 11-8 所示的文本。

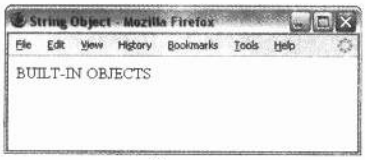


图 11-8

工作原理

这个示例的脚本可以位于文档的主体中，因为它仅用于这个示例(该脚本不需要是一个函数，因为它不会在页面中多次调用，并且不会被其他页面使用)。

值得注意的部分是位于<script>元素内的代码。首先必须创建字符串对象的一个实例，该实例被赋给变量 myString：

```
myString = new String('Learning about Built-in Objects is easy')
```

在创建时，这个字符串对象中保存语句“Learning about Built-in Objects is easy”。但是，这个练习的思想是只选择单词“Built-in Objects”，因此需要使用 substring()方法，该方法的语法如下所示：

```
substring(startPosition, endPosition)
```

因此选择字符串对象(它位于变量 myString 中)，并将其赋值为新的子字符串(即赋值为所需的子字符串)：

```
myString = myString.substring(15, 32)
```

这条语句选择从第 16 个字符到第 33 个字符之间的字符串——因为索引的起始点是 0。

接下来将字符串转换为大写形式，方式是使用 toUpperCase()方法：

```
myString = myString.toUpperCase()
```

最后可以将该字符串写入文档，如下所示：

```
document.write(myString)
```

结果看上去非常简单，但是如果考虑到原始字符串是“Learning about Built-in Objects is easy”，则结果看上去存在本质上的区别。

11.11.2 日期对象

日期对象用于处理日期和时间。可以使用日期构造函数创建一个新的日期对象，如下所示：

```
new Date()
```

可以在创建日期对象的同时将其设置为一个特定的日期或时间，此时需要传递给它如下 4 种参数中的一种：

- milliseconds：这个值应该是从 01/01/1970 开始的毫秒数。
- dateString：可以是格式能够被 parse()方法识别的任意日期。
- yr_num、mo_num、day_num：分别表示年、月和日。
- yr_num、mo_num、day_num、hr_num、min_num、seconds_num、ms_num：分别表示年、月、日、小时、分、秒和毫秒。

下面是一些示例：第一个示例使用毫秒，所代表的日期是 Thu Nov 27 05:33:20 UTC 1975：

```
var birthDate = new Date(8298400000)
document.write(birthDate)
```

第二个示例使用 dateString，所代表的日期是 Wed Apr 16 00:00:00 UTC+0100 1975：

```
var birthDate = new Date("April 16, 1975")
document.write(birthDate)
```

第三个示例使用 yr_num、mo_num 和 day_num，所代表的日期是 Mon May 12 00:00:00 UTC+0100 1975：

```
var birthDate = new Date(1975, 4, 28)
document.write(birthDate)
```

对于日期，需要注意的一些事项包括：

- 第一个令人混淆的事项是，数值 4 对应于月份 May(五月)！这是因为 January(一月)为 0。类似地，在处理日期时，0 表示星期天。
- 您可能会发现获得的时区和此处显示的时区不同。此处的时区是伦敦时区，因此采用的是格林尼治标准时间(GMT)或协调世界时间(UTC)。所有日期对象的操作都采用 UTC 时间执行，即使您的计算机显示的时间可能与所在的时区一致。
- 虽然可以将日期相加或相减，但得到的结果将是毫秒数。例如，如果希望找出从现在到本年结束时的天数，可以使用如下代码：

```
var today = new Date()
var newYear = new Date(2008,11,31)
var daysRemaining = (newYear - today)
document.write(daysRemaining)
```

这段代码存在的问题是，最终得到的结果非常长(无论是在 2008 年中阅读它时采用加法，还是在 2008 年之后阅读它时采用减法)。由于每一天具有 86 400 000 毫秒，因此很可能看到一个非常大的数字。

因此，需要将 daysRemaining 除以 86 400 000 以获得天数：

```
var today = new Date()
var newYear = new Date(2008,11,31)
var daysRemaining = (newYear - today)
daysRemaining = daysRemaining/86400000
document.write(daysRemaining)
```

需要牢记的是，用户的计算机时钟可能不够精确，并且事实上不同的用户可能处于不同的时区，使用日期对象可能很快变得非常复杂。如果提供的时间和用户计算机上的时区不同，则计算某个事件之前的天数可能导致产生不精确的答案。

如果可以的话，最好使用一种服务器端脚本语言向访问者提供类似时间方面的内容。但是，如果通过用户的输入指定日期，或者在日期格式可以控制时通过服务器指定日期，则这种方式就非常有用。

表 11-15 给出日期对象的一些常用方法。

表 11-15

方　法	目　的
date()	返回一个 Data 对象
getDate()	返回 Data 对象的日期(从 1 到 31)
getDay()	返回 Data 对象是星期几(从 0 到 6；0＝星期天，1＝星期一，依此类推)
getMonth()	返回 Data 对象的月份(从 0 到 11；0＝一月，1＝二月，依此类推)
getFullYear()	返回 Data 对象的年份(4 个数字)，它仅被 Netscape 4 浏览器和 IE4 浏览器以及更高版本的浏览器所支持
getYear()	返回 Data 对象的年份，仅使用两个数字(从 0 到 99)。 应当使用 getFullYear()方法取代，因为它提供包含 4 个数字的年份
getHours()	返回 Data 对象的小时(从 0 到 23)
getMinutes()	返回 Data 对象的分钟(从 0 到 59)
getSeconds()	返回 Data 对象的秒(从 0 到 59)
getTime()	返回从 1/1/1970 的午夜开始的毫秒数
getTimezoneOffset()	返回用户计算机和 GMT 之间的时差
parse()	返回一个字符串日期值，该值包含从 January 01 1970 00:00:00 开始的毫秒数
setDate()	设置 Date 对象中月份的日期(从 1 到 31)
setFullYear()	设置 Date 对象中的年份(4 个数字)，该方法仅被 Netscape 4 浏览器和 IE4 浏览器以及以后的版本所支持
setHours()	设置 Date 对象中的小时(从 0 到 23)
setMinutes()	设置 Date 对象中的分钟(从 0 到 59)
setMonth()	设置 Date 对象中的月份(从 0 到 11；0＝一月，1＝二月，依此类推)
setSeconds()	设置 Date 对象中的秒(从 0 到 59)
setTime()	设置从 1/1/1970 之后的毫秒数
setYear()	设置 Date 对象中的年份(从 00 到 99)
toGMTString()	将 Date 对象转换为字符串，设置为 GMT 时区
toLocaleString()	将 Date 对象转换为字符串，设置为当地时区
toString()	将 Date 对象转换为字符串，该方法仅被 Netscape 2 浏览器和 IE4 浏览器以及后面的版本支持

表 11-16 中的许多方法在版本 4 以上的浏览器中添加，这些浏览器提供对 UTC 时间的支持，该时间采用的格式是 Day Month Date, hh,mm,ss UTC Year。

表 11-16

方　法	目　的
getUTCDate()	返回采用 UTC 时间的 Date 对象的日期
getUTCDay()	返回采用 UTC 时间的 Data 对象是星期几.
getUTCMonth()	返回采用 UTC 时间的 Date 对象的月份
getUTCFullYear()	返回采用 UTC 时间的 Date 对象的年份(4 个数字)
getUTCHours()	返回采用 UTC 时间的 Date 对象的小时
getUTCMinutes()	返回采用 UTC 时间的 Date 对象的分钟
getUTCSeconds()	返回采用 UTC 时间的 Date 对象的秒
getUTCMilliseconds()	返回采用 UTC 时间的 Date 对象的毫秒
setUTCDate()	设置采用 UTC 时间的 Date 对象中的日期(从 1 到 31)
setUTCDay()	设置采用 UTC 时间的 Data 对象是星期几(从 0 到 6；0＝星期天，1＝星期一，依此类推)
setUTCMonth()	设置采用 UTC 时间的 Date 对象中的月份(从 0 到 11；0＝一月，1＝二月，依此类推)
setUTCFullYear()	设置采用 UTC 时间的 Date 对象中的年份(4 个数字)
setUTCHour()	设置采用 UTC 时间的 Date 对象中的小时(从 0 到 23)
setUTCMinutes()	设置采用 UTC 时间的 Date 对象中的分钟(从 0 到 59)
setUTCSeconds()	设置采用 UTC 时间的 Date 对象中的秒(从 0 到 59)
setUTCMilliseconds()	设置采用 UTC 时间的 Date 对象中的毫秒(从 0 到 999)

11.11.3　数学对象

数学对象用于处理数值；它不需要构造函数。数学对象具有一些用于表示数学常量的特性，例如 pi 和 10 的自然对数(近似等于 2.3026)；并且具有一些方法，这些方法表示一些数学函数，例如正切函数或正弦函数。

例如，下面的代码将称为 numberPI 的变量设置为保存 pi 的常量，然后将该变量写出到屏幕上(ch11_eg16.html)：

```
numberPI = Math.PI
document.write (numberPI)
```

下面的示例将 pi 舍入为最接近的整数，并且将其写出到屏幕上：

```
numberPI = Math.PI
numberPI = Math.round(numberPI)
document.write (numberPI)
```

1. 特性

表 11-17 列举了 Math 对象的一些特性。

表 11-17

性　质	目　的
E	返回自然对数的基数
LN2	返回 2 的自然对数
LN10	返回 10 的自然对数
LOG2E	返回 E 的基数为 2 的对数
LOG10E	返回 E 的基数为 10 的对数
PI	返回 pi
SQRT1_2	返回 2 的平方根的倒数
SQRT2	返回 2 的平方根

2. 方法

表 11-18 列举了 Math 对象的一些方法。

表 11-18

方　法	目　的
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(y,x)	返回从 x 轴到一个点的角度
ceil(x)	返回大于或等于 x 的最接近的整数
cos(x)	返回 x 的余弦值
exp(x)	返回 E 的 x 次幂
floor(x)	返回小于或等于 x 的最接近的整数
log(x)	返回 x 的自然对数
max(x,y)	返回 x 和 y 之间的最大值
min(x,y)	返回 x 和 y 之间的最小值
pow(x,y)	返回 x 的 y 次幂
random()	返回 0 到 1 之间的一个随机数
round(x)	将 x 舍入为最接近的整数
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

11.11.4　数组对象

数组类似于特殊的变量，它的特殊性在于能够保存多个值，并且这些值能够独立访问。当希望在同一个变量中存储一组值而不是将每个值放到单独的变量中时，数组就非常有用。希望这样操作是因为所有的值对应于一个特殊的项；或者只是因为将一些值放入同一个变量而不是具有不同名称的多个变量中较为方便；或者是因为不知道将有多少信息项需要存储。您经常会看到数组与循环结合使用，其中循环用于将信息添加入数组或者从数组中读取信息。

为了创建数组，需要使用 Array 对象的构造函数，方式是指定数组的名称和它将保存的值的数量，或者直接将所有数据添加到数组中。例如，下面的数组保存乐器的名称：

```
instruments = new Array("guitar", "drums", "piano")
```

可以利用序数索引数组中的元素，序数从 0 开始。因此，可以利用 instruments[0]访问吉他(guitar)，利用 instruments[1]访问鼓(drums)等。

如果在创建数组时不希望提供所有的值，可以仅指示希望在数组中保存的元素数量(注意，这个值不是从 0 开始，因此下面的语句创建具有 3 个(而不是 4 个)元素的数组)：

```
instruments = new Array(3)
```

现在元素数量存储在 Array 对象的 length 特性中，并且实际上还没有对元素赋值。如果希望增加数组的大小，可以将一个新值赋给 length 特性，该新值大于当前的长度。

下面的示例创建了一个具有 5 项的数组，然后使用 length 特性检查数组中具有多少项：

```
fruit = new Array("apple", "banana", "orange", "mango", "lemon")
document.write(fruit.length)
```

下面是一个关于 toString()方法的示例，该方法将数组转换为字符串：

```
document.write('These are ' + fruit.toString())
```

将相关信息保存在一个变量中比将它们保存在 5 个变量(例如 fruit1、fruit2、fruit3、fruit4 和 fruit5)中更容易。使用一个这样的数组所占用的内存少于使用 5 个独立的变量，并且当具有数量不断变化的水果时，数组可以根据需求增加或减少项的数量(而不是创建 10 个变量，其中一半变量可能为空)。

1. 方法

表 11-19 列举了 Array 对象的方法。

表 11-19

方　法	目　的
concat()	连接两个或多个数组，从而创建一个新数组；该方法被 Netscape 4 浏览器和 IE4 浏览器以及以后版本的浏览器所支持
join(separator)	将数组中的所有元素连接在一起，元素之间使用指定的字符分隔符隔开(默认为逗号)；该方法被 Netscape 3 浏览器和 IE4 浏览器以及以后版本的浏览器所支持

(续表)

方　法	目　的
reverse()	反向返回数组；该方法被 Netscape 3 浏览器和 IE4 浏览器以及以后版本的浏览器所支持
slice()	返回数组的指定部分；该方法被 Netscape 4 浏览器和 IE4 浏览器以及以后版本的浏览器所支持
sort()	返回已排序的数组；该方法被 Netscape 3 浏览器和 IE4 浏览器以及更新版本的浏览器所支持

11.11.5 Window 对象

每个浏览器窗口和框架都对应于一个 Window 对象，它是随着<body>或<frameset>元素的每个实例一起创建的对象。

例如，可以使用 status 特性更改出现在浏览器状态栏中的文本。为了完成该任务，首先需要在页面的头部中添加一个函数，该函数将在页面加载时触发，用于指明应当出现在状态栏中的内容：

```
<script type="text/javascript">
  function statusBarText()
  {
    window.status = "Did you see me down here?"
  }
</script>
```

然后在<body>元素的 onload 事件中调用这个函数，如下所示：

```
<body onload="statusBarText()">
```

下面的示例给出如何打开一个新的弹出式窗口；在第 12 章中将介绍用于执行这项任务的更高级的函数(该函数位于文档头部中)，但是如您所见，这个示例在事件处理程序中提供了一个内联的脚本：

```
<input type="button" value="Open Window"
onclick="window.open('http://www.wrox.com')">
```

1. 特性

表 11-20 列举了 Window 对象的特性。

表 11-20

性　质	目　的
closed	一个布尔值，用于确定窗口是否已经关闭。如果窗口已经关闭，则返回值为 true
defaultStatus	定义显示在浏览器窗口状态栏(通常位于页面底部的左边)中的默认消息
document	包含在该窗口中的文档对象

(续表)

性　质	目　的
frames	一个数组，它包含了对当前窗口中的所有已命名子框架的引用
history	一个历史对象，它包含了从该窗口中访问的一些详情和 URL(主要用于创建前进和后退按钮，类似于浏览器上的对应按钮)
location	位置对象：当前窗口的 URL
name	窗口的名称
status	可以随时设置以定义显示在状态栏的临时信息，例如，当用户将鼠标悬停在一个链接上时改变状态栏中的信息，方式是将该特性用于链接的 onmouseover 事件
statusbar	指示状态栏是否可见，具有它自己的特性 visible，该特性的值是布尔值 true 或 false——例如，window.statusbar[.visible=false](该特性被 Netscape 4 浏览器和 IE3 浏览器以及以后版本的浏览器所支持)
toolbar	指示滚动栏是否可见，具有它自己的特性 visible，该特性的值是布尔值 true 或 false——例如，window.toolbar[.visible=false]。仅当创建新窗口之后才可以设置该特性(该特性被 Netscape 4 浏览器和 IE3 浏览器以及以后版本的浏览器所支持)
top	对最顶端的浏览器窗口的引用，前提是桌面上同时打开了多个窗口
window	当前的窗口或框架

方法

表 11-21 列举了 Window 对象的方法。

表 11-21

方　法	目　的
alert()	显示一个警告框，警告框中包含一条消息和一个 OK 按钮
back()	具有与浏览器的 Back 按钮相同的效果
blur()	从当前窗口中移除焦点
close()	关闭当前窗口或另外一个窗口(如果提供另外一个窗口的引用)
confirm()	生成一个对话框，询问用户以确认他们希望执行的操作，使用 OK 或 Cancel 作为选项，它们分别返回 true 和 false
focus()	将焦点赋予指定的窗口，并将其转到其他窗口的顶部
forward()	等价于单击浏览器的 Forward 按钮
home()	将用户转到主页中
moveBy(horizonatalPixels, verticalPixels)	将窗口移动相对于当前坐标指定像素数量的距离
moveTo(Xpostion,Yposition)	将窗口左上角移动到指定的(x，y)坐标处
open(URL,name[,features])	打开一个新的浏览器窗口(将在下一章中更详细地介绍这个方法)

(续表)

方　法	目　的
print()	打印当前窗口的内容
prompt()	创建一个对话框，让用户输入内容
stop()	具有与单击浏览器中的 Stop 按钮相同的效果

11.12 编写 JavaScript 代码

开始编写 JavaScript 代码之前，需要注意以下一些方面：

- JavaScript 区分大小写，因此称为 myVariable 的变量与称为 MYVARIABLE 的变量是不同的，并且它们均与称为 myvariable 的变量不同。
- 当遇到(、{、[、"和'等符号时，必须具有匹配的结束符号'、"、]、}和)。
- 与 XHTML 一样，JavaScript 忽略额外的空格，因此可以在脚本中添加一些空白，以便获得更好的可读性。下面两行代码是等价的，即使第二行中具有更多的空格：

```
myVariable="some value"
myVariable = "some value"
```

- 可以利用一个反斜杠中断代码行中的文本字符串，如下所示，如果字符串很长，这种方式就非常有用：

```
document.write("My first \
  JavaScript example")
```

- 但是，除了字符串之外的其他内容不能中断，因此下面的代码是错误的：

```
document.write \
  ("My first JavaScript example")
```

- 在插入一些保留的特殊字符时，例如"、'、;和&，需要在它们前面添加反斜杠，如下所示：

```
document.write("I want to use a \"quote\" mark \& an ampersand.")
```

该代码在浏览器中输出如下内容：

```
I want to use a "quote" mark & an ampersand.
```

- 如果已经使用过一种完整的编程语言，例如 C++或 Java，则将会知道它们要求每一行代码的末尾都有一个分号。通常来说，在 JavaScript 中分号是可选项，除非希望在一行中放置多条语句。

11.12.1 关于数据类型的注意事项

现在您应当已经了解能够对不同类型的数据执行不同的操作。例如，可以将两个数值

相加在一起，但是不能采用数学方法将字母 A 与字母 B 相加。某些类型的数据要求能够处理具有小数点的数值(浮点数)；货币是最常见的示例。其他一些类型的数据具有一些固有的限制；例如，如果处理的是日期和时间，则可能会希望将小时与特定类型的数据相加，并且得到的时间结果不会是 25:30(即使经常希望一天具有更多的小时)。

不同类型的数据(字母、整数、十进制数、日期)称为具有不同的数据类型；它们使程序能够以不同的方式管理不同类型的数据。例如，如果对字符串使用“＋”运算符，则是将两个字符串连接在一起；而如果将“＋”运算符用于数值，则是将两个数值相加在一起。某些编程语言要求程序员专门指示变量的类型，并且要求程序员能够在类型之间进行转换。虽然 JavaScript 提供了一些不同的数据类型(本章后面将介绍这些数据类型)，但是它自己处理类型之间的转换，因此不需要担心告诉 JavaScript 特定类型的数据是日期或字符串(字符串由一系列字符组成，字符可以包括字母和数值)。

在 JavaScript 中存在 3 种简单的数据类型：

- Number：用于执行算术操作(加、减、乘、除)。没有出现在双引号之间的整数或小数都被视为数值。
- String：用于处理文本。它由一系列的字符组成，并且由双引号包围。
- Boolean：一个布尔值仅具有两种可能的值：true 和 false。这种数据可用于执行逻辑操作以及检查某些事物的真假性。

您很可能遇到过其他两种数据类型：

- Null：表明一个值不存在，它写作为关键字 null。Null 是非常重要的值，因为它显式地表明没有提供任何值。它不同于只包含空格或 0 的字符串。
- Undefined：表明值在前面的代码中没有定义，它写作为 JavaScript 的关键字 undefined。如果声明一个变量但没有对其赋值，则认为该变量是 undefined(当代码不正确时，很可能就是这种原因)。

11.12.2　关键字

您可能已经注意到，在 JavaScript 中存在很多用于执行功能的关键字，例如 break、for、if 和 while，它们具有特殊的意义；因此，这些单词不能用作为变量名、函数名、方法名或对象名。下面列举了一些应该避免使用的关键字(其中的一些关键字实际上没有使用过，但是被保留以供将来使用)：

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

如果操作的页面包含多种脚本语言，为了指示一种默认的脚本语言，应当在文档的<head>部分中使用<meta>元素。

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

11.13 本章小结

本章引入了很多新概念：对象、方法、特性、事件、数组、函数、API、对象模型、数据类型和关键字。虽然不能一次介绍完这些概念所有方面，但是随着阅读下一章中的一些示例，您将会更清晰地了解它们。在阅读完下一章之后，可以重新阅读本章，应当能够理解使用 JavaScript 可以实现的更多示例程序。

在本章中首先介绍如何使用文档对象模型访问文档中的信息。本章关注于介绍 Level 0 DOM，它不像其他已经发布的 3 种级别的 DOM 一样是一种 W3C 规范。相反，它主要基于 Netscape 2 浏览器和 IE3 浏览器以及以后版本的浏览器中的常用功能。

W3C 正转向致力于提供一种访问所有 XML 文档(包括 XHTML 文档)的标准方式；但是，人们已经使用 DOM Level 0 代码编写了太多的脚本，因此它仍然是开始学习 JavaScript 并编写可应用于大多数浏览器的代码的最佳方式。

在了解如何从文档中获得信息之后，可以使用 JavaScript 对文档中的数据执行计算。JavaScript 主要使用如下功能来执行计算：

- 变量(用于在内存中存储信息)
- 运算符(例如算术运算符和比较运算符)
- 函数(函数位于文档的<head>部分中，包含一些被事件调用的代码)
- 条件语句(用于处理基于不同情形的操作选择)
- 循环(为了重复一些语句直到一个条件满足)

在第 12 章中您将看到，这些简单的概念能够组合在一起，从而创建非常强大的结果。特别是在查看一些验证脚本(验证脚本用于检查用户输入在表单中的数据)时，您将看到一些非常高级的 JavaScript 代码，并且将理解如何使用一些基本的构件创建复杂的结构。

最后，本章介绍了其他一些 JavaScript 对象：String、Data、Math、Array 和 Window 对象。每个对象包含一些相关的功能；它们具有一些特性和方法，其中特性提供对象的信息(例如日期、时间、窗口的大小或字符串的长度)，而方法可用于处理与对象相关的数据。

我希望您已经掌握如何使用 JavaScript 在页面中添加交互性，但是在下一章中，在深入了解 JavaScript 库并查看一些有助于使用 JavaScript 的示例之后，您将真正地掌握这一点。

11.14 练习

1. 创建一个脚本，该脚本输出数值 5 的从 1 到 20 的乘法表，方法是使用 while 循环。
2. 修改 ch11_eg06.html，使其能够完成以下功能：

- 向中午 12 点之前到达页面的访问者输出“Good Morning”(使用一条 if 语句)。
- 向中午 12 点到下午 6 点之间到达页面的访问者输出“Good Afternoon”(再次使用一个 if 语句)。(提示：可能需要使用逻辑运算符)。
- 向下午 6 点到午夜之间到达页面的访问者输出“Good Evening”(再次使用一条 if 语句)。

第12章

应用 JavaScript

第 11 章中介绍了关于 JavaScript 语言的一些关键概念；在本章中将介绍如何组合利用这些概念编写脚本。通过学习许多示例，您将了解 JavaScript 能够利用不同的方式与 Web 页面交互。本章也将介绍一些用于编写自己的 JavaScript 脚本的新编码实践，以及一些用于创建交互页面的非常有用的捷径。

本章介绍以下主题，每一个主题与不同的 JavaScript 技术或文档的不同部分相关：

- 表单的验证：检查用户填写合适的表单元素，并且填写的值是否匹配开发人员期望的值。
- 其他表单技术：当页面加载时将焦点赋予元素、字段之间的自动焦点移动、禁止控件以及文本大小写的转换。
- 导航：图像翻转和突出显示导航项
- 窗口：创建弹出式窗口
- 使用已有的库：查看 3 个已有的 JavaScript 库，通过它们可以使用少量的代码实现复杂的功能。

阅读完本章之后，不仅可以学习到很多关于在页面中使用 JavaScript 的知识，而且将具有包含有用函数的库，可以在自己的页面中使用这些函数。

12.1 关于编写脚本的一些实用提示

在开始查看示例之前，应当了解一些关于开发 JavaScript 脚本的实用提示，它们能够节省大量的开发时间。

12.1.1 其他人是否已经编写过这个脚本

Web 中具有数以千计的免费 JavaScript 脚本，在开始编写某个脚本以执行操作之前，最好浏览一下这些站点，查看其他人是否已经完成过相同的工作。当然，某些任务要求创建自己的脚本，但是如果可以利用一些已经编写的脚本，则不需要重新编写它们；应当考虑使用这些脚本。

下面的站点将帮助您获得一些脚本代码(同时不要忘记可以使用搜索引擎搜索脚本代码，例如使用 Google)：

- www.HotScripts.com

- www.JavaScriptKit.com
- http://JavaScript.Internet.com

即使无法确切地复制其他人的脚本，通过查看其他人如何完成相同的任务也可以学到很多知识。

在本章结尾处，当介绍利用已有的 JavaScript 库时将给出关于这个主题的更多内容。

12.1.2 可重用的函数

除了重用其他人的脚本和文件夹，还应当编写自己能够重用的代码。例如，如果打算编写一个抵押计算器的函数，最好在调用该函数时将值传递给它，而不是编写函数以从表单中获取值。考虑如下函数：

```
calculateLoan(loanAmount, repaymentPeriod, interestRate)
```

这个函数采用 3 个参数，当调用时需要将这 3 个参数传递给它。

现在设想文档中具有一个调用这个函数的表单。该<form>元素将具有一个 onsubmit 事件处理程序，从而当用户单击按钮计算应当偿还的借款数量时调用该函数。因为该函数要求传递 3 个参数，对该事件的调用可能如下所示：

```
<form name="frmLoanCalc"
      onsubmit="calculateLoan(document.frmLoanCalc.txtAmount.value,
                              document.frmLoanCalc.txtRepayment.value,
                              document.frmLoanCalc.txtInterest.value)">
```

您可能会认为让函数自己从表单中收集这些值更好；然后可以只在一行中调用该函数如下所示：

```
<form name="frmLoanCalc" onsubmit="calculateLoan()"
```

第二种方法看上去更易于编写，但实际情况并非如此。最好将这些值传递给函数，而不是让函数自己从表单中收集值。因为函数将不得不从收集值开始，并且最终将可能如下所示：

```
function calculateLoan() {
  loanAmount = document.frmLoanCalc.txtAmount.value
  loanValue = document.frmLoanCalc.txtRepayment.value
  interestRate = document.frmLoanCalc.txtInterest.value
```

如果不这样操作，在调用该函数时也需要编写相同的代码量，为什么还会存在问题呢？答案是如果函数从表单中收集信息，则它将仅能用于该页面和该表单。通过将值传递给函数，与第一种方法一样，借贷计算可以用于许多不同的表单。

您可能会认为在许多站点中不需要抵押计算器，但是可能在不同的站点中需要某种其他形式的借贷计算器。例如，可能需要为汽车销售商编写一个站点，销售商希望用户了解如果他们采用分期付款，应当总共偿还多少钱——此时可以再次使用自己的借贷计算器。通过使函数通用化，可以重用它们，而重用自己的代码将节省很多时间。

12.1.3　使用外部 JavaScript 文件

如果打算在多个页面中使用同一个脚本，则最好将其放置在一个外部 JavaScript 文件中(在第 11 章开始部分介绍过该技术)。例如，在本章后面的“图像翻转”一节中将给出一个示例脚本，为导航栏创建图像翻转。导航将出现在每一个页面中，因此与其将图像翻转函数放置在每一个页面中，不如只将一个脚本包含在每一个页面中。这样操作具有如下 3 方面的优点：

- 如果需要改变导航，则仅需要改变一个函数而不是每一个页面。
- 页面的文件尺寸将更小，因为 JavaScript 代码位于一个文件中并被每个页面包含，而不是在每个需要它的页面中重复。
- 不需要将相同的代码复制并粘帖到多个文件中。

12.1.4　将脚本放置在 scripts 文件夹中

当使用外部脚本时，应当创建一个特殊的文件夹 scripts——如同需要一个 images 文件夹一样。这样有助于改进站点和目录结构的组织。当需要查看或改变一个脚本时，可以确切地知道它位于哪里。

也应当为脚本文件使用直观的名称，以便能够快速并容易地找到它们。

12.2　表单验证

表单验证是利用 JavaScript 执行的最常见任务。您很可能已经在 Web 中遇到过一些表单，当没有输入值或者输入错误类型的值时，这些表单会给出提示；这是因为这些表单已经被验证。也就是说，它们检查用户输入的文本或进行的选择是否匹配程序员在页面中写入的规则。这些规则可能包括：e-mail 地址必须包含一个@符号或用户名至少需要 5 个字符长。这些类型的规则有助于确保在提交表单之前用户提供的数据满足需求。

12.2.1　什么时候验证

验证可以在两个位置中发生，即支持 JavaScript 的浏览器处或服务器上。事实上，大多数使用表单收集重要信息的应用程序(例如电子商务订单)将在浏览器端和服务器上同时验证。在浏览器处进行验证的原因是，它可以帮助用户输入完成工作所需的正确数据，而不需要将表单发送给服务器，进行处理，然后当出现问题时再次发送回来。它能够更快地强制用户修改错误，然后再将表单提交给服务器。然后服务器再次检查表单数据，然后才将它们发送给应用程序的其他部分——执行第二层的验证是因为，数据库中的一个简单错误能够阻止应用程序正确地运行，并且如果用户没有启用 JavaScript 功能，则应用程序不会允许用户提交的值没有在浏览器中使用 JavaScript 代码进行检查。

12.2.2 如何验证

当验证表单时，不能始终检查用户是否已经输入正确的信息，但是可以检查他们输入的信息的格式是否正确。例如，不能确保用户输入正确的电话号码；用户可以输入其他人的电话号码，但是可以检查它是数值而不是其他字符，并且可以检查该数值包含了最少需求的数字数量。作为另外一个示例，无法确保某个人输入真实的 e-mail 地址，但是可以检查输入的文本是否遵循 e-mail 地址的通用结构(至少包括一个@符号和一个句点)。因此，表单验证通过验证表单的控件能够最小化用户可能带来的错误。

通常在 onsubmit 事件处理程序中验证表单，该事件处理程序触发存储在文档头部(或者文档头部中指定的一个外部文件)中的验证函数，从而当用户按下 Submit 按钮时检查值。为了发送表单，这个函数必须返回 true。如果遇到一个错误，则该函数返回 false 并且用户的表单不会发送——如果存在错误，表单应当提示用户输入的什么内容存在问题。

onsubmit 事件通常调用具有 validate(form)或 validateForm(form)这样的名称的函数。因为许多表单包含多个要求验证的控件，通常不是将正在检查的每一项的值都传递给验证函数。通常显式地为表单编写验证函数——但是可以在不同表单中重用一些已经学习的技术(或者甚至为登录表单或注册表单重用整个函数)。

注意：

如果使用了验证函数，该函数被 onsubmit 事件处理程序调用，但是用户的浏览器不支持 JavaScript，则表单仍然会提交，但是不执行验证检查。

验证函数中的第一项任务是设置一个变量，该变量用于函数的返回值，初始设置为 true。然后检查输入的值，当函数发现用户的输入中存在错误时，则将该变量的值设置为 false，以阻止表单提交。

1. 检查文本字段

您很可能已经看到 Web 站点上的表单要求提交用户名和密码，然后重新输入密码以确保正确输入。这种表单类似于图 12-1 所示。

图 12-1

在这样的表单中，可能希望检查以下内容：

- 用户名具有要求的最小长度
- 密码具有要求的最小长度
- 两个密码匹配

即将查看的验证函数 validate()位于文档头部中的如下<script> 标签之间(记住，如果打算在其他页面中重用该函数，它可以位于外部 JavaScript 文件中)：

```
<script type="text/JavaScript">
</script>
```

首先，validate()函数将称为 returnValue 的变量赋值为 true；如果没有发现错误，这将是函数的返回值，它将允许发送表单。然后表单收集表单控件的值并放置到不同的变量中，如下所示：

```
function validate(form) {
  var returnValue = true;
  var username = frmRegister.txtUserName.value;
  var password1 = frmRegister.txtPassword.value;
  var password2 = frmRegister.txtPassword2.value;
```

需要执行的第一项操作是检查用户名是否至少具有 6 个字符长：

```
if(username.length < 6) {
  returnValue = false;
  alert("Your username must be at least\n6 characters long.\n
    Please try again.");
  frmRegister.txtUserName.focus();
}
```

username 变量的 length 特性用于检查输入的用户名长度是否大于 6 个字符。如果不是，则函数的返回值为 false，表单不会提交，并且用户将看到一个具有指定错误消息的警告框。注意，通过对存在问题的表单控件使用 focus()方法，可以将焦点传递回该表单控件，以便节省用户再次浏览表单查找该项的时间。也可以看到，这个示例在警告框中使用了一个换行符\n，以将提供给用户的消息换行。

接下来需要检查第一个密码的长度——这次采用相同的方法，而且如果密码的长度不够，则将两个密码框都再次设置为空，并将焦点赋予第一个密码框：

```
if (password1.length < 6) {
  returnValue = false;
  alert("Your password must be at least\n6 characters long.\n
  Please try again.");
  frmRegister.txtPassword.value = "";
  frmRegister.txtPassword2.value = "";
  frmRegister.txtPassword.focus();
}
```

如果代码执行到这里并通过，则表明用户名和第一个密码都足够长。现在仅需要检查第一个密码框的值是否与第二个密码框相同，如下所示。记住，这个条件中使用的“!=”运算符意思是“不等于”：

```
    if (password1.value != password2.value) {
      returnValue = false;
      alter("Your password entries did not match.\nPlease try again.");
      frmRegister.txtPassword.value = "";
      frmRegister.txtPassword2.value = "";
      frm Register.txtPassword.focus();
    }
```

当用户输入了不匹配的密码时，向用户显示一个警告框，警告框中的错误消息提示“输入的密码不匹配”。另外，清除两个密码输入框的内容并且将焦点传递回第一个密码框。

当用户在密码输入框中输入错误密码时，没有必要保留密码表单控件中的值，因为用户不能看到他们已经输入的值(因为密码输入框显示的是点或者星号，而不是字符)。因此，用户将不得不再次输入两个值，因为他们不能看到错误在哪里。

最后执行的操作是将 returnValue 变量的值返回——如果所有的条件都满足，则返回 true，否则返回 false。

```
    return returnValue;
    }
```

下面是用于这个示例的表单(ch12_eg01.html)：

```
    <form name="frmRegister" method="post" action="register.aspx"
      onsubmit="return validate(this);">
      <div class="label"><label for="txtUsername">Username:</label></div>
      <div class="formElement">
        <input type="text" name="txtUserName" id="txtUserName" size="12" />
      </div>
      <div class="label"><label for="txtPassword">Password: </td></label></div>
      <div class="formElement">
        <input type="password" name="txtPassword" id="txtPassword" size="12" />
      </div>
      <div class="label"><label for="txtPassword2">Confirm your
password:</label></div>
      <div class="formElement">
        <input type="password" name="txtPassword2" id="txtPassword2" size="12" />
      </div>

      <div class="label"> </label></div>
      <div class="formElement"><input type="submit" value="Log in"  /></div>

    </form>
```

图 12-2 给出了用户输入的密码不够长时的结果。

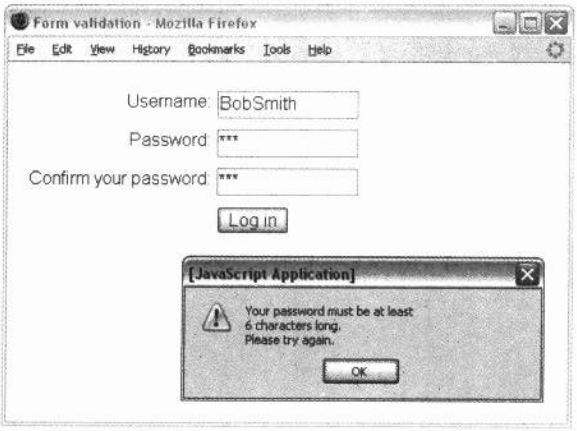


图 12-2

2. 必须填写的文本字段

通常需要确保用户已经在某个文本字段中输入了值。通过使用前面示例中对用户名所使用的技术可以实现这一点。前面介绍过如果用户输入的值少于 6 个字符长，他们会被警告，并且表单不会提交。

另一种可选的技术是使用一个 for 循环遍历所有必须填写的元素，如果其中任意一个元素为空，则返回错误。当使用这种技术时，对于每个必须填写的表单元素，它们需要具有一个值为 required 的 class 属性，以便告诉循环文本输入框必须具有一个值；这些表单元素必须也具有一个 name 属性，该属性的值匹配元素的标签(因为标签通常用于错误消息中)。下面的示例给出了具有 name 属性和 class 属性的文本输入框的形式：

```
<input type="text" name="Username" size="5" class="required" />
```

这一次 validate()函数循环遍历表单的元素，以检查每一个元素的 class 属性值是否为 required。如果是，则函数检查元素值是否为空。使用 onsubmit 事件再次触发该函数。向该函数传递一个表单对象作为参数，并且首先将返回值设置为 true：

```
function validate(form) {
  var returnValue = true;
```

然后函数遍历表单中的元素，以查找必须填写的元素。可以看到，for 循环具有 3 个实参。第一个实参将称为 i 的变量初始化为 0 并作为计数器。第二个实参是一个条件，用于查看计数器是否小于表单中元素的数量。第三个实参在每次循环运行后将计数器递增 1，如下所示：

```
var formElements = form.elements;
for (var i=0; i<formElements.length; i++)
  {
```

然后，在循环内获取当前元素的值：

```
currentElement = formElements[i];
```

现在需要查看 class 属性的值是否是 required，如果是，则检查元素的值是否为空。如果两个条件都满足，则用于返回值的变量被设置为 false，并且警告框告诉用户出错的位置。这里也存在一条 break 语句，以便只要找到某个必须填写的字段为空就终止循环。

```
        if (currentElement.value=="" && currentElement.className=="required") {
        alert("The required field \""+currentElement.name +"\" is empty. Please
        provide a value for it.");
        currentElement.focus();
        returnValue = false;
      break;
      }
    return returnValue;
  }
}
```

注意此处 alert()方法的用法，该方法使用元素 name 属性的值(Current Element.name)告诉用户没有填写哪个元素。

这个函数处理表单的方式与上一个示例非常相似，但是 name 属性的值必须对用户具有描述性并匹配表单的标签(ch12_eg02.html)：

```
<form name="frmEnquiry" method="post" action="register.aspx"
      onsubmit="return validate(this);">

  <div class="label"><label for="Name">Name:</div>
  <div class="formElement">
    <input type="text" class="required" name="Name" size="12" id="Name" />
  </div>

  <div class="label"><label for="E-mail">E-mail:</div>
  <div class="formElement">
    <input type="text" class="required" name="E-mail" size="12" id="E-mail" />
  </div>

  <div class="label"><label for="txtEmail">Please enter your query here:</div>
  <div class="formElement">
    <textarea rows="8" class="required" cols="30" name="Query" id="Query">
    </textarea>
  </div>

  <div class="label"><label for="txtEmail"> </div>
  <div class="formElement">
    <input type="submit" class="" value="Submit your query" />
  </div>

</form>
```

图 12-3 给出了用户没有为 e-mail 地址输入值时所生成的错误消息。双引号中的单词

“e-mail”从该文本输入框的 name 属性中获得。

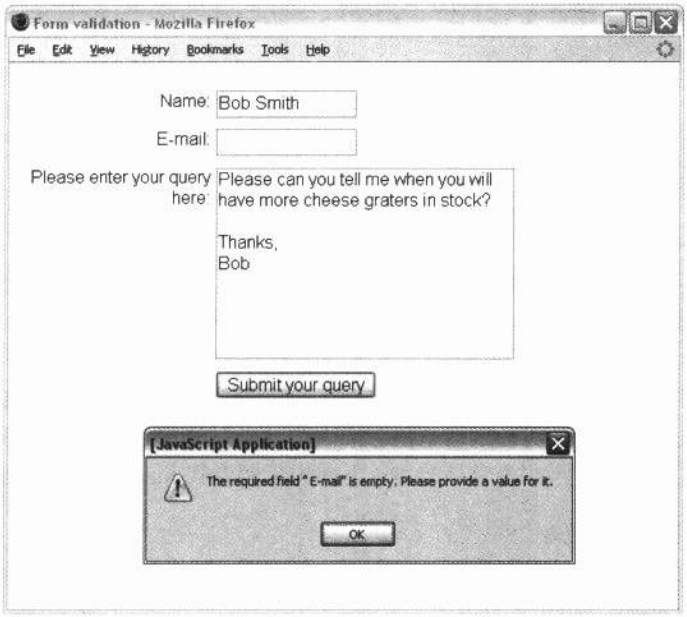


图 12-3

3. 使用 replace()方法查找字符

通常用于文本输入框的一种功能是替换特定的字符。JavaScript 具有一个非常有用的方法 replace()，可以使用它利用备选字符集替换指定的字符。

replace()方法允许指定希望替换的字符或字符集，方式是使用字符串或者正则表达式(本章稍后将介绍正则表达式)；这是该方法的第一个实参。第二个实参是希望替换为的字符。第二个实参通常只是一个替换字符串(替换字符集)，但它可以是用于确定替换字符串应当是什么的函数——如果是函数，则返回值应当用作为替换字符串。因此 replace()方法的语法可以是如下任意一种：

```
string replace(oldSubString, newSubString);
string.replace(regEx, newSubString);
string.replace(regEx, function());
```

下面的简单示例对一个文本区域使用 replace()方法，并且查找文本框中是否有字符串“URL”。找到字符串“URL”之后，该方法将其替换为字符串“ABC”(ch12_eg03.html)。首先，下面是该示例的表单：

```
<form name="myForm">
  <textarea name="myTextArea" id="myTextArea" cols="40" rows="10">I am
interested
in Curl, here is a url for it.</textarea>
  <input type="button" value="Replace characters url"
         onclick="document.myForm.myTextArea.value =
```

```
                    document.myForm.myTextArea.value.replace(/url/gi,
'abc');" />
    </form>
```

但是需要注意，这也会将单词“Curl”改为“Cabc”，因此最好在字符串“URL”的两边都添加一个“\b”以指示单词的界限——表明希望查找的是整个单词——因而仅当字符串“URL”是独立的单词时才会被替换(不能只检查字符串“URL”两边是否存在空格，因为在其周围可能存在标点符号)：

```
onclick="document.myForm.myTextArea.value=
        document.myForm.myTextArea.value.replace(/\burl\b/gi, 'abc');"
```

字符串“URL”周围的正斜杠表明函数查找的是该字符串的匹配。第二个正斜杠后面的“g” (称为标志)表明文档正在整个文本区域中查找全局匹配(如果没有 g 标志，则仅替换字符串中的第一个匹配)，i 标志表明它应当是一个不区分大小写的匹配(因此字符串“URL”也会被替换，或者实际上这些字符的大写、小写的任意混合形式都会被替换)。

也可以使用“|”符号匹配多个字符串；下面的示例查找 link、url 或 homepage 的匹配：

```
/link| url| homepage/
```

注意，如果希望搜索下面的任意字符，则它们必须被转义，因为这些字符在正则表达式中具有特殊的含义：

```
\ | ( ) [ { ^ $ * + ? .
```

如果希望转义这些字符，必须在它们的前面放置一个反斜杠(例如，/\\/匹配一个反斜杠，/\$/匹配一个美元符号)。

表 12-1 列举了一些值得注意的字符。

表 12-1

表 达 式	意　义
\n	换行
\r	回车
\t	制表符
\v	垂直制表符
\f	换页
\d	数字(等同于[0-9]，它意味着 0 到 9 之间的任意数字)
\D	非数字(等同于[^0-9]，其中^意味着非)
\w	单词(包括文字与数字)字符(等同于[a-zA-Z_0-9])
\W	非单词字符(等同于[^a-zA-Z_0-9])
\s	空格字符(等同于[\t\v\n\r\f])
\S	非空格字符(等同于[^\t\v\n\r\f])

因此，如果希望利用 HTML 的
标记替换所有的回车或换行，可以使用如下语句(ch12_eg04.html)：

```
onclick="document.myForm.myTextArea.value=
        document.myForm.myTextArea.value.replace(/\r\n| \r| \n/g),'<br />');"
```

在这个示例中，replace()方法利用\n 查找换行，利用\r 查找回车。然后替换字符串是
。图 12-4 给出了使用
标记替换回车和换行的示例显示效果(事实上，当用户提交表单之后，更有可能使用这个函数，而不是给用户提供一个按钮来执行这个操作)。

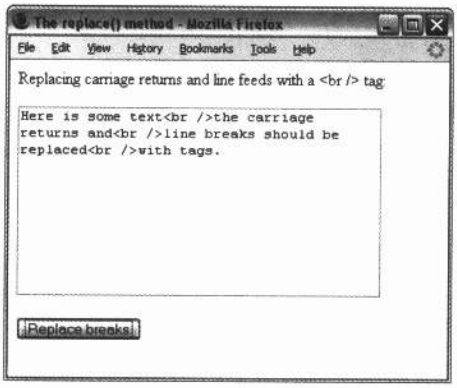


图 12-4

注意：

Netscape 浏览器和 IE 浏览器从版本 3 之后开始支持使用 replace()方法替换字符串，并且它们从版本 4 开始支持正则表达式。如果使用的是较老的浏览器，则需要使用 indexOf()方法替代它。

4. 利用测试和正则表达式测试字符

正则表达式也可以用于测试用户所输入的字符串的模式。例如，它们可以用于测试一个字符串中是否存在空格，字符串是否遵循 e-mail 地址的格式，字符串是否是一个货币值等。test()方法的用法如下：首先设置保存返回值 true 的变量、保存用户输入值的变量以及保存正则表达式的变量(ch12_eg05.html)：

```
function validate(form) {
var returnValue = true;
var amountEntered = document.frmCurrency.txtAmount.value;
var currencyFormat = /^ \d+(\.\d{1,2})?$/;
```

然后测试值是否遵循正确的格式——如果不遵循正确的格式，则提醒用户，将焦点返回到正确的表单元素，并且将变量 returnValue 设置为 false：

```
if (currencyFormat != test(amountEntered))
{
  alert("You did not enter an amount of money");
  document.frmCurrency.txtAmount.focus();
```

```
  returnValue = false;
}
return returnValue;
}
```

下面是测试这个示例的简单表单：

```
<form name="myForm" onsubmit="return validate(this);"
      action="money.aspx" method="get">
  Enter an amount of money here $
  <input type="text" name="txtAmount" id="txtAmount" size="7" />
  <input type="submit" value="Check format" />
</form>
```

图 12-5 给出了这个表单的执行效果。

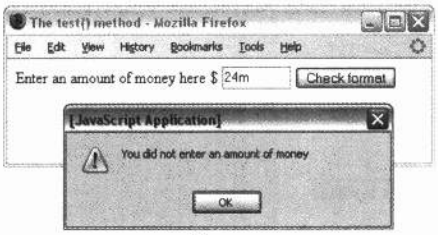


图 12-5

学习编写正则表达式并不是一件容易的事情，如果希望开始编写自己的正则表达式，应当选择一本关于如何编写它们的书籍。但是，表 12-2 中列举了一些非常有帮助的正则表达式，在开始时可以使用它们。

表 12-2

测试对象	说　明	正则表达式
空格	不存在空格字符	\S/;
字母字符	字符串中不存在字母、连字符、句点或逗号字符	/[^a-z \ -\ .']/gi;
字母数字字符	字符串中不存在字母或数字	/[^a-z0-9]/gi;
信用卡卡号	16 位数字信用卡卡号，遵循如下模式：XXXX XXXX XXXX XXXX	/^\d{4}([-]?\d{4}){3}$/;
小数数值	具有小数点的数值	/^\d+(\.\d+)?$/;
货币	一组数字(一个或多个数字)，后面可能跟上小数点加上一到两个数字	/^\d+(\.\d\{1,2})?$/;
e-mail 地址	e-mail 地址	/^\w(\.?[\w-])*@\w(\.?[\w-])*\.[a-z]{2,6}(\.[a-z]{2})?$/i;

5. 选择框选项

如果希望检查用户是否选择了选择框中的一项，需要使用代表该选择框的 select 对象的 selectedIndex 特性。如果用户选择了第一个选项，则 selectedIndex 特性将被赋予值 0；如果用户选择了第二个选项，则 selectedIndex 特性将被赋予值 1，第三个选项将被赋予值 2，依此类推。

默认情况下，当页面加载后如果用户不改变选择框控件具有的值，则标准选择框的值将是 0(当表单加载时自动选择第一个选项)；而对于多项选择框，如果没有选择选项，则值将是 1(它表明用户没有选项任何选项)。

查看下面的简单选择框，它要求用户选择一套卡片(ch12_eg06.html)：

```
<form name="frmCards" action="cards.aspx" method="get"
      onsubmit="return validate(this)">
  <select name="selCards" id="selCards">
    <option>Select a suit of cards</option>
    <option value="hearts">Hearts</option>
    <option value="diamonds">Diamonds</option>
    <option value="spades">Spades</option>
    <option value="clubs">Clubs</option>
  </select>
  <input type="submit" value="Send selection" />
</form>
```

现在检查是否选择某张卡片，这里使用 validate()函数，向该函数传递一个 form 对象作为参数。在这个示例中，如果值是 0，则必须警告用户没有选择某张卡片，并且要求其进行选择。

```
function validate(form) {
  var returnValue = true;
  var selectedOption = form.selCards.selectedIndex;
  if (selectedOption==0)
  {
    returnValue = false
    alert("Please select a suit of cards.");
  }
  return returnValue;
}
```

在图 12-6 中显示了用户没有选择一套卡片时的警告消息。

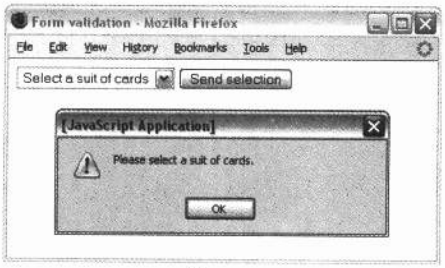


图 12-6

注意，如果希望从下拉框中收集所选择选项的值，则可以使用如下语法：

```
form.selCards.options[selected].value
```

这是因为需要查看选择哪个[option]元素以获得它的值，而不只是获得所选择元素的索引号。

6. 单选按钮

一组单选按钮与其他表单控件的不同之处在于，它们共享相同的 name 属性值，并且一次仅可以选择一个单选按钮。

如果希望确保某个单选按钮被选择，则可以预先选择一个单选按钮值，或者可以遍历 RadioButton 对象的 checked 特性以查看是否已经选择一个单选按钮。

例如，下面是具有 4 个单选按钮的表单(ch12_eg07.html)：

```
<form name="frmCards" action="cards.aspx" method="post"
      onsubmit="return validateForm(this)" >
  <p>Please select a suit of cards.</p>
  <p><input type="radio" name="radSuit" value="hearts" /> Hearts </p>
  <p><input type="radio" name="radSuit" value="diamonds" /> Diamonds </p>
  <p><input type="radio" name="radSuit" value="spades" /> Spades </p>
  <p><input type="radio" name="radSuit" value="clubs" /> Clubs </p>
  <p><input type="submit" value="Submit choice" /></p>
</form>
```

请记住，一组单选按钮将共享同一个名称，因此需要遍历集合中的每一个单选按钮，并查看是否其中一个单选按钮具有 checked 特性；为了完成该任务，需要使用一个 for 循环。该函数使用变量 radioChosen 来指明是否已经选中其中一个单选按钮。如果某个单选按钮被选中，则将它的值设置为 true。然后在遍历每个单选按钮以检查该值之后进行测试：

```
function validate(form) {
   var radioButtons = form.radSuit;
   var radioChosen = false;
   for (var i=0; i<radioButtons.length; i++) {
     if (radioButtons[i].checked)
       {
         radioChosen=true;
         returnValue=true;
       }
   }
   if (radioChosen == false) {
     returnValue = false;
     alert("You did not select a suit of cards");
   }
return returnValue;
}
```

注意，虽然在 XHTML 中元素的属性顺序无关紧要，但是在 Netscape 6 浏览器和某些版本的 Mozilla 浏览器中存在一个程序错误，即仅当<input />元素的第一个属性是 type 属

性时才能识别单选按钮的 checked 特性。

图 12-7 给出了该示例的结果。

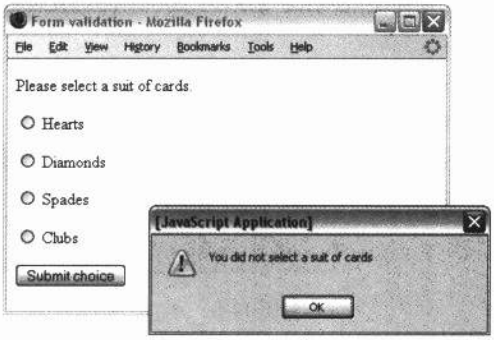


图 12-7

7. 复选框

复选框允许用户从一系列选项中选择零个、一个或多个项。虽然一组复选框能够共享相同的名称，但是与单选按钮不同，它们不是互斥的。然而，它们在 JavaScript 中作为一个数组使用，这一点与单选按钮相同。

下面的代码稍微修改前一个示例中的代码，其中使用复选框取代了单选按钮，并且用户可以选择多套卡片(ch12_eg08.html)：

```
<form name="frmCards" action="cards.aspx" method="post">
  <p>Please select one or more suits of cards.</p>
  <p><input type="checkbox" name="chkSuit" value="hearts" /> Hearts </p>
  <p><input type="checkbox" name="chkSuit" value="diamonds" /> Diamonds
</p>
  <p><input type="checkbox" name="chkSuit" value="spades" /> Spades </p>
  <p><input type="checkbox" name="chkSuit" value="clubs" /> Clubs </p>
  <p><input type="button" value="Count checkboxes"
     onclick="countCheckboxes(frmCards.chkSuit)" /></p>
</form>
```

