CSS
设计彻底研究

本书是一本深入研究和揭示CSS设计技术的书籍。本书在透彻地讲解CSS核心技术的基础上，深入到各个实际应用领域中，充分向读者演示了如何使用CSS的各项技术，实现令人眩目的网页布局和效果。

本书主要结构

封面设计：董志桢

分类建议：计算机/网页设计/CSS
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-17296-9

9 787115 172969 >

ISBN 978-7-115-17296-9/TP
定价：69.00 元（附光盘）

CSS

设计彻底研究

前沿科技 温谦 编著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

CSS 设计彻底研究 / 温谦编著. —北京：人民邮电出版社，2008.2
ISBN 978-7-115-17296-9

Ⅰ. C… Ⅱ. 温… Ⅲ. 主页制作—软件工具，CSS
Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字（2007）第 188430 号

内 容 提 要

本书是一本深入研究和揭示CSS设计技术的书籍。本书在透彻地讲解CSS核心技术的基础上，深入到各个实际应用领域中，充分向读者演示了如何使用CSS的各项技术，实现令人眩目的网页布局和效果。

本书详细介绍了CSS核心基础、盒子模型等知识，力求把道理和方法讲清楚，采用了“探索式”的讲解方法，对于一个问题，例如标准流、浮动、定位等规律，均通过一系列动手实验，使读者自己就能够非常自然地得出结论，使读者不但知其然，而且还知其所以然。

在本书中对设计中常用的网页元素和布局方式都给出详细的分类和归纳，并讲解了完整的解决方法，主要包括各种导航菜单（水平的、竖直的、固定宽度的、自适应宽度的、下拉的等），Tab面板、伸缩面板和折叠面板，以及各种形式的分列布局（固定宽度的、变化宽度的、固定宽度与变化宽度结合的），等等。这样读者在理解了方法的基础上，可以直接将案例用在自己的设计中的，只需要按照所需进行修改即可。

本书适合需要使用CSS的Web设计人员和开发人员阅读，最好具备一定的HTML和网页设计制作基础。

CSS 设计彻底研究

◆ 编　　著　前沿科技　温　谦
责任编辑　杨　璐

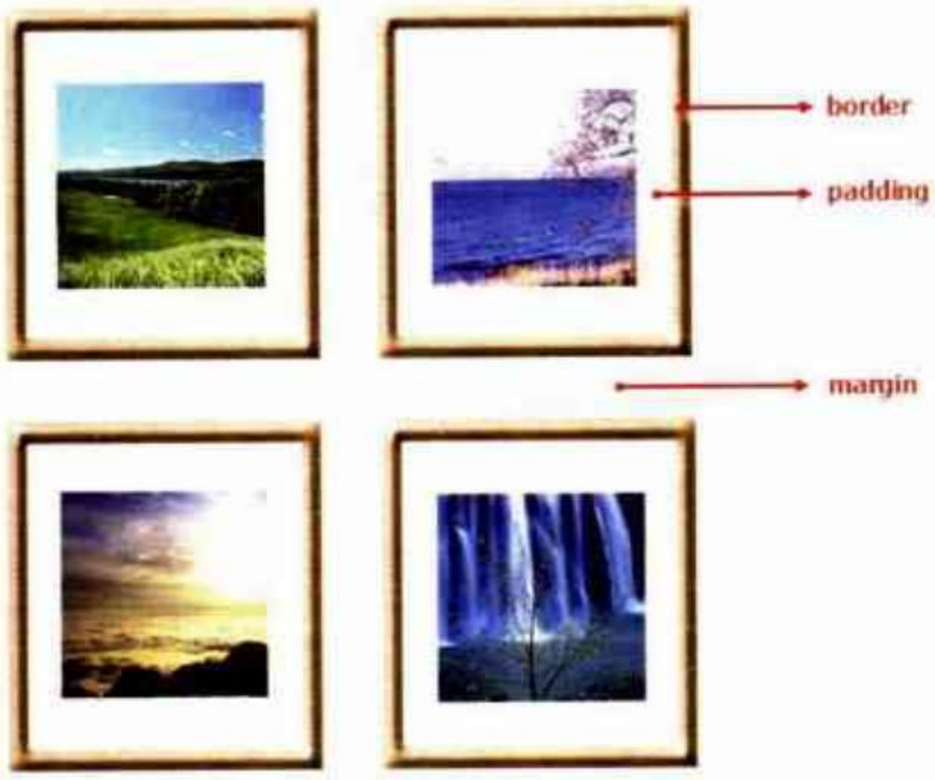
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：29.5　　彩插：4
字数：736 千字　　2008 年 2 月第 1 版
印数：1 – 5 000 册　　2008 年 2 月北京第 1 次印刷

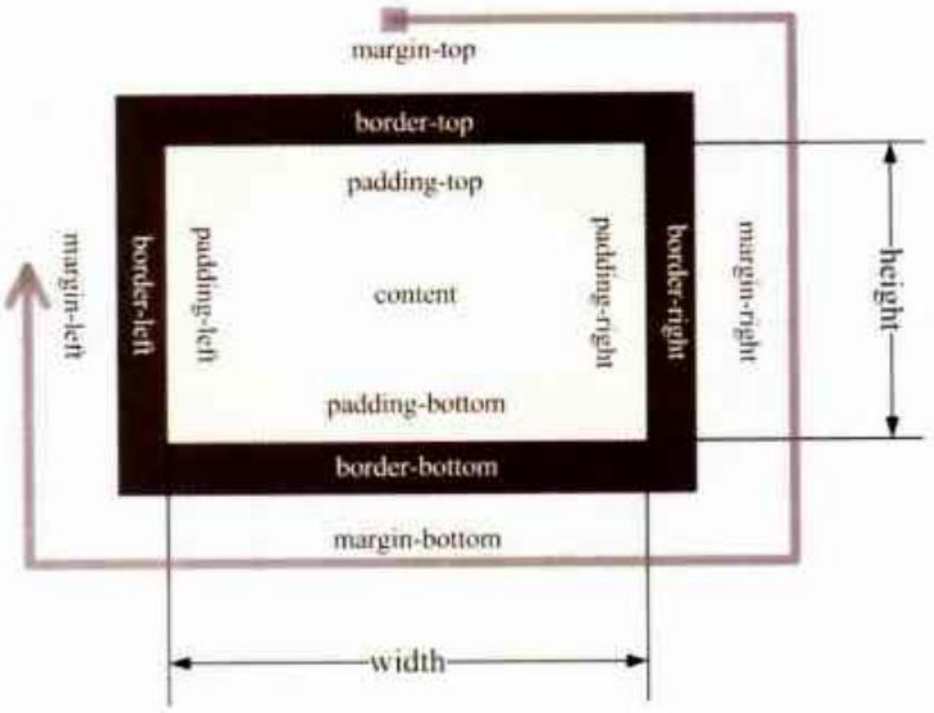
ISBN 978-7-115-17296-9/TP

定价：69.00 元（附光盘）

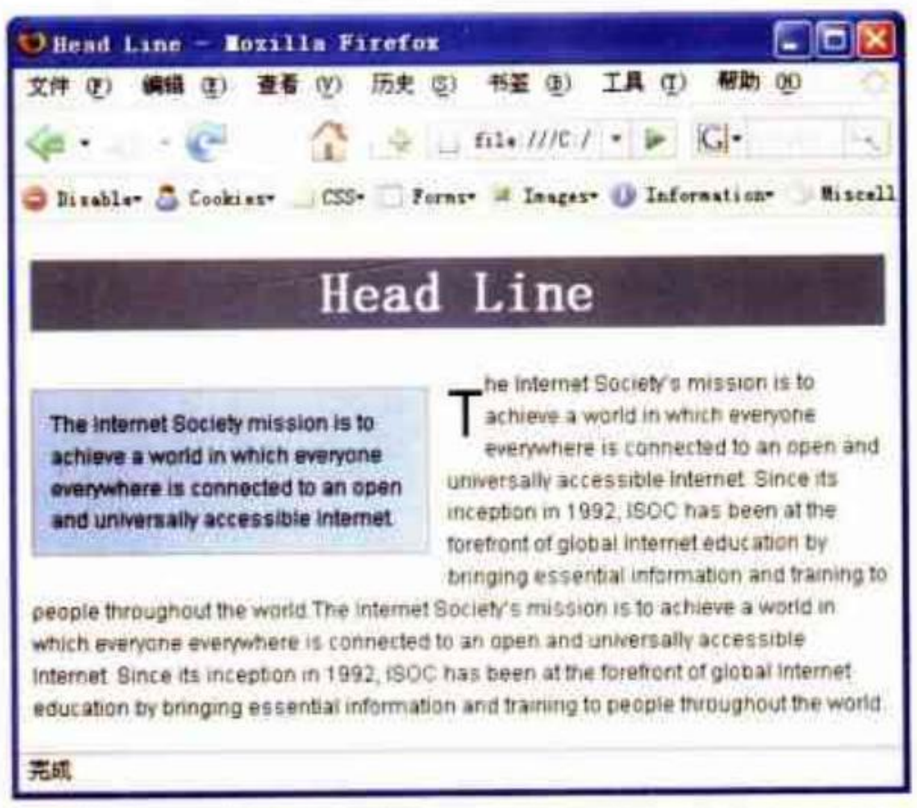
读者服务热线：(010) 67132692　印装质量热线：(010) 67129223
反盗版热线：(010) 67171154



第3章 理解盒子模型的示意图

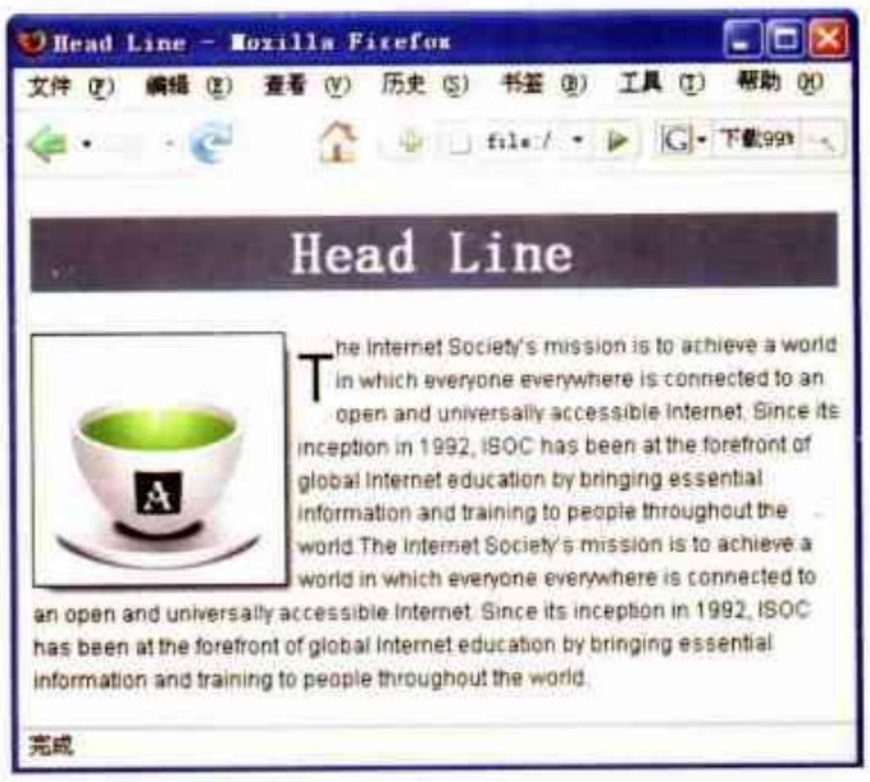


第3章 盒子模型结构图

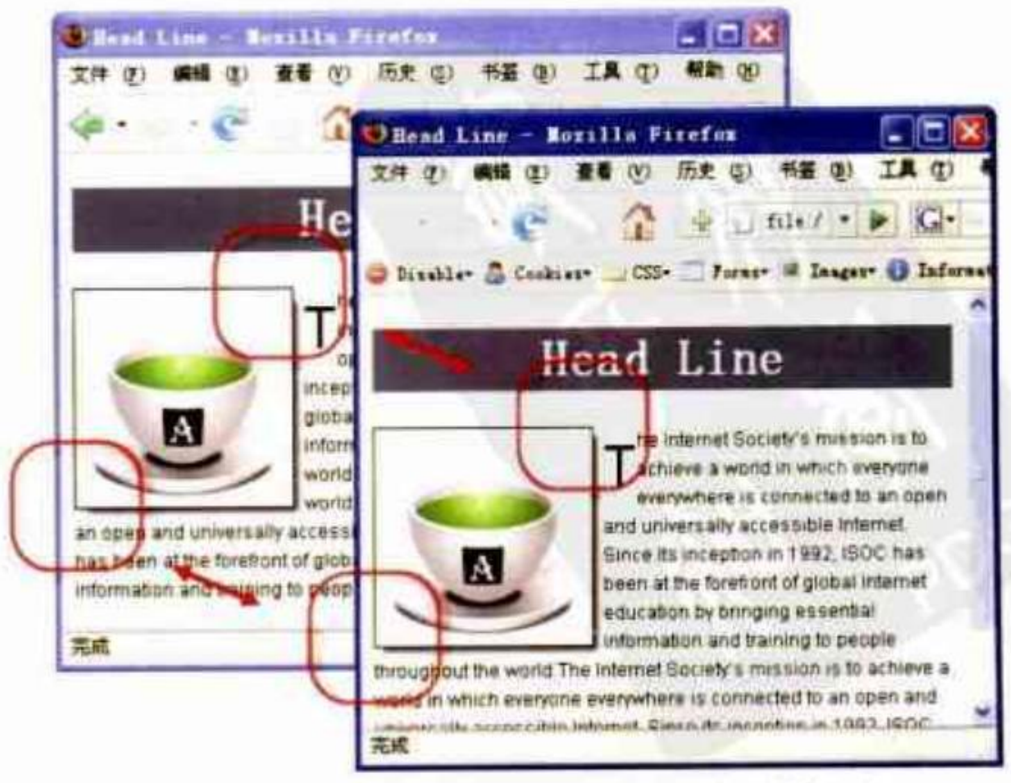


第5章案例 文字页面排版，用CSS可以方便地实现文字环绕等效果

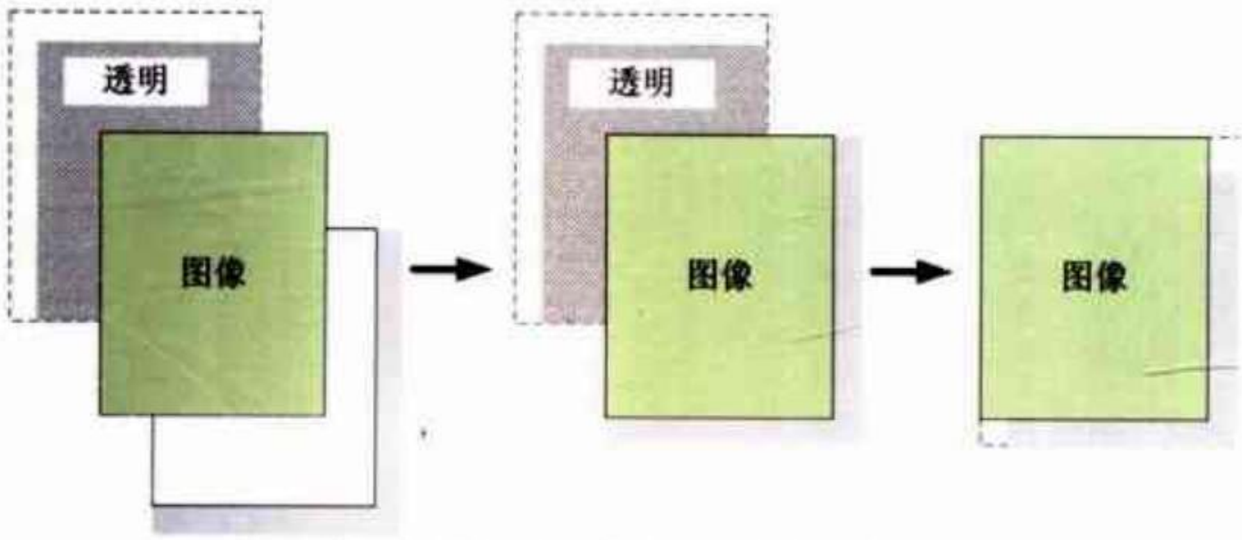
第5章案例 图像的应用



第5章案例 为图像增加阴影效果



第5章案例分析 柔边的阴影与硬边阴影比较



第5章 滑动门技术应用的原理示意图

第6章案例 按钮式超链接

第6章案例 浮雕式超链接

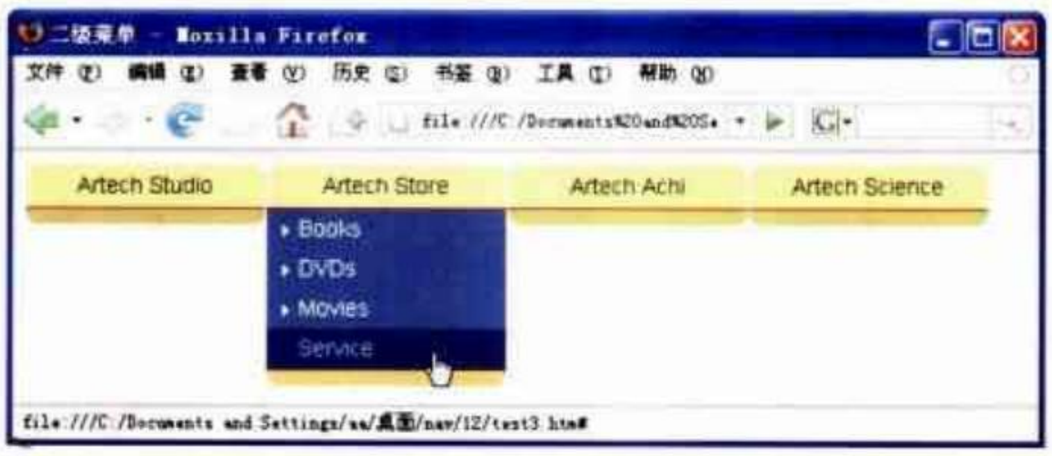
第7章案例 双竖线菜单效果

第7章案例 带说明信息的菜单效果

第8章案例 自适应的斜角水平菜单效果

第8章案例 三状态玻璃效果菜单

第8章案例 会跳起的多彩菜单



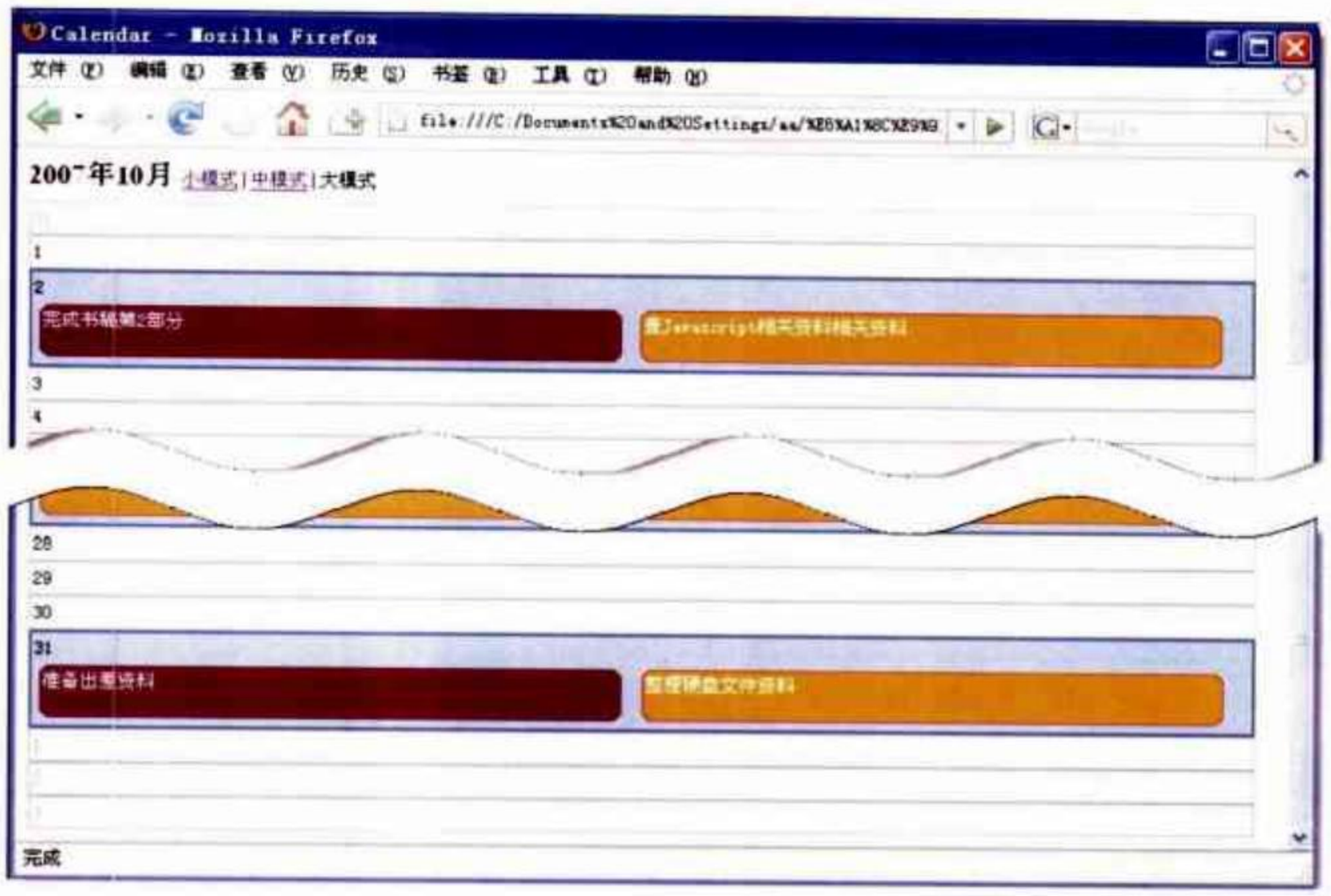
第9章案例 下拉菜单

第10章案例 中视图显示模式的日历效果

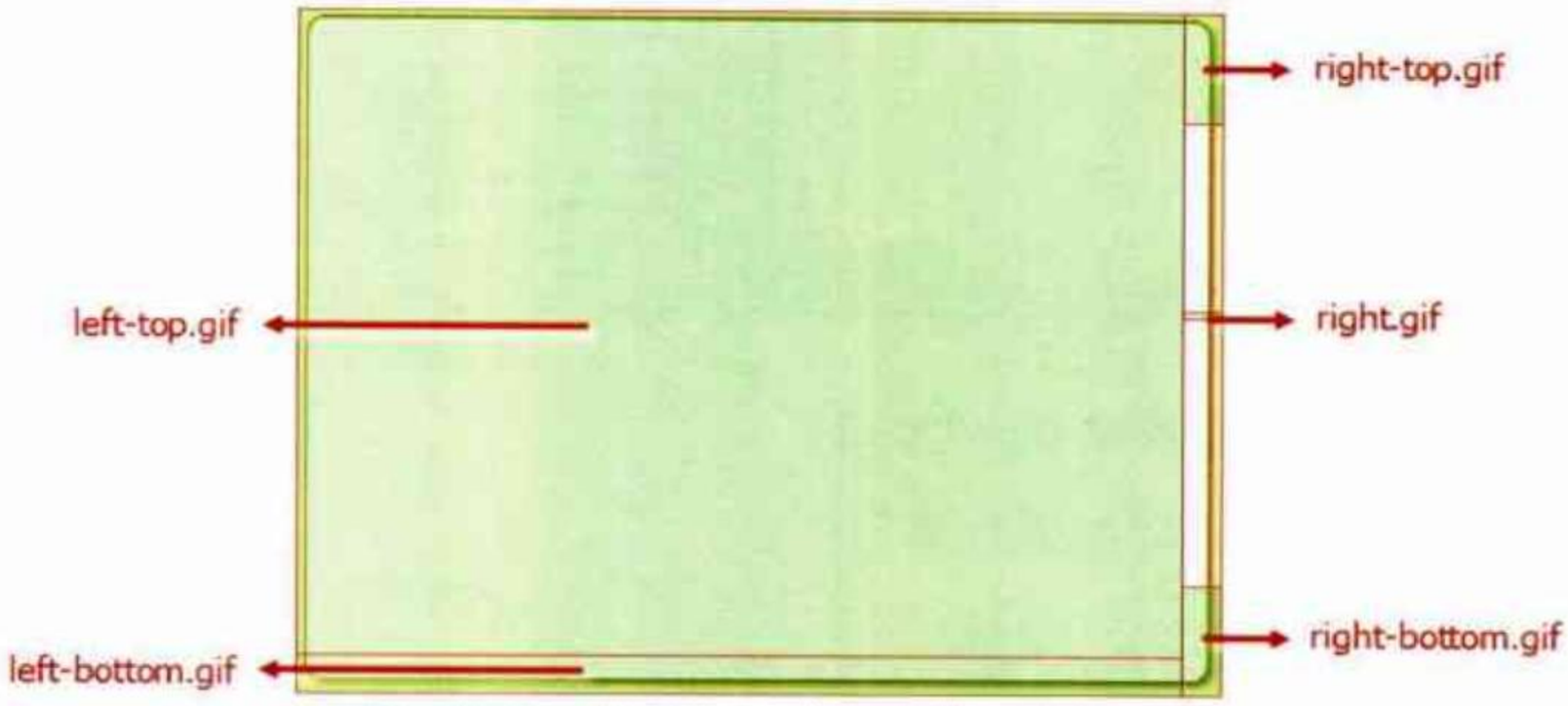
第10章案例 鼠标经过时整行变色提示效果

第10章案例 鼠标经过时行列二维变色提示效果

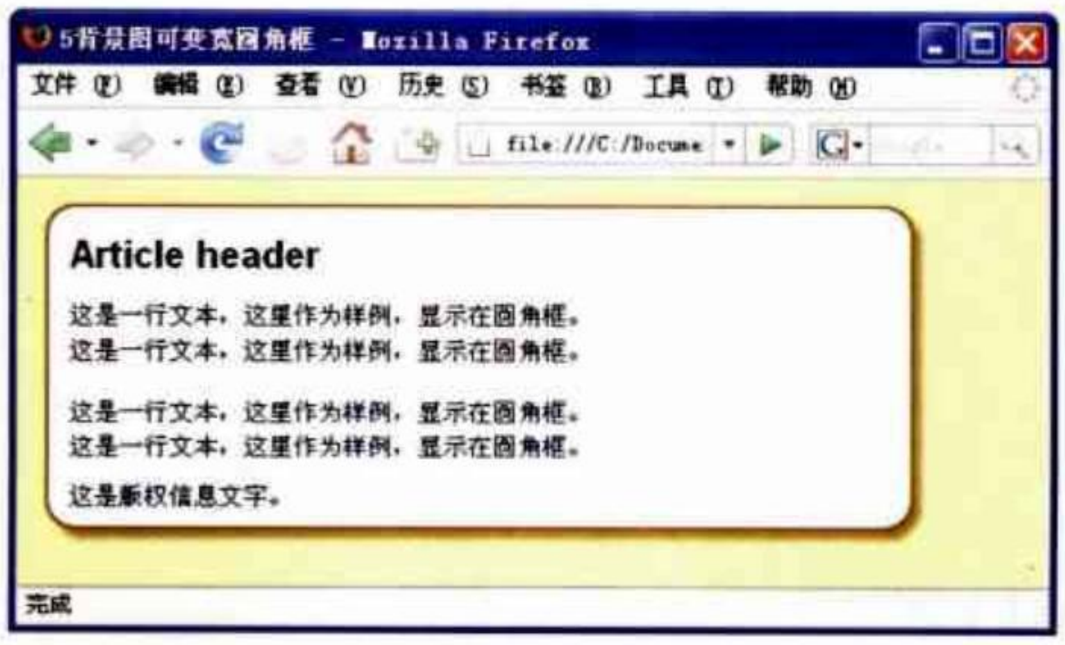
第10章案例 小视图显示模式的日历效果



第10章案例 大视图显示模式的日历效果

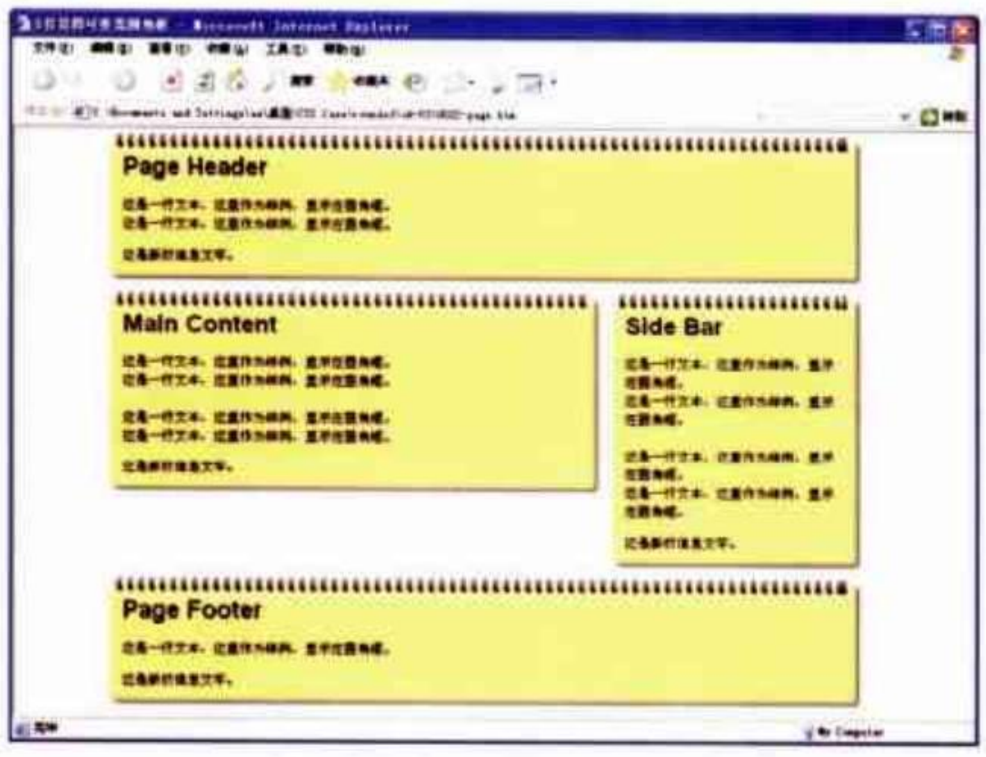


第11章案例 5背景图像经典圆角框背景图像切分示意

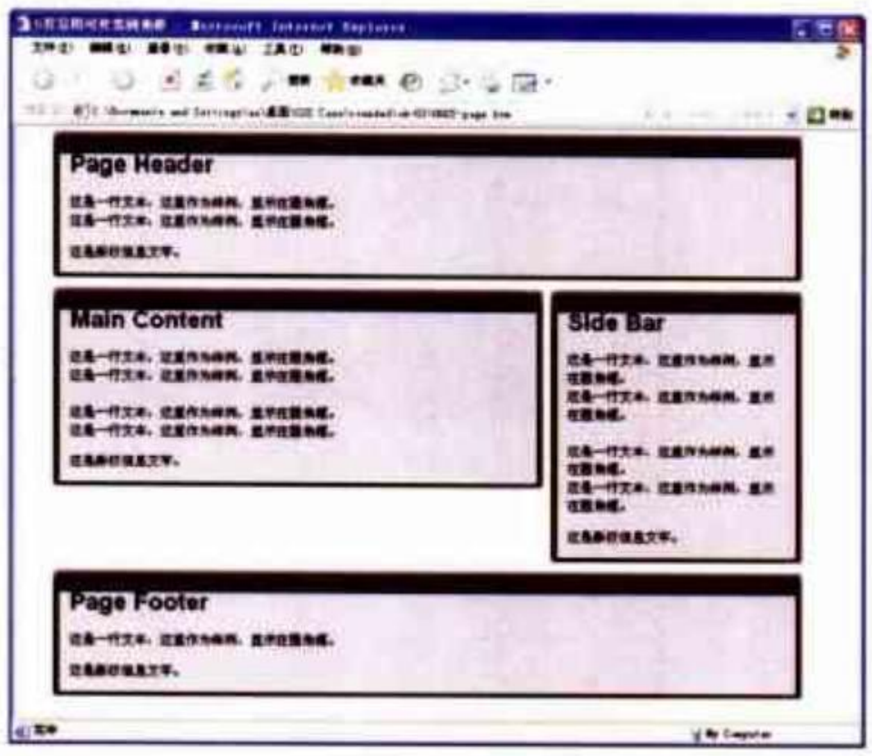


第11章案例 5背景图像经典圆角框

第11章案例 将圆角框应用于整个页面

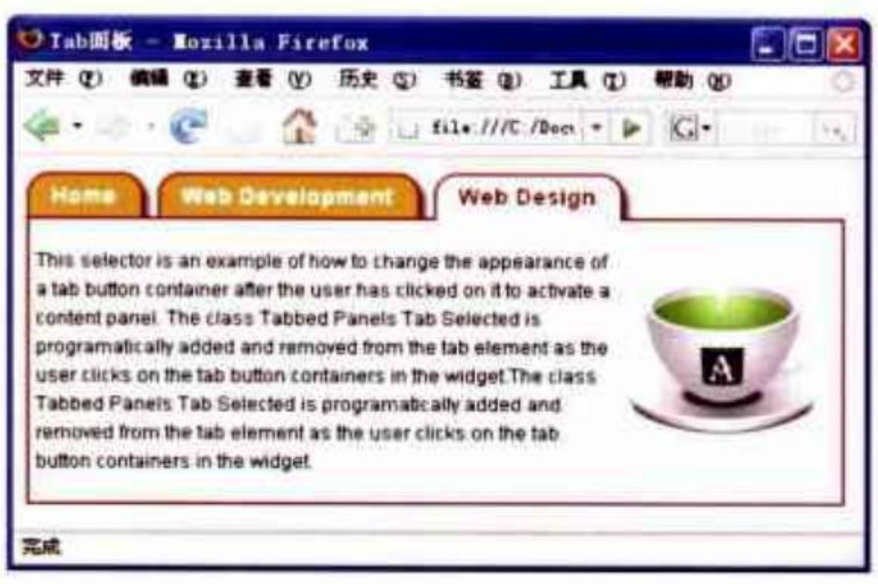


第11章案例 网页换肤效果1

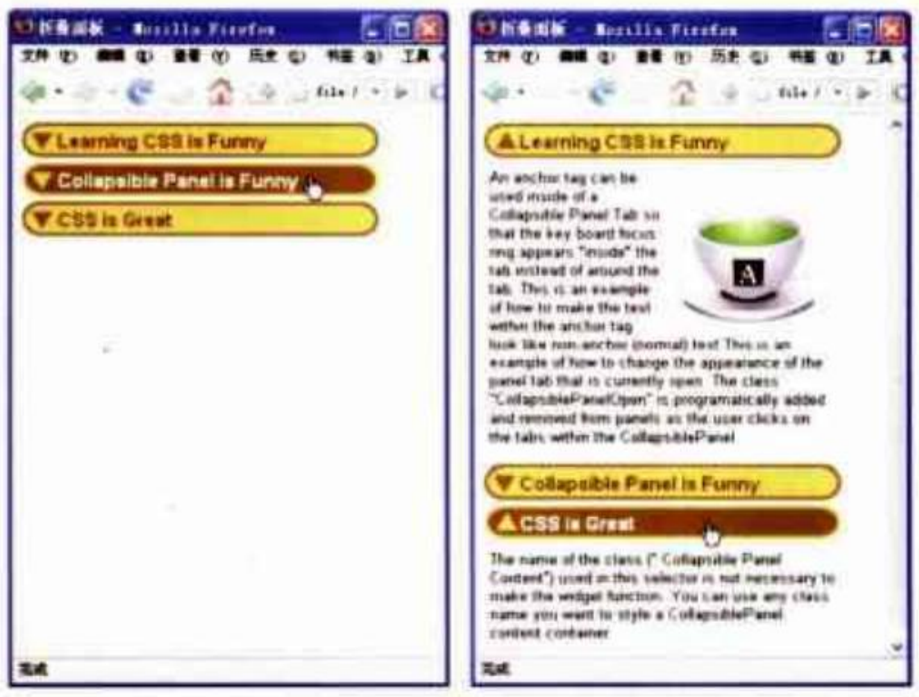


第11章案例 网页换肤效果2

第12章案例 制作基本的Tab菜单

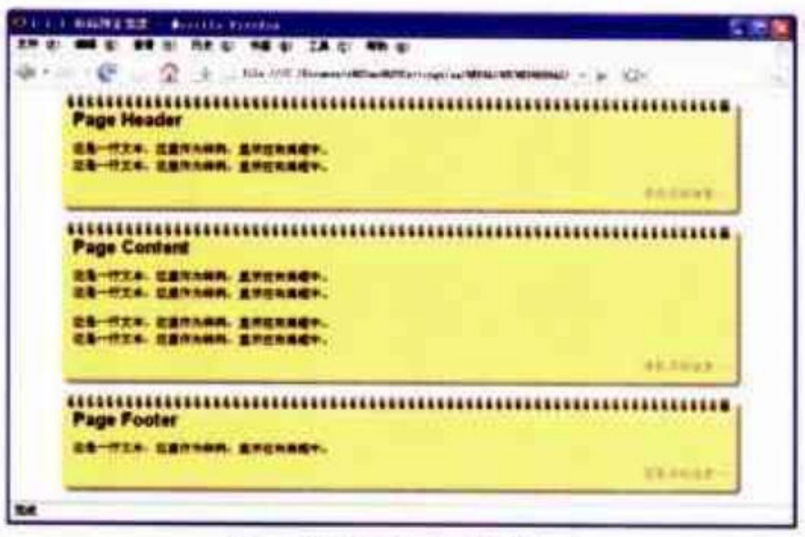


第12章案例 制作完善的Tab面板

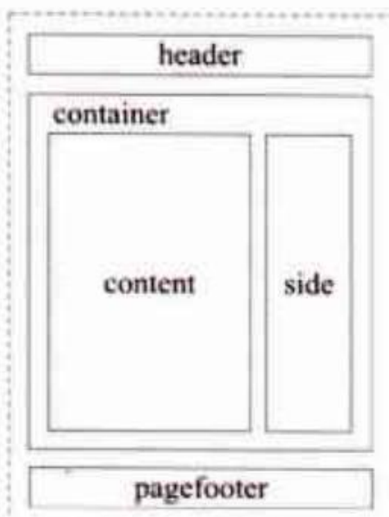


第12章案例 制作折叠面板

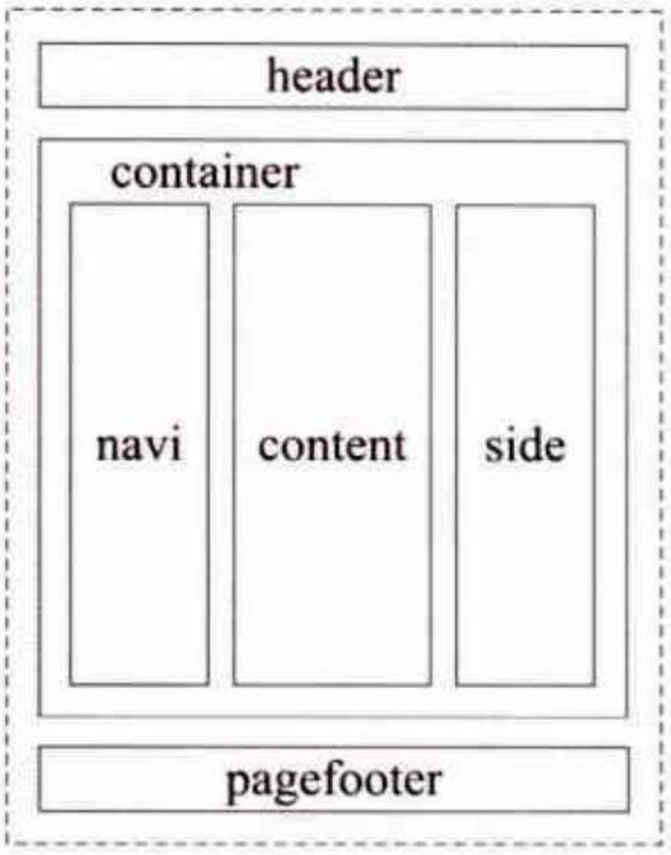
第12章案例 制作伸缩面板



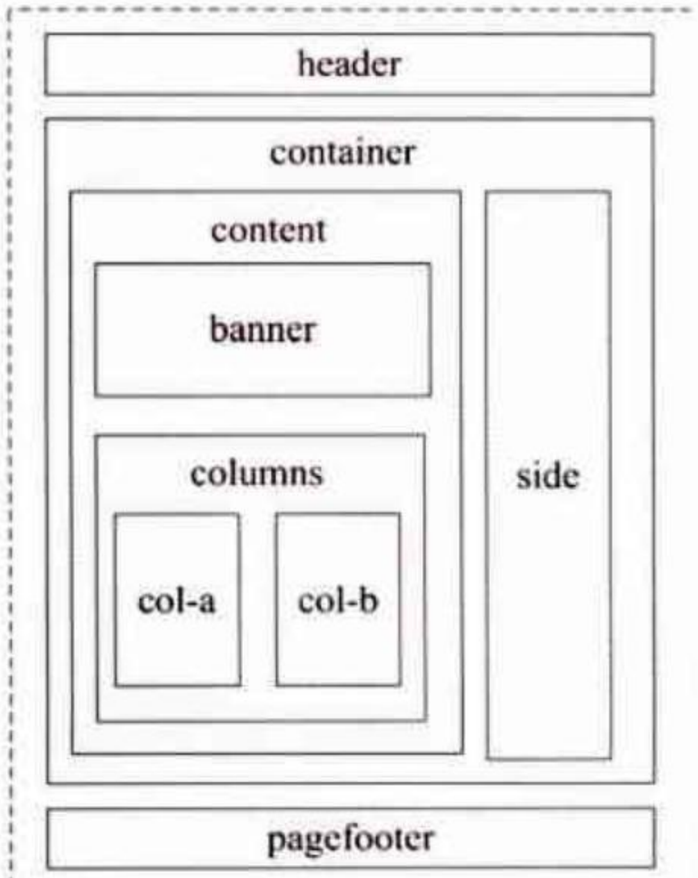
第13章案例 单列布局

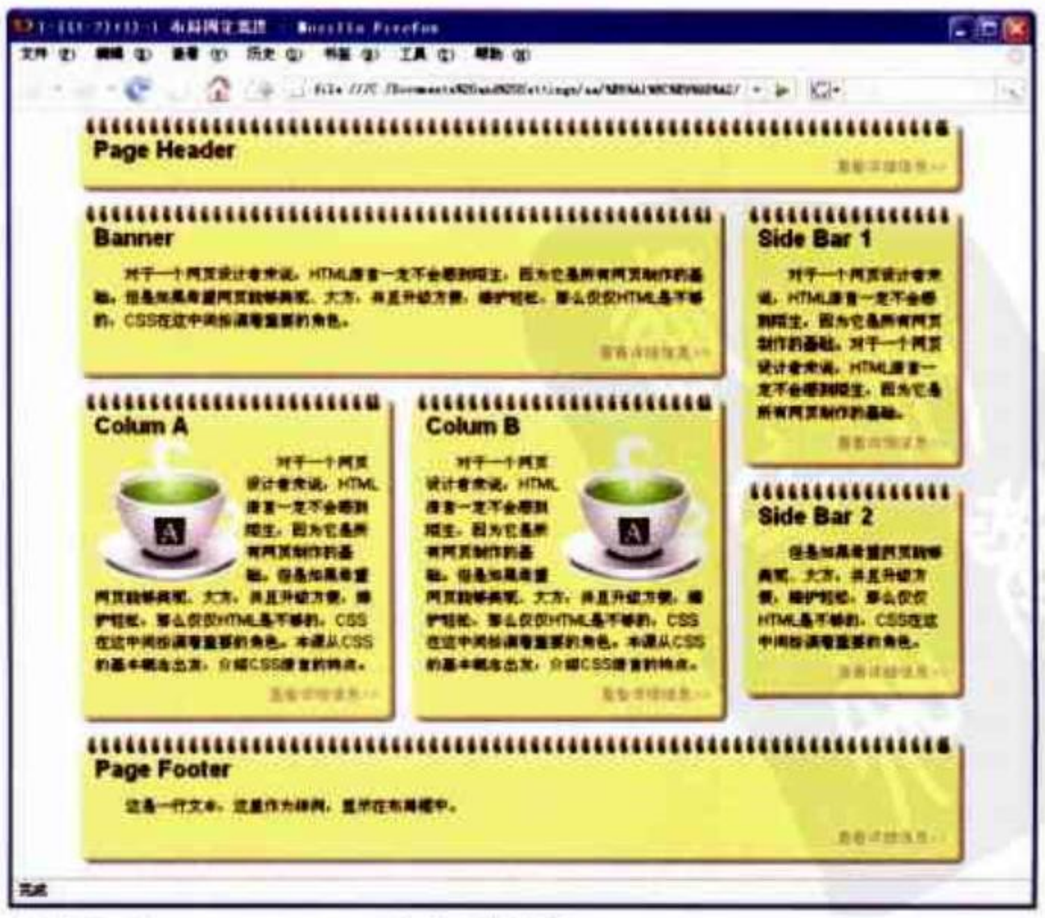


第13章案例 “1-2-1”固定宽度布局



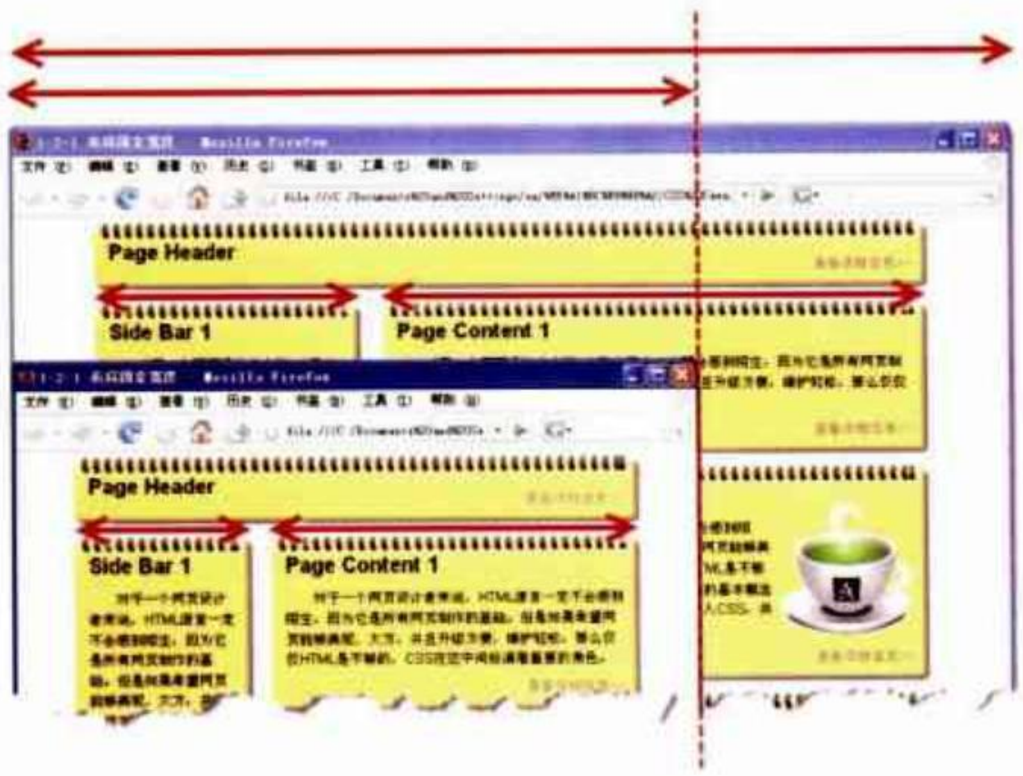
第13章案例 “1-3-1”布局



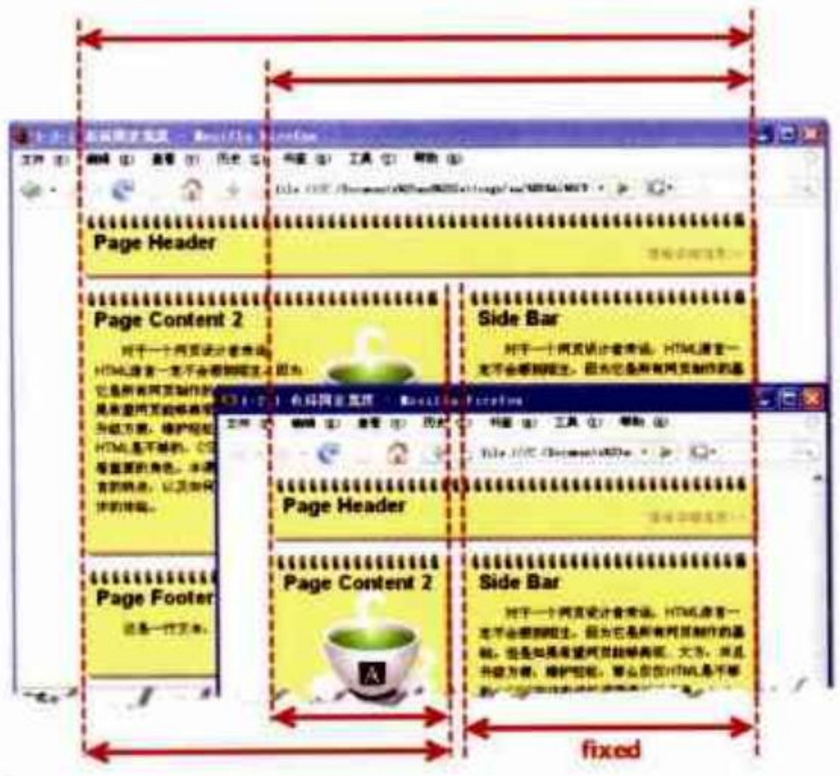


第13章案例 更为复杂的“1-((1-2)+1)-1”布局效果

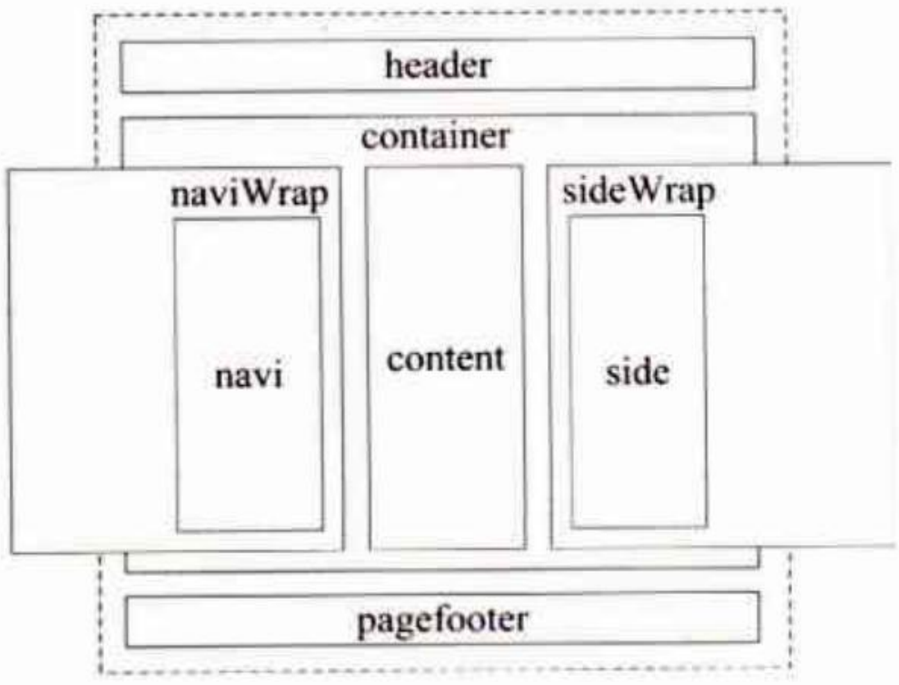
第13章案例 魔术布局效果



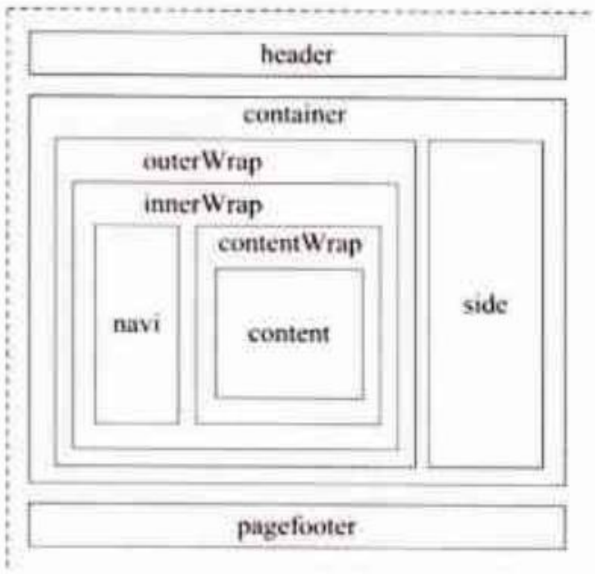
第14章案例 “1-2-1”等比例变宽度布局



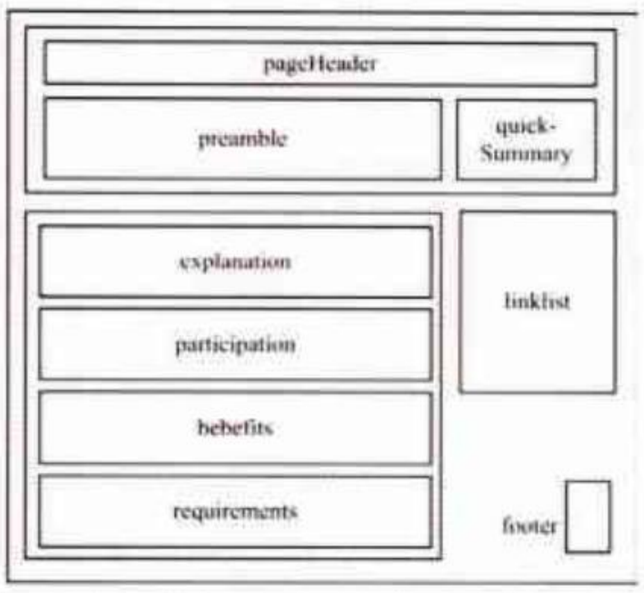
第14章案例 单列宽度固定的“1-2-1”变宽布局



第14章案例 中间列宽度固定的“1-3-1”变宽布局效果

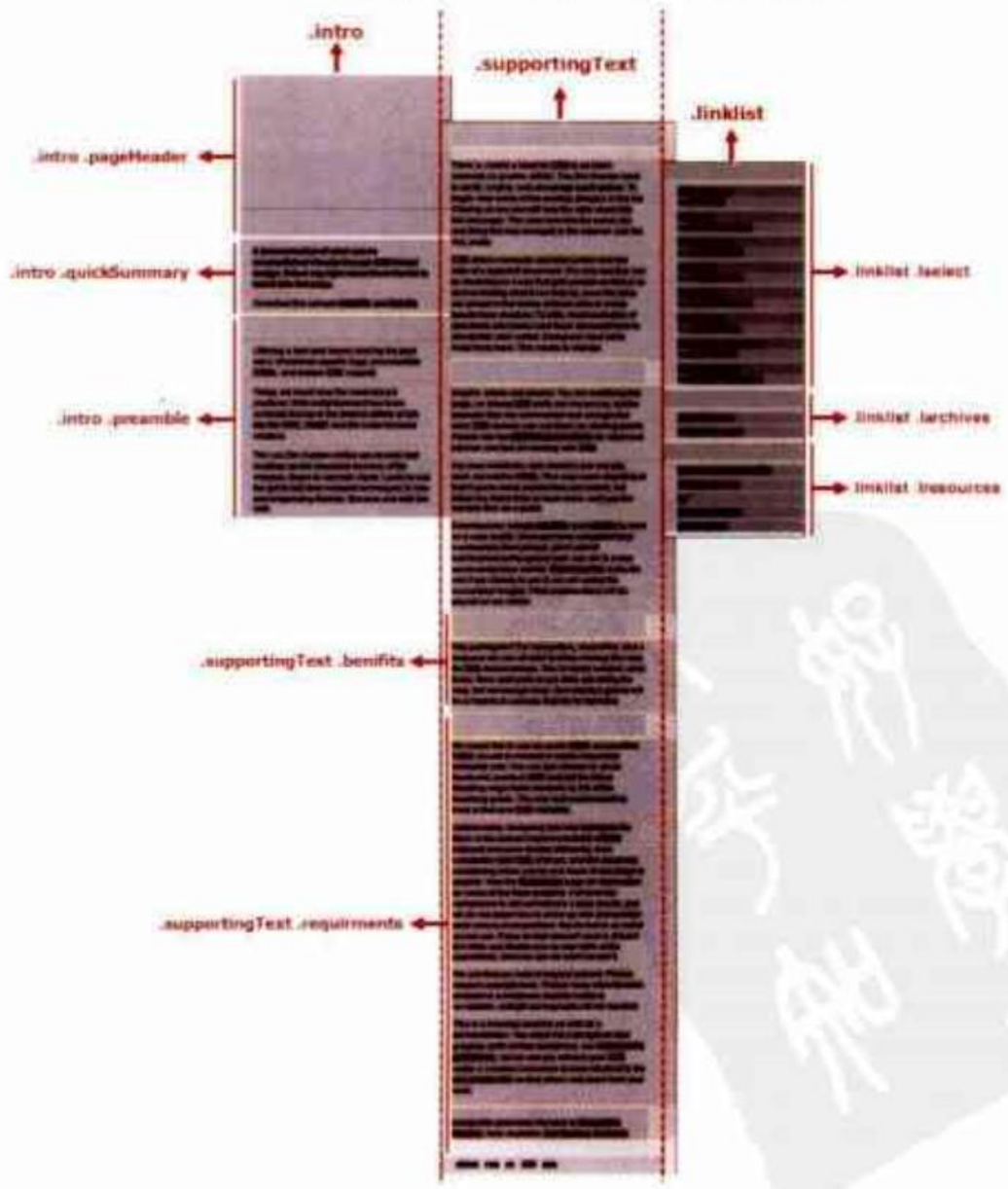


第14章案例 中间列变宽的“1-3-1”变宽布局效果

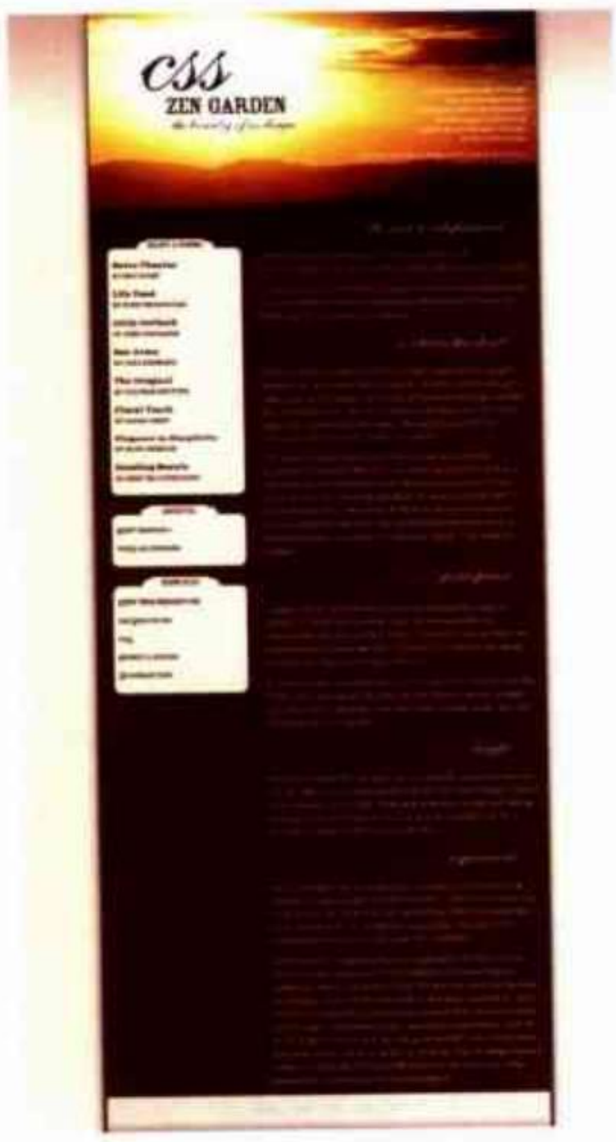


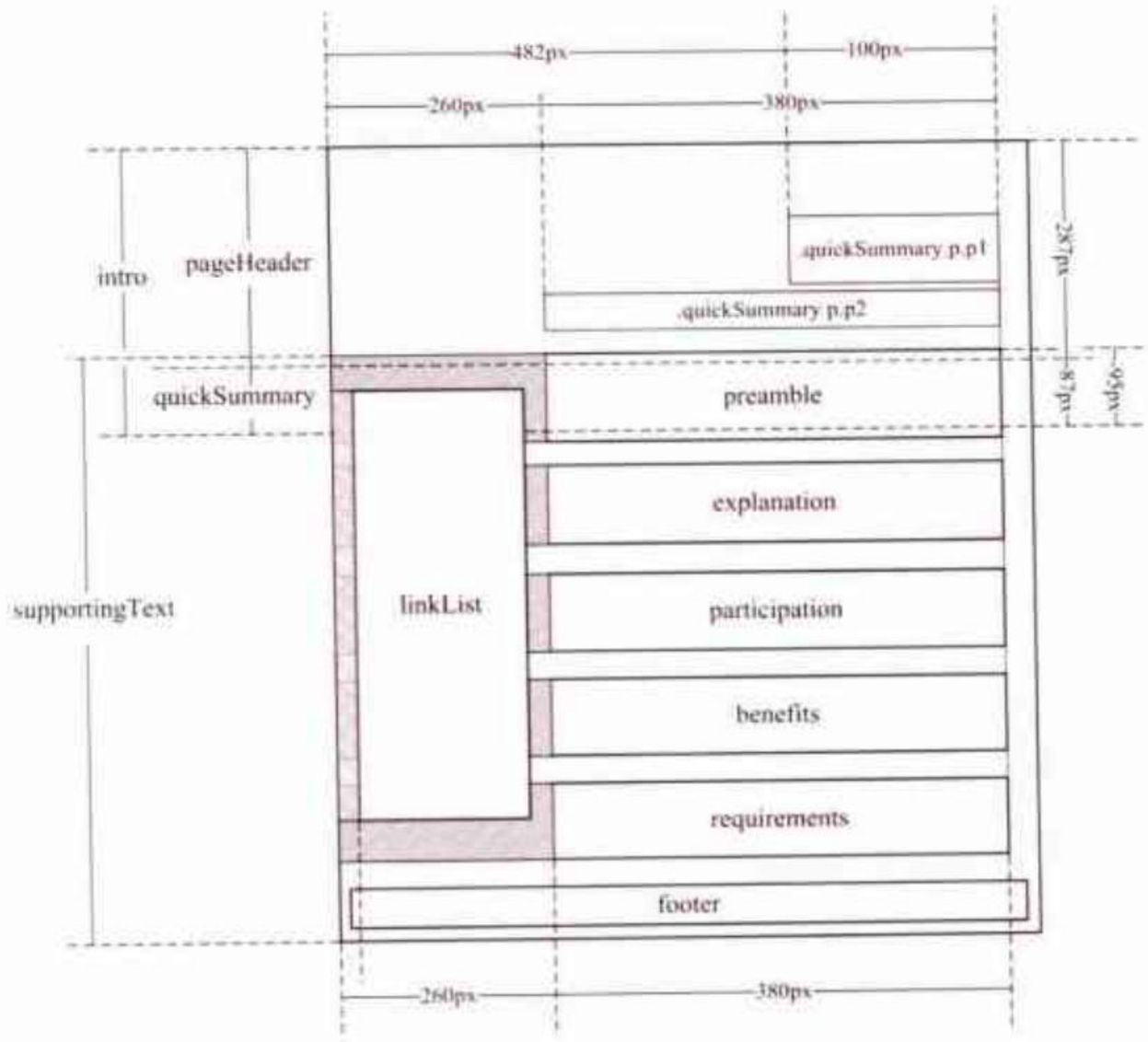
第15章案例 页面的布局示意图

第15章案例 一个简单的禅意花园网页

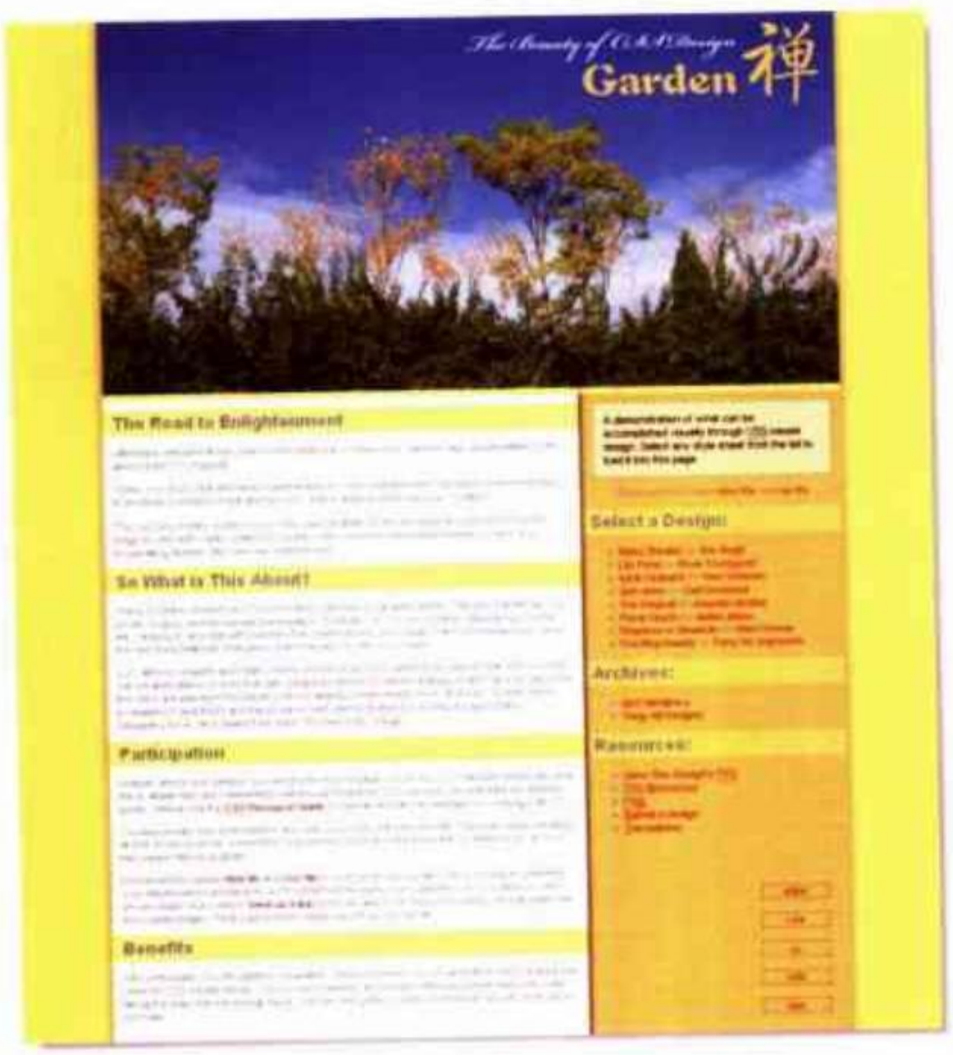


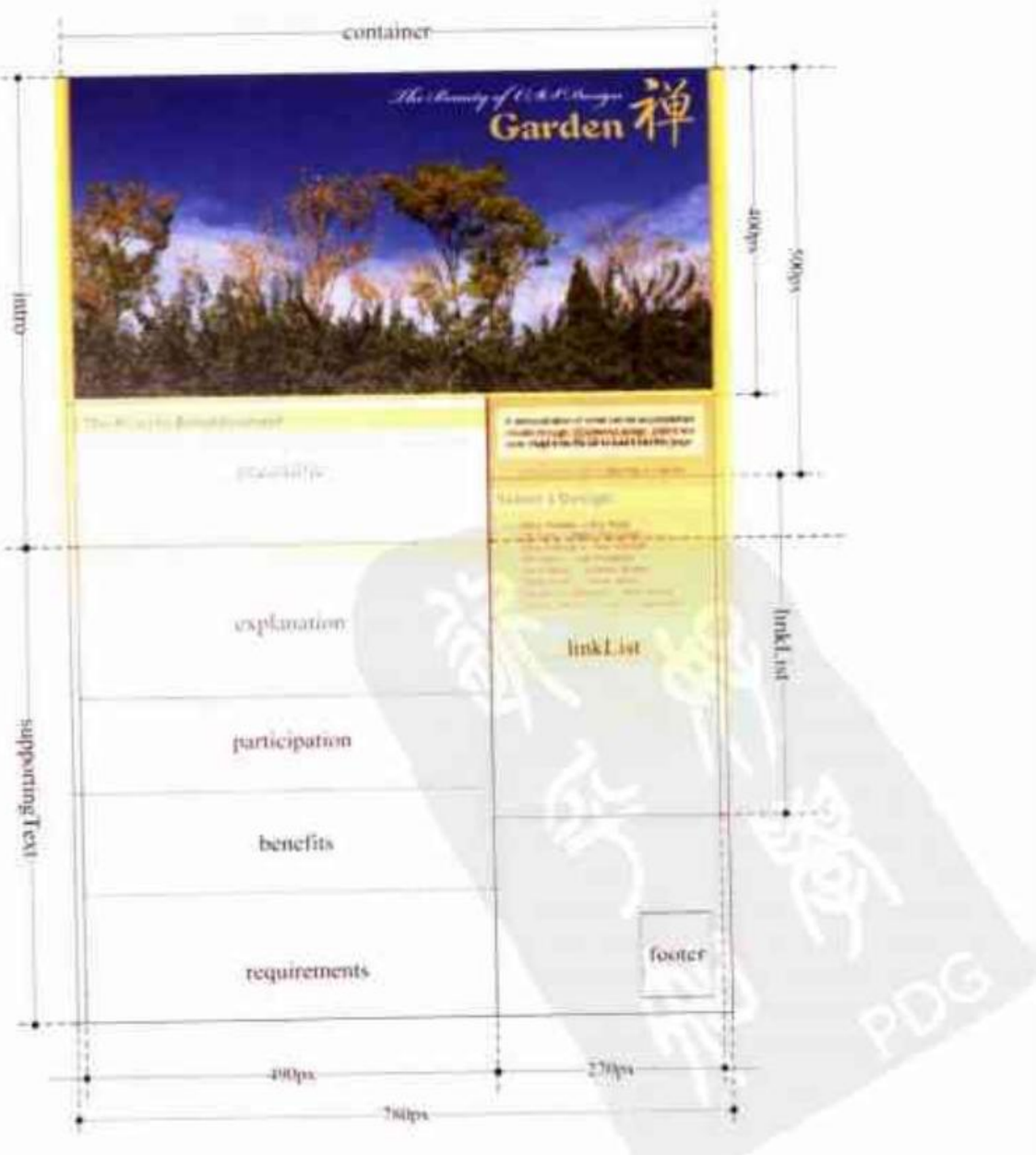
第15章案例 026号禅意花园作品效果分析和布局结构分析



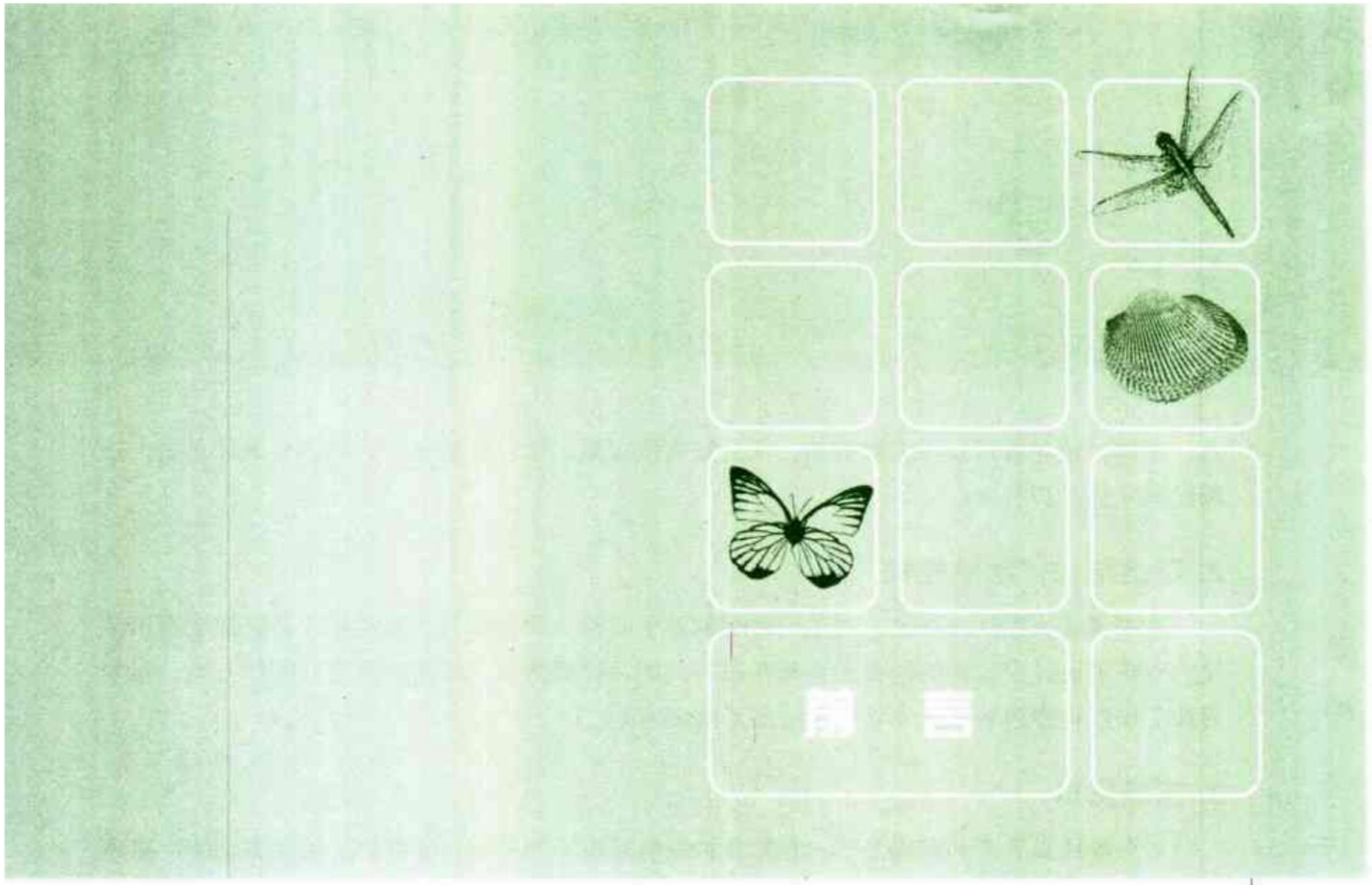


第16章案例 158号禅意花园作品及结构图





第17章案例 设计禅意花园作品及结构图



爱上CSS

CSS是一种崭新的“老技术”。在互联网领域，任何一种诞生于10年前的技术都称得上是“老技术”，但是CSS是一个特例，它是一种崭新的老技术，因为在几乎诞生10年以后，它才闪耀出真正动人心魄的魅力。

学习和研究一项技术的动机有两种，一种是工作，另一种是兴趣。CSS不但可以用来工作谋生，更值得爱上这项技术，是因为它提供了无限的创造空间。

本书将像一位导游那样，带领读者深入CSS的核心，深入剖析CSS的机制和原理，并把这些原理应用到一个个实际的案例之中。

希望读者在读完本书以后，也能爱上CSS。

面向读者

阅读本书的读者最好具有一定的网页制作基础，并对HTML的基本元素有所了解。

（1）对HTML和网页设计有初步的基础。

（2）具有钻研的精神和热情。

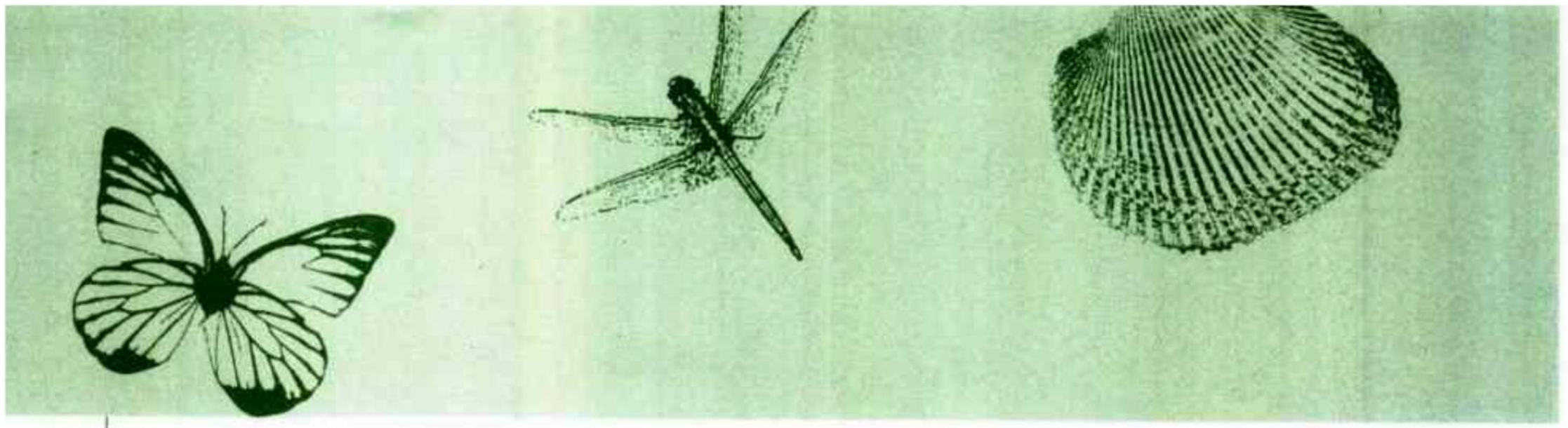
其中第1点的权重占10%，第2点权重占90%。

本书适合每一个需要使用CSS的Web设计人员和开发人员阅读。

本书特点

1. 深入探索

本书以独特的探索式方式对CSS设计技术的原理和方法进行讲解，符合自学者的认知特



点。这样，读者就可以不但知其然，而且知其所以然。学习任何东西，明白了道理之后，应用起来才能游刃有余。

2. 得出结论，总结经验

本书带领大家深入探索，通过一系列的动手实验，使读者自己就能够非常自然地得出结论。对各种设计中常用的网页元素和布局方式的设计都给出了完整的思路和制作方案，对学习和工作过程中遇到的沟沟坎坎也给出了解决方法。

3. 指导应用

对各种网页元素和布局方式，包括各种导航菜单（水平的、竖直的、固定宽度的、自适应宽度的、下拉的等），Tab面板、伸缩面板和折叠面板，以及各种形式的分列布局（固定宽度的、变化宽度的、固定宽度与变化宽度结合的），等等，都给出了详细的分类和归纳，便于读者在理解的基础上，直接修改后使用。

4. 综合实践

我们与“CSS禅意花园”的创建者Dave Shea取得了联系，获得了使用禅意花园作品的授权。本书会结合禅意花园网站中的一些精彩案例，给出分析和说明。最后将综合前面所学，从创意、拍摄素材到实际完成的全过程，实际制作自己的“禅意花园”作品，培养读者的综合实践能力。

本书内容安排

本书共分为17章。

第1章：介绍(X)HTML和CSS相关的核心基础知识。

第2章：向读者展示CSS能够给网页设计带来的效果。

第3章：深入讲解CSS的核心机制——盒子模型。

第4章：讲解CSS布局的重点和难点——浮动和定位。

第5章：介绍文字和图像的排版。其中为图像增加阴影这个案例难度比较大。

第6章：介绍链接和导航相关的设置方法。

第7章：制作比较简单的竖直排列的导航菜单。

第8章：制作复杂的水平排列的导航菜单。

第9章：制作两级的下拉菜单。

第10章：介绍表格样式的设置方法。

第11章：深入讲解制作圆角框的多种方法，深入分析了不同方法的特点和使用范围。

第12章：介绍在近年来出现的一些新的网页元素的制作方法，例如Tab面板、折叠面板和伸缩面板等。

第13和14章：全面地归纳和总结不同形式布局的设计方法，并给出全面的案例。这两章是全书技术难度最大的两章。

第15至17章：以CSS禅意花园的作品为例，在研究成功作品的基础上，制作自己的“CSS禅意花园”作品，从而综合实践整体页面的布局方法。

光盘内容

本书光盘收录了精心编排并制作的多媒体视频教学课程，辅助读者学习CSS。视频教程分为两个部分。

（1）共计6小时针对本书内容的视频讲解，对书中的内容进行说明、演示和必要的补充，建议读者在学习时，将本书和视频教程配合使用，相信对技术的理解会非常有帮助。

（2）为没有任何HTML基础和网页制作基础的读者准备的基础内容，对于有基础的读者可以跳过。

光盘中还包括了本书所有案例的源代码、素材和最终效果文件。

学习方法建议

1. 务必重视对基础的掌握

CSS和HTML的区别在于HTML基本上不涉及逻辑，学习起来非常容易，而CSS涉及了相对比较复杂的逻辑过程，理解起来就复杂得多。因此，读者不要只关心某一个效果的最终结果，更重要的是理解其中的核心原理。

2. 记熟本书中出现的英文单词

很多初学者感到CSS学起来比较困难，很重要的一个原因是，CSS的属性名称众多，这一点对于母语是英语的人来说，几乎不存在困难，因为这些属性名称本身就是一些常用的单词。这里建议读者先把遇到的英语单词集中背一次，就像学英语课文先学单词一样，扫清单词障碍，学起来就会容易很多了。

为此，我们总结了本书中比较频繁出现的160个英文单词，并给出了中文解释，将其放在本书的附录中，读者在学习遇到困难时可以查一查，或者索性把这160个单词先背下来。



3. 在理解的基础上一定要多练习、多实践

“纸上得来终觉浅，绝知此事要躬行。”希望读者能够在自己探索和研究的基础上学习本书，而不要把直接使用本书的例子作为目的。只有经过了自己的思考和消化，才能真正掌握技术。

4. 利用好本书的光盘

本书的光盘中包含的视频教程可以帮助理解书中的一些难点和重点，读者可以充分利用光盘中的视频，提高学习效率。

5. 下载并安装Firefox浏览器

本书介绍的案例均可以在IE 6、IE 7和Firefox这3种最流行的浏览器中正确显示（除个别有特殊说明的案例），而其中Firefox浏览器对CSS和Web标准支持最完善，因此在调试时，建议读者同时用Firefox和IE浏览器测试。下载Firefox浏览器的官方网址是http://www.mozilla.org.cn。

6. 利用支持网站的内容辅助学习

除了与本书配套的视频课程，读者还可以访问前沿科技建立的“前沿视频教室网站”，网址是http://www.artech.cn，里面除了本书相关的内容，还有更多的关于CSS以及其他Web设计与开发相关的内容，供读者学习。

建议那些有更高要求的读者，按照本书中给出的一些扩展内容的网址，做进一步的阅读和学习。读者可以访问http://learning.artech.cn/20071220.css-link-list.html，其中列出了所有的链接地址，以使读者免去输入网址的麻烦，它们大都是一些非常经典的CSS专家的技术文章。

获取更新内容

本书由前沿科技的温谦主要编写，参与编写工作的还有王璐、郝雯、温颜、吕岩岩、张金辉、韩军、温鸿钧、白玉成、王斌、刘璐、陈宾、刘军、黄欢、刘艳茹、曾顺、梁万强、谢伟、黄世明、陈宾、张晓静、武智涛、彭启、蔡庆武、孙琳等。如果读者在学习过程中，遇到有关书中的问题，或有任何建议，请与我们联系，E-mail是luyang@ptpress.com.cn。请登录网站http://learning.artech.cn，我们会及时更新与本书相关的信息和内容。

我们衷心希望本书能给广大读者提供一些真正的帮助，如果您有任何建议或者意见，欢迎和我们联系。

目　录 CONTENTS

Chapter

第1章
(X)HTML与CSS核心基础

本书假设读者已经对HTML有所掌握，如果读者了解以下HTML标记中大多数标记的含义，就可以开始学习本书。

<html>、<head>、<title>、<body>、<p>、<img>、<a>、 、<ul>、<ol>、<li>、<style>、<table>、<tr>、<td>、<form>、<input>、
、<hr>。

如果读者还不清楚上面这些标记的含义，请先跟随本书所附光盘中的“基础知识视频教程”文件夹中的教学视频，学习一下最基础的知识，然后再开始学习本书。

此外，本书将不再占用篇幅抽象地讲解使用CSS布局所具有的优势，相信大部分读者对此已经有所了解。如果对这个问题感兴趣，可登录http://learning.artech.cn/category/css-div- web-design/cartoon-css-div来学习。其内容为1个小时生动有趣的视频课程，清晰地阐述了从传统布局方式到CSS布局的来龙去脉，并给出比较深入的分析。

本章将就一些在工作中，会遇到的实际问题进行讲解，作为学习CSS设计的必备基础。

1.1 HTML与XHTML

HTML与XHTML是一种语言还是两种语言？基本上可以认为，它们是一种语言的不同阶段，有点类似于文言文和白话文之间的关系。因此它们也经常被写作（X）HTML。下面首先从它们的渊源和区别开始本书的讲解。

1.1.1 追根溯源

（X）HTML是所有上网的人每天都离不开的基础，所有网页都是使用（X）HTML编写的。随着网络技术日新月异的发展，HTML也经历了不断的改进。可以认为XHTML是HTML的“严谨版”。

HTML在初期，为了它更广泛地被接受，大幅度放宽了标准的严格性，例如标记可以不封闭，属性可以加引号，也可以不加引号，等等，导致出现了很多混乱和不规范的代码。这显然不符合标准化的发展趋势，影响了互联网的进一步发展。

为此，相关规范的制订者——W3C组织，一直在不断地努力，逐步推出新的版本规范。从HTML到XHTML，大致经历了以下几个版本。

- HTML 2.0：于1995年11月发布。
- HTML 3.2：于1996年1月14日发布。
- HTML 4.0：于1997年12月18日发布。
- HTML 4.01（微小改进）：于1999年12月24日发布。
- XHTML 1.0：于2000年1月发布，后又经过修订于2002年8月1日重新发布。
- XHTML 1.1：于2001年5月31日发布。
- XHTML 2.0：正在制定中。

在正式的标准序列中，没有HTML 1.0版，这是因为在最初阶段，各个机构都推出了自己的方案，没有形成统一的标准。因此，W3C组织发布的HTML 2.0是形成标准以后的第一个正式规范。

这些规范实际上主要是为浏览器的开发者阅读的，因为他们必须了解这些规范的所有细节。而对于网页设计师来说，并不需要了解规范之间的细微差别，这与实际工作并不十分相关。因此，网页设计师通常只要知道一些大的原则就可以了。而且这些规范的文字也都比较晦涩，并不易阅读。当然，如果设计师真的能够花一些时间把HTML和CSS的规范仔细通读一遍，将会有巨大的收获。因为这些规范是所有设计师的“圣经”。

知识： W3C组织就是World Wide Web Consortium（全球万维网联盟）的简称。W3C组织创建于1994年，研究Web规范和指导方针，致力于推动Web发展，保证各种Web技术能很好地协同工作。W3C的主要职责是确定未来万维网的发展方向，并且制定相关的建议（Recommendation，由于W3C是一个民间组织，没有约束性，因此只提供建议）。

> HTML 4.01规范建议（HTML 4.01 Specification Recommendation）就是由W3C所制定的。它还负责制定CSS、XML、XHTML和MathML等其他网络语言规范。

1.1.2 DOCTYPE（文档类型）的含义与选择

从Dreamweaver MX 2004版开始，在使用Dreamweaver新建一个网页文档的时候，默认情况下生成的基本网页代码是这样的：

```
<!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=gb2312"
     />
     <title>无标题文档</title>
</head>
<body>
</body>
</html>
```

可以看到最上面有两行关于“DOCTYPE”（文档类型）的声明，它就是告诉浏览器，使用哪种规范来解释这个文档中的代码。

设计师可以在新建文档的时候选择使用哪种文档类型。在Dreamweaver的新建文档对话框中，在右下方有一个文档类型下拉框，如图1.1所示。

图1.1　在Dreamweaver中选择文档类型

对于HTML 4.01和XHTML 1.0分别对应于一种严格（Strict）类型和一种过渡（Transitional）类型。过渡类型兼容以前版本定义的，而在新版本已经废弃的标记和属性。严格类型则不兼容已经废弃的标记和属性。

注意 目前，建议读者使用XHTML 1.0transitional（XHTML 1.0过渡类型），这样设计师可以按照XHTML的标准书写符合Web标准的网页代码，同时在一些特殊情况下还可以使用传统的做法。

1.1.3 XHTML与HTML的重要区别

尽管目前浏览器都兼容HTML，但是为了使网页能够符合标准，设计师应该尽量使用XHTML规范来编写代码，需要注意以下事项。

1. 在XHTML中标记名称必须小写

在HTML中，标记名称可以大写或者小写。例如，下面的代码在HTML中是正确的。

```
<BODY>
    <P>这是一个文字段落</P>
</BODY>
```

但是在XHTML中，则必须写为：

```
<body>
    <p>这是一个文字段落</p>
</body>
```

2. 在XHTML中属性名称必须小写

HTML属性的名称也必须是小写的。例如，在XHTML中下面的代码的写法是错误的。

```
<IMG SRC="image.gif" WIDTH="200" HEIGHT="100" BORDER="0">
```

正确的写法应该是：

```
<img src="image.gif" width="200" height="100" border="0"/>
```

3. 在XHTML中标记必须严格嵌套

HTML中对标记的嵌套没有严格的规定。例如，下面的代码在HTML中是正确的。

```
<b><i>这行文字以粗体倾斜显示</b></i>
```

然而在XHTML中，必须改为：

```
<i><b>这行文字以粗体倾斜显示</b></i>
```

此外，经常被忽略的是对列表标记的嵌套写法。例如，下面的写法在HTML中是错误的。

```
<ul>
  <li>咖啡</li>
  <li>茶
    <ul>
      <li>红茶</li>
      <li>绿茶</li>
```

```
    </ul>
  <li>牛奶</li>
</ul>
```

正确的写法是：

```
<ul>
  <li>咖啡</li>
  <li>茶
    <ul>
      <li>红茶</li>
      <li>绿茶</li>
    </ul>
  </li>
  <li>牛奶</li>
</ul>
```

4. 在XHTML中标记必须封闭

在HTML规范中，下列代码是正确的。

```
<p> text line 1
<p> text line 2
```

上述代码中，第2个<p>标记就意味着前一个<p>标记的结束，但是在XHTML中，这是不允许的，而必须严格地使标记封闭，正确写法如下所示。

```
<p> text line 1</p>
<p> text line 2</p>
```

5. 在XHTML中，即使是空元素的标记也必须封闭

这里说的空元素的标记，就是指那些<img>、
等不成对的标记，它们也必须封闭，例如下面的写法是错误的。

```
换行<br>
水平线<hr>
图像<img src="happy.gif" alt="Happy face">
```

正确的写法应该是：

```
换行<br/>
水平线<hr/>
图像<img src="happy.gif" alt="Happy face"/>
```

6. 在XHTML中属性值用双引号括起来

在HTML中，属性可以不必使用双引号，例如：

```
<p class=heading>
```

而在XHTML中，必须严格写作：

```
<p class=" heading" >
```

7. 在XHTML中属性值必须使用完整形式

在HTML中，一些属性经常使用简写方式设定属性值，例如：

```
<input disabled>
```

而在XHTML中，必须完整地写作：

```
<input disabled=" true" >
```

8. 在XHTML中，应该区分“内容标记”与“结构标记”

例如<p>标记是一个内容标记，而<table>标记是结构标记，因此不允许将<table>标记置于<p>内部。而如果将<p>标记置于<td></td>之间，则是完全正确的。

有时这种错误不容易被注意到，在Dreamweaver中也得不到提示，但是在微软公司新推出的网页制作软件Expression Web中，则会给出明确的提示，如图1.2所示。

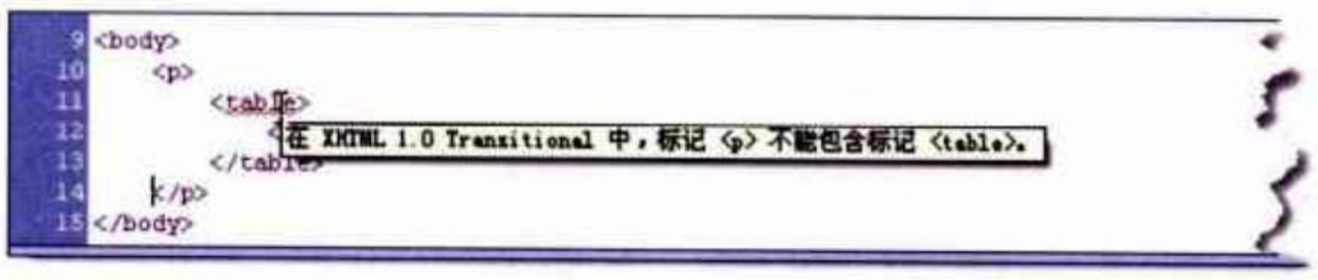


图1.2 在Expression Web中提示错误

在<table>标记的下方出现红色波浪线，表示存在错误。将鼠标指针移动到<table>标记上，则会出现提示文字“在XHTML 1.0 Transitional中，标记<p>不能包含标记<table>”。有兴趣的读者可以尝试一下这个新的网页设计软件。

1.2 （X）HTML与CSS

（X）HTML与CSS的关系就是“内容”与“形式”的关系，由（X）HTML确定网页的内容，而通过CSS来决定页面的表现形式。

1.2.1 CSS标准简介

和HTML类似，CSS也是由W3C组织负责制定和发布的。1996年12月，发布了CSS 1.0规范；1998年5月，发布了CSS 2.0规范。目前有两个新版本正在处于工作状态，即CSS 2.1版和CSS 3.0版。

图1.3所示的是2007年7月19日CSS 2.1发布的待批准的推荐版，读者可以到http://www.w3.org/TR/下载。

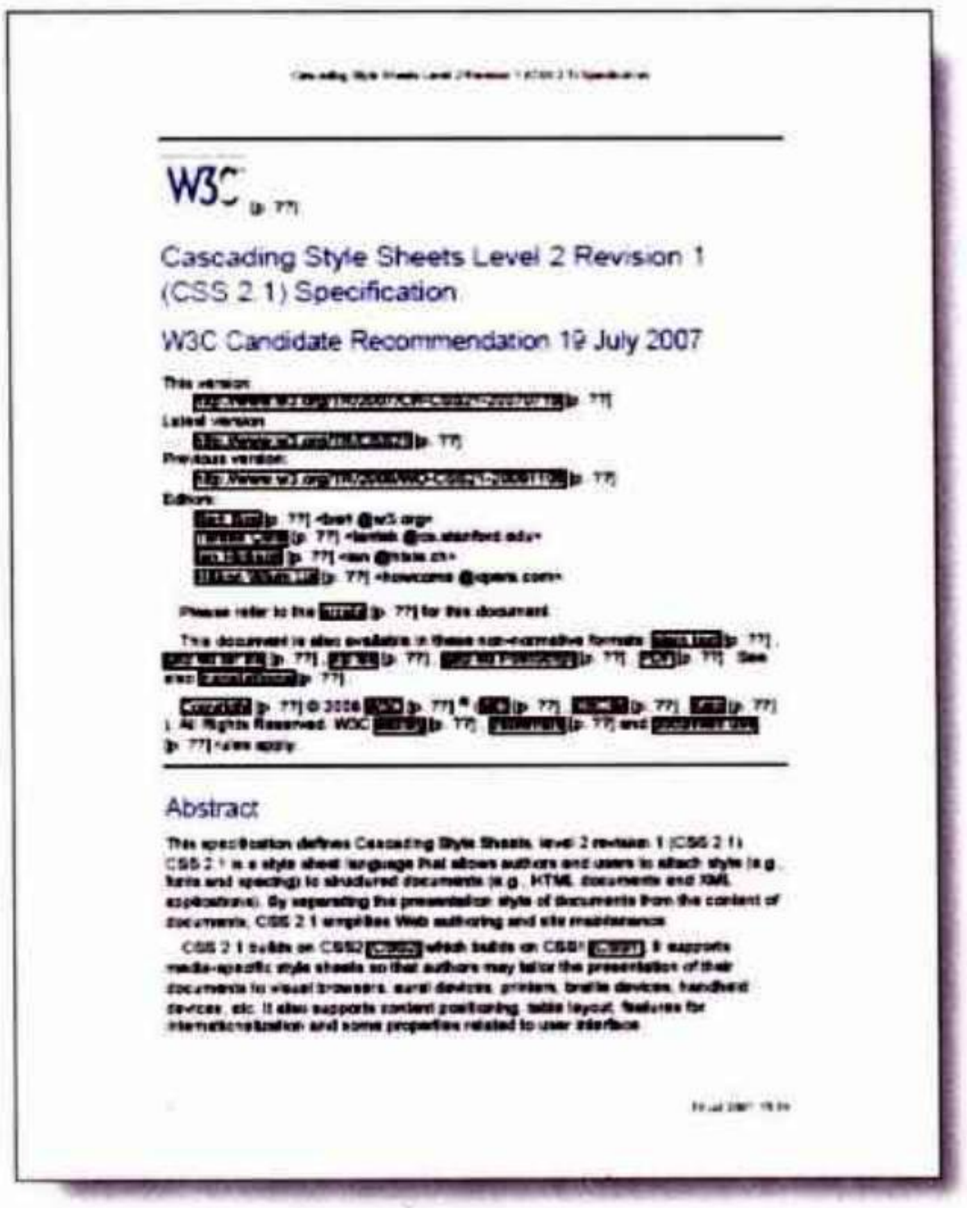


图1.3　WC组织发布的CSS 2.1规范工作稿

然而W3C并没有任何强制力要求软件厂商的产品必须符合规范，因此目前流行的浏览器都没有完全符合CSS的规范，这也就给设计师设计网页带来了一些难题。

但是随着技术的发展，各种浏览器都会逐渐在这方面做更多的努力，相信情况会越来越好。目前，最主流的3种浏览器是IE 6.0、IE 7.0和Firefox，它们在中国的使用率总和超过了99%，而以这3种浏览器为目标，已经完全可以做出显示非常一致的CSS布局页面。

在了解了XHTML与HTML之间的关系以后，为了便于讲解，本书在后面的讲解中都不再使用XHTML这个名词，而统一使用HTML，其含义为（X）HTML。

1.2.2　在HTML中引入CSS的方法

HTML与CSS是两个作用不同的语言，它们同时对一个网页产生作用，因此必须通过一些方法，将CSS与HTML挂接在一起，才能正常工作。

在HTML中，引入CSS的方法主要有行内式、内嵌式、导入式和链接式4种。

1. 行内式

行内式即在标记的style属性中设定CSS样式，这种方式本质上没有体现出CSS的优势，因此不推荐使用。

2. 嵌入式

嵌入式则将对页面中各种元素的设置集中写在<head>和</head>之间，对于单一的网页，

这种方式很方便。但是对于一个包含很多页面的网站，如果每个页面都以内嵌方式设置各自的样式，就失去了CSS带来的巨大优点，因此一个网站通常都是编写一个独立的CSS样式表文件，使用以下两种方式中的一种，引入HTML文档中。

3. 导入式与链接式

导入式和链接式的目的都是将一个独立的CSS文件引入HTML文件，二者的区别不大。这里给出一个比较深入的介绍，因为很多读者对此都有疑问。

事实上，二者最大的区别在于链接式使用HTML的标记引入外部CSS文件，而使用导入式则是使用CSS的规则引入外部CSS文件。因此它们的语法也不同。

如果使用链接式，需要使用如下语句引入外部CSS文件。

```
<link href="mystyle.css" rel="stylesheet" type="text/css" />
```

如果使用导入式，则需要使用如下语句。

```
<style type="text/css">
     @import"mystyle.css";
</style>
```

此外，采用这两种方式后的显示效果也略有区别。使用链接方式时，会在装载页面主体部分之前装载CSS文件，这样显示出来的网页从一开始就是带有样式效果的，而使用导入式时，会在整个页面装载完成后再装载CSS文件，对于有的浏览器来说，在一些情况下，如果网页文件的体积比较大，则会出现先显示无样式的页面，闪烁一下之后再出现设置样式后的效果。从浏览者的感受来说，这是使用导入式的一个缺陷。

对于一些比较大的网站，为了便于维护，可能会希望把所有的CSS样式分类别放到几个CSS文件中，这样如果使用链接式引入，就需要几个语句分别导入CSS文件。如果要调整CSS文件的分类，就需要同时调整HTML文件。这对于维护工作来说，是一个缺陷。如果使用导入式，则可以只引进一个总的CSS文件，在这个文件中再导入其他独立的CSS文件；而链接则不具备这个特性。

因此这里给大家的建议是，如果仅需要引入一个CSS文件，则使用链接方式；如果需要引入多个CSS文件，则首先用链接方式引入一个“目录”CSS文件，这个“目录”CSS文件中再使用导入式引入其他CSS文件。

但是如果希望通过JavaScript来动态决定引入哪个CSS文件，则必须使用链接方式才能实现。

这里给出一个完整的例子，演示各种导入方式的具体写法。为了在本书后面的章节中便于讲解，大多数情况使用内嵌式来完成，因为所举的例子通常就是一个独立的页面。

假设有如下页面代码。

```
<!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>Text Demo</title>
</head>
```

```
<body>
     <h1 style="color:white;background-color:blue">
        This is a line of Text.
  </h1>
</body>
</html>
```

代码中使用的是行内式，也就是直接在h1标记的style属性中设置CSS样式。这里将文字颜色设置为白色，背景颜色设置为蓝色，浏览器中的效果如图1.4所示。

图1.4 使用行内样式设置CSS

这种方式仅对这一个h1标题产生效果，因此如果希望页面中的所有h1标记都使用这种样式，就可以将代码改造为内嵌式。方法是把样式从行内移动到head部分，具体的代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Text Demo</title>
  <style type="text/css">
  h1{
        color:white;
        background-color:blue
        }
  </style>
</head>
<body>
  <h1>This is a line of Text.</h1>
  <h1>This is another line of Text.</h1>
</body>
</html>
```

在head部分，这里的h1就称为“选择器”，即选中了网页中的某些特定的元素，后面的样式规则和前面的行内规则的写法相同。

注意 每一条规则都要以分号结束，最后一条则不必以分号结束。

这样，页面中所有的h1标题都会按照这种效果显示，如图1.5所示。

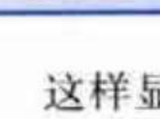
图1.5 使用内嵌方式设置CSS

如果希望把CSS的规则都写到一个外部独立的文件中，然后引入HTML，应再单独写一个文件，文件名的后缀为.css，内容如下：

```
h1{
     color:white;background-color:blue
}
```

然后将HTML文档中的<style></style>部分改为：

```
<style type="text/css">
  @import"mystyle.css";
</style>
```



注意 这里需要指定正确的CSS文件路径。

这样显示效果与上面的例子完全相同。如果要使用链接式引入这个CSS文件，可将上面的<style></style>部分删除，然后在head部分加入如下语句：

```
<link href="mystyle.css" rel="stylesheet" type="text/css" />
```

显示效果也是完全相同的。本节代码读者请参考本书附带光盘中的“第1章”中的01.htm至04.htm。

1.3 基本CSS选择器

选择器（selector）是CSS中很重要的概念，所有HTML语言中的标记样式都是通过不同的CSS选择器进行控制的。用户只需要通过选择器对不同的HTML标签进行选择，并赋予各种样式声明，即可实现各种效果。

为了理解选择器的概念，可以用“地图”作为类比。在地图上都可以看到一些“图例”，比如河流用蓝色的线表示，某公路用红色的线表示，省会城市用黑色圆点表示，等等。本质

上，这就是一种“内容”与“表现形式”对应关系。在网页上，也同样存在着这样的对应关系，例如h1标题用蓝色文字表示，h2标题用红色文字表示。因此为了使CSS规则与HTML元素对应起来，就必须定义一套完整的规则，实现CSS对HTML的“选择”。这就是叫做“选择器”的原因。

在CSS中，有几种不同类型的选择，本节先来介绍“基本”选择器。所谓“基本”，是相对于下一节中要介绍的“复合”选择器而言的。也就是说“复合”选择器是通过对基本选择器进行组合而构成的。

基本选择器有标记选择器、类别选择器和ID选择器3种，下面详细介绍。

1.3.1 标记选择器

一个HTML页面由很多不同的标记组成，CSS标记选择器用来声明哪些标记采用哪种CSS样式。因此，每一种HTML标记的名称都可以作为相应的标记选择器的名称。例如，p选择器就是用于声明页面中所有<p>标记的样式风格。同样可以通过h1选择器来声明页面中所有的<h1>标记的CSS风格，如下所示：

```
<style>
h1{
      color: red;
      font-size: 25px;
      }
</style>
```

以上这段CSS代码声明了HTML页面中所有的<h1>标记。文字的颜色都采用红色，大小都为25px。每一个CSS选择器都包含选择器本身、属性和值，其中属性和值可以设置多个，从而实现对同一个标记声明多种样式风格，如图1.6所示。

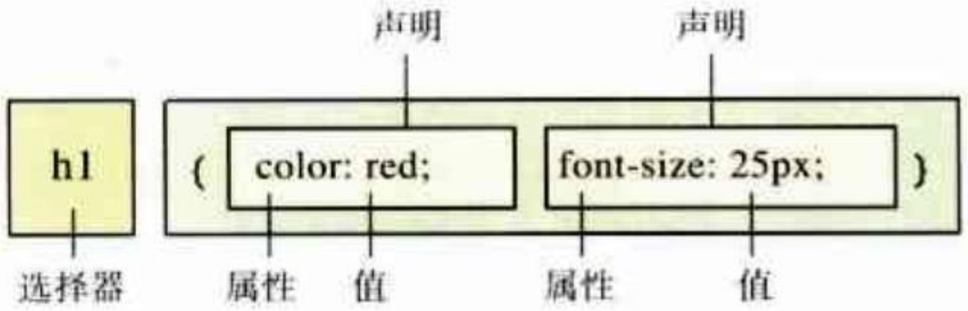


图1.6 CSS标记选择器

如果希望所有<h1>标记不再采用红色，而是蓝色，这时仅仅需要将属性color的值修改为blue，即可全部生效。

注意 CSS语言对于所有属性和值都有相对严格的要求，如果声明的属性在CSS规范中不存在，或者某个属性的值不符合该属性的要求，都不能使该CSS语句生效。下面是一些典型的错误语句：

```
Head-height: 48px; /* 非法属性 */
color: ultraviolet;        /* 非法值 */
```

对于上面提到的这些错误，通常情况下可以直接利用CSS编辑器（如Dreamweaver

> 或Expression Web）的语法提示功能避免，但某些时候还需要查阅CSS手册，或者直接登录W3C的官方网站（http://www.w3.org/）来查阅CSS的详细规格说明。

1.3.2 类别选择器

在1.3.1小节中提到的标记选择器一旦声明，那么页面中所有的该标记都会相应地产生变化。例如当声明了<p>标记为红色时，页面中所有的<p>标记都将显示为红色。但是如果希望其中的某一个<p>标记不是红色，而是蓝色，仅依靠标记选择器是不够的，还需要引入类别（class）选择器。

类别选择器的名称可以由用户自定义，属性和值跟标记选择器一样，也必须符合CSS规范，如图1.7所示。

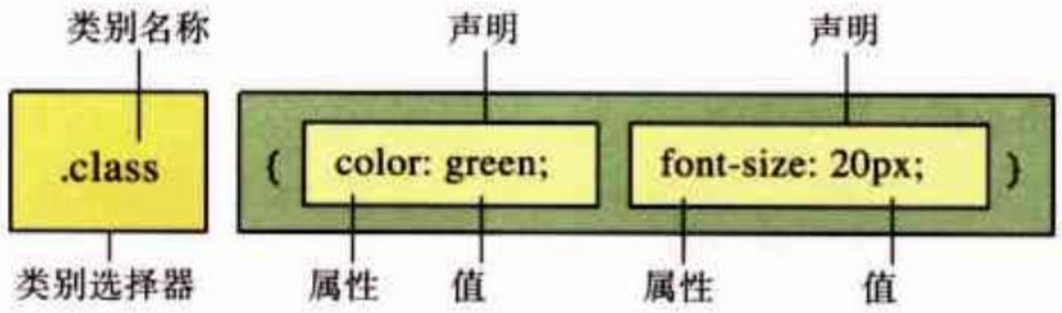


图1.7 类别选择器

例如，当页面中同时出现3个<p>标记时，如果想让它们的颜色各不相同，就可以通过设置不同的class选择器来实现。一个完整的案例如下所示，实例文件位于光盘中的“第1章\05.htm”。

```
<html>
<head>
<title>class选择器</title>
<style type="text/css">
.red{
  color:red;                    /* 红色 */
  font-size:18px;               /* 文字大小 */
}
.green{
  color:green;                  /* 绿色 */
  font-size:20px;               /* 文字大小 */
}
</style>
</head>

<body>
  <p class="red">class选择器1</p>
  <p class="green">class选择器2</p>
  <h3 class="green">h3同样适用</h3>
</body>
</html>
```

其显示效果如图1.8所示。从图中可以看到两个<p>标记分别呈现出了不同的颜色和字体大小，而且任何一个class选择器都适用于所有HTML标记，只需要用HTML标记的class属性声明即可，例如<h3>标记同样使用了.green这个类别。

图1.8 类别选择器示例

在上例中仔细观察还会发现，最后一行<h3>标记显示效果为粗体字，而也使用了.two选择器的第2个<p>标记却没有变成粗体。这是因为在.two类别中没有定义字体的粗细属性，各个HTML标记都采用了其自身默认的显示方式，<p>默认为正常粗细，而<h3>默认为粗体字。

再例如，很多时候页面中几乎所有的<p>标记都使用相同的样式风格，只有1~2个特殊的<p>标记需要使用不同的风格来突出，这时可以通过class选择器与1.3.1小节提到的标记选择器配合来实现。如下代码所示，示例文件位于光盘中“第1章\06.htm”。

```
<html>
<head>
<title>class选择器与标记选择器</title>
<style type="text/css">
p{                                          /* 标记选择器 */
  color:blue;
  font-size:18px;
}
.special{                                   /* 类别选择器 */
  color:red;                                /* 红色 */
  font-size:23px;                           /* 文字大小 */
}
</style>
</head>
<body>
  <p>class选择器与标记选择器1</p>
  <p>class选择器与标记选择器2</p>
  <p>class选择器与标记选择器3</p>
  <p class="special">class选择器与标记选择器4</p>
  <p>class选择器与标记选择器5</p>
  <p>class选择器与标记选择器6</p>
</body>
</html>
```

首先通过标记选择器定义<p>标记的全局显示方案，然后再通过一个class选择器对需要突出的<p>标记进行单独设置，这样大大提高了代码的编写效率，其显示效果如图1.9所示。

图1.9 两种选择器配合

在HTML的标记中，还可以同时给一个标记运用多个class类别选择器，从而将两个类别的样式风格同时运用到一个标记中。这在实际制作网站时往往会很有用，可以适当减少代码的长度，如下例所示，示例文件位于光盘中“第1章\07.htm”。

```
<html>
<head>
<title>同时使用两个class</title>
<style type="text/css">
.one{
  color:blue;              /* 颜色 */
}
.two{
  font-size:22px;          /* 字体大小 */
}
</style>
</head>
<body>
  <h4>一种都不使用</h4>
  <h4 class="one">同时使用两种class，只使用第一种</h4>
  <h4 class="two">同时使用两种class，只使用第二种</h4>
  <h4 class="one two">同时使用两种class，同时使用</h4>
  <h4>一种都不使用</h4>
</body>
</html>
```

显示效果如图1.10所示，可以看到使用第1种class的第2行显示为蓝色，而第3行则仍为黑色，但由于使用了.two，因此字体变大。第4行通过“class="one two"”将两个样式同时加入，得到蓝色大字体。第1行和第5行没有使用任何样式，仅作为对比时的参考。

图1.10 同时使用两种CSS风格

1.3.3 ID选择器

ID选择器的使用方法与class选择器基本相同：不同之处在于ID选择器只能在HTML页面中使用一次，因此其针对性更强。在HTML的标记中只需要利用id属性，就可以直接调用CSS中的ID选择器，其格式如图1.11所示。

图1.11 ID选择器

下面举一个实际案例，示例文件位于光盘中“第1章\08.htm”。

```
<html>
<head>
<title>ID选择器</title>
<style type="text/css">
#one{
  font-weight:bold;                    /* 粗体 */
  }
#two{
  font-size:30px;                      /* 字体大小 */
  color:#009900;                       /* 颜色 */
  }
</style>
</head>
<body>
  <p id="one">ID选择器1</p>
  <p id="two">ID选择器2</p>
  <p id="two">ID选择器3</p>
  <p id="one two">ID选择器3</p>
</body>
</html>
```

显示效果如图1.12所示，第2行与第3行都显示了CSS的方案。可以看出，在很多浏览器下，ID选择器可以用于多个标记，即每个标记定义的id不只是CSS可以调用，JavaScript等其他脚本语言同样也可以调用。因为这个特性，所以不要将ID选择器用于多个标记，否则会出现意想不到的错误。如果一个HTML中有两个相同id的标记，那么将会导致JavaScript在查找id时出错，例如函数getElementById（）。

正因为JavaScript等脚本语言也能调用HTML中设置的id，所以ID选择器一直被广泛地使用。网站建设者在编写CSS代码时，应该养成良好的编写习惯，一个id最多只能赋予一个HTML标记。

另外从图1.12中还可以看到，最后一行没有任何CSS样式风格显示，这意味着ID选择器不支持像class选择器那样的多风格同时使用，类似“id="one two"”这样的写法是完全错误的语法。

图1.12 ID选择器示例

1.4 复合选择器

1.3节介绍了3种基本的选择器，以这3种基本选择器为基础，通过组合，还可以产生更多种类的选择器，实现更强、更方便的选择功能，复合选择器就是基本选择器通过不同的连接方式构成的。

复合选择器就是两个或多个基本选择器，通过不同方式连接而成的选择器。

1.4.1 “交集”选择器

“交集”复合选择器由两个选择器直接连接构成，其结果是选中二者各自元素范围的交集。其中第一个必须是标记选择器，第二个必须是类别选择器或者ID选择器。这两个选择器之间不能有空格，必须连续书写，形式如图1.13所示。

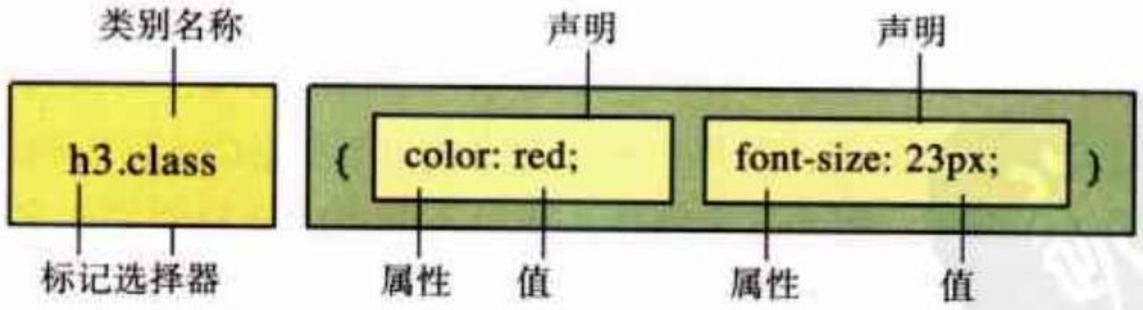


图1.13 标记类别选择器

这种方式构成的选择器，将选中同时满足前后二者定义的元素，也就是前者所定义的标记类型，并且指定了后者的类别或者id的元素，因此被称为“交集”选择器。

例如，声明了p、.special、p.special这3种选择器，它们的选择范围如图1.14所示。

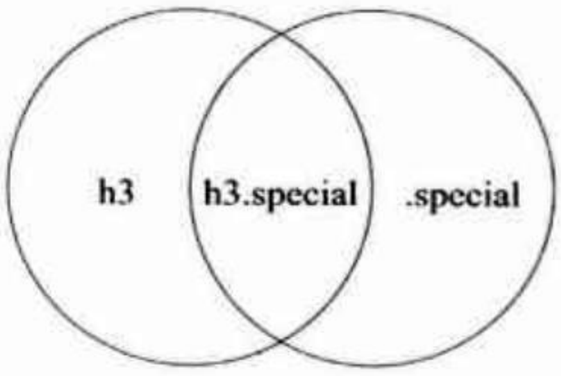


图1.14 交集选择器示意图

下面举一个实际案例，示例文件位于光盘中“第1章\09.htm”。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>选择器.class</title>
<style type="text/css">
p{                                    /* 标记选择器 */
  color:blue;
     }
p.special{                            /* 标记.类别选择器 */
  color:red;                          /* 红色 */
     }
.special{                             /* 类别选择器 */
  color:green;
     }
</style>
</head>
<body>
  <p>普通段落文本</p>
  <h3>普通标题文本</h3>
  <p class="special">指定了.special类别的段落文本</p>
  <h3 class="special">指定了.special类别的标题文本</h3>
</body>
</html>
```

上面的代码中定义了<p>标记的样式，也定义“.special”类别的样式，此外还单独定义了p.special，用于特殊的控制，而在这个p.special中定义的风格样式仅仅适用于<p class="special">标记，而不会影响使用了.special的其他标记，显示效果如图1.15所示。

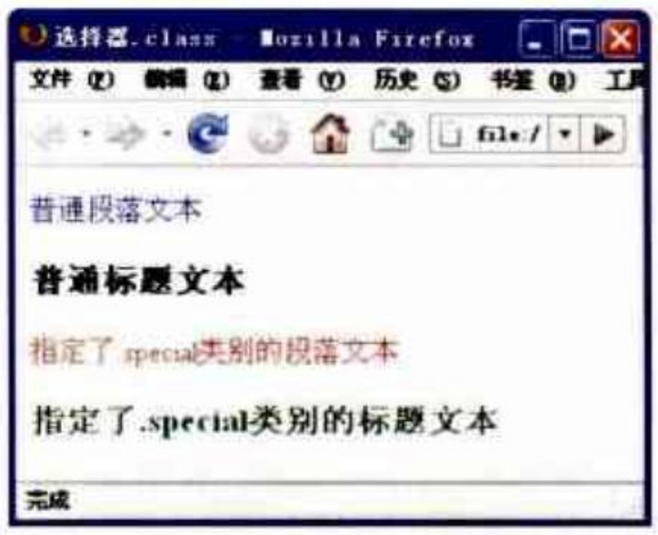


图1.15 标记.类别选择器示例

1.4.2

与交集选择器相对应，还有一种“并集”选择器，它的结果是同时选中各个基本选择器所选择的范围。任何形式的选择器（包括标记选择器、class类别选择器、ID选择器等）都可以作为并集选择器的一部分。

并集选择器是多个选择器通过逗号连接而成的，在声明各种CSS选择器时，如果某些选择器的风格是完全相同的，或者部分相同，这时便可以利用并集选择器同时声明风格相同的CSS选择器。其效果如图1.16所示。

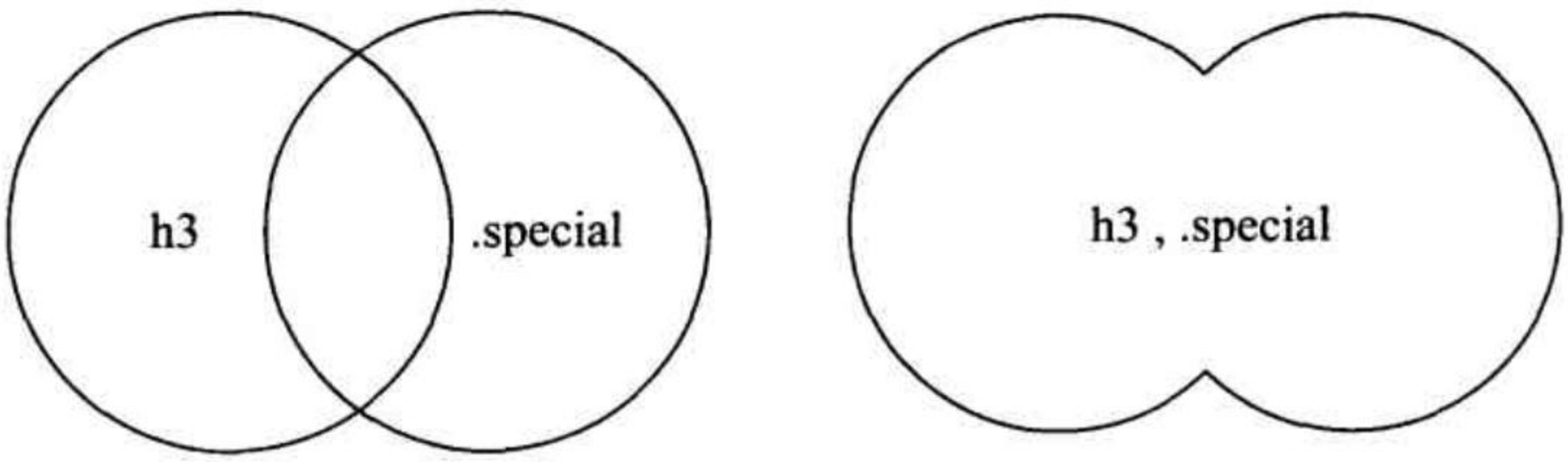


图1.16 并集选择器示意图

下面举一个实际案例，示例文件位于光盘中“第1章\10.htm”。

```
<html>
<head>
<title>并集选择器</title>
<style type="text/css">
h1, h2, h3, h4, h5, p{                    /*并集选择器*/
  color:purple;                           /* 文字颜色 */
  font-size:15px;                         /* 字体大小 */
}
h2.special, .special, #one{               /* 集体声明 */
  text-decoration:underline;              /* 下划线 */
}
</style>
</head>
<body>
  <h1>示例文字h1</h1>
  <h2 class="special">示例文字h2</h2>
  <h3>示例文字h3</h3>
  <h4>示例文字h4</h4>
  <h5>示例文字h5</h5>
  <p>示例文字p1</p>
  <p class="special">示例文字p2</p>
  <p id="one">示例文字p3</p>
</body>
</html>
```

其显示效果如图1.17所示，可以看到所有行的颜色都是紫色，而且字体大小均为15px。这种集体声明的效果与单独声明的效果完全相同，h2.special、.special和#one的声明并不

影响前一个集体声明，第2行和最后两行在紫色和大小为15px的前提下使用了下划线进行突出。

图1.17 集体声明

另外，对于实际网站中的一些页面，例如弹出的小对话框和上传附件的小窗口等，希望这些页面中所有的标记都使用同一种CSS样式，但又不希望逐个来声明的情况，可以利用全局选择器“*”，如下例所示，示例文件位于光盘中“第1章\11.htm”。

```
<html>
<head>
<title>全局声明</title>
<style type="text/css">
*{                                    /* 全局声明 */
  color:purple;                       /* 文字颜色 */
  font-size:15px;                     /* 字体大小 */
}
h2.special, .special, #one{           /* 集体声明 */
  text-decoration:underline;          /* 下划线 */
}
</style>
</head>
<body>
  <h1>全局声明h1</h1>
  <h2 class="special">全局声明h2</h2>
  <h3>全局声明h3</h3>
  <h4>全局声明h4</h4>
  <h5>全局声明h5</h5>
  <p>全局声明p1</p>
  <p class="special">全局声明p2</p>
  <p id="one">全局声明p3</p>
</body>
</html>
```

其效果如图1.18所示，与前面案例的效果完全相同，代码却大大缩减了。

图1.18 全局声明

1.4.3 后代选择器

在CSS选择器中，还可以通过嵌套的方式，对特殊位置的HTML标记进行声明，例如当 与 之间包含标记时，就可以使用后代选择器进行相应的控制。后代选择器的写法就是把外层的标记写在前面，内层的标记写在后面，之间用空格分隔。当标记发生嵌套时，内层的标记就成为外层标记的后代。

例如，假设有下面的代码：

```
<p>这是最外层的文字，<span>这是中间层的文字，<b>这是最内层的文字，
</b></span></p>
```

最外层是 标记，里面嵌套了标记，标记中又嵌套了<b>标记，则称span是p的子元素，b是span的子元素。

下面举一个完整的例子，具体代码如下所示，示例文件位于光盘中“第1章\12.htm”。

```
<html>
<head>
<title>后代选择器</title>
<style type="text/css">
p span{                                  /* 嵌套声明 */
  color:red;                             /* 颜色 */
}
span{
  color:blue;                            /* 颜色 */
}
</style>
</head>
<body>
  <p>嵌套使<span>用CSS</span>标记的方法</p>
  嵌套之外的<span>标记</span>不生效
</body>
</html>
```

通过将span选择器嵌套在p选择器中进行声明，显示效果只适用于 和 之间的标记，而其外的标记并不产生任何效果，如图1.19所示，只有第1行中和之间的文字变成了红色，而第2行文字中和之间的文字的颜色则是按照第2条CSS样式规则设置的，即为蓝色。

图1.19　嵌套选择器

后代选择器的使用非常广泛，不仅标记选择器可以以这种方式组合，类别选择器和ID选择器都可以进行嵌套。下面是一些典型的语句：

```
    .special i{ color: red; }                    /*  使用了属性special的标记里面
包含的<i> */
    #one li{ padding-left:5px; }          /* ID为one的标记里面包含的<li> */
    td.top .top1 strong{ font-size: 16px; }     /* 多层嵌套，同样实用 */
```

上面的第3行使用了3层嵌套，实际上更多层的嵌套在语法上都是允许的。上面的这个3层嵌套表示的就是使用了.top类别的<td>标记中包含的.top1类别的标记，其中又包含了标记，一种可能的相对应的HTML为：

```
<td class="top">
  <p class="top1">
        其他内容<strong>CSS控制的部分</strong>其他内容
  </p>
</td>
```

经验　选择器的嵌套在CSS的编写中可以大大减少对class和id的声明。因此在构建页面HTML框架时通常只给外层标记（父标记）定义class或者id，内层标记（子标记）能通过嵌套表示的则利用嵌套的方式，而不需要再定义新的class或者专用id。只有当子标记无法利用此规则时，才单独进行声明，例如一个<ul>标记中包含多个<li>标记，而需要对其中某个<li>单独设置CSS样式时才赋给该<li>一个单独id或者类别，而其他<li>同样采用“ul li{…}”的嵌套方式来设置。

需要注意的是，后代选择器产生的影响不仅限于元素的“直接后代”，而且会影响到它的“各级后代”。

例如，有如下的HTML结构：

```
  <p>这是最外层的文字，<span>这是中间层的文字，<b>这是最内层的文字，
</b></span></p>
```

如果设置了如下CSS样式：

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

那么“这是最外层的文字”这几个字将以黑色显示，即没有设置样式的颜色；后面的“这是中间层的文字”和“这是最内层的文字”，都属于它的后代，因此都会变成蓝色。

因此在CSS 2中，规范的制定者还规定了一种复合选择器，称为“子选择器”，也就是只对直接后代有影响的选择器，而对“孙子”以及多个层的后代不产生作用。

子选择器和后代选择的语法区别是，使用大于号连接。例如，将上面的CSS设置为：

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

则结果是仅有“这是中间层的文字”这几个字变为蓝色，因为span是p的直接后代，或者叫做“儿子”，b是p的“孙子”，不在选中的范围内。

而IE 6中，不支持子选择器，仅支持后代选择。IE 7和Firefox都既支持后代选择器，也支持子选择器。

1.5 CSS的继承特性

在本节中，对后代选择器的应用再一步作一些讲解，因为它将会贯穿在所有的设计中。

学习过面向对象语言的读者，对于继承（Inheritance）的概念一定不会陌生。在CSS语言中的继承并没有像在C++和Java等语言中的那么复杂，简单的说就是将各个HTML标记看作一个个容器，其中被包含的小容器会继承包含它的大容器的风格样式。本节从页面各个标记的父子关系出发，来详细地讲解CSS的继承。

1.5.1

所有的CSS语句都是基于各个标记之间的继承关系的，为了更好地理解继承关系，首先从HTML文件的组织结构入手，如下例所示，示例文件位于光盘中“第1章\13.htm”。

```
<html>
<head>
  <title>继承关系</title>
</head>
<body>
  <h1><em>前沿</em>教室</h1>
  <p>欢迎来到前沿教室，这里为您提供丰富的学习内容。</p>
  <ul>
```

```
<li>在这里，你可以学习到：
        <ul>
                <li>HTML</li>
                <li>CSS
                <ul>
                        <li>CSS初级</li>
                        <li>CSS中级</li>
                        <li>CSS高级</li>
                </ul>
                </li>
                        <li>Javascript</li>
                </ul>
        </li>
        <li>你还可以学习到：
                <ol>
                        <li>Flash</li>
                        <li>Dreamweaver</li>
                        <li>Photoshop</li>
                </ol>
        </li>
  </ul>
  <p>如果您有任何问题，欢迎联系我们</p>
</body>
</html>
```

相应的页面效果如图1.20所示。

图1.20 包含多层列表的页面

可以看到这个页面中，标题的前两个文字使用了（强调）标记，在浏览器中显示为斜体。后面使用了列表结构，其中最深的部分使用了三级列表。

这里着重从“继承”的角度来考虑各个标记之间的“树”型关系，如图1.21所示。在这个树型关系中，处于最上端的<html>标记称之为“根（root）”，它是所有标记的源头，往下层层包含。在每一个分支中，称上层标记为其下层标记的“父”标记；相应地，下层标记称为上层标记的“子”标记。例如<h1>标记是<body>标记的子标记，同时它也是的父标记。

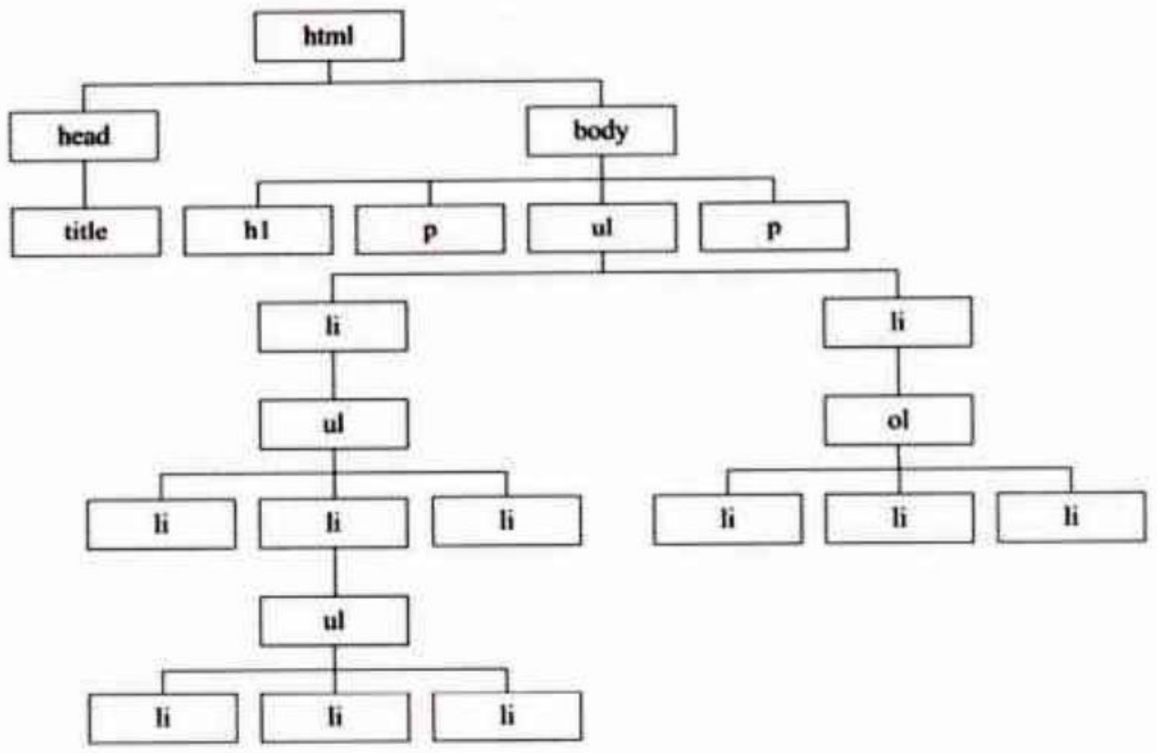


图1.21 继承关系树型图

1.5.2 CSS继承的运用

通过1.5.1节的讲解，已经对各个标记间的父子关系有了认识，下面进一步了解CSS继承的运用。CSS继承指的是子标记会继承父标记的所有样式风格，并可以在父标记样式风格的基础上再加以修改，产生新的样式，而子标记的样式风格完全不会影响父标记。

例如在前面的案例中加入如下CSS代码，即将h1标记设置为蓝色，加上下划线，并将em标记设置为红色，示例文件位于光盘中“第1章\14.htm”。

```
<style>
h1{
  color:blue;                                  /* 颜色 */
  text-decoration:underline;                   /* 下划线 */
  }
em{
  color:red;                                   /* 颜色 */
  }
</style>
```

显示效果如图1.22所示，可以看到其子标记em也显示出下划线，说明对父标记的设置也对子标记产生效果，而em文字显示为红色，h1标题中其他文字仍为蓝色，说明对子标记的设置不会对其父标记产生作用。

图1.22 父子关系示例

CSS的继承贯穿整个CSS设计的始终，每个标记都遵循着CSS继承的概念。可以利用这

种巧妙的继承关系，大大缩减代码的编写量，并提高可读性，尤其是在页面内容很多且关系复杂的情况下。

例如，现在如果要嵌套最深的第3级列表的文字显示为绿色，那么增加如下样式设置：

```
li{
   color:green;
   font-weight:bold;
}
```

效果将如图1.23所示，所有列表项目的文字都变成了绿色。要仅使“CSS”项下的最深的3个项目显示为绿色，其他项目仍显示为黑色，该怎么设置呢？

图1.23　各级列表均变成绿色

一种方法是设置单独的类别，比如定义一个“.green”类别，然后将该类别赋予需要变为绿色的项目，但是这样设置显然很麻烦。

可以利用继承的特性，使用前面介绍的“后代选择器”，这样不需要设置性的类别，即可完成同样的任务，效果如图1.24所示，示例文件位于光盘中“第1章\15.htm”。

```
ul li ul li ul li{
   color:green ;
   font-weight:bold;
}
```

图1.24　正确效果

实际上，对上面的选择器，还可以化简，比如化简为：

```
li li li{
   color:green ;
   font-weight:bold;
}
```

效果也是完全相同的。最后给读者出一个选择题，请读者思考，如果设置改为：

```
li li {
   color:green ;
   font-weight:bold;
}
```

那么在最终的效果中，绿色显示的文字一共有几行，备选答案为：

（A）3行

（B）6行

（C）8行

（D）9行

（E）10行

请读者一定要亲自在浏览器中实验一下，看一看结果如何，很可能和你想象的不同。如果和判断的结果不同，再仔细思考一下其中的原因。

1.6 CSS的层叠特性

作为本章的最后一节，这里主要讲解CSS的层叠属性。CSS的全名叫做“层叠样式表”，读者有没有考虑过，这里的“层叠”是什么意思？为什么这个词如此重要，以至于要出现在它的名称里。

CSS的层叠特性确实很重要，但是要注意，千万不要和前面介绍的“继承”相混淆，二者有着本质的区别。实际上，层叠可以简单地理解为“冲突”的解决方案。

例如有如下一段代码，示例文件位于光盘中“第1章\15.htm”。

```
<html>
<head>
<title>层叠特性</title>
<style type="text/css">
p{
  color:green;
  }
.red{
  color:red;
  }
```

```
.purple{
  color:purple;
  }
#line3{
  color:blue;
  }
</style>
</head>
<body>
  <p >这是第1行文本</p>
  <p class="red">这是第2行文本</p>
  <p id="line3" class="red">这是第3行文本</p>
  <p style="color:orange;" id="line3">这是第4行文本</p>
  <p class="purple red">这是第5行文本</p>
</body>
</html>
```

代码中一共有5组p标记定义的文本，并在head部分声明了4个选择器，声明为不同颜色。下面的任务是确定每一行文本的颜色。

- 第1行文本没有使用类别样式和ID样式，因此这行文本显示为标记选择器p中定义的绿色。
- 第2行文本使用了类别样式，因此这时已经产生了“冲突”。那么，是按照标记选择器p中定义的绿色显示，还是按照类别选择器中定义的红色显示呢？答案是类别选择器的优先级高于标记选择器，因此显示为类别选择器中定义的红色。
- 第3行文本同时使用了类别样式和ID样式，这又产生了“冲突”。那么，是按照类别选择器中定义的红色显示，还是按照ID选择器中定义的蓝色显示呢？答案是ID选择器的优先级高于类别选择器，因此显示为ID选择器中定义的蓝色。
- 第4行文本同时使用了行内样式和ID样式，那么这时又以哪一个为准呢？答案是行内样式的优先级高于ID样式的优先级，因此显示为行内样式定义的橙色。
- 第5行文本中，使用了两个类别样式，应以哪个为准呢？答案是两个类别选择器的优先级相同，此时以前者为准，“.purple”定义在“.red”的前面，因此显示为“.purple”中定义的紫色。

因此，综上所述，上面这段代码的显示效果如图1.25所示。

图1.25 层叠特性示意图

总结 优先级规则可以表述为：

行内样式 > ID样式 > 类别样式 > 标记样式

在复杂的页面中，某一个元素有可能会从很多地方获得样式，例如一个网站的某一级标题整体设置为使用绿色，而对某个特殊栏目需要使用蓝色，这样在该栏目中就需要覆盖通用的样式设置。在很简单的页面中，这样的特殊需求实现起来不会很难，但是如果网站的结构很复杂，就完全有可能使代码变得非常混乱，可能出现无法找到某一个元素的样式来自于哪条规则的情况。因此，必须要充分理解CSS中“层叠”的原理。

注意 计算冲突样式的优先级是一个比较复杂的过程，并不仅仅是上面这个简单的优先级规则可以完全描述的。但是读者可以把握一个大的原则，就是“越特殊的样式，优先级越高”。

例如，行内样式仅对指定的一个元素产生影响，因此它非常特殊；使用了类别的某种元素，一定是所有该种元素中的一部分，因此它也一定比标记样式特殊；以此类推，ID是针对某一个元素的，因此它一定比应用于多个元素的类别样式特殊。特殊性越高的元素，优先级就越高。

最后再次提醒读者，千万不要混淆了层叠与继承，二者完全不同。

1.7 本章小结

本章重点介绍了4个方面的问题。先介绍了HTML和XHTML的发展历程以及需要注意的问题，然后介绍了如何将CSS引入HTML，接着讲解了CSS的各种选择器，及其各自的使用方法，最后重点说明了CSS的继承与层叠特性，以及它们的作用。

作为CSS设计的核心基础，请读者务必真正搞懂这些最基础和核心的基本原理。

Chapter

第2章
“CSS禅意花园”作品鉴赏

在网站设计，特别是在CSS设计领域，有一个世界范围都非常著名的网站——CSS Zen Garden，中文名称为“CSS禅意花园”。这个网站以最有效、最优美的方式展示了CSS的最高境界。

任何一个学习CSS的人都不应该错过对CSS禅意花园的研究。本章将从禅意花园网站上的近千个作品中精选16个作品进行介绍，它们的HTML结构和内容是完全相同的，通过不同的CSS设置，就做出了完全不同的效果。

2.1 "CSS禅意花园"简介

CSS禅意花园是一位加拿大设计师Dave Shea创建的，他在网站设计的实际工作中逐渐认识到CSS的巨大潜力远远没有被发掘出来。为了推动CSS设计的发展，他在2003年建立了"CSS禅意花园"这个网站（网址是http://www.csszengarden.com）。他知道仅仅依靠自己的力量是无法发掘出CSS的全部潜力的，因此采用了征集作品的方式，通过大家的努力来展现CSS的魅力。

首先Dave Shea精心设计了一个网页，把这个网页的HTML结构和内容固定下来，为这个网页设计了5个完全不同主题风格的CSS样式，然后通过网站的发布，邀请全世界的广大设计师参与CSS禅意花园的作品设计，征集作品。

对征集的作品要求是必须完全使用他提供的这个HTML结构和内容，不能做改动，要保证通过调用设计师提供的CSS文件来展现作品。

读者在网址http://www.zengarden.com的后面加上编号，就可以查看相应的作品，例如要查看005号作品，输入网址http://www.zengarden.com/?cssfile=005/005.css即可。

1. 网站作品简介

Dave Shea自己制作的是001~005号作品。其中001号作品名为"Tranquille"（安静），这个作品几年来一直被作为CSS Zen Garden网站的首页，如图2.1所示。

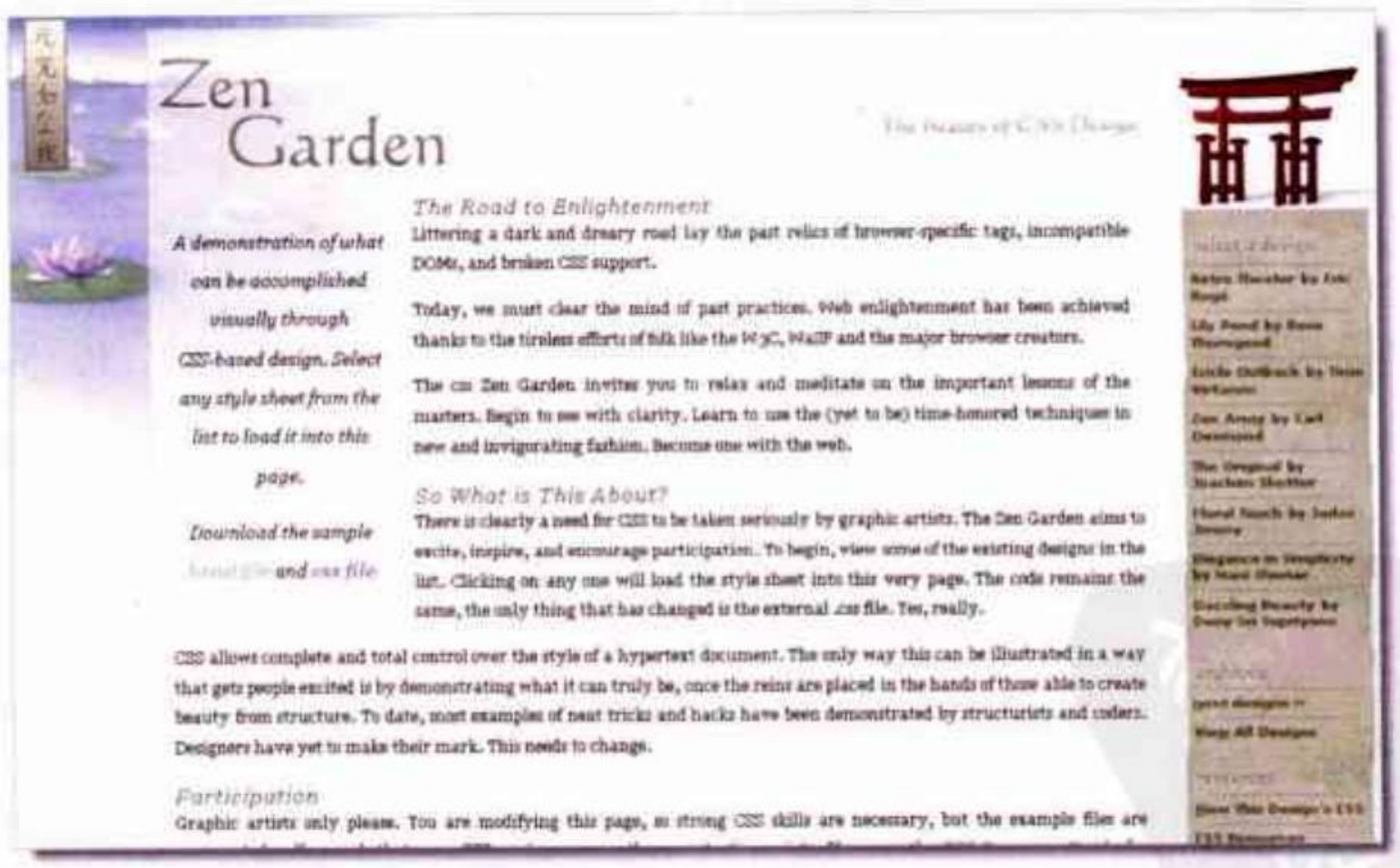


图2.1 001号禅意花园作品

002号作品名为"Salmon Cream Cheese"（鲑鱼奶油奶酪），如图2.2所示。

003号作品名为"Stormweather"（暴风雪），如图2.3所示。

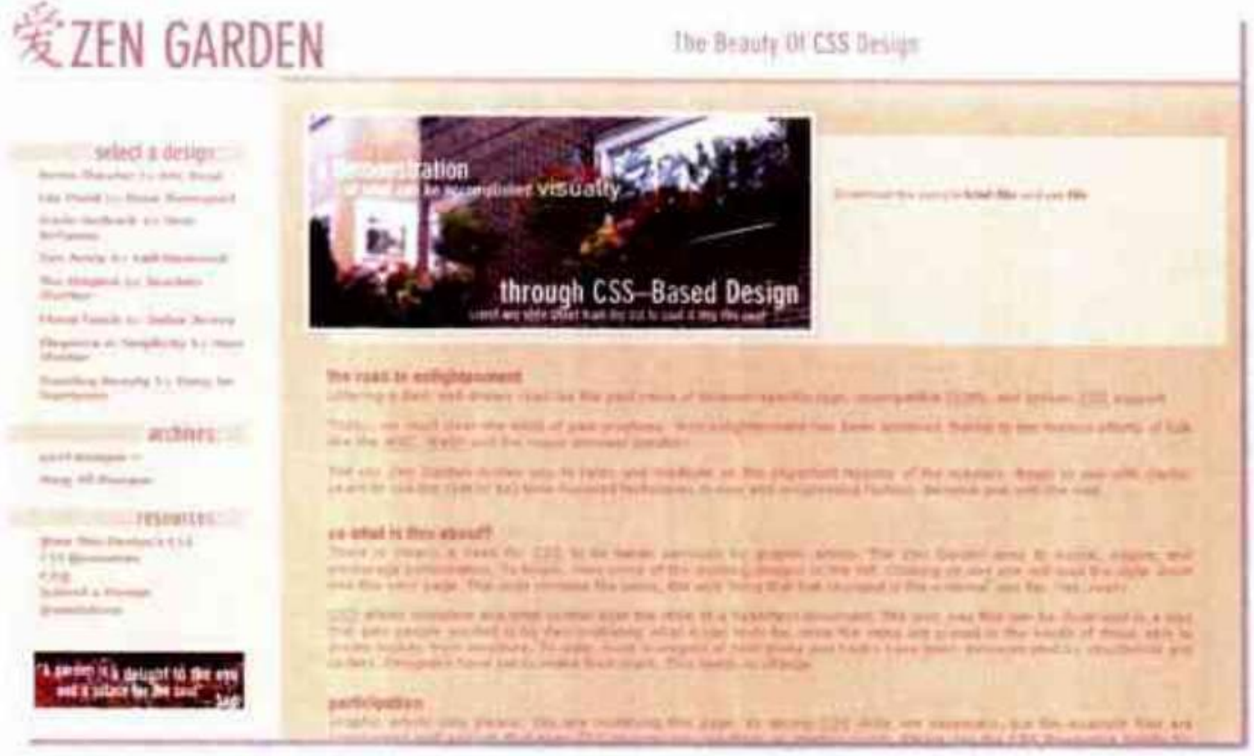


图2.2　002号禅意花园作品

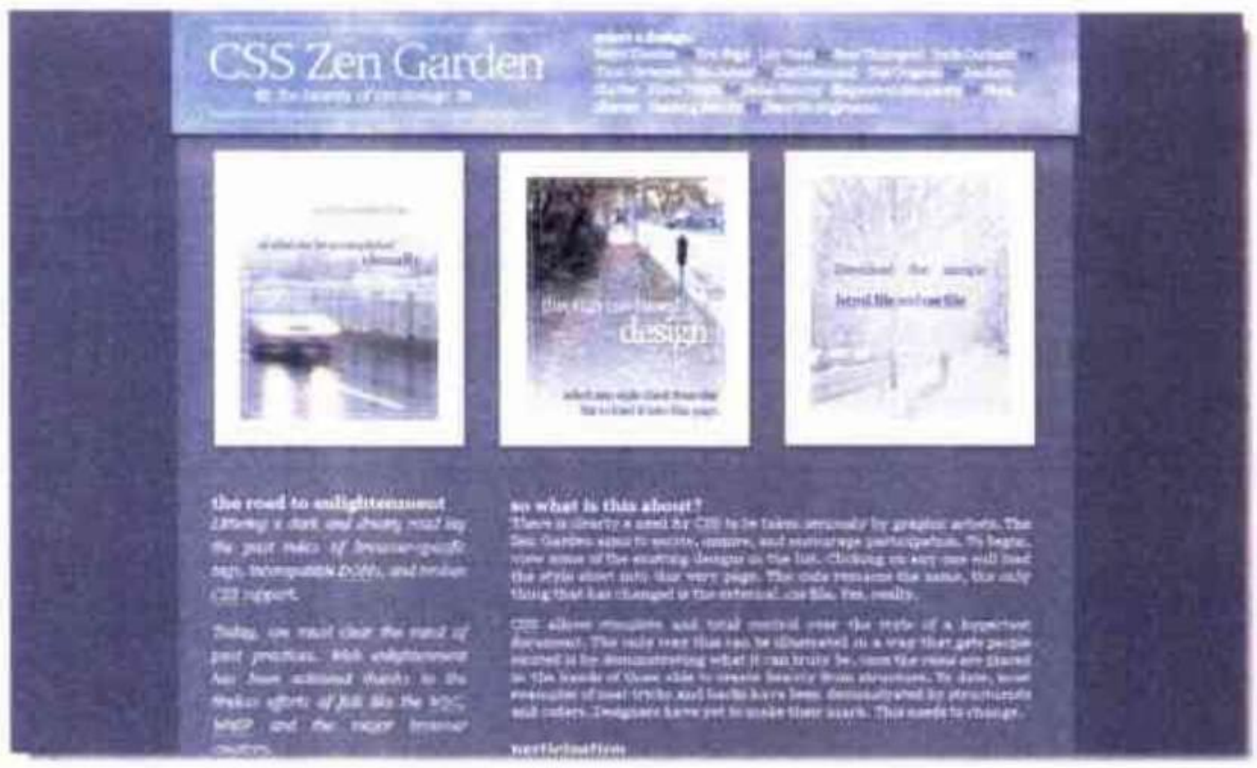


图2.3　003号禅意花园作品

004号作品名为"arch4.20"（拱门），如图2.4所示。

图2.4　004号禅意花园作品

005号作品名为“Blood Lust”（血色欲望），如图2.5所示。

图2.5 005号禅意花园作品

从006号开始，都是世界各地的设计师的作品。目前在csszengarden.com上收录了1000多个作品，它们都充分体现了设计师的丰富的创意，给无数学习CSS的人带来了启发。

注意 在禅意花园目前收录的1000多个作品中，有202个被作为“官方作品”保存在禅意花园的网站上，其余的作品以链接的方式链接到禅意花园的网页目录上。每个“官方作品”都有编号，其他的作品没有编号。

注意 2005年，Dave和一位美国的设计师Molly合作，对CSS禅意花园网站中的若干作品进行介绍、分析和评论，出版了一本书《the Zen of CSS design》，介绍了CSS设计的理念和指导思想。这本书已经被翻译成了十多个国家的文字出版，2007年这本书的中文版也出版了，有兴趣的读者可以参考，中文版书名为《CSS禅意花园》。

2. 为何选择CSS禅意花园

在编写本书时，我们想到如果能够从CSS禅意花园的作品中挑选出典型作品案例，分析并介绍它们的设计思路和制作方法给广大读者，对读者的帮助将会是很大的，这样读者可以学习这些特定案例的设计方法，更重要的是能够熟悉并掌握设计思路，可以从其他数百个作品中寻找并获得灵感。

我们与Dave Shea取得联系之后，他非常高兴地同意我们使用CSS禅意花园中的作品作为本书中的案例进行分析。

在本书中，共安排了4章内容与禅意花园相关，分别是本章和全书的最后3章。

本章中将介绍一些各具特色的作品，为大家解读CSS禅意花园的作品设计方法，使大家通过这些作品更好地了解CSS的巨大潜力。这些作品的设计师来自欧美近10个国家，在每个作品中都附有创作者的个人网站信息，读者如果有兴趣，可以进一步了解他们的工作和生活。

本书的最后一章将挑选其中的典型作品，进行深入的分析，在学习完本书之后，禅意花园中的绝大部分作品读者都应该可以自己制作出来。希望读者在学习完本书之后，能够设计出自己的“禅意花园”作品，甚至被收录为禅意花园的“官方作品”（目前亚洲还只有新加坡和日本等国家的设计师的很少的几个作品被收录）。

2.2 郊野——两列布局

两列布局是所有网站中最常见的布局形式，也是禅意花园的作品中最常见到的。这种布局结构清晰，对访问者的引导性非常好。《郊野》是这类作品的代表，由意大利设计师Mario Carboni设计。在这个作品中，页面划分合理，并且颜色搭配平和而协调，如图2.6所示。

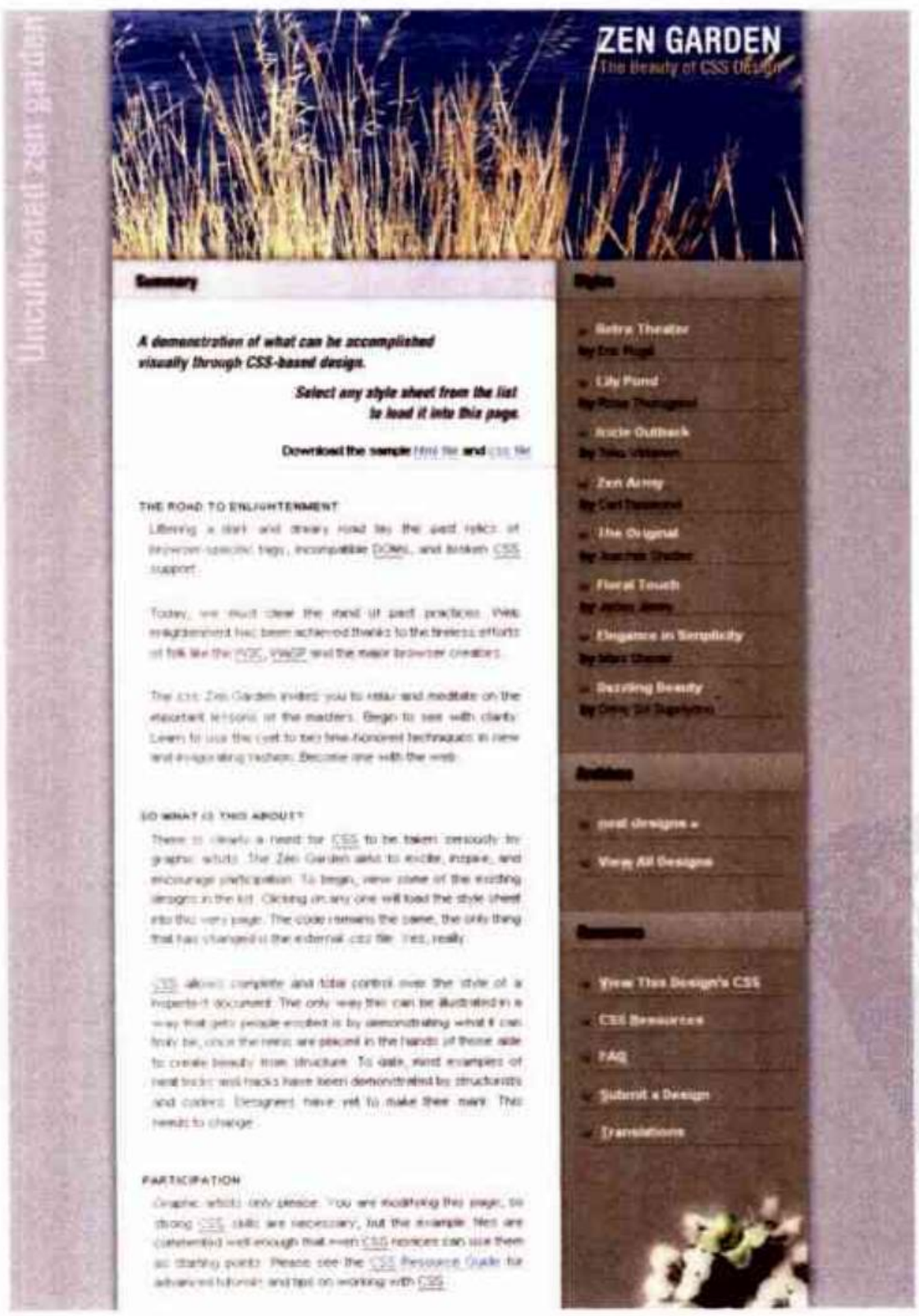


图2.6 149号禅意花园作品

访问这个作品的网址是http://www.csszengarden.com/?cssfile=149/149.css，设计师的个人网站是http://www.mariocarboni.com/。

2.3 像素画——三列布局

这是193号作品，如图2.7所示，由英国设计师Jon Tan设计。这是一个很典型的三列布局设计，设计师专门制作了精致的像素画，作为页面的标题图像，效果相当吸引人。

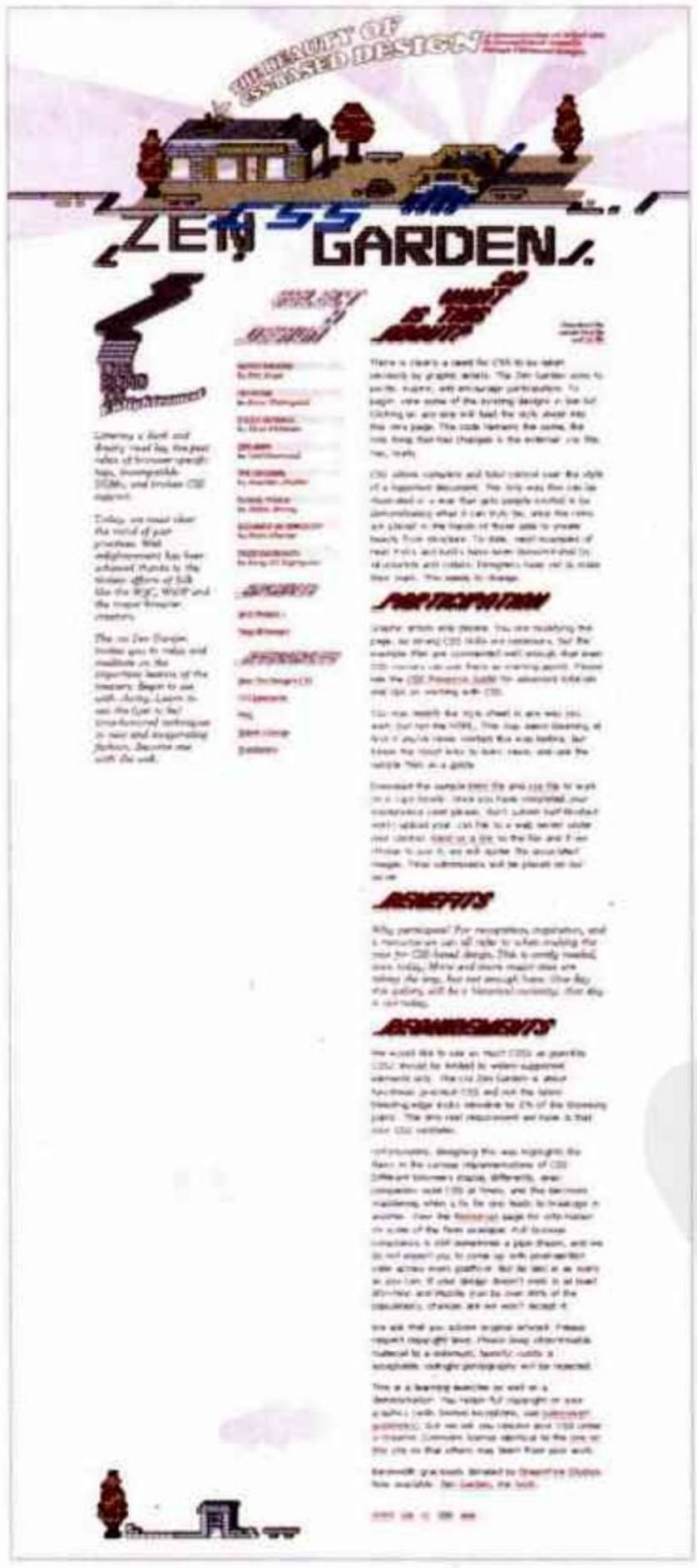


图2.7 193号禅意花园作品

三列布局也是一种非常常见的布局形式，通常一个宽列放置主要内容，两个窄列放置导航链接等内容。访问这个作品的网址是http://www.csszengarden.com/?cssfile=193/193.css，设计师的个人网站是http://www.gr0w.com/。

2.4 百合池塘——三列布局变体

这是201号作品，如图2.8所示，由澳大利亚设计师Rose Thorogood设计。这是一个三列布局的变体形式，在顶部和底部，相邻的两列进行了合并。

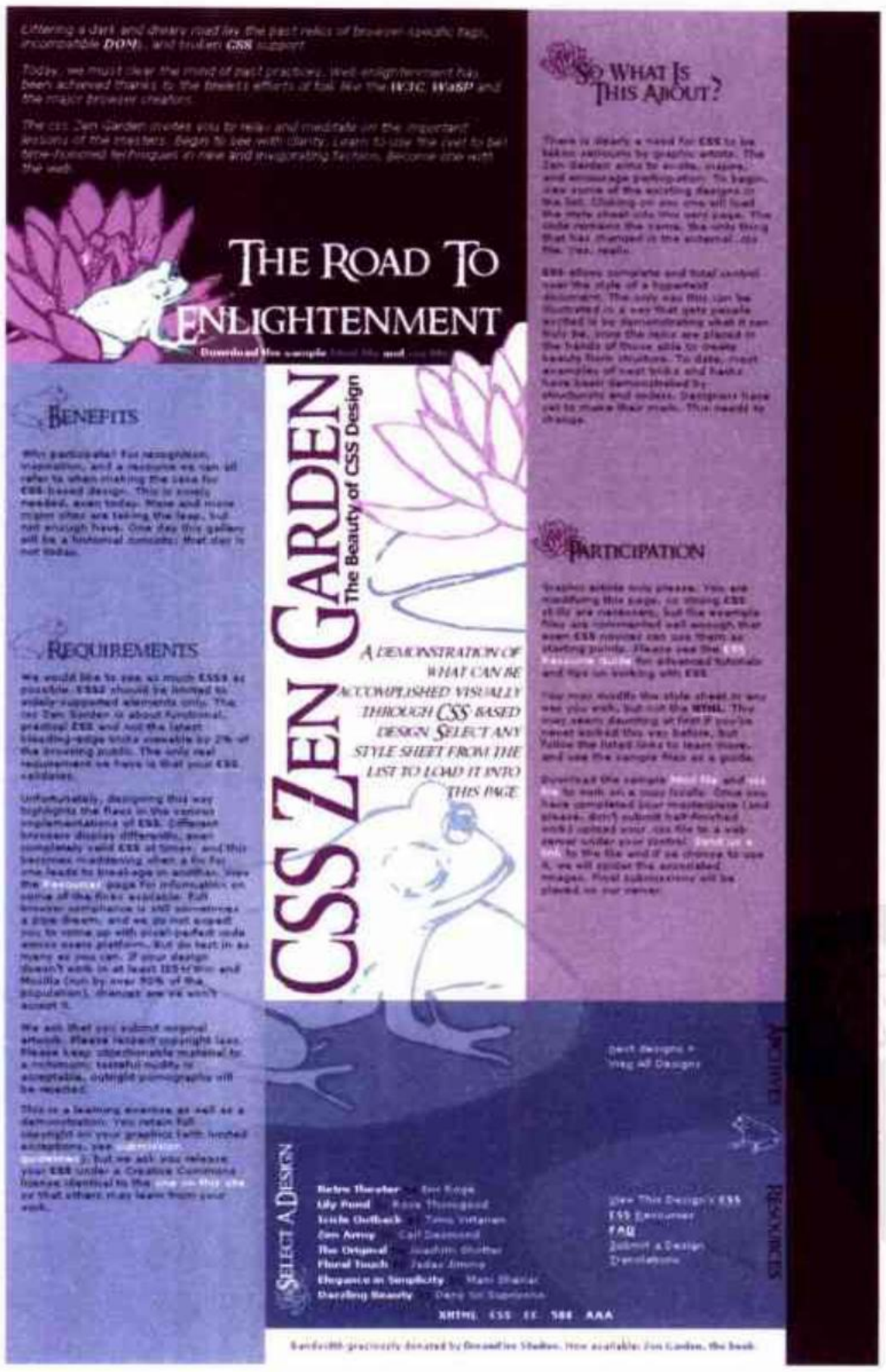


图2.8 201号禅意花园作品

访问这个作品的网址是http://www.csszengarden.com/?cssfile=201/201.css，设计师的个人网站是http://tulips4rose.com。

2.5 郁金香——多列布局

禅意花园征集作品要求使用固定的HTML，这个限制使大多数设计方案都采用两列或三列布局，而实际上也可以制作出非常好的多列效果。在收录的作品中，也有一些非常优秀的多列布局作品，088号作品就是典型代表，如图2.9所示，由美国设计师Eric Shepherd设计。图中没有显示完整的页面，实际页面中右侧还有其他列内容。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=088/088.css，设计师的个人网站是http://arkitrave.com/log。

图2.9 088号禅意花园作品

2.6 日与夜——包含圆角的设计

CSS的框模型限制，使得在使用CSS设计页面的初期，大多数作品都是方形的，缺少圆形和曲线的配合，显得比较呆板。实际上，使用CSS同样可以设计出完美的圆形图形元素的作品。这里展示的这个作品就使用了包含圆角的设计，由比利时设计师Colas Schretter设计，效果如图2.10所示。

访问这个作品的网址是http://homepages.vub.ac.be/~cschrett/zengarden/dayandnight，设计师的个人网站是http://homepages.vub.ac.be/~cschrett。

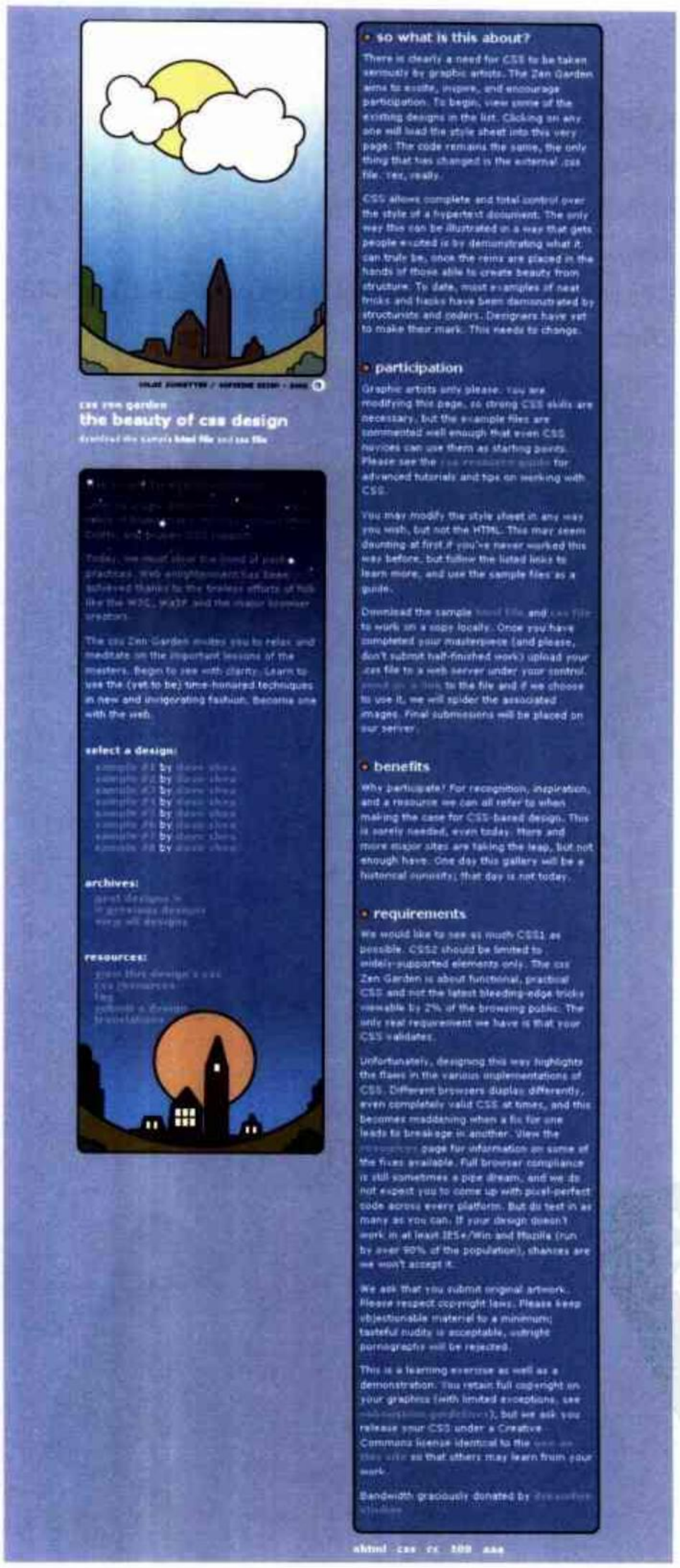


图2.10 禅意花园作品

2.7 Si6——包含倾斜的设计

绝大多数网页的布局都是横平竖直的，而禅意花园网站中可以看到一些包含了倾斜元素的设计作品，不但没有破坏页面的整体平衡感，而且还为页面增添了独特的气质，使访问者感到耳目一新。比较典型的倾斜设计作品是044号作品，效果如图2.11所示，它是由美国设计师Shaun Inman设计的。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=044/044.css，设计师的个人网站是http://www.shauninman.com。

图2.11 044号禅意花园作品

2.8 清茶时光——装饰性设计

再好的设计如果忽视了对细节的把握，也会大打折扣的。使用CSS不但能够布局，还可以为网页添加很多装饰性的元素，使页面看起来非常精致，细节更加突出。这里展示的124号作品是由奥地利设计师Michaela Maria Sampl设计的，她通过使用一系列协调的装饰性元素，使页面的效果和谐而美观，充分体现女性设计师作品的柔美风格，效果如图2.12所示。

图2.12 124号禅意花园作品

访问这个作品的网址是http://www.csszengarden.com/?cssfile=124/124.css，设计师的个人网站是http://www.frausampl.at。

2.9 爱之空气——流体布局

目前，绝大多数网站，尤其是正规的商业网站，都使用固定宽度的布局，CSS禅意花园的作品也是如此，但是仍有一部分适应宽度的作品，或者称为“流体布局”。例如这里展示的170号作品，是由德国设计师Nele Goetz设计的，效果如图2.13所示，页面分为两列，总的宽度会撑满整个页面，左列宽度会随浏览器宽度而变化，右列宽度固定。

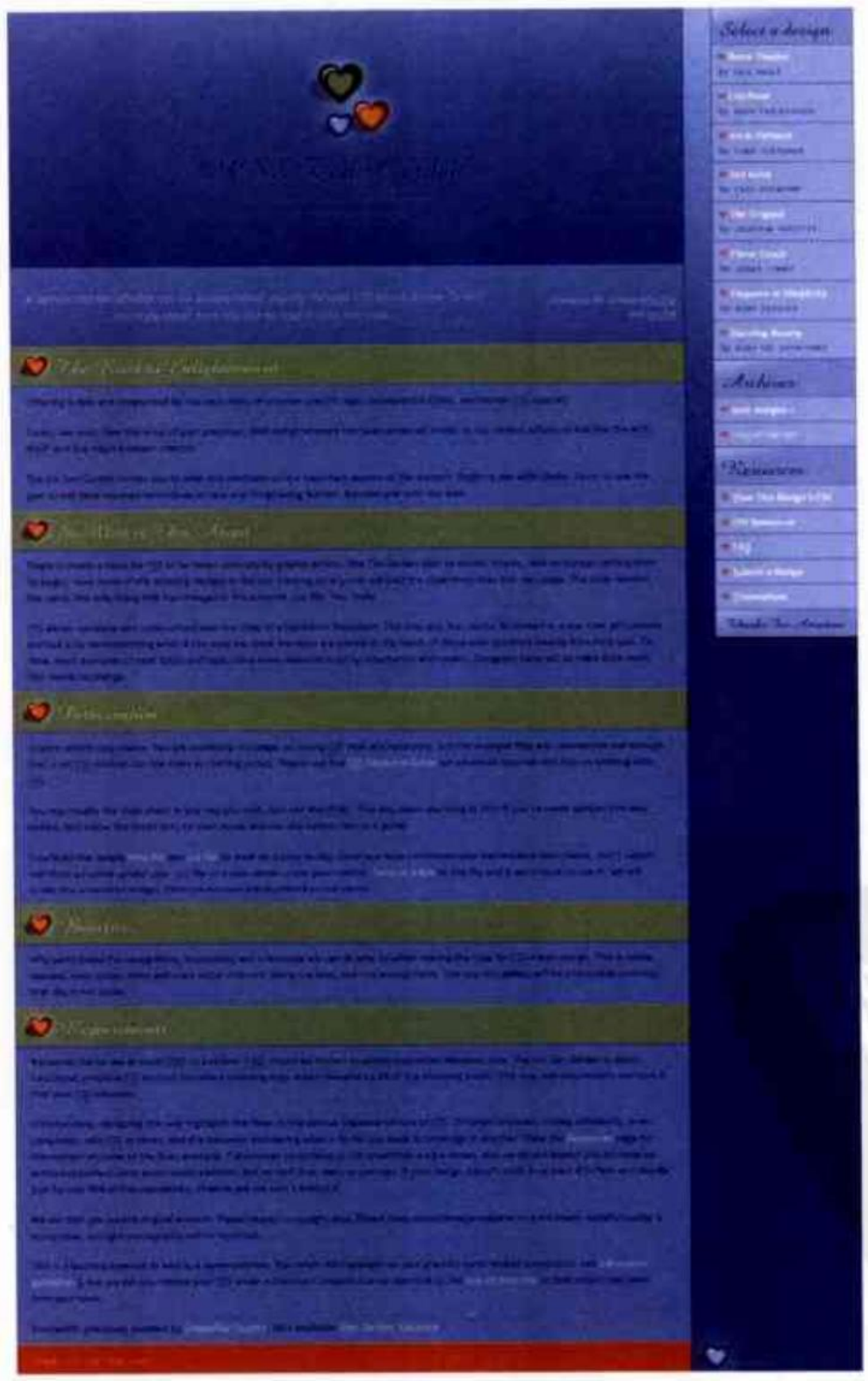


图2.13 170号禅意花园作品

访问这个作品的网址是http://www.csszengarden.com/?cssfile=170/170.css，设计师的个人网站是http://www.april-design.de。

2.10 谷香——食品主题设计

在禅意花园的作品中，还有一些是围绕某一个主题进行设计的。例如这里展示的057号作品就是以食品为主体的设计，效果如图2.14所示，它是由美国设计师Shaun Inman设计的。通过精致的食品照片，是不是能勾起访问者的食欲呢?

