



国内首本CSS 3专著，全面而深入讲解CSS 3的最新特性和布局之道
实战性强，全书囊括近百个精心设计的实战案例，理论与实践完美结合
资深Web前端工程师多年实践经验的结晶，3大社区联袂推荐



全彩印刷



成林 著

CSS 3 in Action

CSS 3 实战



机械工业出版社
China Machine Press

PDF
PDG

實戰

-29



CSS 3 in Action

CSS 3 實戰

成林 著



机械工业出版社
China Machine Press

本书由国内资深 Web 前端工程师撰写，权威性毋庸置疑。如果你是一位有前瞻性的 Web 前端工作者，那么本书也许会让你在即将到来的 Web 技术革命中领先一步。

本书技术新颖，基于 CSS 3 的最新版本撰写，所有新功能和新特性尽含其中；内容全面，不仅讲解了 CSS 3 的方方面面，而且还在一些关键的功能点上与 CSS 2.x 进行了充分的比较；实战性强，几乎所有知识点都配有案例，全书配有实战案例百余个。本书不仅能满足读者系统学习理论知识的需求，还能满足需要充分实践的需求。

全书一共分为 9 章，首先从宏观上介绍了 CSS 3 技术的最新发展现状、新特性，以及现有的主流浏览器对这些新特性的支持情况；然后详细讲解了 CSS 3 的选择器、文本特性、颜色特性、弹性布局、边框和背景特性、盒模型、UI 设计、多列布局、圆角和阴影、渐变、变形、转换、动画、投影、开放字体、设备类型、语音样式等重要的理论知识，这部分内容是本书的基础和核心。不仅每个知识点都配有丰富的、精心设计的实战案例，而且详细介绍了每一种新特性在各种主流浏览器上的兼容性，旨在帮助设计师们提高设计的安全性。

本书全彩印刷，排版、设计和装帧也非常精美，既适合学习参考，也适合收藏。无论你是前端领域的新人，还是有着丰富经验的老手，都能通过本书系统而全面地学习和实践 CSS 3 的最新技术，为迎接新一轮的 Web 技术革命打下坚实的基础。

封底无防伪标均为盗版

版权所有，侵权必究

本书法律顾问 北京市展达律师事务所

图书在版编目 (CIP) 数据

CSS 3 实战 / 成林著 .—北京 : 机械工业出版社, 2011.5

ISBN 978-7-111-34155-0

I. C… II. 成… III. 网页制作工具, CSS 3 IV. TP393.092

中国版本图书馆 CIP 数据核字 (2011) 第 062188 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑：陈佳媛

中国电影出版社印刷厂印刷

2011 年 5 月第 1 版第 1 次印刷

186mm×240mm · 20.5 印张

标准书号：ISBN 978-7-111-34155-0

定价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 ; 88361066

购书热线：(010) 68326294 ; 88379649 ; 68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com





目 录

前言

第 1 章 预览激动人心的CSS 3 1

- 1.1 CSS 3发展概述 1
- 1.2 CSS 3模块化简介 2
- 1.3 CSS 3新特性概览 4
- 1.4 主流浏览器对CSS 3的支持 8
- 1.5 CSS 3的未来和思考 9

第 2 章 CSS 3新增的选择器 11

- 2.1 属性选择器 13
 - 实战体验：文档共享的友善之举 14
- 2.2 结构伪类选择器 17
 - 实战体验1：设计优雅的数据表格 19
 - 实战体验2：CSS大战保龄球 21
 - 实战体验3：让枯燥的列表更有趣 27
 - 实战体验4：清理圆角边框中的垃圾标签 29
- 2.3 UI元素状态伪类选择器 33
 - 实战体验1：设计可用的表单 36
 - 实战体验2：设计友好、易用的表单 40
- 2.4 其他新增选择器 45
 - 实战体验1：设计层序化的数据表格 47
 - 实战体验2：改善页内导航的视觉体验 50



4.7.1 构建博客页的基本结构 117

4.7.2 完善博客页的结构 118

4.7.3 弹性布局设计 119

第5章 完善的盒模型和UI设计 124

5.1 定义多色边框——border-color属性 124

 实战体验：设计立体边框 127

5.2 定义边框背景图——border-image属性 128

 实战体验：设计各种精巧的边框 138

5.3 设计圆角——border-radius属性 140

5.4 设计块阴影——box-shadow属性 147

5.5 CSS 3边框和背景样式综合实战 151

5.6 设计多重背景图象——background属性 155

 实战体验：背景图像合成 157

5.7 定义背景坐标原点——background-origin属性 158

 实战体验：设计信纸背景效果 159

5.8 定义背景裁剪区域——background-clip属性 160

 实战体验1：设计内容区背景 163

 实战体验2：设计按钮效果 163

5.9 定义背景图像大小——background-size属性 164

 实战体验：设计自适应模块大小的背景图像 165

5.10 溢出内容处理——overflow-x和overflow-y属性 166

5.11 定义盒模型解析模式——box-sizing属性 170

5.12 自由缩放——resize属性 171

 实战体验：设计能随意调整大小的壁画 172

5.13 定义外轮廓线——outline属性 173

 实战体验：设计醒目激活和焦点提示框 175

5.14 定义外轮廓线宽度——outline-width属性 177

5.15 定义外轮廓线样式——outline-style属性 178

5.16 定义外轮廓线颜色——outline-color属性 179

5.17 定义外轮廓线位移——outline-offset属性 179

 实战体验：放大激活和焦点提示框 180



- 5.18 定义导航序列号——nav-index属性 181
 - 实战体验：调整表单输入框的键盘激活顺序 182
- 5.19 定义方向键控制顺序 184
 - 实战体验：正确定义键盘控制键顺序 185
- 5.20 为元素添加内容——content属性 187

第6章 CSS 3多列布局 190

- 6.1 定义多列布局——columns属性 190
 - 实战体验：设计文章多栏显示 191
- 6.2 定义列宽度——column-width属性 192
 - 实战体验：设计固定宽度的栏目版面 193
- 6.3 定义列数——column-count属性 194
 - 实战体验：设计固定列数的版面 195
- 6.4 定义列间距——column-gap属性 196
 - 实战体验：设计疏朗的文档版面 197
- 6.5 定义列边框样式——column-rule属性 198
 - 实战体验：为多列布局版面设计边框效果 200
- 6.6 定义跨列显示——column-span属性 201
 - 实战体验：设计文章标题跨列显示 202
- 6.7 定义栏目高度——column-fill属性 203
 - 实战体验：设计不等高的多列布局效果 204
- 6.8 分列打印 206
- 6.9 结合案例实战——设计精美的多列网页版式 207

第7章 CSS 3渐变设计 217

- 7.1 Webkit引擎的CSS渐变实现方法 217
 - 7.1.1 基本语法 218
 - 7.1.2 直线渐变的基本用法 218
 - 7.1.3 径向渐变的基本用法 220
 - 7.1.4 渐变的其他应用 225
- 7.2 Gecko引擎的CSS渐变实现方法 227
 - 7.2.1 直线渐变基本语法 227
 - 7.2.2 直线渐变的基本用法 227



7.2.3	径向渐变基本语法	230
7.2.4	径向渐变的基本用法	231
7.2.5	渐变的应用	234
7.3	IE浏览器引擎的CSS渐变实现方法	235
7.3.1	基本语法	235
7.3.2	IE渐变滤镜实战应用	236
7.4	W3C标准化的CSS渐变实现方法	238
7.5	CSS 3渐变实战	239
7.5.1	CSS渐变下拉菜单	239
7.5.2	CSS渐变按钮	243

第8章 CSS 3动画设计 248

8.1	CSS变形（Transformation）	248
8.1.1	变形基础——transform属性	249
实战体验：设计一个简单的鼠标动画		250
8.1.2	旋转变形——rotate()函数	251
8.1.3	缩放动画——scale()函数	253
8.1.4	移动动画——translate()函数	255
8.1.5	倾斜动画——skew()函数	258
8.1.6	矩阵变形动画——matrix()函数	260
8.1.7	CSS 3实战体验：设计图片墙	262
8.2	CSS变形原点——transform-origin属性	265
CSS3实战体验：定义图片旋转的原点		265
8.3	CSS过渡——transition属性	268
8.3.1	设置过渡的CSS属性——transition-property属性	269
8.3.2	设置过渡的时间——transition-duration属性	270
8.3.3	设置过渡延迟时间——transition-delay属性	271
8.3.4	设置过渡效果——transition-timing-function属性	272
CSS 3实战体验：设计OS X Dock（OS系统的导航码头）		274
8.4	CSS动画——animation属性	276
8.4.1	设置CSS动画名称——animation-name属性	277
8.4.2	设置CSS动画时间——animation-duration属性	278

- 8.4.3 设置CSS动画播放方式——animation-timing-function属性 278
- 8.4.4 设置CSS动画开始播放的时间——animation-delay属性 279
- 8.4.5 设置CSS动画播放次数——animation-iteration-count属性 279
- 8.4.6 设置CSS动画播放方向——animation-direction属性 280

CSS 3实战体验：设计自动翻转的图片效果 280

8.5 CSS 3动画综合实战 282

- 8.5.1 设计动态立体盒子 282
- 8.5.2 设计CSS 3手风琴式折叠面板 285
- 8.5.3 设计能够旋转背景的易拉罐 287
- 8.5.4 设计旋转出仓的光盘动画效果 290

第9章 CSS 3新增的其他功能 295

9.1 引用外部字体类型——@font-face规则 296

- 9.1.1 @font-face规则的用法 296
- 实战体验：设计艺术字体 297

- 9.1.2 关于开放字体格式 298

9.2 定义CSS设备类型——Media Queries 299

- 9.2.1 @media规则的用法 300
- 实战体验：为不同设备设计不同的盒子框样式 302

- 9.2.2 使用Media Queries链接外部CSS文件 304

- 9.2.3 测试Media Queries 305

9.3 定义投影——CSS Reflections 305

CSS实战体验：应用CSS Reflections 306

9.4 定义语音样式——CSS 3 Speech 310

实战体验：体验CSS 3 Speech应用 311

为什么写这本书

CSS 3 真可谓十年磨一剑，从 10 年前开始孕育，到今天逐渐引人瞩目，前端工作者们的确等待了太长的时间。

随着用户要求的不断提高、各种新型网络应用的不断出现，以及 Web 技术自身的高速发展，CSS 2 在 Web 开发中显得越来越力不从心，人们对下一代 CSS 技术和标准——CSS 3 的需求越来越迫切。坦率地讲，CSS 3 的部分特性在几年前就已经公布，但是由于各种主流浏览器的“不作为”，特别是 IE 浏览器的“消极态度”，让很多前端工作者遗忘了 CSS 3 的存在。目前，CSS 3 还在不断完善中，很多功能还处于草稿阶段，但是它展现出来的超强特性和功能已经让人兴奋不已。最近一两年，各种主流浏览器逐渐开始高调支持 CSS 3 的部分或者全部的功能特性，使得 CSS 3 又重新进入了广大前端工作者的视野。特别是 IE 9 对 CSS 3 的全面支持，更是将网页设计师带入了全新的天地。社区里各种关于 CSS 3 的讨论、资料和炫酷的应用开始爆炸式增长，广大前端工作者也开始蜂拥而至。

对于紧追前沿技术的前端工作者来说，充分了解当前和未来的 Web 标准和技术是十分必要的，学习和掌握 CSS 3 更是大势所趋。为了帮助大家在适应趋势和引领趋势的过程中能走得更顺利，受华章公司的盛情邀请，我特意编写了这本 CSS 3 实战教程，希望能起到抛砖引玉之效，为普及 CSS 3 尽绵薄之力。

本书面向的读者

首先，本书非常适合具有丰富开发和设计经验的前端工作者，因为这部分读者应该已经对 CSS 2 了然于胸，通过本书，他们将能非常迅速而又有针对性地掌握 CSS 3 技术。

其次，本书也适合尚处于初级阶段的前端工作者，因为书中不仅系统而全面地介绍了 CSS 3 的各种功能和特性，而且还有大量实战案例和最佳实践，可供他们一边学习理论，一边进行实战演练。

本书内容特色

本书是国内第一本系统、全面地讲解 CSS 3 的图书，它有两个重要的特色：

- 内容全面而详尽。本书几乎讲解了 CSS 3 已经公布的所有可用新特性和新功能的用法、技巧和注意事项。
- 案例丰富，实战性强。本书几乎为每个知识点都精心设计了 1~2 个实战案例，能帮助读者在实战演练的过程中将理论知识融会贯通。

本书约定

在本书的阅读过程中，需要注意下面几个约定：

- 初始值：即默认值，是当用户不显式声明时元素所显示的属性值。需指明的是，属性是元素的本质，而不是后天定义的标签。
- 适用于：说明了该属性适用哪些元素，有些参考资料中所提供的适用元素列表很容易使读者陷于迷茫之中，指导意义不大，妨碍快速参考。
- 继承性：这是 CSS 的基本特性，表明该属性值是否会对当前引用元素的内嵌子元素具有影响力。继承性对 CSS 布局而言具有重要的参考价值。
- 百分比：表示该属性是否可以用百分比（%）或者 em 为单位，以及如果可以用百分比或者 em 为单位时，如何才能把这些值换算成确定的值。例如，百分比是根据自身的宽度进行换算还是根据父元素的宽度进行换算，再或者是根据元素内文本字体的大小进行换算等。默认值为 N/A，表示百分比不符合或者不可用。
- 媒介：说明该属性适用于哪些设备，例如，visual 表示视觉媒体，如电脑屏幕、WAP（如手机）屏幕、打印机等。
- 在没有特别声明的情况下，本书所指的浏览器仅适用于 Windows 系统，不适用于 Mac 系统。
- Webkit 引擎主要指苹果的 Safari 浏览器和谷歌的 Chrome 浏览器，其私有属性前缀为 -webkit-。
- Gecko 引擎主要指代 Mozilla 的浏览器，常指 Firefox，其私有属性前缀为 -moz-。
- Presto 引擎主要指代 Opera 浏览器，其私有属性前缀为 -o-。
- 本书所有案例在 Chrome 4.0+ 或者 Safari 4.0+ 版本的浏览器中能够获得较好的表现。在 IE 8 及其以下的版本中，可能得不到预期的效果。

为了方便阅读，本书中的部分示例代码仅提供了 CSS 样式代码和局部 HTML 结构代码，读者可以把这些 CSS 样式代码放在网页头部区域（即 <head> 标签内），局部 HTML 结构代码放在网页主体区域内（即 <body> 标签内）。

本书不是最终的 CSS 3 技术大全，CSS 3 技术还在不断完善和补充中，所以也无法确保本书中讲解的所有知识将来都不会发生变化。建议读者根据本书所提供的参考地址，即时获取关于 CSS 3 的最新信息。

CSS 3 技术学习延伸

学习 CSS 3 实际上并不难，难的是完全了解浏览器的兼容性问题。在还没有完全普及 CSS 3 标准之前，我们只能够根据各主流浏览器引擎所实现的 CSS 3 私有特性来实现兼容。这势必会导致将简单的问题复杂化，学习的成本和应用的难度也会相应增加。这一方面是因为很多 CSS 3 属性使用比较繁琐，如转换、过渡、渐变等，另一方面是因为设计时还要考虑各浏览器厂商的扩展（兼容方法）。对于广大前端工作者来说，下面这些参考资料也许会非常有用。

- **CSS 3 Selectors Test[⊖]**：这是 CSS 3.info 网站提供的 CSS 选择器测试页面，它能够详细显示当前浏览器对所有 CSS 3 选择器的支持情况。启动测试，浏览器会自动测验，并以列表的方式显示当前浏览器对所有 CSS 3 选择器的支持情况，点击每个 CSS 3 选择器可以查看结果和解释信息。
- **When can I use[⊖]**：这是一个专业的测试网站，为广大网页设计师提供 CSS 3、HTML5、SVG、JavaScript API 技术的浏览器支持情况检测，它能够准确显示什么时候能用 CSS 3、HTML5、SVG，以及其他即将可用的页面技术的浏览器兼容性列表。
- **What's my IP[⊕]**：这也是一个专业的小网站，可用于检测当前浏览器对 CSS 3、HTML5、Forms 2.0、CSS 3 选择器和 Script 等技术的支持情况。
- **MooTools HTML5/CSS 3 feature detection[⊕]**：Modernizr 是一个很有用的 JavaScript 库，可以检测 HTML 5 和 CSS 3 的原生支持，并提供一种维护良好控制级别的方法。如果你喜欢使用 MooTools，可以使用 MooModernizr（MooTools 版本的 Modernizr）。
- **CSS 3 Generator[⊕]**：这是一个 CSS 生成器，可以快速地以可视化的方式生成 CSS 3 新特性的样式，不过该工具仅支持 border radius、box shadow、text shadow、RGBA、@font-face、多列、box resize、box sizing 和 outline 特性，其他特性暂不支持。

⊖ <http://tools.CSS 3.info/selectors-test/test.html>

⊕ <http://caniuse.com/>

⊕ <http://fmbip.com/>

⊕ <http://www.aryweb.nl/projects/mooModernizr/>

⊕ <http://css3generator.com/>

- CSS 3 please!^①：这是跨浏览器的 CSS 规则生成器，支持 border-radius、box-shadow、渐变（线性）、rgba 色彩、transform（旋转）、transition 和 @font-face。
- CSS 3 Sandbox^②：提供了几个 CSS 3 生成器，包括线性渐变、放射渐变、文字阴影、盒阴影、Transforms 和文字描边。
- CSS 3 渐变生成器^③：为 Firefox 和 Webkit 浏览器生成线性渐变。
- @font-face 生成器^④：来自于 Font Squirrel 的很好用的 CSS 3 @font-face 生成器。
- CSS 圆角生成器^⑤：生成用于 Firefox、Webkit 和标准 CSS 3 语法的 border-radius 属性。
- CSS 3.0 参考手册^⑥：这是由腾讯 ISD WebTeam 制作的一个 CHM 文档，由于时间较早，稍显陈旧，错误和遗漏比较多，不过对想了解 CSS 3 基本特性的初学者来说，还是有一定的参考价值。

致谢

本书主要由成林编写，同时参与资料整理及编写的还有：马本连、吴建华、江淑军、李斌、李经键、郑伟、田蜜、陆颖、王慧明、张炜、陈锐、王幼平、杨龙贵、苏震巍、崔鹏飞等，在此对大家的辛勤工作表示衷心的感谢！

由于时间有限，书中难免会有疏漏和不足之处，恳请广大读者提出宝贵意见。有关本书的任何问题，请发电子邮件到 css3shizhan@163.com。

作者

2011 年 3 月于北京

- ① <http://css3please.com/>
- ② <http://westciv.com/tools/index.html>
- ③ <http://gradients.glrzad.com/>
- ④ <http://www.fontsquirrel.com/fontface/generator>
- ⑤ <http://border-radius.com/>
- ⑥ <http://isd.tencent.com/css3/>

预览激动人心的 CSS 3

如果你关注 CSS，那么一定听说过 CSS 3，这个早在几年前就问世的下一代样式表语言，至今还没有完成所有规范化草案的制订。如果你已经迫不及待，想一试身手，那么也不必担心浏览器支持问题，虽然最终的、完整的、规范权威的 CSS 3 标准还没有尘埃落定，但是各主流浏览器已经开始支持其中的绝大部分特性。

在 CSS 3 的支持上，谷歌的 Chrome 和苹果的 Safari 走在最前列，Opera 和 Firefox 奋力紧追，IE 也放下偏执，从 IE 9 开始发力，逐步投身到 CSS 3 的怀抱之中。当然，要全面普及 CSS 3 技术，并完全获得各主流浏览器的支持，还需要走很长的路，至少在可预见的一两年内，你可能还会听到各种激烈的讨论，但这并不意味着你现在不必去了解它。如果想成为前卫的高级网页设计师，那么就应该从现在开始积极去学习和实践。

1.1 CSS 3 发展概述

20 世纪 90 年代初，HTML 语言诞生，各种形式的样式表也开始出现。各种不同的浏览器结合自身的显示特性，开发了不同的样式语言，以便于读者自己调整网页的显示效果。注意，此时的样式语言仅供读者使用，而非供设计师使用。

早期的 HTML 语言只含有很少量的显示属性，用来设置网页和字体的效果。随着 HTML 的发展，为了满足网页设计师的要求，HTML 不断添加了很多用于显示的标签和属性。由于 HTML 的显示属性和标签比较丰富，其他的用来定义样式的语言就越来越没有意义了。

在这种背景下，1994 年初哈坤·利提出了 CSS 的最初想法。伯特·波斯（Bert Bos）当时正在设计一款 Argo 浏览器，于是他们一拍即合，决定共同开发 CSS。当然，这时市面上已经有一些非正式的样式表语言的提议了。

1994 年年底，哈坤在芝加哥的一次会议上第一次展示了他们对 CSS 的构想，1995 年他

与波斯再一次展示了他们的想法。当时 W3C 组织刚刚成立，W3C 对 CSS 的前途很感兴趣，为此组织了一次讨论会。哈坤、波斯和其他一些人（如微软的托马斯·雷尔登）是这个项目的主要技术负责人。

1996 年年底，CSS 语言正式完成，同年 12 月 CSS 的第一版被正式推出 (<http://www.w3.org/TR/CSS1/>)。

1997 年年初，W3C 内组织了专门负责 CSS 的工作组，负责人是克里斯·里雷。于是该工作组开始讨论第一个版本中没有涉及的问题。

1998 年 5 月，CSS 2 正式发布 (<http://www.w3.org/TR/CSS2/>)。

尽管 CSS 3 的开发工作在 2000 年之前就开始了，但是距离最终的发布还有相当长的路要走。为了提高开发速度，也为了方便各主流浏览器根据需要渐进式地支持它，CSS 3 被分割成多个模块，这些模块可以独立实现和发布，这也为日后 CSS 的扩展奠定了基础。

考虑到 CSS 3 的定案还需要很长的时间，2002 年工作组启动了对 CSS 2.1 的开发。这是 CSS 2 的修订版，它纠正了 CSS 2 中的一些错误，并且更精确地描述了 CSS 的浏览器实现。2004 年 CSS 2.1 正式发布，到 2006 年年底得到完善，CSS 2.1 是目前最流行、浏览器支持最完整的版本，它更准确地反映了 CSS 当前的状态。

1.2 CSS 3 模块化简介

CSS 1 主要定义了网页的基本属性，如字体、颜色、空白边等。CSS 2 在此基础上添加了一些高级功能，如浮动和定位；以及一些高级的选择器，如子选择器、相邻选择器和通用选择器等。

CSS 3 开始遵循模块化开发，这将有助于理清模块化规范之间的不同关系，减少完整文件的大小。以前的规范是一个完整的模块，实在是太庞大，而且比较复杂，所以，新的 CSS 3 规范将其分为了多个模块。

CSS 模块化能够帮助我们，根据需要决定哪些 CSS 功能被支持。此外，该规范的模块化特性使得每个独立的模块能根据需要进行更新，从而便于整体规范的及时修订，这样更容易开发出新的技术特性。

2001 年 5 月 23 日，W3C 完成了 CSS 3 的工作草案，在该草案中制订了 CSS 3 的发展路线图，详细列出了所有模块，并计划在未来逐步进行规范。细节信息请参阅：<http://www.w3.org/TR/css3-roadmap/>。下面将简单介绍各个模块的用途、发布时间，以及参考地址。

- 2002 年 5 月 15 日发布了 CSS 3 line 模块 (<http://www.w3.org/TR/css3-linebox/>)，该模块规范了文本行模型。

- 2002年11月7日发布了CSS 3 Lists模块(<http://www.w3.org/TR/css3-lists/>)，该模块规范了列表样式。
- 2002年11月7日发布了CSS 3 Border模块(<http://www.w3.org/TR/2002/WD-css3-border-20021107/>)，新添加了背景边框功能，该模块后来被合并到背景模块中(<http://www.w3.org/TR/css3-background/>)。
- 2003年5月14日发布了CSS 3 Generated and Replaced Content模块(<http://www.w3.org/TR/css3-content/>)，该模块定义了CSS 3的生成及更换内容功能。
- 2003年8月13日发布了CSS 3 Presentation Levels模块(<http://www.w3.org/TR/css3-preslev/>)，该模块定义了演示效果功能。
- 2003年8月13日发布了CSS 3 Syntax模块(<http://www.w3.org/TR/css3-syntax/>)，该模块重新定义了CSS语法规则。
- 2004年2月24日发布了CSS 3 Hyperlink Presentation模块(<http://www.w3.org/TR/css3-hyperlinks/>)，该模块重新定义了超链接表示规则。
- 2004年12月16日发布了CSS 3 Speech模块(<http://www.w3.org/TR/css3-speech/>)，该模块重新定义了语音“样式”规则。
- 2005年12月15日发布了CSS 3 Cascading and inheritance模块(<http://www.w3.org/TR/css3-cascade/>)，该模块重新定义了CSS层叠和继承规则。
- 2007年8月9日发布了CSS 3 basic box模块(<http://www.w3.org/TR/css3-box/>)，该模块重新定义了CSS的基本盒模型规则。
- 2007年9月5日发布了CSS 3 Grid Positioning模块(<http://www.w3.org/TR/css3-grid/>)，该模块定义了CSS的网格定位规则。
- 2009年3月20日发布了CSS 3 Animations模块(<http://www.w3.org/TR/css3-animations/>)，该模块定义了CSS的动画模型。
- 2009年3月20日发布了CSS 3 3D Transforms模块(<http://www.w3.org/TR/css3-3d-transforms/>)，该模块定义了CSS 3D转换模型。
- 2009年3月20日发布了CSS 3 3D Transforms模块(<http://www.w3.org/TR/css3-3d-transforms/>)，该模块定义了CSS 3D转换模型。
- 2009年6月18日发布了CSS 3 Fonts模块(<http://www.w3.org/TR/css3-fonts/>)，该模块定义了CSS字体模型。
- 2009年7月23日发布了CSS 3 Image Values模块(<http://www.w3.org/TR/css3-images/>)，该模块定义了图像内容显示模型。
- 2009年7月23日发布了CSS 3 Flexible Box Layout模块(<http://www.w3.org/TR/css3-flexbox/>)，该模块定义了灵活的框布局模块。

- 2009 年 8 月 4 日发布了 CSS 3 Flexible Box Layout 模块 (<http://www.w3.org/TR/cssom-view/>)，该模块定义了 CSS 的视图模块。
- 2009 年 12 月 1 日发布了 CSS 3 Transitions 模块 (<http://www.w3.org/TR/css3-transitions/>)，该模块定义了动画过渡效果模型。
- 2009 年 12 月 1 日发布了 CSS 3 2D Transforms 模块 (<http://www.w3.org/TR/css3-2d-transforms/>)，该模块定义了 2D 转换模型。
- 2010 年 4 月 29 日发布了 CSS 3 Template Layout 模块 (<http://www.w3.org/TR/css3-layout/>)，该模块定义了模板布局模型。
- 2010 年 4 月 29 日发布了 CSS 3 Generated Content for Paged Media 模块 (<http://www.w3.org/TR/css3-gcpm/>)，该模块定义了分页媒体内容模型。
- 2010 年 10 月 5 日发布了 CSS 3 Text 模块 (<http://www.w3.org/TR/css3-text/>)，该模块定义了文本模型。
- 2010 年 10 月 5 日发布了 CSS 3 Backgrounds and Borders 模块 (<http://www.w3.org/TR/css3-background/>)，该模块重新修订了边框和背景模型。

1.3 CSS 3 新特性概览

与前面几个版本相比较，CSS 3 的变化是革命性的，而不是仅限于局部功能的修订和完善。尽管 CSS 3 的一些特性还不能被很多浏览器支持，或者说支持得还不够好，但是它依然让我们看到了网页样式的发展方向和使命。

简单地说，CSS 3 使得很多以前需要使用图片和脚本才能实现的效果，如今只需要几行代码就能实现。这不仅简化了设计师的工作，而且还能加快页面载入速度。下面我们就来领略一下 CSS 3 的主要新特性。

1. 强大的选择器

CSS 3 的选择器在 CSS 2.1 的基础上进行了增强，它允许设计师在标签中指定特定的 HTML 元素而不必使用多余的类、ID 或者 JavaScript 脚本。

如果希望设计出简洁、轻量级的网页标签，希望结构与表现更好地分离，高级选择器是非常有用的。它可以避免在标签中添加大量的 class 和 id 属性，并让设计师更方便地维护样式表。

2. 半透明度效果的实现

RGBA 和 HSLA 不仅可以设定色彩，还能设定元素的透明度。另外，还可以使用 opacity

属性定义元素的不透明度，类似效果如图 1.1 所示。

当鼠标经过时，产生淡淡的半透明色效果，使按钮效果看起来更加精致 (<http://timvandamme.com/>)



图 1.1 半透明按钮效果

3. 多栏布局

CSS 3 让网页设计师不必使用多个 div 标签就能实现多栏布局。浏览器能解释多栏布局属性并生成多栏，让文本实现纸质报纸的多栏结构，效果如图 1.2 所示。

tweetCC在其首页使用了CSS 3 多栏布局 (<http://tweetcc.com/>)

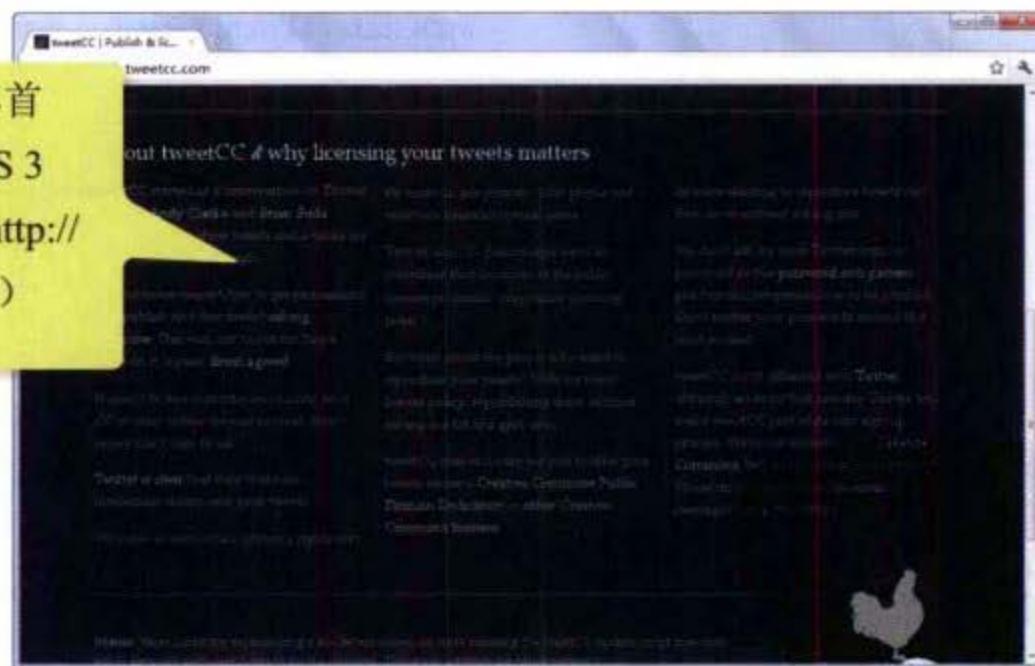


图 1.2 多栏布局

4. 多背景图

CSS 3 允许背景属性设置多个属性值，如 background-image、background-repeat、background-size、background-position、background-originand、background-clip 等，这样就可以在一个元素上添加多层背景图片。如果要设计复杂的网页效果（如圆角、背景重叠等），就不用再为

HTML 文档添加多个无用的标签了，使用该属性还可以优化网页文档的结构。

5. 文字阴影

`text-shadow` 在 CSS 2 中就已经存在，但并没有被广泛应用。CSS 3 采用了该特性，并重新进行了定义。该属性提供了一种新的跨浏览器的方案使文字看起来更醒目。

6. 开放字体类型

`@font-face` 是最被期待的 CSS 3 特性之一，它在 CSS 2 中就已经被引入了，但是它在网站上仍然没有像其他 CSS 3 属性那样被广泛普及，这主要受阻于字体授权和版权问题，嵌入的字体很容易从网站上下载到，这是字体厂商的主要顾虑。开放字体应用效果如图 1.3 所示。

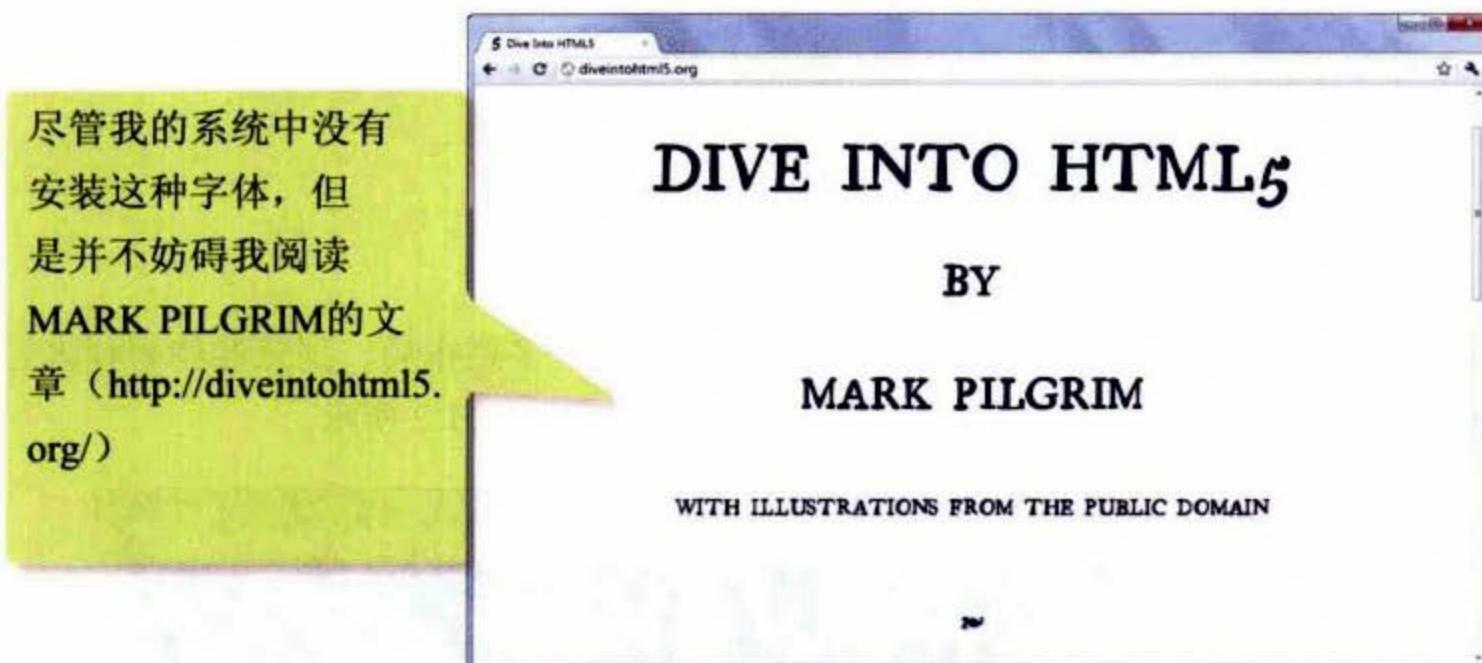


图 1.3 开放字体类型应用

7. 圆角

`border-radius` 属性不需背景图片就能给 HTML 元素添加圆角。它可能是现在使用得最多的 CSS 3 属性，很简单的原因是，使用圆角比较美观，而且不会与设计和可用性产生冲突。它不同于添加 JavaScript 或多个 HTML 标签，仅需要添加一些 CSS 属性。这个方案简洁而有效，可以让你免于花费几个小时来寻找精巧的浏览器方案和基于 JavaScript 圆角。效果如图 1.4 所示：

8. 边框图片

`border-image` 属性允许在元素的边框上设定图片，这使得原本单调的边框样式变得丰富起来。该属性给设计师提供了一个很好的工具，用它可以方便地定义和设计元素的边框样式，比 `background-image` 属性（对高级设计来说）和枯燥的默认边框样式更好用。我们也可以明

确地定义一个边框应该如何缩放或平铺，效果如图 1.5 所示：



图 1.4 圆角效果应用

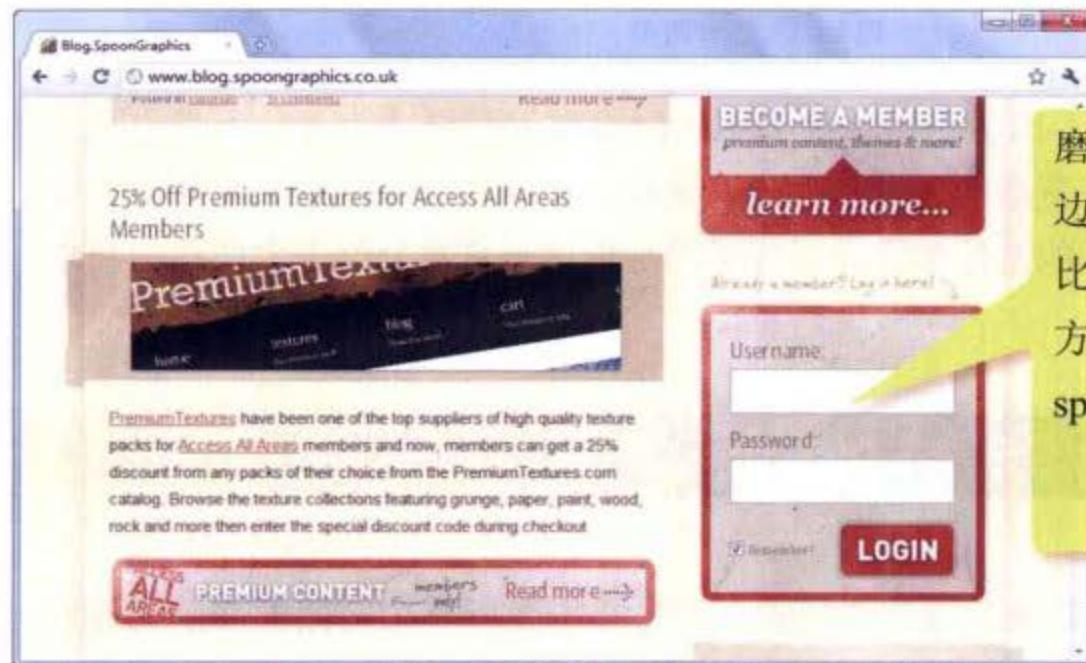


图 1.5 边框图片

9. 盒子阴影

`box-shadow` 属性可以为 HTML 元素添加阴影而不需要使用额外的标签或背景图片。`text-shadow` 属性能增强设计的细节，但并不影响内容的可读性，也不会影响页面布局。效果如图 1.6 所示：

10. 媒体查询

媒体查询（media query）可以用于为不同的显示设备定义与文能力相适配的样式。例如，在可视区域的宽度小于 480 像素的情况下，你可能想让网页的侧栏显示在主要内容的下

边，这样它就不应该浮动并显示在侧边了。



图1.6 盒子阴影

媒体查询是非常有用的，因为不用单独为不同的设备编写样式表了，而且也不需要使用 JavaScript 脚本来确定用户浏览器的属性和功能。这样就可以实现灵活的、更加流行的、基于 JavaScript 脚本解决方案的智能流体布局，以便于满足用户浏览器分辨率多样化的要求。

1.4 主流浏览器对 CSS 3 的支持

CSS 3 给我们带来了众多全新的设计体验，但是并不是所有浏览器都完全支持它。各主流浏览器都定义了自己的私有属性，以便让用户体验 CSS 3 的新特性。

这种“各自为政”的方法固然可以避免不同浏览器在解析相同属性时出现冲突，但是它也给设计师带来了诸多不便，因为不仅需要使用更多的 CSS 样式代码，而且还非常容易导致同一个页面在不同的浏览器之间表现不一致。

当然，网页不需要在所有浏览器中看起来都严格一致，有时候在某个浏览器中使用私有属性来实现特定的效果是可行的。

Webkit 类型的浏览器（如 Safari、Chrome）的私有属性是以 -webkit- 为前缀的，Gecko 类型的浏览器（如 Firefox）的私有属性是以 -moz- 为前缀的，Konqueror 类型的浏览器的私有属性是以 -khtml- 为前缀的，Opera 浏览器的私有属性是以 -o- 为前缀的，而 Internet Explorer 浏览器的私有属性是以 -ms- 为前缀的（目前只有 IE 8+ 支持 -ms- 前缀）。在 Windows 系统下，

各主流浏览器对 CSS 3 各模块的支持情况如表 1.1 所示，后面章节有对这些模块的详细讲解。

表 1.1 各主流浏览器主流版本对 CSS 3 模块的支持

模块	Chrome 4	Safari 4	Firefox 3.6	Opera 10.5	IE 9
RGBA	✓	✓	✓	✓	✓
HSLA	✓	✓	✓	✓	✓
Multiple Backgrounds	✓	✓	✓	✓	✓
Border Image	✓	✓	✓	✓	✗
Border Radius	✓	✓	✓	✓	✓
Box Shadow	✓	✓	✓	✓	✓
Opacity	✓	✓	✓	✓	✓
CSS Animations	✓	✓	✗	✗	✗
CSS Columns	✓	✓	✓	✗	✗
CSS Gradients	✓	✓	✓	✗	✗
CSS Reflections	✓	✓	✗	✗	✗
CSS Transforms	✓	✓	✓	✓	✗
CSS Transforms 3D	✓	✓	✗	✗	✗
CSS Transitions	✓	✓	✓	✓	✗
CSS FontFace	✓	✓	✓	✓	✓

1.5 CSS 3 的未来和思考

“CSS 3 和 HTML 5 将改变未来的 Web 世界。” 经过一年多的热烈讨论和实践，相信没有多少人再怀疑这种说法了。目前，CSS 3 和 HTML 5 还没有完全普及，浏览器的支持也还处于初级试验阶段，但是此时正是我们应该积极去学习和实践的时候。只有这样才能够不落伍。

当然，我们也应该明白，任何新技术的出现从来都不是没有代价的。

CSS 3 的出现意味着新标准只能得到浏览器的有限支持，而当标准草案改变时，这也意味着设计师必须回头重写代码。因为 CSS 3 规范还在不断完善中，所以就不仅是浏览器支持的问题了，因为大部分浏览器都有自己的实现途径——通过前缀的私有属性实现，这些前缀会使开发变得复杂，如果需要把这些私有属性都写入代码里，就会增加大量的额外工作，同时也会给 CSS 代码的优化带来不小的挑战。

例如，如果使用 CSS 3 设计圆角，一般应该使用 border-radius 属性进行定义，但是因为这个特性还没有最终定稿，所以浏览器只支持私有属性。Firefox 浏览器仅支持 -moz-border-radius 私有属性，而 Safari 和 Chrome 浏览器只支持 -webkit-border-radius 私有属性，Opera 浏览器只支持 -o-border-radius 私有属性。

特有的前缀都不会被验证，这不太理想，因为对于 CSS 验证器来说，遇到特定的前缀时只会提出建议而不显示错误。如果你想在 CSS 代码中使用完全标准的方法，那么现阶段就只能远离 CSS 3 了。如果想使用 CSS 3，就应该了解各浏览器的私有前缀，目前也只有使用这些私有属性才能够真正体验到 CSS 3 的魅力。

从纯粹技术的角度来看，使用带有前缀的私有属性是错误的做法，因为最终的规范一定会发布，而只有一种规则会被标准化，届时所有的浏览器都会实现和遵守这一标准。此时我们只需要简单地将既有代码中的特定前缀删除掉即可。这时候特定前缀的好处就体现出来了，因为它很容易被找到和删除。

在批判中吸取营养，这样才能让我们突破技术层面思考技术应用的未来。CSS 3 已经完成了多个模块的开发，但是仍然有很多模块尚待规范，而且没有一个明确的时间表。不管现实如何，各主流浏览器都将会全面支持 CSS 3 的新特性，一些新的探索已经开始了。

CSS 3 将完全向后兼容，所以没有必要修改现在的设计来让它们继续运作，浏览器也会继续支持 CSS 2。对我们来说，CSS 3 带来的改变是不仅能让我们实现新的设计效果（如动态和渐变），而且可以用更简单的方式实现既有的效果（如分栏等）。

CSS 3 确实给我们带来了很多全新的体验，但仍有一些负面的因素必须考虑，我们把这些因素列举出来，以便于我们理性地学习和使用 CSS 3。

- 落伍的 IE 浏览器：46% 的 CSS 3 新功能在当前版本的 IE 中无法看到效果。这种现状会逐渐被改善，因为即将发布的 IE 9 会有很大的进步。出于安全性的考虑，我们应该尽量避免将这些新技术用于重要的设计中，同时应保证，当这些效果不起作用时有替代的解决方案。
- CSS 验证问题：因为目前的 CSS 3 规范并非是最终版本，不同的浏览器都在使用自己的私有属性现它的新功能，这可能会为 CSS 验证埋下隐患。
- 代码臃肿：因为不同的浏览器要使用不同的方法，最终将导致所设计的 CSS 代码十分臃肿。
- 不恰当的使用：对这些特性的不当使用可能带来一些负面效果，阴影效果就是一个例子。

当然，理性的思考并不意味着畏缩，读者应该知难而进，同时也应该明白学习新技术可能会遇到的困难和风险。只有这样，才能够驾驭 CSS 3。

CSS 3 新增的选择器

选择器（Selector，也译为选择符，本书统一为选择器）是 W3C 在 CSS 3 工作草案中独立引入的一个概念（请参阅 <http://www.w3.org/TR/css3-selectors/>）。实际上，在 CSS 1 和 CSS 2 已经非系统性地定义了很多常用选择器，这些选择器基本上能够满足 Web 设计师常规的设计需求。

为了便于初学者温故而知新，我们不妨先回顾一下在 CSS 1 和 CSS 2 中都定义了哪些选择器。CSS 1 中定义的选择器见表 2.1。

表 2.1 CSS 1 中定义的选择器

选择器	类型	说明
E	类型选择器	选择指定类型的元素
E#myid	ID 选择器	选择匹配 E 的元素，且匹配元素的 id 属性值等于 myid。 注意，E 选择符可以省略，表示选择指定 id 属性值等于 myid 的任意类型的元素
E.warning	类选择器	选择匹配 E 的元素，且匹配元素的 class 属性值等于 warning。 注意，E 选择符可以省略，表示选择指定 class 属性值等于 warning 的任意类型的任意多个元素
E F	包含选择器	选择匹配 F 的元素，且该元素被包含在匹配 E 的元素内。 注意，E 和 F 不仅仅是指类型选择器，可以任意合法的选择符组合
E:link	链接伪类选择器	选择匹配 E 的元素，且匹配元素被定义了超链接并未被访问。 例如，a:link 选择器能够匹配已定义 URL 的 a 元素
E:visited	链接伪类选择器	选择匹配 E 的元素，且匹配元素被定义了超链接并已被访问。 例如，a:visited 选择器能够匹配已被访问的 a 元素

(续)

选择器	类型	说明
E:active	用户操作伪类选择器	选择匹配 E 的元素，且匹配元素被激活
E:hover	用户操作伪类选择器	选择匹配 E 的元素，且匹配元素正被鼠标经过
E:focus	用户操作伪类选择器	选择匹配 E 的元素，且匹配元素获取了焦点
E::first-line	伪元素选择器	选择匹配 E 元素内的第一行文本
E::first-letter	伪元素选择器	选择匹配 E 元素内的第一个字符

CSS 1 中的选择器的功能是非常弱的，覆盖范围也非常单薄。例如，上表最后 3 个选择器在 CSS 2 中已经被重新定义，目的是规范和增强这些选择器的功能。升级到 CSS 2 后，选择器类型和功能都获得了极大的扩充和增强，以便 Web 设计师在复杂结构中能自由渲染页面。CSS 2 中定义的选择器见表 2.2。

表 2.2 CSS 2 版本定义的选择器

选择器	类型	说明
*	通配选择器	选择文档中所有的元素
E[foo]	属性选择器	选择匹配 E 的元素，且该元素定义了 foo 属性。 注意，E 选择符可以省略，表示选择定义了 foo 属性的任意类型的元素
E[foo="bar"]	属性选择器	选择匹配 E 的元素，且该元素将 foo 属性值定义为了 "bar"。 注意，E 选择符可以省略，用法与上一个选择器类似
E[foo~="bar"]	属性选择器	选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值是一个以空格符分隔的列表，其中一个列表的值为 "bar" 注意，E 选择符可以省略，用法与上一个选择器类似 例如，a[title~="bar1"] 匹配 ，而不匹配
E[foo = "en"]	属性选择器	选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值是一个用连字符 (-) 分隔的列表，值开头的字符为 "en"。 注意，E 选择符可以省略，用法与上一个选择器类似。 例如，[lang = "en"] 匹配 <body lang="en-us"></body>，而不匹配 <body lang="fr-argot"></body>
E:first-child	结构伪类选择器	选择匹配 E 的元素，且该元素为父元素的第一个子元素

(续)

选择器	类型	说明
E:lang(fr)	:lang() 伪类选择器	选择匹配 E 的元素，且该元素显示内容的语言类型为 fr
E::before	伪元素选择器	在匹配 E 的元素前面插入内容
E::after	伪元素选择器	在匹配 E 的元素后面插入内容
E > F	子包含选择器	选择匹配 F 的元素，且该元素为所匹配 E 的元素的子元素。注意，E 和 F 不仅仅是指类型选择器，可以是任意合法的选择符组合
E + F	相邻兄弟选择器	选择匹配 F 的元素，且该元素位于所匹配 E 的元素后面相邻的位置。注意，E 和 F 不仅仅是指类型选择器，可以是任意合法的选择符组合

2.1 属性选择器

CSS 3 新增加了 3 个属性选择器，详细说明如表 2.3 所示。这 3 个属性选择器与 CSS 2.1 中已经定义的 4 个属性选择器共同构成了 CSS 的功能强大的标签属性过滤体系。

表 2.3 CSS 3 新增的属性选择器列表

选择器	说明
E[foo^="bar"]	选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值包含前缀为 "bar" 的子字符串。注意，E 选择符可以省略，表示可匹配任意类型的元素。 例如，body[lang^="en"] 匹配 <body lang="en-us"></body>，而不匹配 <body lang="fr-argot"></body>
E[foo\$="bar"]	选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值包含后缀为 "bar" 的子字符串。注意，E 选择符可以省略，表示可匹配任意类型的元素。 例如，img[src\$="jpg"] 匹配 ，而不匹配
E[foo*="bar"]	选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值包含 "bar" 的子字符串。注意，E 选择符可以省略，表示可匹配任意类型的元素。 例如，img[src\$="jpg"] 匹配 ，而不匹配

CSS 3 遵循了惯用的编码规则，选用了 ^、\$ 和 * 这 3 个通用匹配运算符，其中，^

表示匹配起始符，\$ 表示匹配终止符，* 表示匹配任意字符，使用它们更符合编码习惯和惯用编程思维。

CSS 3 草案还保留了对选择器 E[foo~="bar"] 和 E[foo|= "en"] 的支持。实际上，E[foo*="b ar"] 和 E[foo^="bar"] 更符合用户使用习惯，可以使用 E[foo*="bar"] 替换 E[foo~="bar"] 和 E[foo|= "en"]，或者使用 E[foo^="bar"] 替换 E[foo|= "en"]，两者执行效率相差无几。

浏览器兼容性检测

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

检验表明，新增的 3 个属性选择器可以在实践中放心使用，不用担心浏览器兼容问题，也不用考虑 IE 浏览器的版本问题。虽然 IE 6 不支持这些选择器，但是它最终会被淘汰。

实战体验：文档共享的友善之举

翻阅百度文库（<http://wenku.baidu.com/>）的下载资料列表，你会发现在下载文档列表的超链接前面都会显示一个文档类型图标（如图 2.1 所示），这是一种体验非常好的设计细节。实际上，实现的技术细节并不难，使用属性选择器就可以轻松实现。



图 2.1 显示类型图标的文档下载链接

CSS 设计思路

使用属性选择器匹配 a 元素中 href 属性值的最后几个字符。由于下载文档的类型不同，文件的扩展名也会不同，根据扩展名不同，分别为不同文件类型的超链接添加不同的图标即可。

根据这一设计思路，我们可以编写一个简化的示例代码（如下所示），演示效果如图 2.2 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>超级链接类型标识图标</title>
<style type="text/css">
p { margin:4px; }
a[href^="http:"] { /* 匹配所有有效超链接 */
    background: url(images/window.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".pdf"] { /* 匹配 PDF 文件 */
    background: url(images/icon_pdf.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".xls"] { /* 匹配 XML 样式表文件 */
    background: url(images/icon_xls.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".ppt"] { /* 匹配演示文稿 */
    background: url(images/icon_ppt.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".rar"] { /* 匹配压缩文件 */
    background: url(images/icon_rar.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".gif"] { /* 匹配 GIF 图像文件 */
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".jpg"] { /* 匹配 JPG 图像文件 */
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".png"] { /* 匹配 PNG 图像文件 */
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$=".txt"] { /* 匹配记事本文件 */
    background: url(images/icon_txt.gif) no-repeat left center;
}
```

```

padding-left: 18px;
}
</style>
</head>
<body>
<h1>超级链接类型标识图标 </h1>
<p><a href="http://www.baidu.com/name.pdf">PDF 文件 </a> </p>
<p><a href="http://www.baidu.com/name.ppt">PPT 文件 </a> </p>
<p><a href="http://www.baidu.com/name.xls">XLS 文件 </a> </p>
<p><a href="http://www.baidu.com/name.rar">RAR 文件 </a> </p>
<p><a href="http://www.baidu.com/name.gif">GIF 文件 </a> </p>
<p><a href="http://www.baidu.com/name.jpg">JPG 文件 </a> </p>
<p><a href="http://www.baidu.com/name.png">PNG 文件 </a> </p>
<p><a href="http://www.baidu.com/name.txt">TXT 文件 </a> </p>
<p><a href="http://www.baidu.com/#anchor"># 锚点超链接 </a></p>
<p><a href="http://www.baidu.com/">http://www.baidu.com/</a></p>
</body>
</html>

```

这就是规律，你发现了吗？

你注意到了吗：a[href^="http:"]{...}这个样式位于顶部，设计之初我把它放在了最后，但是却让俺走了一个很大的弯路。你知道为什么吗？如果不知道，试一试！



在CSS 3的属性选择器还没有得到普及之前，设计师们只能辛苦地使用JavaScript脚本来实现这类效果

图2.2 在IE 8中的演示效果

2.2 结构伪类选择器

结构伪类（Structural pseudo-classes）是 CSS 3 新增的类型选择器。顾名思义，结构伪类就是利用文档结构树（DOM）实现元素过滤，也就是说，通过文档结构的相互关系来匹配特定的元素，从而减少文档内对 class 属性和 ID 属性的定义，使得文档更加简洁，详细说明如表 2.4 所示。

表 2.4 CSS 3 结构伪类选择器列表

选择器	说明
E:root	选择匹配 E 所在文档的根元素。注意，在(X)HTML 文档中，根元素就是 html 元素，此时该选择器与 html 类型选择器匹配的内容相同
E:nth-child(n)	选择所有在其父元素中第 n 个位置的匹配 E 的子元素。 注意，参数 n 可以是数字（1、2、3）、关键字（odd、even）、公式（2n、2n+3），参数的索引起始值为 1，而不是 0。例如： tr:nth-child(3) 匹配所有表格里排第 3 行的 tr 元； tr:nth-child(2n+1) 匹配所有表格的奇数行； tr:nth-child(2n) 匹配所有表格的偶数行； tr:nth-child(odd) 匹配所有表格的奇数行； tr:nth-child(even) 匹配所有表格的偶数行
E:nth-last-child(n)	选择所有在其父元素中倒数第 n 个位置的匹配 E 的子元素。 注意，该选择器的计算顺序与 E:nth-child(n) 相反，但语法和用法相同
E:nth-of-type(n)	选择父元素中第 n 个位置，且匹配 E 的子元素。 注意，所有匹配 E 的子元素被分离出来单独排序。非 E 的子元素不参与排序。 参数 n 可以是数字（1、2、3）、关键字（odd、even）、公式（2n、2n+3），参数的索引起始值为 1，而不是 0。 例如，p:nth-of-type(2) 匹配 <div><h1></h1><p></p><p></p></div> 片段中第 2 个 p 元素，但不匹配片段中位于第 2 个位置的 h1 元素
E:nth-last-of-type(n)	选择父元素中倒数第 n 个位置，且匹配 E 的子元素。 注意，该选择器的计算顺序与 E:nth-of-type(n) 相反，但语法和用法相同
E:last-child	选择位于其父元素中最后一个位置，且匹配 E 的子元素。 例如，h1:last-child 匹配 <div><p></p><h1></h1></div> 片段中 h1 元素
E:first-of-type	选择在其父元素中匹配 E 的第一个同类型的子元素。 注意，该选择器的功能类似于 E:nth-of-type(1)。 例如，p:first-of-type 匹配 <div><h1></h1><p></p><p></p></div> 片段中的第一个 p 元素

(续)

选择器	说明
E:last-of-type	选择在其父元素中匹配 E 的最后一个同类型的子元素。 注意，该选择器的功能类似于 E:nth-last-of-type(1)。 例如，p:last-of-type 匹配 <div><h1></h1><p></p><p></p></div> 片段中的第二个 p 元素
E:only-child	选择其父元素只包含一个子元素，且该子元素匹配 E。 例如，p:only-child 匹配 <div><p></p></div> 片段中的 p 元素，但不匹配 <div><h1><h1><p></p></div> 片段中的 p 元素
E:only-of-type	选择其父元素只包含一个同类型的子元素，且该子元素匹配 E。 例如，p:only-of-type 匹配 <div><p></p></div> 片段中的 p 元素，也匹配 <div><h1><h1><p></p></div> 片段中的 p 元素
E:empty	选择匹配 E 的元素，且该元素不包含子节点。注意，文本也属于节点

浏览器兼容性检测

各主流浏览器对结构伪类选择器的支持存在较大的差异：IE 8 及其以下版本的浏览器完全不支持结构伪类选择器；Firefox 从 3.5 版本开始全面支持，Firefox 3.5 以前的版本支持不是很完善；Opera、Safari 和 Chrome 浏览器对结构伪类选择器的支持比较完善。

鉴于此，在当前浏览器环境下，结构伪类选择器还不适合大范围使用，可作为技术学习和交流使用。在主流浏览器完全支持该类型选择器之前，建议读者辅助 JavaScript 脚本模拟类似的过滤功能，jQuery 等主流 JavaScript 类库中都提供了类似的脚本化结构伪类选择器。

浏览器兼容检测的结果如下所示：

1. E:nth-child(n)、E:nth-last-child(n)、E:nth-of-type(n)、E:nth-last-of-type(n)、E:first-of-type、E:only-of-type、E:last-of-type



✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

2. E:last-child、E:only-child、E:empty

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

3.E:root

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验 1：设计优雅的数据表格

大量的数据显示在一起时容易影响用户的阅读体验。例如，长时间阅读数据易引起视觉疲劳，以及诱发错行误读等问题。因此，Web 设计师在设计数据表格时应该考虑到用可能会遇到的这些问题。建议采用隔行或隔列换色、动态背景色等方法来提升用户浏览大容量数据时的体验。

CSS 设计思路

隔行分色是最经典的数据表设计样式，它主要是从易用性的角度来考虑的，以提高用户浏览数据的速度和准确度。传统设计方法是先定义一个样式类，然后把该类应用到所有奇数行或偶数行上。当然，这种方法会增加设计师的工作量，也给文档添加了很多不必要的属性。采用 CSS 3 的结构伪类选择器后，一切都显得那么轻松自在。可以使用 E:nth-child(n) 选择器快速为偶数行或奇数行定义分色背景，从而便于用户浏览，效果如图 2.3 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>设计优雅的数据表格</title>
<style type="text/css">
```

```

h1 { font-size:16px; }
table /* 色彩恬淡的细表格是设计的主流 */
{
    width:100%;
    font-size:12px;
    table-layout:fixed;
    empty-cells:show;
    border-collapse: collapse;
    margin:0 auto;
    border:1px solid #cad9ea;
    color:#666;
}
th {
    /* 使用背景装饰列表头可以让表格看起来更别致 */
    background-image: url(images/th_bg1.gif);
    background-repeat:repeat-x;
    height:30px;
    overflow:hidden;
}
td { height:20px; } /* 适当撑起单元格，让数据看起来更轻松 */
td, th {
    /* 浅色线分隔数据行和列，会让表格看起来更清爽，而不是那么生硬 */
    border:1px solid #cad9ea;
    padding:0 1em 0;
}
tr:nth-child(even) {
    background-color:#f5fafc;
}
</style>
</head>
<body>
<h1>设计优雅的数据表格 </h1>
<table summary="设计优雅的数据表格">
    <tr>
        <th>排名 </th>
        <th>校名 </th>
        <th>总得分 </th>
        <th>人才培养总得分 </th>
        <th>研究生培养得分 </th>
        <th>本科生培养得分 </th>
        <th>科学研究总得分 </th>
        <th>自然科学研究得分 </th>
        <th>社会科学研究得分 </th>
        <th>所属省份 </th>
        <th>分省排名 </th>
        <th>学校类型 </th>
    </tr>
    <tr>
        <td>1</td>
        <td>清华大学 </td>
        <td>296.77</td>
        <td>128.92</td>
        <td>93.83</td>
    </tr>

```

聪明的设计师在设计表格样式时，总习惯加上这样几个声明。好处显而易见：table-layout:fixed能改善表格呈现性能，empty-cells:show能够隐藏不必要的干扰因素，border-collapse: collapse能让表格看起来更精致

这个样式是本案例的关键，通过结构伪类选择器为表格内的偶数行定义背景色，以实现隔行分色的显示效果

不同的行使用不同的标签，请注意结构的语义性

```

<td>35.09</td>
<td>167.85</td>
<td>148.47</td>
<td>19.38</td>
<td width="16">京 </td>
<td width="12">1 </td>
<td>理工 </td>
</tr>
.....
</table>
</body>
</html>

```

在这里我们省略了大量重
复、不必要的结构代码

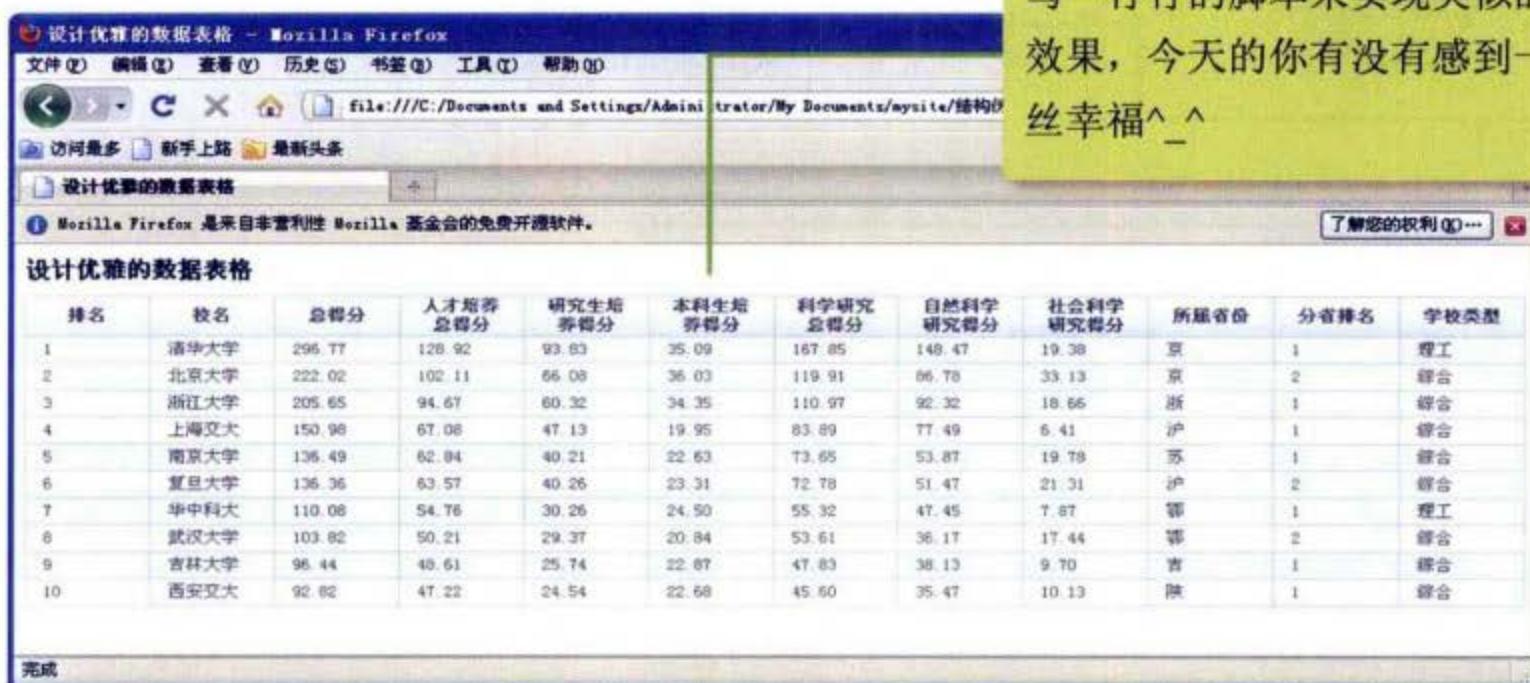


图2.3 在Firefox 3.5中的演示效果

实战体验 2：CSS 大战保龄球

CSS 3 定义了 12 种结构伪类选择器，功能超过了 jQuery 的同类子元素选择器。为了方便读者识记，我们把它们归纳为了四大类：

- 1) 通用子元素过滤器：如 E:nth-child(n)（顺序过滤）和 E:nth-last-child(n)（倒序过滤）。
- 2) 通用子类型元素过滤器：如 E:nth-of-type(n)（顺序过滤）和 E:nth-last-of-type(n)（倒序过滤）。
- 3) 特定位置的子元素：如以所有子元素为参考的 E:first-child 和 E:last-child，以子元素类型为参考的 E:first-of-type 和 E:last-of-type。
- 4) 特定状态的元素：如 E:root（根节点）、E:only-child（独子元素）、E:only-of-type（独子类型元素）和 E:empty（孤节点）。

CSS 设计思路

在下面这个案例中，通过灵活使用各种结构伪类选择器控制页面中保龄球的显示大小、位置和效果，初步演示效果如图 2.4 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS 大战保龄球 </title>
<style type="text/css">
h1 { font-size:16px; }

dl,dt,dd {
    padding:0;
    margin:0;
}

dl {
    position:relative;
    width:778px;
    height:612px;
    background:url(images/bg3.jpg) no-repeat left top;
}

dt /* 把公共样式统一放在前面，能够优化代码 */
position:absolute;
color:red;
font-size:20px;
font-weight:bold;

}

img {
    width:50px;
    position:absolute;
}

/* 以下样式控制 dd 内的 img 元素，设计图像显示效果 */
dd:nth-of-type(1) img /* 第一行第 1 个保龄球 */
left:370px;
top:280px;
z-index:1000;

dd:nth-of-type(2) img /* 第二行第 2 个保龄球 */
left:330px;
top:250px;
width:40px;
z-index:100;

dd:nth-of-type(3) img /* 第二行第 2 个保龄球 */
left:390px;
```

网页设计，应先清除浏览器默认的样式，这个习惯有时很重要

这个声明很重要，它能够强迫保龄球在 dl 元素框内定位

为盒子装饰背景图，就应该固定住它的大小，并禁止平铺

这里主要运用了 E:nth-of-type(n) 选择器，因为 dl 包含 dt 和 dd 这两个类型的元素，不适合使用 E:nth-child(n) 选择器

```
top:250px;  
width:40px;  
z-index:100;  
}  
dd:nth-of-type(4) img /* 第三行第 1 个保龄球 */  
left:300px;  
top:220px;  
width:30px;  
z-index:10;  
}  
dd:nth-of-type(5) img /* 第三行第 2 个保龄球 */  
left:350px;  
top:220px;  
width:30px;  
z-index:10;  
}  
dd:nth-of-type(6) img /* 第三行第 3 个保龄球 */  
left:405px;  
top:220px;  
width:30px;  
z-index:10;  
}  
dd:nth-of-type(7) img /* 第四行第 1 个保龄球 */  
left:270px;  
top:190px;  
width:20px;  
}  
dd:nth-of-type(8) img /* 第四行第 2 个保龄球 */  
left:320px;  
top:190px;  
width:20px;  
}  
dd:nth-of-type(9) img /* 第四行第 3 个保龄球 */  
left:370px;  
top:190px;  
width:20px;  
}  
dd:nth-of-type(10) img /* 第四行第 4 个保龄球 */  
left:420px;  
top:190px;  
width:20px;  
}  
/* 以下样式控制 dt 元素，设计数字显示效果 */  
dt:nth-of-type(1) /* 第一行数字 1 */  
left:392px;  
top:370px;  
z-index:1001;  
}
```

有规律地改变（渐变）
保龄球的大小和位置，
可以设计出立体效果

```
dt:nth-of-type(2) /* 第二行数字 2 */  
    left:344px;  
    top:320px;  
    z-index:101;  
}  
dt:nth-of-type(3) /* 第二行数字 3 */  
    left:408px;  
    top:320px;  
    z-index:101;  
}  
dt:nth-of-type(4) /* 第三行数字 4 */  
    left:310px;  
    top:270px;  
    z-index:11;  
}  
dt:nth-of-type(5) /* 第三行数字 5 */  
    left:360px;  
    top:270px;  
    z-index:11;  
}  
dt:nth-of-type(6) /* 第三行数字 6 */  
    left:415px;  
    top:270px;  
    z-index:11;  
}  
dt:nth-of-type(7) /* 第四行数字 7 */  
    left:274px;  
    top:220px;  
    z-index:1;  
}  
dt:nth-of-type(8) /* 第四行数字 8 */  
    left:324px;  
    top:220px;  
    z-index:1;  
}  
dt:nth-of-type(9) /* 第四行数字 9 */  
    left:374px;  
    top:220px;  
    z-index:1;  
}  
dt:nth-of-type(10) /* 第四行数字 10 */  
    left:420px;  
    top:220px;  
    z-index:1;  
}  
</style>  
</head>  
<body>
```

请注意叠放顺序

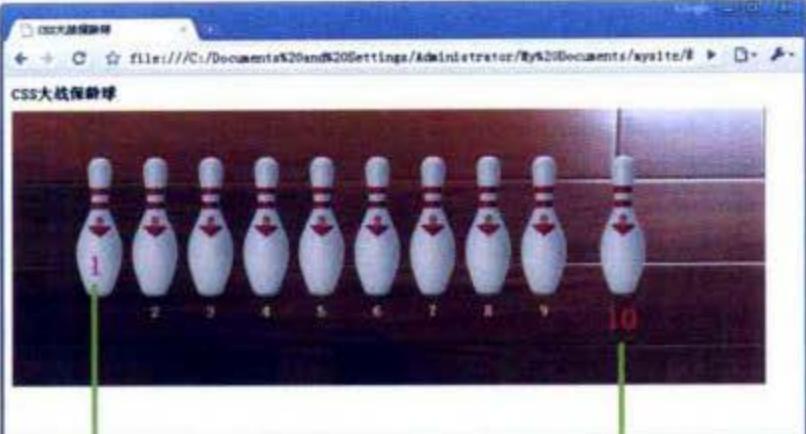
```
<h1>CSS 大战保龄球 </h1>
<dl>
    <dt>1</dt>
    <dd></dd>
    <dt>2</dt>
    <dd></dd>
    <dt>3</dt>
    <dd></dd>
    <dt>4</dt>
    <dd></dd>
    <dt>5</dt>
    <dd></dd>
    <dt>6</dt>
    <dd></dd>
    <dt>7</dt>
    <dd></dd>
    <dt>8</dt>
    <dd></dd>
    <dt>9</dt>
    <dd></dd>
    <dt>10</dt>
    <dd></dd>
</dl>
</body>
</html>
```

设计科学、合理的HTML结构，能够让你的CSS代码更精巧，维护更轻松



图2.4 在Chrome 1中的演示效果

CSS 3实战

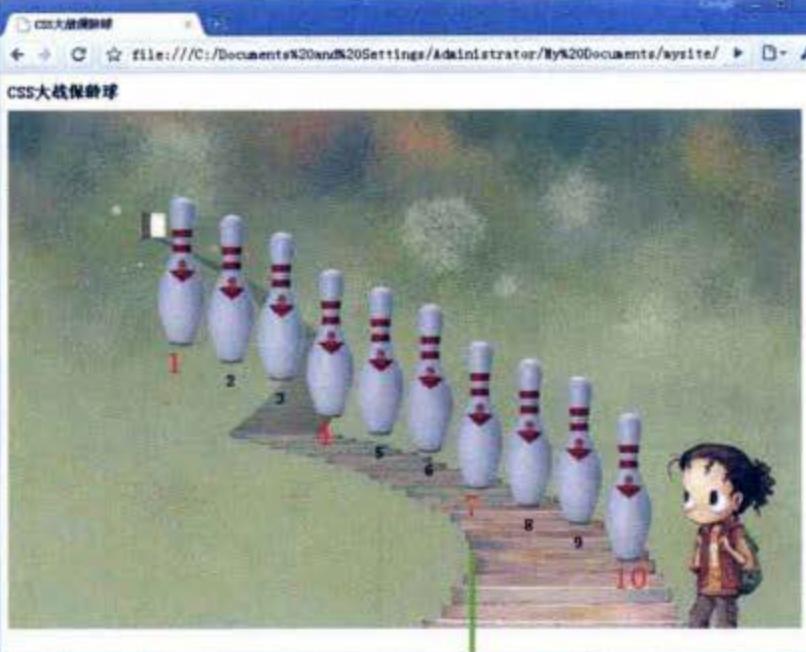


```

<style type="text/css">
  h1 { font-size:16px; }
  dl, dt, dd { padding:0; margin:0; }
  dl {
    width:700px;
    height:200px;
    background:url(images/bg4.jpg) no-repeat right bottom;
    padding:50px;
  }
  dt {
    float:left;
    position:relative;
    left:30px;
    top:160px;
    font-weight:bold;
    color:white;
  }
  dd { float:left; }
  img { width:50px; }
  dt:nth-last-of-type(1) {
    color:red;
    font-size:2em;
    left:40px;
  }
  dt:first-of-type {
    color:#C09;
    font-size:2em;
    top:100px;
    left:30px;
  }
</style>

```

不改变结构，稍微改变CSS样式，就能让页面布局发生变化



```

<style type="text/css">
  h1{font-size:16px;}
  dl,dt,dd{padding:0; margin:0; }
  dl {
    width:650px;
    height:432px;
    background:url(images/bg5.jpg)
      no-repeat right bottom;
    padding:50px 0 50px 150px;
  }
  dt{
    position:relative;
    left:20px; top:180px;
    font-weight:bold;
  }
  dd{float:left;}
  img{width:50px; }
  dt:nth-last-child(3n+2){/* 倒序隔行
    配合 dd 元素 */
    color:red;
    font-size:2em;
    left:10px;
    top:190px;
  }
</style>

```

公式 $3n+2$ 解析：

2（凡偶数）表示在dl子元素中匹配偶数行，即匹配所有dd元素；
 $3n$ （变量前常数表示步长）表示从最后一个dd开始，每隔3个选取一个dd元素。匹配效果如上图所示

图2.5 在Firefox 3.5中的演示效果

上面示例仅演示了 `E:nth-of-type(n)` 选择器的使用。对于读者来说，这不是我们的目的，在下面的练习中，你会看到更多结构伪类选择器粉墨登场。

实战体验 3：让枯燥的列表更有趣

网站就如同一个小雪球，随着时间的流逝，它会越滚越大。一个最简单的例子就是，很多网站的导航列表在一天天“发胖”，变得臃肿伤目。但是如果适当打扮一下，也许就不会觉得难看了，我们来看看国内两大门户网站的导航栏，如图 2.6 所示。



图 2.6 搜狐网和新浪网的导航列表

CSS 设计思路

如果在长长的导航列表中悄悄地在结构中“塞进”几个 class 属性，页面效果也许并不会受太大影响，但是这样的结构总让人不放心，改版或增减栏目都会带来问题，清理和增加 class 属性是件很费心的事情。

如果使用结构伪类选择器，这些问题都可以轻松解决，不用添加很多不必要的元素。下面这个案例模拟新浪的导航列表样式，演示了结构伪类在多列表结构中的应用价值，效果如图 2.7 所示。

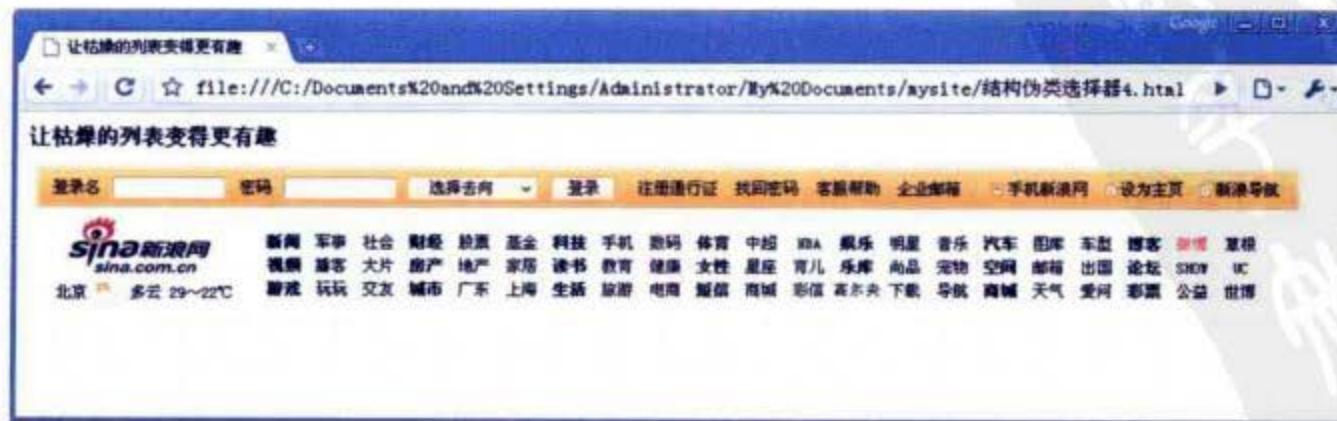


图 2.7 在 Chrome 1 中的演示效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>让枯燥的列表更有趣</title>
<style type="text/css">
h1 { font-size:16px; }

#menu {
    width:965px;
    height:126px;
    background:url(images/bg6.jpg) no-repeat right bottom;
}

ul, li { /* 清除列表默认样式，这是一个好的习惯，切记 */
    padding:0;
    margin:0;
    list-style:none;
}
ul { /* 通过外框定位列表位置和大小 */
    float:right;
    margin-right:0px;
    margin-top:55px;
    width:790px;
    font-size:12px;
}
li { /* 设计列表项样式 */
    float:left;
    width:36px;
    padding:0 0 4px 0;
    text-align:center;
    background:url(images/line1.gif) no-repeat left center;
}
li:nth-child(3n+1) { /* 匹配1、4、7、10、13……（步长为3）项的列表项 */
    font-weight:bold;
    background:none; /* 清除加粗列表项的背景 */
}
li:nth-child(55) { font-size:11px; } /* 为特定位置列表项定义样式，缩小显示 */
li:nth-child(22n+1){ margin-left:-1px; } /* 微调每行第一个列表项，以便对齐 */
li:nth-child(20){ color:red; } /* 为特定位置的列表项定义样式，突出显示 */
</style>
</head>
<body>
<h1>让枯燥的列表更有趣</h1>
<div id="menu">
    <ul>
        <li>新闻</li>
        .....
        <li>世博</li>
    </ul>
</div>

