程序员书库

OSS从入门到精通

第2版

曹方 等编著

化学工业出版社
·北京·

本书共分为四篇，循序渐进地讲述了网页基础知识和 CSS 布局的实战技术。书中从基本概念到具体实践、从对网页的认识到制作网页、从使用 CSS 进行网页局部的设计到进行整个网站的布局制作都进行了详细的阐述，并对具体知识点进行了详细的实例讲解。

本书内容丰富，讲解时注重思维方法的转变和提高实际操作能力，以最简单的方法介绍使用 CSS 进行符合 Web 标准的网页设计的相关知识和技巧。全书以 CSS 布局为主体内容，探讨 CSS 布局的入门知识与网站布局实战技巧、版式布局的细节、浏览器兼容性等，帮助读者改变传统的网站设计思维，进入基于 Web 标准的网页设计领域。本书含有大量实例，详细描述各个 CSS 属性以及代码编写技巧，方便读者模拟实践。

本书适合将要学习或者正在学习使用 CSS 布局技术的用户阅读，并可作为 Web 前端代码架构从业人员的参考手册。

图书在版编目（CIP）数据

CSS 从入门到精通 / 曹方等编著. —2 版. —北京：
化学工业出版社，2011.6
（程序员书库）
ISBN 978-7-122-11580-5
ISBN 978-7-89472-449-6（光盘）

Ⅰ. C… Ⅱ. 曹… Ⅲ. 网页制作工具，CSS
Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字（2011）第 119429 号

责任编辑：王思慧　李　萃　　装帧设计：蓝色印象
责任校对：边　涛

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号 邮政编码 100011）
印　　刷：北京永鑫印刷有限责任公司
装　　订：三河市宇新装订厂
787mm×1092mm　1/16　印张 22¾　字数 560 千字　2011 年 9 月北京第 2 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686）　售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：49.80 元（含 1DVD-ROM）

出版者的话

近年来随着我国 IT 产业的迅猛发展，越来越多的年轻人开始学习程序设计，以使得自己在未来的职业生涯中更有竞争力。但有些人刚开始学习时总觉得不得要领，究其原因主要是学习方法或者思路不得当。作为自学人员，学编程首先需要一本好书，这样才能少走弯路。

基于这种背景，2009 年我们策划出版了《程序员书库》丛书。这套图书讲解细致入微，语言通俗易懂，覆盖了常见的编程语言和开发工具，一上市就受到了读者的欢迎。

随着时间的推移，各种技术已有所更新，我们也收到了读者的大量反馈，他们对本丛书提出了很多具体的意见和建议。为了能适应最新的技术发展和读者需求，我们对本丛书做了必要的修订和改版，同时增加了一些新的品种。改版后的图书无论从技术上还是内容上，都较第 1 版图书有了很大的改进，更适合读者学习和使用。

丛书包含书目

《Java 从入门到精通》（第 2 版）
《PHP 从入门到精通》（第 2 版）
《Visual C++从入门到精通》（第 2 版）
《Visual Basic 从入门到精通》（第 2 版）
《Linux 编程从入门到精通》（第 2 版）
《CSS 从入门到精通》（第 2 版）
《Excel VBA 从入门到精通》（第 2 版）
《SQL Server 2008 从入门到精通》（第 2 版）
《C++从入门到精通》（第 2 版）
《C 语言从入门到精通》
《C#从入门到精通》

丛书主要特色

1. 多媒体视频讲解，加速学习

丛书的每本书都配有专门制作的多媒体学习光盘，方便读者学习，另外也提供了书中所涉及的源代码，以方便读者使用。

2. 由浅入深，讲解细致，轻松入门

本丛书对内容的讲解都是从最基本的配置和概念讲起，然后层层深入，最后还安排了综合案例，很适合读者学习，可以达到轻松入门、快速提高的效果。

3. 程序代码注释详尽，易于理解

丛书的每本书中都给出了典型的程序代码，并配有详尽的注释，便于读者理解，这对读者快速并深入理解编程有很大的帮助。

4．以大量实例为示范，快速掌握

丛书的每本书中都列举了大量实例，最后还提供了综合实例，非常实用，读者可以通过这些例子快速掌握所学内容，学习效果显著。

5．提供了必要的练习题和面试题

为了便于读者巩固所学的知识并有所提高，同时能对程序员面试的题目有一个基本的了解，本丛书提供了必要的练习题和面试题，以方便读者学习。

6．完善的售后服务，后顾无忧

本丛书提供技术论坛（http://www.rzchina.net）和 QQ 群（QQ 群号：21948169）答疑，读者可以在上面提问和交流。

丛书适合的读者对象

本丛书定位于没有编程基础的入门人员：

- 自学编程的入门人员；
- 各类程序设计爱好者；
- 想学习一门技术以便找工作的人员；
- 做课题设计和毕业设计的学生；
- 需要迅速掌握一门编程语言的人员；
- 大中专院校或电脑学校的学生。

学习建议

- 入门人员最好按书中安排的顺序阅读。
- 如果有一定的基础，不妨进行跳跃式阅读，有选择地阅读。
- 先弄懂书中的内容，然后可以适当拓展。
- 要勤思考、多动手，必要时一定要上机操作。
- 学会利用网络资源，经常光顾各技术论坛，不懂的地方可以利用百度来搜索解决方法。

希望这套书能成为你初学编程的良师益友，能带你轻松跨入编程的大门，并使你逐步成为编程高手。

化学工业出版社

2011 年 3 月

前　言

在当前如火如荼的 Web 2.0 新浪潮中，一切都基于 Web 标准。人们常常把 Web 2.0 描述为“作为平台的网络”。把网络看成是一个内容互动的平台之后，我们很容易发现这种变化对于网页设计的重大影响。想象一下，当来自不同领域的群体（个人、公司、政府等）存储了一系列数据之后，我们便能通过一个接口将信息混合，这一点是任何单一传统网站都不可能做到的。因此，这就使得 Web 2.0 网站必然有别于传统的网站，也优于传统网站。

Web 2.0 的网站更趋向于风格和结构的分离，随着 CSS 的流行，人们通过添加一个定义了风格信息的独立 CSS 文件实现了页面风格和结构的分离。Web 2.0 的网站更提倡使用语义标记语言编写页面代码。这些标记语言并非完全没有语义能力，设计师们用它们能够描述页面。对 Web 2.0 来说，对页面的描述不仅是一项需求，更是实现 Web 2.0 的关键。

本书正是基于 Web 标准，讲述使用标记语言 XHTML 和 CSS 层叠样式表来实现风格和结构的分离，并讲解网站布局的实例。

编者精心编写此书，目的是一步步地告诉读者如何开始新的、符合 Web 标准的 CSS 布局设计。本书的重点放在实例制作中，在制作实例的过程中，同时讲解 CSS 样式，将知识点的学习融入到实践之中，有目的地学习必要的 CSS 属性。编者结合自己多年在 Web 前端代码架构的工作实践经验，为广大喜爱或渴望从事 Web 前端代码架构的朋友介绍制作符合 Web 标准页面的各种技巧。

《CSS 从入门到精通》第 1 版出版后，受到了广大读者的普遍欢迎。许多读者反映“读过《CSS 从入门到精通》一书后，CSS 就变得不那么难学了”、“作者深入浅出地讲述了 CSS 的基本知识以及在网页中的应用实例，使得我们可以轻松地掌握 CSS，网页设计变得更加游刃有余了”……另外，由于编者能力有限，在编写过程中难免出现疏漏，热心的读者也反馈了不少宝贵意见。为了更好地帮助读者快速掌握 CSS，针对读者反馈的意见，我们对第 1 版图书进行了修订，并为每章配备了习题，供读者进行自我检查和巩固提升。希望我们的努力可以更好地帮助大家。

本书特点

本书编者是在国内综合门户网站从业的经验丰富的设计师。本书结合目前流行的网站制作实例，深入浅出地讲解了网站制作中各种常用的 XHTML 标签和 CSS 属性。在代码讲解过程中，编者还结合自己在 Web 前端代码架构的从业经验，深入讲解了定义 CSS 属性的各种技巧。在每章的最后，还对这一章讲到的知识点进行归纳总结，使读者能够理清学习思路，有重点地掌握 CSS 属性。

本书涵盖了网站制作过程中常用的 XHTML 标签和 CSS 属性，并且通过实例讲解了网站各个部分和各种布局的实现方法。在制作实例的过程中，穿插了大量实用的 CSS 属性的使用方法，便于读者在使用中完成技能的提高和升华。

本书的特点主要体现在以下几个方面。

- 本书通过常用的网站实例，全面系统地讲解了常用的 XHTML 标签和 CSS 属性的使用方法，方便读者自己进行实践和演练。所有的实例制作方法，均可以在实际制作过程中直接使用，或者根据自己的实际情况进行调整，从而方便读者在实际工作中的使用。
- 本书除讲解基础的 CSS 和 XHTML 使用方法外，还适当加入目前流行的 Web 标准理论的思想，介绍如何使用 CSS 实现结构与表现的分离，使读者在制作符合 Web 2.0 网站的时候有理可依。
- 本书结合编者多年在大型门户网站 Web 前端代码架构的经验，深入浅出地介绍 CSS 属性在实践中的应用，并在每章的最后对本章所涉及的 CSS 属性进行了归纳总结，使读者能够理清学习思路，有重点地掌握 CSS 属性。
- 本书在讲解网站制作实例的过程中，融入了编者多年工作的经验，对很多问题提出了自己独到的见解。这些见解都是在实践的基础上总结出来的，能够帮助和启发读者拓展思路，同时也能指导读者在学习过程中对具体问题进行具体分析。
- 本书中列举了世界上各国的优秀网页，并且突出介绍这些网页的各自特点，并讲解了怎样借鉴它们各自的优点，改进我们制作网页的方法。
- 本书的编排采用循序渐进的方式，适合初级、中级学习者逐步掌握运用 XHTML 和 CSS 制作 Web 标准网站的基本方法。
- 本书的所有例子、源代码和各种免费工具都附在随书光盘中，以方便读者使用。

本书内容

全书分为四篇，共 18 章，基础知识全面，实例实用，讲解透彻，主要内容如下。

第一篇（第 1 章～第 4 章）CSS 布局的基础知识篇。

介绍了 CSS 布局的基础知识，包括网页的基本概念、CSS 的基础知识、XHTML 与 JavaScript 基础以及浏览器的兼容与解析问题，为渴望学习标准网页布局的朋友提供一些必要的基础知识，同时也为后面使用 CSS 进行布局实例制作奠定基础。第 4 章的浏览器的兼容与解析问题，系统归纳了解决浏览器兼容问题的方法，能够帮助有一定页面代码架构基础的朋友提高代码架构的水平。

第二篇（第 5 章～第 13 章）CSS 布局的实例制作篇。

介绍了使用 CSS 进行网页布局的实例，包括使用 CSS 制作背景、使用 CSS 布局页面顶部内容、使用 CSS 制作网站导航、使用 CSS 制作列表、使用 CSS 制作表单、使用 CSS 制作内容的版式、使用 CSS 制作链接样式、使用 CSS 制作数据表格以及使用 CSS 制作页面底部内容，从网页的不同组成部分，以实例的方式讲解使用 CSS 布局制作的方法。第 7 章列举了很多种类的导航制作方法，里面涉及了大量的 CSS 属性和使用方法，是学习 CSS 应用的很好的途径。

第三篇（第 14 章～第 16 章）CSS 布局的整体布局篇。

介绍了 CSS 布局的整体布局方法，包括 CSS 的 9 种基本布局以及 CSS 整体复杂布局，这对在掌握了 CSS 基础之后渴望进一步得到提高的读者来说，具有较大的参考价值。

第四篇（第 17 章～第 18 章）CSS 布局的综合应用篇。

介绍了使用 CSS 布局进行网站制作的综合应用，包括博客类网站的制作实例以及企业类网站的制作实例，综合运用 CSS 和 XHTML 的布局知识，制作一个网站的布局，并且可以举一反三地制作其他类型的网站。

本书由浅入深，由理论到实践，尤其适合初学者逐步学习和完善自己的知识结构。

本书内容丰富，注重思维方法与实践应用，适合初、中级网页设计爱好者和希望使用 Web 标准进行网页布局设计的专业网页设计师，是任何网站开发相关人员手中不可缺少的资料。

本书读者

- 希望进入 Web 前端代码架构行业的新手。
- 迫切希望提高个人 Web 前端代码架构的初级代码设计人员。
- 具备一定的测试理论知识但是缺乏实践的前端代码设计人员。
- 希望深入了解 Web 标准和 Web 2.0 网站制作的从业人员。

本书编者

本书主要由曹方编写，其他参与编写和资料整理的人员有刘成、马臣云、潘娜、阮履学、陶则熙、王大强、王磊、徐琦、许少峰、颜盟盟、杨娟、杨瑞萍、于海波、俞菲、曾苗苗、赵莹、朱存等。

由于时间和水平有限，书中难免存在不足之处，欢迎读者批评指正。

编者

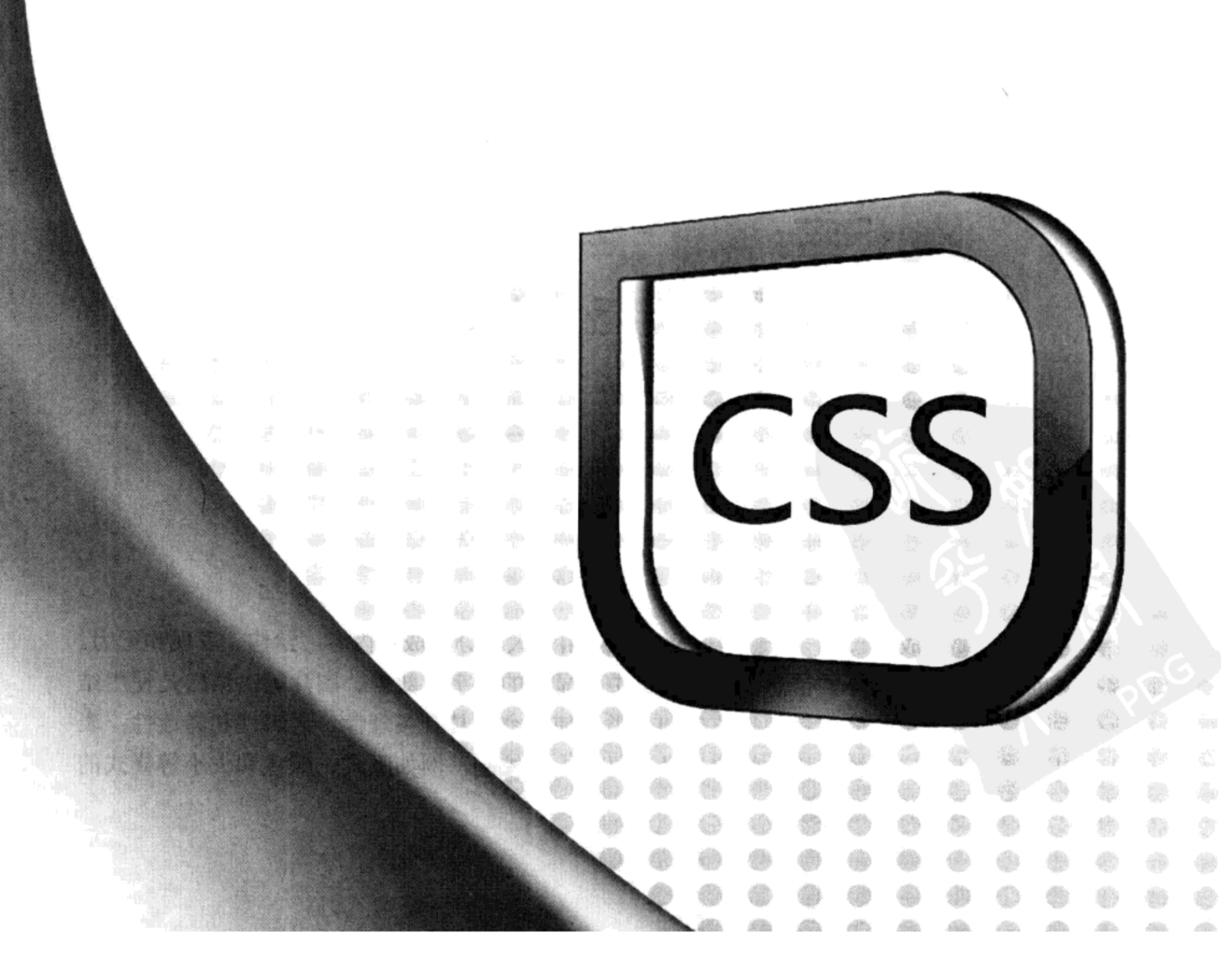
2011 年 3 月

目 录

第一篇 基础知识篇

第三篇 整体布局篇

第一篇　基础知识篇



第1章 与网页有关的基本概念

网页（Web Page）听起来是一个很难懂的概念，极其抽象，看得见摸不着。现代信息化的社会，无论是通过家庭个人电脑，还是公共场所的网吧，都可以在互联网上冲浪。互联网上看到的就是网页，它是由若干种代码编写的文件形式，上面有图片、音乐、视频等丰富的资源。在开始学习 CSS 布局之前，首先来了解一下网页的组成部分以及各部分的作用。

1.1 网页的基本构成

网页就是在电脑的浏览器中呈现的一个个页面。如果把一个网站比作一本书，那么网页就是这本书中的一页。如图 1-1 所示就是用浏览器打开的一个普通的网页。

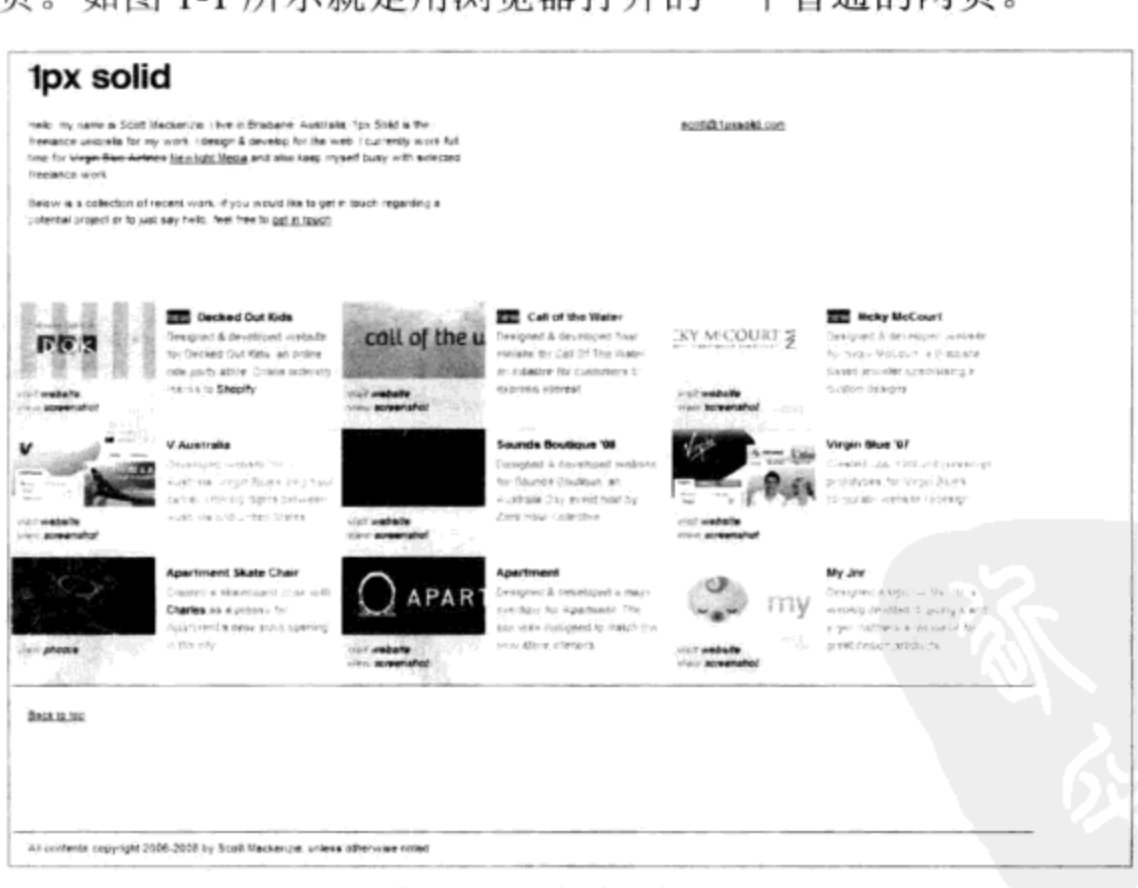


图 1-1 一个普通的网页

从网页的主要构成说起，一个标准的网页一般由 4 大部分组成：内容、结构、表现和行为。内容是网页中要传达的纯粹的信息，例如网页中所显示的文字、数据、图片等；结构是使用结构化的方法对网页中用到的信息进行整理和分类，使内容更具有条理性、逻辑性和易读性；表现是使用表现技术对已经被结构化的信息进行显示上的控制，例如版式、颜色和大小等样式的控制；行为就是网页的交互操作。

把如图 1-1 所示网页的 4 个基本组成部分分离，就得到 4 个层，如图 1-2 所示。

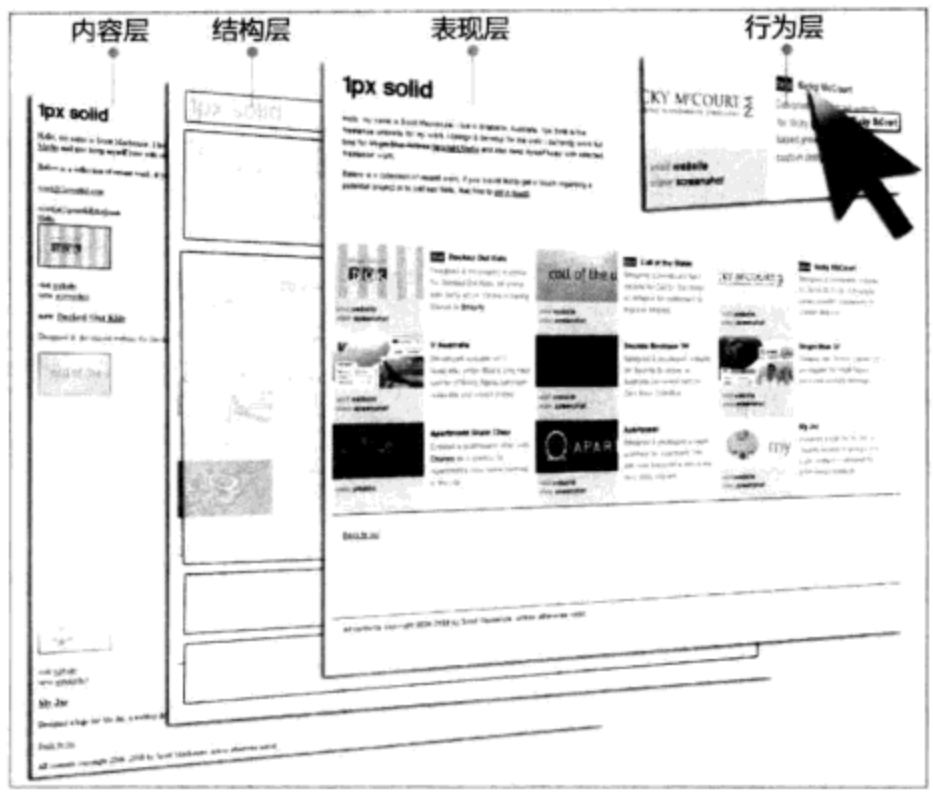


图 1-2　网页 4 个组成部分示意图

总之，网页的 4 个组成部分，可以形象地比喻为：内容是人；结构是用来标明头、身体、四肢等各个部位；表现则是给各部位加上服装、装饰以使人更加漂亮；行为是四肢在特定环境下所呈现的交互行为。

1.2　网页的结构设计——HTML 与 XHTML

用于网页的结构化设计语言有 HTML、XHTML 和 XML 等。使用这样的语言代码，可以将网页的文字、图片或数据等信息进行分类、排版，最终呈现给浏览者。本节主要介绍 HTML、XHTML 的基本概念及它们的区别。

1.2.1　认识 HTML

HTML（Hyper Text Mark-up Language）为超文本标记语言，是网页的一个重要构成，主要负责将网页内容进行格式化，使内容更具逻辑性。

HTML 是网页的基本描述语言，由 Tim Berners-Lee 在 1990 年提出。起初设计 HTML 语言的目的是为了能把存放在一台电脑中的文本或图形与另一台电脑中的文本或图形方便地联系在一起，形成有机的整体，人们不用考虑具体信息是在当前电脑上还是在网络的其他电脑上。只要在某一文档中单击一个图标，Internet 就会马上转到与此图标相关的内容上去，而这些信息可能存放在网络的另一台电脑中。

HTML 文本是由 HTML 命令组成的描述性文本，HTML 命令可以说明文字、图形、动画、声音、表格、链接等。HTML 的结构包括头部（Head）和主体（Body）两大部分，头部描述浏

览器所需的信息；主体包含所要说明的具体内容。

HTML 是一种为普通文件中某些字句加上标记（Tag）的语言，其目的在于运用标记使文件达到预期的显示效果，例如下面的 HTML 代码。

```
<html>
    <head>
        <title>HTML 网页</title>
    </head>
    <body>
        这是 HTML 网页<b>这里的文字是粗体</b>
    </body>
</html>
```

HTML 文件看上去和一般文本类似，但是它比一般文本多了标记，例如<html>、<b>等，通过这些标记，可以告诉浏览器如何显示这个文件。

HTML 不是 C++和 Java 之类的程序语言，它只是标示语言，基本上只要明白了各种标记的用法便学懂了 HTML。HTML 的格式非常简单，只是由文字及标记组合而成。在编辑方面，任何文字编辑器都可以，例如记事本。当然使用专业的网页编辑软件更好，例如 Adobe Dreamweaver、frontpage 等。

1.2.2 认识 XHTML

讲到 XHTML，首先应该了解什么是 XML，因为 XHTML 是基于 XML 的一种标准化结构语言。下面来了解一下 XML 和 XHTML 的区别。

1. XML

XML 是 The Extensible Markup Language（可扩展标记语言）的简写。目前推荐遵循的是 W3C（World Wide Web Consortium）于 2000 年 10 月 6 日发布的 XML 1.0。和 HTML 一样，XML 同样来源于 SGML（一种早在 Web 发明之前就已经存在的定义电子文档结构和描述其内容的国际标准语言，是所有电子文档标记语言的起源），但 XML 是一种能定义其他语言的语言。XML 最初设计的目的是弥补 HTML 的不足，以强大的扩展性满足网络信息发布的需要，后来逐渐用于网络数据的转换和描述。

2. XHTML

XHTML 是 The Extensible Hyper Text Markup Language（可扩展超文本标记语言）的简写，是 HTML 的“升级规范”产品，其中“X”代表可扩展的，是单词“Extensible”的缩写。实际上，XHTML 也属于 HTML 家族，对比以前各个版本的 HTML，它具有更严格的书写标准、更好的跨平台能力。

简单地说，XHTML 就是更加严谨的 HTML。打个比方，如果说 HTML 是汉语，那么 XHTML 就是普通话。

2000 年底，国际 W3C 组织发布了 XHTML 1.0 版本。XHTML 1.0 是一种在 HTML 4.0 基础

上优化和改进的新语言。XML 虽然数据转换能力强大，完全可以替代 HTML，但面对成千上万已有的站点，直接采用 XML 还为时过早。因此，在 HTML 4.0 的基础上，用 XML 的规则对其进行扩展，得到了 XHTML。简单地说，建立 XHTML 的目的就是实现 HTML 向 XML 的过渡。

1.2.3　制作一个简单的 XHTML 网页

现在尝试制作一个简单的使用 XHTML 编写的网页，这里可以先不用知道“<>”及里面的字母所代表的含义。

（1）在 Windows 系统中，启动记事本。

（2）在记事本中编写以下代码。

```
<html>
    <head>
        <title>1_1</title>
    </head>
    <body>
        <b>这是第一个 XHTML 网页</b>
    </body>
</html>
```

（3）单击菜单栏中的“文件”→“另存为”命令，把文件保存在电脑的某个位置，并将其命名为 1_1.html。

（4）启动 IE 浏览器。单击菜单栏中的“文件”→“打开”或“打开页面”命令，这时会弹出一个对话框。单击“浏览”或“选择文件”按钮，找到刚才创建的文件“1_1.html”，选定它并单击“打开”按钮，将会看到对话框中有一行地址，例如“C:\MyDocuments\1_1.html”。单击“确定”按钮，浏览器就会显示这个页面，预览效果如图 1-3 所示。

图 1-3　XHTML 的简单页面

在以上实例中，XHTML 文件中的第一个标签是<html>，此标签告诉浏览器这是 HTML 文件的开始点。文件中最后一个标签是</html>，此标签告诉浏览器这是 HTML 文件的结束点。位于<head>标签和</head>标签之间的文本是头信息，头信息不会显示于浏览器窗口中。<title>标签中的文本是文件的标题，标题会显示在浏览器的标题栏。<body>标签中的文本是将被浏览器显示出来的文本。和标签中的文本将以粗体显示。

这个示例只是一个简单的页面，它在语法上符合 XHTML 的标准，但是如果要作为一个完整的 XHTML 网页，它还缺少一些内容。相关内容将在后面的教程中介绍。

1.2.4　XHTML 的优势

XHTML 作为 XML 应用程序，本质上只是 HTML 4.0 的重新诠释。它采用 HTML 的编程语

言和 XML 的语法结构，是网站向 XML 过渡的第一步。根据 W3C 概括，XHTML 的主要优点有以下 7 点。

1. 可扩展性

作为 XML 的应用程序，XHTML 具有可扩展性。相比于 HTML 的转换进程，它强大的扩展能力为未来语言的转化提供了许多方便。大部分浏览器都已经与 XML 相适应，所以添加一些语言因素只不过是改变一下文件类型的定义和名字空间，也就没必要非要等待浏览器开发者对新因素提供支持。

2. 互用性和可携带性

一个构建正确的 XHTML 文件可以在各种各样的演示设备上重新格式化使用，包括手机、PDA 和其他的一些手持设备。一个 XHTML 文件与其他的 XML 工具和应用程序也是兼容的。对于将来的网络，这是两个具有重大应用功能的宝贵优点。但是大部分网站构建者并不这样看。

3. 推广标准化

XHTML 对所使用的标识标签制订严格的规则，从而解决了困扰 HTML 代码的模糊性和相互矛盾的问题。

4. 提高访问量

XHTML 文件更便于访问，这也就意味着它们能更好地配合屏幕阅读器和其他适应性技术的工作，也就意味着它们更能得到搜索引擎的青睐。

5. 优化压缩网页

XHTML 继续保持 HTML 4.0 的内容与表现层分离的运作。XHTML 标识指定文件结构，表现语言采用 CSS 布局，这便于网站的构建和维护。

6. 加强实例站点

许多力荐的由 HTML 编写的实例站点都使用 XHTML。

7. 提高更多工具的可用性

既然 XHTML 是 XML 的一种应用程序，使用任何流行的 XML 工具都可以来构建、维护和编辑 Web 文件，在 XHTML 文件上使用其他 XML 的应用程序（例如 SVG），也可以使用 XML 工具来应用其他操作，例如转换一个 XHTML 文件成为一个 PDF 文件。

1.3 CSS 入门

本节将介绍 CSS 基本概念以及 CSS 的特点和优势。在理论性地阐述 CSS 这一个抽象的概念之前，先感性地体验 CSS 所带来的视觉效果。

1.3.1 感性体验 CSS 的魅力

如图 1-4 所示是没有添加 CSS 的 XHTML 文件在浏览器中的预览效果。

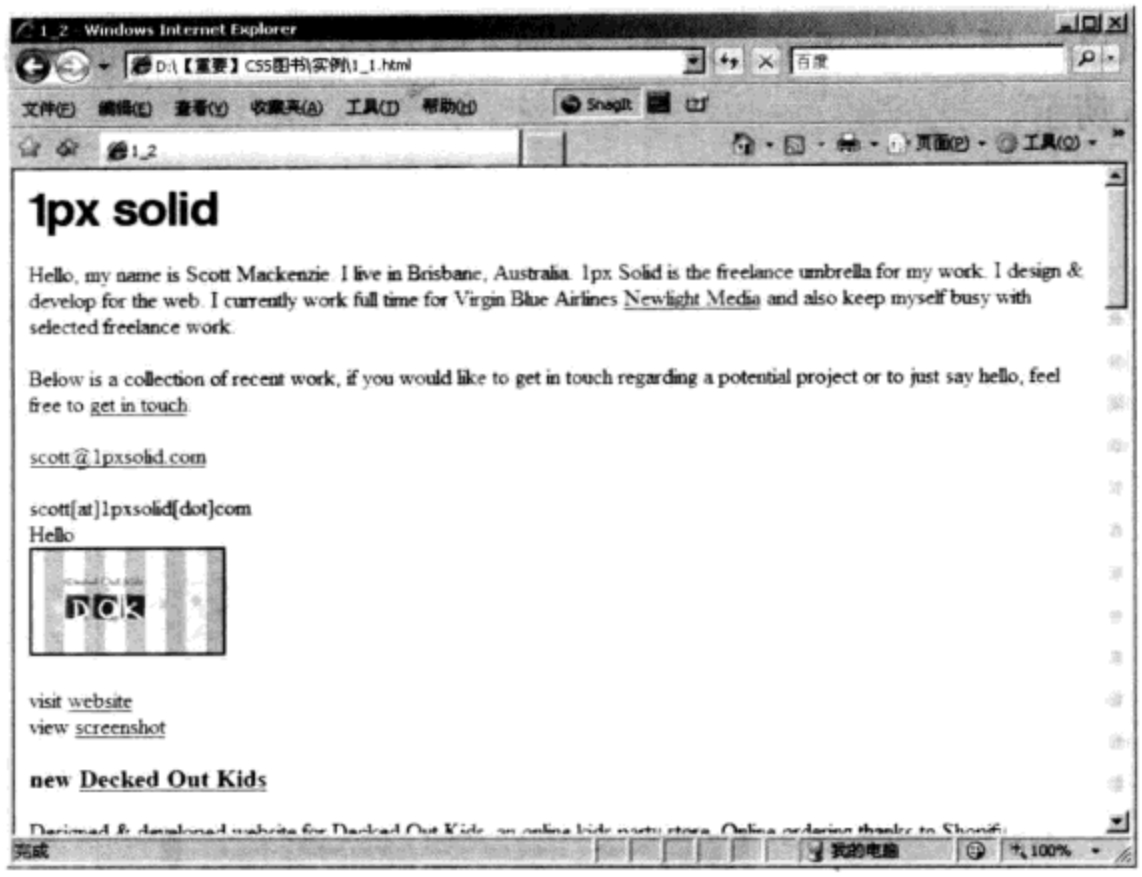


图 1-4　没有添加 CSS 的网页效果

可以看到，这只是一个普普通通的网页。然而，通过给这个文件添加 CSS 样式，可以得到十分美观的网页，如图 1-5 所示。

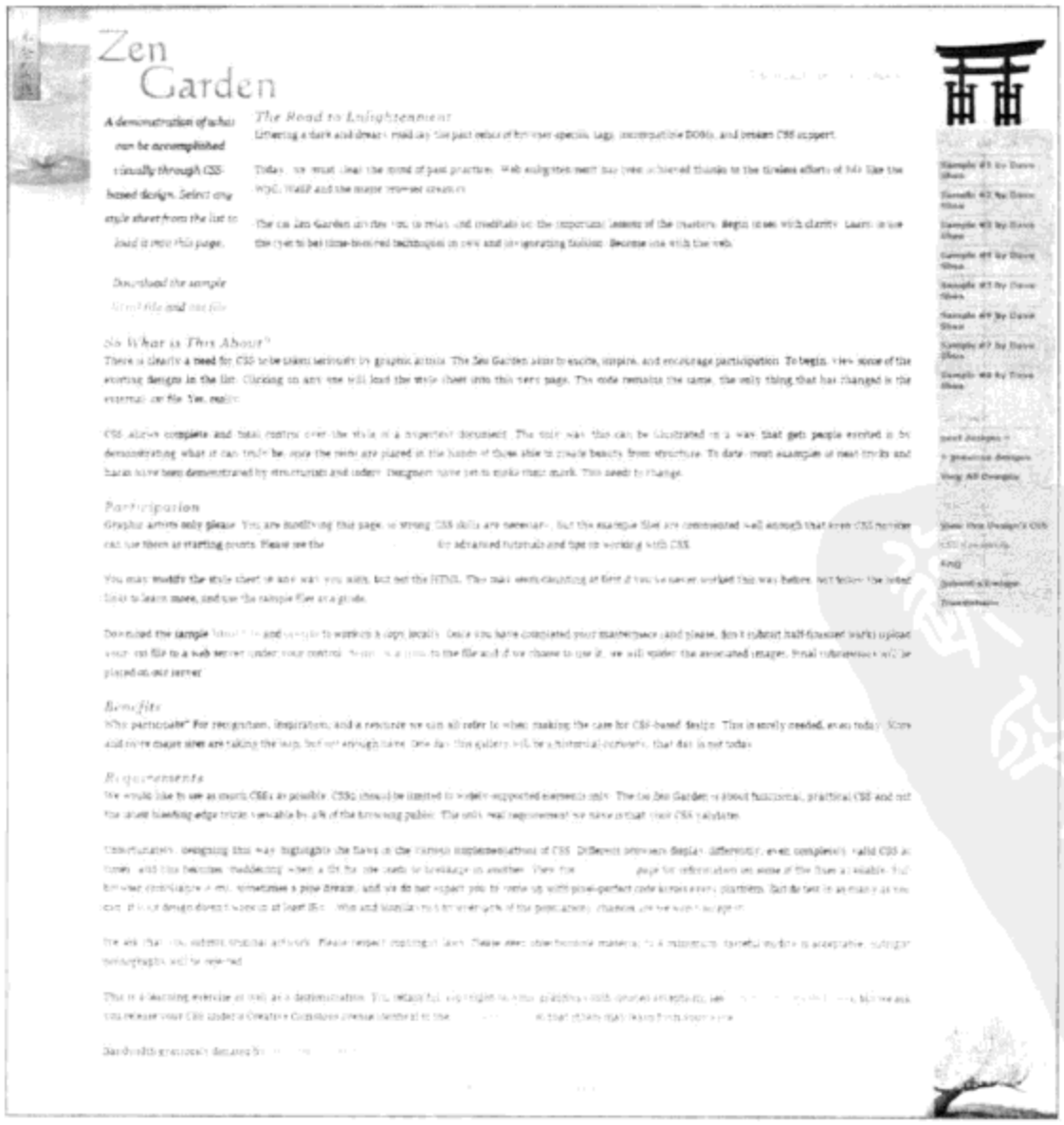


图 1-5　添加 CSS 后的网页效果

在不改动 XHTML 样式的前提下，只通过添加不同的 CSS 规则，就可以得到各种不同样式的网页，如图 1-6 所示。

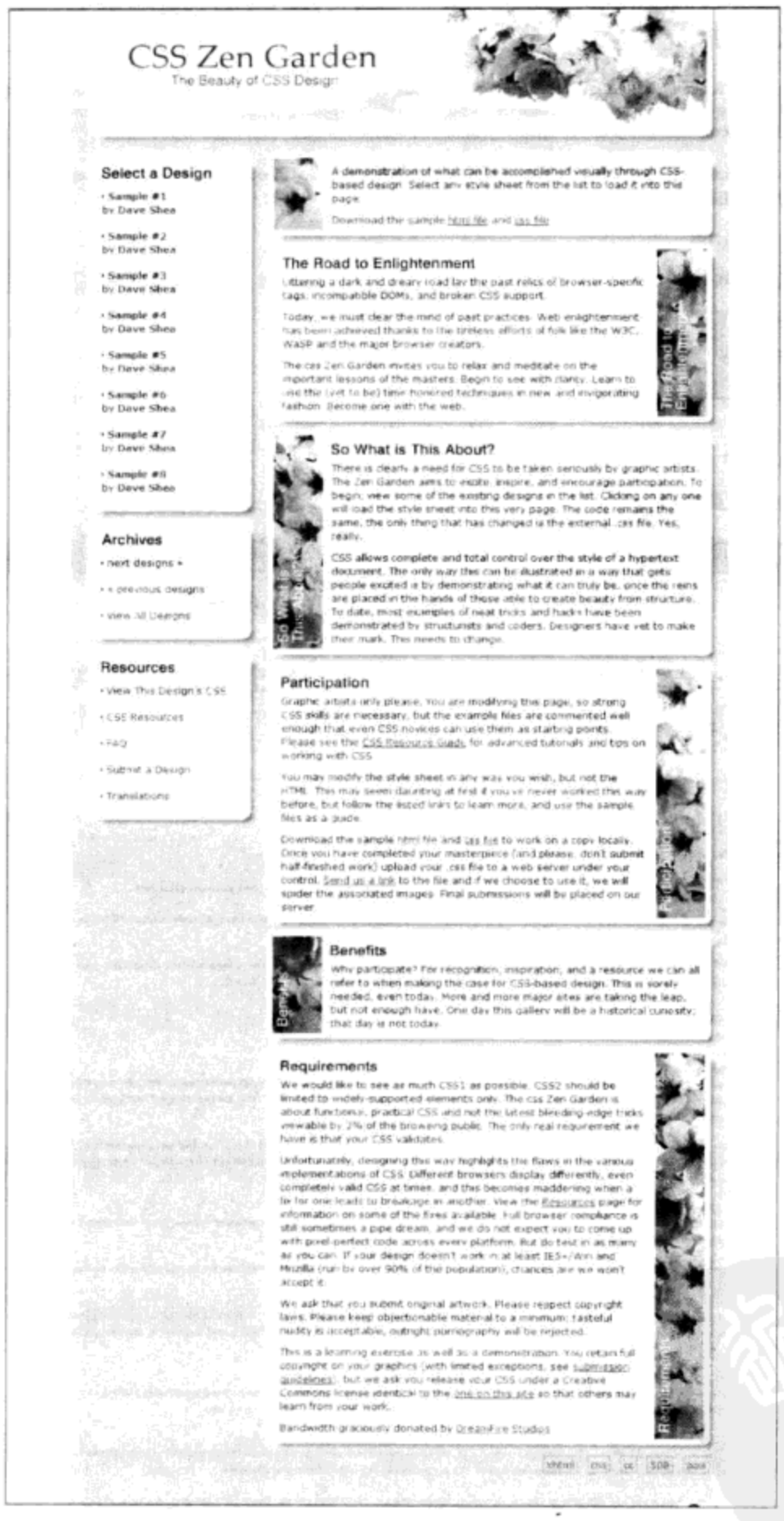


图 1-6　使用不同的 CSS 样式的效果

1.3.2　CSS 的概念

CSS 是 Cascading Style Sheets 的简称，中文含义为“层叠样式单”或“级联样式单”，又简称“样式表”。

为了使网页在视觉上可以有更多元的表现，W3C 在 1996 年推出了 CSS 1.0。1998 年 W3C 正式推出了 CSS 2.0。虽然现在 CSS 3.0 也已经被推出，但仍然处在实验阶段。CSS 2.1 是 W3C 现在推荐使用的。

CSS 的引入主要是用来扩展 XHTML 的，而不是替代 XHTML 的。也就是说 CSS 离不开 XHTML，它只是一项辅助工具，是对 XHTML 功能的一种补充。

CSS 语法的功能就是让网页内容与视觉呈现分离。一方面使得页面维护工作更容易，不会因内容或视觉效果一方的改变影响到另一方，这样的页面设计，也对搜寻引擎更为友善，更容易捕捉到页面内容；另一方面，CSS 也可以增加页面在不同媒介的呈现效果。同一个页面，可依据用途不同（例如在屏幕显示或打印）而自动切换不同的 CSS 语法，呈现最佳化页面。也由于读取页面的媒介越来越多（例如终端设备手机、PDA），CSS 可以弹性调整呈现方式，这都更加彰显 CSS 在网页制作上的优势。

1.3.3　CSS 的特点

且不说过去的网页缺少动感，就是在网页内容的排版布局上也有很多困难，如果不是专业人员或特别有耐心的人，很难让网页按自己的构思和创意来显示信息。即便是掌握了 XHTML 语言精髓的人也要通过多次测试，才能驾驭好这些信息的排版，过程十分漫长和痛苦。

为了促进 Internet 的发展，让更多人早日踏足这个多姿多彩的世界，新的 XHTML 辅助工具呼之欲出。CSS 就是在这种需求下诞生的。

首先，CSS 要做的是为网页上的元素精确定位，可以让网页设计者像导演一样，轻易地控制由文字、图片组成的演员们，在网页这个舞台上按剧本的要求好好地表演。

其次，CSS 把网页上的内容结构和格式控制相分离。浏览者想要看的是网页上的内容结构，而为了让浏览者更好地看到这些信息，就要通过格式控制来帮忙了。以前两者在网页上的分布是交错结合的，查看修改很不方便，而现在把两者分开就会大大方便网页的设计者。内容结构和格式控制相分离，使得网页可以只由内容构成，而将所有网页的格式控制指向某个 CSS 样式表文件，其优势表现在以下两个方面。

- 简化了网页的格式代码，外部的样式表还会被浏览器保存在缓存里，加快了下载显示的速度，也减少了需要上传的代码数量（因为重复设置的格式将被只保存一次）。
- 只要修改保存着网站格式的 CSS 样式表文件就可以改变整个站点的风格特色，在修改页面数量庞大的站点时，显得格外有用。这就避免了一个个网页的修改，大大减少了工作量。

1.3.4　CSS 的优势

CSS 是控制网页布局样式的基础，并能够真正做到网页表现与内容分离的一种样式语言。相对于传统 HTML 的简单样式控制而言，CSS 能够对网页中对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，以及拥有对网页对象和模型样式的控制能力，并能够进

行初步页面交互设计，是目前基于文本展示的最优秀的表现设计语言。归纳起来，CSS 2.0 有以下几个主要优势。

1. 浏览器支持完善

目前 CSS 2.0 版本是众多浏览器支持最完善的版本，最新的浏览器均以 2.0 为 CSS 支持原型进行设计，使用 CSS 2.0 样式设计的网页在众多平台及浏览器下样式表现最为接近。

2. 表现与结构分离

CSS 2.0 真正意义上实现了设计代码与内容分离，而在 CSS 的设计代码中通过 CSS 的内部导入特性，又可以使设计代码根据设计需求进行二次分离。例如为字体专门设计一套样式表，为版式、各个频道等设计一套样式表，根据页面显示的需要重新组织，使得设计代码本身也便于维护和修改。

3. 样式设计控制功能强大

对网页对象的位置排版能够进行像素级的精确控制，支持所有字体、字号样式，优秀的模型控制能力和简单的交互设计能力。

4. 继承性能优越

CSS 2.0 的语言在浏览器的解析顺序上，具体类似 OOP（Object Oriented Programming，OOP，面向对象程序设计）面向对象的基本功能，浏览器能够根据 CSS 的级别先后应用多个样式定义，良好的 CSS 代码设计可以使得代码之间产生继承及重载关系，能够达到最大限度的代码重用，降低代码量及维护成本。

1.3.5 发挥 CSS 的优势

CSS 的应用是本书的重点。相对于结构设计来说，表现层的样式设计变化更丰富，也更难掌握，对于千变万化的网页设计来说，如何将设计编码成机器识别的样式语言则是 CSS 的作用。CSS 丰富的样式表现也对设计者提出了要求。

1. 合理的 CSS 文件结构

虽然 CSS 做到了样式与内容的分离，但 CSS 文件本身也应该拥有良好的层次结构及规范，目的是进一步改善样式设计的可维护性，CSS 本身支持 import 导入功能，针对大型网站的设计，可以使用分离的 CSS 文件来组织样式。例如字体样式可以专门使用 font.css 这样的文件进行编写，表单设计放在 form.css 中，通过合理的组织文件，给后期维护带来便利，也方便网站程序能够根据浏览器版本或是终端设备进行文件的调用，进一步提升 CSS 跨平台的能力。

2. 继承和重用的优势

CSS 的优势就在于其重用性，一段 CSS 设计代码可以为多个区域同时使用。除了重用功能以外，CSS 还可以实现类似面向对象程序设计中的继承机制。例如在 CSS 对应的 XHTML 中，

每一级的标签总是首先使用其本身标签的样式设计，接着使用父级标签的样式，这样部分代码就可以分别放在各级别中，互相发挥作用，统一代码放在最上一级标签。通过这种具有继承机制的功能，能够减小在样式设计中的代码量，进一步改善设计。

3. 设计跨平台的代码

CSS 也有美中不足，由于不同浏览器产品及不同版本之间的渲染方式的不同，它们对 CSS 的解析存在着一定的差异。目前仍然有一些老版本的浏览器，例如 IE 4.0 及 IE 5.0 等，有众多不愿意升级的用户在使用；另外就是 PC 机下与 MAC 机下浏览器产品的不同。针对这些原因，CSS 的设计也应当具有一定的跨平台与兼容性特征，兼容新旧版本的浏览器，注意使用一定的 CSS hack 代码进行编写。

4. 具有良好可用性的 CSS 样式设计

可用性，即随着计算机人机交互技术的发展不断扩充其内容与形式，可用性的目标是使我们的交互式产品（软件、网站）为用户的需求提供最大限度的满足，使得产品更容易被用户使用，它从根本上改变用户与产品交互的主观过程，提升产品价值，为产品及用户带来双方面的效益。CSS 作为样式设计代码，也包含着对于可用性层次的设计内容。

CSS 样式的设计意味着设计者对网站整体风格的把握需要重新考虑，从视觉设计上为了达到最大限度的重用与合理的结构，需要统一的字体、字号和排版形式，这些统一性的设计都有助于视觉设计上的可用性的提升。对于网站上的细节表现，例如链接改变、导航操作等也都是 CSS 在可用性上设计的目标，最终目的都是希望通过良好的设计创造更好的交互式网站以方便用户使用。

5. 基于 DOM 的脚本语言来编写交互

DOM 的产生也是为了实现脚本语言的跨平台与跨浏览器。DOM 的全称是 Document Object Model，即文档对象模型，是由 W3C 制定的一种与浏览器无关的接口，它对网页中的标准组件，例如 HTML 标签制订出技术性的统一规范，使得脚本语言能够访问这些组件，但前提是浏览器需要支持这种基于 DOM 的定义规范。目前大部分浏览器都支持标准的 DOM。使用符合 DOM 的脚本语言基本上不再需要检测浏览器的不同而去编写相对应的代码，同一个代码便能够支持几乎所有的浏览器。JavaScript 就是符合 DOM 标准的脚本语言。

1.4　Web 标准

Web 标准伴随着 Web 表现层技术的发展一直存在于网络技术中，从来没有离开过人们的视线。由于 Web 标准对网站架构进行了丰富的结构定义和技术约定，现在它已经不仅仅只是一堆技术指标，而是引领着网站交互式设计的思想前沿，不仅从网站的代码级设计上带给人们更大的自由度，更使得人们可以通过符合 Web 标准的设计思想来打造高可靠性、高效率以及丰富用户体验的交互平台。

优秀的符合 Web 标准的网站总是能通过良好的设计提高网站可用性，从根本上为网站创造

价值。本节将介绍 Web 标准的概念。

1.4.1　什么是 Web 标准

Web 标准是由 W3C 和其他标准化组织制定的一套规范集合，包含一系列标准，例如大家熟悉的 HTML、XHTML、JavaScript、CSS 等。Web 标准的目的在于创建一个统一的用于 Web 表现层的技术标准，以便通过不同浏览器或终端设备向最终用户展示相同的信息内容。

Web 标准即网站标准。目前通常所说的 Web 标准一般指网站建设采用基于 XHTML 语言的网站设计语言，Web 标准中典型的应用模式是“CSS+DIV”。实际上，Web 标准并不是一个标准，而是一系列标准的集合。

1.4.2　Web 标准的构成

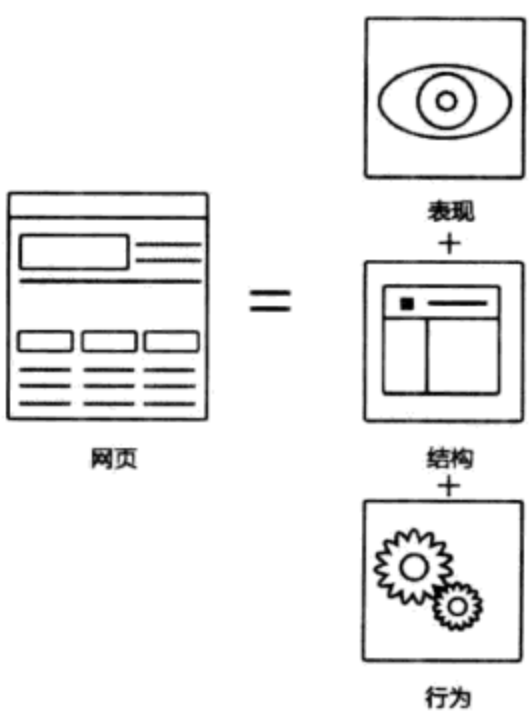


图 1-7　Web 标准的 3 大组成部分

Web 标准由一系列规范组成。由于 Web 设计越来越趋向于整体与结构化，目前的 Web 标准也逐步变为由 3 大部分组成的标准集：表现（Presentation）、结构（Structure）和行为（Behavior），如图 1-7 所示。

1．表现

表现技术用于对已被结构化的信息进行显示上的控制，包含版式、颜色、大小等形式控制。用于表现的 Web 标准技术主要是 CSS 层叠样式表。

2．结构

结构对网页中用到的信息进行整理和分类。用于结构化设计的 Web 标准技术有 HTML、XML、XHTML 等标记语言。

3．行为

行为是指对整个文档内部的一个模型定义及交互行为的编写，用于编写用户可进行交互式操作的文档。表现行为的 Web 标准技术主要有以下两点。

- DOM 文档对象模型：根据 W3C DOM 规范，DOM 是一种让浏览器与内容结构之间沟通的接口，使用者可以借此访问页面上的标准组件。DOM 可以以一种独立于平台和语言的方式访问和修改一个文档的内容和结构。
- ECMAScript（JavaScript 的扩展脚本语言）：是由 CMA（Computer Manufacturers Association）制定的标准脚本语言（JavaScript），用于实现具体界面上的对象的交互操作。

1.4.3　Web 标准的表现层技术

Web 本身由一套非常复杂的技术架构组成，但是其最终目的是为面向浏览器或应用程序的用户提供一个可视化的、便于操作的信息交互平台。

而 Web 表现层技术，指的就是将信息展示给用户并提供给用户交互行为的技术，可以简单理解为样式和技术层面上是一堆程序代码，而表面上带给人们的是视觉所看到的东西。

目前由 W3C 制定的 Web 标准正是这样的一个表现层技术集合，并且 Web 标准是目前唯一跨平台、跨客户端的技术。

1.5　使用 CSS+DIV 建设网站

现在互联网越来越流行建设符合 Web 标准的网站，然而建设符合 Web 标准的网站，就要使用 CSS 样式表和 DIV 标签来编写代码。本节首先介绍 CSS+DIV 的基本概念和使用 CSS+DIV 建站的优势。

1.5.1　CSS+DIV 的含义

DIV 标签是用来为 XHTML 文档内大块的内容提供结构和背景的元素。DIV 的起始标签和结束标签之间的所有内容都是用来构成这个块的，其中所包含元素的特性由 DIV 标签的属性来控制，或者是通过使用样式表格式化这个块来进行控制。

CSS+DIV 是 Web 标准中常用的术语之一，通常为了说明与 HTML 网页设计语言中的表格（table）定位方式的区别。因为 XHTML 网站设计标准中，不再使用表格定位技术，而是采用 CSS+DIV 的方式实现各种定位。

1.5.2　CSS+DIV 网站设计的优势

XHTML 是目前国际上倡导的网站标准设计语言，因为 XHTML 网站设计语言具有的基本特点，这种 CSS＋DIV 模式的网站设计具有一定的优势。

首先，CSS 的极大优势表现在简洁的代码。这对一个大型网站来说，可以节省大量带宽，而且众所周知，搜索引擎喜欢简洁的代码（其真正意义在于，增加了有效关键词占网页总代码的比重），因此使用 CSS＋DIV 的 Web 标准制作的网站具有搜索引擎友好的一定优势。

其次，CSS+DIV 制作的网站使得网站改版相对简单，很多问题只需要改变 CSS 而不需要改动程序，从而降低了网站改版的成本。

最后，CSS+DIV 制作的网站丝毫不比用其他方法制作的效果逊色。如图 1-8 所示为使用 CSS+DIV 制作的 http://procreation.ru/网站。

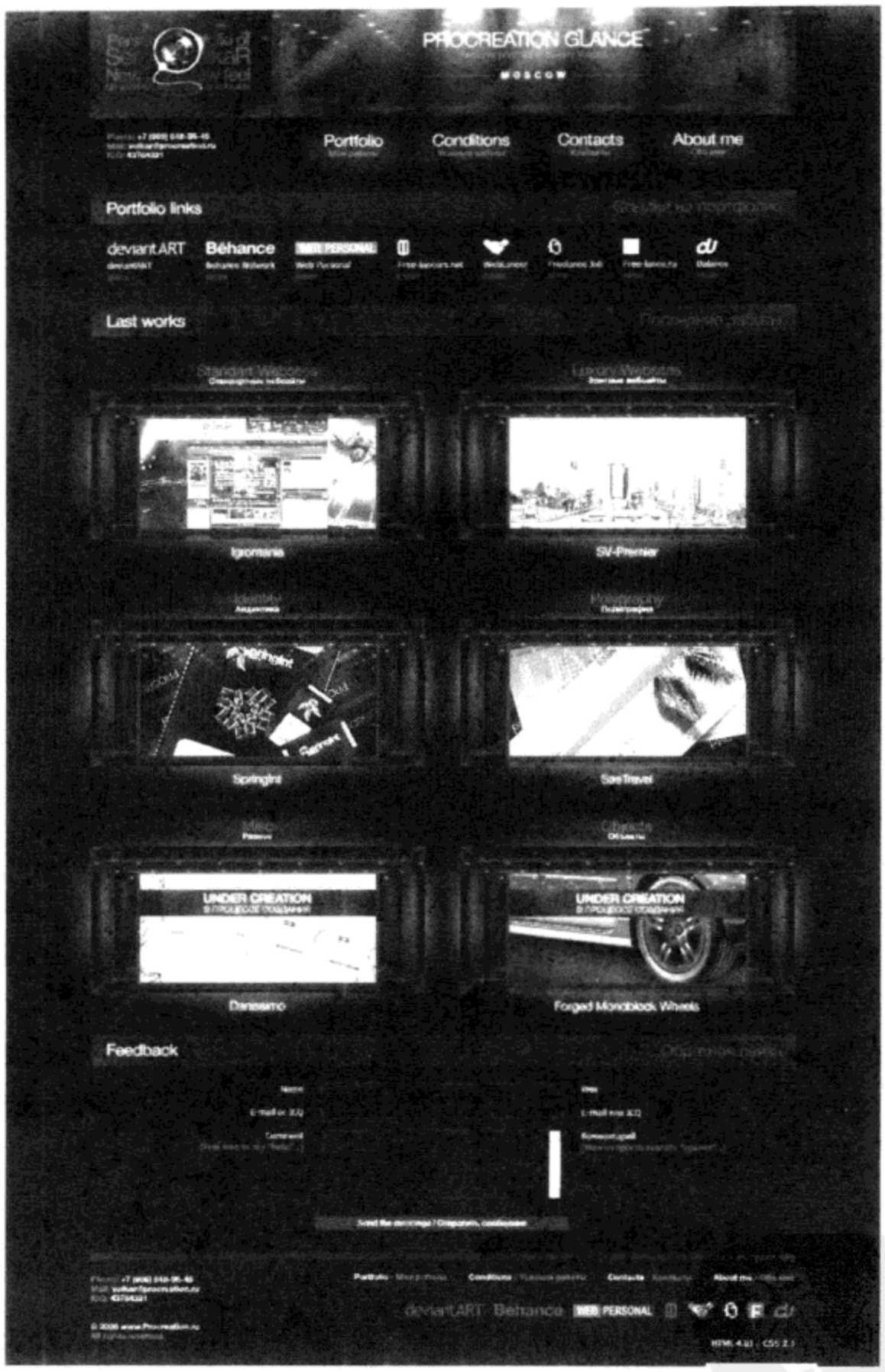


图 1-8 使用 CSS+DIV 制作的 http://procreation.ru/网站

1.6 小结

本章主要讲解了与网页制作有关的基本概念。首先介绍网页的基本构成，然后针对网页的每一个构成部分介绍了每一部分的基本概念，包括 HTML、XHTML 的基本概念，使用 XHTML 的优势，CSS 的特点和优势，还介绍了 Web 标准，最后介绍了使用 CSS+DIV 来建设符合 Web 标准的网站的这一理念。

对本章的知识点进行归纳总结，结构导图如图 1-9 所示。

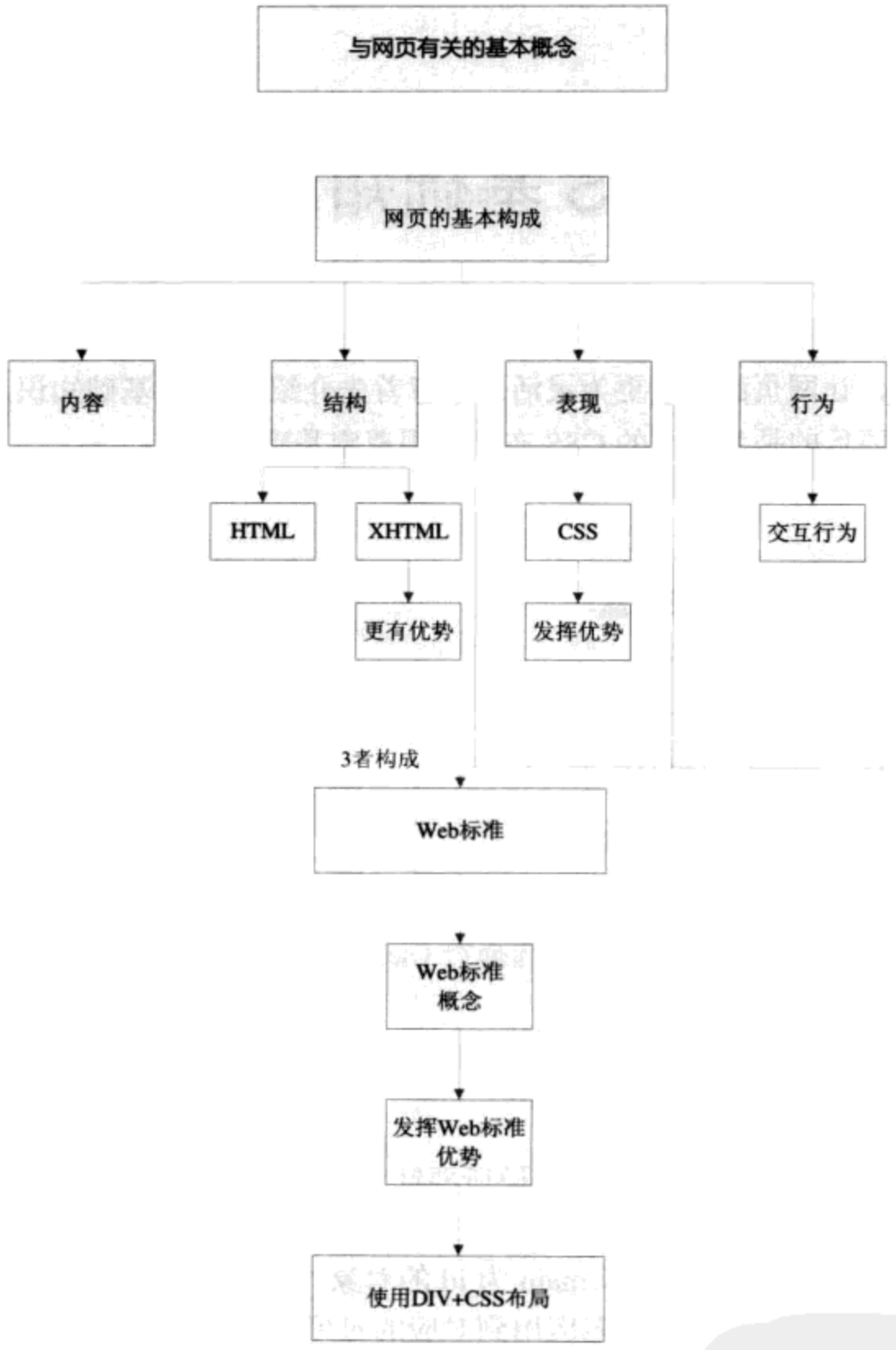


图 1-9　本章知识点结构导图

1.7　习题

1. 简述网页的基本构成。
2. HTML 和 XHTML 有什么区别？XHTML 的优势是什么？
3. 简述 CSS 的概念、特点和优势。
4. 谈谈你对 Web 标准的理解。
5. CSS＋DIV 模式的网站设计优势在哪里？

第2章 CSS 基础知识

CSS 用于网页的排版和风格设计，在 Web 标准建站中是相当重要的工具。CSS 的作用是弥补 XHTML 的不足，让网页的设计更为灵活。本章首先介绍 CSS 的基础知识，作为指引读者的一个入门教程，主要目的是为后面的 CSS 布局应用奠定基础。

2.1 CSS 的基本语法

在使用 CSS 的过程中，首先要接触的是 CSS 的语法结构。CSS 语法包括了 CSS 样式的基本构成元素：选择符、属性以及属性值。本节介绍如何使用 CSS 的语法来编写 CSS 样式。

2.1.1 CSS 的基本语法构成

一个 CSS 样式的语法由 3 部分构成：选择符（selector）、属性（property）和属性值（value）。基本语法代码如下。

```
selector {property: value}
```

在以上代码中，3 个部分的含义如下。

- 选择符（selector）：指这组样式编码所要针对的对象，可以是一个 XHTML 标签，例如 body、h1，也可以是定义了特定 id 或 class 的标签，例如#main 选择符表示选择 ，即一个被指定了 main 为 id 的对象。浏览器将对 CSS 选择符进行严格的解析，每一组样式均会被浏览器应用到对应的对象上。
- 属性（property）：是 CSS 样式控制的核心，对于每一个 XHTML 中的标签，CSS 都提供了丰富的样式属性，例如颜色、大小、定位、浮动方式等。
- 属性值（value）：形式有两种，一种是指定范围的值，例如 float 属性，只可能应用 left、right、none 三种值；另一种为数值，例如 width 能够使用 0～9999px，或指定的其他数学单位。

2.1.2 一个 CSS 样式的简单实例

现在来看一个关于 CSS 样式的简单实例，来体验一下 CSS 的使用方法。在一个 XHTML 的文件中添加 CSS 样式表的代码如下。

```
<html>
    <head>
        <title>1_1</title>
        <style type="text/css">
            body{
                background:#fff;
                color:#333;
                font-size:12px;
                margin-top:5px;
                font-family:"宋体";
            }
        </style>
    </head>
    <body>
        这是第一个网页<b>这里的文字是粗体</b>
    </body>
</html>
```

以上代码中，在 XHTML 头信息标识符<head>里添加 CSS 样式，将 CSS 样式表统一放置在一个固定的位置，即<style></style>之间。

注意：

本书在讲解后面的 CSS 实例代码时，只列出 CSS 样式表部分的代码，均省略其余相同代码的部分。

在实际应用中，当给一个段落设置 CSS 样式的时候，往往使用下面类似的应用形式。

```
p {color:blue}
```

在以上代码中，p 就是选择符，即选择了页面中的<p>这个标签。color 就是属性，这个属性用于控制对象内元素的颜色值，页面中 p 元素内的颜色就是通过这组 CSS 编码定义成了蓝色。

注意：

XHTML 中所有的标签都可以作为选择符。

另外，除了单个属性的定义，同样可以为一个标签定义一个甚至更多个属性，每个属性之间使用分号隔开，代码如下。

```
p { text-align:center; color:red }
```

在以上代码中，p 样式就包含两个属性，一个是对齐方式为居中，另一个是字体颜色为红，中间用分号隔开。为了提高 CSS 样式代码的可读性，也可以分行写，代码如下。

```
p{
    text-align: center;                    /*设置文本居中对齐*/
    color: black;                          /*设置字体颜色*/
    font-family: arial                     /*设置字体类型*/
}
```

此例的效果是使 p 标签内的元素文本居中、颜色为黑色、字体为 Arial。

2.2　CSS 的选择符

将 CSS 应用到 XHTML 之中，首先要做的就是选择合适的选择符，选择符是 CSS 控制 XHTML 文档中对象的一种方式。简单地说，它用于告诉浏览器这段代码样式将应用到哪个对象。CSS 的选择符有多种形式，下面分别介绍每种选择符的使用方法。

2.2.1　类型选择符

所谓类型选择符，就是网页中已有的标签类型作为名称的选择符，即以 XHTML 标签作为选择符，例如 body、p、table……

例如在为网页设置文字颜色的时候，可以使用下面的 CSS 代码。

```
body {
    color: black                                          /*设置字体颜色*/
}
```

此例的效果是使页面中的文字为黑色。

2.2.2　群组选择符

当不同的 XHTML 对象具有相同的样式时，为样式指定多个对象，每个对象之间用“,”分隔，可以减少样式重复定义，这样的选择符叫做群组选择符。

除了对单个 XHTML 对象进行样式指定外，同样还可以对一组对象进行相同的样式指定。例如在为若干标签设置字体类型以及字号的时候，可以使用下面的 CSS 代码。

```
h1,h2,h3,span,p{
    font-size:12px;                                       /*设置字号*/
    font-family: arial;                                   /*设置字体类型*/
}
```

使用逗号对选择符进行分隔，使得页面中所有的 h1、h2、h3、span 以及 p 都将具有相同的样式定义。这样做的好处是对页面中需要使用相同样式的地方只需要书写一次样式即可，以减少代码量，改善 CSS 代码的结构。

2.2.3　包含选择符

所谓包含选择符，就是指选择符组合中前一个对象包含后一个对象，对象之间使用空格作为分隔。如果对某一个元素中的子对象进行样式指定时，包含选择符就被派上了用场。例如在为 h1 标签下的 span 标签设置字体粗细样式的时候，可以使用下面的 CSS 代码。

```
h1 span{
    font-weight:bold;                                     /*设置粗体*/
}
```

上面的代码对 XHTML 标签 h1 下面的标签 span 进行样式指定，此示例的效果是指定只有标签 h1 的子标签 span 的字体加粗。

使用包含选择符的好处是，能够避免过多的 id 及 class 的设置，直接对所需元素进行设置。包含选择符除了可以二级包含，还可以更多级的包含。例如在为多级标签“body\h1\span\strong”设置样式的时候，可以使用下面的 CSS 代码。

```
body h1 span strong{
    font-weight:bold;                                    /*设置粗体*/
}
```

2.2.4　id、class 选择符

id 选择符及 class 选择符均是 CSS 提供的由用户自定义标签名称的一种选择符模式，用户可以使用 id 及 class 对页面中的 XHTML 标签名称进行自定义，从而达到扩展 XHTML 标签及组合 XHTML 标签的目的。自定义名称的方式也有助于用户细化自身的界面结构，使用符合页面需求的名称来进行结构设计，从而增强代码的可读性。

1．id 选择符

id 选择符是根据 DOM 文档对象模型原理所设置的选择符类型。对一个网页而言，其中的每一个标签，均可以使用一个 id=""的形式对 id 属性进行一个名称的指定。id 可以理解为一个标识，在网页中的每个 id 名称只能使用一次。

表示一个容器的 XHTML 代码如下。

```
<div id="content"></div>
```

如上例所示，XHTML 中的标签 div 被指定了一个名为 content 的 id。在 CSS 样式中，id 选择符使用“#”进行标识，如果需要对 id 为 content 的标签设置样式，应当使用如下代码。

```
#content{
    font-size: 14px;                                     /*设置字号*/
    line-height: 130%                                    /*设置行高*/
}
```

此示例的效果是设置字号为 14 像素、行高为 130%。

id 的基本作用是对每一个页面中唯一出现的元素进行定义，如可以对导航命名为 nav，对网页头部和底部的内容分别命名为 header 和 footer。这方面内容在后面讲解 CSS 布局的时候，会做详细介绍。

2．class 选择符

如果说 id 是对 XHTML 标签扩展的话，那么 class 选择符应该是对 XHTML 多个标签的一种组合。对网页设计而言，可以对 XHTML 标签使用一个 class=""的型式对 class 属性进行名称指定。与 id 不同的是，class 允许重复使用，如页面中有多个元素，都可以使用同一个 class 定义，代码如下。

```
<div class="s1"></div>
<h1 class="s1"></h1>
<h3 class="s1"></h3>
```

使用 class 的好处是，对不同的 XHTML 标签，CSS 可以直接根据 class 名称来进行样式设置，上述 XHTML 的 CSS 样式代码如下。

```
.s1{
    margin: 10px;                                    /*设置外边距*/
    background-color: blue;                          /*设置背景颜色*/
}
```

class 在 CSS 中的使用是点符号加 class 名称的形式，如上例所示，对 class 为 s1 的对象进行样式指定。无论是什么 XHTML 标签，页面中所有使用了 class="s1"的标签的样式均被设置了。class 选择符也是对 CSS 代码重用性的很好体现，众多标签均可以使用同一个 class 来进行样式指定，不再需要逐个编写样式代码。

2.2.5　标签指定式选择符

标签指定式选择符在对标签选择的精确度上介于标签选择符及 id/class 选择符之间，并且标签指定式选择符之间没有空格和其他分隔符，也是一种经常能够使用到的选择符形式。

如果既想使用 id 或 class，也想同时使用标签选择符，可以使用标签指定式选择符。标签指定选择符的格式如下。

```
h1#content{}
```

表示针对所有 id 为 content 的 h1 标签。

```
h1.p1{}
```

表示针对所有 class 为 p1 的 h1 标签。

2.2.6　组合选择符

所谓组合选择符，就是对前几节讲到的所有 CSS 选择符而言，无论是什么样的选择符，均可以进行组合使用。组合选择符可以是包含选择符组成的包含关系、单一的选择符组成的并列关系、若干包含选择符组成的并列关系或标签指定式选择符与单一选择符组成的包含关系。

组合一：

```
h1 .p1{}
```

表示 h1 标签下的所有 class 为 p1 的标签，这是一种包含选择符组成的包含关系。

组合二：

```
#content h1{}
```

表示 id 为 content 的标签下的所有 h1 标签，这也是一种包含选择符组成的包含关系。

组合三：

```
h1 .p1,#content h1{}
```

以上二者进行群组选择，中间使用“,”分隔，这是一种两个包含选择符组成的并列关系。

组合四：

```
h1#content h2{}
```

id 为 content 的 h1 标签下的 h2 标签，这是一种使用标签指定式选择符表示的包含关系。CSS 在选择符的使用上可以说是非常自由的，根据页面需要，可以灵活地使用各种选择符。

2.2.7　伪类和伪对象

伪类和伪对象是一种特殊的类和对象，它由 CSS 自动支持，属于 CSS 的一种扩展类型的对象，名称不能被用户自定义，使用时只能够按标准格式应用，表示方法如下。

```
选择符:伪类/伪对象
```

例如我们想要设置背景颜色，代码如下。

```
a:hover{
    background-color: #333333;                    /*设置背景颜色*/
}
```

由此可以看出，伪类和伪对象的标准表示方式如下。

上面所示的代码中，hover 就是一个伪类，用于指定链接标签 a 的光标移上状态。CSS 内置了几个标准的伪类用于用户的样式定义，如表 2-1 所示。

表 2-1　伪类示意

伪类	用途
:link	a 标签未被访问前的样式
:hover	当鼠标移上去时的样式
:active	对象被用户点击或者被点击释放之前的样式
:visited	a 标签链接被访问后的样式

实际上，除了对于链接样式控制的:hover、:active 等几个伪类之外，大多数伪类及伪对象在实际应用中并不常用到。一般所接触到的 CSS 布局中，大部分是关于排版及样式的，对于伪类及伪对象所支持的多数属性基本上很少用到，但是不排除使用的可能，由此也可看到 CSS 为样式及样式中对象的逻辑关系、对象组织提供了很多便利的接口。

2.2.8　通配选择符

所谓通配选择符，就是可以使用模糊指定的方式来选择对象。如果接触过 DOS 命令或者是 Word 中的替换功能，就可以很容易理解通配操作。通配是指使用字符替代不确定的字，例如在 DOS 命令中，使用*.*表示所有文件，使用*.bat 表示所有扩展名为 bat 的文件。因此，CSS 的通配选择符使用*作为关键字，使用方法如下。

```
*{
    color: blue;                                  /*设置字体颜色*/
}
```

*号表示所有对象，包含所有不同 id、不同 class 的 XHTML 的所有标签。使用以上的选择符进行样式定义，页面中所有对象都会使用 color:blue;的字体颜色。

2.3　CSS 的常用属性及属性值

选择合适的选择符之后，就要开始对 CSS 设置属性以及属性值。CSS 样式的属性种类非常丰富，能够实现各种各样的表现形式，CSS 之所以功能强大，和 CSS 的属性密不可分。下面介绍 CSS 最基本的几种属性及其属性值。

2.3.1　color 颜色属性

CSS 颜色属性用来设定前景色，通常用于设定元素的字体颜色。CSS 可以处理 16,777,216 种颜色，可以使用颜色名称、RGB 值或十六进制代码对颜色进行定义。CSS 结构代码如下。

```
Color: 颜色名称| RGB 值| 十六进制颜色代码
```

属性值的使用方法如下。

1．17 种预先确定的颜色名称

CSS 中预先确定的 17 种颜色的名称如下。

aqua（水绿色）、black（黑色）、blue（纯蓝）、fuchsia（紫红色）、gray（灰色）、green（纯绿色）、lime（酸橙色）、maroon（栗色）、navy（海军蓝）、olive（橄榄色）、orange（橙色）、purple（紫色）、red（红色）、silver（银灰色）、teal（水鸭色）、white（纯白色）、yellow（纯黄色）

transparent 也是一个正确的值。

2．RGB 值的格式是 red、green、blue

RGB 代表红、绿、蓝三个通道的颜色，通过对红（R）、绿（G）、蓝（B）3 个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色，每一个颜色分配一个 0～255 范围内的强度值。

例如：纯红色 R 值为 255、G 值为 0、B 值为 0；灰色的 R、G、B 三个值相等，但不能为 0 或 255；白色的 R、G、B 值都为 255；黑色的 R、G、B 值都为 0。RGB 图像只使用 3 种颜色就可以使它们按照不同的比例混合，在屏幕上显示 16581375 种颜色。这些值也可以用百分比表示。

3．十六进制的三或六个数字长度前面带上#字符

三个长度是六个长度的压缩形式。例如#f00 是#ff0000 的缩写形式，#c96 是#cc9966 的缩写形式。三位数很好理解，例如 RGB 值，第一个数字是红色，第二个数字是绿色，第三个数字是蓝色。但六位数给予更多的颜色控制。17 种预先确定的颜色的名称、RGB 值以及十六进制对照表如图 2-1 所示。

	fuchsia	灯笼海棠(紫红色)	#FF00FF	255, 0, 255
	blue	纯蓝	#0000FF	0, 0, 255
	navy	海军蓝	#000080	0, 0, 128
	aqua	水绿色	#00FFFF	0, 255, 255
	teal	水鸭色	#008080	0, 128, 128
	lime	酸橙色	#00FF00	0, 255, 0
	green	纯绿	#008000	0, 128, 0
	yellow	纯黄	#FFFF00	255, 255, 0
	olive	橄榄	#808000	128, 128, 0
	Orange	橙色	#FFA500	255, 165, 0
	red	纯红	#FF0000	255, 0, 0
	Maroon	栗色	#800000	128, 0, 0
	White	纯白	#FFFFFF	255, 255, 255
	Silver	银白色	#C0C0C0	192, 192, 192
	gray	灰色	#808080	128, 128, 128
	black	纯黑	#000000	0, 0, 0

图 2-1　17 种颜色对照表

2.3.2　常用的 CSS 长度单位

CSS 的长度单位分为绝对长度单位和相对长度单位。相对长度单位中的相对二字，表明了其长度单位会随着它的参考值的变化而变化，不是固定的。绝对长度单位是一个固定的值，常用的有 mm，就是毫米的意思。

1．CSS 的常用长度单位

CSS 的常用长度单位有 em、ex、px、pt、%，下面分别介绍每个长度单位的使用方法。

- em：这是一个相对长度单位。相对于当前对象内文本的字体尺寸。如果当前对行内文本的字体尺寸未做设置，则相对于浏览器的默认字体尺寸。
- ex：这是一个相对长度单位。相对于字符“x”的高度，此高度通常为字体尺寸的一半，即如果设字体为 1ex，那么大写字母的高度与小写字母的高度是一样的。由于某些不支持的原因，一般不推荐使用该长度单位。
- px：这是一个相对长度单位。px 的意思就是像素（pixel）。像素是相对于显示器屏幕分辨率而言的。例如，Windows 用户所使用的分辨率一般是 96 像素/英寸，而 MAC 用户所使用的分辨率一般是 72 像素/英寸。
- pt：这是一个绝对长度单位。pt 的意思就是点（point）。这个单位是为打印而设计的。
- %：这是一个相对长度单位。%就是百分比值（percentage），它和 em 长度单位相似。

2．em 和 px 的区别与联系

对计算机的屏幕设备而言，像素或者说 px 是一个最基本的单位，其他所有的单位都和像素成一个固定的比例换算关系。所有的长度单位基于屏幕进行显示的时候，都统一先换算成像素，然后再进行显示。所以，对计算机的屏幕而言，相对长度和绝对长度没有本质差别。任何单位其实都是像素，差别只是比例不同。

em 是指字体高，任意浏览器的默认字体高都是 16px。所以未经调整的浏览器都符合 1em=16px，这就是 em 和 px 之间的换算关系。例如 12px=0.75em，10px=0.625em。em 有如下特点。

- em 的值并不是固定的。
- em 会继承父级元素的字号。

2.3.3 百分比值

一个百分比值由可选的正号“+”或负号“-”接着一个数字组成，还可以使用百分号“%”表示。在一个百分比值之中是没有空格的。

百分比值相对于其他数值，同样用于定义每个属性。CSS 的百分比值是由上级元素的相应属性值而决定的。CSS 百分比的属性值在很多属性中都可以使用到，例如长度单位、透明度、缩放比例等。最经常使用的百分比值是相对于元素的字号。CSS 的字号属性的格式如下。

```
font-size:absolute-size | relative-size | length | percentage
```

各项含义如下。

- absolute-size：绝对字体尺寸。
- relative-size：相对字体尺寸。
- length：长度表示法。
- percentage：百分比表示法。

这里使用百分比表示法来表示字体的大小，代码如下。

```
p{
    font-size:120%;                                    /*设置字号*/
}
```

在以上代码中，将字体设置为默认值或者是父级对象的元素字号的 120%。

2.3.4 URL 路径

URL 是统一资源定位符（Uniform Resource Locator），是用于完整地描述 Internet 上网页和其他资源地址的一种标识方法。

Internet 上的每一个网页都具有一个唯一的名称标识，通常称之为 URL 地址，这种地址可以是本地磁盘，也可以是局域网上的某一台计算机，更多的是 Internet 上的站点。简单地说，URL 就是 Web 地址，俗称“网址”。URL 的一般格式如下（带[]的为可选项）。

```
protocol :// hostname[:port] / path / [;parameters][?query]#fragment
```

- protocol（协议）：指定使用的传输协议，最常用的是 HTTP 协议，它也是目前 WWW 中应用最广的协议。

- hostname（主机名）：是指存放资源的服务器的域名系统（DNS）主机名或 IP 地址。有时，在主机名前也可以包含连接到服务器所需的用户名和密码。
- :port（端口号）：整数，可选，省略时使用方案的默认端口。各种传输协议都有默认的端口号，例如 http 的默认端口为 80。如果输入时省略，则使用默认端口号。有时候出于安全或其他考虑，可以在服务器上对端口进行重定义，即采用非标准端口号，此时，URL 中就不能省略端口号这一项。
- path（路径）：由零或多个“/”符号隔开的字符串，一般用来表示主机上的一个目录或文件地址。
- ;parameters（参数）：这是用于指定特殊参数的可选项。
- ?query（查询）：可选，用于给动态网页传递参数，可有多个参数，用“&”符号隔开，每个参数的名和值用“=”符号隔开。
- fragment：信息片断，字符串，用于指定网络资源中的片断。例如一个网页中有多个名词解释，可使用 fragment 直接定位到某一名词解释。

CSS 属性中使用到的 URL，一般是指图片的地址，正确的格式如下。

```
url(foo);
```

foo 是一个 URL，这里 URL 可以选择使用单引号" "或者双引号" "，并且在 URL 之前或之后可以包含空格。在 URL 中的括弧、逗号、空格、单引号或双引号必须避开反斜杠。下面列举了一个网络上的图片的 URL 路径。

```
url(http://i1.sinaimg.cn/book/temp/2008-02-14/pic.jpg);
```

2.4　CSS 的继承性

CSS 的继承规则是外部的元素样式会保留下来继承给这个元素所包含的其他元素。事实上，大部分的元素中嵌套的元素都会继承外层元素指定的属性值，有时会把很多层嵌套的样式叠加在一起。例如在 XHTML 的标签 div 中嵌套标签 p，代码如下。

```
div{
    color: red;                                    /*设置字体颜色*/
    font-size:12px                                 /*设置字号*/
}
p{
}
```

此示例的效果是使标签 div 内的文字颜色为红色、文字大小为 12 像素。标签 p 内没有属性指定，但是它会继承标签 div 所定义的属性。

注意：

某些特殊情况下内部选择符不继承周围选择符的值。例如，上边界属性值是不会继承的，一个段落不会具有同文档 body 一样的上边界值。

另外，当 CSS 样式表继承遇到冲突时，总是以最后定义的样式为准。如果上例中定义了标签 p 的颜色，代码如下。

```
div{
    color: red;                                    /*设置字体颜色*/
    font-size: 12px                                /*设置字号*/
}
p{
    color: blue                                    /*设置字体颜色*/
}
```

此示例的效果是使段落的文字颜色为蓝色、字号为 12 像素。可以看到，段落里的文字大小为 12 像素是继承 div 属性的，而 color 属性则是依照最后定义的。

2.5　CSS 的添加方法

CSS 编码可以使用多种方式灵活地应用到所设计的 XHTML 页面之中，选择方式可以根据设计的不同要求来制订。添加方法大概可以分成 4 种：在标识符里添加 CSS、在<head></head>里添加 CSS、链接样式表和联合使用样式表。下面分别讲解这 4 种方法。

2.5.1　在 XHTML 标识符里添加 CSS

在 XHTML 标识符里添加 CSS 样式，又称 CSS 的行间样式，类似于如下代码的格式。

```
<h1 style="font-size:12px; color:#00FFFF; font-weight:normal">
    [...文本示例...]
</h1>
```

行间样式表由 XHTML 中的元素的 style 属性所支持，只需要将 CSS 代码用“;”隔开，书写在 style=""之中便可以完成对当前标签的样式定义，是 CSS 样式定义的一种基本形式。

笔者并不推荐使用这样的样式表进行编写。行间样式表仅仅是 XHTML 标签对 style 属性的支持所产生的一种 CSS 样式编写方式，并不符合表现与内容分离的 Web 标准模式。

使用行间样式表与使用表格式布局从代码结构上来说完全相同，只不过它利用了 CSS 对元素精确控制的优势。这种方式仅仅在需要调试 CSS 样式时临时使用。

2.5.2　在 XHTML 头信息标识符<head>里添加 CSS

在 XHTML 头信息标识符<head>里添加 CSS 样式的方法，又称内部样式表。内部样式表与行间样式表的相似之处都是将 CSS 样式编写在页面之中，不同的是，内部样式表可以将样式表统一放置在一个固定的位置，代码如下。

```
<!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>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>CSS 内部样式表</title>
        <style type="text/css">
            ul {
                padding: 0px;
                margin: 0px;
                list-style: none;
            }
            li {
                float: left;
                width: 160px;
            }
        </style>
    </head>

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

样式表作为页面中的一个单独部分，用<style></style>标签标记在<head></head>之中。内部样式表是 CSS 样式编码的初级应用形式，但仍然不是本书推荐的编写方式，它只能够针对当前页面有效，不能够跨页面执行，达不到 CSS 代码重用的目的。

注意:

本书的大部分实例都使用内部样式表方式进行 CSS 编码，是因为本书中的大部分实例均由单独元素及单独页面组成，将 CSS 样式编码编写在单独页面有利于本书的实例整理。在实际大型网站的开发中，还是应当使用后面介绍的外部链接样式表。

2.5.3　链接样式表

链接样式表是 CSS 应用中最好的一种形式。将 CSS 样式编码单独编写在一个独立文件之中，由网页进行调用，多个网页可以调用同一个样式文件，因此能够实现代码的最大化重用及网站文件的最优化配置，也是本书推荐的 CSS 编码方式。

```
    <!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>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>CSS 内部样式表</title>
            <link  rel="stylesheet"  rev="stylesheet"  href="style.css"  type="text/css"  me-
dia="screen" />
```

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

如上述实例代码所示，在<head>中使用<link>标签，可以将 link 指定为 stylesheet 样式表方式，并使用 href="style.css"指明样式表文件的路径，只需要将样式表单独编写在 style.css 文件之中，便可以使页面应用在 style.css 中所定义的样式。media 是可选的属性，表示使用样式表的网页将用什么媒体输出，取值范围如下。

- screen（默认）：输出到电脑屏幕。
- print：输出到打印机。
- TV：输出到电视机。
- projection：输出到投影仪。
- aural：输出到扬声器。
- braille：输出到凸字触觉感知设备。
- tty：输出到电传打字机。
- all：输出到以上所有设备。

如果要输出到多种媒体，可以用逗号分隔取值表。

rel 属性表示样式表将以何种方式与 HTML 文档结合，取值范围如下。

- stylesheet：指定一个外部的样式表。
- alternate stylesheet：指定使用一个交互样式表。

CSS 在页面中的应用方式主要是实现良好的网站文件管理以及样式管理，分离式的结构有助于合理划分内容（XHTML 中的内容）与表现（样式表）。在这里完成了第一步，做到了 CSS 与 XHTML 的分离。

2.5.4　联合使用样式表

在内部链接样式表里面，可以联合导入外部样式表使用，同样是添加在 XHTML 的头信息标识符<head>里，格式如下。

```
@import 外部样式表
```

下面以一个实例说明联合使用样式表的方法，代码如下。

```
<head>
    <style type="text/css">
        < !--
            @import "*.css"
            其他样式表的声明
        -->
    </style>
```

```
</head>
```

以@import 开头的联合样式表导入方法和链接样式表的方法很相似，但联合样式表输入方式更有优势。因为联合法可以在链接外部样式表的同时，针对该网页的具体情况，做出别的网页不需要的样式规则。

使用联合样式表的时候，需要注意以下几点。

- 联合法输入样式表必须以@import 开头。
- 如果同时输入多个样式表有冲突，将按照第一个输入的样式表对网页排版。
- 如果输入的样式表和网页里的样式规则有冲突，将使用外部的样式表。

2.6　CSS 的开发环境

由于 CSS 代码是 CSS 应用的核心，所以本书中的大部分实例均采用直接对代码讲解的形式。制作 CSS 使用到的软件主要有两种，分别是编辑软件和浏览软件。

2.6.1　编辑软件

笔者推荐读者使用 Dreamweaver CS5 作为编辑器。可以使用 Dreamweaver CS5 作为代码编辑器来编写 XHTML 与 CSS，Dreamweaver 软件中的语法加亮功能能够更好地帮助设计者分辨各种代码段，并且 Dreamweaver 的可视化编辑窗口也与浏览器的显示保持一致，如图 2-2 和图 2-3 所示。

在 Dreamweaver CS5 中编辑好代码后，可以按【F12】功能键或直接使用 IE 浏览器打开网页文件来查看实际的浏览效果，如图 2-4 所示。

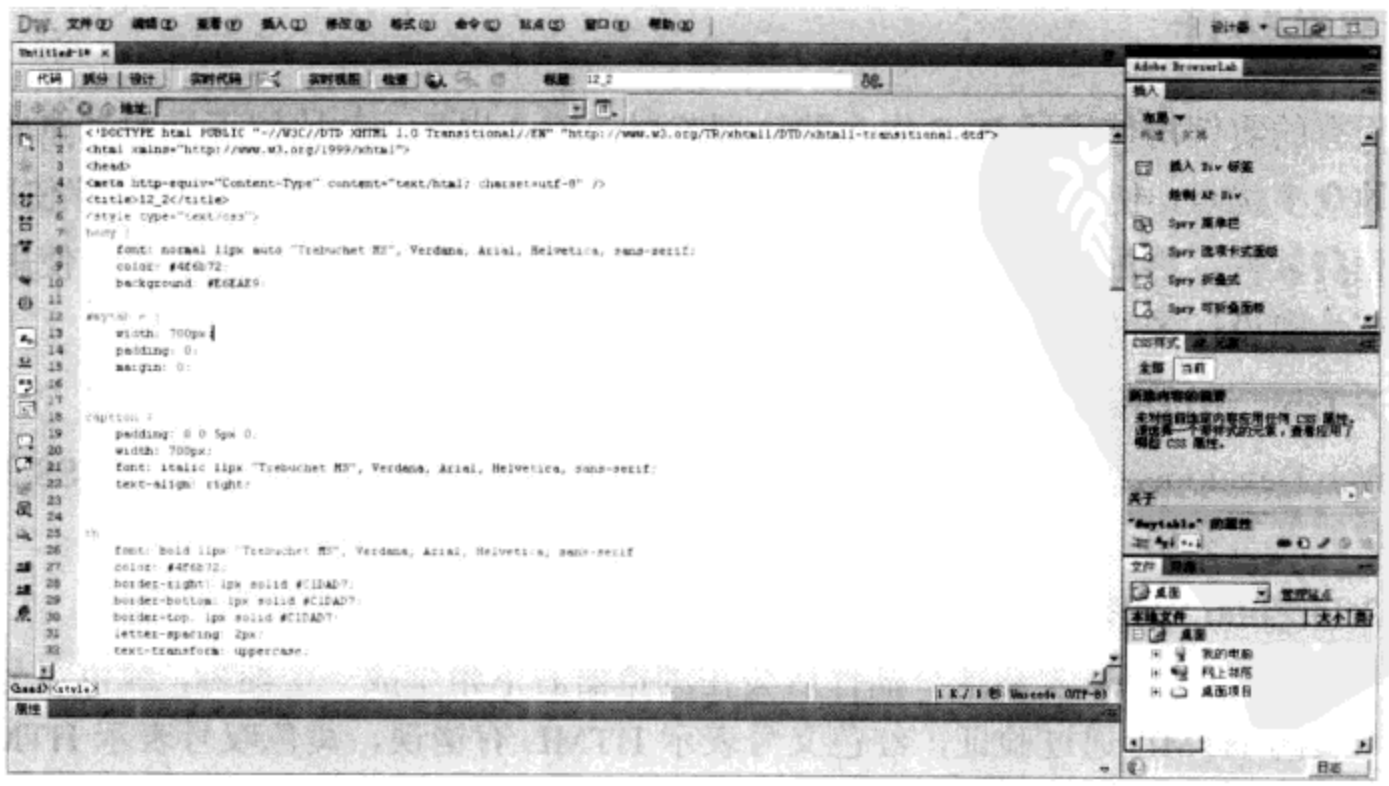


图 2-2　Dreamweaver CS5 中的代码编辑效果

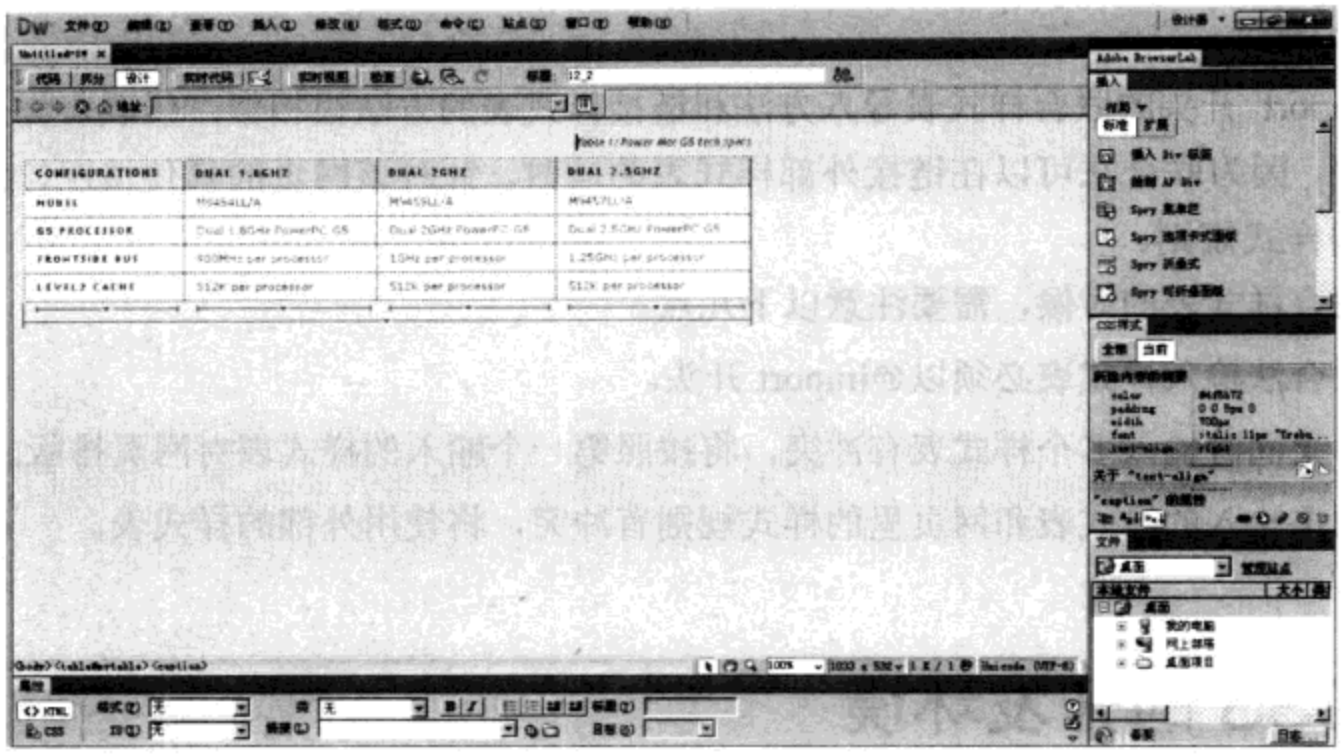


图 2-3 Dreamweaver CS5 中所见即所得效果

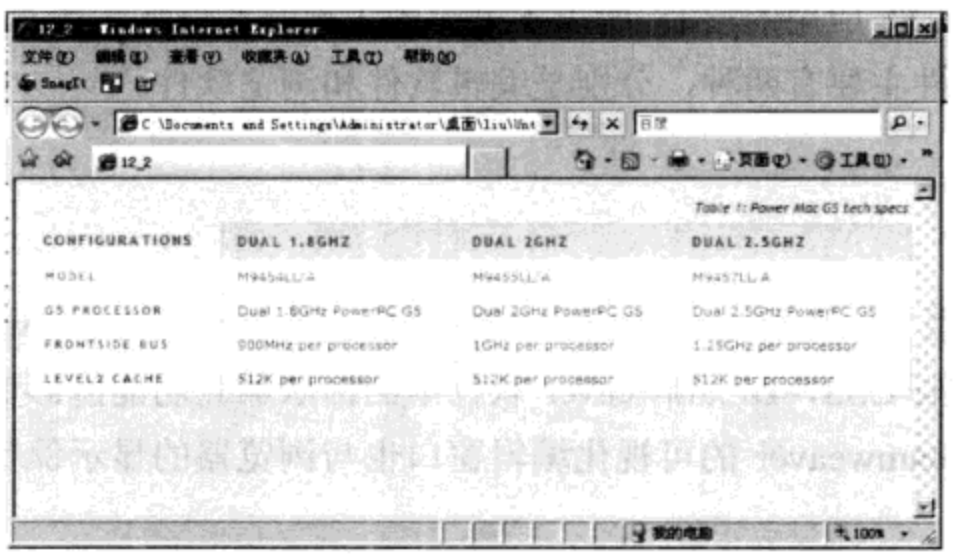


图 2-4 使用浏览器预览效果

2.6.2 浏览软件

本书所有的实例均使用了 IE 7 作为默认浏览器，IE 7 也是一款比较不错的浏览器，继承了微软家族的众多优点，使用方便，但毕竟还不是一款标准的浏览器。浏览软件推荐使用 Firefox。作为一款免费软件，再配上适当的插件，是网页设计者的必备工具，如何选择和使用插件在一定程度上可体现设计者的专业素质，推荐安装如下重要的插件。

- IE Tab：调试网页在 IE 内核浏览器表现的优秀插件。
- Web Developer：直接查看网页 CSS 代码功能，页面信息的显示以及验证 CSS 的 XHTML 功能非常实用。
- ColorZilla：方便提取网页中任何元素的颜色。
- HTML Validator：安装这个插件后会在浏览器右下角生成一个图标，绿色对号表示当前网页的 HTML 通过验证，红色叉号表示 HTML 有错误，黄色叹号表示 HTML 存在无法通过验证的警告语句。双击图标就会高亮显示该网页存在的不能通过验证的语句数

目、位置以及修改建议。

- FireBug：安装这个插件后也会在浏览器右下角生成一个图标，绿色对号表示当前网页的 JavaScript 通过验证，并能对错误的 JavaScript 代码进行调试。需要用到的功能主要是其 Inspector，使用者需要通过定制工具栏，把 Inspector 的“眼镜”图标拖入工具栏。单击 Inspector 图标后，将光标移到网页任意位置，就能在靠下的窗口看到网页 HTML 文件的相应源代码，在网页调试时非常有用。

如图 2-5 所示是一个使用 CSS 布局的网站在 IE 7 中的预览效果。

图 2-5　CSS 布局的网站

2.7　小结

本章主要介绍了 CSS 的基础知识，首先介绍了 CSS 的基本语法；接下来介绍了 CSS 的选择符，包括类型选择符、群组选择符、包含选择符、id 及 class 选择符、标签指定式选择符、组合选择符、伪类和伪对象、通配选择符；然后又介绍了 CSS 的常用属性以及属性值，包括 color 颜色属性值、长度单位、百分比值和 URL 路径；最后介绍了 CSS 在网页中的添加方法，包括在 XHTML 标识符里添加 CSS、在<head></head>里添加 CSS 链接样式表和联合使用样式表。

对本章知识点的前后联系进行归纳总结，结构导图如图 2-6 所示。

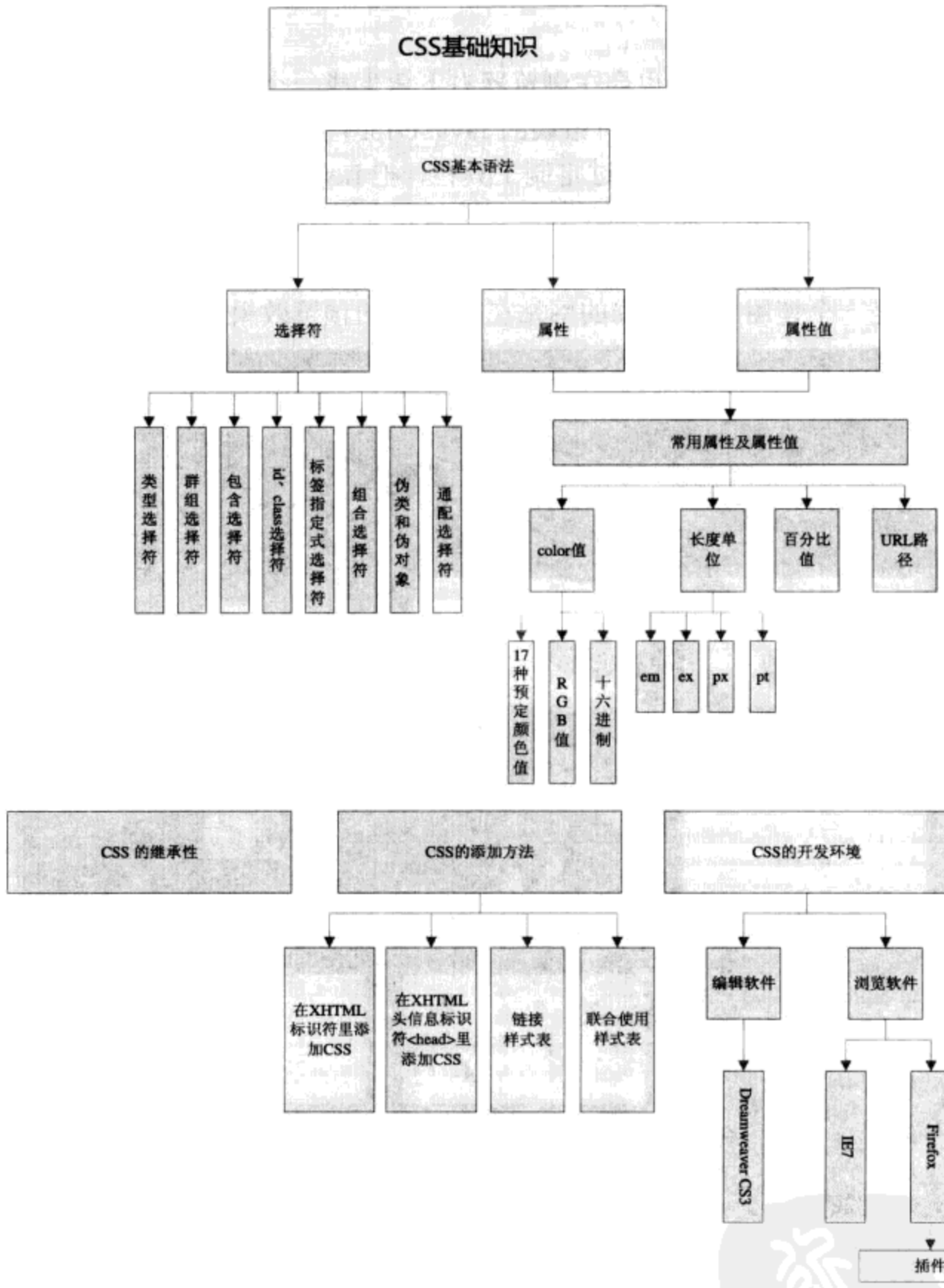


图 2-6 本章知识点结构导图

2.8 习题

1. 一个 CSS 样式的基本语法由哪几部分构成？
2. CSS 的选择符有哪些？
3. 简述 CSS 常用属性及属性值。
4. 如何在 HTML 页面中添加 CSS 编码？
5. 制作 CSS 使用到的软件主要有哪两种？

第3章 XHTML 与 JavaScript 基础

在网页上利用 XHTML 定位文字和图像的效果并不是很好，很难达到精确定位的目的，所以必须使用表格标签和隐式 gif 图像。即使这样也不能保证定位的精确，因为浏览器和操作平台的不同会使显示的结果发生变化。

使用 CSS+DIV 就能够很好地实现想要的所有效果。本章将对如何使用 XHTML 标签和 CSS 样式进行网页布局进行初步介绍。同时，在本章后面也将介绍一些关于 JavaScript 的基础知识，读者能够简单地使用 JavaScript 并且结合 CSS 一起实现某些交互效果。

3.1 XHTML 基础知识

随着网页标准化的普及，使用 XHTML 语言架构网页是大势所趋。学习 XHTML 首先要了解 XHTML 语言构成。本节将介绍 XHTML 的基本结构。

3.1.1 XHTML 的格式文件

初次使用 XHTML 进行网页设计，推荐使用 Dreamweaver 软件直接创建 XHTML 页面。方法是打开 Dreamweaver CS5 软件，单击菜单栏中的“新建”→“HTML”命令，如图 3-1 所示。

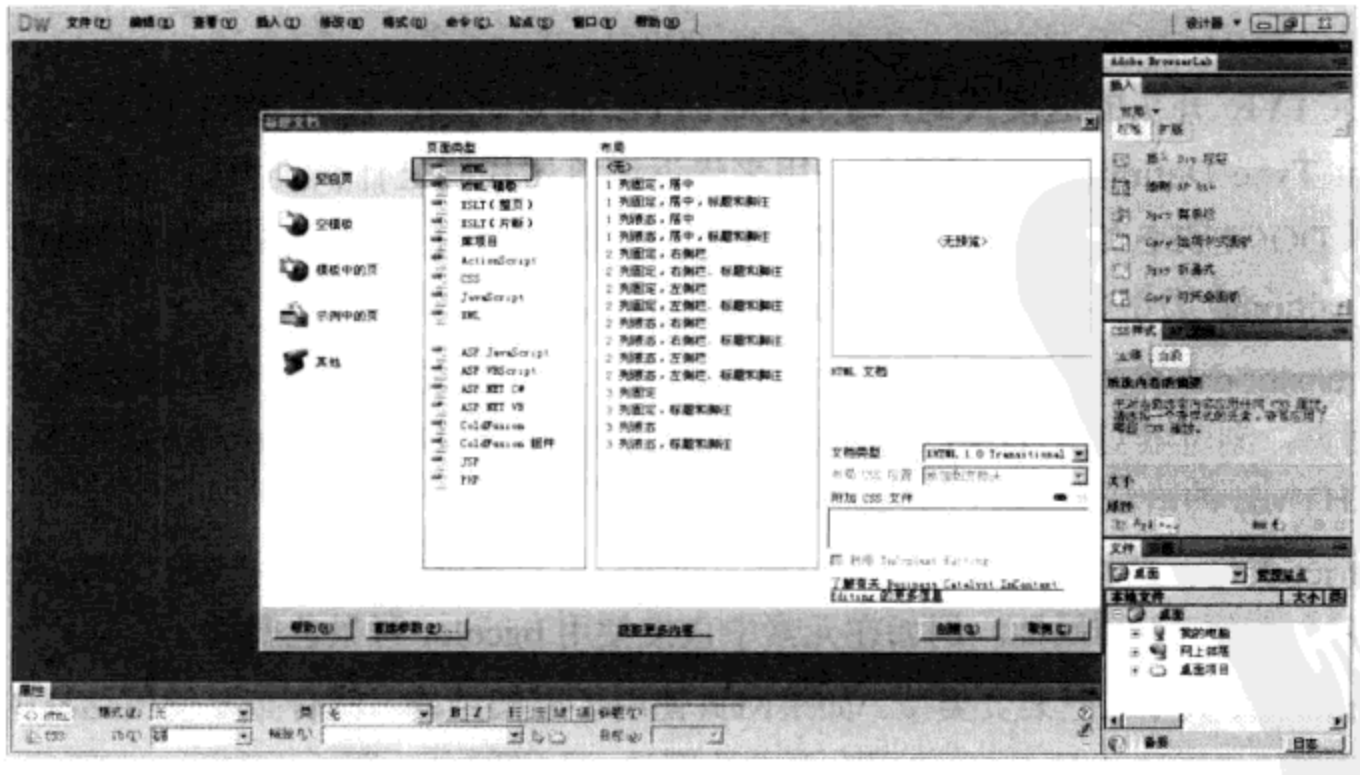


图 3-1 Dreamweaver CS5 开始界面

软件自动生成一套 XHTML 代码，这段代码就是 XHTML 的基本结构，又称格式代码，如图 3-2 所示。

图 3-2 Dreamweaver CS5 新建窗口得到的 XHTML 代码

图 3-2 中的代码便是 XHTML 的格式文件，同时又是 XHTML 的基本结构。下面就来详细讲解这些代码的具体意义。

3.1.2 XHTML 基本结构

XHTML 的基本结构包含文件头和文件体两部分。在文件头中，对 XHTML 文件进行了一些必要的定义；在文件体中，是要显示的各种文档信息的内容。

1. 文档类型定义

对于 3.1.1 节的 XHTML 的格式文件，第一行的代码如下。

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

以 DOCTYE 开头的这段代码，是 DOCTYPE 指定代码，它是 XHTML 的文档类型定义（Ducument Type Definitions，DTD），用来告诉浏览器代码是什么类型。一个标准的 XHTML 文档必须以 DOCTYPE 标签开始。对于 XHTML 而言，可以使用 3 种不同的 XHTML 文档类型，分别是 Transitional 类型、Strict 类型、Frameset 类型，具体说明如下。

- Transitional 类型：XHTML 文档类型的一种，是一种过渡类型。使用过渡类型的 XHTML 网页，浏览器对 XHTML 的解析较宽松，允许使用 HTML 4.01 中的标签，但必须符合 XHTML 的语法。
- Strict 类型：严格类型。使用此类型的网页，浏览器解析将相对严格，不允许使用任何表现样式的标识和属性，例如在元素中直接使用 bgcolor 背景色属性。
- Frameset 类型：框架页类型。如果网页使用了框架结构，就有必要使用这样的文档声明。

下面列举了使用这 3 种 DOCTYPE 方式的网站，如图 3-3、图 3-4 和图 3-5 所示。

图 3-3　cssmix 网站使用了 Transitional 类型标准

图 3-4　catchdigital 网站使用了 Strict 类型标准

图 3-5　k10k.net 网站使用了 Frameset 类型标准

2. XHTML 文件标签

在 3.1.1 节的 XHTML 代码中，第一行的文档类型定义后面便是 XHTML 文件的主体部分，首先是使用一对标签标明 XHTML 代码的起始，代码如下。

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

与最后面的：

```
</html>
```

形成一个包含结构。html 标签是网页的第一个标签，以<html>开始，最后以</html>结束，表示这两个标签之间的内容属于 html 这个类型，浏览器便将其中的内容按 html 进行解析。而 html 只是一个标记，代表一个网页。

3. 文件头和文件体

html 标签内的部分便是 XHTML 文件的主体部分，也是 XHTML 文件的基本组成部分，包含文件头和文件体，代码如下。

```
<head></head>
<body></body>
```

head 指的是 XHTML 的文件头，内容主要放置 IE 浏览器标题栏的名称，或是其他需要给浏览器的信息。body 指的是 XHTML 的文件体，body 中的内容都将被浏览器显示在窗口中。

（1）<head></head>。

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
```

meta 标签告诉浏览器网页的内容是 text/html，并使用的是 utf-8 编码。

```
<title>无标题文档</title>
```

title 标签告诉浏览器的标题栏显示什么文字。

（2）<body></body>。

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

body 标签中的内容是主体，其中可输入文字及其他元素，这些元素都将被浏览器解析并显示在窗口中。XHTML 代码的最终目的就是给浏览器一个标记，告诉浏览器网页中有什么内容，每个内容是什么样的。

3.1.3　XHTML 网页实例

下面是一个最基本的标准 XHTML 文档的实例——使用 XHTML 基本的标签制作一个简单的文字网页，代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">          /*文档类型定义*/
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>                                                          /*文档头*/
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>3_1</title>
    </head>

    <body>
        <h3>欢迎光临我的主页</h3>
        <p>这是我第一次做主页，无论怎么样，我都会努力做好！</p>
    </body>
</html>
```

以上代码在浏览器的预览效果如图 3-6 所示。

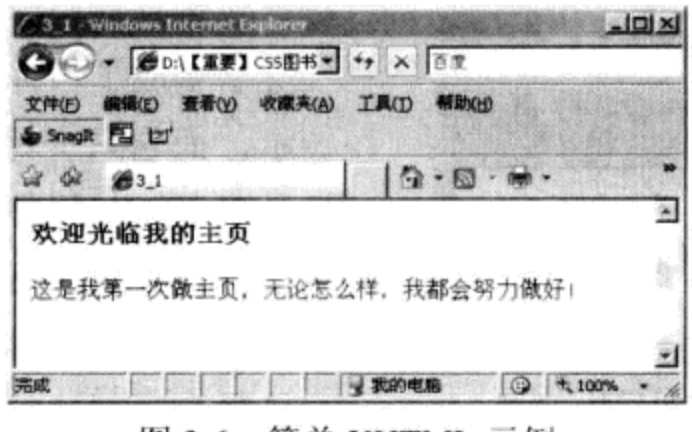


图 3-6　简单 XHTML 示例

3.2　XHTML 的语法构成

XHTML 必须符合 XML 格式，也就是<code></code>格式，它的语法是由标签及其标签属性组成的。本节将分别讲解标签和标签属性的使用方法。

3.2.1　XHTML 中的标签

刚刚接触超文本 XHTML 时，遇到的最大的障碍就是一些用“<”和“>”括起来的句子。这些句子被称为标签，是用来分割和标记文本的元素，以形成文本的布局、文字的格式及五彩缤纷的画面。XHTML 的标签分为单标签、双标签，下面分别进行介绍。

1．单标签

某些标记称为“单标签”，因为它只需单独使用就能完整地表达意思。这类标记的语法代码如下。