下面的函数统计选择多少个复选框，并将该数值显示给用户。与前一个示例一样，如果没有选择任何复选框，则警告用户必须输入一个值。

```
function countCheckboxes(field) {
  var intCount = 0
  for (var i = 0; i < field.length; i++) {
     if (field[i].checked)
        intCount++; }
  alert("You selected " + intCount + " checkbox(es)");
}
```

在图 12-8 中，用户已经选择了两个复选框。

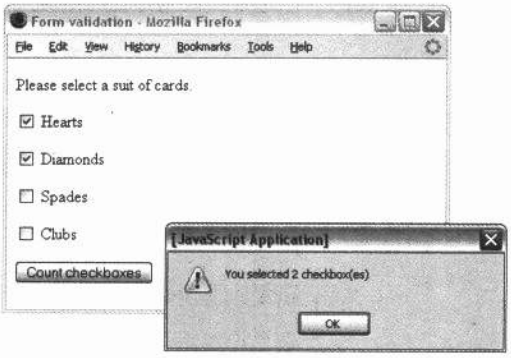


图 12-8

8. 阻止表单提交直到选择复选框

如果希望确保选择复选框——例如，如果希望用户同意某些条款和条件——则可以通过在 onsubmit 事件处理程序中添加函数(类似于前面执行的操作)来实现这一点。该函数检查是否已经选择复选框，如果函数返回 true，则提交表单。如果函数返回 false，则提示用户选择复选框。该函数如下所示(ch12_eg09.html)：

```
function checkCheckBox(myForm){
  if (myForm.agree.checked == false )
  {
    alert('You must agree to terms and conditions to continue');
    return false;
  } else
    return true;
}
```

另一个常用技术是使用脚本简单地禁用 Submit 按钮，直到用户单击复选框以表明他们同意条款和条件。

注意：

如果使用脚本重新启用一个已经禁用的表单控件，则应当在页面加载时在脚本中禁用该控件，而不是使用元素本身的 disable 属性。这对于没有在浏览器中启用 JavaScript 的用户来说是非常重要的。如果在一个<form>元素中使用 disabled 属性，并且用户的浏览器没有启用 JavaScript 功能，则他们将无法使用该表单控件。但是，如果使用脚本在页面加载时禁止控件，则可以知道当用户单击合适的复选框时脚本将能够重新启用表单控件。因此需要牢记的是，JavaScript 应当只用于增强页面的可用性，而不能作为用户使用页面的必备条件。

下面是一个具有表单的非常简单的页面。当页面加载时，Submit 按钮被 onload 事件禁用。如果用户单击 chkAgree 复选框，则重新启用 Submit 按钮(ch12_eg09.html)：

```
<body onload="document.frmAgree.btnSubmit.disabled=true">
<form name="frmAgree" action="test.aspx" method="post">
I understand that this software has no liability:
<input type="checkbox" value="0" name="chkAgree" id="chkAgree"
       onclick="document.frmAgree.btnSubmit.disabled=false" />
<input type="submit" name="btnSubmit" value="Go to download" /><br />
<p>You will not be able to submit this form unless you agree to the
   <a href="terms.html">terms and conditions</a> and check the terms and
   conditions box.</p>
</form>
</body>
```

可以在图 12-9 中看到这个示例的外观。注意，图中解释了 Submit 按钮为什么被禁用。这有助于用户理解为什么他们不能单击 Submit 按钮。

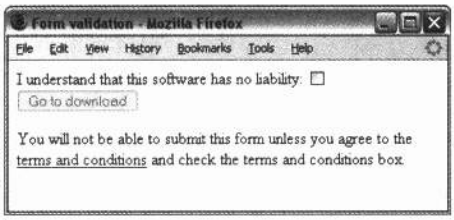


图 12-9

这种技术也可用于其他表单控件——在本章后面将给出启用文本输入框的示例。

12.3　增强表单的可用性

本节中给出的示例实际上不是帮助您验证表单；它们只是增强表单的可用性。

12.3.1　关注第一个表单项

如果表单从一个文本框开始，则可以将焦点赋予该文本框，以便用户不需要移动鼠标，单击文本输入框，然后将他们的手移回键盘才能输入文本。

为了将焦点赋予表单中的第一个文本输入框，可以简单地在文档的<body>元素中添加一个 onload 事件处理程序。这个处理程序选择希望突出显示的表单控件，并且使用该控件的 focus()方法赋予它焦点，如下所示(ch12_eg10.html)：

```
<body onload="document.myForm.myTextBox.focus();">
```

当页面加载时，光标应当在已经选择的表单控件中闪烁，为用户输入某些文本做准备，如图 12-10 所示。

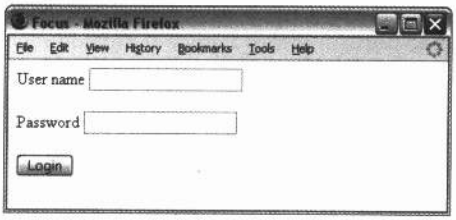


图 12-10

注意，当整个页面已经加载时，onload 事件才会激活(而不是开始加载页面时就激活)。

12.3.2 字段之间的自动焦点移动

focus()也可以用于将焦点从一个控件传递给另外一个控件。例如，如果表单中的一个控件用于提供出生日期(格式为 MM/DD/YYYY)，则只要用户输入了月，然后输入日，就可以在 3 个文本框之间移动焦点(ch12_eg11.html)：

```
<form name="frmDOB">
  Enter your date of birth:<br />
  <input name="txtMonth" id="txtMonth" size="3" maxlength="2"
       onkeyup="if(this.value.length>=2)
       this.form.txtDay.focus();"/>
  <input name="txtDay" id="txtDay" size="3" maxlength="2"
       onkeyup="if(this.value.length>=2)
       this.form.txtYear.focus();" />
  <input name="txtYear" id="txtYear" size="5" maxlength="4"
       onkeyup="if(this.value.length>=4)
       this.form.submit.focus();" />
  <input type="submit" name="submit" value="Send" />
</form>
```

这个示例使用 onkeyup 事件处理程序来检查用户已经输入的文本长度是否等于或大于该字段必须具有的字符数量。如果用户输入了必须数量的字符，则将焦点移动到下一个文本框。

注意，可以利用 this.value.length 获得文本输入框的长度。其中 this 关键字指示当前的表单控件，而 value 特性指示输入到该控件的值。

然后 length 特性返回在该表单控件中输入的值的长度。比起使用完整路径，这是确定当前表单控件中值的长度的一种较快速方式，其中完整路径的形式如下：

```
document.fromDOB.txtMonth.value.length
```

注意：

使用 this 关键字而不是完整路径的另外一个优点是，如果将这些控件复制并粘帖到不同表单中，则代码仍然能够工作，因为没有硬编码表单的名称。

图 12-11 给出了这个示例的显示效果，其中用户已经在一个字段中输入了适当数量的数字，因此将焦点移动到下一个字段。

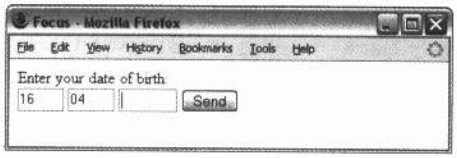


图 12-11

您可能已经注意到，size 属性的值也是一个数字，它大于字段的最大长度，以确保具有足够的空间来输入所有字符(通常控件的宽度都将稍微小一些，从而无法一次看到所有字符)。

注意：

这种技术已经用于允许用户输入他们的信用卡号，信用卡号包括 4 个分别由 4 位数字组成的数字块。虽然信用卡号的长度通常是 16 位数字，并且它们通常打印为 4 块，但是某些信用卡也可能包含 13 位数字，美国的某些运通卡则使用 15 位数字。

12.3.3　禁用文本输入框

有时您可能希望禁用某个文本输入框直到某个特定条件满足——如同图 12-9 中的 Submit 按钮被禁用直到用户单击复选框以同意某些条款和条件。

下面示例中的表单询问用户如何知道这个站点；其中单选按钮的选项包括 Friend、TV Ad、Magazine Ad 以及一个 Other 选项。如果用户选择 Other 选项，则该选项旁边的文本输入框允许用户指明如何知道这个站点。该表单如图 12-12 所示。

图 12-12

在这个示例中，当用户选择其他单选按钮时，不会启用该文本输入框；这实际上需要检查每个选中的单选按钮的值——毕竟，如果用户选择 Other 选项作为首选项，但是后面改变了主意并选择 TV 或者其他某个选项，则需要禁用该文本输入框并再次更改它的值。因此，每次用户选择一个单选按钮时，都会调用文档头部中的某个函数，该函数负责启用和禁用文本输入框控件并设置它的值。

首先，下面是为用户提供选项的表单(ch12_eg12.html)。注意，文本输入框被<body>元素的 onload 事件处理程序禁用，并且该文本输入框没有使用 disabled 属性(这与前面示例中对 Submit 按钮执行的操作相同)。

```
<body onload="document.frmReferrer.txtOther.disabled=true;
              document.frmReferrer.txtOther.value='not applicable' ">
<h2>How did you hear about us?</h2>
<form name="frmReferrer">
  <input type="radio" name="radHear" value="1"
         onclick="handleOther(this.value);" />From a friend<br />
  <input type="radio" name="radHear" value="2"
         onclick="handleOther(this.value);" />TV Ad<br />
  <input type="radio" name="radHear" value="3"
         onclick="handleOther(this.value);" />Magazine Ad<br />
  <input type="radio" name="radHear" value="4"
         onclick="handleOther(this.value);" />Newspaper Ad<br />
  <input type="radio" name="radHear" value="5"
         onclick="handleOther(this.value);" />Internet<br />
  <input type="radio" name="radHear" value="other"
         onclick="handleOther(this.value);" />Other... Please specify:
  <input type="text" name="txtOther" />
</form>
```

从这个表单中可以看到，每次用户选择表单中的一个选项时，onclick 事件调用一个 handleOther()函数，该函数将表单控件的值作为它的参数。

查看这个函数，可以看到它检查表单控件的值是否等于文本“other”(请记住，检查一个值是否等于另外一个值的方式是使用两个等号，因为单个等号用于为变量赋值)。

```
function handleOther(strRadio) {
  if (strRadio == "other") {
    document.frmReferrer.txtOther.disabled = false;
    document.frmReferrer.txtOther.value = '';
  }
  else {
    document.frmReferrer.txtOther.disabled = true;
    document.frmReferrer.txtOther.value = 'not applicable';
  }
}
```

此处使用一条简单的 if...else 语句来查看单选按钮的值，其中单选按钮作为参数传入。如果值是 other，则启用控件，并且将值设置为空——否则禁用该控件，并且值是“not applicable”。

12.3.4 转换大小写

有时需要改变用户输入的文本的大小写，使其都是大写或都是小写——特别是因为 JavaScript 区分大小写。为了改变文本的大小写，可以使用 JavaScript 的 String 对象的两种内置方法：

- toLowerCase()
- toUpperCase()

为了演示这些方法，下面的示例中具有一个文本输入框，当从文本输入框上移走焦点时，将改变文本的大小写 (ch12_eg13.html)：

```
<form>
  <input type="text" name="case" size="20"
         onblur="this.value=this.value.toLowerCase();" />
</form>
```

注意：

如果将表单数据发送给服务器，则最好将大小写改变操作放在服务器端完成，因为这样不会使用户感到困惑——如果改变表单中字母的大小写，则对于使用该表单的用户来说会觉得有点奇怪。

12.3.5　剪裁字段开头和结尾的空格

因为多种原因，您可能希望删除表单中某个字段开头或结尾的空格，甚至只是因为用户无意中在这些位置中输入空格。此处即将演示的技术是使用 String 对象的 substring()方法，它的语法如下所示：

```
substring(startPosition, endPosition)
```

该方法返回指定点的字符串——如果没有指定结束位置，则默认是字符串的结尾。起点位置和结束点位置都基于 0，因此第一个字符是 0。例如，如果某个字符串是“Welcome”，则方法 substring(0, 1)将返回字母 W。

首先查看删除字符串起始部分的前导空格，此时 substring()方法将被调用两次。

首先使用 substring()方法获取用户输入到文本控件中的值并只返回第一个字符，然后检查返回的第一个字符是否是空格：

```
this.value.substring(0,1) == ' '
```

如果该字符是空格，则第二次调用 substring()方法删除该空格。这一次该方法选择字符串中从第二个字符开始到最后一个字符结束的所有字符(即忽略第一个字符)。然后将选择的字符串作为表单控件的新值，从而删除了第一个空格字符。

```
this.value = this.value.substring(1, this.value.length);
```

检查第一个字符是否为空格，如果是，则删除该字符，由 onblur 事件处理程序调用上述的整个过程；因此从表单控件中移走焦点时，将开始执行该过程。该过程使用了一个 while 循环来表明只要字符串的第一个字符是空格，则将第二次调用 substring()方法删除它。该循环确保只要第一个字符是空格就删除它，直到子字符串的第一个字符不是空格为止 (ch12_eg14.html)：

```
<form>
  <input type="text" name="txtName" size="100"
         value=" Enter text leaving whitespace at start. Then change focus."
         onblur="while (this.value.substring(0,1) == ' ')
```

```
            this.value = this.value.substring(1, this.value.length);" /><br />
</form>
```

为了剪裁结尾的空格，采用相似的过程，但是反转执行该过程。第一个 substring()方法收集字符串的最后一个字符，如果该字符是空格，则删除它，如下所示：

```
<form>
<input type="text" name="txtName" size="100"
       value="Enter text leaving whitespace at end. Then change focus. "
       onblur="while (this.value.substring
         (this.value.length-1,this.value.length) == ' ')
         this.value = this.value.substring(0, this.value.length-1);" /><br />
</form>
```

只要用户的浏览器不是 Netscape 4 和 IE4 之前版本的浏览器，则可以使用正则表达式剪裁空格，如下所示：

```
<form>
  <input type="text" name="removeLeadingAndTrailingSpace" size="100"
    value=" Enter text with white space, then change focus. "
    onblur = "this.value = this.value.replace(/^\ \s+/, ").replace(/\s+$/, ");"
  /><br />
</form>
```

这将同时删除开头和结尾的空格。

正则表达式本身是非常庞大的主题。如果希望学习关于正则表达式的更多知识，可以参考 Paul Wilton 编写的书籍《Beginning JavaScript 2nd Edition》(Wrox, 2000)。

12.3.6 选择文本区域中的所有内容

如果希望允许用户选择文本区域中的全部内容(从而他们不需要利用鼠标手动选择所有文本)，则可以使用 focus()方法和 select()方法。

在下面的示例中，selectAll()函数采用一个参数，即希望选择其中内容的表单控件(ch12_eg15.html)：

```
<html>
<head><title>Select whole text area</title>
<script language="JavaScript">
  function selectAll(strControl) {
    strControl.focus();
    strControl.select();
  }
</script>
</head>
<body>
  <form name="myForm">
    <textarea name="myTextArea" rows="5" cols="20">This is some text</textarea>
    <input type="button" name="btnSelectAll" value="Select all"
           onclick="selectAll(document.myForm.myTextArea);" />
```

```
  </form>
</body>
</head>
</html>
```

允许用户选择所有内容的按钮具有 onclick 事件处理程序，用于调用 selectAll()函数，并告诉该函数应该选择哪个控件的内容。

selectAll()函数首先利用 focus()方法赋予该控件焦点，然后使用 select()方法选择该控件的内容。在能够选择其中的内容之前，表单控件必须首先获得焦点。相同的方法也可以应用于单行文本输入框和密码字段。

12.3.7　选中或取消选中所有复选框

如果在一组复选框中存在多个复选框，则允许用户一次选择或取消选择整个组中的所有复选框非常有用。使用下面的两个函数能够完成该任务：

```
function check(field) {
  for (var i = 0; i < field.length; i++) {
    field[i].checked = true;}
}
function uncheck(field) {
  for (var i = 0; i < field.length; i++) {
    field[i].checked = false; }
}
```

为了使这两个函数起作用，组中必须有多个复选框。然后添加两个按钮调用这两个函数，将共享相同名称的复选框元素的数组作为函数的参数，如下所示(ch12_eg16.html)：

```
<form name="frmSnacks" action="">
  Your basket order<br />
  <input type="checkbox" name="basketItem" value="1" />Chocolate
  cookies<br />
  <input type="checkbox" name="basketItem" value="2" />Potato chips<br />
  <input type="checkbox" name="basketItem" value="3" />Cola<br />
  <input type="checkbox" name="basketItem" value="4" />Cheese<br />
  <input type="checkbox" name="basketItem" value="5" />Candy bar<br /><br />
  <input type="button" value="Select All"
         onclick="check(document.frmSnacks.basketItem);" />

  <input type="button" value="Deselect All"
         onclick="uncheck(document.frmSnacks.basketItem);" />
</form>
```

图 12-13 给出了这个表单的外观。

也可以将这些功能组合到一个函数中，并且通过相同的按钮调用该函数，如下所示：

```
function checkUncheckAll(field) {
  var theForm = field.form, z = 0;
  for(z=0; z<theForm.length;z++){
```

```
    if(theForm[z].type == 'checkbox' && theForm[z].name != 'checkall'){
    theForm[z].checked = field.checked;
  }
  }
}
```

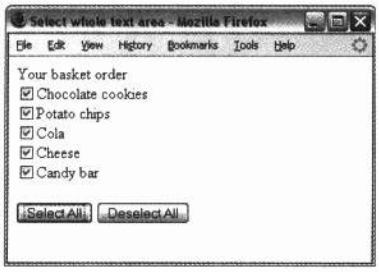


图 12-13

试一试 e-mail 表单

在这个练习中将创建一个 e-mail 表单，该表单具有一些有趣的功能。它使用一个正则表达式检查 e-mail 地址的结构，并且检查具有某种类型条目的所有字段。该表单包括一个快速地址簿，地址簿中包含潜在 e-mail 接收方的地址。图 12-14 给出该表单的外观；图中也给出了用户没有输入任何消息并提交 e-mail 时显示的消息。

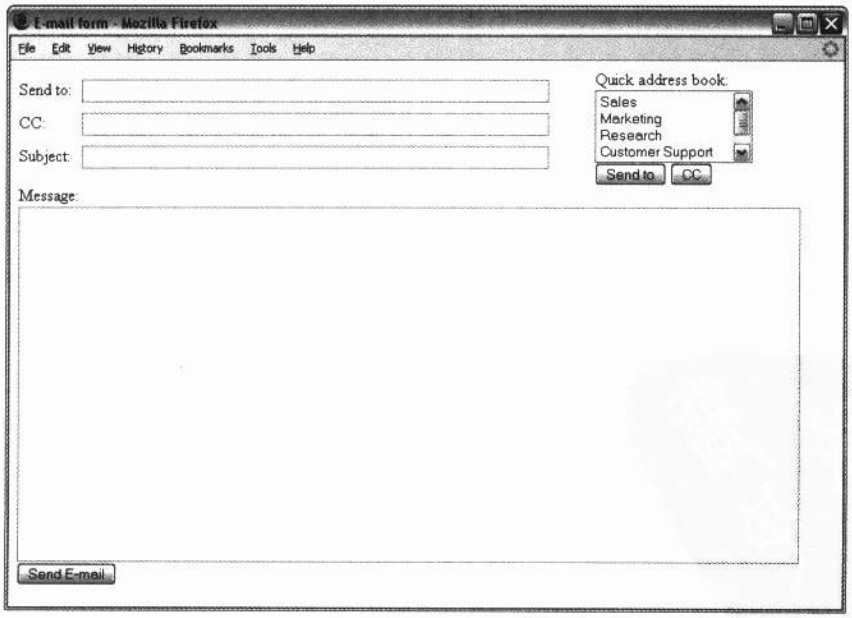


图 12-14

(1) 创建 XHTML 文档的程序框架，它具有<head>、<title>和<body>元素。

(2) 在文档主体中，添加一个<form>元素和两个 元素。第一个 元素保存 To、

CC 和 Subject 字段，而第二个 元素保存快速地址。

```
    <form name="frmEmail" onsubmit="return validate(this)"
action="sendMail.aspx"
          method ="post">
      <div id="toCCsubject">
        <div class="label">Send to:</div>
        <div class="input"><input type="text" size="70" name="txtTo" /></div>

        <div class="label">CC:</div>
        <div class="input"><input type="text" size="70" name="txtCC" /></div>

        <div class="label">Subject:</div>
        <div class="input"><input type="text" size="70" name=
            "txtSubject" /></div>

      </div>
      <div id="addressBook">
        <!-- quick address book will go here --></td>
      </div>
```

(3) 接下来需要将快速地址薄添加到第二个 元素中，该地址簿使用了一个多项选择框。在地址簿的下方是两个按钮：一个用于将地址添加到 txtTo 字段中，另一个用于将地址添加到 txtCC 字段中。当单击时，这两个按钮都调用 add()函数：

```
Quick address book:<br />
<select size="4" name="selectList1" style="width:150px">
  <option value="sales@example.org">Sales</option>
  <option value="marketing@example.org">Marketing</option>
  <option value="research@example.org">Research</option>
  <option value="support@example.org">Customer Support</option>
  <option value="it@example.org">IT</option>
</select><br />
<input type="button" onclick="add(textTo, document.frmEmail.selectList1);"
       value="Send to" />
<input type="button" onclick="add(textCC, document.frmEmail.selectList1);"
       value="CC" />
```

(4) 在表单中添加消息元素<textarea>和一个 Send E-mail 按钮：

```
Message:<br />
<textarea name="message" rows="20" cols="115"></textarea><br />
<input type="submit" value="Send E-mail" />
```

(5) 现在需要添加验证函数和 add()函数。首先，下面是 add()函数，它将 e-mail 地址从地址簿中添加到 To 字段或 CC 字段(如果字段中已经存在地址，则添加分号以隔开多个地址)。

```
function add(objInput, objList){\{}
var strGroup = objList.options[objList.selectedIndex].value;
```

```
  if (objInput.value == "")
  {
    objInput.value = strGroup
  }
  else
  {
    objInput.value += ('; ' + strGroup)
  }
}
```

(6) 下面是 validate()函数，该函数非常长：

```
function validate(form) {
  var returnValue = true;
  var sendTo = form.txtTo.value;
  var cc = form.txtCC.value;
  var subject = form.txtSubject.value;
  var message = form.txtMessage.value;
  if (sendTo == "")
  {
    returnValue = false;
    alert("There are no email addresses in the To field");
    form.txtTo.focus();
  }
  if (subject == "")
  {
    returnValue = false;
    alert("There is no subject line for this e-mail");
    form.txtSubject.focus();
  }
  if (message=="")
  {
    returnValue = false;
    alert("There is no message to this e-mail");
    form.txtMessage.focus();
  }
  var arrTo = sendTo.split("; ");
  var rxEmail=/\^\w(\.?[\w-])*@\w(\.?[\w-])*\.[a-z]{2,6}(\.[a-z]{2})?$/i;
  for (var i=0; i<(arrTo.length); i++) {
    if (!rxEmail.test(arrTo[i]))
    {
      returnValue = false;
      alert("The e-mail address "+ arrTo[i] +" does not appear to be valid");
    }
  }
  var arrCC = cc.split("; ");
  for (var i=0; i<(arrCC.length); i++) {
    if (!rxEmail.test(arrCC[i]))
    {
      returnValue = false;
      alert("The e-mail address "+ arrCC[i] +" does not appear to be valid");
```

```
    }
  }
return returnValue;
}
```

(7) 将文件保存为 emailform.html，当在浏览器窗口中打开它时，结果将与图 12-14 中所示的示例相似。

工作原理

这个示例中的表单包含两个函数。第一个是 add()函数，它将 e-mail 地址从选择框传递到 To 字段或 CC 字段。add()函数非常简单，它采用两个参数：

- objInput：一个字段，已经选择的地址将被发送到该字段
- objList：一个选择列表，包含 e-mail 地址

这个函数首先使用选择列表的 Selected Index 特性收集已经选择的项的值，并将该值放置在称为 strGroup 的变量中。接下来，该函数检查即将加入地址的表单字段是否为空；如果为空，则将 strGroup 属性中存储的 e-mail 地址添加到该字段。如果 To 字段或 CC 字段不为空，则在 e-mail 地址之前添加一个分号和一个空格，因为它们通常用作多个 e-mail 地址之间的分隔符：

```
function add(objInput, objList){
var strGroup = objList.options[objList.selectedIndex].value;
  if (objInput.value == "")
  {
    objInput.value = strGroup
  }
  else
  {
    objInput.value += ('; ' + strGroup)
  }
}
```

validate()函数稍微复杂一点，它首先将 returnValue 变量设置为 true，并将表单的值收集到多个变量中。

```
function validate(form) {
  var returnValue = true;
  var sendTo = form.txtTo.value;
  var cc = form.txtCC.value;
  var subject = form.txtSubject.value;
  var message = form.txtMessage.value;
```

该函数检查 To、Subject line 和 Message body 字段是否为空，如果为空，则将 returnValue 变量设置为 false，并使用一个警告框指示用户必须在该字段中填写内容——这与本章前面给出的某些示例类似：

```
if (sendTo == "")
{
```

```
    returnValue = false;
    alert("There are no e-mail addresses in the To field");
    form.txtTo.focus();
  }
```

当 validate 函数开始检查输入在表单中的 e-mail 地址是否是有效时，该函数变得更加油趣。首先，用于检查 e-mail 地址的正则表达式需要存储到一个变量中——该变量称为 rxEmail：

```
var rxEmail=/\^\w(\.?[\w-])*@\w(\.?[\w-])*\.[a-z]{2,6}(\.[a-z]{2})?$/i;
```

接下来，使用 String 对象的 split()方法将 To 字段的内容拆分到一个数组中。该函数采用一个字符串作为参数，并在遇到指定字符或字符集时将该字符拆分为多个独立的值。在这个示例中，该方法查找任何后面跟上空格的分号的实例，每当找到这样的实例时，则在数组中创建一个新项。

```
var arrTo = sendTo.split("; ");
```

假设具有下面的 e-mail 地址(这些地址只是用于演示 split()方法；它们不是代码的一部分)：

```
sales@example.com; accounts@example.com; marketing@example.com
```

这些 e-mail 地址将被拆分为下面的数组(再次提醒，这不是示例代码的一部分)：

```
arrTo[0] = "sales@example.com"
arrTo[1] = "accounts@example.com"
arrTo[2] = "marketing@example.com"
```

因此，现在代码中必须具有一个 for 循环，用于遍历数组中的每一个 e-mail 地址，并且检查 e-mail 地址是否遵循正则表达式所描述的模式。该 for 循环具有 3 个参数：第一个参数将计数器 i 设置为 0，第二个参数检查该计数器是否小于数组中的项的数量，第三个参数将计数器递增 1。在循环内部是一条 if 语句，它使用 test()方法检查 e-mail 地址是否匹配正则表达式；如果不匹配，则将 returnValue 变量设置为 false，并且警告用户该值看上去不像有效的 e-mail 地址：

```
for (var i=0; i<(arrTo.length); i++) {
  if (!rxEmail.test(arrTo[i]))
  {
    returnValue = false;
    alert("The email address "+ arrTo[i] +" does not appear to be valid");
  }
}
```

接下来是用于 CC 字段的类似设置过程。

```
var arrCC = cc.split("; ");
for (var i=0; i<(arrCC.length); i++) {
   if (!rxEmail.test(arrCC[i]))
   {
     returnValue = false;
```

```
        alert("The e-mail address "+ arrCC[i] +" does not appear to be valid");
      }
    }
    return returnValue;
  }
```

现在就有了一个具有多个函数的表单示例。该示例使用 JavaScript 创建了一个快速地址薄，并且验证每一个条目，以防止用户尝试发送无效的 e-mail 地址。

12.4　图像翻转

在上一章中给出了一个关于图像翻转的简单示例，但是在本章中将给出一个函数，使用该函数可以在同一个页面中改变多幅图像。这个函数可以用于所有页面，而不是在多个页面中重复相同的脚本。

为了创建翻转图像，需要图像的两个不同版本：

- 普通图像，当用户的鼠标没有悬停在图像上时显示该图像。
- 另一幅图像，当用户将鼠标悬停在图像上时显示该图像。

上一章中介绍了一个非常简单的图像翻转脚本，该脚本添加在包含图像的<a>元素中。当用户将鼠标悬停在链接上(包含图像的链接)时 onmouseover 事件激活，并且图像对象的 src 特性被改为翻转的图像。当鼠标离开图像时，onmouseout 事件将图像的 src 特性改回初始图像(如果仅监控其中一个事件，则图像将只会改变，而不会返回到初始状态，因此同时监视两个事件非常重要)。

该图像的 name 属性具有值 button，它用于在事件处理程序中标识该图像：

```
<a href=""
  onmouseover="document.images.button.src='click_red.gif';"
  onmouseout="document.images.button.src='click_green.gif'">
  <img src="click_green.gif" width="100" height="50" border="0"
name="button" />
</a>
```

记住，文档中的每幅图像在 DOM 中具有自己对应的对象，并且 image 对象的一个特性是 src 特性。src 特性是图像文件的位置，它对应于文档中<img>元素的 src 属性指定的值。

注意：

当创建包含文本的翻转图像时，通常应当在两幅图像中使用具有相同大小和量级的文本。突然出现较大或较粗的文本将难以阅读。稍微改变背景色将是更好的选择。

当希望在多个页面中使用相同的图像翻转时，下一个逻辑步骤就是创建一个图像翻转函数——例如，创建一个导航栏，并且该导航栏在用户将鼠标移动到每一项上时改变颜色。图 12-15 中的导航栏具有该功能。

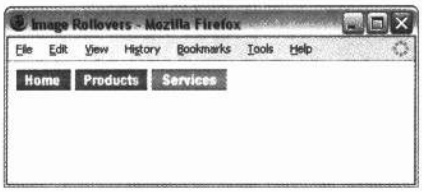


图 12-15

这个导航栏的每幅图像都包含在一个链接中，并且每幅图像必须具有不同的名称。与上一个示例一样，附带事件处理程序的元素是<a>。当用户将鼠标放置在链接上时，onmouseover 事件调用 changeImages()函数，当鼠标从链接上移开时，onmouseout 事件调用相同的函数，但是传递的值表明应当再次显示初始图像。

changeImages()函数具有两个实参——第一个实参是图像的名称，第二个实参是一个变量名，它保存图像的 URL，该图像将替换当前图像。注意，当 onmouseover 事件和 onmouseout 事件激活时，图像的 name 属性的值对应于传递的参数(ch12_eg17.html)：

```
<a href="index.html"
   onmouseover="changeImages('image1', 'image1on')"
   onmouseout="changeImages('image1', 'image1off')">
   <img name="image1" src="images/home.gif" width="99" height="20"
      border="0" alt="home">
</a>
```

执行实际工作的脚本位于 scripts 文件夹中称为 rollover.js 的文件中。该脚本可以包含在任何需要翻转图像的页面中。

请记住，每一个翻转具有两幅图像——当鼠标位于图像上方时，它处于“开”状态；当鼠标离开图像时，它处于“关”状态。

每幅图像被分配两个变量，一个用于鼠标位于图像上时，另一个用于鼠标离开图像时。这两个变量保存图像对象，该图像对象的 src 特性是图像的 URL。首先将看到用于翻转时的图像，然后看到用于普通状态时的图像：

```
if (document.images) {
  image1on = new Image();
  image1on.src = "images/nav_home_on.gif";
  image2on = new Image();
  image2on.src = "images/nav_products_on.gif";
  image3on = new Image();
  image3on.src = "images/nav_services_on.gif";
```

接下来是一些保存图像对象的变量，这些图像对象的 src 特性针对图像处于“关”状态时的情形进行设置。

```
  image1off = new Image();
  image1off.src = "images/nav_home.gif";
  image2off = new Image();
  image2off.src = "images/nav_products.gif";
  image3off = new Image();
```

```
  image3off.src = "images/nav_services.gif";
}
```

下面是相应的函数，它遍历图像，并且采用传递给该函数的实参：

```
function changeImages() {
  if (document.images) {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
      document[changeImages.arguments[i]].src =
        eval(changeImages.arguments[i+1] + ".src");
    }
  }
}
```

完成实际工作的代码行位于中间。如果用户将鼠标移动到第一幅图像上，采用如下方式调用函数：

```
onsubmit="changeImages(image1, image1on)"
```

传入的第一个值是图像 name 特性的值。因此，函数的如下代码行告诉浏览器采用 changeImages()函数的第一个实参(即 image1)并改变这个元素的 src 特性：

```
document[changeImages.arguments[i]].src =
```

该行的最后一个字符是等号(=)。这个特性仍然需要设置，并且下一行代码是实际提供值的代码。下一行代码表明该特性应该被赋予函数中第二个实参的值：

```
eval(changeImages.arguments[i+1] + ".src");
```

上一章中介绍过，for 循环采用如下 3 个实参：

- 第一个实参仅运行一次，在这个示例中它将计数器的值设置为 0(i-0)。
- 第二个实参指明循环是否应当再次运行。在这个示例中，如果计数器的值小于传递给 changeImages()函数的实参的数量，循环将再次运行。
- 第三个实参将计数器递增 2。

这意味着 changeImages()函数可用于改变多幅图像，因为可以利用不同的参数集调用该函数。

12.5　随机脚本生成器

使用一个脚本选择随机值有时会非常有用。下面的脚本可用于从一个预定义的数组中随机选择一段内容。可以使用该脚本添加随机的引用或提示，或者可以使用它循环播放广告或图像。该脚本包含称为 randomContent()的函数，其中包括用于随机选择的内容。

内容添加到称为 arrContent 的数组中，该数组包含希望随机显示的数据：

```
<script language="JavaScript">
function randomContent(){
  var arrContent=new Array()
```

```
arrContent[0]='This is the first message.'
arrContent[1]='This is the second message.'
arrContent[2]='This is the third message.'
arrContent[3]='This is the fourth message.'
arrContent[4]='This is the fifth message.'
```

然后将称为 i 的变量设置为一个从 0 到数组中项数之间的随机值。为了生成这个随机数值，需要调用 Math 对象中的两个方法。random()方法生成一个 0 到 1 之间的随机值，将该值乘以数组中元素的数量。然后将生成的数值舍入为等于或小于所生成数值的最接近的整数，方式是使用 floor()方法。

这里使用 floor()方法而不是 round()方法的原因是，如果使用 round()方法，则最终获得的数值将大于数组中项的数量。

```
var i=Math.floor(Math.random()*arrContent.length)
  document.write(arrContent[i])
}
</script>
```

在希望包含随机内容的位置，可以只调用该函数：

```
<script type="text/JavaScript">
  randomContent();
</script>
```

图 12-16 中给出了该示例的结果。

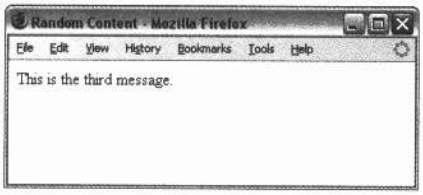


图 12-16

如果希望随机内容出现在多个页面上，可以简单地将该函数放置在一个外部文件中。

12.6 弹出式窗口

弹出式窗口具有不光彩的名声，人们通常将它们与弹出式广告关联在一起，当站点的页面加载时就会显示弹出式广告，并且它们通常用于提供广告或不受欢迎的信息。但是，弹出式窗口也有一些非常合法的用途。例如，您可能希望用户停留在当前的页面，同时允许他们在弹出式窗口中提供一些其他信息，或者可能希望从站点中打开一些内容(例如一幅图像)，并且新打开的内容位于一个新窗口中，从而用户不会丢失位置。

当然，可以创建一个普通链接，并使页面在一个新的窗口中打开，方式是添加 target="_new" 属性。但是，当使用 JavaScript 创建弹出式窗口时，可以控制窗口的尺寸，指明它的大小是否能够调整，以及它是否具有滚动栏(ch12_eg19.html)。

```
<a
  href="http://google.com/"
  onclick="window.open(this.href, 'search',
  'width=400,height=300,scrollbars,resizable');
  return false;"
>
Click here for the link to open in a popup window.
</a>
```

可以看到 window 对象的 open()方法能够采用多个参数；语法如下所示：

```
open(url, 'windowname', 'features')
```

可以在窗口名的后面列举多个功能，表 12-3 中给出了能够列举的功能。这些功能可用于控制窗口的多种特性，包括窗口的大小和位置以及屏幕中是否具有滚动栏——但是请记住，具有不同分辨率的用户可能要求具有滚动栏。

表 12-3

功　能	值	设　置
width	数值	新窗口的宽度，单位为像素
height	数值	新窗口的高度，单位为像素
left	数值	窗口的左边将出现的位置
top	数值	窗口的顶部将出现的位置
location	yes/no	控制浏览器是否应当显示浏览器位置工具栏
menubar	yes/no	控制浏览器是否应当显示菜单栏
resizable	yes/no	允许用户调整浏览器窗口的大小
scrollbars	yes/no	控制是否显示水平或垂直滚动栏
status	yes/no	控制状态栏(浏览器底部的区域)的显示
toolbar	yes/no	控制浏览器是否应当显示按钮工具栏

应当注意的是，某些弹出式窗口阻塞软件可能会阻止类似于这样的功能起作用。在创建弹出式窗口时，应当避免在文件名中使用类似于“pop-up”(或“popup”)的单词，因为某些弹出式窗口阻塞软件会在文件名中查找这些单词，并且不会打开包含它们的文件。

注意：

在 JavaScript 中，可以采用多种方式创建弹出式窗口，但是如果选择利用 JavaScript 创建它们，则强烈推荐使用此处介绍的方法，因为许多其他方法会阻止用户右击链接并在新的窗口中打开它。更有经验的 Web 浏览器开发人员允许用户利用鼠标右键单击在新窗口中打开一个链接，并且某些创建弹出式窗口的方法意味着采用该方式(选择在新窗口中打开连接)打开链接的用户将仅获得一个空白窗口。而上面介绍的方法能够解决这个问题。

12.7 JavaScript 库

本章中到目前为止已经介绍的示例是为了帮助您能更好地理解 JavaScript 如何工作于 XHTML 文档中。接下来介绍的示例将采用一些流行的免费 JavaScript 库，可以通过 Web 免费下载它们。

JavaScript 库是简单的 JavaScript 文件，它们包含的代码有助于程序员执行一些经常用于 web 页面中的任务，并且只需要编写少量代码。接下来介绍的示例允许您执行以下操作：

- 创建动画效果，例如淡出式文本或收缩框
- 重新排序项目列表中的项
- 排序表
- 创建日历
- 自动完成文本字段

存在多种 JavaScript 库，可以从 Web 上下载它们；但是，在本章中将主要介绍 Scriptaculous (它实际上构建于另一个称为 Prototype 的 JavaScript 库之上)、MochiKit 和 Yahoo User Interface(雅虎用户界面，也称为 YUI)3 个库。

本章的下载代码中包含了这些库的多种版本。如果查看第 12 章的代码文件夹，可以看到在脚本文件夹内存在分别称为 scriptaculous、mochikit 和 yui 的文件夹(每个文件夹分别对应于前面 3 个库)。

12.7.1 利用 Scriptaculous 库创建动画效果

Scriptaculous 库能够帮助您完成许多任务：动画制作、拖放功能、编辑工具和文本输入框的自动完成，以及用于帮助创建 DOM 片段的实用程序。在本节中将介绍一些动画效果。

本章前面已经提到，Scriptaculous 库实际上构建于另一个称为 Prototype 的 JavaScript 库之上。在本章的下载代码中包含了 Scriptaculous 1.8.0 和 Prototype 1.6.0；但是，可以从 http://script.aculo.us/处查找一些更新的版本并下载这些文件的剧本。

Scriptaculous 库包含的函数能够用于创建多种不同类型的动画。下面的示例将只演示利用 Scriptaculous 可以实现的 4 种动画效果，但这对于演示这些效果的灵活性以及如何方便地将它们集成到页面中来说已经足够。图 12-17 中给出了这个页面的外观，但是实际上需要试验这个示例以查看动画效果，该示例的文件为 ch12_eg20.html。

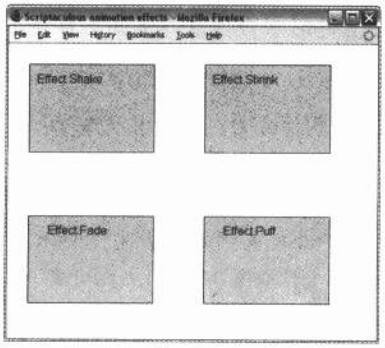


图 12-17

为了使用 Scriptaculous 库，需要创建到 prototype.js 库(该库位于 scriptaculous 文件夹的 lib 文件夹中)和 scriptaculous.js 库(该库位于 scriptaculous 文件夹的 src 文件夹中)的引用(如果查看 src 文件夹，将会发现该 JavaScript 文件需要加载的其他几个脚本)。创建引用的方式如下所示：

```
<script src="scripts/scriptaculous/lib/prototype.js"
  type="text/javascript"></script>
<script src="scripts/scriptaculous/src/scriptaculous.js"
  type="text/javascript"></script>
```

然后是 4 个 元素，它们分别用于演示 4 种动画效果。最外部的 元素用于将 4 种效果隔开；内部的第二个 元素是正在演示的函数的名称：

```
<div class="container">
  <div class="demo">
    Effect.Fade
  </div>
</div>
```

需要查看这些元素的第二行。class 属性值为 demo 的 元素创建了一个框，在该元素内添加用于在脚本内标识这个元素的 id 属性，而 onclick 属性调用 Scriptaculous 库来创建该效果：

```
<div class="container">
  <div class="demo" id="demo-effect-shake" onclick="new Effect.Shake(this)">
    Effect.Shake
  </div>
</div>
<div class="container">
  <div class="demo" id="demo-effect-shrink" onclick="new Effect.Shrink(this);
    window.setTimeout('Effect.Appear(\'demo-effect-shrink\',
    {duration:.3})',2500);">
    Effect.Shrink
  </div>
</div>
<div class="clear"></div>
<div class="container">
  <div class="demo" id="demo-effect-fade" onclick="new Effect.Fade(this);
    window.setTimeout('Effect.Appear(\'demo-effect-fade\',
{duration:.3})',2500);">
    Effect.Fade
  </div>
</div>
<div class="container">
  <div class="demo" id="demo-effect-puff" onclick="new Effect.Puff(this);
    window.setTimeout('Effect.Appear(\'demo-effect-puff\',
{duration:.3})',2500);">
    Effect.Puff
  </div>
</div>
```

不需要知道该脚本如何创建这些效果；而只需要知道利用创建的 Effect 对象(创建的方式是使用 new Effect)访问该效果，以及调用每一种效果的方法的语法。

首先查看第一个框，使用一种振动效果将其左右移动：

```
    <div class="container">
      <div class="demo" id="demo-effect-shake" onclick="new
Effect.Shake(this)">
        Effect.Shake
      </div>
    </div>
```

对于这个元素，需要执行的操作仅是添加 onclick 属性，此处创建一个新的 Effect 对象，并且调用它的 Shake()方法。在利用<form>元素的 onsubmit 特性验证表单时已经看到，可以利用 this 关键字告诉方法正在传递的是当前元素(和任何内容)。因此，这个示例中的 onclick 属性只是告诉 Scriptaculous 库创建一个新的 Effect 对象，以便在单击元素时振动该元素。

您可能已经注意到，下面 3 个元素在调用效果后包含了第二行代码。这是因为这 3 种效果使框消失。因此在固定的时间间隔之后调用 Appear()方法，以便可以再次试验该示例(Appear()方法使用了 id 属性的值指明哪一个元素需要重新显示)；但是，其他效果仍然使用 Effect.methodname(this)方式调用。

```
    <div class="container">
      <div class="demo" id="demo-effect-shrink" onclick="new
Effect.Shrink(this);
        window.setTimeout('Effect.Appear(\'demo-effect-shrink\',
    {duration:.3})',2500);">
        Effect.Shrink
      </div>
    </div>
```

这是利用 JavaScript 创建动画效果的非常简单的方式。

12.7.2 利用 Scriptaculous 库拖放可排序列表

利用 Scriptaculous 库的第二项任务是创建拖放列表。您可能已经在某些站点中看到，可以重新排序列表(例如备忘录列表或前 10 名列表)，方式是拖放相应的元素。

在图 12-18 中可以看到本节即将创建的示例；当页面加载时，框按照数值顺序排序。但是，它们现在已经被拖放为不同的顺序。

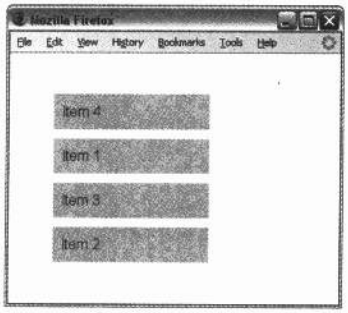


图 12-18

在这个示例(ch12_eg21.html)中，需要再次包含 Scriptaculous 库和 Prototype 库。然后就会拥有一个简单的无序列表(在文档的头部中存在一些 CSS 规则，它们控制列表的外观，以使其看上去类似于框)。

```
<script src="scripts/prototype.js" type="text/javascript"></script>
<script src="scripts/scriptaculous.js" type="text/javascript"></script>
<style type="text/css">
  li {border;1px solid #000000; padding:10px; margin-top:10px;
  font-family:arial, verdana, sans-serif;background-color:#d6d6d6;
  list-style-type:none; width:150px;}
</style>
</head>
<body>
<ul id="items_list">
  <li id="item_1">Item 1</li>
  <li id="item_2">Item 2</li>
  <li id="item_3">Item 3</li>
  <li id="item_4">Item 4</li>
</ul>
```

然后只需要在列表后添加一个<script>元素，以便能够重新排序列表：

```
<script type="text/javascript" language="javascript">
   Sortable.create("items_list",{ dropOnEmpty:true,constraint:false});
</script>
```

这里使用 Sortable 对象的 create()方法创建一个 Sortable 对象。该方法采用如下两个实参：

- 第一个实参是无序列表元素的 id 属性的值。
- 第二个实参是一些选项，它们描述了可排序列表的工作方式。第一个选项是 dropOnEmpty，它的值是 true，表明该元素应当仅在元素之间拖放，而不是拖放到另一个元素的上方；第二个选项是 constraint 特性，它被设置为 false(如果该项没有设置或者设置为 true，则将仅允许项沿着垂直坐标轴移动)。

为了使这种拖放列表能够被人们使用，这种列表通常与某些用于更新数据库的代码关联，例如 ASP.Net、PHP、JSP 或 Ruby on Rails 代码。然而，这个示例演示了通过 Scriptaculous 库可以仅使用少量代码就能够非常容易地实现一些效果。

12.7.3　利用 MochiKit 库创建可排序表

在这个示例中将演示另一个 JavaScript 库——MochiKit 库。通过查看几个不同的示例，您会发现将不同的库(它们提供不同的功能)插入到页面中非常容易实现。可以从 www.mochikit.com/处下载 MochiKit 库的最新版本，但是已经在本章的下载代码中包含了 1.3.1 版本的 MochiKit 库。

在下面的示例中将创建一个表，在其中可以排序表的内容，方式是单击表中任何列的题头以根据该列排序数据。在图 12-19 中可以看到表题头 Date Started 旁边具有一个向上的箭头，表明该表的内容按照员工的开始工作日期排序(按照降序排序)。

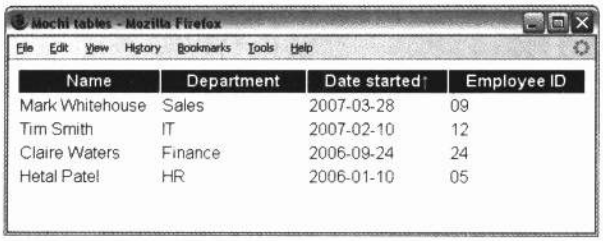


图 12-19

为了创建可排序表，再次需要包含两个脚本；第一个是 MochiKit.js JavaScript 库，第二个是 sortable_tables.js 文件，它们位于 MochiKit 下载代码中(ch12_eg22.html)。

```
<script type="text/javascript"
   src="scripts/MochiKit/lib/MochiKit/MochiKit.js"></script>
<script type="text/javascript"
   src="scripts/MochiKit/examples/sortable_tables/sortable_tables.js"><
/script>
```

接下来添加一些 CSS 样式，区分列与题头并设置使用的字体：

```
<style type="text/css">
  th, td {font-family:arial, verdana, sans-serif;}
  th {background-color:#000000;width:200px;color:#ffffff;}
</style>
```

此处值得注意的部分是表以及如何将表与 MochiKit 脚本集成到一起。需要标识页面的 3 个部分以便利用该脚本：

<table>元素需要具有 id 属性，并且该属性的值是 sortable_table。

<th>(表题头)元素需要具有称为 mochi:sortcolumn 的属性，它的值是用于该列的唯一 id，后面跟上一个空格，然后是该列的数据类型(数据类型可以是用于字符串的 str 或者用于具有给定格式日期的 isoDate)。

<td>元素的第一行需要具有 mochi:content 属性，该属性的值是关键字 item，后面跟上一个句点，然后是用于该列的唯一 id，其中 id 由对应题头中的 mochi:sortcolumn 属性指定。

```
<table id="sortable_table" class="datagrid">
  <thead>
    <tr>
      <th mochi:sortcolumn="name str">Name</th>
      <th mochi:sortcolumn="department str">Department</th>
      <th mochi:sortcolumn="datestarted isoDate">Date started</th>
      <th mochi:sortcolumn="extension str">Employee ID</th>
    </tr>
  </thead>
  <tbody>
    <tr mochi:repeat="item domains">
      <td mochi:content="item.name">Tim Smith</td>
      <td mochi:content="item.department">IT</td>
```

```
      <td mochi:content="item.datestarted">2007-02-10</td>
      <td mochi:content="item.extension">12</td>
    </tr>
    <tr>
      <td>Claire Waters</td>
      <td>Finance</td>
      <td>2006-09-24</td>
      <td>24</td>
    </tr>
    <tr>
      <td>Hetal Patel</td>
      <td>HR</td>
      <td>2006-01-10</td>
      <td>05</td>
    </tr>
    <tr>
      <td>Mark Whitehouse</td>
      <td>Sales</td>
      <td>2007-03-28</td>
      <td>09</td>
    </tr>
  </tbody>
</table>
```

这是如何方便地向表中添加非常复杂功能的另一个示例——创建的效果类似于 Excel 中的“排序数据”选项，该功能在处理大量数据方面非常有用。

12.7.4　利用 YUI 库创建日历

接下来介绍的第三种(也是最后一种)库是雅虎用户界面库。该库由雅虎创建，并且是 3 个库中最大的一个库，具有所有类型的功能。本章的下载代码中包含了该库的 2.4.0 版本；但是，可以从 http://developer.yahoo.com/yui/处下载最新版本。

首先介绍如何使用这种架构方便地将日历表拖动到 Web 页面中。图 12-20 给出了该日历的外观。

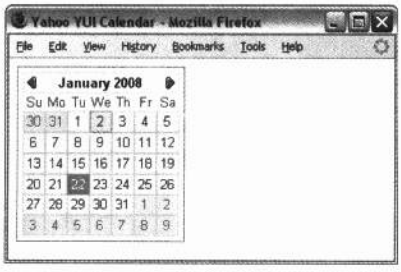


图 12-20

首先需要包含 YUI 库中的两个 JavaScript 文件——第一个是 yahoo-dom-event.js 脚本，第二个是 calendar.js 脚本，其中后者专门用于日历(ch12_eg23.html)。

```
<script type="text/javascript"
  src="scripts/yui/build/yahoo-dom-event/yahoo-dom-event.js"></script>

<script type="text/javascript"
  src="scripts/yui/build/calendar/calendar.js"></script>
```

在这个示例中，还需要包含一些 CSS 文件，这些文件也位于 YUI 的下载代码中：

```
<link rel="stylesheet" type="text/css"
  href="scripts/yui/build/fonts/fonts-min.css" />
<link rel="stylesheet" type="text/css"
  href="scripts/yui/build/calendar/assets/calendar.css" />
```

接下来添加一个 元素，它将被日历填充。

```
<div id="cal1Container"></div>
```

最后，添加调用 YUI 库的脚本，并且在 元素中添加日历。

```
<script type="text/javascript">
  YAHOO.namespace("example.calendar");
  YAHOO.example.calendar.init = function() {
  YAHOO.example.calendar.cal1 = new
YAHOO.widget.Calendar("cal1","cal1Container");
  YAHOO.example.calendar.cal1.render();
  }
 YAHOO.util.Event.onDOMReady(YAHOO.example.calendar.init);
</script>
```

与本节中的其他一些示例一样，日历也可能与一些其他功能关联，例如假日预约表单(对于假日预约表单，可以在其中指定希望旅行的日期或者事件列表，该事件列表列出了在某个特定的日期中发生的事情)。但是，这个示例确实演示了如何使用一些库方便地在页面中添加重要的功能。

12.7.5 利用 YUI 库创建自动完成的文本输入框

本节的最后一个示例是创建一种特殊的文本输入框，该文本输入框可以根据用户输入的内容为用户提供建议。该示例允许输入美国的州名，文本框会根据用户已经输入的内容给出继续输入哪个州的建议。

图 12-21 中给出了该文本输入框的显示效果。

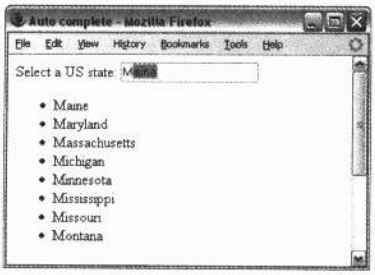


图 12-21

为了实现该示例(ch12_eg24.html)，首先需要包含 3 个 JavaScript 文件：

```
<script type="text/javascript"
  src="scripts/yui/build/yahoo-dom-event/yahoo-dom-event.js"></script>
<script type="text/javascript"
    src="scripts/yui/build/animation/animation.js"></script>
<script type="text/javascript"
    src="scripts/yui/build/autocomplete/autocomplete.js"></script>
```

然后在页面主体中，添加文本输入框和一个 元素，该 元素中包含对即将输入的内容的建议。

```
Select a US state:
<input id="statesinput" type="text">
<div id="statescontainer"></div>
```