此外，这个设计中还设计了非常漂亮的弹出菜单，但是这个效果在Firefox中可以正常显示，而在IE 6和IE 7中都无法显示菜单的效果。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=057/057.css，设计师的个人网站是http://www.shauninman.com/。

图2.14 057号禅意花园作品

2.11 城市——建筑主题设计

与上个案例类似，本案例围绕建筑这个主题进行设计，采用了建筑的不同透视角度，并组织在一个页面中，具有很强的空间感，效果如图2.15所示。它是由加拿大设计师Matt Kim和Nicole设计的。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=146/146.css。

图2.15 禅意花园作品

2.12 "卡通版"禅意花园——特色效果

099号作品完全不使用通常网页采用的形式，而使用卡通连环画的表现形式，别具匠心。完整作品分为若干页，图2.16所示的是第1页。它由澳大利亚设计师Joseph Pearson设计的。该作品的网址是http://www.csszengarden.com/?cssfile=099/099.css。

设计师的个人网站是http://www.make-believe.org。

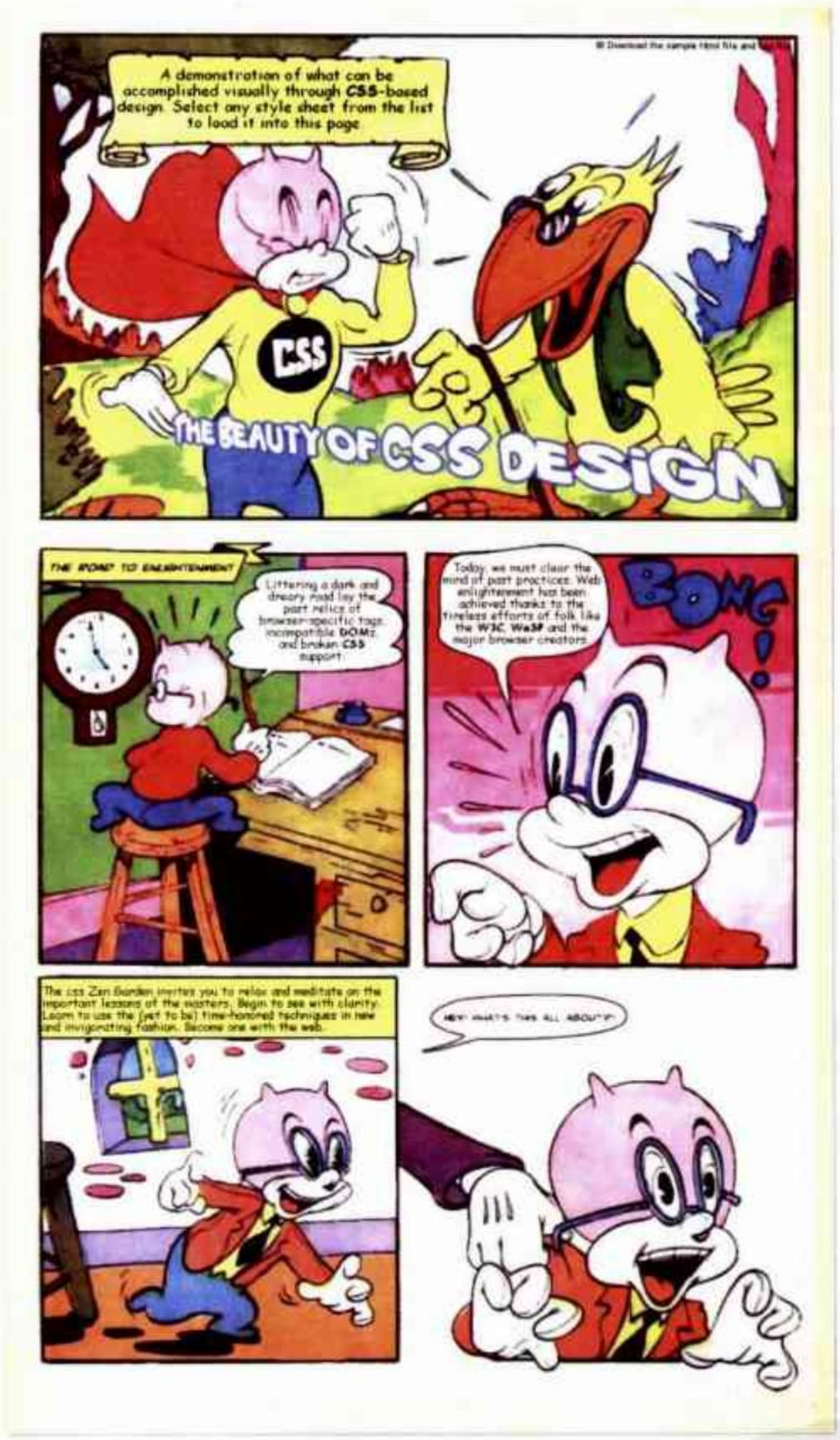


图2.16　099号禅意花园作品

2.13 收音机——特色效果

059号作品的特色在于它不是采用常见的竖直延伸的布局方式，而采用横向分列，更有

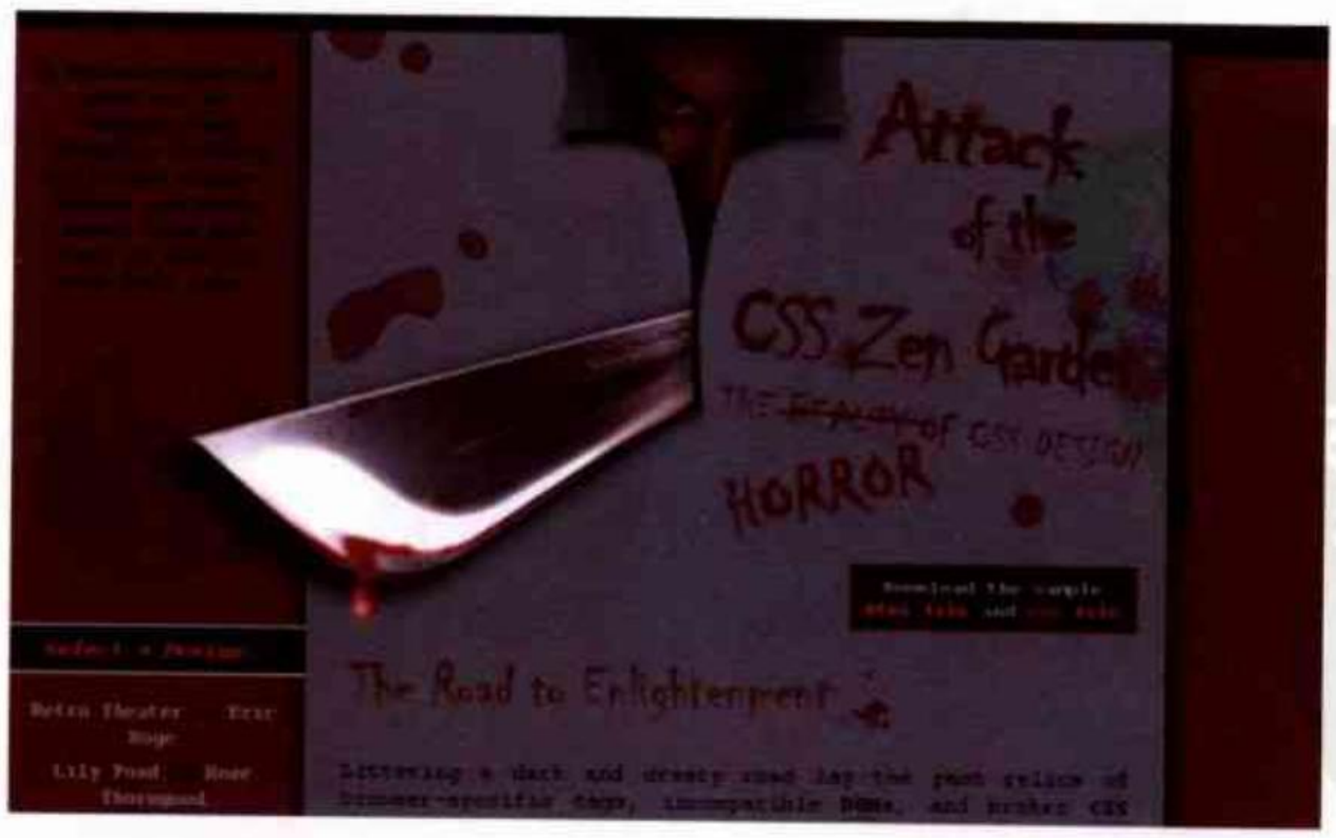
趣的是，在拖动浏览器下侧的滚动条时，页面上侧的指针会移动，就好像在为收音机调台一样，这个创意非常独特，效果如图2.17所示。该作品的网址是http://www.csszengarden.com/?cssfile=059/059.css。

它是由荷兰设计师Marc LA van den Heuvel设计的。设计师的个人网站是http://www.mlavdh.nl。

图2.17 059号禅意花园作品

2.14 杀手风格——特色效果

这里展示的是一个非官方作品，效果如图2.18所示。这个作品初看上去风格有些另类，但是如果拖动一下浏览器的滚动条，就会发现它很有趣。



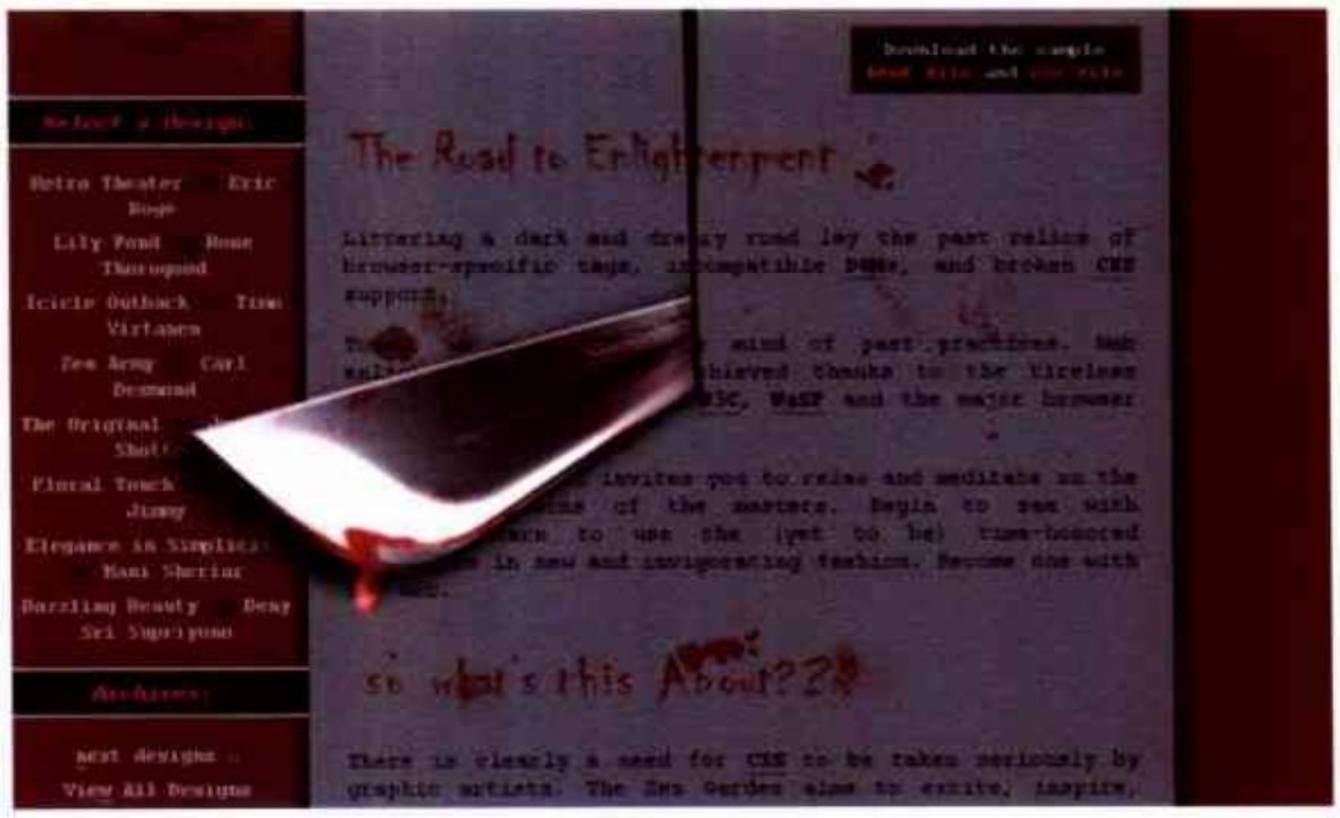


图2.18 禅意花园作品

在图中可以看，有一把刀把页面“切”开了一个豁口，当访问者拖动浏览器的滚动条继续向下浏览时，这把刀会把页面整个切开。这个创意令人叫绝！

需要注意的是，这个效果只有在Firefox和IE 7中可以正确显示，在IE 6中无效。

该作品的访问网址是http://adjustafresh.com/zen，由美国设计师Scott Kiekbusch设计，他的个人网站是http://www.adjustafresh.com。

2.15 海底世界——特色效果

这件作品在CSS Zen Garden近千件作品中，可以称得上是最“神奇”的作品。效果如图2.19所示，它表现了一个精致的海底世界，页面很长，浏览器窗口中只能显示很小的一部分。无论如何拖动浏览器的滚动条，页面中的“潜水员”都会在相同的地方，也就是说“潜水员”会在“海水中”上浮或下潜。页面上部的背景比较浅，这意味着在海面附近，越往下“海洋”的背景颜色就会越深，意味着潜水员在不断地下潜。最神奇的是，“潜水员”手中的探照灯竟然可以随着下潜的深度不断变亮。当浏览器页面滚动到海底时，会冒出气泡，还有小螃蟹会跑出来。图中仅为示意，无法表现出完整的变化过程，读者如果有兴趣，可以实际在网上访问一下这个作品。

需要注意，这个网页在FireFox中的效果最好，在IE 7中虽然可以看出探照灯光的变化，但是由于这个页面使用了特殊技术，背景颜色显示不如在FireFox中自然。在IE 6中，没有任何效果。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=http://www.css-praxis.de/cssocean/zenocean.css，由德国设计师Kai Laborenz设计，他的个人网站是http://www.css-praxis.de。

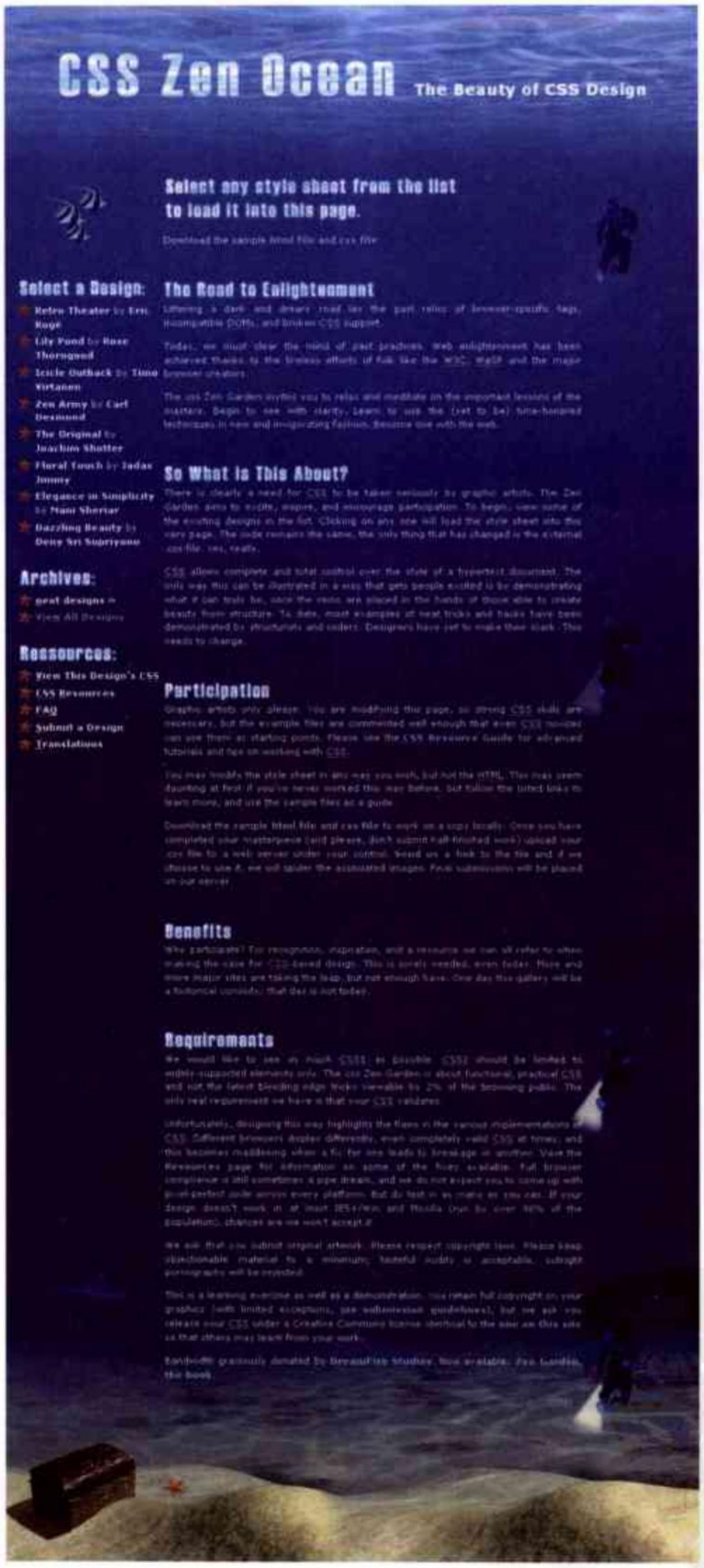


图2.19 禅意花园作品

2.16 博物馆——特色设计

148号作品也是一个很有意思的设计，由比利时设计师Samuel Marin设计。他把整个页面设计为一个博物馆大楼的剖面图，可以看到很多楼层的藏品，而每一个藏品正好就是页面中的一项内容，这个构思非常巧妙。页面效果如图2.20所示。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=148/148.css，设计师的个人网站是http://www.info.fundp.ac.be/~sma/。

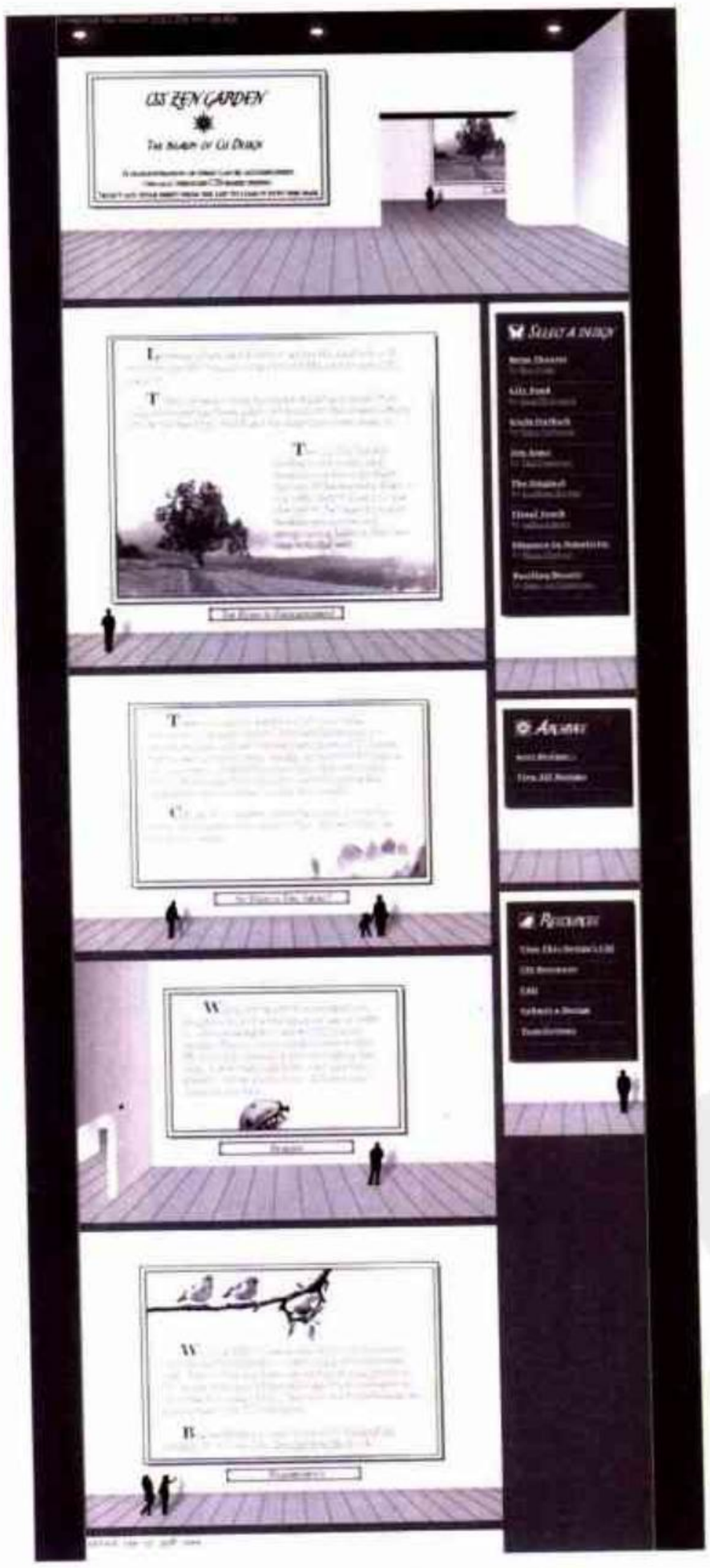


图2.20　148号禅意花园作品

2.17 剧院效果——特色效果

202号作品由法国设计师Eric Rogé设计。作品的剧院效果非常有趣，当拖动浏览器的滚动条时，网页内容就像电影"银幕"上的文字一样向上滚动，极富创意，效果生动。页面效果如图2.21所示。

需要注意，这个效果只有在Firefox和IE 7中可以正确显示，在IE 6中无效。

访问这个作品的网址是http://www.csszengarden.com/?cssfile=202/202.css，设计师的个人网站是http://space-sheeps.info。

图2.21 202号禅意花园作品

2.18 本章小结

在本章中，读者可以欣赏到20多个非常精彩的"CSS禅意花园"网站中的作品。同一个HTML文件，仅仅通过不同的CSS设置，就产生了如此丰富多彩的页面效果，可见CSS对于网页设计的重要性，因此它是网页设计师手中最得力的一个武器。在本书后面的章节中，将深入CSS设计的方方面面，希望读者在学习完本书以后，不但能够制作出本章介绍的这些页面效果，更可以设计和创造出更精彩的网页。

第 3 章
深入理解盒子模型

盒子模型是CSS控制页面时一个很重要的概念。只有很好地掌握了盒子模型以及其中每个元素的用法，才能真正地控制好页面中的各个元素。本章主要介绍盒子模型的基本概念，并讲解CSS定位的基本方法。

所有页面中的元素都可以看成是一个盒子，占据着一定的页面空间。一般来说这些被占据的空间往往都要比单纯的内容大。换句话说，可以通过调整盒子的边框和距离等参数，来调节盒子的位置和大小。

一个页面由很多这样的盒子组成，这些盒子之间会互相影响，因此掌握盒子模型需要从两方面来理解。一是理解一个孤立的盒子的内部结构，二是理解多个盒子之间的相互关系。

在本章中首先讲解独立的盒子相关的性质，然后将介绍在普通情况下盒子的排列关系。在下一章中，将更深入地讲解浮动和定位的相关内容。

3.1 盒子的内部结构

在学习盒子模型之前，先来看一个例子。假设在墙上有4幅画，整齐地排列着，如图3.1所示。对于每幅画来说，都有一个“边框”，在英文中称为“border”；每个画框中，画和边框通常都会有一定的距离，这个距离称为“内边距”，在英文中称为“padding”；各幅画之间通常也不会紧贴着，它们之间的距离称为“外边距”，在英文中称为“margin”。

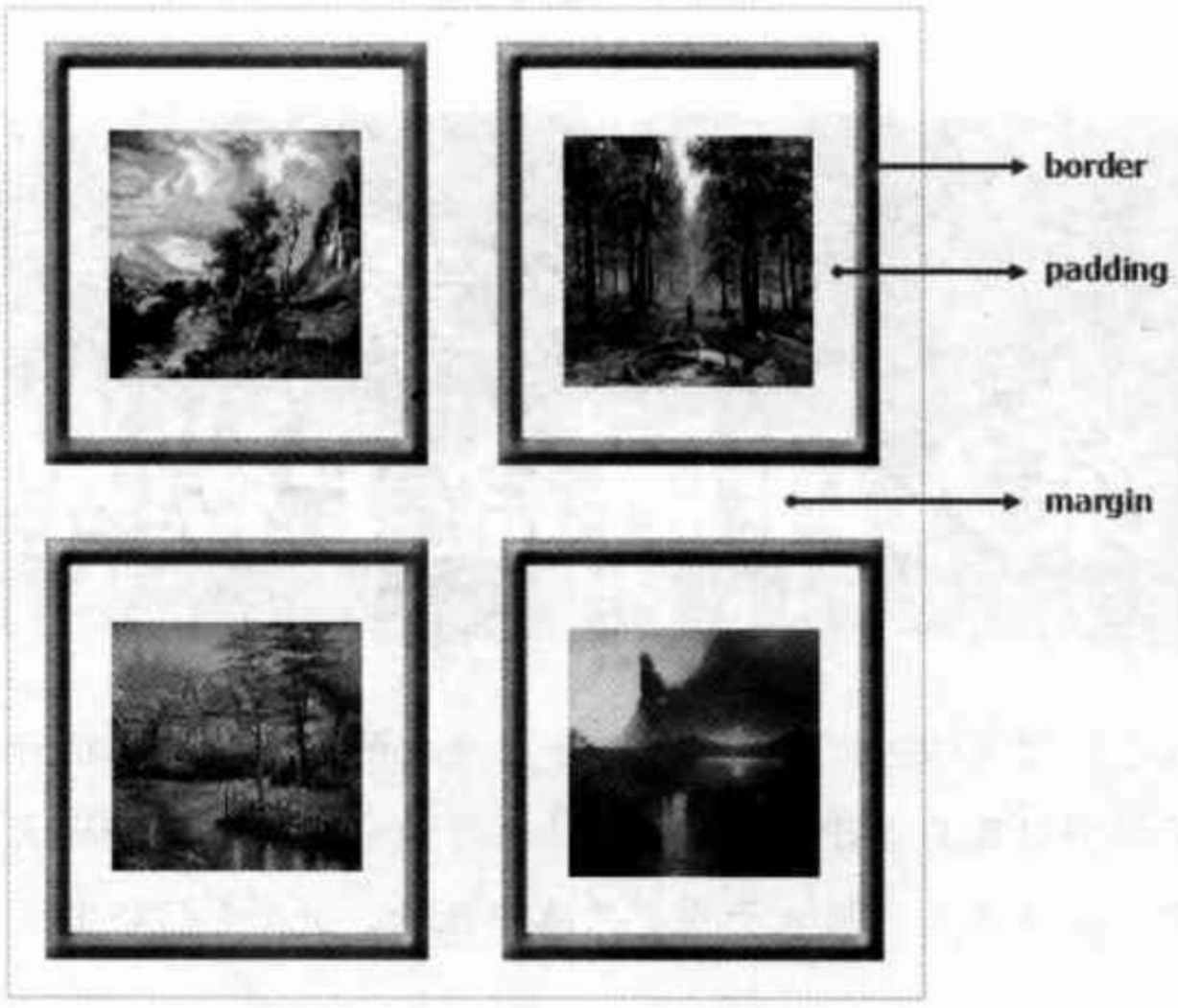


图3.1 画框示意图

这种形式实际上存在于我们生活中的各个地方，如电视机、显示器和窗户等，都是这样的。因此，padding-border-margin模型是一个极其通用的描述矩形对象布局形式的方法。这些矩形对象可以被统称为“盒子”，英文为“Box”。

了解了盒子之后，还需要理解“模型”这个概念。所谓模型就是对某种事物的本质特性的抽象。

模型的种类很多，例如物理上有“物理模型”，科学家牛顿提出了著名的F=ma公式，就是对物体运动的本质特性进行抽象后的精确描述。类似地，创办企业也有其“商业模型”，它是对企业运作和盈利过程的本质进行抽象后，对它的描述。

同样，在网页布局中，为了能够使纷繁复杂的各个部分合理地进行组织，这个领域的一些有识之士对它的本质进行充分的研究后，总结出了一套完整的、行之有效的原则和规范。这就是“盒子模型”的由来。

在CSS中，一个独立的盒子模型由content（内容）、border（边框）、padding（内边距）和margin（外边距）4个部分组成，如图3.2所示。

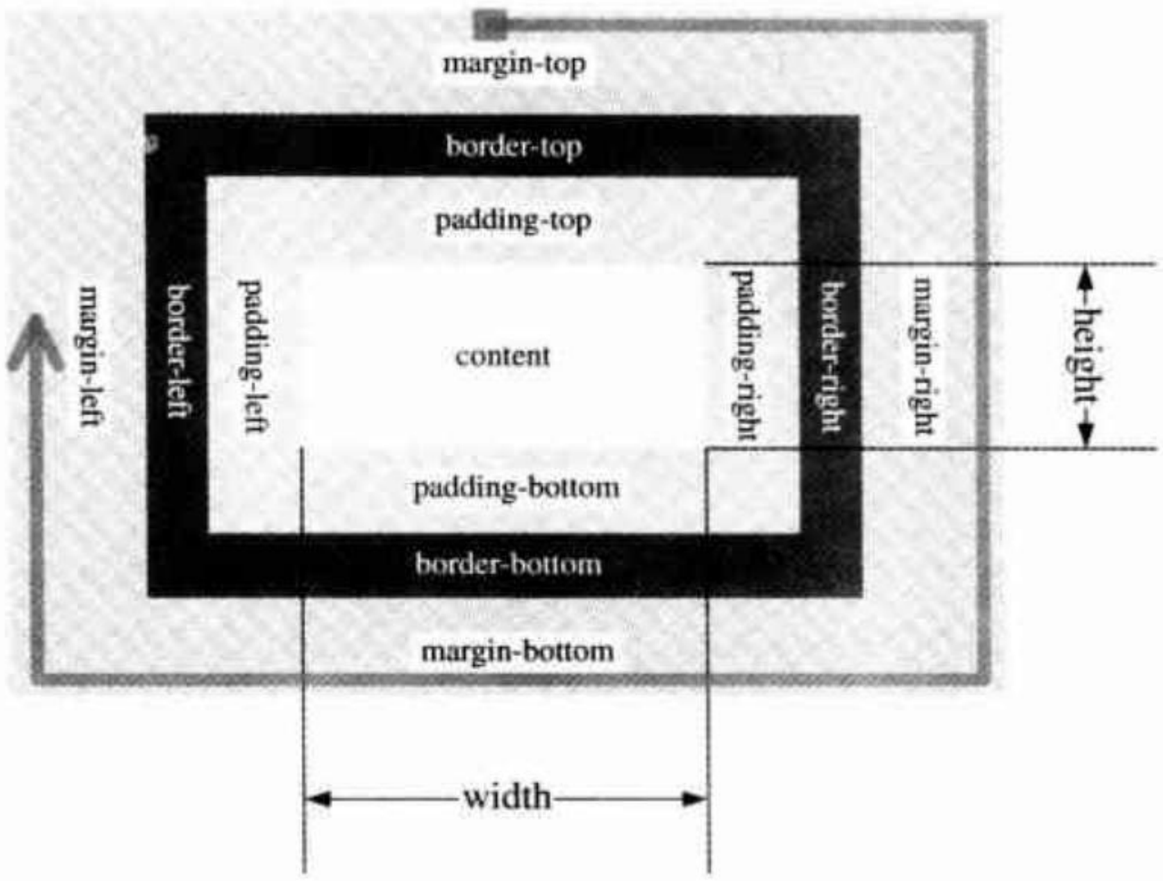


图3.2　盒子模型

可以看到，与前面的图3.1非常相似，盒子的概念是非常容易理解的。但是如果需要精确地排版，有的时候1个像素都不能差，这就需要非常精确地理解其中的计算方法。

一个盒子实际所占有的宽度（或高度）是由“内容+内边距+边框+外边距”组成的。在CSS中可以通过设定width和height的值来控制内容所占的矩形的大小，并且对于任何一个盒子，都可以分别设定4条边各自的border、padding和margin。因此只要利用好这些属性，就能够实现各种各样的排版效果。

注意　并不仅仅是用div定义的网页元素才是一个盒子，事实上所有的网页元素本质上都是以盒子的形式存在的。在人的眼中，一个网页上有各种内容，包括文本、图像等，而在浏览器看来，就是许多盒子排列在一起或者相互嵌套。

图3.2中有一个从上面开始顺时针旋转的箭头，它表示需要读者特别记住的原则，当使用CSS这些部分设置宽度时，是按照顺时针方向确定对应关系的，下一节会详细介绍。

当然还有很多具体的特殊情况，并不能用很简单的规则覆盖全部的计算方法，因此在这一章中，将深入盒子模型的内部，把一般原则和特殊情况都尽可能地阐述清楚。

3.2 边框（border）

border一般用于分隔不同元素，border的外围即为元素的最外围，因此计算元素实际的宽和高时，就要将border纳入。换句话说，border会占据空间，所以在计算精细的版面时，一定要把border的影响考虑进去。如图3.3所示，黑色的虚线框即为border。

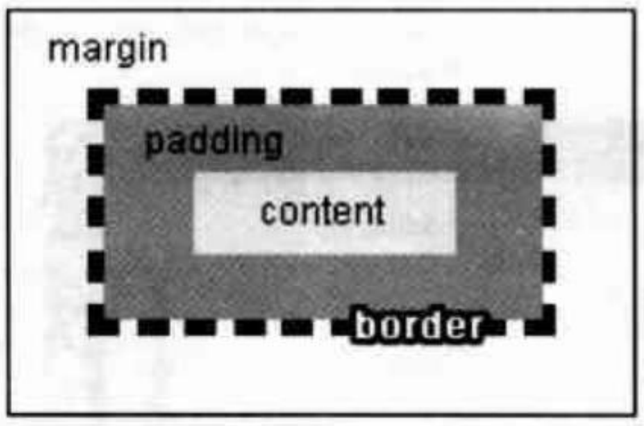


图3.3 border

border的属性主要有3个，分别是color（颜色）、width（粗细）和style（样式）。在设置border时常常需要将这3个属性很好地配合起来，才能达到良好的效果。在使用CSS设置边框的时候，可以分别使用border-color、border-width和border-style设置它们。

● border-color指定border的颜色，它的设置方法与文字的color属性完全一样，一共可以有256³种颜色。通常情况下设置为十六进制的值，例如红色为“#FF0000”。

经验 对于形如“#336699”这样十六进制值，可以缩写为“#369”，当然也可以使用颜色的名称，例如red、green等。

● border-width用来指定border的粗细程度，可以设为thin、medium、thick和<length>。其中<length>表示具体的数值，例如5px和0.1in等。width的默认值为“medium”，一般的浏览器都将其解析为2px宽。

● 这里需要重点讲解的是style属性，它可以设为none、hidden、dotted、dashed、solid、double、groove、ridge、inset和outset等，其中none和hidden都不显示border，二者效果完全相同，只是运用在表格中时，hidden可以用来解决边框冲突的问题。

3.2.1 实验1——border-style

为了了解各种边框样式的具体表现形式，编写如下网页，示例文件位于本书光盘“第3章\01.htm”。

```
<html>
<head>
<title>border-style</title>
<style type="text/css">
div{
  border-width:6px;
  border-color:#000000;
  margin:20px; padding:5px;
  background-color:#FFFFCC;
}
</style>
</head>

<body>
  <div style="border-style:dashed">The border-style of dashed.</div>
```

```
    <div style="border-style:dotted">The border-style of dotted.</div>
    <div style="border-style:double">The border-style of double.</div>
    <div style="border-style:groove">The border-style of groove.</div>
    <div style="border-style:inset">The border-style of inset.</div>
    <div style="border-style:outset">The border-style of outset.</div>
    <div style="border-style:ridge">The border-style of ridge.</div>
    <div style="border-style:solid">The border-style of solid.</div>
</body>
</html>
```

其执行结果在IE和Firefox中略有区别，如图3.4所示。可以看到，对于groove、inset、outset和ridge这几种值，IE都支持得不够理想。

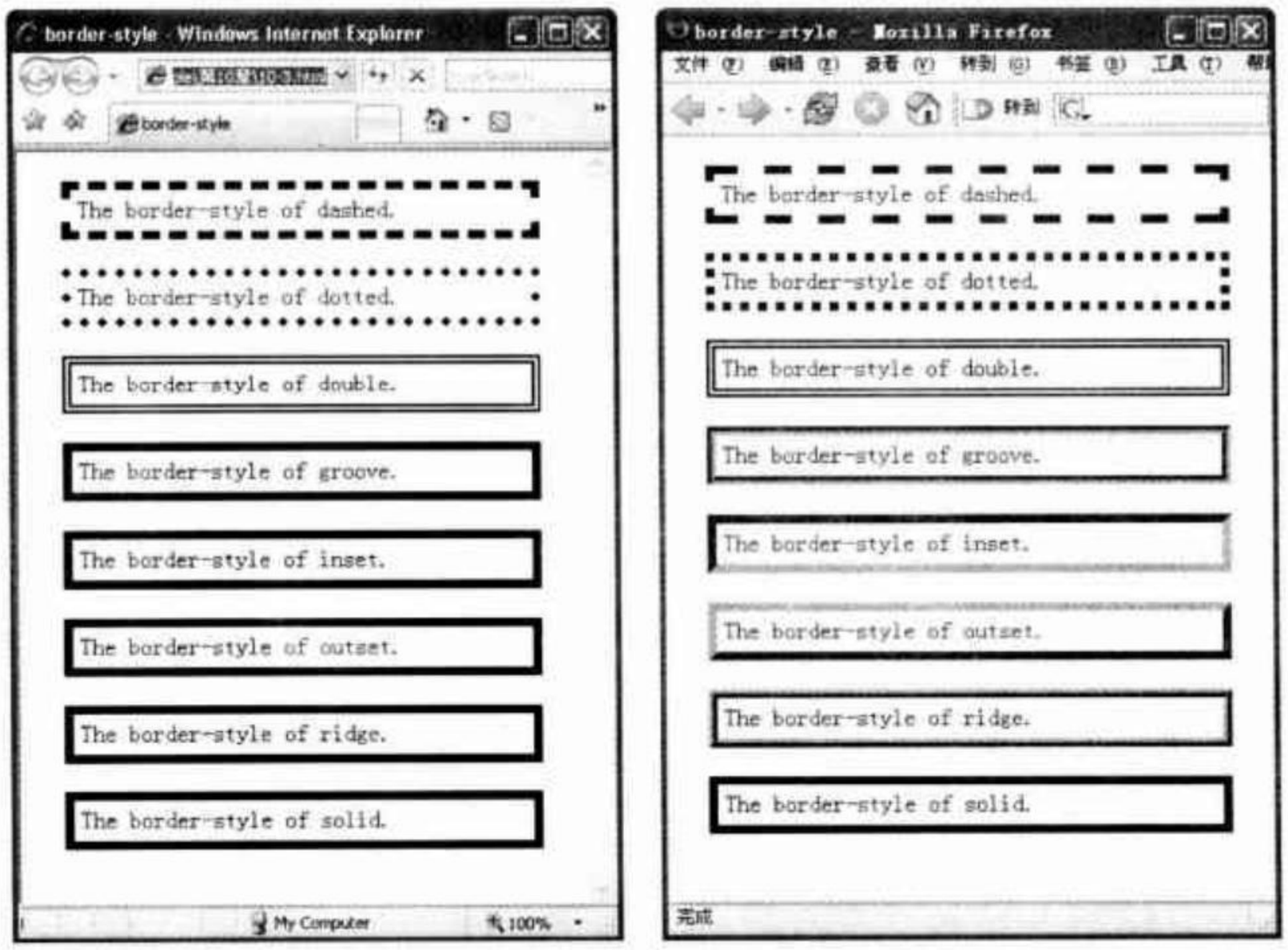


图3.4 border-style

注意 IE浏览器不支持的border-style效果，在实际制作网页的时候，不推荐使用。

3.2.2 属性值的简写形式

1. 对不同的边框设置不同的属性值

3.2.1节的实验代码中，分别设置了border-color、border-width和border-style这3个属性，其效果是对上下左右4个边框同时产生作用。在实际使用CSS时，除了采用这种方式，还可以分别对4条边框设置不同的属性值。

方法是按照规定的顺序，给出2个、3个或者4个属性值，它们的含义将有所区别，具体含义如下：

- 如果给出2个属性值，前者表示上下边框的属性，后者表示左右边框的属性；
- 如果给出3个属性值，前者表示上边框的属性，中间的数值表示左右边框的属性，后者表示下边框的属性；
- 如果给出4个属性值，依次表示上、右、下、左边框的属性，即顺时针排序。

例如，下面这段代码：

```
border-color: red green
border-width:1px 2px 3px;
border-style: dotted、dashed、solid、double;
```

其含义是，上下边框为红色，左右边框为绿色；上边框宽度为1像素，下边框宽度为2像素，左右边框宽度为3像素；从上边框开始，顺时针方向，4个边框的样式分别为点线、虚线、实线和双线。

2. 对一条边框设置与其他边框不同的属性

还可以单独对某一条边框在一条CSS规则中设置属性例如：

```
border: 2px green dashed
border-left: 1px red solid
```

第1行表示将4条边框设置为2像素的绿色虚线，第2行表示将左边框设置为1像素的红色实线。这样，就不需要使用4条CSS规则分别设置4条变框的样式了，仅使用2条规则即可。

3. 对一条边框设置与其他边框不同的属性通过缩写形式可以灵活地设置各种边框。而如果要单独地设置某一条边框的某一个属性，例如要设置左边框的颜色为红色，可以写作：

```
border-left-color:red
```

注意 当有多条规则作用于同一个边框时，会产生冲突，后面的设置会覆盖前面的设置。

3.2.3 实验2——属性的缩写形式

请读者对照属性缩写形式的规则，分析下面这段代码执行后，4条边框最终的宽度、颜色和样式。示例文件位于本书光盘“第3章\01.htm”。

```
<html>
<head>
<style type="text/css">
#outerBox{
  width:200px;
  height:100px;
  border:2px black solid;
```

```
  border-left:4px green dashed;
  border-color:red gray orange blue;  /*上  右  下  左*/
  border-right-color:purple;
}
</style>
</head>
<body>
  <div id="outerBox">
  </div>
</body>
```

在这个例子关于边框的4条CSS规则中，首先把4条边框设置为2像素的黑色实线，然后把左边框设置为4像素绿色虚线，接着又依次设置了边框的颜色，最后把右侧边框的颜色设置为紫色。最终的效果如图3.5所示。

图3.5　设置边框属性

3.2.4 实验3——边框与背景

在设置边框时，还有一点值得注意，在给元素设置background-color背景色时，IE作用的区域为content + padding，而Firefox则是content + padding + border。这在border设置为粗虚线时表现得特别明显，请看如下的实验代码。

这里设置一个div，并将其宽度设置为10像素，使效果非常明显。示例文件位于本书光盘“第3章\03.htm”。

```
<style type="text/css">
#outerBox{
  width:128px;
  height:128px;
  border:10px black dashed;
  background:silver;
}
</style>
</head>
<body>
  <div id="outerBox"></div>
</body>
```

在两种浏览器中的执行结果如图3.6所示，左边是IE中的效果，右边是Firefox中的效果，读者可以通过图中窗口左上角的图标区分浏览器。可看到IE并没有对border的背景上色，而

Firefox的边框中显示出了背景色。

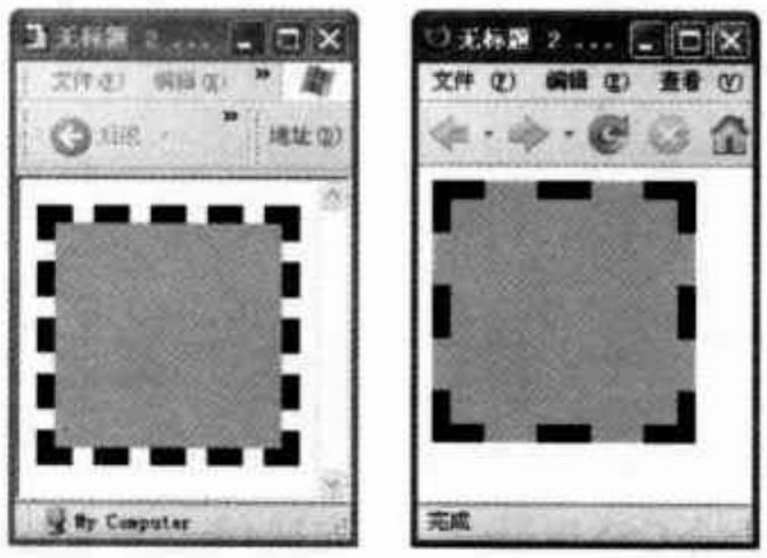


图3.6 IE与Firefox对待背景色的不同处理

虽然这个差别非常细微，但是在设计一些要求很高的页面时，还是需要注意的。

注意 不要因为上面这个例子，就误认为差别是因为IE和Firefox设置背景的基准点不同。实际上它们都是以padding为设置背景的基准点的。要验证这一点，可以把上面例子中的背景设置为一幅图像，这时二者效果如图3.7所示。

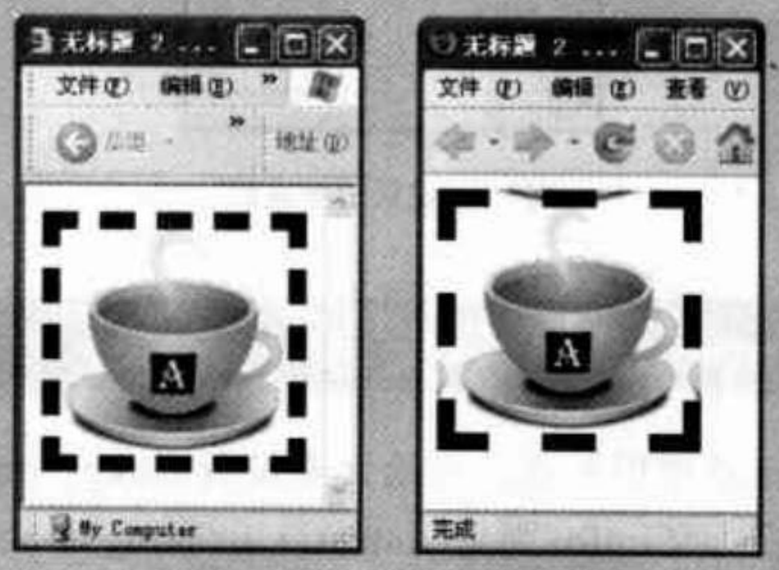


图3.7 IE与Firefox对待背景图像的不同处理

可以看出，二者的背景图像位置是完全相同的，区别只在于边框所占据的面积中，IE并不显示背景图像的内容，Firefox则显示背景图像的内容。因为边框是介于内边距和外边距之间的，当边框设置为虚线时，在IE中，虚线空白的地方露出来的是padding部分的背景，而在Firefox中，虚线空白的地方露出来的是margin部分的背景。

3.3 内边距（padding）

padding，又称为内边距，用于控制内容与边框之间的距离。和前面介绍的边框类似，padding属性可以设置1、2、3或4个属性值，分别如下：

● 设置1个属性值时，表示上下左右4个padding均为该值；

● 设置2个属性值时，前者为上下padding的值，后者为左右padding的值；

● 设置3个属性值时，第1个为上padding的值，第2个为左右padding的值，第3个为下padding的值；

● 设置4个属性值时，按照顺时针方向，依次为上、右、下、左padding的值。

如果需要专门设置某一个方向的padding，可以使用padding-left、padding-right、padding-top或者padding-bottom来设置。例如有如下代码。

```
<style type="text/css">
#outerBox{
  width:128px;
  height:128px;
  padding:20px 20px 10px;     /*上      左右      下*/
  padding-left:10px;
  border:10px gray dashed;
}

<body>
  <div id="outerBox"><img src="image.gif"></img></div>
</body>
```

其结果是上侧和右侧的padding为20像素，下侧和左侧的padding为10像素，如图3.8所示。

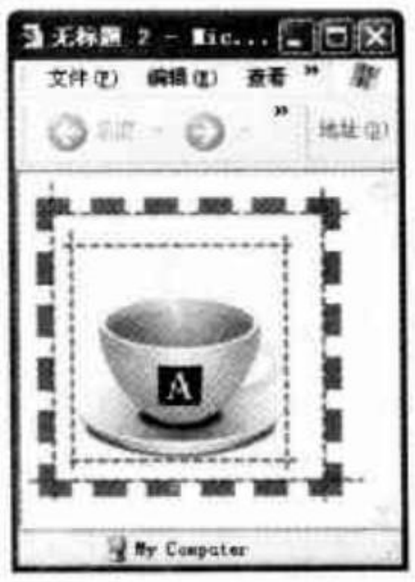


图3.8　设置padding后的效果

经验　当一个盒子设置了背景图像后，默认情况下背景图像覆盖的范围是padding和内容组成的范围，并以padding的左上角为基准点平铺背景图像。

3.4 外边距（margin）

margin指的是元素与元素之间的距离。观察上面的图3.7，可以看到边框在默认情况下

会定位于浏览器窗口的左上角，但是并没有紧贴着浏览器窗口的边框。这是因为body本身也是一个盒子，在默认情况下，body会有一个若干像素的margin，具体数值各个浏览器不尽相同。因此在body中的其他盒子，就不会紧贴着浏览器窗口的边框了。为了验证这一点，可以给body这个盒子也加一个边框，代码如下。

```
body{
  border:1px black solid;
  background:#cc0;
}
```

在body设置了边框和背景色以后，效果如图3.9所示。可以看到，在细黑线外面的部分，就是body的margin。

图3.9 margin的效果

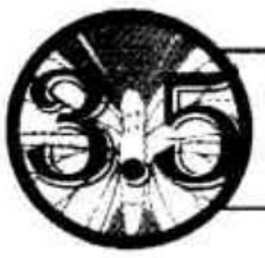
注意 body是一个特殊的盒子，它的背景色会延伸到margin的部分，而其他盒子的背景色只会覆盖“padding+内容”部分（IE浏览器中），或者“border+padding+内容”部分（Firefox浏览器中）。

下面再给div盒子的margin增加20像素，这时效果如图3.10所示。可以看到div的粗边框与body的细边框之间的20像素距离就是margin的范围。右侧的距离很大，这是因为目前body这个盒子的宽度不是由其内部的内容决定的，而是由浏览器窗口决定的，相关的原理本章后面还会深入分析。

图3.10 margin的范围

margin属性值的设置方法与padding一样，也可以设置不同的数值个数，代表相应的含义，这里就不再赘述了。

从直观上而言，margin用于控制块与块之间的距离。倘若将盒子模型比作展览馆里展出的一幅幅画，那么content就是画面本身，padding就是画面与画框之间的留白，border就是画框，而margin就是画与画之间的距离。



3.5 盒子之间的关系

读者要理解前几节的内容并不困难，因为都只涉及一个盒子内部的关系。而实际网页往往是很复杂的，一个网页可能存在着大量的盒子，并且它们以各种关系相互影响着。

要把一个盒子与外部的其他盒子之间的关系理解清楚，并不是简单的事情。在很多CSS资料中大都通过简单的分类，就CSS本身的介绍来说明这个问题，往往只是就事论事。如果不能从站得更高的角度来理解这个问题，那么想真正搞懂它是很困难的，因此这里尝试从更深入的角度来介绍CSS与HTML的关系，希望对读者的理解有所帮助。

为了能够方便地组织各种盒子有序的排列和布局，CSS规范的制定者进行了深入细致的考虑，使得这种方式既有足够的灵活性，以适应各种排版要求，又能使规则尽可能简单，让浏览器的开发者和网页设计师都能够相对容易地实现。

CSS规范的思路是，首先确定一种标准的排版模式，这样可以保证设置的简单化，各种网页元素构成的盒子按照这种标准的方式排列布局。这种方式就是接下来要详细介绍的“标准流”方式。

但是仅通过标准流方式，很多版式是无法实现的，限制了布局的灵活性，因此CSS规范中，又给出了另外若干种对盒子进行布局的手段，包括“浮动”属性和“定位”属性等，这些内容将在下一章中详细介绍。

需要特别提醒读者注意的是，CSS的这些不同的布局方式设计得非常精巧，环环相扣，在后面所有章节中，都是以这些基本的方法和原理为基础的，因此即使是对CSS有一些了解的读者，也应该尽可能仔细地阅读第3章和第4章的内容，把里面的所有实验案例都亲自动手调试一下，这对于深刻理解其中的原理，将会大有益处。

3.5.1 HTML与DOM

这里首先介绍DOM的概念。DOM是Document Object Model的缩写，即“文档对象模型”。一个网页的所有元素组织在一起，就构成了一棵“DOM树”。

假设有一个HTML文档，其中的CSS样式部分省略了，这里只关心它的HTML结构。这个网页的结构非常简单，代码如下，示例文件位于本书光盘“第3章\04.htm”。

```
<!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=gb2312"
/>
    <title>盒子模型的演示</title>
        <style type="text/css">
            ……省略……
        </style>
    </head>

    <body>
      <ul>
        <li>第1个列表的第1个项目内容</li>
        <li class="withborder">第1个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
      </ul>
      <ul>
        <li>第2个列表的第1个项目内容</li>
        <li class="withborder">第2个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
      </ul>
    </body>
  </html>
```

这个HTML在IE和Firefox浏览器中的显示效果是一样的，如图3.11所示。

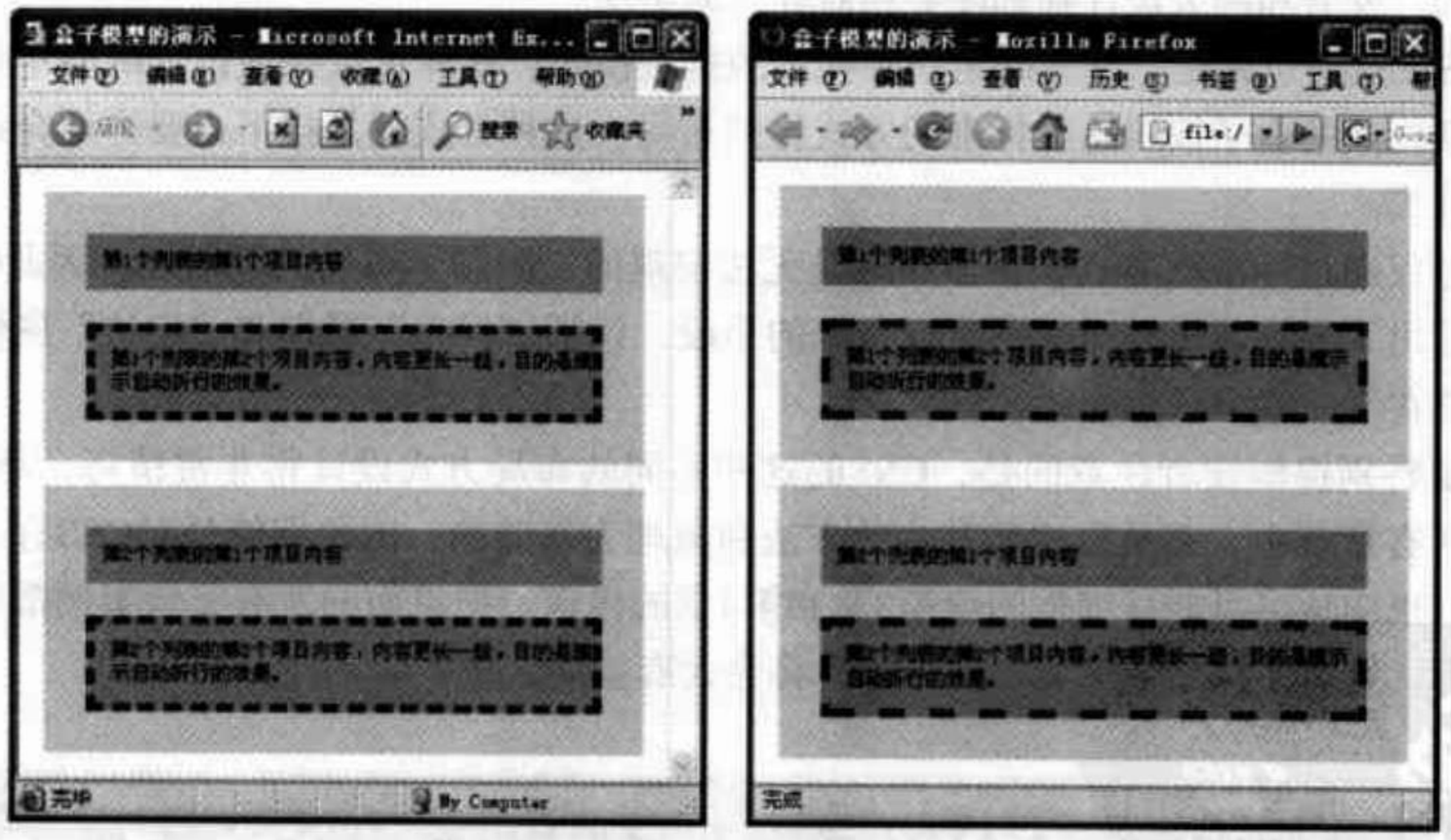


图3.11 在IE与Firefox中的显示效果

为了使读者能够直观地理解什么是“DOM树”，请读者使用Firefox浏览器打开这个网页，然后选择菜单命令“工具→DOM查看器”，这时会打开一个新窗口，如图3.12所示。

图3.12 打开新窗口

窗口左侧列表中的“#document”是整个文档的根节点，双击这个项目，就会打开或关闭它的下级节点，每一个节点都可以打开它的下级节点，直到该节点本身没有下级节点为止。

图3.12中显示的是所有节点都打开的效果。这里使用了一棵“树”的形式把一个HTML文档的内容组织起来，形成了严格的层次结构。例如在本例中，body是浏览器窗口中显示的所有对象的根节点，即ul、li等对象都是body的下级节点。同理，li又是ul的下级节点。在这棵“DOM树”上的各个节点，都对应于网页上的一个区域，例如用鼠标在“DOM查看器”上单击一li节点，立即可以在浏览器窗口中看到一个红色的矩形框，闪烁若干次，如图3.12所示，表示该节点在浏览器窗口中所占的区域，这正是前面所说的CSS“盒子”。

到这里，我们已经和CSS“盒子”联系起来了。读者务必要理解，一个HTML文档并不是一个简单的文本文件，而是一个具有层次结构的逻辑文档，每一个HTML元素（例如p、ul、li等）都作为这个层次结构中的一个节点存在。每个节点反映在浏览器上会具有不同的表现形式，具体的表现形式正是由CSS来决定的。

到这里又印证了一个几乎所有CSS资料中都会提及的一句话：“CSS的目的是使网页的表现形式与内容结构分离，CSS控制网页的表现形式，HTML控制网页的内容结构”，现在读者应该可以更深刻地理解这个原则了。

接下来，就需要理解CSS如何为各种处于层次结构中的元素设置表现形式。

3.5.2 标准文档流

这里又出现了一个新的概念——“标准文档流”，或简称“标准流”。所谓标准流，就是指在不使用其他的与排列和定位相关的特殊CSS规则时，各种元素的排列规则。

仍然以3.5.1节的网页为例，只观察从body开始的这一部分，其内容是body中有两个列表（ul），每个列表中各有两个列表项目（li）。一共有4层结构，顶层为body，第2层为ul，第3层为li，第4层为li中的文本。这4种元素又可以分为以下两类。

1. 块级元素（block level）

li占据着一个矩形的区域，并且和相邻的li依次竖直排列，不会排在同一行中。ul也具有同样的性质，占据着一个矩形的区域，并且和相邻的ul依次竖直排列，不会排在同一行中。因此，这类元素称为“块级元素”（block level），即它们总是以一个块的形式表现出来，并且跟同级的兄弟块依次竖直排列，左右撑满。

2. 行内元素（inline）

对于文字这类元素，各个字母之间横向排列，到最右端自动折行，这就是另一种元素，称为“行内元素”（inline）。

比如标记，就是一个典型的行内元素，这个标记本身不占有独立的区域，仅仅是在其他元素的基础上指出了一定的范围。再比如，最常用的<a>标记，也是一个行内元素。

注意 行内元素在DOM树中同样是一个节点。从DOM的角度来看，块级元素和行内元素是没有区别的，都是树上的一个节点；而从CSS的角度来看，二者有很大的区别，块级元素拥有自己的区域，行内元素则没有。

标准流就是CSS规定的默认的块级元素和行内元素的排列方式。那么具体是如何排列的呢？这里读者不妨把自己想象成一名浏览器的开发者，来考虑一下对一段HTML，应该如何放置这些内容。

```
<body>
  <ul>
    <li>第1个列表的第1个目内容</li>
    <li class="withborder">第1个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
  </ul>
  <ul>
    <li>第2个列表的第1个项目内容</li>
    <li class="withborder">第2个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
  </ul>
</body>
```

（1）第一步

从body标记开始，body元素就是一个最大的块级元素，应该包含所有的子元素，依次把其中的子元素放到适当的位置。例如上面这段代码中，body包含了两个ul，就把这两个块级元素竖直排列。至此第一步完成。

（2）第二步

分别进入每一个ul中，查看它的下级元素，这里是两个li，因此又为它们分别分配了一定的矩形区域。至此第二步完成。

（3）第三步

再进入li内部，这里面是一行文本，因此按照行内元素的方式，排列这些文字。

如果一个HTML更为复杂，层次更多，那么依然是不断地重复这个过程，直至所有的元素都被检查一遍，该分配区域的分配区域，该设置颜色的设置颜色，等等。伴随着扫描的过程，样式也就被赋予到每个元素上了。

在这个过程，一个一个盒子自然地形成一个序列，同级别的兄弟盒子依次排列在父级盒子中，同级父级盒子又依次排列在它们的父级盒子中，就像一条河流有干流和支流一样，这就是被称为“流”的原因。

当然实际的浏览器程序的计算过程要复杂得多，但是大致的过程是这样的，因为我们并不打算自己开发一个浏览器，所以不必掌握所有的细节，但是一定要深入理解这些概念。

3.5.3 标记与标记

为了能够更好地理解“块级元素”和“行内元素”，这里重点介绍在CSS排版的页面中经常使用的 和标记。利用这两个标记，加上CSS对其样式的控制，可以很方便地实现各种效果。本节从二者的基本概念出发，介绍两个标记，并且深入探讨两种元素的区别。

标记早在HTML 3.0时代就已经出现，但那时并不常用，直到CSS的普及，才逐渐发挥出它的优势。标记在HTML 4.0时才被引入，它是专门针对样式表而设计的标记。

（division）简单而言是一个区块容器标记，即 与 之间相当于一个容器，可以容纳段落、标题、表格、图片，乃至章节、摘要和备注等各种HTML元素。可以把 与 中的内容视为一个独立的对象，用于CSS的控制。声明时只需要对 进行相应的控制，其中的各标记元素都会随之改变。

一个ul是一个块级元素，同样div也是一个块级元素，二者的不同在于ul是一个具有特殊含义的块级元素，具有一定的逻辑语义，而div是一个通用的块级元素，用它可以容纳各种元素，从而方便排版。

下面举一个简单的例子，示例文件位于本书光盘“第3章\05.htm”。

```
<html>
<head>
<title>div 标记范例</title>
<style type="text/css">
div{
```

```
    font-size:18px;                          /* 字号大小 */
    font-weight:bold;                        /* 字体粗细 */
    font-family:Arial;                       /* 字体 */
    color:#FFFF00;                           /* 颜色 */
    background-color:#0000FF;                /* 背景颜色 */
    text-align:center;                       /* 对齐方式 */
    width:300px;                             /* 块宽度 */
    height:100px;                            /* 块高度 */
}
</style>
</head>
<body>
  <div>
  这是一个div标记
  </div>
</body>
</html>
```

通过CSS对 块的控制，制作了一个宽300像素、高100像素的黄色区块，并进行了文字效果的相应设置，在IE中的执行结果如图3.13所示。

图3.13 div块示例

标记与 标记一样，作为容器标记而被广泛应用在HTML语言中。在与中间同样可以容纳各种HTML元素，从而形成独立的对象。如果把“ ”替换成“”，样式表中把“div”替换成“span”，执行后就会发现效果完全一样。可以说 与这两个标记起到的作用都是独立出各个区块，在这个意义上说二者没有不同。

与的区别在于， 是一个块级元素，它包围的元素会自动换行。而仅仅是一个行内元素（inline elements），在它的前后不会换行。没有结构上的意义，纯粹是应用样式，当其他行内元素都不合适时，就可以使用元素。

例如有如下代码。

```
<html>
<head>
<title>div与span的区别</title>
</head>
```

```
<body>
  <p>div标记不同行：</p>
  <div><img src="building.jpg" border="0"></div>
  <div><img src="building.jpg" border="0"></div>
  <div><img src="building.jpg" border="0"></div>
  <p>span标记同一行：</p>
  <span><img src="building.jpg" border="0"></span>
  <span><img src="building.jpg" border="0"></span>
  <span><img src="building.jpg" border="0"></span>
</body>
</html>
```

其执行的结果如图3.14所示。 标记的3幅图片被分在了3行中，而标记的图片没有换行。

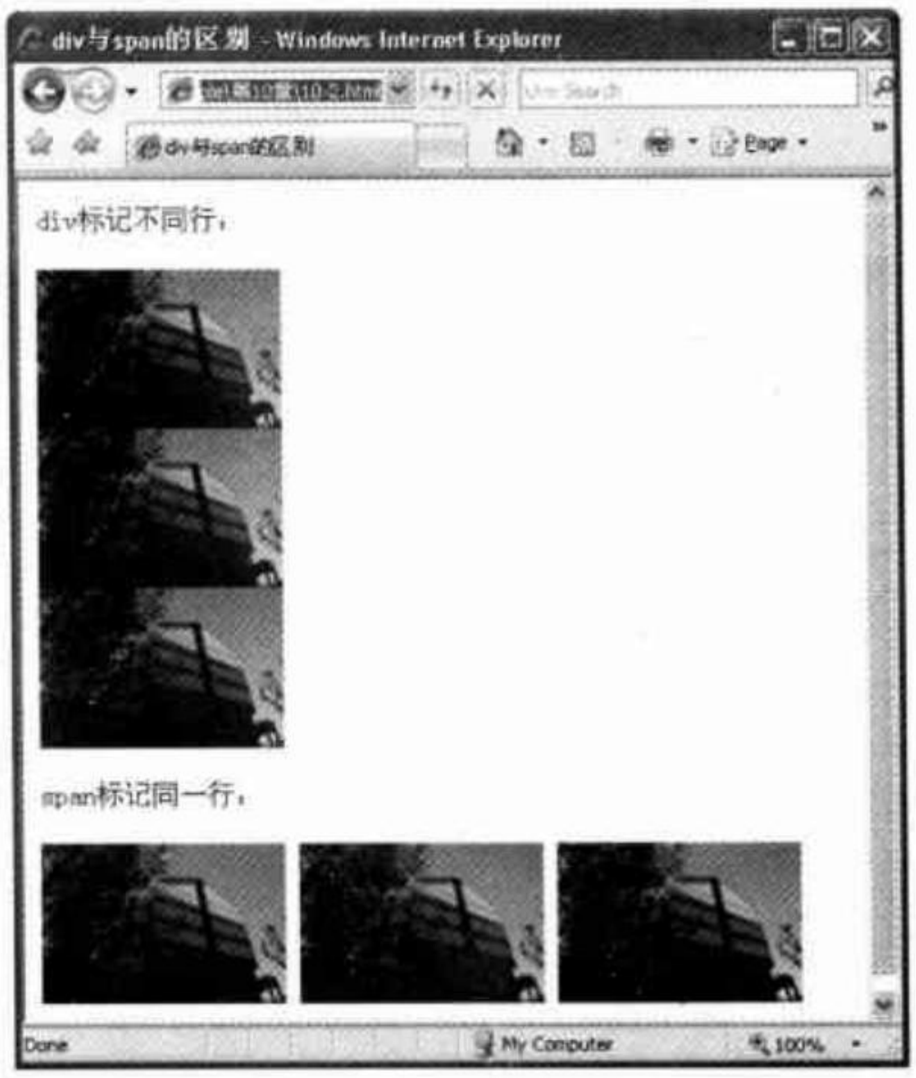


图3.14　与标记的区别

此外，标记可以包含于 标记中，成为它的子元素，而反过来则不成立，即标记不能包含 标记。从div和span之间的区别和联系，就可以更深刻地理解块级元素和行内元素的区别了。

3.6 盒子在标准流中的定位原则

在了解了标准流的基本原理后，来具体制作一些案例，掌握盒子在标准流中的定位原则。

如果要精确地控制盒子的位置，就必须对margin有更深入的了解。padding只存在于一个盒子内部，所以通常它不会涉及与其他盒子之间的关系和相互影响的问题。margin则用于调整不同的盒子之间的位置关系，因此必须要对margin在不同情况下的性质有非常深入的了解。

3.6.1 实验1——行内元素之间的水平margin

这里来看两个块并排的情况，如图3.15所示。

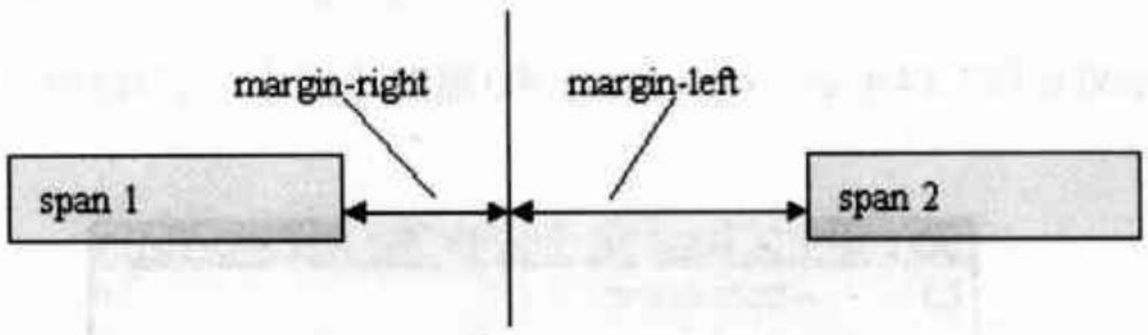


图3.15 行内元素之间的margin

当两个行内元素紧邻时，它们之间的距离为第1个元素的margin-right加上第2个元素的margin-left，代码如下所示，示例文件位于本书光盘“第3章\06.htm”。

```
<html>
<head>
<title>两个行内元素的margin</title>
<style type="text/css">
span{
  background-color:#a2d2ff;
  text-align:center;
  font-family:Arial, Helvetica, sans-serif;
  font-size:12px;
  padding:10px;
}
span.left{
  margin-right:30px;
  background-color:#a9d6ff;
}
span.right{
  margin-left:40px;
  background-color:#eeb0b0;
}
</style>
</head>
<body>
  <span class="left">行内元素1</span><span class"right">行内元素
2</span>
</body>
</html>
```

执行结果如图3.16所示，可以看到两个块之间的距离为30 + 40 = 70px。

图3.16　行内元素之间的margin

3.6.2 实验2——块级元素之间的竖直margin

通过3.6.1节的实验了解了行内元素的情况，但如果不是行内元素，而是竖直排列的块级元素，情况就会有所不同。两个块级元素之间的距离不是margin-bottom与margin-top的总和，而是两者中的较大者，如图3.17所示。这个现象称为margin的“塌陷”现象，意思是说较小的margin塌陷到了较大的margin中。

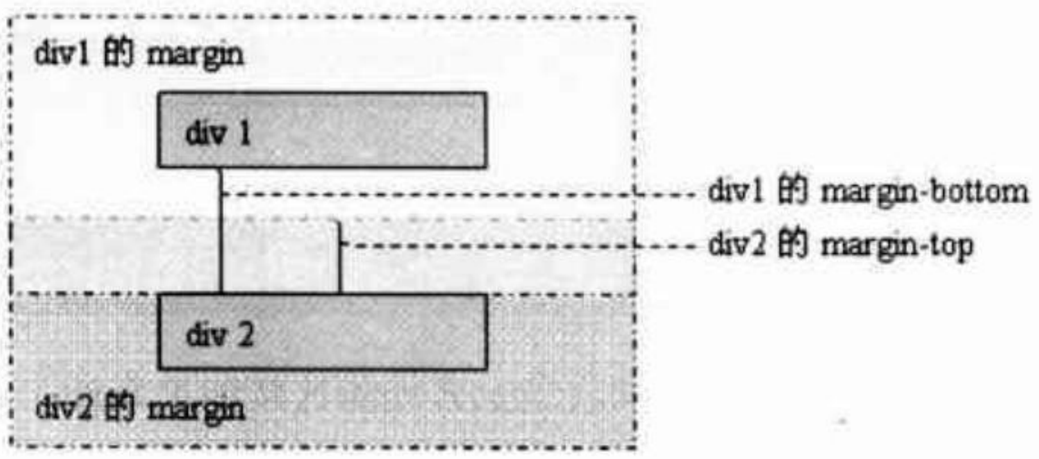


图3.17　块元素之间的margin

这里看一个实验案例，代码如下，示例文件位于本书光盘“第3章\07.htm”。

```
<html>
<head>
<title>两个块级元素的margin</title>
<style type="text/css">
div{
  background-color:#a2d2ff;
  text-align:center;
  font-family:Arial, Helvetica, sans-serif;
  font-size:12px;
  padding:10px;
}
</style>
</head>
<body>
  <div style="margin-bottom:50px;">块元素1</div>
  <div style="margin-top:30px;">块元素2</div>
</body>
</html>
```

执行结果如图3.18所示。倘若将块元素2的margin-top修改为40px，就会发现执行结果没有任何的变化。若再修改其值为60px，就会发现块元素2向下移动了10个像素。

图3.18 块级元素的margin

经验 margin-top和margin-bottom的这些特点在实际制作网页时要特别的注意，否则常常会被增加了margin-top或者margin-bottom值时发现块“没有移动”的假象所迷惑。

3.6.3 实验3——嵌套盒子之间的margin

除了上面提到的行内元素间隔和块级元素间隔这两种关系外，还有一种位置关系，它的margin值对CSS排版也有重要的作用，这就是父子关系。当一个 块包含在另一个 块中时，便形成了典型的父子关系。其中子块的margin将以父块的content为参考，如图3.19所示。

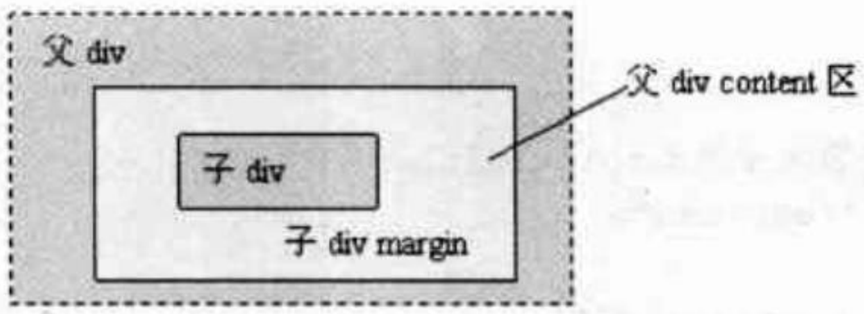


图3.19 父子块的margin

这里有一个实验案例，代码如下，示例文件位于本书光盘“第3章\08.htm”。

```
<head>
<title>父子块的margin</title>
<style type="text/css">
div.father{                                    /* 父div */
  background-color:#fffebb;
  text-align:center;
  font-family:Arial, Helvetica, sans-serif;
  font-size:12px;
  padding:10px;
  border:1px solid #000000;
```

```
}
div.son{                                        /* 子div */
  background-color:#a2d2ff;
  margin-top:30px;
  margin-bottom:0px;
  padding:15px;
  border:1px dashed #004993;
}
</style>
</head>
<body>
  <div class="father">
        <div class="son">子div</div>
  </div>
</body>
```

执行的结果如图3.20所示。可以看到子div距离父div上边为40px（30px margin + 10px padding），其余3条边都是padding的10px。

图3.20　父子块的margin

另外需要注意IE与Firefox在margin的细节处理上有所区别。倘若设定了父元素的高度height值，如果此时子元素的高度超过了该height值，二者的显示结果就完全不同。下面进行实验，编写如下代码，示例文件位于本书光盘“第3章\09.htm”。

```
<head>
<title>设置父块的高度</title>
<style type="text/css">
div.father{                                     /* 父div */
  background-color:#fffebb;
  text-align:center;
  font-family:Arial, Helvetica, sans-serif;
  font-size:12px;
  padding:10px;
  border:1px solid #000000;
  height:40px;                                  /* 设置父div的高度 */
}
div.son{                                        /* 子div */
  background-color:#a2d2ff;
  margin-top:30px; margin-bottom:0px;
```

```
    padding:15px;
    border:1px dashed #004993;
}
</style>
</head>
<body>
    <div class="father">
        <div class="son">子div</div>
    </div>
</body>
```

上面代码中设定的父div的高度值小于子块的高度加上margin的值，此时IE浏览器会自动扩大，保持子元素的margin-bottom的空间以及父元素自身的padding-bottom。而Firefox就不会，它会保证父元素的height高度的完全吻合，而这时子元素将超过父元素的范围，如图3.21所示。

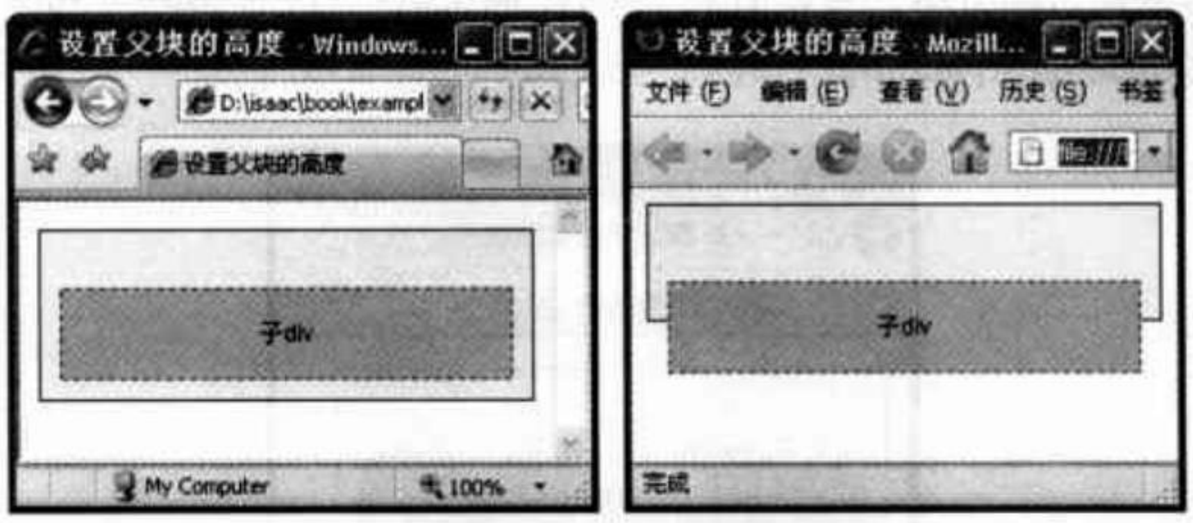


图3.21 IE与Firefox对待父height的不同处理

从CSS的标准规范来说，IE的这种处理方法是不合规范的。它这种方式，本应该由min-height（最小高度）属性承担。

CSS规范中有4个相关属性min-height、max-heght、min-width、max-width，分别用于设置最大、最小宽度和高度，IE没有实现对这4个属性的支持，而Firefox可以非常好地支持它们。本书后面的复杂案例中还会用到相关的内容。

3.6.4 实验4——将margin设置为负值

上面提及margin的时候，它的值都是正数。其实margin的值也可以设置为负数，而且有关的巧妙运用方法也非常多，在后面的章节中都会陆续体现出来。这里先分析margin设为负数时产生的排版效果。

当margin设为负数时，会使被设为负数的块向相反的方向移动，甚至覆盖在另外的块上。在前面例子的基础上，编写代码如下，示例文件位于本书光盘“第3章\10.htm”。

```
<head>
<title>margin设置为负数</title>
<style type="text/css">
span{
  text-align:center;
  font-family:Arial, Helvetica, sans-serif;
```

```
    font-size:12px;
    padding:10px;
    border:1px dashed #000000;
  }
  span.left{
    margin-right:30px;
    background-color:#a9d6ff;
  }
  span.right{
    margin-left:-53px;                    /* 设置为负数 */
    background-color:#eeb0b0;
  }
  </style>
  </head>
  <body>
    <span class="left">行内元素1</span><span class="right">行内元素2</span>
  </body>
```

执行效果如图3.22所示，右边的块移动到了左边的块上方，形成了重叠的位置关系。

图3.22　margin设置为负数

当块之间是父子关系时，通过设置子块的margin参数为负数，可以将子块从父块中“分离”出来，其示意图如图3.23所示。关于它的应用在后面的章节中还会有更详细的介绍。

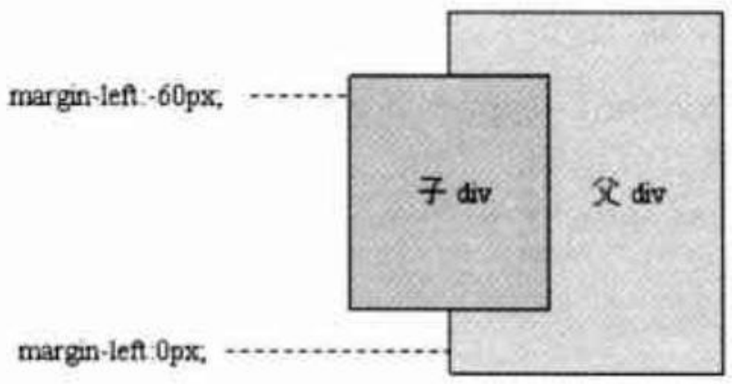


图3.23　父子块设置margin为负数

3.7 CSS中的几何题

经过前面的学习，对标准流中的盒子排列方式应该已经很清楚了。下面来做一个习题，

假设有一个网页，其显示结果如图3.24所示，现在要读者精确地回答出从字母a到x对应的宽度是多少个像素。习题文件位于本书光盘“第3章\04.htm”。

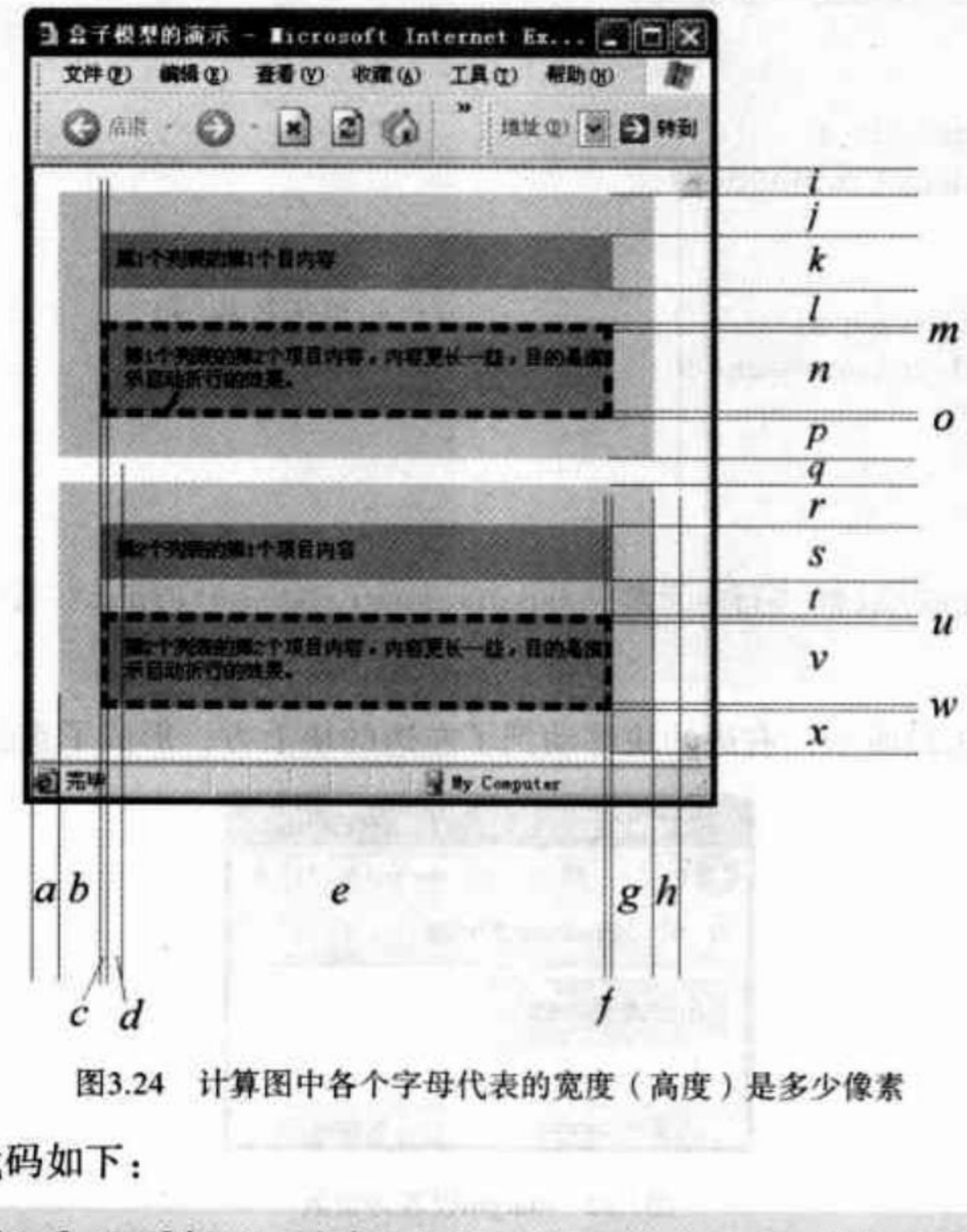


图3.24 计算图中各个字母代表的宽度（高度）是多少像素

网页的完整代码如下：

```
<!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=gb2312" />
<title>盒子模型的演示</title>
    <style type="text/css">
  body{
  margin:0 0 0 0;
  font-family:宋体;
  font-size:12px;

  }
       ul {
         background: #ddd;
         margin: 15px 15px 15px 15px;
         padding: 5px 5px 5px 5px;
                                        /* 没有设置边框 */
       }
       li {
         color: black;                  /* 黑色文本 */
         background: #aaa;              /* 浅灰色背景 */
```

```
        margin: 20px 20px 20px 20px; /* 左侧外边距为0，其余为20像素*/
        padding: 10px 0px 10px 10px; /* 右侧内边距为0，其余10像素 */
        list-style: none             /* 取消项目符号 */
                                     /* 未设置边框 */
      }
      li.withborder {
        border-style: dashed;
        border-width: 5px;           /* 设置边框为2像素 */
        border-color: black;
        margin-top:20px;
      }
    </style>
  </head>
  <body>
    <ul>
      <li>第1个列表的第1个目内容</li>
      <li class="withborder">第1个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
    </ul>
    <ul>
      <li>第2个列表的第1个项目内容</li>
      <li class="withborder">第2个列表的第2个项目内容，内容更长一些，目的是演
示自动折行的效果。</li>
    </ul>
  </body>
</html>
```

下面是具体的计算过程和答案。

先来计算水平方向的宽度，计算过程如下。

① *a*：由于body的magin设置为0，因此a的值为ul的左margin，即15像素。

② *b*：ul的左padding加li的左margin，即25像素。

③ *c*：第2个li的border，即5像素。

④ *d*：li的左padding，即10像素。

⑤ *e*：计算完其他项目后再计算这个宽度，注意这里文字和右边框之间没有间隔，因为右padding为0。

⑥ *f*：第2个li的border，即5像素。

⑦ *g*：ul的右padding加上li的右margin，即25像素。

⑧ *h*：ul的右margin，即15像素。

现在来计算e的宽度。把水平方向除e之外的各项加起来，等于100像素，因此e的宽度为浏览器窗口的宽度减去100像素。

然后计算竖直方向的宽度。

① *i*：由于body的magin设置为0，因此i的值为ul的上margin，即15像素。

② *j*：ul的上padding加上li的上margin，即25像素。

③ *k*：li的上下margin加上文本高度，即34像素。

④ *l*：两个li相邻，因此上面的li的下margin和下面的li的上margin相遇，发生“塌陷”现象，因此l的值为二者中较大者，二者现在相同，因此l的值为20像素。

⑤ *m*：第2个li的border，即5像素。

⑥ *n*：li的上下padding就是上两行文字的高度，即48像素。

⑦ *o*：第2个li的border，即5像素。

⑧ *p*：ul的下padding加上li的下margin，即25像素。

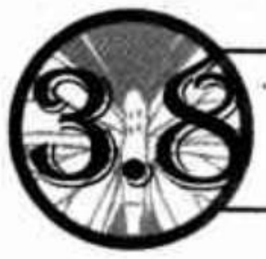
⑨ *q*：两个ul相邻，因此上面的ul的下margin和下面的ul的上margin相遇，发生“塌陷”现象，因此*q*的值为二者中较大者，二者现在相同，因此*q*的值为15像素。

接下来，从*r*开始就前面的相同了，这里不再赘述。

最后，对于盒子的宽度再强调说明一下，上面的这个例子中所有的盒子都没有设置width属性，在没有设置widh属性时，盒子会自动向右伸展，直到不能伸展为止。如果某个盒子设置了width属性，那么盒子的宽度就以该值为准。而盒子实际占据的宽度是width+padding+boarder+margin的总宽度，如图3.25所示。

图3.25 盒子的宽度

注意 在IE 6/7和Firefox中都遵循上述原则，但是低版本的IE对于宽度的计算与此不同，不过现在使用低于IE 6的浏览器的人已经非常少了，一般不用考虑，这里就不做细致讲解了。如果读者希望了解，可以到互联网上搜索一下，相关内容有很多。



3.8 本章小结

盒子模型是CSS控制页面的基础。学习本章之后，读者应该能够清楚在这里“盒子”的含义是什么，以及盒子的组成。

此外，应该理解DOM的基本概念，以及DOM树是如何与一个HTML文档对应的，在此基础上充分理解“标准流”的概念。只有先明白在“标准流”中盒子的布局行为，才能更容易地学习在下一章中将讲解的浮动和定位等相关知识。

第4章
盒子的浮动与定位

理解了第3章介绍的独立的盒子模型，以及在标准流情况下的盒子的相互关系之后，读者也会发现一个重要的问题，如果仅仅按照标准流的方式进行排版，就只能按照仅有的几种可能性进行排版，限制太大了。

在第2章“禅意花园”中的丰富效果是如何实现的呢？CSS的制定者也想到了排版限制的问题，因此又给出了若干不同的手段，实现各种排版需要。

要涉及的最重要的就是CSS中的float和position两个属性。下面就详细介绍它们的应用。

4.1 盒子的浮动

在标准流中，一个块级元素在水平方向会自动伸展，直到包含它的元素的边界；而在竖直方向和兄弟元素依次排列，不能并排。使用“浮动”方式后，块级元素的表现就会有所不同。

CSS中有一个float属性，默认为none，也就是标准流通常的情况。如果将float属性的值设置为left或right，元素就会向其父元素的左侧或右侧靠紧，同时默认情况下，盒子的宽度不再伸展，而是根据盒子里面的内容的宽度来确定。

4.1.1 准备代码

浮动的性质比较复杂，这里先制作一个基础的页面，代码如下，文件位于本书光盘中“第4章\01.htm”。后面一系列的实验将基于这个文件进行。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>float属性</title>
<style type="text/css">
body{
 margin:15px;
 font-family:Arial; font-size:12px;
 }
.father{
 background-color:#ffff99;
 border:1px solid #111111;
 padding:5px;
 }
.father div{
 padding:10px;
 margin:15px;
 border:1px dashed #111111;
 background-color:#90baff;
 }
.father p{
 border:1px dashed #111111;
 background-color:#ff90ba;
 }
.son1{
 /* 这里设置son1的浮动方式*/
 }
```

```
.son2{
  /* 这里设置son1的浮动方式*/
  }
.son3{
      /* 这里设置son3的浮动方式*/
  }
</style>
</head>
<body>
    <div class="father">
          <div class="son1">Box-1</div>
          <div class="son2">Box-2</div>
          <div class="son3">Box-3</div>
          <p>这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字.</p>
    </div>
  </body>
  </html>
```

上面的代码定义了4个 块，其中一个父块，另外3个是它的子块。为了便于观察，将各个块都加上了边框以及背景颜色，并且让<body>标记以及各个div有一定的margin值。

如果3个子div都没有设置任何浮动属性，就为标准流中的盒子状态。在父盒子中，4个盒子各自向右伸展，竖直方向依次排列，效果如图4.1所示。

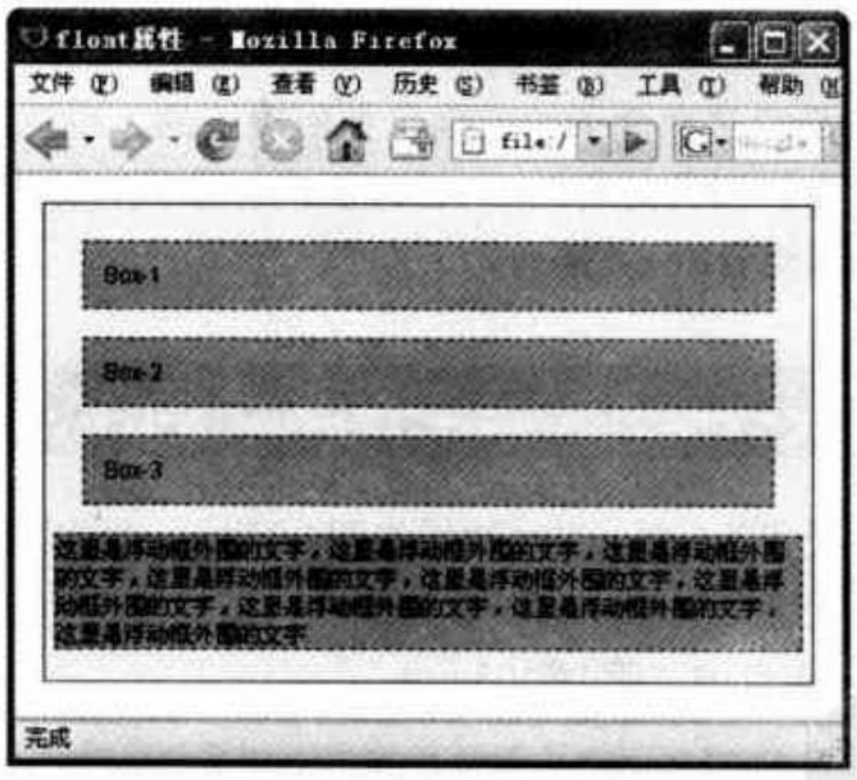


图4.1　没有设置浮动时的效果

下面开始在这个基础上做实验，通过一系列的实验，就可以充分体会到浮动盒子具有哪些性质了。

4.1.2　实验1——设置第1个浮动的div

在上面的代码中找到：

```
.son1{
```

```
    /* 这里设置son1的浮动方式*/
    }
```

将.son1盒子设置为向左浮动，代码为：

```
.son1{
  /* 这里设置son1的浮动方式*/
  float:left;
  }
```

这时效果如图4.2所示，相应的文件位于本书光盘中“第4章\02.htm”。可以看到，标准流中的Box-2的文字在围绕着Box-1排列，而此时Box-1的宽度不再伸展，而是能容纳下内容的最小宽度。

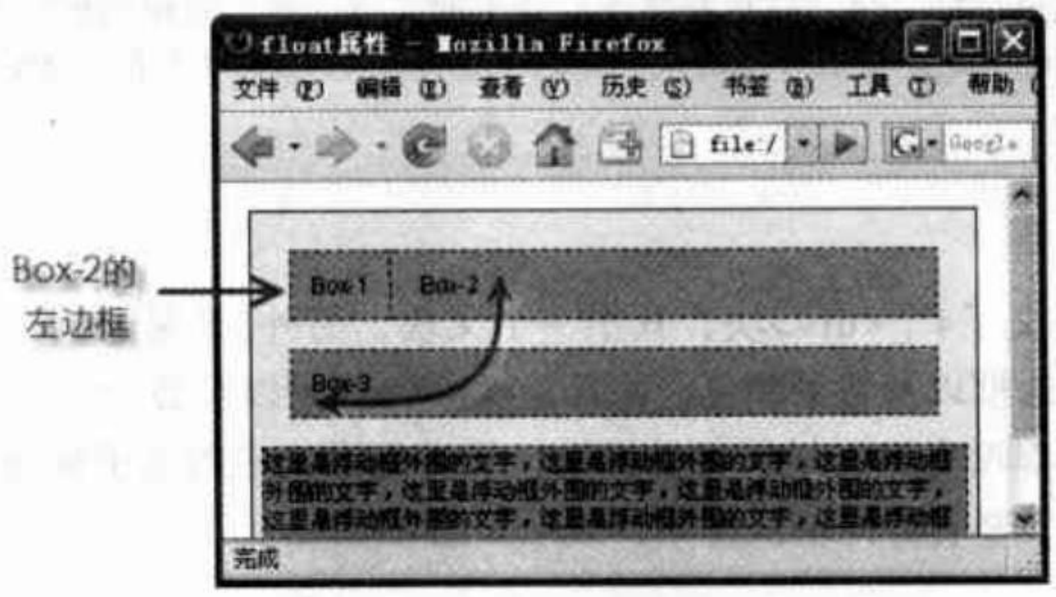


图4.2 设置第1个div浮动时的效果

请读者思考，此时Box-2这个盒子的范围是如何确定的，也就是它的左边框在哪里。答案是与Box-1的左边框重合，因为此时Box-1已经脱离标准流，标准流中的Box-2会顶到原来Box-1的位置，而文字会围绕着Box-1排列。

4.1.3 实验2——设置第2个浮动的div

将Box-2的float属性设置为left，此时效果如图4.3所示。可以看到Box-2也变为根据内容确定宽度，并使Box-3的文字围绕Box-2排列。

相应的文件位于本书光盘中“第4章\03.htm”。

图4.3 设置前两个div浮动时的效果

从图中可以更清晰地看出，Box-3的左边框仍在Box-1的左边框下面。否则Box-1和Box-2之间的空白不会是深色，这个深色实际上是Box-3的背景色，Box-1和Box-2之间的空白是由二者的margin构成的。

4.1.4 实验3——设置第3个浮动的div

接下来，把Box-3也设置为向左浮动。这时效果如图4.4所示，相应的文件位于本书光盘中“第4章\04.htm”。可以清楚地看到，文字所在的盒子的范围，以及文字会围绕浮动的盒子排列。

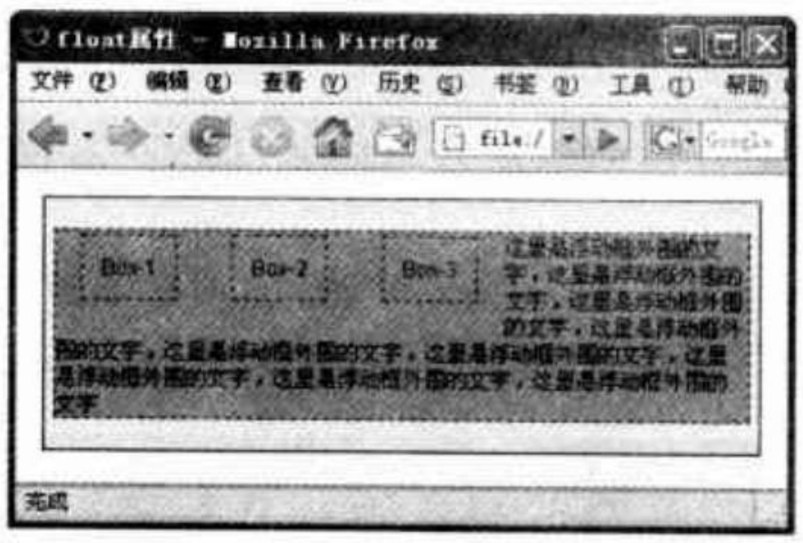


图4.4 设置第3个div浮动时的效果

4.1.5 实验4——改变浮动的方向

将Box-3改为向右浮动，即float:right。这时效果如图4.5所示，相应的文件位于本书光盘中“第4章\05.htm”。可以看到Box-3移动到了最右端，文字段落盒子的范围没有改变，但文字变成了夹在Box-2和Box-3之间。

图4.5 改变浮动方向后的效果

这时，如果把浏览器窗口慢慢调整变窄，Box-2和Box-3之间的距离就会越来越小，直到二者相接触。如果继续把浏览器窗口调整变窄，浏览器窗口就无法在一行中容纳Box-1到Box-3，Box-3会被挤到下一行中，但仍保持向右浮动，这时文字会自动布满空间，如图4.6所示。

图4.6 div被挤到下一行时的效果

4.1.6 实验5——再次改变浮动的方向

将Box-2改为向右浮动，Box-3改为向左浮动。这时效果如图4.7所示，相应的文件位于本书光盘中“第4章\06.htm”。可以看到，布局没有变化，但是Box-2和Box-3交换了位置。

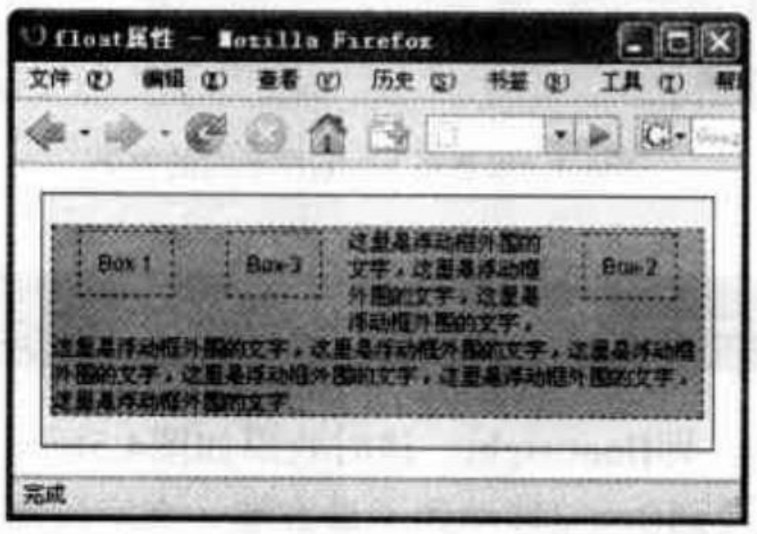


图4.7 交换div位置时的效果

分析： 这里给我们提供了一个很有用的启示，通过使用CSS布局，可以实现在HTML不做任何改动的情况下，调换盒子的显示位置。这个应用非常重要，这样我们就可以在写HTML的时候，通过CSS来确定内容的显示位置；而在HTML中确定内容的逻辑位置，可以把内容最重要的放在前面，相对次要的放在后面。

这样做的好处是，在访问网页的时候，重要的内容先显示出来，虽然这可能只是几秒钟的事情，但是对于一个网站来说，却是很宝贵的几秒钟。研究表明，一个访问者对一个页面的印象往往是由最开始的几秒钟决定的。

此外，搜索引擎是不管CSS的，它只根据网页内容的价值来确定页面的排名，而对于一个HTML文档，越靠前的内容，搜索引擎会赋予越高的权重，因此把页面中最重要的内容放在前面，对于提高网站在搜索引擎的排名是很有意义的。

现在，回到实验中，把浏览器窗口慢慢变窄，当浏览器窗口无法在一行中容纳Box1到Box-3时，和上一个实验一样会有一个Box被挤到下一行。那么被挤到下一行的是哪一个呢？答案是在HTML中，写在后面的，也就是Box-3会被挤到下一行中，但仍保持向左浮动，

会到下一行的左端，这时文字仍然会自动排列，如图4.8所示。

图4.8 div被挤到下一行的效果

4.1.7 实验6——全部向左浮动

下面把页面修改为如图4.9所示的样子，方法是把3个Box都设置为向左浮动，然后在Box-1中增加一行，使它的高度比原来高一些。

图4.9 设置3个div浮动时的效果

请读者思考，如果此时把浏览器窗口调整变窄，结果将会如何？Box-3会被挤到下一行，那么它会在Box-1的下面，还是在Box-2的下面呢？答案如图4.10所示。

图4.10 div挤到下一行被卡住时的效果

在图4.10中绘制了3条示意的虚线，这是Box-2和Box-3的实际分割线。Box-3被挤到下一行，并向左移动，到了这个拐角的地方就会被卡住，而停留在Box-2的下面。

到这里，关于浮动的性质读者应该已经理解了。接下来，很自然地会想到，如何在排版中实现某个盒子浮动，但使它后面的标准流中的盒子不受它的影响。这就需要一个与float属性配合的属性clear，它的作用正是为了消除浮动的盒子对其他盒子的影响。

4.1.8 实验7——使用clear属性清除浮动的影响

参考图4.11所示，修改代码，以使文字的左右两侧同时围绕着浮动的盒子。

这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字，这里是浮动框外围的文字

图4.11　设置浮动后文字环绕的效果

如果不希望文字围绕浮动的盒子，又该怎么办呢？首先找到代码中的如下4行。

```
.father p{
  border:1px dashed #111111;
  background-color:#ff90ba;
  }
```

然后增加一行对clear属性的设置，这里先将它设为左清除，也就是这个段落的左侧不再围绕着浮动框排列，代码如下，相应的文件位于本书光盘中“第4章\07.htm”。

```
.father p{
  border:1px dashed #111111;
  background-color:#ff90ba;
  clear:left;
  }
```

这时效果如图4.12所示，段落的上边界向下移动，直到文字不受左边的两个盒子影响为止，但仍然受Box-3的影响。

接着，将clear属性设置为right，效果如图4.13所示。由于Box-3比较高，因此清除了右边的影响，自然左边就更不会受影响了。

关于clear属性有两点要说明。

- clear属性除了可以设置为了left或right之外，还可以设置为both，表示同时消除左右两边的影响。
- 要特别注意，对clear属性的设置要放到文字所在的盒子里，例如一个p段落的CSS设

置中，而不要放到对浮动盒子的设置里面。经常有初学者没有搞懂原理，误以为在对某个盒子设置了float属性以后，要消除它对外面的文字的影响，就要在它的CSS样式中增加一条clear，其实这是没有用的。

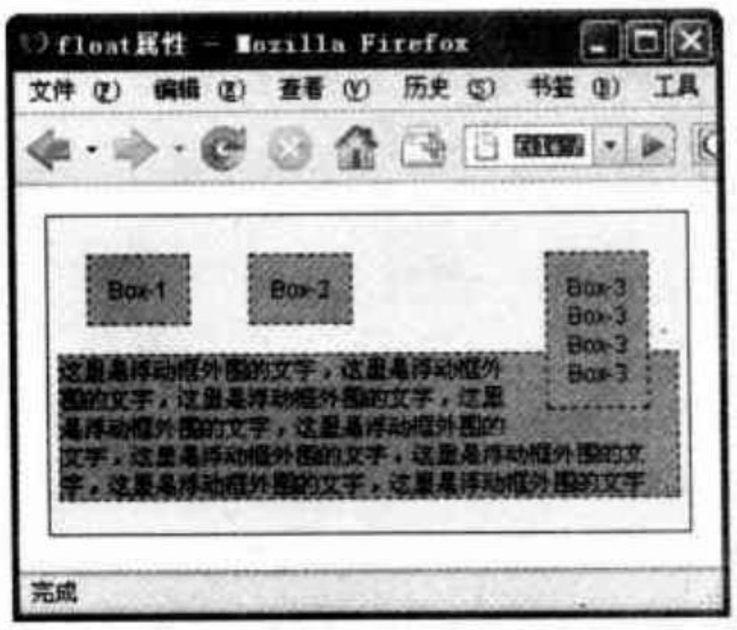


图4.12　清除浮动对左侧影响后的效果

图4.13　清除浮动对右侧影响后的效果

4.1.9　实验8——扩展盒子的高度

关于clear的作用，这里再给出一个例子。在4.1.8节的例子中，将文字所在的段落删除，这时在父div里面只有3个浮动的盒子，它们都不在标准流中，这时观察浏览器中的效果，如图4.14所示。

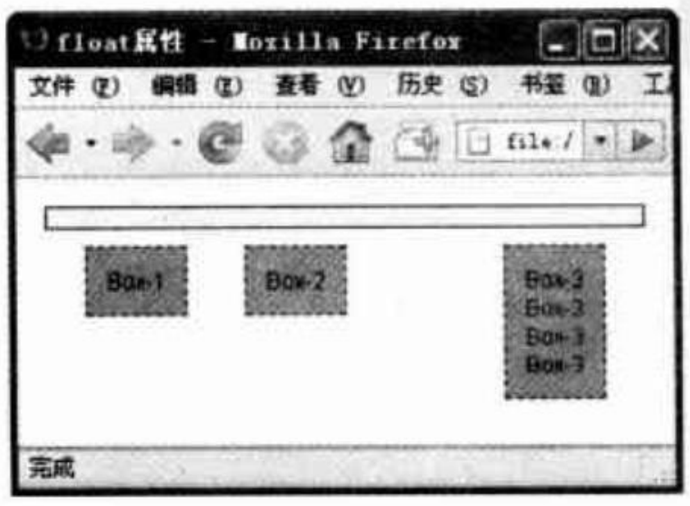


图4.14　包含浮动div的容器将不会适应高度

可以看到，文字段落被删除以后，父div的范围缩成一条，是由padding和border构成的，也就是说，一个div的范围是由它里面的标准流内容决定的，与里面的浮动内容无关。如果要使父div的范围包含这3个浮动盒子，如图4.15所示，那么该怎么办呢？

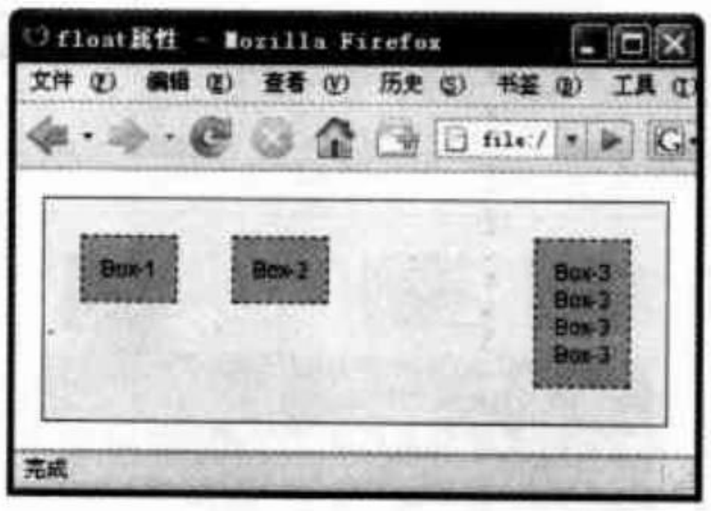


图4.15 希望实现的效果

实现这个效果的方法有几种，但都不完美，都会带来一些不“优雅”的副作用。其中一种方法是在3个div的后面再增加一个div，HTML代码如下：

```
<body>
  <div class="father">
        <div class="son1">Box-1</div>
        <div class="son2">Box-2</div>
        <div class="son3">Box-3<br />
              Box-3<br />
              Box-3<br />
              Box-3</div>
        <div class="clear"></div>
  </div>
</body>
```

然后为这个div设置样式，注意这里必须要指定其父div，并覆盖原来对margin、padding和border的设置。

```
.father .clear{
 margin:0;
 padding:0;
 border:0;
 clear:both;
 }
```

这时效果如图4.15所示，相应的文件位于本书光盘中“第4章\08.htm”。

4.2 盒子的定位

本小节的标题是盒子的定位，实际上对于使用CSS对网页布局这个大主题来说，“定位”

这个词本身有两种含义。

● 广义的“定位”：要将某个元素放到某个位置的时候，这个动作可以称为定位操作，可以使用任何CSS规则来实现，这就是泛指的一个网页排版中的定位操作，使用传统的表格排版时，同样存在定位的问题。

● 狭义的“定位”：在CSS中有一个非常重要的属性position，这个单词翻译为中文也是定位的意思，然而要使用CSS进行定位操作并不仅仅通过这个属性来实现，因此不要把二者混淆。

首先，对position属性的使用方法做一个概述，后面再具体举例子说明。position属性可以设置为4个属性值之一。

● static：这是默认的属性值，也就是该盒子按照标准流（包括浮动方式）进行布局。

● relative：称为相对定位，使用相对定位的盒子的位置常以标准流的排版方式为基础，然后使盒子相对于它在原本的标准位置偏移指定的距离。相对定位的盒子仍在标准流中，它后面的盒子仍以标准流方式对待它。

● absolute：绝对定位，盒子的位置以它的包含框为基准进行偏移。绝对定位的盒子从标准流中脱离。这意味着它们对其后的兄弟盒子的定位没有影响，其他的盒子就好像这个盒子不存在一样。

● fixed：称为固定定位，它和绝对定位类似，只是以浏览器窗口为基准进行定位，也就是当拖动浏览器窗口的滚动条时，依然保持对象位置不变。

读者可能会觉得这4条属性值不太易于理解，这一节的任务就是彻底搞懂它们的含义。

position定位与float一样，也是CSS排版中非常重要的概念。position从字面意思上看就是指定块的位置，即块相对于其父块的位置和相对于它自身应该在的位置。

4.2.1 static（静态定位）

static为默认值，它表示块保持在原本应该在的位置上，即该值没有任何移动的效果。因此，前面的所有例子实际上都是static方式的结构，这里就不再介绍了。

为了讲解清楚后面的其他比较复杂的定位方式，也像上一节一样，使用一系列实验的方法，目标是通过实验的方法找出规律。

这里首先给出最基础的代码，也就是没有设置任何position属性，相当于使用static方式的页面。相应的文件位于本书光盘中“第4章\09.htm”。

```
<!DOCTYPE html PUBliC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>position属性</title>
<style type="text/css">
body{
  margin:20px;
  font :Arial 12px;
}
#father{
  background-color:#a0c8ff;
```

```
    border:1px dashed #000000;
    padding:15px;
}

#block1{
    background-color:#fff0ac;
    border:1px dashed #000000;
    padding:10px;
}
</style>
</head>
<body>
    <div id="father">
        <div id="block1">Box-1</div>
    </div>
</body>
</html>
```

这个页面的效果如图4.16所示，这是一个很简单的标准流方式的两层的盒子。

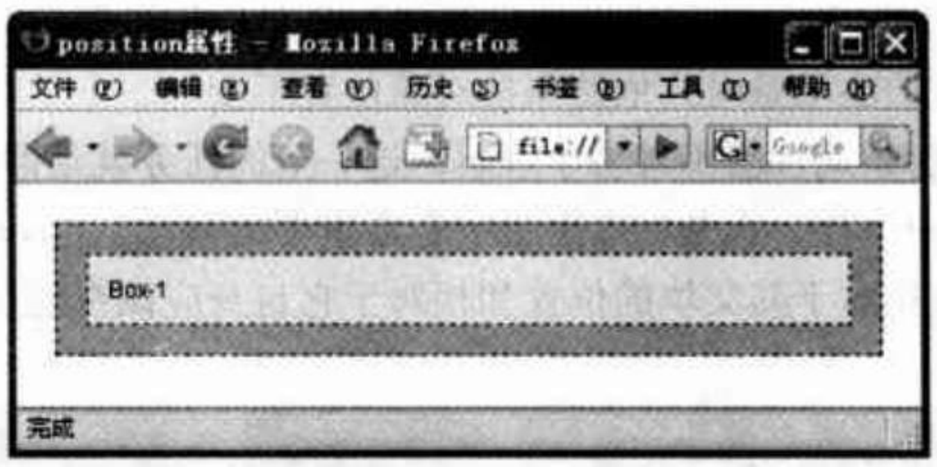


图4.16 没有设置position属性时的状态

4.2.2 relative（相对定位）

使用relative，即相对定位，除了将position属性设置为relative之外，还需要指定一定的偏移量，水平方向通过left或者right属性来指定，竖直方向通过top和bottom来指定。下面还是通过实验的方式找到其中的规律。

1. 实验1——一个子块的情况

下面在CSS样式代码中的Box-1处，将position属性设置为relative，并设置偏移距离，代码如下。

```
#block1{
background-color:#fff0ac;
border:1px dashed #000000;
padding:10px;
position:relative;              /* relative相对定位 */
left:30px;
top:30px;
}
```

效果如图4.17所示，相应的文件位于本书光盘中“第4章\10.htm”。图中显示了Box-1原来的位置和新位置的比较。可以看出，它向右和向下分别移动了30像素，也就是说，“left:30px”的作用就是使Box-1的新位置在它原来位置的左边框右侧30像素的地方，“top:30px”的作用就是使Box-1的新位置在原来位置的上边框下侧30像素的地方。

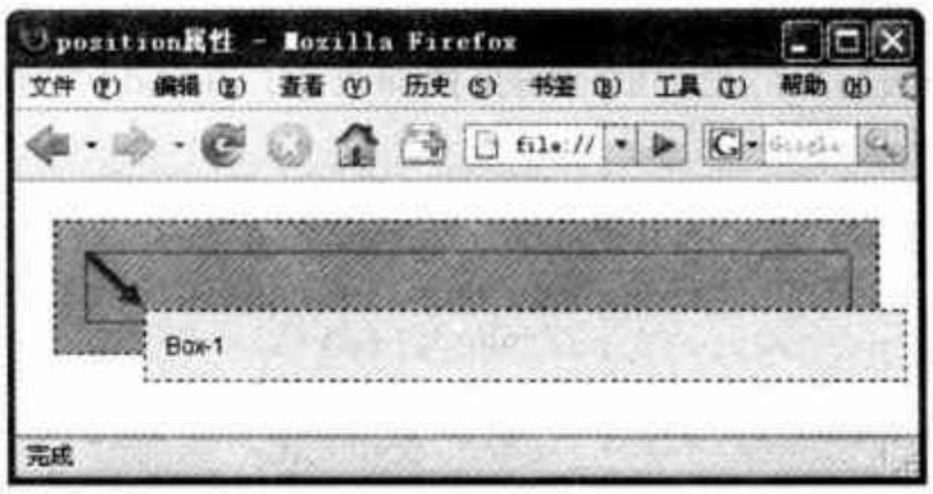


图4.17　一个div设置为相对定定位后的效果

这里用到了top和left这两个CSS属性，实际上在CSS中一共有4个配合position属性使用的定位属性，除top和left之外，还有right和bottom。

这4个属性只有当position属性设置为absolute、relative或fixed时才有效。而且，在position属性取值不同时，它们的含义也不同。当position设置为relative时，它们表示各个边界与原来位置的距离。

top、right、bottom和left这4个属性除了可以设置为绝对的像素数，还可以设置为百分数。此时，可以看到子块的宽度依然是未移动前的宽度，撑满未移动前父块的内容。只是向右移动了，右边框超出了父块。因此，还可以得出另一个结论，当子块使用相对定位以后，它发生了偏移，即使移动到了父盒子的外面，父盒子也不会变大，就好像子盒子没有变化一样。

类似地，如果将偏移的数值设置为：

```
right:30px;
bottom:30px;
```

效果如图4.18所示。

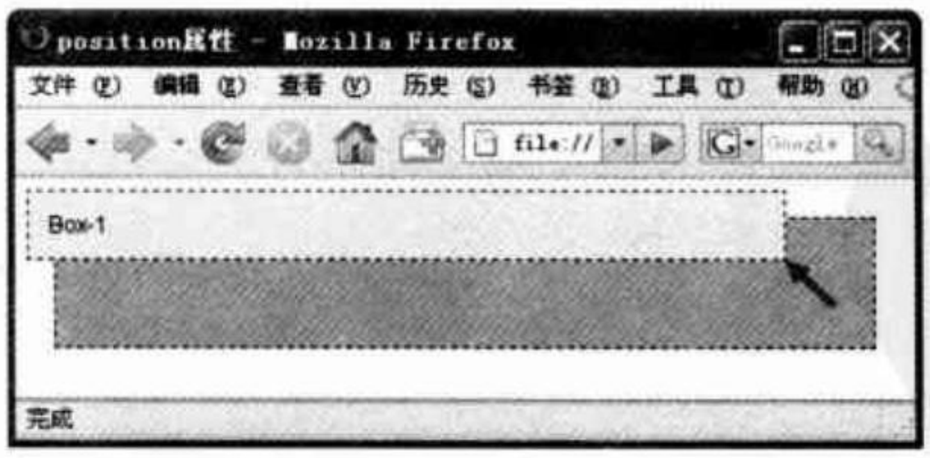


图4.18　以右侧和下侧为基准设置相对定位

对于父块来说，同样没有任何影响，就好像子块没有发生过任何改变一样。因此可以总结出以下两条结论。

● 使用相对定位的盒子，会相对于它原本的位置，通过偏移指定的距离，到达新的位置。

● 使用相对定位的盒子仍在标准流中，它对父块没有任何影响。

2. 实验2——两个子块的情况

下面讨论两个子块的情况。把上面的网页稍加改造，在父div中放两个div。首先对它们都不设置任何偏移，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-Language" content="zh-cn" />
<title>position属性</title>
<style type="text/css">
body{
  margin:20px;
  font-family:Arial;
  font-size:12px;
  }
#father{
  background-color:#a0c8ff;
  border:1px dashed #000000;
  padding:15px;
  }
#father div{
  background-color:#fff0ac;
  border:1px dashed #000000;
  padding:10px;
  }
#block1{
  }
#block2{
  }
</style>
</head>
<body>
  <div id="father">
        <div id="block1">Box-1</div>
        <div id="block2">Box-2</div>
  </div>
</body>
</html>
```

这时效果如图4.19所示，相应的文件位于本书光盘中“第4章\11.htm”。

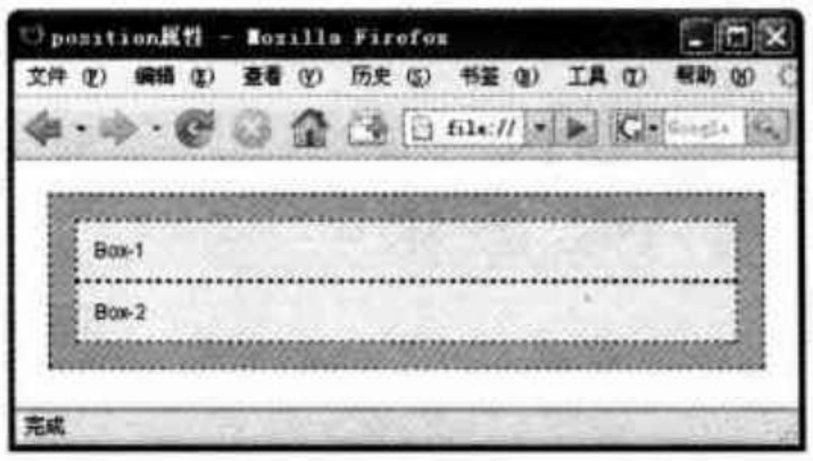


图4.19　设置为相对定位前的效果

在代码中可以看到，现在对两个子块的设置都还空着。下面首先将Box-1盒子的CSS设置为：

```
#block1{
  position:relative;
  bottom:30px;
  right:30px;
  }
```

将子块1的position属性设置成了relative，子块2还没有设置任何与定位相关的属性。此时的效果如图4.20所示，与前面的图4.19对比，可以看到子块1的位置以自身为基准向上和向左各偏移了30像素。而子块2和前面的图4.19所示的相比没有任何变化，就好像子块1还在原来的位置上。

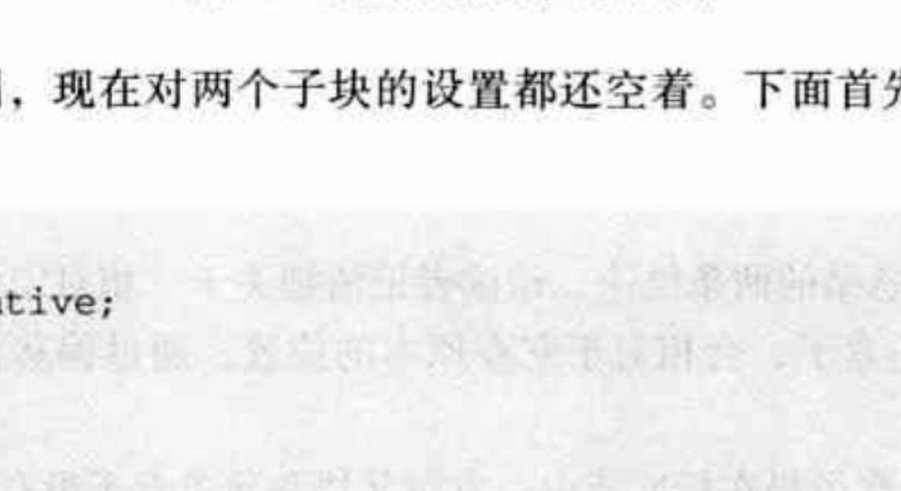


图4.20　两个兄弟div的情况下，其中一个设置为相对定位后的效果

这又一次验证了前面实验1中总结出的两条结论，并且需要把第2条结论再稍稍改进。因为，使用相对定位的盒子不仅对父块没有任何影响，对兄弟盒子也没有任何影响。至此，可以总结出，对于相对定位的规律是：

- 使用相对定位的盒子，会相对于它原本的位置，通过偏移指定的距离，到达新的位置。
- 使用相对定位的盒子仍在标准流中，它对父亲和兄弟盒子都没有任何影响。

如果同时设置两个子块的position属性都为relative，情况又会如何呢？现在把子块2也进行相应的设置，代码如下。

```
#block2{
  position:relative;
  top:30px;
  left:30px;
  }
```

这时的效果如图4.21所示，相应的文件位于本书光盘中“第4章\12.htm”。

图4.21 两个兄弟div都设置为相对定位后的效果

3. 结论

这继续验证了上面总结的两条结论，请读者记清楚关于“相对定位”的定位原则：

● 使用相对定位的盒子，会相对于它在原本的位置，通过偏移指定的距离，到达新的位置。

● 使用相对定位的盒子仍在标准流中，它对父块和兄弟盒子没有任何影响。

需要指出的是，上面的实验是针对标准流方式进行的，实际上，对浮动的盒子使用相对定位也是一样的，例如图4.22中显示的是3个浮动的盒子，它们都向左浮动排在一行中，如果对其中的一个盒子使用相对定位，它也同样相对于它在原本的位置，通过偏移指定的距离，到达新的位置，它旁边的Box-3仍然“以为”它还在原来的位置。

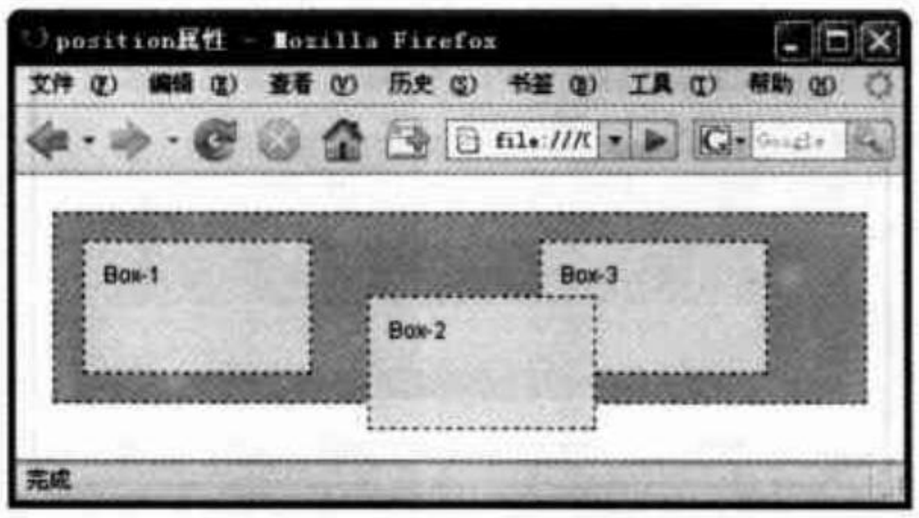


图4.22 在浮动方式下，使用相对定位

4.2.3 absolute（绝对定位）

了解了相对定位以后，下面开始分析absolute定位方式，它表示绝对定位。通过上面的学习，可以了解到各种position属性都需要通过配合偏移一定的距离来实现定位，而其中核心的问题就是以什么作为偏移的基准。

对于相对定位，就是以盒子本身在标准流中或者浮动时原本的位置作为偏移基准的，那么绝对定位以什么作为定位基准呢？

1. 准备网页代码

下面仍然以一个标准流方式的页面为基础，进行一系列的实验，总结出它的规律。先准备如下代码。

```
<!DOCTYPE html PUBliC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>absolute属性</title>
<style type="text/css">
body{
  margin:20px;
  font-family:Arial;
  font-size:12px;
}
#father{
  background-color:#a0c8ff;
  border:1px dashed #000000;
  padding:15px;
}
#father div{
  background-color:#fff0ac;
  border:1px dashed #000000;
  padding:10px;
  }
#block2{
  }
</style>
</head>
<body>
  <div id="father">
        <div >Box-1</div>
        <div id="block2">Box-2</div>
        <div >Box-3</div>
  </div>
</body>
</html>
```

效果如图4.23所示。可以看到，一个父div里面有3个div，都是标准流方式排列。相应的文件位于本书光盘中“第4章\13.htm”。

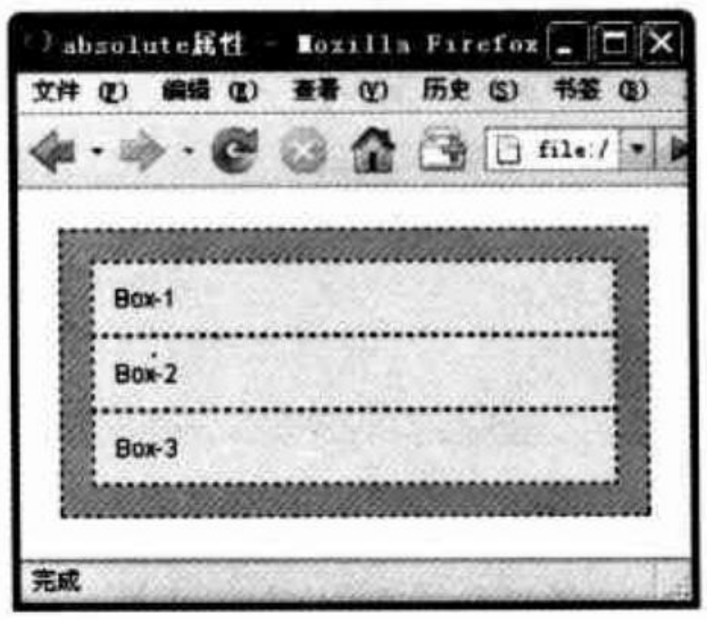


图4.23 设置绝对定位前的效果

2. 实验——使用绝对定位

下面尝试使用绝对定位，代码中找到对#block2的CSS设置位置，目前它是空白的，下面把它改为：

```
#block2{
  position:absolute;
  top:0;
  right:0;
  }
```

这里将Box-2的定位方式从默认的static改为absolute，此时的效果如图4.24所示。这时是以浏览器窗口作为定位基准的。此外，该div会彻底脱离标准流，Box-3会紧贴Box-1，就好像没有Box-2这个div存在一样。相应的文件位于本书光盘中“第4章\13.htm”

图4.24 将中间的div设置为绝对定位后的效果

下面将设置改为：

```
#block2{
  position:absolute;
  top:30px;
  right:30px;
  }
```

这时的效果如图4.25所示，以浏览器窗口为基准，从左上角开始向下和向左各移动30像素，得到图中的效果。

图4.25 设置偏移量后的效果

是不是所有的绝对定位都以浏览器窗口为基准来定位呢？答案是否定的。接下来对上面的代码做一处修改，为父div增加一个定位样式，代码如下。

```
#father{
  background-color:#a0c8ff;
  border:1px dashed #000000;
  padding:15px;
  position:relative;
  }
```

这时效果就变化了，如图4.26所示。偏移的距离没有变化，但是偏移的基准不再是浏览器窗口，而是它的父div了。

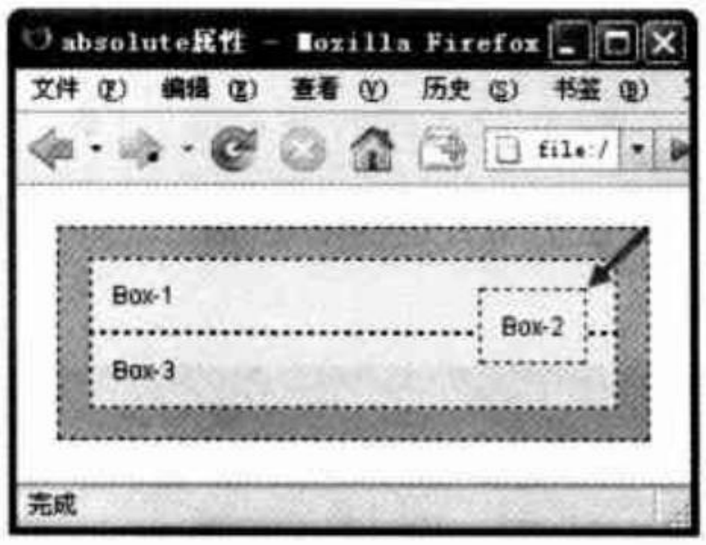


图4.26　将父块设置为“包含块”后的效果

对于绝对定位的正确描述如下。

● 使用绝对定位的盒子以它的“最近”的一个“已经定位”的“祖先元素”为基准进行偏移。如果没有已经定位的祖先元素，那么会以浏览器窗口为基准进行定位。

● 绝对定位的框从标准流中脱离，这意味着它们对其后的兄弟盒子的定位没有影响，其他的盒子就好像这个盒子不存在一样。

在上述第一条原则中，有3个带引号的定语，需要进行一些解释。

（1）所谓“已经定位”元素的含义是，position属性被设置，并且被设置为不是static的任意一种方式，那么该元素就被定义为“已经定位”的元素。

（2）关于“祖先”元素，如果结合本章最前面介绍的“DOM树”的知识，就可以理解了。从任意节点开始，走到根节点，经过的所有节点都是它的祖先，其中直接上级节点是它的父亲，以此类推。

（3）关于“最近”，在一个节点的所有祖先节点中，找出所有“已经定位”的元素，其中距离该节点最近的一个节点，父亲比祖父近，祖父比增祖父近，以此类推，“最近”的就是要找的定位基准。

回到这个实际的例子中，在父div没有设置position属性时，Box-2这个div的所有祖先都不符合“已经定位”的要求，因此它会以浏览器窗口为基准来定位。而当父div将position属性设置为relative以后，它就符合“已经定位”的要求了，它又是所有祖先中惟一一个已经定位的，也就满足“最近”这个要求，因此就会以它为基准进行定位了。本书以后将绝对定位的基准称为“包含块”。

到这里绝对定位已经基本介绍清楚了，最后需要补充说明的是关于IE 6中的一个错误，上面这个例子在IE 6中显示的效果如图4.27所示。

图4.27 在右侧时，IE 6中显示正确

这个效果是正确的，注意现在是用右边框来定位的，如果换成用左边框来定位，效果就会如图4.28所示。

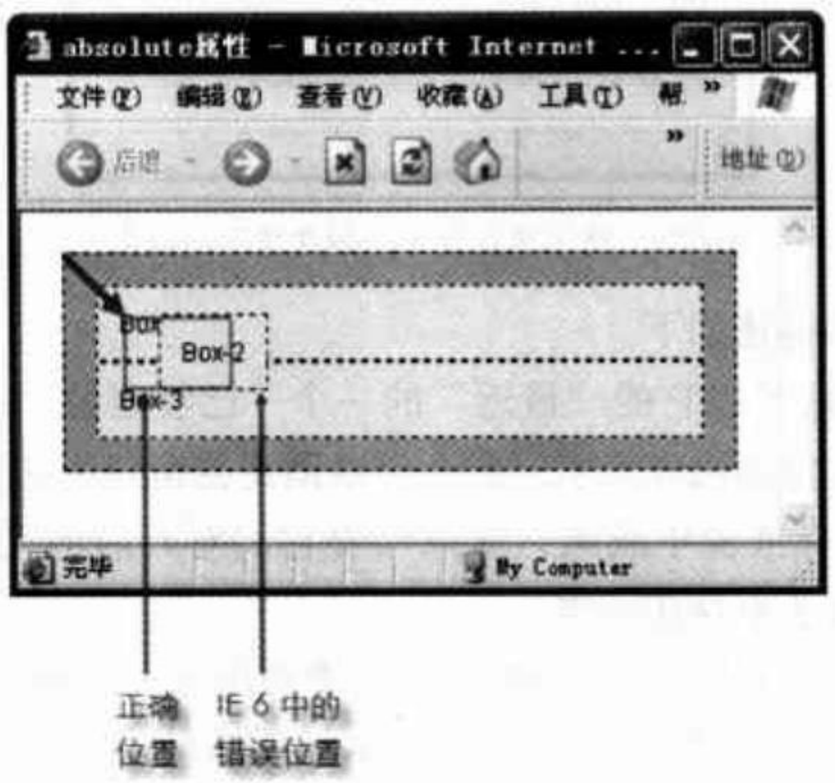


图4.28 在左侧时，IE 6中显示错误

错误的位置和正确的位置相差了父div的padding的宽度，这是IE 6中的固有错误，解决方法是给父div（定位的基准盒子）增加一条CSS样式，代码如下：

```
    height:1%;
```

这时效果如图4.29所示，可以看到位置已经正确了。

图4.29 解决在IE 6中的错误

注意　这种办法通常被称为“CSS hack”，这里的hack不是“黑客”的意思，而是指一些说得清或者说不清道理，但总之很有效的解决办法，就像我们生活中的“偏方”。

3. 浏览器的Bug与Hack

对于这种存在于程序中的小错误，英文中称为“Bug”。任何程序和软件都很难做到清除掉所有Bug，特别是浏览器，加之对于规范的解释不统一，因此类似的错误一直存在。

因此应运而生了许多Hack方法，来解决一些特定的Bug。有的CSS书籍或资料中很喜欢介绍各种“CSS hack”，很多读者也为此花费了大量的精力，其实不需要在这些技巧上投入过多。在实际工作中，这些问题并不常见，而且绝大部分hack技巧都是为了解决IE 5.5及以下版本的错误的，目前除非要制作非常特殊的网站，否则不必考虑IE 5.5的访问者。图4.30所示的是以本书作者的一个网站为例，使用Google提供的网站分析工具对其进行统计，得出的数据，从最近约1万名访问者使用的浏览器状况来看。IE 6/IE 7和Firefox已经占有了绝对统治性的份额。