```
<标签名称 />
```

最常用的单标签是
，它表示换行。

2．双标签

另一类标记称为“双标签”，它由“始标签”和“尾标签”两部分构成，必须成对使用，其中始标签告诉 Web 浏览器从此处开始执行该标记所表示的功能，而尾标签告诉 Web 浏览器在这里结束该功能，标签前加一个斜杠（/）即成为尾标记。这类标记的语法代码如下。

```
<标签> 内容 </标签>
```

在以上代码中，其中“内容”部分就是要被这对标记施加作用的部分。

例如想突出对某段文字的显示，就将此段文字放在标记中，代码如下。

```
<em>第一：</em>
```

3．标签的作用

继续使用 3.1 节使用的示例，将“这是我第一次做主页，无论怎么样，我都会努力做好！”改为“这是我第一次做主页，无论怎么样，我都会努力做好！”，然后保存修改后再次浏览网页，预览效果如图 3-7 所示。

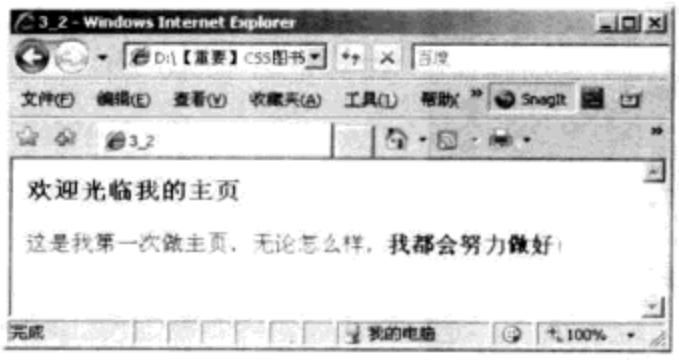


图 3-7　使用标签后的 XHTML 示例

区别很明显，被嵌套在标签中的文字变成了粗体。标签的意思就是粗体显示，而它只会影响到被它包含的内容，这就是 XHTML 标签的作用方式。

把标签和被标签嵌套的内容叫做元素（element）。本示例中“我都会努力做好”这段文字和标签就是网页中的一个元素。

3.2.2　XHTML 的标签属性

许多单标签和双标签的始标记内可以包含一些属性，其语法代码如下。

```
<标签名字 属性 1 属性 2 属性 3 … >
```

各属性之间无先后次序，属性也可省略（即取默认值）。例如单标签<hr>表示在文档当前位置画一条水平线（horizontal line），一般是从窗口中当前行的最左端一直画到最右端。<hr>标签附带一些属性的代码如下。

```
<hr size=3 align=left>
```

在以上代码中，其中 size 属性定义线的粗细，属性值取整数，缺省值为 1；align 属性表示对齐方式，可取 left（左对齐，缺省值）、center（居中）、right（右对齐）。

注意：

普通的 XHTML 文件有两个等级标准（不算框架标准）——过渡标准和严格标准，其中过渡标准主要针对那些习惯于使用 XHTML 开发网站的站长。上面的代码在过渡标准中是合法的，可是在严格标准中，size、align 等属性将被视为非法属性。XHTML 不仅是更加标准、更加严格的 HTML，它还推崇一种标准的构建网站的思路，那就是把网页的内容与表现分开，这在 XHTML 中是通过 CSS 来实现的。

3.3　XHTML 的语法规范

在使用 XHTML 语言进行网页制作时，必须要遵循一定的语法规范。下面进行详细讲解，具体内容可以分为以下几点。

3.3.1　标签不能重叠，可以嵌套

在 HTML 里，一些元素不正确嵌套也能正常显示，例如下面的代码。

```
<b><i>This text is bold and italic</b></i>
```

而在 XHTML 里必须正确嵌套之后才能正常使用，例如下面的代码。

```
<b><i>This text is bold and italic</i></b>
```

注意：

这个错误通常发生在嵌套多层之后的标签里面。

例如下面的错误代码。

```
<ul>
<li>Coffee</li>
<li>Tea
<ul>
```

```
<li>Black tea</li>
<li>Green tea</li>
</ul>
<li>Milk</li>
</ul>
```

正确的代码如下。

```
<ul>
<li>Coffee</li>
<li>Tea
<ul>
<li>Black tea</li>
<li>Green tea</li>
</ul>
</li>
<li>Milk</li>
</ul>
```

观察上述两段代码可以看到正确的代码中，第一个
之后插入了
标签。

3.3.2　XHTML 文件一定要有正确的组织格式

所有的 XHTML 应该正确地被嵌套在以<html>开始、以</html>结束的元素里面，其他的元素可以有子元素，并且子元素也要被正确地嵌套在它们的父元素内，例如下面的代码。

```
<html>
    <head> ... </head>
    <body> ... </body>
</html>
```

3.3.3　标签名字一定要用小写字母

因为 XHTML 文档是 XML 应用程序，XML 对大小写是敏感的。例如
和
是两个不同的标记，例如下面的错误代码格式。

```
<BODY>
<P>This is a paragraph</P>
</BODY>< /p>
```

正确的代码格式如下。

```
<body>
    <p>This is a paragraph</p>
</body>
```

3.3.4　所有的 XHTML 元素一定要关闭

没有关闭的空的元素不能存在于代码中，其实对于这点是比较容易做到的，有开始就应该有结束，例如下面的错误代码。

```
<p>This is a paragraph
<p>This is another paragraph
```

正确的代码如下。

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

3.3.5　独立的一个标签也要用 />来结束

XHTML 有时候会出现独立的一个标签，这样的单标签在使用的时候也要在结束的位置使用“ />”，例如下面的错误代码。

```
This is a break<br>
Here comes a horizontal rule:<hr>
Here's an image <img src="happy.gif" alt="Happy face">
```

正确的代码如下。

```
This is a break<br />
Here comes a horizontal rule:<hr />
Here's an image <img src="happy.gif" alt="Happy face" />
```

通过上面的几个例子基本上可以看出 HTML 和 XHTML 之间的不同，那么应该从现在起就规范我们编写的 HTML，例如都使用小写的标记，在标记之后加上结束标记的符号“ />”。

3.4　div 标签

在进行符合 Web 标准的网页设计时，使用到的 XHTML 的一个重要标签就是 div。下面对这一个标签进行详细讲解。

3.4.1　什么是 div

W3C 官方对 div 的定义是：div（division）是一个块级元素， 可以包含段落、标题、表格甚至任何元素。这使得 div 便于建立不同集成的类，例如章节、摘要和批注等，是 CSS 的好帮手。

div 是一个容器。XHTML 页面中的每一个标签对象几乎都可以称得上是一个容器，被 嵌套起来的文字、图片等任何东西，浏览器都将视为一个物件。就如同我们在打包

物品一样，你可以一个包裹只包一件衣服，也可以一个包裹包衣服、鞋子、帽子等很多东西。div 这个标签容器就是这样的功能，可以用 包含一段文字，这段文字就成了一个物件；也可以用 包含一段文字、一个图片、一个表格，那么这 3 样东西也会被视为一个物件，使用方法如下。

```
<div>内容</div>
```

3.4.2 理解 div

div 在网页中将呈现出什么样的效果？这样一个块级元素的标签，又是怎么样工作的？本节将回答这些问题，以便读者深入理解 div 的特性。

1. div 本身没有样式

在一个没有任何 CSS 应用的页面中，应用了一个 div 标签，这个时候没有任何实际的效果，就如同直接写上了 div 中的内容一样，代码如下。

```
<div>这是一段 div 包含的文本内容，可以看到，没有任何 div 的效果出现。</div>
```

以上代码在浏览器中的预览效果如图 3-8 所示。

2. div 的块状属性

在设计表格时候，使用 XHTML 代码中的 table 表格设计的左右分栏或上下分栏，都能够在浏览器预览中直接看到分栏的效果，如图 3-9 所示。

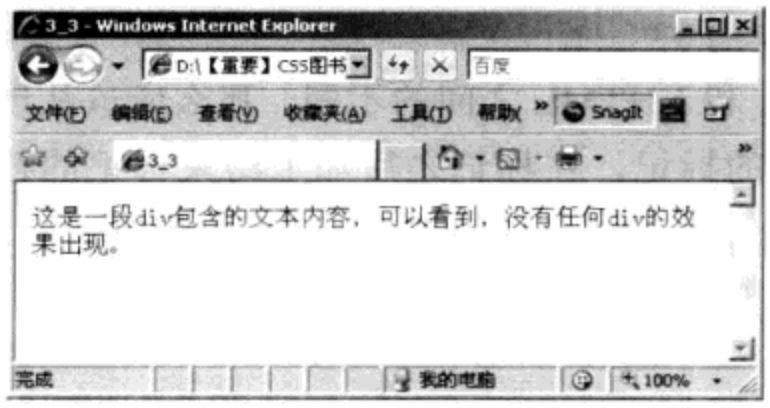


图 3-8 div 包含的文本

图 3-9 表格布局分栏

由于 table 表格自身的代码形式决定了在浏览器中显示的时候，两块内容分别显示在左单元格与右单元格之中。因此不管是否应用了表格线，内容都将存在于两个单元格之中，也达到了分栏的效果。

因为 div 是块状对象，所以在使用 div 布局时完全不同。现在尝试编写两个 div，用于左分栏与右分栏，代码如下。

```
<div>左分栏</div>
<div>右分栏</div>
```

以上代码在浏览器中的预览效果如图 3-10 所示，能够看到仅仅出现的两行文字，并没有看出 div 的任何特征。

从如图 3-10 所示的预览结果，能得出如下两个结论。

图 3-10　div 布局分栏

（1）“左分栏”和“右分栏”这两段文字不是并排放置，而是上下放置。这说明了 div 本身是占据整行的一种对象，不允许其他对象与它在一行并列显示。W3C 官方定义 div 是一个 block 块状对象。XHTML 中所有的对象几乎都默认为如下两种类型。

- block 块状对象：块状对象指的是当前对象显示为一个方块，默认的显示状态下，将占据整行，其他对象在下一行显示。
- in-line 行间对象（内联对象）：in-line 对象恰恰与 block 对象相反，它允许下一个对象与它本身在一行中进行显示。

块状的 div 也说明：div 在页面中并非用于类似于文本一样行间排版，而是用于大面积、大区域的块状排版。

（2）从页面预览的效果可发现，网页之中，除了文字之外没有任何其他效果，两个 div 之间的关系，只是前后关系，并没有出现类似表格的边框，因此可以说，div 本身与样式没有任何关系，样式需要编写在 CSS 中才能实现。所以说，div 对象从本质上实现了结构与样式的分离。

3.4.3　使用 div

与其他 XHTML 对象一样，使用 div 只需应用 这样的标签形式，将内容放置在其中，便可以应用 div 标签。div 对象除了可以直接放入文本，还可以放入其他标签，也可以多个 div 标签进行嵌套使用，最终目的是合理地标识出想要的页面区域。

div 对象在使用时，可以加入其他属性，如 id、class、align、style 等，但是为了实现内容与表现分离，一般不推荐将 align、style 这样的属性写在 div 标签里。因此，div 最终的代码只可能是以下两种形式。

```
<div id="id 名称">[…]</div>
<div class="clash 名称">[…]</class>
```

使用 id 属性，可以将当前这个 div 指定一个 id 名称，在 CSS 中使用 id 选择符进行样式编写。同样可以使用 class 属性，在 CSS 中使用 class 选择符进行样式编写。

注意：

同一个名称的 id 值在当前 XHTML 页面中，只允许使用一次，不管是应用到 div 还是其他对象的 id 中，而 class 名称则可以重复使用。

3.4.4　div 的并列与嵌套

div 可以进行多层嵌套使用。嵌套是为了实现更多复杂的页面排版，例如当设计一个网页时，首先需要有整体布局，需要产生头部、中部和底部，这也许会产生一个较为复杂的 div 结构，代码如下。

```
<div id="header">头部</div>
<div id="center">
    <div id="left">左分栏</div>
    <div id="right">右分栏</div>
</div>
<div id="footer">底部</div>
```

在代码中为每个 div 定义了一个 id 属性。可以看到，id 分别为 header、center 和 footer 的 3 个 div 对象，它们之间属于并列关系，网页的布局结构之中，以垂直方向的布局为例，如图 3-11 所示。

而在 center 之中，为了内容需要，有可能在 center 中使用一个左右分栏的布局，因此在 center 之中拥有两个 id，分别为 left 和 right 的 div。这两个 div 本身是并列关系，而它们都处于 center 之中，因此它们与 center 形成了一种嵌套关系。如果它们两个被样式控制为左右显示的话，那么它们最终的布局关系如图 3-12 所示。

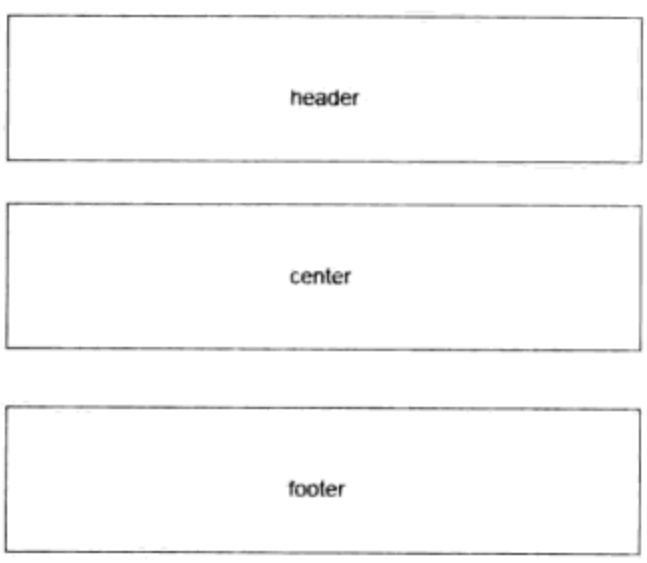


图 3-11　div 布局关系

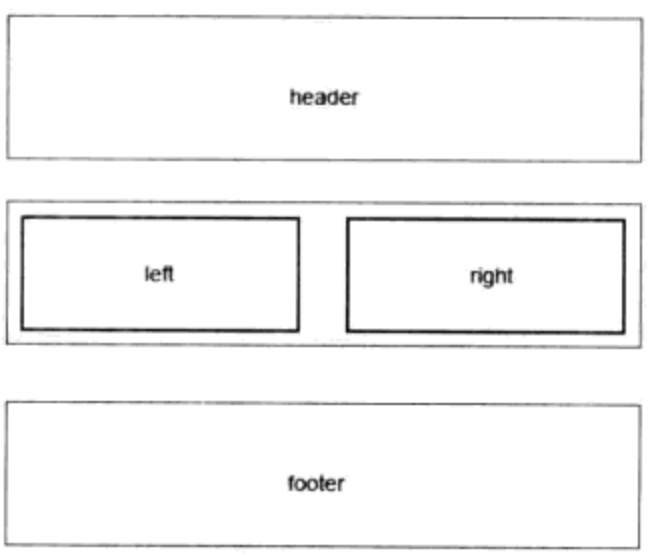


图 3-12　div 最终布局关系

在进行网页布局时，使用这些嵌套的 div 来构成，无论是多么复杂的布局方法，都可以使用 div 之间的并列和嵌套来实现。

注意：

实际操作中，应当尽量减少嵌套，以保证浏览器不用过分消耗资源来对嵌套关系进行解析，这样可以加速页面下载。

3.5　span 标签

span 标签也是一个常用的标签，这个标签与 div 很像，没有什么特定的意义，可以把它看

作是一个袋子，它不像 div 箱子那样可以有自己的宽高，它的宽高只能随着内容的多少而定。实际上，span 与 CSS 结合起来后，其应用十分广泛。本节将对 span 标签做详细讲解。

3.5.1　什么是 span

span 标签也是 XHTML 中的一个对象，是行内元素。它在行内定义一个区域，也就是说一行内可以被 span 划分成好几个区域，从而实现某种特定效果。span 标签也是成对出现的，即，例如下面的代码。

```
<span>这是 span 标签</span>
```

3.5.2　span 与 div 的区别

span 和 div 的区别在于，div 是一个块级元素，可以包含段落、标题、表格，乃至诸如章节、摘要和备注等。而 span 是行内元素，span 的前后是不会换行的，它没有结构的意义，纯粹是应用样式，当其他行内元素都不合适时，可以使用 span。

可通俗地将 理解为大容器，大容器当然可以放一个小容器了，就是小容器。

1．呈现效果的区别

下面以一个实例来说明这两个属性的区别，代码如下。

```
<div id="content">
    <p>This is <span class="bold">crazy</span></p>
</div>
```

span 标记有一个重要而实用的特性，即它什么事也不会做，它的唯一目的就是围绕 XHTML 代码中的其他元素，为这些元素赋予样式。在此示例中，标识符允许将一个段落分成不同的部分。

div 也被用来在 XHTML 文件中建立逻辑部分。但与 span 不同，其工作于文本块一级，div 在它所包含的 XHTML 元素的前面及后面都引入了行分隔，也就是将 div 定义为“块级元素”的原因，实例代码如下。

```
<!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>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>span 与 div 的区别</title>
    </head>
    <body>
        <span>span 元素 1</span><span>span 元素 2</span>
        <div>div 元素 1</div><div>div 元素 2</div>
    </body>
</html>
```

图 3-13　span 和 div 的区别

以上代码在浏览器中的预览效果如图 3-13 所示。

2. 产生区别的根本原因

div 与 span 元素在显示上的不同，源于其默认显示模式（display）的不同。正是由于两个对象不同的显示模式，决定了两个标签的不同用途。

div 对象的任务是呈现一个块状内容，如一大段文本、一个导航区域、一个页脚区域等，为其中的内容进行结构编码并进行样式设计。而作为行内对象的 span 标签，则是对行内元素进行结构编码以方便样式设计。span 默认状态下不会破坏行中元素的显示顺序，例如在一大段文本中，需要改变其中一段文本的颜色，可以对这一小部分文本使用 span 对象，并进行样式设计，这将不会改变这个整段文本的显示方式。

div 所赋予的使命要比 span 重要许多，默认为块状显示模式的 div 对象在实际应用中担负着页面大块布局及版式的所有工作，设计人员可大量使用 div 进行组合和嵌套来实现网页各种版式布局。

3.6　h1 至 h6 标签

在 XHTML 编写过程中，定义标题可以使用从<h1>到<h6>这几个标签，它们对应的终止标签分别为</h1>到</h6>。

<h1>到<h6>随着字号顺序减小，代表内容的重要性也逐渐降低。通常浏览器将在标题的上面和下面自动各空出一行距离，代码如下。

```
<h1>这里是 H1 标题</h1>
<h2>这里是 H2 标题</h2>
<h3>这里是 H3 标题</h3>
<h4>这里是 H4 标题</h4>
<h5>这里是 H5 标题</h5>
<h6>这里是 H6 标题</h6>
```

预览效果如图 3-14 所示。

图 3-14　标题标签的效果

3.7　列表制作标签 ul、ol、li

列表元素是网页设计中使用频率非常高的元素，在很多商业网站设计中，无论是新闻列表，还是产品或者其他内容，均要使用列表元素来体现。如图 3-15 所示为美国 CNN 网站的新闻列表页面。

列表形式在网站设计中占有很大比重，信息的显示非常整齐、直观，便于用户理解与点击。从出现网页开始到现在，列表元素一直是网页中非常重要的应用形式。

updated 22 hours, 32 minutes ago
Time.com: **Could Mugabe Lose His Election?**
Even if Zimbabwe's leader wins in this weekend's polling, a revolt within his party may be laying the seeds for his downfall

updated 1 day ago
This week on Inside Africa...
This weekend, Inside Africa goes live as Zimbabwean voters go to the polls and President Robert Mugabe faces his strongest challenge yet. Will the voters end his 28-year reign or will he hang on to power? We'll look at what's at stake for Zimbabweans, both inside the country and living in exile.
Video

updated Mon March 24, 2008
Mugabe accused of election-rigging plan
The discovery of millions of extra ballots proves that President Robert Mugabe intends to rig next week's elections in his favor, Zimbabwe's main opposition party said Sunday.
Video

updated Wed March 19, 2008
Rights groups: Election corruption rife in Zimbabwe
Serious electoral flaws and widespread political intimidation have made a mockery of Zimbabwe, human rights groups said Wednesday.

图 3-15　cnn.com 网站的新闻列表

3.7.1　ul 无序列表

无序列表就是指列表中的各个元素在逻辑上没有先后顺序的列表形式。如果不需要描述列表中的每一条必须从 1、2、3 或 A、B、C 的形式去递增，那么就可以使用 ul 元素，大部分页面中的信息均可以用 ul 来描述。ul 与 li 元素配合使用，其中的每一个 li 标签均为一条列表。

一段标准的无序列表的 XHTML 代码如下。

```
<ul>
    <li>这里是无序列表的列表项值 1</li>
    <li>这里是无序列表的列表项值 2</li>
    <li>这里是无序列表的列表项值 3</li>
    <li>…</li>
</ul>
```

无序列表的 XHTML 代码标签
表示将打开一个无序列表，包含于其中的
则表示具体的列表项。在无序列表中有 3 种类型的项目符号，如图 3-16 所示。

• disc_1	◦ circle_1	▪ square_1
• disc_2	◦ circle_2	▪ square_2
• disc_3	◦ circle_3	▪ square_3
• …	◦ …	▪ …

图 3-16　ul 无序列表样式

3.7.2　ol 有序列表

有序列表表示列表中的各个元素有序列区分，从上至下可以有序列编号，如 1、2、3 或 a、b、c 等。现在网络上符合 Web 标准的网站，使用 ol 标签制作有序列表的比比皆是，它的好处就是简化了列表在代码结构上的复杂性。它可以用在作息时间表、工作进度表、书籍大纲目录等实际应用中。下面是使用 ol 有序列表制作的新闻排行榜实例，实例的 XHTML 代码如下。

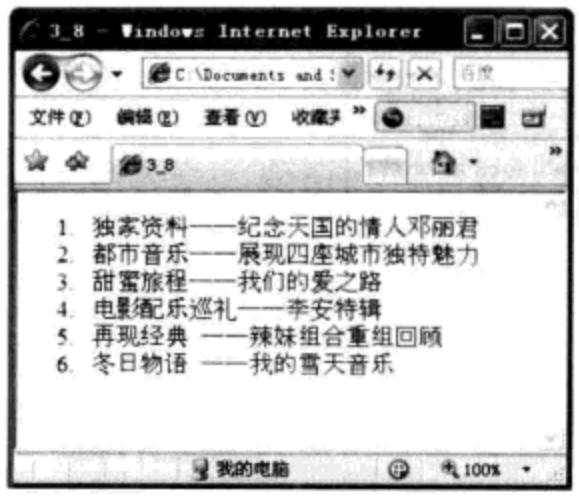


图 3-17　ol 有序列表

```
<ol>
    <li>独家资料——纪念天国的情人邓丽君</li>
    <li>都市音乐——展现四座城市独特魅力</li>
    <li>甜蜜旅程——我们的爱之路</li>
    <li>电影配乐巡礼——李安特辑</li>
    <li>再现经典——辣妹组合重组回顾</li>
    <li>冬日物语——我的雪天音乐</li>
</ol>
```

以上代码在浏览器下的预览效果如图 3-17 所示。

3.8　p 标签和 br 标签

在 XHTML 编写过程中，经常会用到段落文本的排版，这个时候就会用到段落标签 p 和换行标签 br。本节介绍 p 标签和 br 标签的使用方法。

3.8.1　p 标签

p 标签是一个有特定语义的标签，表示段落，是用来区分段落的，代码如下。

```
<p></p>
```

p 标签是双标签，在大部分的浏览器中对 p 标签基本上都要设定上下的边距，但是没有行首缩进，因为行首缩进只是表示段落的方式，不是一定或是必需的。所以在使用 p 标签的时候，如果需要可以针对 p 设定一下行首缩进，下面给出一个实例，代码如下。

```
<p>CSS 具有两面性。就像它在格式化文本、导航栏、图片以及其他小小的网页方面很棒一样，当你准备好布局完整的网页时，它真正可怕的一面也展现出来了。</p>那一面究竟是什么呢？
```

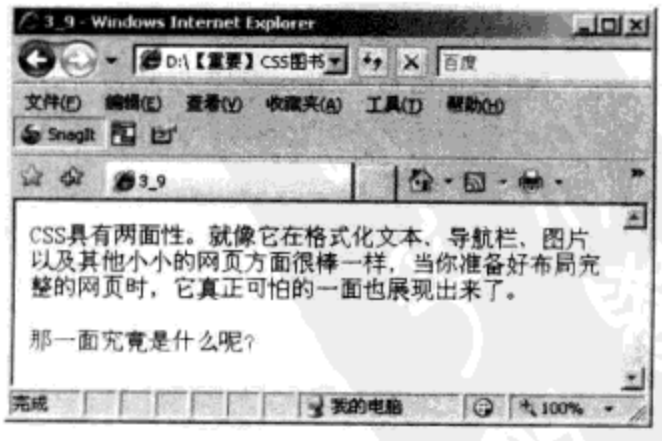


图 3-18　带有 p 标签的文本段落

此示例的效果是一段文本段落，使用 p 标签分隔段落，预览效果如图 3-18 所示。

3.8.2　br 标签

如果希望另起一行书写文字却又不希望另起一个自然段的时候，就可以应用 br 标签了。br 标签的正确格式代码如下。

```
<br />
```

br 标签也是一个单标签，需要加上一个“/”来关闭，以符合 XHTML 的要求，例如下面就是一段使用 br 标签的 XHTML 代码。

```
CSS 具有两面性。<br />就像它在格式化文本、导航栏、图片以及其他小的网页方面很棒一样，当你准备好布局完整的网页时，它真正可怕的一面也展现出来了。<br />那一面究竟是什么呢？
```

这段文字每一个句子结尾处使用了一个
标签，用来换行显示，预览效果如图 3-19 所示。

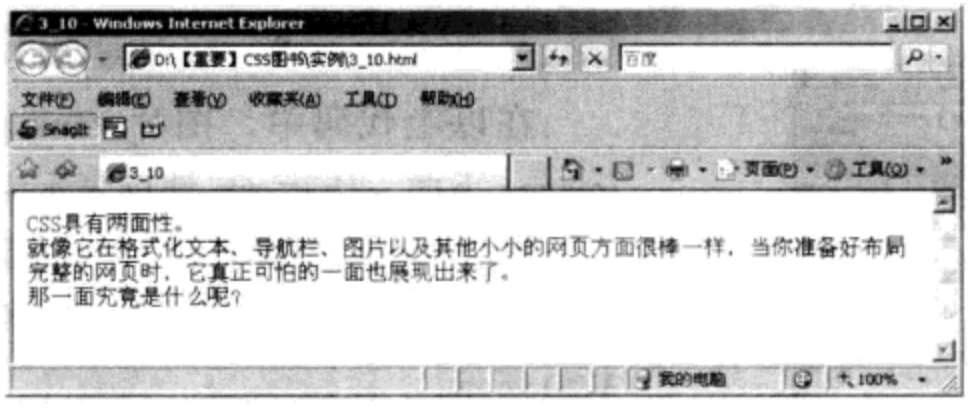


图 3-19　使用 br 标签换行

3.9　img 标签

img 标签是 XHTML 标记语言中很重要的一个标签，很多图片都需要使用 img 标签来显现在网页上。img 标签是单标签，一般书写格式代码如下。

```
<img src="url" />
```

从技术层面讲，img 标签并不会在网页中插入图像，而是从网页上链接图像，<img>标签创建的是被引用图像的占位空间。

3.9.1　img 标签的属性

img 标签的常用属性有 alt 和 src 属性，使用格式代码如下。

```
<img src="" alt="" />
```

在以上代码中，使用到的两个属性代表的意义如下。

- src：告诉浏览器图片的具体位置，可以用 URL 表示，URL 表示方法又分为相对路径和绝对路径。绝对路径指向其他站点（例如 src="http://www.example.com/"）；相对路径指向站点内的文件（例如 src="/i/image.gif"）。

注意：

统一资源定位符 URL 是用于完整地描述 Internet 上网页和其他资源的地址的一种标识方法。Internet 上的每一个网页都具有一个唯一的名称标识，通常称之为 URL 地址，这种地址可以是本地磁盘，也可以是局域网上的某一台计算机，更多的是 Internet 上的站点。简单地说，URL 就是 Web 地址，俗称“网址”。

- alt：图片替代文字，有些浏览者不想看到图片（例如由于网速太慢），有些早期的浏览器也不支持图片，或者是图片的具体位置错了，这些情况浏览者是看不到图片的，这时 alt 属性可以在图片的位置上显示出代替的文字。

3.9.2　img 标签使用方法

在一个网页中，当添加图片的时候，可以使用 img 标签，具体代码如下。

```
<img src="images/temp.jpg" alt="CSS 图片" />
```

图 3-20　img 标签的图片

在以上代码中，图片 temp.jpg 是放在了文件夹 images 下面，所示 src 地址表示为 images/temp.jpg，图片注释文字是“CSS 图片”，最后使用结束符号“/”来关闭 img 标签。当光标悬停在图片上时，预览效果如图 3-20 所示。

可以看到，当光标悬停在图片上方时，就会出现一个图示文字“CSS 图片”，即使图片显示不出来，这个提示文字仍然会出现。

3.10　表单标签

表单是用户提交信息的重要渠道。XHTML 表单标签包括 form 标签、input 标签、fieldset 标签、legend 标签等。本节将介绍这些常用表单标签的使用方法。

3.10.1　form 标签

<form>表单标签成对出现，以<form>开始，以</form>结束。用户注册网站会员、投票等都需要使用<form>表单标签来实现。

<form>标签的结构代码如下。

```
<form></form>
```

3.10.2　input 标签

<input>标签是单行输入文本域，是以单标签出现的。下面分别介绍 input 标签的属性以及属性值的使用方法。

1．input 标签的属性

<input>标签的结构代码如下。

```
<input type="" maxlength="" src="" alt="" size="" value="" accesskey="" checked="" disabled="" />
```

在以上代码中，使用到的属性代表的意义如下。

- type：代表一个输入域的显示方式（分为输入型、选择型、点击型）。
- maxlength：输入域最多可以输入文字的长度。
- src：当使用图片来表示按钮时，代表图片的位置（URL）。
- alt：代表表单的提示文字（当光标停留时）。
- size：输入域的长度。
- value：输入域的值。
- accesskey：表单的快捷键访问方式。
- checked：如果是选择型的输入域，代表已经被选择。
- disabled：输入域无法获得焦点，无法选择，以灰色显示，在表单中不起任何作用。

下面详细讲解以上属性的使用方法。

2．input 标签属性的使用方法

在 input 标签的众多属性里面，type 属性显得尤为重要，它是 input 标签功能多样化的体现。type 属性的可选值如下。

- text：文字输入域（输入型）。
- password：也是文字输入域，但是输入的文字以密码符号“*”显示（输入型）。
- file：可以输入一个文件路径（输入型）。
- checkbox：复选按钮，可以选择零个或多个（选择型）。
- radio：单选按钮，只可以选择一个而且必须选择一个（选择型）。
- hidden：代表隐藏域，可以传送一些隐藏的信息到服务器。
- button：按钮（点击型）。
- image：使用图片来显示按钮，使用 src 属性指定图像的位置（点击型）。
- submit：提交按钮，表单填写完毕可以提交，把信息传送到服务器。可以使用 value 属性来显示按钮上的文字（点击型）。
- reset：重置按钮，可以把表单中的信息清空（点击型）。

下面使用这些属性制作具有各种功能的输入文本框。

3. input 标签常用实例

在下面制作实例的过程中都使用到了 form 标签，form 标签是制作表单的重要标签。

（1）文本输入框，结构代码如下。

```
<form>
    姓：<input type="text" /><br />
    名：<input type="text" />
</form>
```

以上代码制作了一个输入姓名的文本输入框，预览效果如图 3-21 所示。

（2）密码输入域，结构代码如下。

```
<form>
    用户：<input type="text"><br />
    密码：<input type="password">
</form>
<p>
请注意，当您在密码域中键入字符时，浏览器将使用项目符号来代替这些字符。
```

以上代码制作了一个输入用户名和密码的文本输入框，其中密码输入域会显示为“*”符号，预览效果如图 3-22 所示。

图 3-21　文本输入框实例

图 3-22　密码输入域实例

（3）单选按钮，结构代码如下。

```
<form>
    男性：<input type="radio" checked="checked" value="male">
    <br />
    女性：<input type="radio"value="female">
</form>
<p>当用户点击一个单选按钮时，该按钮会变为选中状态，其他所有按钮会变为非选中状态。</p>
```

以上代码制作了选择性别的单选按钮，预览效果如图 3-23 所示。

（4）复选框实例，结构代码如下。

```
<form>
    我喜欢自行车：<input type="checkbox"/><br />
    我喜欢汽车：<input type="checkbox"/>
</form>
```

以上代码制作了调查个人爱好的具有复选功能的实例，预览效果如图 3-24 所示。

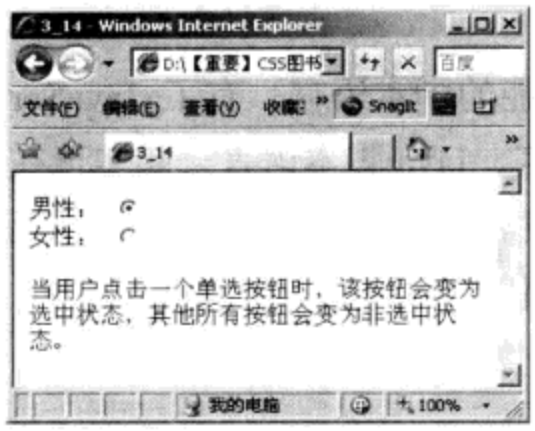


图 3-23　单选按钮实例

图 3-24　复选框实例

3.10.3　分组标签 fieldset、legend

如果一个页面的表单项太多，最好把它们分组显示，就像使用 p 标签分开段落一样，可以使用 fieldset 与 legend 标签对表单内容分组。fieldset 标签是用来对表单进行分组，是成对出现的，以<fieldset>开始，以</fieldset>结束。一个表单可以有多个<fieldset>，每对<fieldset>为一组，每组内容的描述可以使用 legend 标签说明。

legend 标签是用来对表单的每组内容进行说明，成对出现，以<legend>开始，以</legend>结束。

下面使用 fieldset 和 legend 标签来制作一个分组表单实例，结构代码如下。

```
<fieldset>
    <legend>健康信息：</legend>
    <form>
        身高：<input type="text" />
        体重：<input type="text" />
    </form>
</fieldset>
```

以上代码的预览效果如图 3-25 所示。

图 3-25　分组表单的实例

3.11　JavaScript 简介

JavaScript 语言的前身叫做 Livescript。自从 Sun 公司推出著名的 Java 语言之后，Netscape 公司引进了 Sun 公司有关 Java 的程序概念，将自己原有的 Livescript 进行重新设计，并改名为

JavaScript。

JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本语言，有了 JavaScript，可使网页变得生动。使用它的目的是与 HTML 超文本标记语言、Java 脚本语言一起实现在一个网页中链接多个对象、与网络客户交互的作用，从而可以开发客户端的应用程序。它是通过在标准的 HTML 语言中嵌入或调入实现的。JavaScript 具有以下优点。

1. 简单性

JavaScript 是一种脚本编写语言，它采用小程序段的方式实现编程，像其他脚本语言一样，JavaScript 同样是一种解释性语言，它提供了一个简易的开发过程。它的基本结构形式与 C、C++、VB、Delphi 十分类似。但它不像这些语言需要先编译，而是在程序运行过程中被逐行地解释。它与 HTML 标识结合在一起，从而方便用户的使用。

2. 动态性

JavaScript 是动态的，它可以直接对用户的输入做出响应，无须经过 Web 服务程序。它对用户的响应是采用以事件驱动的方式进行的。所谓事件驱动，就是指在主页中执行了某种操作所产生的动作，就称为“事件”，例如按下光标、移动窗口、选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

3. 跨平台性

JavaScript 是依赖于浏览器本身的，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可以正确执行。

4. 节省 CGI 的交互时间

随着 WWW 的迅速发展，也许 WWW 服务器提供的服务要与浏览者进行交流，确认浏览者的身份、需要的服务等，这项工作通常由 CGI/PERL 编写相应的接口程序与用户进行交互来完成。很显然，通过网络与用户的交互过程一方面增大了网络的通信量，另一方面影响了服务器的服务性能。服务器为一个用户运行一个 CGI 时，需要一个进程为它服务，它要占用服务器的资源（例如 CPU 服务、内存耗费等），如果用户填表出现错误，交互服务占用的时间就会相应增加。被访问的热点主机与用户交互越多，服务器的性能影响就越大。

JavaScript 是一种基于客户端浏览器的语言，用户在浏览网页中填表、验证的交互过程只是通过浏览器对调入 HTML 文档中的 JavaScript 源代码进行解释执行来完成的，即使是必须调用 CGI 的部分，浏览器只将用户输入验证后的信息提交给远程的服务器，大大减轻了服务器的负担。

3.12 JavaScript 语言基础

JavaScript 是一种脚本语言，也是一种解释性语言，它被直接嵌入在 HTML 页面来添加交互行为。下面就来学习 JavaScript 脚本语言的基础知识。

3.12.1　插入 JavaScript

JavaScript 可以出现在 XHTML 的任意地方，使用标记如下。

```
<script>…</script>
```

可以在 XHTML 文档的任意地方插入 JavaScript，甚至在<html>之前插入也不成问题。不过如果要在声明框架的网页（框架网页）中插入，就一定要在<frameset>之前插入，否则不会运行。

插入 JavaScript 的基本格式如下。

```
<script type="text/javascript">
    <!--
    ...
     (JavaScript 代码)
    ...
    //-->
</script>
```

第二行和第六行的作用是，让不懂<script>标记的浏览器忽略 JavaScript 代码。第六行前边的双反斜杠“//”是 JavaScript 里的注释标号，以后将会学到。

另外一种插入 JavaScript 的方法是，把 JavaScript 代码写到另一个文件当中（此文件通常应该用“.js”作扩展名），然后使用如下代码，把它嵌入到文档中。

```
<script type="text/javascript" src="javascript.js"></script>
```

注意：

一定要用“</script>”标记。

3.12.2　基本语法

每一句 JavaScript 都有类似于以下的格式。

```
<语句>;
```

其中分号“;”是 JavaScript 语言作为一个语句结束的标识符。语句块是用大括号“{ }”括起来的一个或若干个语句。在大括号里边可以是几个语句，但是在大括号外边，语句块是被当作一个语句的。语句块是可以嵌套的，也就是说，一个语句块里边可以再包含一个或多个语句块，例如下面的代码。

```
var Library = {};
Library.ease = function () {
    this.target = 0;
    this.position = 0;
    this.move = function (target, speed)
    {
        this.position += (target - this.position) * speed;
```

```
        }
    }
```

上面的代码中，Library.ease = function()后面是一个语句块，其里面又包含语句 this.target=0; 和 this.position=0;以及语句块 this.move=function(target,speed){...}。

3.12.3 变量和数据类型

变量是程序语言中的名词，即它们是执行动作或被执行动作的实体。JavaScript 脚本语言中的所有变量都必须有数据类型。变量的数据类型决定了变量可以包含的值和可以在其上进行的操作。下面介绍变量的使用方法。

1. 什么是变量

从字面上看，变量是可变的量；从编程角度讲，变量是用于存储某种/某些数值的存储器。所储存的值可以是数字、字符或其他的一些东西。

2. 变量的命名

变量的命名有以下要求。

- 只包含字母、数字和/或下划线。
- 要以字母开头。
- 不能太长。
- 不能与 JavaScript 保留字（即 keywords、reservedwords 等凡是可以用来做 JavaScript 命令的字）重复。

而且，变量是区分大小写的，例如 variable 和 Variable 是两个不同的变量。不仅如此，大部分命令和“对象”都是区分大小写的。

注意：

给变量命名，最好避免用单个字母“a”、“b”、“c”等，而改用能清楚表达该变量在程序中作用的词语。这样，不仅别人能更容易了解你的程序，而且你在以后要修改程序的时候，也很快会记得该变量的作用。变量名一般用小写，如果是由多个单词组成的，那么第一个单词用小写，其他单词的第一个字母用大写，例如 myVariable 和 myAnotherVariable。这样做仅仅是为了美观和易读，因为 JavaScript 一些命令都是用这种方法命名的，例如 indexOf、charAt 等。

3. 变量声明及赋值

没有声明的变量不能使用，否则会报错：“未定义”。声明变量可以用如下格式。

```
var <变量> [= <值或者表达式>];
```

var 这个关键字用作声明变量。最简单的声明方法就是“var <变量>;”，这将为<变量>准备内存，给它赋初始值“null”。如果加上“= <值>”，则给<变量>赋予自定的初始值<值或者表达式>。

4．数据类型

JavaScript 有 6 种数据类型，主要的类型有 string、number、object 及 boolean，其他两种类型为 undefined 和 null。

- string 字符串类型：字符串是用单引号或双引号来说明的（使用单引号来输入包含引号的字符串），例如"The cow jumped over the moon."。
- number 数据类型：JavaScript 支持整数和浮点数。整数可以为正数、0 或者负数；浮点数可以包含小数点，也可以包含一个“e”（大小写均可，在科学记数法中表示“10 的幂”），或者同时包含这两项。
- object 类型：除了上面提到的各种常用类型外，对象也是 JavaScript 中的重要组成部分。
- boolean 类型（布尔值类型）：可能的 boolean 值有 true 和 false。这是两个特殊值，不能用作 1 和 0。
- undefined 数据类型：一个为 undefined 的值就是指在变量被创建后，但未给该变量赋值以前所具有的值。
- null 数据类型：null 值就是没有任何值，什么也不表示。

3.12.4　语句

JavaScript 所提供的语句分为以下几大类：变量声明、函数定义、条件和分支语句、循环语句、对象操作语句和注释语句。

1．变量声明

变量声明的赋值语句为 var，语法如下。

```
var 变量名称 [=初始值]
```

例如下面的代码。

```
var computer = 32
```

定义 computer 是一个变量，且有初值 32。

2．函数定义

函数需要预先定义，其定义语句为 function、return，语法如下。

```
function 函数名称 (参数)
{
    函数执行部分
}
return 表达式
```

return 语句指明将返回的值。

例如下面的代码。

```
function square ( x )
{
```

```
    return x*x
}
```

上述代码表示定义一个函数，计算 x 的平方。

3. 条件和分支语句

if...else 语句完成了程序流程块中的分支功能：如果其中的条件成立，则程序执行紧接着条件的语句或语句块；否则程序执行 else 中的语句或语句块，语法如下。

```
if(条件)
 {
    执行语句1
} else{
    执行语句2
}
```

例如下面的代码。

```
if (result == true)
 {
    response = "你答对了！"
} else{
    response = "你错了！"
}
```

分支语句 switch 可以根据一个变量的不同取值采取不同的处理方法，语法如下。

```
switch (expression)
{
    case label1: 语句串1;
    case label2: 语句串2;
    case label3: 语句串3;
    ...
    default: 语句串n;
}
```

如果表达式的取值同程序中提供的任何一条语句都不匹配，将执行 default 中的语句。例如，要把学生的成绩按优、良、中、差分类，可使用 switch 语句，代码如下。

```
switch (parseInt(score / 10)) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        result = 'fail';
        break;
    case 6:
    case 7:
        result = 'pass';
        break;
    case 8:
```

```
            result = 'good';
            break;
        case 9:
            result = 'excellent';
            break;
        default:
            if (score == 100)
                result = 'excellent';
            else
                result = 'error';
}
```

4. 循环语句

循环语句包括 for、for...in、while、break、continue，使用方法分别如下。

（1）for 语句的语法如下。

```
for (初始化部分;条件部分;循环部分)
{
    执行部分...
}
```

只要循环的条件成立，循环体就被反复执行，代码如下。

```
for (i = 1; i < 10; i++) document.write(i);
```

本语句先给 i 赋初始值 1，然后执行 document.write(i)语句（作用是在文档中写 i 的值）；重复时 i++，也就是把 i 加 1；循环直到不满足 i<10，也就是 i>=10 时结束。本语句的输出结果为 123456789。

（2）for...in 语句与 for 语句有一点不同，它循环的范围是一个对象所有的属性或是一个数组的所有元素。

for...in 语句的语法如下。

```
for (变量 in 对象或数组)
{
    语句...
}
```

（3）while 语句所控制的循环不断地测试条件，如果条件始终成立，则一直循环，直到条件不再成立。

while 语句的语法如下。

```
while (条件)
{
    执行语句...
}
```

（4）break 语句结束当前的各种循环，并执行循环的下一条语句。

（5）continue 语句结束当前的循环，并马上开始下一个循环，代码如下。

```
for (i = 1; i < 10; i++) {
```

```
        if (i == 3 || i == 5 || i == 8) continue;
        document.write(i);
}
```

本语句当 i 为 3、5、8 时结束当前循环，继续进行下一次循环，所以输出结果为 124679。

5. 对象操作语句

对象操作语句包括 with、new、this，使用方法分别如下。

（1）with 语句的语法如下。

```
with (对象名称){
    执行语句
}
```

如果想使用某个对象的许多属性或方法时，只要在 with 语句的()中写出这个对象的名称，然后在下面的执行语句中直接写这个对象的属性名或方法名就可以了。

（2）new 语句是一种对象构造器，可以用 new 语句来定义一个新对象。

new 语句的语法如下。

```
新对象名称= new 真正的对象名
```

例如，可以这样定义一个新的日期对象：var curr＝new Date()，然后变量 curr 就具有了 Date 对象的属性。

（3）this 运算符总是指向当前的对象。

6. 注释语句

CSS 样式表中经常会使用注释语句，以方便查找修改。

- //：单行注释。
- /*....*/：多行注释。

3.13 JavaScript 的对象及其属性和方法

JavaScript 是基于对象的编程，而不是完全面向对象的编程。对象，通俗地说，是变量的集合体，对象提供对于数据的一致的组织手段，描述了一类事物的共同属性。在 JavaScript 中，可以使用以下几种对象。

- 由浏览器根据 Web 页面的内容自动提供的对象。
- JavaScript 的内置对象，例如 Date、Math 等。
- 服务器上的固有对象。
- 用户自定义的对象。

JavaScript 中的对象是由属性和方法两个基本元素构成。对象的属性是指对象的背景色、长度、名称等。对象的方法是指对属性所进行的操作，就是一个对象自己所属的函数，例如对对象取整，使对象获得焦点，使对象获得随机数等一系列操作。

举个例子来说，将汽车看成是一个对象，汽车的颜色、大小、品牌等叫做属性，而发动、刹车、拐弯等就叫做方法。

可以采用这样的方法来访问对象的属性。

```
<对象名称>.<属性名称>
```

例如：mycomputer.year=1996，mycomputer.owner = "me"。可以采用这样的方法，将对象的方法同函数联系起来。

```
<对象>.<方法名字> = <函数名字或对象.属性.方法名>
```

例如：this.display=display，document.writeln("this is method")。

多看或多写一些程序，就会理解对象的方法和属性的含义了。

3.14　事件处理

事件是浏览器响应用户交互操作的一种机制，JavaScript 的事件处理机制可以改变浏览器响应用户操作的方式，这样就开发出具有交互性并易于使用的网页。浏览器为了响应某个事件而进行的处理过程叫做事件处理。

3.14.1　事件处理的类型

事件定义了用户与页面交互时产生的各种操作，例如单击超级链接或按钮时，就产生一个单击（click）操作事件。浏览器在程序运行的大部分时间都等待交互事件的发生，并在事件发生时自动调用事件处理函数，完成事件处理过程。

事件不仅可以在用户交互过程中产生，而且浏览器自己的一些动作也可以产生事件，例如当载入一个页面时，就会发生 load 事件；而当卸载一个页面时，就会发生 unload 事件等。归纳起来，必须使用的事件有以下 3 大类。

- 引起页面之间跳转的事件，主要是超链接事件。
- 事件浏览器自己引起的事件。
- 事件在表单内部同界面对象的交互。

3.14.2　指定事件处理

指定事件处理程序有以下 3 种方法。

1. 直接在 HTML 标记中指定

这种方法是用得最普遍的，代码如下。

```
<标记 ... ... 事件="事件处理程序" [事件="事件处理程序" ...]>
```

来看下面的例子。

```
<body ... onload="alert('网页读取完成，请慢慢欣赏！')" onunload="alert('再见！')">
```

这样的定义<body>标记，能使文档读取完毕的时候弹出一个对话框，写着“网页读取完成，请慢慢欣赏”；在用户退出文档（或者关闭窗口，或者到另一个页面去）的时候弹出“再见”。

2. 编写特定对象特定事件的 JavaScript

这种方法用得比较少，但是在某些场合还是很好用的，代码如下。

```
<script language="JavaScript" for="对象" event="事件">
    ...
    (事件处理程序代码)
    ...
</script>
```

例如：

```
<script language="JavaScript" for="window" event="onload">
    alert('网页读取完成，请慢慢欣赏！');
</script>
```

3. 在 JavaScript 中说明

代码如下。

```
<事件主角 - 对象>.<事件> = <事件处理程序>;
```

用这种方法要注意的是，“事件处理程序”是真正的代码，而不是字符串形式的代码。如果事件处理程序是一个自定义函数，如无使用参数的需要，就不要加“()”，代码如下。

```
...
function ignoreError( ) {
    return true;
}
...
window.onerror = ignoreError;
```

window.onerror 没有使用“()”，这个例子将 ignoreError()函数定义为 window 对象的 onerror 事件的处理程序。它的效果是忽略该 window 对象下的任何错误（由于引用不允许访问的 location 对象产生的“没有权限”错误是不能忽略的）。

3.15 小结

本章先循序渐进地讲解了 XHTML 语法以及 XHTML 的常用标签，然后又介绍了 JavaScript 语言基础。

在介绍 XHTML 的内容中，由 XHTML 的基本结构开始，讲到 XHTML 的语法构成，然后对 XHTML 语法规范做了介绍，接下来介绍了 XHTML 常用标签的使用方法，这些常用标签归

纳起来包含如下几种：div 标签，span 标签，标题 h1～h6 标签，列表标签 ul、ol、li，p 标签，br 标签，img 标签和表单标签。

在介绍 JavaScript 脚本基础的内容中，由 JavaScript 的语言基础开始，讲到 JavaScript 的对象以及属性，然后简单介绍了 JavaScript 的语言基础，最后对事件处理的基本方法做了介绍。

对本章的 XHTML 知识点的前后联系进行归纳总结，结构导图如图 3-26 所示。

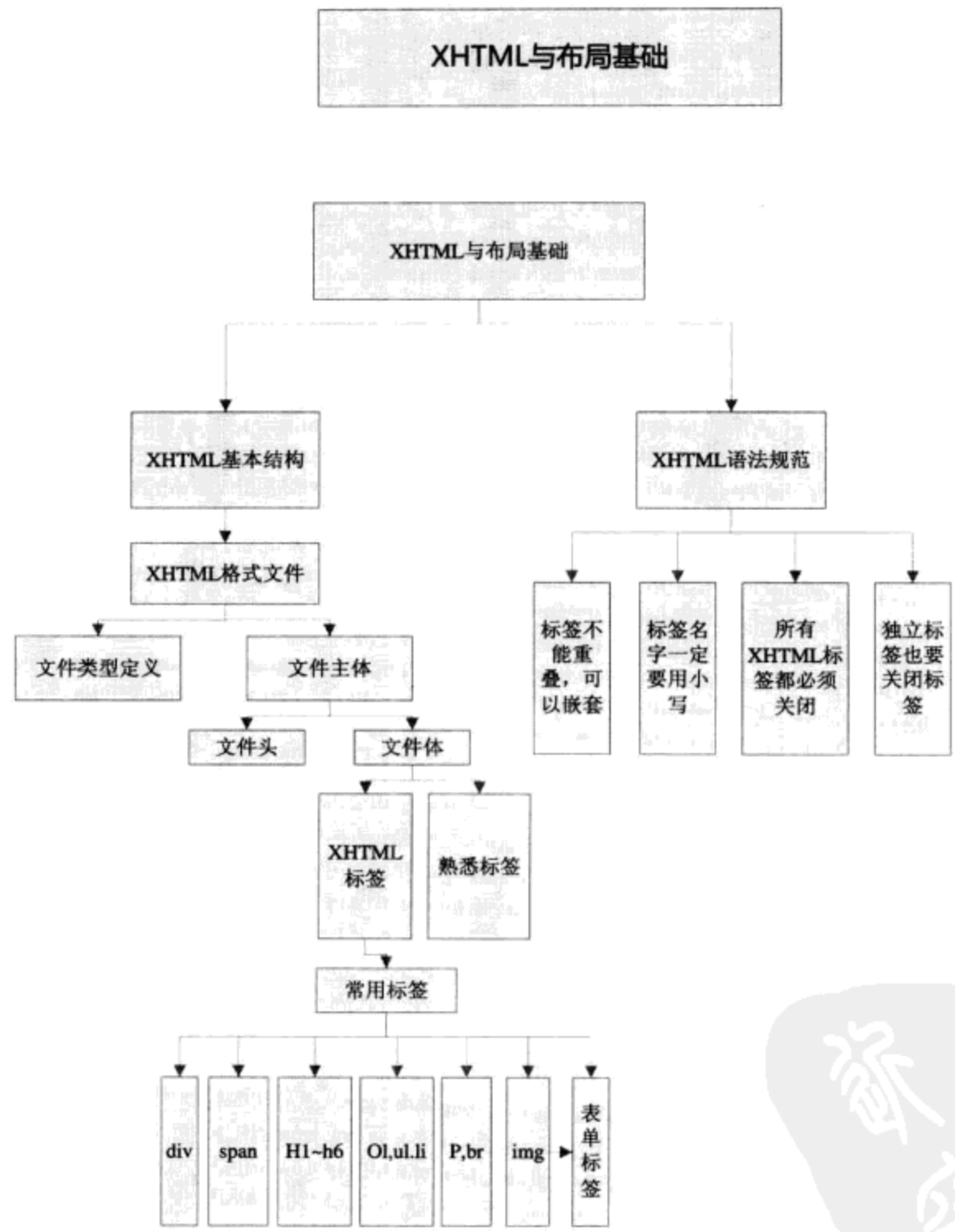


图 3-26　本章知识点结构导图

3.16　习题

1. XHTML 的基本结构及语法构成是什么？
2. XHTML 有哪些语法规范？

3．简述 span 标签与 div 标签的区别。

4．Javascript 脚本语言变量的命名规则是什么？

5．JavaScript 语言指定事件处理有哪些方法？

第4章 浏览器的兼容与解析问题

本章将重点介绍浏览器的兼容问题出现的原因以及兼容需要考虑哪些浏览器，提出思路与解决方案，以帮助初学者快速适应 Web 标准的浏览器的设计开发，改善网站的兼容性。

4.1 兼容问题的由来

W3C 希望有一种标准来改善网络表现层的面貌，于 2000 年推出了 Web 标准。Web 标准的目的是使用统一的技术和模式进行网站设计，提升网站的可用性、改善网站结构、降低建设成本。

而使用统一标准的过程并不是一帆风顺的。

（1）旧版本兼容问题。IE 浏览器大概从 IE 5 开始才对 CSS 2.0 有较好的支持，但是并不完美，对于 CSS 中的盒模型原理、浮动模式等标准的定义并非严格执行，以及部分 CSS 2.0 的属性在 IE 5 中完全没有效果。根据一项网络调查数据显示，目前浏览者还有约 5%的用户在使用 IE 5/IE 5.5 这些老式浏览器。

（2）还有一个问题就是不同浏览器的表现。例如现在流行的主流浏览器 IE 6、IE 7 和 Firefox，分别由两个公司独立开发，在浏览器的核心架构上有着明显的区别。虽然都是以 Web 标准作为开发基础，但是由于架构及开发方式的不同，最终对于 Web 标准的表现上，难免会有很多区别，这样导致了一些 CSS 设计方式在两个浏览器中的表现会有所不同。

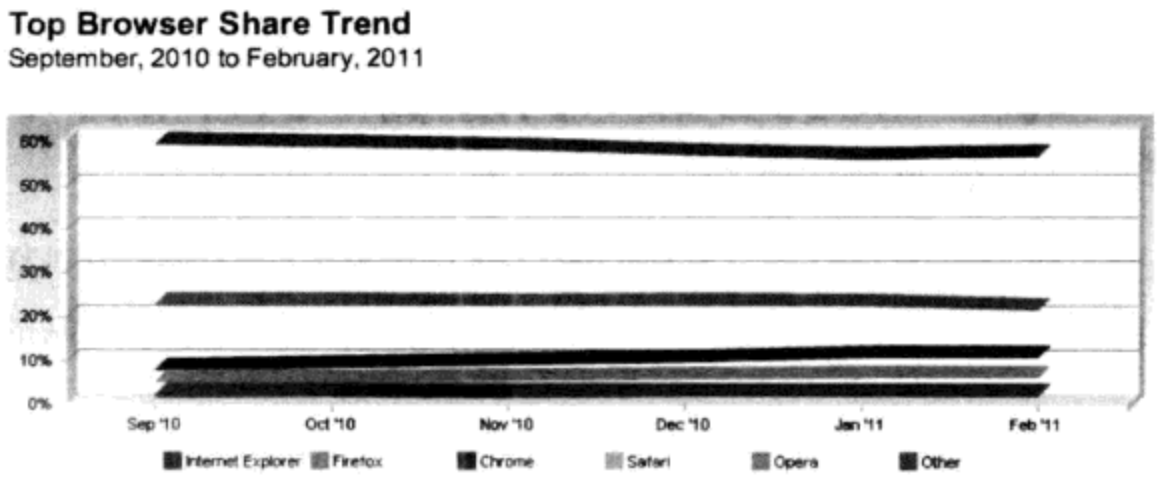
4.2 需要兼容的常用浏览器

解决浏览器兼容问题，也就是决定针对哪些浏览器做兼容处理。根据浏览器的市场占有份额，判断现在在设计网页的时候需要考虑兼容的浏览器类型。美国互联网流量监测机构 Net Application 发布了 2010 年 12 月份最新浏览器市场排行及占有率数据，具体结果如图 4-1 所示。

图 4-1 浏览器市场份额调查图表

从图表中可以看出，现在的市场基本上完全被 IE、Firefox、chrome、Safari、Opera、Mozilla 占领，至于其他的浏览器，在解决浏览器兼容这方面问题的时候，完全可以忽略它们。

下面是来自 marketshare.com 的最新统计数据，浏览器市场从 2010 年 9 月至 2011 年 2 月份额趋势如图 4-2 所示。



Month	Internet Explorer	Firefox	Chrome	Safari	Opera	Other
September, 2010	59.62%	22.97%	7.99%	5.27%	2.40%	1.75%
October, 2010	59.18%	22.83%	8.50%	5.36%	2.29%	1.85%
November, 2010	58.44%	22.76%	9.26%	5.55%	2.20%	1.79%
December, 2010	57.08%	22.81%	9.98%	5.89%	2.23%	2.01%
January, 2011	56.00%	22.75%	10.70%	6.30%	2.28%	1.98%
February, 2011	56.77%	21.74%	10.93%	6.36%	2.15%	2.06%

图 4-2　浏览器最近一年时间内市场份额趋势

从图中可以看出，主要访问浏览器为 IE 7、IE 6、Firefox 2.0、Firefox 3.0、Safari 3.0、Safari 3.1，并且 IE 7 逐渐成为市场占有率最高的浏览器。通过 w3schools 和 marketshare 上的共 5 组数据，可以得出结论：现阶段，解决浏览器兼容问题至少需要考虑的浏览器有 IE 6、IE 7、Firefox、Safari 和 Opera。

注意：

目前，IE 8 也已经面世了，这样对于网页设计人员的要求又增加了一项，就是兼容 IE 8 浏览器的显示问题。

本书主要针对浏览器市场上的 3 大主流浏览器做兼容测试，即 IE 6、IE 7 和 Firefox。

4.3　CSS hack 技术

解决浏览器兼容问题，首先要了解一个概念——CSS hack。本节就来讲解 CSS hack 的基本概念和使用 CSS hack 的方法。

4.3.1　什么是 CSS hack

hack 一词在网络上很流行，一般人们马上便会联想到“黑客”。其实，hack 一词一般是指对程序或者系统进行非官方的修改，或是非官方的补丁。

CSS hack 可以理解为 CSS 黑客程序，是一种改善 CSS 在不同浏览器下的表现形式的技巧方法。CSS hack 技术是通过一些浏览器特殊支持或者不支持的语句，达到 CSS 样式能够被浏览器解析或者不能解析的目的，以实现不同浏览器下的兼容问题。

4.3.2　使用 CSS hack

CSS hack 是针对一些浏览器之间的显示差异问题，而进行单独的样式修复。如何使用 CSS hack，请看下面的示例。

!important 是 CSS 1 所定义的语法，作用是提高指定样式规则的应用优先权，语法格式如下。

```
{ 属性 : 属性值 !important };
```

即!important 写在 CSS 属性值的最后面，代码如下。

```
box{
    color:red !important;
}
```

最重要的一点是，IE 6 之前的版本一直都不支持这个语法，而到了 IE 7 以后才开始对这个语法有了显示。因此就可以利用这一点来分别给 IE 6 和 IE 7 不同的样式定义，利用此方法编写 CSS 样式，解决 IE 6 和 IE 7 浏览器下的样式差异，具体代码如下。

```
.colorHack {
    border:20px  solid #60A179 !important;
    border:20px  solid #00F;
    padding: 30px;
    width : 300px;
}
```

在 IE 7 中浏览时候，能够理解!important 的语法，因此显示#60A179 的颜色，预览效果如图 4-3 所示。在 IE 6 中浏览时候，不能够理解!important 的语法，因此显示#00F 的颜色，预览效果如图 4-4 所示。

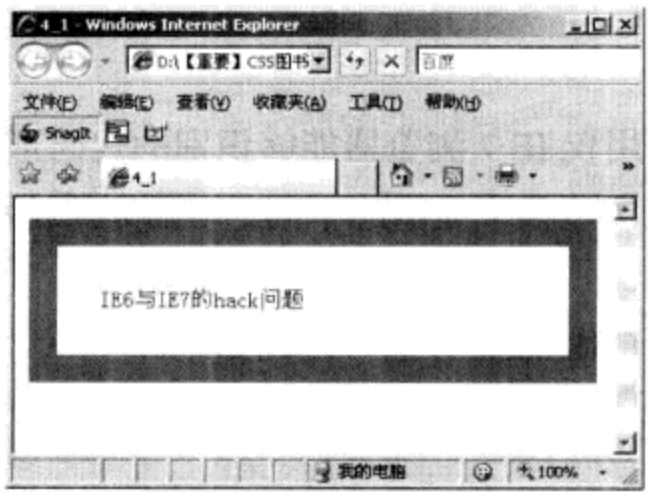


图 4-3　IE 7 显示的效果

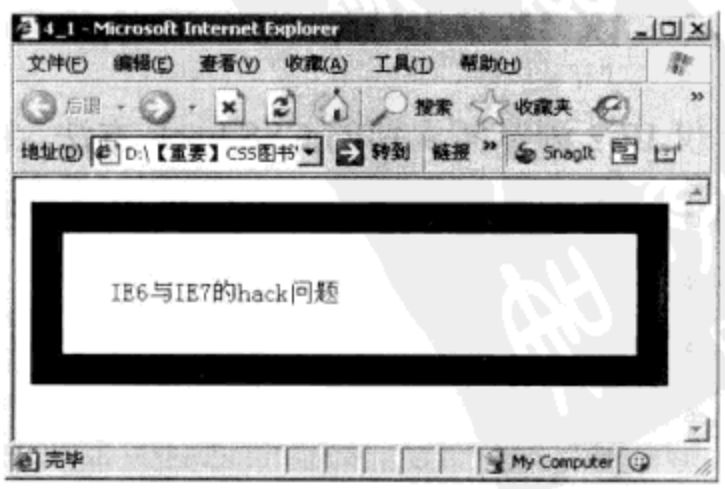


图 4-4　IE 6 显示的效果

4.4　常用 CSS hack 方法介绍

除了!important 针对 IE 6 的隐藏代码处理外，CSS hack 还有许多别的应用形式，包括针对不同浏览器、不同浏览器的不同版本、同一浏览器的不同版本等各种各样的隐藏代码方法。本节将介绍常用的 CSS hack 形式及使用方法。

注意：

在本章中，所讨论的每一个 CSS hack 方法都是针对目前常用的两种浏览器的支持产品，这两种浏览器是 Windows 平台下的 IE 浏览器的各种版本和 Mozilla/Firefox 的支持（Mozilla 及 Firefox 浏览器由于使用同一内核开发，因此在这里理解为同一浏览器产品）。

4.4.1　屏蔽 IE 6 浏览器

IE 浏览器是目前浏览器市场上的主流浏览器，所以可以选择一种 CSS hack 方法先屏蔽 IE 浏览器，针对其他浏览器实现显示 CSS 的样式。

屏蔽 IE 6 浏览器的 CSS hack 方法，原理就是使用仅 IE 6 浏览器不能够识别的代码形式。!important 这个代码在 IE 6 下是不识别的，在 IE 7 和 Mozilla/Firefox 可以识别。依然使用 4.3 节的示例，在 IE 7、IE 6 和 Mozilla/Firefox 浏览器下的显示效果对比如图 4-5 所示。

（a）IE 7　　（b）Firefox　　（c）IE 6

图 4-5　3 种浏览器中的显示效果 1

4.4.2　仅 IE 7 识别

仅 IE 7 浏览器识别的 CSS hack 方法，原理就是使用仅 IE 7 浏览器能够识别的代码形式。使用这种方法，能够区别出 IE 7 与 IE 6、Mozilla/Firefox 浏览器下显示的不同。下面介绍这种 CSS hack 方法。

当需要只针对 IE 7 做样式的时候就可以采用这个 CSS hack 方法，代码如下。

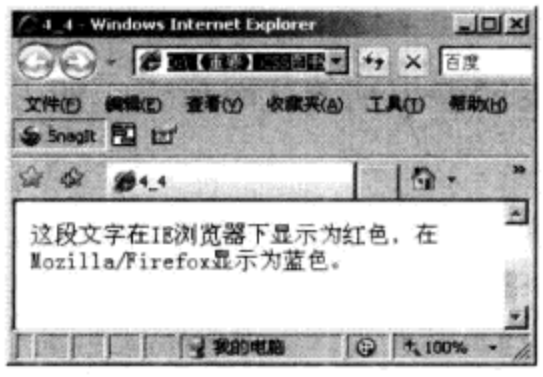
```
*+html select {
    property:value
}
```

在以上代码中，select 是选择符，property 是属性，value 是属性值，关键的代码是“*+html”，这段代码要放在选择符的前面，并且空一格。

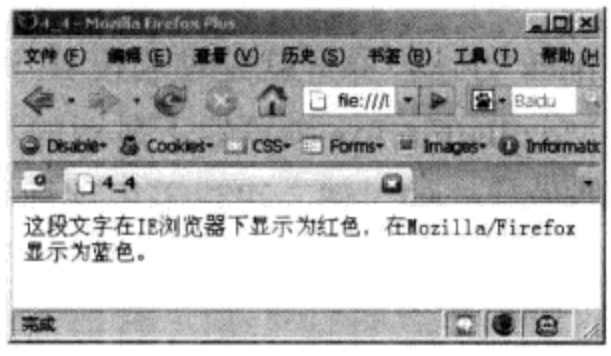
例如下面的代码，span 标签内的文字在 IE 7 下显示为蓝色，而在其他颜色的浏览器下则显示默认颜色黑色。

```
*+html span {
    color:blue;
}
```

以上代码在 IE 7、Mozilla/Firefox 以及 IE 6 浏览器中的显示效果如图 4-6 所示。



（a）IE 7



（b）Firefox

（c）IE 6

图 4-6　3 种浏览器中的显示效果 2

4.4.3　仅 IE 6 识别

仅 IE 6 浏览器识别的 CSS hack 方法，原理就是使用仅 IE 6 浏览器能够识别的代码形式。使用这种方法，能够区别出 IE 6 与 IE 7、Mozilla/Firefox 浏览器下显示的不同。

当需要针对仅是 IE 6 浏览器做样式的时候，可以采用这样的 CSS hack 方法，代码如下。

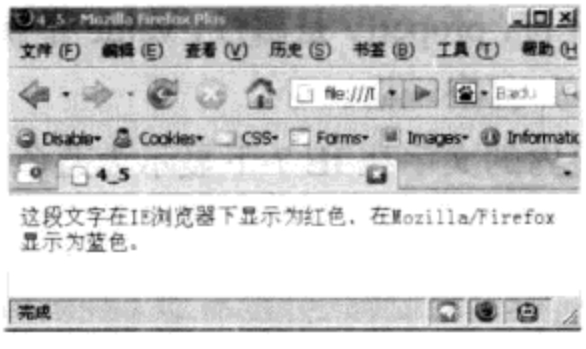
```
select {
    _property: value;
}
```

在以上代码中，下划线“_”只有在 IE 6 浏览器中是可以识别的，在 IE 7、Mozilla/Firefox 这样的浏览器中不能够被识别，所以作为用来区分 IE 6 浏览器和其他浏览器的方法。例如下面的代码，span 标签内的文字在 IE 6 下显示为蓝色，而在 IE 7 以及 Mozilla/Firefox 浏览器下则显示默认颜色黑色。

```
span {
    _color:blue;
}
```

以上代码在 IE 7、Mozilla/Firefox 以及 IE 6 浏览器中的显示效果如图 4-7 所示。

（a）IE 7



（b）Firefox

（c）IE 6

图 4-7　3 种浏览器中的显示效果 3

4.4.4　仅 IE 识别

仅 IE 浏览器识别的 CSS hack 方法，原理就是使用仅 IE 浏览器能够识别的代码形式。使用这种方法，能够区别出 IE 与 Mozilla/Firefox 浏览器下显示的不同。

当需要针对仅是 IE 浏览器做样式的时候，可以采用这样的 CSS hack 方法，代码如下。

```
select {
    property: value;
}
```

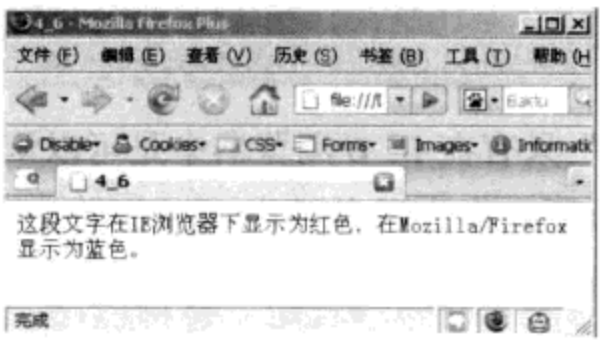
在以上代码中，*在 IE 浏览器中是识别的，只有在 Mozilla/Firefox 这样的浏览器中不能够识别，所以作为用来区分 IE 浏览器和其他品牌的浏览器的方法。

例如下面的代码，span 标签内的文字在 IE 下显示为蓝色，而在 Mozilla/Firefox 浏览器下则显示默认颜色黑色。

```
span {
    color:blue;
}
```

以上代码在 IE 7、Mozilla/Firefox 以及 IE 6 浏览器中显示效果如图 4-8 所示。

（a）IE 7



（b）Firefox

（c）IE 6

图 4-8　3 种浏览器中的显示效果 4

4.4.5　兼容 IE 6、IE 7、Firefox 浏览器

在实际应用中，通常是需要对几种主流浏览器进行同时兼容处理，结合前几小节所介绍的 CSS hack 方法，灵活组合使用，可以完美兼容现在市场上几大主流浏览器。

总结前几小节的 CSS hack 方法，得到 3 种浏览器对 hack 代码的支持关系表，如表 4-1 所示。

表 4-1　3 种浏览器对hack代码的支持关系表

项目	IE 6	IE 7	Firefox
*	√	√	×
!important	×	√	√
_	√	×	×

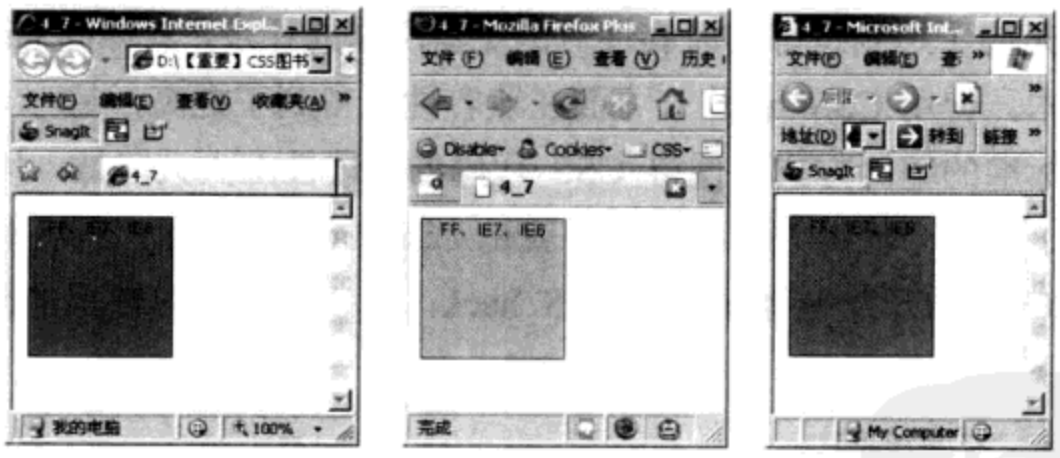
根据上一图表，即可以解决不同浏览器之间的兼容问题了。下面通过两个实例，说明如何实现完美兼容 IE 6、IE 7 和 Firefox 这 3 种浏览器的 CSS 样式的。

1．实例一

CSS 代码如下。

```
#box {
    width:100px;
    height:100px;
    border:#FF3399 1px solid;
    background:#FF99FF;
    *background:#999900 !important;
    *background:#6699CC;
    float:left;
    margin-right:10px;
}
```

在以上代码中，注意书写的顺序是 Firefox 第一、IE 7 第二、IE 6 在最后。以上代码在 IE 7、Mozilla/Firefox 以及 IE 6 浏览器中的显示效果如图 4-9 所示。



（a）IE 7　　（b）Firefox　　（c）IE 6

图 4-9　3 种浏览器中的显示效果 5

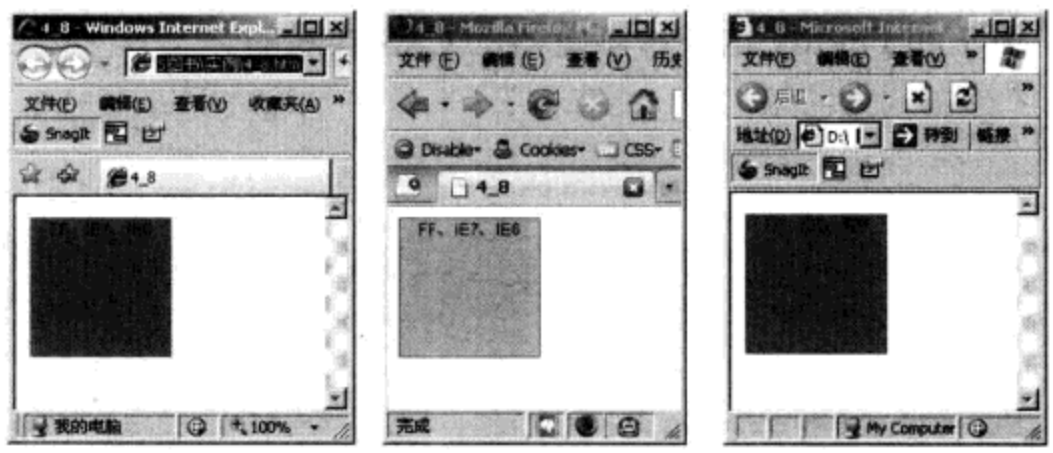
2．实例二

对实例一进行了修改，代码如下。

```
#box{
    width:100px;
    height:100px;
    border:#FF3399 1px solid;
    background:# FF99FF;
    *background:# 99900;
```

```
    _background:#6699CC;
    float:left;
    margin-right:10px;
}
```

在以上代码中，注意书写的顺序是 Firefox 第一、IE 7 第二、IE 6 在最后。其实 IE 6 的 *background 也可以写成_background，因为只有 IE 6 支持下划线，IE 7 和 Firefox 都不支持。以上代码在 IE 7、Mozilla/Firefox 以及 IE 6 浏览器中的显示效果如图 4-10 所示。



（a）IE 7　　（b）Firefox　　（c）IE 6

图 4-10　3 种浏览器中的显示效果 6

利用以上实例所使用到的方法，在实际应用过程中，当某一个浏览器显示的效果与预期所要出现的效果不一致时，可以使用针对该浏览器的 CSS hack 方法解决兼容问题。

4.5　CSS hack 管理

CSS hack 技术是为了达到良好的兼容性对浏览器所作的让步，就如同 hack 一词的本意，只是一个非官方的技术。使用 CSS hack 一方面在兼容性上得到了提高，另一方面由于 CSS hack 方法多数没有遵循标准的编码方式，使得 CSS 代码的可读性上大大降低，因此对于 CSS hack 的使用来说，引发了另一个值得关注的问题，就是 CSS hack 的管理。只有对 CSS hack 进行充分管理，才能够在 CSS 的开发中有序使用 CSS hack，真正做到兼容性上的改善。在 CSS hack 的管理方面给出以下建议。

1．如果没必要，不要使用

没有必要使用的情况很多，有时候设计虽然在不同的浏览器下表现不同，但是只是细微的。例如两个浏览器下，只不过宽度就差不到 10px，或是其他类似的情况下，如果不影响整体效果的话，没有必要完全将两个浏览器下的显示状态通过 CSS hack 调到完全统一，这些多余的代码和后期带来的维护成本相信要远远大于 10px 的程度。

2．充分了解 CSS 与 CSS hack

尽管本节中的内容一直在讲述如何使用 CSS hack 以及各种 CSS hack 在不同浏览器下的表现，但是在实际应用 CSS hack 之前，非常有必要亲自动手了解一下这些 CSS hack 的功能及使用方法。在实际开发中对 CSS 编码及 CSS hack 的要求也是十分严谨的，并非通过直接复制和粘

贴这些 CSS hack 代码这么简单。除了直接应用这些 CSS hack，应当先了解它们的功能，在应用中尽可能地使用最适合的 CSS hack 来完成目标。

3. 经常注释

注释永远是一种了解 XHTML 中的标记与 CSS 的好方法，如果需要使用 CSS hack，尽量去注释 CSS hack 代码。这样才能够帮助自己以及团队在开发中能够很好地理解对方在做些什么，因为毕竟这些人并非都使用一样的 CSS hack，并非每人都使用同一种方法。

4.6　IE 条件注释功能

条件注释是 IE 特有的一种功能，能够对 IE 系列产品进行单独的 XHTML 代码处理。注意，条件注释主要是针对 XHTML 的，而非 CSS。IE 条件注释功能相当强大，可以进行 true 和 false 的条件判断，代码如下。

```
<!-[if IE]>此内容只有 IE 可见<![endif]-->
<!-[if IE6.0]>此内容只有 IE6.0 可见<![endif]-->
```

条件注释能够被 IE 判断是什么版本的浏览器，并在符合条件的情况下显示其中的内容，从 IE 5.0 到 IE 7.0，都支持条件注释功能，而且版本号可以精确到小数点后 4 位。

除了标准判断方式外，条件注释还支持感叹号非操作，代码如下。

```
<!-[if !IE5.0]>此内容除了 IE5.0 之外都可见<![endif]-->
```

另外，条件注释还支持前缀，用于判断更高版本或是更低版本，代码如下。

```
<!-[if gt IE5]>此内容 IE5 以上版本都可见<![endif]-->
```

使用 gt 表示 greater than，当前条件版本的以上版本，不包含当前版本。此外还有如下几个前缀可以使用。

- lt 表示 less than，当前条件的版本的以下版本，不包含当前条件的版本。
- gte 表示 greater than or equal，当前条件版本的以上版本，并包含当前条件的版本。
- lte 表示 less than or equal，当前条件版本的以下版本，并包含当前条件的版本。

条件注释功能非常小巧灵活，能够在 XHTML 代码层实现一些 IE 版本上的条件判断。

4.7　小结

本章主要介绍网页制作过程中出现的浏览器兼容问题，包括兼容问题产生的原因、需要兼容的常用浏览器类型、CSS hack 技术、常用的 CSS hack 方法、CSS hack 管理以及 IE 条件注释功能等内容。

对本章的知识点前后联系进行归纳总结，结构导图如图 4-11 所示。

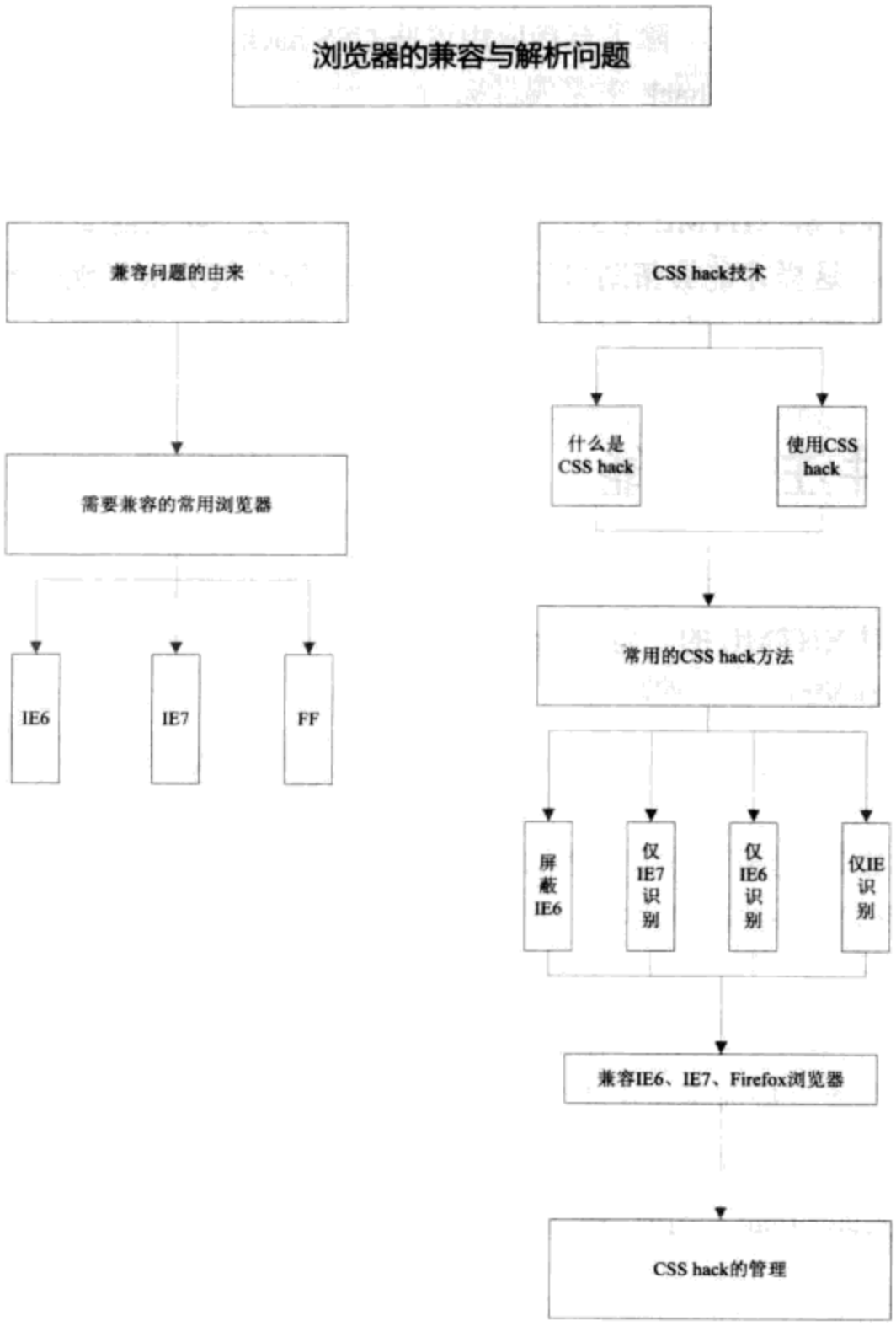
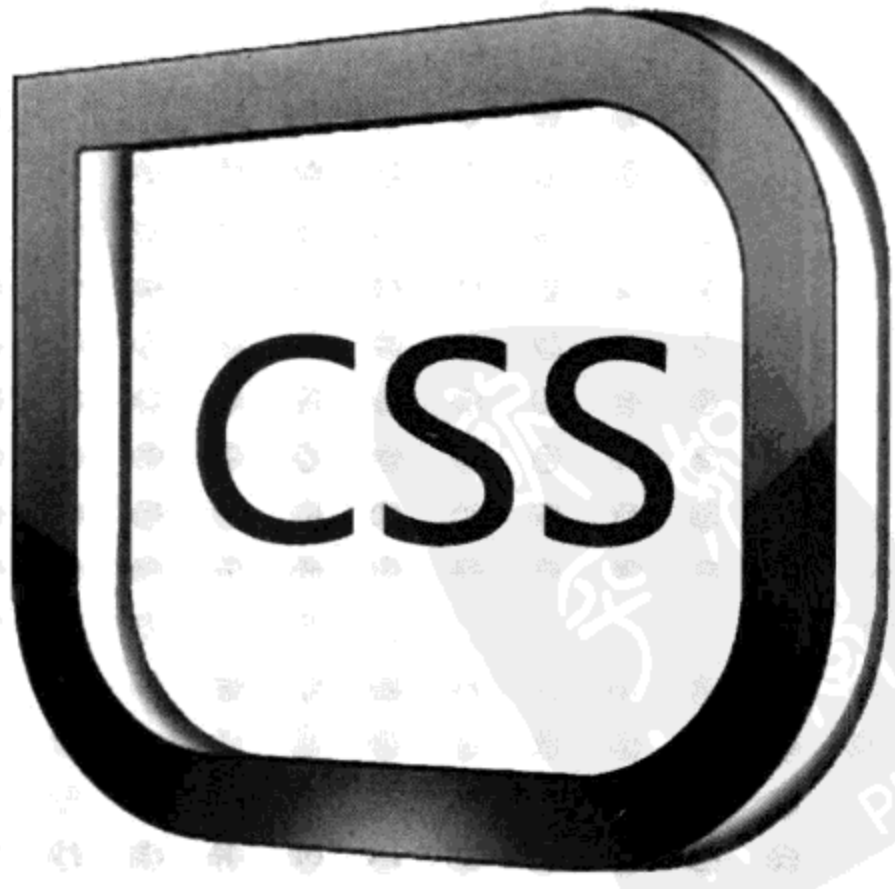


图 4-11 本章知识点结构导图

4.8 习题

1. 目前需要兼容的常见浏览器有哪些？
2. 什么是 CSS hack 技术？
3. 常用的 CSS hack 形式及使用方法是什么？
4. CSS hack 管理应该注意哪些事项？
5. 简述 IE 条件注释功能。

第二篇　实例制作篇



第5章 使用CSS制作背景

使用背景（background）是网页设计制作中的一种常用的手法。无论是直接用背景颜色还是使用背景图，都能够给网页整体带来丰富的视觉效果。HTML 的一些元素已经提供了对background 属性的支持，可以设置背景色或者背景图片，例如 table 表格、tr 表格行和 td 单元格等。不过 HTML 制作出来的背景形式比较单一。CSS 对于元素的背景色设置提供了更多的途径，本章将要探讨使用 CSS 的背景属性来替代传统 table 表格布局的方法。

5.1 制作背景颜色

背景颜色属性是背景应用中最基础的属性。如图 5-1 所示是使用黑色背景颜色的网页。

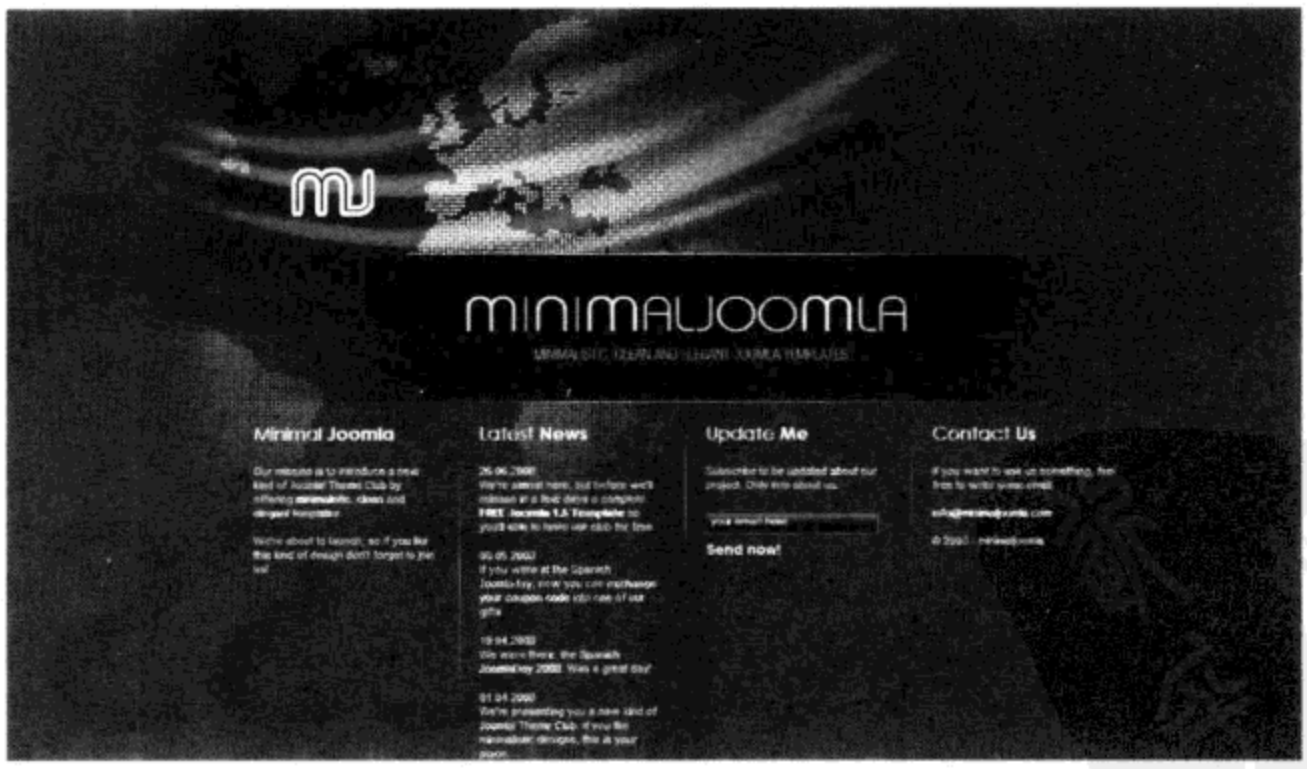


图 5-1 使用黑色背景颜色的网页

在 CSS 中，用来定义背景颜色的属性是 background-color，其语法结构和制作的实例如下。

1. 背景颜色属性的语法结构

背景颜色属性 background-color 的语法结构如下。

```
background-color: transparent | color;
```

其中各个属性值的含义如下。

- transparent：定义背景颜色为透明（默认值）。
- color：使用各种颜色值定义背景颜色。

2. 制作背景颜色实例

本实例讲解制作一个在不同元素中使用不同颜色值定义背景颜色的效果，代码如下。

```
<h1>16 进制颜色值</h1>
<h2>RGB 颜色值</h2>
<h3>颜色名称</h3>
<h4>透明背景</h4>
```

在上述 XHTML 代码中，<h1>、<h2>、<h3>、<h4>是标题元素。在 XHTML 中，共有<h1>至<h6>六个标题元素。每种标题元素都有各自的显示方式，使用 CSS 样式可以改变标题元素默认的显示方式。在以上代码中添加 CSS 样式，代码如下。

```
h1{ background-color:#808080; }
h2{ background-color:rgb(237,237,237); }
h3{ background-color: gray; }
h4{ background-color: transparent; }
```

以上 CSS 代码中分别对 h1、h2、h3、h4 元素定义了不同的背景颜色值，其具体含义和作用如下。

- h1：使用十六进制颜色表示方式，定义背景颜色为灰色。
- h2：使用 RGB 颜色值表示方式，定义背景颜色为灰色。
- h3：使用 IE 浏览器可识别的一种颜色名称。不同浏览器都有自己独特的支持一些特定的颜色名称，每个浏览器对颜色的名称定义可能不会相同，所以会出现不兼容的问题，这种方法不被推荐。
- h4：使用 transparent 的透明模式，使用这个颜色值，页面中的背景显示为透明效果。每个元素所默认的背景色就是透明色。

实例运行后在浏览器中的预览效果如图 5-2 所示。

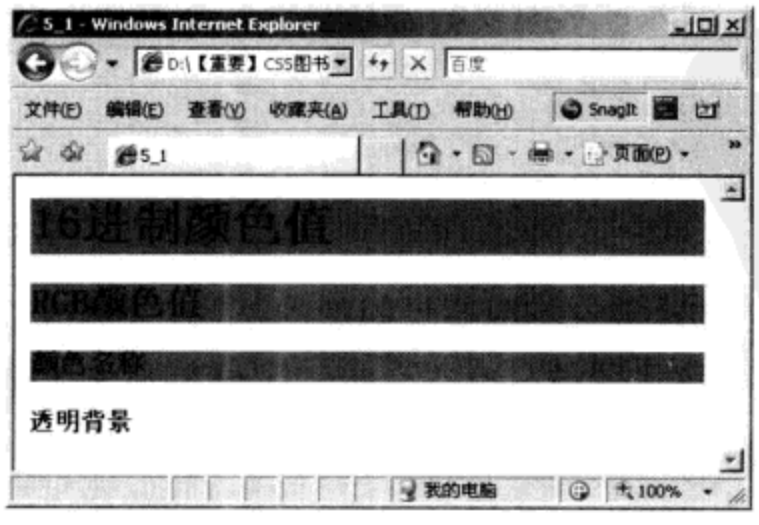


图 5-2　CSS 设置的背景颜色

5.2　给元素添加背景图片

在制作页面的时候，使用背景图片可以完成各种复杂的背景效果，使用背景图片的网页如图 5-3 所示。

图 5-3　使用背景图片的网页

在 CSS 中，通常使用背景图片属性 background-image 定义背景图像的路径。使用背景图像重复属性 background-repeat 定义背景图像的重复效果。本节将详细讲解各种背景图片属性的语法和实例。

5.2.1　指定背景图像

指定背景图像是指，定义元素中使用背景图像的路径。在 CSS 中，用来指定背景图像的属性是 background-image。由于背景图像只有在一定的区域范围内才能够显示，所以在指定背景图像的时候，通常要定义元素的宽度和高度等属性，具体内容如下。

1．背景图像属性

背景图像 background-image 属性用来定义图像背景路径，其语法结构如下。

```
background-image: url(图片地址 | none);
```

background-image 这个属性的主要功能就是用来显示图片，只要在 background-image 后面加上图片链接地址就可以。如果不希望显示背景图片，只要在 background-image 后面设置为 none，none 也是默认值。

2. 宽度和高度属性

宽度属性 width 和高度属性 height 用来界定元素内容的显示范围，其语法结构如下。

```
width: length;
height: length;
```

注意：

宽度和高度属性定义的区域，只是元素内容的大小，并不是元素占有空间的大小（后面的章节将陆续讲解关于元素占有空间大小的相关属性）。

3. 指定元素的背景图像

本实例将制作一个在元素中指定背景图像的实例，其 XHTML 代码如下。

```
<div id="content">
</div>
```

在以上代码中添加 CSS 样式，其代码如下。

```
#content{
    border: 1px solid #FF0000;                          /*边框样式*/
    height: 400px;                                      /*页面高度*/
    background-image: url(images/5_2_01.jpg);           /*背景图片*/
}
```

网页中 id 为 content 的元素设置了使用图片 5_2_01.jpg 作为背景，预览效果发现与传统 HTML 格式布局并无差异。在默认状态下，背景同样以平铺的方式出现在网页元素中，如图 5-4 所示。

图 5-4　CSS 制作平铺背景

5.2.2　制作重复的背景图像

制作重复的背景图像是指，定义元素中背景图像的重复方式。在 CSS 中，用来指定背景图像的属性是 background-repeat。制作重复背景图像的具体内容和相关实例如下。

1．背景图像重复属性

背景图像重复属性 background-repeat，这个属性与 background-image 一起使用，决定图像是否重复，其语法结构如下。

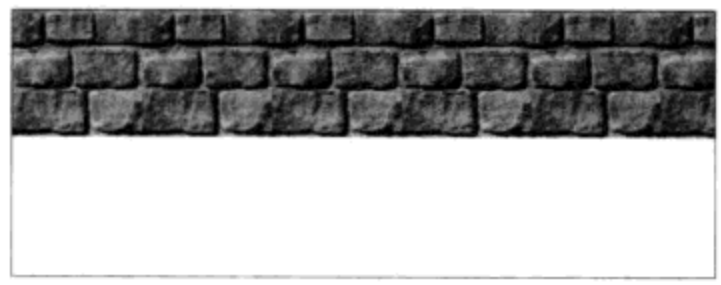
```
background-repeat: repeat | no-repeat | repeat-x | repeat-y;
```

background-repeat 是用来对背景进行平铺控制的属性，可选的属性值如下。

- repeat：即默认方式，完全平铺背景。
- no-repeat：在 X 及 Y 轴方向均不平铺。
- repeat-x：横向平铺背景。
- repeat-y：纵向平铺背景。

以一种形象化的方式表现 background-repeat 属性的 4 种属性值所达到的效果，如图 5-5 所示。

原始图片



●水平平铺

●垂直平铺

●完全平铺

图 5-5　background-repeat 属性的 4 种属性值

下面将使用 background-repeat 属性制作网页的重复背景实例。

2．制作重复的背景图像

背景图像平铺可以单独地横向平铺，单独地纵向平铺，也可以横向和纵向同时平铺。选择

不同的平铺方式，能够实现不同的背景效果。

本实例制作背景图片水平平铺的效果，代码如下。

```
<div id="content">
</div>
```

在以上代码中添加 CSS 样式，代码如下。

```
#content{
    border: 1px solid #FF0000;                          /*边框样式*/
    height: 400px;                                      /*页面高度*/
    background-image: url(images/5_2_01.jpg);           /*背景图片*/
    background-repeat: repeat-x;                        /*背景图片平铺方式*/
}
```

在以上代码中，使用类选择符#content 定义了页面内容区域的高度和背景图片，这个高度超过背景图片的高度，目的是突出背景图片只是横向平铺。定义背景图像重复属性之后，页面的显示效果如图 5-6 所示。

图 5-6　CSS 制作背景横向平铺

使用 background-repeat:repeat-x 属性值，背景只在 X 轴即横向进行平铺，Y 轴即纵向并没有进行平铺。这就是 CSS 对背景的细节控制之一。

5.3　制作不动的背景

除了平铺方式的设置之外，CSS 还提供了另一个强大的功能，即背景的定位。传统表格式布局中，即使使用图像，也没有办法提供精确到像素级的定位方式，往往通过透明的 gif 图片来强迫图片到某一位置。而在 CSS 中，即使是背景，也能够做到精确定位。如图 5-7 所示是一个使用了固定背景图片的网页。

图 5-7　背景图片固定的网页

1．背景定位属性语法结构

背景定位属性 background-position 用来设置背景的起始位置，语法结构如下。

```
background-position:position | x% y% | xpos ypos;
```

其中各个属性值的含义如下。

- position：定义背景图像的位置。可用值有 top left | top center | top right | center left | center center | center right | bottom left | bottom center | bottom right。

position 的属性值使用图示表示相对的位置关系，如图 5-8 所示。

- x% y%：以百分比形式定位背景图像位置，第一个是水平位置，第二个是垂直位置。左上角是 0% 0%，右下角是 100% 100%，还可以混合使用百分号和 position。

这种属性值所表示的位置关系以一种形象化的方式说明，如图 5-9 所示。

- xpos ypos：以坐标形式定位背景图像位置，第一个是水平位置，第二个是垂直位置，左上角是 0px 0px。实际上，这种坐标的形式表示位置关系与 x% y%百分比形式原理一致，只是表示距离的单位不同，这里使用具体的长度单位来表示。

图 5-8　position 属性值的相对位置关系

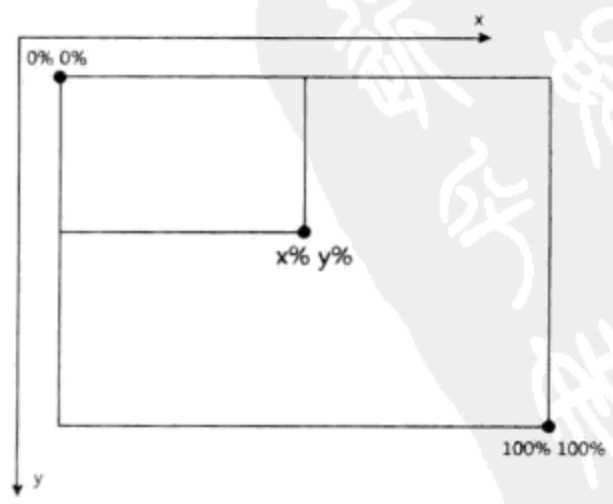


图 5-9　x% y%属性值表示的位置关系

2. 制作背景定位的实例

继续 5.2 节的实例，现在改进一下代码。

```
#content{
    border: 1px solid #FF0000;                          /*边框样式*/
    height: 400px;                                      /*页面高度*/
    background-image: url(images/5_2_01.jpg);           /*背景图片*/
    background-repeat: no-repeat;                       /*背景图片不平铺*/
    background-position: left bottom;                   /*背景图片定位于页面左下方*/
}
```

在改进后的 CSS 代码中添加了 background-position: left bottom 属性，使背景定位在网页的左侧和底部。页面的显示效果如图 5-10 所示。

图 5-10　被定位的背景

预览网页效果，图片在背景中被放置到了左下方。repeat 模式设为 no-repeat，使背景图片只出现一次，不进行平铺。这里的重点在于 background-position 属性。有时网页需要图片根据内容自动改变对边的对齐方式，使用这 5 种对齐方式相当方便。background-position 的设置方式如下。

```
background-position:左对齐方式 上对齐方式
```

本节的实例背景被设置为了左对齐 left、上对齐 bottom 模式。background-position 除了使用对齐方式外，还可以通过百分比及绝对像素单位进行精确控制，代码如下。

```
#content{
    border: 1px solid #FF0000;                          /*边框样式*/
    height:400px;                                       /*页面高度*/
    background-image:url(images/5_2_01.jpg);            /*背景图片*/
    background-repeat: no-repeat;                       /*背景图片不平铺*/
    background-position: 30% 20%;                       /*背景图片定位于距离页面左边 30%、上边 20%*/
}
```

以上 CSS 代码中，背景位置使用的是百分比值进行定位，左对齐 30%位置，上对齐 20%位置，页面效果如图 5-11 所示。

图 5-11 百分比定位背景图片

当使用百分比及像素进行控制时，这里指的是背景在元素中相对于左侧空间的距离、上侧空间的距离。通过改进的代码，使背景左端永远处于元素 30%的位置，上端处于元素 20%的位置。

5.4 给网页添加背景

背景平铺是网页设计中经常会用到的一个功能。现在结合前面讲过的背景属性 background-repeat、background-position、background-image，制作网页添加渐变式背景效果的实例。渐变式背景是网页中经常会用到的一种背景效果，采用 CSS 的 repeat-x 方法制作，只需要将渐变背景图按照需要的高度设计。

本实例将使用 CSS 中 background-repeat:repeat-x 制作一个渐变式背景图片网页的效果，代码如下。

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

在上述 XHTML 代码中，<body></body>作为网页主体内容，将直接显示在浏览器窗口中，里面放置的是页面的所有内容，例如图片、文字、表格、表单和超级链接等。在以上代码中添加 CSS 样式，代码如下。

```
body{
    background-image: url(images/5_3_01.gif);      /*背景图片*/
    background-repeat: repeat-x;                    /*背景图片 x 轴平铺*/
    background-position: top;                       /*背景居顶定位*/
    background-color: # 055269;                     /*背景颜色*/
}
```

以上 CSS 代码中，为 body 元素分别定义了 background-image、background-repeat、background-position 和 background-color 属性，其具体含义和作用如下。

- background-image：定义了网页的背景图像是图片 5_3_01.gif，这是一张渐变的背景图像，如图 5-12 所示。
- background-repeat: repeat-x：设置背景图像的横向平铺。
- background-position: top：把背景图像定位在网页的顶部。
- background-color：把网页的背景颜色设置为# 055269，这个颜色是背景图像底部最后一行像素的颜色值，这样就实现了渐变式图片的网页背景效果了。实际效果如图 5-13 所示。

图 5-12　渐变式背景图像

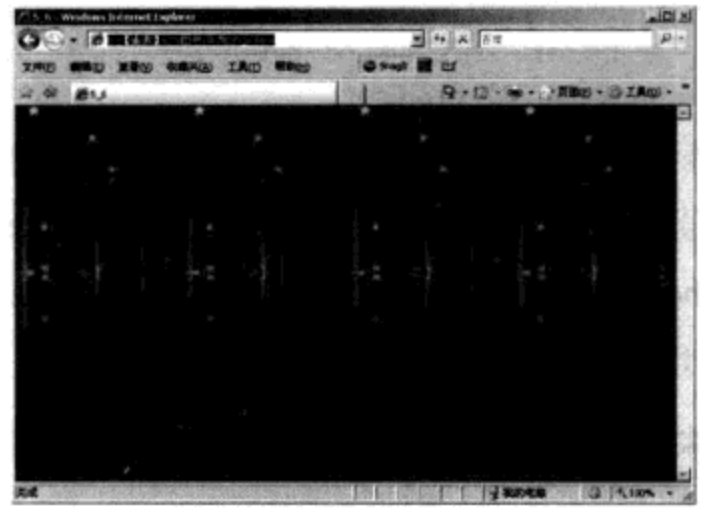


图 5-13　渐变式背景的网页

5.5　制作滚动页面的背景

在传统背景定义中，如果定义了一个网页 body 上的背景，网页的背景往往会自动根据内容的移动而变化。而在 CSS 中，针对于 body 元素上背景的控制，提供了另一个选择，即固定背景模式。使用固定背景模式的情况下，背景将不跟随内容的滚动而进行位移，背景始终保持在固定的位置。下面分别讲解背景附件属性的使用方法和具体实例应用。

1．背景附件属性语法结构

CSS 背景附件属性 background-attachment 主要是用来控制背景的滚动与定位的，对于实现背景的控制是有帮助的。背景附件属性语法结构代码如下。

```
background-attathment: scroll | fixed
```

其中各个属性值的含义如下。

- scroll：默认值，随着页面的滚动，背景图片将移动。
- fixed：随着页面的滚动，背景图片不会移动。这个属性值在 IE 6 中只有 body 支持此属性。

2．制作页面滚动而背景不移动的实例

继续使用 5.4 节的代码进行改进，不同的是这次将对整个 body 进行背景设置，代码如下。

```
<div id="content">
    1<br />2<br />3<br />4<br />5<br />6<br />7<br />8<br />9<br />10<br />11<br />12<br />13<br
/>14<br />15<br />16<br />17<br />18<br />19<br />20<br />21<br />22<br />23<br />24<br />25<br />
```

```
</div>
```

CSS 代码如下。

```
body{
    color: white;                                    /*页面颜色为白色*/
    background-image:url(images/5_2_01.jpg);         /*背景图片*/
    background-attachment: fixed;                    /*设置背景不随页面滚动*/
}
#content{
    borer: 1px solid #FF0000;                        /*边框样式*/
    height: 1000;                                    /*页面高度*/
}
```

页面的显示效果如图 5-14 和图 5-15 所示，分别是页面滚动之前和页面滚动之后的效果。

body 的 background-attachment 属性便是用于控制背景滚动的方式。默认值为 scroll，也可以使用 fixed 值类表示背景为固定位置。

图 5-14 页面滚动但是背景不移动（页面滚动前）

图 5-15 页面滚动但是背景不移动（页面滚动后）

5.6 综合使用背景

前几节介绍了众多的与背景相关的属性，其实 CSS 还提供了一个复合的背景属性，将以上那些背景属性复合在一个属性里面。这样的作用是减少了 CSS 代码的重复使用率，使代码优化。下面分别讲解背景属性的使用方法和具体实例应用。

1. 背景复合属性语法结构

背景复合属性 background 语法结构代码如下。

```
background: background-color | background-image | background-repeat | background-position | background-attachment
```

其中各个属性值含义如下。

- background-color：使用背景颜色属性的颜色值。
- background-image：使用背景图片属性的 url 值。

- background-repeat：使用背景重复属性的 repeat、repeat-x、repeat-y、no-repeat。
- background-position：使用背景定位属性的位置属性值。
- background-attachment：使用背景附件属性的 scroll 或 fixed 属性值。

2. 使用背景复合属性制作网页背景实例

制作网页背景的方法已经在前面的章节中介绍过，不过前面的方法是使用各种背景属性，一个一个地堆叠一起，使得 CSS 代码过于冗余。本节使用一个背景复合属性，就实现一个背景的各种样式定义，代码如下。

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

CSS 代码如下。

```
body {
    background:#000000 url(images/5_1.jpg) no-repeat fixed 0px -10px;          /*背景设置*/
}
```

在以上代码中，使用指定选择符标签 body 为页面设置了背景的样式，这里使用了背景图片，并且设置了平铺方式和定位等属性，背景图片 5_1.jpg 如图 5-16 所示。

此示例在浏览器中的预览效果如图 5-17 所示。

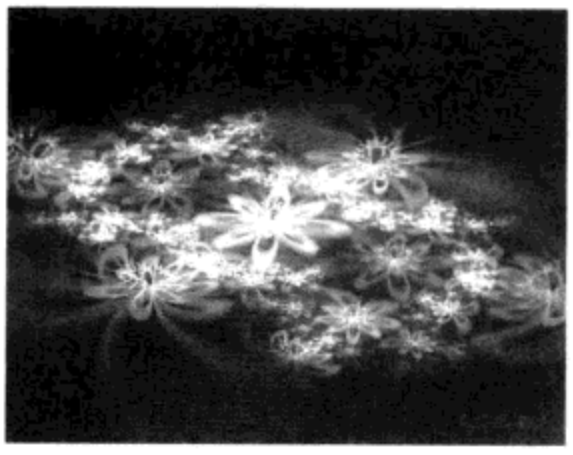


图 5-16　背景图片 5_1.jpg

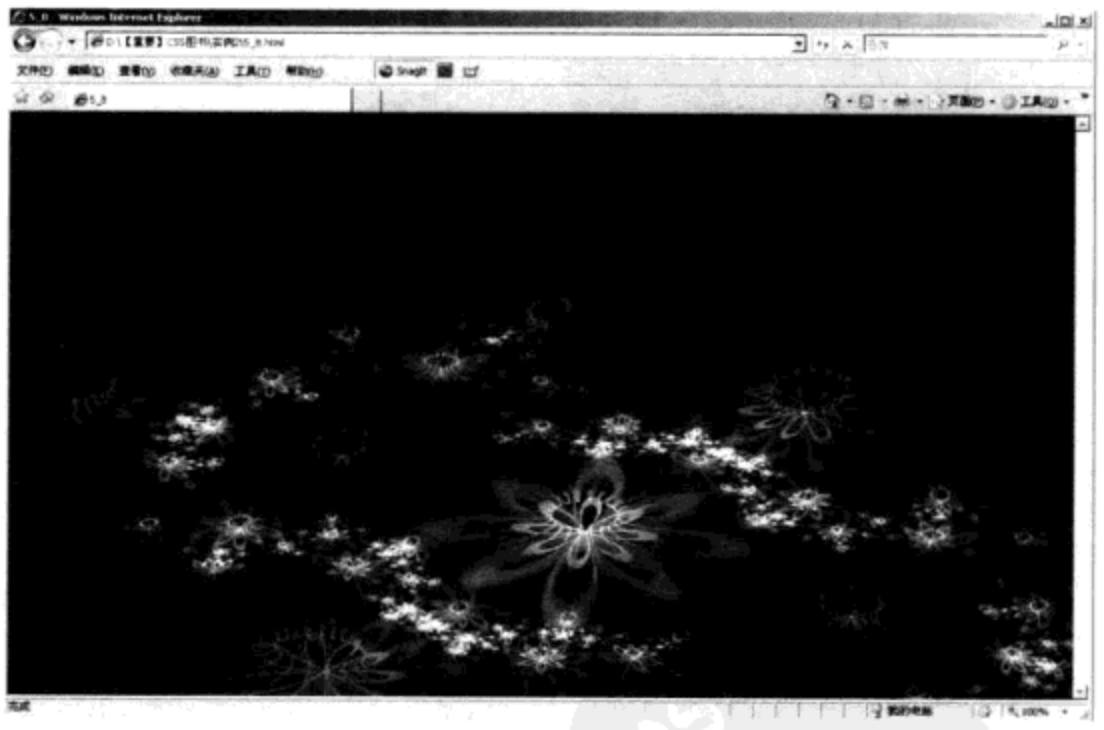


图 5-17　使用背景复合属性制作背景的效果

5.7　小结

本章讲解了使用 CSS 进行背景制作的实例，包括制作纯色的背景、给元素添加背景图片、制作不动的背景、给网页添加背景、制作滚动的背景以及综合使用背景等实例，涉及一些 CSS 背景属性及属性值的应用。主要涉及的背景属性有以下 6 个。

- 背景颜色属性 background-color。

- 背景图片属性 background-image。
- 背景重复属性 background-repeat。
- 背景定位属性 background-position。
- 背景附件属性 background-attachment。
- 背景复合属性 background。

对本章的知识点前后联系进行归纳总结，结构导图如图 5-18 所示。

图 5-18　本章知识点结构导图

5.8　习题

1. 怎样给元素添加背景图片？
2. 在 CSS 中，如何精确定位背景？
3. 如何制作一个渐变式背景图片？
4. CSS 背景附件属性 background-attachment 的作用是什么？
5. 简述背景复合属性的语法结构。

第6章 使用CSS布局页面顶部内容

网页顶部一般包含网站标志、标题及导航等一些网站的重要信息，它是一个网页能否吸引人眼球的重要区域。一个优秀的网页，要看其有没有高质量的页面顶部设计。网页顶部设计决定了整个网站的风格样式和视觉震撼力，优秀的顶部设计才能最大化地吸引读者的视觉注意力。

6.1 制作包含文本logo的页面顶部

logo是标志、徽标的意思，是互联网上各个网站用来区分其他站点的标志。Internet之所以叫做“互联网”，在于各个网站之间可以连接。要让其他人走入你的网站，必须提供一个让其进入的门户。而logo图形化的形式，比文字形式的链接更能吸引人的注意。在如今争夺眼球的时代，这一点尤其重要。就一个网站而言，logo即是网站的名片。而对于一个追求精美的网站，logo更是它的灵魂所在，即所谓的“点睛”之处。

本节将介绍使用CSS样式设计一个简单的文本logo，放置在页面的顶部，增强页面的效果。制作包含文本logo的页面顶部，关键方法就是文本定位。本实例使用的定位属性是外边距属性。

1. 外边距属性

外边距属性margin用于在一个声明中，设置4个外边距的所有属性的简写属性，其语法结构如下。

```
margin: 上边距 | 右边距 | 下边距 | 左边距
```

可以缩写为如下形式。

```
margin: 上下边距 | 左右边距
```

或者

```
margin: 上边距 | 左右边距 | 下边距
```

这个简写属性设置一个元素所有外边距的宽度（或者设置各边外边距的宽度）。在margin属性中，使用的宽度值如下。

- 百分比：基于父对象总高度或宽度的百分比。
- 长度值：定义一个固定的边距，单位是像素。
- auto：浏览器设定的值。

外边距margin属性图解如图6-1所示。

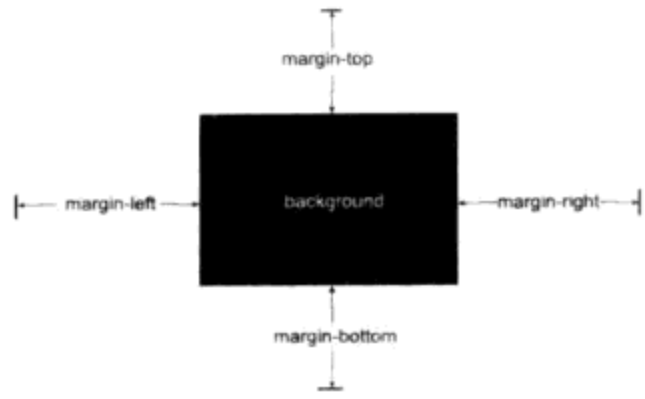


图 6-1　margin 属性图解

另外，块级元素的垂直相邻外边距会合并，而行内元素实际上不占上下外边距。行内元素的左右外边距不会合并。同样的，浮动元素的外边距也不会合并。margin 属性也可以分开设置样式，语法结构代码如下。