接下来创建一个 JavaScript 数组，该数组中包含某人可能输入的所有可能值。

```
<script type="text/javascript">
YAHOO.example.statesArray = [
    "Alabama",
    "Alaska",
    "Arizona",
    "Arkansas",
    "California",
    "Colorado",
    // other states go here
];
</script>
```

最后，在页面中添加 JavaScript 脚本，将文本输入表单控件和数组关联，并且调用 AutoComplete 函数，从而当用户在文本框中输入文本时会出现一些建议。

```
<script type="text/javascript">
YAHOO.example.ACJSArray = new function() {
    // Instantiate first JS Array DataSource
    this.oACDS = new YAHOO.widget.DS_JSArray(YAHOO.example.statesArray);
    // Instantiate first AutoComplete
   this.oAutoComp = new YAHOO.widget.AutoComplete('statesinput','statescontainer',
        this.oACDS);
    this.oAutoComp.prehighlightClassName = "yui-ac-prehighlight";
    this.oAutoComp.typeAhead = true;
    this.oAutoComp.useShadow = true;
    this.oAutoComp.minQueryLength = 0;
    this.oAutoComp.textboxFocusEvent.subscribe(function(){
        var sInputValue = YAHOO.util.Dom.get('statesinput').value;
        if(sInputValue.length === 0) {
            var oSelf = this;
            setTimeout(function(){oSelf.sendQuery(sInputValue);},0);
        }
    });
```

```
};
</script>
```

根据这个简单示例可以了解到，利用 JavaScript 工具箱可以显著增强页面的可用性和功能(而不需要从头开始编写完成该工作所需的所有代码)。

在 Web 中存在更多的 JavaScript 库，每一个库具有不同的功能，并且都在持续开发和改进，因此值得花费一些时间查看可用的不同库，并且经常检查喜好的库，以便了解它们的更新情况。

12.8 何时不使用 JavaScript

本章已经介绍许多需要使用 JavaScript 的示例，但需要提醒的是，有些时候最好避免使用 JavaScript，因此在结束本章之前简要介绍这些情况。

12.8.1 下拉导航菜单

我所服务的一些客户曾经要求使用 JavaScript 为导航创建一些特效。一种最常见的请求是将 JavaScript 用于下拉导航菜单，其中通过菜单上的主项下拉到达子页面。我不建议客户使用这些特效，原因包括 3 个方面：

- 这种技术无法工作于已经在浏览器中关闭 JavaScript 功能的用户。虽然这样的用户所占的比例非常小，但是这种技术会使他们无法访问这些网页。
- 这种技术趋向于在不同的浏览器中执行的效果不同，很难使一个脚本工作于所有的浏览器中。
- 用户会发现难以单击移动的菜单的某个部分(特别是当用户身体残疾或鼠标存在问题时)。

12.8.2 隐藏 e-mail 地址

我已经在 Web 上看到很多文章建议使用 JavaScript 在页面中编写 e-mail 地址(使用文档对象的 write()方法写出 e-mail 地址，而不是利用普通的<a>链接和 XHTML)。目标是避免获得太多的垃圾邮件。垃圾邮件获得资源的方式是使用小程序(这些小程序通常具有的名称是机器人或蜘蛛)搜索各种 Web 站点来查找 e-mail 地址。然后将这些 e-mail 地址作为发送垃圾邮件的目标。这种思想存在的问题是，对于在浏览器中关闭 JavaScript 功能的用户来说，他们将无法看到 e-mail 地址。一种较好的备选方法是提供一个 e-mail 表单，该表单向您发送查询——然后一旦收到查询，您就能够确保用户的查询不是仅为获得一个 e-mail 地址，因此可以安全地将您的 e-mail 地址发送给该用户。

12.8.3 快速跳转选择框

某些站点在表单中提供选择框(类似于第 5 章中介绍的选择框)作为导航菜单——通常

称为快速跳转菜单，当用户从下拉列表框中选择某项时，它能够将用户直接带到不同的页面或站点的不同部分。某些快速跳转菜单使用脚本自动将用户带到所选择的页面，而不需要用户单击 GO 按钮或 Submit 按钮。相反，脚本被设置为检测选择框中的改变，然后将用户带到相应的页面。这是存在问题的实践，原因包括两个方面：

- 用户可以使用上下方向键从选择框中选择某项，这样做的用户将自动被带到第一个选项所代表的页面，前提是该用户首先按下下方向键。如果使用键盘，则将无法跨越该选项。虽然聪明的用户可以快速克服这个问题，但是使用键盘而非鼠标的残疾用户在导航该站点时将遇到很多阻力。
- 此外，如果用户禁用 JavaScript 功能，则该菜单无法工作。

12.8.4　用户需要从站点中获得的任何信息

决定使用 JavaScript 的底线是，它是否可以简单地增强用户的使用体验，或者用户是否需要它来执行操作或查看某些重要信息。不应该设计需要使用 JavaScript 提供必须功能的页面——请记住“禁用提交按钮直到已经选择某个复选框”一节中的经验教训。

12.9　本章小结

在本章中介绍了 JavaScript 的很多用途，您应当已经很好地理解了如何应用从上一章开始学习的这种语言。通过这些脚本的帮助，您现在应当能够在自己的页面中使用这些和其他一些脚本。您也应当已经理解如何剪裁甚至编写自己的脚本。

本章介绍了如何通过提供验证帮助用户正确地填写表单。例如，可以通过检查以确保要求必须填写的字段中具有内容，或者确保 e-mail 地址遵循期望的模式。这种方式告诉用户他们必须执行的操作，以防止错误的页面发送给服务器，进行处理，然后返回错误，从而节省时间。本章给出的验证示例强调了访问提供给文档内容的 DOM，以便能够对用户提供的值执行操作。这是极好的交互性示例——文档不仅提供表单，而且提供填写它的帮助信息。当然，表单是任何希望从访问者处收集信息的 Web 站点的重要部分。

本章也介绍了 DOM 能够使表单具有更好的可用性，方式是将焦点放置在表单合适的位置，以及操作用户已经输入的文本，例如删除或替换特定的字符。

接下来，本章继续介绍了几种其他的 JavaScript 技术，例如图像翻转、随机内容和弹出式窗口。这些技术将帮助理解利用 JavaScript 能够执行的操作，以及如何将脚本集成到页面中。

最后，本章介绍了 3 种流行的 JavaScript 库：Scriptaculous 库、MochiKit 库和雅虎用户界面库。JavaScript 库提供复杂的功能，开发人员仅需要使用少量代码就能够方便地将它们集成到页面中(开发人员仅需要学习如何将脚本集成到页面中)。

但是，需要记住的一件关键事情是，只能使用 JavaScript 技术增强页面的效果，而不能依赖它显示内容或提供一些功能。

12.10 练习

本章只有一个练习题，因为它非常长。所有练习的答案都在附录 A 中给出。

1. 您的任务是为图 12-22 中的竞赛表单创建一个验证函数。

该函数需要检查用户是否执行以下操作：

- 输入姓名
- 提供有效的 e-mail 地址
- 选择其中一个单选按钮作为问题的答案
- 赋予加时赛问题的答案，该答案不超过 20 个单词。

检查的顺序应当遵循控件在表单上显示的顺序。

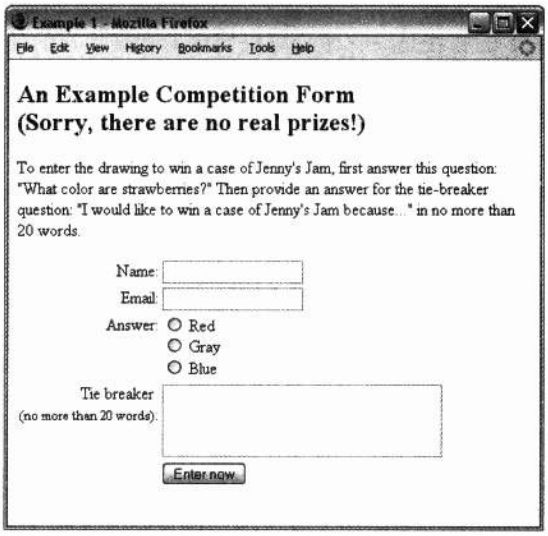


图 12-22

下面是该表单的代码：

```
<form name="frmCompetition" action="competition.aspx" method="post"
onsubmit="return validate(this);">
<h2>An Example Competition Form <br />(Sorry, there are no real prizes!)</h2>
<p> To enter the drawing to win a case of Jenny's Jam, first answer
this question: "What color are strawberries?" Then provide an answer for
the tie-breaker question: "I would like to win a case of Jenny's Jam
because..." in no more than 20 words.</p>
<table>
  <tr>
    <td class="formTitle">Name: </td>
    <td><input type="text" name="txtName" size="18" /></td>
  </tr>
  <tr>
    <td class="formTitle">Email: </td>
```

```
      <td><input type="text" name="txtEmail" size="18" /></td>
    </tr>
    <tr>
      <td class="formTitle">Answer: </td>
      <td><input type="radio" name="radAnswer" value="Red" /> Red<br />
          <input type="radio" name="radAnswer" value="Gray" /> Gray<br />
          <input type="radio" name="radAnswer" value="Blue" /> Blue
      </td>
    </tr>
    <tr>
      <td class="formTitle">Tie breaker <br/ ><small>(no more than 20 words)
      </small>: </td>
      <td><textarea name="txtTieBreaker" cols="30" rows="3"
/></textarea></td>
    </tr>
    <tr>
      <td class="formTitle"></td>
      <td><input type="submit" value="Enter now" /></td>
    </tr>
  </table>
  </form>
```

第13章

在 Web 上发布站点

创建 Web 站点之后，您将希望它可以被所有人访问。在本书的最后一章中将介绍如何准备站点以及如何将其放置到 Web 上。本章也将介绍如何鼓励访问者访问您的站点。

Web 站点位于称为 Web 服务器的特殊计算机上，Web 服务器通常连接 Internet。相比于购买并运行自己的 Web 服务器，通常较为经济的方式是租用虚拟主机公司提供的 Web 服务器空间。为了帮助您选择正确的虚拟主机公司(实际上是选择虚拟主机公司的正确数据包)，需要学习这些公司所使用的关键术语。在本章中将介绍什么是共享主机，什么是专用主机，以及如何确定需要多少空间和带宽等。

但是，在将站点放置在 Web 服务器中之前，应当执行一些检查和测试，从验证文档和检查链接到确保站点能够工作于不同分辨率的屏幕上以及文本的可读性。在将站点放置在 Web 上之后，如果顾客告诉您某些到产品页面的链接无法工作，或者他们不能在其计算机上看到站点，则会降低站点的信誉。因此，在上传站点之前必须进行测试。然后在将站点放置在 Web 服务器上之后，可以执行一些其他类型的检查和测试——毕竟，虽然某个站点在您的计算机上能够良好运作，但在转移到新的服务器上之后可能会存在大量问题。

一旦准备好站点以供公众访问，您将希望确保他们知道该站点！您将希望确保该站点被主要搜索引擎索引，例如 Google 和 Yahoo；为了使您的站点尽可能接近排名的顶部，需要执行涉及大量反复试验的复杂过程。您可能也考虑使用大量其他策略以使人们知道您的站点，例如按点击付费的广告(类似于 Google 的 AdWords 或 Yahoo 的 Overture)。在付出大量艰辛工作创建站点之后，您肯定希望它是一个成功的站点。

熟悉本书中介绍的知识之后，请快速查看一些您可能喜欢的其他技术以开始检测站点。

但是在学习这些内容之前，首先需要了解在本书中还未介绍的最后一个元素<meta>，该元素提供关于文档和它们内容的信息。

在本章中将介绍以下内容：

- 如何使用<meta>元素
- 执行一些测试以确保站点按照预期的方式运作
- 检查站点是否可以访问
- 查找主机，使站点能够被 Web 上的所有人访问
- 利用 FTP 将站点从您的计算机上上传到主机的 Web 服务器
- 将站点提交给搜索引擎
- 增加访问者数量
- 使用按点击付费的广告

- 发现可能希望家下来使用的其他技术
- 控制站点的不同版本，以便可以进行更改而不产生错误

13.1 Meta 标签

在开始了解如何测试站点之前，需要学习最后一个标签：<meta>标签。元标签位于文档的<head>部分中，它包含关于文档的信息。<meta>标签位于文档的<head>部分中而不是<body>部分中，这是因为<body>部分主要用于文档的实际内容，而<meta>标签用于给出关于文档主体中内容的信息。该信息可用于很多目的，包括帮助搜索引擎索引站点，指定文档的作者，以及如果文档具有时效性，则指定页面什么时候过期。

<meta>元素是一个空元素，因此它没有结束标签；相反，<meta>元素在属性内附带信息，因此在该元素的末尾需要使用一个正斜杠字符。例如，下面的<meta>元素提供对一个计算机书籍商店 Web 站点的描述：

```
<meta name="description" content="Buy computer programming books to learn
HTML,XHTML, JavaScript, ASP.Net, PHP, Ruby" />
```

<meta>元素可以附带 8 个属性，其中 4 个是通用属性——dir、lang、xml:lang 和 title。但是，其他 4 个属性是<meta>元素特有的属性：

- schema
- name
- content
- http-equiv

name 属性和 content 属性趋向于一起使用，http-equiv 属性和 content 属性也是如此。下面的小节中将分别介绍每一对属性。

13.1.1 name 属性和 content 属性

name 属性和 content 属性指定文档的特性。name 属性的值是希望设置的特性，content 属性的值是希望提供给在 name 属性中设置的特性的设置。在前面的示例中，<meta>元素设置了文档或站点的内容的 description 特性。name 属性的值为 description，content 属性的值是对站点的描述：

```
<meta name="description" content="Buy computer programming books to learn
HTML,XHTML, JavaScript, ASP.Net, PHP, Ruby" />
```

name 属性的值可以是任何内容；在任何标准中都没有规定限制。因此，如果希望添加关于文档和其内容的信息，可以使用这种方便的技术。但是，也存在一些预定义的值，您将会经常看到它们的使用。这些预定义的值如下所示：

- description：指定页面的描述
- keywords：包含由逗号隔开的关键字列表，用户可以利用这些关键字进行搜索以找到页面

- robots：指示搜索引擎如何索引该页面

description 特性和 keywords 特性可以被称为机器人或蜘蛛的程序使用，大多数搜索引擎使用这些程序帮助索引 Web 站点，因此值得将这两个特性添加到任何 Web 页面中。这些程序扫描各种 Web 站点，并将它们的信息添加到搜索引擎使用的数据库中，然后当遇到链接时跟踪这些链接并索引这些页面——这是搜索引擎实现索引如此多站点的方式。

1. 将 name 属性的值设置为 description

本章前面给出了将 name 特性的值设置为 description 的示例，该示例使用 content 特性指定描述站点的语句。有时当站点返回作为用户搜索的响应时，搜索引擎将显示 description 特性的值。

描述的长度应当最大具有 200 个字符，但是某些搜索引擎，例如 Google，仅显示前 100 个字符，因此应当尝试在前 100 个字符中介绍站点的主要内容。

2. 将 name 属性的值设置为 keywords

keywords 特性提供一个关键字列表，搜索引擎可以使用这些关键字索引站点。如果有人输入的单词或单词组合是您使用的关键字，则搜索引擎返回您的站点的几率将增加。例如，一个在线计算机书店可以使用类似如下形式的关键字：

```
<meta name="keywords" content="computer, programming, books, web, asp,
asp.net, C#, vb, visual basic, c++, Java, Linux, XML, professional, developer,
html, html, css, xslt, access, sql, php, mysql" />
```

站点提供的单词越多，则页面在搜索引擎中显示的几率越高；但是，不能使用与站点内容不直接相关的单词作为关键字，理想情况下关键字应当出现在页面的文本中。

注意：

人们通常认为虽然关键字在过去对搜索引擎索引站点的方式具有很大的影响，但是目前它们的影响有限。然而，如果使用所有可用的策略以使站点被搜索引擎识别，则人们更有可能找到您的站点。

大多数搜索引擎对它们即将索引的关键字数量具有自己的限制，并且该数量对于不同的搜索引擎存在区别，但是通常应当保持关键字的数量少于 1000 个字符。

也可以将 lang 属性与描述和关键字一起使用，以指明描述和关键字所使用的语言，或者以多种语言提供它们。例如，下面是采用美国英语书写的关键字：

```
<meta name="keywords" content="computer, programming, books" lang="en-us" />
```

下面是采用法语书写的关键字：

```
<meta name="keywords" content="livres, ordinatteur, programmation"
lang="fr" />
```

下面是采用德语书写的关键字：

```
<meta name="keywords" content="" lang="programmieren, bucher, computers"
lang="de" />
```

3. 将 name 属性的值设置为 robots

本章前面提及，许多搜索引擎使用一些小程序代表其索引 Web 页面。可以将 name 属性的值设置为 robots 来阻止这些程序索引页面或页面中的链接(因为许多这些程序也跟踪它们在站点上找到的链接并索引这些链接)。例如，您很可能不希望搜索引擎索引您仍在开发的页面或者用于管理站点的页面——因为希望人们碰巧看到它们。

在下面的代码中，<meta>元素告诉搜索引擎不索引这个页面或者跟踪该页面中的任何链接以索引页面。

```
<meta name="robots" content="noindex, nofollow" />
```

content 属性可以具有表 13-1 中列举的值。

表 13-1

值	意　义
all	索引所有页面
none	不索引任何页面
index	索引当前页面
noindex	不索引当前页面
follow	跟踪这个页面中的链接
nofollow	不跟踪这个页面中的链接

默认情况下，该属性的值应当是 all、index 和 follow，即允许 Web 机器人跟踪任何链接和索引所有页面。

如果希望阻止页面被索引，则在采用这种技术时应当结合使用称为 robots.txt 的文件，本章后面的“robots.txt”一节中将讨论该内容。

13.1.2　http-equiv 属性和 content 属性

http-equiv 属性和 content 属性通常成对使用，用于设置 HTTP 头的值。每次 Web 浏览器请求页面时，HTTP 头与请求一起发送，而每次服务器响应请求将页面发送回客户端时，它会将 HTTP 头发送回客户端：

- 当浏览器请求服务器中的页面时，从浏览器发送给服务器的 HTTP 头包含的信息包括：浏览器将接受的格式、浏览器的类型、操作系统、屏幕分辨率、日期和其他关于用户配置的信息。
- 从服务器返回给 Web 浏览器的 HTTP 头包含的信息包括：服务器的类型、页面发送的日期和时间，页面最近一次修改的时间。

当然，HTTP 头可以包含更多信息，并且使用<meta>标签是添加随文档一起发送的新 HTTP 头的一种方法。例如，您可能希望添加一个 HTTP 头以指明页面什么时候过期(不再有效)——如果文档包含类似特价的信息(特价将会过期)，则该功能非常有用——或者在一段时间之后刷新页面。

1. 使页面过期

使页面过期非常重要，因为浏览器具有缓存，缓存是硬盘上的一个空间，浏览器会将已经访问过的 Web 站点页面存储在其中。如果重新访问一个已经访问过的站点，则浏览器可以从缓存中加载某些或所有页面，而不是重新获取整个页面。

在下面的示例中，<meta>标签使页面在 2010 年 4 月 16 日(星期五)的晚上 11 点 59 分 59 秒过期。注意，日期必须遵循所给出的格式。

```
<meta http-equiv="expires" content="Fri, 16 April 2010 23:59:59 GMT" />
```

如果这段代码包含在文档中，并且用户尝试在到达过期日期之后加载该页面，则浏览器将不会使用缓存的版本；浏览器将尝试从服务器中获取新版本。这可以帮助确保用户获得文档的最新版本，从而防止人们使用过期信息。

2. 阻止浏览器缓存页面

可以将 http-equiv 属性的值设置为 pragma，将 content 属性的值设置为 no-cache，以此阻止一些浏览器缓存页面，如下所示：

```
<meta http-equiv="pragma" content="no-cache" />
```

但是，Internet Explorer 4 浏览器以及以后的版本将忽略这条规则并缓存页面。

3. 刷新和重定向页面

可以使用如下的<meta>标签将页面设置为在一段时间之后刷新，其中<meta>标签的 http-equiv 属性值设置为 refresh：

```
<meta http-equiv="refresh" content="10;URL=http://www.wrox.com/latest.aspx" />
```

这将使页面在 10 秒之后刷新。秒数作为 content 属性值的第一部分提供，在其后面是一个分号，然后是关键字 URL，接下来是一个等号，最后是即将刷新的页面地址。

甚至可以将当前页面刷新为不同的页面。例如，如果站点从某个域移动到另外一个域，则可以为访问原有域的访问者保留一个页面，说明站点已经移动，并且在 5 秒之内将用户自动重定向到新的页面。

当使用这种技术重新加载相同页面时，则该技术称为刷新页面；而将用户发送到新页面或站点则称为重定向用户。

注意：

应当避免频繁刷新页面，因为这会使用户分心，特别是当用户阅读文档时。另外需要注意的是，定期刷新文档会给 Web 服务器带来额外的负担。

4. 指定等级

可以指定与页面内容相关的等级。如果站点没有等级，则某些浏览器(或者一些程序，这些程序能够控制查看的内容)将会阻止访问该站点。与所有<meta>标签一样，用户不会看

到等级，但是浏览器能够处理它。如果提供等级，则浏览器能够向允许访问该类内容的用户显示页面。

最初引入 Internet 等级是为了帮助父母和学校防止孩子查看某些内容，但是这个领域中的主要技术 PICS(Platform for Internet Content Selection，用于 Internet 内容选择的平台)已经开发出来，从而可用于其他很多方面。

为了对页面指定一个等级值，http-equiv 属性的值需要设置为 pics-label。实际指示内容的部分称为等级标签，该标签的创建必须符合 Internet 内容分级协会(Internet Content Ratings Association，ICRA)制定的规则。

等级标签由 4 个部分组成：

- ICRA 标识符
- ICRA 标签
- RSACi 标识符(ICRA 的原有名称)
- RSACi 等级

标签看上去可以非常复杂，但是 ICRA Web 站点(www.icra.org/label/)上的表单可以帮助您为自己的站点创建标签。作为过程的一部分，也在该站点上生成 RSACi 等级。

有了标签之后，<meta>标签应当看上去如下所示，为 www.wrox.com/站点创建该标签：

```
<meta http-equiv="pics-label" content='(pics-1.1 "http://www.icra.org/
ratingsv02.html"
comment "ICRAonline EN v2.0" l gen true for
"http://www.wrox.com/" r (nz 1 vz 1 lz 1 oz 1 cz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.wrox.com/"
r (n 0 s 0 v 0 l 0))' />
```

虽然这段代码看上去可能比较复杂，但是 ICRA 站点上的表单使得生成等级变得非常简单，生成过程不会超过几分钟。

5. 设置 cookies

cookies 是一些小的文本文件，浏览器可以将它们存储在用户的计算机上。可以使用一种运行在浏览器中的脚本语言(例如 JavaScript)创建 cookies，或者使用服务器上的技术(例如 ASP.Net、PHP 或 JSP)创建它们。

注意：

您很可能直到深入掌握 JavaScript 或一种服务器端语言才能够使用 cookies，这里提出它们只是为了将来的使用。

可以使用<meta>元素设置 cookies，方式是赋予 http-equiv 属性的值为 set-cookie，然后使用 content 属性指定一种 cookies 名称、值和过期日期，如下所示：

```
<meta http-equiv="Set-Cookie" content="cookie_name=myCookie;
expires="Fri 16 April 2009 23:59:59 GMT" />
```

如果没有提供过期日期，则 cookie 将在用户关闭浏览器窗口之后过期。

6. 指定作者的姓名

可以设置文档的作者姓名，方式是为 http-equiv 属性设置 author 值，然后使用作者的姓名作为 content 属性的值，如下所示：

```
<meta http-equiv="author" content="Jon Duckett" />
```

7. 设置字符编码

字符编码指示存储在文件中的字符使用的字符编码。可以利用<meta>标签指定文档中所使用的字符编码，方式是将 http-equiv 属性的值设置为 Content-Type。然后，应该将 content 属性的值设置为该文档所使用的字符编码；例如：

```
<meta http-equiv="Content-Type" content="ISO-8859-1" />
```

这里的文档使用 ISO-8859-1 编码编写。在附录 E 中将介绍更多字符编码方面的内容。

8. 设置默认样式表语言

可以指定文档中使用的样式表语言的类型，方式是将 http-equiv 属性的值设置为 content-style-type，然后在 content 属性中指定样式表语言的 MIME 类型。

```
<meta http-equiv="content-style-type" content="text/css" />
```

当样式表规则位于一个<style>元素中时，type 属性指明该元素内使用的样式表语言；但是当内置的样式表规则使用了一个元素的样式属性时，则无法显式地指明使用的语言。因此，将 CSS 设置为默认语言可以解决该问题。虽然 CSS 是赋予 HTML 和 XHTML 文档样式的最流行语言，但是某些应用程序支持其他语言，例如 XSLT 和 DSSSL(但是这些语言很少用于 Web 页面，因此您将很少看到在 Web 页面中使用 content-style-type)。

9. 设置默认脚本语言

如果希望在页面中使用脚本，则可以使用<meta>元素指明脚本使用的语言。虽然仍然需要在任何<script>元素中使用 type 属性，但是通常可以在事件处理程序中使用脚本，并使用<meta>标签指明这些属性中使用的语言。

```
<meta http-equiv="content-script-type" content="text/JavaScript" />
```

13.1.3 scheme 属性

scheme 属性可以用于为特性值指定一种模式或格式。例如，如果正在处理的是日期，则可以以多种方式编写它们。在美国，日期格式为 mm-dd-yyyy；而在欧洲，日期格式为 dd-mm-yyyy。因此，可以使用 scheme 属性指定一种日期格式。在美国，可以使用如下格式：

```
<meta scheme="usa" name="date" content="04-16-1975" />
```

在欧洲，可以使用如下格式：

```
<meta scheme="Europe" name="date" content="16-04-75" />
```

使用 scheme 属性时，通常假设正在处理的应用程序支持 scheme 属性和 name 属性的值——如果主流的浏览器不理解该属性，则由脚本或自定义的应用程序来解释该元素的使用。

13.2　测试站点

在向公众推出站点之前，应当执行一些测试。即使站点看上去能够在您的计算机上良好运作，但是无法确保它也能够在其他人的计算机上良好运作。毕竟不同的人具有不同的浏览器版本、不同的 Internet 连接速度、不同的屏幕大小和分辨率，因此能够工作于您的计算机上的页面可能无法很好地工作于其他人的计算机上。

因此，测试的两个阶段如下所示：

- 公布之前的测试：邀请其他人查看站点之前在您的计算机上执行的测试。
- 发布之前的测试：在即将公布站点的 Web 服务器上执行的测试。

在本节中将介绍多种测试方法，这些测试方法可以确保站点可用于尽可能多的人。

13.2.1　目录结构和相对 URL 的重要性

首先花费一点时间回顾如何最合适地编写站点中链接到其他页面的 URL，以及每个页面使用的图像、样式表和外部脚本。下面的示例演示了相对 URL 的价值。假设已经构建了一个站点，并且希望创建一个新版本。您可能希望在 Web 服务器上测试新站点，但是它必须位于一个独立的文件夹中，因此它的 URL 与当前站点的 URL 不同。例如，可能在称为 newsite 的文件夹中测试新站点，因此主页的 URL 可能如下所示：

```
http://www.example.com/newsite/index.html
```

但是，在准备好切换为新站点时，可能希望它的 URL 为：

```
http://www.example.com/index.html
```

如果使用相对 URL 链接其他所有页面、图像、脚本文件等，则将站点移动到新文件夹或者新 URL 不会带来任何问题。但是，如果链接采用的是绝对 URL，例如主页面上的徽标使用如下所示的<img>标签：

```
<img src="http://www.example.com/newsite/images/our_logo.gif" alt="Our
Logo" />
```

则在将站点移动到新域或不同的文件夹时，该图像无法加载。因此，最好使用如下方式：

```
<img src="images/our_logo.gif" alt="our logo" />
```

现在，只要 images 文件夹位于这个页面所在的目录中，则无论站点移动到何处，图像都会正确加载。

13.2.2 验证 HTML、XHTML 和 CSS

为了使站点能够工作于大多数浏览器中，一种最佳测试方法是验证代码并确保它符合所采用语言的规则。存在一种验证器，它能够检查是否正确关闭所有标签，使用的属性是否被它的元素支持等。验证器的作用非常重要，例如如果在文档中忘记编写结束标签</td>，虽然该页面在您的计算机上可能会运行良好，但它很可能无法工作于其他人的计算机上。

注意：

在构建第一个页面之后验证设计非常有用，因为很可能会将第一个页面作为模板，并将文件中的一部分代码复制并粘帖到其他页面中。如果用作为模板的页面中存在错误，并且在测试之前使用它创建了所有其他的站点，则将不得不重写每一个页面。

第 1 章中讨论过，HTML 和 XHTML 的每一个版本都至少具有一个文档，该文档中包含了该语言版本的一些规则，它称为 DTD 或模式。可以根据该文档验证任何 Web 页面，以确保它遵循这些规则。因此，通过验证页面，将能够知道是否遗漏一个标签或标记其他重要部分。页面起始部分的 DOCTYPE 声明将告诉验证工具页面应当匹配哪个 DTD 或模式的规则。

许多 Web 页面制作工具，例如 Dreamweaver，包含了可用于验证站点的工具。但是如果没有使用这样的工具，或者希望采用多种验证工具检查站点，则可以使用 W3C 的免费 Web 页面验证器，其网址为 http://validator.w3.org/。

在图 13-1 中可以看到 W3C 的标记验证器：它允许输入站点的 URL 或者从计算机中上传一个页面。

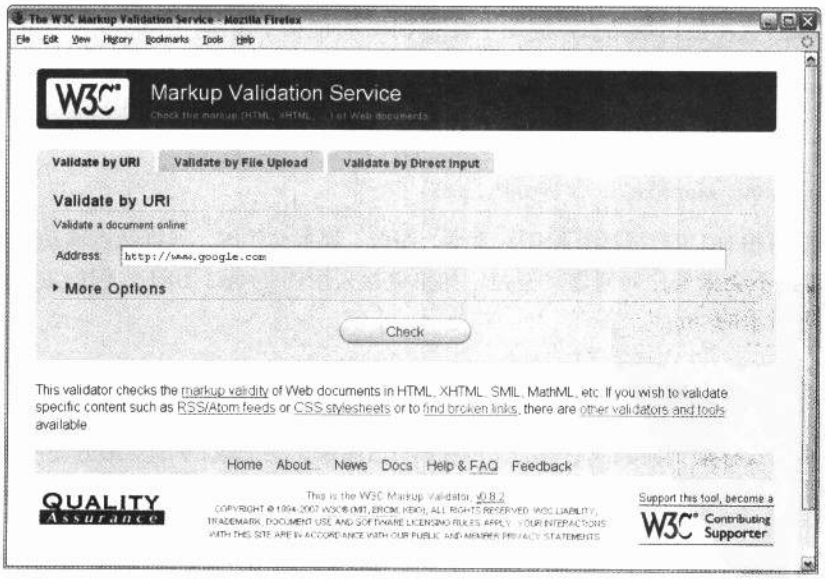


图 13-1

然后，该验证器将告诉您文档中是否存在错误，图 13-2 中显示了该页面存在的错误。

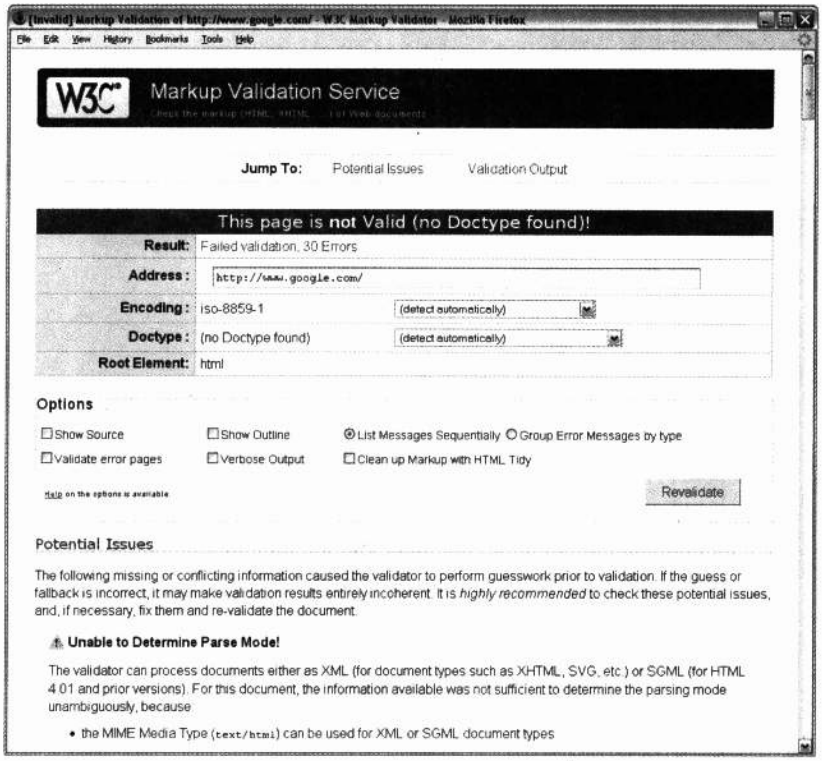


图 13-2

虽然 W3C 验证工具非常有效——并且是免费的——但是必须单独验证每一个页面，因此使用起来有些不便。Dreamweaver(对于专业 Web 开发人员来说，它是到目前为止最流行的 XHTML 制作工具)在它的 Dreamweaver MX 版本中引入了很好的 XHTML 页面验证功能(比早期 Dreamweaver 版本提供的验证功能优秀很多)。验证页面变得非常简单，只需要保存它，然后按下 Shift+F6 键；然后就可以在结果面板中看到页面中存在的错误，如图 13-3 所示。

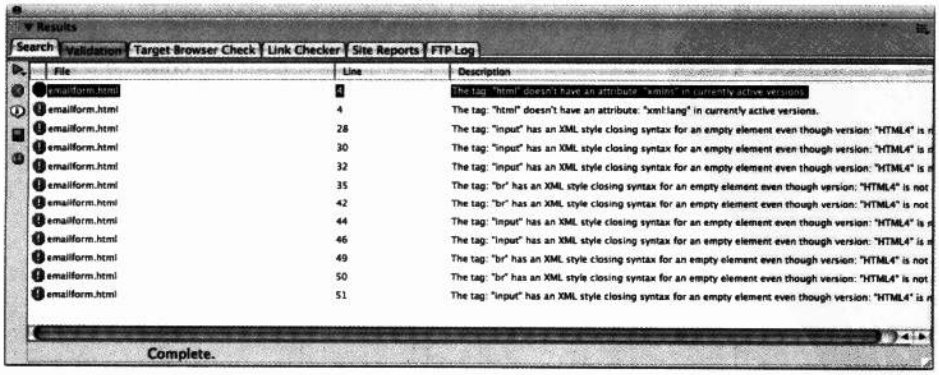


图 13-3

注意，为了使该功能能够正确起作用，必须在 Dreamweaver 中进行正确设置。为了获得设置，可以右击结果面板并选择设置对话框(或者 Mac 计算机上的 Option-click)。然后将在新的 Preferences 对话框中看到文档标准的整个范围。需要确保除了希望再次检查的版本之外没有选中其他任何选项。因此，如果正在验证 Transitional XHTML 1.0，则必须只选中该复选框，如图 13-4 所示。

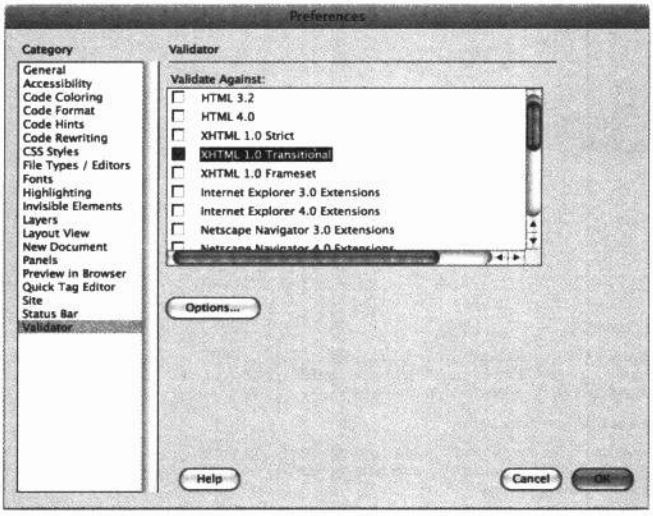


图 13-4

现在重新验证该页面。

13.2.3 检查链接

在站点设计完成以及将其公布在 Web 上之前，检查链接是非常重要的一个过程。存在一些可用于检查链接的工具。如果搜索链接检查工具，则将找到多个有偿提供该服务的站点。但是，也存在一些免费链接检查服务，例如：

- W3C 的链接验证服务，位于 http://validator.w3.org/checklink/。
- HTMLHELP 的 Link Valet 链接验证服务，位于 www.htmlhelp.com/tools/valet/。

也可以使用 Link Valet 工具检查自从某个指定日期之后链接的任何站点是否改变。这个功能非常有用，因为一个外部站点可能改变它的页面结构，并且原有的 URL 将不再有效，或者外部站点可能开始发布您不再希望链接的内容。

在图 13-5 中显示了利用 W3C 的链接验证器验证一个页面时产生的结果。

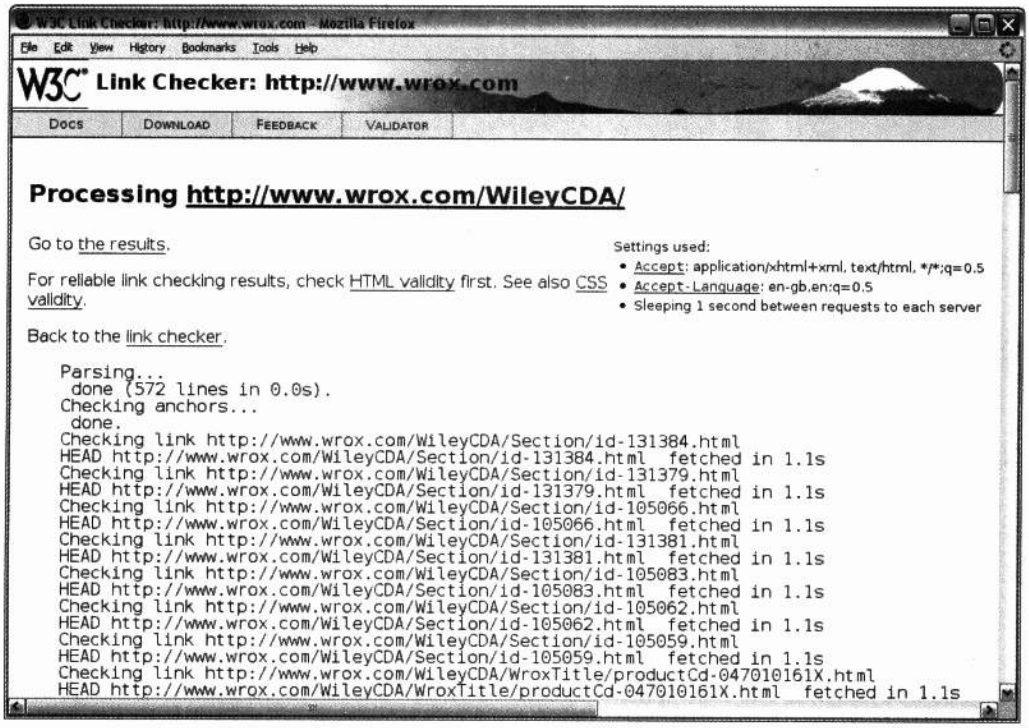


图 13-5

这些服务产生的结果可能会非常冗长，但是通过查看一些突出显示的内容，可以知道哪些链接无效——通常使用红色显示中断的或有问题的链接。

Macromedia 公司的 Dreamweaver 也包含自己的链接检查工具。可以通过“结果”菜单或者按下 Control+Shift+F9 组合键来访问该工具。

存在一些用于检查页面、文件夹或者整个站点的选项。一旦 Dreamweaver 找到存在问题的链接，则可以在 Results 窗口或者 Properties 窗口中修复它们，如图 13-6 所示(或者进入相关页面的代码)。

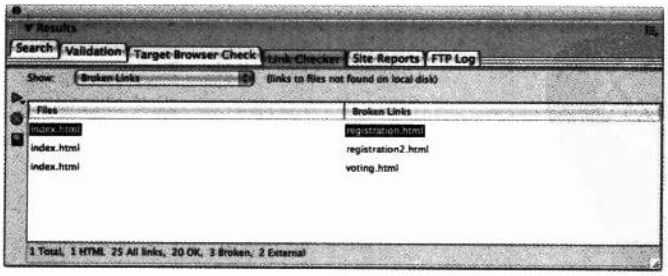


图 13-6

13.2.4 检查不同的屏幕分辨率和颜色深度

第 9 章中已经介绍过，并不是所有人的屏幕分辨率都和您相同；应当在不同的屏幕分辨率下检查您的站点，以确保所有文本仍然可读，并且信息被合理地布局在页面上。

在大多数操作系统中，可以改变屏幕的分辨率(通常是利用显示属性对话框)。通过这种方式可以不同显示器中的图片外观。

也可以将颜色从数百万种颜色改成 256 色，并且确保文本仍然可读(即使图像看上去没有期望的那么美观)。这是检查文本可读性的优秀方式。

13.2.5 可访问性检验工具

创建可访问的 Web 站点非常重要；通常人们认为这是使站点能够被具有视觉或行为障碍的访问者查看，但是它实际上意味着使站点能够被尽可能多的人访问。

在本书中已经介绍了很多关于可访问性的问题，例如为图像提供备选文本，提供链接以使用户跳过重复的导航等。确保所选中的颜色与任何文本具有足够的对比度以阅读任何文本，正确线性化表以及不单独使用颜色传递信息等，这些都是非常重要的工作。

可以利用很多工具检查可访问性的一些主要方面；下面是一些流行的工具：

- Watchfire 公司的 WebExact 工具，位于 http://webxact.watchfire.com/。
- WebAim 公司的 Wave 工具，位于 www.webaim.org/resources/wave/。
- UsableNet 公司的 LIFT 工具，位于 www.usablenet.com/usablenet_liftmachine.html。

这些工具都基于规则，它们都不适合于理解可访问性的内在问题。虽然工具能够检查在每幅图像中是否使用 alt 属性，但是不能检查使用的备选文本是否能够使看不到页面内图像的访问者理解图像的意义。关于可访问性主题的优秀参考书籍是 Jim Thatcher 等人编写的 *Constructing Accessible Web Sites*(Glasshauss，2002)。

13.2.6 开发服务器或主运行服务器

执行到目前为止介绍的检查之后，就可以准备将站点移动到其他人能够测试它的位置。本章后面将介绍 Web 主机托管，但是您很可能希望首先将站点放置在公众无法看到的位置；您将希望首先执行第二阶段的测试。

如果有一个已经投入使用的站点，并且正在进行更新，则需要具有一个不同的位置用于测试新站点——该位置可以是计算机上的一个不同文件夹，或者可以是一个独立的服务器。

注意：

如果正在执行对已有站点的改动，则应当对站点的一个独立副本进行操作，而不是修改公众能够看到的版本。对站点的每一个版本进行备份也是优秀的实践。

13.2.7 在浏览器的不同版本中执行检查

即使在编写页面时尽可能符合各种规范，并且正确验证了页面，但是页面在不同操作

系统上的不同版本浏览器中的显示仍然会不同。在您的显示器上看上去良好的页面，在朋友或同事的显示器上看上去也可能不同。因此，应当尝试在尽可能多的浏览器和平台上测试编写的 Web 页面。至少在您的计算机上应当安装最新版本的 Internet Explorer 浏览器和 Firefox 浏览器。

通常在一台计算机上仅能够安装一个版本的 Internet Explorer 浏览器(除非具有一个分区运行 Windows 的另一个版本，或者运行 Windows 仿真器)，并且您很可能希望每天使用最新的版本。因此如果具有一台闲置的陈旧计算机，则可以在其上安装较老版本的浏览器，在构建页面之后使用它测试这些页面。

如果需要下载较老版本的浏览器，可以访问 http://browsers.evolt.org/。

注意：

某些 Web 站点和服务能够提供 Web 站点每个页面在多种浏览器的不同版本上的屏幕截图，从而可以检查页面在不同浏览器上的外观；但是这种检查方式非常昂贵和费时。这种 Web 站点的一个优秀示例是 www.browsercam.com/。

另一种检查站点是否以期望的方式运作的优秀方法是，在发布站点之前请求所有朋友检查它，希望他们中至少有一到两个人具有较老的浏览器或不同的操作系统。请求他们检查页面的外观，并将他们浏览器中的一些页面的屏幕截图发送给您。如果愿意，甚至可以提供少量奖品作为回报。我的一个同事最近请求客户的所有职员邀请他们的朋友测试新 Web 站点，其中回答相关调查问卷一位幸运的人将赢得一台 iPod。

13.2.8　引导测试

如果可以的话，最好在站点没有发布给公众之前让没有参与项目的人测试站点。这项工作非常重要，因为有些内容对您来说可能是显而易见的，但是对于第一次访问站点的人来说可能并不清楚。在构建站点时，您与设计和工作密切接近，因此如果目标与没有看过该站点的人相同，则检查它就会非常困难。

理想情况下，对站点执行这种测试的人员将是站点的目标观众。

首先可以做的事情是邀请参与者坐在站点之前，然后观察他们执行的操作。必须始终避免干扰他们，无论是希望询问某人即将尝试执行的操作或者告诉他们如何获得您认为他们正在尝试查找的内容。只要和他们对话，就会影响他们的正常行为——从而将不会了解很多信息。

观察人们首先到达何处，他们在每个页面上停留多长时间，以及他们如何导航，这些都能够提供关于站点的很多信息。

也可以坐在这些人的旁边，告诉他们一些虚构的情况；例如，如果具有一个销售自行车的站点，可以请求用户执行以下操作：

- 查找一辆他们认为适合子女的自行车
- 找出特定型号的自行车的价格
- 找出访问商店的方式以及它的开放时间
- 检查一种头盔是否满足特定的安全标准

完成 5 个以上的任务之后，用户将能够习惯站点的布局和操作。当参与者执行这些任务时，有些人喜欢默默地观察，其他人喜欢请求参与者说出他们正在执行的操作。在第二种情况(有时称为大声谈论协议)中，需要确保用户讲出他们每一步操作的意图。通常会获得一些断断续续的语句，但是通过自己首先对一个不同的站点执行这些任务，可以了解期望从他们那里获得的信息。下面是一个最终可能获得的记录副本示例：

(1) “我在为 Julia 查找一辆自行车”

(2) “查看菜单……主页、商店……”

(3) “单击商店……”

(4) “左边显示商标列表，其中很多商标都没有听说过”

(5) “右边的图片分别表明男式自行车、女式自行车、男孩的自行车、女孩的自行车……?”