图4.30左图所示的是访问者中使用各种浏览器的比例，右图所示的则是在使用IE浏览器的访问者中使用不同版本的分布比例。

1.	Internet Explorer	10,146	95.49%
2.	Firefox	429	4.04%
3.	Opera	36	0.34%
4.	Safari	4	0.04%
5.	Mozilla Compatible Agent	3	0.03%
6.	Mozilla	2	0.02%
7.	gzip	2	0.02%
8.	Konqueror	1	0.01%
9.	Netscape	1	0.01%
10.	Openwave	1	0.01%

1.	6.0	8,693	85.68%
2.	7.0	1,411	13.91%
3.	5.01	35	0.34%
4.	5.0	7	0.07%

图4.30　使用各种浏览器的人数比例，以及使用不同版本IE的人数比例

根据这两张图中的数据计算，如果网站能够确保在IE 6/7和Firefox中显示正常，则可以保证99.13%的访问者正常浏览网页。相对来说，满足这3个浏览器要比兼容IE 5.5/5.0等容易得多，基本上不需要太多额外的方法就可以做出兼容性很好的网站。

4. 绝对定位的特殊性质

对于绝对定位，还有一个特殊的性质需要介绍。在有的网页中必须利用这个性质才能实现需要的效果。本书后面章节的案例中，也几次使用到这个性质。

假设有如下的代码：

```
<!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>
```

```
<style>
body{
  margin:0;
}
#outerBox{
  width:200px;
  height:100px;
  margin:10px auto;
  background:silver
  }
#innerBox{
  position:absolute;
  top:70px;
  width:100px;
  height:50px;
  background:orange
  }
</style>
</head>
<body>
    <div id="outerBox">
        <div id="innerBox"></div>
    </div>
</body>
</html>
```

代码中，外面的盒子没有设置position属性，内部的盒子设置了绝对定位，但是只在竖直方向指定了偏移量，没有指定水平方向的偏移量，此时内部的盒子将如何定位呢？

在浏览器中的效果如图4.31所示。可以看到，因为内部的盒子设置了绝对定位属性，而外层的div没有设置position属性，所以它的定位基准是浏览器窗口。但是由于在水平方向上没有设置偏移属性，因此在水平方向它仍然会保持原来应该在的位置，它的左侧与外层盒子的左侧对齐。因为在竖直方向上设置了“top:70px”，所以距离浏览器窗口顶部为70像素。

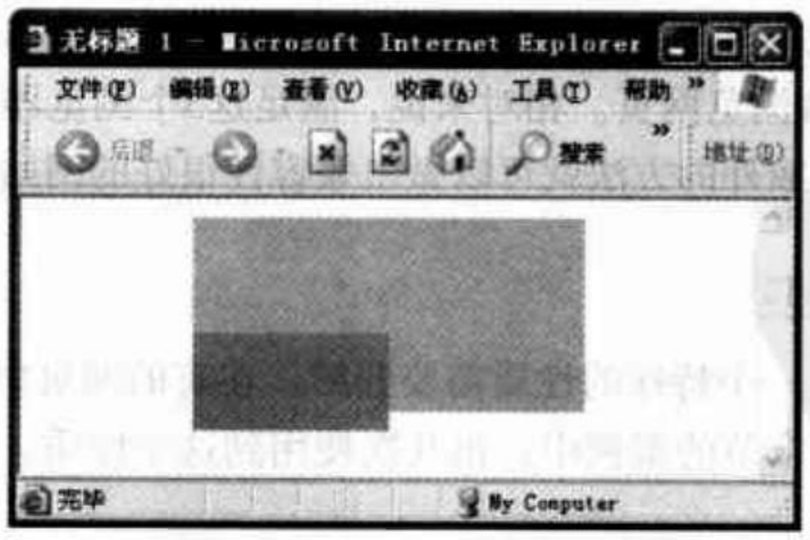


图4.31 使用绝对定位但是不设置偏移属性时的效果

因此，通过这个实验可知，如果设置了绝对定位，而没有设置偏移属性，那么它仍将保持在原来的位置。这个性质可以用于需要使某个元素脱离标准流，而仍然希望它保持在原来的位置的情况。请读者在本书第10章学习到“小视图模式日历”这个案例的时候，再仔细体会其中的用处。

4.2.4 fixed（固定定位）

position属性的第4个取值是fixed，即固定定位。它与绝对定位有些类似，区别在于定位的基准不是祖先元素，而是浏览器窗口或者其他显示设备的窗口。也就是当访问者拖动浏览器窗口滚动条时，固定定位的元素将保持相对于浏览器窗口不变的位置。目前还没有被IE 6浏览器支持，因此这里就不再介绍了。

如果读者有兴趣，可以研究一下本书第2章最后列举的几个很有特色的禅意花园作品，它们就是充分利用固定定位的方式实现的。

4.3 z-index空间位置

z-index属性用于调整定位时重叠块的上下位置，与它的名称一样，想象页面为x-y轴，垂直于页面的方向为z轴，z-index值大的页面位于其值小的上方，如图4.32所示。

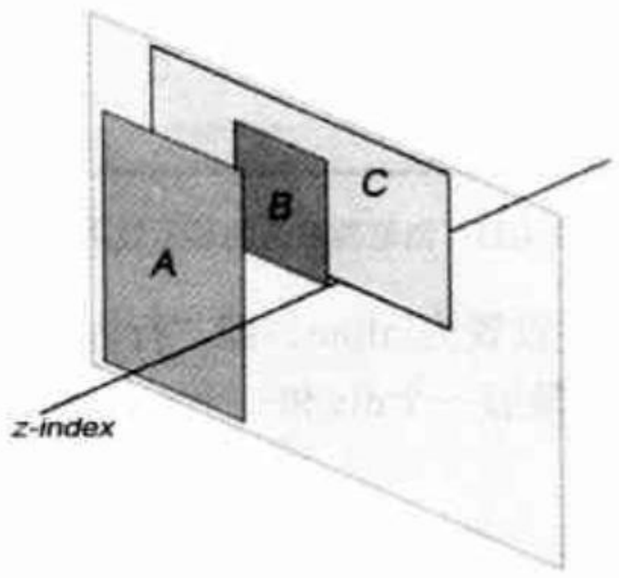


图4.32　z-index轴

z-index属性的值为整数，可以是正数也可以是负数。当块被设置了position属性时，该值便可设置各块之间的重叠高低关系。默认的z-index值为0，当两个块的z-index值一样时，将保持原有的高低覆盖关系。

4.4 盒子的display属性

通过前面的讲解，读者已经知道盒子有两种类型，一种是div这样的块级元素，还有一种是span这样的行内元素。

事实上，对于盒子有一个专门的属性，用以确定盒子的类型，这就是display属性。

假设有如下HTML结构：

```
<body>
  <div>Box-1</div>
  <div>Box-2</div>
  <div>Box-3</div>
  <span>Box-4</span>
  <span>Box-5</span>
  <span>Box-6</span>
  <div>Box-7</div>
  <span>Box-7</span>
</body>
```

这时的效果如图4.33所示，Box-4、Box-5、Box-6是span，因此它们在一行中，其余的都各占一行。

图4.33 浏览器默认的显示效果

下面把前3个div的display属性设置为inline，即“行内”；接着把中间3个span的display属性设置为block，即块级；再把最后一个div和一个span的display属性设置为“none”，即“无”。具体的代码如下：

```
<body>
  <div style="display:inline">Box-1</div>
  <div style="display:inline">Box-2</div>
  <div style="display:inline">Box-3</div>
  <span style="display:block">Box-4</span>
  <span style="display:block">Box-5</span>
  <span style="display:block">Box-6</span>
  <div style="display:none">Box-7</div>
  <span style="display:none">Box-8</span>
</body>
```

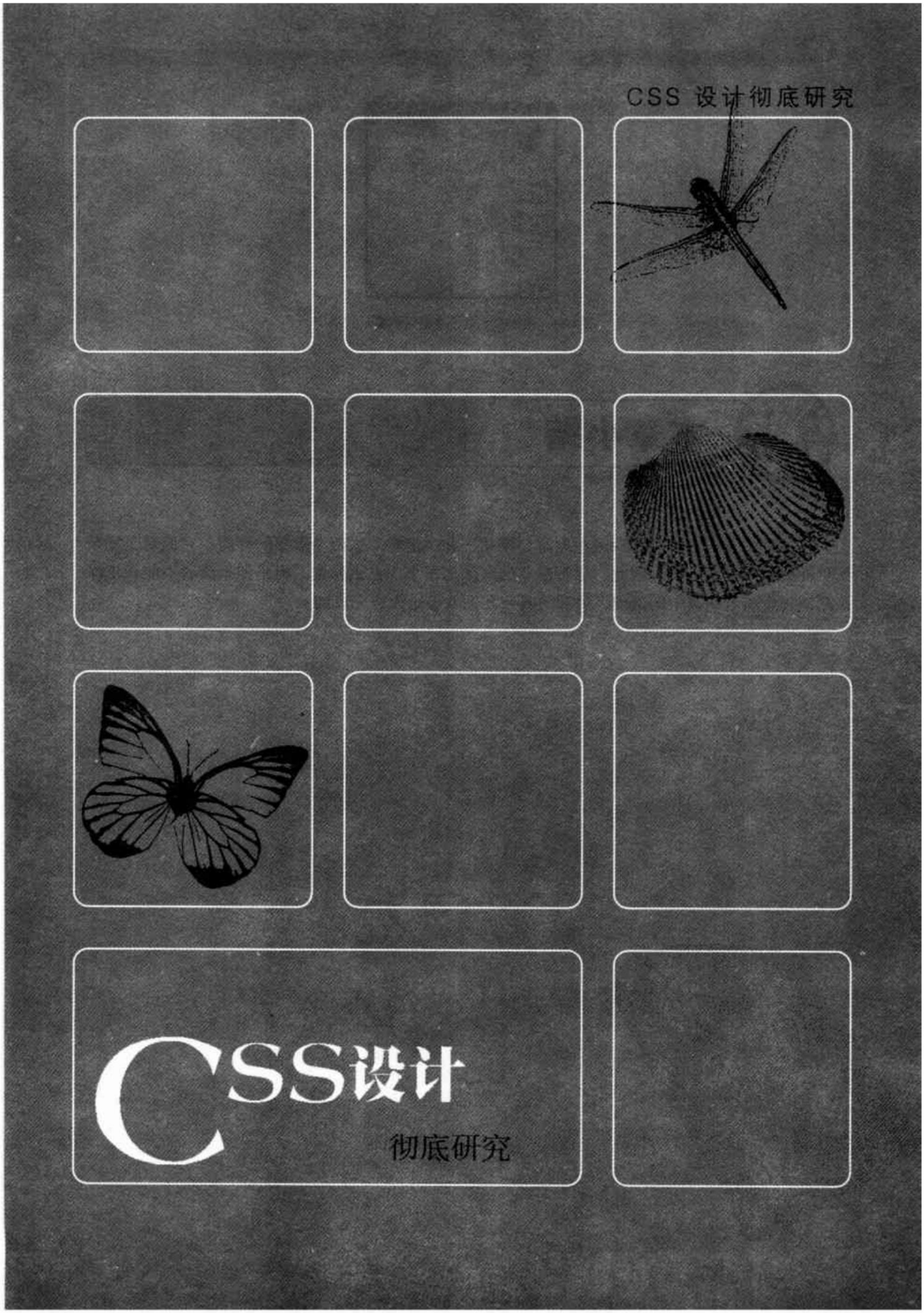
这时效果如图4.34所示。可以看到，原本应该是块级元素的div变成了行内元素，原本应该是行内元素的span变成了块级元素，并且设置为none的两个盒子消失了。

从这个例子可以看出，通过设置display属性，可以改变某个标记本来的元素类型，或者把某个元素隐藏起来。这个性质在后面的案例中，将发挥巨大的作用。

图4.34　强制改变盒子类型后的显示效果

4.5 本章小结

本章的重点和难点是深刻地理解“浮动”和“定位”这两个重要的性质，它们对于复杂页面的排版至关重要。因此，尽管本章的案例都很小，也很朴素，但是如果读者不能真正深刻地理解蕴含在其中的道理，后面的复杂案例效果是无法完成的。



第5章 文字与图像

文字和图像是传达信息的基础，是网页设计永远不可缺少的元素，各种各样的文字和图像效果在整个互联网中无处不在。本章从基础的文字设置开始，详细讲解CSS设置各种文字效果的方法，然后再进一步讲解关于图像的几个重要的应用领域。

5.1 CSS文字样式

使用过任何文字处理软件的用户对文字排版都不会陌生。例如在Word软件中可以对文字的字体、大小和颜色等各种属性进行设置。CSS同样可以对HTML页面中的文字进行全面的设置。

5.1.1 准备网页

为了便于讲解和实践，首先准备一个非常简单的页面，HTML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>Head Line</title>
<style type="text/css">
  /*这里填写CSS样式代码*/
</style>
</head>
<body>
    <h1>Head Line</h1>
      <p class="p1">The Internet Society mission is to achieve a world in
which everyone everywhere is connected to an open and universally
accessible Internet. </p>
      <p class="p2">The Internet Society' s mission is to achieve a world in
which everyone everywhere is connected to an open and universally
accessible Internet.
    ……这里省略其余文字……
     </p>
    </body>
    </html>
```

这个非常简单的网页，由一个h1标题和两个p段落构成。为了对两段文本段落分别进行设置，给它们各自设置了一个类别选择器，p1和p2。在没有设置任何样式时，效果如图5.1所示。该文件请参考本书光盘中的“第5章/文字/basic.htm”。

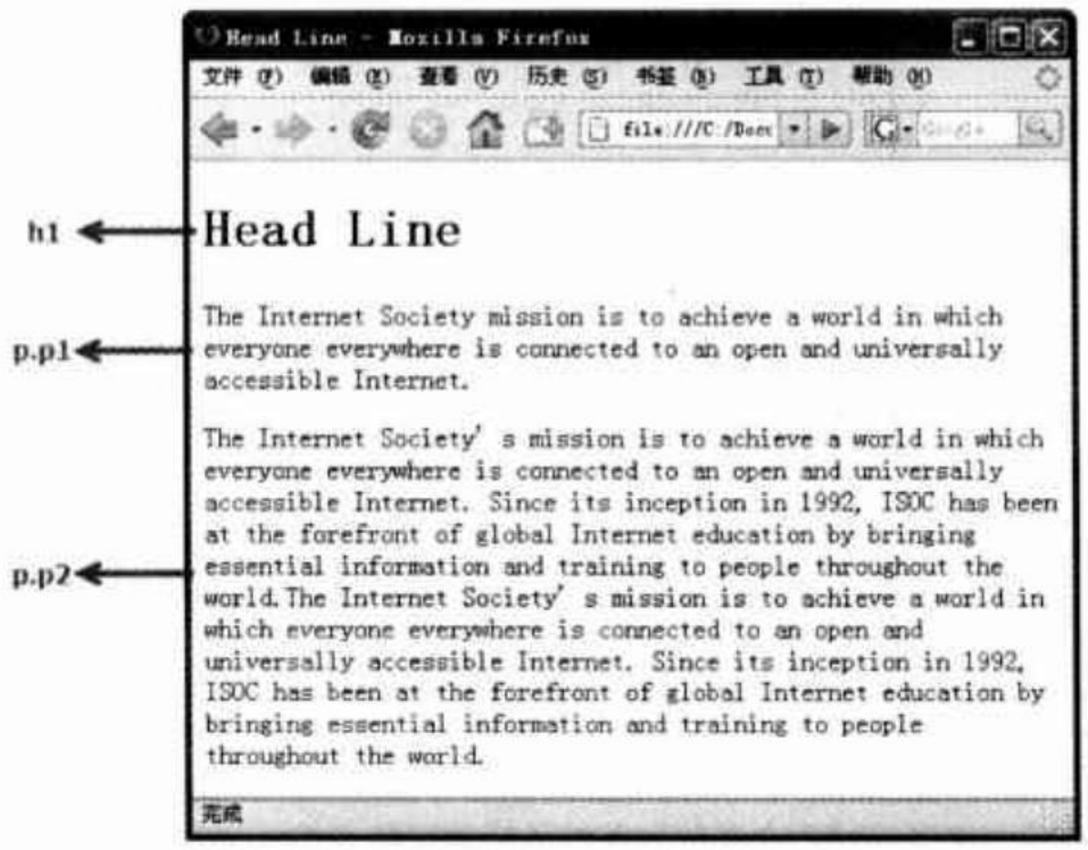


图5.1 准备好的基本页面效果

5.1.2 设置字体

在HTML语言中，文字的字体是通过<font face="字体名称">来设置的，而在CSS中字体是通过font-family属性来控制的。例如针对上面准备好的网页，在样式部分增加如下对p标记的样式设置。

```
<style type="text/css">
p{
  font-family: Arial, "Times New Roman";
  }
</style>
```

以上语句声明了HTML页面中p标记的字体名称，并且同时声明了两个字体名称，分别是Arial字体和Times New Roman字体，其含义是告诉浏览器首先在访问者的计算机中寻找Arial字体。如果该访问者的计算机中没有Arial字体，就寻找Times New Roman；如果这两种字体都没有，则使用浏览器的默认字体显示。

font-family属性可以同时声明多种字体，字体之间用逗号分隔开。另外，一些字体的名称中间会出现空格，例如上面的Times New Roman，这时需要用双引号将其引起来，使浏览器知道这是一种字体的名称。

注意 不要输入中文（全角）的双引号，而要使用英文（半角）的双引号。

这时在浏览器中的效果如图5.2所示。可以看到，两个正文段落中的字体都发生了变化。该文件请参考本书光盘中的“第5章/文字/font-family.htm”。

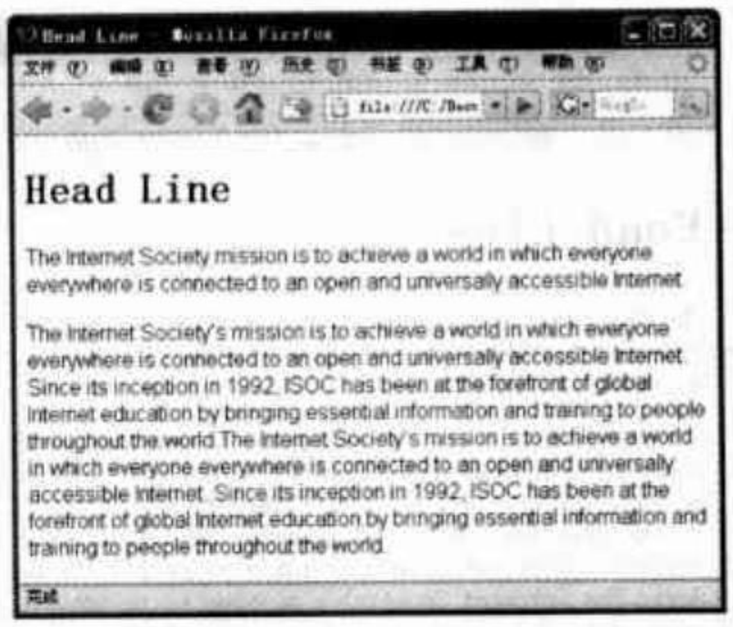


图5.2 设置正文字体

注意 很多设计者喜欢使用各种各样的字体来给页面添彩，但这些字体在大多数用户的机器上都没有安装，因此一定要设置多个备选字体，避免浏览器直接替换成默认的字体。最直接的解决方法是将使用了生僻字体的部分，用图形软件制作成小的图片，再加载到页面中。

5.1.3 文字大小

在网页中通过文字的大小来突出主题是很常用的方法，CSS是通过font-size属性来控制文字大小的，该属性的值可以使用很多种长度单位，这里分别进行介绍。

1. 长度单位px

仍以上面的网页为例子，增加对font-size属性的设置，将其设置为12像素，代码如下。

```
p{
  font-family: Arial, "Times New Roman";
  font-size:12px;
  }
```

这时在浏览器中的效果如图5.3所示。可以看到，此时两个正文段落中的文字都变小了。该文件请参考本书光盘中的“第5章/文字/font-size.htm”。

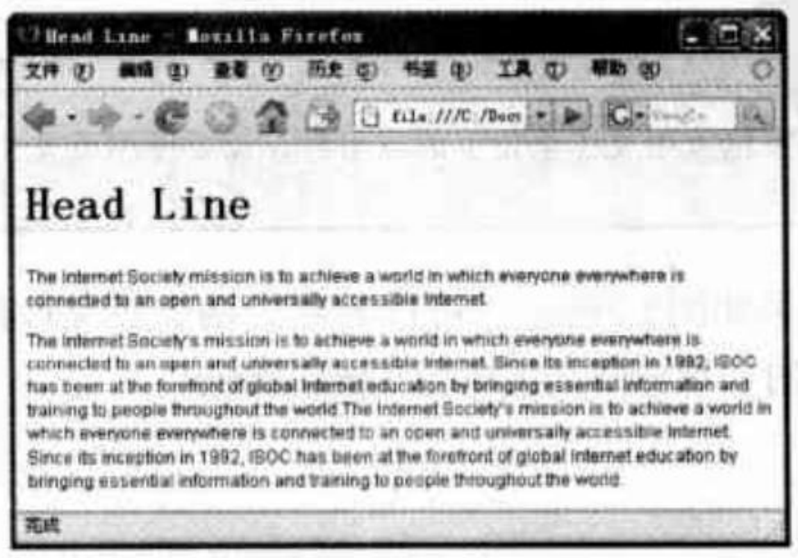


图5.3 设置了正文文字的大小为12像素

代码中的px是一个长度单位，表示在浏览器上1个像素的大小。因为不同访问者的显示器的分辨率不同，而且每个像素的实际大小也不相同，所以px被称为相对单位，也就是相对于1个像素的比例。

在CSS中，除了可以使用px作为长度单位，还可以使用以下5种单位设置大小（包括文字、div的高度和宽度等），这5种单位都被称为绝对长度单位，它们不会随显示器的变化而变化。各个单位的含义如表5.1所示。

表5.1 绝对单位及其含义

长度单位	说明
in	inch，英寸
cm	centimeter，厘米
mm	millimeter，毫米
pt	point，印刷的点数，在一般的显示器中1pt相当于1/72inch
pc	pica，1pc=12pt

2. 长度单位em和ex

此外还有两个比较特殊的长度单位：em和ex。它们与px类似，也是相对长度单位。1em表示的长度是其父元素中字母m的标准宽度，1ex则表示字母x的标准高度。当父元素的文字大小变化时，使用这两个单位的子元素的大小会同比例变化。

例如，在文字排版时，有时会要求第一个字母比其他字母大很多，并下沉显示，就可以使用这个单位。方法是先在上面的HTML中，把第2段文字的第1个字母“T”放入一对标记中，并对它设置一个CSS类别，“.firstLetter”。

```
<p class="p2"><span class=".firstLetter">T</span>he……
```

然后设置它的样式，将font-size设置为2em，并使它向左浮动，代码如下：

```
.firstLetter{
  font-size:3em;
  float:left;
  }
```

这时在浏览器中的效果如图5.4所示。此时第2段的首字母就变为标准大小的3倍，并因设置了向左浮动而实现下沉显示。该文件请参考本书光盘中的“第5章/文字/first-letter.htm”。

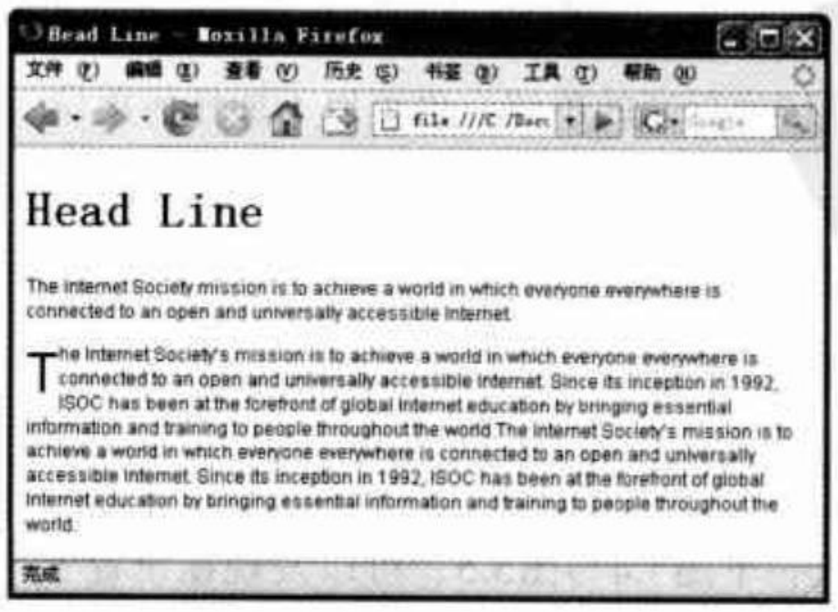


图5.4 设置段首的字母放大并下沉显示

3. 长度单位%

最后一种单位是使用百分比作为单位，例如“font-size:200%”，表示文字的大小为原来的两倍。

5.1.4 行高

在CSS中，还可以设置一个段落中各行文本的高度，这是通过line-height属性设置的。在CSS中line-height的值表示的是两行文字之间基线的距离，也就是每行文字的高度。如果给文字加上下划线，下划线的位置就是文字的基线。

line-height属性的值与CSS中所有设定具体数值的属性一样，可以设定为相对数值，也可以设定为绝对数值。在静态页面中，文字大小固定时常常使用绝对数值，达到统一的效果。而对于论坛和博客这些可以由用户自定义字体大小的页面，通常设定为相对数值，可以随着用户自定义的字体大小而改变相应的行距。

例如，对上面的例子的p标记设置如下CSS规则：

```
    line-height:18px;
```

这时在浏览器中的效果如图5.5所示。每行的高度比原来增大了一些。该文件请参考本书光盘中的“第5章/文字/line-height-1.htm”。

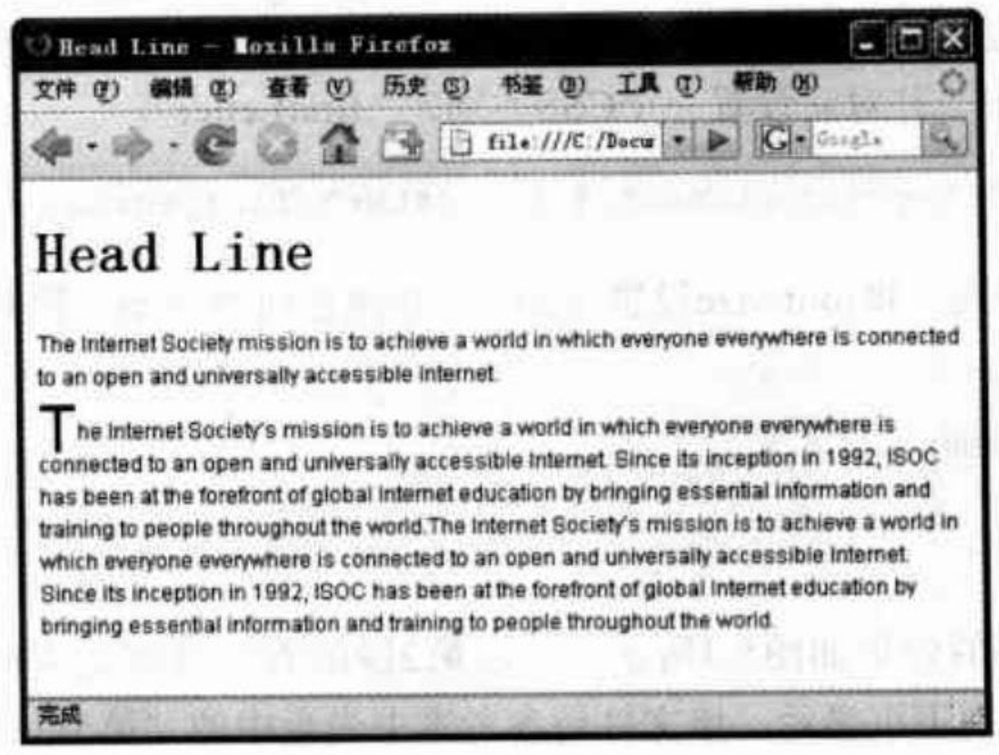


图5.5 设置正文的行高

除了可以使用像素等作为行高的单位，也可以不加任何单位，此时行高应写成与字体大小的比值。例如，字体大小是12像素，有下面这行代码：

```
    line-height:1.5;
```

它所产生的行高效果，与下面的代码的行高效果是相同的。

```
    line-height:18px;
```

下面请读者仔细对比一下图5.4与图5.5，可以发现在设置了“line-height:18px”之后，图5.4中下沉显示的段首字母“T”，不再以下沉的方式显示了。如果此时将行高由绝对高度

18像素改为相对高度1.5，就会发现字母“T”又下沉显示了，如图5.6所示。该文件请参考本书光盘中的“第5章/文字/line-height-2.htm”。

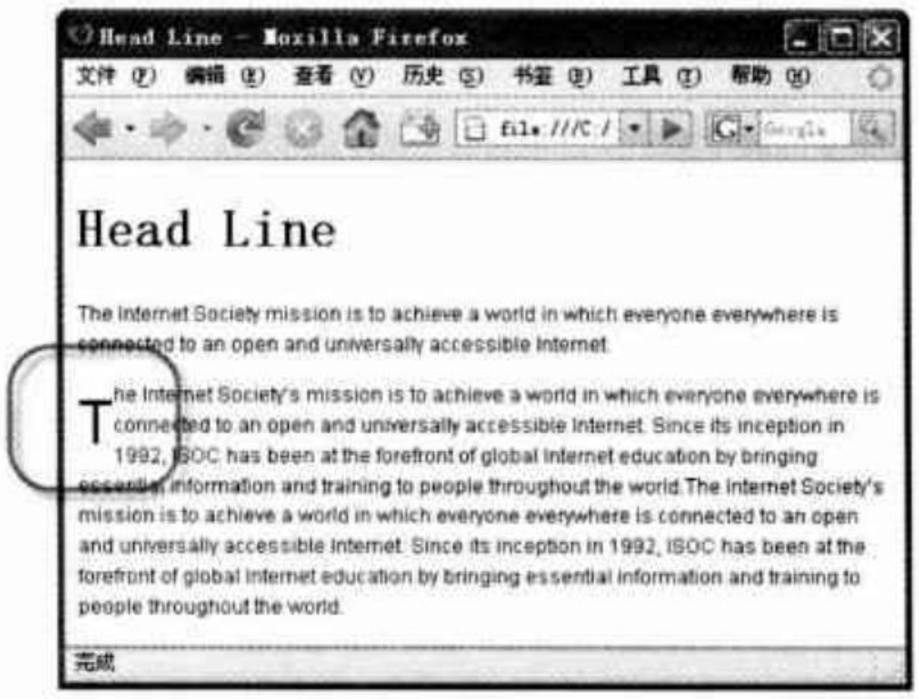


图5.6　使用相对高度后的效果

分析　12像素的1.5倍正是18像素，行高也没有变化，为什么会影响到前面的字母“T”的位置呢？

读者可以好好思考一下这个问题。这里考验的是读者是否已经充分理解了CSS的基本性质。答案是这样的，在本书第1章中曾经重点介绍过CSS的样式具有继承的性质。当p的样式中，将line-height设置为18像素时，字母“T”所在的span是p元素的子元素，因此它继承了这个样式，它的行高也是18像素。这样，尽管它的文字高度变大了，但是它的盒子高度仍然被限制为18像素，旁边的文字围绕它排版的时候是以盒子为界限的，因此效果就如图5.5所示了。（此时，字母“T”已经超出了盒子的范围，请读者试验一下，在IE和Firefox中的不同处理方式。）

当把p的line-height设置为1.5时，这个样式同样被字母“T”所在的span元素继承了，因此当它的文字变为3倍大的时候，行高也随之变大，就会产生如图5.6所示的效果了。

从这个小例子可以看出，在学习和实际工作过程中有时会遇到一些很细小的地方，其中蕴藏着很深的道理，都值得把它探究清楚。

下面还需要说明两点。

（1）上面介绍了设置文字的3个最基本的属性，即字体、大小和行高。在CSS中，还可以把它们组合在一条CSS规则中。例如：

```
font:12px/18px  Arial;
```

这行代码表示使用Arial字体，大小为12像素，行高18像素。注意在字体和行高之间要用斜线隔开。

也可以以相对比例作为行高的数值，例如：

```
font:12px/1.5  Arial;
```

（2）line-height属性可以设置在文本元素中，例如上面的例子就是这样，对p元素设置行高，就是确定了段落中每一行的高度。line-height属性也可以设置在容器元素中，例如对一个div设置line-height属性，那么它里面的文字都将使用这个行高值。

5.1.5 文字颜色与背景颜色

在HTML页面中，颜色统一采用RGB格式，也就是通常人们所说的“红绿蓝”三原色模式。每种颜色都由这3种颜色的不同比重组成，分为0~255档。当红绿蓝3个分量都设置为255时就是白色，例如rgb(100%,100%,100%)和#FFFFFF都指白色，其中“#FFFFFF”为十六进制的表示方法，前两位为红色分量，中间两位是绿色分量，最后两位是蓝色分量。“FF”即为十进制中的255。

当RGB的3个分量都为0时，就显示为黑色，即rgb(0%,0%,0%)和#000000都表示黑色。同理，当红、绿分量都为255，而蓝色分量为0时，则显示为黄色，即rgb(100%,100%,0)和#FFFF00都表示黄色。

文字的各种颜色配合其他页面元素组成了整个五彩缤纷的页面。在CSS中文字颜色是通过color属性设置的。下面的几种方法都可将文字设置为蓝色。

```
h3{ color: blue; }
h3{ color: #0000ff; }
h3{ color: #00f; }
h3{ color: rgb(0,0,255); }
h3{ color: rgb(0%,0%,100%); }
```

第1种方式使用颜色的英文名称作为属性值。

第2种方式是最常用的十六进制数值表示。

第3种方式是第2种方式的简写方式，形如#aabbcc的颜色值，就可以简写为#abc。

第4种方式是分别给出红绿蓝3个颜色分量的十进制数值。

第5种方式是分别给出红绿蓝3个颜色分量的百分比。

注意 如果读者对颜色的表示方法还不熟悉，或者希望了解各种颜色的具体名称，请参考网页http://learning.artech.cn/20061130.color-definition.html。

在CSS中，除了可以设置文字的颜色，还可以设置背景的颜色，使用的属性是background-color。例如继续设置上面的页面，设置h1标题的样式为：

```
h1{
  background:#678;
  color:white;
  }
```

将背景色设置为#678，也就是#667788，并将文字颜色设置为白色，这时页面效果如图5.7所示。该文件请参考本书光盘中的“第5章/文字/color.htm”。

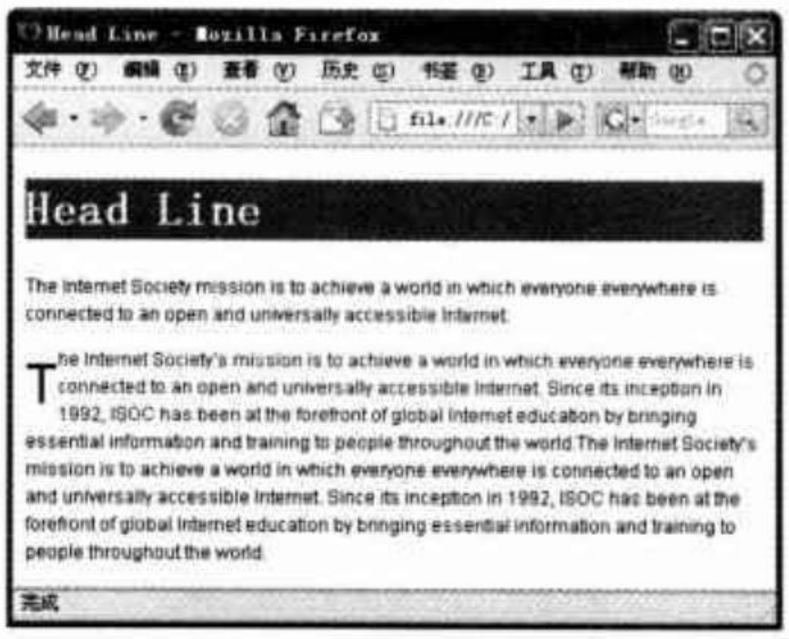


图5.7　设置标题的字体颜色和背景颜色

5.1.6 文字加粗、倾斜与大小写

在HTML语言中可以通过添加<b>标记或者标记将文字设置为粗体。在CSS中，是使用font-weight属性控制文字的粗细的，可以将文字的粗细进行细致的划分，更重要的是CSS还可以将本身是粗体的文字变为正常粗细。

但是目前CSS的规范中设置了相应的属性值，但是实际上在大多数操作系统和浏览器上，还不能很好地实现非常精细的文字加粗设置，通常只能设置“正常”和“加粗”两种粗细，分别如下：

```
font-weight:normal          /*正常*/
font-weight:bold            /*加粗*/
```

倾斜和粗体也有类似之处。大多数人对于字体倾斜的认识都是来自于Word等文字处理软件，例如图5.8左图所示的是一个Time New Roman字体的字母a，中间的图是它的常见的倾斜形式。

注意　文字的倾斜并不是真的通过把文字“拉斜”实现的，例如图5.8左图所示的正常的字体无论怎么倾斜，也不会产生中间图中的字形。倾斜的字体本身是一种独立存在的字体，对应于操作系统中的某一个字库文件。

严格来说，在英文中，字体的倾斜有以下两种。

（1）italic，即意大利体。我们平常说的倾斜都是指“意大利体”，这也就是为什么在各种文字处理软件上，实现字体倾斜的按钮大都使用字母“I”来表示的原因。

（2）oblique，即真正的倾斜，就是把一个字母向右边倾斜一定角度产生的效果，类似于图5.8右图显示的效果。这里说“类似于”是因为Windows操作系统中并没有实现oblique方式的字体，所以这里找了一个接近它的字体来示意。

图5.8　正常字体与“意大利体”、“倾斜体”的对比

CSS中的font-style属性正是用来控制字体倾斜的，它可以设置为“正常”、“意大利体”和“倾斜”3种样式，分别如下：

```
font-style:normal;
font-style:oblique;
font-style:italic;
```

注意 在Windows上，并不能区分oblique和italic，它们都是按照italic方式来显示的。这不是浏览器的问题，而是操作系统不够完善所造成的。

经验 对于中文字体来说，并不存在这么多情况。另外，中文字体的倾斜效果并不好看，因此网页上很少使用中文字体的倾斜效果。

除了加粗和倾斜之外，英文字母的大小写转换是CSS提供的很实用的功能之一，用户只需要设定英文段落的text-transform属性，就能很轻松地实现大小写的转换。

例如，下面3个文字段落分别可以实现单词的首字母大写、所有字母大写和所有字母小写。

```
p.one{ text-transform:capitalize; }        /* 单词首字大写 */
p.two{ text-transform:uppercase; }         /* 全部大写 */
p.three{ text-transform:lowercase; }       /* 全部小写 */
```

这里需要补充说明的是，由于西文字母数量很少，因此对于字母的样式还有很多非常复杂的属性，在CSS 2的规范中有很大篇幅的内容是关于字体属性的定义的。对于普通的设计师而言，不必研究得太深，只要把上面介绍的几点了解清楚，日常工作中基本就够用了。

5.1.7 文字的装饰效果

给文字加上下划线、顶划线和删除线的功能在文档编辑中的使用频率是很高的，在网页中尤其突出。CSS通过设置文字的text-decoration属性来实现这些特殊效果。

例如，下面的代码分别给文本设置了下划线、顶划线、删除线和文字闪烁效果。

```
p.one{ text-decoration:underline; }                /* 下划线 */
p.two{ text-decoration:overline; }                 /* 顶划线 */
p.three{ text-decoration:line-through; }           /* 删除线 */
p.four{ text-decoration:blink; }                   /* 闪烁 */
```

注意 需要注意的是，闪烁效果在IE中无效，在Firefox有效。

如果希望文字不仅有下划线，同时还有顶划线或者删除线，就可以将underline和overline的值同时赋给text-decoration，并用空格分开，如下所示。

```
    p{ text-decoration:underline overline line-through; }        /* 三种同
时 */
```

5.1.8 文字的水平对齐方式与段首缩进设置

在CSS中，段落的水平对齐是通过属性text-align来控制的，它的值可以设置为左、中、右和两端对齐等。控制段落文字的对齐方式就像在Word中一样方便。

例如，下面的代码将使h1标题的文字居中对齐。

```
    h1{text-align:center;}
```

要左对齐或者右对齐，只需将text-align属性设置为left或者right即可。如果要设置两端对齐，就将text-align属性设置为justify。

经验 中文排版的习惯中，在每一个段落的开头都应该空两格（英文排版没有这个习惯）。因此，段首的缩进控制对中文网页特别有用。

在CSS中，段首缩进是通过text-indent属性设置的，直接将缩进距离作为数值即可。对于中文网页，设置为“2em”即可，如下所示。

```
    p{text-indent:2em;}
```

5.1.9 文字布局

前面几节分别介绍了文字的一些样式属性，此外文本还可以和上一章中的盒子模型以及定位属性相结合来实现布局效果。

例如，将前面的页面稍加改进，就可以制作出如图5.9所示的效果。该文件请参考本书光盘中的“第5章/文字/float-layout.htm”。

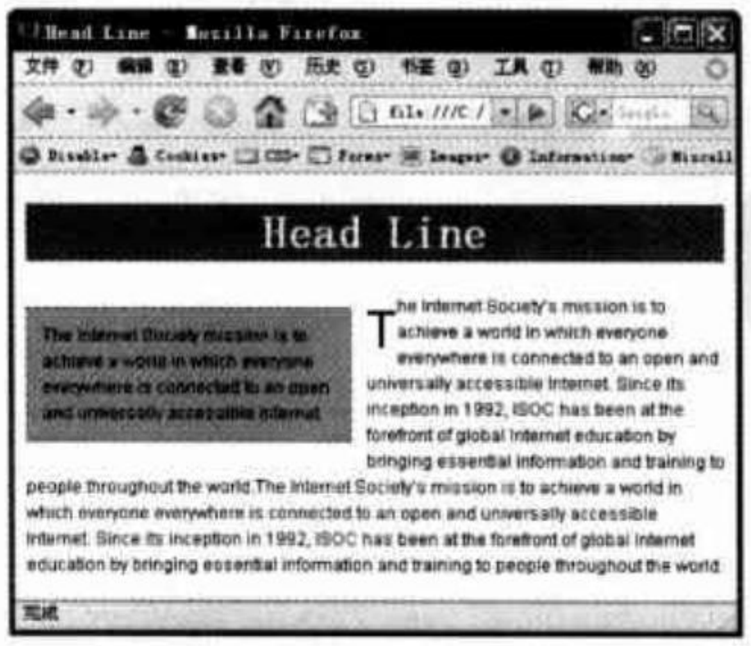


图5.9 设置了布局属性的页面

读者可以把这个例子作为对上一章内容的复习，自己来实践一下。这个页面中一共有两个段落，短的是p1，长的是p2。首先要给p1设置边框线和背景色，然后将p2段落的文字环绕

p1段落，显然需要将p1段落设置为浮动。为了使p1和p2的文字都距离边框有一定的空间，需要设置p1的padding和margin。完整的CSS代码如下。

```
h1{
  background:#678;
  color:white;
  texta-align:center;
  }
p{
  font-family:Arial;
  font-size:12px;
  line-height:1.5;
  /*上面3条规则可以简写为一条 font:12px/1.5  Arial */
  }
.firstLetter{
  font-size:3em;
  float:left;
  }
.p1{
  width:200px;
  float:left;
  margin:10px;
  margin-left:0;
  padding:10px;
  border:1px gray double;
  background:#9BD;
  }
```

5.1.10 段落的垂直对齐方式

上面介绍了文字的水平对齐方式，读者自然会想到竖直方向又该如何对齐呢？下面通过实例来进行讲解。将上面例子中的p1段落的高度设置为150像素，效果如图5.10所示。

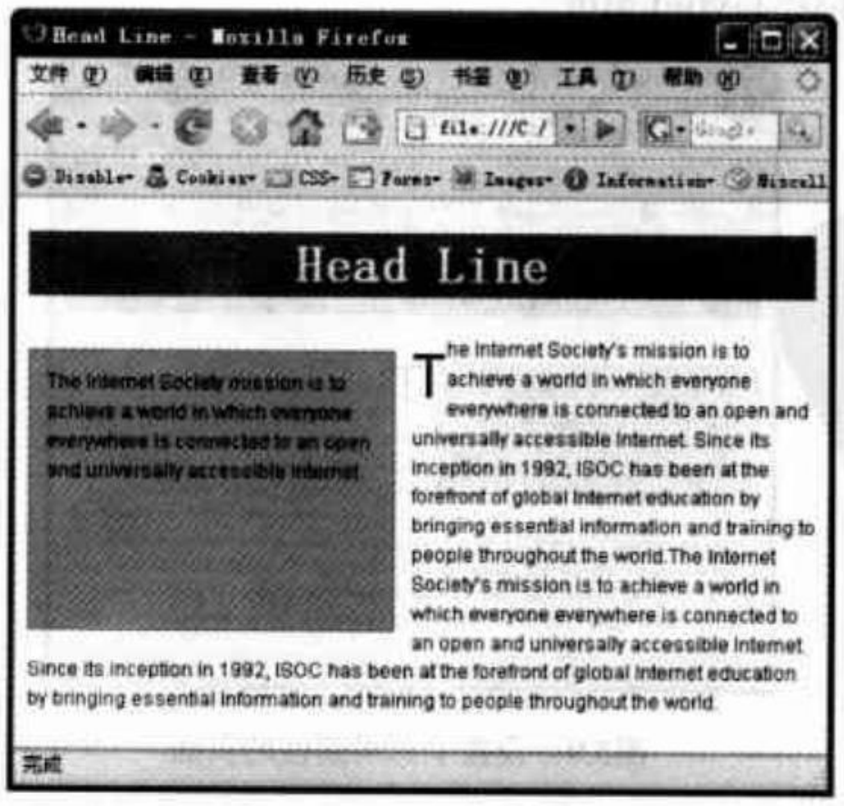


图5.10 将p1的高度设置为固定值

如果要使方框中的文字竖直方向居中对齐，应该如何实现呢？在CSS中有一个用于竖直方向对齐的属性——vertical-align。在目前的浏览器中，只能用表格单元格中的对象竖直方向的对齐设置，而对于一般的块级元素，例如div等，都是不起作用的。

也就是说，当文字在一个表格的单元格中时，如果对该单元格使用vertical-align:middle，那么该单元格中的内容会竖直方向居中对齐。如果文字放在一个div中，那么对这个div使用vertical-align:middle将不会有任何效果。

那么这个问题有解决办法吗？

如果文字内容只有一行，则可以使用line-height与height相同的办法使文字竖直居中。例如，假设有如下HTML代码：

```
<div class="middle">
  Here is ONE line of  text
</div>
```

相应的CSS设置为：

```
.middle{
  height:100px;
  line-height:100px;
  boeder:1px #666 solid;
  }
```

效果如图5.11所示，这里将行高设置为与高度相同的值，就可以保证文字竖直居中了。该文件请参考本书光盘中的“第5章/文字/sigle-vertical.htm”。

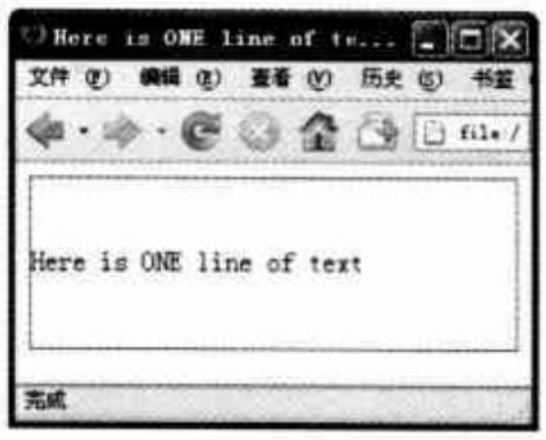


图5.11 单行文字的情况可以正确竖直居中对齐

但是对于超过一行的文本，例如增加文本的长度，或者是浏览器窗口变窄，是文本需要折行显示，这种方法就无效了，效果将如图5.12所示。

图5.12 两行文字的情况无法正确竖直居中对齐

因此上面的方法不是通用的解决方法。实际上解决这个问题的方法并不简单，必须要通过一定的非常规手段才能实现。严格来说，应该实现如下目标：

（1）div容器的高度固定；

（2）内部需要居中对齐的内容高度不固定，例如是服务器动态产生的数据；

（3）不使用表格。

捷克的设计师Dušan Janovský和aka Yuhů给出一个比较完善的解决方案，其中的逻辑比较复杂，需要对不同的浏览器，使用不同的代码，然后合并在一起。这里仅给出一个最终的代码，供读者直接使用在自己的网页中，如果读者有兴趣深入研究其中原理，请访问网址

http://www.jakpsatweb.cz/css/css-vertical-center-solution.html

以下代码引用自Dušan Janovský发表在互联网上的文章。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Universal vertical center with CSS</title>
  <style>
  #outer {height: 100px; overflow: hidden; position: relative;}
  #outer[id] {display: table; position: static;}

  #middle {position: absolute; top: 50%;} /* for explorer only*/
  #middle[id] {display: table-cell; vertical-align: middle; position:
  static;}

  #inner {position: relative; top: -50%} /* for explorer only */
  /* optional: #inner[id] {position: static;} */

  .withBorder{
        border:1px green solid;
        }
</style>
</head>

<body>
<div id="outer" class="withBorder">
  <div id="middle">
    <div id=" inner" >
      any text any height any content,
      everything is vertically centered.
    </div>
  </div>
  </div>
  </body>
  </html>
```

该页面的效果如图5.13所示，浏览器窗口宽度变化、文字折行都不会影响文字的竖直居中效果。该文件请参考本书光盘中的“第5章/文字/multi-vertical.htm”。

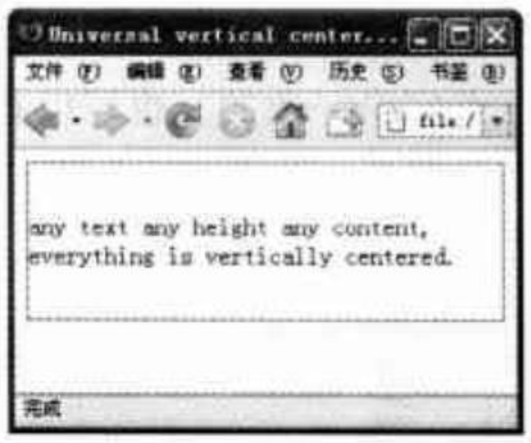


图5.13 多行文字实现在块级容器中竖直居中

这种方法需要使用嵌套的3层div才可以实现。如果读者需要使用这种方法，只需要将相应的高度换成需要的高度，其余的结构和CSS样式都不用修改，直接使用即可。

5.2 CSS图像样式

在五彩缤纷的网络世界中，各种各样的图片组成了丰富多彩的页面，能够让人更直观地感受网页所要传达给浏览者的信息。本节将介绍CSS设置图片风格样式的方法，包括图片的边框、对齐方式和图文混排等，并通过实例综合掌握文字和图片的各种运用。

5.2.1 基本设置

作为单独的图片，虽然它本身的很多属性都可以直接在HTML中进行调整，但是通过CSS统一管理，不但可以更加精确地调整图片的各种属性，还可以实现很多特殊的效果。本节主要讲解用CSS设置图片基本属性的方法，为进一步深入探讨打下基础。

图像的基本设置包括设置图像的边框、内外边距和大小等。在HTML中可以直接通过<img>标记的border属性值为图片添加边框，控制边框的粗细，border="1"表示边框为1像素粗细，当设置该值为0时，则显示为没有边框。

在HTML中，可以设置的样式很少而且很单调，而使用CSS的border属性则可以设置丰富得多的样式效果。

设置图像的边框，与设置其他元素的边框的方法完全相同。

例如，在前一节的基础上，把第一个文字段落换为一个<img>，即图像元素，然后设置如下CSS样式：

```
img{
  border:1px gray dashed;
  margin:10px 10px 10px 0;
  padding:5px;
  float:left;
  }
```

效果如图5.14所示，可以看到设置图像的边框样式、内边距、外边距和浮动等基本属性，与设置其他元素的基本属性的方法是完全一致的。也可以看出，CSS的盒子模型所起的巨大作用。在CSS中，任何一个对象本身都是一个盒子，对于一个图像而言，图像本身就是盒子的内容，因此边框、内外边距等部分就都是顺理成章的了。

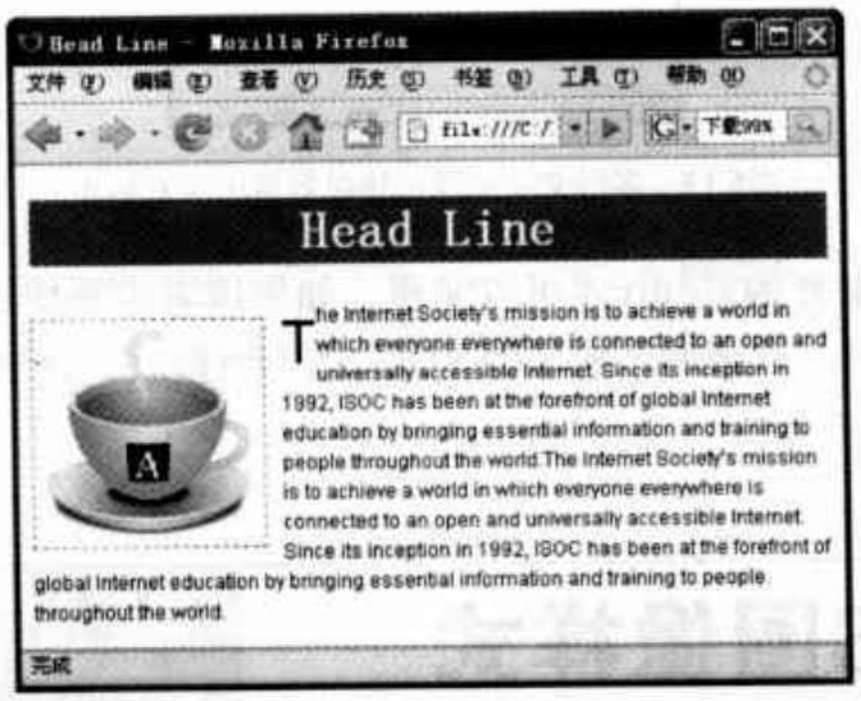


图5.14 对图像元素设置CSS样式

该文件请参考本书光盘中的“第5章/图像/basic.htm”。

在CSS中，也可以设置图像的高度和宽度，同样是使用盒子模型中统一的height和width属性。设置时可以使用具体的长度，例如“100px”，也可以使用相对比例，例如“70%”。

5.2.2 背景图像

在CSS中，图像经常以背景的形式出现，而且用途极为广泛。本书后面的章节中，读者还将学习到通过使用背景图像的方法，实现很多特殊的效果。

1. 设置背景图像

在前面已经介绍了使用background-color属性给元素设置背景颜色。在CSS中，还可以使用图像作为某个元素的背景，例如整个页面的背景使用背景图像来设置。设置背景图像要使用background-image属性。

仍然以上面的实例为基础，在CSS样式部分增加如下样式代码。

```
body{
background-image:url(bg.gif);
}
```

然后准备一个图像文件，如图5.15所示。

图5.15 准备一个图像文件

这个图片的左上部分为灰色，右下部分为白色。为了使页面上的文字不至于和背景混在一起，可以把p标记的背景色设置为白色，这时的效果如图5.16所示。

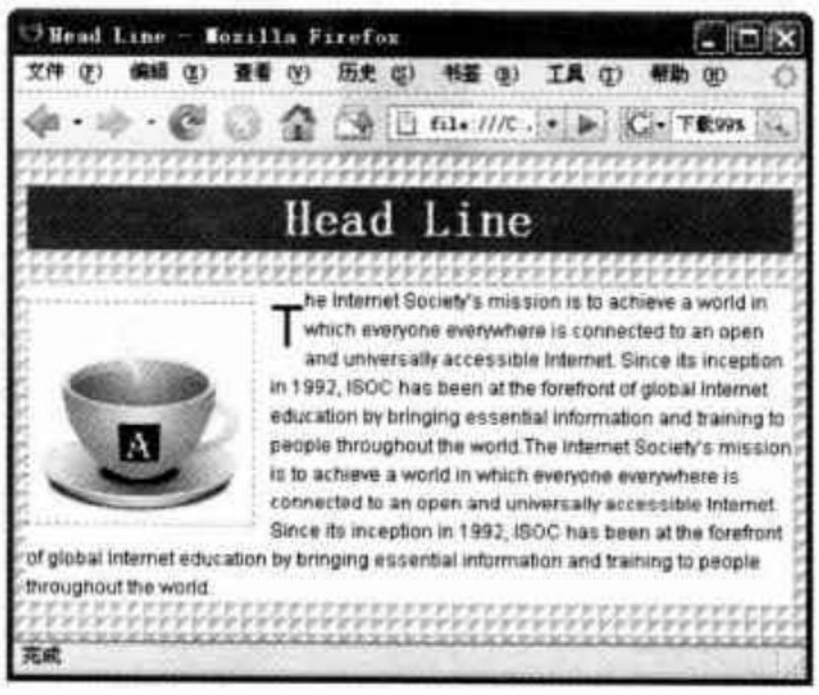


图5.16　页面的body元素设置了背景图像后的效果

可以看到，用这种方式设置背景图像以后，图像会自动沿着水平和竖直两个方向平铺。

其他元素也同样可以使用背景图像，例如将实例中的h1标记的背景由原来的背景色，改为使用图像作为背景，效果如图5.17所示。该文件请参考本书光盘中的“第5章/图像/backgroud.htm”。

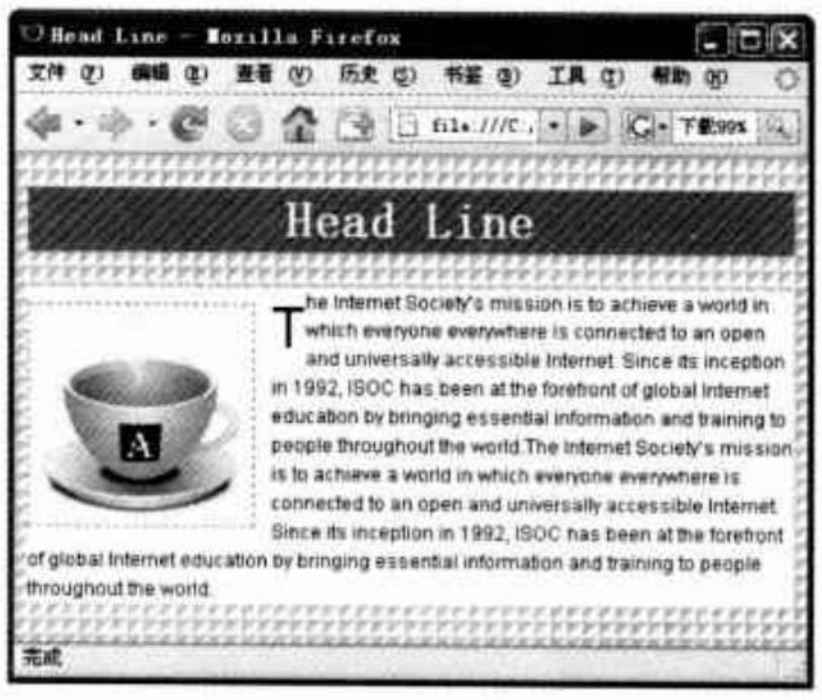


图5.17　h1标题使用背景图像的效果

2. 控制图像平铺方向

在默认情况下，图像会自动向水平和竖直两个方向平铺。如果不希望平铺，或者只希望沿着一个方向平铺，可以使用background-repeat属性来控制。该属性可以设置为4种平铺方式。

- repeat：沿水平和竖直两个方向平铺。
- no-repeat：不平铺，即只显示一次。
- repeat-x：只沿水平方向平铺。
- repeat-y：只沿竖直方向平铺。

例如，先准备一个如图5.18所示的图像。

图5.18 渐变色构成的背景图像

然后，对body元素设置如下CSS样式，并去除刚才对h1标题的背景图像的设置。

```
body{
background-image:url(bg-grad.gif);
background-repeat:repeat-x;
}
```

这时效果如图5.19所示，可以看到，背景图像只是沿着水平方向平铺了。

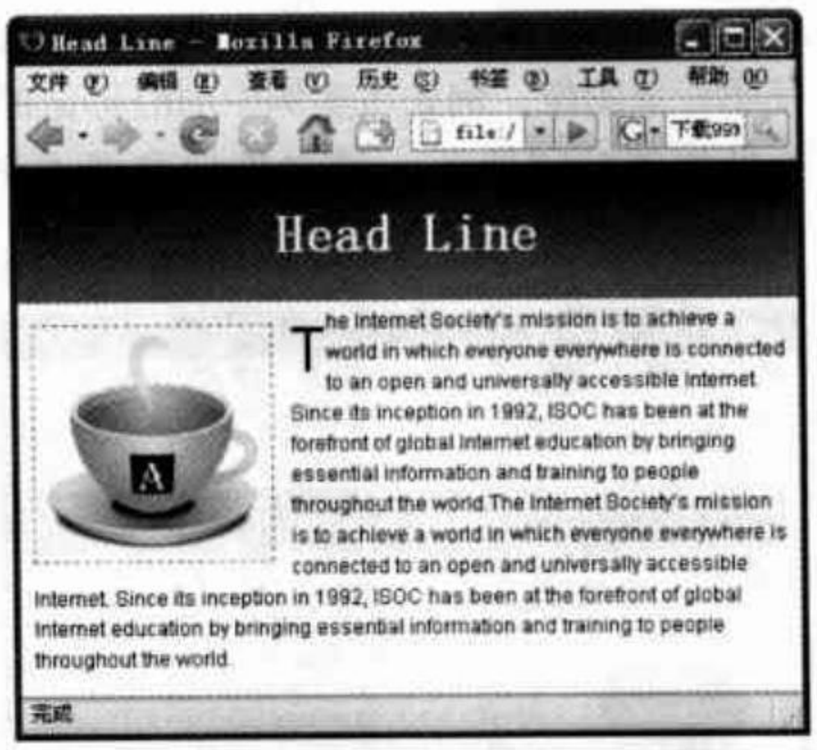


图5.19 水平方向平铺背景图像的效果

3. 同时设置背景图像和背景颜色

在CSS中还可以同时设置背景图像和背景颜色，这样背景图像覆盖的地方就显示背景图像，背景图像没有覆盖到地方就按照设置的背景颜色显示。例如，在上面的body元素CSS设置中，增加一条规则：

```
background-color:#3399FF;
```

这时效果将如图5.20所示。该文件请参考本书光盘中的“第5章/图像/repeat-image.htm”。

这个技巧非常有用，大家看到图5.20中的背景色的过渡非常自然，这是因为背景颜色正好设置为背景图像中最下面一排像素的颜色，这样可以制作出非常自然的渐变色背景。而且可以保证，无论页面多高，颜色都可以一直延伸到页面的最下端。

如图5.21所示的是CSS禅意花园网站的第158号作品，可以看到这个非常漂亮的网页的背景色，正是使用这种方法制作出来的。

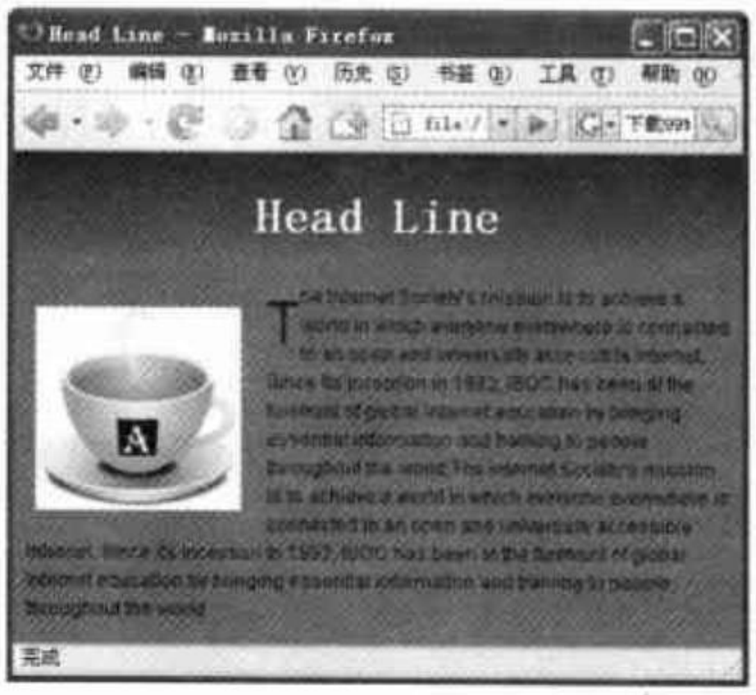


图5.20　同时设置背景图像和背景颜色

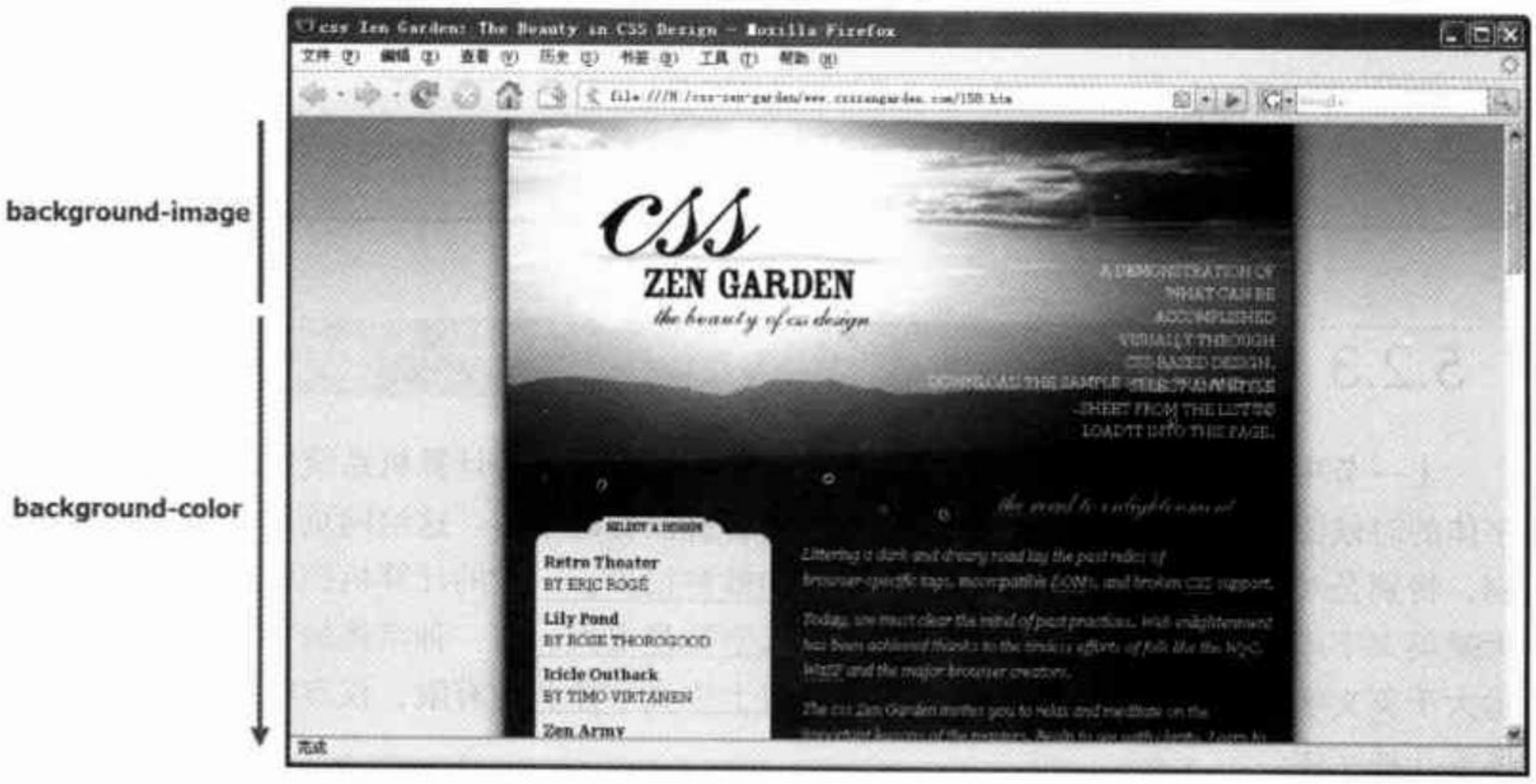


图5.21　CSS禅意花园的158号作品效果图

其上部使用一个水平方向平铺的图像背景，下面则是整体一致的背景颜色，如图5.22所示。

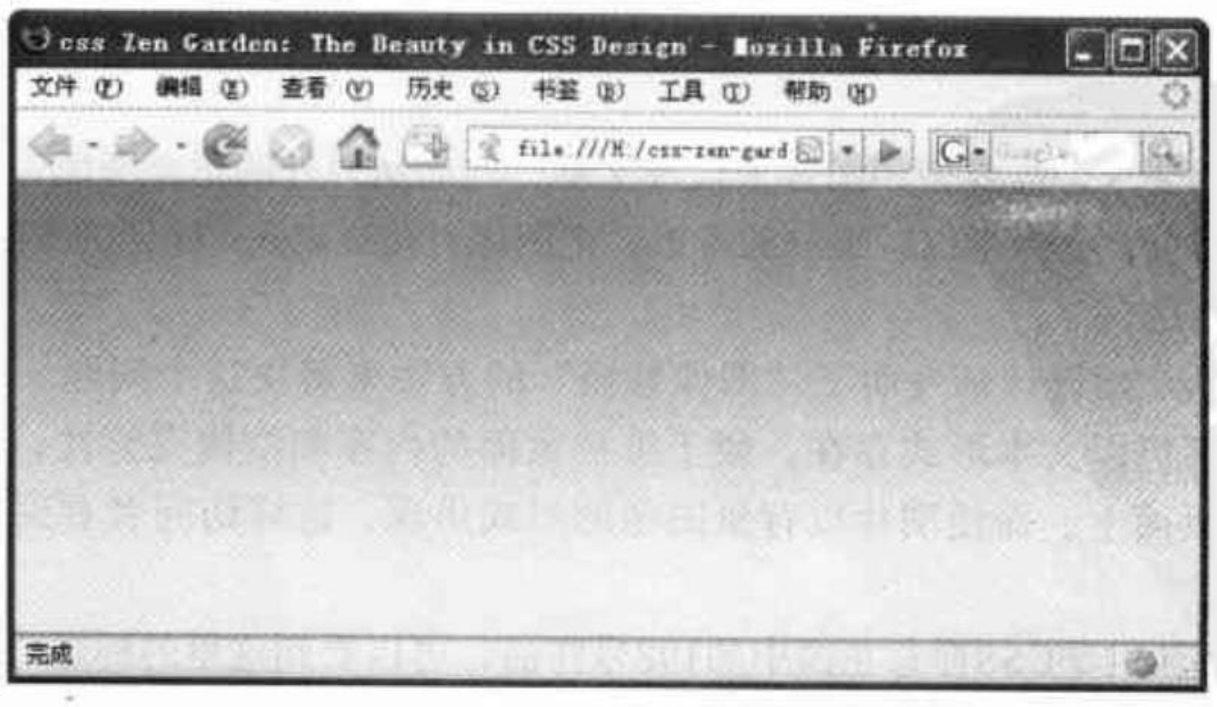


图5.22　隐藏页面其他内容后的效果

4. 背景样式属性的简写

就如同font、border等属性在CSS中可以简写一样，背景样式的CSS属性也可以简写。例如下面这段样式，使用了3条CSS规则。

```
body{
background-image:url(bg-grad.gif);
background-repeat:repeat-x;
background-color:#3399FF;
}
```

它完全等价于下面这条CSS规则。

```
body{
background:  #3399FF  url(bg-grad.gif)  repeat-x;
}
```

注意 属性之间用空格分隔。本书后面的章节中将主要使用这种简写的形式。

5.2.3 标题的图像替换

上一节中曾经讲到，由于文字的显示字体依赖于访问者的计算机系统情况，因此在使用字体的时候要特别谨慎，防止使用大多数浏览者都没有的字体。这给网页设计带来了很大限制，特别是对于中文网页的设计。英文字母的数量很少，一般的计算机操作系统中都配置有大量英文字库，所以字体很丰富。而中文的汉字数量很大，每一种字体的字库文件大小都远远大于英文字库文件，导致在一般人的计算机上中文字体非常有限，仅有很基本的宋体、黑体等几种字体。

1. 标题的图像替换

对于正文字体，通常在几种基本的字体中选择即可，而对于标题文字，如果仍然使用这几种最基本的字体，就会影响网页的美观。因此，设计师通常使用图像代替文本的方法来设置标题。

为了美观的要求，需要使用图像来代替文本，然而从另外的角度考虑，为了便于搜索引擎理解和收录网页，也为了以后维护的考虑，把图像直接以<img>标记的方式嵌入到网页中，也不是一个好办法。

因此，一些CSS设计师发明了“图像替换”的方法来解决这个问题。其核心思想是使HTML中的文字仍以文本形式存在，便于维持页面的内容和结构完整性，然后通过CSS使文字不显示在页面上，而使图片以背景图像的形式出现，这样访问者看到的就是美观的图像了。

如图5.23所示的是CSS禅意花园中的198号作品，可以看到这里的标题显然用文本是不可能制作出来的，只有用图像才能产生这样的效果。

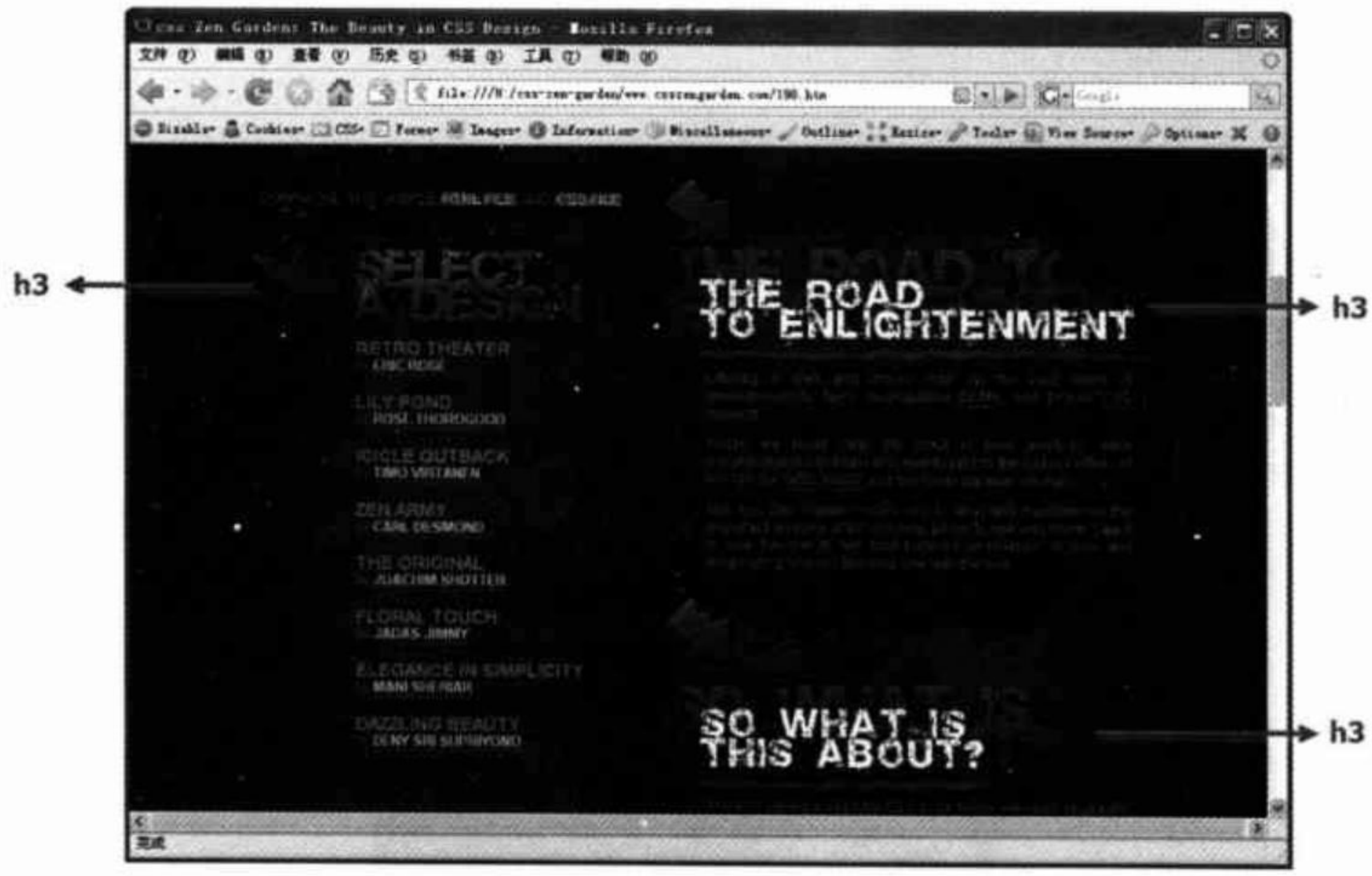


图5.23　CSS禅意花园的198号作品

其中的每一个标题，都对应一个图像文件，如图5.24所示。

图5.24　标题对应的图像

下面就来实际演示一下图像替换的具体方法。仍以图5.19所示的网页为基础，通过使用图像替换的方法，使标题看起来更美观，而且不依赖于访问者的字体文件。

① 首先准备标题图像，基于刚才的渐变色的背景，制作如图5.25所示的图像，这个图像的高度是60像素。

图5.25　制作一个标题背景图像

② 在HTML代码中的h1标题内部加入一对标记。

```
<h1><span>Head Line</span></h1>
```

③ 设置h1的CSS属性，将上面制作好的图像作为h1的背景图像，并设置为不平铺，并给出高度60像素。

```
h1{
 background:url(bg-h1.gif) no-repeat;
```

```
    height:60px;
}
```

④ 将span元素通过display属性，设置为不显示，目的是把原来文本显示的文字隐藏起来。

```
h1 span{
  display:none;
}
```

这时效果如图5.26所示。这时可以看到原先的文字已经由图像代替了。该文件请参考本书光盘中的“第5章/图像/header-image-1.htm”。

图5.26 图像替换的效果

2. 标题图像与背景融合

如果仔细观察图中框线圈起来的部分，可以发现这个标题图像并没有非常好地与背景融合。如果页面的背景是单一的颜色，就不会出现这个问题，因为标题的背景图像使用同样的背景色，可以保证标题的图像融合于页面的背景之中，而这里的页面背景是渐变色，如果页面背景的渐变色和h1标题图像的渐变色在竖直方向的位置没有完全匹配，就会出现图中的这个问题。

解决这个问题的方法是，在图像处理软件（如Fireworks）中，先把标题图像的背景色设置为单一颜色，这种颜色是介于渐变色两端的中间色，如图5.27左图所示。然后再导出为透明的GIF格式图像，而图像中与背景色接触的位置，因消除锯齿，会保留一点点蓝色的痕迹，如图5.27右图所示。

图5.27 改进背景图像

这时，用新的标题背景图像替换原来的标题背景图像，代码不需要修改，效果如图5.28所示。

图5.28　改进后的效果

这时就不存在刚才的生硬的颜色边界了。这种方法是经常用到的，例如在禅意花园的158号作品中使用了如图5.29所示的一些标题图像。

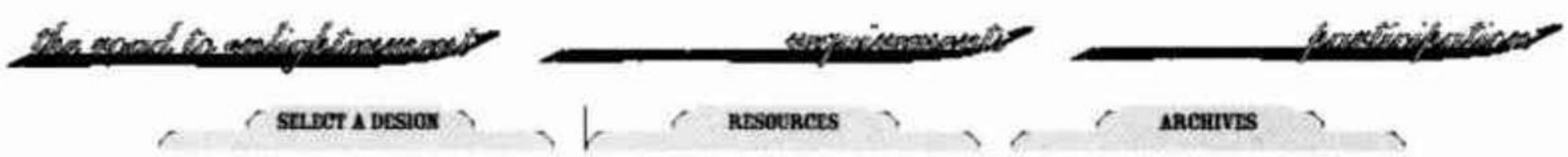


图5.29　禅意花园的158号作品中用到的标题背景图像

可以看到，它们的周围都带有一些“毛刺”，这就是为了使图像与背景融合而产生的背景透明的图像。

3. 标题的对齐方式

如果希望标题能够居中或者右对齐，那么该怎么办呢？这时可以使用background-position属性。

例如，希望标题右对齐，可以在h1的CSS中增加如下设置。

```
background-position:right;
```

这样，背景图像就从h1标题的最右端开始放置了，如图5.30所示。该文件请参考本书光盘中的“第5章/图像/header-image-2.htm”。

图5.30　使标题图像靠右放置

background-position的值可以为left、right、top、bottom和center，分别表示左、右、上、下和中间。

background-position属性可以设置为一个或者两个属性值。如果设置为两个属性值，它们分别用于确定水平位置和竖直位置。例如下面这段代码，表示从元素的右下角开始放置背景图像。

```
background-position:  right  bottom;
```

如果只设置一个值，那么缺省的那个值为center。例如下面这段代码，就表示水平方向为right，竖直方向为center。

```
background-position:  right;
```

而下面的代码表示竖直方向为top，水平方向为center。

```
background-position:  top;
```

此外，background-position属性也与平铺相关。它实际上指定的是平铺的起点。假设有如下CSS代码：

```
body{
  background: white url("pendant.gif");
  background-repeat: repeat-y;
  background-position: center;
}
```

效果如图5.31所示，背景图像将以background-position属性确定的位置作为起点，然后按照background-repeat属性确定的平铺方式进行平铺。

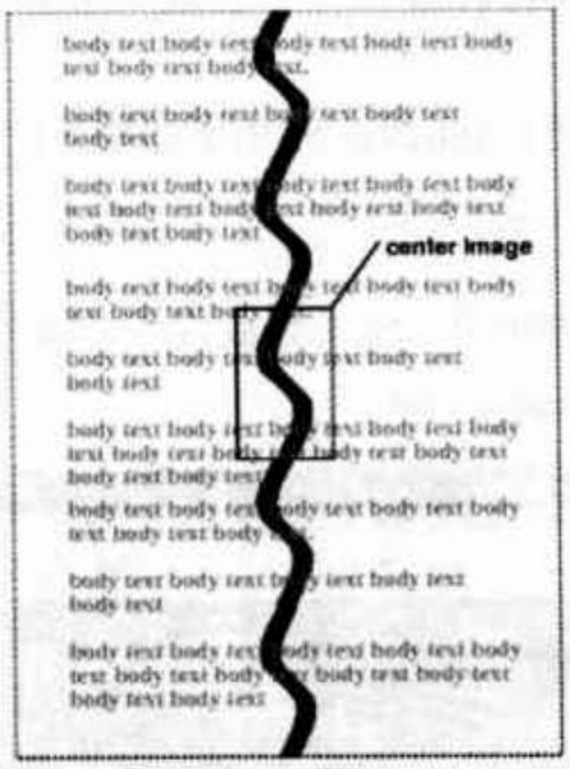


图5.31 背景图像位置与平铺效果示意图

上面案例中使用的这种图像替换方法称为“FIR”（Fahrner Image Replacement）法，是由美国设计师Todd Fahrner和英国设计师C.Z. Robertson开创的。这是最早也是最容易理解的一种方法。然而由于这种方法是通过display属性把文字隐藏起来的，因此会导致非屏幕浏览器的设备无法正确表现相应的内容。这里说的“非屏幕浏览器”包括为盲人设计的阅读器，它可以通过发声的方式表现网页内容，这样被隐藏起来文字就无法正确地被读出来。

目前国内对这方面的要求还不是太高，通常不会遇到太多问题。但是国内相关的软件和设备也开始出现和应用于一些特定的人群中，因此设计师也应该对此有所了解。

4. Phark图像替换方法

除了FIR法之外，还有一种常用的图像替换方法“Phark”法，它是由Mike Rundle发明的，做法是将标题的text-indent属性设置为一个非常大的负值，例如-10000，这样这个文字实际上仍然在网页上，但是不会显示出来，其余部分和FIR法相同。这样做的好处就是，通过屏幕朗读器或软件，仍然可以正确地读出相关的文字。

5. 改进的图像替换法

但是FIR法和Phark法都存在一个缺陷，就是如果一个浏览器不支持或关闭了CSS，那么这些文字就不能显示出来了。为此，TomGlider和Levin Alexander共同发明了另一种改进的图像替换方法。这种方法的核心思想是标题标记内部，加入一对空的标记，例如：

```
<h1><span> </span> Head Line </h1>
```

这对标记没有实际的语义作用，仅作为一个CSS“钩子”，用于设置背景图像。然后通过对span元素使用绝对定位，将图像覆盖在文字的上面，这样即使关闭了CSS，仍然可以看到文本标题。这个方法的缺陷是，标题图像不能使用透明的图像，否则会露出现文字，因此上面制作的实例就不能使用这种方法实现。

事实上，“图像替换”的方法还有很多，读者如果有兴趣，可以到互联网上寻找更多的相关学习资料，在这里就不再进一步深入讲解了。

5.2.4 为图像增加投影效果

在图像处理软件中，为图像添加阴影效果是非常容易实现的功能。为图像添加阴影可以使图像看起来精致很多。但是如果一个网站上需要放置很多图像，每个图像都要先在图像处理软件中增加阴影，就会很费时费力，而且如果以后要去掉阴影效果，又要重新处理图像。因此，如果能够不改变图像本身，而通过CSS实现阴影效果，将是非常实用的一个技术，以后需要调整样式时不用改动图像，也很方便。图5.32中所示的就是一个图像的阴影效果。

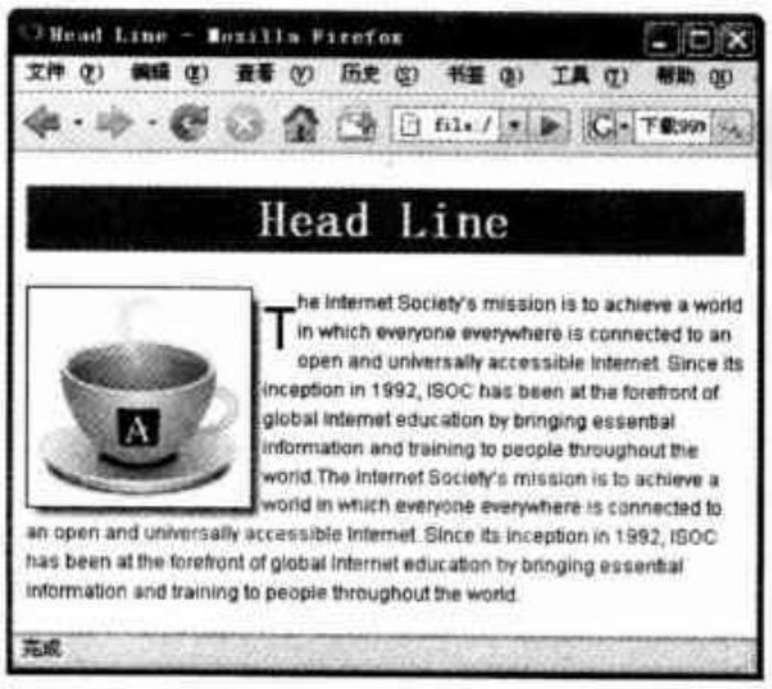


图5.32 给图像增加阴影效果

这里介绍的制作阴影的方法是基于墨西哥设计师Sergio Villarreal在2004年2月和4月发表在电子杂志A List Apart 172期和178期上的两篇文章形成的。如果读者有兴趣，可以到A List Apart网站查阅相关的资料。

1. 基本方法

显然我们的目标不是针对某个固定大小的图像增加阴影，而是可以对不同大小的图像都适用。制作阴影效果的基本方法如下。

① 首先制作一个如图5.33所示的图像，命名为bottom-right.gif。

图5.33 准备一个比较大的阴影图像

经验 这个图像的大小可以随意确定，但是如果图像大小大于这个阴影图像的大小，阴影就不能完全显示。因此，这个图像应该比将来要设置的图像大一些。

② 在HTML代码中的img元素外面增加一个div元素，并设置该div的CSS类别为shadow。

```
<div class="shadow"><img src="cup.gif"/></div>
```

③ 在shadow中设置背景图像，并把原来放在图像中设置的浮动属性移动到div中。

```
.shadow{
background: url(images/bottom-right.gif) no-repeat bottom right;
float:left;
}
```

④ 由于div设置为向左浮动，因此它的大小会自动与里面的图像一样大，这样阴影就被图像遮盖住了。可以通过设置margin，将图像向左和向上各移动6像素，这正是阴影的宽度。

```
img{
border:1px #000 solid;
padding: 4px;
margin:-6px 6px 6px -6px;
}
```

⑤ 这时效果如图5.34所示。可以看到，这个带有阴影的图像就产生了。该文件请参考本书光盘中的“第5章/图像/shaow-1.htm”。

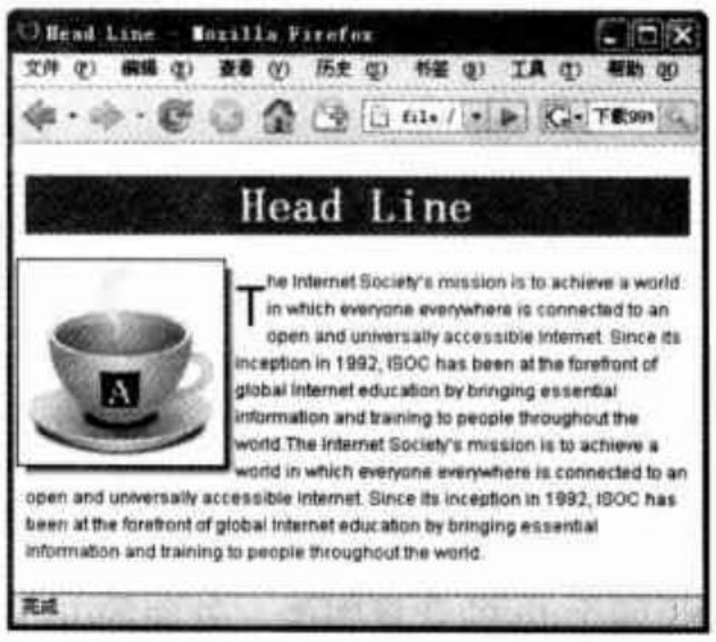


图5.34　最简单的办法实现的阴影效果

这种方法非常简单，但读者要注意，图5.34中使用的是Firefox浏览器。如果用IE 6，效果就如图5.35所示。

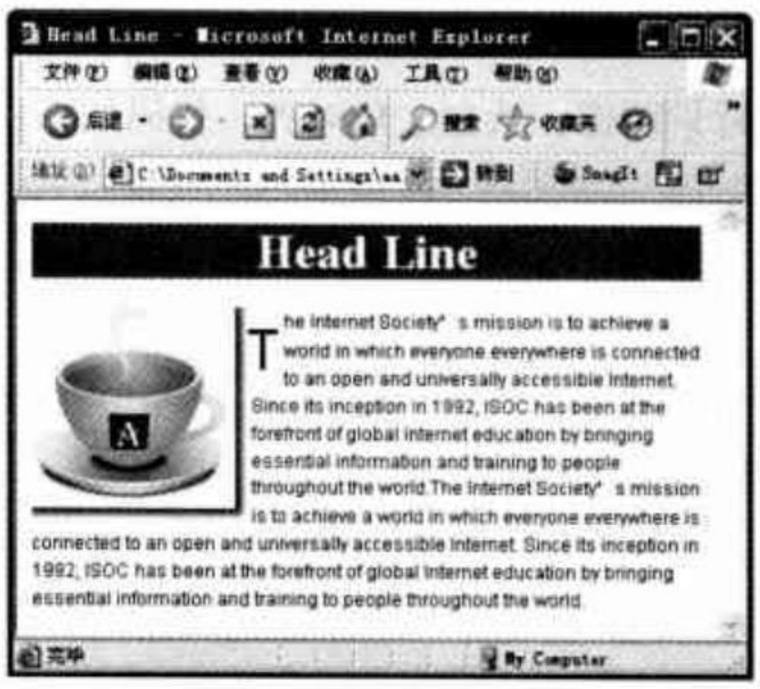


图5.35　在IE中的错误显示效果

可以看到，图像中左上部分超出div盒子的部分被剪裁掉了，因此这种方法在IE 6中是不适用的。

2. 使用滑动门技术

可以使用滑动门技术来解决这个问题。具体的方法如下。

① 现在再准备一个部分透明的GIF格式图像，图像的上面和下面各有6像素的白色不透明区域，其余区域均为透明色，如图5.36所示，命名为top-left.gif。

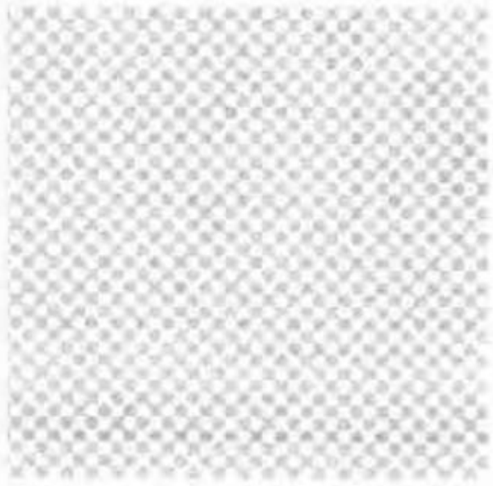
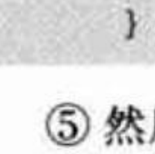


图5.36　准备另一个背景图像

② 改造HTML代码，在图像标记的外面套一层 标记。

```
<div class="shadow">
  <div><img src="cup.gif"/></div>
</div>
```



注意 写代码的时候 与<img>标记之间不要有空格和换行，否则IE浏览器会为空格字符设置空白的空间。

③ 分别将两个图像设置为两个div的背景图像，外面的div用原来的背景图像，从右下角对齐，代码如下：

```
.shadow {
  float:left;
  background: url(images/bottom-right.gif) no-repeat bottom right;
  }
```

④ 里面的div使用新的背景图像，从左上角对齐，代码如下：

```
.shadow div {
  background: url(images/top-left.gif) no-repeat;
  padding: 0 6px 6px 0;
  }
```

⑤ 然后设置图像的基本属性，代码如下：

```
.shadow img {
  border: 1px solid #000;
  padding: 4px;
  }
```

这时在IE中的效果如图5.37所示，就可以得到希望的效果了。在Firefox中当然也同样可以得到正确的效果。该文件请参考本书光盘中的“第5章/图像/shaow-2.htm”。

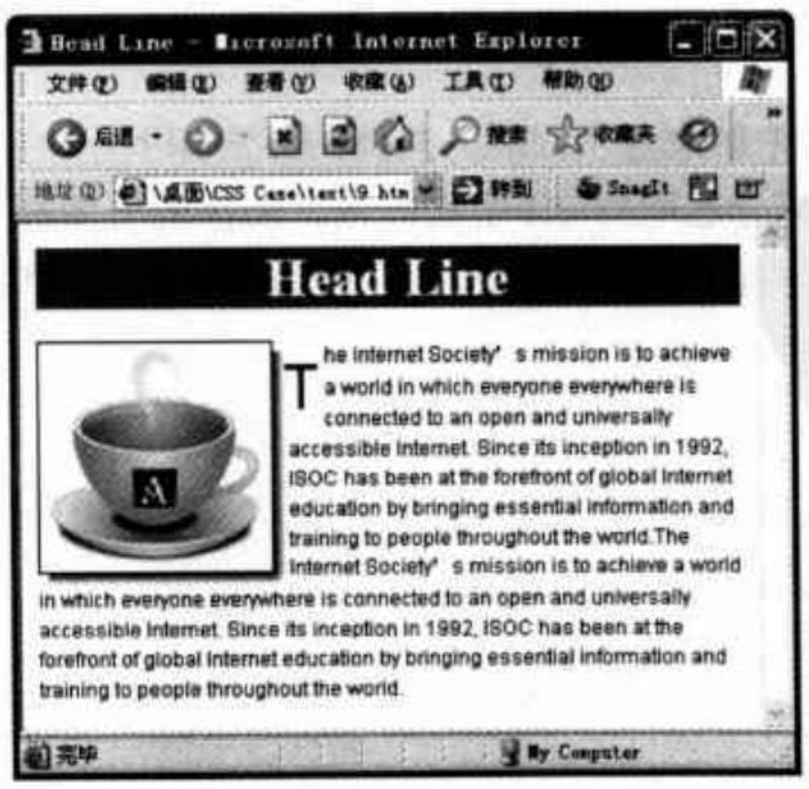


图5.37 使用新的方法在IE中也可以正确显示

对于这种方法的原理，请读者参考如图5.38所示的示意图。

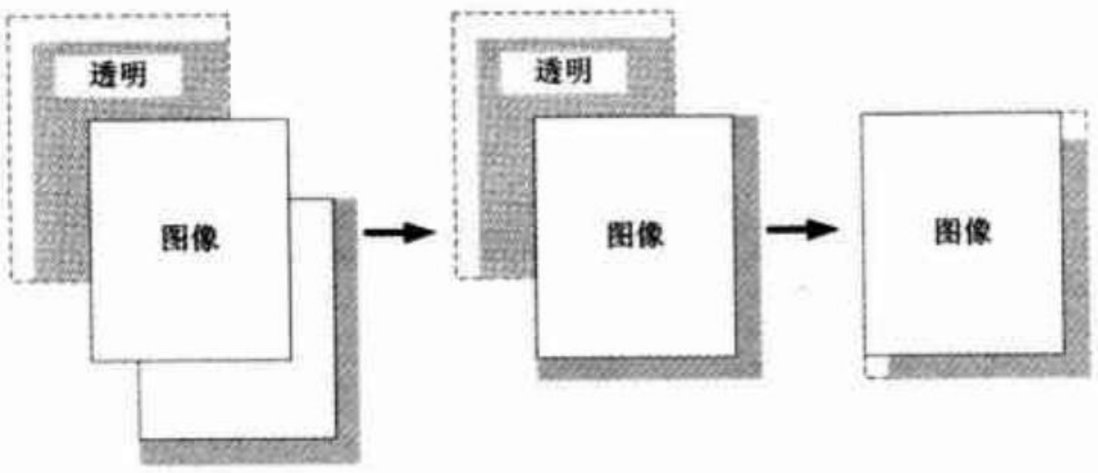


图5.38　原理示意图

图5.38中一共有3个矩形，分别代表两个div和一个图像内容。实际上两个div的大小是一样的，它们的背景图像重叠在一起，上面的div的背景图像中不透明的白色部分遮住阴影的一部分，就产生了最终的效果。

之所以这种方法叫做“滑动门”技术，是因为两个重叠的背景图像共同组成最终的效果，对于不同大小的图像，这种方法可以自动适应（在不超过最大面积的前提下），如图5.39所示。在本书后面的章节中，还会多次遇到滑动门技术，它可以解决很多使用其他常规方法不容易解决的问题。

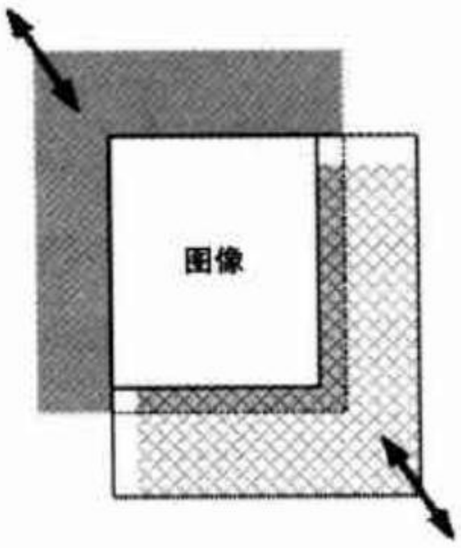


图5.39　滑动门的含义示意图

3. 柔边阴影效果

到这里，阴影效果已经比较完善了，但是还有可以改进的地方，仔细观察图5.40中阴影端点处的效果，显然左图的效果要比右图柔和、自然得多。而前面介绍的方法，只能制作出右图中的效果。

如果把左图放大观察，效果如图5.41所示。

下面讲解如何实现图5.41中的柔边效果。要实现图中左图的效果，需要重新准备图像，替换刚才的top-left.gif文件。

刚才使用的是GIF格式的图像，由于这种格式是很多年前制定的标准，因此不支持半透明的效果，即Gif图像中的像素，或者完全透明，或者完全不透明，不能实现半透明效果。

因此，必须要使用新的图像文件格式——PNG格式，它支持半透明效果，又称为“alpha”透明。具体的操作步骤如下。

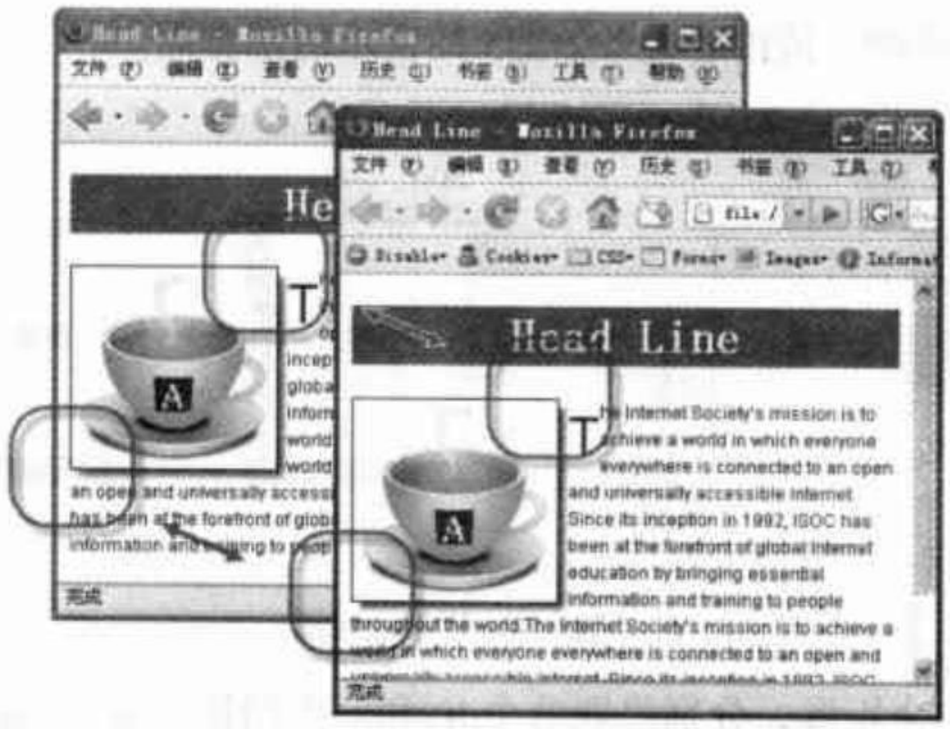


图5.40 柔边的阴影与硬边阴影

图5.41 放大观察柔边效果

① 在Fireworks或者Photoshop中，先把画布设置为一种深颜色，这样便于制作时看清楚渐变的效果。然后制作出如图5.42左图所示的效果，上面和左面各有一条白色羽化边界的横条和竖条，注意过渡要均匀。然后把画布的颜色设置为无色，在软件中看到效果应该如图5.41右图所示，也就是上边和左边是不透明的白色，然后柔和地过渡到完全透明。

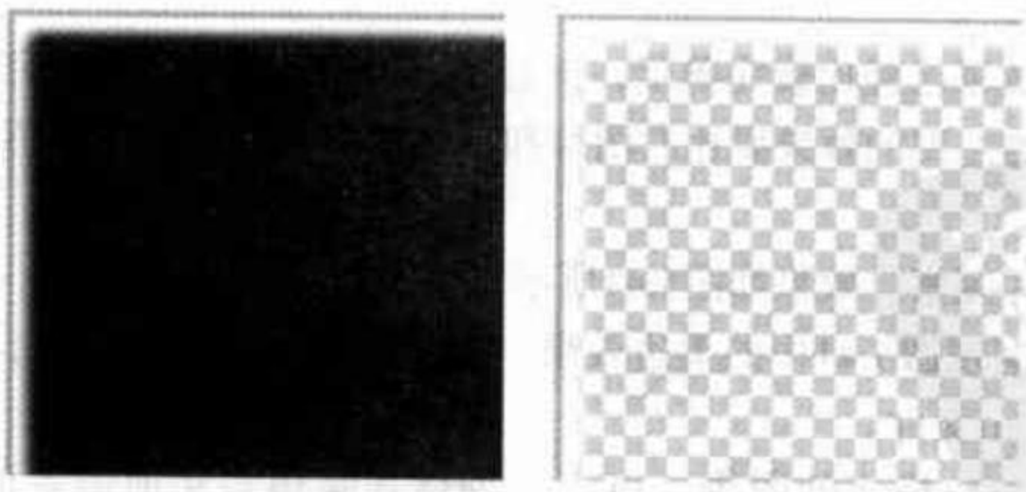


图5.42 制作半透明的PNG格式图像

② 导出为PNG格式的图像。注意在导出时，除了要选择PNG格式之外，还要选择正确的透明模式，应选择“Alpha透明度”选项，如图5.43所示。不要选择“索引色透明”，索引色透明就是GIF的透明方式。

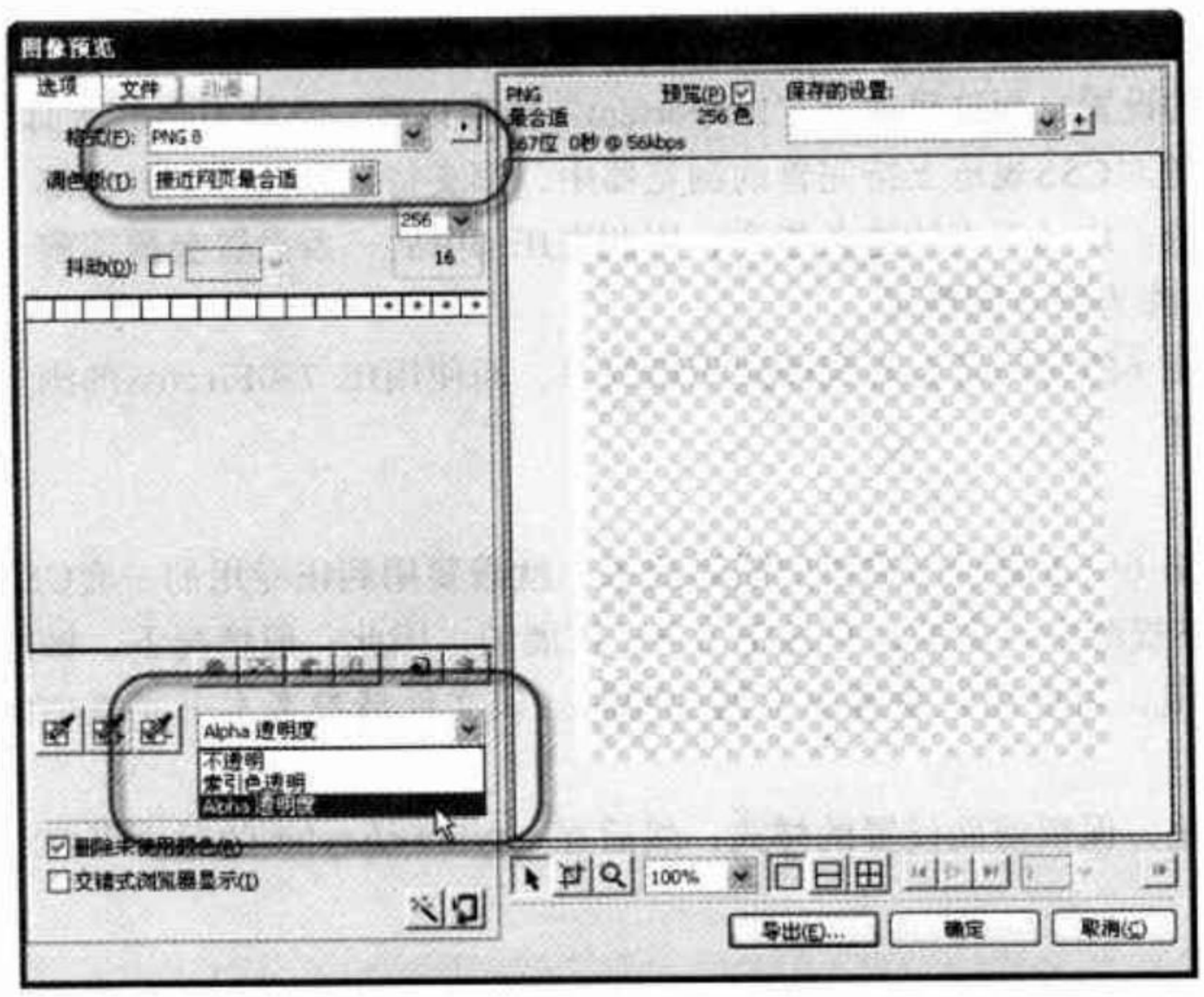


图5.43 在Fireworks中导出为Alpha透明的PNG格式

③ 导出为top-left.png文件以后，把CSS中原来的top-left.gif改为top-left.png就可以了。注意要用Firefox或者IE 7来查看效果。效果应该如图5.40左图所示。该文件请参考本书光盘中的“第5章/图像/shaow-3.htm”。

4. IE 6兼容的方案

遗憾的是，IE 6并不支持PNG格式的Alpha透明，直到IE 7才支持。那么应该如何让IE 6实现所需的效果呢？这里给出两种解决方案。

（1）方案一

首先介绍第一种方案，它比较省事但效果不如第2种方法。该文件请参考本书光盘中的“第5章/图像/shaow-4.htm”。还是以上面这个例子来讲解。已经准备好了top-left.gif和top-left.png这两个文件，修改如下内层div的样式：

```
shadow div {
  background: url(images/top-left.gif) no-repeat;
  padding: 0 6px 6px 0;
  }
```

改为：

```
 shadow div {
 background: url(images/top-left.png) no-repeat  !important;
 background: url(images/top-left.gif) no-repeat;
 padding: 0 6px 6px 0;
 }
```

在代码中出现了两条对background属性的设置规则，前一条使用PNG格式的图像，后一条使用GIF格式的图像。在前一条规则的最后加了一个“!important”指令，它的含义是提升

这条规则的优先级。按照默认的优先级规则，如果两条CSS规则的特殊性相同，那么后面的设置覆盖前面的设置，而这里加上“!important”指令以后，它就不会被后面设置覆盖了。

在Firefox等对CSS规范支持完善的浏览器中，都支持这条指令，因此都会使用PNG格式的图像作为背景。IE 6不支持这条指令，因此在IE 6中后一条设置会覆盖前一条设置，就会用GIF格式图像作为背景图像了。

这样使用IE 6的访问者会看到硬边阴影效果，而使用IE 7和Firefox的浏览者看到的则是柔边的阴影效果。

（2）方案二

如果要在IE 6中也显示出柔边的阴影效果，就需要用到IE专用的一套CSS滤镜了。这些滤镜都不是CSS规范中定义的，只是IE 6自己扩展的，因此一般情况下，做网页都不使用它们。但是在这里，可以用它来解决上面的问题。该文件请参考本书光盘中的“第5章/图像/shaow-5.htm”。

具体方法是，保留前面设置的样式，然后在<style></style>的后面再增加一段样式，代码如下：

```
<!--[if gte ie 6]>
<style type="text/css">
.shadow div {
  filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(
                                    src="images/top-left-
2.png",sizingMethod='crop');
  background:none;
  /*如果必要时可以增加这条规则: width: 1px;*/
  }
</style>
<![endif]-->
```

可以看到，<style></style>前后各有一个语句，它们称为“条件注释”，其作用是当浏览器是IE，并且在IE 6以上，里面内容就会有效。也就是说，增加的这段样式只有在IE 6以上版本才有效，其他浏览器都会忽略掉这段代码，因此这段代码不会影响在Firefox中的显示。

在IE 6中，将使用这段代码中的样式，其中共有两条CSS规则。第一条的作用是使用IE专有的滤镜，装入Alpha透明的PNG图像；第二条的作用是把前面设置的背景图像隐藏。

经验 理论上说，这样修改代码以后，在IE中就应该可以出现柔边效果的阴影了。但是本书测试的结果发现仅这样做，还是无法显示出正确的效果，PNG图像并不会显示出来。经过多次试验以后，发现有两个地方会导致上述方法失效，需要读者特别注意。

（1）PNG格式本身包括不同的类型，有8位的、24位的和32位的3种。在前面制作Alpha透明的PNG时，通常为了减小文件体积，使用8位的PNG格式，然而IE 6的滤镜必须使用32位的PNG图像，才能实现透明的效果。因此导出时，一定要选择32位的PNG格式（24位的PNG格式不支持透明）。

（2）如果放置半透明PNG图像的div没有给出明确的高度或宽度中的一个，有可能不会显示该图像。

因此，读者在操作时，如果发现不显示Alpha透明PNG，就可以将上面的代码中增加下面这条样式，这样就可以正确显示了。

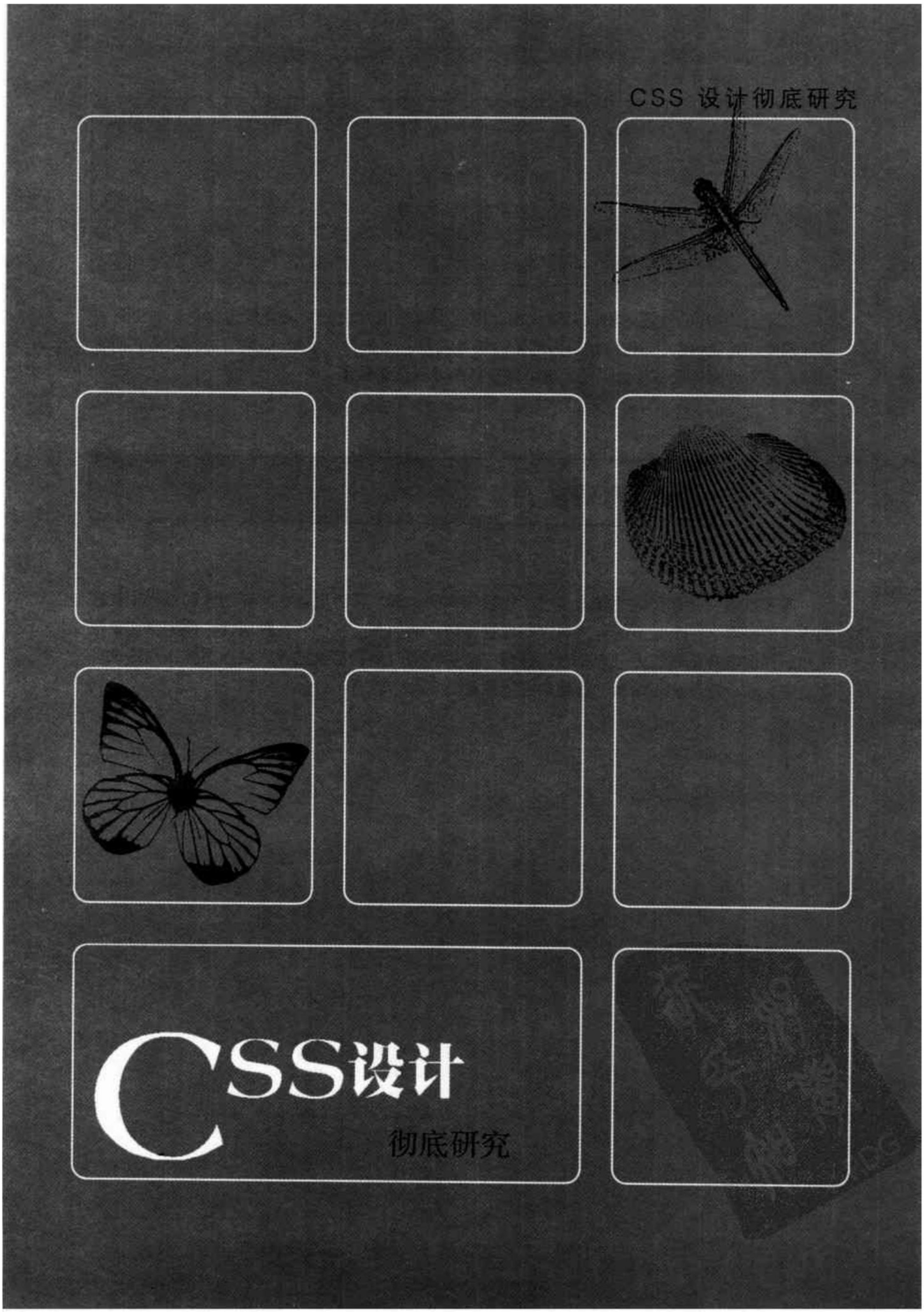
```
width: 1px;
```

以上两条结论是本书作者自己实验得出的结果，没有在相关资料中查到有关的说明。如果读者有进一步的结论，欢迎进一步讨论。

实际上，Alpha透明的用处是很大的。例如上面的网页中，页面背景为白色，如果将页面背景换成其他颜色，GIF格式阴影图像的效果就差，而如果使用Alpha透明的PNG格式阴影图像，无论页面背景是什么颜色，都可以很好地显示阴影效果。

5.3 本章小结

本章针对文字和图像的样式进行了比较深入的讲解。读者应该在实际制作的过程中把握两方面的要点。一个是不同样式属性的设置方法，例如关于字体的各种属性，读者都需要把这些属性的名称记熟；另一点是在设置每一个属性的时候，都要把样式放在整体布局的大环境之中，充分理解盒子模型、浮动和定位等重要布局原理。



第6章 链接与导航

在掌握了CSS布局的基本方法和原理之后，接下来的几章将针对页面中非常重要的部分“链接与导航”进行专门的介绍。本章对链接本身以及通常用于网页导航的HTML结构进行介绍。在后面的章节中再分专题分别讲解。

6.1 丰富的超链接特效

超链接是网页上最普通不过的元素，通过超链接能够实现页面的跳转、功能的激活等，因此超链接也是与用户打交道最多的元素之一。本节主要介绍超链接的各种效果，包括超链接的各种状态、伪类和按钮特效等。

6.1.1 动态超链接

在HTML语言中，超链接是通过标记<a>来实现的，链接的具体地址则是利用<a>标记的href属性，代码如下所示：

```
<a href="http://www.artech.cn">链接文本</a>
```

在默认的浏览器浏览方式下，超链接统一为蓝色并且有下划线，被点击过的超链接则为紫色并且也有下划线，如图6.1所示。

图6.1 普通的超链接

这种最基本的超链接样式现在已经无法满足广大设计师的需求。通过CSS可以设置超链接的各种属性，而且通过伪类别还可以制作很多动态效果。首先用最简单的方法去掉超链接的下划线，代码如下所示：

```
a{                                  /* 超链接的样式 */
  text-decoration:none;             /* 去掉下划线 */
}
```

此时的页面效果如图6.2所示。无论是超链接本身，还是点击过的超链接，下划线都被去掉了，除了颜色以外，与普通的文字没有多大区别。

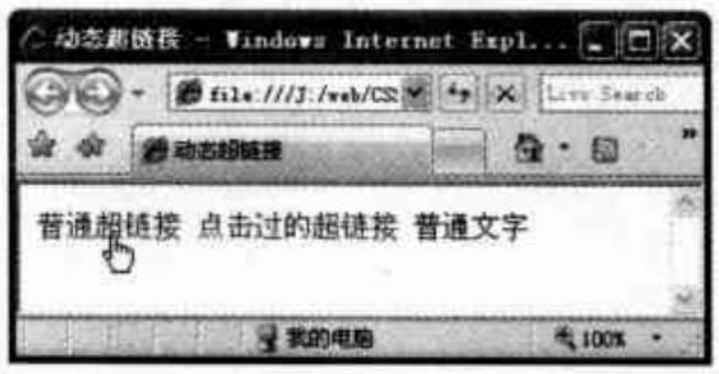


图6.2 没有下划线的超链接

仅仅通过设置标记<a>的样式来改变超链接，并没有太多动态的效果，下面来介绍利用CSS的伪类别（Anchor Pseudo Classes）来制作动态效果的方法，具体属性设置如表6.1所示。

表6.1 可制作动态效果的CSS伪类别属性

属性	说明
a:link	超链接的普通样式，即正常浏览状态的样式
a:visited	被点击过的超链接的样式
a:hover	鼠标指针经过超链接上时的样式
a:active	在超链接上单击时，即“当前激活”时，超链接的样式

注意 CSS就是通过以上4个伪类别，再配合各种属性风格制作出千变万化的动态超链接的。这里必须指出几点要特别注意的问题。

（1）对于这4个伪类别，要注意它们的声明顺序。例如，为了使平常状态超级链接取消下划线，而在鼠标指针经过时出现下划线，可以如下设置：

```
a:link, a:visited {text-decoration:none;}
a:hover, a:active {text-decoration:underline;}
```

但是如果交换上面两行代码的顺序，对鼠标指针经过时的设置将不会产生效果。这是由于前面讲过的“层叠”的原因，上面的这些CSS规则具有同样的特殊性，因此后面的规则将覆盖前面的规则，要按照上面的顺序来进行设置。

（2）a:link与a设置的都是链接在没有鼠标指针经过时的效果，那么它们是否完全等效呢？通常情况下二者是没有区别的，但是严格来说，在个别情况下二者有所区别，在一个a标记没有设置跳转地址（即href属性）时，如果设置a:link则无效，而设置a则仍然有效。在实际工作中，经常使用：

```
a, a:visited {text-decoration:none;}
```

而不用：

```
a:link, a:visited {text-decoration:none;}
```

（3）当前激活状态“a:active”一般被显示的情况非常少，因此很少使用。因为当浏览者单击一个超链接后，焦点很容易就会从这个链接上转移到其他地方，例如新打开的窗口等，此时该超链接就不再是“当前激活”状态了。因此，通常无需设置a:active的样式。

6.1.2 按钮式超链接

除了简单的文字颜色、下划线之外，对链接还可以设置各种属性，产生丰富多彩的效果。例如很多网页上的超链接都制作成各种按钮的效果，这些效果大都采用了各种图片。本节仅通过CSS的普通属性来模拟按钮的效果，如图6.3所示。

本案例文件位于本书光盘“第6章\01\button-style.htm”。

图6.3 按钮式超链接

① 首先建立最简单的菜单结构，本例直接采用<a>标记排列的形式，代码如下所示：

```
<body>
        <a href="#"> Home </a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#"> Map </a>
</body>
```

此时页面的效果如图6.4所示，仅是几个普通的超链接堆砌。

图6.4 普通超链接

② 然后对<a>标记进行整体控制，同时加入CSS的3个伪属性。对于普通超链接和点击过的超链接采用相同的样式，并且利用边框的样式模拟按钮效果。而对于鼠标指针经过时的超链接，相应地改变文字颜色、背景色、位置和边框，从而模拟出按钮“按下去”的特效，样式代码如下：

```
<style>
a{
                                          /* 统一设置所有样式 */
  font-family: Arial;
  font-size: .8em;
  text-align:center;
  margin:3px;
}
a:link, a:visited{                        /* 超链接正常状态、被访问过的样式 */
  color: #A62020;
  padding:4px 10px 4px 10px;
  background-color: #ecd8db;
  text-decoration: none;
  border-top: 1px solid #EEEEEE;          /* 边框实现阴影效果 */
  border-left: 1px solid #EEEEEE;
  border-bottom: 1px solid #717171;
  border-right: 1px solid #717171;
}
a:hover{                                  /* 鼠标经过时的超链接 */
```

```
    color:#821818;                    /* 改变文字颜色 */
    padding:5px 8px 3px 12px;         /* 改变文字位置 */
    background-color:#e2c4c9;         /* 改变背景色 */
    border-top: 1px solid #717171;    /* 边框变换，实现“按下去”的效果 */
    border-left: 1px solid #717171;
    border-bottom: 1px solid #EEEEEE;
    border-right: 1px solid #EEEEEE;
}
</style>
```

上面的样式代码中首先设置了<a>标记的整体样式，即超链接所有状态下通用的样式，然后通过对3个伪属性的颜色、背景色和边框的修改，模拟了按钮的特效。最终显示效果如图6.5所示。

图6.5　最终效果

6.1.3 CSS控制鼠标指针

在浏览网页时，通常看到的鼠标指针的形状有箭头、手形和I字形，而在Windows环境下实际看到的鼠标指针种类要比这个多得多。CSS弥补了HTML语言在这方面的不足，通过cursor属性可以设置各式各样的鼠标指针样式。

cursor属性可以在任何标记里使用，从而可以改变各种页面元素的鼠标指针效果，代码如下所示：

```
body{
  cursor:pointer;
}
```

pointer是一个很特殊的鼠标指针值，它表示将鼠标设置为被激活的状态，即鼠标指针经过超链接时，该浏览器默认的鼠标指针样式在Windows中通常显示为手的形状。如果在一个网页中添加了以上语句，页面中任何位置的鼠标指针都将呈现手的形状。除了pointer之外，cursor还有很多定制好了的鼠标指针效果，如表6.2所示。

表6.2　cursor定制的鼠标指针效果

属性值	指针说明	属性值	指针说明
auto	浏览器的默认值	nw-resize	
crosshair		se-resize	
default		s-resize	
e-resize		sw-resize	

续表

属性值	指针说明	属性值	指针说明
help		text	
move		wait	
ne-resize		w-resize	
n-resize		hand	
all-scroll		col-resize	
no-drop		not-allowed	
progress		row-resize	
vertical-text			

经验 表6.2中的鼠标指针样式，在不同的机器或者操作系统显示时可能存在差异，读者可以根据需要适当选用。很多时候，浏览器调用的是操作系统的鼠标指针效果，因此同一用户浏览器之间的差别很小，但不同操作系统的用户之间还是存在差异的。

6.1.4 浮雕背景超链接

除了背景颜色和边框等传统CSS样式外，如果将背景图片也加入到超链接的伪属性中，就可以制作出更多绚丽的效果。本例通过超链接背景图片的变换，实现浮雕的效果，如图6.6所示。注意本例需要用IE浏览器调试，不要使用Firefox浏览器，因为这里用到了Firefox不支持的属性。

本案例文件位于本书光盘“第6章\02\3d-style.htm”。

图6.6 浮雕式超链接

① 首先用<table>标记搭建整个HTML框架，加入Banner图片、页面背景颜色和超链接的排列等，并且为两个表格添加CSS类别，以便设置样式，代码如下所示。

```
<body>
<table cellpadding="0" cellspacing="0" class="links">
 <tr><td>
      <a href="#"> Home </a><a href="#"> Contact </a><a href="#">
 Web</a><a href="#">Design</a><a href="#"> Map </a>
```

```
    </td></tr>
  </table>
  </body>
```

注意 在输入上面的代码时，相邻的“<a>”和“”之间不要有空格，也不要换行，具体原因后面会讲解。

② 制作浮雕的背景（宽度可以是1px），作为超链接所在行的背景（本书光盘中提供），如图6.7所示。设置为水平方向重复，代码如下所示。

```
table {
  background:url(button1_bg.jpg) repeat-x;
  font-size:12px;
  width:100%
}
```

图6.7 浮雕背景

③ 再制作一个宽度固定的按钮图片（这里为80px，button1.jpg），与图6.7完全一样，但是最左边有一道白色的竖线，作为按钮的背景图片，并添加到统一设置的<a>属性样式中，代码如下所示。

```
a{
  width:80px;
  height:32px;
  padding-top:10px;
  text-decoration:none;
  text-align:center;
  background:url(button1.jpg) no-repeat;    /* 超链接背景图片 */
}
```

以上代码设置了超链接的高度和宽度等统一参数。此时页面的超链接部分如图6.8所示，可以看到作为按钮背景图片的button1.jpg，其左边的白竖线实现了按钮分割的效果。

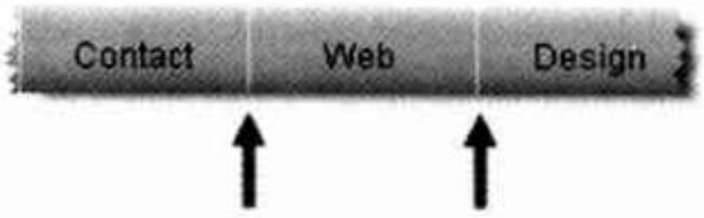


图6.8 添加按钮背景

④ 再用同样的方法制作一个宽度跟button1.jpg一样，但是背景为深色浮雕的图片button2.jpg，图片最左边同样有竖直的白线，将其作为鼠标指针经过超链接时的背景，如图6.9所示。然后给超链接添加CSS伪类别，修改相应的文字颜色，代码如下所示。

```
a{
  width:80px; height:32px;
  padding-top:10px;
  text-decoration:none;
  text-align:center;
}

a, a:visited{
  color:#654300;
  background:url(button1.jpg) no-repeat;    /* 变换背景图片 */
  }

a:hover{
  color:#FFFFFF;
  background:url(button2.jpg) no-repeat;    /* 变换背景图片 */
}
```

图6.9 变换背景图片

通过变换鼠标指针经过超链接时的背景图片，就实现了超链接浮雕的效果，如前面的图6.6所示。

读者还可以根据自己的需要，制作各式各样的背景图片，从而实现各种不同的效果。关于这两个背景的制作方法，请参考Fireworks或者Photoshop的书籍，很简单的。

注意 这里还需要补充说明两点注意事项。

（1）在本例中，各个链接之间是直接相邻的，在书写HTML的时候前一个和它后面的<a>之间不要有空格，也不要在和<a>之间换行，否则会出现如图6.10所示的现象。在鼠标指针经过时，它的后面和下一项之间有一个空白，如图6.10左图所示，而正确的效果则如图6.10右图所示。

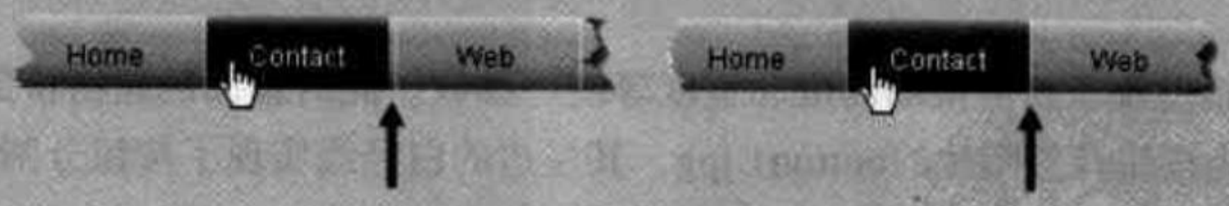


图6.10 和<a>之间不能有空白或换行

（2）在本例中所采用的方法在Firefox中的显示效果并不理想，如图6.11所示。原因在于Firefox不支持直接设置<a>标记的高度和宽度。

图6.11 Firefox中支持得不理想

如果希望能在所有的浏览器中都显示同样的浮雕效果，必须采用<ul>和<li>等项目列表的标记，这在下一节中将会详细介绍，本例的侧重点在于变换背景图片的设计思路。

6.1.5 让下划线动起来

这里是一个有趣的特殊下划线效果，如图6.12所示，在这个链接文字的下面是一条会移动的虚线下划线。普通状态时，虚线向左移动，鼠标指针经过时，虚线向右移动。

本案例文件位于本书光盘“第6章\03\ani-underline.htm”。

图6.12　动画下划线效果

这个案例中“会动”的下划线实际上并不是真的下划线，而是用背景图像来实现的。

① 首先准备两个动画GIF格式的图像，都是高2像素，宽4像素。第1个图像文件中有4帧，依次如图6.13所示。

图6.13　第1个动画GIF格式文件的4帧

当这些帧在动画GIF文件中循环显示的时候，就会出现跑马灯的移动效果，这个原理类似于平常在理发馆门口看到的转动圆筒。这个图像文件的制作方法这里就不介绍了，读者有兴趣可以参考Fireworks或者Photoshop的书籍，在本书光盘中有这个文件，读者可以直接使用。

② 第2个图像文件与第1个文件的内容相同，只是4帧的顺序不同，目的是为了实现相反的移动效果，如图6.14所示。

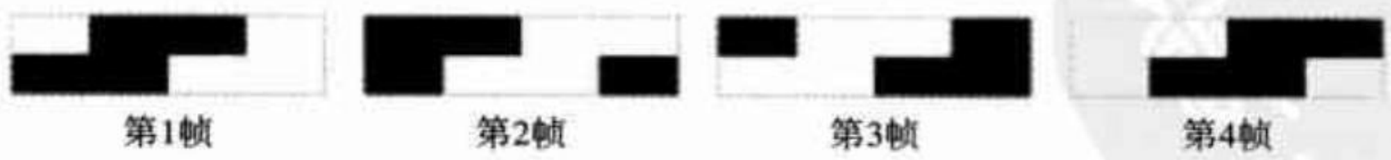


图6.14　第2个动画GIF格式文件的4帧

③ 接下来，页面的代码如下。

```
<html >
<head>
<title>下划线特效</title>
<style type="text/css">
a, a:visited{
```

```
  background: url(bg2.gif) repeat-x left bottom;
  text-decoration:none;
}
a:hover{
  background: url(bg.gif) repeat-x left bottom;
  text-decoration:none;
}
</style>
</head>
<body>
<a href="#">下划线特效</a> <a href="#">下划线特效</a>
</body>
</html>
```

可以从这段非常简单的代码中看出，实现这个效果的关键就是对链接的不同伪类设置不同移动方向的背景图片。

6.2 项目列表

传统的HTML语言提供了项目列表的基本功能，包括顺序式列表的<ol>标记和无顺序列表的<ul>标记等。当引入CSS后，项目列表被赋予了很多新的属性，甚至超越了它最初设计时的功能。本节主要围绕项目列表的基本CSS属性进行相关介绍，包括项目列表的编号、缩进和位置等。

6.2.1 列表的符号

通常的项目列表主要采用<ul>或者<ol>标记，然后配合<li>标记罗列各个项目，简单的列表代码如下，其显示效果如图6.15所示。

本案例文件位于本书光盘“第6章\04\list.htm”。

```
<html>
<head>
<title>项目列表</title>
<style>
ul{
  font-size:0.9em;
  color:#00458c;
  list-style-type:decimal;              /* 项目编号 */
}
</style>
</head>
<body>
```

```
<ul>
  <li>Home</li>
  <li>Contact us</li>
  <li>Web Dev</li>
  <li>Web Design</li>
  <li>Map</li>
</ul>
</body>
</html>
```

图6.15 普通项目列表

在CSS中项目列表的编号是通过属性list-style-type来修改的。无论是<ul>标记还是<ol>标记，都可以使用相同的属性值，而且效果是完全相同的。例如修改<ul>标记的样式为：

```
ul{
  font-size:0.9em;
  color:#00458c;
  list-style-type:decimal;                    /* 项目编号 */
}
```

此时项目列表将按照十进制编号显示，这本身是<ol>标记的功能。换句话说，在CSS中<ul>标记与<ol>标记的分界线并不明显，只要利用list-style-type属性，二者就可以通用，显示效果如图6.16所示。

图6.16 项目编号

当给<ul>或者<ol>标记设置list-style-type属性时，在它们中间的所有<li>标记都将采用该设置；如果对<li>标记单独设置list-style-type属性，则仅仅作用在该条项目上，如下所示。

```
<style>
ul{
  font-size:0.9em;
  color:#00458c;
  list-style-type:decimal;                    /* 项目编号 */
}
li.special{
```

```
    list-style-type:circle;                    /* 单独设置 */
}
</style>
</head>
<body>
<ul>
  <li>Home</li>
  <li>Contact us</li>
  <li class="special">Web Dev</li>
  <li>Web Design</li>
  <li>Map</li>
</ul>
</body>
```

此时的显示效果如图6.17所示，可以看到第3项的项目编号变成了空心圆，但是并没有影响其他编号。

图6.17 单独设置<li>标记

通常使用的list-style-type属性的值除了上面看到的十进制编号和空心圆以外还有很多，常用的如表6.3所示。

表6.3 list-style-type属性值及其显示效果

关键字	显示效果
disc	实心圆
circle	空心圆
square	正方形
decimal	1，2，3，4，5，6，…
upper-alpha	A，B，C，D，E，F，…
lower-alpha	a，b，c，d，e，f，…
upper-roman	Ⅰ，Ⅱ，Ⅲ，Ⅳ，Ⅴ，Ⅵ，Ⅶ，…
lower-roman	ⅰ，ⅱ，ⅲ，ⅳ，ⅴ，ⅵ，ⅶ，…
none	不显示任何符号

6.2.2 图片符号

除了传统的各种项目符号外，CSS还提供了属性list-style-image，可以将项目符号显示为任意的图片。例如有下面一段代码。

```
<html>
<head>
```

```
<title>项目列表</title>
<style>
ul{
  font-size:0.9em;
  color:#00458c;
  list-style-image: url(icon1.jpg);                    /* 项目符号 */
}
</style>
</head>
<body>
<ul>
  <li>Home</li>
  <li>Contact us</li>
  <li>Web Dev</li>
  <li>Web Design</li>
  <li>Map</li>
</ul>
</body>
</html>
```

在IE 7和Firefox中的显示效果如图6.18所示，每个项目的符号都显示成了一个小图标，即icon1.jpg。

图6.18　图片符号

如果仔细观察图片符号在两个浏览器中的显示效果，就会发现图标与文字之间的距离有着明显的区别，因此不推荐这种设置图片符号的方法。如果希望项目符号采用图片的方式，则建议将list-style-type属性的值设置为none，然后修改<li>标记的背景属性background来实现。例如下面这个例子。

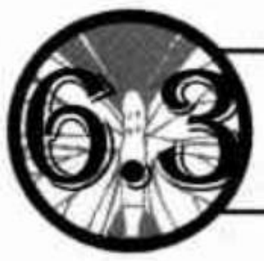
本案例文件位于本书光盘“第6章\04\icon-style.htm”。

```
<html>
<head>
<title>项目列表</title>
<style>
ul{
  font-size:0.9em;
  color:#00458c;
  list-style-type:none;                              /* 不显示项目符号 */
}
li{
  background:url(icon1.jpg) no-repeat;               /* 添加为背景图片 */
```

```
  padding-left:25px;                    /* 设置图标与文字的间隔 */
}
</style>
</head>
<body>
<ul>
  <li>Home</li>
  <li>Contact us</li>
  <li>Web Dev</li>
  <li>Web Design</li>
  <li>Map</li>
</ul>
</body>
</html>
```

这样通过隐藏<ul>标记中的项目列表，然后再设置<li>标记的样式，统一定制文字与图标之间的距离，就可以实现各个浏览器之间的效果一致，如图6.19所示。

图6.19 图片符号



6.3 简单的导航菜单

作为一个成功的网站，导航菜单是永远不可缺少的。导航菜单的风格往往也决定了整个网站的风格，因此很多设计者都会投入很多时间和精力来制作各式各样的导航条，从而体现网站的整体构架。

在传统方式下，制作导航菜单是很麻烦的工作。需要使用表格，设置复杂的属性，还需要使用JavaScript实现相应鼠标指针经过或点击动作。如果用CSS来制作导航菜单，实现起来就非常简单了。

6.3.1 简单的竖直排列菜单

当项目列表的list-style-type属性值为“none”时，制作各式各样的菜单和导航条便成了项目列表的最大用处之一，通过各种CSS属性变幻可以达到很多意想不到的导航效果。首先

看一个案例，其效果如图6.20所示。

本案例文件位于本书光盘“第5章\04\vertical.htm”。

图6.20 无需表格的菜单

① 首先建立HTML相关结构，将菜单的各个项用项目列表<ul>表示，同时设置页面的背景颜色，代码如下。

```
<body>
<div id="navigation">
  <ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">Contact us</a></li>
  <li><a href="#">Web Dev</a></li>
  <li><a href="#">Web Design</a></li>
  <li><a href="#">Map</a></li>
</ul>
</div>
</body>
```

② 然后开始设置CSS样式，首先把页面的背景色设置为浅色，代码如下。

```
body{
  background-color:# dee0ff;
}
```

此时页面的效果如图6.21所示，这只是最普通的项目列表。

图6.21 项目列表

③ 设置整个 块的宽度为固定150像素，并设置文字的字体。设置项目列表<ul>的属性，将项目符号设置为不显示。

```
#navigation {
  width:150px;
  font-family:Arial;
  font-size:14px;
```

```
  text-align:right
}
#navigation ul {
  list-style-type:none;                          /* 不显示项目符号 */
  margin:0px;
  padding:0px;
}
```

通过以上设置后，项目列表便显示为普通的超链接列表，如图6.22所示。

图6.22 超链接列表

④ 为<li>标记添加下边线，以分割各个超链接，并且对超链接<a>标记进行整体设置，如下所示。

```
#navigation li {
  border-bottom:1px solid #9F9FED;               /* 添加下边线 */
}
#navigation li a{
  display:block;
  height:1em;
  padding:5px 5px 5px 0.5em;
  text-decoration:none;
  border-left:12px solid #151571;                /* 左边的粗边 */
  border-right:1px solid #151571;                /* 右侧阴影 */
}
```

以上代码中需要特别说明的是“display:block;”语句，通过该语句，超链接被设置成了块元素。当鼠标指针进入该块的任何部分时都会被激活，而不是仅在文字上方时才被激活。此时的显示效果如图6.23所示。

图6.23 区块设置

⑤ 最后设置超链接的样式，以实现动态菜单的效果，代码如下。

```
#navigation li a:link, #navigation li a:visited{
  background-color:#1136c1;
  color:#FFFFFF;
}
#navigation li a:hover{                          /* 鼠标经过时 */
  background-color:#002099;                      /* 改变背景色 */
  color:#ffff00;                                 /* 改变文字颜色 */
  border-left:12px solid yellow;
}
```

代码的具体含义都在注释中一一说明了，这里不再重复。此时导航菜单便制作完成了，最终的效果如图6.24所示，在IE与Firefox两种浏览器中的显示效果是一致的。

图6.24 导航菜单

注意 此时在IE 6和Firefox中的显示效果是相同的，但是在IE 6中，虽然把链接设置成了块级元素，但是仍然只有在鼠标指针经过文字时，才能触发鼠标经过效果，而不是进入矩形区域就可以触发，在IE 7中已经修正了这个错误。在IE 6中，一个解决方法是在“#navigation li a”中增加一条设置高度的CSS规则：

```
height:1em;
```

这样可强制浏览器重新计算响应鼠标指针的范围，就可以得到正确的结果了。

6.3.2 横竖自由转换菜单

导航条不只是竖直排列，很多时候需要页面的菜单能够在水平方向显示。通过CSS属性的控制，可以轻松实现项目列表导航条的横竖转换。

这里在上面一个例子的基础上仅做两处改动，就能实现一个自由转换的菜单。图6.25显示的是浏览器窗口比较宽的时候，菜单的水平排列效果；图6.26左图显示的是浏览器窗口很窄的时候，菜单的竖直排列效果；图6.26右图显示的是浏览器窗口宽度不宽不窄的时候，菜单的折叠排列效果。

本案例文件位于本书光盘“第6章\04\horizontal.htm”。

图6.25 水平菜单

图6.26 水平菜单可以自由地转换为竖直菜单和折行菜单

这两处改动如下。

（1）把width:120这条CSS规则从“#navigation”移动到“#navigation li a”中。这样，这个列表就没有宽度限制了，同时保证每个列表项的宽度都是120像素。

（2）在“#navigation li”的样式中增加一条“float:left;”。也就是使各个列表项变为向左浮动，这样它们就会依次排列，直到浏览器窗口容纳不下，再折行排列。

通过这两处小小的改动，就可以实现从竖直排列的菜单到自由适应浏览器宽度的菜单的转换了。对于Firefox和IE浏览器都是适用的。

读者通过这个案例是不是深刻地感受到了CSS的强大和灵活。可以套用一句俗语“只有想不到，没有做不到”。

6.4 本章小结

在本章中，主要介绍了超链接文本的样式设计，以及对列表的样式设计。对于超级链接，最核心的是4种类别的含义和用法；对于列表，需要了解基本的设置方法。这二者都是非常重要和常用的元素。在后面的案例中，几乎每个案例都离不开它们，因此一定要把相关的基本要点掌握熟练，为后面制作复杂的例子打好基础。

第 7 章
竖直排列的导航菜单

导航是网页中不可缺少的功能，本章将制作5个风格各异的导航菜单。这些菜单都充分利用了CSS的各种特性，并且将前面介绍的各种关于盒子模型的技术要点都融入到各个案例当中。

本章将集中介绍竖直排列的菜单，第8章中将介绍水平排列的菜单，第9章中将介绍一个更为复杂的下拉菜单。需要注意的是，这些案例中都大量使用了前面章节中的CSS盒子模型以及浮动和定位的技术，因此读者务必先把前面的基础掌握好，再来学习这些实际的案例。这3章中的案例参考了英国设计师Stu Nicholls的设计和文章，如果读者有兴趣，可以访问他的网站http://www.cssplay.co.uk，了解更多内容。

7.1 双竖线菜单

本例制作一个简单的竖直排列的菜单效果，在每个菜单项的左右两边各有一条竖线，当鼠标指针滑过时，竖线由灰色变为黑色，同时菜单文字变为红色，效果如图7.1所示。
该实例文件位于本书光盘的“第7章\01\vertical-border-1.htm”。

图7.1 双竖线菜单效果

7.1.1 HTML框架

首先，从编写基本的HTML文件开始，搭建出这个菜单的基本框架，HTML代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<head>
    <title>双竖线菜单</title>
</head>
<body>
    <div id="menu">
        <a href="#"> Home </a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#"> Map </a>
    </div>
</body>
</html>
```

可以看到，body部分非常简单，5个文字链接被放置到一个id设置为menu的div容器中。此时在浏览器中观察效果，只是最普通的文字超链接样式，如图7.2所示。

注意 由于这个div包括了所有的链接，也就是各个菜单项，因此这里将这个div称为“容器”。

图7.2 没有任何CSS设置时的效果

7.1.2 设置容器的CSS样式

现在设置菜单div容器的整体区域样式，设置菜单的宽度、背景色，以及文字的字体和大小。在HTML文件的head部分增加CSS样式表代码如下。

```
<head>
    <title>双竖线菜单</title>
          <style type="text/css">
          #menu {                                  /*设置#menu的整体区域*/
            width:120px;                           /*宽度*/
            background:#ccc;                       /*背景色为灰色*/
            font-family:Arial;                     /*文字字体*/
            font-size:14px;                        /*文字大小*/
            }
    </style>
</head>
```

这时效果如图7.3所示。可以看到，文字链接都被限制在了#menu容器中。

图7.3 设置了#menu容器后的效果

然后对菜单进行定位，在#menu部分增加如下两行代码。

```
padding:8px;                                       /*设置内边距*/
margin:0 auto;                                     /*设置水平居中*/
```

这时这个菜单在浏览器窗口中就水平居中显示了，并且文字和边界之间空8个像素的距离，如图7.4所示。

图7.4 设置内外边距后的效果

7.1.3 设置菜单项的CSS样式

现在就需要设置文字链接了。为了使5个文字链接依次竖直排列，需要将它们从“行内元素”变为“块级元素”。此外还应该为它们设置背景色和内边距，以使菜单文字之间不要过于局促。具体代码如下：

```
#menu a, #menu a:visited {      /*设置菜单项样式*/
    display:block;              /*设置为块级元素*/
    background-color:#fff;      /*背景色为白色*/
    padding:4px 8px;            /*内边距上下4像素，左右8像素*/
    }
```

效果示意如图7.5所示，斜线部分就是padding属性设置的内边距。

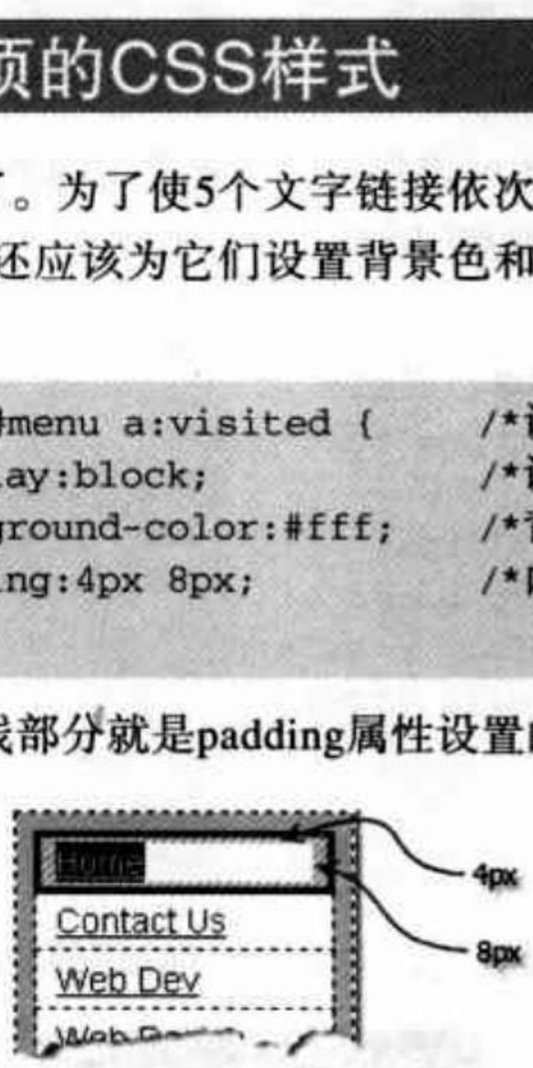


图7.5 内边距示意图

接下来设置文字的样式，取消下划线，并将文字设置为黑色，代码如下：

```
color:#000;
text-decoration:none;
```

现在参考一下前面的效果图，可以发现每个菜单项之间应该有一定的空隙，而不是现在这样连在一起。这个效果可以通过设置margin来实现，代码为：

```
margin:8px 0;
```

此外，左右两侧还各需要一条竖线，这可以通过设置左右边框来实现，代码为：

```
border-left:8px solid #9ab;
border-right:8px solid #9ab;
```

此时的效果如图7.6所示。

图7.6 在Firefox浏览器中的效果

将图7.6和最前面的目标效果图7.1比较，虽然总体上和目标效果已经非常接近了，但是图7.6中灰色背景的上下空白的宽度比左右空白的宽度要宽，而目标效果中，这个灰色的边上下左右都是一样宽的。

下面来具体分析一下目前的情况。如图7.7所示，细斜线区域是padding，粗斜线区域是margin。从这里可以看到一个现象，竖直相邻的两个菜单项之间的margin合并了，即本来每一个菜单都有自己的8个像素高的margin，由于竖直方向的两个margin相遇了，因此它们之间的margin就以高的那个margin为准，这里都是8个像素，因此最终的结果就是8个像素。

图7.7 内外边距示意图

然而，由于最上面的菜单项的margin和menu容器的padding各有8个像素，加在一起就是16个像素高，从而使得上下灰边比左右的灰边要宽一些。下面我们就来解决这个问题。

7.1.4 解决出现的问题

既然问题产生的原因在于上下两端空白的高度等于容器顶部的padding加上第一个菜单项上侧的margin，就会自然想到要如何将这个多余的margin去掉。如果修改链接文本的样式，那么所有的菜单项都会受到影响，因此必须把第一个和最后一个菜单项的样式与其他菜单项的样式区别对待，才能解决问题。具体的方法如下。

① 首先将HTML文件中相关的两个链接的代码修改如下：

```
<body>
    <div id="menu">
        <a href="#" id="first"> Home</a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#" id="last"> Map</a>
```

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

② 在第一个和最后一个链接中，分别增加一个样式类别，设为#first和#last，这里表示第一个和最后一个的意思。然后设置这两个id的样式，将它们的margin设置为0。

```
#menu a#first, #menu a#last {
  margin:0;
}
```

注意 由于中间的3个菜单项都设有margin，因此即使上下两个菜单项的margin为0，也不会和其他菜单项合并到一起。

此时的效果如图7.8所示。

图7.8 修改后的效果

③ 最后，设置鼠标指针经过效果，代码如下：

```
#menu a:hover{
  color:#f00;
  border-left:8px solid #000;
  border-right:8px solid #000;
}
```

此时在IE浏览器和Firefox浏览器中的效果如图7.9所示。

图7.9 在IE和Firefox中的不同效果

可以看到，在Firefox中的显示效果完全正确，只要鼠标指针进入菜单项的矩形就会触发

鼠标经过效果；而在IE浏览器中，只有当鼠标指针移动到文字上的时候才会触发鼠标经过效果，而右图中鼠标指针进入矩形范围时，并没有触发效果，这是IE本身的问题导致的。

解决这个问题的办法是，在“#menu a, #menu a:visited”的样式中增加下面这条CSS规则：

```
height:1em;
```

这样不会改变菜单的外观，但可以解决上面所说的问题。至此，这个案例就全部完成了。为了方便读者分析，代码整理后抄录如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<head>
    <title>双竖线菜单</title>
          <style type="text/css">
          #menu {
            width:120px;
            background:#ccc;
            font-family:Arial;
            font-size:14px;
            padding:8px;
            margin:0 auto;
            }
        #menu a, #menu a:visited {
            display:block;
            background-color:#fff;
            padding:4px 8px;
            color:#000;
            text-decoration:none;
            margin:8px 0;
            border-left:8px solid #9ab;
            border-right:8px solid #9ab;
            height:1em;
                }
          #menu a#first, #menu a#last {
            margin:0;
          }
          #menu a:hover{
            color:#f00;
            border-left:8px solid #000;
            border-right:8px solid #000;
          }
          </style>
</head>
<body>
    <div id="menu">
        <a href="#" id="first"> Home</a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#" id="last"> Map</a>
    </div>
```

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

在这个案例中，一个重要的问题是如何在一系列相同的菜单项中，对两端的元素作特殊处理。上面使用的方法是为它们单独设置CSS类别。实际上，可以进一步考虑，其实只需要对一端的元素作特殊处理就可以了，而不必对上下两端的元素都作特殊处理。例如，可以使第一个菜单项的4个margin都为0，然后将其他4个菜单项的margin设置为“margin: 8px 0 0 0”，即只有上侧的margin为8像素，其他3个margin为0，这样只需要对第一个菜单项设置class="first"就可以了，效果是完全相同的。

事实上，W3C组织在设计CSS规范的时候，就已经替用户考虑到这个问题了，他们专门为此设计一个伪类first-child。例如，找到下面这段代码：

```
#menu a#first, #menu a#last {
  margin:0;
}
```

修改为：

```
#menu a:first-child{
  margin:0;
}
```

这里的选择器的含义就是，对在#menu这个div中的“第一个”a标记进行样式设置，这样就不用在HTML代码中单独为第一个菜单项设置特殊的类了。

注意 first-child伪类虽然是标准的CSS伪类，但是IE浏览器并不支持它。由于IE对margin和padding的处理与Firefox不同，因此在本例中使用first-child伪类的方法，同样可以满足IE和Firefox中都正确显示的要求。但是在其他情况下，如果为了保证在IE浏览器中正确显示，目前还应该使用自定义的class代替first-child伪类的方法。

请读者参考本书光盘中的文件，“vertical-border-1.htm”中使用的是对首尾两个菜单项特殊处理的方法，“vertical-border-2.htm”中使用的是仅对首菜单项特殊处理的方法，“vertical-border-3.htm”中使用的是first-child伪类的方法。如果读者还有不明白的地方，请分别参考这些文件。

7.2 双斜角横线菜单

在这个案例中，将制作如图7.10所示的菜单。鼠标指针经过某个菜单项时，显示灰色背景色，同时在上下两侧各产生一条带有斜角的横线。

该实例文件位于本书光盘的“第7章\02\bevel-navi.htm”。

图7.10 双斜角横线菜单效果

在这个案例中，需要重点考虑的是这个带有斜角的横线是如何产生的。

7.2.1 基本设置

本案例和上一个案例相同的HTML结构如下。

```
<body>
    <div id="menu">
        <a href="#"> Home</a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#"> Map</a>
    </div>
</body>
```

对于#menu容器的设置如下。

```
#menu {                                    /*对menu层设置*/
    width:9em;                             /*宽度*/
    margin:0 auto;                         /*水平居中*/
    font-family:Arial;                     /*字体*/
    font-size:14px;                        /*字号*/
    border:solid 1px #aaa;                 /*细灰色边框*/
    }
```

在对#menu的设置中，宽度使用的是相对单位em，而不像上一个例子，使用像素作为单位，代码如下。

```
#menu a, #menu a:visited {                 /*设置菜单选项*/
  display:block;                           /*设置为块级元素*/
  text-decoration:none;                    /*无下划线*/
  color:#000;                              /*黑色文字*/
  line-height:1.4em;                       /*高度*/
  border:0.5em solid #fff;                 /*白色背景，防止跳动*/
  }

#menu a:hover {
  color:#fff;                              /*白色文字*/
```

```
        background-color:#aaa;                    /*深灰色背景色*/
        border-color:#ddd #aaa;                   /*上下边框浅灰色，左右与背景色相
同*/
        }
```

使用相对单位的优势是，当访问者在浏览器中调整了文字的大小，#menu容器的大小可以自动调整，以适应文字大小的变化，如图7.11所示。

图7.11 从左至右依次为缩小文字、正常文字和放大文字情况下的显示效果

此外，还为#menu容器设置了1个像素宽的灰色的边框，代码为：

```
border:solid 1px #aaa;                    /*细灰色边框*/
```

7.2.2 菜单项设置

菜单项没有鼠标指针经过时效果的设置方法与7.1节的案例相似，代码如下。

```
#menu a, #menu a:visited {                /*设置菜单选项*/
  display:block;                          /*设置为块级元素*/
  text-decoration:none;                   /*无下划线*/
  color:#000;                             /*黑色文字*/
  line-height:1.4em;                      /*高度*/
  border:0.5em solid #fff;                /*与背景色相同*/
  }
```

与上个案例的不同之处在于，这里为了使菜单项的文字之间有一定空白，并且使文字在菜单项中竖直居中，没有采用padding来实现，而是用了line-height属性。当line-height属性设置为1.4em时，文字的上下会各产生0.2em的空白。

注意 这里不能使用height属性。如果使用height属性代替line-height属性，产生的效果就如图7.12所示。可以看到，文字将在竖直方向靠上对齐，而不是竖直居中了，这不是所期望的效果。

图7.12 菜单项文字靠上对齐

接下来，就要实现在鼠标指针经过时产生的背景色和边框了。在前面正常状态下的CSS代码中有如下一行设置。

```
        border:0.5em solid #fff;      /*与背景色相同*/
```

对鼠标指针经过效果设置如下。

```
      #menu a:hover {
        color:#fff;                   /*白色文字*/
        background-color:#aaa;        /*深灰色背景色*/
        border-color:#ddd #aaa;       /*上下边框浅灰色，左右与背景色相同*/
        }
```

可以看出，这个边框在鼠标指针未经过之前就已经存在了，只是和背景色相同，所以看不出是边框。而当鼠标指针经过时，边框的颜色发生改变，边框就显现出来了。

思考 最后请读者思考一下，为什么这里显示出的上下两条边框是斜角的，而7.1节案例中的左右两条边框不是这样的呢?