```
margin-left: 左边距样式;
margin-right: 右边距样式;
margin-bottom: 下边距样式;
margin-top: 上边距样式;
```

2. 制作包含文本 logo 的网页顶部实例

网页顶部一般都包括了 logo、导航菜单和 banner。现在制作包含了文字 logo 和背景图的页面顶部实例，主要使用一个图片作为整个顶部的背景，然后使用嵌套元素的方法，制作文本 logo 的内容，通过外边距属性，定义 logo 的位置。

（1）制作页面的 XHTML 代码，具体内容如下。

```
<div id="top">
    <h1>ENJOY CSS</h1>
</div>
```

这里做了一个 div 容器，用来放置头部的文字 logo 和图片内容。<h1>标签用来定义 logo 的文本内容，并通过关联的 CSS 样式，控制 logo 的显示效果。

（2）制作#top 元素的 CSS 样式，具体 CSS 代码如下。

```
#top{
    background-image: url(images/6_1.jpg);          /*设置背景图片*/
    width: 926px;                                   /*设置宽度*/
    height: 124px;                                  /*设置高度*/
    margin: 0 auto;                                 /*通过设置外边距，实现水平居中*/
}
```

在以上代码中，使用背景图片属性定义背景图片，其中加入的背景图片如图 6-2 所示。

图 6-2　文本 logo 的背景图

首先，制作背景图片之后，定义宽度和高度属性，用来显示背景图片。如果头部这个容器定义的宽度或者高度小于背景图片，背景图片显示会出问题。接下来，定义了#top 的外边距 margin: 0 auto，即左右边距设置为 auto，上下边距为 0px，实现的效果就是可以让#top 层水平居中显示。

（3）制作#top h1 元素的 CSS 样式，具体 CSS 代码如下。

```
#top h1{
    color: #9999CC;                                 /*设置字体颜色*/
```

```
    margin-left: 57px;                                    /*设置左外边距*/
    margin-top: 47px;                                     /*设置上外边距*/
}
```

在以上代码中，定义了 logo 文字样式。页面的 logo 是文字 ENJOY CSS。首先，指定了文字的字体颜色#9999CC。然后，还指定了文本的左边距 57px 和上边距 47px，也就是相对于文本的父级元素 div#top 的位置，使得 logo 定位在一个合适美观的位置。

这样，包含文本 logo 的网页顶部就完成了，预览效果如图 6-3 所示。

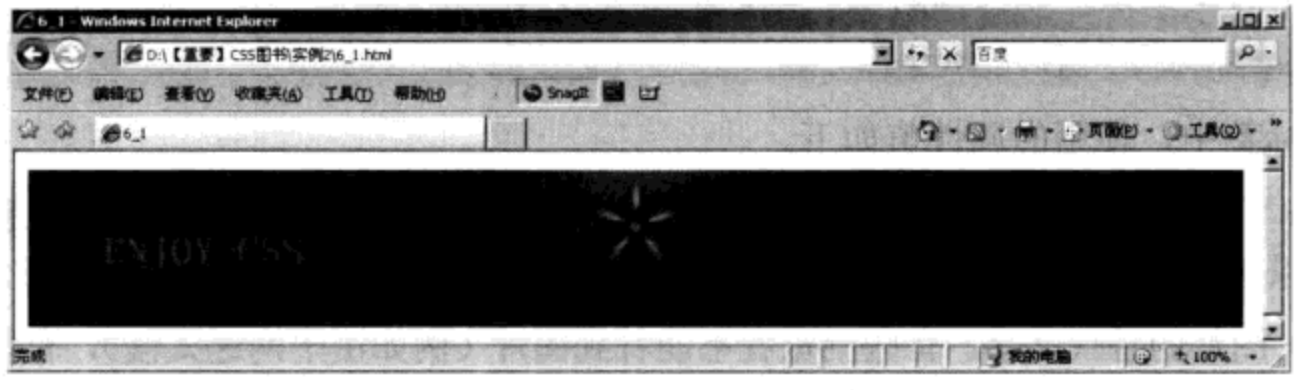


图 6-3　包含文本 logo 的页面顶部

6.2　制作包含图像 logo 的页面顶部

logo 设计的美感在于组成 logo 的元素，最基本的是形象、结构和色彩。好的网站给人印象最深的也是它的 logo 设计，为使 logo 在视觉上富有美感，要充分发挥 logo 特有的表现手段。在网页设计中，运用 CSS 样式，帮助网页实现平面设计时达到的视觉效果。如图 6-4 所示网站的顶部使用了图片 logo。

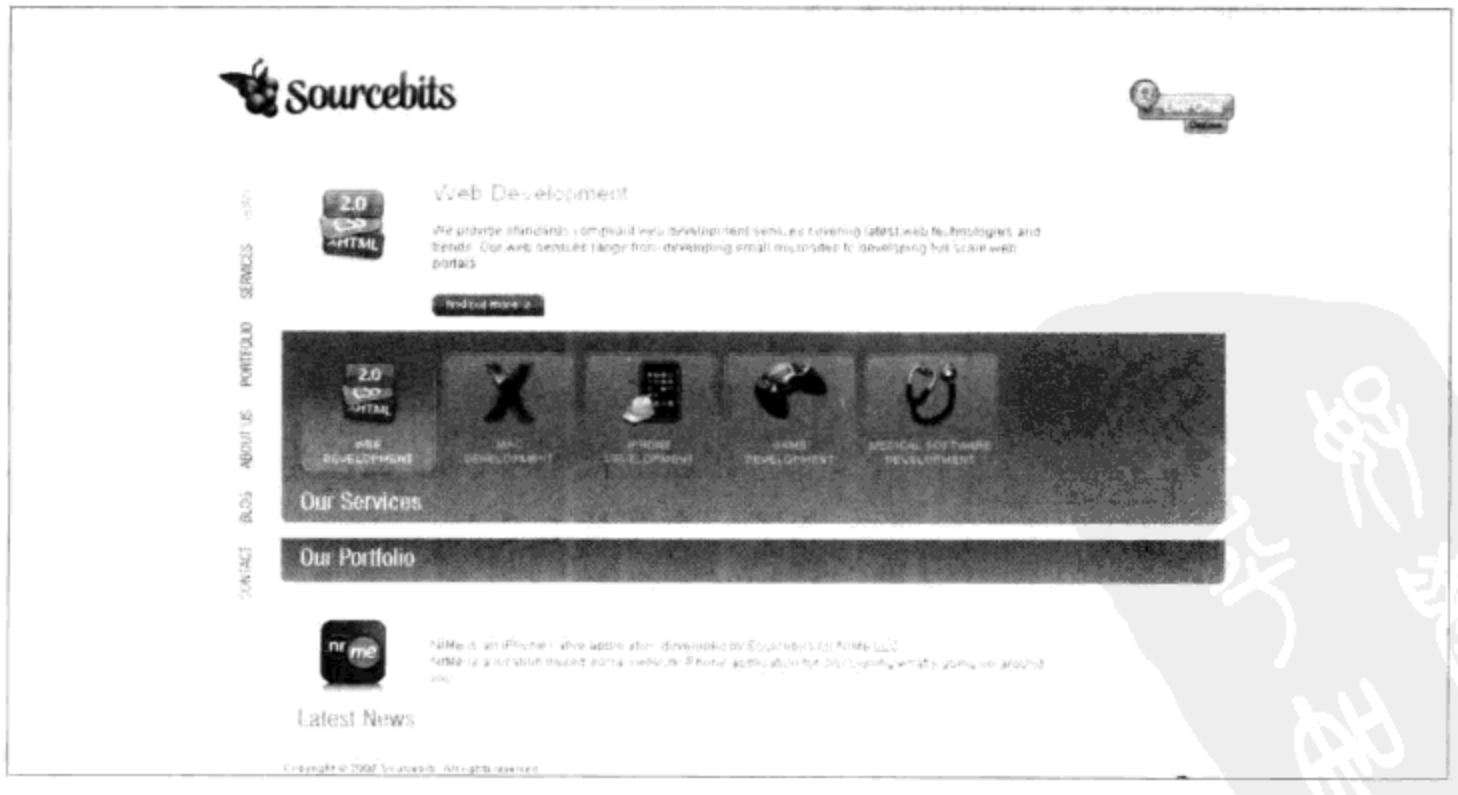


图 6-4　使用图片 logo 的网页

本节将使用 XHTML 的 img 标签，通过 CSS 对 img 标签样式的控制，达到图片定位的目的。

6.2.1　制作实例

在制作实例之前，首先要回顾一下 img 标签的使用方法。

1．img 标签

我们看到丰富多彩的网页，都是因为有了图像的作用。互联网刚刚兴起的时候，网络中全部都是纯文本的网页，非常枯燥。由于图像的使用，才有了今天丰富多彩的效果。可见，图像在网页设计中有多重要。在前面讲过，img 标签结构代码如下。

```
<img src="图片位置" alt="替代文字" />
```

在 img 标签中，使用的属性值如下。

- src：告诉浏览器图片的具体位置，就像链接的 href 标签一样告诉浏览器要链接到的文件，可以用绝对位置或者是相对位置。
- alt：图片的替代文字。有些浏览者不能看到图片（例如由于网速太慢），有些早期的浏览器也不支持图片，还有一种可能是把图片的具体位置写错了，遇到这些情况浏览者是看不到图片的，这时 alt 可以在图片的位置上显示出代替的文字。

注意：

浏览器会自动识别出图像的高度与宽度，一般不用指定，除非有其他的目的，例如有意的增大或缩小图片。

在 Web 标准的建议下，引用图片的时候尽量不要在 img 标签中使用 id 或 class 属性调用 CSS 样式。CSS 样式总是要写在独立的样式文件中的，通过子选择符指定给 img 标签样式。除了以上介绍的一些样式属性外，其他的都建议写在 CSS 代码中。

2．制作包含图像 logo 的页面顶部实例

本节制作包含图像 logo 和背景图片结合的网页顶部，关键就是指定一个背景图和插入一个前景图 logo，对上一节的代码稍作修改。

（1）制作页面的 XHTML 代码，如下。

```
<div id="top">
    <img src="images/6_2_logo.png" />
</div>
```

图 6-5　logo 图片

在以上代码中，id 属性为 top 的容器 div 里，插入了一个图片 6_2_logo.png，图片格式是 png 透底，如图 6-5 所示。

注意：

PNG（Portable Network Graphics）即便携式网络图片，常用的是用 Macromedia Fireworks 制作的。网页最常见的格式有 GIF、JPEG、PNG，3 种格式的图片各有优点。

GIF（Graphics Interchange Format，图片交换格式）格式，最高支持 8 位彩色，有“透明”、“交错”、“动画”3 个特性。GIF 图片之所以被广泛使用，就是因为这 3 个特性。“透明”是指

可以给图片指定一种颜色，使其不被显示而成为透明；“交错”是指在显示图片的过程中可以从概貌逐渐变化到全貌，看上去也就是清晰度的从小变大；“动画”是指将各幅静态的图片连续显示形成动态画面。

JPEG（Joint Photographic Experts Group，联合图片专家组）格式，最大可支持 32 位彩色。JPEG 格式的最大特色就是文件比较小，可以进行高倍率的压缩，是目前所有格式中压缩率最高的格式之一。如果对图像质量要求不高，但又要求存储大量图片，使用 JPEG 无疑是一个好办法。但是 JPEG 格式在压缩保存的过程中会以矢量最小的方式丢掉一些肉眼不易察觉的数据。

PNG 格式是由 Netscape 公司开发出来的格式，可以用于网络图像，但它不同于 GIF 格式图像，只能保存 256 色，PNG 格式可以保存 24 位的真彩色图像，并且支持透明背景和消除锯齿边缘的功能，可以在不失真的情况下压缩保存图像。但由于 PNG 格式不完全支持所有浏览器，所以在网页中的使用要比 GIF 和 JPEG 格式使用少得多。但相信随着网络的发展和因特网传输的改善，PNG 格式将是未来网页中使用的一种标准图像格式。

（2）制作#top 元素的 CSS 样式。

在 6.1 节实例的基础上，改进了的 CSS 代码如下。

```
#top{
    background-image: url(images/shadow.png);                    /*设置背景图片*/
    width: 926px;                                                /*设置宽度*/
    height: 200px;                                               /*设置高度*/
    background-color: #242424;                                   /*设置背景颜色*/
}
```

在以上代码中，首先使用背景图片属性定义背景图片，这里加入了一个格式为 png 的透底背景图片，如图 6-6 所示。

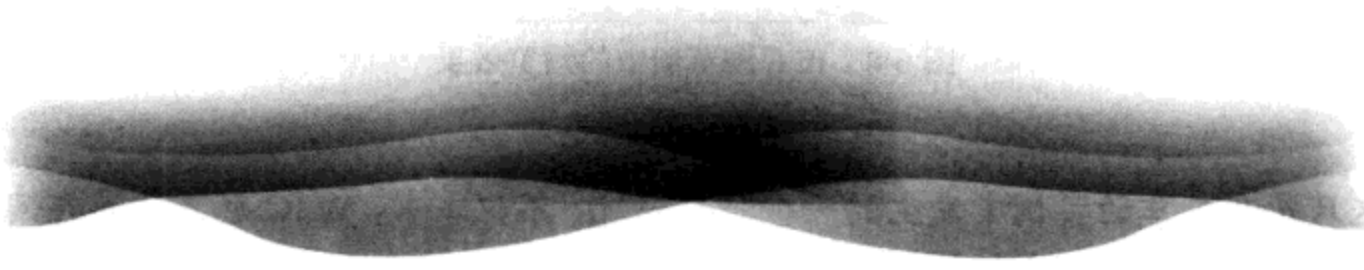


图 6-6　#top 部分的背景图

然后在#top 中又指定了宽度 926px 和高度 200px，添加了背景颜色#242424，使得背景能够显示。

（3）制作图片元素的 CSS 样式，代码如下。

```
#top img{
    margin-left: 25px;                                           /*设置左外边距*/
    margin-top: 20px;                                            /*设置上外边距*/
}
```

使用子选择符#top img 指定了 img 标签的 CSS 样式，定义了左边距和上边距，定位 logo 图片。

这样包含图像 logo 的页面顶部的实例就制作完成了，预览效果如图 6-7 所示。

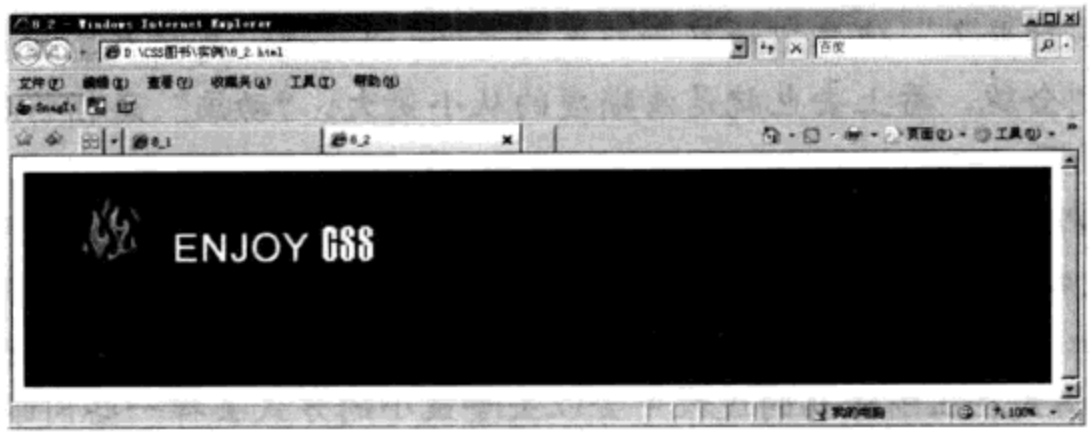


图 6-7　包含图像 logo 的页面顶部

6.2.2　兼容问题

应用 png 图片透明或半透明的特性能做出非常漂亮的网页来。Firefox 和 IE 7 对 png 的支持非常好，但是 IE 6 却无法支持 png 图片。本节的实例，在 IE 6 浏览器中的预览效果如图 6-8 所示。

从图 6-8 可以看出，png 格式的 logo 图片和背景图片在黑色的背景颜色下，都没有显示出效果。为了解决这种 IE 6 浏览器不支持 png 格式图片的问题，CSS 提供了滤镜功能。

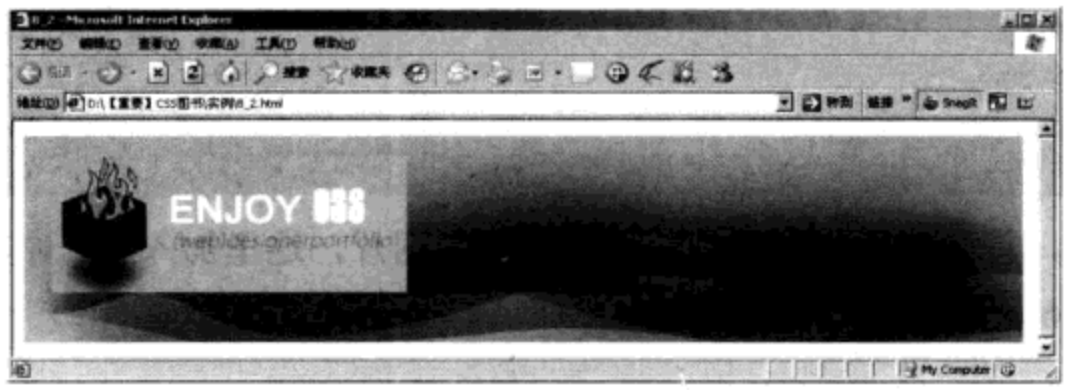


图 6-8　IE 6 浏览器中的预览效果

注意：

当然这只是在本章的实例中发生的一个特别情况，不仅本书，W3C 也不推荐在网页上使用 png 图片，所以本节介绍的解决方法仅供爱好者阅读参考。

代码格式如下。

```
filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(enabled=true,sizingMethod=
scale,src="url"
```

在这个 CSS 滤镜中，参数值的含义如下。

- enabled：可选项。设置或检索滤镜是否激活，可用值为 true 或 false。true 是默认值，表示滤镜激活；false 表示滤镜被禁止。
- sizingMethod：可选项。设置或检索滤镜作用的对象的图片在对象容器边界内的显示方式，可用值有 crop、image 和 scale。crop 表示剪切图片以适应对象尺寸；image 是默认值，表示增大或减小对象的尺寸边界以适应图片的尺寸；scale 表示缩放图片以适应对象的尺寸边界。

- src：必选项。使用绝对或相对 url 地址指定背景图像。假如忽略此参数，滤镜将不会作用。

使用这个 CSS 滤镜，修改后的 XHTML 代码如下。

```
    <div id="top">
        <div style="width:100%;filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src=
'images/6_2_logo.png',sizingMethod='image')"></div>
    </div>
```

修改后的 CSS 代码如下。

```
    #top{
        width: 926px;
        height: 200px;
        background-color: #242424;
        filter:  progid:DXImageTransform.Microsoft.AlphaImageLoader(enabled=true,  sizing-
Method=scale, src="images/shadow.png"
    }
    #top div{
        margin-left: 25px;
        margin-top: 20px;
    }
```

在以上代码中，把 png 格式图片放在了 CSS 滤镜中，在 IE 6 浏览器中的预览效果如图 6-9 所示。

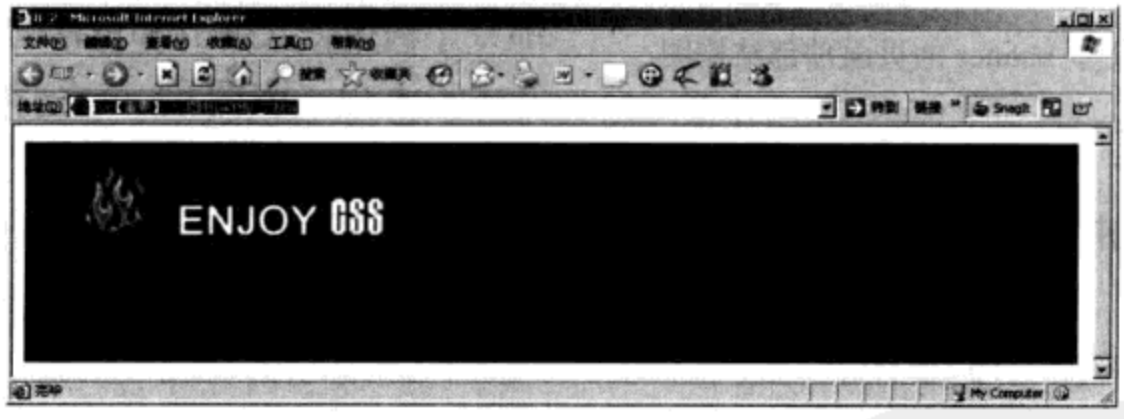


图 6-9　使用 CSS 滤镜兼容 IE 6 浏览器后的预览效果

6.3　制作包含文本 banner 的页面顶部

banner 的中文意思是“网幅图像广告”，通常形式是图片或者段落文本。以 gif、jpg 等格式建立的图像文件或者段落文本，定位在网页中，大多用来表现网络广告内容。如图 6-10 所示，在网页顶部的 banner 区域使用文本。

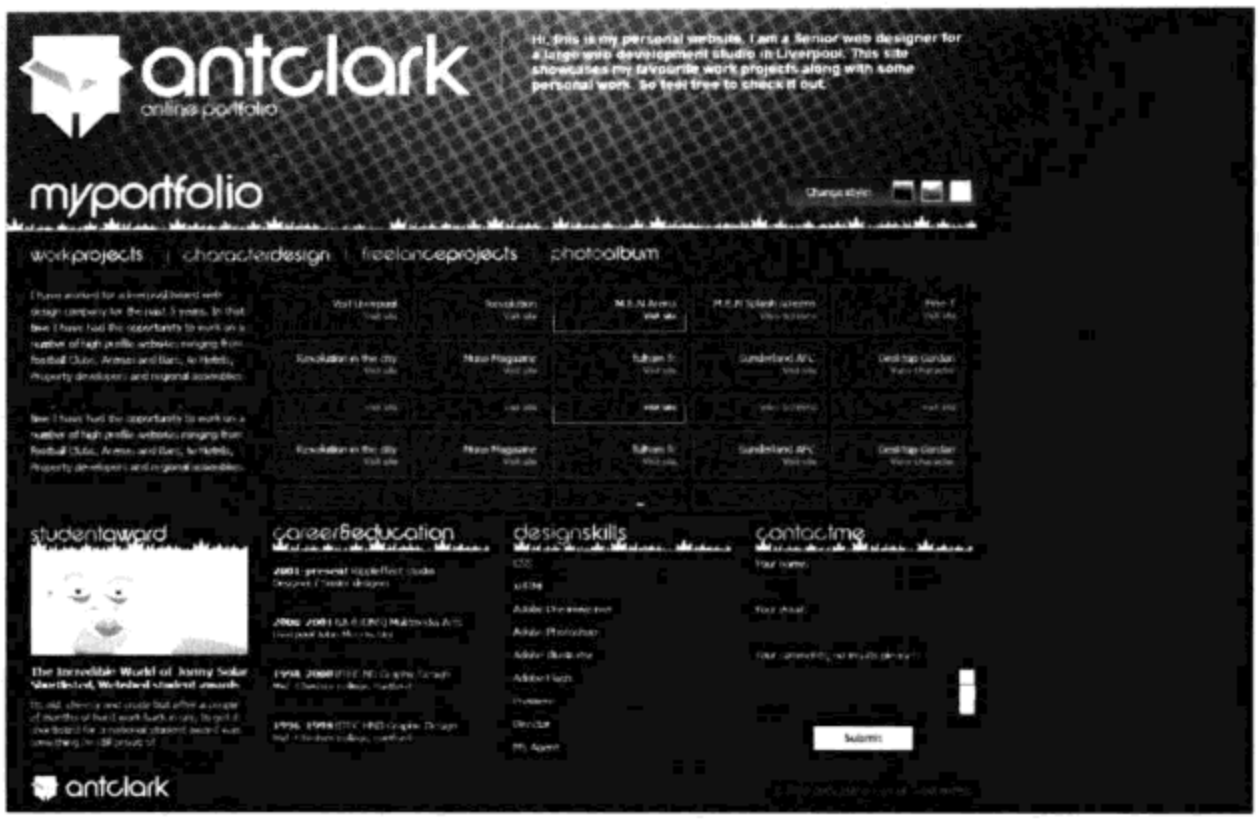


图 6-10 使用了文本 banner 的网页

图 6-11 包含文本 banner 的页面顶部结构图

本节介绍如何使用 CSS 样式实现包含段落文本 banner 的页面顶部效果。关键的方法是利用浮动属性 float 实现页面顶部各个成员的布局定位。

本实例的页面顶部分成了 3 个部分：logo 部分、快捷方式部分和段落文本 banner 部分，基本元素由一张图片、一张背景图、文字链接和一段文本组成，制作的结构图如图 6-11 所示。

下面分别来讲解制作包含文本的 banner 各个部分的过程。

6.3.1 图片 logo 的定位

先制作图片 logo 的定位方式，使用到的 CSS 属性是浮动属性。下面分别讲解浮动定位 float 属性的使用方法以及在本实例中的应用方法。

1. 浮动属性 float

当为一个元素设置了 float 属性时，这个元素就会成为一个区块级盒子。这个盒子能在水平线向左或向右移动，直到它的边缘接触到容器区块的边缘或另一个设置了 float 属性的元素边缘。如果在水平方向没有足够的空间，设置了 float 属性的盒子将会逐行向下移动，直到有足够的水平空间容纳它为止。浮动属性 float 语法结构如下。

```
float: left | right | none
```

其中各个属性值的含义如下。

- left：文本或图像会移至父元素中的左侧。

- right：文本或图像会移至父元素中的右侧。
- none：默认值，文本或图像会显示于它在文档中出现的位置。

在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。一个设置了 float 属性的元素会根据正常流布局中的位置，然后移出正常流并移动到左边或右边。一个设置了 float:left 的盒子，其后面的内容会流向它的右边（right）和下面。一个设置了 float:right 的盒子，其后面的内容能流向它的左边（left）和下面。

浮动并非替换元素，要指定给元素一个明确的宽度；否则，它们会尽可能地窄。在源代码顺序中，在 float 元素前面的区块级元素不受其影响，然而在它后面的元素将受其影响。当不需要在 float 元素后面的元素环绕它时，可以设置 clear 属性清除（clear 属性将在后面详细讲解）。

注意：

假如在一行之上没有足够的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。

可以通过下面图像和段落混排形象化的图示，说明 float 属性的工作原理，如图 6-12 所示。

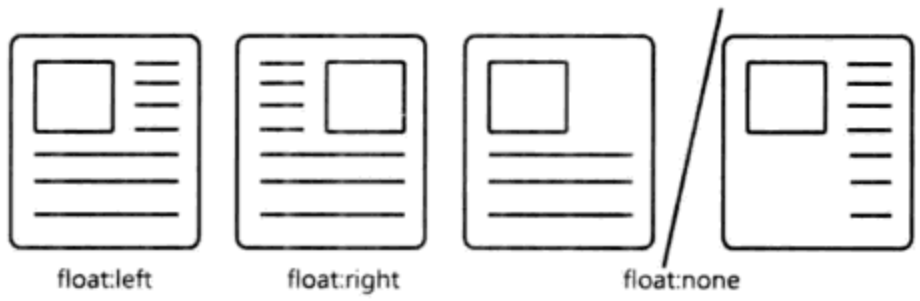


图 6-12　float 属性示意图

需要注意的是，float 属性并不如想象的那么简单，不是通过一个简单图示，就能明白它的工作原理，需要在实践中不断地总结经验，应对所出现的问题。下面就将使用 float 属性，制作简单的 CSS 布局设计。

2. 图片 logo 在页面顶部的定位

使用浮动属性，定位 logo 图片的位置。

（1）制作页面的 XHTML 代码，如下。

```
<div id="logo"><img src="images/6_3_01.jpg" /></div>
```

网页顶部在结构上，使用 div 标签分成了 logo 和 banner 区域，以 id="logo"和 id="banner"区分。首先，logo 区域放置 logo 图片，使用 img 标签定义图片路径，logo 图片如图 6-13 所示。

（2）制作#logo 元素的 CSS 样式，代码如下。

```
#logo{
    float: left;                                    /*设置向左浮动定位*/
}
```

logo 使用了浮动 float 属性。#logo 样式中的 float:left 声明是用来把 logo 区域定位到左侧，其预览效果如图 6-14 所示。

图 6-13　logo 图片

图 6-14　制作 logo 预览效果

6.3.2　定义快捷方式的文本样式

接下来制作 banner 区域的快捷方式文本，所需要的 CSS 属性是文本修饰属性。下面分别讲解文本修饰属性的使用方法以及具体应用。

1．文本修饰属性语法结构

文本修饰属性 text-decoration 语法结构如下。

```
text-decoration: none | underline | overline | line-throught
```

这个属性允许对文本设置某种效果，例如加下划线。如果子元素没有自己的装饰，父级元素设置的样式会继承到当前元素中，它的可选值如下。

- none：默认值，定义标准的文本。
- underline：定义文本下的一条线。
- overline：定义文本上的一条线。
- line-through：定义穿过文本下的一条线。

用图示的效果表示，如图 6-15 所示。

这里是overline属性值
这里是line-through属性值
这里是underline属性值
这里是none属性值

图 6-15　text-decoration 属性的属性值效果

下面将使用 text-decoration 属性，制作链接文字的效果。

2．制作页面顶部的快捷方式

这个实例主要使用文本修饰属性，对链接文字进行样式定义。

（1）制作页面的 XHTML 代码如下。

```
<div id="banner">
    <div class="base">
        <a href="#">web site</a>  |
        <a href="#">contact us</a>  |
        <a href="#">homepage</a>
```

```
        </div>
        <!-info 区域-->
    </div>
```

banner 区域分成了 base 和 info 区域。base 区域使用了 3 个文字链接，用于快速链接到网站的相关页面。info 区域就是网站的文字 banner 了。

（2）制作#banner 的 CSS 样式，代码如下。

```
#banner{
    background: url(images/6_3_02.jpg);                    /*设置背景图片*/
    width: 645px;                                          /*设置宽度*/
    height: 184px;                                         /*设置高度*/
    float: left;                                           /*设置向左浮动定位*/
}
```

在以上代码中，float:left 声明用来把 banner 区域定位到左侧，因为 logo 区域中图片的宽度小于页面宽度，这样就在 logo 区域右侧留出了剩余的空间，才能够使得 banner 区域得以显示。然后，给#banner 区域定义了背景图片，其中背景图片的显示效果如图 6-16 所示。

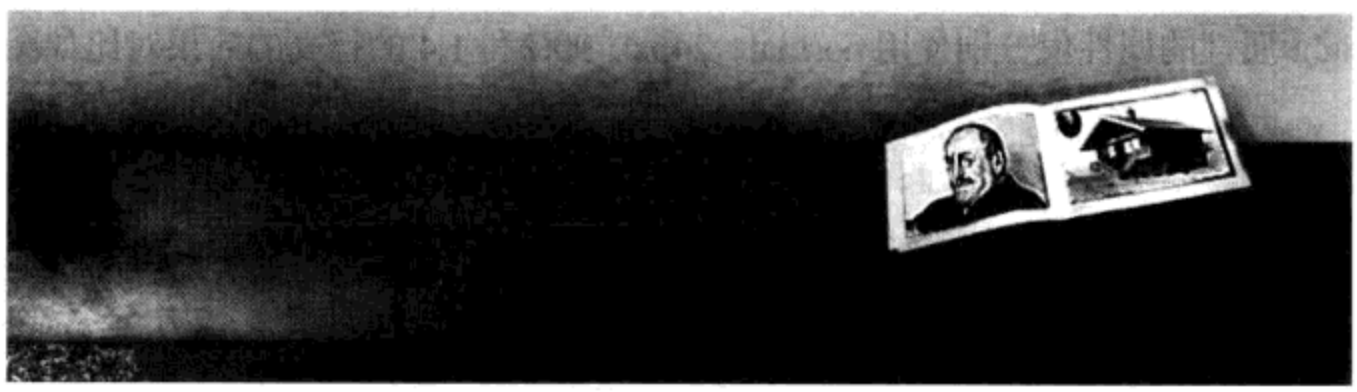


图 6-16　#banner 部分的背景图片

接下来，设置了宽度和高度，这个宽度和高度就是背景图片的宽度和高度，特别指定是为了能够完整的显示背景图片。

（3）制作快捷方式区域的 CSS 样式，代码如下。

```
.base{
    margin-top: 10px;                  /*设置上外边距*/
}
.base a{
    color: #6F3E31;                    /*设置连接文字的颜色*/
    text-decoration: none;             /*设置连接文字的文本样式为默认状态下 none，取消缺省的下划线效果*/
}
.base a:hover{
    text-decoration: underline; /*设置链接文字的文本样式为光标经过时下划线*/
}
```

在以上代码中，使用类选择符.base 指定了快速链接导航的上边距 10px。使用伪类 a 指定了链接样式。首先.base a{}定义了默认状态的字体颜色#6F3E31 以及去除了链接文字的下划线，然后.base a:hover{}定义了光标移上时链接文字的下划线。

6.3.3　定义段落文本 banner 样式

最后，制作段落文本的 banner，所需用到的 CSS 属性是字体行高属性。下面分别讲解字体行高属性的使用方法和具体实例应用。

1．字体行高属性

字体行高属性 line-height 用来设置对象的行高，即字体最底端与字体内部顶端之间的距离，其语法结构如下。

```
line-height: normal | length | number
```

其中各个属性值的含义如下。

- normal：默认值，设置合理的行间距。
- length：设置固定的行间距。
- 百分比：基于当前字体尺寸的百分比行间距。
- number：设置数字，此数字会与当前的字体尺寸相乘来设置行间距。

line-height 属性的属性值分别使用 normal、24px、90%和 1.4 的行高的效果对比如图 6-17 所示。

这个段落使用了行高为normal默认高度。这个段落使用了行高为normal默认高度。这个段落使用了行高为normal默认高度。这个段落使用了行高为normal默认高度。这个段落使用了行高为normal默认高度。

这个段落使用了行高为24px的length类型。这个段落使用了行高为24px的length类型。这个段落使用了行高为24px的length类型。这个段落使用了行高为24px的length类型。

这个段落使用了行高为90%的%类型。这个段落使用了行高为90%的%类型。这个段落使用了行高为90%的%类型。这个段落使用了行高为90%的%类型。这个段落使用了行高为90%的%类型。

这个段落使用了行高为1.4的number类型。这个段落使用了行高为1.4的number类型。这个段落使用了行高为1.4的number类型。这个段落使用了行高为1.4的number类型。

图 6-17　line-height 属性的各属性值对比效果

2．制作包含段落文本 banner 的页面顶部

对文本进行排版，主要使用行高属性定义字体的行高，XHTML 代码如下。

```
<div id="banner">
    <!—base 区域-->
    <div class="info">The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. </div>
</div>
```

为上面的 XHTML 结构代码指定样式，CSS 代码如下。

```
.info{
    color: #FFFFFF;                    /*设置 class 为 info 标签内的文本颜色*/
    font-size: 12px;                   /*设置字号*/
    margin-top: 40px;                  /*设置上外边距*/
    width: 400px;                      /*设置宽度*/
```

```
    line-height: 18px;                                    /*设置行高*/
}
```

以上代码指定了文本 banner 的样式。首先，定义了字体颜色#FFFFFF；然后，定义了字号 12px；接下来，定义了上边距 40px；最后，定义了 info 区域的宽度 400px 以及字体行高属性 18px。整个实例的最终效果如图 6-18 所示。

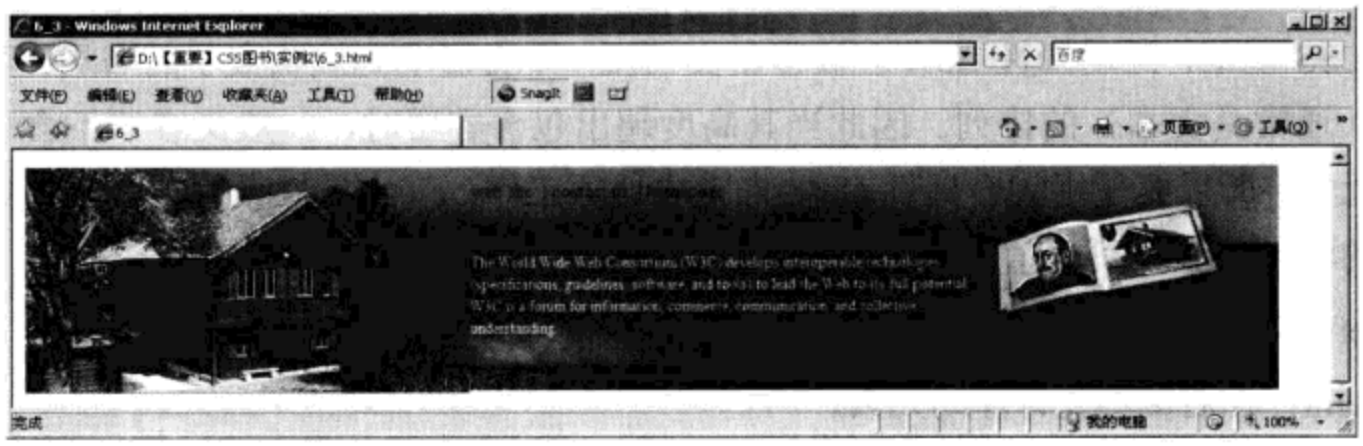


图 6-18　包含文字 banner 的页面顶部

6.4　制作包含图像 banner 的页面顶部

借助图片 banner 的设计，网页顶部在视觉外观上能够表现得更理想，更符合网站需要，如图 6-19 所示。

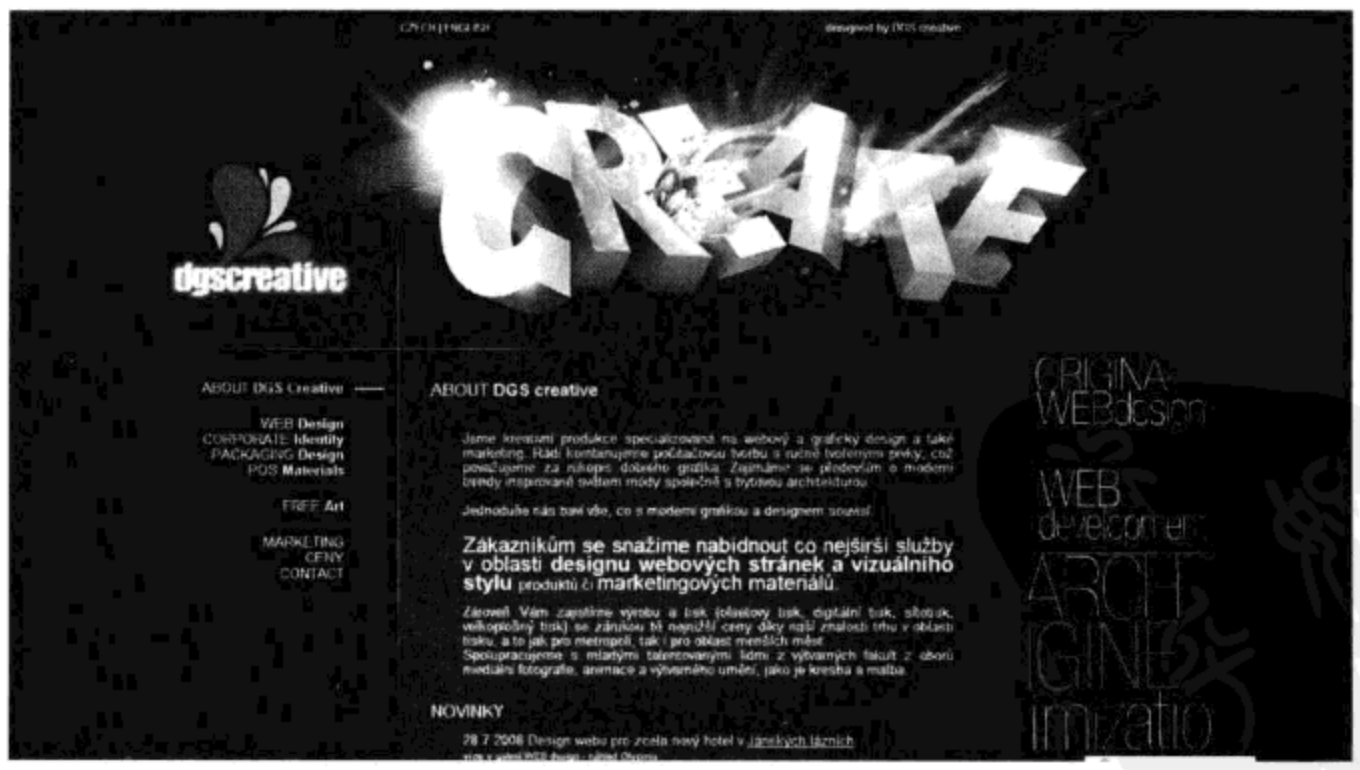


图 6-19　带有图片 banner 的网页

本节介绍使用 CSS 浮动属性样式控制图片、文本区域组合实现多行的页面顶部布局，关键的技巧在于控制多个浮动层的位置。下面介绍一下需要使用到的 CSS 重要属性。

1．闭合浮动属性语法结构

在网页设计中，经常需要设置多个 div 并列排列，往往是使用浮动属性 float:left 或 float:right

来实现。但这样问题就出现了，当前面并列的多个 div 总宽度不足 100%，下面的 div 就很可能向上提，和上一行并列的 div 在同一行，这不是想要的结果。使用闭合浮动属性 clear 属性正好可以解决这一问题。

闭合浮动属性 clear 语法结构如下。

```
clear: left | right | both | none
```

按照 CSS 规范，浮动属性 float 会被移出文档流，不会影响到块状盒子的布局，而只会影响内联盒子（通常是文本）的排列。因此当其高度超出包含容器时，一般父容器不会自动伸长以闭合浮动元素，所以要使用闭合属性 clear 来完成这项工作。clear 属性设置一个元素的侧面是否允许其他的浮动元素，可选值如下。

- left：在左侧不允许浮动元素。
- right：在右侧不允许浮动元素。
- both：在左右两侧均不允许浮动元素。
- none：默认允许浮动元素出现在两侧。

下面列举一个简单的示例，使用 float 浮动属性设置图片的定位方式为自动向左浮动，并且数次排列，直到浏览器的边缘再换行排列，如图 6-20 所示。

图 6-20　图片使用 float 属性浮动定位顺次排列效果

此效果的 XHTML 代码如下。

```
<div><img src="images/6_01.jpg" /></div>
<div><img src="images/6_02.jpg" /></div>
<div><img src="images/6_03.jpg" /></div>
<div><img src="images/6_04.jpg" /></div>
<div><img src="images/6_05.jpg" /></div>
<div><img src="images/6_06.jpg" /></div>
<div><img src="images/6_07.jpg" /></div>
<div><img src="images/6_08.jpg" /></div>
```

CSS 代码如下。

```
div{
    float: left;                                    /*设置 div 浮动属性*/
    margin: 8px;                                    /*设置外边距*/
}
```

现在修改 CSS 代码，在 div 选择符中添加属性 clear:both 如下。

```
div{
    clear: both;                                    /*禁止 div 两侧出现浮动元素*/
    float: left;                                    /*设置 div 的浮动属性*/
    margin: 8px;                                    /*设置外边距*/
}
```

然后再使用浏览器预览效果，如图 6-21 所示。

从图 6-21 可以看出，顺次向左浮动的图片现在变成了一行一个图片地竖直排列，这都是因为 clear:both 属性的作用，它使得 div 容器两侧禁止浮动。下面就将利用这一属性的特点，制作 banner 复杂的布局。

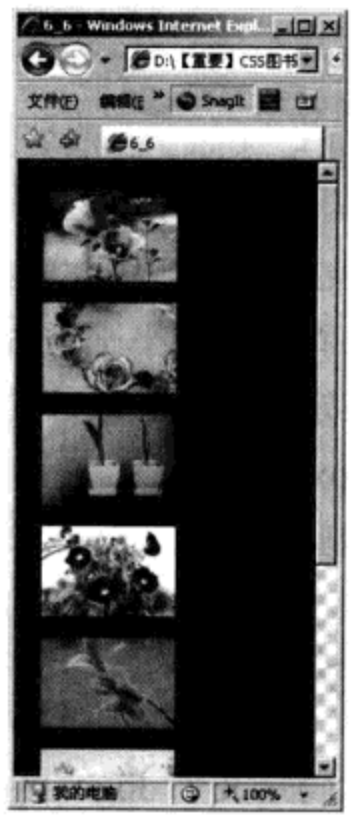


图 6-21　添加了 clear 属性后的效果

div 就像一个容器，这个属性用来控制其间图片的位置。

2. 制作包含图像 banner 的页面顶部实例

大多数网站都会在页面的顶部设计一张宣传网站本身的 banner 图片，使浏览者看到 banner 图片一目了然了解网站的核心内容。现在就来制作一个包含有图片 banner 的页面顶部实例，页面结构分为 3 个部分：logo、base 和 banner。

（1）制作页面的 XHTML 代码，如下。

```
<div id="top">
    <div class="logo"><img src="images/8_4_02.jpg" /></div>
    <div class="base">
        <a href="#">web site</a>  |
        <a href="#">contact us</a>  |
        <a href="#">homepage</a>
    </div>
    <div class="clearit"></div>
    <div class="banner"><img src="images/8_4_01.jpg" /></div>
</div>
```

XHTML 代码中各部分出现的顺序是非常重要的。logo 区域和 base 区域的 div 必须在 banner 区域之前出现。这样才可以让这两个边栏浮动到它们的位置上（屏幕两侧），并让 banner 的内容浮动在它们之下。

图 6-22 logo 图片

页面顶部用一个 id="top"的大容器 div 来装载，其里面的 3 部分 logo、base 和 banner 都使用浮动属性定位。logo 和 banner 都是使用的图片，如图 6-22 和图 6-23 所示的 base 是 3 个快速导航的文本链接。

图 6-23 banner 图片

（2）制作#top 元素的 CSS 样式，代码如下。

```
#top{
    background: url(images/6_4_03.jpg) repeat-x;  /*设置背景图片，并且背景图片横向平铺*/
    height: 178px;                                /*设置高度*/
    width: 100%;                                  /*设置宽度*/
}
```

首先，在以上代码中，定义了页面顶部的背景图片，背景图如图 6-24 所示。背景图水平平铺占据整个顶部区域。

然后，定义了页面顶部的宽度 100%，占满整个浏览器宽度，以及高度 178px。

图 6-24 #top 区域

（3）制作.logo 元素的 CSS 样式，代码如下。

```
.logo{
    float: left;        /*设置 logo 向左浮动定位*/
}
```

以上代码定义了 logo 区域的浮动方式为左侧浮动，用来把 logo 部分定位到左侧。

（4）制作快捷方式区域的 CSS 样式，代码如下。

```
.base{
    margin-top: 10px;                 /*设置 class 为 base 的标签元素上外边距*/
    float: right;                     /*设置向右浮动定位*/
    color: #E2E4E5;                   /*设置字体颜色*/
    margin-right: 40px;               /*设置右外边距*/
}
.base a{
    color: #E2E4E5;                   /*设置 class 为 base 的标签链接文字颜色*/
    text-decoration: none;            /*设置 class 为 base 的标签链接文字的样式*/
}
.base a:hover{
    text-decoration: underline;       /*设置 class 为 base 的标签光标经过链接文字下划线效果*/
}
```

首先，在.base 选择符中，使用 float 属性把 base 区域的 div 定位到右侧，float 把 logo 区域和 base 区域完全定位到浏览器窗口的左边缘和右边缘。然而这些 div 被 div#top 包含，那么 float 将会把它们定位到 div#top 的边缘。

然后，定义了 base 区域的上边距 10px 和右边距 40px，最后又定义了字体颜色#E2E4E5。接下来，定义 base 区域快捷方式的链接字体样式，思路与 6.3.3 节相类似。

（5）制作 banner 图片区域的 CSS 样式，代码如下。

```
.clearit{
    clear: both;                          /*设置元素的浮动清空属性*/
}
```

这是最关键的步骤。在.clearit 选择符中，使用 clear 属性指定了禁止 banner 区域的内容“流淌”在两个边栏之间。这样，虽然 logo 区域和 base 区域向左右浮动过程中产生了一些空间，但是由于 clear: both 的加入，不允许浮动元素出现在 div.clear 层的两侧，这样前面的浮动对象就不再起作用了，也就是以后的浮动对象重新建立浮动属性关系。

这样，包含图像 banner 的页面顶部就制作完成，预览效果如图 6-25 所示。

图 6-25　包含图像 banner 的页面顶部

6.5　小结

本章讲解使用 CSS 制作网页的顶部内容实例，包括制作文本 logo 的页面顶部、制作包含图像 logo 的页面顶部、包含文本 banner 的页面顶部以及包含图像 banner 的页面顶部。制作实例的过程中，涉及以下 5 个 CSS 属性。

- 外边距属性 margin。
- 浮动属性 float。
- 文本修饰属性 text-decoration。
- 字体行高属性 line-height。
- 闭合浮动属性 clear。

对本章的知识点前后联系进行归纳总结，结构导图如图 6-26 所示。

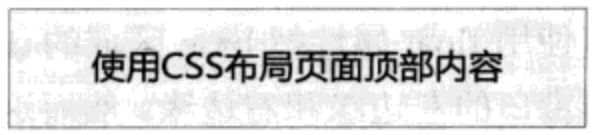


图 6-26　本章知识点结构导图

6.6　习题

1. 简述网站 logo 的含义。
2. 简述 banner 的含义。
3. 简述 CSS 浮动属性 float 的语法结构及作用。
4. 简述文本修饰属性的使用方法。
5. 如何定义段落文本 banner 样式？

第7章 使用CSS制作网站导航

网站导航（Navigation）是一个通称，方便用户浏览网站信息和获取网站服务。它的基本作用就是让用户在浏览网站过程中不致迷失，并且可以方便地回到网站首页以及其他相关内容的页面。

网站导航一般位于网站的 banner 下面或是网页的顶部。网站导航是网站中最重要的元素，主要有横向导航、纵向导航、下拉导航及多级导航等几种形式。本章将介绍如何使用 CSS 来制作各种美观实用的网站导航。

7.1 制作一个简单的横向文字导航条

门户网站的主导航一般都是选用横向导航的形式。这是因为门户网站下方文字较多，而且每个频道又都有不同的样式区别，因此在顶部固定一个区域，设计统一风格又不占用过多空间的横向导航是最理想的选择。例如 apple.com、times.com 及国内的新浪、网易、腾讯等网站都是采用此种导航形式。如图 7-1 所示是腾讯网首页的横向文字导航条。

图 7-1 腾讯网首页的横向文字导航条

本节介绍制作横向文字导航条的方法，使用 ul、li 列表元素罗列导航项，默认情况下，呈现的效果应该是竖向排列，现在使用 CSS 的显示元素属性 display 改变排列的方向，使之呈现出横向排列的效果。下面先介绍 display 属性的使用方法。

1. display 属性

当标签显示的时候，以哪种方式显示，也可以通过 CSS 样式来定义。CSS 中的 display 属性就是负责实现这样的功能，语法格式如下。

```
display: block | inline | none;
```

其实 display 的可用属性值有很多种，这里面只列举了网页设计中常用到的 3 种属性值。这 3 种属性值的使用方法如下。

- block：默认值，此元素将显示为块级元素，元素前后会带有换行符。
- inline：此元素将显示为内联元素，元素前后没有换行符。
- none：隐藏对象，不为被隐藏的对象保留其空间。

下面使用一段代码来解释这 3 个属性值的区别，代码如下。

```
<div class="set1">块级元素 1</div>
<div class="set1">块级元素 2</div>
<div class="set2">内联元素 1</div>
<div class="set2">内联元素 2</div>
<div class="set3">这里的代码将不会显示</div>
```

使用 display 属性的 3 个属性值，分别为这些 div 标签设置相应的 CSS 样式，代码如下。

```
.set1{
    display:block;
}
.set2{
    display: inline;
}
.set3{
    display: none;
}
```

在以上代码中，为选择符.set1 设置了块级元素显示方式，为选择符.set2 设置了内联元素显示方式，为选择符.set3 设置了隐藏对象的显示方式。这样，在 CSS 样式的作用下，不同样式的 div 将显示不同的结果，在浏览器中的预览效果如图 7-2 所示。

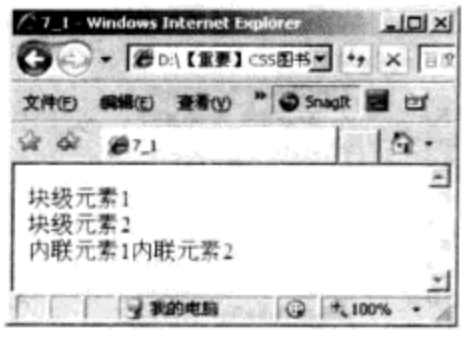


图 7-2 display 属性的 3 个属性值的效果

从图 7-2 可以看出，属性设置为块级元素的对象，按照块级元素的显示方式单独占据一整行显示；属性设置为内联元素的对象，将按照内联元素的显示方式并排显示；而属性设置为 none 的对象，将在浏览器中不显示这个对象。

2. list-style-type 属性

list-style-type 属性是列表属性中的一种，它用来设置列表项目符号的类型。list-style-type 属性的语法结构如下。

```
list-style-type: none | disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman
```

list-style-type 属性值有很多种，这里面只介绍一些常用的属性值，含义如下。

- none：无标记。
- disc：默认值，标记是实心圆。
- circle：标记是空心圆。
- square：标记是实心方块。
- decimal：标记是数字。
- decimal-leading-zero：0 开头的数字标记（例如 01、02、03 等）。
- 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）。

其实，list-style-type 属性就是定义列表项的编号方式，采用哪种编号方式，取决于列表信息的内容、性质等。

3. 制作横向文字导航条的实例

本实例是制作一个简单的只有文字的横向导航条，因此需要用到的标签只有 ul、li 和 a，通过 CSS 对列表显示元素属性的控制，实现横向导航的效果。

（1）制作页面的 XHTML 代码。

```
<ul id="navlist">
    <li><a href="#">Item one</a></li>
    <li><a href="#">Item two</a></li>
    <li><a href="#">Item three</a></li>
    <li><a href="#">Item four</a></li>
    <li><a href="#">Item five</a></li>
</ul>
```

在以上代码中，使用的是前面讲过的
列表元素标签来设计导航条。导航系统也是一种列表，可以理解为导航列表，每一个列表数据就是导航中的一个导航频道，同样也可以使用二层嵌套的 div 来实现一个导航条的代码结构，但是相对于 ul 列表来说，div 显得过于复杂。ul 的使用还是应该放在块状形区域，对于简单的文字导航来说，使用 ul 就更为灵活方便了。

在 XHTML 代码中，为 ul 定义了一个 id 为 navlist 的类，目的是使用 CSS 样式表通过类选择符控制 ul 的样式。

（2）制作#navlist 元素的 CSS 样式，代码如下。

```
#navlist li{
    display: inline;                                    /*设置内联元素显示方式*/
```

```
        list-style-type: none;                                    /*设置列表项目符号*/
    }
```

在以上代码中，把样式编写给了ul下的li元素。导航的关键在于对li元素的样式定义。display: inline是这段代码的重点，它使得链接对象的显示方式由默认的以块状元素垂直排列改为以内联元素水平排列，实现了横向文字导航的效果。list-style-type: none取消了li默认的所有列表样式。

至此，只有文字的横向导航条制作完成，预览效果如图7-3所示。

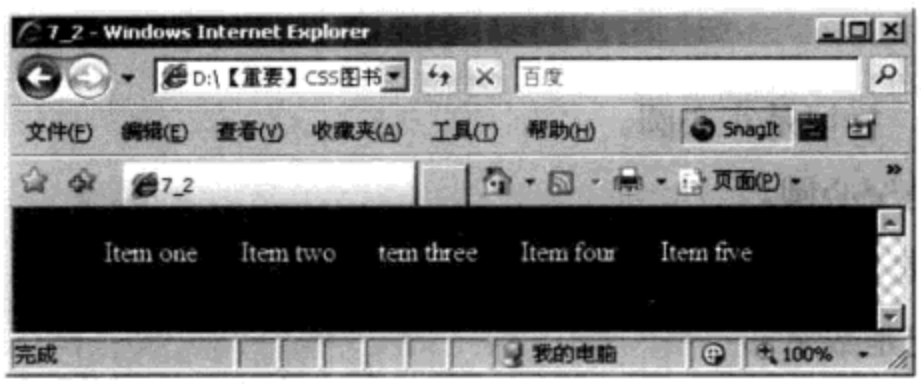


图7-3　横向文字导航条的预览效果

7.2　制作方块导航条

设计师天马行空的进行网页设计，导航条自然也不拘泥于一个简单的文字链接形式。方块导航条可以为设计师提供一些设计灵感。如图7-4所示是一个使用方块导航条的网页。

图7-4　使用方块导航条的网页

制作方块导航条，先制作一个最基本的水平导航条，然后在a元素的样式中使用background-color属性将其定义成有颜色的区块，最后使用a:hover达到当光标经过时改变其颜色的效果。下面分别讲解文本对齐属性的使用方法和具体实例应用。

1．文本对齐属性

网页设计如同任何一种视觉设计，都需要对文本排版进行处理。CSS样式表对文本的处理

也有特定的属性来实现。文本对齐属性 text-align 就是其中很重要的一个属性。文本对齐属性 text-align 能够对齐元素中的文本，其语法结构如下。

```
text-align: left | right | center | justify
```

其中各个属性值的含义如下。

- left：把文本排列到左边。它是默认值，由浏览器决定。
- right：把文本排列到右边。
- center：把文本排列到中间。
- justify：对齐每行的文字。

该属性通过指定行框与哪个点对齐，从而设置块级元素内文本的水平对齐方式。通过允许用户代理调整行内容中字母和字之间的间隔，可以支持值 justify，不同用户代理可能会得到不同的结果。以一种形象化的方式表现这 4 种属性值所达到的效果，如图 7-5 所示。

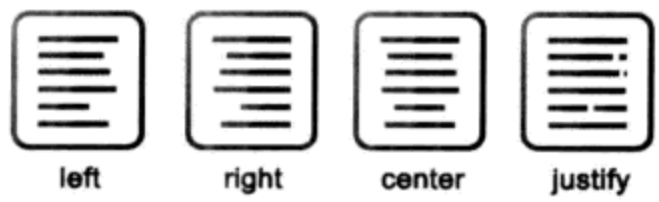


图 7-5　text-align 属性的 4 种属性值实现效果的形象化图示

2. 制作方块导航条实例

制作方块导航条主要的方法是把导航项定义为块状元素，通过指定块状元素的背景色，实现具有方块效果的导航系统。

（1）制作页面的 XHTML 代码，代码如下。

```
<ul id="navlist">
    <li><a href="#">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考 </a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">论坛</a></li>
    <li><a href="#">联系</a></li>
</ul>
```

代码结构与 7.1 节实例是相似的，不同的是 CSS 样式。

（2）制作#navlist li 元素的 CSS 样式，代码如下。

```
#navlist li{
    float: left;
}
```

在以上代码中，ul#navlist li 定义了 li 元素的浮动方式为左浮动。通过浮动定位原理，使得自身向左浮动，从而使下一个对象贴近该对象的右边，最终所有的 li 都具有了向左浮动的特性，从而形成了横向的排列形式。

（3）制作#navlist li a 元素的 CSS 样式，代码如下。

```
#navlist li a{
    color: #000000;                          /*设置链接文字颜色*/
    text-decoration: none;                   /*设置链接文字修饰样式*/
    display: block;                          /*设置块级元素显示方式*/
    padding-top: 4px;                        /*设置内上边距*/
    width: 97px;                             /*设置宽度*/
    height: 22px;                            /*设置高度*/
    text-align: center;                      /*设置文本居中*/
    background-color: #ECECEC;               /*设置背景色*/
    margin-left: 2px;                        /*设置左外边距*/
}
```

本步骤是制作本实例的关键，是使用伪类 a 对其进行链接样式控制的。

首先，#navlist li a{}定义了链接文字的颜色#000000，文本修饰样式 text-decoration: none 去除了默认的下划线。

然后，display: block 是样式的重点，它使得 a 链接对象的显示方式由一段文本改为一个块状对象。这和 div 特性相同，前面讲过 div 默认状态是一个块状对象，而默认状态下的 a 链接对象只是显示为一个普通文本，这样就没有办法使 a 链接对象能够模仿一个方块状按钮的样式。使用 display: block 属性，使 a 链接对象将能够像 div 和其他元素一样成为一个块状对象，这样就可以使用 CSS 的外边距、内边距、边框等属性给 a 链接对象标签加上一系列的样式。

接下来，指定了上内边距 4px 和左外边距 2px，以及 a 元素的宽度 97px、高度 22px。最后，定义了居中对齐文本 center。

（4）制作#navlist li a 元素“光标经过”状态的 CSS 样式，代码如下。

```
#navlist li a:hover{
    background-color: #BBBBBB;               /*设置背景色*/
    color: #FFFFFF;                          /*设置文字颜色*/
}
```

在以上代码中，通过 a:hover 属性样式为“光标滑过”添加了交互式响应。交互式的导航能够告诉用户哪里能点击、当前在哪里及哪里已经看过了。本实例当光标移到一个导航项的时候，导航背景变成深灰底白色，导航更友好。方块导航条实例制作完成，预览效果如图 7-6 所示。

图 7-6　方块导航条的预览效果

7.3　制作标签式导航

在简单的导航系统制作完成后，有必要再为导航增加更多丰富的可用性设计。淘宝网站的标签式导航如图 7-7 所示。

图 7-7　淘宝网站的标签式导航

它的导航采用的是类似文件夹标签的样式，这样的导航样式正是目前网站上常见的，既美观又能让用户非常方便地知道自己所处的位置。本节中也制作这样的一套导航系统。

1．边框样式属性

边框样式属性 border 的语法结构如下。

```
border: 边框宽度 | 样式 | 颜色
border-top: 边框宽度 | 样式 | 颜色
border-right: 边框宽度 | 样式 | 颜色
border-bottom: 边框宽度 | 样式 | 颜色
border-left: 边框宽度 | 样式 | 颜色
```

其中各个属性值的含义如下。

- 边框宽度：默认值为 medium，这个值没有明确定义，不过通常是两个像素。尽管如此，不一定能看到边框，原因是边框的默认值为 none，这样一来，就不会有边框了。如果边框没有样式，就不会存在。
- 样式：可用值如表 7-1 所示。表中有 10 种属性值，除 hidden 属性值外，其他属性值的实际效果如图 7-8 所示。

表 7-1　样式属性值可选值列表

可用值	描述
none	定义无边框
hidden	与“none”相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突
dotted	定义点状边框。在大多数浏览器中呈现为实线
dashed	定义虚线。在大多数浏览器中呈现为实线
solid	定义实线
double	定义双线。双线的宽度等于 border-width 的值
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值
inset	定义 3Dinset 边框。其效果取决于 border-color 的值
outset	定义 3Doutset 边框。其效果取决于 border-color 的值

- 颜色：默认值是元素本身的前景色。如果没有为边框声明颜色，它将与元素的文本颜色相同。另一方面，如果一个元素没有任何文本，那么该元素将继承其父元素的文本颜色。

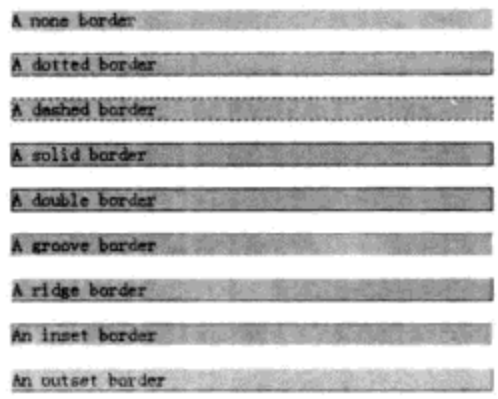


图 7-8　border 属性值的实际效果

2. 制作标签式导航实例

淘宝网在显示时，首页导航标签呈现出了与其他标签不同的颜色以提示用户所处的位置。

还是从 7.2 节的代码继续编写，为了让某一个导航项成为一个当前所属，这个导航项一定有一个与其他导航项不一样的背景色。7.2 节的代码是针对所有的 a 标签统一设置了背景，因此现在首要任务就是设计一个例外情况，即当前导航项。

（1）制作页面的 XHTML 代码，对 a 标签的属性进行了改进，代码如下。

```
<ul id="navlist">
    <li><a href="#" id="current">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">论坛</a></li>
    <li><a href="#">联系</a></li>
</ul>
```

在第一个<li>下的 a 标签中，添加了一个新的 id，名为 current，使“首页”导航项成为当前状态，现在为 current 这个 id 做一个颜色设计。

（2）制作#navlist 元素的 CSS 样式，代码如下。

```
#navlist{
    height: 26px;                                    /*设置高度*/
    border-bottom: 2px solid #2788DA;                /*设置下边框样式*/
}
```

在以上代码中，对 ul 标签编写了代码，给 ul 标签设置了高度 26px，并且底部加上了 2px 实线的边框，再次预览效果，已经和标签式导航大同小异。回到 navlist 元素的定义，border-bottom 是新加入的一个属性，指定了元素的下边框的设置，border-bottom 的参数这里指的是 2px 的宽度、单线样式、颜色为#2788DA。通过这样的设置，ul 标签就拥有了 2px 带颜色的下边框。

（3）制作处于当前状态的导航项的 CSS 样式，代码如下。

```
#navlist li a#current{
    background-color: #2788DA;                       /*设置背景色*/
```

```
    color: #FFFFFF;                                    /*设置文字颜色*/
}
```

在以上代码中，定义了处于当前状态的导航项的样式，文字颜色为#FFFFFF，背景颜色#2788DA 正好与导航系统的下边框颜色相同，这样简单的标签式导航系统就完成了。每当换一个导航项时，只需要将 id="current"移到当前导航项所在的 a 元素中，即可完成颜色切换，不需要重新编写颜色属性等样式，所有属性修改只需在 CSS 中完成。简易的标签导航系统制作完成，预览效果如图 7-9 所示。

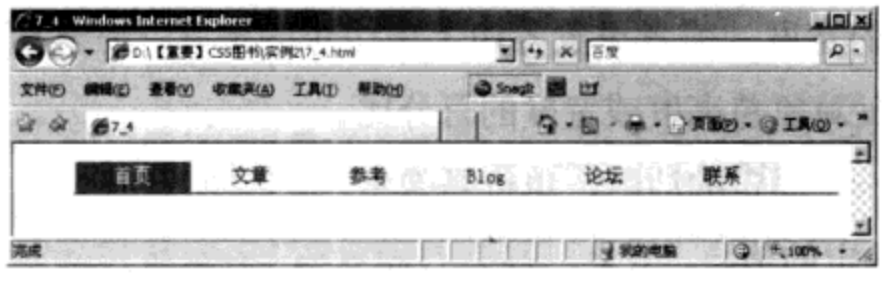


图 7-9　标签式导航的预览效果

注意：

标签式的导航样式编写中涉及了 CSS 中的一个非常有用的特性——继承。本实例中，在外层通过#navlist li a{}指定，对所有 id 为#navlist 下的 li 标签中的 a 对象设计统一的样式，a 无条件的全部优先实行#navlist a{}中的样式设计。

但是对其中的某一个 a 加了 id 叫 current，并对 current 设计了背景色，这个 id 为 current 在执行#navlist li a{}的样式基础上，再次执行#navlist li a#current{}的样式。不会因为指定了#navlist li a #current{}的样式设定而丢掉#navlist li a{}的设定，它在继承#navlist li a{}的基础上，还会执行属于自己的#navlist li a #current{}。

7.4　制作按钮导航条

目前在网页中普遍出现的按钮可以分为两类：一种是有提交功能的按钮；另一种是仅仅表示链接的按钮。网站导航可以使用按钮形式来进行设计，这样的按钮实现的是一种从一个页面链接到另一个页面的功能。

经典外观样式的 Windows 操作系统下，IE 7 浏览器的选项卡样式就是按钮形式，如图 7-10 所示。

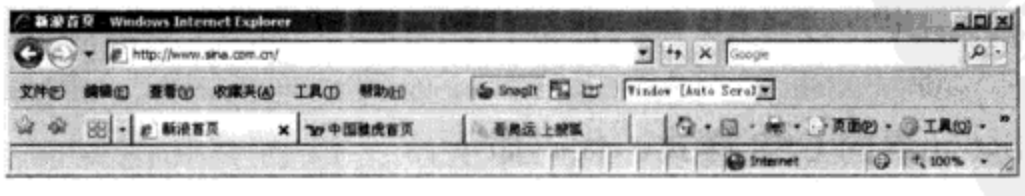


图 7-10　IE 7 浏览器的选项卡样式

1．内边距属性

CSS 的内边距属性 padding 是定义内容与边框（border）之间的空白的属性。此属性不允许使用负值，它可以分别设置上下左右 4 个方向的内边距。

内边距属性 padding 语法结构和外边距 margin 相似，代码如下。

```
padding: 上边距 | 右边距 | 下边距 | 左边距
```

缩写为：

```
padding: 上下边距 | 左右边距
```

或者

```
padding: 上边距 | 左右边距 | 下边距
```

padding 属性是用于在一个声明中设置 4 个内边距的所有属性的简写属性。这个简写属性设置一个元素所有内边距的宽度，或者设置各边上内边距的宽度，宽度值如下。

- 百分比：基于父对象总高度或宽度的百分比。
- 长度值：定义一个固定的边距，单位是像素。
- auto：浏览器设定的值。

可以对 padding 属性分开设置样式，语法结构如下。

```
padding-left: 边距宽度
padding-right: 边距宽度
padding-bottom: 边距宽度
padding-left: 边距宽度
```

内边距样式 padding 是一组控制元素“边框到其内部元素”之间距离的样式。内边距（padding）和外边距（margin）一样，是 CSS 块级元素中很重要的一员。需要注意的是，对象的宽度（width）和高度（height），并不包括它的内边距，也就是说，一个内边距为 10px（左右加起来就是 20px），宽度为 100px 的元素，实际宽度（在网页中所占空间）为 120px。

2. 制作按钮导航条实例

现在仿制系统按钮效果，制作一个导航项呈现按钮效果的导航系统。在 7.3 节实例的代码基础上进行改进。

（1）制作页面的 XHTML 代码，如下。

```
<ul id="navlist">
    <li id="active"><a href="#">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考</a></li>
    <li><a href="#">地址</a></li>
    <li><a href="#">论坛</a></li>
</ul>
```

（2）制作#navlist 元素的 CSS 样式，代码如下。

```
#navlist{
    list-style: none;                    /*设置列表符号样式*/
    padding: 0;                          /*设置内边距*/
    margin: 0 auto;                      /*设置外边距，实现对象居中*/
    width: 80%;                          /*设置宽度*/
```

```
    font-size: 12px;                                    /*设置字号*/
}
```

在以上代码中，定义内边距为 0px，使得背景颜色能够充满整个元素内部，不至于边缘出现空白；定义外边距 margin: 0 auto，它的作用是上下外边距为 0，左右边距自动适应，实际的效果就是元素垂直居中；定义宽度 width: 80%，将导航系统的宽度指定为浏览器宽度的 80%。

（3）制作#navlist li 元素的 CSS 样式，代码如下。

```
#navlist li{
    float: left;                                        /*设置向左浮动定位*/
    width: 15%                                          /*设置宽度*/
    margin: 0;                                          /*设置外边距*/
    padding: 0;                                         /*设置内边距*/
}
```

在以上代码中，依然是通过浮动定位原理，使得自身向左浮动，从而使下一个对象贴进该对象的右边，形成了横向排列的导航系统。同时通过 margin 和 padding 去除了由于浏览器不同而造成的边距，定义了列表项的宽度是导航系统的 15%，预览效果如图 7-11 所示。

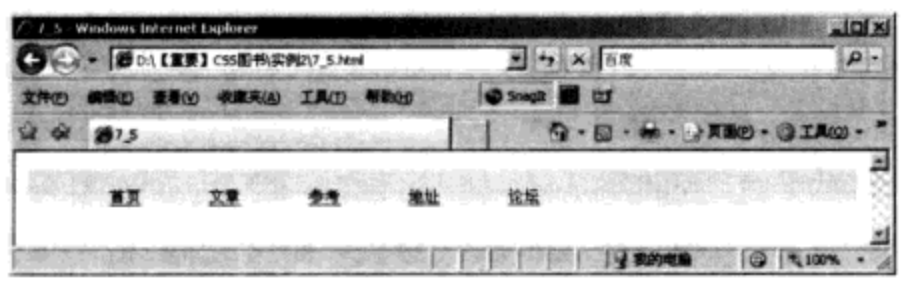


图 7-11　横向排列的预览效果

（4）制作#navlist li a 元素交互效果的 CSS 样式，代码如下。

```
#navlist li a{
    display: block;                                     /*设置为块级元素*/
    width: 100%;                                        /*设置宽度*/
    padding: 8px;                                       /*设置内边距*/
    border-width: 1px;                                  /*设置边框宽度*/
    border-color: #ffe #aaab9c #ccc #fff;               /*设置边框颜色*/
    border-style: solid;                                /*设置边框样式为实线*/
    color: #777;                                        /*设置文字颜色*/
    text-decoration: none;                              /*文本修饰属性，去除了下划线*/
    background-color: #f7f2ea;                          /*设置背景色*/
}
#navlist li#active a{
    background: #f0e7d7;                                /*区别于非当前状态按钮的背景颜色*/
    color: #800000;                                     /*文字颜色*/
}
#navlist li a:hover, #navlist li#active a:hover{
    color: #800000;                                     /*设置字体链接文字光标经过时的文字颜色*/
    border-color: #aaab9c #fff #fff #ccc;               /*设置边框颜色*/
}
```

在以上代码中，首先，在选择符#navlist li a 中，使用伪类的方法定义导航项的样式。定义了不同的边框颜色，模拟出一种按钮凸起的效果。由于样式复杂，所以边框 border 样式分成了

border-color、border-style 和 border-width 分别进行定义，使导航项具有高光边缘的效果，最后指定了导航项的背景颜色。在#navlist li a 选择符中，使用了 width: 100%继承了#navlist li 选择符中的 width: 15%。

然后，在选择符#navlist li#active a 中定义了处于当前状态的按钮样式，效果如图 7-12 所示。

图 7-12　按钮样式预览效果

最后，也是实现按钮交互效果的关键，使用群组选择符#navlist li a:hover, #navlist li#active a:hover 定义“光标滑过”状态时候的样式。与非当前状态时候的按钮比较，改变了边框的颜色，使之呈现出按下去的光线效果，背景颜色也随之调暗。按钮导航条制作完成，预览效果如图 7-13 所示。

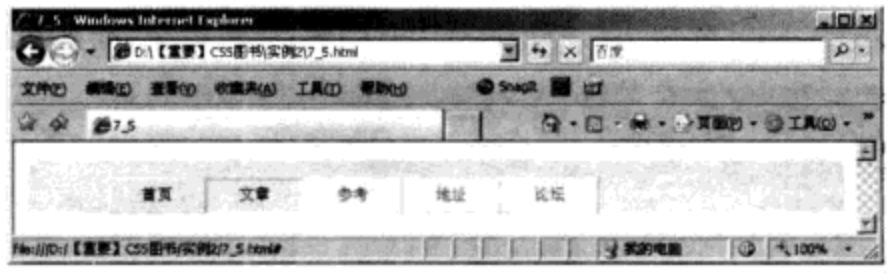


图 7-13　按钮导航条的预览效果

7.5　CSS 盒模型及盒模型 hack

CSS 盒模型（Box Model）是从 CSS 诞生之时便产生的一个概念，对于网页中的大部分对象，实际呈现的形式都是一个盒子形状对象，即块状对象。在 CSS 设计过程中，要做的就是安排一个个盒子的内容。

- 盒子的尺寸与类型：包含盒子的具体大小、边框及 display 的类型。
- 盒子的布局：盒子中内容的流动方式、自身的浮动方式及是否绝对定位等。
- 与其他元素的关系：两个盒对象排列时的边距累加及样式中的继承关系等。

而在这些盒对象的设定内容中，最基本也是最容易让初学者感到迷惑的就是盒对象尺寸的问题。许多初学者往往在没有完全了解盒对象的宽高计算方法的情况下，直接进行网站设计，结果发现最终的效果总是不符合自己的思路。本节将探讨盒模型的尺寸与占位的标准计算方法。

7.5.1　盒模型尺寸

关于盒模型尺寸的问题，先看如图 7-14 所示的示例图（TM 表示 margin-top、TB 表示 border-top、TP 表示 padding-top，其他以此类推）。

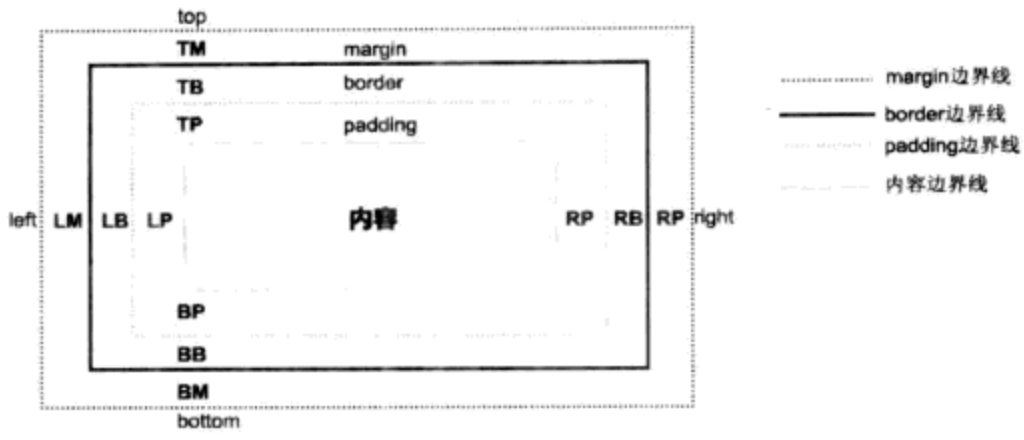


图 7-14　盒模型尺寸示例图

如图 7-14 所示为一个完整的盒对象，由众多属性构成，每个属性在实际的盒对象显示中占据不同的位置。对于盒对象而言，除去用于 position 的 top、left、bottom 和 right 属性，还有如下几点。

- margin 的值将占据 margin 边界线到 border 边界线之间的区域。
- border 的值将占据 border 边界线到 padding 边界线之间的区域。
- padding 的值将占据 padding 边界线到内容边界线之间的区域。

内容的宽和高由 width 和 height 属性来定义，因此在设定几个属性值后，最终整个盒对象的显示尺寸应当如图 7-15 所示。

由图 7-15 所示的各个属性值构成的盒对象，最终显示宽度并非 width 的值 500px，而是众多属性之和。由图 7-15 可以得到一个盒对象的真实宽度计算公式。

```
真实宽度 = margin-lef + border-left + padding-left + width + padding-right + border-right + margin-right
```

对于高度 height 值，也采用同样的计算方法，计算上下所有的属性值之和。因此在设计时，对于宽度及高度的问题，一定要对所有盒对象的 margin、padding 和 border 等属性进行相加。而在实际应用中，有时候会碰到两个盒对象并排或上下排列的情况，这时两个对象之间的间距，实际上是由两个对象的这些参数共同控制的，如图 7-16 所示。

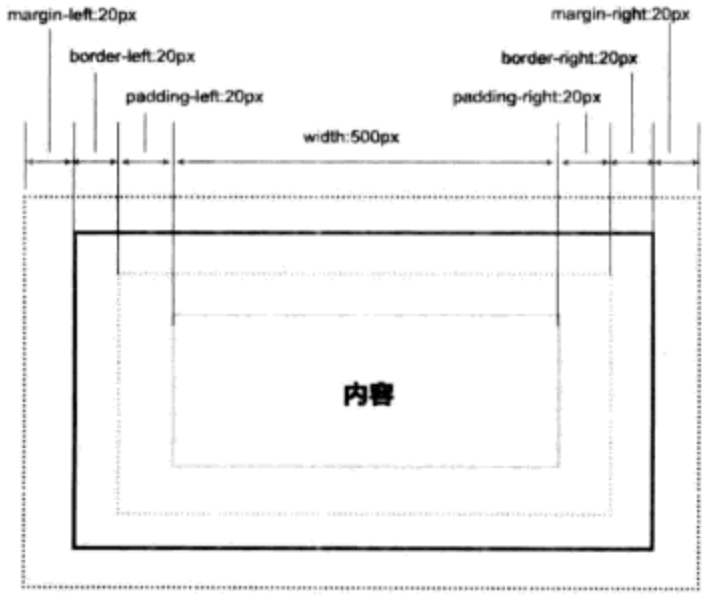


图 7-15　盒模型对象的显示尺寸

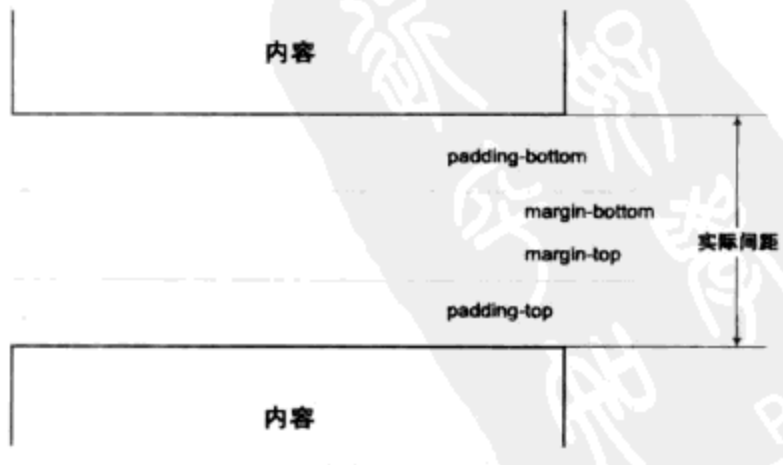


图 7-16　盒模型的间距图示

当两个盒对象上下排列，而且 border 值为 0 时，它们之间的实际间距由上面对象的 padding-bottom 和 margin-bottom 加上下面对象的 margin-top 和 padding-top 的和组成。通过这样的计算，既可以解决边距问题，还可以灵活地合并两个对象的边距，如使用上对象与下对象都采用 1px 的 margin 而形成两个对象间距 2px 的效果，而不必复杂的将上对象 margin 设为 2，下对象 margin 设为 0。

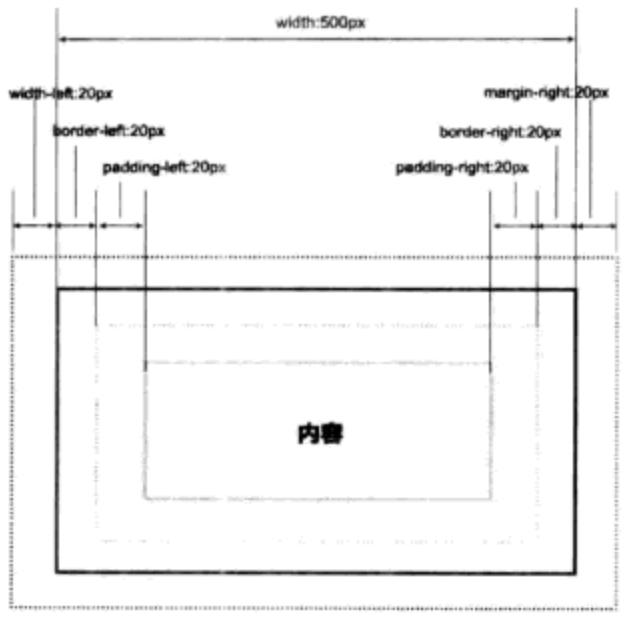


图 7-17　IE 5.5 及以下版本浏览器中的盒尺寸

盒尺寸存在一个致命的缺陷就是浏览器兼容性的问题。在 IE 5.5 及以下版本的浏览器中，盒模型并非严格按照如图 7-15 所示的计算方式，盒对象的宽度就是指 margin 值加上 width 值，如图 7-17 所示。

7.5.2　盒模型 hack

了解了盒模型尺寸的计算原理之后，现在使用 CSS 模拟一个盒模型，观察盒模型会带来什么样的问题。在标准方式下，样式表应当如下。

```
div.box{
    border: 20px solid;      /*设置边框样式*/
    padding: 20px;           /*设置内边距*/
    margin: 20px;            /*设置外边距*/
    background: #ffc;        /*设置背景色*/
    width: 500px;            /*设置宽度*/
}
```

而在当前 CSS 样式之下，相同的对象在不同浏览器下的表现是不一样的，预览效果如图 7-18 所示。

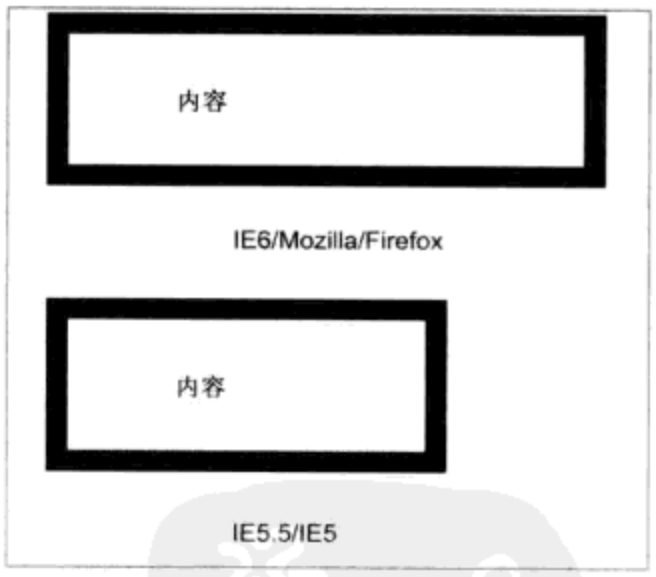


图 7-18　不同浏览器的预览效果

为了保证 IE 6 与 IE 5.5 及以下浏览器都能够显示正常，可以使用如下方法进行 CSS hack 处理，添加的 CSS 代码如下。

```
div.box{
    border: 20px solid;                    /*设置边框样式*/
    padding: 20px;                         /*设置内边距*/
    margin: 20px;                          /*设置外边距*/
    background: #ffc;                      /*设置背景色*/
    width: 580px;                          /*设置宽度，IE5.5 以下显示*/
    voice-family: "\"}\"";                 /*使用 CSS hack 解决浏览器兼容问题*/
    voice-family: inherit;
    width: 500;                            /*设置宽度*/
}
```

经过 CSS hack 处理后的样式表对 IE 5.5 及以下浏览器使用 width: 580px，而对 IE 6、IE 7 及 Mozilla/Firefox 使用 width: 500px，这样保证了同一个对象在两个浏览器下显示的宽度一样。

7.5.3　简单盒模型 hack 方法

令人振奋的是，研究盒模型在不同浏览器下 hack 的时候，开发者也找到了另一种更简单的 hack 方法，称为 SBMH（Simple Box Model Hack，简单盒模型 hack）。根据上面的代码，可以使用简单代码进行替代处理。

```
div.box{
    border: 20px solid;                          /*设置边框样式*/
    padding: 20px;                               /*设置内边距*/
    margin: 20px;                                /*设置外边距*/
    background: #ffc;                            /*设置背景色*/
    width: 500px;                                /*设置宽度*/
    \width: 580px;                               /*设置宽度，仅限在 IE5.5 浏览器下显示*/
}
```

在原来代码的基础上，以上代码填加了反斜线的 width 参数，在 IE 5.5 浏览器下能够执行这段添加的代码，即\width: 580px；而 IE 6、IE 7 及 Mozilla/Firefox 执行 width:500px。这样，利用 IE 5 的一些对 Web 标准不严格执行的 bug，简单地解决了盒模型的解析问题。

7.6　制作会动的长城形导航条

长城形导航条，即像长城烽火台一个接一个似的导航条。基本原理都是使用列表元素制作横向导航条，重点是样式上的设计，效果如图 7-19 所示。

图 7-19　长城形导航条预览效果

1. 定位元素属性 position

定位元素属性 position 的语法结构如下。

```
position: static | relative | absolute | fixed
```

positon 这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。

其中各个属性值的含义如下。

- static：默认值，位置被设置为 static 的元素，它始终会处于页面给予的位置（static 元素

会忽略任何 top、bottom、left 或 right 声明）。

- relative：位置被设置为 relative 的元素，可将其移至相对于其正常位置的地方，因此“left:20px”会将元素移至元素正常位置右边 20 个像素的位置。“left:-20px”会将元素移至元素正常位置左边 20 个像素的位置，如图 7-20 所示。

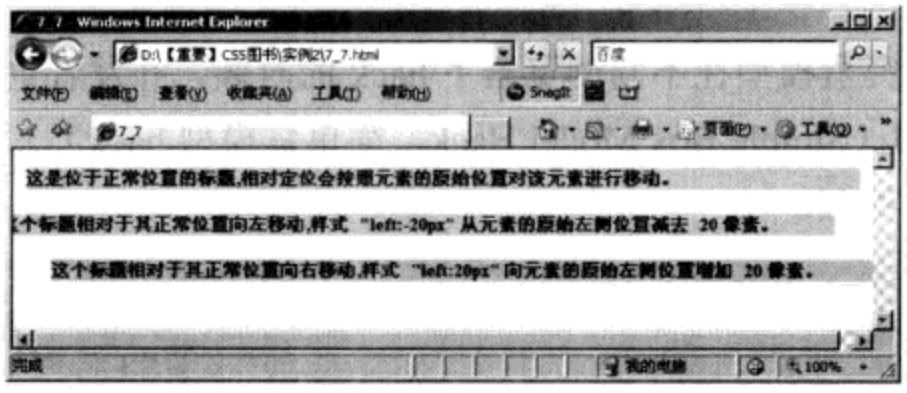


图 7-20　relative 属性值的作用效果

- absolute：位置被设置为 absolute 的元素，可定位于相对于包含它的元素的指定坐标。此元素的位置可通过 left、top、right 及 bottom 属性来规定。通过绝对定位，元素可以放置在页面上的任何位置，如图 7-21 所示。

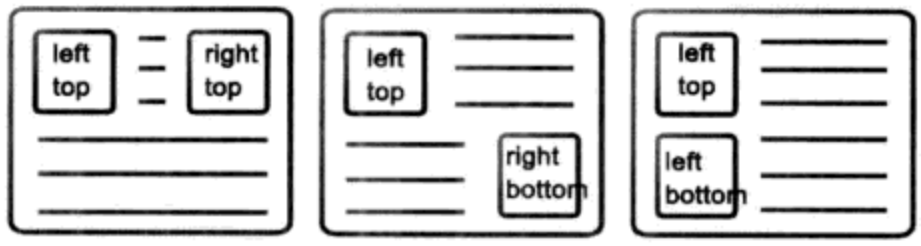


图 7-21　absolute 属性实现的定位效果

- fixed：位置被设置为 fixed 的元素，可定位于相对于浏览器窗口的指定坐标。此元素的位置可通过 left、top、right 及 bottom 属性来规定。不论窗口滚动与否，元素都会留在那个位置。该属性只限工作于 IE 7（strict 模式）。如图 7-22 和图 7-23 所示，图片不会随着窗口的滚动而移动。

图 7-22　窗口滚动前

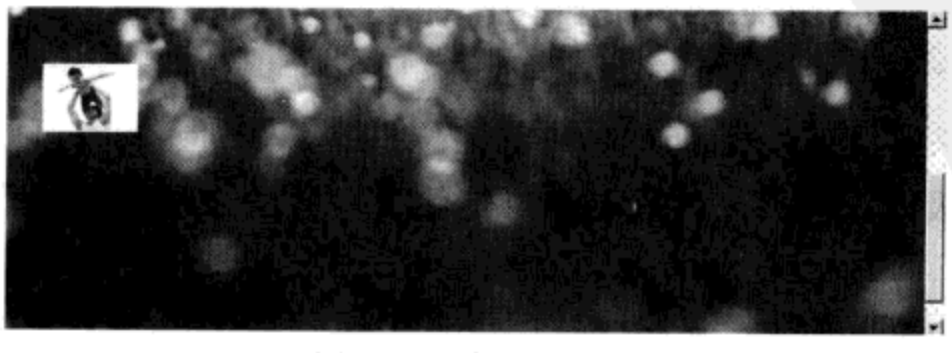


图 7-23 . 窗口滚动后

2. 制作会动的长城形导航条实例

长城形导航系统的制作方法主要是通过使用 position 的定位属性，变换列表元素的位置。

（1）制作页面的 XHTML 代码。

```
<div id="navcontainer">
    <ul id="navlist">
        <li id="active"><a href="#">首页</a></li>
        <li><a href="#">文章</a></li>
        <li><a href="#">参考</a></li>
        <li><a href="#">地址</a></li>
        <li><a href="#">论坛</a></li>
    </ul>
</div>
```

上面的结构代码，基本上都是沿用前几节的导航系统代码，唯一不同的就是本实例的导航是放在了一个 div#navcontainer 容器里面。

（2）制作#navcontainer 元素的 CSS 样式，代码如下。

```
#navcontainer {
    background: #336699;                    /*设置背景色*/
    border-top: 1px solid #99CCCC;          /*设置上边框样式*/
}
```

在以上代码中，定义了导航系统的背景颜色和上边框样式。

（3）制作#navlilst 元素和#navlist li 元素的 CSS 样式，代码如下。

```
#navlist {
    list-style: none;                       /*设置列表项目符号*/
    margin: 0;                              /*设置外边距*/
    padding: 0;                             /*设置内边距*/
}
#navlist li {
    bottom: 11px;                           /*设置下边界的距离*/
    display: inline;                        /*设置内联元素显示方式*/
    line-height: 20px;                      /*设置行高*/
    margin: 0;                              /*设置外边距*/
    padding: 0;                             /*设置内边距*/
    position: relative;                     /*设置相对定位*/
    background-color: #000;                 /*设置背景色*/
    margin: 0px 3px 3px 1px;                /*设置外边距*/
    padding: 0px 0px 1px 0px;               /*设置内边距*/
}
```

在选择符#navlist 中，定义了列表元素样式为 none，去除了由于不同浏览器所造成的边距问题。在选择符#navlist li 中，通过定义属性 display: inline 实现横向导航系统，会动的长城形导航条实例的关键技巧就是 position 定位属性。position 常用的属性值是 relative 和 absolute，应该这么理解两者的异同：对于 position: relative，它在页面上是占用位置的，也就是下一个控件必须在它后面显示；而对于 position: absolute，它本身不占用页面位置，一般会实现浮动，显示时就

会重叠在元素之上。

如果使用了 relative，即相对定位，对象还是会放置在当前对象之中，所不同的是，如果此时使用 bottom，所指的 bottom 值是相对于它的上一级对象而言，在本实例中就是相对于 div#navlist 进行相对定位。

使用 position 之后，对象的大小还是同以前一样，只不过实际位置发生了变化，这是与 margin 所不同的。margin 设置使得对象的边距成为了对象的一部分，无形中扩大了对象占有容器的区域。而使用 position 相对定位，对象不会影响 ul 的位置，预览效果如图 7-24 所示。

图 7-24　添加了列表样式的预览效果

（4）制作#navlist a 元素交互状态的 CSS 样式，代码如下。

```
#navlist a, #navlist a:link, #navlist a:visited {
    background: #900;                          /*设置背景颜色*/
    border: 1px solid #FFF;                    /*设置边框样式*/
    bottom: 2px;                               /*设置下边界距离*/
    color: #FFF;                               /*设置文字颜色*/
    display: inline;                           /*设置内联元素显示方式*/
    height: 16px;                              /*设置高度*/
    margin: 0;                                 /*设置外边距*/
    padding: 3px 5px 3px 5px;                  /*设置内边距*/
    position: relative;                        /*设置相对定位*/
    right: 2px;                                /*设置右边界距离*/
    text-decoration: none;                     /*去除下划线*/
}
#navlist a:hover {
    background: #C00;                          /*设置背景色*/
    bottom: 1px;                               /*设置下边界距离*/
    color: #FFF;                               /*设置光标经过后的文字颜色*/
    position: relative;                        /*设置相对定位*/
    right: 1px;                                /*设置右边界距离*/
}
#navlist a:active {
    background: #999;                          /*设置文本激活状态时的背景色*/
    bottom: 0px;                               /*设置下边界距离*/
    color: #FFF;                               /*设置文字颜色*/
    position: relative;                        /*设置相对定位*/
    right: 0px;                                /*设置右边界距离*/
}
```

在以上代码中，对 a 元素伪类的 4 种状态做了样式定义，即:link（未访问的链接）、:hover（光标经过链接）、:visited（已访问的链接）和:active（激活链接）。

在群组选择符#navlist a、#navlist a:link、#navlist a:visited 中，a 元素的 3 种状态的右边缘定义成相对于#navlist li 元素（父级元素）右边缘的左侧 2px；下边缘定义成相对于#navlist li 元素下边缘的上侧 2px。并且，a 元素定义成了内联属性，使得链接对象的显示方式由默认的以块状元素垂直排列改为以内联元素水平排列，实现了横向文字导航的效果。

在选择符#navlist a:hover{}中，把 a 元素的“光标经过链接”状态定义成下边缘相对于#navlist li 元素的上侧 1px，右边缘相对于#navlist li 元素的左侧 1px。在选择符#navlist a:active{}中，把 a 元素的“激活链接”状态定义成去除了下边缘相对于#navlist li 元素上侧的距离，右边缘相对于#navlist li 元素左侧的距离。这样定义实现了长城形导航条在交互状态时的运动效果，预览效果如图 7-25 所示。

（5）制作#navlist a#active 元素交互状态的 CSS 样式，代码如下。

```
#navlist li#active {
    background: #369;                                    /*设置背景色*/
    bottom: 13px;                                        /*设置下边界距离*/
    display: inline;                                     /*设置内联元素显示方式*/
    margin: 0 3px 0 0;                                   /*设置外边距*/
    padding: 0;                                          /*设置内边距*/
    position: relative;                                  /*设置相对定位*/
}
#navlist #active a, #navlist #active a:link, #navlist #active a:visited, #navlist #active
a:hover {
    background: #369;                                    /*设置背景色*/
    border-bottom: none;                                 /*设置下边框*/
    border-left: 1px solid #9CC;                         /*设置左边框样式*/
    border-right: 1px solid #9CC;                        /*设置右边框样式*/
    border-top: 1px solid #9CC;                          /*设置上边框样式*/
    bottom: 0;                                           /*设置下边界距离*/
    color: #FFF;                                         /*设置文字颜色*/
    margin: 0;                                           /*设置外边距*/
    padding: 2px 5px 0 5px;                              /*设置外边距*/
    position: relative;                                  /*设置相对定位*/
    right: 0;                                            /*设置右边界距离*/
}
```

在以上代码中，为处于当前状态 id=active 的 a 元素定义 CSS 样式。首先，为选择符#navlist li#active 定义了相对位置，与#navlist li 元素的样式相似。然后，使用群组选择符定义了处于当前状态 a 元素的 4 种交互状态的样式，这 4 种状态的样式都是一样的。

通过改变标签 a 的不同触发时候的样式，达到长城形触动的时候导航项能移动的导航系统。会动的长城导航系统制作完成，预览效果如图 7-26 所示。

图 7-25　添加了交互样式的预览效果

图 7-26　会动的长城形导航条的预览效果

当光标移到其他导航项上面的时候，效果如图 7-27 所示。

3. 兼容问题

这个实例的代码稍微复杂一些，所以很有可能存在兼容问题。现在在 IE 6 和 Firefox 浏览器中分别测试，如果发现存在浏览器之间的兼容问题，便进行调试。如图 7-28 所示为在 Firefox 浏览器中的预览效果。

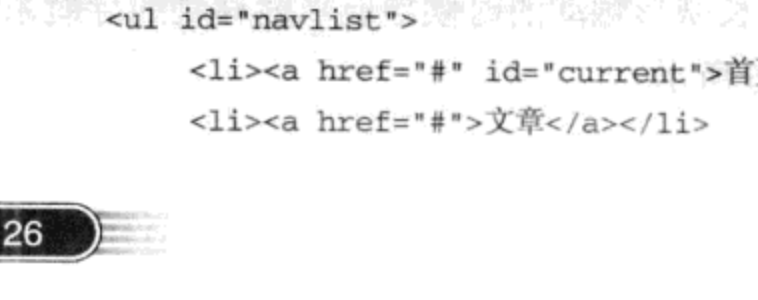


图 7-27　光标移到导航按钮上面时候的效果

图 7-28　Firefox 浏览器中的预览效果

7.7　制作基于背景控制的导航条

现在结合背景控制的知识，可以尝试制作导航系统，并在此基础上对导航进行改进。基于第 5 章背景的基本控制方法和本章前面介绍的导航制作方法，制作一个基于背景控制的导航条。

图 7-29　基于背景控制的导航原理

基于背景控制的导航条，就是利用 a:link 和 a:hover 等不同状态下显示同一张的背景图片，但是控制这张背景图片的不同坐标来实现交互性的导航系统。这样做的优点就在于优化代码，减小网页下载的文件量，既能减轻网络服务器端的负担，也能为浏览者打开网页减少不必要的时间等待。以一种形象化的图示表示，如图 7-29 所示。

下面根据如图 7-29 所示的原理，制作一个导航条。

（1）制作页面的 XHTML 代码，继续沿用前面使用过的导航系统结构。

```
<div id="navcontainer">
    <ul id="navlist">
        <li><a href="#" id="current">首页</a></li>
        <li><a href="#">文章</a></li>
```

```
            <li><a href="#">参考</a></li>
            <li><a href="#">地址</a></li>
            <li><a href="#">论坛</a></li>
        </ul>
    </div>
```

根据布局设计方法，将导航 ul 元素放入了#navcontainer 元素中，实现其包含结构，所不同的仍然是改进 CSS 样式。

（2）制作#navcontainer 元素的 CSS 样式，代码如下。

```
#navcontainer{
    width: 802px;                                    /*设置宽度*/
    height: 98px;                                    /*设置高度*/
    background: url(images/7_6_01.gif);              /*设置背景图片*/
}
```

在以上代码中，对于页面顶部区域定义了#navcontainer 元素，制作背景图片之后，定义背景图片属性，背景图如图 7-30 所示。

图 7-30　#navcontainer 区域的背景图

定义背景的宽度和高度属性用来显示背景图片。对于包含在 navcontainer 中的#navlist，根据 CSS 布局方法，将其放置在#navcontainer 元素内的右下角。

（3）制作#navlist 元素的 CSS 样式，代码如下。

```
#navlist{
    height: 26px;                                    /*设置导航的高度*/
    list-style: none;                                /*去除列表元素的样式*/
    float: right;                                    /*设置向右浮动定位*/
    margin-top: 72px;                                /*定义上边距 72px*/
}
#navlist li{
    float: left;                                     /*设置向左浮动定位*/
}
```

在以上代码中，目的是要把导航系统放在页面的右下位置，所以在#navlist 元素中定义成向右浮动，浮动到它的父级元素#navcontainer 右侧。然而，导航项要使用浮动方式定位，所以在#navlist li 元素中，定义成了向左浮动，浮动到#navlist li 元素的父级元素#navlist 元素的左侧，效果如图 7-31 所示。

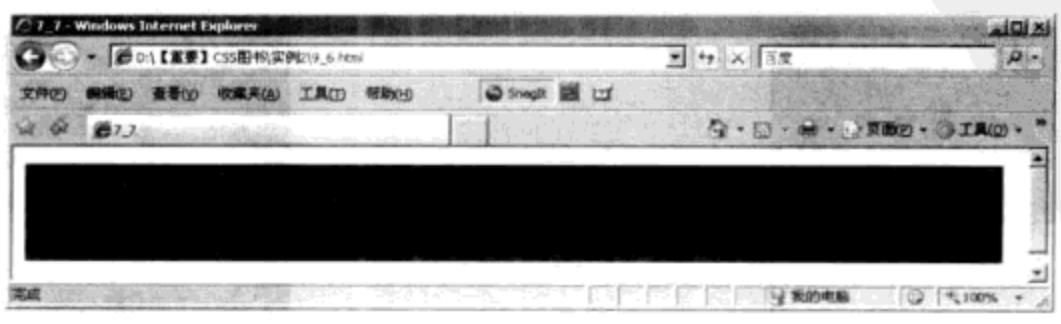


图 7-31　导航项只是基本的文字链接样式

（4）制作#navlist li a 元素交互状态的 CSS 样式，代码如下。

```
#navlist li a{
    color: #FFFFFF;                              /*设置链接文字颜色*/
    text-decoration: none;                       /*设置文本的修饰样式*/
    padding-top: 7px;                            /*设置内上边距*/
    display: block;                              /*设置块级元素显示方式*/
    width: 97px;                                 /*设置宽度*/
    height: 19px;                                /*设置高度*/
    text-align: center;                          /*设置文本居中*/
    background:url(images/7_6_02.gif);           /*设置背景图片*/
    margin-left: 2px;                            /*设置左外边距*/
}
#navlist li a:hover{
    background:url{ images/7_6_02.gif };         /*设置背景图片*/
    background-position: 0px -26px;              /*设置背景图片定位*/
    color: #FFFFFF;                              /*设置文字颜色*/
}
#navlist li a#current{
    background: url(images/7_6_02.gif);          /*设置背景图片*/
    background-position: 0px -52px;              /*设置背景图片定位*/
    color: #000000;                              /*设置文字颜色*/
}
```

在最初实现的使用图片背景制作的交互式导航中，对于每一个导航项中的标准状态、当前状态和光标交互状态 3 个状态中，均使用了 3 张 gif 图片，如图 7-32 所示。通过 CSS 中的 a:hover 进行切换来实现实际效果。

在背景定位一节中，实现一张背景在一个标签中采用 top 和 left 的距离来实现绝对定位，而标签式导航 3 张图片也可以在通栏用一张图片，并且使用绝对定位来实现。

在以上的代码中，通过 CSS 的变化，先前的 3 张图片背景被指定为了一张背景图，当然这不是实现导航图片的核心代码。重点在于 background-position 的定义，由于 3 张图片合为了一张，在有了背景精确定位这项技术之后，可以通过改变图片在标签中的显示位置来实现图片的切换。因此在选择符#navlist li a:hover 中将背景定位向上移了 26px，而在选择符#nav lli a#current 中将图片向上移了 52px。通过在其标签中的不同定位，来达到显示不同图片的效果，如图 7-33 所示。

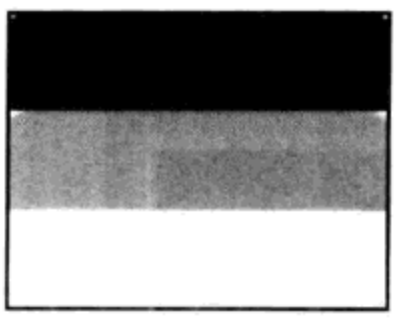


图 7-32　导航项交互所用的 3 个状态的 gif 图片

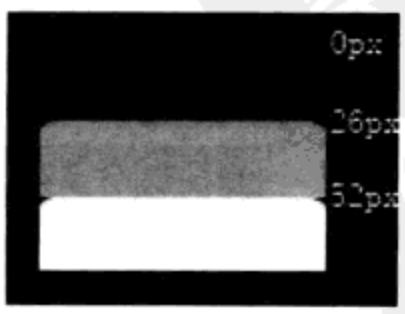


图 7-33　3 个状态在同一张图片的位置

导航页面预览效果如图 7-34 所示。

图 7-34　基于背景控制的导航优化

光标经过某个菜单时的效果如图 7-35 所示。

图 7-35　光标经过某个菜单时的效果

7.8　制作左右自由伸展的导航条

在 7.7 节的实例中，使用了图片翻转技术实现了带圆角的图片背景，并实现了交互效果。但是它却只能够做到使用同一个背景进行设计，并没有实现根据导航文字的长度，自由变化的圆角导航标签。本节将介绍一种新的方法，利用九宫格方式能够实现任意扩展的圆角样式的功能，改进导航系统，制作可以左右自由伸展的导航条，这种方法又叫做——滑动门技术（Sliding Doors）。

7.8.1　九宫格技术原理

在一个 3×3 的表格之中，左上、右上、右下、左下分别放入 4 个圆角图案，内容放置在中间的方格中，其上下左右 4 个方向的方格放置用于拉伸的图案，最终形成了一种可任意变化大小的圆角方框。九宫格技术也是软件的外观设计中常用的技术之一，包括我们使用的 Windows 软件，打开的每一个窗口基本上都是使用九宫格方式进行的样式设计。在表格式布局中通过表格实现圆角样式，而 CSS 2.0 本身是不提供圆角设计模式。

注意：

CSS 3.0 将提供直接对九宫格样式设计的支持。

如图 7-36 所示九宫格方式使得圆角样式能够根据内容的大小而自由变化。

现在要实现一个能够根据内容大小变化的、具有扩展性的圆角框。不管是 CSS 还是表格布

局，实现可自由变化的圆角样式都是基于九宫格技术，在九宫格中用于 4 个角使用的圆角图片，如图 7-37 所示。

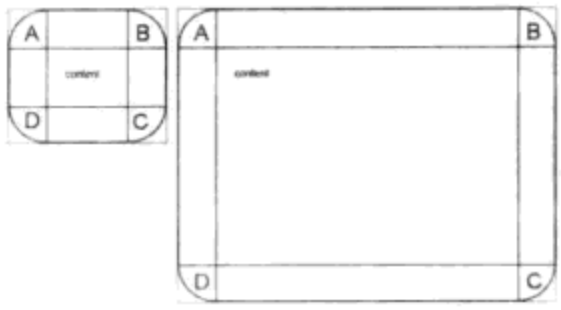


图 7-36　九宫格方式

图 7-37　九宫格用于 4 个角使用的圆角图片

（1）制作 4 个圆角图片将用于方框的 4 个顶点的 XHTML 代码。

```
<div>
    <strong>什么是CSS样式表</strong>
    CSS（Cascading Style Sheets）层叠样式表目前最新版本为 3.0，是控制网页布局样式的基础，并真正能够做到网页表现与内容分离的一种样式语言。相对于传统 HTML 的简单样式控制而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，以及拥有对网页对象盒模型样式的控制能力，并能够进行初步页面交互设计，是目前基于文本展示的最优秀的表现设计语言。
</div>
```

XHTML 本身是由一个 div 格式构成了一个内容块，现在希望为这个内容框加入圆角边框。首先，必须实现将 4 个圆角图片放置到 div 的 4 个角上，由于每一个对象的背景只能指定一次，因此不能够同时把 4 个图片放置在一个 div 中，必须使用 4 个处于相同位置的 div 来放置 4 张图片。改进的 XHTML 代码如下。

```
<div class="tl">
    <div class="bl">
        <div class="tr">
            <div class="br">
                <strong>什么是CSS样式表</strong>
                CSS（Cascading Style Sheets）层叠样式表目前最新版本为 3.0，是控制网页布局样式的基础，并真正能够做到网页表现与内容分离的一种样式语言。相对于传统 HTML 的简单样式控制而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，以及拥有对网页对象盒模型样式的控制能力，并能够进行初步页面交互设计，是目前基于文本展示的最优秀的表现设计语言。
            </div>
        </div>
    </div>
</div>
```

定义了 4 个 div 嵌套在一起，使用 tl、bl、tr、br 作为 class 名，分别表示左上、左下、右上、右下 4 个顶点。

（2）制作九宫格的 CSS 样式，代码如下。

```
.tl{
    background: url(images/tl.gif) 0 0 no-repeat #000000;/*设置左上角背景图片以及背景图片样式*/
}
.tr{
```

```
    background: url(images/tr.gif) 100% 0 no-repeat;   /*设置右上角背景图片以及背景图片样式*/
}
.bl{
    background: url(images/bl.gif) 0 100% no-repeat;   /*设置左下角背景图片以及背景图片样式*/
}
.br{
    background: url(images/br.gif) 100% 100% no-repeat/*设置右下角背景图片以及背景图片样式*/
    padding: 20px;                                     /*设置内边距*/
    color: #FFF;                                       /*设置文字颜色*/
    font-size: 12px;                                   /*设置文字大小*/
}
```

在选择符.tl 元素中，位于 top left 位置的圆角图片，对.tl 的 div 设置了相应的背景图片，并使其放置在坐标为(0,0)的左上角位置，以此类推，其他 3 个顶点也同样是这样完成的。这时候 4 个圆角图片已经被放置到了 4 个指定位置。对于 4 个 div 的背景设置中，分别使用了不同图片作为各顶点的圆角图案。与此同时，为了保证圆角样式能够根据 div 的大小自动放置到相应顶点，在对背景进行定位时使用了百分比，4 个不同的角分别使用宽度或者高度的 100%，使得几个图片能够根据当前元素的大小自动放置在最宽或是最高的地方。

实现了圆角图片的定位，然后对文本进行排版和设置背景颜色。由于 4 个 div 进行嵌套，后一个 div 设置了背景的话，会覆盖前一层的圆角图片，因此这里只对.tl{}也就是第一个 div 进行了背景颜色增加，然后开始进行排版，使用了 CSS 样式属性，预览效果如图 7-38 所示。

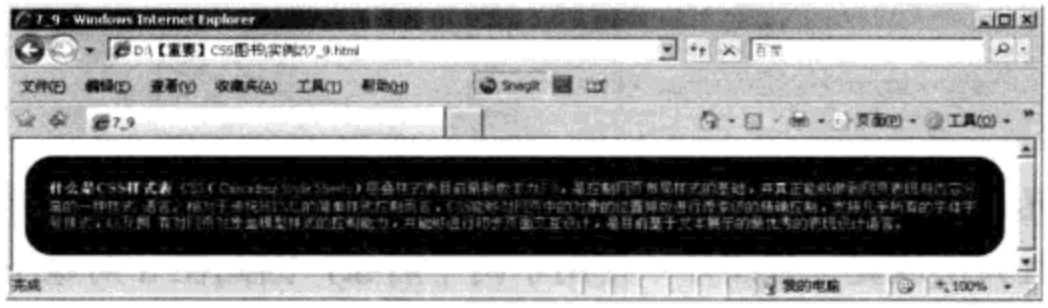


图 7-38　九宫格技术实现圆角矩形边框

改变浏览器的宽度和高度，可以看到圆角边框随着浏览器自由伸缩，如图 7-39 所示。

图 7-39　不同大小浏览器中的显示效果

7.8.2　制作自由伸展的导航条实例

九宫格使用 4 个圆角图片的模式，是为了上下左右 4 个边都可以自由扩展，而对于导航标签来说只需要左右两个方向的扩展即可。按此原理导航的结构如图 7-40 所示。

类似于此的结构需要准备 3 张图片，A 和 C 为标签两边，B 为标签的背景。当前 XHTML 代码中，链接 a 对象只能够使用一张图片，因此如果使用类似九宫格方式的话，需要 3 个对象进行嵌套完成，这样过于复杂。

进一步优化这样的结构，使用二层嵌套完成，因为 B 和 C 或者 A 和 B，都可以合在一起，使 B 部分延续得较长一些，因此滑动门结构应该如图 7-41 所示。

图 7-40　滑动门原理 1

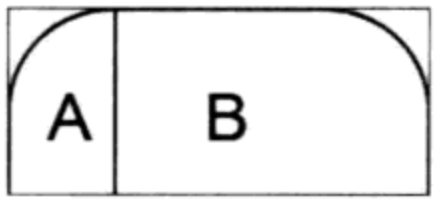


图 7-41　滑动门原理 2

（1）制作页面的 XHTML 代码，改进 7.8.1 节导航的 XHTML 代码。

```
<ul id="navlist">
    <li><a href="#"><span>首页</span></a></li>
    <li><a href="#"><span>长文字的导航标签</span></li>
    <li><a href="#"><span>文章</span></a></li>
    <li><a href="#"><span>参考</span></a></li>
    <li><a href="#"><span>地址</span></a></li>
    <li><a href="#"><span>论坛</span></a></li>
</ul>
```

为了保证具有光标交互效果，必须将所有图片都放置在 a 对象之中，这样才可以应用 a:hover 状态下的样式，而且为了放置一张图片，使用了两个对象，因此在 a 对象下使用 span。导航系统添加了一个导航文字较长的导航项：
长文字的导航标签。

（2）制作#navlist 元素和#navlist li 元素的 CSS 样式，代码如下。

```
#navlist{
    list-style: none;                    /*设置列表符号样式*/
}
#navlist li{
    display: block;                      /*设置块级元素显示方式*/
    float: left;                         /*向左浮动*/
    margiin: 0px;                        /*设置外边距*/
    padding: 0px;                        /*设置内边距*/
}
```

图 7-42　导航系统的雏形骨架

现在只是制作了导航系统的雏形骨架，还分不出间隔位置的链接文字，如图 7-42 所示。

（3）制作滑动门导航部分的 CSS 样式，代码如下。