(6) “单击关于女孩自行车的图片”

(7) “显示年龄，因此单击'first bikes'……”

(8) “查看图片……”

(9) “这一辆自行车看上去很好；单击该图片……”

(10) “不做任何事情……认为它将能够显示更多相关信息……”

(11) “单击'Raleigh Butterfly'”

(12) “那里，那一辆自行车看上去很好。”

如果可以的话，最好进行录音，只要录音不太干扰用户即可。再次提醒，如果用户执行的操作与您认为他们应当执行的操作不同，不要尝试中断他们——毕竟在这个示例中，您了解到用户期望能够单击自行车的图像以查看更多信息，但是目前没有实现该功能。

13.2.9 校对

如果正在设计一个商业站点，通常雇佣一个校对员检查所有的文本会非常有帮助。愚蠢的打字错误可以影响人们对站点的印象，使您和为其开发站点的客户看上去都不专业。

如果您的客户发现在整个站点中四处存在错误，则显得您很粗心——即使客户为站点提供了充满错误的副本。

校对员的工资不会很高，考虑到荣誉问题，他们无疑值得您为其支付工资(这样能够避免可能的尴尬情况，并且潜在地获得一些额外的合同)。

13.3 发布站点

现在您的站点应当已经准备发布给公众，因此可以开始了解如何将其发布到 Web 上。为了完成该任务，需要获得一个域名，找到一些主机托管空间，使用 FTP 软件将站点迁移到新服务器上。在下面的章节中将介绍这些方面的知识。特别是，您将了解在决定由谁驻留您的站点时应当查找什么样的信息。

13.3.1　获得域名

如果正在创建的是个人站点，则不一定需要自己的域名，但是如果为企业创建站点，则最好具有一个域名。域名通常是在站点地址的 www 之后看到的名称的一部分。例如，Wrox 出版社使用域名 wrox.com，而 Amazon 在美国使用域名 amazon.com，在英国使用域名 amazon.co.uk，在德国使用域名 amazon.de 等(但是，您可能已经注意到，有些站点的名称中没有使用 www)。

可以在域名注册公司注册您的域名，目前有很多这样的公司；只需要在习惯使用的搜索引擎中进行搜索，就会获得大量这样的公司。大多数公司选择使用的后缀是.com，但是可以使用其他几种后缀。例如，存在一些国家专用的域后缀(也称为顶级域)，例如用于英国的.co.uk，用于德国的.de，用于澳大利亚的.com.au，用于俄罗斯的.ru。如果站点面向某个国家，则应当选择该国家专用的域。事实上，某些域名(例如.com.au 域)仅能够由具有注册公司的人或者在该国家中按照名称销售产品的公司所购买。也存在一些其他后缀，例如，.me.uk 后缀用于个人站点，.info 后缀用于基于信息的站点，.org 后缀被用于已注册的组织。

注意：

域名不需要具有所有可用的后缀，但是需要确保站点在名称上与不希望关联的某个站点不类似，以防止用户输入错误。例如，您肯定不希望儿童站点具有与成人站点非常类似的 URL。

在注册站点名称之前，首先需要查看它是否可用；所有域名注册公司都应当具有一个表单，利用该表单可以搜索以查看您的域名是否可用。您可能会发现这是一个非常复杂的过程，因为对于.com 域名来说，您能够想像到的大多数域名以及一些最流行的单词(甚至是流行单词的组合)都已经被占用。许多数字也已经被占用。

可以先订购一个域名，即使站点还没有准备好；这种方式有时称为域名停放。只要知道即将创建该站点就可以订购域名(毕竟，您很可能希望在站点的设计中使用 URL，因此在开始设计站点之前需要订购域名)，但是不在该域名上放置任何内容直到已经构建好站点。

很多域名注册公司也提供主机托管，但是订购域名和驻留站点的公司不需要相同；可以让域名注册公司将域名指向主机托管公司的服务器(通常在注册域名的站点上具有一个简单的控制面板，利用它可以控制域名的实际指向位置)。

域名应当能够容易记忆。避免域名过长，否则用户将会发现难以记住它，或者因为太长以至于不想输入它。例如，如果公司的名称是 Sydney Slate Roofing Services Limited，则选择的域名可以是 www.SydneySlate.com，而不是 www.SydneySlateRoofingServicesLimited.com。

当注册域名时，也能够使用它作为您的 e-mail 地址。例如，如果选择域名 www.example.com，则没有得到您的许可的其他人将不能使用 bob@example.com 作为他们的 e-mail 地址(但是发布垃圾邮件的人可以将他们的 e-mail 地址改为如同来源于您的域，方式是简单地改变他们的 e-mail 程序的“from”地址)。

13.3.2 主机托管

为了查看 Web 页面，浏览器将从 Web 服务器处请求页面。Web 服务器是一台特殊计算机，它持续地连接 Internet。

当使用一个域名访问页面时，例如 http://www.example.com/，域名服务器会将域名改为数值。该数值(称为 IP 地址)唯一标识 Web 上的一台计算机，而这台计算机中有放您的 Web 站点。

因此准备将站点发布到 Web 上时，需要在一台 Web 服务器上具有一些空间。很多公司允许您将 Web 站点放置在他们的服务器上，当然需要为该服务付费。这种服务称为 Web 主机托管，因为这些公司为您驻留站点。

在选择许多 ISP 访问 Internet 时，它们会向您提供少量免费的 Web 空间。也有一些其他站点提供免费主机托管(他们的报酬通常是在您的页面加载时显示一些弹出式广告)。对于个人站点，仅需要少量 Web 空间，并且可能会忍受与免费服务一起提供的弹出式广告。但是对于商业站点，最好选择一些付费主机托管——它们仍然可能非常便宜，但是不会提供广告。

1. 选择主机时的关键考虑事项

本章前面已经提到，数以百计的公司提供 Web 主机，选择哪一家公司类似于选择雷区的哪个地方行走一样难。下面是在选择一个站点时需要理解和考虑的关键点(这些关键点的列举顺序是按照它们的字母顺序，而不是按照它们的重要性)：

- 备份：应当检查主机是否对站点执行备份，如果执行备份，则备份的频率是多少。备份只是简单地复制站点，以防止它所驻留的计算机出现问题。虽然正在创建的站点类型可能不需要经常备份，但是最好知道主机托管公司是否备份站点，以防止它的服务器可能出现错误(这将使主机能够修复问题，而不需要获得站点的副本)。当您开发的站点经常改变并且可能会被多个人更新时，则更需要查看备份问题。
- 带宽：这是允许从站点发送的数据量，它可以是每天、每月或每年的速率。如果您的 Web 页面的平均大小是 75KB(包括图像)，每月有 100 个访问者访问您的站点，并且每个访问者查看 10 个页面，则每月将需要至少 75 000kb(或 75MB)带宽。事实上，您将发现主机提供的带宽通常多于该带宽，但这个例子能够帮助您理解如何计算带宽。

  确定需要多少带宽的关键是判断站点能够在多大程度上取得成功。无法预测您的站点将会多流行，如果它被一个流行报纸或杂志提及，则其访问量会急剧提升。关于所需带宽的主要问题是，选择的某种主机托管服务可能会在您的账户所允许的带宽超出限定值之后收取额外费用。应当定期检查是否超出了带宽的限定值，否则月底将会收到高额的账单。如果超出了限定值，也应当确保具有足够的钱支付它，否则将会被停止服务。
- 国家：您可能希望考虑站点应当驻留在哪个国家。最好将站点驻留在期望的大部分顾客所在的国家，因为数据的传播距离更短，从而使用户能够快速访问站点。例如，如果为澳大利亚市场建立站点，则最好将站点驻留在澳大利亚而不是欧洲，因为澳大利亚的访问者能够更快地加载站点页面。但是，实际上很可能看不出驻留在不同地区时具有太大的性能区别。

- 数据中心：很多公司宣称他们具有价值数百万美元的数据中心。这是因为大多数主机托管公司租用大型数据中心中的空间，并在该数据中心中放置他们的服务(这并不意味着您的主机托管公司是具有数百万美元资产的运营公司)。
- 硬盘空间：您将经常看到站点获得的主机空间量以 MB(兆字节)或 GB(吉字节)的形式给出。磁盘空间决定了站点的大小，并且它的总量要大于站点中所有 XHTML、CSS、脚本文件和图像的大小。可以通过简单地查看站点所在的文件夹大小来检查站点的大小(只要没有在该文件夹中放置任何其他文件)。
- e-mail 账户：主机托管公司通常与 Web 主机托管一起提供 e-mail 服务。这里需要考虑两点因素：允许的邮箱大小和允许的邮箱数量。某些主机赋予无限的邮箱数量，但是对所有邮箱的存储空间总量设置限制，因此如果具有 5 个邮箱，并且它们将共享 10MB 的空间，则每个账户仅能够拥有 2MB 的容量。某些主机托管公司允许您指定少量邮箱，但是将允许每个邮箱具有固定的空间量(例如每个邮箱 10MB)。最后，某些主机托管公司允许将分配给域的空间量用于邮箱，从而邮箱的空间量限制等于总存储量限制。
- 共享主机托管与专用主机托管：便宜的 Web 主机托管通常总是以共享主机的方式提供服务。这将意味着您的 Web 站点将与其他很多站点位于相同的物理计算机上。因为较小的站点没有很多访问者，该计算机能够容易地满足驻留多个站点的需求。但是，较大的站点可能一天具有数以千计的访问者或者具有一些大文件(例如音乐下载或许多大型图片)，因此需要额外带宽并且将占用该服务器上更多资源。这也是您的站点开始超出指定的带宽限制并且您的费用开始增长的原因。因此，如果站点变得非常流行，您会发现拥有自己的服务器将会更便宜(或者事实上您的主机可能会坚持要求具有自己的服务器)，该服务器称为专用服务器，因为它仅供您使用。Web 中某些非常流行的站点实际上驻留在多台服务器上——这些站点非常繁忙，以至于一台计算机无法单独处理流量，或者一台计算机需要维护，因此需要具有其他计算机继续服务。银行、大型在线商铺以及跨国公司的站点都会使用这种类型的配置——它们称为负载平衡服务器或者服务器集群。

  请不要因为此处的讨论而推迟购买您的专用或负载平衡服务器。通常，如果您的站点非常流行，以至于需要自己的专用服务器，则应当已经从站点中赚到了足够的金钱来负担额外的费用。
- 统计数据包：每次用户从您的站点请求一个文件时，Web 服务器就会记录用户的某些相关细节信息——例如 IP 地址、浏览器版本、操作系统的语言等。这些信息来源于浏览器的 HTTP 头。统计数据包能够查看包含这些信息的日志文件，并从其中解释一些非常有用的信息。例如，可以看到如下信息：站点为访问者提供了多少页面、用户为了找到站点而在搜索引擎中输入的内容、访问者最常访问的站点页面。这些信息有助于您理解用户在您的站点上执行的操作，并且能够帮助您改进站点并提高它所接收的访问者数量。在本章后面的部分中将介绍关于统计数据包的更多内容。
- 正常运行时间：正常运行时间是指人们能够查看您的站点的时间和您的 Web 服务器的工作时间之间的百分比。您将经常看到 99%的正常运行时间，这意味着在每 100 分钟内平均具有 99 分钟的时间站点可以被访问。但是，这也意味着在 1%的

时间里站点无法使用，也就是 1 年里的 87.6 个小时或 4 天。如果您的站点是收入的主要来源，则应当选择具有更高正常运行时间的站点。

除非正在运营的是非常庞大的公司，否则不值得投资运行自己的服务器，因为很可能需要某人能够管理该服务器并且定期维护。如果决定不需要自己的专用服务器，则很多主机托管公司可以为您管理服务器，并且在需要时为操作系统更新补丁包以修订其中的安全漏洞——这称为托管专用服务器。虽然这种方式仍然很昂贵，但您将发现这比自己雇拥人员管理服务器更为便宜。

2. 利用 FTP 将站点上传到服务器

一旦为 Web 服务器上的空间付款，就需要能够组成 Web 站点的文件上传到该计算机上——该计算机可能位于世界的另一端。最高效的上传方式是使用 FTP。

FTP 代表文件传输协议(File Transfer Protocol)。Internet 使用大量不同的协议发送不同类型的信息。例如，HTTP(Hypertext Transfer Protocol，超文本传输协议)用于传输超文本文件，其中超文本文件更合适的称法是 Web 页面。FTP 协议用于在 Internet 上传输二进制文件，相比于 HTTP，它以更快的速度将整个 Web 站点传输到服务器上。

大多数主机托管提供商实际上要求您使用 FTP 将页面传输到他们的服务器上，这意味着需要使用 FTP 程序(有时称为 FTP 客户端)将文件传输到服务器上。

大多数 FTP 程序具有两个窗口，每一个窗口具有一个文件资源管理器。一个表示您的计算机上的文件和文件夹；另一个窗口表示 Web 服务器上的文件夹。在图 13-7 中可以看到我的计算机上的一些文件和文件夹，在右边可以看到 Web 服务器上的文件和文件夹。

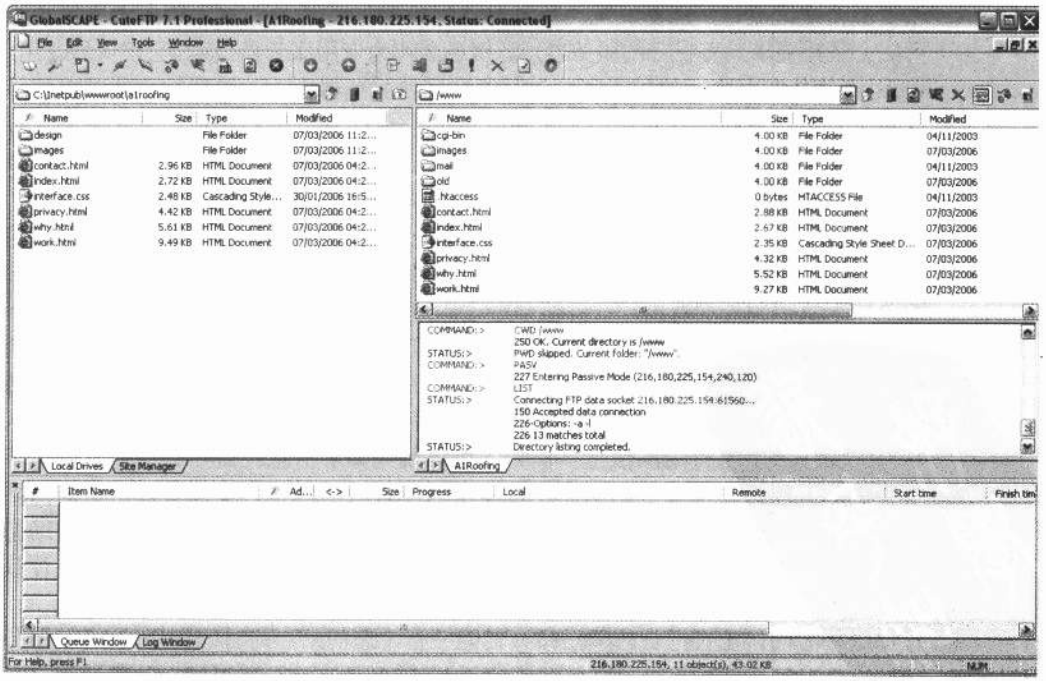


图 13-7

表 13-2 中给出了一些最流行的 FTP 程序。

表 13-2

产 品 名	URL	操 作 系 统
FireFTP	http://fireftp.mozdev.org/	Windows 和 Mac OS X
Cute FTP	www.cuteftp.com/	Windows 和 Mac OS X
FTPX	www.ftpx.com/	Windows
Fetch	www.fetchsoftworks.com/	Mac
Transmit	www.panic.com/transmit/	Mac

每一种程序均稍有不同，但是它们都遵循相似的基本原则。

在主机中注册时，这些主机会发送将站点通过 FTP 上传到服务器上的方式。这包括以下内容：

- 一个 FTP 地址(例如 ftp.example.com)
- 一个 FTP 用户名(通常与您的域的用户名相同)
- 一个 FTP 密码(通常与您的域的密码相同)

图 13-8 中给出了如何在 Mac 上称为 Transmit 的 FTP 程序中输入这些内容。

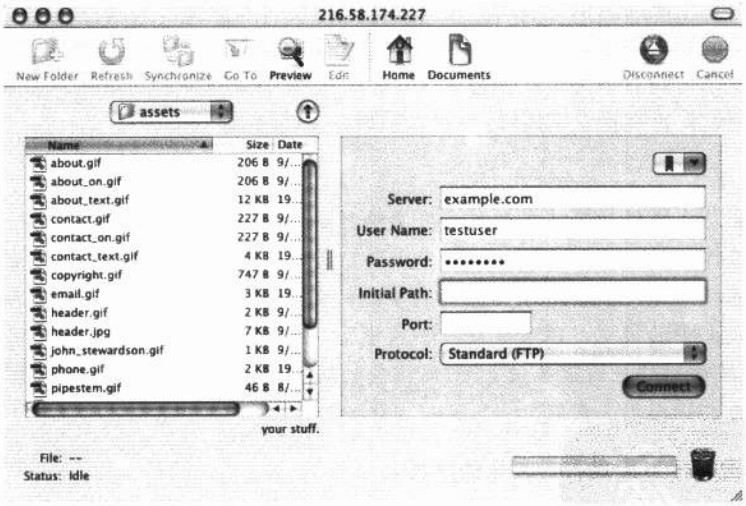


图 13-8

可以在购买某种 FTP 软件之前下载该软件的试用版本，以发现最适合使用哪种软件。大多数这样的软件具有非常相似的图形用户界面。

13.3.3　搜索引擎策略

现在您的站点已经位于服务器上，您无疑希望人们访问该站点。获得站点访问者的一种最佳方式是确保用户在搜索引擎中输入与站点内容相关的单词之后，他们能够找到您的

站点。最好是他们能够在返回的前 10 个值中就找到该站点。

1. meta 标签之外的因素

本章前面介绍过搜索引擎如何使用<meta>标签的 description 特性和 keywords 特性帮助搜索站点，但是搜索引擎索引 Web 中所有站点的处理方式正变得越来越复杂。因此，需要持续增强搜索引擎策略，而不仅是利用添加在<meta>标签中的内容。

注意：

利用各种方法停留在搜索引擎排名的顶部是即将介绍的内容，并且它应当是标准市场营销实践的组成部分。最好经常查看您的搜索引擎排名，至少是一个季度一次——或者可能一个月一次。

不应当只依赖于搜索引擎站点找到您的站点并索引它。例如，类似于 dir.yahoo.com 和 dmoz.org 的站点是基于类别的 Internet 目录站点，必须遍历这些类别来找到所需的站点。这种手动分类站点要求将站点的 URL 提交给他们；然后可能需要几周到几个月的时间您的站点才会出现在他们的站点之中。

某些站点会对列举在它们站点中的站点进行收费，如果是这种情况，则应当仔细考虑是否值得为此付费。虽然一些大型的流行站点可能会值得为此付费，但是除非了解该站点，否则您将发现它不能提供很多有用的参考。下一节中将更多地讨论这方面的内容。

一旦手动通知了搜索引擎，则在再次提交 URL 之前应当至少等待一个月。如果多次提交，则搜索引擎可能会认为是在恶意提交，因此完全不会列举您的站点。如果您的站点改动比较大，则值得再次提交它(但是不要太频繁)，以便标识改动。

注意：

我通常会忽略保证能够将我的站点提交给数以千计 Internet 目录的程序，并且宁愿自己执行一些工作。花费几周中每天的一个小时，首先将站点手动提交给主要的搜索引擎，然后查找其他相关 Web 站点，请求他们列举我的新站点。这样做是因为有些程序会向 Web 搜索引擎生成太多的提交，以至于这些搜索引擎认为这是垃圾信息(并且因此忽略它们)。请记住，如果提供的结果听上去过于好(保证您的站点排在数以千计的搜索引擎的前 10 名)，则可能会因为太好而不真实！

2. 通过设计使页面具有最大化的排名

使用程序自动索引站点的搜索引擎正在使用越来越复杂的规则来确定谁将在 Web 页面中获得最高的排名(最前面的排名)结果。当设计站点时，可以考虑利用下面的一些关键点来确保站点具有尽可能高的排名：

- 页面的标题是站点中最重要的单词，并且它们是被索引的最重要的内容之一。因此，避免在标题中仅包含类似“Home Page”这样的单词，而是尽量使用描述性的标题，例如“Wrox Press—Computer Programming Book Publishers”。然后在某些特定页面中，标题可以改为“XHTML Programming Books, learn to code and build web sites”。如果用户在搜索引擎中输入的单词可以在标题中找到，则搜索引擎将

认为该站点更加相关。但是标题的长度不要长于一句话，否则程序将了解到您在尝试欺骗它，并将这一点统计为对您不利的因素。

- 大多数搜索引擎会浏览页面的文本内容并且将索引它们。页面最前面的一些单词通常被认为是最相关的单词。因此，应当策略性地将站点的关键字放置在页面起始部分附近以及标题中。并且，可以在这些位置中的文本扩展关键字的列表。
- 如果用户搜索的关键字在页面中的出现频率高于其他单词，则它们被认为是更加相关的单词。但是，不要使这些单词出现得太频繁——否则搜索引擎会将这一点统计为对您不利的因素。
- 如果站点使用图像而非文本，则站点仅能索引图像的 alt 文本；因此确保图像所传递的任何信息同时也以文本方式传递。
- 如果尝试通过在文本中重复关键字来欺骗搜索引擎(并且关键字的颜色与背景色相同，以便重复的文本对用户不可见)，则搜索引擎会因为这一点惩罚您。
- 如果关键字与站点的主题或内容无关，则搜索引擎也会将这一点统计为对您不利的因素。
- 链接到您的站点的其他站点越多越好。如果您的站点被很多其他站点链接，则某些搜索引擎将赋予您的站点更高的优先级。但是需要注意，它们也将考虑哪些站点链接您的站点。这些站点应当与您的业务相关——搜索引擎不会将二手车销售商到宠物店的链接作为相关链接。
- 越多的用户通过单击搜索引擎中的链接访问您的站点，则您的排名将会越高。虽然标题、文本中的关键字、<meta>标签和链接数量能够帮助您的站点出现在搜索引擎的顶部，但是如果没有人单击这些链接来访问您的站点，则您的排名将会很快跌落。

虽然提升搜索引擎的排名是长期的事情，但是持续地关注排名将有助于您的站点越来越好。请查看本章稍后的“其他 Web 市场营销策略”一节，了解一些额外的市场营销策略，以提高站点流量。但是，首先将介绍在不希望页面被索引时应当执行的操作。

3. robots.txt

在一些 Web 站点中，存在一些不希望被索引的页面——例如管理页面和测试页面。为了防止这些页面被搜索引擎索引，可以在站点中包含一个简单的文本文件 robots.txt(可以利用类似于 Windows 操作系统中的 Notepad 或 Mac 操作系统中的 TextEdit 这样的简单文本编辑器编写该文件)。

robots.txt 文件中可以包含一些简单命令，这些命令将防止站点的某些部分被 Web 机器人(用于索引站点的小程序)索引。Web 机器人通常会被编程以读取 robots.txt 文件。

站点中最多只能有一个 robots.txt 文件，并且它应当放置在 Web 服务器根文件夹中的 htdocs 文件夹中。某些 Web 主机托管公司将为用户创建 htdocs 文件夹；而有些公司则要求用户自己创建该文件夹。

这个简单文本文件的第一行应当是：

```
USER_AGENT: web_crawler_name
```

假设希望所有的 Web 机器人服从这些规则，则可以简单地使用一个星号取代任何 Web 机器人的名称——星号有时也称为通配符，它指明所有的 Web 机器人应当服从这些规则。

接下来，可以指定不允许 Web 机器人索引的文件夹(这也是良好地组织站点的重要性的另一种原因)，方式是使用 DISALLOW 命令。可以对每一个不希望被索引的文件夹重复使用这条命令：

```
USER-AGENT: *
DISALLOW: /admin/
DISALLOW: /scripts/
```

该命令指示 Web 机器人不尝试索引管理或者脚本文件夹(以及它们的任何子文件夹)。

虽然没有强求 Web 机器人服从这个文件中的命令，但是不索引人们不希望显示的页面(通常是因为这些页面不允许用户执行任何操作)对它们有利，因此主要的搜索引擎通常服从这些规则。

13.3.4 其他 Web 市场营销策略

搜索引擎只是让人们访问您的站点的一种方式——在其他类型的站点市场营销上面花费一定的时间也是完全值得的，无论是 Internet 上的站点还是非 Internet 上的站点。可以利用下面的一些策略吸引访问者访问您的站点：

- 搜索其他与您的行业相关的 Web 站点。其中有些站点可能具有一些到感兴趣站点的链接，可以请求将您的站点加入这些链接。
- 很多行业具有行业特有的目录 Web 站点，这些站点列举了该领域的产品和服务。但是，为了在这类站点上做广告，必须付费给它们。必须确定带来的流量是否值得为其投资。应当经常询问行业内的一些人是否使用了这类站点，这样能够估算是否可以从这类站点中获得许多通过介绍而来的访问者。
- 许多站点提供对等链接；他们链接到您的站点，但回报是您必须链接到他们的站点。这是一种每个人都可以用于提高流量的方式。但是，必须确保不要当其他人将您的站点的链接放置到很少有人问津的页面时，您却将他们的站点的链接放置到您的主页面上——毕竟使用的是对等链接。
- 使用一些搜索引擎搜索相关的公司，并且找到谁链接到这些公司——链接到相关站点的站点也可以链接到您的站点(前提是您提出链接请求)。您可能会找到一些以前没有听说过的站点，但是他们很愿意链接到您的站点。
- 您可以在站点上提供一些按钮或者广告栏，以便人们能够将它们集成到其站点上。相比于商业站点，社区站点更容易被其他人添加链接。业余爱好者通常愿意链接到感兴趣的站点，特别是如果您提供了到他们站点的链接，则他们更愿意链接到您的站点。并且如果链接的外观很美观，则他们更可能希望添加这些链接。
- 查看按点击付费(Pay Per Click，PPC)的广告。Google 具有称为 AdWords 的 PPC 系统，您可以为该系统指定一些关键字，当用户使用这些关键字搜索时，您的广告将出现在页面的右边。这些广告也可能会出现在其他一些特定兴趣的站点上。AdWords 的工作方式非常灵活，仅当用户单击广告时才会为其付费。广告的排名

基于准备为每次单击支付的费用以及实际单击广告的人数。如果用户没有单击您的广告，则它的排名将下降，无论您支付了多少钱(毕竟，Google 坚持赚更多的钱，如果 10 个人单击了一个便宜的链接，则比起一个人单击一个稍微昂贵的链接，前者可以让更多的用户从它们的服务中获得价值)。一般来说，对于为站点产生流量，这种方式非常节约成本。Yahoo 提供了称为 Overture 的相似服务，Microsoft 也有称为 adCenter 的系统。

- Web 中存在大量其他形式的付费广告，可以采用它们。许多站点采用横幅类型的广告，并且许多站点允许您付费以便列举您的站点。必须判断这些站点是否物有所值。记住，许多 Web 用户会忽略广告，他们只浏览页面以查找实际需要的内容——因此如果打算创建一个广告栏，需要确保它具有足够的吸引力，并且人们将愿意单击它，从而使其值得在其他站点上花费广告费。
- 如果在您的行业范围内存在一些新闻组、公告栏或论坛，可以在其中回答问题，并且将您的 Web 站点地址作为您的姓名下方的签名。但是注意，仅当在能够帮助其他人的情况下才这样做——在新闻组中只发布与其主题相关的帖子；否则您可能会使人们产生反感，从而达不到吸引他们访问您的站点的目的。
- 如果您的站点定期改变内容，则考虑在站点中添加时事通讯功能，以便人们签约以接收定期更新。本章后面将更详细地讨论这方面的知识，这是一种让人们了解站点的更新并知道新内容的优秀方式。
- 当然，不能只使用 Web 来推销站点；优秀的站点应当通过人们的交口相传来产生流量。也可以使用打印的传单，在相关杂志上投放广告，将您的站点地址放置在信纸上方或者汽车旁边。甚至可以找到与您站点涉及领域相关的会议或者事件，并且将它作为使行业范围内的人们更加了解您的工作的方式。

13.3.5　统计分析

如果您的主机托管提供商的服务器上具有统计分析数据包，则一旦将站点投入使用您将能够从中找到关于您的访问者的很多有用信息。这些数据包分析 Web 服务器上的日志文件；日志文件中包含服务器发送的文件以及文件接收方等方面的信息。

站点分析中使用的一些术语会令人迷惑；例如，您可能听说过一个站点获得了 10 000 次命中，但是该术语非常误导人。术语“命中”是指从站点下载的文件数量——并且图像也与 XHTML 页面一样被计数为一个文件，因此具有 9 幅图像的单个 Web 页面将等同于 10 次命中(某些图形密集的页面可能具有超过 30 幅的图像)。因此，了解页面的查看数量比命中数更有用，因为这表示站点中已经查看的页面数量。

您也可能会遇到术语“访问数”。但应当注意的是，不同的统计数据包计算访问数的方式不同。某些数据包将使用同一个 IP 地址的多个人作为同一个访问者——因此如果有 10 个人工作在同一个建筑中，并且他们同时查看相同的站点，则某些数据包很可能将认为只有一个人(而不是 10 个人)访问站点。不同的数据包通常也采用不同的时间长度来计算访问数；某些数据包记住一个 IP 地址一整天，因此如果相同的人在早上来到这个站点，然后在晚上再次访问该站点，则仅计数为一次访问。而有些数据包仅记住一个 IP 地址 15 分钟。

您也会发现某些广告商将询问您的站点每月接收到的唯一访问者数量——不同的统计数据包计数唯一用户的方式不同，因此这个数字也会产生误导。

大多数统计数据包将会提供比此处提到的更多的信息。例如，某些数据包通常会告诉您人们到达您的站点的方式——人们从哪些页面或站点到达您的站点，以及从每个页面或站点到达您的站点的次数。这些信息能够帮助您了解人们如何找到并到达您的站点，从而将帮助您确定推广站点的合适位置。

统计数据包也会频繁地告诉您人们通过哪些术语搜索到您的站点——因此能够了解用户在搜索引擎中输入什么关键字来找到您的站点，然后可以通过在页面中增加这些单词的出现频率来提高您的搜索引擎排名。图 13-9 给出了用于查找关于打印设备和服务的站点时使用的一些术语。

您可能会找到的一些其他信息包括：

- 用户到达您的站点的哪个页面以及从哪个页面离开。这些信息非常有用，因为这样能够了解人们到达您的站点的目的以及他们从何处离开。如果某个页面使很多人离开，可以通过改进该页面的设计来尝试使访问者在您的站点中停留更长时间。
- 用户来自于哪个国家
- 用户正在使用的浏览器
- 用户的操作系统被设置为什么语言以及他们可能说哪种语言
- 每个用户在站点中花费多长时间

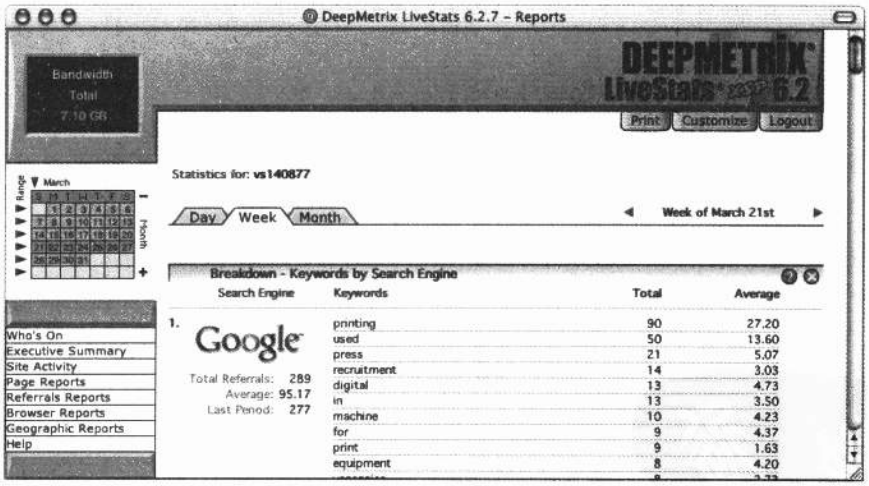


图 13-9

注意：

所有这些数字都是近似的，并且不同的报告数据包给出的数字可能不同，但是这些数字毫无疑问是非常有用的工具，它们可用于分析人们如何找到您的站点以及人们在您的站点中执行的操作。

除了使用 Web 服务器中的内置统计系统，还可以使用一些利用 JavaScript 调用服务器

中的某个文件并创建分析信息的系统。一种非常流行的系统(也是免费的系统)是由 Google 提供的 Google Analytics。为了运行这个系统，只需要将一段代码放置在每个页面的底部；然后 Google 将创建关于访问者的各种类型的报告，从而能够非常好地补充您的 Web 主机中提供的统计数据包。

13.3.6　版本控制

有时您可能希望改变站点。本章前面已经提及，不应该在正在运行的服务器上执行改动，而是必须具有站点的另外一个副本，在使改动生效之前，可以使用该副本在您的本地计算机和 Web 服务器上进行测试。

当处理任何类型的文件时，很容易只操作文档的一个版本并保存执行的改动。但是，这会带来很多问题：

- 可能会覆盖保存某个文件，但是实际上并不打算这样操作，或者存在某个错误以使您希望返回到初始文件。
- 可能打开一个文件，执行一些改动并保存它。然后，当您正在编辑自己的文件版本时，某位同事可能碰巧打开同一个文件并执行改动，然后在您保存之后保存他的版本——这将覆盖您执行的所有改动。
- 您可能为某个客户设计站点，并且决定希望返回到设计的一个早期版本或者某些较早的内容。
- 您可能需要站点在早期时包含的某些文件的副本——但是如果没有这些文件，您将无法实现愿望。

当多个人工作于相同的文件或者您正在更新自己的文件时，最好在保存具有相同名称的新文档之前提出一种命名约定以保存所有较老的文件。例如，在覆盖保存之前，可以简单地在文件名后添加日期和时间。如果希望改变主页面 index.html，可以利用改动它的日期来保存一个副本。这种方法的优点是能够知道什么时候执行最近的改动。

这种方法确实占用更多的硬盘空间，但是如果定期归档站点并删除一些不会再次使用的较老文件，则通常会清理出它们占用的空间并维护一组可管理的文件。

也可以使用自己的<meta>标签来指明一个版本以及最后更新该文件的人的姓名。您可能会记得在本章开始时提及，可以使用任何喜欢的内容作为<meta>标签的 name 属性值；这是应用该功能的优秀示例。例如，下面的<meta>标签指明了最近修改日期以及谁执行最近的改动：

```
<meta type="last-modified" content="16-04-04:12:34:00 GMT" />
<meta type="last-changes-by" content="Jon Duckett" />
```

注意：

也可以使用<ins>元素和<del>元素，但是在简单地更新一个站点时不能过多使用它们——它们对于跟踪文档的版本来说非常有用。

当改动其他人编写的代码时，最好在代码中添加一些注释。例如，如果正在处理一个站点并且希望添加一个新脚本，则可以利用如下方式完成该操作：

```
<!-- start of new section added 12/12/03 by Bob Stewart -->
<b><a href="specials.html">Click here for special offers on end of stock
items.</a>
<!-- end of new section added 12/12/03 by Bob Stewart -->
```

对于较大的改动来说，您很可能不会这样操作；但是对于较小的改动(特别是脚本或者编程语言中的改动)，注释将有助于其他人在后面返回站点并查看执行什么样的改动。

您可以购买软件来处理版本控制。这种软件允许您登记文件，如同它们是书库中的书籍一样——这就能够阻止两个人同时操作相同的文件。某些软件可能会非常昂贵，但是也有一些免费工具，例如 CVS(Concurrent Versions System，并发版本系统)，可以在 www.cvshome.org/ 处下载该工具。

如果使用 Macromedia Dreamweaver 软件创建站点，则可以利用一个函数指明是否一次仅有一个人可以使用某个文件。这就可以在处理某个页面时阻止其他人打开该文件，然后覆盖保存您自从第二个人打开该文件以来已经执行并保存的改动。

13.4 下一步执行的操作

您已经学习了关于 XHTML 和 CSS 的所有知识，在学习 JavaScript 方面有了一个良好的开始，并且学习了如何将站点发布到 Web 中。您可能惊讶于为什么本节的标题是“下一步执行的操作”，本节将介绍如下两方面的主题：

- 一些工具，可以使用它们在站点中添加一些强大的功能，并且使用您已经掌握的知识
- 下一步适合学习的技术

因此，本节的第一部分是介绍 Web 上提供的一些服务，可以使用它们增强站点。您将学习如何创建博客，如何在站点中添加讨论组或论坛，以及如何添加搜索功能。虽然这些方面听上去很复杂——并且它们无疑是高级的功能——但是它们的实现非常容易，您将看到它们如何为站点提供强大的、令人印象深刻的功能。

非常类似于 Hotmail 在 Web 上提供的 e-mail 功能，使用其他公司的服务器和代码实现其中的大多数服务——您所要做的只是定制它们，以使其看上去是您的站点的一部分。

13.4.1 博客

博客是单词 webblog 的缩写。博客最初被设计为用于在个人 Web 站点中添加在线日志或日记。博客的思想是，允许用户方便地在他们的 Web 站点中添加新条目或帖子，而不需要手动重新编码页面(通常称为“单击发布”)。用户到达制作博客的公司的 Web 站点，在一个表单中填写帖子信息，然后该条目将出现在 Web 站点中。

帖子信息在站点上的出现顺序按照日期排序，虽然它们通常用于在线日记或者日志，但是已经广泛用于其他目的，例如作为人们添加新闻、关于共同感兴趣主题的帖子、链接等方面的方式。

事实上，虽然博客首先作为人们利用 Web 的剩余价值共享思想的方式，但是它们很快出现在公司的内部网中(作为共享信息的方式)，并且作为新闻功能出现在公共的 Web 站点中(而不只是用作日记)。

很多不同的公司和 Web 站点提供在您的 Web 站点中添加博客的工具，其中两个较流行的 Web 站点是：

- www.blogger.com/
- www.movabletype.org/

这两个站点提供的工具能够用于在博客中添加帖子，而不需要在每次希望写入一些新内容时都手动更新页面。并且，不需要在您的服务器上安装任何软件和脚本(但是利用软件和脚本可以实现该功能)。另外，这些工具看上去如同是您的站点的一部分(而不是用于制作博客的公司的一部分)，并且它们也可以实现其他功能——例如允许用户发表关于您的帖子的评论，或者添加关键字以帮助人们找到相关的帖子。

13.4.2　讨论板或论坛

讨论板允许用户发表问题或者评论，然后其他用户可以回复这些问题。讨论板是在您的站点中添加一种团体感受的优秀方式。访问者也可以自己提供新内容，而不需要您自己添加这些内容，这样能够吸引访问者定期返回您的站点。例如，如果您正在运行一个关于某种类型的汽车的站点，则可以利用一个讨论组解决技术方面的问题，以及回答该型号汽车的修理问题，并且可以利用另外一个论坛让用户指出他们什么时候购买或销售该类汽车。

关于讨论板的重要事情之一是，如果您的站点以回答问题著称，则人们在遇到问题时就会访问该站点。您可能在最初启动社区时必须自己回答所有的问题，但是随后如果幸运的话，其他成员将会开始添加他们的见解。

与博客一样，一些公司创建相应的软件并在他们的服务器上提供该软件，从而在功能方面如同您的站点上具有一个讨论组(即使是运行在他们的服务器上)。其中最佳的讨论板软件位于 www.ezboard.com/。与博客一样，通常可以利用 CSS 定制讨论板的外观。

但是应当注意，您将为人们在您的讨论板上编写的内容担负法律责任。如果某个人在您的站点的讨论板或论坛中发表具有法律问题的内容，您将被认为是这些内容在 Web 上的发布者——即使您的观点与发表这些内容的人不同。

某些讨论板解决这种问题的方式是允许拥有者审核每一个帖子(在允许该帖子出现在站点中之前阅读它)；而某些讨论板只是定期检查站点中是否具有存在法律问题的内容，并且尽可能地删除他们认为存在法律问题的帖子。

13.4.3　添加搜索实用程序

在第 10 章中提及，您可能希望在站点中添加搜索实用程序。在第 10 章中介绍了可以在站点中添加 Google 搜索实用程序，并且甚至可以在 www.google.com/coop/cse/中定制搜索功能。提供免费的可定制搜索服务并且最多能够搜索 500 个页面的另一个站点是位于 www.Atomz.com/中的 Atomz Express 服务(可能需要跟踪用于测试服务的链接)。

在站点中添加的搜索工具，这将意味着用户能够找到他们希望找到的内容，而不会因为找不到所需的内容而简单地放弃并离开；毕竟如果所需的信息很难访问，许多访问者不会在站点停留很长时间并浏览很多页面。

利用 Google 和 Atomz 的服务，将会获得一段表单代码，这段代码允许用户将查询发送给相应公司的 Web 站点。然后公司的服务器返回一个页面给用户，该页面中包含搜索的结果。这两种服务都允许用户为页面创建定制的标题以使其包含商标，尽管结果由公司的服务器生成。

13.5 其他技术简介

本节将介绍一些其他技术，主要介绍它们的功能，以及如何在 Web 站点中利用它们。在实际体验过本书中介绍的所有知识后，希望这些内容能够帮助您决定下一步希望学习的技术。

13.5.1 服务器端 Web 编程：ASP.NET 和 PHP

本书已经介绍了编程语言(例如 JavaScript)能够在浏览器中执行哪些操作方面的非常基础的知识，但是当编程语言用于 Web 服务器中时，它将变得更加强大。

当希望从站点的访问者处收集信息并为该访问者返回一个自定义页面时，您将希望了解服务器端的编程。下面是关于不同的用户可能需要不同页面的一些应用程序示例：

- 搜索站点中的内容：用户在表单中输入希望搜索的术语，该表单将被发送给服务器中的某个应用程序。然后该应用程序创建一个包含用户所需结果的页面。
- 检查列车时刻：用户输入他的乘车起始点和目的点以及首选的乘车时刻。然后应用程序创建一个包含所请求的旅行时间表的页面。
- 在线购物：用户浏览某类产品并选择他们想要的产品。他们的选择通常反应在每个页面上显示的购物篮中。在选择想要的产品之后，他们提供付款详情和联系/邮寄详情。同时，运营该商铺的人很可能具有一个基于浏览器的界面，以便在站点中添加新产品(而不是必须单独创建每个新页面并链接到该页面)。
- 讨论板和论坛：在本章的“讨论板或论坛”一节中已经介绍过相应的示例，它们依赖于其他公司的服务器端编程和代码来处理所有帖子。

术语“服务器端应用程序”有时可以仅是一个页面，该页面中包含了在服务器中执行的脚本。但是，它也可以非常复杂；它可能由数以百计的代码页面组成，这些代码将使用数据库、组件甚至运行在服务器中的其他程序。这些应用程序的复杂性通常取决于它们具有的功能。

事实上，大多数内容定期改动的站点通常使用一种服务器端编程语言，因为站点的内容将位于数据库中。本章稍后的“内容管理”一节中将介绍关于这方面的更多知识。

13.5.2 选择服务器端语言

可以利用几种不同类型的服务器端编程语言和环境，例如 ASP.NET 和 PHP，它们提供非常相似的功能。一般来说(尽管这些规则也存在一些例外情况)：

- ASP.NET 运行在 Microsoft IIS 和 Windows 服务器中。
- PHP 和 JSP 运行在 UNIX 服务器中。

可以在安装正确软件的台式计算机中编写这些页面，但是需要将已经完成的 Web 站点驻留在 Web 服务器中。

注意：

使用脚本在服务器上创建的第一种应用程序称为 CGI 脚本。仍然会在某些应用程序的 URL 中看到 CGI 或 CGI-bin。但是，此处讨论的语言将具有更广泛的需求并且功能更强大。

关于选择哪种语言，不同的开发人员将具有不同的观点。但是，大多数人学习一种语言和环境并坚持使用它(尽管在已经理解一种编程语言和环境并知道服务器端脚本语言能够执行哪些操作之后，在某种程度上学习第二种语言和环境会非常简单)。

如果为了获得一份工作而学习技术，则最好经常查看关于工作的广告。您将能够跟踪所需要的技术，并且能够及时看到新出现的技术(如果经常关注的话)——开始时可能仅有一到两处位置提到这些新技术，然后将有越来越多的位置提到它们。关于工作的广告因此是应当考虑学习的技术的晴雨表(并且如果您的老板发现您正在查看关于工作的广告，则可以托辞说您正在研究哪些技术将在未来更流行)。

13.5.3　内容管理

许多站点的关键方面之一是内容管理系统，这实际上是一种功能的流行名称，该功能允许您方便地更新 Web 站点中的内容，而不需要为新的文章、帖子或者销售的产品创建新页面。

内容管理系统通常基于关系数据库。关系数据库包含了一个或多个表，每一个表类似于一个电子数据表。图 13-10 给出了一个用于音乐站点中的数据库。

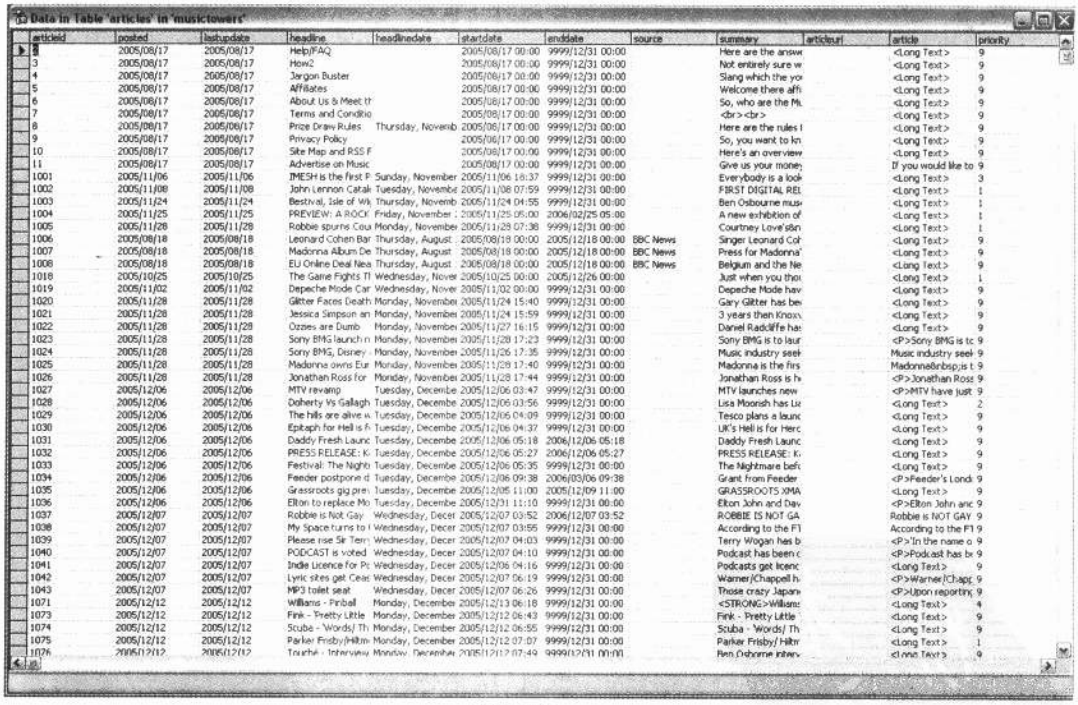


图 13-10

在该表中存在多行，每一行包含了一种不同的广告详情。每一列中包含了关于该行中文章的不同信息：

- articleid 是一个数值，用于唯一标识系统中的每一篇文章。
- posted 是该文章发表的日期。
- lastupdate 是文章最近更新的日期。

- headline 是文章的头条新闻。
- headlinedate 是写作文章的日期。
- startdate 是文章发布的日期。
- enddate 是文章停止在站点上出现的日期(其中多篇文章的日期设置为 9999 年 12 月 31 日——因此如果站点能够运行到这一时间，则管理员必须在该日期执行一些工作，但是在此之前，文章将保持发布状态)。

这个表实际上包含更多字段，但是这可以帮助您了解如何存储信息。当用户到达使用这个数据库的站点时，他们将浏览类别以找到感兴趣的内容。站点没有为每一篇文章利用一个页面包含它的细节，而是仅利用一个页面显示所有新的文章，该页面称为 **article.aspx**。这类似于用于所有文章的模板，并且将标题、头条新闻日期和文章名称添加在每一个文章页面的相同位置。图 13-11 中给出了一篇文章的示例。

图 13-11

请查看这个广告的 URL 结构；这是基于模板系统的工作方式的关键：

```
http://www.musictowers.com/news/features/article.aspx?a=1496
```

该 URL 请求 article.aspx 页面，并且在请求该页面时也请求文章标识符 1496。该标识符对应于图 13-11 中表的第一列中的数值。然后这篇文章的所有细节将被放置在模板中。因此，页面中的文本“Thursday December 6, 2007”对应于数据库中的 headlinedate 字段。

这种方法使许多作者能够更新相同的站点，而不需要知道如何为每个页面编写代码。相反，他们能够登录一个简单的管理工具，并且可以利用一个简单的表单提交文章。图 13-12 中给出了一个页面，该页面允许用户输入新文章。

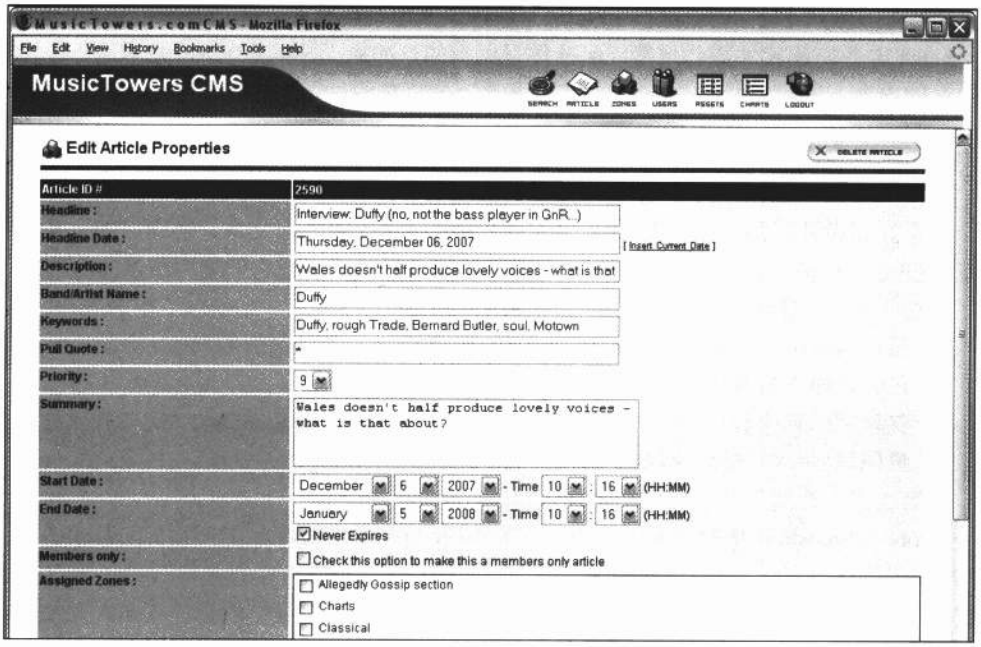


图 13-12

许多不同类型的站点都使用这种在数据库中存储内容的方法。例如，一些拍卖站点(例如 eBay)在数据库的每一行中存储一件销售的商品；同样，电子商务商铺通常将产品的详情存储在一个数据库中，并且每一件产品存储在数据库表的一行中。当这些站点为每一篇文章或每一件产品使用数据库表的一行时，可以利用 XHTML 表单添加新文章或产品(而不需要手动编写页面)，并且内容页面可以列举所有的文章或产品，而不需要在每次添加新文章或商品时都修改该页面。

13.5.4　Flash

使用一种特定的软件(称为 Flash)编写 Flash 文件。用户需要在他们的计算机上安装 Flash 插件(即 Flash Player)以查看 Flash 文件，但是多种来源的统计表明，超过 90%的连接

Web 的计算机已经安装了该插件，并且它非常流行。

Flash 首先作为在 Web 上创建动画的方式——从卡通动画到动采用画方式的徽标或文本。Flash 是一种功能非常强大的工具，可以在如下站点中看到关于它的很多示例：

- www.adobe.com/products/flash/
- www.flashkit.com/gallery/

Falsh 也逐渐用于在 Web 站点中显示音频和视频内容，例如 YouTube 利用 Flash 播放它们的视频。

很少有站点需要完全利用 Flash 进行设计；经常是页面的某些部分采用 Flash 创建(例如横幅广告和动画)。一种原因是利用 XHTML 开发站点更快速，而另一种原因是相比于使用 XHTML 来说，较少的人拥有将 Flash 与数据库很好地集成在一起的技能。

Flash 电影创建软件需要付费，但插件是免费的。如果无法确定 Flash 是否是下一步将要学习的内容，则可以从 Adobe 的站点处下载一个免费的试用版本。

13.5.5 学习图形程序包

如果计划设计页面并编写页面代码，则学习如何正确地处理文本、插图、照片和图像就非常重要。外观一般的站点和看上去美观的站点之间的区别是它们使用图形的方式。

存在两种您可能希望学习的图形程序包：

- 照片编辑和处理程序包，例如 Adobe Photoshop 或它的精简版本 Photoshop Elements。它们处理位图图形。
- 矢量艺术程序包，例如 Adobe Illustrator 或 Macromedia Freehand。这些软件处理矢量图形(利用坐标创建的线条画)，然后对它们填充颜色。

在第 3 章中介绍了关于位图图形和矢量图形之间的区别。

Adobe Photoshop 是到目前为止用于开发 Web 图形的最流行的图形软件。如果查看招聘 Web 设计人员的工作广告，将会发现掌握 Photoshop 软件通常是先决条件。Photoshop 不仅能够用于处理照片，而且可以用于创建文本和徽标(但是有经验的设计人员在从头开始创建徽标和图表时通常喜欢使用矢量图形程序)。

Photoshop 是一种非常有价值的工具，因为它不仅能够用于编辑照片，而且能够创建各种类型的图像，例如导航图像和徽标。然后采用这些图像并创建用于 Web 的优化版本，该版本具有较小文件尺寸以便能够较快下载。

当使用 Photoshop 时，可以创建一幅由许多层组成的图像——每一层类似于一片透明薄膜，它们重叠在第一幅图像上，并且能够在图像上方执行改动。

熟悉了照片程序包之后，您可能希望学习矢量图像程序包，特别是在计划创建许多徽标或者图表时。如果处理的是照片，则很少使用矢量程序包，但是它们在处理基于线条的工作方面非常有用。根据其特性，矢量图形的可伸缩性非常好，通常采用矢量格式创建徽标，因为它们允许将图像放大到非常庞大(用于海报)，或者缩小到非常小(用作小的 Web 图形)。相反，如果将位图图像放大到非常大，则将出现颗粒效果——可以看到组成图像的所有像素。

当然，也可以学习一些其他技术，但是本节中介绍的技术将为您的 Web 开发职业生涯

提供下一个逻辑步骤。如果希望更多地处理图形，建议您首先学习 Photoshop 或 Flash；而如果希望更多地编程，可以首先学习一种服务器端的编程语言。

13.6　本章小结

本章中介绍了如何准备 Web 站点。首先介绍<meta>标签，可以使用它添加关于文档的内容(例如作者、过期日期或者默认的脚本语言)——因此该元素的名称是<meta>标签；它们包含了关于文档的信息，而不是文档本身的一部分。

本章然后介绍不同类型的测试。在将站点放置到服务器中之前，或者在将站点放置在服务器中之后但在希望人们查看它之前，需要执行测试。这些测试包括：验证页面(以确保编写的标记符合相关规范以及您遵守相应的规则)；检查链接以确保所有的链接都有效并且不指向错误的位置；检查您的站点满足可访问性指导原则。

本章接下来介绍了关于选择主机的潜在雷区，您可以将 Web 站点放置在该主机的 Web 服务器上。主机市场灵活多变，因此难于掌握，但是非常值得多查看一些主机，而不是采用遇到的第一个主机。新的主机通常具有更多的存储量、更大的带宽、更大的邮箱，并且较新的功能层出不穷，因此需要货比三家。

一旦站点运行之后，您将希望人们查看它。吸引新的访问者的主要方式之一是使用一些组合技术，例如仔细选择页面中的标题、关键字和内容，并且手动将它们提交给一些站点。这是一个持续的过程，需要很大的耐心。当然，Web 不是推广站点的唯一方式——存在大量其他可用于吸引访问者的方式。

可以利用统计数据包来获得关于访问者的有价值的信息，这些统计数据包分析您的日志文件，能够获得人们如何到达您的站点、他们查看了多少个页面、他们为了到达您的站点而在搜索引擎中搜索了什么术语等信息。

本章还介绍了版本控制，以便当更新站点时不会丢失重要文件，或者不会让其他人覆盖保存您的工作。这里的关键字是安全，您需要保存每次改动的副本，至少直到您已经完成了该工作。然后，可以归档站点的该版本并删除较老的文件。

本章最后一部分介绍了下一步能够对站点执行的操作。其中讨论了一些服务，例如博客、讨论板、搜索功能，它们已经被一些公司开发，可以将这些服务集成到您的站点中。如果您对编程感兴趣，应当考虑学习一种服务器端编程语言，例如 ASP.NET 或 PHP。另外，如果您对可视化的外观或站点的设计更感兴趣，应当考虑学习一种图形软件(例如 Adobe Photoshop)以及某种动画软件(例如 Flash)。

本书已经介绍了很多内容，理解它们的最佳方式是实践它们并构建一些站点。或许可以创建一个关于个人爱好和兴趣的站点，或者可以为朋友创建一个站点，用于运营他们拥有的小型公司。

记住，如果喜欢某些站点中的样式(例如布局或使用字体的大小和类型)，可以简单地利用浏览器上的“查看”菜单并选择某个选项以显示该页面的源代码。虽然不能抄袭其他人的设计或布局，但是可以从查看其他人构建站点的方式中学习到很多知识。但是请记住，这些站点可能不是使用 XHTML 编写的；许多页面是利用早期版本的 HTML 构建的。HTML

对于编写页面的方式的要求并不是很严格，并且许多代码编写人员没有很好地理解类似于哪些元素需要结束标签、什么时候为属性使用引号或者如何更好地使用 CSS 等方面。

虽然较古老、较宽松的编码方式可能看上去更容易，但是通过严格限制使用标签的方式，尽可能地将标签与样式分离，仅使用 JavaScript 增强页面，则最终获得的页面将能够被更多的浏览器正确显示，并且更多的人将访问您的站点更长的时间。

因此，感谢您选择本书并恭喜您完整地阅读它。祝福您在创建第一个 Web 站点时一切顺利，并且希望它是一个良好的开始！

练习题答案

第 1 章

1. 利用相关的表示元素标记下面的语句。

The 1st time the **bold** man wrote in *italics*, he underlined several key words.