```

偷个懒，以背景图的形式直接借用了新浪的模板

浮动显示列表项，该方法比较灵活，但也容易出现错位和排列不整齐的问题。解决的方法就是：固定外框和项目的宽度

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

在类似排行榜这样的排序栏目中，我们经常会看到各种标识新颖的前缀小图标（如图 2.8 所示），可以使用结构伪类选择器轻松实现这种效果。其设计方法是：匹配列表中不同位置的列表项，分别为它们定义不同的背景图标。这样就不用在结构中添加大量无意义的标签，让结构变得简洁，也便于维护和扩展。

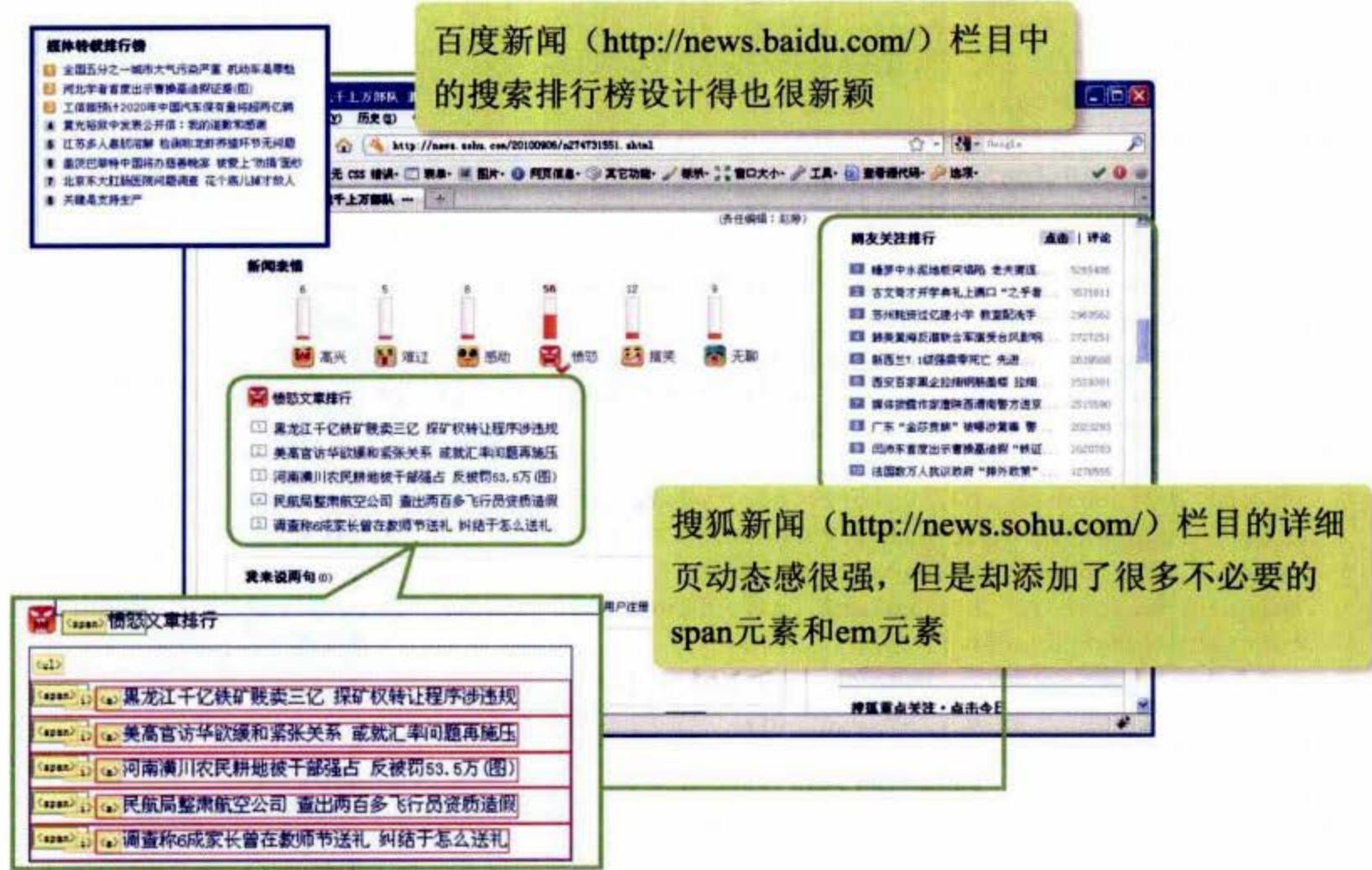


图 2.8 结构伪类选择器可以在排行列表中发挥作用

实战体验 4：清理圆角边框中的垃圾标签

从 2008 年开始，网页设计中开始流行圆角化设计。基于圆角设计的方法也是举不胜举，如一图钉圆角、二图钉圆角、四图钉圆角、山羊角、纯 CSS 圆角、JavaScript 圆角等。

效果相同，但方法却不尽相同，这本无可非议，但是使用纯 CSS 设计的圆角却给网站的升级埋下了深深的隐患。大量无意义的标签充斥文档，怎么办？为了迎接 CSS 3 时代的网页重构，Web 设计师需要重新思考这个问题。

请看下面这个页面，为了设计圆角框，6 个栏目就多出了 48 个无用的标签（**b** 元素）。圆角效果确实好看（如图 2.9 所示），但是也让文档解析和网页重构不堪重负。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title> 纯 CSS 的圆角 </title>
<style type="text/css">
.wrapper{width:98%;margin:0 auto;}
.sharp{width:30%;margin:20px auto 0;float:left; margin-right:2%;}
.content{height:180px; border-right:1px solid; border-left:1px solid; overflow:hidden;}
.b1,.b2,.b3,.b4,.b5,.b6,.b7,.b8{height:1px; font-size:1px; overflow:hidden; display:block;}
.b1,.b8{margin:0 5px;}
.b2,.b7{margin:0 3px; border-right:2px solid; border-left:2px solid;}
.b3,.b6{margin:0 2px; border-right:1px solid; border-left:1px solid;}
.b4,.b5{margin:0 1px; border-right:1px solid; border-left:1px solid; height:2px;}
```

通过CSS控制标签的显示大小和位置，以模拟锯齿形圆角的堆叠效果，呈现圆滑顶角的效果

```

/* 蓝色边框 */
.color1 .b2,.color1 .b3,.color1 .b4,.color1 .b5,.color1 .b6,.color1 .b7,.color1 .content{border-color:#96C2F1;} /* 边框色 */
.color1 .b1,.color1 .b8{background:#96C2F1;} /* 边框色 */
.color1 .b2,.color1 .b3,.color1 .b4,.color1 .b5,.color1 .b6,.color1 .b7,.color1 .content{background:#EFF7FF;} /* 背景色 */

/* 绿色边框 */
.color2 .b2,.color2 .b3,.color2 .b4,.color2 .b5,.color2 .b6,.color2 .b7,.color2 .content{border-color:#9BDF70;} /* 边框色 */
.color2 .b1,.color2 .b8{background:#9BDF70;} /* 边框色 */
.color2 .b2,.color2 .b3,.color2 .b4,.color2 .b5,.color2 .b6,.color2 .b7,.color2 .content{background:#F0FBEB;} /* 背景色 */

/* 绿色边框 */
.color3 .b2,.color3 .b3,.color3 .b4,.color3 .b5,.color3 .b6,.color3 .b7,.color3 .content{border-color:#BBE1F1;} /* 边框色 */
.color3 .b1,.color3 .b8{background:#BBE1F1;} /* 边框色 */
.color3 .b2,.color3 .b3,.color3 .b4,.color3 .b5,.color3 .b6,.color3 .b7,.color3 .content{background:#EEFAFF;} /* 背景色 */

/* 粉色边框 */
.color4 .b2,.color4 .b3,.color4 .b4,.color4 .b5,.color4 .b6,.color4 .b7,.color4 .content{border-color:#E3E197;} /* 边框色 */
.color4 .b1,.color4 .b8{background:#E3E197;} /* 边框色 */
.color4 .b2,.color4 .b3,.color4 .b4,.color4 .b5,.color4 .b6,.color4 .b7,.color4 .content{background:#FFFFDD;} /* 背景色 */

/* 粉色边框 */
.color5 .b2,.color5 .b3,.color5 .b4,.color5 .b5,.color5 .b6,.color5 .b7,.color5 .content{border-color:#F8B3D0;} /* 边框色 */
.color5 .b1,.color5 .b8{background:#F8B3D0;} /* 边框色 */
.color5 .b2,.color5 .b3,.color5 .b4,.color5 .b5,.color5 .b6,.color5 .b7,.color5 .content{background:#FFF5FA;} /* 背景色 */

/* 黄色边框 */
.color6 .b2,.color6 .b3,.color6 .b4,.color6 .b5,.color6 .b6,.color6 .b7,.color6 .content{border-color:#FFCC00;} /* 边框色 */
.color6 .b1,.color6 .b8{background:#FFCC00;} /* 边框色 */
.color6 .b2,.color6 .b3,.color6 .b4,.color6 .b5,.color6 .b6,.color6 .b7,.color6 .content{background:#FFFFF7;} /* 背景色 */

```

以上样式代码都是用来实现不同色彩的圆角效果的

```

</style>
</head>
<body>
<h1> 纯 CSS 的圆角 </h1>
<div class="wrapper">
    <div class="sharp color1">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    <div class="sharp color2">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    <div class="sharp color3">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    <div class="sharp color4">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    <div class="sharp color5">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    <div class="sharp color6">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
</div>
</body>
</html>

```

这些高亮显示的代码都是为了模拟圆角而添加的无意义标签。每个圆角框需要8个多余的标签



图2.9 可爱的圆角

如今，CSS 3 定义了实现圆角效果的 border-radius 属性。设计师轻松使用一个声明即可避免无数不眠之夜的辛苦，但是“历史上”为了实现圆角效果而遗留下的众多垃圾标签又该如何清理呢？

CSS 设计思路

对于网站中为了实现圆角效果而设计的样式表，我们快速切断样式表链接即可，或者在样式表中清理这些圆角样式。但是，对于那些散落于网站中各个页面中的圆角标签，清理难度就非常大了。为了解决这个问题，设计师们不妨借助结构伪类选择器，快速过滤掉这些垃圾标签，把它们隐藏起来。实现的 CSS 代码如下，效果如图 2.10 所示。



图2.10 在Chrome 4中的预览效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>清理圆角边框中的垃圾标签 </title>
<style type="text/css">
.wrapper{width:98%;margin:0 auto;}
.sharp{width:30%;margin:20px auto 0;float:left}
.content{height:180px;overflow:hidden; border:1px solid #ccc; position: relative; width: 100%}
.content{ /* 定义圆角边框样式 */
    border-radius:10px;
}
.sharp b:empty { /* 清除垃圾标签 */
    display:none;
}
/* 蓝色边框 */
.color1 .content{background:#EFF7FF; border-color:#96C2F1;}
/* 绿色边框 */
.color2 .content{background:#F0FBEB; border-color:#9BDF70;}
/* 绿色边框 */
.color3 .content{background:#EEFAFF; border-color:#BBE1F1;}
/* 绿色边框 */
.color4 .content{background:#FFFFDD; border-color:#E3E197;}
/* 粉色边框 */
.color5 .content{background:#FFF5FA; border-color:#F8B3D0;}
/* 黄色边框 */
.color6 .content{background:#FFFFFF7; border-color:#FFCC00;}
</style>
</head>
<body>
<h1>清理圆角边框中的垃圾标签 </h1>
<div class="wrapper">
    <div class="sharp color1">
        <b class="b1"></b><b class="b2"></b><b class="b3"></b><b class="b4"></b>
        <div class="content">
            
        </div>
        <b class="b5"></b><b class="b6"></b><b class="b7"></b><b class="b8"></b>
    </div>
    .....
</div>
</body>
</html>

```

有关border-radius圆角属性的详细讲解，请参阅本书后面章节。为了兼容不同浏览器，设计师还会用到浏览器的私有属性：

```

.content{
    border-radius:10px;
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
}

```

这里省略了相同的结构，结构与上一个案例几乎完全相同，只是更换了图像源

2.3 UI 元素状态伪类选择器

UI 元素状态伪类（The UI element states pseudo-classes）也是 CSS 3 新增的全新类型选择器。其中 UI 是 User Interface（用户界面）的简写，UI 设计是指网页的人机交互、操作逻辑、界面美观的整体设计。优秀的 UI 设计不仅是让网页更具个性和品位，还要让网页操作变

得便利和简单，充分体现网站的定位和特点。

UI 元素的状态一般包括：可用、不可用、选中、未选中、获取焦点、失去焦点、锁定、待机等。CSS 3 定义了 3 种常用的状态伪类选择器，详细说明如表 2.5 所示。

表 2.5 CSS 3 的 UI 元素状态伪类选择器列表

选择器	说明
E:enabled	选择匹配 E 的所有可用 UI 元素。 注意，在网页中，UI 元素一般是指包含在 form 元素内的表单元素。 例如，input:enabled 匹配 <form><input type="text" /><input type="button" disabled="disabled"/></form> 片段中的文本框，但不匹配该片段中的按钮
E:disabled	选择匹配 E 的所有不可用 UI 元素 注意，在网页中，UI 元素一般是指包含在 form 元素内的表单元素。 例如，input:disabled 匹配 <form><input type="text" /><input type="button" disabled="disabled"/></form> 片段中的按钮，但不匹配该片段中的文本框
E:checked	选择匹配 E 的所有可用 UI 元素。 注意，在网页中，UI 元素一般是指包含在 form 元素内的表单元素。 例如，input:checked 匹配 <form><input type="checkbox" /><input type="radio" checked="checked"/></form> 片段中的单选按钮，但不匹配该片段中的复选框

浏览器兼容性检测

除了 IE 浏览器外，各主流浏览器对 UI 元素状态伪类选择器的支持都非常好，但 IE 9 也开始全面支持这些 UI 元素状态伪类。因此，在当前状态下，考虑到 IE 7、IE 8 是国内用户数最多的浏览器，暂时不建议在页面中普及使用。但是，当 IE 9 普及之时，Web 设计师应该积极使用它们。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

什么样的表单是好用的表单——表单设计原则

UI 设计的一个核心就是让表单更可用、易用和好用。使用比较专业的术语来讲，就是表单设计应符合三层模型，即表单应该具有三种属性：感知（页面显示的布局）、对话（内容呈现的问题和回答）、关系（交互的任务结构）。例如，淘宝网注册页面的表单就很好用（如图 2.11 所示），用户注册时不会有畏惧感，这是为什么？

1. 输入会员信息 **2. 通过邮件确认** **3. 注册成功**

电子邮箱: test 请输入正确格式的邮箱

会员名: 5-20个字符,一个汉字为两个字符,推荐使用中文会员名。一旦注册成功会员名不能修改。

登录密码:

确认密码:

验证码: K7VB 看不清? 换一张
 用迅雷等下载文件客户端

同意以下协议, 提交注册

淘宝网服务协议
本协议由您与浙江淘宝网络有限公司共同签订,本协议具有合同效力。
本协议中甲方指双方合营的公司,浙江淘宝网络有限公司在协议中简称“淘宝”。
一、协议内容及签署
1.本协议内容包括协议正文及所有淘宝已发布的或将来可能发布的各类规则。所有规则为本协议不可分割的组成部分,若协议正文具有同样法律效力。除另行明示声明外,任何淘宝及其关联公司提供的服务(以下统称“淘宝网服务”)均为本协议不可分割的组成部分,若该服务具有同样法律效力。
2.本协议正文中具有同样法律效力。除另行明示声明外,任何淘宝及其关联公司提供的服务(以下统称“淘宝网服务”)均为本协议不可分割的组成部分,若该服务具有同样法律效力。

为什么要把简单的注册操作分为三步?

为什么不把提示信息直接显示出来?

为什么要把两个按钮合并在一起?

为什么要把服务协议放在最后?

晕死啦, 多么恐怖的表单, 我什么时候才能够填写完毕呢?

**当你遇到这样的表单, 你还有信心注册吗?
反思一下, 认真阅读一下右侧表单设计的金字塔模型。**

表单设计应遵循三层模型, 请好好看看这个模型中每层、每级的提示和问题

三层 提高关系层 (思考整体关系)

- 14 为什么要问这些问题?
- 13 在某个关系上, 问这个问题合适吗?
- 12 还能从其他途径获得这些信息吗?
- 11 是不是让用户做自己能做的工作?

二层 提升对话层 (把表单当成对话)

- 10 问题的前后顺序要正确: 填写表单的原因放在前面, 要做什么放在后面。
- 9 考虑问题的回答可能会从哪里来。
- 8 根据回答的来源把问题分类。
- 7 在可能的时候提供答案, 但要让用户能不考虑页面提供的答案。
- 6 提供适应答案长度的交互区域。

一层 改进感知层 (即好看又简洁)

- 5 使用标题和颜色来把表单中不同的区域进行分组。
- 4 期待用户能以最快的速度开始对问题做出反应。
- 3 不要期望用户在大量图文内获取信息。
- 2 提示要详细, 但要尽可能简短。
- 1 选用易于阅读的字体和颜色。

图2.11 设计友好的表单示例

实战体验 1：设计可用的表单

当我们访问一个网站时，注册、登录、回复、留言、购买等交互行为都需要表单来辅助。毫不夸张地说，表单设计的好坏，直接关系到用户体验、注册成功率、交易成功率和回头率等。当然，表单的设计也非常复杂，也非本书所能够完全讲透的。但是，设计并实现简洁、美观、可用性好、符合 Web 标准的表单，是 Web 设计师必须追求的目标。

要确保表单可用，设计师首先应该从 HTML 结构开始，设计符合语义的表单结构，然后使用 CSS 呈现表单效果。下面这个示例可以帮助读者认识如何设计合理的结构，以及如何设计简洁的表单样式（如图 2.12 所示）。

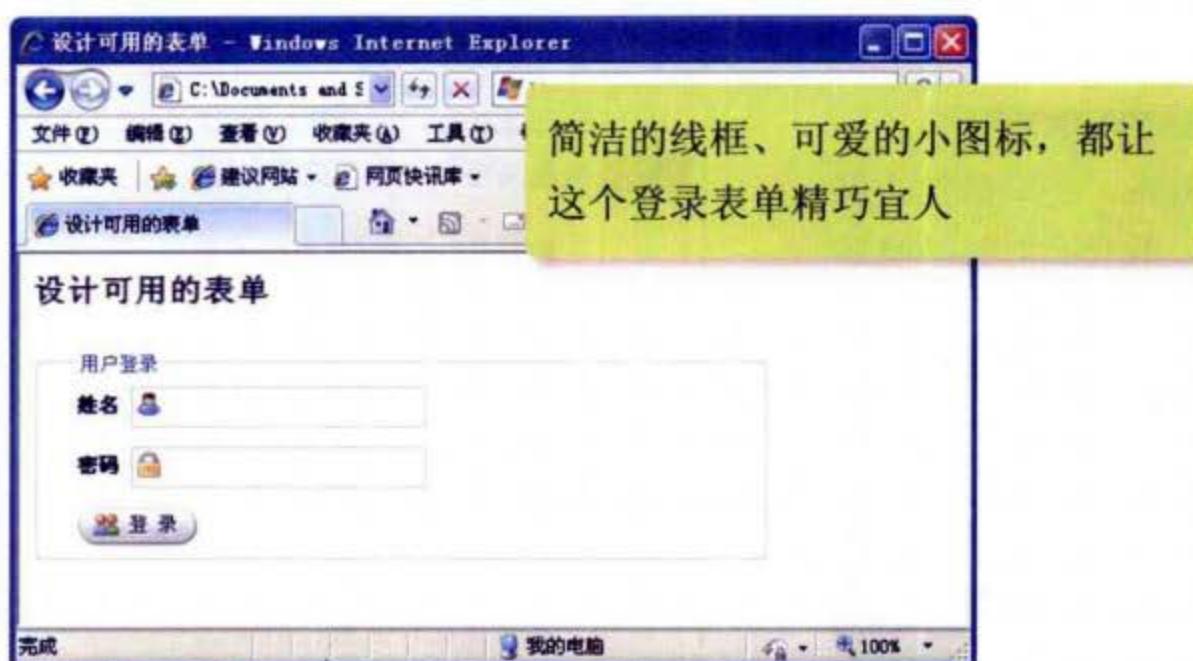


图 2.12 在 IE 8 中的预览效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>设计可用的表单</title>
<style type="text/css">
h1{ font-size:20px; }
#login {
    width:400px;
    padding:1em 2em 0 2em;
    font-size:12px;
}
label { /* 表单项中标签样式 */
    line-height:26px;
    display:block;
    font-weight:bold;
}
#name, #password /* 文本框公共样式 */
    border:1px solid #ccc;
    width:160px;

```

label 是一个行内元素，增加该声明，确保每个表单项独立一行

```

height:22px; /* 固定高度 */
padding-left:20px; /* 挤出位置 */
margin:6px 0;
line-height:20px; /* 让输入文本居中 */
}
#name { background:url(images/name.gif) no-repeat left center; }
#password { background:url(images/password.gif) no-repeat left center; }
.button { margin:6px 0; }
</style>
</head>
<body>
<h1>设计可用的表单 </h1>
<fieldset id="login">
    <legend>用户登录</legend>
    <form action="" method="POST" class="form">
        <label for="name">姓名
            <input name="name" type="text" id="name" value="" />
        </label>
        <label for="password">密码
            <input name="password" type="text" id="password" value="" />
        </label>
        <input type="image" class="button" src="images/login1.gif" />
    </form>
</fieldset>
</body>
</html>

```

设计文本框内左侧的图标需要一点技巧：通过左侧补白挤出一点位置用来显示背景，背景当然是精确定位和静止平铺的，同时还应考虑文本框的高度和输入文本居中等问题

标前缀式 */
标前缀式 */

要设计合理的表单结构，离不开表单辅助元素的配合，其中 **label** 元素应该通过 **for** 属性绑定到表单域上，**for** 的属性值应设置为表单域的 **name** 属性值

当用户登录成功后，不妨通过脚本把文本框设置为不可用（`disabled="disabled"`）状态，这时设计师就可以通过 `E:disabled` 选择器让文本框显示为灰色，以告诉用户该文本框不可用了（如图 2.13 所示）。这样就不用设计“不可用”样式类，并把该类添加到 HTML 结构中。

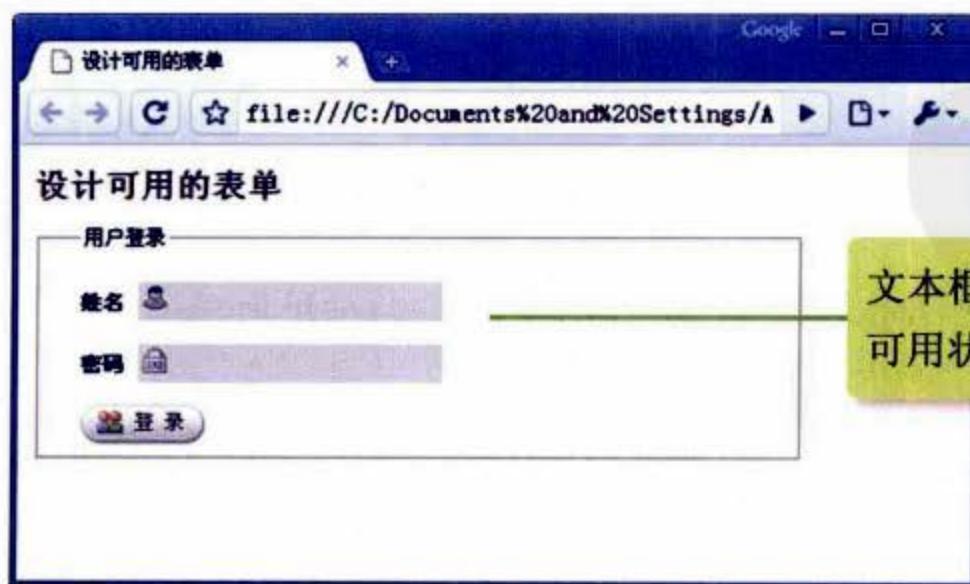


图 2.13 在 Chrome 4 中的预览效果

要实现图 2.13 的效果，则应在上面示例的基础上添加如下样式：

```
#login input:disabled#name { /*姓名文本框处于不可用状态时的样式*/
    background:#ddd url(images/namel.gif) no-repeat 2px 2px;
    border:1px solid #fff;
}
#login input:disabled#password { /*密码域处于不可用状态时的样式*/
    background:#ddd url(images/password1.gif) no-repeat 2px 2px;
    border:1px solid #fff;
}
```

同时在文本框中补加 disabled 属性。

```
<label for="name"> 姓名
    <input name="name" type="text" id="name" value="" disabled="disabled" />
</label>
<label for="password"> 密码
    <input name="password" type="text" id="password" value="" disabled="disabled" />
</label>
```

设计精美的表单能够留住用户的目光，甚至吸引用户登录。如果我们在上面案例的基础上适当进行美化，也许可以得到更美的效果（如图 2.14 所示）。



图 2.14 在 Chrome 4 中的预览效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>设计可用的表单</title>
<style type="text/css">
body { font-family: Helvetica, arial, sans-serif; margin: 0px 0px 0px 0px; font-size:
```

样式代码排列很稠密，但设计思路很简单，故不想占用太多的版面

```

14px; background-color: #ffffff }

/* 设计文本框外框样式 */

div.wrapper { background-image:url(images/bg.gif); background-repeat:no-repeat;
width:348px; height:384px; margin-left:14px; padding-top:75px; }

/* 设计标题样式 */

div.ribbon { background-image:url(images/ribbon.png); background-repeat:no-repeat;
width:358px; height:45px; float:left; margin-top:25px; margin-left:10px; padding-
left:25px; padding-top:5px; color:#ffffff; font-weight:bold; }

/* 设计Logo效果 */

div.logo { background:url(images/logo.png) no-repeat; width:330px; height:115px; }
div.loginwrapper { margin-left:40px; }

/* 设计动态文本框效果 */

span.usertext { color:#478fab; font-weight:bold; }
input.textbox { background:url(images/text_field.png) 0px -25px; width:264px;
height:20px; border:0px; padding-top:5px; padding-left:4px; }
input.textbox:hover { background:url(images/text_field.png) 0px 0px; border:0px;
}

div.bottomwrapper { margin-left:40px; margin-top:50px; }
a:link, a:visited { color:#ffffff; text-decoration:none; }
a:hover { color:#95ddf9; }

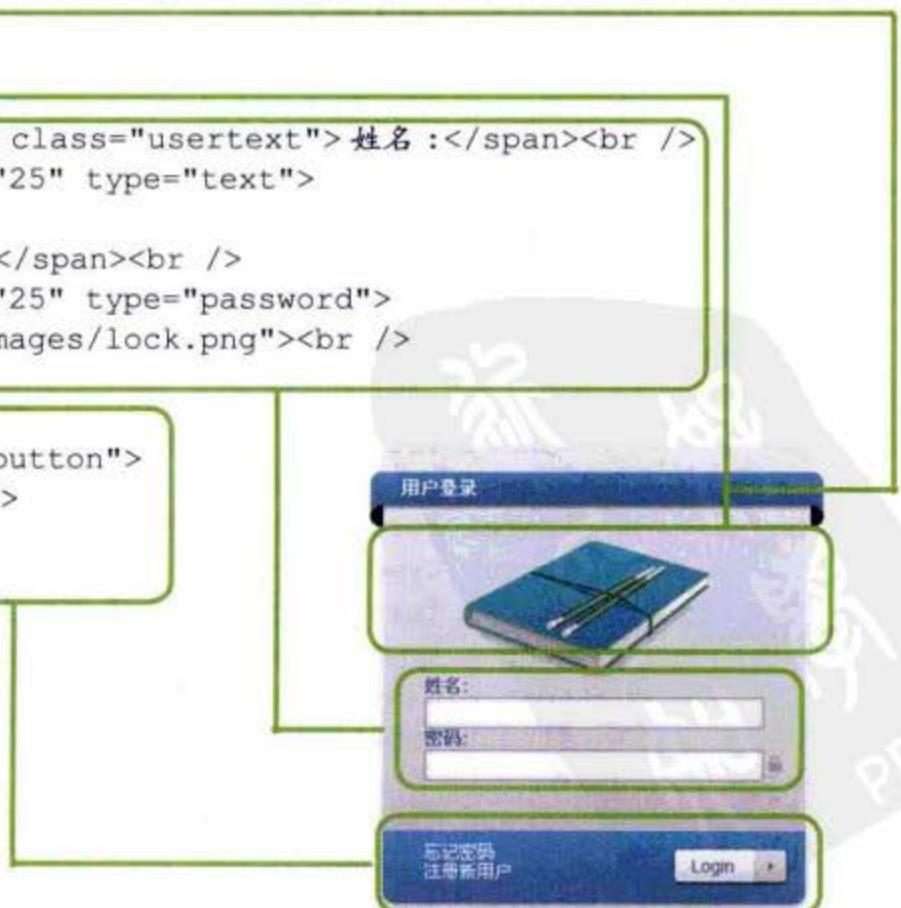
/* 设计动态按钮效果 */

input.button { background:url(images/login_btn.png) 0px 0px; width:92px;
height:31px; border:0px; float:right; margin-right:20px; margin-top:5px; }
input.button:hover { background:url(images/login_btn.png) 0px -31px; }
input.button:active { background:url(images/login_btn.png) 0px -62px; }

</style>
</head>
<body>
<h1>设计可用的表单</h1>
<div class="ribbon">用户登录</div>
<div class="wrapper">
    <div class="logo"></div>
    <div class="loginwrapper"> <span class="usertext">姓名 :</span><br />
        <input class="textbox" size="25" type="text">
        <br />
        <span class="usertext">密码 :</span><br />
        <input class="textbox" size="25" type="password">
        <br />
    </div>
    <div class="bottomwrapper">
        <input class="button" type="button">
        <a href="#">忘记密码</a><br />
        <a href="#">注册新用户</a>
    </div>
</div>
</body>
</html>

```

整个结构分为四块，分别
构建表单的不同区域



实战体验 2：设计友好、易用的表单

制作表单是比较容易的，但是要想设计出友好、易用的表单，设计师就应该考虑很多问题，如编码、结构和样式等。虽然表单设计中的很多细节都微不足道，但对于用户来说却是至关重要的，也是留住用户的关键。为了便于说明问题，我们不妨结合一个案例进行说明。这是一个简单的表单结构，围绕这个结构，我们将探讨不同的设计方式，以方便用户使用。在默认状态下表单显示效果如图 2.15 所示。

图 2.15 一个不带CSS样式的表单

```
<form id="form1" name="form1" method="post" action="">
    <fieldset>
        <legend> 学生信息登记表 </legend>
        <p class="top">
            <label for="textfield" class="title"> 姓名 </label>
            <input type="text" name="textfield" id="textfield" /></p>
        <p>
            <label for="textarea" class="title"> 备注 </label>
            <textarea name="textarea" id="textarea" cols="45" rows="5"></textarea></p>
        <p><span class="title"> 兴趣 </span>
            <label for="checkbox"> 文学 </label>
            <input type="checkbox" name="checkbox" class="checkbox" id="checkbox" />
            <label for="checkbox2"> 艺术 </label>
            <input type="checkbox" name="checkbox2" class="checkbox" id="checkbox2" />
            <label for="checkbox3"> 体育 </label>
            <input type="checkbox" name="checkbox3" class="checkbox" id="checkbox3" />
            <label for="checkbox4"> 其他 </label>
            <input type="checkbox" name="checkbox4" class="checkbox" id="checkbox4" /></p>
        <p><span class="title"> 性别 </span> 女
            <input type="radio" name="radio" id="radio" value="radio" />
            男
            <input type="radio" name="radio" id="radio" value="radio2" /></p>
        <p>
            <label for="select" class="title"> 专业 </label>
```

```

<select name="select" id="select">
    <option value="1"> 法律 </option>
    <option value="2"> 英语 </option>
    <option value="3"> 计算机 </option>
    <option value="4"> 财会 </option>
</select></p>
<p class="center">
    <input type="submit" name="button" id="button" value="提交" />
    <input type="reset" name="button2" id="button2" value="重置" /></p>
</fieldset>
</form>

```

添加如下简单的 CSS 样式后，设计效果如图 2.16 所示。

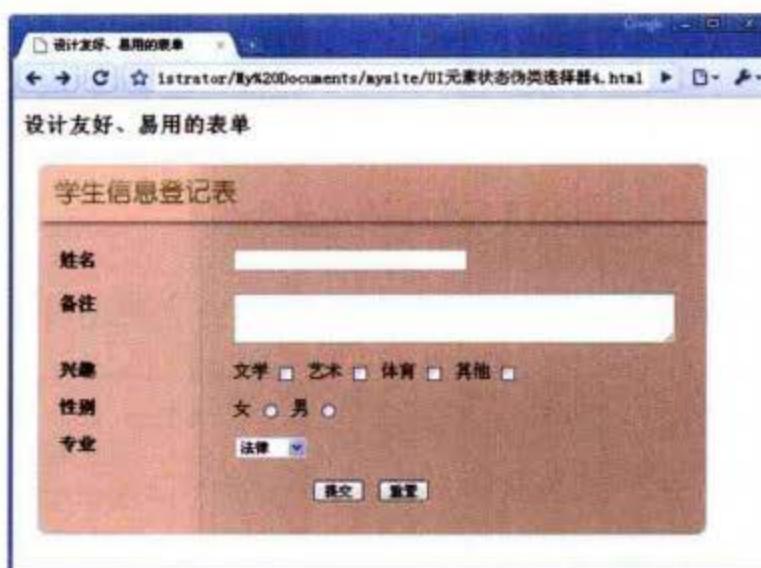


图 2.16 添加CSS样式后的表单效果

```

<style type="text/css">
h1 { font-size:20px; }
fieldset { width:615px; height:346px; background:url(images/bg7.png) no-repeat center; padding:12px; border:none; } /* 表单框样式 */
fieldset legend { display:none; } /* 清除默认的图注，以背景效果显示会更好看 */
/* 以下是各类表单域样式 */
#textfield { width:16em; border:solid 1px #aaa; position:relative; top:-3px; }
#textarea { width:30em; height:3em; border:solid 1px #aaa; }
.checkbox { border:solid 1px #d8bca9; position:relative; top:3px; left:-2px; }
select { border:solid 1px #d8bca9; }
#radio { border:solid 1px #d8bca9; position:relative; top:3px; left:-1px; }
/* 以下是几个辅助设计的样式类 */
.title { width:160px; float:left; font-weight:bold; margin-left:20px; }
.top { margin-top:80px; }
.center { text-align:center; }
</style>

```

好看并不等于好用，图 2.16 所示的表单好用？其实，如果我们稍稍添加一条声明，也许会更好用（如图 2.17 所示）。

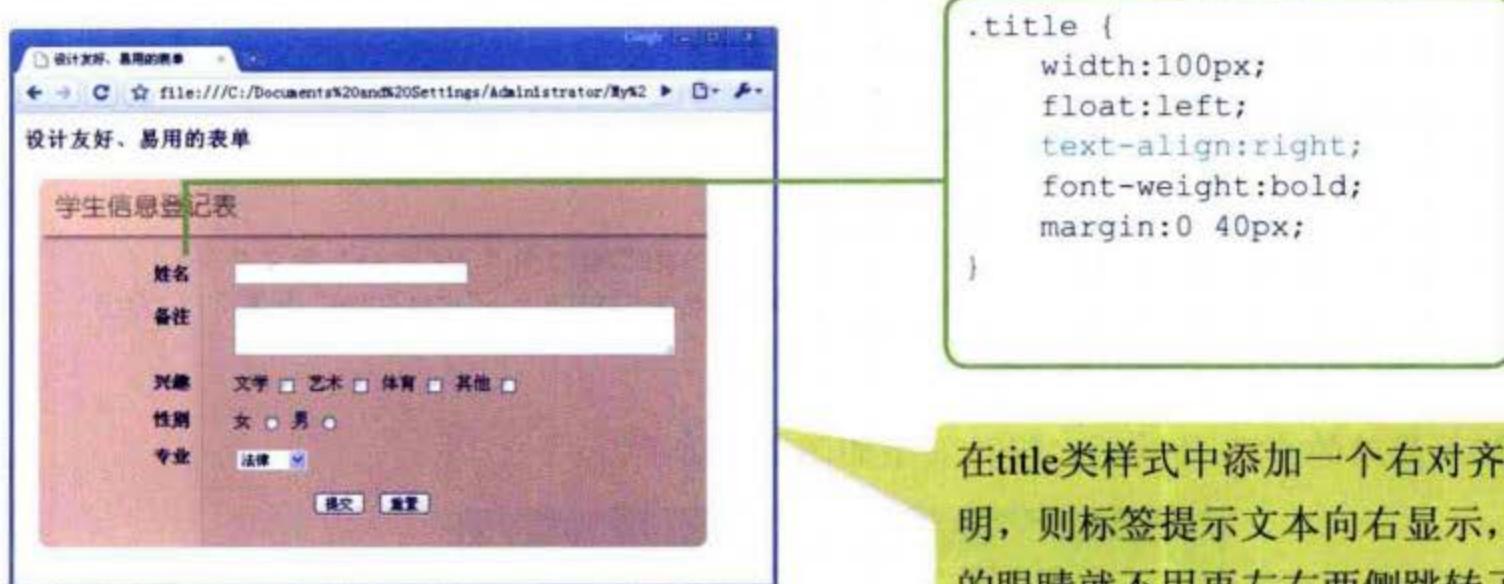


图2.17 改进样式设计效果

对于简单的表单来说，如果避免使用两列布局，也许会更好用（如图 2.18 所示）。

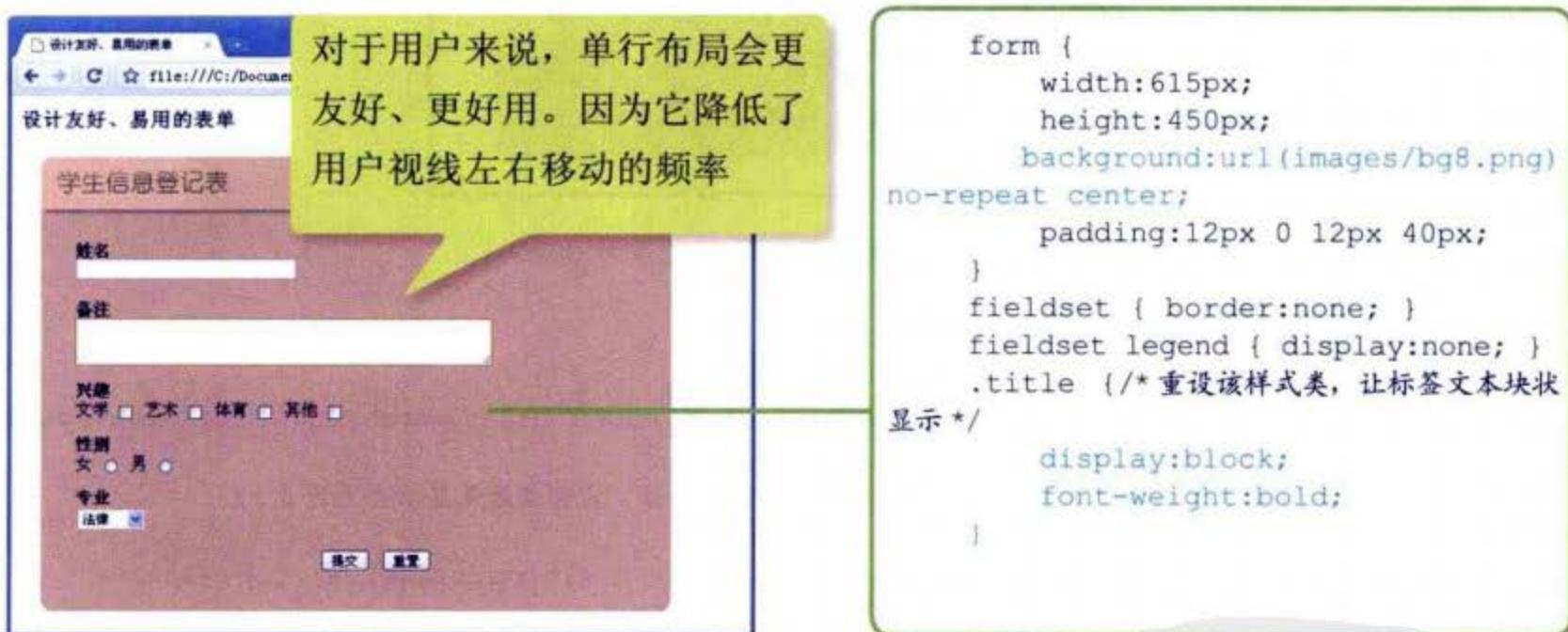
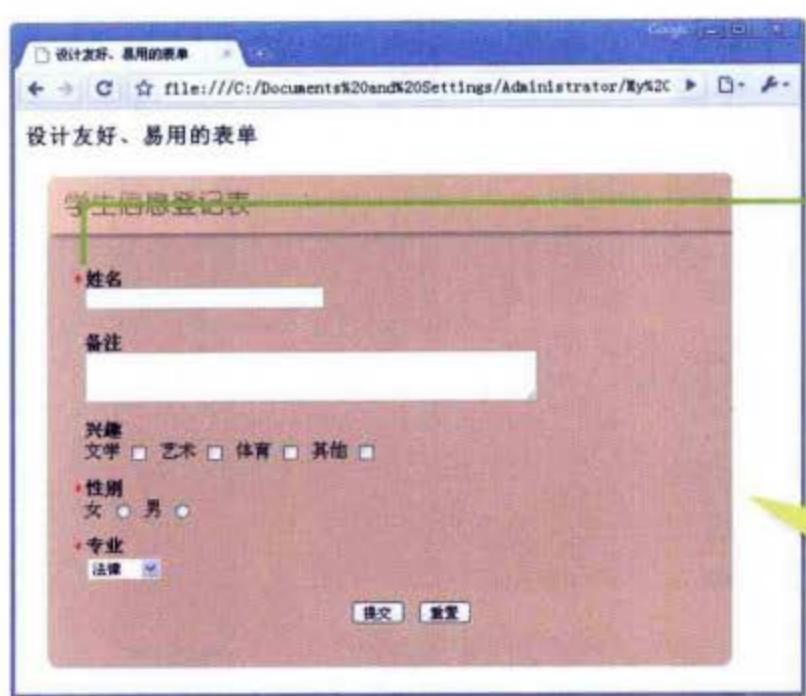


图2.18 继续改进样式设计效果

必填和选填选项应该标识清楚，避免用户摸着石头过河（如图 2.19 所示）。

让验证反馈信息在同一行中显示，这样用户完成表单的速度会更快、成功率会更高（如图 2.20 所示）。

表单选项的排列也很讲究，设计师应该遵循用户的使用习惯。表单项的先后顺序都有约定俗成的习惯，不遵循这个习惯，用户在填写信息时就会感觉不自然。例如，姓名之后一般为性别，而备注一般都会放在最后，所以我们应重新优化一下表单结构的排列顺序，如图 2.21 所示。当表单项目很多时，应该考虑对其进行归类分组，如图 2.22 所示。

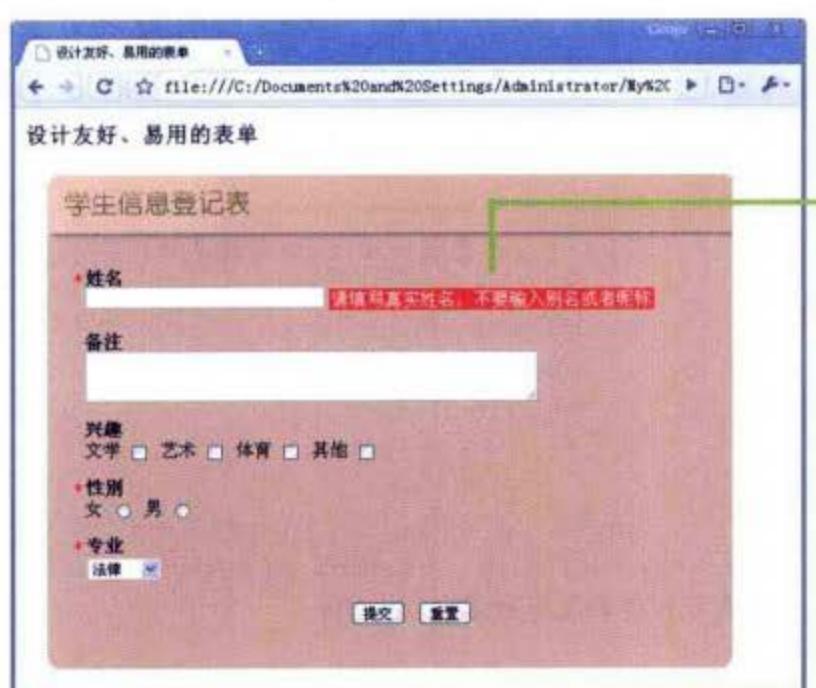


```
.red { /* 定义一个 red 类样式 */
    color:red;
    margin-left:-10px; /* 突出显示 */
    padding-right:2px;
}

<label for="textfield" class="title"><span class="red">*</span> 姓名 </label>
```

只需要添加一个元素和一个
red类样式，就会让用户在填写
时方便很多

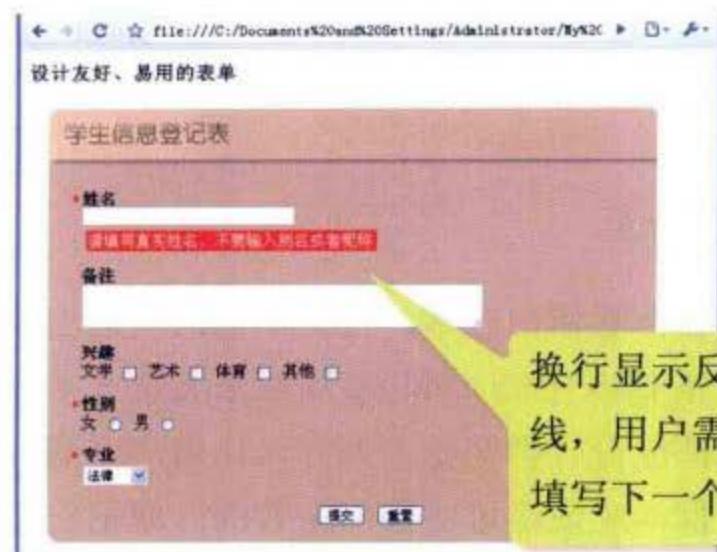
图2.19 添加必填提示标识



```
.error { /* 定义一个 error 类样式 */
    color:white;
    background:red;
    padding:2px;
    margin:0 4px;
}

<input type="text" name="textfield" id="textfield" /><span class="error">请填写真实姓名，不要输入别名或者昵称</span>
```

图2.20 友善的验证反馈信息



设计友善、好用的表单，
很多时候必须借助JavaScript
脚本的配合才行仅仅依靠
CSS的力量是不够的哟。

换行显示反馈信息会影响用户的视
线，用户需要跨越提示信息才能够
填写下一个选项

图2.21展示了表单项排列顺序优化前的界面。所有表单项都是直接堆叠在一起的，缺乏组织。

图2.21 优化表单项的排列顺序

图2.22展示了通过适当分组提升用户体验后的界面。将必填项和选填项分别归类，使用户更容易识别和操作。

图2.22 适当分组用户体验会更好

复杂的表单一般会有很多项目，如果显示一些提示信息，可以让用户填起来更容易。当然，为了避免提示信息的干扰，设计师可以借助 JavaScript 脚本来动态显示提示信息。如图 2.23 所示。

图2.23展示了友好的验证反馈信息。当输入框为空时，显示了一个提示信息：“请填写真实的汉字姓名”，而不是一个错误消息。

```
.tip { /* 定义一个 tip 类样式 */
color:white;
background:blue;
padding:2px;
margin:0 4px;
}

<input type="text" name="textfield"
id="textfield" /><span class="tip">请
填写真实的汉字姓名 </span>
```

图2.23 友善的验证反馈信息

表单设计应该大气，不应捉襟见肘。好用的表单很注意留白艺术，在文本框四周适当地留白，可以让用户操作时很放松，或者也可以通过调整 CSS 中的 padding 属性来实现文本内的留白。同时，文本框的宽度以及文本区域的高度，都应设计得宽松、大方，让用户看到自己输入的文字，尤其是那些用户可能输入较长的文本框，应该尽可能留出足够的区域把它们

都显示出来，这样便于用户检查输入错误，如图 2.24 所示。

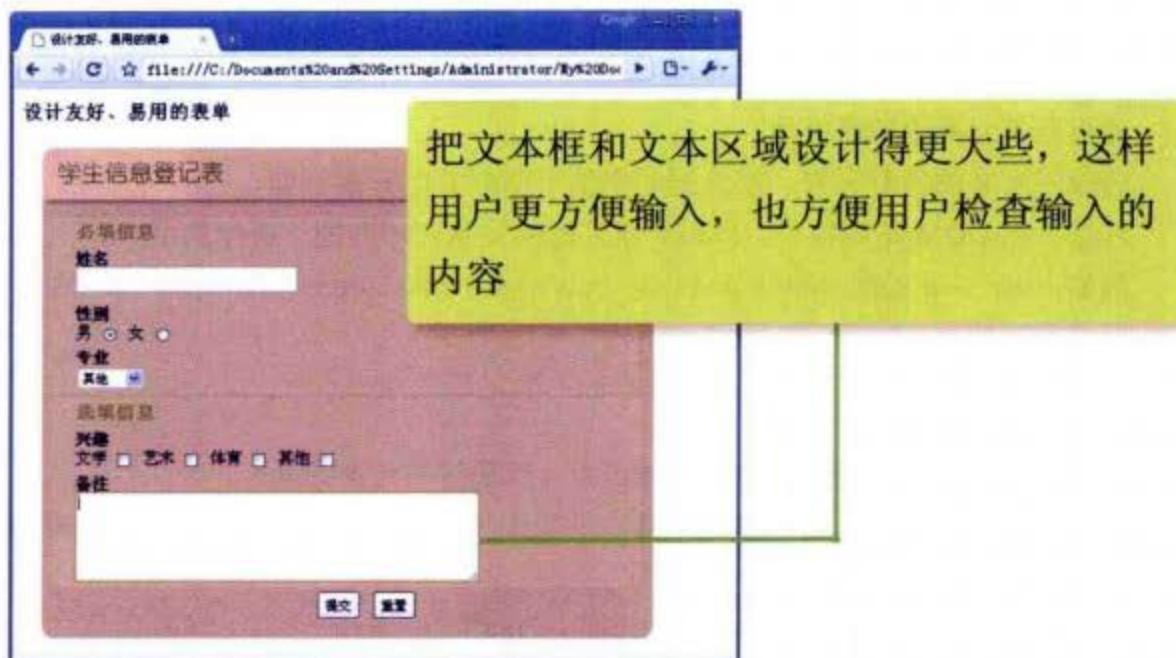


图 2.24 表单设计要大气

另外，当我们在构建表单结构时，还应该时刻注意如下这些表单属性。为表单域设置这些属性，能极大地改善用户体验，让表单变得更好用。

- 1) 使用 value 属性为输入型表单域设置默认值。
- 2) 使用 title 属性帮助屏幕阅读器用户获得足够的提示信息。
- 3) 当标签（label 元素）没有包含所关联的表单域时，可使用 for 属性来匹配和关联表单域（绑定 id 值），这样当用户单击标签时，当前表单域会自动获得焦点。
- 4) 为每个表单域设置 tabindex 属性，方便用户使用 Tab 键切换表单项。
- 5) 对于复选框和单选按钮来说，将标签（label）置于表单域前面，可便于用户先阅读标签说明，然后再执行操作。
- 6) 使用 optgroup 元素来组织 select 列表框中的众多选项，让下拉列表项目一目了然。
- 7) 为表单域定义 accesskey 属性，方便用户使用快捷键快速访问。
- 8) 使用 checked 属性，把常用的复选框和单选按钮定义为默认值，避免用户重复操作。
- 9) 使用 selected 属性，在列表框中把常用选项定义为默认值，同样可以避免用户重复操作。

2.4 其他新增选择器

除了上面介绍的三大类新增的选择器外，本节还要介绍 CSS 3 的其他几个选择器，这些选择器由于无法归类，故放在一起进行说明，如表 2.6 所示。

表 2.6 CSS 3 新增的其他选择器列表

选择器	说明
E ~ F	<p>通用兄弟元素选择器类型。</p> <p>选择匹配 F 的所有元素，且匹配元素位于匹配 E 的元素后面。</p> <p>注意，在 DOM 结构树中，E 和 F 所匹配的元素应该在同一级结构上。</p> <p>例如，div ~ p 匹配 <div><p>1</p></div><div><p>2</p></div><p>3</p> 片段中的 <p>3</p>，但是不匹配 <p>1</p> 和 <p>2</p></p>
E:not(s)	<p>否定伪类选择器类型。</p> <p>选择匹配 E 的所有元素，且过滤掉匹配 s 选择符的任意元素。</p> <p>注意，s 是一个简单结构的选择器，不能使用复合选择器。E:not(s) 选择器相当于二次过滤，适合用于精确地选择元素。</p> <p>例如，div p:not(.red) 匹配 <div><p class="red">1</p></div><div><p>2</p></div><p>3</p> 片段中的 <p>2</p>，但是不匹配 <p class="red">1</p> 和 <p>3</p>。</p> <p>说明：在上面的示例中，浏览器首先使用 div p 包含选择器匹配所有 div 元素包含的 p 元素，此时匹配结果为前两个 p 元素，然后使用 class 选择器 .red 匹配第一个 p 元素，并把它从最终的结果集中过滤掉；最后，返回过滤后的匹配结果</p>
E:target	<p>目标伪类选择器类型。</p> <p>选择匹配 E 的所有元素，且匹配元素被相关 URL 指向。</p> <p>注意，该选择器是动态选择器，只有存在 URL 指向该匹配元素时，样式效果才有效。</p> <p>例如，在下面的文档中（仅包含主体结构和样式），在浏览器地址栏中输入 URL，并附加 "#red"，以锚点方式链接到 <div id="red">，则该元素立即显示为红色背景（如下图所示）</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <pre>CSS : HTML : <style type="text/css"> div:target{ background:red; } </style> <div id="red">盒子 1</div> <div id="blue">盒子 2</div></pre> </div>

浏览器兼容性检测

对于 E ~ F 选择器来说，当前的主流浏览器都完全支持它，大家可以放心使用。但是，E:not(s) 和 E:target 选择器受 IE 浏览器的影响，还不建议推广使用。当 IE 9 普及之时，Web 设计师应该积极使用它们。浏览器兼容检测的详细列表如下。

1. E ~ F

✗ IE 6	✓ Firefox 1.5	✓ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✓ IE 7	✓ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✓ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

2. E:not(s)

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

3. E:target

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验 1：设计层序化的数据表格

当数据需要结构化显示时，设计师可以考虑选用 table（一级）、thead（二级）、tbody（二级）、tfoot（二级）、tr（三级）、th（四级）、td（四级）等表格元素来以结构化的方式组织数据。

但是，如果希望数据表格也能够呈现出层次结构关系，则就应该借助 CSS 来模拟这种结构。

CSS 设计思路

适当完善数据表格的结构，使其更利于树形结构的设计。借助否定伪类选择器和结构伪类选择器，配合 CSS 背景图像技术设计树形结构标志；借助伪类选择器设计鼠标经过时的动态背景效果；利用 CSS 边框和背景色设计标题行的立体显示效果。该案例的设计效果如图 2.25 所示。

排名	校名	总得分	人才培养总得分	研究生培养得分	本科生培养得分	科学研究总得分	自然科学研究得分	社会科学研究得分	所属省份	分省排名	学校类型
1	清华大学	296.77	128.92	93.83	35.09	167.85	148.47	19.38	京	1	理工
2	北京大学	222.02	102.11	86.06	36.03	119.91	86.78	33.13	京	2	综合
一类											
3	浙江大学	205.65	94.67	60.32	34.35	110.97	92.32	18.66	浙	1	综合
4	上海交大	150.98	67.08	47.13	19.95	83.89	77.49	6.41	沪	1	综合
5	南京大学	136.49	62.84	40.21	22.63	73.65	53.87	19.78	苏	1	综合
6	复旦大学	136.36	63.57	40.26	23.31	72.78	51.47	21.31	沪	2	综合
7	华中科大	110.06	54.76	30.26	24.50	55.32	47.45	7.87	鄂	1	理工
8	武汉大学	103.82	50.21	29.37	20.84	53.61	36.17	17.44	鄂	2	综合
9	吉林大学	96.44	48.61	25.74	22.87	47.83	38.13	9.70	吉	1	综合
10	西安交大	92.82	47.22	24.54	22.68	45.60	35.47	10.13	陕	1	综合

图 2.25 在 Chrome 4 中的浏览效果

整个案例的网页源代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>设计层序化的数据表格</title>
<style type="text/css">
body { font-family:"宋体" arial, helvetica, sans-serif; }
table {
    border-collapse: collapse; /* 细线表格必须声明的属性 */
    font-size: 75%;
    line-height: 1.1;
}
tr:hover, td:hover { background: #FF9; } /* 鼠标经过的动态背景色效果 */
th, td {
    padding: .3em .5em;
    cursor:pointer; /* 手指鼠标样式 */
}
th {
```

```

font-weight: normal; /* 清除加粗显示 */
text-align: left;
padding-left: 15px;
}
td:only-of-type {
background: #eee url(images/arrow.gif) no-repeat 12px 50%;
padding-left: 28px;
font-weight: bold;
color: #444;
}
thead th {
background: #c6ceda;
border-color: #fff #fff #888 #fff;
border-style: solid;
border-width: 1px 1px 2px 1px;
padding-left: .5em;
}
tbody th:not(.end) {
background: url(images/dots.gif) 15px 56% no-repeat;
padding-left: 26px;
}
tbody th.end {
background: url(images/dots2.gif) 15px 56% no-repeat;
padding-left: 26px;
}

```

</style>

</head>

<body>

<h1>设计层序化的数据表格 </h1>

<table summary="数据表格信息">

<thead>

<tr>

<th>排名 </th>

<th>校名 </th>

<th>总得分 </th>

<th>人才培养总得分 </th>

<th>研究生培养得分 </th>

<th>本科生培养得分 </th>

<th>科学研究总得分 </th>

<th>自然科学研究得分 </th>

<th>社会科学研究得分 </th>

<th>所属省份 </th>

<th>分省排名 </th>

<th>学校类型 </th>

</tr>

</thead>

<tbody>

<tr>

<td colspan="12">一类 </td>

</tr>

<tr>

<th>1</th>

<td>清华大学 </td>

<td>296.77</td>

<td>128.92</td>

使用结构伪类选择器选择合并单元格所在的行，以背景方式在行前定义指示图标

使用否定伪类选择器选择主体区域中的非最后一个th元素，以背景方式在行前定义结构路径线

使用类选择器选择主体区域中的非最后一个th元素，以背景方式在行前定义结构封闭路径线

使用thead元素把表头标题独立出来，这样对机器更友好，同时也方便CSS控制，避免定义过多的class属性

使用tbody元素把表头标题独立出来，这样对机器更友好，同时也方便CSS控制，避免定义过多的class属性

th元素有两种显示形式，一种用来定义列标题，另一种用来定义行标题

```

<td>93.83</td>
<td>35.09</td>
<td>167.85</td>
<td>148.47</td>
<td>19.38</td>
<td>京 </td>
<td>1 </td>
<td>理工 </td>
</tr>
<tr>
    <th class="end">2</th>                                         定义末尾封口样式类
    <td> 北京大学 </td>
    <td>222.02</td>
    <td>102.11</td>
    <td>66.08</td>
    <td>36.03</td>
    <td>119.91</td>
    <td>86.78</td>
    <td>33.13</td>
    <td>京 </td>
    <td>2 </td>
    <td>综合 </td>
</tr>
<tr>
    <td colspan="12" style="text-align: center;">二类 </td>                                         合并单元格，注意colspan属性的使用
</tr>
<tr>
    <th>3</th>
    <td> 浙江大学 </td>
    <td>205.65</td>
    <td>94.67</td>
    <td>60.32</td>
    <td>34.35</td>
    <td>110.97</td>
    <td>92.32</td>
    <td>18.66</td>
    <td>浙 </td>
    <td>1</td>
    <td>综合 </td>
</tr>
    .....                                         省略大量重复的数据和结构
</tbody>
</table>
</body>
</html>

```

实战体验 2：改善页内导航的视觉体验

通过锚点超链接，用户能够在网页内部自由跳转，以实现页内导航和定位。对于目录结构形式的包含长篇内容的网页来说，设计页内导航是非常重要的。但是，有时候也遇到这样的难题，当用户在页面顶部单击页内导航链接时，却不知道要定位到哪个标题或栏目下，当多个定位项都显示在同一个页面中时，更让人尴尬。

CSS 设计思路

通过 `E:target` 选择器，为页内不同的标题或栏目定义高亮显示的样式，这样当用户定位到该标题或栏目时，会自动高亮显示，以方便用户浏览，效果如图 2.26 所示。



```

<style type="text/css">
body { padding:0; margin:0; }
ul, li { padding:0; margin:0; list-style-type:none; }
h1 { display:none; }

ul /* 定制导航目录外框样式 */
background:url(images/web_bg_1.jpg) no-repeat center;
width:1259px; height:601px;
/* 这个声明很重要：把导航目录框定义为包含块，以方便内部列表项精确定位 */
position:relative;
}

ul a /* 导航目录的超链接样式 */
text-decoration:none;
color:#000;
font-weight:bold; font-family:"黑体"; font-size:20px;
}

ul li { position:absolute; } /* 设置导航目录选项的绝对定位，以便精确控制 */

li:nth-of-type(1) { left:280px; top:200px; }
li:nth-of-type(2) { left:670px; top:200px; }
li:nth-of-type(3) { left:280px; top:300px; }
li:nth-of-type(4) { left:670px; top:300px; }
li:nth-of-type(5) { left:280px; top:400px; }
li:nth-of-type(6) { left:670px; top:400px; }
li:nth-of-type(7) { left:280px; top:500px; }

div /* 将各栏目块的外包装定制为包含块，以便精确定位各个栏目 */
width:1259px;
position:relative;
text-align:center; /* 与下一个样式中的 margin:auto; 配合使用，以实现栏目的居中显示 */
padding-left:117px; /* 以内补白方式微调标题的显示位置 */
}

div h2 {
position:absolute;
width:919px;
margin:auto;
text-align:left;
padding:70px 0 0 100px;
color:#000;
font-weight:bold;
font-family:"黑体";
font-size:20px;
}

/* 下面的各个样式分别使用结构伪类选择器来匹配每个栏目，并精确定位它们的显示位置。 */
h2:nth-of-type(1) {
background:url(images/web_bg_2.jpg) no-repeat center;
height:391px;
top:-50px;
}
h2:nth-of-type(2) {
background:url(images/web_bg_3.jpg) no-repeat center;
height:390px;
top:341px;
}
h2:nth-of-type(3) {

```

清除页边距和列表的默认样式，隐藏主标题

定制导航样式

用结构伪类选择器匹配每个目录项，并精确定位它们的显示位置

把各个栏目的通用样式都汇总到这儿，以优化CSS代码，减少编码的工作量

```

background:url(images/web_bg_4.jpg) no-repeat center;
height:393px;
top:731px;
}
h2:nth-of-type(4) {
background:url(images/web_bg_5.jpg) no-repeat center;
height:392px;
top:1124px;
}
h2:nth-of-type(5) {
background:url(images/web_bg_6.jpg) no-repeat center;
height:392px;
top:1516px;
}
h2:nth-of-type(6) {
background:url(images/web_bg_7.jpg) no-repeat center;
height:390px;
top:1908px;
}
h2:nth-of-type(7) {
background:url(images/web_bg_8.jpg) no-repeat center;
height:432px;
top:2298px;
}
h2:target { color:red; }
</style>
</head>
<body>
<h1>改善页内导航的视觉体验 </h1>
<ul>
<li><a href="#h1">应用程序安装 </a></li>
<li><a href="#h2">音乐下载与管理 </a></li>
<li><a href="#h3">手机视频下载和转码 </a></li>
<li><a href="#h4">联系人、短信和应用程序备份 </a></li>
<li><a href="#h5">在电脑上收发短信 </a></li>
<li><a href="#h6">联系人管理和导入导出 </a></li>
<li><a href="#h7">手机屏幕截图 </a></li>
</ul>
<div>
<h2 id="h1">应用程序安装 </h2>
<h2 id="h2">音乐下载与管理 </h2>
<h2 id="h3">手机视频下载和转码 </h2>
<h2 id="h4">联系人、短信和应用程序备份 </h2>
<h2 id="h5">在电脑上收发短信 </h2>
<h2 id="h6">联系人管理和导入导出 </h2>
<h2 id="h7">手机屏幕截图 </h2>
</div>
</body>
</html>

```

左侧代码看起来很多，但要善于发现规律。当你设计好一个样式后，下面的样式都可以快速地复制。



这个简单的样式是整个案例的点睛之笔，通过它才能够实现我们的设计目的：改善页内导航的视觉体验

构建合理、严谨的结构是设计优雅的CSS代码的基础

第3章

增强的文本和颜色功能

设置文本样式是 CSS 的基本使命。早期的类样式（即 CSS 的雏形）也是基于简化标签在定义文本样式的工作量发展而来，所以 CSS 规范是在类样式的基础上不断丰富和完善的。在 CSS 1 中，W3C 初步制订了文本样式的基本体系（如表 3.1 所示）；在 CSS 2.1 中适当地进行了完善（如表 3.2 所示），同时，CSS 增强了 font 和 vertical-align 属性的功能。

表3.1 CSS 1中定义的字体、颜色和文本属性

属性	类型	说明
font-family	字体	定义字体类型
font-style	字体	定义字体样式。包含值：normal（默认值）、italic（斜体）、oblique（倾斜）
font-variant	字体	定义字体大小写。包含值：normal（默认值）、small-caps（小型的大写字母字体）
font-weight	字体	定义字体粗细
font-size	字体	定义字体大小
font	字体（复合属性）	定义字体样式。注意，可以包含所有字体属性的声明值
color	颜色	定义字体颜色
word-spacing	文本	定义词间距
letter-spacing	文本	定义字符间距
text-decoration	文本	定义文本修饰线。包含值：none（默认值）、blink（闪烁）、underline（下划线）、line-through（贯穿线）、overline（上划线）
vertical-align	文本	定义文本的垂直对齐方式。包含值：auto、baseline、sub、super、top、text-top、middle、bottom、text-bottom、length

(续)

属性	类型	说明
text-transform	文本	定义文本大小写。包含值：none（默认值）、capitalize（单词首字母大写）、uppercase（大写）、lowercase（小写）
text-align	文本	定义文本的水平对齐方式。包含值：left（默认值）、right（右对）、center（居中对齐）、justify（两端对齐）
text-indent	文本	定义文本首行缩进
line-height	文本	定义文本行高

表3.2 CSS 2中新增的字体、颜色和文本属性

属性	类型	说明
font-size-adjust	字体	定义是否强制对文本使用同一尺寸
font-stretch	字体	定义是否横向拉伸变形字体
text-shadow	文本	定义文本阴影效果
direction	文本	文本流入的方向。包含值：ltr（默认值）、rtl（文本从右到左流入）、inherit（文本流入方向由继承获得）
unicode-bidi	文本	定义同一个页面里存在从不同方向读进的文本显示，与 direction 属性一起使用

需要特别提示的是，IE 浏览器私自扩展了很多 CSS 文本属性。例如，text-underline-position（定义下划线位置）、text-overflow（省略文本处理方式，在 CSS 3 中被标准化）、layout-flow（定义文本流动方向）、writing-mode（定义文本书写方向）、word-break（定义字内强制换行）、line-break（定义行内强制换行）、word-wrap（强制文本断开换行，在 CSS 3 中被标准化）、text-autospace（文本空格的自动宽度调节）、text-kashida-space（拉伸字符来调节文本行排列）、text-justify（文本对齐方式）、ruby-align（定义注释或发音的对齐方式）、ruby-overhang（定义注释或发音的位置）、ruby-position（定义注释或发音的位置）、ime-mode（定义输入法）、layout-grid（定义文本字符版式的网格特性）、layout-grid-char（定义文本字符版式的网格尺寸）、layout-grid-line（定义文本字符版式的行网格尺寸）、layout-grid-mode（定义二维网格）、layout-grid-type（定义网格类型）等。当然，这些属性在早期的 CSS 3 版本中都没有被 W3C 组织标准化，在最新的 CSS 3 版本中才开始逐步被接纳，但是如果要普及，还需要各大主流浏览器进一步的努力，读者先了解一下即可。

在文本样式控制方面，CSS 3 加大了革新的力度，新增了几个文本属性（如表 3.3 所示），

同时完善了颜色控制（如表 3.4 所示）功能，实现了对不透明效果的支持。可以说，颜色的不透明度支持是 CSS 3 革新的最大亮点。

表3.3 CSS 3中新增的文本属性

属性	类型	说明
text-shadow	文本	定义文本阴影或模糊效果
text-overflow	文本	定义省略文本的处理方式
word-wrap	文本	定义文本超过指定容器的边界时是否断开转行

表3.4 CSS 3中增强的颜色功能

属性	类型	说明
HSL	颜色表示方式	通过对色调 (H)、饱和度 (S)、亮度 (L) 三个颜色通道的变化以及它们相互之间的叠加来表示各式各样的颜色
HSLA	颜色表示方式	HSLA 是在 HSL 的基础上增加一个透明度 (A) 的设置
RGBA	颜色表示方式	RGBA 是在 RGB 的基础上增加一个透明度 (A) 的设置
opacity	颜色	定义颜色的不透明度。

3.1 文本阴影——text-shadow 属性

实际上，在 CSS 2.1 中，W3C 就已经定义了 text-shadow 属性，但在 CSS 3 中又重新定义了它，并增加了不透明度效果，该属性的基本语法如表 3.5 所示。

表3.5 text-shadow属性的基本语法

语法项目	说明
值	none <length> none [<shadow>,] * <shadow> 或 none <color> [, <color>]*
初始值	无
适用于	所有元素

(续)

语法项目	说明
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <color> 表示颜色；
- <length> 表示由浮点数字和单位标识符组成的长度值，可为负值，指定阴影的水平延伸距离；
- <opacity> 表示由浮点数字和单位标识符组成的长度值，不可为负值，指定模糊效果的作用距离。如果仅仅需要模糊效果，将前两个 length 全部设定为 0 即可。

下面先看一组简单案例。

案例核心代码

```

<style type="text/css">
p {
    text-align: center;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    color: #999;
    font-size: 80px;
    font-weight: bold;
    text-shadow: 0.1em 0.1em #333;
}
</style>

<p>Text Shadow</p>

```



```

<style type="text/css">
p {
    ...
    text-shadow: -0.1em -0.1em #333;
}
</style>

```



```

<style type="text/css">
p {
    ...
    text-shadow: -0.1em 0.1em #333;
}
</style>

```



```
<style type="text/css">
p {
    ...
    text-shadow: 0.1em 0.1em 0.3em #333;
}
</style>
```



```
<style type="text/css">
p {
    ...
    text-shadow: 0.1em 0.1em 0.2em black;
}
</style>
```

简单小结：在上面的示例中，text-shadow属性的第一个值表示水平位移；第二个值表示垂直位移，正值偏右或偏下，负值偏左或偏上；第三个值表示模糊半径，该值可选；第四个值表示阴影的颜色，该值可选。



```
<style type="text/css">
p {
    ...
    color: #fff;
    text-shadow: black 0.1em 0.1em 0.2em;
}
</style>
```

颜色值可以放在前面，实际上它们的位置是不固定的，但几个数值的顺序不能变

通过上面的示例，我们可以得出如下结论：

- 阴影偏移由两个值指定到文本的距离。第一个长度值指定到文本右边的水平距离，负值会把阴影放置在文本的左边。第二个长度值指定到文本下边的垂直距离，负值会把阴影放置在文本上方。
- 在阴影偏移之后，可以指定一个模糊半径。模糊半径是一个长度值，它指出了模糊效果的范围。如何计算模糊效果的具体算法，并没有指定。
- 在阴影效果的长度值之前或之后，还可以指定一个颜色值。颜色值会被用作阴影效果的基础。如果没有指定颜色，那么将使用color属性值来替代。

使用 text-shadow 模拟复杂的文本特效

`text-shadow` 属性可以接受一个以逗号分隔的阴影效果列表，并应用到该元素的文本上。阴影效果按照给定的顺序应用，因此有可能出现互相覆盖，但是它们永远不会覆盖文本本身。阴影效果不会改变边框的尺寸，但可能延伸到它的边界之外。阴影效果的堆叠层次和元素本身的层次是一样的。



```
<style type="text/css">
p {
    ...
    text-shadow: 0.2em 0.5em 0.1em #600,
                -0.3em 0.1em 0.1em #060,
                0.4em -0.3em 0.1em #006;
}
</style>
```



```
<style type="text/css">
p {
    ...
    color: #000;
    text-shadow: 0.5em 0.5em,
                -0.8em 0.8em #060,
                0.7em -0.7em 0.1em;
}
</style>
```

注意，每个阴影效果必须指定阴影偏移值，而模糊半径和阴影颜色是可选参数。

借助阴影效果列表机制，我们可以利用阴影效果设计燃烧的文字效果，如下所示。



你还可以添加更多的阴影列表项，从而实现更复杂的特效

```
<style type="text/css">
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    color: #000;
    background: #000;
    font-size: 80px;
    font-weight: bold;
    text-shadow: 0 0 4px white,
                0 -5px 4px #ff3,
                2px -10px 6px #fd3,
                -2px -15px 11px #f80,
                2px -25px 18px #f20;
}
</style>
```

注意，文本阴影可以使用在 :first-letter 和 :first-line 等伪元素上。

当然，在网页设计中，我们更希望利用该属性设计各种实用文本效果，例如，使用 text-shadow 属性设计立体文本，如下所示。



```
<style type="text/css">
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    color: #D1D1D1;
    background: #CCC;
    font-size: 80px;
    font-weight: bold;
    text-shadow: -1px -1px white,
                 1px 1px #333;
}
</style>
```

通过在文本的左上和右下各添加一个1像素的错位补色阴影，可实现一种淡淡的立体效果

反向思维，利用上面示例的设计思路，我们也可以设计出一种凹体效果，其设计方法是：把上面示例中的左上和右下的阴影的颜色颠倒即可，如下所示。



```
<style type="text/css">
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    color: #D1D1D1;
    background: #CCC;
    font-size: 80px;
    font-weight: bold;
    text-shadow: 1px 1px white,
                 -1px -1px #444;
}
</style>
```

在上面示例的基础上反向设计阴影颜色即可

当然，我们还可以使用 text-shadow 属性为文本描边，其设计方法为：分别为文本的 4 个边添加 1 像素的实体阴影，如下所示。



分别在文本的4个边上添加1像素的黑色阴影

```
<style type="text/css">
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    color: #D1D1D1;
    background: #CCC;
    font-size: 80px;
    font-weight: bold;
    text-shadow: -1px 0 black,
                 0 1px black,
                 1px 0 black,
                 0 -1px black;
}
</style>
```

还可以设计出文本外发光特效，如下所示。



设计阴影不发生位移，同时定义阴影模糊显示即可。

当然，使用一个阴影或者一组阴影都可以

```
<style type="text/css">
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    font-size: 80px;
    font-weight: bold;
    text-shadow: 0 0 0.2em #F87,
                 0 0 0.2em #F87;
}
</style>
```

浏览器兼容性检测

`text-shadow` 属性虽然已经在 CSS 2 中被定义，但是各大主流浏览器却迟迟不支持它，根本原因就在于，浏览器渲染该样式需要耗费更多的资源，在资源有限的 Web 时代初期，这种做法显然是很奢侈的。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计超酷的黑猫警长首页

在企业或者个人的首页中，设计师喜欢使用图形化的首页引导浏览者的视线，富有冲击力的画面、极少的文字说明，都能够让浏览者有一种点击的冲动。当然，这类首页设计大多借助图像来实现的。由于 CSS 缺乏艺术设计能力，设计师前期只能借助 PhotoShop 实现类似的设计。不过，CSS 3 新增加的艺术设计功能，重新燃起了设计师使用代码征服艺术的热情。

CSS 设计思路

借助 text-shadow 属性设计阴影效果，通过颜色的搭配，设计出一种静谧而又神秘的画面，使用两幅 PNG 图像对页面效果进行装饰和点缀，最后的效果如图 3.1 所示。



图 3.1 在 Firefox 3.6 中的演示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>text-shadow</title>
<style type="text/css">
body {
    padding: 0px;
    margin: 0px;
    background: black;
    color: #666;
}

```

定义页面背景色为黑色、前景色为灰色，设计主色调，并清除页边距

```

#text-shadow-box /* 设计盒子外框样式 */
position: relative; /* 让内部的定位元素以这个框为参照物 */
width: 598px;
height: 406px;
background: #666;
overflow: hidden; /* 禁止内容超过设定的区域 */
border: #333 1px solid;
}
#text-shadow-box div.wall /* 设计背景墙样式 */
position: absolute;
width: 100%;
top: 175px;
left: 0px
}
#text /* 设计导航文本样式 */
text-align: center;
line-height: 0.5em;
margin: 0px;
font-family: helvetica, arial, sans-serif;
height: 1px;
color: #999;
font-size: 80px;
font-weight: bold;
text-shadow: 5px -5px 16px #000;
}
div.wall div /* 设计前面挡风板样式 */
position: absolute;
width: 100%;
height: 300px;
top: 42px;
left: 0px;
background: #999;
}
#spotlight /* 设计覆盖在上面的探照灯效果图 */
position: absolute;
width: 100%;
height: 100%;
top: 0px;
left: 0px;
background: url(images/spotlight.png) center -300px;
}
#spotlight a {
color:#ccc;
text-decoration:none;
position:absolute;
left:47%;
top:56%;
float:left;
}
a img { border:none; }
</style>
</head>

<body>
<div id="text-shadow-box">

```

设计右上偏移的阴影，适当进行模糊处理，产生色晕效果，阴影色为深色，营造静谧的效果

设计一个层，让其覆盖在页面上，并使其满窗口显示，通过前期设计好的一个探照灯背景来营造神秘效果

本案例的结构外套

```

<div class="wall">
    <p id="text">黑猫警长</p>
    <div></div>
</div>
<div id="spotlight"><a href="index.htm"></a></div>
</div>
</body>
</html>

```

墙体外结构
外罩，通过它可以为页面覆盖一层桌纸，添加特殊的艺术效果

3.2 溢出文本省略——text-overflow 属性

在信息列表结构中，常常会遇到栏目的宽度固定但信息项的字符过长的矛盾，为了避免超长字符的信息项破坏栏目的布局，设计师经常需要考虑限制栏目信息的显示长度，如设置当信息项字符超出长度时则省略显示。之前，一般多借助 JavaScript 脚本来实现这种效果（如图 3.2 所示）。现在，CSS 3 新增了 text-overflow 属性，使得这个问题迎刃而解。



图 3.2 被省略的标题列表项

text-overflow 属性的基本语法如表 3.6 所示。

表 3.6 text-overflow 属性基本语法

语法项目	说明
值	clip ellipsis ellipsis-word
初始值	无
适用于	块状元素或行内元素

(续)

语法项目	说明
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- clip 属性值表示不显示省略标记，而是简单地裁切。
- ellipsis 属性值表示当对象内文本溢出时显示省略标记，省略标记插入的位置是最后一个字符。
- ellipsis-word 表示当对象内文本溢出时显示省略标记，省略标记插入的位置是最后一个词（word）。

实际上，text-overflow 属性仅用于决定，当文本溢出时是否显示省略标记，并不具备样式定义的功能。要实现溢出时产生省略号的效果，应该再定义两个样式：强制文本在一行内显示（white-space:nowrap）和溢出内容为隐藏（overflow:hidden），只有这样才能实现溢出文本显示为省略号的效果。

浏览器兼容性检测

text-overflow 属性比较特殊，当设置的值不同时，浏览器对其支持也不同，下面分别进行说明。

1. text-overflow:clip

				
✓ IE 6	✗ Firefox 1.5	✓ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✓ IE 7	✗ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✗ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

2. text-overflow:ellipsis

				
✓ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✓ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✗ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