```
#navlist a {
    margin-left: 2px;                                          /*设置左外边距*/
    float: left;                                               /*设置向左浮动定位*/
    background: url("images/l.gif") no-repeat left -25px;      /*设置背景样式*/
    padding-left: 6px;                                         /*设置左内边距*/
    text-decoration: none;                                     /*设置文本修饰样式*/
}
#navlist a span {
    display: block;                                            /*设置块级元素显示方式*/
    background: url("images/r.gif") no-repeat right -25px;     /*设置背景样式*/
    padding: 5px 15px 3px 6px;                                 /*设置内边距*/
    color: #666666;                                            /*定义默认状态的文字颜色*/
    font-size: 12px;                                           /*定义文字大小*/
}
#navlist a:hover span {
    color: #000;                                               /*定义“光标经过链接”状态的文字颜色*/
    background-position: 100% 0px;                             /*设置背景定位*/
}
#navlist a:hover {
    background-position: 0% 0px;                               /*设置背景定位*/
}
```

在以上代码中，首先，将滑动门结构图中的 A 图片放在选择符#navlist a 元素中，对于#navlist a 对象而言，放置了用于标签左半部的图片 l.gif，如图 7-43 所示。

这里重点是背景图片的定义。background 属性定义了背景图不平铺，居左显示，定位向上偏移 25px。并且定义左内边距为 6px，目的是使 a 元素下的 span 隔 6 个像素才能显示，因此不会覆盖掉 a 下的背景图片。

然后，在选择符#navlist a span 中，放置滑动门结构图中的 B 图片，也就是图片 r.gif，如图 7-44 所示。#navlist a span 元素下的背景图片不平铺，居右显示，定位向上偏移 25px。span 默认的显示模式是 inline，在这里通过 display:block 将其改变为块状显示，以便使 span 形成块状显示模式，让 span 的占位能够充满整个 a 对象，然后通过内边距的定义，将文字放置到了 span 的中间。

图 7-43　图片 l.gif

图 7-44　图片 r.gif

设置选择符#navlist a:hover 样式，采用 7.8.1 节介绍的图片翻转技术，在 a 和 a:honver 放置同一张图片，通过偏移其位置来显示不同交互效果。可以自由伸展的导航条制作完成，预览效果如图 7-45 所示。

图 7-45　自由伸展的导航条预览效果

7.9　制作一个简单的纵向导航条

纵向导航也是网站应用中的一种重要形式。所谓纵向导航，是指放置在网页左边或右边的从上至下排列的一种导航形式。现在希望设计一套纵向导航来帮助用户浏览网站。类似于电子商务网站，在每一个页面都需要一套辅助的导航系统来帮助用户查找各个分类的商品，这个时候纵向导航就能派上用场了。纵向导航条在左侧的网页如图 7-46 所示。

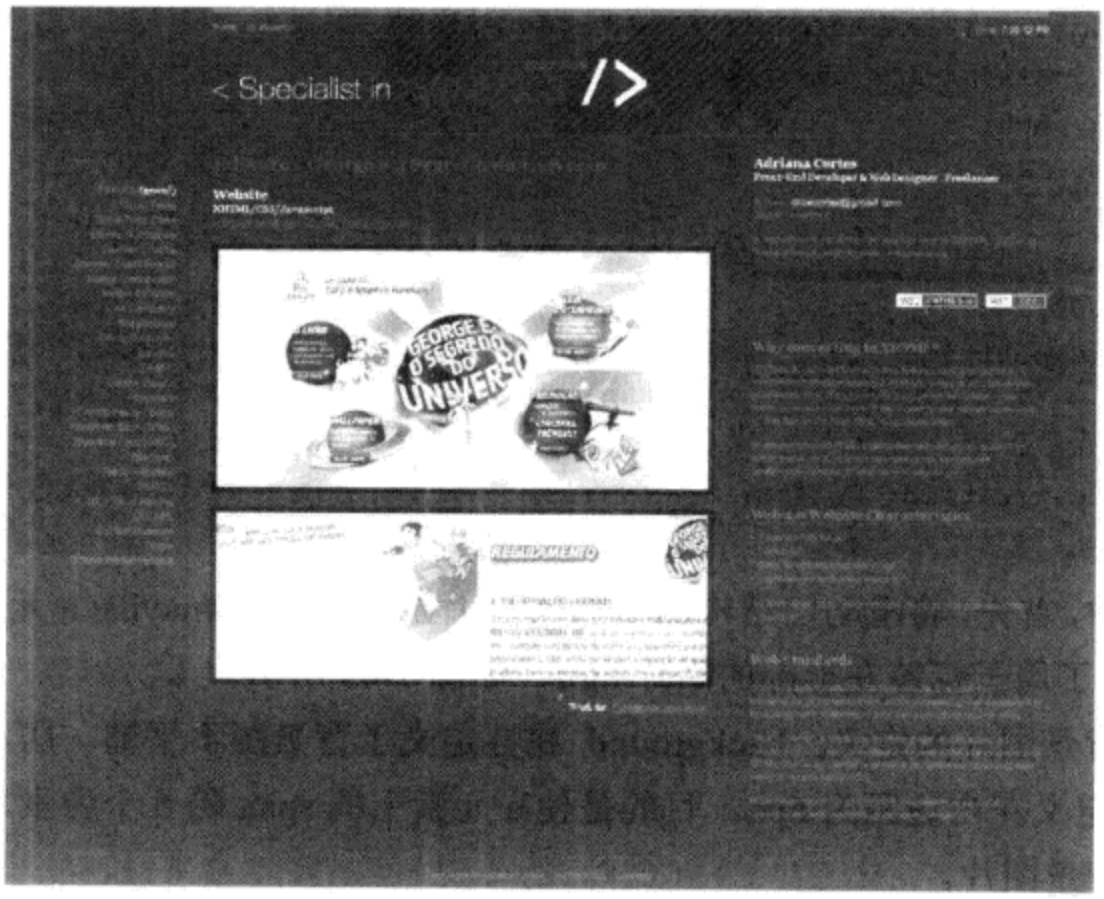


图 7-46　文字的纵向导航条网页

使用纵向导航的目的主要是让用户方便找到网站中的文章，更类似于 msn.com 主页的形式，应该有一个二级分类及其下属的内容，例如 CSS 目录下可能会出现 CSS 入门、CSS 技巧等子栏目。

7.9.1　使用 ul、li 列表标签制作导航条

本章的 7.1 节讲解过制作横向导航的方法，是使用 ul、li 列表元素，默认情况下，呈现的效果便是导航项纵向排列。制作页面的 XHTML 代码。

图 7-47　列表元素制作纵向导航

```
<ul id="navlist">
    <li><a href="#">Item one</a></li>
    <li><a href="#">Item two</a></li>
    <li><a href="#">Item three</a></li>
    <li><a href="#">Item four</a></li>
    <li><a href="#">Item five</a></li>
</ul>
```

制作纵向导航的时候，这段代码不需要添加任何 CSS 样式，便可以呈现希望的效果，预览效果如图 7-47 所示。

7.9.2　使用 div+h1 标签制作

延续横向导航的设计思路，但换一种方式来组织我们的导航部分的 XHTML 结构代码。

为了便于读者阅读，在本段代码中不再给每段文字加上链接<a>标签。这次的 XHTML 部分的代码与横向导航略有不同，没有继续使用 ul 和 li 标签，其实继续使用 ul 元素也能完好地实现纵向导航系统，但是在这里需要更多方法来展现 CSS 布局设计的灵活与方便性，以便于抛砖引玉，开拓更多的设计思想。

1．h 标签

h 标签包括 h1、h2、h3、h4、h5、h6，是 XHTML 专门处理页面上的标题的标签。h1 就是万能的君主，而 h6 就是最底层的百姓。它们的区别是字号大小不同，h1 标签字号最大，随后字号依次减小，h6 标签字号最小。

一个页面只能存在一个 h1 或没有 h1，h2、h3、h4 可以有多个。多个 h1 造成的后果是搜索引擎不知道这个页面使用的是哪个标题内容，会淡化这个页面的标题和关键词。

h 标签比较效果如图 7-48 所示。

2．制作纵向导航条实例

现在尝试换一种思路来制作一个纵向导航条，使用 div+标题标签，利用这两种标签的特性，即单独占据一行显示，制作纵向导航。

（1）制作页面的 XHTML 代码。

```
<div id="navlist">
    <h1>CSS</h1>
    <h2>CSS 入门</h2>
    <h2>CSS 进阶</h2>
    <h2>CSS 高级技巧</h2>
    <h1>WebUI</h1>
    <h2>理论知识</h2>
    <h2>实战应用</h2>
    <h2>高级技巧</h2>
    <h1>DOM</h1>
    <h2>DOM 入门</h2>
    <h2>DOM 应用</h2>
    <h2>DOM 与浏览器 </h2>
    <h1>XHTML</h1>
    <h2>XHTML 参考手册</h2>
    <h2>XHTML 论坛</h2>
</div>
```

在以上代码中，这一次采用的标签是 div+h1+h2 的形式。使用 div 标签设定了一个导航的结构区域。在这个区域中使用 h1 来作二级分类的标题，h2 来作二级分类下面的细节内容。

在 XHTML 的语法意义中，h1、h2、h3 这些标签本身就具有用于对文本进行层级划分的意义，直接使用 h1、h2 来表示层级关系，相对于在标签中加入 id 或 class 来作层级的标记更为简单和直

观，在这里使用 h1、h2 来标记不同级别的分类名称也实在是再合适不过了，效果如图 7-49 所示。

图 7-48　h 标签比较效果

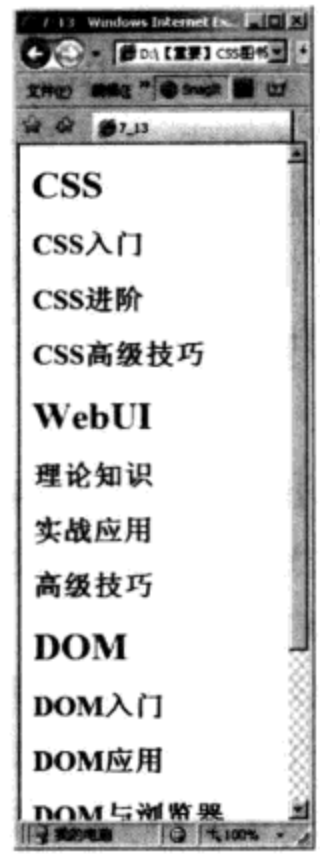


图 7-49　使用 h1、h2 标记的效果

（2）制作导航系统的 CSS 样式，代码如下。

```
#navlist{
    width:100px;                                    /*设置宽度*/
    border-color:#c5c6c4;                           /*设置边框颜色*/
    border-style: solid;                            /*设置边框样式*/
    border-width:0 1px 1px 1px;                     /*设置边框大小*/
}
#navlist h1,#category h2 {
    margin:0;                                       /*设置外边距*/
    padding:4px;                                    /*设置内边距*/
    font-size:12px;                                 /*设置字号*/
}
#navlist h1 {
    border-top:1px solid #c5c6c4;                   /*设置上边框样式*/
    background-color:#f4f4f4;                       /*设置背景色*/
}
#navlist h2 {
    font-weight:normal;                             /*设置字体*/
}
```

CSS 代码部分还是沿用以前的思路。对 id 为 category 的 div 定义了宽度，以及让它的 right、bottom 和 left 3 条边框产生灰色的 1px 宽度的线条。

为什么只设定了 right、bottom 和 left 3 个边线而没有设定顶部，原因就在 h1 部分。看对 #category h1 {}部分的定义，#category 下的 h1 元素定义了间距、字体字号及 top 顶部的线条，颜色与#category 的一致，因此第一个 h1 元素的顶部就自然而然成为了整个导航区域的顶部。如

果让#category 的顶部也有一条线条的话，那么高度就是加上 h1 顶部的 2px，而不是统一的 1px 了，因此对#category 部分的顶部线条可以省略不写了。

另外，在#category 下的 h1 和 h2 部分，都定义了 margin:0;，这里是为了消除 h1 和 h2 元素默认的外边距，显然不符合设计需要，因此在这里重新设定了 margin 及 font-size 元素，以消除默认效果来符合现在的设计目标。

注意：

h 标签元素在不加任何样式的情况下，将拥有自己的一套默认样式，将采用大边距、大字号的形式。

比较一下上面的代码，各方面都简化不少，首先将上面代码中对于#category 的 3 段边框的定义变成一套简化定义，因为 3 段边框除了上边框为 0 之外，其他边框的样式及颜色都一致，因此统一使用 border-color 及 border-style 定义颜色及边框样式，然后在 border-width 中分别设置上、右、下、左 4 边的宽度为 0、1px、1px、1px，改进了原有的代码定义，效果如图 7-50 所示。

另外，重新增加了#category h1、#category h2 这样一套样式，因为 h1 和 h2 都拥有一些相同的属性，因此在这里作为 h1、h2 的父级定义，统一对它们的外边距、内边距及字号作了定义，所在 h1 和 h2 的单独样式中代码量也减少了很多。通过这样的一些优化，不但减少了代码数量，也使得其语义更加明确，可读性大大提高，重复的定义归类到了一起，也便于以后统一修改，预览效果如图 7-51 所示。

图 7-50　没有上边框的效果

图 7-51　纵向导航条的预览效果

7.10　制作分行导航条

分行导航条是纵向导航的一种形式，主要特点就是在导航项之间加入了分割线。分行导航能够清晰地突出导航项之间的关系。

本节介绍制作一个简单明了的分行导航条实例。制作方法是在纵向导航条的基础上，添加水平分割线，没有使用任何新的 CSS 属性。

（1）制作分行导航条 XHTML 代码。

```
<ul id="navlist">
    <li><a href="#">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">论坛</a></li>
    <li><a href="#">联系</a></li>
</ul>
```

结构代码依然是沿用本章使用的导航系统的基本结构。

（2）制作导航条的 CSS 样式，代码如下。

```
#navlist{
    padding-left: 0;                          /*设置左内边距*/
    margin-left: 0;                           /*设置左外边距*/
    border-bottom: 1px solid gray;            /*设置下边框样式*/
    width: 100px;                             /*设置宽度*/
}
#navlist li{
    list-style: none;                         /*设置列表符号项目样式*/
    margin: 0;                                /*设置外边距*/
    padding: 5px;                             /*设置内边距*/
    border-top: 1px solid gray;               /*设置上边框样式*/
    font-size: 12px;                          /*设置字号*/
}
#navlist li a {
    text-decoration: none;                    /*设置文本修饰样式*/
    color: gray;                              /*设置文字颜色*/
}
```

图 7-52　纵向导航条雏形

在 CSS 样式设计上，首先在导航主体框架 #navlist 添加了下边框 border-bottom: 1px solid gray，使用 padding-left: 0 和 margin-left: 0 去除了 ul 由于不同浏览器下可能存在的内边距和外边距问题，同时也设置了导航的宽度为 100px，预览效果如图 7-52 所示。

第二步，主要是为导航项#navlist li{}指定了同导航主体框架一样的上边框 border-top: 1px solid gray，这样形成了分行导航的样式。同样，margin: 0 去除了 li 元素由于不同浏览器可能存在的外边距，重新指定了内边距 padding: 5px，效果如图 7-53 所示。

最后，#navlist li a{}为链接重新指定了样式，去除了 a 标签默认的下划线和将其文字颜色指定为灰色（gray），最终预览效果如图 7-54 所示。

图 7-53　分行导航样式的效果

图 7-54　分行导航条预览效果

7.11　制作增亮导航条

有一种交互效果，当光标移到某一区域的时候，该区域的背景颜色改变。导航系统同样可以应用这种交互效果，实现更人性化的用户体验。如图 7-55 所示是一个使用了纵向导航，当光标经过时导航项变成橙色的网页。

图 7-55　纵向增亮导航的网页

本节介绍制作一种应用这种交互效果的增亮的导航条。关键的方法是指定光标经过和移出时不同的背景颜色，也没有使用新的 CSS 属性方法。结构代码依然同前面一样。

（1）制作增亮导航条的 XHTML 代码。

```
<ul id="navlist">
    <li><a href="#">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考</a></li>
    <li><a href="#">Blog</a></li>
```

```
    <li><a href="#">论坛</a></li>
    <li><a href="#">联系</a></li>
</ul>
```

（2）制作导航链接部分的 CSS 样式，代码如下。

```
#navlist{
    margin-left: 0;                                    /*设置左外边距*/
    padding-left: 0;                                   /*设置左内边距*/
    list-style-type: none;                             /*设置列表符号项目样式*/
    font-family: Arial, Helvetica, sans-serif;         /*设置字体类型，可供选择列举了 3 个*/
    font-size: 12px;                                   /*设置字号*/
}
#navlist a{
    display: block;                                    /*设置块级元素显示方式*/
    padding: 3px;                                      /*设置内边距*/
    width: 160px;                                      /*设置宽度*/
    background-color: #036;                            /*设置背景色*/
    border-bottom: 1px solid #eee;                     /*设置下边框样式*/
}
```

图 7-56　添加导航链接样式的预览效果

在以上代码中，导航项的链接部分#navlist a{}、display: block 将其定义为块状显示方式，同时指定了这个块状元素的宽度为 160px、背景颜色为#036、内边距为 3px 和下边框样式。下边框样式的设计是沿用 7.10 节的分行导航使用分割线的方法，定义了一条颜色为#eee 的实线，把导航项与导航项之间区分开，效果如图 7-56 所示。

（3）制作链接的交互效果的 CSS 代码，代码如下。

```
#navlist a:link, #navlist a:visited{
    color: #EEE;                          /*设置文字颜色*/
    text-decoration: none;                /*设置链接文字的修饰样式*/
}
#navlist a:hover{
    background-color: #369;               /*设置背景色*/
    color: #fff;                          /*设置文字颜色*/
}
```

在以上代码中，使用伪类为导航项链接指定了 3 种不同状态的 CSS 样式，即:link（访问链接）、:vistited（已经被访问的链接）和:hover（光标经过）。在第 4 章介绍过 CSS 的伪类的使用方法，这里使用并列选择符#navlist a:link,#navlist a:visited{}定义了导航项链接的访问和已经被访问的两种状态的样式；使用#navlist a:hover{}，单独定义了光标移上去时候的状态样式，光标经过时，导航项背景颜色变为#369，实现了增亮的效果，完成了本实例的制作过程，最终预览效果如图 7-57 所示。

当光标经过某菜单时的效果如图 7-58 所示。

图 7-57　增亮导航条预览效果

图 7-58　光标经过某菜单时的预览效果

7.12　制作动感的导航条

用一个简单的列表元素（list）和 CSS 制作不同的导航条，是本章的特色所在。现在就来看看当 CSS 用于简单的 list 时的强大威力。如图 7-59 所示是一个使用了动感纵向导航条的网页。

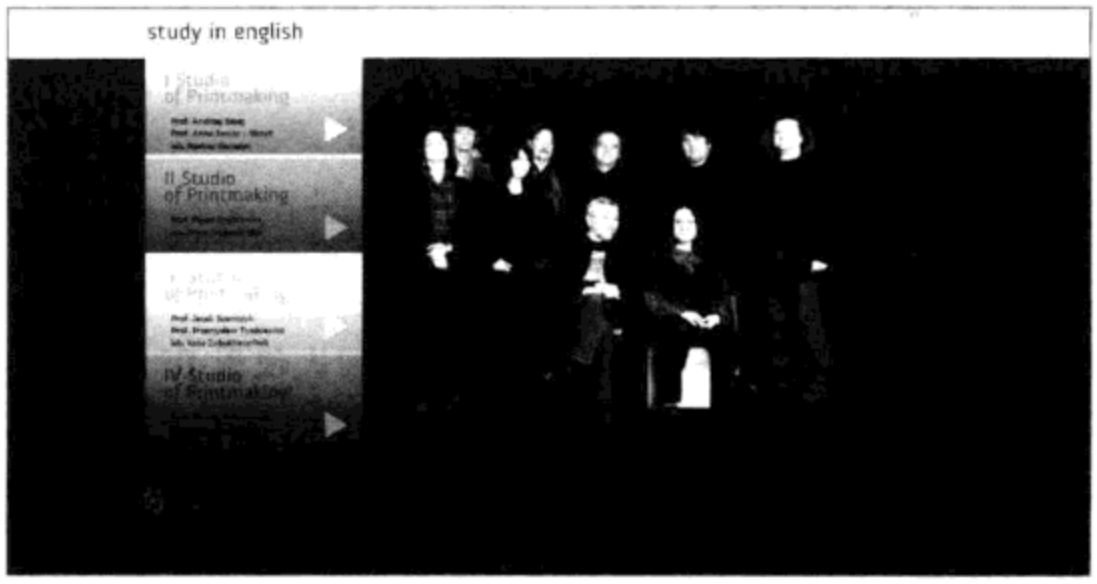


图 7-59　使用动感导航条的网页

本节将介绍制作类似于这种动感的导航条，从简单到复杂，从相貌平平到魅力十足。导航结构依然沿用以前框架。

（1）制作动感导航条的 XHTML 代码。

```
<ul id="navlist">
    <li><a href="#" id="current">首页</a></li>
    <li><a href="#">文章</a></li>
    <li><a href="#">参考</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">论坛</a></li>
    <li><a href="#">联系</a></li>
</ul>
```

这次在“首页”的导航项的 a 标签里面添加了一个 id 为 current 的属性，代表处于当前选中状态的导航项。

（2）制作导航主体部分的 CSS 样式，代码如下。

```
#navlist{
    color: white;                                    /*设置文字颜色*/
    background: #17a;                                /*设置背景颜色*/
    border-bottom: 0.2em solid #17a;                 /*设置下边框样式*/
    border-right: 0.2em solid #17a;                  /*设置右边框样式*/
    padding: 0 1px;                                  /*设置内边距*/
    margin-left: 0;                                  /*设置左外边距*/
    width: 12em;                                     /*设置宽度*/
    font: normal 0.8em Verdana, sans-serif;          /*设置字体类型*/
}
```

在以上代码中，#navlist{}定义了导航的主体部分，其中包括文字颜色、背景颜色、内边距、左外边距和宽度。为了做成具有 3D 效果的导航背景区域，这里指定了右边框和下边框属性，与前面章节不同的是，宽度的单位使用了 em。

注意：

在国内网站中，包括 3 大门户等都是使用了 px 作为字体单位，只有百度做了个可调的表率。而在大洋彼岸，几乎所有的主流站点都使用 em 作为字体单位，也就是可调的。当然 px 比 em 更加容易使用。

关键点如下。

- IE 无法调整那些使用 px 作为单位的字号。
- 国外的大部分网站能够调整的原因在于其使用了 em 作为字体单位。
- Firefox 能够调整 px 和 em，但是 96%以上的中国网民使用 IE 浏览器（或内核）。px 是相对长度单位，是相对于显示器屏幕分辨率而言的。em 是相对长度单位，相对于当前对象内文本的字体尺寸。若当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。

（3）制作导航项的 CSS 样式，代码如下。

```
#navlist li{
    list-style: none;                                /*设置列表符号样式*/
    margin: 0;                                       /*设置外边距*/
    font-size: 1em;                                  /*设置字号*/
}
#navlist a{
    width: 99%                                       /*设置宽度*/
    display: block;                                  /*设置块级元素显示方式*/
    text-decoration: none;                           /*设置链接文字的文本修饰样式*/
    margin-bottom: 0.5em;                            /*设置下外边距*/
    margin-top: 0.5em;                               /*设置上外边距*/
    background: #39c;                                /*设置背景颜色*/
    border-width: 1px;                               /*设置边框宽度*/
    border-style: solid;                             /*设置边框样式*/
    border-color: #5bd #035 #068 #6cf;               /*设置边框颜色*/
    border-left: 1em solid #fc0;                     /*重新设计左边框的综合样式*/
```

```
    padding: 0.25em 0.5em 0.4em 0.75em;                    /*设置内边距*/
}
```

在以上代码中，指定导航项的 CSS 样式。#navlist li{}去除了列表元素默认的样式，去除了外边距，定义了导航项的字号为 1em，也就是 16px。

动感导航条的主要方法就在于导航项样式的定义。#navlist a{}定义了导航项默认的链接样式，这里同样使用 display: block 设置为块状元素，text-decoration: none 去除链接文字的下划线，又定义了外边距和内边距，指定了导航项的背景颜色和宽度。重点是指定了边框，通过边框宽度 border-width: 1px、边框样式 border-style: solid 和边框四周不同颜色 border-color: #5bd #035 #068 #6cf 的定义，做出一个 3D 效果的导航项按钮。最后，border-left: 1em solid #fc0 又重新定义了一次左边框样式（宽度为 1em，样式为 solid，颜色为#fc0），覆盖之前指定的左边框样式（宽度 1px，样式 solid，颜色#6cf），预览效果如图 7-60 所示。

（4）制作当前导航项的 CSS 样式，代码如下。

```
#navlist a#current {
    border-color: #5bd #035 #068 #f30;                     /*设置链接对象的边框样式*/
}
```

在以上代码中，#navlist a#current{}定义了处于当前状态的导航项样式，它本身是继承#navlist a{}的样式属性，现在又为它添加了新的边框颜色，新的边框颜色覆盖之前#navlist a{}里面设置的边框颜色，使得产生一种交互的效果，预览效果如图 7-61 所示。

图 7-60　预览效果 1

图 7-61　预览效果 2

（5）制作导航项交互效果的 CSS 样式，代码如下。

```
#navlist a:hover, #navlist a#current:hover{
    background: #28b;                                      /*设置链接对象光标经过时的背景颜色*/
    border-color: #069 #6cf #5bd #fc0;                     /*设置边框颜色*/
    padding: 0.4em 0.35em 0.25em 0.9em;                    /*设置内边距*/
}
#navlist a:active, #navlist a#current:active{
    background: #17a;                                      /*设置链接对象点击时的背景色*/
    border-color: #069 #6cf #5bd white;                    /*设置边框颜色*/
    padding: 0.4em 0.35em 0.25em 0.9em;                    /*设置内边距*/
}
```

在以上代码中，当光标移到导航项按钮区域的时候，定义样式。使用群组选择符#navlist a:hover{},#navlist a#current:hover{}和#navlist a:active, #navlist a#current:active{}，定义了“光标经

过”和“已经被访问过”这两种状态的样式，所变化的都是背景颜色、边框颜色和内边距，最终预览效果如图 7-62 所示。当光标经过某菜单时的预览效果如图 7-63 所示。

图 7-62 动感导航条预览效果

图 7-63 光标经过某菜单时的预览效果

7.13 制作下拉式导航条

下拉式导航是网页设计中常用的导航形式，这种菜单形式能够充分利用页面现有空间隐藏与显示更多内容，并能对内容进行合理的分类显示，是一种非常优秀的导航形式。adobe.com 网站采用的是下拉式导航，如图 7-64 所示。

图 7-64 adobe.com 网站采用下拉导航

早期的下拉菜单通过隐藏的<layer>或 块来实现内容的隐藏，通过 JavaScript 脚本来响应用户的操作，目前也采用 JavaScript+div 或其他元素的形式来制作此类导航，不同的是整个导航都将使用符合标准的 CSS 布局来打造，不再使用表格来制作菜单。下拉菜单将是前面讲到的横向导航与纵向导航的组合，而且通过 CSS 对于属性的众多支持，同一个菜单不再需要多个 div 相互配合完成。使用 CSS 布局制作下拉菜单，甚至可以直接控制 ul 和 li 元素。现在尝试一个最简单的下拉菜单的制作。

7.13.1　制作实例

本节使用 ul 列表标签以及 CSS 实现样式制作下拉式导航系统，需要应用一段 JavaScript 脚本来实现下拉交互效果。

（1）制作导航系统的 XHTML 代码。

```
<ul id="navlist">
    <li><a href="">文章</a>
        <ul>
            <li><a href="">CSS 教程</a></li>
            <li><a href="">DOM 教程</a></li>
            <li><a href="">XML 教程</a></li>
        </ul>
    </li>
    <li><a href="">参考</a>
        <ul>
            <li><a href="">XHTML</a></li>
            <li><a href="">XML</a></li>
            <li><a href="">CSS</a></li>
        </ul>
    </li>
    <li><a href="">Blog</a>
        <ul>
            <li><a href="">全部</a></li>
            <li><a href="">网页技术</a></li>
            <li><a href="">UI 技术</a></li>
            <li><a href="">FLASH 技术</a></li>
        </ul>
    </li>
</ul>
```

与前面所不同的是，这里的代码结构涉及嵌套。在第一层的
之间，插入了另一个
的结构，这就是多级菜单的一个代码构成模式。XHTML 代码允许通过嵌套元素来实现想要的效果或是结构，预览效果如图 7-65 所示。

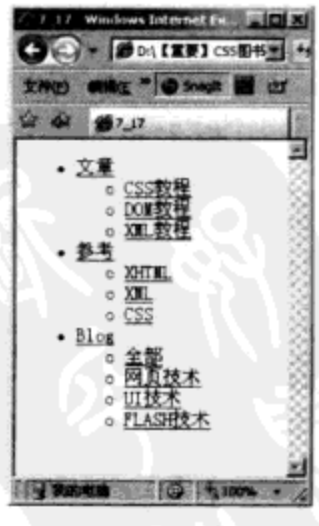


图 7-65　XHTML 结构预览效果

（2）制作 ul 的 CSS 样式，代码如下。

```
ul{
    padding: 0;                    /*设置内边距*/
    margin: 0;                     /*设置外边距*/
    list-style: none;              /*设置列表符号样式*/
}
```

在以上代码中，对导航系统所有 ul 元素进行基本设置，list-style: none 属性能够帮助去掉 ul 中的所有圆点标识。list-style 属性拥有其他更丰富的使用方法，将在后面的列表元素中重点了解。

（3）制作 li 的 CSS 样式，代码如下。

```
ul li{
    float: left;                                    /*设置向左浮动定位*/
    width: 160px;                                   /*设置宽度*/
}
```

在以上代码中，希望导航是横向的，通过 li 设置 float: left 属性。将所有的 li 向左浮动，形成了横向的布局，并尝试使每个 li 的宽度为 160px。制作 ul、li 的 CSS 样式后的效果如图 7-66 所示。

（4）制作 li 标签下的 ul 的 CSS 样式，代码如下。

```
li ul{
    display: none;                                  /*设置块级元素显示方式*/
    top: 20px;                                      /*设置上边界*/
}
```

在以上代码中，li ul 的定义在这里指的是所有 li 下面的 ul 元素。在 XHTML 中，除了顶级的 ul 元素外，所有 li 下面定义的 ul 元素都将受到这部分样式的定义。在这里使用 top 属性设置了整个 ul 的上边框，并使用 display: none 让这部分被隐藏了起来。CSS 中的所有元素基本上都可以使用 display 属性来控制显示的方式，其中 none 表示不显示，是隐藏一个元素的好办法，预览效果如图 7-67 所示。

图 7-66　添加 ul、li 的 CSS 样式后的效果

图 7-67　隐藏元素的预览效果

（5）制作导航项链接文字的 CSS 样式，代码如下。

```
ul li a{
    display: block;                                 /*设置块级元素显示方式*/
    font-size: 12px;                                /*设置字号*/
    border: 1px solid #CCC;                         /*设置边框样式*/
    padding: 3px;                                   /*设置内边距*/
    text-decoration: none;                          /*设置链接文字的文本修饰样式*/
    color: #777;                                    /*设置文字颜色*/
}
ul li a:hover{
    background-color: #F4F4F4;                      /*设置光标经过时的背景颜色*/
}
```

在以上代码中，通过使用 ul li a{}和 ul li a:hover{}的定义，对下拉菜单的样式进行修改，指定了文字的字体、颜色，导航项的边框样式以及内间距等，使它更美观，但是这个时候还没有下拉菜单效果，预览效果如图 7-68 所示。

（6）制作弹出菜单的 CSS 样式，代码如下。

```
li:hover ul,li.over ul{
    display: block;                                    /*设置块级元素显示方式*/
}
```

在以上代码中，li:hover ul 定义了 li 元素的 hover 状态下的 ul 元素。简单地说，是指当光标移上 li 元素时，其下的 ul 元素。

li.over ul 定义了 class 为 over 的 li 元素下的 ul 元素。通过逗号分隔，让这两种情况都使用 display: block 属性，display: block 属性与 display: none 属性是反作用，一个是隐藏，一个是显示。当设置为 display: block 时，不仅其指派的元素将显示，而且将显示为一个块状。如果不设置 display: block 时，元素只会按自己的内容在屏幕上占有的区域进行显示；而使用 display: block 时，元素将自动形成一个方块作为自己的占位符。光标移上 3 个栏目，下拉元素就呈现出来了，预览效果如图 7-69 所示。

图 7-68　没有下拉菜单的效果

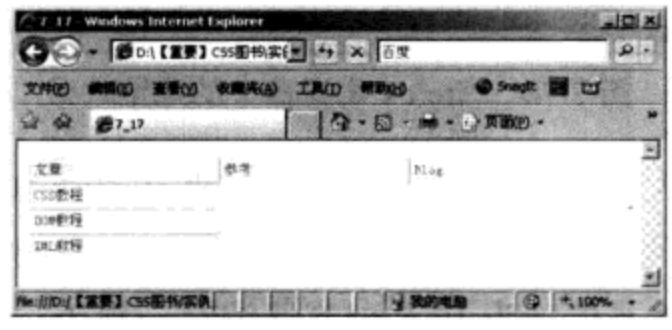


图 7-69　下拉式导航条的预览效果

然而，从理论上说，不加复杂的 JavaScript 脚本，也能够实现这样的一个下拉菜单，原因就在于对 li:hover ul 这里的样式定义。对 CSS 的样式属性来说，当 li 光标移上时，其下的 ul 元素显示为块状。

从语义上，已经满足了所需要的移上显示的目标，但是由于浏览器对 CSS 的部分属性支持的不完善，导致了在 IE 浏览器中无法解析这样一个样式，可以尝试去除 JavaScript 脚本部分的代码，并使用 Firefox 浏览器打开页面，同样可以正常显示。

目前 Firefox 浏览器是公认对 CSS 2.0 支持最完善的浏览器，IE 浏览器因为其捆绑在 Windows 系统中从而有庞大的用户群体，所以有必要针对 IE 浏览器进行特别设置。目前 IE 6 虽然对 CSS 2.0 支持得较为完好，但依然不能够识别 li:hover ul 的定义。

在 CSS 样式表的后面加入了一个叫 li.over ul 的定义，本例中 JavaScript 的含义是，当光标移到所有的 li 元素上时，将 li 的 class 改为 over，即由 JavaScript 的 onMouseOver 事件来触发样式脚本 display: block，实现了预期的效果。相对 Firefox 来说，脚本的编写的确加大了技术难度，不过通过 Firefox 的运行效果可以预见以后的 CSS 控制将是简单方便的。在 IE 7 中，就不需要借助 JavaScript 脚本实现下拉菜单特效了。

7.13.2　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。由于本书所有实例都是使用 IE 7 作为默认浏览器预览的，所以现在需要在其他两个浏览

器再做测试，预览效果。

1．Firefox 浏览器测试

把网页文件放在 Firefox 浏览器中打开，预览效果如图 7-70 所示。

从图 7-65 可以看出，本实例在 Firefox 浏览器下同样正常显示，没有出现兼容性的问题。

2．IE 6 浏览器测试

把网页文件放在 IE 6 浏览器中打开，预览效果如图 7-71 所示。

图 7-70　Firefox 浏览器中的预览效果

图 7-71　IE 6 浏览器中的预览效果

从图 7-66 可以看出，本实例在 IE 6 浏览器下出现问题了，当光标经过导航按钮时，并没有出现下拉菜单的效果，原因就在于 IE 6 对 li:hover ul 代码不能够识别，不支持伪类的交互效果，所以需要单独为 IE 6 浏览器编写一段 JavaScript 代码。

```
<script type="text/javascript" language="javascript">
    navHover = function() {
        var lis = document.getElementById("navlist").getElementsByTagName("LI");
        for (var i=0; i<lis.length; i++) {
            lis[i].onmouseover=function() {
                this.className+=" iehover";
            }
            lis[i].onmouseout=function() {
                this.className=this.className.replace(new RegExp(" iehover\\b"), "");
            }
        }
    }
    if (window.attachEvent) window.attachEvent("onload", navHover);
</script>
```

在以上代码中，它的作用就是当光标移到按钮上的时候，弹出下拉菜单。这个作用和 li:hover ul 在 IE 7 以及 Firefox 浏览器中产生的作用是一致的，再次预览效果如图 7-72 所示。

图 7-72　添加 JavaScript 后在 IE 6 浏览器中的预览效果

7.14　制作多级弹出导航条

多级弹出导航系统，就是在纵向下拉弹出菜单导航基础上，添加了横向弹出菜单。简单地说，就是弹出菜单的复合应用。本节通过一个实例，讲解如何实现多级弹出导航的 CSS 设计。

7.14.1　制作实例

多级弹出导航是由多个类似于下拉式的导航横向或纵向组合而成的。在 7.13 节下拉菜单基础上，再添加横向弹出的菜单，就组成了现在将要介绍的多级弹出导航。

（1）制作多级导航的 XHTML 代码。

```
<ul id="navmenu">
    <li><a href="#">Blog</a></li>
    <li><a href="#">Work +</a>
        <ul>
            <li><a href="#">Websites +</a>
                <ul>
                    <li><a href="#">qrayg</a></li>
                    <li><a href="#">qArcade</a></li>
                    <li><a href="#">qLoM</a></li>
                    <li><a href="#">qDT</a></li>
                </ul>
            </li>
            <li><a href="#">Pen and Ink</a></li>
            <li><a href="#">Free Interfaces</a></li>
        </ul>
    </li>
    <li><a href="#">Learn +</a>
        <ul>
            <li><a href="#">Fireworks +</a>
                <ul>
                    <li><a href="#">aquaButton</a></li>
                    <li><a href="#">aquaButton2</a></li>
                    <li><a href="#">waterDrop</a></li>
                    <li><a href="#">lightFX</a></li>
                    <li><a href="#">lightFX2</a></li>
                </ul>
            </li>
            <li><a href="#">CSS +</a>
                <ul>
                    <li><a href="#">footerStick</a></li>
                    <li><a href="#">spriteNav</a></li>
                    <li><a href="#">@import</a></li>
                </ul>
```

```
            </li>
        </ul>
    </li>
    <li><a href="#">Info</a></li>
    <li><a href="#">Contact</a></li>
</ul>
```

图 7-73 XHTML 的结构预览效果

在以上代码中，依然是使用 ul 结构的菜单构成，但是与 7.13 节讲到单纯的下拉菜单所不同，这次的代码结构进行了 3 级甚至是 4 级嵌套。

在原来下拉菜单的结构代码基础上，将主导航修改成 5 个菜单项，它们分别为 Blog、Works、Learn、Info、Contact。其中 Works、Learn 两个菜单项有下拉菜单项。Works 菜单项下面有 3 个子菜单为 Websites、Pen and Ink、Free Interfaces，其中在 Websites 子目录的
下面分别又嵌套了一次
，构成 3 级嵌套结构，3 级嵌套的菜单项包括 qrayg、qArcade、qLoM、qDT。而同样，在 Learn 导航项下又有两个子目录 Fireworks、CSS，在 Fireworks 又嵌套了一次
，也构成 3 级嵌套结构，3 级嵌套的菜单项包括 aquaButton、aquaButton2、waterDrop、lightFX、lightFX2。

XHTML 代码允许通过多次的嵌套来实现想要的效果或是结构，预览效果如图 7-73 所示。

（2）制作导航系统主体的 CSS 样式，代码如下。

```
body {
    background-color: #CC0000                    /*设置背景色*/
}
ul#navmenu {
    margin: 0;                                   /*设置外边距*/
    border: 0 none;                              /*设置边框样式*/
    padding: 0;                                  /*设置内边距*/
    list-style: none;                            /*设置列表符号样式*/
    height: 24px;                                /*设置高度*/
}
ul#navmenu li {
    margin: 0;                                   /*设置外边距*/
    border: 0 none;                              /*设置边框样式*/
    padding: 0;                                  /*设置内边距*/
    float: left;                                 /*设置向左浮动定位*/
    display: inline;                             /*设置元素内联样式*/
    list-style: none;                            /*设置列表符号样式*/
    position: relative;                          /*设置相对定位*/
    height: 24px;                                /*设置高度*/
```

```
}
ul#navmenu ul {
    margin: 0;                    /*设置外边距*/
    border: 0 none;               /*设置边框样式*/
    padding: 0;                   /*设置内边距*/
    width: 160px;                 /*设置宽度*/
    list-style: none;             /*设置列表符号样式*/
    display: none;                /*设置元素不显示*/
    position: absolute;           /*设置绝对定位*/
    top: 24px;                    /*设置上边界距离*/
    left: 0;                      /*设置左边界距离*/
}
```

在以上代码中，对导航系统所有 ul 元素进行基本设置。list-style: none 属性能够去掉 ul 中的所有圆点标识；margin: 0px 和 padding: 0px 属性去除了 ul 元素默认的边距；position: relative 将元素 ul 设置成相对定位元素，预览效果如图 7-74 所示。

图 7-74　将导航设置成水平方向

（3）制作导航项的 CSS 样式，代码如下。

```
ul#navmenu a {
    border: 1px solid #FFF;                                          /*设置边框样式*/
    border-right-color: #CCC;                                        /*设置右边框颜色*/
    border-bottom-color: #CCC;                                       /*设置下边框颜色*/
    padding: 0 6px;                                                  /*设置内边距*/
    display: block;                                                  /*设置元素显示方式*/
    background: #EEE;                                                /*设置背景颜色*/
    color: #666;                                                     /*设置文字颜色*/
    font: bold 10px/22px Verdana, Arial, Helvetica, sans-serif;      /*设置字体样式*/
    text-decoration: none;                                           /*设置链接文字样式*/
}
```

在以上代码中，主导航是横向的，对导航项进行了样式设置，使每个导航项从视觉上看起来像是一个可以交互的按钮，有下拉项的后边使用一个“+”，表示可以弹出下拉菜单。本步骤主要就是对导航项进行美化工作，预览效果如图 7-75 所示。

（4）制作导航项的一级菜单添加光标交互的 CSS 样式，代码如下。

```
ul#navmenu a:hover, ul#navmenu li:hover a, ul#navmenu li.iehover a {
    background: #CCC;                                                /*设置链接区域的背景色*/
    color: #FFF;                                                     /*设置链接文字的字体颜色*/
}
```

在以上代码中，使用包含选择符、群组选择符以及伪类，为导航项的主菜单、甚至是以后将要添加的二级、三级导航项添加了光标交互效果。这种交互效果是统一的，都是光标移过时，按钮变成灰色。这里先制作了一级导航，预览效果如图 7-76 所示。

图 7-75　对导航项进行美化

图 7-76　光标经过 Learn 导航项时候的交互效果

（5）制作二级、三级菜单项的 CSS 样式，代码如下。

```
    ul#navmenu li:hover li a, ul#navmenu li.iehover li a {
        float: none;                                          /*设置不浮动*/
        background: #EEE;                                     /*设置背景色*/
        color: #666;                                          /*设置文字颜色*/
    }
    ul#navmenu li:hover li a:hover, ul#navmenu li:hover li:hover a, ul#navmenu li.iehover li a:hover,
ul#navmenu li.iehover li.iehover a {
        background: #CCC;                                     /*设置背景色*/
        color: #FFF;                                          /*设置文字颜色*/
    }
    ul#navmenu li:hover li:hover li a, ul#navmenu li.iehover li.iehover li a {
        background: #EEE;                                     /*设置背景颜色*/
        color: #666;                                          /*设置文字颜色*/
    }
    ul#navmenu li:hover li:hover li:hover li a:hover, ul#navmenu li:hover li:hover li:hover a, ul#navmenu
li.iehover li.iehover li a:hover, ul#navmenu li.iehover li.iehover li.iehover a {
        background: #CCC;                                     /*设置背景颜色*/
        color: #FFF;                                          /*设置文字颜色*/
    }
    ul#navmenu li:hover li:hover li:hover li a, ul#navmenu li.iehover li.iehover li.iehover li
a {
        background: #EEE;                                     /*设置背景色*/
        color: #666;                                          /*设置文字颜色*/
    }
```

在以上代码中，分别对二级、三级导航项设置了默认样式以及光标经过的样式，同样使用了包含选择符、组合选择符以及伪类来定义 CSS 的样式。

（6）制作弹出的交互效果 CSS 样式，代码如下。

```
    ul#navmenu li:hover ul ul, ul#navmenu li:hover ul ul ul, ul#navmenu li.iehover ul ul, ul#navmenu
li.iehover ul ul ul {
        display: none;
    }
    ul#navmenu li:hover ul, ul#navmenu ul li:hover ul, ul#navmenu ul ul li:hover ul, ul#navmenu
li.iehover ul, ul#navmenu ul li.iehover ul, ul#navmenu ul ul li.iehover ul {
        display: block;
    }
```

在以上一段代码中，由于所有的弹出效果的原理一样，都是在 CSS 样式中将其设置为 display: block，所以使用了组合选择符，一并将其统一设置了样式。

在 IE 7 中，弹出效果的交互 CSS 就可以完成，所以不需要使用 JavaScript，预览效果如图 7-77 所示。

图 7-77　多级弹出导航条的预览效果

7.14.2　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。这么一个复杂的导航系统，经常会在编写 CSS 样式的时候出现浏览器兼容问题，很有必要在多种浏览器下做测试并且调试。

1．Firefox 浏览器测试

把网页文件放在 Firefox 浏览器中打开，预览效果如图 7-78 所示。

从图 7-71 可以看出，在 Firefox 浏览器中也可以正常显示，同样弹出的交互效果也正常，证明在 Firefox 浏览器下不存在兼容问题。

2．IE 6 浏览器测试

把网页放在 IE 6 浏览器中打开，预览效果如图 7-79 所示。

从图 7-72 可以看出，在 IE 6 浏览器中这个多级弹出导航系统发生了问题，光标经过时，并没有出现弹出项。这是因为 CSS 的伪类在 IE 6 下并不支持，所以使用 CSS 制作的交互效果在 IE 7 以及 Firefox 浏览器下都可以正常显示，但是在 IE 6 下就无能为力了。

图 7-78　Firefox 浏览器中的预览效果

图 7-79　IE 6 浏览器中的预览效果

针对这一问题，现在专门为 IE 6 添加 JavaScript 代码，代码添加在<head></head>头文件里面，具体代码如下。

```
<script language="javascript" type="text/javascript">
    navHover = function() {
        var lis = document.getElementById("navmenu").getElementsByTagName("LI");
        for (var i=0; i<lis.length; i++) {
```

```
            lis[i].onmouseover=function() {
                this.className+=" iehover";
            }
            lis[i].onmouseout=function() {
                this.className=this.className.replace(new RegExp(" iehover\\b"), "");
            }
        }
    }
    if (window.attachEvent) window.attachEvent("onload", navHover);
</script>
```

由于本书的重点并不在于讲解 JavaScript 语言，所以这里的 JavaScript 代码也不做具体的讲解，读者可以参考 JavaScript 相关书籍进行研究，这里只是提供一段代码来为本实例服务，实现想要的效果。

添加了这段代码以后，在 IE 6 浏览器中重新预览的效果如图 7-80 所示。

图 7-80　添加了 JavaScript 代码后在 IE 6 浏览器中的预览效果

这样，完成了在 3 个浏览器下的测试并解决了不同浏览器所产生的兼容问题。

7.15　小结

本章讲解了使用 CSS 属性制作网页导航系统的实例，包括简单的水平导航条、方块导航条、标签式导航条、按钮导航条、会动的长城形导航条、基于背景控制的导航条、左右自由伸展的导航条、纵向导航条、分行导航条、增亮导航条、动感导航条、下拉式导航条及多级弹出式导航条。在讲解实例的过程中，介绍了很多 CSS 属性的使用方法，主要涉及以下 6 个属性。

- display 属性。
- 文本对齐属性 text-align。
- 边框属性 border。
- 内边距属性 padding。
- 元素定位属性 position。
- z-index 属性。

讲解到的 CSS 技巧有以下两个。

- CSS 盒模型。
- 九宫格技术。

对本章的知识点前后联系进行归纳总结，知识点结构导图如图 7-81 所示。

图 7-81　本章知识点结构导图

7.16　习题

1. 网站导航的作用是什么？常用的导航类型有哪些？
2. 什么是 CSS 盒模型？简述盒模型尺寸的计算原理。
3. 简述九宫格技术的原理。
4. 尝试制作一个简单的动感导航条。

第8章 使用 CSS 制作列表

列表元素是网页设计中使用频率非常高的元素，在大多数的网站设计上，无论是新闻列表，还是产品，或者是其他内容，均需要以列表的形式来体现。

列表形式在网站设计中占有很大比重，信息的显示非常整齐直观，便于用户理解与点击。从网页出现到现在，列表元素一直是页面中非常重要的应用形式。

8.1 制作新闻列表

新闻列表是一个网站的重要组成元素。对新闻列表使用标题，可以突出显示很重要的信息。新闻列表既可以是基于文本的内容，也可以是指向现有新闻项目的链接（例如新闻服务上的文章）。新浪网的新闻列表如图 8-1 所示。

网站设计中经常碰到新闻列表，如何处理它的外观显得尤为重要。这个实例中，制作一个基本的新闻列表，将标题与日期制作成单独的浮动，并且使链接的光标在悬停时，文字呈现出不同的色彩变化，如图 8-2 所示。

图 8-1 新浪网的新闻列表

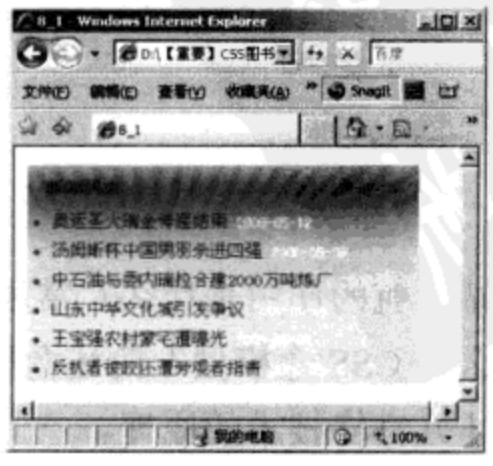


图 8-2 新闻列表实例

8.1.1　制作实例

（1）制作页面的 XHTML 代码。

```
<div id="news">
    <h3>新闻列表</h3>
    <ul id="pagelist">
        <li><a href="#"><span class="lbt">奥运圣火瑞金传递结束</span>
        <span class="ldt">2008-05-12</span></a></li>
        <li><a href="#"><span class="lbt">汤姆斯杯中国男羽杀进四强 </span>
        <span class="ldt">2008-05-09</span></a></li>
        <li><a href="#"><span class="lbt">中石油与委内瑞拉合建 2000 万吨炼厂 </span>
        <span class="ldt">2008-05-06</span></a></li>
        <li><a href="#"><span class="lbt">山东中华文化城引发争议 </span>
        <span class="ldt">2008-05-06</span></a></li>
        <li><a href="#"><span class="lbt">张三农村豪宅遭曝光 </span>
        <span class="ldt">2008-05-05</span></a></li>
        <li><a href="#"><span class="lbt">反扒者被咬还遭旁观者指责 </span>
        <span class="ldt">2008-05-05</span></a></li>
    </ul>
</div>
```

以上代码分为两个部分：标题和新闻列表。<h3>元素是标题部分，<ul>元素及其内部的<li>元素是新闻列表内容。新闻列表项分为新闻标题和新闻时间，分别由元素包含着。如图 8-3 所示为没有添加 CSS 样式的新闻列表。

（2）制作#news 元素的 CSS 样式，代码如下。

```
#news {
    background-image: url(images/8_01_01.jpg);          /*添加背景图片*/
    background-repeat: no-repeat;                       /*设置背景不平铺*/
}
```

在以上代码中，新闻列表框架使用了一个背景图片，如图 8-4 所示，同时设置了背景不平铺的样式。

图 8-3　没有添加 CSS 样式的新闻列表

图 8-4　新闻列表背景图

（3）制作#news h3 元素的 CSS 样式，代码如下。

```
#news h3{
    color: #C63D00;                          /*设置文字颜色*/
    font-size: 14px;                         /*设置字号*/
    padding-left: 15px;                      /*设置左内边距*/
    padding-top: 5px;                        /*设置上内边距*/
    line-height: 30px;                       /*设置行高 30px*/
    margin: 0px;                             /*重新设置了外边距*/
}
```

在以上代码中，定义了列表标题的文字颜色、字号、行高，以及 h3 元素的外边距和部分内边距，使得文字能在合适位置显示。这里重新设置 h3 标签的外边距，新的外边距为 0px，预览效果如图 8-5 所示。

（4）制作新闻列表项的 CSS 样式，代码如下。

```
#pagelist{
    list-style-type: disc;                   /*设置列表类型*/
    margin: 0px;                             /*设置外边距*/
    padding: 0px;                            /*重新设置内边距*/
    margin-left: 20px;                       /*重新设置左外边距，覆盖之前设置的左外边距*/
}
#pagelist li {
    width:330px;                             /*设置列表项的宽度*/
    line-height: 24px;                       /*设置行高*/
    font-size: 14px;                         /*设置字号*/
    color:#C63D00;                           /*设置文字颜色*/
}
#pagelist li a {
    color:#C63D00;                           /*设置链接文字颜色*/
}
```

在以上代码中，使用类选择符，在#pagelist 元素中定义了列表符号样式为小圆点，内边距和外边距都指定为 0px，是为了去除 ul 元素在不同浏览器中可能出现的外边距或内边距问题。最后又重新定义一次左边距为 20px，是为了显示列表符号。在#pagelist li 元素中，定义了列表项的宽度、行高、字号及文字颜色。在#pagelist li a 元素中，定义了链接的颜色，预览效果如图 8-6 所示。

（5）制作新闻标题.lbt 元素的 CSS 样式，代码如下。

```
#pagelist li a .lbt {
    display: inline;                         /*设置元素内联显示方式*/
    width:255px;                             /*设置宽度*/
    text-decoration:none;                    /*设置链接文字的样式*/
}
#pagelist li a:hover .lbt {
    text-decoration: underline;              /*设置链接文字光标经过时的样式*/
}
```

在以上代码中，为了将新闻标题和新闻时间区别开来，这里去掉了新闻标题链接的下划线样式，新闻标题通过 span 元素来控制样式，通过包含选择符#pagelist li a .lbt 指定了 class 为 ibt 的 span 元素的显示方式为 inline 内联显示，预览效果如图 8-7 所示。

图 8-5　添加标题的 CSS 样式效果

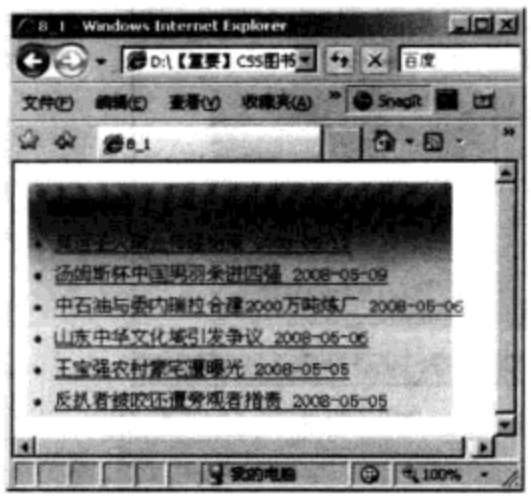


图 8-6　为新闻标题添加 CSS 样式的效果

（6）制作时间.ldt 元素的 CSS 样式，代码如下。

```
#pagelist li a .ldt {
    display: inline;                                    /*设置内联元素显示方式*/
    width:75px;                                         /*设置宽度*/
    text-align:center;                                  /*设置文本居中*/
    color:#FFF;                                         /*设置文字颜色*/
    text-decoration:none;                               /*设置文本修饰样式*/
    font-size: 12px;                                    /*设置字号*/
}
```

在以上代码中，为了进一步区分新闻标题和时间的样式，又设置了时间的 CSS 样式。新闻时间通过 class 为 ldt 的 span 元素来控制，通过包含选择符#pagelist li a .ldt 指定了 span 元素为内联显示方式，同时还指定了链接文字的颜色、字号等属性，最终预览效果如图 8-8 所示。

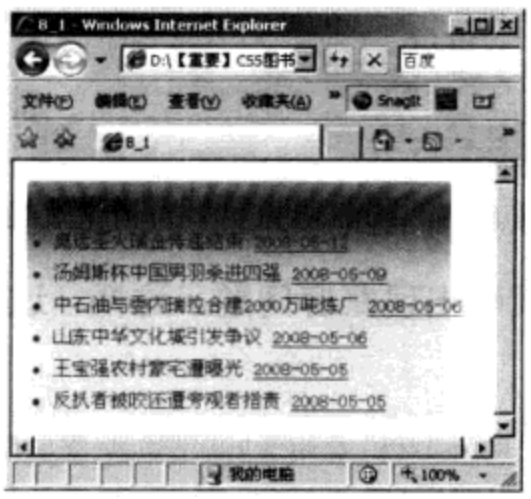


图 8-7　取消了新闻标题链接下划线的效果

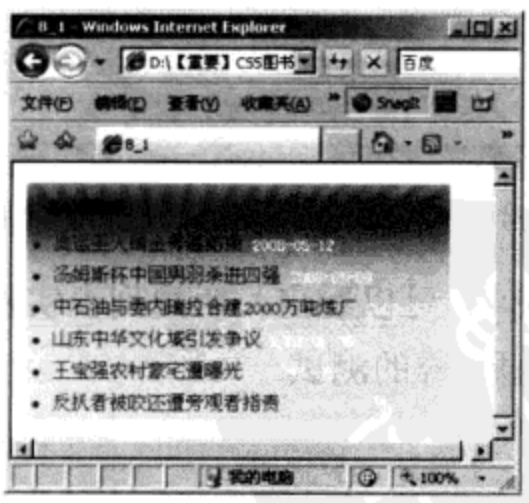


图 8-8　新闻列表实例

8.1.2　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中进行测试。

1．Firefox 浏览器测试

把网页文件放在 Firefox 浏览器中打开，预览效果如图 8-9 所示。

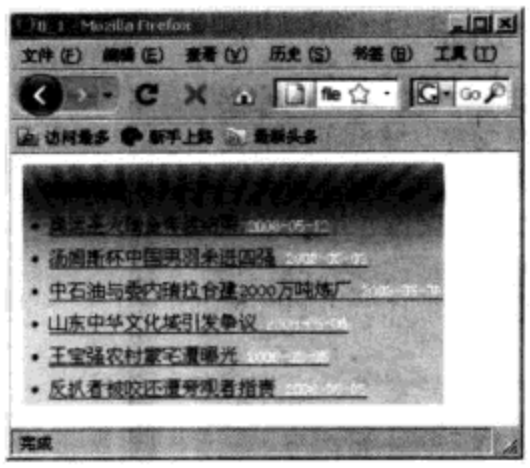


图 8-9　Firefox 浏览器中的预览效果

从图 8-9 可以看出，本实例在 Firefox 浏览器中出现问题了，列表文字的下划线依然还显示。通过观察发现，出现问题的原因是 li 的链接文字样式，在包含选择符#pagelist li a 里面没有对文本样式做特殊设置，导致在标准的浏览器 Firefox 下按照默认的样式显示，而 IE 浏览器在没有设置 CSS 文本样式时，则认为是不显示。

修改代码如下。

```
#pagelist li a {
    color:#C63D00;
    text-decoration: none;
}
```

在以上代码中，只是针对包含选择符#pagelist li a 里面的 CSS 添加了一段文本修饰属性，特殊强调了去掉下划线样式，预览效果如图 8-10 所示。

2．IE 6 浏览器测试

把网页文件放在 IE 6 浏览器中打开，预览效果如图 8-11 所示。

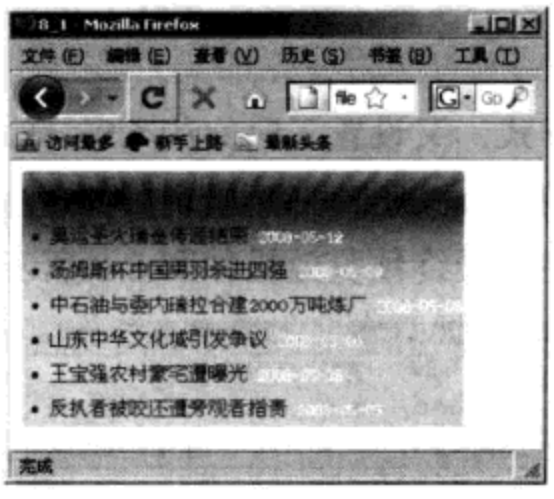


图 8-10　解决兼容问题后的预览效果

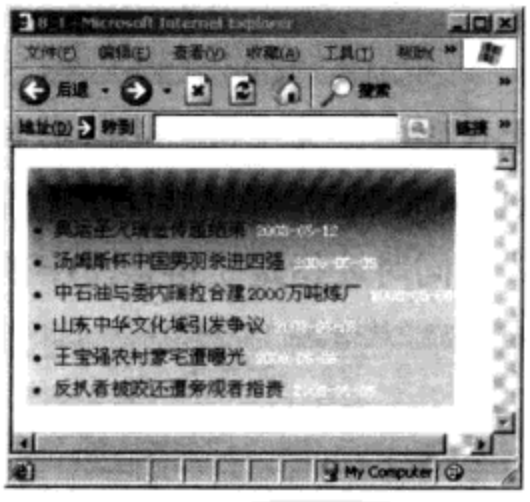


图 8-11　IE 6 浏览器中的预览效果

从图 8-11 可以看出，新闻列表实例在 IE 6 浏览器中没有出现兼容问题。这样，这个实例完成了浏览器兼容的测试。

8.2　制作排行榜

排行榜是一个网站经常会使用到的元素。在网页中设计一个信息统计的排行榜，用来统计某种信息总数的多少。图 8-12 是百度的搜索排行榜页面，里面包括了各种各样新颖独特的排行榜。

图 8-12　百度搜索排行榜页面

本节制作的实例，将电影的票房数量设计成一个排行榜，使用有序列表 ol 元素对票房进行自动排序。

（1）制作页面的 XHTML 代码。

```
<div id="rank">
    <h3>票房排行榜</h3>
    <ol class="decimal">
        <li>《钢铁侠》Iron Man</li>
        <li>《极速赛车手》Speed Racer</li>
        <li>《情迷赌城》What Happens in Vegas</li>
        <li>《新郎不是我》Made of Honor</li>
        <li>《代孕妈妈》Baby Mama</li>
        <li>《忘掉负心女》Forgetting Sarah...</li>
        <li>《寻堡奇遇 2》Harold and Kumar...</li>
        <li>《功夫之王》The Forbidden Kingdom</li>
        <li class="clearit">《尼姆岛》Nim's Island</li>
    </ol>
</div>
```

在以上代码中，排行榜分为两个部分：标题和排行榜列表。h3 标签用来控制标题的内容，有序列表 ol 元素用来控制排行榜列表的内容，结构效果如图 8-13 所示。

（2）制作#rank 元素的 CSS 样式，代码如下。

```
#rank{
    background-image: url(images/8_2.jpg);          /*设置背景图片*/
    background-repeat: no-repeat;                   /*设置背景图片不平铺*/
    height: 264px;                                  /*设置高度*/
    width: 312px;                                   /*设置宽度*/
}
```

在以上代码中，定义了排行榜框架的背景图片，背景不平铺，预览效果如图 8-14 所示。

图 8-13　没有添加 CSS 样式的结构效果

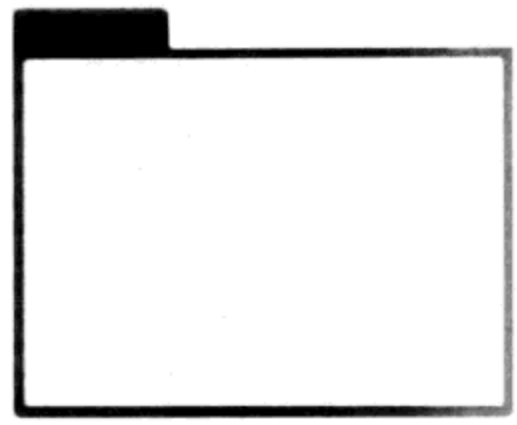


图 8-14　排行榜背景图片

同时这段代码还定义了背景的宽度和高度，使得背景能够完全显示出来。

（3）制作标题的 CSS 样式，代码如下。

图 8-15　标题添加 CSS 样式的效果

```
#rank h3 {
    color: #fff;                    /*设置 h3 标签内的文字颜色*/
    font-size: 14px;                /*设置 h3 标签内的字号*/
    padding-left: 12px;             /*设置 h3 标签的左内边距*/
    padding-top: 3px;               /*设置 h3 标签的上内边距*/
    line-height: 30px;              /*设置 h3 标签内文本的行高*/
    margin: 0px;                    /*设置 h3 标签的外边距*/
}
```

在以上代码中，定义了标题文字的颜色、字号、行高以及 h3 元素的外边距和部分内边距，使得标题文字在适当位置显示，页面更美观。添加了标题的 CSS 样式效果如图 8-15 所示。

（4）制作 ol 的 CSS 样式，代码如下。

```
ol.decimal {
    list-style-type: decimal;                    /*设置列表项符号*/
    margin: 0px;                                 /*设置外边距*/
    padding: 0px;                                /*设置内边距*/
    margin-left: 40px;                           /*重新设置左外边距，覆盖之前的样式*/
}
```

在以上代码中，定义了有序列表 ol 的样式。list-style-type 属性是本实例的关键所在，指定了有序列表的编号样式为阿拉伯数字 1、2、3…。外边距和内边距都指定为 0px，去除了列表元素默认的边距，最后又重新定义了左外边距为 40px，使得序列编号能够显示在可视区域中。

（5）制作列表项的 CSS 样式，代码如下。

```
ol.decimal li{
    color: #15589F;                              /*设置列表项文字的颜色*/
    line-height: 24px;                           /*设置文字行高*/
```

```
    margin-right: 40px;                          /*设置列表项的右外边距*/
    border-bottom: 1px dotted #15589F;           /*设置下边框样式*/
    font-size: 12px;                             /*设置字号*/
}
```

在以上代码中，使用包含选择符 ol.decimal li 定义了排行榜列表项内容的样式。这里面除了定义文字的颜色、字号和行高以外，还特别地定义了 li 元素的下边框样式为 1px、颜色为#15589F 的虚线，使排行榜更美观。margin-right 属性用来控制下边框的长度，预览效果如图 8-16 所示。

（6）制作 ol.decimal li.clearit 元素的 CSS 样式，代码如下。

```
ol.decimal li.clearit{
    border: none;                                /*设置边框样式*/
}
```

以上代码是专门为列表最后一项所定义的样式，去除 li 元素边框线。在排行榜榜单中，“《尼姆岛》Nim's Island”使用的就是这个样式，使得整体效果更美观。这样，排行榜列表的实例就制作完成，预览效果如图 8-17 所示。

图 8-16　列表项添加 CSS 样式的效果

图 8-17　排行榜预览效果

8.3　制作自定义图片项目符号的列表

有时候，CSS 默认的列表效果并不能满足设计师对于列表中项目符号的要求，需要用小一点的圆点，或是其他更具代表性的图片箭头来作为项目符号。例如新浪网的新闻列表就是使用自定义符号的列表，如图 8-18 所示。

对此，CSS 提供了图片替换技术，可以选用符合页面风格的图片符号来替代默认效果。同样 XHTML 代码不需要更改，只需要使用 CSS 提供的 list-style-image 属性完成图片替代项目符号。

图 8-18　自定义项目符号的列表

8.3.1　使用列表符号样式属性制作列表

首先来学习使用列表图片样式 list-style-image 属性制作列表。下面分别讲解列表图片样式属性的使用方法和具体实例应用。

1．列表图片样式属性语法结构

列表图片属性 list-style-image 是使用一幅图像来替换列表项的标记，语法结构如下。

```
list-style-image: url(图片地址)| none;
```

这个属性指定作为一个有序或无序列表项标志的图像。list-style-image 属性需要提供的值只有一个，就是图片路径，图像相对于列表项内容的放置位置通常使用 list-style-position 属性控制。

2．制作自定义图片项目符号的列表实例

现在使用列表图片样式属性制作列表实例，具体制作步骤如下。

（1）制作页面的 XHTML 代码。

```
<div id="listPan">
    <h2><span>All Galleries</span></h2>
    <div id="list_ulPan">
        <ul>
            <li><a href="#">Art Icons</a> <span>(14)</span></li>
            <li><a href="#">Application Icons</a> <span>(14)</span></li>
            <li><a href="#">Animal Icons</a> <span>(16)</span></li>
            <li><a href="#">Business icons</a> <span>(7)</span></li>
            <li><a href="#">Cartoon Icons</a> <span>(5)</span></li>
            <li><a href="#">Computer icons</a> <span>(12)</span></li>
            <li><a href="#">Food icons</a> <span>(6)</span></li>
            <li><a href="#">Funny Icons</a> <span>(19)</span></li>
        </ul>
    </div>
</div>
```

在以上代码中，列表分为两个部分：列表标题和列表项，结构效果如图 8-19 所示。

（2）制作#listPan 元素的 CSS 样式，代码如下。

```
#listPan {
    width:210px;                          /*设置宽度*/
    height:250px;                         /*设置高度*/
    float:left;                           /*设置向左浮动定位*/
    background-color: #2E2E2E;            /*设置背景颜色*/
    color:#7D7D7D;                        /*设置文字颜色*/
}
```

以上代码定义了列表框架的宽度和高度，以及背景颜色、文字颜色属性。

（3）制作标题的 CSS 样式，代码如下。

```
#listPan h2 {
    height:30px;                          /*设置高度*/
    font:20px Arial;                      /*设置字体样式*/
    color:#FFF;                           /*设置文字颜色*/
    margin:20px 0 0 10px;                 /*设置外边距*/
}
```

以上代码定义了列表标题的字号、字体类型和文字颜色，h2 元素的高度以及外边距，使得标题在适当位置显示，效果如图 8-20 所示。

（4）制作#list_ulPan 元素的 CSS 样式，代码如下。

```
#list_ulPan {
    width:160px;                          /*设置宽度*/
    margin:0 0px;                         /*设置外边距*/
}
#listPan ul {
    list-style-image: url(images/folder.jpg);          /*设置列表项符号为图片*/
}
```

以上代码使用普通选择符和包含选择符两种方式定义了列表项的样式。在#list_ulPan 元素中，定义了宽度和外边距。在#listPan ul 元素中，使用列表图片样式属性 list-style-image 定义了列表项目的图片，这是本实例的关键所在，列表项目的图片如图 8-21 所示。

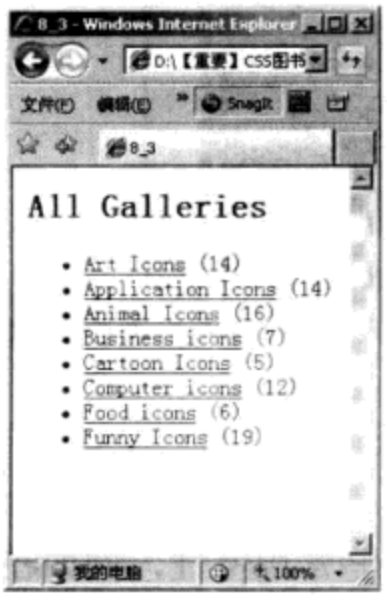


图 8-19　列表结构效果

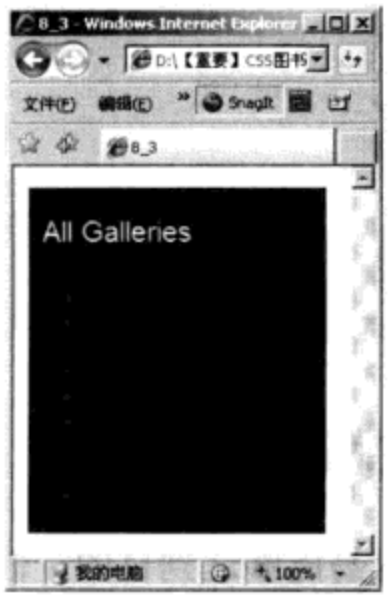


图 8-20　添加背景和标题样式的效果

图 8-21　列表项目图片

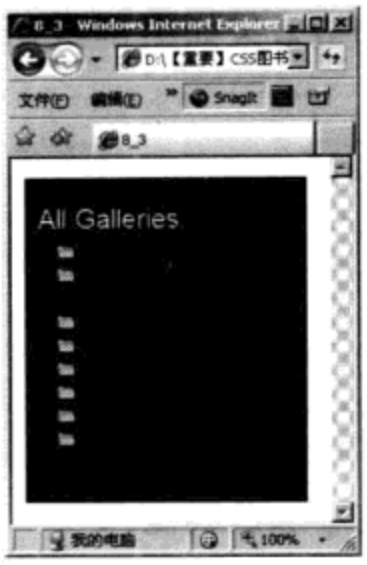


图 8-22　添加了自定义项目符号的预览效果

只需要一行代码，图片就被放置在了列表前面，预览效果如图 8-22 所示。

（5）制作列表项的 CSS 样式，代码如下。

```
#listPan ul li {
    display:block;              /*设置块级显示方式*/
    width:160px;                /*设置宽度*/
    height:21px;                /*设置高度*/
    font-size:13px;             /*设置字号*/
    font-weight:bold;           /*设置文字为粗体*/
    font-family: Arial;         /*设置字型*/
    color:#7D7D7D;              /*设置文字颜色*/
    padding:0px;                /*设置内边距*/
}
```

以上代码定义了列表项为块状显示方式，目的是为其指定宽度和高度。同时指定了字号、颜色、类型等属性，去除了 li 元素默认的内边距。调整列表项的 CSS 样式，效果如图 8-23 所示。

（6）制作链接文字的 CSS 样式，代码如下。

```
#listPan ul li a {
    color:#7D7D7D;                          /*设置链接文字颜色*/
    text-decoration:none;                   /*设置文本样式*/
    padding-left: 20px;                     /*设置左内边距*/
}
#listPan ul li a:hover {
    color:#09A5F6;                          /*设置链接文字光标经过时的文字颜色*/
    text-decoration:none;                   /*设置链接文字的文本样式*/
}
```

以上代码使用伪类 a 元素定义了链接文字的 CSS 样式。这样，自定义图片项目符号的列表实例就制作完成了，预览效果如图 8-24 所示。

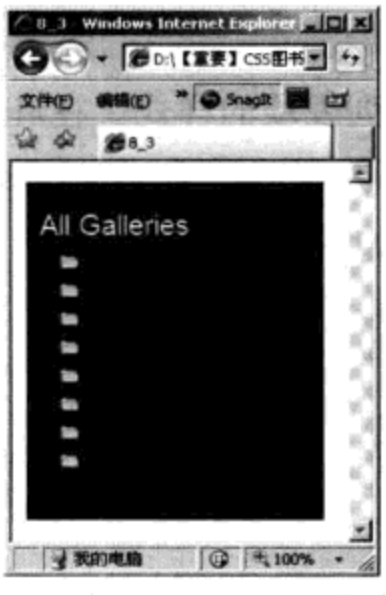


图 8-23　调整了列表项的样式效果

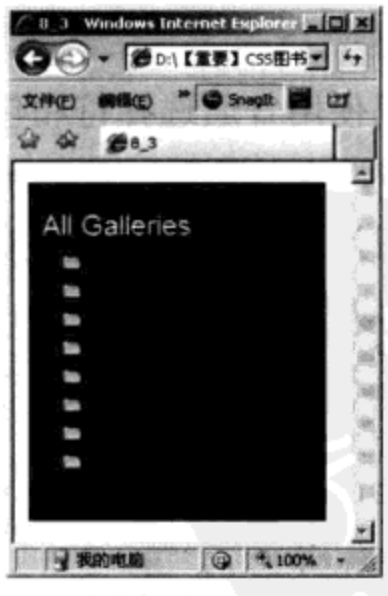


图 8-24　自定义图片项目符号的列表预览效果

8.3.2　使用背景图片属性制作列表符号

虽然使用 list-style-image 很容易能够达到目的，但是也失去了一些常用特性，例如在 ul 的各项属性中，并不能够精确控制图片替换的项目符号距文字的距离。如果有特殊的要求，例如

把项目符号作为一个结束符放置在列表项目的右边，就无能为力了。

制作列表并不是只能通过 list-style 这样一组属性来实现，前面探讨过的背景属性在这个时候也能派上用场。ul 元素和其他元素一样，支持 CSS 中的大部分共用属性，用于设置背景的 background-color、background-image 等一组背景属性同样可以使用在 ul 元素中，因此完全可以使用背景来替代 list-style 的项目符号，而且背景控制中拥有对背景的精确定位属性 background-position，可以利用背景制作精确定位的项目符号。

重新编写 CSS 样式，代码如下。

```
#listPan ul {
    list-style-type: none;                          /*设置列表符号类型*/
}
#listPan ul li a {
    color:#7D7D7D;                                  /*设置链接文字颜色*/
    text-decoration:none;                           /*设置链接文字样式*/
    background-repeat: no-repeat;                   /*设置背景图片不平铺*/
    background-position: left center;               /*设置背景图片位置*/
    Padding-top:3px;                                /*设置上边距*/
    padding-left: 20px;                             /*设置左内边距*/
    background-image: url(images/folder.jpg);       /*设置背景图片*/
}
```

首先使用 list-style-type: none;取消了默认的圆点项目符号，然后对 a 元素定义了一个不平铺的背景，并设置了在每个 a 中的位置，距上边 3px 高度。而对于 li 而言，为了防止 li 中的文字压住了背景，将 li 元素的左内边距设置成了 20px，使背景图片可以展示出来。预览效果与直接使用 list-style-image 属性完全一致，如图 8-25 所示。

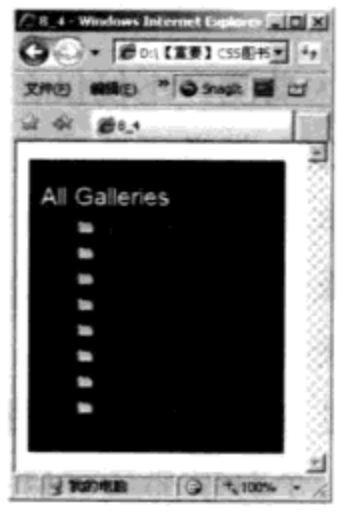


图 8-25　使用背景属性设计列表项目图片

8.3.3　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。

1．Firefox 浏览器测试

把 8.3.1 小节中的实例所用到的网页文件放在 Firefox 浏览器中打开，预览效果如图 8-26 所示。

从图 8-26 可以看出，本实例在 Firefox 浏览器中出现问题了，列表符号的图片没有显示出来。原因是 list-style-image 属性在有些浏览器中无法正常显示或者错位，解决的办法就是使用背景图替换的方法实现列表符号图像的效果，正如 8.3.2 小节使用的方法。

2．IE 6 浏览器测试

把 8.3.1 小节的实例中的网页文件放在 IE 6 浏览器中打开，预览效果如图 8-27 所示。

从图 8-27 可以看出，新闻列表实例在 IE 6 浏览器中没有出现兼容问题。这样，这个实例就完成了浏览器兼容的测试。

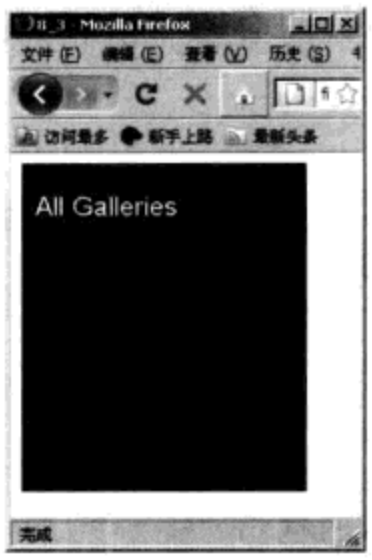


图 8-26　Firefox 浏览器中的预览效果

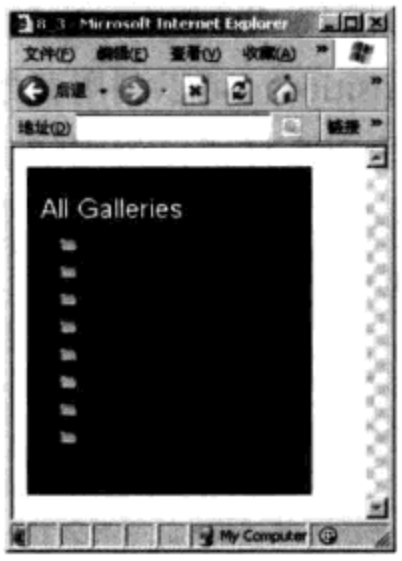


图 8-27　IE 6 浏览器中的预览效果

8.4　使用 CSS 改变列表排版

CSS 的强大功能，不仅仅在于它能够定义网页的样式，它还能实现网页的结构与表现的分离（可以通过 CSS 任意更改表现样式，而不改变结构）。

在一段文本中，使用列表来辅助排版，能使阅读更加轻松。列表元素一直都是一种便于阅读与理解的排版形式，但是有时候排好的列表因为某些文章要求而不得不改变阅读形式，这就需要直接把列表变为段落。

前面探讨过 display 属性的用法，display 属性控制的是一个对象的显示方式，而作为 XHTML 中的成员，每一个对象都具有自身默认的显示方式。对 div 而言，div 默认作为一个 block 的显示方式，而对 a 及 span 而言，它们则是 inline 显示方式。XHTML 中的大部分可视对象默认方式都是基于 block 及 inline 显示方式。

为了使列表元素能够显示为段落，尝试转换 ul 及其中的 li 的 display 类型为 inline 对象。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    <h3>在这一章中，我们主要探讨如下几个方面</h3>
    <ul id="list">
        <li><span>新闻列表</span></li>
        <li><span>排行榜列表</span></li>
        <li><span>自定义图片项目符号的列表</span></li>
        <li><span>列表的缩进排版</span></li>
    </ul>
    <h3 class="rePos">以及其他更富创造性的列表设计</h3>
</div>
```

以上代码指定了段落文字和列表文字。

（2）制作列表排版的 CSS 样式，代码如下。

```
#layout{
    background: url(images/8_4.jpg) no-repeat;    /*设置页面背景图片*/
    width: 587px;                                 /*设置宽度*/
    height: 339px;                                /*设置高度*/
    color: #0099CC;                               /*设置文字颜色*/
}
#layout h3{
    font-size: 14px;                              /*定义字号*/
    padding-top: 40px;
    padding-left: 85px;                /*设置元素的内边距，文字显示在适当位置，使页面更美观*/
}
#list{
    font-size: 14px;
    padding-left: 65px;                           /*设置元素的内边距，和 h3 元素文本对齐*/
}
#layout h3.rePos{
    padding-top: 5px;                             /*设置上内边距*/
}
```

在以上代码中，#layout 元素定义了页面的背景，背景图片如图 8-28 所示。定义了颜色和列表的宽度和高度，用以完整显示背景图片。在列表符号样式定义中，这里使用的是默认的小圆点符号样式，没有特别指定样式。使用包含选择符#layout h3.rePos，重新定义了 h3 元素的上内边距，使排版更美观。

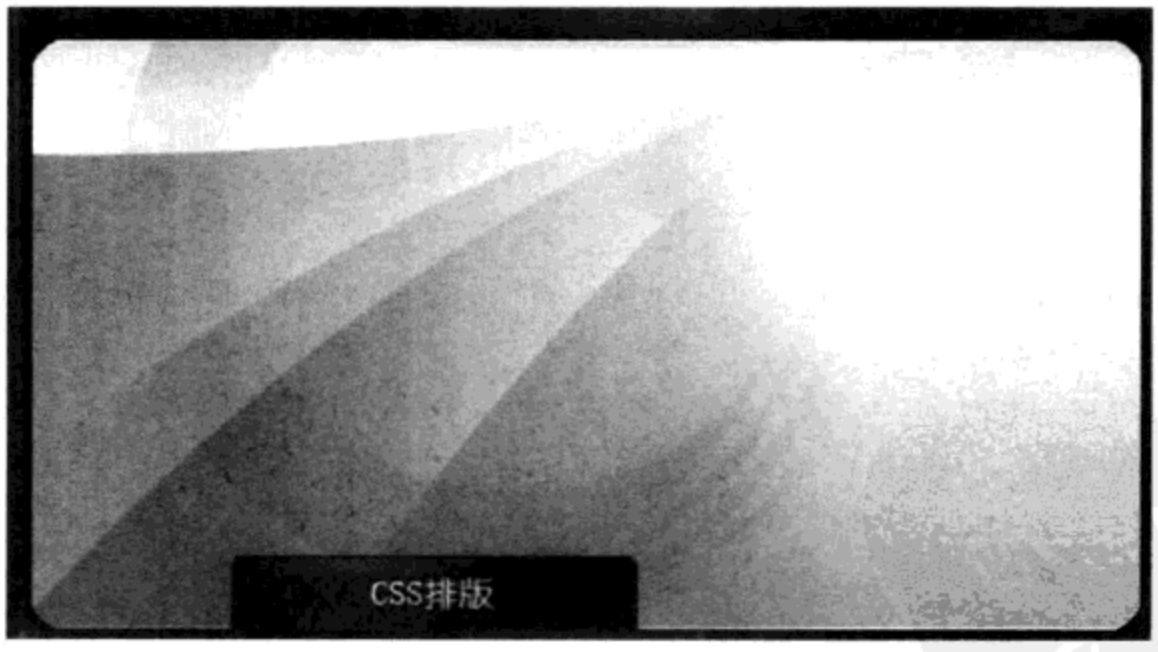


图 8-28　列表背景图片

这样列表排版的段落就制作完成，预览效果如图 8-29 所示。

（3）制作段落排版的 CSS 样式，代码修改如下。

```
#list{
    display: inline;                                  /*设置内联元素显示方式*/
}
#list li{
```

```
    display: inline;                                    /*设置内联元素显示方式*/
}
```

预览效果如图 8-30 所示。

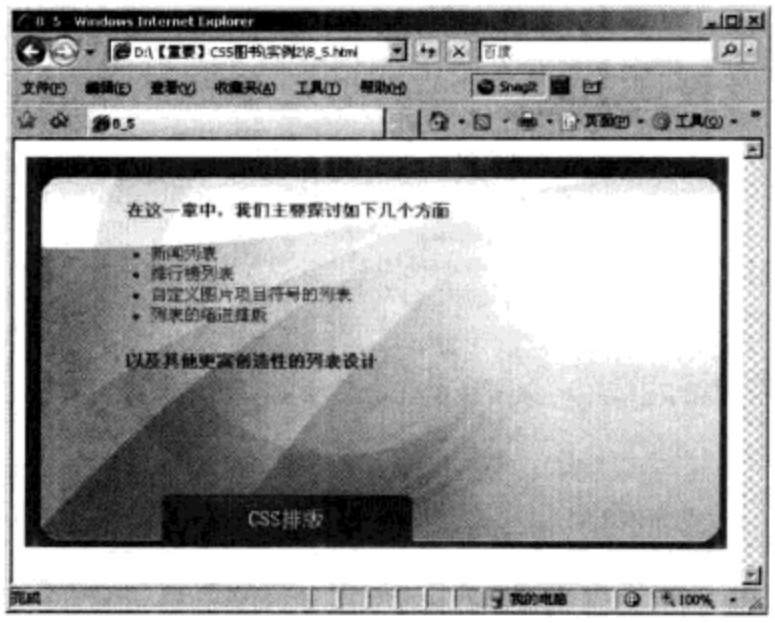


图 8-29　列表排版的效果

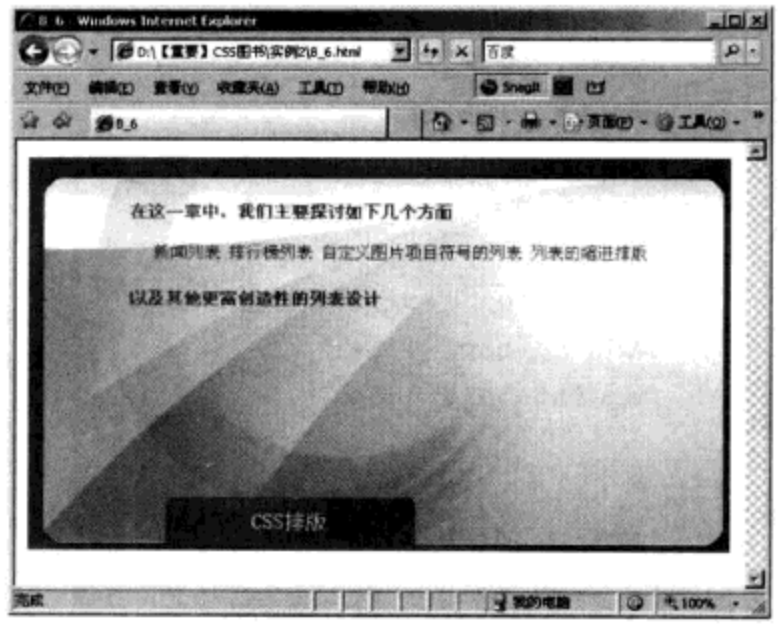


图 8-30　段落排版效果

预览效果发现，li 元素已经呈现为 inline 内联显示方式，排列在一行之中，为了使上下段落一致，也可以继续尝试改变列表元素上下两个元素的显示方式，代码如下。

```
#layout{
    background: url(images/8_4.jpg) no-repeat;          /*设置背景图片，背景不平铺*/
    width: 587px;                                       /*设置宽度*/
    height: 339px;                                      /*设置高度*/
    color: #0099CC;                                     /*设置文字颜色*/
    padding-top: 50px;                                  /*设置上内边距*/
    line-height: 24px;                                  /*设置行高*/
}
#layout h3{
    display: inline;                                    /*设置内联元素显示方式*/
    font-size: 14px;                                    /*设置字号*/
    margin-left: 40px;                                  /*设置左外边距*/
}
#list{
    font-size: 14px;                                    /*设置字号*/
    display: inline;                                    /*设置内联元素显示方式*/
    font-weight: bold;                                  /*设置文字为粗体*/
    margin-left: 10px;                                  /*设置左外边距*/
}
#layout h3.rePos{
    margin-left: 10px;                                  /*设置左外边距*/
}
```

预览效果如图 8-31 所示。

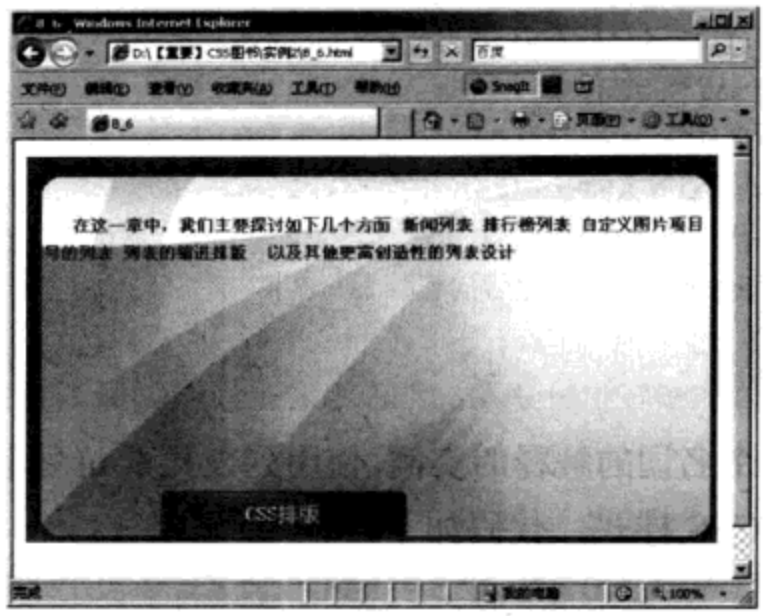


图 8-31　段落排版的效果

预览效果已经达到目的，列表元素成为 inline 内联模式显示为一段文本，与上下文关联起来。

通过对#list 元素和#layout h3.rePos 元素的左边距进一步调整，可以将整个列表块进一步融合进整段文本中。

inline 内联模式使列表变成段落成为可能，CSS 中样式属性的强大功能再一次帮助我们完成了显示模式上的转变。

8.5　列表缩进排版

列表样式不仅可以使用列表项目符号来制作，还可以通过缩进排版来实现。本节将介绍使用文本缩进属性控制段落的排版。

8.5.1　制作实例

制作列表缩进排版需要使用文本缩进属性 text-indent 来进行样式编写，下面分别讲解文本缩进属性的使用方法和具体实例应用。

1．文本缩进属性语法结构

列表项的段落缩进使用的是 CSS 针对文本控制的一个公用属性，语法结构如下。

```
text-indent: value;
```

text-indent 用于控制段落文本的首行缩进效果，允许使用负值。如果使用负值，那么首行会被缩进到左边。缩进以一种形象化的图示表示，如图 8-32 所示，箭头指向的灰色部分为缩进部分。

2．制作列表缩进排版的实例

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    <ul>
```

```
            <li><strong>布局概述</strong>：<br />
                讲述有关CSS中布局的问题</li>
            <li><strong>页面元素入门</strong>：<br />
                导航、列表、背景等页面中常使用到的元素</li>
            <li><strong>高级技巧</strong>：<br />
                CSS hack以及CSS缩写写法等问题</li>
        </ul>
    </div>
```

以上代码制作的是对 3 个名词的解释的实例，使用列表元素 ul 排版，结构效果如图 8-33 所示。

（2）制作整体框架的 CSS 样式，代码如下。

```
#layout {
    padding: 30px;                          /*设置内边距*/
    background-image: url(images/8_4.jpg);  /*设置背景图片*/
    width: 587px;                           /*设置宽度*/
    height: 339px;                          /*设置高度*/
    background-repeat: no-repeat;           /*设置背景图片不平铺*/
}
```

以上代码定义实例框架的背景图片、宽度和高度，使背景图片完全显示，不平铺；定义了内边距，使文本内容显示在布局中心，预览效果如图 8-34 所示。

图 8-32　text-indent 属性的图示

图 8-33　未添加 CSS 样式的 XHTML 结构效果

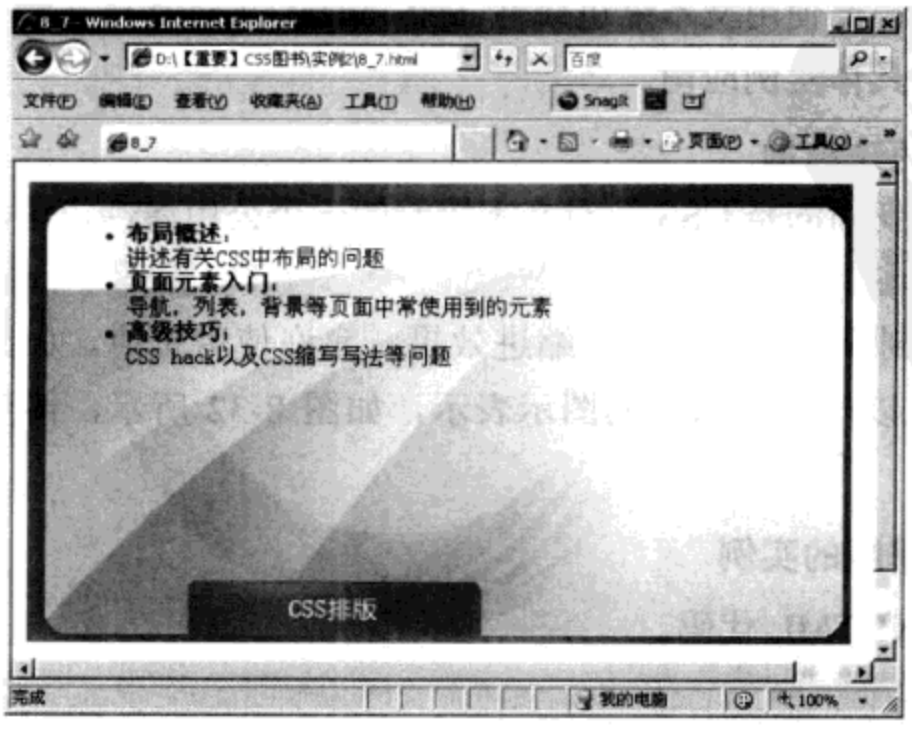


图 8-34　添加背景图片的效果

（3）制作 ul 元素的 CSS 样式，代码如下。

```
ul {
    width: 200px;                                    /*设置宽度*/
    list-style-type: none;                           /*设置列表符号样式*/
    padding-left: 30px;                              /*设置左内边距*/
    color: #0099CC;                                  /*设置文字颜色*/
    font-size: 14px;                                 /*设置字号*/
}
```

以上代码定义了列表的宽度，去除了 ul 列表元素的列表项目样式，定义了左边距，使得页面更美观，并且也设置了字体样式，预览效果如图 8-35 所示。

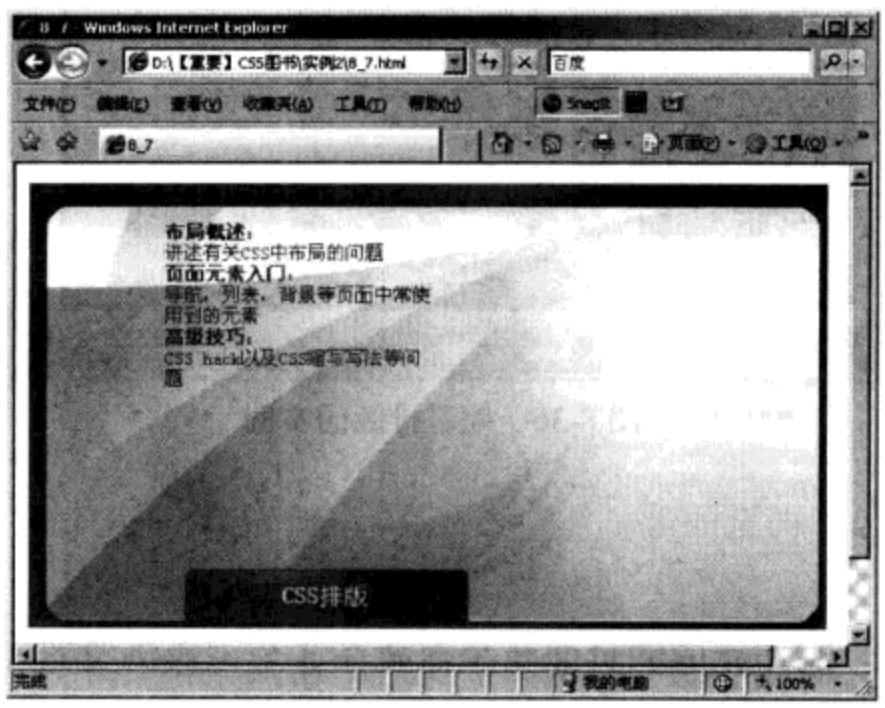


图 8-35　设置字体样式的效果

（4）制作 ul li 元素的 CSS 样式，代码如下。

```
ul li {
    margin: 5px;                                     /*设置外边距*/
    text-indent: -30px;                              /*设置文本缩进*/
    line-height: 24px;                               /*设置行高*/
}
```

以上代码定义了 ul li 元素的外边距和行高。关键的是，第一行显示要比其他行提前一些，是通过 text-indent 的负数取值来实现。text-indent:-30px 使得有可能将首行不是缩进而是向左推进。值得注意的是，单纯的推进会使得首行文字显示在了 ul 的显示区域之外，有必要将 ul 的左边距设置为相同的正数值，以使得 ul 的左边能产生相同距离的空间，最终左推进的首行文字可以显示在 ul 的范围之中。

注意：

text-indent 也可以使用其他单位（例如 50%或 0.8em 等数学单位）。

这样缩进排版的实例就制作完成，预览效果如图 8-36 所示。

text-indent 的缩进效果不但可以应用在列表之中，作为文本控制的一个公共属性，对其他对

象也同样适用。

对列表元素而言，使用 CSS 的这种与列表本身结构无关的样式控制能够帮助随时改变列表外观，而从视觉设计上来说，使用背景控制、文本控制及段落控制等方式又为列表的阅读提供无限可能。

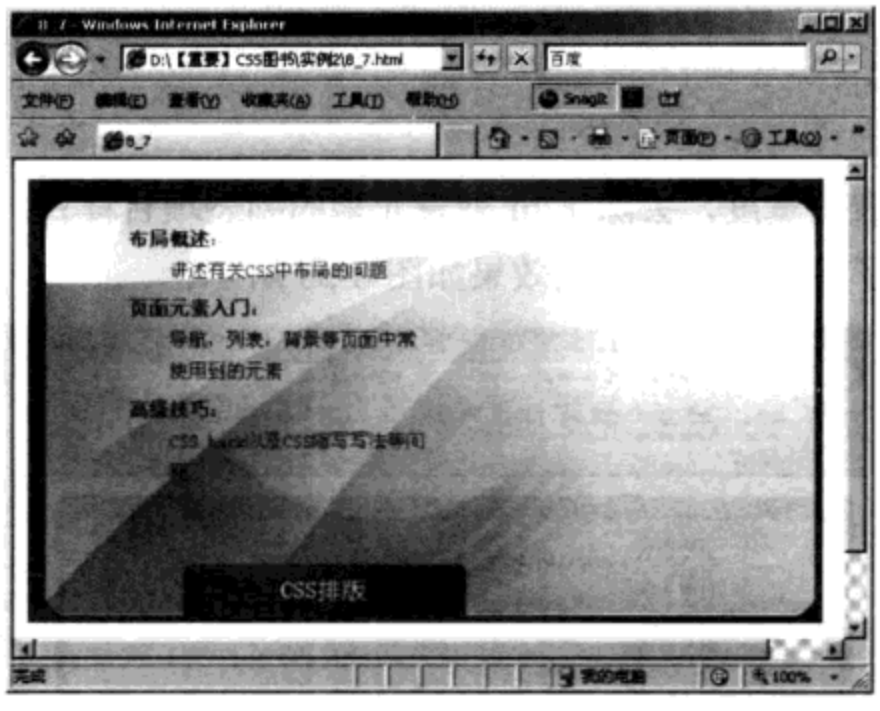


图 8-36　缩进排版的实例

8.5.2　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。

1．Firefox 浏览器测试

把实例所用到的网页文件放在 Firefox 浏览器中打开，预览效果如图 8-37 所示。

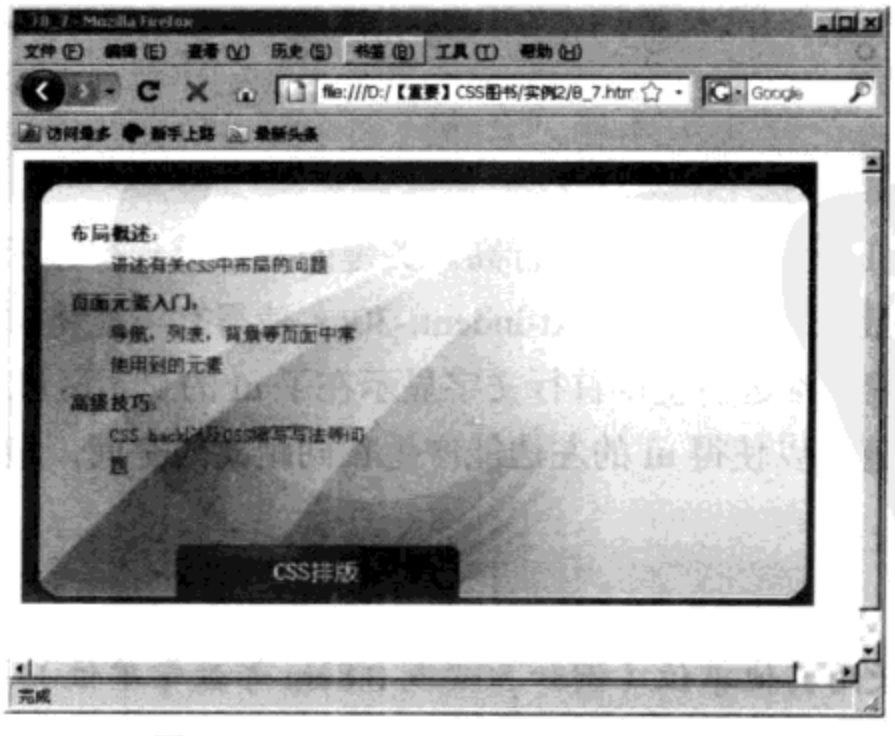


图 8-37　Firefox 浏览器中的预览效果

从图 8-37 可以看出，本实例在 Firefox 浏览器中出现问题了，列表内容整体地向左偏移一段

距离。仔细观察，发现问题的原因在于 ul 在不同浏览器中的表现不同。

ul 列表对象也是容易在 IE 与 Firefox 之中发生问题的对象，主要是由 Firefox 对 ul 对象的默认值设置造成的。ul 在不添加任何属性时，默认情况下是有边距的。在 IE 中，ul 默认边距是由 margin 造成的；而在 Firefox 中，ul 的默认边距是由 padding 造成的。因此解决的方法就是，在设计带有 ul 对象的网页时，使用标签选择符，先统一 ul 的边距，代码如下。

```
ul{
    padding: 0px;
    margin: 0px;
}
```

这样，页面中的所有 ul 对象，都没有了 margin 和 padding 值，当需要针对某一个 ul 进行 margin 或 padding 操作时，再重新进行设置，可以覆盖之前设置的值，就不会再出现表现不同的情况了。

2. IE 6 浏览器测试

通过刚刚介绍的方法，先解决 ul 在不同浏览器表现不同的问题，统一添加一段针对 ul 设置的代码，在 IE 6 浏览器中的预览效果如图 8-38 所示。

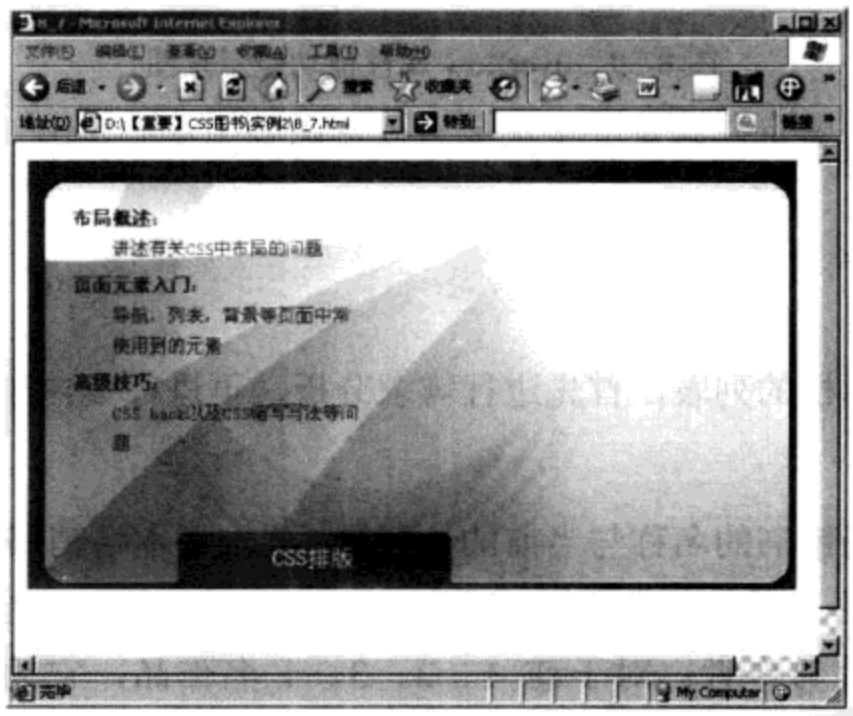


图 8-38　IE 6 浏览器中的预览效果

从图 8-38 可以看出，新闻列表实例在 IE 6 浏览器中没有出现兼容问题。这样，这个实例完成了浏览器兼容的测试，并解决了兼容问题。

8.6　复杂列表排版

列表的设计可简可繁，而网站上的大部分列表应用都以简单为主，主要为了显示一些标题，但是也会出现一些较复杂的应用。本节将介绍非常有特点的列表应用，制作这个实例用到一个 CSS 属性，就是 overflow 属性，首先了解一下这个属性的使用方法。

8.6.1　overflow 属性语法结构

overflow 属性是用来设置当元素的内容溢出其区域时发生的事情，语法结构如下。

```
overflow: visible | hidden | scroll | auto
```

各个属性值的含义如下。

- visible：默认值，内容不会被修剪，会呈现在元素之外。
- hidden：内容会被修剪，但是浏览器不会显示供查看内容的滚动条。
- scroll：内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
- auto：如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

这个属性定义溢出元素内容区时会如何处理。如果值为 scroll，不论是否需要，用户代理都会提供一种滚动机制。因此，即使元素框中可以放下所有内容，有可能也会出现滚动条。以一种形象化的图示表示 overflow 属性值的功能，如图 8-39 所示。

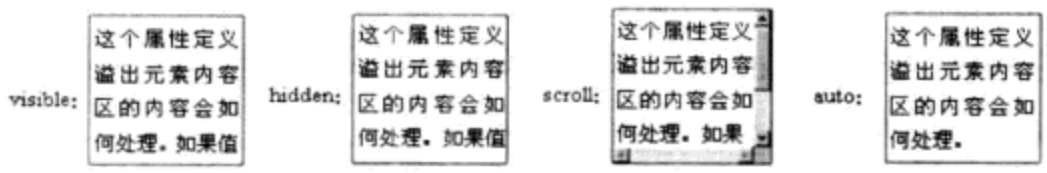


图 8-39　overflow 属性值功能示意图

下面将使用 overflow 属性的特性，应用在复杂排版列表的制作中。

8.6.2　制作实例

设计一个畅销书排行榜的列表，首先进行需求分析，再进行具体制作。

1．需求分析

要放置 1～10 十本畅销书的名称与当前的访问量；要求每本畅销书都必须序号是 1～10，而为了实现较好的视觉效果，要求序号必须采用图片；对于排名第一的畅销书，除了名称与访问量之外，还要求出现图片与作者；对于前 1、2、3、4 名作品，它的背景色也因为其名次而出现从深到浅的丰富变化。

面对这些需求，列表已经变得相当复杂，通过对设计稿的分析可知以下几点。

- 除了第一个内容以外，其他的内容都具有部分相同的特征，除标题外，它们的形式都是标题居左，访问量居右。
- 针对第一个项目，必须有特殊的 CSS 样式。
- 针对 2、3、4 三个项目，需要改变其背景色。
- 针对所有项目要采用不同图片作为标题，因此肯定需要不同的 class 或 id 名称。

2．制作步骤

（1）制作页面的 XHTML 代码。

```
    <div id="layout">
        <div id="top10Title"></div>
        <ul id="top10List">
            <li class="t1">
                <div><a href="#" ><img src="images/8_6_thumb.jpg" width="136" height="113"
alt="闯关东"/></a></div>
                <h1> 《闯关东》</h1>
                <h2>高满堂</h2>
                <h3>3</h3>
            </li>
            <li class="t2"> <img src="images/8_6_t2.gif" width="11" height="8" alt="NO.2"/>
                <h1><a href="#" >沙漏(III终结)</a></h1>
                <h3>2</h3>
            </li>
            <li class="t3"> <img src="images/8_6_t3.gif" width="11" height="8" alt="NO.3"/>
                <h1><a href="#" >我叫刘跃进</a></h1>
                <h3>2</h3>
            </li>
            <li class="t4"> <img src="images/8_6_t4.gif" width="11" height="8" alt="NO.4"/>
                <h1><a href="#" >鬼吹灯 II 之三-怒晴湘西</a></h1>
                <h3>1</h3>
            </li>
            <li class="t5"> <img src="images/8_6_t5.gif" width="11" height="8" alt="NO.5"/>
                <h1><a href="#" >士兵突击</a></h1>
                <h3>1</h3>
            </li>
            其他代码相似，见源代码]
        </ul>
    </div>
```

由以上代码可以看出，在第一个 li 中，使用 div 来放置图片，h1 表示标题，h2 表示作者，h3 表示访问量。而在其他的 li 之中只有 h1 与 h3，并对每一个 li 设置了 t1、t2、t3 这样的 class，不过在实际代码中只需要设置到 t4 即可，因为从第 5 个项目开始将延用相同风格，而 img 元素用于插入带有编号的图片，如图 8-40 所示。


图 8-40　自定义项目图片

（2）制作#layout 元素的 CSS 样式，代码如下。

```
#layout{
    width:278px;                        /*设置宽度*/
    float:left;                         /*设置向左浮动定位*/
    overflow:hidden;                    /*设置内容超过设定区域时，隐藏内容并且不显示滚动条*/
}
```

在以上代码中，用以定义列表框架的#layout 元素指定了列表的宽度，并指定为向左浮动。防止列表的内容超出列表的宽度，所以定义了 overflow 属性为 hidden，超出的内容会被修剪。

（3）制作标题的 CSS 样式，代码如下。

```
#top10Title {
    background:url(images/8_6_top10t.gif) no-repeat;        /*设置背景样式*/
    width:278px;                                            /*设置宽度*/
    height:24px;                                            /*设置高度*/
}
```

在以上代码中，为标题背景指定了一个背景图片，不平铺，背景图片如图 8-41 所示，并为元素指定了宽度和高度，以保证背景图片完整显示。

图 8-41 背景图片

（4）制作#top10List li 元素的 CSS 样式，代码如下。

```
#top10List {
    list-style:none;                                        /*设置列表项目符号样式*/
    margin:0px;                                             /*设置外边距*/
    padding:0px;                                            /*设置内边距*/
}
#top10List li {
    border-top:1px solid #434343;                           /*设置上边框样式*/
    background:url(images/8_6_top10bg.gif) repeat-y;        /*设置背景样式*/
    background-color:#FFFEEE;                               /*设置背景颜色*/
    height:21px ;                                           /*设置高度*/
    overflow:hidden;                                        /*设置剪切超出边界的内容，不显示滚动条*/
    padding-left:9px;                                       /*设置左内边距*/
    margin-bottom:0px;                                      /*设置下外边距*/
}
```

在以上代码中，首先通过#top10List 元素去除了 li 元素的列表样式，并指定了内边距和外边距以保证准确定位。

为#top10List li 元素定义所有列表项的共同样式。设置左内边距、下外边距以及高度以保证准确定位，定义了上边框、背景颜色以及背景图片增强其美观效果，预览效果如图 8-42 所示。

（5）制作具有统一样式的列表项的 CSS 样式，代码如下。

```
#top10List li img {
    float:left;                                             /*设置向左浮动定位*/
    margin-top:6px;                                         /*设置上外边距*/
}
#top10List li h1 a {
    color:#000000;                                          /*设置文字颜色*/
    text-decoration:none;                                   /*设置链接文字文本样式*/
    display:block;                                          /*设置块级显示方式*/
    float:left;                                             /*设置向左浮动样式*/
    width:195px;                                            /*设置宽度*/
    margin:5px 0px 0px 10px;                                /*设置外边距*/
```

```
    font-size: 12px;                                    /*设置字号*/
}
#top10List li h1 a:hover {
    color:#000FFF;                                      /*设置链接文字光标经过时的颜色*/
    text-decoration:underline;                          /*设置文本链接光标经过的文本下划线*/
}
#top10List li h3 {
    margin:10px 0px 0px 3px;                            /*设置外边距*/
    padding-left:3px;                                   /*设置左内边距*/
    color:#A23B0E;                                      /*设置文字颜色*/
    border-left:2px solid #A23B0E;                      /*设置左边框样式*/
    width:44px;                                         /*设置宽度*/
    height:12px;                                        /*设置高度*/
    overflow:hidden;                                    /*设置剪切超出边界的内容*/
}
```

在以上代码中，对于 li 中通过#top10List li img 元素定义表示编号的图片，采取向左浮动，并带有上边距，使图片都处于左侧的固定位置。

而显示标题的 h1 对象，对其下面的 a 对象也同样设置为向左浮动，使得标题都能够贴着图片显示，通过 margin 外边距设置，使得其左边距为 10px；最终距左边的编号 10px 间距，而因为对 a 设置了宽度为 195px，因此后面的 h3 对象将在 195px 的位置显示。用以显示访问量的 h3 对象通过左边框设置使其拥有了线条效果，并设置了一些外边距与内边距以保证定位准确，这样就完成了标准元素设置，预览效果如图 8-43 所示。

图 8-42　列表项背景样式效果

图 8-43　添加列表项标题样式的效果

（6）制作针对第一项特殊的 CSS 样式，代码如下。

```
#top10List li.t1 {
    background:url(images/8_6_top10t1.gif) repeat-y;    /*指定一张背景图片纵向平铺*/
    background-color:#FF9000;                           /*设置背景颜色*/
    height:57px;                                        /*设置高度*/
    overflow:hidden;                                    /*设置剪切超出边界的内容*/
}
```