答案是，7.1节的案例中设置了左右两条边框宽为4px，上下边框为0；而在这个案例中，虽然左右边框在鼠标指针经过和不经过时的颜色不同，但实际上它们是存在宽度的，因此在水平和竖直的边框接角的位置，就产生了斜角。

7.3 立体菜单

在本案例中，将实现如图7.13所示的立体菜单效果。当鼠标指针经过菜单项时，菜单按钮将显示出被按下的样式。

该实例文件位于本书光盘的“第7章\03\3d-navi.htm”。

图7.13 立体菜单效果

7.3.1 基本设置

本案例仍然使用和上一个案例相同的HTML结构，基本代码HTML如下。

```
<body>
    <div id="menu">
        <a href="#"> Home</a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#"> Map</a>
    </div>
</body>
```

对于#menu容器的设置如下。

```
#menu {                                    /*对menu层设置*/
    font-family:Arial;                     /*字体*/
    font-size:14px;                        /*字号*/
    }
```

在对#menu的设置中，设置了基本的字体和字号，其他都没有做设置，留待菜单项中进行相关的设置。

7.3.2 菜单项设置

菜单项的普通状态的设置方法与前一个案例相似，代码如下。

```
#menu a, #menu a:visited {
    text-decoration:none;                  /*文字无下划线*/
    text-align:center;                     /*文字水平居中对齐*/
    color:#fff;                            /*白色文字*/

    display:block;                         /*设置为块级元素*/
    width:10em;                            /*宽度*/
    padding:0.25em;                        /*内边距*/
    margin:0.5em auto;                     /*菜单项之间间隔0.5em，并水平居
中*/
    background-color:#8ab;                 /*背景色*/

    border:2px solid #fff;                 /*边框粗细2像素*/
    border-color:#def #678 #345 #cde;      /*边框颜色显示突起效果*/

    position:relative;                     /*使用相对定位*/
    }
```

前3行样式用于设置文字，接下来的5行用于设置菜单项的盒子模型参数，再接下来的两

行设置边框，2个像素宽。需要特别注意的是4条边框颜色的设置。在实现凸起的效果时，通常都认为光线从左上方照射过来，因此上边框和左边框使用浅色，下边框和右边框使用深色，就会产生凸起的视觉效果。

现在实现鼠标指针经过时的按钮凹陷样式。首先为了产生按钮被按下的效果，可以使按钮向右下方平移2个像素，这样在按钮被按下的一瞬间产生的连续动作可以明显地体现被按下的视觉效果。

为了能够使按钮从原有位置平移，需要将按钮所在的盒子设置为相对定位，也就是上面代码中的最后一行。

```
position:relative;                                    /*使用相对定位*/
```

注意　读者需要特别记住这一点，所谓相对定位，是指以定位元素自身所在的位置为基准，平移指定的距离。

到这里，在浏览器中的显示效果如图7.14所示。

图7.14　实现凸起效果

接下来，在鼠标指针经过的样式设置中，确定平移的方向和距离，代码如下。

```
#menu a:hover {
  top:2px;                                            /*向下移动2像素*/
  left:2px;                                           /*向右移动2像素*/
  border-color:#345 #cde #def #678;                   /*边框颜色显示凹陷效果*/
  }
```

“top:2px”的含义是，距离上界2个像素。由于原来鼠标指针未经过时，距离上界是0像素，因此产生的效果就是向下移动了2个像素。依此类推，“left:2px”的效果是向右移动2个像素。由于前面已经设定定位方式为相对定位，因此这里的移动都是以自身原来的位置为基准进行的。

为了实现凹陷的视觉效果，需要将上下边框的颜色交换，左右边框的颜色交换。这样，这个立体菜单的效果就完成了。

这个案例在IE浏览器中的显示效果和在Firefox中完全相同，如图7.15所示。

图7.15 在IE中的效果

7.4 箭头菜单

本例的目标是实现带有两侧箭头的菜单效果，如图7.16所示。这里的箭头效果没有使用任何背景图像文件，而是完全依靠CSS代码实现的。

图7.16 箭头菜单效果

该实例文件位于本书光盘的“第7章\04\arrow-navi.htm”。

7.4.1 基本思路

既然这里不允许使用背景图像来制作这个三角形的样式，那么如何产生这两个三角形，并放到适当的位置上呢？

读者可以回忆一下7.2节的案例双斜角横线菜单的制作方法。那个案例中通过相邻边框的颜色区分实现了斜角效果。在这里，就继续挖掘它的潜力。

1. 三角形效果

如果将一个CSS盒子的height和width设置为0，然后将它的边框设置得比较粗，并且使左或右边框的颜色不同于背景色，而其余3条边框的颜色和背景色相同，就可以产生所需的

三角形效果了。设置方法如图7.17所示。

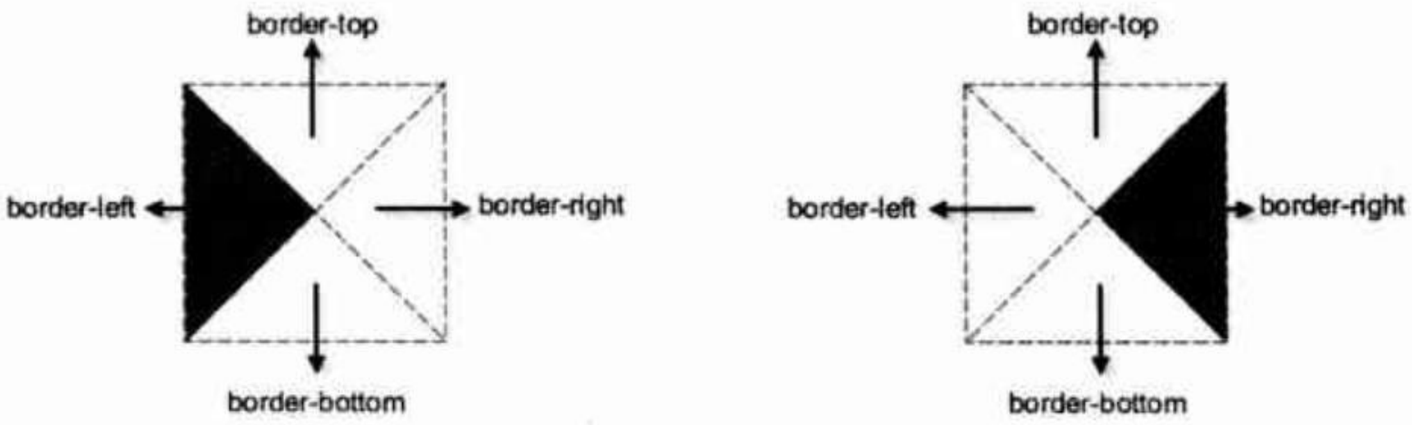


图7.17　使用CSS来制作箭头的方法

2. 放置三角形

在获得了三角形以后，接下来的任务是如何将这个三角形放到菜单项的两端。首先读者可能会想到，是否可以增加一个盒子，并使这个盒子的大小和菜单项大小匹配。当鼠标指针经过某个菜单项的时候，使这个盒子显示出左右两个三角形（本质上是左右两条边框）；当鼠标指针没有经过的时候，4条边框均设置为背景色。这样是否可行呢？这样做会带来一个问题，即这个盒子会很宽，从而遮盖住菜单项。因此只能把这个思路变为：增加两个小盒子，分别放置在菜单项的两端，各用于显示一个三角形。

3. 制作准备

分析到这里，已经可以开始动手制作了。仍然使用前面例子的HTML代码，但是必须对它进行一定的改造，也就是为每个菜单项增加两个盒子来放置三角形。具体代码如下：

```
<body>
    <div id="menu">
        <a href="#">
            <span class="left"></span>
                Home
                <span class="right"></span>
        </a>
        <a href="#">
            <span class="left"></span>
              Contact Us
                <span  class="right"></span>
        </a>
        <a href="#">
                <span class="left"></span>
                  Web Dev
                <span class="right"></span>
        </a>
        <a href="#">
                <span class="left"></span>
                  Web Design
                <span class="right"></span>
        </a>
        <a href="#">
                <span class="lef"></span>
                  Map
```

```
                    <span class="right"></span>
                </a>
            </div>
        </body>
```

可以看到，在每个<a>和标记之间，链接文字的前后各增加了一对和标记，同时分别设置了CSS类别，即left和right。注意它们内部本身是空白的，这样就可以通过CSS的样式把这些span显示为三角形了。

说明 （1）有的参考资料上称这种方法为“钩子”，意思是它像钩子一样可以挂接CSS样式，很形象地说明了这种方法的本质。

（2）CSS的初衷是希望网页的内容和形式完全分离，而这种方法实际上在HTML中增加了没有内容含义的标记，因此并不是完全符合CSS的思想。但是制作网页毕竟最终是用于实际的，因此这里也没有必要过于追求理论上的纯粹性。但是读者应该知道，CSS的根本思想是尽可能使网页的结构与表现形式分离。

7.4.2 基本设置

下面首先设置#menu容器，代码如下。

```
#menu {                                    /*对menu层设置*/
  font-family:Arial;                       /*字体*/
  font-size:16px;                          /*字号*/
  width:140px;                             /*宽度*/
  margin:0 auto;                           /*水平居中*/
  border:solid 1px #ccc;                   /*灰色细边框*/
  }
```

接着设置菜单项普通状态的样式，代码如下。

```
#menu a, #menu a:visited {
  text-decoration:none;                    /*文字无下划线*/
  text-align:center;                       /*文字水平居中对齐*/
  color:#c00;                              /*红色文字*/
  display:block;                           /*设置为块级元素*/
  padding:4px;                             /*内边距*/
  background-color:#fff;                   /*背景色*/
  border:solid 1px #fff;                   /*与背景色相同边框，防止跳动*/
  position:relative;                       /*使用相对定位*/
  }
```

注意 这里需要注意两点。

（1）倒数第2行样式的设置，为每一个菜单项设置了边框，而边框的颜色与背景色相同。这是由于将来在鼠标指针经过时会出现红色的边框，因此为了前后不产生跳动，

这里加上一个与红色边框相同粗细的边框。虽然看不到它，但是可以防止鼠标指针经过时产生跳动。

（2）最后一行样式的设置，将菜单项的CSS盒子设为了相对定位。这实际上并不是要改变菜单项本身的位置，而是要通过这个设置使菜单项的CSS盒子成为一个“包含块”，从而能够使它下属的CSS盒子以它为基准进行定位。在下面的讲解中就可以看到它的作用了。

制作到这里，在浏览器中的显示效果如图7.18所示。

图7.18　鼠标指针未经过时的效果

7.4.3 设置箭头效果

首先，为了产生鼠标指针经过时的红色边框，进行如下设置。

```
#menu a:hover {
  border-color:#c00;                          /*边框颜色设为红色*/
  }
```

这时在浏览器中的效果如图7.19所示。

图7.19　响应鼠标指针经过的效果

接着，需要考虑承担“钩子”任务的span标记了。在鼠标指针没有经过的普通状态时，它就像不存在一样，因此也不用去设置它。而当鼠标指针经过时，就需要对它进行设置了。首先设置左右两个span共同存在的属性，代码如下。

```
#menu a:hover span {
  display:block;                              /*设置为块级元素*/
```

```
position:absolute;                    /*使用绝对定位*/
height:0;                             /*高度为0*/
width:0;                              /*宽度为0*/
overflow:hidden;                      /*防止溢出*/
border:solid 8px #fff;                /*设置默认的边框样式*/
top:4px;                              /*竖直方向的定位*/
}
```

这里依次进行了如下的设置工作。

① 首先需要注意选择器的使用。“#menu a:hover span”表示id为#menu的容器中的a标记在鼠标指针经过时，a标记内部的span元素，这正是需要选择的元素。

② 然后，将它设置为块级元素，因为span在默认情况下是行内元素。

③ 设置为绝对定位方式，这就用到了前面所做的准备工作，将菜单项的a元素设置为相对定位，成为这里的span元素的定位基准。

④ 再将高度和宽度都设置为0，这样才能产生三角形的箭头形状。

⑤ 确保溢出部分隐藏起来，这个最后再进行补充说明。

⑥ 设置边框粗细为8像素。这是因为最前面把菜单的文字大小设置为了16像素，因此这里边框宽度设置为8像素，就会产生高度为16像素的三角形，正好和文字高度匹配。

⑦ 边框颜色与背景色相同，这样使得在默认情况下，上下左右边框都看不出边框的存在。

⑧ 进行竖直方向的定位，设置为距离上边界4像素，由于前面设置的菜单项的padding为4像素，因此这里也设置为4像素。

上面设置的是文字左右的两个span元素的共同属性，接下来需要设置各自不同的属性，代码如下。

```
#menu a:hover span.left {
border-left-color:#c00;
left:8px;
}

#menu a:hover span.right {
border-right-color:#c00;
right:8px;
}
```

经验 为了便于调试，可以先将上面样式中的“:hover”去掉，这时在浏览器中的效果如图7.20所示。

由于把“:hover”暂时去掉了，因此每个span都显示出来了。如果发现形状或位置不正确，可以分析哪里错了，等调整正确之后，再把3处“:hover”添加回去即可。

图7.20 调试时的效果

上面这两段样式代码分别用于设置左边的箭头和右边的箭头。它们的各种属性都和“#menu a:hover span”中设置的相同，除了下面列出的两个。

（1）“border-left-color:#c00;”和“border-right-color:#c00;”分别用于将左边框和右边框的颜色设置为红色。

（2）“left:8px;”的含义是距离左边界8像素，同理“right:8px;”的含义是距离右边界8像素。

这时在Firefox和IE浏览器中，显示效果如图7.21所示。

图7.21　在Firefox和IE中的不同效果

可以看到，Firefox显示的效果正如期望的那样，而在IE中的显示效果还有问题，左边的箭头位置不正确。这是IE对a元素的宽度解释不同造成的，它不是按照自动伸展来计算的，而是根据文字的位置计算，因此需要增加一行代码来解决这个问题。

方法是，在“#menu a:hover span {”这段样式中，增加一行：

```
width:130px;
```

即明确地给出每个菜单项的宽度。这个130像素是如何计算出来的呢？外面的#menu容器宽度为140像素，这里希望它里面的a元素的宽度加上边框和padding的总宽度正好等于140像素。由于已经设定边框为1像素，padding为4像素，因此a元素的宽度应该设置为140-2×1-2×4=130像素。

最后，再来解释一下关于溢出样式的设定。前面已经看到，在对“#menu a:hover span”进行设置的时候，有如下代码。

```
overflow:hidden;                    /*防止溢出*/
```

现在如果把这行代码删除，那么测试效果如图7.22所示。

图7.22　在Firefox和IE中的不同效果

可以看到，Firefox的显示效果依然正确，但是在IE中就不正确了，这是因为即使span中没有任何内容，IE也会认为有默认的行高。如果加入这一行CSS代码，就可以使IE也能正确实现期望的效果了。

7.5 带说明信息的菜单

现在在7.4节案例的基础上，制作一个能够显示说明信息的菜单，如图7.23所示。鼠标指针经过某一个菜单项的时候，会出现相应的说明信息。

该实例文件位于本书光盘的“第7章\05\tip-navi.htm”。

图7.23 带说明信息的菜单效果

在通常的状态下，这个菜单和7.4节中制作的菜单完全相同。而在鼠标指针经过菜单项的时候，在菜单右侧会出现一个矩形区域，里面分别写着对正在经过的菜单项的说明文字，这是一个很实用的效果。

7.5.1 基本思路

如果对7.4节的内容理解了，本案例的解决方法就很容易能想到。仍然是使用“钩子”的方法实现。

以7.4节案例的代码为基础，对于每一个菜单项的a元素，分别再增加一个span元素，里面输入相应的说明文字，代码如下（这里指摘录了前两个菜单项的内容，其他3个菜单项的实现是完全相同的）。

```
<div id="menu">
<a href="#">
    <span class="left"></span>
        Home
        <span class="right"></span>
        <span class="intro">这里说明Home菜单项</span>
    </a>
<a href="#">
```

```
                <span class="left"></span>
                    Contact Us
                        <span class="right"></span>
                        <span class="intro">这里说明Contact Us菜单项</span>
            </a>
        ……其余菜单项类似……
```

接下来的基本思路是，首先在默认状态下，把这些说明信息隐藏起来，当鼠标指针经过某一个菜单项的时候，再打开该span就可以了。

7.5.2 设置方法

由于本例是在7.4节案例的基础上完成的，因此对其修改的内容并不多，这里就把CSS样式代码全部抄录，然后选择修改的位置进行讲解。如果读者对7.4节的案例还有不清楚的地方，也可以参考这里完整的代码。

```
        <style>
        #menu {                                  /*对menu层设置*/
            font-family:Arial;                   /*字体*/
            font-size:16px;                      /*字号*/
            width:140px;                         /*宽度*/
            margin:0;                            /*菜单项之间间隔0.5em，并水平居
中*/
            border:solid 1px #ccc;               /*灰色细边框*/
            }
        #menu a, #menu a:visited {
          text-decoration:none;                  /*文字无下划线*/
          text-align:center;                     /*文字水平居中对齐*/
          color:#c00;                            /*红色文字*/
          display:block;                         /*设置为块级元素*/
          padding:4px;                           /*内边距*/
          background-color:#fff;                 /*背景色*/
          border:solid 1px #fff;                 /*与背景色相同边框，防止跳动*/
          position:relative;                     /*使用相对定位*/
          width:130px;
          }
        #menu a span {
          display:none;
          }
        #menu a:hover {
          border-color:#c00;                     /*边框颜色红色*/
          }
        #menu a:hover span {
          display:block;                         /*设置为块级元素*/
          position:absolute;                     /*使用绝对定位*/
          height:0;                              /*高度为0*/
          width:0;                               /*宽度为0*/
          overflow:hidden;                       /*防止溢出*/
          border:solid 8px #fff;                 /*设置默认的边框样式*/
          top:4px;                               /*竖直方向的定位*/
```

```
}
#menu a:hover span.left {
border-left-color:#c00;
left:8px;
}
#menu a:hover span.right {
border-right-color:#c00;
right:8px;
}
#menu a:hover span.intro {
  color:#000;
  font-size:12px;
  display:block;
  position:absolute;                    /*绝对定位*/
  left:150px;
  top:0px;
  width:100px;
  height:auto;
  padding:5px;
  background-color:#eee;
  border:1px dashed #234;
  }
</style>
```

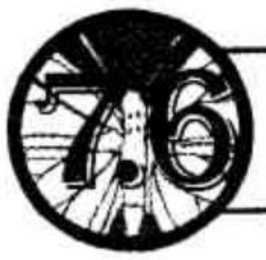
首先观察第3段代码，也就是为“#menu a span”设置的CSS样式，它的作用是在普通状态下，将所有span元素都隐藏起来，使用的CSS语句是：

```
display:none;
```

接着观察最后一段代码，它的作用是设定说明信息span的具体样式。其中的设置方法，包括定位的方法，都与上一节中的案例相同。使用绝对定位，以a元素为定位基准。

需要注意的是关于height的设置。这里将height的值设置为auto，如果没有这一行，在Firefox中的效果将如图7.24左图所示，因此这里设为auto，以使它的高度适应文字的需要，这时效果就如图7.24右图所示了。

图7.24 在Firefox中的效果



7.6 本章小结

在这一章里，制作了5个竖直方向的导航菜单。需要读者特别注意的有以下几个CSS基础知识和技巧。

（1）相邻盒子的margin在竖直方向相遇时的“塌陷”现象，水平方向则没有该现象。

（2）在一系列盒子排列后，如何对头尾盒子进行单独的样式设置，可以为它们单独设置类别，或者是伪类，但需要注意浏览器对伪类的支持情况。

（3）通过粗边框的颜色设置可以产生出斜角效果，如果充分利用可以产生有趣的效果。

（4）为了防止鼠标指针经过时造成边框的跳动，可以在通常状态添加与背景色相同的边框。

（5）相对定位的含义是以自身原来所在位置为基准进行偏移。

（6）“包含块”的定义方法，绝对定位是以包含块为基准进行偏移的。

（7）为了使某个盒子绝对定位，它的上一级元素可以设为相对定位，从而变成“包含块”。

CSS 设计彻底研究

CSS设计

彻底研究

第8章 水平导航菜单

第7章中，介绍了5个竖直方向菜单的制作方法，相信读者可以体会到CSS的功能非常强大。在本章中，我们将制作6个水平排列的菜单。它们也同样各具特色，而且应用了一些CSS设计中非常重要的技术，例如“滑动门”技术。它不仅仅应用在这里，本书后面的很多案例中，都会看到“滑动门”技术的应用。

8.1 自适应的水平菜单

这一节要实现一个简单的水平菜单，效果如图8.1所示。这个菜单可以随着浏览器窗口宽度的变化而改变排列方式。当浏览器窗口宽度不足以容纳所有的菜单项的时候，会自动折行，如图8.1右图所示。

该实例文件位于本书光盘的“第8章\01\horizontal-navi.htm”。

图8.1 自适应的水平菜单效果

它的HTML代码依然是第7章多次使用过的最基本的HTML结构，如下所示。

```
<body>
    <div id="menu">
        <a href="#"> Home </a>
        <a href="#"> Contact Us</a>
        <a href="#"> Web Dev</a>
        <a href="#"> Web Design</a>
        <a href="#"> Map </a>
    </div>
</body>
```

接下来设置#menu容器。这里只需要设置文字的样式就可以了。由于宽度会自动水平适应，因此不需要做额外的设置。代码如下：

```
#menu {
  font-family:Arial;
  font-size:14px;
  }
```

下面对菜单项进行设置。代码如下：

```
#menu a, #menu a:visited {
display:block;                    /*设置为块级元素*/
float:left;                       /*向左浮动*/
color:#000;                       /*黑色文字*/
text-decoration:none;             /*无下划线*/
padding:4px;                      /*内边距*/
margin: 4px;                      /*外边距*/
```

```
        border-top:8px solid #9ab;                    /*上边框*/
        border-bottom:8px solid #9ab;                 /*下边框*/
        }
```

在这段代码中，进行了如下设置。

（1）把a元素设置为块级元素，并设置为向左浮动，这是使菜单项水平排列，并能够自动适应浏览器窗口的关键。

（2）设置了文字的颜色和样式。

（3）设置了内外边距。

（4）设置了上下边框的样式，图中看到的灰色横线就是这里设置的效果。

最后，设置菜单项的鼠标经过效果，将文字的颜色从黑色变为红色，同时将上下两条粗边框变为黑色，以示和其他菜单项的区别，代码如下。

```
#menu a:hover{
  color:#f00;
  border-top:8px solid #000;
  border-bottom:8px solid #000;
}
```

此时在浏览器中的效果如图8.2所示。可以看到，这时已经正确地显示了鼠标经过效果。

图8.2　鼠标经过效果设置完成

分析　仔细观察图8.3中圈圈标示的部分，可以看出，菜单项的水平间隔与竖直间隔相同，这个间距是由相邻的菜单项的margin构成的，根据代码可以知道，菜单项的margin是4像素。在前面的例子中曾经描述过，水平的相邻margin会相加在一起形成间隔，因此这里的水平间隔应该是8个像素；而竖直方向相邻的margin会取二者中较大者为间隔距离，这个现象称为“塌陷”。但是从这里可以看到，当盒子设为“浮动”后，即使在竖直方向上的margin相邻，也不会发生“塌陷”，否则这里的竖直间距应该是水平间距的一半，而不是现在看到的效果。这个结论请读者务必记清。

图8.3　设置为浮动后下相邻margin不再“塌陷”

最后，请读者慢慢地把窗口由宽变窄，在某一个位置将会出现图8.4中左图所示的效果，也就是在菜单项内部发生了折行，这并不是期望的效果。解决办法是在“#menu a, #menu a:visited”的样式中增加如下CSS规则：

```
white-space:nowrap;
```

这条规则的含义是确保在单词之间的空白处不换行。这时效果如图8.4所示，图中这两个窗口宽度相同，左图是没有增加这条规则时在菜单项内部折行的效果，右图则是增加该规则以后的效果。

图8.4 禁止在菜单项内部折行

8.2 自适应的斜角水平菜单

本案例中，制作一个带有斜角的水平菜单。依然和上例一样，它也是可以适应窗口宽度的，效果如图8.5所示。

该实例文件位于本书光盘的“第8章\01\ cut-naiv.htm”。

图8.5 自适应的斜角水平菜单效果

8.2.1 基本思路

本案例的关键在于，如何制作出菜单项左上角这个斜角。如果有了上一章中制作“带箭头的菜单”的经验，掌握了使用“钩子”的方法，本案例的实现就很容易了。

核心的思想就是利用边框的接角位置构造出一个斜角，然后通过使用“钩子”的方法，挂到每一个菜单项的左上角处。当然，读者掌握方法后，可以将斜角放置到菜单的任何一个角上。

对于每一个菜单项的a元素，放置一个span元素，设为corner类别，并作为CSS“钩子”，用于实现斜角效果，代码如下所示。

```
<div id="menu">
   <a href="#">
    <span class="corner"></span>
          Home </a>
   <a href="#">
     <span class=" corner "></span>
        Contact Us </a>
   <a href=" #" >
         <span class=" corner "></span>
              Web Dev </a>
   <a href="#">
         <span class=" corner "></span>
              Web Design </a>
   <a href="#">
         <span class=" corner "></span>
              Map    </a>
</div>
```

8.2.2 基本设置

首先设置#menu容器。仅设置文字的样式即可，无需设置其他属性，代码如下。

```
#menu {
font-family:Arial;
font-size:14px;
}
```

接着，设置每一个菜单项的基本属性，代码如下。

```
#menu a, #menu a:visited {
     display:block;                    /*设置为块级元素*/
     float:left;                       /*向左浮动*/
     position:relative;                /*相对定位，用于设置斜角*/
     background-color:#c00;
     color:#fff;
     text-decoration:none;
     padding:6px;
     margin:1px 0 0 1px;               /*使菜单项之间间隔1像素*/
     }
```

这段代码中，将a元素设置为块级元素，并向左浮动。此时，在浏览器中的效果如图8.6所示。

图8.6 完成基本设置

注意 这里通过position属性将每个菜单项的定位方式设置为相对定位，目的不是要移动菜单项本身，而是通过这个设置将自身变为“包含块”，从而才能使挂在它上面的斜角元素以它为基准来定位。

这时基本设置就完成了，下面就要实现斜角的效果了。

8.2.3 设置斜角效果

这里先介绍一下斜角的产生方法。假设有如下一个简单的完整网页。

```
<!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>
<style>
        div{
        position:absolute;
        height:0;
        width:0;
        overflow:hidden
        border-style:solid;
        border-width:60px;
        border-color:#ccc #999 #666 #333;
        }
</style>
</head>
<body>
   <div></div>
</body>
</html>
```


这个页面在IE和Firefox中的效果分别如图8.7左图和图8.7右图所示。

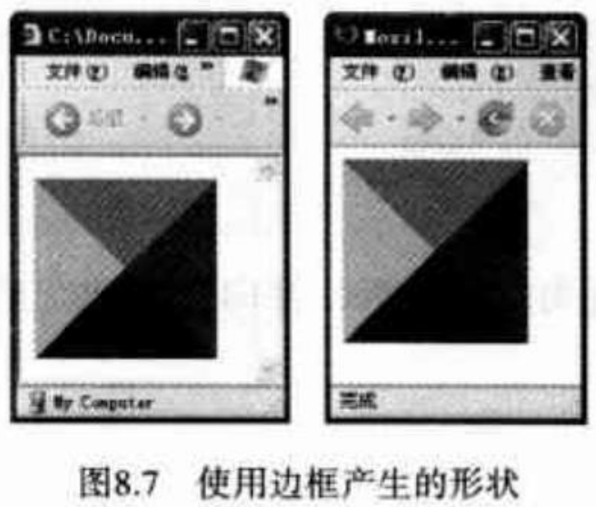


图8.7 使用边框产生的形状

可以看到这种设置方法，对于Firefox和IE都可以实现一个正方形，并且4个三角形独立控制颜色。接下来在上面的样式中找到如下两行：

```
border-width:60px;
border-color:#ccc #999 #666 #333;
```

修改为：

```
border-width:0 0 60px 60px;
border-color:#fff #fff #666 #fff ;
```

这时产生的效果如图8.8所示，正是本案例所需要的斜角形状。

图8.8 使用边框产生的斜角

有了这个基础，再继续完成导航条就很容易了。根据上面设置斜角的方法，对“#menu a .corner”进行设置，代码如下。这里对边框的样式写法稍有不同，本质完全相同。

```
#menu a .corner{
position:absolute;
height:0;
width:0;
overflow:hidden;
border-bottom:solid 6px #c00;
border-left:solid 6px #fff;
top:0;
left:0;
}
```


这里与上面试验的斜角的代码相比，只多了如下两行。

```
top:0;
left:0;
```

这两行的作用是进行斜角的定位。由于前面的a元素已经设置为相对定位，而这里的这个span是a的下级元素，又设置为绝对定位，因此这个斜角将以a元素的位置为基准进行定位。

经验 那么可能读者会想到，既然top和left都设置为0，是不是可以省略这两行语句不写呢？

答案是否定的，如果省略这两个语句，效果将如图8.9所示。这是因为，默认情况下，这个斜角将放置在“内容”的左上角，而我们希望的是放置在以边框为界的左上角。

图8.9 错误的斜角位置

当把这两行语句加上以后，就可以看到正确的效果了，如图8.10所示。接下来的任务就是设置鼠标经过效果了。

图8.10 正确的斜角位置

8.2.4 设置鼠标经过效果

下面设置鼠标经过效果。这里希望鼠标指针经过某个菜单项的时候，菜单项的背景颜色由红色改变为橘黄色，文字由白色变为深灰色，代码如下。

```
#menu a:hover{
background-color: #f90;
color:#333
}
```

这时的效果如图8.11所示，有点像“折角”的效果，看起来也很不错。如果喜欢这个效果，就可以到这里结束了。

图8.11 设置鼠标经过效果

如果希望当鼠标指针经过时，斜角部分也变成橘黄色，可以增加如下代码。

```
#menu a:hover span.corner{
border-bottom:solid 6px #f90;
}
```

这样，当鼠标指针经过菜单项时，包括斜角部分，也会一同变为橘黄色，效果如图8.12所示。至此本案例就全部完成了。

图8.12 设置鼠标经过效果

8.3 应用滑动门技术的玻璃效果菜单

本例中要实现一个玻璃材质效果的水平菜单。为了表现出立体的视觉效果，以及玻璃的质感，必须借助图像才可以实现，完成后的效果如图8.13所示。

该实例文件位于本书光盘的“第8章\03\glass-navi.htm”。

图8.13 玻璃效果的菜单

本例中用到了两个图像，分别如图8.14左图和图8.14右图所示。

图8.14 本案例中用到的两个图像文件

可以看出，左边的图像是作为整个菜单的背景色平铺使用的，右边的玻璃材质图则是当鼠标指针经过某个菜单项的时候显示出来的。

从效果图中可以看出，玻璃材质图是一个固定的图像文件，而菜单中的各个菜单项宽窄不一，却都可以完整地显示出来，这是如何实现的呢？这里使用的就是“滑动门”技术，它被广泛应用于各种CSS效果中，因此希望读者能够真正理解这个案例的本质原理。

8.3.1 基本思路

首先讲解滑动门技术的核心原理。图8.15中的箭头表示了两个圆角矩形图像的滑动方向。较深颜色区域表示二者重叠的部分，当需要容纳较多文字时，重叠就少一些，而需要较少文字时，重叠就多一些。两个图像可以滑动，重叠部分的宽度会根据内容自动调整，就像两扇推拉门一样，因此这种技术就被称为“滑动门”。

图8.15 滑动门技术的原理示意图

本例与前面的例子相比，要更复杂一些。除了菜单项需要设置图像背景外，整个菜单也需要设置图像背景，因此这里的HTML结构将使用无序列表来组织，而不是仅仅使用a标记。这样做的好处是可以更方便地进行控制。

相应的HTML代码如下。

```
<div id="menu">
    <ul>
        <li><a href="#"><b> Home </b></a> </li>
        <li><a href="#"><b> Contact Us </b></a></li>
        <li><a href="#"><b> Web Dev </b></a></li>
        <li><a href="#"><b> Map </b></a></li>
    </ul>
</div>
```

可以看到，每个文字链接都是作为一个列表项目<li>出现的。此外，还对文字设置了加粗显示的效果，这不但可以使字体变粗，而且还可以作为设置玻璃材质背景的“钩子”使用。结合前面的滑动门的原理，可以知道，为了实现滑动门需要两个背景图片，因此就需要两个“钩子”来分别设置背景图片，这里的<a>标记和<b>标记就分别承担了左右门的钩子的任务。

8.3.2 设置菜单整体效果

下面设置菜单的整体效果。

① 设置文字的字体和字号，并设置菜单的总体宽度，这个宽度值可以先设一个比较大的值，等最终效果做好以后，再调整到合适的值，代码如下。

```
#menu {
  font-family:Arial;
  font-size:14px;
  width:500px;
  }
```

② 设置ul的样式，代码如下。

```
#menu ul {
        padding:0 0 0 8px;
        margin:0;
  list-style:none;
  height:35px;
  background:url(under.gif);
  }
```

这里首先设置了padding和margin，然后将list-style属性设置为none，这样可以取消每个列表项目前面的圆点。然后设置高度为35像素，这正是背景图像的高度，最后将背景设置为图像所在的地址。

③ 设置#menu容器中的li的属性。li原本就是块级元素，这里将其设置为向左浮动，这样将使得各列表横向排列，而不是默认的竖直排列，代码如下。

```
#menu ul li {
  float:left;
  }
```

④ 将a元素设置为块级元素，这样整个矩形范围内都会响应鼠标事件，代码如下。

```
#menu ul li a{
    display:block;
    line-height:35px;
    color:#ddd;
    text-decoration:none;
    padding:0 0 0 14px;
    }
```

上面这段代码中，将a标记设置为块级元素以后，设置了行高line-height属性。设置行高可以使文字竖直方向居中排列。然后将文字设置为浅灰色，并取消链接的下划线。最后，设置padding属性，在每个菜单项的左侧设置了14像素的内边距。

这时在浏览器中的效果如图8.16所示。

图8.16 完成基本设置

接下来就是最关键的任务了——设置菜单项的背景。

8.3.3 使用“滑动门”技术设置玻璃材质背景

① 首先设置a元素的鼠标经过效果，代码如下。

```
#menu ul li a:hover{
  color:#fff;
  background: url(hover.gif);
  }
```

这里将文字设置为白色，然后将玻璃质感的图像文件地址作为背景属性的值，此时在浏览器中查看的效果如图8.17所示。

图8.17 设置左侧滑动门

注意 可以看到，鼠标指针经过时，玻璃材质的背景已经出现了，但是它的右边被齐刷刷地切断了，而没有出现背景图像的右端。这个问题如何解决呢？

在CSS中是不能使图像的宽度缩放的。解决方案之一是为每一个菜单项创建各自宽度的背景图像，但是显然适应性要差很多，而且会需要多个图像文件，增加下载的流量，因此不是一个好办法。

另外一个可行的解决方案是使用前面在HTML中设置的文字加粗标记<b>。基本思想就是把<b>标记作为“钩子”来设置CSS样式，因此可以再为它的背景设置一个背景图像。这个背景图像仍然使用惟一的玻璃材质图像文件，不同的是这次从右向左展开，这样就可以出现右边的端点了。具体的方法如下。

② 对b标记的属性进行设置，这里仅需将其设置为块级元素就可以了，代码如下。

```
#menu ul li a b{
 display:block;
 }
```

③ 设置在鼠标指针经过时的b标记样式。这是很关键的一个步骤，代码如下。

```
#menu ul li a:hover b{
      color:#fff;
      background: url(hover.gif) no-repeat right top;
      }
```


上面的代码中首先设置文字颜色为白色，这样鼠标指针经过时效果会更加醒目。然后设置背景图片，这个图片将会覆盖在前面定义的“#menu ul li a:hover”中设置的图片的上面。这两个图片实际上是同一个图片，后面的“no-repeat right top”设定了这个背景图的铺设方式，只显示一次，并从右上角开始铺设，此时在浏览器中的效果如图8.18所示。

图8.18 设置右侧滑动门

④ 这样基本上已经成功了，只是背景图像还不对称，右边还应该增加一些空白，这只需要在“#menu ul li a b”的样式中增加一条内边距的样式，在最右侧对称地增加14像素内边距即可，代码如下。

```
#menu ul li a b{
 display:block;
 padding:0 14px 0 0;
        }
```

此时在浏览器中的效果如图8.19所示，这正是我们需要的效果。

图8.19　完成调整

8.3.4 进一步解决的问题

1. 修饰菜单项的文字

这里需要提示一点。为了能够增加玻璃材质的背景图像，我们使用了b标记作为“钩子”，来挂接CSS样式，这样菜单项的文字就以粗体显示了。如果不想使用粗体，那么也很简单，只需要在“#menu ul li a b”和“#menu ul li a:hover b”两个选择器中分别增加一条样式，使文字的粗细为正常（normal）即可，效果如图8.20所示。

图8.20　重置b标记的字体设置

2. 菜单的背景随浏览器窗口扩展

此外，由于设置了#menu容器的width属性（500px），因此这个导航菜单的宽度就是固定的了，而有时可能希望菜单的背景随浏览器窗口的扩展而扩展。

为了实现自动扩展，可以进行如下尝试。对#menu的设置有如下代码：

```
#menu {
font-family:Arial;
font-size:14px;
width:500px;
}
```

将其修改为：

```
#menu {
  font-family:Arial;
  font-size:14px;
  margin:0 auto 0 0;
        }
```

这样简单的修改确实实现了可以随浏览器窗口扩展，如图8.21所示。

图8.21 使菜单背景图像适应浏览器窗口宽度

3. 设置滚动条

这时又出现了新的问题，当浏览器变窄以后，效果如图8.22左图所示，菜单项会自动折行。在大多数情况下，并不希望出现这种效果，而是希望窗口变窄到一定程度时自动出现滚动条，如图8.22所示。

图8.22 使菜单项不会折行

这是如何实现的呢？原来蓝色的背景图像是设置在ul的下面的，为了实现菜单项不自动折行，就要给ul设置width属性，这样就会导致背景图像也无法扩展。解决方法是将背景图像从ul移动到#menu容器中，这样背景图像会随着#menu容器的扩展而扩展，同时ul设置宽度后，又限制了菜单项的自动折行。修改后的代码如下。至此本案例全部完成。

```
#menu {
  font-family:Arial;
  font-size:14px;
  margin:0 auto 0 0;
  background:url(under.gif);
}

  #menu ul {
    display:block;
    width:500px;
    height:35px;
    padding:0 0 0 8px;
```

```
    margin:0;
list-style:none;
}
```

4. IE和Firefox的兼容性

最后，再来看一下这个菜单的浏览器兼容性。在Firefox中，显示效果没有任何问题，而在IE中，会发现一个小问题，如图8.23所示。

图8.23 在IE中响应鼠标的范围错误

可以看到，在IE中只有当鼠标指针移动到文字上面的时候，才会触发鼠标经过效果。在图中，鼠标指针的尖端虽然离文字只有很小的距离，仍然没有触发鼠标效果。而我们希望的是进入菜单项的矩形范围内，就可以像在Firefox中显示的那样触发鼠标经过效果，如图8.24所示。

图8.24 Firefox中正确的响应鼠标的范围

这是IE浏览器存在的一个问题，解决方法是在“#menu ul li a”的样式中增加一条CSS规则，如下：

```
float:left;
```

这里并没有什么原理需要解释，只是一种可行的解决方法。如图8.25所示，已经显示正确的效果了。

图8.25 扩大IE中响应鼠标的范围

8.4 三状态玻璃效果菜单（双层滑动门应用）

本节中进一步讨论使用背景图片制作菜单的方法，目标是实现一个具有3个状态的菜单，如图8.26所示。

该实例文件位于本书光盘的“第8章\04\3-state-navi.htm”。

图8.26 三状态玻璃效果菜单

这里说的3个状态分别是，普通状态时菜单项为灰色，鼠标指针经过时的菜单项显示为浅紫色，表示当前页时的菜单项为深紫色。

注意 本案例和8.3节的案例比起来，需要有以下3点改变。

（1）在8.3节的案例中，背景色是固定的一个横条，而在本案例中，背景色是针对每一个菜单项适应宽度的。实现的方法就是对背景再次使用“滑动门”技术，因此这里称为“双层滑动门技术”。希望读者在学习的时候，思路保持清晰，不要在众多CSS样式中迷失方向。

（2）增加了当前所在页面的状态显示。这个效果实际上比较简单，只需要增加相应的类别即可，但是要注意做一些善后工作。

（3）在本例的实施过程中，将使用“单背景文件”的方法。尽管这不是必需的，但是目前这种做法比较流行，因此在这里给予介绍。

8.4.1 设置菜单整体效果

在编写代码之前，首先准备所需的图片，本例中只使用一个图像文件来完成这个效果，这个图像文件如图8.27所示。可以看到，图像中有从上到下依次排列的3个玻璃质感的按钮形状，分别使用了不同的颜色。这个文件高度为105像素，即上中下各35像素高，对应菜单项的不同状态，只需要使用图像中不同的部分，就可以产生完美的效果。

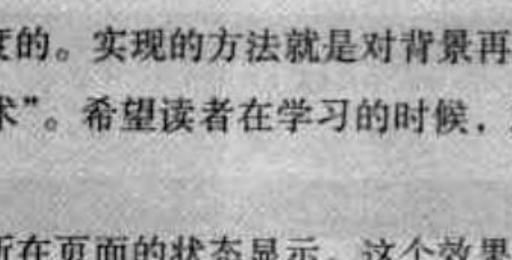


图8.27 本案例中使用的背景图像文件

分析　这样做主要有以下两个优点。

（1）一个优点是便于网站文件的管理，因为只需要一个图像文件。

（2）鼠标指针移动到某个菜单项的时候，如果换为另一个文件，可能会出现暂时的停顿，而这样使用一个文件就不会出现这个问题。

① 现在来设置HTML代码，如下所示。与8.3节的案例非常相似，区别是其中一个li标记设置了current类别，以区别于其他菜单项。

```
<div id="menu">
<ul>
  <li><a href="#"><b> Home </b></a> </li>
    <li><a href="#"><b> Contact Us </b></a></li>
    <li><a href="#"><b> Web Dev </b></a></li>
    <li><a href="#"><b> Web Design </b></a></li>
    <li class="current"><a href="#"><b> Map </b></a></li>
</ul>
  </div>
```

② 对列表元素ul和列表项目li进行设置，代码如下。这里使用自适应方式，因此直接在ul中设置字体等基本样式即可。

```
#menu ul {
  font-family:Arial;
  font-size:14px;
  padding:0 0 0 8px;
  margin:0 auto;
list-style:none;
height:35px;
}
#menu ul li {
  float:left;
  margin:0 2px;
  }
```


这时，在浏览器中观察的效果如图8.28所示。可以看到，将li设置为向左浮动后，列表不再按默认的竖向排列了。

图8.28　向左浮动后的菜单效果

8.4.2 设置第一层滑动门

设置每一个列表项中的链接的属性，这实际上是第一层滑动门。

① 将每一个列表项中的链接设置为块级元素。然后设置向左浮动，这个设置的作用是为了解决在IE中的错误，具体请参考8.3节的案例。其他的设置都与8.3节的案例很类似。

注意 需要提示注意的是padding的设置也和8.3节的案例相同。这里设置为左边留出14像素的padding，用于显示背景图片的左侧边缘。

② 然后设置背景图片。由于浅灰色的图像在文件中的最上面，因此这里不需要设置图片的基准位置。代码如下。

```
#menu ul li a{
      display:block;
float:left;
       line-height:35px;
color:#666;
text-decoration:none;
padding:0 0 0 14px;
background:url('background.gif');
}
```

这时，在浏览器中观察的效果如图8.29所示。可以看到，此时每一个菜单项的普通状态图像已经出现，只是还缺少右边边缘的配合。

图8.29 第一层中的左侧滑动门

③ 与8.3节的案例类似，利用<b>标记，可以挂接右侧边缘所需的图像，代码如下。

注意 这里设置的右侧的14像素padding，以及这里设置的背景图像，都需要设置定位基准点，这里使用整个图像文件的右上部分，因此设置为“right top”。

```
#menu ul li a b{
 display:block;
 padding:0 14px 0 0;
        background:url('background.gif') no-repeat right top;
        }
```

这时，在浏览器中观察的效果如图8.30所示。可以看到，此时每一个菜单项的普通状态图像已完整显示了。下一步就需要设置鼠标经过效果了。

图8.30 第一层滑动门

8.4.3 设置第二层滑动门

现在来设置第二层滑动门，即设置鼠标经过效果，代码如下所示。首先设置文字颜色为白色，以更醒目；其次设置鼠标经过时的背景图像，浅紫色的按钮图像在文件的中间，因此注意这里的背景图像定位基准。

```
#menu ul li a:hover{
  color:#fff;
        background: url('background.gif') left center;
  }

#menu ul li a:hover b{
        background: url('background.gif') right center;
}
```

这时效果如图8.31所示。

图8.31 第二层滑动门

8.4.4 设置表示当前页面的菜单项

① 对表示当前页面所在的菜单项进行特殊设置，即设置class=”current”的列表项目。方法和上面也是类似的，只是把背景图像的定位基准设置为底下的图像，代码如下所示。

```
#menu ul li.current a{
  color:#fff;
        background: url('background.gif') no-repeat left
bottom;
  }

#menu ul li.current a b{
        background: url('background.gif') no-repeat right
bottom;
  }
```

这时在浏览器中观察效果如图8.32所示。可以看到，此时设为current的菜单项颜色为深紫色。

图8.32 设置表示当前页面的菜单项

分析 还应进一步考虑到，如果这时鼠标指针移动到这个设为current的菜单项，依然会像其他菜单项一样，变成浅紫色背景，鼠标指针变为手形光标。但实际上此时的当前页面就是菜单项对应的页面，访问者没有必要点击这个页面，那么鼠标指针移动到这个菜单项上时，应使鼠标指针的形状保持为指针形状。下面就对此进行设置。

② 当鼠标指针经过这个菜单项时，依然使用紫色作为背景，同时强制使用箭头光标，代码如下。

```
            #menu ul li.current a:hover{
                    background: url('background.gif') no-repeat left
bottom;
                   cursor:default;
              }

            #menu ul li.current a:hover b{
                    background: url('background.gif') no-repeat right
bottom;
                   }
```

这时，在浏览器中观察效果如图8.33所示。此时鼠标指针进入该菜单项的时候，鼠标光标仍然是箭头形状，背景还是深紫色，这样就不会吸引访问者点击该菜单项了。

图8.33 修改鼠标指针经过current菜单项时的效果

8.4.5 进一步解决的问题

到这里，该菜单还存在缺陷。这个菜单使用了自适应浏览器宽度的方式，也就是当浏览器窗口变窄的时候，菜单项会自动折行。但是在个别情况下会出现问题，如图8.34所示，当浏览器窗口不足以容纳前3个菜单项的全部时，有可能会在“Web Dev”这个菜单项中间的空白字符处折行，这显然不是希望得到的效果。

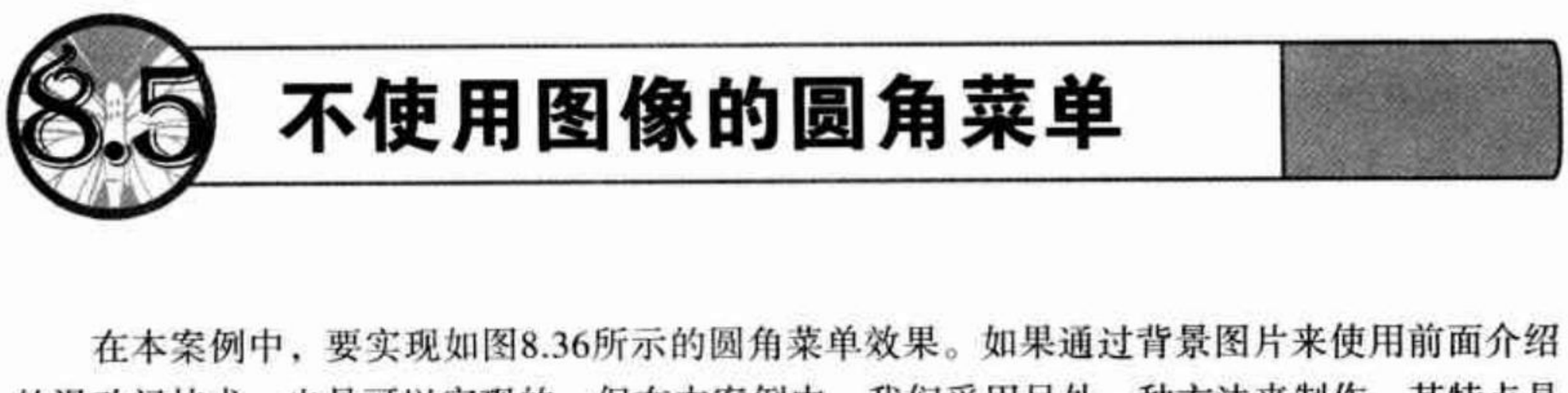
图8.34 菜单项内折行

解决这个问题的方法在前面的案例中曾经应用过的，即在“#menu ul”设置的样式中增加如下规则。

```
white-space:nowrap;
```

它的含义就是一个菜单项不允许从空白字符处折行，这样效果如图8.35所示，“Web Dev”这个项目要么在上一行，要么到下一行，不会像前面那样从中间折断。

图8.35　禁止菜单项内折行



8.5 不使用图像的圆角菜单

在本案例中，要实现如图8.36所示的圆角菜单效果。如果通过背景图片来使用前面介绍的滑动门技术，也是可以实现的。但在本案例中，我们采用另外一种方法来制作，其特点是不使用任何图像文件，仅通过CSS本身来实现这样的效果。

该实例文件位于本书光盘的“第8章\05\rounded-navi.htm”。

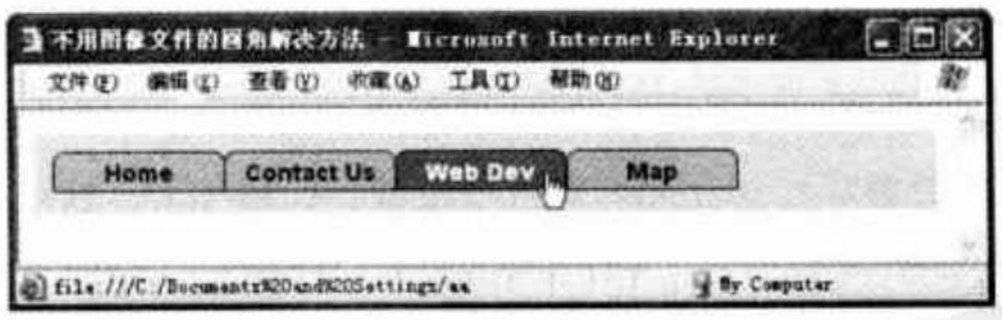


图8.36　不使用图像的圆角菜单

如果读者不看下面的讲解，自己是否可以想出解决方法呢？其实这个方法前面的案例已经使用过了，它类似于制作带箭头的菜单，只不过这次要多挂几个“钩子”。

8.5.1 实现圆角效果

在制作整体圆角菜单之前，先来实现如图8.37所示的一个圆角菜单项。这圆角的文件位于本书光盘的“第8章\05\prepare.htm”。

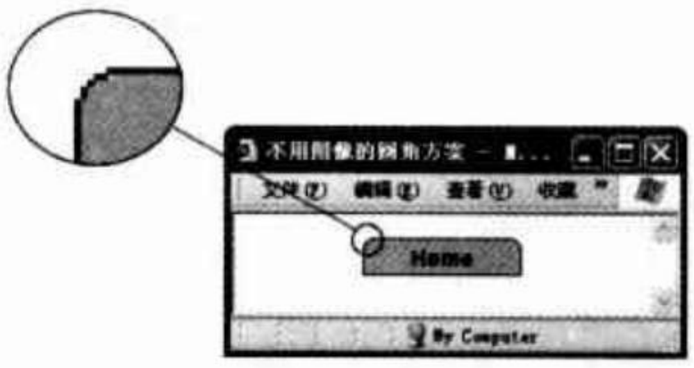


图8.37 圆角的构成

在前面我们曾经制作过不使用图像文件的斜角菜单，是使用CSS的边框来实现的，那么这里怎么做呢？

① 首先观察如下的HTML代码。

```
<body>
  <div class="item">
    <div class="row1"></div>
    <div class="row2"></div>
    <div class="row3"></div>
    <div class="row4"></div>
    <p>Home</p>
  </div>
</body>
```

这正是上面效果所使用的HTML结构，可以看到每一行标记是如何与最终效果中的像素对应的，如图8.38所示。

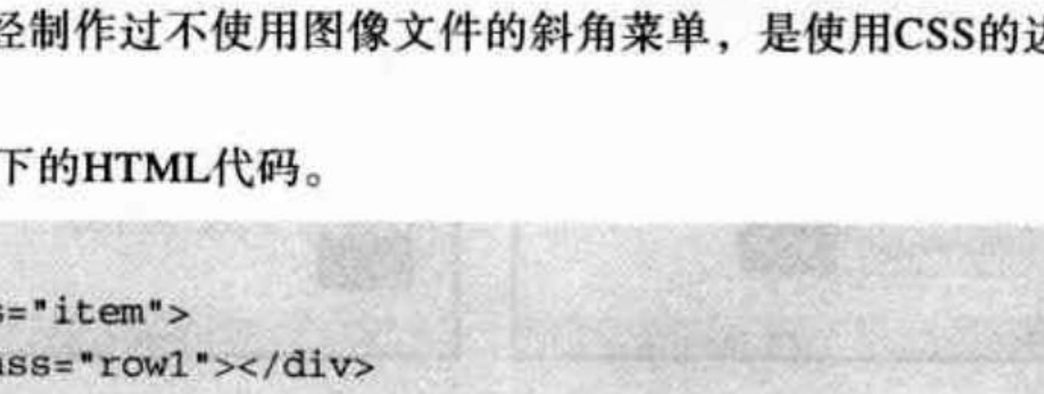


图8.38 构建所需的HTML结构

② 那么CSS又如何设置呢？对.item容器进行设置，代码如下，这里都是一些常规的设置。

```
.item{
  width:100px;
  margin:0 auto;
  padding 0;
  font: 14px Arial;
  font-weight:bold;
  }
```

③ 对p段落的样式进行设置，代码如下。

```
.item p{
  padding:0 0 2px 0;
  margin:0px;
  text-align:center;
```

```
    background:#cc6;
    border:solid 1px  #000;
      border-top-width:0;
  }
```

注意这里对边框的设置。首先设置4条边为1个像素宽，然后把上边框设置为0，这时的效果如图8.39所示，可以看到左边、右边和下边有边框，上边则没有边框。

图8.39　设置菜单项的下部

④ 下面就要对构成圆角的4个div进行设置了。先对它们的公共属性进行设置，代码如下。

```
  .item div{
    height:1px;
    overflow:hidden;
    background:#cc6;
    border-left:solid 1px  #000;
    border-right:solid 1px  #000;
  }
```

它们的共同属性是1像素高，背景颜色和它们下面的矩形背景色相同，并且两端各有1像素来构成边框。

⑤ 设置最上面一行的row1的样式，它需要覆盖共性的背景色。因为它是作为水平的上边框出现的，所以把它的背景色设置为边框颜色即可。同时通过设置左右margin，使它左右各短5个像素，代码如下。

```
  .item .row1{
    margin:0 5px;
    background:#000;
  }
```

此时在浏览器中的效果如图8.40所示。

图8.40　设置菜单项顶部的边线

⑥ 设置第2行row2的样式。这一行覆盖了border属性，使它变为两个像素，从而更接近一个圆角。

```
  .item  .row2{
    margin:0 3px;
```

```
    border:0 2px;
}
```

此时在浏览器中的效果如图8.41所示，可以看到已经出现圆角的雏形了。

图8.41 设置菜单项的圆角

⑦ 使用和上面相同的方法设置接下来的两行，代码如下。

```
.item .row3{
  margin:0 2px;
}

.item .row4{
  margin:0 1px;
  height:2px;
}
```

这样就获得了圆角菜单项的效果了，如图8.42所示，如果读者对这些数据的计算还有不清楚的地方，可以参考前面的图8.38。

图8.42 圆角设置完成

请读者务必先把圆角的制作方法分析清楚，然后再往下学习。

8.5.2 用列表进行改造

上面只是做出了一个菜单项。为了能够实现完整的菜单，还需要改造为利用列表来实现的HTML结构，也就是使用ul代替div，便于以后增加菜单项时能更方便地组织内容。同时增加a标记，使菜单项可以响应鼠标的操作。

① 首先改造THML代码。找到代码中的如下部分。

```
<div class="item">
    <div class="row1"></div>
    <div class="row2"></div>
    <div class="row3"></div>
    <div class="row4"></div>
    <p>Home</p>
</div>
```

改造为：

```
<ul>
        <li class="item">
                    <a href="#">
                    <div class="row1"></div> <div class="row2"></div>
                    <div class="row3"></div> <div class="row4"></div>
                    <p>Home</p>
              </a>
        </li>
    </ul>
```

这样显示的效果和原来是相同的。

② 增加其余几个菜单项，只需要复制后修改p段落中的文字就可以了，这里不再抄录代码。

③ 对CSS样式进行改造。

● 增加对ul的CSS设置，代码如下。

```
ul{
  height:26px;
  margin:0;
  padding:10px;
  list-style-type:none;
  background:#ddd;
}
```

● 对item类增加向左浮动设置，以及对margin设置修改，代码如下。

```
.item{
  float:left;
  width:100px;
  margin:0 -1px 0 0;
  padding 0;
  font: 14px Arial;
  font-weight:bold;
  }
```


分析 这里之所以把右margin的值设置为-1px，是因为希望各菜单项之间有1像素的重叠，这样，在最终的效果中，相邻菜单项之间的竖线就会显示为1像素。

其余代码都不用修改，此时在浏览器中的效果如图8.43所示。

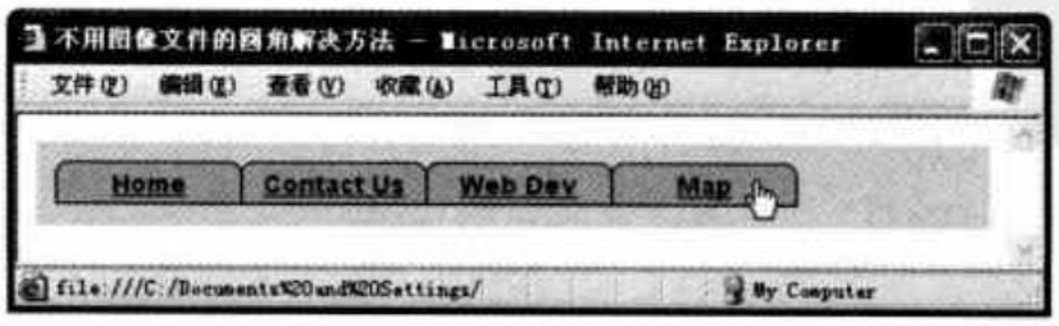


图8.43 增加其他菜单项

8.5.3 设置鼠标响应效果

接下来的任务就是设置鼠标的响应效果了。

① 首先增加链接文字在普通状态时的颜色设置，并取消下划线，代码如下。

```
.item a , .item a:visted{
  color:#000;
  text-decoration:none;
}
```

② 对鼠标指针经过时的p标记进行设置。这样做的目的是使鼠标指针经过时，文字颜色和背景色发生变化，代码如下。

```
.item a:hover p{
  background:#884;
  color:#fff;
}
```

分析 这时在IE浏览器中查看效果，会发现并没有实现所希望的效果，从上面这两段代码本身来说，并没有错误，那么问题出在哪里呢？先来看解决这个问题的方法，增加如下一段代码。

```
.item a:hover{
      background:transparent;
}
```

它的作用是将a标记的鼠标指针经过样式的背景色设置为透明。事实上对于各种元素，默认的背景色就是透明的。但是当某一个元素的上级元素设置了某种背景色之后，它的后代的背景色就继承了这种颜色，因此这里就需要把它恢复为透明色，才可以正确地显示鼠标经过效果。修改后的正确效果如图8.44所示。

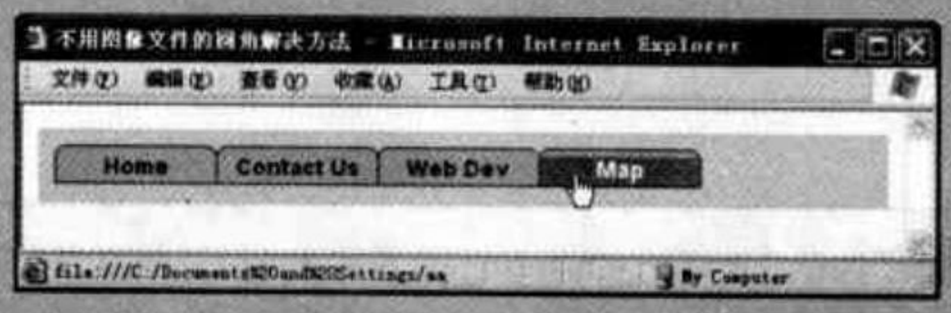


图8.44 响应鼠标经过事件

③ 下面就很简单了。对矩形上面的4个构成圆角的div在鼠标指针经过时的效果进行设置，只要设置背景色就可以了，代码如下。

```
.item a:hover .row2,
.item a:hover .row3,
.item a:hover .row4{
        background:#884;
        }
```

这时在浏览器中的效果如图8.45所示。

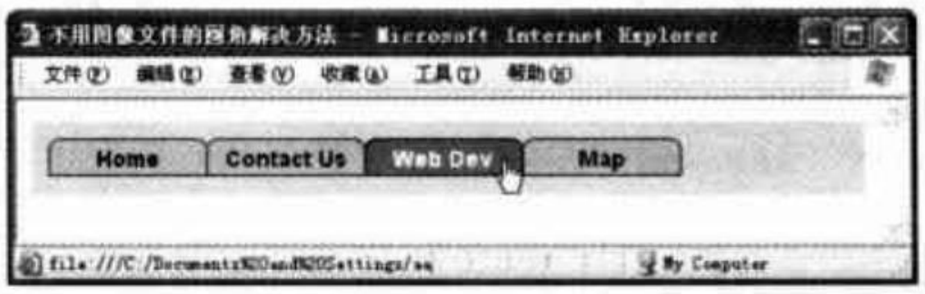


图8.45　完成设置后的效果

8.6 会跳起的多彩菜单

本案例将在8.5节的基础上来制作会跳起的多彩菜单，效果如图8.46所示。在平常状态时，各菜单项的高度都同样，而当鼠标指针经过某一个菜单项的时候，该菜单项就会跳起，高于其他菜单项。这个效果非常有趣，能够吸引访问者的目光。由于本书采用黑白印刷，因此多彩的效果无法体现，请参考本书前面的彩色插页。

该实例文件位于本书光盘的“第8章\06\rounded-navi.htm”。

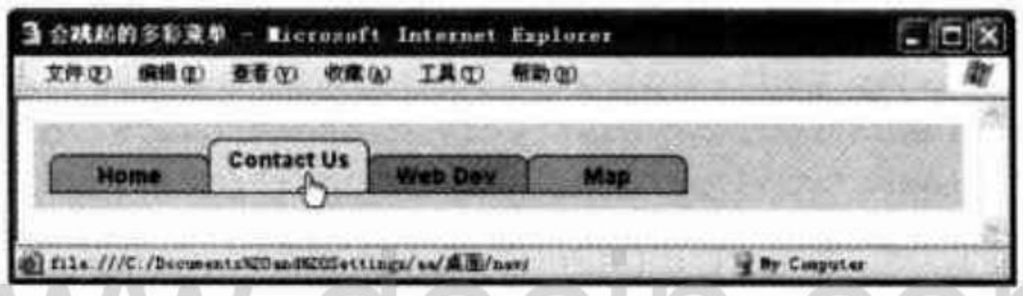


图8.46　会跳起的多彩菜单

注意　本案例的制作完全基于8.5节的最终成果进行，因此请读者一定先做出8.5节的效果，再学习本案例。

8.6.1 实现跳起效果

这里来实现跳起的效果。基于8.5节的制作过程，自然会想到，这里让某个菜单项跳起来，就是让菜单项在鼠标指针经过时，使菜单文字所在的p段落变高一些，那么只要给它的padding-bottom增加若干像素就可以了。

先来试验一下，在“.item a:hover p”中增加对padding-bottom的设置，代码如下。

```
.item a:hover p{
  background:#884;
  color:#fff;
  padding-bottom:10px;
}
```

在浏览器中看到如图8.47所示的效果，并不是希望的效果。

图8.47 错误的跳起方向

分析 这是由于每个菜单性都是向左浮动排列的，这些盒子都是根据上边缘对齐的，如果某个菜单项中的p段落变高了，它自然就会向下延伸，而不是向上延伸。本书光盘中的“第8章\06\wrong-method.htm”文件是这种错误效果的页面，请读者参考。

解决办法是实现为每一个菜单项的上面增加一个预留的高度空间，然后在鼠标指针经过某一个菜单项的时候，使该菜单项的预留空间高度变为0，同时增加下面的p段落的高度。这样，如果计算精确，就可以实现跳起的效果了。

具体的做法如下。

① 首先改造HTML。在每一个li中，增加一个div，这里把类别名设为“pad”，即垫子的意思，代码如下。这里的代码只抄录了一个li的代码，其余3个都是相同的。

```
<li class="item">
  <a href="#">
      <div class="pad"></div>
      <div class="row1"></div> <div class="row2"></div>
      <div class="row3"></div> <div class="row4"></div>
      <p>Home</p>
  </a>
</li>
```

② 对这个“垫子”div设置CSS样式，代码如下。

```
.item .pad{
  height:10px;
  border:0;
  background:transparent;
}
```

上面代码将“垫子”设置为10像素高，边框为0，背景色为透明，这样这个“垫子”看起来就好像不存在一样。

③ 分别设置“垫子”和p段落在鼠标指针经过时的样式，代码如下。

```
.item a:hover p{
  background:#884;
  color:#fff;
  padding-bottom:12px;
}

.item a:hover .pad{
```

```
    height:0px;
}
```

在鼠标指针经过时，“垫子”的高度变为0，菜单文字所在的p段落的padding-bottom变为12像素。注意这里的12像素是因为原本p段落的padding-bottom是2像素，这里加上垫子所占的10像素，因此一共12像素。此时在浏览器中的效果如图8.48所示，已经可以正确地实现“跳起”的效果了。

图8.48　正确的跳起效果

本书光盘中的“第8章\06\popup-navi.htm”文件是制作到这时的文件，请读者参考。

8.6.2 实现多彩效果

为了实现各个菜单项的不同颜色，仍需要对HTML进行改造。为了使各个菜单的颜色不同，可以为每种所需颜色设置一个类别，分别分配给相应的菜单项。

① 若想让第1个菜单项以橘黄色显示，就要在li的类别声明中增加一个orange类别名。注意和前面原先的item类别名之间留一个空格，代码如下：

```
<li class="item orange">
  <a href="#">
      <div class="pad"></div>
      <div class="row1"></div> <div class="row2"></div>
      <div class="row3"></div> <div class="row4"></div>
      <p>Home</p>
  </a>
</li>
```

② 其余3个菜单项也同样处理，颜色分别为黄色、绿色和蓝色，相应增加的类别名称分别为yellow、green和blue。

③ 接下来对增加的这4个类别的CSS样式进行设置。平常状态颜色设置的代码如下。这里将文字所在的p段落以及row2、row3和row4这4个对象的背景色设置为橘黄色（#fa0）。为什么不设置row1呢？因为row1自有一条黑色边框，所以不需要设置背景色。

```
.orange p,
.orange .row2,
.orange .row3,
.orange .row4,
{
  background:#fa0;
}
```

④ 设置鼠标指针经过时的颜色，代码如下。

```
.orange a:hover p,
.orange a:hover .row2,
.orange a:hover .row3,
.orange a:hover .row4,
{
  background:#fa0;
}
```

⑤ 其余3个菜单项做相同处理，黄、绿、蓝3种颜色使用的颜色值依次为#ff0、#0e0和#0cf，读者可以随意选择自己喜欢的颜色。

分析 由于普通状态与跳起状态的背景色相同，使用的都是比较浅的颜色，因此在鼠标指针经过时，文字的颜色就不必变成白色了，否则文字会不容易看清楚。

⑥ 在“.item a:hover p”中，将8.5节对文字颜色的设置删除，就会在鼠标指针经过时仍使用黑色文字。这时在浏览器中的效果如图8.49所示。

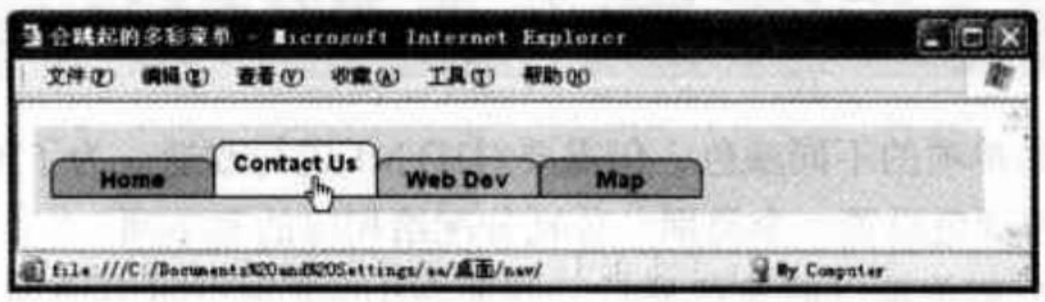


图8.49 实现多彩的菜单效果

本书光盘中的“第8章\06\colorful-navi.htm”文件是制作到这时的文件，请读者参考。这个例子比较复杂，在下一小节中将给出本案例的完整的代码，供读者逐行分析理解。

8.6.3 本案例的完整代码

在进行比较复杂的CSS设计时，可以准备一张草稿纸，经常画一画草图，计算准确是很重要的。本案例的完整代码如下。

```
<!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>会跳起的多彩菜单</title>
<style type="text/css">
ul{
  height:26px;
  margin:0;
  padding:10px;
  list-style-type:none;
```

```
  background:#ddd;
}

.item{
  float:left;
  width:100px;
  margin:0 -1px 0 0;
  padding 0;
  font: 14px Arial;
  font-weight:bold;
  }

.item p{
  padding:0 0 2px 0;
  margin:0px;
  text-align:center;
  background:#cc6;
  border:solid 1px  #000;
    border-top-width:0;
}

.item div{
  height:1px;
  overflow:hidden;
  background:#cc6;
  border-left:solid 1px  #000;
    border-right:solid 1px  #000;
}

.item .pad{
  height:10px;
  border:0;
  background:transparent;
}

.item .row1{
  margin:0 5px;
  background:#000;
}

.item  .row2{
  margin:0 3px;
  border:0 2px;
}

.item .row3{
  margin:0 2px;
}

.item .row4{
  margin:0 1px;
  height:2px;
```

```
}

.item a , .item a:visted{
  display:block;
  color:#000;
  text-decoration:none;
}

.item a:hover{
  background:transparent;
}

.item a:hover p{
  background:#884;
  padding-bottom:12px;
}

.item a:hover .pad{
  height:0px;
}

.item a:hover .row2,
.item a:hover .row3 ,
.item a:hover .row4{
  background:#884;
}

.orange p,
.orange .row2,
.orange .row3,
.orange .row4,
{
  background:#fa0;
}

.orange a:hover p,
.orange a:hover .row2,
.orange a:hover .row3,
.orange a:hover .row4,
{
  background:#fa0;
}

.yellow p,
.yellow .row2,
.yellow .row3,
.yellow .row4,
{
  background:#ff0;
}

.yellow a:hover p,
```

```
.yellow a:hover .row2,
.yellow a:hover .row3,
.yellow a:hover .row4,
{
  background:#ff0;
}

.green p,
.green .row2,
.green .row3,
.green .row4,
{
  background:#0e0;
}

.green a:hover p,
.green a:hover .row2,
.green a:hover .row3,
.green a:hover .row4,
{
  background:#0e0;
}

.blue p,
.blue .row2,
.blue .row3,
.blue .row4,
{
  background:#0cf;
}

.blue a:hover p,
.blue a:hover .row2,
.blue a:hover .row3,
.blue a:hover .row4,
{
  background:#0cf;
}

</style>
</head>

<body>
<ul>
  <li class="item orange"><a href="#">
    <div class="pad"></div>
    <div class="row1"></div> <div class="row2"></div>
    <div class="row3"></div> <div class="row4"></div>
    <p>Home</p>
</a>
</li>
<li class="item yellow"><a href="#">
```

```
    <div class="pad"></div>
    <div class="row1"></div> <div class="row2"></div>
    <div class="row3"></div> <div class="row4"></div>
    <p>Contact Us</p>
  </a>
  </li>
  <li class="item green"><a href="#">
    <div class="pad"></div>
    <div class="row1"></div> <div class="row2"></div>
    <div class="row3"></div> <div class="row4"></div>
    <p>Web Dev</p>
  </a>
  </li>
  <li class="item blue"><a href="#">
    <div class="pad"></div>
    <div class="row1"></div> <div class="row2"></div>
    <div class="row3"></div> <div class="row4"></div>
    <p>Map</p>
  </a>
  </li>
</ul>
</body>
</html>
```

8.6.4 实现向下弹出效果

菜单向上弹出的效果已经比较完善了，最后再给读者提出一个思考题。如果要制作一个向下的弹出菜单，如图8.50所示，制作方法和上面有哪些相同之处，又有哪些不同之处？

图8.50 向下弹出效果

倒转每个菜单项，实际上只需要把row1~row4这4个div移动到p段落的下面就可以了，然后把p段落的下边框设置为0，而上边框设置为1像素，就可实现了。

而要实现向下弹起，就更简单了，不需要使用“垫子”div，直接把p段落的padding-top值增加就可以实现跳起了。

这里只列出修改后的HTML代码，至于CSS的修改方法，相信读者完全可以自己解决了。HTML代码如下。

```
<body>
<ul>
  <li class="item orange"><a href="#">
    <p>Home</p>
    <div class="row4"></div> <div class="row3"></div>
```

```
      <div class="row2"></div> <div class="row1"></div>
    </a>
    </li>

    ……这里是其余三个菜单项……
  </ul>
  </body>
```

本书光盘中的“第8章\06\popdown-navi.htm”文件是制作完成的“向下弹出”效果的文件，请读者参考。

8.7 本章小结

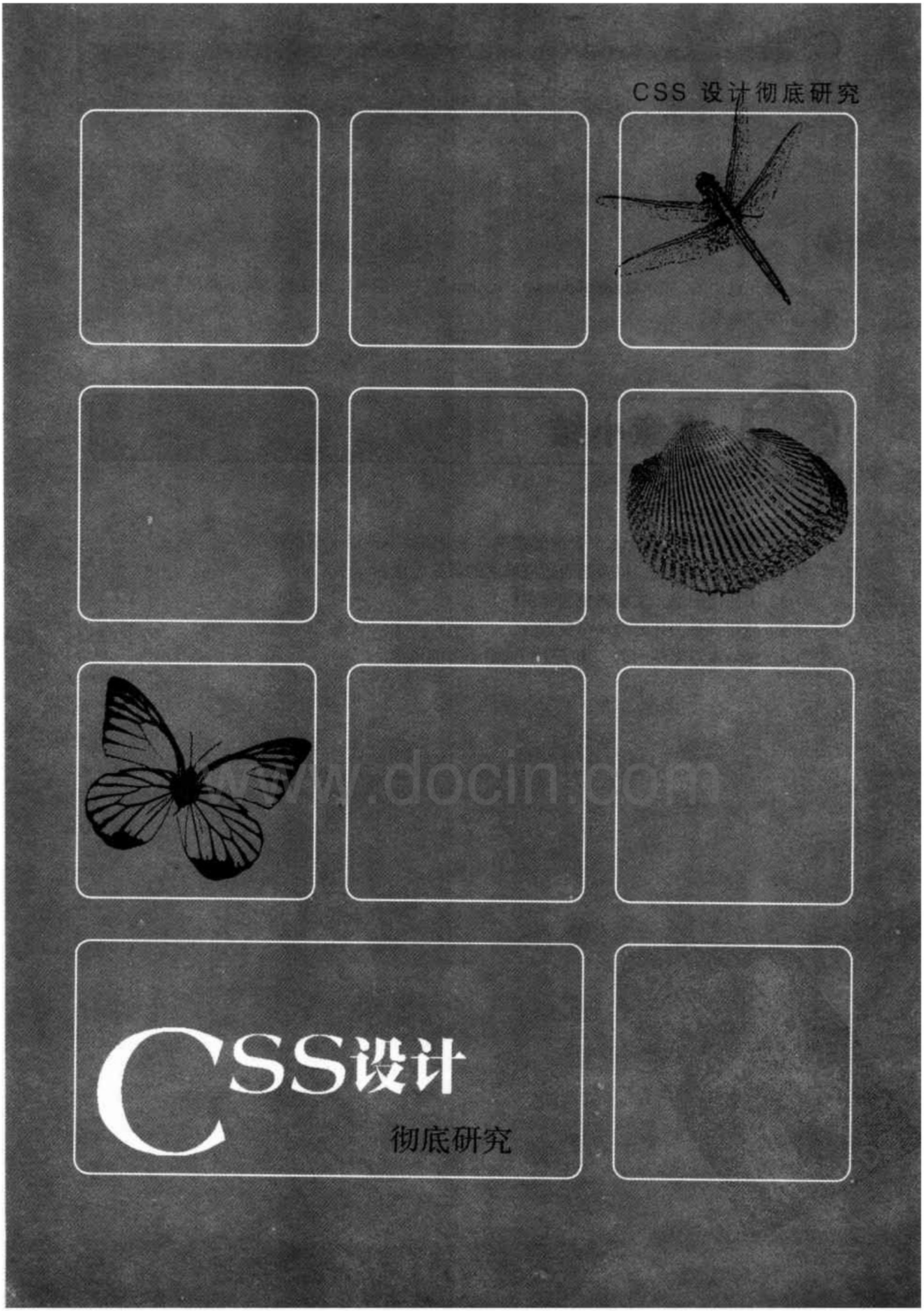
在本章中，实现了6个水平方向的菜单，主要需要掌握以下基本要点。

（1）设置浮动方式，以及是否适应宽度的设置方法。

（2）不使用图像，实现斜角和圆角的方法。

（3）使用背景图片的滑动门技术。

（4）理解本章案例中的“钩子”和“垫子”的用途。



第9章 下拉菜单

本章中，我们将在前面两章的基础上，制作出更为复杂的下拉菜单。当网站的内容比较丰富，结构比较复杂的时候，一级导航的菜单往往就不够用了，这时就需要多级菜单来实现层次结构。实际上任何使用电脑的人对下拉菜单都不会陌生，几乎所有的软件都会通过菜单为用户提供操作的命令接口。

在早期的网站上，制作多级的菜单是件很麻烦的事情，而且不易维护。但是近年来，由于CSS不断普及，使用CSS制作的菜单已经可以非常方便地为网站带来清晰的导航支持。

本案例就来制作一个下拉菜单，图9.1所示为鼠标离开菜单时子菜单未打开的效果，图9.2所示为一个子菜单打开后的效果。

图9.1 关闭子菜单时的效果

图9.2 显示子菜单时的效果

9.1 HTML中的dl、dt和dd标记

这里介绍一组HTML标记——dl、dt和dd。这组标记不像ul和li标记那么常用，但是用于定义本案例中的结构是非常合适的。

dl、dt、dd这组标记和ul、li非常类似，也用于列表结构。dl被称为“定义列表”（definition list），在使用方法上相当于ul；dt和dd分别称为“定义标题”（definition title）和“定义描述”（definition description），它们在使用方法上相当于li。例如有下面一段HTML代码：

```
<body>
  <dl>
<dt>北京市的大学</dt>
        <dd>清华大学</dd>
        <dd>北京大学</dd>
        <dd>人民大学</dd>
    <dt>上海市的大学</dt>
        <dd>复旦大学</dd>
        <dd>同济大学</dd>
```

```
                <dd>上海大学</dd>
            <dt>南京市的大学</dt>
                <dd>南京大学</dd>
                <dd>东南大学</dd>
    </dl>
</body>
```

在浏览器中的效果如图9.3所示。可以看到dl相当于ul，dt和dd相当于li，区别在于dt和dd会被区别对待。如果希望菜单具有子菜单，就可以把主菜单的菜单项文字定义为dt，下属子菜单中的菜单项定义为dd。

图9.3 使用“定义列表”

9.2 菜单的HTML结构

下面就来定义菜单的HTML结构。思路是把整个菜单分为4组，每一组在平常状态时，只显示一级菜单项，当鼠标指针经过时，显示相应的子菜单。首先用ul和li定义出外层结构，然后在每个li项目中，使用dl定义一个菜单项，一级菜单文字定义为dt，子菜单的文字定义为dd。代码如下。

```
<body>
  <ul>
        <li>
            <dl>
                <dt><a href=" #" >Artech Studio</a></dt>
                <dd><a href=" #" >Web Dev</a></dd>
                <dd><a href=" #" >Web Design</a></dd>
                <dd><a href="#">Books</a></dd>
                <dd><a href="#">Contact Us</a></dd>
            </dl>
        </li>
        <li>
            <dl>
```

```
                    <dt><a href="#">Artech Store</a></dt>
                    <dd><a href="#">Books</a></dd>
                    <dd><a href="#">DVDs</a></dd>
                    <dd><a href="#">Movies</a></dd>
                    <dd><a href="#">Service</a></dd>
                </dl>
            </li>
            ……其余两项类似，这里省略……
        </ul>
    </body>
```

在浏览器中的效果如图9.4所示。可以看到每组li中，第一行文字和后面的文字之间有所区别。下面就开始进行CSS样式设置了。

图9.4 HTML结构

9.3 对整体进行设置

① 首先需要对菜单进行整体设置，为上面的代码中的ul设定一个id，代码如下。

```
    <ul id="menu">
```

② 对ul做常规的CSS设置，将margin、padding清零；设定固定宽度610像素，因为每个菜单项要设为150像素宽，再加上一些间隔，所以这里设置610像素；此外清除列表项前面的圆点。

```
#menu {
  margin:0;
  padding:0;
  width:610px;
  list-style-type:none;
  font:14px Arial;
  }
```

③ 对每一个li项目进行设置。设置微向左浮动，以实现横向排列；设定宽度、margin和padding，这里将右边的margin设置为1像素，使各个菜单项之间间隔1像素。然后，把定位

方式设置为相对定位，也使它的下级对象（dl）能够以它为基准进行绝对定位。

```
#menu li {
  float:left;
  width:150px;
  padding:0;
  margin:0 1px 0 0;
  }
```

在浏览器中的效果如图9.5所示，可以看到菜单已经横向排列了。

图9.5 各li横向排列

读者可以参考光盘中这个步骤完成后的“第9章/Final-1.htm”文件。

9.4 对dl进行设置

接下来，对dl对象设置样式，代码如下。

```
/* 设置菜单项*/
#menu li dl {
  margin: 0;
  padding: 0 0 10px 0;
  background: #cb6 url(images/bottom.gif) no-repeat bottom left;
  }
```

此时在浏览器中的效果如图9.6所示，可以看到每个菜单项都增加了背景色，同时下端还产生了圆角的效果，这是如何实现的呢?

读者可以参考光盘中这个步骤完成后的“第9章/Final-2.htm”文件。

图9.6 各li横向排列

分析 下面分析一下其中的这个语句：

```
        background: #cb6 url(images/bottom.gif) no-repeat
bottom left;
```

这句话同时设置了背景色（#cb6）和背景图像（图像文件地址为“images/bottom.gif”）。CSS规定，在这种情况下，背景图像覆盖到的地方显示背景图像，而背景图像没有覆盖的地方则显示背景色。这里的图像只显示一次，即不平铺（no-repeat），并且放置在菜单项的最底下（bottom）。此时，这个图像除了在两侧的圆角位置为白色之外，其他位置都是透明的，这样透明的位置就显示出了背景色，而圆角位置则显示出了两个圆角。图9.7显示了这个图像的样子，灰色阴影部分表示透明部分，两个圆角为白色。

图9.7 用于设置圆角的背景图片

分析 在对dl的设置中，下端有10像素高的padding，圆角图像就在padding中。除了美观之外，padding的作用是，当菜单完全做好以后，子菜单只有在鼠标指针经过时才会显示出来。访问者要选择最下面的菜单项时，如果鼠标指针稍微超出了下边界，菜单就会立即被隐藏起来，这样在易用性上就会存在缺陷。这里设置一个10像素的padding，这样即使稍稍超出菜单项范围，整个菜单也不至于立即消失。

读者可以试验一下，在Windows程序中，比如Word中，打开一个菜单以后，即使鼠标指针离开菜单范围，菜单也不会消失，只有当鼠标指针在菜单外单击一下，菜单才会消失。但是目前在网页上不能开发出和Windows程序一样的完善用户界面。近几年很多新技术的出现，包括CSS、Ajax等，都是在努力使Web应用的用户体验更好，甚至达到本地程序的水平。因此，在进行Web设计和开发的时候，需要注意的不仅仅是美观和功能，实际上易用性等方面也在很多细节之中体现出来，希望读者能够注意到这一点。

9.5 对主菜单项（dt）进行设置

现在对主菜单项进行设置。

① 使用和上面相同的方法设置圆角，此外在主菜单项的下面设置一条暗红色的边框，作为和子菜单的分割，代码如下。

```
    /* 设置菜单项的dt */
```

```
#menu li dt {
  margin:0;
  padding: 5px;
  text-align:center;
  border-bottom:1px solid #b00;
  background:#ed8 url(images/top.gif) no-repeat top left;
  }
```

② 对主菜单项的连接文字进行设置，都是常规设置，代码如下。

```
#menu li dt a ,#menu li dt a:visited {
  display:block;
  color:#333;
  text-decoration:none;
  }
```

此时，在浏览器中的效果如图9.8所示，可以看到，下拉菜单的雏形已经具备了。

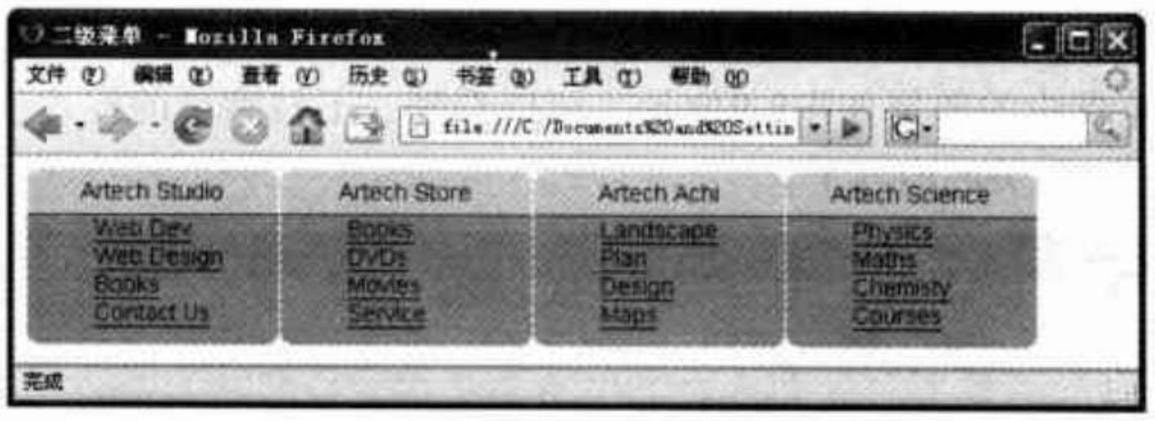


图9.8 设置dt后的效果

读者可以参考光盘中这个步骤完成后的“第9章/final-3.htm”文件。

9.6 对子菜单项（dd）进行设置

设置子菜单项的样式分为3步。

① 首先对每个子菜单项，也就是dd元素进行常规设置，包括文字颜色、背景色等，代码如下。

```
/* 设置菜单项的dd */
#menu li dd {
  margin:0;
  padding:0;
  color: #fff;
  background: #47a;
  }
```

② 为了使最下面的dd的下端和dl的下侧padding之间有一条暗红色分割线，设置1像素的下边框，代码如下。它在视觉上与上面的分割线相呼应，同时也提示了访问者有效的点击位置。

```
#menu li dd.last {
  border-bottom:1px solid #b00;
  }
```

这段代码设置了另一个新的类别.last，因此除了这里的CSS代码之外还需要在HTML中把每组子菜单的最下面一项指定为class=last。例如：

```
<dd class="last"><a href="#">Contact Us</a></dd>
```

③ 设置子菜单连接的样式，这里设置了一个小三角形图像作为背景，代码如下。

```
#menu li dd a, #menu li dd a:visited {
  display:block;
  color:#fff;
  text-decoration:none;
  padding:4px 5px 4px 20px;
  background: #47a url(images/arrow.gif) no-repeat 10px 10px;
}
```

此时，在浏览器中的效果如图9.9所示，可以看到，下拉菜单的样式已经做好了。下面就要设置与鼠标响应相关的内容了。

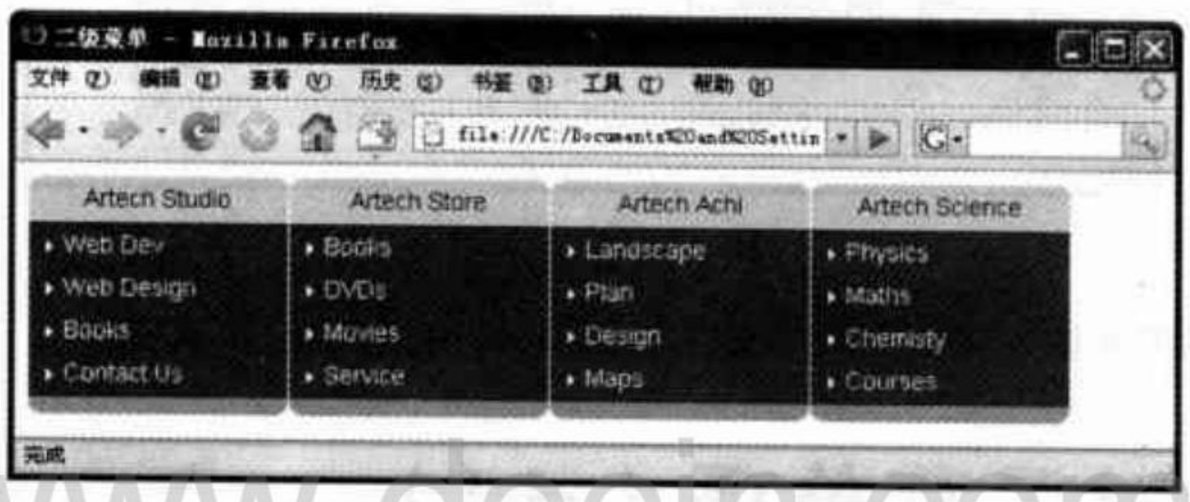


图9.9 设置dd后的效果

读者可以参考光盘中这个步骤完成后的“第9章/Final-4.htm”文件。

9.7 对鼠标响应进行设置

① 首先把子菜单隐藏起来，也就是把所有的dd隐藏起来，代码如下。在浏览器中的效果如图9.10所示。

```
/*关闭子菜单*/
#menu li dd { display:none;}
```

图9.10　把子菜单隐藏起来

② 对li使用:hover伪类，接受鼠标经过事件，然后打开子菜单，代码如下。

```
/* 设置鼠标响应 */
#menu li:hover dd { display:block;}
```

注意　此时在Firefox或者IE 7浏览器中的效果如图9.11所示，已经可以实现鼠标指针经过时打开子菜单了。但是如果使用IE 6浏览器，就不会打开子菜单，因为IE 6浏览器中的“:hover”伪类只对a标记有效，对li是无效的。因此这里先在Firefox中进行调试，后面再介绍如何在IE 6浏览器中调试。

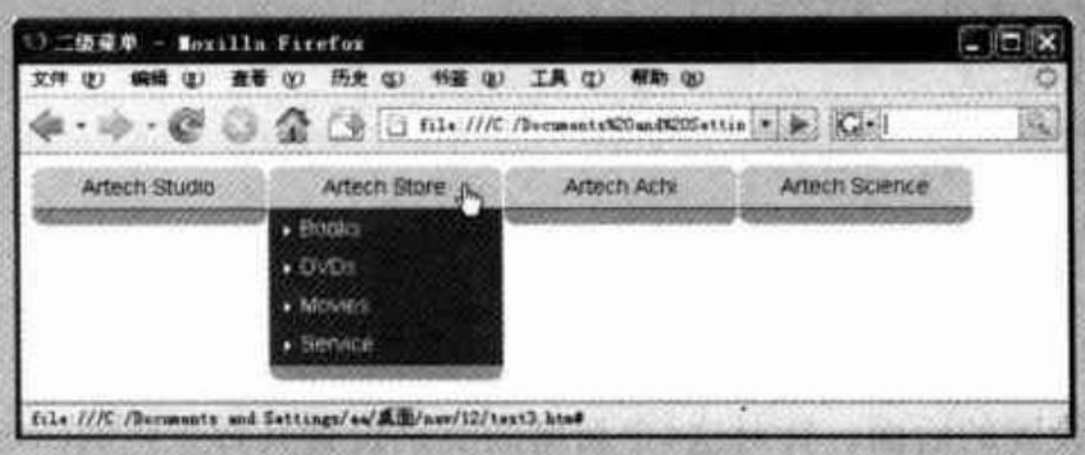


图9.11　显示子菜单

③ 设置在鼠标经过时子菜单项的样式，以体现出区别，代码如下。在浏览器中的效果如图9.12所示。至此，整个菜单在Firefox或者IE 7中的效果就完全实现了。

```
#menu li dd a:hover {
  background: #147;
  color:#9cf;
}
```

读者可以参考光盘中这个步骤完成后的“第9章/final-4.5.htm”文件。

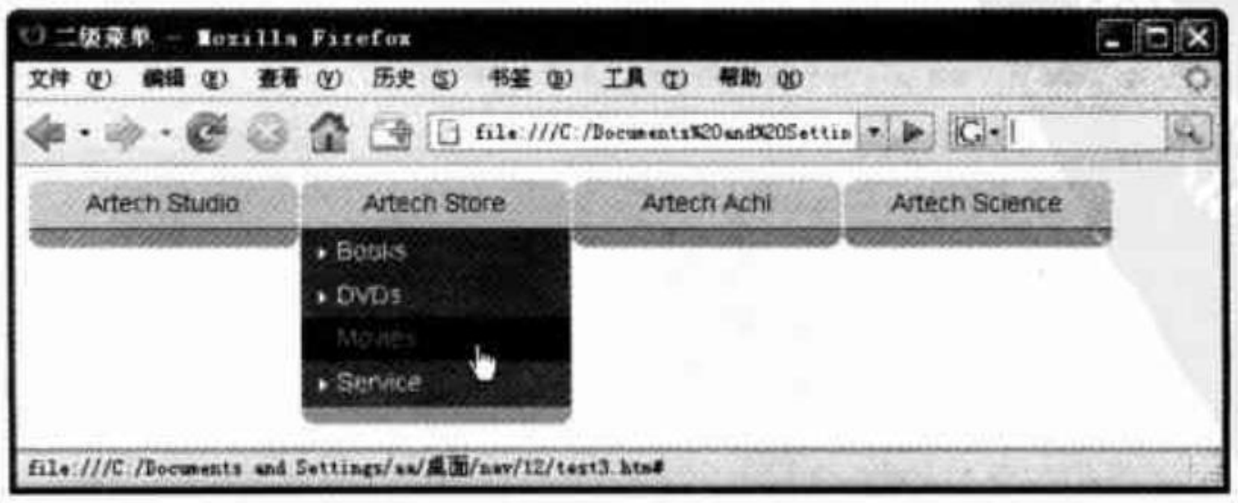


图9.12　设置子菜单项

9.8 实现主菜单项的不同颜色

方法和前面制作跳起的多彩菜单的方法是相同的，这里作为复习。

① 改造HTML代码，方法是在每个dl中，为dt增加一个不同的类别，例如橙色的菜单项代码如下：

```
<dl>
        <dt class="orange"><a href="#">Artech Studio</a></dt>
        <dd><a href="#">Web Dev</a></dd>
        <dd><a href="#">Web Design</a></dd>
        <dd><a href="#">Books</a></dd>
        <dd class="last"><a href="#">Contact Us</a></dd>
</dl>
```

② 在对dt的设置中删除对背景的设置，因为这是共性设置，修改后代码如下：

```
/* 设置菜单项的dt */
#menu li dt {
  margin:0;
  padding: 5px;
  text-align:center;
  border-bottom:1px solid #b00;
  }
```

③ 在共性设置的后面，针对这4个新增的类别设置其背景色的CSS样式，代码如下。

```
    #menu li dt.orange { background:#fa5 url(images/top.gif) no-repeat top
left;}
    #menu li dt.yellow { background:#ee5 url(images/top.gif) no-repeat top
left;}
    #menu li dt.green  { background:#5e5 url(images/top.gif) no-repeat top
left;}
    #menu li dt.blue   { background:#5cf url(images/top.gif) no-repeat top
left;}
```

读者可以参考光盘中这个步骤完成后的“第9章/final-5.htm”文件。此时，在Firefox和IE 7中的效果如图9.13所示。

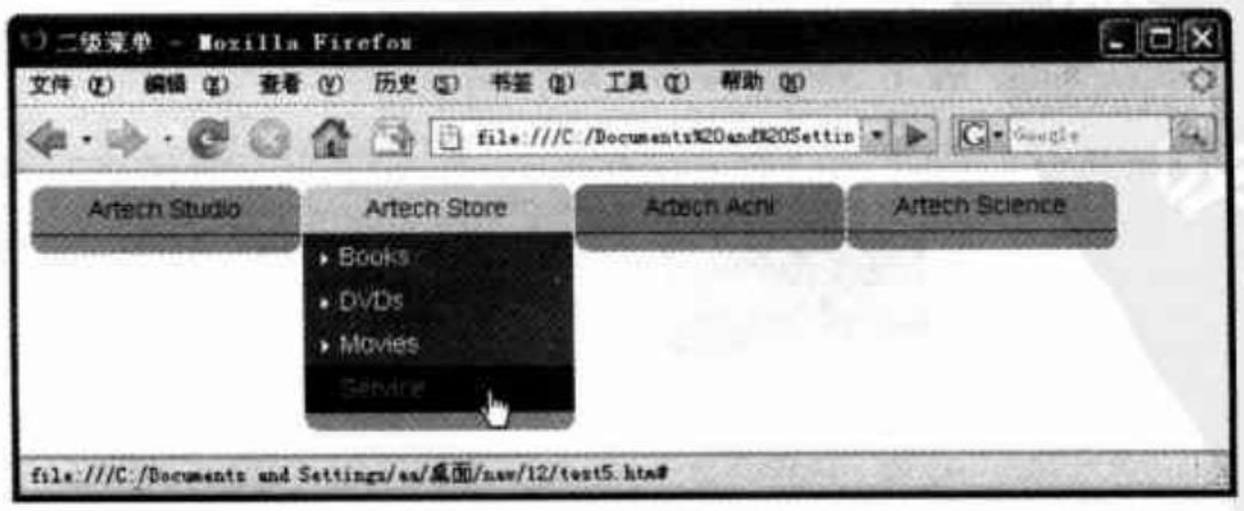
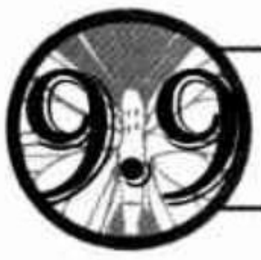


图9.13 设置多彩效果

至此，整个菜单在Firefox和IE 7中的效果就完全实现了。



IE 6兼容

① 改造HTML代码。由于在IE 6中只有a标记才能支持:hove伪类，因此要把每一组“<dl>……</dl>”装到一个表格里，然后再把这个表格装入到<a>标记中，代码如下。

```
            <li>
            <a href="#nogo"><table><tr><td>
                  <dl>
                        <dt     class="orange"><a     href="#">Artech
Studio</a></dt>
                        <dd><a href="#">Web Dev</a></dd>
                        <dd><a href="#">Web Design</a></dd>
                        <dd><a href="#">Books</a></dd>
                        <dd class="last"><a href="#">Contact Us</a></dd>
                  </dl>
            </td></tr></table></a>
            </li>
```

注意增加的代码在第2行和倒数第2行，这样做的目的是为了在后面对a标记使用:hover伪类。此时在Firefox和IE 6中的效果如图9.14所示。

② 由于使用了表格以后，dl的宽度不能自动设置，因此这里人为设置dl的宽度，代码如下。

图9.14 为IE 6增加HTML代码

```
/* 设置菜单项*/
#menu li dl {
  width:150px;/*ie6*/
  margin: 0;
  padding: 0 0 10px 0;
  background: #cb6 url(images/bottom.gif) no-repeat bottom left;
  }
```

此时在Firefox和IE 6中的效果如图9.15所示。

图9.15 设置宽度

③ 由于使用了表格后，导致位置发生了变化，因此需要对表格的样式进行设置，代码如下。

```
/*针对ie6的设置*/
#menu table { border-collapse:collapse;
  padding:0;
  margin:-1px;
  font-size:1em;
  }
```

此时在Firefox和IE 6中的效果如图9.16所示。可以看到，主菜单的效果已经正确了，但是在IE 6中仍然无法打开子菜单。

图9.16 调整表格样式

④ 对鼠标响应进行设置。找到如下一段代码：

```
/* 设置鼠标响应 */
#menu li:hover dd { display:block;}

#menu li dd a:hover {
  background: #147;
  color:#9cf;
  }
```

修改为：

```
#menu li:hover dd , #menu li a:hover dd {
```

```
  display:block;
      } /*ie6*/
#menu li:hover, #menu li a:hover {
  border:0;
      }/*ie6*/
#menu li dd a:hover {
  background: #147;
  color:#9cf;
      }
```

此时在IE 6中的显示效果如图9.17所示。现在子菜单已经可以显示出来了。但是还存在一个问题，在子菜单中，只有当鼠标指针移动到文字上的时候才能激活子菜单项的鼠标经过效果，而如果鼠标指针移到文字旁边则无法触发。我们希望的正确结果是只要鼠标指针进入矩形范围内，都可以触发鼠标经过效果。

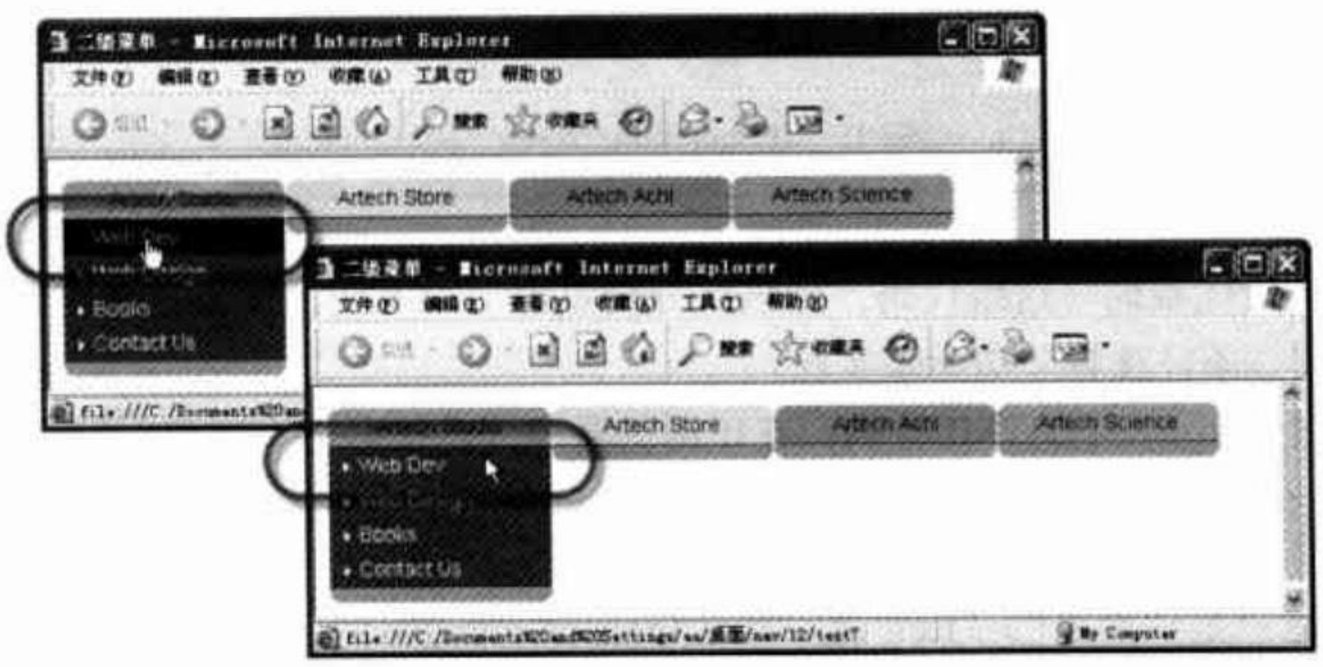


图9.17　在IE 6中存在的问题

⑤ 这是由于IE 6本身存在的固有问题导致的，解决办法是在代码中找到：

```
#menu li dd a, #menu li dd a:visited {
```

然后在它下面增加一行：

```
height:1em;
```

也就是在dd的链接样式中，增加一条高度的CSS样式，这样IE 6就会强制重新计算，从而得到正确的结果，如图9.18所示。

图9.18　问题得到修正

⑥ 左边的一组菜单设置完成了，接下来把右边3组的HTML同样设置即可，这里就不再重复代码了。读者可以参考光盘中这个步骤完成后的“第9章/final-6.htm”文件。

分析 虽然现在的代码在IE 6、IE 7和Firefox中都可以正确显示，但是从代码符合Web标准的角度出发，是不够“优雅”的。如果在微软公司新推出的网页制作软件Expression Web中编辑这个网页，就会对这里给出错误提示，告诉制作者在<a>标记中包含<table>标记是不符合XHTML标准的，如图9.19所示。

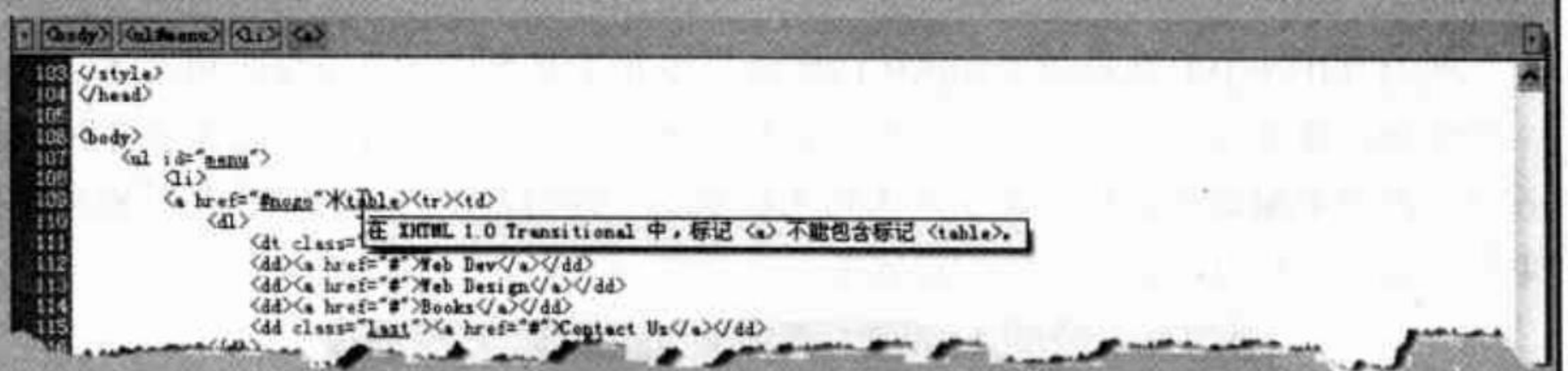


图9.19 在Expression Web软件中提示错误

如果希望更优雅地写这组代码，应该区分浏览器。对不同的浏览器使用不同的HTML代码，可以使用“条件注释”功能来实现。

例如，对于如下一行HTML代码，在IE浏览器中，会根据版本进行选择。

```
<!--[if lte IE 6]><table><tr><td><![endif]-->
```

“if”的意思是“如果”，“lte”是“less than or equal”的缩写，也就是低于或等于IE 6时，后面的“<table><tr><td>”这段代码就有效；如果不满足“低于或等于IE 6”这个条件，“<table><tr><td>”这段代码就会被忽略。

这组特定格式的“条件注释”只有IE浏览器认识，其他浏览器（例如Firefox）根本不认识，会当作普通的注释直接忽略掉。因此对于IE 7或者Firefox浏览器，这行代码就像不存在一样。理解了上述原理之后，就可以对这段代码进行如下修改：

```
<li>
<!--[if lte IE 6]><a href="#nogo"><table><tr><td><![endif]-->
        <dl>
                <dt class="orange"><a href="#">Artech Studio</a></dt>
                <dd><a href="#">Web Dev</a></dd>
                <dd><a href="#">Web Design</a></dd>
                <dd><a href="#">Books</a></dd>
                <dd class="last"><a href="#">Contact Us</a></dd>
        </dl>
<!--[if lte IE 6]></td></tr></table></a><![endif]-->
</li>
```

这样，对于IE 7和Firefox浏览器，就像没有做改造前一样，而对于IE 6流览器，就是改造后的代码，这样就很完美了。读者可以参考光盘中这个步骤完成后的“第9章/final-7.htm”文件。

9.10 本章小结

在本章中，制作了一个比较复杂的可以实现鼠标指针经过时，打开子菜单的导航条。对于大多数网站，两级的下拉菜单已经可以满足绝大多数需要了。使用CSS，也可以制作更多级的菜单。应用多级菜单的网站目前很少见到，因为网页是一个非常灵活的载体，与其使用更多级的菜单来完成复杂的导航，倒不如选用其他更好的形式。

例如，在Windows桌面应用软件中，多级菜单几乎是所有软件中的标准配置，然而在最新版的Office 2007中，令很多人惊奇的一点是，Word、Excel这样命令繁多的软件中，竟然不再有菜单了，取而代之的是一种称为“Ribbon”的命令选择方式，如图9.20所示。微软公司称其为“Ribbon”的工具条，就是在很多网页中使用的Tab面板。它把各种原来需要在多级下拉菜单中选择的命令都以图标的形式放置在各个Tab页上，各个Tab页之间可以自由切换。用户习惯于这种方式以后，恐怕再也不想用原来的多级菜单的命令选择方式了。

从10年前的Word 97开始，Office软件的用户界面就是众多软件模仿的目标，微软公司在软件的易用性上从来不惜花费大量的研发力量。可见，具体使用什么形式的用户界面和导航方式，并没有固定的标准，而是要根据实际软件和网页的需要来决定，使用最恰当的方式。

本书后面也会专门介绍如何制作非常流行的Tab面板，使读者在做网页的时候，有更多可以选择的手段，制作出最恰到好处的网页。

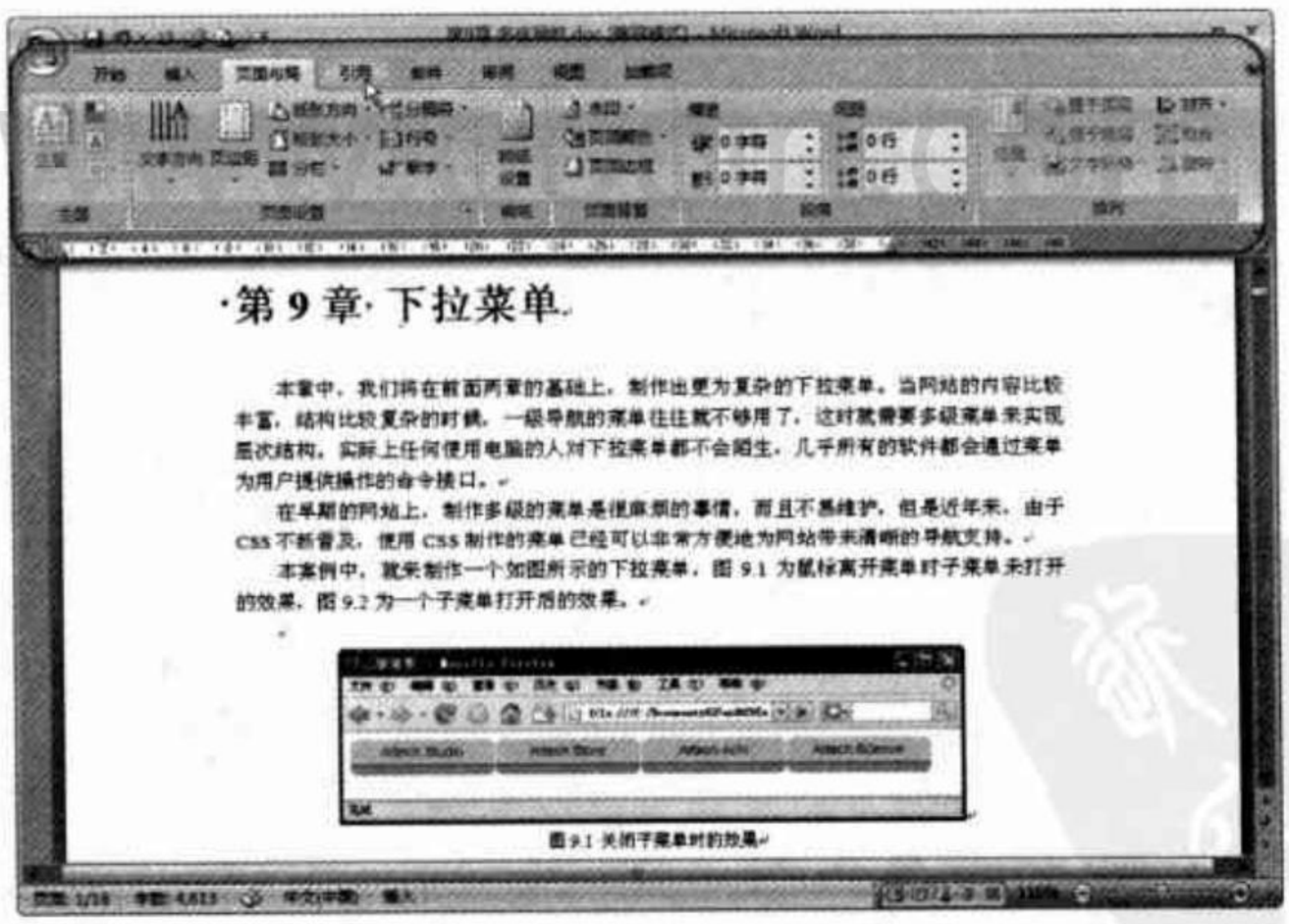
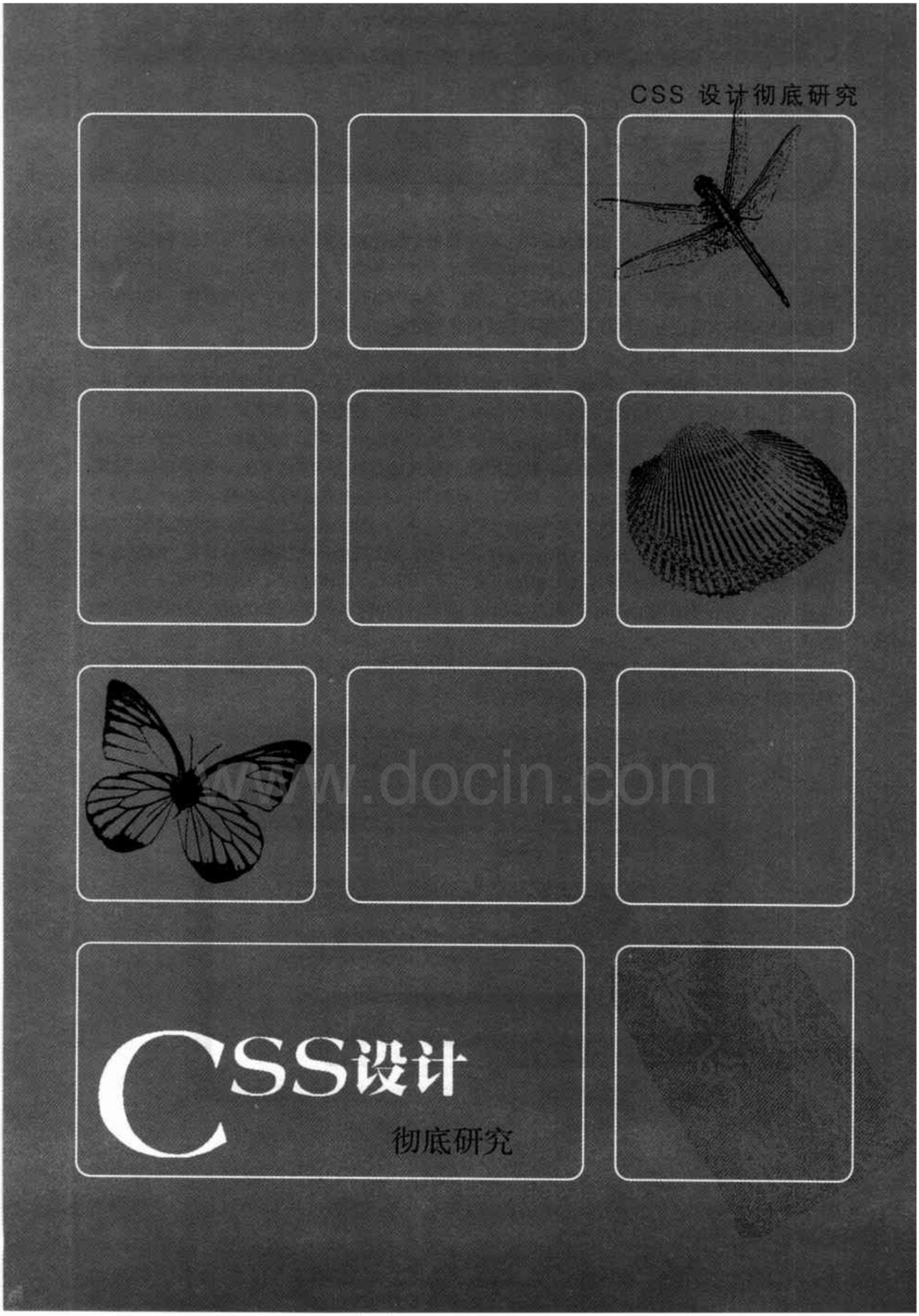


图9.20　“Ribbon”的命令选择方式



第10章 表格也精彩

表格是网页上最常见的元素。在传统的网页设计中表格除了显示数据外，还常常被用来作为整个页面布局的手段。在Web标准逐渐深入设计领域以后，表格逐渐不再承担布局的任务，但是表格仍然都在网页设计中发挥着重要的作用。

本章继续挖掘CSS的强大功能，让普普通通的表格也表现出精彩的一面。

10.1 控制表格

表格作为传统的HTML元素，一直受到网页设计者们的青睐。使用表格来表示数据、制作调查表等应用在网络中屡见不鲜。同时因为表格框架的简单、明了，使用没有边框的表格来排版，也受到很多设计者的喜爱。本节主要介绍CSS控制表格的方法，包括表格的颜色、标题、边框和背景等。

10.1.1 表格中的标记

在最初HTML设计时，表格（<table>标记）仅仅是用于存放各种数据的，例如收支表、成绩单等都适于用表格来组织数据形式。因此表格有很多与数据相关的标记，十分方便。

最常用的3个与表格相关的标记是<table>、<tr>和<td>。例如，一个最简单的表格如下，这是一个3行3列的简单表格。

```
<table>
  <tr>
        <td>32</td>  <td>17</td>  <td>14</td>
  </tr>
  <tr>
        <td>28</td>  <td>16</td>  <td>15</td>
  </tr>
  <tr>
        <td>16</td>  <td>22</td>  <td>12</td>
  </tr>
</table>
```


其中，<table>用于定义整个表格，<tr>定义一行，<td>定义一个单元格。此外，还有两个标记也是比较常用的，尤其使用CSS可以灵活设置表格样式以后，这两个标记就更常用到。

（1）<caption>标记，它的作用跟它的名称一样，就是用于定义表格的大标题，该标记可以出现在<table>与</table>之间的任意位置，不过通常习惯放在表格的第1行，即紧接着<table>标记。设计者同样可以使用一个普通的行来显示表格的标题，但<caption>标记无论是对于好的编码习惯，还是搜索引擎而言，都更有优势。

（2）<th>标记，它是“table header”的缩写，即表头的意思。在表格中主要用于行或者列的名称，行和列都可以使用各自的名称。实际上<th>和<td>是很相似的，主要是可以分别对它们进行设置样式。

下面是一个用到上面5个标记的表格实例，代码如下。实例文件位于本书光盘的“第10章\01\medal-begin.htm”。

```
<!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=gb2312" />
<title>奖牌榜</title>
</head>
<body>
<table border="2" cellpadding="10" cellspacing="10" bgcolor="#eeeeee">
    <caption>中国健儿奥运奖牌榜</caption>
  <tr>
        <th>年份</th> <th>金牌</th> <th>银牌</th> <th>铜牌</th> <th>总计
</th>
    </tr>  <tr>
        <th>2004</th> <td>32</td> <td>17</td> <td>14</td> <td>63</td>
    </tr>
    <tr>
        <th>2000</th> <td>28</td> <td>16</td> <td>15</td><td >59</td>
    </tr>
    <tr>
        <th>1996</th> <td>16</td> <td>22</td> <td>12</td> <td>50</td>
    </tr>
    <tr>
        <th>1992</th> <td>16</td> <td>22</td> <td>16</td> <td>54</td>
    </tr>
    <tr>
        <th>1988</th> <td>5</td> <td>11</td> <td>12</td><td >28</td>
    </tr>
    <tr>
        <th>1984</th> <td>15</td> <td>8</td> <td>9</td> <td>32</td>
  </tr>
</table>
</body>
</html>
```


这个页面的显示效果如图10.1所示。

图10.1 基本的表格样式

分析 这个实例中，没有使用任何CSS样式，而是使用了HTML中规定的设置表格的一些属性，例如在上面的代码中有如下一行：

```
<table border="2" cellpadding="10" cellspacing="10"
bgcolor="#eeeeee">
```

这里border属性用于表格边框，bgcolor用于设定背景色，cellpadding和cellspacing的作用如图10.2所示。

图10.2 cellspacing和cellpadding的含义

在CSS被广泛使用之前，大都使用上述这些属性来设置表格的样式，但是控制能力非常弱，而使用CSS以后，就可以更精确灵活地控制表格的外观了。

10.1.2 表格的边框

本案例中，仍然使用上面奖牌榜的数据，通过CSS来对表格样式进行设置。首先在原来的代码中删除使用的HTML属性，然后为table设置一个类别“ranking”，并进行如下设置。实例文件位于本书光盘的“第10章\01\medal.tm”。


```
<style type="text/css">
.ranking{
    font: 14px 宋体;
 border:2px orange solid;
 text-align:center;
}

.ranking td{
 border:1px orange dashed;
}
.ranking th{
 border:1px orange solid;
}
</style>
```

此时效果如图10.3所示。最外面的粗线框是整个表格边框，里面每个单元格都有自己的边框，th和td可以分别设置各自的边框样式，例如这里th为1像素的实线，td为1像素的虚线。

图10.3　设置表格的框线

可以看到此时每个单元格之间都有一个的空隙，那么有没有办法消除这个缝隙，并设置1像素宽的分割线呢?

经验　有比较丰富网页制作经验的人知道，在不使用CSS时，要制作“1像素分割线表格”，是很需要一些“技巧”的，仅仅通过将cellspacing和border设置为1，并不能真正得到单元格之间的分隔线为1像素，因为相邻单元格各有一条边框，即使都设置为1像素，且它们之间的距离为0，它们并在一起时也会有2像素宽。为此，聪明的设计师想出了一个变通的办法。

把整个表格的背景色设置为希望的边框颜色，给每个单元格设置另一颜色作为单元格的背景色，然后把边框粗细设置为0，cellspacing设置为1，这样在cellspacing的地方露出来的背景色看起来就产生了1像素粗细的边框分隔线的效果。必须承认想出这个办法的设计师确实智商很高。

然而这毕竟是一个hack的办法，并不是HTML规范本身的真正用意。而使用CSS，就可以名正言顺地制作出真正的“1像素分割线表格”了。

1. 设置单元格的边框

CSS提供了两种完全不同的方法来设置单元格的边框。一种用于在独立的单元格中设置分离的边框，另一种适合设置从表格一端到另一端的连续边框。在默认情况下，使用上面讲到的“分离边框”，也就是在上面的表格中看到的效果，相邻的单元格有各自的边框。

而如果在上面的例子中，在“.ranking”的设置中增加一个属性设置：

```
border-collapse: collapse;
```

其他不做任何改变，效果将变成如图10.4所示的样子，可以看到相邻单元格之间原来的两条边框重合为一条边框了。

图10.4　表格框线的重合模式

2. 相邻边框的合并

在图10.4中，我们又会发现一个问题，每个单元格都可以设置各自的边框颜色、样式和宽度等属性，那么相邻边框在合并时将以谁为准呢？例如在上面的例子中可以看到th的实线和td的虚线合并的时候，浏览器选择了th的实线。那么这里的规则是什么样的呢？

在CSS 2的规范中的定义如下。

（1）如果边框的“border-style”设置为“hidden”，那么它的优先级高于任何其他相冲突的边框。任何边框只要有该设置，其他的边框的设置就都将无效。

（2）如果边框的属性中有“none”，那么它的优先级是最低的。只有在该边重合的所有元素的边框属性都是“none”时，该边框才会被省略。

（3）如果重合的边框中没有被设置为“hidden”的，并且至少有一个不是“none”，那么重合的边框中粗的优先于细的。如果几个边框的“border-width”相同，那么样式的优先次序由高到低依次为“double”、“solid”、“dashed”、“dotted”、“ridge”、“outset”、“groove”、“inset”。

（4）如果边框样式的其他设置均相同，只是颜色上有区别，那么单元格的样式最优先，然后依次是行、行组、列、列组的样式，最后是表格的样式。

不过IE浏览器还没有完全执行上面这个规范的规定。在CSS 2的规范中，给出了一个明确的演示。下面的代码来自于CSS 2规范。

```
<HTML>
<HEAD>
<STYLE>
  TABLE{border-collapse: collapse;
        border: 5px solid yellow; }
  *#col1 { border: 3px solid black; }
  TD   { border: 1px solid red;
         padding: 1em;
   }
  TD.solid-blue  { border: 5px dashed blue; }
  TD.solid-green { border: 5px solid green; }
</STYLE>
</HEAD>
<BODY>
<TABLE>
<COL id="col1"><COL id="col2"><COL id="col3">
<TR id="row1">
    <TD> 1
    <TD> 2
    <TD> 3
</TR>
<TR id="row2">
    <TD> 4
    <TD class="solid-blue"> 5
    <TD class="solid-green"> 6
</TR>
<TR id="row3">
    <TD> 7
    <TD> 8
```

```
    <TD> 9
</TR>
<TR id="row4">
    <TD> 10
    <TD> 11
    <TD> 12
</TR>
<TR id="row5">
    <TD> 13
    <TD> 14
    <TD> 15
</TR>
</TABLE>
</BODY>
</HTML>
```

在图10.5中，左图为CSS 2规范中给出的正确显示效果，中间图为在Firefox中的实际显示效果，右图为在IE 6（IE 7与IE 6相同）中的显示效果。可以看到，Firefox的显示效果严格符合CSS 2规范，而IE浏览器则没有完全遵守CSS 2规范。

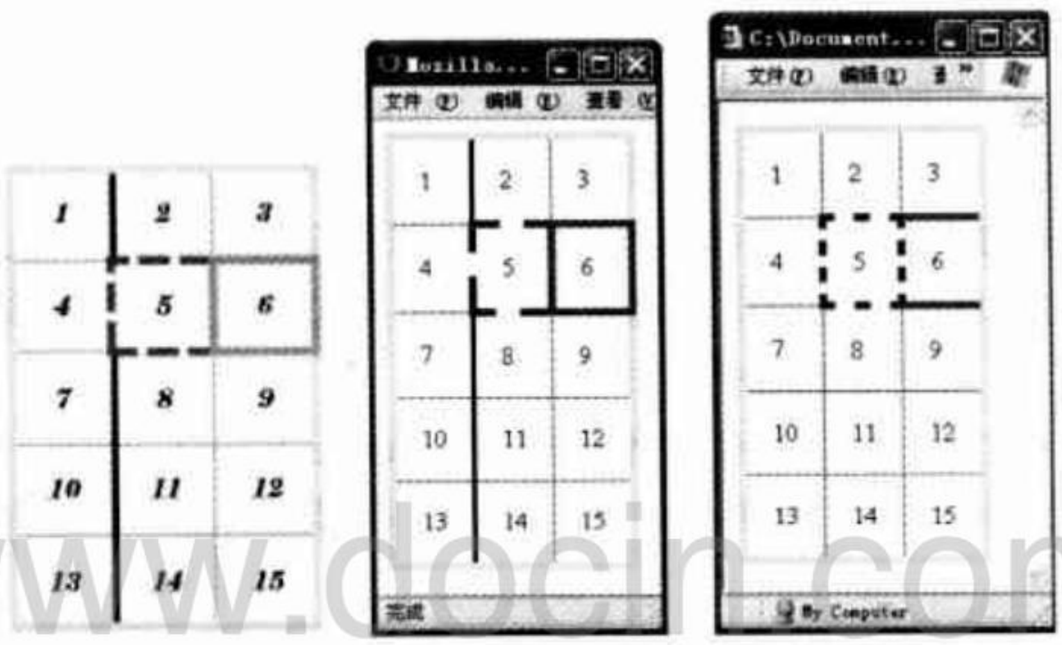


图10.5 重合模式下表格框线的优先级

3. 边框的分离

讲完边框的合并之后，再来补充说明一个边框分离的问题。前面讲到过，在使用HTML属性格式化表格时可以通过使用cellpadding来设置单元格内容和边框之间的距离，以及使用cellspacing设置相邻单元格边框之间的距离。

要用CSS实现cellpadding的作用，只要对td使用padding就可以了；而要用CSS实现cellspacing的作用时，对单元格使用margin是无效的，需要对table使用另一个专门的属性“border-spacing”来代替它。例如，在上面的代码的奖牌榜页面中，在“.ranking”中增加一条样式设置：

```
    border-spacing:10px;
```

在Firefox中的效果如图10.6所示。

遗憾的是，IE 6和IE 7都不支持这个属性，因此如果希望精确地控制相邻边框之间的距离，又能够适用于各种浏览器，目前还只能使用HTML的cellspacing属性来实现，它是目前关于表格的所有HTML属性中惟一还需要用到的属性。

图10.6 框线分离模式下设置边框之间的距离

10.1.3 表格宽度的确定

CSS提供了两种确定表格以及内部单元格宽度的方式。一种与表格内部的内容相关，称为“自动方式”；一种与内容无关，称为“固定方式”。

使用了自动方式时，实际宽度可能并不是“width”属性的设置值，因为它会根据单元格中的内容多少进行调整。而在固定方式下，表格的水平布局不依赖于单元格的内容，而明确地由“width”属性指定。如果取值为“auto”就意味着使用“自动方式”进行表格的布局。

在两种模式下，各自如何计算布局宽度是一个比较复杂的逻辑过程。对于一般用户来说，不需要精确地掌握它，但是知道有这两种方式是很有用的。

在无论各列中的内容有多少，都要严格保证按照指定的宽度显示时，可以使用“固定方式”。例如在后面的“日历”排版中，就用到了固定方式。反之，对各列宽度没有严格要求时，用“自动方式”可以更有效地利用页面空间。

如果要使用固定方式，就需要对表格设置它的“table-layout”属性。将它设置为“fixed”即为固定方式；设置为“auto”时则为自动方式。浏览器默认使用自动方式。

10.1.4 其他与表格相关的标记

除了前面介绍的标记之外，HTML中还有3个标记<thead>、<tbody>和<tfoot>，它们用来定义表格的不同部分，如图10.7所示。

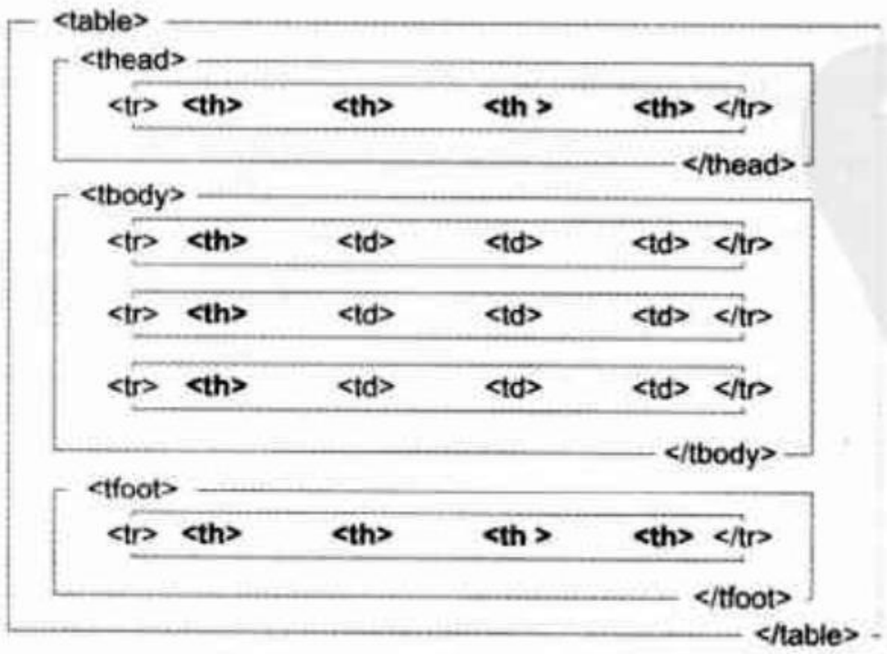


图10.7 表格的HTML结构示意

要使用CSS来格式化表格时，通过这3个标记可以更方便地选择要设置样式的单元格。例如，对在<thead>、<tbody>和<tfoot>中的<th>设置不同的样式，如果使用下面这个标记：

```
tbody th{……}
```

将只对<tbody>中的内容产生作用，这样就不用再额外声明类别了。

在HTML中，单元格是存在于“行”中的，因此如果要对整列设置样式，就不像设置行那么方便，这时可以使用<col>标记。

例如，一个3行3列的表格，如果要将第3列的背景色设置为灰色，可以使用如下代码：

```
<table >
<col></col><col></col><col class="special"></col>
  <tr>
        <td>11</td>
        <td>12</td>
        <td>13</td>
  </tr>
……以下省略……
```

每一对“<col></col>”标记对应于表格中的一列，对需要单独设置的列设置一个类别，然后设置该类别的CSS即可。

注意 由于一个单元格既属于某一行，又属于某一列，因此很可能行列各自的CSS设置都会涉及该单元格，这时以谁的设置为准，就要根据CSS的优先级来确定，如果有些规则非常复杂，制作的时候就要实际试验一下，但是需要特别谨慎。

10.2 美化表格

本案例中，我们对一个简单的表格进行设置，使它看起来更为精致。另外，当表格的行和列都很多，并且数据量很大的时候，为避免单元格采用相同的背景色会使浏览者感到凌乱，发生看错行的情况，为表格设置隔行变色的效果，使得奇数行和偶数行的背景颜色不一样。实例的最终效果如图10.8所示。

实例文件位于本书光盘的“第10章\02\pretty-1.htm”。

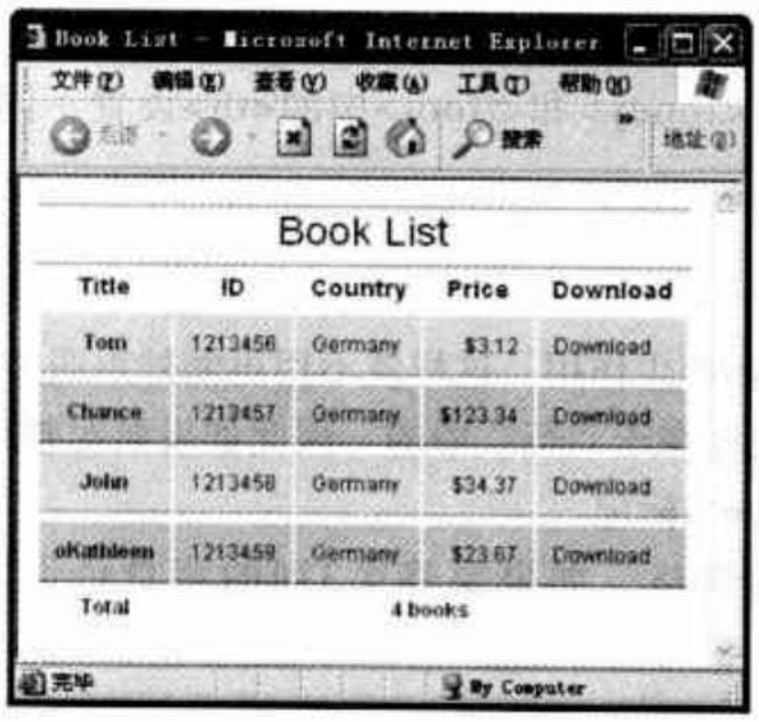


图10.8 交替变色的表格样式

10.2.1 搭建HTML结构

首先确定表格的HTML结构，代码如下：

```
<table summary="book list">
    <caption>Book List</caption>
    <thead>
        <tr>
            <th>Title</th>
            <th>ID</th>
            <th>Country</th>
            <th>Price</th>
            <th>Download</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Tom</th>
            <td>1213456</td>
            <td>Germany</td>
            <td>$3.12</td>
            <td>Download</td>
        </tr>
        ……这里省略3行……
        </tbody>
    <tfoot>
        <tr>
            <th>Total</th>
            <td colspan="4">4 books</td>
        </tr>
    </tfoot>
</table>
```

这个表格中，使用的标记从上至下依次为<caption>、<thead>、<tbody>和<tfoot>。此时在浏览器中的效果如图10.9所示。

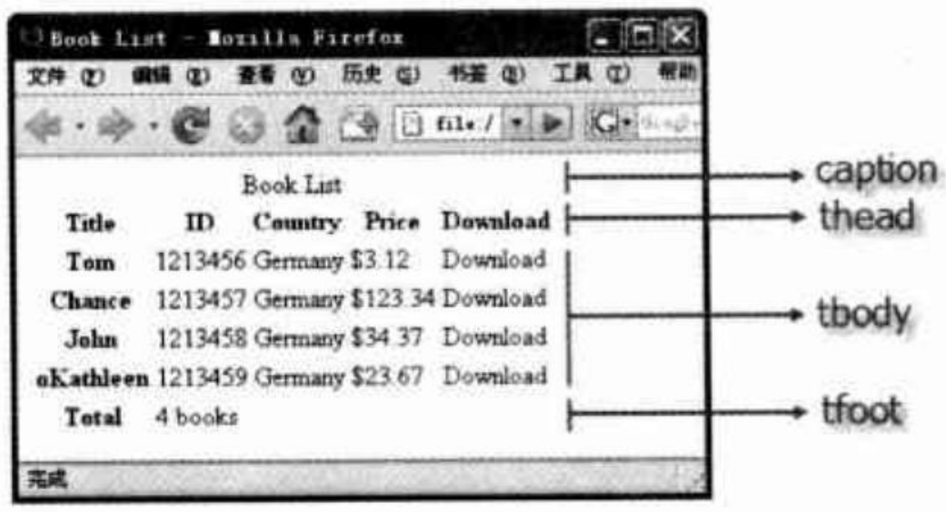


图10.9 交替变色的表格样式

10.2.2 整体设置

接下来对表格的整体和标题进行设置，代码如下：

```
table {
  background-color: #FFF;
  border: none;
  color: #565;
  font: 12px arial;
  text-align:left;
}

table caption {
  font-size: 24px;
  border-bottom: 2px solid #B3DE94;
  border-top: 2px solid #B3DE94;
}
```

此时的效果如图10.10所示，可以看到整体的文字样式和标题的样式已经设置好了。

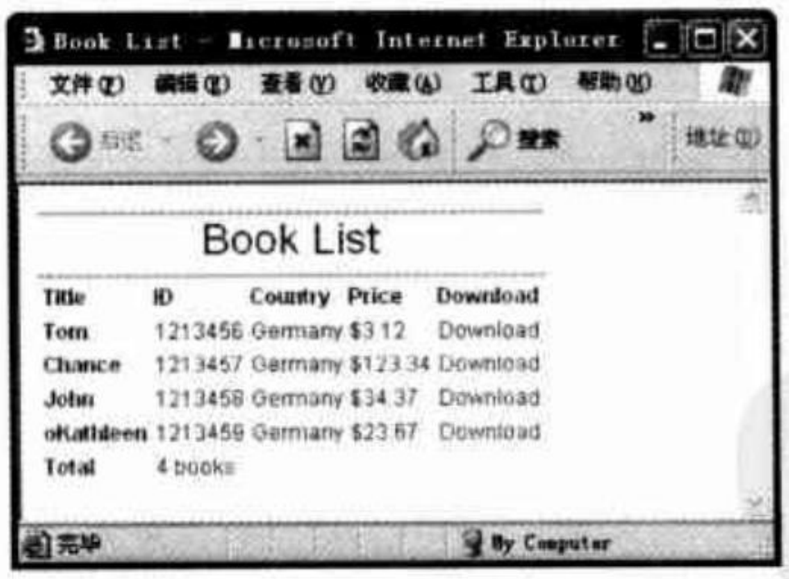


图10.10 设置部分属性的表格样式

10.2.3 设置单元格样式

现在来设置各单元格的样式，代码如下。代码一共分为4段，第1段是设置所有单元格的共性属性，后面3段分别对thead、tbody和tfoot的单元格样式进行设置。

```
table, td, th {
  margin: 0;
  padding: 0;
  vertical-align: middle;
}
tbody td, tbody th {
  background-color: #DFC;
  border-bottom: 2px solid #B3DE94;
  border-top: 3px solid #FFFFFF;
  padding: 9px;
}
tfoot td, tfoot th {
  font-weight: bold;
  padding: 4px 8px 6px 9px;
  text-align:center;
}
thead th {
  font-size: 14px;
  font-weight: bold;
  line-height: 19px;
  padding: 0 8px 2px;
  text-align:center;
}
```

此时的效果如图10.11所示。

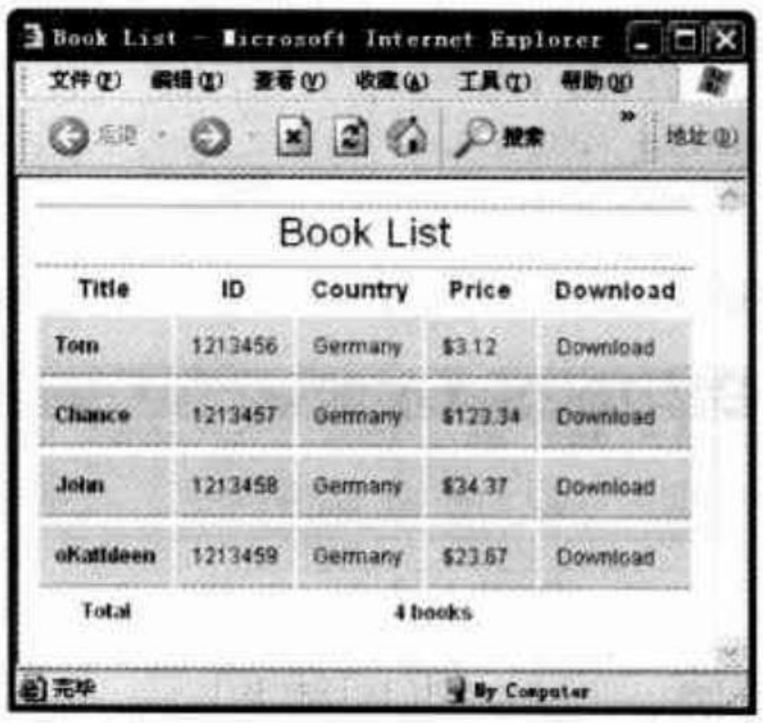


图10.11 设置了单元格的样式

10.2.4 隔行变色

然后就要使数据内容的背景色深浅交替，实现隔行变色。在CSS中实现隔行变色的方法十分简单，只要给偶数行的<tr>标记都添加上相应的类型，然后对其进行CSS设置即可。

① 首先，在HTML中，给所有偶数行的<tr>标记增加一个“even”类别，如下所示：

```
<tr class="even">
```

```
            <th >Chance</th>
            <td>1213457</td>
            <td>Germany</td>
            <td>$123.34</td>
            <td>Download</td>
        </tr>
```

② 设置“.even”与其他单元格的不同的样式，代码如下所示。

```
tbody tr.even th,tbody tr.even td {
  background-color: #CEA;
  border-bottom: 2px solid #67BD2A;
}
```

此时效果如图10.12所示，这里交替的两种颜色应该保证协调美观。

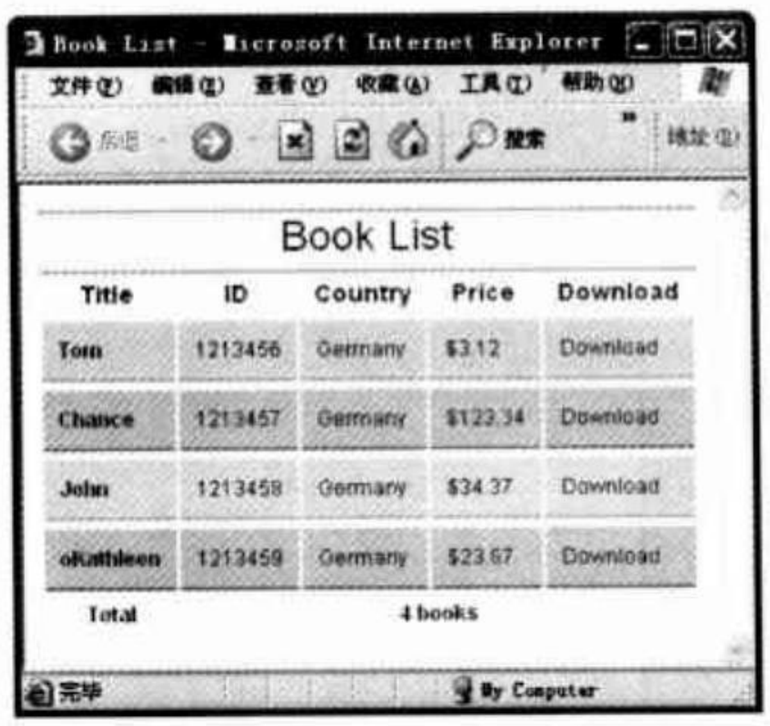


图10.12　交替变色的样式

10.2.5 设置列样式

对列做一些细节设置。例如，在price列中的数据是价格数值，如果能够右对齐，则更方便访问者理解。因此在HTML中thead的前面增加如下代码：

```
                    <col ></col><col></col><col></col><col class="price"
></col><col></col>
```

其中一对<col></col>对应一列，价格列中设置一个price类别。然后在样式中增加如下代码：

```
  col.price{
    text-align:right;
  }
```

到这里，这个案例就完成了，效果如图10.13所示。

图10.13 让price列右对齐

注意 这个案例的所有效果图都是用IE浏览器演示的。Firefox浏览器没有实现对列（即<col>）的支持，已经有很多人向Firefox的开发团队汇报了这个问题，但是至今没有修正，其原因不得而知。在Firefox中的效果如图10.14所示。

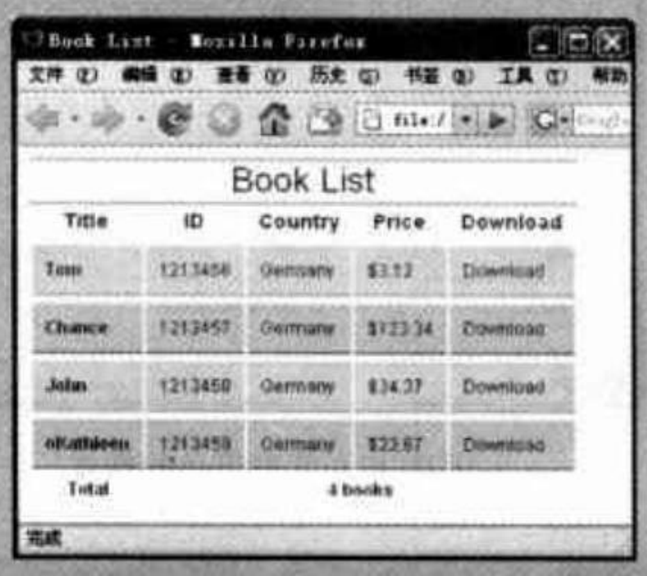


图10.14 Firefox不支持“列”属性

在Firefox中实现对整列样式的设置有两种办法。简单的办法是为这一列的每一个单元格设置单独的属性，当然这样做很不简洁。简洁的办法是使用“邻接”选择器，这种选择前面没有介绍过，它是CSS 2中提出的，在IE 6中不支持，使用方法是将代码：

```
col.price{
  text-align:right;
}
```

修改为：

```
td+td+td ,
col.price{
  text-align:right;
}
```

用加号连接的选择器就称为邻接选择器，上面用两个加号，连接3个td，就表示每一行

的第3个td为选中的元素。注意表格中第1列是th，因此“price”这一列是td的第3列。这时在Firefox中的效果如图10.15所示。

图10.15　在Firefox中使用邻接选择器

注意　（1）在IE 6中，不支持邻接选择器，在IE 7中已经支持了。

（2）在实际网页中，这种隔行变色的效果通常是配合服务器动态生成的，在服务器上读取数据的时候做判断，读第1个数据的时候输出“<tr>”，读第2个数据的时候输出"<tr class="even">"，然后依次循环。

10.3 鼠标指针经过时整行变色提示的表格

近年来，Web 2.0的概念逐渐被广泛接受，其中很重要的一点是强调改善用户体验，例如10.2节的例子中，把表格设置为交替背景色，可以使访问者在浏览表格时有更好的体验。

然而对于长时间审核大量数据和浏览表格的用户来说，即使是隔行变色的表格，长时间阅读这样的表格仍然会感到疲劳。而且对于数据量很大的表格，特别容易看错行或者列。如果参考微软公司Excel软件的做法，我们就可以发现很多可以改进的地方。

例如，在一个行列都很多的表格中，如果能像在Excel中那样，随时以高亮的方式提示一个单元格对应的行号（或行的名称）和列号（或列的名称），就会大大改进浏览者的体验了，如图10.16所示。

图10.16　Excel中对选中单元格的行列提示

在本节的案例中先实现一个比较容易的效果，当鼠标指针经过表中的某一个单元格时，该单元格所在的行能够动态地变色，如图10.17所示，这样就会大大减少访问者看错行的可能性。这实现起来比横竖两个方向的变色要容易一些。

实例文件位于本书光盘的“第10章\02\pretty-2.htm”。

图10.17 鼠标指针经过时数据行的背景变色

10.3.1 搭建HTML结构

仍然以上一节中制作的表格为例，由于本案例的重点是JavaScript的编写，因此先对表格进行化简，把表格设置为最基本的HTML结构，这样便于理解JavaScript的作用原理。把表头的标记全部换为通用的<td>标记，HTML代码如下。

```
<table summary="book list">
        <caption>Book List</caption>
                <tr >
                <td >Title</td>
                <td>ID</td>
                <td>Contry</td>
                <td>Price</td>
                <td>Download</td>
            </tr>
            ……省略4行……
    </table>
```

然后把相应的CSS设置也化简，只保留对table和td的基本设置，代码如下。

```
table {
  color: #565;
  font: 12px arial;
}

td {
  background-color: #DFC;
  border-bottom: 2px solid #B3DE94;
  border-top: 3px solid #FFFFFF;
  padding: 9px;
}
```

此时效果如图10.18所示。

图10.18　化简以后的表格样式

10.3.2 在Firefox和IE 7中实现鼠标指针经过时整行变色

下面使鼠标指针经过某一行的时候，该行的背景变色。对于Firefox浏览器而言，仅仅通过CSS的“:hover”伪类便可以实现该效果，添加如下代码：

```
tr:hover{
  background-color: #595;
  color:#fff;
}
```

这段设置的意图是，某一行在鼠标指针经过时，会使用“tr:hover”设置的背景色（#595）和文字颜色（#fff，白色）。然而，在Firefox中的效果如图10.19所示。

图10.19　只有文字颜色变化，背景色没有变化

可以看到，鼠标指针经过第3行时，该行只有文字的颜色变为了白色，而背景色没有改变，这是什么原因呢？

原因在于前面的样式设置中，背景色是在td中设置的，这里tr的:hove伪类的优先级低于td，因此背景色还是按照td的设置，将背景色的设置从td中移出来，并增加到tr中，效果就会如图10.20所示了。

图10.20　实现了整行的背景变色

这个方法在Firefox和IE 7中都是有效的。IE 6浏览器不支持<tr>标记的“:hover”伪类别，因此下一小节介绍如何采用JavaScript动态的配合，实现同样的效果。

10.3.3 在IE 6中实现鼠标指针经过时整行变色

① 首先，在上面的CSS设置中增加一个“.hover”类选择器。

```
tr:hover, tr.hover{
  background-color: #595;
  color:#fff;
}
```

② 在</table>后添加JavaScript代码，以提取表格中的<tr>标记，并使每一个<tr>标记都动态增加相应的鼠标事件以及相应的函数，以实现当鼠标指针移动到某行上时，调用新的CSS类别使该行背景变色。

在</table>和</body>之间，增加JavaScript语句，代码如下。

```
……以上部分省略……
</table>
<script language="javascript">
var rows = document.getElementsByTagName('tr');
for (var i=0;i<rows.length;i++){
  rows[i].onmouseover = function(){          //鼠标指针在行上面的时候
        this.className = 'hover';
  }
  rows[i].onmouseout = function(){           //鼠标指针离开时
        this.className = '';
  }
  }
</script>
</body>
</html>
```

分析 “document.getElementsByTagName('tr')”的作用是取得一个数组，数组中的每个元素就是DOM树中的各个tr节点，这个数组存储在“rows”变量中。

然后用一个循环结构，对每一个tr节点的onmouseover事件增加处理函数，这个函数将会在该tr被鼠标指针经过的时候执行，这个函数的内容是：

```
this.className = "hover";
```

这里的this就是代表该节点本身，因此该语句的含义是，使得该节点的CSS类名为hover。这个类别前面在CSS部分已经设置了，因此就实现了鼠标指针经过时，背景色发生改变。

接下来就要使鼠标指针离开的时候，恢复原来的背景色，这需要为onmouseover事件（也就是鼠标指针离开）增加处理函数，内容是使类别名为空字符串，即清除了前面设置的“hover”，这样就恢复原来的背景色了。

10.4 Excel方式二维变色提示的表格

在10.3节中，整行变色提示鼠标指针经过的效果已经完成了，接下来继续改进它。实现类似于Excel的行列的二维提示，效果如图10.21所示，当鼠标指针经过某一个单元格时，相应的列头和行头单元格会同时变色。

实例文件位于本书光盘的“第10章\02\pretty-3.htm”。

图10.21　表格的行列二维变色提示

注意　本案例需要不少JavaScript编程的配合，如果读者缺乏相应基础，学起来可能会有一点困难。

10.4.1 改造CSS代码

首先改造CSS设置，这个效果单纯使用CSS是无法实现的，必须要使用JavaScript来实现，因此“tr:hover”选择器就可以删除了。

还应该将“tr.hover”改成“td.hover”，因为我们的目标是使鼠标指针经过的单元格，以及该单元格所在的行和列的第1个单元格的背景变色，而不是整行变色。把变色的对象设置为单元格，将下面的代码：

```
tr:hover, tr.hover{
  background-color: #595;
  color:#fff;
}
```

改为：

```
td.hover{
  background-color: #595;
  color:#fff;
}
```

10.4.2 改造JavaScript代码

接下来改造JavaScript代码。首先将下面这行代码中的“tr”修改为“td”，也就是从DOM树中提取出所有的td节点。

```
var rows = document.getElementsByTagName('tr');
```