在早期 W3C 文档[⊖]中，text-overflow 已被纳入规范，但是在最新修订的文档[⊖]中没有再包含 text-overflow 属性。

由于 W3C 规范放弃了对 text-overflow 属性的支持，所以 Mozilla 类型的浏览器也放弃了对该属性的支持。不过，Mozilla developer center 推荐使用 -moz-binding 的 CSS 属性进行兼容。Firefox 支持 XUL（XUL 是一种 XML 的用户界面语言，也是一种 XML 绑定语言），这样我们就可以使用 -moz-binding 属性来绑定 XUL 里的 ellipsis 属性了。

实战体验：设计固定区域的新闻列表

在这个案例中，我们将使用 text-overflow 属性来实现在固定的版块中，强迫新闻列表有序显示，对于超出指定宽度的新闻项，则通过省略并附加省略号来避免新闻换行或者撑开版面，效果如图 3.3 所示。

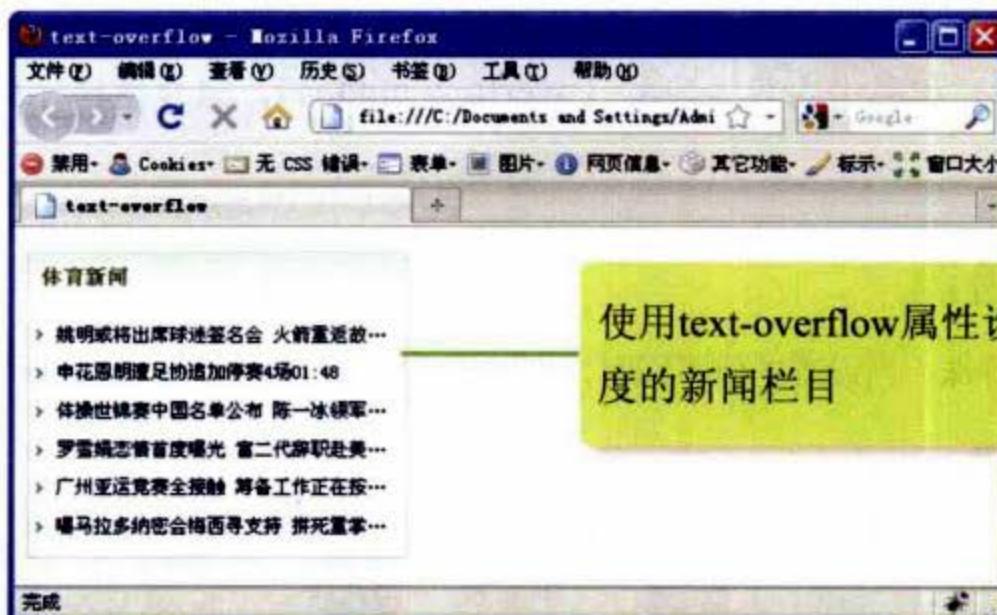


图 3.3 在 Firefox 3.6 中的演示效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>text-overflow</title>
<style type="text/css">
dl { /* 固定新闻栏目外框 */
    width:240px;
    border:solid 1px #ccc;
}
dt { /* 设计新闻栏目标题行 */
    padding:8px 8px;
    background:#7FECAD url(images/green.gif) repeat-x;
}

```

[⊖] <http://www.w3.org/TR/2003/CR-css3-text-20030514/#text-overflow-mode>

[⊖] <http://www.w3.org/TR/css3-text/>

```

font-size:13px;
text-align:left;
font-weight:bold;
color:#71790C;
margin-bottom:12px;
border-bottom:solid 1px #efefef;
}
dd { /* 设计新闻列表项样式 */
font-size:0.78em;
height:1.5em;
width:220px;
padding:2px 2px 2px 18px; /* 为添加新闻项目符号腾出空间 */
background:url(images/icon.gif) no-repeat left 25%; /* 以背景方式添加项目符号 */
margin:2px 0;
white-space: nowrap; /* 为应用 text-overflow 作准备，禁止换行 */
overflow: hidden; /* 为应用 text-overflow 作准备，禁止文本溢出显示 */
-o-text-overflow: ellipsis; /* 兼容 Opera */
text-overflow: ellipsis; /* 兼容 IE, Safari (Webkit) */
-moz-binding: url('ellipsis.xml#ellipsis'); /* 兼容 Firefox */
}
</style>
</head>

<body>
<dl>
<dt>体育新闻</dt>
<dd>姚明或将出席球迷签名会 火箭重返故地拉拢球迷 10:58</dd>
<dd>申花恩朗遭足协追加停赛 4场 01:48 </dd>
<dd>体操世锦赛中国名单公布 陈一冰领军邹凯无缘出征 10:52</dd>
<dd>罗雪娟恋情首度曝光 富二代辞职赴美陪读 10:36 </dd>
<dd>广州亚运竞赛全接触 筹备工作正在按部就班进行 09:53 </dd>
<dd>曝马拉多纳密会梅西寻支持 拼死重掌阿根廷教鞭 10:25</dd>
</dl>
</body>
</html>

```

3.3 文本换行显示——word-wrap 属性

在上面的案例实战中，我们用到了 white-space 属性，这个属性出现在 CSS 1 中。为了增强文本换行显示的功能，CSS 3 吸纳了 IE 定义的 word-wrap 专有属性，并对其进行了标准化。word-wrap 属性的基本语法如表 3.7 所示。

表3.7 word-wrap属性的基本语法

语法项目	说明
值	normal break-word
初始值	normal

(续)

语法项目	说明
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

取值简单说明：

- normal 属性值表示控制连续文本换行。
- break-word 属性值表示内容将在边界内换行。如果需要，词内换行（word-break）也会发生。

换行技术比较分析

简单回顾一下，IE 定义了多个换行处理属性：line-break、word-break、word-wrap，另外 CSS1 定义了 white-space，CSS 3 增加了 word-wrap。

- line-break 专门负责控制日文换行，国内的设计师接触得比较少。
- word-wrap 属性可以控制换行。当属性取值 break-word 时，将强制换行，中文文本没有任何问题，英文语句也没问题。但是对于长串的英文就不起作用，也就是说，word-wrap:break-word 是控制是否断词，而不是断字符。
- word-break 属性主要针对亚洲语言和非亚洲语言进行控制换行。当属性取值 break-all 时，可以允许非亚洲语言文本行的任意字内断开；当属性值为 keep-all 时，表示在中文、韩文、日文中是不允许字断开的。
- white-space 属性具有格式化文本的作用，当属性取值为 nowrap 时，表示强制在同一行内显示所有文本；当属性值为 pre 时，表示显示预定义文本格式。

在 IE 浏览器下，使用 word-wrap:break-word; 声明可以确保所有文本正常显示。

在 Firefox 浏览器下，中文不会出现任何问题，英文语句也不会出现问题，但是长串英文会出现问题。为了解决长串英文的问题，一般将 word-wrap:break-word; 和 word-break:break-all; 声明结合使用。但是，这种方法会导致普通英文语句中的单词被断开显示（在 IE 下也是）。

现在的主要问题是长串英文和英文单词会被断开。其实长串英文就是一个比较长的单词，这也就被归结为英文单词被断开的问题。

为了解决这个问题，可使用 word-wrap:break-word;overflow:hidden;，而不是 word-wrap : break-word;word-break:break-all;。word-wrap:break-word;overflow:auto; 在 IE 下没有任何问题，但是在 Firefox 下，长串英文单词的部分内容就会被遮住。下面将用一个示例来进行演示比较。



```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
}
</style>
```

```
<p>This is all English. This is all English.</p>
<p>全中文的情况。全中文的情况。全中文的情况。</p>
<p>中英文混排的情况。Chinese and English. 中英文混排的情况。Chinese and English.</p>
```

```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
    word-wrap:break-word;
}
</style>
```

强制文本换行显示，以及换行处理方法。不同浏览器显示效果相同

```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
    word-wrap:break-word;
    word-break:break-all;
}
</style>
```

强制文本换行显示，添加
word-break:break-all;声明。
在IE 8下浏览



```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
    word-wrap:break-word;
    word-break:break-all;
}
</style>
```

强制文本换行显示，添加
word-break:break-all;声明。
在Firefox3.6下浏览

```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
    word-wrap:break-word;
    word-break:keep-all;
}
</style>
```

强制文本换行显示，添加
word-break:keep-all;声明。在
IE 8下浏览

```
<style type="text/css">
p{
    width:250px;
    border:1px solid red;
    word-wrap:break-word;
    word-break:keep-all;
}
</style>
```

强制文本换行显示，添加
word-break:keep-all;声明。在
Firefox 3.6下浏览

浏览器兼容性检测

word-wrap 属性尚未被广泛支持，特别是 Firefox 和 Opera 对其支持不好。这是因为，在早期的 W3C 文本模型中[⊖]放弃了对它的支持，并定义了 wrap-option 属性代替，但是在最新的文本模式中[⊖]又继续支持该属性，而且重新定义了属性值。



				
✓ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✓ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：防止表格标题行换行

在上一章中，我们曾经讲解了一个设计优雅表格的案例。在这个案例中，当表格宽度发生变化时，标题行可能会被撑开（如图 3.4 所示），会影响浏览体验。解决这个问题方法有很多种，可以固定表格的宽度，或者通过下面的方法来进行设计。



图 3.4 被撑开的表格标题

该方法的具体操作是为 th 元素添加 nowrap 属性，同时借助 CSS 换行技术进行处理，效果如图 3.5 所示。

⊖ <http://www.w3.org/TR/2003/CR-css3-text-20030514/>

⊖ <http://www.w3.org/TR/css3-text/>



图3.5 避免被撑开的表格标题

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>word-wrap</title>
<style type="text/css">
h1 { font-size:16px; }
table {
    width:100%;
    font-size:12px;
    empty-cells:show;
    border-collapse:collapse;
    border-collapse: collapse;
    margin:0 auto;
    border:1px solid #cad9ea;
    color:#666;
    table-layout: fixed;           /* 定义表格在浏览器端逐步解析和逐步呈现 */
    word-break:keep-all;          /* 禁止词断开显示 */
    word-wrap:normal;             /* 允许内容顶开指定的容器边界，如果声明了word-wrap:break-word；，则会在IE浏览器中出现换行显示，会破坏整个标题行的样式 */
    white-space: nowrap;          /* 强迫在一行内显示 */
}
...
</style>
</head>
<body>
<h1>避免表格标题行换行显示</h1>
<table summary="设计优雅的数据表格">
    <tr>
        <th nowrap="nowrap">排名</th>
        <th nowrap="nowrap">校名</th>
        <th nowrap="nowrap">总得分</th>
        <th nowrap="nowrap">人才培养总得分</th>
        <th nowrap="nowrap">研究生培养得分</th>
        <th nowrap="nowrap">本科生培养得分</th>
        <th nowrap="nowrap">科学研究总得分</th>
        <th nowrap="nowrap">自然科学研究得分</th>
        <th nowrap="nowrap">社会科学研究得分</th>
        <th nowrap="nowrap">所属省份</th>
        <th nowrap="nowrap">分省排名</th>
        <th nowrap="nowrap">学校类型</th>
    
```

省略了其他CSS样式代码

通过手工添加这样一行属性，确保在不同浏览器中都能够很好地单行显示。注意，如果th元素定义了宽度，该属性将不再起作用。

```

</tr>
...
</table>
</body>
</html>

```

省略了其他的tr结构代码

3.4 CSS 3 文本模块解析

从 CSS 3 开始，W3C 提出了文本模块（Text Module）的概念，它单独为与文本相关的属性制订了规范，目的是形成一个独立的标准体系。文本模块的最早版本是在 2003 年制定[⊖]的，2005 年对其进行了修订[⊖]，2007 年又进行了系统的更新[⊖]，最后形成了一个较为完善的文本模型[⊖]。在最终版本的文本模块中，除了新增了文本属性外，还对 CSS 2.1 中已定义的属性取值进行了补充和修订，增加了更多的属性值，以适应复杂环境中文本的呈现。为了方便读者参考和学习，下面将做一些简单的介绍和描述。

1. white-space-collapse

语法项目	说明
值	preserve collapse preserve-breaks discard
初始值	collapse
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索如何处理对象内包含的空格字符，对应 CSS 2.1 中的 white-space 属性。取值简单说明如下：

- collapse：使用一个单一的字符序列呈现空白（或在某些情况下，没有字符）。
- preserve：可以呈现所有空白，换行符将被保留。
- preserve-breaks：抛弃呈现所有空白，但保留换行符。
- discard：抛弃呈现所有空白。

⊖ <http://www.w3.org/TR/2003/CR-css3-text-20030514/>
 ⊖ <http://www.w3.org/TR/2005/WD-css3-text-20050627/>
 ⊖ <http://www.w3.org/TR/2007/WD-css3-text-20070306/>
 ⊖ <http://www.w3.org/TR/css3-text/>

2 white-space

语法项目	说明
值	normal pre nowrap pre-wrap pre-line
初始值	无
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索对象内空格字符的处理方式。它是 white-space-collapse 和 text-wrap 属性的简便用法，并没有包含 white-space-collapse 和 text-wrap 属性的所有功能。与 CSS 2.1 相比，新增了两个属性值。其取值简单说明如下：

- normal 类似于 white-space-collapse:collapse;text-wrap:normal;。
- pre 类似于 white-space-collapse:preserve;text-wrap:none;。
- nowrap 类似于 white-space-collapse:collapse;text-wrap:none;。
- pre-wrap 类似于 white-space-collapse:preserve;text-wrap:normal;。
- pre-line 类似于 white-space-collapse:preserve-breaks;text-wrap:normal;。

3 word-break

语法项目	说明
值	normal keep-all loose break-strict break-all
初始值	normal
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索对象内文本的字内换行行为，在出现多种语言的情况下尤为有用。对于中文，应该使用 break-all。其取值简单说明如下：

- normal 根据语言自己的规则确定换行方式。
- keep-all 同 normal，对于中、日、韩字符而言，不允许字断开。
- loose 类似于 normal，但是允许中、日、韩字符在任意位置断开。
- break-strict 类似于 normal，但是允许非中文字符和日韩字符在任意位置断开。
- break-all 类似于 break-strict，除了中、日、韩以外的字符应遵循 loose 的规则。

4.text-wrap

语法项目	说明
值	normal unrestricted none suppress
初始值	normal
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

CSS 3 定义文本换行是通过 text-wrap 和 word-wrap 这两个属性来控制的。text-wrap 属性用于设置或检索对象内文本的换行模式。其取值简单说明如下：

- normal：自动换行模式。
- none：不换行模式。
- unrestricted：无限制模式。
- suppress：压制模式。

5.word-wrap

该属性专门用于处理字符换行问题，如果当前行超过指定容器的边界，它用于设置或检索是否断开换行，具体内容在前面已经讲过。

6.text-align

语法项目	说明
值	start end left right center justify <string>
初始值	start
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索对象中文本的对齐方式。与 CSS 2.1 相比，CSS 3 增加了 start、end 和 <string> 属性值。其中，start 和 end 属性值主要是针对行内元素的，即在包含元素的头部或尾部显示；而 <string> 属性值主要应用于表格单元格中，将根据某个指定的字符进行对齐。

7.text-align-last

语法项目	说明
值	start end left right center justify
初始值	start
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索对象中最后一行文本的对齐方式，主要针对 text-align 设置为 justify 值时，可以强制换行的文本进行对齐。

8.text-justify

语法项目	说明
值	auto inter-word inter-ideograph inter-cluster distribute kashida tibetan
初始值	auto
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于设置或检索对象内调整文本使用的对齐方式。只有当 text-align 设置为 justify 时，设置该属性才有效。CSS 3 最新版本采纳了 IE 的私有属性 text-justify，但是重新规划了取值。其取值简单说明如下：

- auto：允许浏览器（或者代理用户）使用两端对齐方式。
- inter-word：通过增加字之间的空格对齐文本。该行为是对齐所有文本行的最快方法。它的两端对齐行为对段落的最后一行无效。
- inter-ideograph：针对表意字符设置文本两端完全对齐。为了实现完全对齐，CSS 会强迫浏览器增加或减少字符或词间的空格。
- inter-cluster：调整文本或词间的空格。这种模式的调整用于优化亚洲语言的文档。
- distribute：通过增加或减少字或字母之间的空格来对齐文本，是用于拉丁文字母表两端对齐的最精确格式，适用于东亚文档，尤其是泰文。
- kashida：通过拉长选定点的字符来调整文本。这种调整模式是特别为阿拉伯脚本语言提供的。
- tibetan：两端对齐行的方式与 distribute 相同，也同样不包含两端对齐段落的最后一行，适用于表意字文档。该值可能会在未来的修订中被删除。

9.word-spacing

语法项目	说明
值	normal <length> <percentage>
初始值	normal
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置对象中的单词之间插入的空格。其中，percentage 表示根据空格字符（U+0020）的宽度进行计算。单词间距会受对齐调整的影响。

10.letter-spacing

语法项目	说明
值	normal <length> <percentage>
初始值	normal
适用于	所有元素
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置对象中字符之间的间隔。该属性将指定的间隔添加到每个文字之后，但最后一个字将被排除在外，字符间距会受对齐调整的影响。

11.punctuation-trim

语法项目	说明
值	none [start end adjacent]
初始值	none
适用于	所有元素及其内容
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置标点符号的修剪。其取值简单说明如下：

- none：不修剪。
- start：根据开始位置的标点符号修剪另一半标点符号。

- end** : 根据结束位置的标点符号修剪另一半标点符号。
- adjacent** : 根据相邻位置的标点符号修剪另一半标点符号。

12. text-emphasis

语法项目	说明
值	none [[accent dot circle disc] [before after]?]
初始值	none
适用于	所有元素及其内容
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置重点文本样式。取值简单说明如下：

- none** : 没有重点标记。
- accent** : 马克笔画标记。
- dot** : 点标记。
- circle** : 空心圆标记。
- disc** : 实心圆标记。
- before** : 在顶部标记, 或者在右侧标记 (针对垂直书写的文本)。
- after** : 在文本底部标记, 或者在左侧标记 (针对垂直书写的文本)。

13. text-shadow

该属性用于检索或设置文本阴影, 详细说明请参阅前面小节的讲解。

14. text-outline

语法项目	说明
值	none [<color> <length> <length>? <length> <length>? <color>]
初始值	none
适用于	所有元素及其内容
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置文本的外形轮廓。其中, 第一个长度值表示轮廓的厚度, 第二个长度值是可选的, 表示模糊半径。轮廓不会覆盖文本本身。

15.text-indent

语法项目	说明
值	[<length> <percentage>] hanging?
初始值	0
适用于	块状元素、行内块状元素或者表格单元格
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置对象中的文本的缩进。其中，<percentage> 表示根据包含元素的宽度进行计算。

16.hanging-punctuation

语法项目	说明
值	none [start end end-edge]
初始值	none
适用于	块状元素、行内块状元素或者表格单元格
可否继承	有
百分比	N/A
媒介	视觉

该属性用于检索或设置对象是否悬挂一个标点符号。其取值简单说明如下：

- start：标点符号可以挂在第一行开始处的边缘。
- end：标点符号可以挂在最后一行末尾处的边缘。
- end-edge：标点可能会挂起所有行结束边缘的外面。

3.5 CSS 3 不同版本之间的文本规范的差异

CSS 3 的规范从起草到定型经历了漫长的演化过程，先后一共制订了三个主要版本的工作草案，最新版本的文本模型（<http://www.w3.org/TR/css3-text/>）与 2003 年版本（<http://www.w3.org/TR/2003/CR-css3-text-20030514/>）相比，进行了较大的改动，其中主要改动说明如下：

- line-break 和 word-break-cjk 属性被 word-break 属性替换。
- word-break-inside 属性被 hyphenate 属性替换。
- wrap-option 属性被 text-wrap 和 word-break 属性替换。

- linefeed-treatment、white-space-treatment 和 all-space-treatment 属性被 white-space-collapse 属性替换。
- min-font-size 和 max-font-size 属性被移至下一个 CSS 3 版本的字体模块内。
- 修改了 text-align 属性中 left 和 right 属性值在垂直文本中的行为。
- text-align-last 属性取消了 size 属性值。
- text-justify 属性取消了 newspaper 属性值。
- word-spacing 和 letter-spacing 属性增加了百分比取值。
- text-wrap 属性增加了 suppress 属性值。
- 删除了 linefeed-treatment 属性。
- text-align-last 属性取消了 size 属性值。
- text-justify 属性新增了 tibetan 属性值。
- punctuation-trim 属性新增加了 end 属性值。
- kerning-mode:contextual 被 punctuation-trim:adjacent 替换，其他控制被移至字体模块中。
- text-shadow 属性现在可以继承。
- 新增 text-outline 属性。
- 新增 text-emphasis 属性，用于替换 font-emphasis 属性。
- 重新定义了 text-indent 属性。
- 重新设计了 hanging-punctuation 属性。

最新版本的文本模型与 2005 年的版本[⊖] (<http://www.w3.org/TR/2005/WD-css3-text-20050627/>) 相比，也进行了适当的修订，其中增加了 text-emphasis 和 text-outline 属性，移除了 font-emphasis 属性，更多修订的细节请参阅工作文档。

3.6 HSL 色彩模式

在 Web 设计中，设计师经常需要一种直观的选择颜色的方法，以实现在主观感受上使所选择的颜色再暗一点儿或再亮一点儿的效果。在传统网页设计中，这种主观感受是无法使用 RGB 色彩模式或者十六进制方式实现的。

CSS 3 新增加了 HSL 颜色表现方式[⊖]。HSL 色彩模式是工业界的一种颜色标准，它通过对色调 (H)、饱和度 (S)、亮度 (L) 三个颜色通道的改变以及它们相互之间的叠加来获得各种颜色。这个标准几乎包括了人类视力可以感知的所有颜色，在屏幕上可以重现 16 777 216 种

[⊖] <http://www.w3.org/TR/css-3-color>

颜色，是目前运用最广的颜色系统之一。

在 CSS 3 中，HSL 色彩模式的表示语法如下：

`hsl(<length>, <percentage>, <percentage>)`

`hsl()` 函数的三个参数说明如下。

- `<length>` 表示色调 (Hue)。Hue 衍生于色盘，取值可以为任意数值，其中 0 (或 360, 或 -360) 表示红色，60 表示黄色，120 表示绿色，180 表示青色，240 表示蓝色，300 表示洋红，当然可以设置其他数值来确定不同的颜色。
- `<percentage>` 表示饱和度 (Saturation)，表示该色彩被使用了多少，即颜色的深浅程度和鲜艳程度。取值为 0% 到 100% 之间的值，其中 0% 表示灰度，即没有使用该颜色；100% 的饱和度最高，即颜色最鲜艳。
- `<percentage>` 表示亮度 (Lightness)。取值为 0% 到 100% 之间的值，其中 0% 最暗，显示为黑色，50% 表示均值，100% 最亮，显示为白色。

浏览器兼容性检测

目前，除了 IE 浏览器外，各大主流浏览器对 HSL 色彩模式的支持都比较友好，不需要借助各浏览器的私有方法来实现兼容，兼容检测的结果如下所示。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：网页配色解决方案表

在网页设计中，配色是一个难点，需要有一定的感觉，很多初学者总是配不好颜色。其实，只要我们把握住配色的基本规律和原则，具体动手时就不会很难。首先，在设计之前，应该确定网站的主色调，然后在同一色系中进行选色和配色，这样既能够保证网页色彩丰富，而又不显花哨。

例如，如果想使设计充满生气、稳健、冷清、温暖或寒冷等感觉，我们可以通过整体色调进行设计。那么，怎么控制好整体色调呢？只有控制好构成整体色调的色相、饱和度和亮度，以及不同颜色的面积等，才可以控制好整体色调。首先，要在配色中确定占大面积的颜色，并根据这一颜色来选择不同的配色方案，于是会得到不同的整体色调，从中选择一个满意的方案。

如果用暖色系作为整体色调，则会呈现出温暖的感觉，反之亦然；如果用暖色和饱和度高的颜色作为整体色调，则会给人以火辣刺激的感觉；如果以冷色和饱和度低的颜色为主色调，则会给人清冷和平静的感觉；如果用亮度高的颜色为主色调，则会给人以亮丽和轻快的感觉；如果以亮度低的颜色为主，则显得比较庄重和肃穆。取对比的色相和明度，则会显得活泼；取类似或同一色系的颜色则会让人感到稳健。色相数多，则会显得华丽；色相少，则会显得淡雅和清新。总之，网页配色方案要根据我们所要表达的内容来决定。

CSS 设计思路

首先，确定网站的主色调（即选择一个色相值），然后通过调整颜色的饱和度和亮度比重来分别设计不同的配色方案表。在网页设计中，可以根据网页内容的需要选择恰当的配色方案。使用 HSL 颜色表现方式，可以很轻松地设计出网页配色方案表，模拟效果如图 3.6 所示。

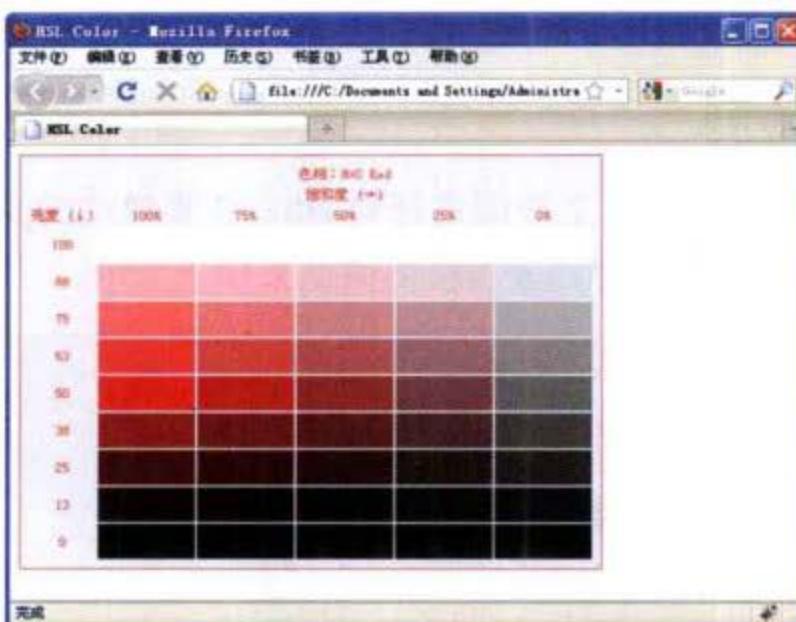


图3.6 网页配色方案表

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HSL Color</title>
<style type="text/css">
table {
    border:solid 1px red;
    background:#eee;
    padding:6px;
}
th {
    color:red;
    font-size:12px;
    font-weight:normal;
}

```

```

td {
    width:80px;
    height:30px;
}
/*第1行*/
tr:nth-child(4) td:nth-of-type(1) { background:hsl(0,100%,100%);}
tr:nth-child(4) td:nth-of-type(2) { background:hsl(0,75%,100%);}
tr:nth-child(4) td:nth-of-type(3) { background:hsl(0,50%,100%);}
tr:nth-child(4) td:nth-of-type(4) { background:hsl(0,25%,100%);}
tr:nth-child(4) td:nth-of-type(5) { background:hsl(0,0%,100%);}

/*第2行*/
tr:nth-child(5) td:nth-of-type(1) { background:hsl(0,100%,88%);}
tr:nth-child(5) td:nth-of-type(2) { background:hsl(0,75%,88%);}
tr:nth-child(5) td:nth-of-type(3) { background:hsl(0,50%,88%);}
tr:nth-child(5) td:nth-of-type(4) { background:hsl(0,25%,88%);}
tr:nth-child(5) td:nth-of-type(5) { background:hsl(0,0%,88%);}

/*第3行*/
tr:nth-child(6) td:nth-of-type(1) { background:hsl(0,100%,75%);}
tr:nth-child(6) td:nth-of-type(2) { background:hsl(0,75%,75%);}
tr:nth-child(6) td:nth-of-type(3) { background:hsl(0,50%,75%);}
tr:nth-child(6) td:nth-of-type(4) { background:hsl(0,25%,75%);}
tr:nth-child(6) td:nth-of-type(5) { background:hsl(0,0%,75%);}

/*第4行*/
tr:nth-child(7) td:nth-of-type(1) { background:hsl(0,100%,63%);}
tr:nth-child(7) td:nth-of-type(2) { background:hsl(0,75%,63%);}
tr:nth-child(7) td:nth-of-type(3) { background:hsl(0,50%,63%);}
tr:nth-child(7) td:nth-of-type(4) { background:hsl(0,25%,63%);}
tr:nth-child(7) td:nth-of-type(5) { background:hsl(0,0%,63%);}

/*第5行*/
tr:nth-child(8) td:nth-of-type(1) { background:hsl(0,100%,50%);}
tr:nth-child(8) td:nth-of-type(2) { background:hsl(0,75%,50%);}
tr:nth-child(8) td:nth-of-type(3) { background:hsl(0,50%,50%);}
tr:nth-child(8) td:nth-of-type(4) { background:hsl(0,25%,50%);}
tr:nth-child(8) td:nth-of-type(5) { background:hsl(0,0%,50%);}

/*第6行*/
tr:nth-child(9) td:nth-of-type(1) { background:hsl(0,100%,38%);}
tr:nth-child(9) td:nth-of-type(2) { background:hsl(0,75%,38%);}
tr:nth-child(9) td:nth-of-type(3) { background:hsl(0,50%,38%);}
tr:nth-child(9) td:nth-of-type(4) { background:hsl(0,25%,38%);}
tr:nth-child(9) td:nth-of-type(5) { background:hsl(0,0%,38%);}

/*第7行*/
tr:nth-child(10) td:nth-of-type(1) { background:hsl(0,100%,25%);}
tr:nth-child(10) td:nth-of-type(2) { background:hsl(0,75%,25%);}
tr:nth-child(10) td:nth-of-type(3) { background:hsl(0,50%,25%);}
tr:nth-child(10) td:nth-of-type(4) { background:hsl(0,25%,25%);}
tr:nth-child(10) td:nth-of-type(5) { background:hsl(0,0%,25%);}

/*第8行*/
tr:nth-child(11) td:nth-of-type(1) { background:hsl(0,100%,13%);}
tr:nth-child(11) td:nth-of-type(2) { background:hsl(0,75%,13%);}
tr:nth-child(11) td:nth-of-type(3) { background:hsl(0,50%,13%);}
tr:nth-child(11) td:nth-of-type(4) { background:hsl(0,25%,13%);}
tr:nth-child(11) td:nth-of-type(5) { background:hsl(0,0%,13%);}

/*第9行*/

```

选择行

选择单元格（列）

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

/* 第 1 列 */
/* 第 2 列 */
/* 第 3 列 */
/* 第 4 列 */
/* 第 5 列 */

```

tr:nth-child(12) td:nth-of-type(1) { background:hsl(0,100%,0%); } /* 第1列 */
tr:nth-child(12) td:nth-of-type(2) { background:hsl(0,75%,0%); } /* 第2列 */
tr:nth-child(12) td:nth-of-type(3) { background:hsl(0,50%,0%); } /* 第3列 */
tr:nth-child(12) td:nth-of-type(4) { background:hsl(0,25%,0%); } /* 第4列 */
tr:nth-child(12) td:nth-of-type(5) { background:hsl(0,0%,0%); } /* 第5列 */
</style>
</head>
<body>
<table class="hslexample">
  <tbody>
    <tr>
      <th>&ampnbsp</th>
      <th colspan="5">色相: H=0 Red </th>
    </tr>
    <tr>
      <th>&ampnbsp</th>
      <th colspan="5">饱和度 (&rarr;) </th>
    </tr>
    <tr>
      <th>亮度 (&darr;) </th>
      <th>100% </th>
      <th>75% </th>
      <th>50% </th>
      <th>25% </th>
      <th>0% </th>
    </tr>
    ...
  </tbody>
</table>
</body>
</html>

```

指定色相 指定饱和度 指定亮度

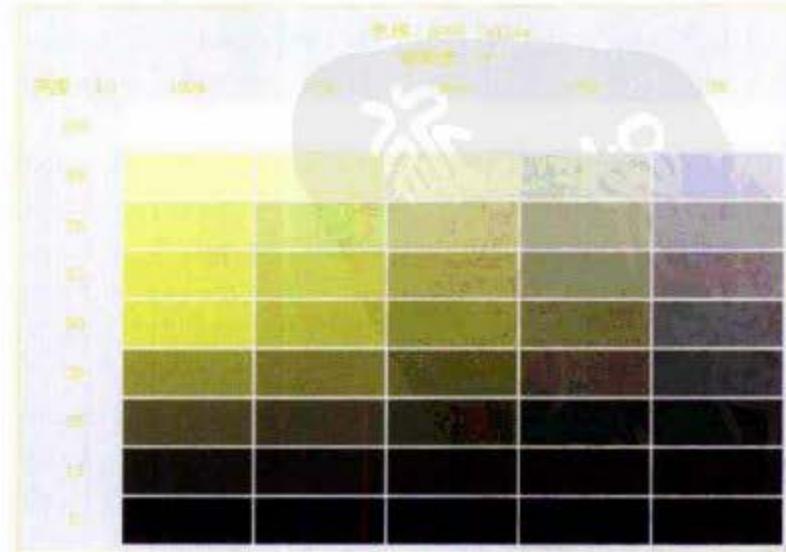
省略的表格结构

在上面案例的基础上，我们可以设计出不同的配色方案。例如，下面的图分别演示了几种基本的配色方案表。

● 橙色系：朝气活泼、豁然开朗



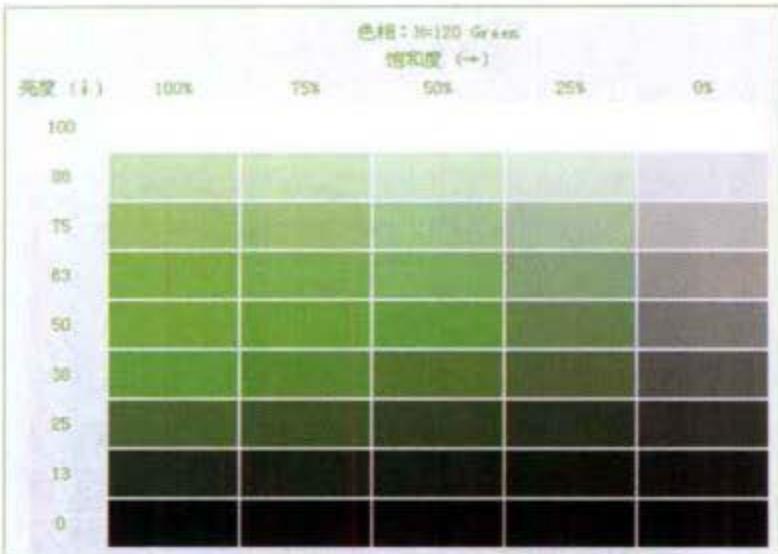
● 黄色系：明亮喜庆、甜蜜幸福



● 黄绿色系：自然清新、年轻且富有希望



● 绿色系：新鲜自然、明朗宁静



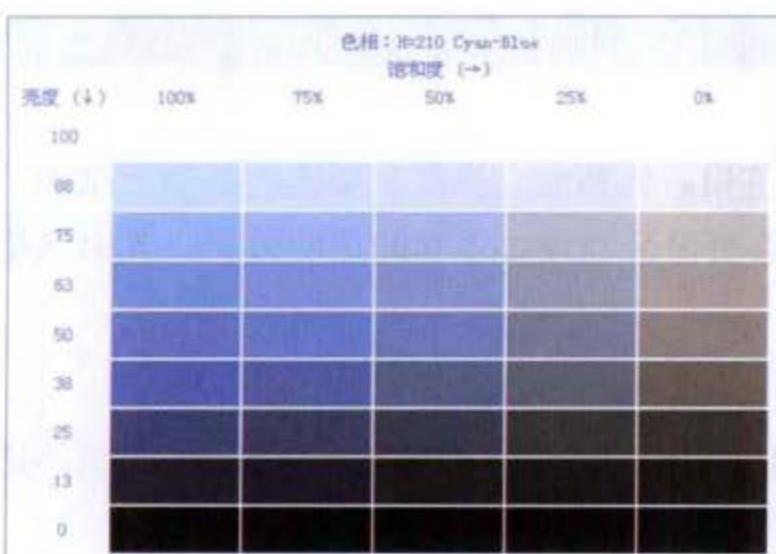
● 青绿色系：健康清新、充满信心和活力



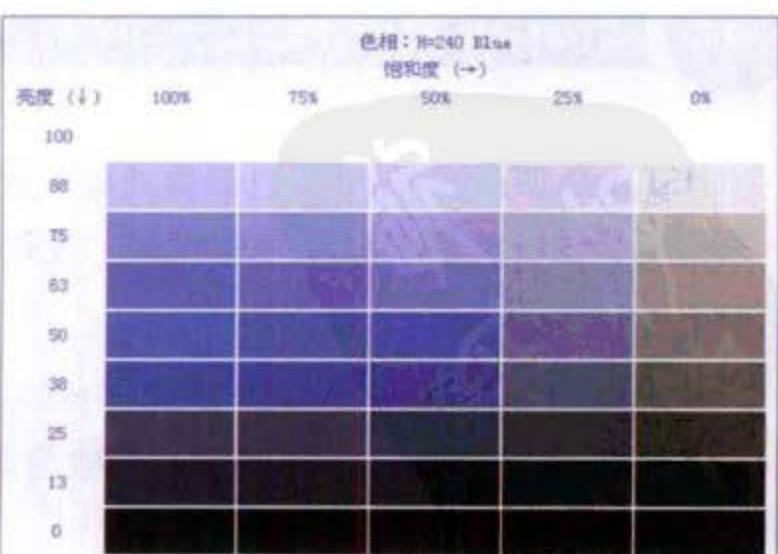
● 青色系：坚定、古朴庄重



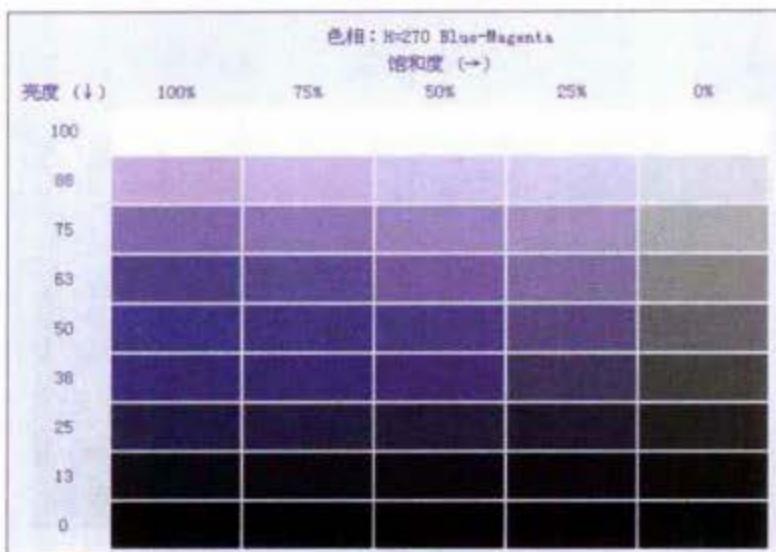
● 青蓝色系：爽朗开阔、清涼高远



● 蓝色系：和平、淡雅、洁净



● 蓝紫色系：成熟、冷静、高贵



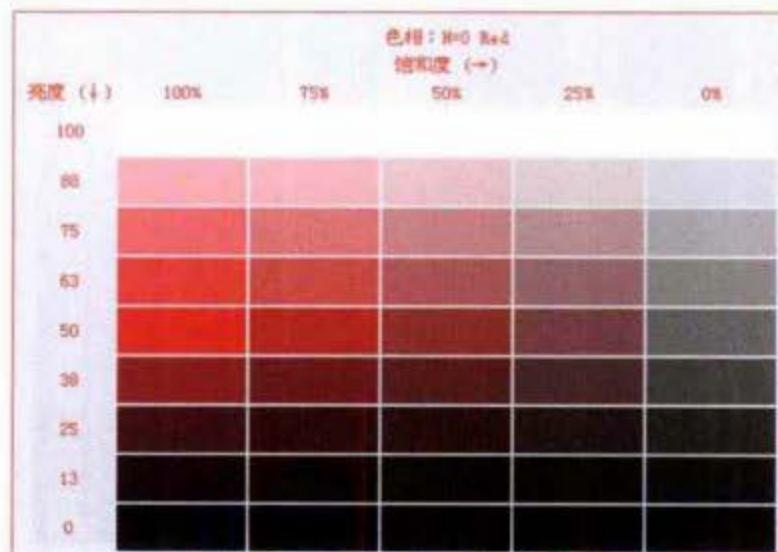
● 紫色系：神秘尊贵、高雅脱俗



● 紫红色系：浪漫柔和、华丽高贵



● 红色系：吉祥幸福、古典



3.7 HSLA 色彩模式

HSLA 色彩模式是 HSL 色彩模式的扩展，在色相、饱和度、亮度三要素的基础上增加了不透明度参数。使用 HSLA 色彩模式，设计师能够更灵活地设计不同的透明效果。其语法格式如下：

```
hsla(<length> , <percentage> , <percentage> , <opacity>)
```

其中前 3 个参数与 hsl() 函数的参数的意义和用法相同，第 4 个参数 <opacity> 表示不透明度，取值在 0 到 1 之间。

浏览器兼容性检测

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✓ Opera 10.0		✗ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✗ Chrome 4.0.x

实战体验：模拟渐变色条

下面这个案例只是一个 Demo，但是通过它能让读者明白 HSLA 颜色模式在网页中的应用，效果如图 3.7 所示。

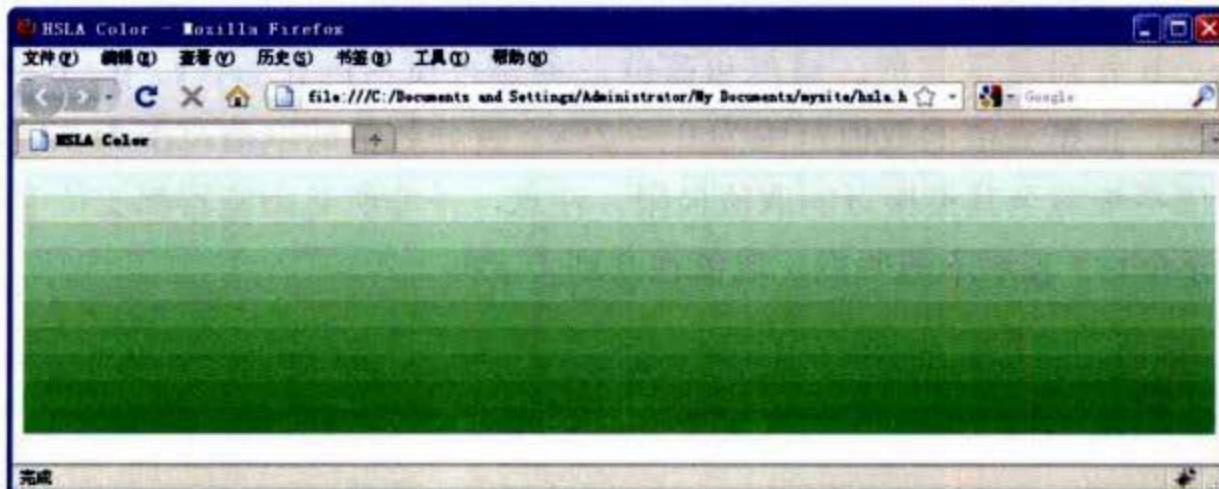


图 3.7 模拟渐变色条效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HSLA Color</title>
<style type="text/css">
div {height:20px;}
div:nth-child(1) { background:hsla(120,50%,50%,0.1); }
div:nth-child(2) { background:hsla(120,50%,50%,0.2); }
div:nth-child(3) { background:hsla(120,50%,50%,0.3); }
div:nth-child(4) { background:hsla(120,50%,50%,0.4); }
div:nth-child(5) { background:hsla(120,50%,50%,0.5); }
div:nth-child(6) { background:hsla(120,50%,50%,0.6); }
div:nth-child(7) { background:hsla(120,50%,50%,0.7); }
div:nth-child(8) { background:hsla(120,50%,50%,0.8); }
div:nth-child(9) { background:hsla(120,50%,50%,0.9); }
div:nth-child(10) { background:hsla(120,50%,50%,1); }

```

颜色的不透明度逐渐增强

```

</style>
</head>
<body>
<div></div><div></div><div></div>
<div></div><div></div><div></div>
<div></div><div></div><div></div><div></div>
</body>
</html>

```

3.8] RGBA 色彩模式

RGBA 色彩模式是 RGB 色彩模式的扩展，在红、绿、蓝三原色的基础上增加了不透明度参数。其语法格式如下所示：

`rgba(r, g, b, <opacity>)`

其中 r、g、b 分别表示红色、绿色和蓝色三种原色所占的比重。r、g、b 的值可以是正整数或者百分数，正整数值的取值范围为 0 ~ 255，百分数值的取值范围为 0.0% ~ 100.0%，超出范围的数值将被截至其最接近的取值极限。注意，并非所有浏览器都支持使用百分数值。第四个参数 `<opacity>` 表示不透明度，取值在 0 到 1 之间。

● 浏览器兼容性检测

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.1	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✓ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：设计带阴影边框的表单

在设计带有阴影或者其他效果的边框时，设计师通常会借助背景图片来实现，因为 CSS 无法实现这种效果。这种方法比较安全，但是前期工作量比较大，也给带宽增加了一定的负担。现在，通过 CSS 3 新增加的功能，这个问题使用几行代码就可以快速搞定。这个方法确实很好，但是在目前主流浏览器尚不完全支持的情况下，还存在一定的风险。

CSS设计思路

使用 CSS 3 新增加的 box-shadow 属性（详细讲解请参阅后面章节的内容），然后使用 RGBA 颜色模式为表单元素设置半透明度的阴影，从而实现一种润边形式的阴影效果，其效果如图 3.8 所示。

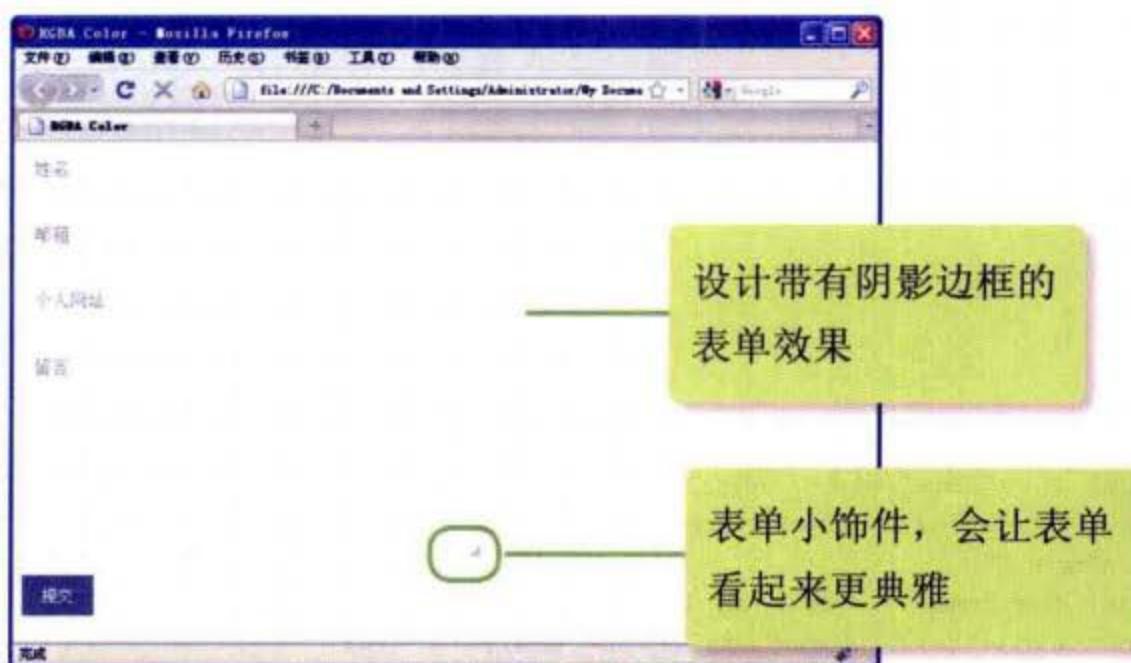


图 3.8 在 Firefox 3.6 中的演示效果

整个案例的源代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>RGBA Color</title>
<style type="text/css">
input, textarea { /* 统一输入域样式 */
    padding: 4px;
    border: solid 1px #E5E5E5;
    outline: 0;
    font: normal 13px/100% Verdana, Tahoma, sans-serif;
    width: 200px;
    background: #FFFFFF;
    box-shadow: rgba(0, 0, 0, 0.1) 0px 0px 8px; /* 设置边框阴影效果 */
    -moz-box-shadow: rgba(0, 0, 0, 0.1) 0px 0px 8px; /* 兼容 Mozilla 类型的浏览器，如 FF */
    -webkit-box-shadow: rgba(0, 0, 0, 0.1) 0px 0px 8px; /* 兼容 Webkit 引擎，如 Chrome 和 Safari 等 */
}
textarea { /* 定义文本区域样式 */
    width: 400px;
    max-width: 400px;
    height: 150px;
    line-height: 150%;
    background: url(images/form-shadow.png) no-repeat bottom right;
}

/* 为文本区域添加小部件 */
input[type="text"], input[type="password"] {
    width: 150px;
    height: 25px;
    border: 1px solid #ccc;
    padding: 2px 5px;
}
```

rgba(0,0,0, 0.1) 表示不透明度为0.1的黑色。这里不宜直接设置为浅灰色，因为对于非白色背景来说，浅灰色会发虚，而半透明效果可以避免这种情况

为文本区域添加小部件

```

input:hover, textarea:hover, input:focus, textarea:focus { border-color: #C9C9C9; }
label /*定义标签样式*/
    margin-left: 10px;
    color: #999999;
    display:block; /*以块状显示，实现分行显示*/
}
.submit input /*设计提交按钮的样式*/
    width:auto;
    padding: 9px 15px;
    background: #617798;
    border: 0;
    font-size: 14px;
    color: #FFFFFF;
}
</style>
</head>
<body>
<form>
    <p class="name">
        <label for="name">姓名</label>
        <input type="text" name="name" id="name" /></p>
    <p class="email">
        <label for="email">邮箱</label>
        <input type="text" name="email" id="email" /></p>
    <p class="web">
        <label for="web">个人网址</label>
        <input type="text" name="web" id="web" /></p>
    <p class="text">
        <label for="text">留言</label>
        <textarea name="text" id="text"></textarea></p>
    <p class="submit">
        <input type="submit" value="提交" /></p>
    </form>
</body>
</html>

```

设计鼠标的动态效果

3.9 不透明度——opacity 属性

CSS3 除了在 HSLA 和 RGBA 颜色表示方式中支持色彩的不透明度外，还专门定义了 opacity 属性，通过设置该属性能够使任何元素呈现出半透明效果，opacity 属性的基本语法如表 3.8 所示。

表3.8 opacity属性的基本语法

语法项目	说明
值	<alphavalue> inherit
初始值	1

(续)

语法项目	说明
适用于	所有元素
可否继承	无
百分比	N/A
媒介	视觉

取值简单说明：

- <alphavalue> 是由浮点数和单位标识符组成的长度值。不可为负值，默认值为 1。 opacity 取值为 1 时，则元素是完全不透明的；反之，值为 0 时，元素是完全透明的，不可见。1 到 0 之间的任何值都表示该元素的不透明程度。
- inherit 表示继承，即继承父元素的不透明性。

浏览器兼容性检测

除了 IE 8 及其以前版本的浏览器外，其他主流浏览器都支持该属性。针对 IE 浏览器，可以使用它的私有属性 filter 来兼容：filter:alpha(opacity=value);。该属性的浏览器兼容情况详细说明如下：

				
✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.1	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计灯箱广告背景布

灯箱广告效果在图片展和应用非常广，应该算是弹出层的一种应用，不过它与弹出层有所区别的是，灯箱广告插件一般都支持图片画廊模式。

其设计原理就是加一个 Div 层，然后让它居中显示，在默认状态下借助 CSS 隐藏这个层，当启动灯箱广告时，显示它，并通过不透明层性实现半透明显示。

整个案例的网页源代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>opacity</title>
<style type="text/css">
body {
    margin:0;
    padding:0;
}
div { position:absolute; }
.bg {
    width:100%;
    height:100%;
    background:#000;
    opacity:0.7;
    filter:alpha(opacity=70);
}
.lightbox {
    left:50px;
    top:50px;
}
</style>
</head>
<body>
<div class="web"></div>
<div class="bg"></div>
<div class="lightbox"></div>
</body>
</html>

```

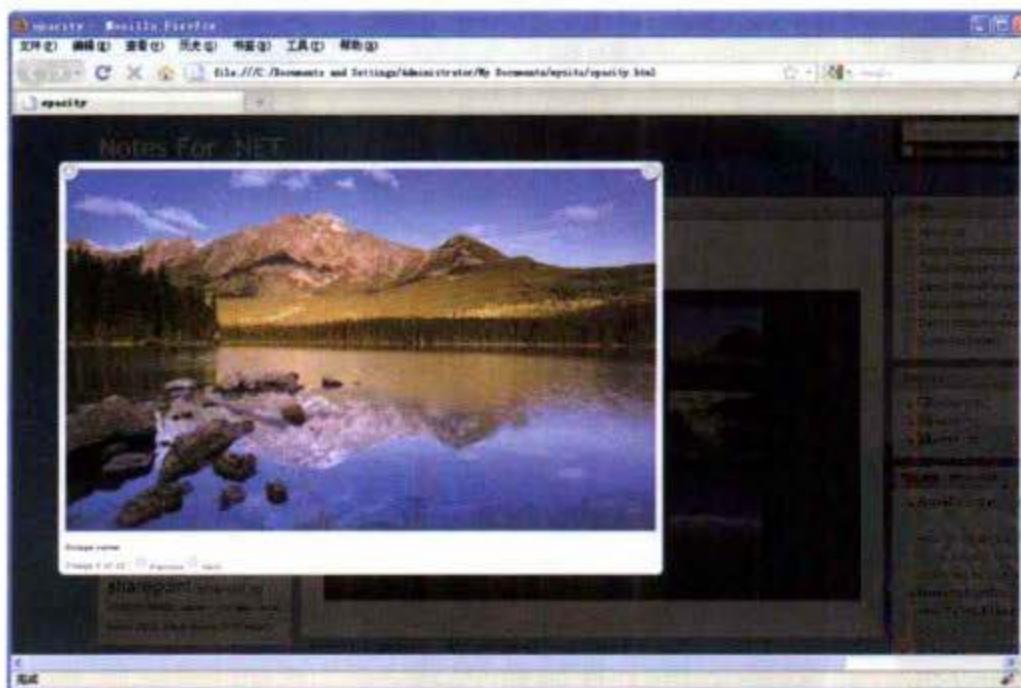


图3.9 在Firefox 3.6中的演示效果

新增的弹性盒模型

在 CSS 1 中，W3C 初步制订了元素的基本外形特征 (<http://www.w3.org/TR/CSS1/#box-properties>)，如宽、高、边框、补白、边界、显示等属性（如表 4.1 所示）。CSS 2.1 中提出了盒模型（box model）的概念。盒模型高度提炼了所有元素的基本特征，详细说明请参阅 <http://www.w3.org/TR/CSS21/box.html> 中的介绍。CSS 3 在 CSS 2 的基础上引入了一些新的盒模型技术参数，提出了弹性盒模型的概念[⊖]，方便设计师更灵活地设计页面上各个容器的大小和位置。

表4.1 CSS 1 定义的盒形元素的基本属性

属性	类型	说明
margin-top	边界	定义元素顶部边界的宽
margin-right	边界	定义元素右侧边界的宽
margin-bottom	边界	定义元素底部边界的宽
margin-left	边界	定义元素左侧边界的宽
margin	边界，复合属性	定义元素各边界的宽
border-top	边框	定义元素顶部边框的样式
border-right	边框	定义元素右侧边框的样式
border-bottom	边框	定义元素底部边框的样式
border-left	边框	定义元素左侧边框的样式
border-width	边框	定义元素各边框的宽度
border-color	边框	定义元素各边框的颜色
border-style	边框	定义元素各边框的形状
border-top-width	边框	定义元素顶部边框的宽度
border-right-width	边框	定义元素右侧边框的宽度
border-bottom-width	边框	定义元素底部边框的宽度

⊖ <http://www.w3.org/TR/css3-flexbox/>

(续)

属性	类型	说明
border-left-width	边框	定义元素左侧边框的宽度
border	边框, 复合属性	定义元素各边框的样式
padding-top	补白	定义元素顶部补白的宽
padding-right	补白	定义元素右侧补白的宽
padding-bottom	补白	定义元素底部补白的宽
padding-left	补白	定义元素左侧补白的宽
padding	补白, 复合属性	定义元素各边补白的宽
width	大小	定义元素的宽度
height	大小	定义元素的高度
float	显示	定义元素浮动显示
clear	显示	清除元素浮动显示

在 CSS 2.1 中, 元素的边框、补白和边界被统一称为为元素的盒模型, 而元素的宽、高、浮动显示等特性归为布局模型的范畴。同时, CSS 2.1 规范对边框样式进行了完善, 增强了对边框颜色和样式的定义, 以方便设计师更有针对性地设计元素的边框样式(如表 4.2 所示)。

表4.2 CSS 2.1新增的盒模型属性

属性	类型	说明
border-top-color	边框	定义元素顶部边框的颜色
border-right-color	边框	定义元素右侧边框的颜色
border-bottom-color	边框	定义元素底部边框的颜色
border-left-color	边框	定义元素左侧边框的颜色
border-top-style	边框	定义元素顶部边框的形状
border-right-style	边框	定义元素右侧边框的形状
border-bottom-style	边框	定义元素底部边框的形状
border-left-style	边框	定义元素左侧边框的形状

CSS 3 引入了新的盒模型处理机制, 即弹性盒模型, 该模型用于决定元素在盒子中的分布方式以及如何处理盒子的可用空间。这与 XUL (Firefox 浏览器的用户交互语言) 相似, 其他语言也使用相同的盒模型, 如 XAML、GladeXML 等。通过弹性盒模型, 设计师可以很轻松地创建自适应浏览器窗口的流动布局或自适应字体大小的弹性布局。

为了适应弹性盒模型的表现需要, CSS 3 新增了 8 个属性(如表 4.3 所示), 借助这些新增属性, 设计师会更加从容自信。

表4.3 CSS 3新增的盒模型属性

属性	类型	说明
box-align	弹性盒模型	定义子元素在盒子内垂直方向上的空间分配方式
box-direction	弹性盒模型	定义盒子的显示顺序
box-flex	弹性盒模型	定义子元素在盒子内的自适应尺寸
box-flex-group	弹性盒模型	定义自适应子元素群组
box-lines	弹性盒模型	定义子元素分列显示
box-ordinal-group	弹性盒模型	定义子元素在盒子内的显示位置
box-orient	弹性盒模型	定义盒子分布的坐标轴
box-pack	弹性盒模型	定义子元素在盒子内水平方向的空间分配方式

4.1 定义盒子的布局取向——box-orient 属性

box-orient 属性可用于定义盒子元素内部的流动布局方向，该属性的基本语法如表 4.4 所示。注意，在使用弹性盒子模型 (Flexible Box Module) 时，用户需要先把父容器的 display 属性设置为 box 或者 inline-box。

表4.4 box-orient属性的基本语法

语法项目	说明
值	horizontal vertical inline-axis block-axis inherit
初始值	inline-axis
适用于	盒子元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- horizontal：盒子元素从左到右在一条水平线上显示它的子元素。
- vertical：盒子元素从上到下在一条垂直线上显示它的子元素。
- inline-axis：盒子元素沿着内联轴显示它的子元素。
- block-axis：盒子元素沿着块轴显示它的子元素。

注意，inline-axis 和 block-axis 是根据书写模式来决定的。例如，在英语关键字中包括垂直、水平和地图等模式，所以内联轴和块轴也会不同。

浏览器兼容性检测

弹性盒模型规范是 W3C 标准化组织于 2009 年发布的，目前还没有主流浏览器对其进行支持，不过采用 Webkit 和 Mozilla 渲染引擎的浏览器都自定义了一套私有属性，用来支持弹性盒模型。

- 采用 Webkit 渲染引擎的浏览器主要包括 Safari 和 Chrome 浏览器，该引擎支持以 -webkit 为前缀的私有属性，即 -webkit-box-orient。
- Mozilla 渲染引擎的浏览器主要包括 Firefox 浏览器，该引擎支持以 -moz 为前缀的私有属性，即 -moz-box-orient。

各主流浏览器对 box-orient 属性的兼容情况如下所示。

				
✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计多栏布局

在基于 CSS 2.1 规范的传统盒模型中，HTML 文档流总是在垂直方向上排列盒子。使用弹性盒模型可以重定义文档流的顺序。但是，要开启弹性盒模型，用户只需要设置盒子的 display 属性值为 box（或 inline-box）即可。为了能够兼容 Firefox、Safari 和 Chrome 浏览器，设计师可以使用 display:-moz-box; 和 display:-webkit-box; 私有属性值进行声明。

例如，在下面这个简单的示例中，三个 div 元素将遵循传统盒模型的流动方式，从上到下垂直并列显示（如图 4.1 所示）。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-orient</title>
<style type="text/css">
div { height:50px; }
#box1 { background:#F6F; }
#box2 { background:#3F9; }
#box3 { background:#FC0; }
</style>
</head>
<body>
```

```
<div id="box1">盒子1</div>
<div id="box2">盒子2</div>
<div id="box3">盒子3</div>
</body>
</html>
```

现在，我们在文档的内部样式表中增加如下样式，根据 W3C 的规则，理论上的效果会如图 4.2 所示。也就是说，body 元素内包含的子盒子元素根据垂直轴（即 Y 轴）进行流动分布。

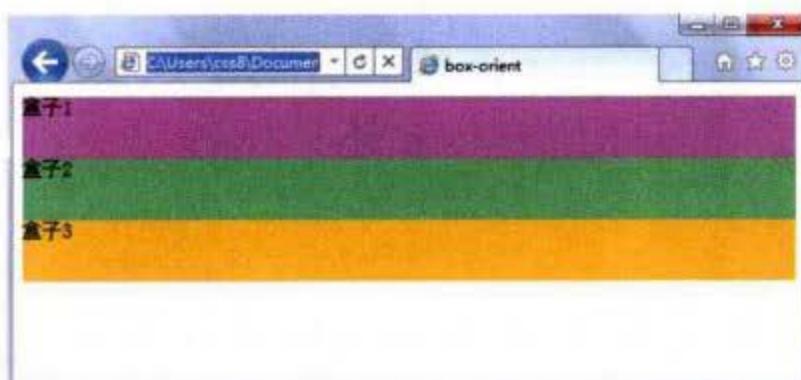


图4.1 在IE 9中演示效果

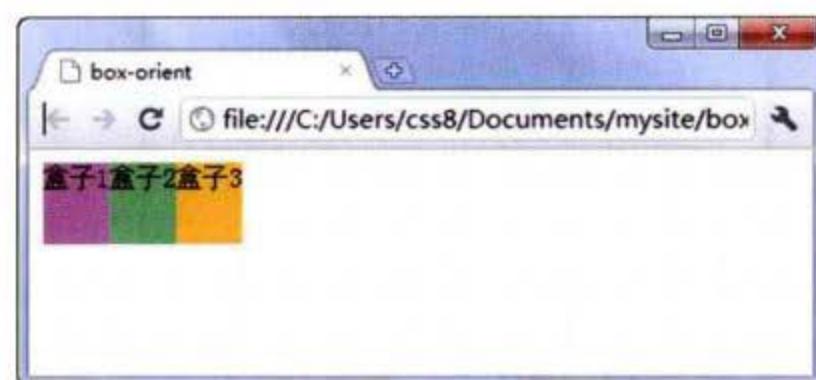


图4.2 在Chrome 1中的演示效果

```
body{
    display : box; /* 定义元素为盒子显示 */
    orient: horizontal; /* 定义盒子元素内的元素从左到右流动显示 */
}
```

由于没有浏览器支持 `display:box;` 样式，则需要采用不同渲染引擎的私有属性。

```
<style type="text/css">
div { height:50px; }
#box1 { background:#F6F; }
#box2 { background:#3F9; }
#box3 { background:#FC0; }
body{
    display : box; /* 标准声明：盒子显示 */
    display : -moz-box; /* 兼容 Mozilla Gecko 引擎浏览器 */
    display : -webkit-box; /* 标准声明：定义盒子内元素的流向 */
    orient: horizontal; /* 定义元素为盒子显示 */
    -mozbox-orient: horizontal; /* 兼容 Mozilla Gecko 引擎浏览器 */
    -webkit-box-orient:horizontal; /* 兼容 Webkit 引擎浏览器 */
}
</style>
```

借助弹性盒模型原理，我们可以模拟设计一个三栏布局样式，效果如图 4.3 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-orient</title>
<style type="text/css">
body { /* 文档样式 */
    margin:0;padding:0; /*清除页边距*/
}
```

```

text-align:center; /*文档居中对齐*/
background:#170843;}
#box {
margin:auto; /*文档居中对齐*/
text-align:left;
width:975px;}
#box1 { width:185px; }
#box2 { width:601px; }
#box3 { width:189px; }

#SUB-BOX {
display : -moz-box;
display : -webkit-box;
display : box;
box-orient: horizontal;
-mozbox-box-orient: horizontal;
-webkit-box-orient:horizontal;
}

}
</style>
</head>
<body>
<div id="box">
<div id="box0"></div>
<div id="sub-box">
<div id="box1"></div>
<div id="box2"></div>
<div id="box3"></div>
</div>
</div>
</body>
</html>

```



图4.3 在Chrome 7.0中的演示效果

实际上，在传统布局方式下，如果定义并列显示的三个栏目块显示为行内块状（`display:inline-block;`）或者内联元素（`display:inline;`），则也可以实现相同的设计效果，但是显示技术却明显不同。

4.2 定义盒子的布局顺序——box-direction 属性

box-direction 属性可以改变盒子元素中内部元素的流动顺序，该属性的基本语法如表 4.5 所示。

表4.5 box-direction属性的基本语法

语法项目	说明
值	normal reverse inherit
初始值	normal
适用于	盒子元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- ❑ normal：正常显示顺序，即如果盒子元素的 box-orient 属性值为 horizontal，则其包含的子元素按照从左到右的顺序显示，即每个子元素的左边总是靠近前一个子元素的右边；如果盒子元素的 box-orient 属性值为 vertical，则其包含的子元素按照从上到下的顺序显示。
- ❑ reverse：反向显示，盒子所包含的子元素的显示顺序将与 normal 相反。
- ❑ inherit：继承上级元素的显示顺序。

浏览器兼容性检测

Webkit 引擎的浏览器（包括 Safari 和 Chrome 等）支持 -webkit-box-direction 私有属性，Mozilla Gecko 引擎的浏览器（包括 Firefox 等）支持 -moz-box-direction 私有属性。

各主流浏览器对 box-direction 属性的支持情况如下所示。

				
✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：反向布局网页

下面我们继续以上一节的案例为基础，演示如何快速反转页面中左右两个侧栏的显示位置，效果如图 4.4 所示。



图 4.4 在 Chrome 7.0 中的演示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-orient</title>
<style type="text/css">
body { margin:0; padding:0; text-align:center; background:#170843; }
#box { margin:auto; text-align:left; width:975px; }
#box1 { width:185px; }
#box2 { width:601px; }
#box3 { width:189px; }
#sub-box {
    display : -moz-box;
    display : -webkit-box;
    display : box;
    box-orient: horizontal;
    -mozbox-orient: horizontal;
    -webkit-box-orient:horizontal;
    -moz-box-direction : reverse; /* 兼容 Mozilla Gecko 引擎 */
    -webkit-box-direction : reverse; /* 兼容 Webkit 引擎 */
    box-direction : reverse; /* 兼容标准 */
}
</style>
</head>
<body>
<div id="box">
    <div id="box0"></div>
    <div id="sub-box">
        <div id="box1"></div>
        <div id="box2"></div>
        <div id="box3"></div>
    </div>
</div>

```

定义盒子元素

定义盒子布局取向

定义盒子布
局顺序

```

</div>
</div>
</body>
</html>

```

4.3 定义盒子布局位置——box-ordinal-group 属性

box-direction 属性可以改变盒子内部元素的流动顺序，而 box-ordinal-group 属性能够设置每个子元素在盒子中的具体显示位置，该属性的基本语法如表 4.6 所示。

表4.6 box-ordinal-group属性的基本语法

语法项目	说明
值	<integer>
初始值	1
适用于	盒子的子元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

<integer> 属性值是一个自然数，从 1 开始，用来设置子元素的位置序号。子元素的分布将根据这个属性值从小到大进行排列。在默认情况下，子元素将根据元素的位置进行排列。

注意，如果没有指定 box-ordinal-group 属性值的子元素，则其序号默认都为 1，并且序号相同的元素将按照它们在文档中加载的顺序进行排列。

浏览器兼容性检测

Webkit 引擎的浏览器支持 -webkit-box-ordinal-group 私有属性，Mozilla Gecko 引擎的浏览器支持 -moz-box-ordinal-group 私有属性。

各主流浏览器对 box-ordinal-group 属性的兼容情况如下所示。

				
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：垂直网页布局

在下面这个案例中，根据文档的结构顺序，4个板块的默认显示顺序为：媒体访百合、合作伙伴、网站特色和网站版权信息，效果如图 4.5 所示。



图 4.5 在 Chrome 7.0 中的演示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-ordinal-group</title>
<style type="text/css">
body {
    margin:0;
    padding:0;
    text-align:center;
    background:#d9bfe8;
}
#box {
    margin:auto;
    text-align:left;
    width:988px;
}
</style>
</head>
<body>
<div id="box">
    <div id="box1"></div>
    <div id="box2"></div>
    <div id="box3"></div>
    <div id="box4"></div>
</div>
```

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

如果根据网站新的设计要求，需要调整这4个板块的显示顺序：网站特色、媒体访百合、合作伙伴和网站版权信息，效果如图4.6所示。



图4.6 在Chrome 7.0版本浏览器下的演示效果

如果要实现该效果，则新增加的CSS样式代码如下所示：

```
<style type="text/css">
body {
    margin:0;
    padding:0;
    text-align:center;
    background:#d9bfe8;
}
#box {
    margin:auto;
    text-align:left;
    width:988px;
}
#box {
    display : -moz-box;
    display : -webkit-box;
    display : box;
    box-orient:vertical;
    -moz-box-orient:vertical;
    -webkit-box-orient:vertical;
}
#box1 /* 设置第1个元素显示在第2的位置 */
    -moz-box-ordinal-group : 2; /* 兼容 Mozilla Gecko 引擎 */
    -webkit-box-ordinal-group : 2; /* 兼容 Webkit 引擎 */
    box-ordinal-group : 2; /* 标注用法 */
}

// 网页原来的样式
// 定义盒型显示
// 定义盒内元素垂直显示
// 定义子元素的显示位置
```

```

#box2 /* 设置第 2 个元素显示在第 3 的位置 */
-moz-box-ordinal-group : 3; /* 兼容 Mozilla Gecko 引擎 */
-webkit-box-ordinal-group : 3; /* 兼容 Webkit 引擎 */
box-ordinal-group : 3; /* 标注用法 */
}

#box3 /* 设置第 3 个元素显示在第 1 的位置 */
-moz-box-ordinal-group : 1; /* 兼容 Mozilla Gecko 引擎 */
-webkit-box-ordinal-group : 1; /* 兼容 Webkit 引擎 */
box-ordinal-group : 1; /* 标注用法 */
}

#box4 /* 设置第 4 个元素显示在第 4 的位置 */
-moz-box-ordinal-group : 4; /* 兼容 Mozilla Gecko 引擎 */
-webkit-box-ordinal-group : 4; /* 兼容 Webkit 引擎 */
box-ordinal-group : 4; /* 标注用法 */
}

</style>

```

定义子元素的
显示位置

4.4 定义盒子的弹性空间——box-flex 属性

box-flex 属性能够灵活地控制子元素在盒子中的显示空间。注意，显示空间包括子元素的宽度和高度，而不只是子元素所在栏目的宽度，也可以说是子元素在盒子中所占的面积。该属性在弹性布局中非常重要，它解决了传统设计中习惯使用百分比定义弹性布局的弊端。box-flex 属性的基本语法如表 4.7 所示。

表4.7 box-ordinal-group属性的基本语法

语法项目	说明
值	<number>
初始值	0.0
适用于	盒子内部的流动子元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

<number> 属性值是一个整数或者小数。当盒子中包含多个定义了 box-flex 属性的子元素时，浏览器将会把这些子元素的 box-flex 属性值相加，然后根据它们各自的值占总值的比例来分配盒子剩余的空间。注意，box-flex 属性只有在盒子拥有确定的空间大小之后才能够正确解析。在设计中，较稳妥的做法是为盒子定义具体的 width 或 height 属性值。

浏览器兼容性检测

Webkit 引擎的浏览器支持 -webkit-box-flex 私有属性，Mozilla Gecko 引擎的浏览器支持 -moz-box-flex 私有属性。各主流浏览器对 box-flex 属性的兼容情况如下所示。

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：自适应栏目宽度设计

在传统的网页设计中，如果要把一个盒子分成三栏，比较简便的方法是把三个子盒子元素的宽度都设置为 33.3%。这种方法无法把父盒子的宽度完全填充，当父盒子的宽度足够大的时候，用户会看到未填满的空白区域，这是很不美观的。但是，如果为子元素设置了固定宽度值，弹性布局会变得更为复杂。如果使用 box-flex 属性，这个问题就会迎刃而解。

CSS 设计思路

把盒子设置为盒型显示，并让所有子元素水平并列显示，通过 box-flex 属性定义每个子元素的宽度比值，如图 4.7 所示。



图 4.7 在 Chrome 7.0 中的演示效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-flex</title>
<style type="text/css">
body {
    margin:0;
    padding:0;
    text-align:center;
}
h1, h2 { margin:2px; }
#box {
    margin:auto;
    text-align:left;
    width:1002px;
    overflow:hidden;
}
#box {
    display : -moz-box;
    display : -webkit-box;
    display : box;
    box-orient: horizontal;
    -mozbox-orient: horizontal;
    -webkit-orient:horizontal;
}
#box1 { width:201px; }
#box2, #box3 {
    border:solid 1px #CCC;
    margin:2px;
}
#box2 {
    -moz-box-flex: 4;
    -webkit-box-flex: 4;
    box-flex: 4;
}
#box3 {
    -moz-box-flex: 2;
    -webkit-box-flex: 2;
    box-flex: 2;
}
#box2 div, #box3 div { display:inline; }
</style>
</head>
<body>
<h1></h1>
<div id="box">
    <div id="box1"></div>
    <div id="box2">
        <h2></h2>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </div>
</div>

```

定义box元素盒型显示，并设置子元素水平分布

定义盒子内左侧栏目的宽度为固定显示

定义盒子内中间栏目的宽度为盒子剩余空间的2/3

定义盒子内右侧栏目的宽度为盒子剩余空间的1/3

左侧栏目

中间栏目

```

</div>
<div id="box3">
    <h2></h2>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
</div>
</div>
</body>
</html>

```

右侧栏目

弹性空间的实现和分配

在默认情况下，子元素并不具有弹性，它会尽可能的宽，以便使其所包含的内容可见，并且不会有任何溢出。如果要改变它的尺寸，则可以使用 `width` 和 `height` 属性来实现，当然也可以使用或 `min-height`、`min-width`、`max-width`、`max-height` 等属性来限制其尺寸。

但是，当 `box-flex` 属性至少为大于 0 的值时，它会变得富有弹性。当子元素具有弹性时，可以通过下面的方式来改变它的尺寸：

- 使用 `width`、`height`、`min-width`、`min-height`、`max-width` 或 `max-height` 属性来定义尺寸。
- 利用盒子的尺寸来限制子元素的弹性尺寸。
- 借助盒子富余的所有空间来限制子元素的弹性尺寸。

如果子元素没有声明大小，那么其尺寸将完全取决于盒子的大小，即子元素的大小等于盒子的大小乘以它的 `box-flex` 属性值在所有子元素的 `box-flex` 属性值总和中的百分比。使用公式表示如下：

$$\text{子元素的尺寸} = \text{盒子的尺寸} * \text{子元素的box-flex属性值} / \text{所有子元素的box-flex属性值的和}$$

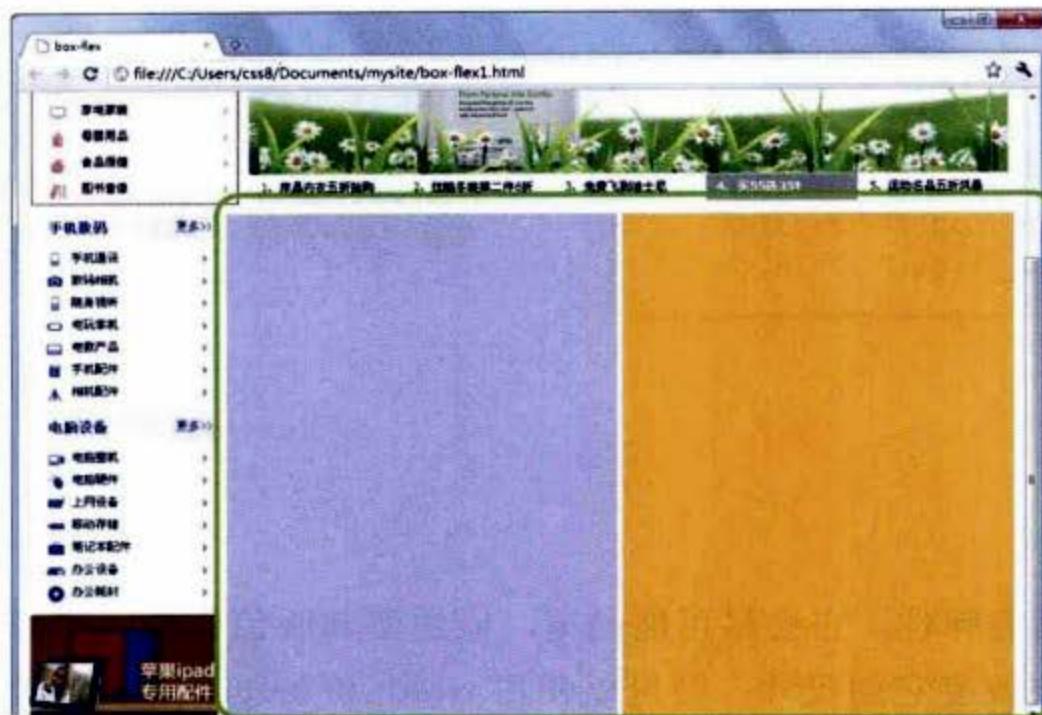
如果一个或多个子元素声明了具体的尺寸，那么其大小将计入其中，余下的弹性盒子将按照上面的原则分享剩下的可利用空间。

所以，在上面的示例中你会看到，中间栏目和右侧栏目的尺寸总是按比例占满外面盒子的富余空间。但是，由于受内部图片大小的支撑，导致子元素的空间总是大于盒子的富余空间，这时就会出现解析异常。关于这个话题，我们将在后面的章节中讨论。下面我们将清除中栏和右栏栏目中包含的图片填充内容，研究 `box-flex` 属性对子元素的布局影响。

```

<body>
<h1></h1>
<div id="box">
    <div id="box1"></div>      <!--左栏-->
    <div id="box2"></div>          <!--中栏-->
    <div id="box3"></div>          <!--右栏-->
</div>
</body>