```
#top10List li.t1 div {
    border:1px solid #000000;                    /*设置边框样式*/
    width:70px;                                  /*设置宽度*/
    height:46px;                                 /*设置高度*/
    overflow:hidden;                             /*设置剪切超出边界的内容*/
    float:left;                                  /*设置向左浮动定位*/
    position:relative;                           /*设置相对定位*/
    top:4px;                                     /*相对定位，设置顶部距离*/
    left:24px;                                   /*相对定位，设置左边距离*/
}
#top10List li.t1 div img {
    margin:-20px 0px 0px -35px;                  /*定义第一项预览图片的外边距*/
}
#top10List li.t1 h1 {
    margin-top:12px;                             /*设置上外边距*/
    color:#A23B0E;                               /*设置文字颜色*/
    border-bottom:1px solid #A23B0E;             /*设置下边框样式*/
    width:173px;                                 /*设置宽度*/
    float:right;                                 /*设置向右浮动定位*/
    font-size: 12px;                             /*设置字号*/
}
#top10List li.t1 h2 {
    color:#B54C0B;                               /*设置文字颜色*/
    width:160px;                                 /*设置宽度*/
    float:right;                                 /*设置向右浮动定位*/
    font-size: 12px;                             /*设置字号*/
    font-weight: normal;                         /*设置字体为正常粗细*/
    line-height: 24px;                           /*设置行高*/
}
#top10List li.t1 h3 {
    width:50px;                                  /*设置宽度*/
    position:relative;                           /*设置相对定位*/
    left:213px;                                  /*相对定位，设置左边距离*/
    top:-25px;                                   /*相对定位，设置顶部距离*/
}
```

在以上代码中，由于第一个项目结构较为复杂，单独使用浮动方式布局已经不能很好地表现效果，因此对 t1 中的各个对象的控制采用了部分进行相对定位的方式，使各对象能够相对于 t1 的整个 li 进行对齐，从而实现最终的布局效果。

在#top10List li.t1 div 元素中，使用浮动方式、相对定位使得图片相对于 t1 进行左对齐。为图片指定了边框样式，使得图片显示美观。在#top10List li.t1 h1 元素中，使用浮动方式、相对定位把表示畅销书名的 h1 元素相对于 t1 进行右对齐。在#top10List li.t1 h2 元素中，也是使用浮动方式、相对定位把表示作者的 h2 元素相对于 t1 进行右对齐。在#top10List li.t1 h3 元素中，用以表示点击数量的 h3 元素被指定为相对定位，相对于#top10List li.t1 h2 元素左

边 213px、上边-25px，使用负值使其距上边外部 25px。完善排行榜第一位的样式效果如图 8-44 所示。

（7）制作第 2～4 个项目的 CSS 样式，代码如下。

```
#top10List li.t2 {
    background-image:url(images/8_6_top10t2.gif);          /*设置背景图片，作为项目符号*/
    background-color:#FFDD36;                              /*设置背景颜色*/
}
#top10List li.t3 {
    background-image:url(images/8_6_top10t3.gif);          /*设置背景图片，作为项目符号*/
    background-color:#FFF76C;                              /*设置背景颜色*/
}
#top10List li.t4 {
    background-image:url(images/8_6_top10t4.gif);          /*设置背景图片，作为项目符号*/
    background-color:#FFFAA5;                              /*设置背景颜色*/
}
```

在以上代码中，为了使第 2～4 个项目拥有不同背景色，使用了 3 个样式表分别设置了 3 个对象不同的背景图片和颜色。

这样复杂列表的排版就制作完成，预览效果如图 8-45 所示。

图 8-44　完善排行榜第一位的样式效果

图 8-45　复杂列表的预览效果

本例的实现实际上是 ul 与排版的结合，ul 本身的设计是相当简单方便的，而有时候也必须在 ul 的各个项目之中应用一些图文排版的技术，就如本例中的列表一样才能实现需要的效果。

8.6.3　兼容问题

这个复杂列表的实例，使用了过多的 CSS 样式和较多的浮动等定位样式，所以出现不同浏览器兼容问题的可能性就会增大。现在在 IE 6 和 Firefox 浏览器中进行调试。

1．Firefox 浏览器测试

把实例中所用到的网页文件放在 Firefox 浏览器中打开，预览效果如图 8-46 所示。

从图 8-46 可以看出，本实例在 Firefox 浏览器中出现了问题，列表内容消失了，并且有些信息错位。仔细观察出现问题的页面，发现主要问题出现在<h1>标签内的元素从 li 标签中消失。由于 h1 这样的标题标签，也有着与 ul 标签一样的问题，在不同浏览器中具有不同的表现，所以，判断这个实例兼容问题出在 h1 标签。

h1 对象也是容易在 IE 与 Firefox 中发生问题的对象，主要是由 Firefox 对 h1 对象的默认值设置造成的。因此解决的方法是在设计带有 h1 对象的网页时，使用标签选择符，先统一 ul 的边距，具体代码如下。

```
#top10List li h1{
    margin: 0px;
    padding: 0px;
}
```

在以上代码中， CSS 样式表中添加了这样的代码，解决 h1 标签的表现不同问题，如图 8-47 所示。

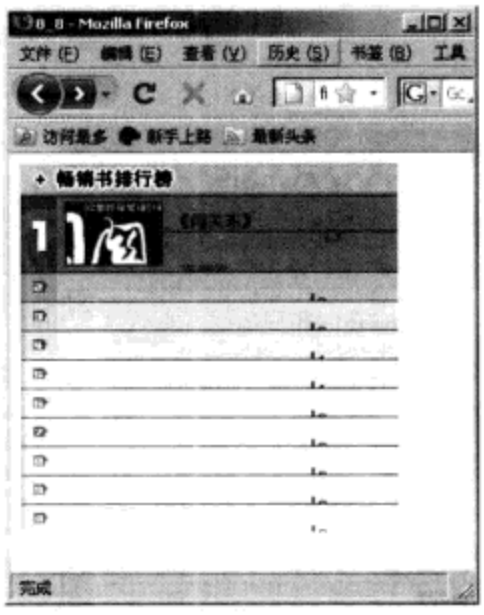


图 8-46　Firefox 浏览器中的预览效果

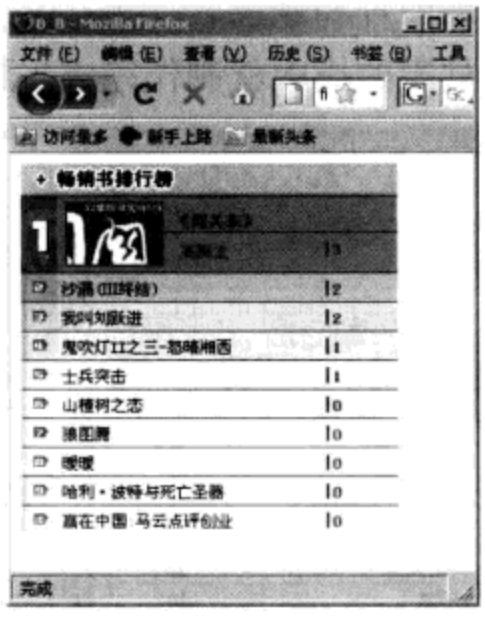


图 8-47　解决了兼容问题的 Firefox 浏览器中的预览效果

2．IE 6 浏览器测试

在 IE 6 浏览器中的预览效果如图 8-48 所示。

从图 8-48 可以看出，本实例在 IE 6 浏览器中出现问题了，列表右侧的数字点击率的位置发生错位。现在使用解决兼容问题的特殊代码，针对 IE 6 浏览器进行特殊设置，代码如下。

```
#top10List li h3 {
    margin:10px 0px 0px 3px;
    padding-left:3px;
    color:#A23B0E;
    border-left:2px solid #A23B0E;
    width:44px;
    height:12px;
    overflow:hidden;
```

```
        font-size: 12px;
        *margin: 5px 0px 0px 3px;
    }
    #top10List li.t1 h3 {
        width:50px;
        position:relative;
        left:213px;
        top:-25px;
        *top: -20px;
    }
```

在以上代码中，针对包含选择符#top10List li h3 和#top10List li.t1 h3 添加了两行特殊代码如下。

```
*margin: 5px 0px 0px 3px;
*top: -20px;
```

第 2 章介绍过*的使用方法，它主要用来解决 IE 6 浏览器的兼容问题。这样，调整了 IE 6 的 CSS 样式后，预览效果如图 8-49 所示。

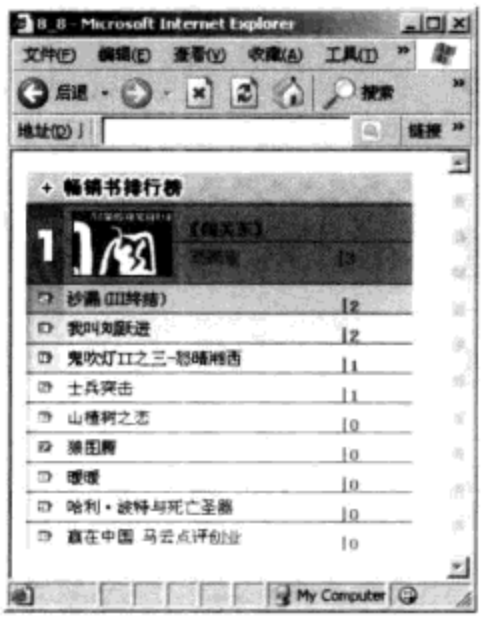


图 8-48　IE 6 浏览器中的预览效果

图 8-49　解决了兼容问题的 IE 6 浏览器中的预览效果

8.7　小结

本章讲解使用 CSS 制作列表的实例，包括了制作新闻列表、排行榜列表、自定义图片的项目符号列表、使用 CSS 进行列表排版、制作复杂列表等内容，主要涉及的 CSS 属性有以下 3 个。

- 列表样式属性 list-style-image。
- 文本缩进属性 text-indent。
- 控制溢出属性 overflow。

对本章的知识点前后联系进行归纳总结，结构导图如图 8-50 所示。

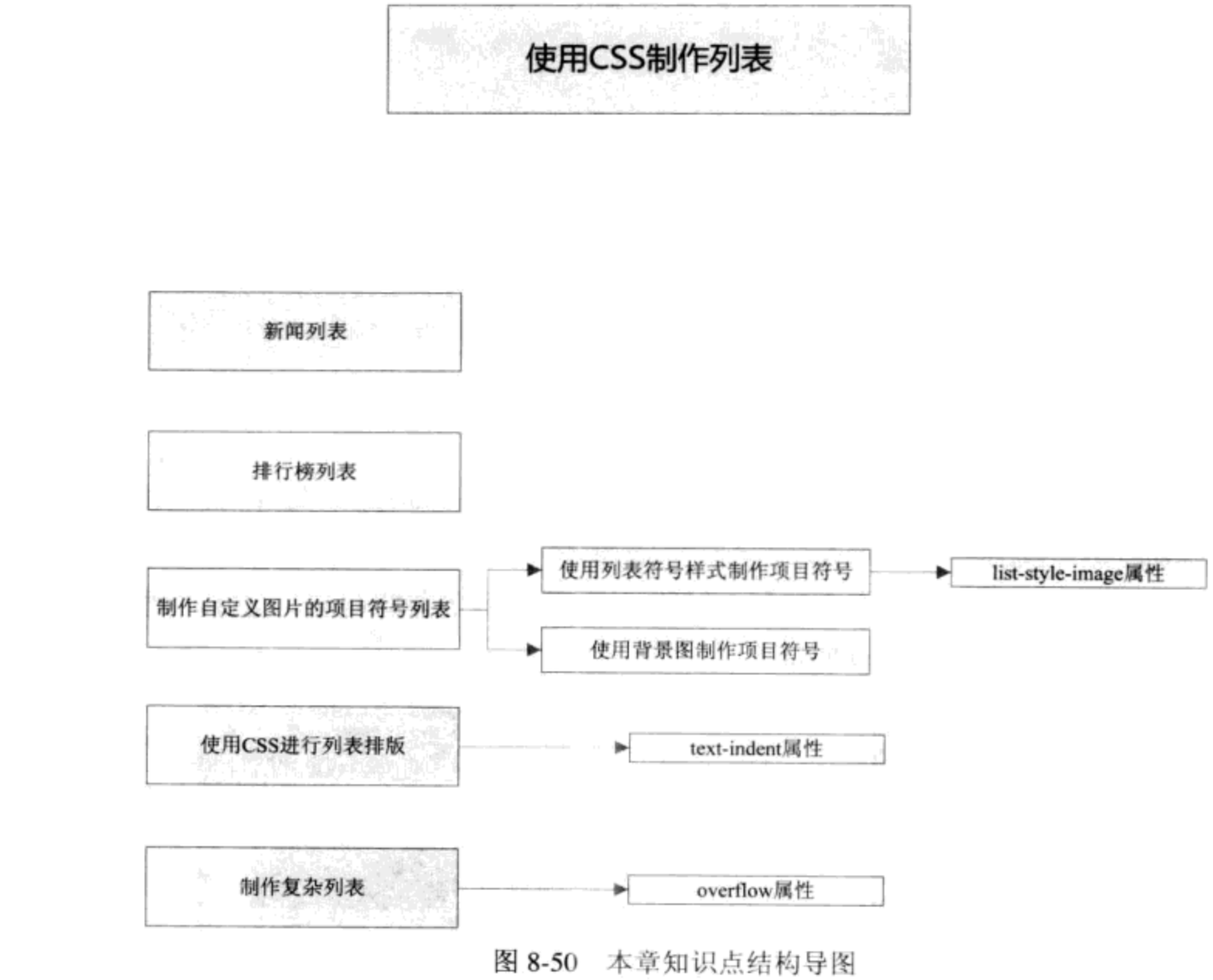


图 8-50　本章知识点结构导图

8.8　习题

1. 什么是新闻列表？
2. 简述制作自定义图片项目符号列表的方法。
3. 怎样利用 CSS 属性直接把列表变为段落？
4. 如何制作列表缩进排版？
5. 简述 overflow 属性的含义及作用。

第9章 使用 CSS 制作表单

表单是功能型网站中经常使用的元素，是网站交互中最重要的元素。在网页中，小到搜索按钮，大到用户注册表单以及用户控制面板，都需要使用表单及其表单元素。

表单元素用来收集用户信息，帮助用户进行功能性控制，表单的交互设计与视觉设计都是网站设计中的重要部分。从表单视觉设计来说，经常需要摆脱 XHTML 提供的默认的比较粗糙的视觉样式。本章将介绍如何使用 CSS 设计表单。

9.1 制作登录表单

登录表单是各大网站都会用到的一个非常实用的模块，是用来进入其他相关页面的入口。如图 9-1 所示是一个带有登录模块的网页。

图 9-1 带有登录模块的网页

本节将使用 form 表单元素以及 label 标签制作一个样式精美的登录表单。

9.1.1　label 标签语法结构

本节将使用 XHTML 语言中表单常用到的 label 标签。在使用表单时，对于小小的单选按钮和复选框，总是需要用光标精确地点击到小框或是小圆上时，才能够完成交互，长期使用时也许会觉得这是一件非常麻烦的事情。表单可用性问题便显露出来，一个不方便用户简单实用的表单是不可取的。XHTML 中提供了一个改善表单交互问题的标签 label，其语法结构代码如下。

```
<label for="#" accesskey="#"></label>
```

label 标签具有提示表单的含义(标签说明)的功能，是成对出现的，以<label>开始，以</label>结束，两个 label 之间的内容是要显示的文字。label 中有两个属性是非常有用的，一个是 for，另外一个就是 accesskey，可用属性及其属性值如下。

- for：表示 label 标签要绑定的 XHTML 元素，当点击这个标签时，所绑定的元素将获取焦点。

```
<label for="InputBox">姓名</label><input id="InputBox" type="text">
```

- accesskey：表示访问 label 标签所绑定元素的热键，当按下热键，所绑定的元素将获取焦点。

```
<label for="InputBox" accesskey="N">姓名</label><input id="InputBox" type="text">
```

注意：

label 标签的 for 属性用于指定该标签所关联的表单元素，当 for 中所指的名称与表单某元素的 id 值相同时，该标签将与该元素关联，点击该标签的同时，该元素也将得到响应。

label 标签是提升表单可用性的简单易行的方法，在能够用到的时候使用这个标签，将会使表单的操作更舒畅。

9.1.2　制作登录表单实例

现在开始使用表单元素和 label 标签设计一个登录表单。分成两个 div 层级关系，外层为 layout 层，用来定义表单框架样式，内层为 member 层，用来放置表单等主体元素。内层 member 又分成标题区域和表单区域，分别由.title 和 form 控制样式。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    <div id="member">
        <div class="title"><img src="images/9_1_02.jpg" /></div>
        <form name="form1" id="form1" method="post" action="#">
            <div>
                <label for="name">Name:</label>
                <input name="name" id="name" type="text" size="10" maxlength="10" />
            </div>
```

```
            <div>
                <label for="pass">Pass:</label>
                <input id="pass" name="pass" type="password" size="10" maxlength="10" />
                <input type="image" src="images/9_1_03.png" class="bt" />
            </div>
        </form>
    </div>
</div>
```

以上代码是制作一个会员登录窗口的表单。登录窗口是一个需要由来访者输入会员账号和密码才能够进入参与相关活动的入口。在整个表单设计中，包含了输入框（input）、图片提交按钮两个表单元素和 label 标签，并且使用 CSS 改进了浏览器默认的外观。

div.title 元素指定了表单标题的图片，标题图片如图 9-2 所示。label 标签用来显示“Name”和“Pass”的文字，使用 for 属性分别指定捆绑的 XHTML 元素是 id 为 name 的 input 元素和 id 为 pass 的 input 元素。input 输入框表单元素，定义两个输入文本框和一个图片提交按钮。输入文本框一个是类型为 text 的单行文本输入框，一个是类型为 password 的密码输入框，文本框宽度都定义为 10 个，最大输入字符为 10。图片提交按钮也是通过 input 元素将类型定义为 image 来实现的，如图 9-3 所示。

图 9-2　标题图片

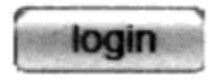


图 9-3　图片提交按钮

页面结构预览效果如图 9-4 所示。

（2）制作#layout 元素的 CSS 样式，代码如下。

```
#layout {
    background-color: #999999;                    /*设置背景颜色*/
    width: 163px;                                 /*设置宽度*/
    height: 142px;                                /*设置高度*/
    font-size: 10px;                              /*设置字号*/
    font-family: Arial;                           /*设置字体类型*/
}
```

在以上代码中，定义表单框架背景颜色为灰色，并为其指定宽度和高度，预览效果如图 9-5 所示。

（3）制作#member 元素的 CSS 样式，代码如下。

```
#member {
    text-align: center;                           /*设置文本居中显示*/
    width: 153px;                                 /*设置宽度*/
    height: 132px;                                /*设置高度*/
    background: url(images/9_1_01.gif) repeat-y;  /*设置背景样式*/
    margin: 5px;                                  /*设置外边距*/
}
.title {
    margin-left: 3px;                             /*设置左外边距*/
```

```
    margin-top: 7px;                                    /*设置上外边距*/
}
```

以上代码定义了#member 元素的背景图片，并且纵向平铺，背景图如图 9-6 所示；指定了#member 元素的宽度和高度以及外边距，用来放置表单各个元素。另外在.title 元素中，指定了标题图片的左外边距和上外边距，保证图片的准确定位，预览效果如图 9-7 所示。

图 9-4　页面结构预览效果

图 9-5　添加了#layout 样式的预览效果

图 9-6　背景图片

图 9-7　添加了#member 和.title 样式的预览效果

从图 9-7 可以看出，输入文本框和 login 按钮都不对齐，继续添加样式。

（4）制作表单元素的 CSS 样式，代码如下。

```
form {
    margin: 0px;                                        /*设置外边距*/
    padding: 4px;                                       /*设置内边距*/
}
label {
    width: 40px;                                        /*设置宽度*/
    float: left;                                        /*设置向左浮动定位*/
    text-align: right;                                  /*设置文本向右对齐*/
    padding: 3px 0px 1px;                               /*设置内边距*/
    color: #FFF;                                        /*设置文字颜色*/
}
input {
    border: 1px solid #000;                             /*设置边框样式*/
    margin: 1px;                                        /*设置外边距*/
    padding: 1px;                                       /*设置内边距*/
    font-family: Arial;                                 /*设置字体类型*/
```

```
    font-size: 12px;                                    /*设置字号*/
    color: #666666;                                     /*设置文字颜色*/
}
```

以上代码是这个实例的关键。制作这个登录表单的实例，主要应用到的是 form 元素、label 标签和 input 元素，在这里分别对它们定义 CSS 样式。使用标签指定式选择符 form 定义了表单外边距和内边距；使用标签指定式选择符 label 定义了文本区域的宽度、内边距、文字颜色以及浮动方式；使用标签指定式选择符 input 定义了输入框和图片按钮的边框、外边距、内边距、字体类型、大小以及颜色属性，预览效果如图 9-8 所示。

从图 9-8 可以看出，login 按钮图片周围还存在黑色的边框，这是 img 标签默认的样式，为了美观，现在需要把边框去掉，继续添加样式。

（5）制作.bt 元素的 CSS 样式，代码如下。

```
.bt {
    width: 67px;                                        /*设置宽度*/
    height: 25px;                                       /*设置高度*/
    border: none;                                       /*设置边框样式*/
    margin-top: 10px;                                   /*设置上外边距*/
}
```

以上代码为图片按钮专门定义了一套 CSS 样式，去除了在标签指定式选择符 label 中定义的边框，又重新指定了上边距，使得按钮距离上面的输入框拉开一定距离，美观大方。同时为按钮定义了宽度和高度，保证图片完整显示出来。这样登录表单的实例就制作完成，预览效果如图 9-9 所示。

图 9-8　为表单标签添加样式的预览效果

图 9-9　登录表单的预览效果

9.1.3　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。下面首先在 Firefox 浏览器下测试。

1．Firefox 浏览器测试

在 Firefox 浏览器中打开该实例，预览效果如图 9-10 所示。从图 9-10 可以看出，本实例在

Firefox 浏览器中出现问题了，表单上边界被切下一块，问题的关键在于类标签#layout 选择符设置的内边距 padding，重新设置 padding 的值，代码如下。

```
#layout {
    background-color: #999999;                    /*设置背景颜色*/
    width: 163px;                                 /*设置宽度*/
    height: 146px;                                /*设置高度*/
    font-size: 10px;                              /*设置字号*/
    font-family: Arial;                           /*设置字体类型*/
    padding-top: 1px;                             /*设置上内边距*/
}
```

在以上代码中，最后一行添加了上内边距的设置 padding-top: 1px，这样就解决了表单上边界缺少一块灰色背景的问题，预览效果如图 9-11 所示。

2. IE 6 浏览器测试

在 IE 6 浏览器中打开该实例，预览效果如图 9-12 所示。

从图 9-12 可以看出，除了 login 图片边缘出现问题以外，其他没有样式问题。而图片边缘的问题，正是 IE 6 浏览器对 png 格式图片的不支持造成的，所以为了避免出现这样的问题，可以改为使用 jpg 等格式的图片。

图 9-10　Firefox 浏览器中的预览效果

图 9-11　解决兼容问题后 Firefox 浏览器中的预览效果

图 9-12　IE 6 浏览器中的预览效果

9.2　制作用户注册表单

表单布局除了需要应用表单中的各个元素之外，还需要使用 table 表格，表格是表单布局的得力助手。对于最终的表单来说，表格对数据的排列方式非常适合于表单元素的排列。对于常见的左右式表单，即左侧为项目名称，右侧为项目输入框的形式，也非常符合表格具有表头及单元格的属性，因此使用表格进行表单排版再合适不过了。如图 9-13 所示为一个带有表单的网页。

本节尝试使用 XHTML 的表格将各表单元素放置于表格中。

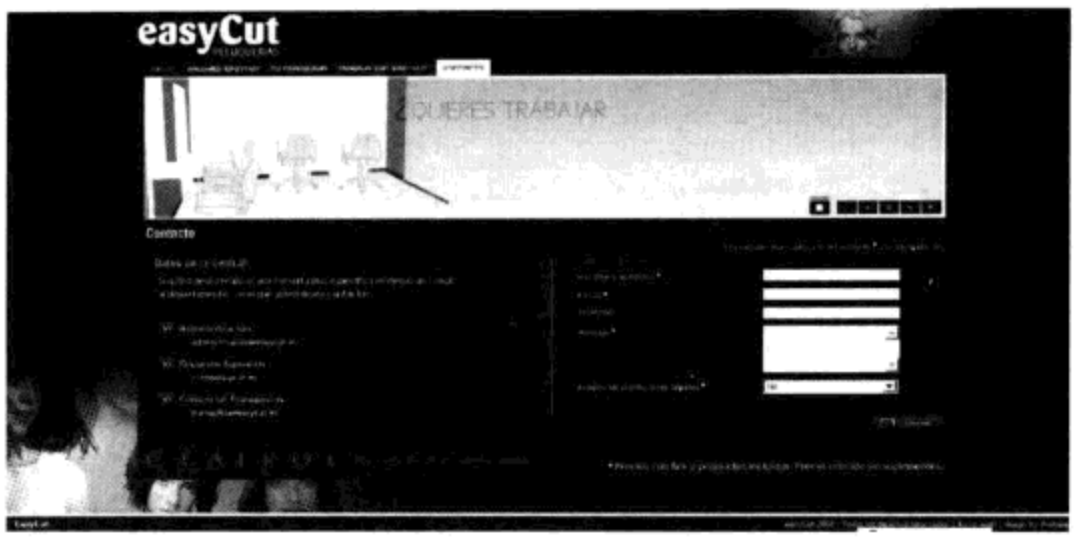


图 9-13　一个带有表单的网页

9.2.1　制作页面的 XHTML 代码

制作页面的 XHTML 代码。

```
    <div id="pageHolder">
        <div class="action">
            <form name="registerActionForm" method="POST" action="#">
                <h2>注册</h2>
                <p>欢迎注册，注册过程将不会收取您任何费用。我们承诺保护您的信息安全，不会将它提供给任何第三
方。</p>
                <table>
                    <tr>
                        <td><label for="username">用户名:</label></td>
                        <td><input  type="text"  name="userId"  maxlength="99"  value=""
style="width:150px" class="textInput" id="username" />
                        <br/>
                        <span class="note">Use alphanumeric characters (<strong>-</strong>,
<strong>_</strong> and <strong>/</strong> are okay too)</span></td>
                    </tr>
                    <tr>
                        <td><label for="password">密码:</label></td>
                        <td><input type="password" name="password" maxlength="99" value=""
class= "textInput" /></td>
                    </tr>
                    <tr>
                        <td><label for="passwordAgain">确认密码:</label></td>
                        <td><input  type="password"  name="passwordConfirm"  maxlength="99"
value="" class= "textInput" id="passwordAgain" /></td>
                    </tr>
                    <tr>
                        <td><label for="email">电子邮件:</label></td>
                        <td><input  type="text"  name="email"  value=""  style="width:250px"
class="textInput" id= "email" /></td>
                    </tr>
                </table>
```

```
                <p>请设置机密问题，当此账号修改密码时会要求您回答机密问题，这将确保是您本人在进行密码修改操
作，使您的账号更安全。</p>
                <table>
                    <tr>
                        <td><label for="secretQ">密码问题:</label></td>
                        <td><input type="text" name="secretQ" value="" style="width:300px"
class="textInput" id="secretQ" />
                        <br />
                        <span class="note">输入一个你可以很容易回答，别人却很难猜到答案的问题。例如“我
的一个宠物的名字叫什么？”</span></td>
                    </tr>
                    <tr>
                        <td><label for="secretA">问题答案:</label></td>
                        <td><input type="text" name="secretA" value="" style="width:300px"
class="textInput" id="secretA" /></td>
                    </tr>
                </table>
                <div class="formAction">
                    <input type="submit" value="提交&raquo;" class="button" />
                </div>
            </form>
        </div>
    </div>
```

在以上代码中，首先为整个界面添加了一个 div#pageHolder 包含页面中的所有元素，以便实现表单的整体外观。然后又添加了一个 div .action 包含表单所有元素，表单具体内容部分放在<form></form>元素中，h2 元素表示的是表单标题，p 元素表示的是说明文字，table 元素里面是表单的各项注册信息，包括用户名、密码、确认密码、电子邮件以及密码提示问题等一些注册所需要填写的信息。表单代码经过表格的嵌套，使得实际表单元素整齐地分入了整个单元格中。

提交按钮使用 input 元素的 submit 属性定义，修改其 value 值为“提交”，页面结构预览效果如图 9-14 所示。

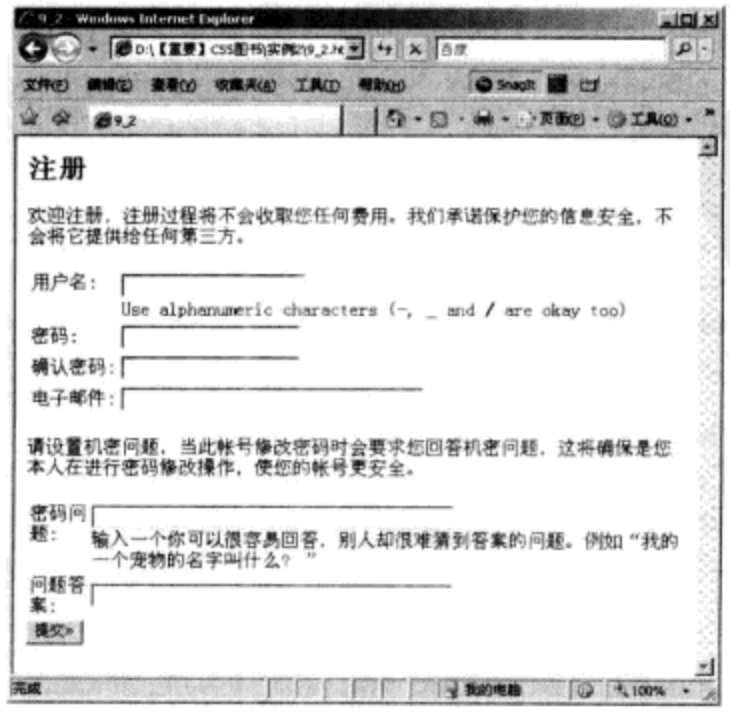


图 9-14　页面结构预览效果

9.2.2　制作页面的 CSS 样式

（1）制作 body 的 CSS 样式，代码如下。

```
body {
    background: #eee url(images/backTileGray.jpg) repeat-x;    /*设置背景样式*/
    text-align: center;                                        /*设置文本居中显示*/
    font: 12px Arial, Helvetica, sans-serif;                   /*设置字号、字体类型*/
    color: black;                                              /*设置字体颜色*/
}
```

在以上代码中，先为页面的 body 标签定义了页面的背景属性，背景图片不平铺，背景如图 9-15 所示。同时定义了整个页面的文本居中对齐，字号、类型以及颜色，页面预览效果如图 9-16 所示。

从图 9-16 可以看出，表单内的元素全都居中显示，排列没有秩序，继续添加样式。

（2）制作#pageHolder 元素的 CSS 样式，代码如下。

```
#pageHolder {
    width: 500px;                  /*设置宽度*/
    margin: 0 auto;                /*设置外边距，实现居中自适应*/
    position: relative;            /*设置相对定位*/
    text-align: left;              /*设置文本向左对齐*/
}
```

图 9-15　body 背景图

在以上代码中，定义了注册表单的框架的宽度，通过指定外边距 margin 使该区域居中页面显示，这是表单排版很关键的一步，预览效果如图 9-17 所示。

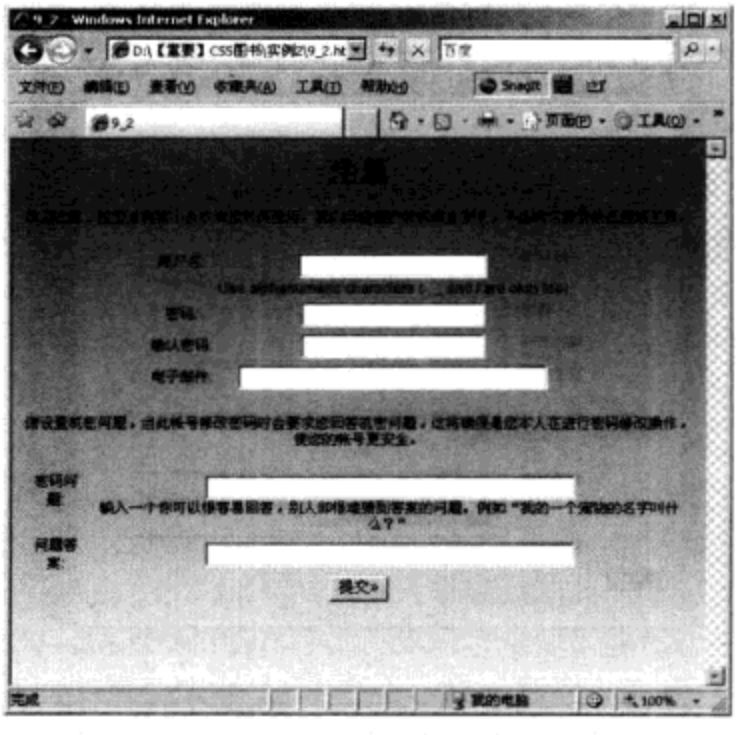


图 9-16　添加 body 样式的页面预览效果

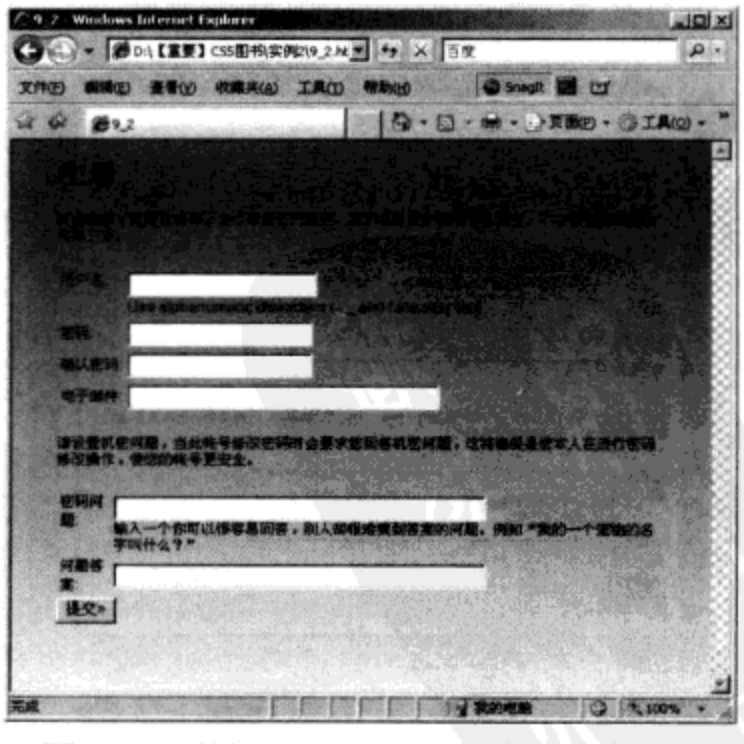


图 9-17　添加 pageHolder 样式的预览效果

（3）制作.action 元素的 CSS 样式，代码如下。

```
.action {
    background: white;                                          /*设置背景颜色*/
    padding: 12px;                                              /*设置内边距*/
    margin-top: 12px;                                           /*设置上外边距*/
    border-bottom: 1px solid #777;                              /*设置下边框样式*/
    border-left: 1px solid #aaa;                                /*设置左边框样式*/
    background: transparent url(images/bg_rounded.gif) no-repeat right top;/*设置背景样式*/
}
```

在以上代码中，定义了.action 元素的背景颜色、内边距、上外边距以及边框样式。背景定义了一个白色的圆角图片，不平铺，定位于.action 元素的右上角，指定圆角图片使页面具有设计感，预览效果如图 9-18 所示。

从图 9-18 可以看出，表单所有内容都被放在一个白色背景的区域中来。

（4）制作 label 标签的 CSS 样式，代码如下。

```
label {
    float: left;                                                /*设置向左浮动定位*/
    width: 120px;                                               /*设置宽度*/
    text-align: right;                                          /*设置文本向右对齐*/
    margin-right: 5px;                                          /*设置右外边距*/
    font-weight: bold;                                          /*设置字体为粗体*/
    font-size: 12px;                                            /*设置字号*/
    margin-top: 1px;                                            /*设置上外边距*/
}
```

在以上代码中，为注册信息的栏目标签定义了宽度、字号及字体粗细，并且指定成浮动显示，指定了右外边距和上外边距，预览效果如图 9-19 所示。

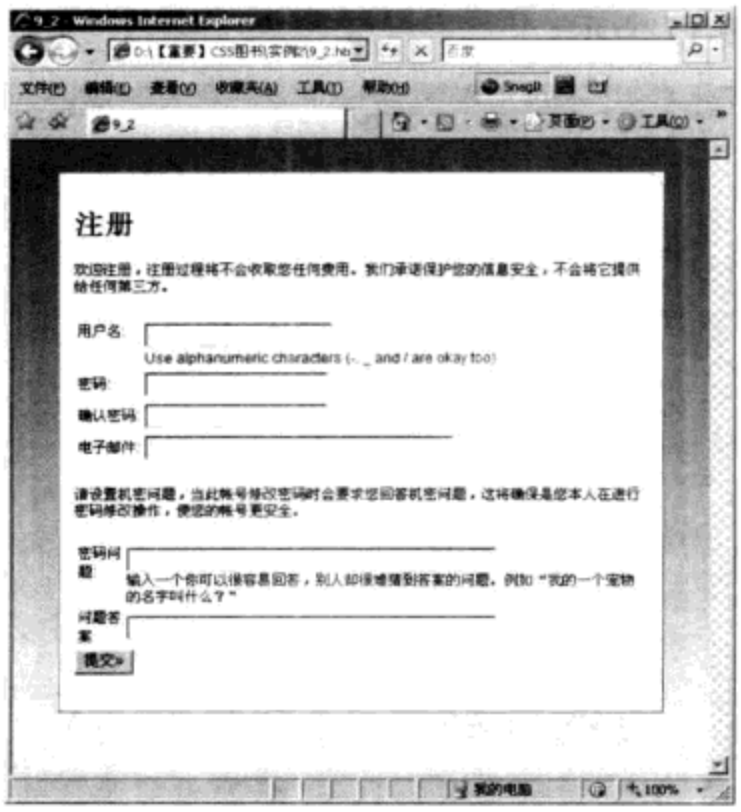


图 9-18　添加.action 样式的预览效果

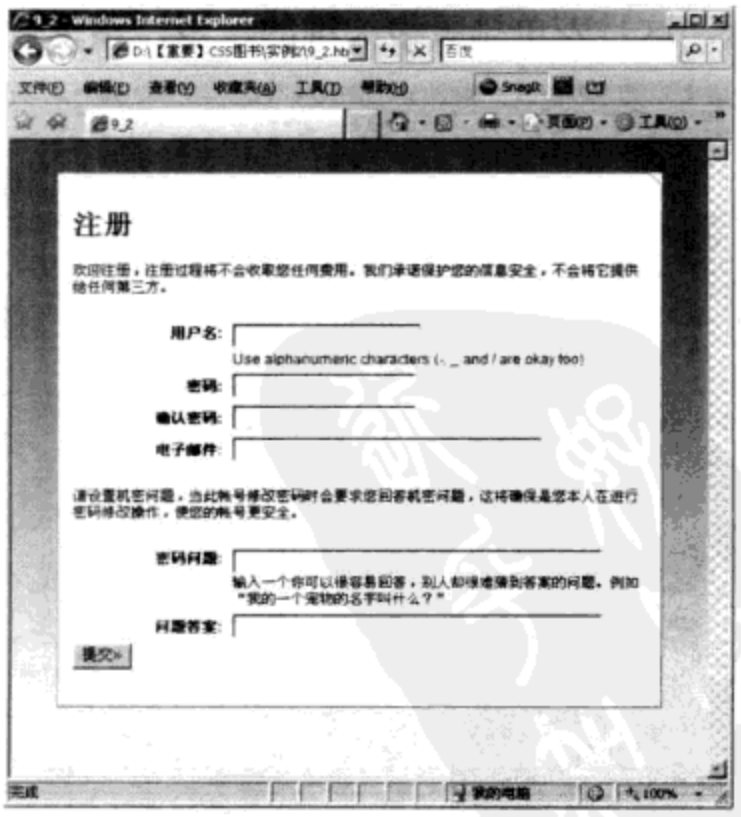


图 9-19　添加 label 样式的预览效果

（5）制作 input .textInput 元素的 CSS 样式，代码如下。

```
input.textInput {
    border: 1px solid #bbb;                                    /*设置边框样式*/
    border-top-color: #666;                                    /*重新设置上边框颜色*/
    border-left-color: #666;                                   /*重新设置左边框颜色*/
    padding: 3px;                                              /*设置内边距*/
    width: 300px;                                              /*设置宽度*/
    background: white url(images/bg_textfield.gif) repeat-x top;   /*设置背景样式*/
}
```

在以上代码中，使用包含选择符 input .textInput 定义单行输入文本框的边框、内边距、宽度及背景样式，定义了不一样的上边框和左边框的颜色。为了使输入框呈现出立体的效果，背景通过 background 属性改变了浏览器默认的样式，重新定义了一张背景图片，横向平铺，并且居顶部显示，背景色为白色，预览效果如图 9-20 所示。

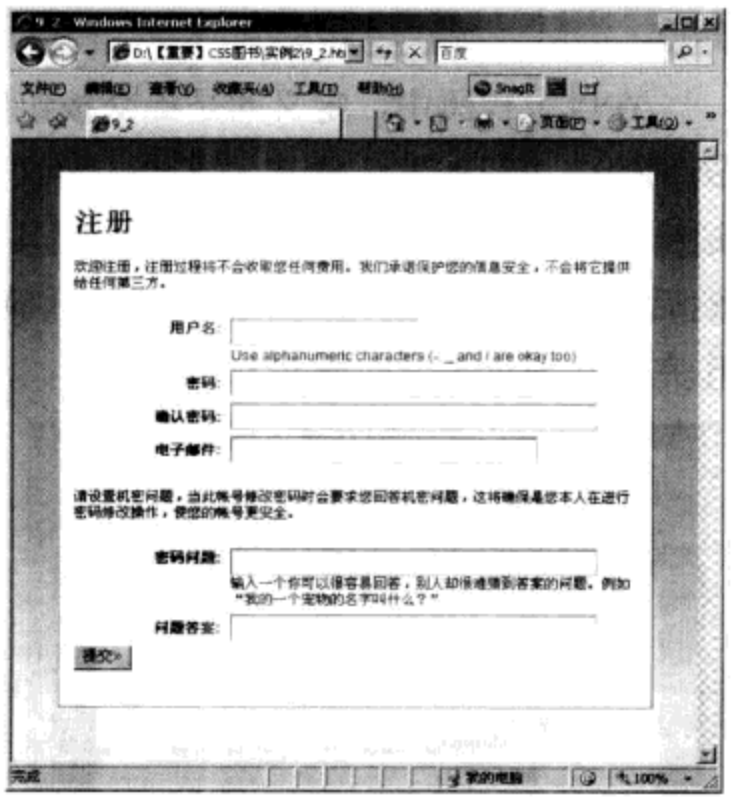


图 9-20　添加输入文本框样式的预览效果

（6）制作提交按钮区域的 CSS 样式，代码如下。

```
div.formAction {
    margin: 20px -12px -12px -12px;                            /*设置外边距*/
    background: url(images/bg_gray.gif);                       /*设置背景样式*/
    text-align: left;                                          /*设置文本向左对齐*/
    padding: 12px 10px;                                        /*设置内边距*/
    width: 478px;                                              /*设置宽度*/
    font-size: 12px;                                           /*设置字号*/
}
div.formAction input {
    font-size: 16px;                                           /*设置字号*/
    margin-left: 12px;                                         /*设置左外边距*/
}
```

在以上代码中，定义的是提交按钮区域的 CSS 样式。首先，在 div.formAciton 元素中，定义了按钮背景区域的图片，并且将这个区域文本定义成向左对齐，指定了该区域的内边距和外边距等属性。然后再通过包含选择符 div.formAction input 定义了图片类型按钮的字号和按钮的左外边距。

（7）制作提交按钮内部的 CSS 样式，代码如下。

```
input.button {
    border: 2px solid #aaa;                                    /*设置边框样式*/
    border-top-color: #ddd;                                    /*重新设置上边框颜色*/
    border-left-color: #ddd;                                   /*重新设置左边框颜色*/
    color: #444;                                               /*设置文字颜色*/
    font-weight: bold;                                         /*设置文字为粗体*/
    background: url(images/bg_button.jpg) left bottom repeat-x;  /*设置背景样式*/
}
```

图 9-21　按钮背景图片

在以上代码中，定义的是提交按钮内部样式，通过 CSS 改变了浏览器中按钮的默认样式。首先，定义了按钮的边框样式，将其左边框和上边框定义成不同的颜色，使其呈现出具有立体感的效果。然后指定了背景图片，如图 9-21 所示横向平铺，并且居左下显示。

这样注册表单实例就制作完成，预览效果如图 9-22 所示。

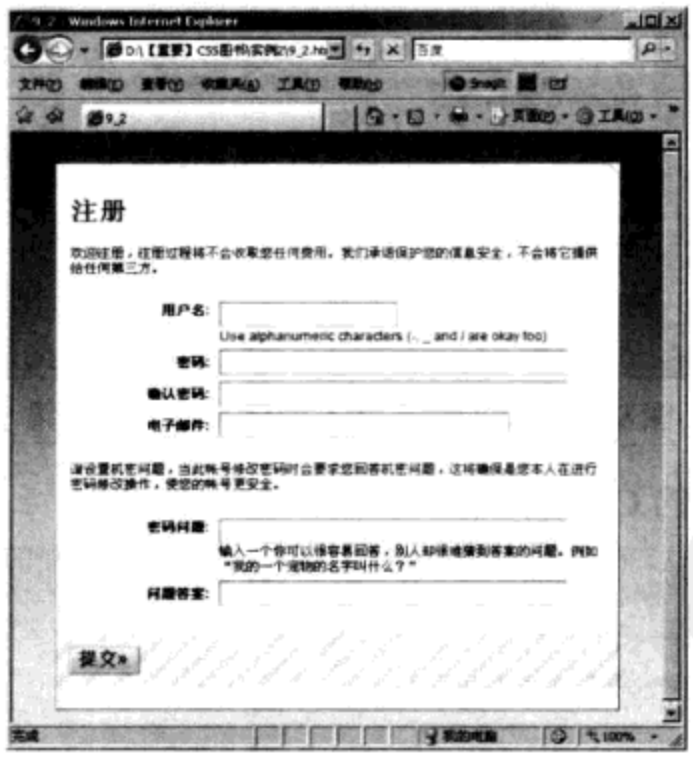


图 9-22　注册表单的预览效果

9.3　制作符合 W3C 标准的表单

所谓符合 W3C 标准的表单，就是指能够在 IE 7、Firefox 和 IE 6 等主流浏览器全兼容、符合用户体验、并且符合 Web 标准的表单。

目前 W3C 标准已经成为高端客户设计网站的首选。国内用户上网用 IE 浏览器的比较多，

但从全世界的上网用户来看，有些客户并不是用 IE 来上网浏览内容的，他们会用一些其他的浏览工具，例如 Netscape、Mozilla、FireFox、Opera 等，如果网站没有采用 W3C 标准，那么使用其他浏览器的用户，就无法看到这样的网站。即此标准是国际上的通用标准，符合此标准的网站可以使用任何浏览器来浏览。

本节将举例讲解使用 CSS 制作符合 W3C 标准的表单的方法。

9.3.1　制作页面的 XHTML 代码

（1）制作页面总体结构的 XHTML 代码。

```
<h3>Standard Form Layout</h3>
<div id="container">
    <p id="fm-intro">Fields in <strong>bold</strong> are required.</p>
    <form id="fm-form" method="post" action="#" >
    </form>
</div>
```

在以上代码中，表单的标题使用 h3 标签的默认样式，然后下面表单的主体包含在使用 id 为 container 的 div 容器里面。表单的主体由 p 标签和 form 标签组成。p 标签内的内容是表单的附加说明文字，用来提示这个表单的粗体部分为必填内容，form 标签放置表单的各个表单项内容，例如“个人信息”、“地址”、“联系方式”等都要放在这对 form 标签内。

（2）制作“个人信息”模块的 XHTML 代码。

```
<fieldset>
    <legend>Personal information</legend>
    <div class="fm-req">
        <label for="fm-firstname">First name:</label>
        <input name="fm-firstname" id="fm-firstname" type="text" />
    </div>
    <div class="fm-opt">
        <label for="fm-middlename">Middle name:</label>
        <input id="fm-middlename" name="fm-middlename" type="text" />
    </div>
    <div class="fm-req">
        <label for="fm-lastname">Last name:</label>
        <input name="fm-lastname" id="fm-lastname" type="text" />
    </div>
</fieldset>
```

在以上代码中，使用 fieldset 标签对表单分组，然后使用 legend 标签对表单每组内容进行说明，这里作为表单组的名称。表单的每一个填充项包含在 div 标签内。这里的填充项包括“First name”、“Middle name”、“Last name”。

（3）制作“地址”模块的 XHTML 代码。

```
<fieldset>
    <legend>Address </legend>
    <div class="fm-opt">
```

```
            <label for="fm-addr">Address:</label>
            <input id="fm-addr" name="fm-addr" type="text" />
        </div>
        <div class="fm-opt">
            <label for="fm-city">City or Town:</label>
            <input id="fm-city" name="fm-city" type="text" />
        </div>
        <div class="fm-opt">
            <label for="fm-state">State:</label>
            <select id="fm-state" name="fm-state">
            <option value="" selected="selected">Choose a State</option>
            <option value="UNK">Outside US / Canada</option>
            <option value="AL">Alabama</option>
            </select>
        </div>
        <div class="fm-req">
            <label for="fm-zipcode">Zip code:</label>
            <input id="fm-zipcode" name="fm-zipcode" type="text" />
        </div>
    </fieldset>
```

在以上代码中，使用 fieldset 标签制作 Address 表单组，里面的填充项包括了“Address”、“City or Town”、“State”和“Zip code”。

（4）制作“联系方式”模块的 XHTML 代码。

```
    <fieldset>
        <legend>Contact information</legend>
        <div class="fm-req">
            <label for="fm-telephone">Telephone:</label>
            <input id="fm-telephone" name="fm-telephone" type="text" title="Enter Phone Number
in xxx-xxx-xxxx format" />
        </div>
        <div class="fm-opt">
            <label for="fm-fax">Fax:</label>
            <input id="fm-fax" name="fm-fax" type="text" title="Enter Fax Number in xxx-xxx-xxxx
format" />
        </div>
        <div class="fm-opt">
            <label for="fm-mobile">Mobile:</label>
            <input id="fm-mobile" name="fm-mobile" type="text" />
        </div>
        <div class="fm-req">
            <label for="fm-email">Email:</label>
            <input id="fm-email" name="fm-email" type="text" tabindex="" />
        </div>
        <div class="fm-opt">
            <label for="fm-url">Web site address:</label>
            <input id="fm-url" name="fm-url" type="text" />
        </div>
        <div class="fm-opt">
            <label for="fm-comments">Comments:</label>
```

```
            <textarea name="fm-comments" cols="10" rows="5" id="fm-comments"></textarea>
        </div>
        <div class="fm-multi">
            <div class="fm-opt">
                <p>Would you like to be notified of future updates?</p>
                <label for="fm-newsopt-yes">
                <input  name="fm-newsopt"  type="radio"  id="fm-newsopt-yes"  value="yes"
checked="checked" />
                Yes</label>
                <label for="fm-newsopt-no">
                <input id="fm-newsopt-no" name="fm-newsopt" type="radio" value="no" />
                No</label>
            </div>
        </div>
    </fieldset>
```

在以上代码中，使用 fieldset 标签制作 Contact information 表单组，里面的填充项包括了“Telephone”、“Fax”、“Mobile”、“Email”、“Web site address”、“Comments”和一组单选按钮。

（5）制作提交按钮的 XHTML 代码。

```
    <div id="fm-submit" class="fm-req">
        <input name="Submit" value="Submit" type="submit" />
    </div>
    <br />
    <div style="text-align: center;" class="fm-req"><a href="#">Back to {style:phreak;}</a></div>
```

以上代码制作的是提交按钮。这样，整个表单的 XHTML 结构就编写完成，预览效果如图 9-23 所示。

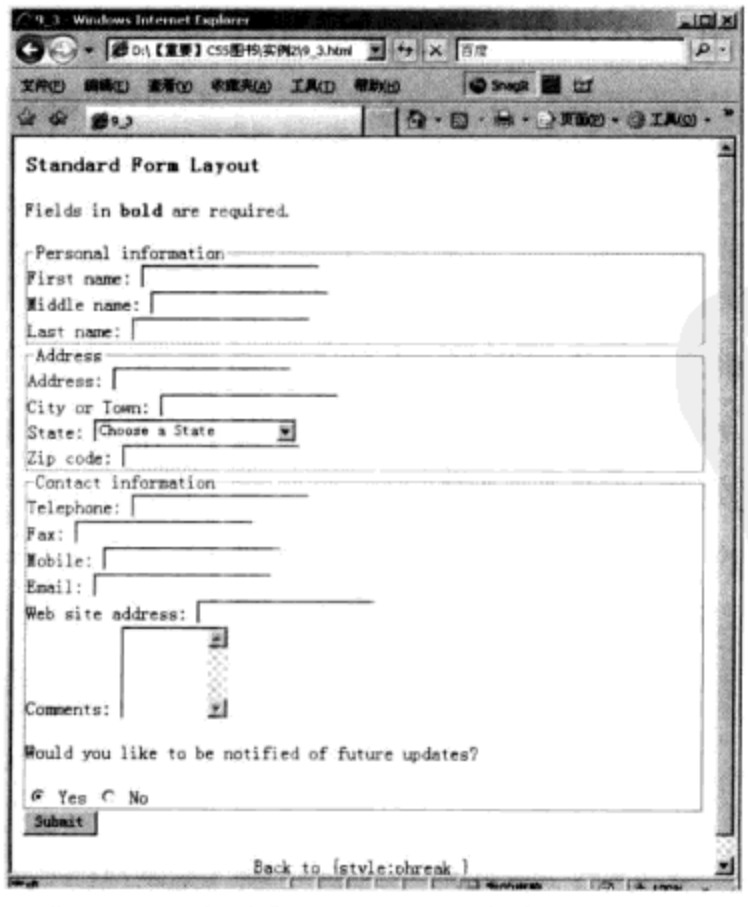


图 9-23　表单的 XHTML 结构的预览效果

9.3.2　制作页面的 CSS 样式

完成了表单的 XHTML 结构，现在开始为结构添加表现样式，即 CSS 样式表。

（1）制作页面总体结构的 CSS 代码，代码如下。

```
body {
    padding: 0;                                          /*设置内边距*/
    margin: 20px;                                        /*设置外边距*/
    color: #333;                                         /*设置文字颜色*/
    background: #fff;                                    /*设置背景颜色*/
    font: 12px arial,verdana,sans-serif;                 /*设置字体样式*/
    text-align: center;                                  /*设置文本居中*/
}
#container {
    margin: 0 auto;                                      /*设置外边距，实现居中对齐*/
    padding: 1em;                                        /*设置内边距*/
    width: 350px;                                        /*设置宽度*/
    text-align: left;                                    /*设置文本左对齐*/
}
p#fm-intro {
    margin: 0;                                           /*设置外边距*/
}
form {
    margin: 0;                                           /*设置外边距*/
    padding: 0;                                          /*设置内边距*/
}
```

在以上代码中，首先为表单的整体结构添加 CSS 样式。使用 body 标签指定式选择符为页面设置了内外边距、背景色、字体类型和文字颜色以及文本的对齐方式。使用#container 设置表单容器的边距、宽度以及表单内文本的对齐方式，预览效果如图 9-24 所示。

（2）制作标签 fieldset 的 CSS 代码，代码如下。

```
fieldset {
    margin: 1em 0;                                       /*设置外边距*/
    border: none;                                        /*设置边框*/
    border-top: 1px solid #ccc;                          /*重新设置上边框样式*/
}
```

在以上代码中，使用 fieldset 标签指定式选择符为表单组设置了外边距、边框样式。首先使用 border 属性将边框取消，改变 fieldset 默认的边框样式，然后再添加 border-top 样式，重新设置表单组的边框样式，预览效果如图 9-25 所示。

（3）制作标签 legend 的 CSS 代码，代码如下。

```
legend {
    margin: 1em 0;                                       /*设置外边距*/
    padding: 0 .5em;                                     /*设置内边距*/
    color: #036;                                         /*设置文字颜色*/
    background:transparent                               /*设置背景色*/
    font-size: 1.3em;                                    /*设置字号*/
    font-weight: bold;                                   /*设置字体为粗体*/
}
```

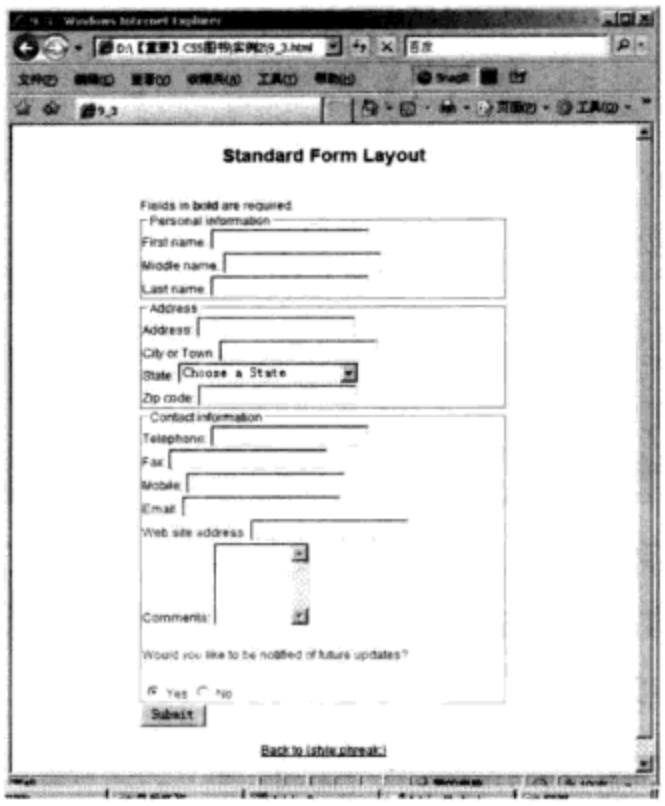


图 9-24　添加 body 等总体结构样式的表单预览效果

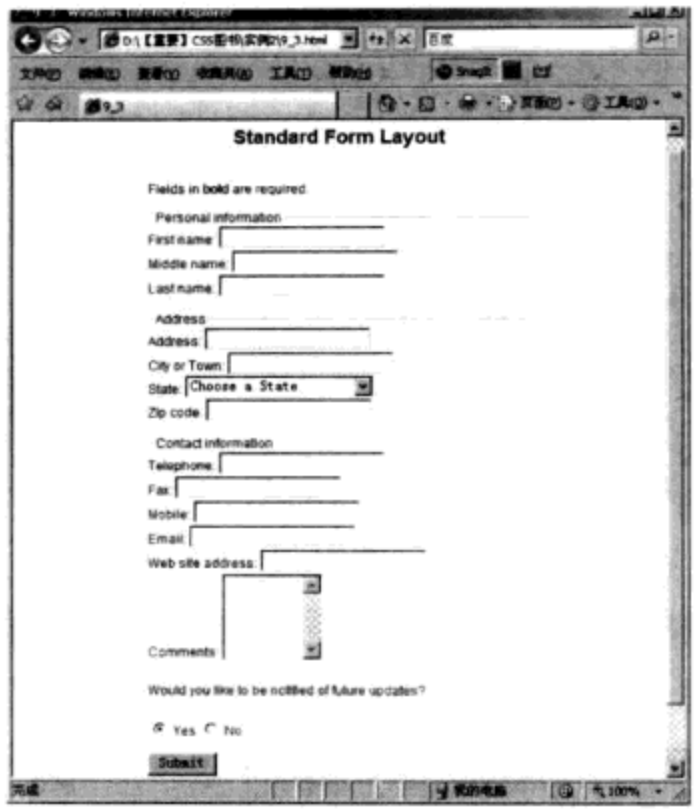


图 9-25　修改 fieldset 边框样式的表单预览效果

在以上代码中，使用 legend 标签指定式选择符为表单组的标题设置了内外边距、文字颜色、字号以及字体粗细，并且设置背景为 transparent 透明。经过修改表单组的标题样式，突出显示标题名称，使每个表单组分隔明显，预览效果如图 9-26 所示。

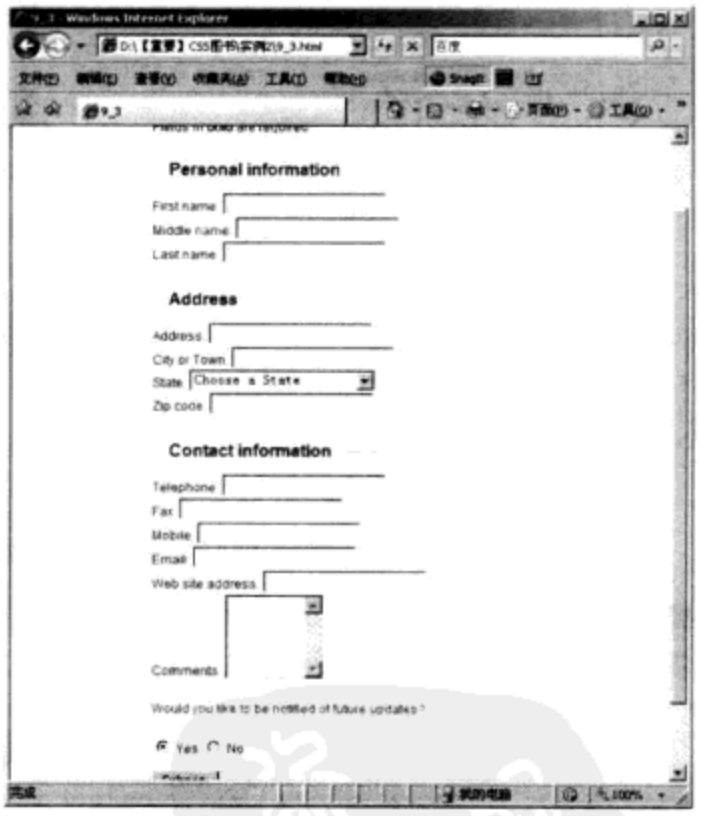


图 9-26　为 legend 标签添加 CSS 样式的预览效果

（4）制作标签 label 的 CSS 代码，代码如下。

```
label {
    float: left;              /*设置向左浮动样式*/
    width: 100px;             /*设置宽度*/
    padding: 0 1em;           /*设置内边距*/
    text-align: right;        /*设置文本右对齐*/
}
```

在以上代码中，使用 label 标签指定式选择符为表单项的标题名称设置浮动属性，同时定义标题名称的宽度、内间距以及文本对齐属性，使表单的整体工整、美观，预览效果如图 9-27 所示。

（5）制作填充项的 CSS 代码，代码如下。

```
fieldset div {
    margin-bottom: .5em;                                    /*设置下外边距*/
    padding: 0;                                             /*设置内边距*/
    display: block;                                         /*设置块级元素显示方式*/
}
fieldset div input, fieldset div textarea {
    width: 150px;                                           /*设置宽度*/
```

```
    border-top: 1px solid #555;                    /*设置上边框样式*/
    border-left: 1px solid #555;                   /*设置左边框样式*/
    border-bottom: 1px solid #ccc;                 /*设置下边框样式*/
    border-right: 1px solid #ccc;                  /*设置右边框样式*/
    padding: 1px;                                  /*设置内边距*/
    color: #333;                                   /*设置文字颜色*/
}
fieldset div select {
    padding: 1px;                                  /*设置内边距*/
}
```

在以上代码中，使用包含选择符分别为每个表单项的 div 容器、输入文本框、输入文本域和下拉框设置样式，主要重新设置了文本框、文本域的边框样式，调节每个表单项的间距，适当地留一定的留白，使页面简单精致，预览效果如图 9-28 所示。

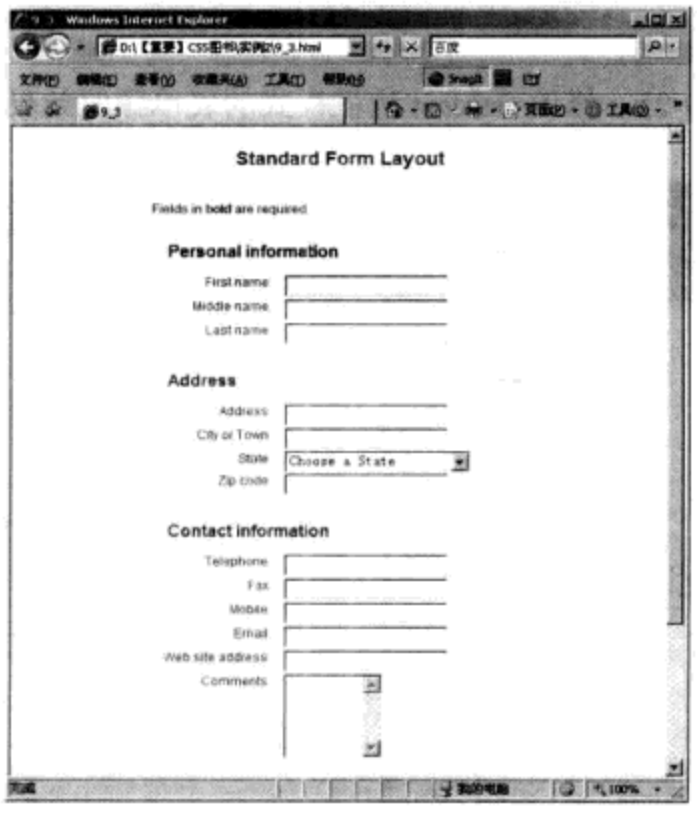


图 9-27　为 label 标签设置样式的预览效果

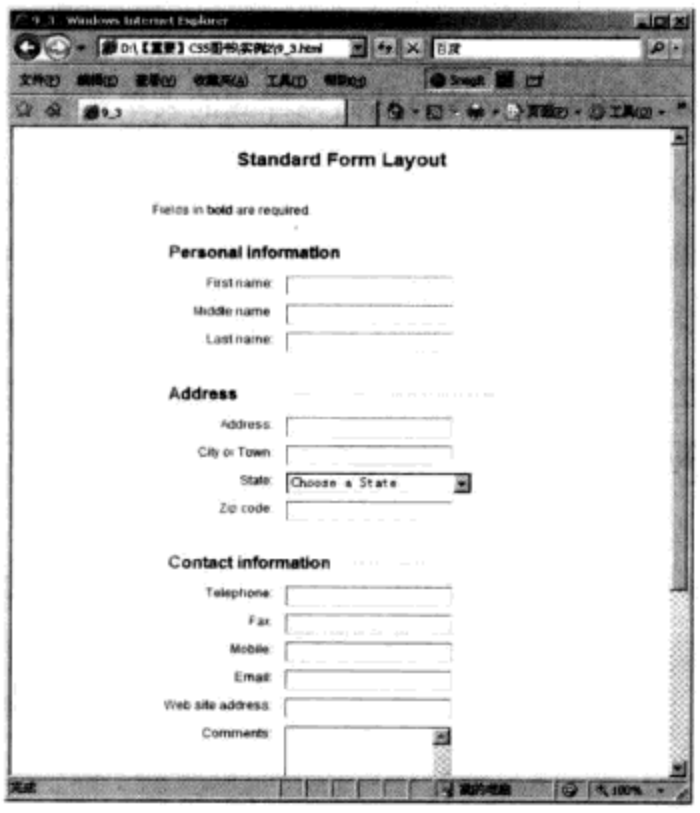


图 9-28　为输入文本框、文本域及下拉框添加样式的预览效果

（6）制作单选按钮的 CSS 代码，代码如下。

```
div.fm-multi div {
    margin: 5px 0;                                 /*设置外边距*/
}
div.fm-multi input {
    width: 1em;                                    /*设置宽度*/
}
div.fm-multi label {
    display: block;                                /*设置块级元素显示方式*/
    width: 200px;                                  /*设置宽度*/
    padding-left: 5em;                             /*设置左内边距*/
    text-align: left;                              /*设置文本左对齐*/
}
```

在以上代码中，使用组合选择符为单选按钮设置了 CSS 样式。在 div.fm-multi label 标签中设置了 display: block 样式，使每个单选按钮独立一行显示，这里面的选项分别是“Yes”和“No”，预览效果如图 9-29 所示。

（7）制作“submit”按钮的 CSS 代码，代码如下。

```
#fm-submit {
    clear: both;                                    /*清除左右浮动元素*/
    padding-top: 1em;                               /*设置上内边距*/
    text-align: center;                             /*设置文本居中对齐*/
}
#fm-submit input {
    border: 1px solid #333;                         /*设置边框样式*/
    padding: 2px 1em;                               /*设置内边距*/
    background: #555;                               /*设置背景颜色*/
    color: #fff;                                    /*设置文字颜色*/
}
```

以上代码设置了“submit”按钮的 CSS 样式。在#fm-submit 标签中，使用 clear:both 属性使按钮独立一行，禁止了两边的浮动元素出现。XHTML 按钮是有默认样式的，这里通过#fm-submit input 标签，重新设置了按钮的边框样式、内边距、背景色、文字颜色及字号，预览效果如图 9-30 所示。

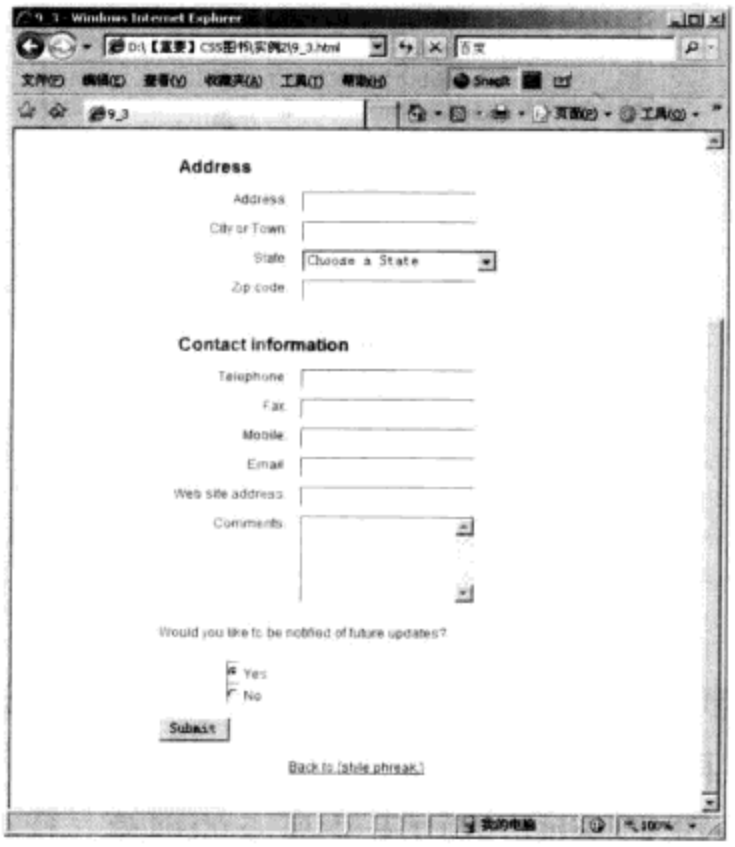


图 9-29　添加单选按钮样式的预览效果

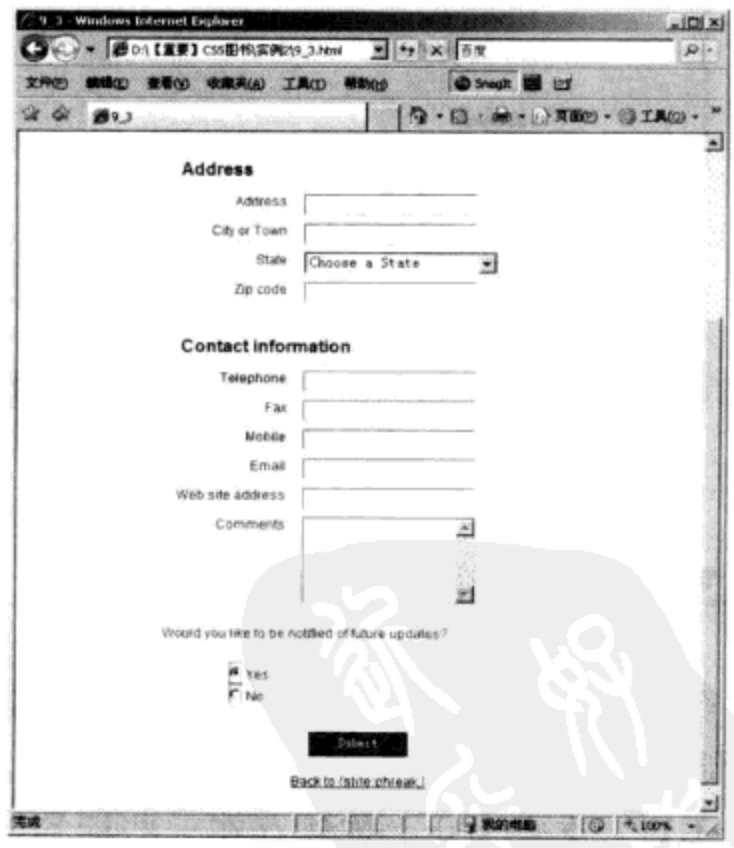


图 9-30　添加按钮样式的预览效果

（8）补充一些相关 CSS 样式，代码如下。

```
input:focus, textarea:focus {
    background: #efefef;                            /*设置背景颜色*/
    color: #000;                                    /*设置文字颜色*/
}
```

```
fieldset div.fm-req {
    font-weight: bold;                                    /*设置文字为粗体*/
}
```

在以上代码中，使用伪类:focus 在输入文本框和输入文本域获得焦点时向它们添加特殊的样式，即添加背景色和文字颜色样式。这种伪类在 IE 浏览器中是不显示的，在 Firefox 浏览器等标准浏览器中可以显示，如图 9-31 所示。

使用组合选择符 fieldset div.fm-req，为必填的内容添加醒目提示样式，即字体加粗显示。这样，整个表单页面就制作完成了，通过 IE 7、Firefox 及 IE 6 浏览器测试，没有不兼容问题，最终预览效果如图 9-32 所示。

图 9-31　Firefox 浏览器中的获得焦点效果

图 9-32　符合 W3C 标准的表单预览效果

9.3.3　兼容问题

符合 W3C 标准的页面，顾名思义，就是使用能够兼容标准浏览器的 CSS 样式代码。这个实例制作的是符合 W3C 标准的表单，因此，不应该再存在兼容问题。

不过，迄今为止，依然存在一些标签以及 CSS 在某些浏览器中无法显示的现象，这些是无法通过兼容问题能够解决的，它是代码本身的缺陷。例如本实例的单选按钮，在 IE 浏览器中显示带有边框的样式，而在 Firefox 这样的标准浏览器中，却能够得到很好的显示，如图 9-33 所示。

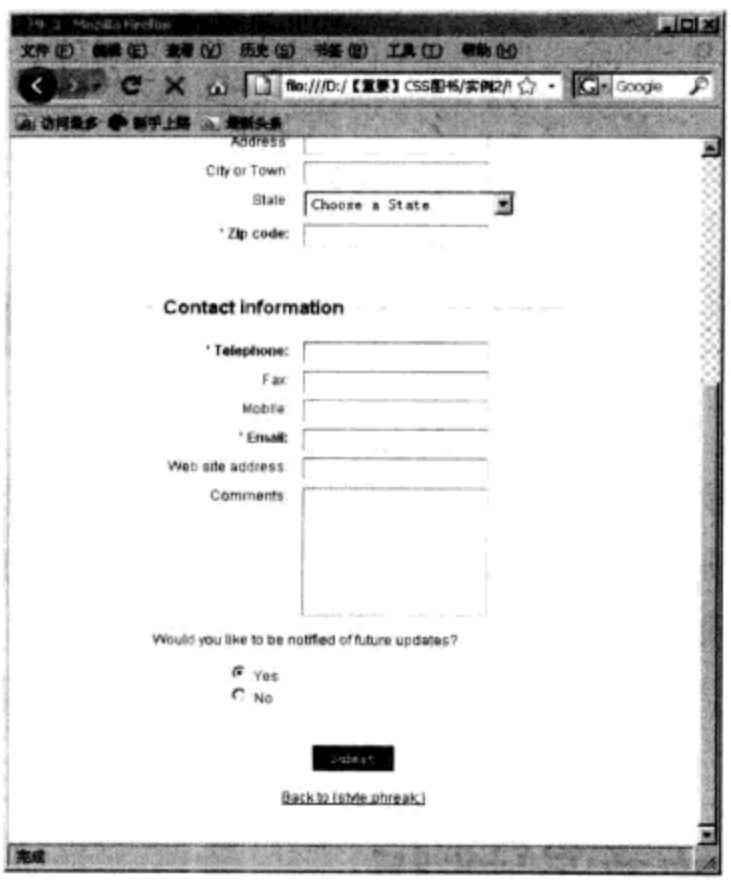


图 9-33　在 Firefox 浏览器中单选按钮得到完美显示

9.4　小结

本章讲解使用 CSS 制作表单的实例，包括了制作登录表单、制作用户注册的表单、制作符合 W3C 标准的表单，主要涉及了 1 个 XHTML 标签 label。

对本章的知识点前后联系进行归纳总结，结构导图如图 9-34 所示。

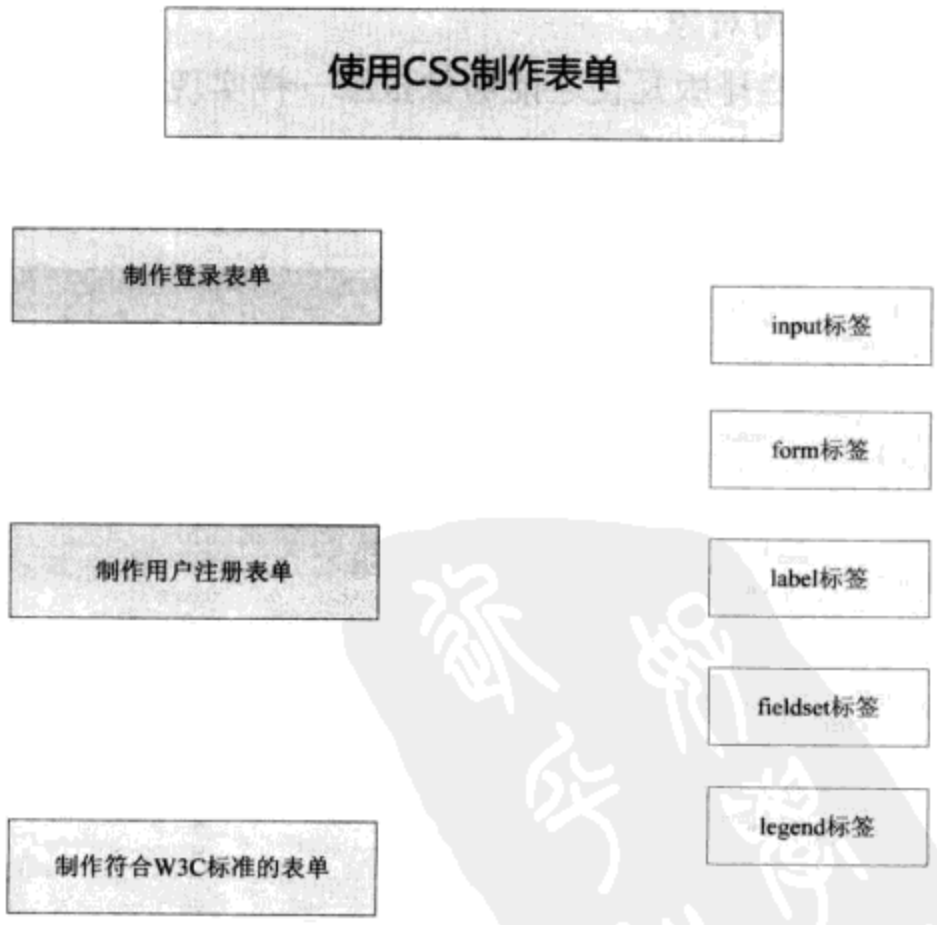


图 9-34　本章知识点结构导图

9.5　习题

1．简述表单的作用。

2．简述 label 标签的语法结构及功能。

3．如何使用表格对表单元素进行排版？

4．什么是符合 W3C 标准的表单？

第10章 使用CSS制作内容的版式

网页设计的第一步是设计版面布局。就像传统的报刊杂志编辑一样，将网页看作一张报纸或者一本杂志来进行排版布局。虽然动态网页技术的发展使得人们开始趋向于学习场景编剧，但是固定的网页版面设计基础依然是值得学习和掌握的。

本章将结合图片和文字，在CSS布局的页面中讲解如何控制图文之间的关系。

10.1 制作分栏的文字排版

在出版物的应用中，一般比较重视图文排版的样式。文字排版一般有两种形式，即通栏排版和分栏排版。进行通栏排版是直接将段落文字放置于p或其他对象中，再对段落文字应用间距、行距、字号等样式控制，便形成了排版雏形。通栏排版是以新闻、法律以及文字内容为主的对象。

分栏排版是使之能够像报纸一样实现分栏显示的排版。在CSS 2技术中，实现类似报纸的二栏或三栏式排版，必须借助于二列布局一样的两个div浮动定位而形成二列的空间，然后再将文字分别填充于二列之中。文本内容进行分栏排版的网页如图10-1所示。

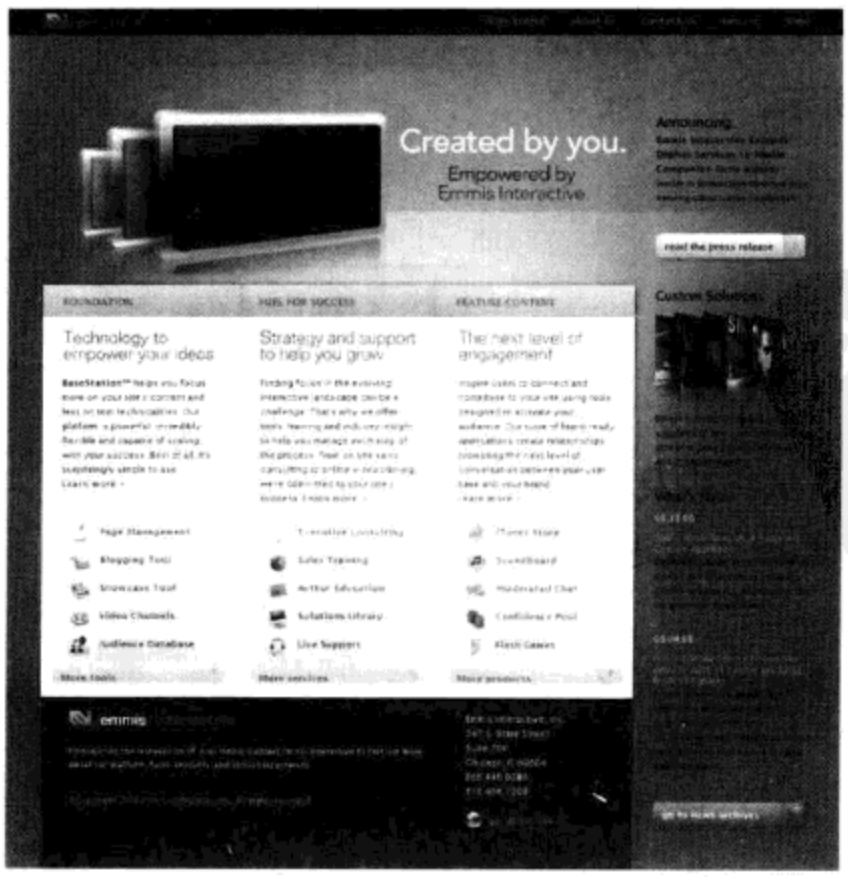


图10-1 采用分栏排版的网页

本节将详细介绍分栏的文字排版样式的实现方法。

注意：

CSS 的基本布局将在第三篇重点介绍，二列布局便是基本布局中的一种。

div 用于对某一个区域的标识，而二列的布局需要使用两个 div。现在使用二列布局的方法来制作分栏的文字排版，制作步骤如下。

（1）制作分栏排版页面的 XHTML 代码。

```
<div id="layout">
    <div class="col">[…]</div>
    <div class="col">[…]</div>
</div>
```

以上代码使用了两个 class 为 col 的 div 的对象。现在所需要做的是为它们指定宽度，然后让两个 div 在水平方向并排显示，从而形成二列式布局。

（2）制作排版的 CSS 样式，代码如下。

```
p{
    line-height: 180%;                                    /*设置行高*/
    font-size: 12px;                                      /*设置字号*/
    color: #FFFFFF;                                       /*设置文字颜色*/
}
#layout{
    width: 500px;                                         /*设置宽度*/
    height: 300px;                                        /*设置高度*/
    margin: 0px auto;                                     /*设置外边距*/
    background: url(images/10_1.jpg) no-repeat;           /*设置背景样式*/
}
.col{
    width: 200px;                                         /*设置宽度*/
    padding: 8px 25px;                                    /*设置内边距*/
    float: left;                                          /*设置向左浮动显示*/
}
```

在以上代码中，首先使用标签指定式选择符定义了 p 元素的行高为 180%，也就是字体高度的 180%，并定义了字号、文字颜色。然后，为整个页面#layout 元素指定了宽度 500px 和高度 300px，同时指定了背景图片，背景平铺，使页面美观，背景图片如图 10-2 所示，指定#layout 居中于浏览器显示。

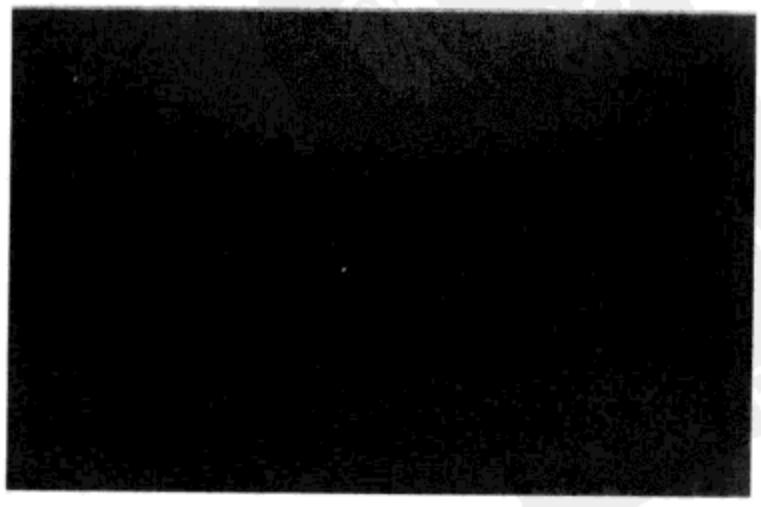


图 10-2　背景图片

最后，直接对用于分栏的.col 元素编写了宽度 200px，内边距上下为 8px、左右为 25px，

目的是使栏与栏之间留出空隙，方便阅读浏览，同时向左浮动，这样实现了二栏式排版，预览效果如图 10-3 所示。

图 10-3　分栏排版预览效果

10.2　制作图文混合排版

在文章段落之中经常会需要插入图片，在 CSS 布局之中，可以通过控制 CSS 代码对图片进行控制，实现图文混排的效果。

10.2.1　图文混排基本方式

在开始图片排版之前，先来探讨图片和文本排版的几种基本样式。

（1）图文混排基本样式一

现在准备了一个段普通的文章页面，应用图片只需将图片插入到所有段落文字的最前面，XHTML 代码如下。

```
<div id="layout">
    <h1>CSS 问答</h1>
    <div class="pimg"><img src="images/10_2.jpg" /></div>
    <p>[…]</p>
</div>
```

在以上代码中，为图片添加了一个 div，是因为考虑到随后在图片上会增加一些细节排版，将图片设计成一个 div 包含，使用 div 来控制 img 对象。直接插入图片后的预览效果如图 10-4 所示。

（2）图文混排基本样式二

现在的 div 对象保持着 div 占据整行的显示方式，尝试改变图像居中。添加了 CSS 样式，代码如下。

```
.pimg{
    text-align: center;   /*设置文本居中*/
}
```

在以上代码中，text-align 属性用于控制对象内的内容居中显示，应用这个属性之后，img 对象在 div 之中也能够保持居中状态，修改后的预览效果如图 10-5 所示。

图 10-4　图文混排基本样式一

（3）图文混排基本样式三

在实际应用中，有时也会应用文字绕图片的排版方式，这时可以使用浮动定位的方法，通过设定对象的 float 属性来使文字内容流入对象的旁边。修改 CSS 样式，代码如下。

```
.pimg{
    float: left;                                    /*设置向左浮动显示*/
    padding: 10px;                                  /*设置内边距*/
}
```

在以上代码中，在图片居左之后，为了使图片与文字有一定的空间，我们使 div 对象具有了 10px 的内边距，预览效果如图 10-6 所示。

图 10-5　图文混排基本样式二

图 10-6　图文混排基本样式三

10.2.2　制作网站图文混排版式的实例

采用单独的 div 对象定位图片与文字是目前网站上非常普遍的做法，通过对象的定位能够很好地形成一个复合的排版布局，在页面内容上也能给用户带来最大的方便。

（1）制作图文混排页面的 XHTML 代码。

```
<div id="layout">
    <h3>XML 标记语言的发展</h3>
    <div class="pimg"><img src="images/10_4.jpg" /></div>
    <p>[…] </p>
</div>
```

在以上代码中，使用一个 id 为 layout 的 div 对象，放置了标题、图片、文章正文内容的信息，具有复杂的结构用以显示这些信息。h3 用以显示标题，p 对象用以显示正文内容，而对于焦点图片则放置于一个 div 对象中用来控制其样式，焦点图片如图 10-7 所示。

网页 XHTML 结构预览效果如图 10-8 所示，在没有添加任何样式的情况下，属于图文混排基本样式一的类型。

图 10-7　焦点图片

图 10-8　XHTML 结构预览效果

（2）制作#layout 元素的 CSS 样式，代码如下。

```
#layout{
    background: url(images/10_3.jpg) no-repeat;          /*设置背景样式*/
    width: 400px;                                        /*设置宽度*/
    height: 300px;                                       /*设置高度*/
    padding: 8px 10px;                                   /*设置内边距*/
}
```

以上代码为整个页面定义了背景图片，不平铺，同时指定了背景的宽度和高度，保证背景图片完整显示；使用了类选择符，还指定了内边距，把主体内容显示在页面的正中位置，背景图片如图 10-9 所示。此时预览效果如图 10-10 所示。

从图 10-10 可以看出，文本主体仍然在背景区域之外，并且排版也不美观，所以继续添加 CSS 样式。

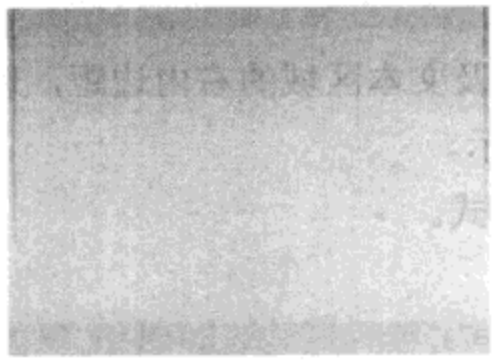


图 10-9　背景图片

图 10-10　添加#layout 样式的页面预览效果

（3）制作 h3 元素的 CSS 样式，代码如下。

```
h3{
    color: #333333;                          /*设置文字颜色*/
    border-bottom: dotted 1px #B1B4B2;       /*设置边框样式*/
    display: block;                          /*设置块状元素显示方式*/
    width: 240px;                            /*设置宽度*/
    line-height: 30px;                       /*设置行高*/
    margin-left: 20px;                       /*设置左外边距*/
}
```

以上代码把标题区域的显示方式定义成了块状显示，这样可以为其指定宽度，定义了下边框为点状线，将标题与正文用一条线分隔开，使排版更有设计感，凸显标题，预览效果如图 10-11 所示。

图 10-11　添加了 h3 样式的页面预览效果

（4）制作 p 元素的 CSS 样式，代码如下。

```
p {
    font-size: 12px;                    /*设置字号*/
    color: #B1B4B2;                     /*设置文字颜色*/
    line-height: 18px;                  /*设置行高*/
    padding-right: 25px;                /*设置右内边距*/
}
```

以上代码定义了正文内容文本的大小、颜色、行高以及文本区域的右内边距，以至于文本不会顶到页面的边界，排版美观，预览效果如图 10-12 所示。

现在还是没有实现图文的合理排版，继续添加 CSS 样式。

（5）制作图片的 CSS 样式，代码如下。

```
.pimg {
    float: left;                        /*设置向左浮动定位*/
    padding: 6px;                       /*设置内边距*/
    padding-top: 0px;                   /*重新设置上内边距*/
}
```

以上代码为图片指定了 CSS 样式，采用的是 10.2.1 小节介绍到的第 3 种排版样式，使图片向左浮动，同时定义了内边距，使图片和文字之间留出一定空间。这样图文混排页面的实例制作完成，预览效果如图 10-13 所示。

图 10-12　添加 p 样式的页面预览效果

图 10-13　图文混排页面预览效果

10.3　制作全图排版的实例

全图的排版最常用在相册类网站中，这类网站往往有大量的图片需要在同一页面显示。如图 10-14 所示是一个全图排版的网页。

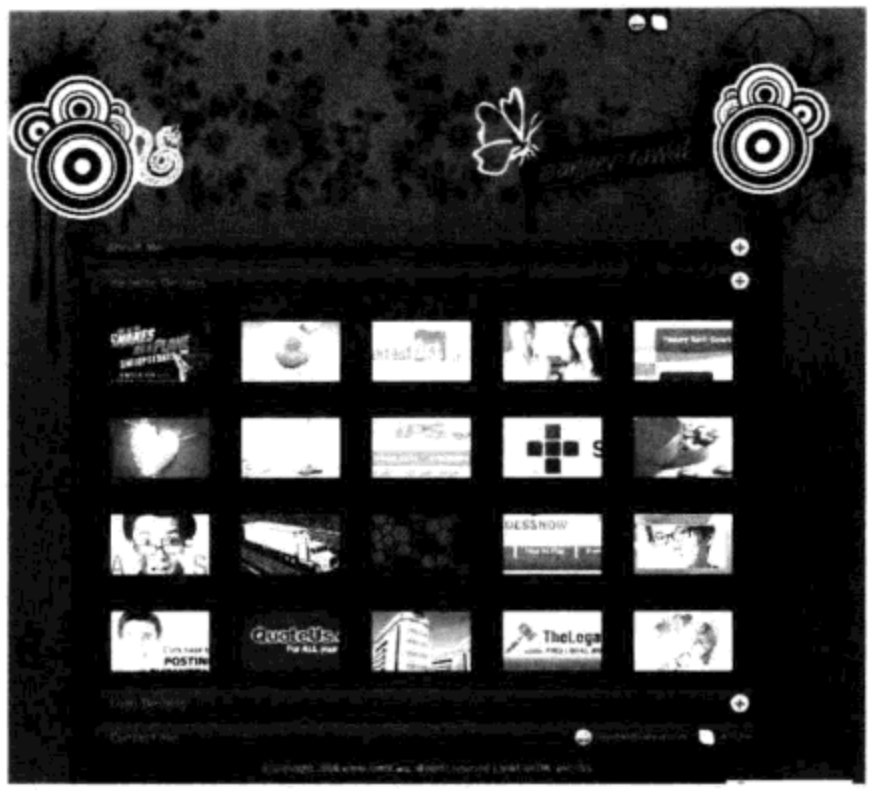


图 10-14　全图排版的网页

使用 CSS 布局进行全图排版的核心在于对浮动定位的控制，而且具有很高的灵活性，可以通过 CSS 随时改变全图排版的样式。

10.3.1　自由浮动布局

浮动布局方式有很多种，本节采用左浮动，不限制页面宽度的自由流入方式布局，所有浮动块根据浏览器宽度大小而改变布局方式。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    <h1>PHOTO GALARRY</h2>
    <div>
        <img src="images/10_5.jpg" />
        <h2>Title:Flower</h2>
        <h3>Data:2008-5-10</h3>
    </div>
    <div>
        <img src="images/10_6.jpg" />
        <h2>Title:Flower</h2>
        <h3>Data:2008-5-10</h3>
    </div>
    <div>
        <img src="images/10_7.jpg" />
        <h2>Title:Flower</h2>
        <h3>Data:2008-5-10</h3>
    </div>
    <div>
        <img src="images/10_8.jpg" />
        <h2>Title:Flower</h2>
        <h3>Data:2008-5-10</h3>
    </div>
```

```
        [...]
</div>
```

在以上代码中，还是以#layout 作为主容器，其中使用了 div 来放置图片，其目的在于控制图片的宽度。有时候图片并不是等宽的，因此为了在排列上整齐，也使用 div 来装入图片，通过控制 div 的宽度相等来实现等宽的排列。

全图排版完成后，继续为图片添加了一些信息，例如图片名称、拍摄时间等，分别放在 h2 和 h3 元素中。页面结构预览效果如图 10-15 所示。从图 10-15 可以看出，这时候的图文都是各自独立一行一行地排列着，下面为页面结构添加 CSS 样式，重新排列它们的位置。

（2）制作背景以及标题的 CSS 样式，代码如下。

```
body{
    background-color: #000000;                          /*设置背景颜色*/
}
h1{
    font-size: 18px;                                    /*设置字号*/
    font-family: Arial;                                 /*设置字体类型*/
    color: #FFF;                                        /*设置文字颜色*/
    display: block;                                     /*设置块级元素显示方式*/
    border-bottom: 1px solid #666666;                   /*设置下边框样式*/
    line-height: 30px;                                  /*设置行高*/
    padding-bottom: 8px;                                /*设置下内边距*/
}
```

在以上代码中，首先，将页面的背景定义为黑色，因为这是图片展示，黑色背景能够烘托出前景的各种颜色的图片。然后，在 h1 元素中定义了标题的字号、字体、颜色等一些属性，预览效果如图 10-16 所示。

从图 10-16 可以看出，图片下面的文字看不见了，这是因为背景色定义为黑色和还没有设置样式文字的默认颜色相同，所以需要继续添加样式。

（3）制作#layout div 元素的 CSS 样式，代码如下。

```
#layout div{
    border: 1px solid #999999;                          /*设置边框样式*/
    float: left;                                        /*设置向左浮动定位*/
    margin: 3px;                                        /*设置外边距*/
    padding: 2px;                                       /*设置内边距*/
    width: 126px;                                       /*设置宽度*/
    text-align: center;                                 /*设置文本居中对齐*/
    height: 110px;                                      /*设置高度*/
    overflow: hidden;                                   /*剪切超过边界的内容*/
}
```

为了实现布局，为每个#layout 下的 div 图片区域添加了 1px 的边框线，并全部采用向左浮动，让图片可以顺序浮动排列，设置外边距控制与其他图像的间距效果。

设定了图片的宽度，使得所有图片框具有统一的宽度，排版美观，并且设置了 padding 内边距使图片、文字与边缘有一定的空间。为了保证因为文本过长而导致 div 被拉高的问题，这里也使用了 overflow:hidden 属性把访问内容隐藏，预览效果如图 10-17 所示。

图 10-15　页面结构预览效果

图 10-16　添加 body 和 h1 的样式预览效果

（4）制作图片的 CSS 样式，代码如下。

```
#layout div img{
    margin-top: 5px;                                    /*设置上外边距*/
}
```

以上代码使用包含选择符#layout div img 为图片设置了上外边距，使图片与边框留出一段距离，预览效果如图 10-18 所示。

图 10-17　使用 overflow:hidden 属性隐藏访问内容的预览效果

图 10-18　添加 img 标签样式的预览效果

（5）制作标题 h2、h3 的 CSS 代码，代码如下。

```
h2,h3{
    font-size: 11px;                                    /*设置字号*/
    font-family: Arial;                                 /*设置字体类型*/
    margin: 0px;                                        /*设置外边距*/
    padding: 0px;                                       /*设置内边距*/
    color: #FFFFFF;                                     /*设置文字颜色*/
```

```
}
```

以上代码使用并列的标签指定式选择符，为 h2 和 h3 标题标签设定了文字样式，如图 10-19 所示。从图 10-19 可以看出，这时候可以清晰地看到文本标题了，不过标题名称和时间没有明显的样式区别，继续添加样式。

（6）制作标题 h3 特别的 CSS 代码，代码如下。

```
h3{
    font-weight: normal;                    /*设置字体*/
    color:#CCCCCC;                          /*设置文字颜色*/
}
```

在以上代码中，为标签指定式选择符 h3 单独设置了文本样式为字体正常粗细、文字颜色为 #CCCCCC，这样时间和名称从样式上就有了明显的区别。至此，全图排版的实例制作完成，预览效果如图 10-20 所示。

图 10-19　为标题文字添加样式的预览效果

图 10-20　全图排版预览效果

10.3.2　其他显示方式布局

使用 CSS 来进行图片排版非常灵活，可以根据需要随时改变排列方式，进行行间操作，将布局改变，这是表格式布局做不到的。

现在希望将 10.3.1 节的图片排列固定为一行两个，并显示更大的文字标题。

（1）对放置图片的 div 的宽高改变 CSS 样式，代码如下。

```
#layout div{
    border: 1px solid #999999;              /*设置边框样式*/
    float: left;                            /*设置向左浮动定位*/
    margin: 3px;                            /*设置外边距*/
    padding: 2px;                           /*设置内边距*/
    width: 275px;                           /*设置宽度*/
    text-align: center;                     /*设置文本居中对齐*/
    height: 110px;                          /*设置高度*/
```

```
    overflow: hidden;                                    /*剪切超过边界的内容*/
}
```

在以上代码中，将宽度修改为 275px。另外，新定义了#layout 元素的宽度，代码如下。

```
#layout{
    width: 600px;                                        /*设置宽度*/
}
```

预览效果如图 10-21 所示。

（2）设置图片与文字之间的浮动对齐方式的 CSS 样式，代码如下。

```
#layout div img{
    margin-top: 5px;                                     /*设置上外边距*/
    float: left;
    margin-left: 5px;
}
h2,h3{
    font-size: 11px;                                     /*设置字号*/
    font-family: Arial;                                  /*设置字体类型*/
    margin: 0px;                                         /*设置外边距*/
    padding: 0px;                                        /*设置内边距*/
    color: #FFFFFF;                                      /*设置文字颜色*/
    float: right;
    width: 65%;
}
h2{
    font-size: 18px;                                     /*设置字号*/
}
```

在以上代码中，去除了#layout div 中的 text-align:center 属性之后，再将所有图片定义成向左浮动，使得右侧留出空间显示文字。为了保证右侧的文字以对齐的方式显示，将 h2 和 h3 设定了宽度，并且向右浮动定位。由于宽度一定，使得所有文本在 div 之中的距离一致，这样就形成了新的全图排版样式，预览效果如图 10-22 所示。

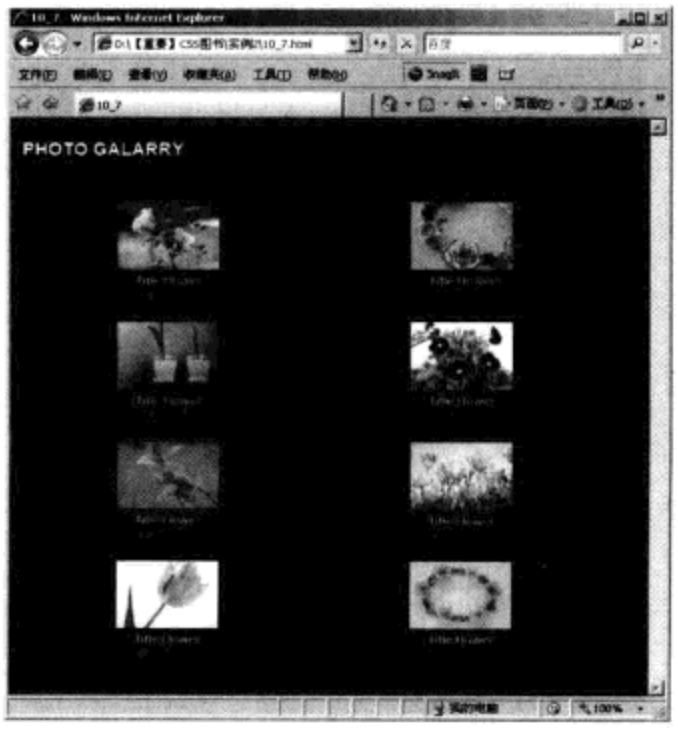


图 10-21　修改#layout div 元素宽度的预览效果

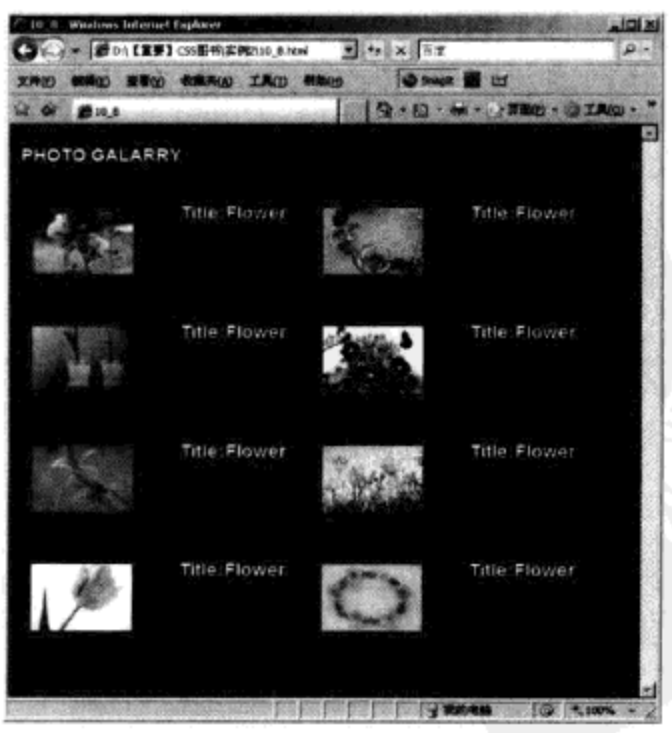


图 10-22　图片和文字浮动后的全图排版样式预览效果

10.4　小结

本章介绍使用 CSS 进行内容版式设计的方法，包括文本的分栏排版、图文的混合排版、全图排版等实例。其中图文混合排版介绍了 3 种图文混排的基本类型，全图排版介绍了通过 CSS 的浮动属性进行的多种多样的自由浮动布局。

对本章的知识点前后联系进行归纳总结，结构导图如图 10-23 所示。

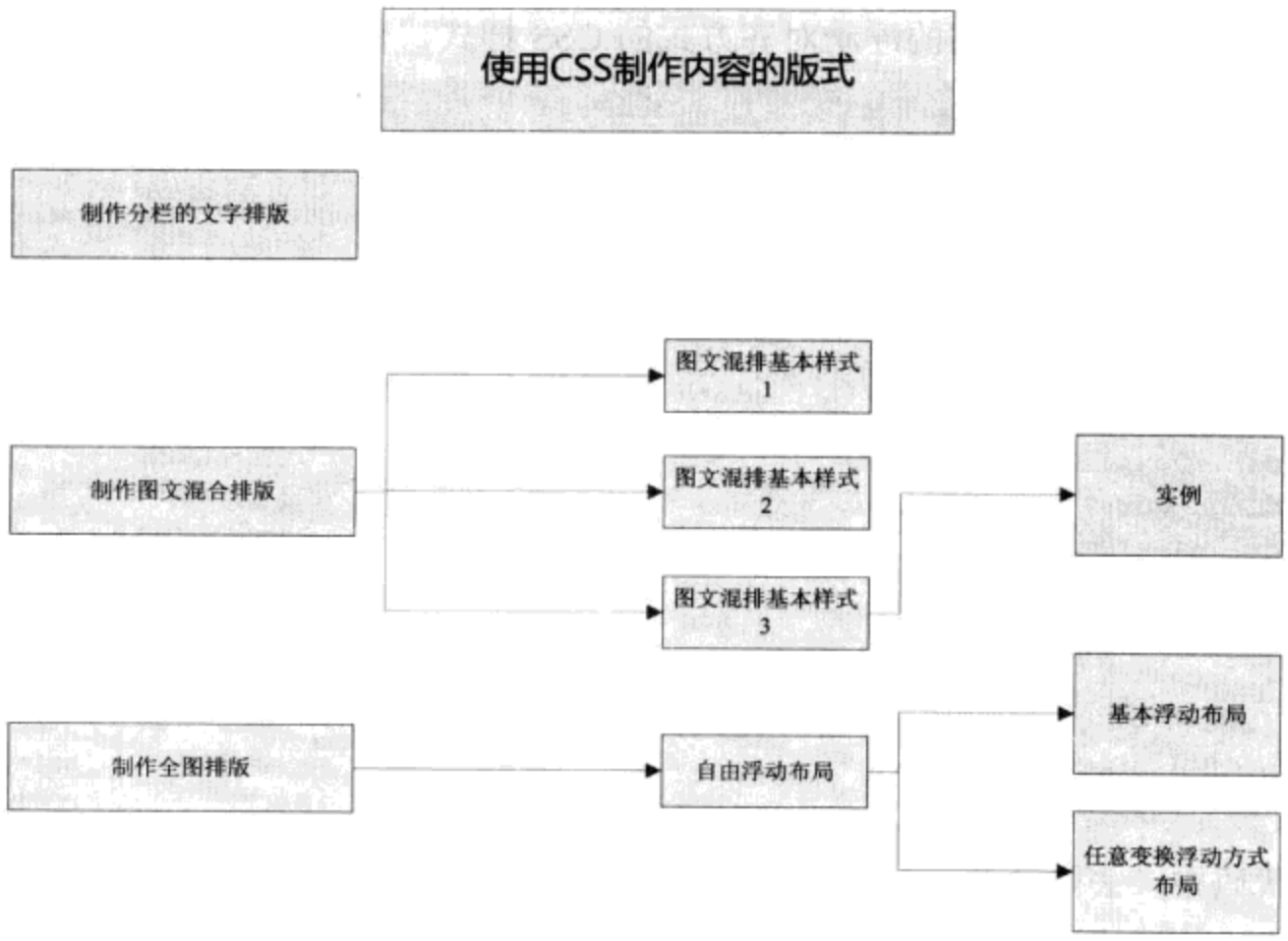


图 10-23　本章知识点结构导图

10.5　习题

1. 简述文字排版的两种类型。
2. 简述分栏的文字排版样式的实现方法。
3. 图文混排的基本方式有哪些？
4. 使用 CSS 布局进行全图排版的核心是什么？

第11章 使用 CSS 制作链接样式

整个网站都是由超级链接串连而成的，无论是首页还是每一个频道页面，甚至是其他网站，都由链接完成了页面间的跳转。CSS 对链接的样式控制是通过伪类来实现的，在导航设计中曾简单介绍过链接的相关设计方法，本章将详细探讨有关链接对象 a 的几种伪类使用及实际应用。

11.1 制作 Windows 风格样式的 CSS 按钮

CSS 能够帮助实现不同状态下的样式，可以借助这些特性制作具有交互特征的按钮。在操作系统中按钮大量存在，凸起状的按钮与下凹状的按钮非常的直观和方便，就像真实键盘一样。

图 11-1 Windows XP 样式和经典样式的按钮

本节借助前面介绍过的边框、背景以及链接的 4 种状态来制作一个 Windows XP 风格的按钮。通过实例分别来实现 Windows 经典样式和 XP 样式，如图 11-1 所示。

11.1.1 仿 Windows 经典样式的 CSS 按钮

首先来实现经典样式的按钮效果。Windows 经典样式实际上是基于对 CSS 中 border 的颜色控制实现的。

（1）制作按钮的 XHTML 代码。

```
<a href="#" class="classic">经典样式</a>
```

（2）制作按钮的 CSS 样式，代码如下。

```
body{
    background-color: #D4D0C8;                    /*设置背景颜色*/
}
a{
    display: block;                               /*设置块级元素显示方式*/
    float: left;                                  /*设置向左浮动定位*/
    color: black;                                 /*设置文字颜色*/
```

```
    font-family: "宋体";                          /*设置字体类型*/
    font-size: 12px;                              /*设置字号*/
    font-weight: bold;                            /*设置字体为粗体*/
}
a.classic{
    text-decoration: none;                        /*设置链接文字文本修饰样式*/
    background-color: #D4D0C8;                    /*设置背景颜色*/
    border: 1px solid #FFFFFF;                    /*设置边框样式*/
    border-right-color: #808080;                  /*重新设置右边框颜色*/
    border-bottom-color: #808080;                 /*重新设置下边框颜色*/
    padding: 5px;                                 /*设置内边距*/
}
a.classic:hover{
    background-color: #EEE9E0;                    /*设置背景颜色*/
}
a.classic:active{
    border: 1px solid #FFFFFF;                    /*设置边框颜色*/
    border-top-color: #808080;                    /*重新设置上边框颜色*/
    border-left-color: #808080;                   /*重新设置左边框颜色*/
    background-color: #D4D0C8;                    /*设置背景颜色*/
}
```

在以上代码中，为了使最终效果能够实现，先将 body 设置成了灰色背景。

然后，在复合型选择符 a.classic:hover 元素中，对链接进行了重新设置。使用 display:block 属性，使链接成为一个块状对象，以方便定义相关样式。对于凸起的按钮而言，之所以能够有按钮的感觉，关键是因为对按钮的 4 个边框进行了颜色设置，使上边框和左边框为白色，就像光线从左上方照下一样，同时使右边框和下边框设置为灰色，模仿光线为照到的部分呈现出来的阴影。这样按钮的视觉效果就呈现的栩栩如生。

接下来，在 a.classic:hover 元素的设置中，对背景颜色进行了稍微的提亮，使得用户光标悬停时会有色彩变化。

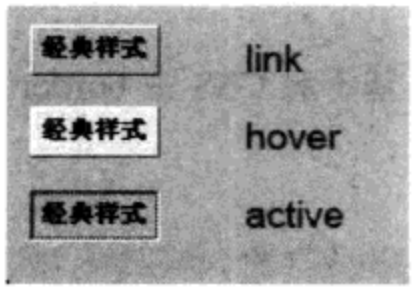


图 11-2　经典风格的按钮预览效果

最后，在 a.classic:active 元素的设置中，定义的是对按钮按下状态的表现样式。实际网站设计中，伪类:active 状态并不是很常用。在 active 状态下，将对象的光线反过来，使上边框和左边框呈现为灰色，右边框和下边框呈现为白色，感觉上就像按钮被按下了一样。这样经典风格的操作系统按钮制作完成，预览效果如图 11-2 所示。

11.1.2　仿 Windows XP 风格的 CSS 按钮

Windows XP 风格样式的按钮是使用图片样式来控制的。下面讲解制作仿 Windows XP 风格的 CSS 按钮的具体步骤。

（1）制作按钮的 XHTML 代码。

```
<a href="#" class="xp">XP 样式</a>
```

（2）制作按钮的 CSS 样式，代码如下。

```
body{
    background-color: #FFFFFF;                    /*设置背景颜色*/
}
a{
    display: block;                               /*设置块级元素显示方式*/
    float: left;                                  /*设置向左浮动定位*/
    color: black;                                 /*设置文字颜色*/
    font-family: "宋体";                          /*设置字体类型*/
    font-size: 12px;                              /*设置字号*/
    font-weight: bold;                            /*设置字体为粗体*/
}
a.xp{
    text-decoration: none;                        /*设置链接文字文本链接样式*/
    background: url(images/link.gif) no-repeat;   /*设置背景样式*/
    width: 73px;                                  /*设置宽度*/
    height: 19px;                                 /*设置高度*/
    text-align: center;                           /*设置文本居中对齐*/
    line-height: 17px;                            /*设置行高*/
}
a.classic:hover{
    background: url(images/hover.gif) no-repeat;  /*设置背景样式*/
}
a.classic:active{
    background: url(images/active.gif) no-repeat;     /*设置背景样式*/
}
```

在以上代码中，为按钮的 3 种状态准备了 3 张背景图片，分别是 link、hover 和 active，在交互时调用，最终预览效果如图 11-3 所示。

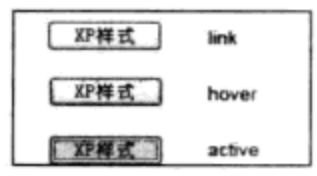


图 11-3　XP 风格的按钮预览效果

11.2　制作仿按钮下陷效果的实例

超链接的 hover 状态，通常可以通过改变颜色、增加或去除下划线等效果来提醒访客。也看到过有一些文字链接，当光标移上去时（hover 状态）产生位移，类似于按钮陷下的效果，这样的实现方法，其实是应用了 position 属性，代码如下。