这时效果可以想象出来，就是当鼠标指针经过某个单元格时，该单元格变色，效果如图10.22所示。

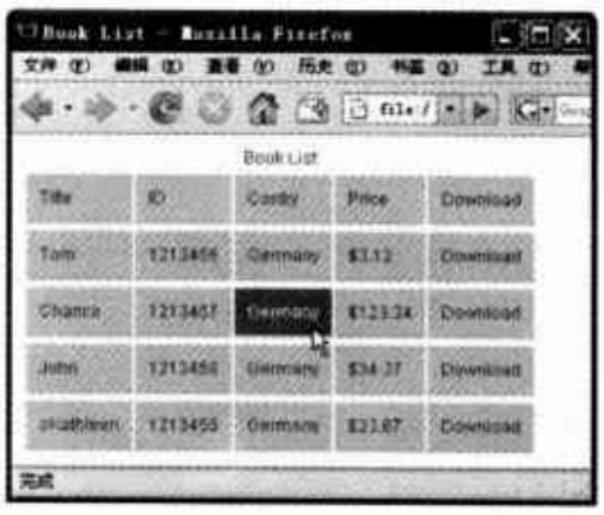


图10.22 鼠标指针经过的单元格变色

注意 为了养成良好的习惯，变量名称应该符合它的实际含义，因此将原来的rows变量改名为cells，row的中文意思是行，cell的中文意思是单元格。

如何使所在行和列的第1个单元格也变色呢？如果能够知道当前鼠标指针经过的行和列的编号（或者叫“坐标”），就可以计算出相应两个单元格的编号。在cells数组中存放着所有的单元格，因此可以尝试如下的代码：

```
var cells = document.getElementsByTagName('td');
for (var i=0;i<cells.length;i++){
  cells[i].onmouseover = function(){          //鼠标指针在行上面的时候
        cells[i].className = 'hover';
  }
  cells[i].onmouseout = function(){           //鼠标指针离开时
        this.className = '';
  }
}
```

在这段代码中，我们仅把第4行中的“this”换成了“cells[i]”。前面说过，“this”就代表该节点本身，这里直接用数组元素来表示，那么由该元素的编号就可以推算出其他元素的编号了。经过试验会发现，这样替换以后，鼠标指针经过时就不会有变色的效果了，如图10.23所示。

图10.23　鼠标指针经过的单元格不再变色

分析　这个问题出在哪里了呢？这需要对这段程序深入地理解。请读者思考，这段程序是在什么时候运行的？是在鼠标指针经过时，还是在页面装载时？

正确答案是后者，而且这段代码只执行一次。这样就找到原因了，当鼠标指针经过某单元格时，所运行的仅是修改className这一个操作，而此时代码中的循环早已结束，因此i变量的值应该等于所有单元格的总数，例如这里一共25个单元格，i应该等于25，而JavaScript中数组的元素编号从0开始，最后一个单元格的编号是24，cell[25]根本就不在，因此没有单元格变色就不奇怪了。

为了验证上面的分析，现在把第4行代码：

```
        cells[i].className = 'hover';
```

改为：

```
        cells[i-1].className = 'hover';
```

请读者猜一猜运行的结果是什么？答案是，无论鼠标指针进入哪个单元格，表格最右下角的一个单元格都会变色，这就证明了i-1恰好等于24。

有了上面的基础，我们就找到了解决的方法。修改完成后的JavaScript代码如下。

```
<script language="javascript">
var cells = document.getElementsByTagName('td');
for (var i=0;i<cells.length;i++){
  cells[i].onmouseover = function(){          //鼠标指针在行上面的时候
        this.className = 'hover';
        for (var j=0;j<cells.length;j++)
              if(cells[j]==this){
                    cells[j%5].className = 'hover';
                    cells[j-j%5].className = 'hover';
                                              }
}
cells[i].onmouseout = function(){           //鼠标指针离开时
        this.className = '';
        for (var j=0;j<cells.length;j++)
              if(cells[j]==this){
                    cells[j%5].className = '';
```

```
                cells[j-j%5].className = '';
                                    }
    }
}
```

代码中，在鼠标指针进入部分增加了5行代码（第6~10行）。其运行过程是首先要获得当前单元格的编号，方法是在cells数组中，从0号元素开始查找。如果某个元素等于“this”，它就是当前鼠标指针经过的单元格，此时循环变量记录的就是它的编号。“j%5”的含义是取得对j除以5后的余数，也就是该列的第一个单元格编号，同理“j-j%5”就是该行第一个单元格的编号。接下来就都是前面介绍过的内容了。

这时在浏览器中可以看到正确的效果，如图10.24所示。

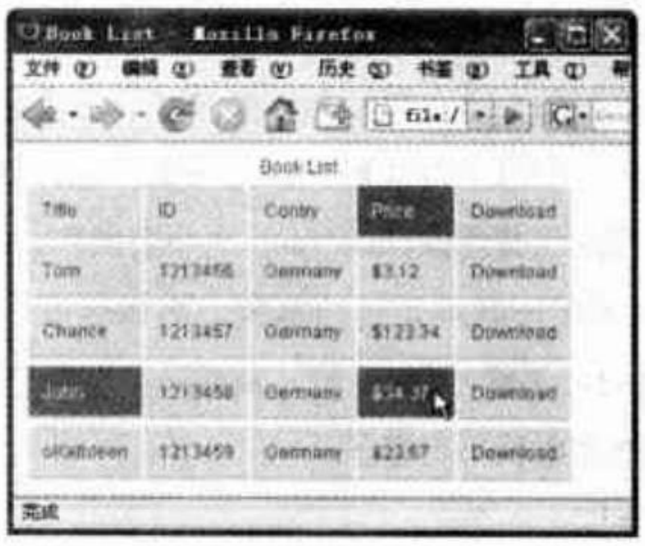


图10.24 表格的行列二维变色提示

本实例巧妙地采用JavaScript读取鼠标指针的状态，从而改变相应的3个单元格的CSS属性，来实现背景颜色的动态变化。

当然，这个例子还很简单，例如本案例为了简化，把所有单元格都用td来定义。如果保留th等标记，本例就需要进行一定的修改了。如果读者有兴趣，可以进一步深入学习JavaScript和DOM编程，这些效果都是可以实现的。利用JavaScript和DOM，可以完全使用编程的方式动态地产生和改变页面中的所有元素，因此任何能用HTML和CSS产生的内容，用编程的方式都能做到。

注意 使用JavaScript以后，或多或少都会影响网页的效率。当然仅做一些少量的运算不会对访问的效果造成任何影响，但是不要过分依赖JavaScript，凡事都是适度最好。

10.5 多视图模式日历案例概述

在接下来的几节中，我们将围绕“日历”这个案例进行实战演练，以提高读者对CSS的

掌握和理解深度。

日历是日常生活中随处可见的工具。计算机出现后，产生了很多供人们记录日程安排的备忘录软件。随着互联网的普及，将日历存储在互联网上就更方便了，无论走到哪里，只要能够登录互联网，就可以随时查询和登记各种日程信息。

Google前不久推出了功能非常强大的日历软件，它不但具有普通日历的功能，还和移动通信相结合，用户可以和手机绑定，到达设定的时间时，用户就会收到一条提示短信。“Google日历”是完全基于Web的应用程序，所有操作都在浏览器中完成，网站界面如图10.25所示。

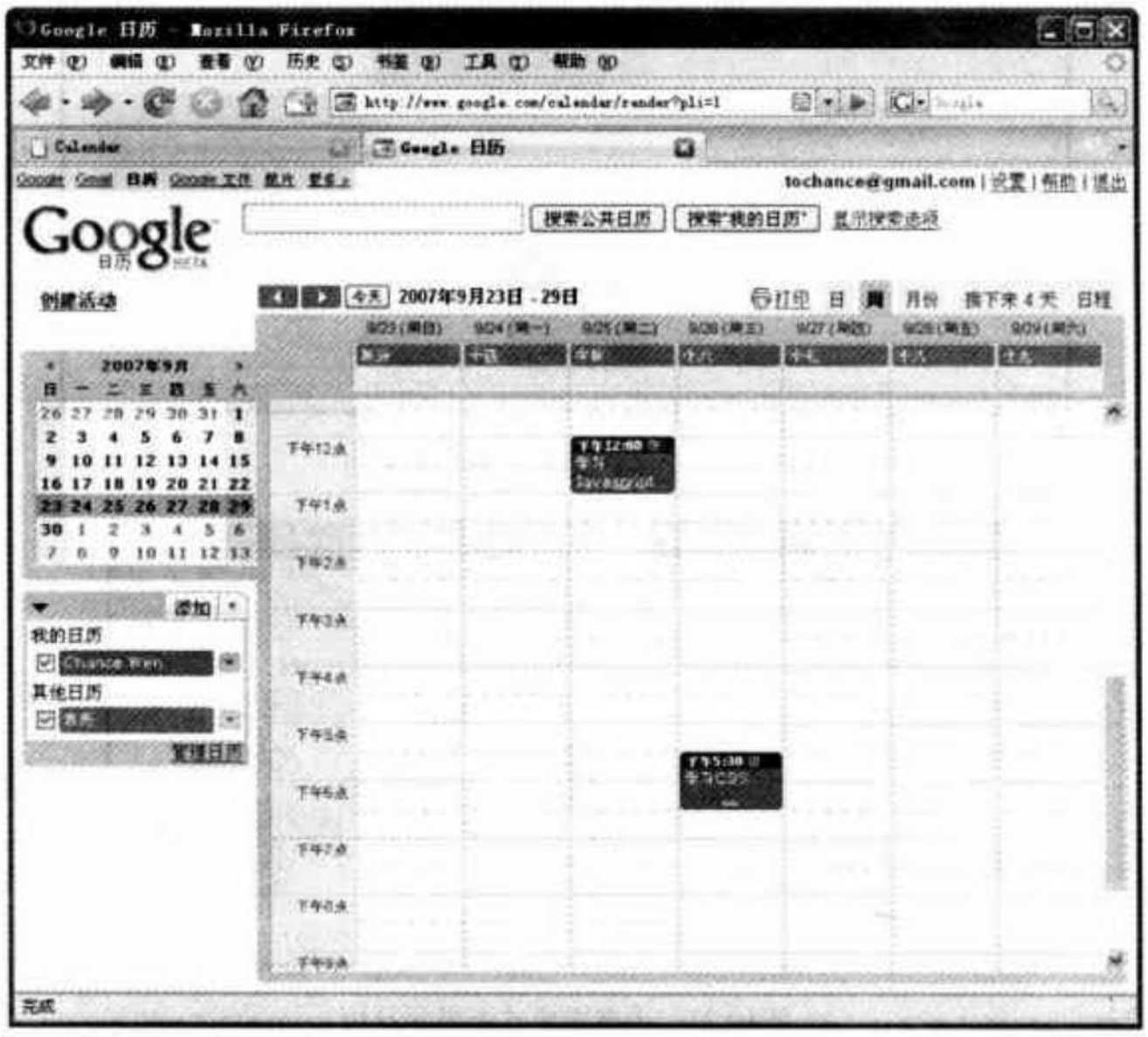


图10.25　Google日历

可以看到，Google日历提供了几种不同的视图模式，有的非常详细，可以查看每个小时的日程信息，而在比较粗略的月视图中，则可以查看整个月的安排情况。读者如果有兴趣，可以亲自试验一下Google日历的功能。

在这章中，我们也来实现一个日历的页面。为了更好地体现CSS的优点，在本案例中将通过不同的CSS设定对相同的日历表格实现3种显示模式，适于不同详细程度和大小的页面空间，三者之间可以方便地切换。为了便于讲解，把这3种模式分别称为小视图模式、中视图模式和大视图模式。中视图模式的效果如图10.26所示。

顶部显示年份和月份，并可以在3种视图模式之间切换。下面的表格中每个单元格放置一天的日程安排信息，同时可以设定每一个安排的重要性。普通的安排显示为橙色背景，重要安排显示为紫色背景。

当切换到小视图模式时，所有的详细安排信息都被隐藏起来，如图10.27左图所示，这样就可以非常节约空间。当鼠标指针经过某个放置了日程安排的单元格时，会在鼠标指针右下方出现一个方框，显示这一天的日程安排，如图10.27右图所示。

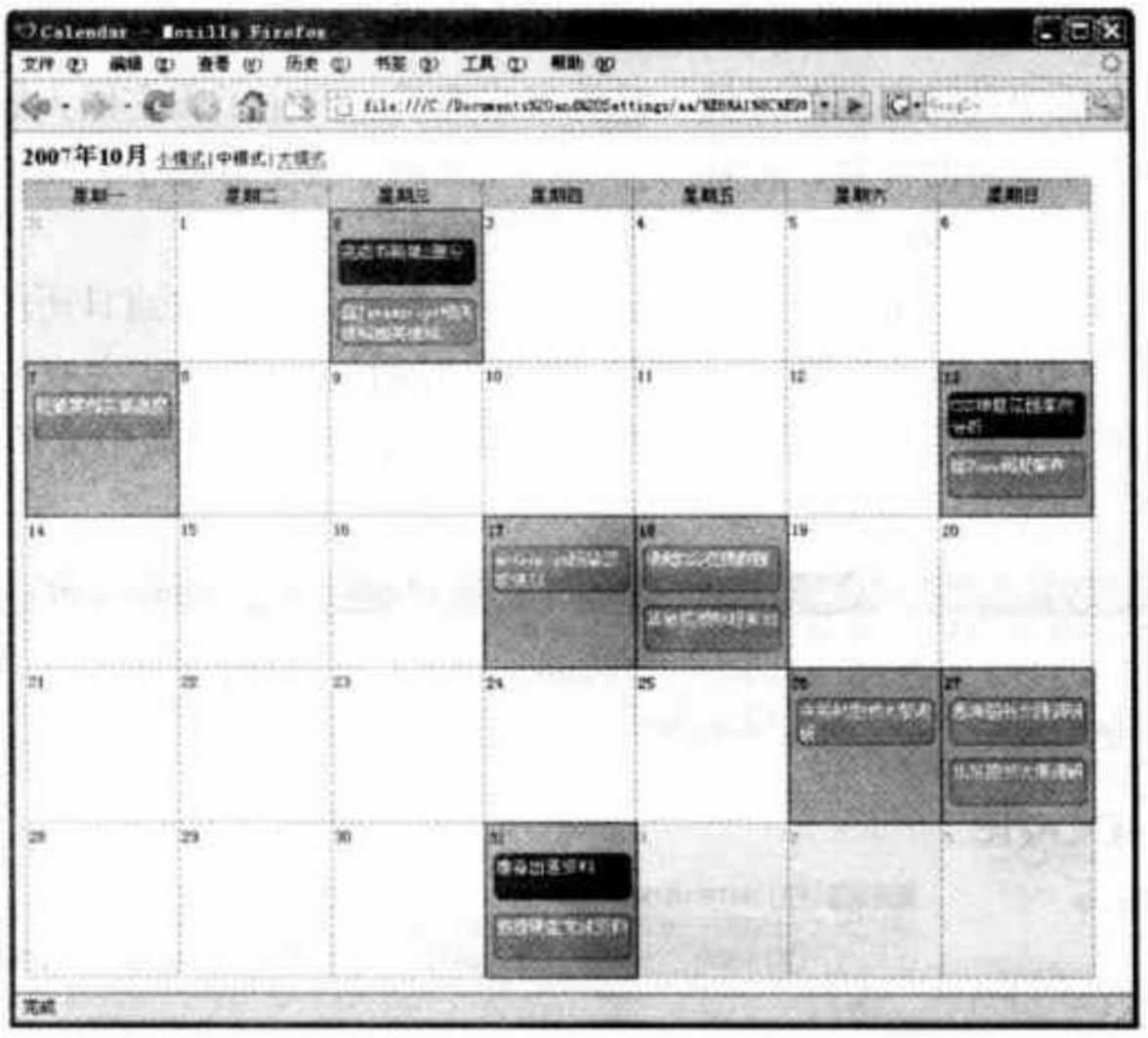


图10.26 中视图模式显示的日历

图10.27 小视图模式下显示日历

当切换到大视图模式时，效果如图10.28所示。

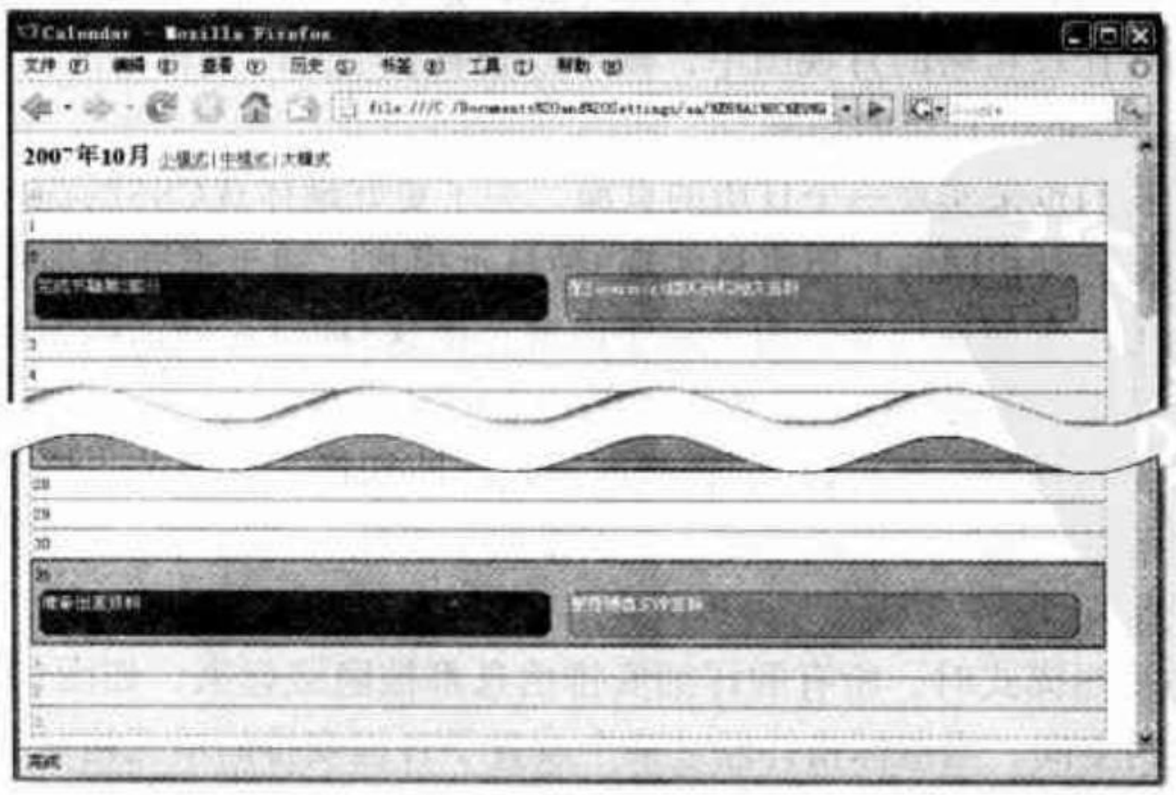


图10.28 大视图模式下显示日历

在中模式中，每条日程信息只显示前面的若干文字，后面的会被隐藏起来，而在大模式中可以显示更多的文字。

10.6 制作中视图模式

首先从中模式开始制作，制作好中模式以后，再以它为基础制作另外两个模式的页面。中视图模式的日历实例文件位于本书光盘的“第10章\03\calendar-normal.htm”。

10.6.1 搭建HTML结构

按照传统的方法建立最简单的表格，包括建立表格的标题<caption>，以及利用<th>表示星期一到星期日，并给表格定义CSS类别，如下所示。在caption部分除了“2007年10月”这个标题之外，还有用于切换模式的文字链接。

```
<table class="month" summary="Calendar for 2007.10">
    <caption>
        <span class="date">2007年10月 </span>
        <a href="calendar-small.html">小模式</a> |
        <a>中模式</a> |
        <a href="calendar-large.html">大模式</a>
    </caption>
    <tr>
        <th scope="col"><span>星期</span>一</th>
        <th scope="col"><span>星期</span>二</th>
        <th scope="col"><span>星期</span>三</th>
        <th scope="col"><span>星期</span>四</th>
        <th scope="col"><span>星期</span>五</th>
        <th scope="col"><span>星期</span>六</th>
        <th scope="col"><span>星期</span>日</th>
</tr>
```

这里需要解释的是，在“星期一”到“星期日”这7个文字中，都分别把“星期”两个字放到span中了，这样做的目的是将来在小视图模式中，可以在不修改HTML代码的情况下通过CSS把所有的“星期”两字隐藏起来，效果参考图10.27。

每天的日程放在具体的单元格中，并且定义各种CSS类型。previous和next分别表示上个月和下个月的日期，设置灰色背景和灰色日期文字以和当月的日期区分开；active用来表示有具体安排的日子，对于重要的日程安排，在li中设置importan类别，以便后期用CSS做特殊样式。示例代码如下：

```
    <tr>
        <td class="previous">31</td>
        <td>1</td>
```

```
        <td class="active">2
            <ul>
                <li class="important">完成书稿第2部分</li>
                <li>查JavaScript相关资料相关资料</li>
            </ul>
        </td>
        <td>3</td>
        <td>4</td>
        <td>5</td>
        <td>6</td>
    </tr>
```

上面的代码中，表格每行包含7个单元格。对于没有安排的单元格，仅输入一个日期数字即可；对于有安排的单元格，用ul列表排列各项日程安排。

依次建立好整个日历表格后，就可以开始加入CSS属性控制其样式风格了。此时还没有CSS控制的日历如图10.29所示。

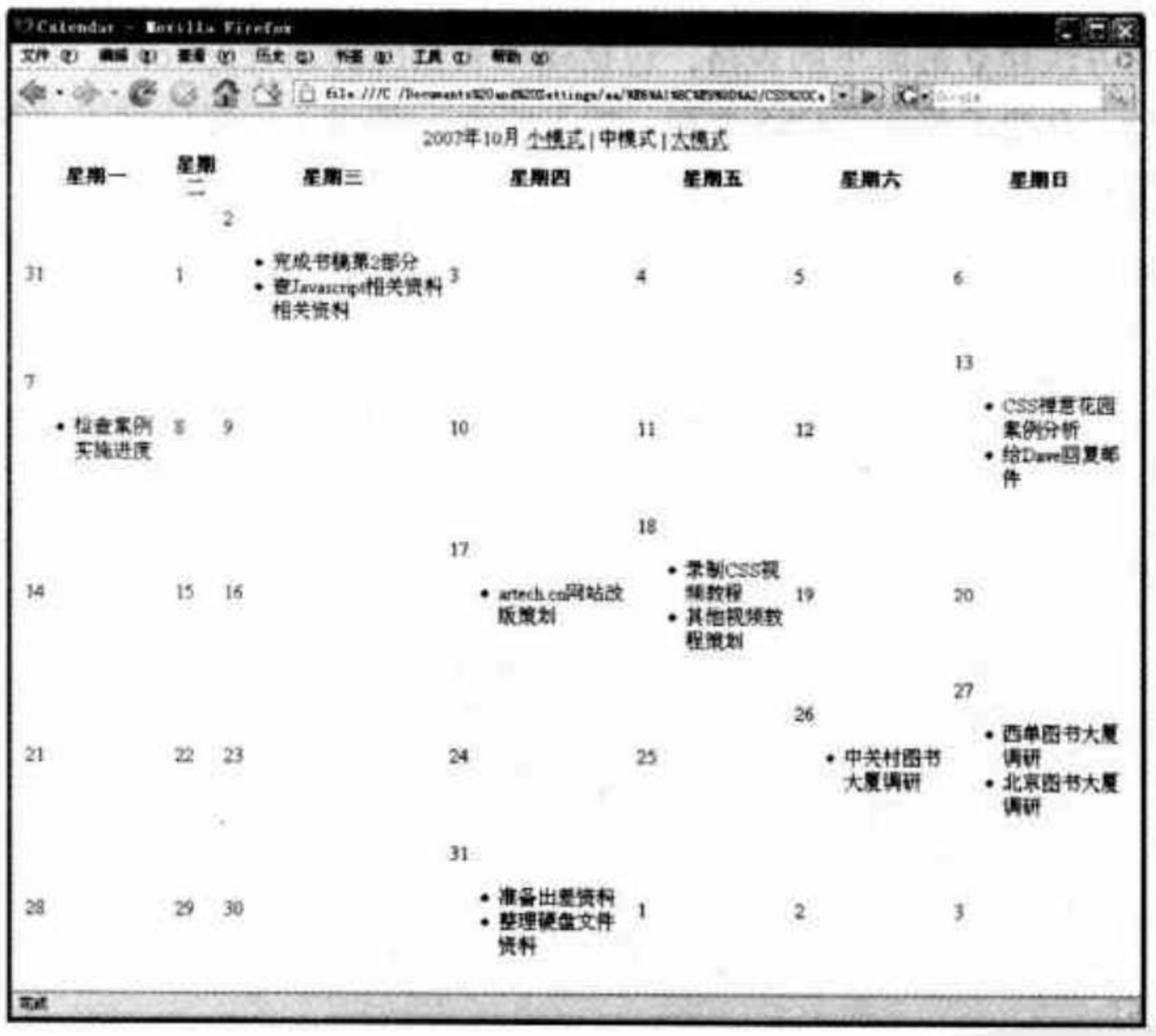


图10.29 未添加CSS的表格

10.6.2 设置整体样式和表头样式

在建立好表格的框架结构后，开始编写CSS样式。

① 首先添加对整个表格的控制，如下所示。

```
.month {
  border-collapse: collapse;
  table-layout:fixed;
  width:780;
}
```

注意 需要特别注意上面的两条CSS样式。

● “border-collapse: collapse;”的作用是使边框使用重合模式，从最终的效果图中可以看到相邻单元格之间的边框是重合在一起的。

● “table-layout:fixed;”的作用是用固定宽度的布局方式，使每一列的宽度都相等。从图中可以看到，由于星期二这一列中没有任何日程安排，因此被挤得很窄。如果希望各列都一样宽，就需要使用固定布局方式，严格按照width属性来确定各列的宽度。

② 设置<caption>和<th>的基本属性。

```
.month caption {
  text-align: left;
  font-family: normal 120% 宋体, arial;
  font-size:12px;
  font-weight:normal;
  padding-bottom: 6px;
}

.month caption .date{
    font-size:150%;
    font-weight:bold;
}

.month th {
  border: 1px solid #999;
  border-bottom: none;
  padding: 3px 2px 2px;
  margin:0;
  background-color: #ADD;
  color: #333;
  font: 80% 宋体;
}
```

此时的表头部分已经初见效果，如图10.30所示，列名称中各个星期的样式都不再显得那么单调了。

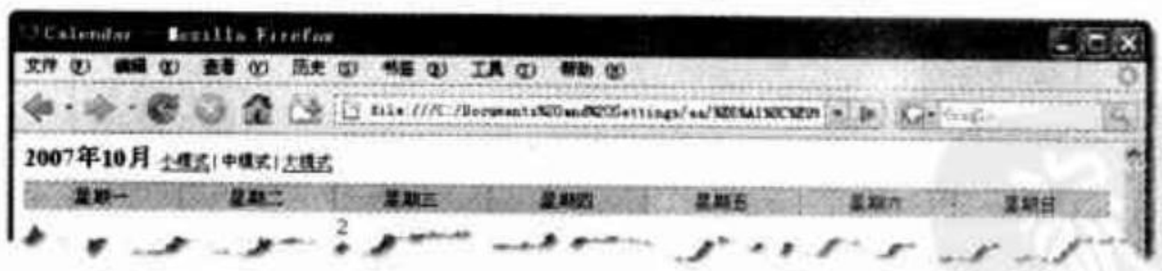


图10.30 控制标题、单元格等

10.6.3 设置日历单元格样式

现在来设置各个单元格的样式。整个表格中的单元格一共分为4种，即“普通的”、“有日程安排的”、“上个月的”和“下个月的”。后三者分别设置了active、previous和next类别，因此先对普通单元格进行设置，它也是后三者的所具有的“共性”样式基础。具体的步骤如下。

① 对普通单元格进行设置，代码如下。

```
.month td {
 border: 1px solid #AAA;
 font: 12px 宋体;
  line-height:16px;
 padding: 2px 2px;
 margin:0;
 vertical-align: top;
 }
```

② 设置previous和next类别的“个性”属性。没有提到的属性都与前面的“共性”设置一致。这里仅将背景色设置为灰色，文字也设置为灰色，因为这几个单元格不是当月的内容，所以希望使它不容易引起访问者注意。

```
.month td.previous, .month td.next {
 background-color: #eee;
 color: #A6A6A6;
}
```

③ 设置有日程安排的单元格。这个设置的目的是使它比较醒目，因此设置了深色的边框，并设为2像素的粗边框。

```
.month td.active {
 background-color: #B1CBE1;
 border: 2px solid #4682B4;
}
```

此时的表格已经初见效果，如图10.31所示，表格和单元格的边框、上个月和下个月的日期单元格有灰色背景，当月单元格为白色。

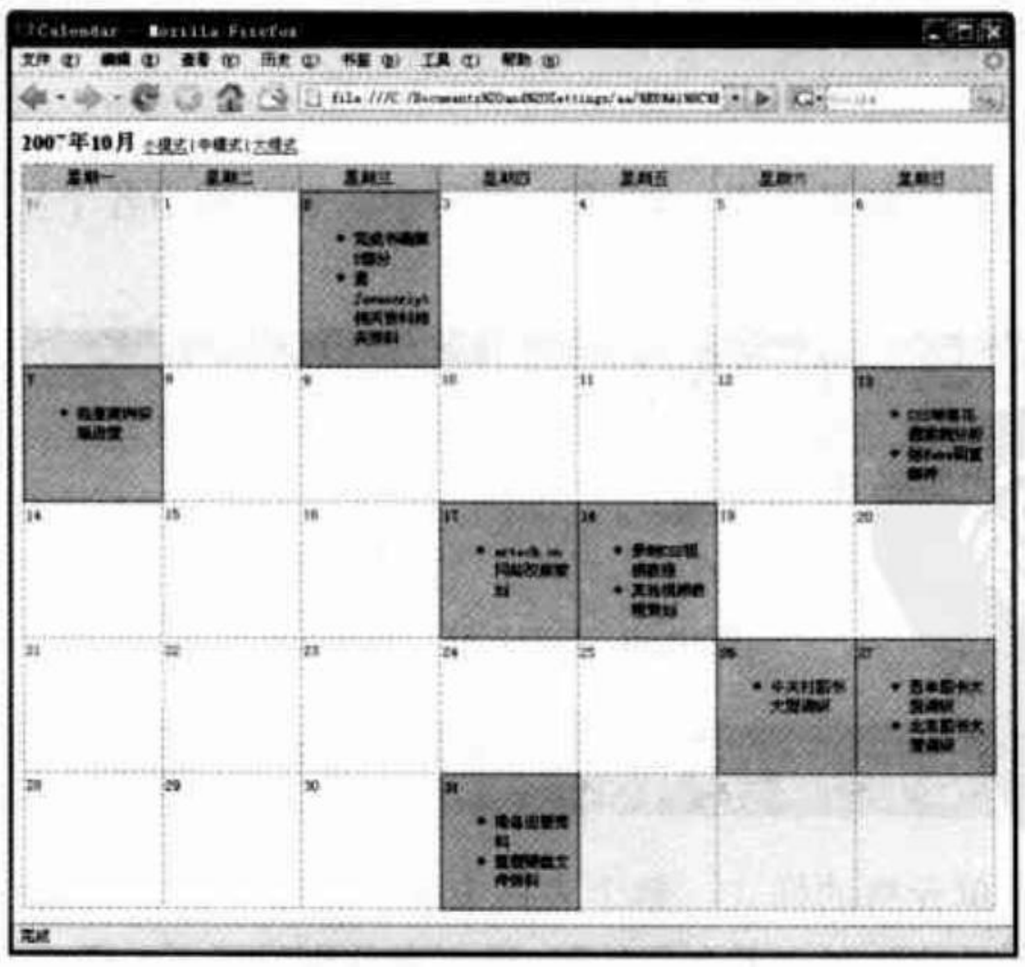


图10.31 对单元格进行设置后的效果

注意　上面为td中的文本设置了line-height，即行高。这里设置的行高会对左上角的日期数字有效，也会对里面的li中的文本有效。之所以设置明确的行高，是因为后面的步骤中需要设置背景图像，而不同浏览器默认的行高有所不同，为了在不同浏览器中都能将两行文字正好放到背景图像中，就必须给出明确的行高，代替各浏览器的默认值。

④ 对日程安排中的事情列表进行CSS控制，清除每个事件前面的小圆点，事件与事件之间添加一定的空隙，并设置背景图像，如下所示。

```
.month ul {
  list-style-type: none;
  margin: 3px;
  padding:0;
}

.month li {
  color:#fff;
    background:transparent url(level-2.gif) no-repeat;
    padding:2px;
  margin-bottom: 6px;
  height:34px;
  overflow:hidden;
  width:100px;
}
```

此时表格的样式结构已经基本定型，如图10.32所示。

注意　在li中的两处设置。

- 高度（height）设置为34像素，这正好是文字行高的两倍，文字行高在td中设置。
- 把溢出（overflow）设置为隐藏，以使显示不下的文字都隐藏起来。

图10.32　设定事件列表

⑤ 把重要日程安排的背景设置为另外一个图片，代码如下。

```
.month td li.important{
    background:transparent url(level-1.gif) no-repeat;
}
```

这样，中模式的页面就制作完成了，效果如图10.33所示。

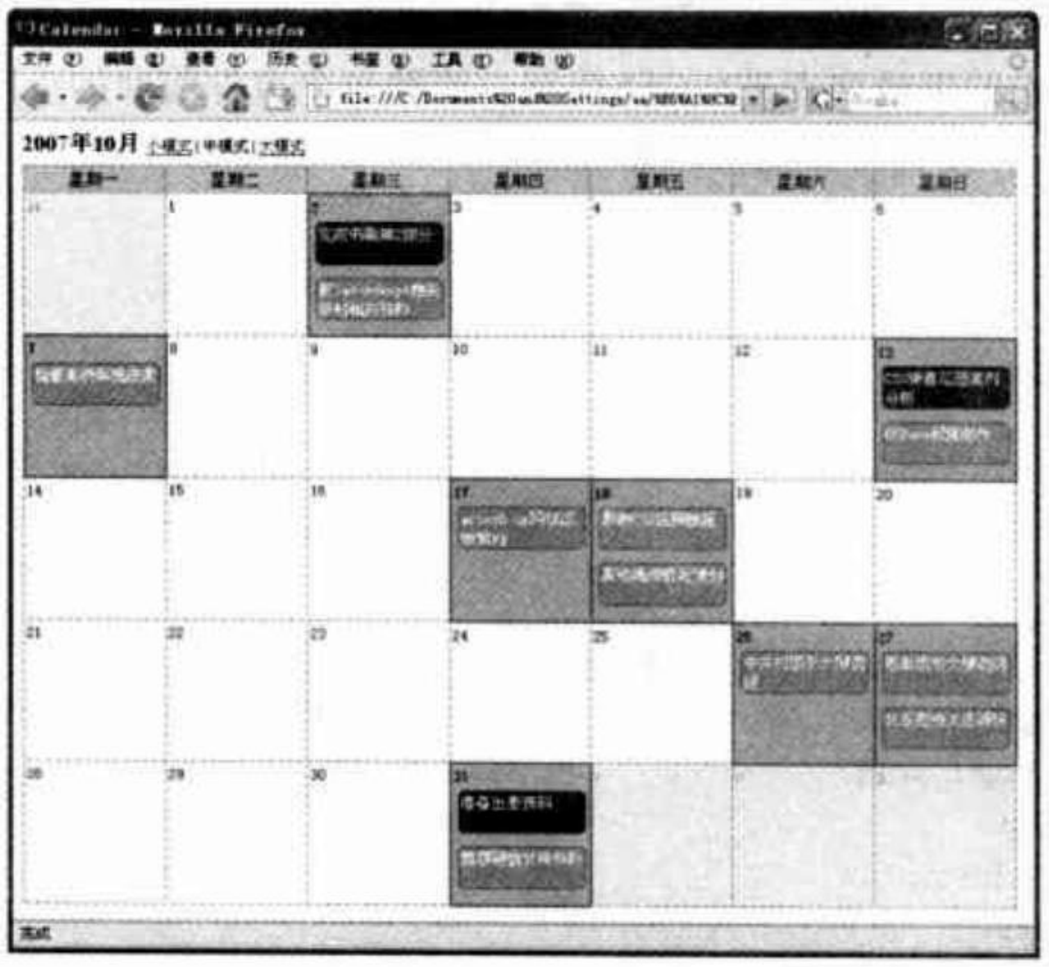


图10.33 对重要活动进行特殊设置

10.6.4 总结经验

对于这样的页面，文字、图像和边框等即使差1个像素，也会影响最终的效果，因此需要精细的调整，一点一点尝试，而且要在不同的浏览器上进行测试。这里有两点经验，供读者参考。

（1）当一个效果在Firefox和IE 6中显示得不一样时，一般来说Firefox显示的是“正确”的效果。这里说的正确，并不是指主观希望的效果达到了就是正确效果，而是说按照真正的CSS规则，应该显示的效果才叫作正确的效果。也就是说，如果Firefox中显示的效果和希望的效果吻合，而与IE显示不吻合，那么是IE有错误的可能性更大；而反之则说明很可能是为了迁就IE 6的错误，而写了错误的代码。

那么为什么不能以IE为标准，把IE的效果当作是正确的呢？这是因为，CSS的规则本身是严格符合逻辑的，是可以计算和预测的，而IE中的很多错误是没有道理的、无法预测的，只是用一个小的错误修正了另一个小的错误，在局部看起来可能效果是正确的，但是很可能在其他地方，或者更大范围内带来不可预知的麻烦，从而严重影响效率。因此，比较好的做法是以Firefox作为正确的效果，想办法让IE来适应它。

（2）测试的时候，不要在一个浏览器中完全做好后，再用另一个浏览器测试。对于一个很复杂的页面，如果先在Firefox中制作并完全测试好了，然后用IE查看时，可能很多地方都是混乱的，此时再针对IE进行一系列调整，等调整好了，回到Firefox查看时，肯定又乱了。如此往复，

结果可想而知。因此，应从空白页面开始，每做一小步，就同时在各种浏览器中查看，一旦发现显示效果不同，就立即查找原因，寻找解决办法。这样做到最后就可以同时满足各种浏览器了。

已完成的这个中视图模式页面中，每一列的宽是固定的，每个单元的高度是根据内容自动伸展的。例如在某一天中增加多个活动安排，该单元格就会变高，并且它所在的一整行都会一起变高，这是表格的本身性质决定的。

例如，要在27日增加一项活动，只需要增加一行代码就可以实现。

```
<td class="active">27
<ul>
    <li>西单图书大厦调研</li>
    <li>北京图书大厦调研</li>
    <li>完成CSS选题调研报告初稿</li>
</ul>
</td>
```

效果如图10.34所示。

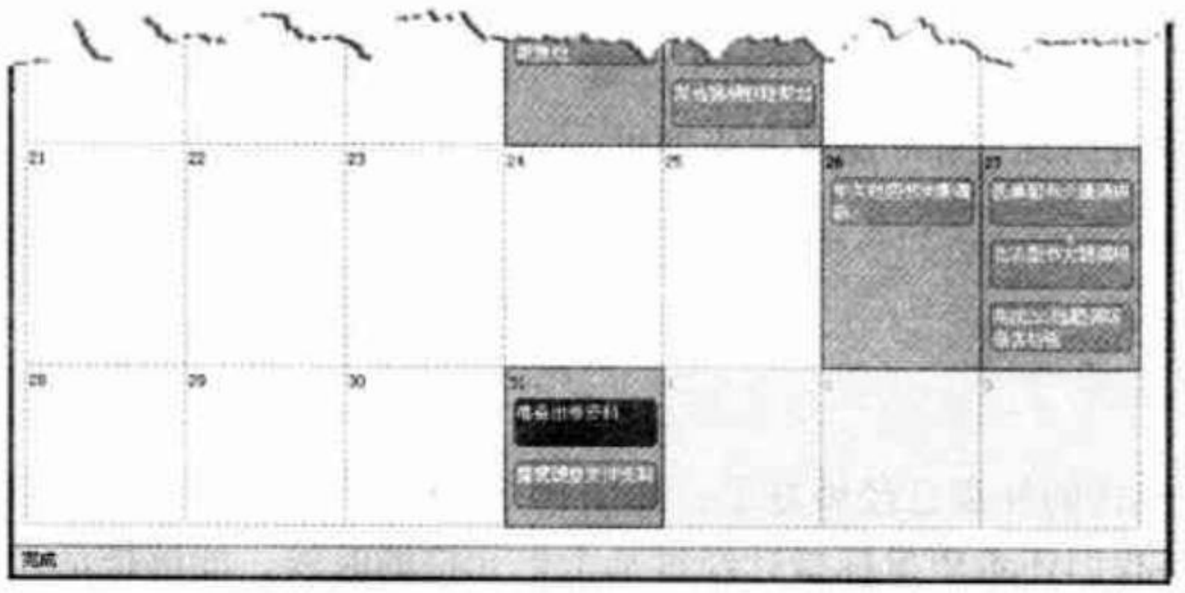


图10.34　在某一天中增加活动安排

10.7 制作小视图模式

下面在中模式的基础上，制作小视图模式的效果。小视图模式的日历实例文件位于本书光盘的“第10章\03\calendar-small.htm”。

为了便于读者理解在中模式的基础上修了哪些CSS设置，这里将完全保持现有的CSS和HTML不变，然后增加若干条CSS样式，实现小模式的效果。HTML部分仅在caption部分的3种模式切换的链接做了修改，与日历本身的所有代码不做任何修改。

10.7.1 整体设置

① 首先将整个表格的宽度设置为245像素，将宽度变窄，代码如下。这时的效果如图10.35所示。

```
.month{
  width:245px;
}
```

图10.35 将单元格宽度变窄

② 把需要隐藏的内容都隐藏起来。代码如下，即把每列的列头单元格中的“星期”两字隐藏，以及把每天的活动安排信息列表隐藏。“星期”两个字在一开始的时候，都已经放置在span中了，现在将这些span隐藏起来即可。

```
.month th span ,
.month ul {
  display:none;
}
```

至此小视图模式的外观已经做好了，如图10.36所示。但是仅做成这样还体现不出足够的“技术含量”，我们还希望鼠标指针经过某个单元格的时候，能够把活动安排的信息内容以弹出框的形式显示出来。

图10.36 将部分内容隐藏起来

10.7.2 为IE 7和Firefox制作鼠标指针经过时弹出的信息框

Firefox和IE 7都支持td的“:hover”伪类，也就是可以设置“td: hover”的样式，那么显示弹出信息框这个任务就可以完全依靠CSS来完成了。

注意　如果要使IE 7能够支持“td: hover”，必须正确设置DOCTYPE，设置成xhtml 1.0 transitional即可。如果没有设置DOCTYPE，在IE 7中的表现会和IE 6中一样。

① 首先实现鼠标指针经过有活动安排的单元格时，使单元格变色。这个要求比较容易实现，具体代码如下。

```
.month td.active:hover {
  background:aqua;
}
```

② 设置显示活动安排。基本思路是把隐藏的ul再显示出来，但是需要对样式做一些调整。假设先仅仅显示该列表，而不做任何其他调整，代码为：

```
.month td.active:hover ul {
  display:block;
}
```

这时效果如图10.37所示。由于显示了活动信息所在的ul，因此导致表格单元格的高度发生了变化，这不是希望的效果。

图10.37　不正确的显示效果

③ 这提示我们，为了显示活动安排信息后，不影响日历表格本身的单元格，必须使ul脱离标准流，因此必须使用绝对定位。这里将设置改为：

```
.month td.active:hover ul{
  display:block;
  position:absolute;
}
```

这时的效果如图10.38所示，可以看到已经相当接近目标了。

图10.38　正确显示了活动列表

④ 设置背景色和边框。背景色使用和日历上的active单元格相同的背景色即可，边框用什么颜色好呢？由于这个边框会和底下的表格重叠，因此用任何颜色都有可能显得比较杂乱，这里用白色是一个比较好的选择，它可以使弹出的信息和背景上的表格有一定的隔离效果。代码如下：

```
.month td.active:hover ul{
  display:block;
  position:absolute;
  border:3px #FFF  solid;
  background:#bde;
}
```

这时效果如图10.39所示。

图10.39 设置活动列表的样式

⑤ 设置margin和padding，使这个矩形内的排版更美观一些。和前面相似，增加下面两行设置。

```
margin:10px 0 0;
padding:7px 2px 0 5px;
```

效果如图10.40所示。这两行代码的设置含义是，通过padding使信息文字摆放到中间位置，通过margin把这个矩形框整体向下移动一些，否则距离鼠标指针距离位置有些太近了。

图10.40 向下移动一定距离

注意 为了把这个矩形框整体向下移动一定距离，使用的是margin属性，而不是通常对绝对定位元素使用的top属性。这是因为，如果使用top属性，就必须使鼠标指针经过的单元格成为这个矩形框（本质是一个ul列表）的定位基准，而td的display属性默认值是一种特殊的类型，不是普通的block，一旦将td的display转换成block，整个表格的

结构就会被破坏，因此这不是可行的方法。如果不使用top属性，绝对定位的元素仍然在它原来的位置上，这时通过设定margin就可以方便地移动它。这也是一个很有用的技巧，在本书第4章，专题讲解定位属性的时候，提到过这个性质。

至此，在Firefox和IE 7中已经完全实现了小视图模式的目标。

10.7.3 为IE 6制作鼠标指针经过时弹出的信息框

上面一小节介绍的方法对于IE和Firefox是有效的，而对于超过80%的访问者使用的IE 6该怎么办呢？答案只能有一个，那就是JavaScript。

首先用类别来代替伪类，将上面两段代码分别增加一个选择器，如下所示。

```
.month td.active:hover,
.month td.hover {
 background:aqua;
}
.month td.active:hover ul ,
.month td.hover ul{
  margin:10px 0 0;
  border:3px #FFF  solid;
  padding:7px 2px 0 5px;
  background:#bde;
  display:block;
  position:absolute;
}
```

为了先检验一下这两个类别是否起作用，可以在HTML中任意找一个有活动安排的单元格，例如7日这天的单元格，代码如下。

```
<td class="active">
        7
        <ul>
            <li>检查案例实施进度</li>
        </ul>
  </td>
```

将它修改为：

```
<td class="active"
        onmouseover="className='active hover';"
        onmouseout="className='active';">
        7
        <ul>
            <li>检查案例实施进度</li>
        </ul>
    </td>
```

这时这个单元格在鼠标指针经过时，动态地把类别由class="active"改为class="active hover"。同理，当鼠标指针离开的时候，再把类别动态地改回class="active"。当鼠标指针经

过“7日”这个单元格时，就出现了弹出的矩形框，而在其他单元格上都没有，这就证明了上面的方法是可行的。

如果读者不会使用JavaScript，就可以为每一个有活动安排的单元格，逐一添加这两句相同的代码。如果读者有一定的JavaScript和DOM编程的基础，则可以写一小段JavaScript代码来完成手工的工作。方法是在整个表格的HTML结束以后，即在</table>之后，添加如下一段代码。

```
……省略表格的HTML代码……
</table>
<script language="javascript">
var days = document.getElementsByTagName('td');
for (var i=0;i<days.length;i++)
  if(days[i].className == 'active')
  {days[i].onmouseover = function(){                    //鼠标在行上面的时候
        this.className = 'active hover';
  }
  days[i].onmouseout = function(){                      //鼠标离开时
        this.className = 'active';
  }
}
</script>
</body>
</html>
```

下面简单讲解一下这段代码的原理。首先通过“document.getElementsByTagName('td');”取得整个文档的DOM树中的所有td节点，并以数组的形式存储在“days”变量中，然后通过一个循环语句，依次检查每一个td节点。如果某个节点的类别被设置为active，就为该节点的onmouseover事件句柄设置一个函数。在函数中把该函数的类别增加一个hover类别，然后把该节点的onmouseout事件句柄设置为恢复类别的函数。

这段代码将在表格装载完成以后执行一次，就相当于手工为每个active单元格添加代码。这样，在IE 6中，也可以实现鼠标指针经过时弹出提示信息的功能了，效果如图10.41所示。

图10.41 在IE 6中实现的效果

可以看到，矩形框的绝对定位在IE中和在Firefox中还是有细微区别的。如果要求严格相同的话，可以用前面介绍过的条件注释的功能，针对IE稍微调整一下margin和padding的值，就可以取得完全相同的效果了。

至此，已经实现了中视图模式和小视图模式，并且都保证了可以在IE 6、IE 7和Firefox浏览器中正确显示。

10.8 制作大视图模式

下面就来制作大视图模式，效果如图10.42所示。大视图模式的日历实例文件位于本书光盘的“第10章\03\calendar-large.htm”。

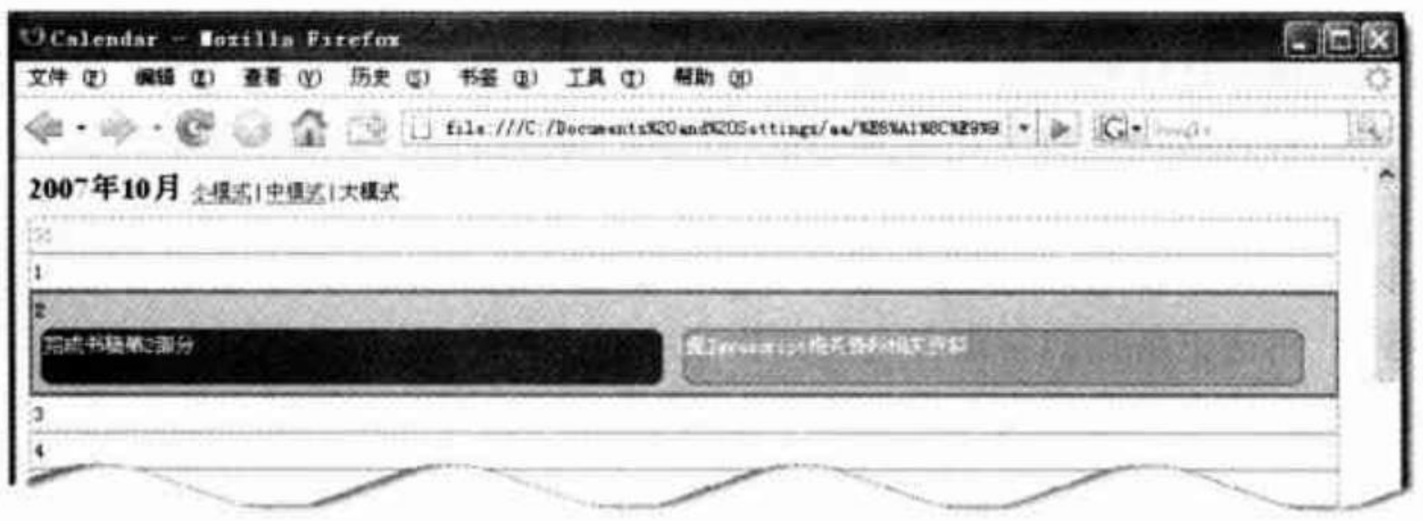


图10.42　大视图模式下显示日历

需要说明的是，本案例要求不修改HTML结构。在此前提下，大视图模式无法在IE浏览器中实现，除非使用JavaScript动态修改DOM结构，而这和修改HTML结构就没有本质区别了，因此实现大视图模式页面仅作为帮助读者理解CSS的核心思想的一个案例。

10.8.1 通过display属性改变盒子的类型

仍然使用原有的HTML结构。要实现这个大模式的日历，最困难的一点是，在原本的HTML中，表格是一个二维结构，而在大模式中，希望它成为一维的线性结构，所有单元格从上向下依次排列。

从某种角度来说，这个要求已经改变了HTML本身的语义。因此，这实际上也是一个不经常遇到的要求，但是从CSS规范的角度来说，是可以做到的。

在讲解盒子模型的第4章最后一节中，增加讲到过，任何元素都有display属性，它经常用的值是none、inline和block。将一个元素的display属性设置为none，它就被隐藏起来了。inline则是行内元素，例如a标记等。而将一个原本是行内元素，例如a标记，设置为block，就成为一个块级元素了。

实际上，display属性的值还远不只这些。在使用具有代码提示功能的软件（如Expression Web）时，输入冒号以后，就会出现一个下拉列表，提示用户某个属性可能的属性值有哪些，display的候选属性就有很多，如图10.43所示。

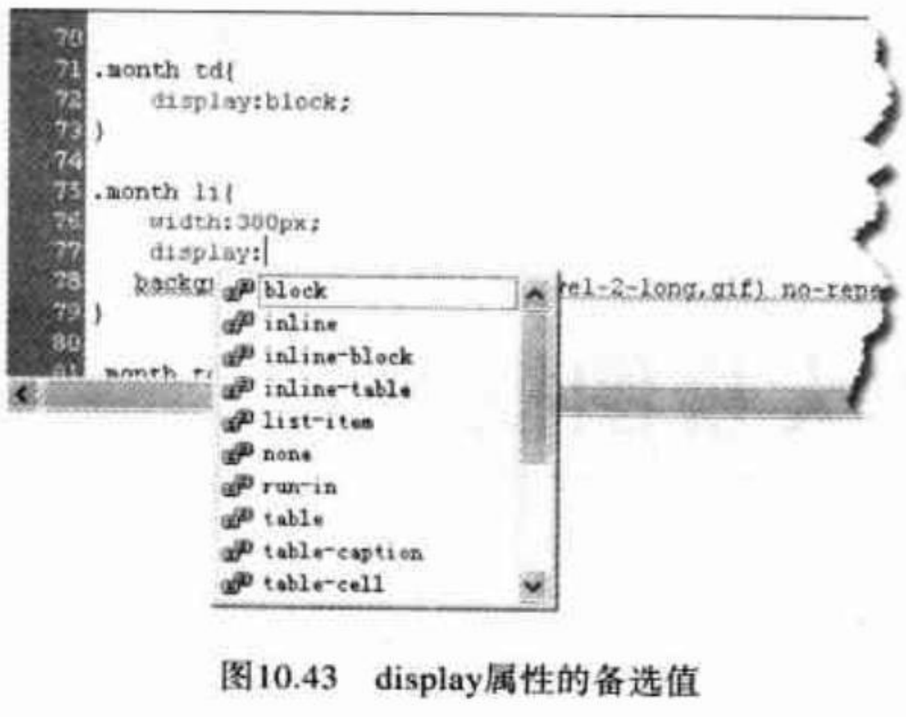


图10.43 display属性的备选值

每一种值都代表着一定的排版行为，例如inline-block就表示该元素占据一定的矩形空间，但是又像行内元素一样排列，等等。具体含义可以查看CSS的规范，里面有详细的定义。

在表格中，各种表格元素（比如单元格、行等）都有自己的display属性值，它们的表现也各不相同。如果强制转换某个元素的display属性，就会随之改变它的表现行为。例如，把td的display属性转换为block，它就会像一个在标准流中的block一样，宽度默认会自动伸展，依次竖直排列，而这正是我们希望在大视图模式中显示的效果。

因此我们在中模式的基础上，在CSS部分增加如下代码：

```
.month td{
  display:block;
}
.month th{
  display:none;
}
```

这样两个设置，首先把单元格的display属性转换为block，然后把星期一到星期日的表头行隐藏起来，因为它已经没有意义了。

可以看到效果如图10.44所示，所有的单元格已经被成功地“拉直”成为一竖排了。

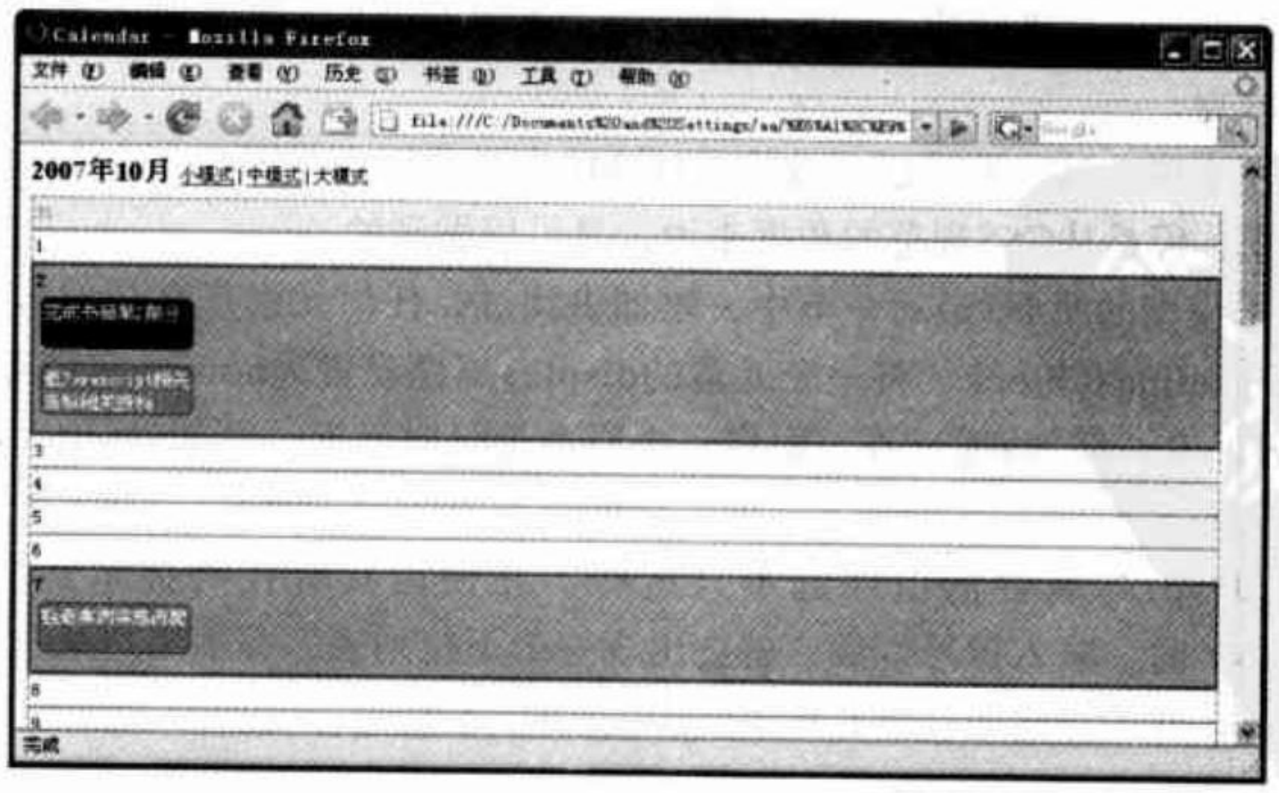


图10.44 将表格“拉直”

注意　这个效果在IE 6和IE 7中都是无效的。

10.8.2 设置日程安排项目

现在的缺陷是每个日程安排信息的位置还是太小了，而仅让各个活动信息的宽度适应整个表格的宽度，又显得长了一些，因此希望每行显示两个活动，如图10.45所示。

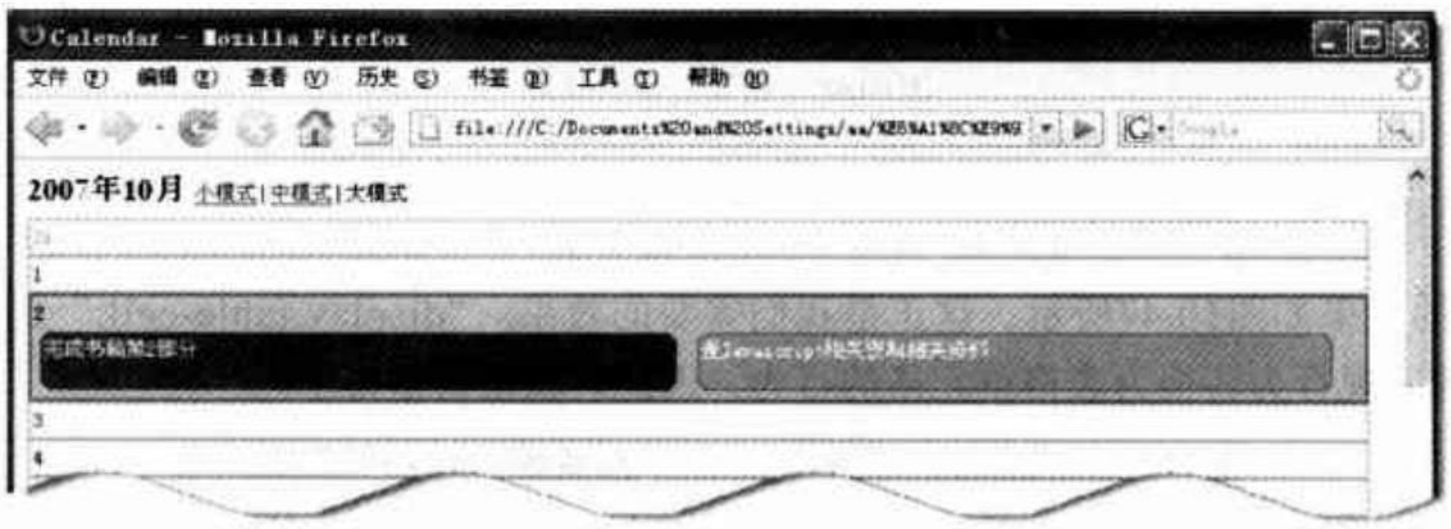


图10.45　希望实现的目标效果

首先，把每个活动安排信息的宽度变宽，并修改背景图像。当然可以使用前面的案例中学到的方法，通过“滑动门”技术以及背景图片的定位，用一个图片来实现需要，但是这不是本案例的关键问题，因此这里就为大模式准备两个相应的图片就可以了。

问题的关键是，如何让原来排在下面的li移动到右边。我们很自然会想到用浮动的方法，设置float属性，代码如下。

```
.month li{
 width:370px;
   background:transparent url(level-2-long.gif) no-repeat;
   float:left;
}
.month td li.important{
   background:transparent url(level-1-long.gif) no-repeat;
}.month li{
```

这样设置之后，原来的竖排一定可以变成横排。如果还是竖排，就是因为li的宽度设置得太宽了，以至于浮动方式下，一行中放不下两个活动。

请读者在这里停止阅读，先不要着急看结果，思考一下，用这种方法得到的并排效果存在什么问题？

这个问题的答案实际上在本书最开始讲解盒子模型的时候已经提到过，当一个盒子内部只有浮动对象的时候，这个盒子本身的高度为0，也就是说，此时活动安排信息的盒子已经脱离标准流了，效果如图10.46所示。

也就是td不会向下伸展，这显然不是希望的效果。要怎么解决这个问题呢？答案还是使用diplay属性，去掉float属性，然后增加一条代码：

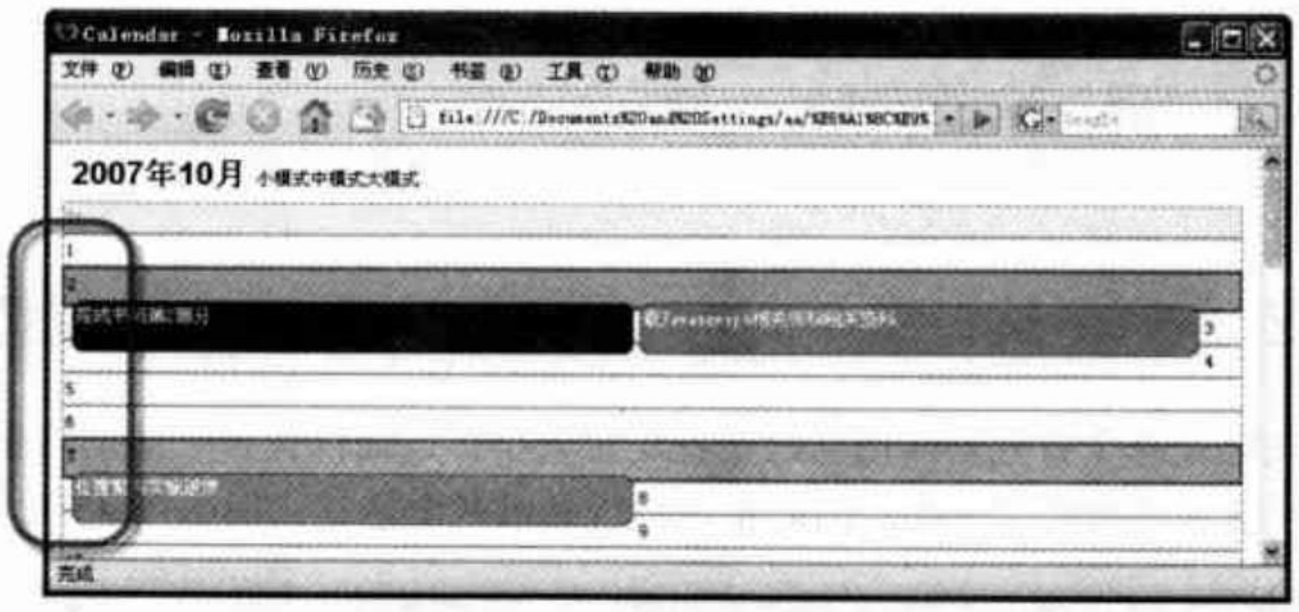


图10.46 错误的显示效果

```
display:table-cell;
```

现在的效果如图10.47所示，这正是我们需要的效果。“display:table-cell;”的作用就是把li元素的表现行为转换为表格单元格的行为，因此它就依次横排了。

图10.47 正确的显示效果

这个方法也不是完美的，例如，如果再增加一个安排，这时的效果将变成如图10.48所示的效果。

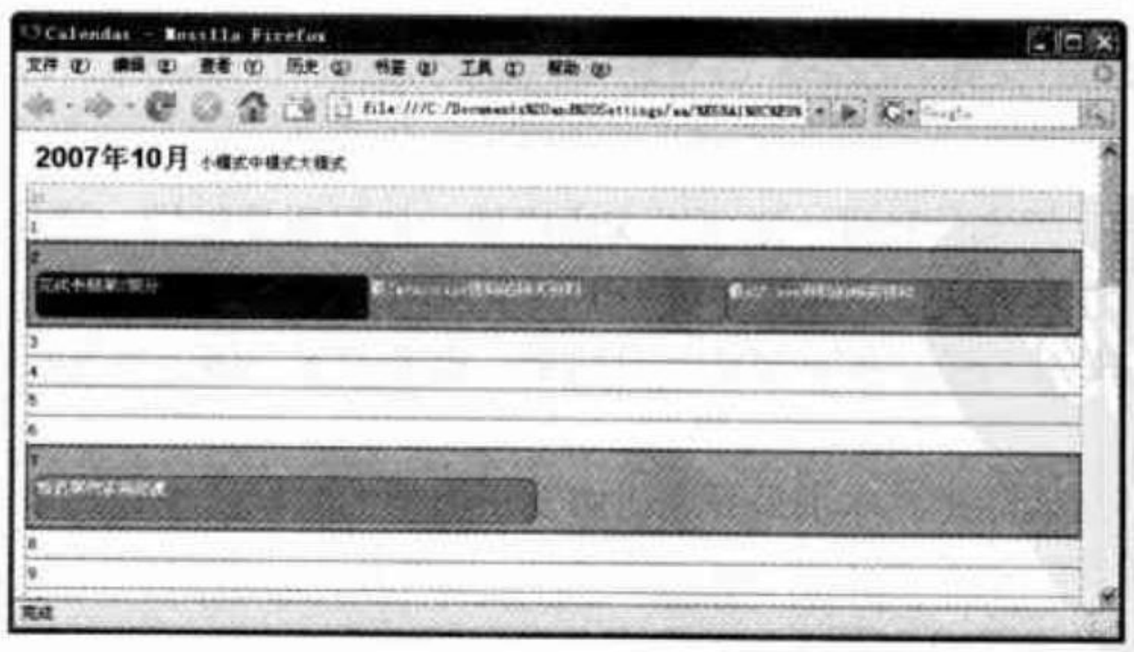


图10.48 增加列表项后，效果产生纰漏

3个项目会共同分享整个宽度，因此在完全不改变HTML结构的情况下，要得到非常完美的效果是比较困难的。或者说，在编写HTML的时候，就要充分考虑到在不同模式下的控

制问题，就好像为了在小视图模式能够隐藏“星期”两字，事先就在HTML中设置了span钩子一样。

通过这个例子，最重要的一点是，希望读者可以更深刻地理解display属性的重要作用。

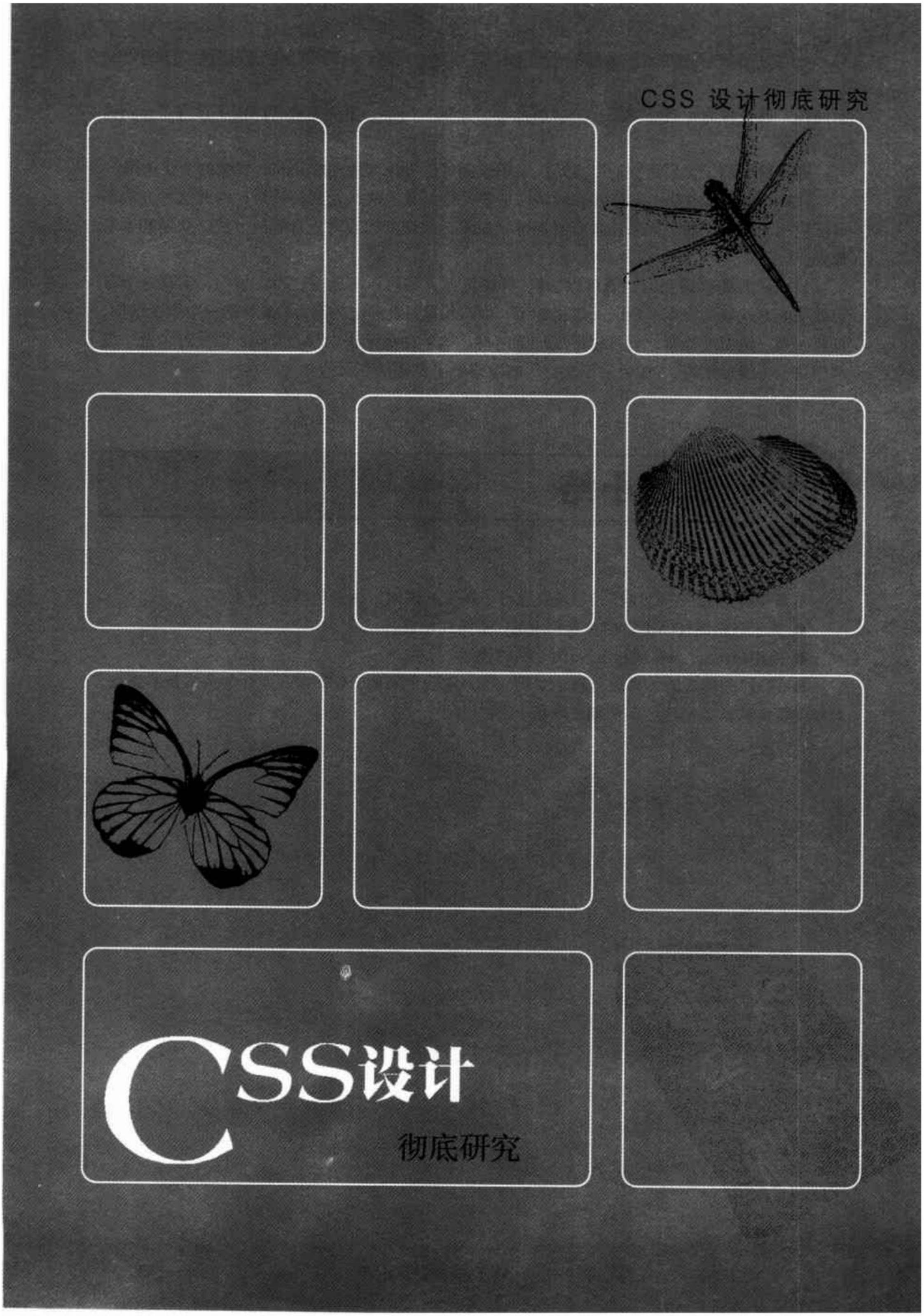
在IE 6和IE 7中都不支持display属性从单元格到普通块级元素的转换，因此这种方法不可能应用于IE浏览器。如果读者有兴趣继续探索，找到变通或者更有趣的方法，欢迎和本书作者探讨。

目前作者能够想到的，既可以适应3种浏览器，又可以不改变HTML结构，实现这个日历效果的方式就是根本不用table表格标记，而改用ul列表标记。因为ul本身是一个线性结构，只要按照一定宽度换行，就可以形成二维结构，这不存在无法克服的障碍；而反过来把二维结构显示为线形结构，“IE目前不支持”的障碍似乎是无法克服的。

10.9 本章小结

本章中针对表格的CSS样式的设置进行了深入的探讨。主要包括3个方面。

- 关于表格的HTML结构及其相应的CSS属性设置。
- 使用JavaScript实现对:hove伪类别的模拟。
- 通过3种模式的“日历”这个案例，深入综合地实践了CSS样式设计的多个技术要点，希望读者能够认真把这个案例彻底吃透。



第11章 圆角设计

在网页设计中，通常需要把网页分为若干个部分，这正是CSS的强项。使用CSS可以方便地使用各种手段把页面灵活地分为多个部分。但是简单分割出来的都是矩形方框，设置背景颜色和边框的颜色，产生的都只能是矩形的方框。而在网页中，经常会需要用到圆角的设计。因此，为了使页面中可以灵活地使用圆角框，有很多专家都研究出了非常好的方法，本章就来专门对圆角的设计进行介绍。

11.1 圆角框的作用

图11.1所示的是一个非常常见的布局形式，上面是页面的头部（header），下面是页脚（footer），页面分为两列。这种部局非常适合内容不是很多的网站。

图11.1 前沿视频教室网页中的圆角效果

再如图11.2所示，这是著名的Adobe公司的首页，这样内容非常多的网站，其结构也会相应复杂。

图11.2 Adobe网页中的圆角效果

从上面的例子中，都可以看到圆角框发挥了很重要的修饰和分隔作用，对引导访问者的阅读起到了非常好的效果。

圆角框的具体分类也很多，制作方法也很多。例如，可以根据对宽度的适应性分为固定的、流动的和弹性的圆角框；根据使用的背景图片个数，又可以分为很多类型；根据使用的颜色情况，可以分为单背景色的还是带有边框的。

下面我们就从最简单的方法开始讲起。

11.2 固定宽度圆角框

因为不必考虑宽度变化带来的麻烦，所以制作宽度固定的圆角框比制作适应宽度的圆角框要简单很多。关键在于如何合理地使用背景图像。

11.2.1 两背景图像法

本例制作如图11.3所示的圆角框，使用两个背景图像文件，宽度固定。这种方法只适用于制作单色圆角框。本实例文件位于本书光盘“第11章\01\basic.htm”。

图11.3　圆角框案例效果

① 使用这种方法首先要确定圆角框的颜色，以及圆角框后面的网页背景的颜色，然后根据这两种颜色制作两个图像文件，如图11.4所示。例如，这里的网页背景为白色，圆角框为咖啡色，因此这两个图像的圆角外的小三角区域为白色，主体颜色为咖啡色。

图11.4　案例中用到的两个背景图片

② 编写HTML的整体结构。实际上只需要一个div作为圆角框的容器，里面有一个h3标题，以及一个或多个p段落，代码如下。

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>固定宽度</title>
        <meta http-equiv="content-type" content="text/html;
```

```
charset=gb2312" />
</head>
<body>
  <div id="rounded">
  <h3>Fixed Rounded</h3>
  <p>这是一个固定宽度的圆角框，由于是固定的宽度，因此制作起来容易和简单很多。这个圆角框的上下随着内容增多可以自由伸展，圆角不会被破坏。</p>
  </div>
</body>
</html>
```

分析 接下来考虑如何实现这个圆角框。参考前面几章中制作导航菜单的经验，显然两个圆角图像文件要作为背景图像出现，它们一个在最上面，一个在最下面，中间的主体的颜色可以通过背景颜色来实现。

③ 哪个图像作为哪个HTML元素的背景图像都可以，只要计算正确就可以了。例如，这里把下面的圆角图像放到div的背景中，代码如下。

```
<style type="text/css">
#rounded{
  font: 12px/1.6 arial;
  background: #cba276 url(images/bottom.gif) no-repeat left bottom;
  width: 280px;
  padding: 0 0 18px;
  margin:0 auto;
}
</style>
```

分析 （1）“background”属性的设置。“#cba276”代表整体背景色为咖啡色，后面的“url(images/bottom.gif)”指明图像文件的地址，“no-repeat”表示不平铺，“left bottom”表示放置在左下角。

（2）padding和margin的设置。这里必须把margin设置为0，因为margin部分不会有背景。“padding:0 0 18px”中的第1个0表示上padding为0，第2个0表示左右padding为0，第3个18px表示下padding为18像素。效果如图11.5所示，可以看到文字和圆角框的最下端有一定空白距离，这就是下padding的18个像素。

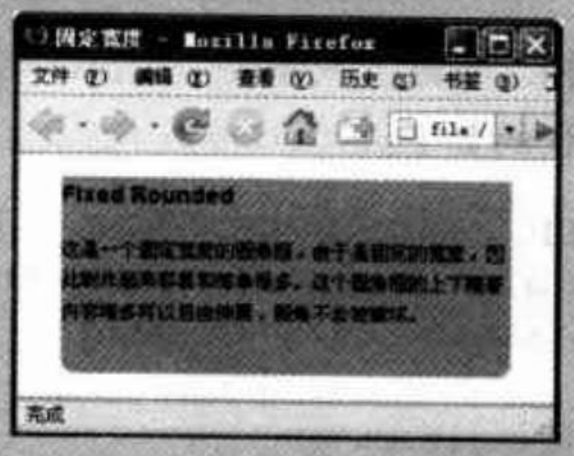


图11.5 设置好下端的圆角背景图像

④ 把上面的圆角图像放置到h3标题的背景中，代码如下。

```
#rounded h3 {
  background: url('images/top.gif') no-repeat;
  padding: 20px 20px 0;
  font-size: 170%;
  color: #FFF;
  line-height: 1;
  margin: 0;
}
```

同样需要理解backgroud、padding和margin的设置方法。这时的效果如图11.6所示。

图11.6 设置好上端的圆角背景图像

⑤ 设置段落的样式，使左右两端留出一些空白，代码如下。最终的效果如图11.7所示。

```
#rounded p {
  padding: 10px 20px 0;
  color: #1B220F;
  margin: 0;
  text-indent:2em;
}
```

图11.7 设置段落中的文本

如果要增加正文段落的个数，效果就如图11.8所示。

图11.8 增加一个文本段落

可以看到，这个圆角框可以自动向下伸展，在段落之间有一定的空白，这是因为设置了上paddding为10个像素。如果希望取消段落之间的空白，可以对第1个p增加一个class，单独设置，保持它和标题之间的距离，其他的段落则取消这个padding。这种方法在前面导航菜单相关的章节已经多次使用，请参考前面的案例，就可以找到具体做法的讲解。

这种两背景图像的方法还有一种变体，可以使用一个背景图像来实现，本质和两图像法相同，请参考本书光盘中的文件“第11章\02\single-round.htm”，这里就不再详细介绍了。

11.2.2 使用透明背景图

这里需要进一步思考另一个问题。前面提到，在制作圆角图像的时候除了要确定圆角框本身的颜色，还要确定网页背景的颜色，使圆角外侧处的颜色与背景色融为一体。那么可不可以把圆角外侧的两个小三角形区域设置为透明色，这样不就可以显示出网页背景颜色了吗？如果这个改进能够实现的话，这个圆角框就可以适用于各种网页背景色了。然而答案是否定的，请读者先不要看答案，仔细思考一下原因。

原因在于，整个div已经设置了背景色，这种颜色就是圆角框的颜色，这时如果把圆角外侧的小三角区域设置为透明色，那么透出来的颜色不是网页的背景色，而是div的背景色，从而无法产生圆角效果。因此，这种方式是不可行的。

虽然不能使圆角外侧的颜色呈透明色，但是可以把圆角图像的主体颜色变为透明色，而圆角外侧三角形仍然为网页背景颜色。这样做的好处是，这个圆角框的颜色可以通过div的背景色任意设置，而不需要在图像中设置了，大大增加了灵活性。例如，图11.9所示的效果就是用这种方法将上面的圆角框换为了蓝色。本实例文件位于本书光盘“第11章\01\transparent-image.htm”。

图11.9 使用透明背景图像

这样做有没有副作用呢？是有的，如图11.10所示，左侧为使用透明方案的效果，右侧为使用不透明方案的效果。

图11.10 两种方案的效果比较

请注意放大的圆圈中的效果，右侧的圆角由白色和咖啡色两种颜色在图像中构成，这样两种颜色在图像处理软件中经过“消锯齿”处理，圆角显得很光滑；而左侧的图像由于只有一种颜色，因此无法做“消锯齿”处理，从而使边界比较生硬。

任何事物都是有利就会有弊，我们要做的就是从全面的角度出发，选择综合最优的效果，因此需要读者在实践中根据实际情况来选择技术方案，因为技术永远是为设计思想服务的。

11.2.3 带边框的固定宽度圆角框

上一节中，制作的圆角框只能使用一种颜色，如果要实现如图11.11所示的效果，就无能为力了。本实例文件位于本书光盘“第11章\01\with-frame.htm”。

图11.11　带有边框的圆角框

因为中间部分的边框无法产生，只能为中间部分增加一个背景图像文件，所以共需要3个图像文件，如图11.12所示。

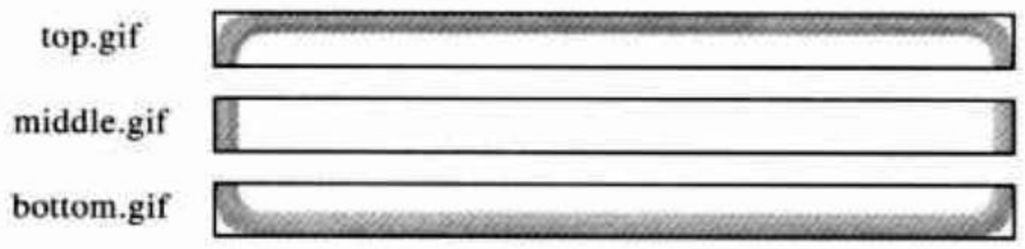


图11.12　案例中用到的背景图片

然后思考，既然增加了一个图像文件，就必须要把它挂到一个元素中。这段HTML中有3个标记，前面用了div和h3放置上下两个圆角图像，那么能否把剩下的p用来放置中间段的背景图像呢？显然不可以，因为标题部分的高度比上端图像要高，如果用p来放置中间段图像，就会产生一个裂缝。

因此必须调整一下，让div来平铺中间段背景图像，那么自然标题就用来放置上端图像，p来放置下端图像。下面仅列出CSS样式代码。

```
<style type="text/css">
#rounded{
  font: 12px/1.6 arial;
  background: url('images/middle-frame.gif') repeat-y;
  width: 280px;
  padding: 0;
  margin:0 auto;
  }
#rounded h3 {
  background: url('images/top-frame.gif') no-repeat;
  padding: 20px 20px 0;
```

```
    font-size: 170%;
    color: #cba276;
    margin: 0;
    }
  #rounded p {
    padding: 10px 20px 18px;
    color: #1B220F;
    margin: 0;
    text-indent:2em;
    background: url('images/bottom-frame.gif') no-repeat left bottom;
    }
  </style>
```

如果需要放置多个段落，则效果如图11.13所示，这是因为背景图像放置在了p标记中，这样每个段落都会有一个下端的背景图像。

图11.13 增加文本段落

解决方法是，改造HTML，为最后一个段落增加一个class，代码如下。

```
    <div id="rounded">
    <h3>Fixed Rounded</h3>
    <p>这是一个固定宽度的圆角框，由于是固定的宽度，因此制作起来容易和简单很多。这个圆角框的上下随着内容增多可以自由伸展，圆角不会被破坏。</p>
    <p  class="last">这是一个固定宽度的圆角框，由于是固定的宽度，因此制作起来容易和简单很多。这个圆角框的上下随着内容增多可以自由伸展，圆角不会被破坏。</p>
    </div>单图像带边框
```

然后将对p的样式设置分开为两段，代码如下，效果如图11.14所示。

```
  #rounded p {
    padding: 10px 20px 0;
    color: #1B220F;
    margin: 0;
    text-indent:2em;
  }
```

```
  #rounded p.last {
    padding: 0px 20px 18px;
    background: url('images/bottom-frame.gif') no-repeat left bottom;
  }
```

图11.14　增加文本段落并设置好样式

用这种方法就可以增加更多的段落了，只需要在最后一个段落设置“class="last"”即可。

11.2.4 单背景图像的带边框固定宽度圆角框

进一步思考，如果边框特殊到只有一个像素宽，或者边框虽然不是1像素宽，但它本身是一种单色，就可以使用CSS的边框属性来做中间段的边框，这样就不需要3个背景图像了。

例如，要制作如图11.15所示的圆角边框，它的左右边框只有1个像素宽，那么中间段的图像就不需要了。在这里演示一下，如何只使用一个背景图像实现这个圆角框。本实例文件位于本书光盘“第11章\02\single-round.htm”。

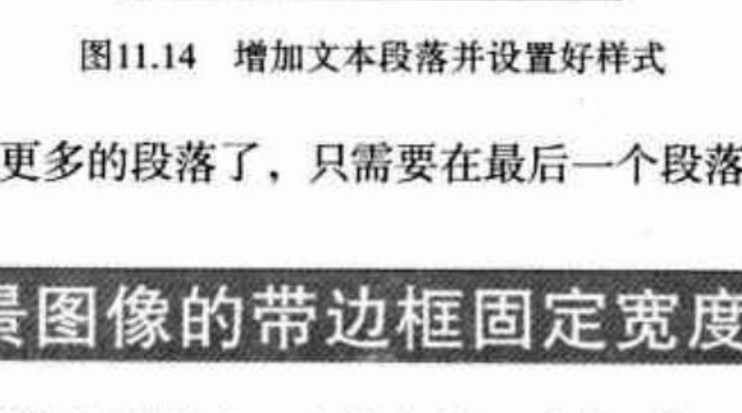


图11.15　单背景图像圆角框效果

使用的背景图像如图11.16所示。

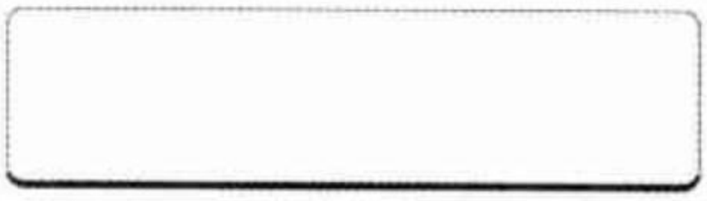


图11.16　案例中用到背景图像

HTML代码仍然使用前面的代码，不需要设置特殊的p段落。CSS的样式代码如下。

```
<style type="text/css">
.rounded {
        font: 12px/1.5 Arial;
        width:430px;
        padding:0 0 14px 0;
        margin:10px auto;
        background:url("bg.gif") bottom left no-repeat;
        }
```

```
.rounded h3 {
        margin:0;
        padding:5px 10px 7px 10px;
        background:url("bg.gif") top left no-repeat;
        color:#666;
        }
.rounded p {
        text-indent:2em;
        margin:0;
        padding:2px 10px 3px;
        border:1px solid #777;
        border-width:0 1px;
        background:#fff;
        }
</style>
```

可以看到，依然是对标题和div使用背景图像，并且使用同样的背景，只是一个显示上半部分，另一个显示下半部分。在对p的设置中，设置了左右border，均为1像素宽的实线，这样就产生了如图11.17所示的效果。

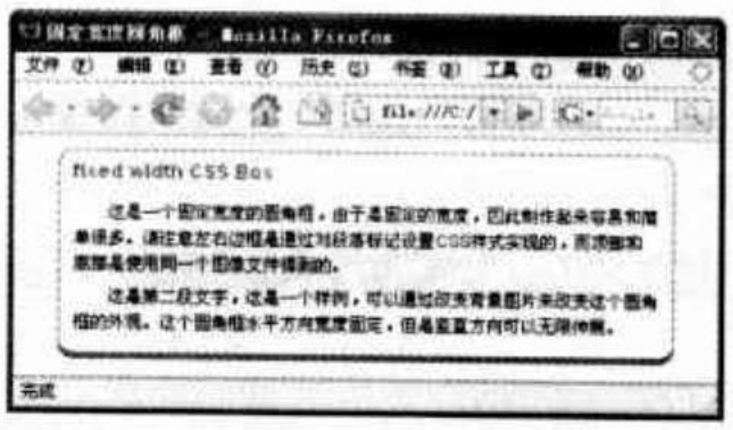


图11.17 完成后的效果

只需要更换背景图像，以及文字的颜色稍加变化，就可以产生很多丰富多彩的效果。例如，将背景图像换成如图11.18所示的样子，则圆角框的效果如图11.19所示。本实例文件位于本书光盘“第11章\02\single-round-2.htm”。

图11.18 更换另一个背景图像

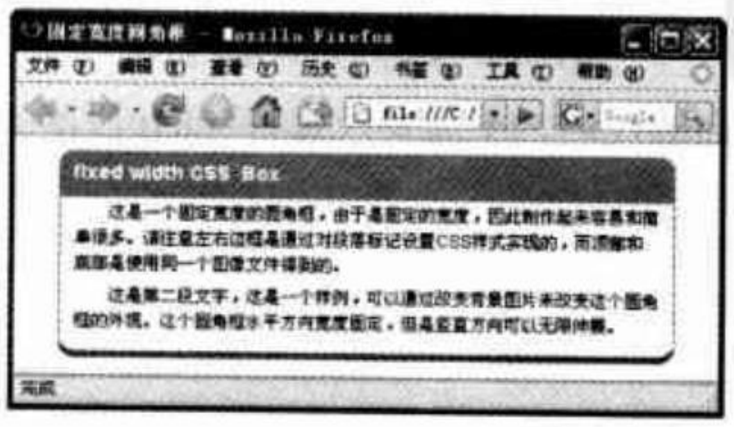


图11.19 第2个效果

使用上面这种方法可以做出很多种不同的边框效果，只需要修改相应的背景图像文件即

可实现不同的圆角框效果。因此，读者在实际工作的时候可以根据任务的实际情况，选择使用哪种方法。

11.3 不固定宽度的圆角框

上面介绍的方法都有一个很大的限制，就是必须固定宽度。如果一个网页如图11.20所示，它需要5种不同宽度的圆角框，就必须分别制作背景图像，这样不但麻烦，而且下载网页文件的字节数也会增多，严重影响显示速度。

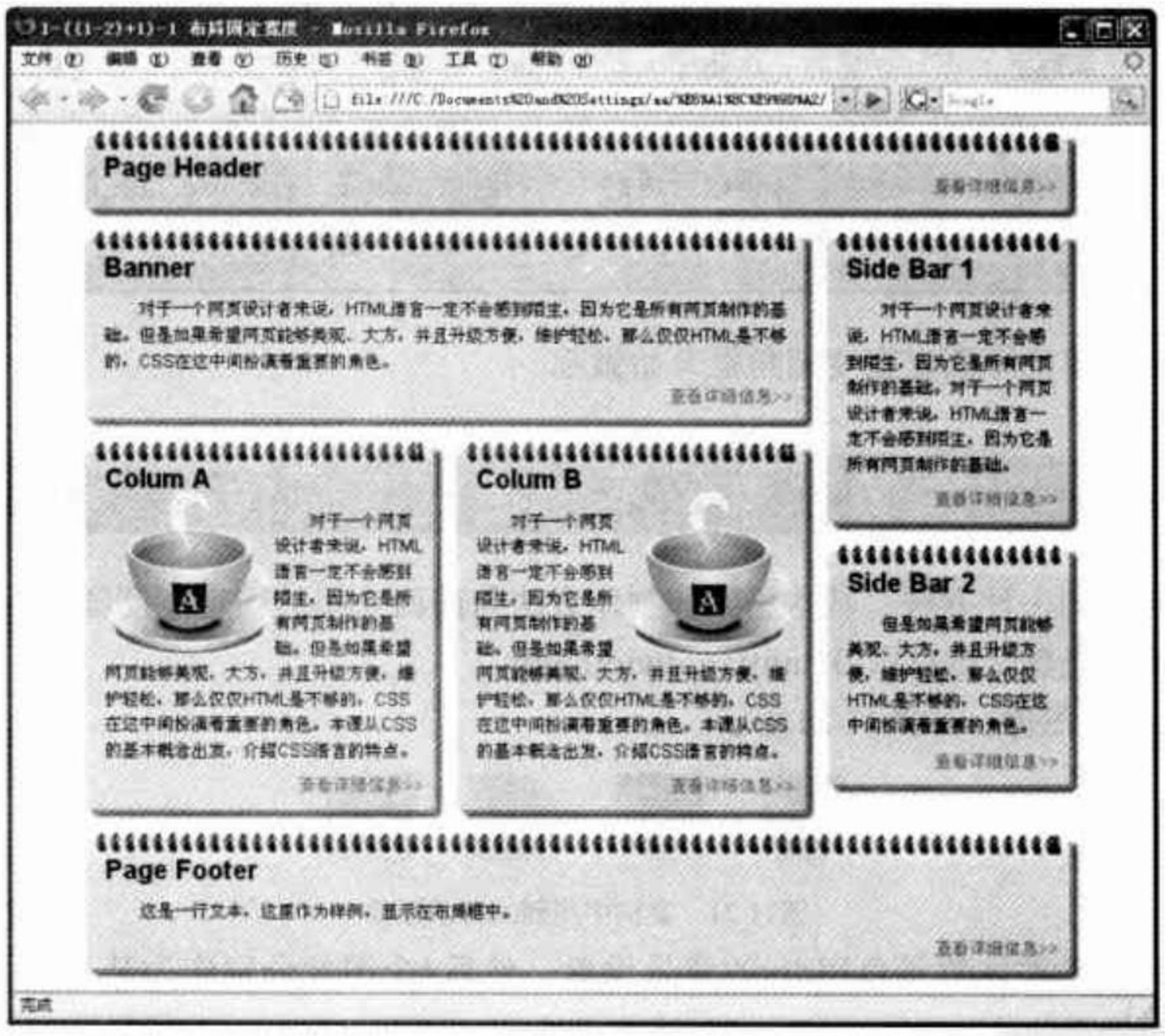


图11.20　复杂的页面

因此，必须找到更好的办法来解决这个问题。本节就来讨论制作可以自由变化宽度的圆角框。

11.3.1 不固定宽度圆角框的含义

对于宽度“不固定”这个说法，实际上又分为多种不同的具体方式。

（1）可变：含义是这个圆角框本身在网页上的宽度是固定的，但是在制作网页的时候，只要简单地设定该圆角框的宽度，就能产生正确的效果，而不必重新制作背景图像。这样如果一个网页上有3种宽度的圆角框，仅使用一套背景图像就都可以完成了。

（2）流动：含义是圆角框的宽度是跟随浏览器窗口宽度的变化而变化的。例如标准流方式的div的宽度就是自动横向伸展的，它的宽度就会随时改变。

（3）弹性：含义是圆角框的宽度在文字大小不变的情况下，宽度不变，而文字大小发生变化的时候，宽度随之改变。这里说的文字大小的变化是通过浏览器菜单中的“查看→文字大小”命令来改变的。

利用这3种不同的具体方式来设计“不固定”宽度的圆角框的原理都是一样的，只是在具体处理时对宽度设置方法有所不同。

（1）可变方式：实际上对于网页的访问者来说就是固定宽度，使用绝对单位（例如像素）来设置宽度。

（2）流动方式：使用百分比或者auto来设置宽度。

（3）弹性方式：使用相对单位（例如em）来设置宽度。

注意 对于流动方式要特别注意，现在的主流显示分辨率越来越大，20英寸以上的显示器普及率日益提高。在使用流动方式时，要确保不要使文本的一行过长，这样会影响访问者的阅读体验。大家可以数一数，一本标准的16开图书，就像现在看的这本书，一行的字数是40个字，这是很合适的一个行跨度。如果一行很长，读完以后，回到最左端的时候，可能会找不到下一行的位置，这无疑会使访问者的感受很不好。

下面从最简单的不固定宽度圆角框开始做起。

11.3.2 “4图像”单色不固定宽度圆角框

这是最直观的一种方法，基本思路和制作方法与前面介绍的固定宽度圆角框相似。本实例文件位于本书光盘“第11章\01\unfixe-dwidth.htm”。

准备4圆角的图像，如图11.21所示。

图11.21 案例中用到的背景图像

基本思路就是整体背景色用div的背景设置，然后4个图像分别作为某一个元素的背景，放置到适当的位置。

现在遇到的问题是，在HTML代码中，一共只有3个元素，即div、h3和p。因此，必须要在HTML中增加一个冗余的标记，这里有很多选择，例如，可以为标题增加一个a标记，连接到详细页面，或者在div外面套一层div，或者在p标记里面套一层span，这些方法都是可以的。这里使用在p标记里面套一层span的方法，代码如下。

```
<body>
  <div id="rounded">
  <h3>UNfixed Rounded</h3>
  <p>    <span>这是一个不固定宽度的圆角框，由于是宽度不固定，因此制作起来复杂一些。这个圆角框的上下随着内容增多可以自由伸展，圆角不会被破坏。此外，设置为不同的宽度，也同样适用。</span></p>
```

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

有了上一节的基础，先把div、h3和p的样式设置好，代码如下：

```
    <style type="text/css">

    #rounded{
      font: 12px/1.6 arial;
       background: #cba276 url('images/right-top.gif') no-repeat right
top;
        width: 200px;
        padding: 0;
        margin:0 auto;
      }
      #rounded h3 {
        background: url('images/left-top.gif') no-repeat;
        padding: 20px 20px 0;
        font-size: 170%;
        color: #FFF;
        margin: 0;
      }
      #rounded p {
        padding: 10px 20px 18px;
        color: #1B220F;
        margin: 0;
        text-indent:2em;
        background: url('images/left-bottom.gif') no-repeat left bottom;
    }
    </style>
```

这时的效果如图11.22所示。

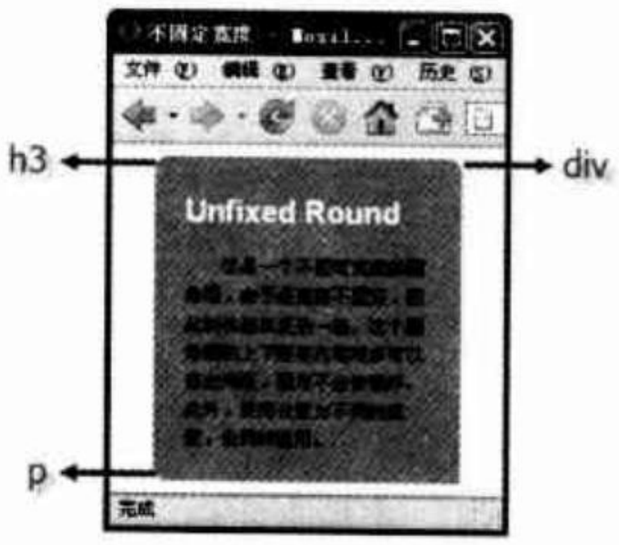


图11.22 各背景图像与HTML元素的对应关系

接下来增加一段对span的设置，代码如下。

```
      #rounded span{
            background:url('images/right-bottom.gif') no-repeat right
bottom;
            }
```

效果如图11.23所示，圆角没有出现在圆角框的右下角，而是出现在最后一个字的右下角。

图11.23 错误的背景图像位置

这显然是由于span是行内元素导致的，因此应将它设置为块级元素，代码如下。

```
    #rounded span{
          display:block;
          background:url('images/right-bottom.gif') no-repeat right
bottom;
          }
```

这时的效果虽然有所改进，但依然不正确，如图11.24所示，圆角没有出现在圆角框的右下角，而是出现在段落范围的右下角。

图11.24 仍然错误的背景图像位置

这里请读者自己先仔细阅读上面的4段CSS样式代码，想一想，为什么其他3个元素的背景图像都靠着最外侧，而这个span的背景却在里面呢？找到原因以后，又该如何修改呢？

CSS里面的逻辑要比HTML复杂得多。就像在中学学习数学的时候一样，遇到问题时，一定要自己先动脑子，只有自己想明白，才是真明白，因此不要着急看答案。

其实出现这个问题的道理很简单。在这个例子里，p的样式中设置了padding，因此span变成块级元素以后，它的范围就是p段落所占据的矩形除去padding以后的范围，因此它的背景图像就在里面了。

如何修改呢？只需把在p中对padding的设置移动到span中就可以了。这样p的padding和margin都是0，span的范围就扩展到了最外面。修改后的p和span的代码如下。

```
    #rounded p {
      color: #1B220F;
```

```
    margin: 0;
    text-indent:2em;
    background: url('images/left-bottom.gif') no-repeat left bottom;
    }
#rounded span{
    padding: 10px 20px 18px;
    display:block;
    background:url('images/right-bottom.gif') no-repeat right bottom;
    }
```

到这里，这个圆角终于放到了它应该在的位置，效果如图11.25所示。

图11.25 正确的背景图像位置

如果把div的宽度设置为70%，然后在不同宽度的浏览器中观看，效果如图11.26所示。可以看到它很好地适应了浏览器窗口宽度的变化。

图11.26 不同浏览器窗口宽度下的效果

11.3.3 “4图像滑动门”单色不固定宽度圆角框

在上一个例子中，上端和下端的颜色实际上是由左右两个图像和它们之间的背景色共同组成的。

这样，上下两端的颜色必须和整个背景的颜色相同。如果希望实现如图11.27所示的不固定宽度圆角框，用上一节的方法就不行了，因为如果用上一节的方法，就只有4个角的颜色是深色，而这里希望的是整个顶部和底部都是同样的颜色。

解决的方法是再一次使用“滑动门”技术。在前面的菜单部分，已经多次使用了这种技术。这种很重要的背景图像技术专门来实现适应宽度的需要。这里同样是为了适应圆角框的

宽度，因此“滑动门”技术也同样有用武之地。

图11.27 使用滑动门技术实现的圆角框

这里依然使用“探索式”的讲解过程，仍然使用上面的例子为基础，目的是把这个圆角框由使用4个图像变为使用两个图像。本实例文件位于本书光盘“第11章\01\slide-door.htm”。

假设仍然使用本章开头的固定宽度案例中的top.gif和bottom.gif，如图11.28所示。

图11.28 案例中用到的背景图像

然后把上一个例子中的左上角和右上角的图像文件都改为使用top.gif。同理，左下角和右下角的图像文件都改为使用bottom.gif，这时网页变成如图11.29所示的效果。

图11.29 出现的缺口现象

在上下两边上各出现了一个缺口，这说明如下情况。

● 当圆角框的宽度大于一个背景图像的宽度时，在上面的图像一端的圆角覆盖了下面图像，这样圆角处的白色小三角形就显出来了。

● 在上边，左边的图像是h3的背景图，右边的是div的背景图，h3是在div里面的元素，因此h3的背景图会盖在div的背景图的上面；而下边的左侧是p的背景图，右侧是span的背景图，span在p的里面，因此span的背景图会盖在p的背景图的上面。从而上下的缺口一个在左边，一个在右边。

为了使代码条理清晰，可以先把缺口交换到一边，例如把上面的缺口交换到左边，方法是交换#rounded和h3的背景图像和对齐方式，但是保持#rounded设置整体的背景色，代码如下。这时效果如图11.30所示。

```
#rounded{
  background:#cba276  url('images/top.gif') no-repeat;
  .
```

```
    font: 12px/1.6 arial;
    width: 70%;
    padding: 0;
    margin:0 auto;
    }
#rounded h3 {
    background: url('images/top.gif') no-repeat right top;
    padding: 20px 20px 0;
    font-size: 170%;
    color: #FFF;
    margin: 0;
    }
```

图11.30　将缺口调整到同一侧

接下来，为了不产生缺口，可以把右上角和右下角的图像换为如图11.31所示的样子。

图11.31　新的右上角和右下角的背景图像

这样就不会出现缺口了，然后把背景色设置为一种浅色，这时的效果如图11.32所示。

图11.32　设置完成后的效果

注意　采用这种方法时需要注意，虽然宽度可以扩展，但是不能无限扩展，如果超过了总的背景图像的宽度，就会露出底下的背景色了，如图11.33所示。

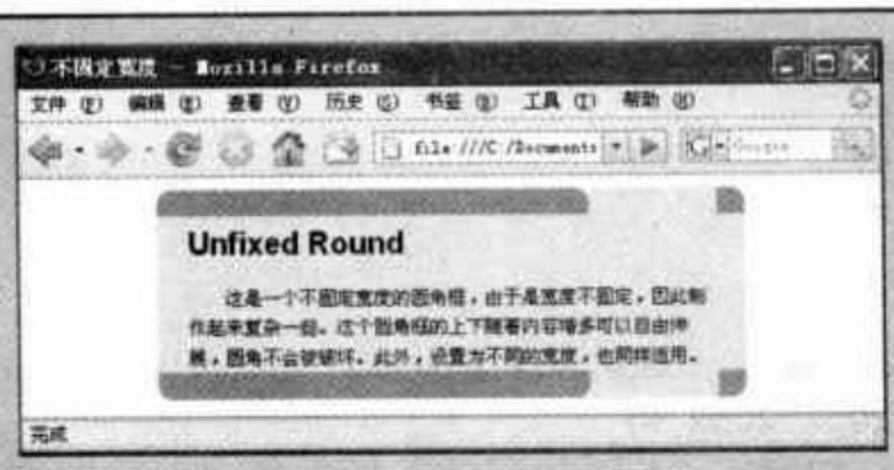


图11.33 当圆角框过宽时出现裂缝

因此，左面的长条可以做得长一些，使圆角框能够适应更宽的宽度。

11.4 “5图像”二维滑动门经典圆角框

11.3节介绍的不固定宽度的圆角框制作方法只适用于单色的圆角框，对于带有复杂边框的圆角框就不适用了。

另外，还需要考虑一个问题，用11.3节介绍的方法制作的圆角框，尽管可以适应不同宽度，但是它里面的正文段落也承担了设置圆角框布局的任务。也就是说，圆角框中的正文并不是独立的，正文内容不能自由地设置样式。这对于通用型的网页来说，局限性很大。例如，如图11.34所示的页面，每个圆角框用于放置不同的栏目，希望达到的目标应该是可以在每个圆角框中放置任何内容，并且可以对这些内容设置样式，还能保证圆角框本身显示正确。简单地说，就是使圆角框的本身与内容完全分离。

图11.34 使用圆角框的复杂页面

这里向读者介绍一种非常经典的利用5个背景图像制作圆角框的方法。这种方法可以实现上面说的要求。

这种方法是由丹麦的设计师Søren Madsen，于2003年12月发表在著名的Web设计与开发电子杂志“A List Apart”的第165期上。

A List Apart的网址是http://www.alistapart.com，Søren Madsen本人的网站网址是http://www.picment.com。

11.4.1 准备图像

① 首先在Photoshop或者Fireworks中绘制一个大约800×600的圆角矩形，如图11.35所示。

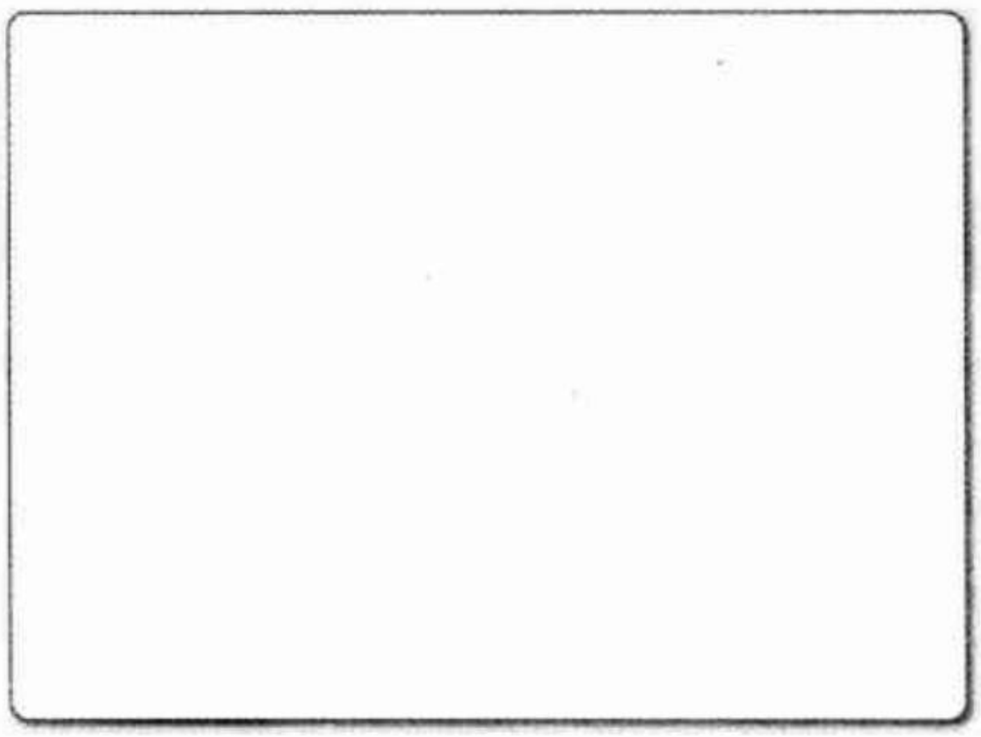


图11.35 在图像处理软件中绘制的圆角框效果

注意 具体的样式和大小读者可以自己决定，最终完成的圆角框的大小不能超过这个大小，如果超过，就会出现裂缝。因此如果需要很大的圆角框，这个图就要做得再大一些。

② 在Photoshop或者Fireworks中进行切片，一共产生5个图像文件，如图11.36所示。

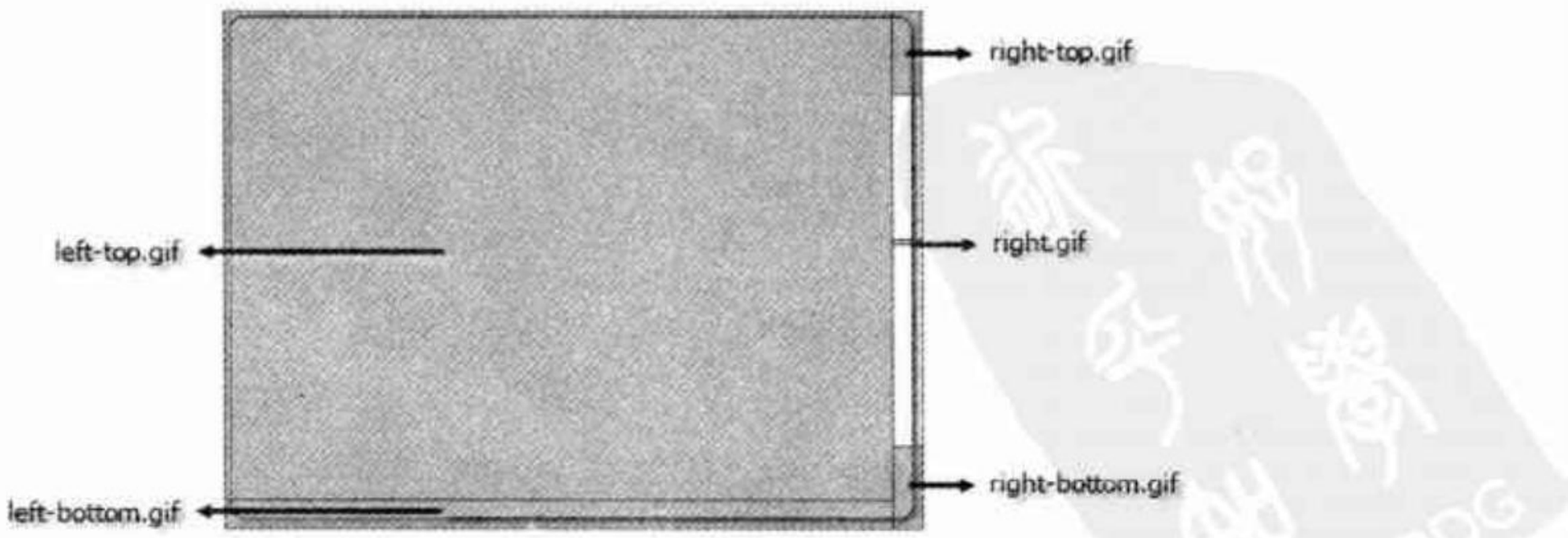


图11.36 图像切片示意图

最终产生的5个图像文件如图11.37所示。图中的各个图像不是按实际比例显示的，实际上，左上角的图像比其他的大很多。

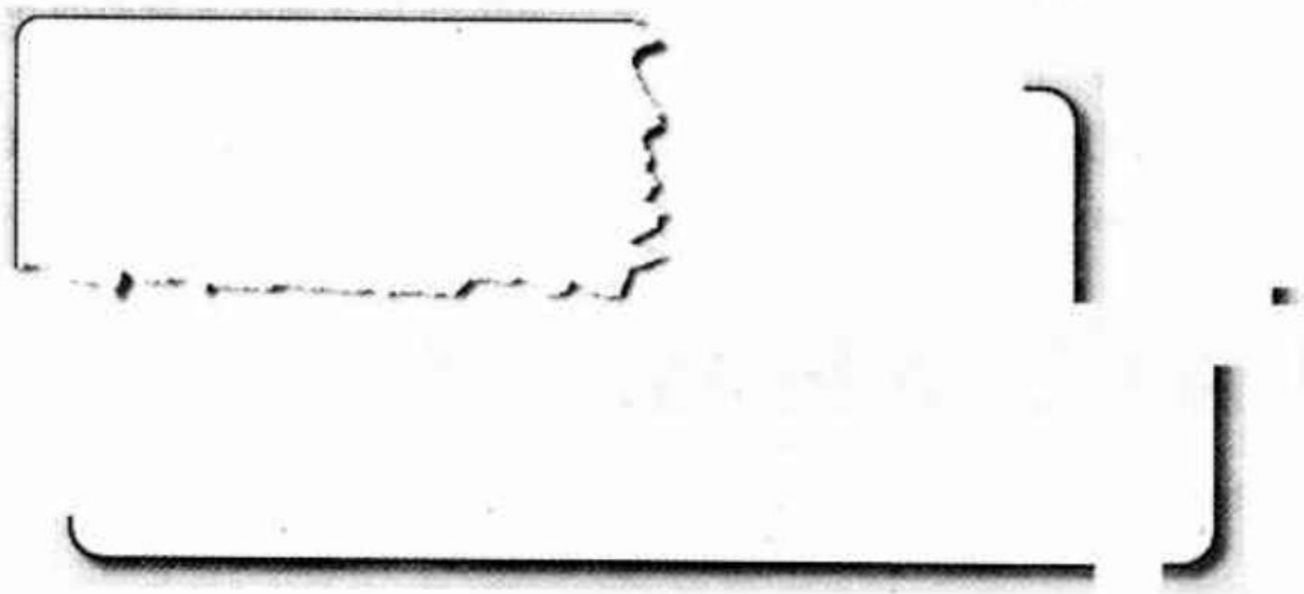


图11.37 最终产生的5个图像文件

11.4.2 搭建HTML结构

接下来搭建HTML整体结构，代码如下。代码中临时设置了一些实线边框，用来确认各个盒子的位置和大小，以便进行分析，后面会把这些边框的属性去掉。该文件位于本书光盘“第11章\03\step-1.htm”。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=" http://www.w3.org/1999/xhtml" >
<head>
<title>5背景图可变宽圆角框</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<style type="text/css">
body {
background: #FFFF99;
font: 12px/1.5 Arial;
}
.rounded {
   width:90%;
   border: 1px solid red;
    }
.rounded h3 {
    border: 1px solid blue;
   }
.rounded .main {
   border: 1px solid black;
   }
.rounded .footer {
   border: 1px solid blue;
   }
.rounded .footer p {
   border: 1px solid magenta;
   }
```

```
</style>
</head>
<body>
<div class="rounded">
        <h3>Article header</h3>
        <div class="main">
                <p>
                这是一行文本，这里作为样例，显示在圆角框。<br />
                这是一行文本，这里作为样例，显示在圆角框。
                </p>
                <p>
                这是一行文本，这里作为样例，显示在圆角框。<br />
                这是一行文本，这里作为样例，显示在圆角框。
                </p>
        </div>
        <div class="footer">
                <p>
                这是版权信息文字。
                </p>
        </div>
  </div>
</body>
</html>
```

在上述代码中，定义了一个div容器，里面有一个标题和两个div，这两个div中，前者为内容主体，后者为页脚。主体中有两段文本，页脚中有一段文本。整个div设置为浏览器宽度的90%，并且给每个元素设置边框，这是为了先看清楚整体的结构，效果如图11.38所示。

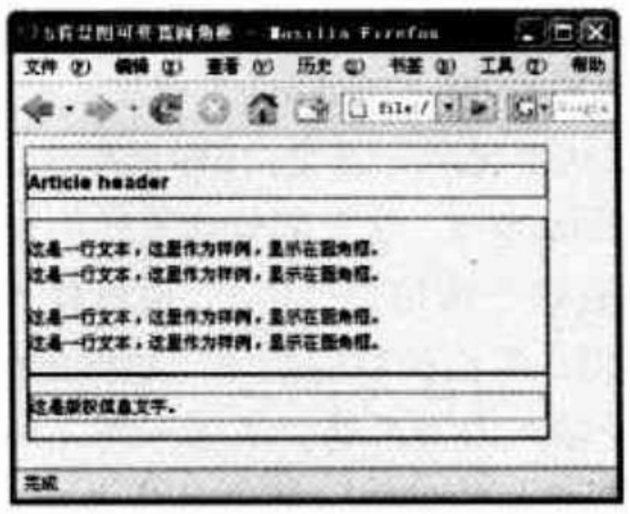


图11.38 各HTML元素所占的区域

11.4.3 放置背景图像

现在的任务就是把前面制作的5个图片分别放置到5个元素的背景中。先不考虑更多细节，仅把它们放进去即可，相关代码如下。该文件位于本书光盘“第11章\03\step-2.htm”。

```
.rounded {
    background: url(images/left-top.gif) top left no-repeat;
   width:90%;
    }
.rounded h3 {
    background: url(images/right-top.gif) top right no-repeat;
```

```
  }
.rounded .main {
  background: url(images/right.gif) top right repeat-y;
  }
.rounded .footer {
  background: url(images/left-bottom.gif) bottom left no-repeat;
  }
.rounded .footer p {
  background: url(images/right-bottom.gif) bottom right no-repeat;
  }
```

这时效果如图11.39所示。

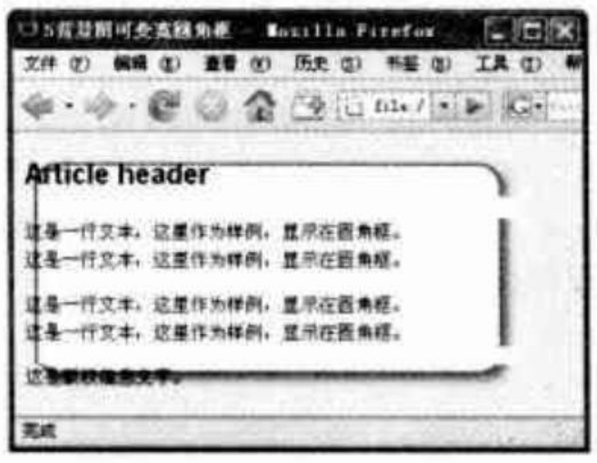


图11.39 使用背景图像以后的效果

现在基本圆角框已经构成了。但是还有两个问题，第一是文字没有放到圆角框内部，第二是右侧的边框有两个裂缝。下面就来分析并解决这个问题。

11.4.4 设置样式并修复缺口

修改文字的位置可以通过设置margin和padding来解决。这两个裂缝产生的原因是什么呢?

它是由p标记产生的。在默认情况下，每个由p标记产生的段落在第一行的上面和最后一行的下面都会有一个自动设置的margin，这个部分会盖住边框。解决办法有两个。

一种方法是重新设置放入到这个圆角框的中的p段落样式。实际上前面的例子中也是这样做的，但是从更高的要求来说，我们的目的是使圆角框的结构和它的CSS样式可以和内容完全分离开，而正文的p标记是输入内容范畴，不属于圆角框本身，也就是最好不需要对置入圆角框的p进行任何与圆角框相关的设置。从设计的角度来说，这样做更加优雅。因此，前面设置的5个背景图片都没有挂在正文的p标记上，也是出于同样的原因。本章前面的所有例子都没有做到的一点。

另外一种方法是不需要对p进行设置。在这种情况下，又该怎么解决呢？其实方法也很简单，只需将margin设置为负值，即向上提高裂缝的高度，就可以盖住这个裂缝。理论上提高1行文字的行高就可以了，但是Søren Madsen经过多次尝试后，发现如果仅提高1个行高的高度，在CSS支持得较为完善的浏览器中显示是没有问题的，但是对于IE浏览器，再多提高一点更合适。例如在这个例子中，行高设定的是文字高度的1.5倍，把上margin设置为-2em，就可以得到完美的效果。

请参考下面的代码，特别注意粗体字的7行代码。该文件位于本书光盘“第11章\03\step-3.htm”。

```
.rounded {
   background: url(images/left-top.gif) top left no-repeat;
   width:90%;
   }
.rounded h3 {
   background: url(images/right-top.gif) top right no-repeat;
   padding:20px 20px 10px;
   margin:0;
   }
.rounded .main {
   background: url(images/right.gif) top right repeat-y;
   padding:10px 20px;
   margin:-2em 0 0 0;
   }
.rounded .footer {
   background: url(images/left-bottom.gif) bottom left no-repeat;
   }
.rounded .footer p {
   background:url(images/right-bottom.gif) bottom right no-repeat;
   display:block;
   padding:10px 20px 20px;
   margin:-2em 0 0 0;
  }
```

这时显示效果如图11.40所示，圆角框的宽度是浏览器窗口宽度的90%，可以随浏览器窗口变化。

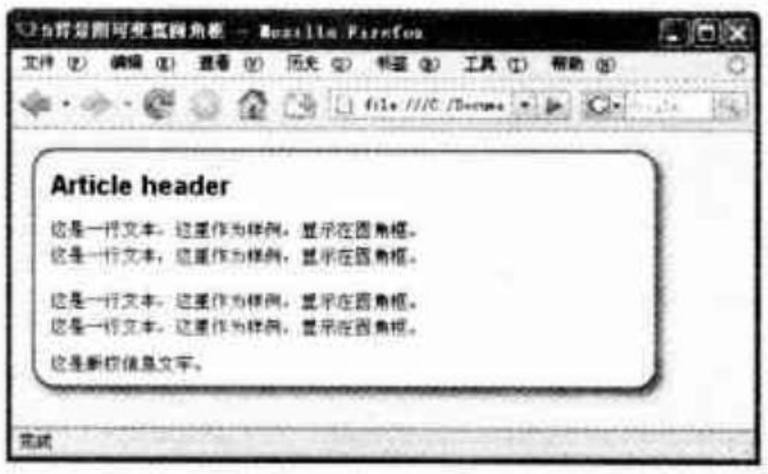


图11.40 设置完成以后的效果

注意 如果浏览器窗口变得过大，有可能会超过背景图像的范围，产生裂缝，如图11.41所示，因此在制作背景图片的时候，需要先考虑好需要多大的图像。

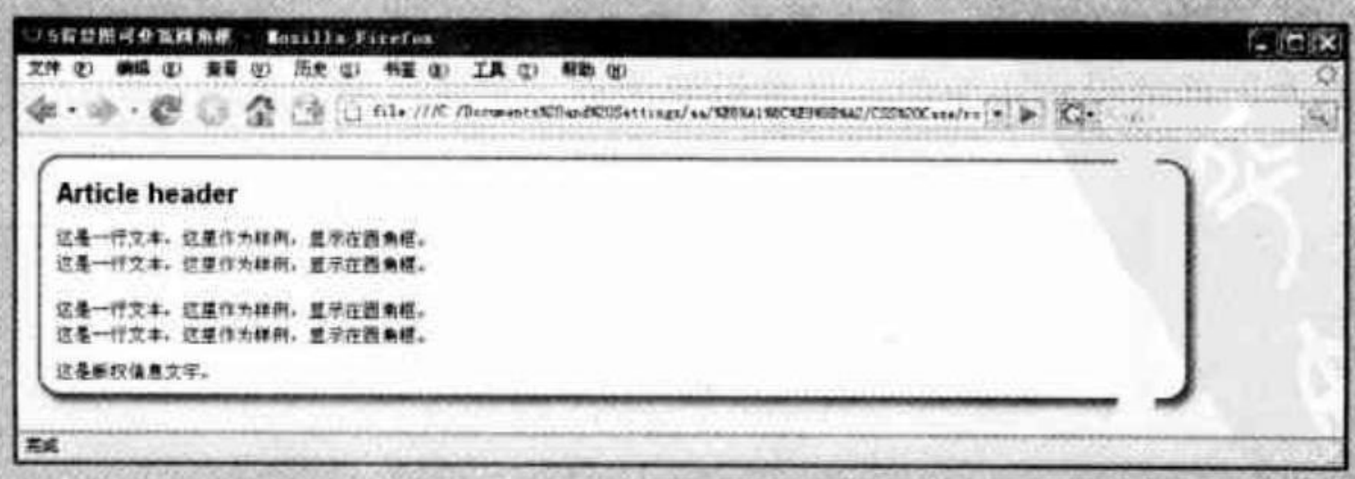


图11.41 当浏览器窗口过宽时产生裂缝

通过上面的介绍可以了解到，制作圆角框的技术是很相似的，都是通过使用若干个背景图片来实现的。各种方法的区别就在于如何分配各个图像所在的元素。读者在互联网上可以搜索到大量的不同方案。

11.4.5 在整体页面中使用圆角框

制作圆角框的目的不是仅仅做出一个圆角框就可以了，更重要的是能在一个页面中灵活地重复使用。下面演示一下如何使用上面制作的圆角框，方便地组合成一个完整的网页，如图11.42所示。在后面两章中，还将基于这里制作的圆角框，设计出各种各样复杂的布局页面，这里仅作一个简单的预览。该文件位于本书光盘“第11章\03\whole-page.htm”。

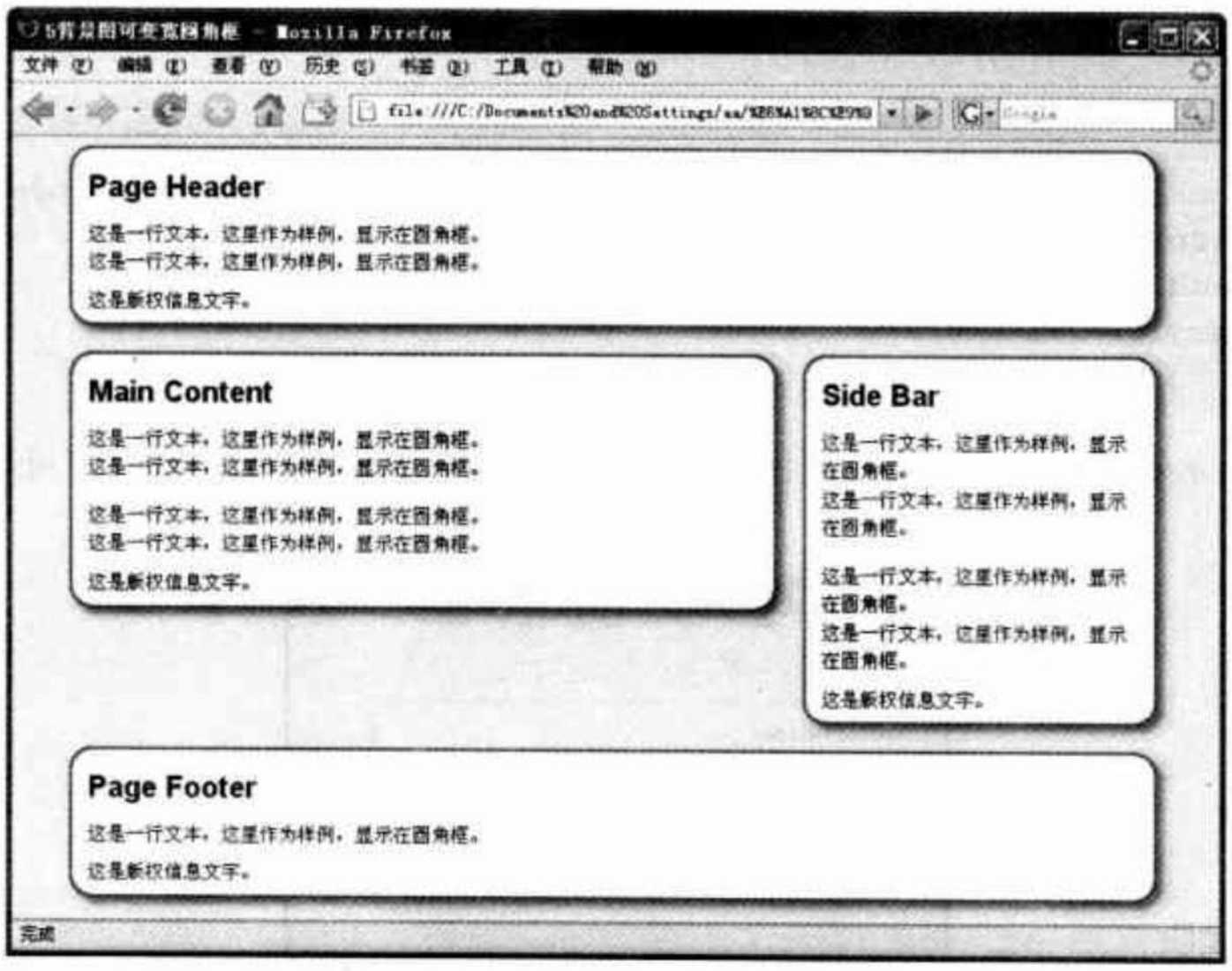


图11.42 在一个页面中组合使用圆角框

① 这个页面一共使用了4个上面制作的圆角框。把圆角框中与宽度有关的设置去掉，即在.rounded中，将“width:90;”这行去掉，因为这里每个圆角框的宽度都是100%的，不用特别设置。

② 考虑一下整体的布局。整个页面分为上中下3个部分，使用宽度固定且居中的布局方式。上部和下部的设置很简单，只要设置为固定宽度，然后通过margin设置为居中就可以了。

```
#header , #pagefooter {
margin:0 auto;
width:760px;
}
```

③ 中间部分为宽窄两部分，需要把它们放到一个div容器中。先对这个容器进行设置。

```
#container{
  position:relative;
  margin:0 auto;
  width:760px;
}
```

④ 将position属性设置为relative，目的是为了让它里面的div可以以它为基准进行绝对定位。设置margin实现居中，最后设定宽度与上下两个部分相同。

⑤ 将容器中左边的圆角框设置为绝对定位，宽度是500像素，代码如下。

```
#wide{
position:absolute;
top:0;
left:0;
width:500px;
}
```

⑥ 设置右边的窄栏，使用标准流方式即可，左边使用margin留出左边圆角框的位置。

```
#side{
margin:0 0 0 500px;
}
```

这里用的都是id选择器，因为每个圆角框在页面中都是惟一的。HTML代码如下。

```
<body>
<div id=" header"  class=" rounded" >
        <h2>Page Header</h2>
        <div class=" main" >
              <p>
              这是一行文本，这里作为样例，显示在圆角框。<br />
              这是一行文本，这里作为样例，显示在圆角框。
              </p>
        </div>
        <div class=" footer" >
              <p>
              这是版权信息文字。
              </p>
        </div>
 </div>

<div id=" container" >
 <div id=" wide"  class=" rounded" >
        ……省略圆角框的内容，请复制前面的代码……
 </div>
 <div id=" side"  class=" rounded" >
        ……省略圆角框的内容，请复制前面的代码……
 </div>
</div>

<div id=" pagefooter"  class=" rounded" >
        ……省略圆角框的内容，请复制前面的代码……
```

```
</div>

</body>
```

注意 这里的每个圆角框的代码都可以直接复制，然后修改里面的标题和正文内容即可，这里我们就省略具体内容了，读者理解清楚他们之间的关系即可。上面代码中，有3处粗体“……**省略圆角框的内容，请复制前面的代码**……”，它们的结构都和第一段粗体代码完全相同，而内容可以随意修改。

可以看到，4个圆角框用的都是相同的代码结构，每一个圆角框都同时有一个class属性和一个id属性，class属性用于指定rounded类别的样式，id属性用于指定定位和布局的样式。

11.4.6 实现网页换肤

通过上面的讲解，已经可以非常明显地看出使用CSS布局的优势。如果使用表格来进行同样的布局，所需要的图片量会远远超过现在这种方式，而且现在可以随意修改各个圆角框的宽度、位置和样式，而使用表格是无法做到的。

使用这种方法实现的页面可以很轻松地实现“换肤”效果，例如上面的页面中，可以进更换5个背景图片文件，将页面改为如图11.43所示的“便笺簿”效果。

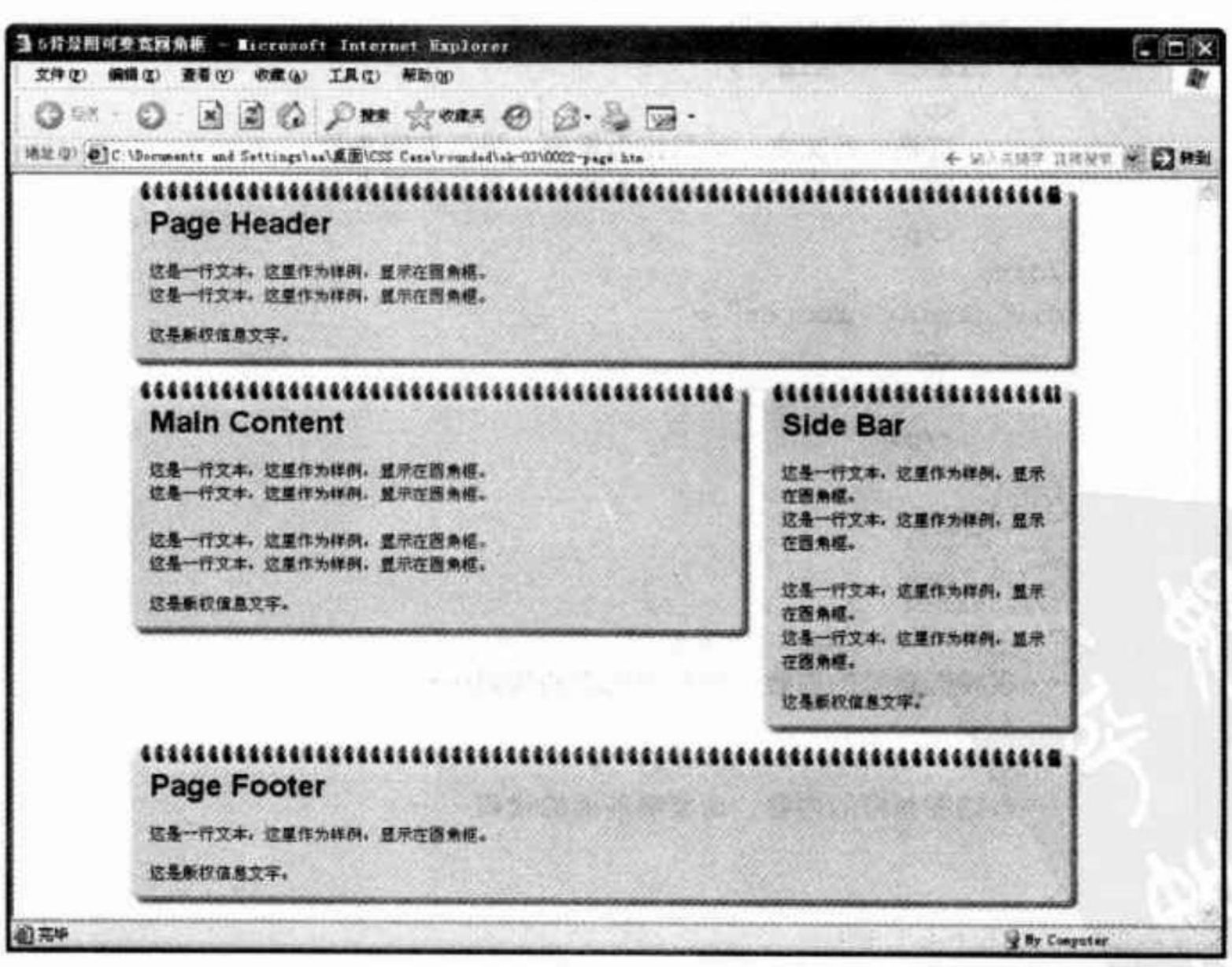


图11.43 将网页整体“换肤”为“便笺簿”效果

每种设计方案都需要先做好一个圆角框的图形设计，这种“便笺簿”的图像设计如图11.44所示。

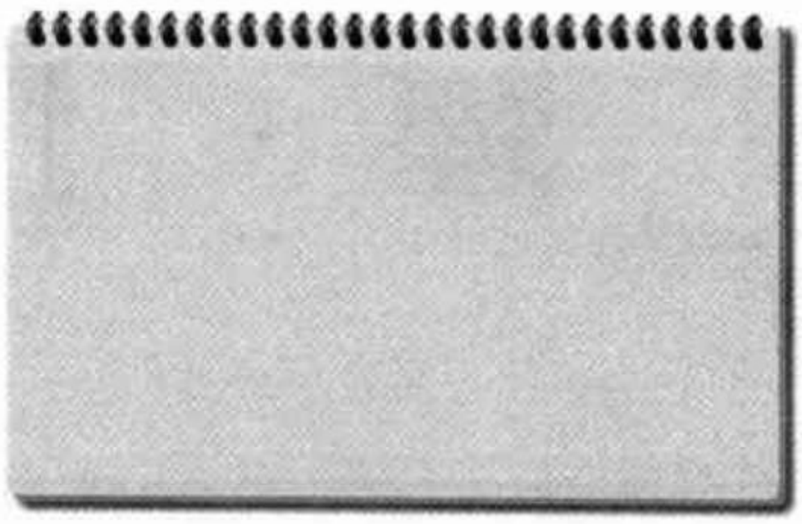


图11.44 设计“便笺簿”效果所需的图像

也可以设计深颜色的边框，效果如图11.45所示。

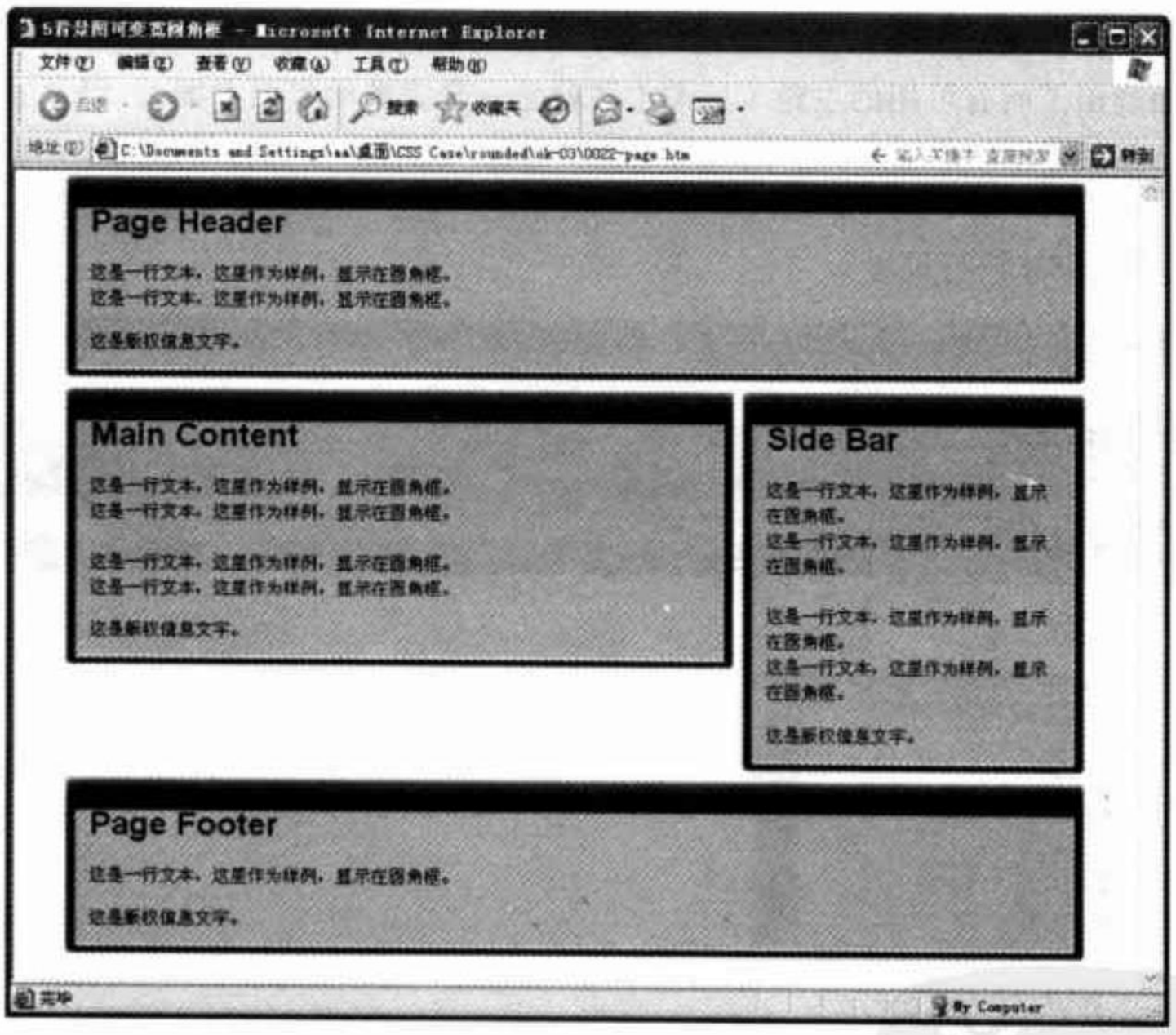


图11.45 将网页整体“换肤”为深色边框效果

本书光盘中放置了4个换肤的例子，请参考本书光盘“第11章\03\page-with-skin”文件夹中的4个网页文件。

在上面案例的代码中，5个背景图像都放置在了images文件夹中，在实际制作网站的时候，可以按照如图11.46所示的方式组织文件结构，为每一种“皮肤”设置一个文件夹，然后将所有的“皮肤”文件夹放在一个总的文件夹中。

如果是手工修改来换肤，则可以不用修改样式表，直接把要使用的那种皮肤的文件夹名称改为样式表中使用的路径即可，这样工作量比较小。

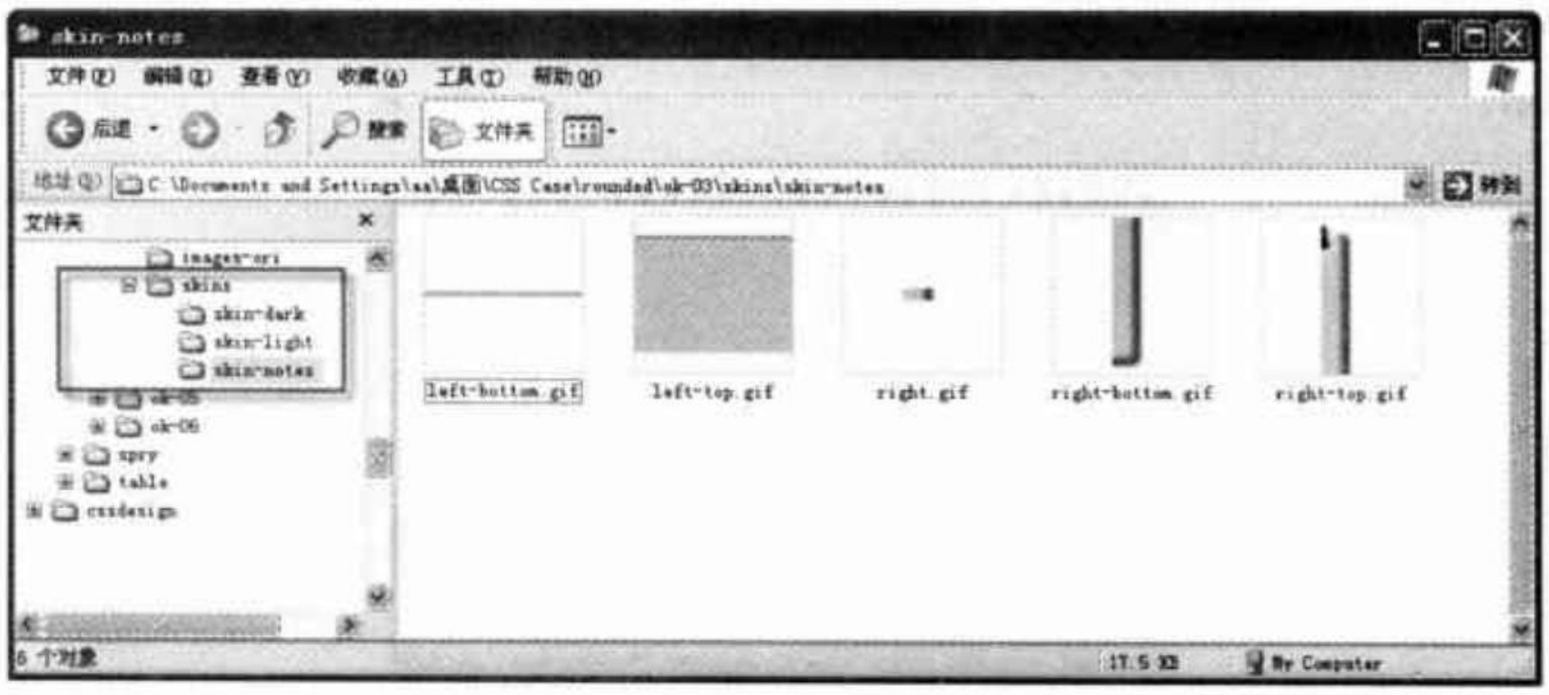


图11.46 合理组织网站的文件结构

如果是动态的程序，例如制作一个博客网站，则可以在服务器端记住需要哪种皮肤，动态修改文件内容。例如图11.47所示的是目前非常流行的博客系统“Wordpress”的后台管理界面，里面列出了所有可用的主题（所谓主题就是一套完整的界面方案），以及当前正在使用的主题，并且可以任意切换所要使用的主题。当然，这样非常完善的系统中，它的主题远远不是通过更换几个背景图片来实现的，还包括布局、样式和图像等都是主题的一部分。当然核心还是用CSS来进行设置。

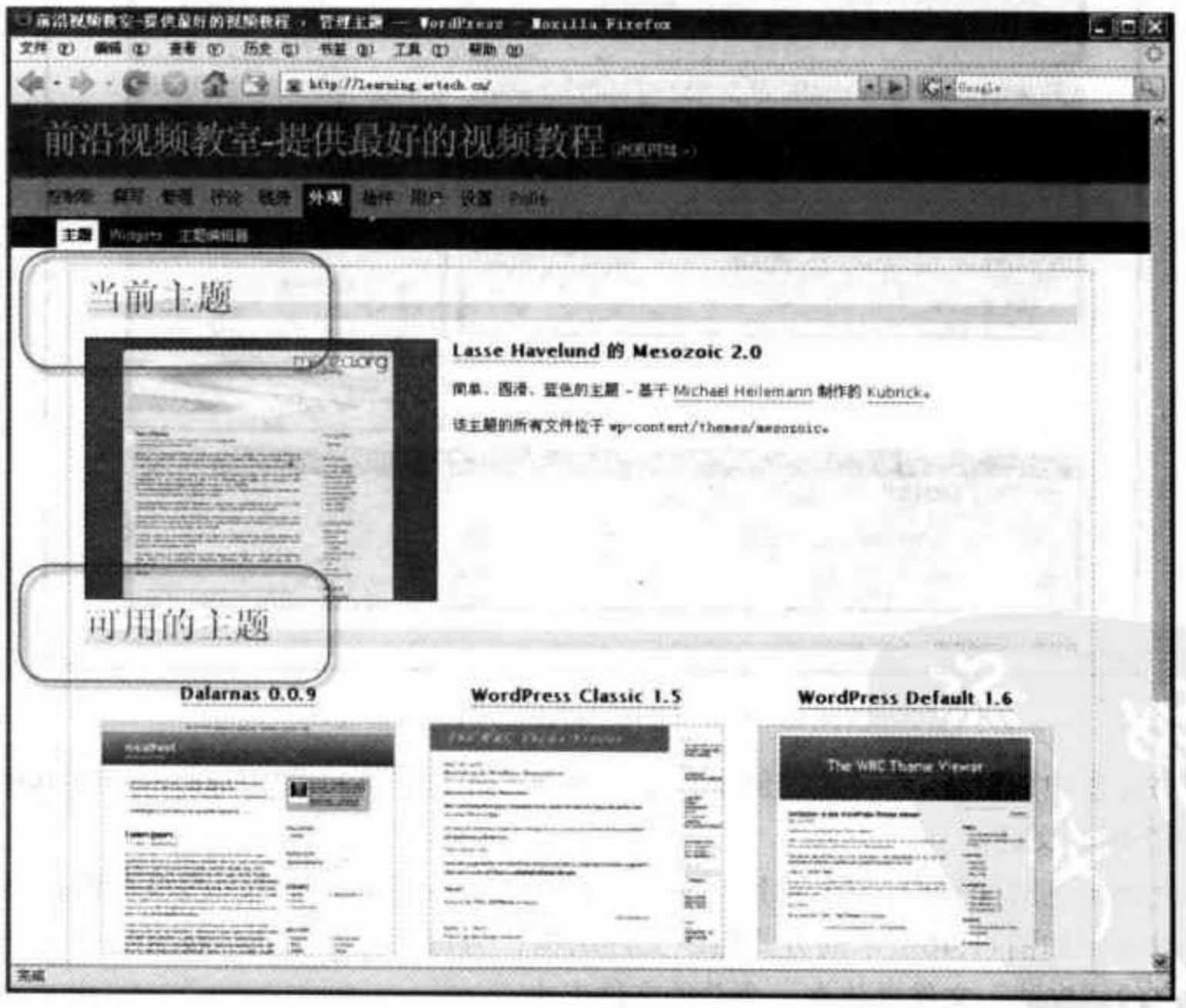
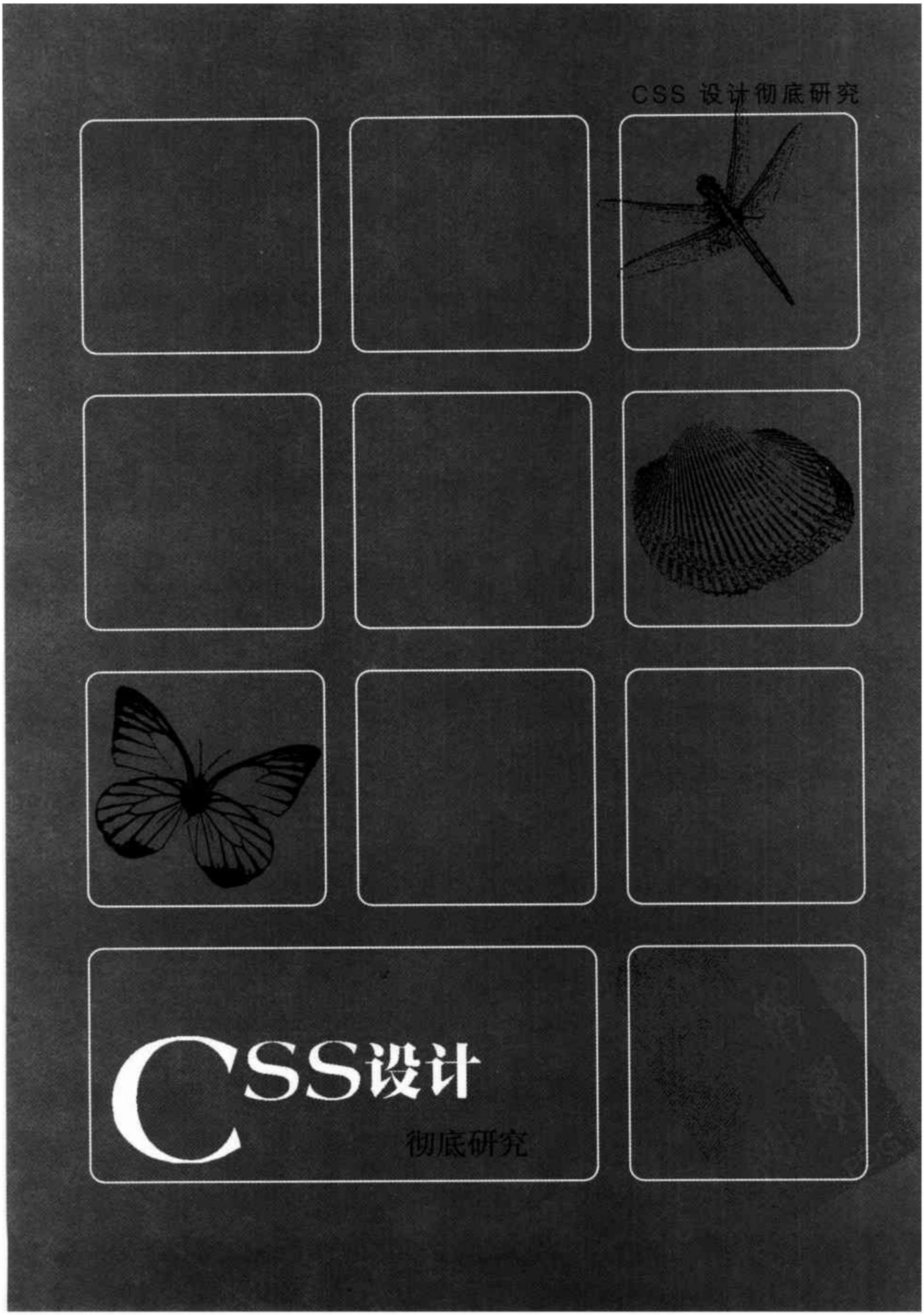


图11.47 在Wordpress系统中可以灵活地设置主题

11.5 本章小结

本章重点讲解了制作比较简单的固定宽度的圆角框的方法，以及比较复杂的不固定宽度的圆角框的方法。

特别是比较详细地介绍了非常经典的Søren Madsen提出的使用5个背景图像的经典通用圆角框的制作方法，并给出在整体页面中的使用案例。在后面的两个章节中，将深入介绍整页布局的方法。在整页布局时，我们都将使用这种方法来实现局部的效果。所以读者要把本章的内容真正搞懂，为下一章的学习打好基础。



第 12 章
应用Spry制作高级网页组件

这里所说的高级网页组件是指一些网页上的比较复杂的用户界面，例如Tab面板、伸缩面板和折叠面板等组件。

Tab风格的面板一直受到广大网站制作者的青睐。Tab面板有时被称为“选项卡”面板，使用鼠标点击各个Tab选项卡就可以在不同页之间切换。

12.1 Tab菜单与Tab面板简介

网上随处可见各式各样的Tab菜单，图12.1所示的是雅虎网站首页上可以看到的Tab面板，网址是http://www.yahoo.com。

图12.1 雅虎网站上的Tab面板

图12.2显示的是微软网站上的Tab菜单，网址是http://office.microsoft.com。

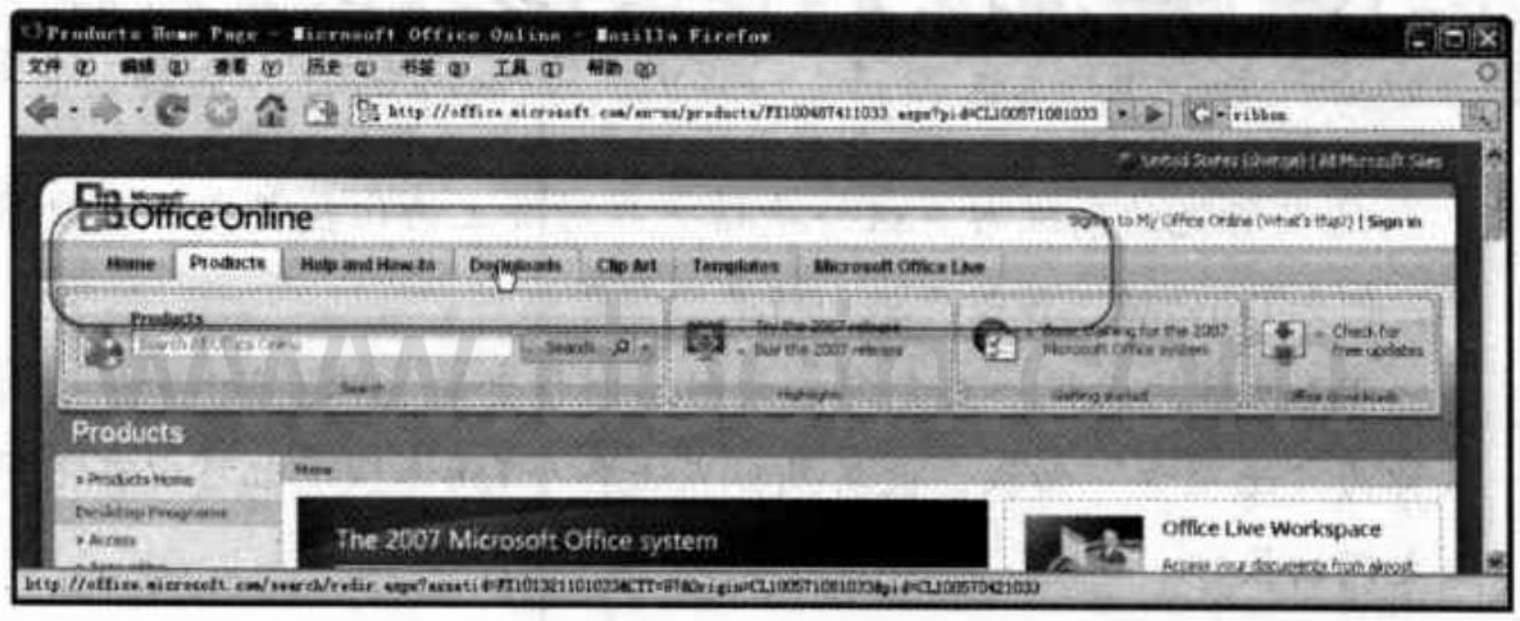


图12.2 微软网站上的Tab菜单

此外，还有很多网站都使用了与它们类似的网页元素。如果读者亲自试验一下，就会发现，这些网站上的Tab面板和菜单实际分为两种。

- 一种是切换各个Tab页中的内容时并不刷新浏览器窗口，说明各个页中的内容实际已经装载到页面上了，只是被隐藏了起来，只有被选中的Tab页内容被显示出来。上面例子中的yahoo.com的页面属于这种类型。
- 另一种是切换各个Tab页中的内容时，会刷新浏览器窗口，实际上就是更换了一个新的HTML页面。上面例子中的微软网页属于这种情况。