答：该语句使用了上标、粗体、斜体和下划线表示元素。

```
<p>The 1<sup>st</sup> time the <b>bold</b> man wrote in <i>italics</i>, he
<u>underlined</u> several key words.</p>
```

2. 标记下面的列表，其中具有插入和删除的内容：

Ricotta pancake ingredients:

- ❑ 1 ~~1/2~~3/4 cups ricotta
- ❑ 3/4 cup milk
- ❑ 4 eggs
- ❑ 1 cup plain white flour
- ❑ 1 teaspoon baking powder
- ❑ ~~75g~~ 50g butter
- ❑ pinch of salt

答：下面是项目列表，其中添加了编辑的元素：

```
<h1>Ricotta pancake ingredients:</h1>
<ul>
  <li>1 <del>1/2</del><ins>3/4</ins> cups ricotta</li>
  <li>3/4 cup milk</li>
  <li>4 eggs</li>
  <li>1 cup plain <ins>white</ins> flour</li>
  <li>1 teaspoon baking powder</li>
  <li><del>75g</del><ins>50g</ins> butter</li>
  <li>pinch of salt</li>
</ul>
```

第 2 章

1. 回顾本章“试一试”中的示例，其中创建了一个菜单和一个新页面，该页面中具有一些指向菜单中每一道菜的链接(类似于菜单页面顶部的链接)。然后添加一个到 Wrox Press Web 站点(www.wrox.com)的链接。

答：编写的代码应当类似如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Links to menu</title>
</head>
<body>
<h1>Links to the Wrox Cafe Menu</h1>
<div id="links">
  <a href="menu.html#starters">Starters</a> |
  <a href="menu.html#mains">Main courses</a> |
  <a href="menu.html#deserts">Deserts</a>
</div>
<p>Wrox Cafe is a fictional cafe brought to you from <a
href="http://www.wrox.com">Wrox Press</a></p>
</body>
</head>
```

为了使其成为一个 Strict XHTML 1.0 文档，组成菜单的链接已经放置在一个 元素中(因为所有的内联元素必须包含在块级元素中)，并且在每个链接的文件名之后使用一个英镑符号(#)，后面跟上 id 属性的值，以指明链接指向页面的哪个部分。

为了链接到 Wrox Web 站点，需要使用完整的 URL(即在浏览器窗口中输入的 URL)作为 href 属性的值。

2. 采用如下语句，在应当具有链接的部分周围放置<a>元素。

```
<p>To find out why advertising on our site works, visit the testimonials
page.</p>
```

答：链接正好放置在单词“testimonials”的周围。记住，当一个链接在文本的中间时，链接的实际内容应当较短并且简明扼要，以便用户能够浏览页面以找到他们感兴趣的关键项。

```
<p>To find out why advertising on our site works, visit the
<a>testimonials</a> page.</p>
```

3. 下面<a>元素放置的位置存在什么错误？

```
<p>You can read the full article <a>here</a>.</p>
```

答：对于浏览页面的用户来说，链接的描述性不充分。单词 here 将会突出显示，而这里可能希望将人们的注意力引向单词 full article。应当更改为：

```
<p>Click on the link to read the <a>full article</a>.</p>
```

第 3 章

1. 在下面的示例中，添加一些图标图像，它们分别表示一本日记、一部照相机和一张报纸。在第 3 章下载代码的 images 文件夹中提供了所有这些图像。

```
<h1>Icons</h1>
<p>Here is an icon used to represent a diary.</p>
<img src="images/diary.gif" alt="Diary" width="150" height="120" /><br />
<p>Here is an icon used to represent a picture.</p>
Camera image goes here<br />
<p>Here is an icon used to represent a news item.</p>
Newspaper image goes here<br />
```

完成的页面应当类似于图 3-16。

答：下面是包含新图像的页面主体；其中包含图像的代码行具有阴影：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1</title>
</head>
<body>
<h1>Icons</h1>
<p>Here is an icon used to represent a diary.</p>
<img src="images/diary.gif" alt="Diary" width="150" height="120" /><br />
<p>Here is an icon used to represent a picture.</p>
<img src="images/picture.gif" alt="Picture" width="150" height="120" /><br />
<p>Here is an icon used to represent a news item.</p>
<img src="images/news.gif" alt="news" width="150" height="120" /><br />
</body>
</html>
```

2. 查看图 3-17 和图 3-18 中显示的 4 幅图像，确定如果以 JPEG 或 GIF 格式保存它们，是否获得较小的文件尺寸以及较好的图像质量。

答：这些图像应当以以下格式保存：

图像 1: JPEG

图像 2: GIF

根据第 3 章中的讨论，具有大面积单调色的图像(例如在图像 1 中仅具有人的黑色轮廓像)采用 GIF 格式保存时能够比采用 JPEG 格式具有更好的压缩率，而 JPEG 格式更适合于保存照片型的图像。也可以使用建议的 PNG 格式取代 GIF，因为 PNG 是 GIF 的替代格式。

第 4 章

1. 用于表的<caption>元素应当放置在文档的什么位置，默认情况下在何处显示标题？

答：<caption>元素应当出现在起始标签<table>之后，但是位于第一个<tr>元素之前。

2. 屏幕阅读器以什么样的顺序阅读图 4-11 中的表单元格？

答：将以如下顺序阅读姓名：Emily、Jack、Frank、Mary、Dominic、Amy、Thomas、Angela 和 David。

3. 创建一个表来存放图 4-12 中的数据。提示如下：文档必须遵循 Transitional XHTML 1.0 规范，因为 width 属性用于表第一行中的单元格。在本章中已经给出一些如何生成边框的示例，即使用另一个逐渐淘汰的属性，但是这个属性应当应用于<table>元素而不是单元格。

答：下面是关于电影院时刻表的示例(cinema.html)。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Classic Movies Times</title>
</head>
<body>
<table border="1" width="500">
<caption>Classic Movie Day</caption>
  <tr>
    <th></th>
    <th width="200">5 pm</th>
    <th width="200">7 pm</th>
    <th width="200">9 pm</th>
    <th width="200">11 pm</th>
  </tr>
  <tr>
    <th>Screen one</th>
    <td>Star Wars</td>
    <td>Empire Strikes Back</td>
    <td>Return of the Jedi</td>
    <td>The Exorcist</td>
  </tr>
  <tr>
    <th>Screen two</th>
    <td colspan="2">Dances with Wolves</td>
    <td colspan="2">Gone With the Wind</td>
  </tr>
  <tr>
    <th>Screen three</th>
    <td colspan="2">2001: A Space Odyssey</td>
    <td>The Conversation</td>
    <td>5 Easy Pieces</td>
```

```
  </tr>
</table>
</body>
</html>
```

第 5 章

1. 创建一个类似于图 5-25 所示的 e-mail 反馈表单。

注意，第一个文本框是一个 readonly 类型的文本框，从而用户不能改变 e-mail 接收人的姓名。

答：下面是 e-mail 反馈表单的代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Reply to ad</title>
</head>
<body>
<h2>Reply to ad</h2>
<p>Use the following form to respond to the ad:</p>
<form action="http://www.example.com/ads/respond.aspx" method="post"
     name="frmRespondToAd">
<table>
  <tr>
    <td><label for="emailTo">To</label></td>
    <td><input type="text" name="txtTo" readonly="readonly" id="emailTo"
         size="20" value="Star Seller" /></td>
  </tr>
  <tr>
    <td><label for="emailFrom">To</label></td>
    <td><input type="text" name="txtFrom" id="emailFrom" size="20" /></td>
  </tr>
  <tr>
    <td><label for="emailSubject">Subject</label></td>
    <td><input type="text" name="txtSubject" id="emailSubject" size="50"
/></td>
  </tr>
  <tr>
    <td><label for="emailBody">Body</label></td>
    <td><textarea name="txtBody" id="emailBody" cols="50" rows="10">
       </textarea></td>
  </tr>
</table>
  <input type="submit" value="Send email" />
</form>
</body>
</html>
```

2. 创建一个类似于图 5-26 所示的投票或排名表单。

注意，将下面的<style>元素添加到文档的<head>部分中，以确保表的每一列具有相同的固定宽度，并且文本居中对齐(第 7 章中将更详细地介绍这方面的知识)。

```
<head>
  <title>Voting</title>
  <style type="text/css">td {width:100; text-align:center;}</style>
</head>
```

答：下面是投票表单的代码。注意如何对表单的中间值使用 checked 属性，以便它在加载时显示平均分数(防止在没有选择值的情况下提交表单)：

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Voting</title>
  <style type="text/css">td {width:100; text-align:center;}</style>
</head>
<body>
<h2>Register your opinion</h2>
<p>How well do you rate the information on this site (where 1 is very poor
and
5 is very good)?</p>
<form action="http://www.example.com/ads/respond.aspx" method="get"
      name="frmRespondToAd">
<table>
  <tr>
    <td><input type="radio" name="radVote" value="1" id="vpoor" /></td>
    <td><input type="radio" name="radVote" value="2" id="poor" /></td>
    <td><input type="radio" name="radVote" value="3" id="average"
        checked="checked" /></td>
    <td><input type="radio" name="radVote" value="4" id="good" /></td>
    <td><input type="radio" name="radVote" value="5" id="vgood" /></td>
  </tr>
  <tr>
    <td><label for="vpoor">1 <br />Very Poor</label></td>
    <td><label for="poor">2 <br />Poor</label></td>
    <td><label for="average">3 <br />Average</label></td>
    <td><label for="good">4 <br />Good</label></td>
    <td><label for="vgood">5 <br />Very Good</label></td>
  </tr>
</table>
<input type="submit" value="Vote now" />
</form>
</body>
</html>
```

第 6 章

1. 重新创建图 6-11 所示的框架集文档(在该文档中，单击一种水果将会加载一个新页面到主窗口中)。当页面加载时，它将附带对应水果的详细描述。

答：第一个示例需要 5 个文件：

- 框架集文档
- 导航文档
- apple 页面
- orange 页面
- banana 页面

下面是框架集文档(example1.html)：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head>
  <title>Fruit example</title>
</head>
<frameset cols="200, 450, *">
  <frame src="frames/fruitNav.html" />
  <frame name="main_frame" src="frames/apple.html" />
  <noframes><body>This site makes uses of a technology called frames.
  Unfortunately the browser you are using does not support this technology.
  We recommend that you update your browser. We apologize for any
  inconvenience this causes.
  </body></noframes>
</frameset>
</html>
```

这是一种框架集文档类型，它包含了具有固定大小的两列，然后窗口的剩余部分保留为空(因此前两列仅具有<frame>元素)。注意，第二个<frame>元素附带了 name 属性，因此导航框架中的链接能够加载在页面的该部分中。

下面是导航面板(fruitNav.html)：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Navigation</title>
  <style type="text/css">img {border-style:none; border-width:0px;}</style>
</head>
<body>
<h1> Navigation </h1>
<p>Click on the fruit to find out more about it.</p>
```

```
<a href="../frames/apple.html" target="main_frame"><img src="../images/
  apple.jpg" alt="apple" /></a>
<a href="../frames/orange.html" target="main_frame"><img src="../images/
  orange.jpg" alt="orange" /></a>
<a href="../frames/banana.html" target="main_frame"><img src="../images/
  banana.jpg" alt="banana" /></a>
</body>
</html>
```

这是一个普通的 XHTML 文档；对于这个文档，唯一需要注意的方面是链接中使用的 target 属性(它指明应当在其他框架中打开链接)和<head>部分中的<style>元素(在第 7 章中介绍了该元素)。

关于水果的页面(apple.html、orange.html 和 banana.html)基本相同，唯一的区别是它们的文本内容。下面是 apple.html：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Apple</title>
</head>
<body>
<h1>Apples</h1>
<p>Apples come in different colors, and there are over 7500 varieties of
apples.</p>
<p>An apple contains about 5g of fiber (1/5 recommended daily average).</p>
</body>
</html>
```

2. 重新创建图 6-12 所示的<iframe>元素。

下面是新的练习代码；它仅改变了本章 iFrame 示例中的文本：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Football focus</title>
</head>
<body>
  <h1>Quarter Final - Wintertons Cup</h1>
  <h3>
    <a href="frames/teamA.html" target="iframe">Manchester Rangers</a>
      vs
    <a href="frames/teamB.html" target="iframe">Birmingham United</a>
  </h3>
  <p><iframe name="iframe" width="300" height="150"
      src="frames/clickForTeams.html" align="left"></iframe>
    Today's big soccer game is between Manchester Rangers and Birmingham
```

```
    United.
<br />The match will be played at Highgate Fields stadium, and
    is sure to be the big game of the week. <br /> All eyes on the underdogs
    Birmingham United who did not expect to get this far in the competition.
  </p>
</body>
</html>
```

答：内联框架示例需要4个文件：

- example2.html，它包含需要加载的页面。
- teamA.html，它包含队伍A中运动员的姓名。
- teamB.html，它包含队伍B中运动员的姓名。
- clickForTeam.html，在用户单击任何队伍之前加载入iframe中。

第一个文件是example2.html，它包含<iframe>元素。它是一个普通的XHTML文档，具有两个链接，并且这两个链接均携带target属性，因此它们能够指明应当在哪个框架中装载文档。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Football focus</title>
</head>
<body>
<h1>Quarter Final - Wintertons Cup</h1>
<h3>
  <a href="frames/teamA.html" target="iframe">Manchester Rangers</a>
  vs
  <a href="frames/teamB.html" target="iframe">Birmingham United</a>
</h3>
  <p>
    <iframe name="iframe" width="300" height="150"
src="frames/clickForTeams.html"
align="left" />
    Today's big soccer game with Manchester Rangers playing Birmingham
    United. The match will be played at Highgate Fields stadium, and is sure
    to be the big game of the week, with all eyes on the underdogs Birmingham
    United who did not expect to get this far in the competition, although
    the gate receipts will be a welcome relief for the team that has been
    facing financial difficulties.
  </p>
</body>
</html>
```

从<iframe>元素中可以看到，它附带一个src属性，用于指明当加载页面时应当将称为clickForTeams.html的页面加载入iframe中。这只是一个纯粹的XHTML页面：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Teams</title>
</head>
<body>
<h3>Click on a team name to load their players here</h3>
</body>
</html>
```

teamB.html 页面中包含了一个表，表中包含该队伍首发阵容中的运动员。teamA.html 页面基本与此相同，只是包含的运动员不同。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Team B</title>
</head>
<body>
<h3>Birmingham United</h3>
  <p>The players of Birmingham United are</p>:
  <table>
    <tr><th>Number</th><th>Name</th></tr>
    <tr><td>1</td><td>Chris Warner</td></tr>
    <tr><td>2</td><td>Felix Thomlinson</td></tr>
    <tr><td>3</td><td>Barry Carr</td></tr>
    <tr><td>4</td><td>Mike Patterson</td></tr>
    <tr><td>5</td><td>Richard Neilson</td></tr>
    <tr><td>6</td><td>Brian Childer</td></tr>
    <tr><td>7</td><td>Micky Stephens</td></tr>
    <tr><td>8</td><td>Richard Brooks</td></tr>
    <tr><td>9</td><td>Nick Evans</td></tr>
    <tr><td>10</td><td>Joseph Barton</td></tr>
    <tr><td>11</td><td>Rob Bishop</td></tr>
  </table>
</body>
</html>
```

第 7 章

1. 返回到本章中的第一个“试一试”示例，在其中添加一些样式，以显示每一种字体的粗体和斜体版本。最终得到的页面将如图 7-36 所示。

只允许在源文档中使用元素和
元素以及在样式表中使用类选择器。在 元素的内容上需要添加一个顶部页边空白，以将它们互相分离。

答：下面的 XHTML 给出了文档的新结构。这里使用元素重复关于机敏的棕毛狐狸(quick brown fox)所在的文本行。每个元素具有一个 class 属性，该属性的值是 bold 或 italic。在每个文本行之后有一个换行元素。<link />元素的 href 属性也指向新的样式表。

下面是新的 font-test2.html 文件：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Font test</title>
  <link rel="stylesheet" type="text/css" href="font-test2.css" />
</head>
<body>
<div class="arial">
  Arial The quick brown fox jumped over the lazy dog.<br />
  <span class="bold">Arial The quick brown fox jumped over the lazy
    dog.</span>
  <br />
  <span class="italic">Arial The quick brown fox jumped over the lazy
    dog.</span><br />
</div>
<div class="helvetica">
  Helvetica The quick brown fox jumped over the lazy dog.<br />
  <span class="bold">Helvetica The quick brown fox jumped over the lazy
    dog.</span><br />
  <span class="italic">Helvetica The quick brown fox jumped over the lazy
    dog.</span><br />
</div>
<div class="TimesNewRoman">
  Times New Roman The quick brown fox jumped over the lazy dog.<br />
  <span class="bold">Times New Roman The quick brown fox jumped over the lazy
   dog.</span><br />
  <span class="italic">Times New Roman The quick brown fox jumped over the
  lazy dog.</span><br />
</div>
<div class="MrsEaves">
  Mrs Eaves The quick brown fox jumped over the lazy dog.<br />
  <span class="bold">Mrs Eaves Bold The quick brown fox jumped over the lazy
    dog.</span><br />
  <span class="italic">Mrs Eaves Italic The quick brown fox jumped over the
  lazy dog.</span><br />
</div>
</body>
</html>
```

现在查看 font-test2.css 样式表。第一个新的特性是 margin-top，它将按照每一种字体划分示例。接下来是用于粗体样式(使用 font-weight 特性)和斜体样式(使用 font-style 样式)的

新的类选择器。

```
/* CSS Style sheet for font-test.html */
body {background-color:#ffffff;}
div {line-height:28px;
margin-top:20px;}
div.arial {font-family:arial, courier;}
div.helvetica {font-family:Helvetica, courier;}
div.TimesNewRoman {font-family:"Times New Roman", courier;}
div.MrsEaves {font-family:"Mrs Eaves", courier;}
.bold {font-weight:bold;}
.italic {font-style:italic;}
```

2. 查看下面的 XHTML 页面：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Font test</title>
  <link rel="stylesheet" type="text/css" href="tableStyles.css" />
</head>
<body>
<table>
  <tr>
    <th>Quantity</th>
    <th>Ingredient</th>
  </tr>
  <tr class="odd">
    <td>3</td>
    <td>Eggs</td>
  </tr>
  <tr>
    <td>100ml</td>
    <td>Milk</td>
  </tr>
  <tr class="odd">
    <td>200g</td>
    <td>Spinach</td>
  </tr>
  <tr>
    <td>1 pinch</td>
    <td>Cinnamon</td>
  </tr>
 </table>
</body>
</html>
```

现在创建 tableStyles.css 样式表，使这个示例的外观如图 3-37 所示。

不要担心获得的尺寸与图 7-37 的屏幕截图不同，但需要确保在单元格中添加内边距，

并且在外部添加边框。IE 浏览器在默认情况下将创建白色边框，在第 8 章中将介绍如何删除它。

答：可以以多种方式创建这个样式表，下面是其中一种方式：

```
/* CSS Style sheet for tableStyles.html */
body {
  background-color:#ffffff;
  font-family:arial, verdana, sans-serif;
  font-size:14px;}
table {
  border-style:solid;
  border-width:1px;
  border-color:#666666;}
th {
  color:#ffffff;
  background-color:#999999;
  font-weight:bold;
  border:none;
  padding:4px;}
tr {background-color:#cccccc;}
tr.odd {background-color:#efefef;}
td {
  color:#000000;
  padding:2px;}
```

第 8 章

1. 在这个练习中，需要创建内容的链接表，它位于一个长文档的上方，以有序列表的方式出现并链接到文档主要部分中的题头。

本书的下载代码中提供了 XHTML 文件 exercise1.html，请为该文件创建样式表。该样式表应该执行以下操作：

- 设置所有链接的样式，包括激活的和访问过的链接
- 列表的内容设置为粗体
- 列表的背景设置为亮灰色，并使用内边距确保项目编号能够显示
- 链接框的宽度为 250 像素
- 将题头项目符号的样式改变为空心圆
- 将链接项目符号的样式改变为正方形

创建的页面应当如图 8-36 所示。

答：下面是内容的链接表的样式表，首先是用于<body>元素的样式：

```
body {
  background-color:#ffffff;
  font-family:arial, verdana, sans-serif;
  font-size:12px;}
```

第一个<ul>元素的选择器应当具有用于 list-style 特性和 font-weight 特性的规则，它们分别被设置为 circle 和 bold。

```
ul {
  list-style:circle;
  font-weight:bold;
```

在第一个<ul>选择器中也应当放置用于链接背景的规则，因此在同一条声明中应当具有如下规则。注意，padding-left 特性确保项目符号保持可见：

```
background-color:#efefef;
padding-left:30px;
width:250px;}
```

然后第二个选择器应当指明位于另外一个<ul>元素内部的<ul>元素应当具有一个 list-style 特性，并且该特性的值为 square，以便在嵌套的链接元素之前添加方框：

```
ul ul {list-style:square;}
```

最后，剩余的规则指明链接的外观：

```
a:link {
  color:#0033ff;;
  text-decoration:none;}
a:visited {
  color:#0066ff;
  text-decoration:none;}
a:active {
  text-decoration:underline;}
a:link:hover {
  color:#003399;
  background-color:#e9e9e9;
  text-decoration:underline;}
```

2. 在这个练习中，测试您的 CSS 定位技能。需要创建一个页面，该页面以不同的方式表示链接到本章不同小节处的链接。每一小节将在不同的块中显示，每一个块将被绝对定位在一个从左上到右下方向的对角线上。中间的框必须出现在最上方，如图 8-37 所示。

可以在本章的下载代码中找到 XHTML 源文件 exercise2.html。

答：首先需要为<body>元素设置一些背景特性：

```
body {
  background-color:#ffffff;
  font-family:arial, verdana, sans-serif;
  font-size:12px;}
```

为了赋予每个 元素一个边框、固定的宽度和内边距，应当将相应的规则放置在用于所有 元素的选择器中。另外该选择器还应当具有一个 background-color 特性(在这个示例中将该特性的值设置为白色)，以防止文本混乱(因为如果不这样操作，框将是透明的)：

```
div {
  background-color:#ffffff;
  padding:10px;
  border-style:groove; border-width:4px; border-color:#999999;
  width:300px;}
```

每个 元素然后需要不同的定位特性，以确保它们出现在对角线位置。这里必须设置 z-index 特性，以便以正确的顺序表示选中的框。

```
div.page1 {
  position:absolute;
  top:70px;
  z-index:2;
  background-color:#f2f2f2;}
div.page2 {
  position:absolute;
  top:170px; left:100px;
  z-index:3;}
div.page3 {
  position:absolute;
  top:270px; left:200px;
  z-index:1;
  background-color:#efefef;}
```

第 9 章

1. 再一次查看本章开始部分的页面；图 9-18 再次显示了该页面。请列举出页面中在设计阶段应当列出的所有不同元素，并将它们放置在相关的分组或者类别中。

例如，对于搜索框可以列举以下元素：

```
Title
Navigation
Main news article
```

答：如何编写页面的元素列表取决于自己，但是当创建这个设计时，我决定在页面中填充如下元素：

```
Title
Navigation
  Local News
  National News
  World News
  Politics
  Science
  Technology
  Travel
  Business
  Education
  Entertainment
```

```
Main Article
  Heading
  Text
  Image
  Read more link
Second and third articles
Heading
  Text
  Image
  Read more link
Text to explain that it is a fixed-width layout example
```

2. 尝试重新创建图 9-18 中看到的页面。提示：该页面是一种固定宽度的设计，因此可以使用如何创建固定宽度布局一节中的样本代码作为起点，并在其中添加一些内容。

答：这个页面完全构建在一个固定宽度的表中。搜索栏、徽标、导航栏、页面的主体和页脚都位于表的不同行中。

页面的主体位于一个包含元素中，该元素用于控制页面的宽度。在该元素之内是一个题头(该题头占用了页面的整个宽度)和两列(每一列位于自己的包含元素 中)。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
  <title>Fixed width page design</title>
  <link rel="stylesheet" type="text/css" href="exercise.css"
</head>
<body>
<div id="maincontainer">
  <div id="topsection"><h1>The London News</h1></div>
  <div id="leftcolumn">
    <a class="nav1" href="">Local News</a>
    <a class="nav2" href="">National News</a>
    <a class="nav3" href="">World News</a>
    <a class="nav4" href="">Politics</a>
    <a class="nav5" href="">Science</a>
    <a class="nav6" href="">Technology</a>
    <a class="nav7" href="">Travel</a>
    <a class="nav8" href="">Business</a>
    <a class="nav9" href="">Education</a>
    <a class="nav10" href="">Entertainment</a>
  </div>
  <div id="contentwrapper">
    <div id="contentcolumn">
      <img src="images/london.jpg" width="350" height="210"
           alt="London Cityscape in Winter" align="left" />
      <h2>The Big Freeze</h2>
      <p>London braces itself for a big freeze as forecasters predict a
```

```
severe drop
        in temperature across the capital. The Met Office weather team
indicated
        that temperatures would remain in negative figures until Wednesday
with
        winds making it feel even colder.</p>
    <div class="readmore"><a href="">Read more</a></div>
    <div class="clear"><br /></div>
    <div class="secondaryStory">
      <img src="images/cows.jpg" width="110" height="72"  alt="Cows on a
farm"
          align="left" />
      <h3>Farms Cleared</h3>
      <p>The three farms at the centre of the latest scare to hit the
         agricultural community have been given the all clear.</p>
      <div class="readmore"><a href="">Read more</a></div>
    </div>
    <div class="secondaryStory">
      <img src="images/houses.jpg" width="110" height="72"  alt="Row of
houses"
          align="left" />
      <h3>Interest Rate Rises</h3>
      <p>The Bank of England has raised the base interest rate another half
         percent - the fourth consecutive rise in interest rates in as many
         quarters.</p>
      <div class="readmore"><a href="">Read more</a></div>
      </div>
     <div class="clear"></div>
    </div>
  </div>
  <div id="footer">Fixed width page design</div>
</div>
</body>
</html>
```

下面是用于赋予页面样式的CSS代码：

```
body{
  margin:0;
  padding:0;
  font-family:arial, verdana, sans-serif;
  background-color:#ffffff;}
#maincontainer{width:800px;}
#topsection{
  color:#ffffff;
  background-color: #000000;
  height: 100px;}
#topsection h1{
  margin: 0;
  padding-top: 15px;}
#contentwrapper{
```

```
  float: left;
  width:600px;}
#contentcolumn{
  padding:10px;}
#contentcolumn img {
  margin: 0px 10px 10px 0px;}
#leftcolumn{
  float: left;
  width: 200px; /*Width of left column*/
  color:#333333;}
#leftcolumn a{
  display:block;
  color:#333333;
  background-color:#d6d6d6;
  margin-bottom:2px;
  padding:2px;
  text-decoration:none;
  font-weight:bold;}
#leftcolumn .nav1 {border-left: 5px solid #3366FF;}
#leftcolumn .nav2 {border-left: 5px solid #6633FF;}
#leftcolumn .nav3 {border-left: 5px solid #CC33FF;}
#leftcolumn .nav4 {border-left: 5px solid #FF33CC;}
#leftcolumn .nav5 {border-left: 5px solid #FF3366;}
#leftcolumn .nav6 {border-left: 5px solid #FF6633;}
#leftcolumn .nav7 {border-left: 5px solid #FFCC33;}
#leftcolumn .nav8 {border-left: 5px solid #66FF33;}
#leftcolumn .nav9 {border-left: 5px solid #33FF66;}
#leftcolumn .nav10 {border-left: 5px solid #33FFCC;}
#footer{
  clear: left;
  width: 800px;
  color: #ffffff;
  background-color:#000000;
  text-align:center;
  padding: 4px;}
.clear {clear:both;}
.secondaryStory {
  float:left;
  width:50%;}
h1 {padding:0px 0px 0px 20px;margin:0px;}
h2, h3 {margin:0px;}
img {border:1px solid #000000;}
```

第 10 章

在这个练习中，需要对本章末尾的“试一试”部分中的表单(registration.html)添加第二个页面。表 A-1 给出了需要在该表单中添加的新项。

另外也需要添加以下内容：

- 在页面顶部添加一条指示，告诉用户已经完成表单的多少内容。
- 在页面底部添加一个 Back 按钮和一个 Proceed 按钮(而不是 Submit 按钮)

表 A-1

信　息	表 单 控 件	是否必须填写
地址 1	文本输入框	是
地址 2	文本输入框	否
城镇/郊区	文本输入框	否
城市/州	文本输入框	是
邮政编码	文本输入框	是

完成上述操作之后，页面应当如图 10-27 所示(registration2.html)。

答：下面是 registration2.html 文件的代码。与许多其他示例一样，该示例首先使用一个外部 CSS 样式表：

```
<html>
  <head>
    <title>Try it out</title>
    <link rel="stylesheet" type="text/css" href="registration.css" />
  </head>
  <body>
```

接下来是一个表，它指明表单具有 3 个页面。这些页面使用不同的样式，以表明用户当前所在的步骤(通过 stepOn 类指示)：

```
<table class="steps">
  <tr>
    <td class="stepOff">Login details</td>
    <td class="stepOn">Contact details</td>
    <td class="stepOff">Confirm details</td>
  </tr>
</table>
```

然后处理表单本身。在一个表内布局表单，以便标签和表单元素能够对齐。每一个表单元素具有一个<label>元素，并且该元素的 for 属性值对应于表单控件的 id：

```
<form name="frmExample" action="" method="post">
  <fieldset>
  <legend>Contact details:</legend>
  <table>
    <tr>
      <td class="label"><label for="address1">Address 1:</label></td>
      <td class="form">
        <input type="text" name="txtAddress1" id="address1" size="30" />
    </td>
    </tr>
```

```
    <tr>
      <td class="label"><label for="address2">Address 2:</label></td>
      <td class="form">
        <input type="text" name="txtAddress2" id="address2" size="30" />
      </td>
    </tr>
    <tr>
      <td class="label"><label for="town">Town/Suburb:</label></td>
      <td class="form">
        <input type="text" name="txtTown" id="town" size="12" />
      </td>
    </tr>
    <tr>
      <td class="label"><label for="city">City/State:</label></td>
      <td class="form">
        <input type="text" name="txtState" id="city" size="12" />
      </td>
    </tr>
    <tr>
      <td class="label"><label for="postcode">Postal/Zip Code:</label></td>
      <td class="form">
        <input type="text" name="txtPostCode" id="postcode" size="12" />
      </td>
    </tr>
  </table><br />
```

最后，页面的左边具有一个 Back 按钮，而右边具有一个 Proceed 按钮。仍然使用一个表定位这两个按钮。是用提示表明必须填写星号所标识的表单字段：

```
  <table class="steps">
    <tr>
      <td class="back"><input type="submit" value="Back" /></td>
      <td class="proceed"><input type="submit" value="Proceed" /></td>
    </tr>
  </table>
  </fieldset>
  <br /><span class="required">*</span> = required
</form>
</body>
</html>
```

第 11 章

1. 创建一个脚本，该脚本输出数值 5 的从 1 到 20 的乘法表，方法是使用 while 循环。

答：这个练习是用的代码与 ch11_eg09.html 非常相似；事实上，仅需要改变该示例中的相应数值——其他代码都相同。文件 ch11_eg09.html 计算 3 的从 1 到 10 的乘法表，而这个练习计算 5 的从 1 到 20 的乘法表。

这个练习基于一个计数器(指明正处于表的哪个位置)；每次代码运行时，计数器自动

递增 1。因此，需要确保计数器能够到达 20 而不是 10。这就需要调整 while 循环的条件：

```
while (i < 21) {
```

接下来需要改变乘数，它被输出并且用于计算中。下面的代码将乘数改为 5：

```
document.write(i + " x 5 = " + (i * 5) + "<br />" );
```

最终的代码应当如下所示：

```
<script type="text/JavaScript">
i = 1
while (i < 21) {
  document.write(i + " x 5 = " + (i * 5) + "<br />" );
  i ++
}
</script>
```

这段代码的长度没有 ch11_eg09.html 中的循环长，但是它将数值写出两次，这就演示了在代码中使用循环的能力。

2. 修改 ch11_eg06.html，使其能够完成以下功能：

- 向中午 12 点之前到达页面的访问者输出“Good Morning”(使用一条 if 语句)。
- 向中午 12 点到下午 6 点之间到达页面的访问者输出“Good Afternoon”(再次使用一条 if 语句)。(提示：可能需要使用逻辑运算符)
- 向下午 6 点到午夜之间到达页面的访问者输出“Good Evening”(再次使用一条 if 语句)。

答：下面的简单脚本由 ch11_eg06.html 修改而来，在早上时使用单词“Good Morning”向用户致敬，在中午时使用单词“Good Afternoon”向用户致敬，在晚上时使用单词“Good Evening”向用户致敬。

该脚本使用 date 对象的 getHours()方法来确定时间，然后使用 if 语句检查适应的时间，以确定向用户显示的内容。

注意，在下午中使用了一个逻辑运算符，以检查时间位于 12 点之后和下午 6 点之前。

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();

  if (time < 12)
  document.write('Good Morning');
  if (time > 12 && time < 18)
  document.write('Good Afternoon')
  if (time > 18)
  document.write('Good Evening');
</script>
```

第 12 章

1. 您的任务是为图 12-22 中的竞赛表单创建一个验证函数，该函数需要检查用户是否执行以下操作：

- 输入姓名
- 提供有效的 e-mail 地址
- 选择其中一个单选按钮作为问题的答案
- 赋予加时赛问题的答案，答案不超过 20 个单词

检查的顺序应当遵循控件在表单上的顺序。

下面是该表单的代码：

```
<form name="frmCompetition" action="competition.aspx" method="post"
onsubmit="return validate(this);">
<h2>An Example Competition Form <br />(Sorry, there are no real prizes!)</h2>
<p> To enter the drawing to win a case of Jenny's Jam, first answer
this question: "What color are strawberries?" Then provide an answer for
the tie-breaker question: "I would like to win a case of Jenny's Jam
because..." in no more than 20 words.</p>
<table>
 <tr>
  <td class="formTitle">Name: </td>
  <td><input type="text" name="txtName" size="18" /></td>
 </tr>
 <tr>
  <td class="formTitle">Email: </td>
  <td><input type="text" name="txtEmail" size="18" /></td>
 </tr>
 <tr>
  <td class="formTitle">Answer: </td>
  <td><input type="radio" name="radAnswer" value="Red" /> Red<br />
      <input type="radio" name="radAnswer" value="Gray" /> Gray<br />
      <input type="radio" name="radAnswer" value="Blue" /> Blue
  </td>
 </tr>
 <tr>
  <td class="formTitle">Tie breaker <br/ ><small>(no more than 20 words)
  </small>:
</td>
  <td><textarea name="txtTieBreaker" cols="30" rows="3"
/></textarea></td>
 </tr>
 <tr>
  <td class="formTitle"></td>
  <td><input type="submit" value="Enter now" /></td>
 </tr>
</table>
</form>
```

答：这个示例的 validate()函数使用在第 12 章中介绍的一些技术。该函数首先设置一个 returnValue 变量，当函数结束运行之后，该变量的值将是 true 或 false。returnValue 变量的初始值为 true，如果表单的任何字段没有满足需求，则它将改为 false。

```
<script type="text/JavaScript">
  function validate(form) {
    var returnValue = true
```

首先必须检查 txtName 字段是否具有值：

```
var name=form.txtName.value
if (name=="")
  {
    returnValue = false;
    alert("You must enter a name")
    document.frmCompetition.txtName.focus();
  }
```

接下来必须检查 e-mail 地址是否遵循应该具有的格式。如果地址为空，则它将不匹配正则表达式；因此，如果该控件最初为空，则不需要检查它：

```
var email=form.txtEmail.value
var rxEmail = /^\ w(\.?[\w-])*@\w(\.?[\w-])*\.[a-z]{2,6}(\.[a-z]{2})?$/i;
if (!rxEmail.test(email))
  {
    returnValue = false;
    alert("You must enter a valid email address")
    document.frmCompetition.txtEmail.focus()
  }
```

然后必须遍历所有单选按钮以查看是否提供答案。这包括遍历这些按钮并测试每个按钮是否具有 checked 特性。如果一个单选按钮，则将一个变量(这里该变量称为 radioChosen)改为具有值 true。遍历所有的单选按钮之后，使用一条条件 if 语句检查这个属性的值是 true 还是 false。

```
var radioChosen = false;
var radioButtons = form.radAnswer;
for (var i=0; i<radioButtons.length; i++) {
if (radioButtons[i].checked)
  {
  radioChosen=true;
  }
}
if (radioChosen == false) {
  returnValue = false;
  alert("You did not answer the question");
}
```

最后需要处理的是<textarea>元素和加时赛。该元素需要具有一个值，但是它的长度不能超过 20 个单词。首先需要检查该元素是否具有值：

```
var tieBreaker=form.txtTieBreaker.value
if (tieBreaker=="")
  {
    returnValue = false;
    alert("You must enter an answer for the tie breaker")
    document.frmCompetition.txtTieBreaker.focus();
  }
```

然后使用字符串对象的 split()函数和一个正则表达式将输入的值拆分为独立的单词。因为 split()函数拆分字符串的依据是空格，所以可以通过查找该函数创建的数组的长度来检查用户输入多少个单词。因为数组基于 0，所以需要确定数组项的数量是否小于或等于 20。如果单词的数量太多，则警告用户，并且告诉他已经输入多少个单词，以帮助他缩减单词数量。

```
var tieBreakerWords = tieBreaker.split(/\s+/g);
wordCount = tieBreakerWords.length;
if (wordCount > 20) {
  returnValue = false;
  alert("Your tie breaker answer must be no more than 20 words. You entered
"+
          wordCount+ "words.");
  document.frmCompetition.txtTieBreaker.focus();
}
```

下面是最后的测试，returnValue 的值(可以是 true 或 false)指明表单是否应当提交。

```
  return returnValue
}
</script>
```

XHTML 元素参考

本附录是 HTML 规范和 XHTML 规范中的元素的快速参考。依次列举这些元素，并且包含每个元素所能够携带的属性及其作用的简短描述。

逐渐淘汰的元素通过它们后面的单词“Deprecated”标识，应当尽量避免使用这些元素。本书也建议应当使用 CSS 规则替换样式方面的标记和元素。

支持元素的 Internet Explorer(IE)浏览器、Netscape (N)浏览器和 Firefox (FF)浏览器的第一个版本将在元素名的后面提供，这些浏览器支持元素的最初版本分别为 IE3、N3 和 FF1。Netscape 浏览器支持的元素也将被 Firefox 浏览器支持。符号“all”表明元素被所有浏览器所支持，包括 IE3、N3 和 FF1 以及较新的版本。但是，并不是所有属性都被相同版本的浏览器支持——某些属性由较新版本的浏览器引入。

关于语法，请注意如下方面：

- 所有的元素名都应当小写。
- 任何没有值的属性应当将其属性名作为它的值，以便与 XHTML 兼容；例如：disabled ="disabled"。
- 所有的属性值也应当出现在双引号内。

核心属性

除非特别说明，否则核心属性可以用于本附录中介绍的所有元素。

表 B-1

class=name	指定元素的类，以将其与样式表中的规则相关联
dir=ltr|rtl	指定显示文本的方向(从左到右或者从右到左)
id=name	在所在文档范围内为该元素定义唯一的标识值
lang=language	为元素的内容指定(人类)语言
onclick=script	指定当用户在这个元素之上单击鼠标时调用的脚本
ondblclick=script	指定当用户在这个元素之上双击鼠标时调用的脚本
onkeydown=script	指定当这个元素具有焦点并且用户按下一个键盘键时调用的脚本
onkeypress=script	指定当这个元素具有焦点并且用户按下和释放一个键盘键时调用的脚本

(续表)

onkeyup=script	指定当这个元素具有焦点并且用户释放一个键盘键时调用的脚本
onmousedown=script	指定当光标位于元素内容之上并且用户按下一个鼠标按键时调用的脚本
onmousemove=script	指定当用户在这个元素的内容之上移动鼠标指针时调用的脚本
onmouseout=script	指定当鼠标位于元素之上并被移动到它的边框外部，以使其不再位于该元素之上时调用的脚本
onmouseover=script	指定当鼠标在这个元素的内容之上移动时调用的脚本
onmouseup=script	指定当光标位于这个元素的内容之上并且用户释放鼠标按键时调用的脚本
style=style	为该元素指定一个内联的 CSS 样式规则
title=string	为该元素指定一个标题
xml:lang	为该元素的内容指定(人类)语言

<a> (all)

该元素用于定义一个链接，必须指定它的 href 属性和 name 属性。

表 B-2

accesskey=key_character	为这个锚点定义一个热键/键盘快捷键
charset=encoding	指定一个字符集，用于编码目标文档
coords=x_ycoordinates	指定坐标的列表
href=url	指定超链接目标的 URL
hreflang=language_code	指定链接目标的语言编码
rel=relationship(same|next|parent|previous| string)	指示该文档与目标文档之间的关系
rev=relationship	指示目标文档和当前文档的反向关系
shape=circ|circle|poly|polygon|rect|rectangle	定义一个区域的形状
tabindex=number	定义这个元素在焦点移动顺序中的位置
target=<window_name>|_parent|_blank|_top|_self	定义框架或窗口的名称，其中框架或窗口将加载链接的文档
type=MIME_type	定义目标的 MIME 类型

<abbr> (IE4+、N6+、FF1+)

指示该元素的内容是一种缩写词。

<acronym> (IE4+、N6+、FF1+)

指示该元素的内容是只取首字母的缩写词。

<address> (all)

指示元素的内容是一个地址。

<applet> Deprecated (all)

用于在页面中放置一个 Java 小应用程序或者可执行代码。
该元素仅采用表 B-3 中列举的属性。

表 B-3

align=top|middle| bottom|left|right| absmiddle|baseline| absbottom|texttop	将 Java 小应用程序与包含它的元素对齐
alt=text	指定替换<applet>元素的备选文本，用于当浏览器支持<applet>元素但不能执行它时
archive=url	指定一个类存档，它必须被下载到浏览器中并搜索
class=name	为元素指定一个类，以将其与样式表中的规则相关联
code=classname	指定代码的类名(必须指定的属性)
codebase=url	指定一个 URL，从该 URL 所在位置能够下载需要的代码
height=number	指定<applet>的高度，单位是像素
hspace=number	指定到<applet>的左边和右边的宽度，单位是像素
id=name	为元素指定唯一的 ID
name=name	为这个小应用程序的实例指定一个名称
object=data	指定用于运行的已编译代码的文件名
vspace=number	指定到<applet>的顶部和底部的高度，单位为像素
width=number	指定<applet>的宽度，单位为像素

<param> (all)

表 B-4

name=name	指定参数的名称
type=MIME_type	定义参数的 MIME 类型
value=string	定义参数的值

<area> (all)

用于指定图像映射中可单击区域或热点的坐标。

表 B-5

accesskey=key_character	为这个区域定义热键/键盘快捷键
alt=text	为这个区域指定一个备选文本，当图像无法加载时使用该文本
coords=string	为该区域指定坐标的列表
href=url	指定超链接目标的 URL
name=string	为该元素指定用于识别它的名称
nohref	指定没有文档与这个区域相关联
notab	指定这个元素不参与该文档的焦点移动顺序
shape=circ|circle|poly| polygon|rect|rectangle	定义一个区域的形状
tabindex=number	定义这个元素在焦点移动顺序中的位置
target=<window_name>| _parent|_blank|_top|_self	定义框架或窗口的名称，该框架或窗口将加载链接的文档

<b> (all)

将以粗体字体显示该元素的内容。

<base>

为文档中的链接指定基 URL。
该元素仅支持表 B-6 中列举的属性。

表 B-6

href=url	为文档中的链接指定基 URL
id=id	为该元素指定唯一的标识符
target=<window_name>| _parent|_blank|_top|_self	定义框架或窗口的名称，该框架或窗口将加载链接的文档

<basefont> Deprecated (all)

指定一个基础字体作为显示文档时的默认字体。
该元素仅支持表 B-7 中列举的属性。

表 B-7

Color=color	指定这个元素中文本的颜色
face=font_family_name	指定这个元素的字体族
size=value	指定字体的大小(必须指定的属性)

<bdo> (IE5+、N6+、FF1+)

为选择的文本片段关闭双向显示算法。

表 B-8

dir=ltr|rtl	指定这个元素使用的字体族

<bgsound> (只有 IE — IE3+)

指定当加载页面时使用的背景声音或音频文件。

表 B-9

loop=number	指定播放音频文件的次数(可以是一个整数或关键字 infinite)
src=url	指定用于播放的音频文件的 URL

<big> (IE4+、N4+、FF1)

以大于包含元素的字体尺寸显示文本。

<blink> (只有 Netscape/Firefox—N3+、FF1+)

该元素的内容闪烁，只有 Netscape 浏览器和 Firefox 浏览器支持该元素。

<blockquote> (all)

该元素的内容为引用，通常用于段落或者更长文本的引用(否则使用<q>元素)。

表 B-10

cite=url	指定引用来源的 URL

<body> (all)

指定页面主体部分的开始和结束。

表 B-11

accesskey=key_character	为页面主体定义一个热键或者键盘快捷键
alink=color	指定活动链接的颜色
background=url	为背景图像指定一个 URL，该背景图像用作为整个文档背景的壁纸
bgcolor=color	为文档指定一种背景颜色
bgproperties=fixed	图像不随着文档内容滚动
leftmargin=number	为文档的左边指定页边空白，单位为像素
link=color	为未访问的链接指定一种颜色
onload=scriptevent handler	指定加载页面时运行的脚本
onunload=scriptevent handler	指定关闭页面时运行的脚本
text=color	为文档中的文本指定一种颜色
topmargin=number	为文档顶部指定页边空白，单位为像素
vlink=color	指定已访问链接的颜色

 (all)

插入一个换行。

该元素仅支持表 B-12 中列举的属性。

表 B-12

class=name	为元素指定一个类，以将其与样式表中的规则相关联
clear=left|right|none| all	中断页面的流程，并将向下移动直到指定的页边空白清晰可见
id=id	为这个元素指定唯一的标识符
style=style	为这个元素指定内联的 CSS 样式规则
title=string	为这个元素指定标题

<button> (IE4+、N3+、FF1+)

用于创建一个 HTML 按钮，该标记中的任何内容都将用作为按钮的标题。

表 B-13

accesskey=key_character	为它定义一个热键或键盘快捷键
disabled=disabled	禁用该按钮，阻止用户干涉
name=name	为表单控件指定一个名称，它将作为名/值对的一部分传递给表单的处理程序(必须指定的属性)
onblur=script	指定当鼠标移开按钮时运行的脚本
onfocus=script	指定当元素获得焦点时运行的脚本
tabindex=number	定义这个元素在移动焦点顺序中的位置
type=button|submit| reset	指定按钮的类型
value=string	指定参数的值，它将作为名/值对的一部分传递给处理程序(必须指定的属性)

<caption> (all)

这个元素的内容指定一个标题，该标题放置在表的旁边。

表 B-14

align=top|_bottom|right| left	对于 IE 浏览器，该属性指定标题的水平对齐方式；对于 Netscape 浏览器，该属性设置标题的垂直位置
valign=bottom|top	指定标题的垂直位置

Deprecated (all)

这个元素(和它的子元素)的内容将在页面中居中显示。

<cite> (all)

这个元素的内容是一段引用，通常以斜体显示它。

<code> (all)

该元素的内容是代码，应当以固定宽度的字体显示它。

<col> (IE3+、N4+、FF1+)

指定表的基于列的默认值。

表 B-15

align=center|left| right|justify|char	指定该列的对齐方式
bgcolor=color	为该列指定一种背景颜色
char=string	指定单元格内文本的对齐字符
charoff=string	指定定位对齐字符的偏移字符
span=number	受<col>标签影响的列的数量
valign=bottom|top	指定元素内的内容的垂直对齐方式
width=number	指定列的宽度，单位为像素

<colgroup> (IE3+、N4+、FF1+)

用于包含一组列。

表 B-16

align=center|left|right|justify|char	指定列中内容的水平对齐方式
bgcolor=color	指定列组的背景颜色
char=string	指定单元格内文本的对齐字符
charoff=string	指定定位对齐字符的偏移字符
valign=bottom|top	指定元素内的内容的垂直对齐方式
width=number	指定列组的宽度，单位为像素

<comment> (只有 IE4+)

该元素的内容是注释，浏览器不显示它(该元素仅被 IE4+浏览器支持——它不是 HTML 或 XHTML 的一部分)。

该元素仅支持表 B-17 中所示的属性。

表 B-17

id=string	为元素指定唯一标识符
Lang=language_type	指定注释的语言
xml:lang=language_type	指定注释的语言

<dd> (all)

定义列表中项的定义。相比于其他文本，该元素的文本缩进显示。

<del> (IE4+、N6+、FF1+)

从文档的较早版本中删除该元素的内容。

表 B-18

cite=url	指定用于说明删除原因的 URL
datetime=date	指定删除该内容的日期和时间

<dfn> (all)

定义术语的一个实例。

<dir> Deprecated (all)

以目录样式的文件列表方式显示该元素的内容。

表 B-19

type=bullet	指定用于显示列表的项目符号的类型

 (all)

表 B-20

type=format	指定用于显示列表项的项目符号的类型
value=number	指定列表项的数量

(all)

用于容纳其他元素的包含元素，它定义了页面的一个部分。这是一个块级容器。

表 B-21

align=center|left|right	指定 元素中文本的对齐方式
nowrap=nowrap	阻止 元素内的单词换行

<dl> (all)

表示一个定义列表。

表 B-22

compact=compact	使该列表在垂直方向上更加紧凑

<dt> (all)

表示定义列表内的一个定义术语。

(all)

该元素的内容是强调文本，通常以斜体字体显示。

<embed> (all)

在需要其他支持应用程序的页面中嵌入文档。

表 B-23

align=absbottom| absmiddle|baseline| bottom|left|middle| right|texttop|top	指定包含元素内的对齐方式
border=number	指定嵌入对象周围的边框的宽度，单位为像素
height=number	指定嵌入对象的高度，单位为像素
hidden=hidden	指定嵌入对象被隐藏
hspace=number	指定添加在嵌入对象的左边和右边的额外空间量
name=name	为嵌入对象指定名称
palette=foreground| background	为嵌入对象设置前景颜色和背景颜色
pluginspage=url	指定某个页面的 URL，可从该页面中下载与对象关联的插件
src=url	指定对象使用的数据的 URL
type=MIME_type	指定对象使用的数据的 MIME 类型
units=en|ems|pixels	设置 height 属性和 width 属性的单位
vpsace=number	指定添加在嵌入对象的上方和下方的额外空间量
width=number	指定嵌入对象的宽度，单位为像素

<fieldset> (IE4+、N6+、FF1+)

在被包含的元素周围创建一个框，以表明它们是表单中的相关项。

表 B-24

align=center|left|right	指定一组元素的对齐方式
tabindex=number	定义这个<fieldset>在焦点移动顺序中的位置

<font> Deprecated (all)

指定该元素内文本使用的字体的字型、大小和颜色。

表 B-25

color=color	指定这个元素内的文本的颜色
face=font_family_list	指定用于这个元素内的文本的字体族
size=value	指定用于这个元素内的文本的大小

<form> (all)

表单控件和元素的包含元素。

表 B-26

accept-charset=list	指定处理应用程序能够处理的可接受字符集列表
action=url	指定用于处理表单的处理应用程序的 URL
enctype=encoding	指定表单值的编码方法
method=get|post	指定数据从浏览器发送到处理应用程序的方式
onreset=script	指定复位表单值时运行的脚本
onsubmit=script	指定在表单提交之前运行的脚本
target=<window_name>| _parent| _blank| _top| _self	定义框架或窗口的名称，该框架或窗口将加载表单的结果

<frame> (all)

指定框架集中的一个框架

该元素仅支持表 B-27 中列举的属性。

表 B-27

[event_name]=script	被大多数元素支持的固有事件
bordercolor=color	指定框架边框的颜色
class=name	指定一个类名，以将样式与该元素相关联
frameborder=no|yes|0|1	指定框架是否具有边框
Id=string	为该元素指定唯一值
lang=language_type	为该框架的内容指定使用的语言
longdesc=url	指定框架内容的描述的 URL
marginheight=number	指定框架的页边空白的高度，单位为像素
marginwidth=number	指定图像的页边空白的宽度，单位为像素

(续表)

noresize=noresize	指定框架的大小不能调整
scrolling=auto|yes|no	指定如果框架的内容无法完全包含在浏览器窗口的空间中，则框架是否具有滚动栏
style=style	指定内联的 CSS 样式规则
src=url	指定框架内容的位置的 URL
title=title	为框架指定标题

<noframes> (all)

如果浏览器不支持框架，将显示这个元素的内容。

<frameset> (all)

指定包含多个框架(和其他可能存在的嵌套框架集)的框架集，这个元素替换文档中的<body>元素。

表 B-28

border=number	指定框架集中每个框架的边框宽度
bordercolor=color	指定框架集中框架边框的颜色
cols=list	指定框架集中列的数量，从而可以控制框架集的布局
frameborder=no|yes|0|1	指定这个框架集中的框架是否具有边框
framespacing=number	指定每个框架之间的空间，单位为像素
onblur=script	指定当鼠标移开框架集时运行的脚本
onload=script	指定当框架集加载时运行的脚本
onunload=script	指定框架集关闭时运行的脚本
rows=number	指定框架集中行的数量，从而可以控制框架集的布局

<head> (all)

用于文档题头信息的容器元素；浏览器不会显示该元素的内容。
该元素仅支持表 B-29 中列举的属性。

表 B-29

class=classname	指定一个类，以将样式规则与这个元素相关联
dir=ltr|rtl	指定这个元素内的文本的方向
Id=string	为这个元素指定唯一标识符
lang=language_type	指定用于这个元素内的语言
profile=url	为文档的配置文件指定一个 URL
xml:lang=language_type	指定这个元素内使用的语言

<h*n*> (all)

从<h1>(最大)到<h6>(最小)之间的题头。

表 B-30

align=left|center|right	指定其包含元素内的题头的水平对齐方式

<hr /> (all)

创建横跨页面(或包含元素)的水平刻度尺。
该元素仅支持表 B-31 中列举的属性。

表 B-31

[event_name]=script	被大多数元素支持的固有事件
align=center|left|right	指定刻度尺的水平对齐方式
class=classname	为该元素指定一个类，以将其与样式表中的规则相关联
color=color	指定水平刻度尺的颜色
dir=ltr|rtl	指定文本的方向
id=string	为这个元素指定唯一标识符
noshade=noshade	指定该刻度尺不具有 3D 阴影
style=string	为该元素指定内联的 CSS 样式规则
title=string	为该元素指定标题
width=number	指定刻度尺的宽度，单位为像素或者占用包含元素的百分比

<html> (all)

HTML 或者 XHTML 页面的包含元素。

表 B-32

class=classname	为该元素指定一个类，以将其与样式表中的规则相关联
dir=ltr|rtl	指定该元素内文本的方向
id=string	为这个元素指定唯一标识符
lang=language_type	指定这个元素内使用的语言
version=url	指定这个文档使用的HTML版本——在XHTML中，该属性被DOCTYPE声明代替
xmlns=uri	指定用于 XHTML 文档中的名称空间
xml:lang=language_type	指定用于这个元素内的语言

<i> (all)

这个元素的内容将以斜体字体显示。

<iframe> (IE3+、N6+、FF1+)

创建页面内的内联浮动框架。

表 B-33

align=absbottom| absmiddle|baseline| bottom|top|left|middle| right|texttop|top	指定框架与周围的内容或页边空白之间的对齐方式
frameborder=no|yes|0|1	指定边框是否显示：1 表示具有边框，0 表示不具有边框
height=number	指定框架的高度，单位为像素
longdesc=url	指定框架内容的描述的 URL
Marginheight=number	指定框架的上方和下方与周围内容之间的距离，单位为像素
marginwidth=number	指定框架的左边和右边与周围内容之间的距离，单位为像素
scrolling=auto|yes|no	指定在内容太多以至于无法被框架容纳时是否允许显示滚动栏
src=url	指定显示在框架中的文件的 URL
width=number	指定框架的宽度，单位为像素

<img> (all)