```
position: relative;
top:1px;
left:1px
```

这段代码的含义为，相对定位，距上部位移 1 像素，距左部位移 1 像素。可以将上面的代

码加入到超链接的 hover 状态中，就实现了类似按钮下陷的效果。本节就将介绍如何制作超链接类似按钮下陷效果的实例。

（1）制作页面的 XHTML 代码。

```
<a href="#"><img src="images/11_1.jpg" ></a>
</p>
```

在以上代码中，使用标签 a 制作一个图片超链接，预览效果如图 11-4 所示，蓝色边框是 img 标签默认的样式。观察图 11-4，并且记住它的位置，下面为这个页面添加 CSS 样式。

（2）制作 img 标签的 CSS 样式，代码如下。

```
img {
    border:none;                                    /*设置边框样式*/
}
```

在以上代码中，使用 border:none 属性取消了图片默认的边框样式，预览效果如图 11-5 所示。

图 11-4　图片超链接预览效果

图 11-5　添加 img 标签样式的预览效果

（3）制作 a 标签的 CSS 样式，代码如下。

```
a {
    text-decoration:none;                           /*设置超链接文本样式*/
    color:#c00;                                     /*设置文字颜色*/
}
a:hover {
    position: relative;                             /*设置相对定位*/
    top:1px;                                        /*相对定位，设置顶部距离*/
    left:1px;                                       /*相对定位，设置左边距离*/
    color:#666;                                     /*设置文字颜色*/
}
```

在以上代码中，使用伪类 a:hover 为按钮设计了交互效果，当光标经过按钮上方时，按钮图片相对地向下和向右各移 1px。

11.3　面包屑导航链接

面包屑导航这个词在国内并不常用，这里所说的面包屑导航，是指网站上经常出现的路径

式导航，如“首页>新闻频道>新闻全文”。而这种导航在国外常常被形象的称为面包屑（英文“crumb”）。如图 11-6 所示是 msdn.com 网站的面包屑导航。

图 11-6　msdn.com 网站的面包屑导航示例

现在仿制 msdn.com 网站的导航，制作一个面包屑导航系统。

11.3.1　制作实例

（1）制作按钮的 XHTML 代码。

```
<div id="crumb">
    <a href="#">CSS 图书</a>
    <a href="#">第一篇 CSS 实例制作</a>
    <a href="#">第 14 章制作 CSS 链接样式</a>
</div>
```

在以上代码中，三个链接和一个 div 组成了面包屑导航的 XHTML 结构代码，对网站中的一个成组元素而言，使用一个 div 或其他对象将其嵌套起来是必要的，首先从代码结构上使得这些元素中的零散的组件组合在了一起，而且从视觉设计上来说，一般一组元素都是一起进行位置调整，页面结构预览效果如图 11-7 所示。

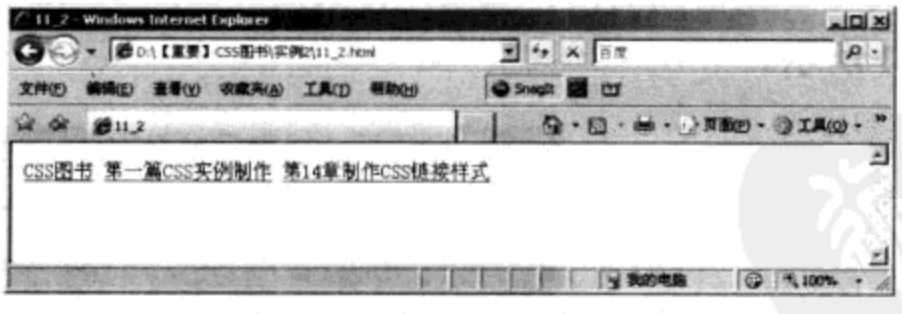


图 11-7　页面结构预览效果

从图 11-7 可以看出，这时候的导航系统只是文本的超级连接，继续添加样式达到预期的样式。

（2）制作 body 的 CSS 样式，代码如下。

```
body{
    background: url(images/11_2.gif) repeat-x;      /*制作背景样式*/
    padding-top: 35px;                              /*设置上内边距*/
}
```

在以上代码中，使用标签指定式选择符 body 为导航系统添加背景，背景图如图 11-8 所示。

为 body 元素指定了背景图片，横向平铺，使用颜色分明的背景图片形成一条文字带。定义上边距，保证链接文字准确定位在要求的位置，预览效果如图 11-9 所示。

图 11-8　面包屑背景图片

图 11-9　添加 body 样式的预览效果

（3）制作面包屑导航的 CSS 代码，代码如下。

```
#crumb a{
    float: left;                                    /*设置向左浮动定位*/
    color: #333;                                    /*设置文字颜色*/
    font-size: 14px;                                /*设置字号*/
    font-family: Arial;                             /*设置字体类型*/
    text-decoration: none;                          /*设置链接文字文本修饰*/
    display: block;                                 /*设置块级元素显示方式*/
    background: url(images/11_1.gif) no-repeat;     /*设置背景样式*/
    padding: 1px 2px 1px 20px;                      /*设置内边距*/
    margin: 0 5px;                                  /*设置外边距*/
}
```

在以上代码中，导航是由文本的超级连接组成的，所以制作面包屑导航就是设计超级连接的 CSS 样式。

导航项的每个 a 标签都使用向左浮动定位，主要有几个原因：为每个超链接的前面添加图标 icon，并且为了超链接之间保持一定的距离，需要设置外边距和内边距；为链接对象放置一张背景图片，用于实现每个链接前面的图标效果，背景图片如图 11-10 所示。

预览效果如图 11-11 所示。

图 11-10　背景图片

图 11-11　添加 a 样式的预览效果

（4）制作超链接交互的 CSS 代码，代码如下。

```
#crumb a:hover{
    border: 1px solid #999999;                      /*设置边框样式*/
    background-color: #F8F8F8;                      /*设置背景颜色*/
}
```

在#crumb a:hover 元素中，定义了边框和背景色，使得当前悬停的项目能实现一个方框状的效果。

注意：

在 a 的两种状态下，如果边框不一样，容易出现内容显示区域的位移，也就是说在标准状态如果没有 1px 的边框，在光标悬停时又出现了 1px 的边框，内容就会出现 1px 的偏移，引起页面上的跳动显示。本节的实例会发生这种偏移的现象，在视觉效果允许的情况下，应该在 a 状态时添加上 1px 的边框。

这样，面包屑导航系统的实例制作完成，预览效果如图 11-12 所示。

图 11-12　面包屑导航系统预览效果

11.3.2　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。

1．Firefox 浏览器测试

把实例所用到的网页文件放在 Firefox 浏览器中打开，预览效果如图 11-13 所示。

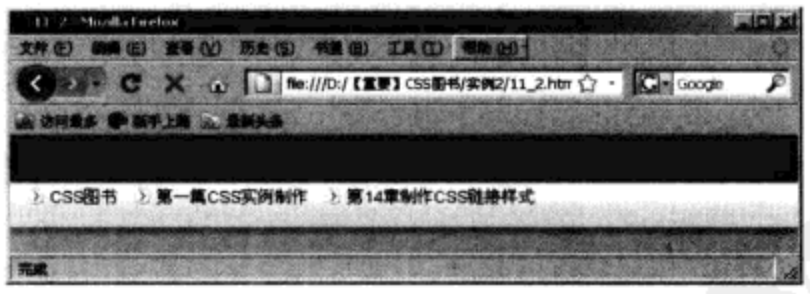


图 11-13　Firefox 浏览器中的预览效果

从图 11-13 可以看出，导航的位置发生了错位，整体向上偏移。针对此问题，修改 CSS 代码，如下。

```
body{
    background: url(images/11_2.gif) repeat-x;
    padding-top: 35px;
    padding-top: 40px !important
}
```

在以上代码中，添加了一行代码，重新设置上内边距，并且使用 CSS hack 的方法，使这段重新设置的样式应用在 Firefox 浏览器中得到显示，预览效果如图 11-14 所示。

2. IE 6 浏览器测试

在 IE 6 浏览器中的中预览效果如图 11-15 所示。

从图 11-15 可以看出，实例在 IE 6 浏览器中没有出现兼容问题。这样，这个实例完成了浏览器兼容的测试，并解决了兼容问题。

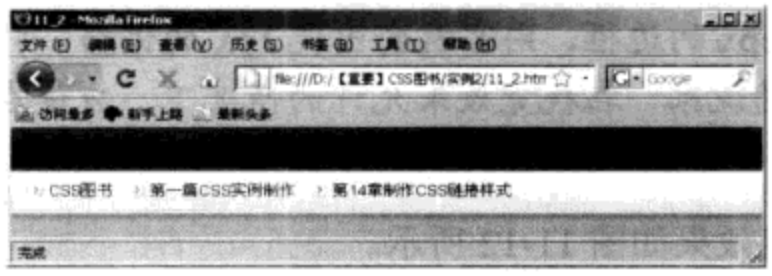


图 11-14　解决兼容问题后 Firefox 浏览器中的预览效果

图 11-15　IE 6 浏览器中的预览效果

11.4　小结

本章讲解了使用 CSS 制作文字和图片链接的实例，包括了 Windows 操作系统风格样式的 CSS 按钮、仿按钮下陷效果的实例以及面包屑文字导航链接。其中在制作仿 Windows 操作系统风格样式的 CSS 按钮过程中，又分别介绍了仿经典样式的按钮和仿 XP 风格的 CSS 按钮。对本章的知识点前后联系进行归纳总结，结构导图如图 11-16 所示。

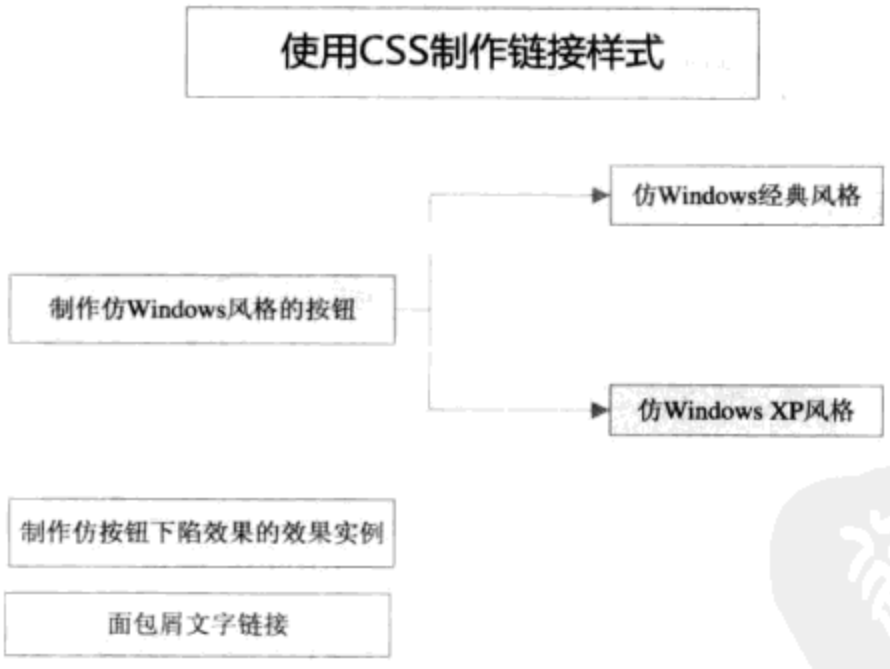


图 11-16　本章知识点结构导图

11.5　习题

1. CSS 对链接的样式控制是通过什么来实现的？
2. Window 经典样式和 Windows XP 风格的 CSS 按钮实现有什么不同？
3. 按钮下陷效果的实现原理是什么？
4. 简述什么是面包屑导航链接。

第12章 使用CSS制作数据表格

表格作为一种非常特殊而且实用的数据表达方式，从没有淡出设计师的视野，因为有很多数据需要通过表格这种形式来体现。本章将介绍如何使用表格代码和CSS来控制表格样式。

12.1 制作基本的数据表格

table标签是网页布局中的常用标签，它把各种内容、数据放在表格单元格内，例如文本、图片、链接、表格等。

但那只是在CSS+DIV布局流行之前，在XHTML时代，不推荐使用table标签来进行布局和定位，但table标签因为它突出的特性还拥有相当多的用武之地。

12.1.1 表示数据的表格对象标签

XHTML代码的设计者充分考虑到了显示数据用的表格应当会有哪些内容出现，因此提供了许多表格使用的专门标签，如表12-1所示。

表12-1 表格使用的专门标签

名称	描述
caption	定义表格的名称
tbody	定义表格内容区，如果一个表格由多个内容区构成，可以使用多个tbody标签
thead	定义表头
tfoot	定义表格页脚
th	定义表头用的单元格

在以上标签中，如果使用thead来标记表头元素部分的话，不必对tr进行特别的class指定，只需使用table thead选择符，便可以对表头部分进行样式设定了，其他标签也是如此。

另一方面，这些标签从名称上就可以看出各部分的含义，例如tfoot（table foot）表示表格页脚，从代码可读性上看也是相当实用。

12.1.2 使用表格标签制作表格

现在制作一个使用这些表格标签制作数据表的实例，XHTML代码如下。

```
<table id="dataList">
    <caption>浏览器兼容性一览表</caption>
    <thead>
        <tr>
            <th>CSS 特征</th>
            <th>MSIE6.0</th>
            <th>Firefox1.0</th>
            <th>Firefox1.5</th>
            <th>Opera 8.5</th>
        </tr>
    </thead>
    <tbody id="html">
        <tr>
            <th colspan="5" class="title">HTML4.01</th>
        </tr>
        <tr>
            <th>a</th>
            <th>81%</th>
            <th>85%</th>
            <th>67%</th>
            <th>89%</th>
        </tr>
        <tr>
            <th>abbr</th>
            <th>N</th>
            <th>97%</th>
            <th>85%</th>
            <th>76%</th>
        </tr>
        <tr>
            <th>acronyn</th>
            <th>94%</th>
            <th>79</th>
            <th>98%</th>
            <th>75%</th>
        </tr>
    </tbody>
    <tbody id="xhtml">
        <tr>
            <th colspan="5" class="title">XHTML1.0change</th>
        </tr>
        <tr>
            <th>HTML in XML</th>
            <th>N</th>
            <th>Y</th>
            <th>Y</th>
            <th>Y</th>
        </tr>
        <tr>
            <th>well-formed</th>
            <th>Y</th>
```

```
            <th>Y</th>
            <th>Y</th>
            <th>Y</th>
        </tr>
        <tr>
            <th>Media Type</th>
            <th>N</th>
            <th>Y</th>
            <th>Y</th>
            <th>Y</th>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="5">资料来源：</td>
        </tr>
    </tfoot>
</table>
```

从以上代码可以看出，在 caption 标签中放置了表格的标题。使用 thead 标签并在其中使用 td 标签放置表格的表头列，由于表格共有两个子表，因此使用 tbody 区分了两个子表区域，并给每个区域设定了自己的 id。

最后，使用了 tfoot 标记出了表格的页脚信息，最终形成了一个丰富的代码结构。这样使用 XHTML 代码制作出来的表格预览效果如图 12-1 所示。

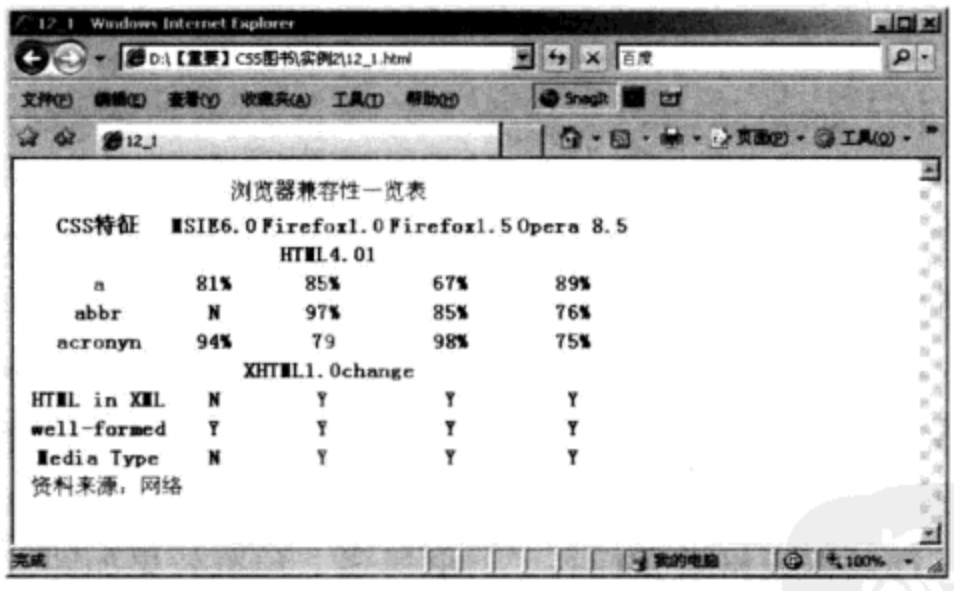


图 12-1　XHTML 代码编写的表格预览效果

从图 12-1 可以看出，使用制作表格的专门标签设计表格，没有 table 默认的边框样式了，这就为添加 CSS 样式提供了很大的方便。

12.1.3　使用 CSS 修饰表格样式

表格的样式与其他对象的样式没有什么区别，都是使用 margin、padding、border、background 这些属性对表格进行操作。而表格较复杂的是，对其中子级内容控制的对象更多。而使用表格标签可以方便地区分开各个区域，继而可以实现对表格各个区域的样式设定。本节将会使用一

种表格边框属性 border-collapse 来合并相邻的两个边框宽度。

1. border-collapse 属性语法结构

border-collapse 属性设置表格的边框是否被合并为一个单一的边框，还是像在标准的 XHTML 中那样分开显示。其属性语法结构如下。

```
border-collapse: collapse | separate
```

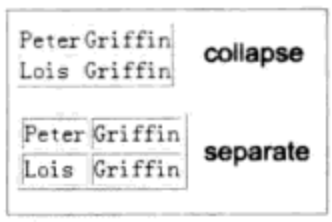


图 12-2 边框样式

- 各属性值含义如下。
- collapse：默认值，如果可能，边框会被合并为一个单一的边框。
- separate：边框会被分开。

以形象化的图示表示，如图 12-2 所示。

2. 制作表格的 CSS 样式

使用 CSS 样式表，可以为表格赋予丰富的样式，添加 CSS 代码的具体步骤如下。

（1）添加表格边框的 CSS 样式，代码如下。

```
#dataList{
    width: 700px;                                    /*设置宽度*/
    margin: 0px auto;                                /*设置外边距，实现对象居中*/
    border-collapse: collapse;                       /*设置边框合并*/
    font-family: Arial;                              /*设置字体类型*/
    text-align: center;                              /*设置文本居中*/
}
#dataList th,td{
    border: 1px solid #aaa;                          /*设置边框样式*/
}
```

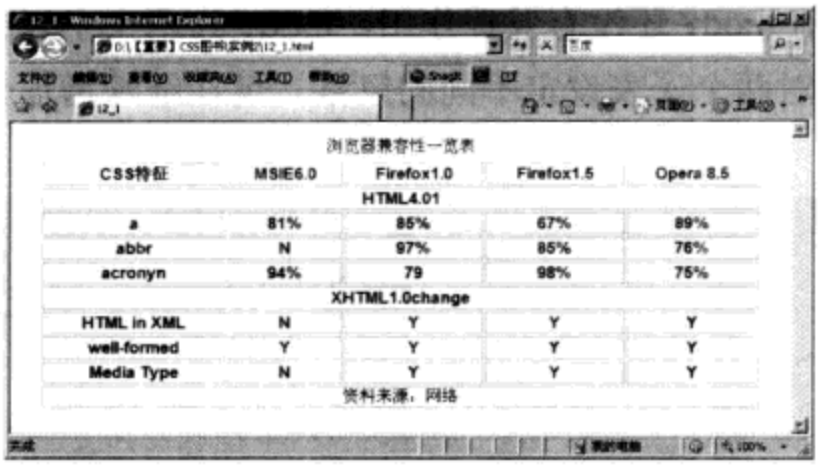
在以上代码中，在复合包含选择符#dataList th,td 元素中为表格中所有的 td 与 th，也就是所有的单元格进行 1px 的边框处理。

这段代码的重点，在于对表格对象#dataList 进行了 border-collapse: collapse 的定义，表示将表格中单元格之间的线条合并，如果不进行合并的话，每个单元格将拥有 1px 的边框，而两个邻近的单元格边框就是二者之和。与之相反的是使用 border-collapse:separate，它将使得各单元格的边框独立存在。这样数据表格首先拥有了边框样式，预览效果如图 12-3 所示。

（2）添加表头的背景色的 CSS 样式，代码如下。

```
#dataList thead th{
    border-bottom: 2px solid #3D580B;                /*设置下边框样式*/
    background-color: #8FC629;                       /*设置背景色*/
    color: #fff;                                     /*设置文字颜色*/
    padding: 10px 0px;                               /*设置内边距*/
}
```

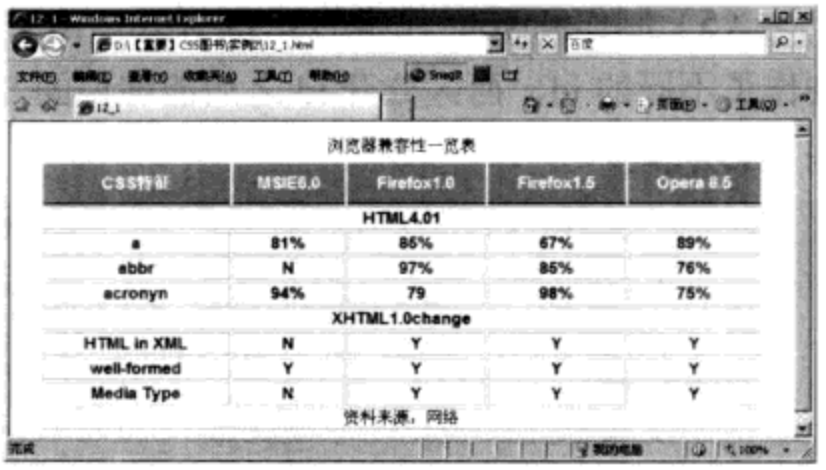
在以上代码中，需要给各个部分的 th 使用不同的背景色，首先给表格的主表头设置背景色。由于主表头已经使用了 thead 作为标签，直接应用#dataList thead th 这样的包含选择符，便可以控制主表头的颜色。为表头添加了背景颜色后，预览效果如图 12-4 所示。


浏览器兼容性一览表

CSS特征	MSIE6.0	Firefox1.0	Firefox1.5	Opera 8.5
HTML4.01				
a	81%	85%	67%	89%
abbr	N	97%	85%	76%
acronyn	94%	79	98%	75%
XHTML1.0change				
HTML in XML	N	Y	Y	Y
well-formed	Y	Y	Y	Y
Media Type	N	Y	Y	Y

资料来源：网络

图 12-3　具有边框的数据表格预览效果


浏览器兼容性一览表

CSS特征	MSIE6.0	Firefox1.0	Firefox1.5	Opera 8.5
HTML4.01				
a	81%	85%	67%	89%
abbr	N	97%	85%	76%
acronyn	94%	79	98%	75%
XHTML1.0change				
HTML in XML	N	Y	Y	Y
well-formed	Y	Y	Y	Y
Media Type	N	Y	Y	Y

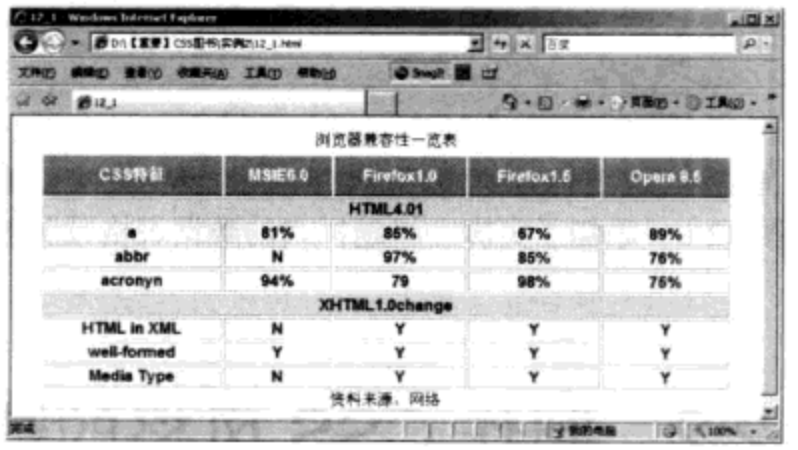
资料来源：网络

图 12-4　为表头添加背景色后的数据表格预览效果

（3）制作 th 元素的 CSS 样式，代码如下。

```
#dataList th{
    background-color: #F2F4B9;                    /*设置背景色*/
}
#dataList th.title{
    background-color: #E3E685;                    /*设置背景色*/
}
```

继续设定代码，控制所有的 th 颜色。td 主要有两部分，一个是每个子表的表头，另一个是每个子表的左列内容。由于前面在 XHTML 中对表头部分使用了 class 名为 title，因此在这里可以使用 class 选择符进行设定，预览效果如图 12-5 所示。


浏览器兼容性一览表

CSS特征	MSIE6.0	Firefox1.0	Firefox1.5	Opera 8.5
HTML4.01				
a	81%	85%	67%	89%
abbr	N	97%	85%	76%
acronyn	94%	79	98%	75%
XHTML1.0change				
HTML in XML	N	Y	Y	Y
well-formed	Y	Y	Y	Y
Media Type	N	Y	Y	Y

资料来源：网络

图 12-5　添加 th 颜色的数据表格预览效果

（4）制作页脚和标题的 CSS 样式，代码如下。

```
#dataList tfoot td{
    border-width: 0px;                            /*设置表格边框宽度*/
    text-align: right;                            /*设置文本右对齐*/
    font-size: 12px;                              /*设置字号*/
    color: #777;                                  /*设置文字颜色*/
}
#dataList caption{
    font-weight: bold;                            /*设置粗体*/
    padding: 6px 0px;                             /*设置内边距*/
    color: #3D580B;                               /*设置文字颜色*/
    font-size: 25px;                              /*设置字号*/
}
```

在以上代码中，最后一部分的样式由两部分组成，tfoot 部分是表格页脚，在页脚上不需要再出现边框效果，因此将边框宽度设为 0px，而剩下的一部分则是表格标题，我们为标题设置了相应的样式，预览效果如图 12-6 所示。

（5）制作#layout 元素的 CSS 样式。由于目前 tbody 还没有发挥作用，tbody 为两个子表分别作了不同的 id，而区分开 id 之后能够做的，是对两个子表设定不同的风格。现在继续增加一些 CSS 代码。

```
#xhtml th.title{
    background-color: #FFD56C;                    /*设置背景色*/
}
#xhtml th{
    background-color: #FFE8AE;                    /*设置背景色*/
}
```

在以上代码中，id 为 xhtml 的 tbody 部分子表重新定义表头和左列的颜色，最终预览效果如图 12-7 所示。

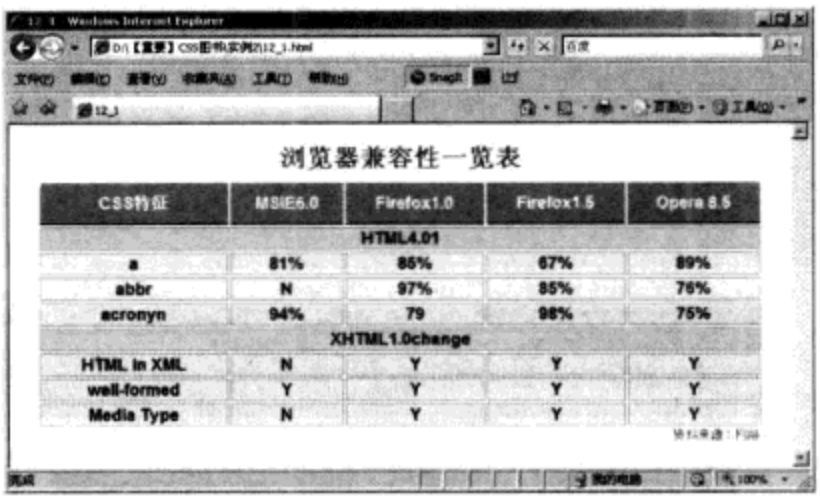


图 12-6　添加了页脚和标题样式的数据表格预览效果

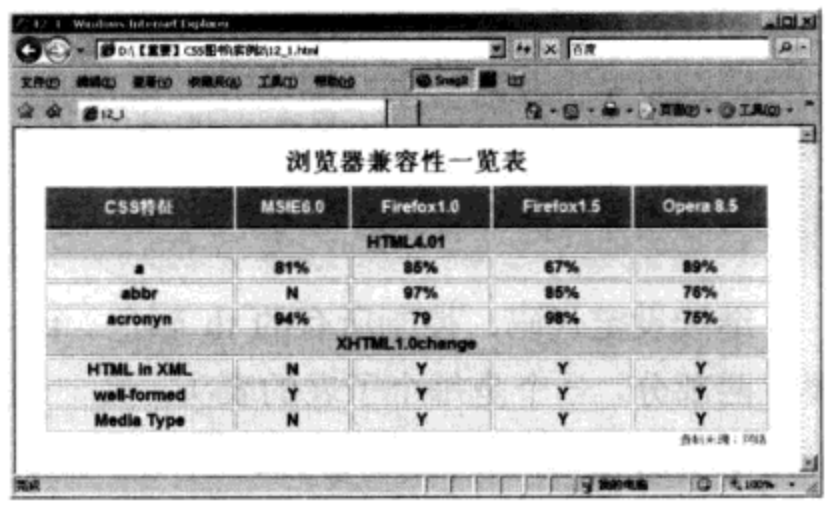


图 12-7　数据表格最终预览效果

12.2　制作 CSS 风格的数据表格

XHTML 中的 table 表格是为展示数据所用的。数据表格要具有可达性的特性。可达性就是物质本身所传达出来的信息的性质，具有良好的可达性的表格往往具有以下特点。

- 确保为表头，即使用 th 元素。
- 为表格添加标题（caption）。
- 为表格元素添加摘要（summary）属性。
- 如果需要，可以通过使用 thead、tfoot、tbody 元素将表格行分组。

本节讲解如何赋予表格的可达性，并且讲解如何用 CSS 来定义表格以获得需要的视觉效果。

12.2.1　表格对象标签属性

XHTML 代码中，th 标签用来定义表格内的表头单元格。此 th 标签内部的文本通常会呈现为粗体。th 标签中有两个重要的属性，即 scope 和 abbr。这两个属性的语法结构如下。

```
<th scope=" col | colgroup | row | rowgroup" abbr=" abbr_text"></th>
```

scope 属性有 4 个属性值，规定此单元格是否为以下部分提供表头信息。

- row：包含此单元格的行的其余部分。
- col：包含此单元格的列的其余部分。
- rowgroup：包含此单元格的行组的其余部分。
- colgroup：包含此单元格的列组的其余部分。

abbr 属性用来规定一个单元格中内容的缩写版本，这个可以在接下来的实例中观察使用方法。另外，table 标签中也有一个重要的属性，即 summary 属性，结构如下。

```
<table summary="说明文字"></table>
```

summary 属性就是对这个 table 的作用以及结构进行解释说明。

12.2.2　CSS 属性 text-transform

本节使用英文制作表格，这样做的目的之一是介绍一个 CSS 属性，即 text-transform，定义文本的大小写状态，这个属性对中文无意义。text-transform 属性语法结构如下。

```
text-transform: capitalize | uppercase | lowercase | none
```

这 4 个可用值的含义如下。

- capitalize：首字母大写。
- uppercase：所有字母转换为大写。
- lowercase：所有字母转换为小写。
- none：正常无变化。

下面将使用这样的属性来定义英文表格的表头。

12.2.3　制作数据表格实例

现在就来使用 XHTML 表格标签的 scope、abbr、summary 属性，以及 CSS 属性 text-transform 来制作一个具有 CSS 样式风格的 table 表格。

首先制作表格页面的 XHTML 代码结构。制作方法和 12.1 节制作的基本数据表格的方法类似，都是使用数据表格标签 table、caption、tr、th 等。

（1）制作表格页面的 XHTML 代码。

```
    <table id="mytable" cellspacing="0" summary="The technical specifications of the Apple
PowerMac G5 series">
        <caption>
        Table 1: Power Mac G5 tech specs
        </caption>
        <tr>
            <th scope="col" abbr="Configurations" class="nobg">Configurations</th>
            <th scope="col" abbr="Dual 1.8">Dual 1.8GHz</th>
```

```
        <th scope="col" abbr="Dual 2">Dual 2GHz</th>
        <th scope="col" abbr="Dual 2.5">Dual 2.5GHz</th>
    </tr>
    <tr>
        <th scope="row" abbr="Model" class="spec">Model</th>
        <td>M9454LL/A</td>
        <td>M9455LL/A</td>
        <td>M9457LL/A</td>
    </tr>
    <tr>
        <th scope="row" abbr="G5 Processor" class="specalt">G5 Processor</th>
        <td class="alt">Dual 1.8GHz PowerPC G5</td>
        <td class="alt">Dual 2GHz PowerPC G5</td>
        <td class="alt">Dual 2.5GHz PowerPC G5</td>
    </tr>
    <tr>
        <th scope="row" abbr="Frontside bus" class="spec">Frontside bus</th>
        <td>900MHz per processor</td>
        <td>1GHz per processor</td>
        <td>1.25GHz per processor</td>
    </tr>
    <tr>
        <th scope="row" abbr="L2 Cache" class="specalt">Level2 Cache</th>
        <td class="alt">512K per processor</td>
        <td class="alt">512K per processor</td>
        <td class="alt">512K per processor</td>
    </tr>
</table>
```

在以上代码中，在 table 表格标签中使用了 summary 属性，用来指定对这个表格进行解释说明的文字内容。在每一个表头 th 标签中，使用了 scope 和 abbr 属性。

scope="col"属性为"Configurations"的表头，作为其他列的其余部分表头，而 scope="row"属性作为其行的其余部分表头。abbr 作为单元格内容的缩写版本，表格结构预览效果如图 12-8 所示。

从图 12-8 可以看出，字体为粗体的单元格部分就是表头部分，有行表头，也有列表头。为了表格内的信息看起来更加醒目，表格更加美观，继续添加 CSS 样式。

（2）制作 body 标签的 CSS 样式，代码如下。

```
body {
    font: normal 11px auto "Trebuchet MS",                    /*设置字体样式*/
    color: #4f6b72;                                           /*设置文字颜色*/
    background: #E6EAE9;                                      /*设置背景颜色*/
}
```

在以上代码中，使用标签指定式选择符 body 为表格设置了背景颜色、表格内的字体类型、字号及文字颜色等，预览效果如图 12-9 所示。

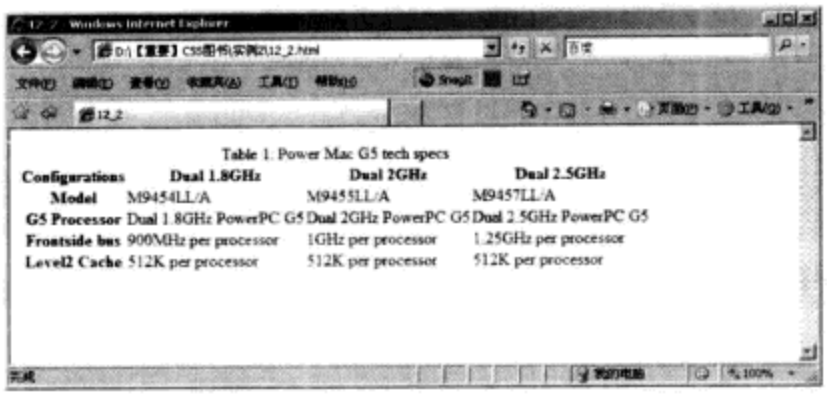


图 12-8　表格结构预览效果

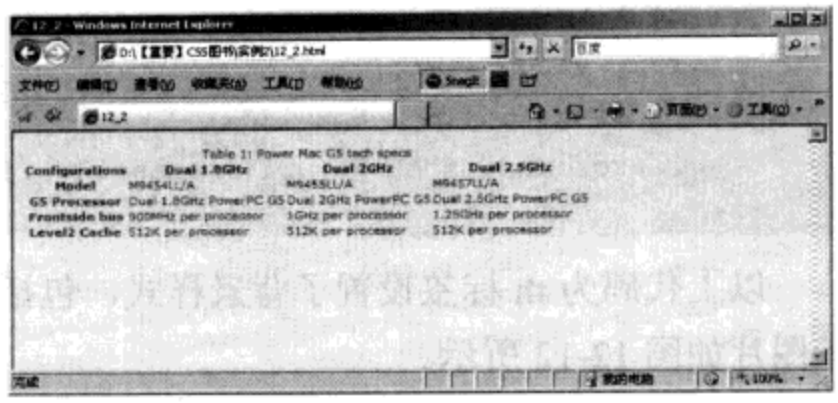


图 12-9　添加 body 样式预览效果

从图 12-9 可以看出，此时表格的行和列并没明显分隔对齐，继续添加样式。

（3）制作#mytable 的 CSS 代码，代码如下。

```
#mytable {
    width: 700px;                                    /*设置宽度*/
    padding: 0;                                      /*设置内边距*/
    margin: 0;                                       /*设置外边距*/
}
```

以上代码设置了#mytable 的宽度和内外边距，使得表格足够宽以容纳信息，表格的行和列能看出明显对齐分隔开来，预览效果如图 12-10 所示。

（4）制作 caption 的 CSS 代码，代码如下。

```
caption {
    padding: 0 0 5px 0;                              /*设置内边距*/
    width: 700px;                                    /*设置宽度*/
    font: italic 11px "Trebuchet MS";                /*设置字体*/
    text-align: right;                               /*设置文本右对齐*/
}
```

以上代码使用标签指定式选择符caption为表格名称设置了基本的样式，预览效果如图12-11所示。

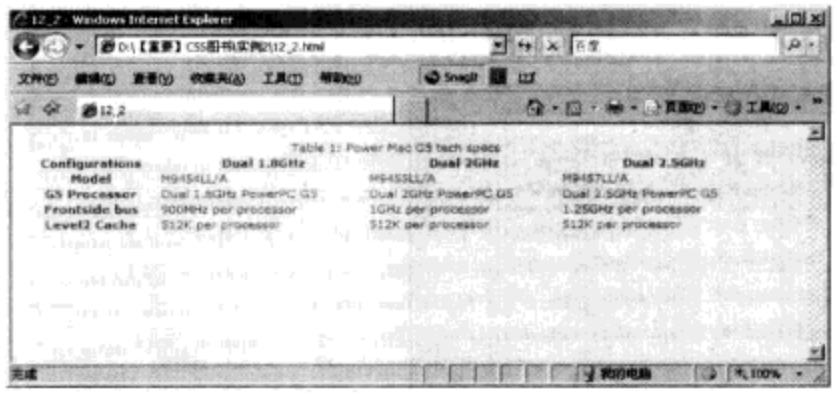


图 12-10　添加#mytable 样式的预览效果

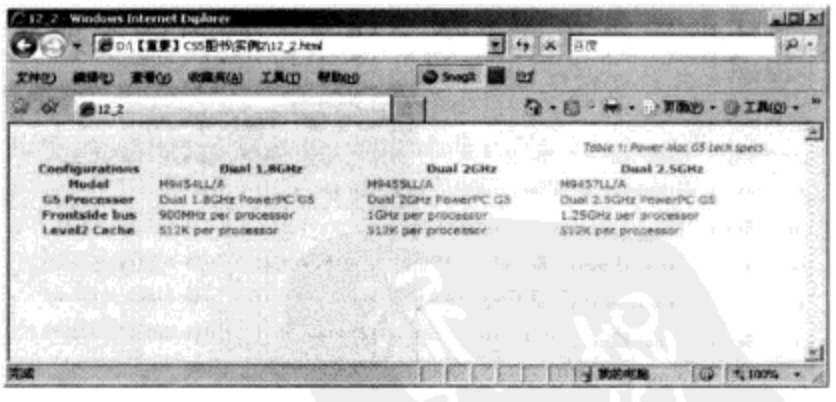


图 12-11　添加 caption 样式的预览效果

（5）制作 th 的 CSS 代码，代码如下。

```
th {
    font: bold 11px "Trebuchet MS";                  /*设置字体*/
    color: #4f6b72;                                  /*设置文字颜色*/
    border-right: 1px solid #C1DAD7;                 /*设置右边框样式*/
    border-bottom: 1px solid #C1DAD7;                /*设置下边框样式*/
    border-top: 1px solid #C1DAD7;                   /*设置上边框颜色*/
    letter-spacing: 2px;                             /*设置字幕间距*/
```

```
    text-transform: uppercase;                                    /*设置小写字母转换为大写字母*/
    text-align: left;                                             /*设置文本左对齐*/
    padding: 6px 6px 6px 12px;                                    /*设置内边距*/
    background: #CAE8EA url(images/12_1.jpg) no-repeat;           /*设置背景样式*/
}
```

以上代码为 th 标签设置了背景样式，包括背景颜色、背景图片以及背景图片平铺方式，背景图片如图 12-12 所示。

使用了 text-transform 属性，把表头的英文全部转换成了大写字母形式，如图 12-13 所示，同时可以对比图 12-8。

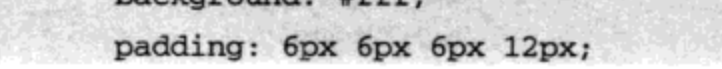


图 12-12　背景图片

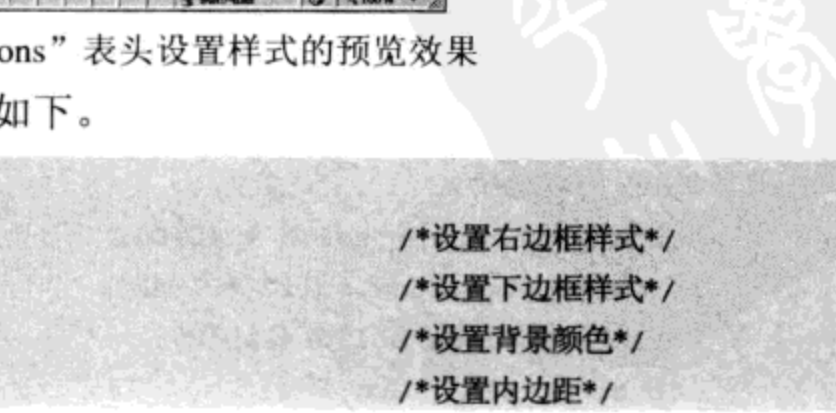


图 12-13　添加 th 样式的预览效果

（6）制作“configurations”表头的 CSS 代码，代码如下。

```
th.nobg {
    border-top: 0;                                                /*设置上边框样式*/
    border-left: 0;                                               /*设置左边框样式*/
    border-right: 1px solid #C1DAD7;                              /*设置右边框样式*/
    background: none;                                             /*设置背景*/
}
```

在以上代码中，通过组合选择符 th.nobg 单独为“configurations”表头设置了样式，重新定义了边框样式并且去除了背景，使得这个表头与众不同，预览效果如图 12-14 所示。

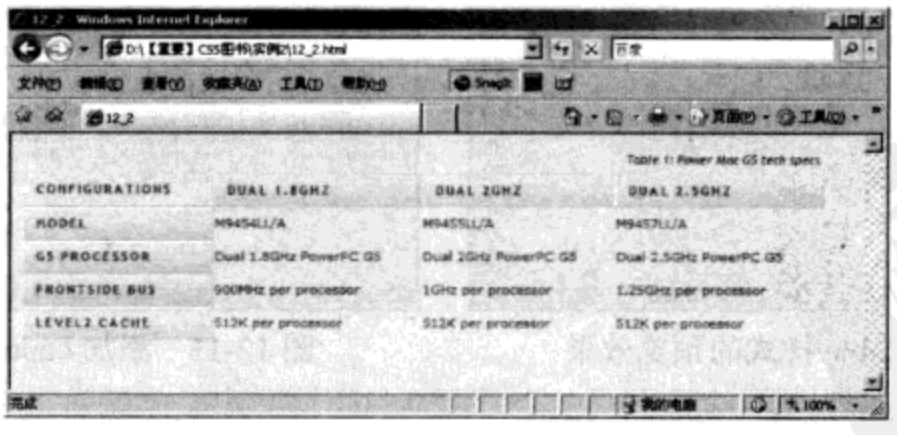


图 12-14　单独为“configurations”表头设置样式的预览效果

（7）制作 td 的 CSS 代码，具体 CSS 代码如下。

```
td {
    border-right: 1px solid #C1DAD7;                              /*设置右边框样式*/
    border-bottom: 1px solid #C1DAD7;                             /*设置下边框样式*/
    background: #fff;                                             /*设置背景颜色*/
    padding: 6px 6px 6px 12px;                                    /*设置内边距*/
```

```
    color: #4f6b72;                                          /*设置字体颜色*/
}
```

以上代码为 td 标签设置了边框样式、背景色、内边距及文字颜色，也就是为表格单元格设置了 CSS 样式，通过这样的设置，突出了表格主要的信息内容，预览效果如图 12-15 所示。

从图 12-15 可以看出，表头和表格内容清晰可见，并且能够明显区分开来，使使用者一目了然。

（8）制作隔行的 CSS 代码，代码如下。

```
td.alt {
    background: #F5FAFA;                                     /*设置背景颜色*/
    color: #797268;                                          /*设置文字颜色*/
}
```

在以上代码中，选择符为 td.alt 的单元格重新设置了背景色和文字颜色，制作出表格隔行换色的醒目效果，预览如图 12-16 所示。

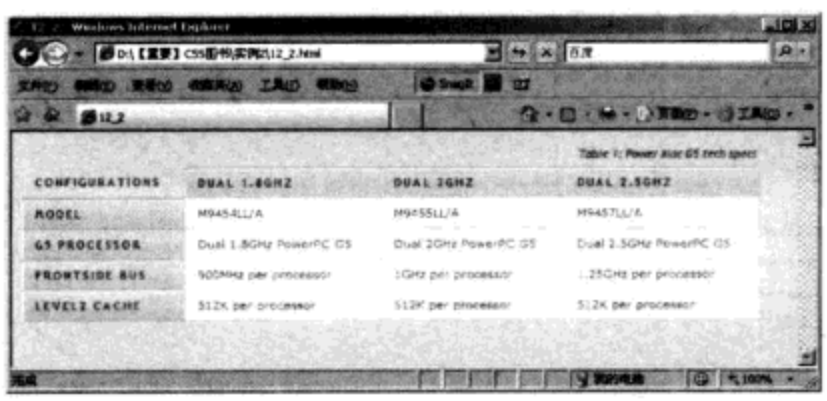


图 12-15　添加 td 样式的预览效果

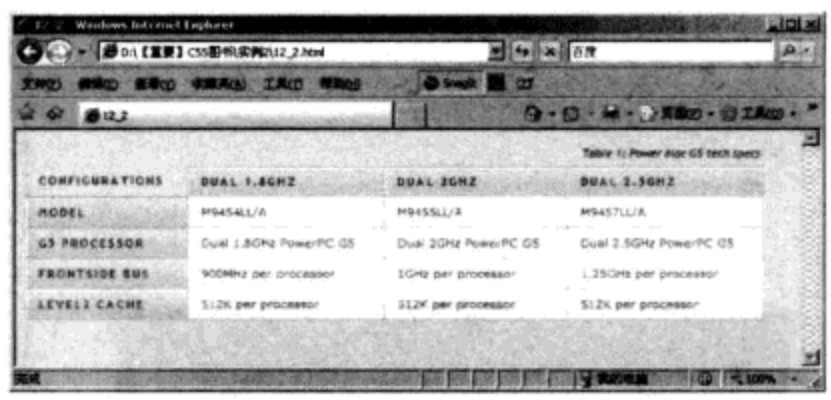


图 12-16　添加隔行换色样式的预览效果

（9）制作表头特殊样式的 CSS 代码，代码如下。

```
th.spec {
    border-left: 1px solid #C1DAD7;                          /*设置左边框样式*/
    border-top: 0;                                           /*设置上边框*/
    background: #fff url(images/12_2.gif) no-repeat;         /*设置背景样式*/
    font: bold 10px "Trebuchet MS";                          /*设置字体样式*/
}
```

在以上代码中，使用背景图片为选择符为 th.spec 的表头设置了样式。

这样按照 XHTML 代码，为“MODEL”和“FRONTSIDE BUS”表头设置了这个背景样式，预览效果如图 12-17 所示。

图 12-17　添加了背景图的表头的预览效果

继续添加样式。

```
th.specalt {
    border-left: 1px solid #C1DAD7;                          /*设置左边框样式*/
    border-top: 0;                                           /*设置上边框*/
    background: #f5fafa url(images/13_3.gif) no-repeat;      /*设置背景样式*/
    font: bold 10px "Trebuchet MS";                          /*设置字体样式*/
    color: #797268;                                          /*设置文字颜色*/
}
```

在以上代码中，同样使用背景图片为选择符为 th.specalt 的表头设置了样式，背景图片如图 12-18 所示。

按照 XHTML 代码，为“G5 PROCESSER”和“LEVEL2 CACHE”表头设置了这个背景样式，预览效果如图 12-19 所示。

图 12-18 背景图片

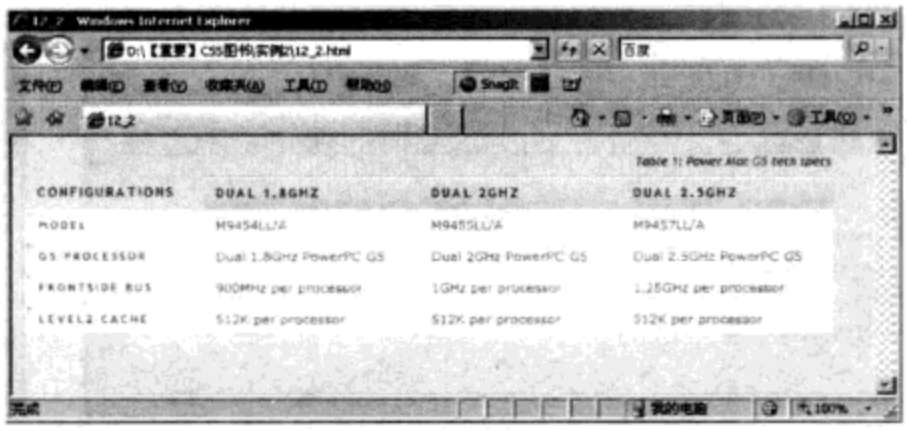


图 12-19 添加背景样式的表头的预览效果

这样，整个 CSS 风格的英文数据表格就制作完成了。

12.3 小结

本章介绍了使用 CSS 属性设计数据表格的方法，主要涉及 5 个 XHTML 专用表格标签。

- caption 标签。
- tbody 标签。
- thead 标签。
- tfoot 标签。
- th 标签。

还涉及一个 CSS 属性。

- text-transform 大小写字母转换属性。

对本章的知识点前后联系进行归纳总结，结构导图如图 12-20 所示。

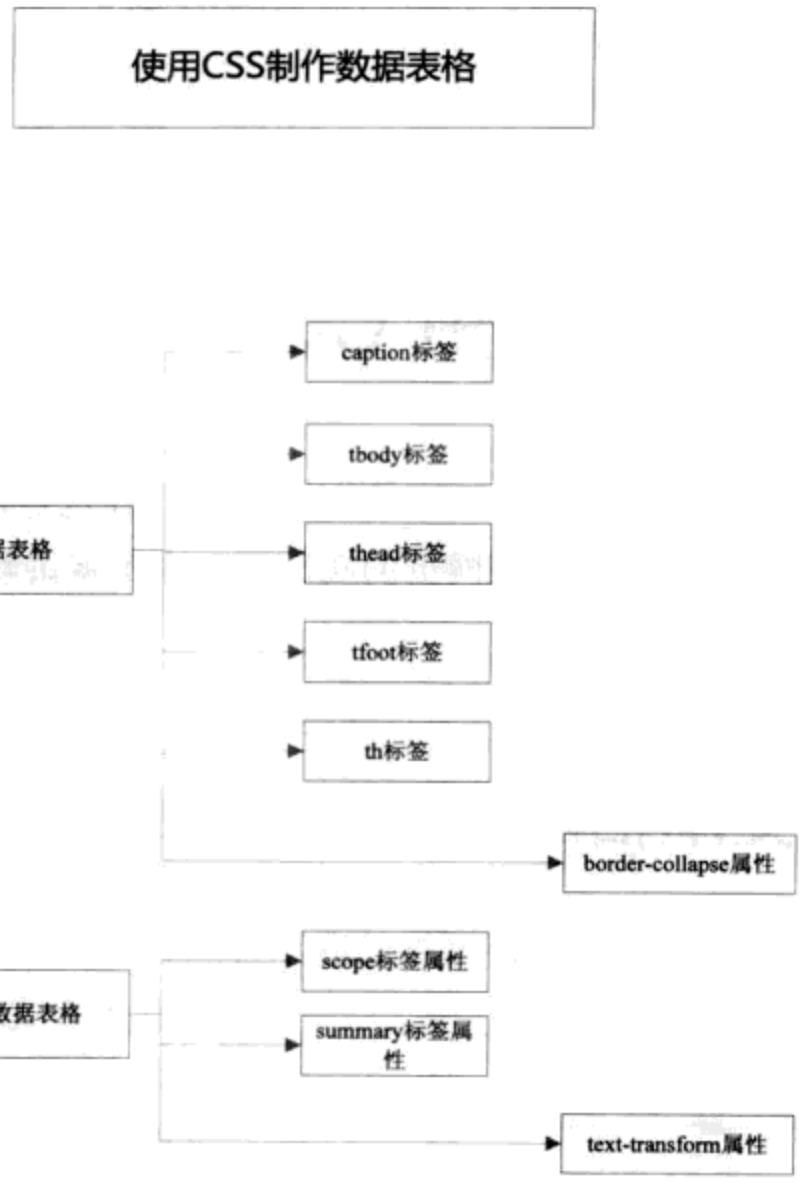


图 12-20　本章知识点结构导图

12.4　习题

1．简述表示数据的表格对象标签有哪些。

2．使用表格标签制作一个简单的表格。

3．简述 CSS 属性 text-transform 的用法。

4．使用 XHTML 表格标签属性以及 CSS 属性 text-transform 来制作一个具有 CSS 样式风格的 table 表格。

第13章 使用CSS制作页面底部内容

写文章都要讲究个虎头豹尾，这个理念对于网页设计也是可以应用的。一些网站的版权信息、联系方式、备案信息等都是放在网页的底部的。本章将讲解使用CSS设计网页的页面底部内容。

13.1 制作文本信息的页面底部内容

页面底部包含的文本信息一般有网站的联系方式、版权信息等重要内容。本节将制作一个包含了联系方式和版权信息的网页的底部，这些信息都是文本信息，部分内容也使用超级链接。

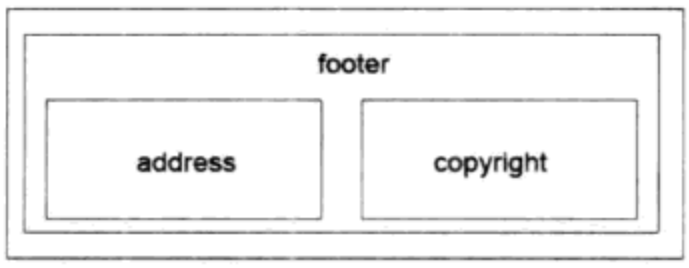


图 13-1 制作思路结构图

在制作过程中，使用到了CSS样式中a元素的一个标签属性title，当光标悬停到链接文字时，出现提示文字。实例制作思路的结构图如图13-1所示。

13.1.1 a标签的title属性

a元素的属性很多，在设定一个超级链接时有很多参数可供选择，以实现不同的链接效果，titel 属性便是其中的一个很重要的属性，其语法结构如下。

```
<a hreff="#" title="提示信息文字"></a>
```

title 属性主要用于设定链接点被选到时显示的标题，其用法如图13-2所示。

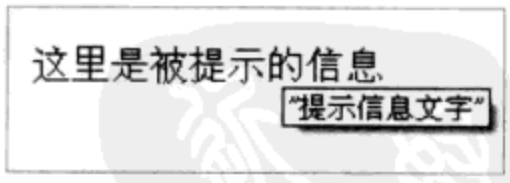


图 13-2 title 属性的用法

13.1.2 制作包含文本信息的页面底部XHTML结构代码

本实例制作的网页底部包含两个部分：一个是联系方式，另一个是版权信息。将底部区域分成两栏，分别放置这两样文本信息。制作页面的XHTML代码如下。

```
<div id="footer">
    <div id="footer-features">
```

```
            <div class="footer-feature1">
                <h3><a href="#" title="联系方式">ADDRESS</a></h3>
                <p><a href="#" title="理想国际大厦">BEIJING IDEAL PLAZA</a> , No. 58 Northwest 4th
Ring Road, Haidian, Beijing 100080, China <br />
                Code:100080<br />
                 [+86 10] 6234 3447<br />
                <a href="#" title="更多内容">more...</a></p>
            </div>
            <div class="footer-feature2">
                <h3><a href="#" title="版权信息">COPYRIGHTS</a></h3>
                <p> Copyright &copy;1996-2008 <a href="#" title="理想工作室">Ideal Studio
Corporation</a>, All Rights Reserved <br />
                <a href="#" title="更多内容">more...</a></p>
            </div>
        </div>
    </div>
```

在以上代码中，使用#footer 作为主容器，用来控制背景的宽度和高度。里面使用了一个 div #footer-features 放置联系方式和版权信息的内容，联系方式使用一个 div，版权信息使用另外一个 div，两者是并列关系，页面结构预览效果如图 13-3 所示。

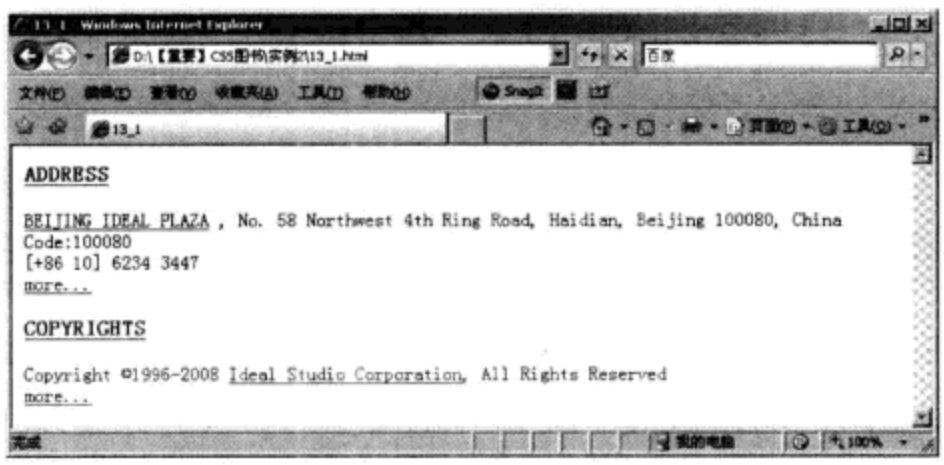


图 13-3　页面结构预览效果

13.1.3　制作页面的 CSS 样式

完成了 XHTML 代码结构，然后就可以为页面代码添加 CSS 样式。

（1）制作#footer 元素的 CSS 样式，代码如下。

```
#footer {
    font-family: arial;                                    /*设置字体类型*/
    height:155px;                                          /*设置高度*/
    border-top:1px solid #255980;                          /*设置上边框样式*/
    background: url(images/13_1.jpg) no-repeat;            /*设置背景样式*/
    width: 750px;                                          /*设置宽度*/
}
```

在以上代码中，首先设置了页面底部区域的宽度和高度以及上边框，为网页底部指定了区域，并且从设计角度来看，上边框分隔了上面和下面部分，指明了接下来为网页的结尾。然后，

使用了一个背景图片，不平铺，底部的宽高保证了背景图片的完整显示，背景图片如图 13-4 所示。

图 13-4　背景图片

这样，添加了#footer 样式后预览效果如图 13-5 所示。

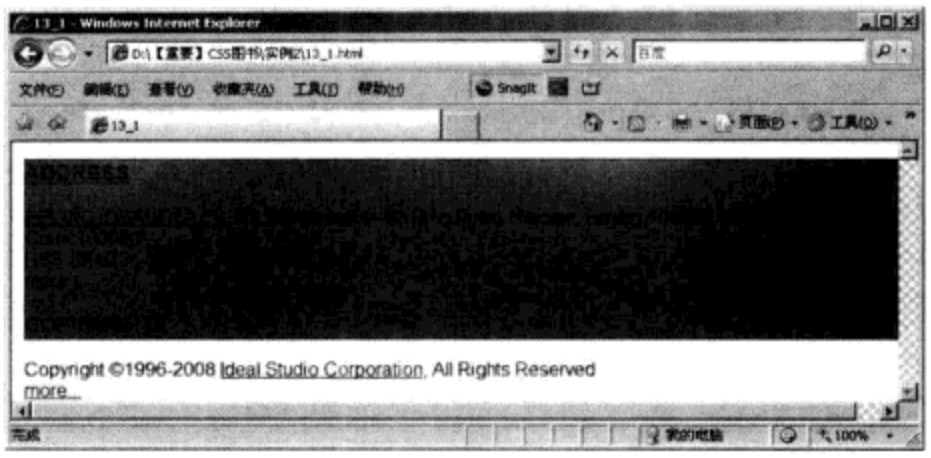


图 13-5　添加#footer 样式的预览效果

从图 13-5 可以看出，只是单纯地使用了背景图片，文本布局和文字颜色仍然不合理，继续添加样式。

（2）制作#footer-features 元素的 CSS 样式，代码如下。

```
#footer-features {
    margin:0 auto;                          /*设置外边距，实现对象的居中*/
    padding-top:15px;                       /*设置上内边距*/
}
```

为了实现布局，在以上代码中，设置#footer-features 元素居中，并且上边距为 15px，使得文字部分和周围有一定空间，预览效果如图 13-6 所示。

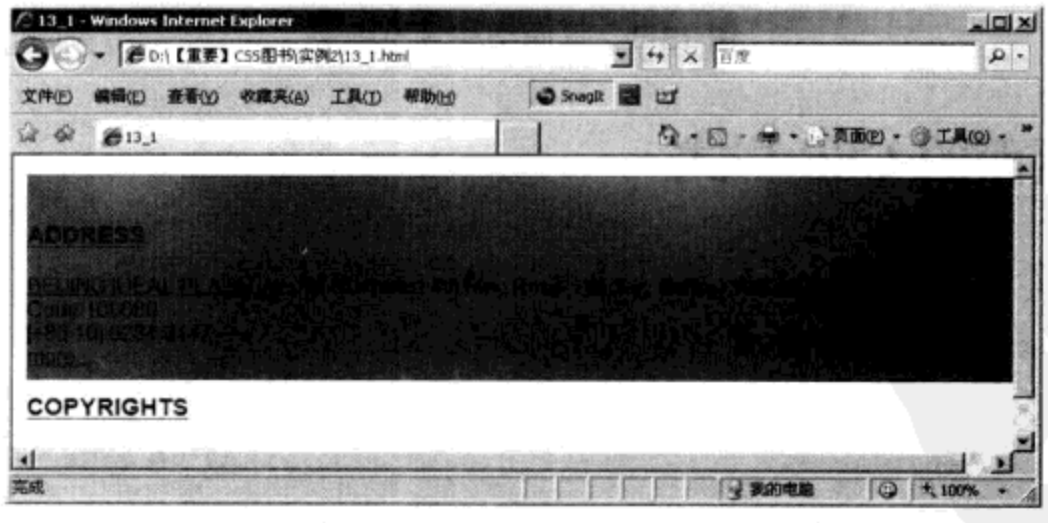


图 13-6　添加#footer-features 样式的预览效果

（3）制作#news 元素的 CSS 样式，代码如下。

```
.footer-feature1 {
    width:400px;                            /*设置宽度*/
    float:left;                             /*设置向左浮动定位*/
    border-right:1px solid #4390BC;         /*设置右边框样式*/
```

```
    margin-right:15px;                                    /*设置右外边距*/
    margin-left:18px;                                     /*设置左外边距*/
}
.footer-feature2 {
    width:250px;                                          /*设置宽度*/
    float:left;                                           /*设置向左浮动定位*/
    margin-right:15px;                                    /*设置右外边距*/
}
```

每一个#footer-feature 元素都定义了宽度，宽度总和不超过底部总宽度，并且都使用 float: left 向左浮动并列排列，设置外边距控制相邻元素之间的间距效果。为第一个#footer-feature1 元素定义了右边框，制作一种分隔线效果，预览效果如图 13-7 所示。

从图 13-7 可以看出，现在完成了文本信息的左右分栏布局，但是字体样式仍然不尽如人意，继续添加样式。

（4）制作文本信息元素的 CSS 样式，代码如下。

```
#footer p {
    font-size:12px;                                       /*设置字号*/
    color:#fff;                                           /*设置文字颜色*/
    padding-right:15px;                                   /*设置右内边距*/
    line-height:1.5em;                                    /*设置行高*/
}
#footer .footer-feature2 p {
    padding-right:0;                                      /*设置右内边距*/
}
```

为了排版美观，在以上代码中，通过包含选择符，指定了实例中的文本信息的字号、颜色以及行高，通过在包含选择符两次设定右内边距，左栏的文本距离中间分隔线有了一定的空间，预览效果如图 13-8 所示。

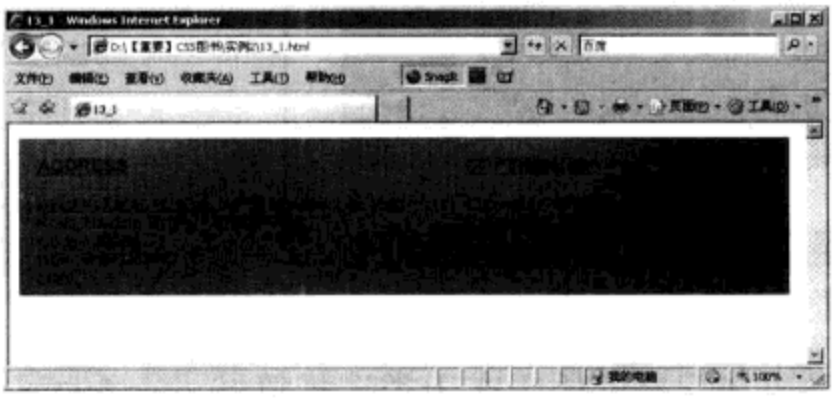


图 13-7　为文本布局设置样式的预览效果

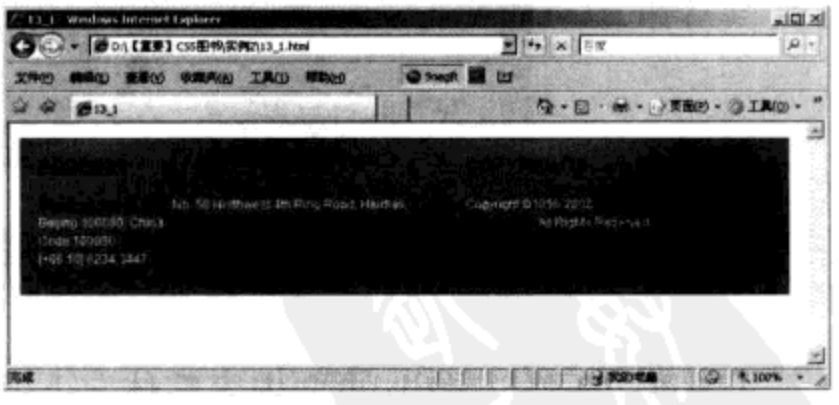


图 13-8　对文本细节添加样式的预览效果

（5）制作链接元素的 CSS 样式，代码如下。

```
#footer a {
    color:#84C9D0;                                        /*设置链接文字颜色*/
    text-decoration:none;                                 /*设置链接文字文本样式*/
    font-weight:bold;                                     /*设置为链接文字为粗体*/
}
#footer a:hover {
```

```
        text-decoration:underline;                    /*设置链接文字光标经过后的文本样式*/
    }
```

在以上代码中，为链接文字定义了样式。这样带有文字信息的页面底部就制作完成，预览效果如图 13-9 所示。

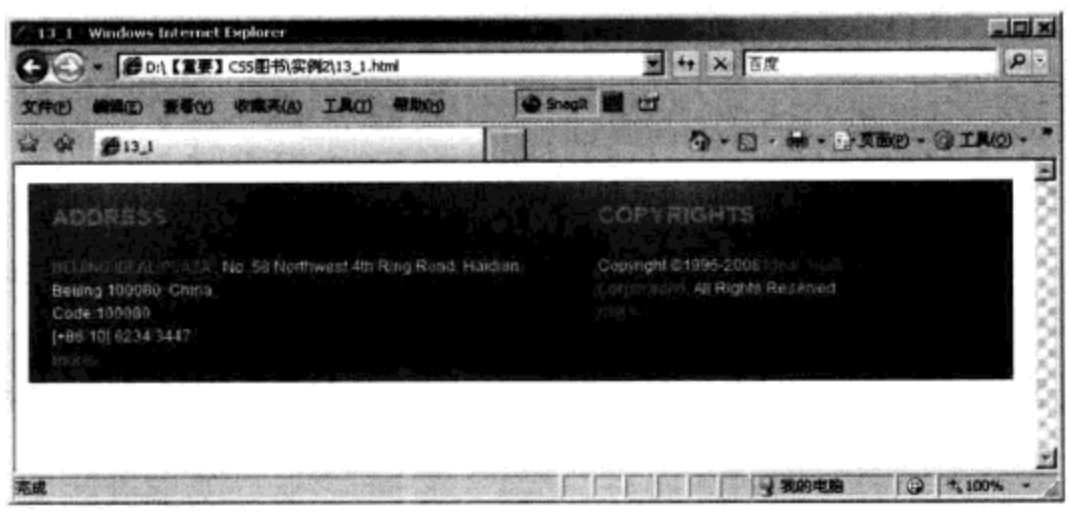


图 13-9　带有文字信息的页面底部

13.1.4　兼容问题

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。

1．Firefox 浏览器测试

把实例所用到的网页文件放在 Firefox 浏览器中打开，预览效果如图 13-10 所示。

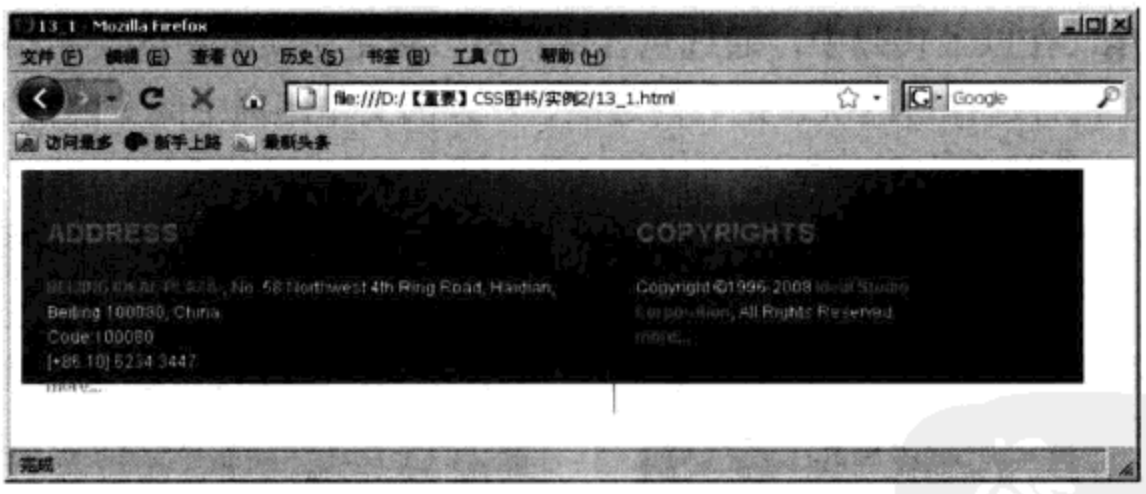


图 13-10　Firefox 浏览器中的预览效果

从图 13-10 可以看出，页面内容整体向下偏移，出现问题的原因在于类选择符#footer-features 设置了上边距。修改上边距，代码如下。

```
#footer-features {
    margin:0 auto;                                    /*设置外边距，使得居中*/
    padding-top:0px;                                  /*设置上内边距*/
    *padding-top: 15px;
    *padding-top: 15px !important;
}
```

在以上代码中，使用 CSS hack 方法，对上内边距，重新编写了针对 3 种浏览器兼容的样式代码，第一个 padding-top，3 个浏览器都能够识别；第二个 padding-top，IE 6、IE 7 能够识别，这样实现了只针对 IE 浏览器做样式设置；第三个 padding-top，只有 IE 7 能够识别，这样就实现了针对 3 中浏览器的兼容问题的解决，最终在 Firefox 浏览器中的预览效果如图 13-11 所示。

2. IE 6 浏览器测试

通过刚刚介绍的方法，解决在 IE 6 浏览器中的兼容问题，预览效果如图 13-12 所示。

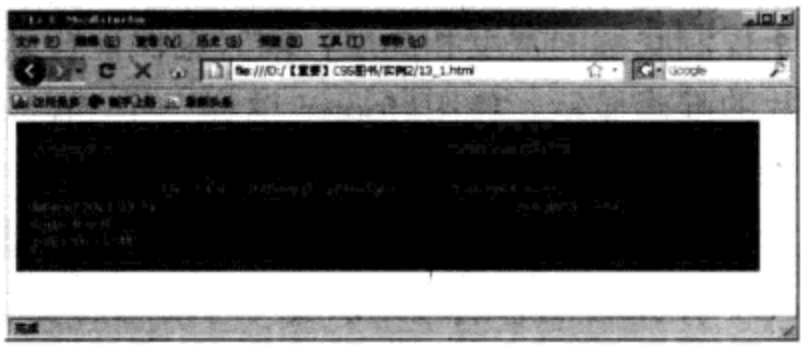


图 13-11　解决兼容问题后 Firefox 浏览器中的预览效果

图 13-12　解决兼容问题后 IE 6 浏览器中的预览效果

13.2　制作包含导航链接的页面底部内容

有的时候，在网页的底部信息中，也会使用导航系统。网页底部的导航系统，不需要华丽的表现，一般选择使用简单的文字链接即可。本节制作包含文字链接导航的页面底部实例，实例制作思路的结构图如图 13-13 所示。

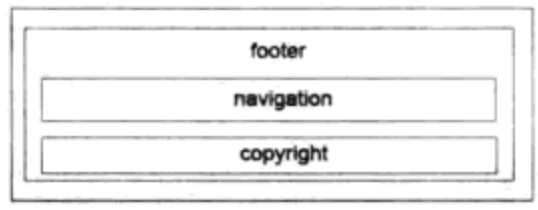


图 13-13　制作思路结构图

13.2.1　制作包含导航链接页面底部的 XHTML 代码结构

制作页面的 XHTML 代码。

```
<div id="layout">
    <div id="footer">
        <div id="block-menu">
            <div class="clear">
                <ul>
                    <li><a href="#" title="About FreshBrain">About</a></li>
                    <li><a href="#" title="">Blogs</a></li>
                    <li><a href="#" title="Copyright &amp; IP">Copyright &amp; IP</a></li>
                    <li><a href="#" title="Privacy Policy">Privacy</a></li>
                    <li><a href="#" title="Terms &amp; Conditions">Terms</a></li>
                    <li><a href="#" title="Frequently Asked Questions">FAQ</a></li>
                    <li class="last"><a href="#" title="Contact Form">Contact</a></li>
                </ul>
```

```
            </div>
        </div>
        <div id="copyright">
            <div class="content ">Copyright &copy; 2008 CSS Book. All rights reserved. </div>
        </div>
    </div>
</div>
```

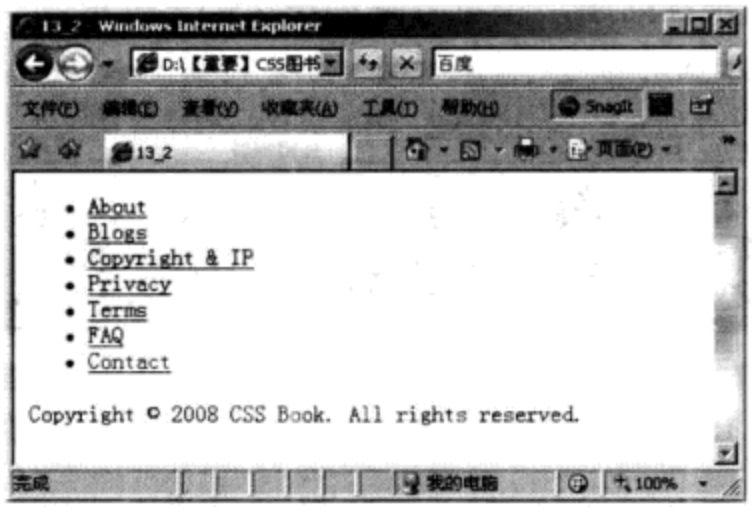


图 13-14　页面结构预览效果

在以上代码中，以#layout 作为主容器，用于控制底部的宽度和高度，同时也保证背景图片的准确显示。使用#footer 这样一个 div，用于控制底部内容区域的显示位置。

在#footer 元素下边，使用了两层嵌套的 div，用来控制总的边距以及内容信息的显示。在#block-menu 元素下，包含了两个 div，分别是导航系统和版权信息，页面结构预览效果如图 13-14 所示。

13.2.2　制作页面的 CSS 样式

完成了 XHTML 代码结构后，就可以为页面代码添加 CSS 样式。

（1）制作整体框架的 CSS 样式，代码如下。

```
body{
    background-color: #000000;                          /*设置背景颜色*/
}
#layout {
    margin:0 auto;                                      /*设置外边距，实现对象居中*/
    width:980px;                                        /*设置宽度*/
    font-family: arial;                                 /*设置字体类型*/
}
#footer {
    padding:40px 0 0 0;                                 /*设置内边距*/
    height:130px;                                       /*设置高度*/
    font-size:12px;                                     /*设置字号*/
    background:url(images/13_4.jpg) no-repeat 1px 0;    /*设置背景样式*/
}
```

在以上代码中，首先在标签指定选择符 body 中定义了网页的背景颜色为黑色。然后，为主容器#layout 指定了宽度，确保背景图片能够完整显示。最后，控制#footer 元素的样式，为了实现布局美观，指定了上内边距，定义了背景图片，背景图片如图 13-15 所示。

图 13-15　背景图片

这时候，页面预览效果如图 13-16 所示。

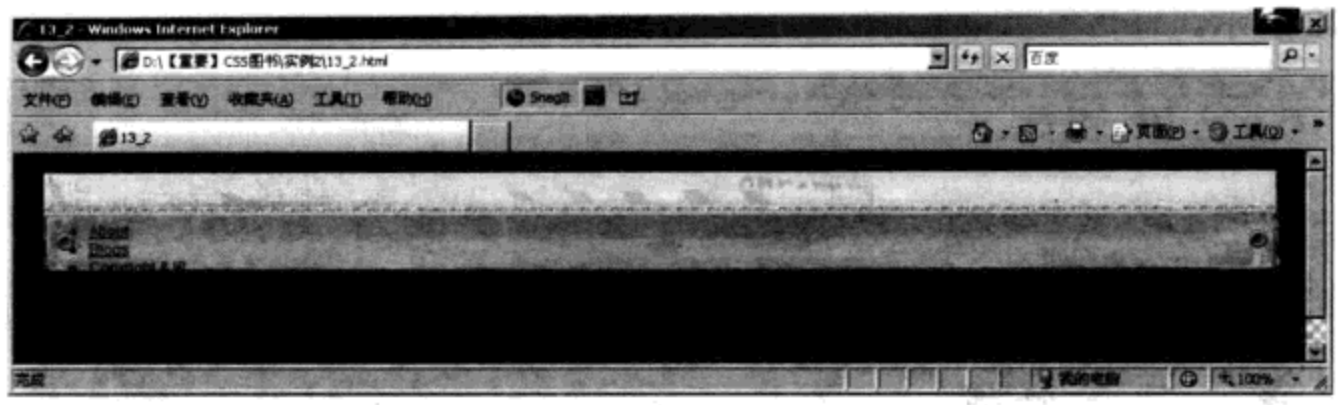


图 13-16　添加了背景样式的预览效果

从图 13-6 可以看出，虽然添加了背景图片使页面美观，但是链接的排列方式没有改变，链接文字样式也不符合页面整体风格，继续添加样式。

（2）制作.clear 的 CSS 样式，代码如下。

```
.clear {
    display:inline-block;                                    /*设置块级元素显示方式*/
}
```

在以上代码中，使用到了 display 属性在这里使用到的属性值为 inline-block，它表示的是此元素显示为块级元素，并且该元素前后没有换行符，是 CSS 2.1 新增的属性值。这样就能使元素可以像 block 属性一样具有块状元素的特点，又能具有 inline 属性的内联显示的特点。

（3）制作#block-menu 的 CSS 样式，代码如下。

```
#block-menu {
    padding:10px 0 0 36px;                                   /*设置内边距*/
}
```

以上代码用来控制内容区域的内边距，使其与边界有一定的空间，排版美观，预览效果如图 13-17 所示。

（4）制作导航系统的 CSS 样式，代码如下。

```
#block-menu ul {
    margin:0;                                                /*设置外边距*/
    padding:0;                                               /*设置内边距*/
}
#block-menu ul li {
    float:left;                                              /*设置向左浮动定位*/
    padding:0 12px;                                          /*设置内边距*/
    list-style:none;                                         /*设置列表符号样式*/
    border-right:2px solid #9e998c;                          /*设置右边框样式*/
}
```

在以上代码中，使用 ul 列表元素制作页面底部导航系统的样式。首先，取消了 ul 元素默认可能存在的外边距和内边距。

然后，继续定义 li 元素，前面已经讲过的导航系统的控制方法，左浮动，并且取消列表样

式。为了使用分隔线将每个导航项分隔开，为 li 元素添加了右边框线以及左右内边距，预览效果如图 13-18 所示。

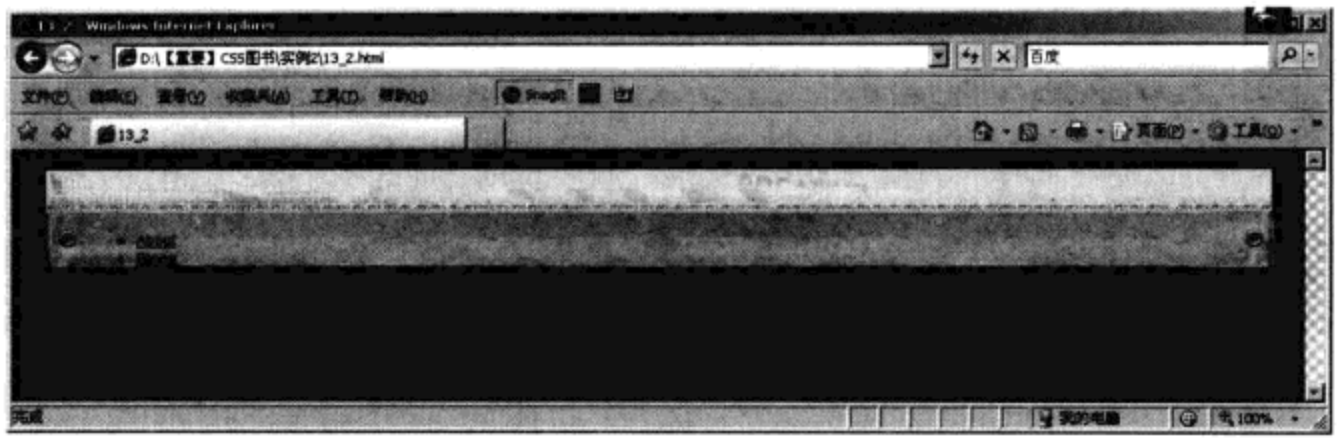


图 13-17　添加了#block-menu 样式的预览效果

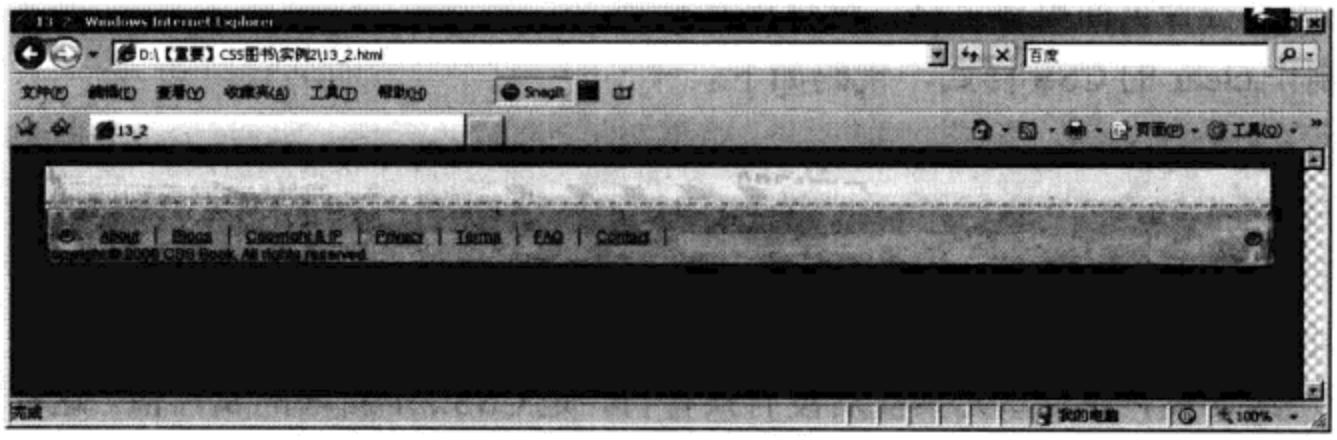


图 13-18　改变了超级链接排列方式的预览效果

（5）制作超级链接文字的 CSS 代码，代码如下。

```
#block-menu ul li a {
    font-weight:bold;                                    /*设置链接文字为粗体*/
    color:#61503B;                                       /*设置文字颜色*/
}
```

以上代码为导航系统定义文字链接，使用包含选择符#block-menu ul li a 设置了链接文字的字体粗细以及文字颜色。这里只是简单地设置了文字链接状态时的样式，其他样式并没有特别指定，预览效果如图 13-19 所示。

（6）制作“contact”超级链接的 CSS 代码，代码如下。

```
#block-menu ul li.last {
    border:0;                                            /*设置边框*/
}
#block-menu ul li.last a {
    padding:0px;                                         /*设置内边距*/
}
```

在以上代码中，由于之前所定义的导航项都具有了右边框效果，为了布局美观，最后一个导航项的右边框应该取消，所以特别地为最后一个导航项定义了链接样式，重新设置了边框为 0px，也就是取消了边框，预览效果如图 13-20 所示。

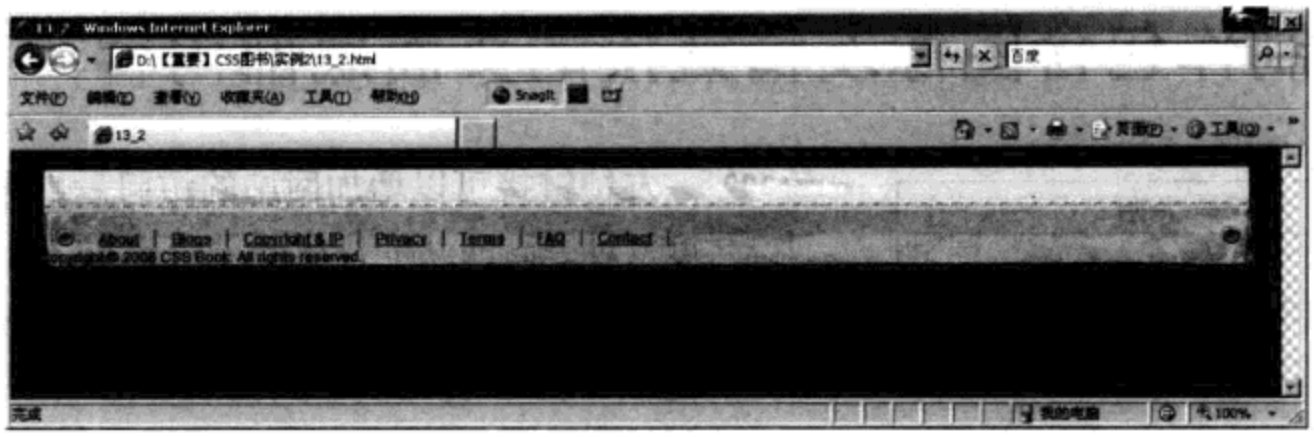


图 13-19　添加超级链接样式的预览效果

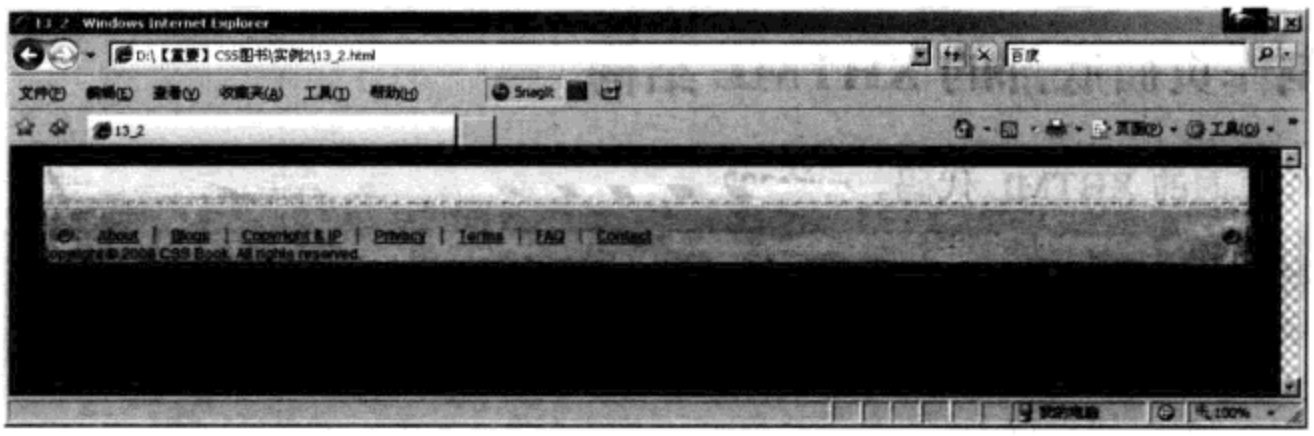


图 13-20　添加“contact”超级链接样式的预览效果

（7）制作版权信息的 CSS 样式，代码如下。

```
#copyright {
    padding:25px 20px;                                    /*设置内边距*/
}
#copyright div.content {
    font-size:11px;                                       /*设置字号*/
    color:#aaa;                                           /*设置文字颜色*/
}
```

在以上代码中，首先在#copyright 元素中，设置了该区域的内边距。然后，为版权信息部分的文本设置了字号、颜色。

这样包含文字链接的导航系统的页面底部就制作完成，预览效果如图 13-21 所示。

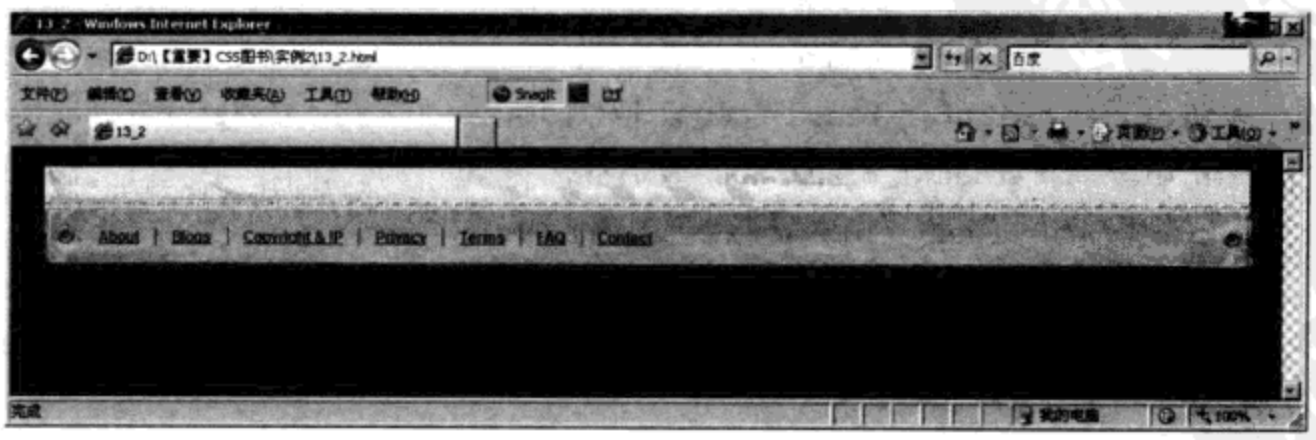


图 13-21　包含文本链接导航的页面底部预览效果

13.3　制作包含图像友情链接的页面底部内容

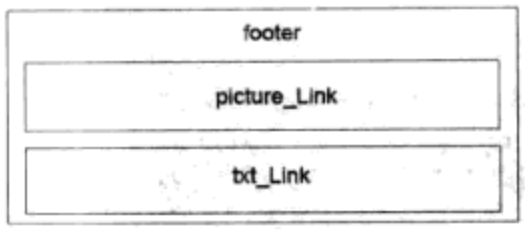


图 13-22　制作思路结构图

友情链接可以增加网站的访问量，提高网站的知名度，对提高网站的排名起着很重要的作用。本节将制作一个包含有友情链接内容的底部，包括标志图片的链接和文字的链接。实例制作思路的结构图如图 13-22 所示。

13.3.1　制作页面底部的 XHTML 结构

首先制作页面的 XHTML 代码。

```
<div class="layout"> <img src="images/13_5.gif" width="950" height="54" />
    <div class="picLink">
        <a href="#" target="_blank"><img src="images/13_2.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_3.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_4.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_11.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_2.jpg" /></a><br />
        <a href="#" target="_blank"><img src="images/13_3.jpg" /></a><br />
        <a href="#" target="_blank"><img src="images/13_4.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_2.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_3.jpg"/></a>
        <a href="#" target="_blank"><img src="images/13_4.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_11.jpg" /></a>
        <a href="#" target="_blank"><img src="images/13_2.jpg" /></a><br />
    </div>
    <div class="txtLink"> ---- 合作链接 ----<br />
        <a href="#" target="_blank">CSS 设计网站 01</a> | <a href="#" target="_blank">CSS 设
计网站 02</a> | <a href="#" target="_blank">CSS 设计网站 03</a> | <a href="#" target="_blank">CSS 设
计网站 04</a> | <a href="#" target="_blank">CSS 设计网站 05</a> | <a href="#" target="_blank">CSS 设
计网站 06</a> | <a href="#" target="_blank">CSS 设计网站 07</a> | <a href="#" target="_blank">CSS 设
计网站 08</a> | <a href="#" target="_blank">CSS 设计网站 09</a><br />
        <a href="#" target="_blank">CSS 设计网站 10</a> | <a href="#" target="_blank">CSS 设
计网站 11</a> | <a href="#" target="_blank">CSS 设计网站 12</a> | <a href="#" target="_blank">CSS 设
计网站 13</a> </div>
</div>
```

在以上代码中，还是以#layout 作为总容器，里面放置标题图片、图片形式的友情链接和文字形式的友情链接共 3 个子容器，XHTML 结构预览效果如图 13-23 所示。

图 13-23　XHTML 结构预览效果

从图 13-23 可以看出，这时候图片超级链接和文字超级链接都是默认的样式，图片有默认边框等。

13.3.2　制作页面的 CSS 样式

完成了 XHTML 代码结构后，就可以为页面代码添加 CSS 样式。

（1）制作#layout 的 CSS 样式，代码如下。

```
.layout {
    margin-top:6px;                                    /*设置上外边距*/
    width:950px;                                       /*设置宽度*/
    font-size: 12px;                                   /*设置字号*/
}
```

在以上代码中，#layout 主容器指定了宽度，使页面底部与上边内容有一定空间，并定义了上边距，预览效果如图 13-24 所示。

图 13-24　添加#layout 样式的预览效果

（2）制作图片区域的 CSS 样式，代码如下。

```
.layout .picLink {
    background:url(images/13_5.gif) repeat-y;          /*设置背景样式*/
    text-align:center;                                 /*设置文本居中*/
}
```

首先为了使布局美观，为图片链接部分设置了背景图片，纵向平铺，同时设置了内部的对象为居中对齐显示。页面预览效果如图 13-25 所示。

图 13-25　添加图片区域样式的预览效果

（3）制作链接图片的 CSS 代码，代码如下。

```
.layout .picLink img {
    border:1px solid #000;                          /*设置边框样式*/
    margin:5px 10px 0 10px;                         /*设置外边距*/
}
```

在以上代码中，为需要做链接的图片设置了边框，并且为了图片之间有间隔，对每个图片设置了外边距，预览效果如图 13-26 所示。

图 13-26　添加图片链接样式的预览效果

（4）制作版权信息的 CSS 样式，代码如下。

```
.layout .txtLink {
    background:url(images/13_16.gif) no-repeat #f7ebdb;        /*设置背景样式*/
    text-align:center;                                         /*设置文本居中*/
    padding:25px 0;                                            /*设置内边距*/
    height:50px;                                               /*设置高度*/
}
```

在以上代码中，对文字链接部分的样式做了定义。为了使背景美观，上下两部分自然过渡，浑然一体，为文字链接部分添加了一个背景图片，不平铺，同时又设置了背景颜色，背景图片覆盖在背景之上。

同样的，内部内容居中显示，并且为了与上面图片链接部分空出一些空间，定义了上下内边距，预览效果如图 13-27 所示。

图 13-27　添加版权信息的样式预览效果

（5）制作版权信息的 CSS 样式，代码如下。

```
.layout .txtLink a{
    text-decoration: underline;                    /*设置链接文本的下划线样式*/
    color: #333333;                                /*设置文字颜色*/
    line-height: 20px;                             /*设置行高*/
}
.layout .txtLink a:hover{
    text-decoration: underline;                    /*设置链接文本光标经过时的下划线样式*/
    color: #FF0000;                                /*设置光标经过时的文本颜色*/
}
```

以上代码设置了文字链接的样式。这样包含有图片形式的友情链接的网页底部就制作完成，预览效果如图 13-28 所示。

图 13-28　图片形式的友情链接的网页底部预览效果

13.4　小结

本章介绍了使用 CSS 制作页面底部的实例，包括了制作文本信息的页面底部、制作导航链

接的页面底部内容、制作包含图像友情链接的页面底部的实例。

对本章的知识点前后联系进行归纳总结，结构导图如图 13-29 所示。

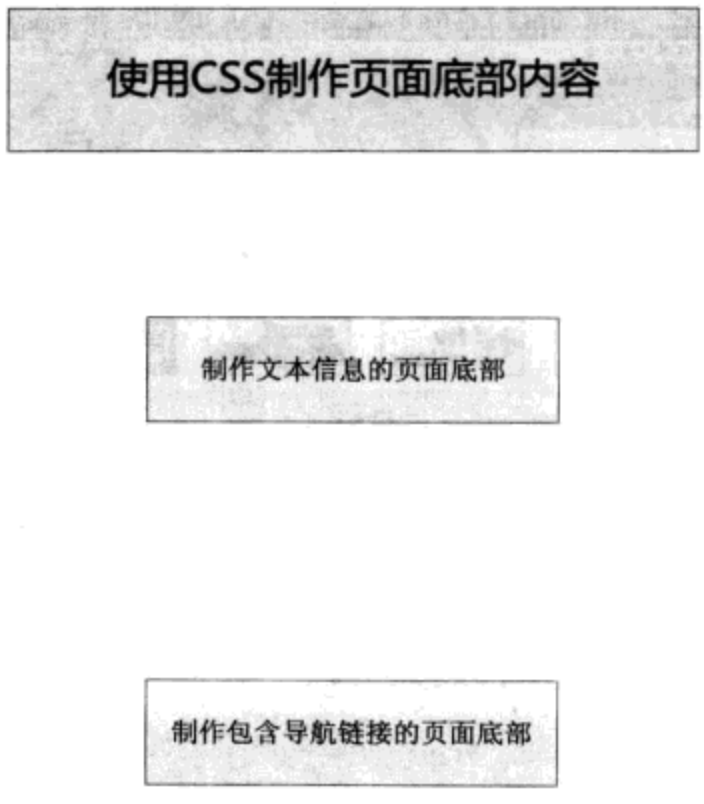
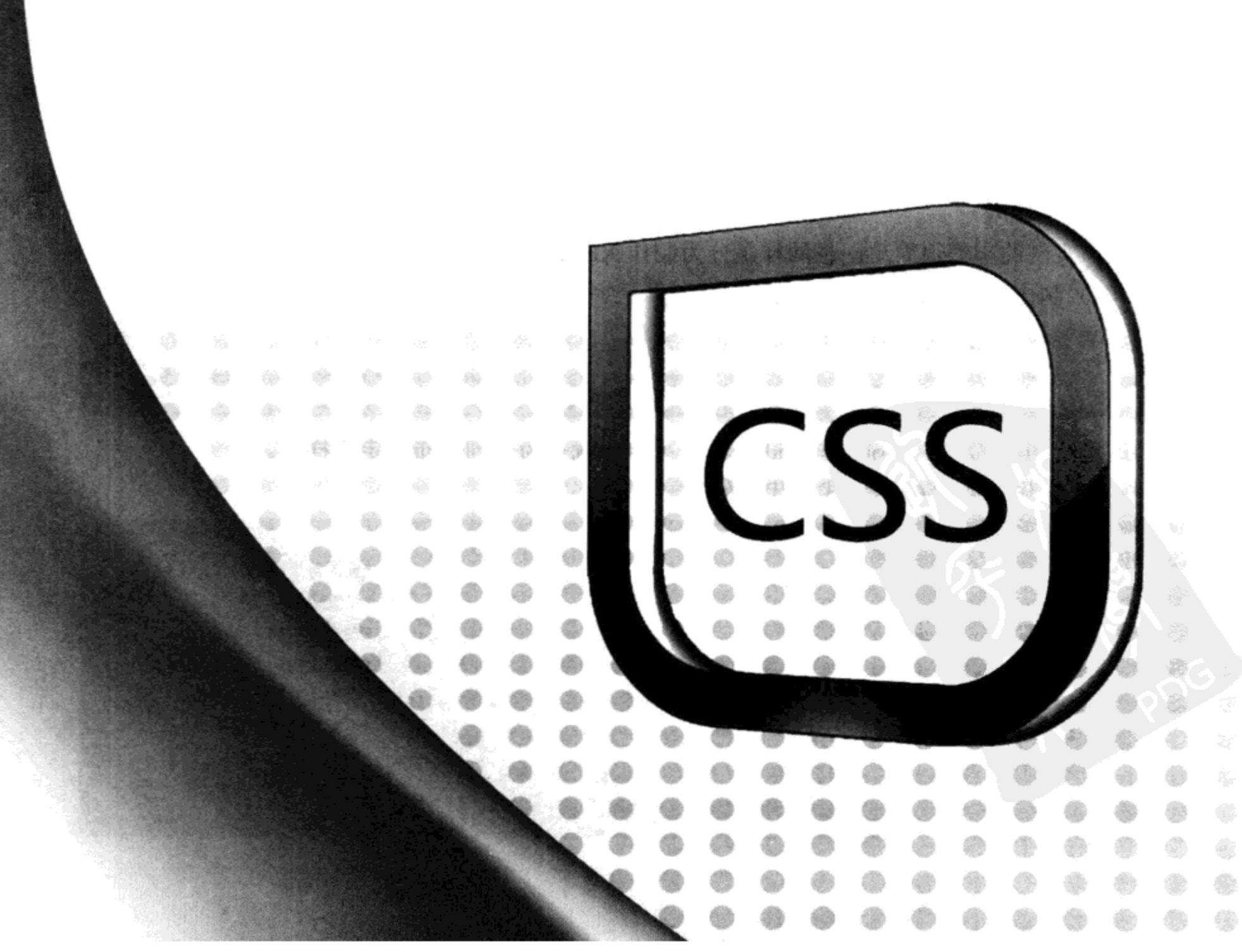


图 13-29　本章知识点结构导图

13.5　习题

1. 使用 CSS 制作页面底部内容主要包括哪些类型？
2. a 标签的 title 属性的作用是什么？
3. 利用本章所学知识制作一个包含文本信息、导航链接以及图像友情链接的页面底部。

第三篇　整体布局篇



第14章 CSS基本布局

CSS+DIV 是 Web 标准中一种新的布局方式，它正逐渐代替传统的表格（table）布局。CSS+DIV 模式具有比表格更大的优势，结构与表现相分离，代码简洁，利于搜索，方便后期维护和修改。本章主要讲解 CSS+DIV 的基本布局种类，同时还在每种类型的布局后面，列举一个实例形象说明布局样式，以帮助读者快速掌握 CSS+DIV 布局的应用方法。

14.1 一列固定宽度

一列式布局是所有布局的基础，也是最简单的布局形式。本节将要讲到的一列式布局是一种固定宽度的布局样式，XHTML 代码相当简单，只需要编写一段 div 即可。

1. 实现原理

div 默认状态下占据整行显示，当为其设置宽度的时候，div 的宽度就会变成所设置的宽度，以实现固定宽度的布局。

2. 制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    一列固定宽度布局
</div>
```

这里给 div 使用了 layout 作为 id 名称，下一步就是为一列式布局定义样式。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    background-color: #FFFF00;                    /*设置背景颜色*/
    border: 1px solid #000000;                    /*设置边框样式*/
    text-align: center;                           /*设置文本居中对齐*/
    padding-top: 40px;                            /*设置上内边距*/
    font-size: 14px;                              /*设置字号*/
    font-family: "宋体";                          /*设置字体类型*/
    font-weight: bold;                            /*设置为粗体*/
}
#layout{
```

```
    height: 300px;                                    /*设置高度*/
    width: 400px;                                     /*设置宽度*/
}
```

在以上代码中，为了便于讲解，首先使用标签选择符 div 定义了一套公共样式属性，用于接下来的每一节当中。使用了 background-color 将 div 设定为黄色背景，并使用 border 属性将 div 设置成了黑色的 1px 的边框。同样，为了布局美观，将 div 内部的字体设定了字号、居中显示以及上内边距，字体与边框上边界有一定空间。

然后，由于是宽度固定的布局，所以直接为#layout 元素设置了宽度属性 400px 与高度属性 300px。在前面讲过，div 默认状态下，宽度将占据整行的空间，因此当设置了 width: 400px 之后，当前的 div 宽度将变为设置的宽度，这样便形成了一列式的固定宽度布局，也是最简单的布局形式。

在接下来讲解 CSS 布局样式的时候都将使用这样的代码。这样一列固定宽度的布局就实现了，预览效果如图 14-1 所示。

很多网站都是在一列固定宽度基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的。如图 14-2 所示的网站是首先在一列固定宽度的布局基础上，再进行细致划分布局。

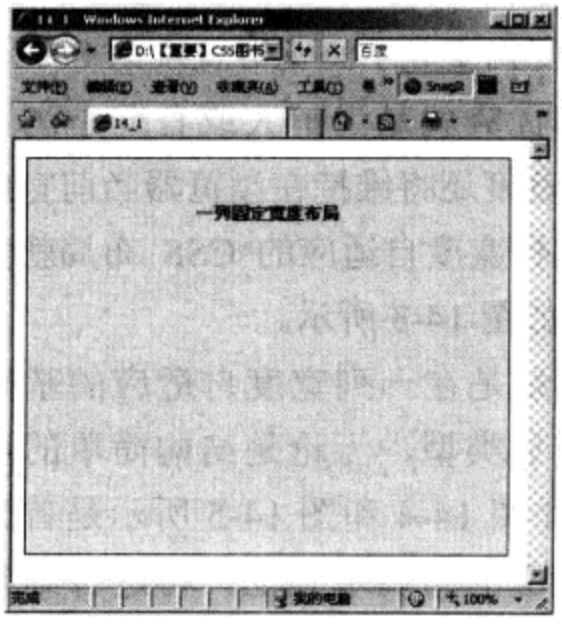


图 14-1　一列固定宽度布局

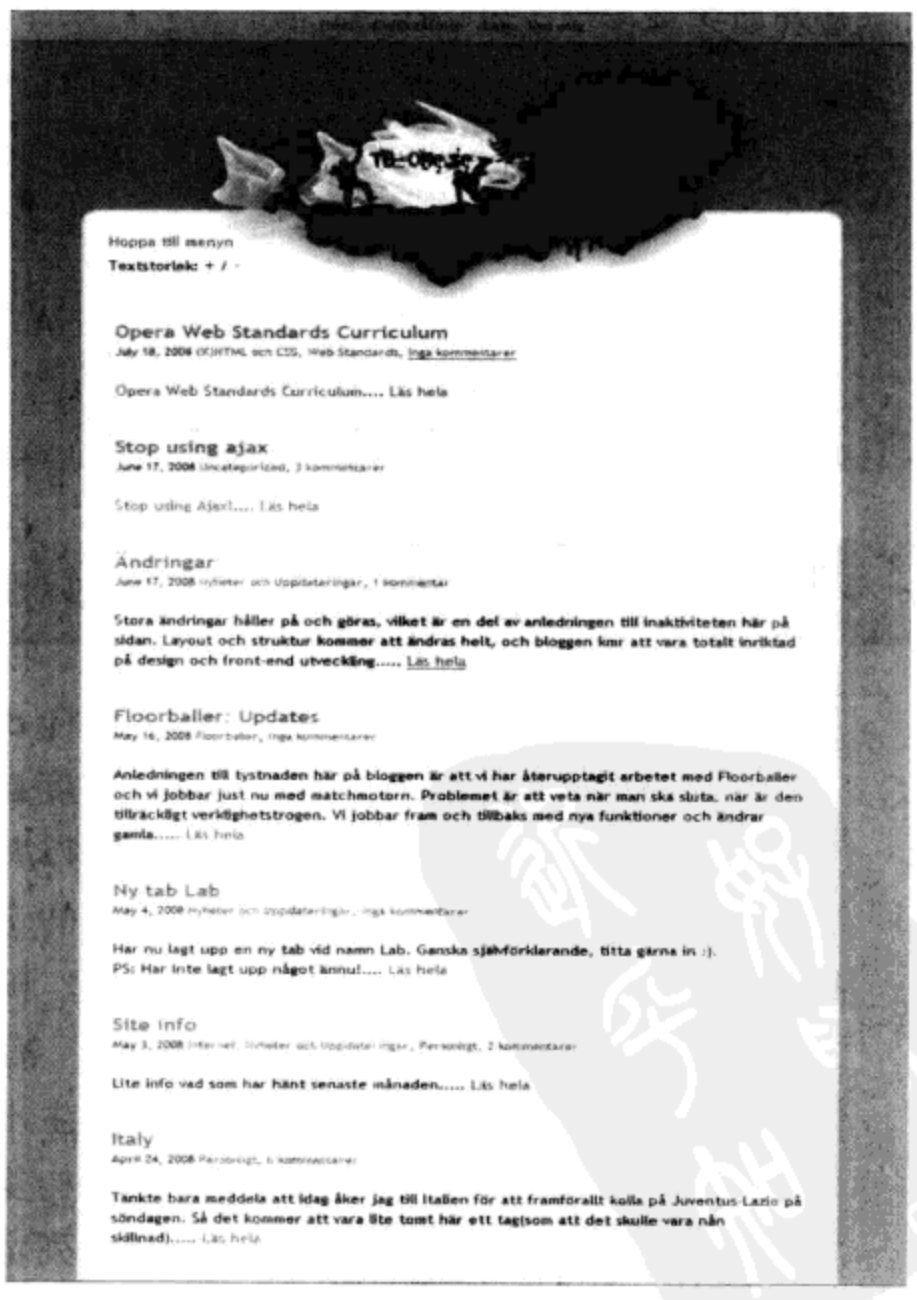


图 14-2　一列固定宽度的 CSS 布局网站

14.2　一列宽度自适应

自适应布局是在网页设计中最常见的一种布局形式，自适应的布局能够根据浏览器窗口的大小，自动改变其宽度或高度值，是一种非常灵活的布局形式，良好的自适应布局网站对不同分辨率的显示器都能提供最好的显示效果。

1．实现原理

实际上 div 默认状态下占据整行空间，便是宽度为 100%的自适应布局的表现形式，一列自适应布局需要做的工作也非常简单，只需要将宽度由固定值改为百分比宽度值。

2．制作实例

制作实例的步骤很简单，同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    一列宽度自适应布局
</div>
```

结构代码还是使用这样的代码。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#layout{
    height: 300px;                    /*设置高度*/
    width: 80%;                       /*设置宽度，单位使用百分比*/
}
```

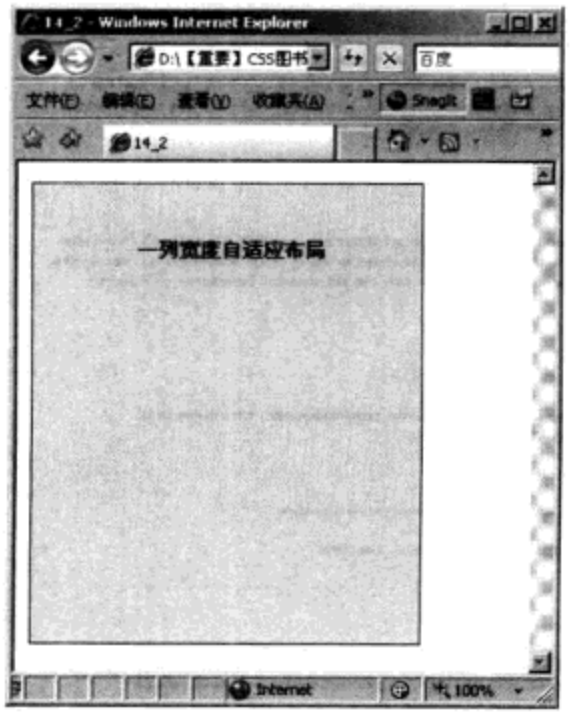


图 14-3　一列宽度自适应布局

自适应布局，大部分使用数值作为参数的样式属性都提供百分比值，width 宽度属性也不例外。在以上代码中，将宽度值重新设置为了 80%，这样 div 的宽度就变成了浏览器宽度的 80%，而宽度自适应的优势就是当扩大或者缩小浏览器窗口大小时，宽度还将维持在浏览器当前宽度比例的 80%。这样宽度自适应的 CSS 布局就实现了，预览效果如图 14-3 所示。

有一些网站是在一列宽度自适应的基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的，如图 14-4 和图 14-5 所示是首先在一列宽度自适应的布局基础上，再进行细致划分布局。

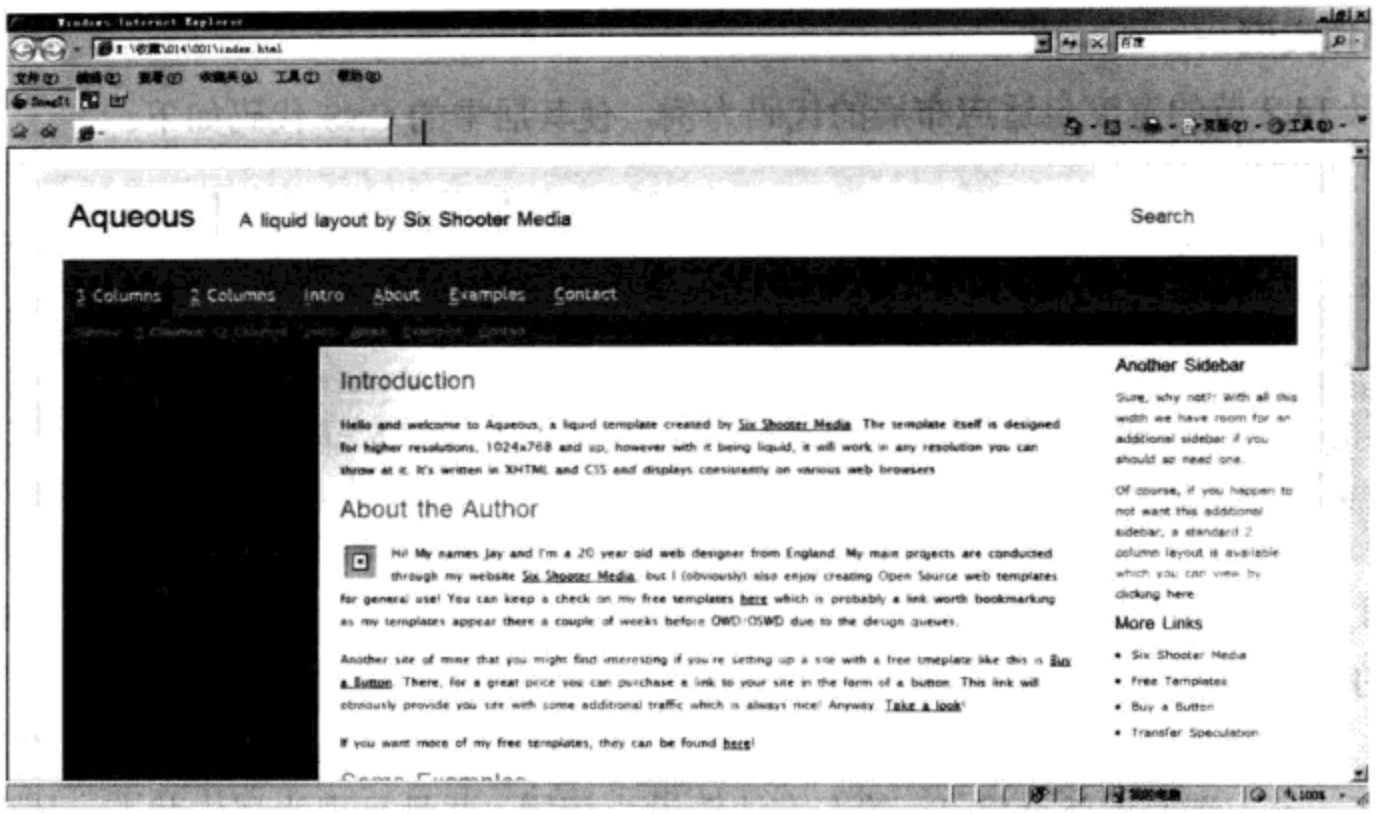


图 14-4　改变浏览器宽度前的页面自适应

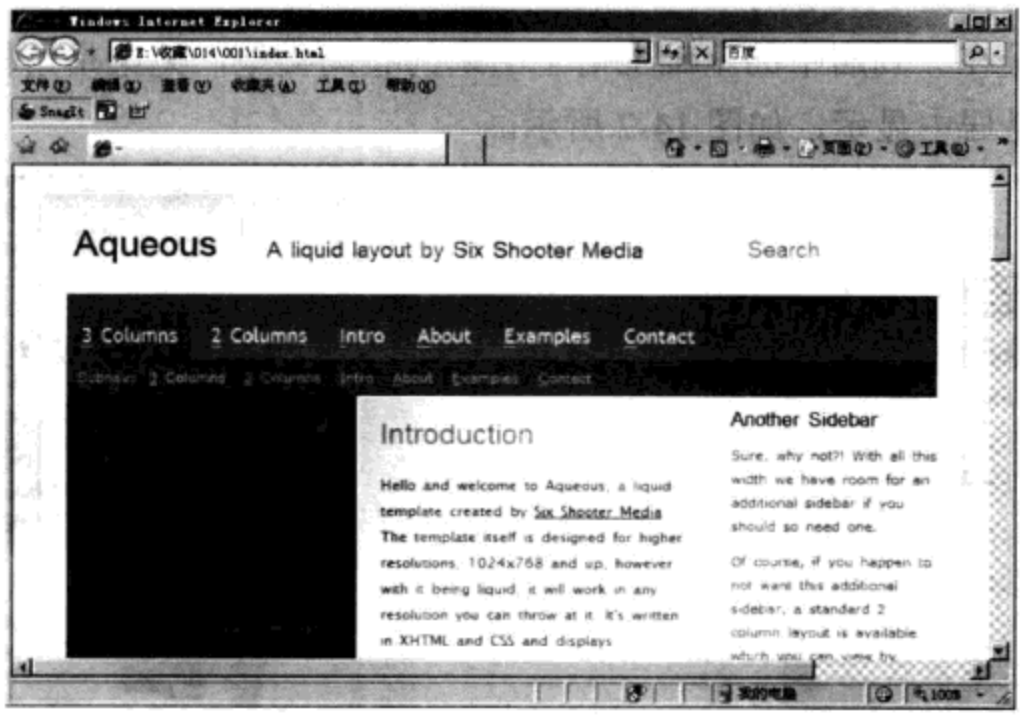


图 14-5　改变浏览器宽度后的页面自适应

14.3　一列居中

页面整体居中是网页设计中常见的形式。在使用 XHTML 表格布局中，使用表格的 align="center"属性来实现整体居中。div 实现居中布局可以使用 align="center"属性，呈现居中状态。然而，CSS 布局是了实现表现与内容的分离，而 align 对齐属性是一种样式代码，写在 XHTML 的 div 属性之中，违背了分离原则，因此应当使用 CSS 的方法实现内容的居中。

1. 实现原理

margin 的一个属性值 auto 是让浏览器自动判断边距，为 div 的左右边距设置了 auto，浏览器就会将 div 的左右边距设为相同，并且呈现为居中显示效果。

2. 制作实例

现在以 14.2 节的宽度自适应布局的代码为例，使其居中的 CSS 代码如下。