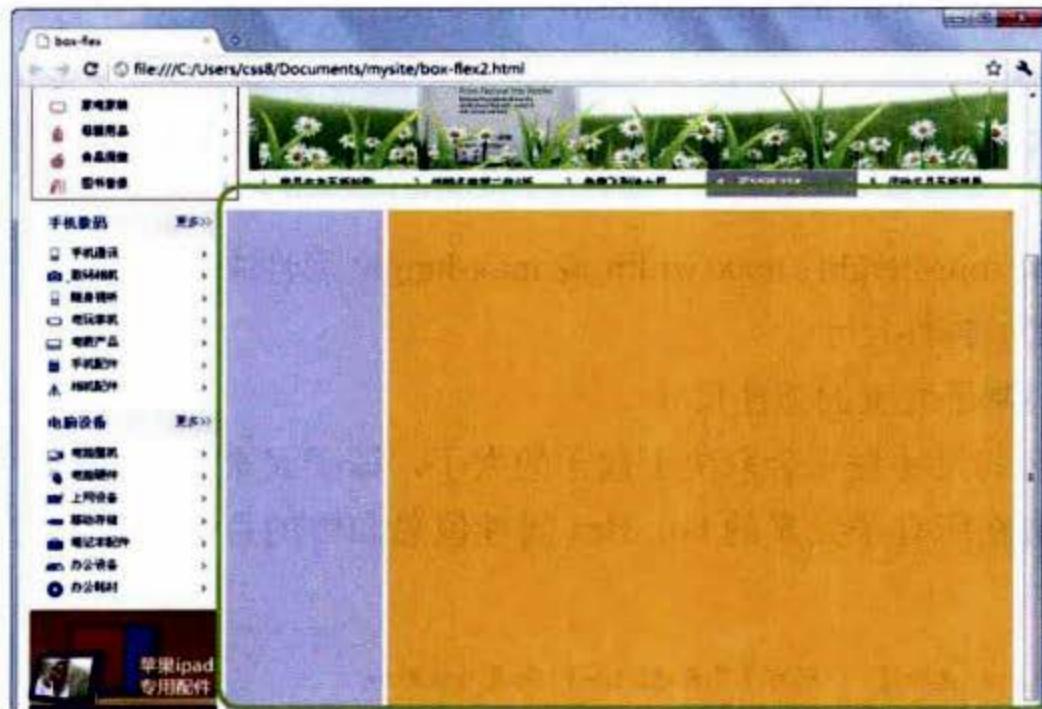
```



```
#box2 {
    -moz-box-flex: 2;
    -webkit-box-flex: 2;
    box-flex: 2;
    background:#CCF;
}
```

```
#box3 {
    -moz-box-flex: 2;
    -webkit-box-flex: 2;
    box-flex: 2;
    background:#FC0;
}
```

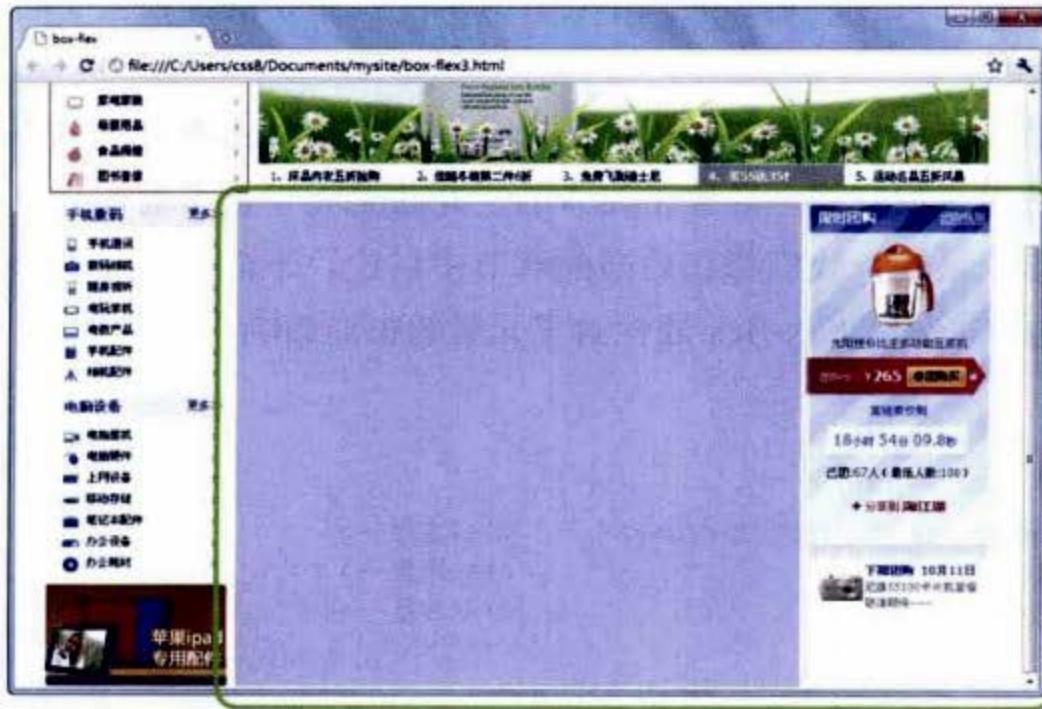
中栏和右栏各占一半空余的空间



```
#box2 {
    -moz-box-flex: 0.5;
    -webkit-box-flex: 0.5;
    box-flex: 0.5;
    background:#CCF;
}
```

```
#box3 {
    -moz-box-flex: 2;
    -webkit-box-flex: 2;
    box-flex: 2;
    background:#FC0;
}
```

中栏占1/5的空余空间，右栏占4/5空余的空间



```
#box2 {
    -moz-box-flex: 0.5;
    -webkit-box-flex: 0.5;
    box-flex: 0.5;
    background:#CCF;
}
```

```
#box3 {
    width:196px;
    background:url(images/web3_03.gif) no-repeat;
}
```

中栏弹性显示，占据所有空余空间，右栏固定大小



中栏失去弹性，收缩显示为一条线，右栏自动左移



中栏弹性为负值，失去弹性，收缩显示为一条线

4.5 管理盒子的空间——box-pack 和 box-align 属性

当弹性与非弹性元素混合排版时，有可能会出现所有子元素的尺寸大于或小于盒子的尺寸，从而出现盒子空间不足或者富余的情况，这时就需要一种方法来管理盒子的空间。如果子元素的总尺寸小于盒子的尺寸，则可以使用 box-align 和 box-pack 属性进行管理。

box-pack 属性可以在水平方向上对盒子的富余空间进行管理，该属性的基本语法如表 4.8 所示。

表4.8 box-pack属性的基本语法

语法项目	说明
值	start end center justify
初始值	start
适用于	块状元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- ❑ start：所有子元素都显示在盒子的左侧，富余的空间显示在盒子的右侧。
- ❑ end：所有子元素都显示在盒子的右侧，富余的空间显示在盒子的左侧。
- ❑ justify：富余的空间在子元素之间平均分配。在第一个子元素之前和最后一个子元素之后不分配空间。
- ❑ center：富余的空间在盒子的两侧平均分配。

box-align 属性可以在垂直方向上对盒子的富余空间进行管理，该属性的基本语法如表 4.9 所示。

表4.9 box-align属性基本语法

语法项目	说明
值	start end center baseline stretch
初始值	stretch
适用于	块状元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- ❑ start：所有子元素沿盒子的上边缘排列，都显示在盒子的上部，富余的空间显示在盒子的底部。
- ❑ end：所有子元素沿盒子的下边缘排列，都显示在盒子的底部，富余的空间显示在盒子的上部。
- ❑ center：富余的空间在盒子的上下两侧平均分配，即上面一半，下面一半。
- ❑ baseline：所有盒子沿着它们的基线排列，富余的空间可前可后显示。
- ❑ stretch：每个子元素的高度被调整到适合盒子的高度显示。

浏览器兼容性检测

Webkit 引擎的浏览器支持 -webkit-box-pack 和 -webkit-box-align 私有属性，Mozilla Gecko 引擎的浏览器支持 -moz-box-pack 和 -webkit-box-align 私有属性。各主流浏览器对 box-pack 和 box-align 这两个属性的支持情况如下所示。

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：自适应布局居中设计

在网页设计中，布局居中是 CSS 技术中的一个难点，自适应垂直居中则是难点中的难点。现在，使用弹性盒模型的 box-align 属性和 box-pack 属性可以轻松解决这个难题。

CSS 设计思路

使用 box-align 属性和 box-pack 属性为盒子定义居中显示布局，则当单击页面中的某个链接时，页面会弹出一个提示对话框，并设置其居中显示，效果如图 4.8 所示。



图 4.8 居中显示的弹出对话框

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-pack、box-align</title>
<style type="text/css">
body, html {
    height:100%;
    width:100%;
}
body {
    /*清除页边距*/
    margin:0;
    padding:0;
    /*定义文档为弹性盒子显示*/
    display : -moz-box;
    display : -webkit-box;
    display : box;
    /*页面元素水平显示，针对本示例可以省略*/
    -moz-box-orient : horizontal;
    -webkit-box-orient : horizontal;
    box-orient : horizontal;
    /*定义提示对话框页面水平居中显示*/
    -moz-box-pack : center;
    -webkit-box-pack : center;
    box-pack : center;
    /*定义提示对话框页面水平居中显示*/
    -moz-box-align : center;
    -webkit-box-align : center;
    box-align : center;
    /*以背景方式模拟页面显示*/
    background: #04082b url(images/map1.jpg) no-repeat top center;
}
#box {
    border:solid 1px red;
    padding:4px;
}
</style>
</head>
<body>
<div id="box"></div>
</body>
</html>

```

明确定义页面文档满窗口显示

定义弹性盒模型布局

定义盒内元素居中显示

关于布局空间管理的进一步分析

在上面案例的基础上，我们来进一步分析布局空间管理中的各种情况。首先，我们构建一个模拟的盒子结构。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

```

```

<title>box-pack, box-align</title>
<style type="text/css">
#box {
    border:solid 1px red;
    width:600px;
    height:400px;
    display : -moz-box;
    display : -webkit-box;
    display : box;
}

#box div { margin:4px; border:solid 1px #eee; }
#box div img { width:120px; }
</style>
</head><body>
<div id="box">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
</div>
</body></html>

```

定义盒型显示，该声明是必需的

下面利用box-align和box-pack属性分别对盒内空间进行管理。

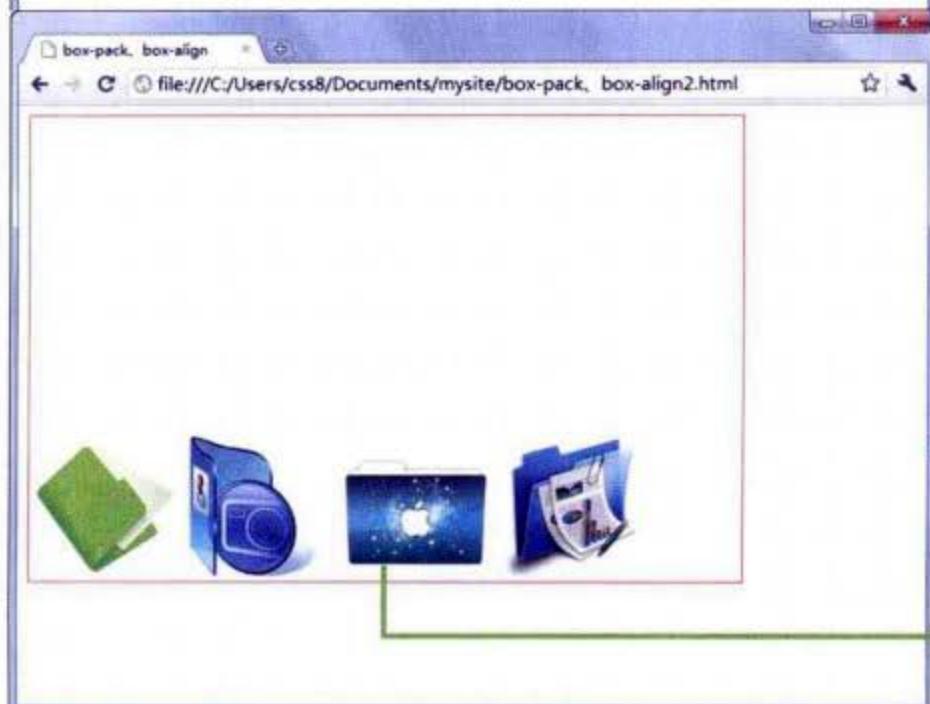


```

#box {
    /*水平空间管理*/
    -moz-box-pack : center;
    -webkit-box-pack : center;
    box-pack : center;
    /*垂直空间管理*/
    -moz-box-align : center;
    -webkit-box-align : center;
    box-align : center;
}

```

子元素全部水平和垂直居中显示

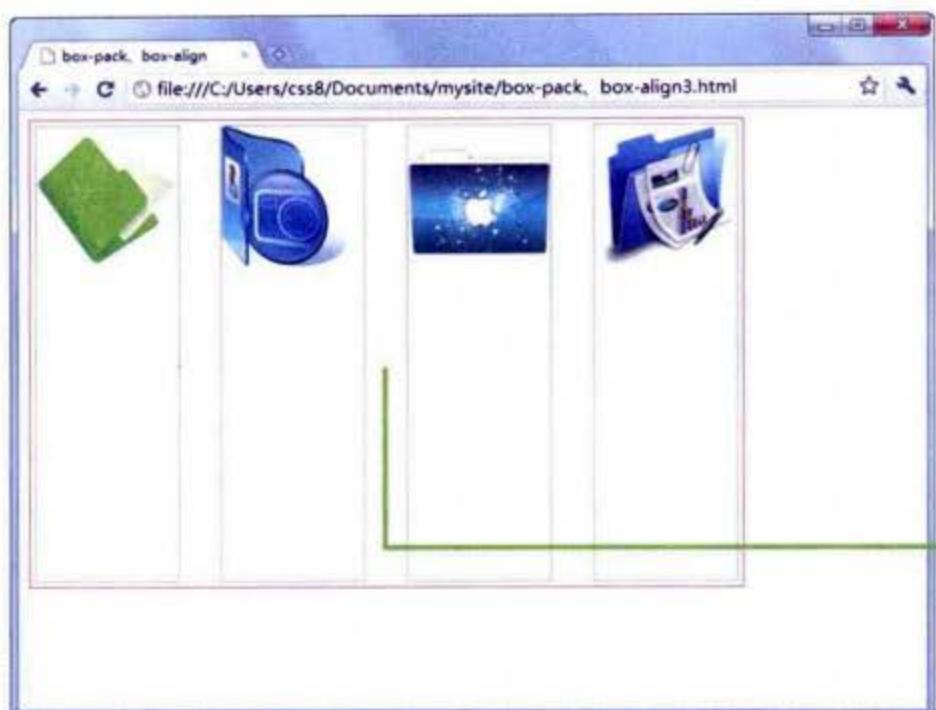


```

#box {
    /* 水平空间管理 */
    -moz-box-pack : start;
    -webkit-box-pack : start;
    box-pack : start;
    /* 垂直空间管理 */
    -moz-box-align : end ;
    -webkit-box-align : end;
    box-align : end;
}

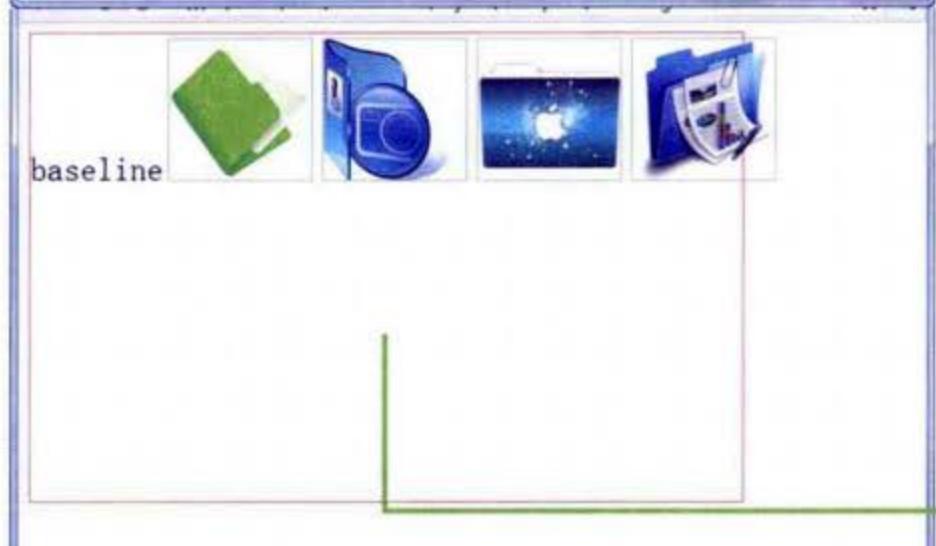
```

子元素全部位于盒子的左下角，富余空间位于右上角



```
#box {
    /* 水平空间管理 */
    -moz-box-pack: justify;
    -webkit-box-pack: justify;
    box-pack: justify;
    /* 垂直空间管理 */
    -moz-box-align: stretch;
    -webkit-box-align: stretch;
    box-align: stretch;
}
```

子元素水平平均分布，垂直伸展显示



```
#box {
    /* 水平空间管理 */
    -moz-box-pack: justify;
    -webkit-box-pack: justify;
    box-pack: justify;
    /* 垂直空间管理 */
    -moz-box-align: baseline;
    -webkit-box-align: baseline;
    box-align: baseline;
    font-size: 28px;
}
```

子元素水平平均分布，在垂直方向上按基线分布



```
#box {
    /* 水平空间管理 */
    -moz-box-pack: justify;
    -webkit-box-pack: justify;
    box-pack: justify;
    /* 垂直空间管理 */
    -moz-box-align: end;
    -webkit-box-align: end;
    box-align: end;
    /* 垂直分布 */
    -moz-box-orient: vertical;
    -webkit-box-orient: vertical;
    box-orient: vertical;
}
```

子元素全部位于盒子的右侧并平均分布，并按垂直顺序进行分布

4.6 空间溢出管理—box-lines 属性

在上一节示例中我们看到，弹性布局中盒子内的元素很容易“跑出”盒子的“包围圈”，这种现象被称为空间溢出。与传统的盒模型一样，CSS 允许使用 overflow 属性来处理溢出内容的显示方式。当然，我们还可以使用 box-lines 属性来避免空间溢出问题。该属性的基本语法如表 4.10 所示。

表4.10 box-lines属性的基本语法

语法项目	说明
值	single multiple
初始值	single
适用于	块状元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- single：所有子元素都单行或单列显示。
- multiple：所有子元素可以多行或多列显示。

浏览器兼容性检测

Webkit 引擎的浏览器支持 -webkit-box-lines 私有属性，Mozilla Gecko 引擎的浏览器支持 -moz-box-lines 私有属性。但是经实际检测，目前各主流浏览器并没有明确支持该属性。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：让子元素分行显示

下面这个示例将演示如何在盒子内设置子元素分行显示，以避免单行显示可能会溢出盒

子空间的问题。但是，由于各大主流浏览器还没有明确支持这种用法，所以导致 `box-lines` 属性被实际应用时显示为无效，如图 4.9 所示。



图4.9 `box-lines`属性的演示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>box-lines</title>
<style type="text/css">
#box {
    border:solid 1px red;
    width:600px;
    height:400px;
    display : -moz-box;
    display : -webkit-box;
    display : box;
}
#box {
    -moz-box-lines : multiple;
    -webkit-box-lines : multiple;
    box-lines : multiple;
}
#box div {
    margin:4px;
    border:solid 1px #aaa;
    -moz-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
}
#box div img { width:120px; }
</style>
```

定义子元素分行显示

```

</head>
<body>
<div id="box">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
</div>
</body>
</html>

```

4.7 弹性布局综合实战——打造新技术含量的博客

个人博客主页一般都是3行2栏式：3行包含标题头、页脚行和主体区域，2列包含任务栏和博文列表，如图4.10所示。



图4.10 博客主页的主流样式

4.7.1 构建博客页的基本结构

根据上述博客的页面布局，我们可以初步构建博客页的基本架构。

```

<div id="page">
    <header>                                <!--标题头区域-->
    </header>

```

```

<div id="main">           <!--主体头区域-->
    <section id="content">      <!--博文列表-->
    </section>
    <section id="sidebar">       <!--任务栏-->
    </section>
</div>
<footer>                  <!--页脚行区域-->
</footer>
</div>

```

4.7.2 完善博客页的结构

在这个基本三级结构基础上，我们继续充实页面的结构，以实现博客主页模板的雏形。文档的详细结构代码如下，博客模板页面的效果如图 4.11 所示。

```

<body>
<div id="page">
    <header>
        <h1>博客标题</h1>
        <p>博客概要</p>
    </header>
    <div id="main">
        <section id="content">
            <article>
                <h2>文章标题1</h2>
                <p>文章内容摘要1</p>
            </article>
            <article>
                <h2>文章标题2</h2>
                <p>文章内容摘要2</p>
            </article>
            <article class="topgroup">
                <h2>文章标题（置顶）</h2>
                <p>文章内容摘要</p>
            </article>
            <article>
                <h2>文章标题3</h2>
                <p>文章内容摘要3</p>
            </article>
            <article>
                <h2>文章标题4</h2>
                <p>文章内容摘要4</p>
            </article>
            <nav> <a href="#">上一页</a>
                  <a href="#">下一页</a> </nav>
        </section>
        <section id="sidebar">
            <div>
                <h2>
                    <label for="recherche">搜索</label>
                </h2>
                <form action="#">
                    <input type="text" id="recherche" />
                    <input type="submit" value="搜索" />
                </form>
            </div>
            <div>
                <h2>友情链接</h2>
            </div>
        </section>
    </div>

```

**标题栏使用HTML5新标签
<header>进行组织，包含博客标题、副标题，或者博客提要**

**博客列表区域通过3个
HTML 5新标签进行搭建：
<section>标签搭建博
文列表容器；<article>
标签组织博客列表；<nav>
标签组织导航行为**

**任务栏区域包含两个模
块：搜索表单和友链
接。当然，还可以添加更
多的功能模块**

```

<ul>
    <li><a href="#">链接1</a></li>
    <li><a href="#">链接2</a></li>
    <li><a href="#">链接3</a></li>
    <li class="topgroup"><a href="#">链接4 (置顶)</a></li>
    <li><a href="#">链接5</a></li>
    <li><a href="#">等</a></li>
</ul>
</div>
</section>
</div>
<footer>
    <p>CSS3新技术体验，与<a href="#">博主</a>一起讨论。</p>
    <p>更多信息请访问<a href="#">博主网站</a></p>
</footer>
</div>
</body>

```

页脚行使用HTML 5新标签`<footer>`进行组织，包含博客版权信息、博主联系方式等附加信息

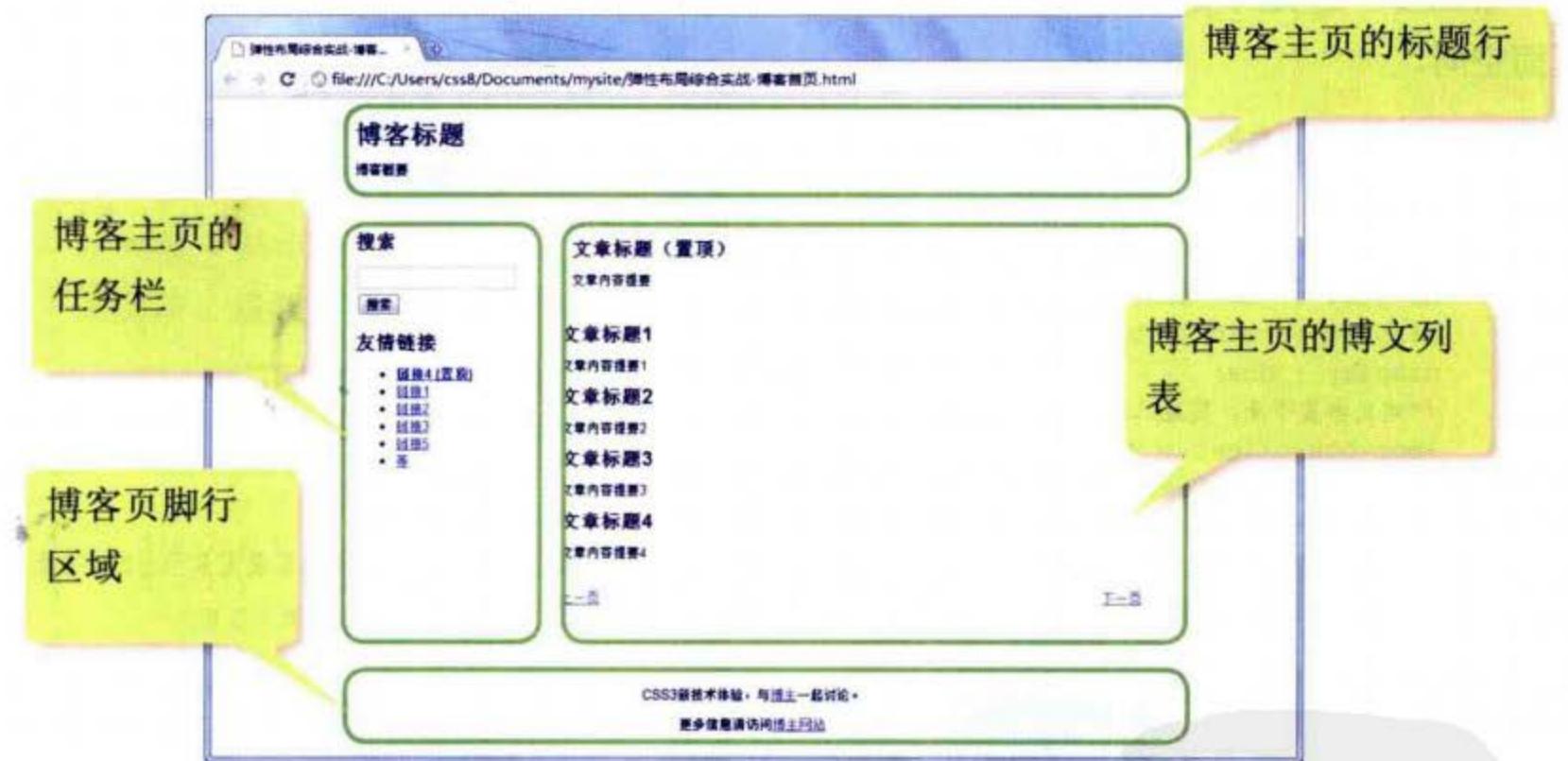


图4.11 博客主页模板效果

4.7.3 弹性布局设计

完成网页的结构设计之后，下面借助 CSS 3 新增加的弹性盒模型来设计页面样式。

第一步，把 `body` 元素定义为盒子，然后在这个盒子内进行弹性布局。

```

<style type="text/css">
html, body { /*清除页边距，初始化页面的宽度和高度*/
    padding : 0;
    margin : 0;
    height : 100%;
}

```

```

width : 100%
}
body { /*把body元素定义为弹性盒子*/
font : 0.8em Arial, Helvetica, sans-serif;
line-height : 1.4em;
/*定义弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*水平并列布局*/
-moz-box-orient : horizontal;
-webkit-box-orient : horizontal;
box-orient : horizontal;
/*设置盒子内元素水平居中*/
-moz-box-pack : center;
-webkit-box-pack : center;
box-pack : center;
}
</style>

```

定义body文档为弹性盒子，
盒内元素水平布局显示，并
设置网页水平居中显示

第二步，定义网页内容包含框为弹性盒子，设置其垂直布局显示，并定义弹性占满整个页面空间。

```

#page {/*网页内容包含框样式*/
height : 100%;
max-width : 800px;
/*定义为弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*定义垂直布局，实现三行布局效果*/
-moz-box-orient : vertical;
-webkit-box-orient : vertical;
box-orient : vertical;
/*定义为弹性尺寸，由于是单列显示，故充满整个页面空间，但是max-width属性限制网页内容最宽显示为800像素*/
-moz-box-flex : 1;
-webkit-box-flex : 1;
box-flex : 1;
}

```

第三步，定义3行的基本结构样式：标题行、主体区域和页脚行。

```

header { /*标题行样式*/
display : block;
padding : 10px;
background : #EFEFEF;
margin-bottom : 20px;
/*设置左下和右下圆角效果*/
-moz-border-radius : 0 0 10px 10px;
-webkit-border-radius : 0 0 10px 10px;
border-radius : 0 0 10px 10px;
}
footer { /*页脚行样式*/
display : block;
padding : 10px;

```

```

background : #EFEFEF;
text-align : center;
margin-top : 20px;
/*设置左上和右上圆角效果*/
-moz-border-radius : 10px 10px 0 0;
-webkit-border-radius : 10px 10px 0 0;
border-radius : 10px 10px 0 0;
}

#main { /*主体区域样式*/
/*定义弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*设置水平布局*/
-moz-box-orient : horizontal;
-webkit-box-orient : horizontal;
box-orient : horizontal;
/*挤占#page盒子内所有剩余空间*/
-moz-box-flex : 1;
-webkit-box-flex : 1;
box-flex : 1;
/*反向排列布局，让任务栏居左显示*/
-moz-box-direction : reverse;
-webkit-box-direction : reverse;
box-direction : reverse;
}

section { display : block; } /*块状显示内容分区*/

```

定义博客页主体区域为弹性布局，自动填充页面的富余空间，并定义其中的子模块为水平显示

第四步，设置博文列表区域的样式。

```

#content { /*博文列表区域样式*/
padding : 0 20px 10px 10px;
/*定义弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*垂直显示博文列表*/
-moz-box-orient : vertical;
-webkit-box-orient : vertical;
box-orient : vertical;
/*挤满固定任务栏区域外的所有富余空间*/
-moz-box-flex : 1;
-webkit-box-flex : 1;
box-flex : 1;
/*设置正常布局排列顺序*/
-moz-box-direction : normal;
-webkit-box-direction : normal;
box-direction : normal;
}

#content article { /*文章列表样式*/
display : block;
margin : 0.5em 0;
/*普通博文排在下面，置顶博文排在上面*/
-moz-box-ordinal-group : 2;
-webkit-box-ordinal-group : 2;
}

```

定义博文列表区域的弹性布局，其包含的子栏目垂直显示，主体框能够自动填充主体区域的富余空间

```

    box-ordinal-group : 2;
}
#content article h2 { margin-top : 0; }
#content nav { /*分页导航样式*/
    border-top : 1px solid #CCC;
    padding-top : 10px;
    margin-top : 10px;
    /*定义弹性盒子*/
    display : -moz-box;
    display : -webkit-box;
    display : box;
    /*水平显示分页导航链接*/
    -moz-box-orient : horizontal;
    -webkit-box-orient : horizontal;
    box-orient : horizontal;
    /*分页导航按默认顺序排在底部*/
    -moz-box-ordinal-group : 2;
    -webkit-box-ordinal-group : 2;
    box-ordinal-group : 2;
}
#content nav a { /*定义导航链接的样式*/
    display : block;
    /*平均分布两个导航按钮的空间*/
    -moz-box-flex : 1;
    -webkit-box-flex : 1;
    box-flex : 1;
}
#content nav a:last-child { text-align : right; }
#content .topgroup { /*置顶博文样式*/
    /*设置置顶博文显示在顶部*/
    -moz-box-ordinal-group : 1;
    -webkit-box-ordinal-group : 1;
    box-ordinal-group : 1;
    /*为置顶博文添加阴影效果*/
    -moz-box-shadow : 0 0 5px #ECC;
    -webkit-box-shadow : 0 0 5px #ECC;
    box-shadow : 0 0 5px #ECC;
    /*为置顶博文设计圆角效果*/
    -moz-border-radius : 5px;
    -webkit-border-radius : 5px;
    border-radius : 5px;
    border : 1px solid #FCC;
    padding : 10px;
}
#content .topgroup h2 { margin-top : 0; }

```

定义分页导航弹性布局，其包含的导航按钮水平显示，并显示在博文列表框的底部

第五步，设置任务栏区域的样式。

```

#sidebar { /*任务栏样式*/
    width : 180px; /*设置任务栏的宽度为固定值*/
    padding : 10px;
    border-right : 1px solid #CCC;
}
#sidebar div:first-of-type h2:first-of-type { margin-top : 0; }
#sidebar ul { /*定义任务栏的列表样式*/

```

```
/*定义弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*设置任务栏列表项垂直分布显示*/
-moz-box-orient : vertical;
-webkit-box-orient : vertical;
box-orient : vertical;
/*按正常顺序显示*/
-moz-box-direction : normal;
-webkit-box-direction : normal;
box-direction : normal;
}
#sidebar .topgroup { /*任务栏列表中的置顶样式类*/
/*在所在盒子内第一个显示*/
-moz-box-ordinal-group : 1;
-webkit-box-ordinal-group : 1;
box-ordinal-group : 1;
font-weight : bold;
}
#sidebar li { /*定义列表项样式*/
/*普通列表项在后面按顺序显示*/
-moz-box-ordinal-group : 2;
-webkit-box-ordinal-group : 2;
box-ordinal-group : 2;
}
#sidebar form {
/*定义弹性盒子*/
display : -moz-box;
display : -webkit-box;
display : box;
/*水平显示表单域*/
-moz-box-orient : horizontal;
-webkit-box-orient : horizontal;
box-orient : horizontal;
}
#sidebar form input[type="text"] { /*定义输入文本框挤满富余空间*/
-moz-box-flex : 1;
-webkit-box-flex : 1;
box-flex : 1;
}
```

第5章

完善的盒模型和 UI 设计

在上一章中，我们介绍了 CSS 3 新增的弹性盒模型的概念及其应用。弹性盒模型是对 CSS 2.1 中盒模型的进一步补充，它试图从布局的角度解决网页设计师经常会感到棘手的浮动布局问题。当然，CSS 3 还对原来的盒模型功能进行了完善。例如，增强了元素边框和背景样式的控制能力 (<http://www.w3.org/TR/css3-background/>)，新增了不少 UI 特性 (<http://www.w3.org/TR/css3-ui/>)，用以解决用户界面设置问题。本章将就这些技术细节展开讲解，帮助读者设计更加灵活和美观的网页模块。

5.1 定义多色边框——border-color 属性

border-color 属性在 CSS 1 中就已经定义，使用它可以设置边框的颜色。不过，CSS 3 增强了这个属性的功能，使用它可以为边框设置更多的颜色，从而方便设计师设计渐变等绚丽的边框效果。border-color 属性的基本语法如表 5.1 所示。

表5.1 border-color属性的基本语法

语法项目	说明
值	<color>
初始值	无
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

<color> 可以为任意合法的颜色值或颜色值列表，支持不透明参数设置。

它可以与 CSS 2.1 中的 border-color 属性混合使用。当只为该属性设置一个颜色值时，则表示将边框设置为纯色；如果设置 n 个颜色值，且边框宽度为 n 像素，那么就可以在该边框上使用 n 种颜色，每种颜色显示 1 像素的宽度；如果边框宽度是 10 个像素，但是只声明了 5 种颜色，那么最后一种颜色将被用于显示剩下的宽度。

派生的子属性

为了避免与 border-color 属性的原功能（即定义边框颜色）发生冲突，CSS 3 在这个属性基础上派生了四个边框颜色属性。

- border-top-color：定义指定元素顶部边框的颜色。
- border-right-color：定义指定元素右侧边框的颜色。
- border-bottom-color：定义指定元素底部边框的颜色。
- border-left-color：定义指定元素左侧边框的颜色。

可以分别使用它们定义元素的边框颜色。为了方便理解，我们不妨先看一个简单的示例，示例演示效果如图 5.1 所示。

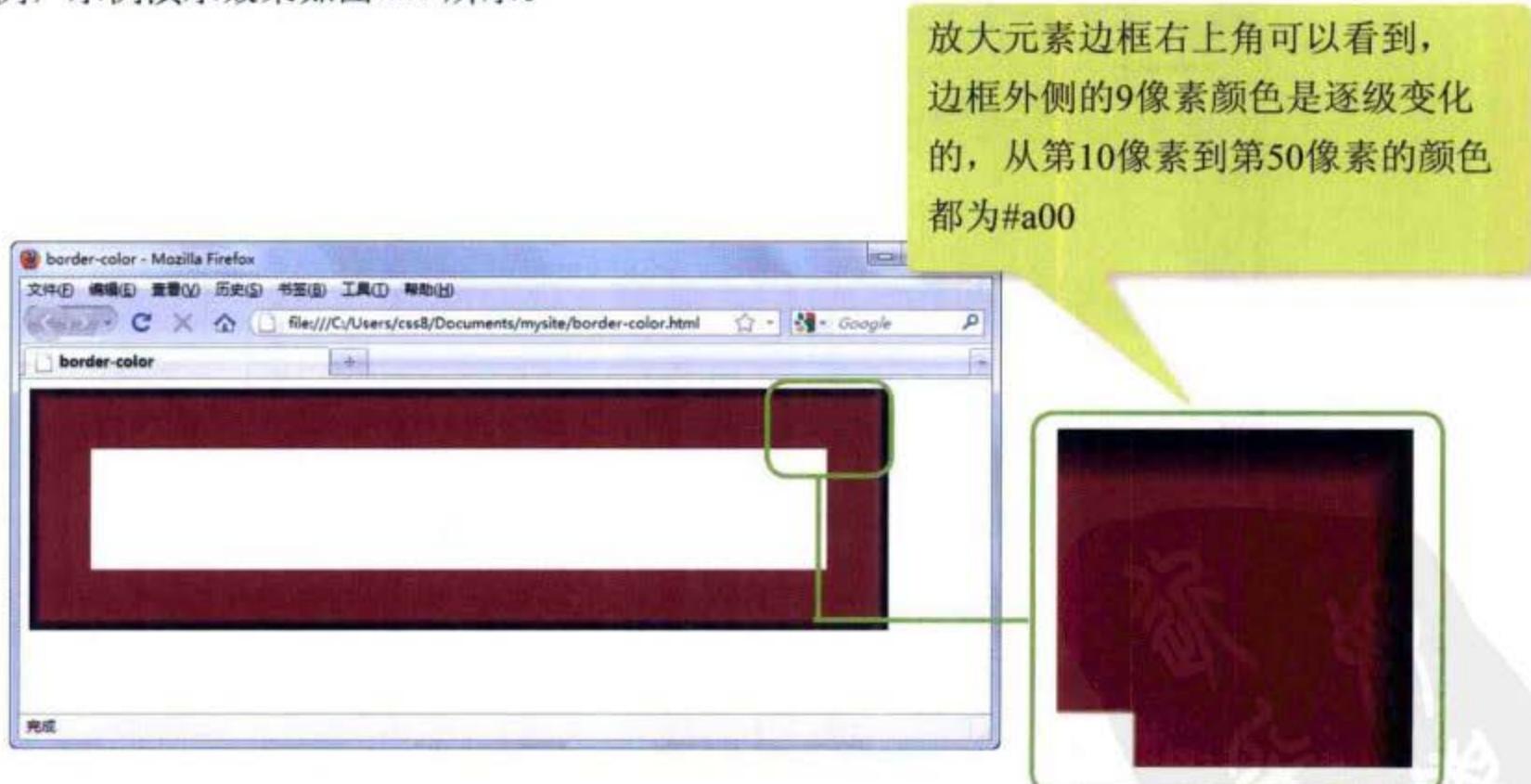


图 5.1 在 Firefox 3.6 中的演示效果

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

```

<title>border-color</title>
<style type="text/css">
div {
    border: 50px solid #dedede;
    height: 100px;
    width: 600px;
    /*兼容Mozilla Gecko引擎*/
    -moz-border-bottom-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    -moz-border-top-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    -moz-border-left-colors: #100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    -moz-border-right-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    /*标准用法*/
    border-bottom-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    border-top-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    border-left-colors: #100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
    border-right-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00;
}
</style>
</head><body>
<div></div>
</body></html>

```

但是，如果直接在 border-color 属性中定义，就会引发歧义，会导致浏览器无法正确解析这项声明，最终显示为 border: 50px solid #dedede; 声明的边框颜色，如图 5.2 所示。

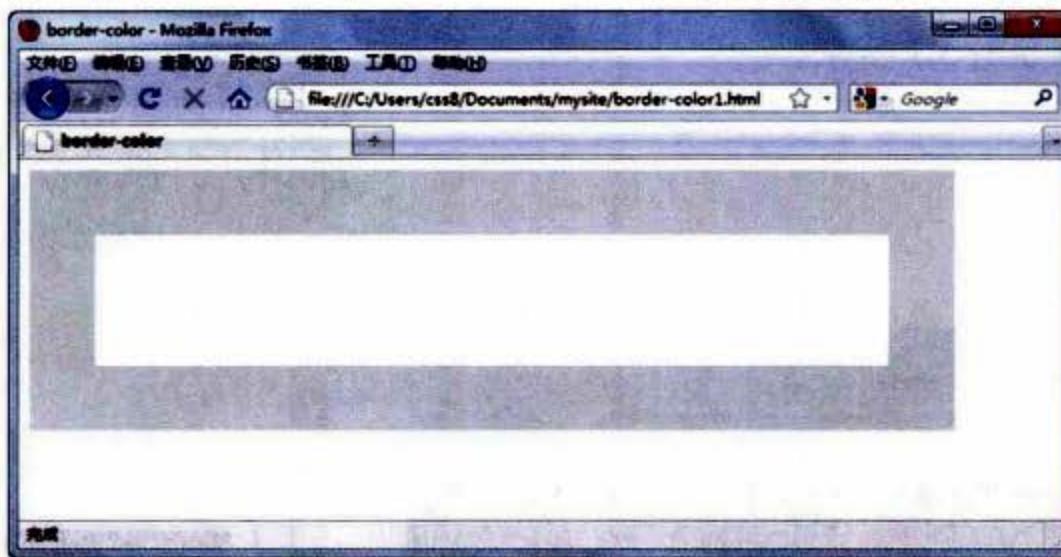


图 5.2 无效的border-color属性用法

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>border-color</title>
<style type="text/css">
div {
    border: 50px solid #dedede;
    height: 100px;
    width: 600px;
    /*兼容Mozilla Gecko引擎*/
    -moz-border-colors:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00 #100 #200 #300
#400 #500 #600 #700 #800 #900 #a00 #100 #200

```

```
#300 #400 #500 #600 #700 #800 #900 #a00;
/*标准用法*/
border-color:#100 #200 #300 #400 #500 #600 #700 #800 #900 #a00 #100 #200 #300 #400
#500 #600 #700 #800 #900 #a00 #100 #200 #300 #400 #500 #600 #700 #800 #900 #a00 #100 #200 #300
#400 #500 #600 #700 #800 #900 #a00;
}
</style>
</head><body>
<div></div>
</body></html>
```

浏览器兼容性检测

CSS 3 对于 border-color 增强的功能并没有得到各大主流浏览器的支持，目前仅有 Mozilla Gecko 引擎支持 -webkit-border-color 私有属性，且该属性只能够使用间接方法实现。各主流浏览器对 border-color 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：设计立体边框

在这个案例中，我们将借助 border-color 属性模拟立体效果的边框，效果如图 5.3 所示。

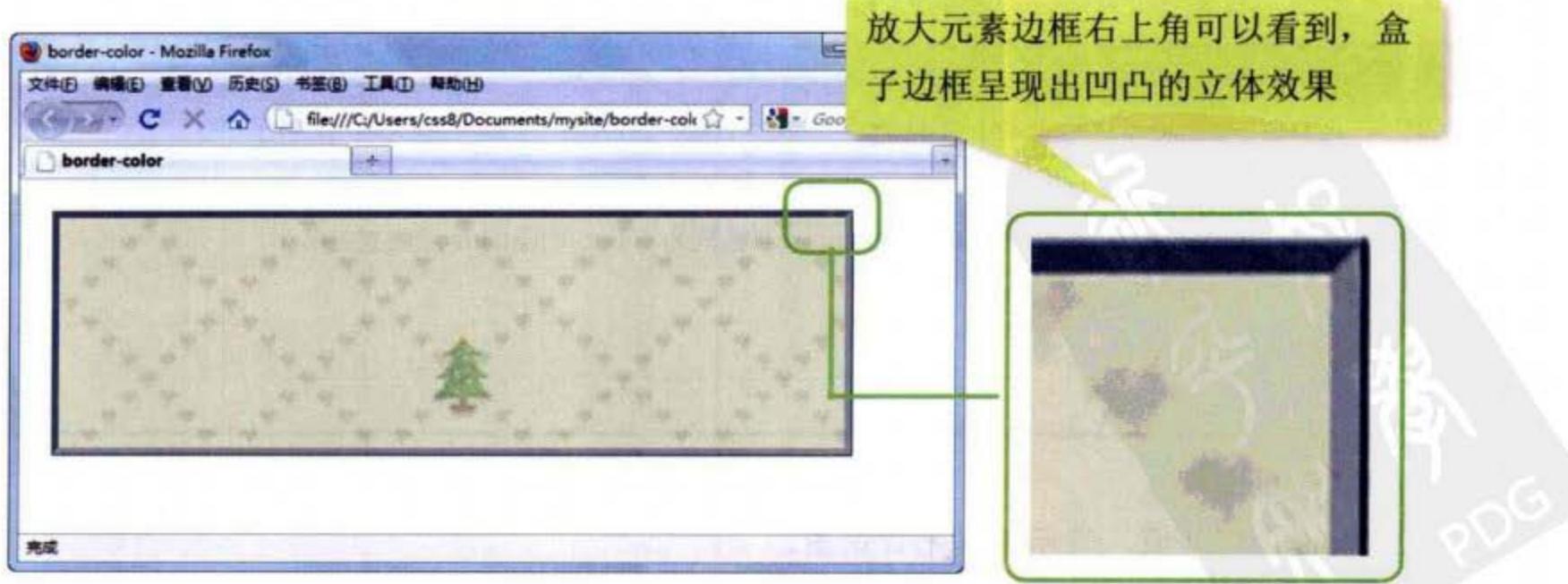


图5.3 在Firefox 3.6中的演示效果

CSS 设计思路

使用深色和浅色交错设计，营造出凹凸的立体效果。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>border-color</title>
<style type="text/css">
div {
    border: 2px solid #dedede;
    height: 60px;
    width: 200px;
    background:url(images/001.gif);
    -moz-border-right-colors:#333 #aaa;
    -moz-border-bottom-colors:#333 #aaa;
    -moz-border-top-colors:#aaa #666;
    -moz-border-left-colors:#aaa #666;
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

5.2 定义边框背景图——border-image 属性

一直以来，CSS 并不太重视边框背景的样式问题。在 CSS 3 中新增了 border-image 属性，该属性能够模拟 background-image 属性功能，不过它的功能更加强大。该属性的基本语法如表 5.2 所示。

表 5.2 border-image 属性的基本语法

语法项目	说明
值	none <image> [<number> <percentage>] {1,4} [/ <border-width> {1,4}]
初始值	none
适用于	所有元素（除 border-collapse 属性值为 collapse 的 table 元素之外）
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- none：默认值，表示边框无背景图。
- <image>：使用绝对或相对 URL 地址指定边框的背景图像。
- <number>：设置边框宽度或者边框背景图像大小，使用固定像素值表示。

<percentage>：设置边框背景图像大小，使用百分比表示。

注意，当table的border-collapse属性设置为collapse时，border-image属性无效。

派生的子属性

为了方便设计师更灵活地定义边框的背景图像，CSS 3 允许 border-image 属性复合定义边框背景样式，同时还派生了众多子属性。一方面，CSS 3 将 border-image 分成了 8 部分，使用 8 个子属性分别定义特定方位上边框的背景图像。

- border-top-image：定义顶部边框背景图像。
- border-right-image：定义右侧边框背景图像。
- border-bottom-image：定义底部边框背景图像。
- border-left-image：定义左侧边框背景图像。
- border-top-left-image：定义左上角边框背景图像。
- border-top-right-image：定义右上角边框背景图像。
- border-bottom-left-image：定义左下角边框背景图像。
- border-bottom-right-image：定义右下角边框背景图像。

另外，根据边框背景图像的处理功能，border-image 属性还派生了下面几个属性。

- border-image-source：定义边框的背景图像源，即图像 URL。
- border-image-slice：定义如何裁切背景图像，与背景图像的定位功能不同。
- border-image-repeat：定义边框背景图像的重复性。
- border-image-width：定义边框背景图像的显示大小（即边框显示大小）。虽然 W3C 定义了该属性，但浏览器还是习惯使用 border-width 实现相同的功能。
- border-image-outset：定义边框背景图像的偏移位置。

浏览器兼容性检测

border-image 是 CSS 3 新增的核心属性之一，也是非常实用的一个属性。在网页设计中，它的价值将会随着各主流浏览器的全面支持而不断放大。目前 Webkit 引擎支持 -webkit-border-image 私有属性，Mozilla Gecko 引擎支持 -moz-border-image 私有属性，Presto 引擎支持 -o-border-image 私有属性。IE 浏览器暂时不支持 border-image 属性，也没有定义支持 -ms-border-image 私有属性。各浏览器对 border-image 属性的支持情况如下。

				
<input checked="" type="checkbox"/> IE 6	<input checked="" type="checkbox"/> Firefox 1.5	<input checked="" type="checkbox"/> Opera 9.0	<input checked="" type="checkbox"/> Safari 3.0	<input checked="" type="checkbox"/> Chrome 1.0.x
<input checked="" type="checkbox"/> IE 7	<input checked="" type="checkbox"/> Firefox 2.0	<input checked="" type="checkbox"/> Opera 9.6	<input checked="" type="checkbox"/> Safari 4.0	<input checked="" type="checkbox"/> Chrome 2.0.x
<input checked="" type="checkbox"/> IE 8	<input checked="" type="checkbox"/> Firefox 3.0	<input checked="" type="checkbox"/> Opera 10.0		<input checked="" type="checkbox"/> Chrome 3.0.x
<input checked="" type="checkbox"/> IE 9	<input checked="" type="checkbox"/> Firefox 3.5	<input checked="" type="checkbox"/> Opera 10.5		<input checked="" type="checkbox"/> Chrome 4.0.x

关于边框背景的用法解析

border-image 的用法相对复杂，因为它是一个复合属性，而且包含多个派生子属性。如果读者熟悉 CSS 2 中的 background 属性，那么应该能够有所体会。例如，background:url(01.jpg) center no-repeat; 样式就表示为指定元素设置背景图像 01.jpg，居中显示，并禁止平铺。这里涉及背景图像的三个特性：图像源、图像位置和图像重复性。

border-image 属性与 background-image 属性的用法比较相似，它也包括图像源、剪裁位置和重复性。例如，border-image:url(01.jpg) 50 no-repeat; 样式就表示设置边框背景图像为 01.jpg，剪裁位置为 50px，禁止重复。

下面将针对边框背景的图像源、剪裁和重复性进行讲解。

□ 边框图像源 (border-image-source)。

与 background-image 属性用法相同，border-image 属性使用 url() 调用背景图像，图像是可以是相对路径或是绝对路径字符串，也可以不使用图像，即 border-image:none;。

□ 边框图像切片 (border-image-slice)。

border-image-slice 属性值在用法上比较特殊，说明如下。

1) 属性值没有单位。默认单位为像素。

2) 支持百分比值。百分比值总是相对于边框图像而言的。例如，边框图像大小为 400px*300px，则当 border-image-slice 属性值为 20% 时，实际的效果就是剪裁了图像的 60px、80px、60px、80px 的四边大小。

3) 剪裁特性。如果读者熟悉 CSS 的 clip 属性 (clip:rect(auto, auto, auto, auto))，则理解起来就会比较轻松。clip 是 CSS 专用剪裁属性，而 border-image-slice 属性虽然在语义上不是剪裁，但是从实现效果上来分析，与剪裁工具类似，它把边框图像四分五裂，再重新安置、变形。

4) border-image-slice 属性值包含 4 个参数。它遵循 CSS 方位规则（如 margin、padding 或 border 等属性的赋值），按照上、右、下、左的顺时针方向赋值剪裁。

使用案例演示背景裁切的方法

例如，假设为元素边框定义背景图像为 images/border1.png，然后设置 border-image-slice 属性值为 (27,27,27,27)，该属性值可以简写为 27。整个示例的代码如下，则页面浏览效果如图 5.4 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>border-image</title>
```

```

<style type="text/css">
div {
    height:160px;
    border-width:27px;
    /* 设置边框背景图像 */
    -webkit-border-image: url(images/border1.png) 27; /*兼容webkit引擎*/
    -moz-border-image: url(images/border1.png) 27;      /*兼容gecko引擎*/
    -o-border-image: url(images/border1.png) 27;        /*兼容presto引擎*/
    border-image: url(images/border1.png) 27;           /*兼容标准用法*/
}
</style>
</head><body>
<div></div>
</body></html>

```

定义边框背景图像

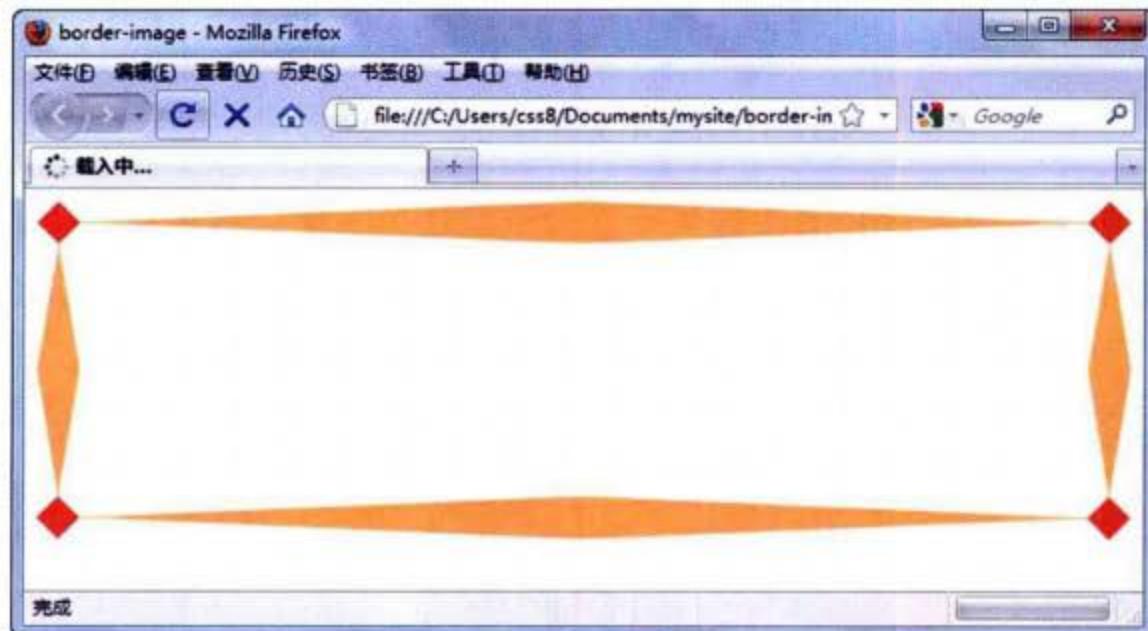


图5.4 在Firefox 3.6版本浏览器下的演示效果

在上面的示例中，使用了一个 $81\text{px} \times 81\text{px}$ 大小的背景图，如图 5.5 所示，这个正方形的背景图被等分成了 9 个方块，每个方块的高和宽都是 $27\text{px} \times 27\text{px}$ 大小。

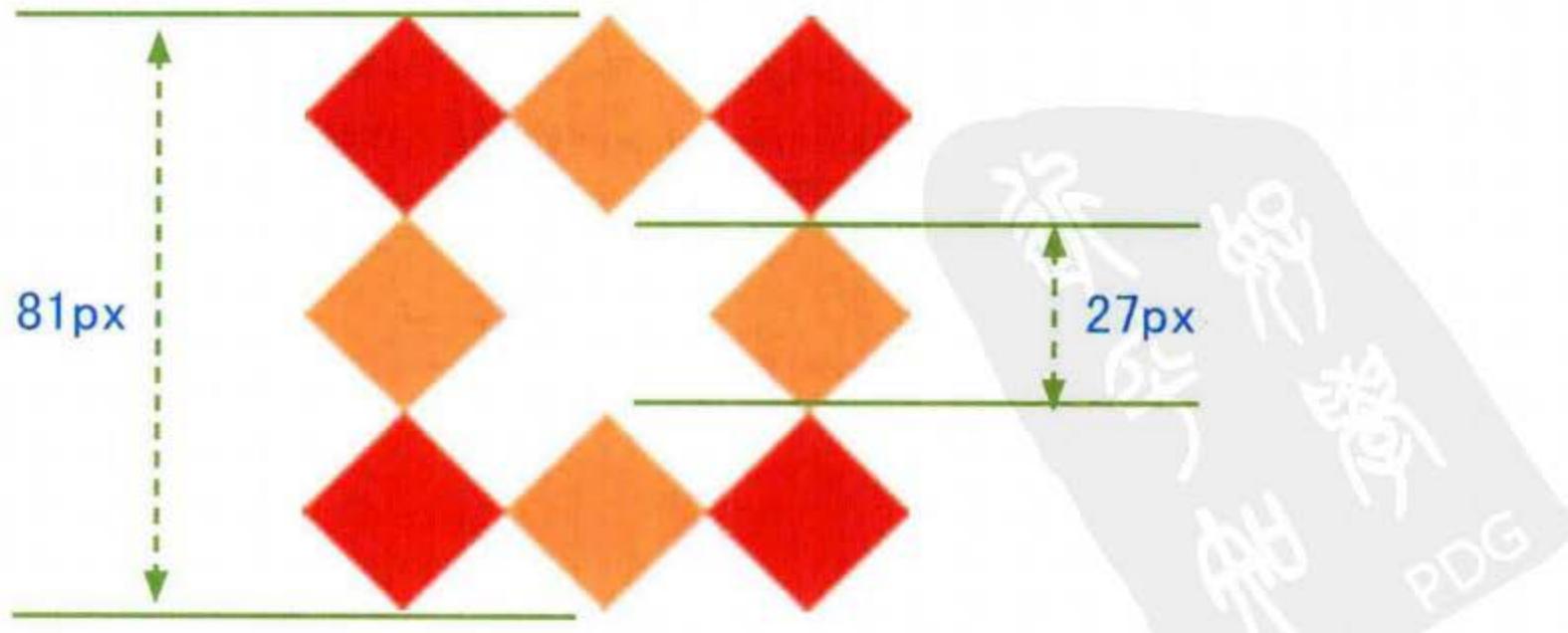
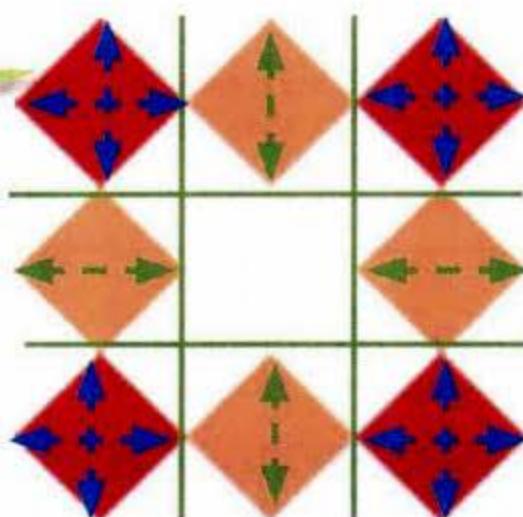


图5.5 设计的正方形背景图像

当声明 border-image-slice 属性值为 (27,27,27,27) 时，则元素四边边框的背景截图示意图如图 5.6 所示。

第一个参数值表示从上向下裁切背景图像，然后显示在顶边。而第三个参数值正好相反，从下向上裁切背景图像，然后显示在底边



第二个参数值表示从右向左裁切背景图像，然后显示在右边。而第四个参数值正好相反，从左向右裁切背景图像，然后显示在左边

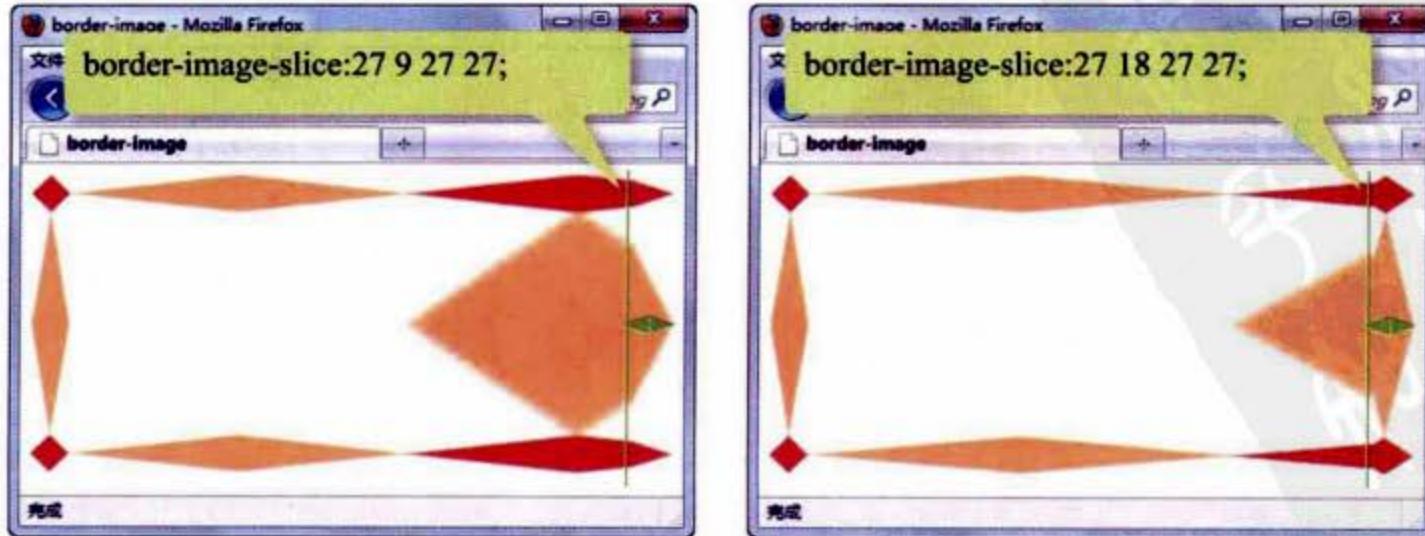
图 5.6 正方形背景图的平铺效果

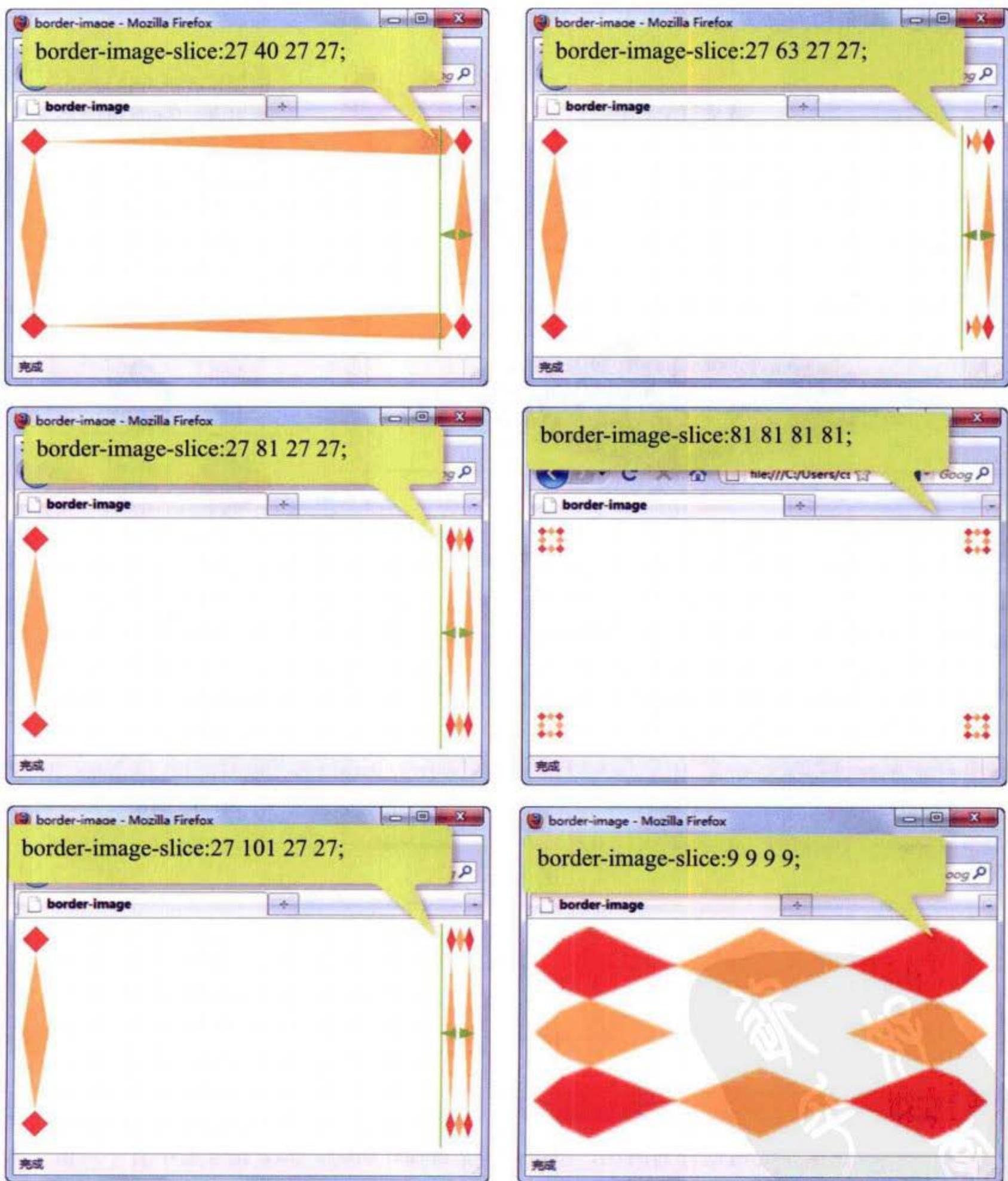
首先，读者应该明白，背景图像被四个参数值裁切为 9 块，然后根据边框的大小进行自适应显示。例如，当分别设置边框为不同大小时，则显示效果除了粗细之外，其他都是完全相同的。如图 5.7 所示。



图 5.7 当边框粗细不同时边框背景效果的比较

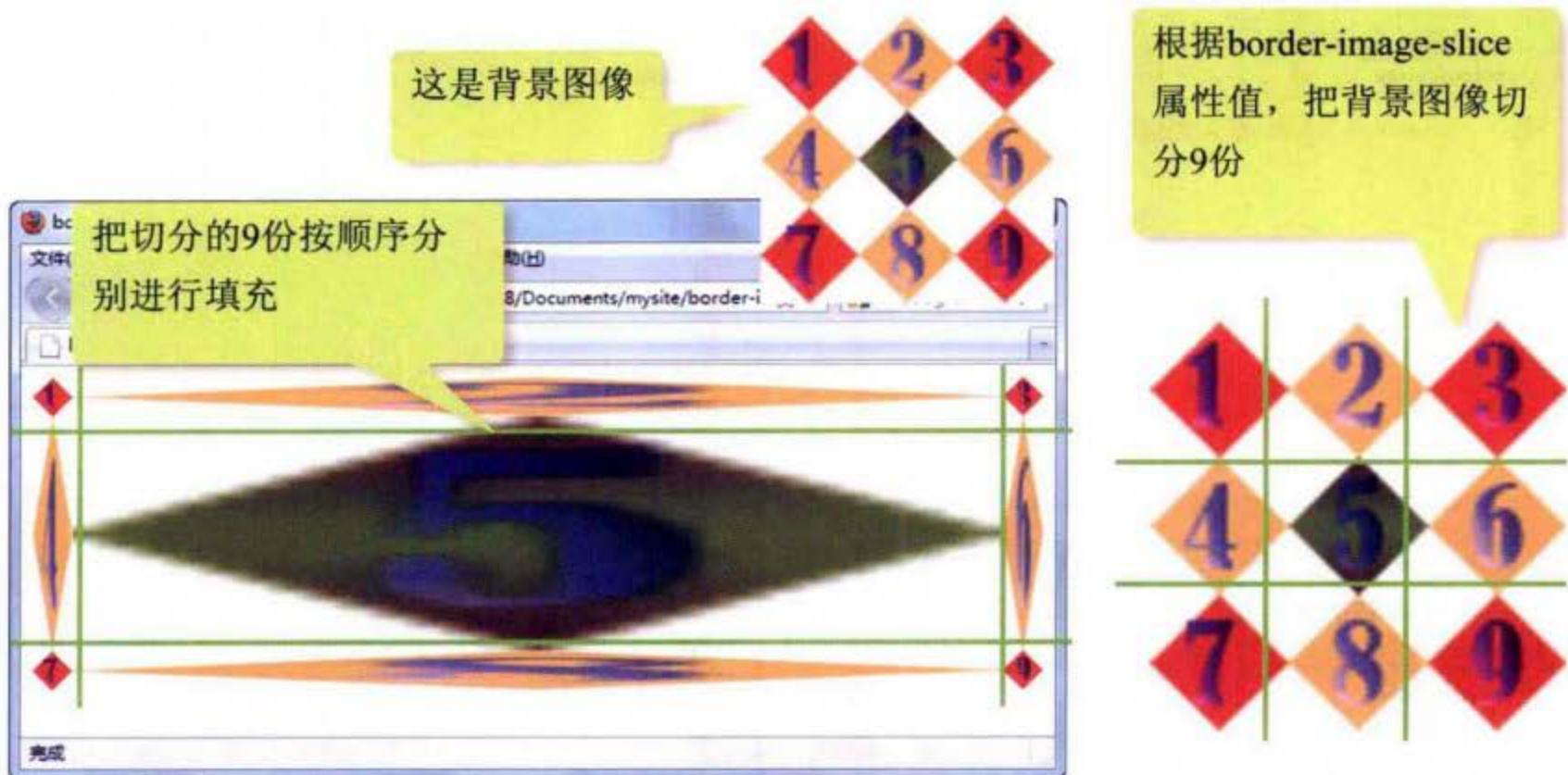
然后，我们重点研究当 border-image-slice 属性取值不同时，分别会发生什么样的变化。



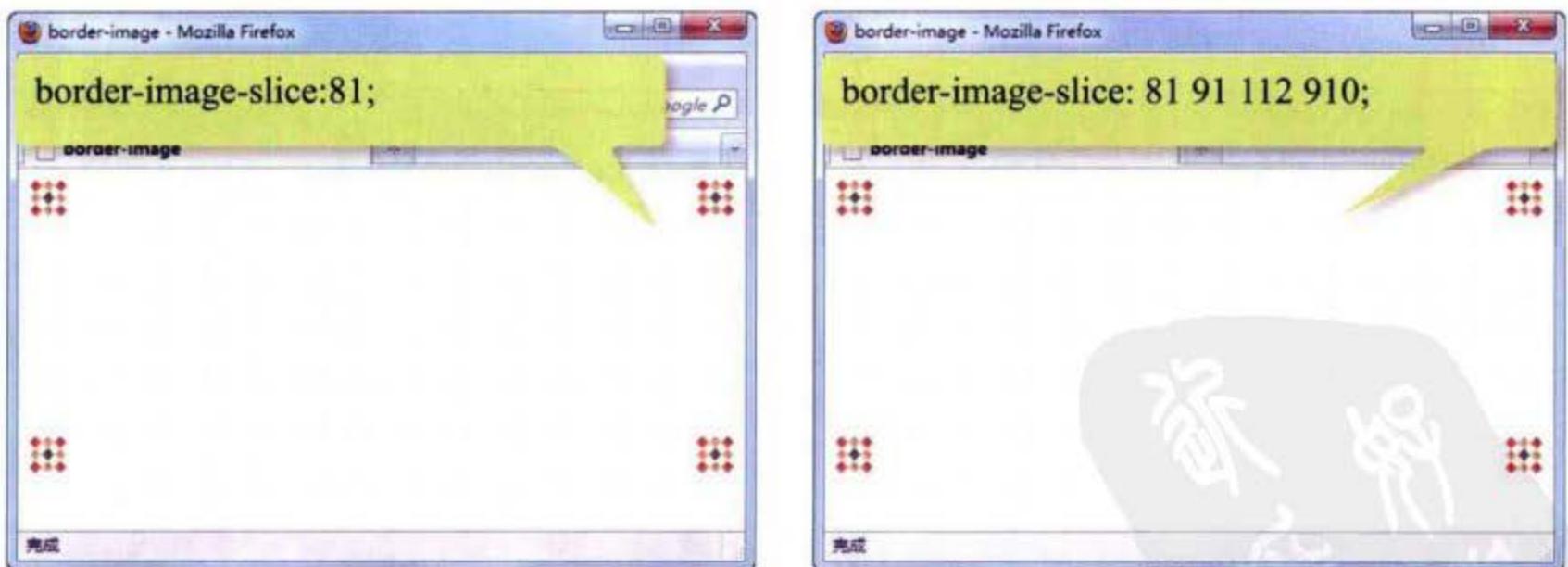


使用 `border-image-slice` 属性定位边框背景图像看起来确实很复杂，但是如果根据九宫格切割法进行分析，问题就变得非常明了了。简单来说，`border-image` 属性能够根据 `border-image-slice` 属性值把背景图像切分为 9 块，然后分别把这 9 块图像切片按顺序分别填充到边

框的四边、四角和内容区域。

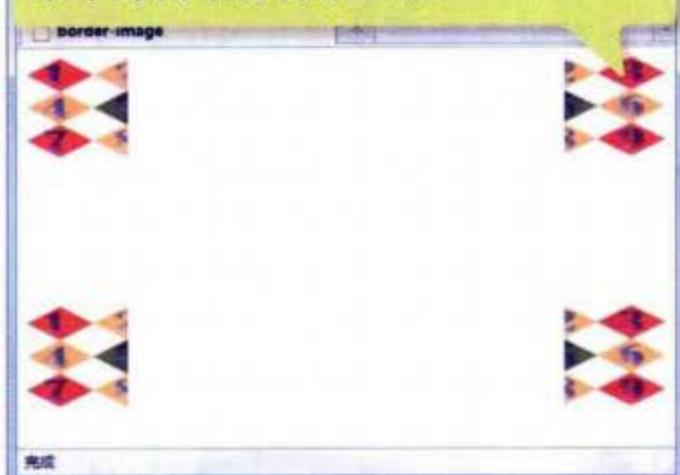


如果背景图像总共被切分为一块，也就是说，`border-image-slice` 属性值都大于或等于背景图像的尺寸，即第一、三个参数值都大于或等于背景图像的高度，第二、四个参数值大于或等于背景图像的宽度，那么背景图像将被缩放填充到边框的四个角中，如下图所示。

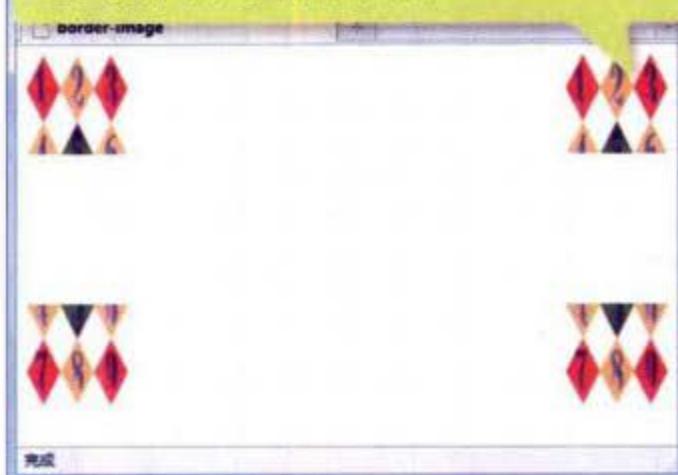


如果背景图像总共被切分为两块，也就是说，`border-image-slice` 属性值中有一对值（水平上或垂直上）大于或等于背景图像的尺寸，即第一、三个参数值（或者第二、四个参数值）大于或等于背景图像的高度（或者宽度），而另一对值之和等于背景图像的宽度（或者高度），那么背景图像将被切分为两半，并按顺序分别缩放填充到边框的四个角中，如下图所示（为了方便观察，案例放大了边框的宽度）。

`border-image-slice: 81 40 112 41;`在水平方向平分背景图像

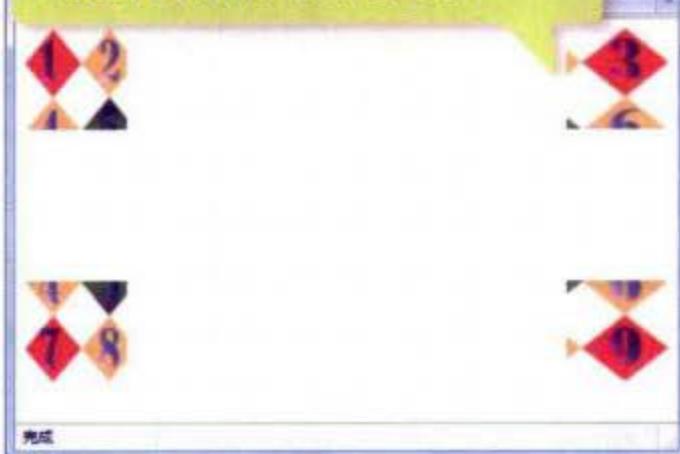


`border-image-slice: 40 81 41 112;`在垂直方向平分背景图像

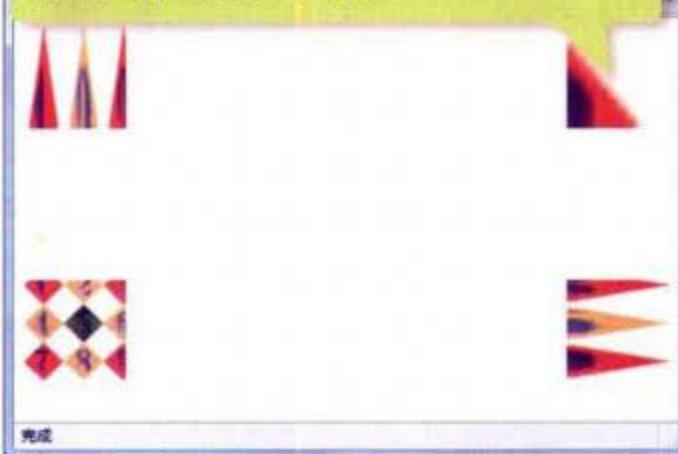


如果背景图像总共被切分为四块，也就是说，`border-image-slice` 属性值中每一对值之和等于背景图像的宽度或者高度，那么背景图像将被切分为四半，并按顺序分别缩放填充到边框的四个角中，如下图所示。

`border-image-slice: 40 32 41 49;`把背景图像切分为4块



`border-image-slice: 10 12 71 69;`把背景图像切分为4块



还有一种特殊的四块切分法，即 `border-image-slice` 属性值中有一对值或者两对值之和大于背景图像的宽度或者高度，但是参数值都不大于背景图像的宽度或者高度，那么背景图像将被切分为四半，这四半背景图像切片存在重叠部分，这时 `border-image` 仍然按顺序分别缩放它们，并填充到边框的四个角中，如下图所示。

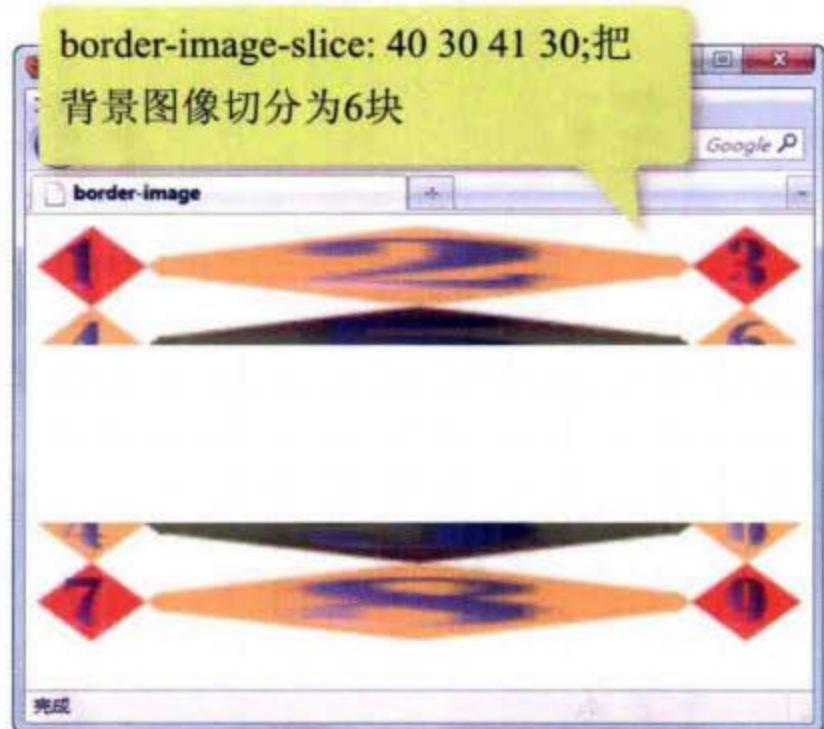
`border-image-slice: 50 60 50 60;`
错位重叠四分背景图像