在文档中嵌入一幅图像。

表 B-34

align=absbottom| absmiddle|baseline| bottom|top|left|middle| right|texttop|top	指定图像相对于它周围的内容的对齐方式
alt=text	指定备选文本，以便在应用程序无法加载图像时显示该文本(必须指定的属性)；也用于可访问性设备
border=number	指定图像边框的宽度，单位为像素——如果图像是一个链接，则必须使用这个特性以阻止显示边框
controls	为视频片段显示回放控制(仅被 IE3 浏览器支持)
dynsrc=url	指定用于播放的视频片段的 URL
height=number	指定图像的高度，单位为像素
hspace=number	指定添加在图像左边和右边的额外空间量
ismap=ismap	指定图像是否是服务器端的图像映射
longdesc=url	指定用于描述图像的内容的 URL
loop=number	指定播放视频的次数；可以采用 infinite 值
lowsrc=url	指定图像的低分辨率版本的 URL，当正在加载整幅图像时可以首先显示该版本
name=name	为该元素指定名称
onabort=script	指定一个脚本，当加载图像失败时运行该脚本
onerror=script	指定一个脚本，当加载图像出现错误时运行该脚本
onload=script	指定一个脚本，当已经加载图像时运行该脚本
src=url	指定图像的 URL
start=fileopenmouseover_number	指定播放视频片段的时间
usemap=url	指定包含坐标和链接的映射，其中坐标和链接定义了图像的链接(服务器端的图像映射)
vspace=number	指定添加在图像上方和下方的额外空间量
width=name	指定图像的宽度

<input type="button"> (all)

创建一个表单输入控件，该控件是用户能够单击的按钮。

表 B-35

accesskey=key_character	为该控件定义一个热键或键盘快捷键
disabled=disabled	禁用按钮，阻止用户干涉
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="checkbox"> (all)

创建一个表单输入控件，该控件是用户能够选择的复选框。

表 B-36

accesskey=key_character	为该控件定义一个热键或键盘快捷键
checked=checked	指定选中复选框(可用于默认选择该复选框)
disabled=disabled	禁用复选框，阻止用户选择它
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
readonly=readonly	阻止用户修改内容
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="file"> (all)

创建一个表单输入控件，该控件允许用户选择一个文件。

表 B-37

accesskey=key_character	为该控件定义一个热键或键盘快捷键
disabled=disabled	禁用文件上传控件，阻止用户干涉
maxlength=number	指定用户可以输入的最大字符数量
name=name	指定表单控件的名称，该名字作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onchange=script	指定一个脚本，当元素的值改变时运行该脚本
onfocus=script	指定一个脚本，当元素获得焦点时运行该脚本
readonly=readonly	阻止用户修改内容
size=number	指定为该元素显示的字符数量
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="hidden"> (all)

创建一个表单输入控件，它类似于文本输入控件，但是被隐藏，用户无法看到它(但是如果用户查看页面的源代码，则仍然能够看到该控件的值)。

表 B-38

name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理程序

<input type="image"> (all)

创建一个表单输入控件，它类似于一个按钮或者提交控件，但是使用图像而不是按钮。

表 B-39

accesskey=key_character	为该元素定义一个热键或键盘快捷键
align=center|left|right	指定图像的对齐方式
alt=string	为图像提供备选文本
border=number	指定边框的宽度，单位为像素

(续表)

Disabled=disabled	禁用图像按钮，阻止用户干涉
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
src=url	指定图像的源文件
Readonly=readonly	阻止用户修改内容
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="password"> (all)

创建一个表单输入控件，它类似于一个单行文本输入控件，但显示的是星号或者项目符号标记，而不是用户输入的字符，以阻止其他人偷窥用户输入的文本。该控件可用于输入敏感信息——但需要注意的是，传递给服务器的值将以纯文本的形式发出(如果传递的是敏感信息，则仍然需要考虑使用类似 SSL 的加密技术)。

表 B-40

accesskey=key_character	为该元素定义一个热键或键盘快捷键
disabled=disabled	禁用文本输入框，阻止用户干涉
maxlength=number	指定用户能够输入的最大字符数量
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onchange=script	指定一个脚本，当元素的值改变时运行该脚本
onfocus=script	指定一个脚本，当元素获得焦点时运行该脚本
onselect=script	指定一个脚本，当用户选择该元素时运行该脚本
readonly=readonly	阻止用户修改内容
size=number	指定输入框的宽度，单位为字符数
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="radio"> (all)

创建一个表单输入控件，该控件是一个单选按钮。这些控件以组的形式显示，一组中的单选按钮具有相同的 name 属性值，并且创建互斥的值组(一组中仅可以选择一个单选按钮)。

表 B-41

accesskey=key_character	为该元素定义一个热键或键盘快捷键
checked=checked	指定默认情况下选中这个单选按钮
disabled=disabled	禁用单选按钮，阻止用户干涉
name=name	指定表单控件的名称，该名称作为名/值对的一部分被传递给处理程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
readonly=readonly	阻止用户修改内容
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="reset"> (all)

创建一个表单输入控件，该控件是一个按钮，用于将表单的值复位为页面加载时显示的默认值。

表 B-42

accesskey=key_character	为该元素定义一个热键或键盘快捷键
disabled=disabled	禁用该按钮，阻止用户干涉
notab=notab	指定这个元素不参与文档的焦点移动顺序
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="submit"> (all)

创建一个表单输入控件，该控件是一个提交按钮，用于将表单的值发送给服务器。

表 B-43

accesskey=key_character	为该元素定义一个热键或键盘快捷键
disabled=disabled	禁用该按钮，阻止用户干涉
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序
notab=notab	指定这个元素不参与文档的焦点移动顺序
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<input type="text"> (all)

创建一个表单输入控件，该控件是一个单行文本输入框。

表 B-44

accesskey=key_character	为该元素定义一个热键或键盘快捷键
disabled=disabled	禁用文本输入框，阻止用户干涉
maxlength=number	指定用户能够输入的最大字符数量
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
notab=notab	指定这个元素不参与文档的焦点移动顺序
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onchange=script	指定一个脚本，当元素的值改变时运行该脚本
onfocus=script	指定一个脚本，当元素获得焦点时运行该脚本
onselect=script	指定一个脚本，当选择元素时运行该脚本
readonly=readonly	阻止用户修改内容
size=number	指定该控件的宽度，单位为字符数
tabindex=number	定义这个元素在焦点移动顺序中的位置
taborder=number	指定这个元素在焦点移动顺序中的位置
value=string	指定控件的值，该值将作为名/值对的一部分传递给处理应用程序

<ins> (IE4+、N6+、FF1+)

相对于文档的早期版本，添加该元素的内容。

表 B-45

cite=url	指定一个 URL，该 URL 说明为什么添加该内容
datetime=date	为添加的内容指定日期和时间

<isindex> Deprecated (all)

标识一个可搜索的索引。
该元素仅支持表 B-46 中列举的属性。

表 B-46

accesskey=key_character	为该元素定义一个热键或键盘快捷键
action=url	IE 仅指定搜索应用程序的 URL
class=classname	为元素指定一个类，以将其与样式表中的规则相关联
ir=ltr|rtl	指定元素内文本的方向
id=string	为这个元素指定唯一的标识符
lang=language_type	为这个元素内的文本指定语言
prompt=string	为该字段的输入指定备选的提示信息
style=string	为该元素指定内联的 CSS 样式规则
tabindex=number	定义这个元素在焦点移动顺序中的位置
title=string	为该元素指定标题
xml:lang=language_type	指定这个元素内使用的语言

<kbd> (all)

应当使用键盘输入该元素的内容，并且以固定宽度的字体显示内容。

<keygen> (只有 Netscape、N3+)

用于生成页面中的密钥材料——密钥材料指的是用于安全的加密密钥。
该元素仅支持表 B-47 中列举的属性。

表 B-47

challenge=string	提供一个字符串，该字符串与密钥打包在一起
class=classname	为元素指定一个类，以将其与样式表中的规则相关联

(续表)

id=string	为这个元素指定唯一的标识符
name=string	为该元素指定名称

<label> (IE4+、N6+、FF1+)

该元素的内容用作为表单元素的标签。

表 B-48

accesskey=key_character	为该元素定义一个热键或键盘快捷键
for=name	指定该元素标记的元素的 id 属性值
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onfocus=string	指定一个脚本，当元素获得焦点时运行该脚本

<layer> (只有 Netscape、N4+)

定义一个页面区域，该区域能够存放不同的页面。该元素仅被 Netscape 浏览器支持；本书没有介绍该元素。

表 B-49

above=name	将该层定位在指定层的上方
background=url	指定该层背景图像的 URL
below=name	将该层定位在指定层的下方
bgcolor=color	为该层设置背景颜色
clip=number[,number, number,number]	指定该层的修剪区域
left=number	指定该层左边相对于包含文档或层的位置
Name=name	指定该层的名称
src=url	指定另外一个文档作为该层的内容
top=number	指定该层相对于包含文档或层的顶部的位置
visibility=show|hide| inherit	指定该层是否可见
width=number	指定该层的宽度，单位为像素
z-index=number	指定该层的堆叠顺序

<legend> (IE4+、N6+、FF1+)

这个元素的内容是标题文本，它放置在一个<fieldset>元素中。

表 B-50

accesskey=key_character	为该元素定义一个热键或键盘快捷键
align=top|left|bottom|right	指定图例相对于字段集的位置

 (all)

这个元素的内容是列表中的一项。该元素称为行项。如果需要了解适当的属性，可查看该类列表的父元素(<ul>、<ol>和<menu>)。

表 B-51

type=bullet_type	指定用于显示列表项的项目符号的类型
value=number	指定列表起始的数值

<link> (all)

定义文档和另外一个资源之间的链接，通常用于在文档中包含样式表。
该元素仅支持表 B-52 中列举的属性。

表 B-52

charset=character_set	指定用于编码链接文件的字符集
href=url	指定链接文档的 URL
hreflang=language_type	指定链接目标的语言编码
media=list	指定计划用于文档的媒体类型
rel=same|next|parent|previous|string	指示该文档与目标文档之间的关系
rev=relation	指定目标文档和该文档之间的反向关系
type=type	指定被链接的文档的 MIME 类型

<listing> Deprecated (IE3+)

以固定宽度的字体显示该元素的内容。

<map> (all)

创建一个客户端图像映射，并指定可单击区域或热点的集合。

表 B-53

name=string	映射的名称(必须指定的属性)

<marquee> (只有 IE、IE3+)

创建一个滚动的文本字幕 (仅被 IE3+浏览器支持)。

表 B-54

accesskey=key_character	为该元素定义一个热键或键盘快捷键
align=top|middle|bottom	定位字幕相对于周围内容的位置
behavior=alternate| scroll|side	指定字幕的操作或行为
bgcolor=color	指定字幕的背景颜色
direction=down|left|up| right	指定文本滚动的方向
height=number	指定字幕的高度，单位为像素
hspace=number	指定添加到字幕左边和右边的额外空间量
id=string	为该元素指定一个唯一标识符
loop=number	指定字幕循环的次数，它的值可以是关键字 infinite
scrollamount=number	指定文本每次滚动的距离，单位为像素
scrolldelay=number	指定每次滚动间的延迟，单位为毫秒
tabindex=number	定义这个元素在 Tab 键顺序中的位置
vspace=number	指定添加到字幕上边和下边的额外空间量
width=number	指定字幕的宽度，单位为像素

<menu> Deprecated (all)

作为独立的项显示子元素。该元素已被列表(<ol>和<ul>)代替，在 HTML 4.01 中标记为逐渐淘汰。

 (all)

表 B-55

type=bullet_type	指定用于显示列表项的项目符号类型

<meta> (all)

该元素中提供关于文档的信息或用于浏览器的指令；不向用户显示这些信息或指令。该元素仅支持表 B-56 中列举的属性。

表 B-56

charset=character_set	指定用于编码文档的字符集
content=meta_content	指定元信息的值
dir=ltr|rtl	指定该元素内文本的方向
http-equiv=string	为元信息指定 HTTP 等价名称；以便使服务器可以在 HTTP 头中包含名称和内容
lang=language_type	指定用于这个元素中的语言
name=string	指定元信息的名称
scheme=scheme	指定用于解释该特性的简介方法
xml:lang=language_type	指定用于这个元素中的语言

<multicol> (只有 N3、N4)

用于定义多列格式(该元素仅被 Netscape 3 浏览器和 Netscape 4 浏览器支持——它不是 XHTML 的组成部分)。

表 B-57

cols=numberofcolumns	指定列数
gutter=number	指定隔条(列之间的空间)的大小，单位为像素
width=number	指定列的宽度，单位为像素

<nextid> (不被任何浏览器支持)

用于指定标识符，该标识符将被 HTML 编辑软件使用，以便编辑软件知道一系列文档中下一个文档的 ID(该元素仅是 HTML 2.0 规范的一部分，没有被浏览器实现)。

该元素仅支持表 B-58 中列举的属性。

表 B-58

n=string	设置 next id 号

<nobr> (all)

表示“不换行(no break)”，阻止该元素的内容换行。

<noembed> (N2、N3、N4)

当浏览器不支持<embed>元素或所需的查看应用程序时，显示该元素的内容。

<noframes> (all)

当浏览器不支持框架时显示该元素的内容。

<nolayer> (只有 N4+)

当浏览器不支持层时显示该元素的内容。

<noscript> (all)

当浏览器不支持脚本时显示该元素的内容。如果脚本被禁用，大多数浏览器也将显示该元素的内容。

<object> (IE3+、N6+、FF1+)

在页面中添加一个对象或者非 HTML 控件。在未来该元素可能会成为包含图像的标准方式。

表 B-59

align=absbottom| absmiddle|baseline| bottom|left|middle| right|texttop|top	指定对象相对于周围文本的位置
archive=url	指定该对象使用的档案或资源的 URL 列表
border=number	指定边框的宽度，单位为像素
classid=url	指定对象的 URL
codebase=url	指定运行对象所需要的代码的 URL
codetype=MIME-type	指定代码基的 MIME 类型
data=url	指定该对象使用的数据
declare	声明一个对象，但是实例化它
height=number	指定对象的高度，单位为像素
hspace=number	指定添加在嵌入对象的左边和右边的额外空间量
name=name	为该对象指定名称
notab=notab	指定这个元素不参与文档的焦点移动顺序
shapes=shapes	指定该对象具有某种外观的超链接
standby=string	定义一条消息，当对象正在加载时显示它
tabindex=number	定义这个元素在焦点移动顺序中的位置
type=MIMEtype	为该对象的数据指定 MIME 类型
usemap=url	定义用于该对象的图像映射
vspace=number	指定添加在嵌入对象的上方和下方的额外空间量
width=number	指定该对象的宽度，单位为像素

<param> (IE3+、N6+、FF1+)

用作为<object>的子元素以设置该对象的特性。

表 B-60

id=id	为该参数指定唯一 ID
name=name	为该参数指定名称
type=MIMEtype	为该参数指定 MIME 类型
value=string	为该参数指定值
valuetype=type	为属性值指定类型

 (all)

创建一个有序或编号列表。

表 B-61

compact=compact	尝试使列表垂直方向更加紧凑
start=number	指定一个数值，列表将从该数值开始
type=bullet_type	指定用于显示列表项的项目符号类型

 (all)

表 B-62

type=bullet_type	指定用于显示列表项的项目符号类型
value=number	指定列表项的数量

<optgroup> (IE6+、N6+、FF1+)

用于将多个<option>元素分组到一个选择框中。

表 B-63

disabled=disabled	禁用该选项组，阻止用户干涉
label=string	为该选项组指定标签

<option> (all)

在下拉列表或选择框中包含一个选项。

表 B-64

disabled=disabled	禁用该选项，阻止用户干涉
label=string	为该选项指定标签
selected=selected	指示当加载页面时选择该选项为默认值
value=string	指定表单控件中这个选项的值，该值将作为名/值对的一部分被传递给处理应用程序

<p> (all)

这个元素的内容是一个段落。

表 B-65

align=center|left|right	指定该段落中文本的对齐方式

<param>

用作为<object>元素或<applet>元素的子元素，以设置该对象的特性。详情可查看<object>元素或<applet>元素。

<plaintext> Deprecated (IE3+、N2、N3、N4)

以不带任何格式的方式显示该元素的内容(该元素在 HTML 3.2 中被标记为逐渐淘汰的类型)。

<pre> (all)

这个元素的内容以固定宽度类型显示，固定宽度类型保留代码中的格式(例如空格和换行)。

表 B-66

width=number	指定预定义格式区域的宽度，单位为像素

<q> (IE4+、N6+、FF1+)

该元素的内容是短引用。

表 B-67

cite=url	指定当前讨论引用的内容的URL

<s> Deprecated (all)

该元素的内容以添加删除线的形式显示。

<samp> (all)

该元素的内容是样本代码清单，通常以较小的固定宽度字体显示。

<script> (all)

该元素的内容是一个脚本代码，浏览器应该执行它。

表 B-68

charset=encoding	指定用于编码脚本的字符集
Defer=defer	推迟脚本的执行
language=name	指定这个元素中使用的语言
src=url	脚本文件位置的 URL
type=encoding	指定脚本的 MIME 类型

<select> (all)

创建一个选择框或下拉列表框。

表 B-69

disabled=disabled	禁用这个选项框，阻止用户干涉
Multiple=multiple	允许选择列表中的多个项
name=name	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onchange=script	指定一个脚本，当元素的值改变时运行该脚本
onfocus=script	指定一个脚本，当元素获得焦点时运行该脚本
size=number	指定一次能够出现的项的数量
tabindex=number	定义这个元素在焦点移动顺序中的位置

<small> (all)

这个元素的内容的字体将稍小于包含它的元素的字体。

(all)

用作为内联元素(相对于块级元素)的分组元素；也用于定义页面中文本的非标准属性。

<strike> Deprecated (all)

将以添加删除线的方式显示这个元素的内容。

(all)

着重强调这个元素的内容，并以粗体字型显示该内容。

<style> (IE3+、N4+、FF1+)

包含应用于该页面的CSS样式规则。

<sub> (all)

这个元素的内容显示为下标。

<sup> (all)

这个元素的内容显示为上标。

<table> (all)

创建一个表。

表 B-70

align=center|left|right	指定表的对齐方式
background=url	指定背景图像的 URL
bgcolor=color	为表指定一种背景颜色
border=number	指定边框的宽度，单位为像素
bordercolor=color	指定边框的颜色
bordercolordark=color	指定较暗的边框颜色

(续表)

bordercolorlight=color	指定较亮的边框颜色
cellpadding=number	指定边框和它的内容之间的距离，单位为像素
cellspacing=number	指定单元格之间的距离，单位为像素
cols=number	指定表中列的数量
frame=above|below| border|box|hsides| lhs|rhs|void|vsides	定义边框的显示位置
height=number	指定表的高度，单位为像素
hspace=number	指定添加在表左边和右边的额外空间量
nowrap=nowrap	阻止表的内容换行
rules=all|cols|groups| none|rows	指定内部分隔线的绘制位置
summary=string	提供表的概要描述
valign=bottom|top	指定表中内容的对齐方式
vspace=number	指定添加在表上方和下方的额外空间量
width=number	指定表的宽度，单位为像素

<tbody> (IE3+、N6+、FF1+)

表示表的主体部分。

表 B-71

align=center|left|right	指定表主体内容的对齐方式
char=string	指定用于对齐内容的偏移字符
charoff=string	指定对齐位置的单元格内的偏移量
valign=bottom|top	指定表主体内容的垂直对齐方式
width=number	指定表主体的宽度，单位为像素

<td> (all)

创建表的一个单元格。

表 B-72

abbr=string	为单元格内容指定缩写词
align=center|left|right	指定单元格内容的对齐方式

(续表)

axis=string	为相关单元格组指定名称
background=url	为单元格的背景图像指定一个 URL
bgcolor=color	指定单元格的背景颜色
border=number	指定单元格边框的宽度，单位为像素
bordercolor=color	指定单元格边框的颜色
bordercolordark=color	指定单元格的暗边框颜色
bordercolorlight=color	指定单元格的亮边框颜色
char=string	指定单元格的对齐字符
charoff=string	指定距离单元格对齐字符的偏移量
colspan=number	指定这个单元格跨越的列的数量
headers=string	指定与这个单元格相关联的题头单元格的名称
height=number	指定该单元格的高度，单位为像素
nowrap=nowrap	阻止单元格的内容换行
rowspan=number	指定单元格跨越的行数量
scope=row|col|rowgroup|colgroup	指定题头单元格的范围
valign=bottom|top	指定单元格内容的垂直对齐方式
width=number	指定单元格的宽度，单位为像素

<textarea> (all)

在表单中创建一个多行文本输入控件。

表 B-73

accesskey=key_character	为该元素定义一个热键或键盘快捷键
cols=number	指定该文本区域应当具有的字符列数量(以字符为单位的宽度)
disabled=disabled	禁用该文本区域，阻止用户干涉
name=string	指定表单控件的名称，该名称作为名/值对的一部分传递给处理应用程序(必须指定的属性)
onblur=script	指定一个脚本，当鼠标离开控件时运行该脚本
onchange=script	指定一个脚本，当元素的值改变时运行该脚本
onfocus=script	指定一个脚本，当元素获得焦点时运行该脚本
onselect=script	指定一个脚本，当选择元素时运行该脚本
readonly=readonly	阻止用户修改内容

(续表)

rows=number	指定不出现滚动栏时出现在文本区域内的文本行数量
tabindex=number	定义这个元素在焦点移动顺序中的位置
wrap=physical_vertical_off	当文本长度达到文本区域的宽度时，指定文本是换行还是继续停留在同一行中。

<tfoot> (IE3+、N6+、FF1+)

表示表的某行或某些行用作为表尾。

表 B-74

align=center|left|right	指定表尾内容的对齐方式
char=string	指定用于对齐的偏移字符
charoff=string	指定对齐位置的单元格内的偏移量
valign=bottom|top	指定表尾内容的垂直对齐方式
width=number	指定表主体的宽度，单位为像素

<thead> (IE3+、N6+、FF1+)

表示表的某行或某些行用作为表头。

表 B-75

align=center|left|right	指定表头内容的对齐方式
char=string	指定用于对齐的偏移字符
charoff=string	指定对齐位置的单元格内的偏移量
valign=bottom|top	指定表头内容的垂直对齐方式
width=number	指定表主体的宽度，单位为像素

<th> (all)

表示表的表头单元格。默认情况下，内容通常以粗体字体显示。

表 B-76

abbr=string	为单元格内容指定一个缩写词
align=center|left|right	指定单元格内容的对齐方式
axis=string	为相关单元格组指定名称
background=url	为单元格的背景图像指定 URL
bgcolor=color	指定单元格的背景颜色
border=number	指定单元格边框的宽度，单位为像素
bordercolor=color	指定单元格边框的颜色
bordercolordark=color	指定单元格的暗边框颜色
bordercolorlight=color	指定单元格的亮边框颜色
char=string	指定单元格的对齐字符
charoff=string	指定距离单元格对齐字符的偏移量
colspan=number	指定这个单元格跨越的列的数量
headers=string	指定与这个单元格相关联的题头单元格的名称
height=number	指定该单元格的高度，单位为像素
nowrap	阻止单元格的内容换行
rowspan=number	指定单元格跨越的行数量
scope=row|col|rowgroup| colgroup	指定题头单元格的范围
valign=bottom|top	指定单元格内容的垂直对齐方式
width=number	指定单元格的宽度，单位为像素

<title> (all)

这个元素的内容是文档的标题，通常显示在浏览器的顶部标题栏中；标题可以仅位于页面的头部。该元素仅支持表 B-77 中列举的属性。

表 B-77

dir=ltr|rtl	指定该元素内的文本方向
id=string	为该元素指定唯一标识符
lang=language_type	指定用于该元素内的语言
xml:lang=language_type	指定用于该元素内的语言

<tr> (all)

表示表的一行。

表 B-78

align=center|left|right	指定该行内容的对齐方式
background=url	指定该行的背景图像的 URL
bgcolor=color	指定该行的背景颜色
border=number	指定该行的边框宽度，单位为像素
bordercolor=color	指定该行的边框颜色
bordercolordark=color	指定该行的暗边框颜色
bordercolorlight=color	指定该行的亮边框颜色
char=string	指定该行的对齐字符
charoff=string	指定距离行对齐字符的偏移量
nowrap=nowrap	阻止单元格的内容换行
valign=bottom|top	指定单元格内容的垂直对齐方式

<tt> (all)

这个元素的内容以固定宽度字体显示，如同使用电传打字机打印一样。

<u> (all)

这个元素的内容以添加下划线的文本形式显示(该元素在 HTML 4.01 中被标记为逐渐淘汰的类型)。

 (all)

创建一个无序列表。

表 B-79

compact=compact	使列表在垂直方向更加紧凑
type=bullet_type	指定用于显示列表项的项目符号类型

<var> (IE3+、N6+、FF1+)

这个元素的内容是一个编程变量，通常以较小的固定宽度字体显示。

<wbr> (IE3、N2、N3、N4)

创建<nobr>元素内的一个软换行。

<xmp> Deprecated (all)

这个元素的内容以固定宽度字型显示，通常用于示例或样本代码。该元素被<pre>元素和<samp>元素代替。

CSS 特性

本附录是主要 CSS 特性的参考，可以使用这些 CSS 特性控制文档的外观。

在介绍每一个特性时，首先简单地描述该特性，然后给出它的一个使用示例。接下来，左边的表中给出了该特性能够采用的可能值以及支持该特性的 IE、Netscape 和 Firefox 浏览器的第一个版本。

在右边的表中给出了该特性是否能够被继承、该特性的默认值以及它能够应用于哪些元素。

本附录的末尾部分给出了一些度量单位。

虽然 Netscape 浏览器和 Firefox 浏览器支持许多特性的 inherit 值，但是如果不能首先将该特性设置为其他值，则这个值没有什么用处。

注意：

该附录的表中还给出了哪些浏览器版本支持某个值时是基于 Windows 平台中的浏览器。Mac 平台中的 Internet Explorer 5 浏览器对许多特性的支持显著优于它在 Windows 中的版本。

字体特性

字体特性可用于改变字型的外观。

font

可用于同时设置几个字体特性，特性之间以空格隔开。在这个特性中可以同时指定 font-size、line-height、font-family、font-style、font-variant 和 font-weight。

```
font {color:#ff0000; arial, verdana, sans-serif; 12pt;}
```

表 C-1

值	IE	N	FF
[font-family]	3	4	1
[font-size]	3	4	1

表 C-2

继承	是
默认值	n/a
应用于	所有元素

(续表)

[font-style]	3	4	1
[font-variant]	4	6	1
[font-weight]	3	4	1
[line-height]	3	4	1
inherit	-	6	1

font-family

用于指定希望使用的字型。它可以附带多个值，每个值之间以逗号隔开。其中第一个值是首选字型，然后是第二选择，最后是一个通用字体族(serif、sans-serif、cursive、fantasy 或 monospace)。

```
p {font-family:arial, verdana, sans-serif;}
```

表 C-3

值	IE	N	FF
[通用字体族]	3	4	1
[特定字体族]	3	4	1
inherit	-	6	1

表 C-4

继承	是
默认值	由浏览器设置
应用于	所有元素

font-size

可用于指定字体的大小。font-size 特性具有自己的特定值：

- 绝对大小：xx-small、x-small、small、medium、large、x-large、xx-large
- 相对大小：larger、smaller
- 百分比：占父字体的百分比
- 长度：一种度量单位(本附录末尾部分将介绍度量单位)

表 C-5

值	IE	N	FF
[绝对大小]	3	4	1
[相对大小]	4	4	1
[百分比]	3	4	1
[长度]	3	4	1
inherit	-	6	1

表 C-6

继承	是
默认值	medium
应用于	所有元素

font-size-adjust

可用于调整字体的方位值。字体的方位值是字体中小写字母 *x* 的高度和字体高度之间的比例。

```
{font-size-adjust:0.5;}
```

表 C-7

值	IE	N	FF
[数值]	-	-	-
none	-	-	-
inherit	-	6	1

表 C-8

继承	是
默认值	特定于字体
应用于	所有元素

font-stretch

可用于指定字体中的字母宽度(而不是它们之间的距离)。

- 相对值：normal、wider、narrower
- 固定值：ultra-condensed、extra-condensed、condensed、semi-condensed、semi-expanded、expanded、extra-expanded、ultra-expanded

```
p {font-family:courier; font-stretch:semi-condensed;}
```

表 C-9

值	IE	N	FF
[相对值]	-	-	-
[固定值]	-	-	-
inherit	-	6	1

表 C-10

继承	是
默认值	特定于字体
应用于	所有元素

font-style

赋予字体样式。如果指定的字体版本可用，则将使用该版本；否则，浏览器将显示普通字体。

表 C-11

值	IE	N	FF
normal	3	4	1
italic	3	4	1
oblique	4	6	1
inherit	-	6	1

表 C-12

继承	是
默认值	normal
应用于	所有元素

font-variant

创建大写字母，它们的尺寸与普通小写字母相同。

表 C-13

值	IE	N	FF
normal	4	6	1
small-caps	4	6	1
inherit	-	6	1

表 C-14

继承	是
默认值	normal
应用于	所有元素

font-weight

指定文本的粗细度——文本的“醒目程度”。

- 绝对值：normal、bold
- 相对值：bolder、lighter
- 数字值：0 到 100 之间的数值

```
p {font-weight:bold;}
```

表 C-15

值	IE	N	FF
[绝对值]	3	4	1
[relative]	4	6	1
[数值 1~100]	4	6	1
inherit	-	6	1

表 C-16

继承	是
默认值	normal
应用于	所有元素

文本特性

文本特性从总体上改变文本的外观和布局(相对于字体)。

letter-spacing

指定字母之间的距离为长度单位。

```
p {letter-spacing:1em;}
```

表 C-17

值	IE	N	FF
[长度]	4	6	1
normal	4	6	1
inherit	-	6	1

表 C-18

继承	是
默认值	normal
应用于	所有元素

text-align

指定文本的对齐方式：left、right、center 或 justified。

表 C-19

值	IE	N	FF
left	3	4	1
right	3	4	1
center	3	4	1
justify	4	4	1
inherit	-	6	1

表 C-20

继承	是
默认值	取决于用户代理和元素(通常是 left，例外情况是<th>元素，该元素的默认值是 center)
应用于	所有元素

text-decoration

指定字体是否应当具有 underline、overline、line-through 或 blink 外观。

```
p {text-decoration:underline;}
```

表 C-21

值	IE	N	FF
none	3	4	1
underline	3	4	1
overline	4	6	1
line-through	3	4	1
blink	-	4	1
inherit	-	6	1

表 C-22

继承	否
默认值	无
应用于	所有元素

text-indent

指定文本的缩进，单位为长度或占用父元素宽度的百分比。

```
p {text-indent:3em;}
```

表 C-23

值	IE	N	FF
[长度]	4	4	1
[百分比]	4	4	1
inherit	-	6	1

表 C-24

继承	是
默认值	0
应用于	块级元素

text-shadow

为文本创建阴影。它应用采用 3 个长度值：前两个长度指定阴影偏移的 X 和 Y 坐标，而第三个长度指定一种模糊效果。然后，跟上一种颜色，颜色可以是名称或者十六进制数值。

```
.dropShadow {text-shadow: 0.3em 0.3em 0.5em black}
```

表 C-25

值	IE	N	FF
[阴影效果]	-	-	-
none	-	-	-
inherit	-	6	1

表 C-26

继承	否
默认值	无
应用于	所有元素

text-transform

指定一个元素中文本的大小写：

- none：删除所继承的设置。
- capitalize：所有的字符大写。
- uppercase：所有的字符小写。
- lowercase：每个单词的第一个字母大写。

```
p {text-transform:uppercase;}
```

表 C-27

值	IE	N	FF
none	4	4	1
uppercase	4	4	1
lowercase	4	4	1
capitalize	4	4	1
inherit	-	6	1

表 C-28

继承	是
默认值	无
应用于	所有元素

white-space

该特性指示空格的处理方式：

- normal：空格应该折叠。
- pre：空格将保留。
- nowrap：除了使用
元素之外，文本将不会换行。

```
p {white-space:pre;}
```

表 C-29

值	IE	N	FF
normal	5.5	4	1
pre	5.5	4	1
nowrap	5.5	6	1
inherit	-	6	1

表 C-30

继承	是
默认值	normal
应用于	块级元素

word-spacing

指定单词之间的距离：

```
p {word-spacing:2em;}
```

表 C-31

值	IE	N	FF
normal	6	6	1
[长度]	6	6	1
inherit	-	6	1

表 C-32

继承	是
默认值	normal
应用于	所有元素

颜色特性和背景特性

下面的特性可用于改变页面以及其他框的颜色和背景。

background

这是用于指定背景特性的简写方式，可以同时指定 color、url、repeat、scroll 和 position 特性；特性之间以空格隔开。默认情况下，背景采用透明方式。

```
body {background: #efefef url(images/background.gif); }
```

表 C-33

值	IE	N	FF
[background-attachment]	4	6	1
[background-color]	3	4	1
[background-image]	3	4	1
[background-position]	4	6	1
[background-repeat]	3	4	1
inherit	-	6	1

表 C-34

继承	否
默认值	未定义(默认情况下，背景是透明的)
应用于	所有元素

background-attachment

指定背景图像是否应当固定在页面中的一个位置还是随着页面滚动：

```
body {background-attachment:fixed;
      background-image: url(images/background.gif);}
```

表 C-35

值	IE	N	FF
fixed	4	6	1
scroll	4	6	1
inherit	4	6	1

表 C-36

继承	否
默认值	scroll
应用于	所有元素

background-color

设置背景的颜色，可以是一种颜色或者两种颜色的混合。可以使用颜色名、十六进制值或 RGB 值指定颜色。默认情况下，框是透明的。

```
body {background-color:#efefef;}
```

表 C-37

值	IE	N	FF
[颜色]	4	4	1
transparent	4	4	1
inherit	-	6	1

表 C-38

继承	否
默认值	transparent
应用于	所有元素

background-image

指定一幅图像用作为背景，默认情况下平铺该图像。该特性的值是图像的 URL。

```
body {background-image: url(images/background.gif);}
```

表 C-39

值	IE	N	FF
[url]	4	4	1
none	4	4	1
inherit	-	6	1

表 C-40

继承	否
默认值	none
应用于	所有元素

background-position

指定背景图像在页面中相对于左上角的位置。该特性的值可以是绝对距离、百分比或者一个关键字。如果仅给出一个值，则假设为水平方向。

- 可用的关键字包括：top、bottom、left、right、center

```
body {background-position:center;
      background-image: url(images/background.gif);}
```

表 C-41

值	IE	N	FF
[长度－x y]	4	6	1
[百分比－x% y%]	4	6	1
top	4	6	1
left	4	6	1
bottom	4	6	1
right	4	6	1
center	4	6	1
inherit	-	6	1

表 C-42

继承	否
默认值	top、left
应用于	块级元素

background-positionX

定位背景图像在页面水平方向的位置。该特性值与 background-position 相同(默认值是 top)。

background-positionY

定位背景图像在页面垂直方向从上到下的位置。该特性值与 background-position 相同(默认值是 left)。

边框特性

边框特性可用于控制任何框周围的边框的外观和尺寸。

border (border-bottom、border-left、border-top、border-right)

这是用于指定 border-style、border-width 和 border-color 特性的简写方式。

表 C-43

值	IE	N	FF
<border-style>	4	6	1
<border-width>	4	6	1
<border-color>	4	6	1
inherit	-	6	1

表 C-44

继承	否
默认值	none、medium、none
应用于	所有元素

border-style(border-bottom-style、border-left-style、border-top-style、border-right-style)

用于指定块框四周的线样式。

```
div.page {border-style:solid;}
```

注意，Netscape 浏览器直到第 6 版才开始支持用于各条边的特性。

表 C-45

值	IE	N	FF
none	4	4	1
dotted	5.5	6	1
dashed	5.5	6	1
solid	4	4	1
double	4	4	1
groove	4	4	1
ridge	4	4	1
inset	4	4	1
outset	4	4	1
hidden	-	-	-
inherit	-	6	1

表 C-46

继承	否
默认值	none
应用于	所有元素

border-width(border-bottom-width、border-left-width、border-top-width、border-right-width)

指定边框线的宽度；该特性的值可以是宽度值或者关键字。

```
div.page {border-width:2px;}
```

表 C-47

值	IE	N	FF
[长度]	4	4	1
thin	4	4	1
medium	4	4	1
thick	4	4	1
inherit	-	6	1

表 C-48

继承	否
默认值	medium
应用于	所有元素

border-color(border-bottom-color、border-left-color、border-top-color、border-right-color)

指定边框的颜色；该特性的值可以是颜色名、十六进制代码或 RGB 值。

```
table {border-color:#000000;}
```

表 C-49

值	IE	N	FF
[颜色值]	4	4	1
inherit	-	6	1

表 C-50

继承	否
默认值	无
应用于	所有元素

面积特性

面积特性用于指定框的大小。

height

指定块元素的垂直高度；可以缩放该元素。

```
table {height:400px;}
```

表 C-51

值	IE	N	FF
auto	4	6	1
[长度]	4	6	1
[百分比]	4	-	1
inherit	-	6	1

表 C-52

继承	否
默认值	auto
应用于	块级元素

width

指定该元素的水平宽度；可以缩放该元素。

```
td {width:150px;}
```

表 C-53

值	IE	N	FF
auto	4	4	1
[长度]	4	4	1
[百分比]	4	4	1
inherit	-	6	1

表 C-54

继承	否
默认值	auto
应用于	块级元素

line-height

指定文本行的高度以及行间距(多行文本之间的空间)。

```
p {line-height:18px;}
```

表 C-55

值	IE	N	FF
normal	3	4	1
[数值]	4	4	1
[长度]	3	4	1
[百分比]	3	4	1
inherit	-	6	1

表 C-56

继承	是
默认值	取决于具体的浏览器
应用于	所有元素

max-height

该特性指定块级元素的最大高度(值与 height 特性的值相同)。

```
td {max-height:200px;}
```

表 C-57

值	IE	N	FF
auto	7	-	1
[长度]	7	-	1
[百分比]	7	-	1
inherit	-	6	1

表 C-58

继承	否
默认值	auto
应用于	块级元素

max-width

该特性指定块级元素的最大宽度(值与 width 特性的值相同)。

```
td {max-width:400px;}
```

表 C-59

值	IE	N	FF
auto	7	-	1
[长度]	7	-	1
[百分比]	7	-	1
inherit	-	6	1

表 C-60

继承	否
默认值	auto
应用于	块级元素

min-height

该特性指定块级元素的最小高度(值与 height 特性的值相同)。

```
td {min-height:100px;}
```

表 C-61

值	IE	N	FF
auto	7	-	1
[长度]	7	-	1
[百分比]	7	-	1
inherit	-	6	1

表 C-62

继承	否
默认值	auto
应用于	块级元素

min-width

该特性指定块级元素的最小宽度(值与 width 特性的值相同)。

```
td {min-width:200px;}
```

表 C-63

值	IE	N	FF
auto	7	-	1
[长度]	7	-	1
[百分比]	7	-	1
inherit	-	6	1

表 C-64

继承	否
默认值	auto
应用于	块级元素

页边空白特性

页边空白特性用于指定框周围的页边空白，从而创建元素边框之间的距离。

margin (margin-bottom、margin-left、margin-top、margin-right)

该特性指定框周围页边空白的宽度。

```
p {margin:15px;}
```

表 C-65

值	IE	N	FF
auto	3	4	1
[长度]	3	4	1
[百分比 — 相对于父元素]	3	4	1
inherit	-	6	1

表 C-66

继承	否
默认值	0
应用于	所有元素

内边距特性

内边距特性设置元素边框和它的内容之间的距离。这些特性对于在文档中(特别是在表的单元格中)添加空白来说非常重要。

padding (padding-bottom、padding-left、padding-right、padding-top)

该特性指定元素的边框和它的内容之间的距离。

```
td {padding:20px;}
```

表 C-67

值	IE	N	FF
auto	4	4	1
[长度]	4	4	1
[百分比—相对于父元素]	4	4	1
inherit	-	6	1

表 C-68

继承	否
默认值	0
应用于	所有元素

列表特性

列表特性影响项目列表、编号列表和定义列表的表现。

list-style

这是一种简写方式，可用于同时指定 list-style-position 特性和 list-style-type 特性。

```
ul {list-style: inside disc}
```

表 C-69

值	IE	N	FF
<position>	4	6	1
<type>	4	4	1
<image>	4	6	1
inherit	-	6	1

表 C-70

继承	是
默认值	取决于具体的浏览器
应用于	列表元素

list-style-position

该特性指定列表的标记符应当放置在列表的每一项之内还是这些项的左边。

```
ul {list-style-position:inside;}
```

表 C-71

值	IE	N	FF
inside	4	6	1
outside	4	6	1
inherit	-	6	1

表 C-72

继承	是
默认值	outside
应用于	列表元素

list-style-type

该特性指示列表应该使用的项目符号或编号的类型。

```
ul {list-style-type:circle;}
```

表 C-73

值	IE	N	FF
None	4	4	1
disc(默认值)	4	4	1
Circle	4	4	1
square	4	4	1
decimal	4	4	1
decimal-leading-zero	-	-	-
lower-alpha	4	4	1
upper-alpha	4	4	1
lower-roman	4	4	1
upper-roman	4	4	1

表 C-74

继承	是
默认值	disc
应用于	列表元素

可以利用 CSS 添加额外的编号列表样式，但是，这些样式不被 IE7、Netscape 7 或 Firefox 2 浏览器支持。

表 C-75

hebrew	传统的希伯来语编号
georgian	传统的乔治亚语编号(an、ban、gan、...、he、tan、in、in-an、...)
armenian	传统的美语编号
cjk-ideographic	纯表意编号
hiragana	(a、i、u、e、o、ka、ki、...)
katakana	(A、I、U、E、O、KA、KI、...)
hiragana-iroha	(i、ro、ha、ni、ho、he、to、...)
katakana-iroha	(I、RO、HA、NI、HO、HE、TO、...)

marker-offset

该特性指定列表项和它的标记符之间的距离。

```
ol {marker-offset:2em;}
```

表 C-76

值	IE	N	FF
[长度]	-	7	1
auto	-	7	1
inherit	-	6	1

表 C-77

继承	否
默认值	auto
应用于	标记符元素

定位特性

定位特性允许您利用 CSS 定位框在页面中的位置。

position

指定用于某个元素的定位模式。当定位一个元素时，也需要使用本章后面介绍的框偏移特性(top、left、bottom 和 right)。注意，不能同时使用 top 和 bottom，也不能同时使用 left 和 right(如果同时使用，则 top 或 left 具有更高的优先级)。

- absolute，可以将该元素固定在工作区上相对于它的包含元素的某个特定位置(包含元素也是另外一个绝对定位元素)；当用户滚动页面时，该元素也将移动。
- static，将该元素固定在页面的同一个位置，即使用户滚动页面也保持在该位置。
- relative，该元素将被放置在距离它的正常位置具有一定偏移量的位置。
- fixed，将该元素固定在页面的背景上，当用户滚动页面时该元素不会移动。

```
p.article{position:absolute; top:10px; left:20px;
```

表 C-78

值	IE	N	FF
absolute	4	4	1
relative	4	4	1
static	4	4	1
fixed	-	6	1
inherit	-	6	1

表 C-79

继承	否
默认值	static
应用于	所有元素

Top

该特性设置元素相对于窗口或包含元素的顶部的垂直位置。

表 C-80

值	IE	N	FF
auto	4	6	1
[长度]	4	6	1
[百分比–相对于父元素的高度]	4	6	1
Inherit	-	6	1

表 C-81

继承	否
默认值	auto
应用于	定位元素

Left

该特性设置元素相对于窗口或包含元素的左边的水平位置。

表 C-82

值	IE	N	FF
Auto	4	6	1
[长度]	4	6	1
[百分比－相对于父元素的宽度]	4	6	1
inherit	-	6	1

表 C-83

继承	否
默认值	auto
应用于	定位元素

bottom

该特性设置元素相对于窗口或包含元素的底部的垂直位置。

表 C-84

值	IE	N	FF
auto	5	6	1
[长度]	5	6	1
[百分比－相对于父元素的高度]	5	6	1
inherit	-	6	1

表 C-85

继承	否
默认值	auto
应用于	定位元素

right

该特性设置元素相对于窗口或包含元素右边的水平位置。

表 C-86

值	IE	N	FF
auto	5	6	1
[长度]	5	6	1
[百分比－相对于父元素的宽度]	5	6	1
inherit	-	6	1

表 C-87

继承	否
默认值	auto
应用于	定位元素

vertical-align

该特性设置内联元素的垂直位置。

- baseline，将元素与它的父元素的基线对齐。
- middle，将元素的中点与它的父元素高度的一半对齐。
- sub，元素的内容将作为下标显示。
- super，元素的内容将作为上标显示。
- text-top，将元素与父元素的字体的顶部对齐。
- text-bottom，将元素与父元素的字体的底部对齐。
- top，将元素的顶部与当前行中最高元素的顶部对齐。
- bottom，将元素的底部与当前行中最低元素的顶部对齐。

```
span.superscript {vertical-align:superscript;}
```

表 C-88

值	IE	N	N
baseline	4	4	1
middle	4	4	1
sub	4	6	1
super	4	6	1
text-top	4	4	1
text-bottom	4	4	1
top	4	4	1
bottom	4	4	1
[百分比－相对于文本行高度]	-	6	1
[长度]	-	-	-
inherit	4	6	1

表 C-89

继承	否
默认值	baseline
应用于	内联元素

z-index

控制多个重叠元素中的哪一个元素将出现在顶部；仅适用于绝对定位元素。允许使用正数和负数。

```
p {position:absolute; top:10px; left:20px; z-index:3;}
```

表 C-90

值	IE	N	FF
auto	4	-	1
[数值]	4	4	1
inherit	-	6	1

表 C-91

继承	否
默认值	取决于元素在 XHTML 源文档中的位置
应用于	定位元素

clip

该特性控制元素的哪一部分可见。位于截取区域之外的部分将不可见。如果该特性的值是 rect()，则它的形式如下所示：

- rect([top] [right] [bottom] [left])

```
rect(25 100 100 25)
```

表 C-92

值	IE	N	FF
auto	4	-	1
rect	4	6	1
inherit	-	6	1

表 C-93

继承	否
默认值	auto
应用于	块级元素

overflow

当的内容太大以至于无法容纳在包含元素中时，该特性指定容器元素显示内容的方式。

```
p {width:200px; height:200px; overflow:scroll;}
```

表 C-94

值	IE	N	FF
auto	4	6	1
hidden	4	6	1
visible	4	6	1
scroll	4	6	1
inherit	-	6	6

表 C-95

继承	否
默认值	visible
应用于	块级元素

overflow-x

与 overflow 特性相同，但仅适用于水平 x 轴。该特性最先被 IE5 浏览器支持。

overflow-y

与 overflow 特性相同，但仅适用于垂直 y 轴。该特性最先被 IE5 浏览器所支持。

外边框特性

外边框类似于边框，但它不占用任何空间——它们位于工作区之上。

Outline (outline-color、outline-style、outline-width)

outline-color、outline-style 和 outline-width 特性的简写方式：

```
outline {solid #ff0000 2px}
```

注意，outline-color、outline-style 和 outline-width 特性采用的值与 border-color、border-style 和 border-width 特性相同。此处不单独介绍它们，因为它们还没有被任何浏览器支持。

表 C-96

值	IE	N	FF
outline-color	-	-	1.5
outline-style	-	-	1.5
outline-width	-	-	1.5
outline	-	-	1.5

表 C-97

继承	否
默认值	none
应用于	所有元素

表特性

表特性可用于影响表、行和单元格的样式。

border-collapse

该特性指定表使用的边框模式(相邻的边框是否折叠为一个边框的值，或者保持独立)。

```
table {border-collapse:separate;}
```

表 C-98

值	IE	N	FF
collapse	5	7	1
separate	5	7	1
inherit	-	6	1

表 C-99

继承	是
默认值	collapse
应用于	表和内联元素

border-spacing

该特性指定相邻单元格的边框之间的距离。

```
table {border-spacing:2px;}
```

表 C-100

值	IE	N	FF
[length]	-	6	1
inherit	-	6	1

表 C-101

继承	是
默认值	0
应用于	表和内联元素

caption-side

该特性指示标题应当放置在表的哪一边。

```
caption {caption-side:bottom;}
```

表 C-102

值	IE	N	FF
top	-	6	1
left	-	6	1
bottom	-	6	1
right	-	6	1
inherit	-	6	1

表 C-103

继承	是
默认值	top
应用于	<table>元素中的<caption>元素

empty-cells

如果单元格为空，该特性指定是否显示边框。

```
td, th {empty-cells:hide;}
```

表 C-104

值	IE	N	FF
show	5	6	1
hide	5	6	1
inherit	-	6	1

表 C-105

继承	是
默认值	show
应用于	表单元格元素

table-layout

指定浏览器如何计算表的布局；该特性能够影响显示大型的或图形密集的表的速度。

表 C-106

值	IE	N	FF
auto	5	6	1
fixed	5	6	1
inherit	-	6	6

表 C-107

继承	否
默认值	auto
应用于	表和内联元素

分类特性

分类特性影响框模型中框的显示方式。

clear

强制一些元素显示在它的下面，这些元素通常围绕在对齐元素周围。该特性的值表明元素的哪一边不能接触对齐元素。

```
p {clear:left;}
```

表 C-108

值	IE	N	FF
none	4	4	1
both	4	4	1
left	4	4	1
right	4	4	1
inherit	-	6	1

表 C-109

继承	否
默认值	none
应用于	所有元素

display

指定如何显示一个元素。如果将该特性设置为 none，则元素不显示并且不占用任何空间。可以强制内联元素作为块级元素显示，反之亦然。

```
span.important {display:block;}
```

表 C-110

值	IE	N	FF
none	4	4	1
inline	5	4	1

表 C-111

继承	是
默认值	inline
应用于	所有元素

(续表)

值	IE	N	FF
block	5	4	1
list-item	5	4	1
inherit	-	6	1

其他的一些特性或者不被任何浏览器支持，或者不是 XHTML 中必须具有的特性。

虽然这个特性的默认值是 inline，但是浏览器趋向于以其固有的显示类型处理元素。块级元素，例如题头和段落，通常被处理为如同默认值是 block；而内联元素，例如<i>、<b>或，则被处理为 inline。

float

随后的元素将换行到该元素的左边或者右边，而不是换行到下方。

```
img.featuredeItem {float:left;}
```

表 C-112

值	IE	N	FF
none	4	4	1
left	4	4	1
right	4	4	1
inherit	-	6	1

表 C-113

继承	否
默认值	none
应用于	所有元素

visibility

指定是否应当显示或者隐藏一个元素。即使隐藏元素，该元素也将占用页面中的空间，但是透明显示。

表 C-114

值	IE	N	FF
visible	4	-	1
show	-	4	1
hidden	4	-	1
hide	-	4	1
collapse	-	-	1
inherit	4	4	1

表 C-115

继承	否
默认值	inherit
应用于	所有元素

国际化特性

国际化特性影响以不同语言显示文本的方式。

direction

指定文本从左到右或者从右到左的方向。该特性应当与 unicode-bidi 特性结合使用。

```
td.word{direction:rtl; unicode-bidi:bidi-override;}
```

表 C-116

值	IE	N	FF
ltr	5	6	1
rtl	5	6	1
inherit	5	6	1

表 C-117

继承	是
默认值	ltr
应用于	所有元素

unicode-bidi

unicode-bidi 特性可用于重写 Unicode 中用于语言的内置方向性设置。

```
td.word{unicode-bidi:bidi-override; direction:rtl; }
```

表 C-118

值	IE	N	FF
normal	5	-	2
embed	5	-	2
bidi-override	5	-	2
inherit	-	6	2

表 C-119

继承	否
默认值	normal
应用于	所有元素

长度

下面的内容是关于 CSS 中使用的长度度量单位。

绝对长度

表 C-120

单位	IE	N	FF
cm	3	4	1
in	3	4	1
mm	3	4	1
pc	3	4	1
pt	3	4	1

相对长度

表 C-121

单位	IE	N	FF
em	4	4	1
ex	4	4	1
px	3	4	1

附录 D

颜色名和颜色值

对于颜色，首先需要掌握的是如何指定所需的颜色；毕竟，存在很多种不同的红、绿和蓝等颜色，选择一种正确的颜色非常重要。

在 XHTML 中，存在两种指定颜色的关键方式：

- 十六进制代码：一种 6 位数字的代码，分别表示组成颜色的红、绿、蓝值，并且数字前面具有一个英镑符号或井号(例如#333333)。
- 颜色名：一组颜色名，表示超过 200 种的颜色，例如 red、lightslategray 和 fuchsia。

在 CSS 中，也可以使用一些值来表示组成每一种颜色的红、绿和蓝值。

使用十六进制代码指定颜色

刚开始使用十六进制代码时，您可能会感到胆怯。这种代码的思想是使用数字和字母的混合方式表示颜色，这种细想看上去有点奇怪，但是#号后面的内容实际上是组成颜色的红、绿、蓝值。十六进制代码的格式如下所示：

```
# rrggbb
```

表 D-1 提供了一些示例。

表 D-1

颜　色	十六进制代码	颜　色	十六进制代码
黑	#000000	绿	#008000
白	#FFFFFF	蓝	#0000FF
红	#FF0000	紫	#800080

计算机显示器使用的颜色空间称为 RGB 颜色空间。当计算机显示器没有打开时，屏幕是黑色的，因为它不发射任何颜色。为了在屏幕上创建图像，组成屏幕的每个像素发射不同数量的红、绿、蓝颜色值，如同电视屏幕一样。

因此毫无疑问的是，可以通过指定组成给定颜色所需的红、绿、蓝值来指定颜色。组成一种颜色所需的红、绿、蓝值位于 0 到 255 之间，因此当红、绿、蓝值都是 0 时，将获

得黑色；而如果红、绿、蓝值都是 255，将获得白色。

某些软件使用 0 到 255 之间的 3 组数值来表示颜色。图 D-1 给出了 Adobe Photoshop 软件中的颜色窗口。

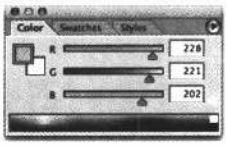


图 D-1

Web 中用于颜色的十六进制代码是 0 到 255 之间的这些值的直接转换，不同之处是它们使用两个字符(而不是 3 个字符)表示 0 到 255 之间的数值。例如，FF 表示 255，00 表示 0。

实际理解十六进制代码工作原理的最佳方法是了解计算机存储信息的方式。

理解十六进制代码

计算机以 0 和 1 存储所有信息，虽然这听起来令人难以置信，但事实就是如此！计算机中存储的最小信息单元是位，一个位的值只能是 0 或 1：

- 0，意味着关闭(或假)
- 1，意味着打开(或真)