```
div{
      [沿用第一节的样式]
}
#layout{
     height: 300px;                                    /*设置高度*/
     width: 80%;                                       /*设置宽度*/
     margin: 0px auto;                                 /*设置外边距，实现对象的居中显示*/
}
```

在以上代码中，最后一行添加了 margin 属性，margin 属性用于控制对象的上、右、下、左 4 个方向的外边框，当 margin 使用了两个参数时，第一个参数表示上下边距，第二个参数表示左右边距。margin 的一个属性值 auto 是让浏览器自动判断边距，在这里就给当前的 div 的左右边距设置了 auto，浏览器就会将 div 的左右边距设为相同，并且呈现为居中状态，从而实现居中布局效果，预览效果如图 14-6 所示。

有很多网站都是在一列居中的基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的，网站要求就是居中显示，如图 14-7 所示。

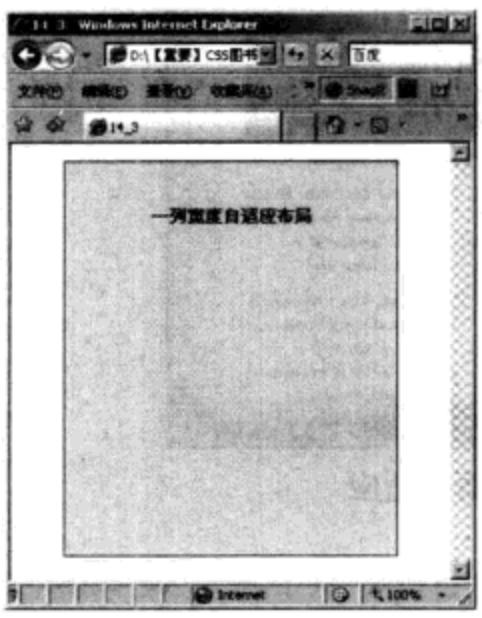


图 14-6　一列居中布局

图 14-7　一列居中网站实例

14.4　二列固定宽度

有了一列固定宽度的基础，二列固定宽度就很简单了。制作一列布局使用到一个 div，那么制作二列布局，自然就需要用到两个 div。本节将讲解如何制作二列固定宽度的 CSS 布局。

1. 实现原理

在一列固定宽度的实现基础上，二列固定宽度就是把每一列的 div 都设定为固定宽度值，div

之间的排列使用浮动定位属性。

2. 制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码。

```
<div id="left">左列</div>
<div id="right">右列</div>
```

以上代码使用了两个 id，分别是 left 和 right，表示两个 div 的名称。需要为它们设置宽度，然后让两个 div 在水平行中并排显示，从而形成二列式布局。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#left
    height: 300px;          /*设置高度*/
    width: 200px;           /*设置宽度*/
    float: left;            /*设置向左浮动定位*/
}
#right{
    height: 300px;          /*设置高度*/
    width: 200px;           /*设置宽度*/
    float: left;            /*设置向左浮动定位*/
}
```

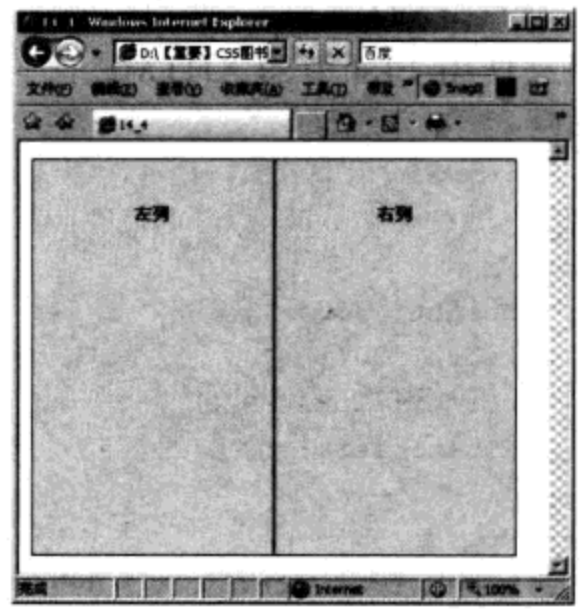


图 14-8　二列固定宽度布局

在以上代码中，left 与 right 两个 div 的代码与前面类似，都使用相同宽高，而为了实现二列式布局，使用浮动属性 float。这里使用向左浮动定位，左栏向左浮动定位，它右侧的右栏对象浮动到它的右侧定位。这样使用 float 属性之后，二列固定宽度的布局就实现了，预览效果如图 14-8 所示。

使用二列固定宽度布局的网站也不占少数，如图 14-9 所示。

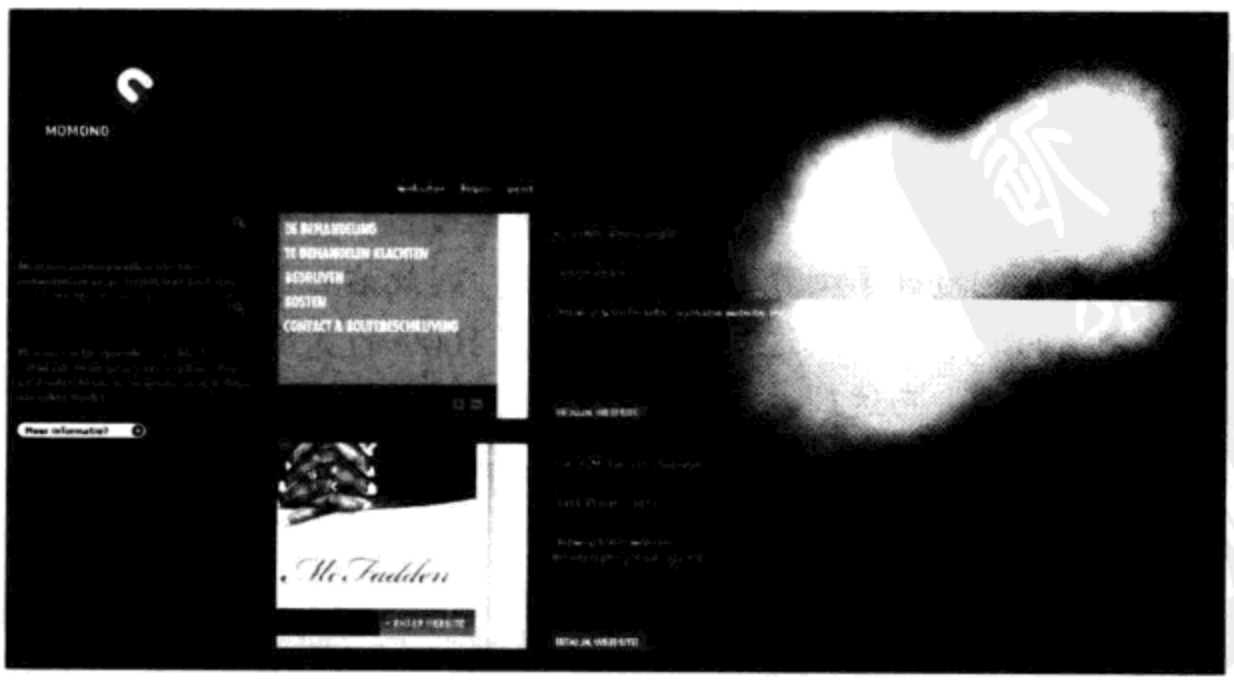


图 14-9　二列固定宽度布局的网站实例

14.5　二列宽度自适应

本节在尝试二列布局的情况下，左右分栏宽度能够做到自适应，由一列宽度自适应布局可知，设定自适应主要通过宽度的百分比值来设置，因此在二列宽度自适应布局中也同样是对百分比宽度值的设计。

1．实现原理

在一列宽度自适应的实现基础上，二列宽度自适应就是把每一列的 div 都设定为百分比值，并且总的宽度百分比值不超过 100%，div 之间的排列使用浮动定位属性。

2．制作实例

继续使用 14.4 节的 XHTML 代码，所做的是重新定义布局的 CSS 样式。

```
div{
      [沿用第一节的样式]
}
#left{
     height: 300px;                          /*设置高度*/
     width: 20%;                             /*设置宽度，使用百分比为单位*/
     border: 1px;                            /*设置 1px 边框线*/
     float: left;                            /*设置向左浮动定位*/
}
#right{
     height: 300px;                          /*设置高度*/
     width: 70%;                             /*设置宽度，使用百分比为单位*/
     float: left;                            /*设置向左浮动定位*/
}
```

在以上代码中，左栏设置为宽度 20%，右栏设置为宽度 70%，看上去像一个左侧为导航、右侧为内容的常见网页布局形式。

注意：

这里没有将浏览器设置为 80%去实现整体的 100%的效果，是为了使布局在预览中更清楚，使用 border 属性，使得左右分栏两个对象都具有 1px 的深色边框线。正如第 6 章提到的，在 CSS 布局之中，一个对象的宽度不仅仅由 width 值来决定，一个对象的真实宽度是由该对象本身的宽、对象的左右外边框以及左右边框，还有内边距这些属性相加而成，因此左侧的对象不仅仅是浏览器窗口的 20%宽度，还应当加上左边的深色边框。这样，左右分栏都超过了自身的百分比宽度，最终的宽度也超过了浏览器窗口的宽度，因此右栏将被浮动定位到第二行显示，那样就达不到左右分栏的效果，因此这里使用了并非 100%的宽度之和。在实际应用中，可以通过避免边框及边距的使用，而达到左右与浏览器填满的效果。

修改 CSS 样式后，修改为二列宽度自适应布局的预览效果如图 14-10 所示。

同样的，CSS 代码也可以如下。

```
#left{
     width: 80%;                             /*设置宽度，使用百分比为单位*/
     float: left;                            /*设置向左浮动定位*/
```

```
}
#right{
     width: 15%;                                    /*设置宽度，使用百分比为单位*/
     float: left;                                   /*设置向左浮动定位*/
}
```

修改两个列的宽度比，左列为 80%，右列为 15%，左列比右列宽，如图 14-11 所示。

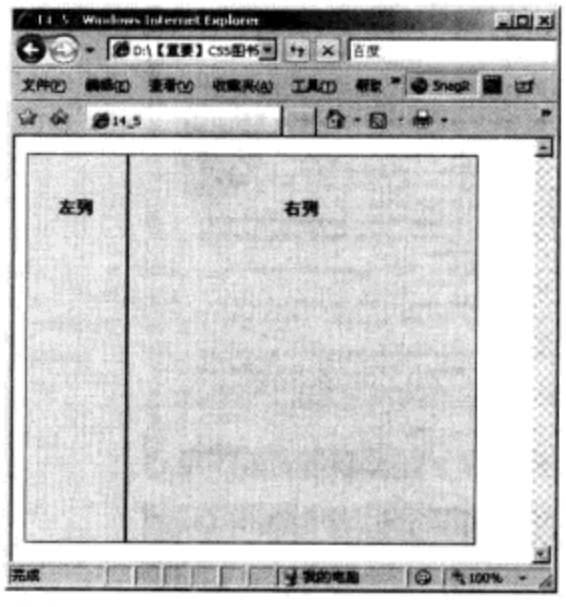


图 14-10　二列宽度自适应布局 1

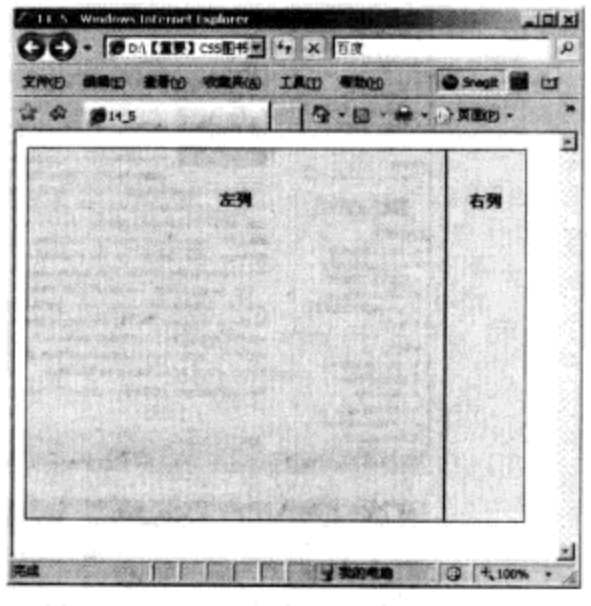


图 14-11　二列宽度自适应布局 2

14.6　两列右列宽度自适应

在实际应用中，有时候需要制作左右两栏的布局，要求右栏固定宽度，左栏根据浏览器窗口的大小自动适应，在 CSS 中实现这样的布局方式是简单可行的，本节就将介绍其制作方法。

1．实现原理

设置左栏的宽度即可，将左栏宽度设定为固定值，右栏不设置任何宽度值，并且右栏不浮动。

2．制作实例

继续使用以前的 XHTML 代码，所做的是重新定义布局的 CSS 样式，代码如下。

```
div{
     [沿用第一节的样式]
}
#left{
     height: 300px;                 /*设置高度*/
     width: 100px;                  /*设置宽度，使用固定宽度*/
     float: left;                   /*设置向左浮动定位*/
}
#right{
     height: 300px;                 /*设置高度*/
     float: left;                   /*设置向左浮动定位*/
}
```

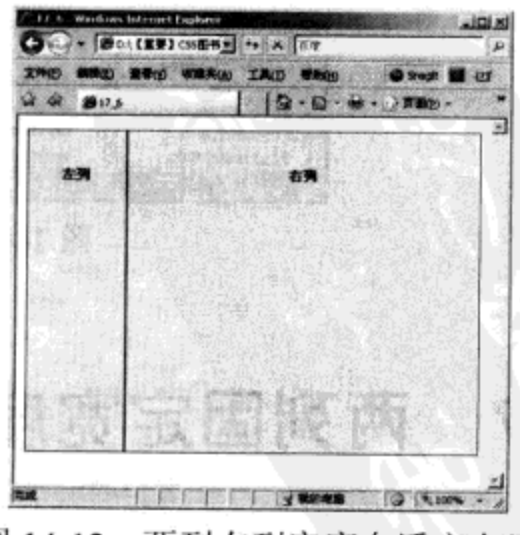


图 14-12　两列右列宽度自适应布局

在以上代码中，左栏将呈现为 100px 的宽度，右栏将根据浏览器窗口的大小自动适应。这样实现了二列右列宽度自适应布局，预览效果如图 14-12 所示。

1. 实现原理

使用一个居中的 div 作为主容器，将二列分栏的两个 div 放置在容器中，从而实现二列的居中显示。

2. 制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码。

```
<div id="layout">
    <div id="left">左列</div>
    <div id="right">右列</div>
</div>
```

在以上代码中，将分栏的两个 div 加上了一个 id 为 layout 的 div 容器。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#layout {
    margin: 0px auto;                    /*设置外边距，实现对象的居中*/
    width: 404px;                        /*设置固定宽度*/
}
#left {
    height: 300px;                       /*设置高度*/
    width: 200px;                        /*设置固定宽度*/
    border: 1px solid #000000;           /*设置 1px 深色边框线*/
    float: left;                         /*设置向左浮动定位*/
}
#right {
    height: 300px;                       /*设置高度*/
    width: 200px;                        /*设置固定宽度*/
    border: 1px solid #000000;           /*设置 1px 边框线*/
    float: left;                         /*设置向左浮动定位*/
}
```

在以上代码中，首先在#layout 元素中将#layout 定义成了居中，这样里面的内容也能够做到居中。

需要注意的是，在#layout 的宽度定义时，将#layout 的宽度设定为 404px。在前面盒模型中讲到过，一个对象的真正宽度是由它的各种属性相加而成，而#left 的宽度为 200px，左右都有 1px 的边距，因此实际宽度是 202px，#right 同样如此。这样，为了让#layout 作为容器能够装下它们两个，宽度则变为#left 和#right 的实际宽度，便得到了 404px 的结构。这样二列固定宽度居中的布局实现了，预览效果如图 14-14 所示。

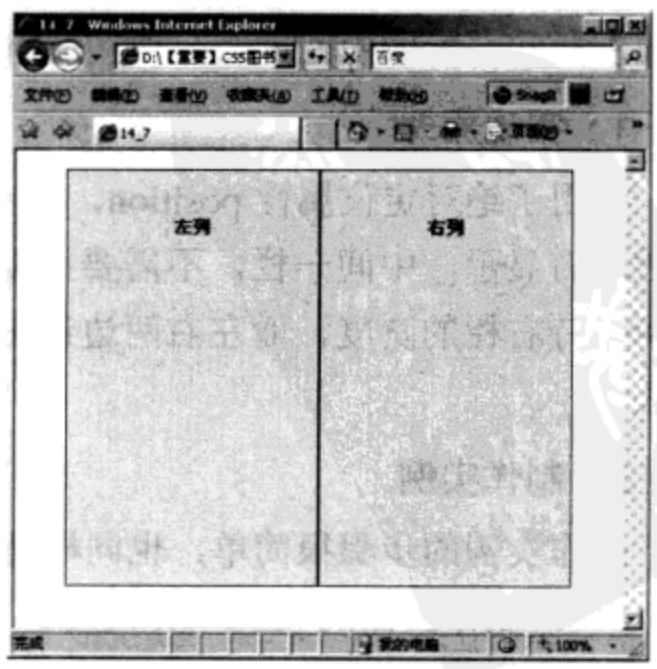


图 14-14　二列固定宽度居中布局

采用两列固定宽度居中的布局网站也有很多，如图 14-15 所示。

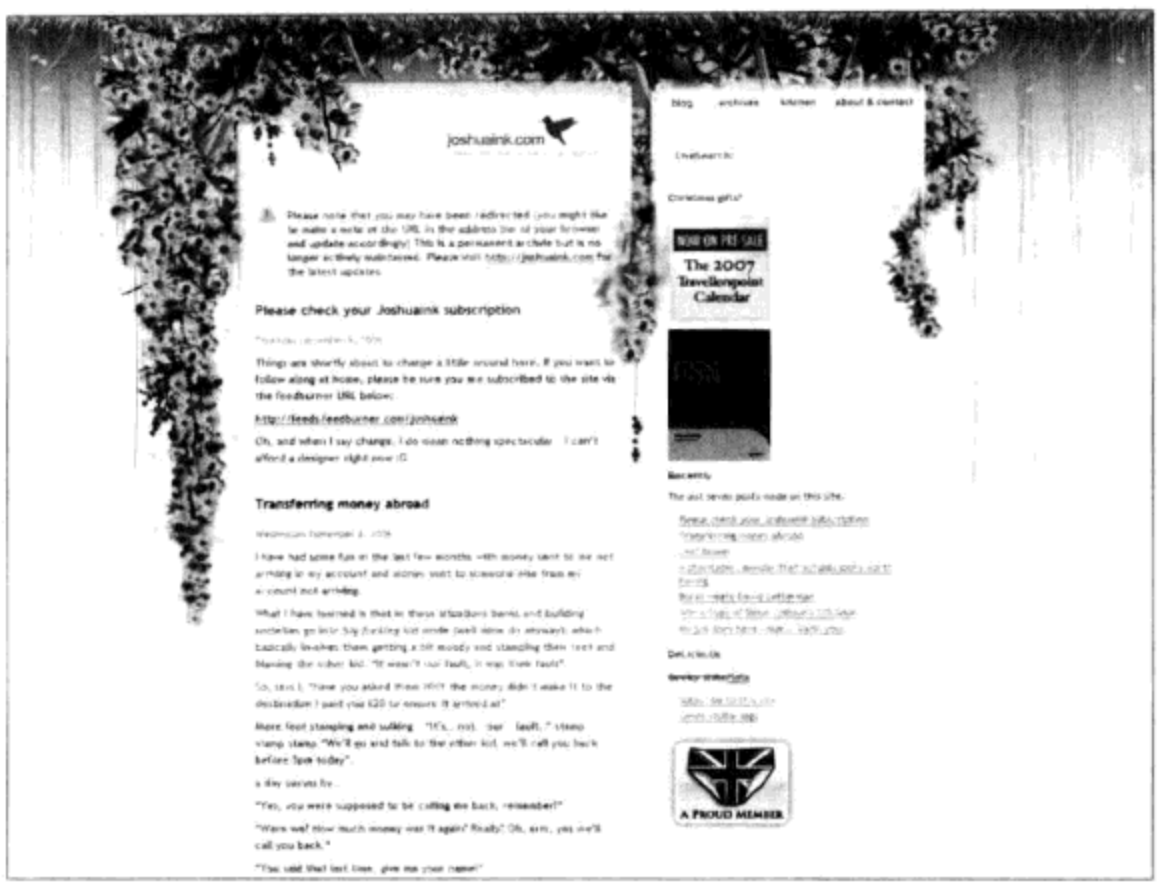


图 14-15　两列固定宽度居中的网站实例

14.8　三列浮动中间列宽度自适应

使用浮动定位方式，从一列到多列的固定宽度以及自适应，基本上可以简单实现，包括三列的固定宽度。

本节将制作一个三列式布局，其中左栏要求固定宽度，并且居左显示，右栏要求固定宽度并居右显示。而中间栏需要在左栏和右栏的中间，根据左右栏的间距变化自适应宽度。这样的布局要求，单纯使用 float 属性与百分比属性并不能够实现，CSS 目前还不支持百分比的计算精确到考虑左栏与右栏的占位，如果对中间栏使用 100%的宽度的话，它将使用浏览器窗口的宽度，而非左栏和右栏的中间间距。现在就将使用定位属性 position 来解决这个问题。

1．实现原理

使用了绝对定位属性 position，将左栏与右栏进行位置控制。左栏贴进左边缘进行显示。而右栏居右显示。中间一栏，不需要再设定其浮动方式，让它的左外边距保持左栏的宽度和右外边距保持右栏的宽度，而左右两边让出的距离，刚好让中栏显示在这个空间中，从而实现布局要求。

2．制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码。

```
<div id="left">左列</div>
```

```
<div id="center">中列</div>
<div id="right">右列</div>
```

以上代码使用了 3 个 div 形式来实现所需要的 3 个分栏结构。

（2）制作左栏的 CSS 样式，代码如下。

```
div{
      [沿用第一节的样式]
}
#left {
     height: 300px;                                    /*设置高度*/
     width: 100px;                                     /*设置高度*/
     position: absolute;                               /*设置相对定位*/
     top: 0px;                                         /*相对定位，设置顶部距离*/
     left: 0px;                                        /*相对定位，设置左边距离*/
}
```

在以上代码中，关键的方法是使用了绝对定位 position: absolute 将左栏进行位置控制，左栏距浏览器左边界 left: 0px，贴近左边缘进行显示，同样它也距浏览器上边界 top: 0px，贴近上边缘进行显示。

（3）制作右栏的 CSS 样式，代码如下。

```
#right {
     height: 300px;                                    /*设置高度*/
     width: 100px;                                     /*设置宽度*/
     position: absolute;                               /*设置相对定位*/
     right: 0px;                                       /*相对定位，设置右边距离*/
     top: 0px;                                         /*相对定位，设置顶部距离*/
}
```

在以上代码中，和左栏设置方法类似，使用了绝对定位 position: absolute 将右栏进行位置控制，右栏将距浏览器右边界 right: 0px，贴近右边缘进行显示，同样它也距浏览器上边界 top: 0px，贴近上边缘进行显示。

（4）制作中栏的 CSS 样式，代码如下。

```
#center{
     height: 300px;          /*设置高度*/
     margin-left: 102px;     /*设置左外边距*/
     margin-right: 102px;    /*设置右外边距*/
}
```

在以上代码中，对于#center，不需要再设定其浮动方式，只需要让它的左外边距永远保持#left 和#right 元素的宽度，便可实现两边各让出 102px 的自适应宽度，而左右两边让出的距离，刚好让左栏和右栏显示在这个空间中，从而实现布局要求。这样三列浮动中间列宽度自适应布局实现了，预览效果如图 14-16 所示。

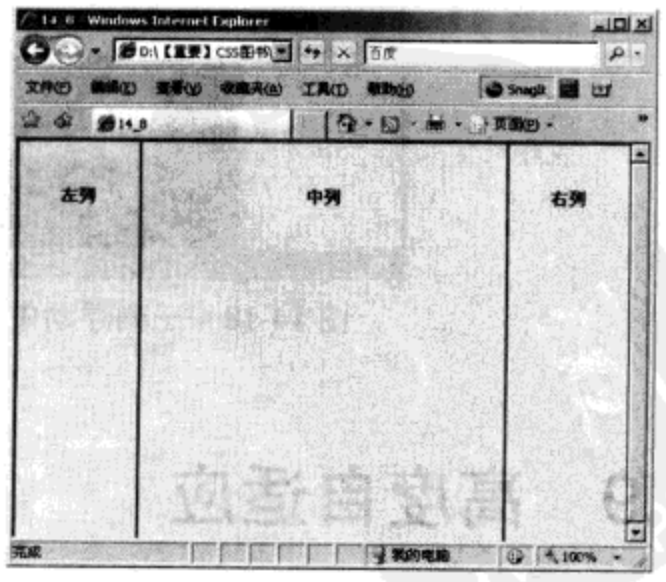


图 14-16　三列浮动中间列宽度自适应布局

三列浮动中间列宽度自适应的布局一般是应用在网站的一部分，例如网站的主体内容，在整个网站的布局基础上，主体部分重新进行布局，所谓布局嵌套布局，如图 14-17 和图 14-18 所示。

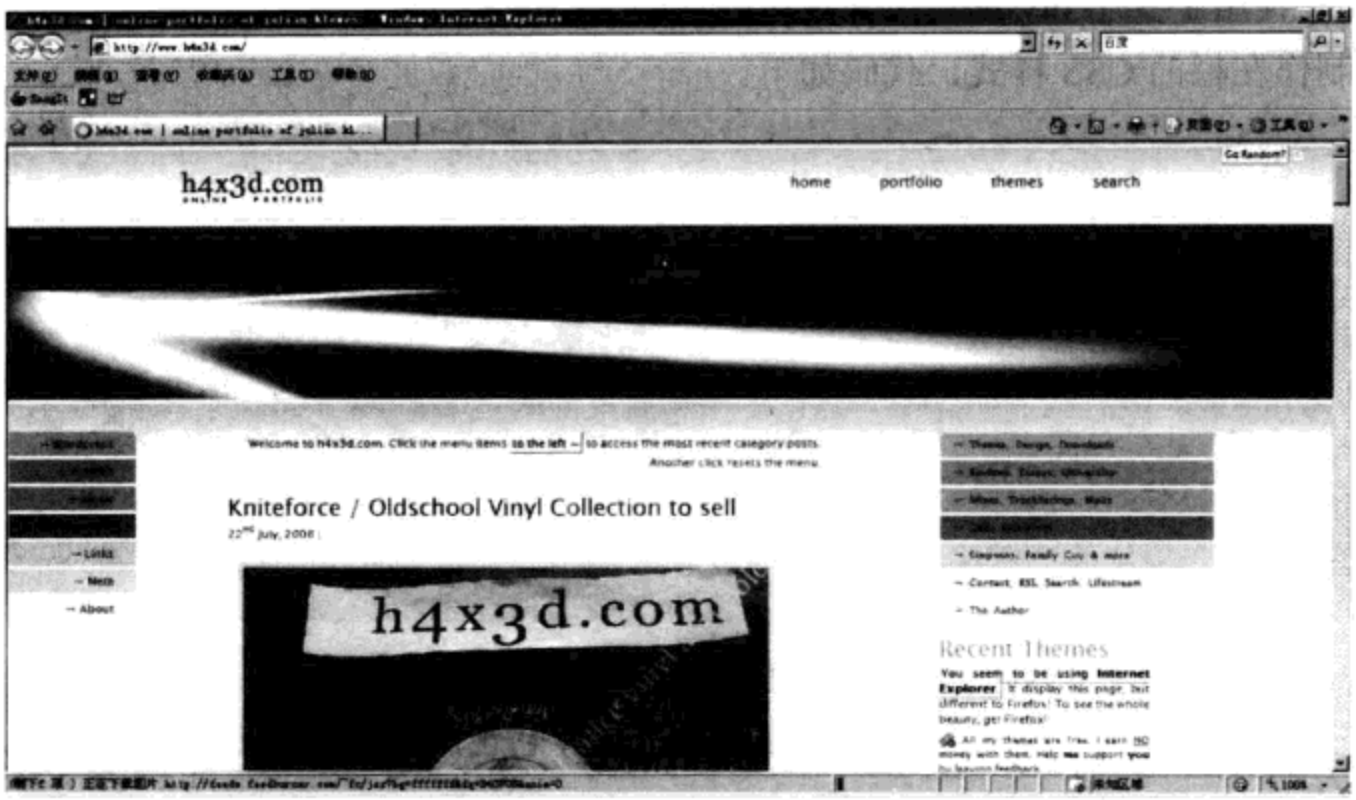


图 14-17　三列浮动中间列宽度自适应的网站实例改变宽度前

图 14-18　三列浮动中间列宽度自适应的网站实例改变宽度后

14.9　高度自适应

前面探讨的都是横向对象之间的排列组合方式，包括横向的宽度自适应，本节将探讨如何进行高度自适应。

实际上，很多人在初次尝试高度自适应时会遇到这样的问题，对象的 height: 100%并不能够直接产生高度自适应的效果，产生的原因是浏览器不支持 height: 100%的编写方法。高度值可以使用百分比进行设置，不同的是，之所以直接使用 height: 100%不能达到效果，与浏览器的解析方式有一定关系。

（1）制作页面的 XHTML 代码。

```
<div id="layout">高度自适应</div>
```

（2）制作布局的 CSS 样式，代码如下。

```
html,body{
    margin: 0px;                          /*设置外边距*/
    height: 100%;                         /*设置高度，使用百分之百高度*/
}
div{
    [沿用第一节的样式]
}
#layout{
    height: 100%;                         /*设置高度，同样使用百分之百高度*/
    width: 300px;                         /*设置宽度*/
    float: left;                          /*设置向左浮动定位*/
}
```

在以上代码中，对#layout 元素设置了 height: 100%，同时为标签选择符 html 和 body 设置了 height:100%，这个就是高度自适应的关键方法所在。

一个对象高度是否可以使用百分比显示，取决于对象的父级对象，#layout 在页面中直接放置在 body 元素之中，因此它的父级就是 body，而浏览器默认状态下，是没有给 body 元素一个高度属性的，因此当直接为#layout 元素设置 height: 100%时，是不会产生任何效果的，而当为 body 设置了 height: 100%之后，它的子级对象#layout 的 height: 100%变发生作用了，这便是浏览器解析规则引发的高度自适应问题。

在 CSS 代码中，除了给 body 设置了高度以外，给 html 元素也设置了相同的样式，这样做是使 IE 与 Firefox 浏览器都能够实现高度自适应，IE 与 Firefox 对页面中对象的解析方法存在一定的差异，这在第 6 章讲到过。在 IE 中 html 元素默认为 100%高度，而 body 却不是。在 Firefox 中 html 元素就不是 100%高度，因此给两个标签都定义为 height: 100%以保证两个浏览器下均能够正常显示。这样高度自适应布局就实现了，预览效果如图 14-19 所示。

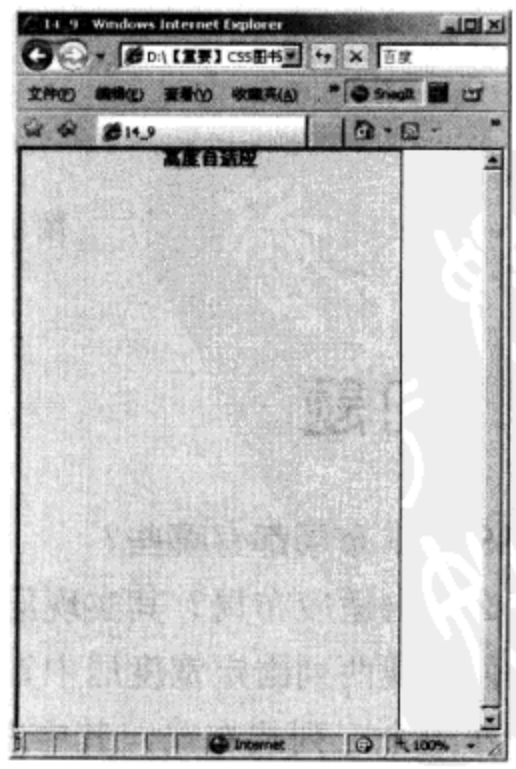


图 14-19　高度自适应布局

14.10　小结

本章讲解的是使用 CSS+DIV 进行网页的基本布局，主要涉及一列固定宽度、一列宽度自适应、一列居中、二列固定宽度、二列宽度自适应、两列右列宽度自适应、两列固定宽度居中、三列浮动中间列宽度自适应和高度自适应 9 种常用的布局样式。

对本章的知识点前后联系进行归纳总结，结构导图如图 14-20 所示。

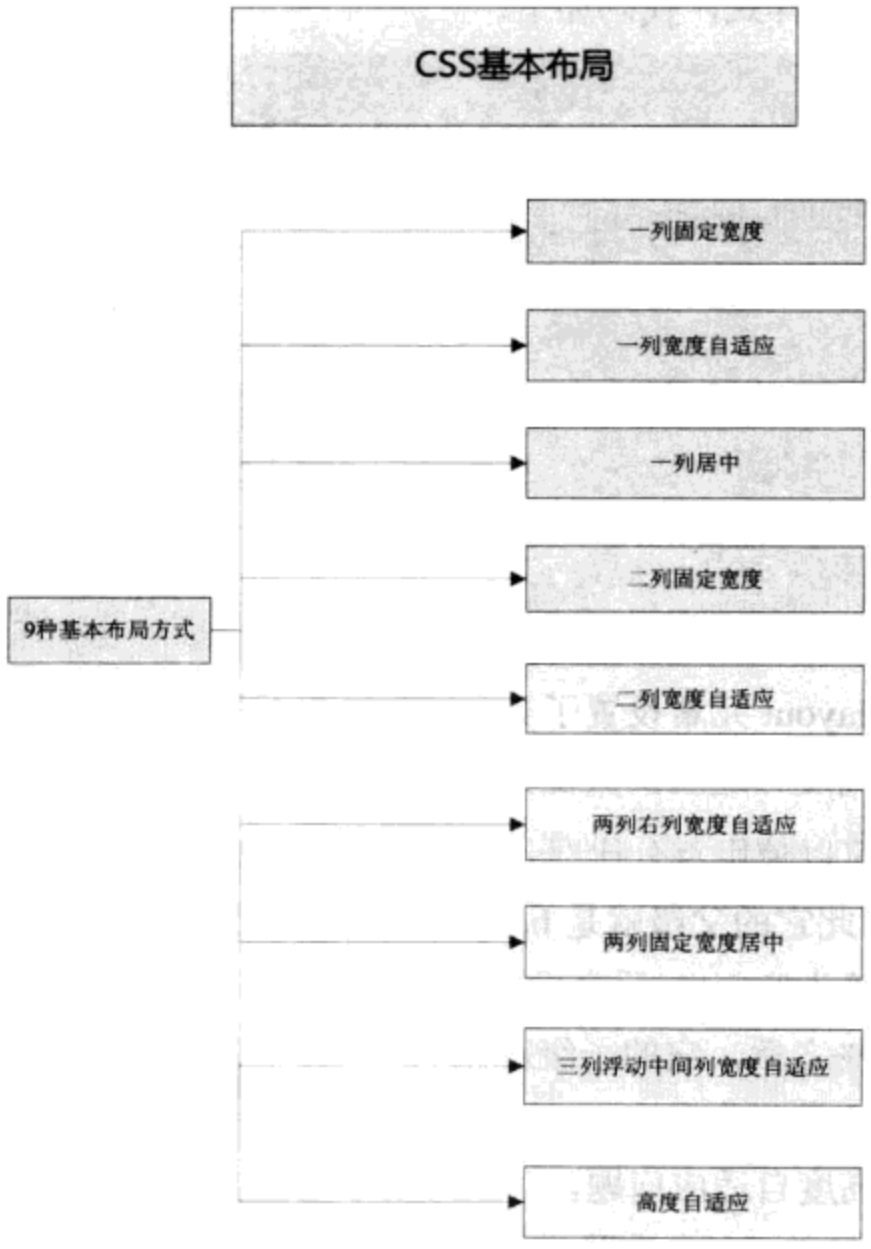


图 14-20　本章知识点结构导图

14.11　习题

1．CSS 基本布局都有哪些？

2．什么是自适应布局？其实现原理是什么？

3．如何实现两列固定宽度居中？

4．制作一个三列式布局，其中左栏要求固定宽度，并且居左显示，右栏要求固定宽度并居右显示，而中间栏需要在左栏和右栏的中间，根据左右栏的间距变化自适应宽度。

5．为了适应浏览器解析规则，如何设置高度自适应？

第15章 CSS整体布局的实现

无论多么复杂的页面布局设计，总是以div为基础，通过几个div之间的组合嵌套来实现。因此在设计较复杂的页面布局之时，要由粗到细的逐步思考，通过一列、二列、三列这种基础的布局方式来相互组合从而实现复杂的布局。本章将探讨制作复杂的CSS页面的实现方法。

15.1 顶行三列布局的实现

这一类布局一般是上部为形象及导航，底部为版权，中间区域分为三列，这样的布局方式得到越来越广泛的应用。

如图15-1所示是http://www.hrajese.cz/网站界面，这个网站是一个很典型的顶行三列式布局。本节将以这个网站为例，介绍如何实现顶行三列式布局。

图15-1 http://www.hrajese.cz/网站界面

15.1.1　制作思路

在制作实例之前，首先计划制作网站的整体构思，以便按照步骤一步一步实现。

（1）勾勒一个草图，理清一下思路，如图 15-2 所示。这样的结构与两列式的布局是非常类似的，区别就是多了一列。

（2）顶部和底部可以设置为一个宽度值，并居中对齐，以适应更大分辨率的需要。将中部的三列，即左栏、中栏、右栏置于一个 div 容器之中，并将此 div 的宽度设置为同顶部一样的宽度值，并居中对齐。再将此 div 容器内的左栏、中栏、右栏分别设置宽度、应用浮动定位属性，以达到顶行三列的布局效果，如图 15-3 所示。

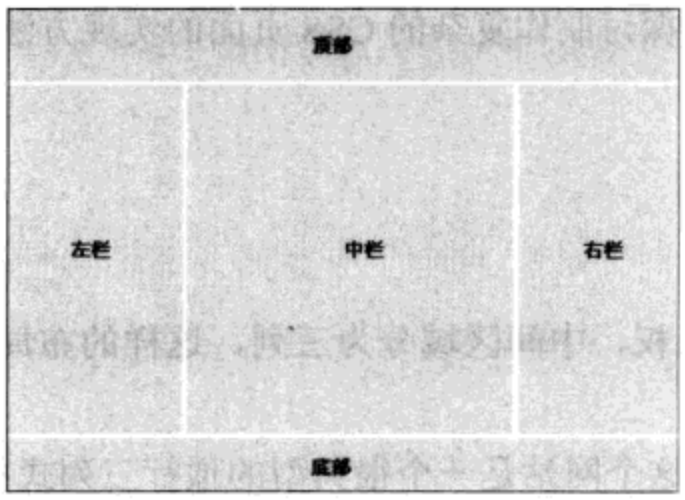


图 15-2　构思草图

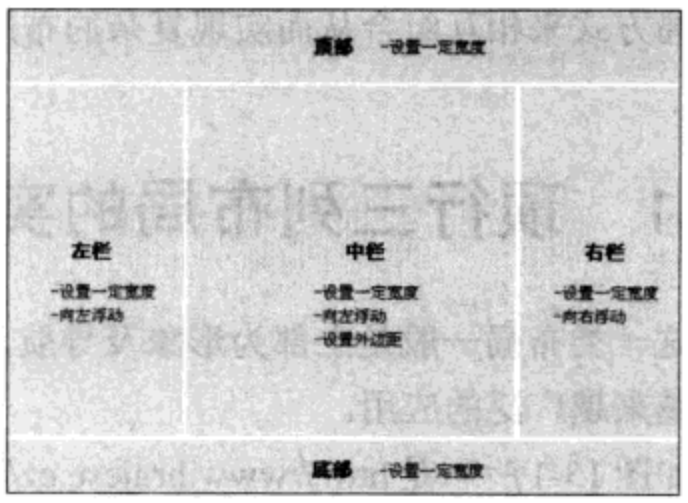


图 15-3　实现方法

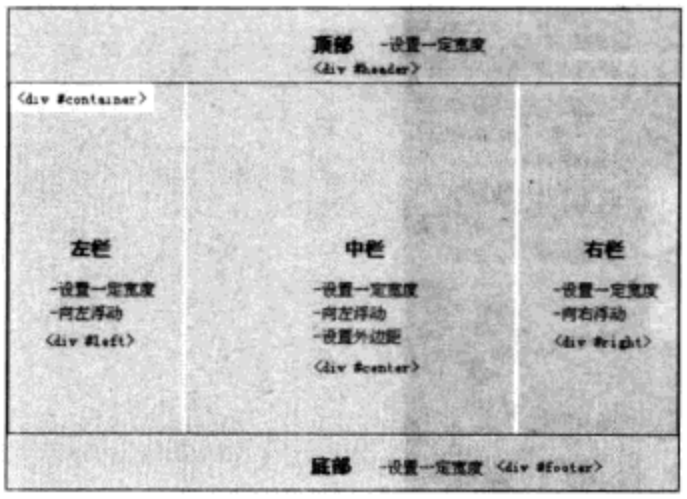


图 15-4　div 层级关系

（3）根据上面的实现方法，整理出各个 div 的 id 以及它们的关系。

- 顶部：#header。
- 中部三列的大容器：#container。
- 左栏：#left。
- 中栏：#center。
- 右栏：#right。
- 底部：#footer。

它们的层级关系如图 15-4 所示。

15.1.2　制作顶部三列式布局实例

（1）制作页面的 XHTML 代码。

```
<div id="header">顶部</div>
<div id="container">
    <div id="left">左栏</div>
    <div id="container">中栏</div>
    <div id="right">右栏</div>
    </div>
```

```
<div id="footer">底部</div>
```

以上代码把 div 层级关系表示在界面预览图上，如图 15-5 所示。

图 15-5　div 层级关系

（2）制作 body 标签元素的 CSS 样式，代码如下。

```
body {
    margin:0;                                    /*设置外边距*/
    padding:0;                                   /*设置内边距*/
    font-size:1em;                               /*设置字号*/
}
```

在以上代码中，使用标签选择符 body 对整体布局进行声明，消除边距，设置文字大小。如果不消除 body 的边距，在 IE 等浏览器中，内容不是从浏览器左上端开始的。

（3）制作纵向布局的主要 CSS 样式，代码如下。

```
#header {
    width:1002px;                                /*设置宽度*/
    height:100px;                                /*设置高度*/
```

```
    margin:0 auto;                                    /*设置外边距，实现对象居中*/
}
#container {
    width:1002px;                                     /*设置宽度*/
    margin:0 auto;                                    /*设置外边距*/
}
#footer {
    width:1002px;                                     /*设置宽度*/
    height:60px;                                      /*设置高度*/
    margin:0 auto;                                    /*设置外边距*/
}
```

在以上代码中，在#header 元素中定义了顶部的宽度和高度，并且水平居中对齐；在#container 元素中，定义了中部大容器的宽度，并且水平居中对齐；在#footer 元素中，定义了底部的宽度和高度，同样是水平居中对齐。

（4）制作中部三列分栏的 CSS 样式，代码如下。

```
#left {
    width:220px;                                      /*设置宽度*/
    float:left;                                       /*设置向左浮动定位*/
}
#center {
    width:580px;                                      /*设置宽度*/
    float:left;                                       /*设置向左浮动定位*/
    margin-left:6px;                                  /*设置左外边距*/
}
#right {
    width:190px;                                      /*设置宽度*/
    float:right;                                      /*设置向右浮动定位*/
}
```

在以上代码中，在#left 元素中定义了左栏的宽度，并且向左浮动定位；在#center 元素中，定义了中栏的宽度，同样是向左浮动定位，并且左边距为 6px。在 CSS 布局的过程中，精确计算是一件很重要的事情。这里之所以将左边距设置成 6px，是由 1002px-220px-580px-190px=12px，12/2=6px 得到的，实现中间两条间隙分割线相等，均为 6px。在#right 元素中，定义了右栏的宽度，设置为向右浮动定位。这样，顶行三列布局便实现了。

15.2　多区域不规则布局

有这样一类网站，如 http://www.uxmag.com，看上去由无数个方块组成，如图 15-6 所示。

该网站从 div 的构成上来看相当复杂，这样的布局的确也需要大量的 div 来划分不同的区域，但是其思路是很简单的，只需要使用一列、二列布局，逐步来做就能够实现。

图 15-6　http://www.uxmag.com/网站界面

15.2.1　制作思路

观察网站发现，uxmag.com 网站实际上是一个居中的页面布局，所有内容都是固定宽度的。制作整体居中，就需要一个 div 作为页面的主框架，也就是一个容器，所有内容都放置在这个主容器中。

（1）勾勒一个草图，理清一下思路，如图 15-7 所示。这样的结构与两列式的布局是非常类似的。

（2）最顶端、顶部、信息提示区和底部可以设置为一个宽度值，并居中对齐，以适应更大分辨率的需要。中间分成两列，即主内容区和右栏，使用浮动定位属性并列排列，这个区域的结构就是二列固定宽度布局，如图 15-8 所示。

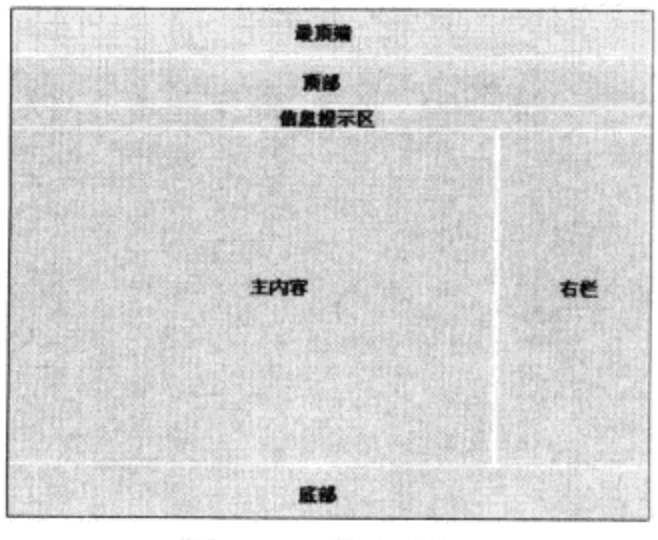


图 15-7　构思草图

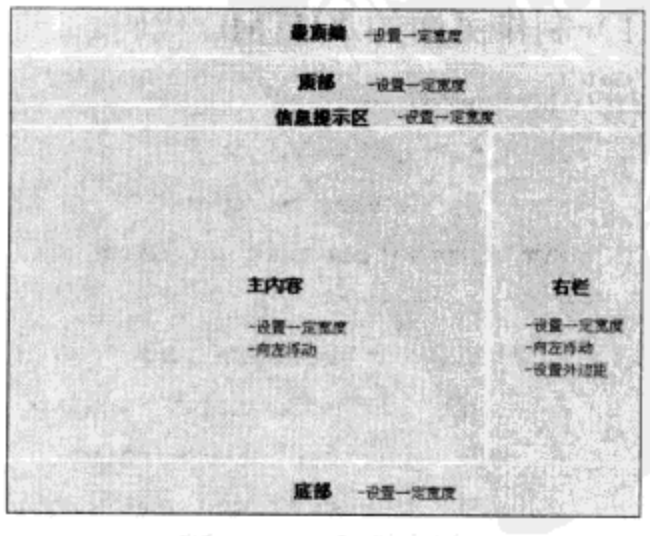


图 15-8　实现方法

（3）根据上面的实现方法，整理出各个 div 的 id 以及它们的关系。

- 最顶端：#top，其中包括网站的搜索区和频道链接。
- 顶部：#header，包括网站 logo。
- 信息提示区：#info，网站的信息提示区，显示一小段提示。
- 主内容：#main，将创建出复杂的页面布局。
- 右栏：#right，用于显示右栏的信息。
- 底部：#footer，是网站的页脚，显示网站的一些功能链接。

它们的层级关系如图 15-9 所示。

（4）主内容区域出现了大量的分栏区域，每一个小区域都使用 div 包含，使用浮动定位属性实现布局，如图 15-10 所示。

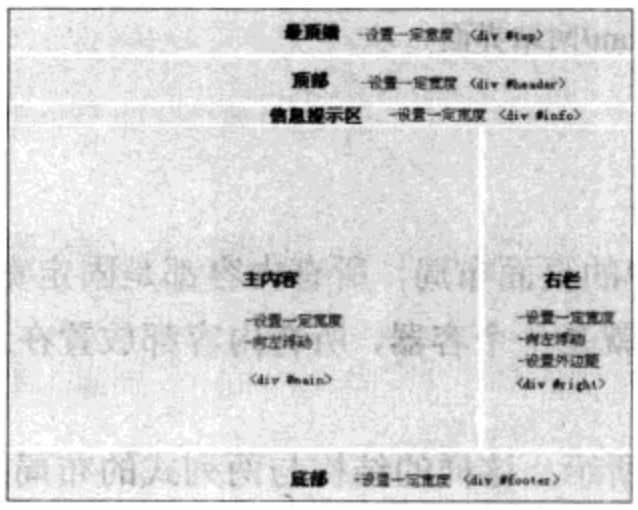


图 15-9　div 层级关系

图 15-10　主内容区域的布局实现

15.2.2　制作多区域不规则布局的实例

（1）制作页面的 XHTML 代码。

```
<div id="container">
    <div id="top"></div>
    <div id="header"></div>
    <div class="clearit"></div>
    <div id="main">
        <div id="featureImage"></div>
        <div id="adSponsor"></div>
        <div id="feature"></div>
        <div class="listItem"></div>
        <div class="listItem"></div>
```

```
            <div id="soundbyte"></div>
            <div class="listItem"></div>
            <div class="listItem"></div>
             [都是div .listItem]
            <div id="copyright"></div>
            <div id="what"></div>
            <div id="who"></div>
        </div>
        <div id="right"></div>
        <div class="clearit"></div>
        <div class="footer"> </div>
    </div>
```

在以上代码中，首先，设计一个主容器 div 作为全页面的框架，id 为 container。然后，在#container 之中，从上至下将内容分为几个区域，分别用 div 包含，这些 div 之间形成了并列关系，并且在样式上基本不需要任何指定，按照顺序从上至下叠放。最后，在这上下叠放的 div 之中，#main 和#right 又形成了一个二列式固定宽度的布局。这样，把 div 层级关系表示在界面预览图上，如图 15-11 所示。

图 15-11　div 层级关系

（2）制作#container 元素的 CSS 样式，代码如下。

```
#container{
```

```
    width: 1002px;                                    /*设置宽度*/
    margin:0 auto;                                    /*设置外边距，实现对象居中*/
}
```

在以上代码中，为主容器定义了宽度值，为了实现居中显示，又设置了外边距为 margin: 0 auto。

（3）制作#container 元素的 CSS 样式，代码如下。

```
#top {
    height:50px;                                      /*设置高度*/
}
#header {
    height:60px;                                      /*设置高度*/
}
#main {
    float: left;                                      /*设置向左浮动定位*/
    width: 750px;                                     /*设置宽度*/
}
#right {
    float: left;                                      /*设置向左浮动定位*/
    width: 250px;                                     /*设置宽度*/
}
#footer {
    height:80px;                                      /*设置高度*/
}
```

在以上代码中，在#top 元素中定义了最顶部的高度，因为 div 元素默认状态下是占据整行显示，所以这里在#container 元素内就占据了#container 元素的整行。后面的元素也是同样。在#header、#footer 元素中，同#top 元素一样，都是定义了顶部和底部的高度值。在#main 元素中，指定了为向左浮动定位显示，使得#right 元素可以浮动到右侧显示，从而实现二列式布局。这里为这两个元素指定宽度，并且宽度综合不超过#container 的宽度，能够显示在同一排中。

（4）制作#container 元素的 CSS 样式，代码如下。

```
#clearit {
    clear: both                                       /*禁止浮动样式*/
}
```

以上代码使用了清除浮动元素的属性，在 div 中使用，将去除掉前后的浮动元素。这也是实现复杂布局时候经常需要使用到的属性。

（5）制作#container 元素的 CSS 样式，代码如下。

```
# featureImage {
    float: left;                                      /*设置向左浮动定位*/
    width: 495px;                                     /*设置宽度*/
    height: 100px;                                    /*设置高度*/
}
# adSponsor {
    float: left;                                      /*设置向左浮动定位*/
    width: 245px;                                     /*设置宽度*/
```

```
    height: 100px;                                    /*设置高度*/
}
# feature {
    float: left;                                      /*设置向左浮动定位*/
    width: 245px;                                     /*设置宽度*/
    height: 100px;                                    /*设置高度*/
}
# soundbyte {
    float: left;                                      /*设置向左浮动定位*/
    width: 215px;                                     /*设置宽度*/
    height: 100px;                                    /*设置高度*/
}
# copyright {
    float: left;                                      /*设置向左浮动定位*/
}
#what {
    float: left;                                      /*设置向左浮动定位*/
}
# who {
    float: left;                                      /*设置向左浮动定位*/
}
.listItem{
    float: left;                                      /*设置向左浮动定位*/
    width: 104px;                                     /*设置宽度*/
    height: 100px;                                    /*设置高度*/
}
```

以上代码充分利用了浮动定位的特性，#main 元素内部的所有 div，全部属于并列关系。之所以能够形成这样的布局，是因为每一个 div 都有固定的宽度以及 float: left 这样的定位方式。

在#main 宽度一定的情况下，第一排的#featureImage 与#adSponsor 的宽度刚好等于#main 的宽度，后面的对象虽然也应该因为浮动定位的原因，跟在#adSponsor 后面显示，但是由于宽度已经被限制了，所以自动转到下一排显示。同样，第二排的#feature、两个.listItem 以及一个#soundByte 的宽度又刚好达到了#main 的宽度，因此后面的对象又将换行显示，以此类推。

这样 20 多个 div 方块就有序的从左向右排列，遇到超过#main 元素的宽度就换行。这种方法也是二列式布局的延伸，充分利用浮动定位原理。这里的关键在于要精确控制 div 的宽度。

15.3　小结

本章主要通过两个网站，分析 CSS 整体布局的制作思路，本章列举了以下两种布局样式。

- 顶行三列布局。
- 多区域不规则布局。

对本章的知识点前后联系进行归纳总结，结构导图如图 15-12 所示。

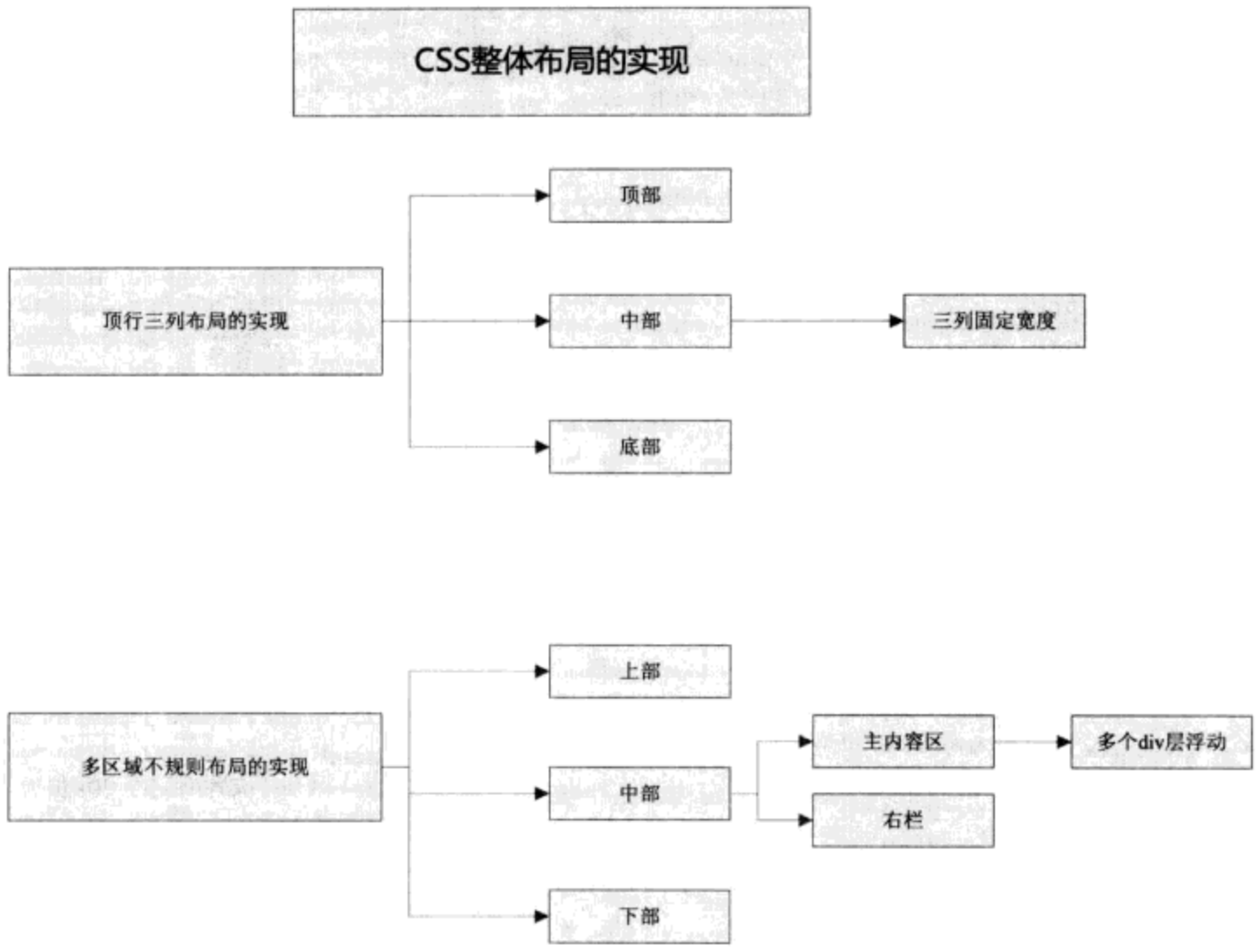


图 15-12　本章知识点结构导图

15.4　习题

1. 简述顶行三列布局的制作思路。
2. 制作一个简单的顶部三列式布局。
3. 简述多区域不规则布局的制作思路。
4. 制作一个简单的多区域不规则布局。

第16章 使用 Dreamweaver 制作页面的实例

掌握了 CSS 的基本布局方法，并且了解了整体布局的实现方式，本章就采用这些方法和制作思想，以一个简单的页面为例，来讲解使用 CSS 和 XHTML 来制作页面的方法和过程。

16.1 框架设计

如图 16-1 所示是本章将要制作的页面设计效果图。

图 16-1 页面的设计效果图

这个页面介绍了 ThinkPad T400 笔记本电脑的配置信息，应用了本书前面所讲到的很多布局实例，例如页面顶部布局设计、面包屑、数据列表、图文排版、页面底部布局设计等。

16.1.1　页面分析

在进行页面代码制作之前，首先对网站进行整体构思，对设计的页面进行分析。

（1）这个页面的结构是两列式布局的一种变形，可以看作上、中、下 3 行，中间行是二列式布局，如图 16-2 所示是这个页面的结构分析图。

从图 16-2 可以看出，现在把页面水平分成了 3 个部分，即顶部、中间和底部，而中间内容，又分成了上、下两个部分，每部分都分为左右两个区域。从整体上来说，页面是上、中、下 3 行布局，从中间部分的划分来说，是二列式的左右布局。无论使用什么样的复杂布局，最终都可以使用 9 种基本的 CSS 布局形式来加以解决。

（2）寻找如何来实现这种框架结构的方法。整体来说，把一个容器分成 3 份，上、中、下分别都使用 div 标签定义空间。中间部分也就是左右宽度固定的二列式布局，使用二列式布局的方法，每列的 div 容器运用浮动定位的 CSS 属性，并且设置合适的宽度，左右定位。以图示的方式表示的框架结构实现方法如图 16-3 所示。

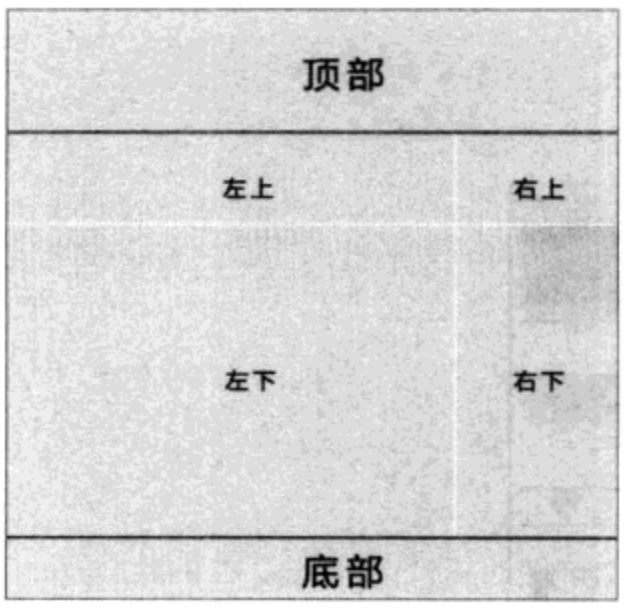


图 16-2　页面结构分析图

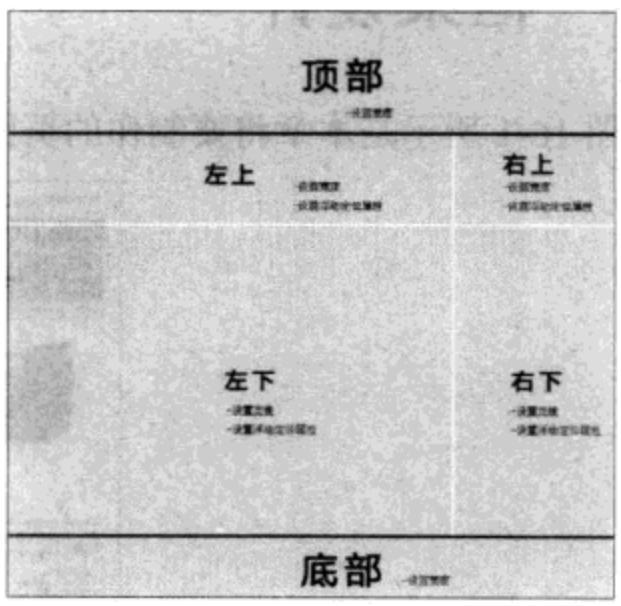


图 16-3　框架结构实现方法示意图

（3）根据上面的实现方法，定义一套 CSS 命名规则，使用类 id 选择符对框架进行定义，根据原页面的内容，整理出各个 div 的 id 以及它们的关系。

- 顶部：#header。
- 中上部：#recommended。
- 中下部：#info。
- 底部：#footer。

页面框架的结构分析清楚之后，就可以为框架制作代码了。

16.1.2　框架制作

框架制作的方法，包括 XHTML 代码的制作和 CSS 样式的制作两个方面。下面分别对这两种方法进行具体讲解。

1．XHTML 代码制作

根据页面分析的结果，现在为页面框架进行代码制作。打开 Dreamweaver CS5，新建一个 HTML 文件，如图 16-4 所示。

在新建的 HTML 文件中，修改标题名字，编码方式修改为 gb2312，如图 16-5 所示。

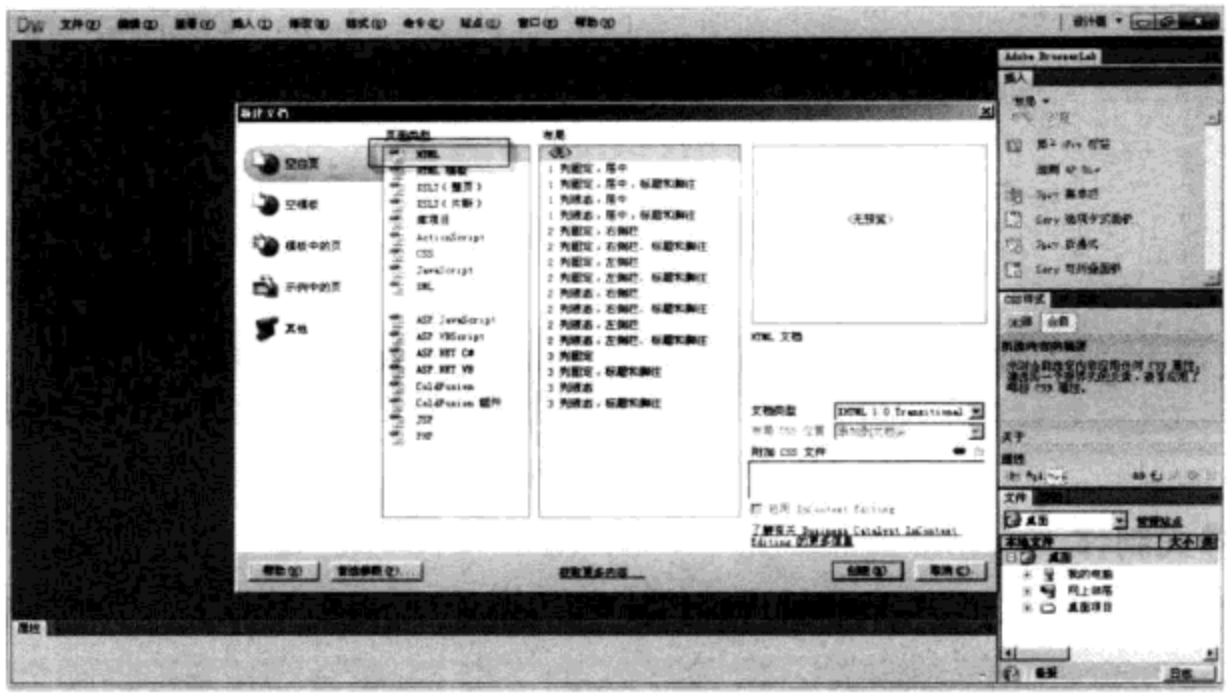


图 16-4　Dreamweaver 打开界面

图 16-5　编辑 XHTML 代码

根据框架结构命名规则，将其应用在代码编写中。首先编写 XHTML 代码，在 XHTML 格式文件的主体<body></body>内添加如下 XHTML 代码。

```
<div id="container">
    <div id="header"></div>
    <div id="recommended"></div>
    <div id="info"></div>
    <div id="footer"></div>
</div>
```

以上代码在 Dreamweaver 中的效果如图 16-6 所示。

图 16-6　XHTML 代码在 Dreamweaver 中的效果

2. CSS 样式制作

现在来添加 CSS 样式，在 XHTML 格式文件的<head></head>中添加，代码如下。

```
<style type="text/css"></style>
```

然后，使用 CSS 选择符为定义了 id 的 XHTML 标签进行样式设置。添加 CSS 的过程中，要经常参照 UI 界面的设计图，并且使用 Photoshop 软件打开设计图，利用参考线精确坐标和页面中每个模块的位置关系，将这个页面的框架结构进行划分，如图 16-7 所示。

在参考线的帮助下，可以掌握页面中每个模块的位置以及长度、宽度，这是 CSS 样式编写的重要辅助方法。这里添加的具体 CSS 文件如下。

```
body {
    margin: 0px;                                          /*设置外边距*/
    padding: 0px;                                         /*设置内边距*/
    font-size: 12px;                                      /*设置字号*/
    font-family: "宋体";                                  /*设置字体类型*/
}
#container {
    width: 800px;                                         /*设置宽度*/
    margin: 0px auto;                                     /*设置外边距，使其居中*/
}
#header {
    margin-top: 24px;                                     /*设置上外边距*/
}
#recommended {
    margin-top: 50px;                                     /*设置上外边距*/
}
#footer {
    background: url(images/footerbg.jpg) top no-repeat;   /*设置背景样式*/
    height: 130px;                                        /*设置高度*/
}
```

在以上代码中，使用标签指定式选择符 body 进行页面设置，使用统一的字号以及字体类型，内边距和外边距设置为 0 的目的是兼顾各种版本浏览器对外边距或者内边距解析的差异，都设

置为 0，就可以统一不同浏览器的解读都为 0。使用类选择符#container 设置了页面的宽度，并且使用外边距设置居中布局的方法，使页面在浏览器中居中显示。

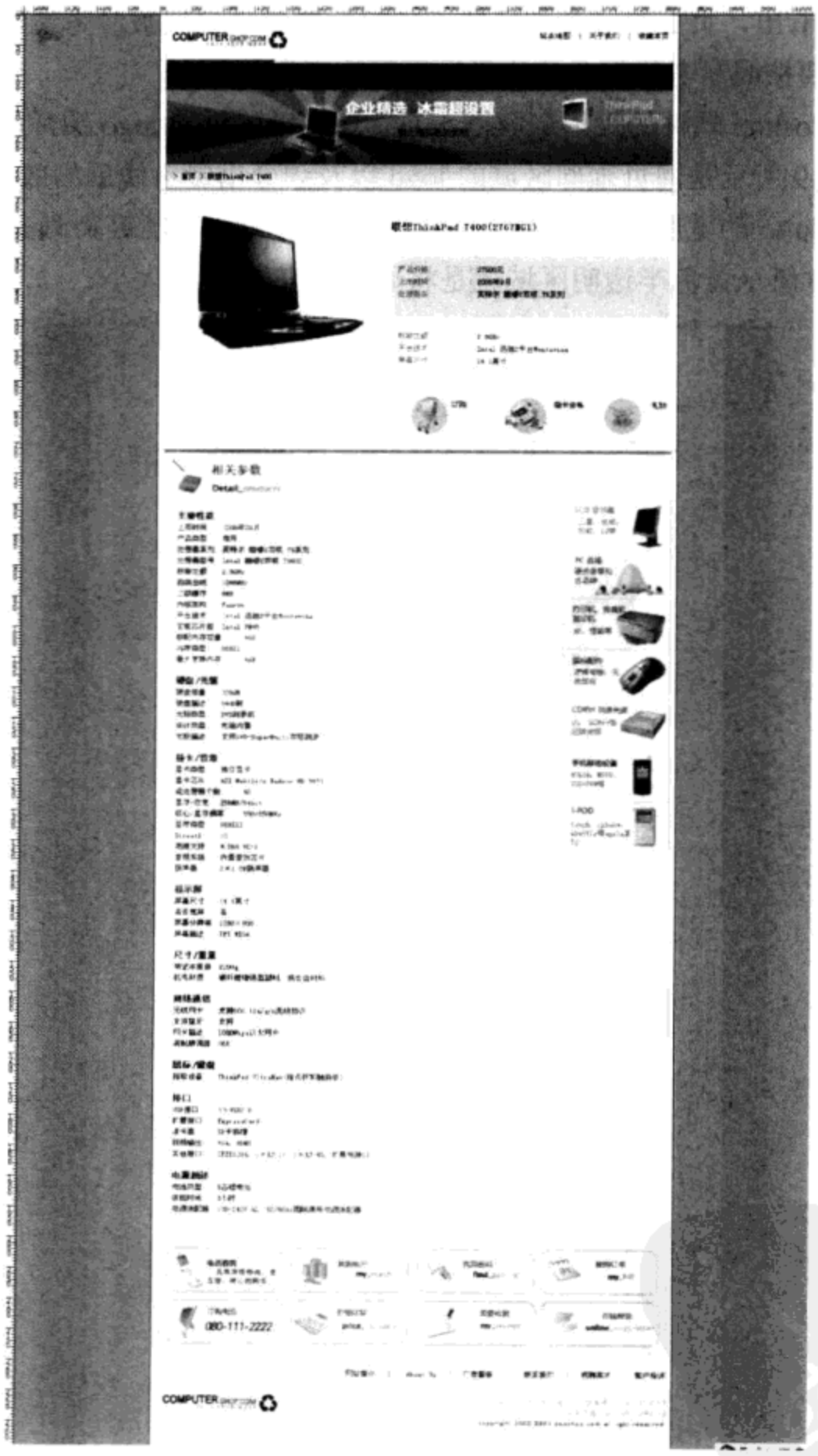


图 16-7　参考线划分框架结构

16.1.3　切图

切图在前端页面代码制作的过程中是一项很重要的工作。本书用一个小节的篇幅，讲解使用 Photoshop 软件来切图的方法。

（1）使用参考线。使用 Photoshop 软件打开设计图源文件，使用组合功能键【Ctrl+R】打开标尺工具，然后使用光标从标尺区域拖出参考线，并且使参考线对齐页面元素，如图 16-8 所示。

从图 16-8 可以看出，页面中各个元素的结构和位置关系已经被参考线明显地划分开来。接下来的工作，只需要按照参考线划分的边界进行图片切割。

（2）使用 Photoshop 软件的矩形选取框工具。现在以网页的 logo 图片的切图为例，讲解切图的方法。如图 16-9 所示是网页顶部区域的 logo 以及划分出参考线后的效果图。

参考线划定了 logo 的宽度和高度，同时通过标尺和参考线也能够得到 logo 的上外边距和左外边距。如图 16-10 所示黄色半透明区域就是将要切出来的 logo 大小。

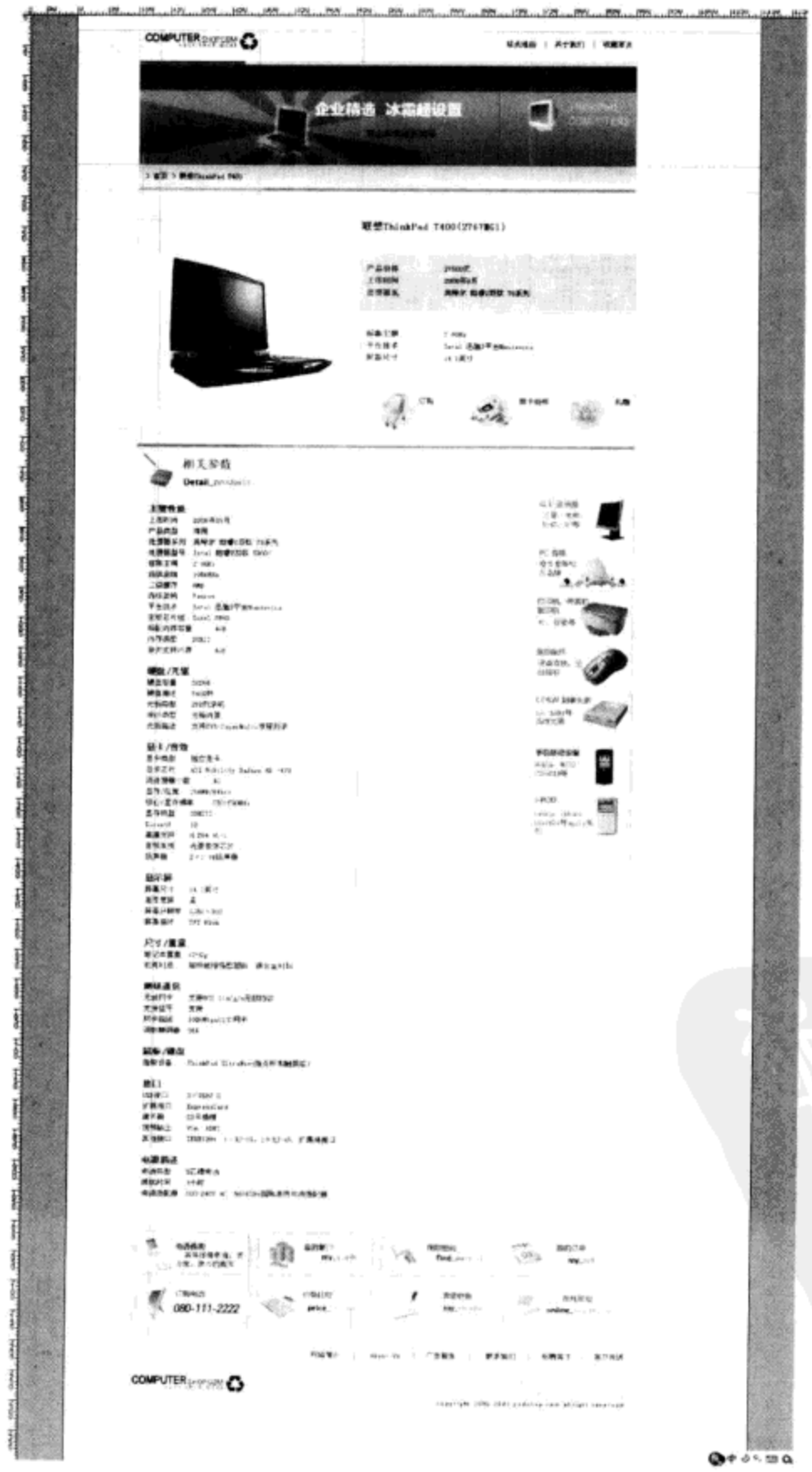


图 16-8　在 Photoshop 软件中打开参考线后的页面预览效果

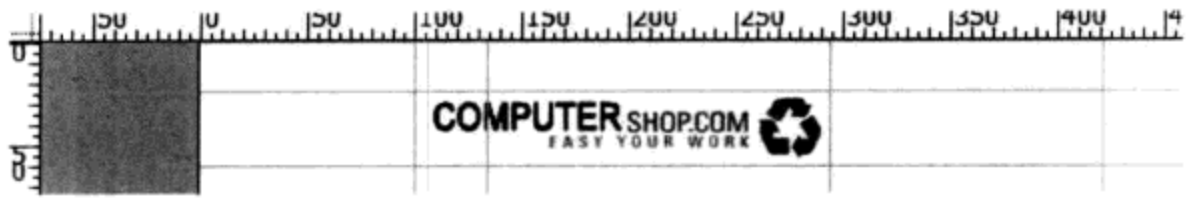


图 16-9　页面顶部区域的 logo 以及划分出参考线后的效果

图 16-10　参考线和标尺能够表示出很多信息

现在使用 Photoshop 工具栏中的“矩形选取”工具，如图 16-11 所示。沿着参考线拖动“矩形选取”工具，选取图 16-10 中的黄色区域，如图 16-12 所示。

图 16-11　“矩形选取”工具

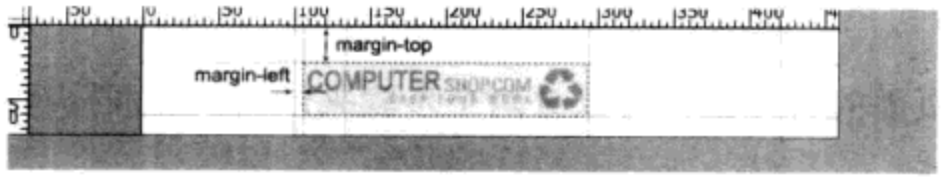


图 16-12　选取 logo 区域

（3）复制图片。选取所需要的图片之后，使用“合并拷贝”命令或按【Ctrl+Shift+C】键，然后再使用“创建新文件”命令或按【Ctrl+N】键，将其复制的图片粘贴在新的文件中，使用“粘贴”命令或按【Ctrl+V】键，完成切图工作，如图 16-13 所示。

图 16-13　切出来的图片

（4）存储图片。切出来的图片，使用“存储为 Web 所用格式”命令或按【Alt+Ctrl+Shift+S】键，存储为一种格式的图片，一般选择 jpg 格式或者 gif 格式，如图 16-14 所示。

可以存在预先准备好的文件夹中，以备页面制作过程中使用。其他的图片都可以按照这种方法一个一个切出来。

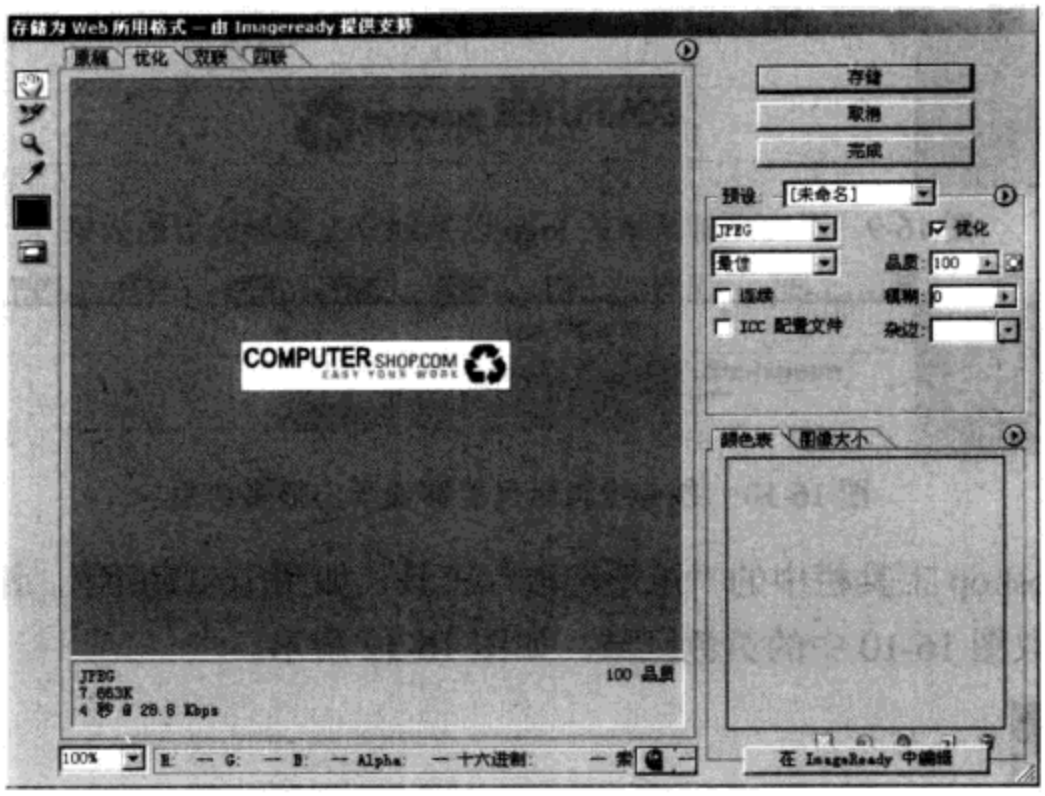


图 16-14　存储图片文件选项

16.2　布局设计

布局设计，就是实现页面中各个部分如何按照设计的样式放置在页面代码中的过程。现在在本章的实例页面中已经完成了框架结构的分析和制作，接下来，就为框架中每一部分的组成部分进行布局设计。需要进行的布局设计包括页面顶部的布局设计、中上部分的布局设计、中下部分的布局设计以及底部的布局设计。

16.2.1　页面顶部布局设计

页面的布局设计可以分两个步骤来制作：页面分析和代码的制作，下面分别讲解具体的实现方法。

1．页面分析

在进行布局制作之前，首先进行顶部内容的整体构思，以便按照步骤一步一步地实现。

（1）观察页面顶部的设计图，如图 16-15 所示。

图 16-15　页面顶部的设计图

根据设计图勾勒一个草图，理清一下思路，如图 16-16 所示。这样的结构与两列式的布局是非常类似的，区别就是多了一列。

（2）logo 部分与快速导航可以视为二列式布局，使用浮动定位方法实现布局设计，并且将二者放在同一个 div 容器中。中间的 banner 部分和下面的“面包屑”分别使用一个 div 容器，将宽度设置为与页面一样的宽度，每个部分占据一行，如图 16-17 所示。

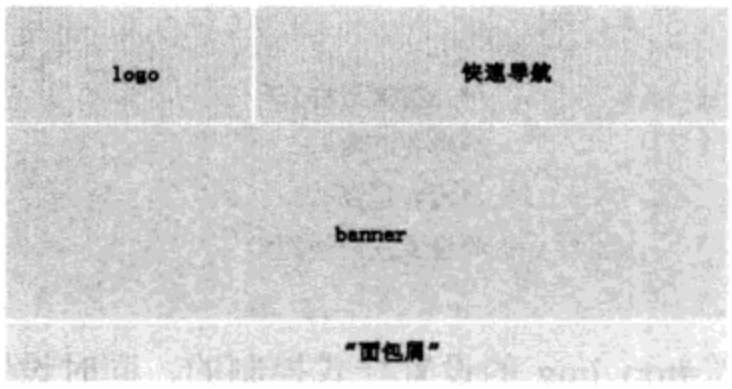


图 16-16　构思草图

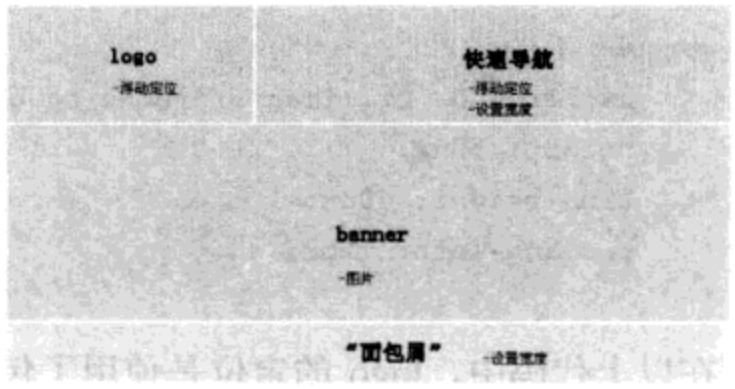


图 16-17　实现方法

（3）根据上面的实现方法，为每个区域的 div 的 id 或者 class 命名。

- logo：#logo。
- 快速导航：.nav。
- banner：#banner。
- “面包屑”：.crumbs。

2. 代码制作

（1）在框架代码的基础上，添加页面顶部的 XHTML 代码，找到如下代码：

```
<div id="header"></div>
```

然后在其中编写代码。

```
<div id="header">
    <div id="top"><img src="images/logo.jpg" alt="logo" />
        <div class="nav"></div>
    </div>
    <div id="banner"><img src="images/banner.jpg" alt="banner" /></div>
    <div class="crumbs"></div>
    <div><img src="images/crumbsline.jpg" /></div>
</div>
```

在以上代码中，logo 图片和快速导航放在了一个 id 为 top 的 div 容器中，然后与 banner 的 div 容器和“面包屑”的 div 容器并列成 3 行。完成结构制作之后，开始添加 CSS 样式，实现每个部分的布局设计。

（2）现在要为每一个部分实现精确定位以及主要样式的制作，代码如下。

```
#top img {
    margin-left:6px;                                    /*设置左外边距*/
    float: left;                                        /*向左浮动定位*/
}
```

```
.nav {
    float: right;                                          /*向右浮动定位*/
    margin-top: 15px;                                      /*设置上外边距*/
}
#banner img {
    margin-top: 13px;                                      /*设置上外边距*/
}
.crumbs {
    background: url(images/crumbs_bg.gif) repeat-x;        /*设置背景样式*/
    height: 38px;                                          /*设置高度*/
    line-height: 40px;                                     /*设置行高*/
    padding-left: 12px;                                    /*设置左内边距*/
}
```

在以上代码中，logo 的定位是使用了包含选择符#top img 的设置样式控制的，同时设置了左外边距。同样，banner 图片也是使用包含选择符#banner img 的设计样式控制的。在设置“面包屑”文字链接的过程中，定义了背景图片的样式以及文本的行高，目的是在以后制作文字的时候，能够使文字在垂直位置居中显示，实现页面设计的效果。页面顶部主体布局的预览效果如图 16-18 所示。

图 16-18　页面顶部主体布局的预览效果

16.2.2　页面中上部的布局设计

1．页面分析

完成了页面顶部的布局设计之后，接下来就可以对页面中上部的整体进行构思，以便按照步骤一步一步地实现。

（1）观察页面中上部的设计图，如图 16-19 所示。

根据设计图勾勒一个草图，理清一下思路，如图 16-20 所示。这部分主体的结构是一个宽度固定的二列式布局，然后右列又分成了 4 行，包括标题文字、数据列表和图片按钮。

图 16-19　页面中上部的设计图

（2）左侧图片和右列使用浮动定位实现二列式布局，并且设置固定宽度。标题文字、数据列表和图片按钮放在右侧一个 div 容器中，标题可以使用标题标签<h1>，不仅能够实现布局，同时又能设置文字样式，数据列表和图片按钮都使用 div 容器，如图 16-21 所示。

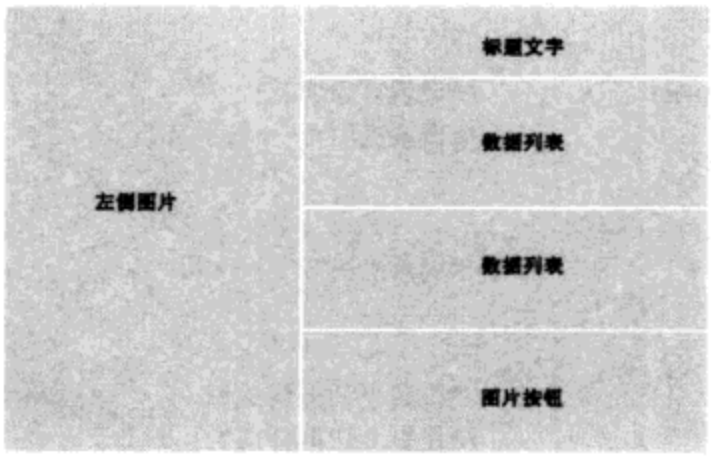


图 16-20　构思草图

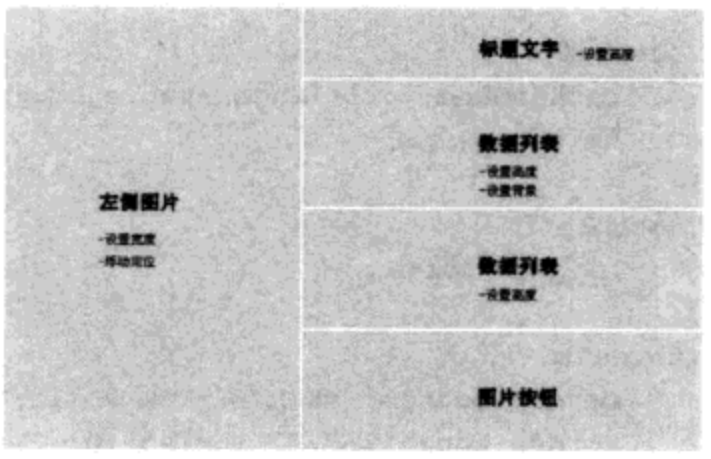


图 16-21　实现方法

（3）根据上面的实现方法，为每个区域的 div 的 id 或者 class 命名。

- 左侧图片：.feature。
- 右列：.intro_info。
- 标题文字：h1。
- 数据列表 1：.bgtable。
- 数据列表 2：.notable。
- 图片按钮：.threeBtn。

2. 代码制作

（1）继续在框架代码的基础上，添加制作页面顶部的 XHTML 代码，找到如下代码：

```
<div id="recommended"></div>
```

然后在其中编写 XHTML 代码。

```
<div id="recommended">
    <div class="feature"><img src="images/productshow.jpg" alt="thinkpad t400" /></div>
    <div class="intro_info">
        <h1>联想 ThinkPad T400(2767MG1)</h1>
```

```
            <div class="bgtable">
            </div>
            <div class="notable">
            </div>
            <div class="threeBtn">
        </div>
</div>
```

在以上代码中，按照分析的结构以及实现方法，制作出来结构代码。

（2）现在实现每个部分的布局，添加 CSS 代码，代码如下。

```
.feature {
    float: left;                                          /*设置向左浮动定位*/
    margin-left: 34px;                                    /*设置左外边距*/
    margin-top: 50px;                                     /*设置上外边距*/
}
.intro_info {
    float: left;                                          /*设置向左浮动定位*/
    margin-left: 38px;                                    /*设置左外边距*/
    width: 441px;                                         /*设置宽度*/
}
.bgtable {
    background: url(images/table_bg.jpg) repeat-x;        /*设置背景样式*/
    height:110px;                                         /*设置高度*/
}
.notable {
    height: 110px;                                        /*设置高度*/
}
.threeBtn {
    border-bottom: #E7E7E7 1px solid;                     /*设置下边框样式*/
    border-top: #E7E7E7 1px solid;                        /*设置上边框样式*/
    padding-top: 12px;                                    /*设置上内边距*/
    padding-bottom: 12px;                                 /*设置下内边距*/
}
```

在以上代码中，使用之前定义的选择符名字，为左列和右列设置浮动定位，实现二列式布局。在代表有背景的数据表格的选择符中，设置了背景样式。在为图片按钮定义样式的代码中，使用上边框和下边框的属性，实现页面设计中的分割线。页面中上部布局的预览效果如图 16-22 所示。

图 16-22　页面中上部布局的预览效果

16.2.3　页面中下部的布局设计

完成了页面中上部的布局设计之后，接下来就可以对页面中下部进行整体构思，以便按照步骤一步一步地实现。

1．页面分析

（1）观察页面中下部的设计图，如图 16-23 所示。

图 16-23　页面中下部的设计图

根据设计图勾勒一个草图，理清一下思路，如图 16-24 所示。这部分主体的结构可以看作是上、中、下 3 行，上面一行是这个区域的标题部分，中间一行又可以看作一个二列式布局，下面一行是图片按钮区域。

（2）从图 16-24 可以看出，页面中下部这个区域一共划分成了 4 个部分，每个部分都使用 div 容器浮动定位，准确设置每个 div 的宽度，这样才能够让各个部分显示在正确的位置，如图 16-25 所示。

（3）根据上面的实现方法，为每个区域的 div 的 id 或者 class 命名。

- 标题：.title。
- 数据列表：.detail_info。
- 图片按钮 1：.relative。
- 图片按钮 2：.function_channel。

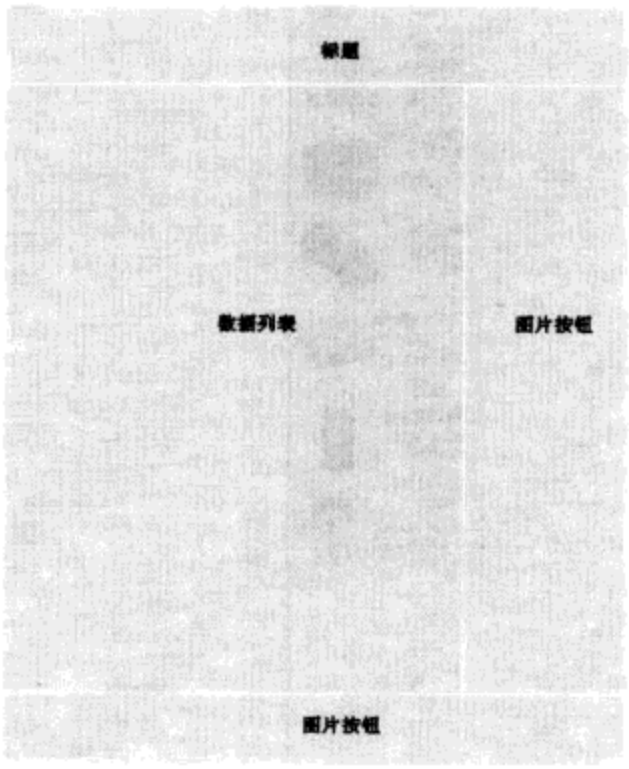


图 16-24　构思草图

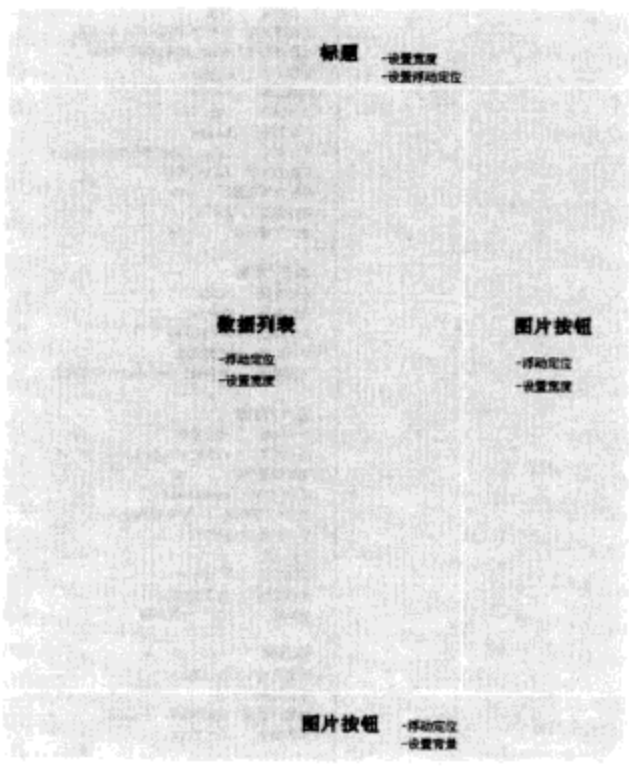


图 16-25　实现方法

2. 代码制作

（1）继续在框架代码的基础上，添加制作页面顶部的 XHTML 代码，找到如下代码。

```
<div id="info"></div>
```

然后在其中编写 XHTML 代码。

```
<div id="info">
    <div class="title"><img src="images/departline.jpg" /><img src="images/set3title.jpg"
/></div>
    <div class="detail_info">
    </div>
    <div class="relative"></div>
    <div class="clearit"></div>
    <div class="function_channel"></div>
</div>
```

在以上代码中，除了按照构思好的实现方法制作出来的结构代码，又多出来一对 div 标签。

```
<div class="clearit"></div>
```

在这里添加代码的原因是，前面的浮动定位元素要实现二列式布局，在它们之后还有浮动

元素，这样如果每个元素的宽度设置不够精确，或者浏览器对盒模型的解析不同，就会造成混乱的浮动定位的效果。为保证前面的两个元素能够按照设计实现二列式布局，这里就要添加一段内容为空的代码，通过 CSS 的设置，使后面的浮动元素以新的参照物开始浮动。

（2）现在实现每个部分的布局，添加 CSS 代码，代码如下。

```
.title {
    border-bottom: #E7E7E7 1px solid;                    /*设置下边框样式*/
    height: 80px;                                        /*设置高度*/
    margin-top: 14px;                                    /*设置上外边距*/
}
.title img {
    float: left;                                         /*设置向左浮动定位*/
}
.detail_info {
    color: #595959;                                      /*设置颜色*/
    float: left;                                         /*设置向左浮动定位*/
    width: 630px;                                        /*设置宽度*/
}
.relative {
    float: left;                                         /*设置向左浮动定位*/
    width: 170px;                                        /*设置宽度*/
}
.clearit {
    clear: both;                                         /*清除浮动样式*/
}
.function_channel {
    background: url(images/channelbg.jpg) no-repeat;     /*设置背景*/
    padding: 6px;                                        /*设置内边距*/
    height: 183px;                                       /*设置高度*/
    margin-top: 40px;                                    /*设置上外边距*/
}
```

在以上代码中，除了使用浮动定位属性和边距的设置为模块进行布局设计之外，重点是对选择符.clearit 的样式定义，这里使用了 clear:none 属性，禁止了元素左右两侧的浮动元素，使得.function_channel 元素不能影响数据列表和右侧按钮部分的二列式布局设计。页面中下部布局的预览效果如图 16-26 所示。

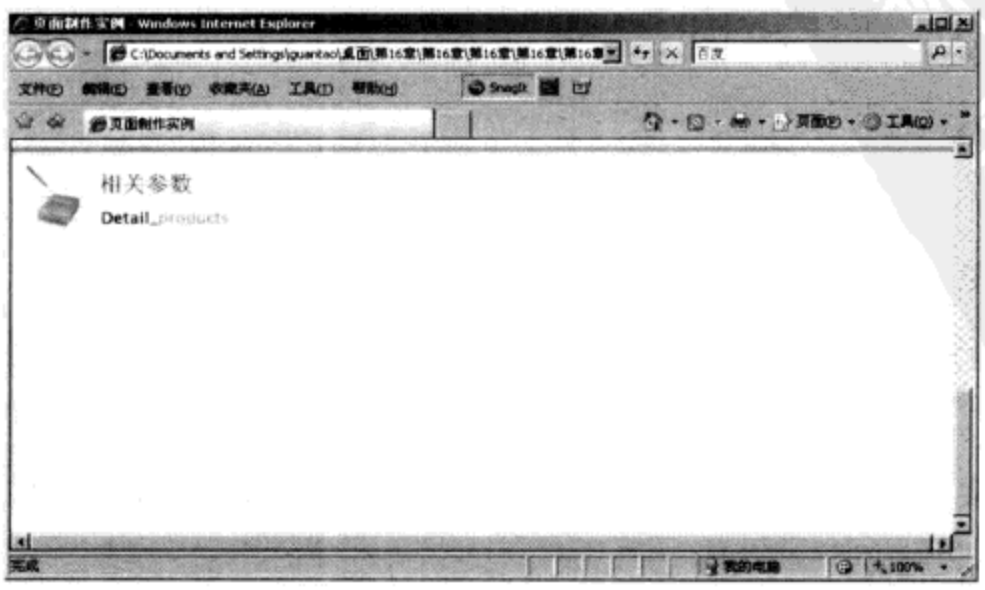


图 16-26　页面中下部布局的预览效果

16.2.4　页面底部的布局设计

完成了页面中下部的布局设计之后，接下来就可以对页面底部进行整体构思，以便按照步骤一步一步地实现。

1．页面分析

（1）观察页面底部的设计图，如图 16-27 所示。

图 16-27　页面底部的设计图

根据设计图勾勒一个草图，理清一下思路，如图 16-28 所示。这部分主体的结构可以看作是二列式的布局，左侧是 logo，右侧是导航、联系方式以及版权信息。右侧包含的 3 个部分，又可分为上、中、下 3 行。

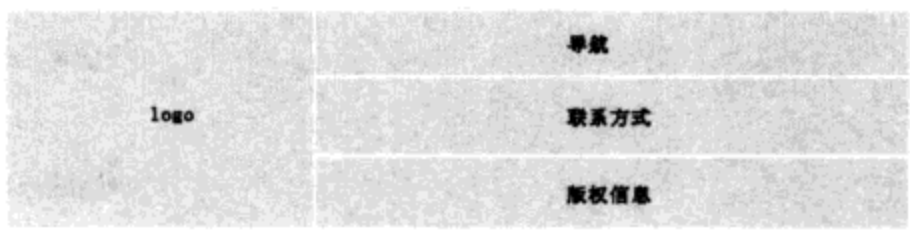


图 16-28　构思草图

（2）从图 16-28 可以看出，页面底部是左右两列的二列式布局，分别使用浮动定位实现布局设计。右列包含了 3 行，分别是导航、联系方式和版权信息，如图 16-29 所示。

图 16-29　实现方法

（3）最后，根据上面的实现方法，为每个区域的 div 的 id 或者 class 命名。

- logo：.logob。
- 右列：.copy。
- 导航：ul、li。
- 联系方式：p.contact。
- 版权信息：p。

2．代码制作

（1）继续在框架代码的基础上，添加制作页面顶部的 XHTML 代码，找到如下代码：

```
<div id="footer"></div>
```

然后在其中编写 XHTML 代码。

```
<div id="footer">
    <div class="logob"><img src="images/logo.jpg" /></div>
    <div class="copy">
        <ul>
        </ul>
        <p class="contact"> </p>
        <p>copyright 2000-2003 psdshop.com allright reserved</p>
    </div>
</div>
```

在以上代码中，导航部分使用了列表的形式来制作。“联系方式”和“版权信息”都是使用<p>标签进行文本排版，根据内容和样式的不同，为“联系方式”单独添加了特殊的 class。

（2）现在实现每个部分的布局，添加 CSS 代码，代码如下。

```
.logob {
    float: left;                                    /*设置向左浮动定位*/
    margin-left: 9px;                               /*设置左外边距*/
    margin-top: 60px;                               /*设置上外边距*/
}
.copy {
    float: right;                                   /*设置向右浮动定位*/
}
.copy ul {
    list-style-type: none;                          /*设置列表符号样式*/
    margin-top: 18px;
}
.copy p {
    text-align: right;                              /*设置文本右对齐*/
    line-height: 20px;                              /*设置行高*/
}
.copy p.contact {
    color: #B4B4B4;                                 /*设置文字颜色*/
}
```

在以上代码中，同样是使用浮动定位为每个模块实现布局定位，并且设置外边距以及宽度、高度，完成精确定位。对“版权信息”和“联系方式”的文本，使用 CSS 的 text-align 属性设置了右对齐，并且使用了行高属性 line-height 调整文本行与行之间的距离。页面底部布局的预览效果如图 16-30 所示。

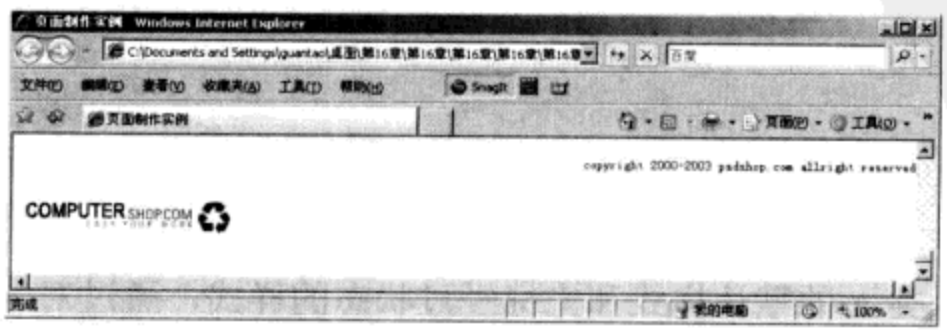


图 16-30　页面底部布局的预览效果

现在完成了整个页面的框架以及主体的布局设计，接下来就要进行每一个模块布局的制作。

16.3　模块设计

模块设计，就是对页面中的具体内容进行布局实现，例如导航的制作、数据列表的制作、图片按钮的排版、文本的排版等。本节将分别对网页中的每一个内容模块进行 CSS 布局设计。

16.3.1　快速导航制作

这个页面有两个位置使用了导航形式：页面顶部的右上角和页面底部的右上角。为了页面代码的多样化，针对这两个导航系统，使用了不同的实现方法。页面顶部右上角的导航使用了普通的文字链接并列的方法，整个导航系统放在 div 容器中控制位置。页面底部右上角的导航使用了列表的形式，导航项使用
标签来控制，整个导航由 ul 控制位置。

1．页面顶部的导航

首先来看一下页面顶部的导航设计效果，如图 16-31 所示。

站点地图 | 关于我们 | 收藏本页

图 16-31　页面顶部的导航

导航系统是文字链接，使用“|”分割线分开每个导航项。

（1）在完成的布局代码基础上，添加导航模块的具体 XHTML 代码，找到如下代码：

```
<div class="nav"></div>
```

在其中添加如下代码：

```
<div class="nav"><a href="#">站点地图</a>|<a href="#">关于我们</a>|<a href="#">收藏本页
</a></div>
```

在以上代码中，使用伪类 a 来控制导航项的文字链接，a 标签外面是分割线“|”，链接样式不影响分割线的样式，分割线的颜色属性继承 class 为 nav 的属性样式。

（2）为页面顶部的导航添加 CSS 样式，代码如下。

```
.nav a {
    color: #000000;                          /*设置文字颜色*/
    text-decoration: none;                   /*设置链接文字的样式*/
    margin-left: 16px;                       /*设置左外边距*/
    margin-right: 16px;                      /*设置右外边距*/
}
.nav a:hover {
    color:#FF0000;                           /*设置文字颜色*/
    text-decoration: underline;              /*设置链接文字鼠标经过时的样式*/
}
```

以上代码设置了伪类 a 的默认样式和光标经过状态的样式，通过对 a 标签左右外边距的设置，控制了导航项之间的距离。制作完成的导航在页面顶部的预览效果如图 16-32 所示。

图 16-32　页面顶部导航布局的预览效果

2. 页面底部的导航

观察页面底部的导航设计效果，如图 16-33 所示。

网站简介 | About Us | 广告服务 | 联系我们 | 招聘英才 | 客户投诉

图 16-33　页面底部导航的设计效果

同样，这里的导航系统也是使用分割线“|”分开，不同的是实现的方法，这个导航系统使用了列表。

（1）在页面底部的代码中，找到如下代码：

```
<div class="copy">
    <ul></ul>
</div>
```

在其
标签内添加导航的 XHTML 代码。

```
<ul>
    <li><a href="#">网站简介</a></li>
      |  
    <li><a href="#">About Us</a></li>
      |  
    <li><a href="#">广告服务</a></li>
      |  
    <li><a href="#">联系我们</a></li>
      |  
    <li><a href="#">招聘英才</a></li>
      |  
    <li><a href="#">客户服务</a></li>
</ul>
```

在以上代码中， 标签内是导航项，导航项使用文字链接，<li>标签外使用了分割线“|”。ul 列表标签的默认样式为纵向排列，所以现在要为其添加 CSS 样式，改变排列的方向为横向排列。

（2）为页面底部的导航添加 CSS 样式，代码如下。

```
.copy ul {
    list-style-type: none;                          /*设置列表符号的样式*/
    margin-top: 18px;                               /*设置上外边距*/
}
.copy ul li {
    display: inline;                                /*设置元素内联样式*/
}
.copy ul li a {
    color: #000000;                                 /*设置文字颜色*/
    text-decoration: none;                          /*设置链接文字样式*/
}
.copy ul li a:hover {
    color: #FF0000;                                 /*设置链接文字光标经过时的文字颜色*/
    text-decoration: underline;                     /*设置链接文字光标经过时的样式*/
}
```

在以上代码中，首先在 ul 选择符中，设置了列表符号为 none，去除了任何的符号样式，然后通过包含选择符，设置了 li 选择符的显示方式为 inline，实现横向排列，预览效果如图 16-34 所示。

图 16-34　页面底部导航布局的预览效果

16.3.2　“面包屑”制作

“面包屑”的制作方法在前面已经介绍过，核心的方法就是文字链接样式。在制作带有背景的“面包屑”时，需要注意的是设置好文本在背景图中的显示位置，通过 CSS 的行高属性，能够达到这一目的。本章实例中的“面包屑”如图 16-35 所示。

> 首页 > 联想ThinkPad T400

图 16-35　“面包屑”效果

“面包屑”中连接文字之间使用“>”分隔，表示子属关系，后者是前者的分支。

（1）在页面底部的代码中，找到如下代码：

```
<div class="crumbs"></div>
```

在其中添加 XHTML 代码。

```
<div class="crumbs">&gt; <a href="#">首页</a>&gt; <a href="#">联想 ThinkPad T400</a></div>
```

在以上代码中，使用了 HTML 的转义字符“>”表示“>”。

注意：

转义字符串（Escape Sequence）也称字符实体（Character Entity）。在 HTML 中，定义转义字符串的原因有两个：第一个原因是像“<”和“>”这类符号已经用来表示 HTML 标签，因此就不能直接当作文本中的符号来使用。为了在 HTML 文档中使用这些符号，就需要定义它的转义字符串。当解释程序遇到这类字符串时就把它解释为真实的字符。在输入转义字符串时，要严格遵守字母大小写的规则。第二个原因是，有些字符在 ASCII 字符集中没有定义，因此需要使用转义字符串来表示。表 16-1 是几个常用的转义字符串。

表 16-1　常用转义字符串

HTML 源代码	显示结果	描述
&lt;	<	小于号或显示标记
&gt;	>	大于号或显示标记
&amp;	&	可用于显示其他特殊字符
&quot;	"	引号
&reg;	®	已注册
&copy;	©	版权
&trade;	™	商标
		一个空白位
		不断行的空白

（2）为“面包屑”的文字链接添加 CSS 样式，代码如下。

```
.crumbs a {
    color: #000000;                          /*设置文字颜色*/
    text-decoration: none;                   /*设置链接文字样式*/
    margin-left: 5px;                        /*设置左外边距*/
}
.crumbs a:hover {
    color: #FF0000;                          /*设置链接文字光标经过时的文字颜色*/
    text-decoration: underline;              /*设置链接文字光标经过时的样式*/
}
```

在以上代码中，通过对伪类 a 的样式设置，实现“面包屑”的定位和交互效果，预览效果如图 16-36 所示。

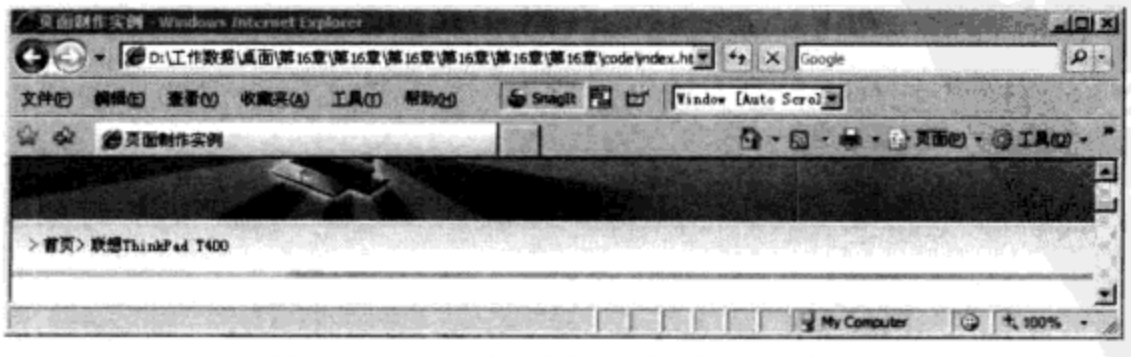


图 16-36　“面包屑”制作的预览效果

16.3.3　数据表格制作

数据表格在本章所使用的实例中多处用到。在页面中上部分别使用了有背景的数据表格、无背景的数据表格，在页面中下部使用了若干个数据表格。

1．有背景的数据表格

有背景的数据表格与无背景的数据表格的区别就是一个背景图片，背景图片的实现就是通过 CSS 定义一张背景图。

（1）在页面中上部的代码区域，找到如下代码：