`border-image-slice: 40 42 42 40;`
错位重叠四分背景图像



如果背景图像总共被切分为六块，也就是说，`border-image-slice` 属性值中有一对值（水平上或垂直上）之和小于背景图像的尺寸，即第一、三个参数值之和（或者第二、四个参数值之和）小于背景图像的高度（或者宽度），而另一对值之和等于背景图像的宽度（或者高度），那么背景图像将被切分为六半，并按顺序分别缩放填充到边框的四个角和对应两条边中，如下图所示（为了方便观察，案例放大了边框的宽度）。



也可以把背景图像切分为三块，即 `border-image-slice` 属性值中有一对值都不大于背景图像的宽度或者高度，而另一对值之和小于背景图像的高度或者宽度，那么背景图像将被切分为三半，这时 `border-image` 按顺序分别缩放它们，并填充到边框一边的两个角和边中，然后重复相同的方法，填充另一边的两角和边，如下图所示。



最后把背景图像切分为九块，即 border-image-slice 属性值中每一对值之和都小于背景图像的宽度或者高度，那么背景图像将被切分为九分，这时 border-image 按顺序分别缩放它们，并填充到元素的四边、四角和内容区域，如右上图所示。

如果 border-image-slice 属性值单位为百分比，则根据背景图像的大小进行百分比裁切，然后根据前面分析的规律填充边框。例如，四分背景图像，各填四角效果，效果如右中图所示。

```
<style type="text/css">
div {
    height:120px;
    border-width:54px;
    -moz-border-image: url(images/border2.png) 66%;
    -webkit-border-image: url(images/border2.png) 66%;
    -o-border-image: url(images/border2.png) 66%;
    border-image: url(images/border2.png) 66%;
}
</style>
```

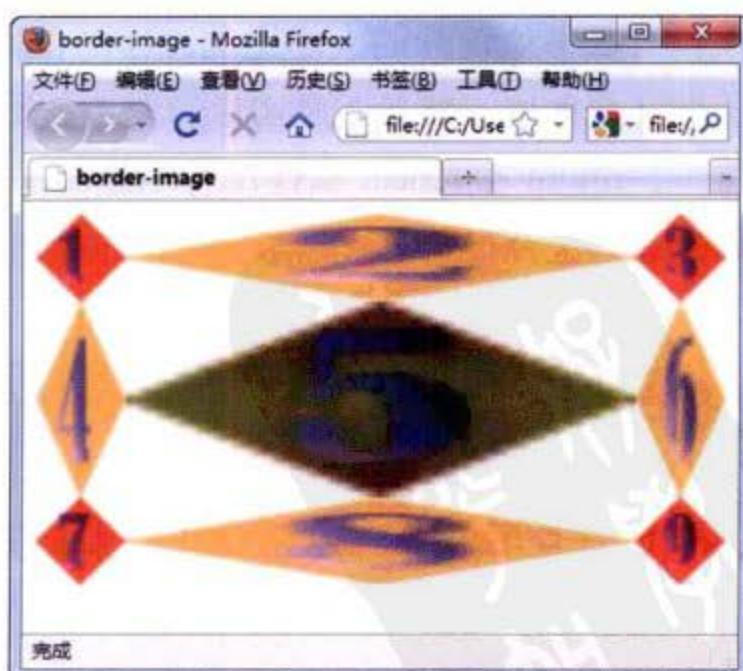
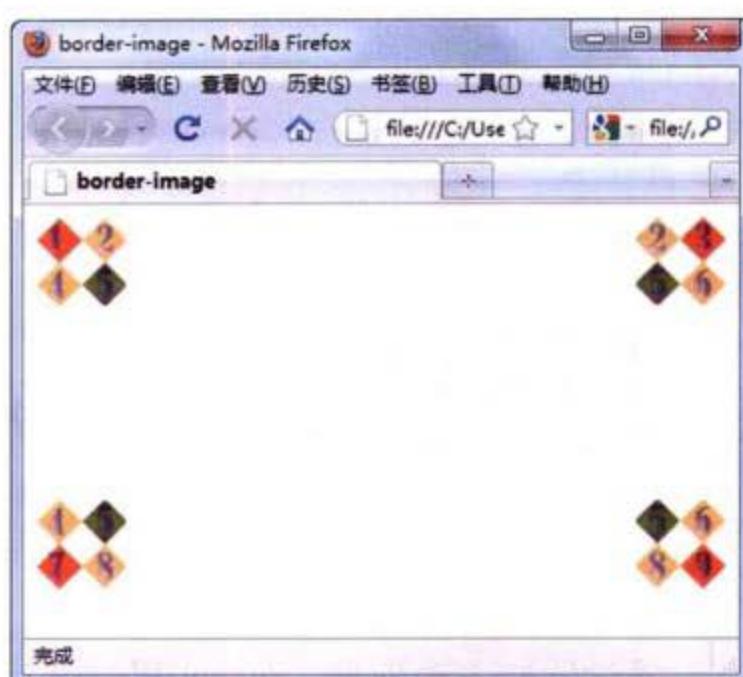
再如，九分背景图像，分别填充元素九宫格位置，效果如右下图所示。

```
<style type="text/css">
div {
    height:120px;
    border-width:54px;
    -moz-border-image: url(images/border2.png)
66%;
    -webkit-border-image: url(images/border2.
png) 66%;
    -o-border-image: url(images/border2.png)
66%;
    border-image: url(images/border2.png) 66%;
}
</style>
```

关于背景图像的重复性解析

border-image-repeat 属性包含三个值，简单说明如下。

- stretch：拉伸，为默认值。
- repeat：重复。



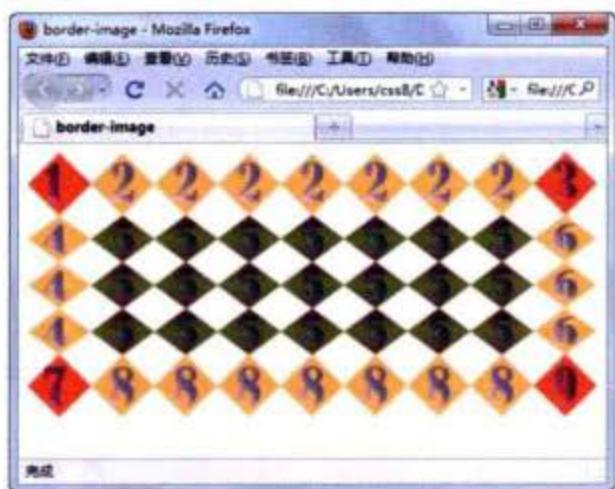
PDF

□ round : 平铺。

这些属性值与 Windows 桌面背景图像的显示方式完全相同。下面我们演示当边框背景图像定义为重复和平铺时的效果。



```
<style type="text/css">
div {
    height:120px;
    border-width:54px;
    -moz-border-image: url(images/border2.png) 33%
repeat;
    -webkit-border-image: url(images/border2.png) 33%
repeat;
    -o-border-image: url(images/border2.png) 33%
repeat;
    border-image: url(images/border2.png) 33% repeat;
}
</style>
```



```
<style type="text/css">
div {
    height:120px;
    border-width:54px;
    -moz-border-image: url(images/border2.png) 33%
round;
    -webkit-border-image: url(images/border2.png) 33%
round;
    -o-border-image: url(images/border2.png) 33%
round;
    border-image: url(images/border2.png) 33% round;
}
</style>
```

通过比较不难发现，repeat 属性值是从中间开始，不断重复平铺到四周，在平铺过程中保持背景图像切片的大小比例，这样在边缘区域可能会被部分隐藏显示。而 round 属性值会压缩（或伸展）背景图像切片的大小，使其正好在区域内显示。注意，在 Webkit 引擎的浏览器下，round 和 repeat 属性没有明显区分，所以在不同浏览器下会看到不同的显示效果。当然，这些区分仅是解析细节，可以忽略不计。

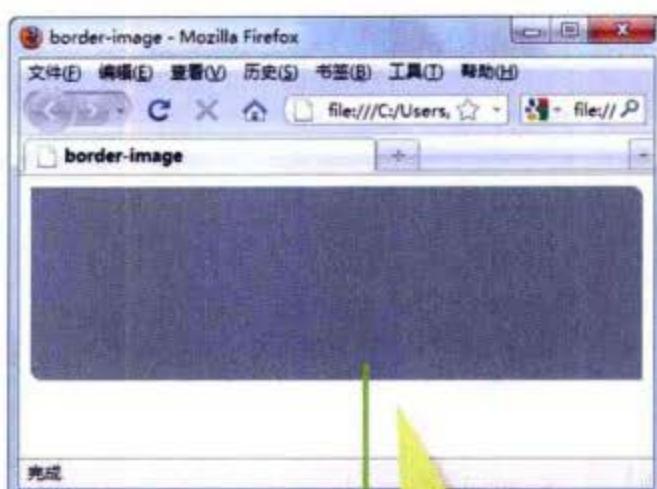
实战体验：设计各种精巧的边框

border-image 是一个非常实用的属性，它拓展了设计师的设计灵感，抛弃了传统上借助背景图像实现边角设计的笨拙做法，提高了网页传输速度，降低了前期劳动量。下面这些案例虽然很小，但是它能够帮助读者拓展设计思路、灵活应用 border-image 属性。

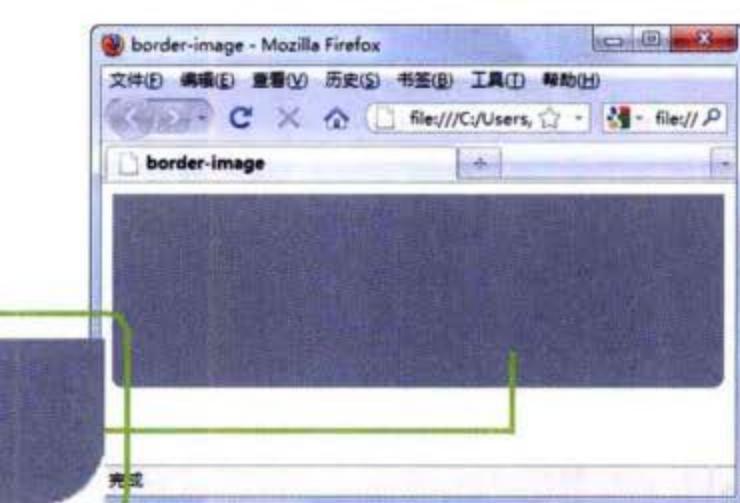
□ 设计局部或者全部圆角版块

```
<style type="text/css">
div {
    height:120px;
    border-width:10px;
    -moz-border-image: url(images/r1.png) 20;
    -webkit-border-image: url(images/r1.png) 20;
    -o-border-image: url(images/r1.png) 20;
    border-image: url(images/r1.png) 20;
}
</style>

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



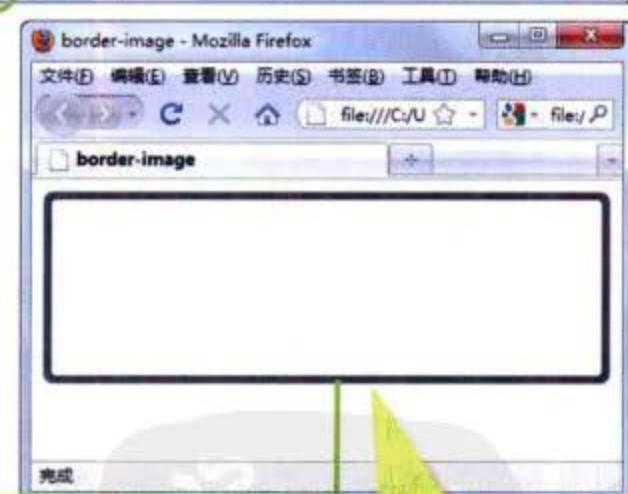
```
<style type="text/css">
div {
    height:120px;
    border-width:10px;
    -moz-border-image: url(images/r2.png) 20;
    -webkit-border-image: url(images/r2.png) 20;
    -o-border-image: url(images/r2.png) 20;
    border-image: url(images/r2.png) 20;
}
</style>
```



□ 设计圆环边框版块

```
<style type="text/css">
div {
    height:120px;
    border-width:10px;
    -moz-border-image: url(images/r3.png) 20;
    -webkit-border-image: url(images/r3.png) 20;
    -o-border-image: url(images/r3.png) 20;
    border-image: url(images/r3.png) 20;
}
</style>

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



42px*42px, 圆环角为20px

□ 设计阴影效果

```
<style type="text/css">
img {
    height:400px;
    border-width:2px 5px 6px 2px;
    -moz-border-image: url(images/r4.png) 2 5 6 2;
}
```

```

    -webkit-border-image: url(images/r4.png) 2 5 6 2;
    -o-border-image: url(images/r4.png) 2 5 6 2;
    border-image: url(images/r4.png) 2 5 6 2;
}
</style>

```

```

<body>

</body>

```

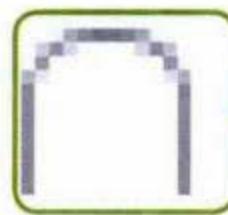
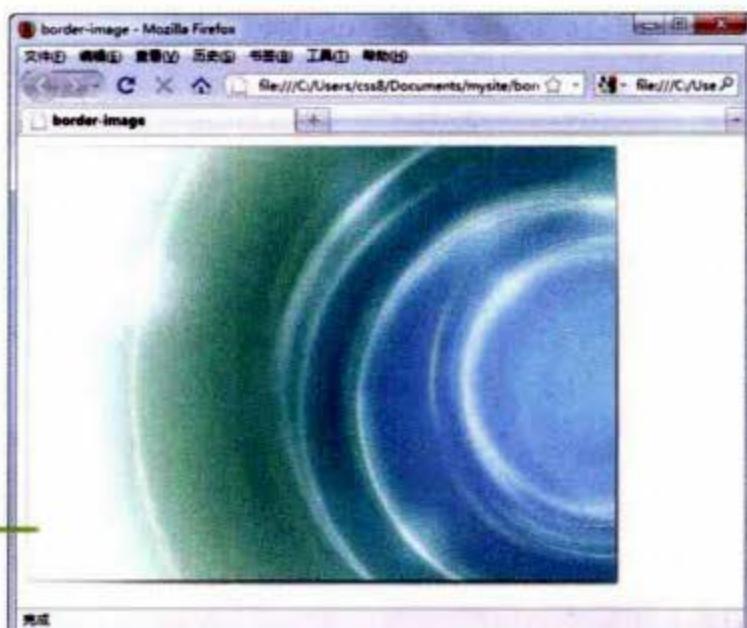
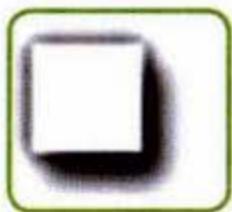
□ 设计选项卡

```

<style type="text/css">
div {
    width:200px;
    height:20px;
    padding:10px 0;
    text-align:center;
    border-width:5px 5px 0px;
    -moz-border-image: url(images/r5.png) 5 5 0;
    -webkit-border-image: url(images/r5.png) 5 5 0;
    -o-border-image: url(images/r5.png) 5 5 0;
    border-image: url(images/r5.png) 5 5 0;
}
</style>

<body>
<div>选项卡</div>
</body>

```



注意，如果 `border-image-slice` 属性值包含三个参数，则第一个参数表示顶部裁切值，第二个参数表示左右两侧裁切值，第三个参数表示底部裁切值。例如，`border-image-slice:5 5 0;` 就等同于 `border-image-slice:5 5 0 5;`，它表示把边框背景图像分成六块（底部没有裁切），然后分别填充到左上角、顶边、右上角、左边、中间内容区域和右边。

当然，`border-image` 属性的应用还是非常灵活的，读者可以设计不同样式的背景图，然后设置不同的 `border-image-slice` 属性值，从而设计各种特殊边框样式。

5.3 设计圆角—`border-radius` 属性

圆角曾经让很多网页设计师又爱又恨，爱是因为圆角页面看起来更让人舒服，版块显得圆润而不失灵巧，但是为了设计圆角，设计师需要花费更多的时间去实现、兼容，以及灵活应用。实际上，W3C 早在 2002 年的 CSS 3 草案中就已经定义了 `border-radius`

属性，使用它可以设计以圆角样式显示的元素。`border-radius` 属性的基本语法如表 5.3 所示。

表5.3 `border-radius`属性的基本语法

语法项目	说明
值	<code>none <length>{1,4} [/ <length>{1,4}]?</code>
初始值	<code>none</code>
适用于	所有元素（除了 <code>border-collapse</code> 属性值为 <code>collapse</code> 的 <code>table</code> 元素）
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- `none`：默认值，表示元素没有圆角。
- `<length>`：由浮点数字和单位标识符组成的长度值，不可为负值。

派生的子属性

为了方便设计师更灵活地定义元素的四个顶角圆角，派生了 4 个子属性。

- `border-top-right-radius`：定义右上角的圆角。
- `border-bottom-right-radius`：定义右下角的圆角。
- `border-bottom-left-radius`：定义左下角的圆角。
- `border-top-left-radius`：定义左上角的圆角。

浏览器兼容性检测

目前，Webkit 引擎支持 `-webkit-border-radius` 私有属性，Mozilla Gecko 引擎支持 `-moz-border-radius` 私有属性，Presto 引擎和 IE 9 支持 `border-radius` 标准属性。IE 8 及其以前版本的浏览器暂时不支持 `border-radius` 属性。相应地，Webkit 和 Gecko 引擎还分别支持下面几个私有属性。

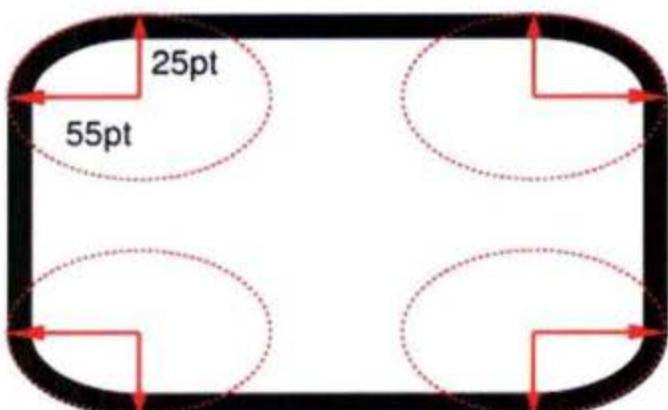
- `-moz-border-radius-bottomleft` 和 `-webkit-border-bottom-left-radius`。
- `-moz-border-radius-bottomright` 和 `-webkit-border-bottom-right-radius`。
- `-moz-border-radius-topleft` 和 `-webkit-border-top-left-radius`。
- `-moz-border-radius-topright` 和 `-webkit-border-top-right-radius`。

借助兼容方式，各主流浏览器对 `border-radius` 属性的兼容情况如下所示。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

关于圆角的用法解析

border-radius 属性可包含两个参数值：第一个值表示圆角的水平半径，第二个值表示圆角的垂直半径（如左图所示），两个参数值通过斜线分隔。如果仅包含一个参数值，则第二个值与第一个值相同，表示这个角就是一个 1/4 的圆。如果参数值中包含 0，则这个角就是矩形，不会显示为圆角。



例如，在下面这个示例中，给 border-radius 属性设置一个值，则圆角是一个 1/4 的圆，如图 5.8 所示。

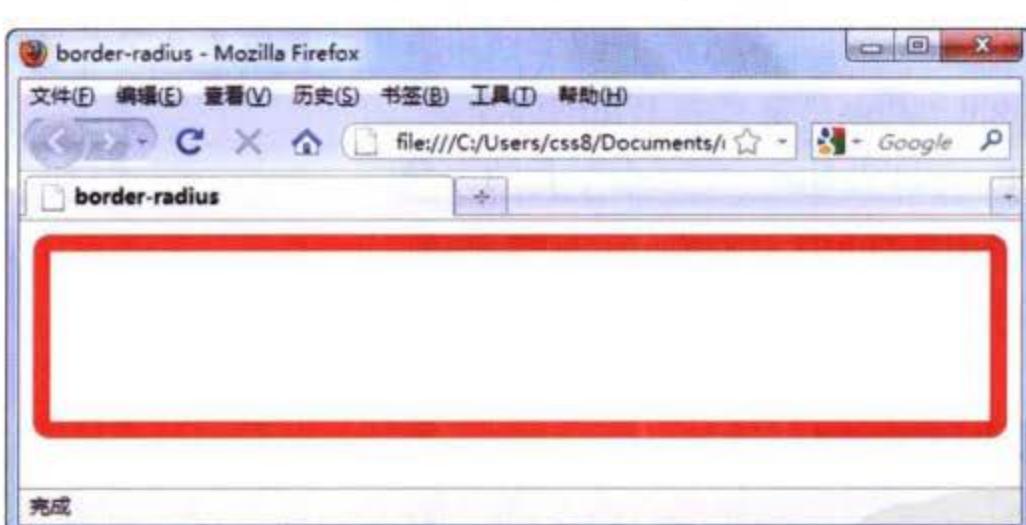


图 5.8 设计圆角边框效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>border-radius</title>
<style type="text/css">
div {
    height:100px;
    border:10px solid red;
}

```

```

-moz-border-radius:10px; /*兼容Gecko引擎*/
-webkit-border-radius:10px; /*兼容Webkit引擎*/
border-radius:10px; /*标准用法*/
}
</style>
</head>
<body>
<div></div>
</body>
</html>

```

如果为 border-radius 属性设置两个参数，则效果如下图所示。

```

<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:40px/20px;
    -webkit-border-radius:40px/20px;
    border-radius:40px/20px;
}
</style>

```

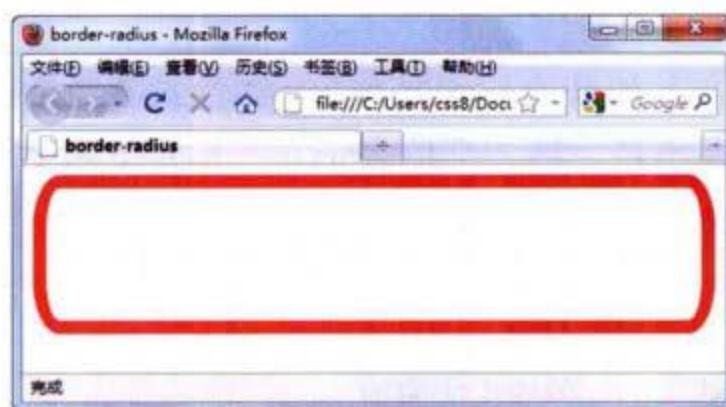


如果调整 border-radius 属性值中两个参数的位置，则效果如下图所示。

```

<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:20px/40px;
    -webkit-border-radius:20px/40px;
    border-radius:20px/40px;
}
</style>

```



当然我们也可以为元素的四个角定义不同半径的圆角，实现的方法有两种：

一种方法是利用 border-radius 属性，为其赋一组值。当为 border-radius 属性赋一组值时，将遵循 CSS 赋值规则，可以包含 2 个、3 个或者 4 个值集合。但是，此时无法使用斜杠方式定义圆角水平和垂直半径。

- 如果是四个值，则这四个值将按照 top-left、top-right、bottom-right、bottom-left 的顺序来设置。
- 如果 bottom-left 的值省略，那么它等于 top-right。
- 如果 bottom-right 的值省略，那么它等于 top-left。
- 如果 top-right 的值省略，那么它等于 top-left。

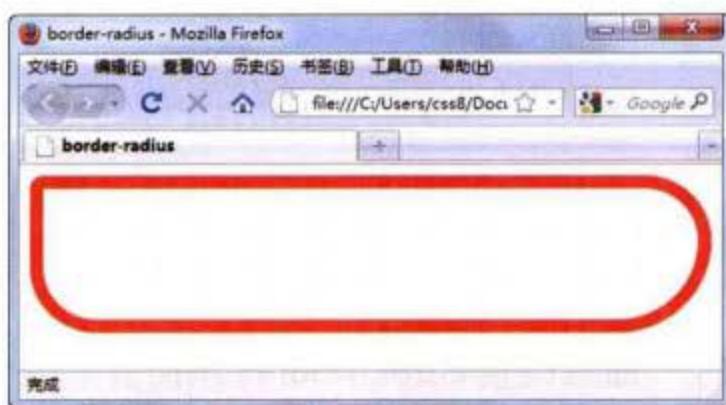
如果为 border-radius 属性设置 4 个值的集合参数，则每个值表示每个角的圆角半径，例如，下面的代码将定义不同角度的圆角半径，如下所示。

```
<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:10px 30px 50px 70px;
    -webkit-border-radius:10px 30px 50px 70px;
    border-radius:10px 30px 50px 70px;
}
</style>
```



如果为 border-radius 属性设置 3 个值的集合参数，则第一个值表示左上角的圆角半径，第二个值表示右上和左下两个角的圆角半径，第三个值表示右下角的圆角半径。例如，下面的代码将定义不同角度的圆角半径，如下所示。

```
<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:10px 50px 70px;
    -webkit-border-radius:10px 50px 70px;
    border-radius:10px 50px 70px;
}
</style>
```



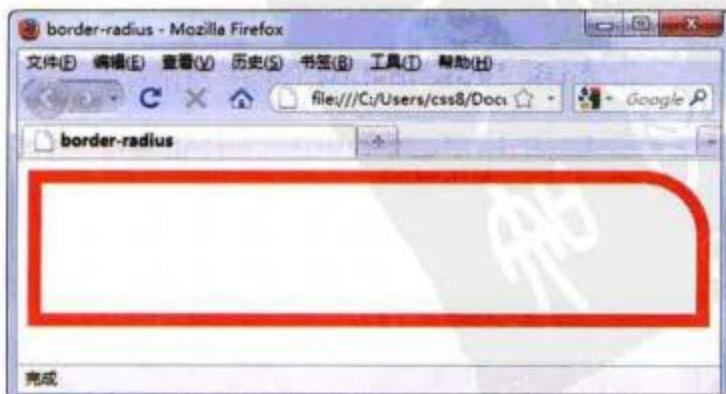
如果为 border-radius 属性设置 2 个值的集合参数，则第一个值表示左上角和右下角的圆角半径，第二个值表示右上和左下两个角的圆角半径。例如，下面的代码将定义不同角度的圆角半径，如下所示。

```
<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:10px 50px 70px;
    -webkit-border-radius:10px 50px 70px;
    border-radius:10px 50px 70px;
}
</style>
```



另一种方法是利用派生子属性进行定义，如 border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius、border-top-left-radius。注意，Gecko 和 Presto 引擎在写法上存在很大差异。例如，下面的代码定义 div 元素右上角为 50 像素的圆角。

```
<style type="text/css">
div {
    height:100px;
```



```

border:10px solid red;
-moz-border-radius-topright:50px;
-webkit-border-top-right-radius:50px;
border-top-right-radius:50px;
}
</style>

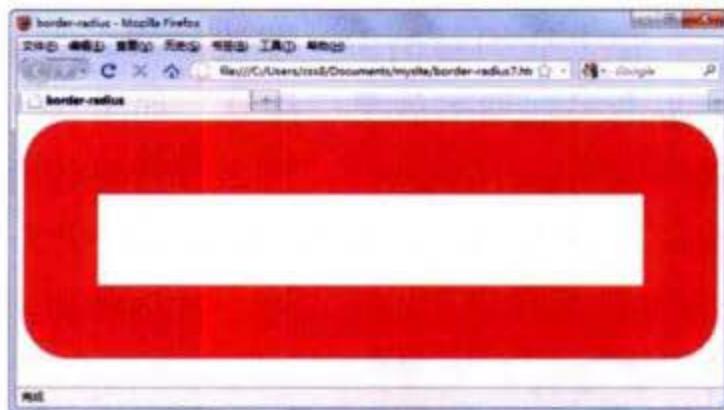
```

内边半径等于外边半径减去对应边的宽度。如果差值为负值时，内边半径是0，则会显示为内直角，而不是内圆角，所以内外边曲线的圆心并不必然是一致的。

```

<style type="text/css">
div {
    height:100px;
    border:80px solid red;
    -moz-border-radius:50px;
    -webkit-border-radius:50px;
    border-radius:50px;
}
</style>

```



在上面的代码基础上，把边框宽度设置为小于圆角半径，则内半径大于0，呈现小幅圆角效果，如下图所示。

```

<style type="text/css">
div {
    height:100px;
    border:40px solid red;
    -moz-border-radius:50px;
    -webkit-border-radius:50px;
    border-radius:50px;
}
</style>

```



继续在上面的代码基础上，把边框宽度设置为远远小于圆角半径，则内半径远远大于0，呈现大幅圆角效果，如下图所示。

```

<style type="text/css">
div {
    height:100px;
    border:10px solid red;
    -moz-border-radius:50px;
    -webkit-border-radius:50px;
    border-radius:50px;
}
</style>

```



使用 border-radius 属性也可以定义圆形。例如，设置元素长宽相同，同时设置圆角半径为元素大小的一半即可。

```

<style type="text/css">
div {

```

```

height:300px;
width:300px;
background:url(images/img9.jpg) no-repeat;
border:1px solid red;
-moz-border-radius:150px;
-webkit-border-radius:150px;
border-radius:150px;
}
</style>

```

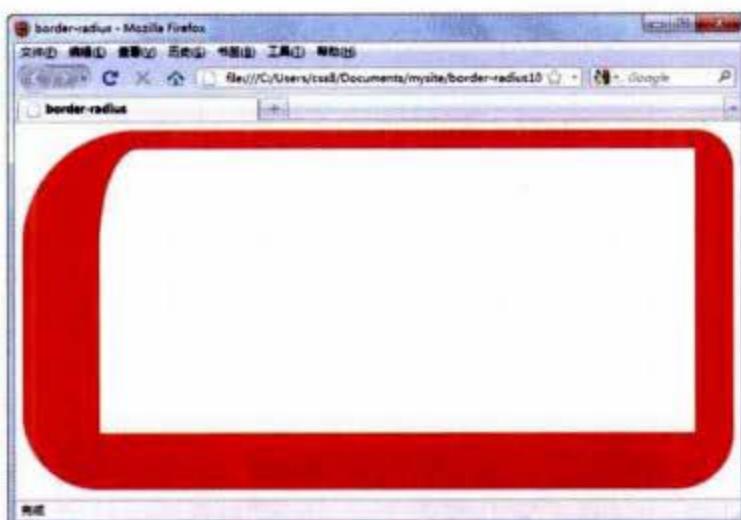
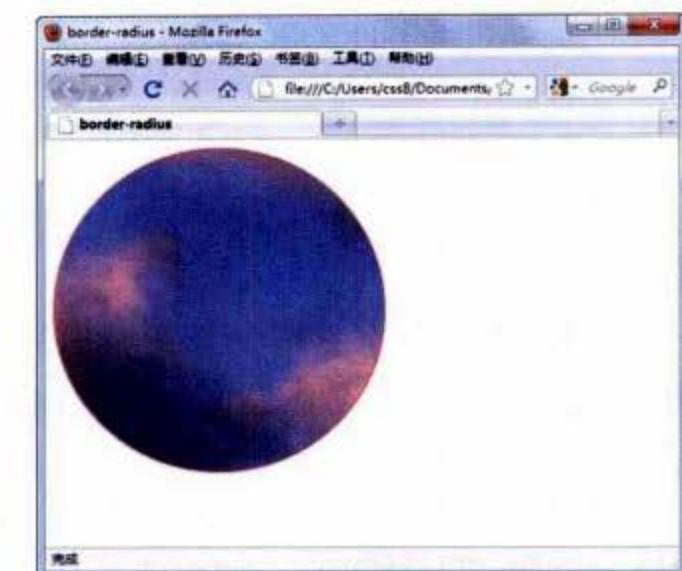
在上面的示例中，即使 border 的属性值为 none，则也会呈现圆形效果。注意，如果 background-clip 的属性值为 padding-box，那么背景会被曲线的圆角内边裁剪。如果 background-clip 的属性值为 border-box，那么背景会被圆角外边裁剪。border 和 padding 属性定义的区域也一样会被曲线裁剪。另外，所有边框样式（如 solid、dotted、inset 等）都遵循边框圆角的曲线，即使是定义了 border-image 属性，曲线以外的边框背景都会被裁剪掉。

如果角的两个相邻边的宽度不同，那么这个角将会从宽的边圆滑过渡到窄的边，其中一条边甚至可以设置为 0。例如，

```

<style type="text/css">
div {
    height:300px;
    border:solid red;
    border-width:20px 40px 60px 80px;
    -moz-border-radius:120px 40px 60px 80px;
    -webkit-border-radius:20px 40px 60px 80px;
    border-radius:20px 40px 60px 80px;
}
</style>

```

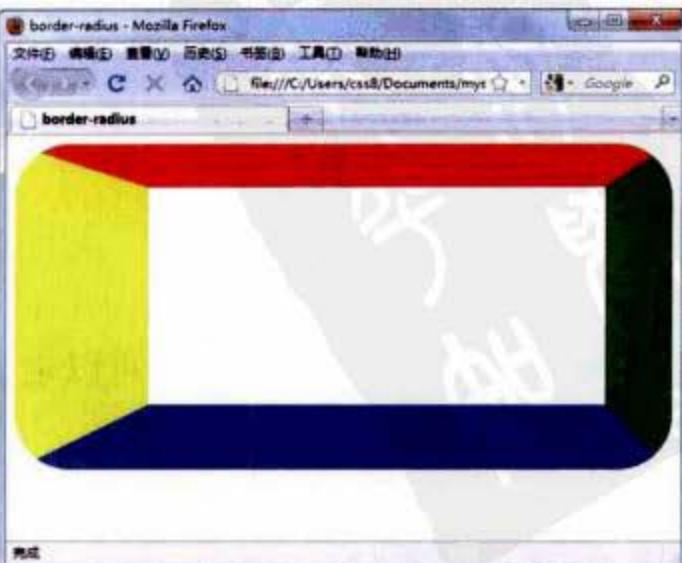


两条相邻边的颜色和样式改变的中心点是在一个与两边宽度成正比的角上。如果两条边宽度相同，那么这个分界点应该就是在一个 45° 的角上。如果一条边的宽度是相邻另一条边的宽度的两倍，那么这个点就在一个 30° 的角上。

```

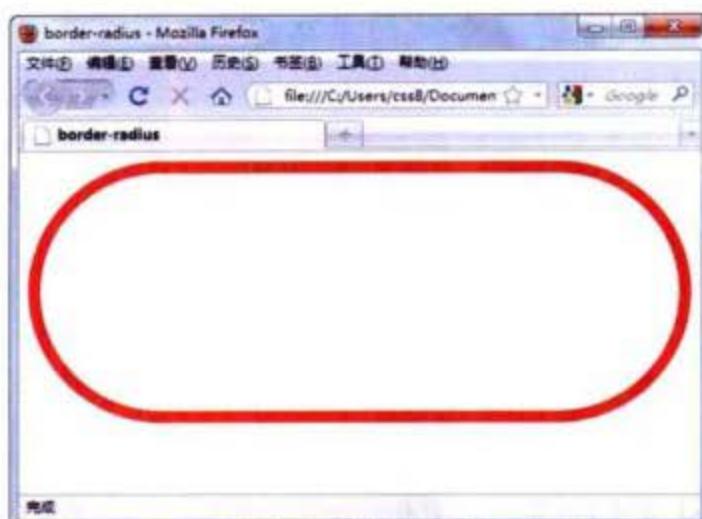
<style type="text/css">
div {
    height:200px;
    border-style:solid;
    border-color:red green blue yellow;
    border-width:40px 60px 60px 120px;
    -moz-border-radius:50px;
    -webkit-border-radius:50px;
    border-radius:50px;
}
</style>

```



圆角是不允许彼此重叠的，所以当相邻两个圆角的半径之和大于元素的宽或高时，浏览器在解析时会强制缩小一个或多个圆角半径。

```
<style type="text/css">
div {
    height:200px;
    border:solid red 10px;
    -moz-border-radius:200px;
    -webkit-border-radius:200px;
    border-radius:200px;
}
</style>
```



5.4 设计块阴影——box-shadow 属性

box-shadow 属性定义元素的阴影，它与 text-shadow 属性的功能是相同的，但是作用对象略有不同。该属性的基本语法如表 5.4 所示。

表5.4 box-shadow属性的基本语法

语法项目	说明
值	none <shadow> [, <shadow>]*
初始值	none
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- none：默认值，表示元素没有阴影。
- <shadow>：该属性值可以使用公式表示为 inset? && [<length>{2,4} && <color>?]。其中：inset 表示设置阴影的类型为内阴影，默认为外阴影；<length> 是由浮点数字和单位标识符组成的长度值，可取正负值，用来定义阴影水平偏移、垂直偏移，以及阴影大小、阴影扩展（即阴影模糊度）；<color> 表示阴影颜色。

浏览器兼容性检测

目前，Webkit 引擎支持 -webkit-box-shadow 私有属性，Mozilla Gecko 引擎支持 -moz-

box-shadow 私有属性，Presto 引擎和 IE 9 支持 box-shadow 标准属性。IE 8 及其以前版本的浏览器暂时不支持 box-shadow 属性。借助兼容方式，各主流浏览器对 box-shadow 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

关于元素阴影的用法解析

box-shadow 属性包含 6 个参数值：阴影类型、X 轴位移、Y 轴位移、阴影大小、阴影扩展和阴影颜色，这 6 个参数值可以有选择地省略。

如果不设置阴影类型，默認為投影效果；当设置为 inset 时，则阴影效果为内阴影。X 轴位移和 Y 轴位移定义阴影的偏移距离。阴影大小、阴影扩展和阴影颜色是可选值，默認為黑色实影。box-shadow 属性必须设置阴影的位移值，否则没有效果。如果定义了阴影大小，此时定义阴影位移为 0，才可以看到阴影效果。下面结合案例进行演示说明。

例如，定义简单的实影投影效果，则效果如下图所示。

```
<style type="text/css">
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:5px 5px;
    -webkit-box-shadow:5px 5px;
    box-shadow:5px 5px;
}
</style>
```

```

```

定义位移、阴影大小和阴影颜色，则效果如下图所示。

```
<style type="text/css">
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:2px 2px 10px #06C;
    -webkit-box-shadow:2px 2px 10px #06C;
    box-shadow:2px 2px 10px #06C;
}
</style>
```

```

```



仅定义阴影大小和阴影颜色，则效果如下图所示。

```
<style type="text/css">
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:0 0 10px #06C;
    -webkit-box-shadow:0 0 10px #06C;
    box-shadow:0 0 10px #06C;
}

</style>


```



带渐变的阴影效果

在上面的示例基础上，添加 10px 的阴影扩展，则效果如下图所示。

```
<style type="text/css">
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:0 0 10px 10px #06C;
    -webkit-box-shadow:0 0 10px 10px #06C;
    box-shadow:0 0 10px 10px #06C;
}

</style>



```



带晕边的阴影效果

定义内阴影、位移 5 像素、阴影大小为 10px、颜色为 #06C，则效果如下图所示。

```
<style type="text/css">
div {
    height:200px;
    width:400px;
    -moz-box-shadow:inset 5px 5px 10px #06C;
    -webkit-box-shadow:inset 5px 5px 10px #06C;
    box-shadow:inset 5px 5px 10px #06C;
}

</style>

<div></div>
```



内阴影效果

通过设置多组参数值可以定义多色阴影，效果如下图所示。

```
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:-10px 0 12px red,
                    10px 0 12px blue,
```



多彩阴影效果

```
    0 -10px 12px yellow,
    0 10px 12px green;
-webkit-box-shadow:-10px 0 12px red,
    10px 0 12px blue,
    0 -10px 12px yellow,
    0 10px 12px green;
box-shadow:-10px 0 12px red,
    10px 0 12px blue,
    0 -10px 12px yellow,
    0 10px 12px green;
}
</style>
```

```

```

通过多组参数值还可以定义渐变阴影，效果如下图所示。

```
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:0 0 10px red,
                    2px 2px 10px 10px yellow,
                    4px 4px 12px 12px green;
    -webkit-box-shadow:0 0 10px red,
                    2px 2px 10px 10px yellow,
                    4px 4px 12px 12px green;
    box-shadow:0 0 10px red,
               2px 2px 10px 10px yellow,
               4px 4px 12px 12px green;
}
</style>
```

```

```

注意，当给同一个元素设计多个阴影时，需要注意它们的顺序，最先写的阴影将显示在最顶层。如在上面这段代码中，先定义一个 10px 的红色阴影，再定义一个 10px 大小、10px 扩展的阴影。显示结果就是红色阴影层覆盖在黄色阴影层之上，此时，如果顶层的阴影太大，就会遮盖底部的阴影。例如，在上面的示例中，如果把第三个参数调整到第一个位置，则下面两个阴影就会被遮盖。

```
<style type="text/css">
img{
    height:300px;
    -moz-box-shadow:4px 4px 12px 12px green,
                    0 0 10px red,
                    2px 2px 10px 10px yellow;
    -webkit-box-shadow:0 0 10px red,
                    2px 2px 10px 10px yellow,
                    4px 4px 12px 12px green;
    box-shadow:0 0 10px red,
               2px 2px 10px 10px yellow,
               4px 4px 12px 12px green;
}
</style>
```



```
</style>  
  

```

5.5 CSS 3 边框和背景样式综合实战

CSS 3 增强的边框样式具有强大的生命力，灵活以使用这些属性可以设计很多精巧的 UI 效果。为了方便读者理解，本节介绍一个 UI 设计案例，以模拟 Windows 7 的界面效果。该案例综合应用了 box-shadow、border-radius、text-shadow、border-color、border-image 等属性，同时还用到了渐变设计属性。关于该技术，请参阅后面章节的讲解。整个案例的演示效果如图 5.9 所示。

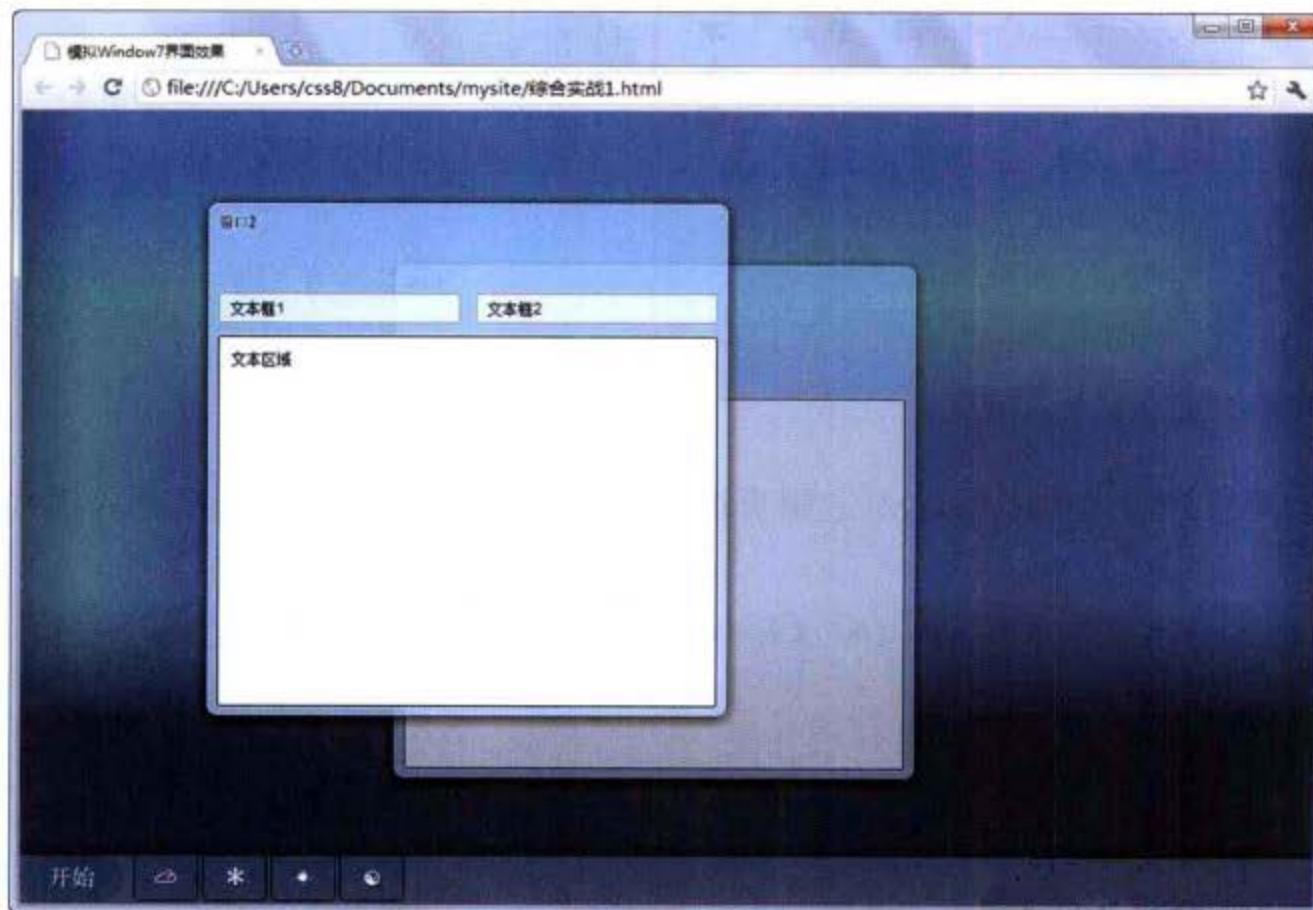


图 5.9 使用 CSS 3 模拟的 Windows 7 界面效果

构建 UI 界面结构

整个 UI 界面的结构比较简单，代码如下所示。

```
<body>  
  <!--桌面结构-->  
  <div id="desktop">  
    <!--窗口1-->  
    <div id="bgWindow" class="window secondary">  
      <span>窗口1</span>
```

```

<div class="content"></div>
</div>
<!--窗口2-->
<div id="frontWindow" class="window">
    <span>窗口2</span>
    <div id="winInput">
        <input type="text" value="文本框1">
        <input type="text" value="文本框2">
    </div>
    <div id="winContent" class="content">文本区域</div>
</div>
<!--开始菜单按钮-->
<div id="startmenu">
    <button id="winflag">开始</button>          <!.....开始按钮.....>
    <span id="toolBtn">                          <!.....任务栏图标.....>
        <button class="application">♣</button>
        <button class="application">*</button>
        <button class="application">☀</button>
        <button class="application">❶</button>
    </span>
</div>
</div>
</body>

```

设计桌面效果

在本案例的文档样式表中，先定制页面样式，然后设置桌面显示背景，样式代码如下所示。

```

html, body { /*页面样式定制，清除边距，显式定义高度*/
    padding:0;
    margin:0;
    height:100%;
}

#desktop { /*定制桌面背景效果*/
    background: #2c609b;
    height:100%; /*满窗口显示*/
    font: 12px "Segoe UI", Tahoma, sans-serif;
    position: relative; /*定义包含框，为后面的桌面定位元素提供参考*/
    -moz-box-shadow: inset 0 -200px 100px #032b5c,
                    inset -100px 100px 100px #2073b5,
                    inset 100px 200px 100px #1f9bb1;
    -webkit-box-shadow: inset 0 -200px 100px #032b5c,
                    inset -100px 100px 100px #2073b5,
                    inset 100px 200px 100px #1f9bb1;
    box-shadow: inset 0 -200px 100px #032b5c,
                inset -100px 100px 100px #2073b5,
                inset 100px 200px 100px #1f9bb1;
    overflow: hidden; /*隐藏超出的内容*/
}

```

定义桌面内阴影，使用一组3个内阴影设计梦幻效果

设计“开始”菜单和任务栏

“开始”菜单和任务栏主要用到了圆角样式和盒子阴影。在设计任务栏中的图标时，还用到了渐变效果，该技术将在后面章节中进行详细说明。该部分的样式代码如下所示。

```
#startmenu { /*设置任务栏效果*/
    /*固定显示在页面底部*/
    position: absolute;
    bottom: 0;
    /*固定大小*/
    height: 40px;
    width: 100%;
    background: rgba(178, 215, 255, 0.25); /*增加半透明效果*/
    -webkit-box-shadow: 0 -2px 20px rgba(0, 0, 0, 0.25);
    -moz-box-shadow: 0 -2px 20px rgba(0, 0, 0, 0.25),
                    inset 0 1px #042754,
                    inset 0 2px #5785b0;
    box-shadow: 0 -2px 20px rgba(0, 0, 0, 0.25),
                inset 0 1px #042754,
                inset 0 2px #5785b0;
    overflow: hidden;
}

#startmenu button {
    font-size: 1.6em;
    color: #fff;
    text-shadow: 1px 2px 2px #00294b; /*为按钮文字增加阴影效果*/
}

#startmenu #winflag { /*设计“开始”按钮样式*/
    float: left;
    margin: 2px;
    height: 34px;
    width: 80px;
    margin-right: 10px;
    border: none;
    background: #034aa76;
    -moz-border-radius: 40px;
    -webkit-border-radius: 40px;
    border-radius: 40px;
    -moz-box-shadow: 0 0 1px #fff,
                    0 0 3px #000,
                    0 0 3px #000,
                    inset 0 1px #fff,
                    inset 0 12px rgba(255, 255, 255, 0.15),
                    inset 0 4px 10px #cef,
                    inset 0 22px 5px #0773b4,
                    inset 0 -5px 10px #0df;
    -webkit-box-shadow: 0 0 1px #fff,
                    0 0 3px #000,
                    0 0 3px #000;
    box-shadow: 0 0 1px #fff,
                0 0 3px #000,
                0 0 3px #000,
                inset 0 1px #fff,
                inset 0 12px rgba(255, 255, 255, 0.15),
                inset 0 4px 10px #cef,
                inset 0 22px 5px #0773b4,
                inset 0 -5px 10px #0df;
}
```

为任务栏设计顶部外
阴影，以及在内部添
加两道阴影效果

设计“开始”按钮
以圆角显示

设计“开始”按钮
的内外阴影特效

```

}

#startmenu .application { /*设计任务栏图标样式*/
    position: relative;
    bottom: 1px;
    height: 38px;
    width: 52px;
    background: rgba(14, 59, 103, 0.25);
    border: 1px solid rgba(0, 0, 0, 0.8);
    /*设计渐变特效*/
    -o-transition: .3s all;
    -webkit-transition: .3s all;
    -moz-transition: .3s all;

    -moz-border-radius: 4px;
    -webkit-border-radius: 4px;
    border-radius: 4px;

    -moz-box-shadow: inset 0 0 1px #fff,
                    inset 4px 4px 20px rgba(255, 255, 255, 0.33),
                    inset -2px -2px 10px rgba(255, 255, 255, 0.25);
    box-shadow: inset 0 0 1px #fff,
                inset 4px 4px 20px rgba(255, 255, 255, 0.33),
                inset -2px -2px 10px rgba(255, 255, 255, 0.25);
}

/*设计鼠标经过时，图标显示为半透明的色彩变化效果*/
#startmenu, application:hover { background-color: rgba(255, 255, 255, 0.25); }

```

设计任务栏图**标以圆角显示****设计任务栏图标
的内外阴影特效**

设计窗口效果

窗口 UI 主要涉及圆角和半透明效果设计，样式代码如下所示。

```

/*设计窗口外框效果*/
.window {
    /*定位窗体大小和位置*/
    position: absolute;
    left: 150px;
    top: 75px;
    width: 400px;
    height: 400px;
    padding: 7px;
    /*设计半透明度效果的边框和背景效果*/
    border: 1px solid rgba(255, 255, 255, 0.6);
    background: rgba(178, 215, 255, 0.75);

    -webkit-border-radius: 8px;
    -moz-border-radius: 8px;
    border-radius: 8px;

    -moz-box-shadow: 0 2px 16px #000,
                    0 0 1px #000,
                    0 0 1px #000;
    -webkit-box-shadow: 0 2px 16px #000,
                    0 0 1px #000,
                    0 0 1px #000;
    box-shadow: 0 2px 16px #000,
                0 0 1px #000,

```

**设计窗体外框
以圆角显示****设计窗体外框的
外阴影特效**

```

        0 0 1px #000;
/*设计晕边效果*/
text-shadow: 0 0 15px #fff, 0 0 15px #fff;
}

.window span { display: block; }
.window input { /*文本输入框样式*/
-webkit-border-radius: 2px;
-moz-border-radius: 2px;
-moz-box-shadow: 0 0 2px #fff,
                0 0 1px #fff,
                inset 0 0 3px #fff;
-webkit-box-shadow: 0 0 2px #fff,
                0 0 1px #fff;
box-shadow: 0 0 2px #fff,
            0 0 1px #fff,
            inset 0 0 3px #fff
}

.window input + input { margin-left: 12px; }

.window.secondary { /*定位第二个窗体位置和不透明度*/
    left: 300px;
    top: 125px;
    opacity: 0.66;
}

.window.secondary span { margin-bottom: 85px; }

/*设计窗口内文本区域样式*/
.window .content {
    padding: 10px;
    height: 279px;
    background: #fff;
    border: 1px solid #000;
    -webkit-border-radius: 2px;
    -moz-border-radius: 2px;
    border-radius: 2px;
    -moz-box-shadow: 0 0 5px #fff,
                    0 0 1px #fff,
                    inset 0 1px 2px #aaa;
    -webkit-box-shadow: 0 0 5px #fff,
                    0 0 1px #fff;
    box-shadow: 0 0 5px #fff,
            0 0 1px #fff,
            inset 0 1px 2px #aaa;
    text-shadow: none; /*取消文本阴影*/
}

```

设计文本输入
框以圆角显示

设计文本输入框
的内外阴影特效

设计文本区域
框以圆角显示

设计文本区域框
的内外阴影特效

5.6 设计多重背景图象——background 属性

应该说，background 是 CSS 中使用频率最高的属性。为了方便设计师更灵活地设计网页效果，CSS 3 增强了该属性的功能，允许设计师在同一个元素内叠加多个背景图像。该属性的基本语法如表 5.5 所示。

表5.5 background属性的基本语法

语法项目	说明
值	[<bg-layer>,]* <final-bg-layer>
初始值	根据个别属性而定
适用于	所有元素
可否继承	否
百分比	根据个别属性而定
媒介	视觉

取值简单说明：

关于 background 属性的用法，相信很多读者都已经熟悉了。在 CSS 3 中，background 属性依然保持以前的用法，不过可以允许在该属性中添加多个背景图像组，背景图像之间通过逗号进行分隔。其中 <bg-layer> 表示一个背景图像层。每个背景图像层都可以包含下面的值：

[background-image] | [background-color] | [background-origin] | [background-clip] |
 [background-repeat] | [background-size] | [background-position] | [background-attachment]

派生的子属性

为了方便设计师更灵活地定义背景图像，background 属性又派生了 8 个子属性。

- background-image：定义背景图像。
- background-color：定义背景颜色。
- background-origin：指定背景的显示区域。
- background-clip：指定背景的裁剪区域。
- background-repeat：设置背景图像是否及如何重复铺排。
- background-size：定义背景图片的大小。
- background-position：设置背景图像的位置。
- background-attachment：定义背景图像的显示方式。

浏览器兼容性检测

各大主流浏览器都没有支持 background 属性的私有属性，能够支持 background 属性定义多重背景的检测情况如下。

				
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.5	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.6	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：背景图像合成

`background` 属性支持多重背景功能极大地提升了设计师的设计体验，使设计师部分摆脱了对 Photoshop 等绘图工具的依赖。例如，如果我们希望把下面两幅图合成在一起，并设计为页面的背景，现在可以直接在 `background` 属性中完成，而不用 Photoshop 事先合成（如图 5.10 所示）。

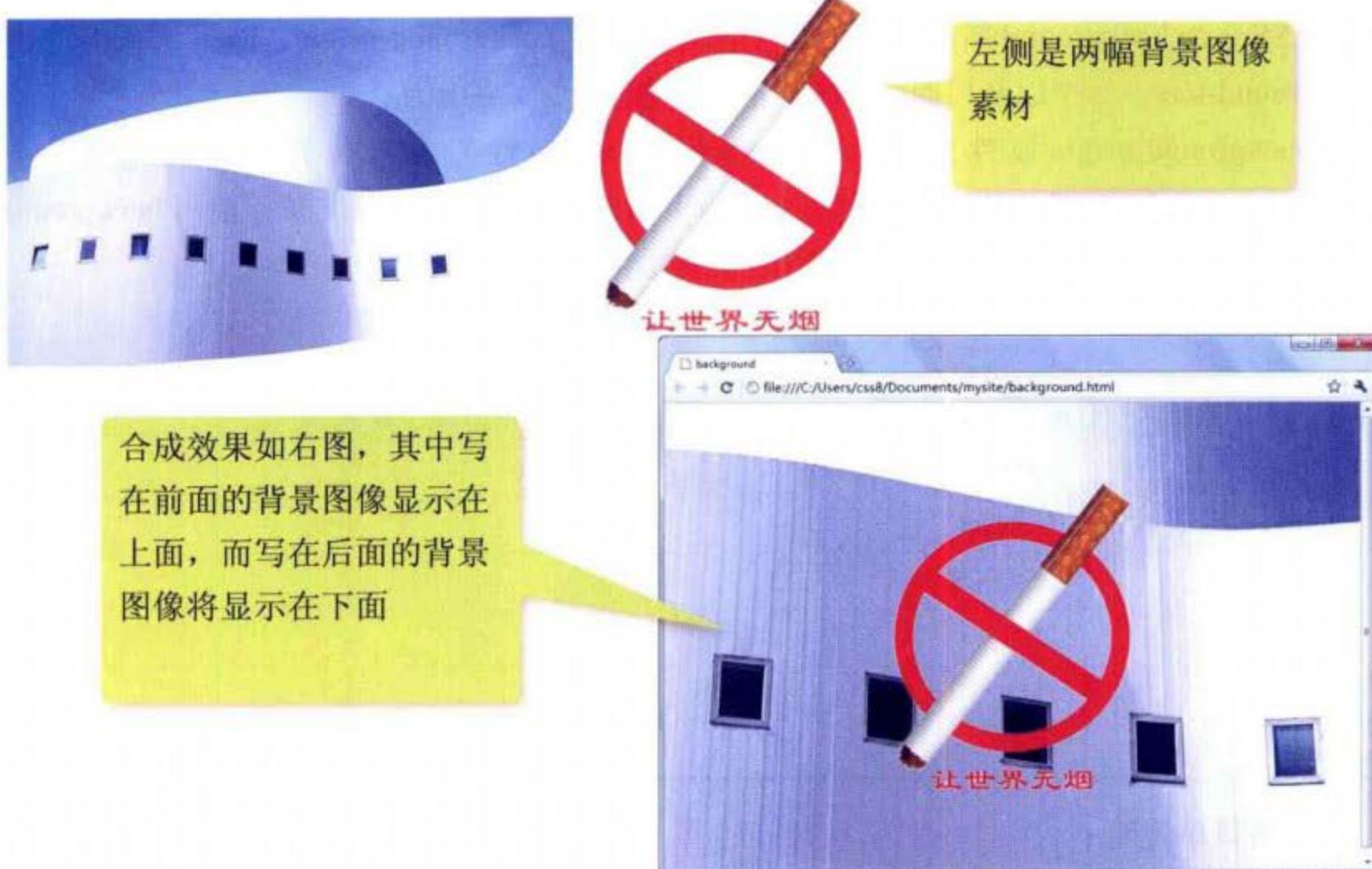


图 5.10 多背景合成效果

实现本案例的代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background</title>
<style type="text/css">
html, body { height:100%; } /*显式定义页面高度显示为100%，否则网页显示为一条线*/
body {
    background: url(images/bg3.png) center no-repeat,
                url(images/img15.jpg) center 70% no-repeat;
}
</style>
</head>
<body>
```

定义多重网页背景，用法类似多个 `background` 属性值分组合并在一个 `background` 属性中显示

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

5.7 定义背景坐标原点——background-origin 属性

CSS 3 为 background 新增了 3 个派生的子属性：background-origin、background-clip 和 background-size。本节以及后面两节将详细介绍这三个属性的用法。

background-origin 属性定义 background-position 属性的参考位置。在默认情况下，background-position 属性总是以元素左上角为坐标原点进行背景图像定位。使用 background-origin 属性可以改变这种定位方式。该属性的基本语法如表 5.6 所示。

表 5.6 background-origin 属性基本语法

语法项目	说明
值	border padding content
初始值	padding
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- border：从边框区域开始显示背景。
- padding：从补白区域开始显示背景。
- content：仅在内容区域显示背景。

注意，在最新版本的 CSS 背景模块规范中 (<http://www.w3.org/TR/css-3-background/#background-origin>)，W3C 规定该属性取值为 padding-box、border-box 和 content-box，不过目前还没有得到主流浏览器的支持。

浏览器兼容性检测

目前，Webkit 引擎支持 -webkit-background-origin 私有属性，Mozilla Gecko 引擎支持 -moz-background-origin 私有属性，Presto 引擎和 IE 浏览器暂不支持该属性。借助兼容方式，各主流浏览器对 background-origin 属性的支持情况如下。

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计信纸背景效果

`background-origin` 属性改善了背景图像定位的方式，使设计师能够更灵活地决定背景图像应该显示的位置。例如，在下面这个案例中，就是利用 `background-origin` 属性重设背景图像的定位坐标，以便更好地控制背景图像的显示，效果如图 5.11 所示。

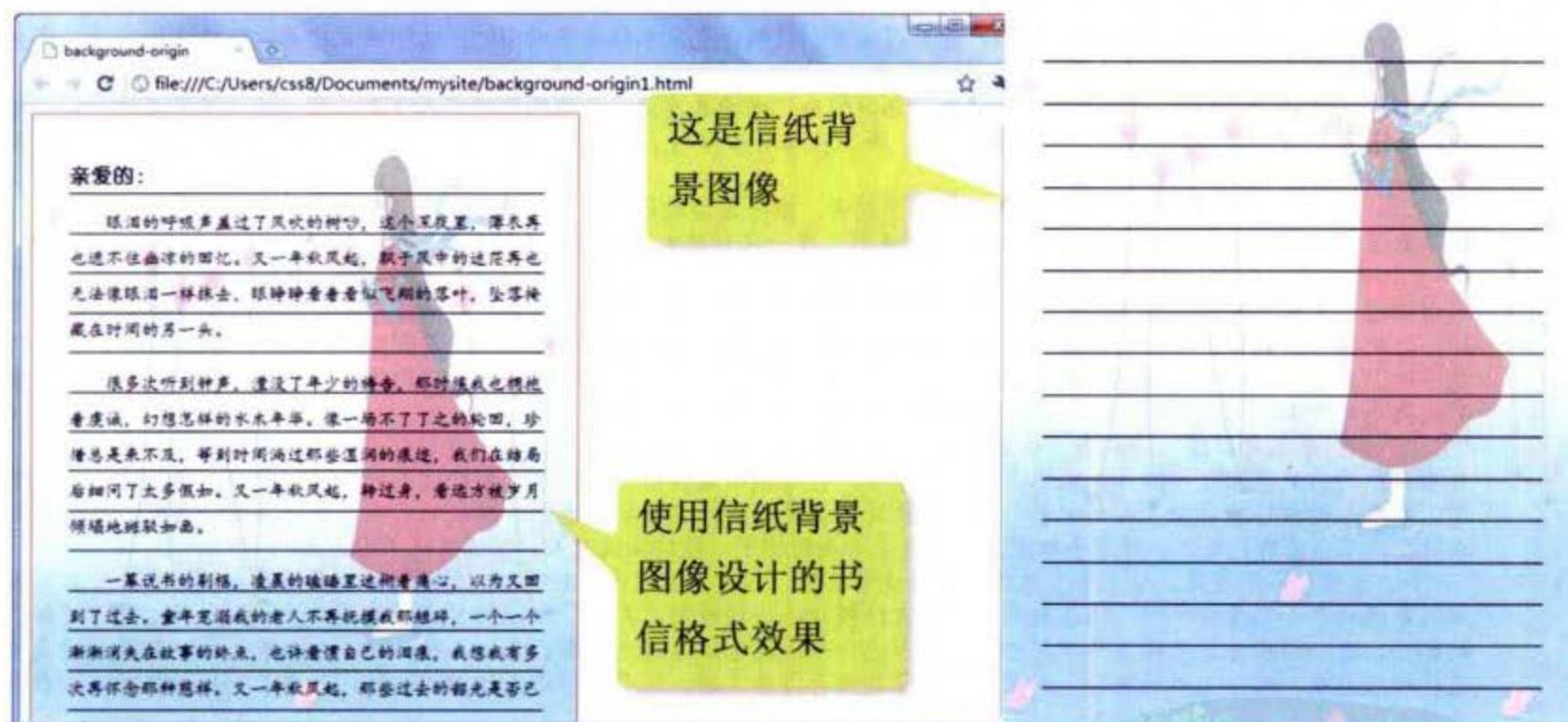


图 5.11 设计的信纸格式效果

实现本案例的代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background-origin</title>
<style type="text/css">
div {
    height:600px;
    width:416px;
    background-color:#f0f0ff;
    background-image: url(rose.jpg);
    background-position: center;
    background-repeat: no-repeat;
    background-size: cover;
    background-origin: border-box;
    padding: 10px;
}
```

```

border:solid 1px red;
padding:32px 2em 0;
background:url(images/p2.jpg) no-repeat;
-moz-background-origin:padding;
-webkit-background-origin:padding;
background-origin:padding;
overflow:hidden;
}
div h1 {
    font-size:18px;
    font-family:"幼圆";
}
div p {
    text-indent:2em;
    line-height:2em;
    font-family:"楷体";
}
</style>
</head>
<body>
<div>
    <h1>亲爱的：</h1>
    <p>眼泪的呼吸声盖过了风吹的树吵，这个深夜里，薄衣再也遮不住幽凉的回忆。又一年秋风起，飘于风中的迷茫再也无法像眼泪一样抹去，眼睁睁看着看似飞翔的落叶，坠落掩藏在时间的另一头。</p>
    <p>很多次听到钟声，湮没了年少的祷告，那时候我也拥抱着虔诚，幻想怎样的水木年华。像一场不了了之的轮回，珍惜总是来不及，等到时间淌过那些湿润的痕迹，我们在结局后细问了太多假如。又一年秋风起，转过身，看远方被岁月倾塌地斑驳如画。</p>
    <p>凌晨的瞌睡里迷糊着痛醒，以为又回到了过去。童年宠溺我的老人不再抚摸我，一个一个渐渐消失在故事的终点，也许看惯自己的泪痕，我想我有多次再怀念那种慈祥。又一年秋风起，那些过去的韶光是否已经疲惫？</p>
    <p>有很多次后来，那时候的星光有父母轻声低诉，如果那时候没有那么多倔强，我想父母也不会有那么多叹息。等到明白初醒，才会在眼泪中发觉，有些爱，有些人，其实一直窝在你的心里。也许半夜来给你盖被子的人还会半夜来偷偷看你，会抚摸你开始褪去稚嫩的脸庞，看一种宠爱的眼神。他们的念叨里会写满心慌，没有阻碍，他们一直想着你，即便你就在我身边。又一年秋风起，那些银色发梢的年轮是否会上色他们的蹒跚？皱纹垒起的落寞，我想那些厚茧爬着手上有我熟悉的汗味。</p>
    <p>渐行渐远的路途，一起去上体育课的室友，那些安置在操场上的器械，像一只没有水的鱼。站在多愁善感的门外，青春舒张的感慨像面包一样开始吸水，青涩，躁动，也许我们是在悬崖边跳舞。那些远去的影子，都在画着他们自己的梦想，每个人的归路都会涂上我的祝福。又一年秋风起，是否我的那些朋友还会念到我的名字，我是在听一首歌，我会控制我自己，不会看自己哭泣。时常会碰到悉如挚友上发夹的样子，然后怔怔发呆，坚强得说不出话来。</p>
    <p>这样相似的夜晚，噎着太多感伤，爱情说的故事，只有多年以后才会明白它的温度。当年那个男孩，那个女孩，在那些命运的手心里虚掷年华。你瘦瘦的身影，曾经模样的我们，充满了稚嫩的誓言。那个戏里的人，谁才是不可缺少的部分，有一种感觉总在那种年少的稚嫩里永恒成一幅永远无法超越的独赏。又一年秋风起，学会长大着，那年的他是否已经功成名就？缘分书写的是否、如果，我在深夜里青涩着半醉，不管心中沾满的泪印。时间改变了一切的样子，却在心里留下挥之不去的影子。</p>
    <p>再添一件衣，铜铃乐着的清幽像一个铁爪，伶碎了过往。又一阵秋风起，我该怎样述说那些未来得及舒展的心情，天花板上有我脸色一样的惨白。夏去，秋来，那件白色裙子已然褶皱，惆怅纠结着太多为什么。远方看不到的人海，在这阵秋风下延伸得更加寂寥。失落默默收拾着心情，我想，比秋天更冷的冬天已经不远了。</p>
    <p>灯渐熄，看窗外抖落的树叶，不可安然入睡，拥着枕头，我面无表情。</p>
</div>
</body>
</html>

```

为了避免背景图像重复平铺到边框区域，应禁止它平铺

设计背景图像的定位坐标点为元素补白区域的左上角

5.8 定义背景裁剪区域——background-clip 属性

background-clip 属性定义背景图像的裁剪区域。background-clip 属性与 background-

origin 属性有几分关联。通俗地说，background-clip 属性用来判断背景是否包含边框区域，而 background-origin 属性用来决定 background-position 属性定位的参考位置，它们的属性取值也很相似。该属性的基本语法如表 5.7 所示。

表5.7 background-clip属性的基本语法

语法项目	说明
值	border padding content no-clip
初始值	border-box
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- border：从边框区域向外裁剪背景。
- padding：从补白区域向外裁剪背景。
- content：从内容区域向外裁剪背景。
- no-clip：从边框区域向外裁剪背景。

注意，在最新版本的 CSS 背景模块规范中 (<http://www.w3.org/TR/css3-background/#background-origin>)，W3C 规定该属性取值为 padding-box、border-box 和 content-box，不过目前还没有得到主流浏览器的支持。

浏览器兼容性检测

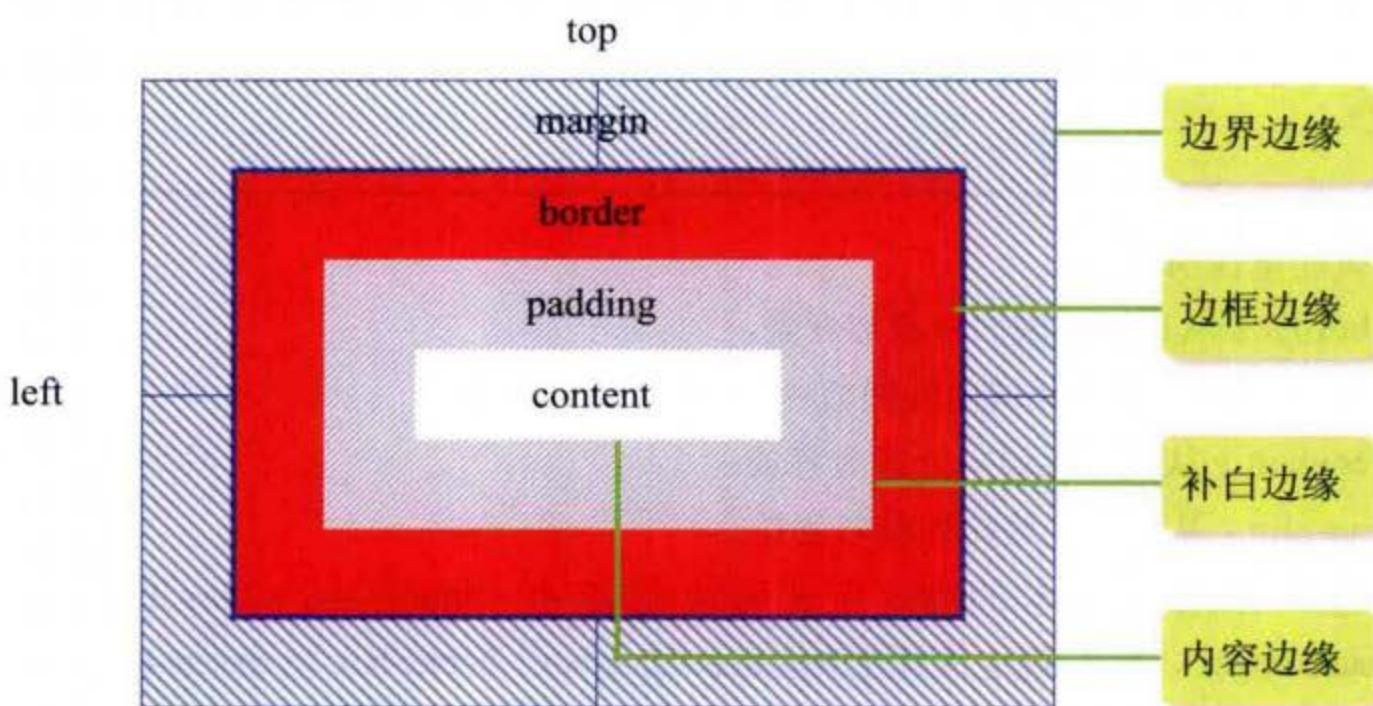
目前，Webkit 引擎支持 -webkit-background-clip 私有属性，Mozilla Gecko 引擎支持 -moz-background-clip 私有属性，Presto 引擎和 IE 9 浏览器支持该属性。借助兼容方式，各主流浏览器对 background-clip 属性的支持情况如下。

				
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

background-clip 属性和 background-origin 属性实现的效果基本相同，但是它们的实现原理是不同的。在具体设计中，设计师可以选择使用 background-clip 属性和 background-origin 属性。

background-clip 与 background-origin 属性用法比较与分解

首先我们再来认识一下盒模型基本结构。对于任何一个元素来说，它都会包含四区域、四边沿，即边界（margin）区域、边框（border）区域、补白（padding）区域和内容（content）区域，以及边界边缘、边框边缘、补白边缘、内容边缘（如下图所示）。



对于 background-clip 属性来说，如果取值为 padding，则 background-image 将忽略补白边缘，此时边框区域显示为透明；如果取值为 border，则 background-image 将包括边框区域；如果取值为 content，则 background-image 将只包含内容区域；如果 background-image 属性定义了多重背景，则 background-clip 属性值可以设置多个值，并用逗号分隔。

对于 background-origin 属性来说，如果取值为 padding，则 background-position 相对于补白边缘进行定位。其中，当 background-position 属性值为 "0 0" 时，定位点为补白边缘的左上角，而当 background-position 属性值为 "100% 100%" 时，定位点为补白边缘的右下角。如果取值为 border，则 background-position 相对于边框边缘进行定位。如果取值为 content，则 background-position 相对于内容边缘进行定位。与 background-clip 属性相同，多个值之间使用逗号分隔。

如果 background-clip 属性值为 padding，background-origin 属性取值为 border，且 background-position 属性值为 "top left"（默认初始值），则背景图左上角将会被截取掉部分。

总之，background-clip 默认效果类似于 background-clip:border。background-origin 默认效果类似于 background-origin:padding。

实战体验 1：设计内容区背景

background-clip 属性的用法很简单，在使用时应适当考虑浏览器兼容性问题。例如，如果希望背景图像仅在内容区域内显示，可以使用下面的方法来设计，演示效果如图 5.12 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background-clip</title>
<style type="text/css">
div {
    height:50px;
    width:200px;
    border:dotted 50px red;
    padding:50px;
    background:url(images/img16.jpg) no-repeat;
    -moz-background-clip:content;
    -webkit-background-clip:content;
    background-clip:content;
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

设计背景图像的裁切方式，
以内容边缘进行裁切

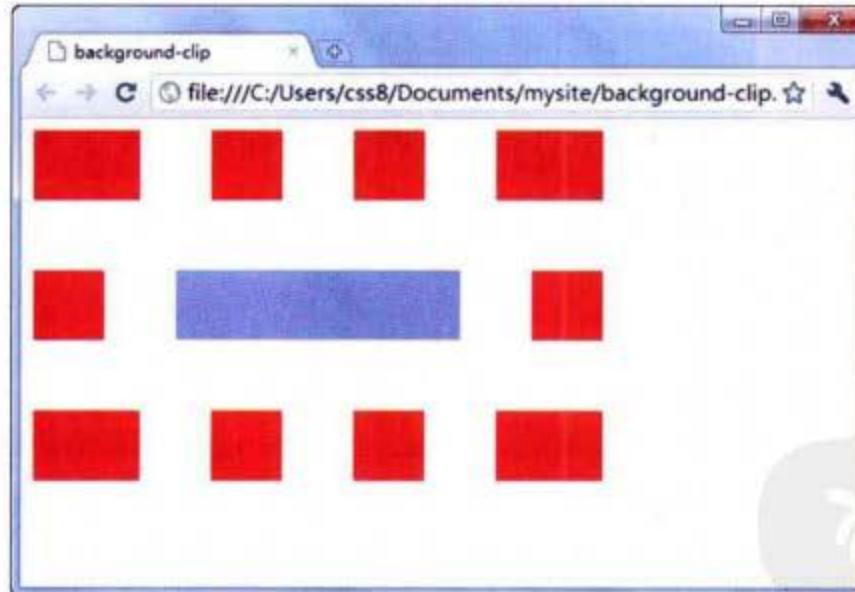


图 5.12 以内容边缘裁切背景图像效果

实战体验 2：设计按钮效果

background-clip 和 background-origin 属性配合使用，可以避免浏览器兼容问题。例如，在下面的示例中同时定义 background-clip 和 background-origin 属性值为 content，可以设计比较特殊的按钮样式，效果如图 5.13 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background-clip</title>
<style type="text/css">
button {
    height:40px;
    width:150px;
    padding:1px;
    cursor:pointer;
    color:#fff;
    border:3px double #95071b;
    border-right-color:#650513;
    border-bottom-color:#650513;
    background:url(images/img5.jpg) no-repeat;
    -moz-background-origin:content;
    -webkit-background-origin:content;
    background-origin:content;
    -moz-background-clip:content;
    -webkit-background-clip:content;
    background-clip:content;
}
</style>
</head>
<body>
<button>设计按钮特殊样式</button>
</body>
</html>

```

为了避免背景图像重复平铺到边框区域，应禁止它平铺

设计背景图像的定位坐标点为元素内容区域的左上角

设计以内容区域的边缘裁切背景图像



图 5.13 设计特殊样式的按钮效果

5.9 定义背景图像大小——background-size 属性

background-size 是 CSS 3 新增的比较实用的属性，使用它可以随心所欲地控制背景图像的显示大小。在 CSS 2 及其以前版本中，背景图像的大小是不可以控制的，如果要想使背景图像填充元素背景区域，则需要事先设计更大的背景图像，要么就只能让背景图像以平铺的方式来填充元素。background-size 属性的基本语法如表 5.8 所示。

表5.8 background-size属性的基本语法

语法项目	说明
值	[<length> <percentage> auto]{1,2} cover contain
初始值	auto
适用于	所有元素
可否继承	否
百分比	根据文本
媒介	视觉

取值简单说明：

- <length>：由浮点数字和单位标识符组成的长度值。不可为负值。
- <percentage>：取值为 0% 到 100% 之间的值。不可为负值。
- cover：保持背景图像本身的宽高比例，将图片缩放到正好完全覆盖所定义背景的区域。
- contain：保持图像本身的宽高比例，将图片缩放到宽度或高度正好适应所定义背景的区域。

注意，background-size 属性可以设置 1 个或 2 个值，1 个为必填，1 个为选填。其中，第 1 个值用于指定背景图像的宽度，第 2 个值用于指定背景图像的高度，如果只设置 1 个值，则第 2 个值默认为 auto。