单独利用这两个值无法存储很多信息，但是如果将 4 个位组合在一起，则可以获得 16 种不同的值。例如，使用 4 个 0 和 1 的组合，可以表示数字 0 到 9(并且还有一些空余的值)：

```
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
  0    1    2    3    4    5    6    7    8    9    -    -    -    -    -    -
```

可以使用单个十六进制数字替换 4 个位。在十六进制记数法中共有 16 个数字，它们分别表示由 4 个 0 或 1 组成的 16 种可能的值：

```
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
  0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
```

其中，0 是最小的值；F 是最大的值。

但是，计算机需要处理的值超过 16 种，因此它们需要以更大的段来存储信息。一组 8 个位称为一个字节。因此，一个字节可以使用两个十六进制数字表示。例如：

```
二进制       0100  1111
十六进制       4    F
```

这就给出了 0 和 1 的 256 种可能组合，足够表示英语中的所有字符，并且这是采用 0 到 255 之间的数值表示颜色的原因。

因此，虽然用于 Web 颜色的十六进制代码可能有点复杂，但是#4F4F4F 无疑比 010011110100111101001111 更方便阅读。表 D-2 给出了一些十六进制代码和它们对应的十进制数值。

表 D-2

十六进制	十进制	十六进制	十进制
00	0	BB	187
33	51	CC	204
66	102	DD	221
99	153	EE	238
AA	170	FF	255

利用颜色名指定颜色

除了使用十六进制值指定颜色之外，也可以使用颜色名指定所需的颜色，例如 red、green 和 white。Netscape、Firefox 和 IE 浏览器支持超过 200 种不同的颜色名，本附录的末尾列举了所有这些颜色名。

尽管颜色名可能比十六进制代码更容易理解，但是某些颜色名比起其他颜色名更容易记忆，并且要求记住 200 种颜色名中每一种颜色的效果是一种无礼的要求。下面是一些颜色名：

```
aqua, beige, coral, darkcyan, firebrick, green, honeydew, indianred,
lavenderblush, maroon, navy, oldlace, palegreen, red, saddlebrown,
tan, white, yellow
```

另外，如果是为较大规模的公司工作，这些公司通常希望指定非常精确的颜色来表示它们的商标，并且他们的颜色可能没有 HTML 名称。事实上，当客户指定所需的颜色时，他们通常会指定十六进制代码。

十六进制代码与颜色名的对比

相对于十六进制代码，颜色名的使用似乎更为直观；如果使用 red、orange、green、blue、black 和 white 等颜色，则很容易记住并使用它们。但是，记住每一种颜色名以及它们提供的颜色是非常困难的。

事实上，无论是利用十六进制代码还是颜色名，通常最终都会参考颜色图表来找到所需的颜色。由于十六进制代码在颜色的阴影、色彩和色调方面能够提供比颜色名更多的选择，并且许多公司要求使用特定的颜色来表示他们的公司，所以十六进制代码是 Web 专家的首选。

如果使用的是一种图形设计程序或者一种 Web 页面制作工具，则该程序通常会为您生成所需的颜色代码，并且许多图形程序包还提供一种颜色选取工具，可以帮助您选择所需的确切颜色。一些 Web 站点中也提供颜色选取工具，例如 www.visibone.com/colorlab/。图 D-2 给出了 Photoshop 的颜色选取工具。

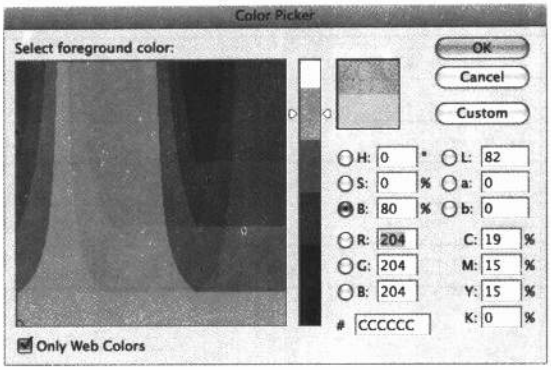


图 D-2

注意，该窗口左下角中的复选框提供了一个选项，指示是否仅使用能够安全用于 Web 的颜色。选择该选项之后，将使用一种受限制的调色板(它包含所有可用颜色的子集)。该调色板称为 Web Safe Color Palette，它是在计算机还不能支持很多种颜色的年代设计的调色板。现在，大多数计算机能够处理超过 Web Safe Color Palette 中指定的 256 种颜色，因此可以安全地忽略该选项。

颜色名和数值参考

表 D-3 给出了 HTML3.2 中引入的 16 种颜色名，它们用于支持 8 位图形卡提供的 16 种颜色。

表 D-3

颜 色 名	十六进制值	颜 色 名	十六进制值
aqua	#00ffff	navy	#000080
black	#000000	olive	#808000
blue	#0000ff	purple	#800080
fuchsia	#ff00ff	red	#ff0000
green	#008000	silver	#c0c0c0
gray	#808080	teal	#008080
lime	#00ff00	white	#ffffff
maroon	#800000	yellow	#ffff00

表 D-4 中列举的所有颜色可以应用于 IE 浏览器，Netscape 浏览器和 Firefox 浏览器也支持其中的大部分颜色。但是，它们是浏览器扩展的颜色，而不是 HTML 或 XHTML 规范的一部分。

表 D-4

颜 色 名	十六进制值	颜 色 名	十六进制值
aliceblue	#f0f8ff	black	#000000
antiquewhite	#faebd7	blanchedalmond	#ffebcd
aqua	#00ffff	blue	#0000ff
aquamarine	#7fffd4	blueviolet	#8a2be2
azure	#f0ffff	brown	#a52a2a
beige	#f5f5dc	burlywood	#deb887
bisque	#ffe4c4	cadetblue	#5f9ea0
chartreuse	#7fff00	darkturquoise	#00ced1
chocolate	#d2691e	darkviolet	#9400d3
coral	#ff7f50	deeppink	#ff1493
cornflowerblue	#6495ed	deepskyblue	#00bfff
cornsilk	#fff8dc	dimgray	#696969
crimson	#dc143c	dodgerblue	#1e90ff
cyan	#00ffff	firebrick	#b22222
darkblue	#00008b	floralwhite	#fffaf0
darkcyan	#008b8b	forestgreen	#228b22
darkgoldenrod	#b8860b	fuchsia	#ff00ff
darkgray	#a9a9a9	gainsboro	#dcdcdc
darkgreen	#006400	ghostwhite	#f8f8ff
darkkhaki	#bdb76b	gold	#ffd700
darkmagenta	#8b008b	goldenrod	#daa520
darkolivegreen	#556b2f	gray	#808080
darkorange	#ff8c00	green	#008000
darkorchid	#9932cc	greenyellow	#adff2f
darkred	#8b0000	honeydew	#f0fff0
darksalmon	#e9967a	hotpink	#ff69b4
darkseagreen	#8fbc8f	indianred	#cd5c5c
darkslateblue	#483d8b	indigo	#4b0082
darkslategray	#2f4f4f	ivory	#fffff0
khaki	#f0e68c	maroon	#800000
lavender	#e6e6fa	mediumaquamarine	#66cdaa
lavenderblush	#fff0f5	mediumblue	#0000cd
lawngreen	#7cfc00	mediumorchid	#ba55d3

(续表)

颜 色 名	十六进制值	颜 色 名	十六进制值
lemonchiffon	#fffacd	mediumpurple	#9370db
lightblue	#add8e6	mediumseagreen	#3cb371
lightcoral	#f08080	mediumslateblue	#7b68ee
lightcyan	#e0ffff	mediumspringgreen	#00fa9a
lightgoldenrodyellow	#fafad2	mediumturquoise	#48d1cc
lightgreen	#90ee90	mediumvioletred	#c71585
lightgrey	#d3d3d3	midnightblue	#191970
lightpink	#ffb6c1	mintcream	#f5fffa
lightsalmon	#ffa07a	mistyrose	#ffe4e1
lightseagreen	#20b2aa	moccasin	#ffe4b5
lightskyblue	#87cefa	navajowhite	#ffdead
lightslategray	#778899	navy	#000080
lightsteelblue	#b0c4de	oldlace	#fdf5e6
lightyellow	#ffffe0	olive	#808000
lime	#00ff00	olivedrab	#6b8e23
limegreen	#32cd32	orange	#ffa500
linen	#faf0e6	orangered	#ff4500
magenta	#ff00ff	orchid	#da70d6
palegoldenrod	#eee8aa	sienna	#a0522d
palegreen	#98fb98	silver	#c0c0c0
paleturquoise	#afeeee	skyblue	#87ceeb
palevioletred	#db7093	slateblue	#6a5acd
papayawhip	#ffefd5	slategray	#708090
peachpuff	#ffdab9	snow	#fffafa
peru	#cd853f	springgreen	#00ff7f
pink	#ffc0cb	steelblue	#4682b4
plum	#dda0dd	tan	#d2b48c
powderblue	#b0e0e6	teal	#008080
purple	#800080	thistle	#d8bfd8
red	#ff0000	tomato	#ff6347
rosybrown	#bc8f8f	turquoise	#40e0d0
royalblue	#4169e1	violet	#ee82ee
saddlebrown	#8b4513	wheat	#f5deb3

(续表)

颜 色 名	十六进制值	颜 色 名	十六进制值
salmon	#fa8072	white	#ffffff
sandybrown	#f4a460	whitesmoke	#f5f5f5
seagreen	#2e8b57	yellow	#ffff00
seashell	#fff5ee	yellowgreen	#9acd32

附录 E

字 符 编 码

在附录 D 中讨论了计算机如何存储信息，字符编码方案如何是一个表(该表在字符字间进行转换)，以及字符如何存储在计算机中。

计算机中最常使用的字符集(或字符代码)是 ASCII(American Standard Code for Information Interchange，美国信息交换标准代码)，并且它很可能是用于以电子方式编码文本的最常用字符集。可以认为所有浏览 Web 的计算机都能够理解 ASCII 字符集。

表 E-1

字 符 集	描　述
ASCII	美国信息交换标准代码，大多数计算机使用这种代码

使用 ASCII 字符集的问题是它仅支持大写和小写拉丁字母、数字 0～9 以及一些额外的字符：总共 128 个字符。下面是 ASCII 字符集的可打印字符(其他字符是类似于换行或回车的字符)。

	!	``	#	$	%	&	`	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

但是，许多语言使用带声调的拉丁字符或者完全不同的字母表。而 ASCII 字符集不支持这些字符，因此如果希望使用任何非 ASCII 字符集，则需要了解一些其他字符编码。

如果希望使用一些符号，则字符编码也非常重要，因为无法保证在不同的编码之间正确转换这些符号(从一些破折号到一些引号字符)。如果不指明编写文档的字符编码，则可能无法显示某些特殊字符。

国际标准化组织(International Standards Orgarnization，ISO)创建了一个字符集范围，用于处理不同国家的字符。其中 ISO-8859-1 经常用于 Web 页面制作工具(例如 Macromedia Dreamweaver)的西方版本以及类似于 Windows Notepad 的应用程序中。

表 E-2

字符集	描述
ISO-8859-1	拉丁字母表 1，它覆盖了北美、西欧、拉丁美洲、加勒比海、加拿大、非洲等地区使用的字符
ISO-8859-2	拉丁字母表 2，它覆盖了东欧使用的字符(包括波斯尼亚、克罗地亚、捷克、匈牙利、波兰、罗马尼亚、塞尔维亚(拉丁文)、塞尔维亚克罗埃西亚、斯洛伐克、斯洛文尼亚、上索布文和下索布文)
ISO-8859-3	拉丁字母表 3，它覆盖了欧洲东南部、世界语、马耳他、土耳其以及一些其他地区使用的字符
ISO-8859-4	拉丁字母表 4，它覆盖了斯堪的那维亚半岛和波罗的海等地区使用的字符(以及一些其他不在 ISO-8859-1 中的字符)
ISO-8859-5	拉丁/古斯拉夫语字母表 5
ISO-8859-6	拉丁/阿拉伯语字母表 6
ISO-8859-7	拉丁/希腊语字母表 7
ISO-8859-8	拉丁/希伯来语字母表 8
ISO-8859-9	拉丁 5 字母表 9(与 ISO-8859-1 类似，除了土耳其语字符替代冰岛语字符)
ISO-8859-10	拉丁 6 拉普兰语、日耳曼语和爱斯基摩语
ISO-8859-15	与 ISO-8859-1 相同，但是添加了更多的字符
ISO-8859-16	拉丁字母表 10，它覆盖了欧洲东南部使用的字符(阿尔巴尼亚、克罗地亚、匈牙利、波兰、罗马尼亚和斯洛文尼亚)，加上可用于法国、德国、意大利和爱尔兰语中的字符。
ISO-2022-JP	拉丁/日本字母表 1
ISO-2022-JP-2	拉丁/日本字母表 2
ISO-2022-KR	拉丁/韩国字母表 1

需要注意的是，ISO-8859-1 中前 128 个字符匹配 ASCII 中的相应字符，因此可以像使用 ASCII 字符一样安全地使用它们。

然后，Unicode 联盟成立并设计了一种能够显示不同语言的所有字符的方式，而不是让这些不同的、不兼容的字符代码用于不同的语言。

因此，如果希望创建一个使用多种字符集中字符的文档，则可以仅使用一种 Unicode 字符编码。此外，如果用户的处理程序(和字体)支持 Unicode 标准，则他们能够查看以不同字符集编写的文档，而与他们使用的平台和所在的国家无关。通过只使用一种字符编码，可以降低软件开发成本，因为不需要通过设计程序来支持多种字符编码。

使用 Unicode 存在的一种问题是许多较老的程序仅支持 8 位的字符集(字符数量限制为 256 个字符)，这无法满足有语言所需要的字符数量。

因此 Unicode 指定一些编码，这些编码能够以特殊的方式处理字符串，从而为它包含的大字符集预留足够的空间。这些编码称为 UTF-8、UTF-16 和 UTF-32，表 E-3 说明了这些编码。

表 E-3

字 符 集	描　述
UTF-8	一种 Unicode 转换格式，由 8 位的单元组成，即它由字节组成。UTF-8 中的字符长度可以是 1 到 4 个字节长，因此 UTF-8 字符具有不同的宽度
UTF-16	一种 Unicode 转换格式，由 16 位的单元组成，即它由短字组成。UTF-16 中的字符长度可以是 1 到 2 个短字，因此 UTF-8 字符具有不同的宽度
UTF-32	一种 Unicode 转换格式，由 32 位的单元组成，即它由长字组成。UTF-32 中的字符具有固定宽度格式，它的长度始终是 1 个长字

Unicode 字符集中的前 256 个字符对应于 ISO-8859-1 中的 256 个字符。

默认情况下，HTML 4 处理程序应当支持 UTF-8，XML 处理程序应该支持 UTF-8 和 UTF-16；因此，所有 XHTML 兼容的处理程序应当也支持 UTF-16(因为 XHTML 是 XML 的一种应用)。

关于国际化以及不同字符集和编码的更多信息，请查看 www.i18nguy.com/。

特 殊 字 符

XHTML 中的某些字符是保留字符；例如，不能在 XHTML 文本中使用大于号、小于号或尖括号，因为浏览器会错误地将它们当作为标识。XHTML 处理程序必须支持表 F-1 中列举的 5 个特殊字符。

表 F-1

符　号	描　述	实 体 名	数 字 代 码
&	“与”符号	&amp;	&#38;
<	小于号	&lt;	&#60;
>	大于号	&gt;	&#62;
“	双引号	&quot;	&#34;
	不间断空格		

为了将元素和属性编写入页面，以便向用户显示代码而不是由浏览器处理(例如，希望写入)，应当按如下方式编写：

```
&lt;div id=&quot;character&quot;&gt;
```

也存在一个非常长的特殊字符列表，支持 HTML 4.0 的处理程序应该支持这些特殊字符。为了让这些特殊字符出现在文档中，既可以使用数字代码，也可以使用实体名。例如，为了插入一个版权符号，可以使用如下两种方式中的一种：

```
&copy; 2008
&#169; 2008
```

这些特殊字符拆分为如下几个部分：

- 用于 ISO 8859-1 字符的字符实体参考
- 用于符号、数学符号和希腊字母的字符实体参考
- 用于重要的国际化字符标记的字符实体参考

这些字符实体参考来源于 W3C 的 Web 站点 www.w3.org/TR/REC-html40/sgml/entities.html。

表 F-2

用于 ISO 8859-1 字符的字符实体参考

实体名	符　号	数字代码	描　述
			不间断空格
&iexcl;	¡	&#161;	反感叹号
&cent;	¢	&#162;	分的符号
&pound;	£	&#163;	英镑符号
&curren;	¤	&#164;	货币符号
&yen;	¥	&#165;	人民币符号＝元符号
&brvbar;	¦	&#166;	断竖线
&sect;	§	&#167;	章节符号
&uml;	¨	&#168;	分音符号
&copy;	©	&#169;	版权符号
&ordf;	ª	&#170;	阴性序指示器
&laquo;	«	&#171;	左双尖引号=左书名号
&not;	¬	&#172;	非 符号
&shy;	SHY	&#173;	软连字符＝自由选定连字符
&reg;	®	&#174;	注册符号＝注册商标符号
&macr;	¯	&#175;	长音符号＝间隔长音符号＝上划线＝APL 上划线
&deg;	°	&#176;	度数符号
&plusmn;	±	&#177;	加减符号＝加或减符号
&sup2;	2	&#178;	上标 2＝上标数字 2＝平方
&sup3;	3	&#179;	上标 3＝上标数字 3＝立方
&acute;	´	&#180;	尖音符号＝间隔尖音
&micro;	µ	&#181;	微符号
&para;	¶	&#182;	段落符号
&middot;	·	&#183;	中间点＝格鲁吉亚逗号＝希腊中间点
&cedil;	¸	&#184;	变音符号＝间隔变音
&sup1;	1	&#185;	上标 1=上标数字 1
&ordm;	º	&#186;	阴性序指示器
&raquo;	»	&#187;	右双尖引号=右书名号
&frac14;	¼	&#188;	普通分数四分之一＝分数四分之一
&frac12;	½	&#189;	普通分数二分之一＝分数二分之一
&frac34;	¾	&#190;	普通分数四分之三＝分数四分之三
&iquest;	¿	&#191;	反问号＝翻转问号

(续表)

用于 ISO 8859-1 字符的字符实体参考

实体名	符号	数字代码	描述
À	À	À	具有抑音符号的拉丁大写字母 A
Á	Á	Á	具有尖音符号的拉丁大写字母 A
Â	Â	Â	具有长音符号的拉丁大写字母 A
Ã	Ã	Ã	具有颚化符号的拉丁大写字母 A
Ä	Ä	Ä	具有分音符号的拉丁大写字母 A
Å	Å	Å	上方具有圆环符号的拉丁大写字母 A
Æ	Æ	Æ	拉丁大写字母 AE=拉丁大写连字 AE
Ç	Ç	Ç	具有变音符号的拉丁大写字母 C
È	È	È	具有抑音符号的拉丁大写字母 E
É	É	É	具有尖音符号的拉丁大写字母 E
Ê	Ê	Ê	具有长音符号的拉丁大写字母 E
Ë	Ë	Ë	具有分音符号的拉丁大写字母 E
Ì	Ì	Ì	具有抑音符号的拉丁大写字母 I
Í	Í	Í	具有尖音符号的拉丁大写字母 I
Î	Î	Î	具有长音符号的拉丁大写字母 I
Ï	Ï	Ï	具有分音符号的拉丁大写字母 I
Ð	Ð	Ð	拉丁大写字母 ETH
Ñ	Ñ	Ñ	具有颚化符号的拉丁大写字母 N
Ò	Ò	Ò	具有抑音符号的拉丁大写字母 O
Ó	Ó	Ó	具有尖音符号的拉丁大写字母 O
Ô	Ô	Ô	具有长音符号的拉丁大写字母 O
Õ	Õ	Õ	具有颚化符号的拉丁大写字母 O
Ö	Ö	Ö	具有分音符号的拉丁大写字母 O
×	×	×	乘号
Ø	Ø	Ø	具有斜线的拉丁大写字母 O
Ù	Ù	Ù	具有抑音符号的拉丁大写字母 U
Ú	Ú	Ú	具有尖音符号的拉丁大写字母 U
Û	Û	Û	具有长音符号的拉丁大写字母 U
Ü	Ü	Ü	具有分音符号的拉丁大写字母 U
Ý	Ý	Ý	具有尖音符号的拉丁大写字母 Y
Þ	Þ	Þ	拉丁大写字母 THORN
ß	ß	ß	拉丁小写字母 Sharp s

(续表)

用于 ISO 8859-1 字符的字符实体参考

实体名	符号	数字代码	描述
à	à	à	具有抑音符号的拉丁小写字母 a
á	á	á	具有尖音符号的拉丁小写字母 a
â	â	â	具有长音符号的拉丁小写字母 a
ã	ã	ã	具有颚化符号的拉丁小写字母 a
ä	ä	ä	具有分音符号的拉丁小写字母 a
å	å	å	上方具有圆环符号的拉丁小写字母 a
æ	æ	æ	拉丁小写字母 ae
ç	ç	ç	具有变音符号的拉丁小写字母 c
è	è	è	具有抑音符号的拉丁小写字母 e
é	é	é	具有尖音符号的拉丁小写字母 e
ê	ê	ê	具有长音符号的拉丁小写字母 e
ë	ë	ë	具有分音符号的拉丁小写字母 e
ì	ì	ì	具有抑音符号的拉丁小写字母 i
í	í	í	具有尖音符号的拉丁小写字母 i
î	î	î	具有长音符号的拉丁小写字母 i
ï	ï	ï	具有分音符号的拉丁小写字母 i
ð	ð	ð	拉丁小写字母 eth
ñ	ñ	ñ	具有颚化符号的拉丁小写字母 n
ò	ò	ò	具有抑音符号的拉丁小写字母 o
ó	ó	ó	具有尖音符号的拉丁小写字母 o
ô	ô	ô	具有长音符号的拉丁小写字母 o
õ	õ	õ	具有颚化符号的拉丁小写字母 o
ö	ö	ö	具有分音符号的拉丁小写字母 o
÷	÷	÷	除号
ø	ø	ø	具有斜线的拉丁小写字母 o
ù	ù	ù	具有抑音符号的拉丁小写字母 u
ú	ú	ú	具有尖音符号的拉丁小写字母 u
û	û	û	具有长音符号的拉丁小写字母 u
ü	ü	ü	具有分音符号的拉丁小写字母 u
ý	ý	ý	具有尖音符号的拉丁小写字母 y
þ	þ	þ	拉丁小写字母 thorn
ÿ	ÿ	ÿ	具有分音符号的拉丁小写字母 y

表 F-3

用于符号、数学符号和希腊字母的字符实体参考

实体名	符号	数字代码	描述
拉丁扩展 B			
ƒ	f	ƒ	具有钩状符号的拉丁小写字母 f=函数=弗罗林货币
希腊字母			
Α	Α	Α	希腊大写字母 alpha
Β	Β	Β	希腊大写字母 beta
Γ	Γ	Γ	希腊大写字母 gamma
Δ	Δ	Δ	希腊大写字母 delta
Ε	Ε	Ε	希腊大写字母 epsilon
Ζ	Ζ	Ζ	希腊大写字母 zeta
Η	Η	Η	希腊大写字母 eta
Θ	Θ	Θ	希腊大写字母 theta
Ι	Ι	Ι	希腊大写字母 iota
Κ	Κ	Κ	希腊大写字母 kappa
Λ	Λ	Λ	希腊大写字母 lambda
&Mu	Μ	Μ	希腊大写字母 mu
Ν	Ν	Ν	希腊大写字母 nu
Ξ	Ξ	Ξ	希腊大写字母 xi
Ο	Ο	Ο	希腊大写字母 omicron
Π	Π	Π	希腊大写字母 pi
Ρ	Ρ	Ρ	希腊大写字母 rho
Σ	Σ	Σ	希腊大写字母 sigma
Τ	Τ	Τ	希腊大写字母 tau
Υ	Υ	Υ	希腊大写字母 upsilon
Φ	Φ	Φ	希腊大写字母 phi
Χ	Χ	Χ	希腊大写字母 chi
Ψ	Ψ	Ψ	希腊大写字母 psi
Ω	Ω	Ω	希腊大写字母 omega
α	α	α	希腊小写字母 alpha
β	β	β	希腊小写字母 beta
γ	γ	γ	希腊小写字母 gamma
δ	δ	δ	希腊小写字母 delta
ε	ε	ε	希腊小写字母 epsilon

(续表)

用于符号、数学符号和希腊字母的字符实体参考

实体名	符号	数字代码	描述
ζ	ζ	ζ	希腊小写字母 zeta
η	η	η	希腊小写字母 eta
θ	θ	θ	希腊小写字母 theta
ι	ι	ι	希腊小写字母 iota
κ	κ	κ	希腊小写字母 kappa
λ	λ	λ	希腊小写字母 lambda
μ	μ	μ	希腊小写字母 mu
ν	ν	ν	希腊小写字母 nu
ξ	ξ	ξ	希腊小写字母 xi
ο	ο	ο	希腊小写字母 omicron
π	π	π	希腊小写字母 pi
ρ	ρ	ρ	希腊小写字母 rho
ς	ς	ς	希腊小写字母 final sigma
σ	σ	σ	希腊小写字母 sigma
τ	τ	τ	希腊小写字母 tau
υ	υ	υ	希腊小写字母 upsilon
φ	φ	φ	希腊小写字母 phi
χ	χ	χ	希腊小写字母 chi
ψ	ψ	ψ	希腊小写字母 psi
ω	ω	ω	希腊小写字母 omega
ϑ	ϑ	ϑ	希腊小写字母 theta 符号
ϒ	ϒ	ϒ	具有钩状符号的希腊 upsilon
ϖ	ϖ	ϖ	希腊 pi 符号
通用标点符号			
•	•	•	项目符号＝黑色小圆
…	…	…	水平省略号＝三个点的前导字符
′	′	′	单引号＝分钟＝英尺
″	″	″	双引号＝秒＝英寸
‾	‾	‾	上划线＝间隔划线
⁄	⁄	⁄	反斜杠

表 F-3

字母式符号			
℘	℘	℘	手写体大写 P＝幂集＝维尔斯特拉斯函数 p
ℑ	ℑ	ℑ	黑体字大写 I＝虚数部分符号
ℜ	ℜ	ℜ	黑体字大写 R＝实数部分符号
™	™	™	商标符号
ℵ	ℵ	ℵ	Alef 符号＝第一个超穷基数
箭头			
←	←	←	左箭头
↑	↑	↑	上箭头
→	→	→	右箭头
↓	↓	↓	下箭头
↔	↔	↔	左右箭头
↵	↵	↵	具有左向拐角的下箭头＝回车
⇐	⇐	⇐	左双箭头
⇑	⇑	⇑	上双箭头
⇒	⇒	⇒	右双箭头
⇓	⇓	⇓	下双箭头
⇔	⇔	⇔	左右双箭头
数学运算符			
∀	∀	∀	尽管
&part ;	∂	∂	偏微分
∃	∃	∃	存在
∅	∅	∅	空集＝零集＝直径
∇	∇	∇	微分算符＝反向差分
∈	∈	∈	属于
∉	∉	∉	不属于
∋	∋	∋	包含
∏	∏	∏	N 元乘＝乘号
∑	∑	∑	N 元加
−	−	−	减号

(续表)

字母式符号			
∗	*	∗	星号运算符
√	√	√	平方根＝根号
∝	∝	∝	与...成比例
∞	∞	∞	无限
∠	∠	∠	角
∧	∧	∧	逻辑与＝楔形
&or ;	∨	∨	逻辑或＝V 字形
∩	∩	∩	交集＝帽形
∪	∪	∪	并集＝杯形
∫	∫	∫	积分
∴	∴	∴	因此
∼	~	∼	否定运算符＝与...不同＝相似于
≅	≅	≅	近似等于
≈	≈	≈	几乎等于＝渐进于
≠	≠	≠	不等于
≡	≡	≡	恒等于
≤	≤	≤	小于或等于
≥	≥	≥	大于或等于
⊂	⊂	⊂	子集
⊃	⊃	⊃	超集
⊄	⊄	⊄	非子集
⊆	⊆	⊆	子集或等于
⊇	⊇	⊇	超集或等于
⊕	⊕	⊕	圈加＝直接和
⊗	⊗	⊗	圈乘＝向量乘
⊥	⊥	⊥	正交于＝垂直于＝铅垂线
⋅	·	⋅	点运算符
其他技术符号			
⌈	⌈	⌈	左上限＝APL 上阶梯符号
⌉	⌉	⌉	右上限
⌊	⌊	⌊	左下限＝APL 下阶梯符号
⌋	⌋	⌋	右下限
⟨	〈	〈	左尖括号

（续表）

字母式符号			
⟩	〉	〉	右尖括号
几何形状			
◊	◇	◊	菱形
其他符号			
♠	♠	♠	黑色黑桃牌
♣	♣	♣	黑色梅花牌＝三叶草
♥	♥	♥	黑色红桃牌＝情人
♦	♦	♦	黑色方块牌

表 F-4

用于重要的国际化字符标记的字符实体参考

实体名	符号	数字代码	描述
"	"	"	双引号＝APL 引号
&	&	&	与
<	<	<	小于号
>	>	>	大于号
Œ	Œ	Œ	拉丁大写连字 OE
œ	œ	œ	拉丁小写连字 oe
Š	Š	Š	具有倒折音符号的拉丁大写字母 S
š	š	š	具有倒折音符号的拉丁小写字母 s
Ÿ	Ÿ	Ÿ	具有分音符号的拉丁大写字母 Y
空白修饰字母			
ˆ	ˆ	ˆ	修饰符字母抑扬音符号
˜	˜	˜	小颚化符号
通用标点符号			
			单倍间距
			双倍间距
			窄空格
‌		‌	零宽度非连接符

(续表)

用于重要的国际化字符标记的字符实体参考

实 体 名	符 号	数 字 代 码	描 述
&zwj;		&#8205;	零宽度连接符
&lrm;		&#8206;	从左到右标记
&rlm;		&#8207;	从右到左标记
&ndash;	–	&#8211;	短破折号
&mdash;	—	&#8212;	长破折号
&lsquo;	'	&#8216;	左单引号
&rsquo;	'	&#8217;	右单引号
&sbquo;	'	&#8218;	单 low-9 引号
&ldquo;	"	&#8220;	左双引号
&rdquo;	"	&#8221;	右双引号
&bdquo;	"	&#8222;	双 low-9 引号
&dagger;	†	&#8224;	剑形符号
&Dagger;	‡	&#8225;	双剑形符号
&permil;	‰	&#8240;	千分率号
&lsaquo;	‹	&#8249;	左单尖引号(建议的符号，但没有标准化)
&rsaquo;	›	&#8250;	右单尖引号(建议的符号，但没有标准化)
&euro;	€	&#8364;	欧元符号

附录 G

语言代码

表 G-1 给出了两个字母的 ISO 639 语言代码，这些代码用于 lang 和 xml:lang 属性中以声明文档的语言。该表涵盖了世界上的许多主要语言。

表 G-1

国　家	ISO 代码	国　家	ISO 代码
Abkhazian	AB	Bengali; Bangla	BN
Afan (Oromo)	OM	Bhutani	DZ
Afar	AA	Bihari	BH
Afrikaans	AF	Bislama	BI
Albanian	SQ	Breton	BR
Amharic	AM	Bulgarian	BG
Arabic	AR	Burmese	MY
Armenian	HY	Byelorussian	BE
Assamese	AS	Cambodian	KM
Aymara	AY	Catalan	CA
Azerbaijani	AZ	Chinese	ZH
Bashkir	BA	Corsican	CO
Basque	EU	Croatian	HR
Czech	CS	Icelandic	IS
Danish	DA	Indonesian	ID
Dutch	NL	Interlingua	IA
English	EN	Interlingue	IE
Esperanto	EO	Inuktitut	IU
Estonian	ET	Inupiak	IK
Faroese	FO	Irish	GA
Fiji	FJ	Italian	IT
Finnish	FI	Japanese	JA

(续表)

国　家	ISO 代码	国　家	ISO 代码
French	FR	Javanese	JV
Frisian	FY	Kannada	KN
Galician	GL	Kashmiri	KS
Georgian	KA	Kazakh	KK
German	DE	Kinyarwanda	RW
Greek	EL	Kirghiz	KY
Greenlandic	KL	Kurundi	RN
Guarani	GN	Korean	KO
Gujarati	GU	Kurdish	KU
Hausa	HA	Laothian	LO
Hebrew	HE	Latin	LA
Hindi	HI	Latvian; Lettish	LV
Hungarian	HU	Lingala	LN
Lithuanian	LT	Romanian	RO
Macedonian	MK	Russian	RU
Malagasy	MG	Samoan	SM
Malay	MS	Sangho	SG
Malayalam	ML	Sanskrit	SA
Maltese	MT	Scots Gaelic	GD
Maori	MI	Serbian	SR
Marathi	MR	Serbo-Croatian	SH
Moldavian	MO	Sesotho	ST
Mongolian	MN	Setswana	TN
Nauru	NA	Shona	SN
Nepali	NE	Sindhi	SD
Norwegian	NO	Singhalese	SI
Occitan	OC	Siswati	SS
Oriya Pashto;	OR	Slovak	SK
Pushto	PS	Slovenian	SL
PERSIAN (Farsi)	FA	Somali	SO
Polish	PL	Spanish	ES
Portuguese	PT	Sudanese	SU
Punjabi	PA	Swahili	SW

(续表)

国　家	ISO 代码	国　家	ISO 代码
Quechua	QU	Swedish	SV
Rhaeto-Romance	RM	Tagalog	TL
Tajik	TG	Ukrainian	UK
Tamil	TA	Urdu	UR
Tatar	TT	Uzbek	UZ
Telugu	TE	Vietnamese	VI
Thai	TH	Volapuk	VO
Tibetan	BO	Welsh	CY
Tigrinya	TI	Wolof	WO
Tonga	TO	Xhosa	XH
Tsonga	TS	Yiddish	YI
Turkish	TR	Yoruba	YO
Turkmen	TK	Zhuang	ZA
Twi	TW	Zulu	ZU
Uigur	UG		

附录 H

MIME 媒体类型

本书中的大量元素使用了 type 属性，该属性的值是一个 MIME 媒体类型。

最初设计 MIME(Multipurpose Internet Mail Extension，多用途 Internet 邮件扩展)媒体类型的目的是使 e-mail 能够包含除纯文本之外的其他信息。MIME 媒体类型指示以下内容：

- 如何将一个消息的多个部分(例如文本和附件)组合到消息中。
- 指定消息的每一部分的方式。
- 用于传送的项编码方式，从而即使软件被设计为仅操作 ASCII 文本也能够处理该消息。

但是，MIME 类型不只是用于 e-mail；它们也被 Web 服务器用于告诉 Web 浏览器将发送给它们何种类型的材料，以便它们能够正确处理该类型的文件。

MIME 内容的类型由两部分组成：

- 主类型
- 子类型

主类型与子类型之间由整斜杠隔开——例如，用于 HTML 的 text/html。

本附录以主类型进行组织：

- text
- image
- multipart
- audio
- video
- message
- model
- application

例如，text 主类型包含纯文本文件的类型，如下所示：

- text/plain，用于纯文本文件
- text/html，用于 HTML 文件
- text/rtf，用于使用富文本格式的文本文件

在官方上，MIME 类型由 Internet 编号管理局(Internet Assigned Numbers Authority，IANA)分配和列举。

在这个列表中，许多流行的 MIME 类型(均以“x-”开头)并不是由 IANA 分配，并且

没有获得官方认可(需要提醒的是，其中一些 MIME 类型非常流行，并且浏览器支持它们，例如 audio/x-mp3。可以在 www.iana.org/assignments/media-types/处找到官方的 MIME 类型列表)。

以.vnd 开头的 MIME 类型是供应商特有的类型。

在本附录中，以粗体字型列举一些最流行的 MIME 类型，从而方便找到它们。

text

注意，当指定一个内容类型字段(例如在<meta>元素中)的 MIME 类型时，也可以同时指示该文本使用的字符集。例如：

```
content-type:text/plain; charset=iso-8859-1
```

如果不指定字符集，则默认字符集是 US-ASCII。

表 H-1

calendar	parityfec	richtext
css	**plain**	**rtf**
directory	prs.fallenstein.rst	**sgml**
enriched	prs.lines.tag	t140
html	rfc822-headers	tab-separated-values
uri-list	vnd.in3d.spot	vnd.sun.j2me.app-descriptor
vnd.abc	vnd.IPTC.NewsML	vnd.wap.si
vnd.curl	vnd.IPTC.NITF	vnd.wap.sl
vnd.DMClientScript	vnd.latex-z	vnd.wap.wml
vnd.fly	vnd.motorola.reflex	vnd.wap.wmlscript
vnd.fmi.flexstor	vnd.ms-mediapackage	**xml**
vnd.in3d.3dml	vnd.net2phone.commcenter.command	**xml-external-parsed-entity**

image

表 H-2

bmp	tiff-fx	vnd.mix
cgm	vnd.cns.inf2	vnd.ms-modi
g3fax	vnd.djvu	vnd.net-fpx
gif	vnd.dwg	vnd.sealed.png
ief	vnd.dxf	vnd.sealedmedia.softseal.gif

(续表)

jpeg	vnd.fastbidsheet	vnd.sealedmedia.softseal.jpg
naplps	vnd.fpx	vnd.svf
png	vnd.fst	vnd.wap.wbmp
prs.btif	vnd.fujixerox.edmics-mmr	vnd.xiff
prs.pti	vnd.fujixerox.edmics-rlc	x-portable-pixmap
t38	vnd.globalgraphics.pgb	x-xbitmap
tiff	vnd.microsoft.icon	

multipart

表 H-3

alternative	form-data	report
appledouble	header-set	signed
byteranges	mixed	voice-message
digest	parallel	
encrypted	related	

sound

表 H-4

32kadpcm	G726-24	MP4A-LATM
AMR	G726-32	mpa-robust
AMR-WB	G726-40	mpeg
basic	G728	mpeg4-generic
CN	G729	parityfec
DAT12	G729D	PCMA
dsr-es201108	G729E	PCMU
DVI4	GSM	prs.sid
EVRC	GSM-EFR	QCELP
EVRC0	L8	RED
EVRC-QCP	L16	SMV
G722	L20	SMV0

(续表)

G.722.1	L24	SMV-QCP
G723	LPC	telephone-event
G726-16	MPA	tone
VDVI	vnd.nokia.mobile-xmf	vnd.sealedmedia.softseal.mpeg
vnd.3gpp.iufp	vnd.nortel.vbk	vnd.vmx.cvsd
vnd.cisco.nse	vnd.nuera.ecelp4800	**x-aiff**
vnd.cns.anp1	vnd.nuera.ecelp7470	**x-midi**
vnd.cns.inf1	vnd.nuera.ecelp9600	**x-mod**
vnd.digital-winds	vnd.octel.sbc	**x-mp3**
vnd.everad.plj	vnd.qcelp — 逐渐淘汰，改为使用 audio/qcelp	**x-wav**
vnd.lucent.voice	vnd.rhetorex.32kadpcm	

video

表 H-5

BMPEG	MP4V-ES	vnd.mpegurl
BT656	MPV	vnd.nokia.interleaved-multimedia
CelB	**mpeg**	vnd.objectvideo
DV	mpeg4-generic	vnd.sealed.mpeg1
H261	nv	vnd.sealed.mpeg4
H263	parityfec	vnd.sealed.swf
H263-1998	pointer	vnd.sealedmedia.softseal.mov
H263-2000	quicktime	vnd.vivo
JPEG	SMPTE292M	x-sgi-movie
MP1S	vnd.fvt	x-msvideo
MP2P	vnd.motorola.video	
MP2T	vnd.motorola.videop	

message

表 H-6

CPIM	http	s-http
delivery-status	news	sip
disposition-notification	partial	sipfrag
external-body	rfc822	

model

表 H-7

iges	vnd.gdl	vnd.parasolid.transmit.binary
mesh	vnd.gs-gdl	vnd.parasolid.transmit.text
vnd.dwf	vnd.gtw	vnd.vtu
vnd.flatland.3dml	vnd.mts	vrml

application

表 H-8

activemessage	cpl+xml	eshop
andrew-inset	cybercash	font-tdpfr
applefile	dca-rft	http
atomicmail	dec-dx	hyperstudio
batch-SMTP	dicom	iges
beep+xml	dvcs	index
cals-1840	EDI-Consent	index.cmd
cnrp+xml	EDIFACT	index.obj
commonground	EDI-X12	index.response
index.vnd	pkcs10	sgml-open-catalog
iotp	pkcs7-mime	sieve
ipp	pkcs7-signature	slate
isup	pkix-cert	timestamp-query
mac-binhex40	pkixcmp	timestamp-reply
macwriteii	pkix-crl	tve-trigger
marc	pkix-pkipath	vemmi
mathematica	**postscript**	vnd.3gpp.pic-bw-large

(续表)

mpeg4-generic	prs.alvestrand.titrax-sheet	vnd.3gpp.pic-bw-small
msword	prs.cww	vnd.3gpp.pic-bw-var
news-message-id	prs.nprend	vnd.3gpp.sms
news-transmission	prs.plucker	vnd.3M.Post-it-Notes
ocsp-request	qsig	vnd.accpac.simply.aso
ocsp-response	reginfo+xml	vnd.accpac.simply.imp
octet-stream	remote-printing	vnd.acucobol
oda	riscos	vnd.acucorp
ogg	rtf	vnd.adobe.xfdf
parityfec	sdp	vnd.aether.imp
pdf	set-payment	vnd.amiga.ami
pgp-encrypted	set-payment-initiation	vnd.anser-web-certificate-issue-initiation
pgp-keys	set-registration	vnd.anser-web-funds-transfer-initiation
pgp-signature	set-registration-initiation	vnd.audiograph
pidf+xml	sgml	vnd.blueice.multipass
vnd.bmi	vnd.ecowin.chart	vnd.fujitsu.oasysgp
vnd.businessobjects	vnd.ecowin.filerequest	vnd.fujitsu.oasysprs
vnd.canon-cpdl	vnd.ecowin.fileupdate	vnd.fujixerox.ddd
vnd.canon-lips	vnd.ecowin.series	vnd.fujixerox.docuworks
vnd.cinderella	vnd.ecowin.seriesrequest	vnd.fujixerox.docuworks.binder
vnd.claymore	vnd.ecowin.seriesupdate	vnd.fut-misnet
vnd.commerce-battelle	vnd.enliven	vnd.genomatix.tuxedo
vnd.commonspace	vnd.epson.esf	vnd.grafeq
vnd.cosmocaller	vnd.epson.msf	vnd.groove-account
vnd.contact.cmsg	vnd.epson.quickanime	vnd.groove-help
vnd.criticaltools.wbs+xml	vnd.epson.salt	vnd.groove-identity-message
vnd.ctc-posml	vnd.epson.ssf	vnd.groove-injector
vnd.cups-postscript	vnd.ericsson.quickcall	vnd.groove-tool-message
vnd.cups-raster	vnd.eudora.data	vnd.groove-tool-template
vnd.cups-raw	vnd.fdf	vnd.groove-vcard

(续表)

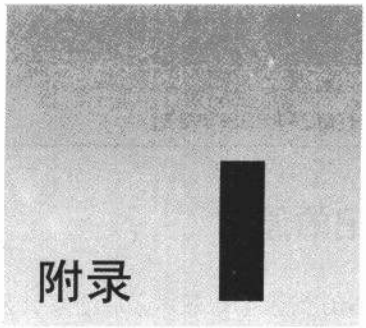
vnd.curl	vnd.ffsns	vnd.hbci
vnd.cybank	vnd.fints	vnd.hhe.lesson-player
vnd.data-vision.rdz	vnd.FloGraphIt	vnd.hp-HPGL
vnd.dna	vnd.framemaker	vnd.hp-hpid
vnd.dpgraph	vnd.fsc.weblaunch	vnd.hp-hps
vnd.dreamfactory	vnd.fujitsu.oasys	vnd.hp-PCL
vnd.dxr	vnd.fujitsu.oasys2	vnd.hp-PCLXL
vnd.ecdis-update	vnd.fujitsu.oasys3	vnd.httphone
vnd.hzn-3d-crossword	vnd.japannet-setstore-wakeup	vnd.lotus-notes
vnd.ibm.afplinedata	vnd.japannet-verification	vnd.lotus-organizer
vnd.ibm.electronic-media	vnd.japannet-verification-wakeup	vnd.lotus-screencam
vnd.ibm.MiniPay	vnd.jisp	vnd.lotus-wordpro
vnd.ibm.modcap	vnd.kde.karbon	vnd.mcd
vnd.ibm.rights-management	vnd.kde.kchart	vnd.mediastation.cdkey
vnd.ibm.secure-container	vnd.kde.kformula	vnd.meridian-slingshot
vnd.informix-visionary	vnd.kde.kivio	vnd.micrografx.flo
vnd.intercon.formnet	vnd.kde.kontour	vnd.micrografx.igx
vnd.intertrust.digibox	vnd.kde.kpresenter	vnd.mif
vnd.intertrust.nncp	vnd.kde.kspread	vnd.minisoft-hp3000-save
vnd.intu.qbo	vnd.kde.kword	vnd.mitsubishi.misty-guard .trustweb
vnd.intu.qfx	vnd.kenameaapp	vnd.Mobius.DAF
vnd.ipunplugged.rcprofile	vnd.kidspiration	vnd.Mobius.DIS
vnd.irepository.package+xml	vnd.koan	vnd.Mobius.MBK
vnd.is-xpr	vnd.liberty-request+xml	vnd.Mobius.MQY
vnd.japannet-directory-service	vnd.llamagraphics.life-balance.desktop	vnd.Mobius.MSL
vnd.japannet-jpnstore-wakeup	vnd.llamagraphics.life-balance.exchange+ xml	vnd.Mobius.PLC
vnd.japannet-payment-wakeup	vnd.lotus-1-2-3	vnd.Mobius.TXF
vnd.japannet-registration	vnd.lotus-approach	vnd.mophun.application
vnd.japannet-registration-	vnd.lotus-freelance	vnd.mophun.certificate
vnd.motorola.flexsuite	vnd.netfpx	vnd.pvi.ptid1
vnd.motorola.flexsuite.adsi	vnd.noblenet-directory	vnd.pwg-multiplexed [RFC3391]

(续表)

vnd.motorola.flexsuite.fis	vnd.noblenet-sealer	vnd.pwg-xhtml-print+xml
vnd.motorola.flexsuite.gotap	vnd.noblenet-web	vnd.Quark.QuarkXPress
vnd.motorola.flexsuite.kmr	vnd.novadigm.EDM	vnd.rapid
vnd.motorola.flexsuite.ttc	vnd.novadigm.EDX	vnd.s3sms
vnd.motorola.flexsuite.wem	vnd.novadigm.EXT	vnd.sealed.doc
vnd.mozilla.xul+xml	vnd.obn	vnd.sealed.eml
vnd.ms-artgalry	vnd.osa.netdeploy	vnd.sealed.mht
vnd.ms-asf	vnd.palm	vnd.sealed.net
vnd.mseq	vnd.paos.xml	vnd.sealed.ppt
vnd.ms-excel	vnd.pg.format	vnd.sealed.xls
vnd.msign	vnd.pg.osasli	vnd.sealedmedia.softseal.html
vnd.ms-lrm	vnd.picsel	vnd.sealedmedia.softseal.pdf
vnd.ms-powerpoint	vnd.powerbuilder6	vnd.seemail
vnd.ms-project	vnd.powerbuilder6-s	vnd.shana.informed.formdata
vnd.ms-tnef	vnd.powerbuilder7	vnd.shana.informed .formtemplate
vnd.ms-works	vnd.powerbuilder7-s	vnd.shana.informed .interchange
vnd.ms-wpl	vnd.powerbuilder75	vnd.shana.informed.package
vnd.musician	vnd.powerbuilder75-s	vnd.smaf
vnd.music-niff	vnd.previewsystems.box	vnd.sss-cod
vnd.nervana	vnd.publishare-delta-tree	vnd.sss-dtf
vnd.sss-ntf	vnd.vectorworks	vnd.yamaha.smaf-audio
vnd.street-stream	vnd.vidsoft.vidconference	vnd.yamaha.smaf-phrase
vnd.svd	vnd.visio	vnd.yellowriver-custom-menu
vnd.swiftview-ics	vnd.visionary	watcherinfo+xml
vnd.triscape.mxs	vnd.vividence.scriptfile	whoispp-query
vnd.trueapp	vnd.vsf	whoispp-response
vnd.truedoc	vnd.wap.sic	wita
vnd.ufdl	vnd.wap.slc	wordperfect5.1

(续表)

vnd.uiq.theme	vnd.wap.wbxml	x400-bp
vnd.uplanet.alert	vnd.wap.wmlc	**x-debian-package**
vnd.uplanet.alert-wbxml	vnd.wap.wmlscriptc	**x-gzip**
vnd.uplanet.bearer-choice	vnd.webturbo	**x-java**
vnd.uplanet.bearer-choice-wbxml	vnd.wqd	**x-javascript**
vnd.uplanet.cacheop	vnd.wrq-hp3000-labelled	**x-msaccess**
vnd.uplanet.cacheop-wbxml	vnd.wt.stf	**x-msexcel**
vnd.uplanet.channel	vnd.wv.csp+xml	**x-mspowerpoint**
vnd.uplanet.channel-wbxml	vnd.wv.csp+wbxml	**x-rpm**
vnd.uplanet.list	vnd.wv.ssp+xml	**x-zip**
vnd.uplanet.list-wbxml	vnd.xara	**xhtml+xml**
vnd.uplanet.listcmd	vnd.xfdl	**xml**
vnd.uplanet.listcmd-wbxml	vnd.yamaha.hv-dic	**xml-dtd**
vnd.uplanet.signal	vnd.yamaha.hv-script	**xml-external-parsed-entit y**
vnd.vcx	vnd.yamaha.hv-voice	**zip**



逐渐淘汰的和浏览器专用的标记

随着 HTML 和 XHTML 版本的逐渐开发，大量标记已经被逐渐淘汰，这意味着 XHTML 规范已经删除了这些标记，或者在将来的版本将删除它们。虽然仍然可以在 Transitional XHTML 中使用大多数这样的标记，但是 Strict XHTML 1.0 已经删除了本附录中列举的大量元素、属性和样式。

本书中包含逐渐淘汰的或过期的标记，因为事实上很可能会在其他人的代码中遇到它们，并且偶尔可能需要依靠其中的一些标记来完成特殊的工作(例如，使代码工作于非常老的浏览器中，如 IE3 浏览器和 Netscape 3 浏览器)，并且某些浏览器不支持希望利用 CSS 完成的功能。

除了逐渐淘汰的标记之外，本附录也会介绍一些浏览器专用标记，很可能已经遇到过这样的标记。这些是 Microsoft 或 Netscape 添加到它们的浏览器中的标记(有时两个公司添加相同的标记)，以便允许用户实现更多的功能——但是这些元素和属性从来都不是 HTML 规范的一部分，因此称为浏览器专用标记。

因此，本附录不仅帮助您处理遇到的逐渐淘汰的标记，而且可以使用一些技术来获得所需的结果。本附录将介绍以下内容：

- 在 HTML 和 XHTML 的最近版本中已经逐渐淘汰的元素和属性
- 在不使用 CSS 的情况下指定字体外观
- 在不使用 CSS 的情况下控制背景
- 在不使用 CSS 的情况下控制链接、列表和表的表示
- 一些用于控制文档格式的元素和属性
- Microsoft 公司添加到 IE 浏览器中的一些元素、属性和样式(但它们不被其他类型的浏览器支持)
- Netscape 公司添加到 Navigator 浏览器中的一些元素、属性和样式(但它们不被其他类型的浏览器所支持)

但是，在查看这些标记之前，有必要简短地说明本附录花费大量篇章介绍逐渐淘汰的标记的原因。

存在逐渐淘汰的标记的原因

第 1 章中介绍过，在 HTML 到达 4.01 版本之后创建了 XHTML1.0。在 HTML 的每一个版本中，添加了一些新元素和属性，并删除了一些陈旧的元素和属性。这些改变是必须的，不仅是因为 Web 页面的作者希望创建更复杂的页面，而且是因为访问 Internet 的浏览器(也称为客户端)类型已经改变。

虽然您习惯于在桌面计算机上浏览 Web，但是许多新设备也可用于访问 Web，并且这些新设备是描述文档内容(题头、段落等)的标记与表示规则相分离的部分原因。

在引入 CSS 之前，较老版本的 HTML 中包含了能够用于控制 Web 页面外观的标记(例如能够控制文档中使用的字体的元素，或者能够设置页面背景颜色的 bgcolor 属性)。样式与内容的分离和 CSS 的引入是逐渐淘汰标记出现的最重要原因。

遗憾的是，创建于 CSS 和一些较新标记出现之前的较老浏览器无法理解执行操作的较新方法，如果必须为 IE3 浏览器或者 Netscape 3 浏览器创建 Web 站点，则将不得不考虑是使用 CSS(这两种浏览器基本上完全无法理解 CSS)还是使用这些逐渐淘汰的元素和属性。事实上，为类似于这样的浏览器创建标记的可能性非常小；但是，可能需要理解如何以较陈旧的元素和属性编写页面，然后使其工作。

较老的页面不遵循许多规则

应当注意的是，Web 上的很多页面很可能不遵循本书到目前为止介绍的许多规则。您将会看到元素名和属性名以大写和小写形式书写，属性值遗漏引号，甚至属性没有值，并且将会看到元素没有结束标签。页面没有 DOCTYPE 声明，并且页面中到处都是逐渐淘汰的标记。但是请记住，许多不遵循规则的页面是在没有严格要求规则时编写的，并且这些代码在编写时能够被很好地接受。

注意：

即使是具有存在问题的或逐渐淘汰的标记的页面也能够在浏览器中正常显示，但是尽可能地避免这些不良的习惯仍是明智的举措；否则将无法通过很多未来的浏览器查看这些页面。

不只是人类编写的代码会在将来逐渐淘汰。一些早期版本的 Web 页面制作工具，例如 Microsoft FrontPage 和 Macromedia Dreamweaver，有时生成的代码也具有奇怪的大小写或者遗漏一些引号，或者某些属性不带任何值。这种情况显然不会使这些软件沿着正确方向前进。这些软件的第一个版本开发于 XHTML 和其较严格的规则出现之前(第 1 章中讨论过这方面的问题)。较老的浏览器极大限度地容忍不正确编写的代码，它们能够以任何方式显示页面。这也是浏览器的程序大小不断增加的主要原因——将来的浏览器将不需要具有这样的容忍度。