要实现前者的效果，必须通过JavaScript的配合，仅通过CSS来实现是比较困难的，而且实现的功能非常有限。

实际上，也有一些网站使用Ajax技术，可以实现切换到某一页之后，在动态地从服务器

上获取数据，局部刷新Tab页内的区域，而整个页面的其他部分则不需要重新从服务器获取数据。

本章将实现JavaScript配合的“隐藏/显示”方式的Tab面板，但对Ajax方式的Tab页不进行介绍，如果读者有兴趣，可以查找相关资料。

为了便于本章后面的讲解，将前一种方式（浏览器无刷新）制作的效果称为“Tab面板”，因为它是一个页面中的一个区域；而后一种方式（浏览器刷新）制作的效果称为“Tab菜单”，因为它实际上就是多个页面之间的导航菜单。

12.2 Tab菜单

在介绍制作“隐藏/显示”方式的不刷新Tab面板之前，先来制作一个比较简单的Tab菜单效果，也就是先从简单的方式引入，为后面的案例做准备。本案例介绍的方法是将每个Tab页作为一个独立的网页文件来实现的。

这个Tab菜单的制作方法与前面介绍的菜单制作方法类似，不同之处在于需要设置一些特殊的CSS效果来实现“Tab”的样式。菜单本身仍然是项目列表的扩展，本例的最终效果如图12.3所示。本实例的文件位于本书光盘“第12章\01\home.htm”。

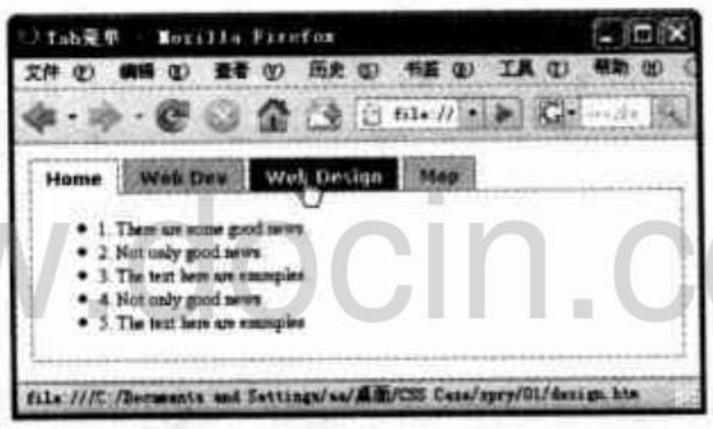


图12.3 Tab菜单

12.2.1 搭建HTML结构

首先跟所有的菜单一样建立相应的项目列表，所不同的是因为有多个页面，所以需要给每个项目都定义一个CSS类型，并为<body>标记分配各自的id，代码如下所示。

```
<body id="home">
<ul id="tabnav">
  <li class="home"><a href="home.htm">Home</a></li>
  <li class="dev"><a href="dev.htm">Web Dev</a></li>
  <li class="design"><a href="design.htm">Web Design</a></li>
  <li class="map"><a href="map.htm">Map</a></li>
</ul>
<div id="content">
  <ul id="list">
```

```
            <li>1. There are some good news.</li>
            <li>2. Not only good news.</li>
            <li>3. The text here are examples.</li>
            <li>4. Not only good news.</li>
            <li>5. The text here are examples</li>
    </ul>
</div>
</body>
```

除了每个页面的具体内容，即“<div id="content>"”中包含的部分以外，所有页面的整体框架都是完全相同的。每个页面都采用<link>语句调用相同的CSS外部文件，而页面的具体内容所采用的CSS，则放在页面内用<style>标记控制，代码如下所示。

```
<link href="tab.css" type="text/css" rel="stylesheet">
```

12.2.2 设置整体的样式

在外部的tab.css文件中定义各个CSS属性。首先给正文的内容“#content”添加蓝色的边框，但是只添加左侧、右侧和下端，空出上端。此时的页面效果如图12.4所示。

```
body{
  margin:10px;
  }
#content{                                    /* 具体内容 */
  border-left:1px solid #11a3ff;             /* 左边框 */
border-right:1px solid #11a3ff;              /* 右边框 */
border-bottom:1px solid #11a3ff;             /* 下边框 */
padding:5px;
font-size:12px;
}
```


图12.4 正文边框

12.2.3 设置Tab的样式

现在设置<ul>标记的CSS属性。除了将项目符号隐藏外，还要为其添加下边框，用来当

作正文内容的上边框，代码如下所示。这样在<ul>标记中设置的边框便可以被稍后设置的<li>标记中的边框所覆盖，从而实现Tab的效果。此时页面的效果如图12.5所示。

```
ul#tabnav{
  list-style-type:none;
  margin:0px;
  padding-left:0px;                                    /* 左侧无空隙 */
  padding-bottom:23px;
  border-bottom:1px solid #11a3ff;                     /* 菜单的下边框 */
  font:bold 12px verdana, arial;
}
```

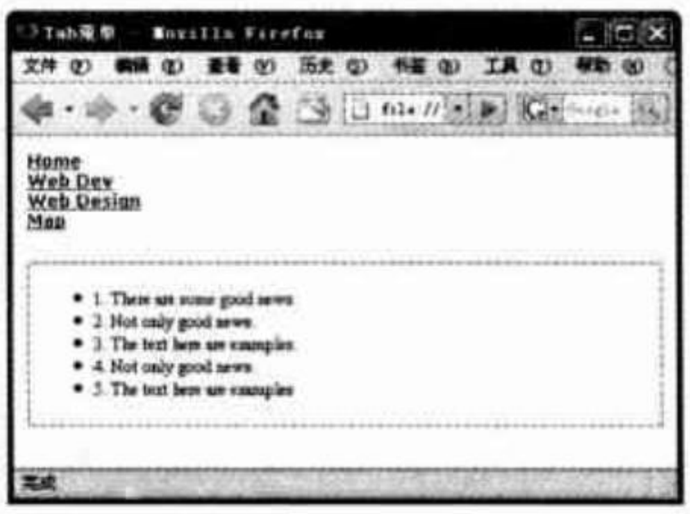


图12.5 设置<ul>样式

接着设置<li>标记的样式，将所有的列表项水平排列，设置相应的背景颜色和边框等，并通过margin属性适当地调整其位置，代码如下所示。此时页面的效果如图12.6所示，可以看到Tab菜单已经初现雏形。

```
ul#tabnav li{
  float:left;
  height:22px;
  background-color:#a3dbff;
  margin:0px 3px 0px 0px;
  border:1px solid #11a3ff;
  }
```

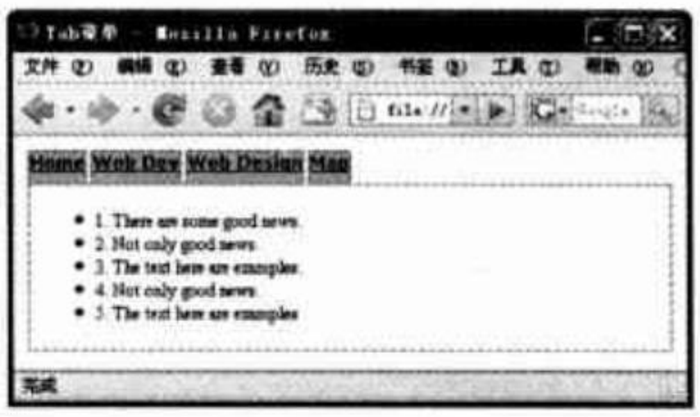



图12.6 设置<li>样式

设置所有超链接的3个伪类别。同样将<a>设置为块元素，并配合页面的整体色调和Tab菜单的大小等，代码如下所示。此时页面的效果如图12.7所示。

```
ul#tabnav a:link, ul#tabnav a:visited{
  display:block;                                       /* 块元素 */
```

```
    color:#006eb3;
    text-decoration:none;
    padding:5px 10px 3px 10px;
    }
  ul#tabnav a:hover{
    background-color:#006eb3;
    color:#FFFFFF;
    }
```

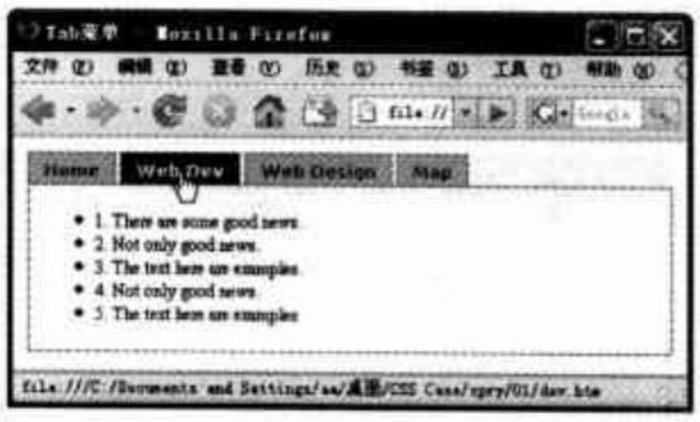


图12.7 设置<a>的伪类别

12.2.4 设置当前页的Tab样式

由于为每个页面的<body>标记都添加了惟一的id，因此可以设置当前页面的菜单项，如home.htm的“Home”菜单等，代码如下所示。关键在于给当前页面的菜单项添加白色的下边框，从而覆盖了<ul>标记中设置的蓝色下边框，实现了Tab菜单的效果。此时页面的效果如图12.8所示。

```
    body#home li.home,
    body#dev li.dev,
    body#design li.design,
    body#map li.map{
      border-bottom:1px solid #FFFFFF;   /* 白色下边框，覆盖<ul>中的蓝色下边框
*/
      color:#000000;
      background-color:#FFFFFF;
      }
```

图12.8 设置当前页面的<li>项

现在已经完成了Tab菜单的核心部分，下面为当前页面的菜单项添加单独的超链接效果，以区别于其他页面，代码如下所示。此时页面的效果如图12.9所示。

```
body#home li.home a:link, body#home li.home a:visited,
body#dev li.dev a:link, body#dev li.dev a:visited,
body#design li.design a:link, body#design li.design a:visited,
body#map li.map a:link, body#map li.map a:visited {
  color:#000000;
  background-color:#FFFFFF;
  }
body#home li.home a:hover, body#dev li.dev a:hover,
body#design li.design a:hover, body#map li.map a:hover {
color:#006eb3;
text-decoration:underline;
}
```

图12.9 当前页面菜单项的超链接

最后还可以为每个页面添加相应的内容。内容部分使用的CSS与公共的CSS样式文件分别存放，可以使用<style>嵌入到页面中，代码如下所示。

```
<link href="tab.css" type="text/css" rel="stylesheet">
<style type="text/css">
/*各个页面单独的具体内容所使用的CSS */
</style>
```

本案例重点介绍了如何制作Tab菜单，因此每个页面内部的样式就不再进行设置了，这样整个Tab菜单模块便制作完成了。

12.3 借助于Spry实现Tab面板

前面说过，如果要实现一个真正的Tab菜单，需要JavaScript配合，开发起来必须有JavaScript的使用经验。没有JavaScript经验的设计师可以借助于Adobe公司2007年春天发布的Dreamweaver CS3中新增加的Spry功能来方便地实现。

在本案例中，我们最终实现的效果如图12.10所示。当用鼠标单击某个Tab时，就会切换到该Tab页。

本实例的文件位于本书光盘“第12章\02\tab.htm”。该目录下还有一个tab-hover.htm文件。

二者的区别是，在tab.htm中需要鼠标单击才能实现换页，而在tab-hover.htm中，鼠标指针经过某个tab就可以切换到该Tab页，而无需单击鼠标。

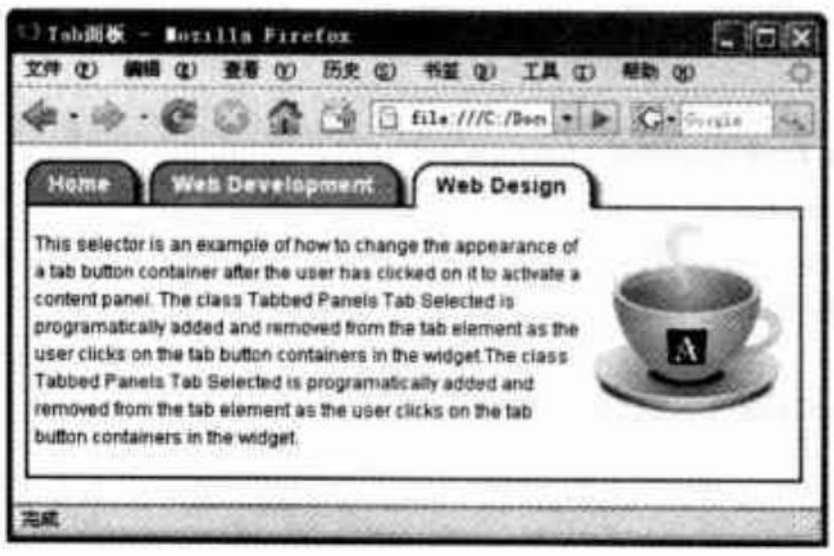


图12.10 当前页面菜单项的超链接

12.3.1 建立基本的Tab面板

Dreameavaer CS3中的Spry是一套丰富的JavaScript框架，可以大幅度简化Web开发的复杂度，提高开发效率。

① 首先安装好Dreamweaver CS3软件，创建一个新文档。先保存这个文件，然后在窗口上侧的“插入”工具栏中选择“Spry”项目。单击从右数第3个按钮，如图12.11所示。这时在视图窗口中就会出现一个最基本的Tab面板了。

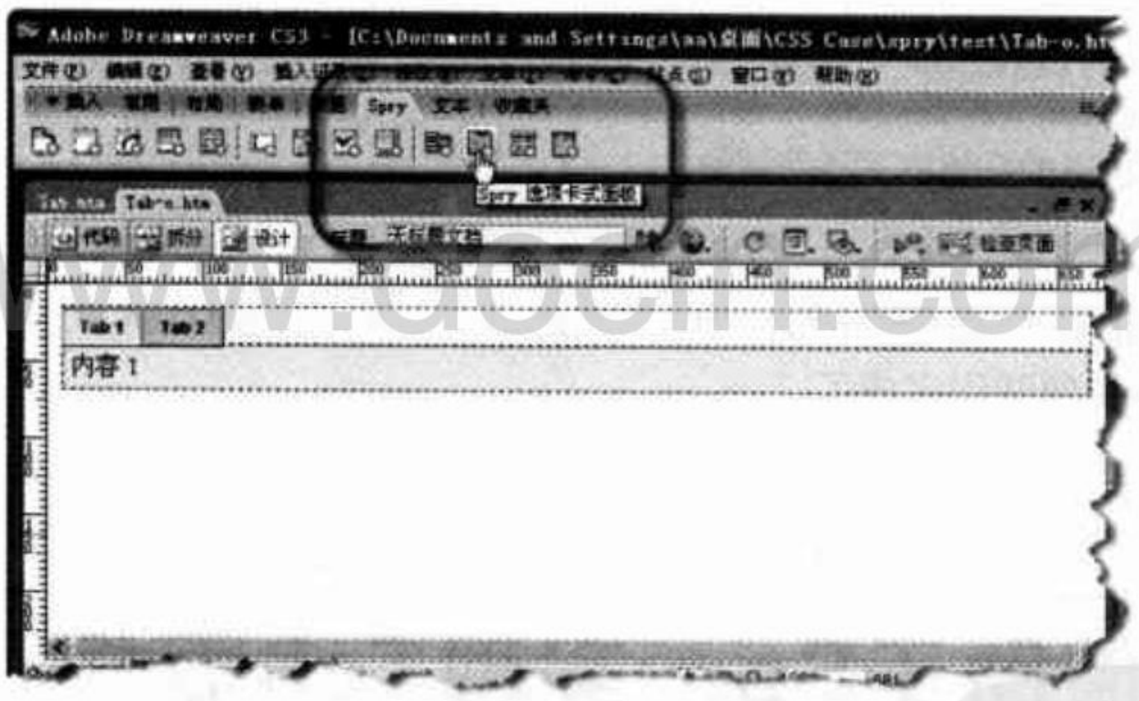


图12.11 Dreamweaver CS3中提供的Spry

② 保存这个页面，在浏览器中预览，效果如图12.12所示。这个Tab面板就已经做好了，非常简单。

图12.12 Spry生成的基本Tab面板效果

③ 现在查看一下页面的源代码，可以看到该页面使用了一个JavaScript文件和一个CSS文件。在向服务器上传这个页面的时候，同时要把这两个文件也要上传到相应的目录中，否则页面就无法正常工作了。

```
    <script                              src="SpryAssets/SpryTabbedPanels.js"
type="text/javascript"></script>
    <link  href="SpryAssets/SpryTabbedPanels.css"  rel=" stylesheet"
type="text/css" />
```

④ 再查看一下它的HTML结构，如下：

```
<body>
<div id="TabbedPanels1" class="TabbedPanels">
  <ul class="TabbedPanelsTabGroup">
    <li class="TabbedPanelsTab" tabindex="0">Tab 1</li>
    <li class="TabbedPanelsTab" tabindex="0">Tab 2</li>
    </ul>
  <div class="TabbedPanelsContentGroup">
    <div class="TabbedPanelsContent">内容 1</div>
    <div class="TabbedPanelsContent">内容 2</div>
  </div>
</div>
<script type="text/javascript">
<!--
var TabbedPanels1 = new Spry.Widget.TabbedPanels("TabbedPanels1");
//-->
</script>
</body>
```

可以看到，通过一些div和列表定义了这个面板的结构，最外层div的id为“TabbedPanels1”。里面分为两部分，上面是一个ul列表，id为“TabbedPanelsTabGroup”，它容纳了各个Tab，每个Tab就是一个列表项；下面是一个div，里面有与Tab个数相同的div，每一个里面放置相应的Tab页内容。

⑤ 如果需要增加Tab的个数，可以选中整个Tab面板，然后在窗口下侧的如图12.13所示的“属性”面板中单击“加号”按钮。如果希望调整几个Tab之间的顺序，可以通过旁边的三角形箭头按钮实现。最右边的“默认面板”下拉列表用于选择几个Tab页面中，哪一个作为默认打开的页面。

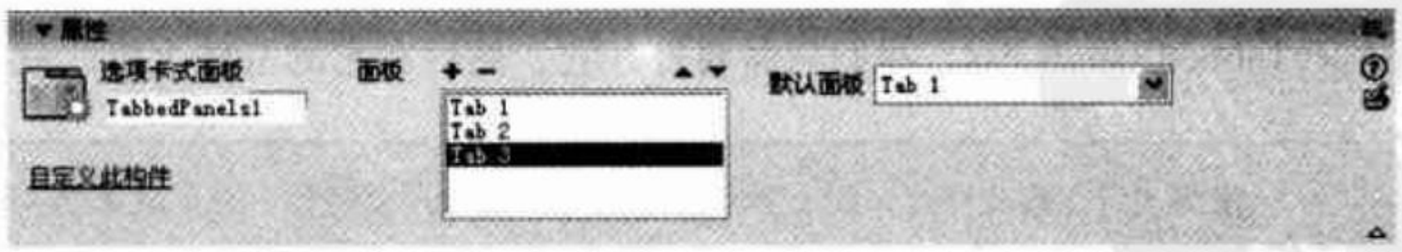


图12.13 在Dreamweaver CS3中可以增加或删除Tab

下面的任务就是以Spry生成的Tab面板为基础，通过对CSS样式的设置，实现更为美观的Tab面板。读者可以根据讲解的方法，设计出自己希望的Tab面板效果。

12.3.2 准备背景图片

各个Tab的宽度可以自动适应Tab文字的宽度，并且当鼠标指针经过某个Tab时，会发生颜色变化，如图12.14所示。

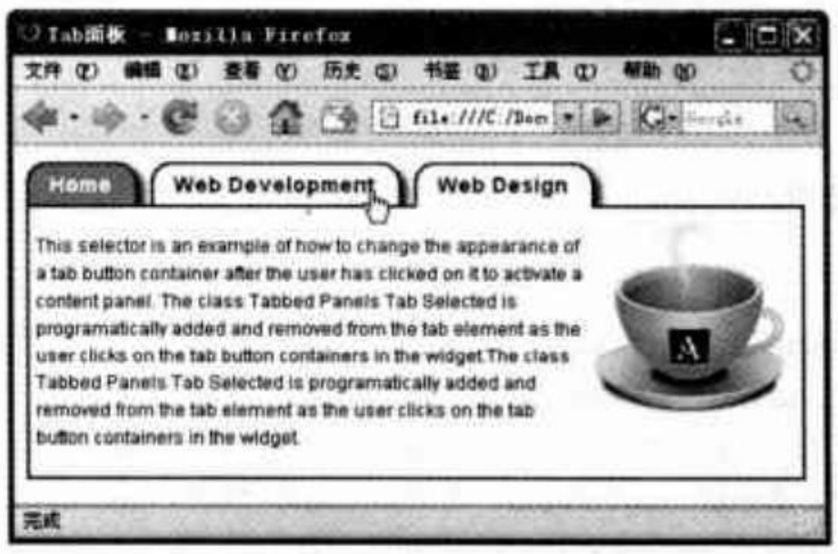


图12.14 Tab的3种状态

显然，这个Tab的效果需要背景图片来实现，因此要在Fireworks或者Photoshop等图像处理软件中制作3个图像，分别用于当前Tab、其他的Tab和鼠标指针经过的Tab，如图12.15所示。

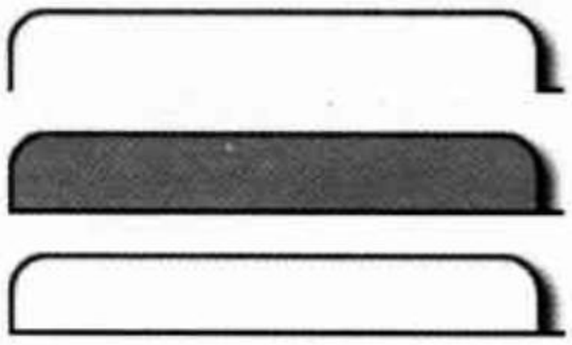
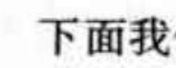


图12.15 当前页面菜单项的超链接

图12.15中的边线为2像素宽，整个图像200像素宽，30像素高，为了美观，可以做出一些装饰效果，例如图中的投影效果。为了Tab的背景能够适应文字的宽度，显然这里需要使用滑动门技术。

12.3.3 设置整体的CSS样式

下面就开始为它设置CSS。前面已经提到，Dreamweaver已经自动生成了一个CSS文件SpryAssets/SpryTabbedPanels.css，我们只需要打开它，然后在它的基础上修改就可以了。



注意 只能修改样式，而不能修改已经存在的选择器名称，否则就无法实现相应的功能了。

下面我们一段一段来改造它。首先找到第1段，代码如下：

```
.TabbedPanels {
 margin: 0px;
 padding: 0px;
```

```
    float: left;
    clear: none;
    width: 100%;
}
```

其作用是对整体进行设置，可以不用改，也可以把宽度改为固定宽度，例如500px。然后找到第2段，代码如下：

```
.TabbedPanelsTabGroup {
margin: 0px;
padding: 0px;
}
```

这段的作用是设置Tab组的样式，也就是ul列表的总体样式，在本案例中不用修改。

12.3.4 设置单个Tab的CSS样式

接下来就到关键的位置了，找到如下样式代码。

```
.TabbedPanelsTab {
 position: relative;
 top: 1px;
 float: left;
 padding: 4px 10px;
 margin: 0px 1px 0px 0px;
 font: bold 0.7em sans-serif;
 background-color: #DDD;
 list-style: none;
 border-left: solid 1px #CCC;
 border-bottom: solid 1px #999;
 border-top: solid 1px #999;
 border-right: solid 1px #999;
 -moz-user-select: none;
 -khtml-user-select: none;
 cursor: pointer;
}
```

这段的作用是设定各个Tab的样式，下面简要描述基本思路。

首先将postion属性设置为相对定位（relative），这样的目的是配合后面的top:1px，使Tab能够向下移动1个像素，使Tab最下面的1像素高度能够和下面内容区域的上边框有1个像素的重合。本案例中，由于边框的宽度为2像素，因此这里将“top:1px”改为“top:2px”。

然后将padding和margin都暂时设置为0，将背景设置为前面准备好的图像文件，然后删除4行关于border的样式，重新设置文字样式。

此外，为了使用滑动门技术来设定两层背景图像，这里将display属性设置为blcok，并确定高度为30像素。

修改完成后，这段代码如下所示：

```
.TabbedPanelsTab {
```

```
    display:block;
    height:30px;
    float: left;
    position: relative;
    top: 2px;
    padding: 0px 0px 0px 0px;
    margin: 0px 0px 0px 0px;
    background:url('tab-back.gif');
    list-style: none;
    cursor: pointer;
    color:#fff;
    font: bold 14px  Arial;
    -moz-user-select: none;
    -khtml-user-select: none;
}
```

这时效果如图12.16所示，可以看到基本的雏形已经产生了。

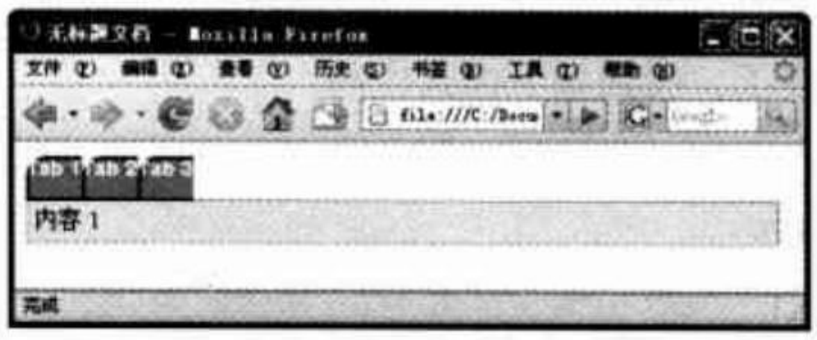


图12.16 为Tab设置了背景图像

12.3.5 设置单个Tab的滑动门背景

为了使用滑动门，必须有两个CSS“钩子”来放置背景图像，因此这里改造一下HTML文件。原来Dreamweaver生成的文件如下。

```
<body>
<div id="TabbedPanels1" class="TabbedPanels">
  <ul class="TabbedPanelsTabGroup">
    <li class="TabbedPanelsTab" tabindex="0">Tab 1</li>
    <li class="TabbedPanelsTab" tabindex="0">Tab 2</li>
  </ul>
……以下省略……
```

现在，在每个
和
之间增加一层和，这个span元素就可以作为CSS“钩子”使用。修改一下Tab文字内容，使它们的长度有所不同，以便看出背景图像设置是否正确。代码如下：

```
<body>
<div id="TabbedPanels1" class="TabbedPanels">
  <ul class="TabbedPanelsTabGroup">
  <li class="TabbedPanelsTab" tabindex="0"><span>Home</span></li>
   <li class="TabbedPanelsTab" tabindex="0"><span>Web Development
</span></li>
  <li class="TabbedPanelsTab" tabindex="0"><span>Web Design</span></li>
```

```
    </ul>
……以下省略……
```

这时效果如图12.17所示。

图12.17 增加span标记以便设置滑动门

接下来设置滑动门的背景图像。左边的图像是在前面的“.TabbedPanelsTab”中设置的。为了使文字的左面留有一定空白，应将“.TabbedPanelsTab”的padding由“0px 0px 0px 0px”改为“0px 0px 0px 15px”，这样左边就空出了15像素空白，如图12.18所示。

图12.18 设置Tab文字左边padding

然后针对li中的span新增加一段CSS样式，代码如下：

```
.TabbedPanelsTab  span {
  display:block;
  background:url('tab-back.gif') right top;
  padding:7px 25px 7px 0;
}
```


这段代码的作用是将span设置为块级元素，设定背景图像，并使背景图像右对齐。然后设定对Tab文字的padding，其中右侧padding是基于左侧padding的15像素再加上背景图像中阴影部分的10像素。上下padding各7个像素，可以使文字竖直方向居中。这时效果如图12.19所示。

图12.19 滑动门设置完成

12.3.6 设置鼠标经过效果

现在来设置鼠标经过效果。在原来的代码中找到下面的一段：

```
.TabbedPanelsTabHover {
  background-color: #CCC;
}
```

更改为：

```
.TabbedPanelsTabHover {
  background:url('tab-hover.gif');
  color:#900;
}

.TabbedPanelsTabHover span {
  display:block;
  background:url('tab-hover.gif') right top;
  padding:7px 25px 7px 0;
}
```

这里的原理和前面相同，设置滑动门背景，并把文字的颜色由白色变为深色。这时效果如图12.20所示。

图12.20 设置Tab的鼠标经过效果

12.3.7 设置当前页效果

设置好鼠标经过效果后，再设置表示当前Tab页的背景图像。首先找到下面一段CSS设置：

```
.TabbedPanelsTabSelected {
  background-color: #EEE;
  border-bottom: 1px solid #EEE;
}
```

替换为：

```
.TabbedPanelsTabSelected {
  background:url('tab-current.gif');
  color:#900;
}

.TabbedPanelsTabSelected span {
display:block;
  background:url('tab-current.gif') right top;
  padding:7px 25px 7px 0;
}
```

这时效果如图12.21所示，当前Tab的背景图形已不同于其他两个Tab。

图12.21 设置当前页的Tab样式

12.3.8 设置Tab页内容的CSS样式

现在对内容部分进行设置即可。找到如下CSS代码：

```
.TabbedPanelsContentGroup {
  clear: both;
  border-left: solid 1px #CCC;
  border-bottom: solid 1px #CCC;
  border-top: solid 1px #999;
  border-right: solid 1px #999:
  background-color: #EEE;
  }
```

以上代码的作用是设置内容div容器的样式，这里只需要把背景色以及边框的颜色和粗细按需要修改即可，代码如下：

```
.TabbedPanelsContentGroup {
  clear: both;
  border: solid 2px #900;
  background-color: #fff;
}
```


这时效果如图12.22所示。

图12.22 设置Tab页内容区域的背景色和边框

可以看到此时的Tab面板效果已经实现了。接下来就是要在每个Tab页中添加内容了，各页面内容依次添加到如下HTML的相应位置上。

```
<div class="TabbedPanelsContentGroup">
  <div class="TabbedPanelsContent">内容 1</div>
  <div class="TabbedPanelsContent">内容 2</div>
```

```
      <div class="TabbedPanelsContent">内容 3</div>
    </div>
```

如果要设置Tab页里面的样式，例如文字的样式、图像的样式等，只要在CSS文件中找到下面一段，在里面设置即可。设置好的效果如图12.23所示。

```
.TabbedPanelsContent {
  padding: 4px;
}
```

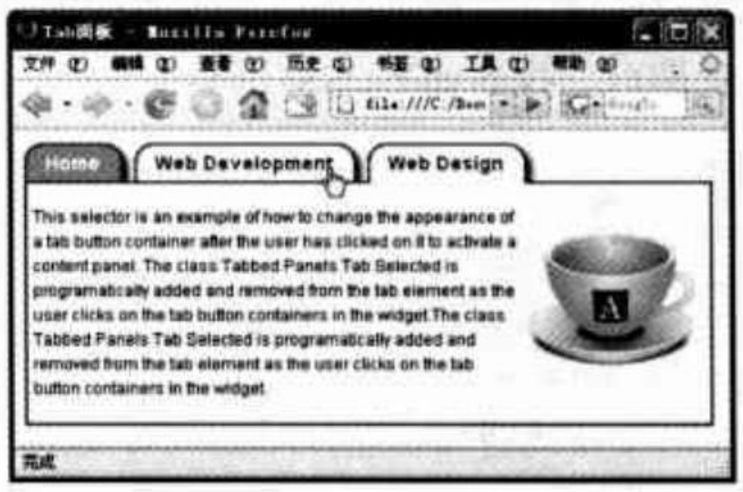


图12.23 设置完成后的效果

12.3.9 鼠标经过时换页

在默认情况下，单击某个Tab时才会切换到该页。也有很多网站上的Tab面板是在鼠标指针经过某个Tab时，就切换到该页。

使用Spry可以实现这个效果。方法是在代码中找到如下几行：

```
<body>
<div id="TabbedPanels1" class="TabbedPanels">
  <ul class="TabbedPanelsTabGroup">
    <li class="TabbedPanelsTab" tabindex="0" ><span>Home</span></li>
      <li class="TabbedPanelsTab" tabindex="0" ><span>Web Development
</span></li>
      <li class="TabbedPanelsTab" tabindex="0" ><span>Web Design
</span></li>
  </ul>
……以下部分省略……
```

然后在每个li的属性中增加一个JavaScript语句。

```
<body>
<div id="TabbedPanels1" class="TabbedPanels">
  <ul class="TabbedPanelsTabGroup">
    <li class="TabbedPanelsTab" tabindex="0"
      onmouseover="TabbedPanels1.showPanel(0)">
      <span>Home</span></li>
    <li class="TabbedPanelsTab" tabindex="0"
      onmouseover="TabbedPanels1.showPanel(1)">
      <span>Web Development</span></li>
```

```
        <li class="TabbedPanelsTab" tabindex="0"
            onmouseover="TabbedPanels1.showPanel(2)">
            <span>Web Design</span></li>
    </ul>
……以下部分省略……
```

这3个li实际就是各个Tab，增加它的onmouserover事件处理语句，当鼠标指针经过时就会显示该Tab页了。到这里Tab面板的制作就全部完成了。

本书光盘“第12章\03”文件夹中有两个网页文件，分别实现了鼠标单击Tab换页和鼠标指针经过Tab时换页。如果读者制作中遇到困难，请参考光盘中的正确文件。

12.4 折叠面板

在基于Spry的基础上实现了Tab面板之后，再来制作一个折叠面板的页面，效果如图12.24所示。图中一共有3个折叠面板，每个折叠面板都有一个标题。单击一个标题，就可以实现面板的折叠隐藏或展开。图12.24的左图为折叠起来的效果，右图为展开后的效果。

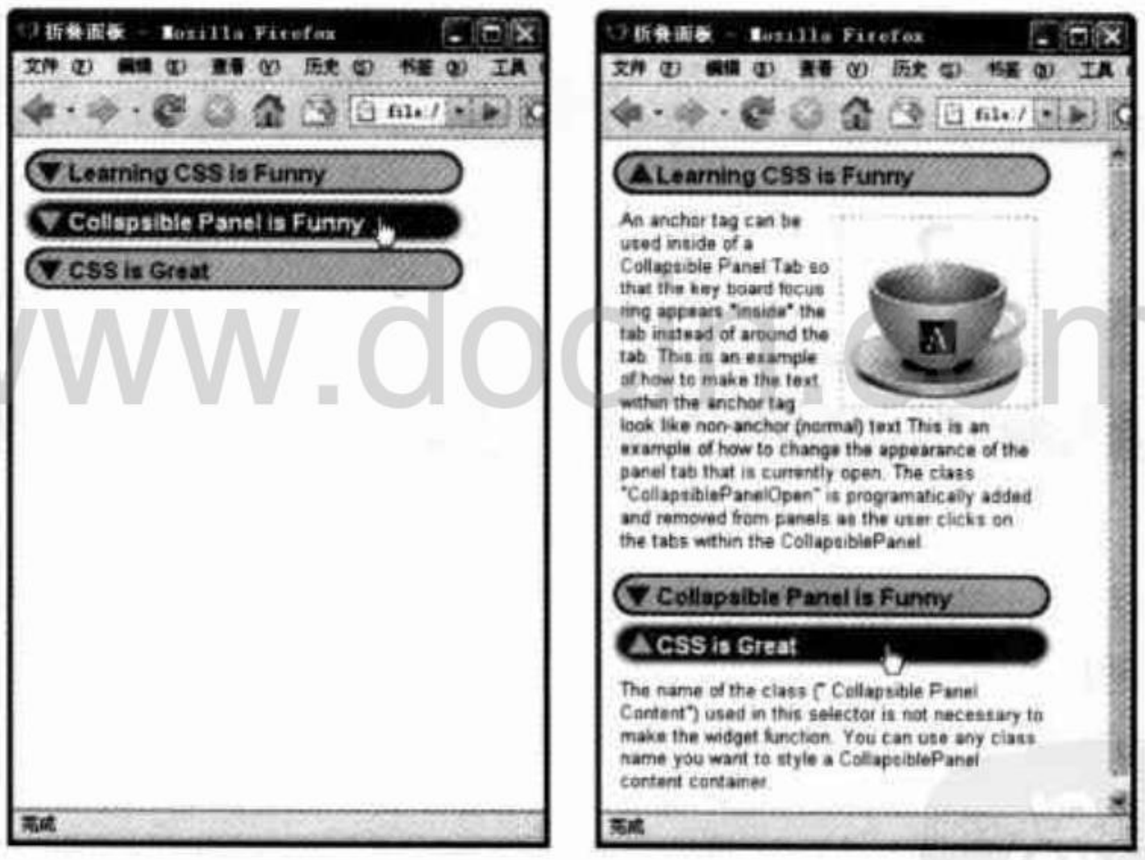


图12.24 设置当前页的Tab样式

本实例的文件位于本书光盘“第12章\03\collapsible.htm”。

12.4.1 建立基本的折叠面板

首先新建一个HTML文档，保存后，插入Spry工具栏中的最右边的“Spry可折叠面板”，如图12.25所示。

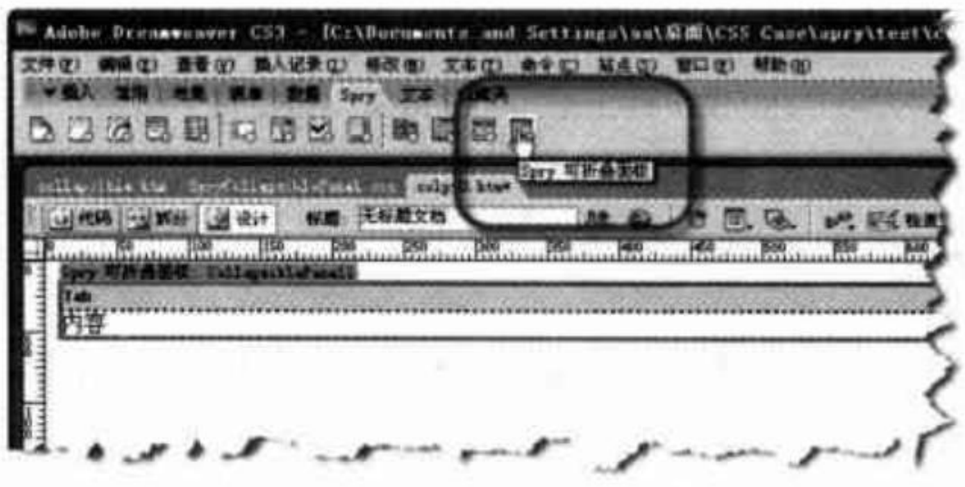


图12.25 在Dreamweaver CS3中找到折叠面板

连续插入多个可折叠面板，这时在浏览器中的效果如图12.26所示。不需要做任何设置，折叠面板的功能已经可以实现了，每一个折叠面板都可以折叠起来或者展开。

图12.26 插入多个可折叠面板

下面的任务就是通过设置CSS来使效果更加美观了。

12.4.2 准备背景图片

首先准备4个背景图像，如图12.27所示，它们都是300像素宽，30像素高。

图12.27 设置折叠面板所需的4个背景图像

● 其中上面的两个浅色图像作为平常状态的标题背景，下面的两个深色图像作为鼠标指针经过时的标题背景。

● 左端向上的两个三角形用于展开的面板标题背景，左端向下的两个三角形用于关闭的面板标题背景。

注意 请读者先尝试一下本书光盘中的实例文件，这样对于理解这里的讲解会有所帮助。

12.4.3 整体设置

先来研究一下HTML文件。可以看到和前面的Tab面板相同，也需要一个JavaScript文件和一个CSS文件。

```
<script src="SpryAssets/SpryCollapsiblePanel.js"
                                type="text/javascript"></script>
<link href="SpryAssets/SpryCollapsiblePanel.css"
                                rel="stylesheet" type="text/css">
```

定义折叠面板结构的方法很简单，代码如下：

```
<div id="CollapsiblePanel1" class="CollapsiblePanel">
  <div class="CollapsiblePanelTab" tabindex="0">Tab</div>
  <div class="CollapsiblePanelContent">内容</div>
</div>
```

外面是一个div，用于定义整个折叠面板。它的内部又有两个div，分别用于放置标题和内容。

下面先打开在文件的head部分引用的CSS文件，其中有几段CSS设置，分别针对不同的元素。第1段代码如下：

```
.CollapsiblePanel {
  margin: 0px;
  padding: 0px;
  border-left: solid 1px #CCC;
  border-right: solid 1px #999;
  border-top: solid 1px #999;
  border-bottom: solid 1px #CCC;
}
```

这里设置了一个折叠面板的整体效果和边框。本案例中不设置边框，但是需要设置一个300像素宽的总宽度，因此代码修改为：

```
.CollapsiblePanel {
  width:300px;
  margin: 0px;
  padding: 0px;
}
```

12.4.4 设置折叠面板的标题

第2段代码用于设置折叠面板的标题，修改前代码如下：

```
.CollapsiblePanelTab {
  font: bold 0.7em sans-serif;
  background-color: #DDD;
  border-bottom: solid 1px #CCC;
  margin: 0px;
  padding: 2px;
  cursor: pointer;
  -moz-user-select: none;
  -khtml-user-select: none;
}
```

这里要把背景设置为前面制作好的背景图像。由于使用了浅色背景图像，因此需要将文字设置为深色，并设置标题的高度为30像素。为了使标题文字竖直居中，可以将line-height属性也设置为30像素，通过padding把文字移动到适当的位置。修改后的代码如下：

```
.CollapsiblePanelTab {
  font: bold 1em Arial;
  color:#630;
  background: url(col-close-back.gif) no-repeat;
  line-height:30px;
  height:30px;
  margin: 0px;
  padding: 2px 30px;
  cursor: pointer;
  -moz-user-select: none;
  -khtml-user-select: none;
}
```

这时在浏览器中的效果如图12.28所示。

图12.28 设置了标题背景图像后的效果

12.4.5 设置折叠面板的初始状态

图12.28中的背景是用于折叠状态的，这里把各个面板的初始状态设置为折叠，而不是打开。这个操作可以在“属性”面板完成，也可以通过修改HTML代码来实现。

在HTML中，找到如下代码：

```
<script type="text/javascript">
<!--
var CollapsiblePanel1 =
      new Spry.Widget.CollapsiblePanel("CollapsiblePanel1");
var CollapsiblePanel2 =
      new Spry.Widget.CollapsiblePanel("CollapsiblePanel2");
var CollapsiblePanel3 =
      new Spry.Widget.CollapsiblePanel("CollapsiblePanel3");
//-->
</script>
```

修改为：

```
<script type="text/javascript">
```

```
    <!--
    var CollapsiblePanel1 =
     new Spry.Widget.CollapsiblePanel("CollapsiblePanel1",{contentIsOpen:
false});
    var CollapsiblePanel2 =
     new Spry.Widget.CollapsiblePanel("CollapsiblePanel2",{contentIsOpen:
false});
    var CollapsiblePanel3 =
     new Spry.Widget.CollapsiblePanel("CollapsiblePanel3",{contentIsOpen:
false});
    //-->
    </script>
```

也就是在每行增加“,{contentIsOpen:false}”参数，注意大小写不要写错。这时在浏览器中的显示效果如图12.29所示，可以看到面板内容在初始状态都隐藏起来了。

图12.29　所有折叠面板在初始状态都设置为关闭状态

12.4.6 设置展开状态的标题背景

接下来设置展开状态时的标题背景，找到如下代码：

```
.CollapsiblePanelOpen .CollapsiblePanelTab {
 background-color: #EEE;
}
```

改为：

```
.CollapsiblePanelOpen .CollapsiblePanelTab {
 background:url(col-open-back.gif) no-repeat;
}
```

在浏览器中单击某一个标题，该面板就会展开，如图12.30所示，展开的面板标题下面有一个蓝色横条，这是由于还没有清理自动生成的代码导致的，暂时不用管它。

图12.30　设置展开的标题背景

12.4.7 设置鼠标指针经过时的标题背景

接下来设置鼠标指针经过时的标题背景。找到如下一段代码：

```
    .CollapsiblePanelTabHover,    .CollapsiblePanelOpen  .CollapsiblePanel
TabHover {
      background-color: #CCC;
    }
```

这段代码中同时设置了展开和折叠状态的面板标题在鼠标指针经过时的背景颜色，由于本案例需要设置在鼠标指针经过时，展开和折叠状态的面板标题各自的背景图像，因此需要拆分成两段CSS设置，代码如下。

```
.CollapsiblePanelTabHover {
  background:url(col-close-hover.gif) no-repeat;
  color:#FFF;
}
```

```
.CollapsiblePanelOpen .CollapsiblePanelTabHover {
  background:url(col-open-hover.gif) no-repeat;
  color:#FFF;
}
```

其中，前面的代码设置的是处于折叠状态的面板，在鼠标指针经过标题时的背景图像，后者反之。

接下来把最后一段代码删掉，这段代码如下。

```
.CollapsiblePanelFocused .CollapsiblePanelTab {
  background-color: #3399FF;
  }
```


它的作用是设置处于焦点的面板的背景图像，本案例不做设置，因此删除即可。这时折叠面板就设置完成了，可以看到，鼠标指针经过时，标题的背景颜色会变化，非常醒目。

接下来设置面板内部的内容样式。如果将上面设置的面板标题和它的内容配合起来，就会取得很好的视觉效果了，如图12.31所示，这里就不再赘述。

至此，这个折叠面板的效果就全部完成了。

12.5 伸缩面板

本章的最后一个案例要实现一个“伸缩面板”。伸缩面板具有固定的高度，包含若干页面，其中只有一个页面处于打开状态。当用鼠标单击某一个页面的标题时，该页面就会展开，原来展开的页面同时自动隐藏起来。在展开和隐藏的过程中，会有动画效果演示。

本实例的文件位于本书光盘“第12章\04\accordion.htm”。光盘的该目录下还有一个accordion-hover.htm文件。二者的区别是，在accordion.htm中，鼠标单击标题才能实现换页；而在accordion -hover.htm中，鼠标指针经过某个标题时就会切换到该页，而无需单击鼠标。

本案例完成后的效果如图12.32所示。

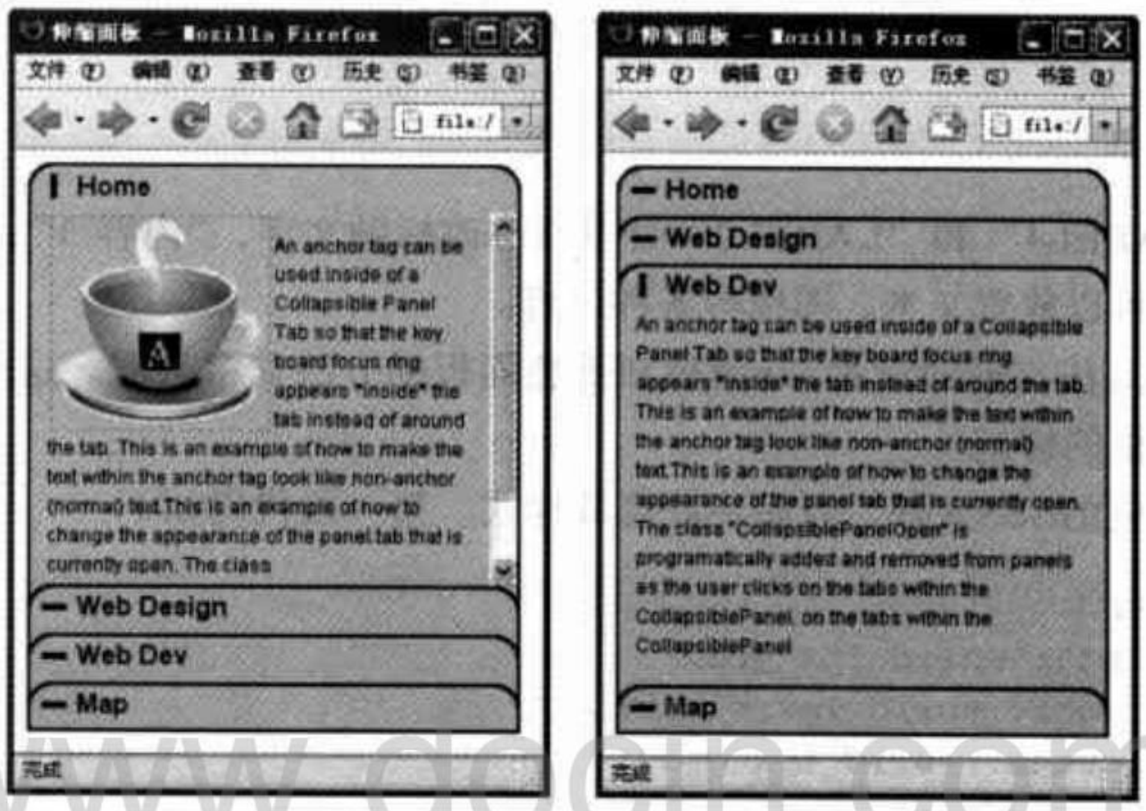


图12.32　伸缩面板的效果

12.5.1 建立基本的伸缩面板

Spry工具栏的右边第2个按钮是制作伸缩面板的工具，如图12.33所示。

图12.33　制作伸缩面板的工具——Spry折叠式

在页面中插入一个伸缩面板之后，可以通过“属性”面板加入新的页面，并调整页面的顺序，如图12.34所示。

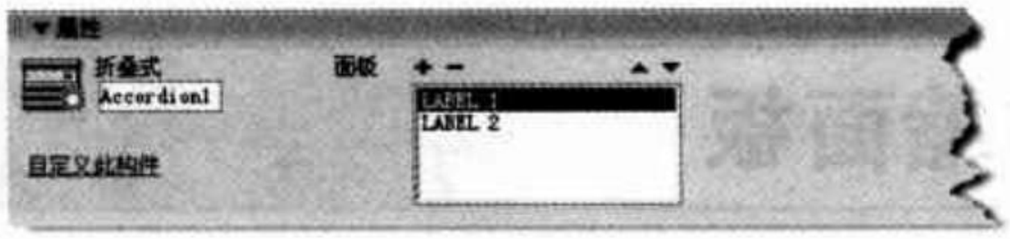


图12.34 使用“属性”面板可以增加或删除页面

相应的HTML结构如下所示。

```
<body>
  <div id="Accordion1" class="Accordion" tabindex="0">
    <div class="AccordionPanel">
      <div class="AccordionPanelTab">LABEL 1</div>
      <div class="AccordionPanelContent">内容 1</div>
    </div>
    <div class="AccordionPanel">
      <div class="AccordionPanelTab">LABEL 2</div>
      <div class="AccordionPanelContent">内容 2</div>
    </div>
  </div>
```

其中的“LABEL1”和“LABEL2”处可以修改标题文字，“内容 1”和“内容 2”处为各页面的内容，可以放置文本、图像等各种内容。

修改样式的方法与前面设置Tab面板和折叠面板的方法非常类似。这里为读者演示一下如何制作带有边框的样式。

打开页面相应的CSS文件，开始逐段修改样式。第一段代码如下：

```
.Accordion {
  border-left: solid 1px gray;
  border-right: solid 1px black;
  border-bottom: solid 1px gray;
  overflow: hidden;
}
```

它的作用是对整个伸缩面板进行设置。本案例中，设置面板的宽度，仅保留底端的边框（注意本案例使用2像素宽的边框），左右边框删除，因此这段代码修改为：

```
.Accordion {
  width:300px;
  overflow: hidden;
  border-bottom:2px #006 solid;
}
```

12.5.2 设置标题的样式

接下来准备两个图像，每个都是300像素宽，30像素高，其中一个左端有一短横线，表示处于收缩状态，另外一个左端有一短竖线，表示处于展开状态，如图12.35所示。

图12.35 准备用于展开和收缩状态的背景图像

接着找到如下代码：

```
.AccordionPanelTab {
  background-color: #CCCCCC;
  border-top: solid 1px black;
}
```

这段代码的作用就是设置收缩状态的面板的标题部分，因此这里设置文字的样式、高度、padding等属性，代码如下：

```
.AccordionPanelTab {
  background: url(ac-open-back.gif) no-repeat;
  font: bold 16px Arial;
  color:#006;
  line-height:30px;
  height:30px;
  margin: 0px;
  padding: 0px 30px;
  cursor: pointer;
  -moz-user-select: none;
  -khtml-user-select: none;
}
```

然后找到下面这段代码：

```
.AccordionPanelOpen .AccordionPanelTab {
}
```


它的作用是设置展开的标题背景，设置如下：

```
.AccordionPanelOpen .AccordionPanelTab {
  background: url(ac-open-back-open.gif) no-repeat;
  }
```

12.5.3 对最上面的标题进行特殊处理

这时需要注意的是，最上面的一个标题背景应该与其他标题背景有所不同，请注意图12.36中画方框的位置，最上面的圆角处没有向上伸展的竖线。

图12.36 最上面的标题背景图像需要单独设置

因此现在需要为第一个页面的标题制作两个图像，分别在第一个页面处于收缩和展开时使用，如图12.37所示。

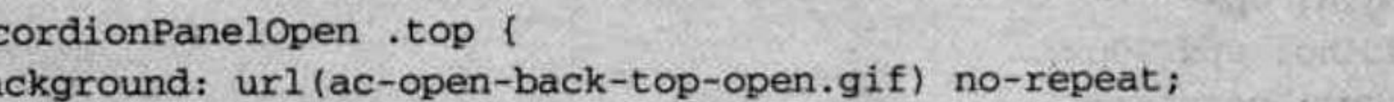


图12.37 最上面的标题所需的背景图像

接下来对第一个标题的HTML稍作修改，为它增加一个类别“top”，代码由：

```
<div class="AccordionPanelTab ">Home</div>
```

改为：

```
<div class="AccordionPanelTab top">Home</div>
```

然后分别设定第一个标题收缩和伸展时的背景图像。增加如下两段代码，这两段代码原CSS文件中并没有，需要手工添加。

```
.top{
  background: url(ac-open-back-top.gif) no-repeat;
}
.AccordionPanelOpen .top {
  background: url(ac-open-back-top-open.gif) no-repeat;
}
```

对每个页面里的内容进行设置。注意设置左右2像素的边框，使其和背景图像的左右竖线连接上。注意这里的高度将控制页面展开后的高度，如果内容超出这个高度，将自动出现滚动条。

```
.AccordionPanelContent {
  background:#ADF;
  font:12px/18px Arial;
  overflow: auto;
  margin: 0px;
  padding: 0 10px;
  height: 230px;
  border-left:2px #006 solid;
  border-right:2px #006 solid;
}
```

最后把没有用到的代码删除，即可实现最终的效果。在一些网站上可以看到这种伸缩面板，有的是通过鼠标单击来展开某一个页面，也有的是鼠标指针经过某个标题时就展开相应的页面。如果要实现后面这个效果，也是可以的，方法是为每个标题所在的div增加一句JavaScript语句，例如将原来的如下语句：

```
<div class="AccordionPanelTab">Web Design</div>
```

改为：

```
<div class="AccordionPanelTab"
  Onmouseover="Accordion1.openPanel(this.parentNode);">
        Web Design </div>
```

就可以实现用鼠标指针经过而不是鼠标单击实现切换页面。请注意，4个标题都要加上这句话。

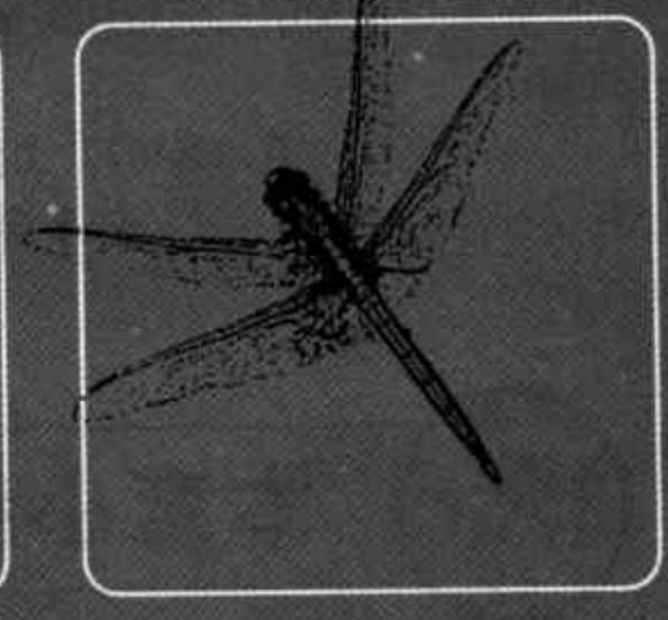
本书光盘“第12章\04”文件夹中有两个文件，分别实现了鼠标单击标题实现换页和鼠标指针经过标题时换页。如果读者制作中遇到困难，请参考光盘中的正确文件。

12.6 本章小结

本章一共介绍了4个案例，第1个Tab菜单是仅使用CSS设置就可以完成的效果，后面的3个案例都是用Dreamweaver CS3中内置的Spry框架实现的、功能更强大的Web页面元素。

使用Spry可以为不会JavaScript的设计师提供巨大的帮助，只要懂得CSS，就可以制作出效果复杂的网页效果和界面元素。由于篇幅所限，本书并没有深入讲解Spry的各个方面，实际上它的功能相当强大，值得读者深入研究。

CSS 设计彻底研究



CSS设计

彻底研究

第13章
固定宽度布局剖析与制作

CSS的排版是一种很新的排版理念，完全有别于传统的排版习惯。它将页面首先在整体上进行 标记的分块，然后对各个块进行CSS定位，最后再在各个块中添加相应的内容。利用CSS排版的页面，更新起来十分容易，甚至连页面的拓扑结构，都可以通过修改CSS属性来重新定位。

在本章中，我们将就固定宽度的网页布局进行深入的剖析，并给出一系列的实例，使读者能够自如地掌握这些布局方法。

13.1 CSS排版观念

在过去使用表格布局的时候，在设计的最开始阶段，就要确定页面的布局形式。由于使用表格来进行布局，一旦确定下来就无法再更改了，因此有极大的缺陷。使用CSS布局则完全不同，设计者首先考虑的不是如何分割网页，而是从网页内容的逻辑关系出发，区分出内容的层次和重要性。然后根据逻辑关系，把网页的内容使用div或其他适当的HTML标记组织好，再考虑网页的形式如何与内容相适应。

实际上，即使是很复杂的网页，也都是一个模块一个模块逐步搭建起来的。下面我们将举一些访问量非常大的网站为例，看看它们都是如何布局的。

13.1.1 MSN的首页

图13.1左图显示的是微软公司的msn.com的首页。msn.com是全世界访问量前3名的网站，内容繁多。从网页布局角度来说，其实并不复杂，可以简单地划分一下区域，如图13.1右图所示。这个网站是一个内容宽度固定，水平居中放置的页面，顶部是一组通栏的内容，它的下面分为左右两栏，各自独立，互不干扰。每一栏中都是依次排列各种图文内容。最下面是页脚，在图中没有显示，但是基本上所有网站的底部都会有一个页脚，放置版权信息等内容。

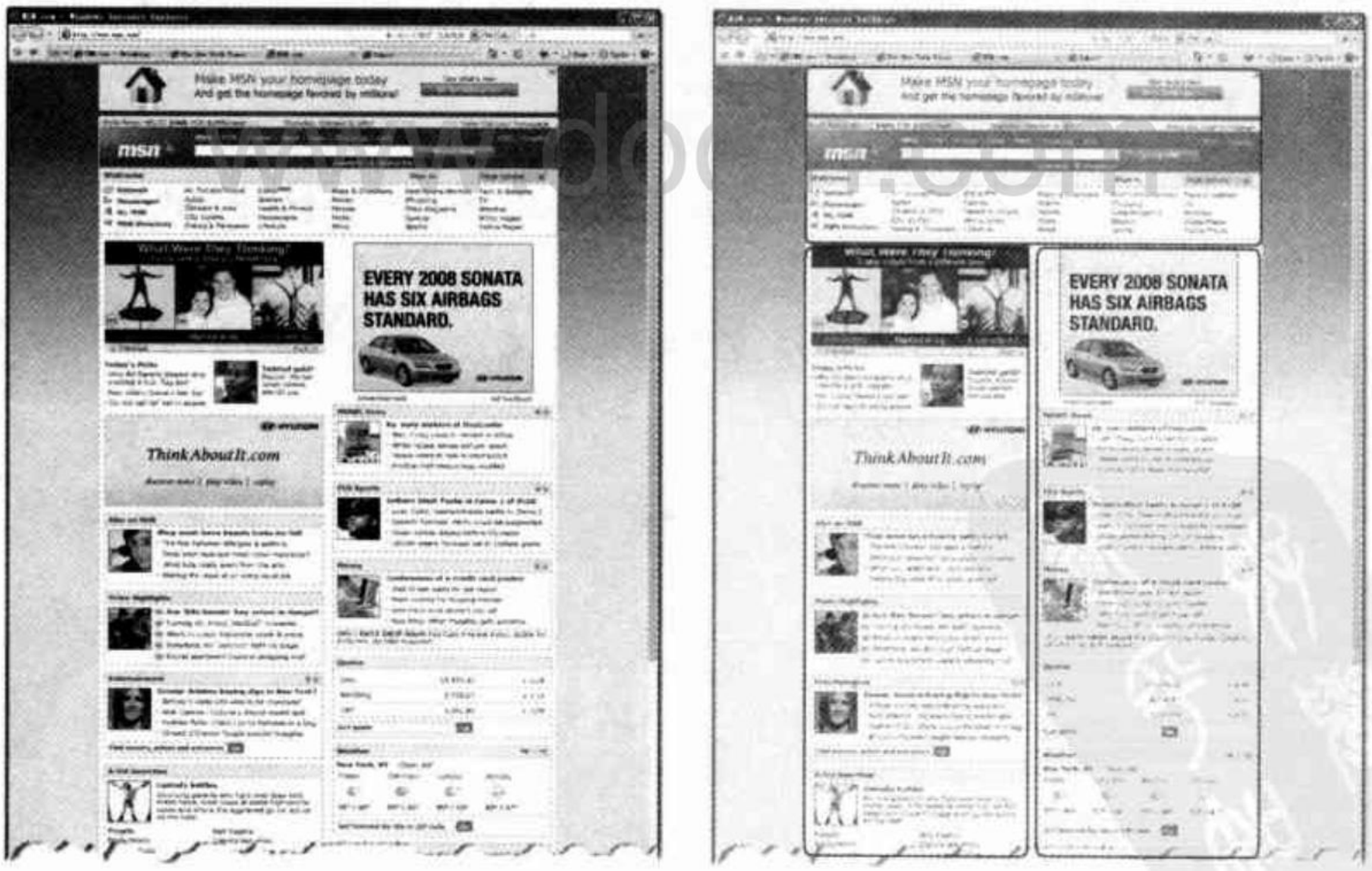


图13.1 msn.com的首页

这样的页面可以简单地抽象为如图13.2所示的页面样式。

对于这样的页面布局实际上我们已经制作过一个了，就是在前面11.4节学习制作圆角框中，实现的正是这种形式的页面。

为了便于称呼，本书中使用统一的命名方式，此类型的页面布局称为“1-2-1”布局，“减号”表示竖直方向排列，即最上面是1列，它的下面分为两列，再下面又是1列的这种布局形式。

下面我们再列举几个著名站点的页面布局形式，进行一些简单的分析，使读者先有一个感性的认识，然后再具体讲解各种布局形式的实现方法。

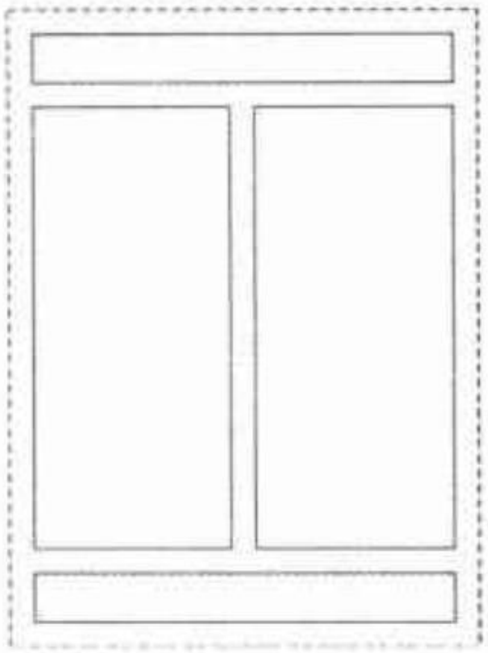
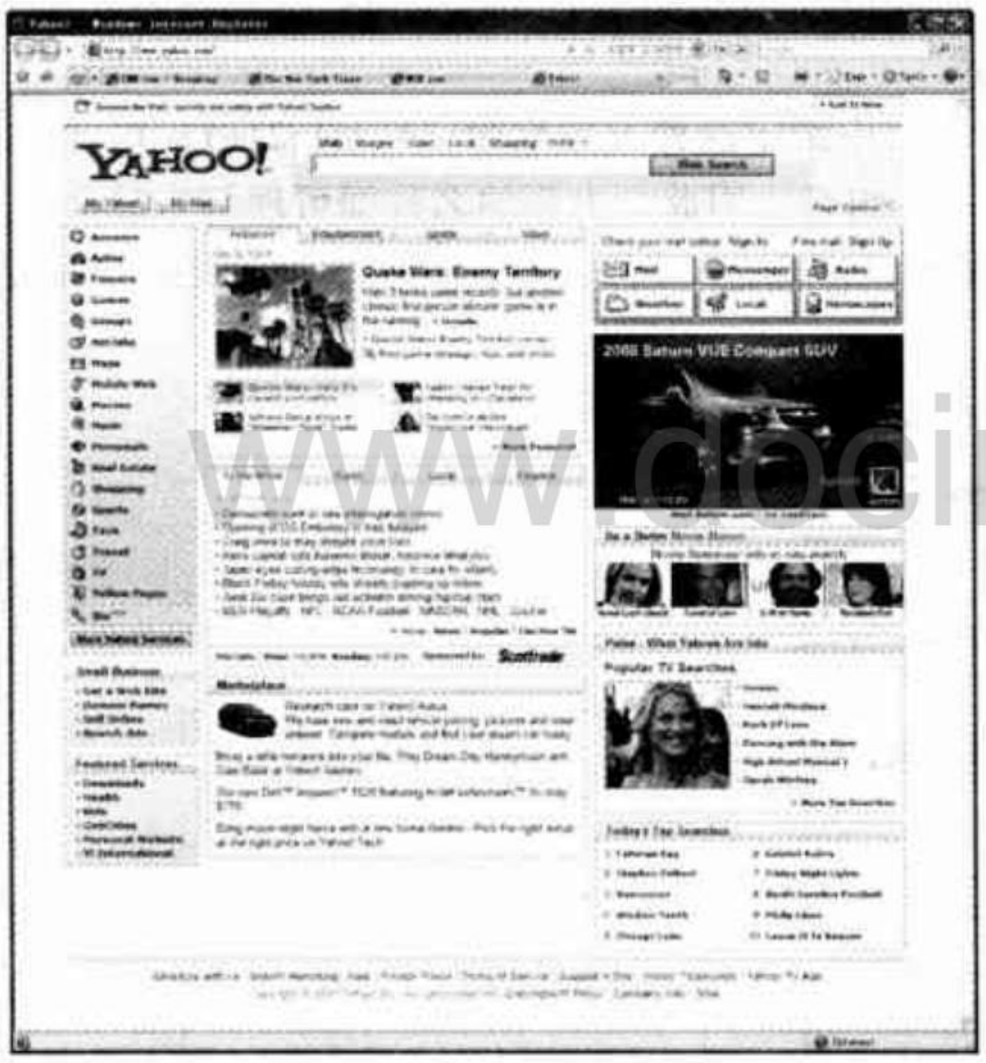


图13.2　抽象为“1-2-1”布局的示意图

13.1.2 Yahoo.com

Yahoo.com是目前访问量排名第一的网站（alexa.com的排名数据），然而它的页面非常简洁，如图13.3左图所示。它的抽象出来的页面布局形式如图13.3右图所示，是一个典型的“1-3-1”布局。



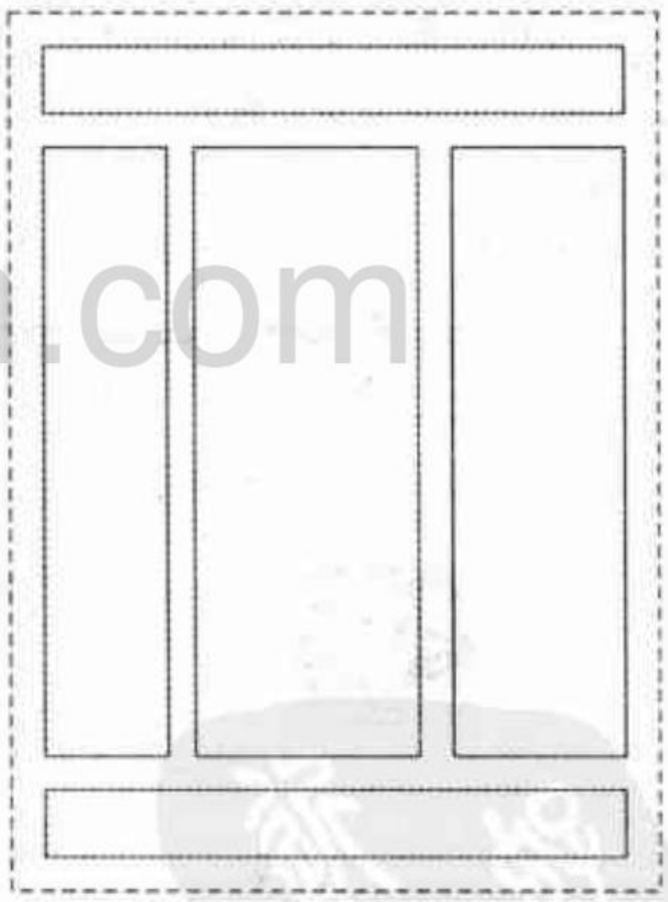


图13.3　“1-3-1”布局的yahoo.com网页及其示意图

13.1.3 Times.com

Times.com也是一个新闻类的知名站点。由于信息内容类目繁多，因此它采用了“1-4-1”的布局形式，如图13.4所示，并且各分栏宽度不同，适应于不同类别的内容。

图13.4 “1-4-1”布局的times.com网页

13.1.4 CNN.com

CNN.com使用的形式较前面几种布局方式稍有变化，如图13.5左图所示。可以看到在页面的顶部，仍是1列的形式；它的下面分为两列，左宽右窄；再接下来，左侧的部分又分为了两列。抽象后的结构如图13.5右图所示。

那么对这种方式我们如何称呼呢，本书中约定这种方式称为“1-((1-2)+1)-1”的布局形式。这里的加号表示横向分割，括号表示组合。因此，上面的名称即可理解为最上面为1列，它的下面分为左右两列，其中左边的列的上侧是一列，下侧是两列，页面的最下侧是一个单列。

读者可以看到，这种标记方式比文字描述要简单清楚得多。

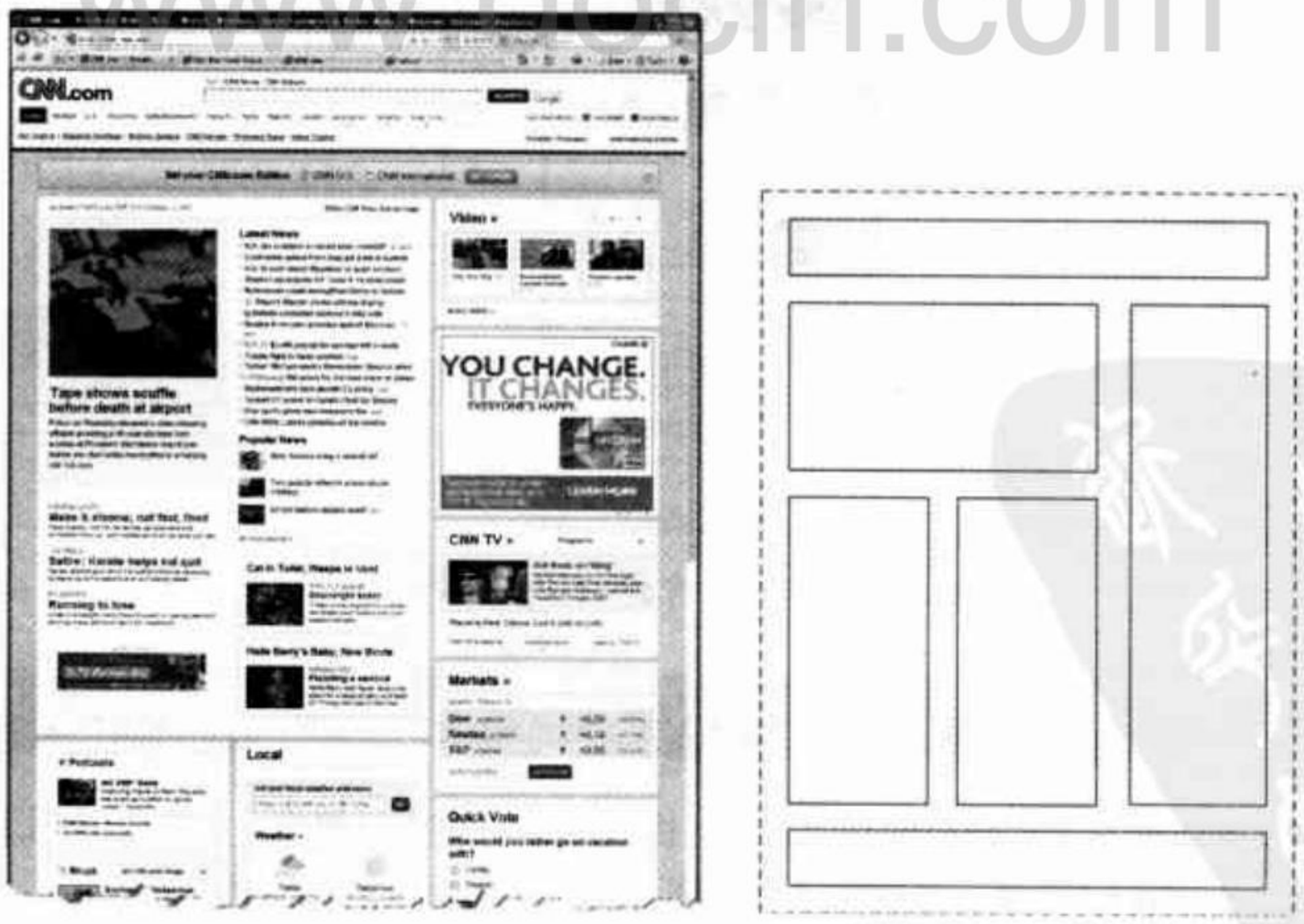


图13.5 “1-((1-2)+1)-1”布局的CNN.com网页及其示意图

这里请读者思考，这个页面为什么是“1-((1-2)+1)-1”结构，而不是“1-2-3-1”结构。答案是，如果是“1-2-3-1”结构，那么2列部分和3列部分之间，应该存在一条横向贯穿页面的分割线，而图中不存在这样的分割线，因此它不是“1-2-3-1”结构。

13.1.5 163.com

前面举了几个国外网站的例子，下面再来看看国内163.com的首页布局情况，如图13.6所示。163.com的页面大体上是一个两栏布局，但是中间穿插一些单列的内容，通常是用来发布广告的位置。

因此这种结构可以写作“1-(2-1)*”结构，这里的星号表示重复1次或多次。

图13.6　163.com网页的布局

为了使读者能够快速地掌握页面结构的分析方法，这里给出两个页面布局示意图，如图13.7所示。请读者自己思考一下它们的结构表达式。

其中图13.7左图的结构表达式应该是“2-1-(1+(2-1))”，右图的结构表达式应该是“1-((2-1-2)+1)-1”。

在了解了一些常见的布局结构以后，下面就可以开始正式学习如何制作各种布局的页面了。

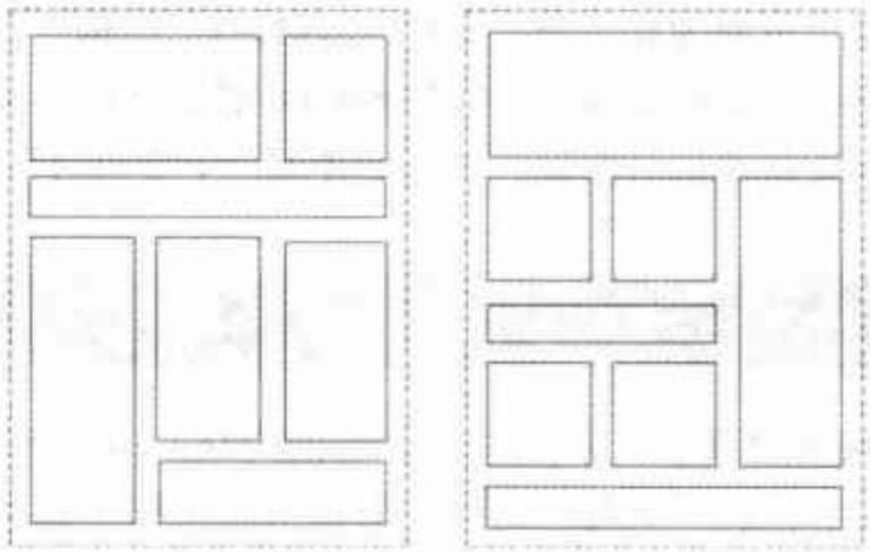


图13.7 布局结构示意图

注意 本章的学习目的是掌握如何以整页为对象进行布局，页面的各个组成部分应该事先已经准备好，否则大量的代码将用于局部的样式，这样学习起来就会非常困难。因此，在本章中，将以前面第11章圆角框一章制作的案例为基础，具体来说是使用的是圆角框中的不固定宽度带边框的案例，该案例中实现的圆角框可以方便地嵌入任何页面，作为页面的一个组成部分。

在学习本章之前，读者务必已经掌握该圆角框的制作原理和使用方法。本章中的部分案例是由多个圆角框组成的，导致页面代码很长，如果不熟悉圆角框部分的代码，分析代码时就会比较吃力。因此先掌握圆角框的做法，再学习本章将会事半功倍。

13.2 单列布局

这显然是最简单的一种布局形式。通过这个例子，希望读者能够顺便复习前面圆角框的制作方法。实现的效果如图13.8所示。本案例文件位于本书光盘“第13章\1-1-1.htm”。

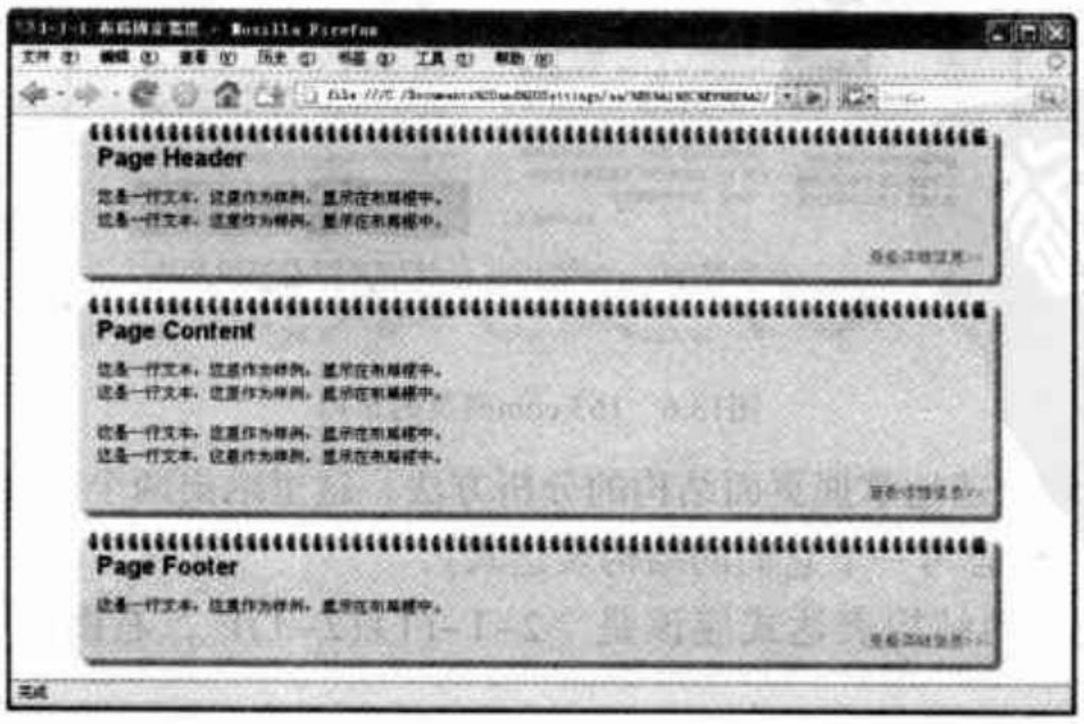


图13.8 单列固定宽度的页面布局

13.2.1 放置第一个圆角框

先在页面中放置第一个圆角框，HTML代码如下。

```
<body>
<div class="rounded">
        <h2>Page Header</h2>
        <div class="main">
                <p>
                        这是一行文本，这里作为样例，显示在布局框中。<br />
                        这是一行文本，这里作为样例，显示在布局框中。
                </p>
        </div>
        <div class="footer">
                <p>
                        查看详细信息
                </p>
        </div>
</div>
</body>
```

这组 …… 之间的内容是固定结构的，其作用就是实现一个可以变化宽度的圆角框。要修改内容，只需要修改相应的文字内容或者增加其他图片内容即可。

注意　不要修改这组代码的结构。当需要多个圆角框时，直接复制并修改其中相应内容即可。

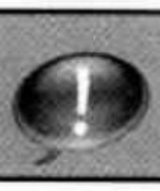
13.2.2 设置圆角框的CSS样式

为了实现圆角框效果，相应的CSS样式代码如下：

```
body {
  background: #FFF;
  font: 13px/1.5 Arial;
  margin:0;
  padding:0;
  }
.rounded {
  background: url(images/left-top.gif) top left no-repeat;
  }
.rounded h2 {
  background: url(images/right-top.gif) top right no-repeat;
  padding:20px 20px 10px;
  margin:0;
  }
.rounded .main {
  background: url(images/right.gif) top right repeat-y;
  padding:10px 20px;
```

```
  margin:-2em 0 0 0;
  }
.rounded .footer {
  background: url(images/left-bottom.gif) bottom left no-repeat;
  }
.rounded .footer p {
  color:#888;
  text-align:right;
  background:url(images/right-bottom.gif) bottom right no-repeat;
  display:block;
  padding:10px 20px 20px;
  margin:-2em 0 0 0;
  }
```

上面代码中的第一段是对整个页面的样式定义，例如文字大小等，其后的5段以.rounded开头的CSS样式都是为实现圆角框进行的设置。



注意 背景图片的路径不要弄错，否则将无法显示背景图片。

以上CSS代码在后面的制作中，都不需要调整，直接放置在<style></style>之间即可。此时网页的效果如图13.9所示，目前这个圆角框还没有设置宽度，因此它会自动伸展。

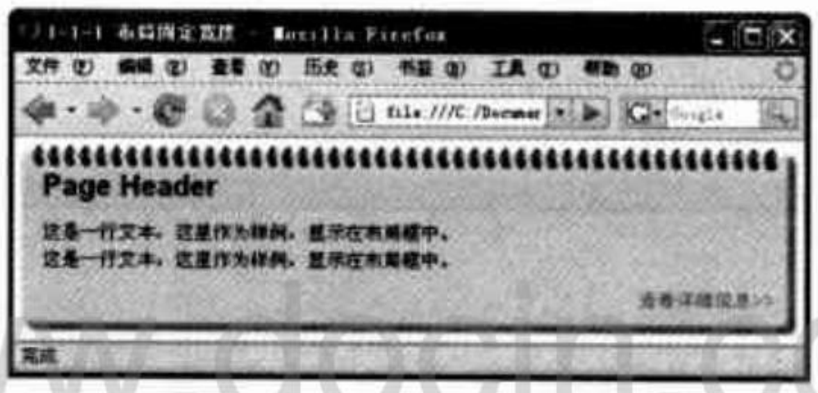


图13.9 放置第一个圆角框

现在来给它设置固定的宽度。注意这个宽度不要设置在“.round”相关的CSS样式中，因为该样式会被页面中的各个部分公用，如果设置了固定宽度，其他部分就不能正确显示了。

因此，应该为该圆角框单独设置一个id，把针对它的CSS样式放到这个id的样式定义部分。设置margin实现在页面中居中，并用width属性确定固定宽度，代码如下：

```
#header,#pagefooter,#content{
  width:760px;
  margin:0 auto;
  }
```

然后，在HTML部分的 …… 的外面套一个div，代码如下：

```
<div id="header">
  <div class="rounded">
  ……固定结构代码省略……
  </div>
<div>
```

这时，在Firefox中的效果如图13.10所示，正确地实现了期望的效果。

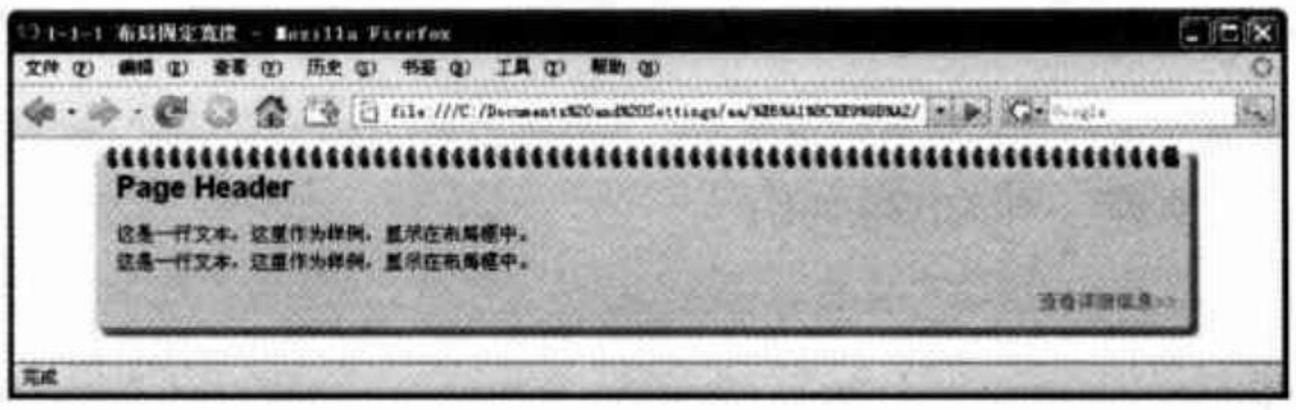


图13.10　在Firefox中显示正确的效果

但是在IE 6中的效果如图13.11所示。

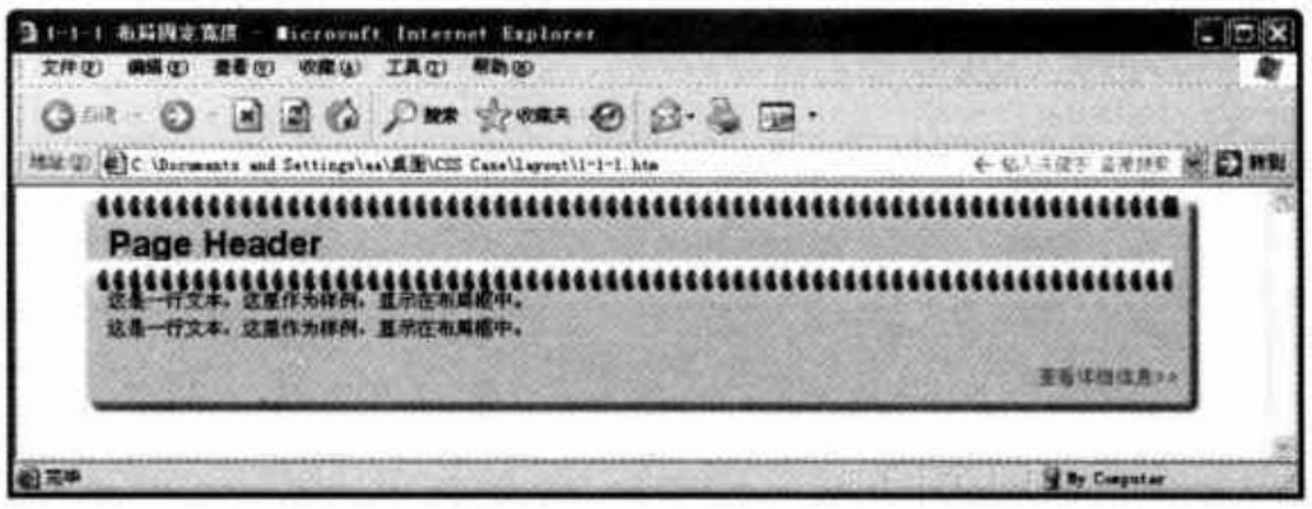


图13.11　在IE 6中显示错误的效果

可以看到背景图像发生错误，这是由于IE 6本身的错误造成的，在IE 7中已经修正了这个错误。为了使背景图像在IE 6中也能正确显示，需要对圆角框设置增加对宽度的设置。实际上如果不设置为100%，也应该按照100%显示，但是人为设置100%后，会强制IE 6重新计算相关数值，从而正确显示背景图片。

```
.rounded {
 background: url(images/left-top.gif)  top left no-repeat;
 width:100%;
 }
```

修改后，在IE 6中的效果如图13.12所示。这是本章中最后一次修改关于“.rounded”部分的样式代码，以后就不会在涉及它的代码了，我们将把精力集中在如何制作出各种各样的完整页面布局上。

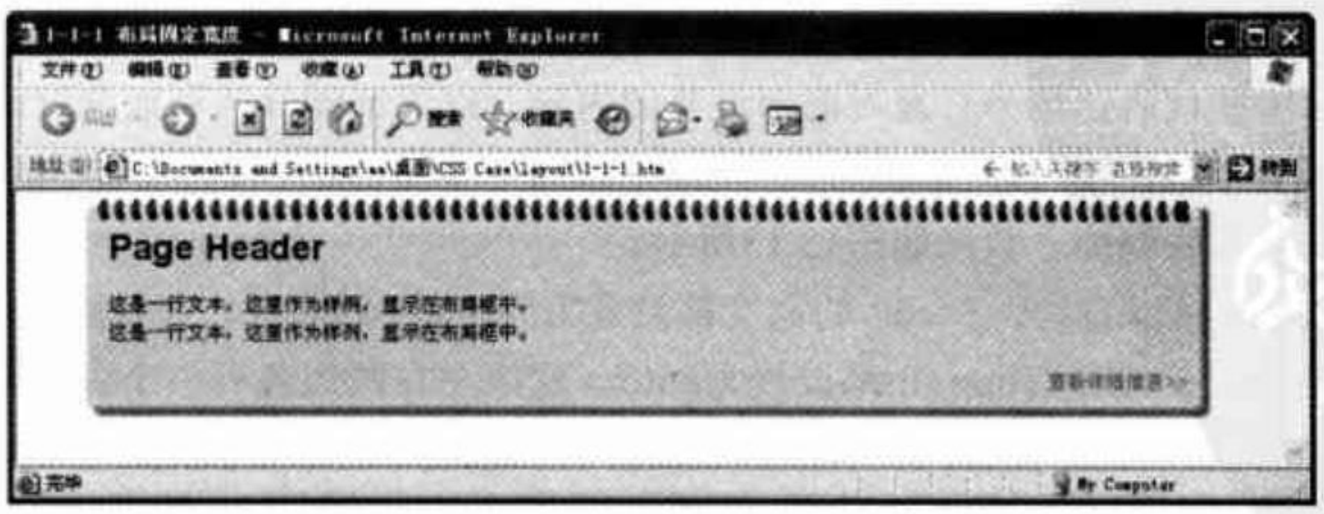


图13.12　修正后在IE 6中显示正确的效果

13.2.3 放置其他圆角框

接下来，将放置的圆角框再复制出两个，并分别设置id为“content”和“footer”，分别代表“内容”和“页脚”。相关代码如下：

```
<style type="text/css" media="screen">
body {
  ……整体设置……
  }

……这里省略5段固定的关于“.rounded”的样式设置……

/*************以下为增加的样式************************/
#header,
#pagefooter,
#content{
  margin:0 auto;
  width:760px;
  }
</style>

<body>
<div id="header">
  <div class="rounded">
  ……这里省略固定结构的内容代码……
  </div>
</div>
<div id="content">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
<div id="pagefooter">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
```

本章以后的代码都采用这种省略的写法，以省略号代替重复部分。如果读者阅读到这里，不理解省略了哪些代码，请务必复习前面章节的内容，以保证能够顺利地继续学习。

每一个部分中的内容可以随意修改，例如更改每一个部分的标题，以及相应的内容，也可以把段落文字彻底删掉。效果如图13.13所示。

从CSS代码中可以看到，3个div的宽度都设置为固定值760像素，并且通过设置margin的值来实现居中放置，即左右margin都设置为auto，就像左右两边各有一个弹簧一样，把内容挤在页面中央。

此外还需要注意，在每个圆角框的下部都有一行文字，即图中用灰色文字写作“查看详细信息”，这段文字是不能直接删除的，因为它承担着放置背景图像的任务。如果不希望看到这行文字，可以把这几个字删除，但是 和 这对标记不能删除。

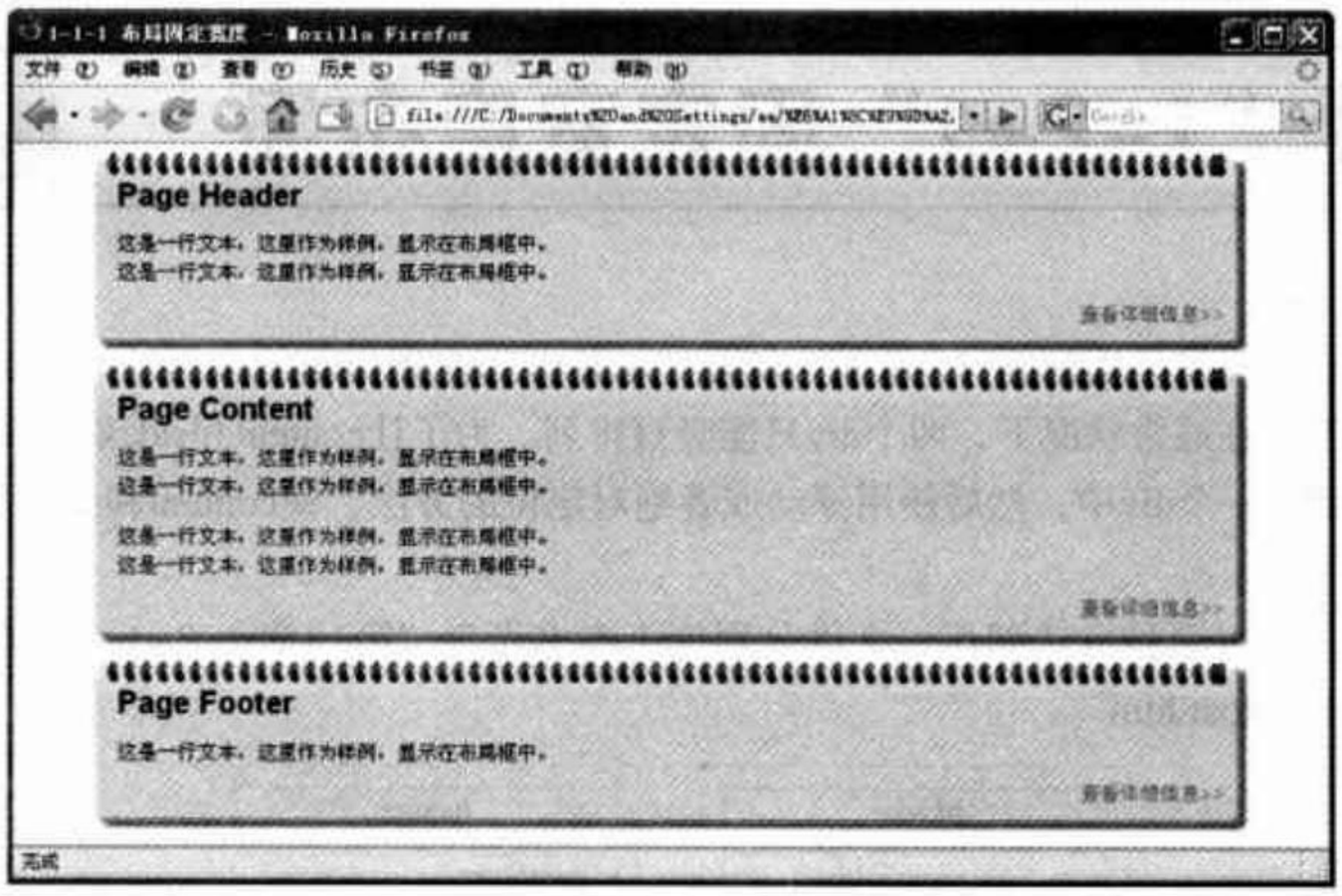


图13.13 实现了单列布局的效果

然而即使删除了这几个字，只要保留了p标记，就会有一定的高度，这样会使每个部分的下部留有较大的一段空白。如果要减小这段空白，可以用CSS将这个p标记的文字高度和行高设置为0，这时效果如图13.14所示。

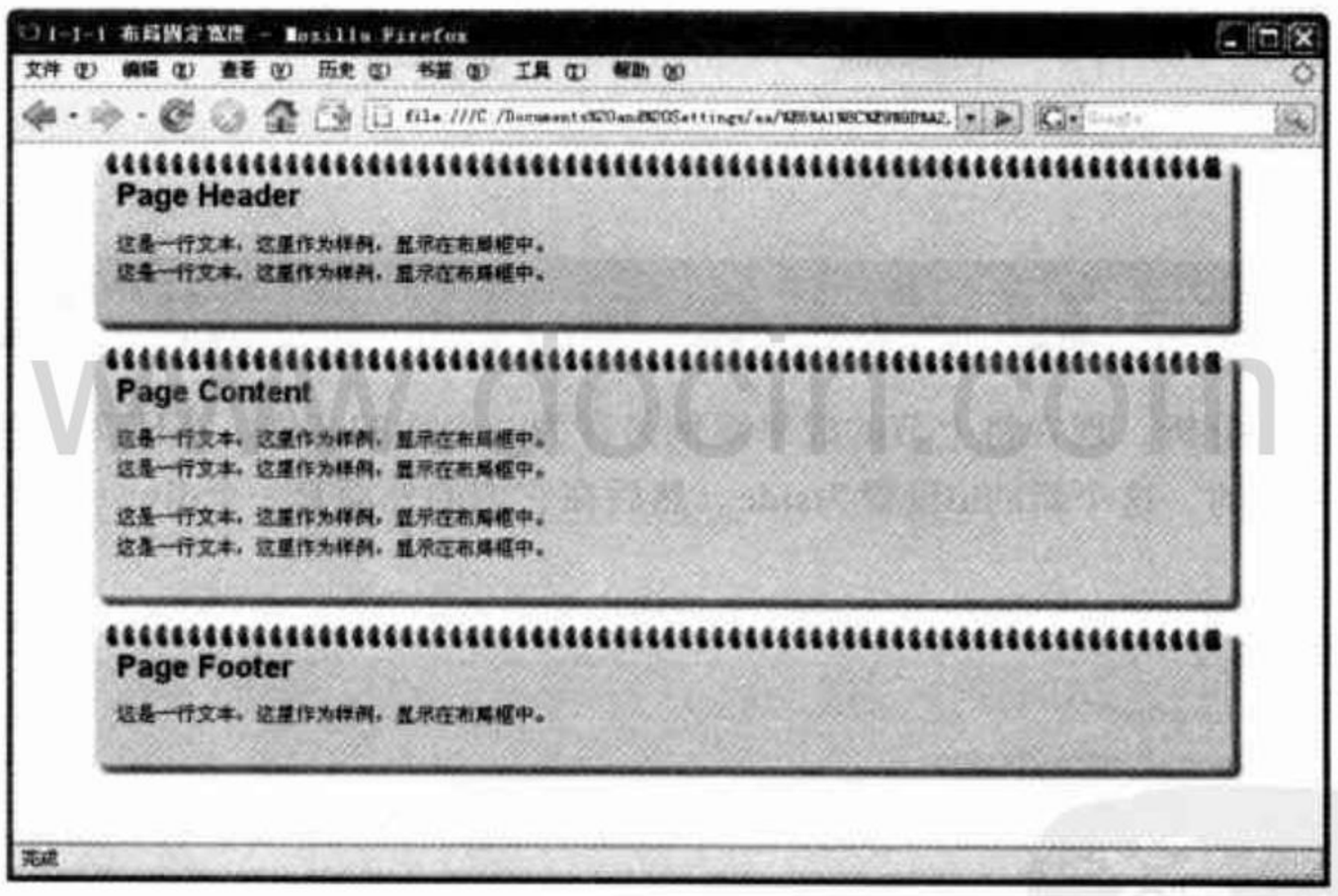


图13.14 隐藏圆角框底部的文字

至此，最简单的一种布局就完成了。

如果希望3个div都紧靠页面的左侧或者右侧，又该怎么办呢？方法很简单，只需要修改3个div的margin值即可，具体的步骤如下。

如果要使它们紧贴浏览器窗口左侧，可以将margin设置为“0 auto 0 0”，即只保留右侧的一根“弹簧”，就会把内容挤到最左边了；反之，如果要使它们紧贴浏览器窗口右侧，可以将margin设置为“0 0 0 auto”，即只保留左侧的一根“弹簧”，就会把内容挤到最右边了。

13.3 “1-2-1”固定宽度布局

现在来制作最经常用到的“1-2-1”布局。如图13.15左图所示的布局结构中，增加了一个“side”栏。但是在通常状况下，两个div只能竖直排列。为了让content和side能够水平排列，必须把它们放到另一个div中，然后使用浮动或者绝对定位的方法，使content和side并列起来，如图13.15右图所示。

本案例将通过两种方法制作，文件分别位于本书光盘“第13章\1-2-1-absolute.htm”和“第13章\1-2-1-float.htm”。

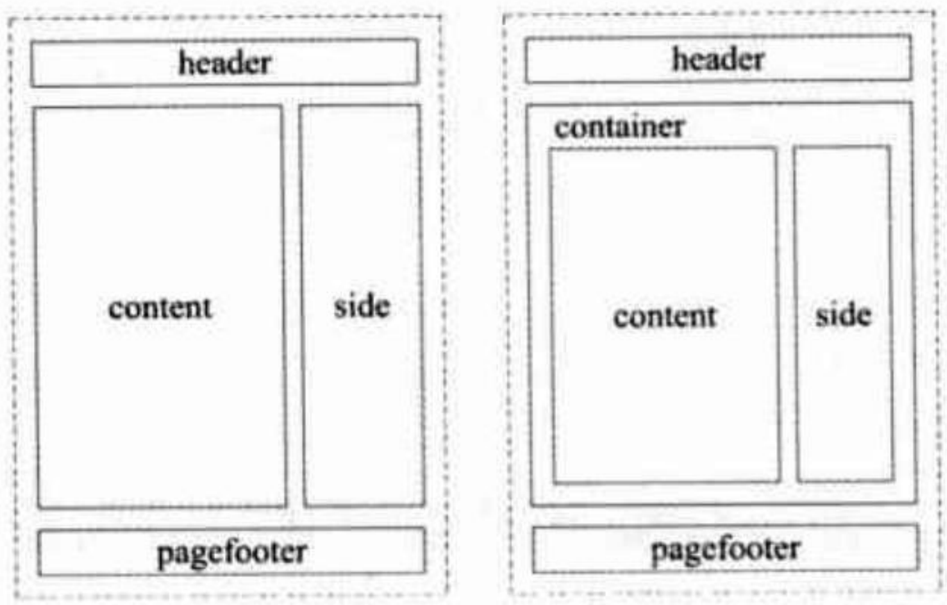


图13.15 “1-2-1”布局的结构示意图

13.3.1 准备工作

基于上面的分析，现在将上节的成果案例另存为一个新的文件。在HTML中把content部分复制出一个新的，这个新的id设置为side。然后在它们的外面套一个div，命名为container。关键代码如下：

```
<body>
<div id="header">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
<div id="container">
  <div id="content">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
</div>
<div id="side">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
```

```
    </div>
  </div>
  <div id="pagefooter">
    <div class="rounded">
          ……这里省略固定结构的内容代码……
    </div>
  </div>
```

在content框中除了文字还可以插入一个图片。下面设置CSS样式，代码如下：

```
#header,#pagefooter,#container{
  margin:0 auto;
  width:760px;
  }
#content{
  }
#content img{
  float:right;
  }
#side{
  }
```

其中，#container、#header和#pagefooter并列使用相同的样式；而#content和#side的样式暂时先空着；“#content img”用来把其中的图片设置为右对齐。这时的效果如图13.16所示。

可以看到，杯子图片由于使用了浮动设置，已经脱离标准流，文字会围绕它排列。对于content和side两个div，现在的关键是如何使它们横向并列。这里有不同的方法可以实现。

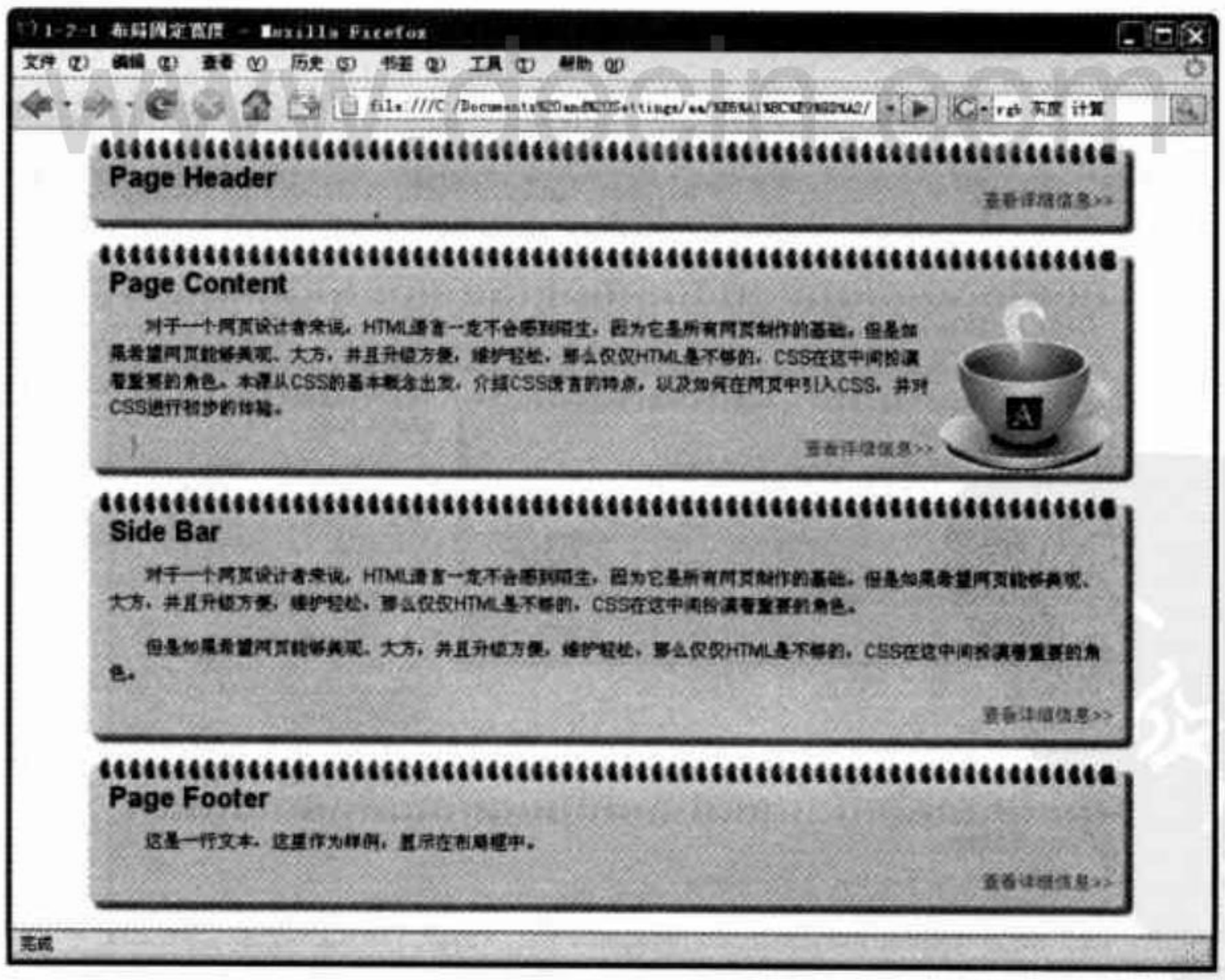


图13.16　“1-2-1”布局准备工作完成后的效果

13.3.2 绝对定位法

首先我们用绝对定位的方法实现，相关代码如下。这种方法制作的案例文件位于本书光盘“第13章\1-2-1-absolute.htm”。

```
#header,#pagefooter,#container{
  margin:0 auto;
  width:760px;
  }
#container{
  position:relative;
  }
#content{
  position:absolute;
  top:0;
  left:0;
  width:500px;
  }
#side{
  margin:0 0 0 500px;
  }
```

为了使content能够使用绝对定位，必须考虑用哪个元素作为它的定位基准。显然应该是container这个div。因此将#contatiner的position属性设置为relative，使它成为下级元素的绝对定位基准，然后将content这个div的position设置为absolute，即绝对定位，这样它就脱离了标准流，side就会向上移动占据原来content所在的位置。将content的宽度和side的左margin设置为相同的数值，就正好可以保证它们并列紧挨着放置，且不会相互重叠。

这时的效果如图13.17所示。读者可以参考光盘“第13章\1-2-1-absolute.htm”文件。

图13.17 使用“绝对定位法”实现的“1-2-1”布局

注意　这种方法实现了中间的两列左右并排，但是它存在一个缺陷。当右边的side比左边content高时，显示效果不会有问题，但是如果左边的content栏比右边的side栏高的话，显示就会有问题了。因为此时content栏已经脱离标准流，对container这个div的高度不产生影响，从而pagefooter的位置只根据右边的side栏确定。例如，现在在content栏中增加一个圆角框，这时效果如图13.18所示。

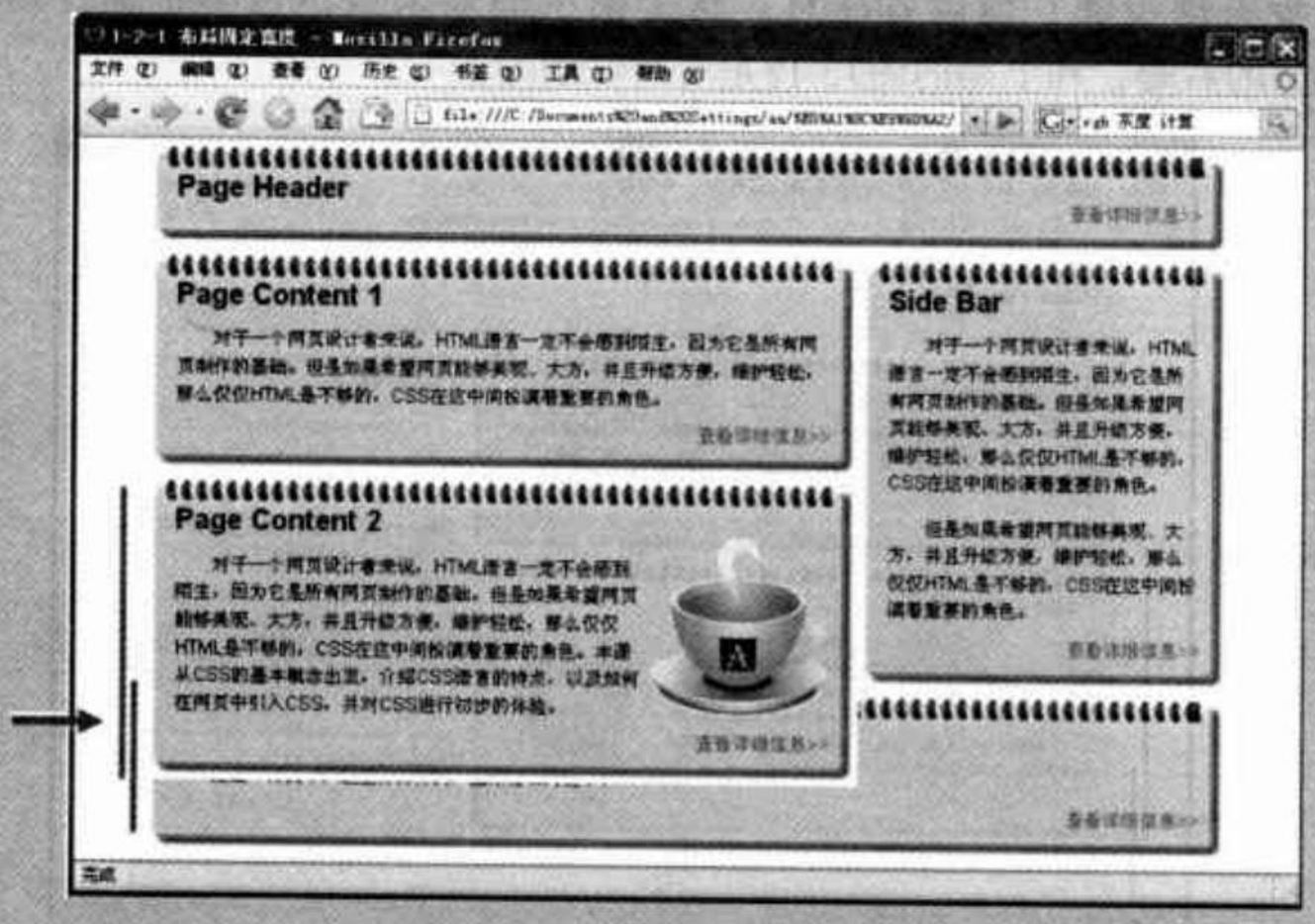


图13.18　出现问题的页面

这是绝对定位带来的固有问题。如果用这种办法使几个div横向并列，就必须知道哪一列是最高的，并使该列保留在标准流中，使它作为“柱子”，撑起这一部分的高度。

13.3.3 浮动法

还可以换一个思路，使用“浮动”来实现这种布局。将刚才的文件另存为一个新文件。在新文件中，HTML部分代码完全不作修改。在CSS样式部分，稍作修改，将#container的position属性去掉，#content和#side都设置为向左浮动，二者的宽度相加等于总宽度。例如这里将它们的宽度分别设置为500像素和260像素。

相关代码如下。这种方法制作的案例文件位于本书光盘“第13章\1-2-1-float.htm”。

```
#header,#pagefooter,#container{
  margin:0 auto;
  width:760px;
  }
#content{
  float:left;
  width:500px;
  }
#content img{
  float:right;
  }
```

```
#side{
  float:left;
  width:260px;
  }
```

此时的效果如图13.19所示。为什么pagefooter的位置还是不正确呢？请读者思考，到这里还差哪一步关键步骤？请读者注意，这个图中的效果虽然也不正确，但是仔细观察pagefooter部分的右端，和上面的图13.18是有所区别的。

图13.19 使用浮动方法设置的布局效果

答案是此时还需要对#pagefooter设置clear属性，以保证清除浮动对它的影响，代码如下：

```
#pagefooter{
  clear:both;
  }
```


这时就可以看到正确的效果了，如图13.20所示。

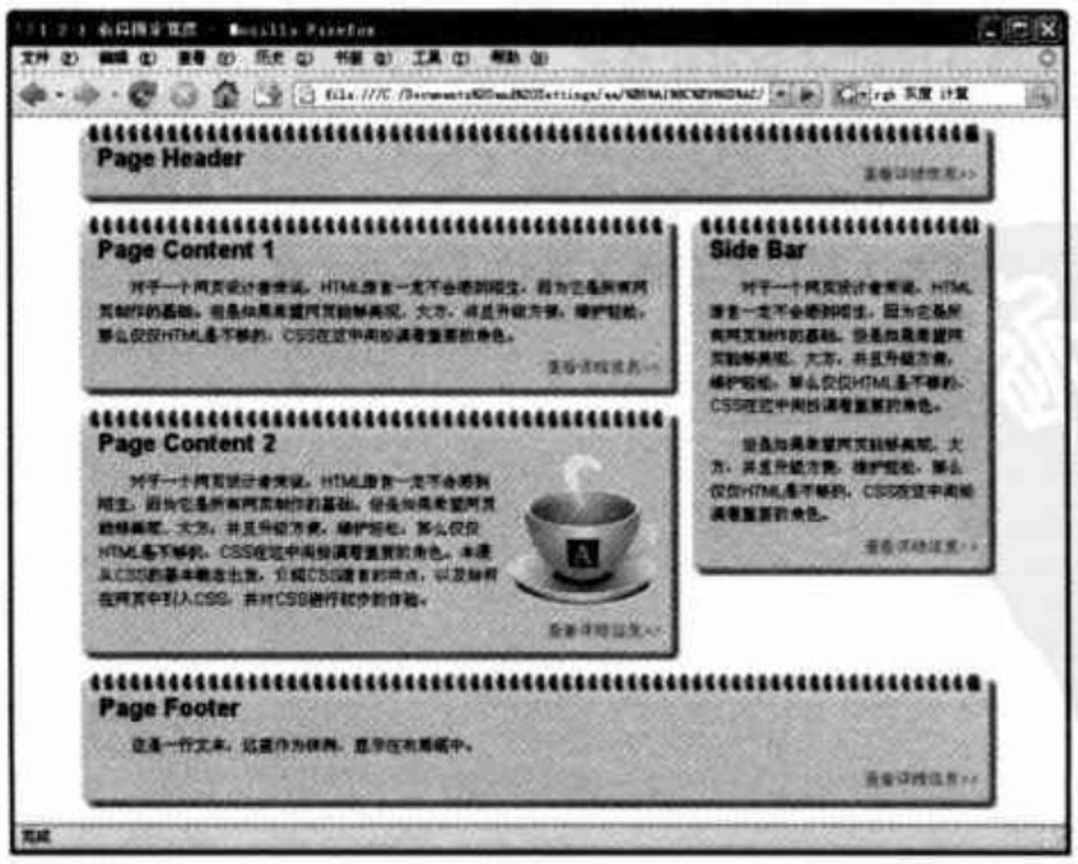


图13.20 使用浮动方法设置的布局效果

使用这种方法时，并排的两列中无论哪一列内容变长，都不会影响布局。例如右边又增加了一个模块，使内容变长，排版效果同样是正确的，如图13.21所示。

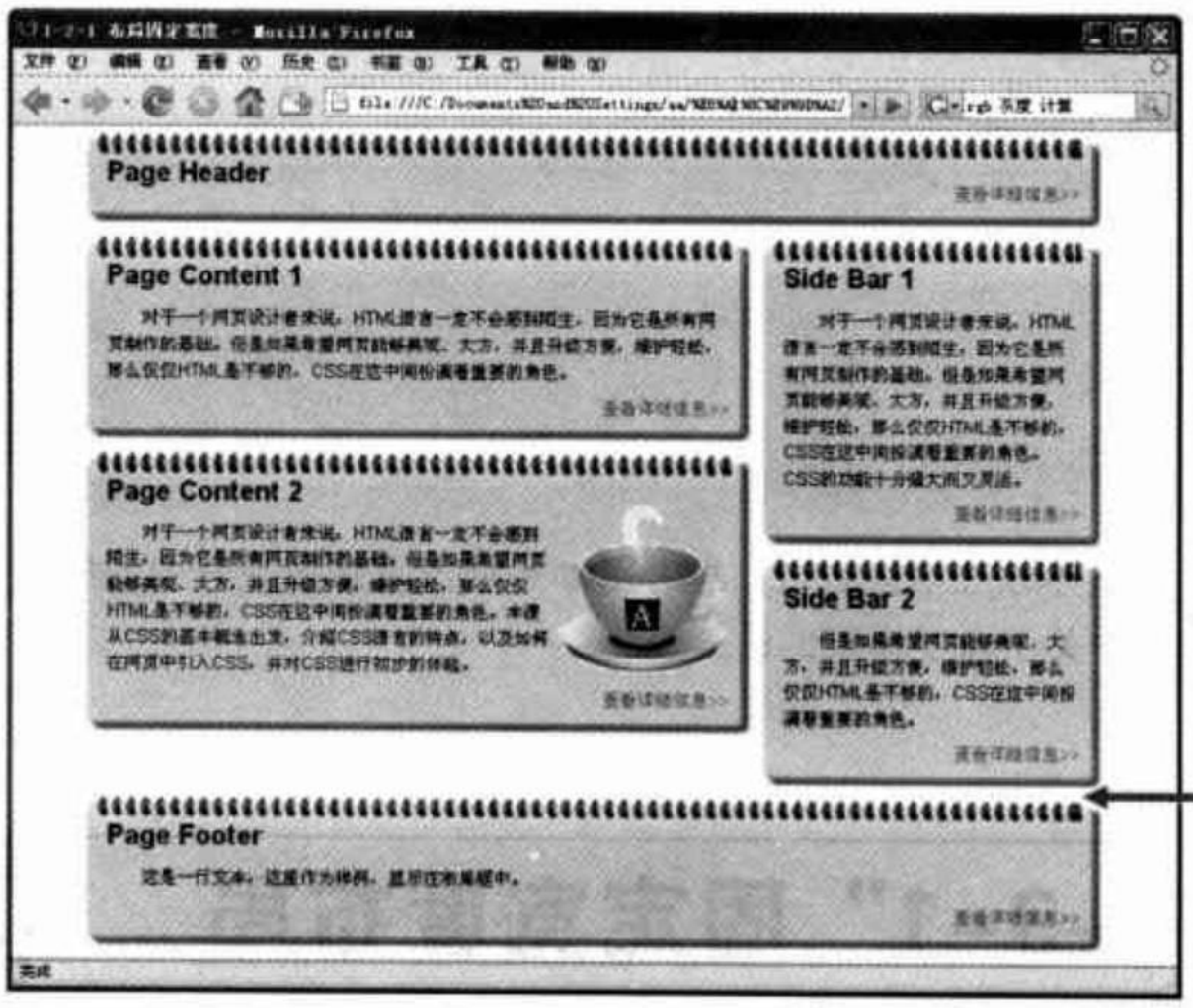


图13.21　右侧的列变高效果同样正确

到这里“1-2-1”布局方式我们已经完全可以自由发挥了。只要保证每一个模块自身代码正确，同时使用正确的布局方式，就可以非常方便地放置各模块。

这种方法非常灵活，例如要side从页面右边移动左边，即交换与content的位置，只需要稍微修改一处CSS代码，即可以实现。请读者思考，应该如何修改，以实现如图12.22所示的效果？

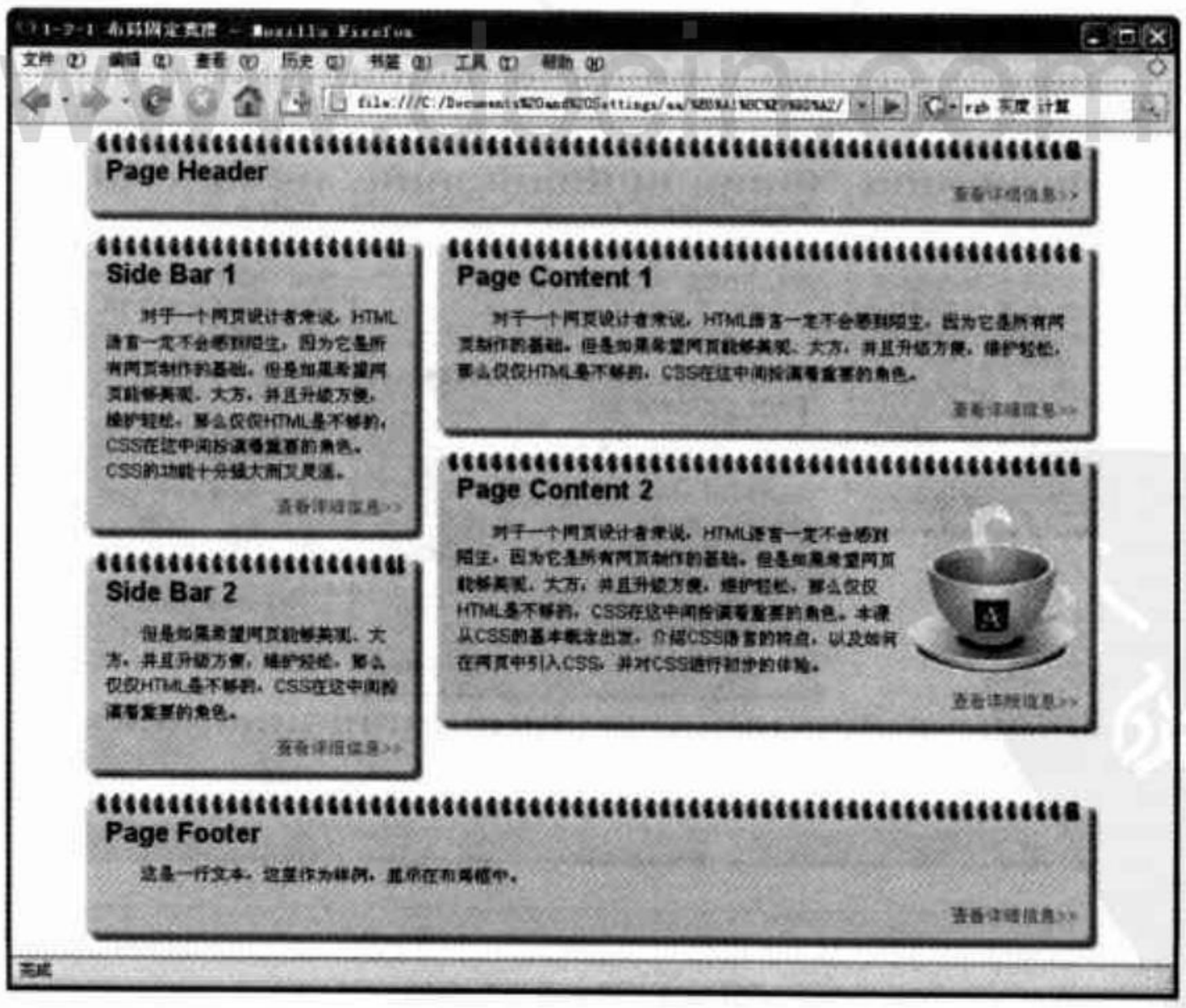


图13.22　左右两侧的列交换位置

答案是将#content的代码由：

```
#content{
  float:left;
  width:500px;
  }
```

修改为：

```
#content{
  float:right;
  width:500px;
  }
```

这样就可以了。具体原理请读者自己思考。如果还没有想清楚其中的奥妙，请仔细阅读本书的第3章和第4章中关于盒子模型的讲解。

13.4 “1-3-1”固定宽度布局

下面以“1-2-1”布局为基础制作“1-3-1”布局。这里仍然使用浮动方式来排列横向并排的3栏，效果如图13.23所示。

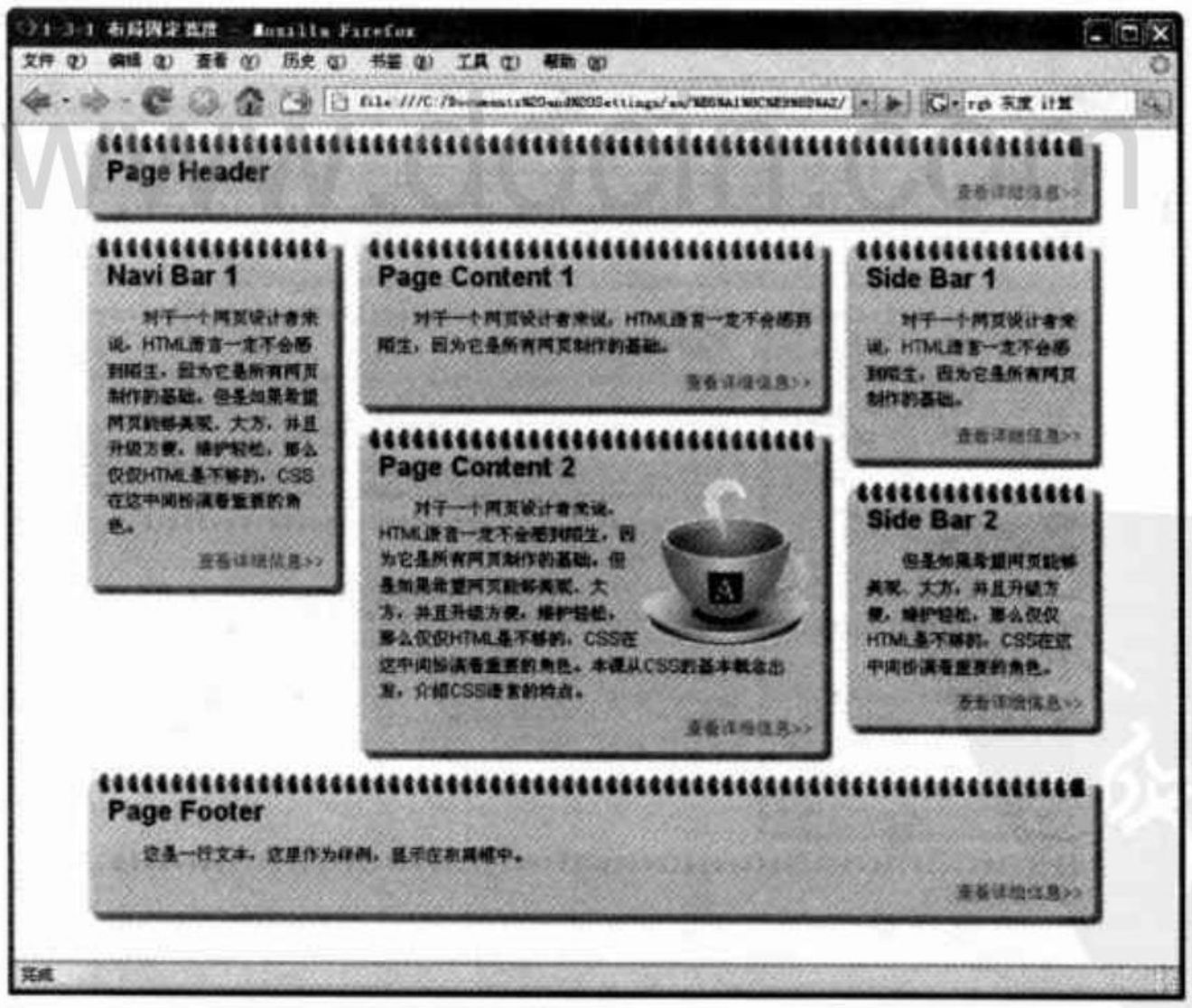


图13.23 “1-3-1”布局

这种布局同样可以用两种方法制作，案例文件分别位于本书光盘“第13章\1-3-1-absolute.htm”和“第13章\1-3-1-float.htm”。

对于这个页面，要在“1-2-1”布局的基础上修改HTML的结构，只需在container中的左边增加一列即可，这里将新增加的列命名为navi，结构如图13.24所示。

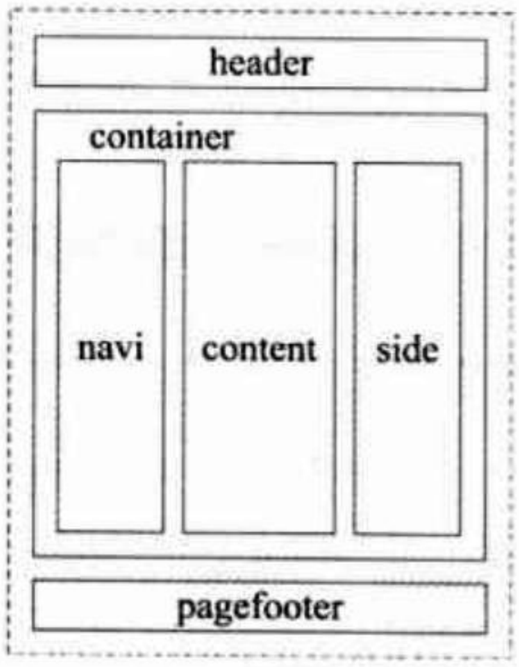


图13.24 “1-3-1”布局结构示意图

相信读者已经可以自己写出相应的HTML代码，并使用“绝对定位法”和“浮动法”实现所需的效果，这里就不再赘述了。这里仅给出“浮动法”的CSS样式关键代码。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:760px;
  }
#navi{
  float:left;
  width:200px;
  }
#content{
  float:left;
  width:360px;
  }
#content img{
  float:right;
  }
#side{
  float:left;
  width:200px;
  }
#pagefooter{
  clear:both;
  }
```

#navi、#content和#side这3栏都使用浮动方式，3列的宽度之和正好等于总宽度。

13.5 “1-((1-2)+1)-1”固定宽度布局

下面再来实现一个“1-((1-2)+1)-1”的布局。如果读者还不清楚这种布局方式的表示方法，请阅读本章的13.1.4节。本案例的最终效果如图13.25所示。

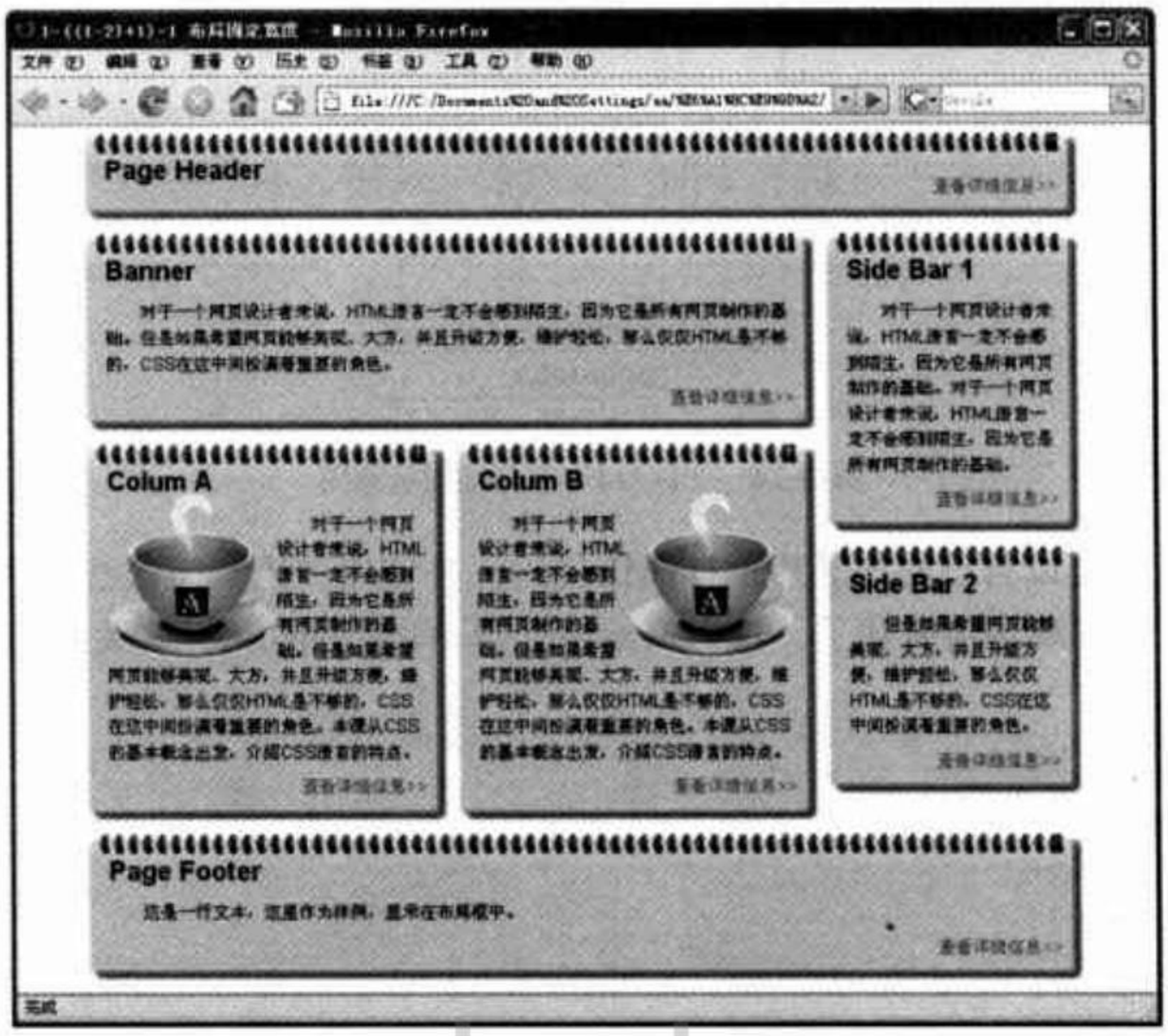


图13.25 “1-((1-2)+1)-1”布局效果

案例文件位于本书光盘“第13章\1-((1-2)+1)-1.htm”。

这种布局的示意图如图13.26左图所示。然而当真正要实现这个布局的时候，仅通过这个图还不能表现出各个div之间的结构关系，因为还需要有嵌套的div藏在中间。把这些div都展示出来，如图13.26右图所示。

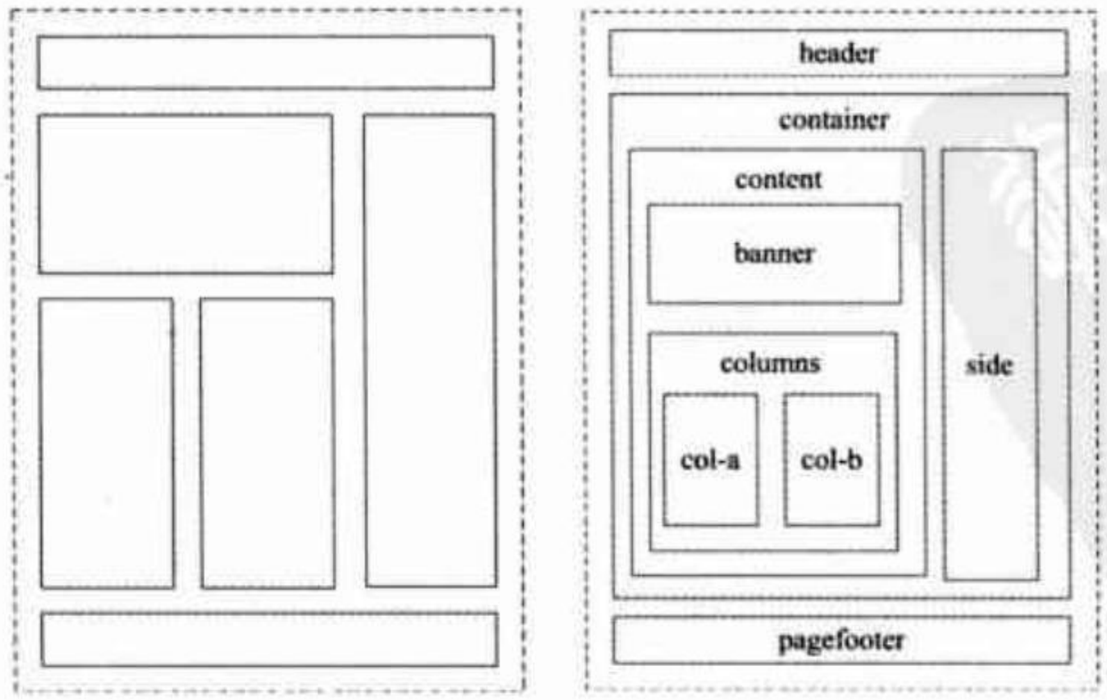


图13.26 “1-((1-2)+1)-1”布局结构示意图

这个案例的HTML结构比较复杂，在写代码的时候，应尽可能缩进排列代码，并加上易于理解的注释。特别是每个 是和哪个 相互对应的，应该通过注释的方式写清楚。请注意下面代码中粗体字的注释语句，这样就不容易混乱了。HTML的关键代码如下。

```
<body>
<div id="header">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div> <!-- end of header -->

<div id="container">
  <div id="content">
        <div id="banner">
              <div class="rounded">
                    ……这里省略固定结构的内容代码……
              </div>
        </div><!-- end of banner -->
        <div id="columns">
              <div id="col-a">
                    <div class="rounded">
                          ……这里省略固定结构的内容代码……
                    </div>
              </div><!-- end of col-a -->
              <div id="col-b">
                    <div class="rounded">
                          ……这里省略固定结构的内容代码……
                    </div>
              </div><!-- end of col-b -->
        </div><!-- columns -->
  </div><!-- end of content -->
  <div id="side">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
  </div><!-- end of side -->
</div><!-- end of container -->

<div id="pagefooter">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div><!-- end of pagefooter -->
</body>
```

相应的CSS样式代码如下所示，可以看到是非常简洁的。

```
#header,
#pagefooter,
```

```
#container{
  margin:0 auto;
width:760px;
  }
#container  #content{
  float:left;
  width:560px;
  }
#container  #content  #columns  #col-b,
#container  #content  #columns  #col-a {
  float:left;
  width:280px;
  }
#container  #content  #columns  #col-a  img{
  float:left;
  }
#container  #content  #columns  #col-b  img{
  float:right;
  }
#container  #side{
  float:left;
  width:200px;
  }
#pagefooter{
  clear:both;
  }
```

上面一共有7段CSS样式代码，其中：

- 第1段是设置最外层的3个div的宽度和居中样式；
- 第2段设置container中的content向左浮动，并设置固定宽度560像素；
- 第3段设置content的columns中的左右两个分栏向左浮动，并均设置为280像素宽；
- 第4段和第5段分别设置columns中两个分栏中的图像的文字环绕方式；
- 第6段设置container中右侧边栏向左浮动，并设置它的宽度为200像素；
- 第7段设置最底下的pagefooter，清除上面的浮动对它的影响。

13.6 魔术布局

先来看一个有趣的网页，网址是http://www.uxmag.com，这是一个在线杂志网站，希望读者能够立即上网访问这个网站的首页。显示器必须是1024×768以上的分辨率，才能看到它的特殊效果。当浏览器窗口的宽度达到1024像素的时候，页面如图13.27左图所示；当浏览器窗口的宽度变窄时，页面最右侧的部分会移动到页面左侧的下端，如图13.27右图所示。这个效果确实非常有趣，读者可以仔细比较一下左图中的右侧和右图中的下侧相同部分的位置

变化。

在这里我们简单地模仿一下这个效果。当然uxmag.com这个页面作的相当完善，使用了很多JavaScript的基础库，我们这里仅通过布局简单地模仿它的布局变化效果。

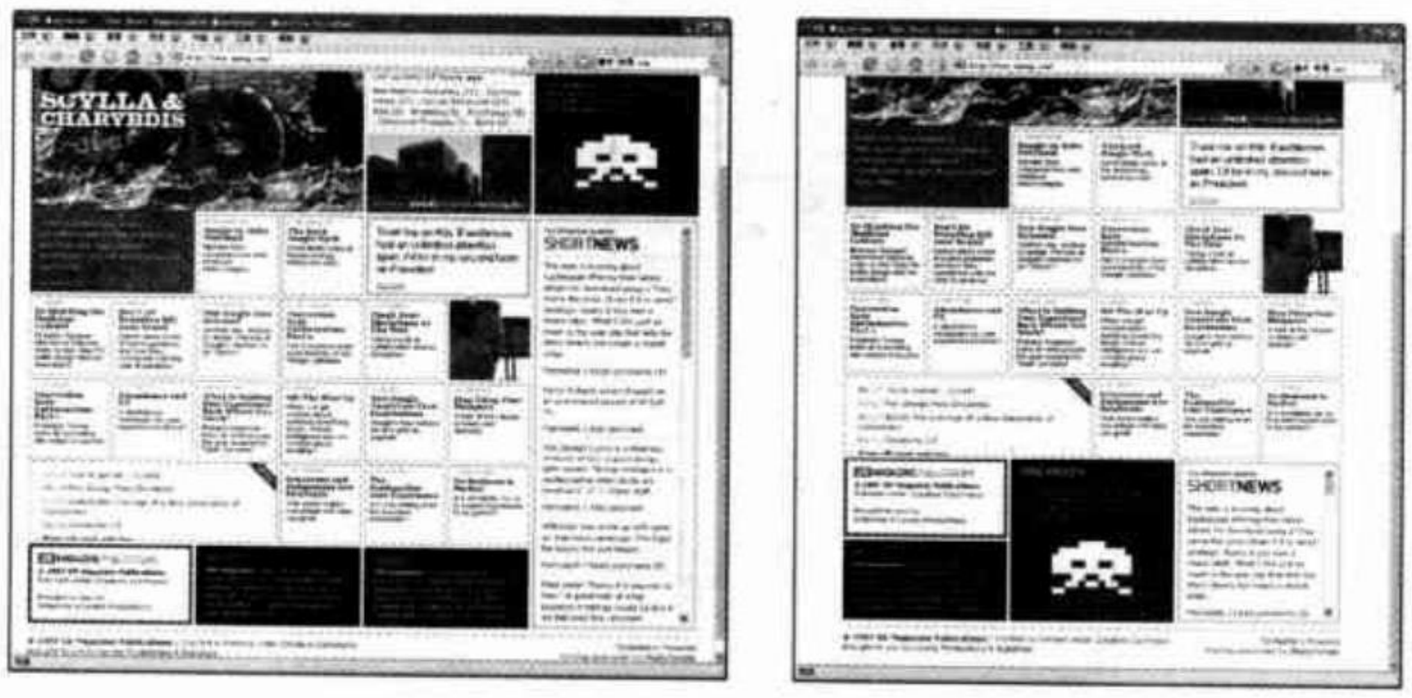


图13.27 “魔术布局”效果

13.6.1 制作步骤

仍以前面的案例为基础，通过一些简单的改造，实现下面的效果。首先，当浏览器超过800像素宽的时候，页面内容宽度固定为800像素，并居中显示，如图13.28所示。

图13.28 右侧栏中竖直排列着3个模块

当浏览器窗口逐渐变窄，小于800像素的时候，右侧的3个竖直排列的模块会自动移到左侧的下面横向排列，如图13.29所示。如果浏览器窗口的宽度小于600像素时，页面的内容不再变窄，而是在浏览器下端出现横向滚动条。

完成以后的案例文件位于本书光盘“第13章\magic-hover.htm”。

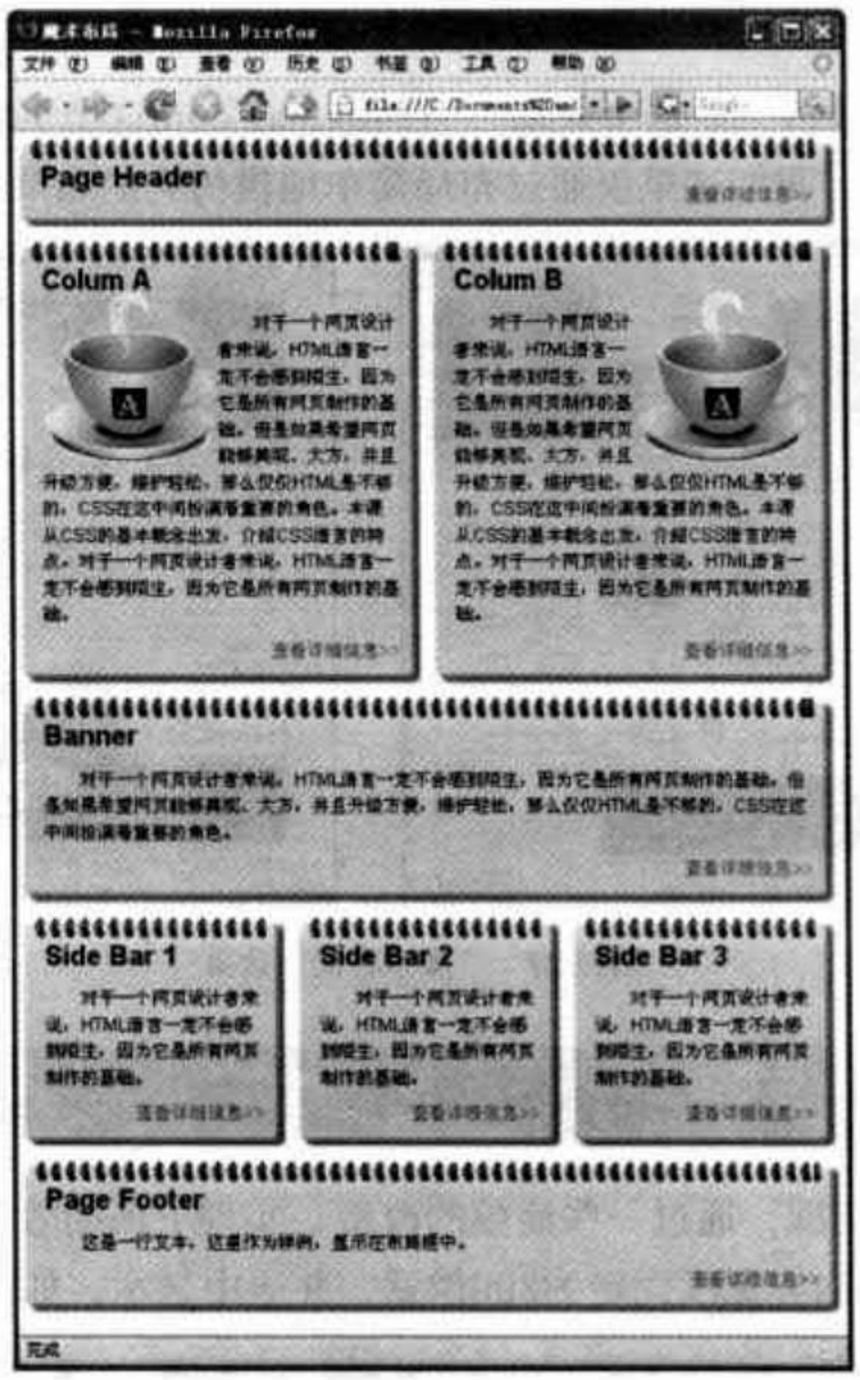


图13.29 窗口变窄时右侧的模块移动到页面下方

下面具体介绍如何实现这个效果。从结构来分析，在浏览器窗口宽于800像素时，布局结构为“1-((2-1)+1)-1”；当浏览器窗口窄于800像素时，其布局结构为“1-2-1-3-1”。

① 首先把前面制作的“1-((1-2)+1)-1”布局页面改造为“1-((2-1)+1)-1”。这一步很容易实现，只需要把banner这个div从columns这个div的上面移动columns的下面。但是要注意一点，由于移动到左右两列的下面，而这两列又是浮动的，因此要对banner这个div增加clear属性，以保证清除上面的col-a和col-b对它的影响。

② 原来在右侧的是一个id为side的div，会导致在side中的各个div都按照标准流方式排列，这样即使side将来被移动到banner的下方，它也无法横向排列，因此必须对它进行改造。方法是，把side由id选择器变为类选择器，然后将每一个右侧的3个“魔术div”都设置为“.side”类别。另外，原来的案例中，右侧是两个模块，现在再增加一个，这样3个.side类别的div就直接与content这个大div并列了。

③ 调整一下宽度。原来案例中，content的宽度是560像素，side的宽度是200像素，现在将content的宽度调整为600像素，col-a和col-b的宽度调整为300像素，这样在content的下面正好可以放下3个原来在右边的矩形。

测试一下实际的效果。在宽度小于800像素的时候，这3个侧边的模块确实如期望的那样移动到了页面下侧，如图13.30所示。

现在已经基本上实现了最开始希望获得“魔术布局”效果。但是如果严格要求的话，还有一些缺陷。

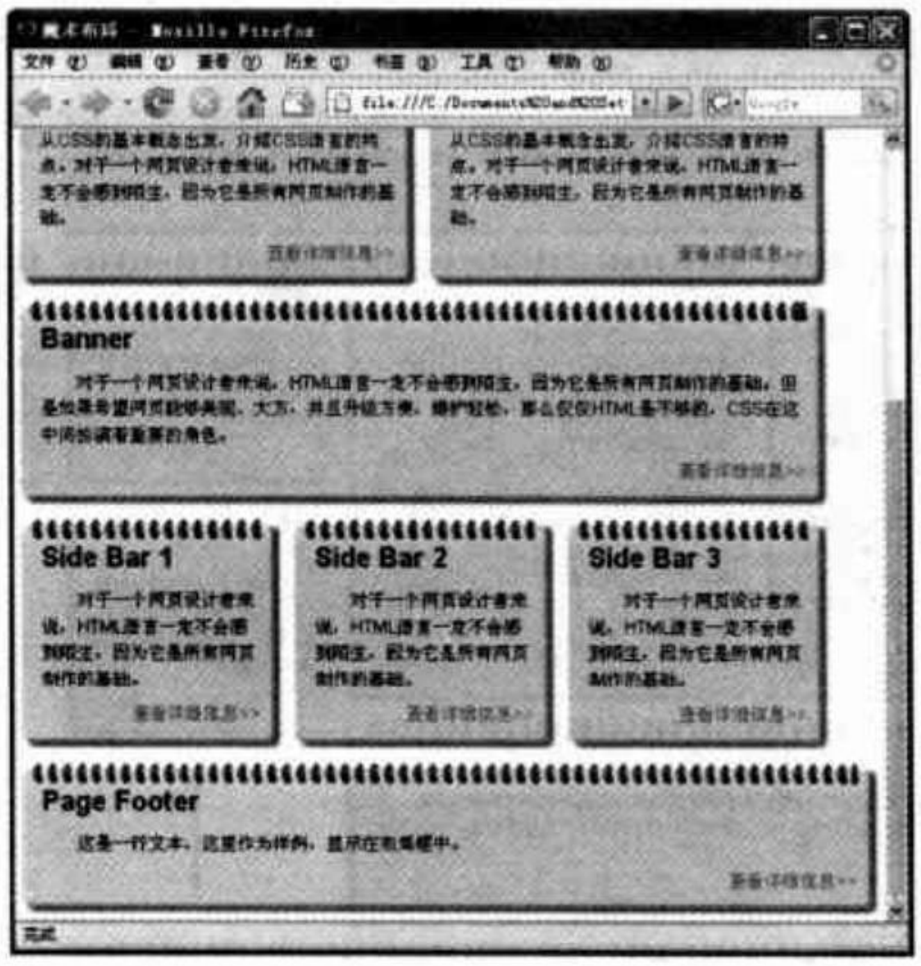


图13.30　右侧的“魔术div”移动到页面下方

13.6.2 修正缺陷

当浏览器窗口继续变窄，小于600像素的时候，横排已经放不下并排的3个模块了，这时第3个模块被挤到了下面，如图13.31所示。

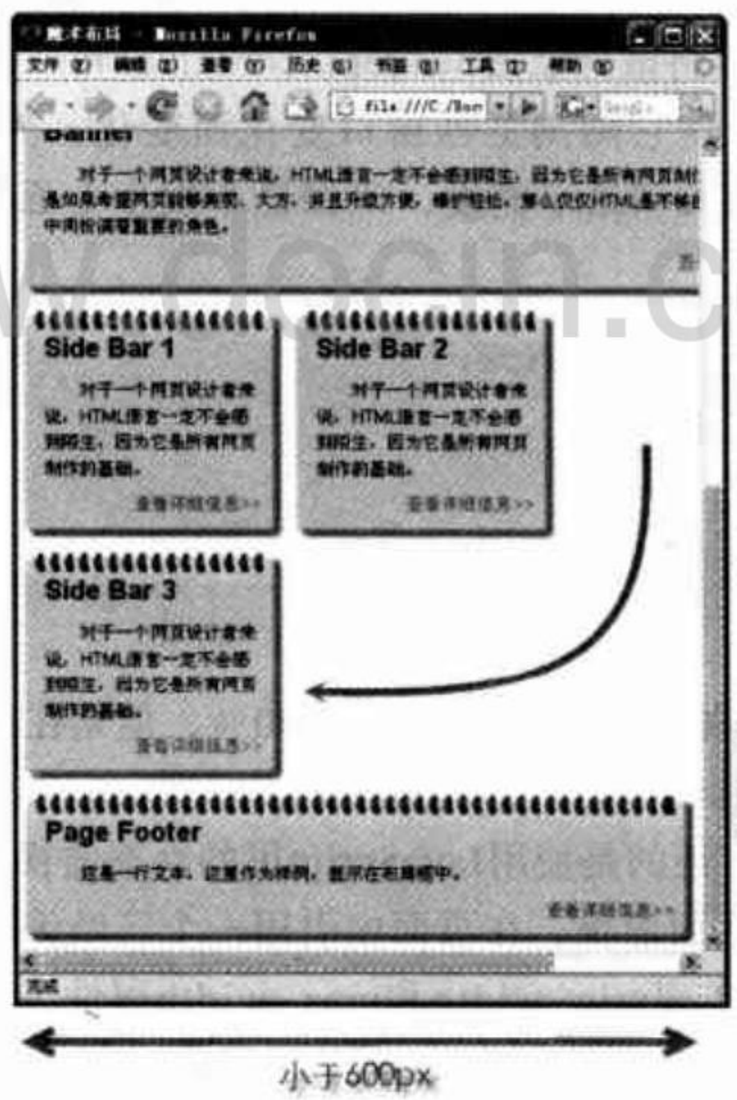


图13.31　窗口继续变窄时“魔术div”继续向下移动

相反，如果浏览器窗口宽度超过1000像素，“Side Bar 2”模块由于向左浮动，就会跑到“Side Bar 1”模块的右边，如图13.32所示。

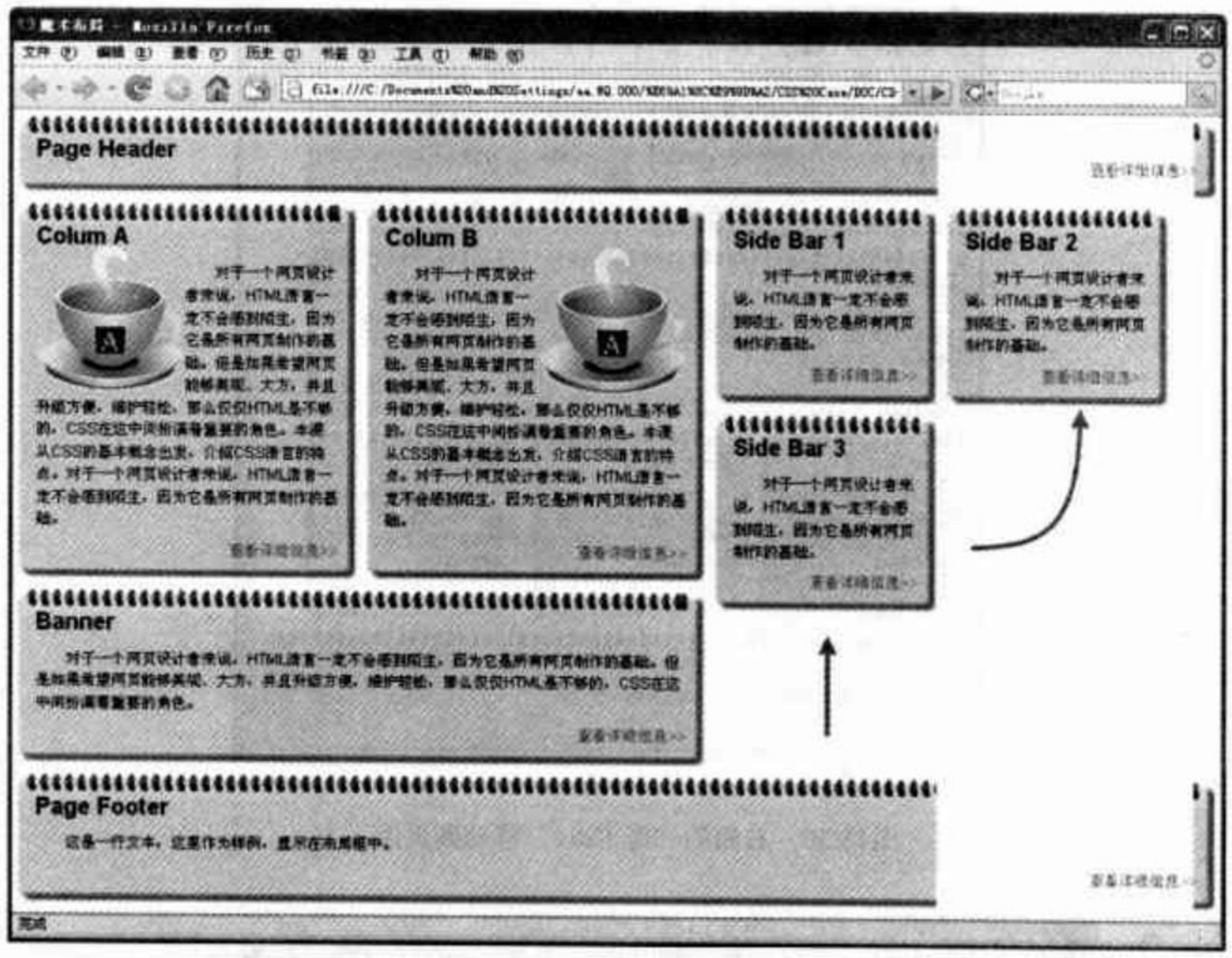


图13.32 窗口宽度过宽时继续向右浮动

为了避免上述这两种情况的发生，需要限制页面内容的宽度。此时要用到CSS中的两个属性——min-width和max-width，即“最小宽度”和“最大宽度”。

利用这两个属性，可以将最外层的header、container和pagefooter这3个div的宽度（即width属性）设为100%，使它随浏览器窗口变化而变化，然后通过min-width和max-width这两个属性限制其最大和最小宽度，以保证不发生上面显示的两种错误情况。代码如下。