浏览器兼容性检测

目前，Webkit 引擎支持 -webkit-background-size 私有属性，Mozilla Gecko 引擎支持 -moz-background-size 私有属性，Presto 引擎和 IE 9 浏览器支持该属性。借助兼容方式，各主流浏览器对 background-size 属性的支持如下。

				
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.5	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.6	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计自适应模块大小的背景图像

借助 background-size 属性自由定制背景图像大小的功能，允许背景图像自适应盒子的大小，从而可以设计与模块大小完全适应的背景图像，避免了因区块尺寸不同而需要设计师单独为它们设计不同的背景图像，因此降低了开发的成本。本案例效果如图 5.14 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background-size</title>
<style type="text/css">
div {
    margin:2px;
    float:left;
    border:solid 1px red;
    background:url(images/img8.jpg) no-repeat center;
    -moz-background-size:cover;
    -webkit-background-size:cover;
    background-size:cover;
}
.h1 { height:120px; width:192px; }
.h2 { height:240px; width:384px; }
.h3 { height:360px; width:576px; }
</style>
</head>
<body>
<div class="h1"></div>
<div class="h2"></div>
<div class="h3"></div>
</body>
</html>

```



图 5.14 能够自适应模块区域大小的版块效果

5.10 溢出内容处理——overflow-x 和 overflow-y 属性

overflow 是 CSS 2.1 规范中的特性，而 overflow-x 和 overflow-y 属性则是 CSS 3 基础

盒模型 (<http://www.w3.org/TR/css3-box/>) 草案中新加入的特性。overflow 属性定义当一个块级元素的内容溢出元素的框（它作为内容的包含块）时，是否剪切显示。overflow-x 属性定义对左右边（水平方向）的剪切，而 overflow-y 属性定义对上下边（垂直方向）的剪切。overflow-x 和 overflow-y 属性的基本语法如表 5.9 所示。

表5.9 overflow-x和overflow-y属性的基本语法

语法项目	说明
值	visible hidden scroll auto no-display no-content
初始值	visible
适用于	非替换的块元素，或者非替换的行内块元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- visible：不剪切内容，也不添加滚动条。该属性值为默认值，元素将被剪切为包含对象的窗口大小，且 clip 属性设置将失效。
- auto：在需要时剪切内容并添加滚动条。该属性为 body 和 textarea 元素的默认值。
- hidden：不显示超出元素尺寸的内容。
- scroll：当内容超出元素尺寸时，overflow-x 显示为横向滚动条，而 overflow-y 显示为纵向滚动条。
- no-display：当内容超出元素尺寸时不显示元素，此时类似添加了 display:none 声明。该属性值是最新添加的。
- no-content：当内容超出元素尺寸时不显示内容，此时类似添加了 visibility: hidden 声明。该属性值是最新添加的。

浏览器兼容性检测

目前，所有浏览器都能够正确解析这两个属性。但是部分浏览器在解析时，会存在一些细节差异，在下面的案例实战中将详细分解。浏览器对 overflow-x 和 overflow-y 的支持情况如下。

				
IE 6	Firefox 1.5	Opera 9.0	Safari 3.0	Chrome 1.0.x
IE 7	Firefox 2.0	Opera 9.6	Safari 4.0	Chrome 2.0.x
IE 8	Firefox 3.0	Opera 10.0		Chrome 3.0.x
IE 9	Firefox 3.5	Opera 10.5		Chrome 4.0.x

overflow-x 和 overflow-y 取值的兼容性处理

根据 CSS 3 基础盒模型草案规范，overflow-x 和 overflow-y 的计算值与所设置的值应该相等，除非这一对值不合理。如果其中一个属性值被设置成了 scroll 或 auto，而另一个属性值为 visible，那么 visible 会被设置成 auto。如果 overflow-x 和 overflow-y 属性值相同，则 overflow 的计算值与前两者的指定值相同。否则，它的值是一个 overflow-x 和 overflow-y 的计算值对。关于 overflow 的详细资料，请参考 CSS 2.1 规范 (<http://www.w3.org/TR/CSS2/visufx.html#overflow>)；关于 overflow-x 和 overflow-y 的详细资料，请参考 CSS 3 基础盒模型草案 (<http://www.w3.org/TR/css3-box/#overflow>)。

当 overflow-x 或 overflow-y 属性值中的一个为 hidden、另一个为 visible 时，该元素最终渲染使用的 overflow-y 或 overflow-x 属性值不同。IE 浏览器使用 hidden，其他浏览器使用 auto。此问题可能造成页面内容显示不完全，或在不同浏览器下最终显示效果不一致，如图 5.15 所示。

CSS 3 草案中并没有说明，当 overflow-x 和 overflow-y 中的一个属性值为 hidden、另一个属性值为 visible 时，该 visible 值应该设置为什么，各浏览器有自己的实现。例如，我们可以通过以下代码进行测试和说明。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>background-size</title>
<style type="text/css">
#cont1 div, #cont3 div, #cont5 div {
    width:300px;
    height:200px;
}
#cont2 div, #cont4 div, #cont6 div {
    width:100px;
    height:50px;
}
.cont {
    float:left;
    margin:4px;
    overflow-y:visible;
    padding:10px;
    width:200px;
    height:100px;
}
.cont, .cont div { border : solid 2px red; }
</style>
</head>
<body>
<div id="cont1" class="cont" style="overflow-x:scroll; ">
    <div></div>

```

```

</div>
<div id="cont2" class="cont" style="overflow-x:scroll; ">
    <div></div>
</div>
<div id="cont3" class="cont" style="overflow-x:auto; ">
    <div></div>
</div>
<div id="cont4" class="cont" style="overflow-x:auto; ">
    <div></div>
</div>
<div id="cont5" class="cont" style="overflow-x:hidden; ">
    <div></div>
</div>
<div id="cont6" class="cont" style="overflow-x:hidden; ">
    <div></div>
</div>
</body>
</html>

```

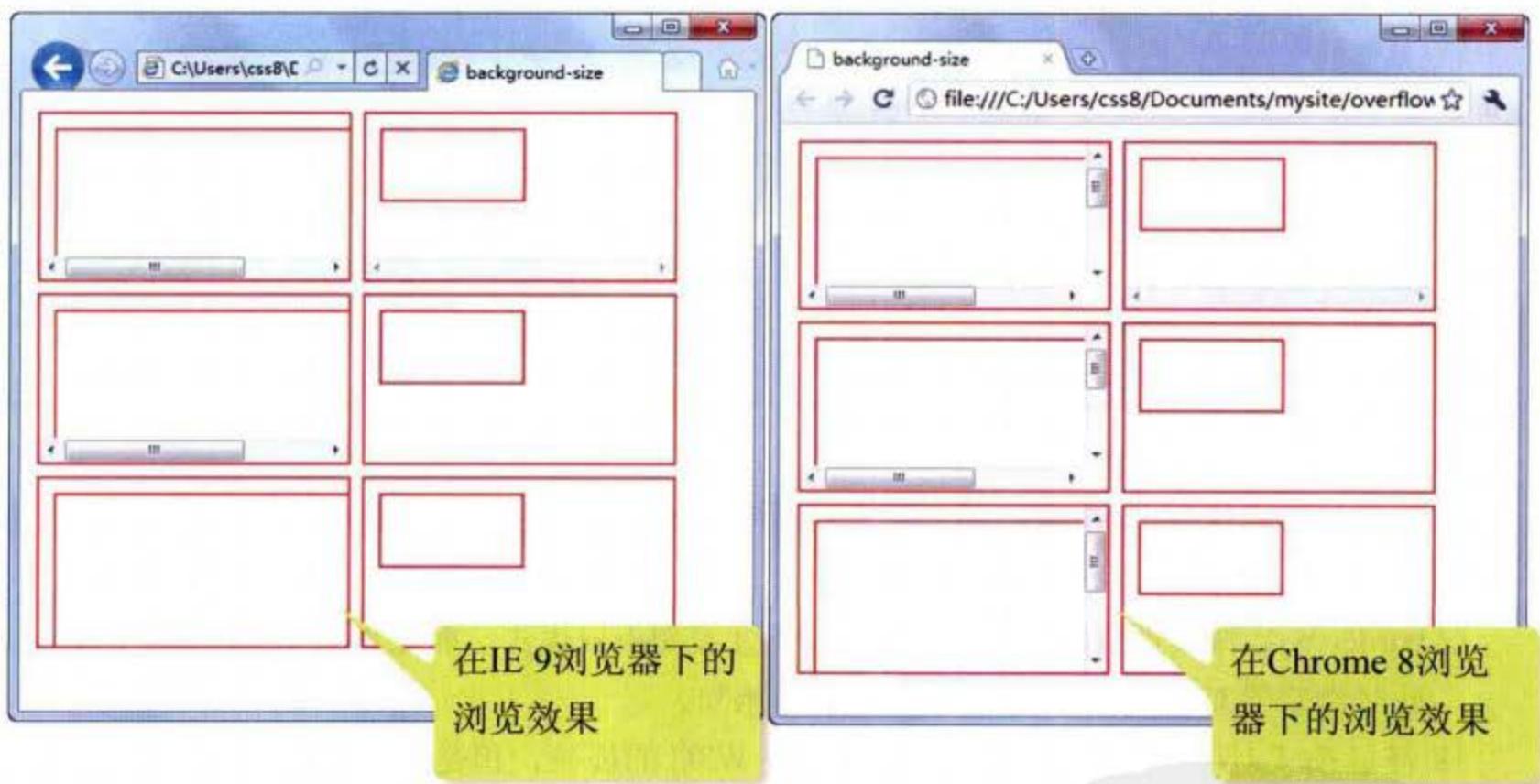


图5.15 比较IE 9和其他浏览器在解析内容溢出时的差异

通过上面案例的比较分析，可以看到：对于 overflow-x 和 overflow-y 的组合渲染，所有浏览器均依照规范处理。但是当 overflow-x 属性值为 hidden 且 overflow-y 属性值为 visible 时，IE 浏览器将 overflow-y 渲染为 hidden，其他浏览器则把该属性渲染为 auto。也就是说，在 IE 浏览器中所有容器的 overflow-y 计算值都为 visible，而其他浏览器中其值却为 auto。

要避免不同浏览器在解析上的差异，在使用时应该同时设置 overflow-x 和 overflow-y 的属性值，不要出现其中一个值为 hidden，而另一个值为 visible 的情况。

5.11 定义盒模型解析模式——box-sizing 属性

CSS 3 规范新增加了 UI 模块 (User-Interface 样式模块)，该模块用来控制与用户界面相关效果的呈现方式，详细资料请参阅 <http://www.w3.org/TR/css3-ui/>。

在 IE 5.x 以及 Quirks (怪异) 模式的 IE 6、IE 7 浏览器中，border 和 padding 都包含在 width 或 height 之内。这为设计师平添了不少麻烦。而在符合标准的浏览器中，width 和 height 仅仅包含 content，刨去了 border 和 padding 区域。

为了兼顾这种问题的存在，CSS 3 对盒模型进行了改善，定义了 box-sizing 属性，该属性能够事先定义盒模型的尺寸解析方式。box-sizing 属性的基本语法如表 5.10 所示。

表5.10 box-sizing属性的基本语法

语法项目	说明
值	content-box border-box inherit
初始值	content-box
适用于	所有能够定义宽和高的元素
可否继承	否
百分比	根据文本
媒介	视觉

取值简单说明：

- content-box：该属性值将维持 CSS 2.1 盒模型的组成模式，即元素 $\text{width}/\text{height} = \text{border} + \text{padding} + \text{content}$ 。
- border-box：该属性值将重新定义 CSS 2.1 盒模型组成模式，即元素 $\text{width}/\text{height} = \text{content}$ 。此时浏览器对盒模型的解释与 IE 6 解析相同。

IE 怪异模式对于盒模型的解释固然不符合 W3C 的规范，但是这种解析方法也有它的好处：无论如何改动元素边框或者补白大小，都不会影响元素的总尺寸，也就不会打乱页面的整体布局。而在标准浏览器下，如果要改变一下 border 或者 padding 的值，就不得不重新计算元素的尺寸，从而影响整个页面的布局。虽然 IE 怪异模式不符合标准，但这种做法还是值得设计师学习。

浏览器兼容性检测

目前，Webkit 引擎支持 -webkit-box-sizing 私有属性，Mozilla Gecko 引擎支持 -moz-box-sizing 私有属性，Presto 引擎和 IE 浏览器直接支持该属性。借助兼容方式，各主流浏览器对 box-sizing 属性的支持情况如下。

✗ IE 6	✓ Firefox 1.5	✓ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✓ Firefox 3.5	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.6	✓ Opera 10.5		✓ Chrome 4.0.x

5.12 自由缩放——resize 属性

resize 是 CSS 3 新增的一个非常实用的属性，该属性能够允许用户自由缩放浏览器中某个元素的大小。在此之前，设计师要实现相同的 UI 效果，需要借助 JavaScript 编写大量的脚本才能够实现，这样既费时费力，且执行效率也很低。resize 属性的基本语法如表 5.11 所示。

表5.11 resize属性的基本语法

语法项目	说明
值	none both horizontal vertical inherit
初始值	none
适用于	所有 overflow 属性不为 visible 的元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- none：浏览器不提供尺寸调整机制，用户不能操纵机制调节元素的尺寸。
- both：浏览器提供双向尺寸调整机制，允许用户调节元素的宽度和高度。
- horizontal：浏览器提供单向水平尺寸调整机制，允许用户调节元素的宽度。
- vertical：浏览器提供单向垂直尺寸的调整机制，允许用户调节元素的高度。
- inherit：默认继承。

注意，resize 属性属于 UI 模块（User-Interface 样式模块），详细资料请参阅 [http://www.w3.org/TR/css3-ui/。](http://www.w3.org/TR/css3-ui/)

浏览器兼容性检测

各浏览器对 resize 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	Safari 3.0	Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	Safari 4.0	Chrome 2.0.x
✗ IE 8	✗ Firefox 3.5	✗ Opera 10.0		Chrome 3.0.x
✗ IE 9	✗ Firefox 3.6	✗ Opera 10.5		Chrome 4.0.x

实战体验：设计能随意调整大小的壁画

目前仅有 Safari 和 Chrome 主流浏览器允许元素缩放，但尚未完全支持，只允许双向调整。CSS 3 允许将该属性应用到任意元素，这将使网页缩放功能拥有跨浏览器的支持。在下面这个示例中，我们将演示如何使用 `resize` 属性设计可以自由调整大小的壁画，效果如图 5.16 所示。



图 5.16 设计能够自由调整大小的壁画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>resize</title>
<style type="text/css">
#resize {
    background:url(images/img19.jpg) no-repeat center;
    -moz-background-clip:content;
    -webkit-background-clip:content;
    background-clip:content;
}
```

以背景方式显示图像，这样可以更轻松地控制缩放操作

设计背景图像仅在内容区域显示，留出补白区域

```

width:200px;
height:120px;
max-width:800px;
max-height:600px;
padding:6px;
border: 1px solid red;
resize: both;
overflow: auto;
}

</style>
</head>
<body>
<div id="resize"></div>
</body>
</html>

```

设计元素最小和最大显示尺寸，用户也只能在该范围内自由调整

必须同时定义overflow和resize，否则resize属性声明无效，因为元素默认溢出显示为visible

5.13 定义外轮廓线——outline 属性

有些时候，设计师可能希望在可视对象（如按钮、活动窗体域、图形地图等）周围添加一圈外部轮廓线，使对象突出显示。当然这里的外轮廓线与元素边框是不同的，外轮廓线不占用网页布局空间，且不一定是矩形，外轮廓线属于动态样式，只有当对象获取焦点或者被激活时呈现。

outline 属性可以定义块元素的外轮廓线，该属性在 CSS 2.1 规范中已被明确定义，但是并未得到各主流浏览器的广泛支持，CSS 3 增强了该特性。outline 属性的基本语法如表 5.12 所示。

表5.12 outline属性的基本语法

语法项目	说明
值	[outline-color] [outline-style] [outline-width] [outline-offset] inherit
初始值	根据具体的元素而定
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <outline-color>：定义轮廓边框颜色。
- <outline-style>：定义轮廓边框样式。
- <outline-width>：定义轮廓边框宽度。

- <outline-offset>：定义轮廓边框偏移位置的数值。
- inherit：默认继承。

注意，outline 属性创建的外轮廓线是画在一个框“上面”，也就是说，外轮廓线总是在顶上，不会影响该框或任何其它框的尺寸。因此，显示或不显示外轮廓线不会影响文档流，也不会破坏网页布局。

外轮廓线可能是非矩形的。例如，如果元素被分割成好几行，那么外轮廓线就至少是能够包含该元素所有框的外廓。和边框不同的是，外廓在线框的起讫端都不是开放的，它总是完全闭合的。

派生的子属性

为了方便设计师灵活定制，CSS 3 在这个属性基础上派生了四个外轮廓线属性。

- outline-color：定义轮廓边框颜色。outline-color 属性接受所有的颜色，还包括关键字 invert。invert 应该在屏幕上对像素点颜色进行一次反转。这个技巧保证焦点框可见，而不管背景颜色是什么。
- outline-style：定义轮廓边框的样式。outline-style 属性与 border-style 属性的接受值和用法相同，如 none、dotted、dashed、solid、double、groove、ridge、inset、outset。但是，hidden 属性值并不是一个合法的外轮廓样式。
- outline-width：定义轮廓边框的宽度。outline-width 属性与 border-width 属性的接受值和用法相同。
- outline-offset：定义轮廓边框偏移位置的数值。

outline 属性是个快速属性，可以设置 outline-style、outline-width 和 outline-color 属性值。注意，外轮廓线在各边都是一样的，这与边框不同，没有诸如 outline-top 或 outline-left 属性。CSS 不支持定义多个互相覆盖的外轮廓线，也没有定义如何定义因在其他元素之后而有部分不可见的框的外廓。

浏览器兼容性检测

各主流浏览器对 outline 属性的支持情况如下。

				
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✗ Firefox 3.5	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.6	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计醒目激活和焦点提示框

焦点是文档中用户交互的主题（如输入文本、选择一个按钮等），图形化的用户界面可以使用元素周围的外轮廓线来告诉用户页面上哪个元素获得了焦点。这些外轮廓线是不同于任何边框样式的，切换外轮廓线的显示和不显示都不会使文档流发生变化。浏览器如果支持交互媒介，则必须跟踪焦点在何处，而且必须显示焦点。这个可以通过动态外轮廓线和`:focus`伪类的联合使用完成。例如，在一个元素获得焦点时在周围画一个粗实线外廓，而在它活动时也画一个不同色的粗实线外廓，从而提高用户交互效果，效果如图5.17所示。



图5.17 激活获取焦点时的外轮廓线效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>outline</title>
<style type="text/css">
body { /*统一页面字体和大小*/
    font-family:"Lucida Grande", "Lucida Sans Unicode", Verdana, Arial, Helvetica, sans-serif;
    font-size:12px;
}
p, h1, form, button { /*清除常用元素的边界、补白、边框默认样式*/
    border:0;
    margin:0;
    padding:0;
}
.spacer { /*定义一个强制换行显示类*/
    clear:both;
    height:1px;
}

```

```
.myform { /* 定义表单外框样式 */
    margin: 0 auto;
    width: 400px;
    padding: 14px;
}
#stylized { /* 定制当前表单样式 */
    border: solid 2px #b7ddf2;
    background: #ebf4fb;
}
#stylized h1 {
    font-size: 14px;
    font-weight: bold;
    margin-bottom: 8px;
}
#stylized p {
    font-size: 11px;
    color: #666666;
    margin-bottom: 20px;
    border-bottom: solid 1px #b7ddf2;
    padding-bottom: 10px;
}
#stylized label { /* 定义表单标签样式 */
    display: block;
    font-weight: bold;
    text-align: right;
    width: 140px;
    float: left;
}
#stylized .small { /* 定义小字体样式类 */
    color: #666666;
    display: block;
    font-size: 11px;
    font-weight: normal;
    text-align: right;
    width: 140px;
}
#stylized input { /* 统一输入文本框样式 */
    float: left;
    font-size: 12px;
    padding: 4px 2px;
    border: solid 1px #aacfe4;
    width: 200px;
    margin: 2px 0 20px 10px;
}
#stylized button { /* 定义图形化按钮样式 */
    clear: both;
    margin-left: 150px;
    width: 125px;
    height: 31px;
    background: #666666 url(images/button.png) no-repeat;
    text-align: center;
    line-height: 31px;
    color: #FFFFFF;
    font-size: 11px;
    font-weight: bold;
}
```

设计表单内div和p通用样式
效果

```

}
input:focus, button:focus { outline: thick solid #b7ddf2 }
input:active, button:active { outline: thick solid #aaa }
</style>
</head>
<body>
<div id="stylized" class="myform">
  <form id="form1" name="form1" method="post" action=">
    <h1>登录表单</h1>
    <p>这是一个简单的演示表单...</p>
    <label>Name <span class="small">输入你的姓名</span> </label>
    <input type="text" name="textfield" id="textfield" />
    <label>Email <span class="small">输入你的电子邮箱</span> </label>
    <input type="text" name="textfield" id="textfield" />
    <label>Password <span class="small">输入登录密码</span> </label>
    <input type="text" name="textfield" id="textfield" />
    <button type="submit">登 录</button>
    <div class="spacer"></div>
  </form>
</div>
</body>
</html>

```

设计表单内文本框和按钮在被激活和获取焦点状态下时，外轮廓线的宽、样式和颜色

5.14 定义外轮廓线宽度——outline-width 属性

outline-width 属性可以单独设置外轮廓线的宽度。该属性的基本语法如表 5.13 所示。

表5.13 outline-width属性的基本语法

语法项目	说明
值	thin medium thick <length> inherit
初始值	medium
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- thin：定义细轮廓。
- medium：定义中等的轮廓。
- thick：定义粗的轮廓。
- <length>：定义轮廓粗细的值。
- inherit：默认继承。

注意：outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时，该属性才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置负长度值。

浏览器兼容性检测

各主流浏览器对 outline-width 属性的支持情况如下。

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✓ Firefox 3.5	✗ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✓ Firefox 3.6	✗ Opera 10.5		✓ Chrome 4.0.x

5.15 定义外轮廓线样式——outline-style 属性

outline-style 属性可以单独设置外轮廓线的样式。该属性的基本语法如表 5.14 所示。

表 5.14 outline-style 属性的基本语法

语法项目	说明
值	auto <border-style> inherit
初始值	none
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- auto：根据浏览器自动设置。
- <border-style>：沿用边框样式。包括 none、dotted、dashed、solid、double、groove、ridge、inset、outset。详细说明请参阅 CSS 2.1 中有关 border-style 属性值的内容。
- inherit：默认继承。

浏览器兼容性检测

该属性的浏览器兼容性与 outline-width 属性相同，详细说明请参考 outline-width 属性。

5.16 定义外轮廓线颜色——outline-color 属性

outline-color 属性可以单独设置外轮廓线的颜色。该属性的基本语法如表 5.15 所示。

表5.15 outline-color属性的基本语法

语法项目	说明
值	<color> invert inherit
初始值	invert
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <color>：可以是颜色名，如 red；函数值，如 rgb(255,0,0)；或者是十六进制值，如 #ff0000。
- invert：执行颜色反转（逆向的颜色）。这样可以确保外轮廓线在不同的背景颜色中都是可见的。
- inherit：默认继承。

注意，轮廓的样式不能是 none，否则轮廓不会出现。

浏览器兼容性检测

该属性的浏览器兼容性与 outline-width 属性相同，详细说明请参考 outline-width 属性。

5.17 定义外轮廓线位移——outline-offset 属性

outline-offset 属性可以单独设置外轮廓线的偏移位置。该属性的基本语法如表 5.16 所示。

表5.16 outline-offset属性的基本语法

语法项目	说明
值	<length> inherit
初始值	0

(续)

语法项目	说明
适用于	所有元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <length>：定义轮廓距离容器的值。
- inherit：默认继承。

浏览器兼容性检测

该属性的浏览器兼容性与 outline-width 属性相同，详细说明请参考 outline-width 属性。

实战体验：放大激活和焦点提示框

下面我们继续在前面的案例的基础上，通过 outline-offset 属性放大外轮廓线，使其看起来更大方，如图 5.18 所示。

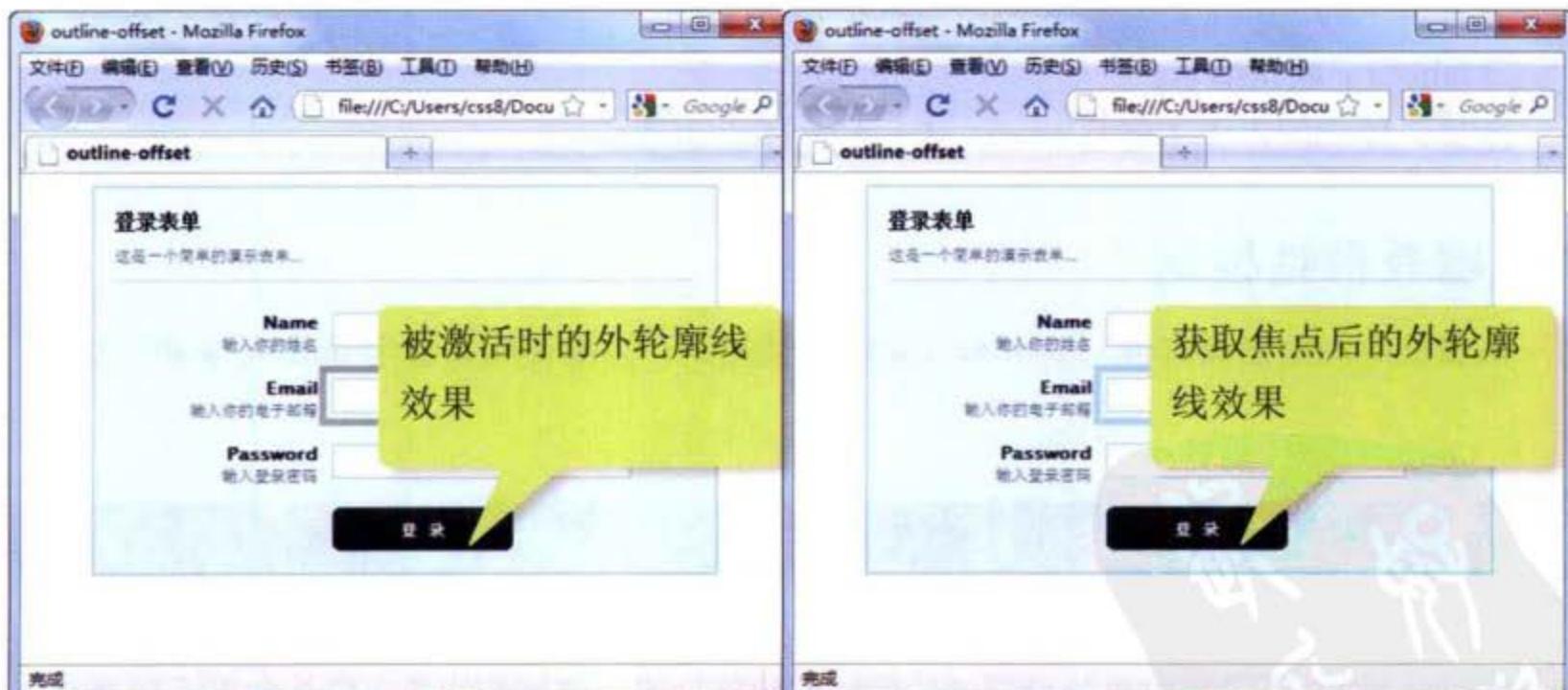


图 5.18 放大激活和获取焦点时的外轮廓线效果

在前面示例的样式基础上，再添加如下样式即可，其他相同的代码就不再显示。

```
<style type="text/css">
...
input:focus, button:focus { outline: thick solid #b7ddf2 }
```

```

input:active, button:active { outline: thick solid #aaa }
input:active, button:active { outline-offset: 4px; }
input:focus, button:focus { outline-offset: 4px; }
</style>

```

通过outline-offset属性
放大外轮廓线

5.18 定义导航序列号——nav-index 属性

在 HTML 4 和 XHTML 1 语言中，曾定义了 tabindex 标签属性，这个属性为当前元素指定了其在当前文档中导航的序列号。导航的序列号可以修改页面中元素通过键盘操作获得焦点的顺序。该属性可以存在于嵌套的页面元素中。

CSS 3 新增加了 nav-index 属性，用来代替标签属性 tabindex。为了使浏览器能按顺序获取焦点，页面元素需要遵循如下规则：

- 该元素支持 nav-index 属性，而被赋予正整数属性值的元素将会被优先导航。浏览器将按照 nav-index 属性值从小到大的顺序进行导航。属性值无须按顺序，也无须以特定的值开始。拥有同一 nav-index 属性值的元素将以它们在文档流中出现的顺序进行导航。
- 对于那些不支持 nav-index 属性，或者 nav-index 属性值为 auto 的元素，将以它们在文档流中出现的顺序进行导航。
- 对那些禁用的元素，将不参与导航的排序。

用户实际上使用的开始导航和激活页面元素的快捷键依赖于浏览器的设置，例如，通常 Tab 键用于按顺序导航，而 Enter 键则用于激活选中的元素。

浏览器一般也会定义反向导航的快捷键。当通过 Tab 键导航到序列的结束或者开始时，浏览器可能会循环到导航序列的开始或者结束。Shift+Tab 组合键通常用于反向导航。

nav-index 属性的基本语法如表 5.17 所示。

表5.17 navindex属性的基本语法

语法项目	说明
值	auto <number> inherit
初始值	auto
适用于	所有可用元素
可否继承	否
百分比	N/A
媒介	交互

取值简单说明：

- auto：浏览器默认的顺序。

- <number>：必须是正整数，该数字指定了元素的导航顺序。1 意味着最先被导航。当若干个元素的 nav-index 值相同时，则按照文档的先后顺序进行导航。
- inherit：默认继承。

浏览器兼容性检测

各主流浏览器对 nav-index 属性的支持情况如下。

				
IE 6	Firefox 1.5	Opera 9.0	Safari 3.0	Chrome 1.0.x
IE 7	Firefox 2.0	Opera 9.6	Safari 4.0	Chrome 2.0.x
IE 8	Firefox 3.5	Opera 10.0		Chrome 3.0.x
IE 9	Firefox 3.6	Opera 10.5		Chrome 4.0.x

实战体验：调整表单输入框的键盘激活顺序

对于下面这个表单案例来说，传统做法是使用 tabindex 属性来改变表单输入框的键盘激活和响应顺序，如图 5.16 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>nav-index</title>
</head>
<body>
<div id="leftSide">
    <fieldset>
        <legend>个人信息登记表</legend>
        <form action="" method="POST" class="form">
            <label for="name">姓名</label>
            <div class="div_textbox">
                <input name="name" type="text" class="textbox" tabindex="1" id="name" value="" />
            </div>
            <label for="address">地址</label>
            <div class="div_textbox">
                <input name="address" type="text" class="textbox" tabindex="4" id="address" value="" />
            </div>
            <label for="city">区/县</label>
            <div class="div_textbox">
                <input name="city" type="text" class="textbox" tabindex="3" id="city" value="" />
            </div>
            <label for="country">省/市</label>
        </form>
    </fieldset>
</div>
```

```

<div class="div_texbox">
<input name="country" type="text" class="textbox" tabindex="2" id="country" value="" />
</div>
<div class="button_div">
<input name="Submit" type="button" value="Submit" tabindex="5" class="buttons" />
</div>
</form>
</fieldset>
</div>
</body>
</html>

```

现在使用 nav-index 直接在 CSS 样式表中调整表单输入框的键盘焦点顺序，如图 5.19 所示。

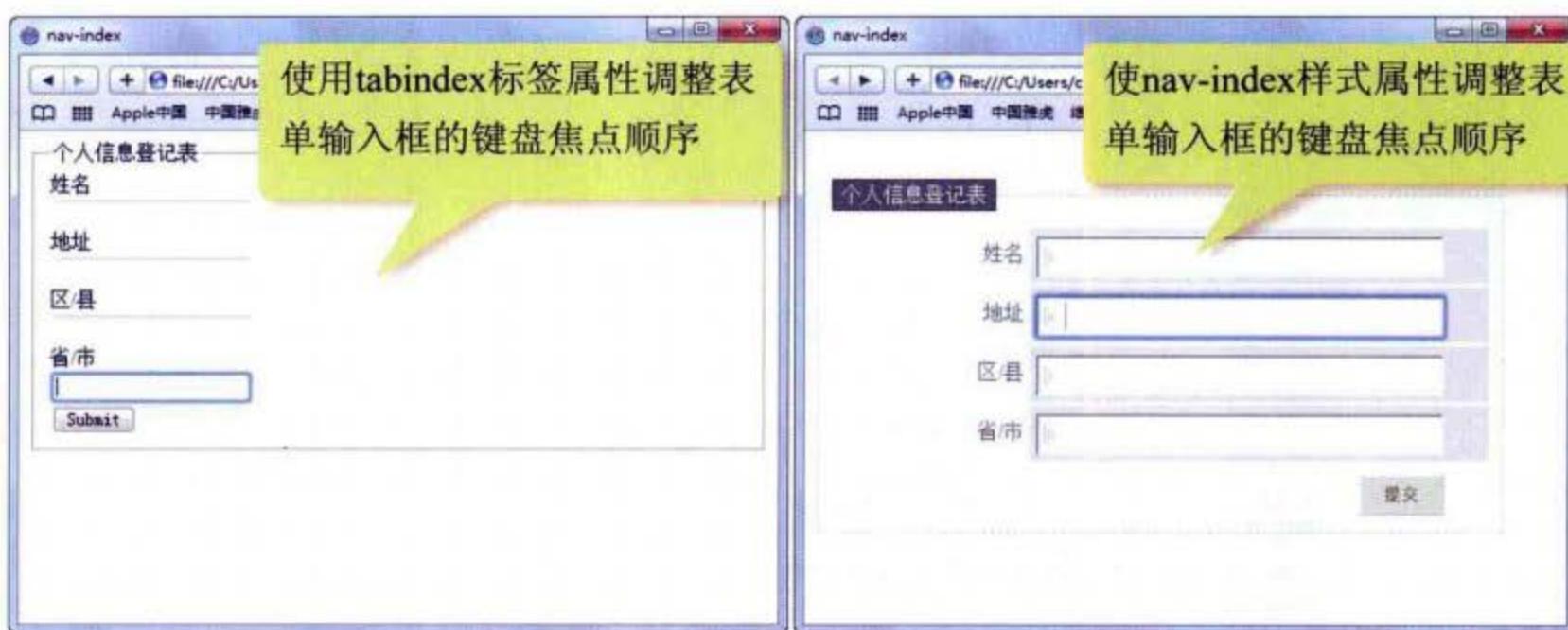


图 5.19 改变表单输入框的键盘焦点顺序的演示效果

使用 CSS 设计和调整键盘焦点顺序的代码如下所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>nav-index</title>
<style type="text/css">
/*统一默认样式*/
body { background-color:#F7F7F7; }
fieldset {border:1px dashed #CCC; padding:10px; }
legend { font-family:Arial, Helvetica, sans-serif; color:#fff; background: #666; border: 1px solid #333; padding: 2px 6px; }
label { width:142px; height:32px; margin:3px 2px 0 0; padding:11px 0 0 6px; float:left; color:#666; text-align:right; }
/*清除表单默认样式*/
.form {margin:0; padding:0; }
/*定制表单外框样式*/
#leftSide { width:530px; padding-top:30px; float:left; }
/*定义表单输入框区块样式*/

```

```

.div_texbox { width:347px; float:right; background-color:#E6E6E6; height:35px; margin-top:3px; padding-top:5px; padding-bottom:3px; padding-left:5px; }
.textbox {background-image: url(images/16t.gif); background-repeat: no-repeat; background-position:left; width:285px; font:normal 18px Arial; color: #999999; padding:3px 5px 3px 19px; }
/*定义表单输入框获取焦点和鼠标经过时的样式*/
.username:focus, .username:hover { background-color:#F0FFE6; }
/*定义按钮样式*/
.button_div { width:287px; float:right; text-align:right; height:35px; margin-top:3px; padding:5px 32px 3px; }
.buttons { padding:6px 14px; border:2px solid; border-color: #fff #d8d8d0 #d8d8d0 #fff; background: #e3e3db; color: #989070; font-weight:bold; }
/*定义表单输入框的键盘焦点顺序*/
input.nav1 { nav-index:1; }
input.nav2 { nav-index:2; }
input.nav3 { nav-index:3; }
input.nav4 { nav-index:4; }
input.nav5 { nav-index:5; }
</style>
</head>
<body>
<div id="leftSide">
  <fieldset>
    <legend>个人信息登记表</legend>
    <form action="" method="POST" class="form">
      <label for="name">姓名</label>
      <div class="div_texbox">
        <input name="name" type="text" class="textbox nav1" id="name" value="" />
      </div>
      <label for="address">地址</label>
      <div class="div_texbox">
        <input name="address" type="text" class="textbox nav4" id="address" value="" />
      </div>
      <label for="city">区/县</label>
      <div class="div_texbox">
        <input name="city" type="text" class="textbox nav3" id="city" value="" />
      </div>
      <label for="country">省/市</label>
      <div class="div_texbox">
        <input name="country" type="text" class="textbox nav2" id="country" value="" />
      </div>
      <div class="button_div">
        <input name="Submit" type="button" value="提交" class="buttons nav5" />
      </div>
    </form>
  </fieldset>
</div>
</body>
</html>

```

5.19 定义方向键控制顺序

输入设备默认四个方向键将根据文档流的顺序依次控制元素的焦点切换，但为了得到更好的用户体验，CSS 3 增加定义了切换焦点的控制顺序方向键的特性。它主要由四个属性来配合实现：

- nav-up : 控制向上方向键。
- nav-right : 控制向右方向键。
- nav-down : 控制向下方向键。
- nav-left : 控制向左方向键。

这些属性的基本语法如表 5.18 所示。

表5.18 方向键控制属性的基本语法

语法项目	说明
值	auto <id> [current root <target-name>]? inherit
初始值	auto
适用于	所有可用元素
可否继承	否
百分比	N/A
媒介	交互

取值简单说明：

- auto : 根据浏览器默认的顺序切换。
- <id> : 要切换元素的 id 名。
- root | <target-name> : 该参数不能以 "_" 命名，指出 frameset 目标页面之间的元素焦点切换。如果指定的目标页面不存在，则被视为当前页面的焦点。该属性是以关键节点 "root" 标示，浏览器将把整个 frameset 框架页定为目标。
- inherit : 继承。

● 浏览器兼容性检测

各主流浏览器对 nav-up、nav-right、nav-down 和 nav-left 这四个属性的支持情况如下。

兼容性检测				
IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
IE 8	✗ Firefox 3.5	✗ Opera 10.0		✗ Chrome 3.0.x
IE 9	✗ Firefox 3.6	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：正确定义键盘控制键顺序

nav-up、nav-right、nav-down 和 nav-left 属性虽然在网页设计中不是很常用，浏览器对

其支持也不是很完美，但是读者应该知道它们的用法。下面这个案例将帮助读者了解这些键盘方向键样式控制的一般方法，演示效果如图 5.20 所示。



图 5.20 键盘方向键控制顺序的演示效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>nav-up、nav-right、nav-down、nav-left</title>
<style type="text/css">
body { background:url(images/img20.jpg) center no-repeat; }
button { position:absolute; }
button#b1 { /*顶部按钮*/
    top:0; left:50%;
    nav-index:1;
    nav-right:#b2; nav-down:#b2;
    nav-left:#b4; nav-up:#b4; }

button#b2 { /*右侧按钮*/
    top:50%; left:100%;
    nav-index:2;
    nav-right:#b3; nav-down:#b3;
    nav-left:#b1; nav-up:#b1; }

button#b3 { /*底部按钮*/
    top:100%; left:50%;
    nav-index:3;
    nav-right:#b4; nav-down:#b4;
    nav-left:#b2; nav-up:#b2; }

button#b4 { /*左侧按钮*/
    top:50%; left:0;
    nav-index:4;
    nav-right:#b1; nav-down:#b1;
    nav-left:#b3; nav-up:#b3; }

</style>
</head>

```

按向左或向上方向键，将焦点移动到按钮4上
按向右或向下方向键，将焦点移动到按钮2上

按向左或向上方向键，将焦点移动到按钮1上
按向右或向下方向键，将焦点移动到按钮3上

按向左或向上方向键，把焦点移动到按钮2上
按向右或向下方向键，把焦点移动到按钮4上

按向左或向上方向键，把焦点移动到按钮3上
按向右或向下方向键，把焦点移动到按钮1上

```

<body>
<div>
    <button id="b1">按钮1（位于窗口顶部）</button>
    <button id="b2">按钮2（位于窗口右侧）</button>
    <button id="b3">按钮3（位于窗口底部）</button>
    <button id="b4">按钮4（位于窗口左侧）</button>
</div>
</body>
</html>

```

5.20 为元素添加内容——content 属性

content 属性属于内容生成和替换模块 (<http://www.w3.org/TR/css3-content/>)，该属性能够为指定元素添加内容。实际上内容生成和替换行为已经超越了 CSS 样式表的核心功能，这部分功能替代了原需 JavaScript 脚本来实现的角色任务。不过 content 属性比较实用，它能够满足设计师在样式设计中临时添加非结构性的样式服务标签，或者添加补充说明性内容等的需要。content 属性的基本语法如表 5.19 所示。

表5.19 content属性的基本语法

语法项目	说明
值	normal string attr() uri() counter() none
初始值	normal
适用于	所有可用元素
可否继承	否
百分比	N/A
媒介	所有

取值简单说明：

- ❑ normal：默认值。
- ❑ string：插入文本内容。
- ❑ attr()：插入元素的属性值。
- ❑ uri()：插入一个外部资源，如图像、音频、视频或浏览器支持的其他任何资源。
- ❑ counter()：计数器，用于插入排序标识。
- ❑ none：无任何内容。

浏览器兼容性检测

各主流浏览器对 content 属性的支持情况如下。

✗ IE 6	✓ Firefox 1.5	✓ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✓ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✓ IE 8	✓ Firefox 3.5	✓ Opera 10.0		✓ Chrome 3.0.x
✓ IE 9	✗ Firefox 3.6	✓ Opera 10.5		✓ Chrome 4.0.x

content 应用分解

content 属性的简单用法就是在文档中添加内容，例如，为 div 元素添加一段文本，演示效果如图 5.21 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>content</title>
<style type="text/css">
div {
    text-align:center;
    padding:50px;
    border:solid 1px red;
}
div:before { content:"当前浏览器支持 content 属性"; }
</style>
</head>
<body>
<div></div>
</body>
</html>
```



图 5.21 使用 content 属性在当前元素内添加文本演示效果

在下面的示例中，我们为 div 元素插入一幅图像，演示效果如图 5.22 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>content</title>
<style type="text/css">
div {
    text-align:center;
    padding:50px;
    border:solid 1px red;
}
div { content: url(images/bg2.jpg); }
</style>
</head>
<body>
<div></div>
</body>
</html>
```



图5.22 使用content属性在当前元素内插入图像的演示效果

第6章

CSS 3 多列布局

在 CSS 2.1 及其以前的版本中，如果需要设计多列布局，网页设计师常用两种基本方法：浮动布局和定位布局。浮动布局比较灵活，但是容易发生错位，影响网页的整体效果，这种布局方式需要设计师编写大量附加样式代码，或者添加无用的换行标签，这些都给网页设计增添了不必要的工作量；定位布局可以实现精确定位，但是这种布局方式无法满足模块的自适应能力，以及模块之间的文档流联动的需要。

为了解决多列布局的难题，CSS 3 新增了 Multi-column Layout（Multiple Columns 布局特性），即多列自动布局功能（可参考 <http://www.w3.org/TR/css3-multicol/>）。利用多列布局属性可以自动将内容按指定的列数排列，这种特性特别适合报纸和杂志类网页布局。本章将详细讲解多列布局的基本属性、用法和实战技巧。

6.1 定义多列布局——columns 属性

columns 是多列布局特性的基本属性，类似边框特性中的 border 属性，该属性可以同时定义多列的数目和每列的宽度。columns 属性的基本语法如表 6.1 所示。

表 6.1 columns 属性的基本语法

语法项目	说明
值	<column-width> <column-count>
初始值	根据元素个别属性而定
适用于	不可替换的块元素、行内块元素、单元格，但是表格元素除外
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <column-width>：定义每列的宽度，用法请参阅 column-width 属性。
- <column-count>：定义列数，用法请参阅 column-count 属性。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-columns 私有属性，其他类型的浏览器暂时不支持 columns 属性，也没有定义支持 columns 的私有属性。各主流浏览器对 columns 属性的支持情况如下。

✖ IE 6	✖ Firefox 1.5	✖ Opera 9.0	✔ Safari 3.0	✔ Chrome 1.0.x
✖ IE 7	✖ Firefox 2.0	✖ Opera 9.6	✔ Safari 4.0	✔ Chrome 2.0.x
✖ IE 8	✖ Firefox 3.0	✖ Opera 10.0		✔ Chrome 3.0.x
✖ IE 9	✖ Firefox 3.5	✖ Opera 10.5		✔ Chrome 4.0.x

实战体验：设计文章多栏显示

在传统报刊杂志中，文章常用多栏显示，这样方便阅读。在网页中显示大段文字时，不妨采用这种多栏显示，能够方便利用浏览器进行阅读。使用浮动布局实现这种效果往往会在两个难题：一是多列浮动显示不容易进行控制；二是多列显示后各列内容无法互通，这样就为后期加工带来诸多不便。如果使用 CSS 3 多列布局特性，这些问题就可以轻松解决，如图 6.1 所示。

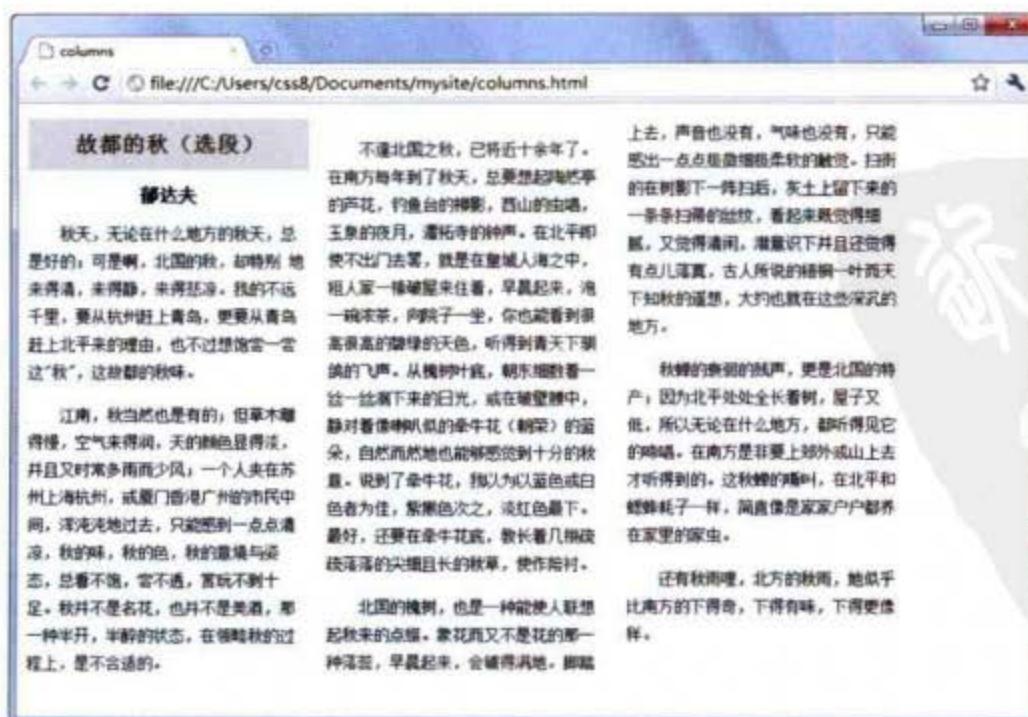


图 6.1 设计文章多栏显示的演示效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css" media="screen">
body {
    -webkit-columns: 250px 3;
    columns: 250px 3;
}
设计网网页文档分3栏显示，每
栏宽度为250像素
h1 { color:#333333; background:#DCDCDC; padding:5px 8px; font-size:20px;text-align:center;
padding:12px;}
h2 { font-size:16px; text-align:center; }
p {color:#333333; font-size:14px; line-height:180%; text-indent:2em; }
</style>
<title>columns</title>
</head>
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p> 秋天，无论在什么地方的秋天，总是好的；可是啊，北国的秋，却特别地来得清，来得静，来得悲凉。我的不远千里，要从杭州赶上青岛，更要从青岛赶上北平来的理由，也不过想饱尝一尝这“秋”，这故都的秋味。 </p>
<p>江南，秋当然也是有的；但草木凋得慢，空气来得润，天的颜色显得淡，并且又时常多雨而少风；一个人夹在苏州上海杭州，或厦门香港广州的市民中间，混混沌沌地过去，只能感到一点点清凉，秋的味，秋的色，秋的意境与姿态，总看不饱，尝不透，赏玩不到十足。秋并不是名花，也并不是美酒，那一种半开，半醉的状态，在领略秋的过程上，是不合适的。 </p>
<p>不逢北国之秋，已将近十余年了。在南方每年到了秋天，总要想起陶然亭的芦花，钓鱼台的柳影，西山的虫唱，玉泉的夜月，潭柘寺的钟声。在北平即使不出门去罢，就是在皇城人海之中，租人家一椽破屋来住着，早晨起来，泡一碗浓茶，向院子一坐，你也能看到很高很高的碧绿的天色，听得到青天下驯鸽的飞声。从槐树叶底，朝东细数着一丝一丝漏下来的日光，或在破壁腰中，静对着像喇叭似的牵牛花（朝荣）的蓝朵，自然而然地也能够感觉到十分的秋意。说到了牵牛花，我以为以蓝色或白色者为佳，紫黑色次之，淡红色最下。最好，还要在牵牛花底，教长着几根疏疏落落的尖细且长的秋草，使作陪衬。 </p>
<p> 北国的槐树，也是一种能使人联想起秋来的点缀。像花而又不是花的那一种落蕊，早晨起来，会铺得满地。脚踏上去，声音也没有，气味也没有，只能感出一点点极微细极柔软的触觉。扫街的在树影下一阵扫后，灰土上留下来的一条条扫帚的丝纹，看起来既觉得细腻，又觉得清闲，潜意识下并且还觉得有点儿落寞，古人所说的梧桐一叶而天下知秋的遥想，大约也就在这些深沉的地方。 </p>
<p>秋蝉的衰弱的残声，更是北国的特产：因为北平处处全长着树，屋子又低，所以无论在什么地方，都听得见它的啼唱。在南方是一定要上郊外或山上去才听得到的。这秋蝉的嘶叫，在北平和蟋蟀耗子一样，简直像是家家户户都养在家里的家虫。 </p>
<p> 还有秋雨哩，北方的秋雨，她似乎比南方的下得奇，下得有味，下得更像样。 </p>
</body>
</html>

```

6.2 定义列宽度——column-width 属性

column-width 属性可以定义指定对象单列显示的宽度，该属性可以与其他多列布局属性配合使用，也可以单独使用。column-width 属性的基本语法如表 6.2 所示。

表 6.2 column-width 属性的基本语法

语法项目	说明
值	<length> auto
初始值	auto

(续)

语法项目	说明
适用于	不可替换的块元素、行内块元素、单元格，但是表格元素除外
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <length>：由浮点数字和单位标识符组成的长度值。不可为负值。
- auto：根据浏览器计算值自动设置。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-column-width 私有属性，Mozilla Gecko 引擎支持 -moz-column-width 私有属性，Presto 引擎和 IE 浏览器暂时不支持 column-width 属性，也没有定义支持 column-width 属性的私有属性。

借助兼容方式，各主流浏览器对 column-width 属性的支持情况如下。

				
✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计固定宽度的栏目版面

下面继续以上一节案例为基础，演示 column-width 属性在多列布局中的应用。column-width 可以与其他多列布局属性配合使用，设计指定固定列数、列宽的布局效果；也可以单独使用，限制模块的单列宽度，当超出宽度时，则会自动以多列进行显示。

例如，设计 body 元素的列宽度为 300 像素：如果网页内容能够在单列内显示，则会以单列显示；如果窗口足够宽，且内容很多，则会在多列中进行显示。演示效果如图 6.2 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

禁书网 PDF

```

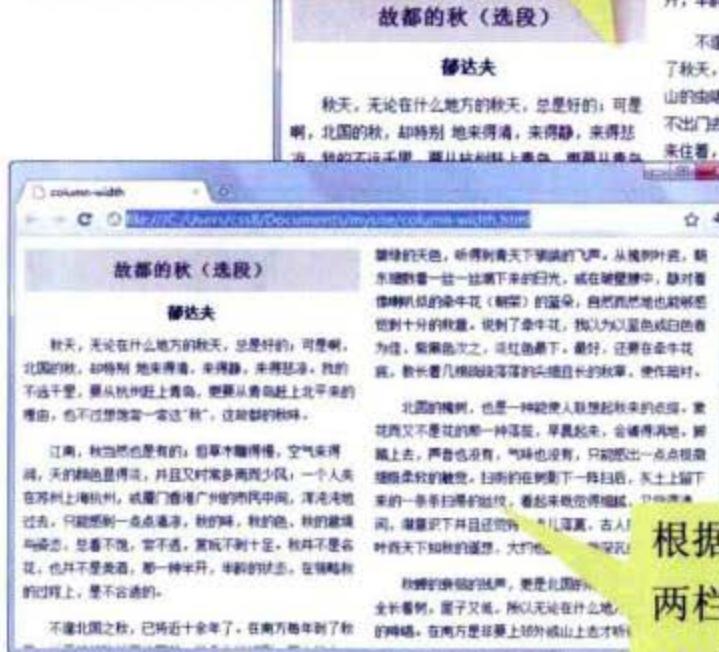
<style type="text/css" media="screen">
body {
    -webkit-column-width:300px;
    -moz-column-width:300px;
    column-width:300px;
}
...
</style>
<title>column-width</title>
</head>
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p>...</p>
</body>
</html>

```

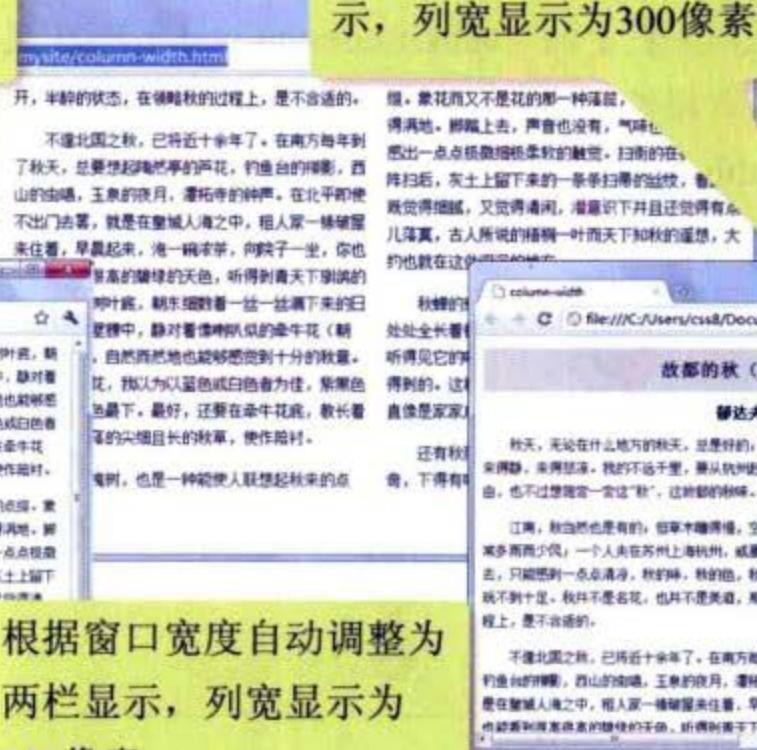
省略的样式代码

定义网页列宽为300像素，则网页中每个栏目的最大宽度为300像素

根据窗口宽度自动调整为三栏显示，列宽显示为300像素



根据窗口宽度自动调整为一栏显示，列宽显示为300像素



根据窗口宽度自动调整为两栏显示，列宽显示为300像素

6.3 定义列数——column-count 属性

column-count 属性可以定义指定对象显示的列数，该属性的基本语法如表 6.3 所示。

表6.3 column-count属性的基本语法

语法项目	说明
值	<integer> auto
初始值	auto
适用于	不可替换的块元素、行内块元素、单元格，但是表格元素除外
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <integer>：定义栏目的列数，取值为大于0的整数。如果column-width和column-count属性没有明确值，则该值为最大列数。
- auto：根据浏览器计算值自动设置。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-column-count 私有属性，Mozilla Gecko 引擎支持 -moz-column-count 私有属性，Presto 引擎（包括 Opera 浏览器等）和 IE 浏览器暂时不支持 column-count 属性，也没有定义支持 column-count 属性的私有属性。

借助兼容方式，各主流浏览器对 column-count 属性的支持情况如下。

    				
× IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
× IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
× IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
× IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计固定列数的版面

在前面示例的基础上，如果定义网页列数为3，你会看到，不管浏览器窗口怎么调整，页面内容总是遵循3列布局，演示效果如图6.3所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

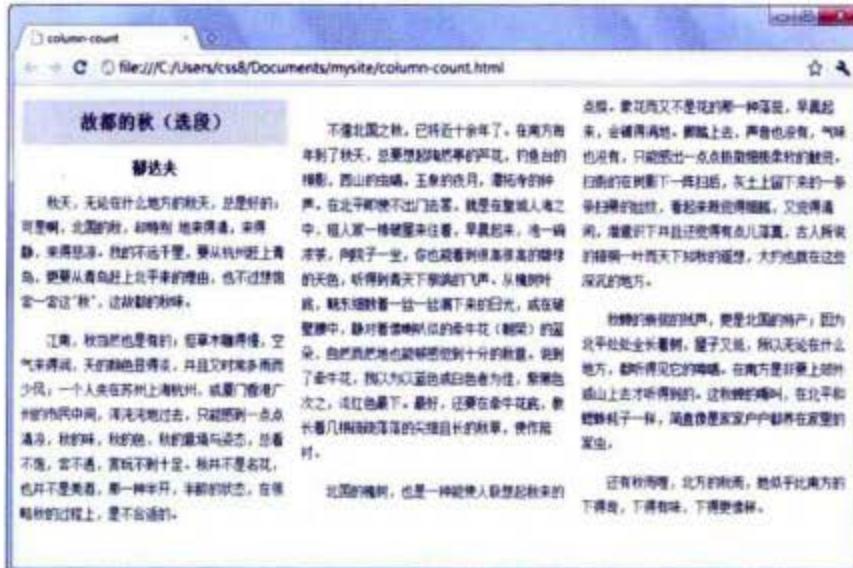
```

```

<style type="text/css" media="screen">
body {
    -webkit-column-count: 3;
    -moz-column-count: 3;
    column-count: 3;
}
...
</style>
<title>column-width</title>
</head>
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p>...</p>
</body>
</html>

```

省略的文档内容



根据窗口宽度自动调整列宽，但是整个页面总是显示为3列

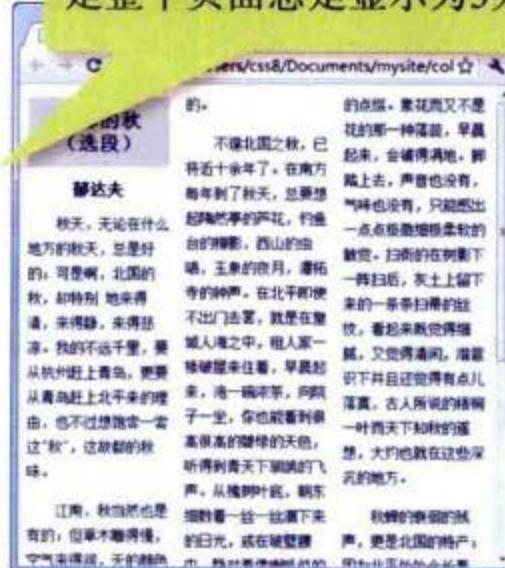


图 6.3 设计固定列数的多列布局效果

6.4 定义列间距——column-gap 属性

column-gap 属性可以定义两列之间的间距，该属性的基本语法如表 6.4 所示。

表 6.4 column-gap 属性的基本语法

语法项目	说明
值	normal <length>
初始值	normal
适用于	多列布局元素

(续)

语法项目	说明
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- normal：根据浏览器默认设置进行解析，一般为 1em。
- <length>：由浮点数字和单位标识符组成的长度值，不可为负值。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-column-gap 私有属性，Mozilla Gecko 引擎支持 -moz-column-gap 私有属性，Presto 引擎（包括 Opera 浏览器等）和 IE 浏览器暂时不支持 column-gap 属性，也没有定义支持 column-gap 属性的私有属性。

借助兼容方式，各主流浏览器对 column-gap 属性的支持情况如下。

✗ IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
✗ IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计疏朗的文档版面

通过 column-gap 和 line-height 属性配合使用，设计师可以把文档版面设计得疏朗大方，以方便阅读。例如，在前面示例的基础上，如果列间距为 3em、行高为 2.5em，则页面内文字内容看起来更明晰，演示效果如图 6.4 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css" media="screen">
body {
    -webkit-column-count:3;
    -moz-column-count:3;
    column-count:3;
}

```

-webkit-column-count:3;
-moz-column-count:3;
column-count:3;

定义页面内容显示为
3列

```

    -webkit-column-gap: 3em;
    -moz-column-gap: 3em;
    column-gap: 3em;
    line-height: 2.5em; /* 定义页面文本行高 */
}

... 省略的样式代码

</style>
<title>column-width</title>
</head>
<title>column-width</title>
</head>
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p>...</p> 省略的文档内容
</body>
</html>

```

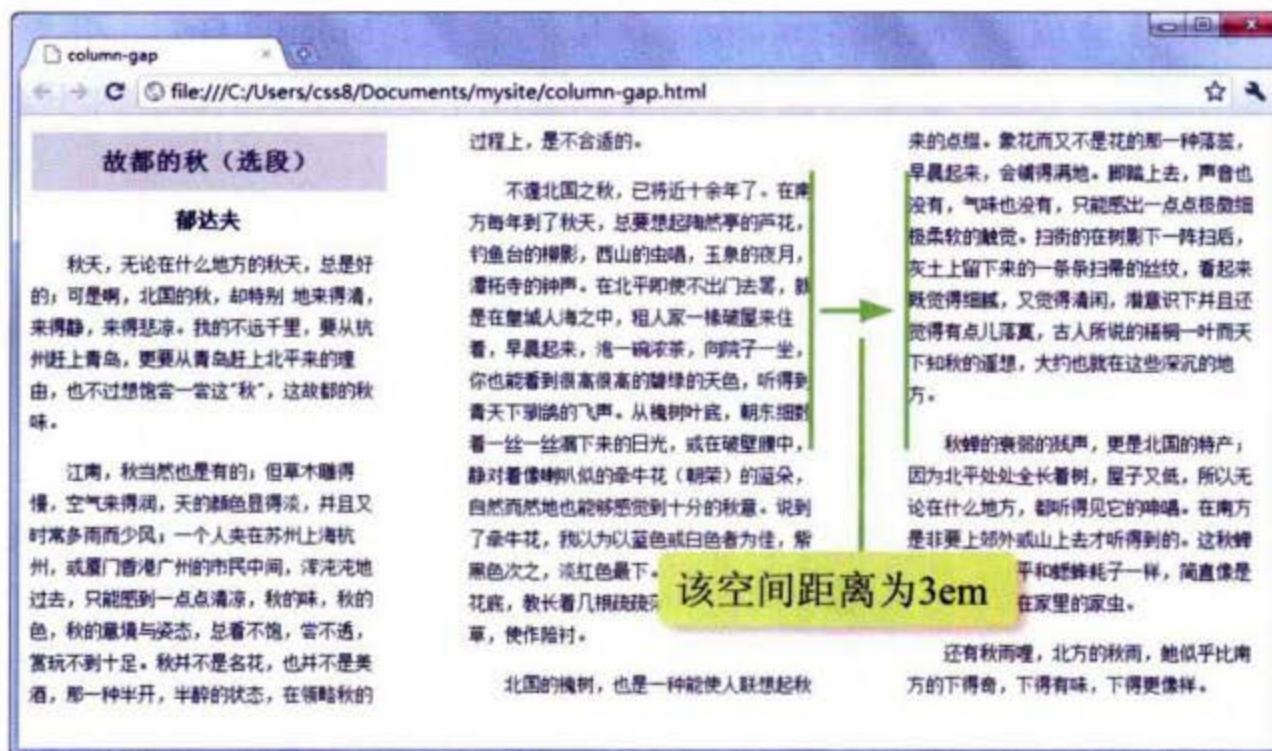


图 6.4 设计疏朗的页面布局

6.5 定义列边框样式——column-rule 属性

column-rule 属性可以定义每列之间边框的宽度、样式和颜色。该属性的基本语法如表 6.5 所示。

表 6.5 column-rule 属性的基本语法

语法项目	说明
值	<length> <style> <color> <transparent>

(续)

语法项目	说明
初始值	根据个别属性而定
适用于	多列布局元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- <length>：由浮点数字和单位标识符组成的长度值，不可为负值。功能与 column-rule-width 属性相同。
- <style>：定义列边框样式。功能与 column-rule-style 属性相同。
- <color>：定义列边框的颜色。功能与 column-rule-color 属性相同。
- <transparent>：设置边框透明显示。

派生的子属性

为了方便设计师灵活设计列边框，CSS 3 在 column-rule 属性的基础上派生了三个列边框属性。

- column-rule-color：定义列边框颜色。column-rule-color 属性接受所有的颜色。Webkit 引擎支持 -webkit-column-rule-color 私有属性，Mozilla Gecko 引擎支持 -moz-column-rule-color 私有属性。
- column-rule-width：定义列边框宽度。column-rule-width 属性接受任意浮点数，但不接受负值。Webkit 引擎支持 -webkit-column-rule-width 私有属性，Mozilla Gecko 引擎支持 -moz-column-rule-width 私有属性。
- column-rule-style：定义列边框样式。column-rule-style 属性值与 border-style 属性值相同，包括 none、hidden、dotted、dashed、solid、double、groove、ridge、inset、outset。Webkit 引擎支持 -webkit-column-rule-style 私有属性，Mozilla Gecko 引擎支持 -moz-column-rule-style 私有属性。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-column-rule 私有属性，Mozilla Gecko 引擎支持 -moz-column-rule 私有属性，Presto 引擎和 IE 浏览器暂时不支持 column-rule 属性，也没有定义支持 column-rule 属性的私有属性。

借助兼容方式，各主流浏览器对 column-rule 属性的支持如下。

× IE 6	✓ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
× IE 7	✓ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.x
× IE 8	✓ Firefox 3.0	✗ Opera 10.0		✓ Chrome 3.0.x
× IE 9	✓ Firefox 3.5	✗ Opera 10.5		✓ Chrome 4.0.x

实战体验：为多列布局版面设计边框效果

为列边框设计样式，能够有效区分各个栏目列之间的关系。例如，在前面示例的基础上，如果为每列之间的边框定义一个虚线分割线、线宽为 2 像素、灰色显示，则演示效果如图 6.5 所示。

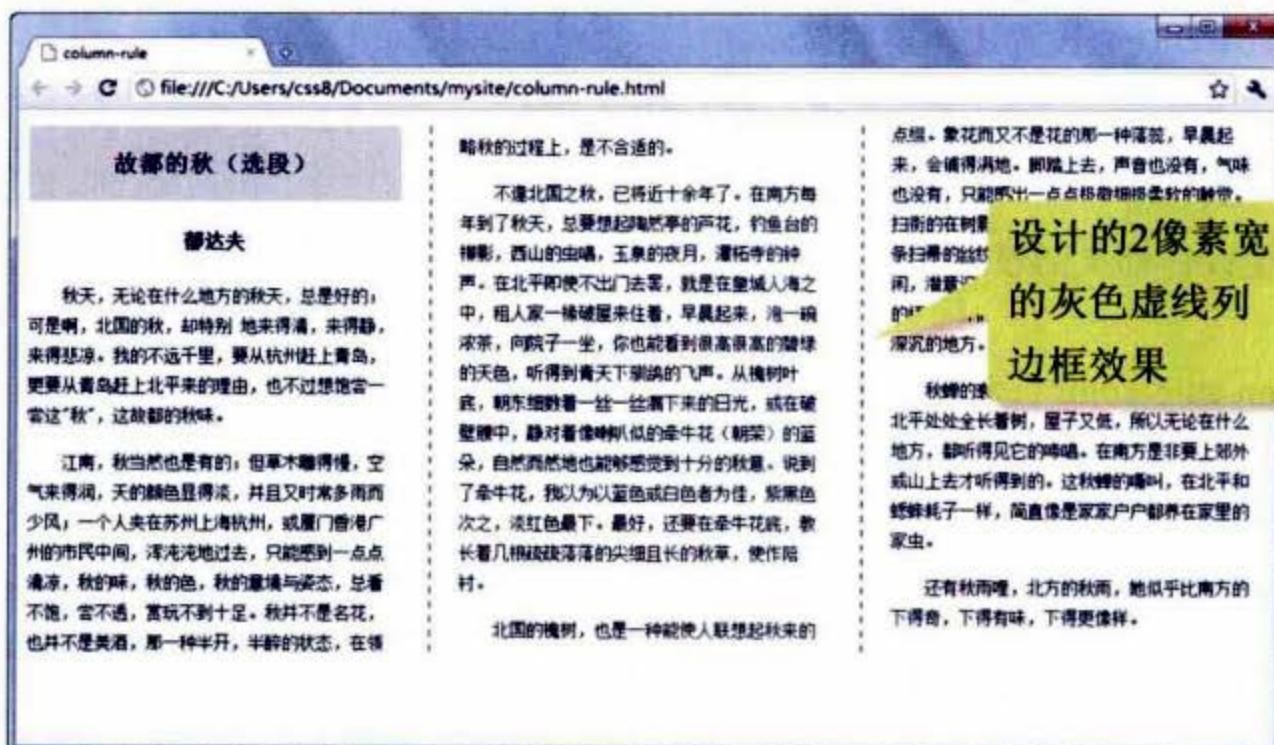


图 6.5 设计列边框效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<style type="text/css" media="screen">
body {
    -webkit-column-count: 3;
    -moz-column-count: 3;
    column-count: 3;
    -webkit-column-gap: 3em;
    -moz-column-gap: 3em;
    column-gap: 3em;
    line-height: 2.5em;
}

```

定义页面内容显示为
3列

定义列间距为3em，
默认为1em

```

-webkit-column-rule:dashed 2px gray;
-moz-column-rule:dashed 2px gray;
column-rule:dashed 2px gray;
}

... 省略的样式代码

</style>
<title>column-rule</title>
</head>
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p>...</p>
</body>
</html>

```

省略的文档内容

定义列边框为2像素宽的灰色虚线

6.6 定义跨列显示——column-span 属性

column-span 属性可以定义元素跨列显示，也可以设置元素单列显示。该属性的基本语法如表 6.6 所示。

表 6.6 column-span 属性基本语法

语法项目	说明
值	1 all
初始值	1
适用于	静态的、非浮动元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- 1：只在本栏中显示。
- all：将横跨所有列，并定位在列的 Z 轴之上。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-column-span 私有属性，Mozilla Gecko 引擎、Presto 引擎和 IE 浏览器暂时不支持 column-span 属性，也没有定义支持 column-span 属性的私有属性。

借助兼容方式，各主流浏览器对 column-span 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 3.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 4.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✓ Chrome 8.0.x

实战体验：设计文章标题跨列显示

在纸质报刊杂志中，经常会看到文章标题跨列居中显示。现在使用 column-span 属性可以实现这种效果，演示效果如图 6.6 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css" media="screen">
body {
    -webkit-column-count:3;
    -moz-column-count:3;
    column-count:3;
    -webkit-column-gap:3em;
    -moz-column-gap:3em;
    column-gap:3em;
    line-height:2.5em;
    -webkit-column-rule:dashed 2px gray;
    -moz-column-rule:dashed 2px gray;
    column-rule:dashed 2px gray;
}
h1 {
    color:#333333; background:#DCDCDC;
    font-size:20px; text-align:center;
    padding:12px;
    -webkit-column-span:all;
    -moz-column-span:all;
    column-span:all;
}
h2 {
    font-size:16px; text-align:center;
    -webkit-column-span:all;
    -moz-column-span:all;
    column-span:all;
}
p { color:#333333; font-size:14px; line-height:180%; text-indent:2em; }
</style>
<title>column-span</title>
</head>
```

The diagram highlights several CSS properties in the code and connects them to callout boxes explaining their purpose:

- Definition:** `-webkit-column-count:3;`, `-moz-column-count:3;`, `column-count:3;` (in the body block) are grouped under a callout labeled "定义页面内容显示为3列" (Define page content display as 3 columns).
- Gap:** `-webkit-column-gap:3em;`, `-moz-column-gap:3em;`, `column-gap:3em;` (in the body block) are grouped under a callout labeled "定义列间距为3em, 默认为1em" (Define column gap as 3em, default is 1em).
- Rule:** `line-height:2.5em;`, `-webkit-column-rule:dashed 2px gray;`, `-moz-column-rule:dashed 2px gray;`, `column-rule:dashed 2px gray;` (in the body block) are grouped under a callout labeled "定义列边框为2像素宽的灰色虚线" (Define column border as 2px gray dashed line).
- Span:** `-webkit-column-span:all;`, `-moz-column-span:all;`, `column-span:all;` (in the h1 and h2 blocks) are grouped under two separate callouts: one for h1 labeled "设置一级标题跨越所有列显示" (Set first-level title to span all columns) and one for h2 labeled "设置二级标题跨越所有列显示" (Set second-level title to span all columns).

```
<body>
<h1>故都的秋（选段）</h1>
<h2>郁达夫</h2>
<p>...</p>
</body>
</html>
```

省略的文档内容

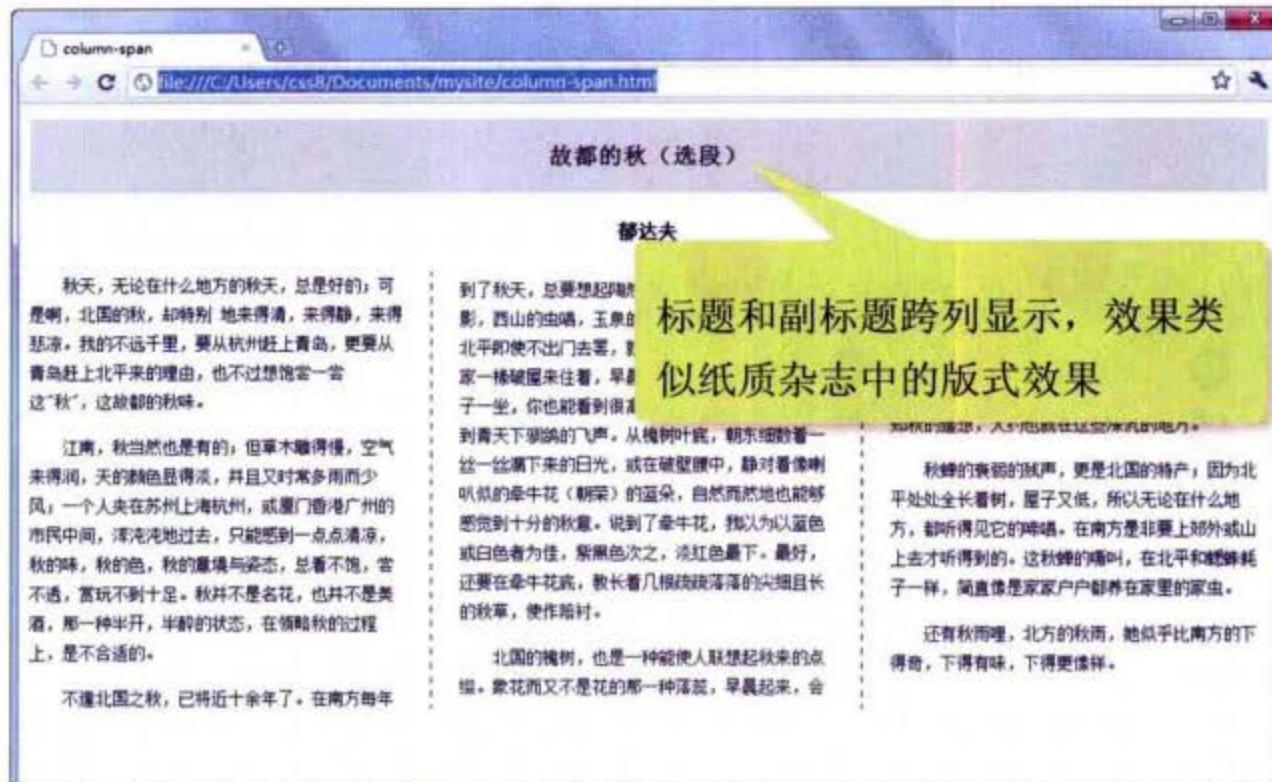


图6.6 设计标题跨列显示效果

6.7 定义栏目高度——column-fill 属性

column-fill 属性可以定义栏目的高度是否统一。该属性的基本语法如表 6.7 所示。

表6.7 column-fill属性的基本语法

语法项目	说明
值	auto balance
初始值	balance
适用于	多列布局元素
可否继承	否
百分比	N/A
媒介	视觉

取值简单说明：

- auto：各列的高度随其内容的变化而自动变化。
- balance：各列的高度将会根据内容最多的那一列的高度进行统一。

浏览器兼容性检测

目前，Webkit 引擎支持 -webkit-column-fill 私有属性，Mozilla Gecko 引擎、Presto 引擎和 IE 浏览器暂时不支持 column-fill 属性，也没有定义支持 column-fill 属性的私有属性。

借助兼容方式，各主流浏览器对 column-span 属性的支持情况如下。注意，从实战角度分析，Webkit 引擎对 column-fill 属性的支持还存在较大分歧，不同浏览器的解析效果会存在很大不同。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 3.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 4.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 8.0.x

实战体验：设计不等高的多列布局效果

在下面这个示例中，我们将演示 column-fill 属性的用法，当然 Safari 和 Chrome 浏览器的解析效果还是存在很大差异的，演示效果如图 6.7 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css" media="screen">
body {
    -webkit-column-count:3;
    -moz-column-count:3;
    column-count:3;
    -webkit-column-gap:3em;
    -moz-column-gap:3em;
    column-gap:3em;
    line-height:2.5em;
    -webkit-column-rule:dashed 2px gray;
    -moz-column-rule:dashed 2px gray;
    column-rule:dashed 2px gray;
    -webkit-column-fill:auto;
    -moz-column-fill:auto;
    column-fill:auto;
}
.c1 {
    width:100%;
    height:500px;
    background:red;
}
.c2 {
}

```