前面已经提到，本附录将包含一些奇怪的内容，但是如果坚持在本书中介绍的原则(不

要忘记引号、属性值和结束元素，并且不要学习其他人的不良习惯)，则创建的页面将能够被更多的人访问，您的技能将更加有市场。

字体

在本部分中将介绍几个元素(和它们的属性)，这些元素将影响文本和字体的外观，但是它们都已经逐渐被淘汰。

元素

元素在 HTML 3.2 中引入，并在 HTML 4.0 中成为逐渐淘汰的元素，但是直到现在仍然被广泛使用。该元素可用于指示起始标签和结束标签之间的字体的字型、大小和颜色。您可能已经发现很多站点中包含<font>标签，页面中每次文本样式改变时都具有一个<font>标签。

表 I-1 中给出了<font>元素依赖的 3 个属性。

表 I-1

属 性 名	用 途	值
face	指定应该使用的字型	使用的字型的名称(可以包含多个名称，具体顺序根据个人喜好而定)
size	指定字体的大小	1 到 7 之间的数值，其中 1 是最小的字体大小，7 是最大的字体大小
color	指定字体的颜色	一种颜色名或者十六进制值(详情请查看附录 D)

下面的示例给出了<font>元素的用法(ai_eg01.html)。在该示例中存在 3 个<font>元素：

```
<html>
  <head>
    <title>Example of &lt;font&gt; Element</title>
  </head>
  <body>
    <p>This is the browser's default font.</p>
    <font face="arial, verdana, sans-serif" size="2">
      <h1>Example of the &lt;font&gt; Element</h1>
      <p><font size="4" color="darkgray">Here is some size 3 writing
      in the color called darkgray. The typeface is determined by the
      previous &lt;font&gt; element that contains this paragraph.</font></p>
      <p><font face="courier" size="2" color="#000000">Now here is a courier
              font, size 2, in back</font></p>
    </font>
  </body>
</html>
```

这个示例的结果如图 I-1 所示。

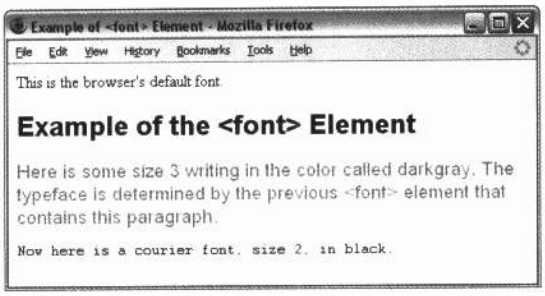


图 I-1

从图 I-1 中可以看出，<font>元素中的所有文本遵循起始标签<font>的属性中的规则。第一段采用浏览器的默认字体(很可能是黑色 3 号 Times 字体族)。第一个<font>元素直接位于该段落的后面并包含页面的剩余部分。第一个<font>元素是该页面中除第一个段落之外的所有内容的默认设置。该元素直到结束标签</body>之前的位置才结束，因此文档的剩余部分应当采用 Arial 字型。

Arial 字型的名称后面跟着 Verdana 字型；如果无法使用 Arial 字型，则 Verdana 字型是第二选择。然后，如果 Verdana 字型也无法使用，则采用浏览器的默认 sans-serif 字体：

```
<font face="arial, verdana, sans-serif" size="2">
```

该<font>元素还指明文档剩余部分文本的默认字体大小是 2。注意，这个<font>元素不会重写<h1>元素的大小，但是它影响使用的字型——题头的字型将是 Arial。

虽然这个<font>元素是页面大部分内容的默认设置，但是如果希望页面的特定部分具有其他字体性质，则可以在另一个<font>元素中指示。

第二个段落的字体颜色被改为黑灰色，字体大小被改为 4。

```
<p><font size="4" color="darkgray">Here is some size 4 darkgray
    writing</font></p>
```

然后第三个段落使用不同的字型、更小的字体大小以及黑色：

```
<p><font face="courier" size="2" color="#000000">Now here is a courier
    font, size 2, in back</font></p>
```

注意，可能必须在<td>和<th>元素中使用<font>元素，因为在表外部指定的样式不会被单元格内的文本继承。图 I-2 给出了从 1 到 7 的不同字体大小(ai_eg02.html)。

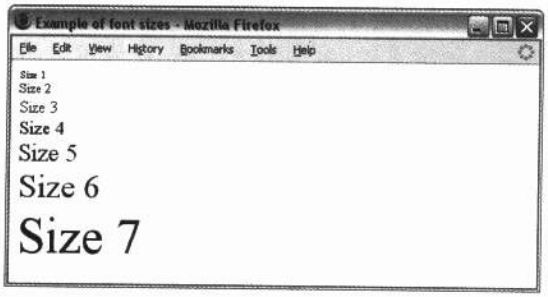


图 I-2

注意：

不同浏览器显示的字体大小可能稍有不同，因此不能依赖它们在布局中精确地指定相同像素数量的高或宽。

利用 CSS 的首选方法是在包含希望赋予样式的文本的元素中使用 font-family、font-size 和 color 性质。第 7 章中介绍了这些 CSS 性质。

text 属性

text 属性在<body>元素中用于指示文档中文本的默认颜色；在 HTML 4 中该属性被标记为逐渐淘汰，它的值应该是一个颜色名或者十六进制颜色值。例如(ai_eg03.html)：

```
<body text="#999999">
  This text should be in a different color than the next bit
    <font color="#000000">which is black</font>, and now back to gray.
</body>
```

结果如图 I-3 所示。

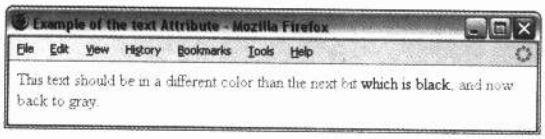


图 I-3

<basefont>元素

<basefont>元素用于为文档中没有包含在<font>元素中的文本设置默认字体大小、颜色和字型。然后可以使用<font>元素重写<basefont>元素的设置。但是，Firefox 浏览器或 Netscape 6 浏览器不支持该元素，并且其他浏览器并不总是遵守表或题头中的设置。

<basefont>元素能够附带的属性与<font>元素完全相同。再次强调，类似于题头元素的元素将保持它们自己的大小。

也可以将字体的大小设置为相对于<basefont>元素指定的大小，方式是赋予它们一个数值，例如+1 表示比<basefont>元素指定的字体大 1 号，-2 表示比<basefont>元素指定的字体小 2 号(尺寸标准从 1 到 7)。

重新访问前一个示例并执行一些改动，以此来产看这些效果——突出显示的部分是执行的改动(ai_eg04.html)：

```
<html>
  <head>
    <title>Example of &lt;basefont&gt; Element</title>
  </head>
  <body>
    <basefont face="arial, verdana, sans-serif" size="2" color="#ff0000">
    <p>This is the page's default font.</p>
      <h2>Example of the &lt;basefont&gt; Element</h2>
      <p><font size="+4" color="darkgray">Here is some darkgray text
          four sizes larger</font></p>
      <p><font face="courier" size="-1" color="#000000">Here is a courier
          font, a size smaller, in black</font></p>
  </body>
</html>
```

图 I-4 给出了该示例在 Internet Explorer 浏览器中的结果(因为该示例无法工作于 Firefox 浏览器中)。

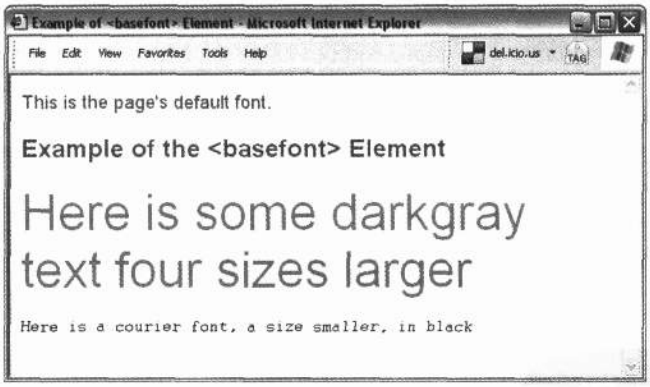


图 I-4

现在默认字体取自于<basefont>元素中指定的性质；该字体为红色，大小为 2 号，并且使用 Arial 字型。

<h2>元素之后的段落使用的字体尺寸比默认字体大 4 号，并且是灰色文本，而下面的段落使用的字体尺寸比默认字体小 1 号——并且该字体的颜色是黑色(重写默认设置)。

因为这个元素在 HTML 4 中被标记为逐渐淘汰，所以首选方法是对<body>元素使用 CSS 样式，为文档设置默认字体性质。

<s>元素和<strike>元素

<s>元素和<strike>元素在 HTML 3.2 中引入，但在 HTML 4 中被标记为逐渐淘汰。它们指示其中内容应该具有一种删除线样式。例如(ai_eg05.html)：

```
<s>This text will have a line through it</s>
<strike>This text will also have a line through it.</strike>
```

结果如图 I-5 所示。

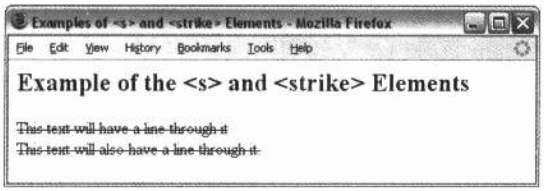


图 I-5

现在应当使用 CSS 中的 text-decoration 性质，并将该性质的值设置为 strikethrough，除非是尝试指示删除的内容，此时应当使用<del>元素。

<u>元素

<u>元素中的内容之下具有一条下划线。该元素在 HTML 3.2 中引入，但在 HTML 4 中被标记为逐渐淘汰。

```
<u>This text should be underlined.</u>
```

结果如图 I-6 所示。

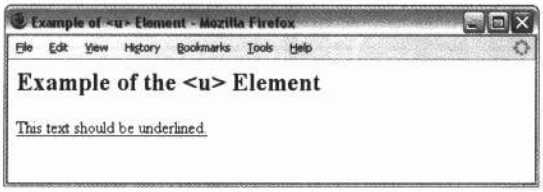


图 I-6

现在应当使用 CSS 中的 text-decoration 性质，并且将该特性的值设置为 underline，除非是尝试指示添加的内容(当修订文档时)，此时应当使用<ins>元素。

<listing>、<plaintext>和<xmp>元素

这 3 个元素都已经被淘汰；它们在 HTML 2 中引入，并在 HTML 4 中删除。在这里包括它们是因为可能会在以前的示例中遇到它们。

所有 3 个元素都以等宽字体显示文本，与<pre>元素相同。

<xmp>元素用于显示小段的示例代码，它不能包含其他任何标记；类似于元素名中的

尖括号的任何字符都像文本一样显示，因此不需要为它们使用转义字符。建议设计人员在任何一行中最多使用 80 个字符。

<listing>元素的推荐限制是每行中最多使用 132 个字符，并且趋向于以小字体显示文本。

<plaintext>标签指明它后面出现的任何内容都应该以纯文本显示，即使是标记也是如此。因为<plaintext>元素后面的任何内容都显示为普通文本，包括标签，因此没有结束标签(如果在代码中使用了</plaintext>标签，则它也将显示为普通文本)。

下面是关于这 3 个元素的示例(ai_eg07.html)：

```
<body>
  <h2>Example of the &lt;listing&gt;, &lt;plaintext&gt;, and &lt;xmp&gt;
     Elements</h2>
  <listing>These words are written inside a &lt;listing&gt; element.</listing>
  <xmp>These words are written inside an <xmp> element.</xmp>
  <plaintext>These words are written inside a <plaintext>
     element.
</body>
```

图 I-7 中给出了该示例的结果。注意，<xmp>元素中的转义字符被忽略，并且没有执行转义(也可以包含尖括号，并且尖括号将正常显示)。</body>标签和</html>标签也会显示出来，因为起始标签<plaintext>之后的任何内容都被视为纯文本：

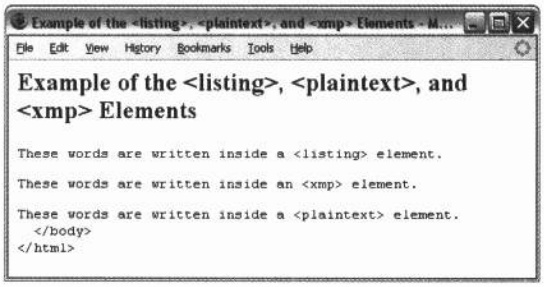


图 I-7

XHTML 中的替代元素是<pre>元素和<samp>元素。

背景

在 HTML 中存在以下两个属性，它们可用于改变整个页面或者部分页面的背景：

- bgcolor，可用于为<body>元素和各种表元素指定一种背景颜色
- background，可用于为<body>元素指定一种背景图像

bgcolor 属性

bgcolor 属性用于为整个文档或者部分文档指定一种背景颜色。它可以用于以下元素中：

```
<body> <table> <tr> <th> <td>
```

该属性的值应当是一种颜色名或者十六进制颜色，附录 D 中包含了相关描述。下面的示例文档中使用了一些不同的背景颜色(ai_eg08.html)：

```
<html>
  <head>
    <title>Example of bgcolor Attribute</title>
  </head>
  <body bgcolor="#efefef">
     <h2>Example of the bgcolor Attribute</h2>
     <table bgcolor="#999999">
       <tr>
         <th bgcolor="#cccccc">Heading One</th>
         <th bgcolor="#cccccc">Heading Two</th>
       </tr>
       <tr bgcolor="#f2f2f2">
         <td>Cell One</td>
         <td>Cell Two</td>
       </tr>
       <tr>
         <td>Cell Three</td>
         <td>Cell Four</td>
       </tr>
  </body>
</html>
```

这个页面的显示效果如图 I-8 所示。

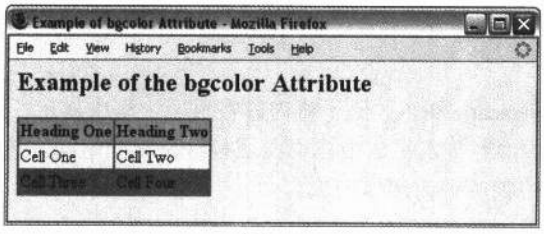


图 I-8

该页面具有一种非常亮的灰色背景颜色，该背景颜色在<body>元素中指定。然后表具有一种背景颜色，如表的最下面行和表的所有边所示，这是表的默认颜色。然后可以看到 bgcolor 属性用于<th>元素(表的题头)和紧跟其后的<tr>元素(表的第一行)。

现在改变背景颜色的首选方法是使用 CSS 中的 background-color 性质。

background 属性

background 属性可用于为整个页面指定背景图像，它的值应该是指向背景图像的 URL(该 URL 可以是绝对 URL 或相对 URL)。Netscape 和 Microsoft 也允许将这个属性用于表，从而为表创建背景图像。

下面的示例中使用了 background 属性(ai_eg09.html):

```
<html>
  <head>
    <title>Example of background Attribute</title>
  </head>
  <body background="images/background_large.gif" bgcolor="#f2f2f2">
    <h2>Example of the background Attribute</h2>
  </body>
</html>
```

注意，该示例的<body>元素中也使用了 bgcolor 属性，如果无法找到背景图像，则将使用该属性。该示例的结果如图 I-9 所示。

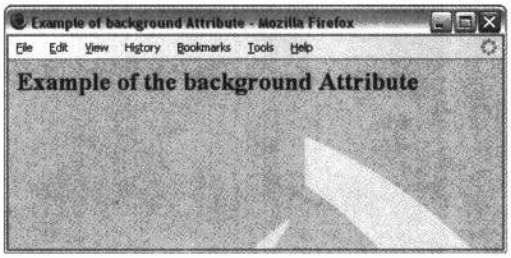


图 I-9

格式化

下面的一些属性和元素用于格式化和定位页面中的元素以及它们的内容。

元素

元素由 Netscape 引入，用于将内容在页面中居中显示。起始标签 和结束标签 之间的内容将水平居中在页面或包含元素的中间。该元素在 HTML 3.2 中引入，并在 HTML 4 中被标记为逐渐淘汰。

下面的示例介绍了 元素的用法。该示例也包含一个表，用于显示 元素中处理表的方式(ai_eg10.html):

```
<body>
  <h2>Example of the &lt;center&gt; Element</h2>
  <center>
    Anything inside a &lt;center&gt; element is centered on the page, or within
    its containing element.<br /><br />
    <table width="600" border="1">
      <tr>
        <td>Cells whose content is written inside a &lt;center&gt; will be
            centered within the cell, like the one to the right.</td>
        <td><center>This cell's content should be centered.</center></td>
      </tr>
      <tr>
```

```
        <td><center>This cell's content should be centered.</center></td>
        <td>Cells whose content is written inside a &lt;center&gt; will
           be centered within the cell, like the one to the left.</td>
      </tr>
    </table>
  </center>
</body>
```

在这个示例中(如图 I-10 所示)， 标签(位于<h1>元素之后)将页面剩余部分的内容居中显示。值得注意的是，它居中显示页面中的任何文本和表本身，但不会居中显示单元格中的文本，除非在<td>元素中包含 元素(这个示例中的表被赋予边框，方式是使用 border 属性指示各条边所在的位置)。

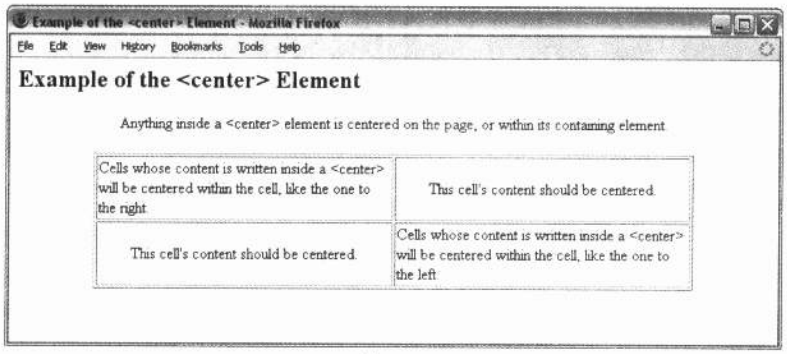


图 I-10

对齐文本内容的首选方法是使用一些 CSS 性质，例如 text-align。

align 属性

许多元素都可以使用 align 属性来指示元素在浏览器中或它的包含元素中的位置。该属性在 HTML 4.01 中被标记为逐渐淘汰。

align 属性的可能值如表 I-2 所示。注意，值 justify 只适用于文本，并且浏览器对 top、middle 和 bottom 值的支持度不如 left、right 和 center 值。

表 I-2

值	目 的
left	将元素与页面或包含元素的左边对齐
right	将元素与页面或包含元素的右边对齐
center	将元素在页面或包含元素中居中显示
justify	调整单词在页面或包含元素中的位置，从而使文本的左边和右边都接触容器
top	将元素与页面或包含元素的顶部对齐
middle	将元素在浏览器窗口或者包含元素中的中间垂直对齐
bottom	将元素与浏览器窗口或包含元素的底部对齐

下面的元素可以附带 align 属性：

```
<caption> <applet> <iframe> <img> <input> <object> <legend> <table> <hr>
<div> <h1> <h2> <h3> <h4> <h5> <h6> <p>
```

下面的代码包含了一些示例，这些示例演示了 align 属性的用法(ai_eg11.html)。

```
<body>
  <h2 align="center">Example of the align Attribute</h2>

    <table width="600" align="center" border="1">
      <tr>
        <td align="left">This cell's content should be left-aligned.</td>
        <td align="right">This cell's content should be right-aligned.</td>
      </tr>
      <tr>
        <td align="center">This cell's content should be centered.</td>
        <td width="300" align="justify">This cell's content should be
        justified, but it needs to spread across more than one line to
        show it working.</td>
      </tr>
    </table>
</body>
```

这里的<h1>元素和<table>元素被居中显示，然后表中的每一个单元格使用不同的对齐方式。

为了使文本两端对齐，文本需要换行(这也是这个示例中的<td>元素附带 width 属性的原因)。两端对齐段落的最后一行不需要像其他行一样伸展到浏览器或其包含元素的左边框或者右边框。

图 I-11 给出了这个示例的外观。

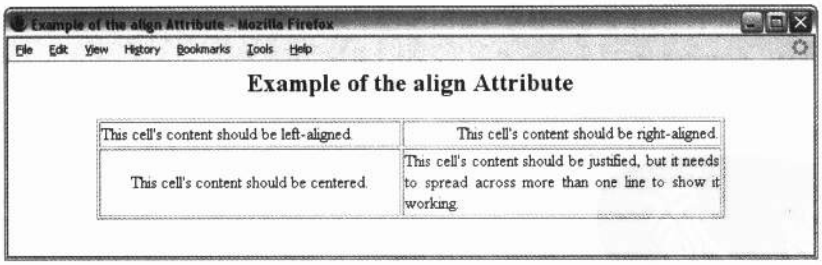


图 I-11

在 CSS 中对齐内容的首选方法是使用 text-align 性质和 vertical-align 性质，并且利用 float 性质定位。

width 属性

width 属性用于设置元素的宽度，单位是像素。它可以用于下面的元素中：

```
<td> <th> <table> <hr> <applet>
```

直到现在 width 属性还经常使用，特别是利用表进行布局的页面。依赖表定位内容的站点需要固定页面的宽度，以便正确显示页面并使页面对读者有意义，并且虽然较少的访问者将使用版本 3 或更老的浏览器，但是如果表不使用固定宽度，这些少数访问者将无法正确查看站点，因为他们的浏览器不支持 CSS 的 width 性质。

下面示例的表中使用了 width 属性和一个<hr />元素(ai_eg12.html)：

```
<body>
  <h2>Example of the width Attribute</h2>

     <table width="600" border="1">
       <tr>
         <td width="200">This cell should be 200 pixels wide.</td>
         <td width="400">This cell should be 400 pixels wide.</td>
       </tr>
       <tr>
         <td width="200">This cell should be 200 pixels wide.</td>
         <td width="400">This cell should be 400 pixels wide.</td>
       </tr>
     </table>
   <br /><br />
  <hr width="300" />
</body>
```

图 I-12 给出了这个示例在浏览器中的外观。

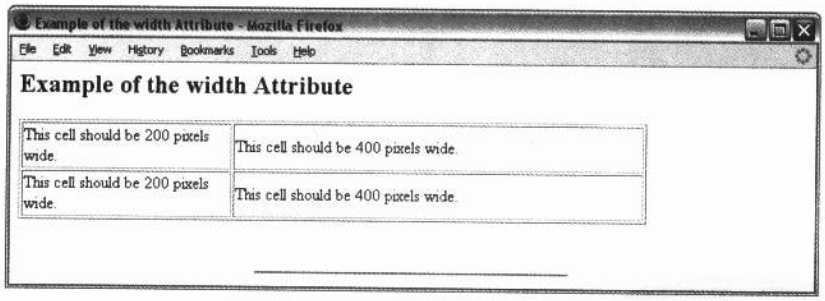


图 I-12

为这些元素设置宽度的首选方法是使用 CSS 中的 width 性质。

height 属性

height 属性用于设置元素的高度，单位是像素。属性用于<th>、<td>和 <applet>元素中。下面示例中的<td>元素使用了 height 属性(ai_eg13.html)：

```
<body>
  <h2>Example of the height Attribute</h2>
```

```
        <table width="600" border="1">
          <tr>
            <td width="300" height="300">This cell should be 300 pixels
high.</td>
            <td width="300" height="300">This cell should be 300 pixels
high.</td>
          </tr>
        </table>

    </body>
```

该示例的显示效果如图 I-13 所示，其中表单元格都是正方形。

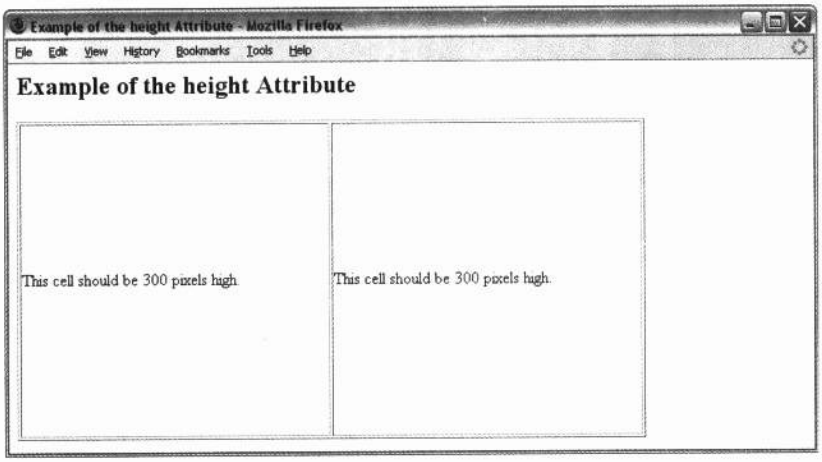


图 I-13

为这些元素设置高度的首选方法是使用 CSS 中的 height 性质。

vspace 属性

vspace 属性指定出现在 HTML 元素的上方或下方的空白或内边距量，它的值的单位是像素。

下面的示例演示了<img>元素中的 vspace 属性如何确保图像上方和下方具有 20 个像素的空白，从而将其与文本隔开(ai_eg14.html)：

```
<body>
    <h2>Example of the vspace Attribute</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod  tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam,  quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo  consequat. Duis aute irure dolor in reprehenderit
in voluptate velit esse  cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat  non proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.
```

```
        <img src="images/logo_small.gif" alt="wrox logo" vspace="20"
border="1" />
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
    quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
    non proident, sunt in culpa qui officia deserunt mollit anim id est
    laborum.</p>

    </body>
```

该示例的结果如图 I-14 所示。

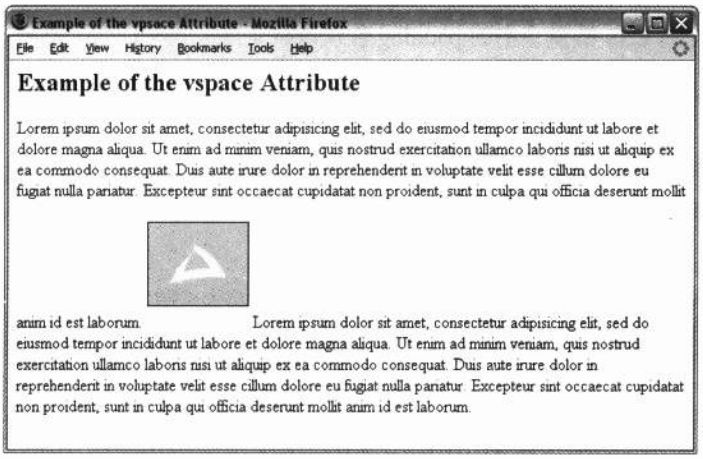


图 I-14

该属性已经被 CSS 框模型中的 padding 性质取代。

hspace 属性

hspace 属性是 vspace 属性在水平方向的等价属性，它确保元素的左边或右边存在内边距或者空白。

在下面的示例中，使用 hspace 属性在图像的左边和右边创建了 40 个像素的内边距 (ai_eg15.html)：

```
    <body>
        <h2>Example of the vspace Attribute</h2>

        <p><img src="images/logo_small.gif" alt="wrox logo" hspace="40"
border="1" />
        There should be 40 pixels between the image and the edge of the window,
and another 40 pixels between the edge of the image and this text.</p>

    </body>
```

该示例的结果如图 I-15 所示。

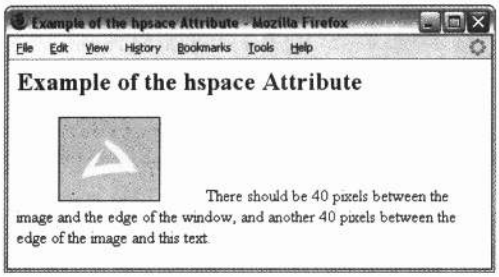


图 I-15

该属性已经被 CSS 框模型中的 padding 性质取代。

clear 属性(位于
元素中)

clear 属性用于换行元素
中，以指示浏览器显示
元素之后的文本行的方式。clear 属性可以附带值 left、right、all 和 none，如下示例很好地解释了它的用法(ai_eg16.html)：

```
    <body>
        <img src="images/logo_small.gif" alt="wrox logo" align="left"
border="1" />
        The text after this image will be displayed next to the image and wrap
        to the next line until you see the line break element.<br clear="left">
Now it should be on a new line underneath (not next to) the image.
    </body>
```

如果 clear 属性用于
元素中，则它后面的文本或元素将在指示为 clear 属性值的边框被清除之后才会显示。在这个示例中，因为
元素具有 clear 属性，并且该属性的值为 left，所以
元素之后的文本直到它的左边(包含元素或框内)不存在任何内容时才会显示。在这个示例中，文本直到显示图像之后才继续显示，其中图像位于该文本的左边。

该示例的结果如图 I-16 所示——注意，一些文本在图像的下方继续显示。如果没有 clear 属性，则该文本将简单地显示在下一行中。

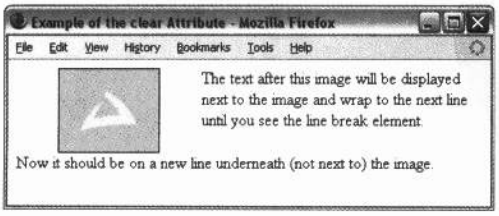


图 I-16

如果设置了值 all，则文本或元素的左边和右边必须不存在任何内容。

CSS 具有自己的 clear 性质，以取代这个属性。

链接

您可能已经在某些 Web 站点上注意到，当访问某个页面或单击链接时，链接的颜色会改变。表 I-3 中列举的 3 个属性可用于改变链接的颜色：alink、link 和 vlink，应当在<body>元素中指定它们。

表 I-3

属　性	用　途	值
alink	指定活跃链接或者选中链接的颜色	十六进制代码或者颜色名
link	指定文档中所有链接的默认颜色	十六进制代码或者颜色名
vlink	指定已访问链接的颜色	十六进制代码或者颜色名

下面的示例演示了这些属性如何影响链接的颜色(ai_eg17.html)：

```
<body alink="0033ff" link="#0000ff" vlink="#333399">
   <h2>Example of the Link Attribute</h2>
   <p>This example contains some links, which you should play with to see
how they behave:</p>
   <ul>
     <li>The <a href="http://www.wrox.com/">Wrox Web site</a> tells you
about existing and forthcoming Wrox books.</li>
     <li>The <a href="http://www.w3.org/">W3C Web site</a> is the home of
the XHTML and CSS recommendations.</li>
     <li>The <a href="http://www.google.com/">Google Web site</a> is a
popular search engine.</li>
   </ul>

</body>
```

在这个示例中，用户已经访问的链接和没有访问的链接具有不同的蓝色阴影。这有助于用户浏览站点，因为他们能够确定已经访问过哪些链接(这有助于他们再次找到一个页面，方式是跟踪一些已经访问过的链接)，而且有助于他们避免访问同一个页面两次。

通常，已经访问的和没有访问的链接的颜色非常相似，如图 I-17 中的链接所示(很难区分其中哪些链接被访问过)。为了更好地理解这个示例的工作方式，请读者自己实现它(可以利用下载代码和本附录剩余的代码实现它)。

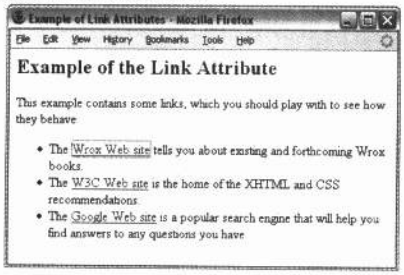


图 I-17

列表

与列表相关的多个元素和属性已经被逐渐淘汰或者不再允许使用，从帮助排序和赋予列表样式的属性到其他用于创建列表可视化效果的元素。

start 属性

start 属性用于有序列表的<ol>元素，以指示列表编号的起始数字。当然，默认值是 1。例如(ai_eg18.html)：

```
<body>
  <ol start="4">
    <li>This list should start at four</li>
    <li>Therefore this item should be five</li>
    <li>And this item should be six</li>
  </ol>
</body>
```

该示例的结果如图 I-18 所示。

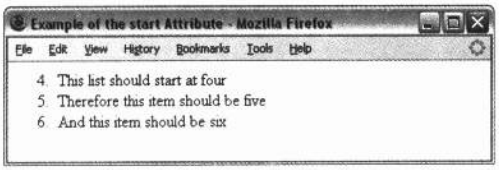


图 I-18

该属性已经被 CSS 的计数器和 counter-reset 性质取代。但是，CSS 计数器还未被浏览器很好地支持，因此如果希望列表的起始编号不是 1，并且使页面能够被更多浏览器支持，最好使用这个属性。

value 属性

value 属性用于<li>元素，以指明编号列表中该项的编号。因此，利用它可以创建编号不按顺序排列的编号列表。如下面的示例所示：

```
<body>
  <ol>
    <li value="3">one</li>
    <li value="7">two</li>
    <li value="1">three</li>
    <li value="9">four</li>
    <li value="4">five</li>
  </ol>
</body>
```

结果如图 I-19 所示，其中列表的编号不按顺序排列。

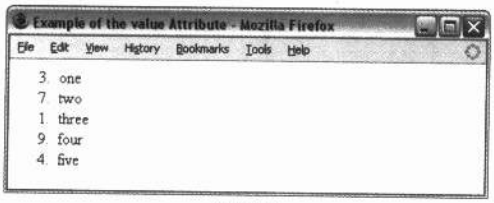


图 I-19

type 属性

type 属性控制列表的项目符号或编号(也称为标记符)的类型。该属性可以用于<li>、<ol>或<ul>元素中。

表 I-4 给出了项目符号种编号系统的不同类型标记符。

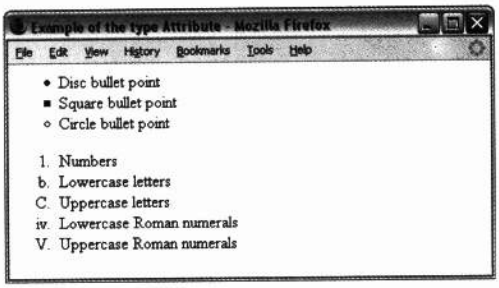
表 I-4

值	描 述
disc	实心圆
square	实心正方形
circle	空心圆
1	数值 1、2、3、4
a	小写字母 a、b、c、d
A	大写字母 A、B、C、D
i	小写罗马数字 i、ii、iii、iv
I	大写罗马数字 I、II、III、IV

无序列表的默认值是实心圆，有序列表的默认值是阿拉伯数字，例如 1、2、3 等。下面示例中的 type 属性分别使用了这些值(ai_eg20.html)：

```
<body>
  <ul>
    <li type="disc">Disc bullet point</li>
    <li type="square">Square bullet point</li>
    <li type="circle">Circle bullet point</li>
  </ul>
  <ol>
    <li type="1">Numbers</li>
    <li type="a">Lowercase letters</li>
    <li type="A">Uppercase letters</li>
    <li type="i">Lowercase Roman numerals</li>
    <li type="I">Uppercase Roman numerals</li>
  </ol>
</body>
```

该示例的显示效果如图 I-20 所示。



I-20

<dir>元素和<menu>元素

<dir>元素和<menu>元素在 HTML 2.0 规范中引入，用于创建无序项目列表和嵌套列表。它们相互之间以及与<ul>元素之间几乎相同(ai_eg21.html)。

```
<dir>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <dir>
    <li>Item 4.1</li>
    <li>Item 4.2</li>
    <li>Item 4.3</li>
    <li>Item 4.4</li>
  </dir>
</dir>

<menu>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <menu>
    <li>Item 4.1</li>
    <li>Item 4.2</li>
    <li>Item 4.3</li>
    <li>Item 4.4</li>
  </menu>
</menu>
```

这些元素的结果如图 I-21 所示。

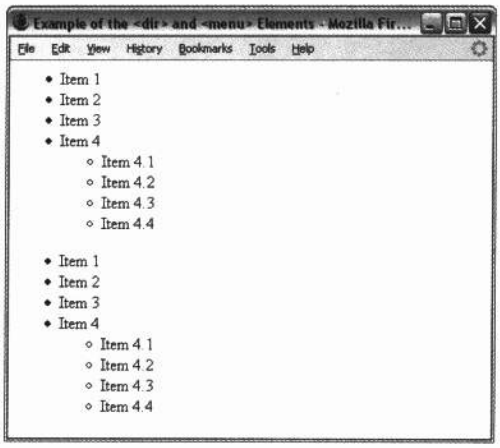


图 I-21

<dir>元素最初用于列举目录中的文件，而<menu>元素被设计用于由链接组成的菜单，因此它的内容在一些浏览器中的显示比<ul>元素和<dir>元素的内容紧凑。应当使用<ul>元素取代这两个已经逐渐淘汰的元素。

表

存在很多以前允许用于<table>元素的逐渐淘汰的属性——尤其是 align 属性和 bgcolor 属性(本附录前面已经介绍过它们)以及 nowrap 属性，下面将介绍 nowrap 属性。

nowrap 属性

nowrap 属性可用于<td>元素和<th>元素中，它阻止文本在表单元格中换行。例如：

```
<table width="200">
  <tr>
    <td nowrap>This text should not wrap even though the table is only
        supposed to be 200 pixels wide.</td>
  </tr>
</table>
```

该示例的结果如图 I-22 所示，尽管该表仅有 200 像素宽，但实际上它能够延伸到所在行的末尾——因为文本不换行。

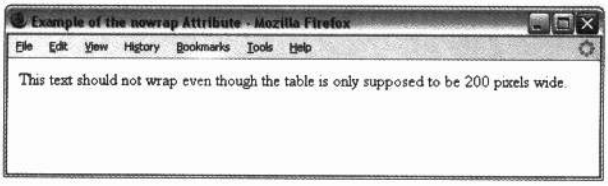


图 I-22

CSS 中的 white-space 性质取代了 nowrap 属性，该性质的值为 nowrap。

其他各种属性

本节介绍一些其他元素和属性，它们都已经逐渐淘汰，但是无法归纳在前面的章节中。

border 属性

border 属性指定元素周围的边框厚度，单位为像素。例如，下面示例中的<img>元素具有 border 属性(ai_eg23.html)：

```
<body>
   <img src="images/logo_small.gif" border="4" alt="wrox logo" />
</body>
```

该示例的结果如图 I-23 所示。

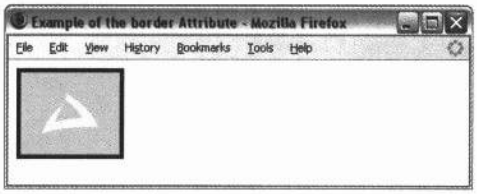


图 I-23

这个属性已经被 CSS 中的 border-width 性质取代，但是它仍然经常使用——特别是当使用图像作为链接时，因为 IE 默认情况下会在作为链接的图像周围添加一个像素宽的蓝色边框。

compact 属性

compact 属性指示浏览器以比普通情况更小的行间距显示文本，它不采用任何值(但是如果用于 Transitional XHTML 1.0，则需要值 compact)。

该属性的默认值是 false，它已经被 CSS 的 letter-spacing 性质和 word-spacing 性质取代。

language 属性

language 属性指示用于 HTML 元素(通常是<script>元素)的脚本语言类型。例如：

```
<script language="JavaScript">
```

language 属性已经被 type 属性取代，它的值是一个 MIME 类型(例如 type="text/JavaScript")。

version 属性

version 属性指定编写文档所遵循的 HTML DTD 版本。

该属性已经被删除，因为它的信息与 DOCTYPE 声明提供的信息重复。

<applet>元素

<applet>元素用于将 Java 小应用程序嵌入到 HTML 页面中。该元素和它的属性在 HTML 3.2 中引入，但已经被<object>元素取代，第 3 章中已经讨论了<object>元素。

表 I-5 给出了<applet>元素能够附带的属性。

表 I-5

属　性	用　途
code	Java 小应用程序编译代码的文件名。code 指定的小应用程序文件的路径相对于该小应用程序的 codebase(而不是 URL 或绝对路径)
codebase	指定 Java 小应用程序代码所在的目录。如果没有指定 codebase 属性，则该小应用程序文件被假设位于与 HTML 文件相同的目录中
object	指定 Java 小应用程序编译代码的文件名，该文件存储小应用程序状态的串行化表示。该文件的路径必须仅相对于该小应用程序的 codebase(而不是 URL 或绝对路径)
name	指定该元素的名称，以便脚本能够与它通信(该属性仅在用于<applet>元素时才是逐渐淘汰的属性)
archive	由空格分隔的一组 URL，这些 URL 具有多个 Java 类或其他将加载到浏览器中以帮助改进小应用程序性能的资源(该属性仅在用于<applet>元素时才是逐渐淘汰的属性)
width	小应用程序的宽度，单位为像素
height	小应用程序的高度，单位为像素

<embed>元素

<embed>元素用于在 HTML 中引入<object>元素之前，使用它包含需要特定插件应用程序的文件。例如，使用它在页面中包含 Flash 动画。

使用 src 属性标记将要包含的对象——如同包含图像一样。可以使用 type 属性指明所包含内容的类型，该属性的值是表示该资源的 MIME 类型(或者不指定，由浏览器自己决定)。

<embed>元素能够附带的属性如表 I-6 所示。

表 I-6

属　性	用　途
align	指定页面或其包含元素内的对象的对齐方式
border	指定该对象的边框宽度，单位为像素

(续表)

属　性	用　途
height	指定该对象的高度，单位为像素
hidden	隐藏对象，使用户无法看到它(将其设置为 0 像素宽、0 像素高)，该属性对于音频来说非常有用
hspace	指定对象的左边和右边预留的水平空间量，单位为像素
name	与其他元素中的 name 属性一样，该属性用于标记元素
palette	在 IE 浏览器中，该属性的值是由竖线隔开的一对十六进制颜色值。第一个颜色值是前景颜色，第二个颜色值是背景颜色。在 Netscape 浏览器中，palette 属性既可以是前景也可以是背景，用于指明插件应当使用哪一种窗口系统颜色调色板
pluginspage	用于指定 Web 页面的 URL，从该 Web 页面中能够下载使用该文件所需的插件(该属性仅被 Netscape 浏览器支持)
src	希望嵌入的对象的 URL
type	指明将包含在页面中的对象的 MIME 类型(它确定用于查看该对象的插件)
units	可将指明嵌入对象的高度和宽度的度量单位从默认的像素改为相对 en 单位(一个 en 单位是文本磅值的一半宽度)
vpsace	指定对象的顶部和底部应当预留的垂直空间量，单位为像素
width	指定该对象的宽度，单位为像素

<embed>元素还可以附带一些专门用于查看它们所需的插件的属性。对于这些属性，需要查阅与特定插件相关的文档。由于这样的属性太多，此处不列举它们。

注意：

如果使用 Macromedia Flash 在页面中包含图形，您将发现一些公开发行的工具(它们提供一些 HTML 代码，可用于在页面中包含 Flash 动画)不仅使用<object>元素在页面中包含动画，而且提供<embed>元素来包含动画，以便较老的或不支持<object>元素的浏览器能够正常显示页面。

<isindex>元素

<isindex>元素在 HTML 2.0 中引入，用于创建单行文本字段，而不需要使用<form>元素(将使用 HTTP get 方法发送用户的输入)。当用户按下 Enter(或 Return)键时，将提交表单并且使用+字符替代空格(然后服务器中的程序或页面将响应或处理发送的数据)。

当显示时，文本框的上方和下方将具有一条水平线。

虽然可以使用多个<isindex>标签，但仅将最后一个该标签的内容发送给服务器。该元素也可以附带 prompt 属性，用于向用户提供一条提示，告诉他们应当在文本框中输入什么内容。例如，下面是一个用于创建搜索框的<isindex>元素(ai_eg24.html)：

```
<body>
    <isindex prompt="search">
</body>
```

该示例的结果如图 I-24 所示，注意其中的水平线。

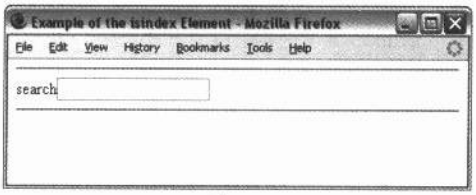


图 I-24

<nobr>元素

Firefox、Netscape 和 IE 浏览器都支持 XHTML 规范中阻止换行的扩展元素：<nobr>元素(该元素保留其包含元素的普通样式，并且不会导致文本以等宽字体显示)。如果选择使用<nobr>元素，则它可以包含一个称为<wbr>的子元素，以指明在<nobr>元素内的什么位置可以换行，但是该元素也是扩展元素。

IE 浏览器专用的元素

表 I-7 列举了 5 个 IE 支持的元素，这些元素不是 HTML 规范的组成部分。通常应当避免使用这些元素，除非是为不同的浏览器提供不同的页面，或者知道所有的访问者都将使用 IE 浏览器。

表 I-7

元　素	IE 版本	目　的
<bgsound>	2	在后台中播放声音文件(已被<object>元素取代)
<marquee>	2	以一种滚动模式显示文本
<ruby>	5	提供发音支持
<rt>	5	提供发音支持
<xml>	5	创建一个 XML 数据岛，将 XML 记录集嵌入到页面中

IE 浏览器专用的属性

表 I-8 列举了 IE 浏览器专用的属性

表 I-8

属　性	目　的
atomicselection	指定分组元素和它的内容是否必须作为整体选择
balance	平衡左边和右边扬声器的声音(用于<bgsound>元素)
behavior	指定<marquee>元素内容的滚动方式
bgproperties	为页面设置固定的背景图像；也称为水印
bordercolordark	当表中的单元格具有 3D 边框时，指定使用较暗的颜色。用于<table>元素
bordercolorlight	当表中的单元格具有 3D 边框时，指定使用较亮的颜色。用于<table>元素
bottommargin	为页面指定底部页边空白，单位为像素。用于<body>元素
contenteditable	确定分组元素的内容是否能够被用户编辑
dataformatas	设置或获取分组元素所包含的数据是否应该显示为文本或者 HTML
datalfd	用于当浏览器连接到服务器端数据库时的数据绑定(关于这方面的更多信息，请查看关于 ASP 的参考手册)
datasrc	用于当浏览器连接到服务器端数据库时的数据绑定(关于这方面的更多信息，请查看关于 ASP 的参考手册)
datapagesize	用于当浏览器连接到服务器端数据库时的数据绑定(关于这方面的更多信息，请查看关于 ASP 的参考手册)
direction	指示<marquee>元素内文本的滚动方向
dynsrc	用于将电影嵌入到客户端缓存中
framespacing	指定框架集中框架之间的空间量，单位为像素
hidefocus	当元素获得焦点时,用于阻止在它周围显示可见的线
leftmargin	指定页面左边的页边空白，单位为像素。用于<body>元素中
rightmargin	指定页面右边的页边空白，单位为像素。用于<body>元素中
loop	指定<marquee>元素的内容滚动的次数
lowsrc	用于指定<img>元素中图像的一个低分辨率版本，以便在加载时首先显示该版本
scrolldelay	指定每次绘制<marquee>元素时的延迟时间，单位为毫秒(默认值是每隔 60 毫秒重新绘制一次)
topmargin	指定页面顶部的页边空白，单位为像素。用于<body>元素中
truespeed	一个布尔属性，指明是否应该使用 scrolldelay 值。默认值是 false；如果值为 true，则<marquee>元素将使用 scrollamount 属性和 scrolldelay 属性中指示的值(忽略小于 60 毫秒的任何值)
unselectable	指示元素不能选择
volume	指示音量，<bgsound>元素的内容将以该音量播放，它的值从－10000 到 0(默认值是 0，代表最大音量)

注意，当使用表控制页面的整体布局，并且页面出现在 IE 浏览器的左上角且它的边不

具备白色边框时，页面有时使用<body>元素中的 topmargin 属性和 leftmargin 属性来指明边周围不存在页边空白。

```
<body topmargin="0" leftmargin="0" border="0">
```

在 Transitional XHTML 中可以放心地使用 topmargin 属性和 leftmargin 属性，因为 Netscape 和其他浏览器将简单地忽略它们无法理解的属性，但是这些属性将无效，因为它们不是标记的一部分(第 13 章中讨论了验证问题)。

IE 浏览器专用的 CSS 样式

表 I-9 列举了只有 IE 浏览器才支持的一些 CSS 样式(以及引入它们时的浏览器版本)。

表 I-9

性　质	IE 版本	目　的
behavior	5	确定<marquee>元素中文本的滚动方式
ime-mode	5	当用于输入方法指示器时，可允许输入中文、日文和韩文字符
layout-grid	5	其他几个 layout-grid 性质的简写形式(即可以同时设置其他几个 layout-grid 性质)
layout-grid-char	5	指定用于显示文本的字符网格的大小(类似于 line-height 性质)
layout-grid-charspacing	5	指定字符之间的间距(效果类似于 line-height)
layout-grid-line	5	指定用于绘制文本的网格线值(类似于 line-height)
layout-grid-mode	5	指定网格使用一个还是两个尺寸单位
layout-grid-type	5	指定当绘制元素内容时使用的页面布局网格的类型(如果有的话)
line-break	5	指定日文文本的换行规则
ruby-align	5	指定<rt>元素中文本的水平对齐方式
ruby-overhang	5	当<rt>元素中的文本宽于 non-ruby 的内容时，指定文本是否将悬浮 non-ruby 内容的边之上
ruby-position	5	指定<rt>元素中指定的文本的位置(上方或者内联)
text-autospace	5	控制文本调整时自动添加的空间宽度；通常用于亚洲语言中使用的象形文字
text-justify	5	两端对齐元素中的文本
text-kashida-space	5.5	当两端对齐元素中的文本时，用于控制 kashida 扩展与空白扩展之间的比例。kashida 是一种印刷术上的效果，该效果通过延长特定点中的特定字符来两端对齐文本行；通常用于阿拉伯语
text-underline-position	5.5	当使用 text-decoration 性质时，指定下划线应当距离文本下方多远

(续表)

性　质	IE 版本	目　的
word-break	5	控制单词之间的换行；通常用于包含多种语言的文档
word-wrap	5.5	当长单词太长而无法显示在其包含元素中时，该性质控制长单词在何处中断
writing-mode	5.5	控制对象中的内容的水平和垂直方向
zoom	5.5	指定对象的放大比例

layout-grid 性质常用于一些亚洲语言中，这些语言通常对字符使用页面布局，以便使用一个或者两个尺寸单位的网格格式化文本。

也存在几个与滚动栏外观相关的 CSS 样式。将这些性质添加到 CSS 样式表中是无害的，因为不支持这些性质的浏览器将会忽略它们。可以使用颜色名、十六进制代码或者 RGB 值指定所有颜色(与 CSS 中的所有颜色一样)。

表 I-10

性　质	IE 版本	目　的
scrollbar-3dlight-color	5.5	滚动栏中滚动框和滚动箭头的左上方边的颜色
scrollbar-arrow-color	5.5	滚动箭头上的箭头颜色
scrollbar-base-color	5.5	滚动栏主要元素的颜色，包括滚动框、滑条和滚动箭头
scrollbar-darkshadow-color	5.5	滚动栏的槽的颜色
scrollbar-face-color	5.5	滚动栏的滚动框和滚动箭头的颜色
scrollbar-highlight-color	5.5	滚动栏中滚动框和滚动箭头左上方边的颜色
scrollbar-shadow-color	5.5	滚动栏中滚动框和滚动箭头右下方边的颜色

Netscape 浏览器专用的元素和属性

表 I-11 给出了仅被 Netscape 浏览器支持的元素，其中支持这些元素的 Netscape 版本号位于“版本”列中。

表 I-11

元　素	版　本	用　途
<blink>	2、3、4、6、7	使元素的内容闪烁(CSS2 的 text-decoration 性质具有 blink 值，完成相同的功能，但当前的浏览器不支持该性质)

(续表)

元 素	版 本	用 途
<ilayer>	4	创建一个内联层，该层能够在 HTML 文档的独立部分中包含与当前查看的页面不同的页面。HTML 4 中的<iframe>元素能够创建相似的效果。该元素不同于<layer>标签，因为它采用相对定位——而不是绝对定位
<keygen>	2、3、4、6、7	用于为从 HTML 文档中提交的表单生成加密密钥(该元素驻留于表单中，创建可用加密密钥大小的选择列表，要求客户端安装证书，并使用专用的 Netscape 加密方案)。
<layer>	4	创建一个层，该层能够在 HTML 文档的独立部分中包含一个与当前查看的页面不同的页面。HTML 4 中的 元素能够创建相似的效果
<multicol>	2、3、4	允许用户定义多列格式化——例如报纸样式的列。在 XHTML 中可以使用表或 CSS 定位性质来创建相似的效果
<noembed>	2、3、4、6、7	为不支持<embed>元素的浏览器显示 HTML 文本
<nolayer>	4	为不支持<layer>元素的浏览器显示 HTML 文本
<spacer>	3、4	在 HTML 文档中显示空格

Netscape 浏览器专用的属性

表 I-12 中包含的属性仅被 Netscape 浏览器支持。

表 I-12

属 性	Netscape 版本	目 的
above	4	如果两个层重叠，指示哪一层出现在上方。用于<layer>和<ilayer>元素(例如桌面发布程序中的“返回到顶部”)
below	4	如果两个层重叠，指示哪一层出现在下方。用于<layer>和<ilayer>元素(类似于桌面发布程序中的“发送到底部”)
challenge	2	在<keygen>元素中用于指定加密密钥值将被压缩进的字符串值
clip	4	指定一个将被切除的区域(单位为像素)，使得浏览器仅显示指示的内容，区域的指定方法是使用 4 个值，这 4 个值分布表示该区域的左上角和右下角(x,y)坐标。用于<layer>和<ilayer>元素
gutter	3、4	用于<multicol>元素中用于指示每一列之间的像素数量
hidden	4	用于<embed>元素中用于指示一个对象对访问者不可见。页面中的其他项将正常地布局在它周围。通常用于将声音文件嵌入页面中，并且不希望用户看到该对象
left	4	指定文档内父元素的水平偏移量，该属性的值的单位是像素。用于<layer>和<ilayer>元素