

CSS Mastery

Advanced Web Standards Solutions Second Edition

# 精通CSS

## 高级Web标准解决方案（第2版）

Andy Budd

[英] Simon Collison 著

Cameron Moll

陈剑瓯 译

- Amazon第一CSS畅销书全新改版
- 令人叫绝的CSS技术汇总
- 涵盖CSS 3和HTML 5



人民邮电出版社  
POSTS & TELECOM PRESS

CSS Mastery Advanced

Second Edition

# 精通CSS 高级Web标准解决方案 (第2版)

“Andy Budd对CSS设计的底层技术和方法有着深刻的理解，而且更善于将这些知识娓娓道来。在跨浏览器支持问题上，无人可以望其项背。”

——Molly E. Holzschlag, Web标准项目负责人和W3C HTML工作组专家

“Andy Budd多年来一直在思考、实践和宣传基于标准的网页设计，在本书中，Andy将最实用的解决方案、经验和技巧和盘托出，毫无保留。对广大Web设计人员来说，这无疑是一大幸事。”

——Dan Cederholm, 《Web标准实战》的作者

CSS作为Web标准的一部分，已经成为现代网页设计中必不可少的关键要素。CSS看似简单，但真正精通CSS绝非易事。在使用CSS开发网站时，会遇到形形色色的浏览器bug和不一致问题，而解决方案又五花八门，往往让使用者感觉千头万绪，不知从何着手。

本书将最有用的CSS技术汇总在一起，总结了CSS设计中的最佳实践，讨论了解决各种实际问题的技术，在很多方面填补了其他CSS图书的空白。正因如此，英文版出版后，一时洛阳纸贵，多次重印，并迅速登上Amazon图书排行榜前列，最高时甚至与《哈利·波特》并驾齐驱，创造了计算机图书的销售奇迹。

作为最新的升级版，本书淘汰了过时的内容，补充了大量CSS领域的新技术，涵盖了CSS 3和HTML 5，更无愧于Web设计人员必读的第一经典著作。

Apress®

本书相关信息请访问：图灵网站 <http://www.turingbook.com>

读者/作者热线：(010)51095186

反馈/投稿/推荐信箱：[contact@turingbook.com](mailto:contact@turingbook.com)



ISBN 978-7-115-22673-0



9 787115 226730 >

ISBN 978-7-115-22673-0

定价：49.00元

人民邮电出版社网址：[www.ptpress.com.cn](http://www.ptpress.com.cn)

**TURING** 图灵程序设计丛书 Web开发系列

**CSS Mastery**  
**Advanced Web Standards Solutions Second Edition**

# **精通CSS**

## **高级Web标准解决方案**

### **(第2版)**

**Andy Budd**  
[英] **Simon Collison** 著  
**Cameron Moll**  
陈剑瓯 译

人民邮电出版社  
北京

## 图书在版编目 (C I P) 数据

精通CSS：高级Web标准解决方案：第2版 / (英)  
巴德 (Budd, A.)，(英) 科利森 (Collison, S.)，(英)  
莫尔 (Moll, C.) 著；陈剑瓯译。— 北京：人民邮电出  
版社，2010.5

(图灵程序设计丛书)

书名原文：CSS Mastery: Advanced Web Standards  
Solutions Second Edition  
ISBN 978-7-115-22673-0

I. ①精… II. ①巴… ②科… ③莫… ④陈… III.  
①主页制作—软件工具，CSS—程序设计 IV.  
①TP393.092

中国版本图书馆CIP数据核字(2010)第050871号

## 内 容 提 要

本书汇集了最有用的 CSS 技术，介绍了 CSS 的基本概念和最佳实践，结合实例探讨了图像、链接和列表的操纵，还有表单设计、数据表格设计、纯 CSS 布局等核心 CSS 技术。此外，书中着眼于创建跨浏览器的技术，讨论了 bug 及其捕捉和修复技术，还将所有技术组合成两个精彩的实例，讲述这些技术的工作原理和实际用法。

本书适合具有 HTML 和 CSS 基础知识的读者阅读。

## 图灵程序设计丛书 精通CSS：高级Web标准解决方案（第2版）

◆ 著 [英] Andy Budd Simon Collison Cameron Moll  
译 陈剑瓯  
责任编辑 傅志红  
执行编辑 谢灵芝

◆ 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>  
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16  
印张：17.5  
字数：413千字 2010年5月第1版  
印数：1~4 000册 2010年5月北京第1次印刷  
著作权合同登记号 图字：01-2009-7278号  
ISBN 978-7-115-22673-0

定价：49.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223

反盗版热线：(010)67171154

# 序

---

在网页设计的精彩世界里，实现同一个目标有千百种方法，而且新的方法还在不断地出现。对于特定的问题没有唯一正确的解决方法，丰富的选择使我们这些网页设计人员受益良多，同时也困扰着我们。这些选择虽然能使网页设计变得生动有趣，但同时也会令人无所适从。本书将帮助你减少麻烦，理清头绪。

Andy Budd多年来一直在编写、设计和宣讲基于标准的网页设计，我们现在有幸在本书中亲眼目睹他以简洁清晰的方式讲授最重要的CSS技术。本书提供了一套网页设计人员不可或缺的解决方案、技巧和建议。

有些图书中仅仅提出一种实现某一目标的正确方法，我很不喜欢这样的做法，Andy所做的正好相反，他为各种任务提供了多种方法，如对链接应用样式，创建标签页式导航，使用节省时间的CSS 3解决方案，或创建固定、流式的、灵活的布局，这些都有多种途径加以实现，书中还针对如何消除使用CSS设计时出现的那些恼人的浏览器bug给出了许多提示。掌握了常见设计元素的这些时髦漂亮的设计方法，你就可以做出更明智的选择。

不只如此，Andy还邀请两位出色的设计人员将这些技术组合在一起，通过两个实例研究向我们展示这些基本技术如何组合在一起。长期以来，我一直推崇Cameron和Simon的作品，看了他们写的两个绝佳实例，讨论流式布局、无懈可击的布局和灵活多样的样式解决方案，真的是受益匪浅。

好了，现在请开始深入研究并逐步消化这千百种设计方法，祝你早日成为精通CSS的高手。

Dan Cederholm，经典著作*Web Standards Solutions*一书的作者

PDG

# 前　　言

---

尽管CSS资源的数量越来越多，但是在CSS邮件列表上仍然总是看到有人问同样的问题：如何让设计居中？最好的圆角框技术是什么？如何创建三列布局？

如果你熟悉CSS设计社区，那么寻找解决方案时无非就是回想一下某篇文章或某种技术曾在哪个网站重点介绍过。但是，如果你是CSS的初学者，或者没有时间阅读所有博客，那么这些信息可能并不好找。

CSS有些方面（比如定位模型和特殊性）比较晦涩，即使是有经验的CSS开发人员也会遇到问题。这是因为大多数CSS开发人员都是靠自学的，他们从各种文章和别人的代码中学习经验，而没有全面理解CSS规范。这也不奇怪，因为CSS规范本身十分复杂，常常还自相矛盾，它的目标读者是浏览器厂商而不是网页开发人员。

此外，还得应付浏览器问题。浏览器的bug和不一致性是现代CSS开发人员面对的一个最大问题。不幸的是，许多bug都没有很好地记载，它们的修复方法基本上只是在开发人员之间口口相传。你知道自己必须以某种方式做某件事，否则在某种浏览器中就会出问题。但是，你记不住是在哪种浏览器中会出问题，也说不清为什么会出现问题。

所以，我产生了写这么一本书的想法。这本书将最有用的CSS技术汇总在一起，集中介绍实际的浏览器问题，从而弥补人们欠缺的CSS知识。本书会帮助你加快学习CSS的进程，使你的编码技术很快达到CSS专家的水平。

## 读者对象

本书适合具有HTML和CSS基础知识的任何人<sup>①</sup>阅读。无论你是刚刚接触CSS设计，还是已经开发纯CSS站点好几年了，书中都有适合你的内容。如果你已经使用CSS一段时间了，但还没有达到专家级水平，那么你能够从本书获得最大的收益。本书为你提供了各种实用的建议和示例，可以帮助你精通现代CSS设计。

## 本书结构

本书前3章讨论基本的CSS概念和最佳实践，帮助你轻松地入门。你将学习如何建立代码结

---

<sup>①</sup> 如果你不具备这些基础知识，可以阅读人民邮电出版社出版的《HTML XHTML与CSS基础教程》（第6版）。

——编者注

构和添加注释，了解CSS定位模型的细节以及浮动和清理的工作原理。你也许已经掌握了其中的许多内容，但是可能会发现自己有遗漏或理解不充分的地方。因此，前3章是不错的CSS入门材料，也可以帮助你重温已经知道的知识。

介绍了基本知识之后，后面5章讨论核心CSS技术，比如操纵图像、链接和列表、设计表单和数据表格，以及进行纯CSS布局。每一章都由浅入深，最后讨论比较复杂的示例。在这几章中，你将学习如何创建圆角框、带透明阴影的图像、标签页式导航条和交互式按钮。许多情况下，我会先展示传统技术，然后说明如何用CSS制作出同样的效果。如果你想研究本书中的示例，可以从[www.cssmastery.com](http://www.cssmastery.com)或[www.friendsofed.com](http://www.friendsofed.com)下载所有示例代码<sup>①</sup>。

浏览器bug是许多CSS开发人员最头疼的问题，所以本书中的所有示例都着眼于创建跨浏览器的技术。此外，本书还用一整章讨论bug和bug修复。在这一章中，你将全面学习bug捕捉技术，学会在bug作乱之前就发现并消灭它，甚至还会学习是什么造成了IE中许多看似毫无规律的CSS bug。

最后两章是真正的“大餐”。Simon Collison和Cameron Moll是两位最杰出的CSS设计人员，他们将本书讨论的各种技术组合成两个精彩的实例来研究。从而，你不但会学习这些技术的工作原理，而且会看到如何将它们用在实际项目中。

本书可以从头到尾地阅读，也可以放在计算机旁边作为参考资料，随时查阅提示、技巧和技术，决定权在你。

## 本书约定

本书使用了几个约定，需要注意。本书采用了以下术语。

- “HTML”指HTML和XHTML这两种语言。
- 除非特别声明，“CSS”是指CSS 2.1规范。
- “Windows的IE 6和更低版本”指Windows的IE 5.0~6.0。
- “现代浏览器”是指最新版的Firefox、Safari、Opera、IE 7以及IE 7以上版本。
- 本书中的所有HTML示例都应该嵌套在一个有效文档的<body>中，同时，CSS包含在外部样式表中。偶尔为了尽量简短，HTML和CSS放在了同一个代码示例中。但是在真实的文档中，这些代码需要放在各自的位置上才能正常工作。

最后，对于包含重复数据的HTML示例，我们不会列出每一行，而是适时地使用省略号表示部分代码。

<sup>①</sup> 本书示例代码也可从图灵网站[www.turingbook.com](http://www.turingbook.com)本书网页免费注册下载。——编者注

# 致 谢

---

感谢所有直接或间接地帮助我们撰写本书的人。

感谢我在Clearleft的朋友和同事，他们在我撰写本书的过程中提供了鼓励和反馈意见。特别感谢Natalie Downe为本书贡献其广博的知识和经验。他的支持和指导是无价的，我到现在还没搞清楚他究竟是怎样挤出那么多时间的。

感谢Chris Mills在我开始动笔时就一直引导着我，帮助我将想法变成现实。感谢所有不知疲倦地帮助本书按时出版的Apress出版社的工作人员，他们的奉献精神和职业态度令人敬佩。

感谢我的同事一直以来与我分享他们的开发经验，这些知识在不断美化Web环境。如果没有下面各位的工作，本书是不可能完成的：Cameron Adams、John Allsopp、Pachel Andrew、Nathan Barley、Holly Bergevin、Mark Boulton、Douglas Bowman、The BritPack、Dan Cederholm、Tantek Çelik、Joe Clark、Andy Clarke、Simon Collison、Mike Davidson、Garrett Dimon、Derek Featherstone、Nick Fink、Patrick Griffiths、Jon Hicks、Molly E. Holzschlay、Shaun Inman、Roger Johansson、Jeremy Keith、Ian Lloyd、Ethan Marcotte、Drew McLellan、Eric Meyer、Cameron Moll、Dunstan Orchard、Veerle Pieters、D. Keith Robinson、Richard Rutter、Jason Santa Maria、Dave Shea、Jeffrey Veen、Russ Weakley、Simon Willison、Jeffrey Zeldman，等等。

感谢我的博客的所有读者和过去两年中我在各种会议、讨论会和培训活动中遇到的所有人，他们的意见和思想丰富了本书的内容。

最后，感谢你阅读本书，希望本书能够帮助你将CSS技能提升到新的层次。

——Andy Budd

首先，感谢你选购本书，希望它助你在日复一日的工作中改进作品质量。从事这一行业的人潜能无限，总能让我深受鼓舞，你肯定也不会例外。

我响应Andy的话，也要感谢那些改造和美化Web的重要人物，他们让今日的Web超越了以往任何时候。若干年以后，这些人必将深受推崇和爱戴，比肩那些将人类首次送上月球的科学家们。

特别感谢Aaron Barker([aaronbarker.net](http://aaronbarker.net))帮助我实现了实例研究中的几个jQuery和AJAX实例。

最重要的是，我要向我美丽的夫人Suzanne和四个儿子Everest、Edison、Isaac和Hudson表示最深的感激之情。没有他们的爱、耐心和大力支持，我不可能做出这些成就。

——Cameron Moll

我要感谢我的同事和朋友Gregory Wood帮我构思并实现了“Climb the Mountains”。他的设计总能给我启发，我以后就要成为他那样的设计师。我还要感谢我在Erskine设计公司的所有同事，他们对我狂热地从事这本书的写作睁只眼闭只眼，承担了不少份外工作。非常感谢Simon Campbell、Jamie Pittock、Glen Swinfield、Phil Swan、Vicky Twycross和Angela Campbell。

最重要的是，我要借此机会感谢我的母亲，还有几位上一版出版后逝去的亲人，我的祖父和外祖父，尤其是我父亲。尽管他们已离我而去，但我依然在努力让他们为我自豪。

——Simon Collison



# 版 权 声 明

Original English language edition, entitled *CSS Mastery: Advanced Web Standards Solutions, Second Edition* by Andy Budd, Simon Collison and Cameron Moll, published by Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705 USA.

Copyright © 2009 by Andy Budd, Simon Collison and Cameron Moll. Simplified Chinese-language edition copyright © 2010 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由 Apress L.P. 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。



# 目 录

---

<b>第1章 基础知识</b>	1
1.1 设计代码的结构	2
1.1.1 标记简史	2
1.1.2 文档类型、DOCTYPE 切换和 浏览器模式	13
1.1.3 有效性验证	14
1.2 小结	17
<b>第2章 为样式找到应用目标</b>	18
2.1 常用的选择器	18
2.2 通用选择器	20
2.3 高级选择器	20
2.3.1 子选择器和相邻同胞选择器	21
2.3.2 属性选择器	22
2.3.3 层叠和特殊性	26
2.3.4 继承	29
2.4 规划、组织和维护样式表	31
2.4.1 对文档应用样式	31
2.4.2 样式指南	35
2.5 小结	37
<b>第3章 可视化格式模型</b>	38
3.1 盒模型概述	38
3.1.1 IE 和盒模型	40
3.1.2 外边距叠加	41
3.2 定位概述	43
3.2.1 可视化格式模型	43
3.2.2 相对定位	44
3.2.3 绝对定位	45
3.2.4 浮动	47
3.3 小结	53
<b>第4章 背景图像效果</b>	54
4.1 背景图像基础	54
4.2 圆角框	57
4.2.1 固定宽度的圆角框	57
4.2.2 山顶角	62
4.3 投影	67
4.3.1 简单的 CSS 投影	68
4.3.2 来自 Clagnut 的投影方法	70
4.4 不透明度	73
4.5 图像替换	78
4.5.1 FIR	79
4.5.2 Phark	80
4.5.3 sIFR	80
4.6 小结	82
<b>第5章 对链接应用样式</b>	83
5.1 简单的链接样式	83
5.2 让下划线更有趣	85
5.2.1 简单的链接修饰	85
5.2.2 奇特的链接下划线	86
5.3 已访问链接的样式	87
5.4 为链接目标设置样式	87
5.5 突出显示不同类型的链接	88
5.6 创建类似按钮的链接	91
5.6.1 简单的翻转	92
5.6.2 图像翻转	93
5.6.3 Pixy 样式的翻转	93
5.6.4 CSS 精灵	95
5.6.5 用 CSS 3 实现翻转	96
5.7 纯 CSS 工具提示	98
5.8 小结	100
<b>第6章 对列表应用样式和创建导航条</b>	101
6.1 基本列表样式	101
6.2 创建基本的垂直导航条	102
6.3 在导航条中突出显示当前页面	105

## 2 目 录

---

6.4 创建简单的水平导航条 .....	106	第 9 章 bug 和修复 bug .....	183
6.5 创建图形化导航条 .....	108	9.1 捕捉 bug .....	183
6.6 简化的“滑动门”标签页式导航 .....	110	9.2 捕捉 bug 的基本知识 .....	189
6.7 Suckerfish 下拉菜单 .....	112	9.2.1 尽量在一开始就避免 bug .....	190
6.8 CSS 图像映射 .....	114	9.2.2 隔离问题 .....	190
6.9 远距离翻转 .....	124	9.2.3 创建基本测试案例 .....	191
6.10 对于定义列表的简短说明 .....	130	9.2.4 修复问题，而不是修复症状 .....	191
6.11 小结 .....	131	9.2.5 请求帮助 .....	192
<b>第 7 章 对表单和数据表格应用样式 .....</b>	<b>132</b>	9.3 拥有布局 .....	192
7.1 对数据表格应用样式 .....	132	9.3.1 什么是布局 .....	192
7.1.1 表格特有的元素 .....	134	9.3.2 布局的效果 .....	193
7.1.2 数据表格标记 .....	135	9.4 解决方法 .....	195
7.1.3 对表格应用样式 .....	136	9.4.1 IE 条件注释 .....	195
7.1.4 添加视觉样式 .....	137	9.4.2 关于 hack 和过滤器的一个警告 .....	196
7.2 简单的表单布局 .....	139	9.4.3 明智地使用 hack 和过滤器 .....	197
7.2.1 有用的表单元素 .....	140	9.4.4 应用 IE for Mac 带通过滤器 .....	197
7.2.2 基本布局 .....	140	9.4.5 应用星号 HTML hack .....	198
7.2.3 其他元素 .....	142	9.4.6 应用子选择器 hack .....	199
7.2.4 修饰 .....	144	9.5 常见 bug 及其修复方法 .....	199
7.3 复杂的表单布局 .....	145	9.5.1 双外边距浮动 bug .....	199
7.3.1 可访问的数据输入元素 .....	146	9.5.2 3 像素文本偏移 bug .....	200
7.3.2 多列复选框 .....	147	9.5.3 IE 6 的重复字符 bug .....	201
7.3.3 表单反馈 .....	150	9.5.4 IE 6 的“藏猫猫”bug .....	202
7.4 小结 .....	152	9.5.5 相对容器中的绝对定位 .....	203
<b>第 8 章 布局 .....</b>	<b>153</b>	9.5.6 停止对 IE 的批评 .....	204
8.1 计划布局 .....	153	9.6 分级浏览器支持 .....	204
8.2 设置基本结构 .....	156	9.7 小结 .....	206
8.3 基于浮动的布局 .....	158	<b>第 10 章 实例研究: Roma Italia .....</b>	207
8.3.1 两列的浮动布局 .....	158	10.1 关于这个实例研究 .....	207
8.3.2 三列的浮动布局 .....	161	10.2 基础 .....	209
8.4 固定宽度、流式和弹性布局 .....	163	10.2.1 着眼于 HTML 5 .....	210
8.4.1 流式布局 .....	164	10.2.2 reset.css .....	211
8.4.2 弹性布局 .....	166	10.3 1080 布局和网格 .....	212
8.4.3 流式和弹性图像 .....	168	10.4 高级 CSS 2 和 CSS 3 特性 .....	215
8.5 faux 列 .....	170	10.4.1 网站需要在每种浏览器中看起来完全一样吗 .....	216
8.6 高度相等的列 .....	173	10.4.2 属性选择器 .....	217
8.7 CSS 3 列 .....	176		
8.8 CSS 框架与 CSS 系统 .....	177		
8.9 小结 .....	181		

---

10.4.3 box-shadow、RGBa 和 text-overflow.....	218	11.2.3 使用条件注释的 IE 样式表 .....	239
10.5 字体链接和更好的 Web 排版 .....	221	11.3 网格灵活性 .....	240
10.5.1 按以前的方式设置 font-size.....	221	11.4 用 body 类控制导航 .....	241
10.5.2 标点符号悬挂 .....	222	11.4.1 突出显示当前页面 .....	241
10.5.3 多栏文本布局 .....	224	11.4.2 控制 blockquote 所处 的层 .....	244
10.5.4 @font-face .....	225	11.5 战略性地选择元素 .....	245
10.5.5 Cufón, 向@font-face 发展的过渡手段 .....	228	11.5.1 深层后代选择器 .....	245
10.6 用 AJAX 和 jQuery 增加交互性 .....	230	11.5.2 :first-child 伪类 .....	248
10.6.1 AJAX .....	230	11.5.3 相邻同胞选择器 .....	249
10.6.2 jQuery .....	231	11.6 透明度、阴影和圆角 .....	250
10.6.3 使用 AJAX 和 jQuery 实现搜索 .....	232	11.6.1 我们的目标 .....	251
10.7 小结 .....	234	11.6.2 说明图像覆盖和 RGBa 透明度 .....	252
<b>第 11 章 实例研究：Climb the Mountains .....</b>	<b>235</b>	11.6.3 组合类 .....	254
11.1 关于这个实例研究 .....	235	11.6.4 border-radius .....	255
11.2 样式表的组织和约定 .....	237	11.6.5 box-shadow .....	256
11.2.1 screen.css .....	238	11.7 定位列表和显示内容 .....	257
11.2.2 reset .....	239	11.7.1 圆角 .....	259
		11.7.2 主海拔图 .....	260
		11.8 小结 .....	266





## 第1章

# 基础 知识

人类天生就是一种好奇的动物，我们都很喜欢摆弄新鲜玩意儿。这不，最近我买了一台新的iMac，还没看说明书呢，自己就先把它鼓捣了一番。我们喜欢自己去琢磨，对新东西形成自己的看法。我们会自己先胡乱摸索一阵子，发觉不对劲了，才会去查阅手册。

学习CSS（层叠样式表）最好的一种方式是直接开始使用它。实际上，我认为大多数人学习Web编程的过程都是这样：先从博客上看到了一些出色的效果，于是通过查看源代码研究它们是如何实现的，然后就在自己的个人网站上大胆尝试。人们几乎不会先去读完整的CSS规范，这些规范能把任何人送入梦乡。

修改别人的代码是很好的起步方法，但是如果不小心的话，就可能误解重要的概念，或者给日后造成问题。这一点我很清楚，因为我犯过好几次了。本章将讲解一些基本的但常常被误解的概念，并讨论如何让HTML和CSS保持清晰且结构良好。

在本章中，你将学习以下内容：

- 设计代码的结构；
- 有意义的文档的重要性；
- 命名约定；
- 什么时候使用ID和类名；
- 微格式；
- HTML和CSS的不同版本；
- 文档类型、DOCTYPE切换和浏览器模式。

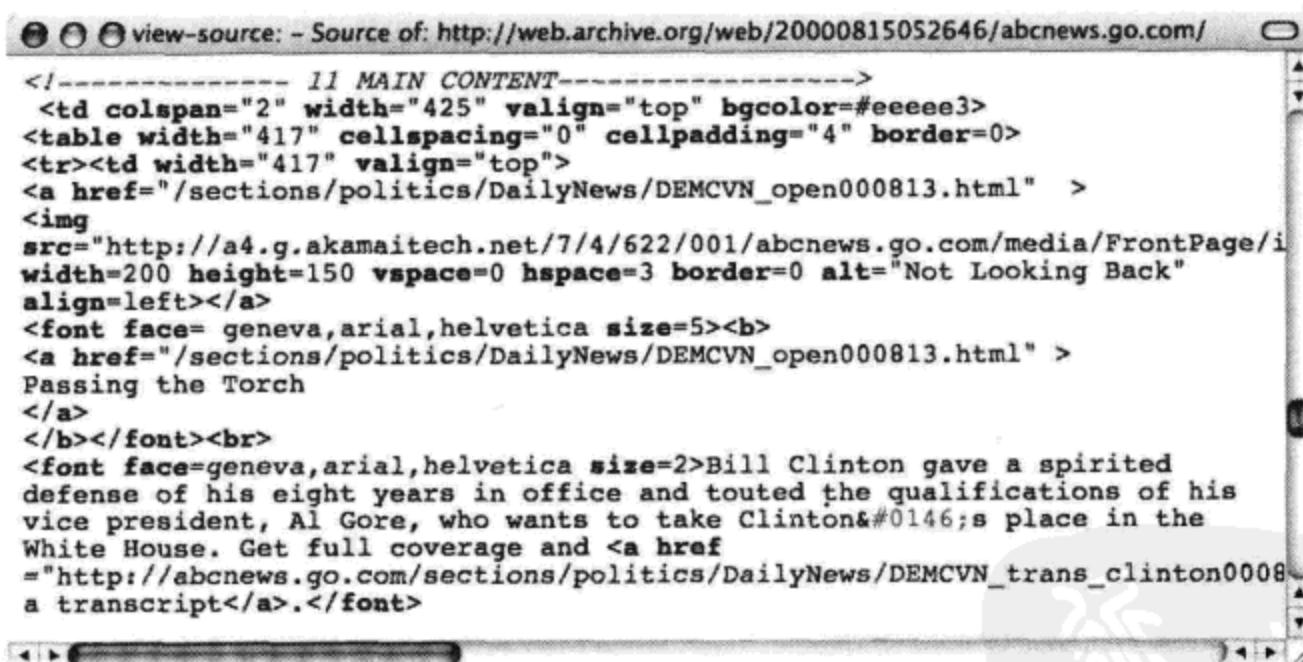
## 1.1 设计代码的结构

大多数人不关心建筑物的地基。但是，如果没有坚固的地基，建筑物的主体也就不会存在了。虽然本书讨论的是高级的CSS技术，但是如果缺少结构良好且有效的HTML文档，那么我们要做的许多事情都是不可能实现的（至少实现起来非常困难）。

在本节中，你将明白为什么结构良好且有意义的HTML文档在基于标准的CSS开发中非常重要，还将学习如何丰富文档的意义，从而让自己的开发工作更轻松。

### 1.1.1 标记简史

早期的Web仅仅是一系列相互链接的研究文档，使用HTML添加基本的格式和结构。但是，随着万维网的流行，HTML开始用来表现页面。人们结合使用字体和粗体标签来创建所需的视觉效果，而不仅仅是用标题元素突出显示页面的标题。表格成了一种布局工具而不是显示数据的方式，人们使用块引用（blockquote）来添加空白而不是表示引用。Web很快就含义不清，成了字体和表格标签的大杂烩。Web设计者把这样的标记称为“标签汤”（见图1-1）。



```
<!-- ----- 11 MAIN CONTENT----->
<td colspan="2" width="425" valign="top" bgcolor="#eeeeee3>
<table width="417" cellspacing="0" cellpadding="4" border=0>
<tr><td width="417" valign="top">
<a href="/sections/politics/DailyNews/DEMCVN_open000813.html" >
</a>
<font face= geneva,arial,helvetica size=5><b>
<a href="/sections/politics/DailyNews/DEMCVN_open000813.html" >
Passing the Torch
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>Bill Clinton gave a spirited
defense of his eight years in office and touted the qualifications of his
vice president, Al Gore, who wants to take Clinton's place in the
White House. Get full coverage and <a href
="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_trans_clinton0008
a transcript</a>.</font>
```

图1-1 abcnews.com（2000年8月14日）上的新闻头条页面的标记，它使用表格进行布局，对标题使用大的粗体字。代码结构不明，很难理解

网页变得越来越具表现力，代码却变得越来越难以理解和维护了。WYSIWYG（所见即所得）编辑器让设计者可以摆脱这些复杂性，它宣称可以提供全新的图形布局环境。遗憾的是，这些工具并没有使事情简化，反而添加了它们自己的复杂标记。使用FrontPage或Dreamweaver等编辑器能够通过简单的鼠标操作构建复杂的表格布局，但是嵌套的表格和“分隔线GIF”把代码弄得非常混乱（见图1-2）。更糟糕的是，这些布局极其脆弱，很容易被破坏。因为标记中有许多无意义的代码，很容易意外删除错误的标签，整个布局就可能会崩溃。另外，由于代码的复杂性，要找

到bug几乎是不可能的，这时从头编写页面往往比寻找bug更容易。如果是大型网站，情况会更复杂。因为网站的表现锁定到了各个页面，所以即使是最小的全站修改，也必须进行细致的“搜索并替换”。我曾经由于轻率地执行“搜索并替换”，导致弄坏了不止一个网站。因此，页面模板很快就不同步了，即使是简单的修改，也需要手工编辑网站上的每个页面。

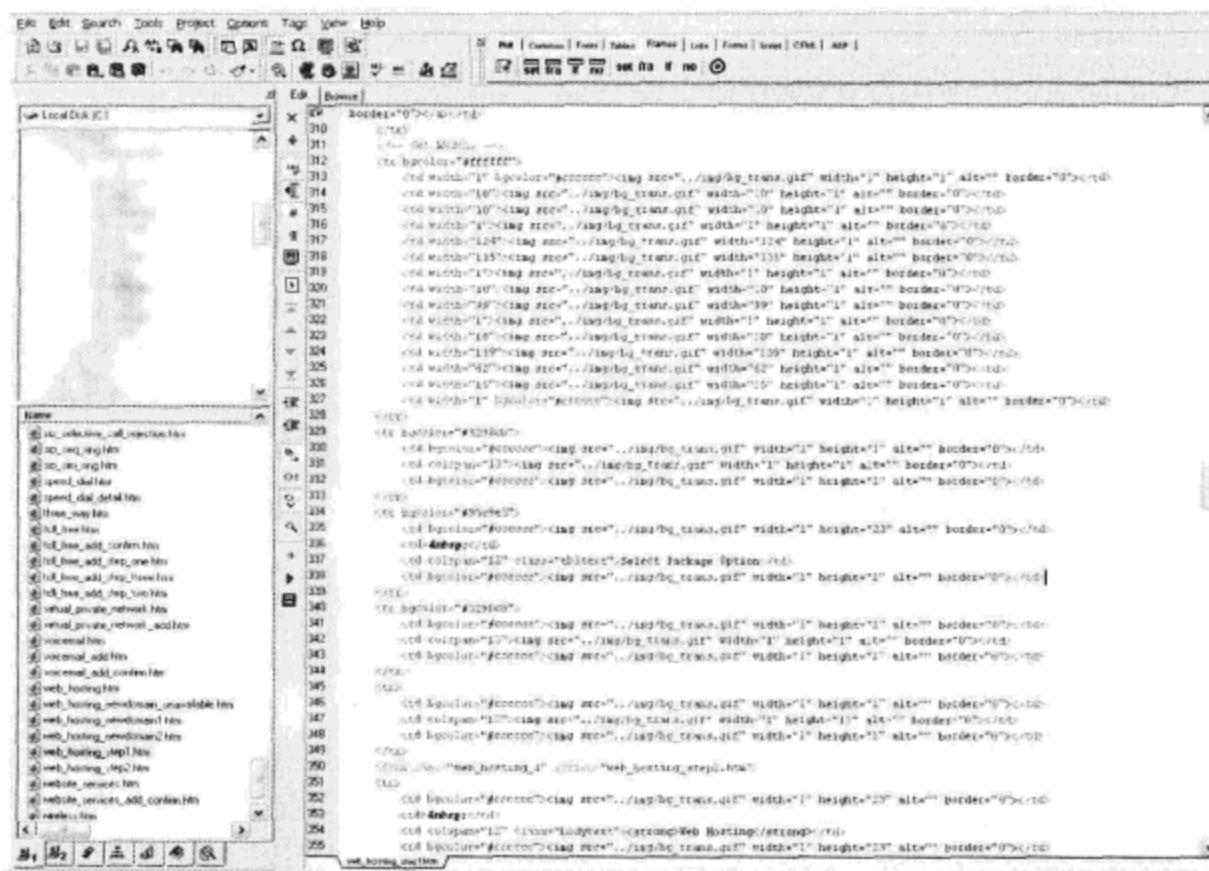


图1-2 使用大量分隔线GIF的基于表格的复杂布局（由Jeff L.提供）

表格本来不适合用来实现布局，所以David Siegel发明了一种聪明的解决方法。为了避免表格水平或垂直收缩，Siegel建议使用1像素的透明GIF。把这些隐藏的图像放在它们自己的表格单元格中，然后垂直或水平伸缩它们，这样就可以人为地控制单元格的最小宽度，从而保护布局不被破坏。这种图像也称为“shim GIF”，因为Dreamweaver中使用这个文件名。在基于表格的老式布局中常常会看到这种图像。好在这种做法已经过时了，现在代码中通常没有这些混乱的表现性元素。

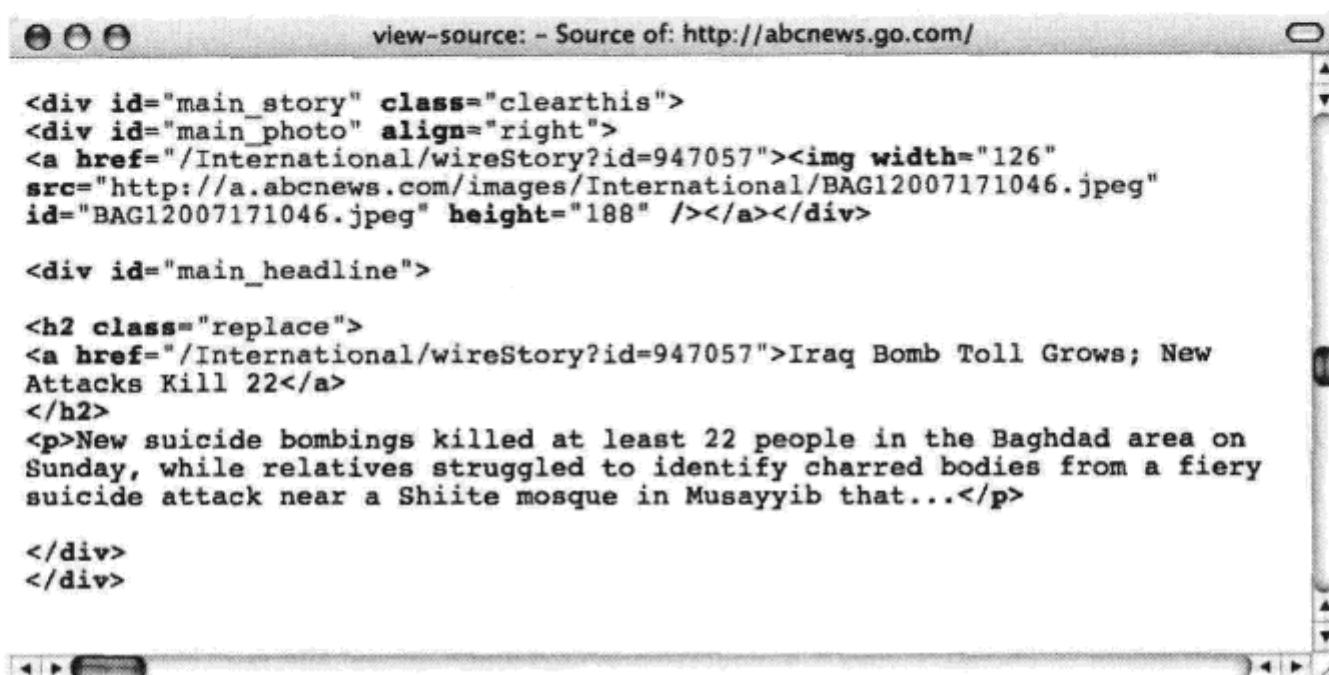
HTML并没有被看做简单的标记语言，反而得到了复杂、混乱和容易出错的坏名声。因此，许多人害怕直接编写代码，这导致人们过分依赖于可视化编辑器，造成整整一代设计者不知道如何编写代码。

千禧年之际，Web设计业简直是一团糟，必须采取措施了。

就在这种背景下，CSS出现了。有了CSS，就可以控制页面的外观，并且将文档的表现部分与内容分隔开。现在，只需在一个地方进行站点修改，修改就会贯彻到整个系统。可以去掉表现标签（比如字体标签），而且可以使用CSS而不是表格来控制布局。标记返朴归真，人们又开始

对底层代码感兴趣了。

文档又有意义了。浏览器的默认样式可以被覆盖，所以可以将某些内容标为标题，而不需要为它指定大号的、加粗的、难看的字体。可以创建列表，而这些列表不一定要用一系列项目符号来表示，可以使用没有关联样式的块引用。开发人员开始按照HTML元素的原义使用它们，无需管它们的外观（见图1-3）。



```
<div id="main_story" class="clearthis">
<div id="main_photo" align="right">
<a href="/International/wireStory?id=947057"></a></div>

<div id="main_headline">

<h2 class="replace">
<a href="/International/wireStory?id=947057">Iraq Bomb Toll Grows; New
Attacks Kill 22</a>
</h2>
<p>New suicide bombings killed at least 22 people in the Baghdad area on
Sunday, while relatives struggled to identify charred bodies from a fiery
suicide attack near a Shiite mosque in Musayyib that...</p>

</div>
</div>
```

图1-3 今年早些时候abcnews.com的新闻头条页面的标记，它具有良好的结构，容易理解。虽然它仍然包含一些表现标记，但是与图1-1中的代码相比有了显著的改进

## 1. 意义的重要性

有意义的标记为开发人员提供了几个重要的好处。与表现性的页面相比，有意义的页面更容易处理。例如，假设需要修改页面中的一个引用，如果这个引用加上了正确的标记，那么很容易搜索代码，找到第一个块引用元素。但是，如果这个引用只是另一个段落元素标签，就很难寻找了。再举一个更复杂但不太现实的例子，假设你需要在主页中添加一栏，只需把新内容放在右边，然后在CSS中更新宽度。要想在基于表格的布局中完成相同的任务，就需要在表格中添加一列，修改colspan设置，修改所有单元格的宽度，修改所有shim GIF的宽度。实际上，为了完成这个简单的修改，必须修改整个页面结构。

除了人之外，程序和其他设备也可以理解有意义的标记（也称为语义标记）。例如，搜索引擎可以识别出标题（因为它被包围在h1标签中）并予以重视。屏幕阅读器的用户可以依靠标题进行页面导航。

对于本书来说，更重要的是，有意义的标记可以简便地将元素调整为你所需的样式。它在文档中添加结构并且创建一个底层框架。可以直接设置元素的样式，而不需要添加其他标识符，因此避免了不必要的代码膨胀。

HTML包含丰富的有意义元素，比如：

- h1、h2等；
- ul、ol和dl；
- strong和em；
- blockquote和cite；
- abbr、acronym和code；
- fieldset、legend和label；
- caption、thead、tbody和tfoot。

因此，如果元素有恰当的含义，就应该使用。

几年来，在博客、邮件列表和开发人员论坛上，对于使用CSS还是表格有许多争论。出现这些争论常常是由于一部分开发人员习惯了基于表格的方法，他们不愿意学习新的技能。我可以理解他们的想法，因为基于CSS的布局最初看起来的确很难，尤其是当前的方法似乎仍然可行时，他们更不愿意接受新东西。但是，CSS的好处非常多，包括代码更少、下载更快和更容易维护等。大多数专业开发人员已经认识到Web标准的好处，而且大多数组织都不愿意按老方法做。因此，如果你仍然在使用基于表格的布局，会越来越难找到工作。好在这些老习惯正在被摒弃，新一代开发人员已经很难忍受麻烦的表格布局了。

## 2. ID和类名

有意义的元素会提供很好的基础，但是可用元素并不全面。HTML 4是作为简单的文档标记语言创建的，而不是界面语言。因此，没有用于内容区域或导航栏等的专用元素。虽然可以使用XML创建自己的元素，但是由于它太复杂，这在目前还不太现实。

HTML 5能为开发人员提供更丰富的元素，有望解决其中一部分问题。这包括header、nav、article、section和footer等结构性元素，以及data inputs和menu元素等新的UI特性。为了准备迎接HTML 5，许多开发人员已经开始采用这些名称作为ID和类名的命名约定。

次优的解决方案是使用现有的元素，并且通过添加ID或类名给它们赋予额外的意义。这会在文档中添加额外的结构，并给样式提供有用的“钩子”（hook）。因此，可以建立一个简单的链接列表，并且给它分配ID nav，从而创建出定制的导航元素：

```
<ul id="nav">
  <li><a href="/home/">Home</a></li>
  <li><a href="/about/">About Us</a></li>
  <li><a href="/contact/">Contact</a></li>
</ul>
```

ID用于标识页面上的特定元素（比如站点导航），而且必须是唯一的。ID也可以用来标识持久的结构性元素，例如主导航或内容区域。ID还可以用来标识一次性元素，例如某个链接或表单

元素。

一个ID名只能应用于页面上的一个元素，而同一个类名可以应用于页面上任意多个元素，因此类的功能强大得多。类非常适合标识内容的类型或其他相似的条目。例如，假设有一个新闻页面，其中包含多篇新闻，代码如下所示。

```
<div id="story-id-1">
    <h2>Salter Cane win Best British Newcomer award</h2>
    <p>In a surprise turn of events, alt folk group, Salter Cane, won ←
        Best British Newcomer and the Grammys this week...</p>
</div>

<div id="story-id-2">
    <h2>Comic Sans: The Movie wins best documentary at the BAFTAs </h2>
    <p>The story of this beloved typeface one the best documentary ←
        category. Director Richard Rutter was reported to be speechless...</p>
</div>
```

不必给每篇新闻分配不同的ID，可以给所有新闻分配一个类名news。

```
<div class="news">
    <h2>Salter Cane win Best British Newcomer award</h2>
    <p>In a surprise turn of events, alt folk group, Salter Cane, won ←
        Best British Newcomer and the Grammys this week...</p>
</div>
<div class="news">
    <h2>"Comic Sans: The Movie" wins best documentary at the BAFTAs </h2>
    <p>The story of this beloved typeface one the best documentary ←
        category. Director Richard Rutter was reported to be speechless...</p>
</div>
```

### 3. 为元素命名

在分配ID和类名时，一定要尽可能保持名称与表现方式无关。例如，如果希望所有表单通知消息显示为红色的，可以给它们分配类名red。只要页面上没有其他红色的元素，这就没问题。但是，如果还希望让4个必需的表单标签也显示为红色，就必须猜测这个类引用的是哪种元素，这时情况就开始有些混乱了。想象一下，如果在整个网站上到处使用表现性元素，代码会多么混乱。如果决定把表单通知由红色改为黄色，就更复杂了。在这种情况下，必须修改所有类名，否则名为red的元素就会显示为黄色。

因此，应该根据“它们是什么”来为元素命名，而不应该根据“它们的外观如何”来命名。这种方式会让代码更有意义，并且避免代码与设计不同步。对于前面的示例，不要给通知分配类名red，而是应该分配更有意义的名称，比如.warning或.notification（见图1-4）。有意义的类名的最大优点是可以在整个网站中重用它们。例如，还可以在其他类型的消息上使用.notification类，可以根据它们在文档中的位置应用完全不同的样式。

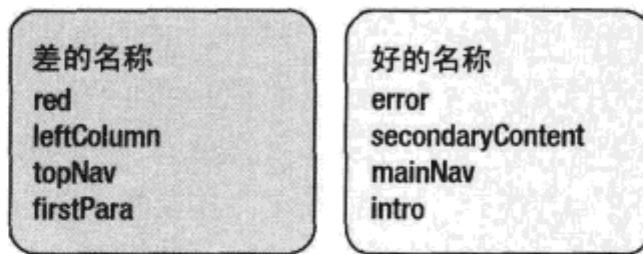


图1-4 好的ID名和差的ID名

在写类名和ID名时，需要注意区分大小写，因为浏览器认为`.andybudd`和`.andyBudd`是不同的类名。处理这个问题最好的方式是使用统一的命名约定。为了容易辨认，我总是采用完全小写的类名和ID，多个单词之间用连字符分隔。例如，`andy-budd`比`andyBudd`清楚得多。

#### 4. ID还是类

常常很难决定一个元素应该使用ID还是类名。一般原则是，类应该应用于概念上相似的元素，这些元素可以出现在同一页面上的多个位置，而ID应该应用于不同的唯一的元素。但是，究竟哪些元素是概念上相似的，哪些元素是唯一的？

例如，假设你的网站顶部包含主导航，在搜索结果页面的底部包含基于页面的导航，在页脚上还有另一个导航。是给它们分配不同的ID（比如`main-nav`、`page-nav`和`footer-nav`），还是都给它们指定`nav`类，然后根据它们在文档中的位置应用样式？我以前喜欢使用前一种方法，因为它的针对性更强一点儿。但是，这有几个问题。如果我现在需要把搜索结果导航放在搜索页面的顶部和底部，或者需要在页脚中有两级导航，那会怎么样？

如果使用大量ID，很快就会难以找到唯一的名称，最终不得不创建非常长、非常复杂的命名约定。因此，现在我比较喜欢使用类名。只有在目标元素非常独特，绝不会对网站上其他地方别的东西使用这个名称时，才会使用ID。换句话说，只有在绝对确定这个元素只会出现一次的情况下，才应该使用ID。如果你认为以后可能需要相似的元素，就使用类。保持命名约定通用，并且使用类，就不会出现一长串ID选择器都与非常相似的样式相关联的现象：

```
#andy, #rich, #jeremy, #james-box, #cennydd, #paul, #natalie, #sophie {  
    font-size: 1.6em;  
    font-weight: bold;  
    border: 1px solid #ccc;  
}
```

只需用一个普通的类替代它们：

```
.staff {  
    font-size: 1.6em;  
    font-weight: bold;  
    border: 1px solid #ccc;  
}
```

由于类具有灵活性，所以它们是非常强大的。同时，它们也可能被过度使用或滥用。CSS新手常常在几乎所有东西上添加类，从而试图更精细地控制它们的样式。早期的WYSIWYG编辑器也倾向于在应用样式的每个地方都添加类。许多开发人员在使用编辑器生成的代码学习CSS时继承了这个坏习惯。这种现象称为“多类症”(classitis)，在某种程度上，这和使用基于表格的布局一样糟糕，因为它在文档中添加了无意义的代码。

```
<h2 class="news-head">Andy wins an Oscar for his cameo in Iron Man</h2>
<p class="news-text">
  Andy Budd wins the Oscar for best supporting actor in Iron Man ←
  after his surprise cameo sets Hollywood a twitter with speculation.
</p>
<p class="news-text"><a href="news.php" class="news-link">More</a></p>
```

在前面的示例中，通过使用一个与新闻相关的类名，每个元素都被标识为新闻的一部分。这使新闻标题和正文可以采用与页面其他部分不同的样式。但是，不需要用这么多类来区分各个元素。可以将新闻条目放在一个部分中，并且加上类名news，从而标识整个新闻条目。然后，可以使用层叠(cascade)来识别新闻标题和文本。

```
<div class="news">
  <h2>Andy wins an Oscar for his cameo in Iron Man </h2>
  <p>Andy Budd wins the Oscar for best supporting actor in Iron Man ←
  after his surprise cameo sets Hollywood a twitter with speculation.</p>
  <p><a href="news.php">More</a></p>
</div>
```

只要你发现类名中出现了重复的单词，比如news-head和news-link或者section-head和section-foot，就应该考虑是否可以把这些元素分解成它们的组成部分。这会让代码更“组件化”，会大大提高灵活性。

以这种方式删除不必要的类有助于简化代码，使页面更简洁。稍后，将简要讨论CSS选择器和为样式寻找目标。无论如何，这种对类名的过度依赖是完全不必要的。如果你发现自己添加了许多类，那么这很可能意味着你的HTML文档的结构有问题。

## 5. div和span

有助于在文档中添加结构的一个元素是div元素。许多人误以为div元素没有语义。但是，div实际上代表部分(division)，它可以将文档分割为几个有意义的区域。所以，通过将主要内容区域包围在div中并分配content类，就可以在文档中添加结构和意义。

为了将不必要的标记减到最少，应该只在没有现有元素能够实现区域分割的情况下使用div元素。例如，如果使用主导航列表，就不需要将它包围在div中。

```
<div class="nav">
  <ul>
```

```

<li><a href="/home/">Home</a></li>
<li><a href="/about/">About Us</a></li>
<li><a href="/contact/">Contact</a></li>
</ul>
</div>

```

可以完全删除div，直接在列表上应用类：

```

<ul class="nav">
<li><a href="/home/">Home</a></li>
<li><a href="/about/">About Us</a></li>
<li><a href="/contact/">Contact</a></li>
</ul>

```

过度使用div常常称为“多div症”(divitus)，这是代码结构不合理而且过分复杂的一个信号。一些CSS新手会尝试用div重建自己原来的表格结构。但是，这只是用一套不必要的标签替换了另一套不必要的标签。实际上，应该使用div根据条目的意义或功能（而不是根据它们的表现方式或布局）对相关条目进行分组。

div可以用来对块级元素进行分组，而span可以用来对行内元素进行分组或标识：

```

<h2>Andy wins an Oscar for his cameo in Iron Man </h2>
<p>Published on <span class="date">February 22nd, 2009</span>
by <span class="author">Harry Knowles</span></p>

```

尽管目标是让代码尽可能简洁且有意义，但是有时候为了以自己希望的方式显示页面，无法避免添加额外的无语义的div或span。如果是这样，那也不必过分为此担心。我们正处在一个过渡时期，CSS 3有望提供更强大的文档控制能力。而且，实际需要常常比理论出现得早。关键是要知道在什么时候必须进行折中，并且要有正确的原因进行折中。

## 6. 微格式

由于HTML中缺少相应的元素，很难突出显示人、地点或日期等类型的信息。为了解决这个问题，有一组开发人员决定开发一套标准的命名约定和标记模式来表示这些数据。这些命名约定基于vCard和iCalendar等现有的数据格式，现在称为微格式(microformat)。下面以我的联系信息为例，代码采用hCard格式。

```

<div class="vcard">
<p><a class="url fn" href="http://andybudd.com/">Andy Budd</a>
<span class="org">Clearleft Ltd</span>
<a class="email" href="mailto:info@andybudd.com">info@andybudd.com</a> ↵
</p>
<p class="adr">
<span class="locality">Brighton</span>,
<span class="country-name">England</span>
</p>
</div>

```

微格式让我们可以以一种特定的方式标记数据，让其他程序和服务可以访问它。一些人编写了脚本，可以提取出以hCalendar格式标记的事件信息，并把它直接导入日历应用程序（见图1-5和图1-6）。

**Day 2 – Workshops**

Time	Event	Speaker	Description
9:00 – 12:00	Information Architecture: Just the Essentials	Donna Spencer	This half-day workshop will cover the essential aspects of Information Architecture. No fluff; no fluff, just pure IA.
12:30 – 13:30	Lunch		
13:30 – 17:00	Designing for Content-Rich Sites	Jared Spool	At the end of this workshop, you'll know exactly what you need to do to greatly enhance the usability of your content-rich site.
	Designing for people	Donna Spencer	This half-day workshop teaches a set of fundamental principles that are useful for all types of design work – information architecture, interaction design, visual design and even industrial design.
	Product Strategy and Planning Tools	Peter Merholz	Successful user experience design requires planning and strategic thinking. What are the new design methods that will ensure success in your work?

Add the schedule to your calendar: [Subscribe](#)

图1-5 UX伦敦会议的日程以hCalendar格式标记

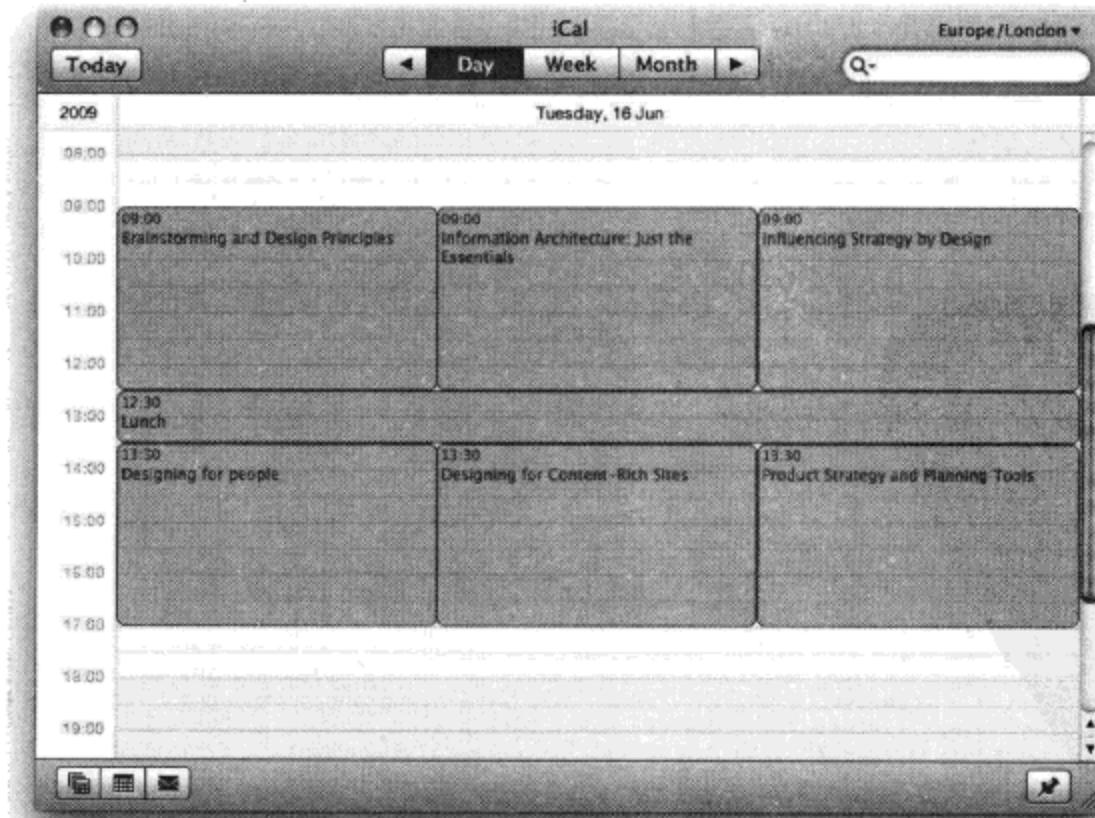


图1-6 这意味着访问者可以通过鼠标操作把整个日程添加到他们的日历中

还有一些人编写了插件，让Firefox可以提取出以hCard格式标记的联系信息，并通过蓝牙发送到手机上（见图1-7和图1-8）。

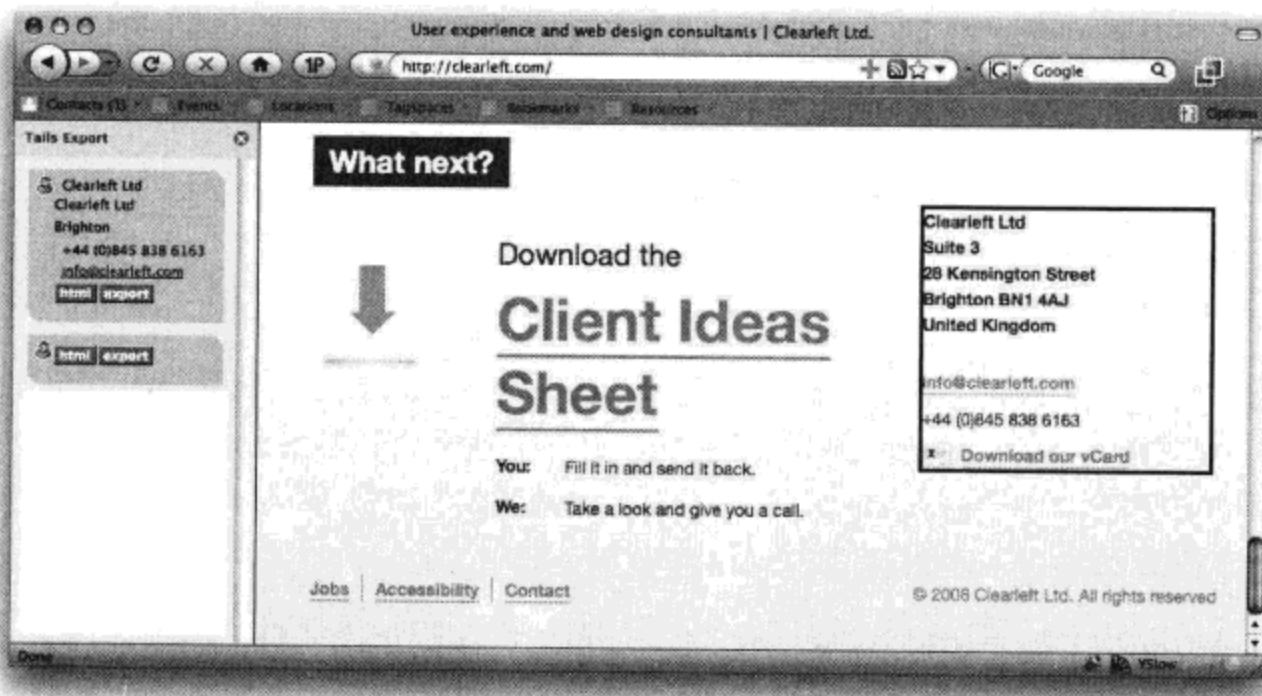


图1-7 Clearleft网站上的联系信息也以hCard格式标记

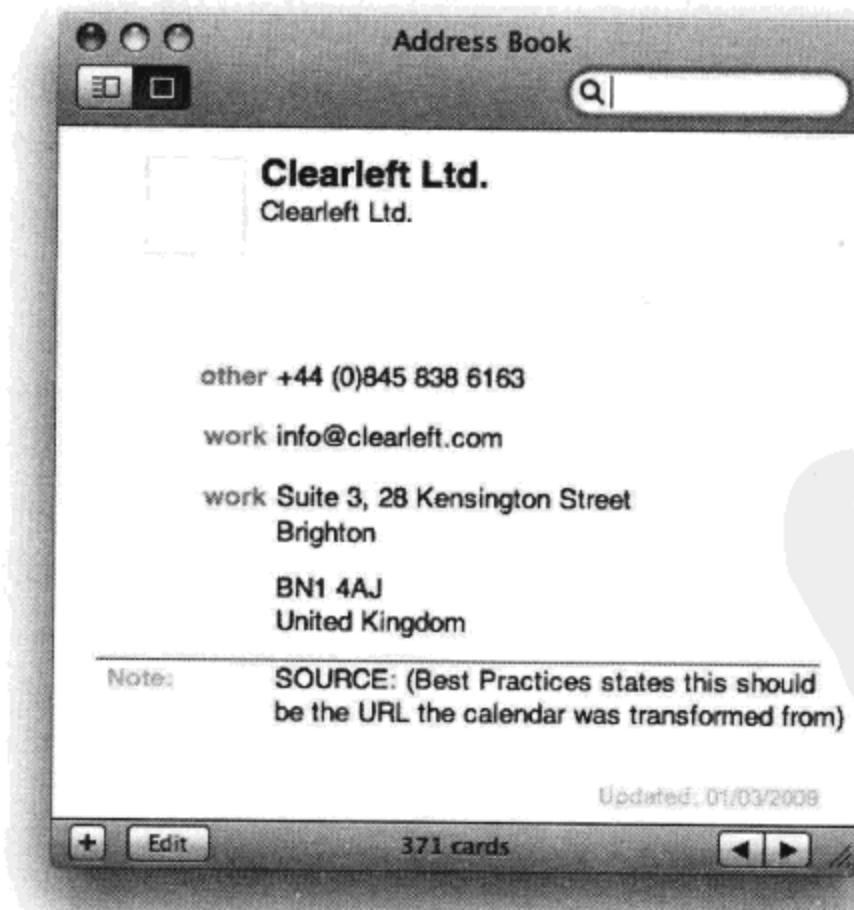


图1-8 在Firefox上，可以使用Operator或比较老的Tails附加组件把这些联系信息直接发送到地址簿

当前有9个正式的微格式，还有14个格式草案。包括：

- 用于日期、日历和事件的hCalendar
- 用于人和组织的hCard
- 用于人之间的关系的XFN
- 用于产品说明的hProduct（草案）
- 用于原料和烹调步骤的hRecipe（草案）
- 用于产品和事件审查的hReview（草案）
- 用于博客帖子等片段式内容的hAtom（草案）

许多大型网站已经支持微格式了。例如，Google Maps在地图搜索结果中使用hCard格式标记地址信息。类似地，Yahoo!的许多特性支持微格式，包括流行的Flickr照片共享网站。Yahoo!的Kelkoo购物搜索引擎中使用hListing格式，发布了2600万条使用微格式的信息。在网站上添加采用微格式的数据非常容易，所以我建议尽可能使用微格式。

这里只简要介绍了微格式可以实现的功能。如果需要了解更多信息，请阅读John Allsopp撰写的*Microformats: Empowering Your Mark-up for Web 2.0*，还可以在<http://microformats.org>查阅正式规范。

## 7. 不同的HTML和CSS版本

CSS有多个版本，所以知道要使用哪个版本是很重要的。CSS 1在1996年末成为推荐标准，其中包含非常基本的属性，比如字体、颜色、外边距。CSS 2在1998年发布，它添加了高级概念（比如浮动和定位）以及高级的选择器（比如子选择器、相邻同胞选择器和通用选择器）。

万维网联盟（W3C）的行动非常缓慢，所以尽管CSS 3的开发工作在新千年到来之前就开始了，但是距离最终的发布还有相当长的路要走。为提高开发和浏览器实现的速度，CSS 3被分割为多个模块，这些模块可以独立发布和实现。CSS 3包含一些令人兴奋的新特性，包括一个高级布局模块、全新的背景属性和一批新的选择器。其中一些模块原计划于2009年下半年发布。但是，目前还没有什么进展，而且一些模块在发布前夕又回到了“最后公告”或“工作草案”状态，所以很难确定实际发布会多少个模块。希望到2011年会有许多模块成为正式的推荐标准。更让人担心的是，一些模块还没有开始开发，其他模块好几年没有更新了。如此缓慢的开发进度让人觉得CSS 3似乎不可能全部完成了。

不过，尽管有一些拖延，还是有许多浏览器厂商已经实现了CSS 3规范中比较有意思的一些特性。因此，目前已经可以使用许多新的选择器了。

因为预期从CSS 2到CSS 3的发布之间时间会很长，2002年启动了CSS 2.1的开发。这是CSS 2的修订版，它计划纠正一些错误，并且更精确地描述CSS的浏览器实现。CSS 2.1正在逐渐完成，因此我建议使用这个CSS版本。

HTML 4.01于1999年末成为推荐标准，这是大多数人目前使用的HTML版本。2000年1月，

W3C发布了HTML 4.01的XML版并命名为XHTML 1.0。XHTML 1.0和HTML 4.01之间的主要差异是它遵守XML编码约定。这意味着与常规的HTML不同，所有XHTML属性必须包含引号，所有元素必须是封闭的。因此，以下代码在HTML中是合法的，而在XHTML中不合法：

```
<h2>Peru Celebrates Guinea Pig festival
<p><img src=pigonastick.jpg alt=Roast Guinea Pig>
<p>Guinea pigs can be fried, roasted, or served in a casserole.
```

在XHTML 1.0中，必须写成下面这样：

```
<h2>Peru Celebrates Guinea Pig festival</h2>
<p></p>
<p>Guinea pigs can be fried, roasted, or served in a casserole.</p>
```

XHTML 1.1比XHTML 1.0更接近XML。这两种语言之间没有实际的差异。但是，有一个重大的概念性差异。XHTML 1.0页面可以作为HTML文档，而XHTML 1.1页面是作为XML发送给浏览器的。这意味着，即使XHTML 1.1页面只包含一个错误（比如未编码的&符号），Web浏览器也不会显示页面。大多数网站开发人员显然不喜欢这样，所以XHTML 1.1不受欢迎。

是把XHTML 1.0页面作为HTML提供，还是应该坚持使用HTML 4.01？对于这个问题仍然有争论。但是，显然不应该使用XHTML 1.1，除非使用正确的mime类型，而且能够容忍一个错误导致整个页面无法显示。

HTML 5相对较新，是一个不断变动的规范草案。但是，它的发展势头很强，一些流行的浏览器已经开始着手支持它了。XHTML 2的开发非常缓慢而且过时了，让开发人员失去了信心。因此，一批开发人员决定开发自己的规范，由此催生出了HTML 5。这一行动非常成功，HTML 5已经成为正式的W3C项目，而XHTML 2的开发则停止了。

前面提到过，HTML 5的目标是建立一种现代的标记语言，可以更好反映在Web上发布的信息类型。因此它引入了新的结构性元素，比如header、nav、article、sections和footer。它还包含一批新的表单特性，可以大大简化Web应用程序开发。

### 1.1.2 文档类型、DOCTYPE 切换和浏览器模式

DTD（文档类型定义）是一组机器可读的规则，它们定义XML或HTML的特定版本中允许有什么，不允许有什么。在解析网页时，浏览器将使用这些规则检查页面的有效性并且采取相应的措施。浏览器通过分析页面的DOCTYPE声明来了解要使用哪个DTD，由此知道要使用HTML的哪个版本。

DOCTYPE声明是指HTML文档开头处的一行或两行代码，它描述使用哪个DTD。在下面的示例中，要使用的DTD是XHTML 1.0 Strict的DTD：

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

DOCTYPE通常——但不总是——包含指定的DTD文件的URL。例如，HTML 5就不需要URL。浏览器一般不读取这些文件，而是只识别常见的DOCTYPE声明。

DOCTYPE当前有两种风格，严格（strict）和过渡（transitional）。顾名思义，过渡DOCTYPE的目的是帮助开发人员从老版本迁移到新版本。因此，HTML 4.01和XHTML 1.0的过渡版本仍然允许使用已经废弃的元素，比如font元素。但这些语言的严格版本禁止使用废弃的元素，从而把内容和表现分隔开。

### 1.1.3 有效性验证

除了根据语义加标记之外，HTML文档还需要用有效的代码来编写。如果代码是无效的，浏览器会尝试解释标记本身，有时候会产生错误的结果。更糟的是，如果发送具有正确的MIME类型的XHTML文档，理解XML的浏览器将不显示无效的页面。因为浏览器需要知道要使用什么DTD才能正确地处理页面，所以对页面进行有效性验证时要求有DOCTYPE声明。

可以使用W3C验证器（一个验证器bookmarklet）或Firefox Web Developer Extension等插件检查HTML是否是有效的。许多HTML编辑器现在内置了验证器，还可以在计算机上本地安装W3C验证器。验证器会指出页面是否是有效的，如果是无效的页面，它还会指出原因是什么（见图1-9）。

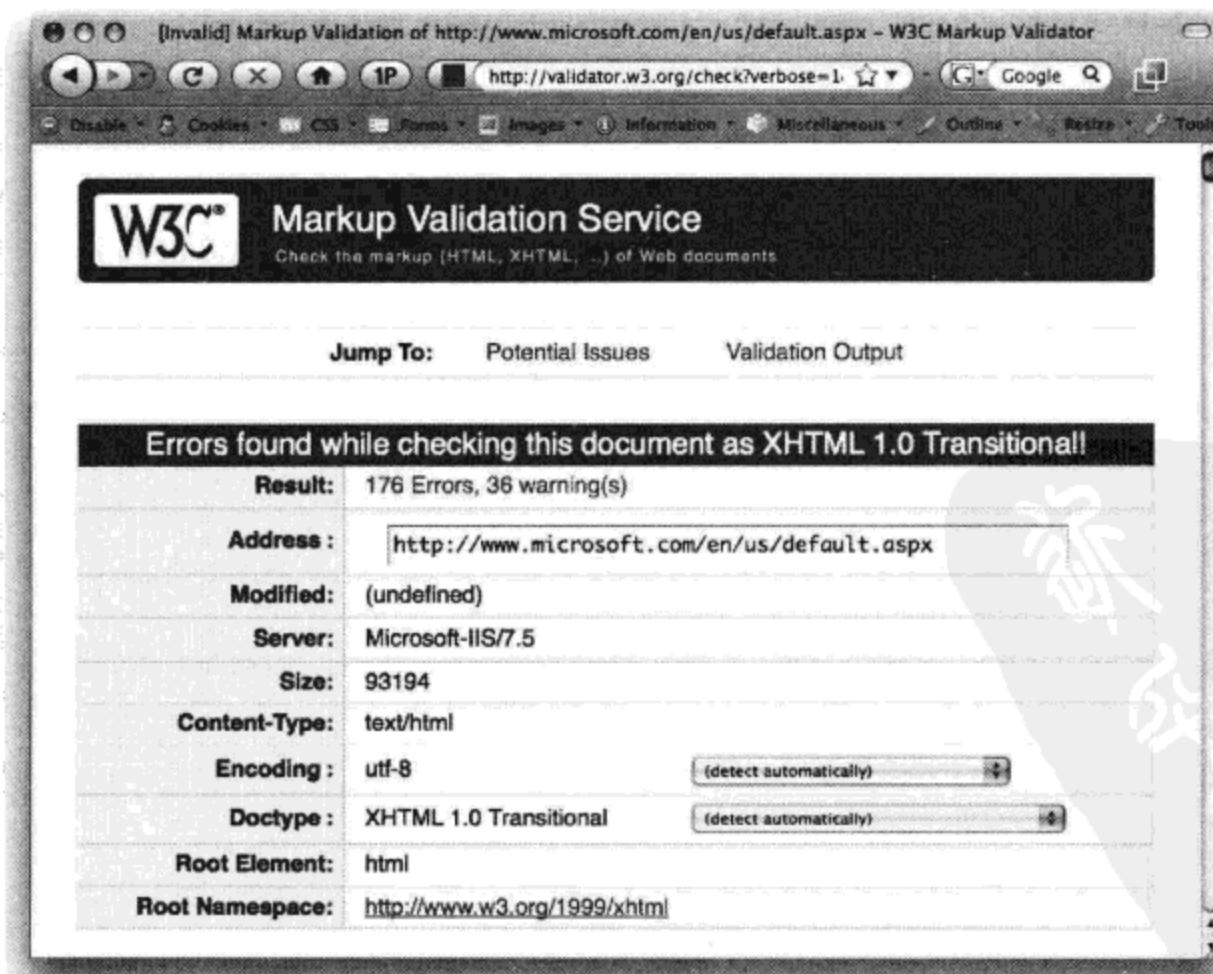


图1-9 microsoft.com主页有176个HTML错误和36个CSS错误

有效性验证很重要，因为它有助于找到代码中的bug。因此，尽早和经常进行有效性验证是个好习惯。但是，有效性验证太苛刻了，许多好页面由于很小的错误（比如&符号没有编码）或者因为遗留的内容而无法通过有效性验证。所以，尽管有效性验证很重要，但是在实际中，我们需要有一定的常识。

代码验证工具有很多种。可以访问 <http://validator.w3.org/> 并且输入自己的 URL 来对自己的站点进行在线验证。但是，如果要经常进行验证（这是个好习惯），那么每次都输入自己的 URL 有点儿麻烦。我使用的是一个方便的验证器 bookmarklet（也称 favelet），这是一小段可以存储在浏览器的书签或收藏夹中的 JavaScript。单击这个书签就会触发 JavaScript 动作。对于验证器 bookmarklet，它通过 W3C 验证器运行当前页面并显示结果。在 <http://favelets.com> 上可以找到这个验证器 bookmarklet 和其他许多方便的 Web 开发 bookmarklet。

如果使用的浏览器是 Firefox，那么可以下载和安装各种插件。在大量验证器插件中，我个人喜欢的是 Web Developers Extension 插件。除了可以验证 HTML 和 CSS 之外，它还可以执行许多其他任务，比如描述各种 HTML 元素、关闭样式表以及在浏览器中编辑样式。可以从 <http://chrispederick.com/work/firefox/web-developer/> 下载 Firefox Web Developers Extension，这是使用 Firefox 的 CSS 开发人员必须具备的插件。另一个出色的工具是 Firefox Validator Extension，可以从 <http://users.skynet.be/mgueury/mozilla/> 下载它。

还有一个用于 IE 6 和 IE 7 的开发人员工具栏，可以从 <http://tinyurl.com/7mnyh> 下载。尽管它的特性不如 Firefox 工具栏那么丰富，但是仍然非常有用。IE 8 浏览器中直接构建了一套开发工具，Safari 4 也是如此。

DOCTYPE 声明除了对有效性验证很重要之外，还被浏览器用于另一个用途。

## 1. 浏览器模式

当浏览器厂商开始创建与标准兼容的浏览器时，他们希望确保向后兼容性。为了实现这一点，他们创建了两种呈现模式：标准模式和混杂模式（quirks mode）。在标准模式中，浏览器根据规范呈现页面；在混杂模式中，页面以一种比较宽松的向后兼容的方式显示。混杂模式通常模拟老式浏览器（比如 Microsoft IE 4 和 Netscape Navigator 4）的行为以防止老站点无法工作。

对于这两种模式之间的差异，最显著的一个例子与 Windows 上 IE 专有的盒模型有关。在 IE 6 出现时，在标准模式中使用的是正确的盒模型，在混杂模式中使用的则是老式的专有盒模型。为了维持对 IE 5 和更低版本的向后兼容性，Opera 7 和更高版本也在混杂模式中使用有缺点的 IE 盒模型。

呈现方面的其他差异比较小，而且是与特定浏览器相关的，包括对于十六进制颜色值不需要 # 号、假设 CSS 中没有指定单位的长度的单位是像素，以及在使用关键字时将字号增加一级。

Mozilla和Safari还有第三种模式，称为“几乎标准的模式（almost standards mode）”，除了在处理表格的方式上有一些细微的差异之外，这种模式与标准模式相同。

在Firefox中，可以使用Web Developer Extension查看页面的呈现模式。如果网站以标准模式呈现，工具栏上会显示一个绿色的钩；若以混杂模式呈现，则显于红色的叉。IE 8中的开发工具也显示浏览器使用的模式。

## 2. DOCTYPE切换

浏览器根据DOCTYPE是否存在以及使用的哪种DTD来选择要使用的呈现方法。如果XHTML文档包含形式完整的DOCTYPE，那么它一般以标准模式呈现。对于HTML 4.01文档，包含严格DTD的DOCTYPE常常导致页面以标准模式呈现。包含过渡DTD和URI的DOCTYPE也导致页面以标准模式呈现，但是有过渡DTD而没有URI会导致页面以混杂模式呈现。DOCTYPE不存在或形式不正确会导致HTML和XHTML文档以混杂模式呈现。

根据DOCTYPE是否存在选择呈现模式，被称为DOCTYPE切换或DOCTYPE侦测。并非所有浏览器都采用这些规则，但是这些规则很好地说明了DOCTYPE切换的工作方式。要了解更全面的内容，可查阅网站<http://hsivonen.iki.fi/doctype/>，这里的图表说明了不同浏览器如何根据DOCTYPE声明来选择呈现方法。

DOCTYPE切换是浏览器用来区分遗留文档和符合标准的文档的手段。无论是否编写了有效的CSS，如果选择了错误的DOCTYPE，那么页面就将以混杂模式呈现，其行为就可能不会有错误或不可预测。因此，一定要在站点的每个页面上包含形式完整的DOCTYPE声明，并且在使用HTML时选择严格的DTD。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
      "http://www.w3.org/TR/html4/strict.dtd">  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
  
<!DOCTYPE html>
```

许多HTML编辑器会自动添加DOCTYPE声明。如果创建XHTML文档，一些老的编辑器还可能在DOCTYPE声明前面添加XML声明：

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

XML声明是XML文件使用的可选声明，它定义使用的XML版本和字符编码类型等设置。不幸的是，如果DOCTYPE声明不是页面上的第一个元素，那么IE 6会自动切换到混杂模式。这个问题在IE 7中已经纠正了，但是除非要将页面用做XML文档，否则最好避免使用XML声明。

## 1.2 小结

在本章中，你了解了语义性命名约定和有意义的标记如何让代码更容易阅读和维护，还了解了ID和类名之间的差异，以及各自适用的场合。我们还介绍了CSS和HTML可用的版本，以及浏览器决定如何处理这些差异的方式。

在下一章中，我们将回顾一些基本的CSS选择器，学习一些新的CSS 3选择器，还要学习特性和层叠，以及如何通过组织和计划样式表来简化维护。





## 第2章

# 为样式找到应用目标

---

有效且结构良好的文档为你要应用的样式提供了一个框架。要想使用CSS将样式应用于特定的HTML元素，需要想办法找到这个元素。在CSS中，执行这一任务的样式规则部分称为选择器(selector)。

在本章中，我们学习以下内容。

- 常用的选择器
- 高级选择器
- 新的CSS 3选择器
- 特殊性和层叠的作用
- 计划和维护样式表
- 如何在代码中加注释

## 2.1 常用的选择器

最常用的选择器类型是类型选择器和后代选择器。类型选择器用来寻找特定类型的元素，比如段落或标题元素，只需指定希望应用样式的元素的名称。类型选择器有时候也称为元素选择器或简单选择器。

```
p {color: black;}  
h1 {font-weight: bold;}
```

后代选择器可用来寻找特定元素或元素组的后代。后代选择器由其他两个选择器之间的空格表示。在这个示例中，只缩进是块引用的后代的段落元素，其他所有段落不受影响。

```
blockquote p {padding-left: 2em;}
```

这两种选择器适合于应用那些应用范围广的一般性样式。要想寻找更特定的元素，可以使用ID选择器和类选择器。顾名思义，这两种选择器用于寻找那些具有指定ID或类名的元素。ID选择器由一个#字符表示，类选择器由一个点号表示。下面示例中的第一条规则使简介段落中的文本以粗体显示，第二条规则让日期显示为绿色：

```
#intro {font-weight: bold;}  
.date-posted {color: #ccc;}  
  
<p id="intro">Happy Birthday Andy</p>  
<p class="date-posted">24/3/2009</p>
```

前面提到过，许多CSS开发人员过度依赖类选择器和ID选择器。如果他们希望以一种方式对主要内容区域中的标题应用样式，而在第二个内容区域中采用另一种方式，那么他们很可能创建两个类并且在每个标题上应用一个类。一种简单得多的方法是结合使用类型、后代、ID和类这几种选择器：

```
#main-content h2 {font-size: 1.8em;}  
#secondaryContent h2 {font-size: 1.2em;}  
  
<div id="main-content">  
  <h2>Articles</h2>  
  ...  
</div>  
<div id="secondary-content">  
  <h2>Latest news</h2>  
  ...  
</div>
```

这是一个非常简单明朗的示例。但是，让你吃惊的是，只使用前面讨论的4种选择器就可以成功地找到许多元素。如果你发现自己在文档中添加了许多不必要的类，那么这可能是文档结构不合理的一个警告信号。这时应该分析这些元素之间的差异。你常常会发现唯一的差异是它们在页面上出现的位置。不要给这些元素指定不同的类，而应将一个类或ID应用于它们的祖先，然后使用后代选择器定位它们。

## 伪类

有时候，我们需要根据文档结构之外的其他条件对元素应用样式，例如表单元素或链接的状态。这要使用伪类选择器来完成。

```
/* makes all unvisited links blue */  
a:link {color:blue;}
```

```
/* makes all visited links green */
a:visited {color:green;}

/* makes links red when hovered or activated.
focus is added for keyboard support */
a:hover, a:focus, a:active {color:red;}

/* makes table rows red when hovered over */
tr:hover {background-color: red;}

/* makes input elements yellow when focus is applied */
input:focus {background-color:yellow;}
```

:link和:visited称为链接伪类，只能应用于锚元素。:hover、:active和:focus称为动态伪类，理论上可以应用于任何元素。大多数浏览器都支持这个功能。但是，IE 6只注意应用于锚链接的:active和:hover选择器，完全忽略:focus。IE 7在任何元素上都支持:hover，但是忽略:active和:focus。

最后，值得指出的是：通过把伪类连接在一起，可以创建更复杂的行为，比如在已访问链接和未访问链接上实现不同的鼠标悬停效果。

```
/* makes all visited linkes olive on hover */
a:visited:hover {color:olive;}
```

## 2.2 通用选择器

通用选择器可能是所有选择器中最强大却最少使用的。通用选择器的作用就像是通配符，它匹配所有可用元素。与其他语言中的通配符一样，通用选择器由一个星号表示。通用选择器一般用来对页面上的所有元素应用样式。例如，可以使用以下规则删除每个元素上默认的浏览器内边距和外边距：

```
* {
  padding: 0;
  margin: 0;
}
```

在与其他选择器结合使用时，通用选择器可以用来对某个元素的所有后代应用样式，或者跳过一级后代。在本章稍后将看到这么做的实际效果。

## 2.3 高级选择器

CSS 2.1和CSS 3有其他许多有用的选择器。不过，虽然大多数现代浏览器支持这些高级选择

器，但是IE 6和更低版本不支持。好在在创建CSS时考虑到了向后兼容性。如果浏览器不理解某个选择器，那么它会忽略整个规则。因此，可以在现代浏览器中应用样式和易用性方面的改进，同时不必担心它在老式浏览器中造成问题。但是要记住，在对于站点功能或布局很重要的任何元素上，都应该避免使用这些高级选择器。

### 2.3.1 子选择器和相邻同胞选择器

第一个高级选择器是子选择器。后代选择器选择一个元素的所有后代，而子选择器只选择元素的直接后代，即子元素。在下面的示例中，外层列表中的列表项显示一个定制的图标，而嵌套列表中的列表项不受影响（见图2-1）：

```
#nav>li {
    background: url(folder.png) no-repeat left top;
    padding-left: 20px;
}

<ul id="nav">
    <li><a href="/home/">Home</a></li>
    <li><a href="/services/">Services</a>
        <ul>
            <li><a href="/services/design/">Design</a></li>
            <li><a href="/services/development/">Development</a></li>
            <li><a href="/services/consultancy/">Consultancy</a></li>
        </ul>
    </li>
    <li><a href="/contact/">Contact Us</a></li>
</ul>

    ■ Home
    ■ Services
        Design
        Development
        Consultancy
    ■ Contact Us
```

图2-1 子选择器指定列表的子元素的样式，但是不影响它的孙元素

IE 7和更高版本都支持子选择器。但是，在IE 7中有一个小bug，如果父元素和子元素之间有HTML注释，就会出问题。

在IE 6和更低版本中，可以使用通用选择器模拟子选择器的效果。为此，先在所有后代上应用你希望子元素具有的样式。然后，使用通用选择器覆盖子元素的后代上的样式。所以，要实现与前面的子选择器示例相同的效果，可以这样做：

```
#nav li {
    background: url(folder.png) no-repeat left top;
```

```

    padding-left: 20px;
}

#nav li * {
    background-image: none;
    padding-left: 0;
}

```

有时，你可能需要根据一个元素与另一个元素的相邻关系对它应用样式。相邻同胞选择器可用于定位同一个父元素下某个元素之后的元素。可以使用相邻同胞选择器让顶级标题后面的第一个段落显示为粗体、灰色，并且字号比后续段落略微大一点儿（见图2-2）。

```

h2 + p {
    font-size: 1.4em;
    font-weight: bold;
    color: #777;
}

<h2>Peru Celebrates Guinea Pig festival</h2>
<p>The guinea pig festival in Peru is a one day event to celebrate
these cute local animals. The festival included a fashion show where
animals are dressed up in various amusing costumes.</p>
<p>Guinea pigs can be fried, roasted, or served in a casserole. Around
65 million guinea pigs are eaten in Peru each year.</p>

```

## Peru Celebrates Guinea Pig festival

The guinea pig festival in Peru is a one day  
event to celebrate these cute local animals.  
The festival included a fashion show where  
animals are dressed up in various amusing  
costumes.

Guinea pigs can be fried, roasted or served in a casserole.  
Around 65 million guinea pigs are eaten in Peru each year.

图2-2 可以使用相邻同胞选择器对标题后面的第一个段落应用样式，这样就不需要使用多余的类

与子选择器一样，如果在目标元素之间有注释，这在IE 7中也会出问题。

### 2.3.2 属性选择器

顾名思义，属性选择器可以根据某个属性是否存在或属性的值来寻找元素，因此能够实现某些非常有意思和强大的效果。

例如，当鼠标悬停在具有title属性的元素上时，大多数浏览器会显示一个工具提示。可以

使用这种特性解释某些内容（比如首字母缩拼词和缩写词）的含义：

```
<p>The term <acronym title="self-contained underwater breathing apparatus">SCUBA</acronym> is an acronym rather than an abbreviation as it is pronounced as a word.</p>
```

但是，如果不把鼠标悬停在这个元素上，那么没有任何迹象能够表明存在这一额外信息。为了解决这个问题，可以使用属性选择器对具有title属性的acronym元素应用与其他元素不同的样式——在下面的示例中，在它们下面加上点。还可以在鼠标悬停在这个元素上时将鼠标指针改问号，表示这个元素与众不同，从而提供更多的上下文相关信息。