```

width:100%;
height:300px;
background:green;
}
.c3 {
width:100%;
height:100px;
background:blue;
}
</style>
<title>column-fill</title>
</head>
<body>
<div class="c1"></div>
<div class="c2"></div>
<div class="c3"></div>
</body>
</html>

```

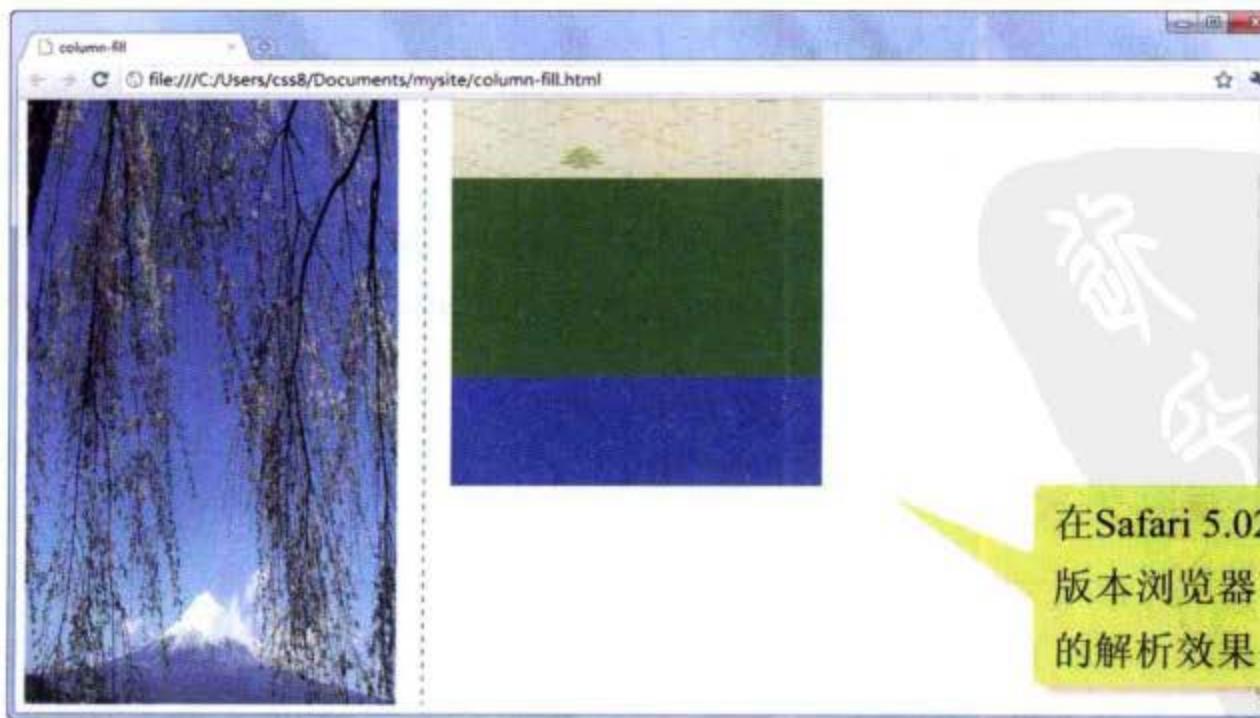
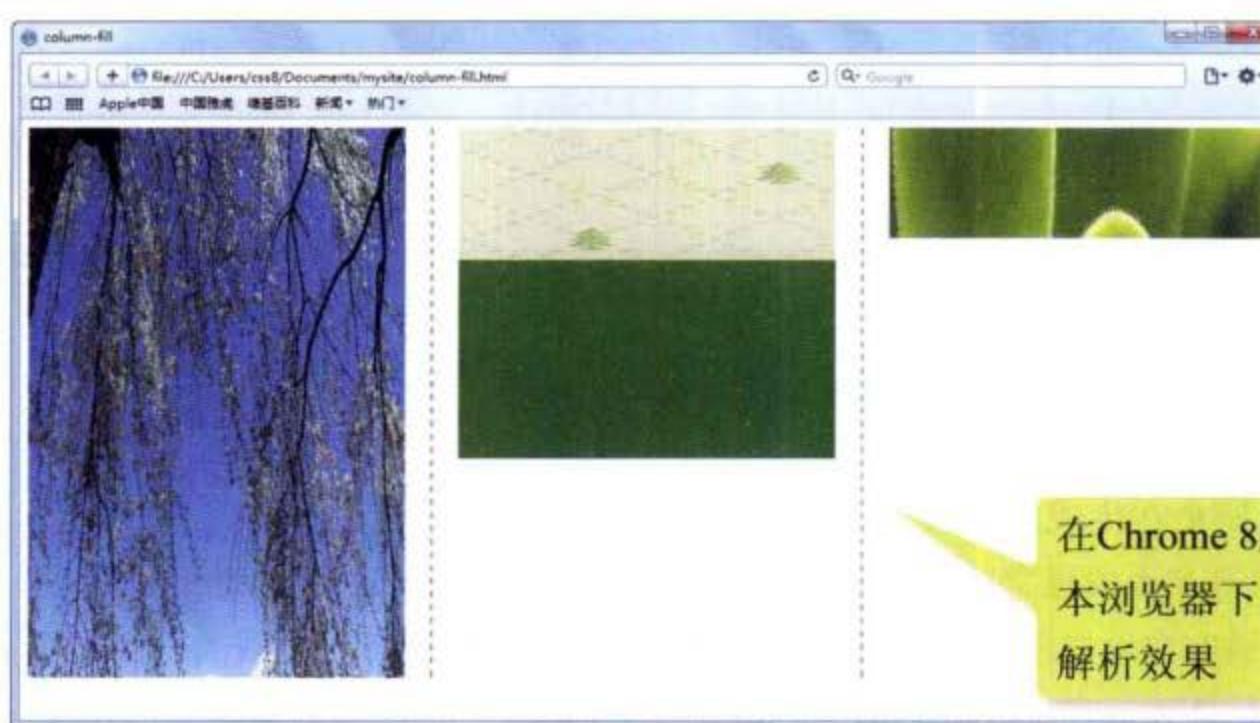


图6.7 设计列自适应高度的效果比较

6.8 分列打印

`break-before`、`break-after` 和 `break-inside` 是新增的用来控制页面分列打印的三个属性，它们的功能与 CSS 2.1 规范中的 `page-break-before`、`page-break-after` 和 `page-break-inside` 三个属性相同，用法也相同。这些属性的基本语法如表 6.8 所示。

表6.8 分列打印属性的基本语法

语法项目	说明
值	<code>auto</code> <code>always</code> <code>avoid</code> <code>left</code> <code>right</code> <code>page</code> <code>column</code> <code>avoid-page</code> <code>avoid-column</code>
初始值	<code>auto</code>
适用于	块元素
可否继承	否
百分比	N/A
媒介	打印输出

取值简单说明：

- `auto`：不强迫也不禁止在生成框之前（之后、之间）分页。
- `always`：总是强迫在生成框之前（之后）分页。
- `avoid`：避免在生成框之前（之后、之间）分页。
- `left`：强迫在生成框之前（之后）分一个或两个页，使下一页成为一个左页。
- `right`：强迫在生成框之前（之后）分一个或两个页，使下一页成为一个右页。
- `page`：总是强迫在生成框之前（之后）分页。
- `column`：总是强迫在生成框之前（之后）分列。
- `avoid-page`：总是避免在生成框之前（之后）分页。
- `avoid-column`：总是避免在生成框之前（之后）分列。

可能的分页位置通常受到父元素的 `break-inside` 属性、前继元素的 `break-after` 属性或后续元素的 `break-before` 属性的影响。如果这些属性值不为 `auto`，则 `always`、`left` 和 `right` 属性值将优先于 `avoid`。

浏览器兼容性检测

目前，Webkit 引擎支持 `-webkit-break-before`、`-webkit-break-after` 和 `-webkit-break-inside` 私有属性，Mozilla Gecko 引擎、Presto 引擎和 IE 浏览器暂时不支持这些属性，也没有定义支持这些标准属性的私有属性。借助兼容方式，各主流浏览器对 `column-span` 属性的支持情况

如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✓ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 3.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✓ Chrome 4.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✓ Chrome 8.0.x

6.9

结合案例实战——设计精美的多列网页版式

多列布局适合纯文字版式设计，不适合做网页结构布局。灵活使用多列布局特性，可以实现在多列中显示文字和图片，从而节省大量的网页空间。如果网页上的文字很长，多列布局特性就能够发挥它的用武之地。在下面这个案例中，我们把 CSS 禅意花园官方网站主页翻译过来，并重新进行排版设计。

CSS Zen Garden (<http://www.csszengarden.com/>) 是 Dave Shea 于 2003 年创建的 CSS 标准推广小站，但就是这么一个小站却闻名全球，获得众多奖项。站长 Dave Shea 是一位图像设计师，致力于推广标准 Web 设计。该站被中国台湾设计师薛良斌和李士杰汉化为中文繁体版 (<http://www.csszengarden.com/tr/chinese/>) 之后，就有人把它称为 CSS 禅意花园，从此禅意花园就成了 CSS Zen Garden 网站的代名词。

CSS 禅意花园默认设计效果如图 6.8 所示。整个页面通过左上、右下对顶角定义背景图像，这些荷花、梅花以及汉字形体修饰配合右上顶角的宗教建筑，完全把人带入禅意的后花园之中。

也许很多网友浏览这样的页面时，看不出它有什么优点：内容简单、结构单纯、样式也很朴实。相信很多设计师第一眼看到它时，也不会认为这个页面有多好。但是，如果仔细分析它的结构和样式，一定会学到很多东西。

CSS 禅意花园整个页面的信息简洁，结构层次清晰。信息从上到下，按照网页标题、网页菜单、主体栏目信息、次要导航和页脚信息有顺序地排列在一起。从 SEO 设计的角度来考察，我们可以看到，Dave 把所有导航菜单等功能信息全部放在结构的后面，这也是经典之笔，很值得设计师认真研究和学习。下面我们借助多列布局特性来重新设计 CSS 禅意花园的页面效果，如图 6.9 所示。在设计中，主要利用了多列布局、多重背景和半透明效果设计等技术手段。



图6.8 CSS禅意花园默认设计效果



图6.9 使用多列布局设计的CSS禅意花园页面效果

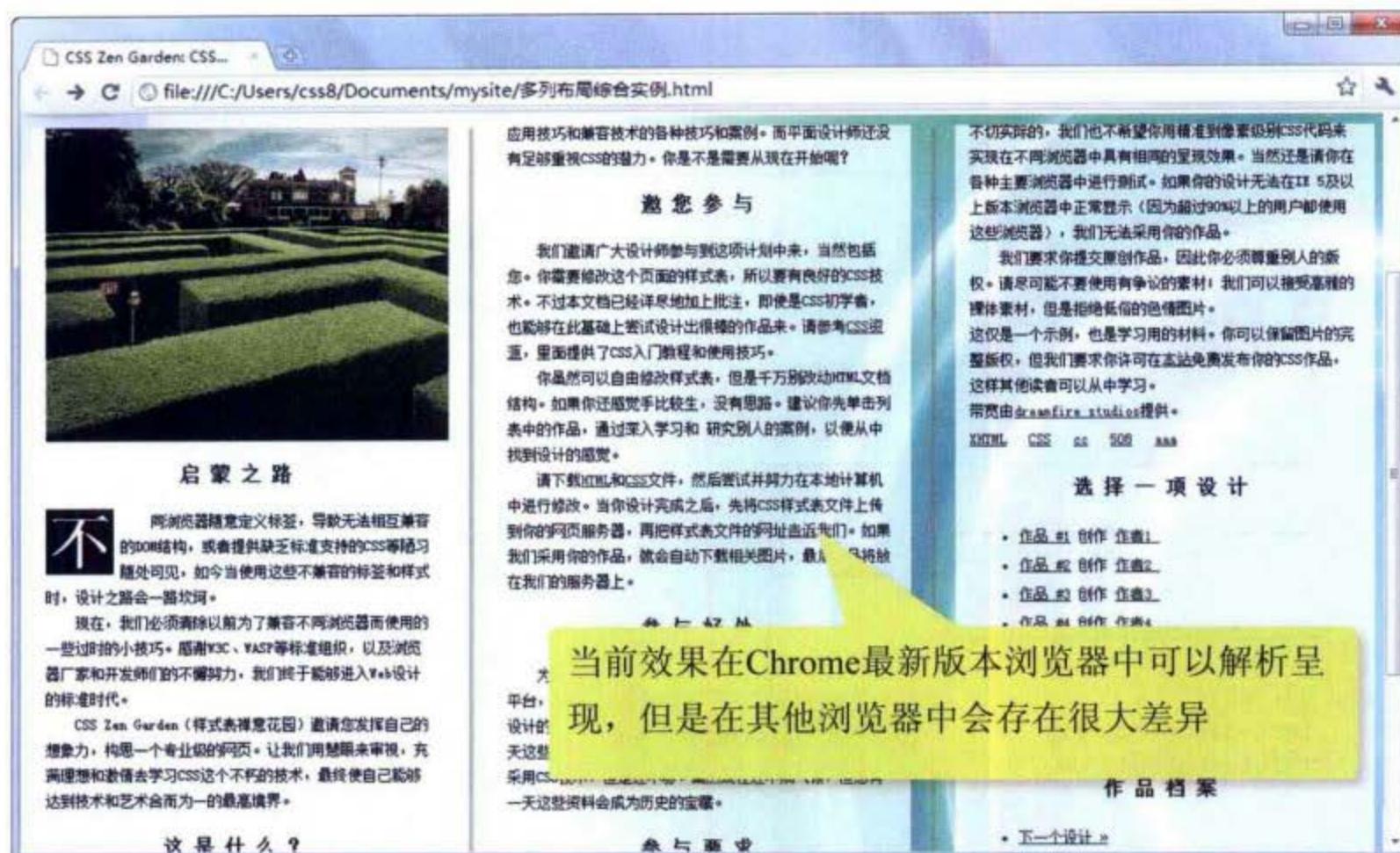


图6.9 (续)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS Zen Garden: CSS设计之美</title>
<meta content="dave shea" name="author">
<meta content="标准设计, CSS, 样式表, XHTML, 平面设计, W3C, 网页标准, 可视化, 显示" name="keywords">
<meta content="展示以CSS为基础的设计技术, 以实现最佳视觉效果。只要选择列表中的任一个样式表, 就可以将它加载到这个页面中。" name="description">
<meta content="all" name="robots">
<!-- 修正会一闪而过的无样式内容。请参考 http://www.bluerobot.com/web/css/fouc.asp -->
<script type="text/javascript"></script>
<!--
```

这是一份XHTML结构文档，用来提供给设计师进行最大弹性设计的基础。
里面有不少没有用到的类和标签。在实际应用的时候，
你可以有选择地使用或设计。

不过我想你可能同意，如果使用表格来设计相同的页面效果，可以会有更好的效果。但这不是我们的目标。

```
-->
<STYLE type="text/css media=all">
body {
```

```
background:url(images/page1.gif) no-repeat right 20px,
    url(images/bg.jpg) no-repeat right bottom,
    url(images/page3.jpg) no-repeat left top;
background-size:auto, 74% 79.5%, auto;
```

设计多重网页背景，并设置其显示大小

```

color:#000;
font-size:12px;
font-family:"新宋体", Arial, Helvetica, sans-serif;
-webkit-column-count:3;
-moz-column-count:3;
column-count:3;
-webkit-column-gap:3em;
-moz-column-gap:3em;
column-gap:3em;
line-height:2em;
-webkit-column-rule:double 3px gray;
-moz-column-rule:double 3px gray;
column-rule:double 3px gray;
}

.allcols {
    -webkit-column-span:all;
    -moz-column-span:all;
    column-span:all;
}

h1, h2, h3 {
    text-align:center;
    margin-bottom:1em;
}

h2 {
    color:#666;
    text-decoration:underline;
}

h3 {
    letter-spacing:0.4em;
    font-size:1.4em;
}

p {
    margin:0;
    line-height:1.8em;
}

#quickSummary .p2 { text-align:right; }
#quickSummary .p1 { color:#444; }
.p1, .p2, .p3 { text-indent:2em; }
#quickSummary { margin:4em; }

a { color:#222; }

a:hover {
    color:#000;
    text-decoration:underline;
}

.first:first-letter {
    font-size:50px;
    float:left;
    margin-right:6px;
    padding:2px;
    font-weight:bold;
    line-height:1em;
    background:#000;
    color:#fff;
    text-indent:0;
}

```

定义页面内容显示为3列

**定义列间距为3em，
默认为1em**

**定义列边框为3像素
宽的灰色虚线**

设计跨列显示类

**设计报刊杂志的首字
下沉显示类**

```

#preamble img {
    height:260px;
    -webkit-column-span:all;
    -moz-column-span:all;
    column-span:all;
}

#container {
    background:rgba(255, 255, 255, 0.8);
    padding:0 1em;
}

</STYLE>
</head>
<body id="css-zen-garden">
<div id="container">
    <div id="intro">
        <div id="pageHeader" class="allcols">
            <h1><span>CSS禅意花园</span></h1>
            <h2><span><acronym title="cascading style sheets">CSS</acronym>设计之美</span></h2>
        </div>
        <div id="quickSummary" class="allcols">
            <p class="p1"><span>展示以<acronym
                title="cascading style sheets">CSS</acronym>技术为基础，并提供超强的视觉冲击力。只要选择列表中任意一个样式表，就可以将它加载到本页面中，并呈现不同的设计效果。</span></p>
            <p class="p2"><span>下载<a title="这个页面的HTML源代码不能够被改动。"
                href="http://www.csszengarden.com/zengarden-sample.html">HTML文档</a> 和 <a
                title="这个页面的CSS样式表文件，你可以更改它。"
                href="http://www.csszengarden.com/zengarden-sample.css">CSS文件</a>。</span></p>
        </div>
        <div id="preamble">
            <h3><span>启蒙之路</span></h3>
            <p class="p1 first"><span>不同浏览器随意定义标签，导致无法相互兼容的<acronym
                title="document object model">DOM</acronym>结构，或者提供缺乏标准支持的<acronym
                title="cascading style sheets">CSS</acronym>等陋习随处可见，如今当使用这些不兼容的标签和样式时，设计
                之路会一路坎坷。</span></p>
            <p class="p2"><span>现在，我们必须清除以前为了兼容不同浏览器而使用的一些过时的小技巧。感谢
                <acronym
                title="world wide web consortium">W3C</acronym>、<acronym
                title="web standards project">WASP</acronym>等标准组织，以及浏览器厂家和开发师们的不懈努力，我们终于
                能够进入Web设计的标准时代。</span></p>
            <p class="p3"><span>CSS Zen
                Garden（样式表禅意花园）邀请您发挥自己的想象力，构思一个专业级的网页。让我们用慧眼来审视，充满理想和激情去学习CSS这个不朽的技术，最终使自己能够达到技术和艺术合而为一的最高境界。</span></p>
        </div>
    </div>
    <div id="supportingText">
        <div id="explanation">
            <h3><span>这是什么？</span></h3>
            <p class="p1"><span>对于网页设计师来说应该好好研究<acronym
                title="cascading style sheets">CSS</acronym>。Zen Garden致力于推广和使用CSS技术，努力激发和鼓励您的
                灵感和参与。你可以从浏览高手的设计作品入门。只要选择列表中的任一个样式表，就可以将它加载到这个页面中。<acronym
                title="hypertext markup language">HTML</acronym>文档结构始终不变，但是你可以自由的修改和定义
                <acronym
                title="cascading style sheets">CSS</acronym>样式表。</span></p>
            <p class="p2"><span><acronym
                title="cascading style sheets">CSS</acronym>具有强大的功能，可以自由控制HTML结构。当然你需要拥有驾驭

```

设计插图跨列显示，但实
际浏览无效果

设计栏目框半透明显示，
从而设计网页背景半透明
显示效果

CSS技术的能力和创意的灵感，同时亲自动手，用具体的实例展示CSS的魅力，展示个人的才华。截至目前，很多Web设计师和程序员已经介绍过许多关于CSS应用技巧和兼容技术的各种技巧和案例。而平面设计师还没有足够重视CSS的潜力。你是不是需要从现在开始呢？

```

    </span></p>
    </div>
    <div id="participation">
        <h3><span>邀您参与</span></h3>
        <p class="p1"><span>我们邀请广大设计师参与到这项计划中来，当然包括您。你需要修改这个页面的样
式表，所以要有良好的<acronym
            title="cascading style sheets">CSS</acronym>技术。不过本文档已经详尽地加上标注，即使是<acronym
            title="cascading style sheets">CSS</acronym>初学者，也能够在此基础上尝试设计出很棒的作品来。请参考<a
title=CSS相关资源列表
            href="http://www.mezzoblue.com/zengarden/resources/"><acronym
            title="cascading style sheets">CSS</acronym>资源</a>，里面提供了CSS入门教程和使用技巧。</span></p>
            <p class="p2"><span>你虽然可以自由修改样式表，但是千万别改动<acronym
            title="hypertext markup language">HTML</acronym>文档结构。如果你还感觉手比较生，没有思路，建议你先单
击列表中的作品，通过深入学习和研究别人的案例，以便从中找到设计的感觉。</span></p>
            <p class="p3"><span>请下载<a title=="这个页面的HTML结构请不要改动。"
            href="http://www.csszengarden.com/zengarden-sample.html">HTML</a>和<a
            title=这个页面的CSS范例文件，你可以更改它。
            href="http://www.csszengarden.com/zengarden-sample.css">CSS</a>文件，然后尝试并努力在本地计算机中
进行修改。当你设计完成之后，先将<acronym
            title="cascading style sheets">CSS</acronym>样式表文件上传到你的网页服务器，再把样式表文件的网址<a
            title=请用联络窗体告诉我们你的CSS档案
            href="http://www.mezzoblue.com/zengarden/submit/">告诉</a>我们。如果我们采用你的作品，就会自动下载
相关图片，最后成品将放在我们的服务器上。</span></p>
        </div>
        <div id="benefits">
            <h3><span>参与好处</span></h3>
            <p class="p1"><span>为什么要邀请您参与这项计划呢？因为这里提供了一个平台，能够展现你的技术水平、激发你的设计灵感，并且所设计的<acronym
            title="cascading style sheets">CSS</acronym>案例文档，将成为重要的学习资料。即使到了今天这些资料都是
很珍贵的和重要的。越来越多主流网站开始采用CSS技术，但是还不够。虽然现在还不成气候，但总有一天这些资料会成为历史的宝
藏。</span></p>
            </div>
            <div id="requirements">
                <h3><span>参与要求</span></h3>
                <p class="p1"><span><acronym
                    title="cascading style sheets, version 1">CSS1</acronym>已经落伍了。<acronym
                    title="cascading style sheets, version 2">CSS2</acronym>现在成为网页设计的主要技术标准。CSS Zen
                    Garden采用的是可行、实用的<acronym title="cascading style sheets">CSS</acronym>
语法，而不是那些仅能被个别超前浏览器所能解析的技术（如<acronym title="cascading style sheets, version
3">CSS3</acronym>）。实际上，我们唯一的要求就是你的<acronym
            title="cascading style sheets">CSS</acronym>要符合标准。</span></p>
                <p class="p2"><span>即便如此，你还需要考虑<acronym
                    title="cascading style sheets">CSS</acronym>的兼容性。因为即使是完全符合标准的<acronym
                    title="cascading style sheets">CSS</acronym>语法，不同浏览器所呈现的效果也是不一致的。很多设计师整天
忙于CSS兼容性处理，这实在是件让人恼火的事情。也请你参考相关<a
            title=CSS相关资源列表
            href="http://www.mezzoblue.com/zengarden/resources/">资源</a>了解一些Bug修正的信息。希望所有浏览器
都能够完整、正确、统一的支持<acronym
            title="cascading style sheets">CSS</acronym>是不切实际的，我们也不希望你用精准到像素级别CSS代码来实现
在不同浏览器中具有相同的呈现效果。当然还是请你在各种主要浏览器中进行测试。如果你的设计无法在IE 5及以上版本浏览器中
正常显示（因为超过90%以上的用户都使用这些浏览器），我们无法采用你的作品。</span></p>
            <p
            class="p3"><span>我们要求你提交原创作品，因此你必须尊重别人的版权。请尽可能不要使用有争议的素材；我们可以

```

接受高雅的裸体素材，但是拒绝低俗的色情图片。

<p class="p4">这仅是一个示例，也是学习用的材料。你可以保留图片的完整版权，但我们要求你许可在本站免费发布你的<acronym title="cascading style sheets">CSS</acronym>作品，这样其他设计师可以从中学习。</p>

<p class="p5">带宽由dreamfire studios提供。</p>

</div>

<div id="footer">XHTML &nbsp CSS &nbsp cc &nbsp 508 &nbsp aaa </div>

</div>

<div id="linkList">

<!-- 这是最具弹性的设计部分 - 需要你花费很大的心思来设计这个列表。 -->

<div id="linkList2">

<!-- 你也许会因为链接而感到讶异，没有办法，这是为了配合 wcag 1 accessibility 使用的技巧。 -->

<!-- 我也不喜欢这样，不过这是个视觉效果的练习。 -->

<div id="lselect">

<h3 class="select">选择一项设计</h3>

<!-- 这是链接列表的起点，每页不会超过 8 个链接。 -->

作品 #1 创作 作者1

作品 #2 创作 作者2

作品 #3 创作 作者3

作品 #4 创作 作者4

作品 #5 创作 作者5

作品 #6 创作 作者6

作品 #7 创作 作者7

作品 #8 创作 作者8


```

</div>
<!--
如果你的页面还没有塞满，这里都不会显示，只是准备内容。
如果你为「选择一项设计」的 h3 标签订制了不少样式，这里也必须订制好。
<div id="lfavorites">
<h3 class="favorites"><span>最爱：</span></h3>
<ul>
<li><a href="#">filename</a> by <a href="#" class="c">submitter</a>&nbsp;</li>
<li><a href="#">filename</a> by <a href="#" class="c">submitter</a>&nbsp;</li>
</ul>
</div>
-->
<div id="larchives">
<h3 class="archives"><span>作品档案</span></h3>
<ul>
<li><a title="浏览下一个设计。快捷键: n" accesskey="n"
href="http://www.csszengarden.com/tr/chinese/?CSSfile=/001/001-zh.css&page=1">下一个设计
&raquo;</a>
<li><a title="浏览Zen Garden上的所有设计。快捷键: w" accesskey="w"
href="http://www.mezzoblue.com/zengarden/alldesigns/">浏览所有设计</a> </li>
</ul>
</div>
<div id="lresources">
<h3 class="resources"><span>参考资源</span></h3>
<ul>
<li><a title="检视这一个设计的CSS档案。快捷键: v" accesskey="v"
href="../001-zh.css">查看这个设计的样式表<acronym
title="cascading style sheets">CSS</acronym></a>
<li><a title="链接到提供CSS相关信息的好网站。快捷键: r" accesskey="r"
href="http://www.mezzoblue.com/zengarden/resources/"><acronym
title="cascading style sheets">CSS</acronym>参考资料</a>
<li><a title="关于Zen Garden的常见问答案集。快捷键: q" accesskey="q"
href="http://www.mezzoblue.com/zengarden/faq/">常见问题</a>
<li><a title="提交你的CSS样式表。快捷键: s" accesskey="s"
href="http://www.mezzoblue.com/zengarden/submit/">投稿</a>
<li><a title="浏览这个页面的翻译版本。快捷键: t" accesskey="t"
href="http://www.mezzoblue.com/zengarden/translations/">翻译文件</a> </li>
</ul>
</div>
</div>
</div>
<!-- 这些额外的标签，你可以发挥想象力进行其他的设计。 -->
<!-- 你可以为它们加上背景图片，利用width和height控制它们的大小，最后使用绝对定位方式在页面进行放置。 -->
<!-- 这种做法会在mozilla v1.0 到 v1.3 造成讨厌的透明gif色彩偏移问题，所以请务必使用这些浏览器测试你的作品。 -->
<div id="extraDiv1"><span></span></div>
<div id="extraDiv2"><span></span></div>
<div id="extraDiv3"><span></span></div>
<div id="extraDiv4"><span></span></div>
<div id="extraDiv5"><span></span></div>
<div id="extraDiv6"><span></span></div>
</body>
</HTML>

```

CSS 禅意花园通过一个经典的结构，倡议设计师以此为基础设计不同效果的作品。下面

我们再补充讲解一下它的结构，相信通过学习它的结构，能够帮助设计师学习更多的网页操作技巧和设计体验。

一般每个页面都会存在很多共同的信息模块，如标题（Logo）、广告（Banner）、导航（Menu）、功能（Serve）、内容（Content）和版权（Copyright）等信息。而不同类型的网站有不同的页面需求，对于各种公共信息模块的取舍，多少会略有不同，这时就应该具体情况具体分析。在设计网页基本结构时，设计师不妨根据信息需求的简单分析和信息的重要性来对页面各个模块进行适当排序，然后列出基本的结构。

```
<div id="c-xms">
  <div id="header"></div>
  <div id="nav"></div>
  <div id="content"></div>
  <div id="cnav"></div>
  <div id="footer"></div>
</div>
```

!-- 网页结构外套 -->
!-- 网页标题模块 -->
!-- 网页菜单模块 -->
!-- 网页信息模块 -->
!-- 次要导航模块 -->
!-- 版权信息模块 -->

构建基本结构应该注意几个问题。

- 在设计基本结构时，可以不考虑标签的语义性，统一使用 div 元素来构建。
- 应该为基本结构的每一个 div 元素进行命名（设置 ID 属性），以便后期控制。
- 可以考虑为整个页面结构设计一个外套（即定义一个结构根元素），以方便控制。

在设计结构时，不要考虑后期如何显示，也不要顾虑结构的顺序是不是会影响页面的显示效果，应该完全抛弃页面效果和 CSS 样式等概念对结构的影响。

有了基本的框架结构，可以继续深入，这时不妨去完善主体区域的结构（即网页内容模块），这部分是整个页面的核心，也是设计工作的重点。应该说，在编辑网页结构的全部过程中，我们都应该去考虑页面显示效果问题，而是应该静下心来单纯考虑结构。但是在实际操作中，会不可避免地联想到页面的显示问题，例如，分几行几列显示（这里的行和列涉及网页基本结构的走向问题）。不同的行列结构肯定都有适合自己的显示情况，所以当设计师进入这一步时，适当考虑页面显示问题也无可厚非，但是不要考虑过多。

恰当的嵌套结构需要结合具体的信息介绍，这里先暂不详细分析。抽象地说，模块的结构关系可以分为三种基本模型。

□ 平行结构。

```
<div id="A"></div>
<div id="B"></div>
<div id="C"></div>
```

□ 包含结构。

```
<div id="A">
  <div id="B"></div>
  <div id="C"></div>
</div>
```

□ 嵌套结构。

```
<div id="A"></div>
<div>
    <div id="B"></div>
    <div id="C"></div>
</div>
```

具体采用哪种结构都不重要，可以根据信息的结构关系来选择。如果 `<div id="latest">` 和 `<div id="m2">` 两个信息模块内容比较接近，而 `<div id="subcol">` 模块与它们在内容上相差很远，不妨采用嵌套结构。如果这些栏目的信息类型雷同，使用并列式会更经济。

禅意花园的整个文章包含三部分：站点介绍、支持文本和链接列表。

- 站点介绍。站点介绍部分犹如抒情散文，召唤你赶紧加入 CSS 标准设计中来。该部分包含三段，即网页标题信息（包括主副标题）、内容简介（呼唤网友赶紧加入进来）、启蒙之路（回忆和总结当前标准之路的艰巨性和紧迫性）。
- 支持文本。支持文本部分犹如叙事散文，娓娓道来，详细介绍活动的内容，用户参与的条件、支持、好处等。该部分包含四段，即这是什么、邀您参与、参与好处、参与要求。另外末尾还包含了相关验证信息。
- 链接信息。第三部分很简洁地列出了所有超链接信息。该部分也包含三小块链接信息。

当然，本综合案例所实现的多列布局效果，由于 CSS 3 的多列布局特性并未得到各大主流浏览器的支持，同时支持该特性的浏览器的解析效果也存在差异，所以当你在不同的浏览器中预览时，看到的效果也会不同。如图 6.10 所示是在 Firefox 浏览器下的浏览效果。

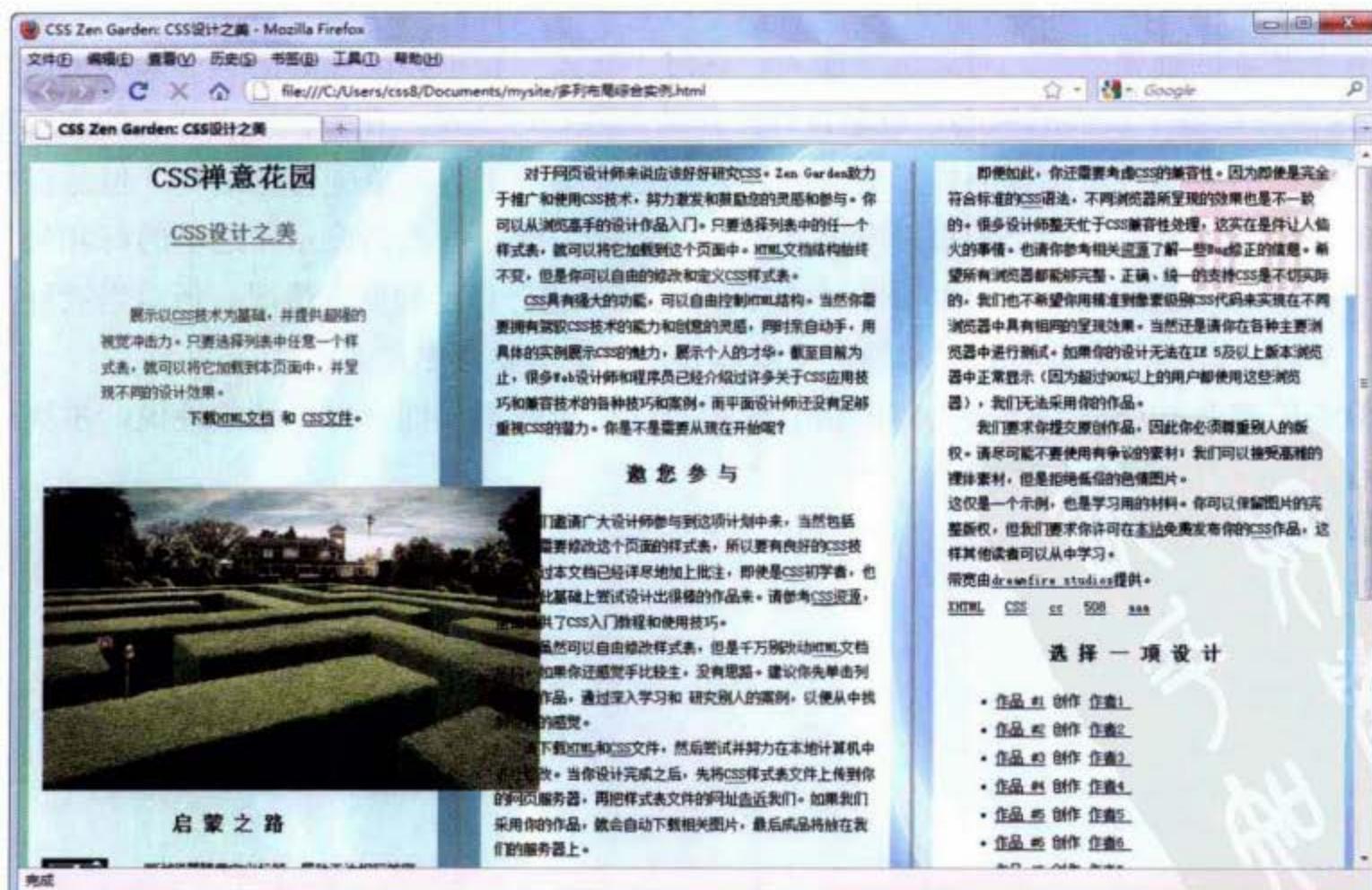


图 6.10 在 Firefox 3.6 版本浏览器下的浏览效果

CSS 3 漐变设计

渐变背景是网页设计中不可或缺的审美元素。一直以来，网页设计师必须依赖现成的图片来实现渐变背景效果。这是一种笨拙的方法。为了显示渐变效果而专门制作一个图片的做法是不灵活的，这样制作出来的效果很快会成为视觉优化、网页升级的障碍。但遗憾的是，网页设计师暂时还必须这样做，以确保网页效果能够安全地呈现给所有浏览者。

感谢 Firefox、Safari 和 Chrome 浏览器的大胆尝试，现在设计师可以用少量的努力实现强大的渐变效果。W3C 组织也将渐变设计收入 CSS 3 标准中（详细说明请参阅 <http://dev.w3.org/csswg/css3-images/>）。

基于 CSS 的渐变与图片渐变相比，最大的优点是便于修改，同时支持无级缩放，过渡更加自然。目前，实现 CSS 渐变的只有基于 Webkit 和 Gecko 引擎的浏览器，基于 Presto 引擎的 Opera 浏览器暂时不支持渐变，基于 Trident 的 IE 虽然可以通过滤镜的方式实现，但并不提倡。本章将详细讲解 CSS 渐变的简单实现以及在不同浏览器中的实现，并通过大量精美的案例帮助读者应用渐变设计。

7.1 Webkit 引擎的 CSS 渐变实现方法

目前，CSS 渐变设计还没有统一的标准，用法差异很大。不同渲染引擎实现渐变的语法不同，均为私有属性，这给用户应用带来诸多不便。虽然 W3C 最后提出标准的渐变设计方法，但是还没有获得各主流浏览器的认可。下面我们将分别针对不同的实现方法进行讲解。

Webkit 和 Gecko 引擎对于 CSS 3 属性一般都采取同样的语法，但是对于渐变，目前还无法达成一致，最新推出的 W3C 标准语法也是千差万别，所以我们也只能对它们的实现分别进行讲解。

7.1.1 基本语法

Webkit 是第一个支持渐变的浏览器引擎 (Safari 4 及其以上版本支持), 读者可以访问 <http://webkit.org/blog/175/introducing-css-gradients/> 网页参考更详细的信息。

Webkit 引擎支持的渐变方法如下：

```
-webkit-gradient(<type>, <point> [, <radius>]? , <point> [, <radius>]? [, <stop>]*)
```

该函数的参数说明如下：

- <type> : 定义渐变类型, 包括线性渐变 (linear) 和径向渐变 (radial)。
- <point> : 定义渐变起始点和结束点坐标, 即开始应用渐变的 x 轴和 y 轴坐标, 以及结束渐变的坐标。该参数支持数值、百分比和关键字, 如 (0,0) 或者 (left,top) 等。关键字包括 top、bottom、left 和 right。
- <radius> : 当定义径向渐变时, 用来设置径向渐变的长度, 该参数为一个数值。
- <stop> : 定义渐变色和步长。包括三个类型值, 即开始的颜色, 使用 from(color value) 函数定义; 结束的颜色, 使用 to(color value) 函数定义; 颜色步长, 使用 color-stop(value, color value) 定义。color-stop() 函数包含两个参数值, 第一个参数值为一个数值或者百分比值, 取值范围为 0 ~ 1.0 (或者 0% ~ 100%), 第二个参数值表示任意颜色值。

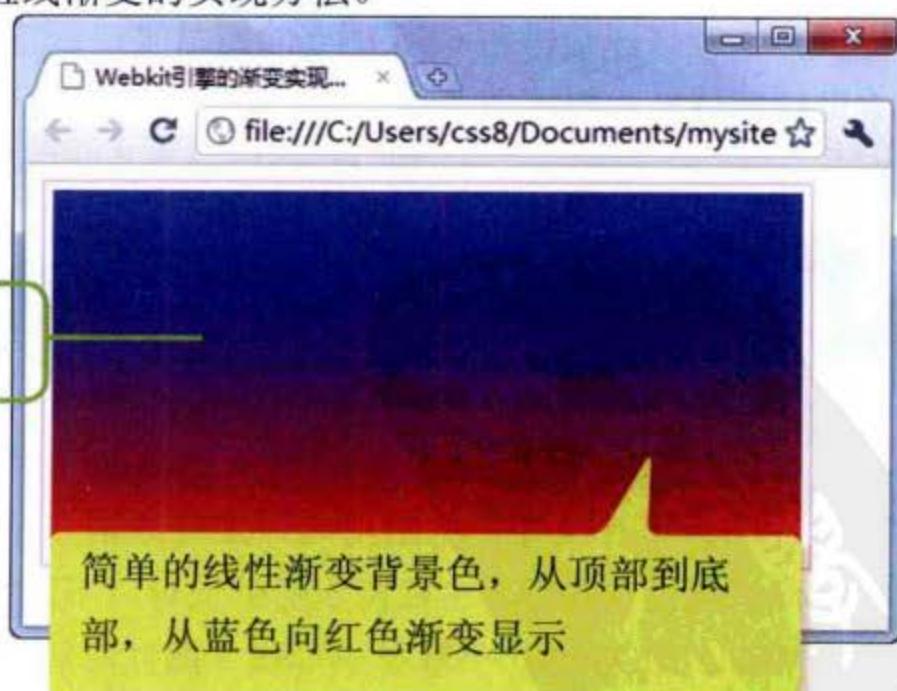
7.1.2 直线渐变的基本用法

下面结合示例分别演示 Webkit 引擎的直线渐变的实现方法。

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(linear,
        left top, left bottom,
        from(blue), to(red));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>

<div></div>
<style type="text/css">
div {
```

width:400px;
height:200px;
border:2px solid #FCF;
padding: 4px;



```

background: -webkit-gradient(linear,
    left top, left bottom,
    from(blue), to(red),
    color-stop(50%, green));
-webkit-background-origin: padding-box;
-webkit-background-clip: content-box;
}
</style>

```

```
<div></div>
```

```
<style type="text/css">
div {
```

```
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
```

```
    background: -webkit-gradient(linear,
        left top, left bottom,
        from(blue), to(red),
        color-stop(0.5, #fff),
        color-stop(0.5, #000))
```

```

-webkit-background-origin: padding-box;
-webkit-background-clip: content-box;
}

```

```
</style>
```

```
<div></div>
```

```
<style type="text/css">
```

```
div {
```

```
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
```

```
    background: -webkit-gradient(linear,
        left top, left bottom,
        from(blue), to(red),
        color-stop(0.4, #fff),
        color-stop(0.6, #000));
```

```

-webkit-background-origin: padding-box;
-webkit-background-clip: content-box;
}

```

```
</style>
```

```
<div></div>
```

更进一步的线性渐变背景色，从顶部到中间，再从中间到底部，从蓝色到绿色，再到红色渐变显示



设计二重渐变，从顶部到底部，先是从蓝色到白色渐变显示，再从黑色到红色渐变显示



通过设置不同的步长值，从而设计多重渐变效果，从顶部到底部，先是从蓝色到白色渐变，再从白色到黑色渐变，最后是从黑色到红色渐变显示

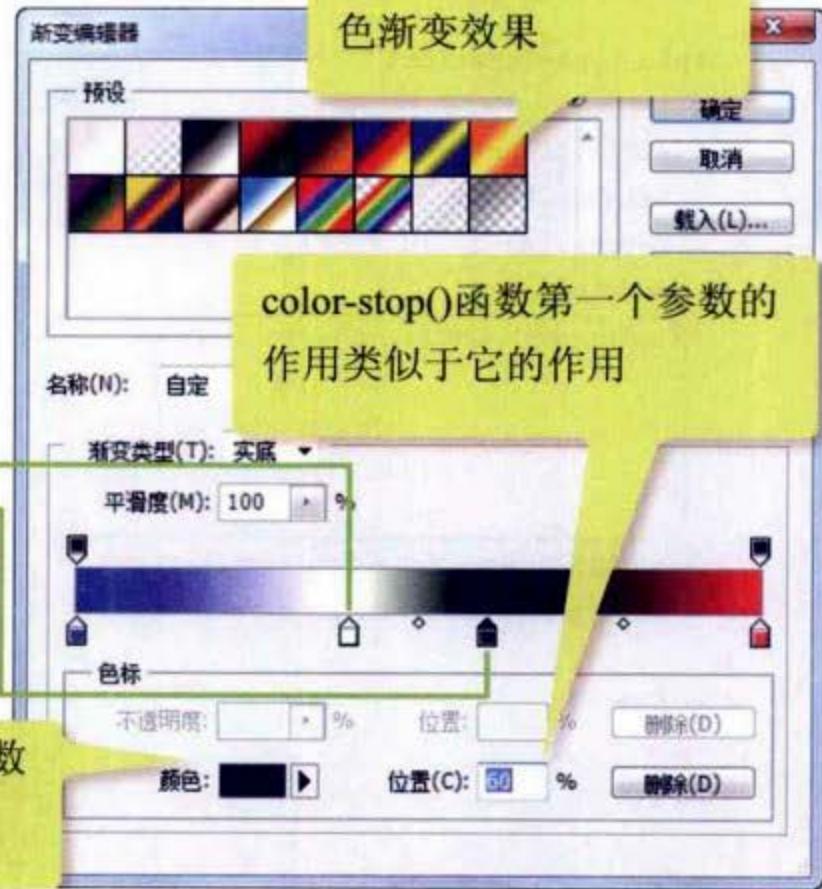
简单小结一下：color-stop()函数包含两个参数值，第一个参数值指定色标位置，第二个参数指定色标颜色。一个渐变可以包含多个色标，位置值为0~1之间的小数，或者0~100%之间的百分数，指定色标的位置比例，其用法与Photoshop中的直线渐变工具用法相似，如下图所示。



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(linear,
        left top, left bottom,
        from(blue), to(red),
        color-stop(0.4, #fff),
        color-stop(0.6, #000));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>

<div></div>
```

color-stop()函数第二个参数的作用类似于它的作用

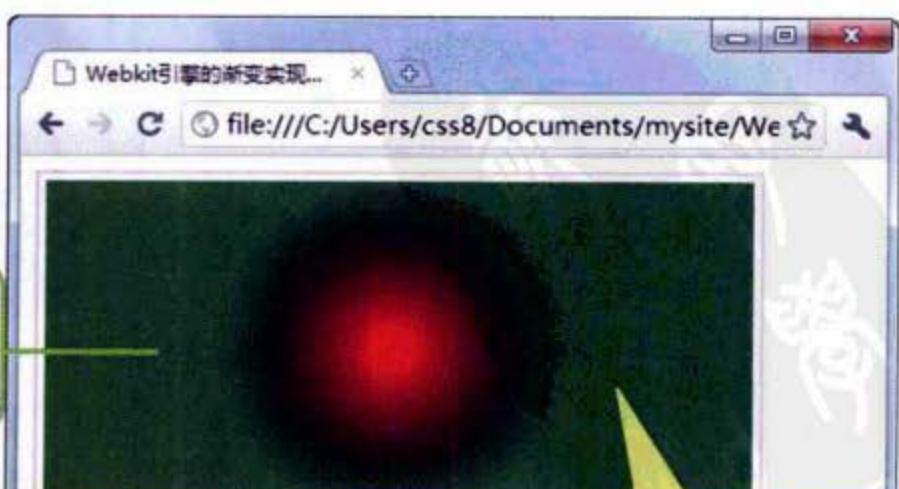


在Photoshop中编辑多色渐变效果

7.1.3 径向渐变的基本用法

相对于直线渐变，径向渐变的用法稍显复杂些。为了方便理解，下面结合示例分别演示Webkit引擎的径向渐变实现方法。

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 10,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>
```



同心圆（圆心坐标为200,100），内圆半径为10，外圆半径为100，内圆小于外圆半径，从内圆红色到外圆绿色径向渐变，超出外圆半径显示为绿色，内圆显示红色

```
<div></div>
```

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 100,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: padding-box;
}
</style>

<div></div>
```

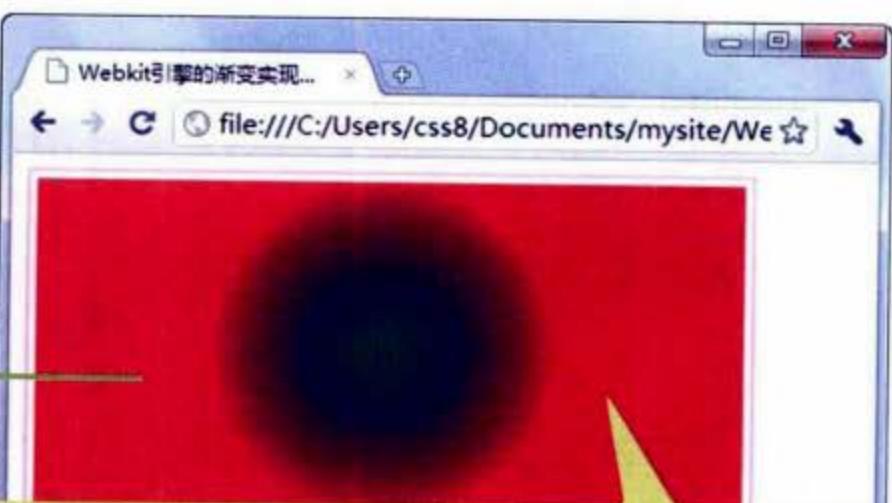
同心圆（圆心坐标为200,100），内圆半径为100，外圆半径为100，从内圆红色到外圆绿色径向渐变。当内圆和外圆半径相等时，则渐变无效



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 100,
        200 100, 10,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>

<div></div>
```

同心圆（圆心坐标为200,100），内圆半径为100，外圆半径为10，内圆大于外圆半径，从内圆红色到外圆绿色径向渐变，超出内圆半径显示为红色，外圆显示绿色



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        110 100, 10,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>
```

非同心圆，内圆圆心和外圆圆心距离等于两圆半径的差，则显示上图效果，不呈现径向渐变效果



```
</style>

<div></div>

<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        120 100, 10,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>
```

<div></div>

<style type="text/css">

```
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        100 100, 10,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>
```

<div></div>

<style type="text/css">

```
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 10,
        200 100, 100,
        from(red), to(green)),
        color-stop(90%, blue);
}
```



非同心圆，内圆圆心和外圆圆心的距离小于两圆半径的差，则显示上图效果，呈现锥形径向渐变效果。锥形的尖锐性与两圆圆心距离成正比



非同心圆，内圆圆心和外圆圆心的距离大于两圆半径的差，则显示上图效果，呈现锥形径向渐变效果，且部分区域被红色填充。锥形的尖锐性与两圆圆心距离成正比



同心圆，在内圆到外圆中间90%的位置，即距离外环内添加一个蓝色色标，设计多层次径向渐变，如上图所示。当然，还可以添加更多的色标，以设计多层次径向渐变效果

```
</style>

<div></div>
```

简单小结一下：径向渐变的起点坐标和结束坐标分别定义内圆和外圆坐标。在定义坐标的同时，还应该指定内圆和外圆的半径，当然也可以为径向渐变设置多个步长，用来添加多色径向渐变效果，其用法与Photoshop中的径向渐变工具用法相似，如下图所示。



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 10,
        200 100, 100,
        from(red), to(green));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>

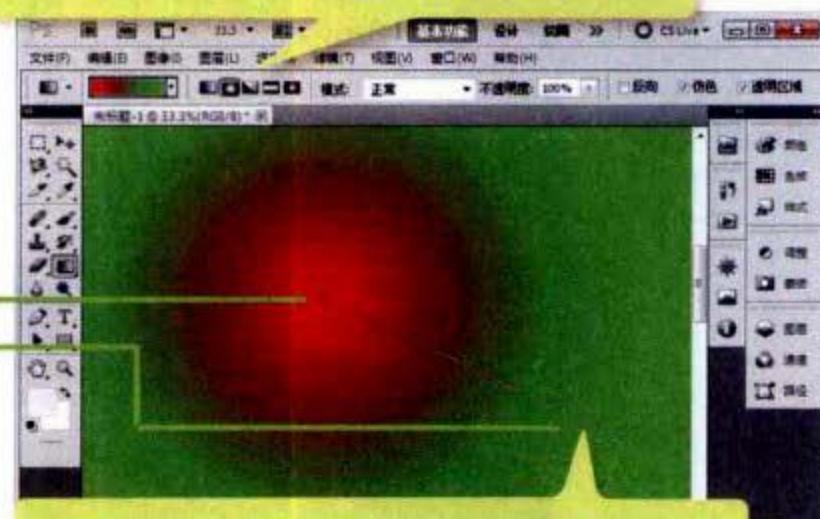
<div></div>
```

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -webkit-gradient(radial,
        200 100, 10,
        200 100, 90,
        from(red),
        to(rgba(1,159,98,0)));
    -webkit-background-origin: padding-box;
    -webkit-background-clip: content-box;
}
</style>

<div></div>
```

```
<style type="text/css">
```

在Photoshop中编辑径向渐变效果



在Photoshop中应用渐变效果，效果与同心圆且内圆半径小于外圆半径的CSS径向渐变效果相同



通过设置to()函数的颜色值为透明，可以设计发散的圆形效果

```


</style>



<div></div>



A screenshot of a web browser window titled "WebKit引擎的渐变实现...". The page displays a single, perfectly spherical object with a vibrant blue gradient. The text on the right side of the image provides a tip for creating similar effects.



通过设置to()函数的颜色值为透明，同时设计相似色，可以设计球形效果



A screenshot of a web browser window titled "WebKit引擎的渐变实现...". The page displays several overlapping bubbles of different colors (green, blue, red) with distinct radial gradients. A callout bubble on the right side provides a tip for creating such effects.



通过为背景图定义  
多个径向渐变，可  
以设计多个气泡效  
果，如图所示，代  
码如下所示



```

<style type="text/css">
div {
 width:400px;
 height:200px;
 border:2px solid #FCF;
 padding: 4px;
 background:
 -webkit-gradient(radial, 45 45, 10, 52 50, 30, from(#A7D30C), to(rgb(1,159,98,0)),
color-stop(90%, #019F62)),
 -webkit-gradient(radial, 105 105, 20, 112 120, 50, from(#ff5f98), to(rgb(255,1,136,0)),
color-stop(75%, #ff0188)),
 -webkit-gradient(radial, 95 15, 15, 102 20, 40, from(#00c9ff), to(rgb(0,201,255,0)),
color-stop(80%, #00b5e2)),
 -webkit-gradient(radial, 300 110, 10, 300 100, 100, from(#f4f201), to(rgb(228,
199,0,0)), color-stop(80%, #e4c700));
 -webkit-background-origin: padding-box;
 -webkit-background-clip: content-box;
}

```



224


```

```
<div></div>
```

7.1.4 渐变的其他应用

除了定义渐变背景外，还可以定义渐变边框、填充内容，以及设计图标等。下面分别结合示例进行说明。

□ 定义渐变效果的边框。

```
<style type="text/css">
div {
    border-width: 20px;
    width: 400px;
    height: 200px;
    margin: 20px;
    -webkit-border-image: -webkit-gradient(linear, left top, left bottom, from(#00abeb),
to(#fff), color-stop(0.5, #fff), color-stop(0.5, #66cc00)) 20;
}
</style>

<div></div>
```



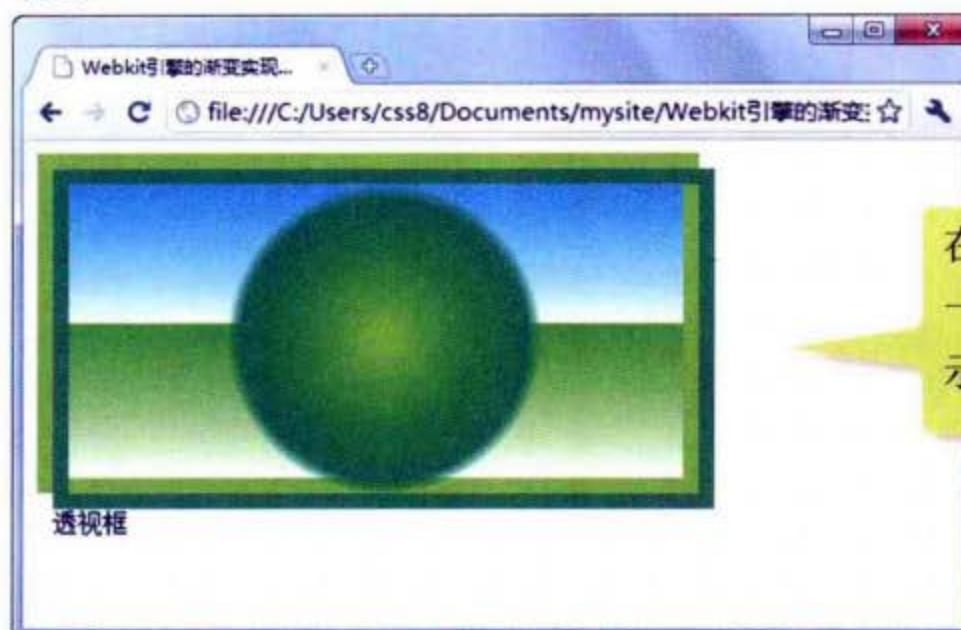
显示渐变效果的边框

□ 定义填充内容效果。

```
<style type="text/css">
.div1 {
    width:400px;
    height:200px;
    border:10px solid #A7D30C;
    /*为div元素定义渐变背景*/
    background: -webkit-gradient(linear, left top, left bottom, from(#00abeb), to(#fff),
color-stop(0.5, #fff), color-stop(0.5, #66cc00));
    float:left;
}
.div1::before {
    width:400px;
    height:200px;
```

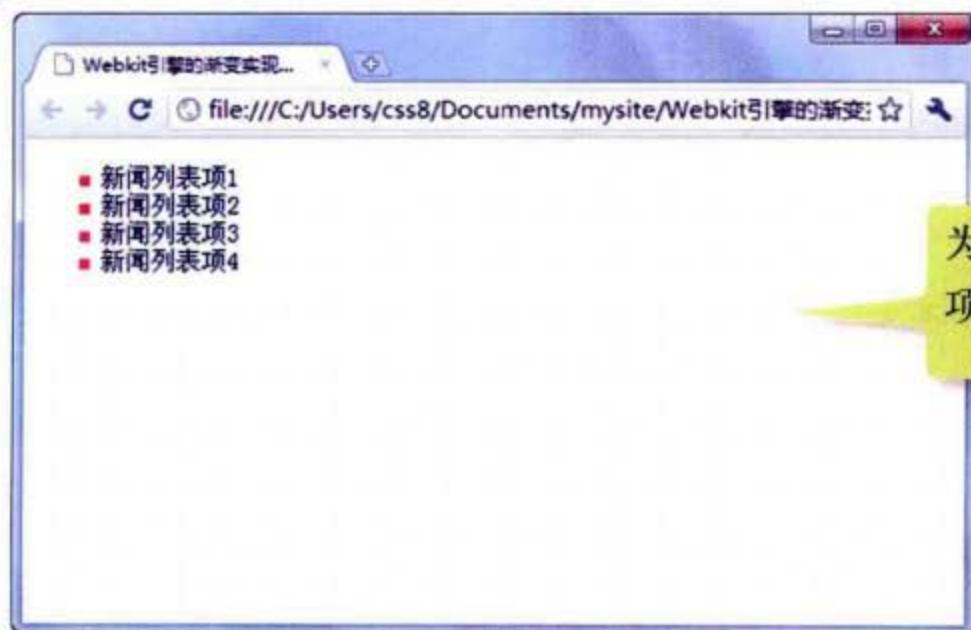


```
border:10px solid #019F62;  
/*在div元素前面插入一个内容对象，在该对象中绘制一个球体，并定义显示边框效果*/  
content: -webkit-gradient(radial, 200 100, 10, 200 100, 100, from(#A7D30C),  
to(rgb(1, 159, 98, 0)), color-stop(90%, #019F62));  
display: block;  
}  
</style>  
  
<div></div>
```



□ 定义列表图标。

```
<style type="text/css">  
ul {  
    list-style-image: -webkit-gradient(radial, center center, 4, center center, 8,  
from(#ff0000), to(rgb(0, 0, 0, 0)), color-stop(90%, #dd0000))  
}  
</style>  
  
<ul>  
    <li>新闻列表项1</li>  
    <li>新闻列表项2</li>  
    <li>新闻列表项3</li>  
    <li>新闻列表项4</li>  
</ul>
```



7.2 Gecko引擎的CSS渐变实现方法

Firefox浏览器从3.6版本开始支持渐变设计，详细信息请参阅<http://hacks.mozilla.org/2009/11/css-gradients-firefox-36/>页面说明。不过，Gecko引擎与Webkit引擎的用法不同，下面将详细说明。

7.2.1 直线渐变基本语法

Gecko引擎与Webkit引擎在渐变设计时用法不同，Gecko引擎定义了两个私有函数，分别用来设计直线渐变和径向渐变。基本语法说明如下。

```
-moz-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]* )
```

该函数的参数说明如下：

- <point>：定义渐变起始点，取值包含数值、百分比，也可以使用关键字，其中left、center和right关键字定义x轴坐标，top、center和bottom关键字定义y轴坐标。用法与background-position和-moz-transform-origin属性中的定位方式相同。当指定一个值时，则另一个值默认为center。
- <angle>：定义直线渐变的角度。单位包括deg（度，一圈等于360deg）、grad（梯度，90度等于100grad）、rad（弧度，一圈等于2*PI rad）。
- <stop>：定义步长，用法与Webkit引擎的color-stop()函数相似，但是该参数不需要调用函数，直接传递参数即可。其中第一个参数值设置颜色，可以为任何合法的颜色值，第二个参数设置颜色的位置，取值为百分比（0~100%）或者数值，也可以省略步长设置。

关于-moz-linear-gradient()函数用法的详细信息，请参阅<https://developer.mozilla.org/en/CSS/-moz-linear-gradient>网页。

7.2.2 直线渐变的基本用法

下面结合示例分别演示Gecko引擎的直线渐变实现方法。

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -moz-linear-gradient(red, blue);
}
</style>

<div></div>
```

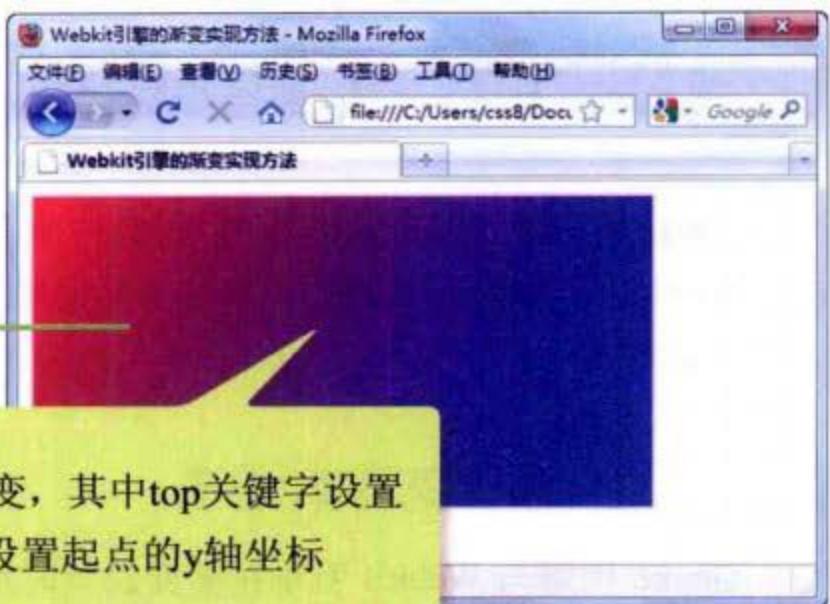


最简单的线性渐变，只需要指定开始颜色和结束颜色，则默认从上到下实施线性渐变

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-linear-gradient(top left, red, blue);
}
</style>
```

<div></div>

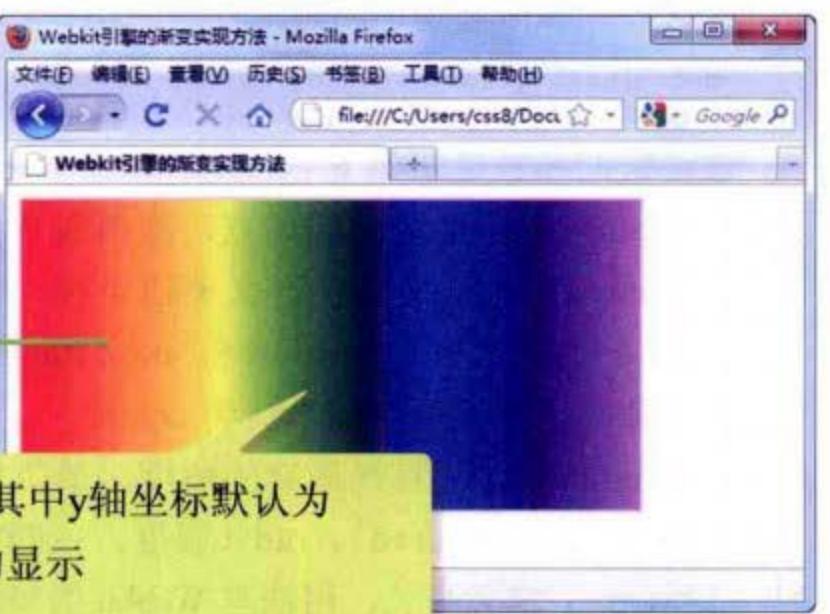
从左上角到右下角的线性渐变，其中top关键字设置起点的x轴坐标，left关键字设置起点的y轴坐标



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-linear-gradient(left, red, orange,
                            yellow, green, blue, indigo, violet);
}
</style>
```

<div></div>

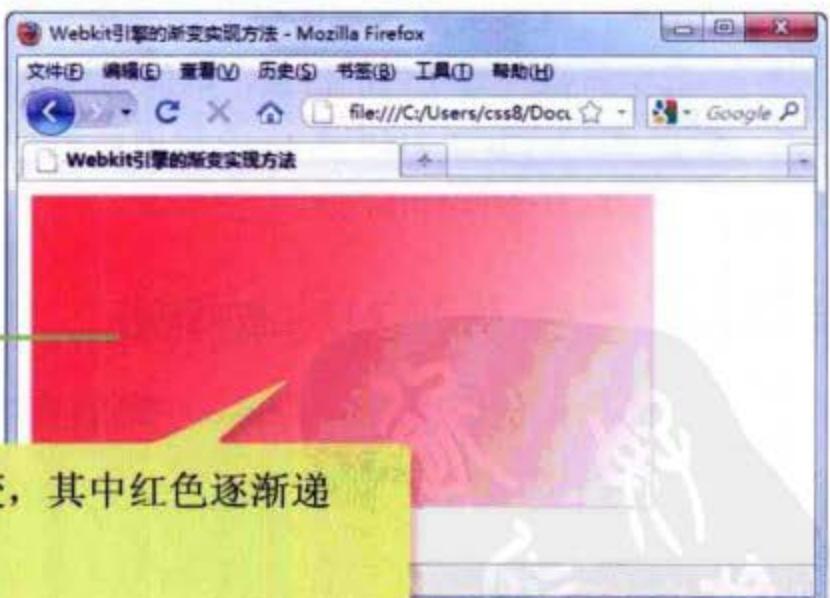
设计从左到右的五彩渐变，其中y轴坐标默认为center，多个色标按步长平均显示



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-linear-gradient(top left,
                            red, rgba(255,0,0,0));
}
</style>
```

<div></div>

从左上角到右下角的红色渐变，其中红色逐渐递弱，并最终显示为透明



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
```

```

background:
-moz-linear-gradient(left 90deg,
red, rgba(255,0,0,0));
}

</style>

<div></div>

```

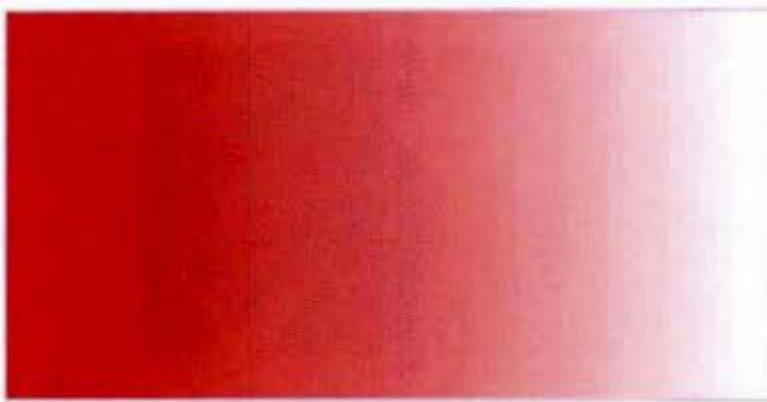


从下到上的红色渐变，其中红色逐渐递弱，本示例通过定义旋转角度定义渐变的显示方向

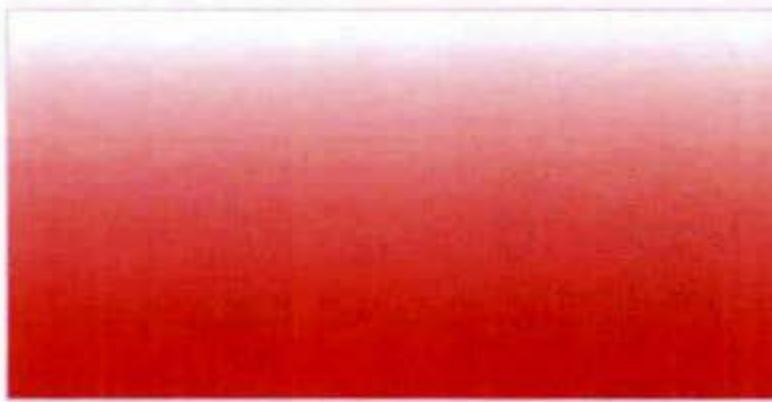
简单小结一下：当指定角度时，请记住，它是沿水平线按逆时针旋转定位的。因此，设置0deg，将产生从左向右的水平渐变，而设置90度，将创建一个从底部到顶部的渐变，如下图所示。



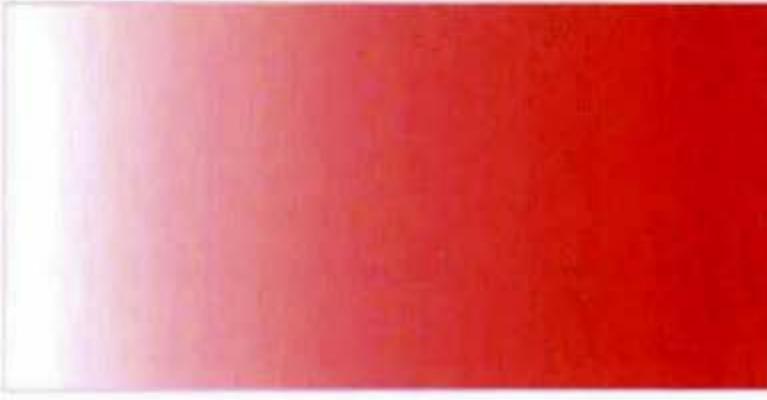
-moz-linear-gradient(0deg, red, rgba(255,0,0,0)



-moz-linear-gradient(90deg, red, rgba(255,0,0,0)



-moz-linear-gradient(180deg, red, rgba(255,0,0,0)



-moz-linear-gradient(-90deg, red, rgba(255,0,0,0)



```

<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
}

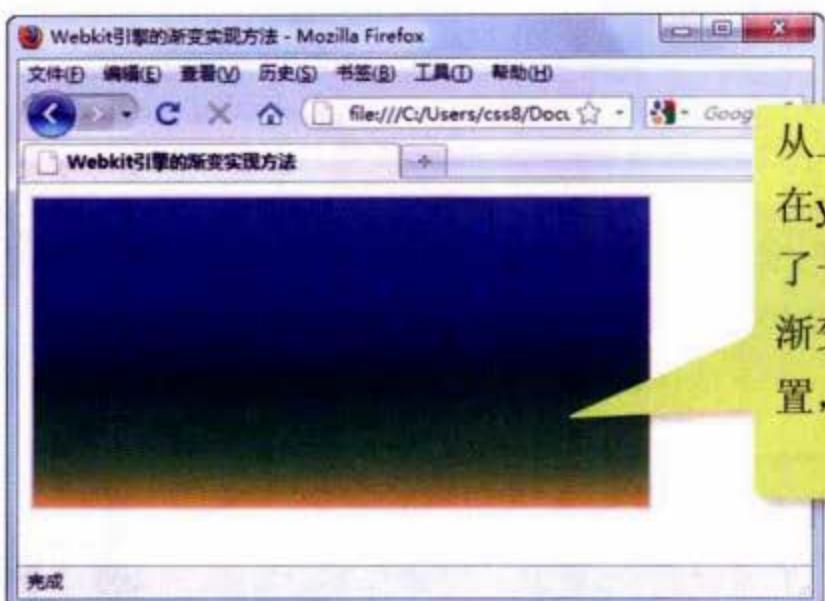
```

```

background: -moz-linear-gradient(top, blue, green 80%, orange);
}
</style>

<div></div>

```



从上到下的多色渐变，其中在y轴的80%的位置，添加了一个绿色色标，设计三色渐变效果。如果没有指定位，则三色会均匀分布

```

<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -moz-linear-gradient(right,
        rgba(255,255,255,0),
        rgba(255,255,255,1)),
        url(images/bq4.jpg);
}
</style>

<div></div>

```



设计渐变半透明效果的背景图像。在背景图像上面覆盖一层从左到右为白色到透明的渐变填充层

7.2.3 径向渐变基本语法

Gecko 引擎定义的径向渐变基本语法如下。

```
-moz-radial-gradient( [<position> || <angle>,]? [<shape> || <size>,]? <stop>, <stop>[,<stop>]* )
```

该函数的参数说明如下：

- <point>：定义渐变起始点，取值包含数值、百分比，也可以使用关键字，其中 left、center 和 right 关键字定义 x 轴坐标，top、center 和 bottom 关键字定义 y 轴坐标。用法与 background-position 和 -moz-transform-origin 属性中的定位方式相同。当指定一个值时，则另一个值默认为 center。
- <angle>：定义渐变的角度。单位包括 deg（度，一圈等于 360deg）、grad（梯度，90

度等于 100grad)、rad(弧度，一圈等于 2π rad)，默认值为 0deg。

- <shape>：定义径向渐变的形状，包括 circle(圆) 和 ellipse(椭圆)，默认值为 ellipse。
- <size>：定义圆半径，或者椭圆的轴长度。
- <stop>：定义步长，用法与 Webkit 引擎的 color-stop() 函数相似，但是该参数不需要调用函数，直接传递参数即可。其中第一个参数值设置颜色，可以为任何合法的颜色值，第二个参数设置颜色的位置，取值为百分比(0 ~ 100%) 或者数值，也可以省略步长设置。

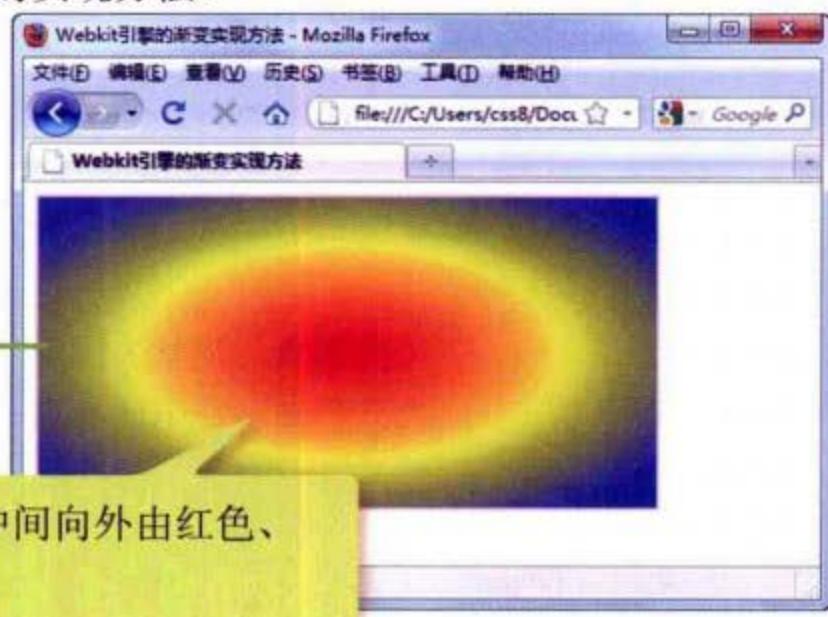
有关 -moz-radial-gradient() 函数用法的详细信息，请参阅 <https://developer.mozilla.org/en/CSS/-moz-radial-gradient> 网页。

7.2.4 径向渐变的基本用法

下面结合示例演示 Gecko 引擎的径向渐变的实现方法。

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-radial-gradient(red, yellow, blue);
}
</style>

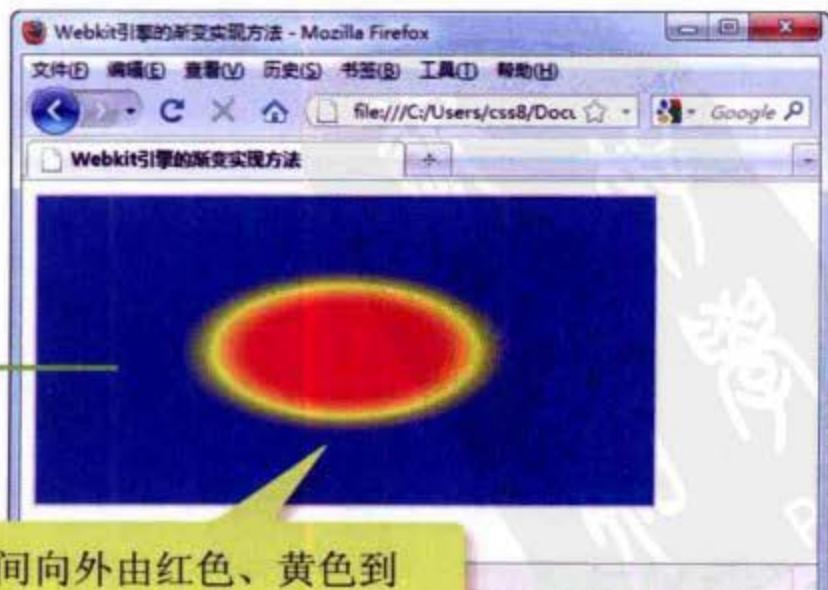
<div></div>
```



最简单的径向渐变，从中间向外由红色、黄色到蓝色渐变显示

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-radial-gradient(red 20%,
                            yellow 30%, blue 40%);
}
</style>

<div></div>
```



最简单的径向渐变，从中间向外由红色、黄色到蓝色渐变显示，并设置不同色标的显示位置

```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-radial-gradient(bottom left,
        red, yellow, blue 80%);
}
</style>
```

径向渐变，从左下角向外由红色、黄色到蓝色渐变显示，并设置蓝色色标的显示位置



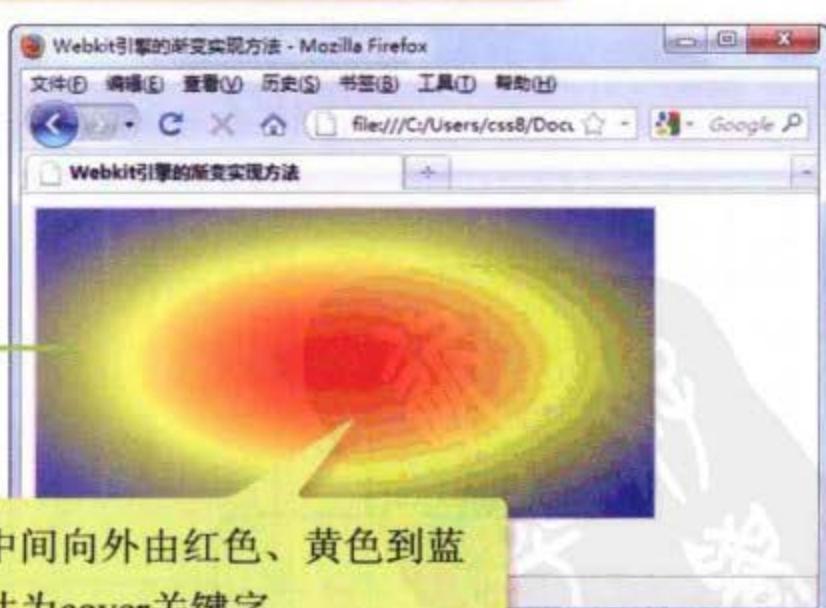
```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-radial-gradient(left,
        circle, red, yellow, blue 50%);
}
</style>
```

径向渐变，形状为圆形。从左侧中间向外由红色、黄色到蓝色渐变显示，并设置蓝色色标的显示位置



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background:
        -moz-radial-gradient(ellipse cover,
        red, yellow, blue);
}
</style>
```