```
#header,#pagefooter,#container{
  margin:0 auto;
  width:100%;
  min-width: 600px;
  max-width: 800px;
  }
```

实际上，不设置width属性时，其值相当于100%，但是在IE 6中，不支持min-width和max-width这两个属性，因此必须要想办法在IE 6中实现min-width和max-width的效果。目前有几种解决方法，比较方便的是使用JavaScript里的动态监视浏览器窗口，然后利用DOM设置对象的宽度。具体做法很简单，在页面中引用一个已经编写好的JavaScript程序，其他不用作任何设置，就可以实现min-width和max-width属性。但是经过测试，如果不设置width:100%，效果就不是很好，因此这里就设置了“width:100%”属性。

这个JavaScript程序是一位英国程序员Andrew Clover在2003年编写的，他的个人网站的网址是http://www.doxdesk.com。

Firefox和IE 7都支持min-width和max-width属性，因此只需对IE 6使用这个JavaScript程序。这里可以使用IE的条件注释语句，判断一下浏览器，只有当前使用的是IE 6或更低版本

的浏览器时才装载这个js文件。代码如下：

```
<!--[if lte IE 6]>
     <script type="text/javascript" src="minmax.js"></script>
<![endif]-->
```

这样，就可以比较完美地实现“魔术布局”的效果了。本案例最终的代码如下。

```
<head>
<style>
……前面的部分省略……
#header,#pagefooter,#container{
  margin:0 auto;
  width:100%;
  min-width: 600px;
  max-width: 800px;
  }
#container #content{
  float:left;
  width:600px;
  }
#container #content #col-b,
#container #content #col-a {
  float:left;
  width:300px;
  }
#container #content #col-a img{
  float:left;
  }
#container #content #col-b img{
  float:right;
  }
#container .side{
  float:left;
  width:200px;
  }
#container #content #banner,
#pagefooter{
  clear:both;
}
</style>
<!--[if lte IE 6]>
     <script type="text/javascript" src="minmax.js"></script>
<![endif]-->
</head>

<body>
<div id="header">
        ……这里省略固定结构的内容代码……
</div> <!-- end of header -->
<div id="container">
  <div id="content">
        <div id="columns">
```

```
<div id="col-a">
        <div class="rounded">
                        ……这里省略固定结构的内容代码……
        </div>
</div><!-- end of col-a -->
<div id="col-b">
        <div class="rounded">
                        ……这里省略固定结构的内容代码……
        </div>
</div><!-- end of col-b -->
        </div><!-- columns -->
        <div id="banner">
                <div class="rounded">
                                ……这里省略固定结构的内容代码……
                </div>
        </div><!-- end of banner -->
</div><!-- end of content -->
<div class="side">
        <div class="rounded">
                        ……这里省略固定结构的内容代码……
        </div>
</div>
<div class="side">
        <div class="rounded">
                        ……这里省略固定结构的内容代码……
        </div>
</div>
  <div class="side">
        <div class="rounded">
                        ……这里省略固定结构的内容代码……
        </div>
  </div><!-- end of side -->
</div><!-- end of container -->
<div id="pagefooter">
  <div class=" rounded" >
                ……这里省略固定结构的内容代码……
  </div>
</div><!-- end of pagefooter -->
</body>
```


13.7 本章小结

在本章中，以几种不同的布局方式，演示了如何灵活地运用CSS的布局性质，使页面按照需要的方式进行排版。特别需要读者掌握的有以下3个方面。

（1）页面结构的分析方法。只有先正确地分析出布局结构的表达式，然后画出结构示意图，才能正确地进行下一步编写代码的工作。

（2）对横向并列的div使用“绝对定位法”进行布局，并了解它的缺陷及其产生原因。

（3）对横向并列的div使用“浮动法”进行布局。

第14章
变宽度网页布局剖析与制作

在上一章中，对固定宽度的页面布局做了比较深入地分析和讲解。在本章中，将对变宽度的页面布局做进一步的分析。变宽度的布局要比固定宽度的布局复杂一些，根本的原因在于，宽度不确定，导致很多参数无法确定。因此必须使用一些技巧来完成。

这里将介绍一些国外的CSS领域著名的设计师的研究成果，并对变宽度网页布局的总体情况进行归纳。希望读者能够保持清晰的思路，这样在实际工作中遇到具体的案例时，就可以灵活地选择解决方法。

14.1 “1-2-1”变宽度网页布局

“1-1-1”布局过于简单，因此这里就不再介绍了。我们从“1-2-1”布局开始，逐步向读者展示更为复杂的布局结构，并逐步归纳出普遍的通用解决方案。

对于变宽度的布局，首先要使内容的整体宽度随浏览器窗口宽度的变化而变化。因此，中间的container容器中的左右两列的总宽度也会变化，这样就会产生不同的情况。这两列是按照一定的比例同时变化，还是一列固定，另一列变化。这两种情况都是很常用的布局方式。下面先从等比例方式讲起。

注意 在分列情况下，某一列有可能是固定宽度的，也可能是变化宽度的。因此为了方便区分，这里再修订一下布局的表达方法，这样对于光盘中的文件命名也会比较方便。规定为，对于并列的若干列，如果某一列是固定列，就用字母“f”（英文单词fixed的第一个字母）表示；如果某一列是变宽的列，就用字母“l”（英文单词“liquid”的第一个字母）表示。

例如，如果某一个“1-2-1”布局中两列并排，左边的是固定宽度列，右边的是变化宽度的列，那么这种布局记作“1-(f-l)-1”。

再例如，如果某一个“1-3-1”布局中3列并排，左右两边的是固定宽度列，中间的是变化宽度的列，那么这种布局记作“1-(f-l-f)-1”。

本书光盘中的案例文件采用了这种方法命名，读者一看就可以知道是如何布局的了。

14.1.1 “1-2-1”等比例变宽布局

首先实现按比例的适应方式，如图14.1所示。在这个页面中，网页内容的宽度为浏览器窗口宽度的85%，页面中左侧的边栏的宽度和右侧的内容栏的宽度保持1：2的比例，可以看到无论浏览器窗口宽度如何变化，它们都等比例变化。

本案例的文件位于本书光盘的“第14章\1-2-1\1-(l+l)-1-absolute.htm”和“第14章\1-2-1\1-(l+l)-1-float.htm”。前者使用“绝对定位法”制作，后者使用“浮动法”制作。

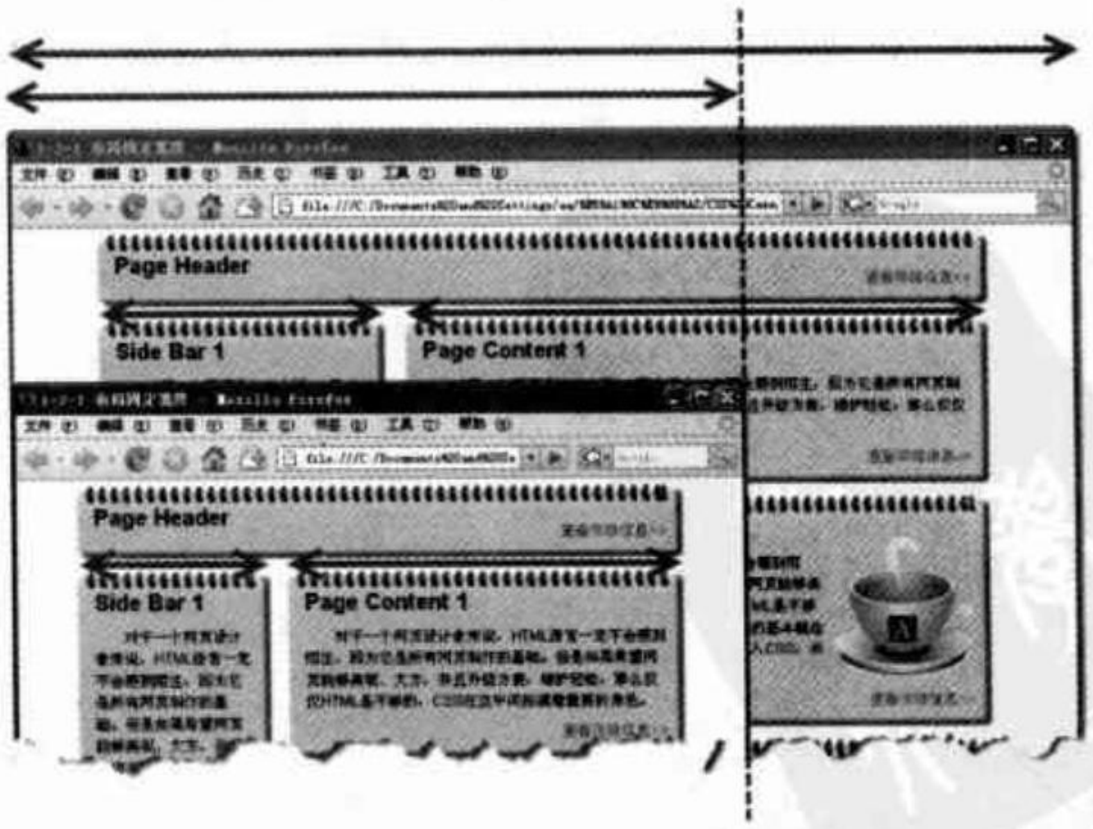


图14.1 “1-2-1”等比例布局

我们将在前面制作的“1-2-1”浮动布局的基础上完成本案例。原来的“1-2-1”浮动布局中的宽度都是用像素数值确定的固定宽度，下面就来对它进行改造，使它能够自动调整各个模块的宽度。

实际上只需要修改3处宽度就可以了。代码如下。

```
#header,#pagefooter,#container{
  margin:0 auto;
  ~~width: 760px;~~ /*删除原来的固定宽度*/
  width:85%;    /*改为比例宽度*/
  }
#content{
  float:right;
  ~~width: 500px;~~ /*删除原来的固定宽度*/
  width:66%;      /*改为比例宽度*/
  }
#side{
  float:left;
  ~~width: 260px;~~ /*删除原来的固定宽度*/
  width:33%; /*改为比例宽度*/
  }
```

经验　container等外层div的宽度设置为85%是相对浏览器窗口而言的比例；而后面content和side这两个内层div的比例是相对于外层div而言的。这里分别设置为66%和33%，二者相加为99%，而不是100%，这是为了避免由于舍入误差造成总宽度大于它们的容器的宽度，而使某个div被挤到下一行中。如果希望精确，写成99.9%也可以。

这样就实现了各个div的宽度都会等比例适应浏览器窗口。这里需要注意两点。

（1）确保不要使一列或多个列的宽度太大，以至于其内部的文字行宽太宽，造成阅读困难。

（2）我们制作的每一个圆角框都是使用前面介绍的方法制作的，由于用这种方法制作的圆角框的最宽宽度有限制，因此如果超过此限度就会出现裂缝，如图14.2所示。

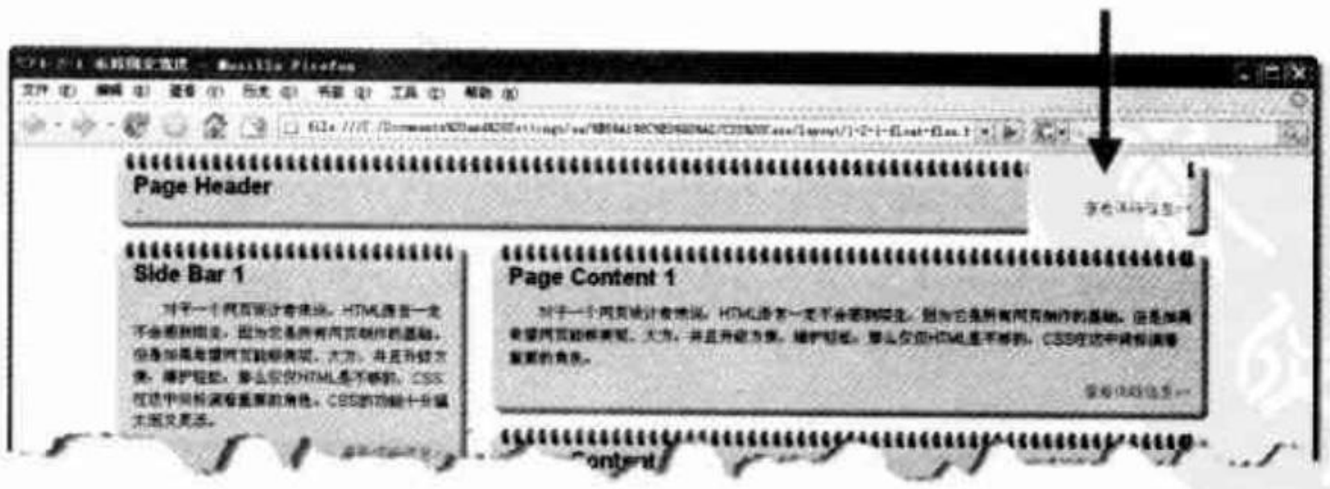


图14.2　宽度过大出现裂缝

针对上述第2点，解决的办法是，如果确实希望某一个分栏要这么宽，就修改背景图片。只需要修改5个图像中的left-top.gif，使它的覆盖范围更大就可以了。

如果并不需要这么宽，就可以对最大宽度进行限制。也就是说，当浏览器口超过一定宽度时，即使变得再宽，其内容也不再继续扩展。这需要用到前面制作“魔术布局”时介绍的max-width属性。同理，如果一个分栏过窄，视觉效果也会不好，因此也可以通过min-width属性限制最窄宽度。效果如图14.3所示。代码如下。

```
#header,#pagefooter,#container{
  margin:0 auto;
  width:85%;
  min-width:500px;
  max-width:800px;
  }
```

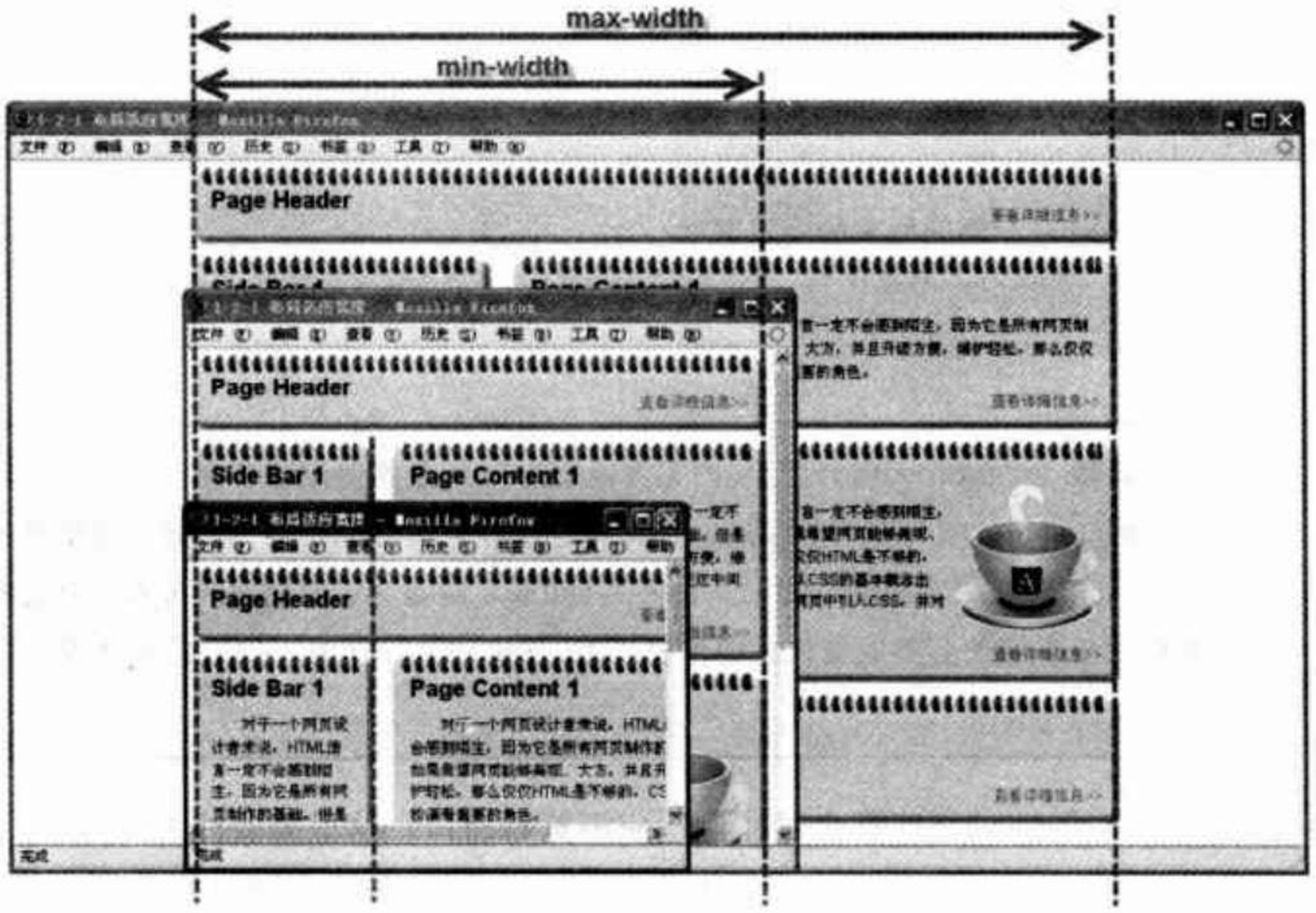


图14.3 设置宽度限制

前面在讲解“魔术布局”的时候提到过，这个方法存在一个问题，即IE 6不支持min-width和max-width这两个属性。同时给出了解决方法，即使用一个JavaScript小程序即可。在HTML的head部分加入如下语句，并将minmax.js文件保存到和页面同一个文件夹中即可。

```
<!--[if lte IE 6]>
    <script type="text/javascript" src="minmax.js"></script>
<![endif]-->
```

14.1.2 “1-2-1”单列变宽布局

在上面的例子中，当宽度变化时，各个部分的宽度都会变化，且它们之间的比例保持不变。实际上，只有单列宽度变化，而其他保持固定的布局可能会更实用。一般在存在多个列

的页面中，通常比较宽的一个列是用来放置内容的，而窄列放置链接、导航等内容，这些内容一般宽度是固定的，不需要扩大。因此如果能把内容列设置为可以变化，而其他列固定，会是一个很好的方式。

例如在图14.4中，右侧的Side Bar的宽度固定，当总宽度变化时，Page Content部分就会自动变化。

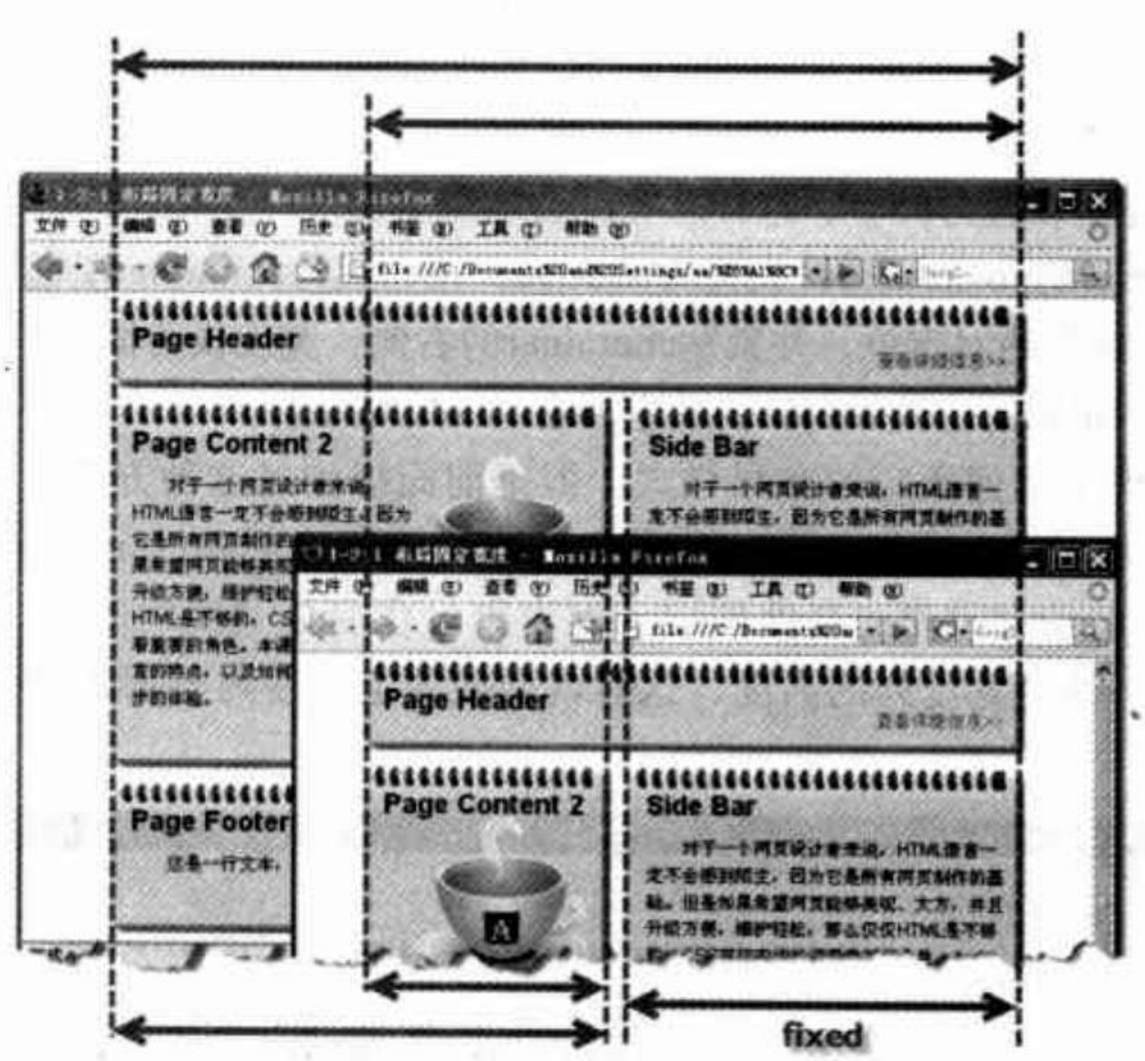


图14.4　“1-2-1”单列变宽布局

如果仍然使用简单的浮动布局，是无法实现这个效果的，如果把某一列的宽度设置为固定值，例如200像素，那么另一列（即活动列）的宽度就无法设置了。因为总宽度未知，活动列的宽度也无法确定，那么怎么解决呢？

这里仍然给出两种方法，首先介绍比较容易理解的“绝对定位”法，然后再对“浮动”法进行改造。

1. “绝对定位”法

在前面讲解固定的“1-2-1”布局时，我们除了使用浮动之外，还使用绝对定位实现过的“1-2-1”布局，现在就以该方案为基础来实现单列适应宽度的制作方法，代码如下。

本案例的文件位于本书光盘的“第14章\1-2-1\1-(l+f)-1-absolute.htm”。

```
#header,#pagefooter,#container{
    margin:0 auto;
    width:85%;
    }
#container{
    position:relative;
}
#side{
    position:absolute;
```

```
        top:0;
        right:0;
        width:300px;
        }
#content{
        margin:0 300px 0 0;
    }
```

对上面的代码原理进行分析如下。

（1）总宽度还是设置为85%，这样总宽度会随浏览器窗口变化。

（2）将container的position属性设置为relative，使它成为container里面的列的定位基准。

（3）使side列成为绝对定位，并紧贴container的右侧，宽度设为固定值300像素。

（4）设置content列的右侧margin，使它不会与side列重叠。

这样，就实现了单列固定的布局样式。但是前面提到过这种方法有一个固有的缺陷，也就是绝对定位的列将脱离标准流，从而它的高度将不会影响container的高度。这样页脚部分的位置只由content列的高度确定，而当窗口在变化宽度的时候，有可能会使固定宽度列的高度大于活动宽度列的高度，这时就会使固定宽度列与页脚部分重叠，如图14.5所示。

图14.5 出现重叠现象

因此，使用这种方法的时候，要注意保证变宽度列的高度是最高的，就不会发生重叠的现象了。由于HTML代码没有变化，因此这里就不再罗列HTML代码了。

2. “改进浮动”法

现在考虑，使用浮动的方法，实现的困难在哪里。核心问题就是浮动列的宽度应该等于“100%-300px”，而CSS显然不支持这种带有加减法运算的宽度表达方法。但是通过margin可以变通地实现这个宽度。

实现的原理如图14.6所示。在content的外面再套一个div（图中的contentWrap），使它的宽度为100%，也就是等于container的宽度。然后通过将左侧的margin设置为负的300像素，就使它向左平移了300像素。再将content的左侧margin设置为正的300像素，就实现了“100%-300px”这个本来无法表达的宽度。

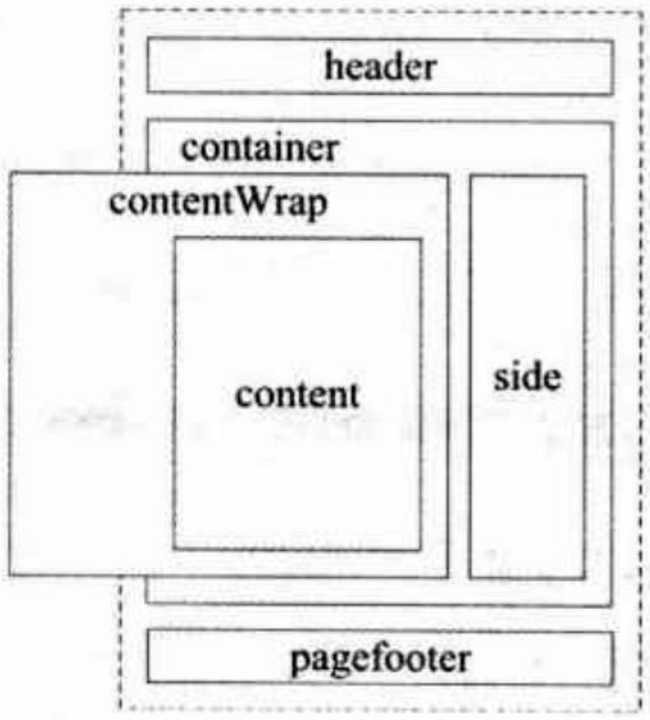


图14.6　结构示意图

CSS样式代码如下，本案例的文件位于本书光盘的“第14章\1-2-1\1-(1+f)-1-float.htm”。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
  }
#contentWrap{
  margin-left:-300px;
  float:left;
  width:100%;
  }
#content{
  margin-left:300px;
  }
#side{
  float:right;
  width:300px;
  }
#pagefooter{
  clear:both;
  }
```

注意，最核心的一点是在活动宽度列（即这里的content）外面又套了一层div，其id设置为contentWrap，中文的意思是content的“包装”，即它把content包裹起来。

contentWrap的宽度设置为100%宽度，同时将右侧的margin设置为“-300px”。注意这里是负值，即向左平移了300像素，并设置为向左浮动。content在它的里面，以标准流方式存在，将它的左侧margin设置为300像素，这样就可以保证里面的内容不会溢出到布局的外面，效果如图14.7所示。

这种方法的本质就是实现了content列的“100%-300像素”的宽度，确实非常巧妙。

这种方法的最大好处就是可以不用考虑各列的高度，通过设置页脚的clear属性，就可以保证不会发生重叠的现象。代码如下。

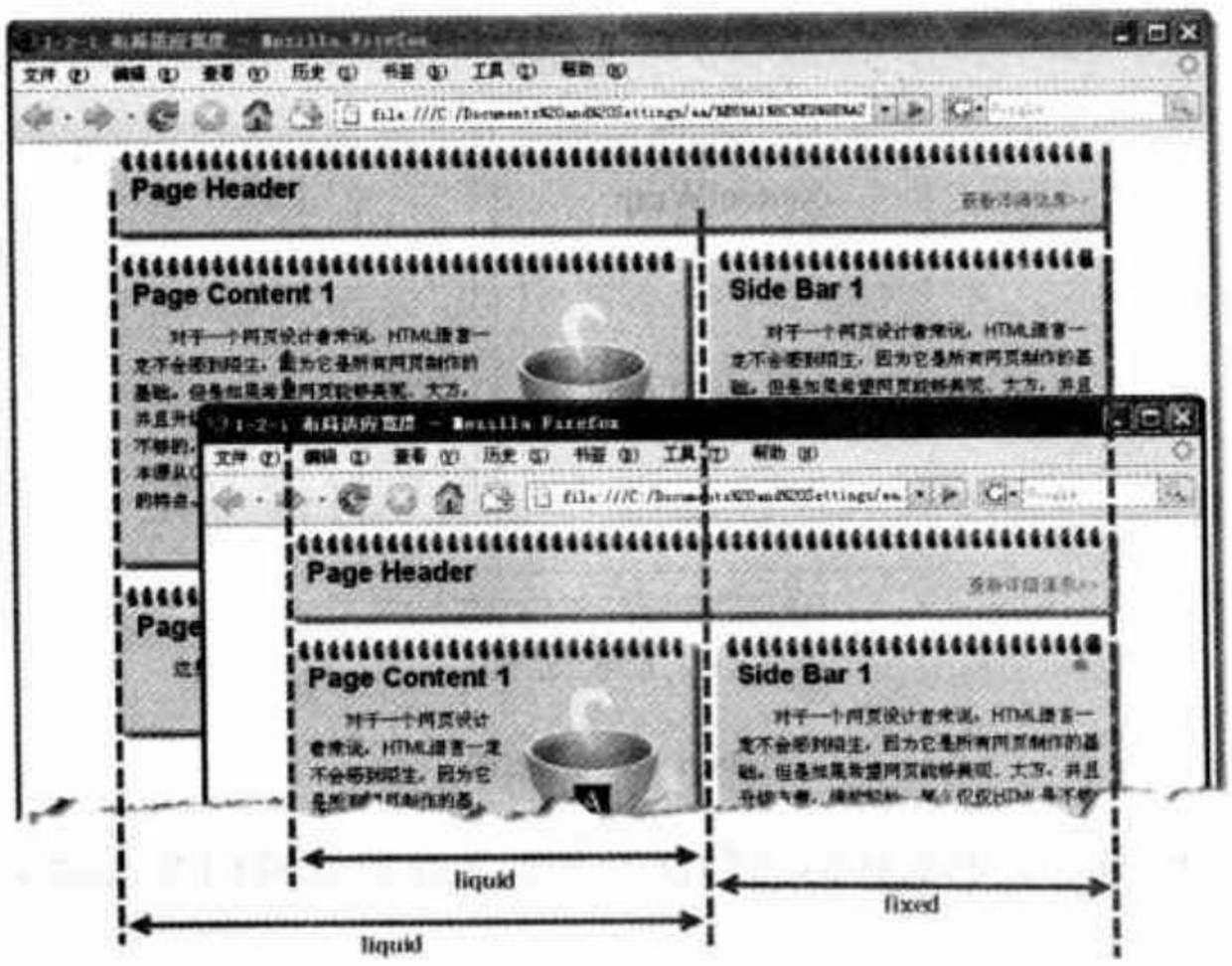


图14.7 单列固定的变宽布局

```
<body>
<div id="header">
  <div class="rounded">
……这里省略固定结构的内容代码……
  </div>
</div>
<div id="container">
  <div id="contentWrap">
        <div id="content">
              <div class="rounded">
                    ……这里省略固定结构的内容代码……
              </div>
        </div>
  </div>
  <div id="side">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
  </div>
</div>
<div id="pagefooter">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
</body>
```

CSS布局中的很多方法都是一些非常杰出的设计师的研究成果，正是他们不断地探索，才想出了很多非常好的方法，这里列举几篇非常重要的文章题目，这些设计师和他们的文章都极大地影响了Web设计和开发技术的发展。

- Ryan Brill：《Negative margins technique》
- Alex Robinson：《Any Order Columns》
- Eric Meyer：《Multi-unit Any Order Colums》

如果读者有兴趣，可以到互联网上找到这些文章，仔细研究。

前面介绍了按比例的宽度适应方法，以及单列宽度适应的制作方法。它们都是基于“1-2-1”布局来做的，制作3列布局或者更为复杂布局的页面的方法也是一样的。

14.2 “1-3-1”宽度适应布局

“1-3-1”布局可以产生很多不同的变化方式，如下：

- 三列都按比例来适应宽度；
- 一列固定，其他两列按比例适应宽度；
- 两列固定，其他一列适应宽度。

对于后两种情况，又可以根据特殊的一列与另外两列的不同位置，产生出若干种变化。这就像武侠小说中的武功，武林秘籍中的招数总是有限的，而实战中的变化则是无穷的，关键在于是否真正把其中的原理吃透了。最高的武功是所谓“大象无形”的境界，招数已经不重要了。

14.2.1 “1-3-1”三列宽度等比例布局

对于“1-3-1”布局的第一种情况，即三列按固定比例伸缩适应总宽度，和前面介绍的“1-2-1”的布局完全一样，只要分配好每一列的百分比就可以了。这里就不再介绍具体的制作过程了。

14.2.2 “1-3-1”单侧列宽度固定的变宽布局

对于一列固定、其他两列按比例适应宽度的情况，如果这个固定的列在左边或右边，那么只需要在两个变宽列的外面套一个div，并且这个div宽度是变宽的。它与旁边的固定宽度列构成了一个单列固定的“1-2-1”布局，就可以使用“绝对定位”的方法或者“改进浮动”法进行布局，然后再将变宽列中的两个变宽列按比例并排，就很容易实现了。

绝对定位法的制作过程就不再介绍了，这里仅给出使用浮动法的制作过程。假设现在希望side列宽度固定为200像素，而navi列和content列按照2：3的比例分配剩下的宽度。

请读者思考，如果按照图14.8所示的结构建立HTML结构，能否实现所需的效果？

答案是否定的。wrap这个容器内部如果只有一个活动列，就像前面的“1-2-1”布局那样，这个活动列以标准流方式放置，它的宽度是自然形成的，这样显示效果是没有问题的。而当wrap容器中有两个浮动的活动列时，就需要分别设置宽度，分别为40%和60%（为了避

免四舍五入误差，这里设置59.9%）。请特别注意，这时wrap列的宽度等于container的宽度，因此这里的40%并不是总宽度减去side的宽度以后的40%，而是总宽度的40%，这显然是不对的。

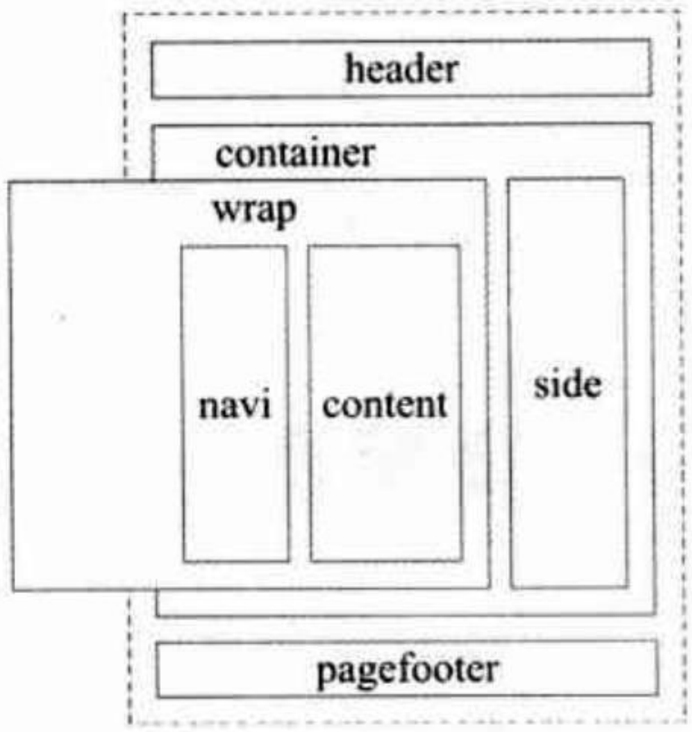


图14.8 结构示意图

解决的方法就是在容器里面再套一个div，即由原来的一个wrap变为两层，分别叫做outerWrap和innerWrap，结构如图14.9所示。

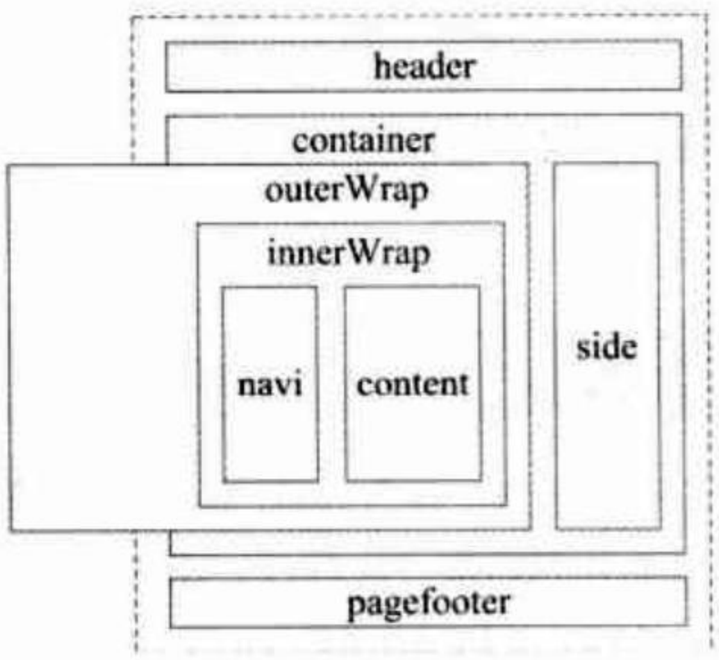


图14.9 修正后的结构示意图

这样，outerWrap就相当于上面错误方法中的wrap容器。新增加的innerWrap是以标准流方式存在的，宽度会自然伸展，由于设置了200像素的左侧margin，因此它的宽度就是总宽度减去200像素了，这样，innerWrap里面的navi和content就会都以这个新宽度为宽度基准。

CSS代码如下所示。本案例的文件位于本书光盘的“第14章\1-3-1\一个固定列\1-(1-1-f)-1.htm”。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
  }
```

```
#outerWrap{
  float:left;
  width:100%;
  margin-left:-200px;
  }
#innerWrap{
  margin-left:200px;
  }
#navi{
  float:left;
  width:40%;
  }
#content{
  float:right;
  width:59.5%;
  }
#side{
  float:right;
  width:200px;
  }
#pagefooter{
  clear:both;
  }
```

HTML部分的关键代码如下。

```
<body>
<div id="header">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
<div id="container">
  <div id="outerWrap">
        <div id="innerWrap">
              <div id="navi">
                    <div class="rounded">
                          ……这里省略固定结构的内容代码……
                    </div>
              </div>
              <div id="content">
                    <div class="rounded">
                          ……这里省略固定结构的内容代码……
                    </div>
              </div>
        </div>
  </div>
  <div id="side">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
  </div>
```

```
</div>
<div id="pagefooter">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
</body>
```

14.2.3 “1-3-1”中间列宽度固定的变宽布局

本小节将要介绍的布局形式是，固定列被放在中间，它的左右各有一列，并按比例适应总宽度。这是一种很少见的布局形式（最常见的是两侧的列固定宽度，中间列变化宽度）。在中文网站中几乎看不到对这种方式的讨论，英文网站中对这种布局的讨论也大都是作为对CSS的一种练习和研究。这里可以假设要做一个博客网站的页面，可设计为中间列放置链接、导航等内容，左边列放置各篇文章，右边列放置照片。当浏览器窗口变化时，左右两列会自动变化。这个博客页面的结构是很不错的。

如果读者已经充分理解了前面介绍的“改进浮动”法制作单列宽度固定的“1-2-1”布局，就可以把“负margin”的思路继续深化，实现这种不多见的布局。

假设，现在希望页面中间列的宽度是300像素，两边列等宽（不等宽的道理是一样的），即总宽度减去300像素后剩余宽度的50%。此时制作的关键是如何实现“（100%-300px）/2”的宽度。

注意 这里所讲的案例是基于荷兰设计师Gerben提出来的方法实现的。该设计师的网站的网址是http://algemeenbekend.nl/misc/challenge_gerben_v2.html。遗憾的是他的网站不是英文的。

下面就来讲解“固定单列居中，两侧列适应”的布局方法。

这里以固定的“1-3-1”布局为基础。现在需要在navi和side两个div外面分别套一层div，把它们“包裹”起来，如图14.10所示。

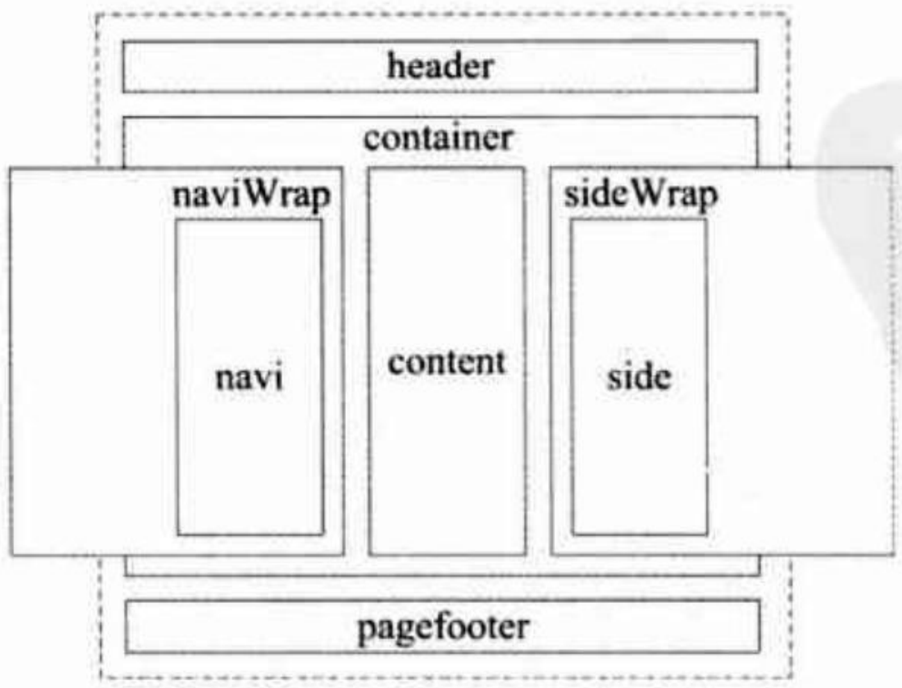


图14.10 结构示意图

在“改进浮动”法中已经了解这样做的原因，就是依靠嵌套的两个div，实现相对宽度和绝对宽度的结合。

首先设置好HTML结构，代码如下。本案例的文件位于本书光盘的“第14章\1-3-1\一个固定列\1-(l-f-l)-1.htm”。

```
<body>
<div id="header">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
<div id="container">
  <div id="naviWrap">
        <div id="navi">
              <div class="rounded">
                    ……这里省略固定结构的内容代码……
              </div>
        </div>
  </div>
  <div id="content">
        <div class="rounded">
              ……这里省略固定结构的内容代码……
        </div>
  </div>
  <div id="sideWrap">
        <div id="side">
              <div class="rounded">
                    ……这里省略固定结构的内容代码……
              </div>
        </div>
  </div>
</div>
<div id="pagefooter">
  <div class="rounded">
        ……这里省略固定结构的内容代码……
  </div>
</div>
</body>
```

设置好HTML代码之后，CSS的相关部分代码如下所示。

```
#header,#pagefooter,#container{
  margin:0 auto;
  width:85%;
  }
#naviWrap{
  width:50%;
  float:left;
  margin-left:-150px;
}
#navi{
  margin-left:150px;
```

```
  }
#content{
  float:left;
  width:300px;
  }
#sideWrap{
  width:49.9%;
  float:right;
  margin-right:-150px;
  }
#side{
  margin-right:150px;
  }
#pagefooter{
  clear:both;
  }
```

将左侧的naviWrap设置为50%宽度，向左浮动，并通过将左侧margin设置为-150像素，向左平移了150像素。然后在里面的navi中，左侧margin设置为-150像素，补偿回来这150像素。

接着，将content设置为固定宽度，先做浮动，这样就紧贴着navi的右边界。

最后将sideWrap做与navi部分相似的处理，设置为50%宽度，向左浮动。这时本来宽度已经超过100%，会被挤到下一行，但是将右侧margin设置为-150像素后，就不会超过总宽度了。

注意 在实际代码中，并不是将两个活动列宽度都设置为50%，而是将其中一个设置为49.9%。这是为了避免浏览器在计算数值时因四舍五入而导致总宽度大于100%，因此稍微窄一点点，就可保证最右边的列不会被挤到下一行。

这时的效果如图14.11所示，可以看到，浏览器宽度变化后，中间列的宽度保持不变，而两侧的列宽度发生了变化。

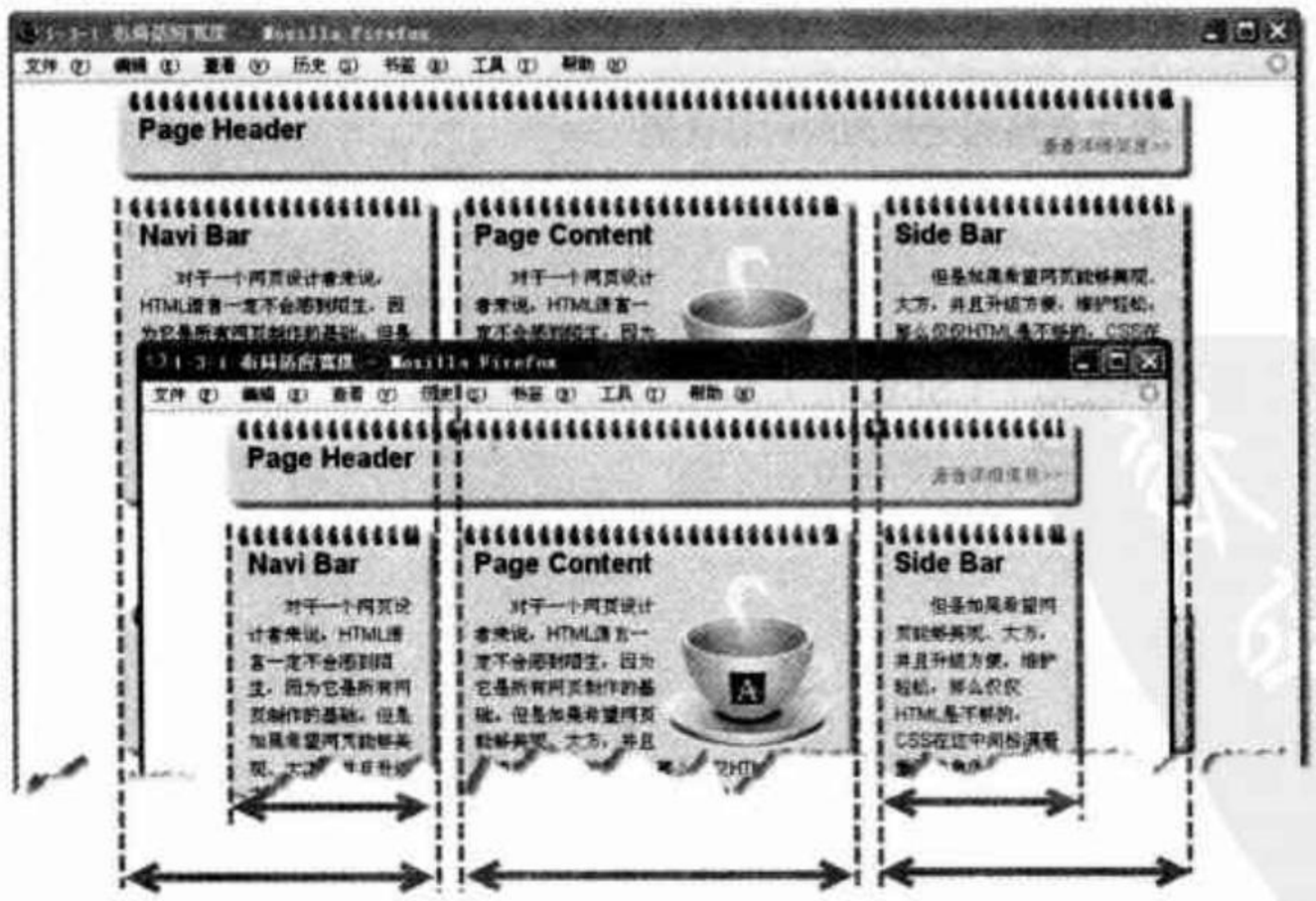


图14.11 中间列宽度固定的布局效果

14.2.4 进一步的思考

在使用“改进浮动”法制作中间列固定和侧列固定这两个案例的时候，使用了不同的思路。二者之中，哪一个更具有通用性呢？显然是后者，因为使用这个方法同样可以实现固定列在中间的布局，而用前者的方法是无法实现单侧列固定宽度的。

使用后面介绍的这种方法不但可以实现左中右3列中任意列固定，其余两列按比例分配宽度，而且可以仅通过CSS任意调换3列的位置。这3列都是并列关系，因此可以在只进行比较小的改动的情况下实现HTML中的各列任意排序。

注意　这里提出了一个新问题，即“任意列排序”。由于这个问题过于深入，本书篇幅有限，因此这里仅提出这个问题，有兴趣的读者可以自行深入研究。

假设有一个3列布局的页面，在HTML中一定会有依次排列的3个“ …… ”段。如果通过CSS设置可以实现在HTML中，无论这些“ …… ”的顺序如何，都可以得到希望的显示顺序，那么这样的排版方法将会有很大优势。

（1）可以根据各div的内容来组织HTML结构，而不是根据页面的表现形式来确定顺序，在更大的程度上实现了内容与形式的分离。

（2）对于访问者来说，即使他的浏览器不支持CSS，也依然可以按照符合内容逻辑的顺序浏览页面。

（3）对于设计师来说，可以灵活地调整各列的顺序，而不必修改HTML结构。

（4）对于搜索引擎来说，它们通常对页面中越靠前的内容越重视，因此如果实现了内容顺序不需要考虑页面表现时的顺序，则可以更有利于搜索引擎的排名。

本书关于“任意列排序”的介绍只能到此为止了，建议读者仔细研究Alex Robinson的文章《Any Order Columns》，它详细地介绍了任意列排序的意义、原理和方法，请参见网页http://positioniseverything.net/articles/onetruelayout/anyorder。

14.2.5 “1-3-1”双侧列宽度固定的变宽布局

对于三列布局，一种很实用的布局是3列中的左右两列宽度固定，中间列宽度自适应。通过前面的学习已经知道，这个布局同样可以有两种思路来实现，一种是用绝对定位，另一种是完全浮动实现。

1. “绝对定位”法

首先介绍是绝对定位的方法，代码如下。本案例的文件位于本书光盘的“第14章\1-3-1\两个固定列\1-(f-l-f)-1-absolute.htm”。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
```

```
  }
#container{
  position:relative;
  }
#navi{
  position:absolute;
  top:0;
  left:0;
  width:150px;
  }
#side{
  position:absolute;
  top:0;
  right:0;
  width:250px;
  }
#content{
  margin:0 250px 0 150px;
  }
```

这段代码中，把container的position属性设置为relative，使它成为它的下级元素的绝对定位基准。然后，将左边的navi列绝对定位，并设置为150像素宽，紧贴左侧；右边的side列也是绝对定位，250像素宽，紧贴右侧。这样，中间的content列没有设置任何与定位相关的属性，因此它仍然在标准流中，将它的左右margin设置为两个绝对定位列的宽度，正好让出它们的位置，这样就实现了三者的并列放置。实现的效果如图14.12所示。

当然，这种方法制作的三列布局无法避免“绝对定位”造成的固有缺陷，即页脚永远紧贴着中间的content列，而不管左右两侧列的高度，并且当中间列的高度小于两侧列中的一个或两个时，会造成重叠的现象。

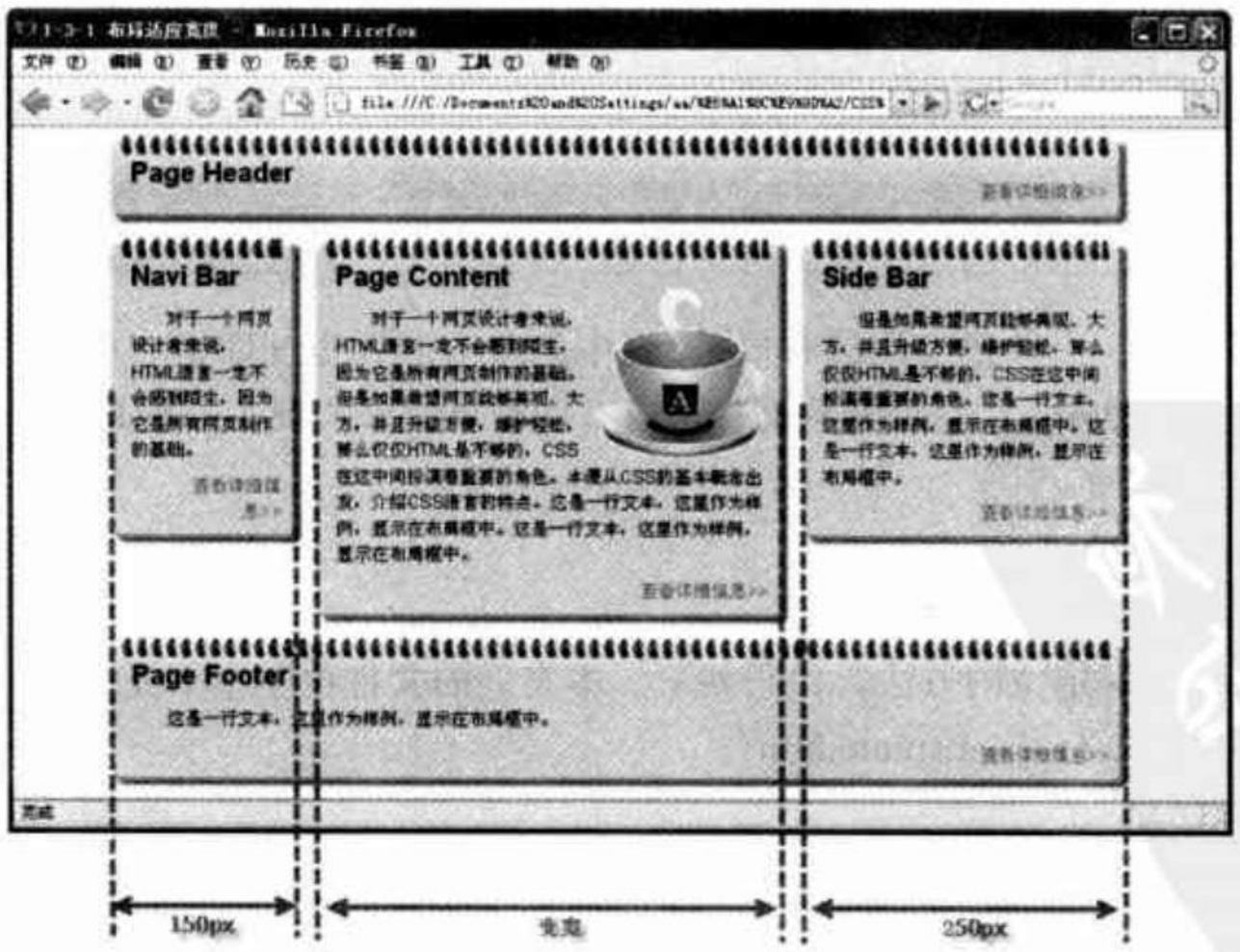


图14.12 中间列变宽的布局效果

2. 二次“改进浮动”法

为了避免使用绝对定位带来的缺陷，可以使用类似前面用过的“改进浮动法”。利用margin的负值来实现3列都使用浮动的方法。具体的思路就是把3列的布局看作是嵌套的两列布局，如图14.13所示。

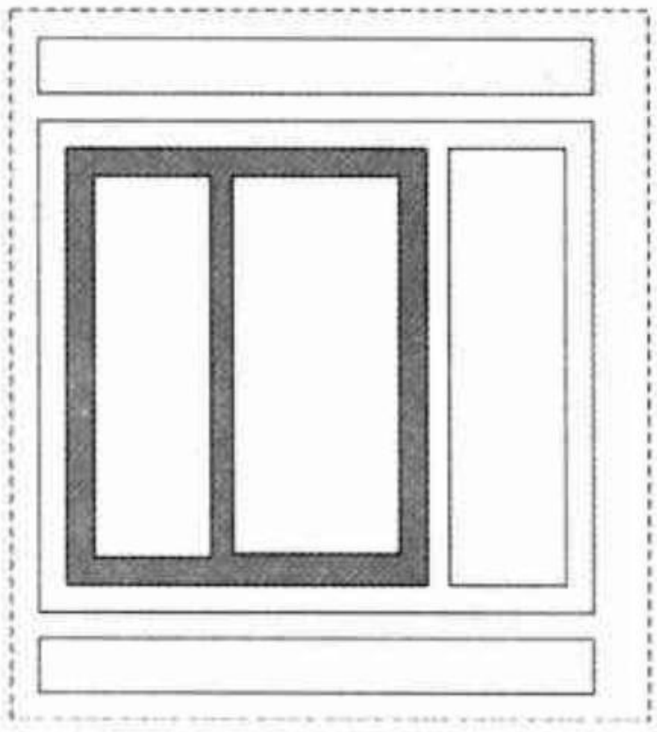


图14.13　结构示意图

先把左边和中间两列看作一组（图中灰色背景部分），作为一个活动列，而右边的一列作为固定列，使用前面的“改进浮动”法就可以实现。然后，再把两列（灰色背景部分）各自当作独立的列，左侧列为固定列，再次使用“改进浮动”法，就可以最终完成整个布局。

需要注意的是，使用这种方法需要增加比较多的辅助div，结构会变得非常复杂，如图14.14所示。

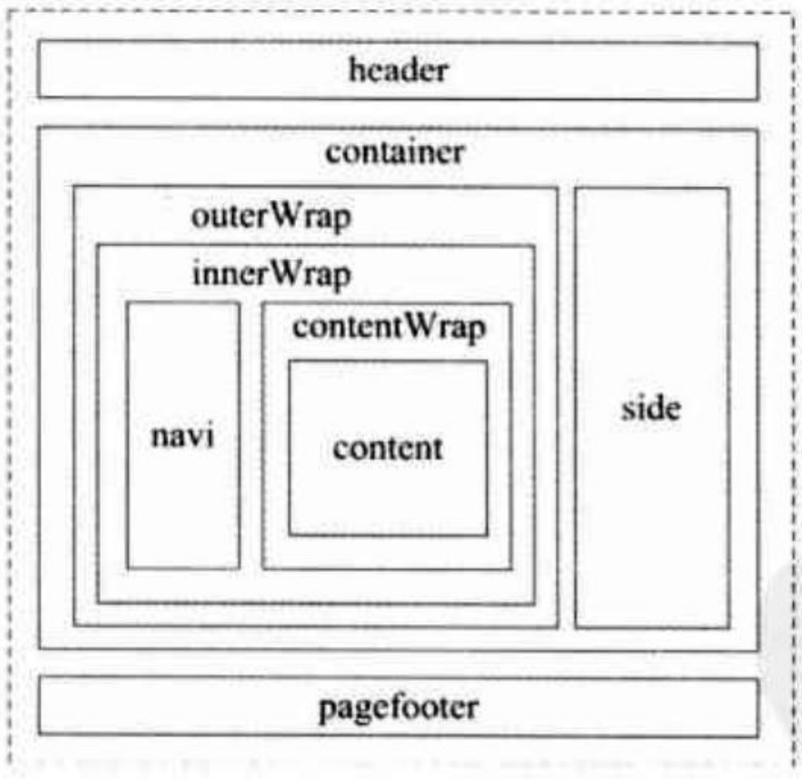


图14.14　详细的结构示意图

使用“改进浮动”法时，每实现一个活动列都需要增加额外的辅助div。从图中可以看出，这里的思路是，在内层，为了使navi固定，content变宽，在二者外面套了一个“innerWrap” div；为了在innerWrap中使content能够变宽，在content外面又套了也contentWrap；同理，为了使innerWrap能够变宽，又为它套了一个outerWrap，从而使结构变得复杂。但实际上原理还是相同的。

CSS部分的代码如下所示，本案例的文件位于本书光盘的“第14章\1-3-1\两个固定列\1-(f-l-f)-1-float.htm”。

注意 读者如果没有亲自调试过前面的“1-2-1”单列固定变宽布局，学习本例可能会比较吃力。本案例的本质就是两次使用“改进浮动”法，因此请读者务必深刻理解“改进浮动”法的本质原理，这样就可以保持条理清晰，逻辑明确了。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
  }
#side{
  width:200px;
  float:right;
  }
#outerWrap{
  width:100%;
  float:left;
  margin-left:-200px;
}
#innerWrap{
  margin-left:200px;
  }
#navi{
  width:150px;
  float:left;
}
#contentWrap{
  width:100%;
  float:right;
  margin-right:-150px;
}
#content{
  margin-right:150px;
  }
#pagefooter{
  clear:both;
}
```

请读者自己分析这段代码，最核心的要点就是深刻理解改进浮动法的原理。

14.2.6 “1-3-1”中列和侧列宽度固定的变宽布局

这节介绍的布局方式即中间列和它一侧的列是固定宽度，另一侧列宽度自适应。很显然这种布局就很简单了，同样可以使用绝对定位法和改进浮动法来实现。

1. “绝对定位”法

如果使用绝对定位的方法，代码如下。本案例的文件位于本书光盘的“第14章\1-3-1\两个固定列\1-(f-f-l)-1-absolute.htm”。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
  }
#container{
  position:relative;
  }
#navi{
  position:absolute;
  top:0;
  left:0;
  width:150px;
  }
#content {
  position:absolute;
  top:0;
  left:150px;
  width:250px;
  }
#side {
  margin:0 0 0 400px;
}
```

对上面的代码原理进行分析如下。

（1）总宽度还是设置为85%，这样总宽度会随浏览器窗口变化。

（2）然后将container的position属性设置为relative，使它成为container里面的列的定位基准。

（3）使navi列成为绝对定位，并紧贴container的左侧，宽度设为固定值150像素。

（4）设置content列也成为绝对定位，也以左侧对齐，将left属性值设置为150像素，即正好等于它左边的navi列的宽度，使它不会与side列重叠。设置content的固定宽度值250像素。

（5）最后设置右边的side列，它使用标准流方式，使它左侧的margin正好等于content和navi两列的宽度总合。

这样，就实现了左侧和中间列固定且右侧自动适应宽度的布局样式。但是这种方法依然避免不了它固有的缺陷，即当窗口在变化宽度的时候，有可能会使固定宽度列的高度大于活动宽度列的高度，导致固定宽度列与页脚部分重叠。

2. “改进浮动”法

使用改进浮动法也可以实现这种布局，并且避免绝对定位方法带来的缺陷。缺点则是要增加辅助div的使用，使结构变复杂。使用改进浮动法时，由于两个固定宽度列是相邻的，因此就不用使用两次改进浮动法了，只需要一次就可以做到。

假设仍然希望左侧的navi和content列的宽度分别固定为150像素和250像素，右侧的side列宽度变化。那么side列的宽度就等于“100%-150px-250px”。因此根据改进浮动法，在side列的外面再套一个sideWrap列，使sideWrap的宽度为100%，并通过设置负的margin，使它向右平移400像素（即150像素与250像素之和）。然后再对side列设置正的margin，限制右边界，这样就可以实现希望的效果了。

相关的代码如下。本案例的文件位于本书光盘的“第14章\1-3-1\两个固定列\1-(f-f-l)-1-float.htm”。

```
#header,
#pagefooter,
#container{
  margin:0 auto;
  width:85%;
  }
#navi{
  float:left;
  width:150px;
  }
#content{
  float:left;
  width:250px;
  }
#sideWrap{
  float:right;
  width:100%;
  margin-right:-400px;
  }
#side{
  margin-right:400px;
  }
#pagefooter{
  clear:both;
  }
```

14.3 变宽布局方法总结

实际上，关于三列布局的方法还有很多，各有优缺点，适用的范围也各不相同。如果读者有兴趣深入研究，可以参考下面网址http://css-discuss.incutio.com/?page=ThreeColumnLayouts，里面列出了数十种不同的三列布局方法，并根据不同的标准进行分类，读者可以根据自己的需要寻找适合的方法。

如果对本章介绍的各种布局进行总结，可以得到下列3个结构图。

单列布局的结构如图14.15所示。

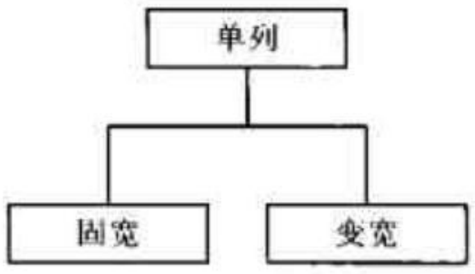


图14.15　单列布局的分类示意图

双列布局的结构如图14.16所示。

图14.16　双列布局的分类示意图

三列布局的结构如图14.17所示。

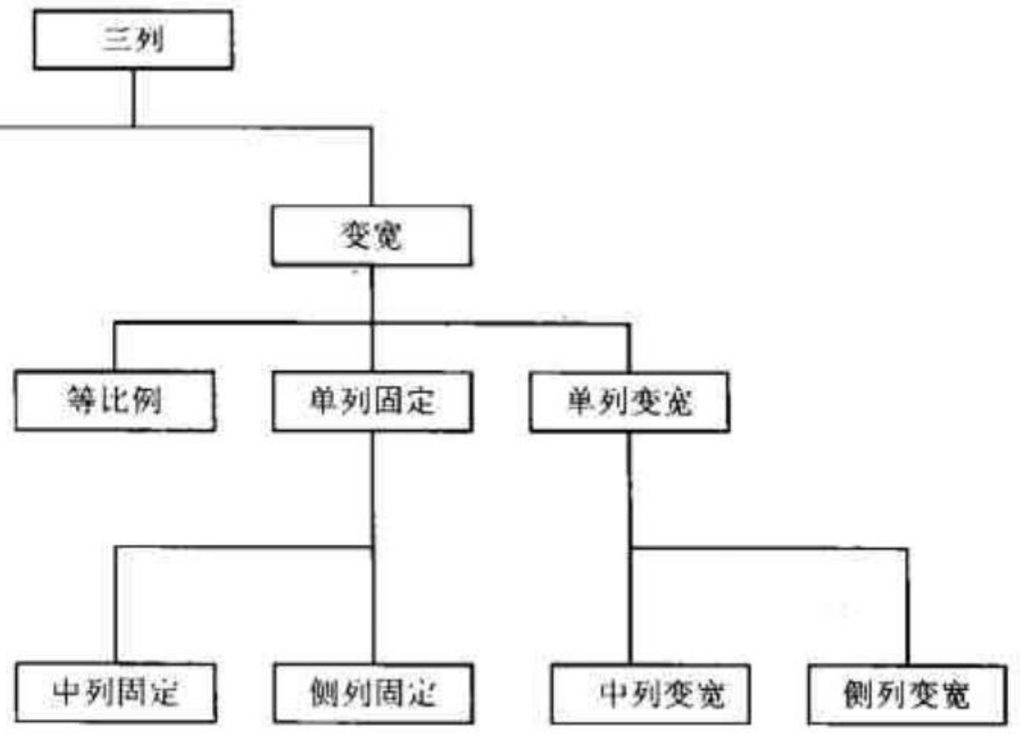


图14.17　三列布局的分类示意图

希望读者顺着这3个图的线路，回忆一下每种布局的实现方法。如果能够清楚地说清每种布局的方法，那么本章的学习就成功了。实际上，更多的分列或变化都可以看作是基本方法的重复使用或者嵌套使用，因此掌握了上面的这些布局形式，对于更多列的布局也就都可以解决了。

14.4 分列布局背景色问题

在前面的各种布局案例中，都是使用带有边框的圆角框实现的。可以发现，所有的例子

都没有设置背景色，但是在很多页面布局中，对各列的背景色是有要求的，例如希望每一列都有各自的背景色。

前面案例中每个布局模块都有非常清晰的边框，这种页面通常不设置背景色。还有很多页面分了若干列，每一列或列中的各个模块并没有边框，这种页面通常需要通过背景色来区分各个列。

下面就来对页面布局中的分栏背景色问题进行一些讲解。为了简化页面，我们首先制作一个如图14.18所示的页面。

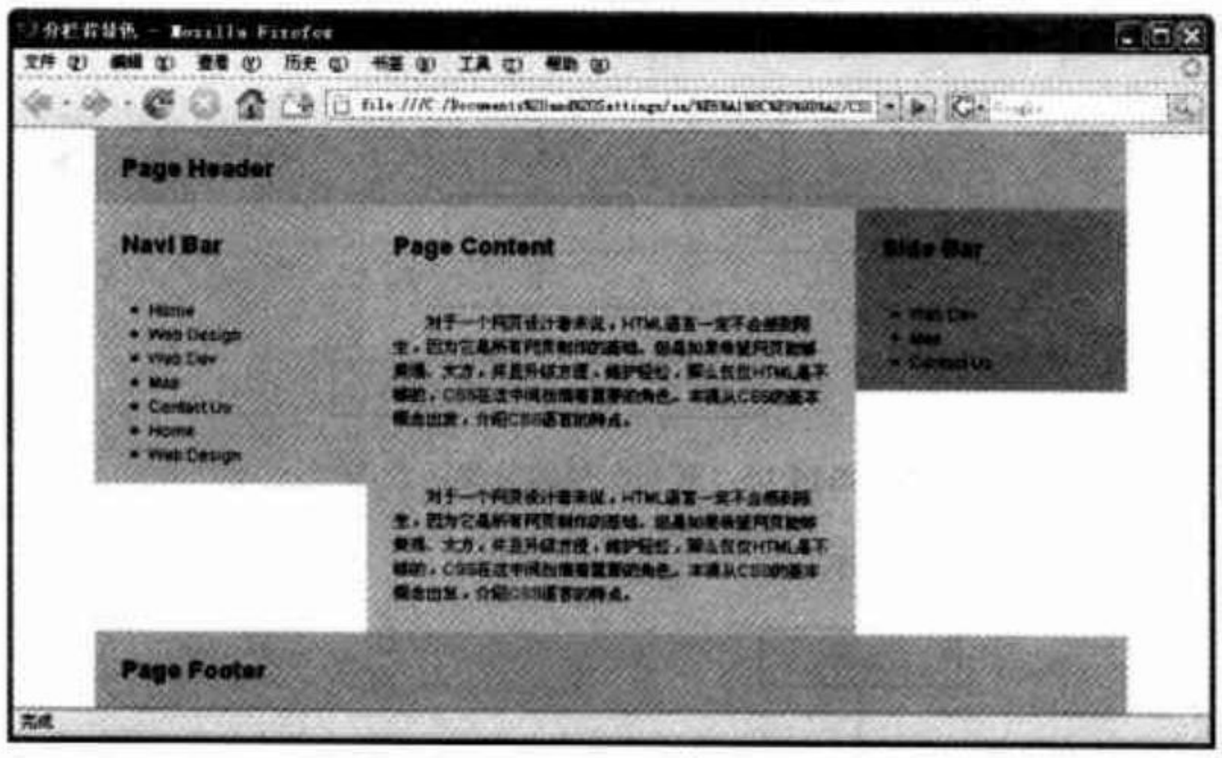


图14.18 基本的三列布局

这是一个很简单的“1-3-1”布局页面，通过前面的学习，相信读者都可以用各种办法制作出这个页面。本文件位于本书光盘的“第14章\1-3-1\背景色\basic.htm”。

14.4.1 固定宽度布局的列背景色设置

这里先假设它是固定宽度的，总宽度760像素，左右列各200像素，中间列360像素。使用绝对定位的方式布局。

本案例文件位于本书光盘的“第14章\1-3-1\背景色\fixed.htm”。

HTML部分代码如下：

```
<body>
<div id="header">
  <h2>Page Header</h2>
</div>
<div id="container">
  <div id="navi">
          <h2>Navi Bar</h2>
          <ul>
                <li>Home</li>
                ……省略其余列表项……
          </ul>
</div>
<div id="content">
          <h2>Page Content</h2>
```

```
        <p>                对于一个网页设计者来说，……省略其余文字……</p>
        ……省略其余文字段落……
</div>
<div id="side">
        <h2>Side Bar</h2>
        <ul>
                <li>Web Dev</li>
                ……省略其余列表项……
        </ul>
  </div>
</div>
<div id="footer">
  <h2>Page Footer</h2>
</div>
</body>
```

CSS样式代码如下：

```
body{
  font:12px/18px Arial;
  margin:0;
  }
#header,#footer {
  background:#99CCFF;
  width:760px;
  margin:0 auto;
  }
h2{
  margin:0;
  padding:20px;
  }
p{
  padding:20px;
  text-indent:2em;
  margin:0;
  }
#container {
  position: relative;
  width:760px;
  margin:0 auto;
  }
#navi {
  width: 200px;
  position: absolute;
  left: 0px;
  top: 0px;
  background:#99FFCC;
}
#content {
  margin-right: 200px;
  margin-left: 200px;
  background:#FFCC66;
```

```
  }
#side {
  width: 200px;
  position: absolute;
  right: 0px;
  top: 0px;
  background:#CC99FF;
  }
```

可以看到，各列的背景色只能覆盖到其内容的下端，而不能使每一列的背景色都一直扩展到最下端。这个要求在表格布局的方式中，是很容易实现的，而在CSS布局中，却不是这样。根本的原因在于，表格会自然地使各列等高，而每个div只负责自己的高度，根本不管它旁边的列有多高，要使并列的各列的高度相同是很困难的。

解决问题的思路之一是想办法使各列等高。有很多Web设计师从这个思路出发，并通过使用JavaScript配合，找到了解决方法。但是这里我们通过单纯的CSS来解决这个问题。

如果是各列固定宽度的布局方式，就很容易通过另一种思路来解决这个问题，即通过“背景图像”法。例如，在本例中，已经知道3列的宽度依次为200像素、360像素和200像素，就可以在Fireworks或者Photoshop等图像处理软件中制作一个760像素宽的图像，通过竖向平铺图像来产生各列的分隔效果。

例如，图14.19中显示的是一个760像素宽、10像素高的图像。

注意 图中的灰色边框是为了表示图像的范围，在制作的时候，不要加边框。

图14.19 背景图像

将上面代码中的3列div的背景色设置全部去掉，然后将container这个容器div的背景设置为“url(background-760.gif)”，即该图像文件的路径。这时在浏览器中的效果如图14.20所示。

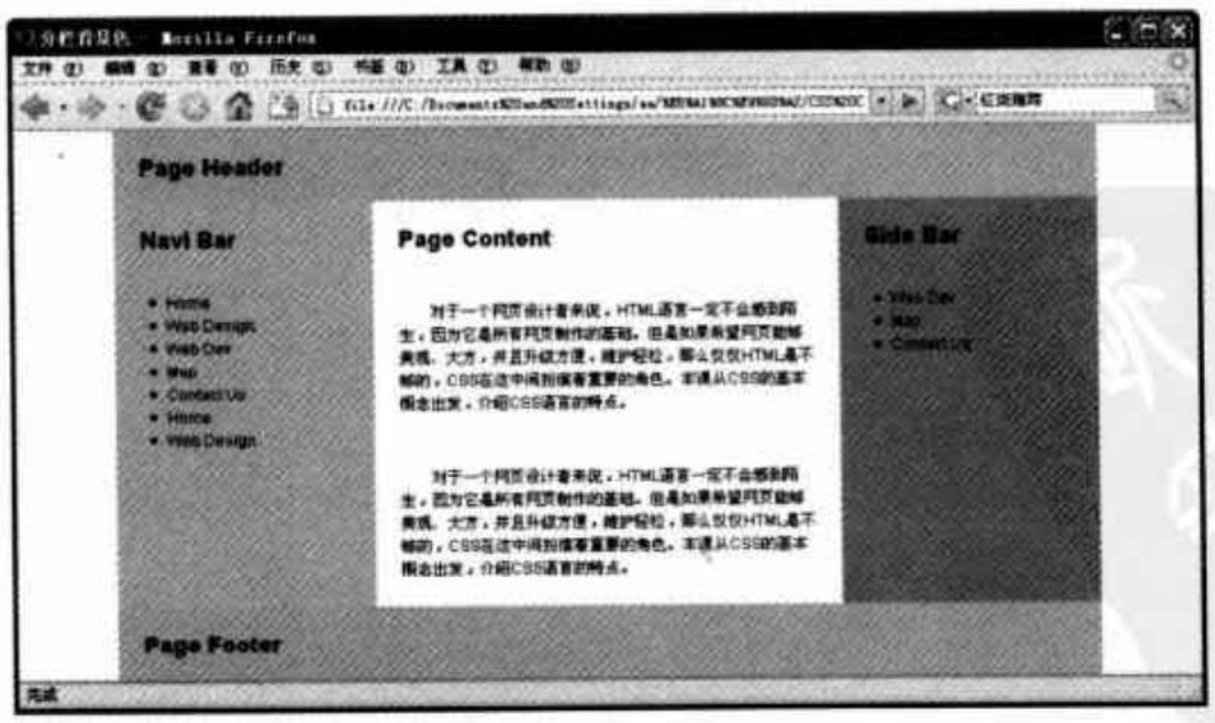


图14.20 使用了背景图像的分列效果

现在无论一列的高度是多少，背景色都可以一直贯穿到底。用这种办法还可以制作出一

些更精致的效果，例如为背景图像制作一些投影的效果，如图14.21所示。

图14.21　带阴影效果的背景图像

这时产生的效果如图14.22所示，页面的感觉一下精致了不少。

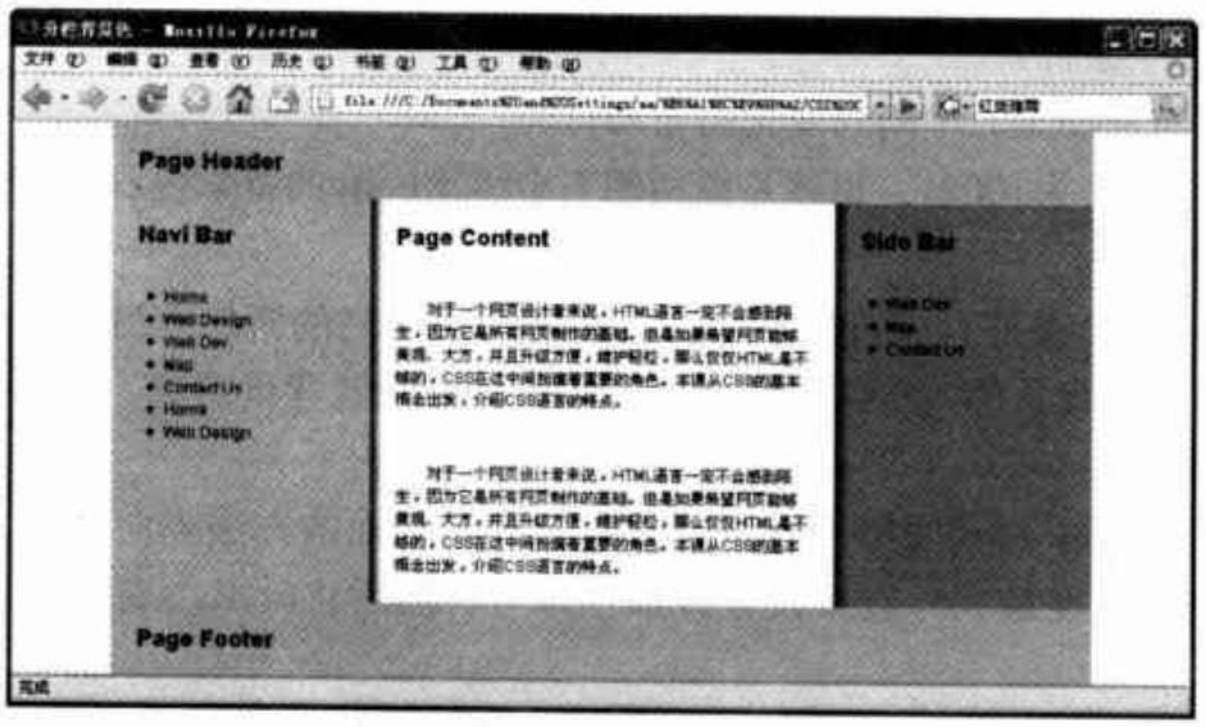


图14.22　使用带阴影的背景图像的页面效果

图14.23所示的是“CSS禅意花园”网站的第026号作品，是由设计师Radu Darvas设计的。可以看到，通过非常精致的设计，页面的分栏取得了出人意料的效果，在视觉上摆脱了固定的“框”通常成生的呆板的样式，而形成了重叠的效果。这种效果表现了光和影之间、形状和空间之间的相互影响，给人清新、明朗、积极的印象。

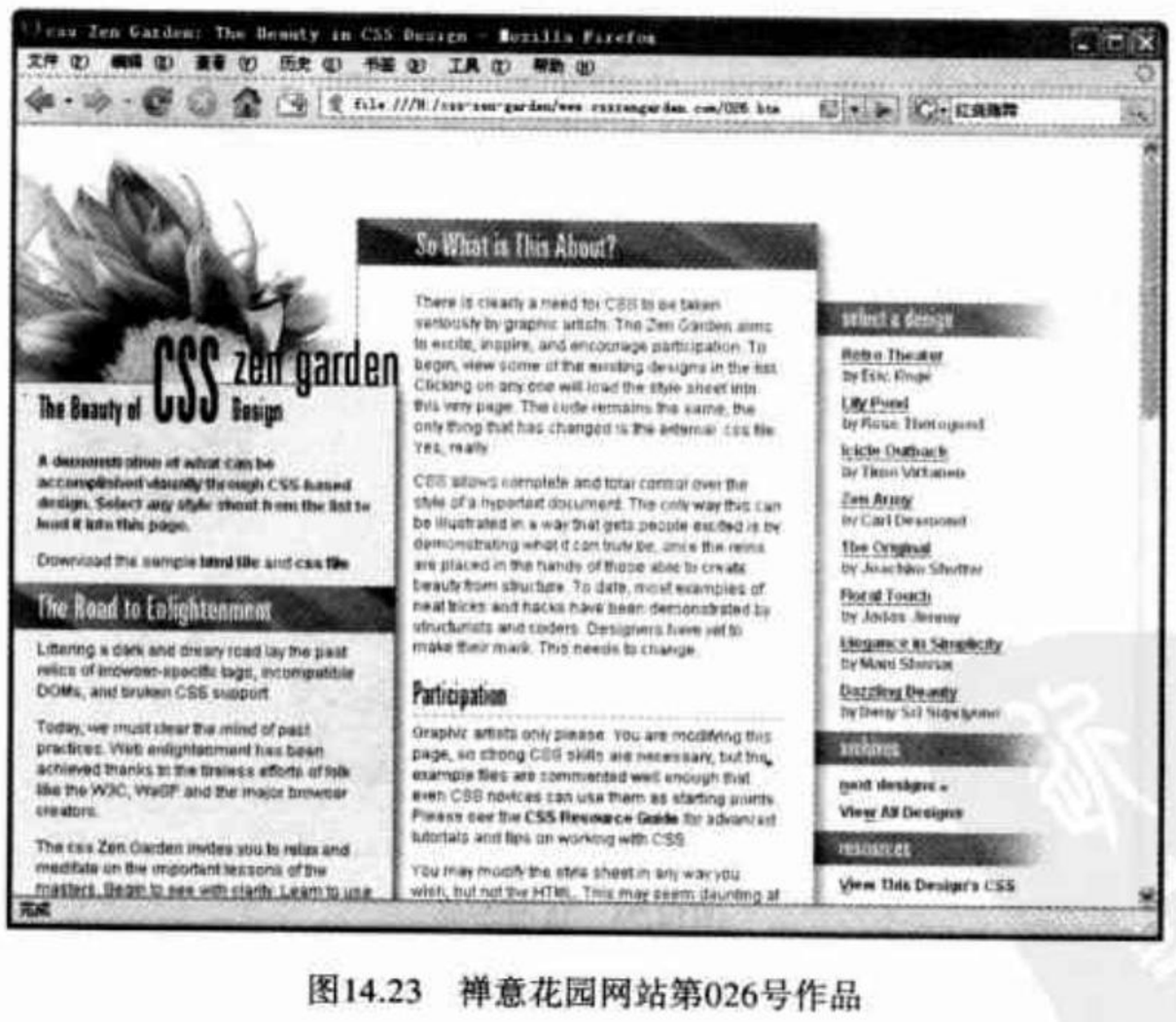


图14.23　禅意花园网站第026号作品

读者可以到http://www.csszengarden.com/026查看该网页。从这个例子可以看出，只有技术和艺术的完美结合，才能创造出真正打动人心的作品。

从技术角度说，上面介绍的方法适用于各列宽度固定的布局。无论是使用浮动方式布局，还是绝对定位方式布局，对于上面这种背景图像平铺的方法都是适用的。

14.4.2 特殊宽度变化布局的列背景色设置

在解决了固定宽度的分栏背景色问题之后，再来考虑宽度变化的布局分栏背景色问题。如果列宽不确定，就无法在图像处理软件中制作这个背景图，那么应该怎么办呢?

假设有如下3个条件：

（1）两侧列宽度固定，中间列变化的布局；

（2）3列的总宽度为100%，也就是说两侧不会露出body的背景色；

（3）中间列最高。

如果满足了这3个条件，就可以利用body来实现右侧栏的背景。另外，中间列的高度最高，可以设置自己的背景色，左侧可以使用contatiner来设置背景图像。

具体方法如下。

首先制作一个与左列宽度相同的背景图片，按照上面的方法竖向平铺，左列就设置好了。而中间列由于高度最大，直接设置背景色即可。然后将body的背景色设置为右栏的背景色。

例如，图14.24中所示的这个页面一共包括3个竖列和1个横行，整个页面被分为了4个部分，这4个部分各使用一种颜色作为背景色。可以看到，整个页面横向撑满这个页面，对于这样的页面，就可以使用上面介绍的这种方法来实现各栏的背景色。

图14.24 版面布局

14.4.3 单列宽度变化布局的列背景色设置

上面例子虽然实现了分栏的不同背景色，但是它的限制条件太多了。能否找更通用一些的方法呢?

仍然假设布局是中间活动，两侧列宽度固定的布局。由于container只能设置一个背景图像，因此可以在container里面再套一层div，这样两层容器就可以各设置一个背景图像，一个左对齐，一个右对齐，各自竖直方向平铺。由于左右两列都是固定宽度，因此所有图像的宽度分别等于左右两列的宽度就可以了。

假设将上面完全固定的布局改为：3列总宽度为浏览器窗口宽度的85%，左右列各200像素，中间列自适应。本案例文件位于本书光盘的“第14章\1-3-1\背景色\center-liquid.htm”。

代码稍作修改，header、footer和container的宽度改为85%，然后在container里面套一个innerContatiner，设置为：

```
#container {
  width:85%;
  margin:0 auto;
  background:url(background-right.gif) repeat-y top right;
  position: relative;
  }
#innerContainer {
  background:url(background-left.gif) repeat-y;
  }
```

这样效果如图14.25所示，注意contatiner和innerContainer的背景图像设置方法，右边的背景图像除了设置竖向平铺之外，还要确定左对齐还是右对齐。

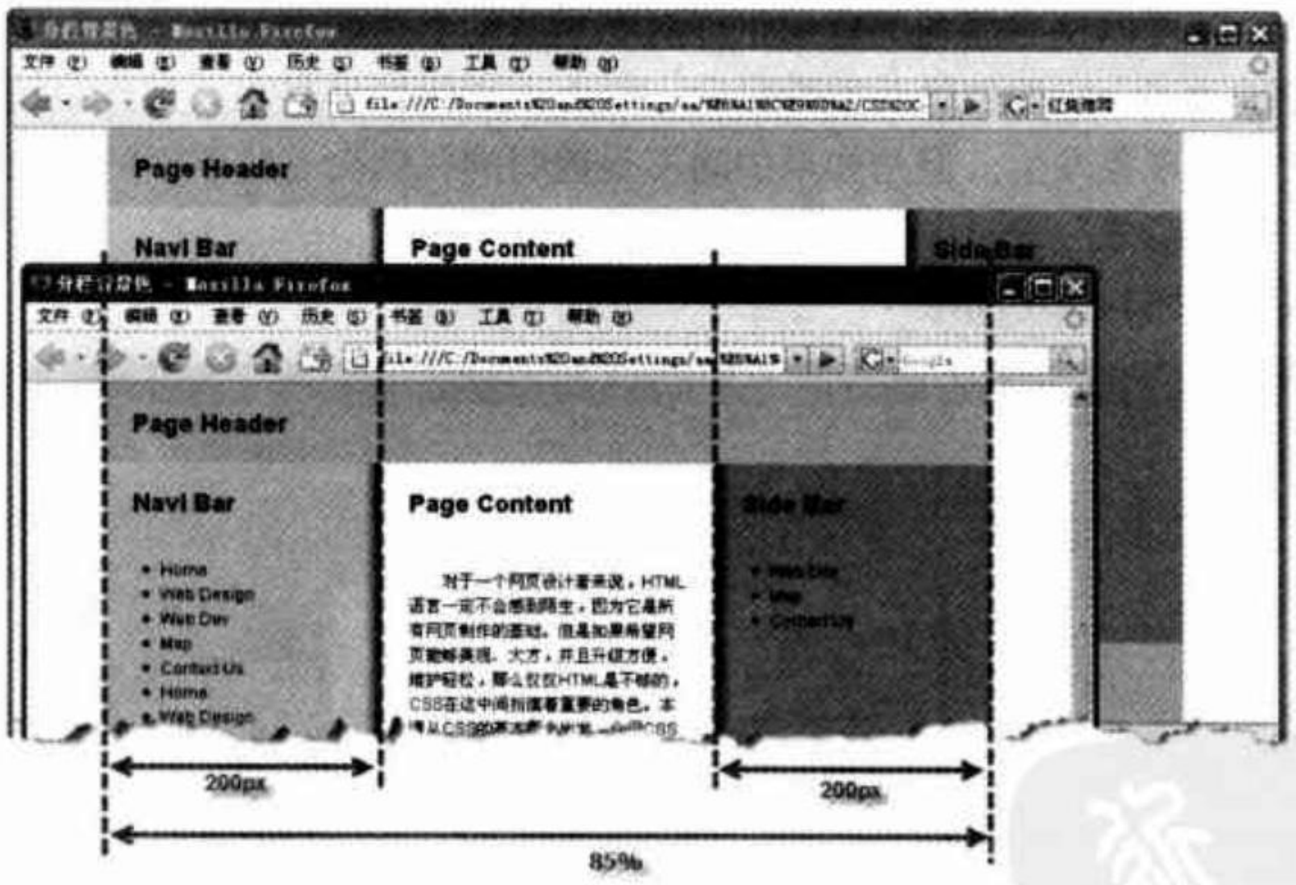


图14.25 变宽布局的背景色

14.4.4 多列等比例宽度变化布局的列背景色设置

对于3列按比例同时变化的布局，上面的方法就无能为力了，这时可以采用如下方法。

假设3列按照“1：2：1”的比例同时变化，也就是左、中、右3列所占的比例分别为25%、50%和25%。

制作方法是，制作一个很宽的背景图像，背景图像同样按照“1：2：1”设置3列的颜色。

例如图14.26所示的图像宽度是2000像素，高度10像素，左、中、右3段颜色的宽度分别是500像素、1000像素和500像素。中间段必须使用透明色，最终生成一个中间1000像素透明，两侧各500像素不同颜色的GIF格式图像文件。

图14.26 背景图像

对于网页的代码，仍然保持使用container和innerContainer两层容器，这两个容器的背景图像都设置为上面制作的这个文件。下面进行关键操作。

现在请读者回忆一下，是否前面介绍过相似的方法。

注意 背景图片的定位使用“百分比定位”，而不是通常的绝对像素值。

当设置背景图像时，如果使用像素值设置背景的位置，那么图像的左上角会定位在距离元素（如一个div）左上角指定像素数的地方；如果使用百分数定位，就会对图像上的对应点进行定位。例如，如果为一个div设置背景图像，并将水平位置设为25%，那么将会把这个图像距离左侧25%的位置定位到div距离左侧边界25%的位置。这是一个很有趣的特性。

将contatiner这个div的背景图像的水平定位值设置为25%，这样就会正好将图像左侧起25%（即500）的像素与该div的25%位置对齐。同理，将innerContainer的背景图像的水平位置设置为75%，就可以使右侧列的背景正好对上。实际上这两个背景图像是同一个图像，也就是一个图像错动了一定的位置并重叠显示。但由于图像做得很宽，并且中间是透明的，因此并不会看到重叠现象发生。但是如果中间不是透明的，就会盖住下面的图像了，因此中间段的颜色必须是透明的。

这时的效果如图14.27所示。

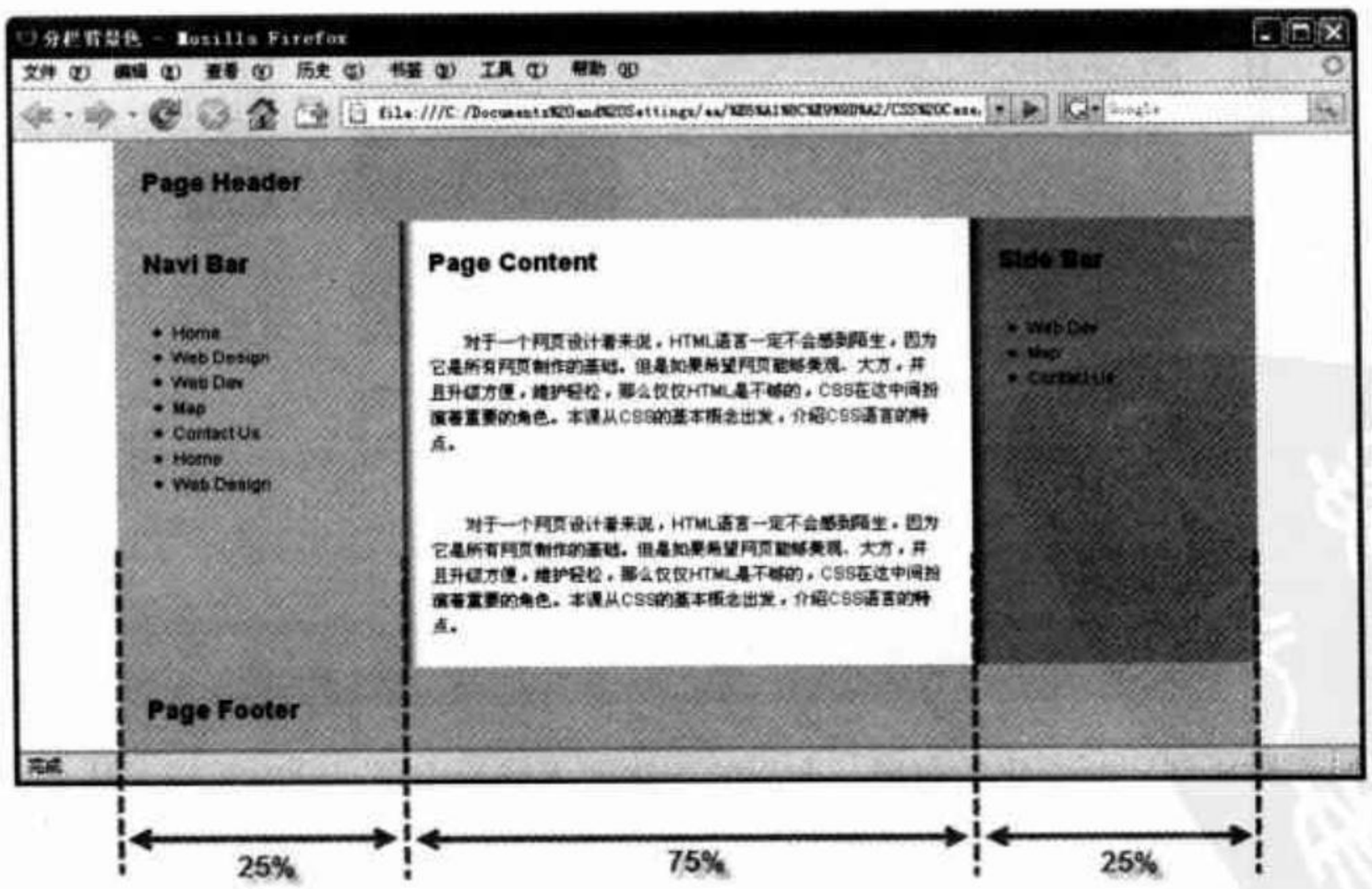


图14.27 等比例变宽布局的背景色

前面请读者思考了这种方法是否前面曾经使用过？没错，如果读者仔细分析一下这种方法的本质，实际上它就是一个“滑动门”技术的变化应用。所以“滑动门”技术是一个CSS效果的非常有用的基础性方法，通过灵活地应用“滑动门”技术，可以制作出很多不容易实现的效果。

图14.28中显示了这种方法的原理。长条矩形表示一个背景图像，两端阴影部分为各25%的背景色部分，中间白色是透明部分，两条竖直的虚线表示定位的基准线，上面的背景图像代表的是右侧列的背景图像，以右侧的基准线定位，下面的背景图像是左侧的背景图像，以左侧的基准行定位，中间的大矩形表示3列的容器div，超过它范围的部分都不会被显示。

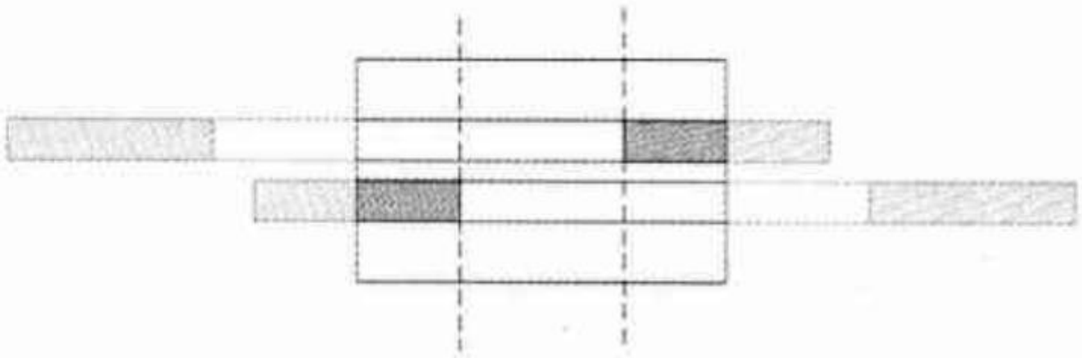


图14.28　原理示意图（1）

当左右的图像都竖直平铺以后，效果如图14.29所示，可以看到两侧的背景图像就这样实现了上面可能到的效果。

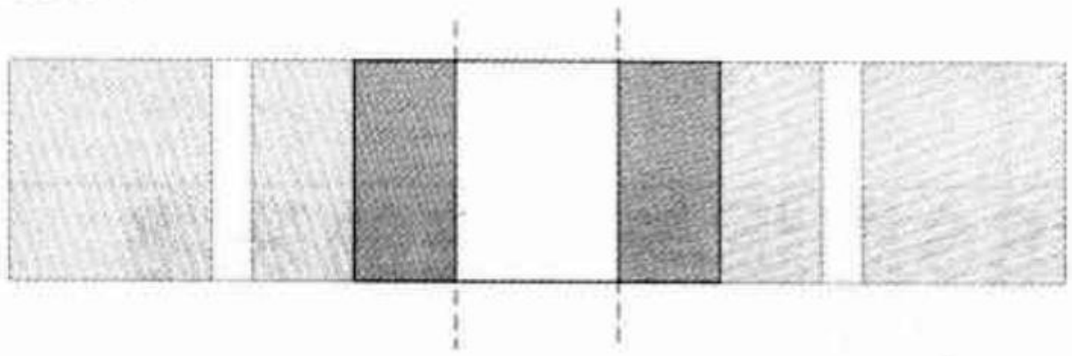


图14.29　原理示意图（2）

本案例文件位于本书光盘的“第14章\1-3-1\背景色\all-liquid.htm”。代码如下。

```
#header,#footer {
  background:#CCCCFF;
  width:85%;
  margin:0 auto;
  }
#container {
  width:85%;
  margin:0 auto;
  background:url(background-liquid.gif) repeat-y 25% top;
  position: relative;
  }
#innerContainer {
  background:url(background-liquid.gif) repeat-y 75% top;
  }
#navi {
  width: 25%;
  position: absolute;
  left: 0px;
```

```
    top: 0px;
}
#content {
    right: 0px;
    top: 0px;
    margin-right: 25%;
    margin-left: 25%;
    }
#side {
    width: 25%;
    position: absolute;
    right: 0px;
    top: 0px;
    }
```

要特别注意的是，其中：

```
background:url(background-liquid.gif)  repeat-y  25%  top;
```

这个语句中要先写25%，它的后面写top，而不要写成：

```
background:url(background-liquid.gif)  repeat-y  top  25%;
```

如果写反了，在Firefox中将不显示背景图像（在IE中仍然可以正确显示）。上面代码中有两个类似的语句，都要注意这一点。

14.5 CSS排版与传统的表格方式排版的分析

在学习完使用CSS的布局方法之后，再来回顾一下传统的使用表格布局的方法。实际上，在十多年前，互联网刚刚开始普及的时候，网页内容非常简单，形式也非常单调。1997年，美国设计师David Siegel出版了一本里程碑式的网页制作指导书《Creating Killer Web Sites》（创建杀手级网站），表明使用GIF透明间隔图像和表格可以创建出“魔鬼般迷人”的网站。

此后，使用表格布局几乎成为每一个设计师必须掌握的技术，而且Macromedia公司推出的Fireworks和Adobe公司的Photoshop等软件都提供了非常方便的自动生成表格布局的HTML代码的功能，使得这种方法更加普及。

这里简单介绍一下表格布局的原理，并与CSS布局进行一些比较。<table>标记的border属性可以设置为0，即表格可以不再显示边框以来，传统的表格排版便一直受到广大设计者的青睐。用表格划分页面的思路很简单，以左中右排版为例，只需要建立如下表格便可以轻松实现如图11.30所示的排版方式，代码如下。

```
<table border="0">
  <tr><td>banner</td></tr>
```

```
    <tr>
        <td>
            <table border="0">                    <!-- 嵌套表格 -->
                <tr>
                    <td>left</td>
                    <td>middle</td>
                    <td>right</td>
                </tr>
            </table>
        </td>
    </tr>
    <tr><td>footer</td></tr>
</table>
```

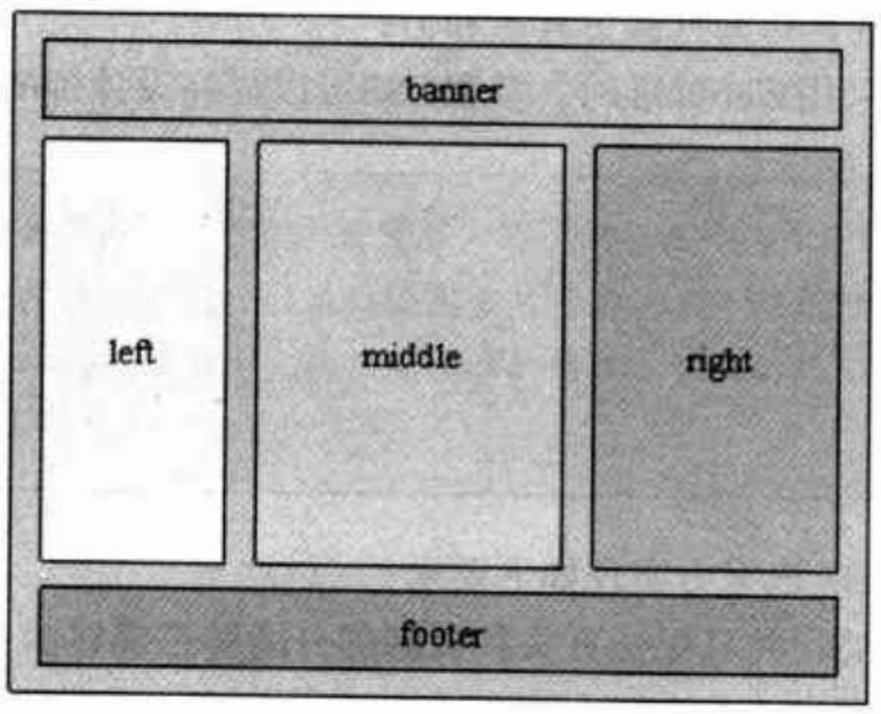


图14.30 版面布局

利用上面代码中的<table>标记就可以轻松地将整个页面划分成需要的各个模块，如果各个模块中的内容需要再划分，则可以通过再嵌套一层表格来实现。表格布局的整体思路清晰明了，无论是HTML的初学者还是熟手，制作起来都十分容易。这相对CSS排版中复杂的float和position而言无疑是很大的优势，也是目前网络上大多数网站都采用<table>标记排版的原因。

再者，由于表格中的各个单元格都是随着表格的大小自动调整的，因此表格排版不存在类似CSS排版中14.4节谈到的背景色问题，更不需要利用父表格的属性来调整。表格中块与块之间的关系十分清晰，这也是CSS排版所无法比拟的。而且表格中的<tr>和<td>等标记同样可以加入padding和border等CSS属性，简单地进行调整，更加方便易学。

表格排版也存在着各式各样的问题。首先利用表格排版的页面很难再修改或升级。像图14.30所示的构架，当页面制作完成后，如果希望将#left和#right的位置对调，那么表格排版的工作量相当于重新制作一个页面。而CSS排版利用float和position属性可以很轻松地移动各个块，实现让用户动态选择界面的功能。

利用表格排版的页面在下载时必须等整个表格的内容都下载完毕之后才会一次性显示出来，而利用div块的CSS排版的页面在下载时就科学得多，各个子块可以分别下载显示，从而提高了页面的下载速度，搜索引擎的排名也会因此而提高。

CSS的div排版方式使得数据与CSS文件完全分离，美工在修改页面时不需要关心任何后台操作的问题。而表格排版由于依赖各个单元格，因此美工必须在大量的后台代码中寻找排版方式。

总而言之，使用表格布局存在着大量无法克服的固有缺陷，因此当CSS布局方法通过一些先行者的探索逐渐被“驯服”以后，已经完全可以被普通的设计师所接受。

最后，总结一下CSS布局方法与表格布局方法比起来，有如下几点明显的优势。

- CSS使页面载入更快；
- CSS可以降低网站的流量费用；
- CSS使设计师在修改设计时更有效率，而代价更低；
- CSS使整个站点保持视觉的一致性；
- CSS使站点可以更好地被搜索引擎找到；
- CSS使站点对浏览者和浏览器更具亲和力；
- 在越来越多的人采用Web标准时，掌握CSS可以提高设计师的职场竞争实力。

注意 由于本书的篇幅限制，尽管在CSS布局方面还有很多值得进一步探索的内容，但是这里就不再深入了，这里仅提出几个值得思考的问题，如果读者有兴趣深入探讨，可以在互联网上查找相关的资料，也可以访问本书作者的网站，与作者交流。

对于CSS布局的网页应该努力实现如下要求：

- 宽度适应多列布局，并且保证页头和页脚部分能够正确显示；
- 可以指定列宽度固定，其余列宽度自适应；
- 在HTML中，各列以任意顺序排列，最终效果都正确显示；
- 任意列都可以是最高的一列，且保证不会破坏布局，不会产生重叠；
- HTML和CSS都应该能够通过Web标准的验证；
- 良好的浏览器兼容性。

以上的要求中的第3条，上面的讲解中没有深入介绍，请读者自己探索，这一条原则的目的有两个：

（1）如果页面最终效果和各列在HTML中的顺序相关，就没有真正实现内容与形式的彻底分离；

（2）无论显示效果如何，在HTML中如果能按照内容的重要程度排列，会对网站在搜索引擎的排列很有帮助，因为搜索引擎通常更重视一个页面中前面的内容。

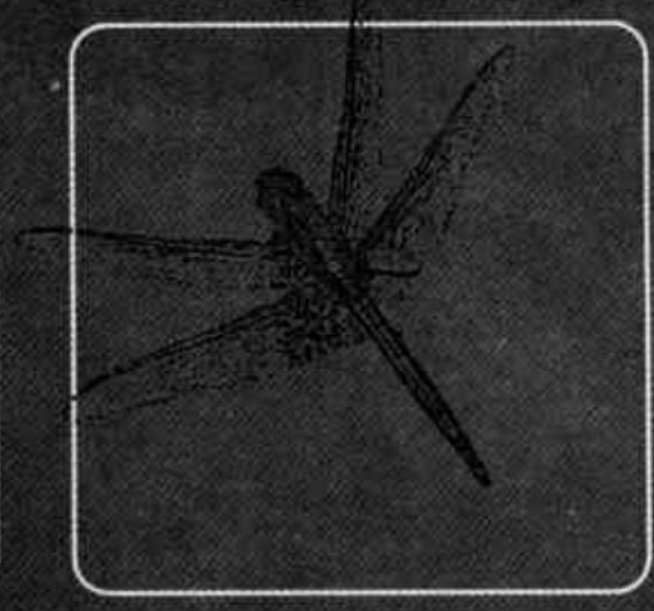
14.6 本章小结

本章核心的内容就是灵活地使用“绝对定位法”和“改进浮动法”，实现各种实际工作

可能会遇到的布局要求。

本章几乎算是全书最不容易理解的一章，因此如果读者希望透彻地理解和掌握本章的内容，就需要反复多实验几次，然后跟据14.3节的布局分类结构图中列出的各种分类，彻底地把它们搞清楚。

CSS 设计彻底研究



CSS设计

彻底研究

15

Chapter

第15章 “CSS禅意花园”作品研究

在本书的第2章中，我们介绍了一些“CSS禅意花园”的精美作品。在那时，读者还刚刚开始了解CSS，无法深入进行介绍和实践。

而现在，读者已经经过了十几章CSS和HTML案例的磨练，相信在这个过程中既感受到了思考的辛苦，也感受到了成功的喜悦。

在本章中，就结合前面所有章节介绍的技术，利用CSS禅意花园的网页和作品，进行一些综合的练习。

在本章中，介绍禅意花园的侧重点与《the Zen of CSS design》一书的区别在于，本书将更重视一个方案是如何做出来的，使读者更容易地理解其结构。在学习完本书后，还可以花一些时间，结合自己的创意，做出更多的禅意花园作品。如果读者制作出了令自己满意的作品，欢迎读者发给我们，共享读者收获的心得。

15.1 “禅意花园”页面HTML结构分析

首先介绍一下Dave Shea设计的这个HTML文档，因为所有CSS都是基于这同一个HTML文档的。该文件位于本书光盘“第15章/no-css.htm”。

在不使用任何CSS时显示效果如图15.1所示，虚线分隔开的3个部分各为一个div容器，每个容器里面又有几个div，每个div的id也在图15.1中标出了。

图15.1 没有设置CSS样式的禅意花园效果

页面完整的结构如图15.2所示。

读者只要知道这个页面中的内容分为3个部分，这3个部分的id分别为“intro”、“supportingText”和“linklist”，它们都放置在一个id为“container”的div中。每一个部分又

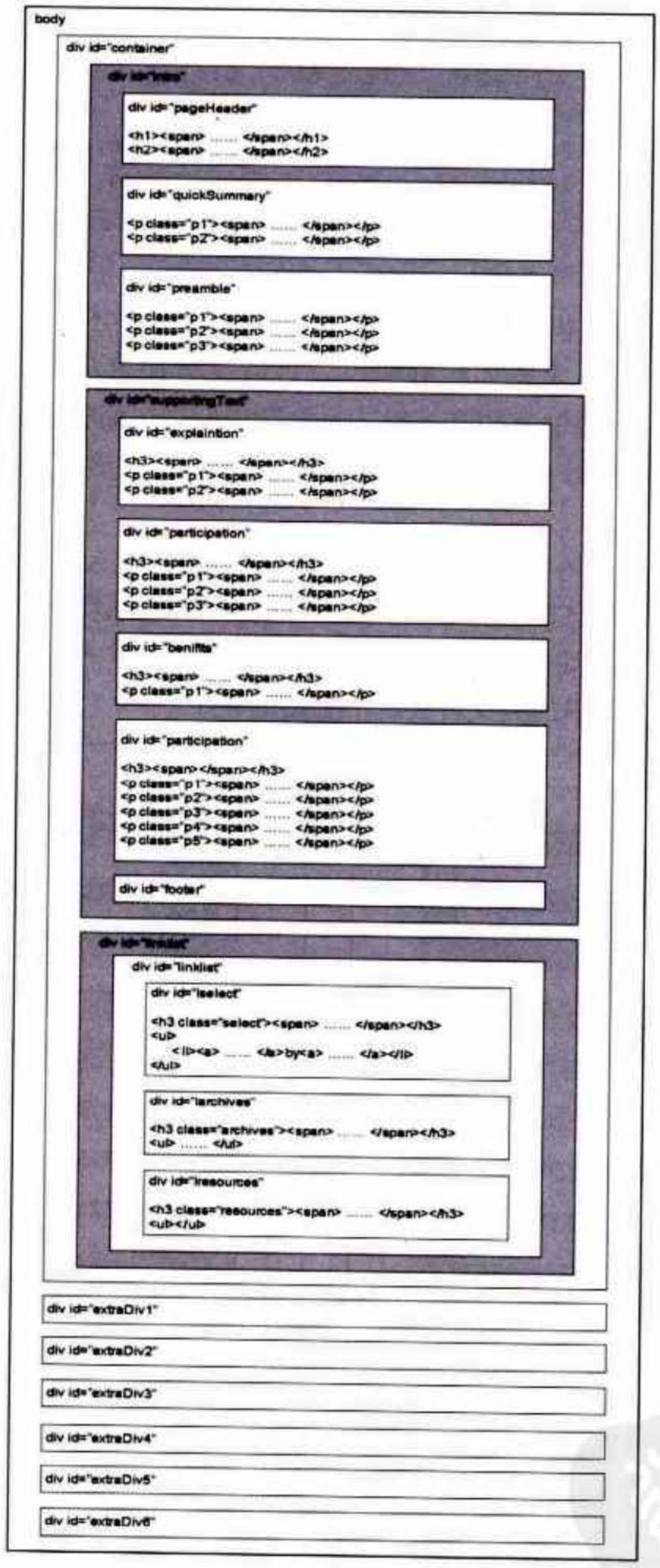


图15.2 页面的结构图

分为若干个div，每个div都有各自的id，以便使用CSS设置它们的样式。

如果仔细读一下HTML代码，就会发现它和一般的网页HTML代码有一定的区别。例如下面摘录的一小段代码：

```
<div id="preamble">
    <h3><span>The Road to Enlightenment</span></h3>
```

```
                <p class="p1"><span>Littering a dark and dreary road lay
the past relics of browser-specific tags, incompatible <acronym
title="Document Object Model">DOM</acronym>s, and broken <acronym
title="Cascading Style Sheets">CSS</acronym> support.</span></p>
                <p class="p2"><span>……</span></p>
                <p class="p3"><span>……</span></p>
            </div>
```

在每一个具体放置文字内容的div中，例如“preamble”div中，为每个文字段落都设置了id，目的是使设计师有足够的控制能力，可以灵活地单独控制任何一个段落盒的样式。此外每一个段落和标题中都套有一对和标记，这样设计师可以灵活地做出很多效果，例如可以通过这种方式使用滑动门技术，或者使用图像来替代文字内容，等等。

在实际制作一个网站的时候，当然不用这样繁琐地加入这么多标记，这里增加了这些标记，是为了实现完全不改动HTML而产生各种效果的目的，希望读者理解其中的道理。

最后，为了便于英语不是很好的读者学习，这里简单介绍一下这些id的中文含义，如图15.3所示。具体每一个段落中的英文含义就不必搞懂它了。

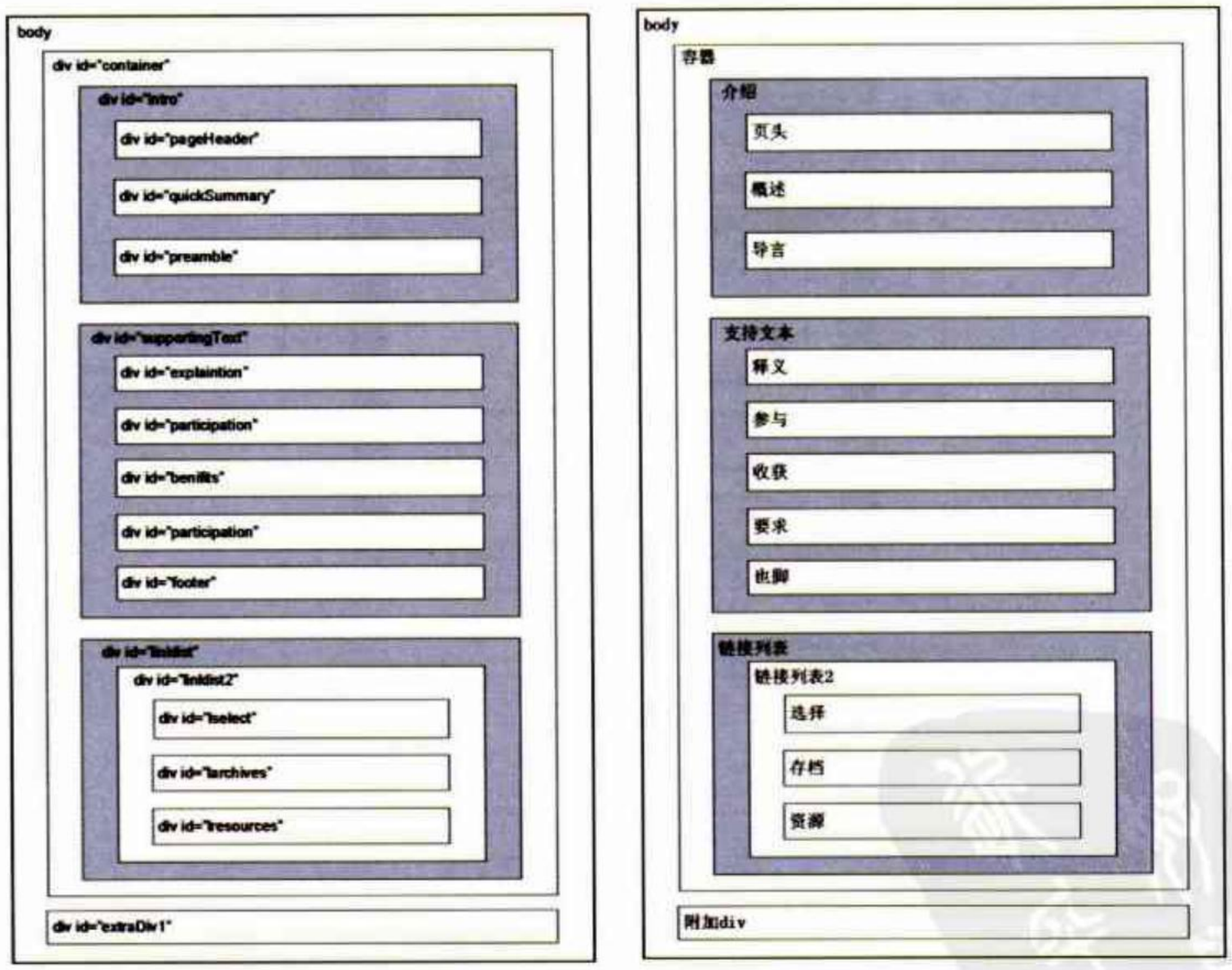


图15.3 中英文对照的结构图

现在请读者在网页编辑软件中，打开这个HTML文档，浏览一遍代码，找到上面介绍的各个代码段对应的位置，尽快熟悉这个网页，包括它的各个组成部分。

最后说明的是，在文档的末尾还有6个空的div，这几个div是留给设计师在一些特殊情况下使用的，绝大多数作品里没有用到这些附加的div。

15.2 亲自动手

下面在详细介绍比较复杂的作品之前，先自己动手为禅意花园制作一个CSS设计方案。希望读者在学习完本书之后，也能够制作出漂亮的作品，提交到禅意花园网站上。

首先制作的页面效果如图15.4所示。本例最终效果文件位于本书光盘"第15章/9999.htm"。

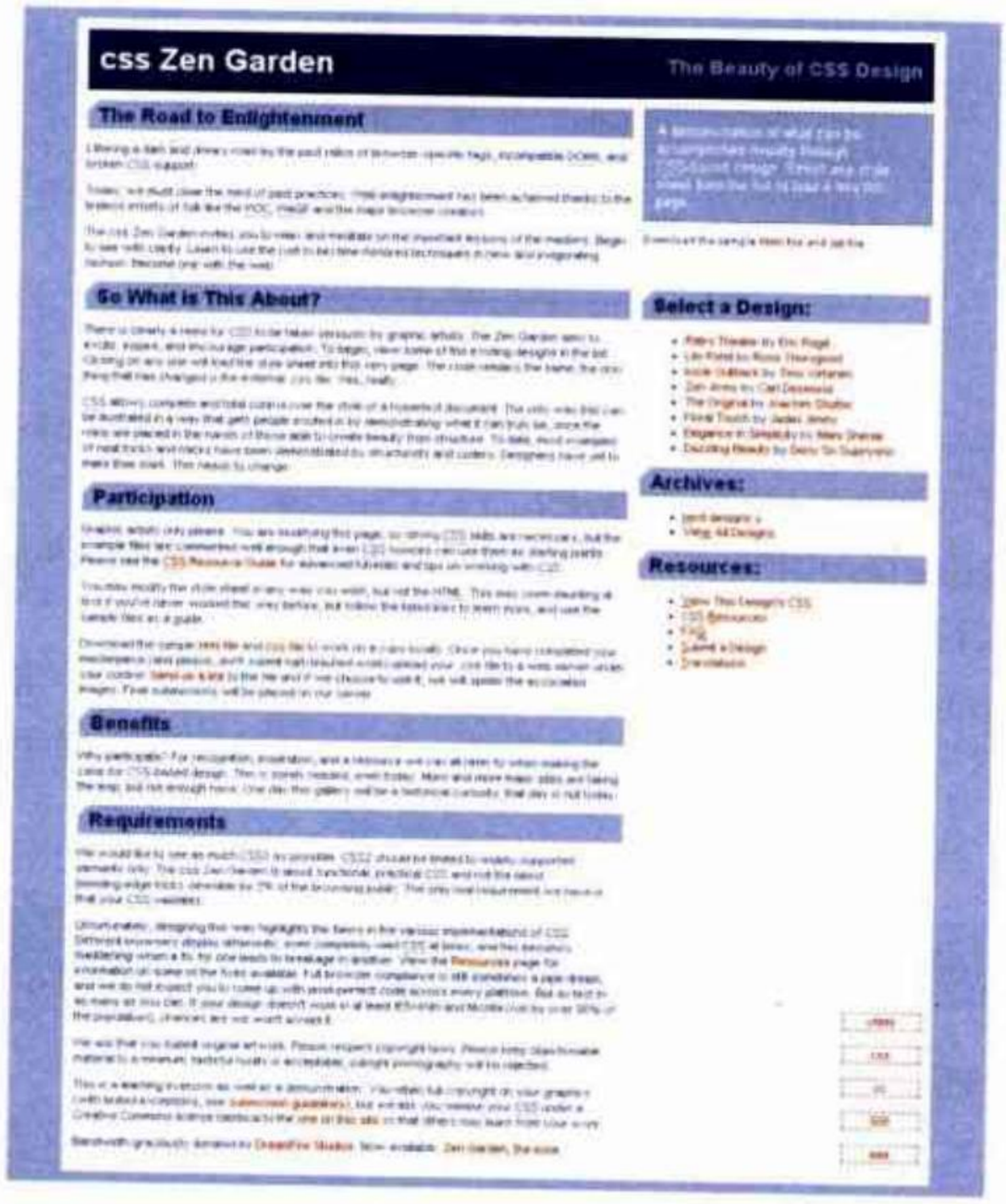


图15.4 简单的效果

15.2.1 结构分析

分析一下布局结构，如图15.5所示。希望读者，特别是英语不是很好的读者，能够记熟这几个id的英文单词，这样能够方便地理解其中的道理。

在这个布局方案中，整体的container宽度固定，水平居中。intro在最上面，pageHeader和preamble采用标准流方式，而quickSummary使用浮动方式，这样就形成了左右并列效果。在pageHeader中两个标题并排，只需要使h2向右浮动就可以了。

接下来的supportingText部分按照普通的标准流方式布局，为了给右边的linklist留出空间，可以将它的右侧margin或padding设置为适当的宽度。

此时linklist只能通过绝对定位放置到需要的位置了。它的上级元素是container，因此必须要设定到container的上边的距离。实际上，如果是我们自己来写这个HTML，那么可以把

intro部分放到container的外面，这样就可以将top设置为0，而不用管intro部分的高度了。但是由于HTML结构已经定死了，因此只能给出固定的距离顶部的高度。

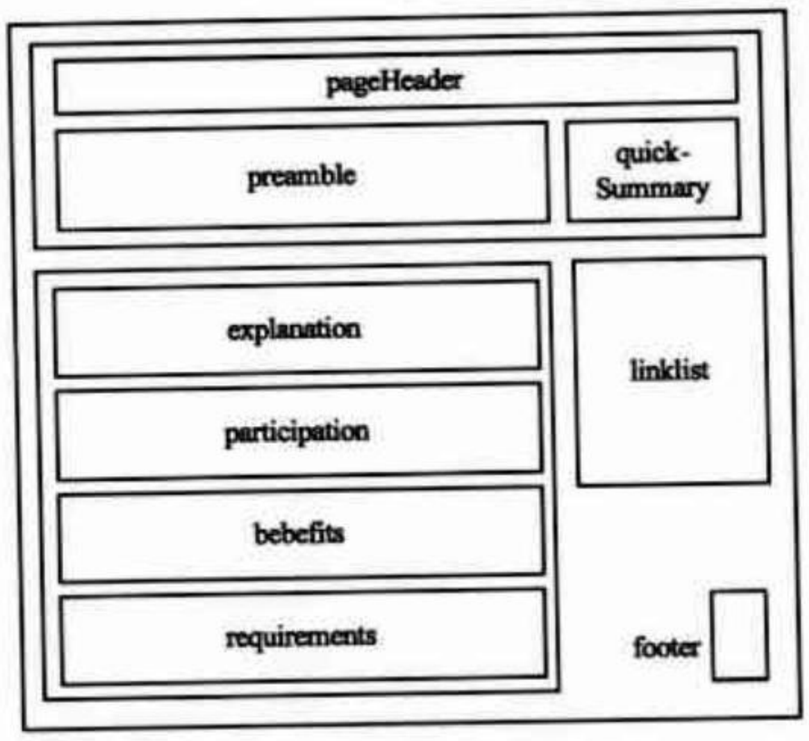


图15.5 页面的布局示意图

这里请读者思考一个问题，如果supportingText部分使用向左浮动布局，linklist向右浮动，是否可行？会产生什么问题？应该如何解决？

最后，把supportingText中的最后一个div——footer设置为绝对定位方式，放置到页面右下角，这个页面就全部完成了。

在了解基本结构和方法的基础上，下面来具体讲解一下制作过程，希望读者能够通过这个制作过程，熟悉这个页面的结构，并经常思考一下，如何把本书前面各个章节中的技术灵活地运用到这里。

15.2.2 整体设置

首先设置页面的整体背景色、文字等样式。

```
body{
  margin: 0px;
  font-family : Arial, Helvetica, sans-serif;
  font-size : 11px;
  color: #557788;
  background-color: #C0CEDC;
}
```

然后对文字段落的上下margin进行设置，这是为了使段落之间的距离在各个浏览器中相同。如果不设置的话，IE和Firefox中段落的间距会不同，而linklist需要绝对定位，这样效果就会有较大不同，因此在这里设置p的上下margin为10像素。

```
p {
  margin:10px 0;
}
```

接下来设置container的属性，宽度固定760像素，水平居中，白色背景，四周设置10像素padding。重要的一点是将container的定位属性设置为relative，这样做的目的是使它里面

的绝对定位元素可以以它为基准进行定位。

```
#container {
  background-color: White;
  width: 760px;
  margin: 10px auto;
  padding: 10px;
  position:relative;
}
```

此时的效果如图15.6所示。

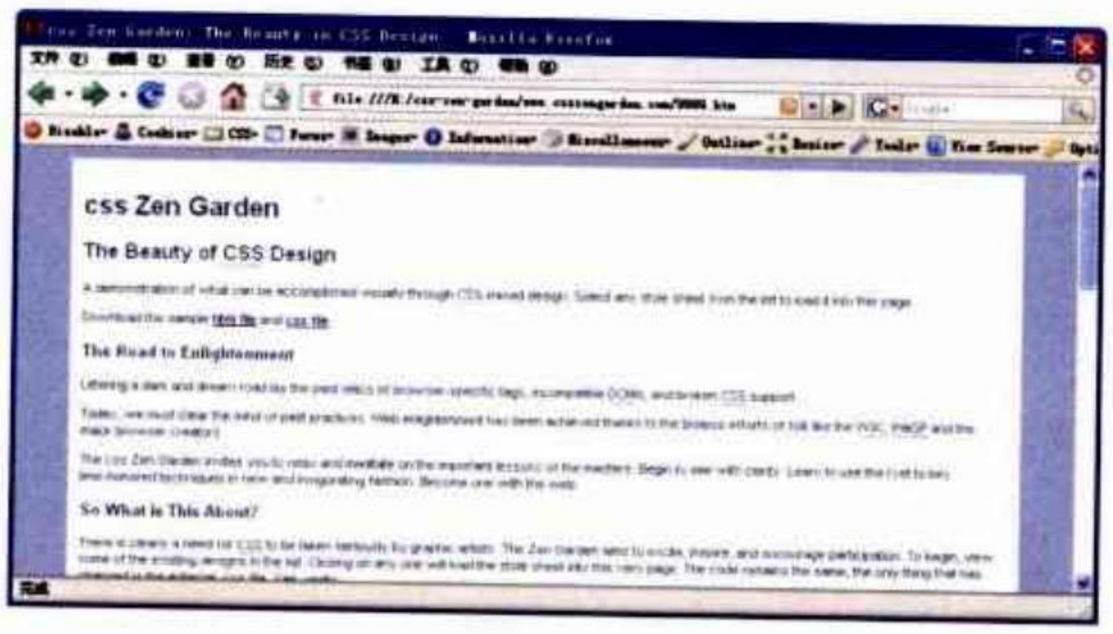


图15.6 设置页面整体效果

15.2.3 设置页头

① 现在来设置页面的大标题。注意为了使副标题靠右，可以使它向右浮动，代码如下。

```
#pageHeader {
  background-color: #557788;
  padding: 10px;
  height: 35px;
}
h1 {
  font-size : 28px;
  color: White;
  margin: 0px;
  float: left;
}
h2 {
  font-size : 18px;
  color: #C0CEDC;
  margin-top: 5px;
  float: right;
}
```

此时页面效果如图15.7所示，醒目的标题部分已经形成了。

② 设置quickSummary部分的样式，使它的宽度为260像素，向右浮动。然后设置其内部的两段文字的样式，可以做一些装饰性的修饰。

图15.7 设置页面标题效果

```
#quickSummary {
  float: right;
  width: 260px;
  }
#quickSummary .p1 {
  color: White;
  background-color: #C0CEDC;
  border:1px #578 dashed;
  padding: 10px;
  margin: 10px 0px;
  font-size : 13px;
}
#quickSummary .p2 {
  font-size : 10px;
  margin: 0;
}
```

此时效果如图15.8所示，quickSummary中的第一段文字设置了虚线的方框。

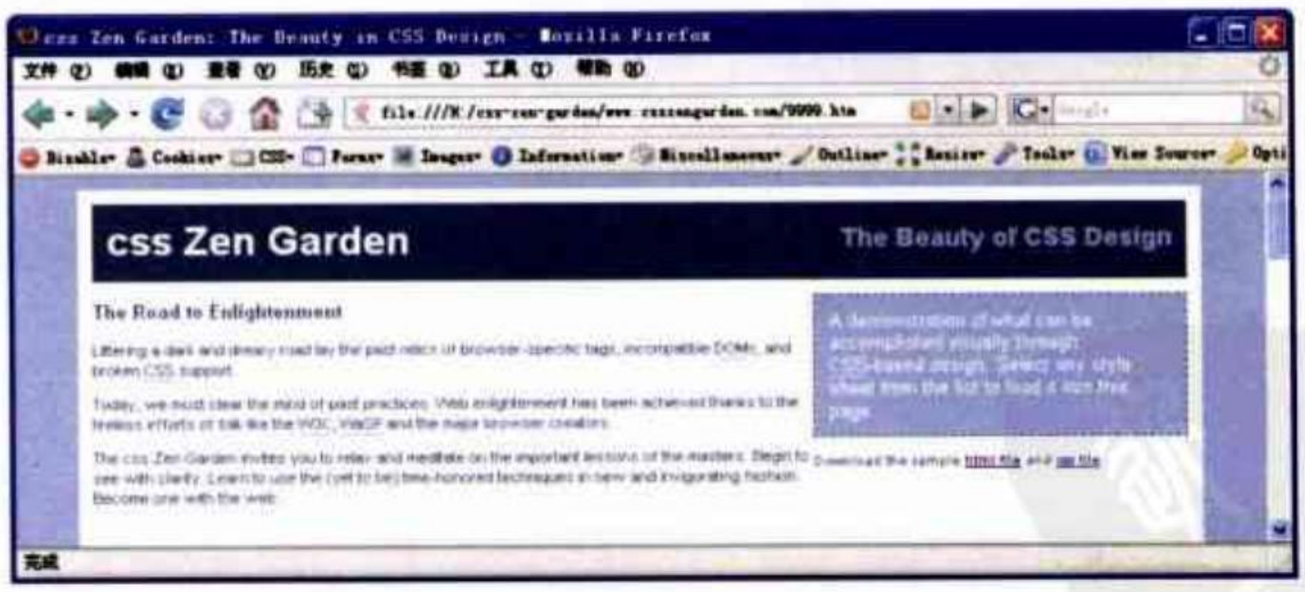


图15.8 设置quickSummary后的页面效果

③ 设置quickSummary左侧的preamble部分，设定为向左浮动，宽度为490像素。此外，还需要设置preamble中的标题样式，这里对h3进行设置，它对整个页面上的所有h3标题都会发生作用。

```
#preamble {
  width: 490px;
```

```
  float: left;
}
h3 {
  font-size: 18px;
  margin: 10px 0px 10px 0px;
  background: #C0CEDC url(corner.gif) no-repeat;
  line-height:27px;
  padding-left:10px;
}
```

此时效果如图15.9所示，preamble以及其他部分中的h3标题都使用了统一的样式，浅色背景和深色文字。

图15.9 设置preamble后的页面效果

15.2.4 设置supportingText部分和linkList部分

现在来设置supportingText部分和linklist部分，代码如下。

① supportingText使用标准流方式，因此设置右侧的margin为270像素，为linkList部分留出空间。

```
#supportingText {
  margin-right:270px;
  width:490px;
}
```

② 现在要对linkList进行绝对定位。这里希望linkList的上边缘与它左边的supportingText的最上边缘对齐，因此必须知道intro部分的高度（因为supportingText正好在intro部分的下面）。这里可以手工为intro设置一个高度，可以试验几次找到一个适当的高度，代码如下。

```
#linkList {
position:absolute;
right:10px;
top:230px;
```

```
  width: 260px;
}
#intro {
  height:230px;
}
```

可以看到上述代码中linkList绝对定位的top值就正好等于#intro的高度值。

③ 还需要注意，由于h3标题设置了上下margin，因此需要对supportingText中最上面的那个h3标题进行特殊设置，将上侧的margin的值设置为0，代码如下。

```
#explanation h3{
  margin-top: 0px;
}
```

这时的效果如图15.10所示，可以2看到这样才能保证linkList的位置看起来非常整齐。

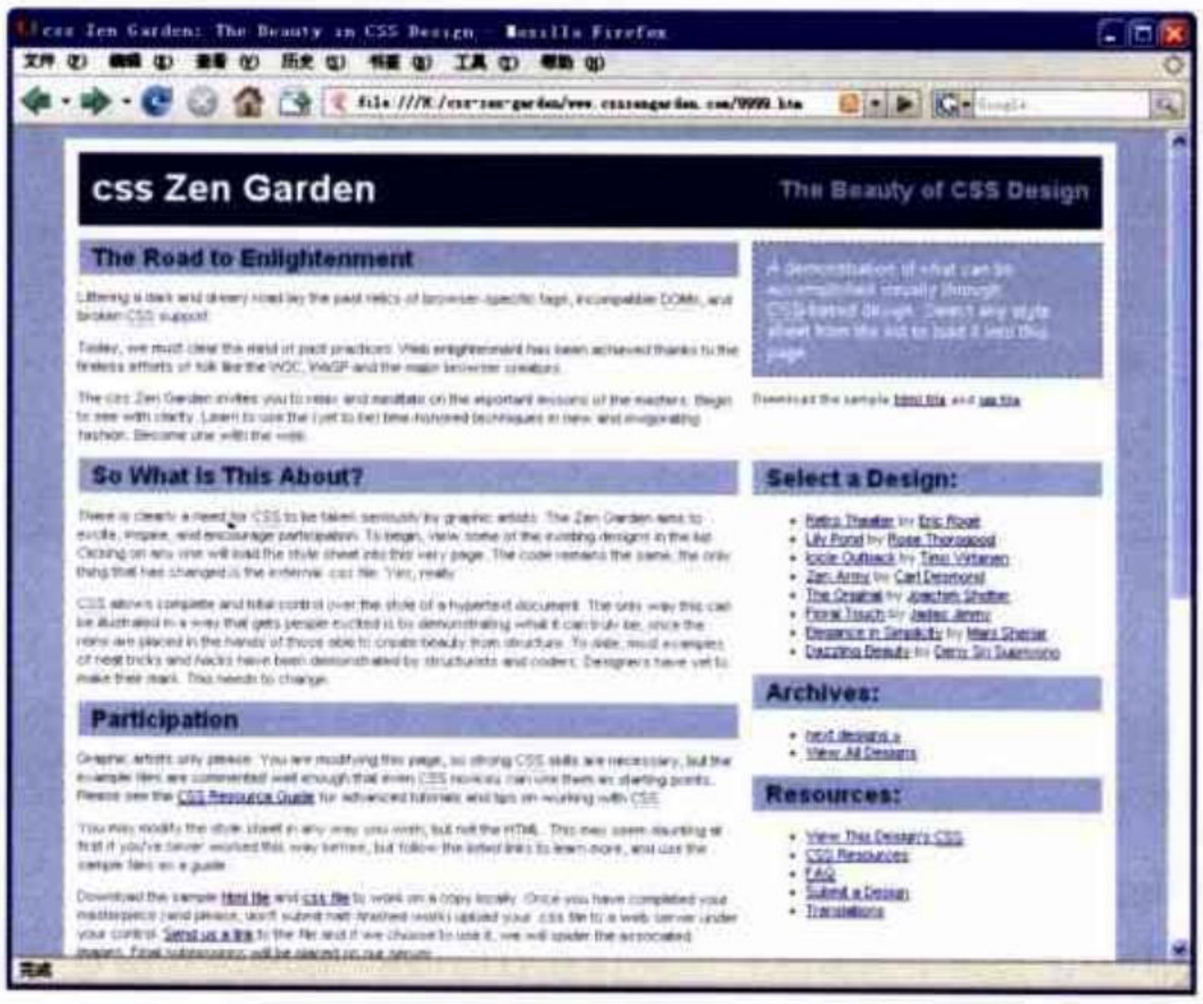


图15.10 设置supportingText和部分linkList部分后的页面效果

④ 仍然使用绝对定位，把footer放置到页面右下角，并对链接的样式进行设置，代码如下所示。

```
#footer{
  width:70px;
  position:absolute;
  bottom:20px;
  right:10px;
}
#footer a{
  text-align:center;
  display:block;
  padding:0 20px;
  border:1px #c00 dashed;
}
```

这时的效果如图15.11所示。

图15.11 设置footer后的页面效果

⑤ 至此页面的布局已经完成了，最后可以对正文中的链接样式进行设置，加上虚线边框，代码如下：

```
a:link, a:visited {
  text-decoration: none;
  color: #c00;
}
a:hover, a:active {
  text-decoration: underline;
}
```

细心的读者可能会注意到，本案例最开始的效果图中，每个h3标题的左上角有一个斜角，有了本书前面各章的基础，这是很容易实现的。

⑥ 准备一个10像素见方的图像，左上侧是白色，右下侧是背景色，然后将h3的背景图像设置为这个文件即可。

如果要实现标题左右两端都有斜角或圆角，就需要为左右两边各准备一张图片，将右边的图像放在h3中的span元素背景上即可。对这一点理解有困难的读者请参考前面涉及"滑动门"技术的章节。

到这里，这个案例就全部完成了。通过案例，熟悉了禅意花园页面的基本结构，以及设置的基本方法。这个案例设置的样式非常简洁，如果读者仔细浏览一下禅意花园中的作品，就能看到很多创意超群、精彩绝伦的页面都是基于这同一个简单的HTML代码产生的。

15.3 禅意花园作品的赏析与借鉴方法指导

在本书的第2章中，向读者展示了一些禅意花园网站上很出色的作品，但是没有深入地介绍作品的设计和制作方法。在本章中，将更深入地分析几个典型的作品，看看它们是如何布局、如何实现出来的。此外，通过这几个页面的分析过程，向读者介绍分析网页的一些基

本方法和工具，以便读者能够自己分析更多的作品，收获更多对自己有益的技术和技巧。

不少初学者在学习了一些CSS知识后，往往感觉虽然理解了基本的知识点，但是真正实践时无从下手。对于看到的一些比较复杂的页面，更是找不到分析思考的突破口。

实际上，学习技能的方法都是类似的，需要从模仿开始。这里说的模仿不是指抄袭，而是努力地去了解别人的成功作品，找出优点，进行总结。之后，再努力把这些技术用在自己的作品上，实现自己的创意，这样我们的技术水平就逐渐提高了。

其次，读者要注重实际的动手制作。有不少初学者总是在还没有把CSS的基本原理搞清楚的时候，就希望一步到位，自己创作出很有创意、很复杂的作品来，然而实际上，没有实际的动手能力是无法把创意付诸实践的，这就好像要做一名成功的服装设计师，先要能做一名裁缝，而一个好的裁缝不一定能够成为好的服装设计师一样。

总之，不断地从别人的成功作品中吸取对自己有用的东西，就能很快地提高。

15.3.1 191号作品分析

下面以191号作品为例，来进行一些讲解。图15.12所示的是第191号作品的效果图。这是俄罗斯设计师Alexander Shabuniewicz（个人网站地址http://beholder-eye.info）的作品。作品名为《The Diary》，以一个日记本为背景，右上角配有一个红色的"书签"，左下角有一个黄色的"贴纸"，作为装饰，效果非常精致。这个作品布局平衡，疏密得当，错落有致，色彩谐调，细节精致，看起来非常舒服。读者可以访问http://www.csszengarden.com/?cssfile=191/191.css查看页面效果。

这个页面是如何搭建出来的呢？初看起来这个页面似乎是一个两列的布局方案，但是仔细观察可以发现，每个部分都是标题和内容一左一右，交替错落。实际上，这是个单列的布局，右下角的linklist部分是使用绝对定位盖上去的。

注意 任何一个页面都存在多种布局方法，我们分析某一个作品案例，并不是说必须要用这种方法，而仅是为了说明一些典型的布局方法，供读者借鉴。

Dave在网站上专门提到，禅意花园网站并不是一个模板网站，里面的作品的设计和图像都不能被用于其他网站，但是鼓励大家研究这些案例，受到一些启发之后，做出更好的网站。

禅意花园的作品有一个很大的限制，就是它的HTML结构不能变。从前面的HTML结构可以看到，整个页面实际上就被分为了3个部分，并且都是依次排列的，因此很多本书前面介绍的布局方式，在这里是不能使用的。禅意花园中的很多作品都是使用了绝对定位的。

在基本了解了这个作品之后，进行具体的分析。为了了解191号作品的基本布局结构，一种方法是仔细分析它的CSS代码。

如果希望对整个页面有一个总体的认识，还可以使用一些辅助的工具。例如，使用Firefox浏览器，则可以使用它的一个扩展插件——"Web Developer"，这个插件对分析、调试页面非常有帮助。因此，这里插入一节内容，简单介绍一下这个插件的作用和基本用法。

图15.12 191号作品整体效果

首先确保安装了Firefox浏览器，因为Web Developer只能运行在Firefox浏览器上，然后打开网页https://addons.mozilla.org/en-US/firefox/addon/60，如图15.13所示。

图15.13 下载Web Developer扩展

单击“Install Now”按钮，就会自动在浏览器上安装这个插件。安装完成后，在Firefox的工具栏中会出现一个工具栏，有一横排按钮。如果没有看到这个工具栏，可以用鼠标右键单击工具栏，在弹出菜单中选择打开或关闭这个工具栏，如图15.14所示。

图15.14 打开Web Developer扩展

这一排按钮具有很多帮助设计分析网页的功能，这里结合案例介绍常用的功能。

1. 了解元素的位置和占据的空间

如果要了解一个页面中各个元素所在的位置和占据的空间，可以使用它的“outline”（轮廓线）功能，如图15.15所示。

例如按下Web Developer工具栏中的“outline”按钮，在打开的菜单中选择“outline Headings”（为标题绘制轮廓线）命令，就会为所有使用h1~h6这些标题标记的网页内容绘制各自的轮廓线框，不同的级别会用不同的颜色表示。例如，图15.15中可以看到，最上面红色线框表示的是h1元素，它下面的绿色线框表示的是h2元素，再下面有若干个蓝色线框，分别对应h3元素。可以了解到，原文件中的各级标题文字都被隐藏了，而使用图片代替了相应的文字。

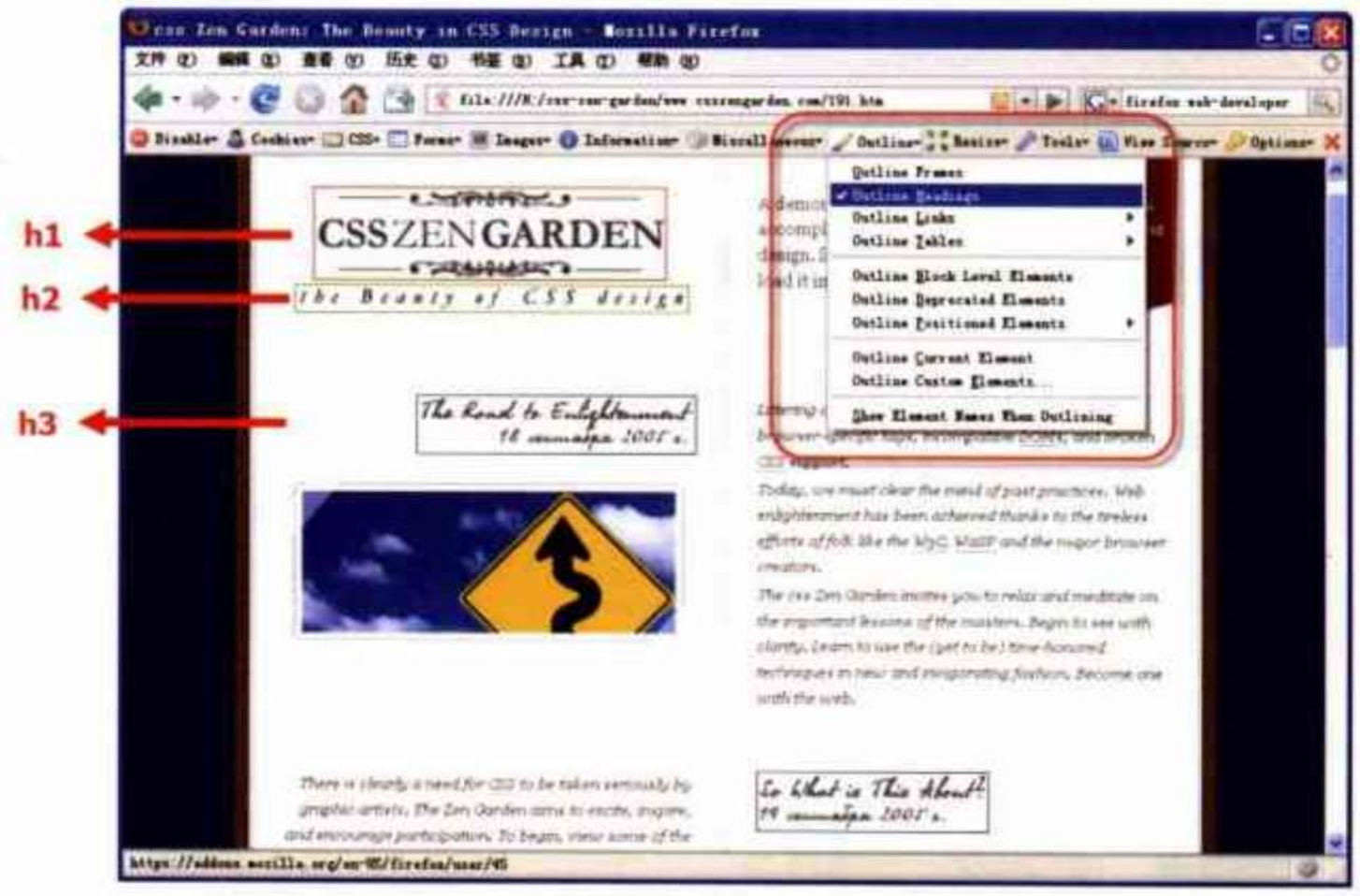


图15.15 显示轮廓线选项的作用

接着再次选中该命令，就会取消标题的轮廓线框。

2. 了解绝对定位元素

选中同一个菜单中的“Outline Positioned Elements → Absolute”（“为定位的元素绘制轮廓线→绝对定位”）命令，这时页面中使用了绝对定位也就是“position:absolute”的元素，就会被轮廓线包围。例如在191号作品中，如图15.16所示，整个内容被轮廓线包围，说明这个页面的水平居中是通过负margin实现的。此外，左下角的“linklist”区域也出现了轮廓线，这就验证了前面的判断，它确实是通过绝对定位“贴”到左下角位置的。

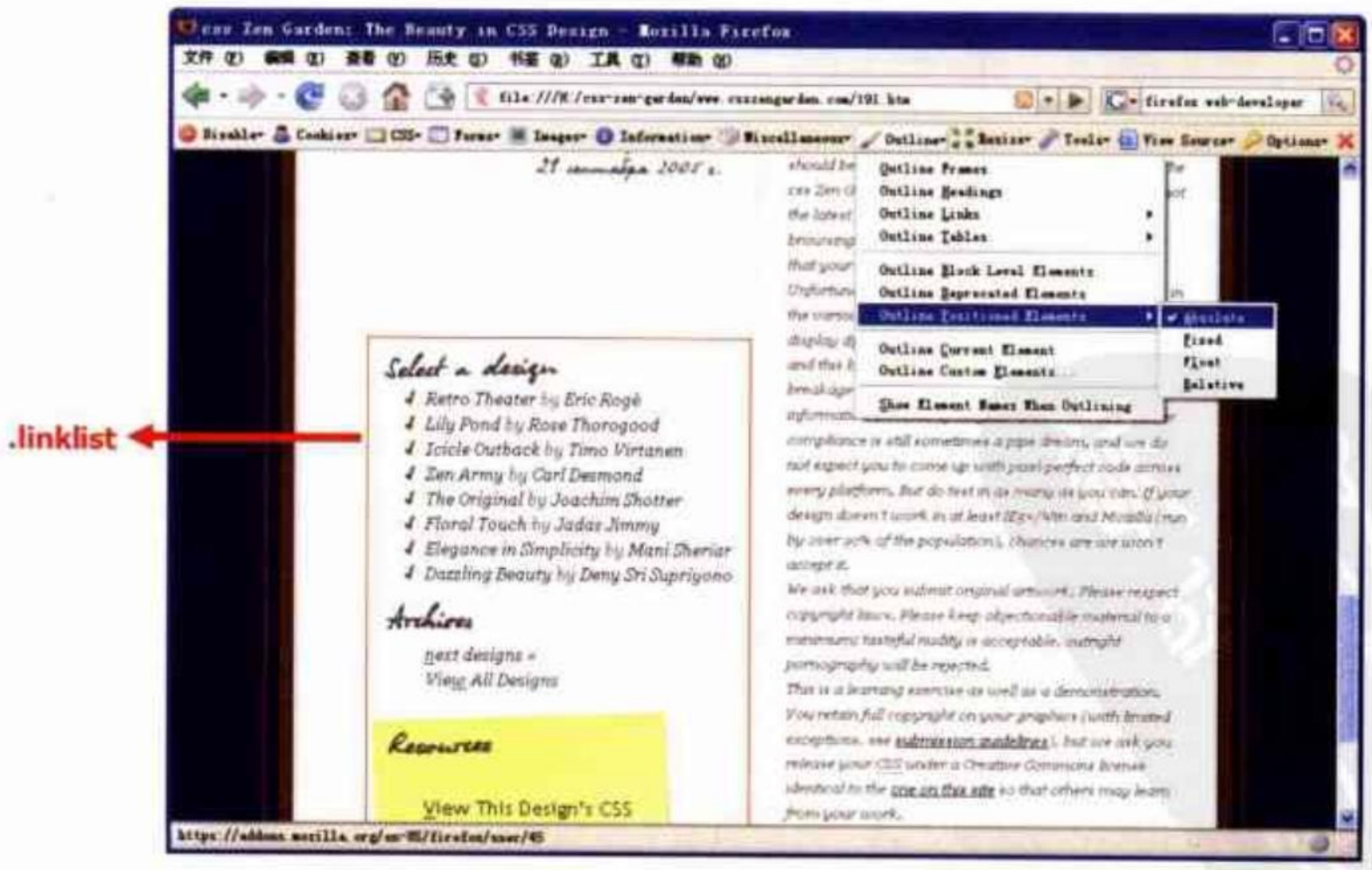


图15.16 显示出绝对定位的元素的轮廓线

3. 了解块级元素

还可以通过“Outline Block Level Elements”命令，让所有块级元素显示轮廓线，如图15.17所示。

图15.17 显示块级元素的轮廓线

如果感觉这样比较混乱，看不清楚它们之间的关系，可以使用“Outline Custom Elements”（定置轮廓线）命令，仅显示选中的元素的轮廓线。选择这个命令后，会出现一个对话框，如图15.18所示。