```
acronym[title] {  
    border-bottom: 1px dotted #999;  
}  
  
acronym[title]:hover, acronym[title]:focus {  
    cursor: help;  
}
```

除了根据某个属性是否存在对元素应用样式之外，还可以根据属性值应用样式。例如，使用rel属性值nofollow链接的站点无法从Google获得评级收益（ranking benefit）。以下规则在这种链接旁边显示一个图像，以此表示不推荐这个目标站点：

```
a[rel="nofollow"] {  
    background: url(nofollow.gif) no-repeat right center;  
    padding-right: 20px;  
}
```

包括IE 7的现代浏览器都支持这些选择器。然而，由于IE 6不支持属性选择器，可以利用它们对IE 6应用一种样式，对更符合标准的浏览器应用另一种样式。例如，Andy Clarke利用这种技术为IE 6提供网站的黑白版本（见图2-3），为其他所有浏览器提供彩色版本（见图2-4）。

```
#header {  
    background: url(branding-bw.png) repeat-y left top;  
}  
#id="header" {  
    background: url(branding-color.png) repeat-y left top;  
}
```

一些属性可以有多个值，值之间用空格分隔。属性选择器允许根据属性值之一寻找元素。例如，XFN微格式允许在锚链接的rel属性中添加关键字来定义你与站点的关系。假设某个站点属于我的同事，我就可以在博客上的链接中添加co-worker关键字来表示这一关系。然后可以在这位同事的姓名旁边显示一个特殊的图标，以此表明我和这个人一起工作。



图2-3 Andy Clarke使用属性选择器和其他技术为IE 6提供网站的黑白版本

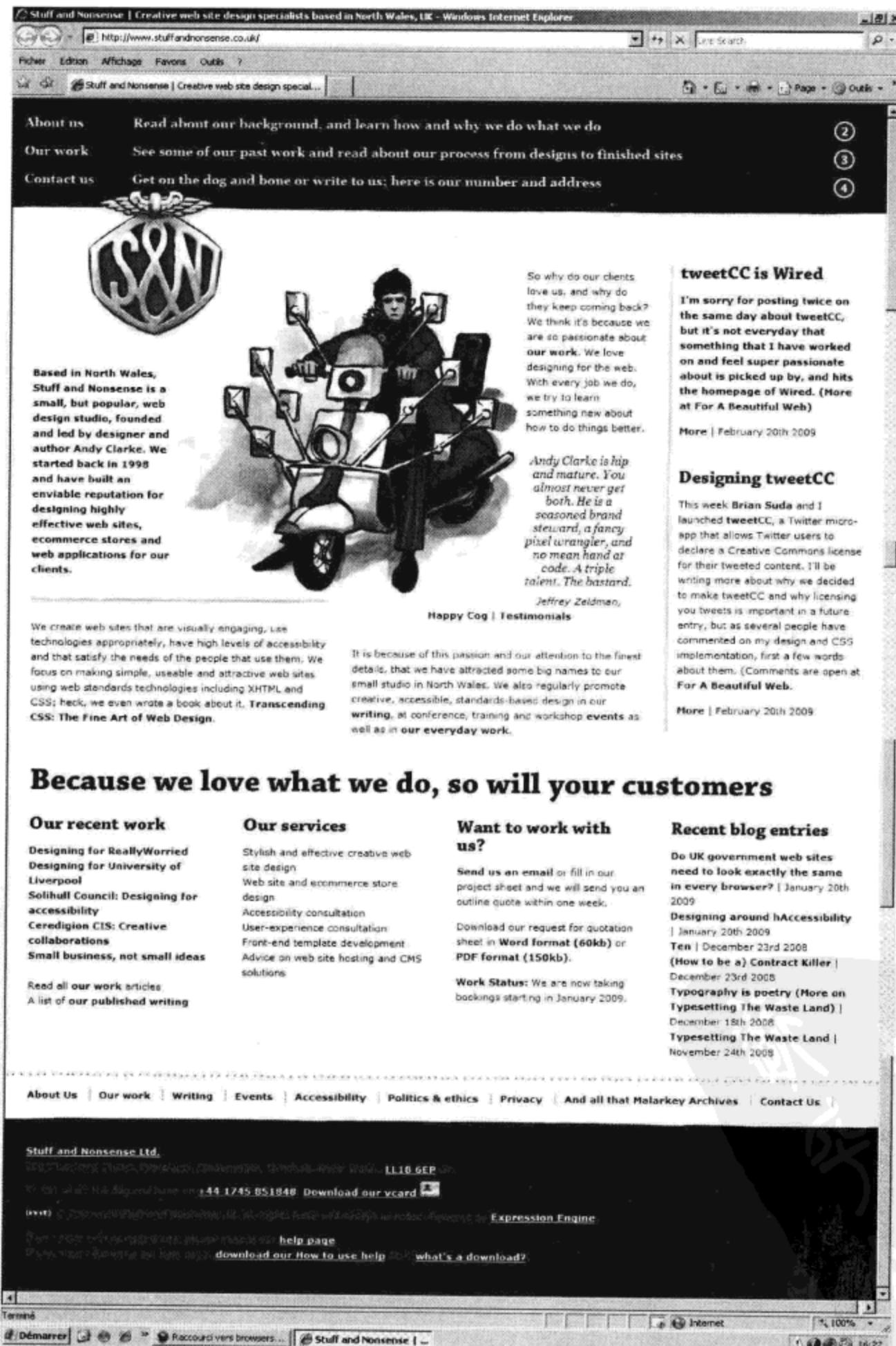


图2-4 更现代的浏览器显示的彩色版本

```
.blogroll a[rel~="co-worker"] {  
    background: url(co-worker.gif) no-repeat left center;  
}  
  
<ul class="blogroll">  
    <li>  
        <a href="http://adactio.com/" rel="friend met colleague co-worker"> ←  
            Jeremy Keith</a>  
    </li>  
    <li>  
        <a href="http://clagnut.com/" rel="friend met colleague co-worker"> ←  
            Richard Rutter</a>  
    </li>  
    <li>  
        <a href="http://hicksdesign.com/" rel="friend met colleague"> ←  
            John Hicks</a>  
    </li>  
    <li>  
        <a href="http://stuffandnonsense.co.uk/" rel="friend met colleague"> ←  
            Andy Clarke</a>  
    </li>  
</ul>
```

### 2.3.3 层叠和特殊性

即使在不太复杂的样式表中，要寻找同一元素可能有两个或更多规则。CSS通过一个称为层叠（cascade）的过程处理这种冲突。层叠给每个规则分配一个重要度。作者的样式表是由站点开发者编写的，被认为是最重要的样式表。用户可以通过浏览器应用自己的样式，这些样式表的重要度低一级。最后是浏览器或用户代理使用的默认样式表，它们的重要度是最低的，所以你总是可以覆盖它们。为了让用户有更多的控制能力，可以通过将任何规则指定为!important来提高它的最重要度，让它优先于任何规则，甚至优先于作者加上!important标志的规则。这种设置能够满足特殊的可访问性需求，例如如果用户有某种读写障碍，可以使用中等对比度的用户样式表。

因此，层叠采用以下重要度次序。

- 标有!important的用户样式。
- 标有!important的作者样式。
- 作者样式。
- 用户样式。
- 浏览器/用户代理应用的样式。

然后，根据选择器的特殊性决定规则的次序。具有更特殊选择器的规则优先于具有一般选择器的规则。如果两个规则的特殊性相同，那么后定义的规则优先。

### 1. 特殊性

为了计算规则的特殊性，给每种选择器都分配一个数字值。然后，将规则的每个选择器的值加在一起，计算出规则的特殊性。可惜特殊性的计算不是以10为基数的，而是采用一个更高的未指定的基数。这能确保非常特殊的选择器（比如ID选择器）不会被大量一般选择器（比如类型选择器）所超越。但是，为了简化，如果在一个特定选择器中的选择器数量少于10个，那么可以以10为基数计算特殊性。

选择器的特殊性分成4个成分等级：a、b、c和d。

- 如果样式是行内样式，那么a = 1。
- b等于ID选择器的总数。
- c等于类、伪类和属性选择器的数量。
- d等于类型选择器和伪元素选择器的数量。

使用这些规则可以计算任何CSS选择器的特殊性。表2-1给出了一系列选择器以及相应的特殊性。

表2-1 特殊性示例

选择器	特 殊 性	以10为基数的特殊性
style=""	1,0,0,0	1000
#wrapper #content {}	0,2,0,0	200
#content .datePosted {}	0,1,1,0	110
div#content {}	0,1,0,1	101
#content {}	0,1,0,0	100
p.comment .dateposted {}	0,0,2,1	21
p.comment {}	0,0,1,1	11
div p {}	0,0,0,2	2
p {}	0,0,0,1	1

初看上去，上面的特殊性和更高的未指定的基数可能有点儿让人糊涂，所以再解释一下。基本上，用style属性编写的规则总是比其他任何规则特殊。具有ID选择器的规则比没有ID选择器的规则特殊，具有类选择器的规则比只有类型选择器的规则特殊。最后，如果两个规则的特殊性相同，那么后定义的规则优先。

在修复bug时特殊性极其重要，因为你需要了解哪些规则优先及其原因。例如，假设有以下这组规则，你认为两个标题会是什么颜色的？

```
#content div#main-content h2 {  
    color: gray;  
}  
#content #main-content>h2 {  
    color: blue;  
}
```

```

body #content div[id="main-content"] h2 {
    color: green;
}
#main-content div.news-story h2 {
    color: orange;
}
#main-content [class="news-story"] h2 {
    color: yellow;
}
div#main-content div.news-story h2.first {
    color: red;
}

<div id="content">
    <div id="main-content">
        <h2>Strange Times</h2>
        <p>Here you can read bizarre news stories from around the globe.</p>
        <div class="news-story">
            <h2 class="first">Bog Snorkeling Champion Announced Today</h2>
            <p>The 2008 Bog Snorkeling Championship was won by Conor Murphy ←
                with an impressive time of 1 minute 38 seconds.</p>
            </div>
        </div>
    </div>

```

令人吃惊的是，两个标题都是灰色的。第一个选择器由两个ID选择器组成，因此它具有最高的特殊性。后面一些选择器看起来更复杂，但是因为它们只包含一个ID，所以特殊性总是低于第一个选择器。

如果你遇到了似乎没有起作用的CSS规则，很可能是出现了特殊性冲突。请在你的选择器中添加它的一个父元素的ID，从而提高它的特殊性。如果这能够解决问题，就说明样式表中其他地方很可能有更特殊的规则，它覆盖了你的规则。如果是这种情况，你可能需要检查代码，解决特殊性冲突，让代码尽可能简洁。

## 2. 在样式表中使用特殊性

在编写CSS时特殊性非常有用，因为它可以对一般元素应用一般样式，然后在更特殊的元素上覆盖它们。例如，如果你希望站点上大多数文本是黑色的，但介绍说明文本是灰色的。可以这样做：

```

p {color: black;}
p.intro {color: grey;}

```

对于小网站，这很好。但是，在大型站点上，你会发现例外情况越来越多。例如，你可能希望新闻文章上的介绍文本是蓝色的，而主页上的介绍文本使用灰色背景。每当创建更特殊的样式

时，可能需要覆盖一些一般规则。这可能需要一些额外的代码。而且，因为元素可以从许多地方获得样式，情况可能变得非常复杂。

为了避免过分混乱，我尽量保持一般性样式非常一般，特殊样式尽可能特殊，这样就不需要覆盖特殊样式了。如果发现不得不多次覆盖一般样式，那么从更一般的规则中删除需要覆盖的声明，并且将它显式地应用于需要它的每个元素，这样可能比较简单。

### 3. 在主体标签上添加类或ID

一种有意思的特殊性用法是在主体（body）标签上应用类或ID。这样做之后，就可以根据页面或在站点范围内覆盖样式。例如，如果希望新的页面具有特殊的布局，那么可以在主页的主体元素上添加一个类名，并且使用它覆盖样式：

```
body.news {  
    /* do some stuff */  
}  
  
<body class="news">  
    <p>My, what a lovely body you have.</p>  
</body>
```

有时候，在特殊页面上需要覆盖这些样式，比如在新闻存档页面上。在这种情况下，可以在主体标签上添加ID来标识这个页面。

```
body.news {  
    /* do some stuff */  
}  
  
body#archive {  
    /* do some different stuff */  
}  
  
<body id="archive" class="news">  
    <p>My, what a lovely body you have.</p>  
</body>
```

使用类标识页面类型，使用ID标识特定页面，就可以非常灵活地控制站点的设计和布局。我很喜欢使用这种方法编写可维护的代码。

#### 2.3.4 继承

人们常常将继承和层叠混为一谈。尽管它们初看上去有点儿相似，但是这两个概念实际上是很不一样的。好在，继承是一个非常容易理解的概念。应用样式的元素的后代会继承样式的某些属性，比如颜色和字号。例如，如果将主体元素的文本颜色设置为黑色，那么主体元素的所有后

代也显示黑色的文本。对于字号，也是这样的。如果将主体的字号设置为1.4 em，那么页面上的所有内容应该会继承这个字号。我说“应该会”是因为Windows的IE和Netscape在继承表格中的字号方面有问题。为了解决这个问题，必须指定表格应该继承字号，或者在表格上单独设置字号。

如果在主体上设置字号，你会注意到页面上的任何标题都没有采用这个样式。你可能会认为标题没有继承文本字号。但是，实际上是浏览器的默认样式表设置了标题字号。直接应用于元素的任何样式总会覆盖继承而来的样式。这是因为继承而来的样式的特殊性为空。

继承这一性质非常有用，因为它使开发人员不必在元素的每个后代上添加相同的样式。如果打算设置的属性是继承而来的属性，那么也可以将它应用于父元素。可以编写：

```
p, div, h1, h2, h3, ul, ol, dl, li {color: black;}
```

但是下面的写法更简单：

```
body {color: black;}
```

正如恰当地使用层叠可以简化CSS，恰当地使用继承也可以减少代码中选择器的数量和复杂性。但是，如果大量元素继承各种样式，那么判断样式的来源就会变得困难。在Firefox中可以使用Firebug查明样式的来源（图2-5）。

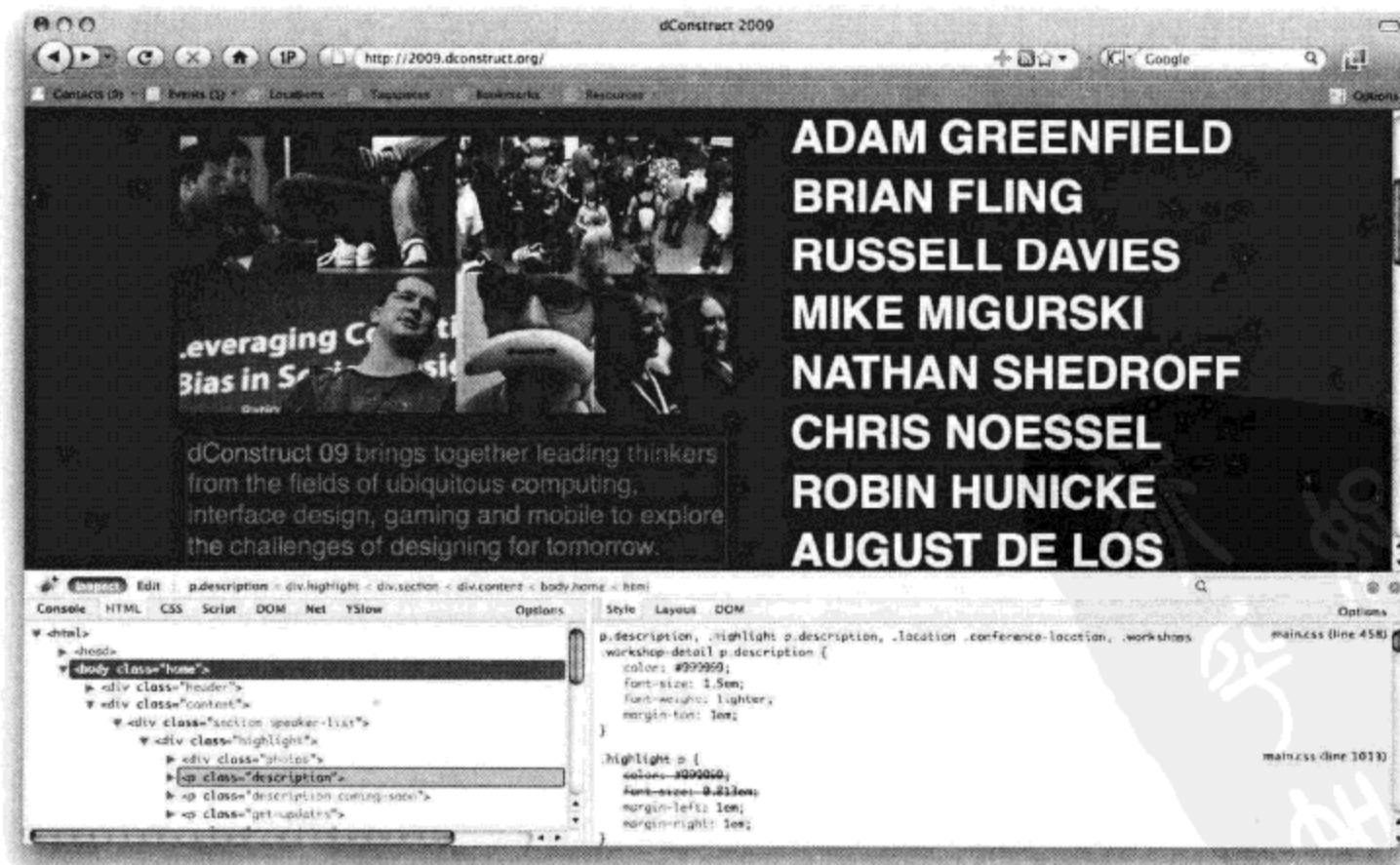


图2-5 Firebug是一个方便的Firefox附加组件，可以用它检查各个元素，查明它们的样式的来源

## 2.4 规划、组织和维护样式表

站点越大、越复杂，图形越丰富，CSS就越难管理。本节将讨论管理代码的方法，包括按逻辑对样式进行分组以及通过添加注释使代码更容易阅读。

### 2.4.1 对文档应用样式

可以将样式放在style标签之间，从而直接在文档头上添加样式，但是，这不是对文档应用样式的好方法。如果要创建使用相同样式的另一个页面，那么就不得不在新页面上复制CSS。如果以后要修改样式，那么就不得不在两处都进行修改。幸好CSS允许将所有样式放在一个或多个外部样式表中。将外部样式表附加到网页上有两种方法。可以链接它们，也可以导入它们：

```
<link href="/css/basic.css" rel="stylesheet" type="text/css" />
<style type="text/css">
<!--
@import url("/css/advanced.css");
-->
</style>
```

除了导入到HTML文档之外，还可以从另一个样式表导入样式表。因此，你可以从HTML页面链接基本样式表，然后将复杂的样式表导入这个样式表：

```
@import url(/css/layout.css);
@import url(/css/typography.css);
@import url(/css/color.css);
```

把CSS分割为多个样式表是一种常见的做法，我在本书的第1版中也推荐过这种方法。但是，最近的浏览器基准测试已经表明，导入样式表比链接样式表慢。

在使用多个CSS文件时，还有两个与速度相关的问题。首先，多个文件会导致从服务器发送更多数据包，这些数据包的数量（而不是内容）会影响下载时间。另外，浏览器只能从同一个域同时下载数量有限的文件。对于老式的浏览器，这个限制常常只是两个文件，现代浏览器把这个限制提到了8个。因此，如果有3个样式表，那么在老式浏览器中必须等下载完前两个文件，才能开始下载第三个。由于这些原因，使用结构良好的单一CSS文件可以显著提高下载速度。

使用单一CSS文件还能够把代码集中在一个地方。为了便于维护，我过去建议分割代码。但是，总是很难判断某一声明是与站点的布局相关，还是与排版相关。它常常与两者都相关，最终你会随意选择安置它的位置。分割代码还意味着必须打开多个样式表，经常要在文件之间切换。最现代的CSS编辑器中内置了代码折叠等特性，所以现在编辑单一文件非常容易。由于这些原因，我现在倾向于使用单一CSS文件而不是多个小文件。当然，应该根据站点的实际情况做出决定，没有硬性规定。

在编写自己的样式表时，你可能很清楚它们的结构、曾经遇到的问题以及为什么采用某种方式。但是，如果6个月之后再来看这个样式表，你很可能已经忘了许多事情。另外，你可能需要将自己的CSS交给其他人去实现，或者其他开发人员以后需要编辑你的代码。因此，对代码进行注释是一种好做法。

在CSS中添加注释非常简单。CSS注释以`/*`开头，以`*/`结束。这种注释称为C风格的注释，因为C语言中使用这种注释。注释可以是单行的，也可以是多行的，而且可以出现在代码中的任何地方。

```
/* Body Styles */  
body {  
    font-size: 67.5%; /* Set the font size */  
}
```

如果CSS文件非常长，那么寻找特定的样式会很困难。一种改进方法是在每个注释头中添加一个标志。这个标志仅仅是头文本前面的一个额外字符，一般不会出现在CSS文件中。搜索这个标志和注释头中的前几个字母，就可以立即找到要寻找的文件部分。所以，在下面的示例中，搜索“`@group typ`”就会立即找到样式表中的版式部分：

```
/* @group typography */
```

OS X编辑器CSS Edit就可使用这种格式创建一种简单但有效的样式表导航方法。

### 1. 设计代码的结构

为了便于维护，最好把样式表划分为几大块。显然，常常把最一般的规则放在最前面。这包括应用于`body`标记的、应该由站点上所有元素继承的样式。接下来是可能需要的所有全局`reset`样式，然后是链接、标题和其他元素。

完成一般样式之后，开始处理更特殊的样式和辅助样式。这些是在整个站点中使用的一般类，包括表单和错误消息等方面。然后，处理布局和导航等结构性元素。

随着在样式表中的移动，我们在一层样式上构建另一层样式，处理的样式越来越特殊。处理完页面结构元素之后，我们把注意力转到与特定页面相关的组件上。最后，在文档的底部处理覆盖和例外情况。整个文档的结构像下面这样：

- 一般性样式
  - 主体样式
  - `reset`样式
  - 链接
  - 标题
  - 其他元素

- 辅助样式
  - 表单
  - 通知和错误
  - 一致的条目
- 页面结构
  - 标题、页脚和导航
  - 布局
  - 其他页面结构元素
- 页面组件
  - 各个页面
- 覆盖

这里使用一种风格统一的大注释块分隔每个部分。

```
/* @group general styles
-----*/  
  
/* @group helper styles
-----*/  
  
/* @group page structure
-----*/  
  
/* @group page components
-----*/  
  
/* @group overrides
-----*/
```

并非所有东西都能够自然地分成定义明确的块，这需要开发人员进行判断。请记住，代码的分隔越细致、越合理，就越容易理解，而且能够更快地找到要寻找的规则。

因为我的许多CSS文件可能具有相似的结构，所以我可以创建一个预先加上注释的CSS模板供所有项目使用，从而节省时间。还可以添加在所有站点上都使用的常用规则，形成某种原型CSS文件，这可以节省更多的时间。这样的话，在开始新项目时就不必总是重复以前的工作。在本书的下载代码中可以找到一个原型CSS文件示例。

## 2. 自我提示

对于复杂的大型项目，在CSS文件中添加临时的注释常常对开发有帮助。这些注释可以提醒

你在启动前需要完成哪些工作，或者提醒你提供列宽度等常用值的查询表。

例如，如果设计中使用了许多种颜色，你可能常常需要到图形应用程序中查询十六进制值，然后再返回到文本编辑器。这非常浪费时间，所以一些人认为需要CSS变量。尽管这是一个很有意思的主意，但是这会让CSS更接近真正的编程语言，可能让非程序员畏惧它。因此，我常常使用一种简单的方法。我在样式表的开头添加一个小的颜色查询表，这样就可以在开发期间经常参考它。完成开发之后，我通常会删除它。

```
/* Color Variables

@colordef #434343; dark gray
@colordef #f2f6e4; light green
@colordef #90b11f; dark green
@colordef #369; dark blue
*/
```

为了使注释更有意义，可以使用关键字来区分重要的注释。我使用`@todo`来表示某些东西需要在以后进行修改、修复或复查，用`@bugfix`表示代码或特定浏览器遇到的问题，用`@workaround`表示并不完善的权宜之计：

```
/* :@todo Remember to remove this rule before the site goes live */
/* @workaround: I managed to fix this problem in IE by setting a small
negative margin but it's not pretty */
/* @bugfix: Rule breaks in IE 5.2 Mac */
```

用编程术语来说，这些关键字称为gotcha，它们在以后的开发阶段中非常有帮助。实际上，这些词汇都是CSSDoc项目（<http://cssdoc.net>）的一部分，这个项目的目的是为样式表开发一套标准的注释语法。

### 3. 删除注释和优化样式表

注释会使CSS文件显著加大。因此，你可能需要从样式表中去掉一些注释。许多HTML/CSS和文本编辑器都有“搜索并替换”选项，因此从代码中删除注释很容易。另外，还可以使用几种在线CSS优化器（比如[www.cssoptimiser.com/](http://www.cssoptimiser.com/)上提供的优化器）。优化器不但能够删除注释，还可以删除空白，这可以从代码中去掉额外的字节。如果要从当前使用的样式表中删除注释，一定要保留带注释的版本。管理这个过程的最好方法是创建一个部署脚本，当你让修改的内容在生产环境中生效时，它自动地删除注释。但是，因为这是一种高级技术，可能只适用于很大的复杂站点。

减小文件大小的最好方法可能是启用服务器端压缩。如果使用Apache服务器，那么应该安装`mod_gzip`或`mod_deflate`。所有现代浏览器都可以处理用GZIP压缩的文件并进行即时解压。这些Apache模块探测浏览器是否能够处理这种文件，如果可以，就发送压缩的版本。服务器端压缩能够将HTML和CSS文件减小大约80%，这可以减少对带宽的占用，大大加快页面的下载速度。如果无法使用这些Apache模块，那么仍然可以按照<http://tinyurl.com/8w9rp>上的教程的说明对文件

进行压缩。

### 2.4.2 样式指南

大多数Web站点有多个开发人员，而大型站点甚至有多个团队负责处理站点的不同方面。程序员、内容管理员和其他前端开发人员可能需要了解代码的元素和设计是如何工作的。因此，建立样式指南是一种非常好的做法。

样式指南可以是一个文档、网页或小型站点，它们解释代码和站点的视觉设计是如何组合在一起的。最基本的样式指南应该描述一般性设计原则，比如如何处理标题和其他排版元素，网格结构的设计方式是什么，以及使用哪个调色板。好的样式指南还应该描述文章、新闻条目和通知等重复元素的处理方法，决定它们应该如何实现和不该如何实现。更详细的样式指南甚至可以包含编码标准，比如使用的XHTML/CSS版本、选择的可访问性级别、浏览器支持细节和一般的编码最佳实践（见图2-6）。

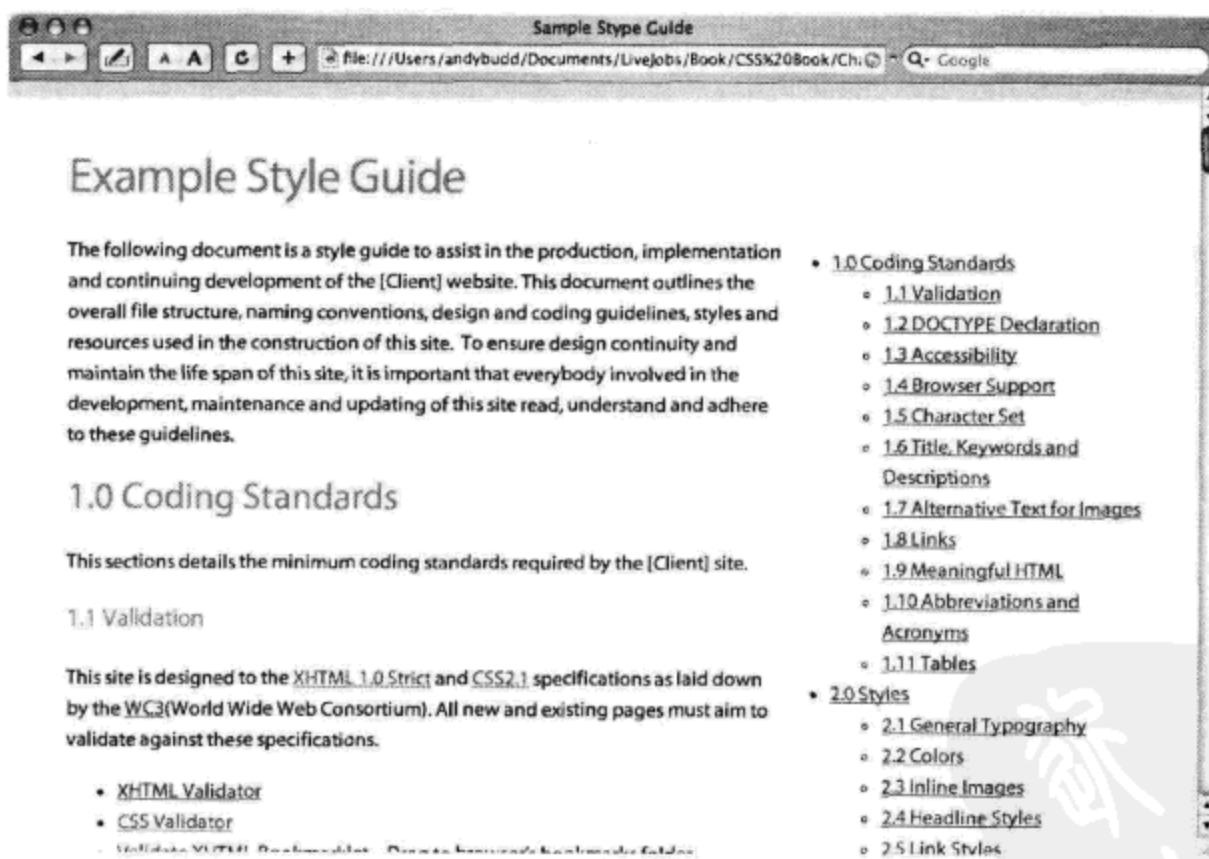


图2-6 样式指南示例

样式指南是帮助维护或实现站点的好方法。通过制定一些简单的原则，可以确保站点的开发以可控制的方式进行，同时防止样式随着时间的推移变得散乱。样式指南还非常有助于向新职员或承包商介绍他们不熟悉、可能很复杂的系统。

糟糕的是，及时更新样式指南需要花费大量精力，所以它们常常很快就与当前的站点不同步了。因此，我们更喜欢使用一种动态的样式指南形式，我们称之为“模式组合”（见图2-7）。



图2-7 WWF International站点的模式组合的一部分。实际页面的长度大约是这里显示的5倍，包含这个站点上可能出现的每种排版和布局组合

模式组合是一个页面或一系列页面，它们使用当前的样式表显示站点上可能出现的每种样式排列组合，从标题级别和文本样式直到特定的内容和布局类型。这些页面为后端和前端开发人员提供极有价值的资源，让他们可以方便地构建页面组合。因为它们使用当前的样式表，所以也可以用它们进行回归测试，检查对CSS的修改是否导致了问题。

## 2.5 小结

在本章中，我们回顾了常用的CSS 2.1选择器，学习了一些新的CSS 3选择器；详细了解了特殊性的作用以及如何使用层叠控制CSS规则的结构，帮助规则找到目标；还学习了如何通过合理的注释和样式表结构提高可维护性。

在下一章中，我们将学习CSS盒模型，讨论如何实现外边距叠加及其原因，以及浮动和定位是如何工作的。





## 第3章

# 可视化格式模型

你要掌握的3个最重要的CSS概念是浮动、定位和盒模型。这些概念控制在页面上安排和显示元素的方式，形成CSS的基本布局。如果你习惯于用表格控制布局，那么这些概念初看上去可能有点儿奇怪。实际上，大多数人只有在使用CSS开发站点一段时间之后，才能完全掌握盒模型的复杂性、绝对定位和相对定位之间的差异以及浮动和清理的实际工作方式。在切实掌握这些概念之后，使用CSS开发站点就会变得容易多了。

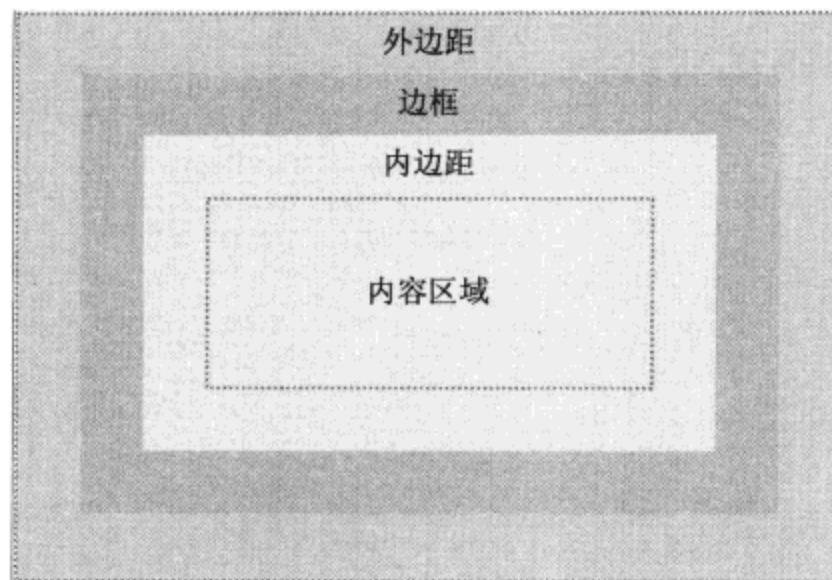
在本章中，你将学习以下内容。

- 盒模型的复杂性和特点。
- 如何以及为什么使用外边距叠加。
- 绝对定位和相对定位之间的差异。
- 浮动和清理是如何工作的。

### 3.1 盒模型概述

盒模型是CSS的基石之一，它指定元素如何显示以及（在某种程度上）如何相互交互。页面上的每个元素被看做一个矩形框，这个框由元素的内容、内边距、边框和外边距组成（见图3-1）。

内边距出现在内容区域的周围。如果在元素上添加背景，那么背景会应用于由内容和内边距组成的区域。因此，我们常常使用内边距在内容周围创建一个隔离带，使内容不会与背景混在一起。添加边框会在内边距的区域外边加一条线。这些线可以有多种样式，比如实线、虚线或点线。在边框外边的是外边距。外边距是透明的。一般使用它控制元素之间的间隔。



3

图3-1 盒模型的示意图

CSS 2.1 还包含 `outline` 属性。与 `border` 属性不同，轮廓绘制在元素框之上，所以它们不影响元素的大小或定位。因此，轮廓有助于修复 bug，因为它们不影响页面的布局。大多数现代浏览器（包括 IE 8）支持轮廓，但是 IE 7 和更低版本不支持轮廓。

内边距、边框和外边距都是可选的，默认值为零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 `margin` 或 `padding` 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

```
* {  
    margin: 0;  
    padding: 0;  
}
```

请记住，这种技术不区分元素，所以它对 `option` 等元素有不利影响。因此，使用全局 `reset` 把内边距和外边距显式地设置为零可能更安全。

在 CSS 中，`width` 和 `height` 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。假设框的每个边上有 10 像素的外边距和 5 像素的内边距，如果希望这个框达到 100 像素宽，就需要将内容的宽度设置为 70 像素（见图 3-2）：

```
#myBox {  
    margin: 10px;  
    padding: 5px;  
    width: 70px;  
}
```

内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。外边距还可以是负值，这可以用在多种技术中。

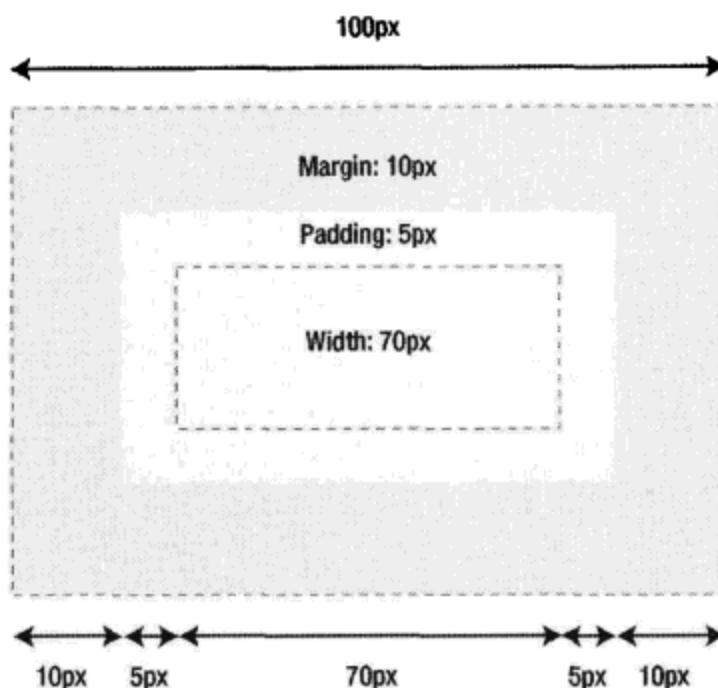


图3-2 正确的盒模型

### 3.1.1 IE 和盒模型

令人遗憾的是，IE的早期版本，包括IE 6，在混杂模式中使用自己的非标准盒模型。这些浏览器的width属性不是内容的宽度，而是内容、内边距和边框的宽度总和。这实际上很有道理，因为在现实世界中框具有固定的尺寸，而且内边距是放在框里面的。添加的内边距越多，给内容留下的空间就越少。尽管符合逻辑，但是这些IE版本不符合规范，这会造成严重的问题。例如，在前面的示例中，在IE 5.x中框的总宽度只有90像素。这是因为IE 5.x认为每个边上5像素的内边距是70像素的宽度的一部分，而不是在宽度之外附加的（见图3-3）。

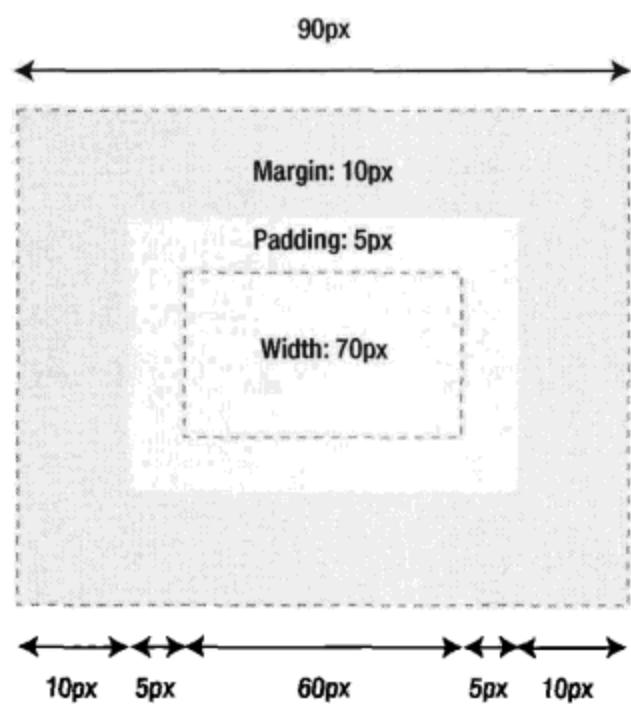


图3-3 IE专有的盒模型使元素比预期的小

CSS 3 的 `box-sizing` 属性可以定义要使用哪种盒模型，但是除了一些非常特殊的情况之外很少需要使用这个属性。

幸运的是，有几个方法可以解决这个问题，这些方法的细节可以在第9章中找到。但是，目前最好的解决方案是回避这个问题。也就是，不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素或子元素。

### 3.1.2 外边距叠加

3

外边距叠加是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。简单地说，当两个或更多垂直外边距相遇时，它们将形成一个外边距。这个外边距的高度等于两个发生叠加的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的底外边距与第二个元素的顶外边距发生叠加（见图3-4）。

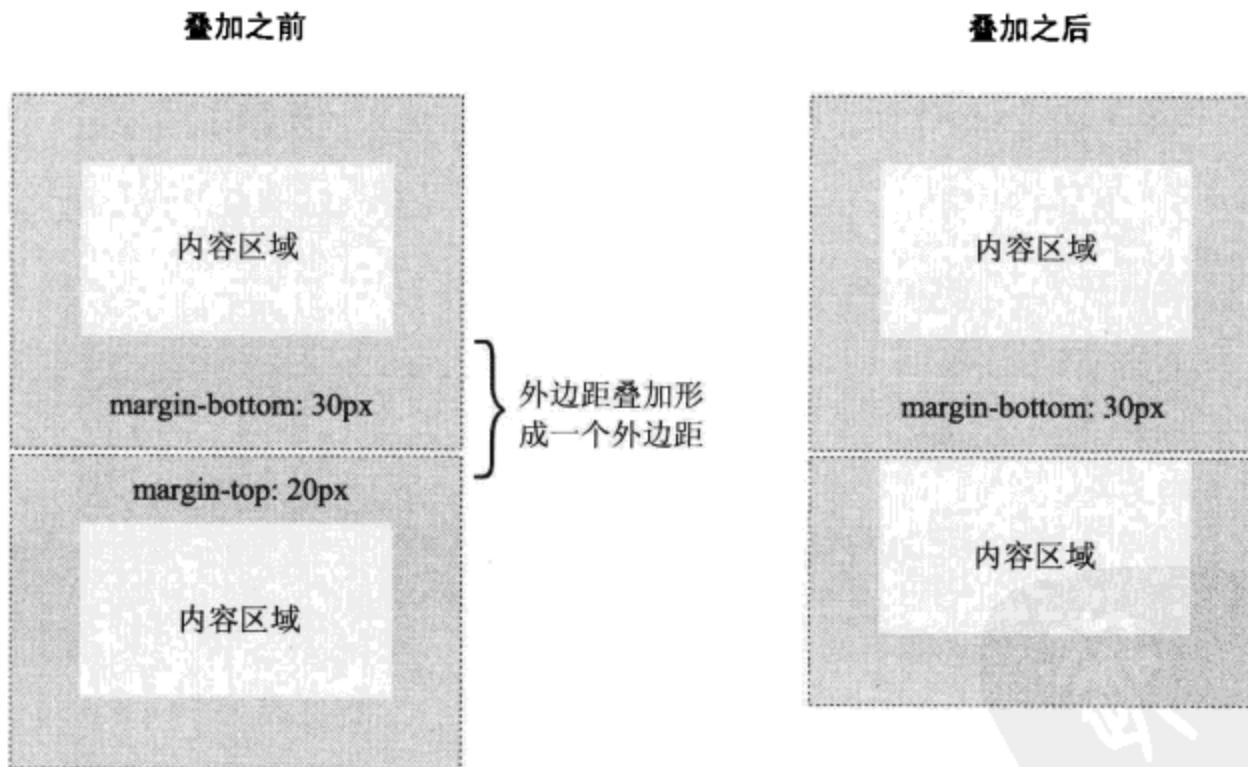


图3-4 元素的顶外边距与前面元素的底外边距发生叠加

当一个元素包含在另一个元素中时（假设没有内边距或边框将外边距分隔开），它们的顶和/或底外边距也会发生叠加（见图3-5）。

尽管初看上去有点儿奇怪，但是外边距甚至可以与本身发生叠加。假设有一个空元素，它有外边距，但是没有边框或内边距。在这种情况下，顶外边距与底外边距就碰到了一起，它们会发生叠加（见图3-6）。



图3-5 元素的顶外边距与父元素的顶外边距发生叠加

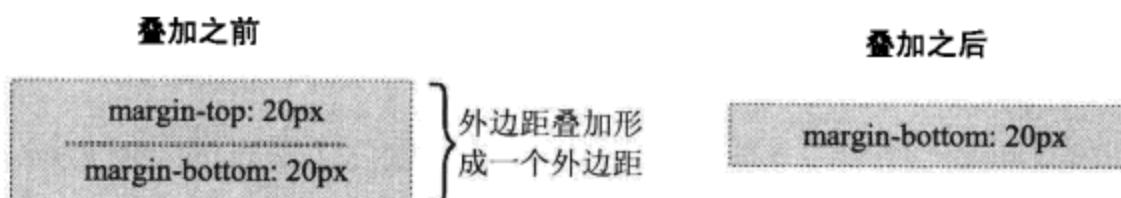


图3-6 元素的顶外边距与底外边距发生叠加

如果这个外边距碰到另一个元素的外边距，它还会发生叠加（见图3-7）。

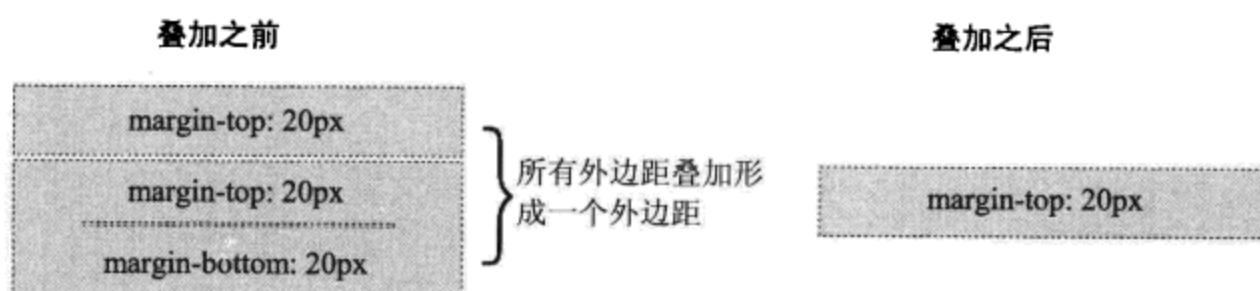


图3-7 空元素中已经叠加的外边距与另一个空元素的外边距发生叠加

这就是一系列空的段落元素占用的空间非常小的原因，因为它们的所有外边距都叠加到一起，形成一个小的外边距。

外边距叠加初看上去可能有点儿奇怪，但是它实际上有重要的意义。以由几个段落组成的典型文本页面为例（见图3-8）。第一个段落上面的空间等于段落的顶外边距。如果没有外边距叠加，后续所有段落之间的空间将是相邻顶外边距和底外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距叠加，段落之间的顶外边距和底外边距就叠加在一起，这样各处的距离就一致了。

只有普通文档流中块框的垂直外边距才会发生外边距叠加。行内框、浮动框或绝对定位框之间的外边距不会叠加。

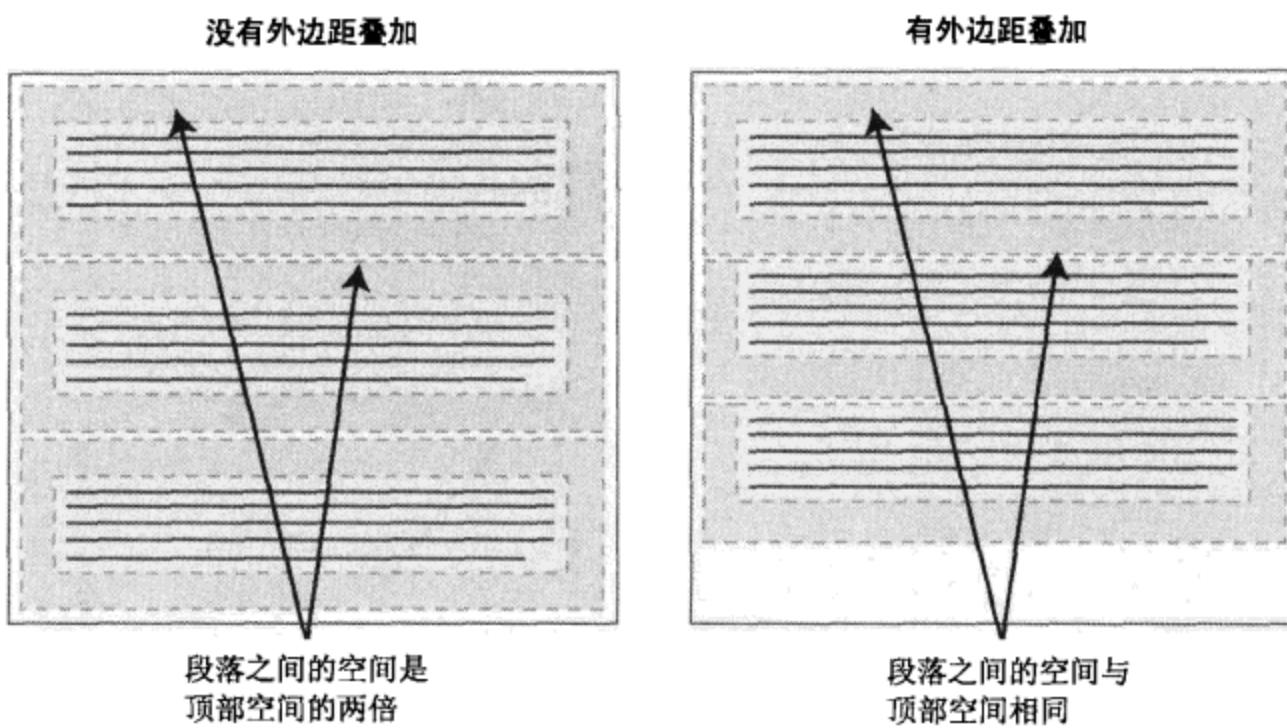


图3-8 外边距叠加在元素之间维持了一致的距离

3

## 3.2 定位概述

既然已经熟悉了盒模型，我们就来看看可视化格式模型和定位模型。理解这两个模型的细微差异是非常重要的，因为它们一起控制着如何在页面上布置每个元素。

### 3.2.1 可视化格式模型

p、h1或div等元素常常称为块级元素。这意味着这些元素显示为一块内容，即“块框”。与之相反，strong和span等元素称为行内元素，因为它们的内容显示在行中，即“行内框”。

可以使用display属性改变生成的框的类型。这意味着，通过将display属性设置为block，可以让行内元素（比如锚）表现得像块级元素一样。还可以通过将display属性设置为none，让生成的元素根本没有框。这样，这个框及其所有内容就不再显示，不占用文档中的空间。

CSS中有3种基本的定位机制：普通流、浮动和绝对定位。除非专门指定，否则所有框都在普通流中定位。顾名思义，普通流中元素框的位置由元素在HTML中的位置决定。

块级框从上到下一个接一个地垂直排列，框之间的垂直距离由框的垂直外边距计算出来。

行内框在一行中水平排列。可以使用水平内边距、边框和外边距调整它们的水平间距（见图3-9）。但是，垂直内边距、边框和外边距不影响行内框的高度。同样，在行内框上设置显式的高度或宽度也没有影响。由一行形成的水平框称为行框，行框的高度总是足以容纳它包含的所有行内框。但是，设置行高可以增加这个框的高度。因此，修改行内框尺寸的唯一方法是修改行高或者水平边框、内边距或外边距。

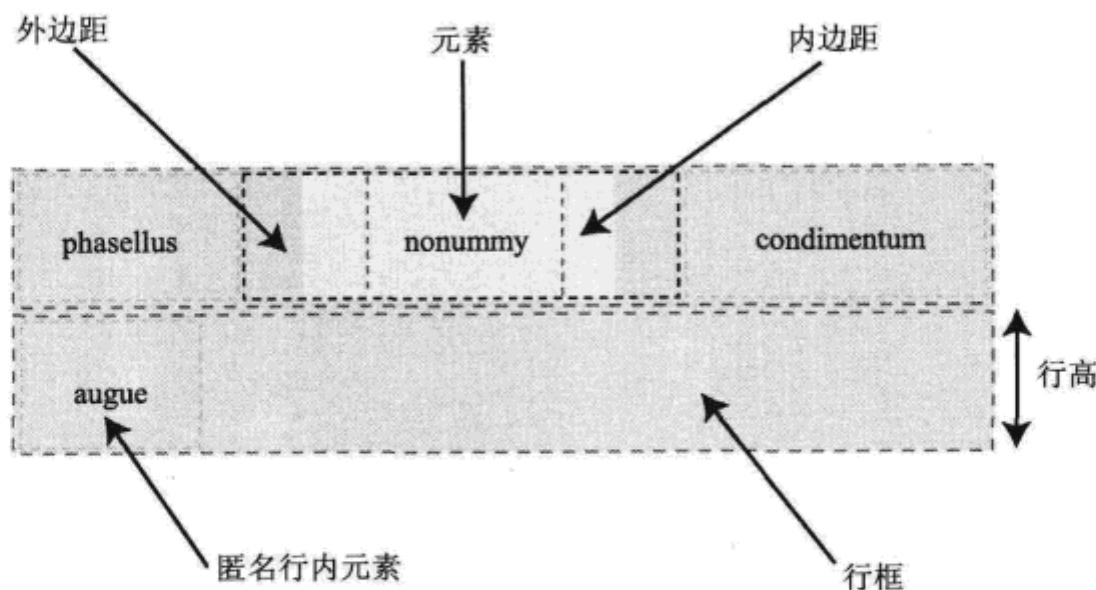


图3-9 行框中的行内元素

好在CSS 2.1允许把元素的display属性设置为`inline-block`。顾名思义，这个声明让元素像行内元素一样水平地依次排列。但是，框的内容仍然符合块级框的行为，例如能够显式地设置宽度、高度、垂直外边距和内边距。过去，浏览器对这个属性的支持很差，所以很少使用它。现在，Firefox 3.0及更高版本、IE 8以及Safari和Opera的最新版本支持`inline-block`，所以我认为以后几年会出现用`inline-block`创建的有意思的布局。

框可以按照HTML元素的嵌套方式包含其他框。大多数框由显式定义的元素形成。但是，在一种情况下，即使没有进行显式定义，也会创建块级元素。这种情况发生在将一些文本添加到一个块级元素（比如`div`）的开头时。即使没有把这些文本定义为块级元素，它也会被当成块级元素对待：

```
<div>
  some text
  <p>Some more text</p>
</div>
```

在这种情况下，这个框称为匿名块框，因为它不与专门定义的元素相关联。

块级元素内的文本行也会发生类似的情况。假设有一个包含3行文本的段落。每行文本形成一个匿名行框。无法直接对匿名块或行框应用样式，除非使用不常用的`:first-line`伪元素。但是，这有助于理解在屏幕上看到的所有东西都形成某种框。

### 3.2.2 相对定位

相对定位是一个非常容易掌握的概念。如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点移动。如果将`top`设置为20像素，那么框将出现在原位置顶部下面20像素的地方。如果将`left`设置为20像素，那么

会在元素左边创建20像素的空间，也就是将元素向右移动（见图3-10）。

```
#myBox {
    position: relative;
    left: 20px;
    top: 20px;
}
```

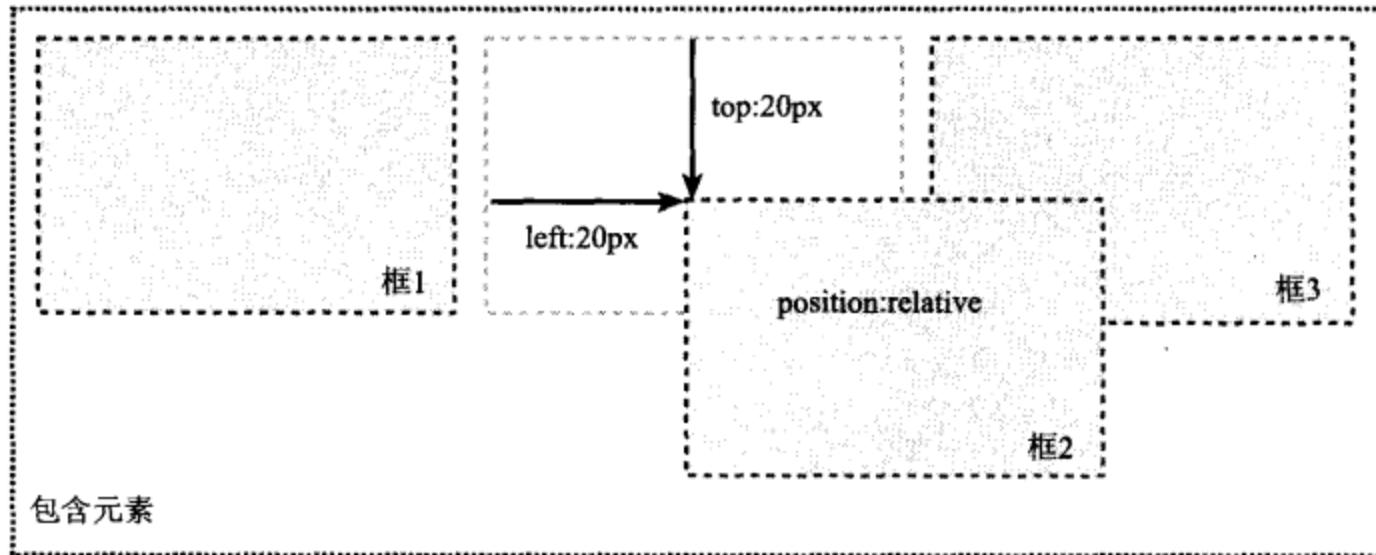


图3-10 对元素进行相对定位

在使用相对定位时，无论是否移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其他框。

### 3.2.3 绝对定位

相对定位实际上被看做普通流定位模型的一部分，因为元素的位置是相对于它在普通流中的位置的。与之相反，绝对定位使元素的位置与文档流无关，因此不占据空间。普通文档流中其他元素的布局就像绝对定位的元素不存在时一样（见图3-11）。

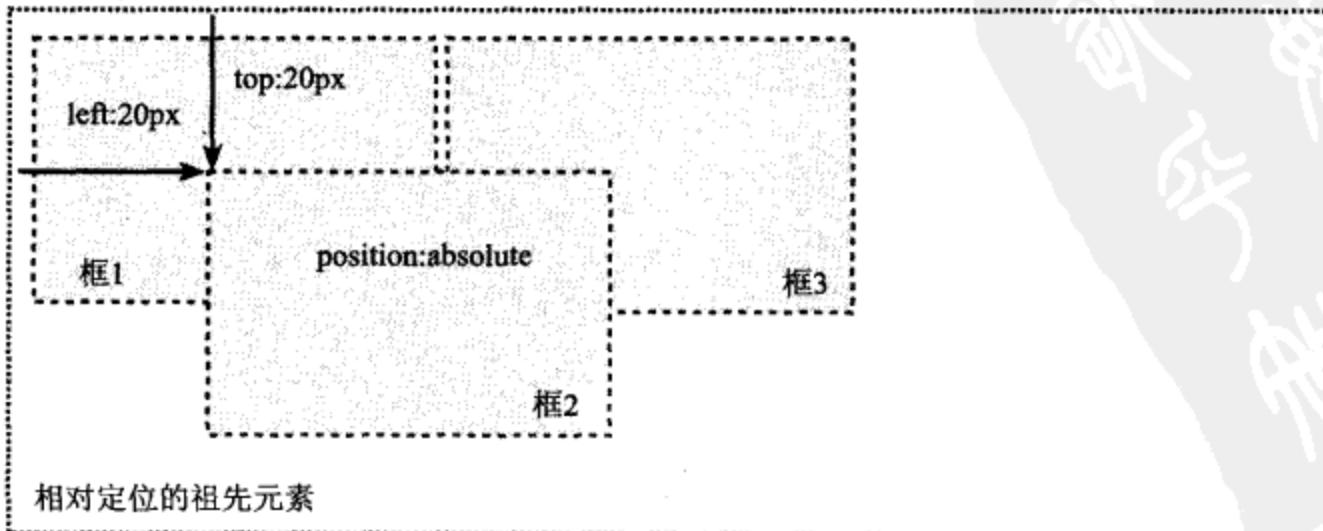


图3-11 对元素进行绝对定位

绝对定位的元素的位置是相对于距离它最近的那个已定位的祖先元素确定的。如果元素没有已定位的祖先元素，那么它的位置是相对于初始包含块的。根据用户代理的不同，初始包含块可能是画布或HTML元素。

与相对定位的框一样，绝对定位的框可以从它的包含块向上、下、左、右移动。这提供了很大的灵活性，你可以直接将元素定位在页面上的任何位置。

对于定位的主要问题是要记住每种定位的意义。相对定位是“相对于”元素在文档流中的初始位置，而绝对定位是“相对于”距离它最近的已定位祖先元素，如果不存在已定位的祖先元素，那么相对于初始包含块。

因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其他元素。可以通过设置 `z-index` 属性来控制这些框的叠放次序。`z-index` 值越高，框在栈中的位置就越高。

相对于最近的已定位祖先元素来定位绝对定位的元素，能够实现一些非常有意思的效果。例如，假设希望让一个文本段落对准一个大框的右下角，只需对包含框进行相对定位，然后相对于这个框对段落进行绝对定位：

```
#branding {  
    width: 70em;  
    height: 10em;  
    position: relative;  
}  
  
#branding .tel {  
    position: absolute;  
    right: 1em;  
    bottom: 1em;  
    text-align: right;  
}  
  
<div id="branding">  
    <p class="tel">Tel: 0845 838 6163</p>  
</div>
```

相对于已相对定位的祖先元素对框进行绝对定位，这在大多数现代浏览器中实现得很好。但是，在 Windows 上的 IE 5.5 和 IE 6 中有一个 bug。如果要相对于相对定位的框的右边或底部设置绝对定位的框的位置，那么需要确保相对定位的框已经设置了尺寸。如果没有，那么 IE 会错误地相对于画布定位这个框。在第 9 章中可以进一步了解这个 bug 和修复方法。简单的解决方案是为相对定位的框设置宽度和高度，从而避免这一问题。

在进行页面布局时，绝对定位是非常有用的，尤其是在使用相对定位的祖先元素的情况下。你完全可能只使用绝对定位就创建出整个设计。为此，这些元素需要具有固定尺寸，这样就能够

将它们定位在任何地方而不会有重叠的风险。

因为绝对定位的元素与文档流无关，所以它们不影响普通流中的框。如果扩大绝对定位的框（例如，通过增加字号），周围的框不会重新定位。因此，尺寸的任何改变都会导致绝对定位的框产生重叠，从而破坏精心调整过的布局。

### 固定定位

固定定位是绝对定位的一种。差异在于固定元素的包含块是视口（viewport）。这使我们能够创建总是出现在窗口中相同位置的浮动元素。老的snook.ca网站上就有这样的例子（见图3-12）。博客评论表单采用固定定位，这使它在页面滚动时一直出现在屏幕上的固定位置。这有助于改进易用性，用户不必为了发表评论而一直滚动到页面底部。

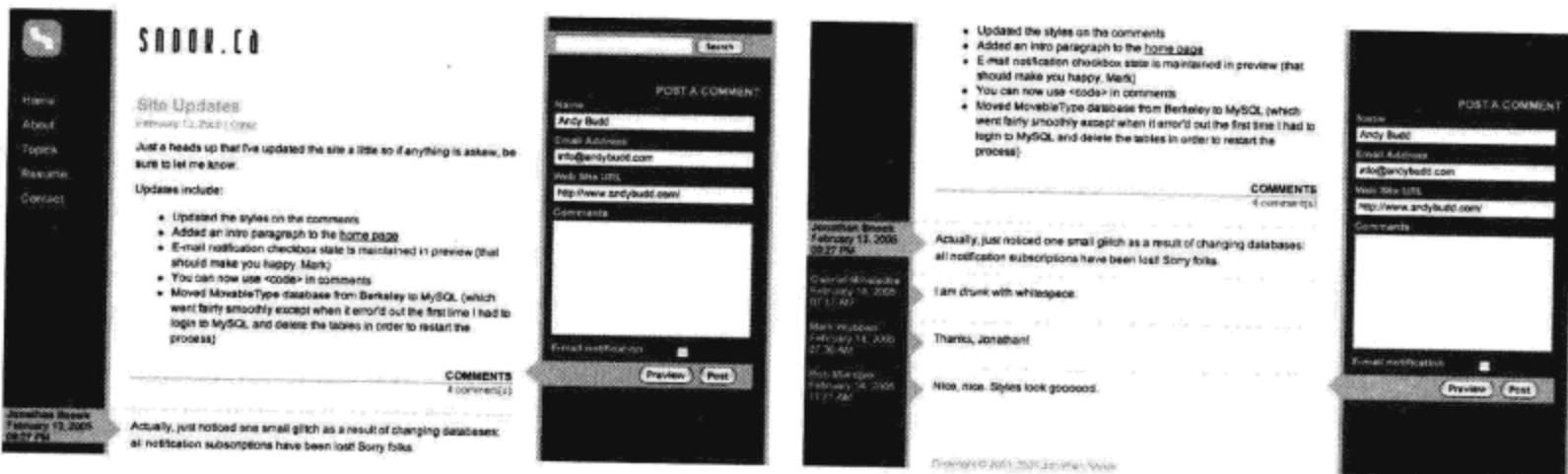


图3-12 在老的snook.ca网站上，屏幕右边的评论区域采用固定定位，因此一直出现在视口中的固定位置

不过，IE 6和更低版本不支持固定定位。IE 7部分支持这个属性，但是实现中有许多bug。为了解决这个问题，Jonathan Snook使用JavaScript在IE中重现了这个效果。

### 3.2.4 浮动

最后一种定位模型是浮动模型。浮动的框可以左右移动，直到它的外边缘碰到包含框或另一个浮动框的边缘。因为浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

如图3-13所示，当把框1向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘。

在图3-14中，当把框1向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框2，使框2从视图中消失。如果把所有3个框都向左浮动，那么框1向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。

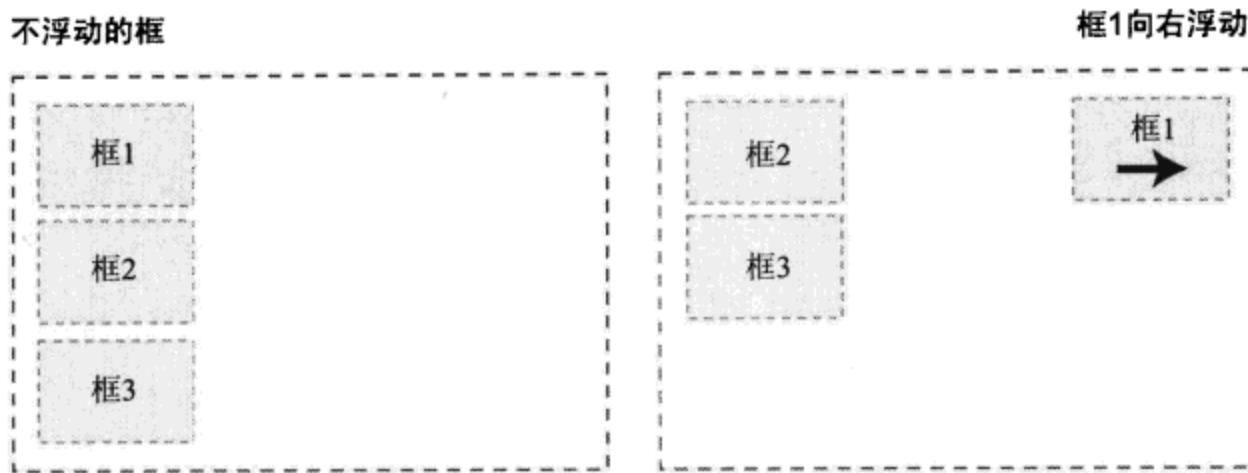


图3-13 向右浮动的元素

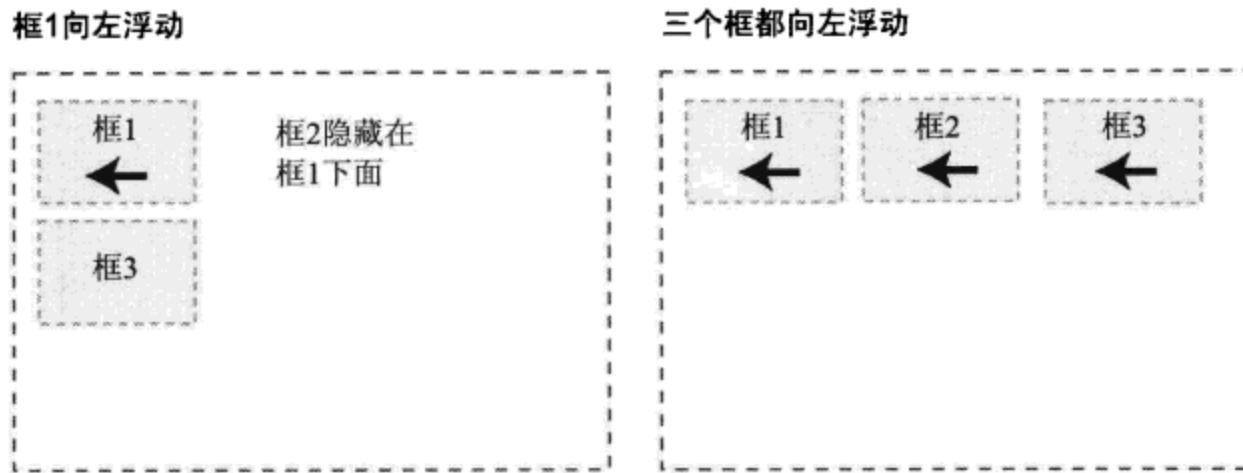


图3-14 向左浮动的元素

如果包含块太窄，无法容纳水平排列的3个浮动元素，那么其他浮动块向下移动，直到有足够的空间的地方（见图3-15）。如果浮动元素的高度不同，那么当它们向下移动时可能会被其他浮动元素“卡住”。

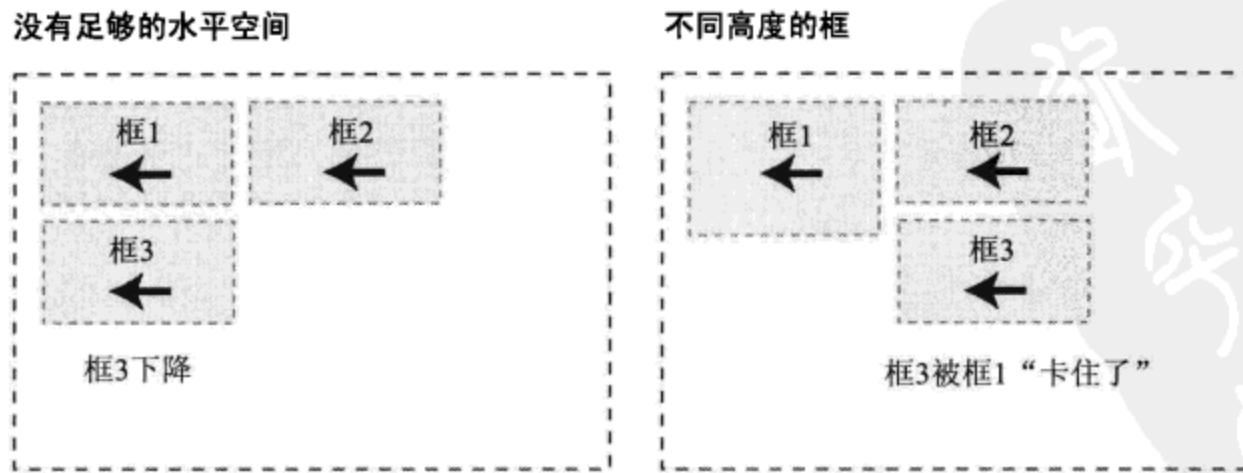


图3-15 如果没有足够的水平空间，浮动元素将向下移动，直到有足够的空间的地方

## 行框和清理

前一节指出，浮动会让元素脱离文档流，不再影响不浮动的元素。实际上，并不完全如此。如果浮动的元素后面有一个文档流中的元素，那么这个元素的框会表现得像浮动根本不存在一样。但是，框的文本内容会受到浮动元素的影响，会移动以留出空间。用技术术语来说，浮动元素旁边的行框被缩短，从而给浮动元素留出空间，因此行框围绕浮动框。实际上，创建浮动框使文本可以围绕图像（见图3-16）。

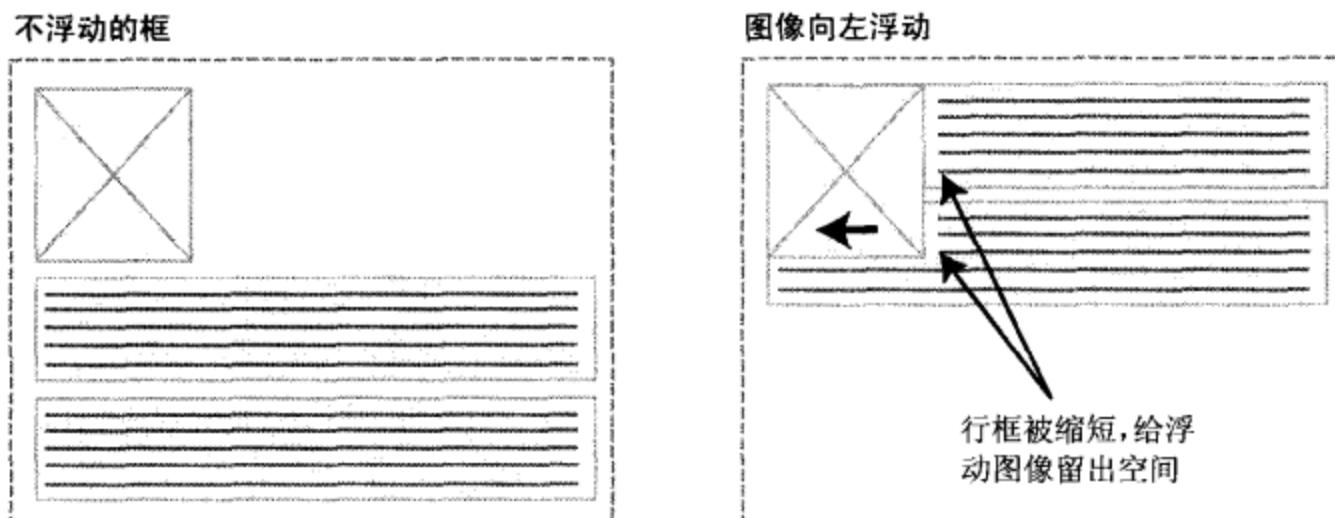


图3-16 浮动框旁边的行框被缩短

要想阻止行框围绕在浮动框的外边，需要对包含这些行框的元素应用clear属性。clear属性的值可以是left、right、both或none，它表示框的哪些边不应该挨着浮动框。我以前总是认为clear属性会自动地抵消前面的浮动。但是，实际情况有意思得多。在清理元素时，浏览器在元素顶上添加足够的外边距，使元素的顶边缘垂直下降到浮动框下面（见图3-17）。

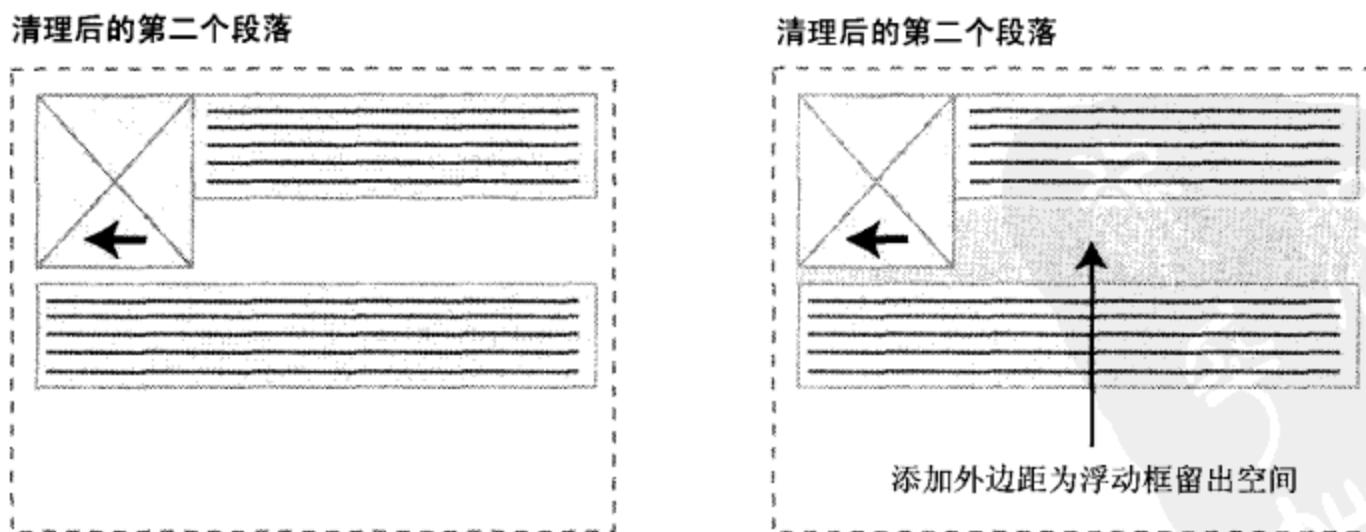


图3-17 清理元素的顶外边距，为前一个浮动框留出足够的垂直空间

浮动元素脱离了文档流，不影响周围的元素。但是，对元素进行清理实际上为前面的浮动元素留出了垂直空间。

这是一个有用的布局工具，它让周围的元素为浮动元素留出空间。这解决了前面你看到的绝对定位的问题——垂直高度的改变不影响周围的元素，从而破坏了设计。

我们来更详细地看看浮动和清理。假设有一个图片，你希望让它浮动到一个文本块的左边。你想将这个图片和文本包含在另一个具有背景颜色和边框的元素中。你可能会编写下面这样的代码：

```
.news {
    background-color: gray;
    border: solid 1px black;
}

.news img {
    float: left;
}

.news p {
    float: right;
}

<div class="news">
    
    <p>Some text</p>
</div>
```

但是，因为浮动元素脱离了文档流，所以包围图片和文本的div不占据空间。如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用clear（见图3-18）。可惜这个示例中没有现有的元素可以清理，所以需要在最后一个段落下面添加一个空元素并且清理它。

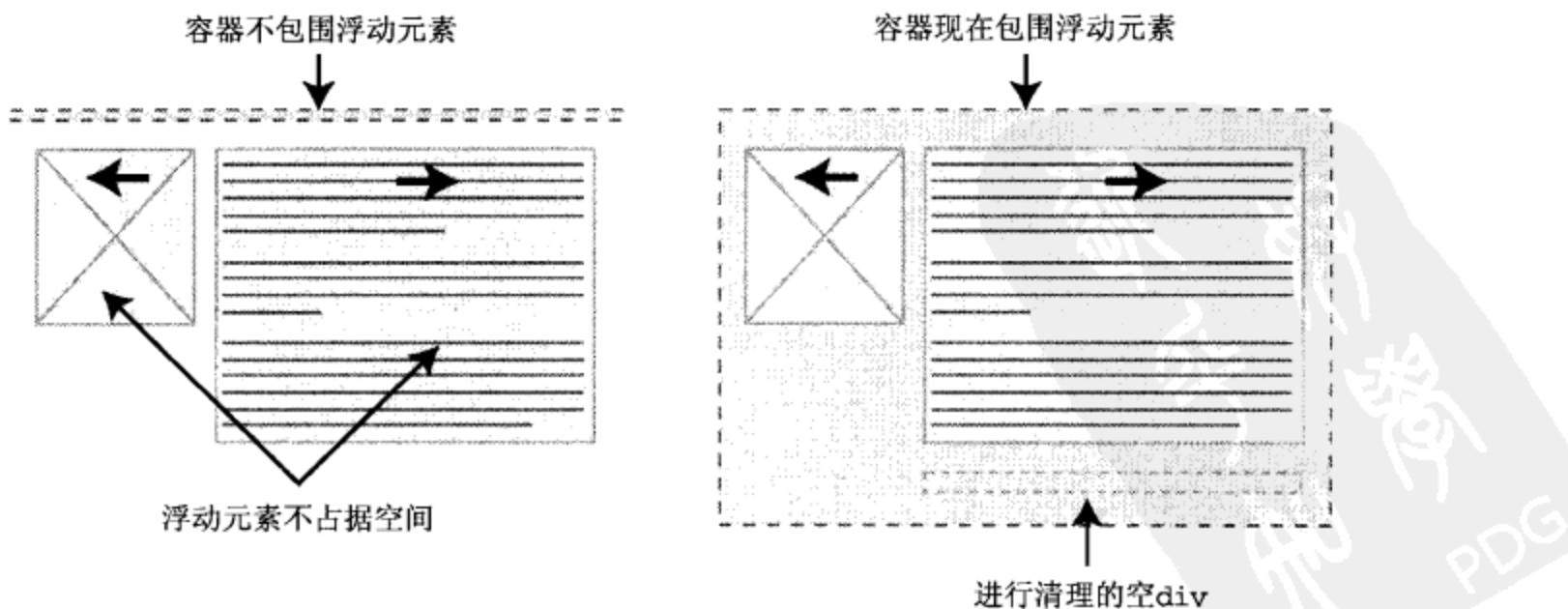


图3-18 因为浮动元素不占据空间，所以容器元素不包围它们。添加一个进行清理的空元素可以迫使容器元素包围浮动元素

```
.news {  
    background-color: gray;  
    border: solid 1px black;  
}  
  
.news img {  
    float: left;  
}  
  
.news p {  
    float: right;  
}  
  
.clear {  
    clear: both;  
}  


  
    <p>Some text</p>  
    <br class="clear" />  
</div>


```

3

这会实现我们希望的效果，但是要添加不必要的代码。常常有现成的元素可以应用clear，但是有时候不得不忍受巨大痛苦布局添加无意义的标记。

还可以不对浮动的文本和图像进行清理，而是选择浮动容器div:

```
.news {  
    background-color: gray;  
    border: solid 1px black;  
    float: left;  
}  
  
.news img {  
    float: left;  
}  
  
.news p {  
    float: right;  
}  


  
    <p>Some text</p>  
</div>


```

这也会产生我们想要的结果。但是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择浮动布局中的几乎所有东西，然后使用合适的元素（常常是站点的页脚）对这

些浮动元素进行清理。这有助于减少或消除不必要的标记。但是，浮动会变得复杂，而且一些老式浏览器在处理有许多浮动元素的布局时有困难。因此，许多人喜欢添加少量的额外标记。

`overflow`属性定义在包含的内容对于指定的尺寸太大的情况下元素应该怎么样。在默认情况下，内容会溢出到框外，进入相邻的空间。应用值为`hidden`或`auto`的`overflow`属性有一个有用的副作用，这会自动地清理包含的任何浮动元素。因此这是一种有用的元素清理方法，不需要添加额外的标记。这个方法并不适合所有情况，因为设置框的`overflow`属性会影响它的表现。更具体地说，这种方法在某些情况下会产生滚动条或截断内容。

然后，一些人使用CSS生成的内容或JavaScript对浮动元素进行清理。这两种方法的基本概念是相同的，并不直接向标记中添加进行清理的元素，而是将它动态地添加到页面中。对于这两种方法，需要指定进行清理的元素应该出现在哪里，而且常常要添加一个类名：

```
<div class="news clear">
  
  <p>Some text</p>
</div>
```

在使用CSS方法时，结合使用`:after`伪类和内容声明在指定的现有内容的末尾添加新的内容。在这个示例中，我添加了一个点，因为它是个非常小的不引人注意的字符。因为不希望新内容占据垂直空间或者在页面上显示，所以需要将`height`设置为0，将`visibility`设置为`hidden`。因为被清理的元素在它们的顶外边距上添加了空间，所以生成的内容需要将它的`display`属性设置为`block`。这样设置之后，就可以对生成的内容进行清理：

```
.clear:after {
  content: ".";
  height: 0;
  visibility: hidden;
  display: block;
  clear: both;
}
```

这个方法在大多数现代浏览器中是有效的，但是在IE 6和更低版本中不起作用。有各种解决方案，其中许多记录在[www.positioniseverything.net/easyclearing.html](http://www.positioniseverything.net/easyclearing.html)中。最常用的解决方案要用到Holly hack（见第8章），从而迫使IE 5和IE 6应用“布局”（见第9章）和错误地清理浮动元素。