```
<div class="bgtable"> </div>
```

在其中添加数据表格的 XHTML 代码。

```
<div class="bgtable">
    <table>
        <tr>
            <th>产品价格</th>
            <td>27500 元</td>
        </tr>
        <tr>
            <th>上市时间</th>
            <td>2008 年 9 月</td>
        </tr>
        <tr>
            <th>处理器系</th>
            <td>英特尔 酷睿 2 双核 T9 系列</td>
        </tr>
    </table>
</div>
<div class="notable">
    <table>
        <tr>
            <th>标称主频</th>
            <td>2.8GHz</td>
        </tr>
        <tr>
            <th>平台技术</th>
            <td>Intel 迅驰 2 平台 Montevina</td>
        </tr>
        <tr>
            <th>屏幕尺寸</th>
            <td>14.1 英寸</td>
        </tr>
    </table>
</div>
```

在以上代码中，使用了前面讲解的制作数据表格所使用的标签，数据表格放在了一个 class 为 bgtable 的 div 容器中，目的有两个：一是控制表格的位置；二是设置背景。

（2）为带有背景的数据表格添加 CSS 样式，代码如下。

```
.bgtable {
    background: url(images/table_bg.jpg) repeat-x;          /*设置背景样式*/
    height:110px;                                           /*设置高度*/
}
.bgtable table{
    margin-left: 10px;                                      /*设置左外边距*/
    margin-top: 25px;                                       /*设置上外边距*/
}
.bgtable table th{
    font-style: normal;                                     /*设置字体样式*/
    color: #898989;                                         /*设置文字颜色*/
    width: 115px;                                           /*设置宽度*/
    height: 18px;                                           /*设置高度*/
    text-align: left;                                       /*设置文本左对齐*/
}
```

在以上代码中，使用 CSS 的背景属性设置了一个背景图片，并且采用平铺样式，设置容器的高度，以保证背景图片的显示。使用包含选择符定义表格的外边距，实现表格的定位等，预览效果如图 16-37 所示。

图 16-37　带背景的数据表格预览效果

2. 无背景的数据表格

掌握了制作带背景的数据表格方法后，制作无背景的数据表格就易如反掌，即不设置背景。现在以制作页面中上部的无背景的数据表格为例进行讲解。

（1）在页面中上部的代码中，找到如下代码：

```
<div class="notable"> </div>
```

在其中添加 XHTML 代码。

```
<div class="notable">
    <table>
        <tr>
            <th>标称主频</th>
            <td>2.8GHz</td>
        </tr>
        <tr>
            <th>平台技术</th>
            <td>Intel 迅驰 2 平台 Montevina</td>
        </tr>
        <tr>
            <th>屏幕尺寸</th>
```

```
                <td>14.1 英寸</td>
            </tr>
        </table>
    </div>
```

在以上代码中，代码结构与带背景的数据表格相似，只是这里的表格放在了 class 为 notable 的 div 容器中，这样就可以对其设置不同于带背景的数据表格的 CSS 样式。

（2）为无背景的数据表格添加 CSS 样式，代码如下。

```
.notable table {
    margin-left: 10px;                          /*设置左外边距*/
    margin-top: 25px;                           /*设置上外边距*/
}
.notable table th {
    font-style: normal;                         /*设置字体样式*/
    color: #898989;                             /*设置文字颜色*/
    width: 115px;                               /*设置宽度*/
    height: 18px;                               /*设置高度*/
    text-align: left;                           /*设置文本左对齐*/
}
.notable {
    height: 110px;                              /*设置高度*/
}
```

在以上代码中，并没有为 div 容器设置背景图片，而只是设置了一个高度，能够保证表格数据的显示。其余的表格内的标签样式与带背景的数据表格相同，预览效果如图 16-38 所示。

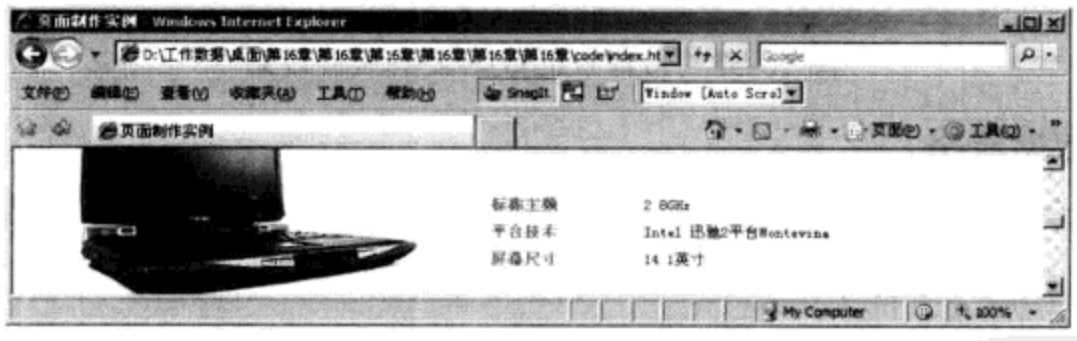


图 16-38　无背景的数据表格布局的预览效果

对于有相同 CSS 样式的数据表格，为了优化代码，建议把相同的 CSS 代码合并在一个组合选择符中。这里的.bgtable table{},bgtable table th{}与.notable table{},notable table th{}就可以合并在一起。

16.3.4　图片排版

图片排版的内容在图文排版的章节中讲解过，主要方法就是使用 CSS 的浮动定位属性布局图片位置。本章实例的图片主要用来制作按钮，排版的方式依然是浮动定位，涉及的图片排版主要在页面中下部的右栏和最下面一行区域，如图 16-39 和图 16-40 所示。

（1）按照页面设计的效果图，制作 XHTML 结构代码，找到如下代码：

```
<div class="relative"></div>
<div class="function_channel"></div>
```

图 16-39　右栏图片排版

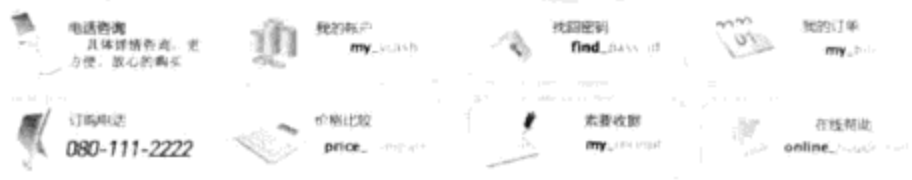


图 16-40　最下面一行的图片排版

分别在其中添加 XHTML 代码。

```
<div class="relative">
    <a href="#"><img src="images/sideBtn01.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn02.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn03.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn04.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn05.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn06.jpg" border="0" /></a>
    <a href="#"><img src="images/sideBtn07.jpg" border="0" /></a>
</div>
<div class="function_channel">
    <a href="#"><img src="images/functionBtn01.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn02.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn03.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn04.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn05.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn06.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn07.jpg" border="0" hspace="4" vspace="3" /></a>
    <a href="#"><img src="images/functionBtn08.jpg" border="0" hspace="4" vspace="3" /></a>
</div>
```

在以上代码中，由于图片设置了链接样式，所以在图片标签<img>中，设置边框为 0，去除由于链接样式为图片以外增加的边框效果。并且在<img>标签中，使用 hspace 属性和 vspace 属性设置了图片水平间距和垂直间距。

（2）为图片链接添加 CSS 样式，代码如下。

```
.relative a {
    float: left;                                    /*设置向左浮动定位*/
    margin-bottom: 3px;                             /*设置下外边距*/
}
```

```
.function_channel a{
    float: left;                                    /*设置向左浮动定位*/
}
```

在以上代码中，通过伪类 a 的样式定义，实现对图片的位置控制。在浏览器中的预览效果如图 16-41 和图 16-42 所示。

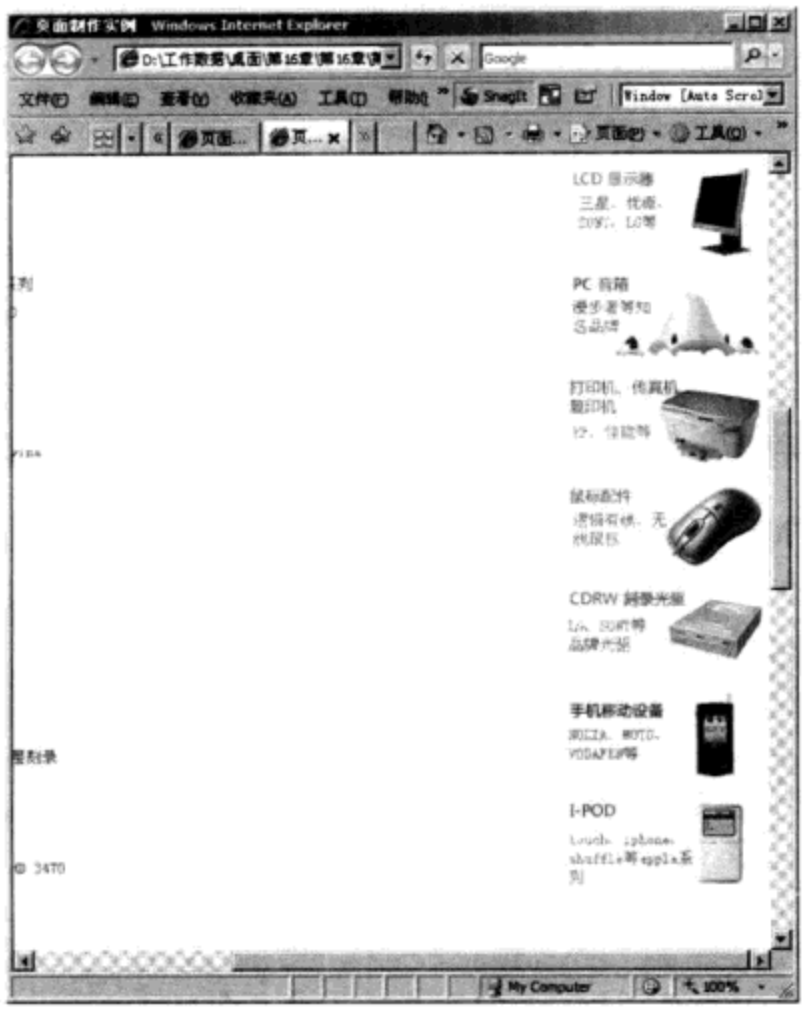


图 16-41　右栏的图片排版布局的预览效果

图 16-42　最下面的图片排版布局的预览效果

现在，整个页面就全部制作完成了，在 IE 7 浏览器中的预览效果和设计图没有差别。

16.4　兼容性测试

为了符合 Web 标准，制作完成的页面至少需要在 3 个主流浏览器（IE 7、Firefox、IE 6）中测试。由于本书所有实例都是使用 IE 7 作为默认浏览器预览的，所以现在需要在其他两个浏览

器中进行测试。

1．在 Firefox 浏览器中测试

把网页文件放在 Firefox 浏览器中打开，预览效果如图 16-43 所示。

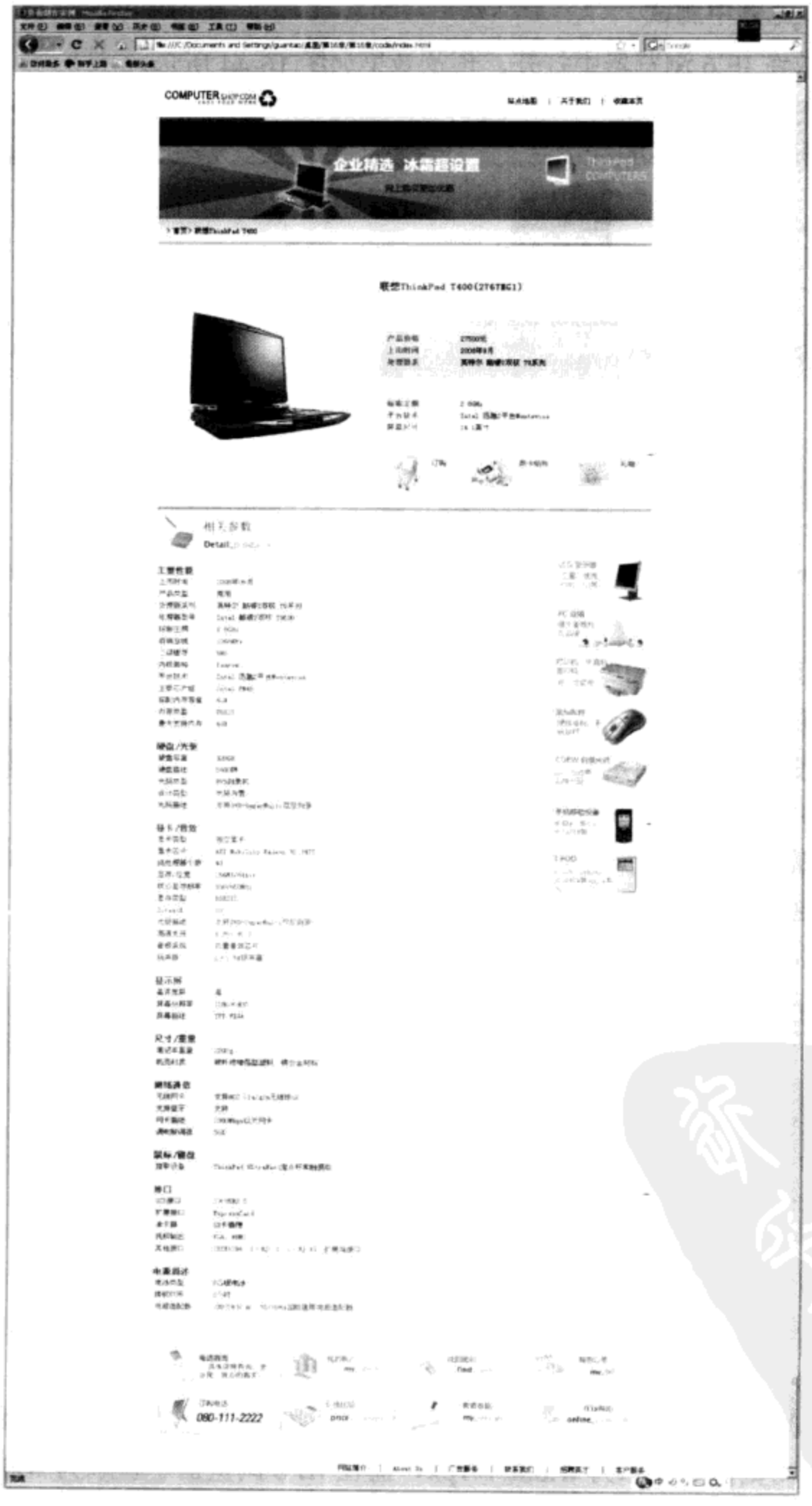


图 16-43　Firefox 浏览器中的预览效果

从图 16-43 可以看出，页面在 Firefox 浏览器中显示正常，证明代码既能兼容 IE 7，又能兼容 Firefox 浏览器。能够在 Firefox 这样标准的浏览器下正确显示，也说明了本章实例制作的代码是符合 Web 标准的。

2. 在 IE 6 浏览器中测试

把网页文件放在 IE 6 浏览器中打开，预览效果如图 16-44 所示。

图 16-44　IE 6 浏览器中的预览效果

从图 16-44 可以看出，在 IE 6 浏览器中显示会出现问题，页面中上部的二列式布局没有正确实现，而是左右两列错位分成两行显示。分析问题的原因，可以很容易想到盒模型的问题。由于二列式布局使用的是浮动定位方式，为元素设置固定宽度，如果作为浮动定位的两个元素的宽度总和大于页面的宽度，就会出现现在这样的错位问题。所以出现如图 16-44 所示的问题的原因，可以判断为 IE 6 浏览器对于之前设置的左右两列元素的宽度解析与 IE 7、Firefox 的不一样，下面按照这个线索解决问题。

修改选择符.intro_info 内的 CSS 样式代码。

```
.intro_info {
    float: left;                                    /*设置向左浮动定位*/
    margin-left: 38px;                              /*设置左外边距*/
    width: 441px;                                   /*设置宽度*/
    *width: 441px!important;
    *width: 401px;
}
```

在以上代码中，针对属性及属性值 width:441px 进行了修改，第一次的 width:441px 是 3 种浏览器都可以解析的代码，第二次的 width: 441px 是只有 IE 可以解析的代码，第三次的 width:401px 就是只能用 IE 6 解析了。

根据不同浏览器对特殊代码的支持不同，解决了浏览器的兼容问题，现在这段 CSS 代码在 IE 7 以及 Firefox 浏览器中显示为 widht:441px，而在 IE 6 浏览器中显示为 width:401px，预览效果如图 16-45 所示。

图 16-45　修改后 IE 6 浏览器中的预览效果

16.5　小结

本章主要介绍了使用 Dreamweaver 进行网页制作的流程和方法，主要步骤包括：框架设计、布局设计、模块设计以及兼容性检查。在制作过程中，需要掌握的一个重要方法就是切图。

对本章的知识点的前后联系进行归纳总结，结构导图如图 16-46 所示。

图 16-46　本章知识点结构导图

16.6　习题

1．使用 Dreamweaver 制作页面主要从哪几个方面进行设计？

2．怎样使用 Photoshop 软件来切图？尝试选择一个源文件进行切图，并存储。

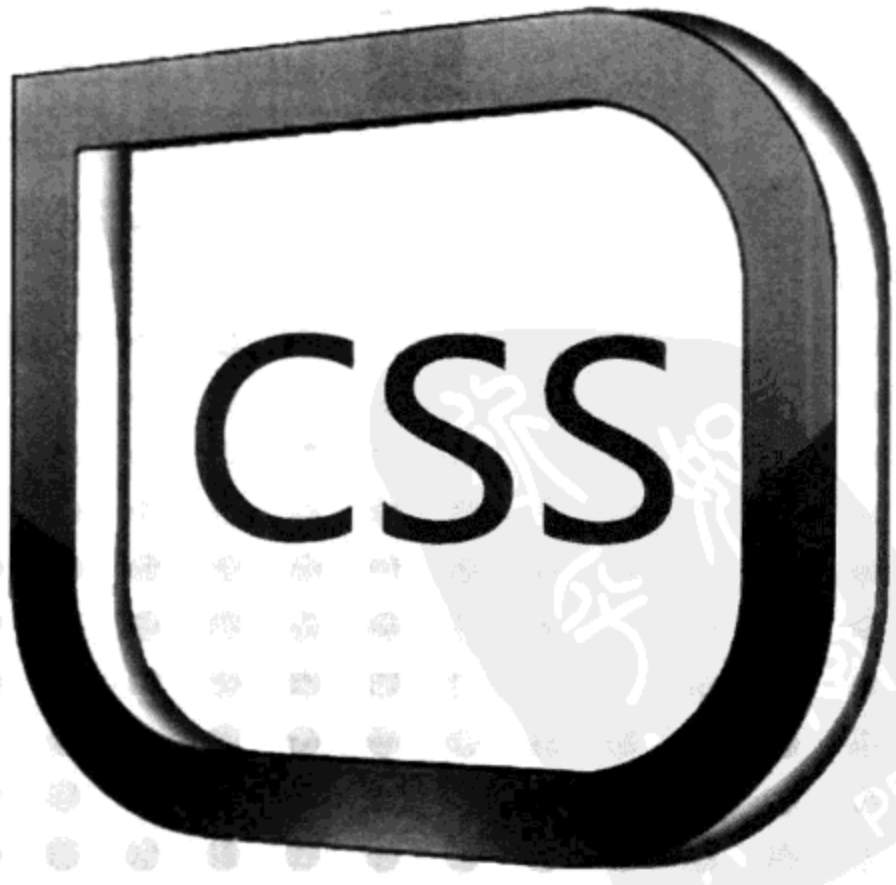
3．制作一张带有背景的数据表格。

4．在页面底部区域设计一行简单的图片排版样式。

第四篇　综合应用篇

第 17 章　博客类网页布局设计

第 18 章　企业类网页布局设计



第17章 博客类网页布局设计

博客（blog）是最近几年非常流行的网络日志。写博客的目的，是为了在里面表达观点、传播信息、存放心情。博客真正吸引人的地方，是里面的文章，而不是华丽的界面。博客类的网页，归根到底是为了让用户能够顺利地浏览你的东西，从而达到沟通与展现的目的。所以，博客页面的设计应该遵循这样一个理念：形式为内容服务。突出日志功能，界面略加修饰即可。

本章将以一博客网站为例（如图 17-1 所示），详细介绍博客类网站的页面规划与制作。

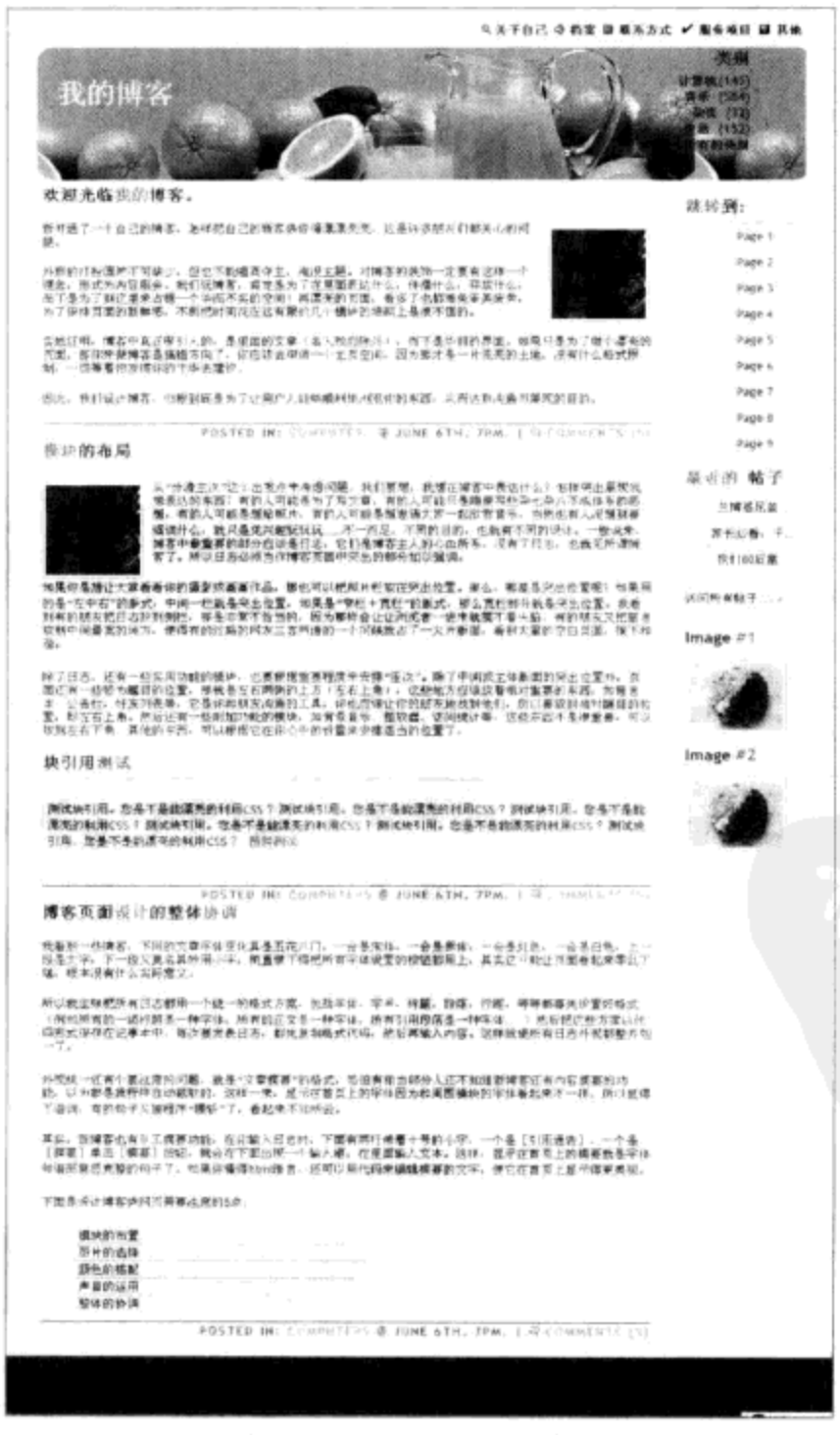


图 17-1 博客网站界面

17.1　页面布局和规划

首先要明确博客类网站的目标，完成网站的构思创意即总体设计方案，主要包括界面设计、色彩设计和规划页面布局。

17.1.1　界面设计分析

界面设计是网站设计的重要环节，而在 CSS 布局的网站中更为重要。在进行 CSS 布局设计时，要在设计中更多地考虑到后期 CSS 布局上的可用性。

如果只是一个页面，那么界面上对 CSS 布局的考虑，更多是考虑 CSS 代码的重用，使编写的样式达到最大的使用率，使得多个版块能够经常共享这些代码，以减少开发的时间。

CSS 布局设计倾向于统一的网站界面与布局设计，能够提高网站统一性。在统一的基础上，也需要进一步考虑每部分内容的特色，对于网站来说，各频道应该具有相同的布局模板，各个频道在使用相同模板的基础上，在细节处体现差异，这也体现了在最终的 CSS 代码设计中具有继承性，各频道继承统一的样式而又有自己的特色，最终完成一个良好的界面设计。

17.1.2　规划页面布局

现在根据设计图来规划一下页面的布局，仔细分析一下首页的界面设计图，不难发现，图片大致分为以下几个部分，如图 17-2 所示。

从图 17-2 可以看出，把网页分成了 5 个大部分，每个部分代表的内容分别如下。

- 顶部 navigation 部分，包括了网站的导航菜单。
- banner 部分，包括网站的主题图片以及文章分类菜单。
- content 部分，是博客首页的主体部分，包括日志的标题、内容。
- side 部分，主要放置了博客的其他相关模块，包括分页功能、最新日志以及相册功能。
- 底部 footer 部分，包括一些版权信息。

根据图 17-2，再画一个实际的 CSS 布局的页面布局图，表明层的嵌套关系，这样理解起来就会更简单明了，如图 17-3 所示。

div 层级结构关系如下。

```
└# body {}
└#container {}                    /*页面层容器*/
└#navigation {}                   /*页面头部*/
└#banner {}                       /*页面 banner*/
└#content{}                       /*页面主体*/
└#sidebar{}                       /*主体内容*/
└#footer {}                       /*页面底部*/
```

至此，页面布局与规划已经完成，接下来要做的就是开始编写 XHTML 代码和 CSS 样式。

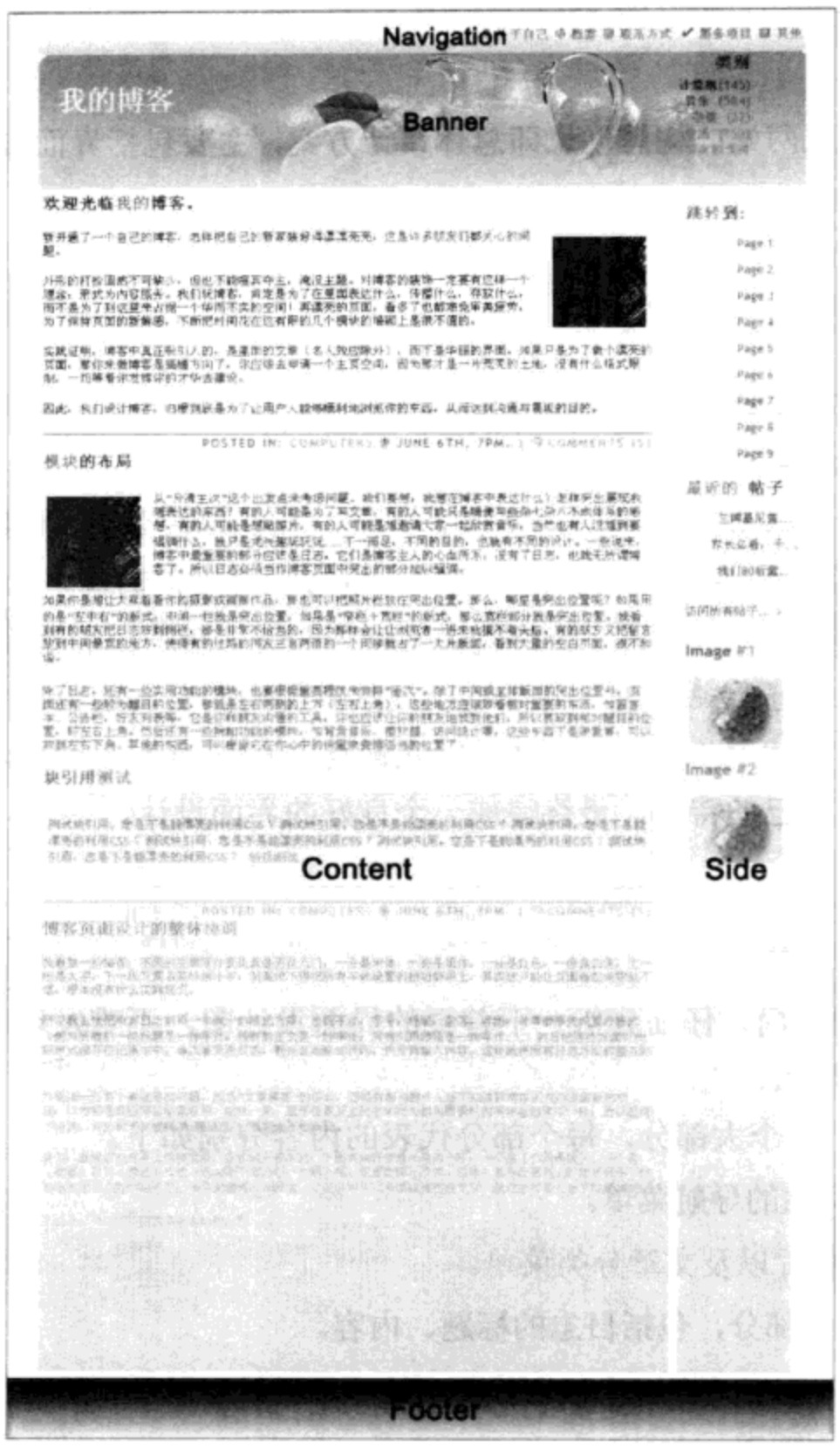


图 17-2　博客首页规划图

图 17-3　页面布局图

17.2　CSS 结构设计与整体布局设计

页面布局确定下来后，接下来就应该进行页面制作。在页面制作的过程中，首先要进行 CSS 的结构设计，规划 CSS 样式的文件结构。

17.2.1　CSS 结构设计

CSS 布局的特点，是结构、表现、行为的相互分离，这样做带来的好处之一便是浏览速度

的提升。CSS 的结构设计包括 3 个步骤：链接外部 CSS、CSS 结构设计、CSS 命名。

1. 链接外部 CSS

CSS 实现了表现与内容的分离，然而在开始为页面去除混乱的样式代码的同时，也引入了另一个新的问题——CSS 文件日益庞大。这种情况会导致网站出现两个致命的问题：①初次载入网站速度慢；②维护困难。

为了解决这样的问题，CSS 还提供了链接外部文件功能和导入命令。对于本章示例网站的 CSS 样式来说，根据页面的种类，编写一套外部的 CSS 文件，命名为 default.css，通过 XHTML 代码的链接功能，链接到每个具体页面当中，具体 XHTML 代码如下。

```
<link rel="stylesheet" href="default.css" type="text/css" />
```

可以从外部导入样式表，从而实现 CSS 层面的需求并且服务于自己的优秀代码结构，以便于网站的维护。

2. CSS 结构设计

对于大型网站的样式设计而言，可以通过样式与结构的分离技术，链接外部 CSS 文件，让样式各自写在独立的文件之中。CSS 文件的结构优化可以根据网站的需求依照一定的原则进行设计，可以在 CSS 编码之初就定下基本框架，良好的 CSS 结构设计往往使网站的设计事半功倍。

本章讲解的博客网站的页面，是根据页面上各元素的功能及网站内容进行样式划分的。本实例博客页面可以分为以下几个部分。

- 导航栏：用来放置导航系统。
- banner：用来放置网页的主题图片。
- 顶部栏目：用来显示日志文章的分类信息以及该分类日志的数目。
- 正文内容区：用来放置日志正文内容。
- 右侧边栏：用来放置分页项等相关栏目。
- 页面尾部：用来放置版权信息等内容。

划分好了 CSS 结构可以根据这些结构，就可为每部分的 CSS 样式进行分组命名。

3. CSS 命名

按照功能划分出来的每一部分，CSS 选择符通常以类来命名。常用 CSS 类的命名应尽量以常见英文单词为准，做到通俗易懂，并在适当的地方加以注释，一般命名规则如下。

- 页眉：header。
- 内容：content。
- 容器：container。
- 页脚：footer。
- 版权：copyright。
- 导航：menu。

- 主导航：mainMenu。
- 子导航：subMenu。
- 标志：logo。
- 标语：banner。
- 标题：title。
- 侧边栏：sidebar。

按照这些命名规范为页面的 CSS 样式表命名，就可使 CSS 代码得到很大程度的优化。

17.2.2 整体布局设计

博客页面的首页是这类网站的核心，它的布局好坏决定了网站的成败。本节讲解使用 XHTML 与 CSS 制作首页布局的方法。

1. 制作页面主体 div 结构

根据页面规划出来的 div 层级关系，编写首页主体 XHTML 代码。

```
<div id="container">
    <div id="nav">[导航系统]</div>
    <div id="banner">
        <h1>[标题]</h1>
        <div id="topbar">[顶部栏目]</div>
    </div>
    <div id="content">[日志正文内容]</div>
    <div id="sidebar">[分页等相关栏目]</div>
</div>
<div id="footer">[网站版权信息、联系方式等]</div>
```

2. 编写主体结构的 CSS 样式

编写好结构代码后，在样式表中，为结构编写 CSS 样式代码。

（1）制作 body 标签元素的 CSS 样式，代码如下。

```
#html, body {
    font: 13px trebuchet ms;                    /*设置字体样式*/
    margin: 0;                                  /*设置外边距*/
    padding: 0;                                 /*设置内边距*/
    color: #666;                                /*设置文字颜色*/
    height: 100%;                               /*设置高度*/
}
```

在以上代码中，通过标签指定式选择符 body 设置了页面的字体样式、内外边距、文字颜色以及页面的高度，特别是将页面的高度设置成了 100%，目的是实现页面自适应高度。

（2）制作主容器 div 的 CSS 样式，代码如下。

```
#container {
    background: #fff;                                    /*设置背景颜色*/
    width: 820px;                                        /*设置宽度*/
    margin: 10px auto;                                   /*设置外边距，实现对象居中*/
    padding: 2px;                                        /*设置内边距*/
}
```

在以上代码中，通过类选择符#container 设置了装载网页容器的背景色、宽度以及内外边距，这里再次设置外边距 margin: 10px auto，是应用了一列固定宽度居中的 CSS 基本布局类型。

（3）制作 banner 区域的 CSS 样式，代码如下。

```
#banner {
    width: 819px;                                        /*设置宽度*/
    background: url('images/bgb.jpg') top left no-repeat;  /*设置背景样式*/
    margin: 0 0 3px 0;                                   /*设置外边距*/
    position: relative;                                  /*设置相对定位*/
    height: 150px;                                       /*设置高度*/
    color: #fff;                                         /*设置文字颜色*/
}
```

在以上代码中，使用类选择符#banner 为网页标语部分设置了宽度、背景图片、外边距、文字颜色以及定位方式，背景图片如图 17-4 所示。

图 17-4　banner 的背景图片

这里设置宽度和高度的作用，就是保证背景图片完整显示。

（4）制作导航系统的 CSS 样式，代码如下。

```
#nav {
    background: #fff;                                    /*设置背景色*/
    width: 810px;                                        /*设置宽度*/
    text-align: right;                                   /*设置文本右对齐*/
    padding: 5px;                                        /*设置内边距*/
    margin: 0 0 5px 0;                                   /*设置外边距*/
}
```

在以上代码中，使用类选择符#nav 设置了导航系统的背景色、导航宽度、内外边距以及文本的对齐方式。重要的是文本对齐方式，通过 text-align 属性，设置了导航文本右对齐。

（5）制作日志正文主体结构的 CSS 样式，代码如下。

```
#content {
    height: 100%;                                        /*设置高度，百分之百，实现高度自适应*/
    float: left;                                         /*设置向左浮动定位*/
    background: #fff;                                    /*设置背景色*/
    padding: 5px;                                        /*设置内边距*/
```

```
    width: 650px;                                    /*设置宽度*/
    margin-bottom: 20px;                             /*设置下外边距*/
}
```

在以上代码中，使用类选择符#content 设置了日志正文区域采用浮动定位方式，向左浮动定位。还通过设置 height: 100%的属性，并且在前面已经在 body 选择符中设置的 height:100%的情况下，实现日志正文区域的高度自适应。这是博客网页普遍采用的方法。同时还设置了这部分的宽度为 650px，小于总宽度，留出一定宽度放置边栏内容。

（6）制作右边栏区域的 CSS 样式，代码如下。

```
#sidebar {
    float: right;                                    /*设置向右浮动定位*/
    width: 130px;                                    /*设置宽度*/
    border-left: 1px dotted #ccc;                    /*设置左边框样式*/
    padding: 5px 0 0 10px;                           /*设置内边距*/
    margin: 10px 0 10px 15px;                        /*设置外边距*/
}
```

在以上代码中，使用类选择符#sidebar 设置了边栏定位方式使用浮动定位，并且浮动方式为向右浮动定位，同时设置了它的宽度为 130px，与之前日志正文的宽度 650px 之和为 780px，再加上设置的边框宽度，总共小于页面的总宽度 820px，所以边栏和日志正文两部分分成两列显示在一排中，实现二列固定宽度的基本布局。

（7）制作顶部栏目区域的 CSS 样式，代码如下。

```
#topbar {
    text-align: right;                               /*设置文本右对齐*/
    color: #666;                                     /*设置文字颜色*/
    padding: 5px;                                    /*设置内边距*/
    margin-right: 55px;                              /*设置右外边距*/
}
```

在以上代码中，使用类选择符#topbar 设置了 banner 区域的顶部栏目样式，这里主要放置的是日志文章的分类以及分类文章数目，为这里的分类文本设置了向右对齐，并且设置了文字颜色以及边距。

（8）制作页面尾部的 CSS 样式，代码如下。

```
#footer {
    clear: both;                                     /*禁止周围的浮动对象出现*/
    background: url('images/blackbg.gif') repeat;    /*设置背景样式*/
    border-top: 4px solid #666;                      /*设置上边框样式*/
    text-align: center;                              /*设置文本居中对齐*/
    padding-bottom: 5px;                             /*设置下内边距*/
}
```

在以上代码中，使用类选择符#footer 设置了页面尾部样式，使用 clear:both 属性保证了<div #footer>容器独立占据一行，禁止两侧任何浮动元素。这样，这个页面的主体布局设计就完成了。下面分别为每一部分进行深入的布局设计。

17.3　页面头部布局设计

博客页面的头部包括导航系统、banner 以及 banner 区域的顶部栏目。下面就为这 3 个部分逐一添加内容以及样式。

17.3.1　制作头部的结构代码

在整体布局代码的基础上，首先开始为页面头部进行局部的具体内容的布局设计，XHTML 结构代码如下。

```
<div id="nav">
    <img src="images/about.gif" alt=""/><a href="#">关于自己</a>
    <img src="images/products.gif" alt=""/><a href="#">档案</a>
    <img src="images/contact.gif" alt=""/><a href="#">联系方式</a>
    <img src="images/services.gif" alt=""/><a href="#">服务项目</a>
    <img src="images/help.gif" alt=""/><a href="#">其他</a>
</div>
<div id="banner">
    <h1>我的博客</h1>
    <div id="topbar">
        <h3>类别</h3>
        <ul>
            <li><a href="#">计算机(145)</a></li>
            <li><a href="#">音乐 (584)</a></li>
            <li><a href="#">杂谈 (32)</a></li>
            <li><a href="#">生活 (152)</a></li>
            <li><a href="#">所有的类别</a></li>
        </ul>
    </div>
</div>
```

在以上代码中，在 容器内添加了导航系统，在 容器内添加了 banner 以及顶部栏目模块的结构代码，预览效果如图 17-5 所示。

图 17-5　添加头部的结构代码预览效果

17.3.2　编写头部的 CSS 代码

头部的结构在没有添加任何样式前，都是按照标签默认的样式呈现在浏览器中，现在为这些结构添加 CSS 样式。

（1）制作导航条的 CSS 代码，代码如下。

```
#nav a {
    color: #666;                                    /*设置文字颜色*/
    font-weight: bold;                              /*设置粗体*/
    text-decoration: none;                          /*设置文本修饰样式*/
}
#nav a:hover {
    text-decoration: underline;                     /*设置光标经过时的文本样式*/
}
```

在以上代码中，通过对伪类 a 的样式设置，来完成超级链接文字的导航，预览效果如图 17-6 所示。

图 17-6　添加导航样式的预览效果

（2）制作 banner 的 CSS 代码，代码如下。

```
#banner h1 {
    position: absolute;                             /*设置绝对定位*/
    letter-spacing: -2px;                           /*设置字间距*/
    font-size: 32px;                                /*设置字号*/
    top: 35px;                                      /*设置上边界距离*/
    left: 20px;                                     /*设置左边界距离*/
}
```

在以上代码中，使用包含选择符为 banner 区域中的网页标题设置了字体的样式以及定位的方式，定位方式使用绝对定位。也就是说，无论页面怎么变化，网页标题的位置始终都会在距离 banner 容器的顶部 35px、距离 banner 容器的左边 20px 的位置，并且它的存在将不再影响其他元素的定位，如图 17-7 所示。

从图 17-7 可以看出，banner 的标题和顶部栏目两个部分上下堆叠在了一起。

（3）制作顶部栏目的 CSS 代码，代码如下。

```
#topbar a {
    color: #87af7b;                          /*设置文字颜色*/
    text-decoration: none;                   /*设置文本修饰*/
    color: #666;                             /*设置文字颜色*/
    font-weight: bold;                       /*设置粗体*/
}
#topbar a:hover {
    text-decoration: underline;              /*设置文本修饰样式*/
}
#topbar h3 {
    margin: 0 0 3px 0;                       /*设置外边距*/
    padding: 0;                              /*设置内边距*/
}
#topbar ul {
    margin: 0;                               /*设置外边距*/
    padding: 0;                              /*设置内边距*/
}
#topbar li {
    list-style: none;                        /*设置列表符号样式*/
    width: auto;                             /*设置宽度*/
}
```

在以上代码中，设置了 ul 列表元素的边距为 0px，解决盒模型的兼容问题。同时设置了顶部栏目的标题样式、分类信息链接文字的样式等，预览效果如图 17-8 所示。

图 17-7　添加 banner 标题样式的预览效果

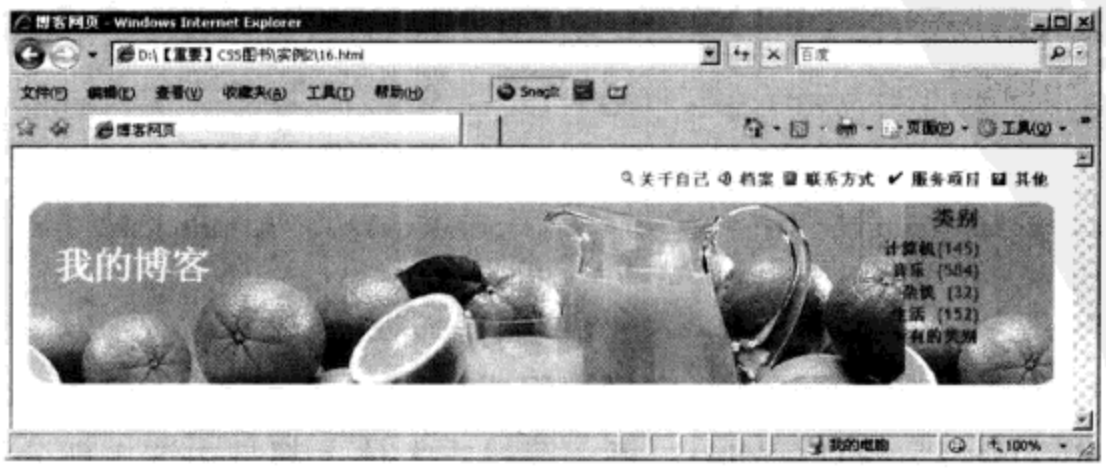


图 17-8　添加顶部栏目样式的预览效果

17.4　日志部分的布局设计

日志部分是博客类网站的重要组成部分，博客的主要功能就是供博主写日志，日志就像日记本一样，要能够满足写东西、发图片等一些功能，博客的价值才能够体现。本节讲解如何对博客日志部分进行布局设计。

17.4.1　制作日志部分的结构代码

完成了页面头部布局设计之后，接下来为博客首页的日志部分进行布局设计，XHTML 结构代码如下。

```
    <div id="content">
        <h1 class="headline"> 欢迎光临<strong>我的</strong>博客。 </h1>
        <p>  <img  class="imageright"  src="images/questionmark.jpg"  width="100"  height="100"
alt=""/> 新开通了一个自己的博客，怎样把自己的新家装修得漂漂亮亮，这是许多朋友们都关心的问题。<br />
        <br />
        外形的打扮固然不可缺少，……<br />
    ……
        <div class="meta"> posted in: <a href="#">computers</a> @ june 6th, 7pm. | <img src= "images
/comment.gif" alt=""/> <a href="#">comments (5)</a> </div>
        <h1 class="headline"> <strong>模块</strong>的布局 </h1>
        <p>  <img  class="imageleft"  src="images/questionmark.jpg"  width="100"  height="100"
alt=""/> 从“分清主次”这个出发点来考虑……</p>
        <p>如果你是想让大家看看你的摄影或画画作品，……</p>
        <p> 除了日志，还有一些实用……</p>
        <h1 class="headline"> 块引用<strong>测试</strong> </h1>
        <blockquote>
            <p> 测试块引用。您是……<a href="#">链接测试</a> </p>
        </blockquote>
        <div class="meta"> posted in: <a href="#">computers</a> @ june 6th, 7pm. | <img src= "images
/comment.gif" alt=""/> <a href="#">comments (5)</a> </div>
        <h1 class="headline"> 博客页面<strong>设计</strong>的整体<strong>协调</strong> </h1>
        <p> 我看到一些博客，……</p>
        <p>所以我主张把所有日志都用一个统一的格式方案，……</p>
        ……
        <ul>
            <li>模块的布置</li>
            <li>图片的选择</li>
            ……
        </ul>
        <div class="meta"> posted in: <a href="#">computers</a> @ june 6th, 7pm. | <img src="images
/comment.gif" alt=""/> <a href="#">comments (5)</a> </div>
    </div>
```

在以上代码中，日志的部分主要由 3 个模块组成，即日志标题<h1></h1>、日志正文 、日志相关信息栏 ，每一篇日志大体上都是由这 3 部分组成，预览效果如图 17-9 所示。

图 17-9　日志正文部分结构的预览效果

17.4.2　编写日志部分的 CSS 代码

按照制作步骤，接下来为日志部分的结构添加样式。

（1）制作日志标题的 CSS 代码，代码如下。

```
.headline {
    font-size: 18px;                                    /*设置字号*/
    margin: 3px 0 3px 0;                                /*设置外边距*/
}
strong {
    color: #87af7b;                                     /*设置文字颜色*/
}
```

在以上代码中，设置了标题的字体样式和外边距，使之与上下文之间保持一定距离，同时对 strong 标签设置了特殊的颜色和背景样式，这里的背景使用了继承属性，继承了#content 选择符的属性，即无背景，预览效果如图 17-10 所示。

图 17-10　添加日志标题样式的预览效果

（2）制作日志正文部分的相关元素的 CSS 代码，代码如下。

```
.imageright {
    float: right;                                    /*设置向右浮动定位*/
    border: 1px solid #ccc;                          /*设置边框样式*/
    padding: 3px;                                    /*设置内边距*/
    margin: 3px 3px 0 7px;                           /*设置外边距*/
}
.imageright:hover {
    border: 1px solid #666;                          /*设置边框样式*/
}
.imageleft {
    float: left;                                     /*设置向左浮动定位*/
    border: 1px solid #ccc;                          /*设置边框样式*/
    padding: 3px;                                    /*设置内边距*/
    margin: 3px 10px 0 0px;                          /*设置外边距*/
}
.imageleft:hover {
    border: 1px solid #666;                          /*设置边框样式*/
}
#content a {
    color: #87af7b;                                  /*设置文字颜色*/
    text-decoration: none;                           /*设置文本修饰样式*/
}
#content a:hover {
    text-decoration: underline;                      /*设置链接文字交互时的文本修饰样式*/
}
```

在以上代码中，为日志部分中的图片设置了样式，为左侧显示的图片设置了向左浮动定位属性，为右侧显示的图片设置了向右浮动定位属性，同时使用伪类:hover 为图片的交互状态设置了样式。另外，还为日志的链接文字设置了样式，预览效果如图 17-11 所示。

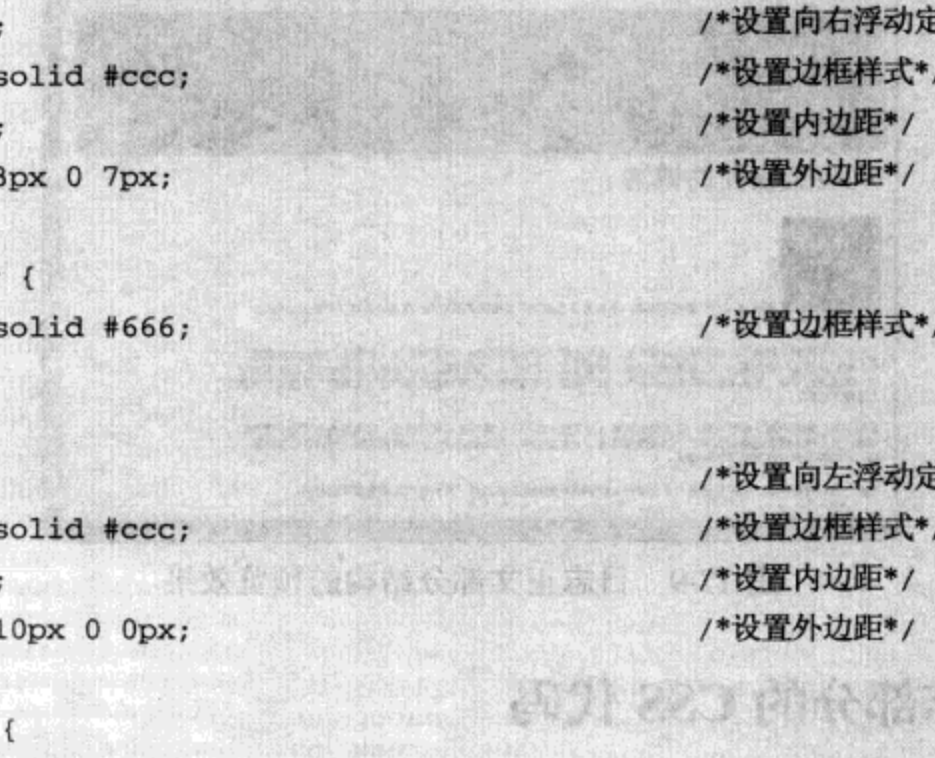


图 17-11　添加日志正文样式的预览效果

（3）制作日志正文部分的附加部分的 CSS 代码，代码如下。

```
#content ul {
    margin: 5px 0 10px 30px;                    /*设置外边距*/
    padding: 0;                                 /*设置内边距*/
    border-top: 1px solid #ccc;                 /*设置上边框样式*/
    width: 320px;                               /*设置宽度*/
}
#content li {
    display: block;                             /*设置块级元素显示方式*/
    text-align: left;                           /*设置文本左对齐*/
    margin: 0;                                  /*设置外边距*/
    padding: 0 0 0 10px;                        /*设置内边距*/
    list-style-type: none;                      /*设置列表项目符号样式*/
    border-bottom: 1px solid #ccc;              /*设置下边框样式*/
}
#content li:hover {
    background: #deebd2;                        /*设置背景色*/
}
blockquote {
    background: url('images/blockbg.jpg') repeat-x;    /*设置背景样式*/
    color: #666;                                /*设置文字颜色*/
    padding: 5px;                               /*设置内边距*/
    border: 1px solid #ccc;                     /*设置边框样式*/
    margin: 5px 0 15px 0px;                     /*设置外边距*/
}
```

在以上代码中，为日志正文中一些其他附件的文本格式设置了特殊的样式，例如 ul 列表、引用文本等格式，预览效果如图 17-12 所示。

图 17-12　添加了日志特殊文本样式的预览效果

（4）制作日志相关信息栏的 CSS 代码，代码如下。

```
.meta {
    background: url('images/bg.gif');                    /*设置背景样式*/
    border-top: 1px solid #666;                          /*设置上边框样式*/
    text-align: right;                                   /*设置文本右对齐*/
    color: #666;                                         /*设置文字颜色*/
    text-transform: uppercase;                           /*设置小写字母转换为大写字母*/
    letter-spacing: 2px;                                 /*设置字符间距*/
}
```

在以上代码中，为日志正文的相关信息栏设置了背景图片、边框、文本对齐样式，并且设置了小写英文字母转换为大写的属性等，背景图片使用默认的平铺方式，背景图片如图 17-13 所示。

页面预览效果如图 17-14 所示。

图 17-13 信息栏的背景图片

图 17-14 添加信息栏样式的预览效果

17.5 边栏区域的布局设计

边栏在博客页面中一般用来放置博主信息、相关博文、相册等一些版块，本章制作的博客页面在边栏部分主要放置了分页的功能、最新博文以及相册 3 个功能模块。

17.5.1 制作边栏区域的结构代码

日志部分的右侧便是博客首页的边栏部分，现在为边栏区域进行布局设计，XHTML 代码如下。

```
<div id="sidebar">
    <h1 class="headline"><strong>跳转</strong>到:</h1>
    <ul>
        <li><a href="#">Page 1</a></li>
        <li><a href="#">Page 2</a></li>
        <li><a href="#">Page 3</a></li>
        <li><a href="#">Page 4</a></li>
```

```
            <li><a href="#">Page 5</a></li>
            <li><a href="#">Page 6</a></li>
            <li><a href="#">Page 7</a></li>
            <li><a href="#">Page 8</a></li>
            <li><a href="#">Page 9</a></li>
        </ul>
        <h1 class="headline"><strong>最近的</strong> 帖子</h1>
        <ul>
            <li><a href="#">兰博基尼盖...</a></li>
            <li><a href="#">家长必看：千...</a></li>
            <li><a href="#">我们80后童...</a></li>
        </ul>
        <a href="#">访问所有帖子... ></a><br />
        <h1 class="headline">Image <strong>#1</strong></h1>
        <img src="images/no.jpg" alt="Picture" class="sideimage"/>
        <h1 class="headline">Image <strong>#2</strong></h1>
        <img src="images/no.jpg" alt="Picture" class="sideimage"/>
    </div>
```

在以上代码中，使用 h1 和 ul 标签组合制作了 3 个部分的结构，每对 h1 和 ul 标签构成一个模块，预览效果如图 17-15 所示。

图 17-15　制作右边栏的预览效果

17.5.2　编写右边栏区域的 CSS 代码

（1）制作超级链接的 CSS 代码，代码如下。

```
#sidebar a {
    display: block;                          /*设置块级元素显示方式*/
    color: #666;                             /*设置文字颜色*/
    text-decoration: none;                   /*设置文本修饰样式*/
    padding: 5px 0 5px 0;                    /*设置内边距*/
```

```
}
#sidebar a:hover {
    background: #deebd2;                                    /*设置背景色*/
    color: #666;                                            /*设置文字颜色*/
}
```

在以上代码中，为右边栏的超级链接文字设置了样式，并且将超级链接设置为块状元素，每个链接独立占据一行，独立显示，预览效果如图 17-16 所示。

图 17-16　添加右边栏超级链接文字样式的预览效果

（2）制作列表项的 CSS 代码，代码如下。

```
#sidebar ul {
    margin: 5px 0 15px 20px;                                /*设置外边距*/
    padding: 0;                                             /*设置内边距*/
    border-top: 1px solid #ccc;                             /*设置上边框样式*/
}
#sidebar li {
    margin: 0;                                              /*设置外边距*/
    padding: 0;                                             /*设置内边距*/
    text-align: center;                                     /*设置文本居中*/
    list-style-type: none;                                  /*设置列表符号样式*/
    border-bottom: 1px solid #ccc;                          /*设置下边框样式*/
}
```

在以上代码中，为 ul 列表项设置了样式，取消了 ul、li 标签默认的列表项样式，重新添加了分隔线，预览效果如图 17-17 所示。

（3）制作相册的 CSS 代码，代码如下。

```
.sideimage {
    border: 1px solid #ccc;                                 /*设置边框样式*/
    padding: 3px;                                           /*设置内边距*/
    margin: 10px 0 5px 0;                                   /*设置外边距*/
}
```

```
.sideimage:hover {
    border: 1px solid #666;                                    /*设置边框样式*/
}
```

在以上代码中，为相册模块设置了边距和边框样式，同时设置了当光标经过相册图片时图片呈现的样式，预览效果如图 17-18 所示。

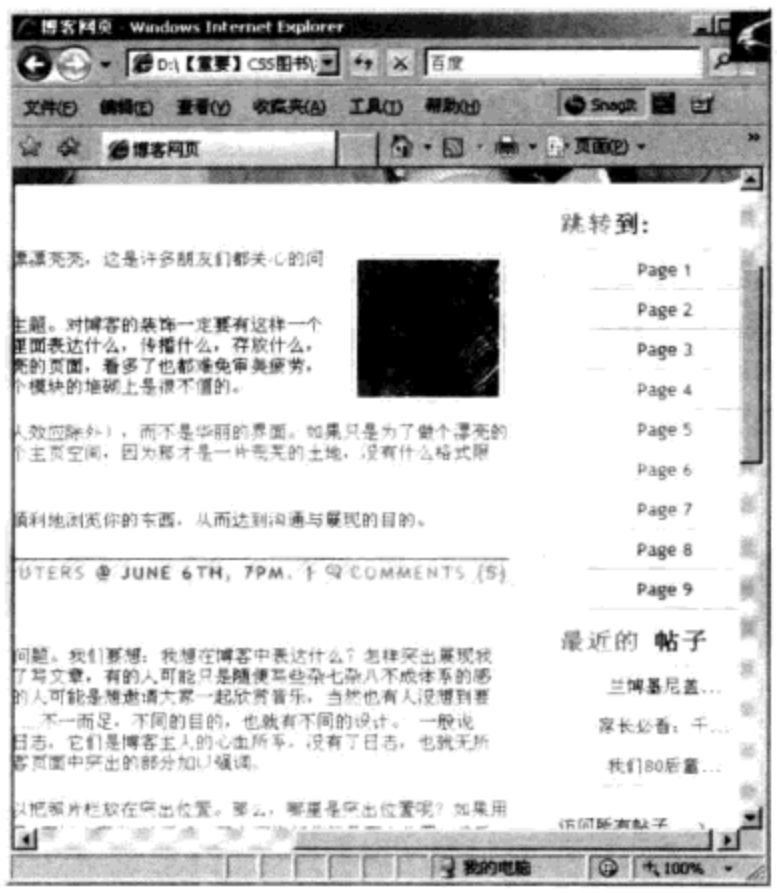


图 17-17　添加了列表项样式的预览效果

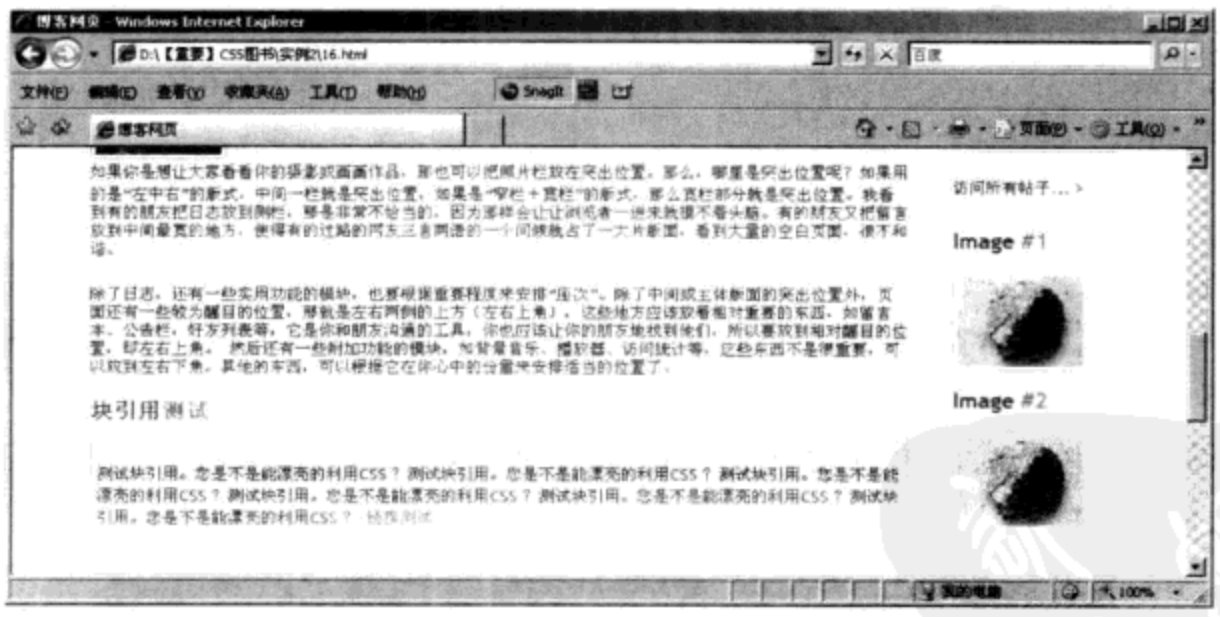


图 17-18　添加相册样式的预览效果

17.6　页面底部的布局设计

网页尾部一般包括版权信息、联系方式、关于网站的介绍等文字链接。博客网页也不例外，页面尾部一般也放置这些内容。

17.6.1　制作底部的结构代码

在完成了以上所有的区域布局设计以后，最后还剩下页面底部的区域，现在进行底部的布局设计，XHTML 代码如下。

```
    <div id="footer">
        <p>copyright (&copy;) (you) | <a href="#">Valid XHTML 1.1</a> design by: <a
href="#">G_T</a> </p>
    </div>
```

在以上代码中，放置了版权信息以及网站的设计者，预览效果如图 17-19 所示。

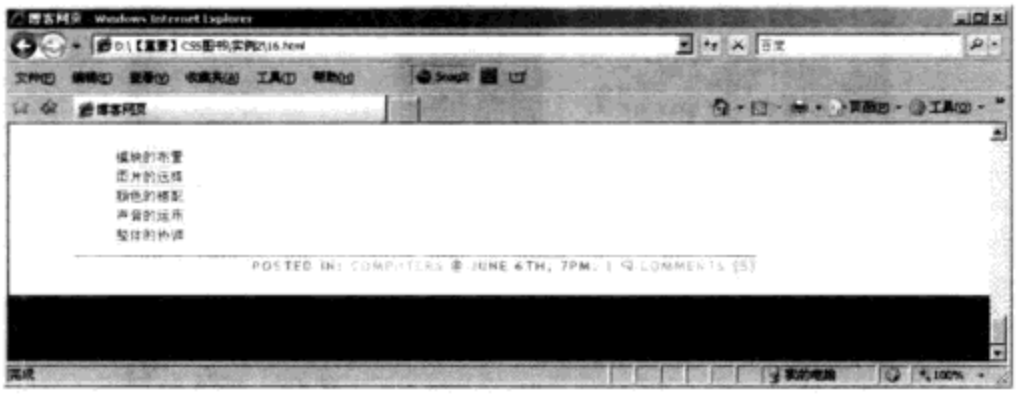


图 17-19　没有添加任何样式的网页尾部

17.6.2　编写底部的 CSS 代码

现在为底部添加 CSS 样式，代码如下。

```
#footer a {
    color: #666;                                    /*设置文字颜色*/
    font-weight: bold;                              /*设置粗体*/
    text-decoration: none;                          /*设置文本修饰样式*/
}
#footer a:hover {
    text-decoration: underline;                     /*设置光标经过时的文本修饰样式*/
}
```

在以上代码中，主要就是通过设置超级链接文字的样式，来完成网页底部的布局设计，预览效果如图 17-20 所示。

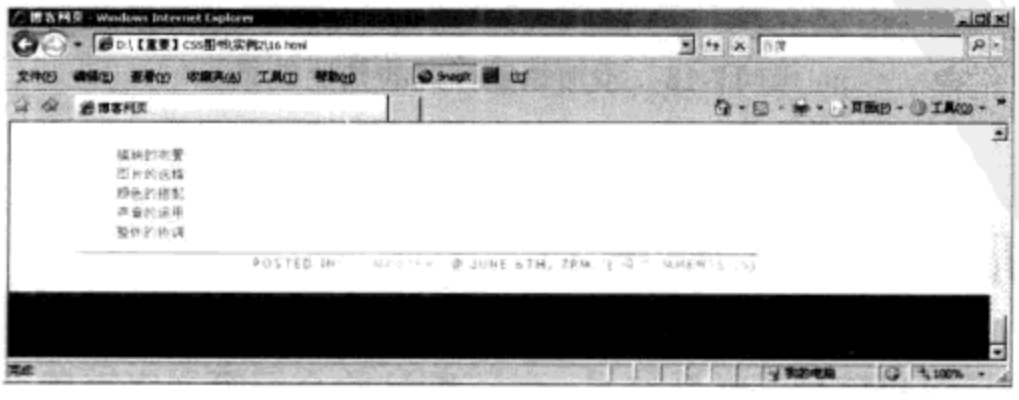


图 17-20　添加底部样式的预览效果

这样，整个博客网页的布局设计就完成了。

17.7　小结

本章主要介绍制作博客类网页的流程。

（1）页面布局与规划。

（2）CSS 结构设计。

（3）整体布局设计。

（4）页面头部布局设计。

（5）日志部分布局设计。

（6）边栏区域的布局设计。

（7）页面底部的布局设计。

对本章知识点的前后联系进行归纳总结，结构导图如图 17-21 所示。

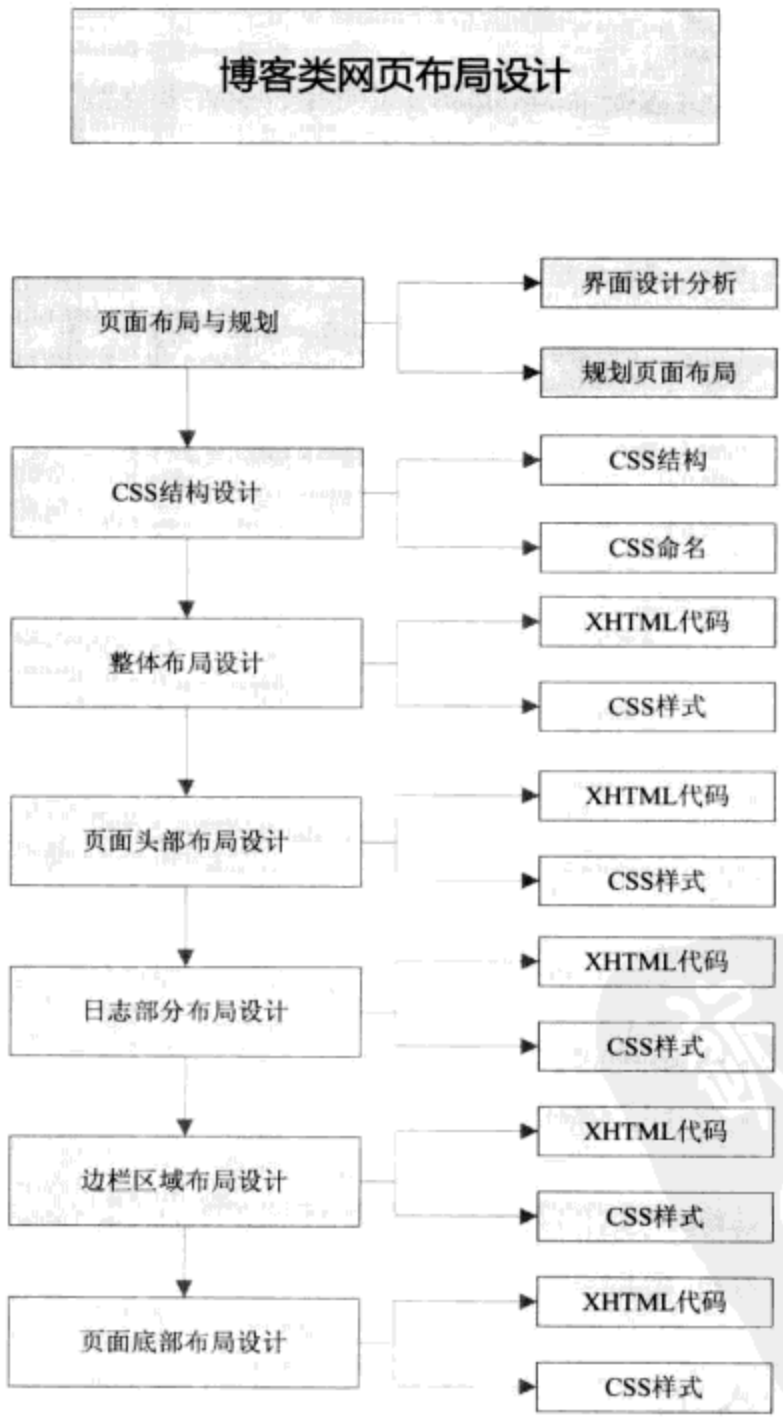


图 17-21　本章知识点结构导图

第18章 企业类网页布局设计

随着社会信息量的与日俱增，不论个人还是部门、企业都需要使用方便而有效的方式来宣传自己。为了便于企业能实时更新公司产品信息和更快地了解客户的需求，需要有效的企业网站设计来管理和宣传自己。本章将以一企业网站为例（如图 18-1 所示），详细介绍企业类网站的页面规划与制作。

图 18-1　企业类网站界面

18.1　页面布局和规划

首先要明确制作企业类网站的目标，完成网站的构思创意即总体设计方案，主要包括界面设计、色彩设计和规划页面布局。

18.1.1　界面设计分析

界面设计是网站设计的重要环节，在 CSS 布局的网站中更为重要。在 CSS 布局设计之中，要更多地考虑到后期布局上的可用性。

界面上对 CSS 布局的考虑，更多是考虑 CSS 代码的重用，使编写的样式达到最大的使用率，使得多个版块、多个页面能够经常共享这些代码，以减少开发的时间，也是为了代码的优化。

不仅要追求样式的重用性，也要对页面中的元素进行统一思考。多个频道在同一种类型内容的展示上，可以使用同一种方式。使用者只需要习惯统一阅读方式，便可以方便地从网站获取自己需要的信息。

CSS 布局设计倾向于统一的网站界面与布局设计，能够提高网站统一性。在统一的基础上，也需要进一步考虑每部分内容的特色，对网站来说，各频道应该具有相同的布局模板，各个频道在使用相同模板的基础上，在细节处体现差异，这也体现了在最终的 CSS 代码设计中具有继承性，各频道继承统一的样式而又有自己的特色，最终完成一个良好的界面设计，如图 18-2 所示。

图 18-2　首页与内容页的布局比较

分析网站上的两个页面，能找到一些共同点和变化。除去顶部的设计之外，在内容区的设计上，包含上面的两个页面在内，其他内容页也都是二列式布局。从布局角度上，二列式布局中右栏的宽度和样式完全相同。而对于二列式布局中的左栏以及更细节的部分，在大的方面外观上保持相同的样式，细节上根据各内容页的设计采用不同的布局样式。这样便形成了可以重用的统一样式表，并且在各内容页的设计上又可以进一步进行布局与样式上的扩展。

在各内容页的细节样式设计上，使用了不同的布局样式，如图 18-3 所示。

图 18-3　内容页布局的细节不同

18.1.2　规划页面布局

现在根据设计图来规划一下页面的布局，仔细分析一下首页的界面设计图，不难发现，图片大致分为以下几个部分，如图 18-4 所示。

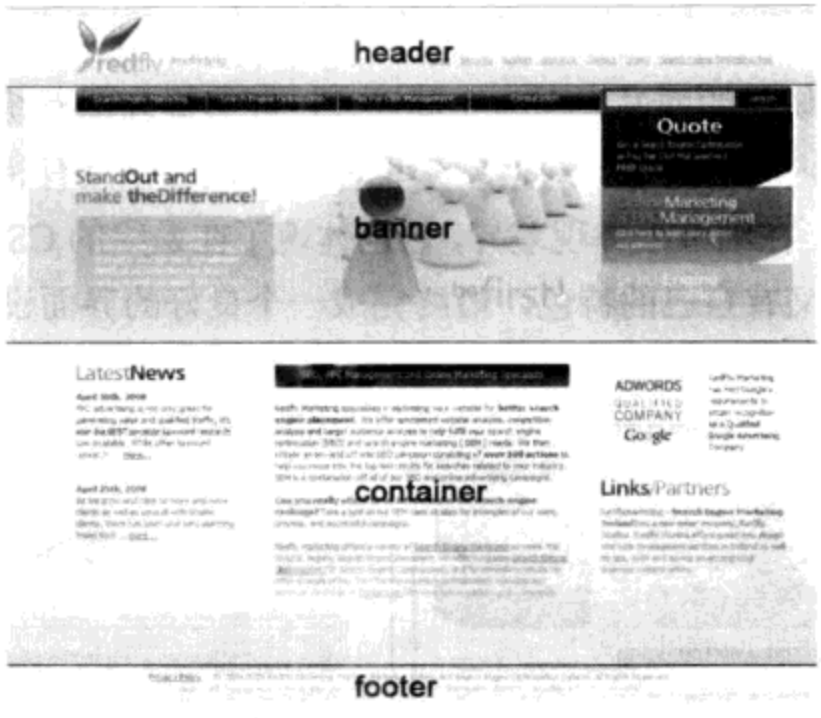


图 18-4　首页规划图

从图 18-4 可以看出，把网页分成了 4 个大部分，每部分代表的内容分别如下。

- 顶部 header 部分，其中又包括了网站的 logo、导航菜单。
- banner 部分，包括网站的主题图片、快速通道等。
- 内容 container 部分，又可分为主内容区、右栏。
- 底部 footer 部分，包括一些版权信息。

根据图 18-4，再画一个实际的 CSS 布局的页面布局图，表明层的嵌套关系，这样理解起来就会更简单明了，如图 18-5 所示。

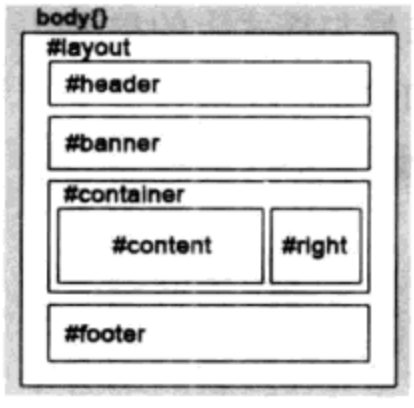


图 18-5　页面布局图

div 层级结构关系结构如下。

```
└# body {}
└#layout {}                    /*页面层容器*/
└#header {}                    /*页面头部*/
└#banner {}                    /*页面 banner*/
└#container {}                 /*页面主体*/
└#content {}                   /*主体内容*/
└#right {}                     /*右边栏*/
└#Footer {}                    /*页面底部*/
```

至此，页面布局与规划已经完成，接下来要做的就是开始编写 XHTML 代码和 CSS 样式。

18.2　CSS 结构设计与整体布局设计

页面布局确定下来后，接下来就应该进行页面制作。在页面制作的过程中，首先要进行 CSS

的结构设计，规划 CSS 样式文件结构。

18.2.1　CSS 文件结构设计

CSS 文件结构在大型网站的建设中起到相当大的作用，一个企业的网站，也需要有一套严格划分与使用的 CSS 文件。

对 redflymarketing.com 网站的 CSS 样式来说，根据页面的种类，首先编写一套全站共用的 CSS 文件，命名为 main.css，通过 XHTML 代码的链接功能，链接到每个页面当中，具体 XHTML 代码如下。

```
<link rel="stylesheet" href="/css/main.css" type="text/css" />
```

main.css 文件包含了全站统一的分栏样式、导航样式、文本样式以及每个页面都具有的功能，例如搜索区、文字链接。

除此之外，main.css 还可以通过 import 功能导入其他的样式文件，例如字体样式、导航样式、页脚样式等，代码如下。

```
@import url{font.css};
@import url{nav.css};
@import url{footer.css};
```

在以上代码中，font.css 用于控制全站所有的字体样式，nav.css 用于控制导航的样式，footer.css 用于控制页脚的样式。通过 main.css 文件，使每一个页面都具有了相同的一套样式表。

然而，对于首页以及每个内容页在设计上又有一定区别。在设计之中除了每个频道都具有 main.css 中所包含的样式表之外，首页还有自己的 home.css 文件，用于控制首页中的一些特殊版式。以此类推，每个内容页上根据页面的用途单独拥有自己的一套 CSS 样式。

18.2.2　首页布局设计

作为网站核心的首页，是网站浏览量最大的页面，也是一个复杂的引导用户走向的导向页，结构设计的好坏直接影响二级频道的访问量。

1. 制作首页主体 div 结构

根据 div 层级关系，编写首页主体 XHTML 代码。

```
<div id="layout">
    <div id="header">[页面头部]
    </div>
    <div id="container">
        <div id="content">[主体内容]
        </div>
        <div id="right">[右边栏]
        </div>
```

```
    </div>
    <div id="banner-container">[页面banner]
    </div>
    <div id="footer">[页脚]</div>
</div>
```

2. 编写整体 CSS 样式

（1）制作 body 标签元素的 CSS 样式，代码如下。

```
body {
    margin: 0;                          /*设置外边距*/
    padding: 0;                         /*设置内边距*/
    font-family: Tahoma;                /*设置字体类型*/
    font-size: 11px;                    /*设置字号*/
    line-height: 15px;                  /*设置行高*/
    text-align: center;                 /*设置文本居中*/
    color: #666666;                     /*设置文字颜色*/
}
```

在以上代码中，使用 body 标签选择符定义了整个页面的统一样式，去掉了外边距和内边距，指定了字体类型、字号、文字颜色等属性。

（2）制作主容器 div 的 CSS 样式，代码如下。

```
#layout {
    width: 970px;                       /*设置宽度*/
    margin: 0 auto;                     /*设置外边距*/
    text-align: left;                   /*设置文本左对齐*/
}
```

在以上代码中，#layout 是页面内容区的主容器 div，采用了宽度 970px，采用外边距使其在浏览器里面居中显示，文本对象左对齐。

（3）制作顶部区域的 CSS 样式，代码如下。

```
#header {
    clear: both;                        /*禁止对象左右出现浮动元素*/
    height: 100px;                      /*设置高度*/
}
```

在以上代码中，指定了顶部的高度，div 对象自身的块级元素特性，可以占据一整行显示。采用 clear 属性，指定了头部 div 区域的两侧禁止浮动对象出现，也就是说#header 元素将占据页面布局的一整行显示。

（4）制作 banner 区域的 CSS 样式，代码如下。

```
#banner{
    position:absolute;                  /*设置绝对定位*/
    top: 100px;                         /*设置上边界距离*/
    width: 970px;                       /*设置宽度*/
    height: 335px;                      /*设置高度*/
```

```
    overflow:hidden;                                    /*设置内容溢出指定区域时，自动剪切*/
    color: white;                                       /*设置文字颜色*/
    clear: both;                                        /*禁止对象左右出现浮动元素*/
}
```

在以上代码中，使用绝对定位属性 position，将放置 banner 部分 div 容器定位在距浏览器顶部 100px 的位置，并且位置不依赖于任何其他的对象，这个位置正好是#header 区域的高度，使得显示在#header 元素的下面。

定义了 div 容器的宽度和高度，指定了其内部的颜色，并且当内部对象的大小超过 banner 容器的宽高时，使用 overflow 属性定义不显示超出的内容。同样，clear 属性设置了 banner 容器两侧禁止浮动对象。

（5）制作主体内容区域的 CSS 样式，代码如下。

```
#container {
    padding-top: 335px;                                 /*设置上内边距*/
    margin-left: auto;                                  /*设置左外边距*/
    margin-right: auto;                                 /*设置右外边距*/
}
```

在以上代码中，设置了主题内容区域的左右外边距都是 auto，自动等宽，使这个区域同头部和 banner 部分一样，都是居中显示。

需要特别说明的是，这里实现#container 元素的定位的方法，不是采用 div 的浮动定位，也不是采用 position 属性的绝对定位，而是使用 div 的上内边距 335px，使得主内容区能够显示在#header 和#banner 两个 div 之下。

（6）制作页脚的 CSS 样式，代码如下。

```
#footer {
    width:970px;                                        /*设置宽度*/
    clear:both;                                         /*禁止对象左右出现浮动元素*/
    height:39px;                                        /*设置高度*/
    background: url('images/bg_footer.png');            /*设置背景图片*/
    color: #999999;                                     /*设置文字颜色*/
    text-align:center;                                  /*设置文本居中*/
    position:relative;                                  /*设置相对定位*/
    padding-top: 7px;                                   /*设置上内边距*/
    margin-top:-50px;                                   /*设置上外边距*/
}
```

在以上代码中，指定了页脚的宽度和高度，保证了背景图片的完整显示。为了布局美观，采用 text-align 属性居中对齐了文本。clear 属性禁止了页脚的 div 两侧出现浮动对象。

这里关键还在定位问题上，使用了相对定位 position: relative 属性。由于#banner 元素采用了绝对定位，不受#layout 的区域控制，所以页脚区域的上一个 div 块状元素就是#container 的 div，这样页脚就相对于#container 元素定位在其下面。

18.3 页面头部布局设计

整体页面的结构设计出来后，开始逐一地对每一部分进行细致的布局设计。企业网站页面的头部包括 logo 以及导航系统两个部分。下面就为这两个部分逐一添加内容以及样式。

18.3.1 制作头部的结构代码

首先，从页面的头部开始。观察设计图，分析页面头部的结构，由两个部分组成：logo 和导航系统，如图 18-6 所示。

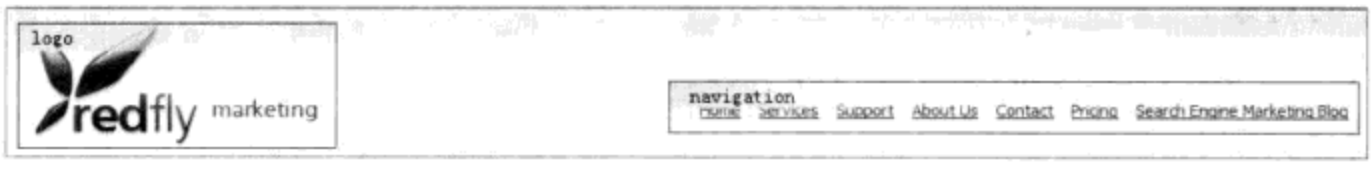


图 18-6 页面头部结构分析图

分析后得到 div 层级关系的结构图，如图 18-7 所示。

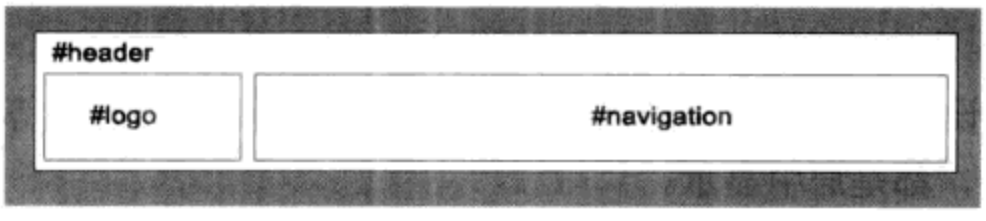


图 18-7 div 层级结构关系

根据层级关系，制作头部的 XHTML 代码。

```
<div id="header">
    <a href="#"><img src="images/20_01.jpg" border="0" /></a>
    <ul id="navigation">
        <li><a href="#">Home</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Support</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Contact</a></li>
        <li><a href="#">Pricing</a></li>
        <li><a href="#">Search Engine Marketing Blog</a></li>
    </ul>
</div>
```

18.3.2 编写头部的 CSS 样式

XHTML 代码制作完成后，就可以为结构编写表现样式。本着尊重原设计稿不改变的原则，使用 CSS 样式实现原设计。在将图片制作成代码的过程中，最重要的就是对设计稿一丝不苟的观察，思考如何合理的布局，对于“距离”这样的细节，1 个像素都不能有偏差。

为头部编写 CSS 样式，代码如下。

```
# header img {
    float: left;                                    /*设置向左浮动定位*/
    margin: 11px 0 0 0;                             /*设置外边距*/
    padding: 0 0 0 65px;                            /*设置内边距*/
}
#navigation {
    float: right;                                   /*设置向右浮动定位*/
    color: #dcc5cc;                                 /*设置文字颜色*/
    margin: 61px 0 0 0;                             /*设置外边距*/
    padding: 0 22px 0 0;                            /*设置内边距*/
}
#navigation li {
    display: inline;                                /*设置内联元素显示方式*/
    list-style: none;                               /*设置列表项目符号样式*/
    font-size: 11px;                                /*设置字号*/
    height: 15px;                                   /*设置高度*/
    line-height: 15px;                              /*设置行高*/
    padding: 0 5px 2px 6px;                         /*设置内边距*/
    margin: 0;                                      /*设置外边距*/
    border-left: 1px solid #dcc5cc;                 /*设置左边框样式*/
    text-align: center;                             /*设置文本居中对齐*/
}
#navigation li a {
    color: #b5143e;                                 /*设置文字颜色*/
}
#navigation li a:hover {
    color: #ff0000;                                 /*设置文字颜色*/
}
```

在以上代码中，使用浮动定位属性，将头部里面的两个对象（网站 logo 和导航），定位在左右两侧。导航采用的是横向文字链接导航系统，使用列表 ul 元素制作。

这样页面头部就实现了 CSS 的布局设计，预览效果如图 18-8 所示。

图 18-8　页面头部 CSS 布局后的预览效果

18.4　页面 banner 区域的布局设计

完成了页面头部布局设计之后，接下来为企业网站首页的 banner 部分进行布局设计。

18.4.1　制作 banner 区域主体的布局设计

进行 banner 区域主体的布局，包括制作 XHTML 代码和编写 CSS 样式。首先制作主体结构

的 XHTML 代码。

1．制作主体的结构

如图 18-9 所示的 banner 区域是由二列式布局实现的，分为 banner-left 和 hints 两个分栏，其中左侧分栏又由 sub-menu 和 themepic 两部分组成。

图 18-9　banner 部分结构分析图

分析后得到 div 层级关系的结构图，如图 18-10 所示。

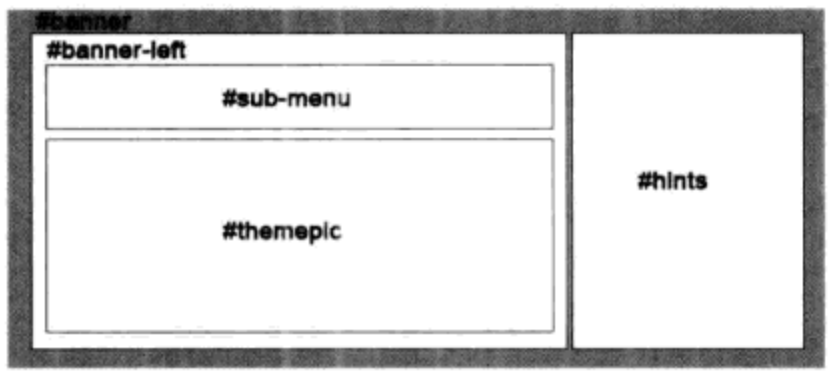


图 18-10　banner 部分布图

根据层级关系，制作头部的 XHTML 代码。

```
<div id="banner-container">
    <div id="banner-left">
        <ul id="sub-menu">
            <li><a href="#">Search Engine Marketing</a></li>
            …
        </ul>
        <div id="themepic">[主题图片]
        </div>
    </div>
    <div id="hints">
        <div id="search">[搜索部分]
        </div>
        快速链接]
    </div>
</div>
```

2．编写主体 CSS 样式

按照 18.3.2 节的结构，为其编写 CSS 代码。

```
#banner-container {
     width: 970px;                                          /*设置宽度*/
     height: 335px;                                         /*设置高度*/
     background: url('images/bg_header.jpg');               /*设置背景图片*/
}
#banner-left {
     width: 660px;                                          /*设置宽度*/
     float: left;                                           /*设置向左浮动定位*/
     padding-left: 64px;                                    /*设置左内边距*/
}
#hints {
     width: 240px;                                          /*设置宽度*/
     float: left;                                           /*设置向左浮动定位*/
     padding: 5px 5px 0 0;                                  /*设置内边距*/
}
```

在以上代码中，首先制作一个 id 名称为 banner-container 的 div 容器，用来控制并且放置二列式分栏。二列式分栏依然是采用浮动定位属性，向左浮动，关键是把每个分栏的宽度掌握准确，确保左右分栏总宽度不超过容器宽度。

重要的是，在#banner-container 元素中，定义了 banner 区域的背景图片，同时设置容器的宽高，使得图片完整显示。

18.4.2　制作 banner 区域各个元素的布局设计

18.4.1 小节把主体框架实现出来，现在为内部的每个版块和功能区进行布局设计，使页面更加饱满、充实。

继续分析 banner 区域的设计，把页面中的各个元素分成区块，如图 18-11 所示。

图 18-11　banner 区域 CSS 样式结构与 div 层级关系

1．制作区块之间的结构

根据图 18-11 所示的 div 层级关系，可以制作这些区块之间的 XHTML 代码。

```
<div id="blog-container">
     <div id=blog-left">
          <ul id="sub-menu">
               <li><a href="#">Search Engine Marketing</a></li>
```

```
            <li><a href="#">Search Engine Optimisation</a></li>
            <li><a href="#">Pay Per Click Management</a></li>
            <li><a href="#">Consultation</a></li>
        </ul>
        <div id="themepic">
            <p>The purpose of your website is to generate sales, leads…</p>
        </div>
    </div>
    <div id="hints">
        <div id="search">
            <form method="get" id="searchform" action="http://www.redflymarketing.com/">
                <div>
                    <fieldset>
                        <legend>Search</legend>
                        <label for="s">Search</label>
                        <input type="text" value="" name="s" id="s" />
                        <input type="submit" id="searchsubmit" value="Search" />
                    </fieldset>
                </div>
            </form>
        </div>
        <a href="#" id="free-quote"><span>Get a Search Engine Optimisation<br />
        or Pay Per Click Management<br />
            FREE Quote</span></a> <a href="#" id="ppc-management"><span>Click here to learn more
about<br />
            our services</span></a> <a href="#" id="search-marketing"><span>Stay in touch with
the latest news in<br />
            our Search Engine Marketing Blog</span></a> </div>
    </div>
```

在以上代码中，对 div #banner 元素的进一步扩展，为元素里面添加了更加细节化的元素。

2. 制作 CSS 布局设计

根据结构代码中为每个元素所命名的名称，制作 CSS 样式。

（1）制作#sub-menu 元素 CSS 样式，代码如下。

```
#sub-menu {
    margin: 0 0 0 0px;                          /*设置外边距*/
    width: 660px;                               /*设置宽度*/
    float: left;                                /*设置向左浮动定位*/
    padding: 0;                                 /*设置内边距*/
}
#sub-menu li {
    display: inline;                            /*设置内联元素显示方式*/
    list-style: none;                           /*设置列表项目符号样式*/
    float: left;                                /*设置向左浮动定位*/
    margin: 0 0 0 1px;                          /*设置外边距*/
    text-align: center;                         /*设置文本居中*/
```

```
    }
#sub-menu li a {
    background: url('images/bg_menu.png');          /*设置背景图片*/
    display: block;                                 /*设置块级元素显示方式*/
    width: 164px;                                   /*设置宽度*/
    height: 30px;                                   /*设置高度*/
    margin: 5px 0 0 0;                              /*设置外边距*/
    line-height: 27px;                              /*设置行高*/
    color: white;                                   /*设置文字颜色*/
    text-decoration: none;                          /*设置文本修饰样式*/
}
#sub-menu li.active a {
    background: url('images/bg_menu_active.png');   /*设置背景图片*/
    display: block;                                 /*设置块级元素显示方式*/
    width: 164px;                                   /*设置宽度*/
    height: 35px;                                   /*设置高度*/
    margin: 0;                                      /*设置外边距*/
    line-height: 35px;                              /*设置行高*/
    color: white;                                   /*设置文字颜色*/
    text-decoration: none;                          /*设置文本修饰样式*/
}
#sub-menu li a:hover {
    background: url('images/bg_menu_hover.png');    /*设置背景图片*/
    width: 164px;                                   /*设置宽度*/
    margin: 0;                                      /*设置外边距*/
    height: 35px;                                   /*设置高度*/
    line-height: 35px;                              /*设置行高*/
}
```

以上代码制作的是一个图片形式导航系统。这个导航系统的导航项有 3 种状态，默认状态 a、点击链接状态 a:active 和光标经过链接状态 a:hover，每种状态都使用图片表示其不同效果。这个导航系统浮动定位在 banner-container 容器的左上角，预览效果如图 18-12 所示。

图 18-12　#sub-menu 对象的实现效果

（2）制作#search-engines 元素的 CSS 样式，代码如下。

```
#search-engines {
    background: url('images/bg_se.jpg');            /*设置背景图片*/
    float: left;                                    /*设置向左浮动定位*/
    width: 250px;                                   /*设置宽度*/
    height: 150px;                                  /*设置高度*/
    padding: 0 0 0 0;                               /*设置内边距*/
    margin: 135px 0 0 1px;                          /*设置外边距*/
}
#search-engines p {
    margin: 20px 30px 0 25px;                       /*设置外边距*/
}
```

在以上代码中，为#search-engines 元素以及元素内的文本定义了样式。

（3）制作#search 元素的 CSS 样式，代码如下。

```
#hints fieldset {
    margin: 0;                                          /*设置外边距*/
    padding: 0;                                         /*设置内边距*/
    border: 0;                                          /*设置边框样式*/
}
#hints label, #hints legend {
    display: none;                                      /*设置不显示元素*/
}
#search {
    background: url('images/bg_search.jpg');            /*设置背景图片*/
    width: 240px;                                       /*设置宽度*/
    height: 30px;                                       /*设置高度*/
}
#search form {
    margin: 0;                                          /*设置外边距*/
    padding: 0;                                         /*设置内边距*/
}
#s {
    background: url('images/bg_search_form.png');       /*设置背景图片*/
    border: 0;                                          /*设置边框*/
    margin: 6px 0 0 6px;                                /*设置外边距*/
    padding: 0;                                         /*设置内边距*/
    width: 162px;                                       /*设置宽度*/
    height: 19px;                                       /*设置高度*/
    line-height: 19px;                                  /*设置行高*/
    float: left;                                        /*设置向左浮动定位*/
}
#searchsubmit {
    background: url('images/bg_search_button.png');     /*设置背景图片*/
    color: white;                                       /*设置文字颜色*/
    border: 0;                                          /*设置边框*/
    margin: 6px 0 0 6px;                                /*设置外边距*/
    padding: 0 0 2px 0;                                 /*设置内边距*/
    width: 60px;                                        /*设置宽度*/
    height: 19px;                                       /*设置高度*/
    font-family: Tahoma;                                /*设置字体类型*/
    float: left;                                        /*设置向左浮动定位*/
    font-size: 11px;                                    /*设置字号*/
    cursor: pointer;                                    /*设置光标经过时的样式*/
}
```

在以上代码中，使用 CSS 改变了表单默认的样式，输入文本框制作成有渐变效果的背景，提交按钮的质感以及文字颜色都进行了重新设置，使其与页面风格相融合。这也是 CSS 的一大特点，美化页面，预览效果如图 18-13 所示。

（4）制作快速通道的图片按钮的 CSS 样式，主要是制作图片与文字的排版，代码如下。

```
a#free-quote, a#ppc-management, a#search-marketing, a#search-marketing {
    width: 240px;                                          /*设置宽度*/
    height: 100px;                                         /*设置高度*/
    display: block;                                        /*设置块级元素显示方式*/
    color: white;                                          /*设置文字颜色*/
    text-decoration: none;                                 /*设置文本修饰样式*/
}
a#free-quote span {
    padding: 37px 0 0 20px;                                /*设置内边距*/
    display: block;                                        /*设置块级元素显示*/
}
a#ppc-management span, a#search-marketing span {
    padding: 52px 0 0 20px;                                /*设置内边距*/
    display: block;                                        /*设置块级元素显示方式*/
}
#free-quote {
    background: url('images/bg_quote.jpg');                /*设置背景图片*/
}
#ppc-management {
    background: url('images/bg_ppc.jpg');                  /*设置背景图片*/
}
#search-marketing {
    background: url('images/bg_sem.jpg');                  /*设置背景图片*/
}
```

在以上代码中，通过 CSS 背景属性，制作了 3 个图片式的按钮，按钮上面放置文字，效果如图 18-14 所示。

图 18-13　#search 对象的实现效果

图 18-14　3 个图片式按钮实现效果

（5）制作页脚的结构，效果如图 18-15 所示。

图 18-15　页脚

明确了页脚的布局以后，就开始为页脚编写结构代码。制作页脚部分的 XHTML 代码。

```
    <div    id="footer"><a    href="#">Privacy    Policy</a>     &copy;
2004-2008 RedFly Marketing Internet Marketing Ireland and Search Engine Optimisation Ireland. All
Rights Reserved.<br />
```

```
        RedFly Ltd. Registered in Ireland, Reg. No. 420836, 29 Forest Court, Rivervalley, Swords,
Co.Dublin +353 (01) 4433840
    </div>
```

18.5 页面主内容区的布局设计

页面的主内容区是内容经常要变换的地方。频道页面的变化，主要也是在主内容区内容上的变化。所以，制作好主内容区的布局设计，对整个网站的建设起到至关重要的作用。主内容区的布局分为左栏和右栏两列，如图 18-16 所示。

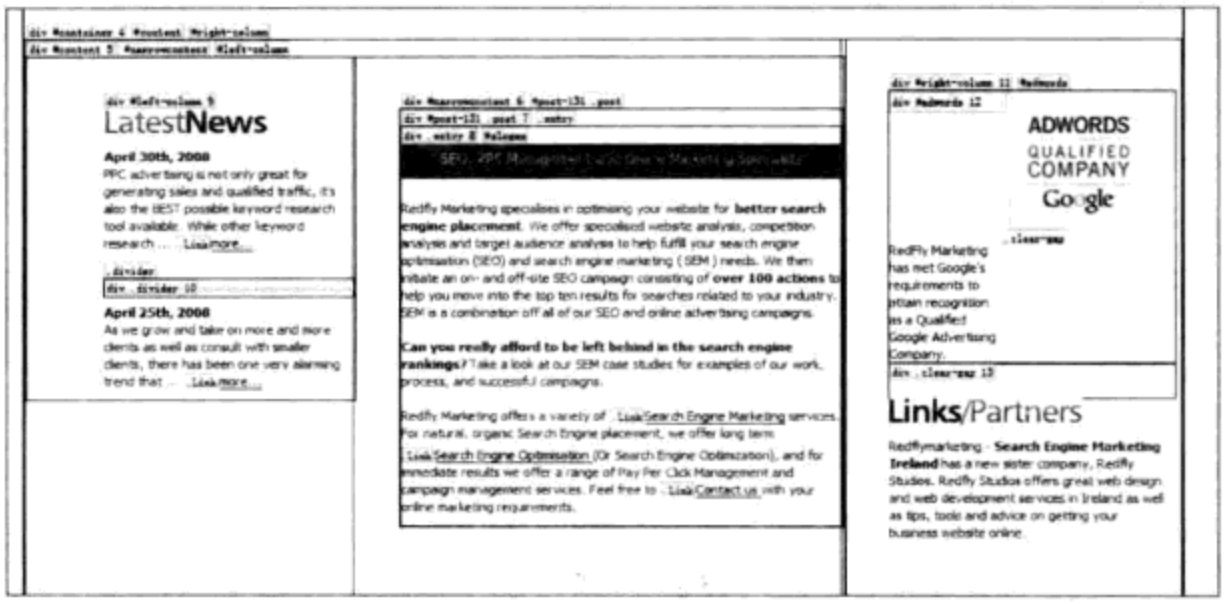


图 18-16 主内容区的 CSS 样式结构与 div 层级关系

18.5.1 制作页面主内容区结构

根据图 18-16 所示的 CSS 结构与 div 层级关系，可以制作这些区块之间的 XHTML 代码。

```
    <div id="container">
        <div id="content">
            <div id="narrowcontent">
                <div class="post" id="post-131">
                    <div class="entry">
                        <h1 id="slogan">SEO, PPC Management and Online Marketing Special-
ists</h1>
                        <p>Redfly Marketing specialises in optimising your website…
                    </div>
                </div>
            </div>
            <div id="left-column"><img src="images/latest_news.png" alt="RedFly News &amp;
Press" longdesc="#" />
                <p><strong>June 4th, 2008</strong><br />
                    Today, Google announced that it would be holding a website optimizer
competitio... <a href="#" class="Link">more...</a></p>
```

```
                <div class="divider"></div>
                <p><strong>April 30th, 2008</strong><br />
                PPC advertising is not only great for generati... <a href="#"
class="Link">more...</a></p>
            </div>
        </div>
        <div id="right-column">
            <div id="adwords"><a href="#"><img src="images/adwords.png" alt="Adwords Qualified
Company" longdesc="#" /></a> <span>RedFly Marketing has met Google's requirements to…</span></div>
            <div class="clear-gap"> </div>
            <img src=" images/links_partners.png" alt="Links/Partners" />
            <p>Redflymarketing - <strong>Search Engine Marketing Ireland</strong> has a…
        </div>
    </div>
```

在以上代码中，对页面的主内容区进行布局的代码制作。

18.5.2　编写 CSS 样式

根据结构代码中为每个元素所命名的名称，制作 CSS 样式。

（1）制作二列式布局的 CSS 样式，代码如下。

```
#container {
    padding-top: 335px;                                    /*设置上内边距*/
    margin-left: auto;                                     /*设置左外边距*/
    margin-right: auto;                                    /*设置右外边距*/
}
#content {
    width: 685px;                                          /*设置宽度*/
    float: left                                            /*设置向左浮动定位*/
}
#right-column {
    float: left;                                           /*设置向左浮动定位*/
    width: 240px;                                          /*设置宽度*/
    padding: 29px 5px 57px 40px;                           /*设置内边距*/
}
```

在以上代码中，#container 元素是与头部、页脚等并列的对象，所以它也需要设置成左右自动对齐，居中显示，使用的是左右外边距属性。

二列式布局就是使用浮动定位属性，定义成向左浮动定位，并且两栏都要设置宽度，总宽度要小于#container 主容器的宽度，才能保证分栏显示在一行。

（2）制作内容主体的分栏布局的 CSS 样式，代码如下。

```
#left-column {
    float: left;                                           /*设置向左浮动定位*/
    width: 210px;                                          /*设置宽度*/
```

```
    padding-left: 65px;                                    /*设置左内边距*/
    padding-top: 30px;                                     /*设置上内边距*/
    position: relative;                                    /*设置相对定位*/
}
#narrowcontent {
    float: left;                                           /*设置向左浮动定位*/
    width: 370px;                                          /*设置宽度*/
    padding: 30px 0px 57px 40px;                           /*设置内边距*/
    position: relative;                                    /*设置相对定位*/
}
```

在以上代码中，同样是使用浮动定位属性，实现二列式布局。

（3）制作#left-column 元素的 CSS 样式，代码如下。

```
#left-column img,{
    display: block;                                        /*设置块级元素显示方式*/
    vertical-align: bottom;                                /*设置垂直底部定位*/
    border: 0;                                             /*设置边框*/
    margin: 0;                                             /*设置外边距*/
    padding: 0;                                            /*设置内边距*/
}
#left-column p {
    width: 210px;                                          /*设置宽度*/
}
.divider {
    background: url('images/divider.png') center repeat-x;    /*设置背景样式*/
    height: 15px;                                          /*设置高度*/
    clear: both;                                           /*禁止对象左右出现浮动定位*/
}
```

在以上代码中，首先定义了#left-column 元素内的图片为块状显示方式，占据容器的一整行显示。然后，对元素内部的文本进行了定义，使用包含选择符#left-column p，定义了 p 标签的宽度。最后，定义了一个背景图片作为新闻条目之间的分割线。

这样主内容区左侧的新闻更新内容的布局就实现了，如图 18-17 所示#left-column 内的所有新闻条目都可以应用这套样式。

（4）制作# narrowcontent 元素的 CSS 样式，代码如下。

```
#narrowcontent {
    float: right;                                          /*设置向右浮动定位*/
    width: 370px;                                          /*设置宽度*/
    padding: 30px 0px 57px 40px;                           /*设置内边距*/
    position: relative;                                    /*设置相对定位*/
}
#narrowcontent .post {
    width: 370px;                                          /*设置宽度*/
}
.post {
```

```
    margin-bottom: 11px;                                          /*设置下外边距*/
}
h1#slogan {
    width: 370px;                                                 /*设置宽度*/
    height: 30px;                                                 /*设置高度*/
    background: url('images/slogan.png') no-repeat;               /*设置背景样式*/
    margin: 0;                                                    /*设置外边距*/
    padding: 0 0 12px 0;                                          /*设置内边距*/
    text-align: center;                                           /*设置文本居中*/
    color: white;                                                 /*设置文字颜色*/
    line-height: 26px;                                            /*设置行高*/
    font-size: 12px;                                              /*设置字号*/
    font-weight: normal;                                          /*设置字体*/
}
.entry {
    margin: 0 0 12px 0;                                           /*设置外边距*/
    padding: 0;                                                   /*设置内边距*/
}
```

在以上代码中，首先使用浮动定位属性将#narrow 对象向右浮动定位，与 left-column 对象实现二列式布局。正文放置在.entry 元素内，正文标题的样式由 h1#slogan 元素控制，效果如图 18-18 所示。

图 18-17　#left-column 效果

图 18-18　#narrowcontent 效果

（5）制作# right-column 元素的 CSS 样式，代码如下。

```
#right-column {
    float: right;                                                 /*设置向右浮动定位*/
    width: 240px;                                                 /*设置宽度*/
    padding: 29px 5px 57px 40px;                                  /*设置内边距*/
    position: relative;                                           /*设置相对定位*/
}
#adwords {
    background: url('images/bg_adwords.png') top repeat-x;        /*设置背景样式*/
    width: 240px;                                                 /*设置宽度*/
    height: 120px;                                                /*设置高度*/
    margin:0;                                                     /*设置外边距*/
```

```
    padding:0;                                          /*设置内边距*/
}
.clear-gap {
    clear:both;                                         /*禁止对象左右出现浮动元素*/
    height:30px;                                        /*设置高度*/
    margin:0;                                           /*设置外边距*/
    padding:0;                                          /*设置内边距*/
    font-size:0px;                                      /*设置字号*/
}
```

在以上代码中，首先实现二列式的浮动布局设计，#adwords 元素等一些文字信息都是#right-column 的元素。.clear-gap 元素的设置，禁止浮动元素出现，起到分隔线的作用。这样#right-column 元素就实现了，效果如图 18-19 所示。

图 18-19 #right-column 效果

18.6 频道页面布局设计概述

按照 CSS 布局的统一性与继承性，在首页布局的基础上，分析网站上频道页面与首页的异同，除去顶部与页脚的设计之外，在内容区的设计上，都是二列式布局。从布局角度上看，二列式布局中右栏的宽度和样式完全相同。而对于二列式布局中的左栏以及更细节的部分，根据各频道页面的内容不同采用不同的布局样式。以其中一个频道页面为例，分析它的布局设计，如图 18-20 所示。

从图中可以看出，新的#banner-blog 区域高度减小了，是由于右侧的快速通道其中一个图片按钮移到了#container 区域，形成了新的#container-blog 区域。新的#container-blog 区域，右栏除了多了一个快速通道的图片按钮以外，其他的都和首页相同。

这样，把首页的 div 结构的 XHTML 代码稍作修改，只是将#hints 元素内的 a: search-marketing 元素放在#right-column 元素内就可以了。对 CSS 的命名稍作修改，区别于首页的名称。按照这种方法制作的模板应用到每个频道页面的布局设计中来，很容易制作。

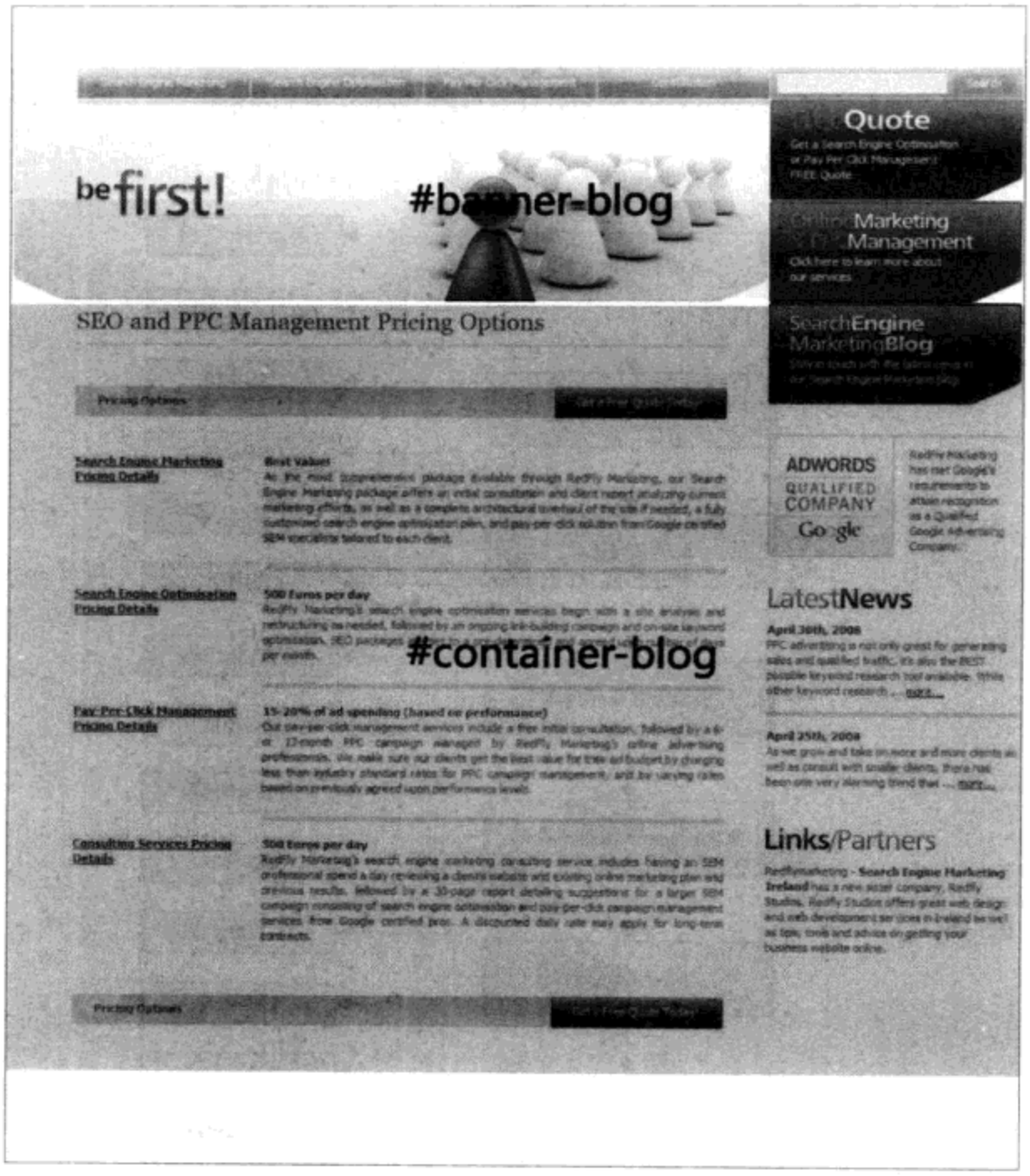


图 18-20　频道页面布局分析

18.7　小结

本章主要介绍制作企业类网页的流程。

（1）页面布局与规划。

（2）CSS 结构设计。

（3）整体布局设计。

（4）页面头部布局设计。

（5）页面 banner 区域布局设计。

（6）主内容区域的布局设计。

（7）频道页面的布局设计。

对本章的知识点前后联系进行归纳总结，结构导图如图 18-21 所示。

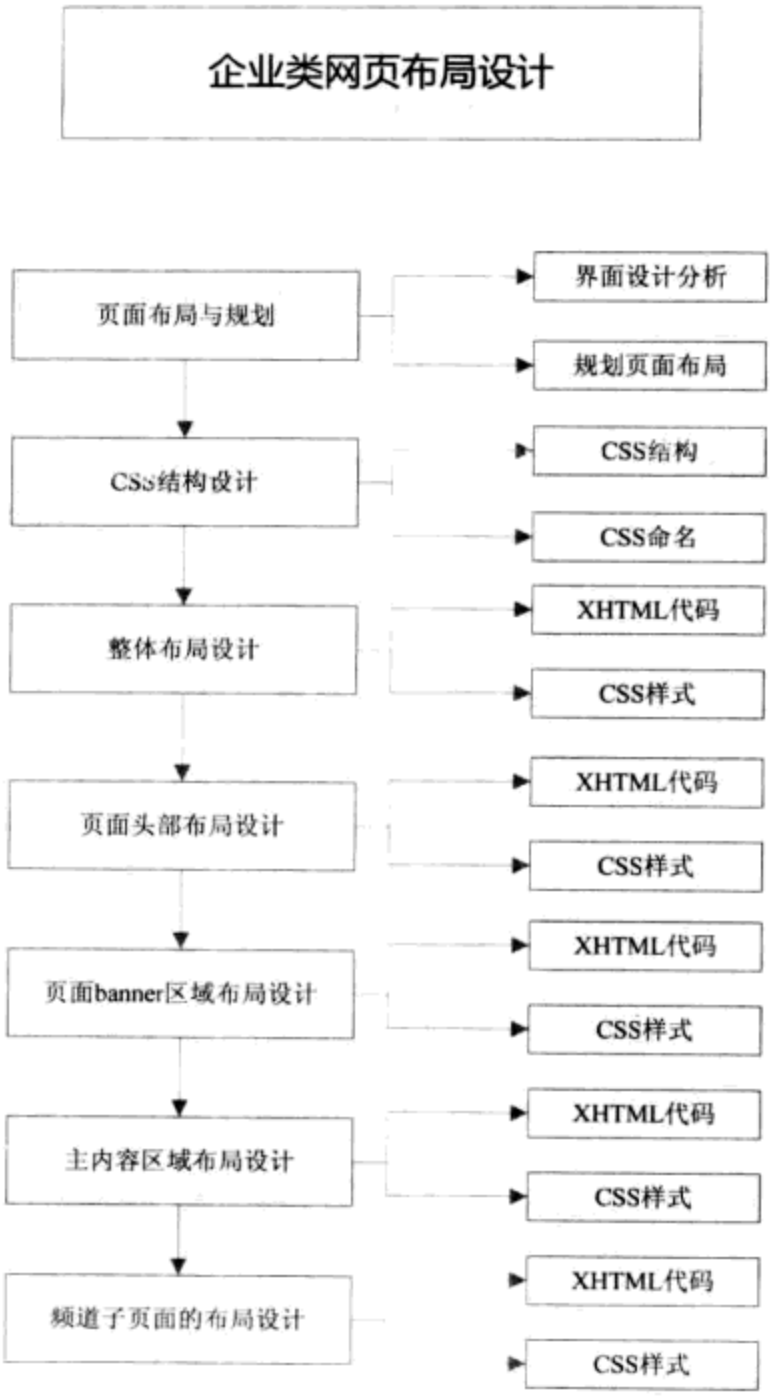


图 18-21 本章知识点结构导图