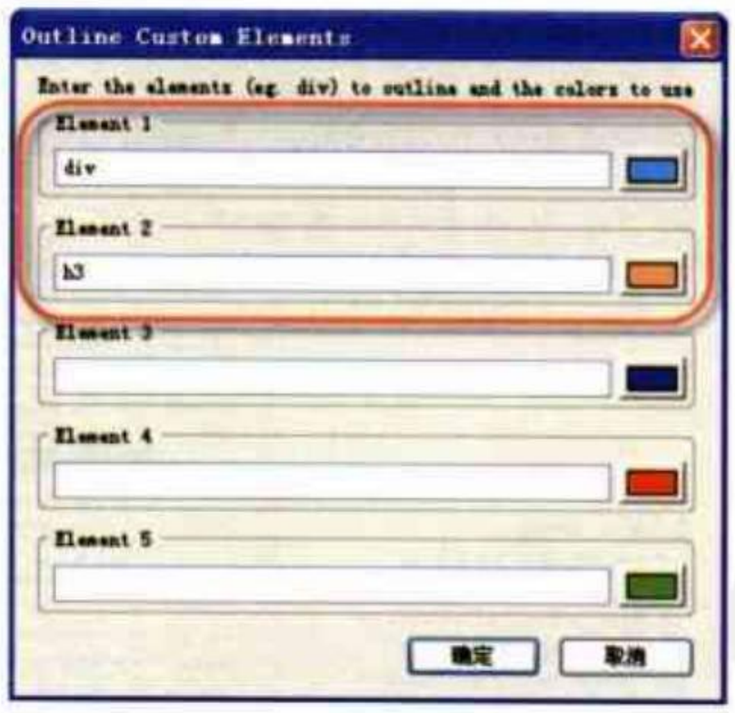


图15.18 定制显示轮廓线的方式

在对话框中可以输入一个或多个HTML标记名称，并制定轮廓线的颜色。在设置了为div和h3标记绘制轮廓线之后，可以看到页面如图15.19所示，div和h3都以轮廓线方式表现它们的位置和所占的空间。

通过上面几个简单的操作，这个页面的基本布局已经很清楚了。

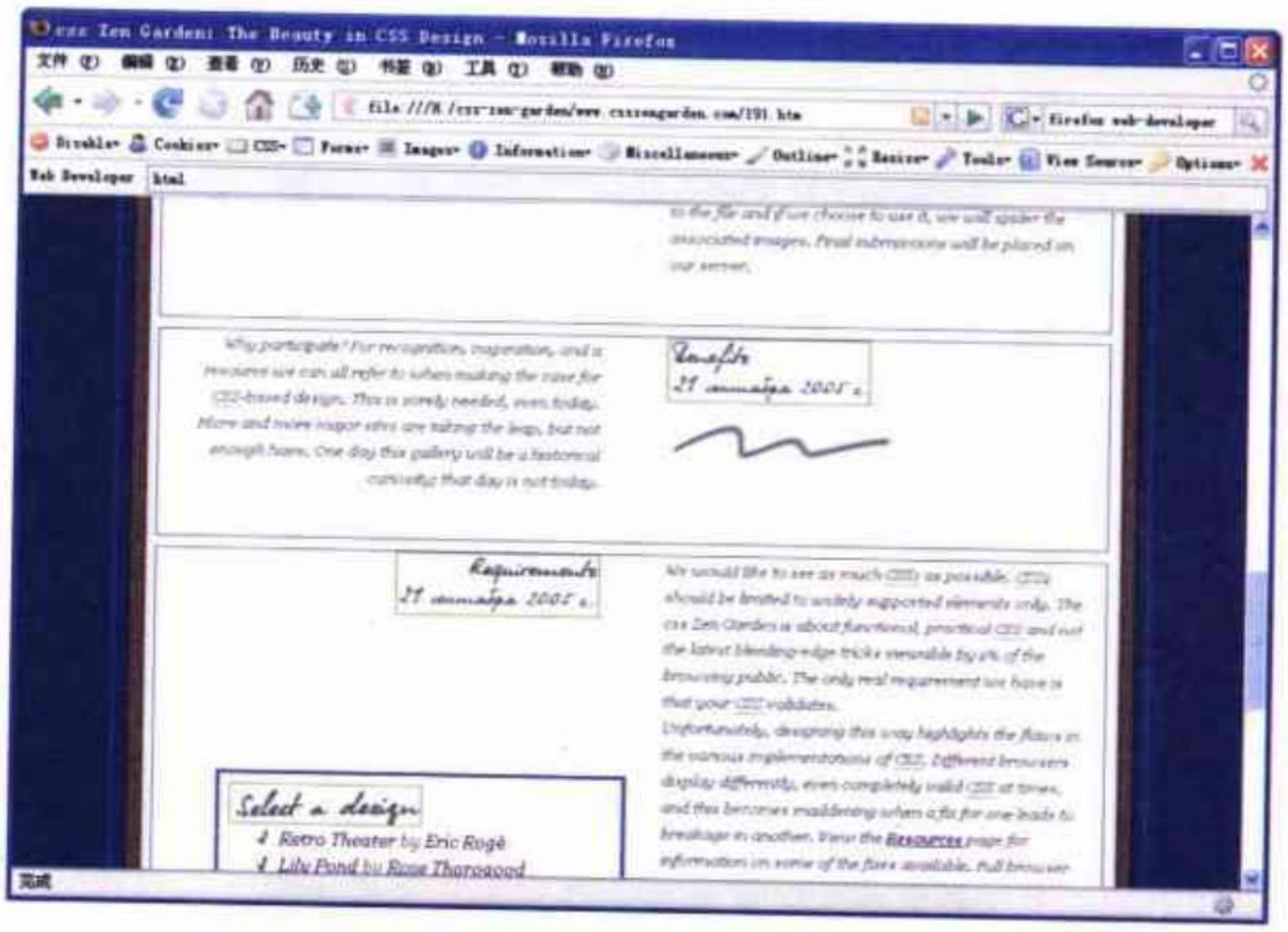


图15.19 显示div元素和h3元素的轮廓线

4. 了解网页拓扑结构

还可以选择Web developer工具栏的“Information”按钮下的“Display Topographic Infomation”（显示拓扑图信息）命令，这时效果如图15.20所示。

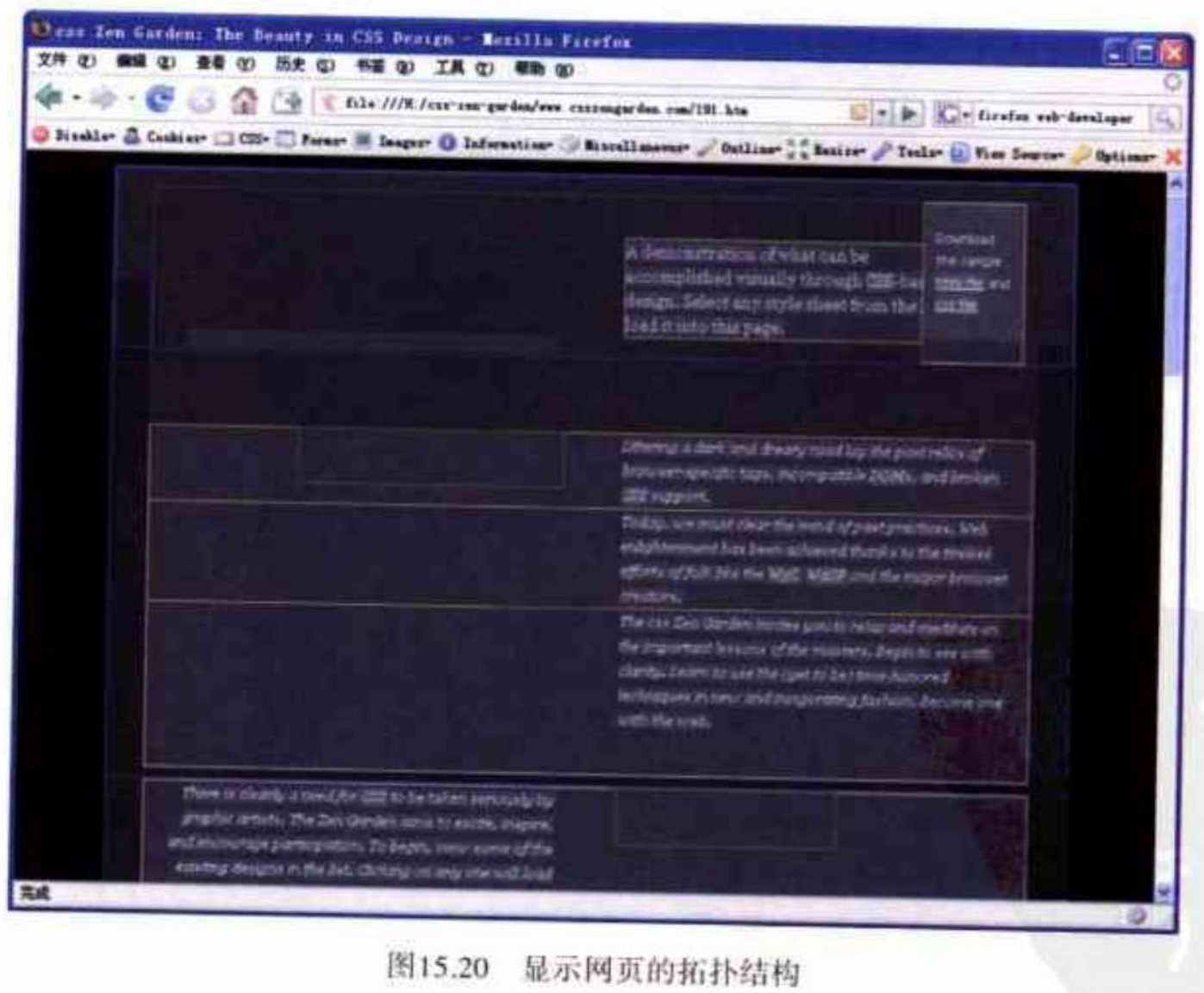


图15.20 显示网页的拓扑结构

此时页面中所有的背景图像都不显示了，文字以白色显示，各个部分的背景色是深度不同的灰色，表示了元素的嵌套深度，越浅的灰色，表示该元素的嵌套深度越浅。例如，图中文字所在区域的颜色比较浅，因为它嵌在container里面。

读者还可以通过上面介绍的方法，继续对这个页面进行分析，最终可以了解到整个页面的层次结构将如图15.21所示。

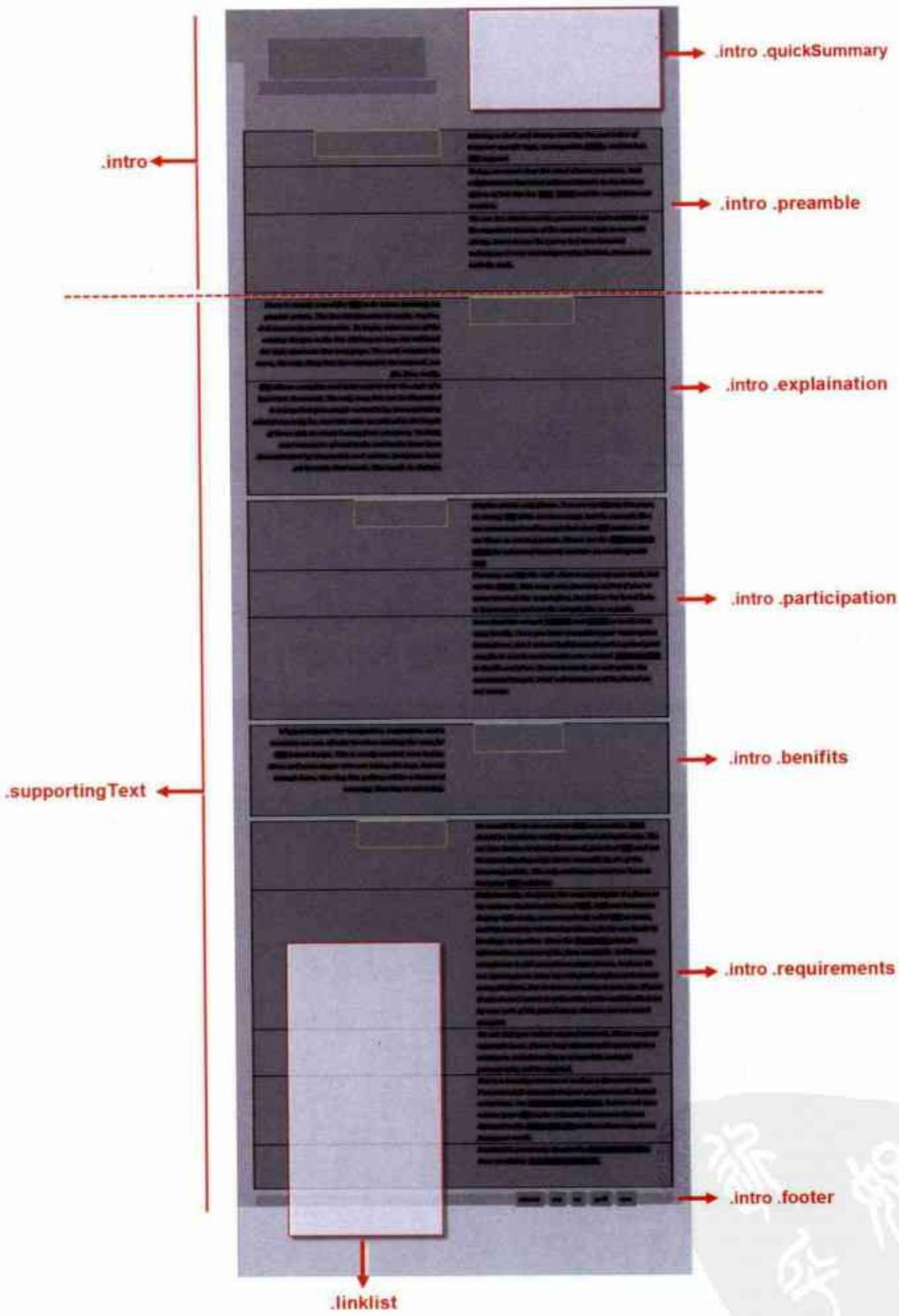


图15.21 191号作品整体结构示意图

图中虚线以上是“.intro”部分，虚线以下是“.supportingText”部分，左下角的半透明框表示“.linklist”部分。整个页面就是由这3个部分构成的，化简后的布局结构如图15.22所示。

至此，191号页面的整体结构就十分清楚了。接下来，再用两个案例说明下一页面结构的总体分析方法。

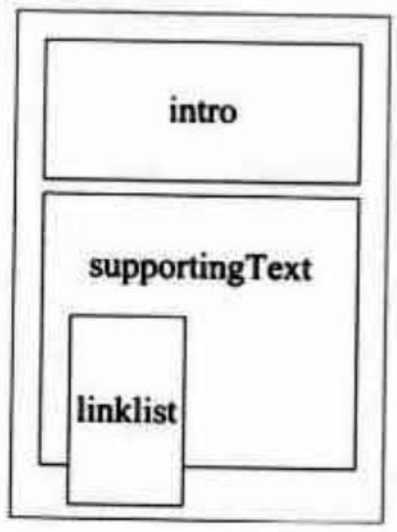


图15.22 191号作品的布局示意图

15.3.2 026号作品

026号作品是一个非常成功的作品。在前面介绍CSS布局的章节中，曾介绍过这个页面，完整的页面效果如图15.23所示。

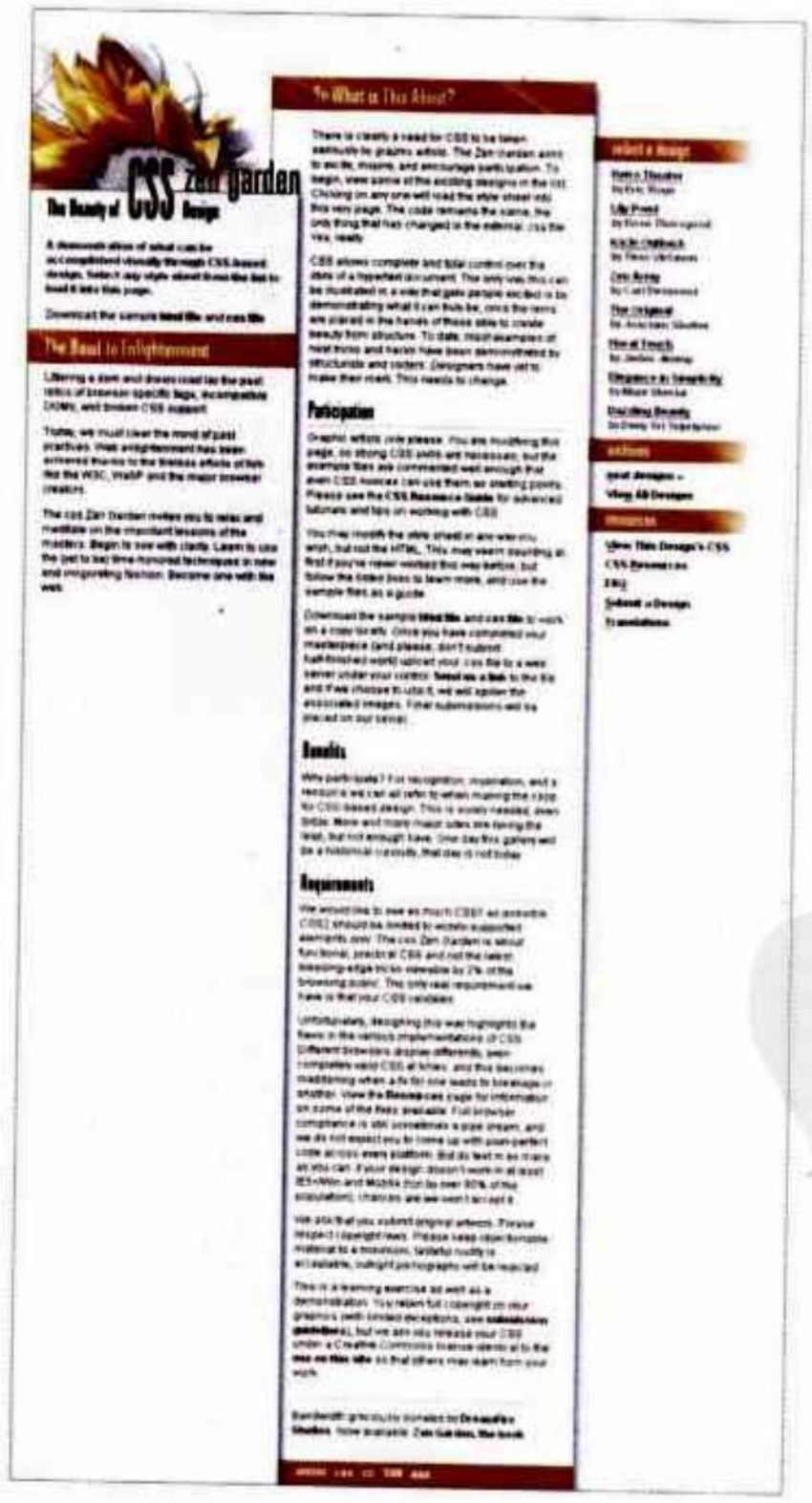


图15.23 026号作品效果图

它的布局结构如图15.24所示。

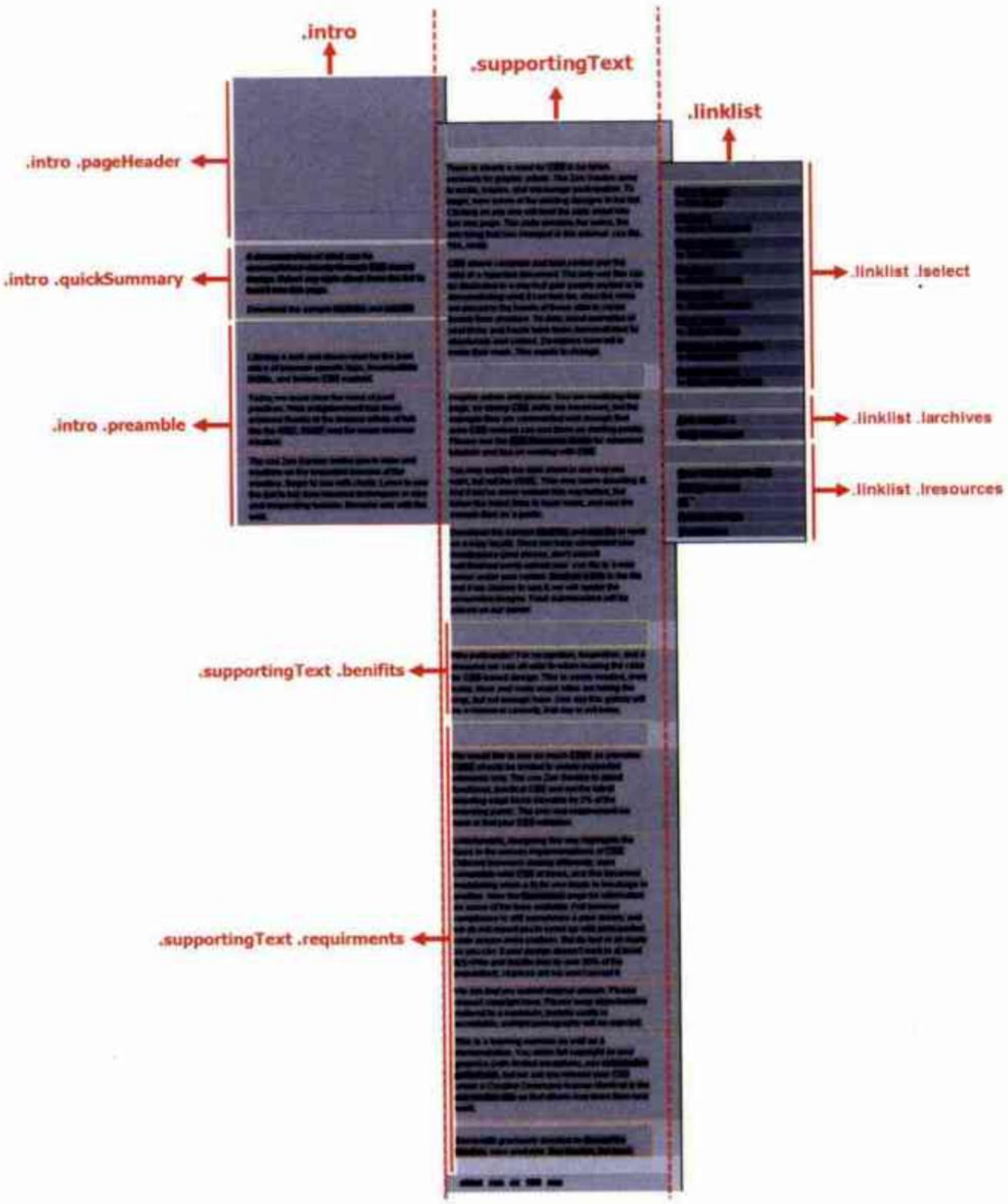


图15.24 026号作品的布局示意图

026号作品简化以后的布局结构如图15.25所示。

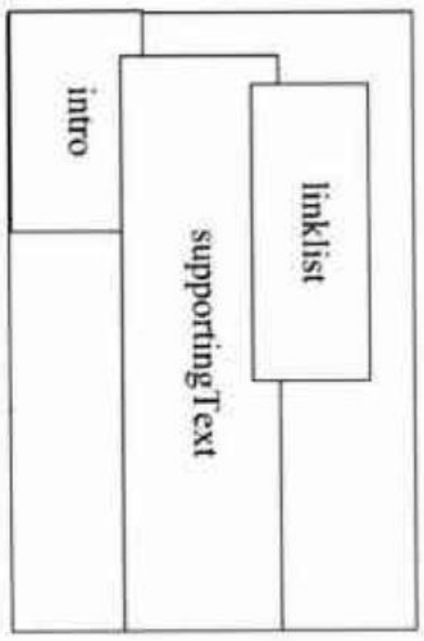


图15.25 026号作品简化后的布局示意图

15.3.3 175号作品

下面再以175号作品为例，这个作品如图15.26所示，是一个中规中矩的版式。

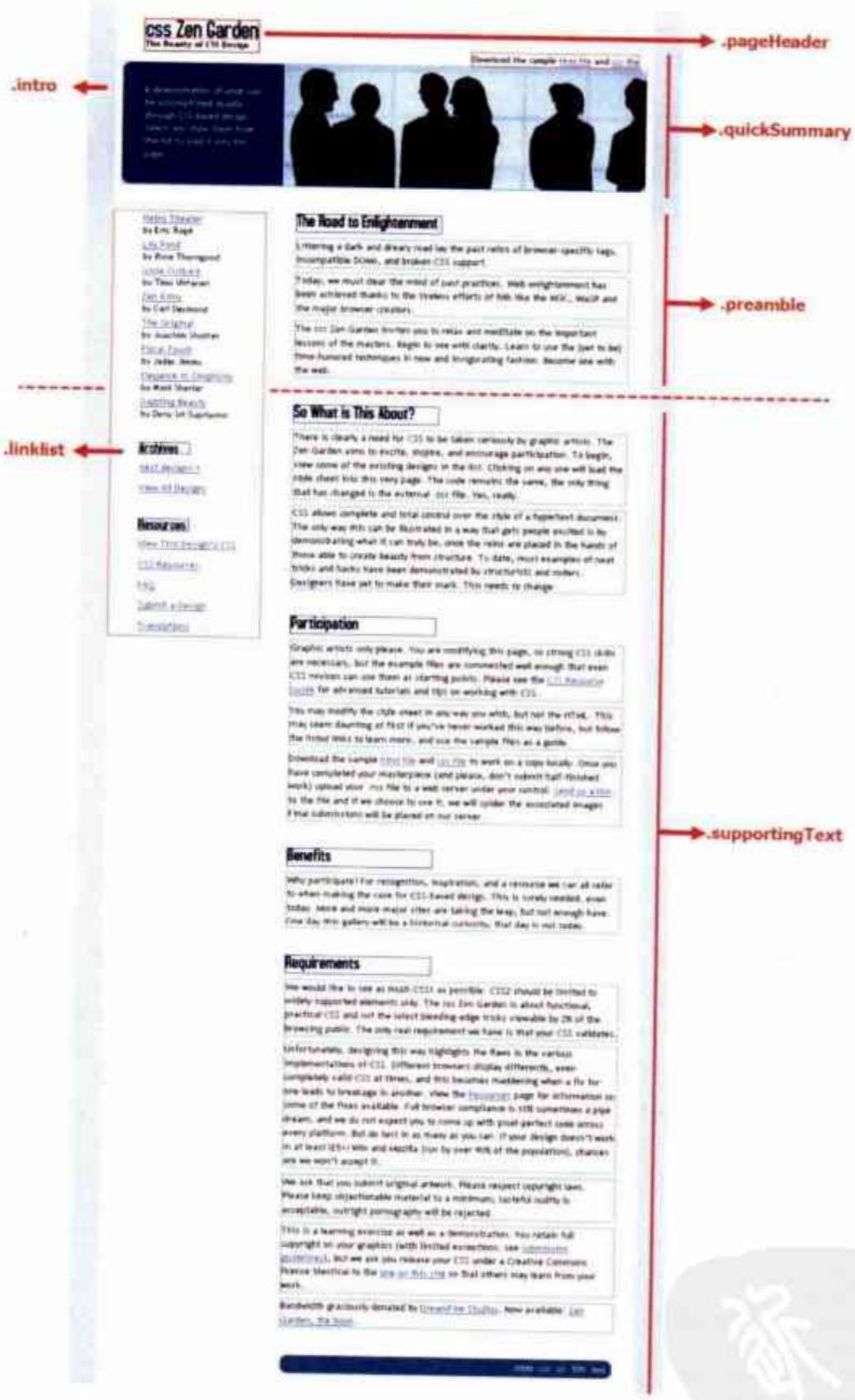


图15.26 175号作品的布局示意图

可以看到，虽然页面布局不同，但是各个div的id与前面的案例都是相对应的。这就是利用CSS禅意花园中的作品学习CSS的好处，可以不用每做一个方案就要彻底重新编写一次HTML。CSS禅意花园的HTML具有很好的灵活性，逐渐记住它的基本结构以后，就可以大量地分析、借鉴和练习，这样对提高是很有帮助的。

15.4 本章小结

本章对禅意花园网站的HTML结构进行了介绍，并在此基础上，制作了一个简单的CSS布局的页面。然后讲解了分析比较复杂的页面结构的方法。

任何在互联网上看到的好作品，读者都可以借助本章介绍的软件工具和分析方法，对它们进行分析和研究，学习它们的特点。

16

Chapter

第16章
综合案例研究

上一章中，对“CSS禅意花园”网站的标准HTML文档进行了结构讲解和分析，并制作了一个简单的网页。

在这一章中，就对禅意花园的158号作品和191号作品进行深入细致的分析和研究，彻底搞清楚它们是如何制作出来的。这两个作品的效果都相当精致，可以给读者很好的启发。

16.1 《简约夕阳》（158号作品）布局方法剖析

先来分析第158号作品，名称为《简约夕阳》（A Simple Sunset），由美国设计师Rob Soule设计，如图16.1所示。读者可以参考本书光盘中的“第16章/158.htm”。

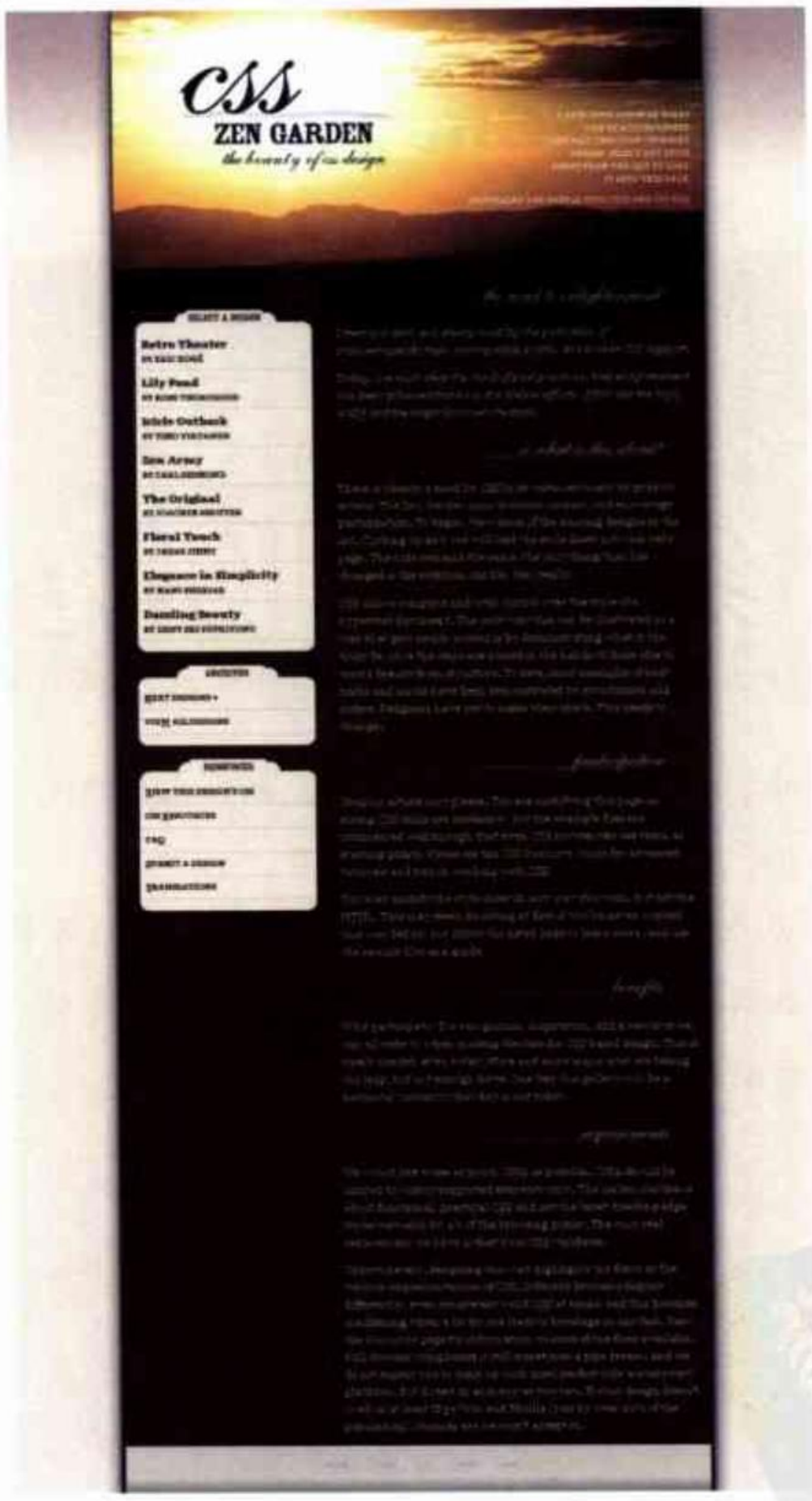


图16.1 第158号作品的效果图

在这个作品中，充分使用了图片的技巧，制作出了非常协调的作品意境。下面就来详细地看一下这个作品的设计思路和过程。

在前面已经分析了几个禅意花园作品，读者应该能够不用看代码，就知道本案例的布局方式了。

这个页面最上面的落日图片是pageHeader部分的背景，下面的背景是由一个带有左右边框的图竖直平铺获得的。整体背景很自然，上部是渐变色，下部是固定色，这可以通过对整体的背景进行设置来实现。其内部的各个div排列是比较规整的，linklist使用绝对定位，放在页面左侧。具体做法请看下面的详细介绍。

16.1.1 设置渐变的页面背景

为了使页面的整体背景在最上面有渐变的效果，准备如图16.2所示的图像。这个图像会水平方向平铺在页面的最上端，竖直方向不平铺。注意它最下端的颜色是#F0E9CE，也就是页面中背景图像没有覆盖到的地方的颜色。

图16.2　页面顶端的背景图像

对body的CSS样式设置代码如下。

```
body {
  font: 11px georgia,times,serif;
  color: #F0E9CE;
  background: #F0E9CE url(mainbg.gif) repeat-x top;
  margin: 0;
  text-align: center;
  }
```

注意其中的background首先设置了背景色，后面又设置了背景图像。这样做的效果就是背景图像没有覆盖到的地方颜色为#F0E9CE。这种颜色与背景图像最下端的颜色相同，所以过渡非常自然，如图16.3所示。

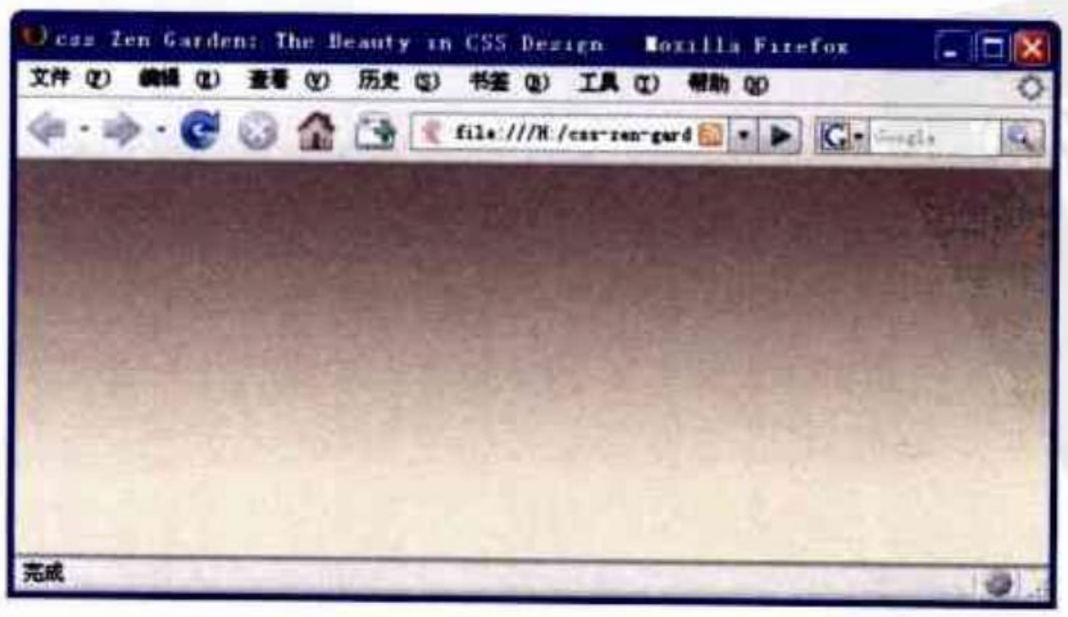


图16.3　设置页面背景图像和背景色后的效果

本案例最重要的3个图像文件如图16.4所示。这3个图像竖直紧密排列，贯穿整个页面。

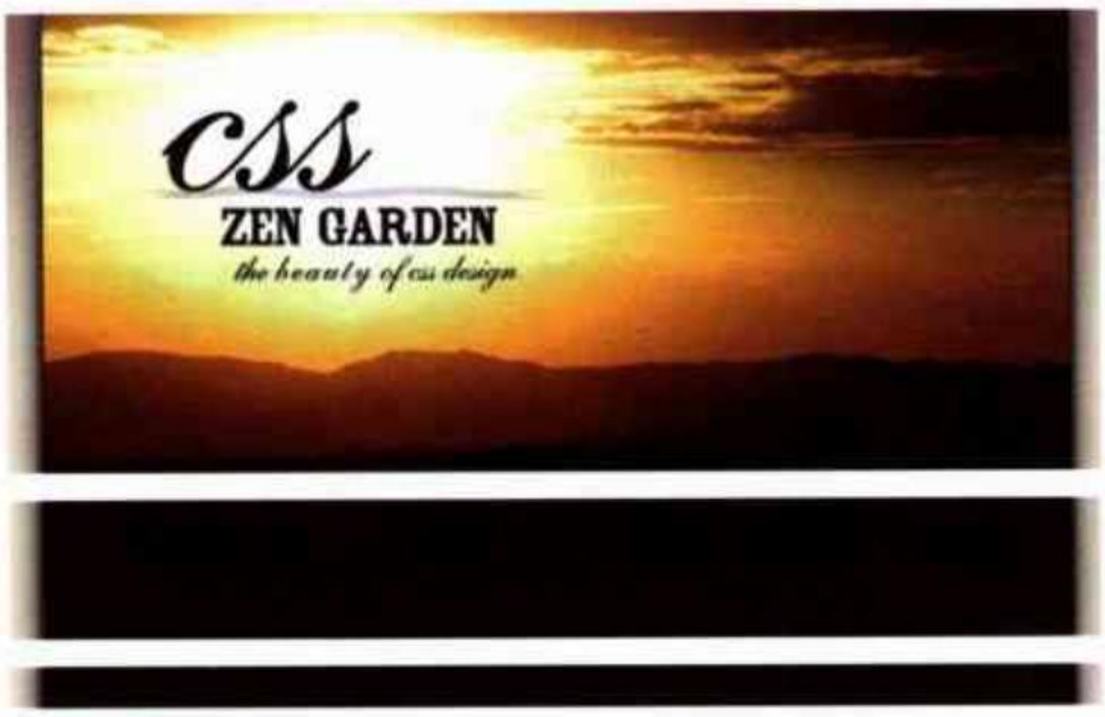


图16.4 container部分的3个背景图像

在3个背景图像中，前两个分别是pageHeader和quickSummary的背景图像，第3个是用于竖直方向平铺的背景。

```
#container {
  margin: 0 auto;
  text-align: left;
  background: url(contbg.gif) repeat-y center;
  width: 672px;
  }
#intro,#supportingText {
  line-height: 19px;
  }
#pageHeader {
  height: 287px;
  background: url(top.jpg) no-repeat;
  }
#quickSummary {
  background: url(prebg.gif) no-repeat center top;
  height: 87px;
  font-size: 9px;
  text-transform: uppercase;
  }
#preamble {
  width: 380px;
  margin-top: -95px;
  padding: 0 0 0 260px;
  font-style: italic;
  }
#supportingText {
  width: 380px;
  padding: 0 0 0 260px;
  }
```

这时的效果如图16.5所示。

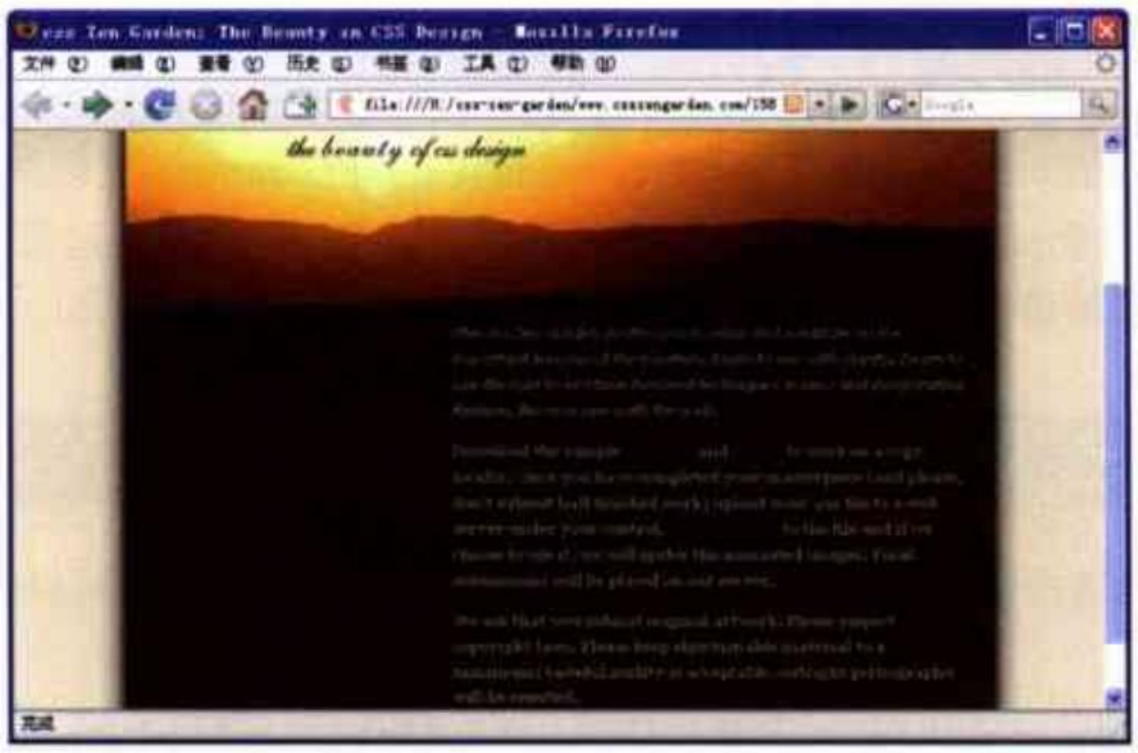


图16.5　总体设置完成后的效果

16.1.2 设置整体结构

在设置上面的背景色之后，整个页面的外层框架就搭建起来了。为了使读者能够清晰理解上面的设置，并对下面要进行的设置有所了解，这里给出了页面布局的整体结构，如图16.6所示。

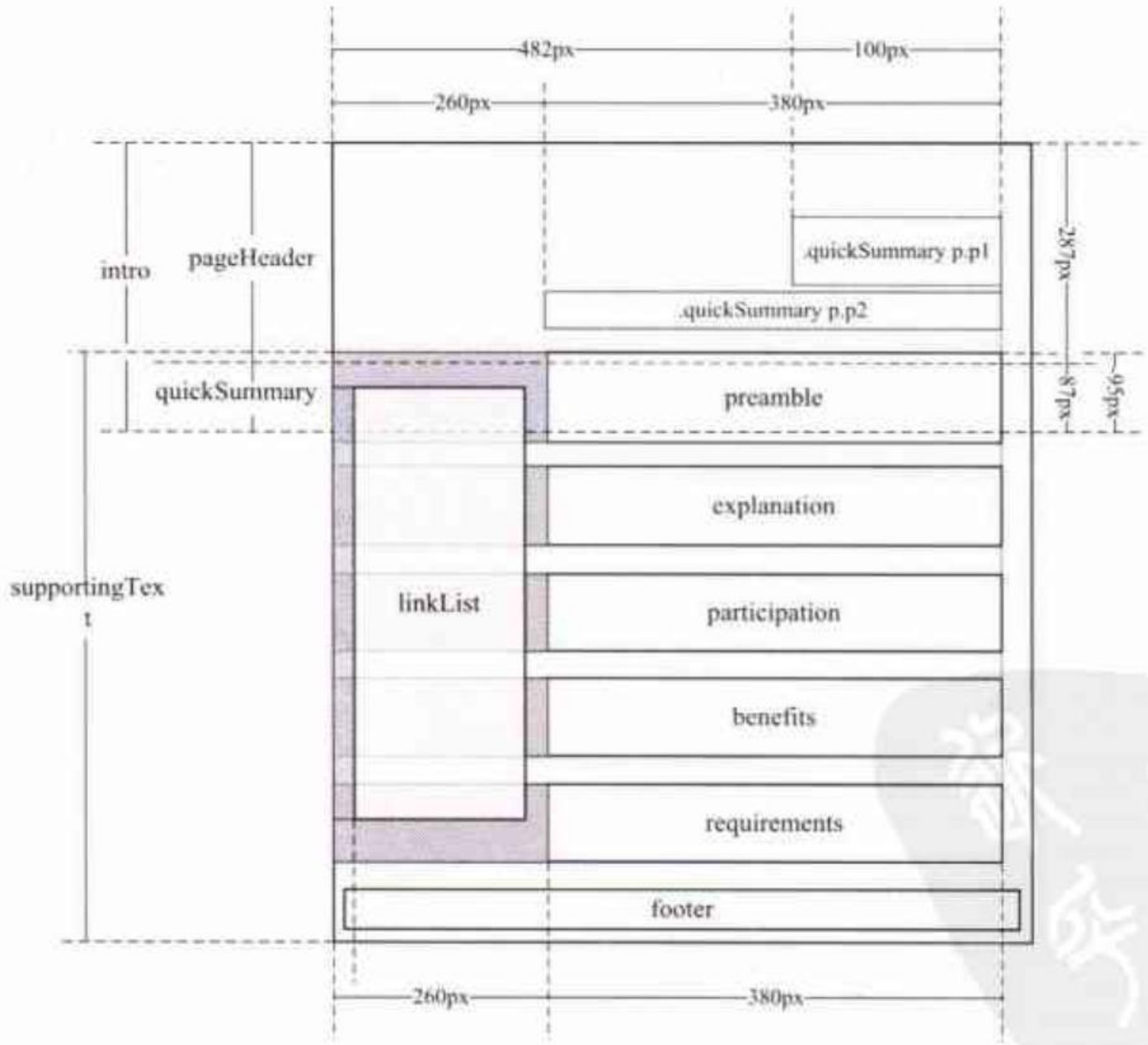


图16.6　页面布局结构图

在图中可以看出，最上面的pageHeader高度为287像素，它的下面是suickSummary，高度为87像素。原本preamble应该在quickSummary的下面，现在通过设置负的margin-top值，

将它向上移动了95像素，这样preamble实际上就和quickSummary重叠了。注意quickSummarry中的两段文字都通过绝对定位移动到了pageHeader部分的图片上。

supportingText在preamble的下面。supportingText和preamble的宽度都是380像素，并都设置了260像素的左侧padding，这样就给linkList留出了空间。然后通过设定margin和width，使supportingText中的最后一个div不再挤到页面右侧，而是贯穿整个页面。

16.1.3 设置linkList

在了解了基本结构以后，继续设置各个部分的布局。在设置时，可以随时参考上面的结构图。

linkList部分原本应该在supportingText的下面，现在通过绝对定位，把它放到页面的左侧。代码如下。

```
#linkList {
  position: absolute;
  top: 325px;
  padding-left: 45px;
  color: #2B0101;
  }
```

注意代码中使用了绝对定位，但是它的上级元素中都没有设置定位属性，因此它是以浏览器窗口定位的。这里需要注意，只需设置top的值，而不要设定left的值，这样就可以使水平方向仍保持在原来的位置上。为了使linkList的内容向右偏移，这里将左padding设置为45像素。

至此，3个大部分就都各就各位了。下面的任务就是参考上面结构图中的数据，把3个大部分中的具体内容放置到适当的位置上。

16.1.4 设置各个部分的标题

在premeable中有一个h3标题；supportingText部分的前4个div中，各有一个h3标题；在linkList的3个div中，各有一个h3标题。为了使用图像替换这8个h3标题的文字，需要把文字隐藏起来。以前的案例使用的方法是，将里面的span的display设置为none，就可以不显示它们。这里使用了另外一种方法，代码如下：

```
#preamble h3,
#explanation h3,
#participation h3,
#benefits h3,
#requirements h3{
  text-indent: -5000px;
  height: 37px;
}
#lselect h3, #lresources h3, #larchives h3 {
  height: 23px;
  width: 189px;
```

```
    margin: 0;
    text-indent: -5000px;
  }
```

可以看到，这些文字都通过设置一个绝对值很大的负的text-indent（文字缩进）属性，使文字移动到页面外边，并且设置了它们的高度，这样这些h3标题的文字虽然看不到了，但它们仍然占据着高度空间。

接下来，把这8个h3标记的背景属性依次设置为相应的背景图像，如图16.7所示。

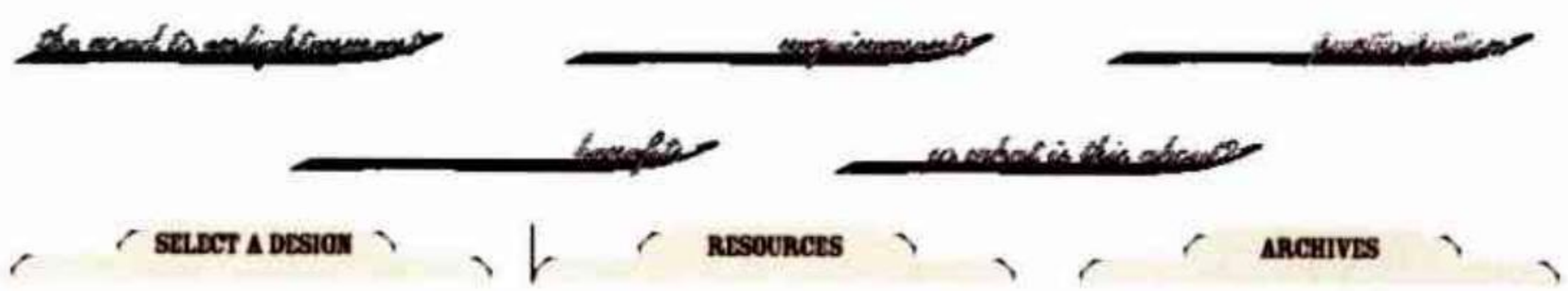


图16.7　页面中用到的标题背景图像

设置背景图像的代码如下：

```
#preamble h3 {
  background: url(h3road.gif) no-repeat right;
  }
#explanation h3 {
  background: url(h3allabout.gif) no-repeat right;
  }
#participation h3 {
  background: url(h3part.gif) no-repeat right;
  }
#benefits h3 {
  background: url(h3benefits.gif) no-repeat right;
  }
#requirements h3 {
  background: url(h3require.gif) no-repeat right;
  }
#lselect h3{
  background: url(lselectspan.gif);
  margin: -15px 0 0 0;
  }
#lresources h3 {
  background: url(lresourcesspan.gif);
  }
#larchives h3 {
  background: url(larchivesspan.gif);
  }
```

注意，linkList中最上面的一个标题通过设置margin向下移动了15像素。在linkList内部的3个div中，为了实现圆角框的效果，每一个div的最下面都有一个相同圆角图像背景，如图16.8所示。

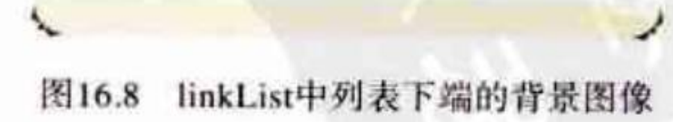


图16.8　linkList中列表下端的背景图像

需要将它放入每一个div中，代码如下。

```
#lselect,#larchives,#lresources {
```

```
background: url(lselectbttm.gif) no-repeat bottom;
padding-bottom: 12px;
margin-top: 15px;
}
```

然后需要把quickSumary中的两个文字段落移动到网页头部的图片上。显然这需要使用绝对定位。

```
#quickSummary .p2 {
  position: absolute;
  top: 190px;
  width: 280px;
  text-align: right;
  margin-left: 360px;
  }
#quickSummary .p1 {
  position: absolute;
  top: 100px;
  width: 158px;
  margin-left: 482px;
  text-align: right;
  line-height: 14px;
  }
```

请注意，这里和linkList的定位方法很相似，position属性设置为绝对定位，但是只设置top的属性值，而不设置水平方向的属性值，这样可以使得在水平方向上仍保持在原来的位置的。然后通过margin和padding来控制水平的位置。

16.1.5 设置footer

现在来设置supporttingText中最下面的footer部分，代码如下。

```
#footer {
  display: block;
  width: 625px;
  margin-left: -235px;
  text-align: center;
  padding: 15px 0 25px 0;
  font: 10px verdana,arial,serif;
  text-transform: uppercase;
  letter-spacing: -1px;
  background: #DCD5B8 url(footerbg.gif) repeat-x bottom;
  }
```

由于supportingText已经使用padding-left把左边空出来，因此为了使footer能够把宽度扩展到整个页面的最右端，这里将左侧的margin设置为-235像素，然后再配合padding，获得最终需要的效果。

到这里所有的布局工作都完成了，剩下的就是一些文字、链接文件和列表文字等局部样式的设置了，这里就不再详细介绍了。最终的效果请参见本案例最开始的效果图。

16.1.6 本案例的总结

本案例的重点有如下3点。

（1）学习整体结构的搭建，掌握灵活划分和组织页面的方法。

（2）各种使用背景图像的技巧，例如这里制作linkList的圆角框、设置整个页面的渐变背景色等。

（3）使用绝对定位时，如果只设置一个方向上的定位属性，例如只设置top，或者只设置right，则只在一个方向上使用绝对定位，另一个方向上仍然保持原来的位置。这在前面表格一章的“小视图模式日历”的案例中曾经遇到过。在那个案例中，对“日程安排”设置了绝对定位，目的是使它脱离标准流，但是并不对它设置top或left等距离属性，这样它就仍然在原来的位置。这个技巧非常有用。

16.2 《日记》（191号作品）布局方法剖析

下面分析191号作品。在上一章中已经对它的基本结构进行了宏观的分析。在本节中，我们将在代码级给出详细的剖析。读者可以参考本书光盘中的“第16章/191.htm”。

图16.9　第191号作品的效果图

16.2.1 准备图片

从前面的“简约夕阳”和这个“日记”作品中都可以看出，一个好的作品中，完美的美术设计是必不可少的，因此使用CSS制作页面时，既要兼顾艺术性，又要通过技术手段使艺术性完美地在页面中体现出来。这确实是一个很不容易的工作。

下面来看一下本案例使用的图片。页面的整体背景由如图16.10所示的3个图像构成，中间的图像在竖直方向平铺。

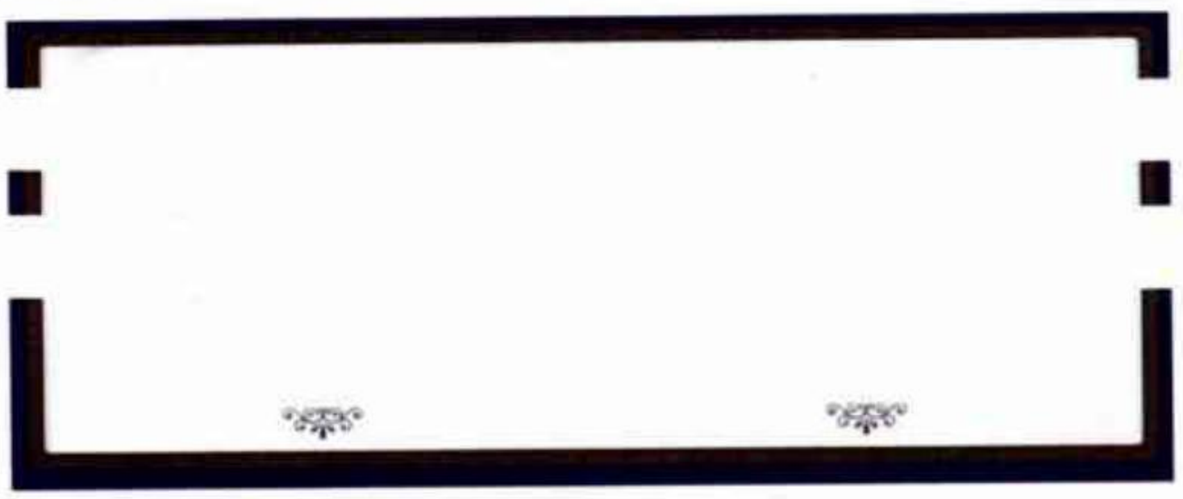


图16.10 页面背景图像

在.pageHeader部分，h1和h2这两个标题都用图像代替了，如图16.11和图16.12所示。

图16.11 h1标题的背景图像

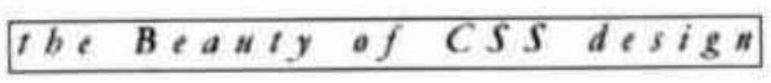


图16.12 h2标题的背景图像

在quickSummary部分，使用的背景图像如图16.13所示。

图16.13 quickSummary部分的背景图像

在preamble部分有一个h3标题，使用图片代替，如图16.14所示。

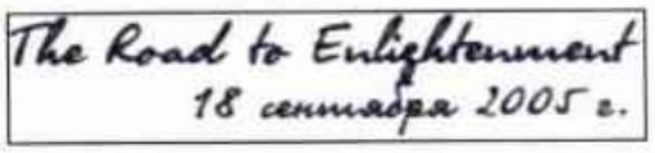


图16.14 preamble部分的h3标题的背景图像

此外，还使用了一个装饰性的图片，如图16.15所示。

在SupportingText部分，一共分为5个div，前4个div中都有一个h3标题和若干个文字段落。这4个h3标题的文字也都用图片代替了，如图16.16所示。

图16.15　preamble部分的装饰图像

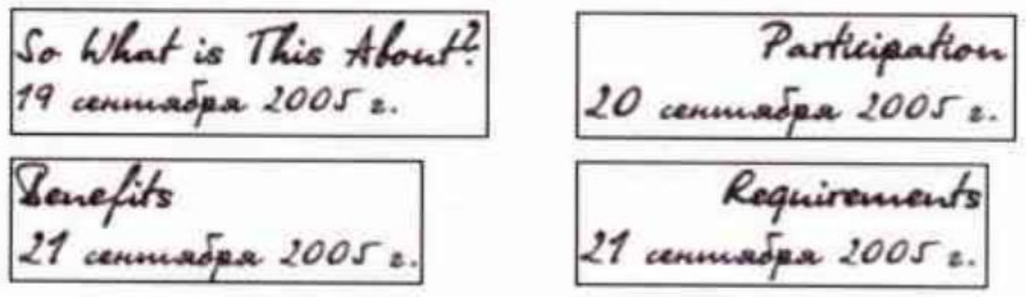


图16.16　supportingText部分h3标题的背景图片

在explaination和paticipation这两个div中各有一个装饰的图片，如图16.17所示。

图16.17　explaination和paticipation部分的装饰图片

在benefits部分，由于空间有限，使用了一个比较小的装饰图片，如图16.18所示。

图16.18　benefits部分的装饰图片

在绝对定位的linkList部分一共有3个div，各自的h3标题也都使用了图片，依次如图16.19所示。

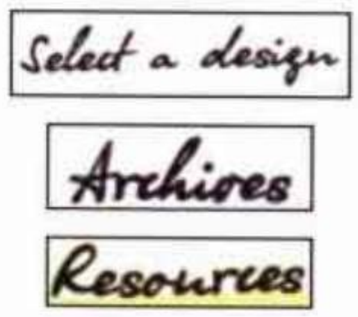


图16.19　linkList部分的标题背景图片

此外，还用了一个黄色贴纸图像作为背景，如图16.20所示。

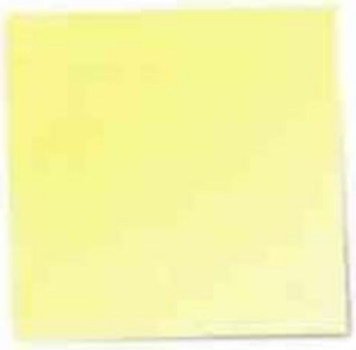


图16.20　linkList部分的贴纸装饰背景图像

16.2.2 设置页头

作为演示，这里将详细构建出这个页面，希望读者通过这个案例的学习，能够掌握分析和构建页面的方法。

① 首先对整体设置。为了将所有的标题都替换为图像，因此将它们的margin和padding均设置为0，代码如下。

```
body{
  font:11px/20px Georgia,"Times New Roman",Times,serif;
  color:#666;
  background:#00496C;
  margin:0;
  padding:0
  }
h1,h2,h3{
  margin:0;
  padding:0
  }
```

② 设置container容器的宽度为770像素，并居中。这里用的是负margin的方法，而不是本书经常使用的左右margin设置为auto的方法。这是因为早期版本的IE浏览器对margin设置为auto的支持不好，目前已经不需要使用这种方法了。代码如下。

```
#container{
  background:url("cover_bot.png") no-repeat left bottom;
  left:50%;
  position:absolute;
  width:770px;
  margin:10px 0 20px -385px;
  padding-bottom:120px
  }
```

③ 设置intro部分。这里需要注意，为了使intro内部的内容距离边界有一定空间，将左右padding设置为30像素。

```
#intro{
  padding:0 30px;
  background:url("cover.png") repeat-y;
  padding-bottom:1px
  }
```

④ 使用图像替换h1和h2标题。

```
#pageHeader{
  margin:0 -30px;
  background:url("cover_top.png") no-repeat
  }
#pageHeader h1{
  background:url("csszengarden.jpg") no-repeat;
  height:71px;
```

```
  width:268px;
  position:relative;
  top:50px;
  left:71px
  }
#pageHeader h1 span{
  display:none
  }
```

上面的代码中，第2段代码为h1标题设置了一个背景图像，并给出了标题所在的位置和大小；第二段代码将套在<h1>和</h1>之间的span的display属性设置为none，也就是标题文字被隐藏起来了，这样main上就只留下了背景图像。

与直接把图像用img标记放到HTML中相比，这样做的优点在于它可以保留HTML中的文本，保持了HTML的语义结构，对于搜索引擎来说也可以更好地理解网页的内容。

⑤ 用同样的方法设置h2标题。

```
#pageHeader h2{
  background:url("thebeauty.jpg") no-repeat;
  height:20px;
  width:299px;
  position:relative;
  top:55px;
  left:57px
  }
#pageHeader h2 span{
  display:none
  }
```

这时效果如图16.21所示。

图16.21　设置页头的效果

⑥ 设置quickSummary部分，它使用绝对定位的方法。前面的container设置为绝对定位，因此这里的quickSummary将以container为定位基准。

```
#quickSummary{
  position:absolute;
  top:41px;
  padding:0 0 0 380px
  }
```

注意，上面的代码中设置了top属性为41像素，而没有设置左右的定位属性，因此它的左侧padding仍以它原本的位置为基准，如图16.21中的红色框线和箭头所示。

⑦ 设置quickSummary中两端文本的属性。

```
#quickSummary .p1{
  margin:10px 81px 0 0;
  font-size:120%
  }
#quickSummary .p2{
  margin:0;
  background:url("bookmark.jpg") no-repeat;
  position:absolute;
  top:-25px;
  left:620px;
  width:81px;
  height:131px
  }
#quickSummary .p2 span{
  display:block;
  padding:20px 10px;
  color:#fff;
  font:11px/18px "Trebuchet MS",Arial,Helvetica,sans-serif
  }
  #quickSummary .p2 a{
  color:#fff
  }
#quickSummary .p2 a:hover{
  text-decoration:none;
  color:#ccc
  }
```

这时效果如图16.22所示。

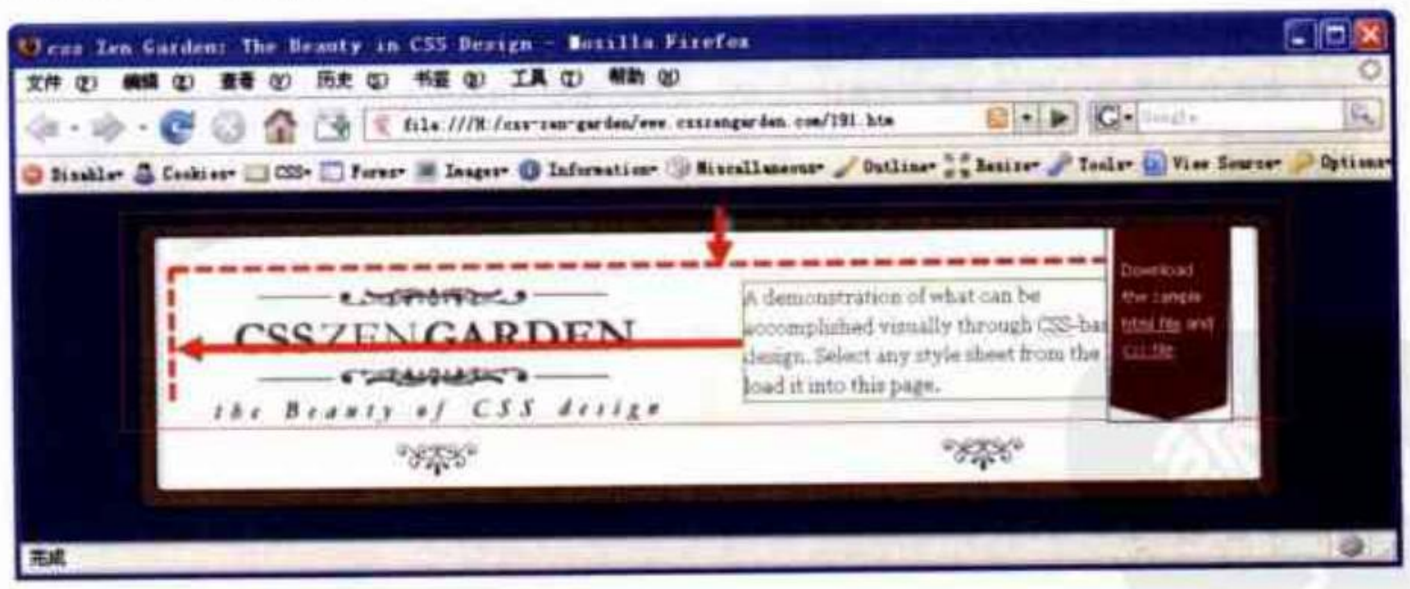


图16.22 设置quickSummary的样式

⑧ 设置preamble部分的样式，以及其中的h3标题的图像替换。

```
#preamble{
  padding-top:120px;
  background:url("img_1.jpg") 10px 170px no-repeat;
  height:1%
```

```
  }
#preamble h3{
  background:url("roadto.png") no-repeat;
  position:relative;
  left:120px;
  height:44px;
  width:212px
  }
#preamble h3 span{
  display:none
  }
#preamble p{
  text-align:left;
  margin-top:-44px;
  font-style:italic;
  padding:0 10px 35px 380px
  }
```

上面这段代码中，涉及的尺寸数据比较多，情况也比较复杂，请读者参考图16.23中标出来的数据，并仔细研究代码，来彻底搞清这段代码的含义。

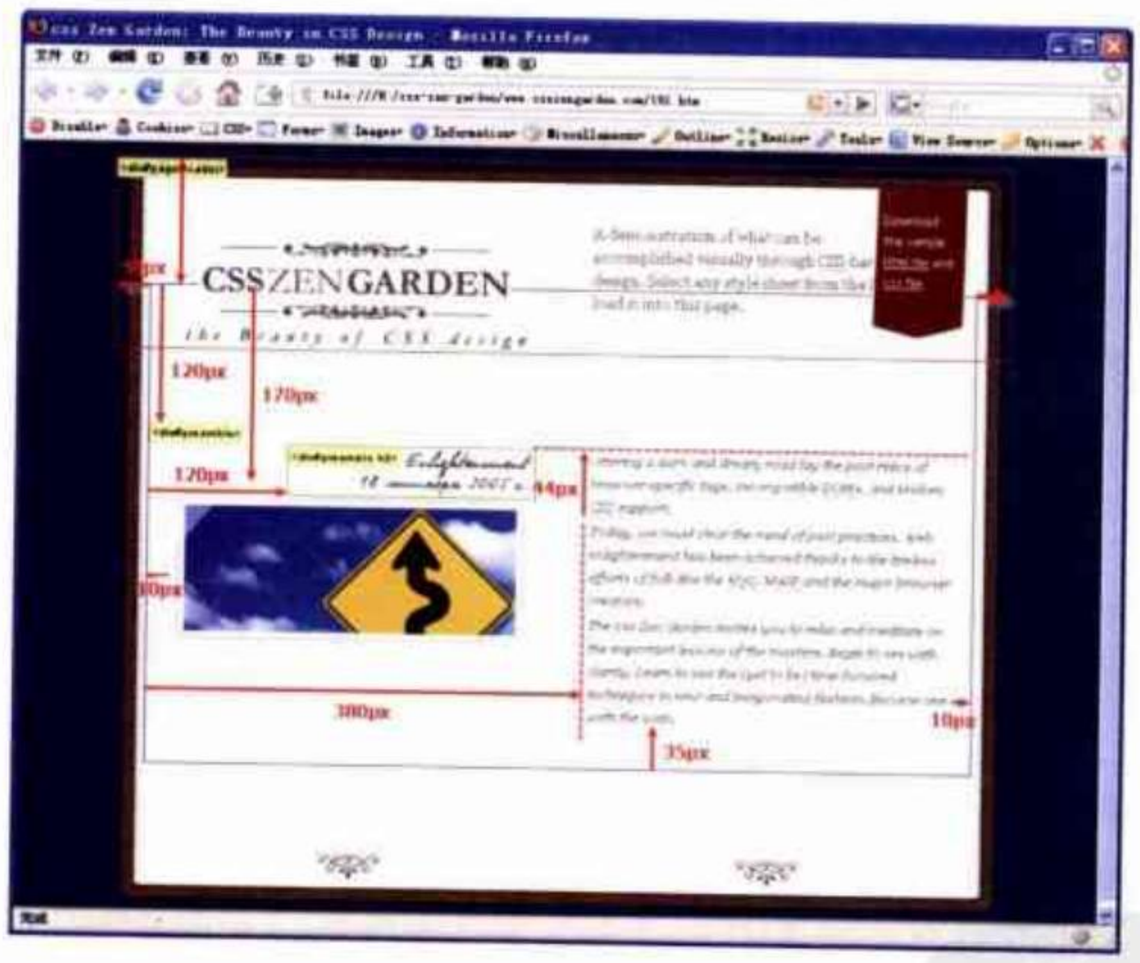


图16.23　设置preamble部分的样式

下面对图中的数据计算方法，进行一些说明。紫色方框是#preamble的范围，红色方框是#pageHeader的范围。

（1）intro设置了左右各30像素的padding，由于pageHeader需要放置背景图像，因此将pageHeader的左右margin设置为-30像素，扩展它范围，可以看到紫色框线比红色框线的左右边线都向内缩小了30像素。

（2）图中最上面的箭头的长度确定了#preamble的上边线位置，它等于#pageHeader中两个图像的高度综合，因为这两个图像使用相对定位，即位置移动到了现在的位置，但是#preamble仍然把它们当作在原来的位置看待。

（3）#preamble设置了120像素的上侧padding，因此把其中的内容整体向下推了120像素。蓝天白云的图片是作为#preamble的背景出现的，它的定位以紫色边框的左上角为基准，向下170像素，向右10像素。

（4）h3标题也使用图像代替文字，且仍在标准流中，但是通过相对定位的方式向右推了120像素，到现在的位置。

（5）剩下的3个文字段落仍然在标准流中，因此它本来会出现h3标题的下面。为了让第一个文字段落的顶部和标题图像的顶部对齐，将3个文字段落的上侧margin设置为-44像素，44像素正是h3标题图像的高度。同时通过设置文字段落的padding，把它们挤到页面右侧。

至此，整个页头部分就设置完成了。

16.2.3 设置supportingText部分

下面开始设置supportingText部分的样式，从第一个段落explanation开始。

```
#supportingText{
  background:url("cover.png") repeat-y;
  padding:0 30px
  }
#explanation{
  height:1%;
  background:url("img_2.jpg") 370px 50% no-repeat
  }
#explanation h3{
  background:url("sowhat.png") no-repeat;
  position:relative;
  left:380px;
  height:45px;
  width:176px
  }
#explanation h3 span{
  display:none
  }
#explanation p{
  text-align:right;
  margin-top:-45px;
  font-style:italic;
  padding:0 380px 35px 10px
}
```

这时效果如图16.24所示。

在#supportingText部分一共有5个div，它们都是很类似的。首先来分析#supportingText中的第一个div，即#explanation。图中红色线框表示的是#explanation的范围，绿色线框表示的是#supportingText的范围，#supportingText设置的背景图像和#intro部分相同，都是平铺实现左右边框效果的。由于#supportingText设置了左右margin各30像素，因此#explanation的左右边线都比#supportingText向内缩进了30像素。

接下来要设置#explanation的背景图像，也就是图中绿色的CSS字样的图片。它水平方向使用像素数值进行定位，距离左侧边线370像素。竖直方向使用百分比定位，设置为50%，

它的含义是背景图像竖直方向的中点与#explanation的高度方向的中点对齐，也就是实现图像正好竖直居中放置。

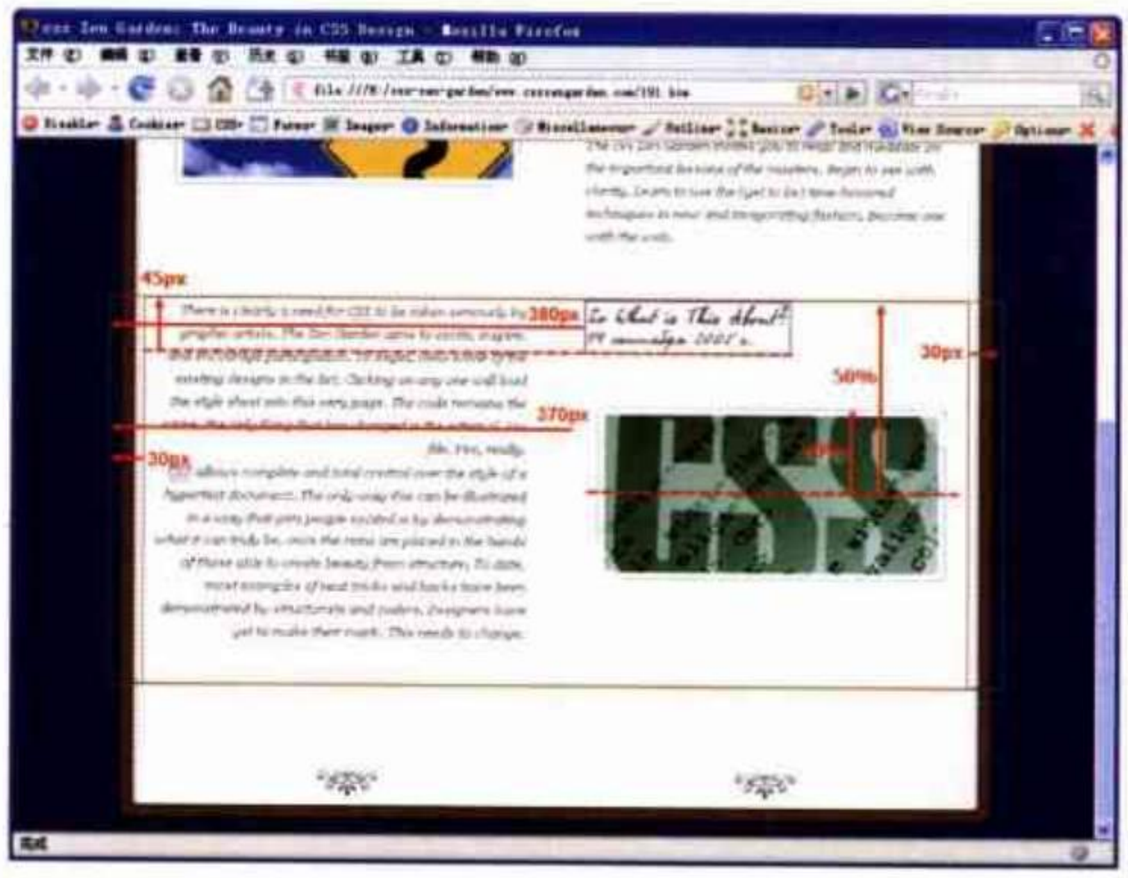


图16.24　设置supportingText部分第一段的效果

然后设置h3标题。这里同样使用背景图像代替文字，通过使用相对定位将标题图像向右推380像素，到图中所示的位置。

最后设置几个文字段落。关键的设置是使文字段落的顶端与图像的顶端对齐，因此将文字段落的margin设置为负值，从而把文字段落向上提升，提升的距离正好就是标题图像的高度44像素。此外通过设置文字段落的padding，把文字限制在页面的左侧，并保持好距离。

至此#supportingText部分的#explanation就设置好了。接下来的#participation、#bebefits和#requirements这3个div的设置与#explanation几乎完全一样，因此就不再赘述了。

#supportingText部分的最后一个div是#footer，它的设置很简单，只需设置几个链接样式即可，因此也不再详细讲解了。#supportingText设置完成后，效果如图16.25所示。

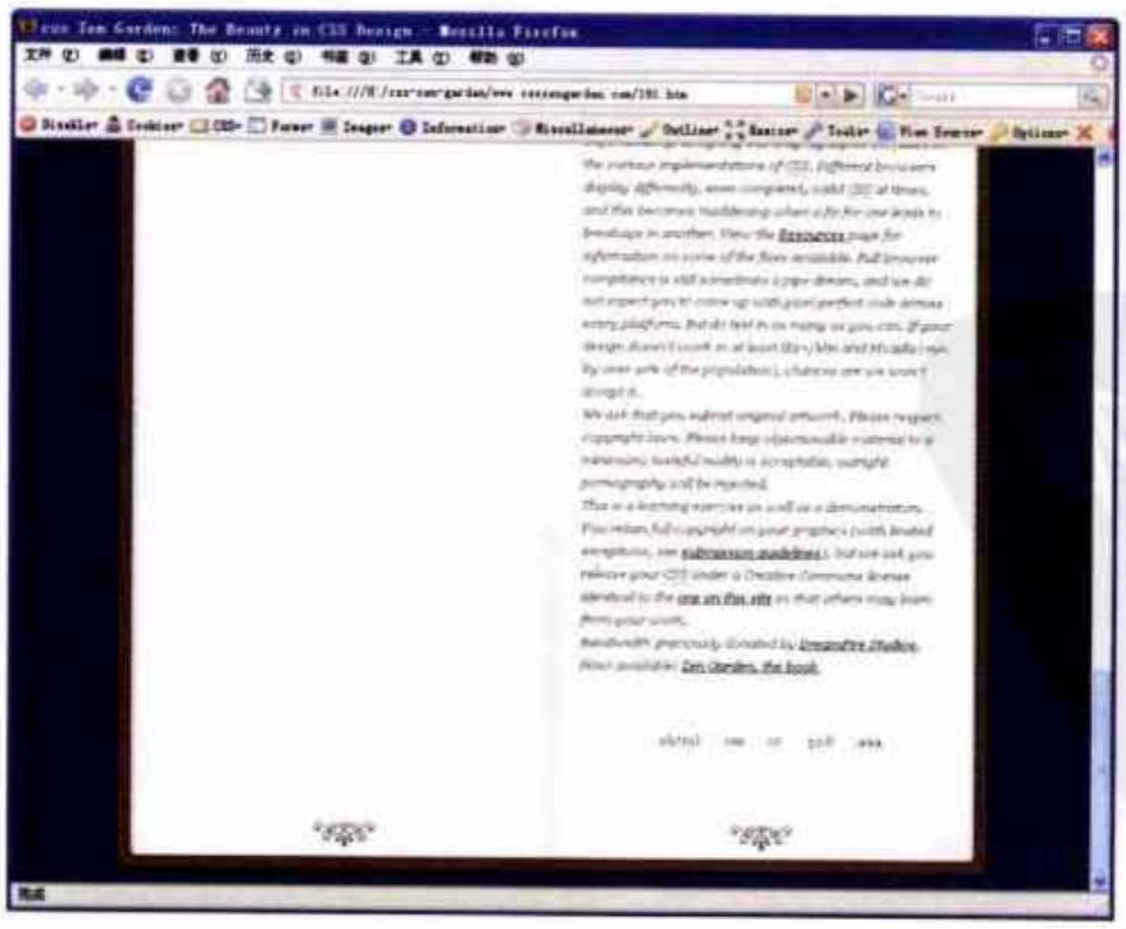


图16.25　设置supportingText部分后的效果

可以看到左下角有一大块空间，正好用来放置页面的第3部分内容#linkList。

16.2.4 设置linkList部分

下面开始设置linkList部分。首先设置最上面的select列表，代码如下。

```
#linkList{
  font-size:120%;
  font-style:italic;
  position:absolute;
  width:300px;
  bottom:70px;
  left:80px
  }
#linkList h3.select{
  width:142px;
  height:29px;
  background:url("select.png") no-repeat;
  position:relative;
  left:8px;
  top:8px
  }
#linkList h3.select span{
  display:none
  }
#linkList ul{
  margin:10px 5px 10px 40px;
  padding:0 5px
  }
#linkList ul li{
  list-style-type:none;
  list-style-image:none
  }
#lselect ul li{
  list-style:outside url("bull.png")
  }
#linkList ul li a{
  text-decoration:none
  }
```

这时效果如图16.26所示。请读者按照分析前面两个大部分的方法来分析这个部分的数据计算方法，可以参考图16.26所示的尺寸数据。

使用相同的方法设置第2个列表以后，效果如图16.27所示。

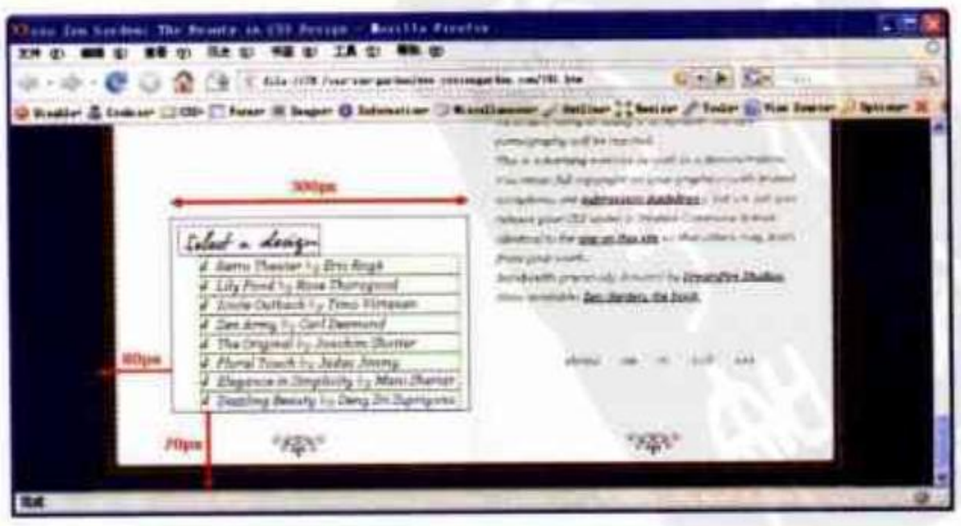


图16.26 设置linkList部分的第1个列表的效果

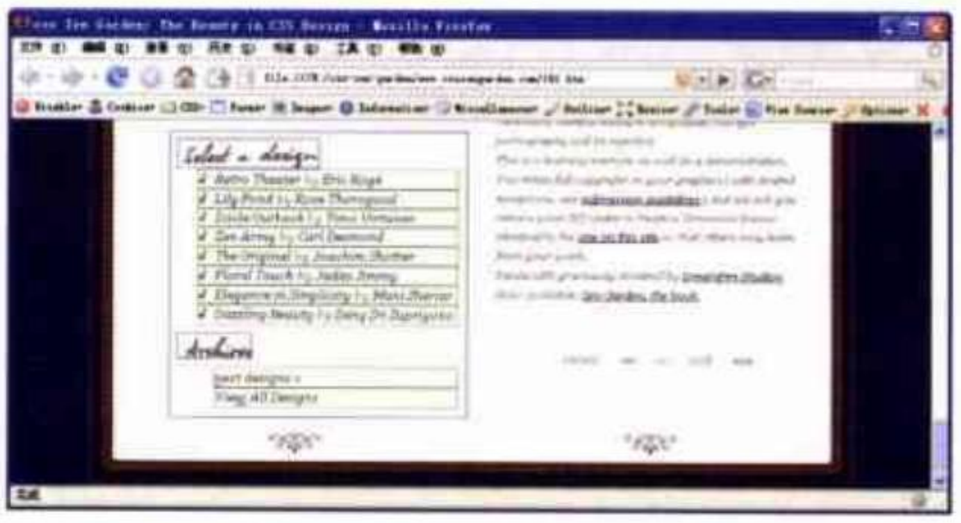


图16.27　设置linkList部分的第2个列表的效果

最后，设置linkList部分的第3个列表，代码如下。

```
#lresources{
  font:16px/22px "Trebuchet MS",Arial,Helvetica,sans-serif;
  background:url("notes.jpg") no-repeat;
  padding:20px 15px;
  width:252px
  }
#lresources ul{
  margin:30px   20px 50px 25px
  }
#lresources h3.resources{
  width:85px;
  height:22px;
  background:url("resources.png") no-repeat
  }
#lresources h3.resources span{
  display:none
  }
```

这时效果如图16.28所示，请读者自己分析其数据尺寸，然后和图16.28中的数据对比一下。

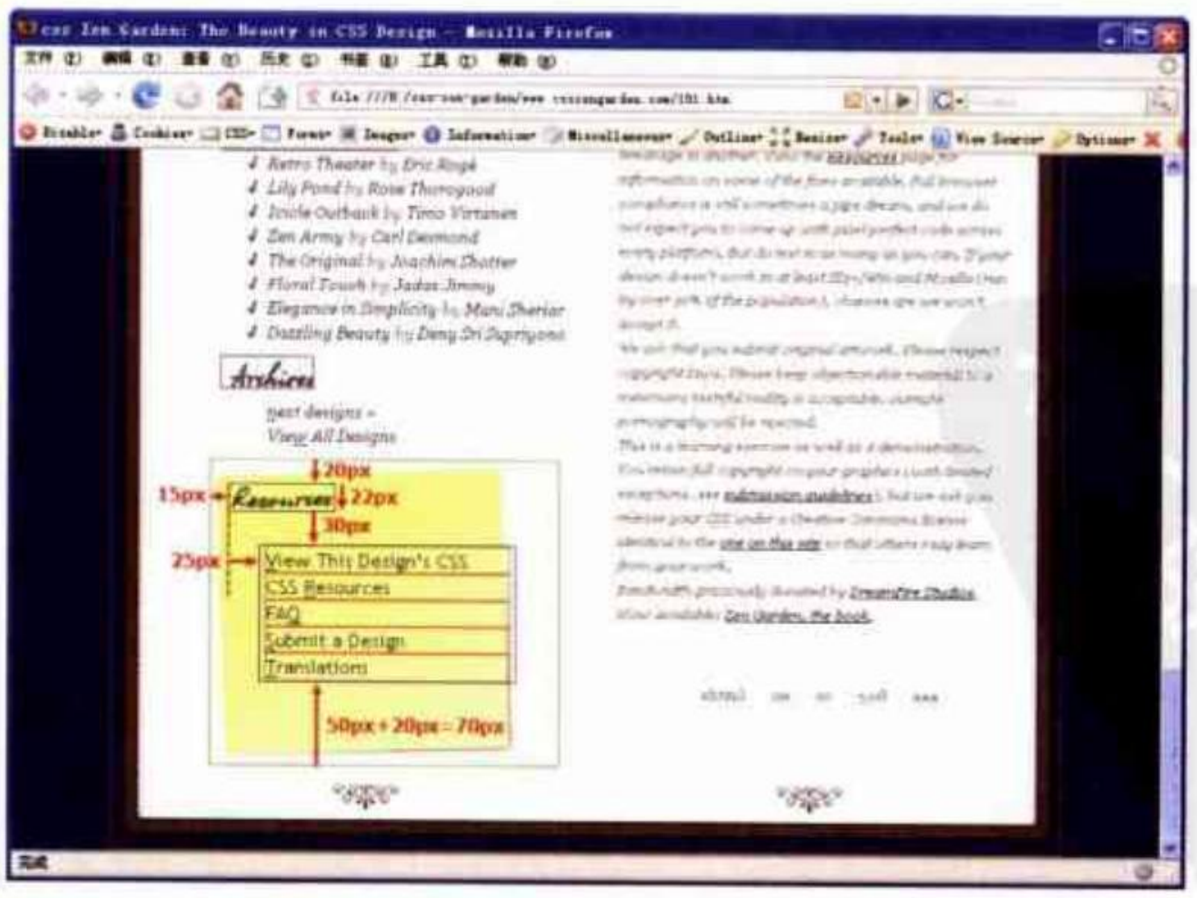


图16.28　设置linkList部分的第3个列表的效果

16.2.5 本案例总结

本案例使用了比较多的定位数据，因此计算起来比较复杂。

此外，如果比较本案例和《简约夕阳》这个作品的设置方法，就会发现二者存在着明显的差别。

《日记》作品中，设计师是按照页面内容的顺序，从上向下，一个部分一个部分地依次设置；而《简约夕阳》这个作品中，设计师是把相似的成分放到一起来设置，例如先把几个大块分别放好，然后把所有的标题设好，再设置所有列表文字，等等。

这就好像我们在日常生活中装修一个三居室，一种思路是先把每个屋子的墙面刷好，然后把每间屋子的地面抹平，再把每间屋子的地板铺好；而另一种思路是，先把一间屋子装修好之后，再去装修另一间。

这两种思路和工作流程，对于CSS的页面设计来说，都是可以的，同样可以做出非常漂亮的作品。总而言之，无论采用哪种思路，都需要设计师思路清楚、条理明晰。一个作品中可能需要设置几百条CSS样式，如果不能做到思路明确，就很可能会造成混乱，大大降低工作的效率，影响最终的效果。

16.3 本章小结

在本章中，分析了两个案例的全部CSS样式代码，从中学到了很多有用的技巧和方法。学习CSS就像练字，练字通常都是通过“临帖”开始，逐步领会其中的精妙之处，然后再自成一体，自由发挥。学习CSS也是一个道理，首先可以多研究一些成功的页面，然后再创作出自己的作品，不断追求超越前人的境界。

17

Chapter

第17章 从学习到创作

在本章中，将制作一个自己的禅意花园作品。要制作一个比较像样的网页，是离不开美工的配合的，因此网站的设计与开发二者是不可分割的。在本书中，尽可能不涉及过多的美工技术，以保证CSS技术的单纯性，也使得学习本书更容易。但是如果要做一个真正的网页，则不可能离开美工而实现。要想让网页漂亮，就必须有好的图像素材。最基础的图像素材无外乎两种：一种是绘制出来的；另一种是拍摄出来的，然后再进行加工，适应网页的需要。本章的案例就是基于实际拍摄的照片进行设计的。

秋天是一年中最美的季节，本案例中实现的禅意花园作品以秋天为主题，效果如图17.1所示。按照禅意花园网站的习惯，每个作品都应该有一个名字，这个案例也应该有一个名字，因此就称为“秋意”了。

这个方案以15.2节“亲自动手”中完成的案例为基础，加上适当的图像文件得到的效果。读者在学习这个案例前，不妨先复习一下那个案例。

下面就来讲解这个方案的设计思路和制作步骤。本案例的最终效果请参考本书光盘“第17章/1/9998.htm”。

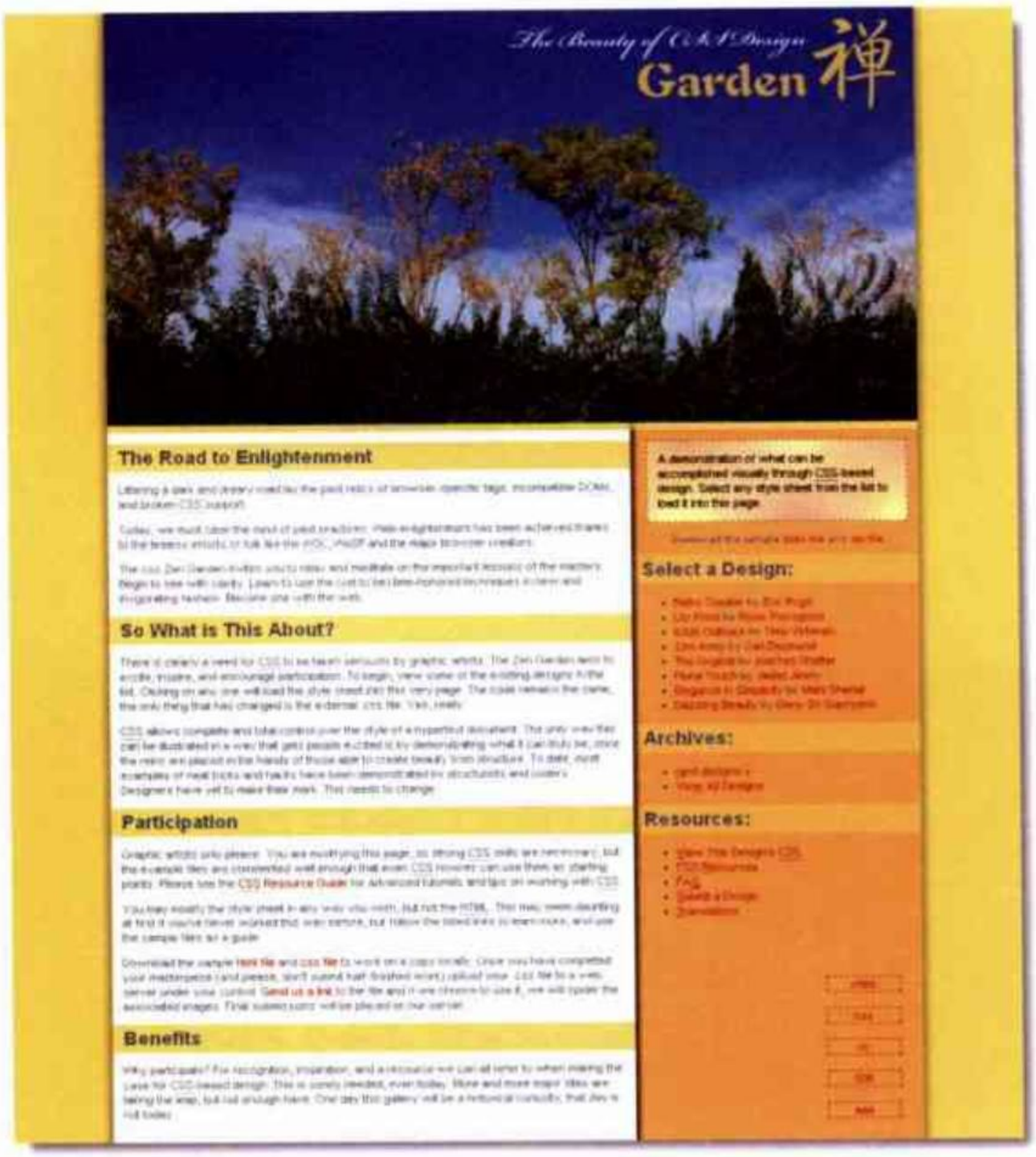


图17.1 自己动手制作的禅意花园作品——“秋意”

17.1 拍摄素材照片

这是一个以金色秋天为主题的设计方案。首先需要准备一幅图像作为标题图片，如果能够自己拍摄一幅照片，当然是最好的。

构思这个案例时正是秋季，秋高气爽，而且当天阳光和空气的透明度都非常好，因此可以到很多地方去寻找一些景色拍摄下来。如果远一些的地方可以去拍红叶、拍长城，如果不去远的地方，就可以拍附近公园的景色，例如本案例的照片就是如此。下面选几张照片来说明一些拍摄时的注意事项。

（1）近两年数码单反（Digital Single Lens Reflex）相机的价格大幅下降，如果有条件，购买一台数码单反相机，稍作训练，拍摄的效果会比使用小数码相机有明显提高。

（2）通常网页上部的标题图像都是高度比较矮，而宽度比较宽的图像。因此，在拍摄时最好使用比较大的广角镜头，可以拍出比较宽阔的效果，例如图17.2所示的照片，就是使用17mm的广角镜头拍摄的，上下都有比较大的剪裁余地。

图17.2　使用广角镜头拍摄的照片

将图17.2的上下各剪裁掉一部分，然后加上一个细线边框和标题文字，这个标题图像就可以用于网页了，效果如图17.3所示。

图17.3　将照片加工为标题图像

（3）在拍摄风景主题的时候，一定要使用小光圈拍摄，这样才能保证尽可能大的景深，提高清晰度，如图17.4所示。

图17.4　使用小光圈拍摄大景深的照片

（4）如果希望突出主体，可以使用大光圈，使其前后景物虚化，如图17.5所示。但是不要盲目追求虚化效果，适当即可。

图17.5　大光圈拍摄小景深的照片

（5）构图不需要复杂，并要考虑在网页中使用时的剪裁需要。例如图17.6所示，在拍摄时，可以偏上一些取景，这样蓝天的效果会比较好，使用时把最上面不需要的部分剪裁掉即可。

图17.6 简单的构图效果

在制作这个案例时，选用了图17.6所示的照片作为标题图片。

17.2 在图像软件中设计方案

有了标题图形的照片素材，就可以在图像软件中设计方案了。首选的是Abode公司的Fireworks和Photoshop这两款软件。Photoshop的功能更强大，而Fireworks做网页相关设计有一些非常方便的工具，因此可以根据个人的习惯选择使用哪个软件。

这里以使用Fireworks为例，简单讲解一下设计的基本过程。

① 首先确定主色调。这张照片中主要有3种颜色，上部天的蓝色、下部树的绿色，以及中部树的黄色。

为了体现秋天的主题，可以将页面的色调设置为金黄色，黄色与蓝色搭配，是一个兴奋度比较高的颜色搭配方案。

② 在Fireworks软件中，创建一个780像素宽，600像素高的文档，背景色为#FFDB4A，如图17.7所示。

图17.7 黄色背景的画布

③ 在图像的中间绘制一个白色矩形，高度与整个背景相同，宽度为760像素，即左右各留出10像素，对这个矩形使用发光滤镜，使两侧有渐变的效果，如图17.8所示。

④ 在左侧绘制一个宽度为490像素的矩形，并使用投影滤镜，使它产生一个灰色的阴影，如图17.9所示，用来区分左右两列。

图17.8　绘制中间内容区域的白色

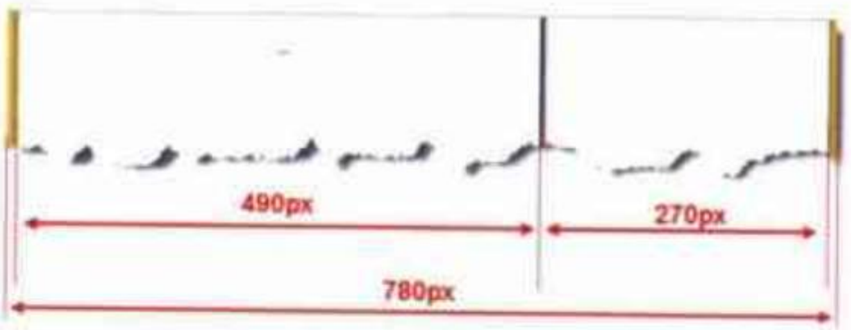


图17.9　绘制分隔线

⑤ 依次绘制其他部分，并导入照片，效果如图17.10所示。由于本书的内容和篇幅限制，这里不详细讲解绘制的步骤了。如果读者不会使用Fireworks或者Photoshop制作这个图像，可以直接使用光盘中提供的文件。

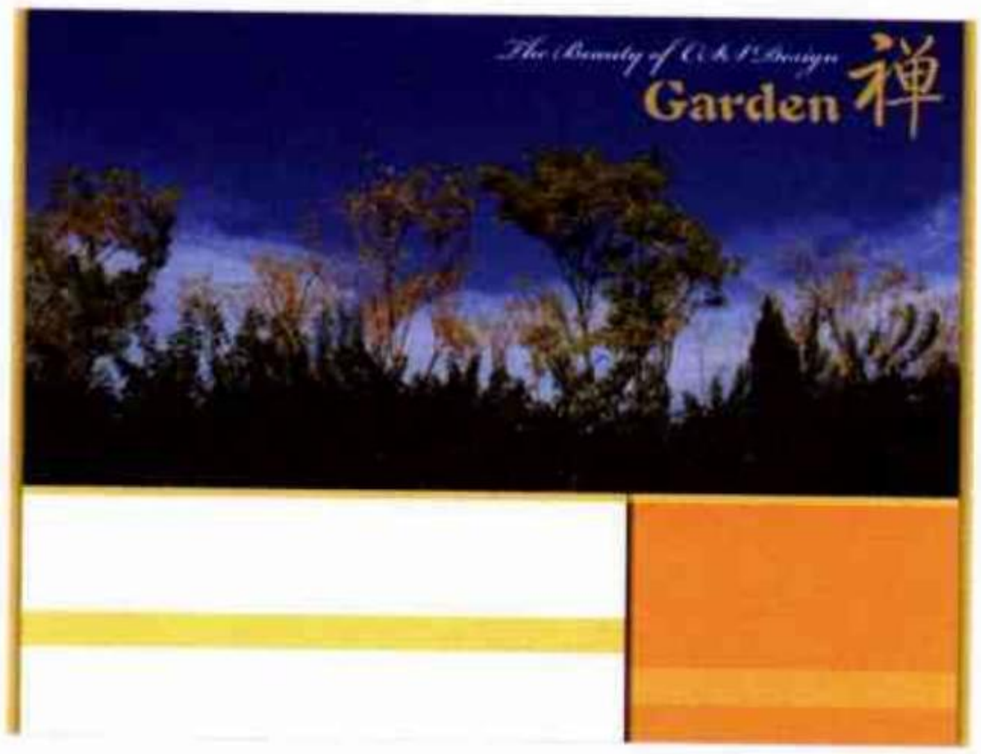


图17.10　完成后的设计图

⑥ 使用Fireworks的切片导出功能，从这幅图像中导出4个最终需要用在网页上的图像文件，如图17.11所示。

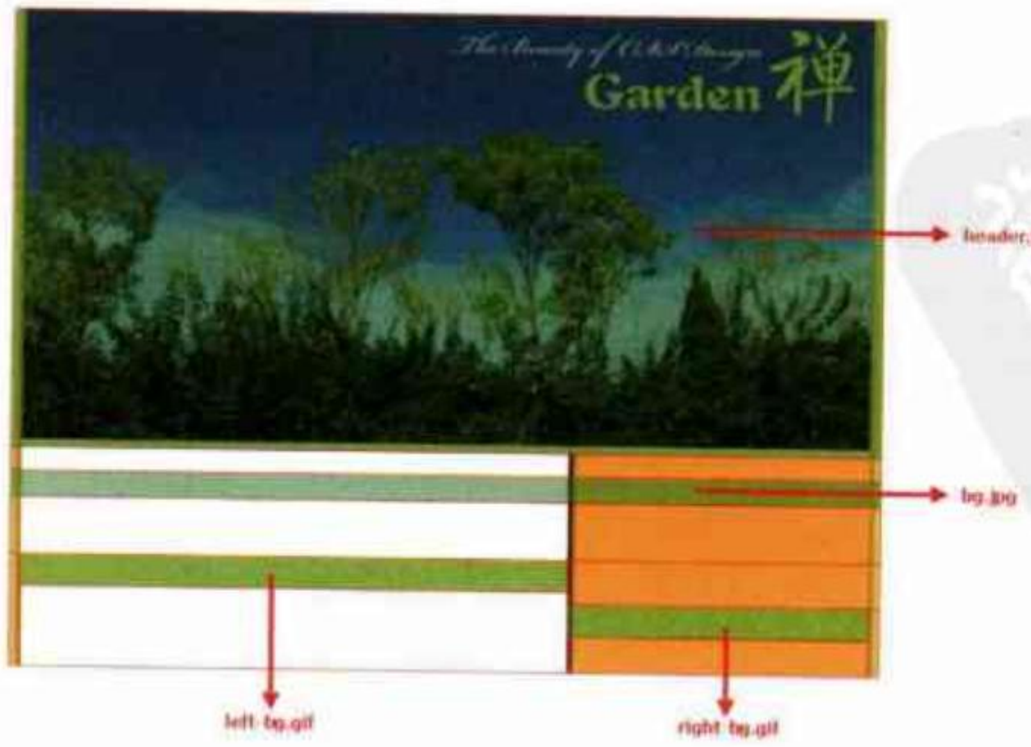


图17.11　导出4个图像文件

⑦ 这4幅图像依次如图17.12至图17.15所示。

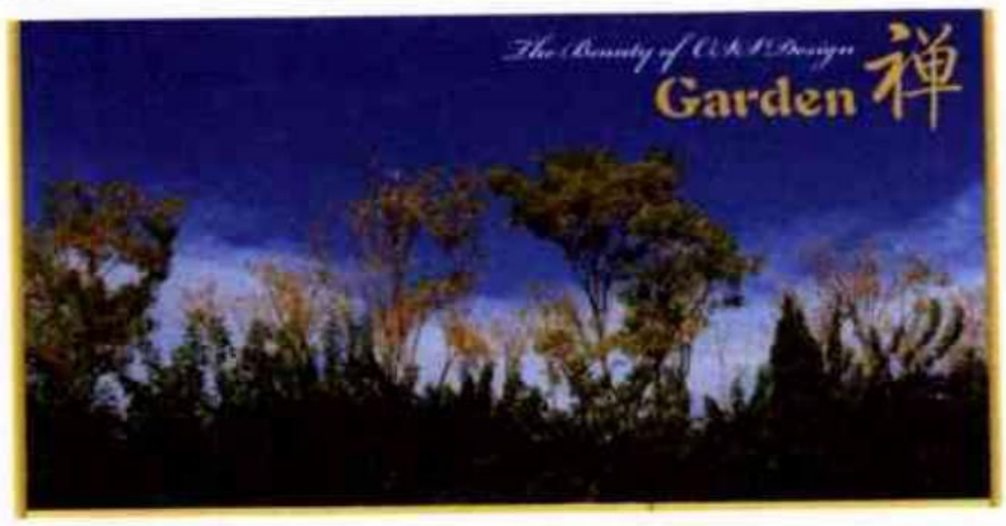


图17.12 header.gif用于标题的背景图像

图17.13 Bg.jpg用于整页的背景图像

图17.14 right-bg.gif用于右侧列标题的背景图

图17.15 left-bg.gif用于左侧列标题的背景

17.3 整体的结构分析

在开始编写CSS代码之前，需要对整体结构进行分析，以确定布局和定位方式。HTML文件仍然使用禅意花园提供的标准HTML文件。

在container中，3大部分为intro、supportingText和linkList。首先需要确定结构和定位方式，如图17.16所示。

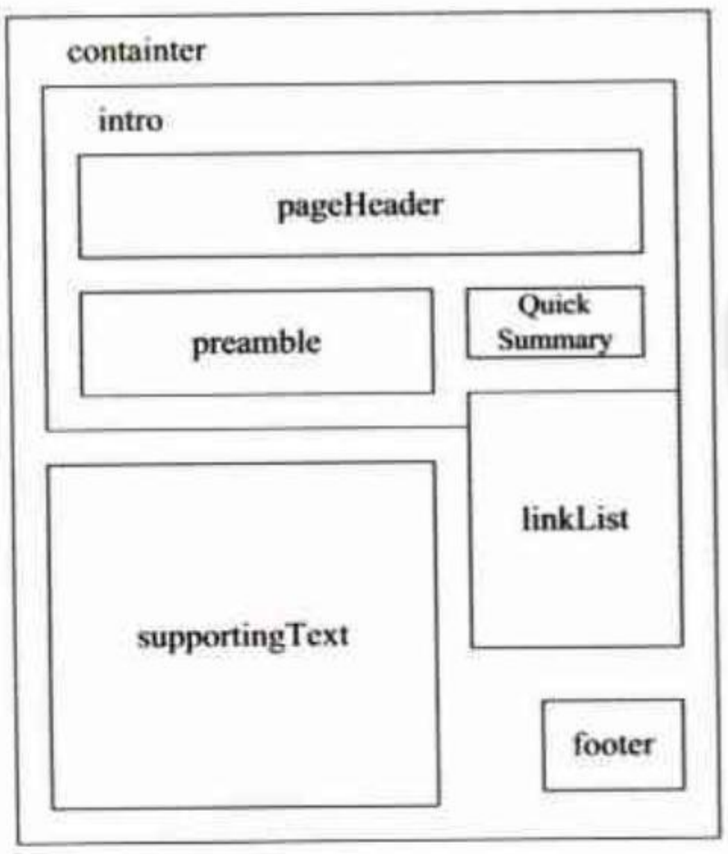


图17.16 结构示意图

下面具体解释一下各个部分的布局技术要点。

- Intro部分在最上面，其中pageHeader在intro部分的最上面，然后使quickSummary部分向右浮动，固定宽度为270像素，右侧与边界有10像素外边距。reamble使用标准流，向下排列。
- supportingText部分使用标准流，最下面的footer使用绝对定位，放置到页面的右下角。
- LinkList使用绝对定位，放置到页面右侧。

17.2节中制作了4幅背景图像，应用于下面各个部分中。

- pageHeader使用header.jpg作为背景。
- container使用bg.jpg作为背景，竖直平铺。
- 页面中左侧列（preamble和supportingText）中的h3标题使用left-bg.gif作为背景图像。
- 右侧列（linkList）的h3标题使用right-bg.gif作为背景图像。

通过计算以后，可以绘制出详细的方案规划图，如图17.17所示。

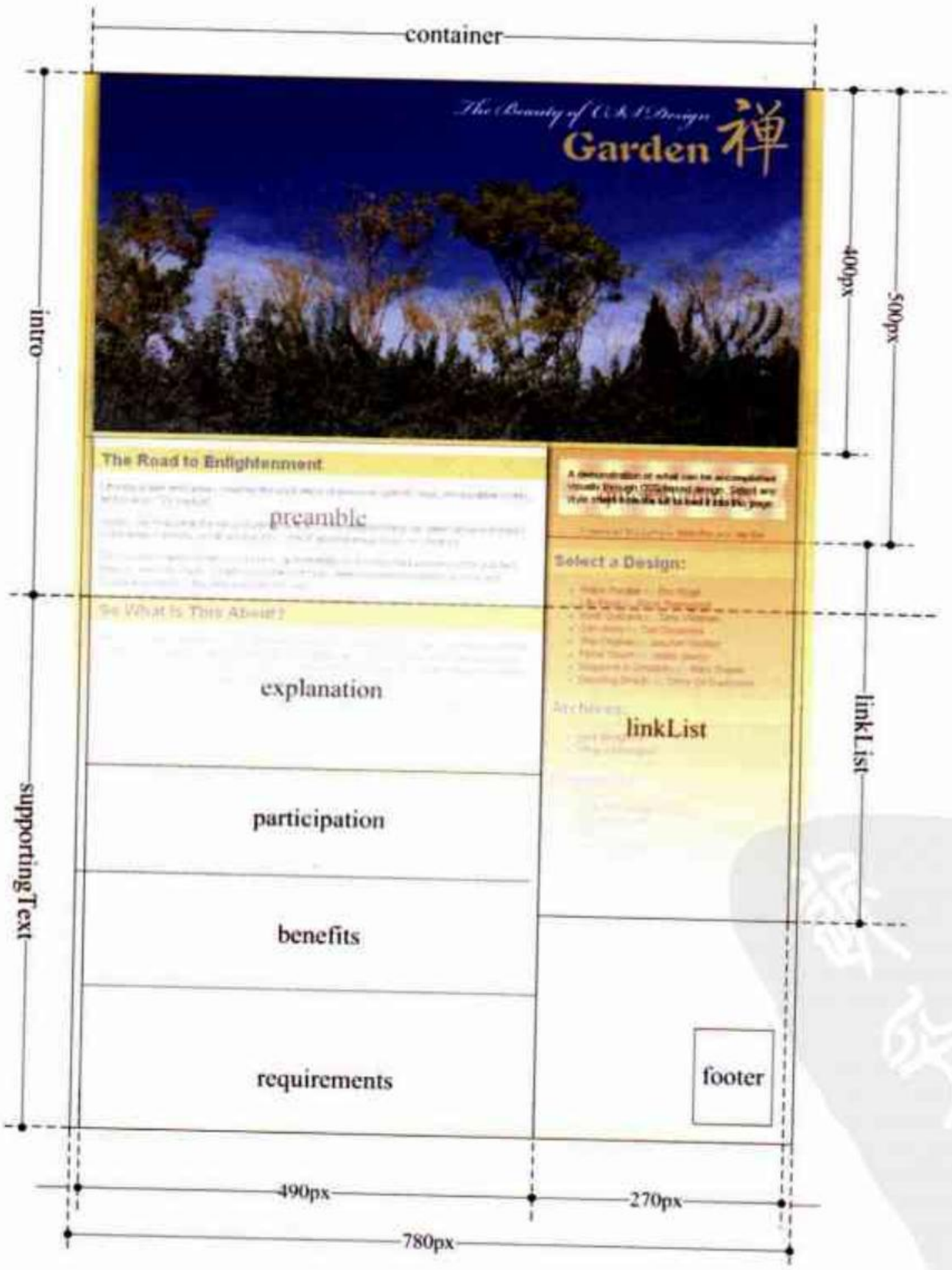


图17.17　详细结构图

17.4 CSS样式设计与编码

① 对页面整体进行设置，margin设置为0，消除内容与浏览器边框之间的空白，在body中设置黄色的背景色。

```
body{
  margin: 0;
  font-family : Arial, Helvetica, sans-serif;
  font-size : 11px;
  color: #557788;
  background-color: #FFDB4A;
}
```

② 设置container容器的宽度为780像素，通过margin使其水平居中，背景是用图像平铺的方式获得的。注意padding部分，底部设置10像素的内边距，这时的效果如图17.18所示。

```
#container {
  background:url(bg.jpg);
  width: 780px;
  margin: 0 auto;
  padding: 0 0 10px 0;
  }
```

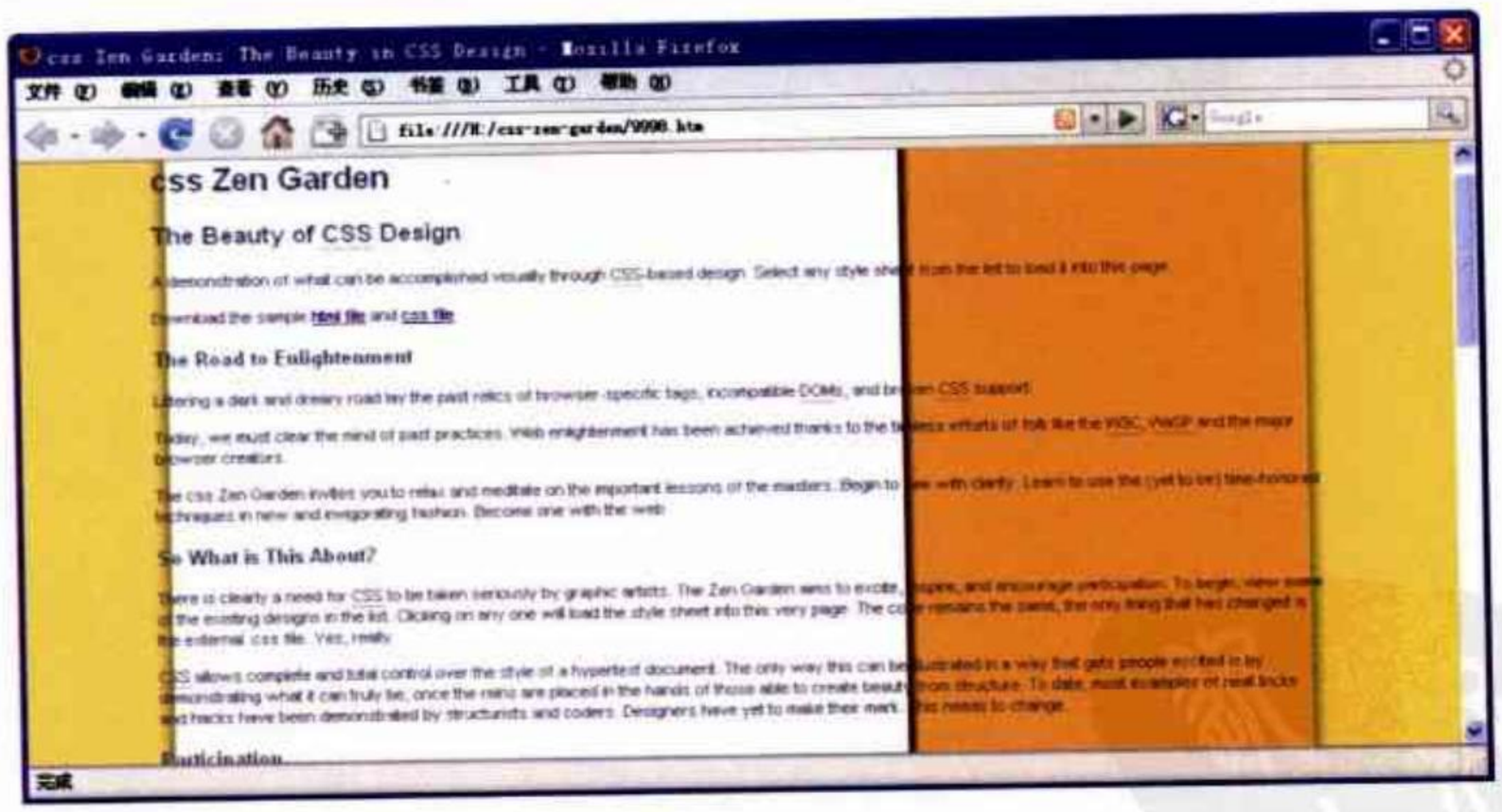


图17.18 设置整体背景图像

③ 设置pageHeader部分。设置背景图像，并设置高度，然后将两个标题都通过display属性隐藏起来。这时的效果如图17.19所示。

```
#pageHeader {
  background:url(header.jpg) ;
  height: 400px;
```

```
  }
h1,h2 {
  display:none;
  }
```

④ 设置quickSummary部分。它使用向右浮动方式，宽度设置为265像素，右侧的margin设置为10像素，使它离开右边界一定距离。

```
#quickSummary {
float: right;
width: 265px;
margin-right:10px;
}
```

⑤ 对quickSummary中的两个文本段落分别设置样式。第一个段落设置虚线的边框，并通过padding和margin使这个段落的边框内外都留有一定的空白，不至于显得很拥挤。这个段落范围内使用的背景图像文件trans.gif是一个白色像素和透明像素间隔图像，可以造成半透明的效果。

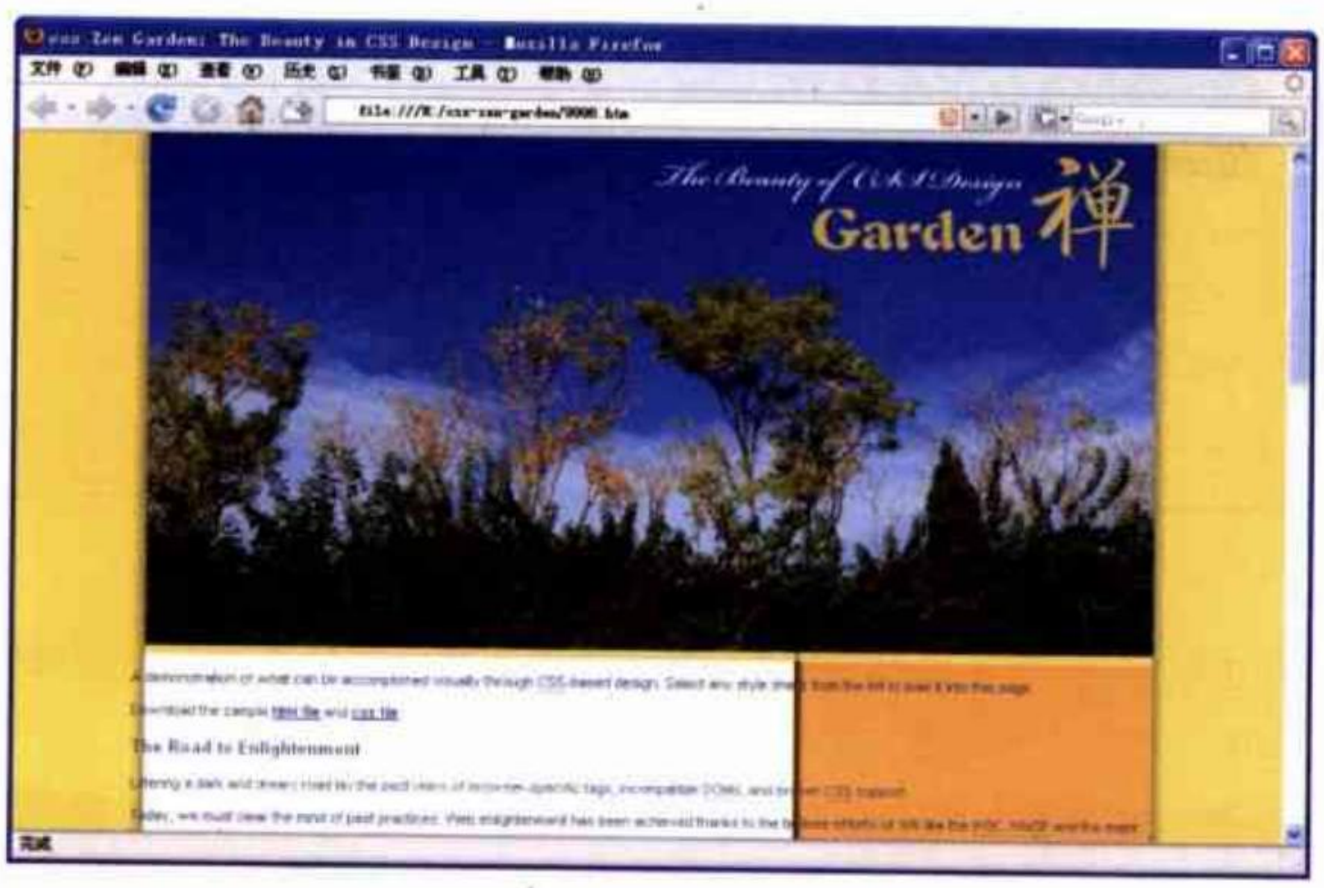


图17.19　设置标题图像

```
#quickSummary .p1 {
    color:#000;
 background: url(trans.gif);
 border:1px #578 dashed;
 padding: 10px;
 margin: 10px;
}
#quickSummary .p2 {
  font-size : 10px;
  text-align:center;
}
```

这时的效果在Firefox和IE 6分别如图17.20和图17.21所示。

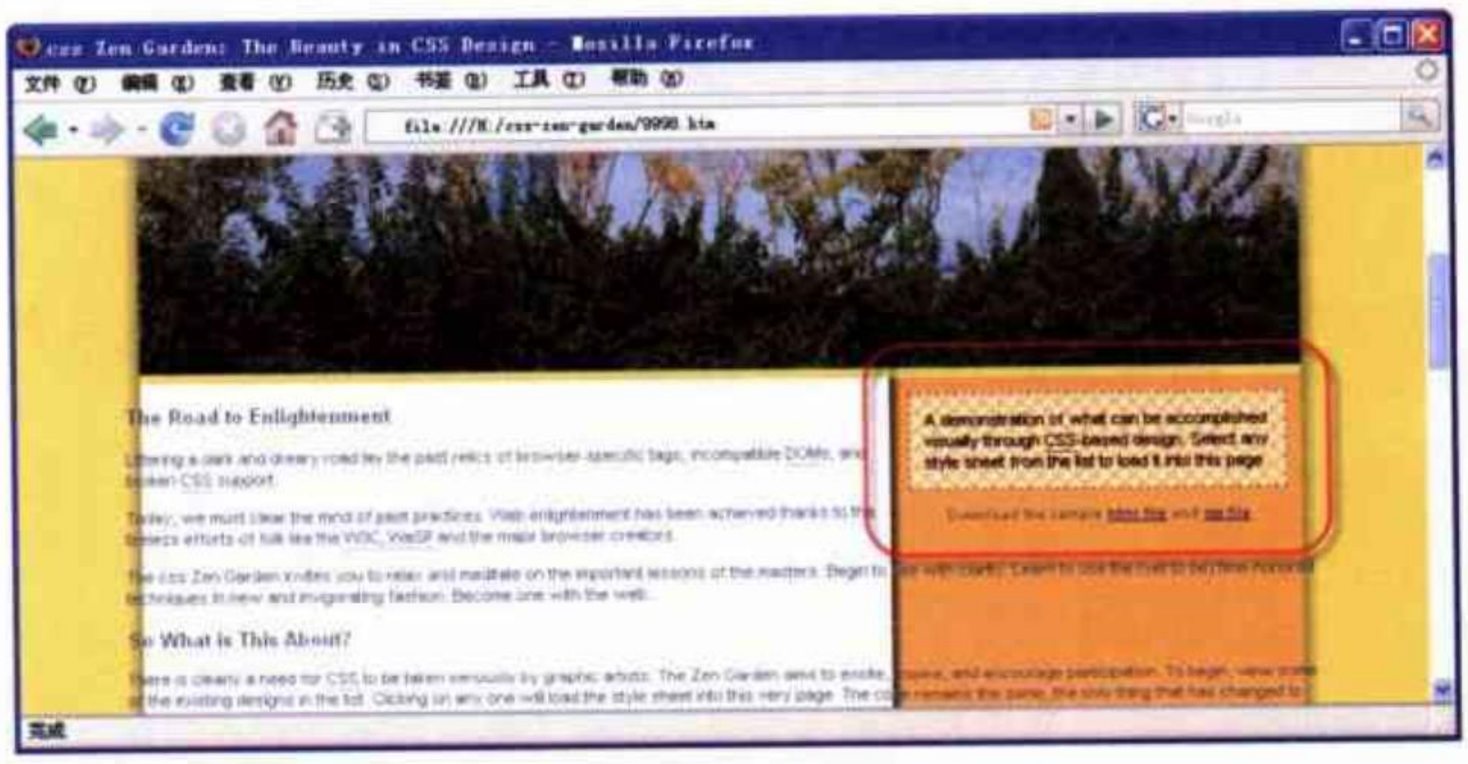


图17.20 在Firefox中quickSummary的效果

图17.21 在IE 6中quickSummary的效果

可以发现在IE中右侧的margin是Firefox中的两倍。这两种效果哪个是正确的？如果不好判断，不妨再在IE 7中看一下效果，如图17.22所示。

图17.22 在IE 7中quickSummary的效果

这就很清楚了，在IE 6中显示的效果有问题，这是IE 6的一个Bug，而在IE 7中已经得到修复。

经验　这是一个有用的技巧。如果在IE 6中看到的某个效果不是预期的效果，就到IE 7中去看一看。如果和IE 6中的效果相同，那么基本可以断定是代码有问题；如果在IE 7中的效果符合预期，而在IE 6中不符合预期，那么很可能是IE 6的错误。

这里遇到这个错误是IE 6的“浮动元素margin加倍”Bug。也就是一个浮动的元素，如果设置了margin，那么在IE 6中会按照设置值的两倍显示，正如上面quickSummary中看到的那样，解决方法是对这个浮动元素增加如下的设置：

```
display:inline;
```

不要问这是为什么，可能只有IE的开发人员才知道其中的原因，但是这种方法确实有效。因此，quickSummary的设置在上面的基础上改为：

```
#quickSummary {
  float: right;
  width: 265px;
  margin-right:10px;
  display:inline;
}
```

这样，在IE 6、IE 7和Firefox中就都可以正确显示了。

⑥ 接下来设置preamble部分，使用标准流，因此不需要额外设置浮动和定位属性。通过设置宽度，将它限制在页面左侧，并设置左边的margin为10像素。

```
#preamble {
  width: 490px;
  margin-left:10px;
}
```

⑦ 设置h3标题的样式，把背景设置为前面已经做好的背景图像left-bg.gif。

```
h3 {
  font-size: 18px;
  margin: 10px 0px 10px 0px;
  line-height:27px;
  padding-left:10px;
  background: url(left-bg.gif) no-repeat;
}
```

⑧ 文字段落的两侧各留10像素的空白，因此将p标记的margin设置为10像素。这时的效果如图17.23所示。

```
p{
  margin:10px;
}
```

注意　preamble的左margin设置的是图中白色区域左边阴影部分的10像素，p的margin设置的是白色区域内，文字左右两侧的空白的10像素，不要混淆。

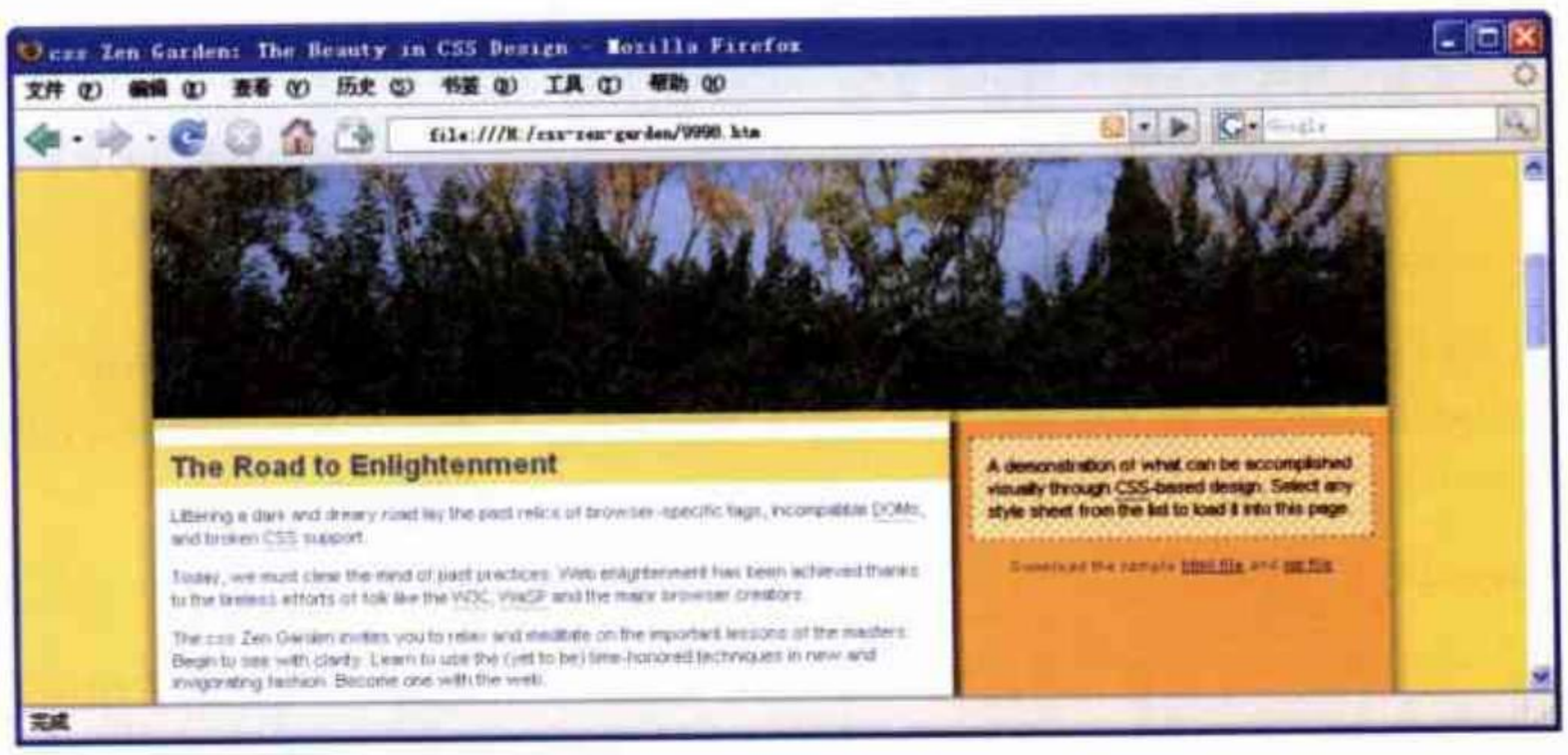


图17.23 设置preamble部分的样式

⑨ 设置supportingText部分的样式，和preamble完全相同。另外，由于h3标题在preamble部分也设置好了，因此也不需要再设置，效果如图17.24所示。

```
#supportingText{
  margin-left:10px;
  width:490px;
}
```

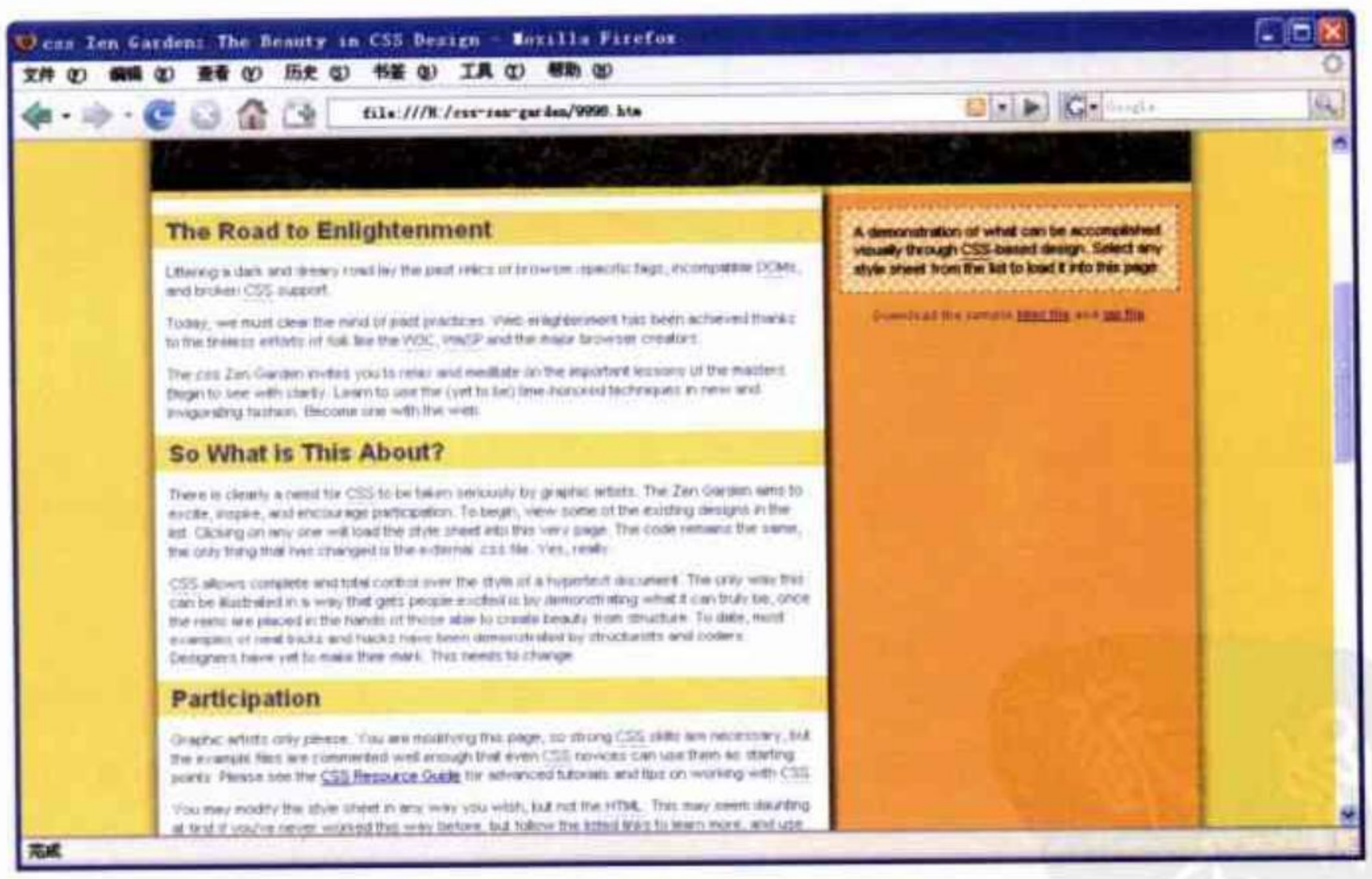


图17.24 设置supportingText部分的样式

⑩ 设置linkList部分，使用绝对定位方式，把它放置到页面右侧。linkList里面的h3标题不能使用左侧的h3标题的背景图像，因此需要对它进行设置。此外，还有重要的一点是，linkList是相当于container定位的，因此要使contatiner成为它的包含块，就需要设置container的定位属性。相关代码如下，此时的效果如图17.25所示。

```
#linkList {
  position:absolute;
  right:10px;
  top:500px;
  width: 270px;
  }
#linkList h3{
  background: url(right-bg.gif) no-repeat;
  }
#container{
position:relative;
}
```

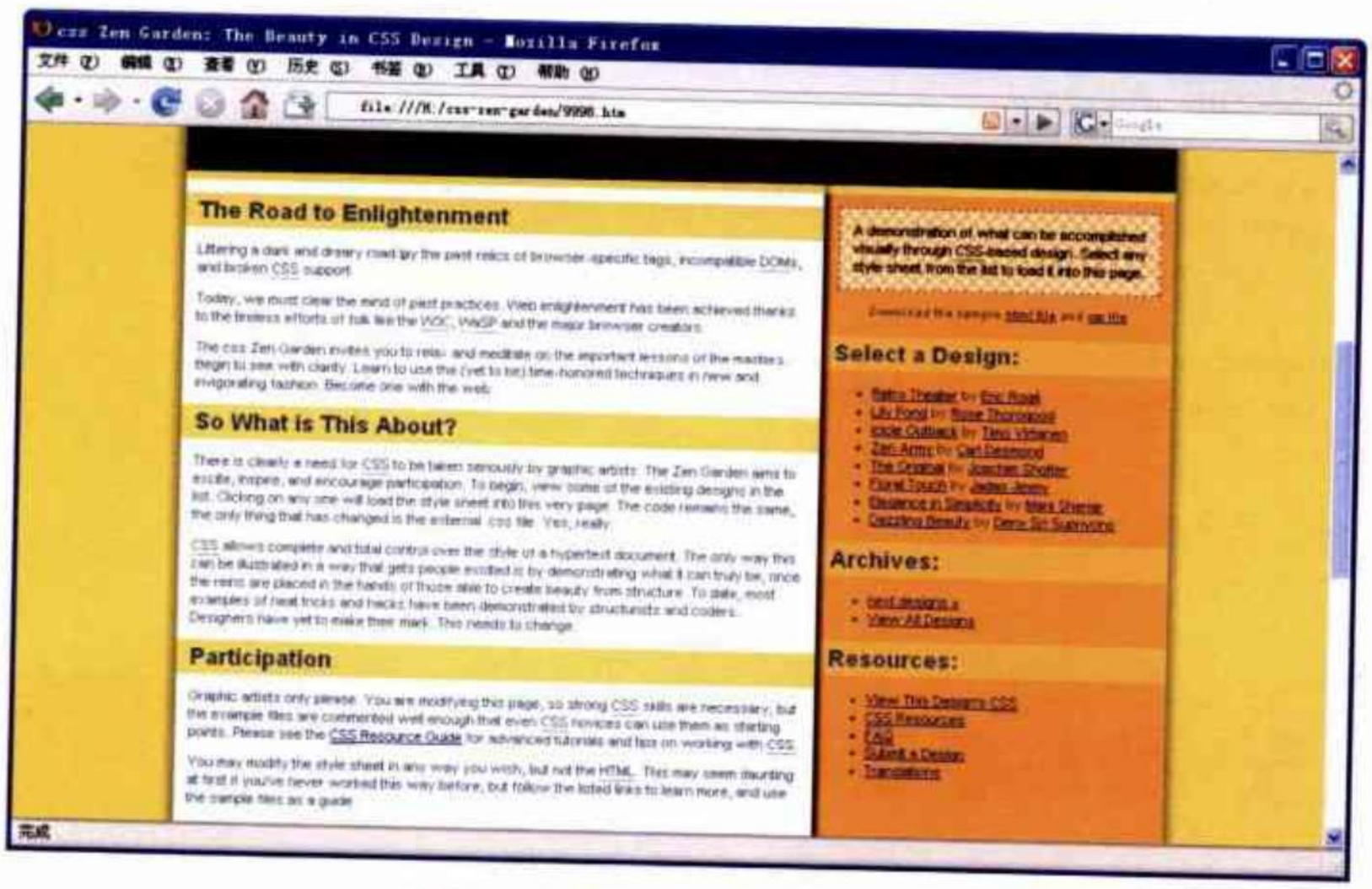


图17.25　设置supportingText部分的样式

⑪ 设置supprtingText中的footer部分，这个部分也使用绝对定位，同样以container为定位基准，并通过设置宽度，使这几个连接竖直排列。代码如下，效果如图17.26所示。

```
#footer{
  width:70px;
  position:absolute;
  bottom:20px;
  right:30px;
  }
#footer a{
  text-align:center;
  display:block;
  padding:0 20px;
  border:1px #c00 dashed;
  }
```

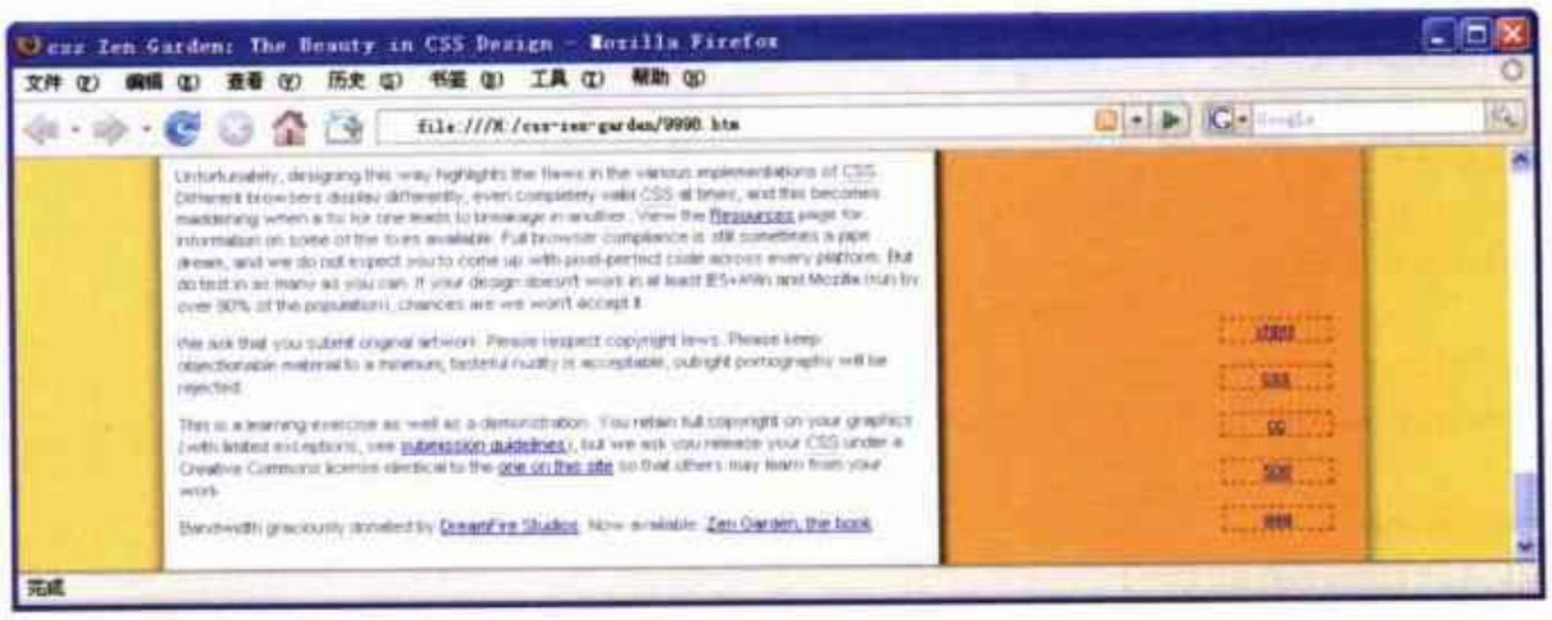


图17.26 设置footer

⑫ 对整个页面的连接文字的样式进行设置。使用暗红色来表示链接文字，在整个页面中起到点缀作用。代码如下。

```
a:link,a:visited {
  text-decoration: none;
  color: #c00;
}
a:hover,a:active {
  text-decoration: underline;
}
```

至此，这个作品就完成了。

17.5 修改新页面方案

如果换一张主题图像，再从前面准备的图片中挑选一张，只需要稍做加工，就可以立即使这个页面焕然一新。

假设选择如图17.27所示的这张菊花图片，对它按照虚线所示进行剪裁，然后水平反转，这样右上角会有一个黑色的区域，正好放置英文的标题文字。

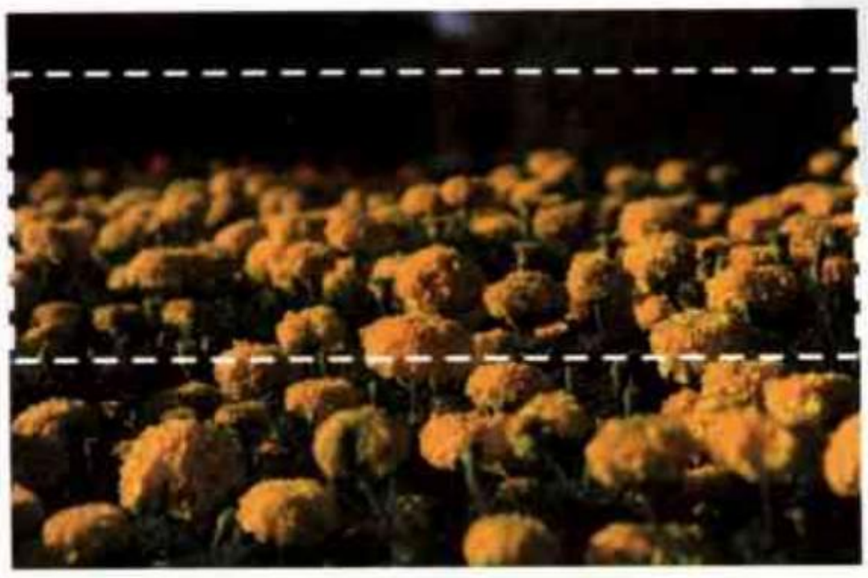


图17.27 对照片进行剪裁

然后将页面背景的颜色和标题背景的颜色也进行更换，就会得到一个新的页面了，效果如图17.28所示。修改完成后案例的最终效果请参考本书光盘“第17章/2/9998.htm”。

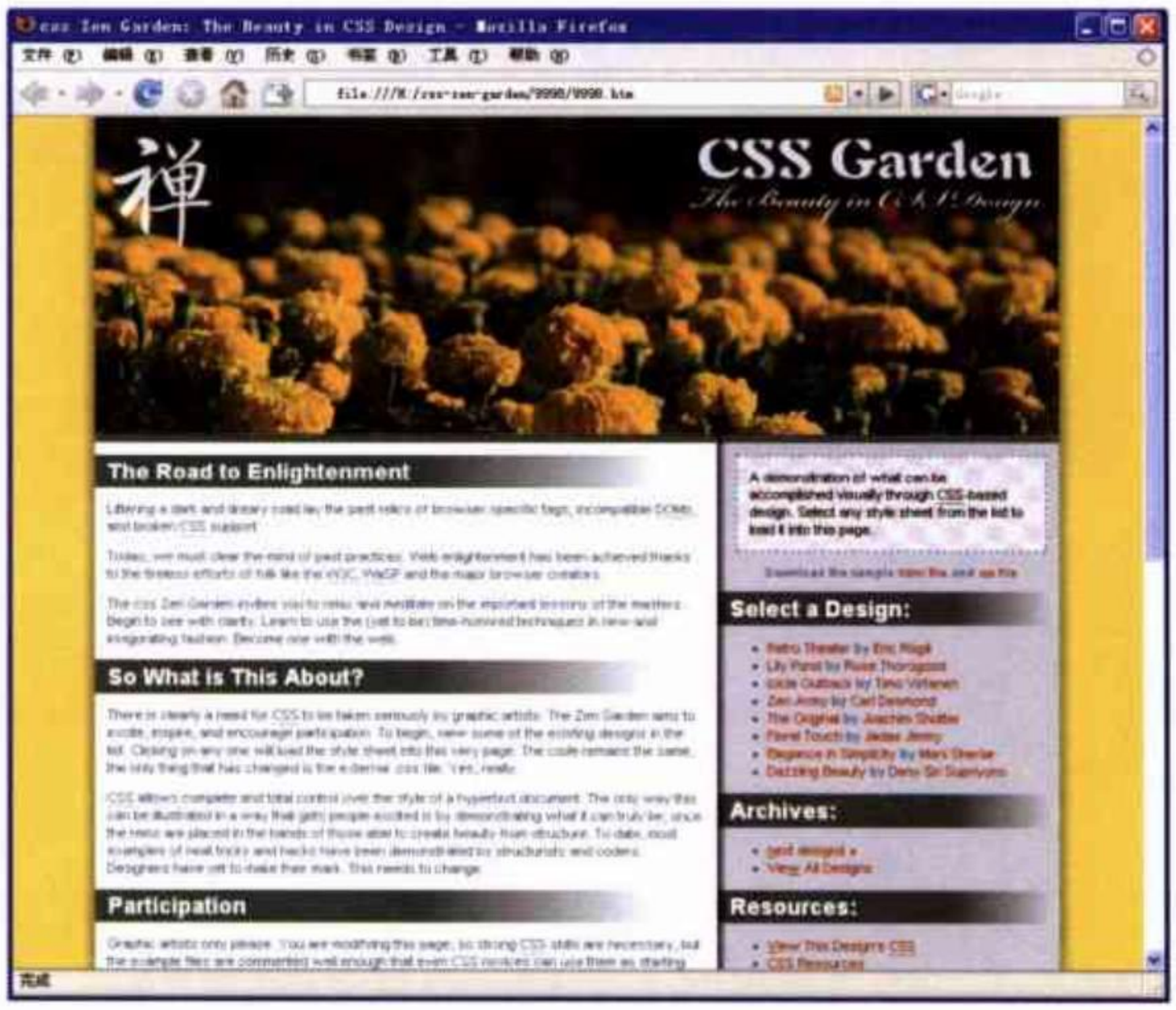


图17.28　稍加修改得到的新页面

17.6 本章小结

在本章中，完成了一个作品从创意、拍摄素材到实际完成的全过程。通过这个案例，读者应该完全可以根据自己的创意，制作出与众不同的网页作品了。

相信读者学习完这本书，已经对CSS进行页面设计和表现的思路和方法有了比较全面、深入的理解和掌握。古人说“纸上得来终觉浅，绝知此事要躬行”，这是在任何时候都不会过时的真理，希望读者能够把掌握本书中的技术作为学习的开始，以它为基础，继续不断地探索、提高，甚至发掘出别人所没有发现的技术和方法，这才是真正值得骄傲的事情。

CSS 设计彻底研究

CSS设计

彻底研究

附录　CSS英文小字典

本字典包含160个英文单词，系《CSS设计彻底研究》一书中出现频度最高的160个单词。对于英语基础不太好的读者，可以先集中把这些英文单词学一遍。

	英文	中文
A	absolute	绝对的
	active	活动的，激活的，<a>标记的一个伪类
	align	对齐
	alpha	透明度，半透明
	anchor	锚记<a>标记是这个单词的缩写
	arrow	箭头
	auto	自动

	英文	中文
B	background	背景
	banner	页面上的一个横条
	black	黑色
	blink	闪烁
	block	块
	blue	蓝色
	body	主体，一个HTML标记
	bold	粗体
	border	边框
	both	二者都，是clear属性的一个属性值
	bottom	底部，是一个CSS属性
	box	盒子
	br	换行标记
	bug	软件程序中的错误
	building	建立
	button	按钮

	英文	中文
C	cell	表格的单元格
	center	中间，居中
	centimeter	厘米
	child	孩子
	circle	圆圈
	class	类别
	clear	清除

续表

C	英文	中文
	cm	厘米
	color	颜色
	connected	连接的
	contact	联系
	content	内容
	crosshair	十字叉丝
	css	层叠样式表
	cursor	鼠标指针

D	英文	中文
	dashed	虚线
	decimal	十进制
	decoration	装饰
	default	默认的
	definition	定义
	design	设计
	display	显示，CSS的一个属性
	division	分区，div就是这个单词的缩写
	document	文档
	dotted	点线
	double	双线

E	英文	中文
	element	元素

F	英文	中文
	father	父亲
	filter	滤镜，过滤器
	first	第一个
	fixed	固定的
	float	浮动
	font	字体
	for	在循环语句中的一个保留字
	four	4个

G	英文	中文
	gif	一种图像格式
	gray	灰色
	green	绿色

H	英文	中文
	hack	常用于CSS中的一些招数，或者类似于偏方的技巧
	hand	手
	head	头部
	height	高度
	help	帮助
	here	这里
	hidden	被隐藏
	home	首页
	horizontal	水平的
	hover	属表指针经过获称为悬停状态

I	英文	中文
	image	图像
	important	重要的
	indent	缩进
	index	索引
	inline	行内
	inner	内部的
	italic	意大利体，斜体

J	英文	中文
	jpg	一种图像格式
	justify	两端对齐

L	英文	中文
	language	语言
	last	最后一个
	left	左边
	length	长度
	level	级别，例如block-levle就是块级。
	line	线
	link	链接
	list	列表
	lowercase	小写

M	英文	中文
	margin	外边距
	max	最大的
	medium	中间

续表

M	英文	中文
	menu	菜单
	middle	中间
	millimeter	毫米
	min	最小的
	model	模型
	move	移动

N	英文	中文
	navigation	导航
	new	新的
	none	无，不，没有
	normal	标准

O	英文	中文
	object	对象
	oblique	一种斜体
	one	一个
	only	仅仅
	open	打开
	optional	可选的
	orange	橙色
	outer	外面的
	overflow	溢出

P	英文	中文
	padding	内边距
	point	点
	pointer	指针，指示器
	position	定位，位置
	progress	进度
	public	公开的
	purple	紫色

R	英文	中文
	red	红色
	relative	相对的
	repeat	重复，平铺
	replacement	替换

续表

	英文	中文
R	resize	重新设置大小
	right	右边
	row	行

	英文	中文
S	scroll	滚动
	shadow	阴影
	silver	银色
	size	尺寸
	solid	固体，实线
	solution	方案
	son	儿子
	span	一个HTML标记
	special	特殊的
	square	方块
	static	静态的
	strong	强壮，加粗的
	style	样式

	英文	中文
T	table	表格
	td	单元格的HTML标记
	text	文本
	thick	粗的
	thin	细的
	three	三个
	through	穿过
	title	标题
	top	顶部
	tr	表格中“行”的HTML标记
	transitional	过渡的
	two	两个
	type	类型

	英文	中文
U	underline	下划线
	upper	上面的
	uppercase	大写
	url	网址

V	英文	中文
	vertical	竖直的
	visited	访问过的

W	英文	中文
	white	白色
	width	宽度

Y	英文	中文
	yellow	黄色