```
.clear {
  display: inline-block;
}
/* Holly Hack Targets IE Win only */
* html .clear {height: 1%;}
.clear {display: block;}
/* End Holly Hack */
```

但是，由于其复杂性，这个方法可能不适合所有人采用，这里提到它主要是出于历史原因。

对JavaScript方法的解释不在本书范围内，但是需要简要地提及一下。与前面的方法不同，JavaScript方法在所有主流浏览器上都是有效的（在打开脚本功能的情况下）。但是，如果使用这个方法，就需要确保在关闭脚本功能的情况下内容仍然是可读的。

### 3.3 小结

在本章中，我们学习了盒模型以及内边距、外边距、宽度和高度如何影响框的尺寸；还学习了外边距的叠加以及这如何影响布局；探讨了CSS中的3种格式化模型：普通流、绝对定位和浮动；此外，还了解了行内框和块框之间的差异，如何在相对定位的祖先元素中进行绝对定位，并讨论了清理的实际实现方式。

具备了这些知识之后，我们就可以开始运用它们了。在本书的下一部分中，我们将讲解许多CSS核心概念，你将看到如何使用这些概念创建各种有用且实用的技术。打开你喜欢的文本编辑器，我们要开始编程了。





## 第4章

# 背景图像效果

既然你已经掌握了理论，我们就开始把理论运用到实践中。当今的Web是一种视觉效果非常丰富的媒体。简便的图像标签使网页设计人员能够将毫无趣味的文档变成图形丰富的浏览体验。图形设计人员很快就掌握了image标签（原本专门用于向网站中添加可视内容的方式），将它作为对页面进行视觉修饰的方式。实际上，如果没有发明image标签，那么可能就没有网页设计师这种职业。

然而，对image标签的滥用导致纯修饰性的图像把页面弄乱了。好在CSS使我们能够在页面上显示图像，而不需要让图像成为标记的一部分。实现方法是将图像作为背景添加到现有的元素中。本章将通过一系列实际示例讲解如何使用背景图像创建各种有意思且有用的技术。

本章将介绍以下内容：

- 固定宽度和可变宽度的圆角框
- 滑动门技术
- 多个背景图像和border-radius属性
- CSS投影
- 不透明度和RGBa
- 让PNG适用于IE的老版本
- 视差滚动
- 图像替换

## 4.1 背景图像基础

应用背景图像是很容易的。如果希望网站有一个好看的背景，那么只需将图像作为背景应用

于主体元素：

```
body {  
    background-image:url(/img/pattern.gif);  
}
```

默认情况下，浏览器水平和垂直地重复显示背景图像，让图像平铺在整个页面上。可以选择背景图像是垂直平铺、水平平铺，还是根本不平铺。

目前渐变非常时髦，你可能希望在页面上应用垂直渐变。为此，需要创建一个很高但很窄的渐变图像，然后将这个图像应用于页面的主体并让它水平平铺：

```
body {  
    background-image: url(/img/gradient.gif);  
    background-repeat: repeat-x;  
}
```

因为这个渐变图像的高度是固定的，所以如果页面内容的长度超过了图像的高度，那么渐变就会突然终止。可以创建一个非常长的图像，逐渐变化到一个固定的颜色。但是，很难预测页面会有多长。实际上，只需再添加一个背景颜色。背景图像总是出现在背景颜色的上面，所以当图像结束时，颜色就会显示出来了。如果选择的背景颜色与渐变底部的颜色相同，那么图像和背景颜色之间的转换就看不出来了。

```
body {  
    background-image: url(/img/gradient.gif);  
    background-repeat: repeat-x;  
    background-color: #ccc;  
}
```

平铺图像在某些情况下很有用。但是，在大多数情况下，需要在页面上添加不是平铺的图像。例如，假设希望在网页的开头显示一个大的品牌图像，那么只需将图像直接添加到页面上，在许多情况下这样做就够了。但是，如果图像不包含信息，是纯表现性的，那么可能希望将图像从其余内容中分离出来。实现的方法是在HTML中为这个图像创建一个“钩子”，然后使用CSS应用这个图像。在下面的示例中，我在标记中添加一个空的div并且给它设置ID branding。然后可以将这个div的尺寸设置为与品牌图像相同，作为背景应用并指定不重复。

```
#branding {  
    width: 700px;  
    height: 200px;  
    background-image:url(/img/branding.gif)  
    background-repeat: no-repeat;  
}
```

还可以设置背景图像的位置。假设要在站点的每个标题上添加一个项目符号，如图4-1所示。可以编写下面这样的代码：

```

h1 {
    padding-left: 30px;
    background-image: url(/img/bullet.gif);
    background-repeat: no-repeat;
    background-position: left center;
}

```



图4-1 使用背景图像创建项目符号

最后两个关键字指出图像的位置。在这个示例中，图像定位在元素的左边并且垂直居中。除了使用关键字之外，还可以使用像素或百分数等单位设置背景图像的位置。

如果使用像素设置背景位置，那么图像左上角到元素左上角的距离为指定的像素数。所以，如果指定垂直和水平位置都是20像素，那么图像左上角将出现在元素左上角下面20像素、左边20像素的地方。但是，使用百分数进行背景定位的工作方式不太一样。百分数定位并不对背景图像的左上角进行定位，而是使用图像上的一个对应点。所以，如果指定垂直和水平位置都是20%，那么实际上是在将图像上距离左上角20%的点定位到父元素上距离左上角20%的位置（见图4-2）。

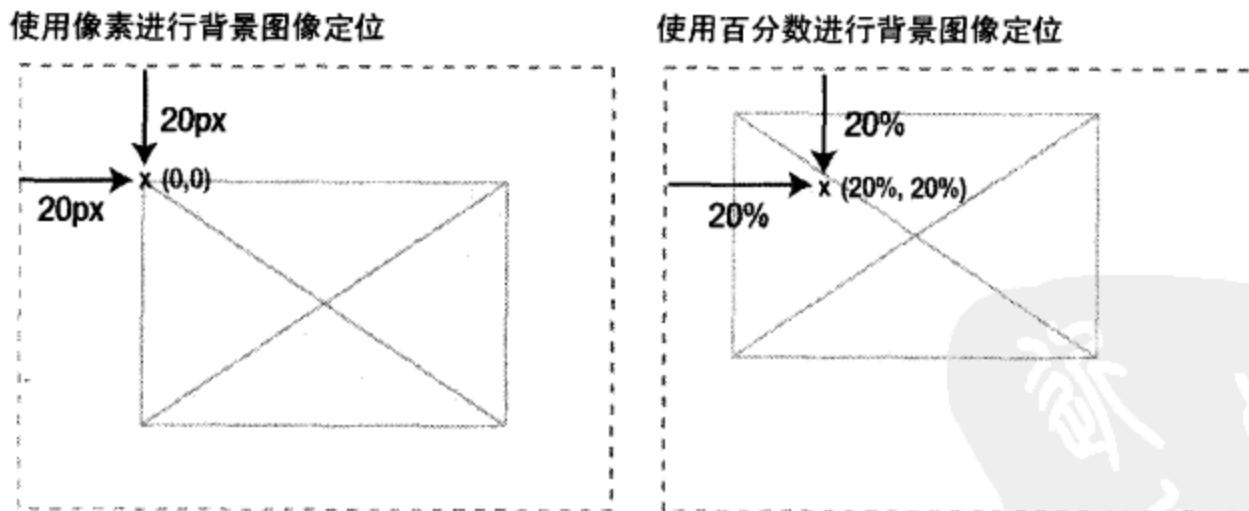


图4-2 在使用像素进行背景图像定位时，使用图像的左上角。在使用百分数进行背景图像定位时，使用图像上的对应位置

如果希望使用百分数而不是关键字实现前面的项目符号示例，那么要将垂直位置设置为50%，这会使项目符号图像垂直居中：

```

h1 {
    padding-left: 30px;
    background-image: url(/img/bullet.gif);
}

```

```

background-repeat: no-repeat;
background-position: 0 50%;
}

```

规范指出，不要将像素或百分数等单位与关键字混合使用。这似乎是一个没有意义的规则，而且许多现代浏览器故意忽略了这个规则。但是，混合使用单位和关键字在某些浏览器上会导致错误，而且很可能使CSS失效。因此，最好不要混合使用单位和关键字。

为了简便，CSS还提供了background属性的简写版本。可以通过它同时设置所有属性，不需要分别设置。

```

h1 {
  background: #ccc url(/img/bullet.gif) no-repeat left center;
}

```

尽管背景图像是一个容易掌握的概念，但是它们构成了许多高级CSS技术的基础。

## 4.2 圆角框

对基于CSS的设计最初的批评意见之一是CSS太死板了，只能建立方框。为了解决这个问题，人们开始创建具有曲线的设计。圆角框很快成为最时髦的CSS技术之一。创建圆角框有好几种方法。每种方法各有优缺点，对这些方法的选择主要取决于实际情况。

### 4.2.1 固定宽度的圆角框

最容易创建的是固定宽度的圆角框。它们只需要两个图像：一个图像用于框的顶部，另一个用于底部。例如，假设要创建图4-3这样的框样式。

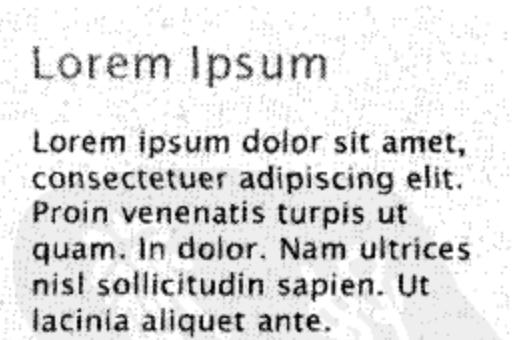
这个框的标记如下：

```

<div class="box">
  <h2>Headline</h2>
  <p>Content</p>
</div>

```

需要用图形软件创建两个图4-4这样的图像：一个图像用于框的顶部，另一个用于底部。这个示例以及本书中其他所有示例的代码和图像可以从[www.cssmastery.com](http://www.cssmastery.com)或[www.friendsofed.com](http://www.friendsofed.com)下载。



The image shows a browser window displaying a div element with a class of "box". Inside the div is an h2 tag with the text "Headline" and a p tag with the text "Content". Above the div, there is a decorative top border consisting of two curved images: "top.gif" at the top-left and "bottom.gif" at the top-right, which together create a rounded effect. The background of the page is white.

**Lorem Ipsum**

*Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin venenatis turpis ut quam. In dolor. Nam ultrices nisl sollicitudin sapien. Ut lacinia aliquet ante.*

图4-3 简单的圆角框样式

top.gif                    bottom.gif

图4-4 顶部和底部曲线图像

然后，将顶部图像应用于标题元素，将底部图像应用于div框的底部。因为这个框样式是单色的，所以可以在div框上添加背景颜色，从而形成框的主体。

```
.box {  
    width: 418px;  
    background: #effce7 url(/img/bottom.gif) no-repeat left bottom;  
}  
  
.box h2 {  
    background: url(/img/top.gif) no-repeat left top;  
}
```

我们不希望内容碰到框的边界，所以还需要在div中的元素上添加一些内边距：

```
.box {  
    width: 418px;  
    background: #effce7 url(/img/bottom.gif) no-repeat left bottom;  
    padding-bottom: 1px;  
}  
  
.box h2 {  
    background: url(/img/top.gif) no-repeat left top;  
    margin-top: 0;  
    padding: 20px 20px 0 20px;  
}  
  
.box p {  
    padding: 0 20px;  
}
```

这个方法对于单色而且没有边框的简单框是有效的。但是，如果希望创建像图4-5这样更生动的样式，该怎么办？

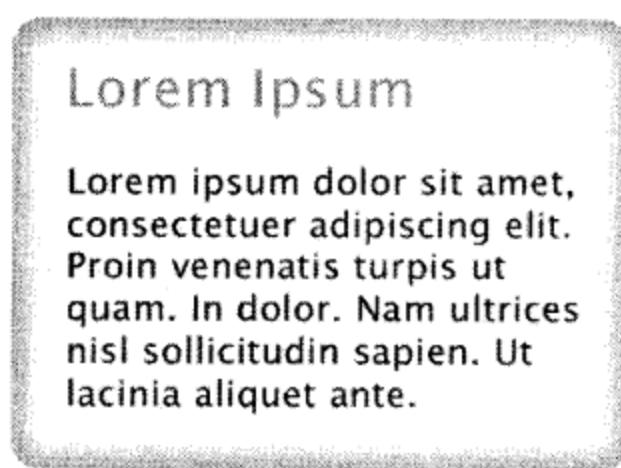


图4-5 样式更特殊的圆角框

实际上，可以使用相同的方式，但是这一次不在框上设置背景颜色，而是设置一个重复显示

的背景图像。还需要将底部曲线图像应用到另一个元素上。在这个示例中，我使用框中的最后一个段落元素：

```
.box {
    width: 424px;
    background: url(/img/tile2.gif) repeat-y;
}

.box h2 {
    background: url(/img/top2.gif) no-repeat left top;
    padding-top: 20px;
}

.box .last {
    background: url(/img/bottom2.gif) no-repeat left bottom;
    padding-bottom: 20px;
}

.box h2, .box p {
    padding-left: 20px;
    padding-right: 20px;
}

<div class="box">
    <h2>Headline</h2>
    <p class="last">Content</p>
</div>
```

图4-6所示为生成的框。因为没有给这个框设置高度，所以它会随着文本尺寸的增加进行垂直扩展。

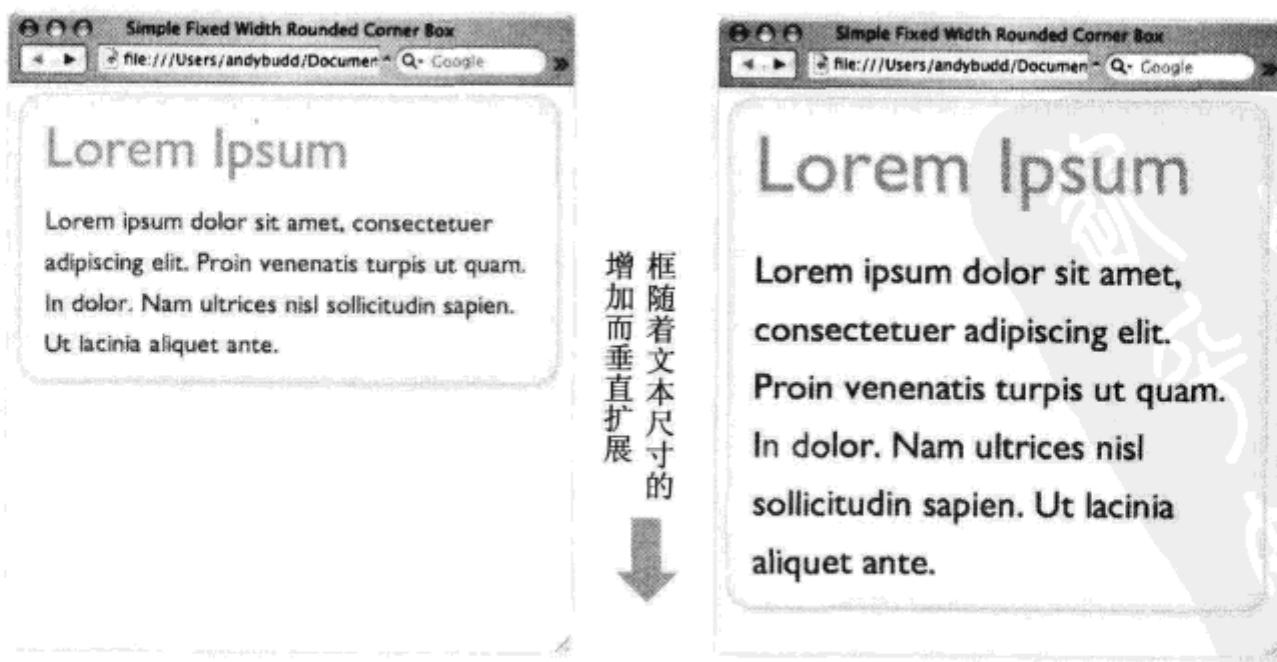


图4-6 应用了样式的固定宽度框。框的高度会随着文本尺寸的增加而扩展

## 灵活的圆角框

如果加大文本字号，前面的示例都会垂直扩展。但是，它们不会水平扩展，因为框的宽度必须与顶部和底部图像的宽度一致。如果希望创建灵活的框，那么需要采用略有不同的方法。不要用一个图像组成顶部和底部曲线，而应用两个相互重叠的图像（见图4-7）。

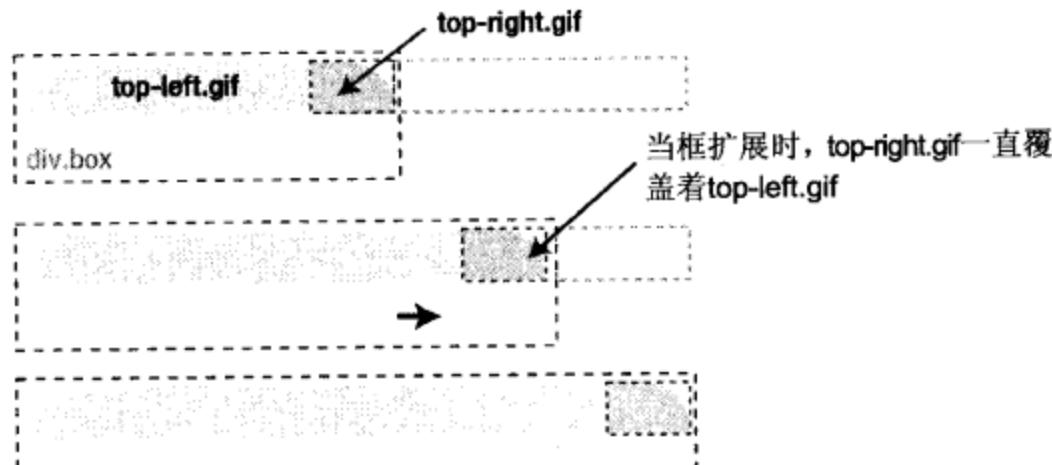


图4-7 如何扩展顶部图像来形成灵活的圆角框

随着框尺寸的增加，大图像有更多的部分显露出来，这样就实现了框扩展的效果。这个方法有时候被称为滑动门技术（sliding doors technique），因为一个图像在另一个图像上滑动，将它的一部分隐藏了起来。这个方法需要更多的图像，所以必须在标记中添加两个额外的无语义元素。

```
<div class="box">
  <div class="box-outer">
    <div class="box-inner">
      <h2>Headline</h2>
      <p>Content</p>
    </div>
  </div>
</div>
```

这个方法需要4个图像：两个顶部图像组成顶部曲线，两个底部图像组成底部曲线和框的主体（见图4-8）。因此，底部图像的高度必须与框的最大高度相同。分别将这些图像命名为 `top-left.gif`、`top-right.gif`、`bottom-left.gif` 和 `bottom-right.gif`。

首先，将 `bottom-left.gif` 应用于主框 `div`，将 `bottom-right.gif` 应用于外边的 `div`。接下来，将 `top-left.gif` 应用于内部的 `div`，将 `top-right.gif` 应用于标题。最后，添加一些内边距以便在内容周围形成一点儿空白。

```
.box {
  width: 20em;
  background: #effce7 url(/img/bottom-left.gif) no-repeat left bottom;
}

.box-outer {
```

```

background: url(/img/bottom-right.gif) no-repeat right bottom;
padding-bottom: 1em;
}

.box-inner {
background: url(/img/top-left.gif) no-repeat left top;
}

.box h2 {
background: url(/img/top-right.gif) no-repeat right top;
padding-top: 1em;
}

.box h2, .box p {
padding-left: 1em;
padding-right: 1em;
}

```

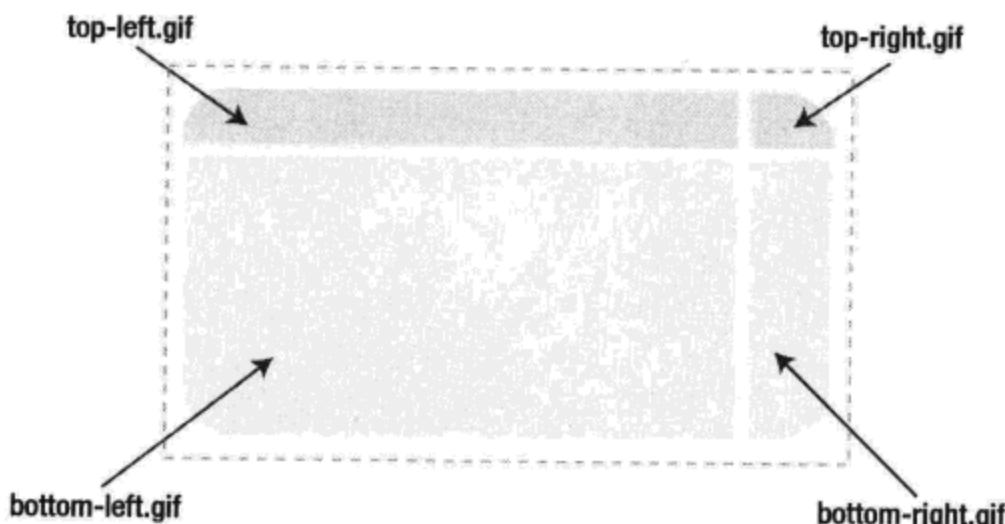


图4-8 创建灵活的圆角框所需的图像

在这个示例中，我以em为单位设置框的宽度，所以在浏览器中增加文本尺寸时框会伸展（见图4-9）。当然，可以用百分数设置宽度，这使框根据浏览器窗口的尺寸进行扩展或收缩。这是弹性和流式布局背后的主要原则之一，本书后面会进一步讨论这些原则。

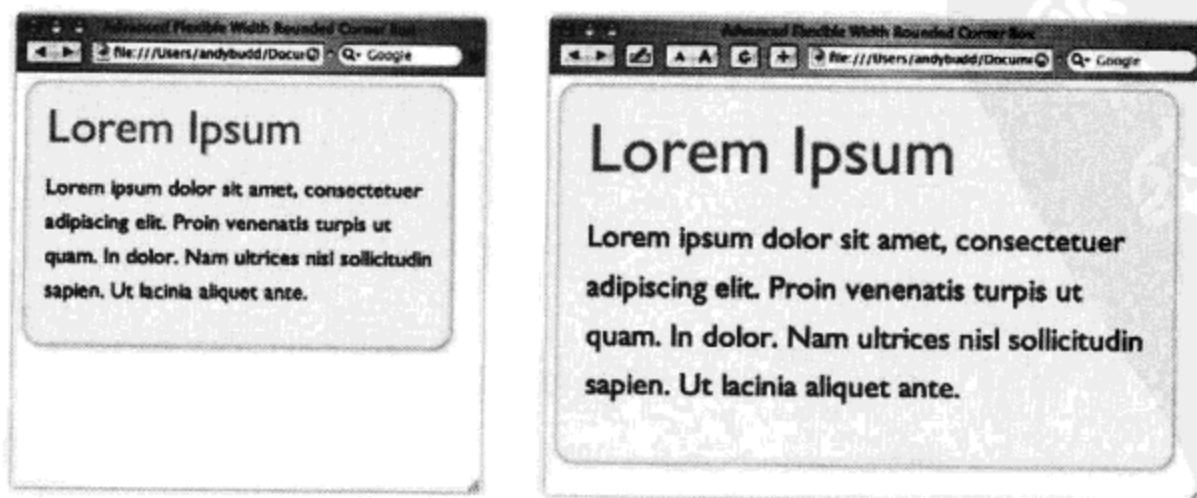


图4-9 灵活的圆角框会随着文本尺寸的增加进行水平和垂直扩展

添加两个额外的无语义元素是不理想的。如果只有很少的几个框，那么这是可以容忍的。但是，如果用到圆角框的地方很多，那么可以使用 JavaScript（和 DOM）添加额外元素。关于这个主题的更多细节，请参考 [www.456bereastreet.com/archive/200505/transparent\\_custom\\_corners\\_and\\_borders](http://www.456bereastreet.com/archive/200505/transparent_custom_corners_and_borders) 上 456 Berea Street 的 Roger Johansson 所写的很棒的文章。

#### 4.2.2 山顶角

山顶角（mountaintop corner）是一个简单但非常灵活的概念，是由[www.simplebits.com](http://www.simplebits.com)的Dan Cederholm首创的，他是畅销图书《Web标准实战》（人民邮电出版社2008年出版）的作者。若要创建一系列具有不同颜色的圆角框，如果使用前面的方法，就必须为每种颜色方案创建不同的角图像。如果只有几个方案，那么这个方法也可以，但是，如果想让用户创建自己的颜色方案，那么该怎么办？可能必须在服务器上动态地创建角图像，这是非常复杂的。

幸好还有另一个办法，不用创建有颜色的角图像，而是创建曲线形的位图角蒙板（见图4-10）。蒙板区域盖住你正使用的背景颜色，而角区域实际上是透明的。当放在有颜色的框上时，它们形成曲线形框的效果（见图4-11）。

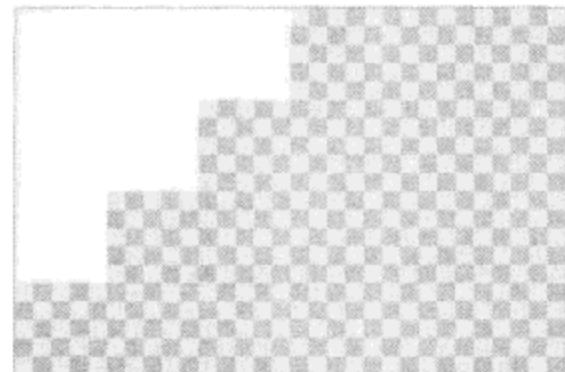


图4-10 位图角蒙板。白色的蒙板将覆盖背景颜色，产生简单的曲线效果

Lorem Ipsum

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit.  
Proin venenatis turpis ut  
quam. In dolor. Nam ultrices  
nisl sollicitudin sapien. Ut  
lacinia aliquet ante.

图4-11 山顶角框

因为这些角蒙板是位图，所以对于小曲线效果最好。如果使用大曲线，那么它会出现锯齿，不好看。

基本的标记与前一个方法相似，它需要4个元素来应用4个角蒙板：

```
<div class="box">
  <div class="box-outer">
    <div class="box-inner">
      <h2>Headline</h2>
      <p>Content</p>
    </div>
  </div>
</div>
```

CSS也非常相似：

```
.box {
  width: 20em;
  background: #effce7 url(/img/bottom-left.gif) no-repeat left bottom;
}

.box-outer {
  background: url(/img/bottom-right.gif) no-repeat right bottom;
  padding-bottom: 5%;
}

.box-inner {
  background: url(/img/top-left.gif) no-repeat left top;
}

.box h2 {
  background: url(/img/top-right.gif) no-repeat right top;
  padding-top: 5%;
}

.box h2, .box p {
  padding-left: 5%;
  padding-right: 5%;
}
```

4

除了使用不同的图像之外，主要的差异是在主框div上添加了背景颜色。如果要修改框的颜色，只需修改CSS中的颜色值，不必重新创建任何新图像。这个方法只适合创建非常简单的框，但是它提供了很大的灵活性，而且可以在不同的项目中重复使用。

## 1. 多个背景图像

前面的示例很有意思，但是其中大多数都必须在代码中添加非语义性标记。需要这些多余的标记是因为在一个元素中只能添加一个背景图像。如果可以添加多个背景图像，那不是很方便吗？这通过CSS 3就可以实现。另外，语法非常简单，它采用与一般背景图像相同的形式。主要

差异是，不是定义一个背景图像，而是可以使用任意数量的图像。具体做法如下：

```
.box {  
background-image: url(/img/top-left.gif),  
                url(/img/top-right.gif),  
                url(/img/bottom-left.gif),  
                url(/img/bottom-right.gif);  
  
background-repeat: no-repeat,  
                  no-repeat,  
                  no-repeat,  
                  no-repeat;  
  
background-position: top left,  
                    top right,  
                    bottom left,  
                    bottom right;  
}  
  
<div class="box">  
  <h2>Headline</h2>  
  <p>Content</p>  
</div>
```

首先，使用background-image属性定义要使用的所有图像。接下来，指定它们是否应该重复显示。最后，使用background-position属性设置它们的位置。结果见图4-12。Safari从1.3版开始支持多个背景图像，Firefox和Opera的最新版本现在也支持这个特性。IE目前还不支持多个背景图像，但是这并不妨碍使用这种技术，只是IE用户看到的是直角框。

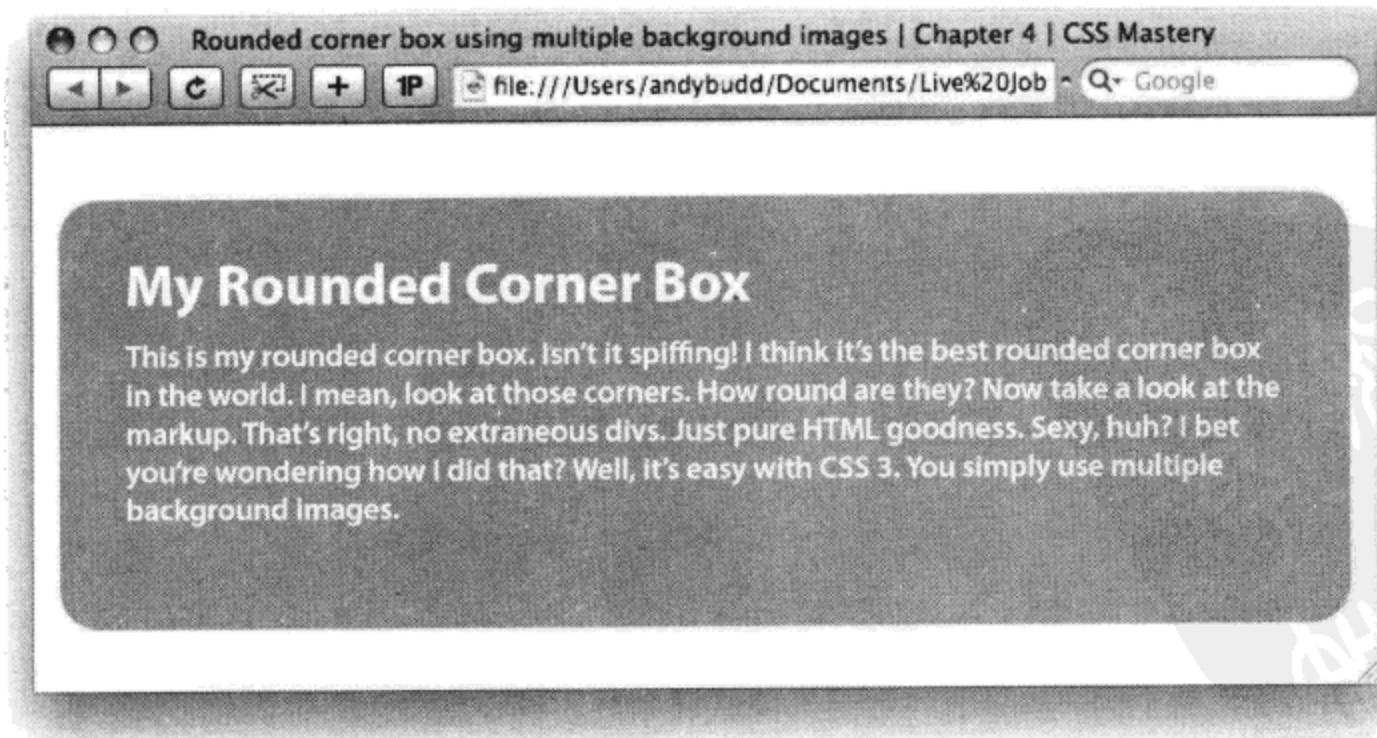


图4-12 使用CSS 3多背景特性实现的圆角框

## 2. border-radius

目前有许多高水平的计算机游戏使用动态的纹理映射，图形技术已经普及到这种程度，所以你会认为浏览器本身应该能够绘制简单的圆角框，根本不需要我们添加图像。是的，由于有了CSS 3的border-radius属性，现在确实可以。我们只需设置边框角的半径，浏览器就会实现这种效果（见图4-13）。

```
.box {
    border-radius: 1em;
}
```

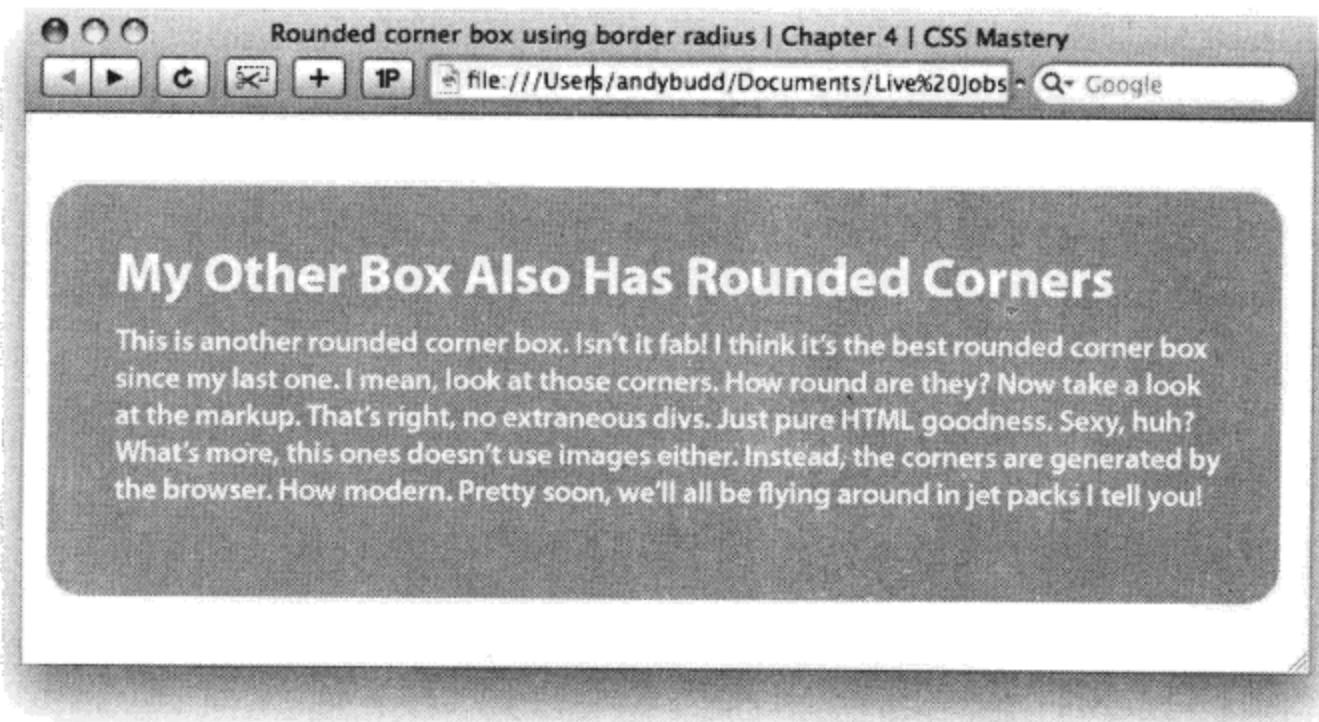


图4-13 使用CSS 3 border-radius属性实现的圆角框

这是一个新属性，对于它的实际实现方式还有争议。所以在这个属性得到广泛使用之前，需要使用与浏览器相关的扩展调用它。当前，Firefox和Safari支持这个属性，所以我使用-moz和-webkit前缀。

```
.box {
    -moz-border-radius: 1em;
    -webkit-border-radius: 1em;
    border-radius: 1em;
}
```

浏览器生产商一直在试验CSS的新扩展。一些生产商率先实现了CSS的新特性，其他生产商还在努力。一些扩展可能从不会出现在正式规范中，比如那些用于在iPhone上显示UI元素的Safari扩展。

因此，为了避免与其他用户代理混淆或者破坏代码的有效性，可以通过在选择器、属性或值上添加与厂商相关的前缀来调用这些扩展。例如，Mozilla 使用 -moz 前缀，Safari 使用 -webkit 前缀。IE、Opera 和所有主流浏览器都有相似的前缀。可以使用这些前缀访问每种浏览器特有的特性，可能需要在厂商的开发人员站点上查阅可用的特性。

通过使用这种机制，可以在新的 CSS 3 特性成为正式规范之前试用它们。但是，使用这些扩展时一定要小心，因为在不同的浏览器中这些试验性特性的格式可能不一样，而且当正式规范发布时它们可能有变化或取消了。

### 3. border-image

要讨论的最后一种CSS 3新特性是border-image属性。这个新属性允许指定一个图像作为元素的边框。一个图像有什么好处呢？这个属性的优点是，可以根据一些简单的百分比规则把图像划分为9个单独的部分，浏览器会自动地使用适当的部分作为边框的对应部分。这种技术称为九分法缩放（nine-slice scaling），它有助于避免在调整圆角框大小时通常会出现的失真。它有点儿不容易理解，所以我认为应该提供一个例子。

假设有一个100像素高的曲线框图像，见图4-14。在距离框的顶边和底边25%的地方画两条线，再在距离左边和右边25%的地方画两条线，这个框就分成了9个部分。

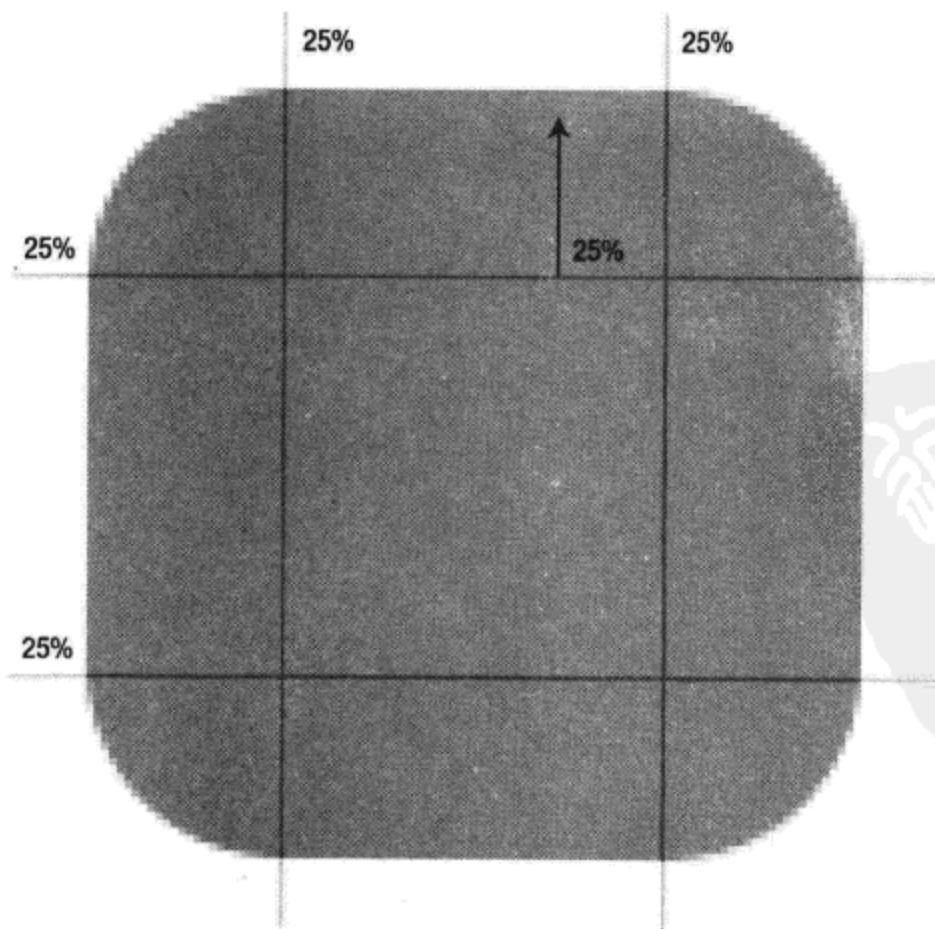


图4-14 边框图像的源文件。为了说明，加上了分割线

`border-image`属性自动地把图像的每个部分用于对应的边框。因此，图像的左上部分用作左上边框，右边中间部分用作右边的边框。我希望边框的宽度为25像素，所以在CSS中设置这个宽度。如果图像不够大，它们会自动地平铺，产生一个可扩展的框（见图4-15）。实现这种效果的代码如下：

```
.box {  
    -webkit-border-image: url(/img/corners.gif)  
    25% 25% 25% 25% / 25px round round;  
}
```

Safari支持这个属性，但是要使用像本例中的Webkit特有的扩展。Firefox 3.5和Opera 9.5现在也支持`border-image`，这大大增加了它的受众范围。

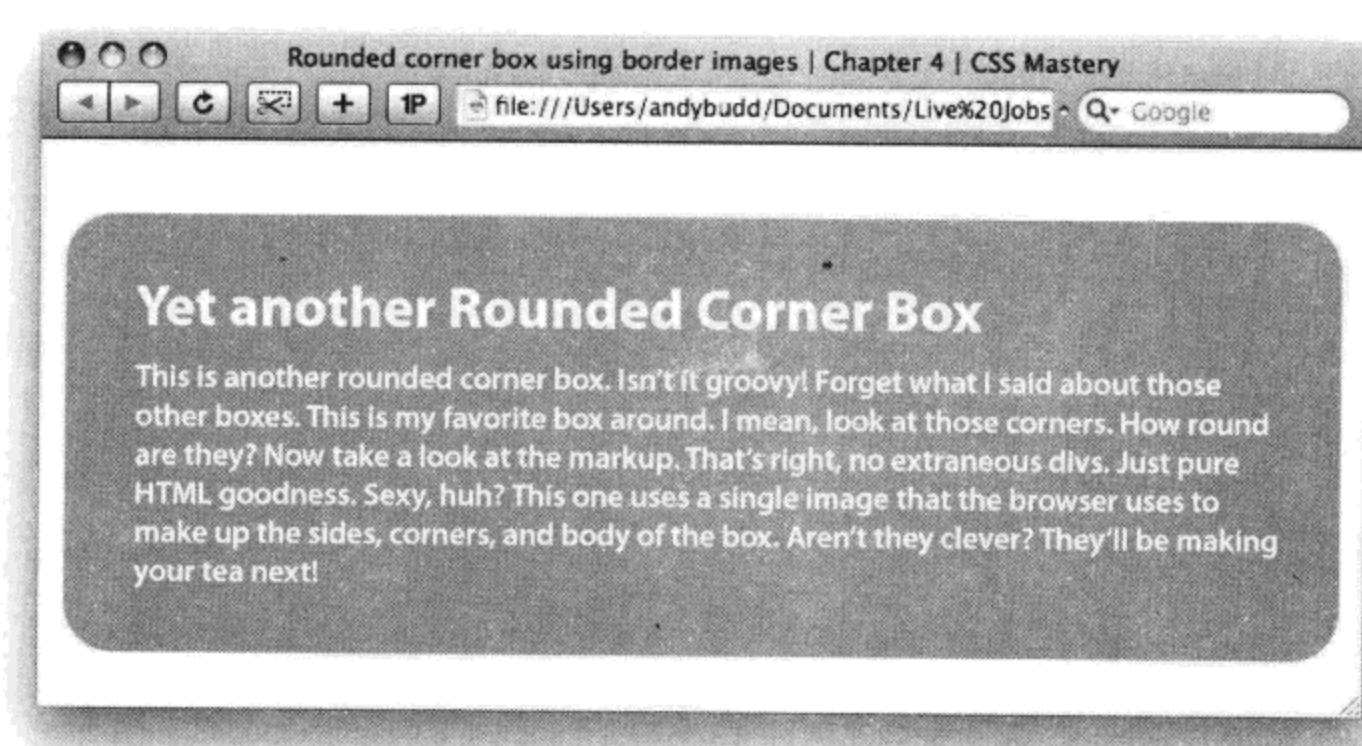


图4-15 使用`border-image`属性实现的圆角框

## 4.3 投影

投影是一种很流行、很有吸引力的设计特性，它给平淡的设计增加了深度，形成立体感。大多数人使用Photoshop这样的图形软件直接给图像添加投影。但是，可以使用CSS产生简单的投影效果，而不需要修改底层的图像。

这么做有几个原因。例如，你可能让非技术人员管理站点，而他不会使用Photoshop，或者要从无法使用Photoshop的地方（比如网吧）上传图像。通过使用预定义的投影样式，只需上传常规图像，它在站点上就会显示投影。

使用CSS的最大好处之一是灵活性。如果以后想去掉投影效果，那么只需要在CSS文件中修改几行代码，而不必重新处理所有图像。

### 4.3.1 简单的CSS投影

www.1976design.com的Dunstan Orchard首先描述了这个非常简单的投影方法。它的工作原理是：将一个大的投影图像应用于容器div的背景。然后使用负的外边距偏移这个图像，从而显示出投影。

首先需要创建投影图像，我使用的是Adobe Photoshop。创建一个新的Photoshop文件，其尺寸与图像的最大尺寸一样。为了保险，我创建一个800×800像素的文件。然后打开背景层并且填充一种颜色，投影将放在这种颜色上面。我让背景层保持白色。接着创建一个新的层并且填充上白色。现在，将这个层向左上方移动4或5个像素，然后对这个层应用4或5像素宽的投影。保存这个文件并将它命名为shadow.gif（见图4-16）。

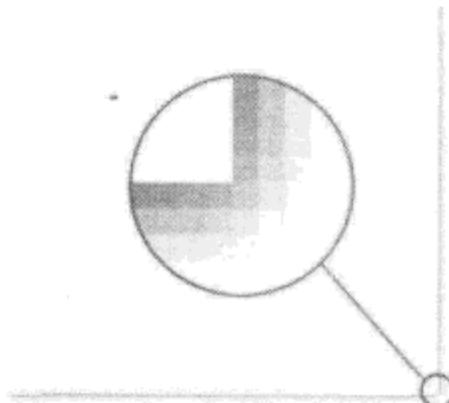


图4-16 800×800像素的shadow.gif，在放大图中可以看到5像素宽的投影

这种技术的标记非常简单：

```
<div class="img-wrapper"></div>
```

为了产生投影效果，首先需要将投影图像应用于容器div的背景。因为div是块级元素，所以它们会水平伸展，占据所有可用空间。在这种情况下，我们希望div包围图像。可以显式地设置容器div的宽度，但是这么做会限制这种技术的用途。可以浮动div，让它在现代浏览器上产生“收缩包围”的效果。Mac上的IE 5.x不支持这种技术，可以对Mac上的IE 5.x隐藏这些样式。关于对各种浏览器隐藏规则的更多信息，请参见第8章，那里会讨论各种hack和过滤器。

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

为了露出投影图像并产生投影效果（见图4-17），需要使用负的外边距偏移这个图像：

```
.img-wrapper img {  
margin: -5px 5px 5px -5px;  
}
```



图4-17 应用了投影的图像

还可以给图像加上边框和一些内边距，从而产生类似照片边框的效果（见图4-18）：

```
.img-wrapper img {  
background-color: #fff;  
border: 1px solid #a9a9a9;  
padding: 4px;  
margin: -5px 5px 5px -5px;  
}
```



图4-18 简单的投影技术的最终结果

这种技术对于大多数符合标准的现代浏览器都是有效的。但是，为了在IE 6中产生正确的效果，还需要添加两个简单的规则：

```
.img-wrapper {  
    background: url(/img/shadow.gif) no-repeat bottom right;  
    clear: right;  
    float: left;  
    position: relative;  
}  
  
.img-wrapper img {  
    background-color: #fff;  
    border: 1px solid #a9a9a9;  
    padding: 4px;  
    display: block;  
    margin: -5px 5px 5px -5px;  
    position: relative;  
}
```

投影效果现在可以在IE 6中实现了。

### 4.3.2 来自 Clagnut 的投影方法

www.clagnut.com的Richard Rutter提供了一个相似的创建投影的方法。他不使用负的外边距，而是使用相对定位来偏移图像：

```
.img-wrapper {  
    background: url(/img/shadow.gif) no-repeat bottom right;  
    float:left;  
    line-height:0;  
}  
  
.img-wrapper img {  
    background:#fff;  
    padding:4px;  
    border:1px solid #a9a9a9;  
    position:relative;  
    left:-5px;  
    top:-5px;  
}
```

#### Box-shadow

尽管前面的技术可以到达目的，但是很麻烦。如果浏览器本身可以创建投影，就不需要使用Photoshop滤镜和图像了，这会方便得多。CSS 3也支持这种做法，这需要使用box-shadow属性。这个属性需要4个值：垂直和水平偏移、投影的宽度（也就是模糊程度）和颜色。在下面的示例中，把投影偏移3像素，宽度设置为6像素，颜色为中等灰色（见图4-19）。

```
img {
  box-shadow: 3px 3px 6px #666;
}
```



4

图4-19 使用box-shadow属性实现的投影

这也是一个试验性的CSS 3属性，所以目前需要使用Safari和Firefox扩展。但是，这个属性可能很快就会得到广泛支持了。

```
img {
  -webkit-box-shadow: 3px 3px 6px #666;
  -moz-box-shadow: 3px 3px 6px #666;
  box-shadow: 3px 3px 6px #666;
}
```

这个特性最让人兴奋的特点之一是它会与border-radius属性相互配合（见图4-20）。这意味着可以通过编程在圆角框上创建投影，根本不需要使用图形软件！

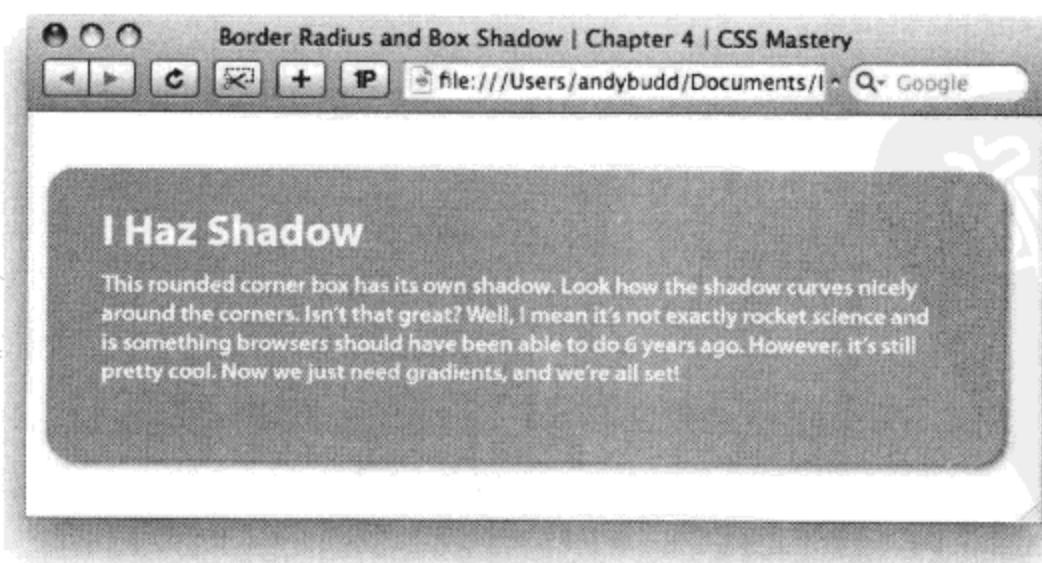


图4-20 圆角框上的投影

我们在UX London 2009网站上大量使用了这种效果，这在现代浏览器中会显示投影（见图4-21），在功能落后的浏览器中显示一般的框架（见图4-22）。



图4-21 Firefox显示的UX London网站。请注意使用CSS 3创建的投影



图4-22 在IE中看到的UX London网站。投影没有了，但是大多数用户不会注意到

## 4.4 不透明度

适当地使用不透明度可以让设计的效果更丰富。对于相互重叠的元素，还可以用它显露下面的元素。除了这个很酷的用途外，这还可以改进站点的易用性。

### 1. CSS不透明度

大多数现代浏览器支持CSS不透明度已经有相当一段时间了，但让我吃惊的是设计者不经常使用它。IE的老版本不支持它。但是，专门针对IE的代码很容易解决这个问题。下面举一个例子。假设要弹出一个警告消息，它应该覆盖在现有文档上面，同时你仍然可以看到下面的东西（见图4-23）。

```
.alert {
    background-color: #000;
    border-radius: 2em;
    opacity: 0.8;
    filter: alpha(opacity=80); /*proprietary IE code*/
}
```

CSS不透明度的主要问题是，除了对背景生效之外，应用它的元素的内容也会继承它。因此，如果仔细看看图4-23，会透过警告文本看到页面上的文本。如果使用非常高的不透明度和高对比度的文本，这可能不是问题。但是，如果不透明度低，框的内容就会难以辨认。RGBa就是为了解决这个问题而设计的。

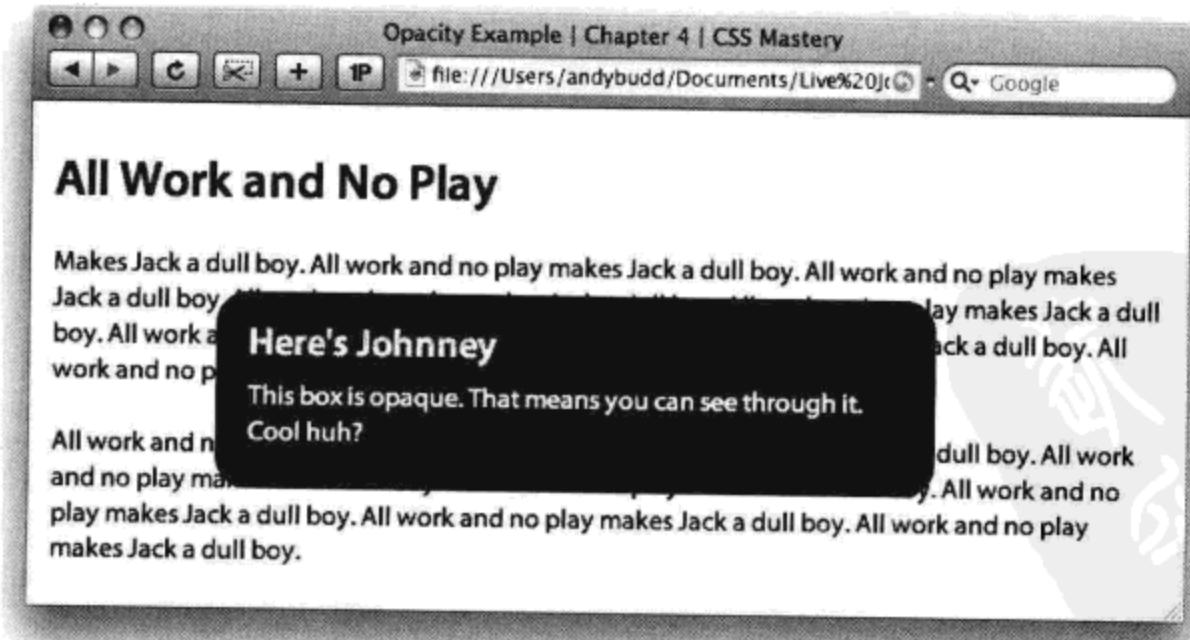


图4-23 不透明度为80%的圆角警告框

### 2. RGBa

RGBa是一种同时设置颜色和不透明度的机制。RGB代表红色、绿色和蓝色，a代表alpha透

明度。在前一个示例中使用RGBa的方法如下：

```
.alert {  
background-color: rgba(0,0,0,0.8);  
border-radius: 2em;  
}
```

前3个数字表示颜色的红、绿和蓝值。在这里，警告框是黑色的，所以这3个值都设置为0。与不透明度一样，最后一个数字是十进制的不透明度值，所以0.8表示这个背景的不透明度是80%，换句话说，透明度是20%。值为1表示100%不透明，值为0表示完全透明。这种技术的结果见图4-24。

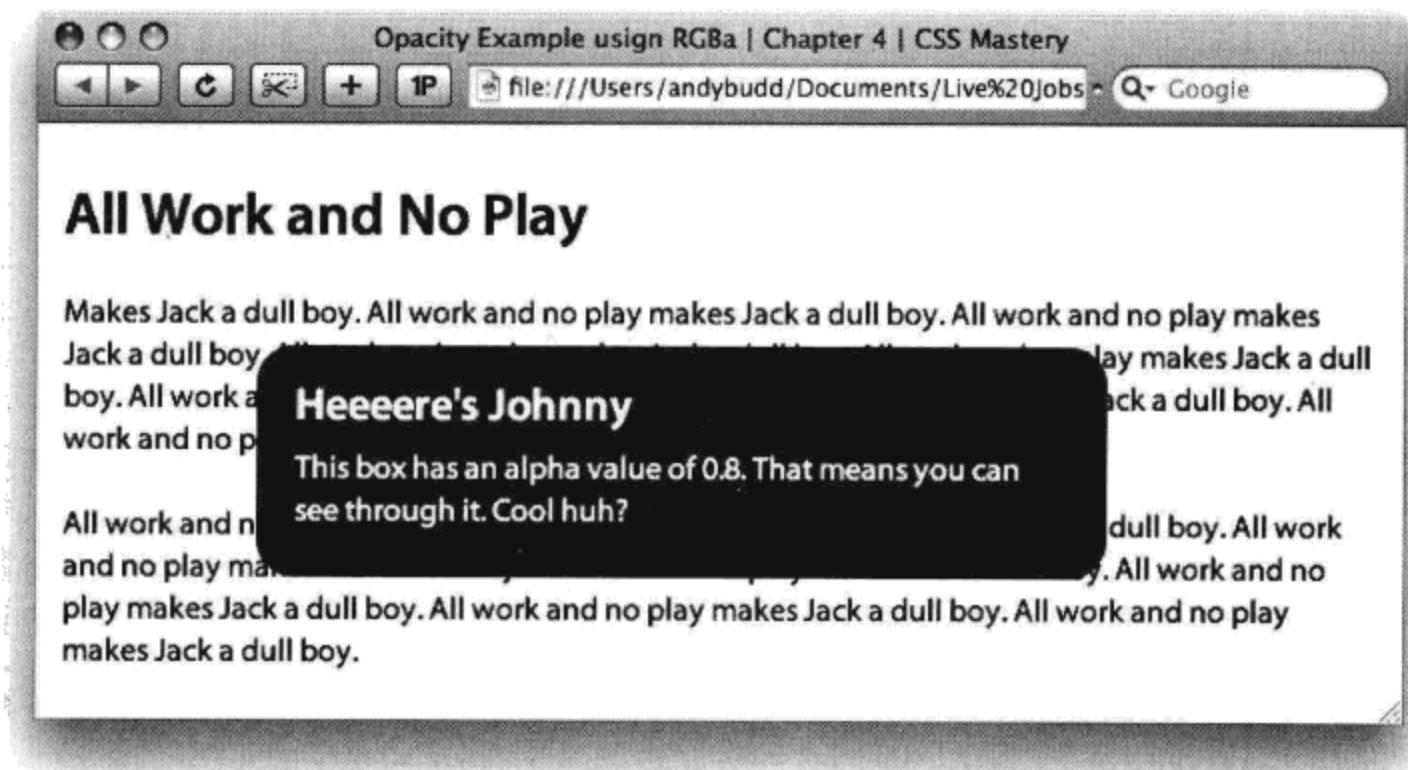


图4-24 使用RGBa实现的不透明度为80%的圆角警告框

### 3. PNG透明度

PNG文件格式最大的优点之一是它支持alpha透明度。这可以使设计具备真正的创意（见图4-25）。但是，IE 6不直接支持PNG透明度，而IE 7和IE 8支持。对于IE的老版本，有两种解决方法。

在IE 6中支持PNG透明度的方法是使用专有的AlphaImageLoader过滤器。为此，需要在CSS中包含以下代码行。

```
filter:progid:DXImageTransform.Microsoft.AlphaImageLoader  
(src=/img/my-image.png', sizingMethod='crop');
```



图4-25 老的Revyver站点在视口底部有一个漂亮的PNG透明度示例。在滚动页面时，会透过树枝和彩虹看到内容

但是，使用这行代码会导致CSS无效，所以最好把它放在IE 6专用的样式表中。

```
.img-wrapper div {
    filter: progid:DXImageTransform.Microsoft.AlphaImageLoader
    (src='/img/shadow2.png', sizingMethod='crop');
    background: none;
}
```

第一个规则使用专有的过滤器加载PNG并应用alpha透明度。原来的背景图像仍然会显示，所以第二个规则隐藏原来的背景图像。

IE还有另一种称为“有条件注释”的专有代码，这让我们可以向IE的特定版本提供特定的样式表。这里希望只让IE 6看到这个新的样式表，所以可以在页面顶部添加以下代码：

```
<!--[if ie 6]>
<link rel="stylesheet" type="text/css" href="ie6.css"/>
<![endif]-->
```

目前不需要太关注有条件注释，第8章会详细讨论它。

这种技术的问题是，对于要使用的每个透明PNG，都必须包含这行代码。因此使用起来有点儿麻烦。

另一种方法是使用IE PNG fix技术。这需要使用一种不太为人所知的Microsoft专有CSS扩展——行为（behavior）。下载合适的.htc文件并在IE 6专用的样式表中引用它，就可以在任何元素上启用PNG透明度。

```
img, div {  
    behavior: url(iepngfix.htc);  
}
```

如需了解关于这种技术的更多信息和下载.htm文件，请访问[visit www.twinhelix.com/css/iepngfix](http://www.twinhelix.com/css/iepngfix)。

#### 4. CSS视差效果

背景图像不仅可以创建圆角框和投影，还可以实现许多有趣的效果。访问我们的Silverback易用性测试应用程序就可以体会到这一点。如果访问[www.silverbackapp.com](http://www.silverbackapp.com)并重新调整浏览器窗口的大小，就会看到一种奇怪的效果（见图4-26）。背景图像会以稍微不同的速度移动，让人觉得这个页面有深度。这种现象称为视差滚动，许多老式计算机游戏大量使用了这种技术。

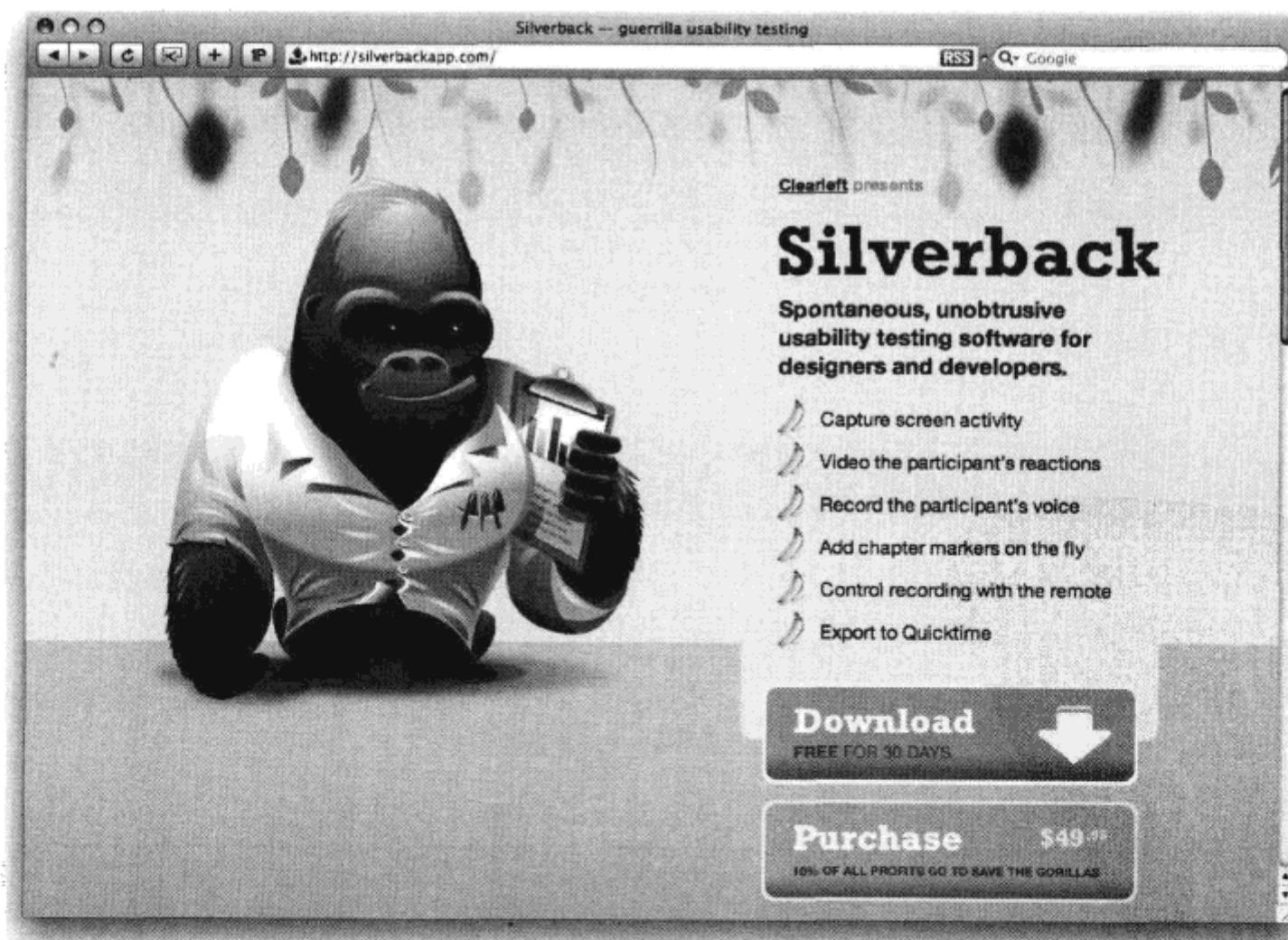


图4-26 在[www.silverbackapp.com](http://www.silverbackapp.com)上改变窗口大小，看看效果

要想实现这种效果，首先需要创建几个不同的背景图像。一个图像是绿色背景上的藤蔓，另两个图像是alpha透明背景上的藤蔓。这样中景和前景图像可以相互覆盖并盖在背景上，同时不会把视图弄得模糊。

主背景将应用在body元素上。但是，如果我们不使用CSS 3的多个背景图像特性，就需要添加两个新元素来应用背景。页面的内容需要出现在这些元素的前面，让用户可以与它交互。可以把前景div放在内容前面，但是这会挡住部分内容，用户很难与它交互。所以标记结构应该像这样：

```
<body>
  <div class="midground">
    <div class="foreground">
      <p>Your content will go here!</p>
    </div>
  </div>
</body>
```

首先，需要在body元素上添加主背景。若希望这个图像水平平铺，就需要把image-repeat属性设置为repeat-x。还需要让body元素显示背景颜色（在这里是浅绿色）。最后，把图像相对于窗口大小水平偏移20%，这是关键之处。当水平调整窗口大小时，背景图像的位置会改变，看起来像是移动了。

```
body {
  background-image: url(/img/bg-rear.jpg);
  background-repeat: repeat-x;
  background-color:#d3ff99;
  background-position: 20% 0;
}
```

现在需要对中景和前景图像做同样的处理，但是选择比较高的百分数，让它们相对移动得更快，产生有深度的感觉。我们把中景的位置设置为40%，把前景位置设置为150%。你可以调整这些位置，得到自己需要的效果。

```
body {
  background-image: url(/img/bg-rear.jpg);
  background-repeat: repeat-x;
  background-color:#d3ff99;
  background-position: 20% 0;
}

.midground {
  background-image: url(/img/bg-mid.png);
  background-repeat: repeat-x;
  background-color: transparent;
  background-position: 40% 0;
```

```
}

.foreground {
    background-image: url(/img/bg-front.png);
    background-repeat: repeat-x;
    background-color: transparent;
    background-position: 150% 0;
}
```

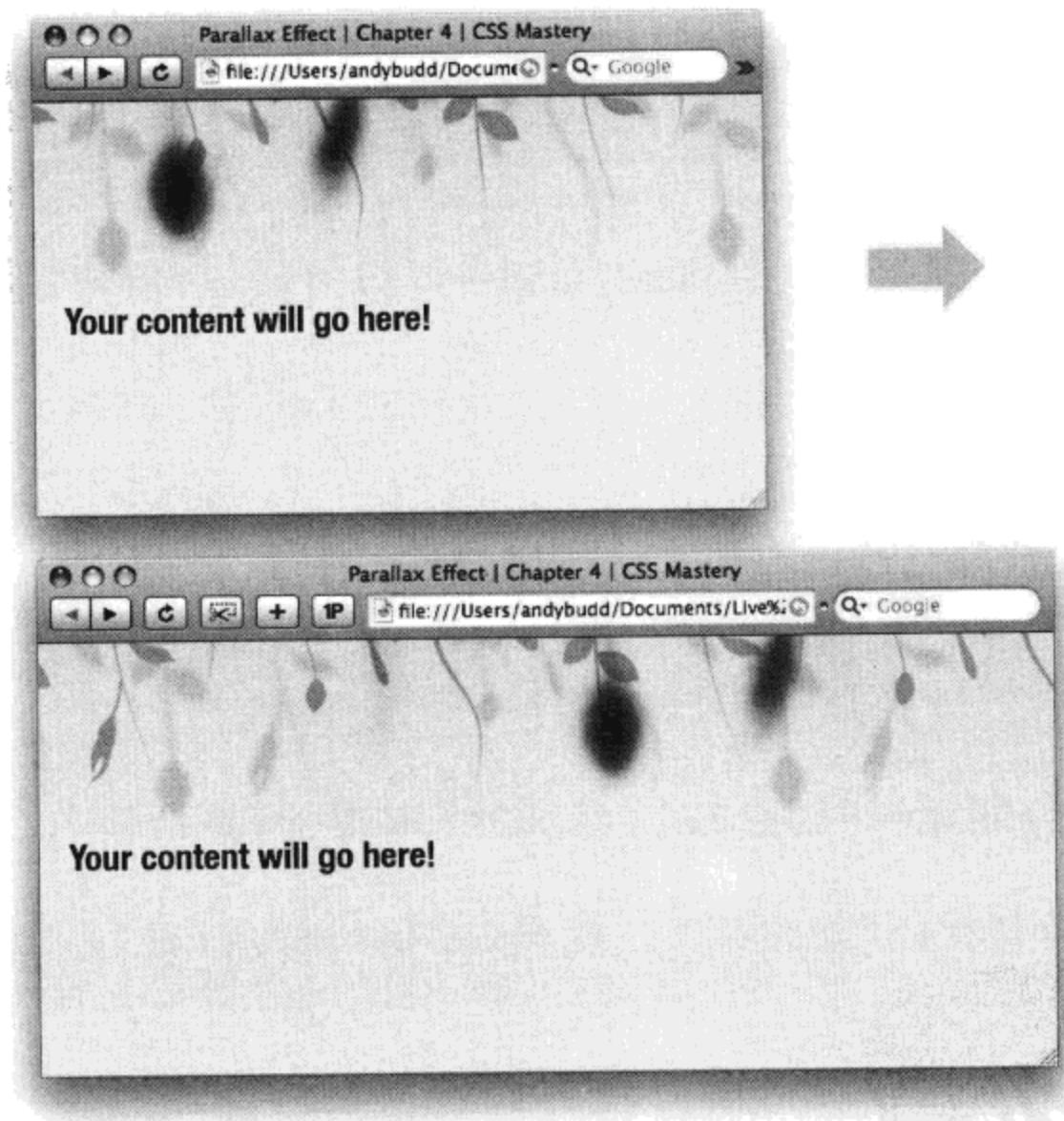


图4-27 在水平调整窗口大小时，背景中的藤蔓以不同的速度移动，产生有深度的感觉

## 4.5 图像替换

HTML文本具有很多优点。搜索引擎可以读取它，开发人员可以复制并粘贴它，如果在浏览器中增加文本字号，就可以让它放大。因此，尽可能使用HTML文本而不是文本的图像是一种好想法。遗憾的是，页面设计人员能选择的字体很有限。另外，尽管可以使用CSS在某个范围内控制版式，但是某些效果对于文本是不可能实现的。因此，在某些情况（通常是为了树立品牌）下

还是需要使用文本的图像。

由于不愿意将这些图像直接嵌入页面中，CSS作者发明了图像替换（image replacement）的概念。你可以像平常一样将文本添加到文档中，然后使用CSS隐藏文本并在它的位置上显示一个背景图像。这样的话，搜索引擎仍然可以搜索HTML文本，而且如果禁用CSS，文本仍然会显示。

在各种缺陷暴露出来之前，这个概念看起来非常棒。但是，一些比较流行的图像替换方法对于屏幕阅读器<sup>①</sup>是无效的，而且如果关闭图像但是打开CSS，就会出现内容缺失。因此，许多CSS作者停止使用图像替换方法，恢复为使用一般文本。虽然我也主张尽可能避免图像替换，但是仍然相信在某些情况下它是合适的，比如由于公司品牌策略的要求，需要使用特定的字体。为此，你应该很好地掌握各种技术并且了解它们的局限性。

### 4.5.1 FIR

由Todd Fahrner开创的FIR（Fahrner图像替换）是最早且可能最流行的图像替换技术。我要解释这个方法是因为它在历史上具有重要意义，而且它是最容易理解的方法之一。但是，这个方法有一些严重的可访问性问题（稍后会提到），因此应该避免使用。

基本概念非常简单。把要替换掉的文本放在标签中：

```
<h2>
  <span>Hello World</span>
</h2>
```

然后将替换图像作为背景图像应用于标题元素：

```
h2 {
  background:url(hello_world.gif) no-repeat;
  width: 150px;
  height: 35px;
}
```

并且将的display值设置为none，从而隐藏span的内容：

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

这个方法似乎很有效，但是最后一个规则会造成问题。许多流行的屏幕阅读器会忽略那些display值为none或visibility为hidden的元素。因此会完全忽略这个文本，造成严重的可访问性问题。所以，这种试图改进站点可访问性的技术实际上产生了相反的效果。因此，最好不要

<sup>①</sup> 所谓屏幕阅读器，是指将屏幕显示内容转为声音或布莱叶盲文显示，以帮助视力障碍者使用电脑的软件。

——译者注

使用这种技术。

### 4.5.2 Phark

[www.phark.net](http://www.phark.net)的Mike Rundle发明了一种适合屏幕阅读器的图像替换技术，而且不需要添加额外的无语义div:

```
<h2>  
    Hello World  
</h2>
```

Phark方法并不使用display属性来隐藏文本，而是对标题进行非常大的负值文本缩进：

```
h2 {  
    text-indent: -5000px;  
    background:url(/img/hello_world.gif) no-repeat;  
    width: 150px;  
    height:35px;  
}
```

这个方法效果很好，解决了屏幕阅读器的问题。但是，与FIR方法一样，这个方法在关闭图像但是打开CSS的情况下是无效的。这是一种很少见的情况，可能只有使用非常慢的连接或者使用手机作为调制解调器的人才会这样设置。站点访问者能够打开图像，但是他们可能选择不打开。要记住某些人可能看不见被替换的文本，所以对于重要信息或导航信息，最好避免使用这个方法。

### 4.5.3 sIFR

图像替换试图解决的主要问题之一是在大多数计算机上缺少可用的字体。为了避免将文本换成文本的图像，Mike Davidson和Shaun Inman一起发明了一种更新颖的方法。

Flash允许将字体嵌入SWF文件，所以他们并不把文本换成图像，而是用Flash文件替换文本。进行这一替换的方法是使用JavaScript搜索文档，找到特定元素或者具有特定类名的元素中的所有文本。然后，JavaScript将文本替换为一个小的Flash文件。接下来是真正精彩的部分。这种技术并不为每段文本创建单独的Flash文件，而是将被替换的文本放回一个重复的Flash文件中。因此，进行图像替换所要做的只是添加一个类，Flash和JavaScript会完成余下的工作。另一个好处是Flash文件中的文本是可选择的，这意味着可以轻松地复制和粘贴它。

Shaun Inman公开了他的Flash图像替换方法，并且将它命名为IFR（Inman Flash替换）。IFR是一种非常方便的方法。关于这个方法的细节（包括源代码）可以在[www.shauninman.com/plete/2004/04/ifr-revisited-and-revised](http://www.shauninman.com/plete/2004/04/ifr-revisited-and-revised)上找到。

Mike Davidson对这个方法进行了扩展，创建了sIFR（可伸缩Inman Flash替换）方法。这个方法可使多行文本替换和改变文本字号。sIFR现在由Mark Wubben维护和开发，包含许多有意思

新特性。

要想在站点上使用sIFR，首先需要从`http://novemberborn.net/sifr3`下载最新的版本。在站点上安装sIFR是非常简单的，但是应该先阅读文档。首先需要做的是打开Flash文件、嵌入希望使用的字体并且导出视频。为了让sIFR能够正确地工作，接下来需要应用包含的打印和屏幕样式，或者创建自己的样式。现在，将`sifr.js` JavaScript文件添加到希望使用sIFR的每个页面中。可以对这个文件进行许多配置，可以指定要替换的元素、文本颜色、内边距、大小写以及其他许多样式属性。但是，内边距和行高都影响文本大小，所以，实际上这不容易。完成配置之后，将所有文件上传到服务器上，就会看到原来的字体换成了动态的Flash内容。

这些技术的主要问题涉及加载时间。页面必须完全加载，然后JavaScript才能替换文本。因此，在所有文本被替换为Flash内容之前常常有短暂的闪烁（见图4-28）。

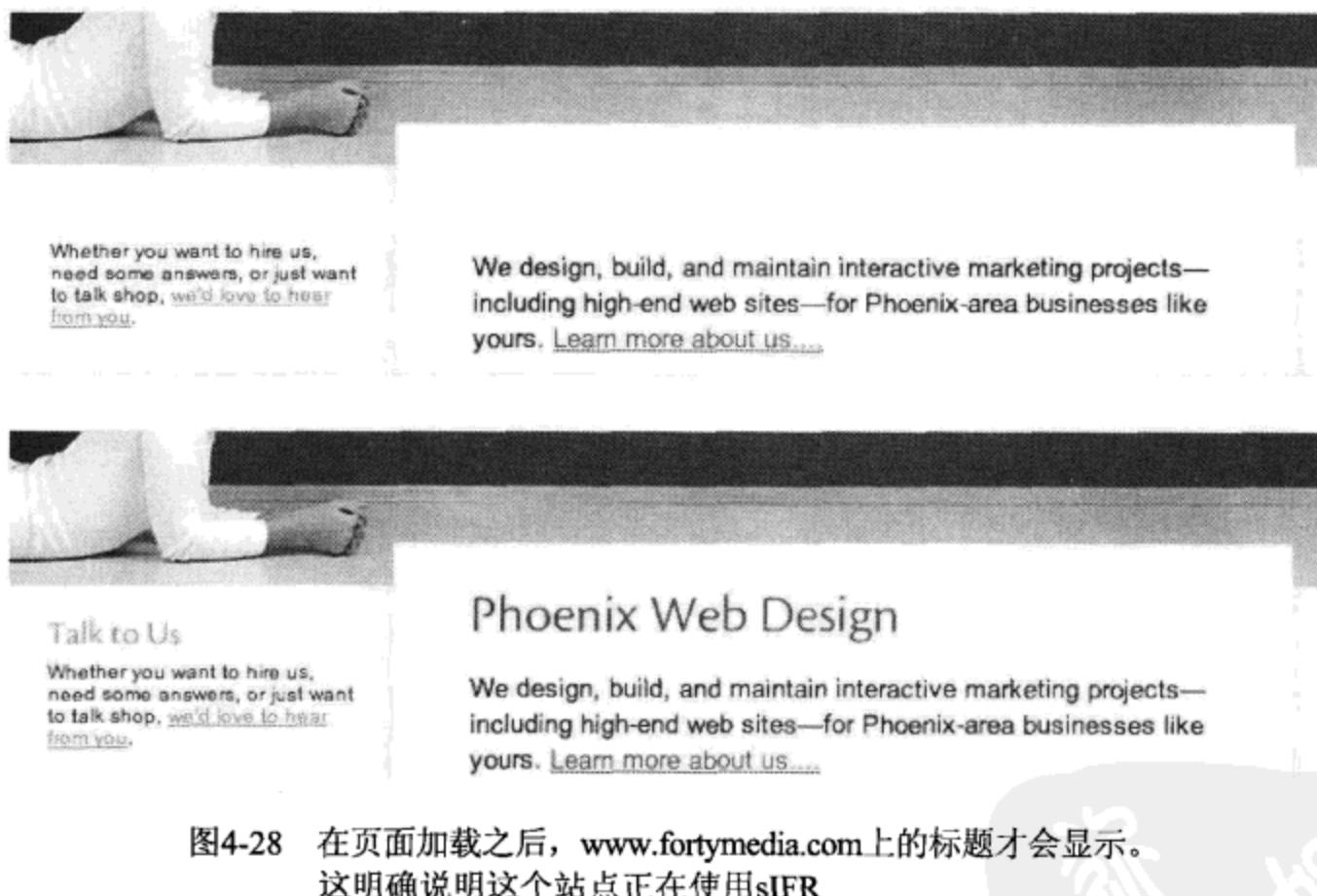


图4-28 在页面加载之后，`www.fortymedia.com`上的标题才会显示。  
这明确说明这个站点正在使用sIFR

尽管这不是个大问题，但是会被访问者注意到，会给人留下页面加载慢的印象。另外，如果进行许多Flash替换，一些页面感觉起来有点儿迟钝。用户还可能看到“无样式内容的Flash”，这会让用户觉得页面很糟糕，让人很糊涂。最好将替换减少到最少，只对主要标题使用这种技术。sIFR是在Web上提供丰富的排版效果的好方法，非常适合那些采用一致的排版处理的小型站点。但是，对于那些具有多种标题大小、样式和颜色的大型站点，应用sIFR需要细致地处理。如果一些标题跨多行或者有背景颜色，也比较麻烦。所以我的建议是不要在大型项目中使用sIFR，只在个人站点上使用它。

## 4.6 小结

在本章中，我们学习了如何将背景图像应用于元素，从而产生各种有意思的效果，比如灵活的圆角框和纯CSS投影。你看到了border-radius和box-shadow等新的CSS 3属性如何替代这些效果，还学习了不透明度、几种图像替换方法以及如何在IE中实现PNG支持。

在下一章中，我们将学习如何结合使用背景图像和链接来实现一些有意思的效果。





## 第5章

# 对链接应用样式

5

锚链接是万维网的基础。这种机制使网页可以相互连接，人们可以在页面之间切换。锚链接的默认样式很没有吸引力，但是只使用一点儿CSS就可以产生令人吃惊的效果。

在本章中，你将学习以下内容。

- 基于层叠对链接选择器进行排序。
- 创建应用了样式的链接下划线。
- 使用属性选择器对外部链接应用样式。
- 使链接表现得像按钮。
- 创建已访问链接样式。
- 创建纯CSS的工具提示。

## 5.1 简单的链接样式

对链接应用样式最容易的方式是使用锚类型选择器。例如，以下规则使所有锚显示为红色：

```
a {color: red;}
```

但是，锚可以作为内部引用，也可以作为外部链接，所以使用类型选择器不总是理想的。例如，下面的第一个锚包含一个片段标识符，当用户单击这个锚时，页面将跳到第二个锚的位置：

```
<p><a href="#mainContent">Skip to main content</a></p>
...
<h1><a name="mainContent">Welcome</a></h1>
```

虽然只想让真正的链接变成红色，但是标题的内容也成了红色的。为了避免这个问题，CSS提供了两个特殊的选择器，称为链接伪类选择器。`:link`伪类选择器用来寻找没有被访问过的链接，`:visited`伪类选择器用来寻找被访问过的链接。所以，在下面的示例中，所有没有被访问过的链接将是蓝色的，所有被访问过的链接将是绿色的：

```
a:link {color: blue;} /* Makes unvisited links blue */
a:visited {color: green;} /* Makes visited links green */
```

可以用来对链接应用样式的另外两个选择器是`:hover`和`:active`动态伪类选择器。`:hover`动态伪类选择器用来寻找鼠标悬停处的元素，`:active`动态伪类选择器用来寻找被激活的元素。对于链接来说，激活发生在链接被单击时。所以，在下面的示例中，当鼠标悬停在链接上或单击链接时，链接将变成红色：

```
a:hover, a:active { color: red; }
```

为了尽可能提高页面的可访问性，在定义鼠标悬停状态时，最好在链接上添加`:focus`伪类。在通过键盘移动到链接上时，这让链接显示的样式与鼠标悬停时相同。

```
a:hover, a:focus { color: red; }
```

其他元素也可以使用`:hover`、`:active`或`:focus`伪类选择器。例如，可以在表格行上添加`:hover`伪类，在提交按钮上添加`:active`伪类，在输入框上添加`:focus`伪类，从而突出显示各种交互形式。IE 7和更低版本不支持在除链接之外的其他元素上使用伪类选择器，但是所有现代浏览器都支持。

```
/* makes table rows yellow when hovered over */
tr:hover {
    background: yellow;
}

/* makes submit buttons in some browsers yellow when pressed */
input[type="submit"]:active {
    background: yellow;
}

/* makes inputs yellow when selected */
input:focus {
    background: yellow;
}
```

大多数人最初使用这些选择器做的第一件事就是去掉链接的下划线，然后在鼠标悬停在链接上或单击链接时取消下划线。实现方法是将未访问和已访问的链接的`text-decoration`属性设置为`none`，将鼠标悬停其上和已激活的链接的`text-decoration`属性设置为`underline`：

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

在前面的示例中，选择器的次序非常重要。如果次序反过来，鼠标悬停和激活样式就不起作用了：

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

这是由层叠造成的。在第1章中提到过，当两个规则具有相同的特殊性时，后定义的规则优先。在这个示例中，两个规则具有相同的特殊性，所以`:link`和`:visited`样式将覆盖`:hover`和`:active`样式。为了确保不会发生这种情况，最好按照以下次序应用链接样式：

```
a:link, a:visited, a:hover, a:focus, a:active
```

记住这个次序的简单方法是记住Lord Vader Hates Furry Animals，其中的L代表link，V代表visited，H代表hover，F代表focus，A代表active。

5

## 5.2 让下划线更有趣

从易用性和可访问性的角度来说，通过颜色之外的某些方式让链接区别于其他内容是很重要的。这是因为许多有视觉障碍的人很难区分对比不强烈的颜色，尤其是在文本比较小的情况下。例如，有色盲症的人无法区分具有相似亮度或饱和度的某些颜色。因此，默认情况下，链接是有下划线的。

### 5.2.1 简单的链接修饰

但是，设计人员往往不喜欢链接的下划线，因为下划线让页面看上去比较乱。如果决定去掉链接的下划线，那么可以让链接显示为粗体。这样的话，页面看起来没那么乱，而链接仍然醒目：

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

当鼠标悬停在链接上或激活链接时，可以重新应用下划线，从而增强其交互状态：

```
a:hover, a:focus, a:active {
  text-decoration: underline;
  font-weight: bold;
}
```

但是，也可以使用边框创建不太影响美观的下划线。在下面的示例中，取消默认的下划线，

将它替换为不太刺眼的点线。当鼠标悬停在链接上或激活链接时，这条线变成实线，从而向用户提供视觉反馈：

```
a:link, a:visited {  
    text-decoration: none;  
    border-bottom: 1px dotted #000;  
}  
  
a:hover, a:focus, a:active {  
    border-bottom-style: solid;  
}
```

### 5.2.2 奇特的链接下划线

通过使用图像创建链接下划线，可以产生非常有意思的效果。例如，创建一个非常简单的下划线图像，它由斜线组成（见图5-1）。



图5-1 简单的下划线图像

可以使用以下代码将这个图像应用于链接：

```
a:link, a:visited {  
    color:#666;  
    text-decoration: none;  
    background: url(/img/underline1.gif) repeat-x left bottom;  
}
```

在图5-2中可以看到这个链接样式的效果。

This is a link

图5-2 定制的链接下划线

这种方式并不限于link和visited样式。在下面的示例中，我为hover和active状态创建了一个动画GIF，然后使用以下CSS应用它：

```
a:hover, a:focus, a:active {  
    background-image: url(/img/underline1-hover.gif);  
}
```

当鼠标悬停在链接上或单击链接时，斜线从左到右滚动出现，这就产生了一种有意思的脉冲效果。并非所有浏览器都支持背景图像动画，但是不支持这个特性的浏览器常常会显示动画的第一帧，这能确保效果在老式浏览器中可以平稳退化。

请记住，使用动画要小心，因为它会对某些用户造成可访问性问题。如果有疑问，请查阅 [www.w3.org/TR/WAI-WEBCONTENT/](http://www.w3.org/TR/WAI-WEBCONTENT/) 上的 *Web Content Accessibility Guidelines (WCAG 1.0)*。

## 5.3 已访问链接的样式

设计人员和开发人员常常忘记处理已访问链接的样式，导致已访问的链接和未访问的链接采用同样的样式。但是，不同的已访问链接样式可以帮助用户，让他们知道哪些页面或站点他们已经访问过了，避免不必要的“回溯”操作。

通过在每个已访问链接的旁边添加一个复选框，就可以创建一种非常简单的已访问链接的样式：

```
a:visited {
    padding-right: 20px;
    background: url(/img/check.gif) no-repeat right middle;
}
```

5

## 5.4 为链接目标设置样式

除了链接到特定的文档之外，还可以使用包含片段标识符的链接链接到页面的特定部分。实现的方法是在`href`的末尾加一个#字符，然后加上要链接的元素的ID。这非常适合指向很长的评论页面中的某一评论。例如，假设希望链接到这个页面上的第三个评论：

```
<a href="http://example.com/story.htm#comment3">
    A great comment by Simon
</a>
```

在单击前面的链接时，就会转到相应的文档，而且页面向下滚动到`comment3`元素。但是，如果页面内容非常多，常常很难看出链接把你转到了哪个元素。为了解决这个问题，CSS 3允许使用`:target`伪类为目标元素设置样式。在下一个示例中，我要给目标元素设置黄色背景以突出显示它（见图5-3）。

```
.comment:target {
    background-color: yellow;
}
```

如果希望更清楚一些，可以给这个元素设置动画背景图像，图像从黄色逐渐褪色为白色，从而模拟37 Signals等公司常用的黄色褪色技术。

```
.comment:target {
    background-image: url(img/fade.gif);
}
```

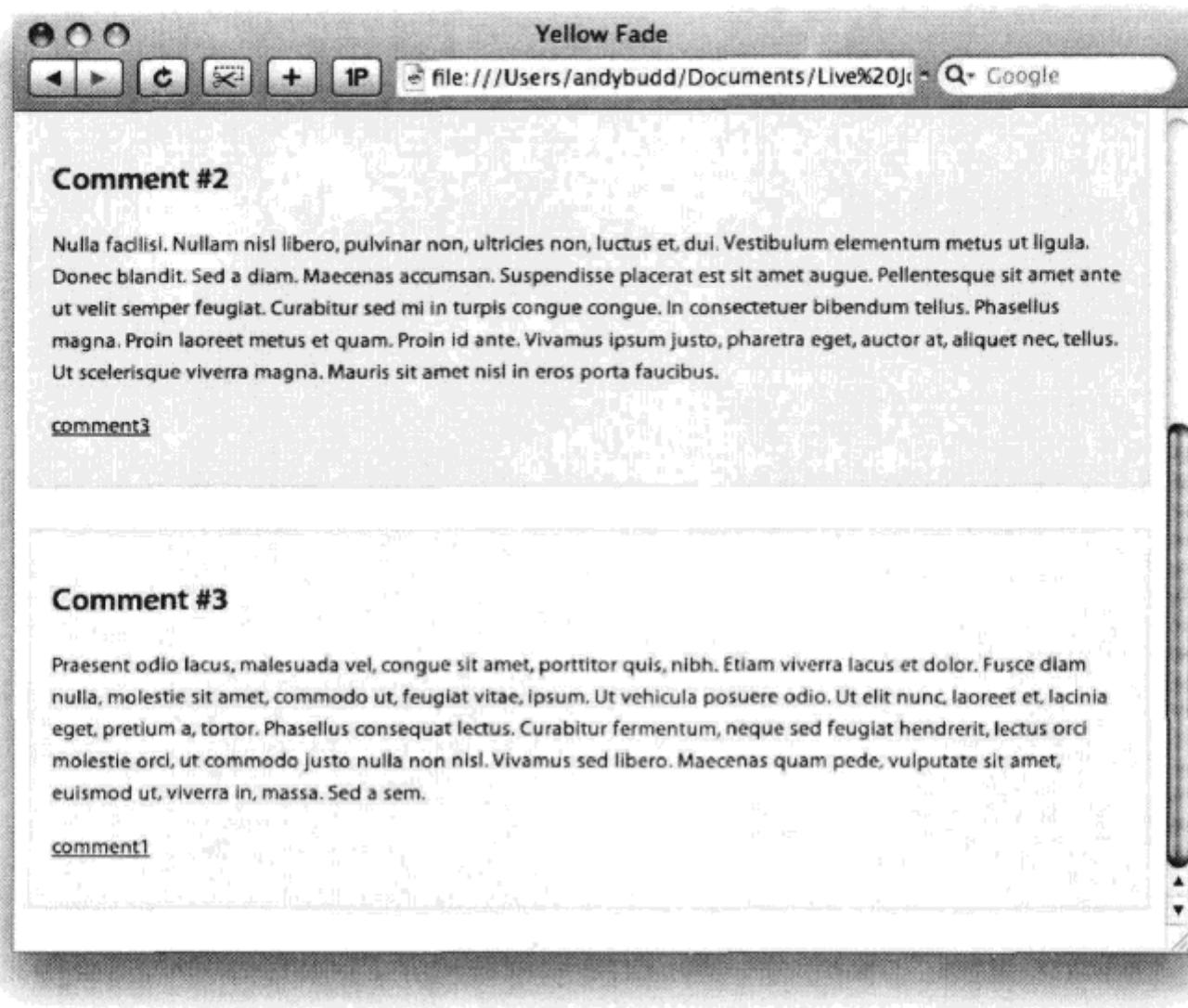


图5-3 使用:target选择器后，链接到的第三个评论以黄色背景突出显示

Safari和Firefox的所有近期版本都支持目标选择器，但是IE还不支持。

## 5.5 突出显示不同类型的链接

在许多站点上，很难看出链接是指向本站点上的另一个页面，还是指向另一个站点。我们曾经都有这样的经历：单击一个链接，期望浏览器转到当前站点上的另一个页面，却被带到了别处。为了解决这个问题，许多站点在新窗口中打开外部链接。但是，这不是好办法，因为它使用户失去了控制能力，而且这些多余的窗口可能会弄乱用户的桌面。如果没有通知用户出现了新窗口，这还会给屏幕阅读器用户造成问题。另外，新窗口实际上无法使用后退按钮，因为不可能返回到前一个屏幕。

较好的解决方案是让外部链接看起来不一样，让用户自己选择是离开当前站点，还是在新窗口或新的标签页中打开这个链接。为此，可以在外部链接旁边加一个小图标。wikipedia.com等站点就是这么做的，而且对于离站链接的图标已经出现了一种约定：一个框加一个箭头（见图5-4）。



图5-4 外部链接图标

在页面上包含外部链接最容易的方法是在所有外部链接上加一个类，然后将图标作为背景图像应用。在下面的示例中，给链接设置少量的右内边距，从而给图标留出空间，然后将图标作为背景图像应用于链接的右上角（见图5-5）。

```
.external {
    background: url(/img/externalLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

This is an external link 

图5-5 外部链接样式

尽管这个方法是有效的，但是它不太灵巧，也不够优雅，因为必须手工地在每个外部链接上添加类。有办法让CSS判断链接是否是外部链接吗？实际上，确实有办法，这就是使用属性选择器。

第1章中介绍过，属性选择器允许根据特定属性是否存在或属性值来寻找元素。CSS 3扩展了它的功能，提供了子字符串匹配。顾名思义，这些选择器允许通过对属性值的一部分和指定的文本进行匹配来寻找元素。CSS 3还没有成为正式的规范，所以使用这些高级选择器可能会使代码失效。但是，大多数符合标准的浏览器已经支持这些CSS 3选择器了。

这种技术的工作方式是使用`[att^=val]`属性选择器寻找以文本`http:`开头的所有链接：

```
a[href^="http:"] {
    background: url(/img/externalLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

这应该会突出显示所有外部链接。但是，它也会选中使用绝对URL而不是相对URL的内部链接。为了避免这个问题，需要重新设置指向自己站点的所有链接，删除它们的外部链接图标。方法是匹配指向自己域名的链接，删除外部链接图标，重新设置右内边距（见图5-6）。

```
a[href^="http://www.yoursite.com"],
a[href^="http://yoursite.com"] {
    background-image: none;
    padding-right: 0;
}
```

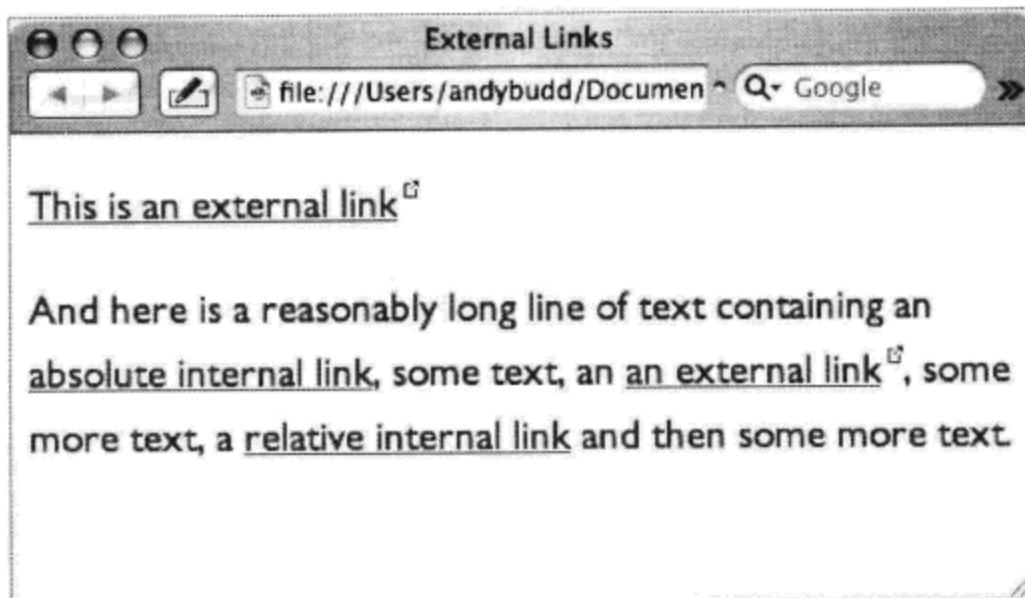


图5-6 这个页面对外部链接和内部链接使用不同的样式

大多数现代浏览器都支持这种技术，而老式浏览器（比如IE 6和更低版本）会忽略它。

如果愿意，还可以扩展这种技术，对邮件链接也进行突出显示。在下面的示例中，在所有[mailto](mailto:)链接上添加一个小的邮件图标：

```
a[href^="mailto:"] {
  background: url(img/email.png) no-repeat right top;
  padding-right: 10px;
}
```

甚至可以突出显示非标准的协议，比如用小的AIM图标突出显示AIM即时消息协议（见图5-7）：

```
a[href^="aim:"] {
  background: url(img/im.png) no-repeat right top;
  padding-right: 10px;
}

<a href="aim:goim?screenname=andybudd">instant message</a>
```

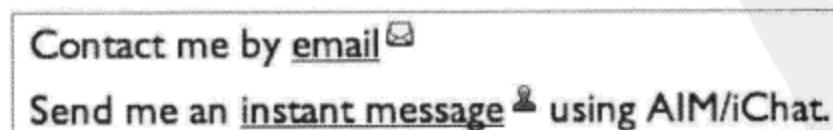


图5-7 邮件和即时消息链接样式

### 突出显示可下载的文档和提要

另一种烦人的常见情况是，单击一个链接，本以为会进入另一个页面，却发现网站开始下载一个PDF或Microsoft Word文档。好在CSS也可以帮助区分这些类型的链接。这要使用[att\$=val]属性选择器，它寻找以特定值（比如.pdf或.doc）结尾的属性：

```
a[href$=".pdf"] {
    background: url(img/pdfLink.gif) no-repeat right top;
    padding-right: 10px;
}

a[href$=".doc"] {
    background: url(img/wordLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

采用与前面示例相似的方式，可以用不同的图标突出显示Word文档和PDF，告诉访问者它们是下载链接而不是另一个页面的链接。

最后，许多人在自己的网站上提供了RSS提要（feed）。访问者可以将这些链接复制到提要阅读器中。但是，无意间单击这些链接就会进入一个页面，其中的数据似乎是没有意义的。为了避免可能发生的混淆，可以通过类似的方法用自己的RSS图标突出显示RSS 提要：

```
a[href$=".rss"], a[href$=".rdf"] {
    background: url(img/feedLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

所有这些技术都有助于改进用户在站点上的浏览体验。通过提醒用户注意离站链接或可下载的文档，让他们明确地了解在单击链接时会发生的情况，避免了不必要的回溯操作和烦恼。

5

很遗憾，IE 6 和更低的版本不支持属性选择器。好在，可以通过在每个元素中添加类，使用 JavaScript 和 DOM 实现相似的效果。最好的方法之一是使用 Simon Willison 编写的 `getElementsBySelector` 函数，在 <http://simonwillison.net/2003/Mar/25/getElementsBySelector/> 上可以找到更多细节。

另外，使用 jQuery 也可以做类似的事情。

## 5.6 创建类似按钮的链接

锚是行内元素，这意味着只有在单击链接的内容时它们才会激活。但是，有时候希望实现更像按钮的效果，有更大的可单击区域。为此，可以将锚的`display`属性设置为`block`，然后修改`width`、`height`和其他属性来创建需要的样式和单击区域。

```
a {
    display: block;
    width: 6.6em;
    line-height: 1.4;
    text-align: center;
```

```

text-decoration: none;
border: 1px solid #66a300;
background-color: #8ccca12;
color: #fff;
}

```

产生的链接应该像图5-8这样。



图5-8 与按钮相似的链接样式

由于链接现在显示为块级元素，单击块中的任何地方都会激活链接。

从这段CSS可以看到宽度是以em为单位显式设置的。由于其性质，块级元素会扩展，填满可用的宽度，所以如果它们父元素的宽度大于链接所需的宽度，那么需要将希望的宽度应用于链接。如果希望在页面的主内容区域中使用这种样式的链接，就很可能出现这种情况。但是，如果这种样式的链接出现在宽度比较窄的地方，如边栏中，那么可能只需设置边栏的宽度，而不需要为链接的宽度担心。

你可能想知道我为什么使用line-height来控制按钮的高度，而不是使用height。这实际上是一个小技巧，能够使按钮中的文本垂直居中。如果设置height，就必须使用内边距将文本压低，模拟出垂直居中的效果。但是，文本在行框中总是垂直居中的，所以如果使用line-height，文本就会出现在框的中间。但是，有一个缺点。如果按钮中的文本占据两行，按钮的高度就是需要的高度的两倍。避免这个问题的唯一方法是调整按钮和文本的尺寸，让文本不换行，至少在文本字号超过合理值之前不会换行。

如果选用这种技术，那么要确保元素是真正的链接，而不要更新服务器。否则，可能会出现意外的结果。在Google加速程序启动时，人们发现CMS或Web应用程序中的内容神秘地消失了。有时候，站点的全部内容在一夜之间就消失了。这是因为这些工具的开发人员使用锚链接而不是用表单元素作为删除按钮。Google加速程序会访问这些链接以便缓存它们，这样就会无意间删除内容！搜索引擎的spider可以造成相同的结果，递归地删除大量数据。因此，决不要使用链接更新服务器。或者用技术术语来说，链接应该只用于GET请求，决不要用于POST请求。

### 5.6.1 简单的翻转

在过去，人们使用庞大且过分复杂的JavaScript函数实现翻转效果。现在，使用: hover伪类就可以创建翻转效果，而不需要使用JavaScript。可以扩展前面的示例，在鼠标悬停时设置链接的背景和文本颜色，从而实现非常简单的翻转效果（见图5-9）。

```
a:hover,
```

```
a:focus {  
background-color: #f7a300;  
border-color: #ff7400;  
}
```



图5-9 显示激活区域的鼠标悬停样式

## 5.6.2 图像翻转

修改背景颜色对于简单的按钮很合适，但是对于比较复杂的按钮，你可能会想到使用背景图像。在下一个示例中，我创建了3个按钮图像，一个用于默认状态，一个用于鼠标悬停和焦点状态，一个用于激活状态（见图5-10）。



图5-10 正常状态、鼠标悬停状态和激活状态的图像

这个示例的代码与前面的示例相似，主要的差异是使用的是背景图像而不是背景颜色。

```
a:link, a:visited {  
display: block;  
width: 203px;  
height: 72px;  
text-indent: -1000em;  
background: url(/img/button.png) left top no-repeat;  
}  
  
a:hover, a:focus { background-image: url(/img/button-over.png);  
}  
  
a:active {  
background-image: url(/img/button-active.png);  
}
```

这个示例使用固定宽度和高度的按钮，所以我在CSS中设置显式的像素尺寸。为了实现我希望的文本效果，我把按钮文本放在图像上，然后使用大的负文本缩进隐藏锚文本。但是，也可以创建特大的按钮图像，或者结合使用背景颜色和图像创建流式的或弹性的按钮。

## 5.6.3 Pixy 样式的翻转

多图像方法的主要缺点是，在浏览器第一次加载鼠标悬停的图像时有短暂的延迟。这会造成

闪烁效果，让人感觉按钮有点儿反应迟钝。可以将鼠标悬停的图像作为背景应用于父元素，从而预先先载它们。但是，有另一种方法：不切换多个背景图像，而是使用一个图像并切换它的背景位置。使用单个图像的好处是减少了服务器请求的数量，而且可以将所有按钮状态放在一个地方。这个方法称为 Pixy 方法，Pixy 是它的发明者 Petr Staniček 的昵称。（你可以在他的网站 <http://wellstyled.com/css-nopreload-rollovers.html> 上获得更多信息。）

首先，创建组合的按钮图像（见图 5-11）。在这个示例中，只使用正常状态、鼠标悬停状态和激活状态，但是如果愿意，也可以使用已访问状态。

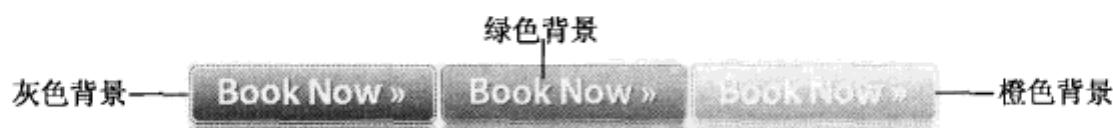


图 5-11 用一个图像表示这些按钮状态

代码几乎与前面的示例相同。但是，这一次背景图像的对齐方式是：正常状态下，背景图像在中间；鼠标悬停状态下，背景图像在右边；激活状态下，背景图像在左边。

```
a:link, a:visited {
    display: block;
    width: 203px;
    height: 72px;
    text-indent: -1000em;
    background: url(/img/buttons.png) -203px 0 no-repeat;
}

a:hover, a:focus {
    background-position: right top;
}

a:active {
    background-position: left top;
}
```

可惜，即使只是改变图像的对齐方式，Windows 上的 IE 仍然会向服务器请求新的图像。这会产生轻微的闪烁，有点儿烦人。为了避免闪烁，需要将翻转状态应用于链接的父元素，例如包含它的段落。

```
p {
    background: url(/img/ buttons.png) ←
    no-repeat right top;
}
```

在图像重新加载时，它仍然会消失一段时间。但是，现在会露出相同的图像，消除了闪烁。

消除闪烁的另一种方式是在 IE 专用的 CSS 文件中包含以下代码，这会启用背景缓存。

```
html {  
    filter: expression(document.execCommand("BackgroundImageCache", false, true));  
}
```

#### 5.6.4 CSS 精灵

多个服务器请求会对站点的性能产生显著的影响，所以Pixy方法把所有按钮状态包含在一个图像中，从而减少请求数量。但是，为什么要就此止步呢？为什么不更进一步，把所有图标甚至站点导航都包含在一个图像中？这样就可以把对多个图像的服务器调用数量减少到二三个。这就是CSS精灵——包含许多不同的图标、按钮或图形的单个图像。许多大型网站使用这种技术，包括Yahoo!的主页。实际上，我们的Clearleft站点导航也使用这种技术（见图5-12）。

```
#nav li a {  
    display: block;  
    text-indent: -9999px;  
    height: 119px;  
    width: 100px;  
    background-image: url('/img/nav.png');  
    background-repeat: no-repeat;  
}  
  
#nav li.home a,  
#nav li.home a:link,  
#nav li.home a:visited {  
    background-position: 0 0;  
}  
  
#nav li.home a:hover,  
#nav li.home a:focus,  
#nav li.home a:active {  
    background-position: 0 -119px;  
}  
  
#nav li.who-we-are a,  
#nav li.who-we-are a:link,  
#nav li.who-we-are a:visited {  
    background-position: -100px 0;  
}  
#nav li.who-we-are a:hover,  
#nav li.who-we-are a:focus,  
#nav li.who-we-are a:active {  
    background-position: -100px -119px;  
}
```



图5-12 Clearleft站点上使用的CSS精灵文件

使用这种技术可以减少Web浏览器发出的服务器请求，这会显著加快下载速度。另外，使用精灵把所有按钮、图标和各种图形集中在一个地方可以提高可维护性。所以这是一种非常好的方法。

### 5.6.5 用 CSS 3 实现翻转

CSS 3包含text-shadow、box-shadow和border-radius等属性，可以创建样式很丰富的按钮，而不需要使用任何图像。为了创建这种按钮，我首先复制第一个示例中的代码：

```
a {
    display: block;
    width: 6.6em;
    height: 1.4em;
    line-height: 1.4;
    text-align: center;
    text-decoration: none;
    border: 1px solid #66a300;
    background-color: #8ccca12;
    color: #fff;
}
```

接下来，添加曲线边框和投影。我还要让按钮文本有点儿投影（见图5-13）。

```
a {
    display: block;
    width: 6.6em;
```

```

height: 1.4em;
line-height: 1.4;
text-align: center;
text-decoration: none;
border: 1px solid #66a300;
-moz-border-radius: 6px;
-webkit-border-radius: 6px;
border-radius: 6px;
background-color: #8ccal2;
color: #fff;
text-shadow: 2px 2px 2px #66a300;
-moz-box-shadow: 2px 2px 2px #ccc;
-webkit-box-shadow: 2px 2px 2px #ccc;
box-shadow: 2px 2px 2px #ccc;
}

```



图5-13 只使用CSS实现的圆角按钮

为了实现渐变，Safari 4 beta版支持专有的值`-webkit-gradient`。尽管我一般不建议使用专有的代码，但是这可以展示CSS以后的发展。这个专有的值包含几个参数，包括渐变的类型（直线或放射状）、渐变的方向（在这里是左上到左下）、初始颜色、结束颜色和中间的过渡点。显然，如果不希望使用这种专有的代码，可以使用渐变的背景图像。

```

a {
display: block;
width: 6.6em;
height: 1.4em;
line-height: 1.4;
text-align: center;
text-decoration: none;
border: 1px solid #66a300;
-moz-border-radius: 6px;
-webkit-border-radius: 6px;
border-radius: 6px;
background-image: -webkit-gradient(linear, left top, left bottom, ↬
from(#abe142), to(#67a400));
background-color: #8ccal2;
color: #fff;
text-shadow: 2px 2px 2px #66a300;
-moz-box-shadow: 2px 2px 2px #ccc;
-webkit-box-shadow: 2px 2px 2px #ccc;
box-shadow: 2px 2px 2px #ccc;
}

```

最后，Safari还提供另一个称为`box-reflect`的专有属性，顾名思义，它可以创建对象的倒

影。这个属性包含多个参数，包括倒影的位置和距离以及蒙板图像。有意思的是，可以使用`-webkit-gradient`值自动地生成这个蒙板。在这里，我把倒影定位在按钮下面2像素，并且使用一个褪到白色的蒙板（见图5-14）。

```
a {  
display: block;  
width: 6.6em;  
height: 1.4em;  
line-height: 1.4;  
text-align: center;  
text-decoration: none;  
border: 1px solid #66a300;  
-moz-border-radius: 6px;  
-webkit-border-radius: 6px;  
border-radius: 6px;  
background-image: -webkit-gradient(linear, left top, left bottom, ←  
from(#abe142), to(#67a400));  
background-color: #8ccal2;  
color: #fff;  
text-shadow: 2px 2px 2px #66a300;  
-moz-box-shadow: 2px 2px 2px #ccc;  
-webkit-box-shadow: 2px 2px 2px #ccc;  
box-shadow: 2px 2px 2px #ccc;  
-webkit-box-reflect: below 2px -webkit-gradient ←  
(linear, left top, left bottom, from(transparent), ←  
color-stop(0.52, transparent), to(white));  
}
```

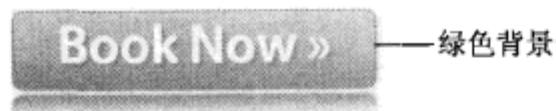


图5-14 使用Safari专有CSS扩展实现的圆角按钮

关于是否应该使用CSS实现这些效果还有争议，所以不确定它们是否会成为正式规范。由于这个原因以及缺少跨浏览器支持，所以不应该在生产环境中使用这些技术。但是，只要你知道它们是无效的CSS，在以后的浏览器中可能会取消或改变，那么完全可以在个人站点上试用它们。

## 5.7 纯CSS工具提示

工具提示是当鼠标悬停在具有`title`属性的元素上时一些浏览器弹出的黄色小文本框。一些开发人员结合使用JavaScript和CSS创建了样式独特的自定义工具提示。但是，通过使用CSS定位技术，可以创建纯CSS工具提示。这种技术需要符合标准的现代浏览器（比如Firefox）才能正确地工作。因此，它不是日常使用的技术。但是，它演示了高级CSS的能力，让你能够体会一下当CSS得到更好的支持之后会是什么情况。

与本书中的所有示例一样，需要先创建结构良好且有意义的HTML：

```
<p>
  <a href="http://www.andybudd.com/" class="tooltip">
    Andy Budd<span> (This website rocks) </span></a> is a web developer based in ↵
    Brighton England.
  </p>
```

这个链接设置的类名为`tooltip`，以便从其他链接中区分出来。在这个链接中，添加希望显示为链接文本的文本，然后添加包围在`span`中的工具提示文本。我将工具提示包围在圆括号中，这样的话在样式关闭时这个句子仍然是有意义的。

首先需要做的是将锚的`position`属性设置为`relative`。这样就可以相对于父元素的位置对`span`的内容进行绝对定位。因为不希望工具提示一开始就显示出来，所以应该将它的`display`属性设置为`none`：

```
a.tooltip {
  position: relative;
}

a.tooltip span {
  display: none;
}
```

当鼠标悬停在这个锚上时，表示希望显示`span`的内容。方法是将`span`的`display`属性设置为`block`，但是只在鼠标悬停在这个链接上时才这样做。如果现在测试此代码，当鼠标悬停在这个链接上时，链接的旁边会出现`span`文本。

为了让`span`的内容出现在锚的右下方，需要将`span`的`position`属性设置为`absolute`，并且将它定位到距离锚顶部`1em`，距离左边`2em`的地位。

```
a.tooltip:hover span {
  display: block;
  position: absolute;
  top: 1em;
  left: 2em;
}
```

请记住，绝对定位元素的定位相对于最近的已定位祖先元素（如果没有，就相对于根元素）。在这个示例中，已经定位了锚，所以`span`相对于锚进行定位。

这就是这种技术的主体部分。余下的工作是添加一些样式让`span`看起来像工具提示。可以给`span`加一些内边距、一个边框和背景颜色：

```
a.tooltip:hover span, a.tooltip:focus span {  
    display: block;  
    position: absolute;  
    top: 1em;  
    left: 2em;  
    padding: 0.2em 0.6em;  
    border: 1px solid #996633;  
    background-color: #FFFF66;  
    color: #000;  
}
```

在Firefox中查看，效果应该像图5-15这样。

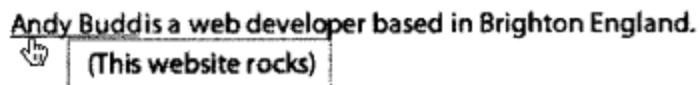


图5-15 纯CSS工具提示

## 5.8 小结

在本章中，你学习了如何以各种方式对链接应用样式，了解了如何根据链接到的站点或文件对链接应用样式。你可以让链接表现得像按钮一样，还可以使用颜色或图像创建翻转效果，甚至还可以创建纯CSS工具提示这样的高级效果。

下一章将介绍如何操纵列表，以及如何使用本章中学到的知识创建导航列表、纯CSS图像映射和远距离翻转。让我们开始吧！





## 第6章

# 对列表应用样式和创建导航条

人们天生喜欢对周围的世界进行组织排列。科学家创建了动物、植物和化学元素的列表；杂志热衷于评选10大电影、最新的时尚趋势或者装最差的名人；人们会写出采购列表、任务列表和圣诞礼物列表。我们太喜欢编写列表了。

6

列表让我们能够对相关的元素进行分组，并由此给它们添加意义和结构。大多数网页都包含某种形式的列表，比如最近的新闻列表、喜欢的网页的链接列表或到站点其他部分的链接列表。将这些条目标识为列表并且加上标记会在HTML文档中增加结构，提供应用样式的钩子（hook）。

在本章中，你将学习以下内容。

- 用CSS对列表应用样式。
- 使用背景图像作为项目符号。
- 创建垂直导航条和水平导航条。
- 使用滑动门标签页式导航。
- 纯CSS下拉菜单。
- 创建CSS图像映射。
- 创建远距离翻转。
- 使用定义列表。

## 6.1 基本列表样式

基本列表样式非常简单。假设有下面这个任务列表：

```
<ul>
  <li>Read emails</li>
```

```

<li>Write chapter</li>
<li>Go shopping</li>
<li>Cook dinner</li>
<li>Watch Lost</li>
</ul>

```

为了添加定制的项目符号，可以使用list-style-image属性。但是，这种方法对项目符号图像的位置的控制能力不强。更常用的方法是关闭项目符号，并且将定制的项目符号作为背景添加在列表元素上。然后可以使用背景图像的定位属性精确地控制自定义项目符号的对准方式。

IE的早期版本和Opera使用左外边距控制列表的缩进，而包括Safari和Firefox在内的大多数现代浏览器则选择使用左内边距。因此，首先需要将列表的外边距（margin）和内边距（padding）设置为零，从而去掉这个缩进。要去掉默认的项目符号，只需将列表样式类型设置为none：

```

ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
}

```

添加定制的项目符号非常简单。在列表项左边添加内边距，为符号留出所需的空间。然后将项目符号作为背景图像应用于列表项。如果列表项跨越多行，你可能希望将项目符号放在接近列表项顶部的位置。但是，如果知道列表项的内容不会跨越多行，那么可以将垂直位置设置为middle或50%，从而让项目符号垂直居中：

```

li {
  background: url(/img/bullet.gif) no-repeat 0 50%;
  padding-left: 30px;
}

```

产生的列表样式见图6-1。

- ✓ Read emails
- ✓ Write chapter
- ✓ Go shopping
- ✓ Cook dinner
- ✓ Watch Lost

图6-1 具有自定义项目符号的简单列表样式

## 6.2 创建基本的垂直导航条

在前面的示例中使用在第5章中学到的链接样式技术，就可以创建图形丰富的垂直导航条和

CSS翻转，如图6-2所示。



图6-2 垂直导航条

与平常一样，首先需要一个良好的语义标记：

```
<ul class="nav">
  <li><a href="home.htm">Home</a></li>
  <li><a href="about.htm">About</a></li>
  <li><a href="services.htm">Our Services</a></li>
  <li><a href="work.htm">Our Work</a></li>
  <li><a href="news.htm">News</a></li>
  <li><a href="contact.htm">Contact</a></li>
</ul>
```

6

首先要做的是去掉默认的项目符号并将外边距和内边距设置为零：

```
ul.nav {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
```

然后可以开始处理图形样式。在这里，我给导航菜单设置浅绿色的背景和深绿色的边框。还要以em为单位设置导航列表的宽度。

```
ul.nav {
  margin: 0;
  padding: 0;
  list-style-type: none;
  width: 8em;
  background-color: #8BD400;
  border: 1px solid #486B02;
}
```

不对列表项应用样式，而是对其中包含的锚链接应用样式，由此提供更好的浏览器兼容性。为了创建与按钮相似的单击区域，需要将锚的display属性设置为block。然后扩展锚链接，让

它占据可用空间（在这里，由列表的宽度决定）。可以显式地设置锚的宽度，但是我发现如果设置父列表的宽度，代码更容易维护。最后两个规则只是修饰性的，它们设置链接文本的颜色并关闭下划线。

```
ul.nav a {
    display: block;
    color: #2B3F00;
    text-decoration: none;
}
```

为了在菜单项上创建斜面效果，需要把顶边框设置得比背景颜色浅，让底边框深一点儿。现在，还可以设置一个背景图像作为图标。

```
ul.nav a {
    display: block;
    color: #2B3F00;
    text-decoration: none;
    border-top: 1px solid #E4FFD3;
    border-bottom: 1px solid #486B02;
    background: url(/img/arrow.gif) no-repeat 5% 50%;
    padding: 0.3em 1em;
}
```

在理想情况下，可以把箭头定位在距锚的左边缘 10 像素的地方。但是，CSS 规范不允许混合使用多种单位，所以我使用百分数。实际上，大多数浏览器接受混合单位，我认为规范在这方面错了。

所有边框叠在另一个链接的顶上，但是最后一个链接的底边框与列表的底边框形成了双线。在这里，我采用简单的方法，去掉列表的底边框。但是，这在某些情况下不可行，那么可以在第一个或最后一个列表项上添加类，这样就可以直接删除边框。以后还可以使用:last-child 伪类，但是目前浏览器支持还很有限。

```
ul.nav .last a {
    border-bottom: 0;
}
```

这个列表现在看起来像一个漂亮的垂直导航条了。为了完成这个效果，最后还需要应用: hover、: focus 和: selected 状态。为此，只需修改背景和文本颜色。还可以通过修改边框颜色创建按下的按钮效果。当鼠标悬停在锚链接上时，这些样式应用于锚链接。它们还应用于具有类selected的父列表项中的锚。

```
ul.nav a:hover,
ul.nav a:focus,
```

```
ul.nav .selected a {
  color: #E4FFD3;
  background-color: #6DA203;
}
```

在除Windows上的IE 6和更低版本之外的所有主流浏览器上，这种技术都是有效的。但是，IE 6在列表项上下添加了额外的空间，这令人费解。为了修复这个bug，需要将列表项上的display属性设置为inline：

```
ul.nav li {
  display: inline; /* :KLUDGE: Removes large gaps in IE/Win */
}
```

现在你看见了一个漂亮的垂直导航条以及翻转效果。

## 6.3 在导航条中突出显示当前页面

在前面的垂直导航条示例中，我使用一个类表示当前页面。对于在页面中嵌入导航的小站点，只需逐个页面地添加这个类。对于大型站点，导航很可能是动态建立的，在这种情况下可以在后端添加类。但是，对于主导航不改变的中等规模的站点，往往通过外部文件包含导航。在这些情况下，如果有办法突出显示当前页面，而不需要动态地在菜单中添加类，那不是很棒吗？是的，用CSS可以实现这种效果。

这个概念的工作方式是，在每个页面的主体元素中添加一个ID或类名，从而指出用户当前在哪个页面或部分中。然后，在导航列表中的每个项中添加一个对应的ID或类名。可以使用主体的ID和列表ID/类的唯一组合在站点导航中突出显示当前部分或页面。

以下面的HTML片段为例。当前页面是主页，这由主体上的ID home表示。主导航中的每个列表项都被分配了一个类名，这个类名根据与列表项相关联的页面的名称而定：

```
<body id="home">
<ul class="nav">
  <li><a href="home.htm">Home</a></li>
  <li><a href="about.htm">About</a></li>
  <li><a href="services.htm">Our Services</a></li>
  <li><a href="work.htm">Our Work</a></li>
  <li><a href="news.htm">News</a></li>
  <li><a href="contact.htm">Contact</a></li>
</ul>
</body>
```

为了突出显示当前页面，只需寻找以下ID和类名的组合：

```
#home .nav .home a,
#about .nav .about a,
```

```
#news .nav .news a,
#products .nav .products a,
#services .nav .services a {
    background-position: right bottom;
    color: #fff;
    cursor: default;
}
```

当用户在主页上时，具有`home`类的导航项将显示为被选择状态；在新闻页面上，具有`news`类的导航项将显示为被选择状态。为了增加效果，我将鼠标光标改为显示默认的箭头样式。这样的话，如果鼠标经过被选择的链接，那么光标不会改变状态，因此不会诱使用户单击当前页面的链接。

## 6.4 创建简单的水平导航条

假设有一个搜索结果页面，要创建一个基于页面的简单导航列表，比如像图6-3这样。为此，需要先创建导航项的有序列表。

```
<ol class="pagination">
<li><a href="search.htm?page=1" rel="prev">Prev</a></li>
<li><a href="search.htm?page=1">1</a></li>
<li class="selected">2</li>
<li><a href="search.htm?page=3">3</a></li>
<li><a href="search.htm?page=4">4</a></li>
<li><a href="search.htm?page=5">5</a></li>
<li><a href="search.htm?page=3" rel="next">Next</a></li>
</ol>
```



图6-3 水平搜索结果导航条

注意，我使用`rel`属性表示结果集中的前一个和下一个页面。这是一种非常好的`rel`属性使用方法，在以后给链接设置不同的样式时很方便。

与本章前面的其他列表示例一样，首先需要去掉默认的浏览器外边距、内边距和列表样式。包括我在内的许多开发人员喜欢在样式表的开头使用全局`reset`来完成这个任务。如果你使用全局`reset`，可以跳过这个步骤。

```
ol.pagination {
    margin: 0;
    padding: 0;
    list-style-type: none;
}
```

为了让列表项水平排列而不是垂直排列，可以把display属性设置为inline。但是，对于比较复杂的水平列表样式，如果浮动列表项，然后使用外边距把它们分开，就会更灵活。

```
ol.pagination li {  
    float: left;  
    margin-right: 0.6em;  
}
```

现在，列表项都水平显示了，可以开始应用图形样式了。在这里，我希望所有页码都出现在灰色背景的方框中。当用户把鼠标悬停在这些链接上时，我希望背景变成蓝色，链接文本变成白色。

```
ol.pagination a,  
ol.pagination li.selected {  
    display: block;  
    padding: 0.2em 0.5em;  
    border: 1px solid #ccc;  
    text-decoration: none;  
}  
ol.pagination a:hover,  
ol.pagination a:focus,  
ol.pagination li.selected {  
    background-color: blue;  
    color: white;  
}
```

这对于页码非常合适，但是我希望prev和next链接的样式稍有不同。为此，我使用属性选择器寻找它们的rel属性。我不希望前一个和下一个链接有边框，所以去掉边框。

```
ol.pagination a[rel="prev"],  
ol.pagination a[rel="next"] {  
    border: none;  
}
```

另外，我还希望在列表的开头和末尾添加箭头。可以在HTML中直接编写它们，但是，也可以使用CSS注入它们，这样便于以后修改或删除它们。这需要使用:before和:after伪选择器以及content属性。

```
ol.pagination a[rel="prev"]::before {  
    content: "\00AB";  
    padding-right: 0.5em;  
}  
  
ol.pagination a[rel="next"]::after {  
    content: "\00BB";  
    padding-left: 0.5em;  
}
```

第一个声明应用于列表开头的锚链接，它使用字符编码“00AB”在这个链接前面添加双左箭头。第二个声明应用于最后一个锚链接，它在这个链接后面添加双右箭头。

你现在看见了一个简单但灵活的水平页面导航条。

## 6.5 创建图形化导航条

简单的导航条很适合分页的内容，但是你可能希望为主导航创建图形更丰富的菜单。在这个示例中，我要演示如何创建图6-4这样的图形化导航条。



图6-4 水平导航条

与前面的示例一样，先建立一个简单的无序列表：

```
<ul class="nav">
    <li><a href="home.htm">Home</a></li>
    <li><a href="about.htm">About</a></li>
    <li><a href="news.htm">News</a></li>
    <li><a href="products.htm">Products</a></li>
    <li><a href="services.htm">Services</a></li>
    <li><a href="clients.htm">Clients</a></li>
    <li><a href="case-studies.htm">Case Studies</a></li>
</ul>
```

然后将padding和margin设置为零，并且去掉默认的项目符号。对于这个示例，我想让水平导航条的宽度为72 em，并且以重复的桔红色渐变作为背景。

```
ul. nav {
    margin: 0;
    padding: 0;
    list-style: none;
    width: 72em;
    background: #FAA819 url(img/mainNavBg.gif) repeat-x;
}
```

这个列表当前是垂直显示的。为了让它水平显示，让列表项向左浮动。

```
ul. nav li {
    float: left;
}
```

请记住，当元素浮动时，它不再占据文档流中的任何空间。因此，父列表实际上没有内容，它就会收缩，从而会隐藏列表背景。正如在第3章中学到的，有几种方法可以让父元素包含浮动的子元素。一种方法是添加一个进行清理的元素。但这会在页面中增加不必要的标记，所以应该

尽可能避免。另一种方法是让父元素浮动，并且使用某个元素（比如站点页脚）对它进行清理以便换行。第三种方法是使用`overflow:hidden`技术，我通常使用这种方法：

```
ul.nav {
    margin: 0;
    padding: 0;
    list-style: none;
    width: 72em;
    overflow: hidden;
    background: #FAA819 url(img/mainNavBg.gif) repeat-x;
}
```

与页面导航条示例一样，这个水平导航条中每个链接的`display`属性也设置为`block`，从而让它们表现得像按钮一样。如果希望每个按钮具有固定的尺寸，那么可以显式地设置它的高度和宽度。但是，这会导致可维护性问题。因此我希望每个按钮的宽度由锚文本的尺寸决定。为此，不设置显式的宽度，而是在每个锚链接的左边和右边应用`3em`的内边距。与前面的示例一样，使用行高让链接文本垂直居中。最后，关闭链接下划线并且将链接颜色改为白色：

```
ul.nav a {
    display: block;
    padding: 0 3em;
    line-height: 2.1em;
    text-decoration: none;
    color: #fff;
}
```

我希望在导航条中的每个链接之间创建分隔线。可以通过在列表项或锚上设置水平边框来完成。但是，为了简单，我将在锚链接中应用一个背景图像。

```
ul.nav a {
    display: block;
    padding: 0 2em;
    line-height: 2.1em;
    background: url(img/divider.gif) repeat-y left top;
    text-decoration: none;
    color: #fff;
}
```

但是，导航条中的第一个链接会有不必要的分隔线。在第一个列表项上添加一个类并且将背景图像设置为`none`，就可以去掉它：

```
ul.nav .first a {
    background-image: none;
}
```

另外，如果你不太在乎IE 6的支持问题，也可以不用其他类，而是使用`:first-child`伪类。

```
ul.nav li:first-child a {
    background: none;
}
```

最后，这个示例中的翻转状态仅仅是改变链接颜色：

```
ul.nav a:hover,
ul.nav a:focus {
    color: #333;
}
```

你应该看见了一个样式漂亮的水平导航条，具有良好的跨浏览器支持。

## 6.6 简化的“滑动门”标签页式导航

第4章学习了Douglas Bowman的滑动门技术，以及如何使用它创建灵活的圆角框。这种技术也可以用来创建灵活的可扩展的标签页式导航。使用这个方法时，用一个大图像和一个侧边图像创建标签页。随着标签页中文本的扩展，大图像的更多部分露出来。较小的图像留在左边，盖住大图像的硬边缘，实现图6-5所示的效果。

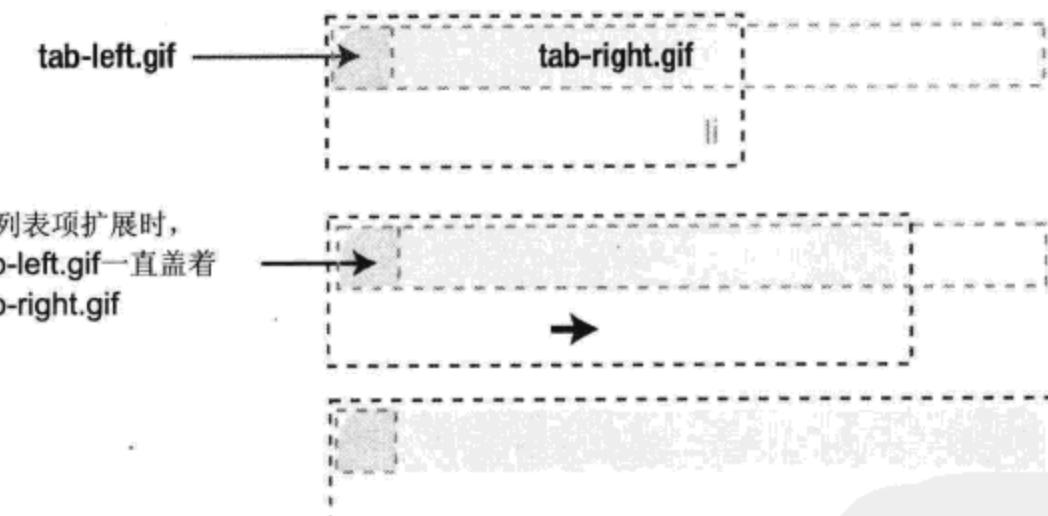


图6-5 “滑动门”技术的示例

在以下示例中用来创建标签页的图像（见图6-6）。这两个图像都非常大，这允许字号增加几倍，标签页也不会断裂。

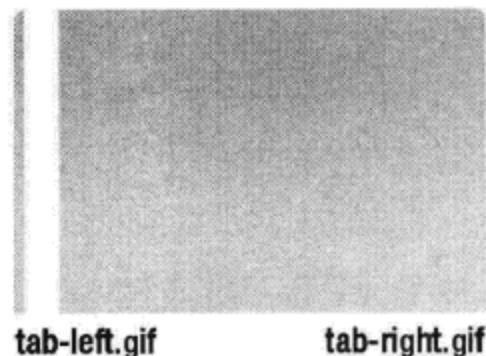


图6-6 组成标签页的两个图像

这个示例的HTML与前面的水平导航条示例完全一样：

```
<ul class="nav">
<li><a href="home.htm">Home</a></li>
<li><a href="about.htm">About</a></li>
<li><a href="news.htm">News</a></li>
<li><a href="products.htm">Products</a></li>
<li><a href="services.htm">Services</a></li>
<li><a href="clients.htm">Clients</a></li>
<li><a href="case-studies.htm">Case Studies</a></li>
</ul>
```

与前面的示例一样，将margin和padding设置为零，去掉项目符号，并且设置导航条的宽度，还把导航列表的overflow设置为hidden以便清理内部的浮动元素：

```
ul.nav {
    margin: 0;
    padding: 0;
    list-style: none;
    width: 72em;
    overflow: hidden;
}
```

与前面的示例一样，列表元素向左浮动，从而让它们水平显示而不是垂直显示。但是这一次，将组成标签页的两个图像中比较大的图像作为背景图像应用于列表项。由于这个图像形成标签页的右边缘，所以将它定位到右边：

```
ul.nav li {
    float: left;
    background: url(img/tab-right.gif) no-repeat right top;
}
```

与前面的示例一样，锚显示为块级元素以使整个区域可单击。每个标签页的宽度由内容的宽度控制，设置行高可以控制高度。为了完成标签页效果，将标签页的左边图像作为背景应用于锚并且左对齐。当标签页改变尺寸时，这个图像总是对准左边，盖在大图像上面，盖住左边的硬边缘。最后，为了确保这种技术在Mac上的IE 5.2中有效，也让锚浮动。

```
ul.nav li a {
    display: block;
    padding: 0 2em;
    line-height: 2.5em;
    background: url(img/tab-left.gif) no-repeat left top;
    text-decoration: none;
    color: #fff;
    float: left;
}
```

为了创建翻转效果，只需改变链接颜色：

```
ul.nav a:hover,
ul.nav a:focus {
    color: #333;
}
```

产生的标签页式导航应该像图6-7这样。



图6-7 正常文本字号的“滑动门”标签页式导航

如果在浏览器中加大文本字号，那么应该会看到标签页扩展了，见图6-8。

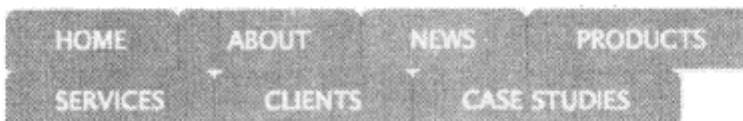


图6-8 几次增加文本字号之后的“滑动门”标签页式导航

用这个方法可以轻松地创建有吸引力的容易访问的标签页式导航条。

## 6.7 Suckerfish 下拉菜单

尽管在易用性方面有一些问题，但是下拉菜单仍然是Web上流行的界面元素。有一些纯JavaScript解决方案，但是其中许多都有先天的可访问性问题——即在禁用JavaScript的浏览器中无效。因此，一些开发人员研究了纯CSS下拉菜单。其中之一是Patrick Griffiths的Suckerfish下拉菜单技术（<http://www.alistapart.com/articles/dropdowns/>）。

这种技术极其简单，只需把子导航嵌套在无序列表中，把列表定位到屏幕之外，然后当鼠标悬停在父列表项上时重新定位它。在图6-9中可以看到最终效果。

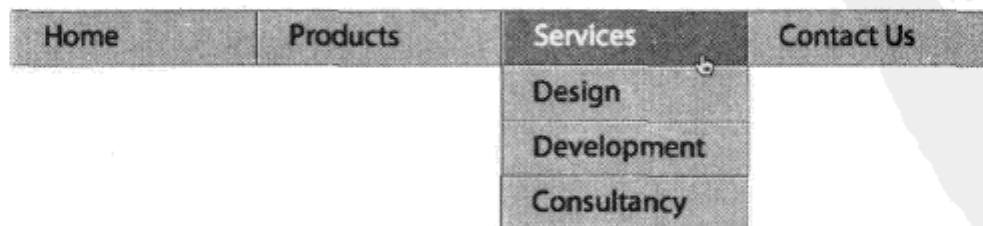


图6-9 纯CSS Suckerfish下拉菜单的效果

首先，建立多层导航列表。

```
<ul class="nav">
<li><a href="/home/">Home</a></li>
```

```

<li><a href="/products/">Products</a>
  <ul>
    <li><a href="/products/silverback/">Silverback</a></li>
    <li><a href="/products/fontdeck/">Font Deck</a></li>
  </ul>
</li>

<li><a href="/services/">Services</a>
  <ul>
    <li><a href="/services/design/">Design</a></li>
    <li><a href="/services/development/">Development</a></li>
    <li><a href="/services/consultancy/">Consultancy</a></li>
  </ul>
</li>

<li><a href="/contact/">Contact Us</a></li>
</ul>

```

与本章中的所有导航示例一样，首先需要将外边距和内边距设置为零，并且去掉默认的项目符号。因为这应该是一个水平导航，需要给列表项设置宽度并让它们向左浮动。出于格式上的考虑，我希望给导航列表设置边框和背景颜色。但是，因为其中包含的列表项都是浮动的，它们不占据空间，这会导致列表收缩。为了解决这个问题，我也让列表浮动。

```

ul.nav, ul.nav ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
  float: left;
  border: 1px solid #486B02;
  background-color: #8BD400;
}

ul.nav li {
  float: left;
  width: 8em;
  background-color: #8BD400;
}

```

为了确保下拉菜单中的菜单项垂直对齐，需要把列表的宽度设置为与列表项相同。下拉菜单现在初步成型了。

为了在激活之前隐藏实际的下拉菜单，需要把它们的position设置为absolute，然后把它们隐藏到屏幕左边之外。

```

ul.nav li ul {
  width: 8em;
  position: absolute;
  left: -999em;
}

```

现在到了关键之处。在父列表项中添加鼠标悬停伪选择器，把下拉菜单的位置改回正常位置，这样下拉菜单就会重新出现。

```
.nav li:hover ul {  
    left: auto;  
}
```

最后几个样式把导航链接设置为块级元素，然后修改列表的外观，设置背景颜色和斜面边框。

```
ul.nav a {  
    display: block;  
    color: #2B3F00;  
    text-decoration: none;  
    padding: 0.3em 1em;  
    border-right: 1px solid #486B02;  
    border-left: 1px solid #E4FFD3;  
}  
  
ul.nav li li a {  
    border-top: 1px solid #E4FFD3;  
    border-bottom: 1px solid #486B02;  
    border-left: 0;  
    border-right: 0;  
}  
  
/*remove unwanted borders on the end list items*/  
ul.nav li:last-child a {  
    border-right: 0;  
    border-bottom: 0;  
}  
  
ul a:hover,  
ul a:focus {  
    color: #E4FFD3;  
    background-color: #6DA203;  
}
```

这就是想要的结果：一个简单的只使用CSS的下拉导航条。这种技术适用于大多数现代浏览器，但是在IE的老版本中无效，因为它们不支持在非锚元素上使用: hover伪类。为了解决这个问题，可以使用几行JavaScript或.htc行为文件启用这个功能。

在IE中纠正下拉导航的JavaScript代码不属于本书内容，读者可以在<http://htmldog.com/articles/suckerfish/dropdowns/>找到更多信息。

## 6.8 CSS 图像映射

利用图像映射可以将图像的一些区域指定为热点。图像映射在几年前非常流行，但是近来不太常见了。部分原因是Flash流行起来了，还有部分原因是发展出了更简单、表现性更低的标记。

虽然图像映射仍然是HTML的有效部分，但是它们将表现方式与内容混在了一起。但是，可以结合使用列表、锚和一些高级CSS创建出简单的图像映射。

对于本节的示例，我使用了Clearleft团队的一些成员在办公室外的涂鸦墙前模仿独立乐队拍的一张照片（见图6-10）。当鼠标悬停在每个人身上时，我希望出现一个矩形框。单击这个框就会进入这个人的网站。



6

图6-10 Rich、Sophie、Cath、James和Paul在办公室外的涂鸦墙前模仿乐队成员

首先需要做的是将图像添加到页面中，放在一个已命名的div中：

```
<div class="imagemap">
  
</div>
```

然后，在图像后面添加每个人的网站链接的列表。需要给每个列表项分配一个类以便标识列表项中的人。还可以给每个链接设置title属性，其中包含这个人的名字。这样的话，当鼠标悬停在链接上时，在大多数浏览器上显示的工具提示中会显示人名。

```
<div id="imagemap">
  
<ul>
  <li class="rich">
    <a href="http://www.clagnut.com/" title="Richard Rutter"> ↵
      Richard Rutter</a>
  </li>
  <li class="sophie">
    <a href="http://www.wellieswithwings.org/" title="Sophie Barrett"> ↵
      Sophie Barrett</a>
  </li>
  <li class="cath">
    <a href="http://www.electricelephant.com/" title="Cathy Jones"> ↵
      Cathy Jones</a>
  </li>
  <li class="james">
    <a href="http://www.jeckecko.net/blog/" title="James Box"> ↵
      James Box</a>
  </li>
  <li class="paul">
    <a href="http://twitter.com/nicepaul" title="Paul Annett"> ↵
      Paul Annett</a>
  </li>
</ul>
</div>
```

设置div的宽度和高度，让它匹配图像的尺寸。然后，将div的position属性设置为relative。最后这一步是这种技术的关键，因为它让包含的链接可以相对于div（也就是图像）的边缘进行绝对定位。

```
.imagemap {
  width: 333px;
  height: 500px;
  position: relative; /* The key to this technique */
}
```

因为不希望显示项目符号，所以通过将list-style属性设置为none来去掉它们。为了保持完整性，还可能需要将列表的margin和padding设置为零：

```
.imagemap ul {
  margin: 0;
  padding: 0;
```

```
list-style: none;
}
```

下一件事情是对链接应用样式。对锚链接进行绝对定位，它们都将移动到容器div左上角。然后可以将它们分别定位到相应的人身上，形成热点。但是，首先需要设置它们的宽度和高度，从而创建需要的单击区域。链接文本仍然显示，因此，需要使用一个大的负数作为文本缩进量，从而让它们从屏幕上消失：

```
.imagemap a {
  position: absolute;
  display: block;
  width: 50px;
  height: 60px;
  text-indent: -1000em;
}
```

现在可以将各个链接定位在相应的人身上：

```
.imagemap .rich a {
  top: 50px;
  left: 80px;
}

.imagemap .sophie a {
  top: 90px;
  left: 200px;
}

.imagemap .cath a {
  top: 140px;
  left: 55px;
}

.imagemap .james a {
  top: 140px;
  left: 145px;
}
```

```
.imagemap .paul a {
  top: 165px;
  left: 245px;
}
```

最后，为了创建翻转效果，将一个白色的实线边框应用于鼠标悬停时的链接：

```
.imagemap a:hover,
imagemap a:focus {
  border: 1px solid #fff;
}
```

基本的技术就是这样。如果将鼠标悬停在一个图片上，那么应该会看到与图6-11相似的情况。



图6-11 翻转状态下的CSS图像映射

当然，这里假设你使用的是Safari或Firefox等功能比较强的浏览器。如果使用IE，就什么东西也看不到！看起来IE不显示其内容隐藏在屏幕之外的链接，即使显式地设置宽度和高度也不行。不过幸好在撰写本书的第一版之后，我发现了一种解决方法。

如果给锚链接设置某种背景，就可以让IE的表现正常。唯一的问题是我们实际上不希望链接有背景，因为它们应该是隐藏的！可以尝试把背景设置为透明的，但是这不管用！那么为什么不使用透明的图像（比如透明的PNG或GIF）？

```
.imagemap a {  
    position: absolute;  
    display: block;
```

```

background-image: url(/img/shim.gif);
width: 60px;
height: 80px;
text-indent: -1000em;
}

```

很奇怪的是，实际上不必指向真实的图像！只需指定一个不存在的URL，这就可以让IE的表现正常。但是，链接到不存在的URL感觉很怪异，即使是为了纠正浏览器的bug，所以我仍然使用真实但多余的图像。

## flickr 风格的图像映射

如果你使用过照片共享服务flickr，那么可能见过对图像进行说明的相似技术（见图6-12）。当翻转加了说明的图像时，在包含每个说明的区域上出现一个双边框的框。当鼠标悬停在这些框上时，它将突出显示并且显示说明。只需稍加修改，就可以使用前面的技术实现这种效果。

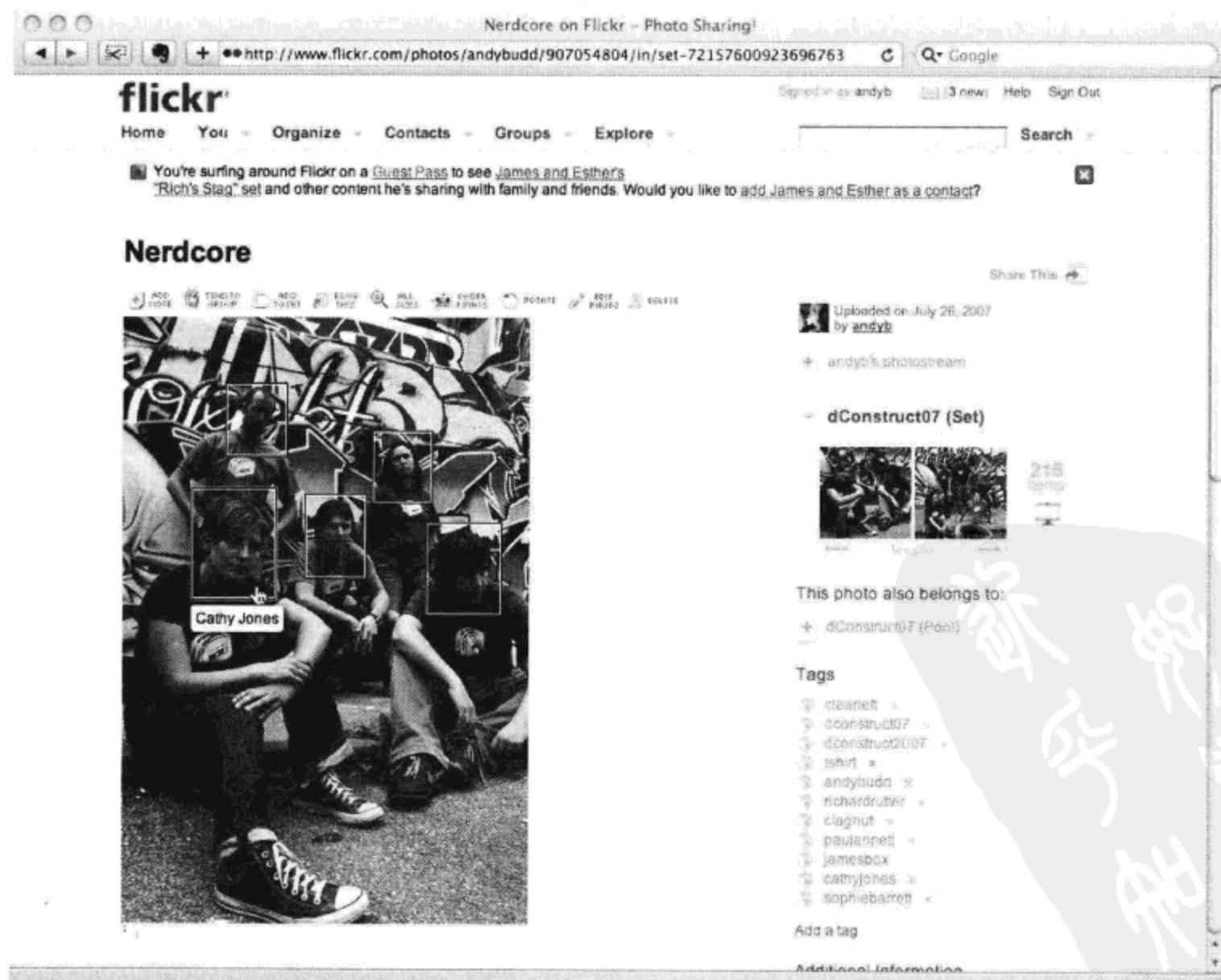


图6-12 flickr上的图像说明

为了创建双边框的框，需要在每个锚链接内部添加两个额外的。说明也需要添加一个额外的。添加这些额外的之后，列表应该像下面这样：

```
<ul>
  <li class="rich">
    <a href="http://www.clagnut.com/">
      <span class="outer">
        <span class="inner">
          <span class="note">Richard Rutter</span>
        </span>
      </span>
    </a>
  </li>
  ...
</ul>
```

CSS的开头与前面的示例相同，将容器div的尺寸设置为图像尺寸，将position属性设置为relative。将列表的margin和padding设置为零并且去掉项目符号：

```
.imagemap {
  width: 333px;
  height: 500px;
  position: relative;
}

.imagemap ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

与前面一样，包含的锚链接进行绝对定位。但是，这一次我要设置内部span的尺寸，让外部span和锚链接根据它们确定尺寸。我给外部span设置深色边框，给内部span设置浅色边框，从而突出显示它们在图像上的位置。最后，我不希望隐藏锚链接中的文本，而是想把它显示为工具提示，因此我给文本设置一些基本样式：

```
.imagemap a {
  position: absolute;
  display: block;
  background-image: url(/img/shim.gif);
  color: #000;
  text-decoration: none;
  border: 1px solid transparent;
}

.imagemap a .outer {
  display: block;
```

```
border: 1px solid #000;  
}  
  
.imagemap a .inner {  
    display: block;  
    width: 50px;  
    height: 60px;  
    border: 1px solid #fff;  
}
```

与前面一样，需要将锚定位到每个人身上：

```
.imagemap .rich a {  
    top: 50px;  
    left: 80px;  
}  
  
.imagemap .sophie a {  
    top: 90px;  
    left: 200px;  
}  
  
.imagemap .cath a {  
    top: 140px;  
    left: 55px;  
}  
  
.imagemap .james a {  
    top: 140px;  
    left: 145px;  
}  
  
.imagemap .paul a {  
    top: 165px;  
    left: 245px;  
}
```

6

然后对锚链接应用翻转效果。实现的方法是在鼠标悬停并获得焦点时将锚的边框颜色从透明改为黄色：

```
.imagemap a:hover,  
.imagemap a:focus {  
    border-color: #d4d82d;  
}
```

为了在热点翻转时显示说明，首先需要将说明的内容定位到热点的下面。为此，将说明的位置设置为absolute并且将底部位置设置为负值。为了让说明部分更美观，设置宽度、一些内边距和背景颜色，然后让文本居中：

```
.imagemap a .note {  
    position: absolute;  
    bottom: -3em;  
    width: 7em;  
    padding: 0.2em 0.5em;  
    background-color:#ffc;  
    text-align: center;  
}
```

如果在浏览器中查看这个页面，它应该像图6-13这样。



图6-13 flickr风格的翻转已经初步成型了

可以看出，效果已经初步成型了。说明部分看起来不错，但是如果它们在热点下面水平居中，而不是对准左边，就更好了。实现的方法是将说明的左边缘定位到热点的中点；接下来，使用负的外边距值将说明向左移动说明部分宽度的一半。这个示例中的热点有50像素宽，所以将说明的左边设置为50像素。说明部分有8em宽（包括内边距），所以设置4em的负的左外边距就会使说明在热点下面水平居中。

```
.imagemap a .note {
    position: absolute;
    bottom: -3em;
    width: 7em;
    padding: 0.2em 0.5em;
    background-color:#ffc;
    text-align: center;
    left: 25px;
    margin-left: -4em;
}
```

现在说明已经居中了，应该处理它们的交互性了。在默认情况下，说明应该隐藏起来，只在鼠标悬停在热点上时才显示。为此，可以将display属性设置为none，然后在鼠标悬停在锚链接上时将它改为block。但是，这会使某些屏幕阅读器无法访问说明的内容。所以我不这样做，而是将文本隐藏到屏幕左边之外，并且在鼠标悬停时对它重新定位：

```
.imagemap a .note {
    position: absolute;
    bottom: -3em;
    width: 7em;
    padding: 0.2em 0.5em;
    background-color:#ffc;
    text-align: center;
    left: -1000em;
    margin-left: -5em;
}

.imagemap a:hover .note,
.imagemap a:focus .note {
    left: 25px;
}
```

现在差不多完成了，只需要再做两处调整。如果不是一直显示热点的双边框，而是只在图像翻转时显示边框，那就更好了。这样的话，人们就可以不受热点的干扰，正常地欣赏图像。但是，当鼠标悬停在图像上时显示热点，就能让访问者知道还有更多的信息。实现的方法是在默认情况下把内外的边框设置为透明的，当鼠标悬停在图像上时设置它们的颜色：

```
.imagemap a .outer {
    display: block;
```

```
border: 1px solid transparent;
}

.imagemap a .inner {
  display: block;
  width: 50px;
  height: 60px;
  border: 1px solid transparent;
}

.imagemap:hover a .outer,
.imagemap:focus a .outer {
  border-color: #000;
}

.imagemap:hover a .inner,
.imagemap:focus a .inner {
  border-color: #fff;
}
```

可惜，IE 6只支持锚链接上的鼠标悬停。为了解决这个问题，当鼠标直接悬停在热点上时显示边框也是个好主意：

```
.imagemap:hover a .outer,
.imagemap:focus a .outer,
.imagemap a:hover .outer,
.imagemap a:focus .outer {
  border: 1px solid #000;
}

.imagemap:hover a .inner,
.imagemap:focus a .inner,
.imagemap a:hover .inner,
.imagemap a:focus .inner {
  border: 1px solid #fff;
}
```

现在你看见了一个flickr风格的高级CSS图像映射（见图6-14）。

## 6.9 远距离翻转

远距离翻转是一种鼠标悬停事件，它在页面的其他地方触发显示方式的改变。实现的方法是：在锚链接内嵌套一个或多个元素；然后，使用绝对定位对嵌套的元素分别定位。尽管显示在不同的地方，但是它们都包含在同一个父锚中，所以可以对同一个鼠标悬停事件做出反应。因此，当鼠标悬停在一个元素上时，可以影响另一个元素的样式。



图6-14 flickr风格的图像映射的最终效果

在这个示例中，我们将扩展基本的CSS图像映射技术，在图像下面放一个链接列表。当鼠标悬停在链接上时，图像热点将突出显示。同样，当鼠标悬停在图像上的热点区域时，文本链接也将突出显示。

这个示例的HTML与基本CSS图像映射示例相似。但是，需要添加两个：一个包围链接文本，一个空的作为热点。由此，链接文本可以定位在图像下面，将热点放在对应的人身上。

```
<div class="remote">
  
```

```
<ul>

    <li class="rich">
        <a href="http://www.clagnut.com/" title="Richard Rutter">
            <span class="hotspot"></span>
            <span class="link">&raquo; Richard Rutter</span>
        </a>
    </li>

    <li class="sophie">
        <a href="http://www.wellieswithwings.org/" title="Sophie Barrett">
            <span class="hotspot"></span>
            <span class="link">&raquo; Sophie Barrett</span>
        </a>
    </li>

    <li class="cath">
        <a href="http://www.electricelephant.com/" title="Cathy Jones">
            <span class="hotspot"></span>
            <span class="link">&raquo; Cathy Jones</span>
        </a>
    </li>

    <li class="james">
        <a href="http://www.jeckecko.net/blog/" title="James Box">
            <span class="hotspot"></span>
            <span class="link">&raquo; James Box</span>
        </a>
    </li>

    <li class="paul">
        <a href="http://twitter.com/nicepaul" title="Paul Annett">
            <span class="hotspot"></span>
            <span class="link">&raquo; Paul Annett</span>
        </a>
    </li>

</ul>
</div>
```

基本的列表样式与图像映射示例相同：

```
.remote {
    width: 333px;
    height: 500px;
    position: relative;
}

.remote ul {
    margin: 0;
```

```
padding: 0;  
list-style: none;  
}
```

首先需要做的是将热点的position属性设置为absolute，然后指定它们的尺寸。在这个示例中，3个热点的尺寸相同，另外两个大一点儿。因此，我先定义默认尺寸，然后在需要的地方覆盖它们。与前一种技术一样，将所有锚定位到图像的左上角。然后使用top和left定位属性，将每个热点定位到图像中相关的人身上。

```
.remote a .hotspot {  
    width: 50px;  
    height: 60px;  
    position: absolute;  
}  
  
.remote .rich a .hotspot {  
    top: 50px;  
    left: 80px;  
}  
  
.remote .sophie a .hotspot {  
    top: 90px;  
    left: 200px;  
}  
  
.remote .cath a .hotspot {  
    top: 140px;  
    left: 55px;  
    width: 60px;  
    height: 80px;  
}  
  
.remote .james a .hotspot {  
    top: 140px;  
    left: 145px;  
}  
  
.remote .paul a .hotspot {  
    top: 165px;  
    left: 245px;  
    width: 60px;  
    height: 80px;  
}
```

同样，包含链接文本的span也进行绝对定位，并且将宽度设置为15em。它们的定位也是相对于包含它们的列表的，在这个示例中，使用负的右位置值定位到图像的右边。最后，给链接设置光标样式，确保在IE中显示正确的图标。

```
.remote a .link {  
    position: absolute;  
    display: block;  
    width: 10em;  
    right: -11em;  
    cursor: pointer;  
}  
  
.remote .rich a .link {  
    top: 0;  
}  
  
.remote .sophie a .link {  
    top: 1.2em;  
}  
  
.remote .cath a .link {  
    top: 2.4em;  
}  
  
.remote .james a .link {  
    top: 3.6em;  
}  
  
.remote .paul a .link {  
    top: 4.8em;  
}
```

热点和链接文本现在应该会出现在正确的位置上。

为了在鼠标悬停在热点或文本上时在热点上实现翻转效果，当鼠标悬停在父锚上时需要在热点span上应用一个边框：

```
.remote a:hover .hotspot,  
.remote a:focus .hotspot {  
    border: 1px solid #ffff;  
}
```

同样，为了在鼠标悬停在文本或热点span上时改变文本颜色，当鼠标悬停在父锚上或以其他方式获得焦点时需要改变span上的样式：

```
.remote a:hover .link ,  
.remote a:focus .link {  
    color: #0066FF;  
}
```

如果在Safari和Firefox中测试这个示例，效果会很好（见图6-15）。如果鼠标悬停在人名上，

链接文本会改变颜色，同时图像中这个人身上出现一个框。如果鼠标悬停在图像中某个人身上，也会发生同样的情况。

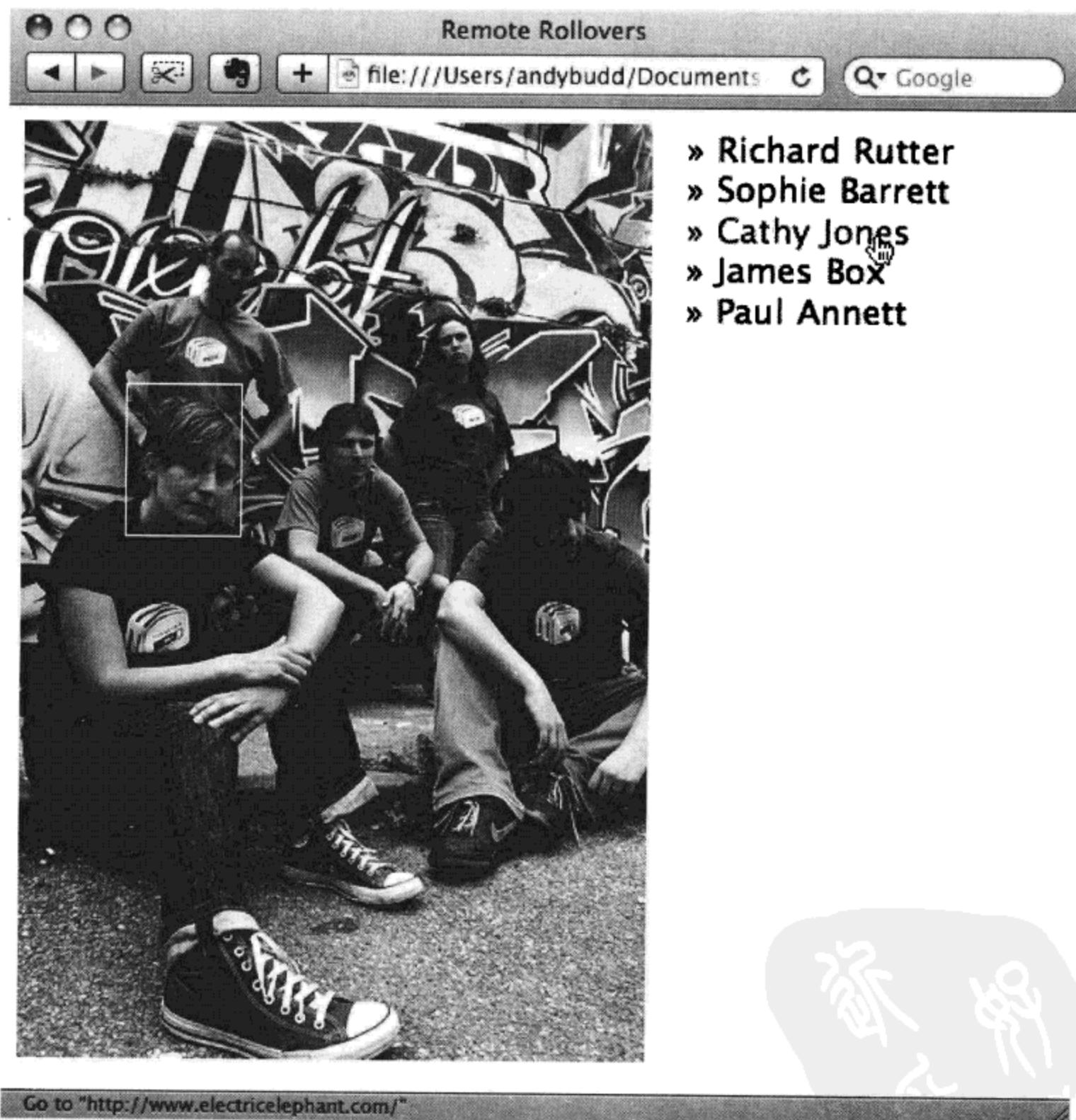


图6-15 远距离翻转演示。当图像下面的链接文本翻转时，图像中相关的人身上出现一个框

尽管这个示例的样式非常简单，但是可以尽可能发挥想像力来扩展它。实际上，我们在 Clearleft 站点的 Who we are 部分 (<http://clearleft.com/is/>) 中使用了这种技术，但是有一些变化（见图6-16）。

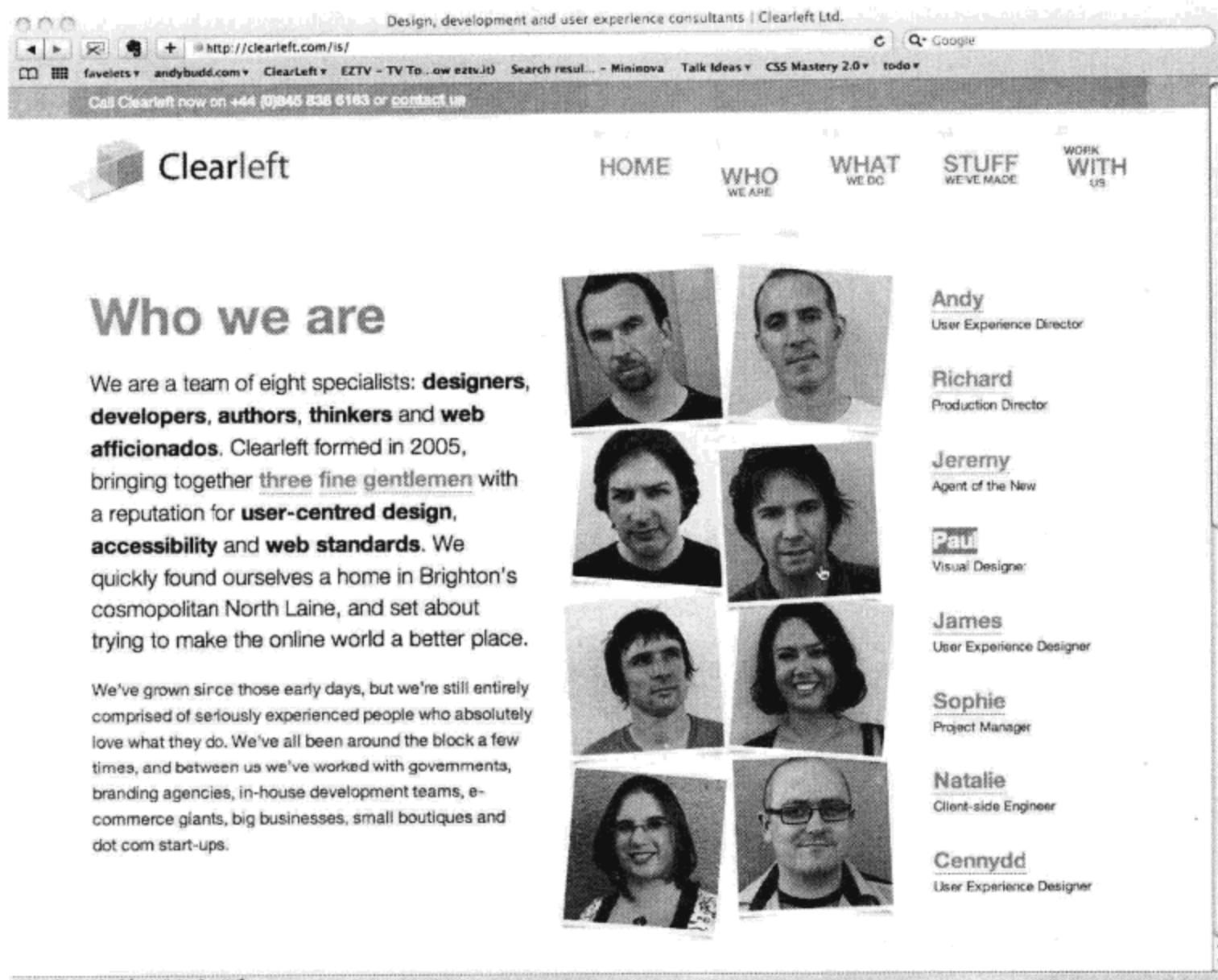


图6-16 当鼠标悬停在Clearleft团队的照片上时，右边列表中的人名突出显示

## 6.10 对于定义列表的简短说明

本章讨论了如何用无序列表创建各种效果（也可以扩展到有序列表）。但是，还有第三种常常被忽视的列表类型——定义列表，这种列表受到越来越多的关注。定义列表由两个核心组件组成：定义术语`<dt>`和一个或多个定义描述`<dd>`。

```

<dl>
  <dt>Apple</dt>
    <dd>Red, yellow or green fruit</dd>
    <dd>Computer company</dd>
  <dt>Bananna</dt>
    <dd>Curved yellow fruit</dd>
</dl>

```

顾名思义，定义列表的主要用途是组成“定义”。但是，HTML规范相当含糊，它建议可以

将定义列表用于产品属性或会话等其他用途。这使“定义”的概念有了一定程度的延伸，但是HTML是一种简单的文本格式化语言，在这个上下文中“定义”的意义还是比较明确的。

许多网络标准先驱者意识到，定义列表可以用来对一系列相关元素进行结构性分组，因此开始使用它们创建各种东西，从产品清单到图像库，再到表单和页面布局。毫无疑问，这些技术非常精彩，但是我个人认为他们把定义列表的含义延伸得太远了。

以这种风格使用定义列表的理由之一是，没有其他HTML元素能够提供这种类型的关联。但是，这不完全正确，因为div元素的用途就是把文档分成逻辑部分。更令人担心的是，这个理由与使用表格进行布局的理由完全一样。这使人很担心定义列表开始被滥用。

如果你想了解关于定义列表样式的更多信息，我推荐你阅读Mark Norman Francis在24 Ways发表的文章（[http://24ways.org/2007/my-other-christmas-present-is-adefinition-list](http://24ways.org/2007/my-other-christmas-present-is-a-definition-list)）。

## 6.11 小结

本章学习了如何实现灵活的列表样式，如何创建垂直导航条和水平导航条，包括可访问的标签页式导航，最后学习了如何通过定位创建纯CSS图像映射和远距离翻转。

下一章将学习如何创建可访问的表单布局和数据表格，以及如何用CSS对它们应用样式。



## 第7章

# 对表单和数据表格应用样式

---

随着Web的交互性要求越来越多，表单成为现代Web应用程序中越来越重要的部分。表单使用户能够与系统交互，完成从注册反馈到预定复杂的旅行安排的各种活动。因此，表单可能很简单（比如只有邮件地址和消息字段），也可能非常复杂，跨越多个页面。过去常常使用表格布局表单；但是，在本章中，你将了解到使用CSS可以设计出复杂的表单。

表格正在慢慢地恢复它们原来的用途，也就是只用来显示表格数据，而不用来进行页面布局。除了需要捕获用户数据之外，Web应用程序还越来越需要以容易使用且容易理解的格式显示这些数据。表单和数据表格设计在喜欢时髦的设计界很不受重视。但是，出色的信息和交互设计对于现代Web应用程序是很重要的。

在本章中，你将学习：

- 创建有吸引力且可访问的数据表格；
- 创建简单和复杂的表单布局；
- 对各种表单元素应用样式；
- 提供可访问的表单反馈。

## 7.1 对数据表格应用样式

许多开发人员认识到了基于表格的设计的缺陷，所以尽可能避免使用布局表格。一小部分开发人员甚至更绝，他们试图完全抛弃表格，用纯CSS重新创建日历布局等效果。尽管这种战略的用意很好，但是日历从本质上讲仍然是基于表格的内容。毕竟，它们基本上是由星期构成的行和天构成的列组成的。因此，在Web上仍然有需要使用表格的地方。

即使是相当简单的数据表格，如果包含的行和列比较多的话，它们也可能很难阅读。如果在数据单元格之间没有分隔线，信息就会混在一起，形成杂乱的布局（见图7-1）。

<u>&lt; January 2008 &gt;</u>						
S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

图7-1 紧凑的数据表格初看上去非常混乱

与之相反，有大量空白的表格也很难阅读，因为列和单元格之间距离太远，失去了视觉上的关联。如果用户沿着表格的行查看信息，而列的距离非常远（如图7-2所示的表格），那么这尤其成问题。在列之间移动时，如果不小心，就很容易看串行。这种现象在表格中央最明显，因为在图中表格上下边缘提供的视觉校准作用比较弱。

<u>&lt; January 2008 &gt;</u>						
S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

图7-2 间距太大的表格也很难马上理解

相比之下，花几分钟设计数据表格就可以大大改进可理解和阅读信息的速度。例如，我们给图7-3中的日期添加一点垂直和水平内边距，让它们彼此隔开。还通过隐约的斜面效果突出显示它们，让它们看起来是可以点击的。主要的列标题因不同的背景颜色、底边框和排版处理区别于数据。这样就产生了一个容易使用的日历组件。

January 2008						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

图7-3 应用了样式的数据表格

### 7.1.1 表格特有的元素

如果数据表格对于视力没问题的用户都难以阅读，那么对于那些必须使用辅助技术（比如屏幕阅读器）的人来说，它们是多么复杂和混乱。幸亏HTML规范提供了许多元素和属性来提高数据表格对于这些设备的可访问性。并非所有这些元素都得到了屏幕阅读器的支持，但是尽可能使用它们肯定没错。

#### 1. **summary**和**caption**

第一个元素是表格的**caption**，它基本上用作表格的标题。尽管这不是必须有的元素，但是尽可能使用**caption**总是好的。在这个示例中，我使用**caption**显示用户当前看到的月份。另一个有用的元素是表格的**summary**。**summary**属性可以应用于表格标签，用来描述表格的内容。与图像的**alt**文本相似，**summary**应该总结表格中的数据，编写得出色的**summary**可以减少用户阅读表格内容的需要。

```
<table class="cal" summary="A calendar style date picker">
  <caption>
    <a href="cal.php?month=dec08" rel="prev">&lt;&gt; January 2008 &lt;&gt;</a>
    <a href="cal.php?month=feb09" rel="next">&gt;&lt; February 2009 &lt;&gt;</a>
  </caption>
</table>
```

#### 2. **thead**、**tbody**和**tfoot**

利用**thead**、**tbody**和**tfoot**可以将表格划分为几个逻辑部分。例如，可以将所有列标题放在**thead**元素中，这样就能够对这个特殊区域单独地应用样式。如果选择使用**thead**或**tfoot**元素，那么必须至少使用一个**tbody**元素。在一个表格中只能使用一个**thead**和**tfoot**元素，但是可以使用多个**tbody**元素将复杂的表格划分为更容易管理的几部分。

行标题和列标题应该使用**th**而不是**td**标记，但是如果某些内容既是标题又是数据，那么它仍然应该使用**td**。表格标题可以设置值为**row**或**col**的**scope**属性，定义它们是行标题还是列标题。它们还可以设置**rowgroup**或**colgroup**的值，表示它们与多行或多列相关。

```
<thead>
  <tr>
    <th scope="col">Sun</th>
    <th scope="col">Mon</th>
    <th scope="col">Tue</th>
    <th scope="col">Wed</th>
    <th scope="col">Thu</th>
    <th scope="col">Fri</th>
    <th scope="col">Sat</th>
  </tr>
</thead>
```

### 3. col和colgroup

虽然利用tr元素能够对整行应用样式，但是很难对整列应用样式。为了解决这个问题，W3C引入了colgroup和col元素。colgroup能够使用col元素对一个或多个列定义和分组。不过，支持col和colgroup元素样式的浏览器并不多。

```
<colgroup>
  <col id="sun" />
  <col id="mon" />
  <col id="tue" />
  <col id="wed" />
  <col id="thur" />
  <col id="fri" />
  <col id="sat" />
</colgroup>
```

## 7.1.2 数据表格标记

将所有这些HTML元素和属性组合在一起，就可以创建出图7-3这样的日历表格的基本轮廓。

```
<table class="cal" summary="A calendar style date picker">
  <caption>
    <a href="#" rel="prev">&lt;</a> January 2008 <a href="#" rel="next">&gt;</a>
  </caption>
  <colgroup>
    <col id="sun" />
    <col id="mon" />
    <col id="tue" />
    <col id="wed" />
    <col id="thur" />
    <col id="fri" />
    <col id="sat" />
  </colgroup>

  <thead>
    <tr>
      <th scope="col">Sun</th>
      <th scope="col">Mon</th>
      <th scope="col">Tue</th>
      <th scope="col">Wed</th>
      <th scope="col">Thur</th>
      <th scope="col">Fri</th>
      <th scope="col">Sat</th>
    </tr>
  </thead>
```



```

<tbody>
  <tr>
    <td class="null">30</td>
    <td class="null">31</td>
    <td><a href="#">1</a></td>
    <td><a href="#">2</a></td>
    <td><a href="#">3</a></td>
    <td><a href="#">4</a></td>
    <td><a href="#">5</a></td>
  </tr>
  <tr>
    <td><a href="#">6</a></td>
    <td><a href="#">7</a></td>
    <td class="selected"><a href="#">8</a></td>
    <td><a href="#">9</a></td>
    <td><a href="#">10</a></td>
    <td><a href="#">11</a></td>
    <td><a href="#">12</a></td>
  </tr>
  ...
</tbody>
</table>

```

### 7.1.3 对表格应用样式

CSS规范有两个表格边盒模型：单独的和叠加的。在单独模型中，在各个单元格周围有边框，而在叠加模型中，单元格共享边框。大多数浏览器默认采用单独模型，但是我认为叠加模型更有用。因此，通常首先要做的一件事就是将表格的border-collapse属性设置为collapse。但是，为了便于讲解，我希望使用双边框创建斜面效果。因此，首先把border-collapse属性设置为separate。然后，让表格中的所有文本居中，删除默认的内边距和外边距。

```

table.cal {
  border-collapse: separate;
  border-spacing: 0;
  text-align: center;
  color: #333;
}

.cal th, .cal td {
  margin: 0;
  padding: 0;
}

```

CSS的border-spacing属性可以控制单元格之间的距离。可惜IE 7和更低版本不理解这个属性，因此需要使用老式但可靠的cellspacing属性。严格地说，这个属性在本质上是表现性的。但是，它仍然是有效的HTML，而且是当前在IE 6和IE 7中控制单元格间距的唯一方法。

```
<table cellspacing="0" class="cal" summary="A calendar style date picker">
```

#### 7.1.4 添加视觉样式

现在已经了解了基本概念，就该添加视觉样式了。为了让表格标题更像是常规的标题，可以增加字号并且采用粗体显示。还可以通过应用垂直内边距增加标题上下的空间。

```
.cal caption {
    font-size: 1.25em;
    padding-top: 0.692em;
    padding-bottom: 0.692em;
    background-color: #d4dde6;
}
```

为了确定当前月份两边的前一个和下一个链接的位置，给它们设置水平外边距，然后让它们分别向左边和右边浮动。然后，可以通过应用内边距让它们的点击区域更突出。为了给这些链接设置样式，我决定使用属性选择器来选择rel属性。但是，如果希望支持比较老式的浏览器，可以在每个链接中添加一个类。选择这些链接之后，可以应用你喜欢的任意样式。在这个示例中，我将仅在鼠标悬停时改变链接的背景颜色。

```
.cal caption [rel="prev"] {
    float: left;
    margin-left: 0.2em;
}

.cal caption [rel="next"] {
    float: right;
    margin-right: 0.2em;
}

.cal caption a:link,
.cal caption a:visited {
    text-decoration: none;
    color: #333;
    padding: 0 0.2em;
}

.cal caption a:hover,
.cal caption a:active,
.cal caption a:focus {
    background-color: #6d8ab7;
}
```

为了区分包含表格标题的第一行，我给它们设置的背景比表格其余部分浅，并且添加下划线。我还让文本比表格其余部分略小一点儿。

```
.cal thead th {
    background-color: #d4dde6;
    border-bottom: 1px solid #a9bacb;
    font-size: 0.875em;
}
```

在默认情况下，希望表格体中的文本是灰色的，这表示不可以选择它们。我还给文本设置了文本阴影。

```
.cal tbody {
    color: #a4a4a4;
    text-shadow: 1px 1px 1px white;
    background-color: #d0d9e2;
}
```

为了让表格单元格产生斜面效果，需要在各个边上设置不同的颜色；顶部和左边颜色浅，底部和右边颜色深。然后需要设置锚链接的样式。在这里，我把它们都设置为block并应用内边距，创建像按钮一样的点击区域。还要加粗字体并设置比较深的背景颜色。

```
.cal tbody td {
    border-top: 1px solid #e0e0e1;
    border-right: 1px solid #9f9fa1;
    border-bottom: 1px solid #acacad;
    border-left: 1px solid #dfdfc0;
}

.cal tbody a {
    display: block;
    text-decoration: none;
    color: #333;
    background-color: #c0c8d2;
    font-weight: bold;
    padding: 0.385em 0.692em 0.308em 0.692em;
}
```

最后，设置锚链接的鼠标悬停状态。前面选择的日期将通过包含selected类继承这个样式。在这里，我让链接变成蓝色背景上的白色文本并设置细微的文本阴影。

```
.cal tbody a:hover,
.cal tbody a:focus,
.cal tbody a:active,
.cal tbody .selected a:link,
.cal tbody .selected a:visited,
.cal tbody .selected a:hover,
.cal tbody .selected a:focus,
.cal tbody .selected a:active {
    background-color: #6d8ab7;
    color: white;
    text-shadow: 1px 1px 2px #22456b;
}
```

当鼠标悬停在日期上时，你会发现日期仍然是斜面的。如果希望显示日期被按下的外观，那么修改单元格的边框颜色，让顶和左边框比较深，底和右边框比较浅。注意，因为这个样式在非锚元素上使用hover伪选择器，所以它在IE 6中是无效的。如果需要在IE 6中使用这种技术，需要在链接中添加边框。

```
.cal tbody td:hover,  
.cal tbody td.selected {  
    border-top: 1px solid #2a3647;  
    border-right: 1px solid #465977;  
    border-bottom: 1px solid #576e92;  
    border-left: 1px solid #466080;  
}
```

看到吧，一个与图7-3相似的漂亮的日期选择组件。

## 7.2 简单的表单布局

当表单标签出现在相关联的表单元素正上方时，简短的表单最容易填写。用户只需顺着表单向下移动，依次阅读每个标签并填写后面的表单元素。这个方法最适合用在短的表单上，用来收集相当简单的且可预测的信息，比如联系人的详细信息（见图7-4）。

The form is titled "Your Contact Details". It includes the following fields:

- Name: (Required)
- Email Address:
- Web Address:
- Comments
- Message: (Required)

图7-4 简单的表单布局

### 7.2.1 有用的表单元素

HTML提供了许多有用的元素，可以帮助在表单中添加结构和赋予含义。其中第一个元素是 `fieldset` 元素。`fieldset` 用来对相关的信息块进行分组。在图7-4中，使用了两个 `fieldset`：一个用于联系人详细信息，另一个用于注释。大多数用户代理在 `fieldset` 周围加上一个细线边框，将 `border` 属性设置为 `none` 可以关闭它。

为了识别每个 `fieldset` 的用途，可以使用 `legend` 元素。`legend` 就像是 `fieldset` 的标题，它常常在 `fieldset` 的顶部垂直居中，并且向右缩进一点儿。但是，在 `legend` 上很难应用样式，因为各种浏览器处理它们的方式不一致。有一些浏览器（比如 Firefox 和 Safari）通过内边距形成一点儿缩进。但是，其他浏览器（比如 Opera 和 IE）有较大的默认缩进，而且不能使用内边距、外边距或通过定位进行控制。因此，如果使用 `legend`，那么不得不接受浏览器之间的差异。

#### 表单标签

`label` 元素极其重要，因为它可以帮助添加结构和增加表单的可用性和可访问性。顾名思义，这个元素用来在每个表单元素中添加有意义的描述性标签。在许多浏览器中，单击标签元素将导致相关联的表单元素获得焦点。使用标签的真正好处是增加了表单对于使用辅助设备的用户的易用性。如果表单使用标签，屏幕阅读器会正确地将表单元素和它的标签关联起来。如果没有标签，屏幕阅读器就必须“猜测”哪些文本与哪个表单元素相关，有时候它会猜错。屏幕阅读器用户还可以收集表单中所有标签，这样他们就能够以听的方式像正常人看表单一样浏览表单。

将标签与表单控件关联起来非常容易，而且可以采用两种方式之一来实现。一种是隐式方式，即把表单元素嵌套在 `label` 元素中：

```
<label>email <input name="email" type="text"/></label>
```

另一种是显式方式，即把 `label` 的 `for` 属性设置为相关联的表单元素的 `id` 名称：

```
<label for="email">email</label>
<input name="email" id="email" type="text"/>
```

你会注意到，这个输入控件和本章中的所有表单控件都包含 `name` 和 `id` 属性。在表单输入控件和标签之间创建关联需要 `id` 属性，而将表单数据发送回服务器需要 `name` 属性。`id` 和 `name` 不必相同，但是为了保持一致，我喜欢尽可能让它们相同。

如果使用 `for` 属性将标签和表单控件关联起来，那么就不要求这些控件在源代码中放在一起，它们可以在文档的不同部分中。但是，从结构化的观点来看，把控件和它们的标签分开是不明智的，应该尽可能避免这样做。

### 7.2.2 基本布局

可以使用这3个结构性元素对第一个 `fieldset` 的内容加标记，从而形成表单的基本布局。没

有应用样式的表单见图7-5。

```
<fieldset>
  <legend>Your Contact Details</legend>
  <div>
    <label for="author">Name:</label>
    <input name="author" id="author" type="text" />
  </div>
  <div>
    <label for="email">Email Address:</label>
    <input name="email" id="email" type="text" />
  </div>
  <div>
    <label for="url">Web Address:</label>
    <input name="url" id="url" type="text" />
  </div>
</fieldset>
```

图7-5 没有应用样式的表单

首先，需要设置`fieldset`和`legend`元素的一般样式。`fieldset`必须使用外边距进行垂直分隔，可以使用内边距在内容周围增加一些空间。为了突出显示`fieldset`，可以给它们设置浅色的背景和略深的1像素边框。不要让背景颜色太深，因为这会使表单看上去太沉重，使它更难理解。让`legend`元素显示为粗体也可以突出信息，让它更容易理解。

```
fieldset {
  margin: 1em 0;
  padding: 1em;
  border: 1px solid #ccc;
  background: #f8f8f8;
}

legend {
  font-weight: bold;
}
```

将标签定位在表单元素的上方实际上非常简单。标签在默认情况下是行内元素。但是，将它们的`display`属性设置为`block`会使它们产生自己的块框，迫使输入元素转到下一行。在不同的浏览器中，文本输入框的宽度不一样，因此为了一致，应该显式地设置文本输入框的宽度。在这

在本示例中，我使用em创建可伸缩的表单布局。

```
label {
    display: block;
    cursor: pointer;
}

input {
    width: 20em;
}
```

在这里，最好把标签的光标样式改为pointer，这表明可以与标签交互。

### 7.2.3 其他元素

这个布局同样适用于其他表单元素，比如文本区域：

```
<fieldset>
    <legend>Comments</legend>
    <div>
        <label for="text">Message: </label>
        <textarea name="text" id="text">
        </textarea>
    </div>
</fieldset>
```

文本区域的尺寸在不同的浏览器中也不一样，所以也应该显式地设置它们的宽度和高度。在这里，我把宽度设置为100%，所以宽度实际上由父元素决定。这样设置宽度是个好方法，这会让布局更灵活、更独立。

```
textarea {
    width: 100%;
    height: 10em;
}
```

与文本区域和文本输入控件不同，需要以不同的方式处理单选按钮和复选框。这些元素通常并不把标签放在它们上方，而是放在右边。当垂直排列时，所有元素左对齐，使用户容易选择（见图7-6）。

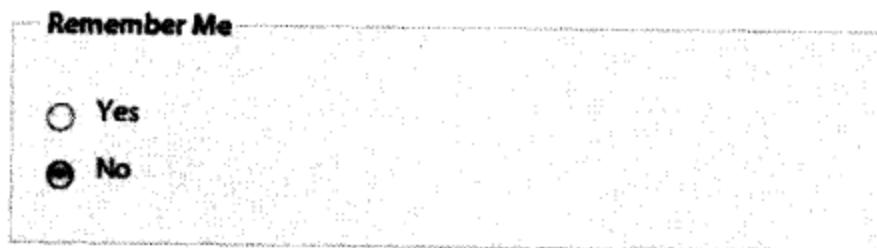


图7-6 单选按钮布局

在前面的示例中，通过在输入元素上应用宽度，定义了文本框的宽度。但是，输入元素包括其他表单控件（如复选框、单选按钮和提交按钮）和比较常用的文本输入框。因此，如果将输入元素设置为20 em宽的话，所有输入元素都会成为20 em宽。

解决这个问题的一种方法是，使用属性选择器寻找特定类型的表单元素。所以并不把所有输入元素都设置为20 em，而是专门寻找文本输入元素：

```
input[type="text"] {
    width: 20em;
}
```

遗憾的是，属性选择器只在比较现代的浏览器中可用，在IE 6和更低版本中是无效的。在属性选择器得到更广泛的支持之前，区分输入元素的最好方法是给它们分配一个类。

例如，可以给单选按钮分配类名radio：

```
<fieldset>
    <legend>Remember Me</legend>
    <div>
        <label for="remember-yes"><input id="remember-yes" class="radio" type="radio" name="remember" value="yes" />Yes</label>
    </div>

    <div>
        <label for="remember-no"><input id="remember-no" class="radio" type="radio" name="remember" value="no" checked="checked" />No</label>
    </div>
</fieldset>
```

然后可以将单选按钮的宽度设置为auto，从而覆盖前面对输入元素的设置。对于复选框和提交按钮也可以这样做：

```
input.radio, input.checkbox, input.submit {
    width: auto;
}
```

注意这里如何让标签围绕表单元素。在前面，我把表单中的所有标签设置为块级元素，这让相关联的表单控件出现在另一行上。显然，对于单选按钮标签不希望这样，所以通过让标签围绕表单控件来避免它。

最后，需要在单选按钮中添加一点儿外边距，从而在标签之间提供适当的距离。

```
#remember-me .radio {
    margin-right: 1em;
}
```

## 7.2.4 修饰

布局现在完成了，但是还可以为比较高级的浏览器添加一些修饰。例如，在元素获得焦点时可以改变背景颜色，从而帮助用户轻松地找到要填写的表单域：

```
input[type="text"]:focus, textarea:focus {
    background: #ffc;
}
```

还可以通过定制边框，让文本框和文本区域元素的外观保持一致。这对于Firefox尤其有用，Firefox将这些元素的底边框和右边框显示为白色，所以在白色背景上它们不够清晰（见图7-7）。

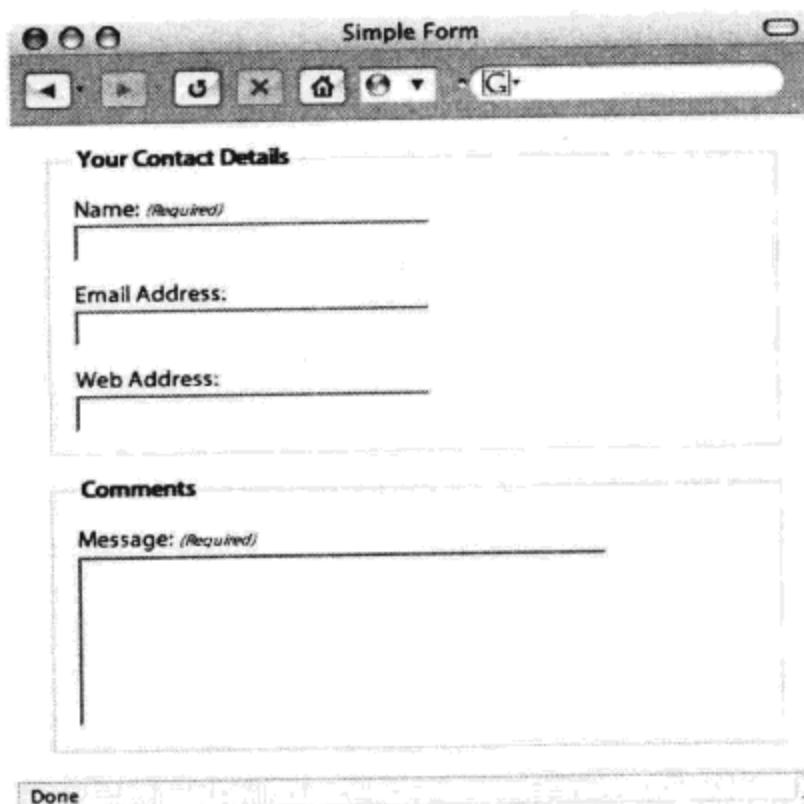


图7-7 在Firefox中文本输入框和文本区域的底边框和右边框是白色的，这使得它们在白色背景上不够清晰

在这个示例中，使用属性选择器来寻找文本输入元素，因为这个样式主要针对的是Firefox，而Firefox理解这种选择器。

```
input[type="text"], textarea {
    border-top: 2px solid #999;
    border-left: 2px solid #999;
    border-bottom: 1px solid #ccc;
    border-right: 1px solid #ccc;
}
```

在这个示例中，我们没有使用任何密码字段。但是，如果要为整个站点创建通用的表单样式，还需要在前两个示例中包含 [type="password"]。

### 必填域

许多表单包含必须填写的域。可以在这些必填域旁边放上有样式的文本或星号，以此来表示必须填写这些域。因为这一信息用来强调域是必须填写的，最适合这一信息的元素是em或strong元素：

```
<div>
<label for="author">Name:<em class="required">(required)</em>/label>
<input name="author" id="author" type="text" />
</div>
```

但也可以对这一信息应用样式。在这个示例中，减小字号并且让文本成为红色：

```
.required {
  font-size: 0.75em;
  color:#760000;
}
```

现在你明白了，我们使用纯CSS建立了一个简单但美观的表单布局。

## 7.3 复杂的表单布局

对于比较长的复杂的表单，垂直间距会造成问题，表单也不容易查看。为了便于浏览，减少使用的垂直空间，有必要将标签和表单元素水平布置，而不是垂直布置。创建如图7-8所示的表单实际上非常简单，而且使用的代码几乎与前一个示例完全一样。

The form is titled "Your Contact Details". It contains the following fields:

- Name:** (Required)
- Email Address:**
- Web Address:**
- Comments**
- Message:** (Required)
- Remember Me**
- Yes     No
- Submit**

图7-8 水平表单布置

这个示例和前一个示例之间的唯一差异是，并不将标签设置为块级元素，而是将标签向左浮动。我们还需要给标签设置宽度，让所有表单元素排齐：

```
label {
    float: left;
    width: 10em;
    cursor: pointer;
}
```

如果表单标签可能跨多行，还应该清理容器div。这会避免它们干扰下一组标签和弄乱布局。

```
form div {
    clear: left;
}
```

很少有表单像图7-8那样简单，而且常常需要处理基本表单样式规则以外的情况，比如单行上的多个表单控件或者复选框或单选按钮的列（见图7-9）。下面两小节将解释如何处理这些例外情况。

<b>Place of Birth:</b>	<input type="text" value="USA"/>		
<b>Date of Birth:</b>	<input type="text" value="24"/>	<input type="text" value="March"/>	<input type="text" value="1972"/>
<b>Favorite Color:</b>	<input type="checkbox"/> red	<input type="checkbox"/> orange	
	<input type="checkbox"/> yellow	<input type="checkbox"/> purple	
	<input type="checkbox"/> pink	<input type="checkbox"/> blue	
	<input type="checkbox"/> green	<input type="checkbox"/> other	

图7-9 比较复杂的表单布局

### 7.3.1 可访问的数据输入元素

从前面的示例可以了解到，表单标签对于表单的可访问性很重要。但是，有时候我们不希望为每个元素都显示标签。例如，在图7-9中可以看到一组用于收集日期信息的表单元素。在这种情况下，显示每个标签就显得太烦人了，因为它将生日分割成3个单独的部分，看起来不像一个实体。但是，即使不希望显示标签，让标签出现在源代码中并且可供屏幕阅读器使用仍然很重要。

```
<div>
    <label for="dateOfBirth">Date of Birth:</label>
    <input name="dateOfBirth" id="dateOfBirth" type="text" />
    <label id="monthOfBirthLabel" for="monthOfBirth"> ←
    Month of Birth:</label>
```

```
<select name="monthOfBirth" id="monthOfBirth">
  <option value="1">January</option>
  <option value="2">February</option>
  <option value="3">March</option>
</select>
<label id="yearOfBirthLabel" for="yearOfBirth">Year of Birth:</label>
<input name="yearOfBirth" id="yearOfBirth" type="text" />
</div>
```

为了创建这个布局，首先需要隐藏“month of birth”和“year of birth”标签。将标签的display属性设置为none会阻止标签显示，但是也会阻止许多屏幕阅读器访问它们。所以不这样做，而是通过值较大的负文本缩进将标签定位到屏幕之外。在前面创建的基本表单样式中，标签已经设置了宽度，为了防止标签影响布局，还需要将这些标签的宽度设置为零：

```
#monthOfBirthLabel, #yearOfBirthLabel {
  text-indent: -1000em;
  width: 0;
}
```

然后可以单独设置各个表单控件的尺寸，并且用外边距控制它们的水平间距：

```
input#dateOfBirth {
  width: 3em;
  margin-right: 0.5em;
}

select#monthOfBirth {
  width: 10em;
  margin-right: 0.5em;
}

input#yearOfBirth {
  width: 5em;
}
```

### 7.3.2 多列复选框

将大的复选框组或单选按钮组创建成两列有点儿复杂。标签只能分别用于各个元素，而不能用于元素组。理想情况下，我们将整个组放在一个fieldset中，并且将legend用作这个组的标签。遗憾的是，由于浏览器处理legend的定位的方式不一致，所以这在当前不是可行的解决方案。因此，在浏览器提供更一致的支持之前，最好的解决方案是使用标题元素。

为了创建列效果，要将复选框分成两组，每一组放在一个div中各有一个col类。然后将这两个div元素一起放在一个具有描述性ID的fieldset中：

```

<fieldset id="favoriteColor">
  <h2>Favorite Color:</h2>
  <div class="col">
    <div>
      <label><input class="checkbox" id="red" name="red" type="checkbox" value="red" />red</label>
      ...
    </div>
  </div>

  <div class="col">
    <div>
      <label><input class="checkbox" id="orange" name="orange" type="checkbox" value="orange" />orange</label>
    </div>
    ...
  </div>
</fieldset>

```

因为已经创建了基本的fieldset样式，所以首先需要覆盖这些样式，将内边距和外边距设置为零，去掉边框并且将背景颜色设置为透明：

```

fieldset#favoriteColor {
  margin: 0;
  padding: 0;
  border: none;
  background: transparent;
}

```

标题将用作标签，所以它需要像其他标签一样向左浮动并且将宽度设置为10 em。标题还需要看起来像标签，所以需要将字体粗细设置为normal并且减小字号。

```

#favoriteColor h2 {
  width: 10em;
  float: left;
  font-size: 1em;
  font-weight: normal;
}

```

然后给div设置宽度并且让它们向左浮动，从而创建出两列的布局。但是，因为这个表单中的所有div在默认情况下已经被清理了，需要使用clear:none覆盖那个声明。

```

#favoriteColor .col {
  width: 8em;
  float: left;
  clear: none;
}

```

这个表单中的所有标签向左浮动并且将宽度设置为10 em。但是，复选框的标签不需要浮动，所以应该在这里覆盖那个声明。

```
#favoriteColor label {
  float: none;
}
```

现在有了一个相当复杂的表单布局。基本表单样式负责一般的布局，然后我们可以通过覆盖这些样式分别处理各种例外情况。

### 提交按钮

表单是在站点上添加交互功能并把数据提交给服务器的好方法。为了激活表单，需要某种按钮控件，通常使用的是一个type值设置为submit的input元素。input按钮是把数据提交给服务器的最常用方法，但是它们也有一些问题，包括无法只使用元素选择器选择它们。可以使用属性选择器选择它们，但是IE的老版本不支持这个属性，所以只能使用ID或类选择器直接选择它们。那么，为什么不使用button元素替代input元素呢？

现在button元素开始被人们接受，但是它仍然不太为人所知，使用还不广泛。不应该是这样的，因为button元素会提供很强的灵活性。首先，可以在button标签中放一个图像，让图像成为控件（见图7-10）。

```
<div>
<button type="submit">

</button>
</div>
```



图7-10 使用按钮图像的button元素

因为按钮有一些默认样式，我们希望关闭它们。

```
button {
  border: none;
  background: none;
  cursor: pointer;
}
```

OS X等许多操作系统禁止修改input按钮的样式，这是为了在整个操作系统中保持一致。但是，button元素不受这些限制的影响。因此，可以只使用CSS创建非常丰富的按钮样式。例如，假设最初有一个简单的提交按钮。

```
<p>
<button type="submit">Book Now ></button>
</p>
```

首先，可以给按钮设置显式的尺寸和有颜色的边框。然后，可以使用border-radius生成圆角并应用文本阴影。最后，可以通过使用图像或Webkit特有的渐变功能应用渐变的背景。结果如图7-11所示。

```
button.two {
    width: 200px;
    height: 50px;
    border: 1px solid #989898;
    -moz-border-radius: 6px;
    -webkit-border-radius: 6px;
    border-radius: 6px;
    background: url(/img/button-bg.png) #c5e063 bottom left repeat-x;
    -moz-box-shadow: 2px 2px 2px #ccc;
    -webkit-box-shadow: 2px 2px 2px #ccc;
    box-shadow: 2px 2px 2px #ccc;
    color: #fff;
    font-size: 26px;
    font-weight: bold;
    text-shadow: 1px 1px 1px #666;
}
```



图7-11 使用纯CSS的button元素

button元素的主要限制是IE 6（在某种程度上还有IE 7）处理提交的方式。其他浏览器提交value属性的内容，但是IE 6和IE 7提交元素本身的内容。因此，如果页面上有多个按钮，IE 6会提交所有按钮的内容，而不仅仅是被点击的按钮。如果希望在一个页面上使用多个按钮，就需要确保它们的功能是相同的，因为在IE的老版本中无法指出点击了哪个按钮。

### 7.3.3 表单反馈

表单常常需要某种形式的反馈消息，从而突出显示那些忘了填写或填写得不正确的域。采用的方法常常是在适当的域旁边添加一个错误消息（见图7-12）。

为了产生这种效果，可以将反馈文本放在一个em中，将这个em放在源代码中文本输入元素的后面。但是，为了让它们正确地排列，em和前面的input元素都需要浮动。这会对包围它们的段落产生影响，进而对整个布局产生影响。另外，许多屏幕阅读器会忽略表单元素之间的文本，除非把文本放在标签中。为了避免这些问题，最好的方法是将错误消息文本放在表单标签中，然

后使用CSS进行定位：

```
<div>
  <label for="email">Email Address:
  <em class="feedback">Incorrect email address. Please try again. ←
  </em>
  </label>
  <input name="email" id="email" type="text" />
</div>
```

图7-12 表单反馈示例

为了对反馈`em`进行定位，首先需要将表单中所有段落的`position`设置为`relative`，从而建立一个新的定位上下文。然后可以对反馈`em`进行绝对定位，让它出现在文本输入元素的右边。我们知道标签的宽度是10 em，文本框的宽度是20 em，所以可以把反馈`em`的左位置设置为30 em。

```
form div {
  position: relative;
}

form .feedback {
  position: absolute;
  left: 30em;
  right: 0;
  top: 0.5em;
}
```

但讨厌的是，IE 6和更低版本错误地将反馈`em`的宽度设置为最小宽度。为了解决这个问题，需要为这种浏览器设置显式的宽度。一种做法是使用第8章介绍的条件注释：

```
form .feedback{
  width: 10em;
}
```

然后可以对反馈消息应用需要的样式。在这个示例中，我让文本显示为红色的粗体，并且在消息的左边应用一个警告图像：

```
form div {  
    position: relative;  
}  
  
.feedback {  
    position: absolute;  
left: 30em;  
right :0;  
top: 0.5em;  
font-weight: bold;  
color: #760000;  
padding-left: 18px;  
background: url(/img/error.png) no-repeat left top;  
}
```

还可以使用这种技术提供正面的反馈或建议，指导用户填写表单的特定部分。

## 7.4 小结

在本章中，我们学习了在不同的情况下不同表单布局的效果，现在可以使用CSS布置出复杂的表单，而不需要借助于表格。我们还学习了应该如何使用表格显示数据（而不是进行布局），以及如何使数据表格的设计更有趣。

下一章将使用到目前为止学到的所有知识开始构建基于CSS的布局。





## 第8章

# 布 局

---

CSS的主要好处之一是，它能够控制页面布局而不需要使用表现性标记。但是，CSS布局被误认为是难以理解的，在初学者中，这种想法尤其普遍。这在一定程度上是由于浏览器的不一致造成的，但主要原因是Web上不同布局技术的激增所致。似乎每个CSS作者都用自己的技术创建多列的布局，而且新的CSS开发人员常常并不真正了解所使用的技术的工作原理。所谓的CSS框架的出现进一步恶化了这种局面。CSS框架试图通过在标记和表现之间建立强耦合来简化CSS布局，但是这种耦合正是我们抛弃基于表格的局的主要原因。这种进行CSS布局的“黑盒”方式可能会让人迅速地得到想要的结果，但是最终会阻碍开发人员理解这种语言和实现修改的能力。

所有CSS布局技术的根本都是3个基本概念：定位、浮动和外边距操纵。这些CSS布局技术其实没有本质的差异，如果理解了核心概念，那么创建自己的布局是相当容易的。实际上，布局是CSS最容易的部分，只需花些时间掌握它。

8

在本章中，你将学习：

- 让设计在页面中水平居中；
- 创建两列和三列的基于浮动的布局；
- 创建固定宽度、流式和弹性布局；
- 创建高度相等的列；
- CSS框架与CSS系统。

### 8.1 计划布局

在开始把设计转换为功能完整的模板时，设计者往往很想马上开始编写页面和处理图像。但

是，这样做很快就会遇到麻烦。相反，先做一点儿计划可以避免许多问题。正如俗话说的，“要三思而后行”。

要想创建可伸缩且容易维护的CSS系统，首先应该检查设计，寻找重复的模式，这包括页面结构中的模式或在站点中元素反复出现的方式。在这个阶段，不应该太关注表现方式，而是应该注意结构和意义。我喜欢打印出每个设计，圈出模式，然后在每个页面上随便写上批注（见图8-1）。但是，有些人喜欢在Photoshop文件上加批注或使用灰色框设计图。



图8-1 标记指南

先把页面划分为大的结构性区域，比如容器、页眉、内容区域和页脚。这些区域在整个站点中往往是一致的，很少改变。如果用建筑来类比，可以把这些区域看做建筑物的外墙。

然后，将注意力转移到内容区域本身，开始建立网格结构。设计有多少个不同的内容区域？它们有什么差异？从布局的角度来说，内容区域是确实不一样，还是可以被同等看待？大多数设计只有几个独特的内容区域，所以应该寻找共同的特征而不是视觉表现。你可以把这些内容区域看做建筑物的内部承重墙。

最后，在各个内容区域中寻找不同的布局结构。是否需要把某些信息分为两列、三列或四列？与前面不同，这些布局结构往往非常灵活，在各个页面之间常常有变化。可以把它们看做建筑物的隔断墙。将这些结构与前一步相结合，可以形成每个页面的结构计划。现在，要拿起绘图纸和彩色铅笔，开始详细设计结构和尺寸（见图8-2）。

结构设计完之后，现在可以开始关注不同类型的内容。内容是新闻稿、文章还是公告？给每个内容块起一个有意义的名称，然后分析它们的关系。可能新闻稿和公告实际上没什么差别，在这种情况下，把它们组合成一个内容类型是有道理的。

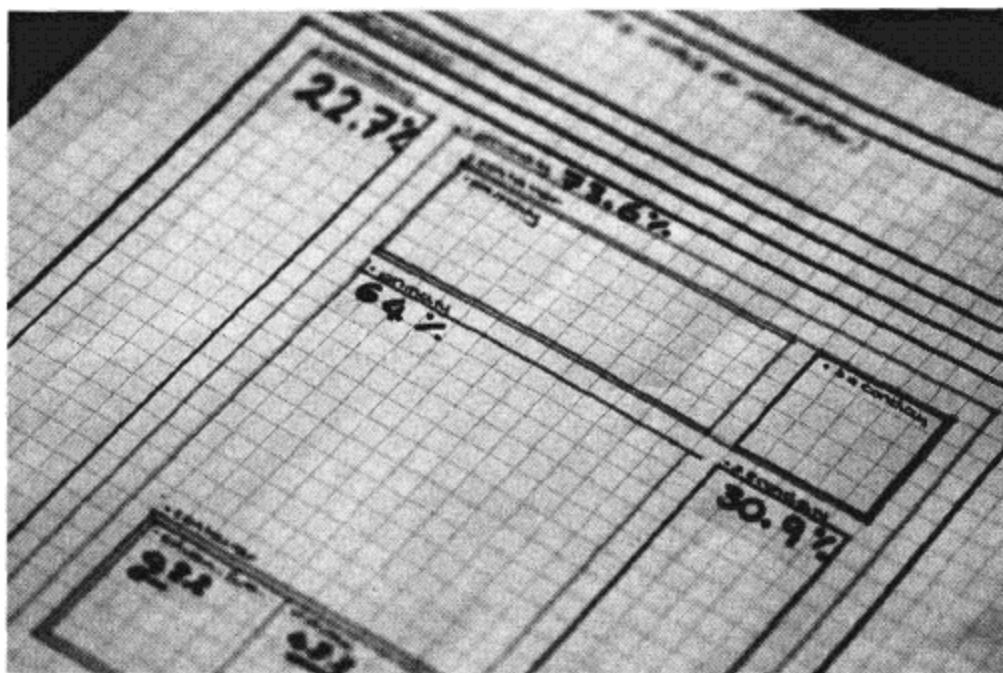


图8-2 在绘图纸上确定尺寸

查看每个内容块的结构，看看不同的类型中是否有共同的模式。例如，文章和新闻稿可能都有醒目的页眉和页脚，所以可以加上相应的标识。即使页眉和页脚看起来不一样，实际上也没关系，因为以后可以根据上下文应用样式。对于错误消息、搜索框和菜单项等也是这样。应该尽量采用一般的类名，然后根据上下文应用样式。

找出模式并确定命名约定之后，最好马上开始定义将使用的元素，这是有好处的。例如，链接列表可能是一个无序列表，而文章可能是一个

，其中包含h2、段落和锚元素。与几位同事一起提前定义元素，比随时添加元素容易得多。我还发现应该记下颜色编码、尺寸等信息，这在开发阶段会有帮助。另外，可以在设计的打印稿上添加批注以便快速参考，见图8-3。

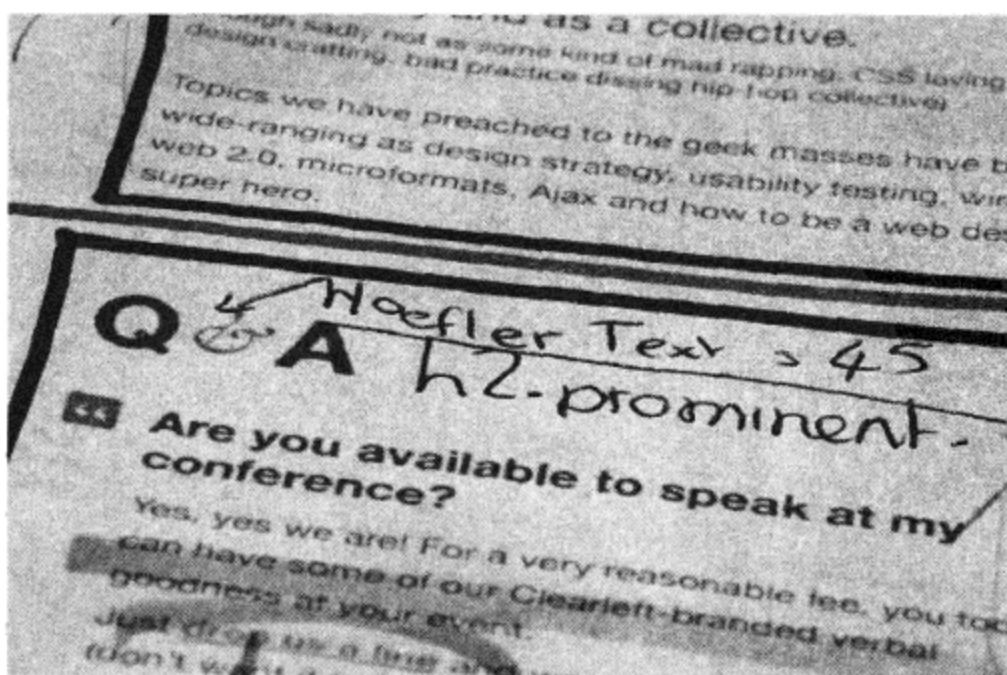


图8-3 确定不同类型细节

## 8.2 设置基本结构

假设我们要开发一个典型的三列博客模板，见图8-4。

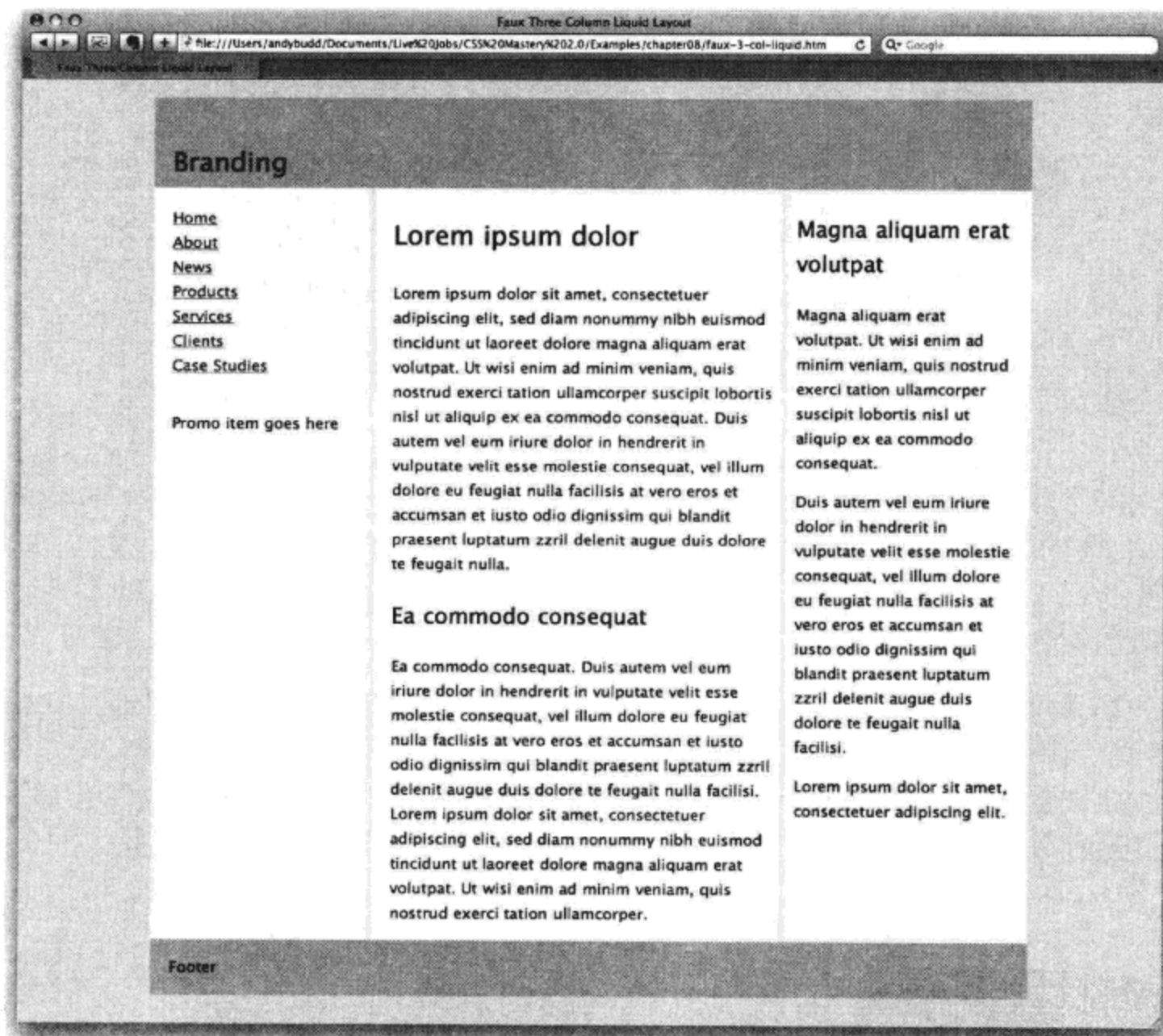


图8-4 典型的三列布局

通过分析设计发现，我们需要用一个容器元素让设计居中，其中包含页眉、内容区域和页脚。因此，标记应像下面这样：

```
<body>
  <div class="wrapper">

    <div class="header">
      <!--Your header content goes here-->
    </div>
```

```

<div class="content">
    <!--Your page content goes here-->
</div>

<div class="footer">
    <!--Your footer content goes here-->
</div>

</div>
</body>

```

因为后3个区域放在容器中，我们先设置容器元素的样式。

## 使用外边距让设计居中

长文本行可能阅读起来很困难，让人讨厌。现代显示器的尺寸越来越大，屏幕可读性问题变得越来越重要。缓解这个问题的一种方法是让设计居中。居中是指只占据屏幕的一部分，而不是横跨屏幕的整个宽度，这样就会创建比较短的容易阅读的行。

假设有一个典型的布局，希望让其中的容器div在屏幕上水平居中：

```

<body>
    <div class="wrapper">
    </div>
</body>

```

为此，只需定义容器div的宽度，然后将水平外边距设置为auto：

```

.wrapper {
    width: 920px;
    margin: 0 auto;
}

```

在这个示例中，我决定以像素为单位指定容器div的宽度，让它适合分辨率为1024×768的屏幕。但是，也可以将宽度设置为主体的百分之几，或者使用em相对于文本字号设置宽度。

这在所有现代浏览器中都是有效的。但是，混杂模式中的IE 5.x和IE 6不支持margin:auto声明。幸运的是，IE将text-align: center误解为让所有东西居中，而不只是文本。这一点是可以利用的，方法是让主体标签中的所有东西居中，包括容器div，然后让容器的内容重新向左对齐：

```

body {
    text-align: center;
}

.wrapper {
    width: 920px;
}

```

```

margin: 0 auto;
text-align: left;
}

```

以这种方式使用text-align属性是一种hack，但是这个hack是无害的，对代码没有严重的影响。现在，容器在IE的旧版本以及比较符合标准的浏览器中都是居中的（见图8-5）。

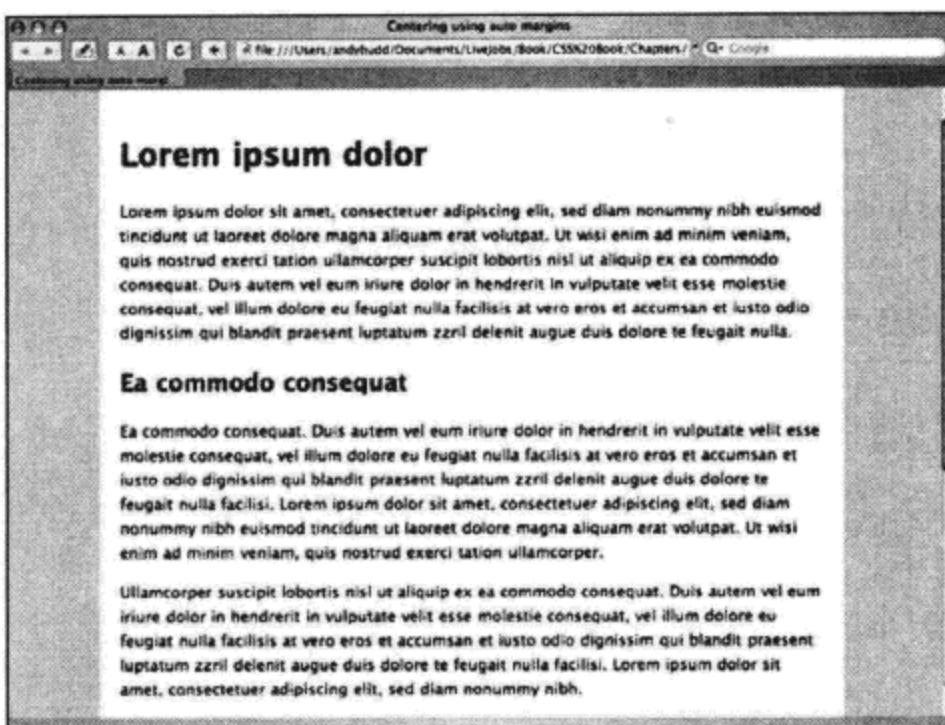


图8-5 使用margin:auto让设计居中

## 8.3 基于浮动的布局

用CSS进行布局有几种不同的方式，包括绝对定位和使用负的外边距值。我发现基于浮动的布局是最容易使用的，也是最可靠的。顾名思义，在基于浮动的布局中，只需设置希望定位的元素的宽度，然后将它们向左或向右浮动。

因为浮动的元素不再占据文档流中的任何空间，它们就不再对包围它们的块框产生任何影响。为了解决这个问题，需要对布局中各个点上的浮动元素进行清理。非常常见的做法不是连续地浮动和清理元素，而是浮动几乎所有东西，然后在整个文档的“战略点”（比如页脚）上进行一次或两次清理。还可以使用溢出方法清理某些元素的内容。这是我目前最喜欢的方法，所以在后面的示例中都使用这种方法。

### 8.3.1 两列的浮动布局

要在内容区创建一个两列布局，首先需要创建一个基本的HTML结构。

```

<div class="content">
<div class="primary">

```

```

<!-- main content goes here -->
</div>

<div class="secondary">
    <!--navigation and secondary content goes here -->
</div>

</div>

```

这个设计的次要内容区域（包括站点导航）将位于页面的左边，主内容位于右边。但是，为了提高易用性和可访问性，我选择在源代码中将主内容区域放在次要内容区域的前面。首先，主要内容是页面上最重要的东西，所以在文档中应该先出现。其次，这样可以方便屏幕阅读器用户，他们用不着经过可能很长的链接列表和不太重要的内容（比如站点宣传）就能找到主要内容部分。

在创建基于浮动的布局时，一般将两列都向左浮动，然后使用外边距或内边距在两列之间创建一个隔离带。在使用这种方法时，列在可用空间内包得很紧，没有喘息的空间。如果浏览器表现良好的话，这不是问题；但是差劲的浏览器会打乱紧密的布局，迫使一列退到另一列的下面。

在IE上就会发生这种情况，因为IE考虑元素内容的尺寸，而不是元素本身的尺寸。在符合标准的浏览器中，如果元素的内容太大，它只会超出框之外。但是，在IE上，如果元素的内容太大，整个元素就会扩展。这可能是由很小的东西引起的，比如把一些文本设置为斜体。如果这发生在非常紧密的布局中，那么就没有足够的空间可以让元素并排出现，有一个浮动元素就会退到下面去。其他IE bug（比如3像素文本偏移bug和双外边距浮动bug，见第9章）以及各种浏览器取整错误也会导致浮动元素下降。

为了防止发生这种情况，需要避免浮动布局在包含它们的元素中太挤。可以不使用水平外边距或内边距来建立隔离带，而是让一个元素向左浮动，让另一个元素向右浮动，从而创建视觉上的隔离（见图8-6）。如果一个元素的尺寸意外地增加了几个像素，那么它不会立刻占满水平空间并下降，而只是扩展到视觉隔离带中。

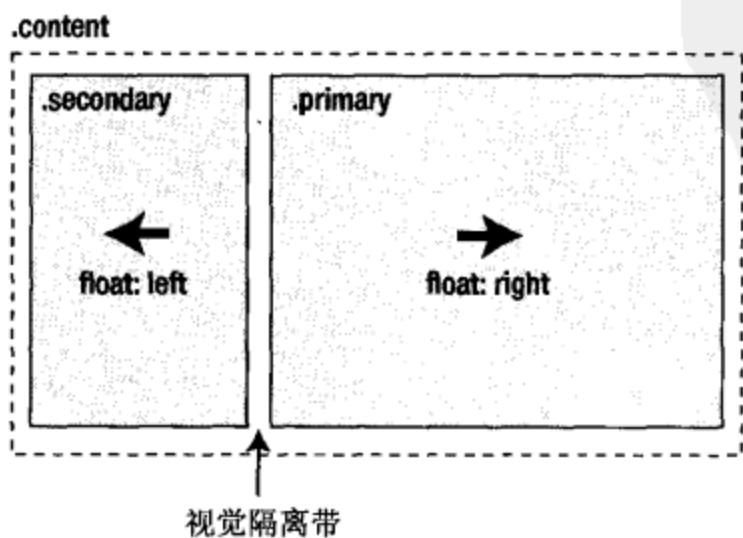


图8-6 使用浮动创建两列布局

实现这个布局的CSS非常简单。只需为每个列设置想要的宽度，然后将次要内容向左浮动，将主内容向右浮动。还需要在主内容中添加一点儿内边距，以避免其中包含的文本紧挨着元素的右边缘。注意，我还在所有浮动元素中添加了`display:inline`。这个预防措施用来防止IE中的双外边距浮动产生的bug（详细信息见下一章）。

```
.content .primary {
    width: 650px;
    padding-right: 20px;
    float: right;
    display: inline;
}

.content .secondary {
    width: 230px;
    float: left;
    display: inline;
}
```

因为总宽度是920像素，这些设置在每个浮动元素之间留出20像素宽的隔离带。正如前面提到的，这可以防止内容扩展导致浮动元素下降。

因为这些元素是浮动的，它们不再在文档流中占据任何空间，这会导致页脚向上升。为了避免这种情况，需要对它们的父元素（在这里是内容div）应用溢出方法，从而清理浮动元素。

```
.content {
    overflow: hidden;
}
```

现在你看到效果了：一个简单的两列CSS布局（见图8-7）。



图8-7 浮动的两列布局

注意，我没有把创建的两个元素命名为primary-content和secondary-content，而是简称为primary和secondary。这两个元素嵌套在content元素中，可以创建关联，我利用了这一事实。这有两个好处。首先，不需要为要应用样式的每个元素创建新的类名，而应使用层叠找到元素。更重要的是，可以多次使用primary和secondary类，这会创建一个非常灵活的命名系统。例如，假设我们希望创建的是三列布局而不是两列布局，具体方法请见下一节。

### 8.3.2 三列的浮动布局

创建三列布局所需的HTML与两列布局的HTML非常相似，唯一的差异是在内容div中添加了两个新的div：一个用于主内容，另一个用于次要内容。因此，可以重用灵活的primary和secondary类名。

```
<div class="content">

    <div class="primary">
        <div class="primary">
            <!-- your primary primary content goes here -->
        </div>
        <div class="secondary">
            <!-- your secondary primary content goes here -->
        </div>
    </div>

    <div class="secondary">
        <!--navigation and secondary content goes here -->
    </div>

</div>
```

可以使用与两列布局技术相同的CSS，将次要内容向左浮动，将主内容向右浮动。然后，在主内容div中，将主div向左浮动，将次要div向右浮动（见图8-8）。这其实就是将主内容区域分成两列，从而形成三列的效果。

与前面一样，所用的CSS非常简单。只需设置想要的宽度，然后将主div向左浮动，将次要div向右浮动，在中间留出20像素的间距：

```
.content .primary .primary {
    width: 400px;
    float: left;
    display: inline;
}

.content .primary .secondary {
    width: 230px;
    float: right;
    display: inline;
}
```

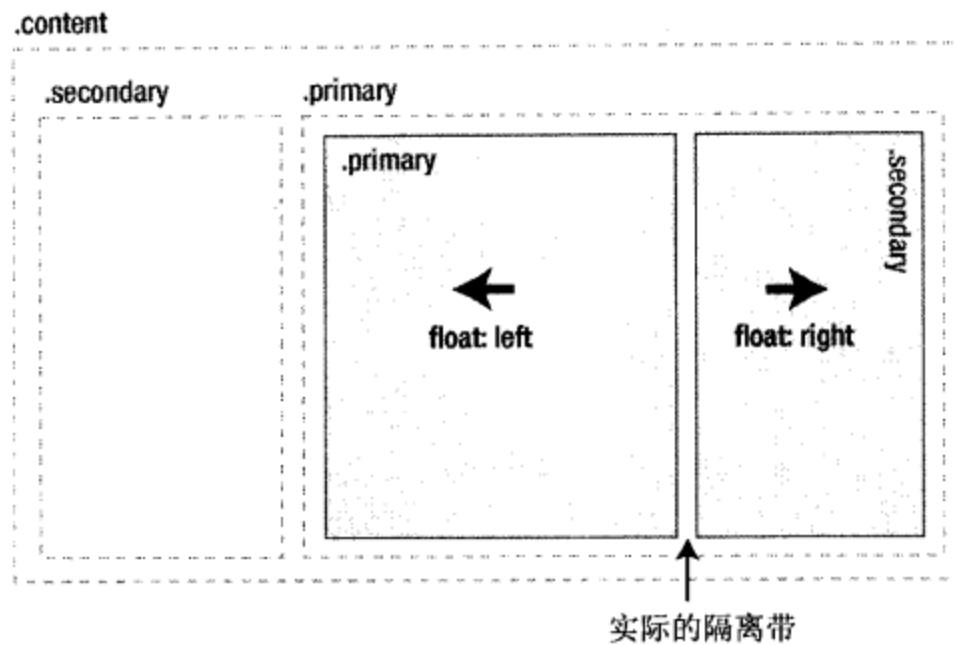


图8-8 通过将内容列分成两列，形成三列布局

注意，第一个示例中主div中添加的右内边距现在要应用在第二个示例的主div上。因此，需要从一般样式中删除内边距，把它应用于更特定的样式。

```
.content .primary {
    width: 670px; /* width increased and padding removed */
    float: right;
    display: inline;
}

.content .secondary {
    width: 230px;
    float: left;
    display: inline;
}

.content .primary .primary {
    width: 400px;
    float: left;
    display: inline;
}

.content .primary .secondary {
    width: 230px;
    padding-right: 20px; /* padding applied here instead */
    float: right;
    display: inline;
}
```

这样就形成了一个漂亮的三列布局（见图8-9）。

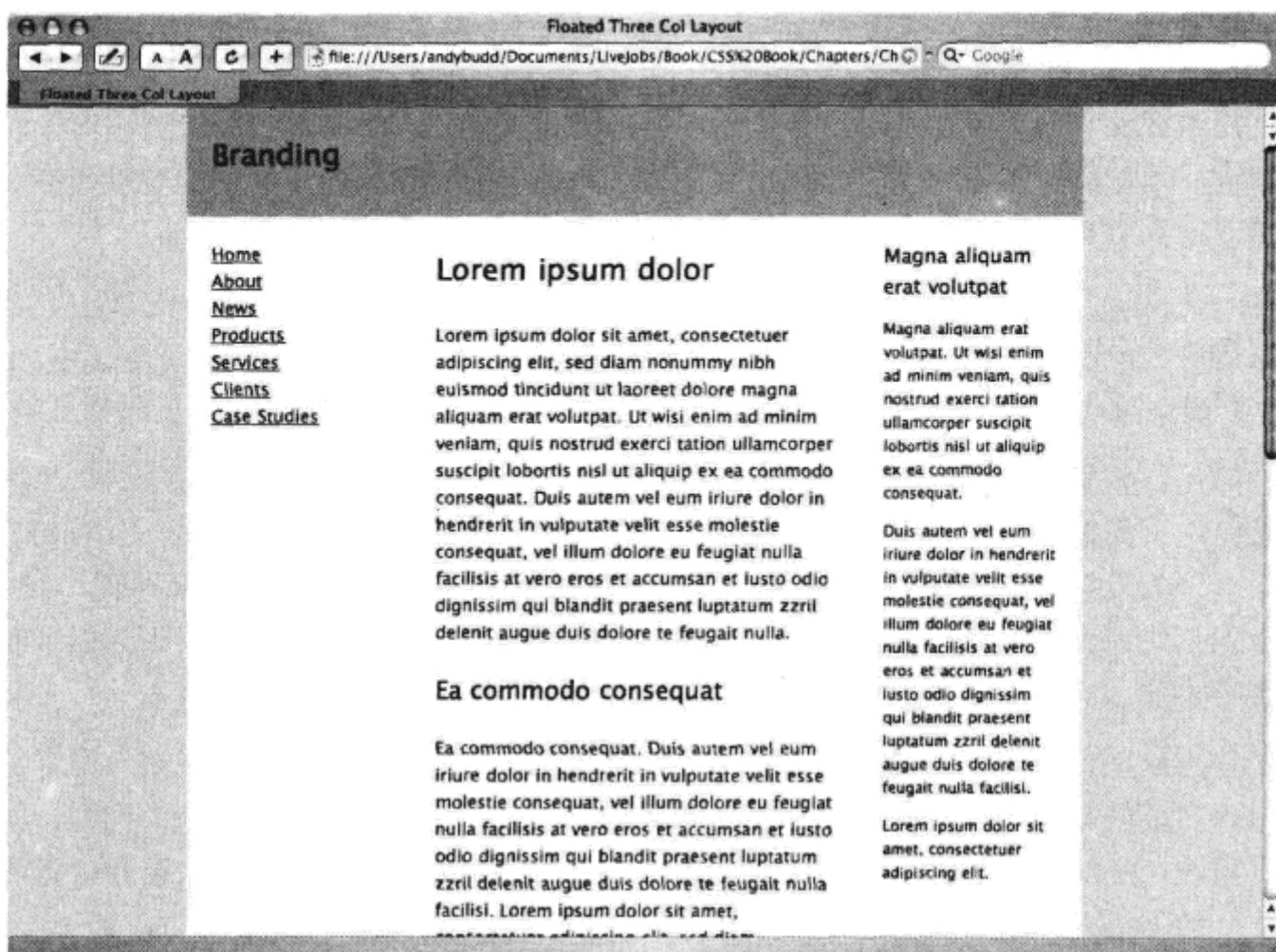


图8-9 使用浮动的三列布局

## 8.4 固定宽度、流式和弹性布局

到目前为止，所有示例中的宽度都以像素为单位。这种布局类型称为固定宽度的布局。固定宽度的布局非常常见，因为它们使开发人员对布局和定位有更大的控制能力。如果将设计的宽度设置为960像素，它就总是960像素宽。这样的话，如果想让一个品牌图像横跨设计的顶部，那么这个图像需要宽为960像素。知道了每个元素的精确宽度，就能够对它们进行精确的布局，而且知道所有东西在什么地方。这使固定宽度的布局成为最简便、最常用的布局方法。

但是，固定宽度的布局也有缺点。首先，因为它们是固定的，所以无论窗口的尺寸有多大，它们的尺寸总是不变。因此，它们无法充分利用可用空间。在高分辨率的屏幕上，为分辨率为1024×760的屏幕创建的设计会缩小并且出现在屏幕的中间。反之，为分辨率为1024×760的屏幕创建的设计在低分辨率的屏幕上会导致水平滚动。随着屏幕尺寸范围越来越大，固定宽度设计越来越不适应灵活的Web。因此，它们常常被认为是糟糕的权宜之计。

固定宽度设计的另一个问题是关于行长和文本易读性的。固定宽度的布局往往适合于浏览器默认文本字号。但是，只要将文本字号增加几级，边栏就会挤满空间并且行长太短，阅读起来不舒服。

为了解决这些问题，可以使用流式布局或弹性布局替代固定宽度的布局。

### 8.4.1 流式布局

在使用流式布局时，尺寸是用百分数而不是像素设置的。这使流式布局能够相对于浏览器窗口进行伸缩。随着浏览器窗口变大，列也会变宽。相反，随着窗口变小，列的宽度也减小。流式布局可以非常高效地使用空间，最好的流式布局甚至不会引起用户的注意。

但是，流式布局也不是没有问题的。在窗口宽度较小的时候，行变得非常窄，很难阅读。在多列布局中尤其如此。因此，有必要添加以像素或em为单位的min-width，从而防止布局变得太窄。但是，如果min-width设置得太大，流式设计也会遇到与固定宽度布局相同的限制。

与之相反，如果设计横跨浏览器窗口整个，那么行就变得太长，也很难阅读。可以采取几个措施来避免这个问题。首先，不要横跨浏览器而是让容器只跨越宽度的一部分，比如85%。还可以考虑用百分数设置内边距和外边距。这样的话，内边距和外边距的宽度将随着窗口尺寸而变，可以防止列太快地变得过宽。最后，对于非常严重的情况，也可以选择为容器设置最大宽度，以防止内容在大显示器上变得过宽。

可以使用这些技术将前面的固定宽度的三列布局转换为流式三列布局。首先，将容器宽度设置为窗口总宽度的百分数。大多数人根据自己屏幕上的效果选择尺寸，这很好。但是，如果希望更精确一点儿，可以通过查看浏览器统计数据确定最常用的窗口尺寸，然后根据固定宽度版本与这个尺寸的比例，选择容器百分数。实现这个目的的一个好工具是Liquid Fold (<http://liquidfold.net/>)。例如，如果设计者使用的宽度为960像素，而大多数用户的浏览器窗口设置为1250像素，那么使用的百分数是 $(960 \div 1250) \times 100 = 76.8\%$ 。

接下来，以容器宽度的百分数形式设置主要内容区域和次要内容区域的宽度。在前面的示例中，主要内容div的宽度是670像素。因为总宽度是920像素，计算结果为72.82%。以相似的方法计算出次要内容div的宽度是25%。这在导航元素和容器之间留出2.18%的隔离带，从而应对任何取整错误和宽度差异。

```
.wrapper {  
    width: 76.8%;  
    margin: 0 auto;  
    text-align: left;  
}  
  
.content .primary {  
    width: 72.82%;  
    float: right;  
    display: inline;  
}  
  
.content .secondary {  
    width: 25%;  
    float: left;  
    display: inline;  
}
```

然后需要设置主内容区域中列的宽度。这需要一点儿技巧，因为内容div的宽度依据的是主内容元素的宽度，而不是整个容器。现在，主div的宽度是400像素，即父元素的59.7%。以相似的方法计算出次要div的宽度是34.33%。最后，还需要20像素的槽，即父元素的2.63%。

```
.content .primary .primary {
    width: 59.7%;
    float: left;
    display: inline;
}

.content .primary .secondary {
    width: 34.33%;
    padding-right: 2.63%;
    float: right;
    display: inline;
}
```

这会产生一个最适合1250像素窗口的流式布局，但是在屏幕分辨率更大或更小时阅读起来也比较舒服（见图8-10）。

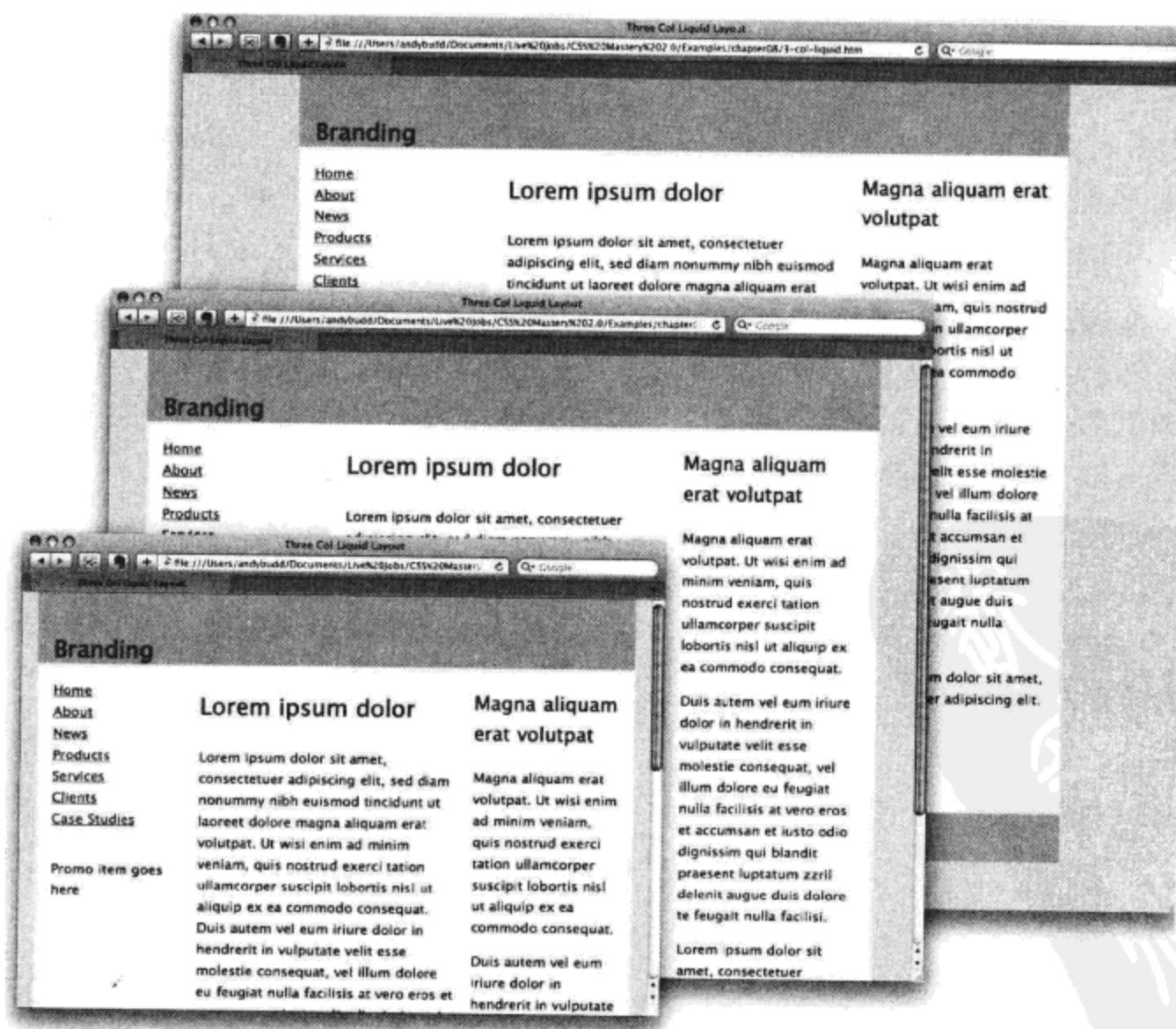


图8-10 三列流式布局在屏幕分辨率为800×600、1024×768和1152×900时的效果

因为这个布局会恰当地伸缩，所以不需要添加max-width属性。但是，为确保文本行的长度适合阅读，最好添加以em为单位的max-width。对于比较小的窗口尺寸，这个布局会有点儿挤，所以还添加以em为单位的min-width。

```
.wrapper {  
    width: 76.8%;  
    margin: 0 auto;  
    text-align: left;  
    max-width: 125em;  
    min-width: 62em;  
}
```

现在你应该看到了一个漂亮灵活的流式布局。

#### 8.4.2 弹性布局

虽然流式布局可以充分利用可用空间，但是在大分辨率显示器上，行仍然会过长，让用户不舒服。相反，在窄窗口中或者在增加文本字号时，行会变得非常短，内容很零碎。对于这个问题，弹性布局可能是一种解决方案。

弹性布局相对于字号（而不是浏览器宽度）来设置元素的宽度。以em为单位设置宽度，可以确保在字号增加时整个布局随之扩大。这可以将行长保持在可阅读的范围，对于视力弱或有认知障碍的人尤其有用。

与其他布局技术一样，弹性布局也有自己的问题。弹性布局的一些问题与固定宽度布局相同，比如不能充分利用可用空间。另外，因为在文本字号增加时整个布局会加大，所以弹性布局会变得比浏览器窗口宽，导致水平滚动条出现。为了防止这种情况，可能需要在容器div上添加100%的max-width。IE 6和更低版本不支持max-width，但是较新的版本都支持它。如果要在IE 6中支持max-width，可以使用JavaScript。

创建弹性布局比创建流式布局容易得多，因为所有HTML元素的相对位置基本上一直都是相同的，它们仅仅是同时随文本字号而增大。将固定宽度布局转换为弹性布局是相对简单的任务。技巧是要设置基字号，让1em大致相当于10像素。

大多数浏览器上的默认字号是16像素，10像素大约是16像素的62.5%，所以在主体上将字号设置为62.5%就可以了：

```
body {  
    font-size: 62.5%;  
    text-align: center;  
}
```

因为在使用默认字号时1em现在相当于10像素，所以可以相当轻松地将固定宽度布局转换为弹性布局。在本书的前一个版本中，我建议以em为单位设置所有宽度。但是，我的同事和技术审稿人Natalie Downe建议内部宽度仍然使用百分数，只以em为单位设置容器的宽度。这样的话，内部宽度仍然是相对于字号的。这样就可以方便地修改布局的总尺寸，不必修改每个元素的宽度，这种解决方案更灵活、更容易维护。

```
.wrapper {  
    width: 92em;  
    max-width: 95%;  
    margin: 0 auto;  
    text-align: left;  
}  
  
.content .primary {  
    width: 72.82%;  
    float: right;  
    display: inline;  
}  
  
.content .secondary {  
    width: 25%;  
    float: left;  
    display: inline;  
}  
  
.content .primary .primary {  
    width: 59.7%;  
    float: left;  
    display: inline;  
}  
  
.content .primary .secondary {  
    width: 34.33%;  
    padding-right: 2em;  
    float: right;  
    display: inline;  
}
```

在使用常规文本字号时，这些代码产生的布局与固定宽度布局看起来一样（见图8-11），但是它会随着文本字号的增加而漂亮地增大（见图8-12）。

由于页面缩放在现代浏览器中越来越流行了，一些人开始觉得不需要弹性布局了。但是，除非所有浏览器都默认支持页面缩放，否则，对于老式浏览器，可能仍然需要考虑使用弹性布局。



图8-11 使用默认文本字号时的弹性布局

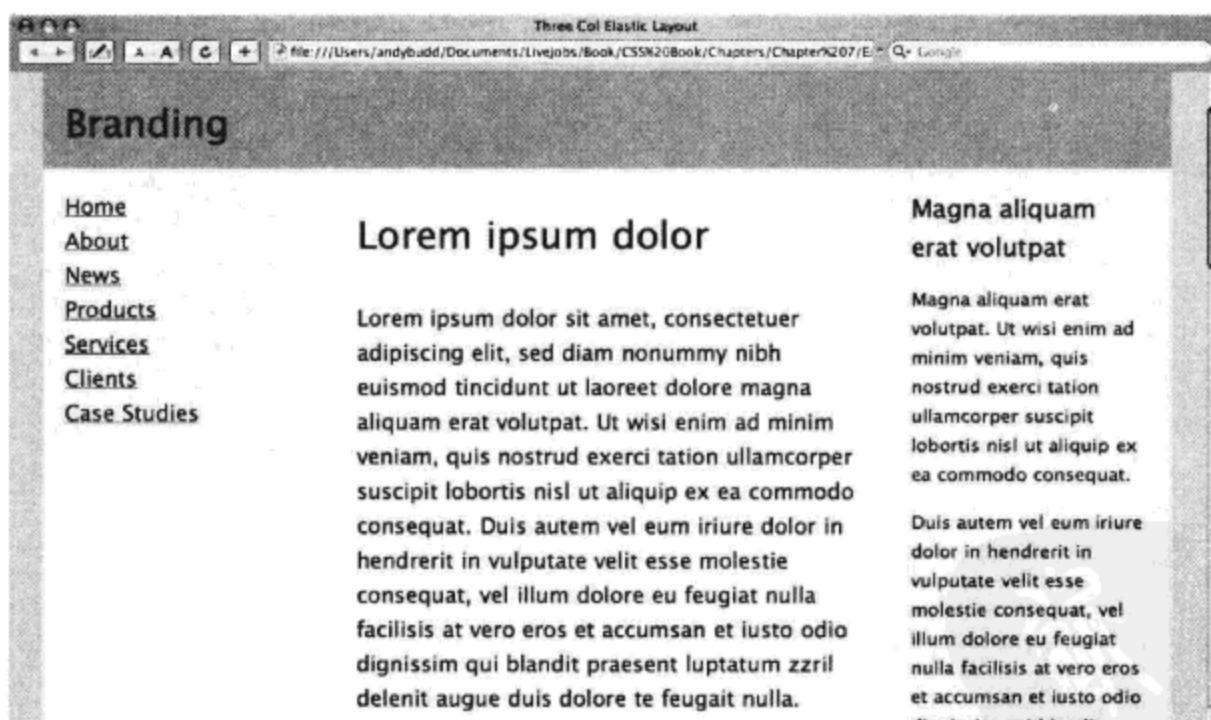


图8-12 文本字号增加几次之后的弹性布局

### 8.4.3 流式和弹性图像

如果选择使用流式或弹性布局，那么固定宽度的图像就会对设计产生强烈的影响。当布局的宽度减小时，图像会相对它移动，可能对彼此产生消极影响。图像会以自然的最小宽度显示，从而防止某些元素的尺寸减小。有些图像会超出包含它们的元素，从而破坏精心调整过的设计。增加布局的宽度也会产生戏剧性的结果，形成空隙过大、不平衡的设计。但是不要担心，有几个办

法可以避免这些问题。

对于需要跨过大区域的图像，比如站点页眉或品牌区域中的图像，可以考虑使用背景图像而不是图像元素。随着branding元素的伸缩，背景图像或多或少会露出来一些：

```
#branding {  
    height: 171px;  
    background: url(/img/branding.png) no-repeat left top;  
}  
  
<div id="branding"></div>
```

如果图像需要用作页面上的图像元素，那么将容器元素的宽度设置为100%并且将overflow属性设置为hidden。这样的话，图像右边会被截短，使它适合branding元素，而不会随着布局伸缩：

```
#branding {  
    width: 100%;  
    overflow: hidden;  
}  
  
<div id="branding">  
      
</div>
```

对于常规内容图像，你可能希望它们垂直和水平伸缩以避免被剪切。为此，可以将图像元素添加到没有设置任何尺寸的页面上。然后设置图像的百分数宽度，并且添加与图像宽度相同的max-width以避免像素失真（pixelization）。

例如，假设希望创建一种新闻稿样式，在左边是窄的图像列，右边是比较大的文本列。图像的宽度需要大约是包含它的框的四分之一，文本占据余下的空间。为此，只需将图像的宽度设置为25%，然后将max-width设置为图像的尺寸（在这个示例中是200像素宽）：

```
.news img {  
    width: 25%;  
    max-width: 200px;  
    float: left;  
    display: inline;  
    padding: 2%;  
}  
  
.news p {  
    width: 68%;  
    float: right;  
    display: inline;  
    padding: 2% 2% 2% 0;  
}
```

随着news元素的扩展或收缩，图像和文本段落也会扩展或收缩，从而保持视觉上的平衡（见图8-13）。但是，在符合标准的浏览器上，图像不会超过它的实际尺寸。



图8-13 给图像设置百分数宽度，使它们能够相对于周围元素漂亮地伸缩

## 8.5 faux列

你可能已经注意到了，所有这些布局上的导航元素和次要内容区域都有一个浅灰色的背景。理想情况下，背景应该拉长到整个布局的最大高度，从而形成列的效果。但是，因为导航元素和次要内容区域没有扩展到最大高度，所以它们的背景也不会拉长。

为了创建列的效果，需要创建一个伪列。方法是在一个占据布局最大高度的元素（比如一个容器div）上应用重复的背景图像。Dan Cederholm创造出“faux列”这个术语来描述这种技术。

对于固定宽度的两列布局，只需在容器元素上应用一个垂直重复的背景图像，其宽度与导航区域相同（见图8-14）：

```
#wrapper {
    background: #fff url(/img/nav-bg-fixed.gif) repeat-y left top;
}
```

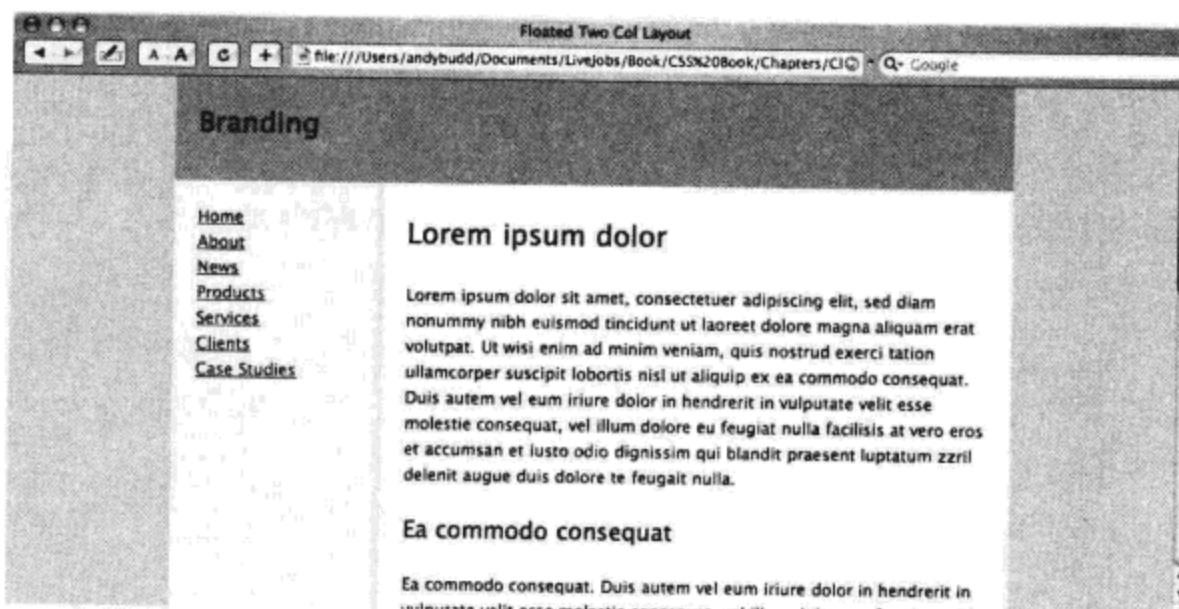


图8-14 固定宽度的faux列

对于固定宽度的三列布局，可以使用相似的方法。但是，这一次重复的背景图像需要横跨整个容器，其中包含两列（见图8-15）。按照与前面一样的方式应用这个图像，就会形成漂亮的faux三列效果（见图8-16）。



图8-15 用来创建faux三列效果的背景图像

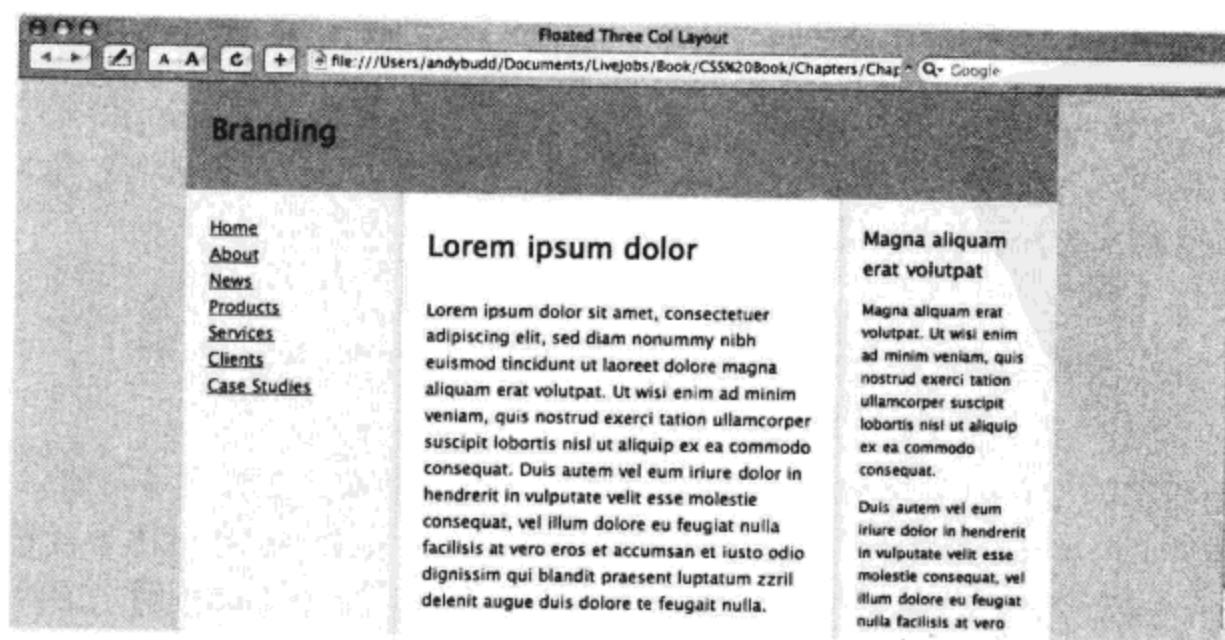


图8-16 faux三列效果

为固定宽度的设计创建faux列非常容易，因为我们知道列的尺寸和位置。为流式布局创建faux列就有点儿复杂了，因为列的尺寸和位置随浏览器窗口缩放而变化。建立流式faux列的技巧是按百分比对背景图像进行定位。

如果使用像素设置背景的位置，那么图像的左上角会定位在距离元素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。所以，如果将垂直和水平位置设置为20%，那么实际上会把图像上距离左上角20%的点定位到父元素上距离左上角20%的位置（见图8-17）。

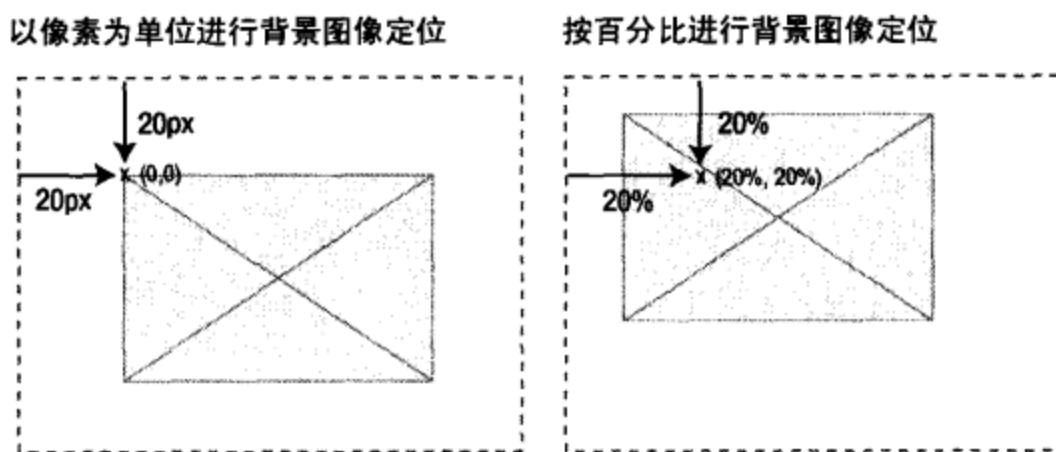


图8-17 在按百分比进行定位时，使用图像上的对应位置

按百分比定位背景图像非常有用，因为这样可以创建水平比例与布局相同的背景图像，然后把背景图像定位到希望列出现的地方。

为了给次要内容区域创建faux列，首先创建一个非常宽的faux图像。在这个示例中，我创建的图像有4000像素宽，5像素高。接下来，需要在背景图像上创建作为faux列的区域。次要内容区域已经设置为容器宽度的25%，所以需要在背景图像上创建一个宽25%的对应区域。对于4000像素宽的背景图像，图像的faux列部分的宽应该为1000像素。将这个图像保存为GIF，可以确保faux列没有覆盖的区域是透明的。

faux列的右边缘与图像的左边相距25%。次要内容区域的右边缘与容器元素的左边相距25%。这意味着，如果将这个图像作为背景应用于容器元素，并且将水平位置设置为25%，那么faux列的右边缘会正好与次要内容区域的右边缘对齐。

```
.wrapper {
  background: #fff url(/img/secondary-faux-column.gif) repeat-y 25% 0;
}
```

可以使用相似的方法为主内容区域创建背景。这个faux列的左边缘应该与图像左边缘相距72.82%，从而与主内容元素和容器的相对位置匹配。因为容器元素上已经应用了背景图像，所以需要在第一个容器元素内添加第二个容器元素。然后就可以将第二个faux列背景图像应用于这个新的容器元素。

```
.inner-wrapper {
  background: url(/img/primary-faux-column.gif) repeat-y 72.82% 0;
}
```

如果正确地设置了比例，那么应该会形成漂亮的三列流式布局，其中的列背景会拉长到与容器等高（见图8-18）。

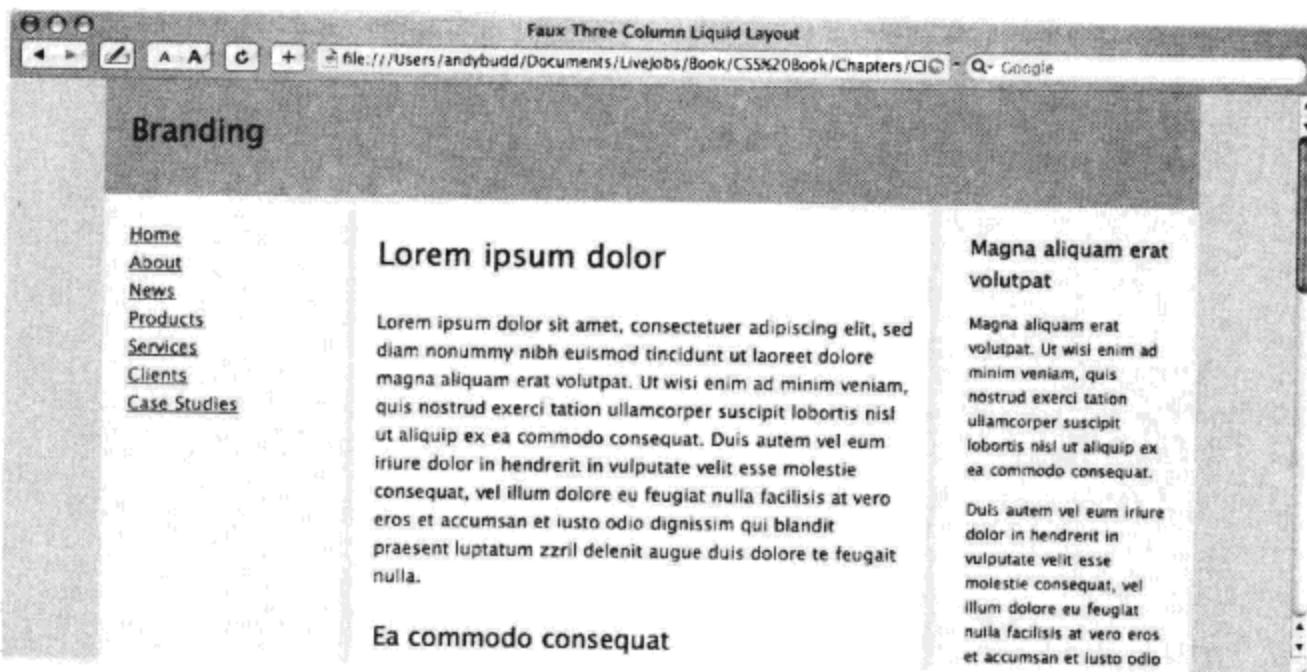


图8-18 faux三列布局

## 8.6 高度相等的列

除了在主布局中创建列，你还可能需要在其他地方创建高度相等的列，比如像图8-19这样。尽管使用表格很容易实现这种效果，但是在CSS中实现时需要一点儿技巧。

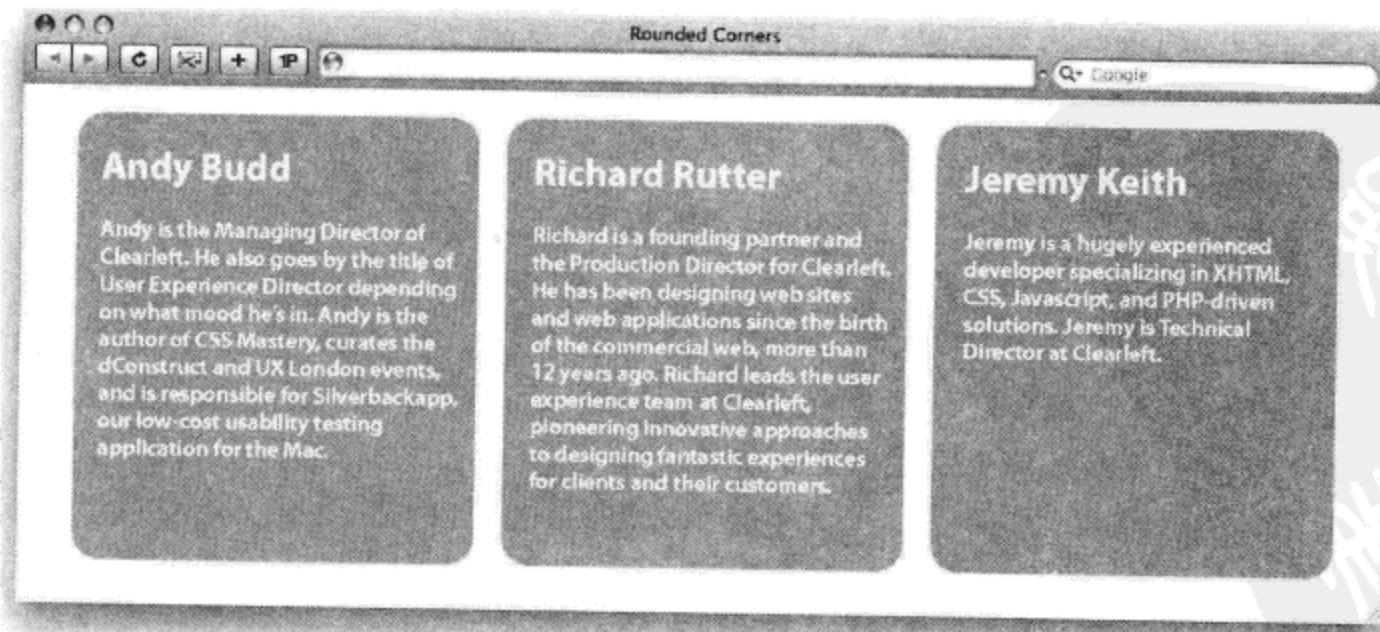


图8-19 3个高度相等的列

先看看标记。

```
<div class="wrapper">
  <div class="box">
    <h1>Andy Budd</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>

  <div class="box">
    <h1>Richard Rutter</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>

  <div class="box">
    <h1>Jeremy Keith</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>
</div>
```

对于这个示例，需要3个div，每个列一个。在每个div中，需要标题、一些内容和一个空的div，这个div作为底角的钩子。然后，把这3个div放在容器div中，我们将使用容器div限制高度。现在开始给框设置样式。

```
.wrapper {
  width: 100%;
}

.box {
  width: 250px;
  margin-left: 20px;
  float: left;
  display: inline;
  padding: 20px;
  background: #89ac10 url(/img/top.gif) no-repeat left top;
}
```

如图8-20所示，这会产生3个高度不一致的列。

这种技术的关键是给每个框设置大的底内边距，然后用数值相似的负外边距消除这个高度。这会导致每个列溢出容器元素（见图8-21）。如果把容器的overflow属性设置为hidden，列就在最高点被裁切。在这个示例中，给每个元素设置520像素的底内边距和500像素的底外边距。20像素的差值在每个框的底部形成可见的内边距。

```
.wrapper {
  width: 100%;
  overflow: hidden;
```

```

}
.box {
  width: 250px;
  padding-left: 20px;
  padding-right: 20px;
  padding-top: 20px;
  padding-bottom: 520px;
  margin-bottom: 500px;
  margin-left: 20px;
  float: left;
  display: inline;
  background: url(/img/top.gif) #89ac10 top left no-repeat;
}

```

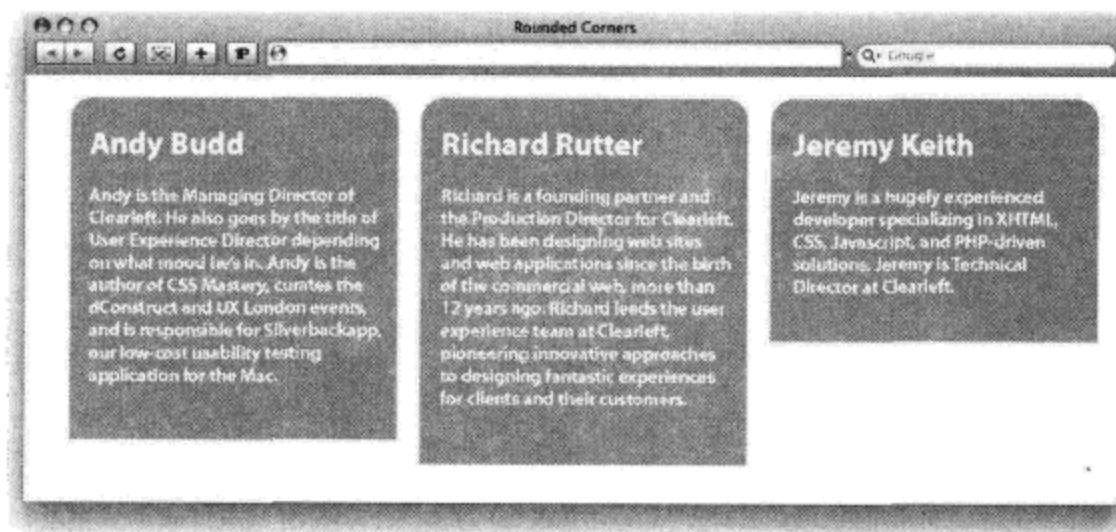


图8-20 在应用主要技术之前的3个列

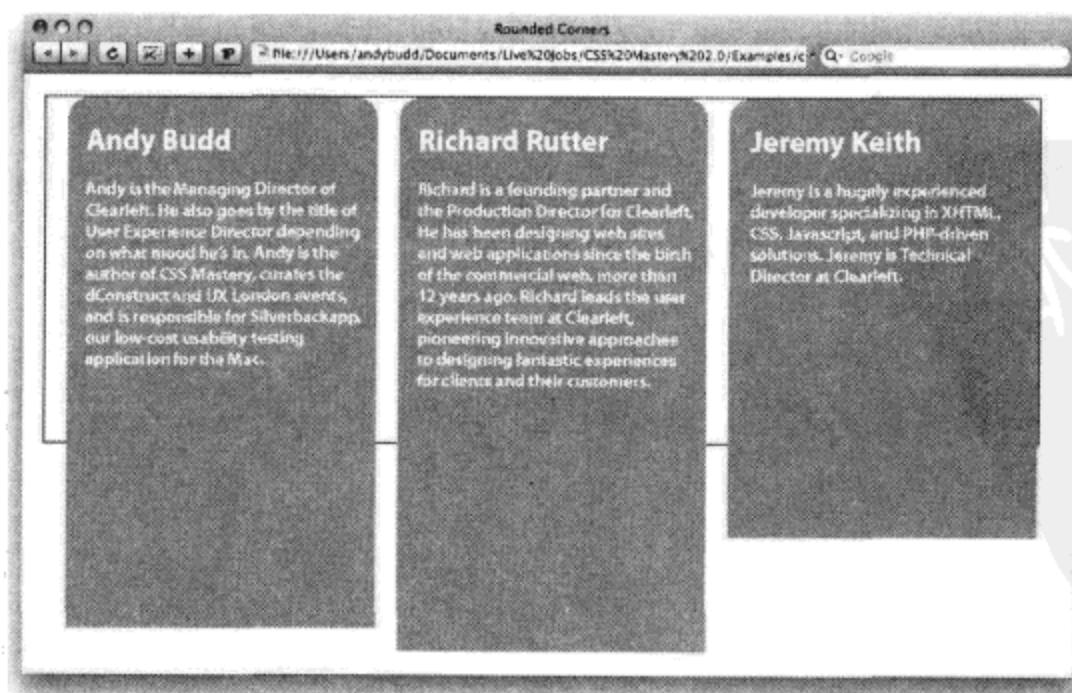


图8-21 红色框显示容器div的边界，可以看出3个列溢出了这个元素

为了把列的底边定位在正确的位置，需要让它们与齐容器元素的底部对齐。为此，首先把容器的position设置为relative。然后把空div的position设置为absolute，把它们的bottom属性设置为零。现在，只需设置正确的宽度和高度，应用底部背景图像。

```
.wrapper {  
    width: 100%;  
    overflow: hidden;  
    position: relative;  
}  
  
.box {  
    width: 250px;  
    padding-left: 20px;  
    padding-right: 20px;  
    padding-top: 20px;  
    padding-bottom: 520px;  
    margin-bottom: 500px;  
    margin-left: 20px;  
    float: left;  
    display: inline;  
    padding: 20px;  
    background: url(/img/top.gif) #89ac10 top left no-repeat;  
}  
  
.bottom {  
    position: absolute;  
    bottom: 0;  
    height: 20px;  
    width: 290px;  
    background: url(/img/bottom.gif) #89ac10 bottom left no-repeat;  
    margin-left: -20px;  
}
```

结果是一个高度与最长列相等的三列布局，见图8-19。

## 8.7 CSS 3 列

CSS 3也可以创建等高文本列，见图8-22。这要通过column-count、column-width和column-gap属性实现。

假设使用的标记如下：

```
<h1>Socrates</h1>  
<div class="col">  
    <p>After philosophizing for a while...</p>  
</div>
```

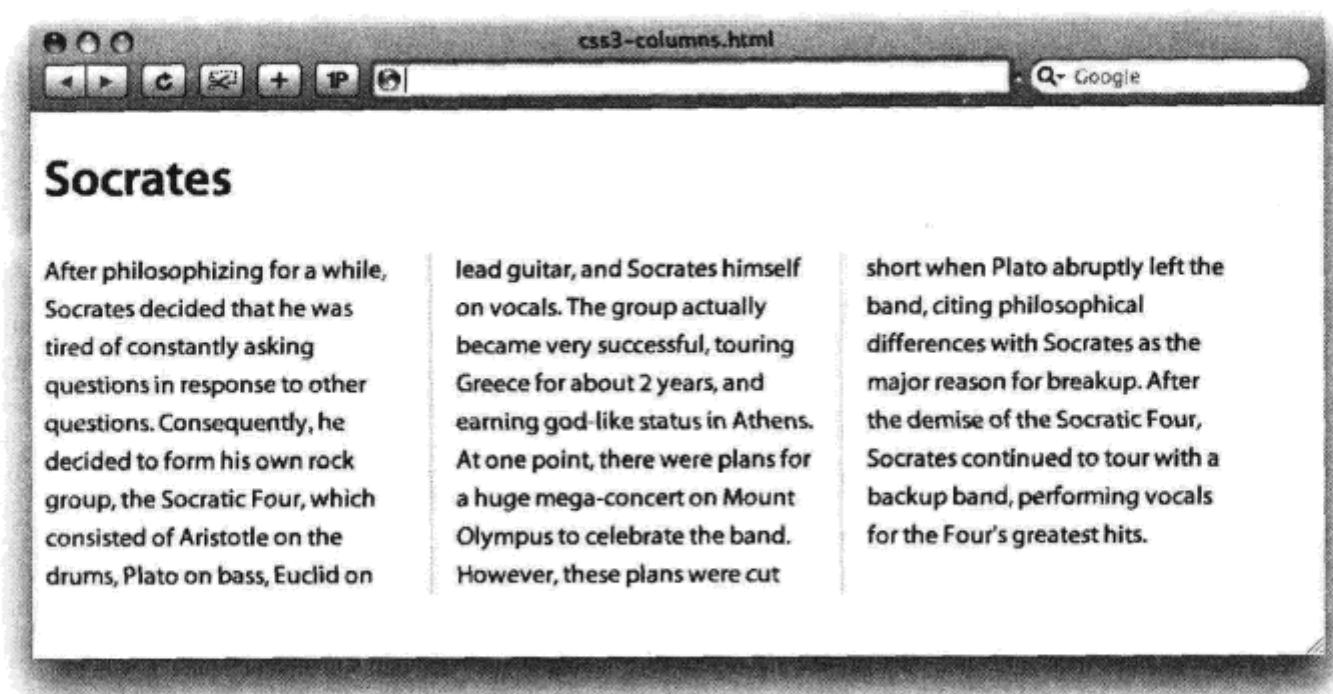


图8-22 使用CSS 3列属性实现的文本列

应用以下规则创建一个三列布局，每个列的宽度为14 em，列之间有2 em的间距。CSS列的优点之一是在可用空间小于已定义列的宽度时的处理方式。列不会像使用浮动时那样回绕，而是会减少列数。因此，如果空间不够显示三列，就会减少到两列。

```
.col {
    -moz-column-count: 3;
    -moz-column-width: 14em;
    -moz-column-gap: 2em;
    -moz-column-rule: 1px solid #ccc;
    -webkit-column-count: 3;
    -webkit-column-width: 14em;
    -webkit-column-gap: 2em;
    -webkit-column-rule: 1px solid #ccc;
    column-count: 3;
    column-width: 14em;
    column-gap: 2em;
    column-rule: 1px solid #ccc;
}
```

从前面的代码中可以看出浏览器对CSS列的支持还不广泛。因此，除了常规代码，还需要使用特定于浏览器相关的扩展。

## 8.8 CSS 框架与 CSS 系统

在编程领域，Rails或Django等框架采用的是常见的Web开发模式（比如在数据库中添加记录），并且把这些模式抽象成简单的可重用组件。这让开发人员可以简便地开发相当复杂的应用

程序，不需要从头开发这些功能。与独立函数库不同，框架的功能往往很全面。框架的抽象程度很高，所以开发人员不需要了解底层语言（但是最好有所了解），也可以构建完整的应用程序。

最近几年，出现了所谓的CSS框架。这些框架的目标是简化CSS的使用，帮助用户方便地创建各种常用的布局，而不需要编辑底层CSS。这些框架鼓励开发人员使用一系列标记模式和命名约定，然后在幕后管理布局。最受欢迎的3个框架是YUI Grids、Blueprint和960（见图8-23），但是还有其他一些框架。

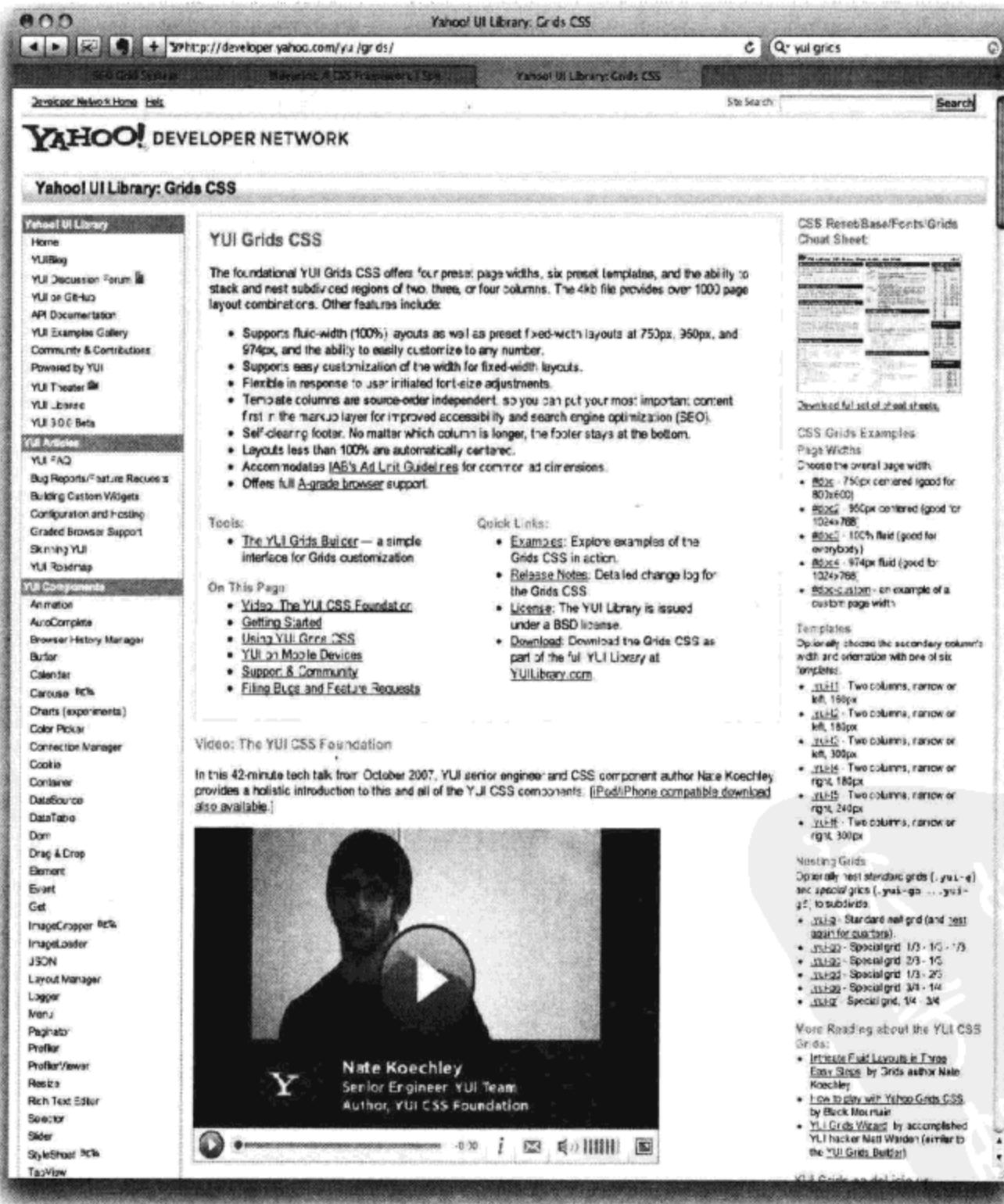


图8-23 YUI、Blueprint和960网站

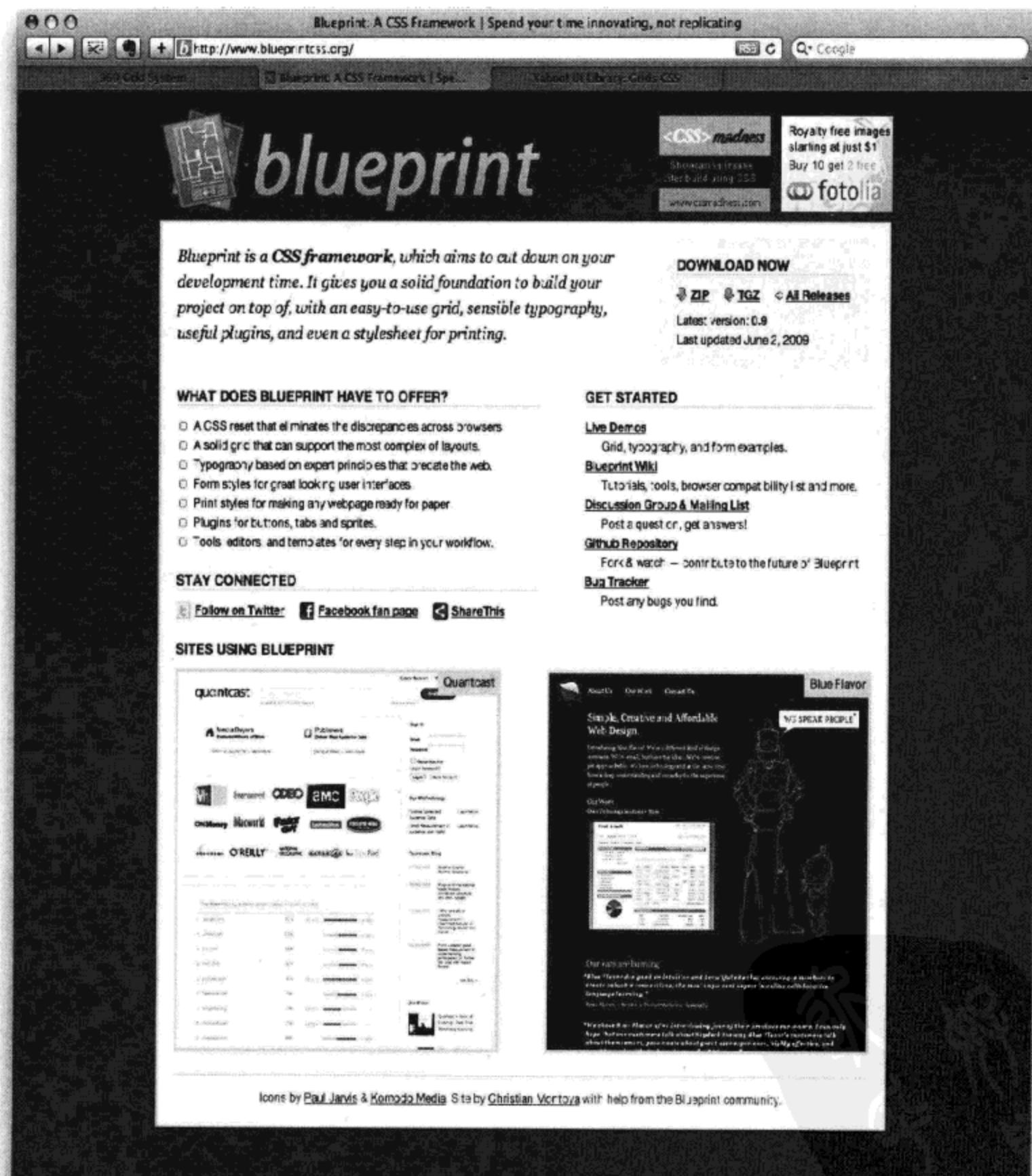


图8-23 (续)



图8-23（续）

这些框架提供了许多有助于提高生产力的特性，包括全局`reset`样式、全站点范围的排版处理和一致的表单处理，大多数项目都需要这些特性。但是，框架也会改变编写标记的方式，破坏表现与意义的分离。例如，Blueprint框架中使用`column`和`column span`这样的标记，这显然是表现性的。

```
<div class="column span-24">
  <!-- header -->
</div>
<div class="column span-4">
  <!-- left sidebar -->
</div>
<div class="column span-16">
  <!-- main content -->
</div>
<div class="column span-4 last">
  <!-- right sidebar -->
</div>
```

如果使用框架控制布局，开发人员就不得不使用表现性的标记，这与基于表格的设计很相似。实际上，可以说CSS框架还不如表格，因为表格具有纯粹的表现性标记，不需要下载额外的CSS。框架还迫使开发人员除了学习底层语言之外还要学习框架。这常常是不现实的，开发人员往往对这两方面都是一知半解。

框架的另一个缺点是，它们要求在设计中必须使用特定的网格结构。如果框架定义的宽度和外边距正好适合你的设计，那没问题。但是，正如编程框架不应该决定网站的用户体验，CSS框架也不应该决定站点的设计。如果选用某个框架，你可能会在每个项目中都使用它，因此思维方式很可能逐渐僵化。正如俗话所说的，如果你只有一把锤子，那么什么东西看起来都像是钉子。

如果了解了框架的起源，就会明白为什么会出现这些问题。大多数框架是为了在特定的站点（比如Yahoo或Laurence Kansas Journal）上使用而创建的，而不是作为适合任何设计的灵活的布局系统从头设计的。这些站点已经有定义良好的网格结构和样式指南，所以开发人员知道每个新页面都采用相同的模式。随着时间的推移，开发人员发现其他人也可能需要使用这些系统，所以对这些系统进行一般化并向公众发布。但是，在这些框架的设计中仍然可以看出原来站点的痕迹。

那么，如何能够享有CSS框架的生产力优势，同时避免其显著缺点呢？这就需要CSS系统。CSS系统实际上是一个工具箱，其中包含可重用的样式和标记模式，可以用它们开发站点专用的框架。这个工具箱可以包含全局reset、排版样式和表单处理，以及登录表单、日历表格和导航列表等常用HTML组件的标记模式。然后，可以使用从本书学到的技术为客户开发作为定制框架使用的系统，在其中包含客户需要的各种布局选项。这个过程最初需要多做一些工作，但是它能够提供CSS框架的优点，同时避免其缺点。

## 8.9 小结

本章学习了如何使用浮动创建简单的两列和三列固定宽度的布局，然后学习了如何将这些布局轻松地转换为流式和弹性布局，还学习了与这些布局相关的一些问题，以及如何通过设置以em或像素为单位的最大宽度解决这些问题。你还看到了如何使用垂直重复的背景图像在固定宽度和

流式布局上产生高度为最大值的列效果。本章讨论了用来创建基于CSS的布局的一些技术，但是，这类技术很多，足够另写一本书了。最后，我们了解了CSS框架固有的一些缺点以及开发自己的CSS系统的重要性。

在使用CSS布局时，开发人员面对的一个大问题是浏览器不一致。为了解决浏览器显示问题，需要充分了解各种bug及其修复方案。在下一章中，我们将学习一些比较有名的浏览器bug以及CSS调试的基础知识。





## 第9章

# bug和修复bug

与许多编程语言相比，CSS是一种相对容易学习的语言。它的语法简单明了，而且由于它的表现本质，开发人员并不需要处理复杂的逻辑。但是，当在不同的浏览器上测试代码时，困难就会出现了。浏览器bug和不一致的显示方式是大多数CSS开发人员面临的主要难题。你的设计在一种浏览器上看起来很好，但是在另一种浏览器上布局可能会支离破碎。

“CSS难以掌握”的误解并不来源于语言本身，而是由于为了让站点在老式浏览器上工作正常需要采取很多措施。关于bug的信息很难找到，而且又缺乏文档记录，因此bug常常被误解。许多人把hack看做魔法子弹，认为它们就像神秘的咒语一样，当应用于代码时，它们会神奇地修复破碎的布局。hack的确是有力的工具，但是需要谨慎地应用，而且一般是作为最后的手段来使用。更为重要的技能是跟踪、隔离和识别bug。只有在知道bug是怎么回事之后，才能想办法对付它。

在本章中，你将学习：

- 如何跟踪CSS bug；
- 神秘的hasLayout属性；
- hack和过滤器；
- 最常见的浏览器bug及其修复方法；
- 分级浏览器支持。

### 9.1 捕捉 bug

我们都知道浏览器是有bug的，而且有的浏览器多，有的浏览器少。当CSS开发人员在自己的代码中遇到问题时，他们往往马上就认为这是浏览器bug，并且寻求hack或其他方法。但是浏

览器bug并没有一般人认为的那么常见。最常见的CSS问题并非来源于浏览器bug，而是来源于对CSS规范的理解不完整。为了避免这些问题，在处理CSS bug时最好假设是自己做错了什么事。只有在确定这不是自己的错之后，才应该考虑问题是否是浏览器bug的结果。

## 常见的 CSS 问题

最简单的一些CSS问题是由于代码中的打字和语法错误造成的。例如，在声明末尾忘了加分号，或者在应该输入`font-family`时输入了`font-face`。解决这种问题的简单方法是，选择一个包含语法突出显示和代码补全功能的CSS编辑器，比如SKEdit或CSS Edit。这些特性有助于防止基本错误，但是不能替代适当的检查。通过CSS Validator (<http://jigsaw.w3.org/css-validator/>) 等服务运行代码可以突出显示所有语法错误，并且显示错误所在的行和对每个错误的简短描述（见图9-1）。

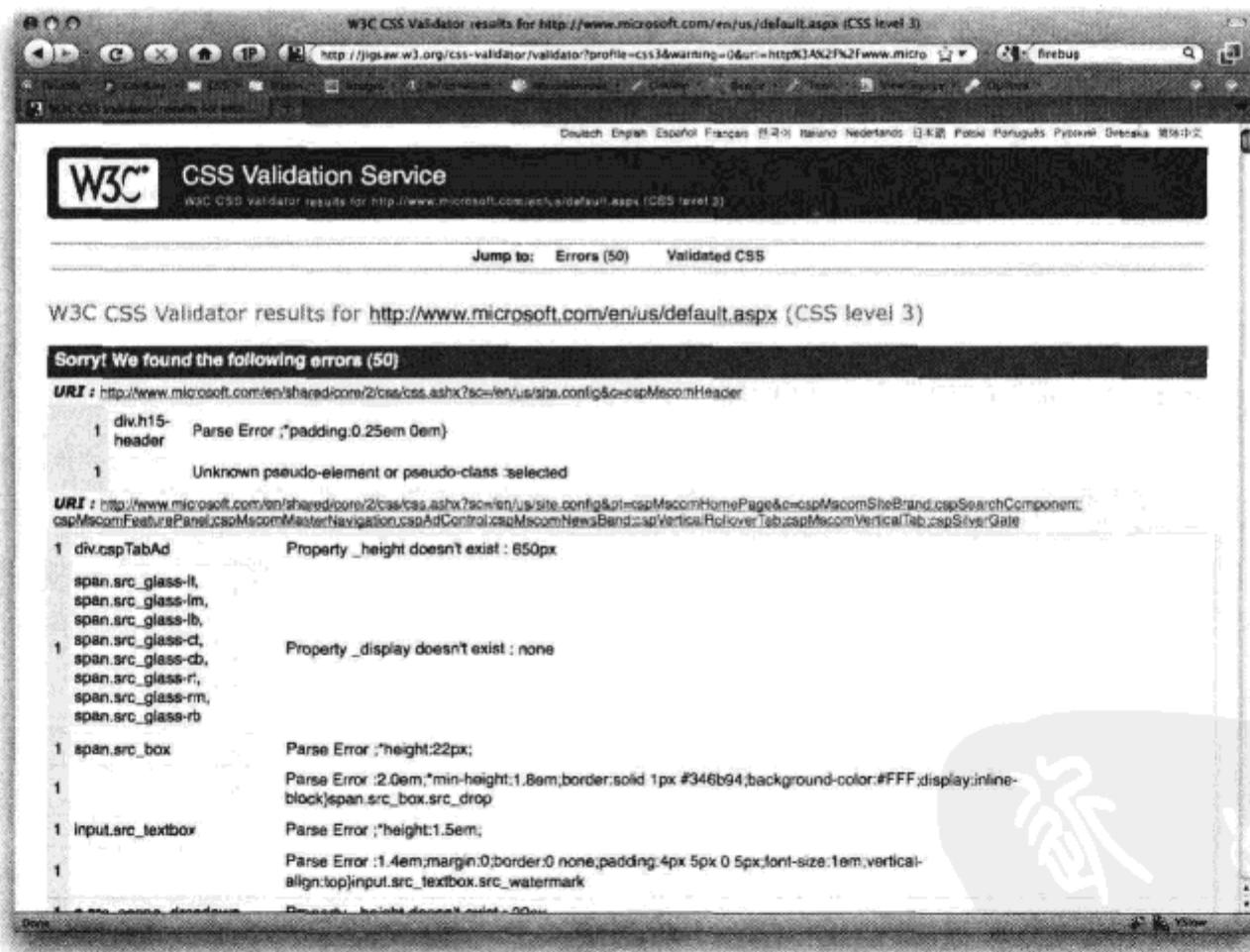


图9-1 通过CSS Validator检查Microsoft网站

Firefox Web Developer Toolbar扩展 (<https://addons.mozilla.org/en-US/firefox/addon/60>) 包含HTML和CSS验证器的在线版本的链接。还有受欢迎的HTML Validator for Firefox (<http://users.skynet.be/mgueury/mozilla/>)。

在检验自己的HTML和CSS时，你可能会看到页面充满了错误。初看上去这非常让人沮丧，

但是不必担心。这些错误中的大多数将是一两个实际错误的结果。如果修复了第一个错误并且重新进行检验，可能会看到原来的许多错误消失了。这样修改几次之后，代码很快就会没有错误了。

请记住，验证器只是一个自动化工具，并不是完全可靠的。被报告的验证器bug数量正在增加，所以如果你认为某些东西是对的，但是验证器却认为不对，那么一定要检查最新的CSS规范。例如，到撰写本书时，CSS验证器仍然认为厂商特有的扩展（如`moz-border-radius`）是错的，但是CSS规范允许这些扩展。如果有疑问，应该使用CSS 3分析工具检查代码，然后查阅规范。

### 1. 特殊性和分类次序的问题

除了语法错误之外，比较常见的一个问题与特殊性和分类次序有关。在将一个规则应用于一个元素时，如果发现没有任何效果，这时往往存在特殊性问题。可以应用其他规则而且它们会工作正常，但是某些规则似乎不起作用。在这些情况下，问题往往是已经在文档的其他地方使用更特殊的选择器为这个元素定义了规则。

在下面的示例中，CSS开发人员将内容区域中所有段落的背景颜色设置为白色。但是，他们希望`intro`段落是橙色的，因此在`intro`段落上直接应用了这个规则：

```
.content p {  
    background-color: white;  
}  
  
.intro {  
    background-color: orange;  
}
```

如果在浏览器中测试这段代码，会看到`intro`段落仍然是白色的。这是因为与选择`intro`段落的选择器相比，选择内容区域中所有段落的选择器的特殊性更强。为了实现想要的结果，需要让选择`intro`段落的选择器更特殊。在这种情况下，最好的方法是在`intro`段落选择器的开头添加内容元素的`class`：

```
.content p {  
    background-color: white;  
}  
  
.content.intro {  
    background-color: orange;  
}
```

不要随便添加更特殊的选择器，因为这可能会给代码的其他部分带来特殊性问题。更好的方法往往是删除额外的选择器，让它们尽可能一般化，只在需要细粒度的控制时添加更特殊的选择器。

正如第1章提到的，用于Firefox的Firebug附加组件（<https://addons.mozilla.org/en-US/firefox/>

addon/1843) 是调试CSS的好工具。它有许多有用的特性，其中之一是可以通过查看元素了解哪些CSS样式被覆盖了。对于在样式表中其他地方被覆盖的样式，它会加上删除线，见图9-2。

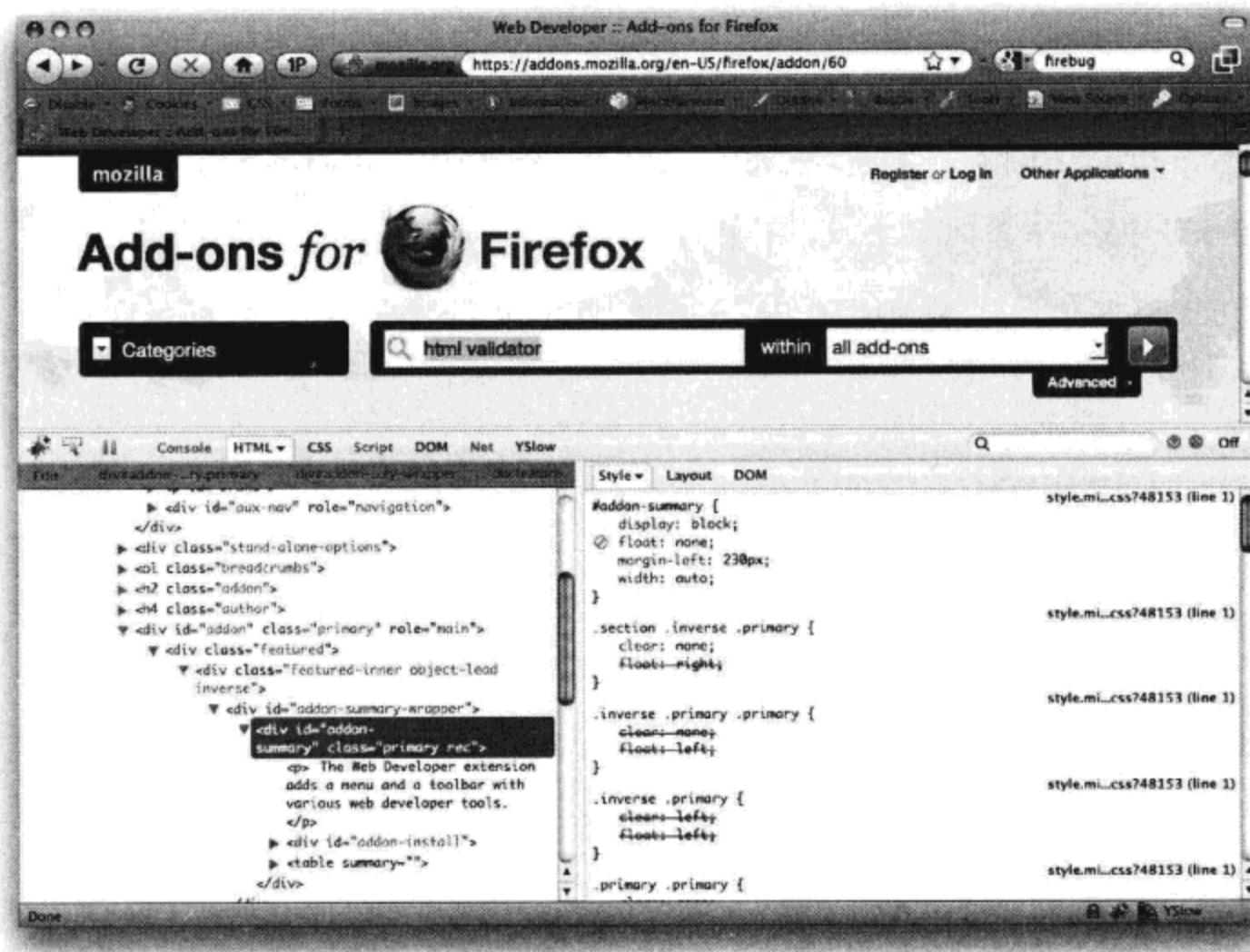


图9-2 如果样式在样式表中其他地方被覆盖了，就会加上删除线

## 2. 外边距叠加的问题

外边距叠加（见第3章）是另一个如果误解就会导致许多麻烦的CSS特性。以div元素内嵌套的一个段落为例：

```
<div id="box">
  <p>This paragraph has a 20px margin.</p>
</div>
```

div框设置了10像素的外边距，段落设置了20像素的外边距：

```
#box {
  margin: 10px;
  background-color: #d5d5d5;
}

p {
```

```
margin: 20px;
background-color:#6699FF;
}
```

你会自然地认为产生的样式会像图9-3那样，在段落和div之间有20像素的外边距，在div外边围绕着10像素的外边距。

但是，产生的样式实际上是像图9-4这样。

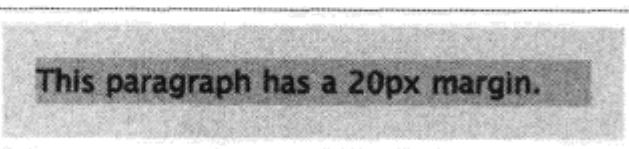


图9-3 对前面样式的预期

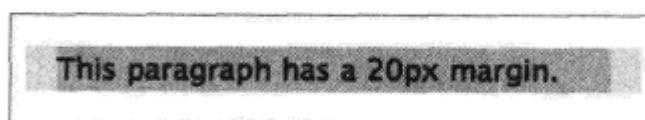


图9-4 样式的实际效果

这里发生了两个情况。首先，段落的20像素顶外边距和底外边距与div的10像素外边距叠加，形成一个20像素的垂直外边距。其次，这些外边距不是被div包围，而是突出到div的顶部和底部的外边。出现这种情况是由于具有块级子元素的元素计算其高度的方式造成的。

如果元素没有垂直边框或内边距，那么它的高度就是它包含的子元素的顶部和底部边框边缘之间的距离。因此，包含的子元素的顶部和底部外边距就突出到容器元素的外边。有一个简单的解决方案：通过添加一个垂直边框或内边距，外边距就不再叠加了，而且元素的高度就是它包含的子元素的顶部和底部外边距边缘之间的距离。

为了让前面的示例看起来像图9-3，只需在div周围添加内边距或边框：

```
#box {
  margin: 10px;
  padding: 1px;
  background-color:#d5d5d5;
}

p {
  margin: 20px;
  background-color:#6699FF;
}
```

外边距叠加的大多数问题可以通过添加一点内边距或与元素背景颜色相同的小边框来修复。

用于了解元素的相互作用的一种好工具是Web Developer Toolbar中的topographic视图。启用这个选项，就可以根据元素在文档中的位置给每个元素加上彩色背景。这样就很容易看出元素在文档中的相对位置（见图9-5）。

另一个有用的工具是Firebug中的layout视图（见图9-6），它显示正在检查的元素的各个尺寸。

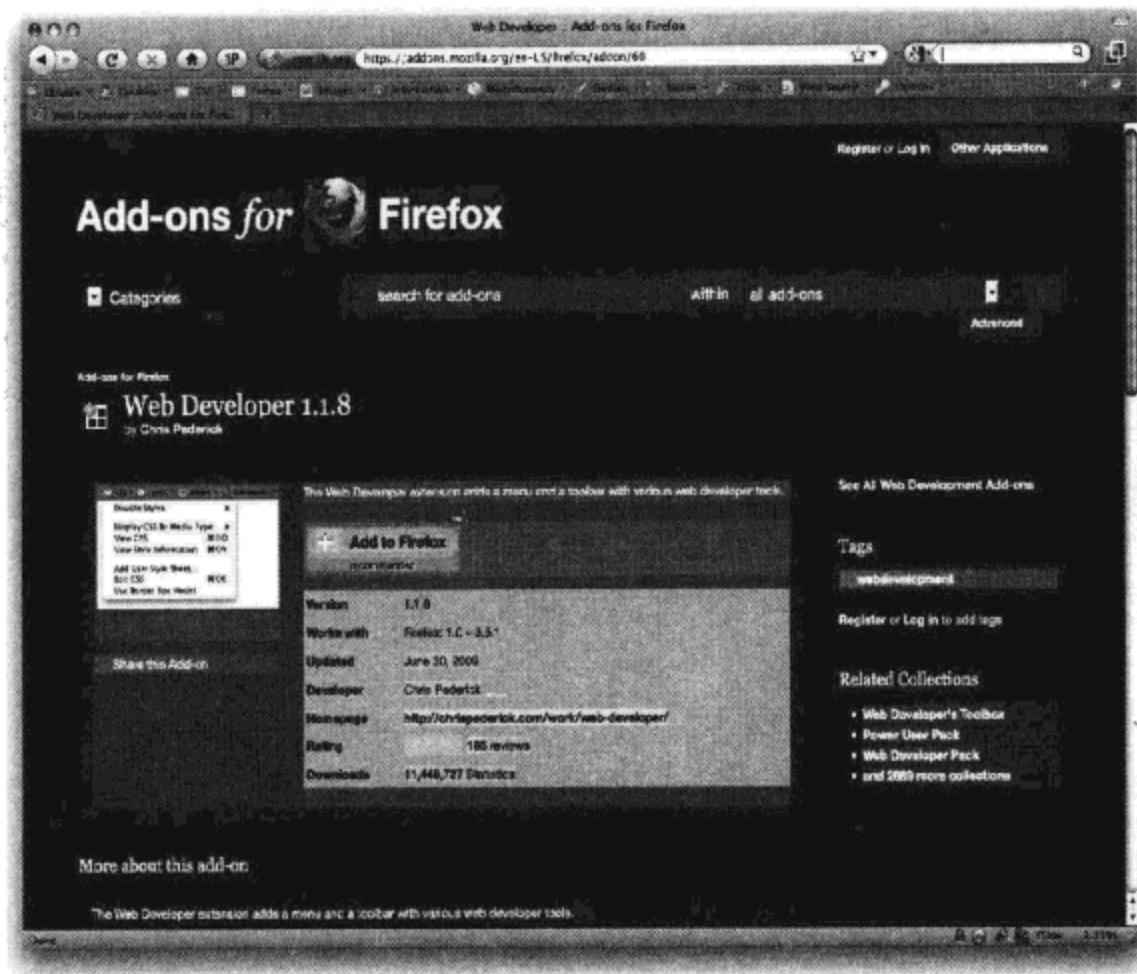


图9-5 Mozilla Add-ons站点的topographic视图



图9-6 Mozilla Add-ons站点的标题的layout视图

## 9.2 捕捉 bug 的基本知识

跟踪bug的第一步是检验你的HTML和CSS，检查打字或语法错误。某些显示错误是由于浏览器以混杂模式显示页面造成的。因此，最好检查使用的是否是适合自己的标记语言的DOCTYPE，从而让页面以标准模型显示（见第1章）。可以通过安装Firefox开发人员工具条，了解页面将以什么模式显示。如果页面以混杂模式显示，那么工具条最右边的对钩符号是灰色的。如果页面以标准模式显示，那么对钩符号是绿色的。单击这个对钩符号会提供关于页面的更多信息，并且显式地定义显示模式（见图9-7）。

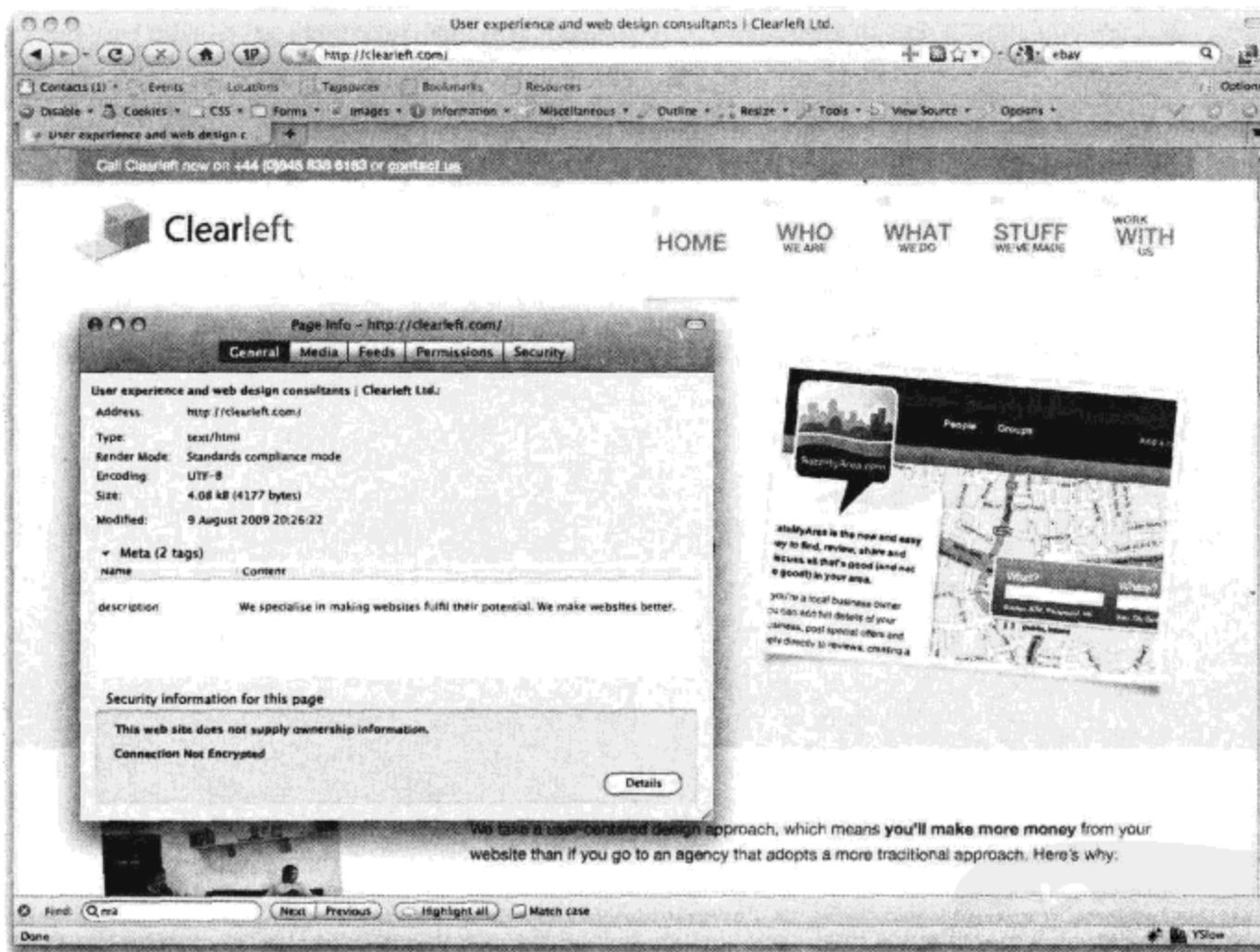


图9-7 Firefox网页开发人员工具条显示了页面将以标准模式还是混杂模式显示

许多Windows开发人员以前主要使用IE开发页面，所以每次修改页面之后，他们会在IE中预览页面，看看效果是否正确。在页面差不多准备好之后，他们在各种浏览器中进行测试并且尽力修复出现的任何bug。但是，这是一种危险的做法，可能会导致许多长期的问题。

IE 6是一种声名狼藉的浏览器，有很多bug和几个重要的CSS缺陷。由于使用IE作为主要的开发浏览器，许多开发人员错误地把IE的表现解释为正确的，因此反而奇怪为什么更现代的浏览器“破坏了”他们精心设计的CSS布局。页面实际上是在IE中被“破坏了”，在与标准更兼容的浏览

器中的显示方式是正确的。

更安全的方法是使用更符合标准的浏览器（比如Firefox或Safari）作为主要的开发浏览器。如果你的布局在这其中一种浏览器中工作正常，那么它很可能在所有符合标准的浏览器中都会表现正常，这意味着你的做法是正确的。然后，你可以在能力比较差的浏览器中测试页面，并且为发现的任何显示问题寻找解决方案。请记住，不要把浏览器测试留到项目快结束时。应该采用连续测试方法，在项目开发过程中用所有主流浏览器检查页面，这样就不会在项目快结束时又突然发现许多问题。

### 9.2.1 尽量在一开始就避免bug

这个建议似乎是不言而喻的，但是消除bug的一个最好方法确实是从一开始就避免问题。许多bug是由过分复杂的HTML或CSS造成的。因此，应该尽可能使用简单的代码实现所需的结果。所以要避免使用过分花哨的技术，只在绝对必需的情况下才使用hack。

因为实现同一个效果有许多种方法，在花费数小时调试某种技术之前，应该考虑是否可以采用另一种方法。只有在确认了没有解决问题的简单方法之后，才应该开始考虑复杂的解决方案。

### 9.2.2 隔离问题

一旦你确信出现了bug，就需要尽力隔离问题。通过隔离问题和识别症状，有可能查明问题的原因并修复它。隔离问题的一种方法是在相关的元素上应用边框或轮廓，看看它们的反应：

```
.promo1 {  
    float: left;  
    margin-left: 5px;  
    border: 1px solid red;  
}  
  
.promo2 {  
    float: left;  
    border: 1px solid green;  
}
```

我一般喜欢在代码中直接添加边框，你可以使用Web开发人员工具条中的轮廓选项，或者使用用来给不同元素加轮廓的许多bookmarklet中的一个。有时候，仅仅添加边框就会修复问题，这往往说明存在外边距叠加问题。

尝试修改几个属性，看看它们是否影响bug，如果有影响，查明是怎样影响的。尝试使bug的效果扩大化可能是有帮助的。例如，如果两个框之间的间隙在IE中比你预期的要大，那么加大外边距，看看会发生什么。如果框之间的间隙在IE中加倍了，那么可能是遇到了IE的双外边距浮动bug。

```
.promo1 {  
    float: left;  
    margin-left: 40px;  
    border: 1px solid red;  
}  
  
.promo2 {  
    float: left;  
    border: 1px solid green;  
}
```

尝试一些常见的解决方案，例如，将position属性设置为relative、将display属性设置为inline（在浮动元素上）或者设置宽度等尺寸，就可以修复许多IE bug。在本章后面，你将了解这些常见的修复方法以及它们的工作原理。

你可以轻松快速地找到和修复许多CSS问题。如果问题比较棘手，应该考虑创建一个基本测试案例。

### 9.2.3 创建基本测试案例

基本测试案例就是重现bug所需的最少量的HTML和CSS。通过创建基本测试案例，可以去掉一些“变量”，使问题尽可能简单。

要创建基本测试案例，首先应该复制出问题文件。删除多余的HTML，只留下最基本的内容。然后开始注释掉样式表，从而查明是哪些样式表导致了这个问题。进入这些样式表，并且开始删除或注释掉代码块。如果bug突然消失了，那么就知道是刚刚注释掉的最后一块代码导致了这个问题。继续注释掉代码块，直到只留下造成问题的代码。

然后，就可以开始详细研究这个bug了。删除或注释掉声明，看看会发生什么。这个修改对bug有什么样的影响？修改属性值，看看问题是否消失了。使用一些常用的修复方法，看看它们是否有效。编辑HTML，看看是否有影响。使用不同的HTML元素组合。有些浏览器有奇怪的空白bug，所以要尝试从HTML中删除空白。可能需要检查的地方非常多。

### 9.2.4 修复问题，而不是修复症状

知道问题的根源，对于实现正确的解决方案是非常有利的。因为给站点应用CSS样式有许多方式，最容易的解决方案是干脆回避这个问题。如果外边距导致了问题，那么可以考虑用内边距来代替。如果一种HTML元素组合导致了问题，那么可以考虑换一种组合。

许多CSS bug有描述性非常强的名称。所以在网络上搜索解决方案是相当容易的。例如，如果你注意到IE将所有浮动元素上的外边距加倍了，那么搜索“Internet Explorer Double Margin Float Bug”，就会找到解决方案。

如果发现无法回避这个bug，那么可能不得不设法消除症状。这往往需要将规则过滤到一个单独的样式表中，并且向这个浏览器提供修复方法。

### 9.2.5 请求帮助

如果创建了基本测试案例，尝试了常用的解决方案，搜索了可能的修复方法，但是仍然无法找到解决方案，那么就要请求帮助。可以求助于许多活跃的CSS社区，比如CSS-Discuss ([www.css-discuss.org/](http://www.css-discuss.org/))、Web Standards Group (<http://webstandardsgroup.org/>) 和 Stackoverflow (<http://stackoverflow.com>)。这些社区中有许多多年从事CSS站点开发的开发人员，因此很可能有人以前遇到过你的bug并且知道如何修复它。如果你遇到新的或特别有意思的bug，人们可能愿意提出建议或者帮助你查明bug的原因。

在请求帮助时要记住，大多数网页开发人员都非常忙。如果你还没有检验自己的代码，或者只是张贴自己站点的链接，期望他们研究数百行HTML/CSS代码，那么别指望得到很多帮助。在邮件列表或论坛上请求帮助最好的方法是，使用一个准确描述问题的标题，写出对问题的简短总结，然后粘贴基本测试案例，如果基本测试案例比较长的话，可以粘贴站点上测试案例的链接。带注解的屏幕截图也有帮助，因为有些问题光靠文字说明不容易说清楚，尤其是只影响某些浏览器版本的问题。

## 9.3 拥有布局

我们都知道浏览器有bug，而且Windows上的IE 6的bug似乎比大多数浏览器都多。IE的表现与其他浏览器不同的原因之一是，显示引擎使用一个称为布局（layout）的内部概念。因为布局是一个专门针对显示引擎内部工作方式的概念，所以一般情况下不需要了解它。但是，布局问题是许多IE/Win显示bug的根源，所以理解这个概念以及它如何影响CSS是有帮助的。

### 9.3.1 什么是布局

Windows上的IE使用布局概念来控制元素的尺寸和定位。那些“拥有布局”（have layout）的元素负责本身及其子元素的尺寸设置和定位。如果一个元素“没有拥有布局”，那么它的尺寸和位置由最近的拥有布局的祖先元素控制。

IE显示引擎利用布局概念减少它的处理开销。在理想情况下，所有元素都控制自己的尺寸和定位。但是，这会在IE中导致很大的性能问题。因此，IE开发团队决定只将布局应用于实际需要它的那些元素，这样就可以充分地减少性能开销。

在默认情况下拥有布局的元素包括：

- body

- html (标准模式中)
- table
- tr、td
- img
- hr
- input、select、textarea、button
- iframe、embed、object、applet
- marquee

布局概念是Windows上的IE特有的，而且它不是CSS属性。尽管设置某些CSS属性会使元素拥有布局，但是在CSS中无法显式地设置布局。可以使用JavaScript函数hasLayout查看一个元素是否拥有布局。如果元素拥有布局，这个函数就返回true，否则返回false。hasLayout是一个只读属性，所以无法使用JavaScript进行设置。

设置以下CSS属性会自动地使元素拥有布局。

- float: left或right。
- display: inline-block。
- width: 任何值。
- height: 任何值。
- zoom: 任何值 (Microsoft属性——不能通过检验)。
- writing-mode: tb-rl (Microsoft属性——不能通过检验)。

在IE 7中，以下属性也成了布局触发器。

- overflow: hidden、scroll或auto。
- min-width: 任何值。
- max-width: 除none之外的任何值。

### 9.3.2 布局的效果

布局是许多IE显示bug的根源。例如，如果一个文本段落靠着一个浮动元素，那么我们期望文本围绕这个元素。但是，在Windows上的IE 6和更低版本中，如果段落拥有布局（例如，由于设置了高度），那么它就被限制为矩形，因此阻止文本围绕浮动元素（见图9-8）。

各种浏览器上的显示都不同，这会导致浮动布局的各种问题。更糟的是，许多人使用IE作为主浏览器，他们会错误地认为这才是正确的表现，在其他浏览器以不同方式处理浮动元素时，他们反而会迷惑不解。另外，对于某些元素，布局似乎清理了其中包含的浮动元素，就像是设置了overflow:hidden。

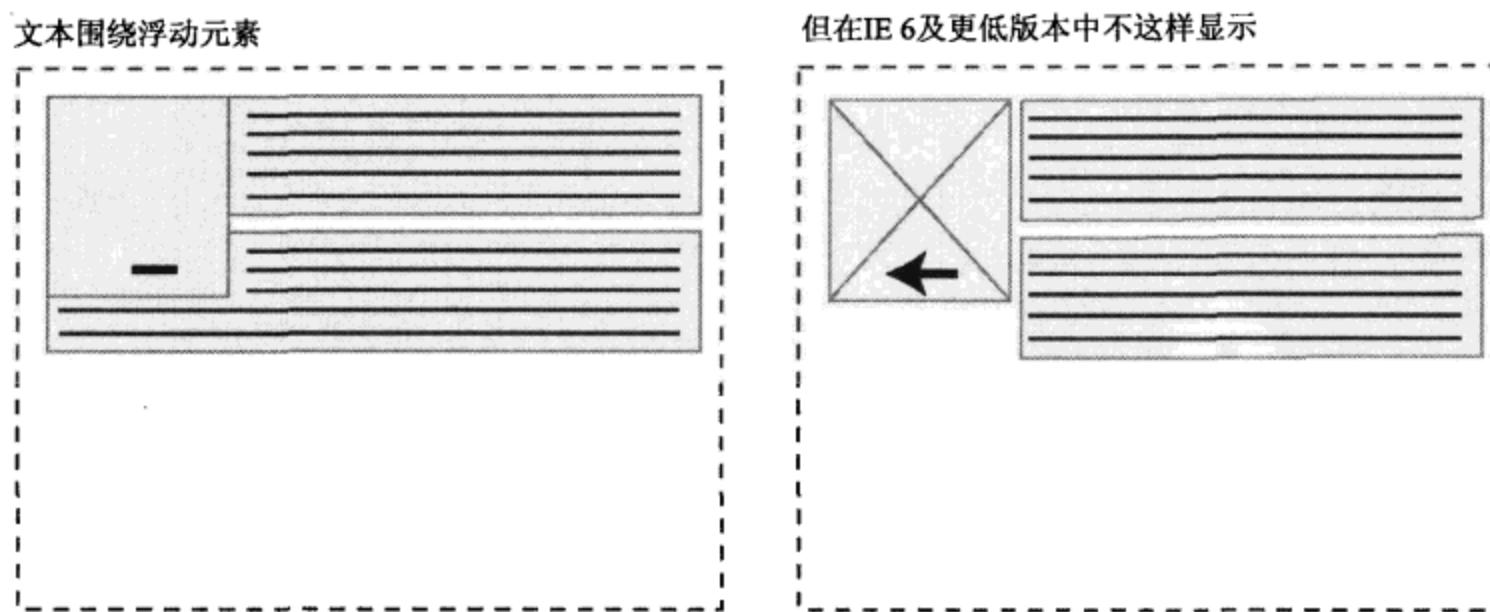


图9-8 期望文本围绕相邻的浮动元素。但是，在Windows的IE中，如果文本元素拥有布局，就不会这样显示

另一个问题是拥有布局的元素如何确定自己的尺寸。如果元素的内容变得比元素本身大，那么我们希望内容溢出到元素外。但是，在IE 6和更低版本中，拥有布局的元素会错误地扩展以便适应内容的尺寸（见图9-9）。

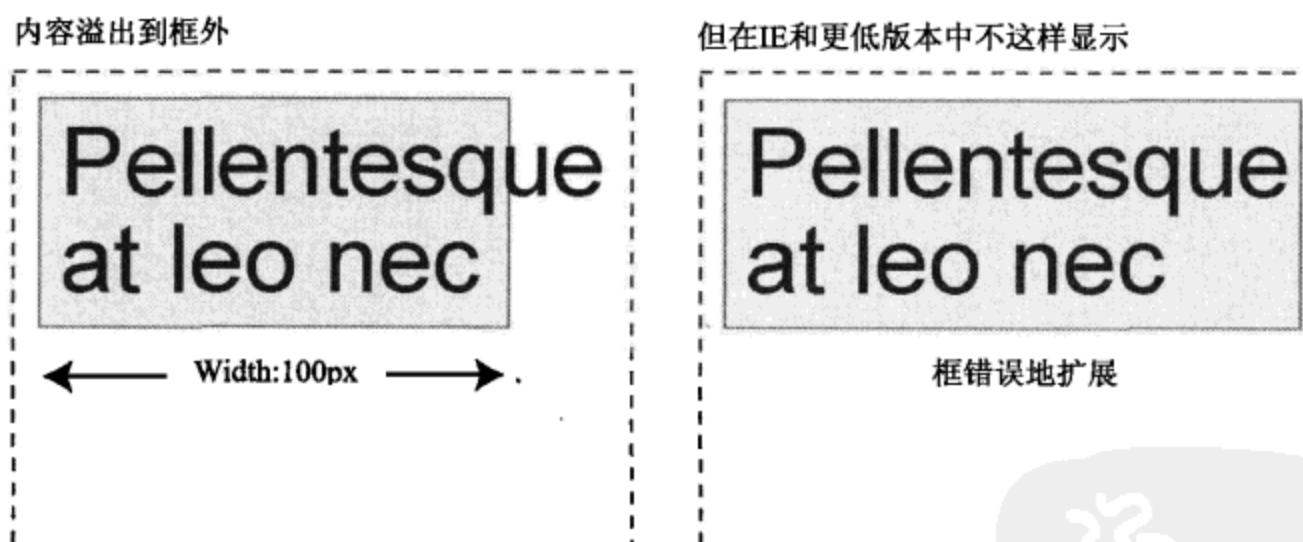


图9-9 拥有布局的元素会错误地扩展以便适应内容

这种显示错误意味着Windows上的IE中的width实际上更像是min-width。这种行为也是在IE中许多浮动布局被破坏的原因。当浮动框的内容错误地迫使框的宽度增加时，框对于可用空间来说太大了，因此下降到其他浮动元素下面。

其他问题包括：

- 拥有布局的元素不会收缩；
- 布局元素对浮动进行自动清理；
- 相对定位的元素没有布局；

- 在拥有布局的元素之间外边距不叠加；
- 在没有布局的块级链接上，单击区域只覆盖文本；
- 在滚动时，列表项上的背景图像间歇性地显示和消失。

你会注意到IE的许多修复方法都涉及通过设置属性迫使元素拥有布局。实际上，如果遇到一个IE bug，首先应该做的一件事情就是尝试通过应用规则迫使元素拥有布局，看看这是否可以修复问题。

如果希望进一步了解IE的内部hasLayout属性，我建议阅读<http://www.satzansatz.de/cssd/onhavinglayout.html>上的“On Having Layout”。

好在，IE团队在IE 7中已经修复了大多数与布局相关的问题。但是，他们解决问题的方法是找到常见的显示bug，然后通过在代码中创建例外来处理它们，而不是解决底层根源。因此，可能有一些不显著的布局bug还没有被发现。IE 8使用全新的显示引擎，据称不使用hasLayout属性，因此解决了这些问题的根源。

## 9.4 解决方法

在理想的情况下，编写正确的CSS在支持CSS的所有浏览器中都会正常工作。但是，与所有复杂的软件一样，浏览器也有bug和不一致的地方。以前，浏览器对CSS的支持非常差，所以开发人员不得不发挥创造性。开发人员过去要分析bug和没有起作用的CSS，对不同的浏览器有选择地应用不同的规则，从而以自己的方式解决问题。因此，hack和过滤器成了CSS开发人员的重要工具。

好在现代浏览器的CSS支持已经好多了，所以我们不再需要太关注hack了。但是，在老式浏览器彻底消失之前，我们可能还要维护遗留代码。因此，最好熟悉一些比较流行的hack和过滤器，只有了解它们，才能从代码中消除它们。但是，在开始讨论这些hack和过滤器之前，我们先看看条件注释。

### 9.4.1 IE 条件注释

条件注释是一种专有的（因此是非标准的）对常规HTML注释的Microsoft扩展。顾名思义，条件注释能够根据条件（比如浏览器版本）显示代码块。尽管是非标准的，条件注释在其他所有浏览器看来是常规注释，因此本质上是无害的。因此条件注释通常被看做处理IE特有的bug的最好方法。条件注释在Windows上的IE 5中首次出现，并且得到了Windows浏览器的所有后续版本的支持。

要想将特定的样式表提供给IE 5和更高的所有版本，那么可以在HTML文档的开头放置以下代码：

```
<!-- [if IE]
<link rel="stylesheet" type="text/css" href="/css/ie.css" />
-->
```

Windows上的IE 5和更高版本会接收样式表ie.css，而其他所有浏览器只会看到一些被注释掉的文本。这很有意思，但是用处不大，因为很少有在IE的所有版本中都存在的bug。因此，可能需要指定某一版本或版本范围。

可以使用条件注释指定一种特定的浏览器，比如IE 6.0，代码如下：

```
<!-- [if IE 6]
<link rel="stylesheet" type="text/css" href="/css/ie6.css" />
-->
```

还可以指定一组浏览器，比如IE 5和IE5.5：

```
<!-- [if lt IE 6]
<link rel="stylesheet" type="text/css" href="/css/ie5x.css" />
-->
```

除了使用条件注释向IE提供样式表之外，还可以使用它们隐藏特定的样式表。下面的语法称为向下显露的条件注释，它对IE的所有版本隐藏高级样式：

```
<!--[if !IE]>-->
<link rel="stylesheet" type="text/css" href="/css/advanced.css" />
<!--<! [endif]-->
```

下面的代码对IE 5.x隐藏样式：

```
<!--[if gte IE 6]><!-->
<link rel="stylesheet" type="text/css" href="/css/modern.css" />
<!--<! [endif]-->
```

条件注释极其有效，而且非常容易记住。主要的缺点是这些注释需要放在HTML中，而不是放在CSS中。如果发布了IE的新版本，你可能不得不更新站点的每个页面中的条件注释。但是，只要记得这么做，应该没有什么问题。

#### 9.4.2 关于 hack 和过滤器的一个警告

作为一种语言，CSS被设计成具有很强的向前兼容性。如果浏览器不理解某个选择器，那么它会忽略整个规则。同样，如果它不理解某个属性或值，它就会忽略整个声明。这个特点意味着添加新的选择器、属性和值应该不会对老式浏览器产生严重的影响。

可以利用这个特点，对比较高级的浏览器应用规则和声明，同时可以确保老式浏览器会平稳地退化。当浏览器的新版本发布时，如果它支持现在作为过滤器使用的CSS，那么它应该会如预

期那样工作。如果使用比较高级的CSS克服老式浏览器中的问题，这个问题很有希望在新版本中得到解决。因此，使用不被支持的CSS作为过滤机制是一种相对安全的方法。我说“相对”是因为浏览器的新版本有可能支持新的CSS，但是仍然表现出你试图修复的bug。

使用依赖于解析bug的过滤器是一种有点儿冒险的方法，因为它们依赖于bug而不是特性。与前面的方法相似，如果解析bug被修复了，而你试图修复的bug还没有得到解决，那么可能会再次遇到问题。但是，更严重的是，解析bug在浏览器的新版本中可能有新的表现形式。例如，假设Firefox的一个新版本有某个解析bug。如果使用这个bug作为过滤器向IE提供不同的宽度值，从而解决它专有的盒模型问题，那么Firefox可能会突然继承这个宽度，这可能会破坏许多站点的设计。还要记住一点：这些hack和过滤器常常会使你的代码失效。作为一般规则，应该使用依赖于不被支持的CSS的过滤器，而不是依赖于某种浏览器bug的过滤器，这可能比较安全。最好完全避免使用过滤器。

### 9.4.3 明智地使用 hack 和过滤器

很不幸，开发人员往往过分信赖于hack和过滤器，在CSS新手中这种倾向尤其明显。当某些东西在一种浏览器中不起作用时，许多CSS开发人员往往立即应用某种hack，把它当做某种魔法子弹来使用。实际上，一些开发人员误认为，自己掌握的hack和过滤器数量越多，自己的专业水平就越高。

如果经过充分研究，你认识到只能应用某种hack或过滤器，那么需要以明智且受控制的方式应用它。如果CSS文件短小且简单，并且只需要应用很少几个hack，那么将这些hack放在主CSS文件中并通过注释加以说明可能是安全的。但是，hack往往相当复杂，使代码更难阅读。如果CSS文件很长很复杂，或者需要使用的hack比较多，那么最好将它们放在它们自己的样式表中。这不但使代码容易阅读，而且如果hack在新浏览器中造成了问题，可以准确地知道它们的位置。与此相似，如果决定取消对某种浏览器的支持，那么只需删除适当的CSS文件，就可以删除相关联的hack。

### 9.4.4 应用 IE for Mac 带通过滤器

Tantek Çelik创建了一系列基于浏览器解析错误的过滤器（<http://tantek.com/CSS/Examples/>），允许使用@import规则将样式表提供给选择的浏览器。在条件注释出现之前，这些过滤器是过滤出各个IE版本的推荐方法。但是，现在对于某些情况，这些过滤器仍然很方便，比如可以显式地指定Mac上的IE 5.2。这可以使用Tantek的IE 5 for Mac带通过滤器（band pass filter）来实现，它利用CSS注释中的转义bug：

```
/*\*\*\*/
@import "ie5mac.css";
\***\
```

IE 5 for Mac会错误地转义第二个星号，导致应用了@import规则。因此，IE 5 for Mac看到的代码是：

```
/* blah */  
  @import "ie5mac.css";  
/**/
```

其他所有浏览器都会正确地忽略转义元素（因为它放在注释中），并认为@import规则被注释掉了。事实上，其他所有浏览器看到的规则是：

```
/* blah *//*  
  blah  
*/
```

与其他带通过滤器一样，不需要理解这个过滤器的工作原理，直接使用即可。这些过滤器的出色之处在于，它们巧妙地利用了老式浏览器中的bug。因此，应该可以放心地使用它们，它们应该不会在新式浏览器中造成问题。

#### 9.4.5 应用星号HTML hack

最著名、最有用的CSS过滤器之一是星号HTML hack。这个过滤器非常容易记住，它针对IE 6和更低版本。正如你知道的，HTML元素被认为是网页上的第一个元素（即根元素）。但是，IE的老版本有一个匿名的根元素，它包围着HTML元素。可以使用通用选择器指定包围在另一个元素中的HTML元素。因为这种情况只在IE 6和更低版本中出现，所以可以将特定的规则应用于这些浏览器：

```
* html {  
  width: 1px;  
}
```

因为这个bug在IE 7中已经修复了，它是针对IE老版本的可靠方法。

修改后的简化盒模型hack (MSBMH) 也使用了这种方法，MSBMH用于在老式浏览器中管理IE专有的盒模型。

```
#content {  
  width: 80px;  
  padding: 10px;  
}  
  
* html #content {  
  width: 100px;  
  w\idth: 80px;  
}
```

尽管我现在不建议使用这种技术，但是在遇到遗留代码时它可能有用。

#### 9.4.6 应用子选择器 hack

如果希望创建会被IE的老版本忽略的规则，不需要显式地指定这些浏览器。可以使用子选择器hack。这种技术实际上不是hack，它只是使用了IE老版本不理解而现代浏览器能够理解的选择器。

在这个示例中，使用子选择器hack对Windows上的IE 5和IE 6隐藏透明的背景PNG图像：

```
html>body {  
    background-image: url(bg.png);  
}
```

这个规则对于IE的老版本是隐藏的。但是，IE 7支持子选择器和PNG透明，所以它会正确地解释这些代码。

## 9.5 常见 bug 及其修复方法

CSS开发人员最重要的技能之一是发现常见浏览器bug。通过了解导致这些bug的各种元素，可以在它们造成问题之前发现并且修复它们。

### 9.5.1 双外边距浮动 bug

最常见且最容易发现的一个bug是IE 6和更低版本中的双外边距浮动bug。顾名思义，这个Windows bug使任何浮动元素上的外边距加倍（见图9-10）。

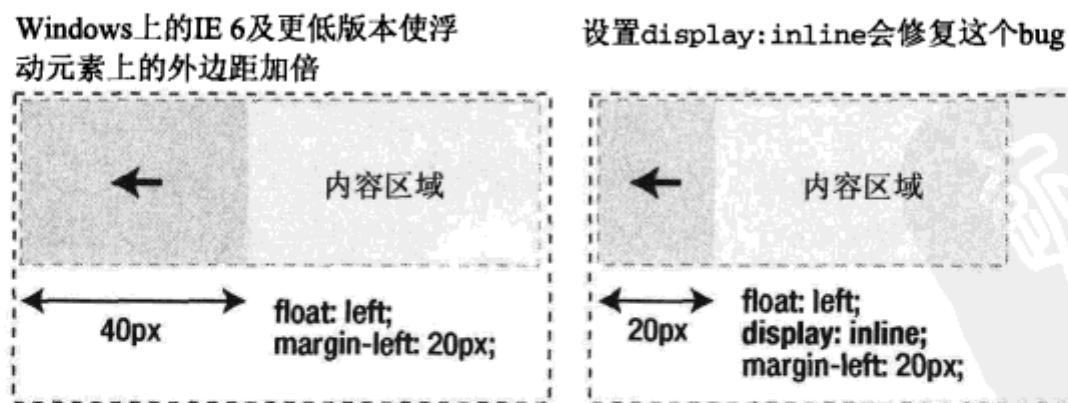


图9-10 IE的双外边距浮动bug示意图

这个bug很容易修复，将元素的display属性设置为inline就行了。因为元素是浮动的，将display属性设置为inline实际上不会影响显示方式。但是，这似乎会阻止Windows上的IE 6和更低版本将所有外边距加倍。这是一个非常容易发现和修复的bug：每当对具有水平外边距的元素进行浮动时，都应该很自然地将display属性设置为inline，以备外边距将来被加大。

### 9.5.2 3像素文本偏移bug

Windows上IE 5和IE 6上另一个非常常见的bug是3像素文本偏移bug。当文本与一个浮动元素相邻时，这个bug就会表现出来。例如，假设将一个元素向左浮动，并且不希望相邻段落中的文本围绕浮动元素。你可能会在段落上应用一个左外边距，其宽度等于浮动元素的宽度：

```
.myFloat {
    float: left;
    width: 200px;
}

p {
    margin-left: 200px;
}
```

如果这么做，在文本和浮动元素之间就会出现一个莫名其妙的3像素间隙。一旦浮动元素停下来，3像素间隙就会消失（见图9-11）。

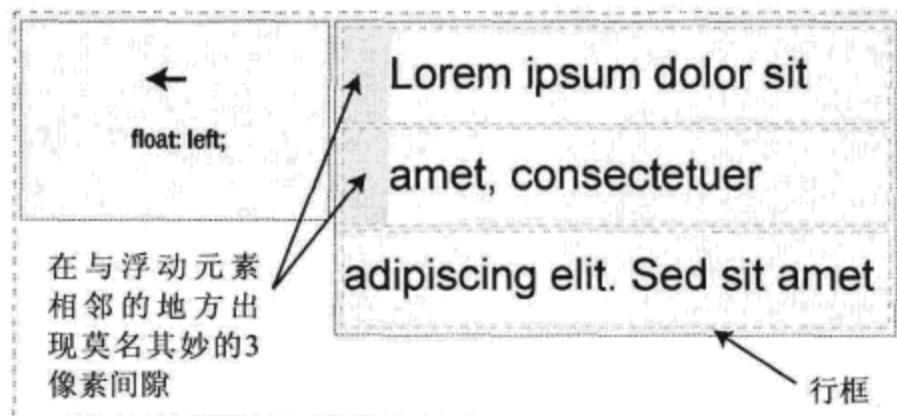


图9-11 IE 5和IE 6的3像素文本偏移bug示意图

修复这个bug需要双管齐下。首先，给包含文本的元素设置任意的高度。这会迫使元素拥有布局，这在表面上会消除文本偏移。因为Windows上的IE 6和更低版本将height作为min-height那样对待，所以设置一个小的高度并不会影响元素在这些浏览器中的实际尺寸。但是，这会影响其他浏览器，所以要对除了Windows上的IE 6和更低版本之外的所有其他浏览器隐藏这个规则。最好的方法是使用条件注释把这些样式转移到单独的CSS文件中。

```
p {
    height: 1%;
}
```

但是，这项技术会导致另一个问题。前面学过，拥有布局的元素被限制为矩形的，并且出现在浮动元素的旁边而不是它们的下面。添加200像素的外边距实际上会在IE 5和IE 6中在浮动元素和段落之间产生200像素的间隙。为了避免这个间隙，需要将IE 5-6/Win上的外边距重新设置为零：

```
p {  
    height: 1%;  
    margin-left: 0;  
}
```

文本偏移被修复了，但是现在另一个3像素间隙出现了，这一次是在浮动元素上。为了去掉这个间隙，需要在浮动元素上设置一个负的3像素右外边距：

```
p {  
    height: 1%;  
    margin-left: 0;  
}  
  
.myFloat {  
    margin-right: -3px;  
}
```

如果浮动元素是除了图像之外的任何其他东西，那么这个问题已经修复了。但是，如果浮动元素是图像，那么还有最后一个问题需要解决。Windows上的IE 5.x在图像的左右都添加3像素的间隙，而IE 6不改变图像的外边距。因此，如果需要支持IE 5.x，那么对这些浏览器使用以下样式表：

```
p {  
    height: 1%;  
    margin-left: 0;  
}  
  
img.myFloat {  
    margin: 0 -3px;  
}
```

用于IE 6的样式表如下：

```
p {  
    height: 1%;  
    margin-left: 0;  
}  
  
img.myFloat {  
    margin: 0;  
}
```

### 9.5.3 IE 6 的重复字符 bug

另一个涉及浮动元素的奇怪的bug是IE 6的重复字符bug。在某些情况下，一系列浮动元素的最后一个元素中的最后几个字符会在浮动元素下面重复出现，见图9-12。

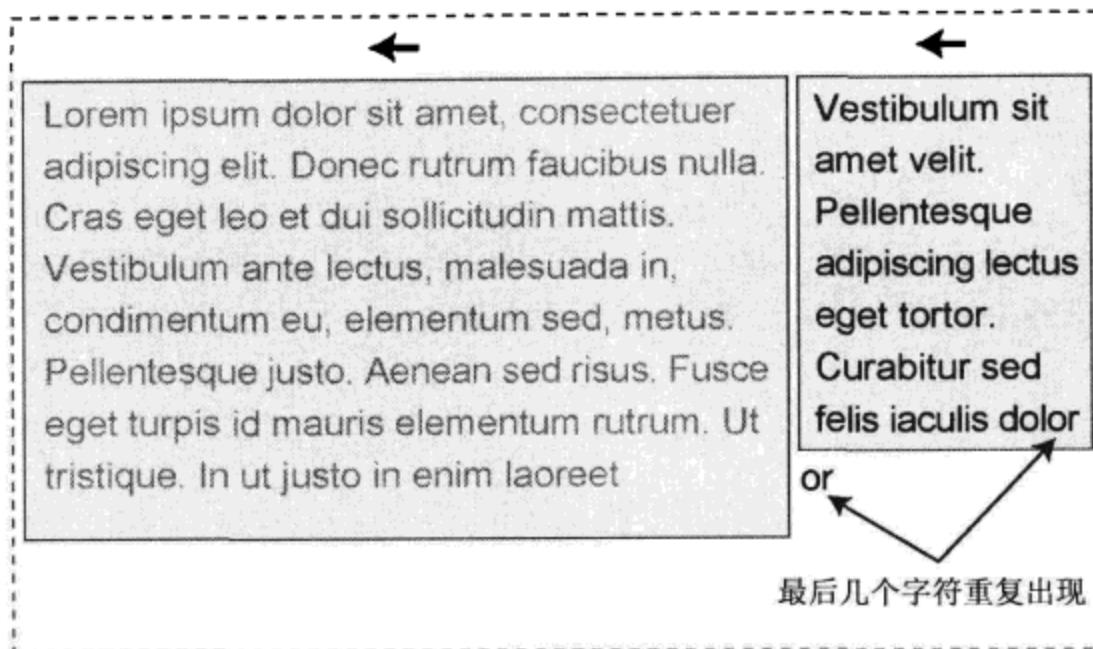


图9-12 IE 6的重复字符bug示意图

当在一系列浮动元素的第一个和最后一个元素之间有多个注释时，就会出现这个bug。前两个注释没有影响，但是后续的每个注释会导致两个字符重复出现。所以，3个注释会导致两个重复字符，4个注释会导致4个重复字符，5个注释会导致6个重复字符。

```
<div id="content">
  <!-- mainContent -->
<div id="mainContent">
  ...
</div><!-- end mainContent -->
  <!-- secondaryContent -->
<div id="secondaryContent">
  ...
</div>
```

奇怪的是，这个bug似乎与前面的3像素文本偏移bug相关。为了修复这个bug，可以通过设置负的右外边距从最后一个浮动元素上去掉3像素，或者使容器扩大3像素。但是，这两种方法可能在IE 7中造成问题，IE 7中可能没有这个bug。因此，避免这个bug最容易、最安全的方法是从HTML代码中删除注释。

#### 9.5.4 IE 6的“藏猫猫”bug

另一个奇怪而且很烦人的bug是IE 6的“藏猫猫”(peek-a-boo) bug，之所以起这个名称是因为在某些条件下文本看起来消失了，只有在重新加载页面时才再度出现。出现这个bug的条件是一个浮动元素后面跟着一些非浮动元素，然后是一个清理元素，所有这些元素都包含在一个设置了背景颜色或图像的父元素中。如果清理元素碰到了浮动元素，那么中间的非浮动元素看起来消失了，隐藏到了父元素的背景颜色或图像后面，只有在刷新页面时才重新出现(见图9-13)。

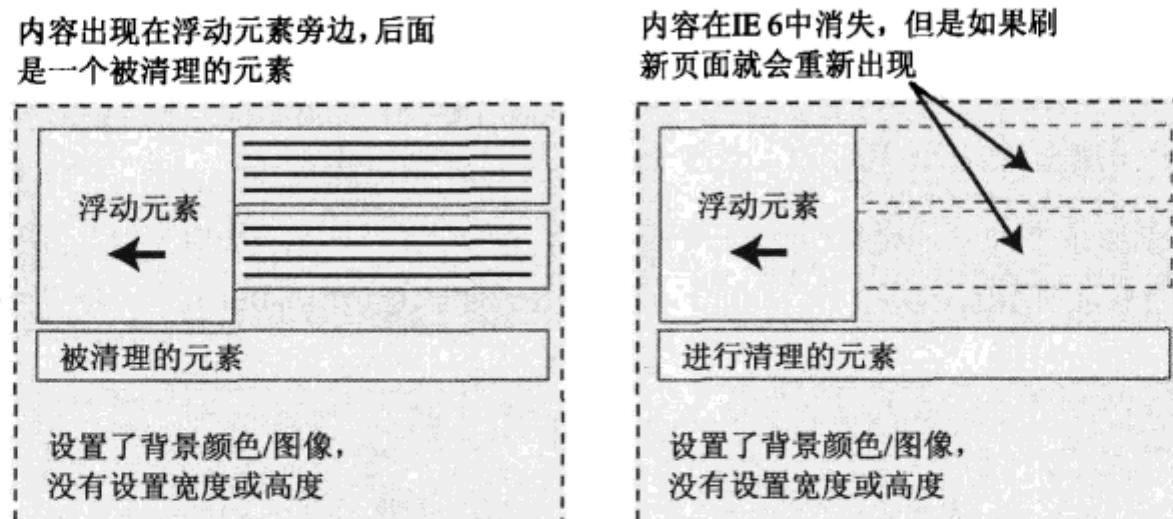


图9-13 IE 6的“藏猫猫”bug示意图

好在有许多方法可以解决这个bug。最容易的方法可能是去掉父元素上的背景颜色或图像。但是，这常常是不可行的。另一个方法是避免清理元素与浮动元素接触。如果容器元素应用了特定的尺寸，那么这个bug似乎就不会出现了。如果给容器指定行高，这个bug也不会出现。最后，将浮动元素和容器元素的position属性设置为relative也会减轻这个问题。

### 9.5.5 相对容器中的绝对定位

我要讨论的最后一个主要浏览器bug与相对定位容器中的绝对定位元素有关。在前面几章中你学到将绝对定位的元素嵌套在相对容器中是多么有用。但是，IE 6和更低版本在使用这种技术时有许多bug。

这些bug的原因在于相对定位的元素没有获得IE/Win的内部hasLayout属性。因此，它们不创建新的定位上下文，所有绝对定位元素相对于视口进行定位（见图9-14）。

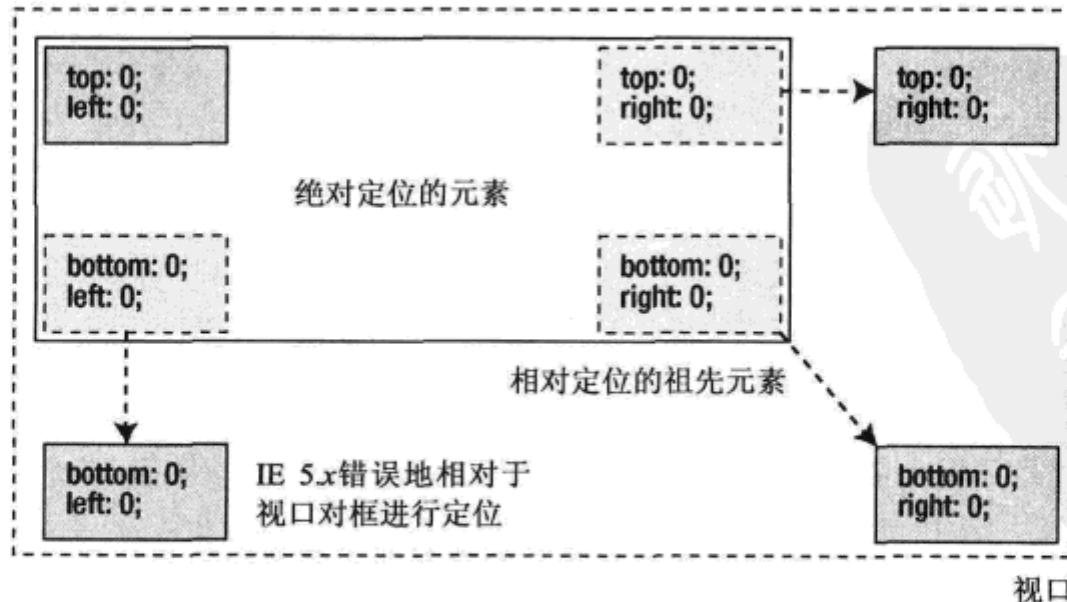


图9-14 IE 5.x对相对容器中的绝对定位元素的定位方式不正确

为了使Windows上的IE 6和更低版本表现正确，需要迫使相对定位的容器拥有布局。一种方法是在容器上显式地设置width和height。但是，我们常常希望在不知道容器的width和height的情况下，或者在需要这些属性保持灵活的情况下使用这种技术。

我们可以使用条件注释过滤IE 5和IE 6，为容器布局提供一个任意的高度。这会让容器拥有布局，但是因为IE 6和更低版本中的元素会不正确地扩展以适应它们的内容，所以设置小的高度不会影响实际高度。

```
.container {  
    height: 1%;  
}
```

### 9.5.6 停止对IE的批评

IE并不是唯一一种有bug的浏览器，所以你可能会奇怪我为什么只关注IE bug。不必担心，我并不是和微软过不去，这么做是有理由的。

首先，IE目前仍然占有很大的浏览器市场份额，所以bug往往很快被发现并且很好地记录下来。但是，IE的开发节奏比其他浏览器慢得多。Firefox和Safari每过几个月就会发布新版本，而要看到IE的新版本则要等好几年。因此，IE的bug往往会长时间存在。

在Firefox和Safari中以如此快的速度发现和修复bug确实很棒，但是这也有问题。开发人员要应付的浏览器版本不是两三个，而是10或20个。你无法确定用户是否使用最新的版本，这使测试变得极其困难。另一方面，IE差不多5年没有发布主要修订版了。因此，bug有更多的时间可以暴露出来，开发人员也有更强的寻找修复方法的动力。

幸而，与以前的版本相比，IE 8更符合标准了。许多有名的IE bug已经被解决，而且对高级CSS 2.1特性的支持也在增加。与所有浏览器一样，新的bug也会出现，IE 8远不是完美的。但是，人们越快地升级到IE 8和Firefox等现代浏览器，IE 6等老式浏览器就会越快地退役。

## 9.6 分级浏览器支持

讨论bug就不能不提到浏览器支持。每当IE的新版本发布时，都会引发对是否应该停止支持以前版本的激烈争论。毕竟，如果Microsoft不再正式支持IE 6，那么我们为什么要操心呢？但是，浏览器支持问题没有明确的解决方案，这取决于具体站点。

如果你的站点是为Web开发人员准备的，那么可能许多用户都使用Firefox和Mac浏览器，在这种情况下IE 6的使用量可能很低，不需要为它操心。但是，对于每个月有一百万访问者的站点，几个百分点就意味着有好几万客户不满意。对于商业或消费者站点，IE 6用户数量可能很高。在某些站点上，IE 6用户数量甚至超过IE 7。这是因为许多公司的IT部门只允许用户使用特定的浏览器版本，而许多家庭用户往往只在买新机器时更新浏览器。因此，不应该简单地决定是否停止

支持某种浏览器，而是需要划分浏览器支持的级别，根据具体的站点决定支持的实际意义。这就是分级浏览器支持的作用。

Yahoo和BBC等大型组织认识到浏览器的水平并不一致，很难确保站点在所有浏览器中的外观和表现完全相同，如果勉强追求这个目标，会增加维护成本并阻碍创新。为了避免只能针对所有浏览器都有的特性设计站点，这些公司开始采用分级支持表（见图9-15和图9-16）。这些图表并不把浏览器支持看做只有支持或不支持两种选择，而是提供多种支持级别，包括在现代浏览器上显示完整的设计，直到在老版本上只显示内容。尽管每个公司处理这个问题的方法不一样，但基本步骤是相同的。

	Win 2000	Win XP	Win Vista	Mac 10.4.+	Mac 10.5.+
Firefox 3.0.+		A-grade	A-grade		A-grade
Firefox 2.0.+		A-grade			A-grade
IE 8.0		A-grade	A-grade		
IE 7.0		A-grade	A-grade		
IE 6.0	A-grade	A-grade			
Opera 9.6+		A-grade			A-grade
Safari 3.2+				A-grade	A-grade

图9-15 Yahoo的A级浏览器分级浏览器支持表

Browser	IE	Mozilla	Opera	Safari	Konqueror	IE	NS 4-
Platform	Windows	All	All	Mac	Linux	Mac	All
Level 1	6, 7	FF 1.5.x/ 2.0.x	9	1.3+, 2.x, 3.x			
Level 2	5, 5.5	FF 1.0.x	8	1.0, 1.1, 1.2	3+		
Level 3	1, 2, 3, 4		7-		2-	5-	1, 2, 3, 4
Must test	6, 7	FF 2.0.x	9	2.0			
Should test	5.5	FF 1.5	8	1.3, 3.x			
Notes		#1	#1	#1	#2		
Engine	Trident (4-7), Ohare (1-3)	Gecko	Presto	Webcore	KHTML	Tasman	Mariner

图9-16 BBC分级浏览器支持表

首先，需要确定一组浏览器，希望确保站点在这些浏览器中的表现一致。然后，在所有这些浏览器上执行测试。这个组常常包含最新的、你的用户使用最多的浏览器。Firefox、Safari和Opera的最新版本以及IE 7和IE 8可能属于这一类。对于这些浏览器，希望站点的外观大体上相同，但是由于实际原因，也允许有一些细微差异，只要差异不大即可。

接下来，确定一组已经过时但仍然比较重要的浏览器。这可能包含Firefox和Safari的老版本以及IE 6。在这些浏览器中随机选择几个执行测试，尝试修复找到的问题。但是，即使显示效果不完美并且在浏览器之间有差异，也是可以接受的，只要不影响访问内容即可。

最后，确定一组完全过时的你不想正式支持的浏览器。这包括IE 4、Netscape Navigator 4和Opera 7等浏览器。对于这些浏览器，你仍然希望确保内容和功能是可用的，但是表示方式无所谓。因此，可以接受非常显著的设计变化。更好的方法是对于这些浏览器完全取消样式。

分级支持方法可以非常灵活地处理各种浏览器和其他用户代理。BBC的图表是一个好起点，但是每个站点都是独特的，我强烈建议你针对每个具体项目创建自己的分级支持表。

## 9.7 小结

在本章中，你学习了跟踪和修复CSS bug的一些重要技术，了解了Windows上IE的内部hasLayout属性以及它为什么会成为许多Windows上的IE浏览器bug的根源，知道了一些最常见的浏览器bug和修复方法。最后，了解了如何使用分级支持表处理各种浏览器。

接下来，你会看到由当今最好的两位CSS设计师和开发人员创建的两个极棒的实例研究，从而体会如何结合使用本书讲述的所有知识。





## 第 10 章

# 实例研究：Roma Italia

尽管多了一些批注，也破旧了一点儿，但是本书的第一版仍然留在我的书架上。自从出版以来，这3年我经常参考它。在这段时间，行业内发生了许多变化，最引人注目的是IE 7和IE 8的发布。这带来了一系列新的CSS 2和CSS 3特性，现在的主流浏览器都支持这些特性。我们将在这个实例中讨论其中几个特性。

但是，许多东西没有变。标记仍然是标记，标准仍然是标准。仍然需要那些知道如何用HTML、CSS和JavaScript创建优美易用的体验的人才，这种需求甚至更迫切了。

自从第一版出版以来，读者的知识和经验也更丰富了。因此，我希望这个实例研究比以前的更有挑战性。在这个实例研究中，将学习以下主题：

- 1080布局和网格
- 高级CSS2和CSS3特性
- 字体链接和更好的Web排版
- 用Ajax和jQuery增加交互性

请访问[roma.cssmastery.com](http://roma.cssmastery.com)体验这个实例研究的效果，还可以从[www.friendsofed.com](http://www.friendsofed.com)获得源代码文件。

### 10.1 关于这个实例研究

Roma Italia是专门为这个实例研究创建的虚构网站（图10-1显示的是主页）。但是，这个实例研究中使用的CSS技术并不是虚假的。每种技术都是精心选择的，力求为读者提供一套可靠的高级CSS技术，其中许多技术都可以在真实环境中使用。其他试验性技术在当前的浏览器中还没有

得到一致的支持，使用它们是为了展示在不远的将来可能实现的功能。



图10-1 Roma Italia主页

这个实例研究网站的用途是提供意大利罗马旅游指南，它由两个页面组成：主页和视频页面。

主页上的一些链接指向真实的资源，其他链接是只用于演示的死链接。Roma Italia上所有的照片、视频和内容都来源于我和妻子最近去意大利罗马的旅行。如果所有链接都链接到真实的资源，这个虚构的网站完全可以成为真正的网站。

特别感谢Aaron Barker ([aaronbarker.net](http://aaronbarker.net)) 帮助完成这个实例研究中的几个jQuery和Ajax示例。感谢我的妻子Suzanne拍摄了网站上使用的一些照片。

## 10.2 基础

在编写标记时，我考虑的最重要的因素是，应该尽可能有意义而且尽可能轻量。“有意义”是指我们选择的HTML元素和选择器名称应该以适当的方式表示内容，让内容的层次关系和结构在去掉所有样式的情况下仍然是有意义的。分隔线GIF和重复的br元素早就过时了，它们已经被在逻辑上或语义上表示内容的元素所取代，比如：

- 畅销商品的有序列表 (ol);
- 页面上的主标题 (h1);
- 引用顾客的好评 (blockquote和cite)。

这种方式要求我们从思维中抛弃“表现性信息”的概念，Andy Clarke在他的著作*Transcending CSS* (New Riders, 2006年) 中详细描述过这个概念。我仍然清楚地记得早期使用CSS时的经历。当时，我们在编写一个相当大的Web应用程序，我们自作聪明地创建了一系列表现性的类名，这让我们可以用下面这样简洁明确的形式标记内容：

```
<p class="arial red 10">
```

但是，当需要重新设计这个应用程序时，麻烦就来了，几十个模板都要变成其他字体，不能是Arial 10像素。

“轻量”，是指对于各种标记（HTML的元素、属性和值，CSS的选择器、属性和值），能简化就简化。例如：

```
background-color: #c27e28;  
background-image: url(..../img/feature-orange.jpg);  
background-repeat: no-repeat;
```

简化为：

```
background: #c27e28 url(..../img/feature-orange.jpg) no-repeat;
```

在这个实例研究中，可以看到许多有意义的轻量标记的示例，我会在这里描述其中一部分，大多数标记你都能自己看明白。

### 10.2.1 着眼于HTML 5

在有意义的轻量标记方面，我选择了HTML 4.01 Strict作为DOCTYPE，而没有采用XHTML 1.0 Strict和HTML 5。下面简要地解释我的理由。

**XHTML 1.0 Strict。**过去几年，许多业内人士（包括我自己）都使用这个版本。但是，Dave Shea提出了一个非常引人注目的观点，他认为应该放弃XHTML，着眼于HTML 5。

“六年前，许多人认为是XHTML是Web的未来，而我们目前处于XML时代。但是在这个过渡时期，我们明确地发现XHTML2实际上是没有前途的，至少在我们关心的领域是这样……让网站使用HTML5 DOCTYPE需要应对许多变化，我还没有准备好，所以只能使用HTML的最新版本……在确认HTML5时代已经到来之前，4.01是最合适的选择，以后四五年可能一直处于这个阶段。”（“Switched”，<http://mezzoblue.com/archives/2009/04/20/switched/>）

**HTML 5。**简单地说，HTML 5是超文本标记语言的下一个主要版本。好消息是，没有意义的div和span元素将被更有意义的元素替代，比如nav、header和video。

这意味着可以不再使用下面这样的代码：

```
<div class="header">
  <h1>Page Title</h1>
</div>
```

或

```
<object><param/><embed src="http://vimeo.com/3956190"></embed></object>
```

同样的HTML可写成：

```
<header>
  <h1>Page Title</h1>
</header>
```

和

```
<video src="http://vimeo.com/3956190">
```

坏消息是，在这个实例研究发表之前，主流浏览器（尤其是IE）对HTML 5的支持还很不够。估计还要等几个月到几年HTML 5才能得到全面的支持，成为我们在创建网站时可以选用的版本。

为了准备向HTML 5转移，可以使用当前的DOCTYPE编写标记，但是采用与HTML 5相似的语义性类名。Jon Tan在“Preparing for HTML 5 with Semantic Class Names”(<http://jontangerine.com/log/2008/03/preparing-for-html5-with-semantic-class-names>)中详细讨论了这种方法。

例如，使用nav元素的HTML 5标记如下：

```
<nav>
  <ul>
    <li><a href="">Menu item 1</a></li>
    ...
  </ul>
</nav>
```

而使用HTML 4或XHTML 1的与HTML 5相似的语义性标记如下：

```
<div class="nav">
  <ul>
    <li><a href="">Menu item 1</a></li>
    ...
  </ul>
</div>
```

但这种方法的缺点是，最终可能出现许多多余的div。既然我们的目标是有意义的轻量标记，那么目前最好的标记应该是：

```
<ul class="nav">
  <li><a href="">Menu item 1</a></li>
  ...
</ul>
```

那么，我究竟怎么看HTML 5呢？当HTML 5得到全面支持，进入它的黄金时期时，我们都会很好地适应这一变化。思维模式的变化会非常小。在此之前，我仍然会采用一贯的方式。

如果需要关于HTML 5的更多参考资料，请访问：

- 12 Resources for Getting a Jump on HTML 5: [http://cameronmoll.com/archives/2009/01/12\\_resources\\_for\\_html5/](http://cameronmoll.com/archives/2009/01/12_resources_for_html5/)
- Coding a HTML 5 Layout from Scratch: <http://smashingmagazine.com/2009/08/04/designing-a-html-5-layout-from-scratch/>
- 关于HTML 5的Wikipedia文章: [http://en.wikipedia.org/wiki/HTML\\_5](http://en.wikipedia.org/wiki/HTML_5)
- The Rise of HTML 5: <http://adactio.com/journal/1540>
- Google Bets Big on HTML 5: <http://radar.oreilly.com/2009/05/google-bets-big-on-html-5.html>

## 10.2.2 reset.css

在几年前我刚开始编写CSS风格的网站时，常见的做法是在主样式表的开头声明几个“全局”样式：body、a、img、h1、h2、h3等。这样做就会覆盖浏览器的默认样式。这种习惯最终发展

成了目前使用“reset”样式表（通常命名为reset.css）的习惯做法。

正如Yahoo团队指出的，reset样式表“可以避免和缓解HTML元素默认样式不一致的问题，跨所有主流浏览器创建一个一致的舞台……”。我个人很喜欢Eric Meyer编写的Reset CSS，这个实例研究中使用的就是这个样式表，可以从<http://meyerweb.com/eric/tools/css/reset/>下载。

我使用master.css样式表导入网站使用的所有样式表。首先声明reset样式表，后面导入的所有样式表可以根据需要覆盖reset样式：

```
@import url("reset.css");
@import url("screen.css");
@import ...
```

用于屏幕显示的所有样式都放在screen.css中。在这个实例研究中，还使用了另外3个样式表。

- autocomplete.css包含即时搜索特性的样式。
- datepicker.css包含日历日期选择器的样式。
- ie.css是使用条件注释引用的（见下一节），它包含专门用于IE的样式。

其实把autocomplete.css和datepicker.css中的样式直接插入screen.css中也很容易，但是为了便于学习这个实例研究，还是采用单独的样式表。

### 10.3 1080布局和网格

2006年，在为分辨率为 $1024 \times 768$ 或更高的显示器选择最佳宽度时，我陷入了进退两难的局面（见<http://www.cameronmoll.com/archives/001220.html>）。当时，许多人已经开发适合 $800 \times 600$ 分辨率的网站相当长的时间了，正在转向适合1024像素分辨率的宽度。

我在那篇文章中提出，对于 $800 \times 600$ 以上的分辨率，960像素是理想的宽度。这考虑到了浏览器的差异以及许多用户并不以全屏模式浏览。更重要的是，960是一个相当神奇的数字：它可以被2、3、4、5、6、8、10、12、15和16整除。这为分割网格提供了多种选择（稍后讨论网格）。

那篇文章发表之后，960差不多成了Web上固定宽度设计的事实标准。许多Photoshop、浏览器和操作系统插件都以960作为默认宽度，甚至出现了在960网格上构建的完整的CSS框架，它就叫做960.gs（<http://960.gs/>）。3年之后，一个新问题出现了：现在应该转向更大的宽度吗？我不确定，但是这个实例研究提供了研究这个问题的好机会。

为了避免固定宽度的反对者们误解，我必须声明一点：我非常喜欢采用最小宽度和最大宽度的流式设计，我的Extensible CSS系列（[http://cameronmoll.com/archives/2008/02/the\\_highly\\_extensible\\_css\\_interface\\_the\\_series/](http://cameronmoll.com/archives/2008/02/the_highly_extensible_css_interface_the_series/)）和本书第一版的实例研究设计（<http://tuscany.cssmastery.com/>）就是证据。实际上，通过使用Cameron Adams的依赖于分辨率的布局方法（<http://themaninblue.com/>

writing/perspective/2006/01/19/) 和 Ethan Marcotte 的流式图像 (<http://unstoppablerobotninja.com/entry/fluid-images/>) 等技术，可以用流式布局实现一些精彩的效果。但是，我相信固定宽度总还是需要的，而且坦率地说，在许多方面它比流式宽度更实用。

因此，假设我们都同意现在至少应该讨论一下是否让宽度超过960，那么理想的宽度是多少？有下面几种选择。

- 1020可以被2、3、4、5、6、10、12、15整除，但是不能被8和16整除，它比960宽不了多少。
- 1040可以被2、4、5、8、10、16整除，但是不能被3、12或15整除。目前，它是介于960和全屏宽度之间的合理宽度（正如前面提到的，许多用户不浏览全屏）。
- 1080可以被2、3、4、5、6、8、10、12、15整除，但是不能被16整除。它提高了宽度范围的上限，如果处理不当，尺寸（行长）会成为问题。

应该注意，等分并不是分割网格的唯一方法，在某些情况下甚至不是最佳方法。可以考虑比例分割，比如黄金分割 ([http://en.wikipedia.org/wiki/Golden\\_ratio](http://en.wikipedia.org/wiki/Golden_ratio))。但是，正如 Jason Santa Maria 指出的，比例分割在 Web 上可能并不实用，因为它们依赖于当前可见的水平和垂直分割，而后者常常是不确定的（见 <http://jasonsantamaria.com/articles/whats-golden/>）。

总之，如果现在转移到宽度超过960，我不确定会像以前一样成功。这里列出的宽度的可扩展性似乎都不如960，至少在数学上是这样。但是，对于这个实例研究，我选用1080。它为网格分割提供丰富的选择，而且它比960大很多，值得尝试一下。

## 在 Web 设计中使用网格

在图形设计中使用网格已经有几十年了，但是直到最近几年网格才受到 Web 设计者的喜爱。 Wikipedia 上对网格及其优点做了简洁的说明：“排版网格是用于组织内容的二维结构，它由一系列相交的垂直轴和水平轴组成。设计者可以利用网格以合理、简便的方式组织文本和图像。” ([http://en.wikipedia.org/wiki/Grid\\_\(page\\_layout\)](http://en.wikipedia.org/wiki/Grid_(page_layout)))

网格包含列、行、外边距、隔离带（各列和各行之间的空白）、流水线（水平分割）和其他部分。在纸张这样的介质上，这些部分的宽度和高度受制于材料的尺寸，设计者很容易计算出它们的尺寸。但是，在 Web 设计中，宽度常常比较容易计算，而高度很难确定。这是因为可滚动页面的高度可能是无限的。

因此，Roma Italia 的网格主要用于垂直分割，如图 10-2 所示。

可以通过取消以下标记的注释显示和隐藏这个网格：

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

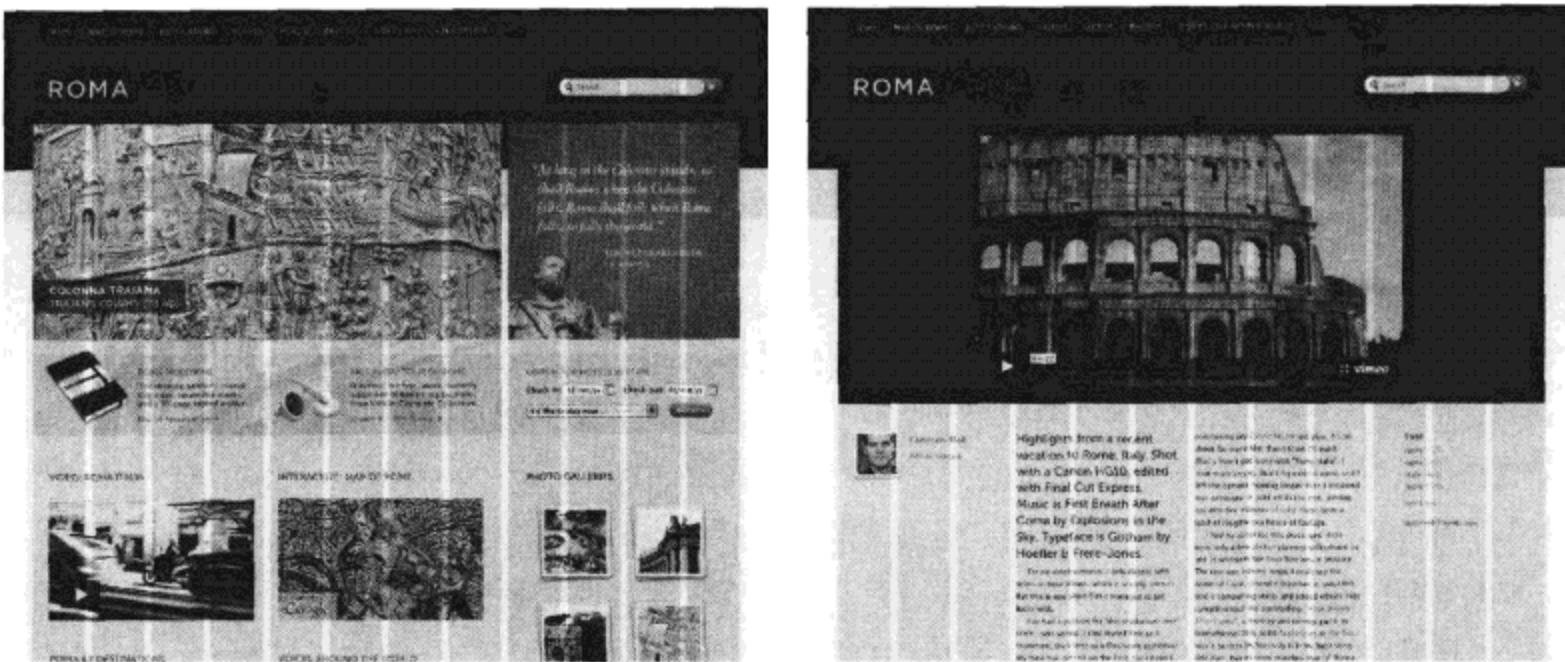


图10-2 用来设计网站的12等分网格

如果你有Firebug这样的浏览器插件 (<http://getfirebug.com/>)，就更容易了，这个插件在浏览器中临时修改文档的标记。打开Firebug，双击body标签以显示注释，然后在Firebug中直接编辑HTML。

可以看到，这个网格分割为12列。每列的宽度为80像素，右边有10像素的隔离带（图10-3），布局的总宽度为1080像素。左边的外边距中添加一个10像素的偏移（本质上是一个额外的隔离带）以平衡网格。但是，这个偏移和最后一列右边的隔离带对于浏览者是不可见的。你可以认为，如果不计算不可见部分，这个网格实际上是1070像素的；或者相反，让这些部分可见，就是1090像素。无论如何，我们的网格总宽度为1080。

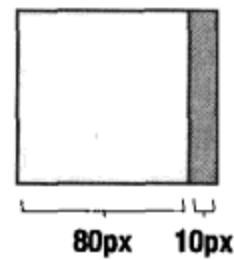


图10-3 每列的宽度为80像素，右边有10像素的隔离带

大多数文本和图像与列和隔离带对齐。当然，这正是使用网格的意图。但是你会注意到，我没有把所有元素都严格地对齐。这里应该注意的要点是，网格不一定要指定元素的准确位置。相反，它帮助大体定位元素，精确的定位要依靠设计者的判断。在*Making and Breaking the Grid* (Rockport出版社, 2005年) 中，Timothy Samara更明确地说明了这个概念：

“一定要理解一点：尽管网格是精确的定位工具，但是元素不应该附属于网格。网格的作用是提供总体的统一性，而不应该抑制元素组合的灵活性。在大多数情况下，在给定的网格中可以以各种各样的方案布置页面，有时候不理会网格也没关系。设计者不应该畏惧网格，应该敢于突破它的限制。真正良好的网格会提供无数探索的机会。”

但是，只看一个页面看不出网格真正的价值，必须整体观察页面集。例如，在印刷的小册子上，网格可以统一整本册子中元素的布局。同样，如果Roma Italia是一个真正的网站，它的所有页面（不只是在这里看到的两个）都使用相同的网格，那么对于用户会产生视觉上的连贯性，同时不会限制设计者的布局灵活性。

本节只简要地讨论了网格。在The Grid System (<http://www.thegridsystem.org/>) 和Smashing杂志的“Designing with Grid-Based Approach” ([http://www.smashingmagazine.com/2007/04/14/designing-withgrid-based-approach/](http://www.smashingmagazine.com/2007/04/14/designing-with-grid-based-approach/)) 中可以找到更多参考资料。

## 10.4 高级 CSS 2 和 CSS 3 特性

在IE 7于2006年10月发布之前，本节介绍的特性还是不合理或者在技术上不可行。Firefox、Safari和其他浏览器早就支持CSS 2和CSS 3规范中的许多高级特性，IE7至少是跟上了它们的步伐。其中最引人注目的特性包括min-width和max-width、属性选择器、相邻同胞选择器、子选择器和PNG图像中的alpha透明度。

通过结合使用这些得到良好支持的特性和其他还未得到良好支持的特性，我们可以实现一些非常棒的效果。一个极端的例子是，John Allsopp只使用CSS重新创建了Apple的导航条，完全没有使用图像（见[http://westciv.com/style\\_master/blog/apples-navigation-bar-using-only-css](http://westciv.com/style_master/blog/apples-navigation-bar-using-only-css)）。本节给出一些没这么极端的例子（但是我认为它们的效果仍然很好）。

Roma Italia中使用了下面这些高级的CSS 2和CSS 3特性：

- 相邻同胞选择器
- 属性选择器
- box-shadow
- opacity
- RGBa
- content
- 多栏
- text-overflow
- 多背景
- @font-face
- min-/max-width和min-/max-height
- PNG图像中的alpha透明度

在这个实例研究中，我们将讨论其中一部分特性：属性选择器、box-shadow、RGBa、文本溢出、多栏、多背景和@font-face。对于不讨论的特性，我通过在标记中加注释帮助你理解。把这份CSS简要说明（电子文件或打印件）放在手边也会有帮助：<http://cameronmoll.com/articles/widget/cheatsheet.pdf>。

### 10.4.1 网站需要在每种浏览器中看起来完全一样吗

在浏览器的地址栏中输入“[Dowebsitesneedtolookexactlythesameineverybrowser.com](http://Dowebsitesneedtolookexactlythesameineverybrowser.com)”，就会发现本节标题的答案。这个简单的网站是由Dan Cederholm于2008年开发的，他以这种方式鲜明地反对网站必须在所有浏览器中看起来完全一样的谬论。他呼吁Web开发者们更追求创新，不要被绝对的统一所束缚。Dan的网站只用一个字明确地反对把视觉不一致当做异类的偏见。

在Roma Italia中，视觉上最明显的不一致是使用多背景特性造成的。当前每个元素只能有一个图像，这个特性将允许给一个元素设置多个背景图像。圆角效果的爱好者会很喜欢这个特性。

到撰写本书时，在主流浏览器中只有Safari支持多背景。（有意思的是，Safari从1.3版开始就支持这个特性，而这个版本是在2005年发布的！）这意味着，在Firefox和IE等浏览器中这个网站看起来略有不同。使用这个特性不仅是出于这个实例研究本身的需要，而且是为了证明一点：只要不对总体用户体验产生负面影响，在不同的浏览器中提供略有不同的体验是完全合理的。

如果得到充分的支持，多背景特性一定会给Web专业人员提供很大方便，很容易流行起来。只需用逗号分隔每个图像及其值：

```
background: url(image1.png) no-repeat top left,
            url(image2.png) no-repeat top right,
            url(image3.png) no-repeat bottom left;
```

也可以单独定义属性和值：

```
background-color: #000;
background-image: url(image1.png), url(image2.png), url(image3.png);
background-repeat: no-repeat;
background-position: top left, top right, bottom left;
```

这个实例研究中在两个地方使用了多背景，见图10-4。请注意Safari与Firefox和IE之间的差异。

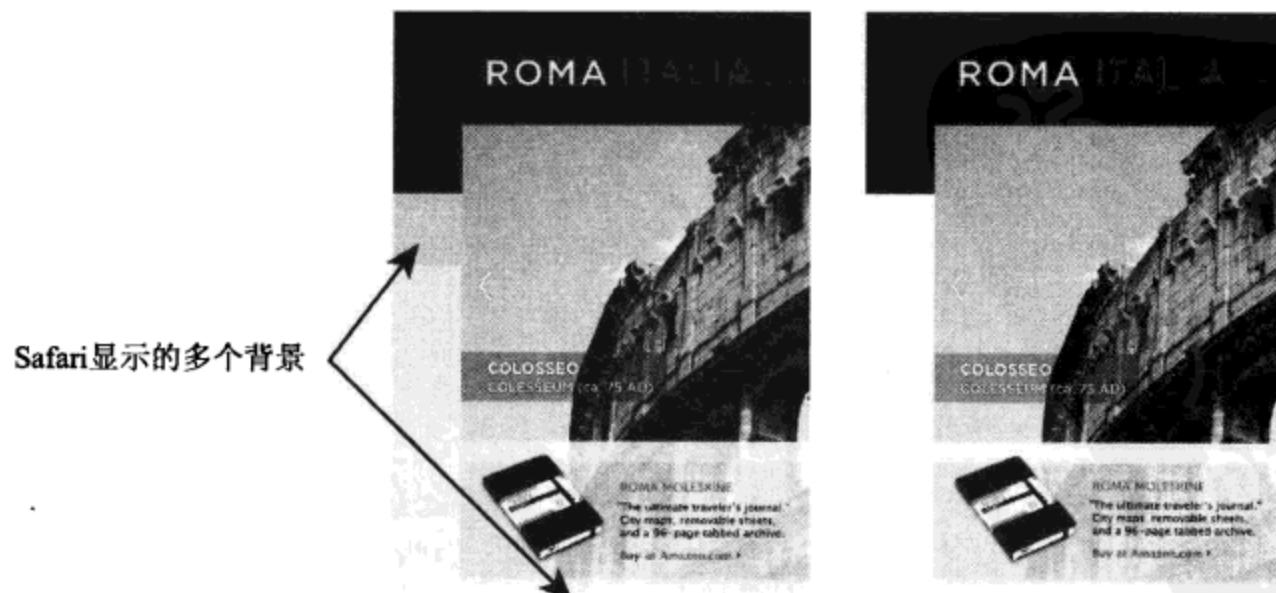


图10-4 在Safari（左图）与Firefox和IE（右图）之间背景图像的差异

在body中使用两个背景图像为网站提供背景：大的深棕色图像和轻微渐变的浅棕色图像，分别名为bg-dark.jpg和bg-light.jpg。CSS如下：

```
body {
    background: url(..../img/bg-dark.jpg) repeat-x top center,
    url(..../img/bg-light.jpg) repeat-x 239px left;
    background-color: #f1efe8;
}
```

因为Firefox和IE还不支持多背景，如果只使用上面的CSS，这两个图像都不会显示。这会导致背景完全是空的，这不是我要的效果。因此，在这个background属性前面插入同一个属性，这样至少会显示深色的图像：

```
background: #f1efe8 url(..../img/bg-dark.jpg) repeat-x;
```

Firefox读取这个属性并忽略另一个。我们在ie.css中也使用相同的属性，因为IE不接受这种组合。screen.css中最终的CSS如下：

```
body {
    background: #f1efe8 url(..../img/bg-dark.jpg) repeat-x;
    background: url(..../img/bg-dark.jpg) repeat-x top center,
    url(..../img/bg-light.jpg) repeat-x 239px left;
    background-color: #f1efe8;
}
```

要说明一点：这不是实现这里所示效果的最有效方式。首先，可以使用结合了这两个图像的单个图像，这样就不需要使用多个背景图像了。其次，为了让Firefox和IE至少显示一个图像，添加了重复的标记。使用这种效率不高的做法只是为了证明在浏览器之间有轻微的视觉不一致是可以的，同时演示多背景能够实现的效果。希望这个特性尽早得到充分的支持。

## 10.4.2 属性选择器

有了属性选择器，就可以引用元素中包含的任何属性或属性值，不再需要在元素中添加class或ID。可以对包含固有属性的任何元素使用属性选择器。例如，`img[alt]`寻找一个属性，而`img[src="small.gif"]`寻找一个属性值。另外，可以使用语法字符串寻找相似的属性值，比如`img[src^="sm"]`寻找以前缀“sm”开头的任何属性值（例如“small”）。更多示例请参见“CSS3: Attribute selectors”（<http://www.css3.info/preview/attribute-selectors/>）。

属性选择器在许多情况下都可以提供方便，但是最有用的地方可能是表单。如果通过一般的选择器`input { }`对元素应用样式，就会影响表单中的所有`input`元素。这意味着，如果希望只对文本框或提交按钮应用样式，就必须添加额外的class和ID。因此，属性选择器是一种寻找特定元素的简便方法。

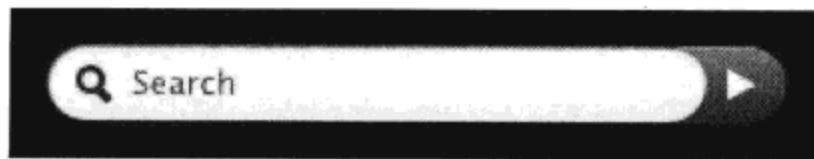


图10-5 这个搜索框使用两个input元素，分别由一个属性选择器选择

接近网站顶部的搜索框使用的是属性选择器（见图10-5）。HTML如下：

```
<form action="#" method="get" accept-charset="utf-8">
  <fieldset>
    <legend></legend>
    <label for="search-input">Search</label>
    <input type="text" id="search input" name="search" value="" title="Search">
    <input type="image" name="" src="img/search-go.gif">
  </fieldset>
</form>
```

在这里，希望对文本框应用几个样式属性，让search-go.gif向左浮动。粗体代码是要用属性选择器选择的两个input元素。注意，这两个input元素中都没有class或ID属性。这是因为我们可以使用type属性选择它们，代码如下：

```
#header form input[type="text"] {
  display: block;
  ...
  background: url(..../img/search-bg.gif) no-repeat;
}

#header form input[type="image"] {
  float: left;
}
```

包含属性type="image"的所有input元素都会向左浮动，包含属性type="text"的所有input元素都会应用指定的样式。另外，在jquery.plugins.js中使用相同的语法添加jQuery和AJAX功能：

```
$('#header form input[type="text"]').searchField();
```

后面一节将讨论jQuery和AJAX。

### 10.4.3 box-shadow、RGBa 和 text-overflow

在Roma Italia网站的中间，靠近页面底部的地方，会看到“Voices Around the World”短片。如果这个短片是真实的，只要Twitter更新(tweet)中包含hash标签#romaitalia，tweet就会根据Twitter用户的地理位置随机地出现在地图上。访问这个网站的用户可以单击这些随机的引述，就会进入包含完整tweet、作者的用户名和来自其他罗马迷的tweet的页面，这样就可以通过Twitter

更新实时地了解罗马。这个实例研究中显示的tweet是虚构的，但是你可以试试@roma\_italia，这是我为这个实例研究注册的真实的Twitter账号。

每几秒淡入和淡出的地图标志需要相当复杂的代码（见图10-6），我们以此演示CSS 3的box-shadow、RGBa和text-overflow特性。



图10-6 淡入和淡出的地图标志，在鼠标悬停时显示文本

每个标志由3个部分组成：tweet文本、带阴影的白色背景和表示地图标志的圆点图像。标志放在一个列表项（li）中，列表项放在无序列表（ul）中，无序列表包含一个世界地图背景图像：

```
<ul>
  <li class="l1" id="map2" style="top: 61px; left: 53px;"><a href="#">
    <em>Absolutely divine. Don't skip the Il Vittoriano. Its size alone is
    impressive. There's a stunning view from the top.</em></a></li>
    <li>...</li>
  </ul>
```

对ul应用以下样式属性：

```
#voices ul {
  position: relative;
  width: 310px;
  height: 178px;
  background: url(..../img/bg-map.gif) no-repeat;
}
```

注意，我们把position设置为relative。这样就可以相对于ul对每个列表项进行绝对定位。如果不这样做，列表项就会相对于另一个父元素定位，很可能是body。（关于绝对定位的知识，请回顾第3章。）

每个地图标志li使用以下样式属性：

```
#voices ul li.l1 {
  position: absolute;
  padding-top: 16px;
  background: url(..../img/mapmarker-dot.png) no-repeat 2px top;
}
```

地图标志图像作为背景图像嵌入，类11表示一号位置（圆点在右边），类12表示二号位置（圆点在左边）。没有在地图上定位每个标志的位置属性。这是因为在标志淡入时我们使用行内样式定位每个标志，对于这个标志是style="top: 61px; left: 53px;"。也就是，距离ul顶部61px，距离它左边53px。

tweet文本上的白色背景略微透明，在左边、右边和底边有阴影。分别使用RGBa和box-shadow实现这两个样式：

```
#voices ul li.11 a {
    display: block;
    padding-left: 11px;
    font: 11px/14px Georgia, serif;
    color: #32312a;
    -webkit-box-shadow: 0 2px 3px rgba(0, 0, 0, 0.35);
    -moz-box-shadow: 0 2px 3px rgba(0, 0, 0, 0.35);
    background-color: rgba(255, 255, 255, 0.78);
}
```

对box-shadow和RGBa的解释请参见第11章中Simon的实例研究。但是注意，RGBa不透明度与另一个CSS 3特性opacity不一样。RGBa不透明度可以应用于特定的属性（比如背景），它只影响这个属性。另一方面，opacity会影响应用它的元素中的所有东西，比如

```
#voices ul li.11 a {
    opacity: 0.35
    ...
}
```

opacity的值与RGBa相似：0（完全透明）到1（完全不透明）。但是，它会影响整个元素。如果在这里使用它，那么不但白色背景会有35%的不透明度，tweet文本也会有。

当用户把鼠标停在地图标志上时，标志的显示改变，见图10-7。

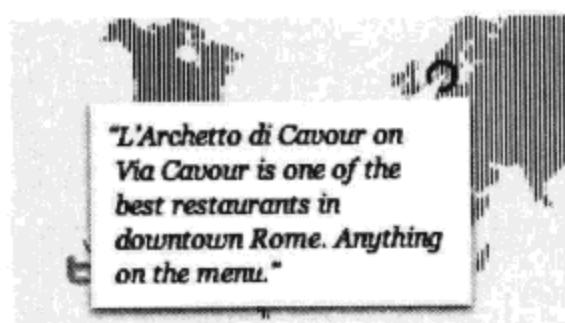


图10-7 在鼠标悬停时显示完整的文本

这里要使用text-overflow。我经常使用这个特性——如果每次使用它，我就得到一美元，现在就可以在巴哈马群岛买海滩别墅了。主流浏览器现在对它的支持相当好，实际上，IE对它的支持比Firefox好，Safari也是。如果文本太长，在包含它的元素中放不下，text-overflow会截断它。使用ellipsis值在被截断的文本后面加上省略号（...）。

对于每个地图标志，使用text-overflow把文本截断为一行：

```
#voices ul li.11 a em {  
    white-space: nowrap;  
    width: 135px;  
    overflow: hidden;  
    text-overflow: ellipsis;  
    -o-text-overflow: ellipsis;  
    -moz-text-overflow: ellipsis;  
    -webkit-text-overflow: ellipsis;  
}
```

因为并非每种浏览器都正式支持它（尽管所有主流浏览器都已经支持它了），我们加上了前缀-o-（Opera）、-moz-（Mozilla）和-webkit-（Webkit）。然后，为了实现鼠标悬停效果，在元素中添加: hover伪类，把height改为72px，把overflow设置为visible。

```
#voices ul li.11 a em:hover {  
    white-space: normal;  
    overflow: visible;  
    text-overflow: inherit;  
    -o-text-overflow: inherit;  
    cursor: hand;  
    cursor: pointer;  
    background: #fff none;  
    height: 72px;  
    padding-left: 11px;  
    padding-bottom: 5px;  
    margin-left: -9px;  
}
```

这样这个效果就完成了；感谢CSS3.info提供的text-overflow示例，我在创建这个效果时借鉴了这些示例。在他们的网站http://www.css3.info/上还可以找到其他CSS 3特性。

## 10.5 字体链接和更好的 Web 排版

Web排版技术非常多。由于这个实例研究篇幅的限制，我们在这里只讨论几种技术：

- 对font-size使用px
- 标点符号悬挂
- 多栏文本布局
- 字体链接和嵌入

### 10.5.1 按以前的方式设置 font-size

多年以来，在用font-size设置文本大小时，px是事实上的标准单位。这让设计者在从

Photoshop（或其他软件）转移到HTML时可以使用一致的绝对的文本大小单位。随着我们越来越关注和了解可访问性，相对的文本大小（em或%）逐渐成为首选单位。这让用户（尤其是视力差的用户）可以通过浏览器的设置永久地改变浏览器的默认文本大小，或者用键盘命令Ctrl+和Ctrl-（Windows）或Command+和Command-临时地改变。

因此，直到最近，所有主流浏览器都会放大或缩小文本的大小，同时保持页面的格式和布局。这通常称为文本缩放。这要求创建的标记允许任何包含文本的元素采用相对大小。例如，如果一个包含文本的div设置了背景图像，那么必须重复这个图像以适应文本缩放导致的div增长，或者创建更大的图像为增长留出空间。我在系列文章“*The Highly Extensible CSS Interface*”中详细讨论过这个问题（见[http://cameronmoll.com/archives/2008/02/the\\_highly\\_extensible\\_css\\_interface\\_the\\_series/](http://cameronmoll.com/archives/2008/02/the_highly_extensible_css_interface_the_series/)）。

但是，对于Ctrl+/-和Command+/-命令，每种主流浏览器（Safari、Firefox、Google Chrome、Opera和IE）的最近版本现在默认采用页面缩放而不是文本缩放。页面缩放直接缩放整个页面——布局、格式和文本大小。元素保持它们的大小和形状，这会大大减少文本缩放补偿处理。实际上，浏览器承担了处理相对大小的责任。

这究竟意味着什么呢？这意味着，px可以再次成为适合font-size的单位。这还意味着，用户分辨不出用绝对单位和用相对单位设置的文本之间的差异。但是，对于设计者，差异很明显。绝对单位可以消除在整个CSS文档中计算相对单位的负担——无论父元素的font-size是多少，在文档中的任何地方14px就是14px。

请记住，这个实例研究网站是现实的，也是试验性的。我通过它探索一些问题的答案，试验一些做法是否可行。你的项目可能没这么自由，应该根据访问者的情况做出正确的选择。与Web专业人员所做的几乎所有决定一样，必须根据访问者和用户做出正确的选择。无论Web行业发生什么变化，这条原则都不会变。简单地说，如果相对大小适合你的项目，就采用相对大小，我并不反对。

关于对font-size使用px的更多讨论，请参见以下文章：

- The Problem with Pixels: <http://www.wilsonminer.com/posts/2007/mar/16/problem-pixels/>
- IE 7 Does not Resize Text Sized in Pixels: [http://www.456bereastreet.com/archive/200703/ie\\_7\\_does\\_not\\_resize\\_text\\_sized\\_in\\_pixels/](http://www.456bereastreet.com/archive/200703/ie_7_does_not_resize_text_sized_in_pixels/)
- Mezzoblue – Zoom: <http://mezzoblue.com/archives/2008/10/07/zoom/>
- Hello Old Friend: <http://orderedlist.com/articles/hello-old-friend>

### 10.5.2 标点符号悬挂

标点符号悬挂是出色设计者的特色标志之一。大多数Adobe设计应用程序都提供这个特性，但是目前还无法通过CSS属性实现。实际上，在CSS 3规范中有一个名为hanging-punctuation

的属性（见`http://www.w3.org/TR/css3-text/#hanging-punctuation`），但是我还没有看到任何浏览器支持这个属性。

标点符号悬挂把标点符号放到文本块的外边，从而避免影响文本的视觉连贯性。图10-8是一个使用引号的示例。

*“This text does not use hanging punctuation. Notice how the quotation mark aligns with the left edge of the text block.”*

*“This text uses hanging punctuation. Notice how the entire text block aligns with the left edge, while the quotation mark sits outside it.”*

图10-8 这个示例显示标点符号悬挂（下部）与标点符号对齐文本边缘（上部）的差异。后者是大多数图形设计软件和浏览器中文本的默认显示方式

主页上的3个地方使用了这种技术：Roma Moleskine标题下面的第一行文本（见图10-9）、Voices Around the World地图标志中的tweet文本以及橙色的Venerable Bede语录。其中最后一处是一个图像，所以我们将讨论第一处，第二处使用的技术也相同。

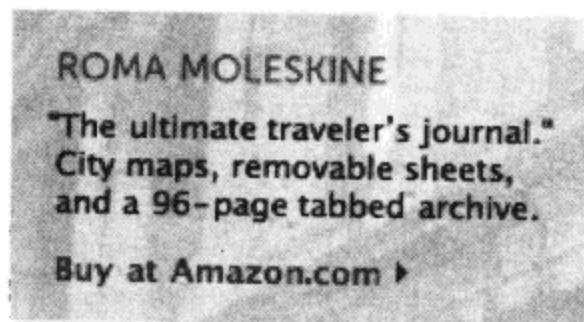


图10-9 Roma Italia中实现的标点符号悬挂

HTML很简单：

```
<div id="featurette1">
  ...
  <p>
    &ldquo;The ultimate traveler&rsquo;s journal.&rdquo; City maps,
    removable sheets, and a 96-page tabbed archive.</p>
  ...
</div>
```

HTML实体“、”和“表示引号。对于标点符号悬挂，并不一定要使用HTML实体，但这是出色的版面设计的另一个特色标志。对于新手来说，这些实体会让源代码显得有点儿乱，但是对于有经验的设计者，他能想象出浏览器显示这些标点的样子。

这里使用的CSS如下：

```
#featurette1 p {
    text-indent: -.3em;
}
```

就这么简单。在你自己的项目中，应该根据使用的字体和文本大小调整这个值。

### 10.5.3 多栏文本布局

对于多栏文本这么宽的布局来说，保持适合阅读的尺寸（即行长）是个问题。尺寸是文本块的宽度，表示为每行的字符数（包括空格）。关于最佳的每行字符数有许多种观点，范围从每行45个字符到95个字符，而且因介质而异。本节并不讨论最佳尺寸，而是讨论如何保持合理的尺寸。

因为这个布局的总宽度为1080像素，很适合试试CSS3规范中的多栏文本特性。当前，WebKit和Mozilla浏览器支持这个特性。其他浏览器按单栏显示文本，宽度等于多栏的总宽度。

Robert Bringhurst在他的著作*The Elements of Typographic Style* (Hartley and Marks, 2004年)中建议，对于多栏文本设置每行40到50个字符。为了方便，我在这个布局中采纳了他的建议。

图10-10是video.html的一部分。

Cameron Moll  
Add as contact

**Highlights from a recent vacation to Rome, Italy. Shot with a Canon HG10, edited with Final Cut Express.**

**Music is First Breath After Coma by Explosions in the Sky. Typeface is Gotham by Hoefler & Frere-Jones.**

I'm no video virtuoso. I only dabble with video as time allows, which it usually doesn't. But this is one short film I managed to get lucky with.

I've had a passion for film production ever since I was young. I saw myself first as a stuntman, then later as a film score composer. My time has passed on the first, but I hope I still have a shot at the second sometime in life. In more recent years, I've found myself behind the camera and in the editor's chair. I have much to learn about what it takes to shoot and edit a great film.

If there's one thing I've done right since

purchasing my Canon HG10 last year, it's to shoot far more film than I think I'll need. That's how I got lucky with "Roma Italia". I shot more angles than I figured I'd need, and I left the camera running longer than I assumed was adequate. It paid off in the end, leaving me with five minutes of solid shots from a total of roughly two hours of footage.

I had no script for this piece, and there were only a few shots I planned beforehand to use in whatever the final film would become. The rest was editing mojo. I analyzed the material I had, pieced it together in way I felt told a compelling story, and added effects that complimented the storytelling. "First Breath After Coma", a moving and driving piece by instrumental rock band Explosions in the Sky, was a perfect fit. Not only is it the best song title ever, but its story matches that of "Roma Italia": One of awakening, surprise, and climatic rush and release.

In the end, I hope you enjoy the story. It's a chance to see the magnificent city of Rome through my eyes. Perhaps it'll encourage you to see it through yours.

**Tags**  
[rome](#) (171)  
[roma](#) (189)  
[italy](#) (461)  
[italia](#) (393)

Add a tag

Uploaded 2 weeks ago.

图10-10 视频页面上的多栏文本布局

注意，文本分为两栏。HTML是标准代码：

```
<div id="main-video">
  <h3>
    Highlights from a recent vacation to Rome, Italy. Shot with a...</h3>

    <p>I'm no video virtuoso. I only dabble with video as time...</p>
    ...
</div>
```

而CSS不是标准的，比较特殊：

```
#main-video {
  float: left;
  margin: 40px 10px 70px;
  width: 520px;
  -moz-column-count: 2;
  -moz-column-gap: 20px;
  -webkit-column-count: 2;
  -webkit-column-gap: 20px;
}
```

为了浏览器的需要，这里同样加上了-moz-（Mozilla）和-webkit-（Webkit）前缀。注意，这里有两个属性：column-count和column-gap。这些属性很容易理解和使用——前者是需要的栏数，后者是栏之间的间距。只需指定这两个值，整个文本块就会自动地排列成指定数量的栏。还可以使用第三个属性column-rule在栏之间加边框（例如，column-rule: 1px solid #000;）。

对于在Web上使用多栏文本的实用性以及上下滚动页面时产生的问题存在一些争议，但是我相信，对于经验丰富的版面设计者，多栏布局可以提供更多Web版面设计选择。

#### 10.5.4 @font-face

Typekit (<http://typekit.com/>) 的创始人Jeffrey Veen的疑问可以很好地引出本节的内容：

“关于CSS Web字体的W3C建议[@font-face]推出已经快7年了。为什么过了这么多年Web版面设计还没有进步？为什么设计者们不愿意在设计中链接可下载的字体？” (<http://blog.typekit.com/2009/06/02/fonts-javascript-and-how-designers-design/>)

问得好，Jeffrey。当本书出版时，Jeffrey的Typekit产品很可能在相当程度上解决了他提出的问题。Typekit集中地存储字库开发商已经同意链接的字体（见图10-11），从而解决@font-face的实现和安全性问题（稍后讨论）。

简单地说，@font-face让我们可以在设计中使用任何字体显示HTML文本，而不需要考虑用户的机器上是否安装了这种字体。这通常称为字体链接或字体嵌入。不必在文档开头使用：

```
body {
  font-family: Georgia, serif;
  ...
}
```



图10-11 Typekit提供Web字体链接，消除@font-face实现和安全性问题

而是可以这样做：

```
@font-face {
  font-family: "Garamond Premier Pro";
  src: url(fonts/GaramondPremrPro.otf);
}
```

然后，按我们已经习惯的做法引用font-family：

```
h1 {
  font-family: "Garamond Premier Pro", serif;
}
```

我在输入这些代码时有点儿头晕（滑稽吗，我知道）。但是，请想象一下，可以在网站设计中使用你拥有的任何字体，而文本作为真正的HTML文本显示——不需要sIFR、Cufón或图像。现在，你很可能也有点儿头晕吧。

当然，如果事情真这么容易，我们早在7年前就开始使用@font-face了。确实有几个障碍。首先是浏览器支持问题。Safari 3和更高版本以及Firefox 3.1和更高版本支持@font-face，IE 4和更高版本也支持它。但是，IE只支持.eot（Embedded OpenType）格式，这实际上是Microsoft专

有的字体格式。只能从.ttf（TrueType）文件生成.eot文件，.otf（OpenType）等其他字体格式必须先转换为.ttf，然后再转换为.eot。难怪@font-face推广不起来了。

第二，字库开发商和字库销售商非常担心Web上的字体链接，主要是两个问题：字体文件存储在网站上，允许公共访问，很容易被下载和非法使用；许多厂商的最终用户许可协议（EULA）还没有更新以允许字体链接。

但是，也有两个好消息：出现了Typekit等新技术，可以解决这两个问题；@font-face鼓励设计者使用Arial、Georgia等非标准字体，这无疑会增加对商业字体的需求。因此，字库开发商和销售商对于字体链接和嵌入会带来的繁荣非常期待。实际上，在我撰写本章期间，有几家开发商宣布允许在Web上链接几种新字体，一些开发商甚至宣布对他们的EULA做了重大修改。

Museo Sans是由Jos Buivenga设计并于2008年发布的一种字体（见图10-12），Museo Sans 500还是免费的，这里使用的就是它。它的EULA是所有EULA中最好的，允许字体链接。（注意，我最初想使用Gotham作为字体链接的示例，Gotham就是在覆盖淡入特性图像的徽标和标题中使用的字体。但是，在开发这个网站时，Gotham的EULA不允许字体链接。）

**The Oneironauts**  
Inducing lucid dreams  
**Run 32 miles**  
SACRED GEOMETRY  
**Picking parts**

图10-12 Museo字体样本。取自MyFonts.com

在Roma Italia中，@font-face用于演示现在和不远的将来可能实现的效果。在screen.css中，在接近文档顶部的地方可以看到以下代码：

```
@font-face {
    font-family: "Museo Sans X";
    src: url(../fonts/MuseoSans_500.otf);
}
```

几个标题和顶部导航条中使用Museo Sans（见图10-13）：

```
#home h3, #home h4, #home #header h2, #home #header ul a {
    font-family: "Museo Sans X", "Lucida Grande", "Lucida Sans Unicode",
    Arial, sans-serif;
}
```



图10-13 顶部导航条中使用的Museo Sans字体

注意，我仍然指定了备用字体，以防用户的浏览器不支持@font-face。还要注意，字体的名称是Museo Sans X。在CSS中设置@font-face时，可以把font-family命名为任何名称。使用Musei Vaticani作为名称也没关系，只要引用正确的字体文件（MuseoSans\_500.otf）即可。因为Museo Sans是免费的，你的机器上可能已经安装了它。因此，我特意加上X，以确保你看到的是@font-face使用我的Museo Sans的效果，而不是使用你机器上的本地副本。

注意，我没有把Museo Sans的.otf文件转换为.eot，因此IE无法识别它。如果在IE中看到Museo Sans字体，这是Cufón而不是@font-face的效果（见下一节）。

更多信息请参见：

- 从<http://myfonts.com/fonts/exljbris/museo-sans/>下载Museo Sans；
- 关于@font-face和EOT的全面讨论，见Jon Tan的“@font-face in IE: Making Web Fonts Work”：<http://jontangerine.com/log/2008/10/font-face-in-iemaking-web-fonts-work>。

### 10.5.5 Cufón，向@font-face发展的过渡手段

我在个人网站上发表过一个关于Cufón的教程，网址是[http://cameronmoll.com/archives/2009/03/cufon\\_font\\_embedding/](http://cameronmoll.com/archives/2009/03/cufon_font_embedding/)。因此，在这个实例研究中只简要讨论Cufón。简单地说，利用Cufón可以用所选的字体显示HTML，而不需要使用任何图像或@font-face（见图10-14）。



图10-14 Cufón的字体脚本生成器

首先提一下sIFR。许多人都知道sIFR，这是一种使用Flash和JavaScript的组合实现相似效果的技术，它“把一般的浏览器文本替换为你选择的字体，无论用户的系统上是否安装了这种字体”（见<http://www.mikeindustries.com/blog/sifr/>）。Shaun Inman、Mark Wubben、Mike Davidson和其他几个人花了很长时间开发和改进IFR和sIFR，由于他们的努力，Web排版技术前进了一大步。在过去几年，`@font-face`缺少浏览器支持，字库开发商也犹豫不决，在这个时期sIFR弥补了空缺。

但是，对于许多人来说，这些技术的Flash部分往往很难设置和使用。另一方面，在网站上设置和运行Cufón只需大约5分钟。因此我个人认为，如果你无法使用字体链接，可以把Cufón当做sIFR和`@font-face`之间的过渡手段。

使用Cufón的过程如下。

- (1) 从<http://wiki.github.com/sorccu/cufon>下载Cufón脚本文件。
- (2) 使用Cufón生成器上传所选的字体，然后你会得到第二个脚本文件。
- (3) 在文档头中，添加对Cufón脚本和生成器提供的字体脚本的引用，比如：

```
<script src="js/cufon-yui.js" type="text/javascript" charset="utf-8"></script>
<script src="js/Museo_400.font.js" type="text/javascript" charset="utf-
8"></script>
```

还应该在body结束标签前添加以下代码，以避免在IE中出现闪烁问题：

```
<script type="text/javascript">Cufon.now();</script>
```

另外，在文档头中指定哪些HTML元素或选择器应该替换为你的字体，比如：

```
<script type="text/javascript">
  Cufon.replace('h1');
</script>
```

或

```
<script type="text/javascript">
  Cufon.replace('h1')('h2')('blockquote');
</script>
```

(4) 如果在使用Cufón的网站上使用jQuery等JavaScript框架（Roma Italia包含jQuery），Cufón会利用框架的选择器引擎，因此可以指定特有的选择器，比如：

```
<script type="text/javascript" charset="utf-8">
  Cufon.replace('#header h2,#header ul a');
</script>
```

(5) 在CSS文件中，按与其他文本相同的方式，修改由Cufón替换的文本的样式——`color: #333;`、`font-size: 12px;`、`text-transform: uppercase;`等。

就这么简单。当前，IE 6、IE 7和IE 8、Firefox 1.5和更高版本、Safari 3和更高版本、Opera 9.5和更高版本以及Google Chrome支持Cufón。这个实例研究网站同时使用Cufón和@font-face，让你可以对比这两种技术。注意，在许可协议方面Cufón与@font-face面对相同的问题——所选字体的EULA必须允许在Web上进行字体嵌入。

注意，因为IE不读取@font-face使用的.otf字体文件，我把Cufon.replace放在了一个条件注释中。因此，对于这个实例研究，在IE中Cufón替代@font-face。如果想看看Cufón在其他浏览器中的效果，只需删除条件注释，它会覆盖@font-face。

## 10.6 用AJAX和jQuery增加交互性

几年前，我在一个工作室首次谈到AJAX时，问在场的人是否开发过使用AJAX网站和应用程序，当时只有几个人举手。如果现在问这个问题，本书的许多读者可能都会举手。如果问的是jQuery而不是AJAX，结果可能差不多。

的确，AJAX和jQuery在复杂网站和小项目中都非常流行，它们差不多成了在Web上实现富交互性的事实上的标准技术组合。当然，可以单独使用它们，但是常常结合使用。Adobe Flex和Microsoft Silverlight等新技术正在挑战它们的地位，但是我认为在至少几年内AJAX和jQuery仍然是Web上的主流技术。

本节并不打算详细讲解这两种技术，已经有许多出色的书、教程和博客文章讨论它们。这里只简要介绍AJAX和jQuery以及在Roma Italia中是如何使用它们的。如果你熟悉这些技术，可以跳到10.6.3节。

### 10.6.1 AJAX

AJAX是Asynchronous JavaScript and XML的缩写，通常包含至少3个部分：

- 异步的服务器通信，通常通过XMLHttpRequest实现；
- 通过操纵DOM（Document Object Model，文档对象模型）进行动态显示和交互；
- 使用JavaScript把所有部分组合在一起。

异步通信是AJAX（或任何富Internet技术）的关键特性，它在Web环境中提供与本机应用程序相同的机制。传统的请求和响应模型通过完整的服务器往返通信获取整个页面，而异步通信只获取页面的一部分所需的数据（例如，在注册账号时检查用户名的可用性）。

在Roma Italia中，通过使用一些JavaScript并从几个静态PHP页面提取数据，模拟异步服务器通信：

- imageLoad.php用于淡入和淡出的大图像；
- search.php用于搜索框自动补全特性。

由于异步通信是模拟的，因此你可以下载代码示例并在运行PHP的任何机器上打开这个界面，不需要真正的服务器通信。

在讨论jQuery之后，我们将讨论自动补全特性的代码。

### 10.6.2 jQuery

Karl Swedberg和Jonathan Chaffer的*Learning jQuery*<sup>①</sup>（Packt Publishing, 2007年）把jQuery描述为“用于Web脚本编程的通用抽象层”。我喜欢把它看做“适合脚本编程新手的JavaScript”。

jQuery让我们可以：

- 在DOM中移动；
- 修改页面的外观；
- 动态地修改页面的内容。

而且实现这些不需要编写任何JavaScript。更好的是，它按照CSS语法在文档中使用选择器作为创建交互性的钩子。

我为了解jQuery的组件编写了另一个网站，现在看看其中的一个示例。这也是一个虚构的网站，网址是`http://cameronmoll.com/articles/widget/`。

我们看看页面顶部的黄色通知条中的Dismiss按钮（见图10-15）。单击它时，黄色条会慢慢向上滑动，直到看不见为止。

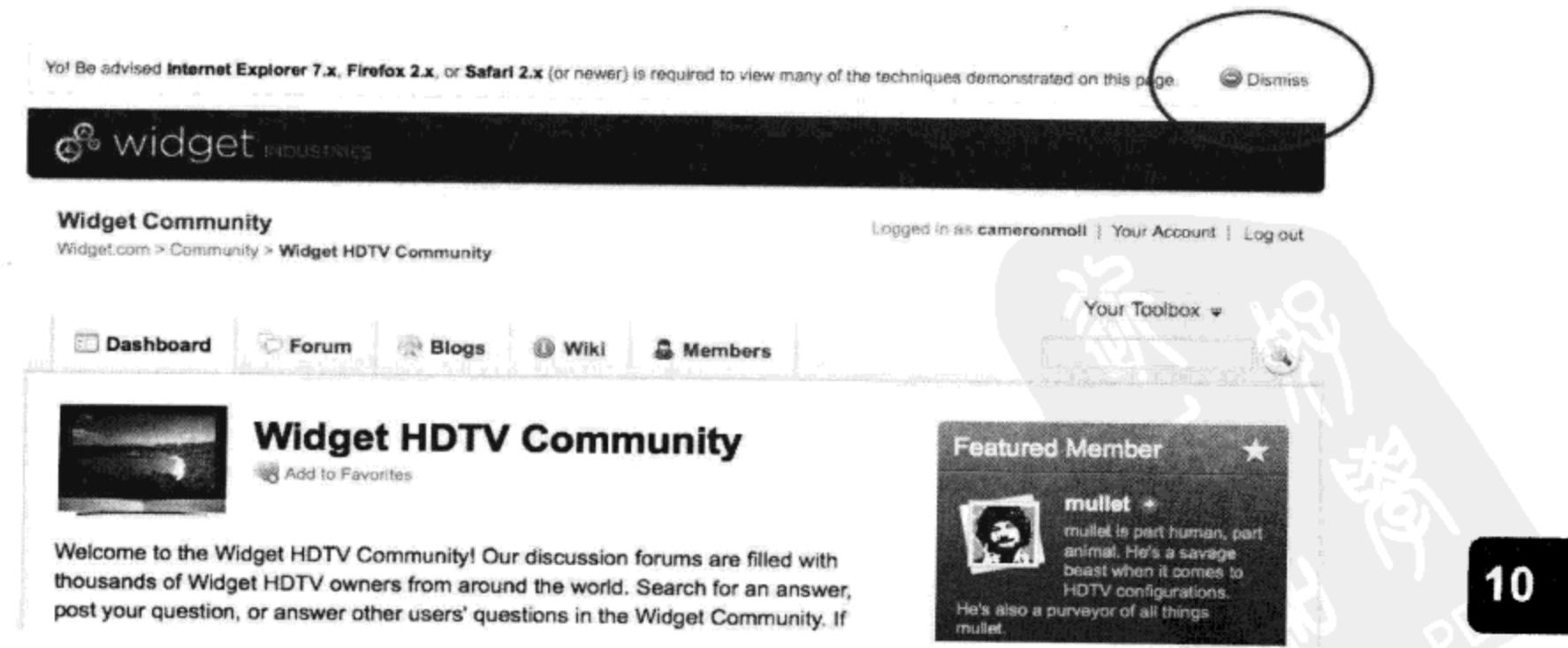


图10-15 为我的“The Highly Extensible CSS Interface”系列创建的Widget演示网站

① 本书第二版中文版《jQuery基础教程（第2版）》已经由人民邮电出版社出版。——编者注

下面是在按钮的锚标签中添加的代码：

```
$('#alert').slideUp('slow');
```

下面介绍各个部分。

- `$( )`：这个基本jQuery构造（或函数）用于选择文档的一些部分。在这个示例中，选择的是ID为`alert`的元素。
- `.slideUp`：这是一个jQuery方法。方法实质上是大量JavaScript代码的“快捷方式”。显然，`slideUp`方法的作用是让所选的元素（`#alert`）向上滑动。
- `('slow')`：这个预定义的字符串指定方法的执行方式。在这里，它要求元素慢慢地向上滑动。

此代码可以直接添加在行内，也可以添加在单独的.js文件中（或动态地添加），后一种方法比较好。但是，真正的优点是：不需要为此做任何准备工作。构造、方法和字符串都是jQuery预先构建的。我只需知道我希望让元素慢慢地向上滑动，然后在jQuery库中查找与所需的动画和运动效果对应的引用。

### 10.6.3 使用AJAX和jQuery实现搜索

回顾了AJAX和jQuery的基本知识之后，我们看看Roma Italia中是如何使用AJAX和jQuery的。我们要讨论的特性是搜索特性。在这个实例研究前面，我们看到了如何使用属性选择器寻找搜索表单中的特定元素。现在，我们来看看其他部分。

搜索特性实际上是整个网站中代码最复杂的特性之一。当用户输入关键字查询时，放大镜图标替换为表示正在加载的图标，同时显示匹配的结果。这有时候称为即时搜索。AJAX和jQuery共同作用实现这个交互效果。尽管比较复杂，但是把各个部分拼在一起并不太难。

首先，下面是标记：

```
<form action="#" method="get" accept-charset="utf-8">
  <fieldset>
    <legend></legend>
    <label for="search-input">Search</label>
    <input type="text" id="search-input" name="search" value="" title="Search">
    <input type="image" name="" src="img/search-go.gif">
  </fieldset>
</form>
```

要切换正在加载图标，我能想出的最简便的方法是，把`input`框的背景图像和正在加载的图标组合成一个动画GIF（见图10-16）（我在其他网站上还没有见过这么做的）。使用CSS根据交互的状态定位这个图像，随着用户输入上下移动图像，从而在放大镜图标和正在加载的图标之间切换。

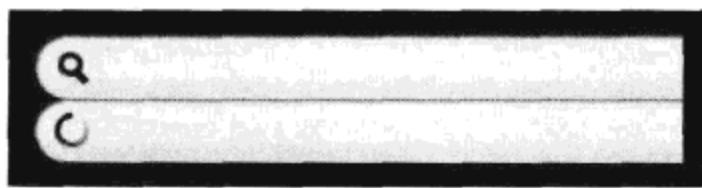


图10-16 search-bg.gif是一个动画GIF，包含两个状态

下面是input框的CSS：

```
#header form input[type="text"] {
    ...
    padding: 6px 0 0 28px;
    height: 20px;
    background: url(..../img/search-bg.gif) no-repeat;
}
```

在默认情况下，背景图像定位在左上角，高度限制为20像素。添加6像素的顶部内边距，这意味着只显示这个图像中高26像素的部分——正好是图像高度的一半儿。因此，只显示放大镜部分。

当用户开始输入时，会发生几件事。首先，用户每输入一个字符，就会调动4个文件：autocomplete.css、jquery.plugins.js、jquery.autocomplete.js和search.php。在开始输入时，在input框中动态地添加class="ac\_input"，在即时搜索显示结果时删除它。这个选择器在autocomplete.css中，样式设置如下：

```
.ac_loading {
    background: url(..../img/search-bg.gif) no-repeat 0 -26px !important;
}
```

注意，背景图像现在定位在距离顶部-26px的地方，这会把图像向上移动，露出下半部分（正在加载图标）。这向用户表明正在异步地从服务器（search.php）获取数据。

第二，在显示正在加载图标的同时，与search.php交换数据，寻找与用户输入的字符匹配的结果。打开search.php，会看到一些词条，比如Ancient Ostia、Ancient Rome、Arch of Constantine等。

第三，把匹配的结果发送回页面，在input框下面显示一个像select的菜单，其中显示匹配的结果（见图10-17）。这个菜单实际上只是一个无序列表（ul），是由jquery.autocomplete.js和jquery.plugins.js生成的，由autocomplete.css提供样式。然后，用户可以用鼠标或键盘选择一个匹配，也可以继续输入并按Enter键。这样这个交互就完成了。

jquery.autocomplete.js包含几百行代码，但这是jQuery的另一个优点——这些代码都不是我编写的。它是社区编写的一个jQuery插件，还有许多这样的插件。实际上，这个实例研究中的大多数jQuery功能都是插件提供的。自动补全插件是jQuery Autocomplete，可以在<http://bassistance.de/jquery-plugins/jquery-plugin-autocomplete/>找到它。

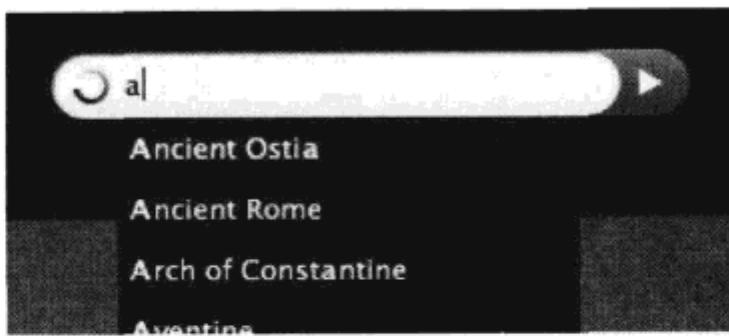


图10-17 完成的搜索特性

请参考以下资料和教程：

- Jeremy Keith的*Bulletproof Ajax*: <http://bulletproofajax.com/>
- Ajaxian.com: <http://ajaxian.com/>
- DHTML Site上的*Ajax Tutorials and Scripts*: <http://dhtmlsite.com/ajax.php>
- jQuery官方网站: <http://jquery.com/>
- Digital Web杂志的jQuery课程: [http://www.digital-web.com/articles/jquery\\_crash\\_course/](http://www.digital-web.com/articles/jquery_crash_course/)
- Simon Willison的*jQuery for JavaScript programmers*: <http://simonwillison.net/2007/Aug/15/jquery/>
- Web Designer Wall的*jQuery tutorials for designers*: <http://www.webdesignerwall.com/tutorials/jquery-tutorials-for-designers/>
- Noupe 的 50 多个 jQuery 教程: <http://www.noupe.com/jquery/50-amazing-jquery-examples-part1.html>
- jQuery的240种插件: <http://www.sastgroup.com/jquery/240-plugins-jquery>

## 10.7 小结

现在，你了解了Roma Italia使用的许多技术。还有许多技术没有介绍——深入研究代码，你会有更大的收获。

注意，这个网站的文件相当大，尤其是脚本和主题图像。但是，如果这是一个真正的网站，可以使用实际的AJAX加载图像，可以简化脚本。例如，`jquery-1.3.2.js`大约为120KB，但是经过简化和压缩，它只有19KB。（可以从`jquery.com`下载压缩版。）这些优化技术会显著减小页面的总大小。

但是，这个实例研究说明的最有意义的一点是，原始的HTML标记完全能够实现漂亮的设计效果。如果禁用所有样式，用户仍然可以顺利地阅读和浏览这个网站。尽管在Web设计者看来可能不漂亮，但是有意义的轻量标记对于屏幕阅读器、搜索引擎机器人等软件很有意义。总之，这个网站可以把两个领域的优势结合起来——优美的视觉设计和优雅的源代码。



## 第 11 章

# 实例研究：Climb the Mountains

Simon Collison

在本书前一个版本中，我介绍了我的More Than Doodles实例研究，讨论了设计者可以使用的多种设计手段。当时，我们正处于关键时期，行业中各种Web标准都在发展。在那个令人兴奋的时期，我们使用CSS 2.1创建了极好的布局，但是一些老式浏览器给我们带来了许多问题。

在3年多的时间，通过实现CSS 3规范中的技术，布局设计已经方便多了。我们可以把一些装饰性背景图像替换为border-radius和box-shadow等CSS 3规则的组合；由于出现了RGBa，我们可以更好地控制透明层，而不必借助于半透明的背景图像。更重要的是，由于改进是渐进式的，在不支持CSS 3及其工具和技术的浏览器上，我们仍然可以提供相当简洁的用户体验。

在这个实例研究中，我们将学习：

- HTML和CSS的组织和约定；
- 网格灵活性；
- 根据body类突出显示当前页面；
- 用伪类和相邻同胞选择器寻找元素；
- 通过组合类增强功能和灵活性；
- RGBa、border-radius和box-shadow属性；
- 定位列表项和显示内容。

### 11.1 关于这个实例研究

这个实例研究采用一个固定的XHTML结构，尽可能简单、组织良好且强大。具体地说，

XHTML不包含任何只用于关联CSS的额外标记。标记中的东西都是必需的，没有多余的。因此本章的目标是，展示如何使用已有的东西和CSS选择器寻找特定的XHTML元素，而不需要使用额外的div、清理div等不必要的东西。

Climb the Mountains（登山，后面简称为CTM）是一个专门为勇敢的登山者、徒步旅行者和攀岩者设计的虚构的Web应用程序，这些人最大的爱好是离开温暖舒适的家，在野外漫游几小时、几天甚至几周，寻找大自然的乐趣。CTM是一个社交网站，为成员提供交流的机会（见图11-1）。

The screenshot shows the homepage of the Climb the Mountains website. At the top, there's a navigation bar with links for Home, Routes, About, and Shop. A user account section indicates 'Logged in as collogic' with options to 'ACCOUNT' or 'LOG OUT'. Below the navigation is a quote by John Muir: "Climb the mountains and get their good tidings. Nature's peace will flow into you as sunshine flows into trees. The winds will blow their own freshness into you, and the storms their energy, while cares will drop away from you like the leaves of Autumn." attributed to JOHN MUIR, 1895.

The main content area features a large map showing mountain peaks with elevations: Scafell Pike (9,781 ft), Great Gable (9,123 ft), Lingmell (8,442 ft), and Scafell (8,299 ft). To the right of the map is a scenic photograph of a composite bridge over a valley with mountains in the background, with the caption: "From the composite bridge towards the village, Great Gable and Lingmell."

Below the map, there are sections for 'Your latest route' (Scafell Pike via Corridor Route, 12th May 2009), 'Your other routes' (Kinderscout circuit, Great Gable, Snowdon Horseshoe, Kinderscout circuit), 'Recommended for you' (Win Hill and Ladybower), and 'Members' routes' (Kinderscout ridgewalk circuit, Castleton Ridges Walk, Branston and Eaton villages, Ilkley Moor and Otley).

At the bottom, there are sections for 'From the forums' (Help! I am lost, Best boots for under £100, Scrambling in Wales, Do you take your dog walking?, I'm too old for this!) and 'Flickr' (a grid of nine thumbnail images of mountain scenes). The footer includes links for Climb the mountains, About, Contact, Help, Privacy, Terms, FAQ, API, and copyright information: 'Copyright © 2009 Simon Collison. All rights reserved.' and 'This is a fictitious website created strictly for non-commercial purposes. It may not be reproduced without permission. Buy CSS Mastery at Amazon.'

图11-1 Climb the Mountains主页

它的一个关键功能是，把GPS路线上传和导出到成员的GPS设备，它在存档的每个路线中添加详细的统计信息。除了这些数据之外，每个路线还附有照片、地图、下载和相关信息，信息体系结构（IA）中的数据很丰富。在这里，我们可以使用一些灵巧的CSS实现各种效果，最终确保简洁地显示所有信息和图像，同时整个布局仍然非常灵活。

这个设计分成许多信息块，这使我们更容易把注意力集中在正在讨论的领域，而不是过分关注装饰性样式。通过本章中对多种技术的分析，你有望掌握如何在设计中应用这些技术和本书讨论的其他特性。

非常感谢我的同事Greg Wood在登山知识方面提供的帮助。所有照片都来自我自己的Flickr账号，大多数是今年早些时候在英国著名的Lake District拍摄的。采用的字体包括Palatino系列的各种字体、比较常用的Helvetica、Georgia和老式的Verdana等，以及默认的Arial或Times New Roman。

这个实例研究的网址是<http://www.climbthemountains.com/cssm/>。

## 11.2 样式表的组织和约定

毫无疑问，我参与的每个网站都需要在坚固的“地基”上建设起来，这就是约定包。在过去两年，我和我的同事一起开发了一套基本的规则和约定，它们作为编写HTML、CSS、JavaScript和ExpressionEngine的起点。它规定了连接的CSS文件、命名约定、模块、插件和库脚本，确保由任意团队成员领导或参与的任何项目都坚持统一的约定，让任何人在任何时候更容易参加项目的开发（见图11-2）。经过不断的发展，这个包已经成了我们最基本的工具之一。

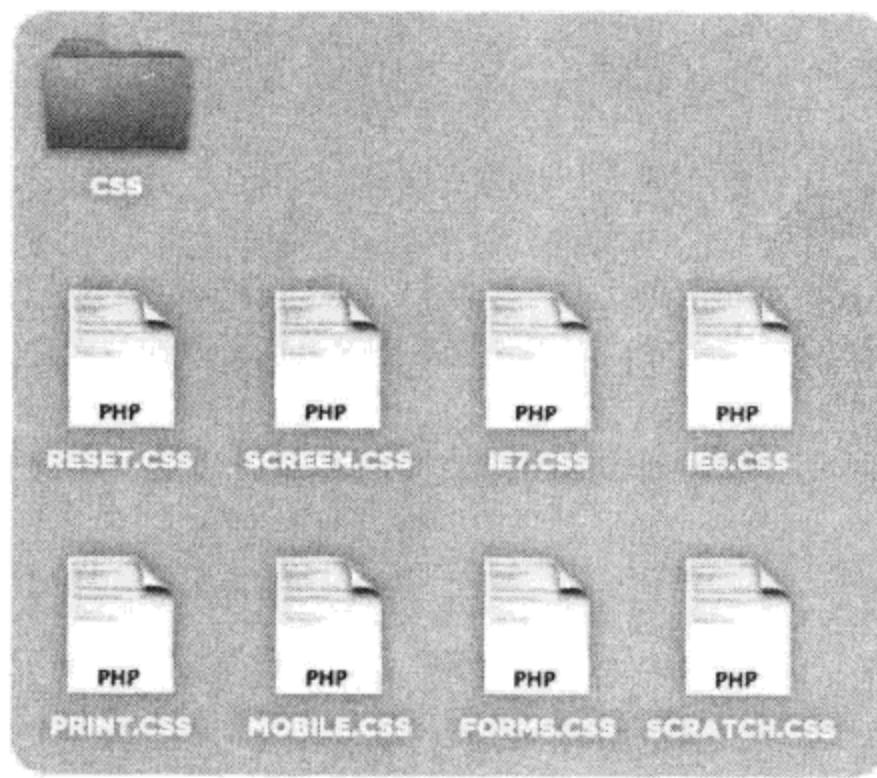


图11-2 基本文件包中的典型样式表集

由于特别关注CSS，我们使用了一套CSS，这提供了更好的灵活性，便于应对浏览器的不一致，还让团队成员能够通过自己的草稿文件提交工作成果。通过使用草稿文件，团队成员可以添加自己的CSS改进方案或规则，这些规则根据它们在层叠中的位置在浏览器中显示。如果项目主管认可这个CSS，就把它从草稿文件中取出，添加到相关的主样式表中，替代现有的声明。我们还有非常基本的打印样式表和手持设备样式表，所有表单使用一个单独的样式表。这就是我们采用的方式。

### 11.2.1 screen.css

如果使用Mac浏览器或者PC上的IE 8或Firefox，那么只需要screen.css样式表和reset.css样式表即可，screen.css样式表包含CTM案例研究所需的所有声明。如果使用IE 6或IE 7，那么还需要IE专用的样式表，这将在下一节介绍。

链接screen.css文件的代码如下：

```
<link href="assets/css/screen.css" type="text/css" rel="stylesheet" media="screen">
```

后面会讨论screen.css样式表中使用的许多方法，但是先来谈谈我认为非常必要的两个措施。

#### 内容说明

设计者往往认为这个部分是不必要的，因而容易忽视甚至忽略它。毕竟，即使没有描述性说明，CSS仍然会起作用。但是再想一下，如果你作为一个大型团队的成员开发网站，会怎么样？如果样式表常常相当大，又会怎么样？怎么保证其他人可以轻松地参与你的设计？怎么保证所有内容组织良好？

这就是，样式表说明（尤其是内容介绍）有价值的地方。请记住，可以按以下语法把任何普通说明添加在样式表中的任何地方：

```
/* I am a simple note */
```

可以使用这种方式提供最新的样式表内容目录。这样，团队中的其他设计者和开发人员很容易确认他们是不是要查看这个样式表，可以快速地查明他们需要的规则是否在这个样式表中。

```
/*
CLIMB THE MOUNTAINS by SIMON COLLISON
VERSION 1.0
```

```
CONTENTS -----

```

- 1.BODY
- 2.DEFAULT STYLING
- 3.HEADING
- 4.LINKS

```

5. IMAGES
6. LAYOUT
7. BRANDING/MASTHEAD
8. NAVIGATION
9. SITEINFO/FOOTER
10. GLOBAL ELEMENTS
    10.1 CAPTIONED IMAGE
    10.2 ELEVATION
    10.3 DISTANCE/ELEVATION PARAGRAPH
11. HOMEPAGE
    11.1 CONTENT PRIMARY
    11.2 CONTENT SECONDARY
    11.3 CONTENT TERTIARY
*/

```

具体的布局由设计者自己或团队决定。在前面的示例中，我使用由换行符和制表符组成的结构创建了一个非常清晰的内容目录。重要的是，在样式表中添加规则、移动其中的规则或删除规则时，要相应地修改内容目录。

### 11.2.2 **reset**

`reset.css`样式表的用途是跨所有浏览器和设备创建一个一致的“舞台”。例如，一些浏览器的默认样式表可能设置不同的`margin`值、`padding`、标题字体大小、`line-height`等。

我们通过`screen.css`中的以下代码链接`reset.css`样式表：

```
@import url(reset.css); /* RESET CSS */
```

这样就可以确保（对于大多数`reset.css`）我们要处理的XHTML元素没有设置`margin`、`padding`、`line-height`、`font-size`等。现在，可以放心地在`screen.css`中应用我们需要的值，不必担心从浏览器样式表继承值。

CSS专家Eric Meyer把`reset.css`样式表称为“起点而不是无法触碰的黑箱”(<http://meyerweb.com/eric/tools/css/reset/>)，CTM网站使用他提供的`reset.css`样式表，但是我自己做了一点儿调整和补充。

### 11.2.3 使用条件注释的IE样式表

区分特定Microsoft浏览器版本的方法最初是在IE 5和IE 5.X中引入的。通过把XHTML标记放在XHTML注释内的条件语句中，可以把特定的信息发送给特定的浏览器。可以在XHTML文档中的任何地方使用这种特殊的语法组合。

```
<!--[if IE 6]> Anything here is only seen by IE6 <![endif]-->
```

在下面的示例中，使用这种语法组合分别为使用IE 6、IE 7或IE 8的用户调用不同的样式表。

```
<!--[if IE 6]><link href="assets/css/screen-ie6.css" type="text/css" rel="stylesheet" media="screen" /><![endif]-->

<!--[if IE 7]><link href="assets/css/screen-ie7.css" type="text/css" rel="stylesheet" media="screen" /><![endif]-->

<!--[if IE 8]><link href="assets/css/screen-ie8.css" type="text/css" rel="stylesheet" media="screen" /><![endif]-->
```

这种做法的优点是，可以在IE样式表中创建与浏览器相关的规则，避免在（由所有其他浏览器使用的）screen.css现有的CSS中添加IE专用的代码。CTM设计就采用这种方法，因为我们需要对CSS做许多IE专用的调整（后面将讨论这些调整措施）。

### 11.3 网格灵活性

网格可以作为任何网站中任何页面的基础。应该使用网格解放你，而不是限制你。不要害怕突破网格和经验。网格应该作为提示、指导和提供信心的依据。

网格通常有预先确定的宽度，包含指定数量的列和可选的隔离带。网格是你最好的朋友，可以帮助你克服困难。它是Photoshop和CSS之间的“中间人”，可以帮助你做出与浮动、定位、外边距、内边距、边框等相关的初步布局决定。

与其他许多设计者一样，无论是使用Photoshop、Fireworks还是浏览器本身进行原型设计，我都使用一个可以随意开关的网格层（见图11-3）。

### CTM布局的工作方式

为了让对网格的讨论更切合实际，Climb the Mountains实例研究布局采用一个健壮而灵活的宽1000像素的网格。在这个宽1000像素的空间中有12列，每个列由隔离带分隔。每个列的宽度为65像素，每个隔离带的宽度为20像素，见图11-4。

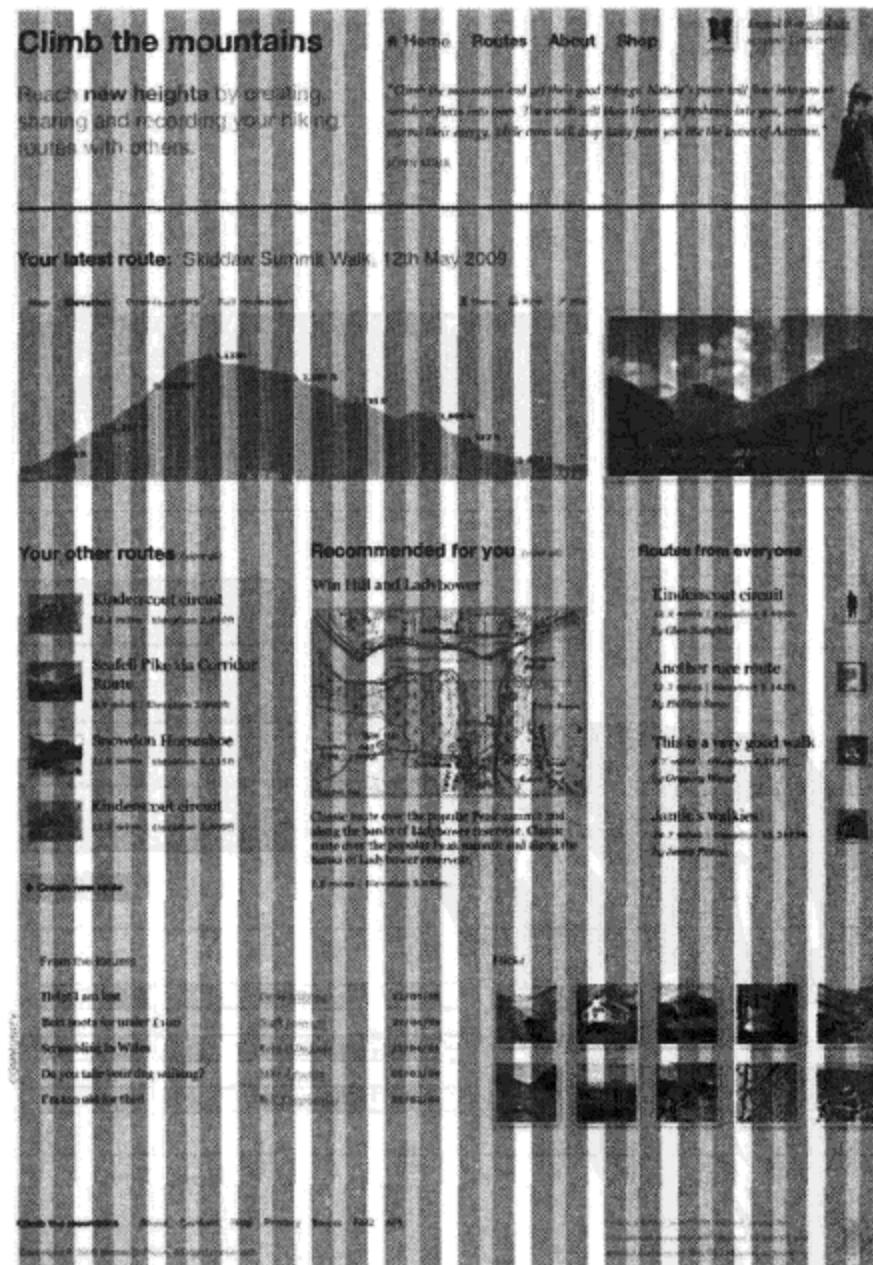


图11-3 Climb the Mountains和它的底层列网格



图11-4 列布局

每个列还有自己的结构。在宽65像素的列中，有3个子列，（从左到右）宽度分别为25、15和25像素（见图11-5）。

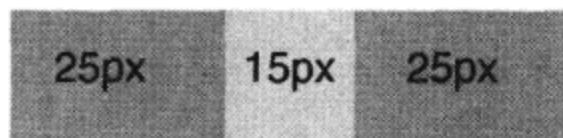


图11-5 内部子列宽度

这些子列在某种程度上像是网格中的网格。它们提供额外的参照点，如果12列不够细，不足以满足我们的需要，元素可以按子列确定尺寸或位置。仔细看看全屏示例图，可以看到一些元素与主列对齐，而一些元素与子列对齐。

## 11.4 用 **body** 类控制导航

可以使用分配给body元素的值和CSS改变页面布局、控制行为以及进行其他重要的修改等。在本书的第一版中，我使用每个页面的body元素的唯一ID控制布局，使用类确定位置，比如`<body id="threeColLayout" class="home">`。这一次，我只使用类。

```
<body class="home">
```

无论是使用ID还是类，这个超级父元素提供的功能是相同的。

### 11.4.1 突出显示当前页面

有许多种突出显示用户当前所在的页面的方法，许多设计者使用PHP脚本切换CSS，如果当前在主页上，就突出显示主导航中的Home链接。这很酷，但是一些CSS应用程序只依靠body类和导航类就很容易实现这种效果。我们来看一下。

```
<body class="home">
```

这个页面是主页。现在要确保每个导航项都有相关的类：

```
<ul id="navigation_pri">
  <li class="nav_home"><a href="#">Home</a></li>
  <li class="nav_routes"><a href="#">Routes</a></li>
  <li class="nav_about"><a href="#">About</a></li>
  <li class="nav_shop"><a href="#">Shop</a></li>
</ul>
```

在前面的代码片段中，Home链接的类是nav\_home。还在Routes页面中添加了一个body类，以便后面测试突出显示效果：

```
<body class="routes">
```

接下来，使用CSS对导航列表应用样式。注意，我们以绝对定位方式把这个元素放在距主容器的左上角指定距离（top和left）的坐标上。

```
ul#navigation_pri {
    list-style:none;
    margin:0;
    position:absolute;
    top:0;
    left:415px;
    font-size:19px;
    font-weight:bold;
    font-family:Helvetica,Arial,sans-serif;
}
ul#navigation_pri li {
    float:left;
    margin:0;
    padding:30px 10px 0 10px;
    height:3000px;
}
ul#navigation_pri li a {
    color:#000;
}
ul#navigation_pri li a:hover,
ul#navigation_pri li a:focus {
    color:#333;
    text-decoration:underline;
}
```

这会产生图11-6所示的基本导航样式，但是其中没有突出显示当前查看的页面。（注意，本节后面将讨论位于导航层之上的blockquote。）

**Home Routes About Shop**



图11-6 基本的主导航

下一步是使用选择器定义body类和导航home类之间的关系。注意，我们把用于主页和Routes页面的两个相同的规则分组在一起：

```
.home ul#navigation_pri li.nav_home,
.routes ul#navigation_pri li.nav_routes {
    background-color:#f5f5f5;
}
```

选择器的第一部分（.home或.routes）确保用户正在查看此页面。ul#navigation\_pri元素必须是.home或.routes的子元素，才会应用样式。如果找到匹配，就应用样式。这会产生填满整个导航项区域的浅灰色背景。

接下来，可以为链接行为定义样式，同样把.home和.routes的规则分组在一起：

```
.home ul#navigation_pri li.nav_home a,
.routes ul#navigation_pri li.nav_routes a {
    color:#278dab;
    background:#f5f5f5 0 center no-repeat;
    padding:0 0 0 20px;
}
.home ul#navigation_pri li.nav_home a:hover,
.home ul#navigation_pri li.nav_home a:focus,
.routes ul#navigation_pri li.nav_routes a:hover,
.routes ul#navigation_pri li.nav_routes a:focus {
    text-decoration:none;
    color:#000;
}
```

这样就得到了所需的蓝色文本链接。最后，可以只在主页链接上添加一些装饰。当查看主页时，在Home链接左边显示一个小房子图标：

```
.home ul#navigation_pri li.nav_home a {
    background-image:url(../images/site/nav_back.gif);
}
```

这些代码在主页上产生的效果见图11-7。

**↑ Home Routes About Shop**

*"Climb the mountains and get their good tidings. Nature's peace will flow into you as sunshine flows into trees. The winds will blow their own freshness into you, and the storms their energy, while cares will drop away from you like the leaves of Autumn."*

JOHN MUIR, 1903



图11-7 选择了Home链接

图11-8显示的是Routes页面。

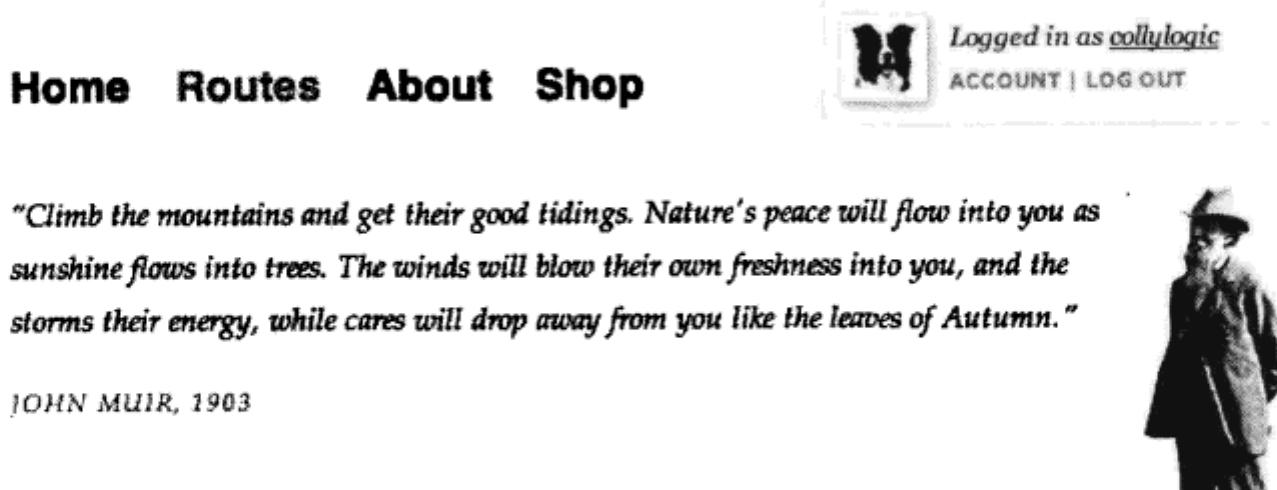


图11-8 选择了Routes链接

#### 11.4.2 控制 **blockquote** 所处的层

现在，回到位于主导航之上的John Muir语录。我真的很喜欢这段话——它激励我无论天气如何都要走出去。不要忘记走出去，到真实的世界中去！在CTM上页面的这个部分，我们通过显示语录鼓励访问者参与登山活动。下面是标记：

```
<blockquote id="johnmuir">
  <p>&ldquo;Climb the mountains and get their good tidings. Nature's peace ↔
  will flow into you as sunshine flows into trees. The winds will blow their ↔
  own freshness into you, and the storms their energy, while cares will drop ↔
  away from you like the leaves of Autumn.&rdquo;</p>
  <p><cite>John Muir, 1903</cite></p>
</blockquote>
```

除了使用专门的字符实体表示引号之外，这里没有任何不常见的东西，我们可以开始定义样式了。最初，我们使用position: absolute把ul#navigation\_pri绝对定位到距顶边0px、距左边415px的位置。

```
ul#navigation_pri {
  list-style:none;
  margin:0;
  position:absolute;
  top:0;
  left:415px;
  font-size:19px;
  font-weight:bold;
  font-family:Helvetica,Arial,sans-serif;
}
```

现在可以定义blockquote的样式：

```
div#branding blockquote {
    width:505px;
    float:right;
    padding:0 70px 20px 0;
    background:url(..../images/site/branding_johnmuir.jpg) no-repeat right top;
}
```

但是，它会被蓝色的导航项区域盖住，见图11-9。

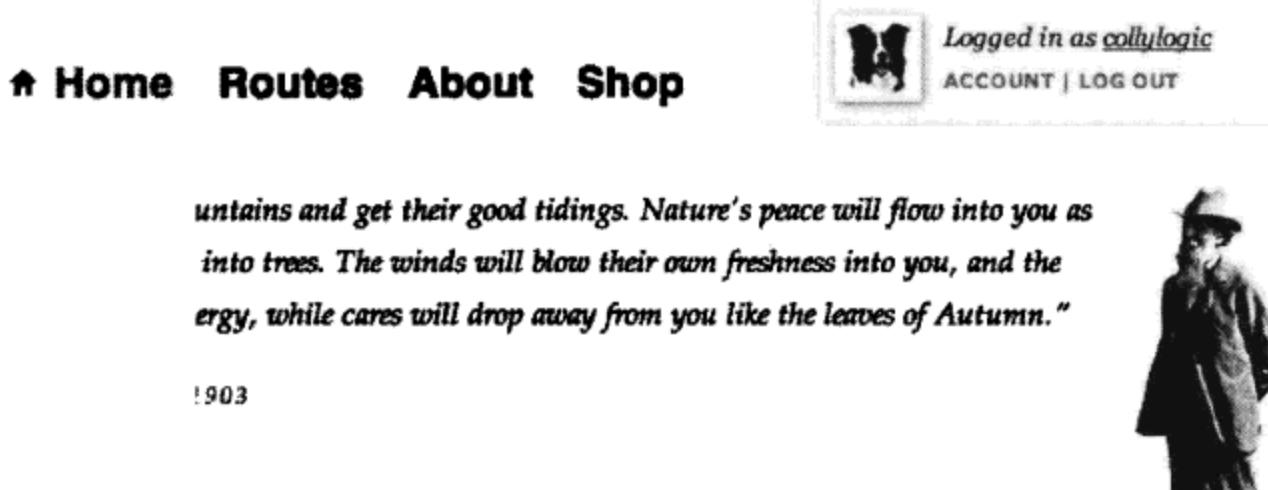


图11-9 blockquote的一部分隐藏在Home链接后面

但是，考虑到导航是绝对定位的，如果在blockquote中添加position:relative，就可以让它出现在蓝色导航项区域之上：

```
div#branding blockquote {
    position:relative;
    width:505px;
    float:right;
    padding:0 70px 20px 0;
    background:url(..../images/site/branding_johnmuir.jpg) no-repeat right top;
}
```

这样blockquote就会出现在主导航之上，如前面的导航示例见前面的图11-7。

## 11.5 战略性地选择元素

在前一节中，我们讨论了如何使用后代选择器根据每个页面的body类控制主导航的样式。现在，再看看如何通过合理的标记方法为深层后代选择器提供更强的灵活性和控制能力，以及这种方法如何为通过高级元素选择实现更强的控制能力提供基础。

### 11.5.1 深层后代选择器

首先，看一下基本结构。在CTM实例研究的右边，可以看到浅黄色的Members' Routes面板，

这里提供了网站的其他用户提交的路线。标记中没什么特别值得注意的东西，但是在讨论选择器之前先分析一下标记的结构。

注意，我们在每个无序列表项中包含几个元素，比如h3、p和img。我经常吃惊地发现许多设计者不知道可以在li元素中添加各种元素。在li元素中常常只有a和img，但是可以添加的元素远不止这些。

```
<div id="others_routes">
  <h2>Members' routes <a href="#" class="more">(view all)</a></h2>
  <ul>
    <li>
      <h3><a href="#">Kinderscout circuit</a></h3>
      <p class="dist_elev">13.6 miles | Elevation 2,400ft</p>
      <p class="username"><a href="#">from Glen Swinfield </a></p>
    </li>
    <li>
      <h3><a href="#">Castleton Ridge Walk</a></h3>
      <p class="dist_elev">12.2 miles | elevation 1,343ft</p>
      <p class="username"><a href="#">from Phil Swan </a></p>
    </li>
    <li>
      <h3><a href="#">Branston Circular</a></h3>
      <p class="dist_elev">5.7 miles | elevation 1,213ft</p>
      <p class="username"><a href="#">from Gregory Wood </a></p>
    </li>
    <li>
      <h3><a href="#">Ilkley Moor and Otley</a></h3>
      <p class="dist_elev">24.7 miles | elevation 2,473ft</p>
      <p class="username"><a href="#">from Jamie Pittock </a></p>
    </li>
  </ul>
</div>
```

通过这样组织内容，可以把信息块整理成列表的形式，提供我们喜欢的列表的所有层次和样式控制能力。

这样，很容易使用基本的选择器选择包含div的others\_routes中的无序列表项以及li元素中的各个元素。注意，我们使用border-radius、-webkit-border-radius和-moz-border-radius规则在ul元素上应用圆角，这个实例研究后面会讨论这些属性。

```
div#others_routes ul {  
    list-style:none;  
    border:1px solid #dedeaf;  
    background:#ffffcc;  
    border-radius:5px;  
    -webkit-border-radius:5px;  
    -moz-border-radius:5px;  
    margin:0;  
    padding:10px;  
}  
div#others_routes ul li {  
    margin:0;  
    padding:10px 55px 10px 0;  
    position:relative;  
    border-bottom:1px solid #dedeaf;  
    border-top:1px solid #fff;  
}  
div#others_routes ul li h3 {  
    margin-bottom:5px;  
}  
div#others_routes ul li img {  
    position:absolute;  
    top:10px;  
    right:10px;  
}  
div#others_routes ul li p.username {  
    margin:3px 0 0 0;  
    font-style:italic;  
    font-size:12px;  
}  
div#others_routes ul li p.username a {  
    color:#666;  
}  
div#others_routes ul li p.username a:hover,  
div#others_routes ul li p.username a:focus {  
    text-decoration:underline;  
}
```

在前面的标记中，我们用一些简单的后代选择器选择了比较深层的HTML元素。例如，可以使用

很好。成员提交的路线的列表组织得不错，看起来相当简洁，大多数人可能对此很满意。但是等一下！我们是追求完美的人，而且我们有强大的CSS。为什么不再改进一下呢？

在后两个示例中，我们要使用CSS整理路线容器的顶部和底部。

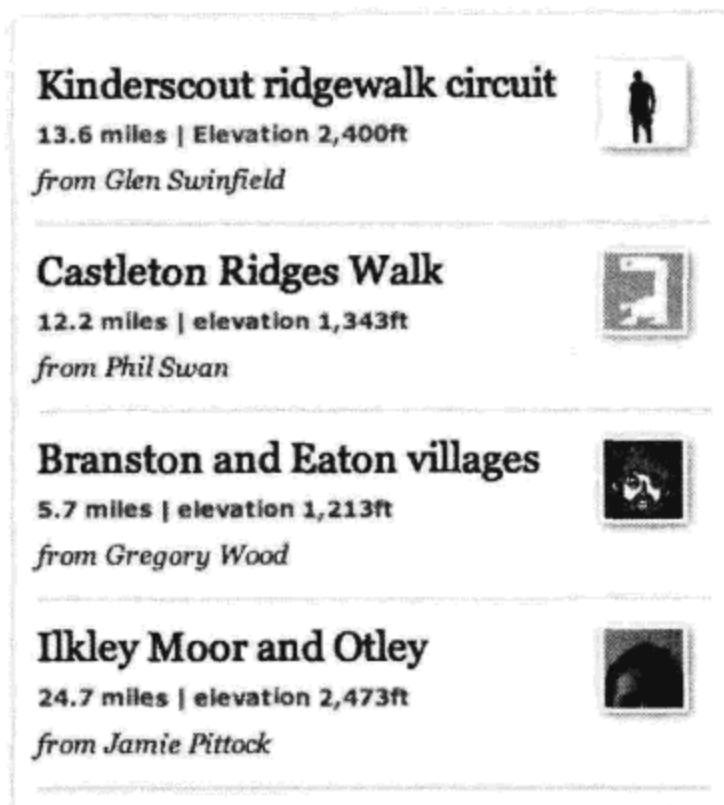


图11-10 最初的Members' Routes容器

### 11.5.2 :first-child 伪类

在选择文本块的第一个字母或第一行时，设计者可能会使用`:first-letter`或`:first-line`等伪类。这些技术让我们可以根据简单的逻辑对元素应用样式。

`:first-child`伪类只选择包含元素的第一个子元素。

在这个实例研究中，成员路线的容器是一个无序列表，每个列表项包含路线的详细信息。每个`li`元素有相同的内边距，顶部和底部有细的边框。

```
div#others_routes ul li {
    margin:0;
    padding:10px 55px 10px 0;
    position:relative;
    border-bottom:1px solid #dedeaf;
    border-top:1px solid #fff;
}
```

这会让列表的内容均匀地间隔，但是我希望减少顶部列表项（在这个示例中，这是Kinderscout ridgewalk circuit路线）的内边距量。实际上，我不希望顶部有任何内边距，也不希望有边框。

这就需要使用`:first-child`伪类。在这里，我们创建一个新规则，使用相同的选择器选择包含`div`的`#others_routes`中的无序列表项，但是在`li`元素后面加上`:firstchild`，这实际上表示“进入这个容器，找到无序列表，只在找到的第一个`li`元素上执行以下样式覆盖。”

```
div#others_routes ul li:first-child {
    padding-top:0;
    border-top:none;
}
```

如图11-11所示，Kinderscout ridgewalk circuit项现在没有顶边框和顶内边距，紧贴在父容器的顶边。

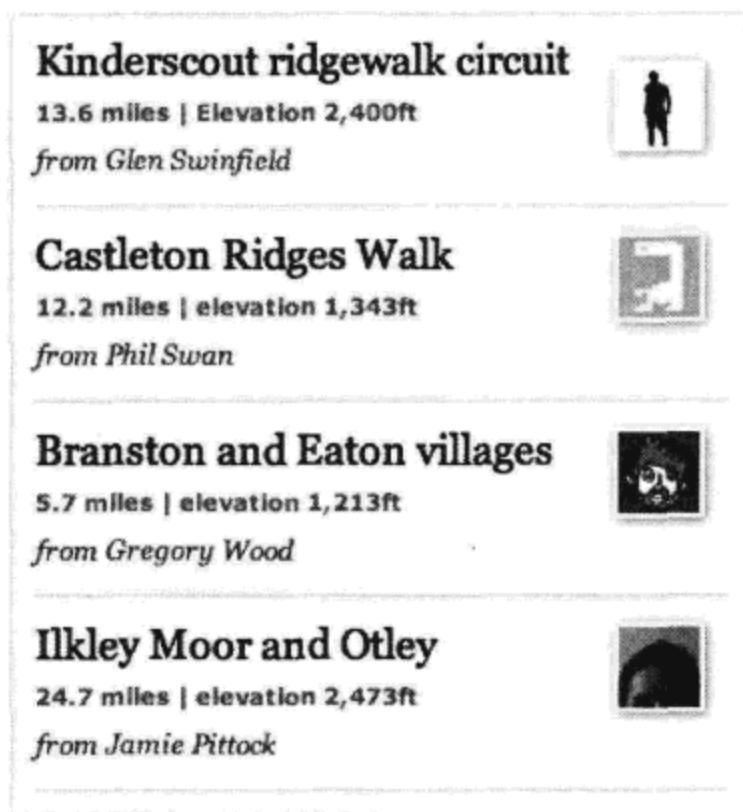


图11-11 成功地去除了顶内边距和顶边框

这非常简单，但它是非常强大的选择某一元素的方法，可能有许多用途。既然已经处理了列表的顶部，现在看看底部的处理。

### 11.5.3 相邻同胞选择器

刚才介绍了:`:first-child`，现在介绍一下:`:last-child`的作用，这个伪类选择特定父容器中的最后一个子元素。使用方法与`:first-child`相同，你可以自己试试。但是，只有最近的浏览器版本支持这个方法，比如Safari、Firefox、Google Chrome和Opera，所以需要为IE 6、IE 7和IE 8提供替代方法，这需要使用相邻同胞选择器。

在这个示例中，我们需要做的与刚才用`:first-child`伪类做的事情相反。正如前面提到的，每个无序列表项都有顶部和底部内边距和边框。我们成功地关闭了第一个`li`元素的这些样式，现在需要为最后一个元素关闭它们。

但是，怎么做呢？样式表怎么知道哪个元素是容器中的最后一个元素，我们怎么准确地选择

它。这需要用一点儿技巧。相邻同胞选择器包含多个选择器，由“+”组合符分隔。它匹配与第一个元素相邻的下一个同胞元素。注意，元素的父元素必须相同，第二个元素必须紧挨着第一个元素。

因此，与：`:first-child`示例一样，再次选择包含`others_routes`的父`div`，然后用选择器向下寻找，直到到达希望应用样式的元素。我们的无序列表总是只有4个`li`元素，所以可以使用以下代码：

```
div#others_routes ul li + li + li + li {  
    padding-bottom:0;  
    border-bottom:none;  
}
```

这里创建的选择器的意思是“在`others_routes` `div`中寻找无序列表，找到第四个`li`元素，只对它应用样式”。

因此，结果（见图11-12）是第四个`li`元素没有底内边距和底边框，这样更加紧凑，信息更突出。只需使用CSS选择器就能实现这样的效果。

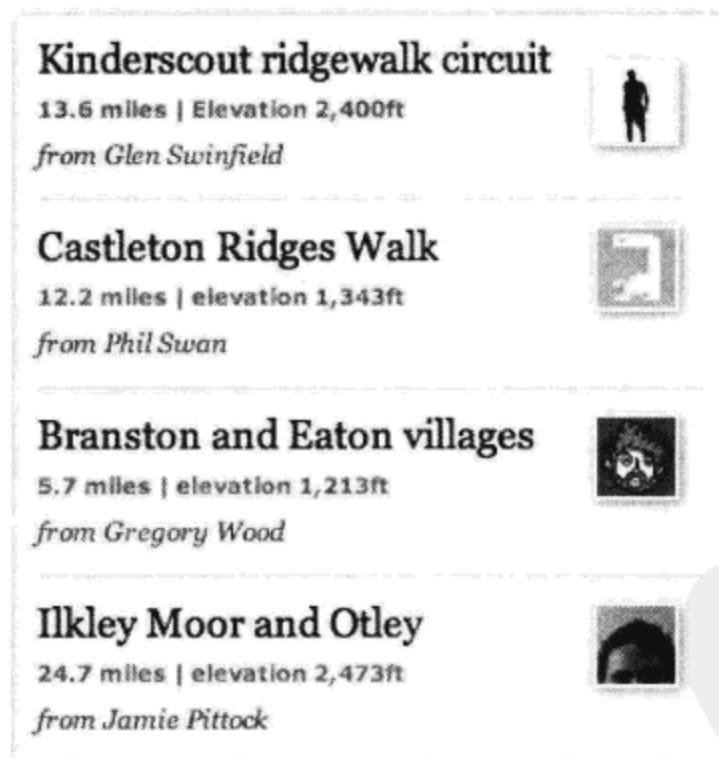


图11-12 成功地去除了底边框和内边距

## 11.6 透明度、阴影和圆角

在本书的第一版中，我的实例研究大量使用了圆角框。所有人都希望用圆角加强视觉效果，直角太简单，有点儿沉闷。

有无数种创建圆角的方法，我过去一直采用的方法是结合使用大量JavaScript代码、几个背景图像和一些额外的标记。这种方法很笨拙，但是当时确实没有更好的方法。

我写这一章时非常郁闷，一直想大喊：“CSS 3！”情况已经变了：用户的期望变了，工具也发展了。当然，浏览器还没有完全跟上形势（不出所料），但是我们勇于也乐于尝试新的思想，推动行业的发展。

在本节中，我要在CTM主页的图像和说明上实现一些出色的CSS 3效果，而不使用任何JavaScript、多余的图像或标记。万岁！

### 11.6.1 我们的目标

我们使用一个简单的 $310 \times 185$ 像素的JPG，campsite.jpg（见图11-13）。然后，在图像上覆盖半透明的灰色区域，在这个区域中显示白色的说明文本。然后在图像上应用Polaroid风格的照片边框，让它有完美的圆角和逼真的投影。实现这些效果只需要使用CSS。



图11-13 最初的campsite.jpg图像

标记非常简单。图像和说明需要包含在一个div中，在本例中它名为captioned\_image。在段落中设置class="caption"，这让我们可以直接选择它。

```
<div class="captioned_image">
  
  <p class="caption">From the campsite bridge towards the village, Great Gable and Lingmell.</p>
</div>
```

标记就位之后，就可以试试3种CSS 3技术了。

## 11.6.2 说明图像覆盖和RGBa透明度

有多种指定颜色的方法。许多人用十六进制的RGB三原色值（3个十六进制值）指定颜色。其他人喜欢使用某些颜色的英文名称。还可以使用RGB百分数或小数。下面的示例都可以有效地表示红色：

```
color: #f00  
color: #ff0000  
color: red  
color: rgb(255,0,0)  
color: rgb(100%, 0%, 0%)
```

RGB代表红色、绿色和蓝色，大多数设计者都熟悉这个概念。RGBa引入了第四个信道——处理透明度的alpha信道。CSS 3的优点是，我们可以继续用RGB指定颜色，还可以用第四个小数值设置颜色的alpha透明度。可以使用0.0（完全透明）到1.0（完全不透明）的任何值。

在下面的示例中，仍然用RGB设置红色，通过把alpha透明度声明为0.5，设置50%的透明度。

```
color: rgb(255,0,0,0.5)
```

RGBa值只分配给我们声明的元素，任何子元素都不继承透明度。这与透明度属性明显不同，透明度属性总是会继承。

在CTM网站上，对照片和说明应用以下声明：我们对容器div进行相对定位，然后对说明进行绝对定位，让它定位在图像上我们希望的位置。

```
div.captioned_image {  
    position: relative;  
}  
div.captioned_image p.caption {  
    position: absolute;  
    bottom: 0;  
    left: 0;  
    margin: 0;  
    color: #fff;  
    font-size: 13px;  
    line-height: 16px;  
    font-style: italic;  
    padding: 5px;  
}
```

接下来，把RGBa值声明为rgba(0,0,0,0.5)，其中前3个值表示黑色，alpha透明度值0.5设置中等透明度，可以调整透明度，直到我们对总体效果满意为止。

```
div.captioned_image p.caption {  
    position: absolute;
```

```
bottom:0;  
left:0;  
margin:0;  
background:rgba(0,0,0,0.5);  
color:#fff;  
font-size:13px;  
line-height:16px;  
font-style:italic;  
padding:5px;  
}
```

这会产生我们想要的说明覆盖效果，见图11-14。

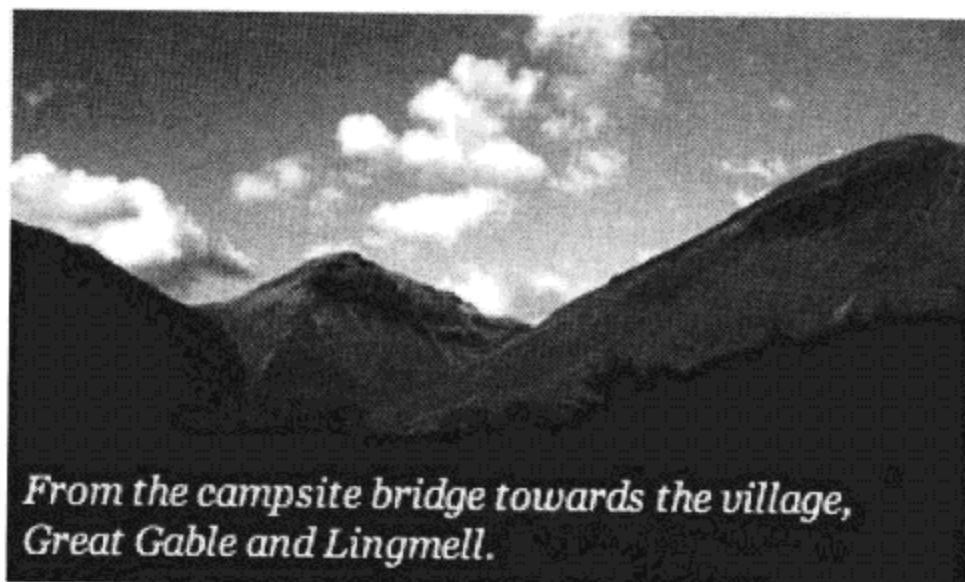


图11-14 透明的说明

与许多令人兴奋的CSS 3新技术一样，一些浏览器不支持alpha透明度，尤其是IE（包括当前的IE 8）。例如，IE 7默认采用合理的不透明层，就像是在提供透明的PNG图像，但不实施alpha透明度支持时看到的效果（见图11-15）。

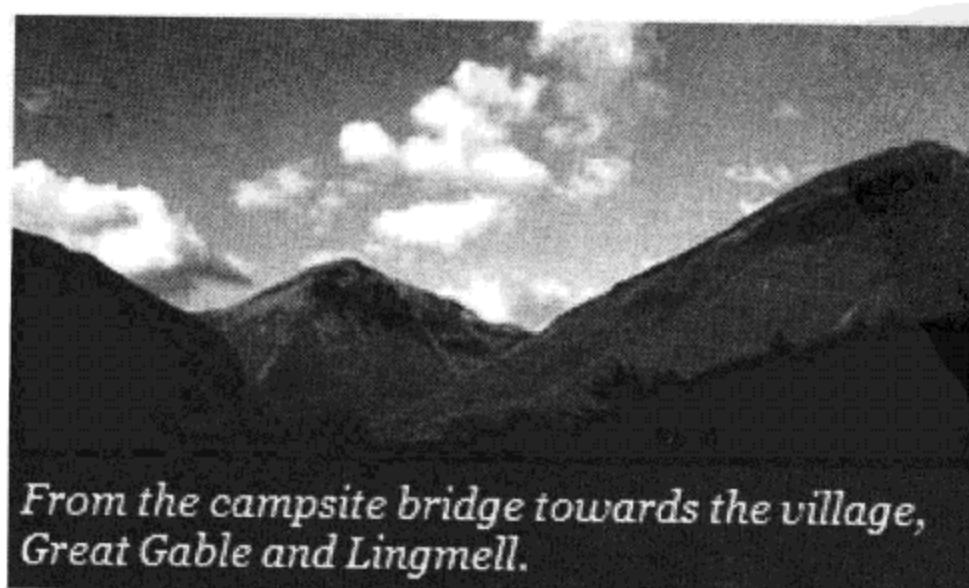


图11-15 IE 7显示的说明覆盖

IE 8仍然不支持RGBa，它把说明文本直接显示在图像上，不加任何背景。为了解决这个问题，可以在screeni-ie8.css样式表中添加一个规则，确保在文本背后放上灰色背景。

```
div.captioned_image p.caption { background:#666; }
```

IE及其缺陷不应该阻碍我们尝试新技术。我希望所有喜欢使用十六进制三原色值的设计者在布局中的各种元素中试试RGBa。你不会后悔的，代码会清晰得多！

### 11.6.3 组合类

我吃惊地发现一些设计者不知道可以通过组合类为元素提供更大的灵活性。

例如，你可能在任意页面中多次使用class="profile"，然后分配统一的颜色和布局信息。但是，假设你希望只根据某一变量（比如用户是成员还是访客）修改背景颜色。不需要通过创建两个profile样式来指定不同的颜色，只需在单独的规则中设置颜色，把这些规则与profile类组合起来。

```
.profile {  
width:300px;  
margin:0 10px;  
padding:10px;  
font-size:11px;  
}  
.guest {  
background-color:#ff9900;  
}  
.member {  
background-color:#ff0000;  
}
```

然后，可以使用组合的类根据用户状态应用样式。可以组合任意数量的类，只需用空格分隔每个类，如下所示：

```
<div class="profile member">  
<p>Member options...</p>  
</div>
```

在CTM网站上，可以使用组合的类只在某些带说明的图像上添加框架。注意，除了captioned\_image之外，还添加了polaroid类：

```
<div class="captioned_image polaroid">  
  
<p class="caption">From the campsite bridge towards the village, Great Gable and Lingmell.</p>  
</div>
```

现在可以定义这个polaroid框架的样式，这需要用到CSS 3规范中的几种技术。

### 11.6.4 border-radius

在本书第一版中，Andy和我详细介绍了一种在图像上添加框架和阴影的技术，这种技术很有用但是比较麻烦。这种技术使用几个div和背景图像，需要非常小心地设置样式和位置。在2005年只能这么做。

现在有了border-radius属性，它让我们可以只使用CSS声明给像素加上圆角。但是，IE根本不支持这个属性，所以在IE中显示直角，这对于我是可以接受的效果。当前，正在发展的Opera浏览器也是这样。

到撰写本章时，还没有主流浏览器保证支持标准的border-radius属性，因此要想使用这个属性，在短期内还需要添加两个声明——一个声明用于基于Mozilla的浏览器，比如Firefox，一个声明用于基于WebKit的浏览器，比如Safari（它还支持椭圆角）。在<http://www.the-art-of-web.com/css/borderradius/>可以找到更多信息、示例和技巧。

下面定义这3个声明：

```
.polaroid {
    border: 5px solid #f00;
    border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
}
```

注意，直接看圆角效果不容易看清楚，所以我指定了一个临时的红色边框，这样可以看出图11-16中的变化。可以看到每个角上有半径为5px的圆弧。这实际上是用圆的四分之一，定义了准确的角。与外边距、内边距和边框一样，有4个border-radius属性（分别针对元素的一个角）和一个简写属性。

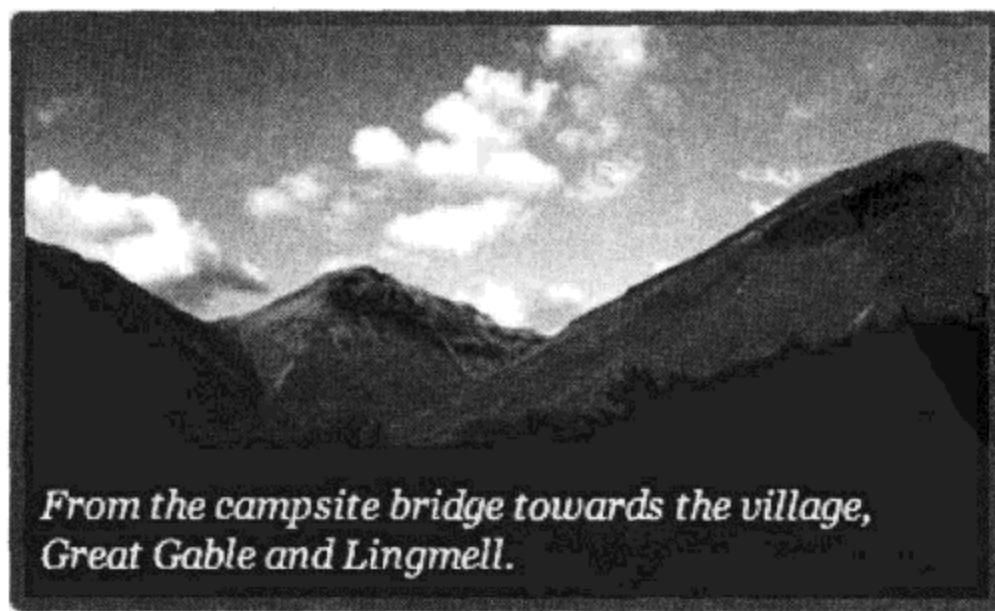


图11-16 使用红色边框清楚地显示圆角

这比第一版中介绍的方法简单多了。实现圆角之后，现在可以考虑应用简单的投影，让图像增加一点儿真实感。

### 11.6.5 box-shadow

CSS 3提供一种创建投影的简单方法，Safari 3和更高版本以及Firefox 3.5和更高版本支持它。这个属性的参数是3个长度值（水平偏移、垂直偏移和模糊半径）和一个颜色值。

如果对阴影的水平偏移应用正值，阴影就出现在元素的右边。负偏移让阴影出现在元素的左边。

如果对垂直偏移应用负值，阴影就出现在元素的上面。正值让阴影出现在元素的下面。

模糊半径确实很方便。如果值设置为0，阴影就是清晰的，值越高，阴影越模糊。

在polaroid类的规则中添加以下设置，就可以用4个值创建投影，投影出现在有说明文字的图像的右下方，模糊半径为5像素，颜色为中等灰色：

```
.polaroid {  
    border:5px solid #fff;  
    border-radius:5px;  
    -webkit-border-radius:5px;  
    -moz-border-radius:5px;  
    -webkit-box-shadow:1px 1px 5px #999;  
    -moz-box-shadow:1px 1px 5px #999;  
}
```

box-shadow会考虑到border-radius值的影响，所以阴影会与圆角的图像框架完美地融合，见图11-17。

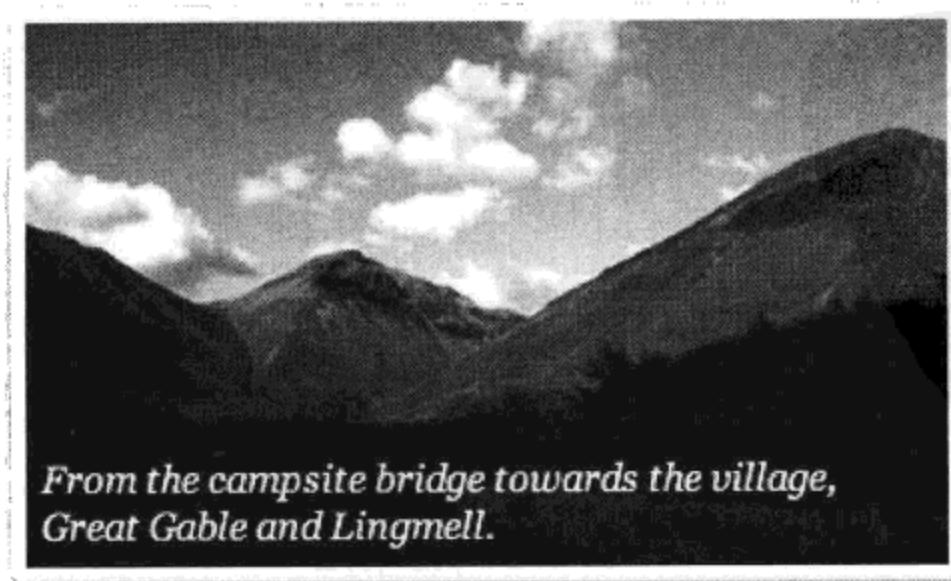


图11-17 图像现在有说明、框架和圆角

当然，在IE中还不是这样的效果，见图11-18。

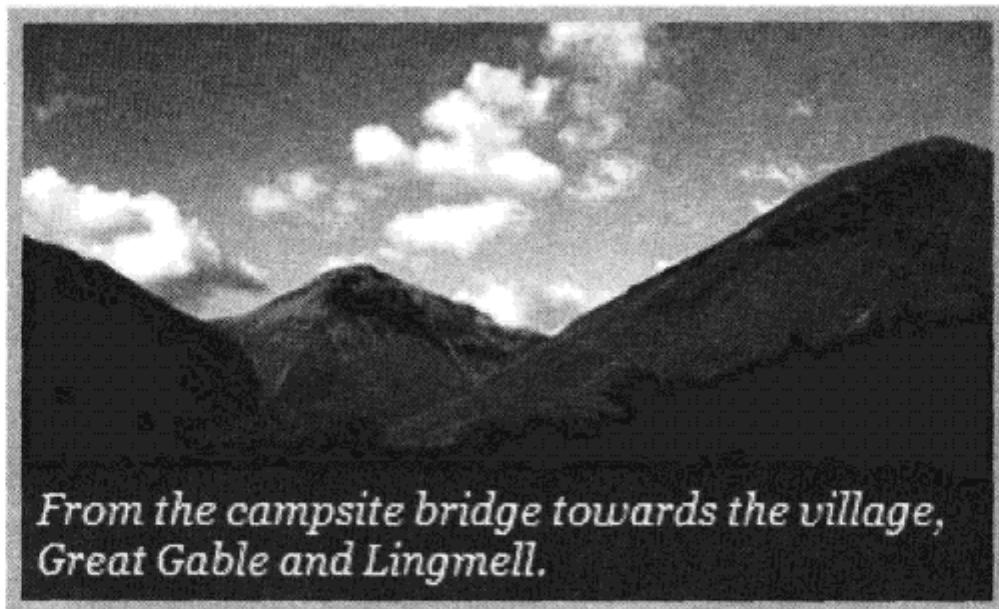


图11-18 IE不支持CSS 3属性，但结果是可以接受的

前面提到过，IE不支持说明上的RGBa透明度，而且框架显示为灰色的直角框架。框架的宽度也是5px，但不是白色的，也没有圆角。显然，也没有漂亮的阴影。请记住，我们在screen-i8.css样式表中添加了一个规则，从而确保说明文本至少有灰色背景。真不知道要等到什么时候IE才能跟上形势！

## 11.7 定位列表和显示内容

在本节中，我们看看布局左边的Your latest route区域。这个区域提供与特定路线相关的统计数据、地图和图表，可以通过不同的标签查看每个面板。

首先，为CTM主页的统计数据部分添加导航。标记需要两个列表，一个用于左边的统计数据标签，另一个用于右边的Share、Print和Edit选项：

```
<ul id="route_nav">
  <li><a href="#">Map</a></li>
  <li class="cur"><a href="#">Elevation</a></li>
  <li><a href="#">Download GPS</a></li>
  <li><a href="#">Full routesheet</a></li>
</ul>
<ul id="route_action">
  <li class="share"><a href="#">Share</a></li>
  <li class="print"><a href="#">Print</a></li>
  <li class="edit"><a href="#">Edit</a></li>
</ul>
```

注意，我们在Elevation标签中添加了class="cur"，这样就可以通过这个额外的类直接选择

这个链接。

由于这两个无序列表从文档中其他地方继承了一些样式，所以已经有基本的蓝色链接状态及 font-family 和 font-size 规则，见图11-19。

- Map
- Elevation
- Download GPS
- Full routesheet
  
- Share
- Print
- Edit

图11-19 基本的“Your latest route”导航列表

在其他地方，我已经把 Share、Print 和 Email 列表浮动到了右边。因此，从现在开始我们只关注左边的列表。我们将把它浮动到左边，添加几个基本样式声明，包括为选择的标签设置边框颜色：

```
ul#route_nav {  
    list-style:none;  
    font-family:Verdana,sans-serif;  
    font-size:11px;  
    font-weight:bold;  
    float:left;  
    margin:0;  
}  
ul#route_nav li {  
    float:left;  
    margin:0;  
    padding:7px 10px;  
}  
ul#route_nav li a {  
    color:#666;  
}  
ul#route_nav li a:hover,  
ul#route_nav li a:focus {  
    color:#333;  
}  
ul#route_nav li.cur a {  
    color:#000;  
}
```

这会让两个列表产生图11-20所示的效果——一个向左边浮动，另一个向右边浮动，还有一些基本样式。



Map Elevation Download GPS Full routesheet Share Print Edit

图11-20 两个列表分别向左右浮动

我们将在这个导航后面的统计数据容器上使用`clear:both`, 从而清除这些浮动, 接下来, 在选择的类上添加背景颜色:

```
ul#route_nav li.cur {
background:#dff1f1;
}
```

这突出显示选择的标签的准确区域, 见图11-21。



Map Elevation Download GPS Full routesheet Share Print Edit

图11-21 两个列表分别向左右浮动, 选择的标签突出显示

这就是基本的Your latest route导航效果, 但是还可以使用另一种巧妙的CSS 3技术。

### 11.7.1 圆角

在前面, 我们讨论过CSS 3的`border-radius`属性, 用它为图像创建了简单的圆角。

现在, 我们将使用相似的属性在一个基本形状(具体地说, 是导航中选择的列表项)上添加圆角。

我们使用与浏览器相关的`border-radius-top-left`和`border-radius-top-right`属性变体只在顶部应用圆角:

```
ul#route_nav li.cur {
background:#dff1f1;
-moz-border-radius-topleft:3px;
-webkit-border-top-left-radius:3px;
-moz-border-radius-topright:3px;
-webkit-border-top-right-radius:3px;
}
```

CSS这个简单的调整会在Firefox和Safari中实现更优美的标签, 见图11-22。



Map Elevation Download GPS Full routesheet Share Print Edit

图11-22 选择的标签现在有顶部圆角, 这只需使用CSS

这个导航完成了, 现在看看更有意思的海拔数据。

### 11.7.2 主海拔图

在Your latest route导航条下面，我们现在要添加统计面板，Elevation实例研究在默认情况下显示海拔图。

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

这里要注意的最重要的一点是（前面介绍过），我们使用route\_elevation div清除导航列表的浮动。因此，这个div的样式中首先定义clear:both:

```
.home div#route_elevation {  
    clear:both;  
    background:#dff1f1 url(../images/site/elevation_home.gif) 0 bottom no-repeat;  
    position:relative;  
    height:195px;  
}
```

elevation div也是相对定位的，这有助于后面在图上放置li元素（稍后添加这些元素）。我们还应用背景图像elevation\_home.gif（见图11-23），用一个简写的规则指定color、image、position和repeat属性。

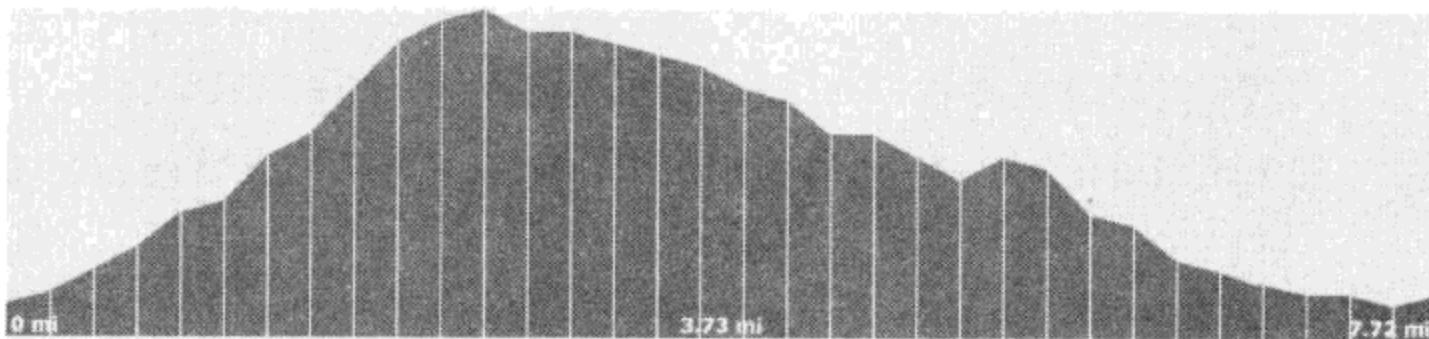


图11-23 海拔图背景图像

这显示了图11-24所示的导航数据和海拔图像的组合。

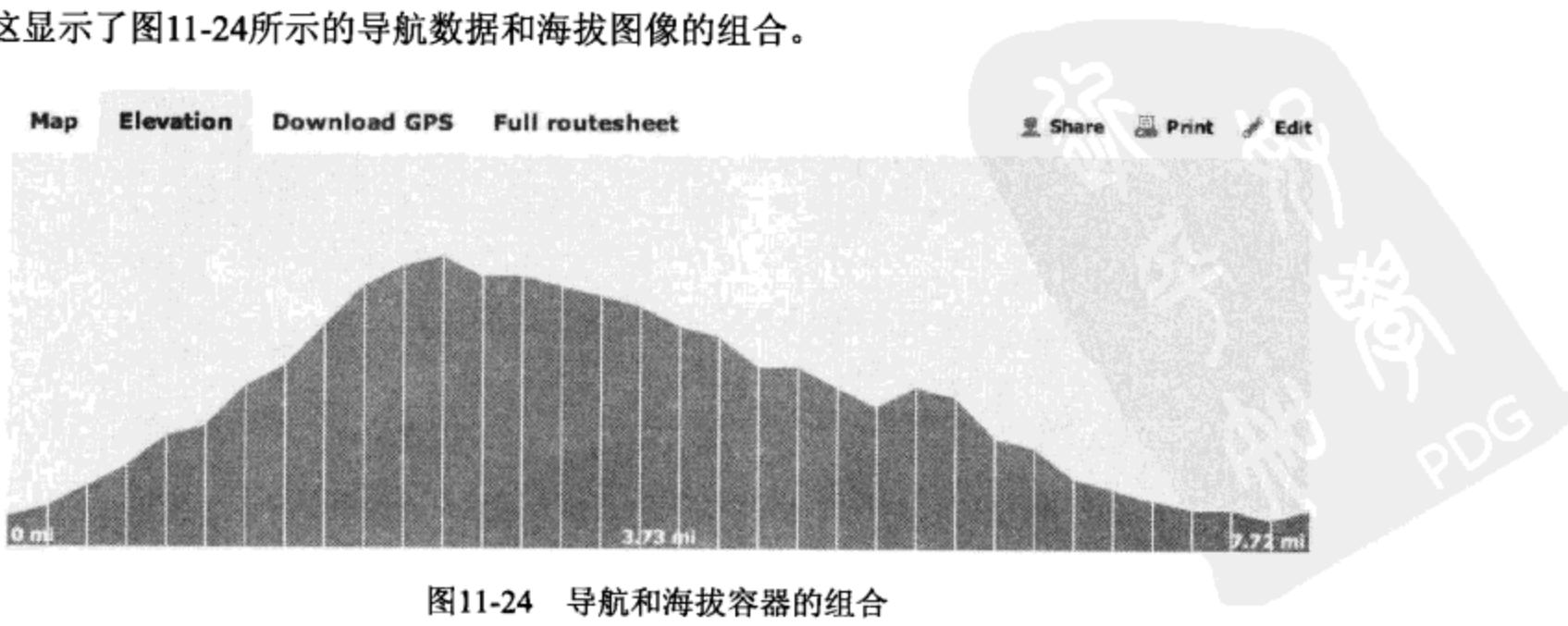


图11-24 导航和海拔容器的组合

接下来，可以创建一个新的无序列表，它将包含海拔参照点。每个参照点提供高度和来自Flickr的相关图像。注意，每个列表项有一个唯一的类，比如marker\_01和marker\_02。

```
<div id="route_elevation">
  <ul>
    <li class="marker_01">
      <a href="#">
        <strong>1,442 ft</strong>
        
      </a>
    </li>
    <li class="marker_02">
      <a href="#">
        <strong>3,133 ft</strong>
        
      </a>
    </li>
    <li class="marker_03">
      <a href="#">
        <strong>2,398 ft</strong>
        
      </a>
    </li>
    <li class="marker_04">
      <a href="#">
        <strong>1,286 ft</strong>
        
      </a>
    </li>
  </ul>
</div>
```

我们将使用这些唯一的类定义每个列表项在图上的显示位置（即到海拔图容器的top和left的坐标）。

```
.home div#route_elevation li.marker_01 {
  top:123px;
  left:97px;
}
.home div#route_elevation li.marker_02 {
  top:50px;
  left:237px;
}
.home div#route_elevation li.marker_03 {
```

```

    top:95px;
    left:377px;
}
.home div#route_elevation li.marker_04 {
    top:137px;
    left:517px;
}

```

但现在还是初期阶段，所以目前我们只需要把一组列表项排列在容器的左边，每个列表项包含高度和图像（见图11-25）。

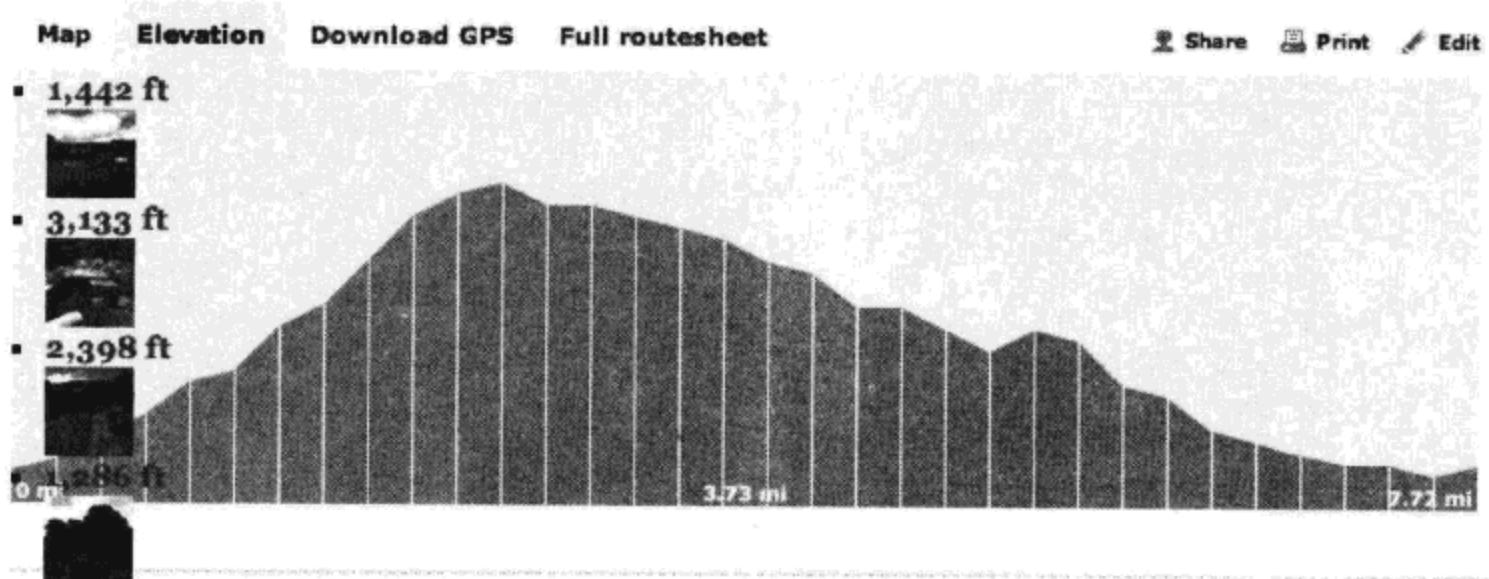


图11-25 列表项在左边排成一行

在把列表项定位到所需的位置之前，我们先隐藏图像，选择一个更合适的标志图形。我们要使用的图形是典型的项目符号（见图11-26），文件名为elevation\_marker.png。



图11-26 标志图形

现在，设置列表样式，比如指定font-family和font-size、删除或定义外边距和内边距等。重要的是，为div#route\_elevation ul li a元素声明背景图像elevation\_marker.png，应用15像素的margin-left在项目符号和海拔高度之间留出空间。

```

div#route_elevation ul {
    list-style:none;
    margin:0;
    font-family:Verdana,sans-serif;
    font-size:9px;
    font-weight:bold;
}
div#route_elevation ul li {

```

```

margin:0;
}
div#route_elevation ul li a {
  color:#333;
  display:block;
  background:url(..../images/site/elevation_marker.png) no-repeat 0 5px;
  padding:0 0 0 15px;
}
div#route_elevation ul li a:hover,
div#route_elevation ul li a:focus {
  color:#000;
}
div#route_elevation ul li a img {
  display:none;
}

```

我们使用div#route\_elevation ul li a img选择Flickr缩略图，使用display:none禁止显示它们。

海拔图现在显示的列表样式合适多了，但是列表项仍然在容器的左边排成一行（见图11-27）。

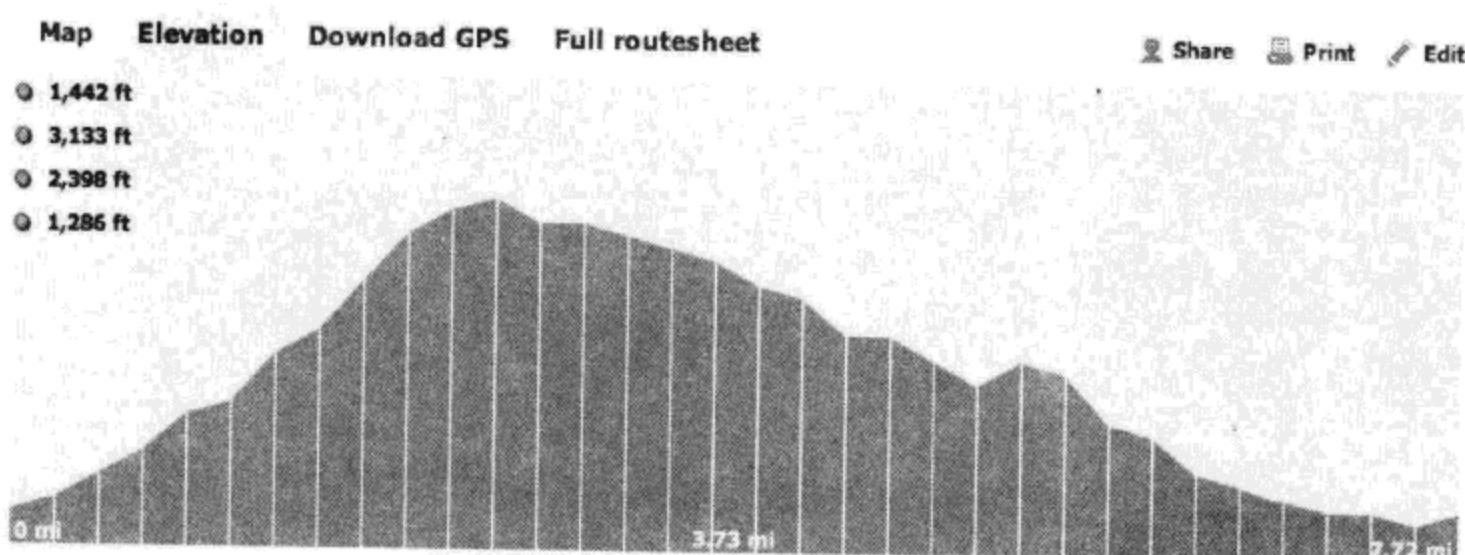


图11-27 列表项应用了样式，但是仍然排在左边

要想确保列表项按指定的坐标显示，只需使用position:absolute对它们进行绝对定位，如下所示：

```

div#route_elevation ul li {
  margin:0;
  position:absolute;
}

```

这么做是有效的，因为父容器是相对于它自己的父容器定位的。如果不是这种情况，绝对定位的列表项会根据到浏览器窗口顶边和左边的坐标定位，这根本不是我们要的结果。如图11-28所示，海拔点现在在图上准确定位了。

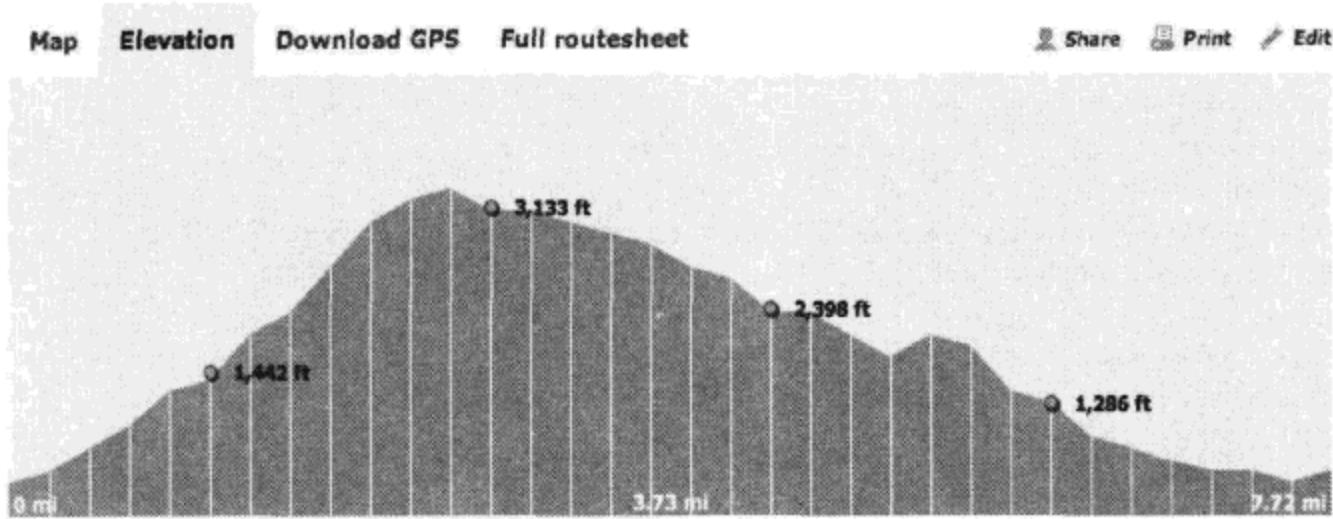


图11-28 通过绝对定位，标志显示在正确的坐标上

现在需要处理前面隐藏的图像，当鼠标悬停在海拔点上时显示相应的图像。

应用样式的目标是`:hover`和`:focus`伪类。通过控制它们，可以轻松地应用显示图像的声明。

注意，在设置高度和宽度之后，同样对元素进行绝对定位，使用负的`top`和`right`值让图像出现在相对于列表标志的指定位置上。

```
div#route_elevation ul li a:hover img,
div#route_elevation ul li a:focus img {
    display:block;
    width:40px;
    height:40px;
    padding:4px 9px 10px 12px;
    position:absolute;
    top:-16px;
    right:-65px;
}
```

因此，在鼠标悬停时，在标志的右边会出现缩略图。我们还在缩略图周围加了一些内边距（见图11-29）。

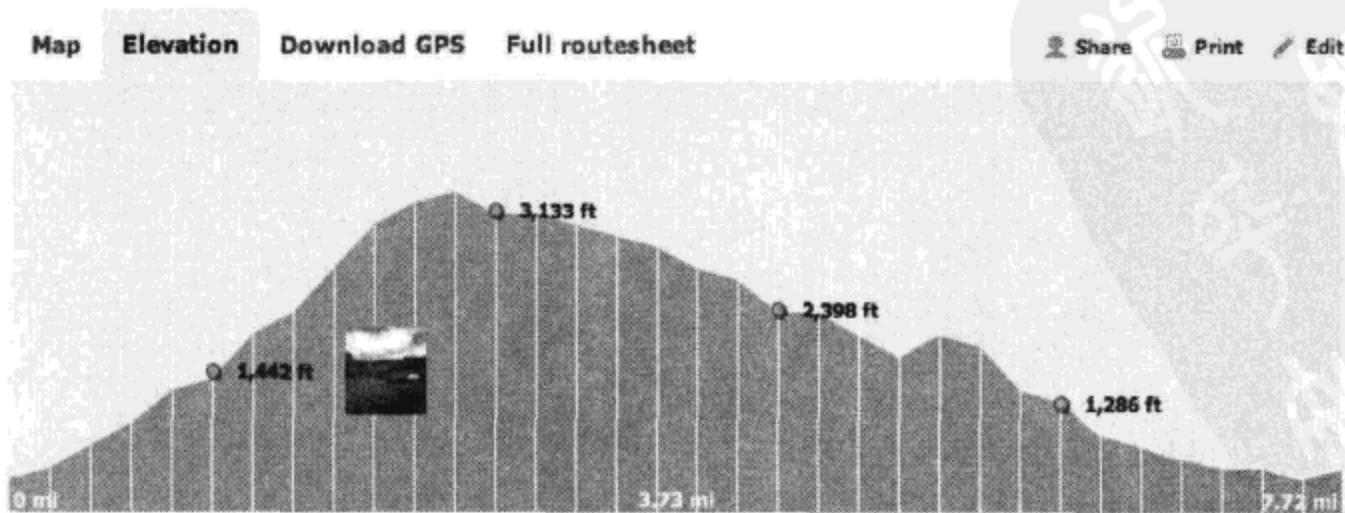


图11-29 在鼠标悬停时出现基本的缩略图

现在，利用内边距提供的空间，在缩略图背后放一个框架和箭头图像。这里使用elevation\_marker\_image\_bg.png（见图11-30）。注意，阴影是在Photoshop中添加的，然后作为透明的PNG导出。



图11-30 缩略图背后的简单图像

然后，只需在CSS中应用这个背景图像，如下所示：

```
div#route_elevation ul li a:hover img,
div#route_elevation ul li a:focus img {
    display:block;
    width:40px;
    height:40px;
    padding:4px 9px 10px 12px;
    position:absolute;
    top:-16px;
    right:-65px;
    background:url(../images/site/elevation_marker_image_bg.png) no-repeat 0 0;
}
```

如果希望实现相似的效果，可能需要精确调整内边距和定位等，但是基本代码就是这些。经过本节中的这些工作，页面现在的Your latest route区域非常完善，交互式的海拔图会根据鼠标位置显示Flickr图像，见图11-31。

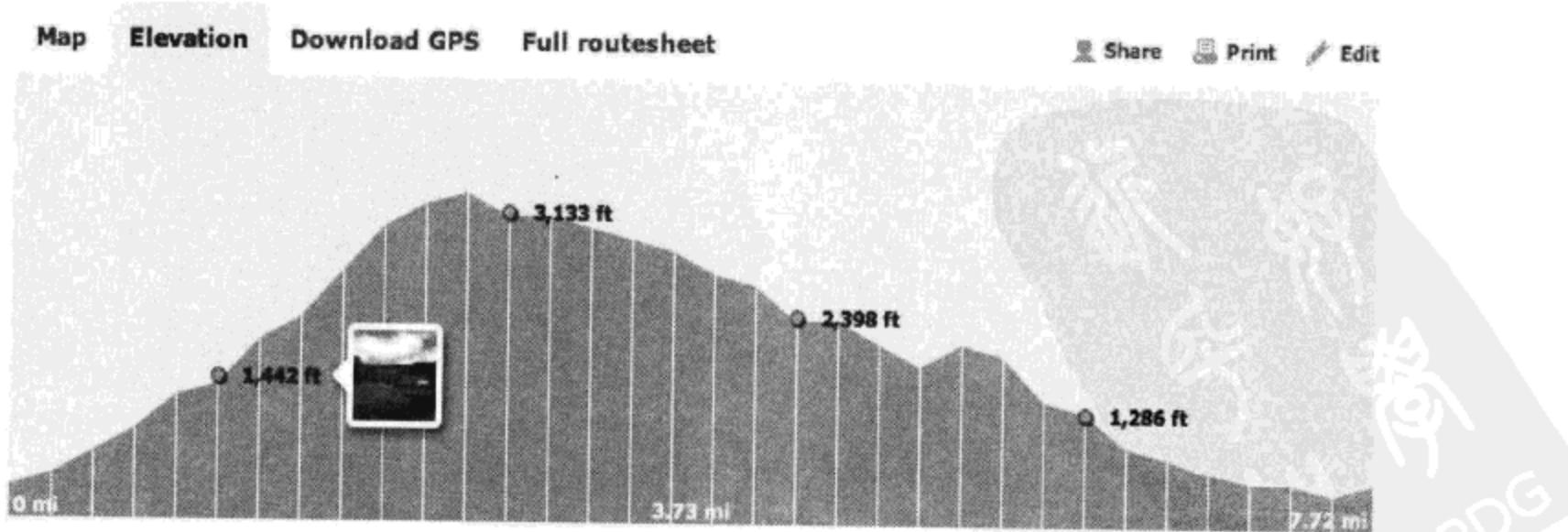


图11-31 完成的“Your latest route”区域提供简洁的图像翻转器

当然，要想实现真正惊人的效果，最终需要与强大的CMS连接，但是本书只讨论CSS。

## 11.8 小结

就到这里了。虽然这个实例研究非常简短，但是我希望它对你有所启发。我确实很高兴在本书第2版中介绍Climb the Mountains的概念，尤其是可以自由地讨论一些新颖的CSS 3思想。

当然，在Climb the Mountains网站中还有许多我想介绍的东西。但是，由于篇幅的限制，无法更深入地讨论了。如果你喜欢网站中的某种效果或处理方式，而本章没有涉及它，我认为通过研究源代码应该不难弄清内部原理。我已经尽量确保源代码的结构清晰，还添加了许多有帮助的说明和参考资料。

请记住，你任何时候都可以访问<http://www.climbthemountains.com/cssm>，还可以从[www.friendsofed.com](http://www.friendsofed.com)获得源代码。欢迎下载和研究它，希望它可以激发你的创造力，帮助你用CSS实现精彩的效果。



[ General Information ]

书名=精通CSS 高级Web标准解决方案

作者=(英)Simon Collison,Cameron Moll著

页码=266

ISBN=266

SS号=12643938

d x Number=000006876407

出版时间=2010.05

出版社=该引擎未能查询到

定价:49.00

试读地址=http://book.szdn.net.org.cn/bookDetail.jsp?

d x Number=000006876407&d=F473ABE72B209B6FF52  
0025FA46EA5EA4&fenlei=181704&sw=css

全文地址=http://img6.5read.com/image/ss2jpg.dll?  
?did=b56&pid=90F4AF51A3F9891A343B1B9C96C4F6  
0AD537907BC8F69434914DA956CED00FB1413A2164F  
63152755F9A79D857CE0DA02F0F708ADAC9ABEECE8A  
84254DE39354D0F68F0D74B1DFFFBB26C773F7C8E28C  
C37D9CA16DAD6F7998EF2F9F6F3D73132BCF35F56FB  
4285688F64C4E50EB97B05FD6F&jid=/