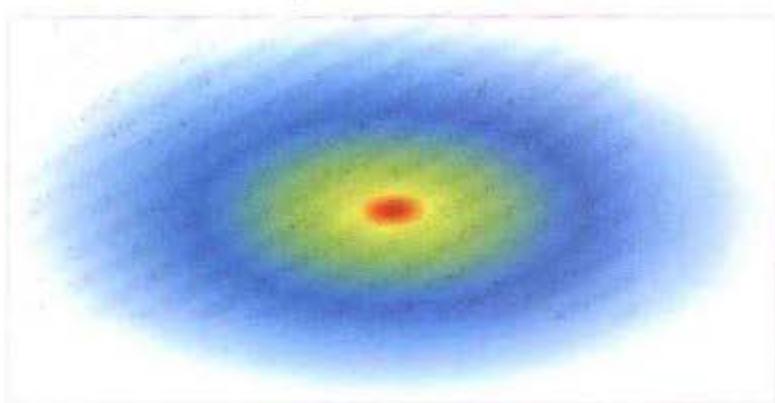
径向渐变，形状为椭圆。从中间向外由红色、黄色到蓝色渐变显示，并设置渐变尺寸为cover关键字



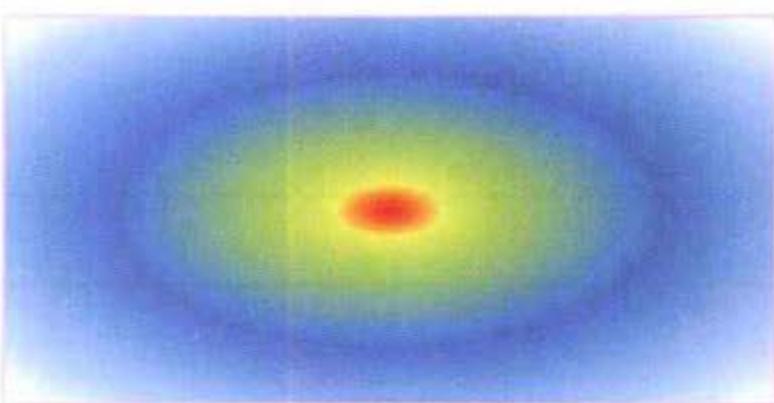
简单小结一下：size参数包含多个关键字，closest-side、closest-corner、farthest-side、farthest-corner、contain和cover。使用这些关键字可以定义径向渐变的大小。



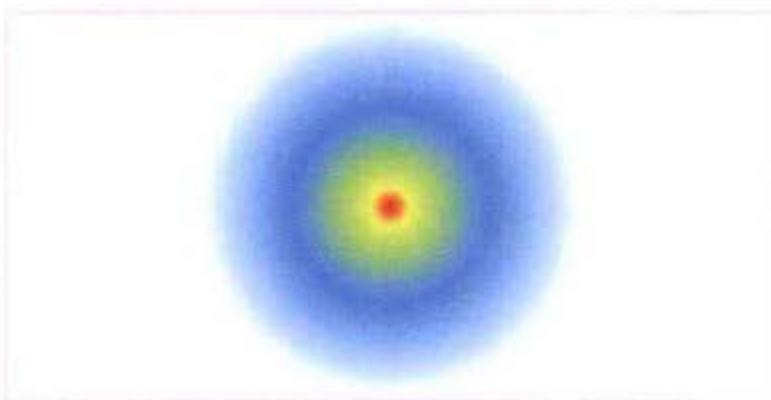
-moz-radial-gradient(ellipse closest-side,
red, yellow 10%, #1E90FF 50%, white)



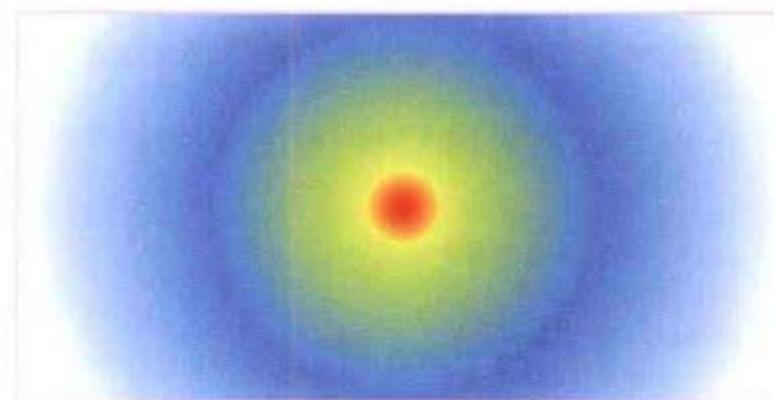
-moz-radial-gradient(ellipse farthest-corner, red, yellow 10%, #1E90FF 50%, white)



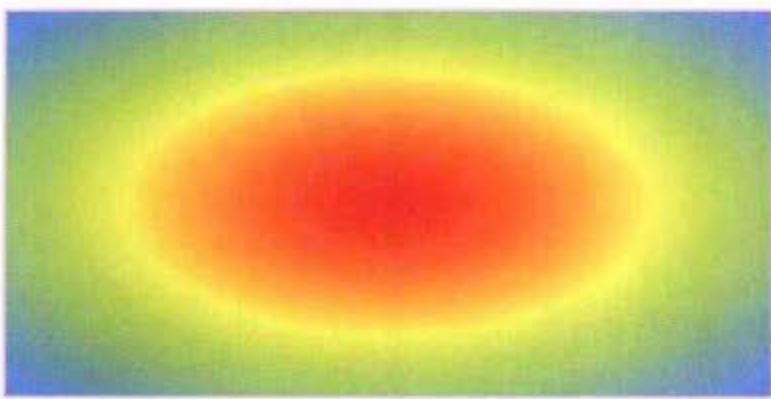
-moz-radial-gradient(circle closest-side,
red, yellow 10%, #1E90FF 50%, white)



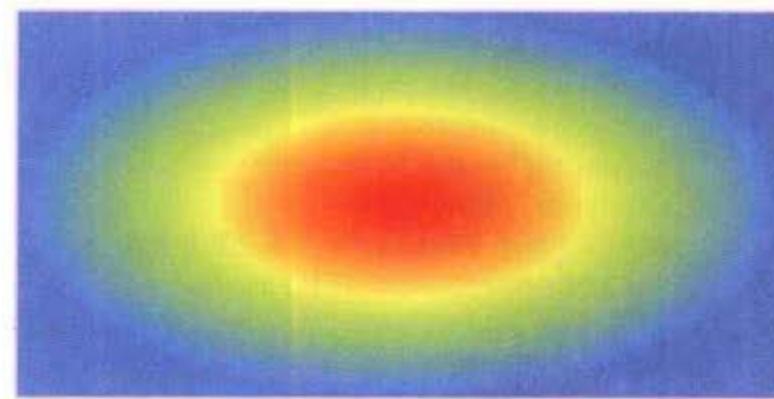
-moz-radial-gradient(circle farthest-side,
red, yellow 10%, #1E90FF 50%, white)



-moz-radial-gradient(red, yellow, #1E90FF)



-moz-radial-gradient(contain, red,
yellow, #1E90FF)



另外，Gecko引擎还定义了-moz-repeating-linear-gradient 和-moz-repeating-radial-gradient两个属性，用来定义重复直线渐变和重复径向渐变。如下图所示。



```
<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -moz-repeating-radial-gradient(
        circle, black, black 10px, white 10px, white 20px);
}</style>
```



```

<style type="text/css">
div {
    width:400px;
    height:200px;
    border:2px solid #FCF;
    padding: 4px;
    background: -moz-repeating-linear-gradient(top
left 60deg,black, black 10px, white 10px, white 20px);}
</style>

```



7.2.5 渐变的应用

下面结合两个示例演示 Gecko 引擎渐变的应用。示例代码如下，演示效果如图 7.1 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Webkit引擎的应用</title>
<style type="text/css">
body { /*页面初始化*/
    background-color: #454545;
    margin: 0;
    padding: 0;
}
.box { /*设计模块样式*/
    -moz-border-radius: 10px; /* 设计圆角 */
    -moz-box-shadow: 0 0 12px 1px rgba(205, 205, 205, 1); /* 设计阴影特效 */
    border: 1px solid black;
    padding: 10px;
    max-width: 600px; /* 最大显示宽度 */
    margin: auto; /* 居中显示 */
    text-shadow: black 1px 2px 2px; /* 设计包含文本阴影 */
    color: white;
    /* 设计直线渐变背景 */
    background-image: -moz-linear-gradient(bottom, black, rgba(0, 47, 94, 0.2), white);
    background-color: rgba(43, 43, 43, 0.5);
}
.box:hover { /* 设计鼠标经过时，放大阴影亮度 */
    -moz-box-shadow: 0 0 12px 5px rgba(205, 205, 205, 1);
}
h2 {
    font-size: 120%;
    font-weight: bold;
    text-decoration: underline;
}
h2:before { /* 在标题前面添加额外内容 */
    content: "标题：";
}
p {
    padding: 6px;
    text-indent: 2em;
}

```

```

        line-height:1.8em;
        font-size:14px;
    }
</style>
</head>
<body>
<div class="box">
    <h2>关于文字</h2>
    <p>我对随便哪种感觉的文字上手都很快。曾经我用一天的时间看完《第一次亲密接触》然后第二天就写出了两万多字类似的东西，把同学吓得目瞪口呆。尽管我认为那种东西几乎没有存在的价值，时光可以轻而易举地把它淹没得不留一丝痕迹。</p>
    <p>我把考试中得到满分的作文随便丢掉，却把老师说的毫无内涵的文章装订好放在抽屉里。我常把自己的故事写下来，然后拿给同学看，然后他们感动得一塌糊涂。</p>
    <p>.....</p>
</div>
</body>
</html>

```

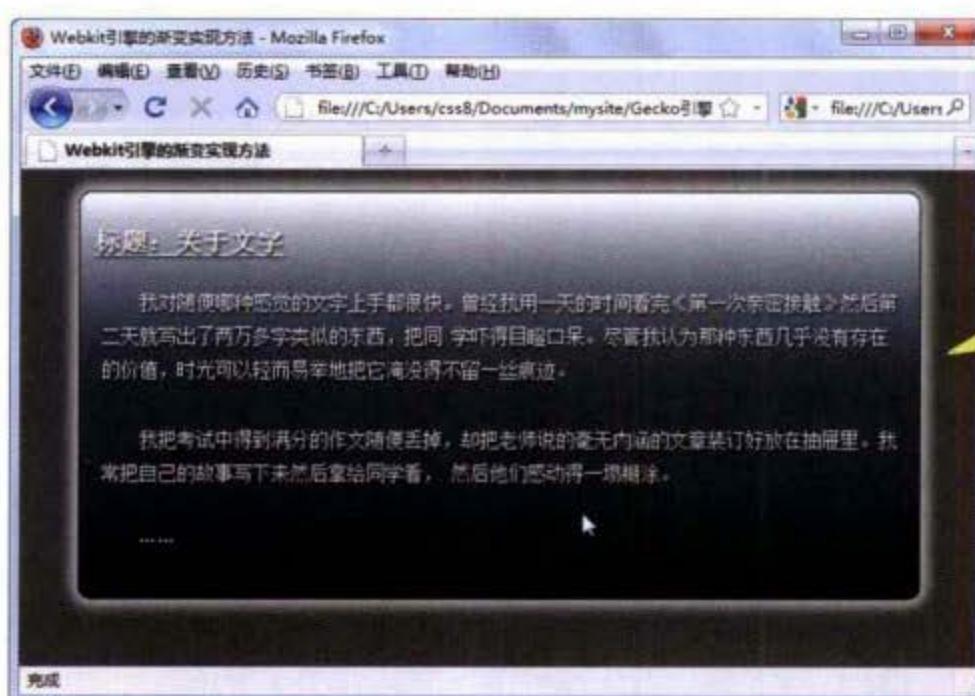


图7.1 为模块设计渐变背景

7.3 IE 浏览器引擎的 CSS 渐变实现方法

IE并不支持CSS渐变，但是提供了渐变滤镜，可以用来实现简单的渐变效果。虽然在标准设计中不提倡使用IE滤镜设计页面效果，但是考虑到IE浏览器的市场占有率，下面我们简单介绍IE渐变滤镜的使用。

7.3.1 基本语法

IE浏览器中渐变滤镜的基本语法如下。

```
filter:progid:DXImageTransform.Microsoft.Gradient(enabled=bEnabled,startColorStr=iWidth,endColorStr=iWidth)
```

该函数的参数说明如下：

- enabled**：设置或检索滤镜是否激活。可选布尔值，包括 true 和 false，默认值为 true，表示激活状态。
- startColorStr**：设置或检索色彩渐变的开始颜色和透明度。可选项，其格式为 #AARRGGBB。AA、RR、GG、BB 为十六进制正整数，取值范围为 00 ~ FF。RR 指定红色值，GG 指定绿色值，BB 指定蓝色值。AA 指定透明度，00 是完全透明，FF 是完全不透明。超出取值范围的值将被恢复为默认值。取值范围为 #FF000000 ~ #FFFFFF，默认值为 #FF0000FF，即不透明蓝色。
- EndColorStr**：设置或检索色彩渐变的结束颜色和透明度。默认值为 #FF000000，即不透明黑色。

关于 IE 滤镜用法的详细信息，请参阅 [http://msdn.microsoft.com/en-us/library/ms532997\(VS.85,loband\).aspx](http://msdn.microsoft.com/en-us/library/ms532997(VS.85,loband).aspx)。注意，IE 滤镜仅在 IE 5.5 及以上版本的浏览器中有效。

7.3.2 IE 滤镜实战应用

下面结合示例演示 IE 滤镜所提供的渐变应用方法。在下面这个示例中，利用 IE 滤镜设计渐变背景，然后通过背景图像设计图文插画效果，如图 7.2 所示。

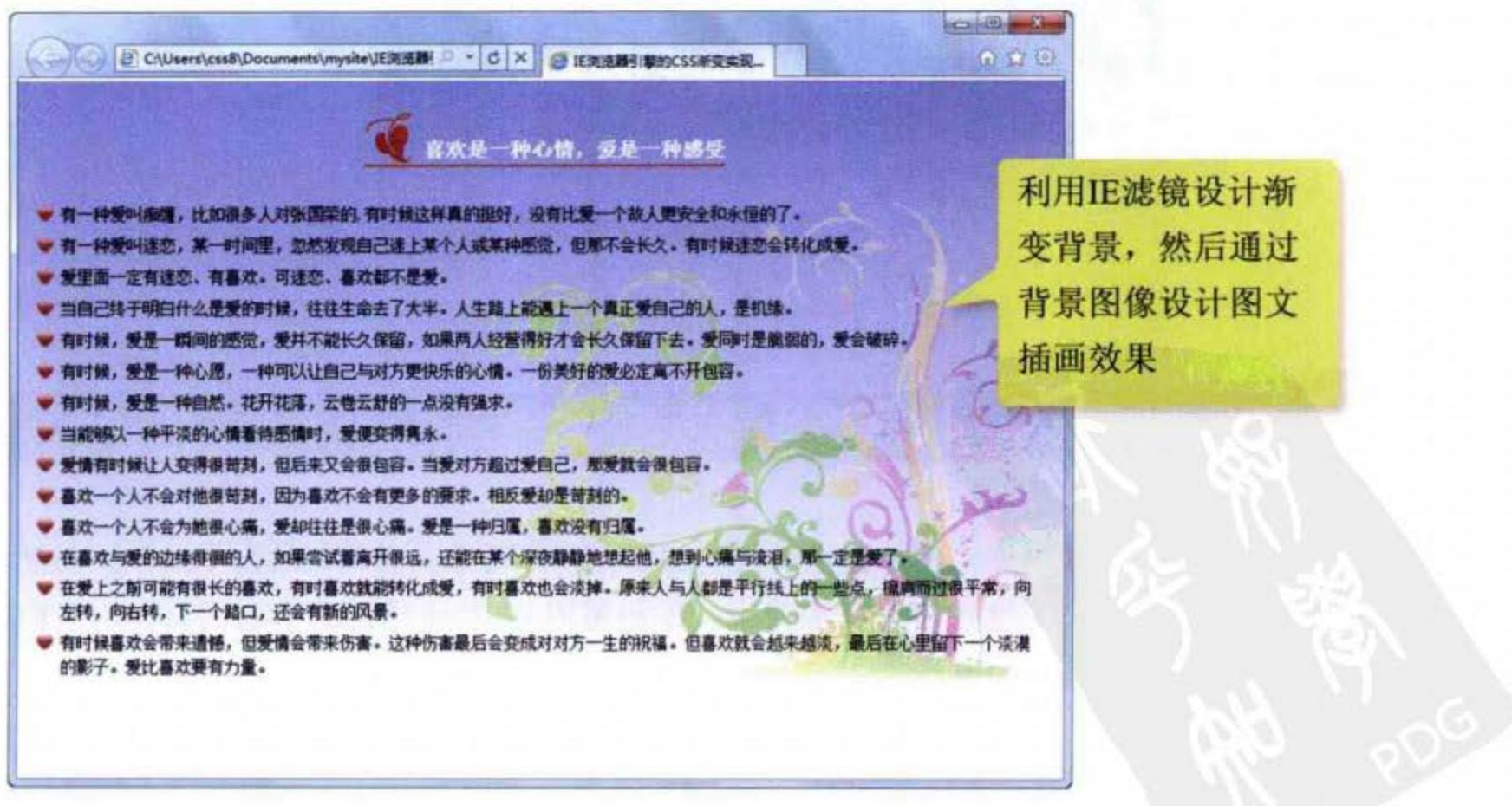


图 7.2 使用 IE 滤镜设计的渐变背景

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>IE浏览器引擎的CSS渐变实现方法</title>
<style type="text/css">
body {
    padding:1em;
    margin:0;
    text-align:center;
    /* 为网页设计垂直渐变背景 */
    filter: progid:DXImageTransform.Microsoft.Gradient(gradientType=0, startColorStr=#9999FF,
endColorStr=#ffffff);
}
h1 { /* 标题样式 */
    color:white;
    font-size:18px;
    height:45px;
    line-height:65px; /* 控制文本显示位置 */
    position:absolute;
    left:50%;
    margin-left:-150px;
    border-bottom:solid 2px #c72223;
    background:url(images/icon4.png) no-repeat left center;
    padding-left:50px;
}
ul { /* 列表框样式 */
    /* 清除列表默认样式 */
    list-style-type:none;
    margin:90px 0 0 0;
    padding:0;
    background:url(images/bg6.png) no-repeat right bottom; /* 设计插画背景 */
    text-align:left; /* 恢复默认对齐方式 */
}
li {
    line-height:1.5em;
    margin:6px auto;
    font-size:14px;
    background:url(images/icon3.png) no-repeat left 3px;
    padding-left:20px;
}
</style>
</head>

<body>
<h1>喜欢是一种心情，爱是一种感受</h1>
<ul>
    <li>有一种爱叫痴缠，比如很多人对张国荣的，有时候这样真的挺好，没有比爱一个故人更安全和永恒的了。</li>
    <li>有一种爱叫迷恋，某一时间里，忽然发现自己迷上某个人或某种感觉，但那不会长久。有时候迷恋会转化成爱。</li>
    <li>爱里面一定有迷恋、有喜欢。可迷恋、喜欢都不是爱。</li>
    <li>当自己终于明白什么是爱的时候，往往生命去了大半。人生路上能遇上一个真正爱自己的人，是机缘。</li>
    <li>有时候，爱是一瞬间的感觉，爱并不能长久保留，如果两人经营得好才会长久保留下去。爱同时是脆弱的，爱会破碎。</li>
    <li>有时候，爱是一种心愿，一种可以让自己与对方更快乐的心情。一份美好的爱必定离不开包容。</li>
    <li>有时候，爱是一种自然。花开花落，云卷云舒的一点没有强求。</li>

```

让渐变背景填满整个页面

以绝对定位方式实现块元素居中显示

为标题插入一个装饰图标

为列表项设计个性化的列表图标

```

<li>当能够以一种平淡的心情看待感情时，爱便变得隽永。</li>
<li>爱情有时候让人变得很苛刻，但后来又会很包容。当爱对方超过爱自己，那爱就会很包容。</li>
<li>喜欢一个人不会对他很苛刻，因为喜欢不会有更多的要求。相反爱却是苛刻的。</li>
<li>喜欢一个人不会为她很心痛，爱却往往是很心痛。爱是一种归属，喜欢没有归属。</li>
<li>在喜欢与爱的边缘徘徊的人，如果尝试着离开很远，还能在某个深夜静静地想起他，想到心痛与流泪，那一定是爱了。</li>
<li>在爱上之前可能有很长的喜欢，有时喜欢就能转化成爱，有时喜欢也会淡掉。原来人与人都是平行线上的一些点，擦肩而过很平常，向左转，向右转，下一个路口，还会有新的风景。</li>
<li>有时候喜欢会带来遗憾，但爱情会带来伤害。这种伤害最后会变成对对方一生的祝福。但喜欢就会越来越淡，最后在心里留下一个淡漠的影子。爱比喜欢要有力量。</li>
</ul>
</body>
</html>

```

另外，IE渐变滤镜还可以设置渐变类型，通过GradientType属性来定义，取值包括0和1，其中1表示水平渐变，该值为默认值，0表示垂直渐变。应用可以参考上面示例中的滤镜函数Gradient()的第一个参数值。



7.4 W3C 标准化的 CSS 渐变实现方法

W3C 组织于 2010 年 11 月才正式发布支持渐变设计的工作草案，不过渐变设计并没有单独列作为一个模块，而是作为“图像值和图像被替换内容”模块的一部分列出来的，详细资料请参阅 <http://dev.w3.org/csswg/css3-images/#gradients>。由于提出这个工作草案的时间还不长，所以目前还没有一家主流浏览器对其进行支持，相信不久的将来，主流浏览器会对其进行支持。

标准草案沿袭 Gecko 引擎的渐变设计方法，语法和用法也基本相同，简单比较如表 7.1 所示。

表 7.1 标准渐变和Gecko引擎渐变用法比较

类型	Gecko引擎语法	W3C标准语法
线性渐变	<pre> -moz-linear-gradient([<point> <angle> ,]? <stop>, <stop> [, <stop>]*); </pre>	<pre> linear-gradient([[[top bottom] [left right]] <angle> ,]? <color-stop>[, <color-stop>]+); </pre>
径向渐变	<pre> -moz-radial-gradient([<position> <angle>],? [<shape> <size>,]? <stop>, <stop>[, <stop>]*); </pre>	<pre> radial-gradient([<bg-position> <angle>],? [[<shape> <size>] [<length> <percentage>]{2} ,]? <color-stop>[, <color-stop>]+); </pre>

W3C 标准用法与 Gecko 引擎的渐变用法基本相同，本节就不再重复解释了，请读者参阅前面的章节对照学习，同时也可以访问 <http://dev.w3.org/csswg/css3-images/#gradients> 阅读标准渐变的基本用法。考虑到目前浏览器还没有支持，所以我们也无法进行测试。

最后，针对 CSS 3 渐变给初学者提出几点建议：

- 积极使用 CSS 渐变，虽然各主流浏览器用法不统一，支持不尽相同，可以考虑定义背景色作为备用方案，在无法显示渐变背景时以纯色进行显示。
- IE 8 及其以下版本的浏览器、Opera、Safari 3 及其以下版本的浏览器、Firefox 3 及其以下版本的浏览器不能渲染 CSS 3 渐变。不过 Firefox、Safari 和 Chrome 都可以主动升级浏览器，从而能够避开这个问题。
- 针对 IE 浏览器，可以通过渐变滤镜进行兼容，虽然不提倡使用，但是可以作为一种备用用法来了解。
- 在设计 CSS 渐变时，考虑到不同浏览器用法的差异很大，不用设计在每个浏览器里面完全相同的渐变效果。
- 考虑到不是所有浏览器都支持 CSS 渐变，为安全起见，不应该在编写布局代码的时候依赖 CSS 渐变，而是仅使用 CSS 渐变来增强布局。

7.5 CSS 3 渐变实战

本节将结合多个实战案例帮助读者快速掌握 CSS 3 渐变设计与开发。这些效果对于实践具有很强的指导性，希望读者能够学有所获。

7.5.1 CSS 渐变下拉菜单

多级菜单的设计方法有多种，实战中多使用 JavaScript+CSS 方法实现，纯 CSS 设计的多级菜单也有很多，但是考虑到浏览器的兼容性以及代码高效性的要求，实际使用比较少。当然，对于初学者来说，通过纯 CSS 的多级菜单练习，可以掌握 CSS 的灵活用法。

下面是一个纯 CSS 渐变下拉菜单。其中结合使用了 CSS 3 的 text-shadow、radius-border、box-shadow 等属性，设计出一个比较漂亮的多级渐变菜单，页面没有用到任何脚本和图片，是一个完全 CSS 实现的效果，如图 7.3 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS渐变下拉菜单</title>
```



图 7.3 CSS 演变多级菜单效果

```

<style type="text/css">
body { background: #ebebeb; width: 900px; margin: 20px auto; color: #666; }
a { color: #333; }
#nav { /* 定义菜单外框样式 */
    margin: 0;
    padding: 7px 6px 0;
    line-height: 100%; /* 实现菜单项垂直居中 */
    border-radius: 2em;
    -webkit-border-radius: 2em;
    -moz-border-radius: 2em;
    -webkit-box-shadow: 0 1px 3px rgba(0, 0, 0, .4);
    -moz-box-shadow: 0 1px 3px rgba(0, 0, 0, .4);
    /* 下面将为菜单外框添加渐变背景效果 */
    background: #8b8b8b; /* 兼容不支持CSS3的浏览器 */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#a9a9a9',
    endColorstr='#7a7a7a'); /* 兼容 IE */
    background: -webkit-gradient(linear, left top, left bottom, from(#a9a9a9), to(#7a7a7a));
    /* 兼容 Webkit 引擎浏览器 */
    background: -moz-linear-gradient(top, #a9a9a9, #7a7a7a); /* 兼容 Firefox 3.6+ */
    border: solid 1px #6d6d6d;
}
#nav li { /* 定义子菜单列表项样式 */
    margin: 0 5px;
    padding: 0 0 8px;
    float: left; /* 实现水平并列显示 */
    position: relative; /* 定义包含块，以便绝对定位子元素 */
    list-style: none;
}
/* 主菜单项链接样式：默认效果 */
#nav a {
    font-weight: bold;
    color: #e7e5e5;
    text-decoration: none;
    display: block;
    padding: 8px 20px;
    margin: 0;
}

```

设计圆角外框及其
阴影效果

```

-webkit-border-radius: 1.6em;
-moz-border-radius: 1.6em;
text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
}

/* 主菜单项链接样式：鼠标经过时效果 */
#nav .current a, #nav li:hover > a {
    background: #d1d1d1; /* 兼容不支持 CSS3 的浏览器 */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#ebebeb',
endColorstr='#alalal'); /* 兼容 IE */
    background: -webkit-gradient(linear, left top, left bottom, from(#ebebeb), to(#alalal));
/* 兼容 Webkit 浏览器 */
    background: -moz-linear-gradient(top, #ebebeb, #alalal); /* 兼容 Firefox 3.6+ */
    color: #444;
    border-top: solid 1px #f8f8f8;
    -webkit-box-shadow: 0 1px 1px rgba(0, 0, 0, .2);
    -moz-box-shadow: 0 1px 1px rgba(0, 0, 0, .2);
    box-shadow: 0 1px 1px rgba(0, 0, 0, .2);
    text-shadow: 0 1px 0 rgba(255, 255, 255, .8);
}

/* 子菜单链接样式：鼠标经过时效果 */
#nav ul li:hover a, #nav li:hover li a {
    background: none;
    border: none;
    color: #666;
    -webkit-box-shadow: none;
    -moz-box-shadow: none;
}

#nav ul a:hover {
    background: #0399d4 !important; /* 兼容不支持 CSS3 的浏览器 */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#04acec',
endColorstr='#0186ba'); /* 兼容 IE */
    background: -webkit-gradient(linear, left top, left bottom, from(#04acec), to(#0186ba))
!important; /* 兼容 Webkit 浏览器 */
    background: -moz-linear-gradient(top, #04acec, #0186ba) !important; /* 兼容 Firefox 3.6+ */
    color: #fff !important;
    -webkit-border-radius: 0;
    -moz-border-radius: 0;
    text-shadow: 0 1px 1px rgba(0, 0, 0, .1);
}

/* 二级列表项样式 */
#nav ul {
    background: #ddd; /* 兼容不支持 CSS3 的浏览器 */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffffff',
endColorstr='#cfcfcf'); /* 兼容 IE */
    background: -webkit-gradient(linear, left top, left bottom, from(#fff), to(#cfcfcf)); /* 兼容 Webkit 浏览器 */
    background: -moz-linear-gradient(top, #fff, #cfcfcf); /* 兼容 Firefox 3.6+ */
    display: none; /* 在默认状态下隐藏子菜单列表框 */
    margin: 0;
    padding: 0;
    width: 185px;
    position: absolute; /* 通过绝对定位，能够更精确地控制子菜单的显示位置 */
    top: 35px;
    left: 0;
    border: solid 1px #b4b4b4;
}

```

设计圆角及其字体
阴影效果

设计盒子阴影及其
字体阴影效果

```

        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
        -webkit-box-shadow: 0 1px 3px rgba(0, 0, 0, .3);
        -moz-box-shadow: 0 1px 3px rgba(0, 0, 0, .3);
        box-shadow: 0 1px 3px rgba(0, 0, 0, .3);

    }

    /* 定义鼠标经过时显示子菜单 */
    #nav li:hover > ul { display: block; }

    #nav ul li {
        float: none;
        margin: 0;
        padding: 0;
    }

    #nav ul a {
        font-weight: normal;
        text-shadow: 0 1px 1px rgba(255, 255, 255, .9);
    }

    /* 三级菜单列表框样式 */
    #nav ul ul {
        left: 181px;
        top: -3px;
    }

    /* 为第一个和最后一个子元素设置圆角效果 */
    #nav ul li:first-child > a { /* 设置第一个子元素的圆角效果 */
        -webkit-border-top-left-radius: 9px;
        -moz-border-radius-topleft: 9px;
        -webkit-border-top-right-radius: 9px;
        -moz-border-radius-topright: 9px;
    }

    #nav ul li:last-child > a { /* 设置最后一个子元素的圆角效果 */
        -webkit-border-bottom-left-radius: 9px;
        -moz-border-radius-bottomleft: 9px;
        -webkit-border-bottom-right-radius: 9px;
        -moz-border-radius-bottomright: 9px;
    }

    /* 清除固定高度，让菜单框自动张开包含所有子元素 */
    #nav:after {
        content: ".";
        display: block;
        clear: both;
        visibility: hidden;
        line-height: 0;
        height: 0;
    }

    #nav { display: inline-block; } /* 标准用法 */
    html[xmlns] #nav { display: block; } /* 兼容IE7 */
    * html #nav { height: 1%; } /* 兼容IE6 */

```

设计子菜单显示圆角边框，以及阴影效果

```

<li><a href="#">子菜单1</a></li>
<li><a href="#">子菜单2</a></li>
<li><a href="#">子菜单3 >></a>
    <ul>
        <li><a href="#">三级菜单1</a></li>
        <li><a href="#">三级菜单2</a></li>
        <li><a href="#">三级菜单3 >></a>
            <ul>
                <li><a href="#">四级菜单1</a></li>
                <li><a href="#">四级菜单2</a></li>
                <li><a href="#">四级菜单3</a></li>
            </ul>
        </li>
    </ul>
</li>
<li><a href="#">主菜单2</a></li>
<li><a href="#">主菜单3</a></li>
</ul>
</body>
</html>

```

设计符合标准、简洁且又有规律的菜单结构，是整个案例效果得以实现的基础

7.5.2 CSS 渐变按钮

按钮是网页设计中的重要对象，也是设计师非常重视的设计角色。在传统设计中，设计师主要借助 Photoshop 前期实现按钮效果，这种方法比较灵活，也比较安全，唯一的缺陷就是适应能力比较弱，需要重复返回 Photoshop 中进行修改或者设计，重用性和扩展性不是很高。

现在，依赖 CSS 3 新增的渐变、阴影、圆角等功能可以摆脱 Photoshop 的束缚，直接使用 CSS 快速设计各种精巧、美观的按钮。纯 CSS 渐变按钮可以根据字体大小自动伸缩，设计师也可以通过修改 padding 和 font-size 属性值来调整按钮大小，同时还可以应用到任何 HTML 元素，如 div、span、p、a、button、input 等。简单概括来说，本案例所设计的纯 CSS 渐变按钮具有如下优势，演示效果如图 7.4 所示。

- 不需要使用图片和 JavaScript。
- 能够兼容 IE、Firefox 3.6+、Chrome 和 Safari 等主流浏览器，不兼容 Opera 浏览器。
- 支持三种按钮状态，如正常、悬停和激活。
- 可以应用到任何 HTML 元素，如 a、input、button、span、div、p、h3 等。
- 安全兼容不支持 CSS 3 的浏览器，如果不兼容 CSS 3，则显示没有渐变和阴影的普通按钮。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS渐变按钮</title>

```

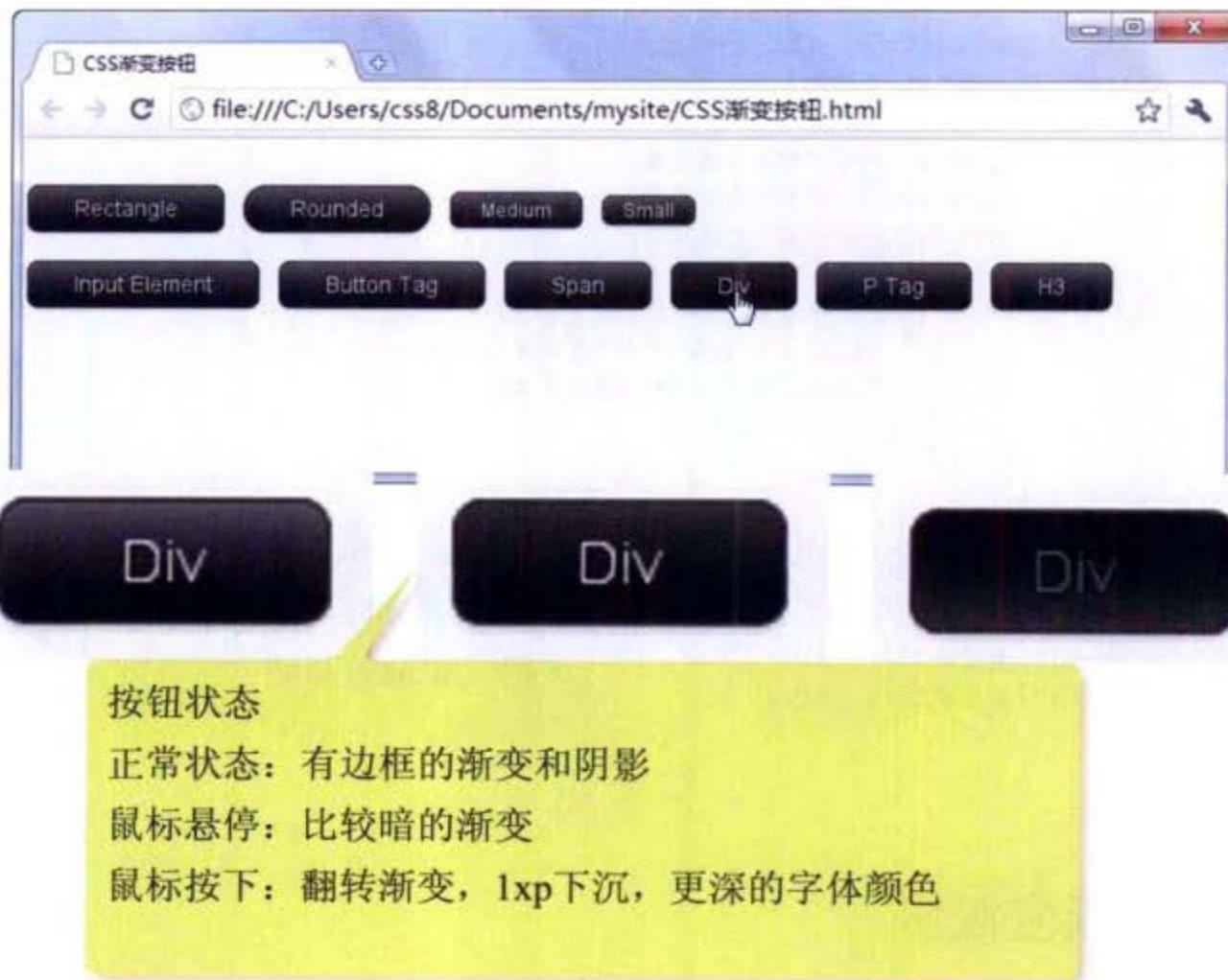


图 7.4 CSS 渐变按钮的效果

```

<style type="text/css">
body {
    background: #ebeded;
    margin: 30px auto;
    color: #999;
}

.button { /* 定义渐变按钮样式类 */
    display: inline-block;
    zoom: 1;
    *display: inline;
    vertical-align: baseline;
    margin: 0 2px;
    outline: none;
    cursor: pointer;
    text-align: center;
    text-decoration: none;
    font: 14px/100% Arial, Helvetica, sans-serif;
    padding: .5em 2em .55em;

    text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
    -webkit-border-radius: .5em;
    -moz-border-radius: .5em;
    border-radius: .5em;
    -webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
    -moz-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
    box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
}

```

zoom和*display属性为了兼容IE 7，都具有display:inline-block特性

设计按钮圆角、盒子阴影和文本阴影特效

```
.button:hover { text-decoration: none; }
.button:active {
    position: relative;
    top: 1px;
}
.bigrounded { /* 定义大圆角样式类 */
    -webkit-border-radius: 2em;
    -moz-border-radius: 2em;
    border-radius: 2em;
}
.medium { /* 定义大按钮样式类 */
    font-size: 12px;
    padding: .4em 1.5em .42em;
}
.small { /* 定义小按钮样式类 */
    font-size: 11px;
    padding: .2em 1em .275em;
}
/* 设计颜色样式类：黑色风格的按钮 */
/* 通过设计不同颜色的样式类，可以设计不同风格的按钮效果 */
.black { /* 黑色样式类 */
    color: #d7d7d7;
    border: solid 1px #333;
    background: #333;
    background: -webkit-gradient(linear, left top, left bottom, from(#666), to(#000));
    background: -moz-linear-gradient(top, #666, #000);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#666666',
endColorstr='#000000');
}
.black:hover { /* 黑色鼠标经过样式类 */
    background: #000;
    background: -webkit-gradient(linear, left top, left bottom, from(#444), to(#000));
    background: -moz-linear-gradient(top, #444, #000);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#444444',
endColorstr='#000000');
}
.black:active { /* 黑色鼠标激活样式类 */
    color: #666;
    background: -webkit-gradient(linear, left top, left bottom, from(#000), to(#444));
    background: -moz-linear-gradient(top, #000, #444);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#000000',
endColorstr='#666666');
}
</style>
</head>
<body>
    <div> <a href="#" class="button black">Rectangle</a> <a href="#" class="button black
bigrounded">Rounded</a> <a href="#" class="button black medium">Medium</a> <a href="#">
class="button black small">Small</a> <br />
    <br />
    <input class="button black" type="button" value="Input Element" />
    <button class="button black">Button Tag</button>
    <span class="button black">Span</span>
    <div class="button black">Div</div>
    <p class="button black">P Tag</p>
```

```
<h3 class="button black">H3</h3>
</div>
</body>
</html>
```

在上面的示例中，通过修改颜色样式类，可以设计不同色彩风格的按钮效果。例如，要设计品红色按钮效果，可以新定义 3 个品红色样式类，然后将它们应用到按钮中。



```
<style type="text/css">
/* 公共样式省略 */
/* pink */
.pink {
    color: #feeef5;
    border: solid 1px #d2729e;
    background: #f895c2;
    background: -webkit-gradient(linear, left top, left bottom, from(#feb1d3), to(#f171ab));
    background: -moz-linear-gradient(top, #feb1d3, #f171ab);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#feb1d3',
endColorstr='#f171ab');
}
.pink:hover {
    background: #d57ea5;
    background: -webkit-gradient(linear, left top, left bottom, from(#f4aacb), to(#e86ca4));
    background: -moz-linear-gradient(top, #f4aacb, #e86ca4);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#f4aacb',
endColorstr='#e86ca4');
}
.pink:active {
    color: #f3c3d9;
    background: -webkit-gradient(linear, left top, left bottom, from(#f171ab), to(#feb1d3));
    background: -moz-linear-gradient(top, #f171ab, #feb1d3);
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#f171ab',
endColorstr='#feb1d3');
}
</style>

<div> <a href="#" class="button pink">Pink</a> <a href="#" class="button pink
bigrounded">Rounded</a> <a href="#" class="button pink medium">Medium</a> <a href="#" class="button pink small">Small</a> <br /><br />
<input class="button pink" type="button" value="Input Element" />
<button class="button pink">Button Tag</button>
<span class="button pink">Span</span>
<div class="button pink">Div</div>
```

```
<p class="button pink">P Tag</p>
<h3 class="button pink">H3</h3>
</div>
```



当然，设计师还可以设计其他色系的按钮效果，如下图所示。



第8章

CSS 3 动画设计

CSS 擅长呈现静态样式，网页设计师也习惯把它作为页面效果的设计工具。不过，CSS 3 将要改变设计师的这种思维定势，很多必须依赖 JavaScript 脚本的二维或三维动画，现在借助 CSS 3 也能够实现了。

网络是一个飞速发展的技术试验场：当我们需要圆角时，CSS 实现了它；当我们需要多重背景时，CSS 实现了它；当我们需要边框图时，CSS 又实现了它。需求在前，实现在后。CSS 技术正是在不断满足设计师的各种常规设计要求的过程中进行大胆技术革新的。所以需求从来不是问题，否则，我们可能仍然还在使用表格进行网页布局。这一切都说明 CSS 变得越来越强大，越来越无所不能。

CSS 3 所提供的动画功能，主要包括变形、转换和动画技术。下面我们将结合具体应用案例以及浏览器支持情况，详细讲解这些动画类型的实现和实践。

8.1 CSS 变形（Transformation）

2009 年 3 月，W3C 组织正式发布 3D 变形动画标准草案 (<http://www.w3.org/TR/css3-3d-transforms/>)。为了更容易普及，同年 12 月，W3C 在 3D 草案基础上又发布了 2D 变形动画标准草案 (<http://www.w3.org/TR/css3-2d-transforms/>)。两个草案的核心内容基本相似，但是针对的主体不同，一个是 3D 动画，另一个是 2D 动画。其中 3D 动画在 Windows 系统暂时还未实现，仅能够在 Mac 系统特定版本中呈现。浏览器兼容说明如下所示。

CSS 2D 变形：

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 3.2	✓ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0	✓ Safari 4.0	✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

CSS 3D 变形：

目前，仅能够在 Mac 系统下的 Safari 4.0+ 版本浏览器中获得支持。

8.1.1 变形基础——transform 属性

transform 属性实现了一些可用 SVG 实现的变形功能。它可用于内联（inline）元素和块级（block）元素。该属性可以旋转、缩放和移动元素。熟练使用 transform 属性可以控制文字的变形，这种纯 CSS 的方法，可以确保网页内的文字保持可选，这是 CSS 相对于使用图片（或背景图片）的一个巨大优势。transform 属性的基本语法如表 8.1 所示。

注意，由于 CSS 3D 变形还没有获得 Windows 系统下主流浏览器的支持，所以本书主要围绕 CSS 2D 变形进行讲解。

表8.1 transform属性的基本语法

语法项目	说明
值	none <transform-function> [<transform-function>]*
初始值	none
适用于	块元素和行内元素
可否继承	否
百分比	引用盒子的尺寸
媒介	视觉

取值简单说明：

<transform-functions>：设置变形函数，可以是一个或多个变形函数列表。transform-functions 函数包括 matrix()、translate()、scale()、scaleX()、scaleY()、rotate()、skewX()、skewY()、skew() 等。关于这些常用变形函数的功能，简单说明如下。

- matrix()：定义矩阵变换，即基于 X 和 Y 坐标重新定位元素的位置。
- translate()：移动元素对象，即基于 X 和 Y 坐标重新定位元素。
- scale()：缩放元素对象，可以使任意元素对象尺寸发生变化，取值包括正数和负数，以及小数。
- rotate()：旋转元素对象，取值为一个度数值。
- skew()：倾斜元素对象，取值为一个度数值。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-transform 私有属性，Mozilla Gecko 引擎支持 -moz-transform 私有属性，Presto 引擎支持 -o-transform 私有属性，IE 浏览器暂时不支持 transform 属性，也没

有定义支持 transform 属性的私有属性，不过可以采用 IE 滤镜来进行兼容。

实战体验：设计一个简单的鼠标动画

在下面的示例中，我们通过一个简单的鼠标动画体验 CSS 3 动画是如何工作的。在这个动画中，当鼠标经过图像时，div 元素会放大，并旋转渐隐显示蓝色背景，移开之后又恢复默认效果，如图 8.1 所示。

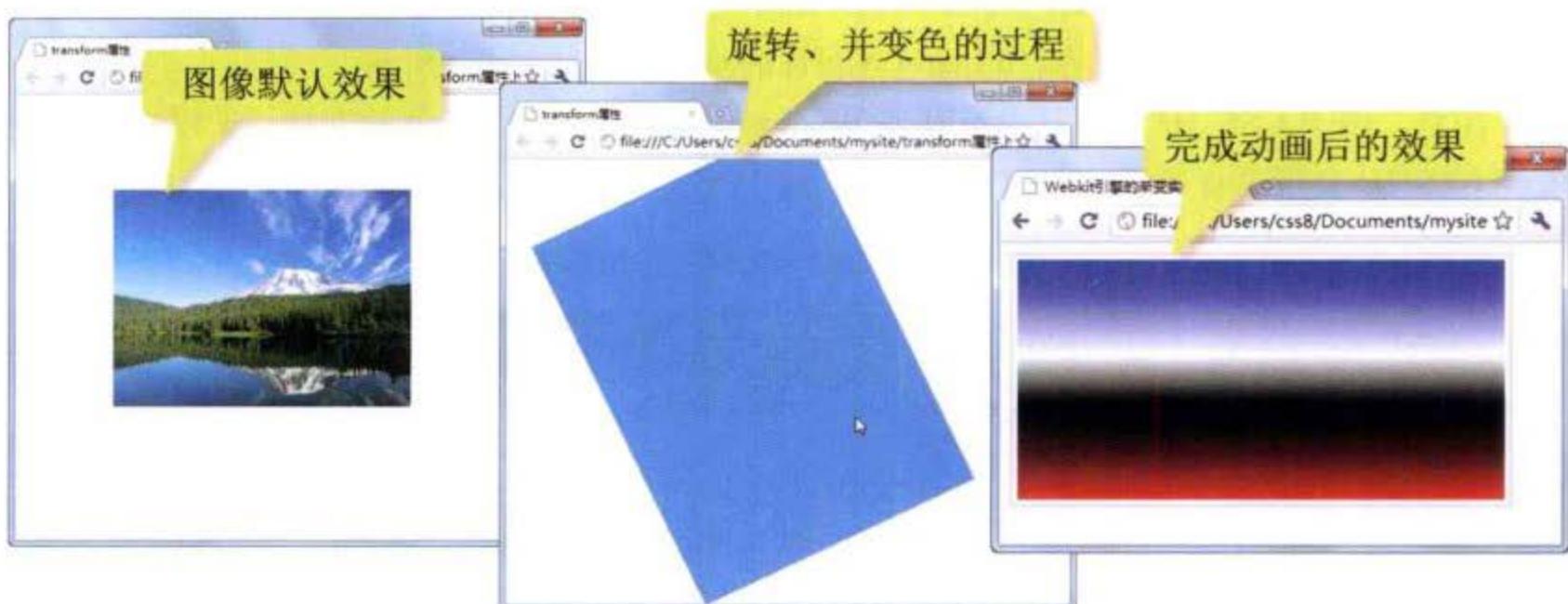


图 8.1 不断旋转并放大、变色的动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>transform 属性</title>
<style type="text/css">
div {
    position: absolute;
    left: 100px;
    top: 100px;
    width: 306px;
    height: 226px;
    background: #0FF url(images/bgl.jpg) no-repeat;
    -webkit-background-size: cover ;
    -o-background-size: cover ;
    background-size: cover ;
    /* 定义动画的过程 */
    -webkit-transition:-webkit-transform .5s ease-in, background .5s ease-in;
    -moz-transition:-moz-transform .5s ease-in, background .5s ease-in;
    -o-transition:-o-transform .5s ease-in, background .5s ease-in;
}
```

使用背景图像覆盖 div 元素
盒子

```

        transition:transform .5s ease-in, background .5s ease-in;
    }
    div:hover {
        /* 定义动画的状态 */
        -webkit-transform: rotate(180deg) scale(2);
        -moz-transform: rotate(180deg) scale(2);
        -o-transform: rotate(180deg) scale(2);
        transform: rotate(180deg) scale(2);
        background: blue;
    }

```

</style>

</head>

<body>

<div></div>

</body>

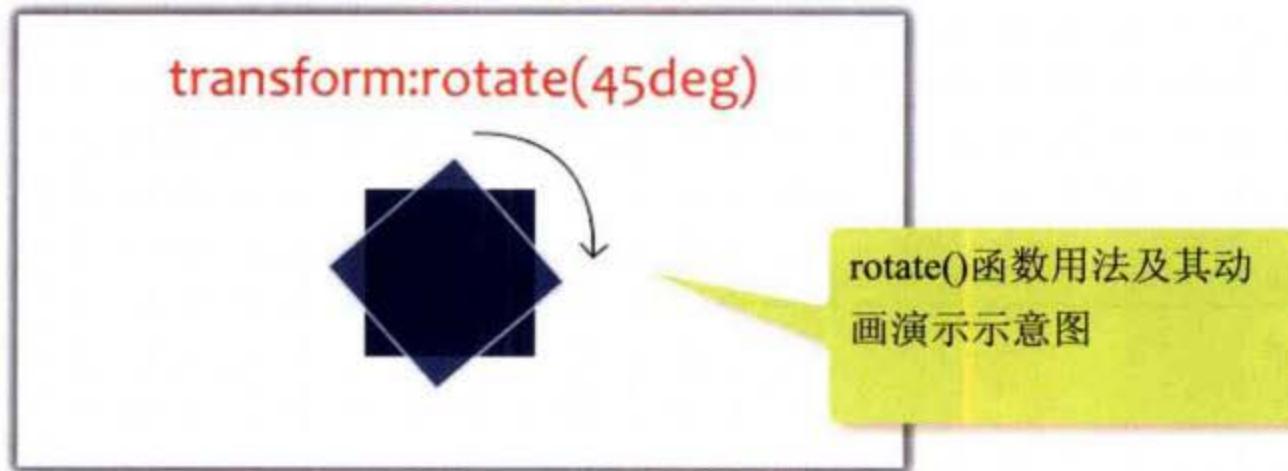
</html>

旋转并放大div
元素对象

8.1.2 旋转动画——rotate() 函数

rotate() 函数能够旋转指定的元素对象，它主要在二维空间内进行操作，接受一个角度参数值，用来指定旋转的幅度。元素对象可以是内联元素和块级元素。语法格式如下：

`rotate(<angle>)`



针对旋转动画，IE 浏览器虽然不支持，但是可以考虑使用图形旋转滤镜进行兼容。语法格式如下：

```
filter:progid:DXImageTransform.Microsoft.BasicImage(enabled=bEnabled, grayScale=bGray, mirror=bMirror, opacity=fOpacity, XRay=bXRay )
```

参数说明如下。

- ❑ **enabled**：可选项，布尔值。设置或检索滤镜是否激活。默认值 true 表示滤镜激活，取值 false 表示滤镜被禁止。
- ❑ **grayScale**：可选项，布尔值。设置或检索是否以灰度显示对象内容。取值 1 表示以灰度效果显示对象内容。0 为默认值，表示显示对象的原始色彩。

- ❑ mirror：可选项，布尔值。设置或检索是否反转显示对象内容。取值 1 表示反转显示对象内容。0 为默认值，表示正常显示对象内容。
- ❑ opacity：可选项，浮点数。设置对象内容的透明度。取值范围为 0.0~1.0，默认值为 1.0，表示不透明黑色，0.0 表示为完全透明。
- ❑ XRay：可选项，布尔值。设置或检索是否以 X 光效果显示对象内容。取值 1 表示以 X 光效果显示对象内容。0 为默认值，表示正常显示对象内容。

IE 图形旋转滤镜可以有 4 个旋转值：0、1、2 和 3。它无法像 Webkit 和 Gecko 引擎那样精确控制旋转的角度，不过可以在一定程度上保持兼容。

例如，下面的示例演示了如何逆时针旋转对象 90 度，如图 8.2 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>rotate() 函数</title>
<style type="text/css">
div {
  margin:100px 0;
  width: 400px;
  height: 100px;
  background:url(images/bg2.jpg) center;
}
div:hover {
  /* 定义动画的状态 */
  -webkit-transform: rotate(-90deg);
  -moz-transform: rotate(-90deg);
  -o-transform: rotate(-90deg);
  transform: rotate(-90deg);
  filter: progid:DXImageTransform.Microsoft.BasicImage(rotation=3);
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

设置div元素在鼠标经过时逆时针旋转90度



图 8.2 旋转动画效果

注意，IE 8版本的浏览器在怪异模式下不支持filter属性，应该使用IE私有属性进行定义，即-ms-filter，而不是filter。例如，在上面的示例中可以添加这样一个声明：-ms-filter: progid:DXImageTransform.Microsoft.BasicImage(rotation=3);。



8.1.3 缩放动画——scale() 函数

scale() 函数能够缩放元素。该函数包含两个参数值，分别用来定义宽和高缩放比例。语法格式如下：

```
scale(<number>[, <number>])
```

<number> 参数值可以是正数、负数和小数。正数值基于指定的宽度和高度放大元素。负数值不会缩小元素，而是翻转元素（如文字被反转），然后再缩放元素。使用小于 1 的小数（如 0.5）可以缩小元素。如果第二个参数省略，则第二个参数等于第一个参数值。



scale() 函数目前只有 Firefox、Safari/Chrome 以及 Opera 10.50 支持，到目前为止没有 IE 版本支持。缩放对象是相当有意义的功能，使用它可以渐进增强：hover 的可用性。例如，在导航菜单中添加缩放功能，让导航菜单更好用，如图 8.3 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>scale() 函数</title>
<style type="text/css">
.test ul { list-style:none; }
.test li {
    float:left;
    width:100px;
    background:#CCC;
    margin-left:3px;
    line-height:30px;
}
```

```

}
.test a {
    display:block;
    text-align:center;
    height:30px;
}
.test a:link {
    color:#666;
    background:url(images/icon1.jpg) #CCC no-repeat 5px 12px;
    text-decoration:none;
}
.test a:visited {
    color:#666;
    text-decoration:underline;
}
.test a:hover {
    color:#FFF;
    font-weight:bold;
    text-decoration:none;
    background:url(images/icon2.jpg) #F00 no-repeat 5px 12px;
    -webkit-transform: scale(2);
    -moz-transform: scale(2);
    -o-transform: scale(2);
}
</style>
</head>
<body>
<div class="test">
    <ul>
        <li><a href="1">首页</a></li>
        <li><a href="2">产品介绍</a></li>
        <li><a href="3">服务介绍</a></li>
        <li><a href="4">技术支持</a></li>
        <li><a href="5">立刻购买</a></li>
        <li><a href="6">联系我们</a></li>
    </ul>
</div>
</body>
</html>

```

设置a元素在鼠标经过时放大2倍进行显示



图 8.3 缩放动画效果

注意，当为scale()函数传递不同的参数值时，缩放动画效果是不同的。例如，下面几个案例分别演示了改变scale()函数的参数值时不同的动画效果。



```
.test a:hover {
    /* a元素的宽度保持不变，而高度显示为原来的2倍 */
    -webkit-transform: scale(1, 2);
    -moz-transform: scale(1, 2);
    -o-transform: scale(1, 2);
}
```

鼠标经过时，长宽不等比放大的效果



```
.test a:hover {
    /* a元素的宽度保持不变，而高度显示为原来的双倍，并且水平翻转 */
    -webkit-transform: scale(1, 2);
    -moz-transform: scale(1, 2);
    -o-transform: scale(1, 2);
}
```

鼠标经过时，长宽不等比放大，然后旋转180度后的效果

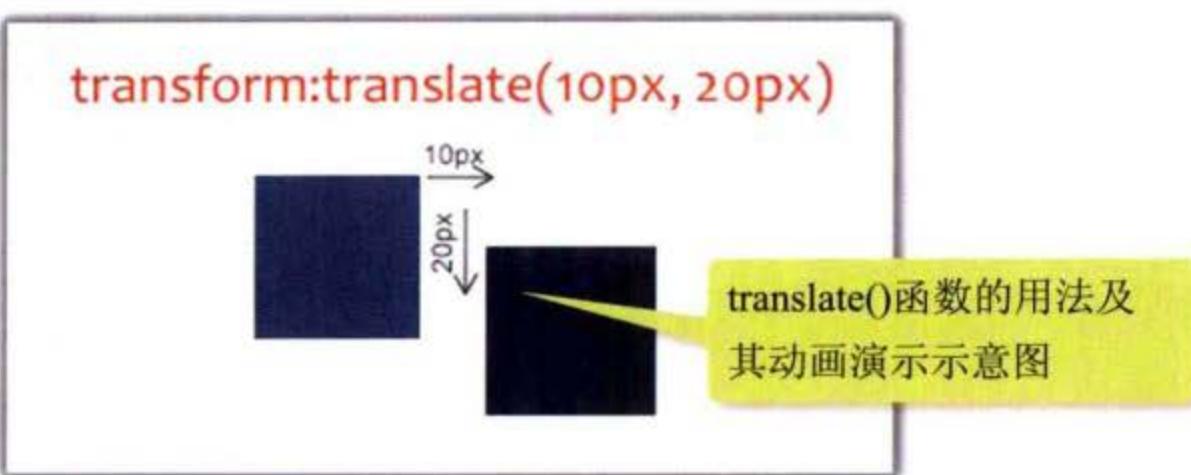


8.1.4 移动动画——translate() 函数

translate() 函数能够重新定位元素的坐标，该函数包含两个参数值，分别用来定义 x 轴和 y 轴坐标。语法格式如下：

```
translate(<translation-value>[, <translation-value>])
```

<translation-value> 参数表示坐标值，第一个参数表示相对于原位置的 x 轴偏移距离，第二个参数表示相对于原位置的 y 轴偏移距离，如果省略了第二个参数，则第二个参数值默认为 0。



translate() 目前只有 Firefox、Safari/Chrome 以及 Opera 10.50 支持，到目前为止没有 IE 版本支持。移动对象是相当有意义的功能，使用它可以渐进增强：hover 的可用性。例如，在导航菜单中添加定位功能，让导航菜单更富动感，如图 8.4 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>translate() 函数</title>
<style type="text/css">
.test ul { list-style:none; }
.test li {
    float:left;
    width:100px;
    background:#CCC;
    margin-left:3px;
    line-height:30px;
}
.test a {
    display:block;
    text-align:center;
    height:30px;
}
.test a:link {
    color:#666;
    background:url(images/icon1.jpg) #CCC no-repeat 5px 12px;
    text-decoration:none;
}
.test a:visited {
    color:#666;
    text-decoration:underline;
}
.test a:hover {
    color:#FFF;
    font-weight:bold;
    text-decoration:none;
}

```

```

background:url(images/icon2.jpg) #F00 no-repeat 5px 12px;
-moz-transform: translate(4px, 4px);
-webkit-transform: translate(4px, 4px);
-o-transform: translate(4px, 4px);
}

</style>
</head>
<body>
<div class="test">
<ul>
    <li><a href="1">首页</a></li>
    <li><a href="2">产品介绍</a></li>
    <li><a href="3">服务介绍</a></li>
    <li><a href="4">技术支持</a></li>
    <li><a href="5">立刻购买</a></li>
    <li><a href="6">联系我们</a></li>
</ul>
</div>
</body>
</html>

```

设置a元素在鼠标经过时向右下角位置偏移4个像素



图8.4 偏移动画效果

注意，当为translate()函数传递一个参数值时，则表示水平偏移，如果垂直偏移，则应设置第一个参数值为0，第二个参数值为垂直偏移值。如果设置负数，则表示反向偏移，但是参考距离不同。

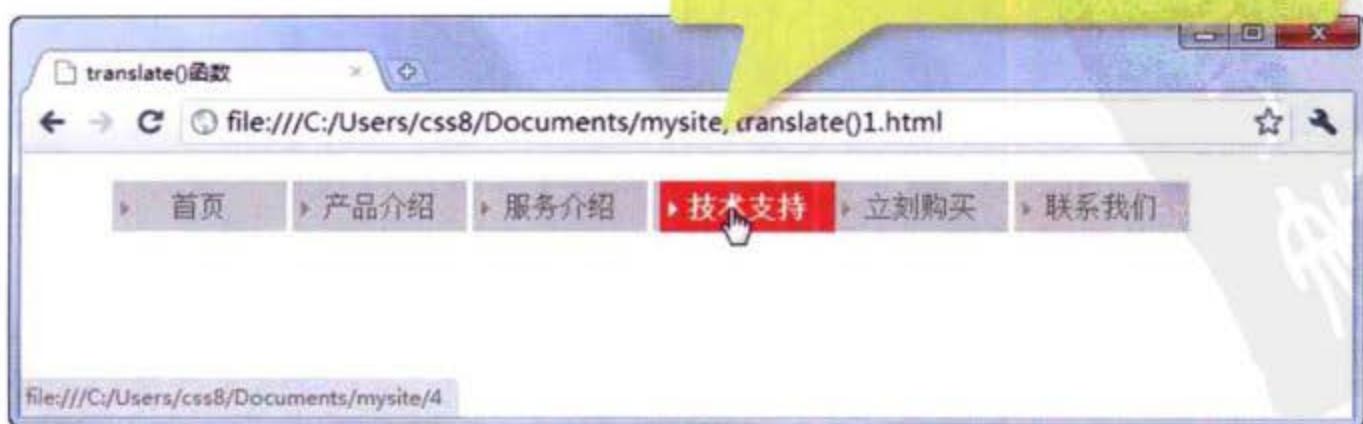


```

.test a:hover {
    /* a元素水平右移4个像素 */
    -moz-transform: translate(4px);
    -webkit-transform: translate(4px);
    -o-transform: translate(4px);
}

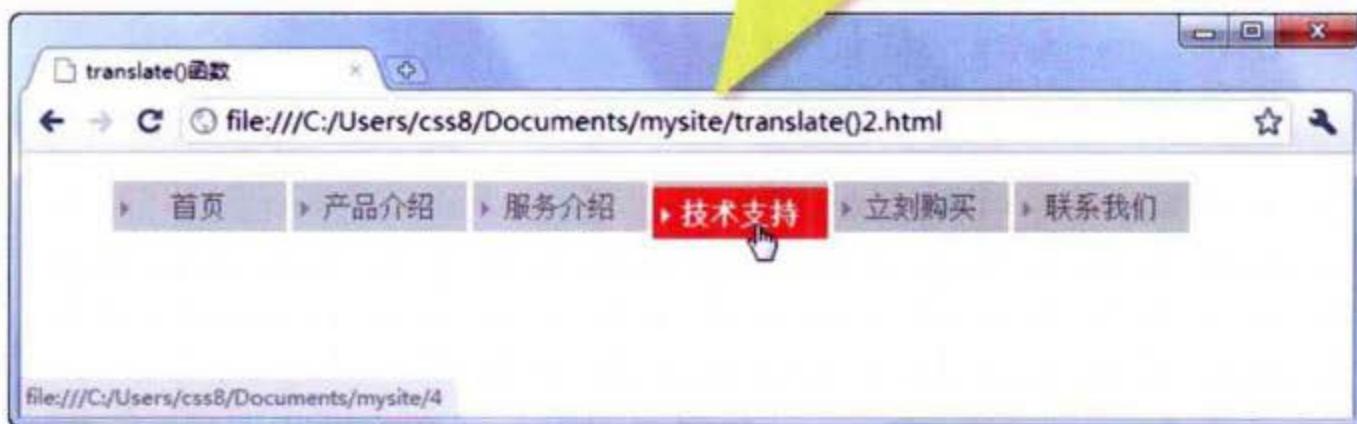
```

鼠标经过时，向右偏移4个像素



```
.test a:hover {  
    /* a元素垂直下移4个像素 */  
    -moz-transform: translate(0, 4px);  
    -webkit-transform: translate(0, 4px);  
    -o-transform: translate(0, 4px);  
}
```

鼠标经过时，向下偏移4个像素



```
.test a:hover {  
    /* a元素向左上角偏移4个像素 */  
    -moz-transform: translate(-4px, -4px);  
    -webkit-transform: translate(-4px, -4px);  
    -o-transform: translate(-4px, -4px);  
}
```

鼠标经过时，向左上偏移4个像素



8.1.5 倾斜动画——skew() 函数

skew() 函数能够让元素倾斜显示，该函数包含两个参数值，分别用来定义 x 轴和 y 轴坐标倾斜的角度。语法格式如下：

```
skew(<angle> [, <angle>])
```

<angle> 参数表示角度值，第一个参数表示相对于 x 轴进行倾斜，第二个参数表示相对于 y 轴进行倾斜，如果省略了第二个参数，则第二个参数值默认为 0。

skew() 也是一个很有用的变形函数，它可以将一个对象围绕着 x 和 y 轴按照一定的角度倾斜。这与 rotate() 函数的旋转不同，rotate() 函数只是旋转，而不会改变元素的形状。skew()

函数会改变元素的形状。例如，在导航菜单中添加倾斜变形功能，让导航菜单更富情趣，如图 8.5 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>translate()函数</title>
<style type="text/css">
.test ul { list-style:none; }
.test li {
    float:left;
    width:100px;
    background:#CCC;
    margin-left:3px;
    line-height:30px;
}
.test a {
    display:block;
    text-align:center;
    height:30px;
}
.test a:link {
    color:#666;
    background:url(images/icon1.jpg) #CCC no-repeat 5px 12px;
    text-decoration:none;
}
.test a:visited {
    color:#666;
    text-decoration:underline;
}
.test a:hover {
    color:#FFF;
    font-weight:bold;
    text-decoration:none;
    background:url(images/icon2.jpg) #F00 no-repeat 5px 12px;
    -moz-transform: skew(30deg, -10deg);
    -webkit-transform: skew(30deg, -10deg);
    -o-transform: skew(30deg, -10deg);
}
</style>
</head>
<body>
<div class="test">
    <ul>
        <li><a href="1">首页</a></li>
        <li><a href="2">产品介绍</a></li>
        <li><a href="3">服务介绍</a></li>
        <li><a href="4">技术支持</a></li>
        <li><a href="5">立刻购买</a></li>
        <li><a href="6">联系我们</a></li>
    </ul>
</div>
</body>
```

设置a元素在鼠标经过时向左
下角位置倾斜

```
</html>
```



图 8.5 倾斜动画效果

8.1.6 矩阵变形动画——matrix() 函数

matrix() 是矩阵函数，矩阵是高等数学中的一个基本概念，CSS 3 引入了 matrix() 函数，可以非常灵活地实现各种变形效果。该函数包括 6 个参数 (a,b,c,d,e,f)，实际上 matrix() 函数是一个 3×3 矩阵，经过该矩阵将旧的参数转换成新值。

```
| a b e |
| c d f |
| 0 0 1 |
```

如果读者想深入研究该函数的矩阵方程，可以参考 <http://www.w3.org/TR/SVG/coords.html#TransformMatrixDefined>，这是 SVG 的一个文档，但是对于 matrix() 函数的变换原理是通用的。matrix() 函数的语法格式如下。

```
matrix(<number>, <number>, <number>, <number>, <number>, <number>)
```

虽然 IE 不支持 CSS 3 的大部分变形功能，但是它支持 matrix 滤镜，使用这个滤镜基本上也可以实现相同的效果，不过读者需要理解矩阵运算原理。

例如，在导航菜单中利用 matrix() 函数的矩阵变形设计特殊变形效果，如图 8.6 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>matrix() 函数</title>
<style type="text/css">
.test ul { list-style:none; }
.test li {
    float:left;
    width:100px;
    background:#CCC;
    margin-left:3px;
    line-height:30px;
}
```

```

}
.test a {
    display:block;
    text-align:center;
    height:30px;
}
.test a:link {
    color:#666;
    background:url(images/icon1.jpg) #CCC no-repeat 5px 12px;
    text-decoration:none;
}
.test a:visited {
    color:#666;
    text-decoration:underline;
}
.test a:hover {
    color:#FFF;
    font-weight:bold;
    text-decoration:none;
    background:url(images/icon2.jpg) #F00 no-repeat 5px 12px;
    -moz-transform: matrix(1, 0.4, 0, 1, 0, 0);
    -webkit-transform: matrix(1, 0.4, 0, 1, 0, 0);
    -o-transform: matrix(1, 0.4, 0, 1, 0, 0);
}
</style>
</head>

<body>
<div class="test">
    <ul>
        <li><a href="1">首页</a></li>
        <li><a href="2">产品介绍</a></li>
        <li><a href="3">服务介绍</a></li>
        <li><a href="4">技术支持</a></li>
        <li><a href="5">立刻购买</a></li>
        <li><a href="6">联系我们</a></li>
    </ul>
</div>
</body>
</html>

```

设置a元素在鼠标经过时矩阵变形

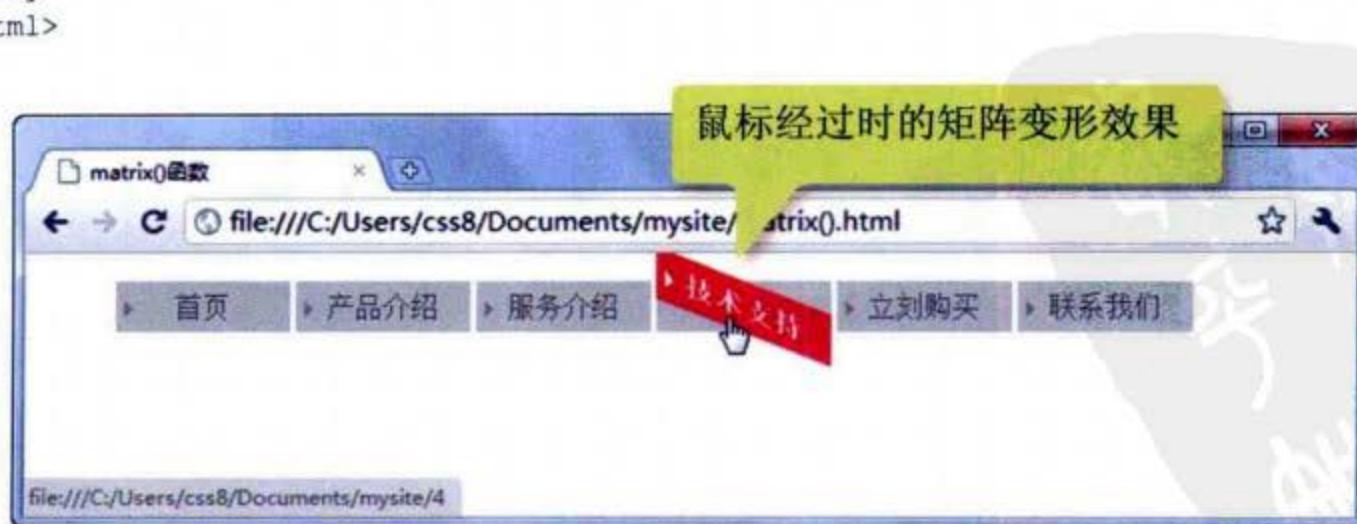
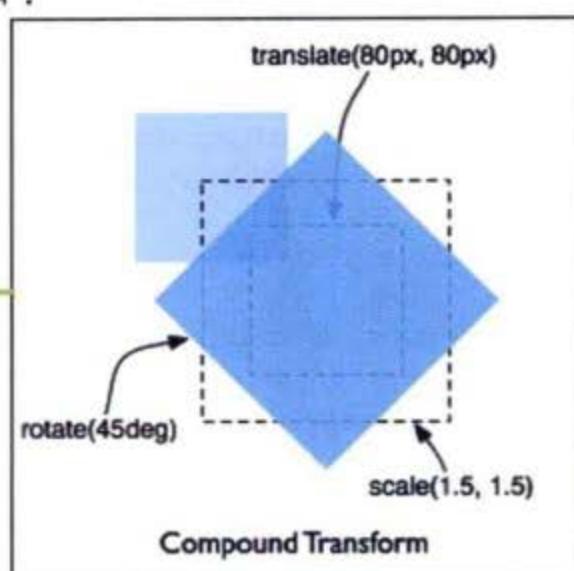


图8.6 矩阵动画效果

8.1.7 CSS 3 实战体验：设计图片墙

在实战中，我们可能不会单独使用某种变形操作，而是同时使用多种变形，以设计复杂的变形效果。为了方便代码编写，CSS 3 支持缩写形式。例如：

```
div {
    -moz-transform: translate(80, 80);
    -webkit-transform: translate(80, 80);
    -o-transform: translate(80, 80);
    -moz-transform: rotate(45deg);
    -webkit-transform: rotate(45deg);
    -o-transform: rotate(45deg);
    -moz-transform: scale(1.5, 1.5);
    -webkit-transform: scale(1.5, 1.5);
    -o-transform: scale(1.5, 1.5);
}
```



对于上面的样式，我们可以缩写为：

```
div {
    -moz-transform: translate(80, 80) rotate(45deg) scale(1.5, 1.5);
    -webkit-transform: translate(80, 80) rotate(45deg) scale(1.5, 1.5);
    -o-transform: translate(80, 80) rotate(45deg) scale(1.5, 1.5);
}
```

综合使用移动、旋转、缩放等函数来控制元素，而不使用 JavaScript，就可以创建更丰富和更轻量级的界面和应用。下面我们来看一个综合案例，在这个综合案例中，我们将用到 CSS 3 阴影、透明效果以及变形动画，让图片随意贴在墙上，当鼠标移动到图片上时，会自动放大并垂直摆放，效果如图 8.7 所示。





图8.7 图片墙效果

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3变形综合应用</title>
<style type="text/css">
body { background:#fff url(images/wood-bg.jpg); }
ul.polaroids li { display: inline; }
ul.polaroids a {
    background: #fff;
    display: inline;
    float: left;
    margin: 0 0 27px 30px;
    width: auto;
    padding: 10px 10px 15px;
    text-align: center;
    font-family: "Marker Felt", sans-serif;
    text-decoration: none;
    color: #333;
    font-size: 18px;
    -webkit-box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
    -moz-box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
    -webkit-transition: -webkit-transform .15s linear;
    -webkit-transform: rotate(-2deg);
    -moz-transform: rotate(-2deg);
}

```

为图片外框设计阴影效果

顺时针旋转2度

```
ul.polaroids img {  
    display: block;  
    height: 100px;  
    margin-bottom: 12px;  
}  
/* 获取a元素的title属性值，并把它作为内容插入到a元素后面 */  
ul.polaroids a:after { content: attr(title); }  
ul.polaroids li:nth-child(even) a {  
    -webkit-transform: rotate(5deg);  
    -moz-transform: rotate(5deg);  
}  
ul.polaroids li:nth-child(3n) a {  
    position: relative;  
    top: -5px;  
    -webkit-transform: none;  
    -moz-transform: none;  
}  
ul.polaroids li:nth-child(5n) a {  
    position: relative;  
    right: 5px;  
    -webkit-transform: rotate(-10deg);  
    -moz-transform: rotate(-10deg);  
}  
ul.polaroids li:nth-child(8n) a {  
    position: relative;  
    right: 5px;  
    top: 8px;  
}  
ul.polaroids li:nth-child(11n) a {  
    position: relative;  
    left: -5px;  
    top: 3px;  
}  
ul.polaroids li a:hover {  
    -webkit-transform: scale(1.25);  
    -moz-transform: scale(1.25);  
    -webkit-box-shadow: 0 3px 6px rgba(0, 0, 0, .5);  
    -moz-box-shadow: 0 3px 6px rgba(0, 0, 0, .5);  
    position: relative;  
    z-index: 5;  
}  
</style>  
</head>  
<body>  
<ul class="polaroids">  
    <li> <a href="1" title="风景1">  </a> </li>  
    <li> <a href="2" title="风景2">  </a> </li>  
    <li> <a href="3" title="风景3">  </a> </li>  
    <li> <a href="4" title="风景4">  </a> </li>  
    <li> <a href="5" title="风景5">  </a> </li>  
    <li> <a href="6" title="风景6">  </a> </li>  
    <li> <a href="7" title="风景7">  </a> </li>  
    <li> <a href="8" title="风景8">  </a> </li>  
    <li> <a href="9" title="风景9">  </a> </li>  
    <li> <a href="10" title="风景10">  </a> </li>
```

逆时针旋转5度

不选择对象

顺时针旋转5度

放大对象1.25倍

```

<li> <a href="11" title="风景11"> </a> </li>
<li> <a href="12" title="风景12">  </a> </li>
</ul>
</body>
</html>

```

8.2 CSS 变形原点——transform-origin 属性

CSS 变形的原点默认为对象的中心点，如果要改变这个中心点，可以使用 transform-origin 属性进行定义。例如，rotate 变形的默认原点是对象的中心点，使用 transform-origin 属性可以将原点设置在对象左上角，或者左下角，这样 rotate 变形的结果就不相同了。transform-origin 属性的基本语法如表 8.2 所示。

表8.2 transform-origin属性的基本语法

语法项目	说明
值	[[<percentage> <length> left center right] [<percentage> <length> top center bottom]?] [[left center right] [top center bottom]]
初始值	50% 50%
适用于	块状元素和内联元素
可否继承	否
百分比	根据元素盒子的尺寸
媒介	所有

取值简单说明：

transform-origin 接受两个参数，它们可以是百分比、em、px 等具体的值，也可以是 left、center、right，或者 top、middle、bottom 等描述性关键字。

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-transform-origin 私有属性，Mozilla Gecko 引擎支持 -moz-transform-origin 私有属性，Presto 引擎支持 -o-transform-origin 私有属性，IE 浏览器暂时不支持 transform-origin 属性，也没有定义支持 transform-origin 属性的私有属性。

CSS3 实战体验：定义图片旋转的原点

通过改变变形对象的原点，可以实现不同的变形效果。例如，在下面的示例中我们分别

以图像的四个角点为原点来旋转图像，则效果如图 8.8 所示。



图 8.8 定义变形对象操作的原点

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3 变形综合应用</title>
<style type="text/css">
body { background:#fff url(images/wood-bg.jpg); }
ul.polaroids li { display: inline; }
ul.polaroids a {
    background: #fff;
    float: left;
    margin: 0 0 27px 30px;
}
```

```

width: auto;
padding: 10px 10px 15px;
text-align: center;
text-decoration: none;
color: #333;
-webkit-box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
-moz-box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
}

ul.polaroids img {
    display: block;
    height: 180px;
    margin-bottom: 12px;
}

ul.polaroids a:after { content: attr(title); }

ul.polaroids li:nth-child(1) a /* 变形第1个对象 */
    -moz-transform-origin: 0 0;
    -webkit-transform-origin: 0 0;
    -o-transform-origin: 0 0;
    -webkit-transform: rotate(30deg);
    -moz-transform: rotate(30deg);
    -o-transform: rotate(30deg);

}

ul.polaroids li:nth-child(2) a /* 变形第2个对象 */
    -moz-transform-origin: top right;
    -webkit-transform-origin: top right;
    -o-transform-origin: top right;
    -webkit-transform: rotate(-30deg);
    -moz-transform: rotate(-30deg);
    -o-transform: rotate(-30deg);

}

ul.polaroids li:nth-child(3) a /* 变形第3个对象 */
    -moz-transform-origin: 0 100%;
    -webkit-transform-origin: 0 100%;
    -o-transform-origin: 0 100%;
    -webkit-transform: rotate(60deg);
    -moz-transform: rotate(60deg);
    -o-transform: rotate(60deg);

}

ul.polaroids li:nth-child(4) a /* 变形第4个对象 */
    -moz-transform-origin: 100% 100%;
    -webkit-transform-origin: 100% 100%;
    -o-transform-origin: 100% 100%;
    -webkit-transform: rotate(60deg);
    -moz-transform: rotate(60deg);
    -o-transform: rotate(60deg);

}

</style>
</head>
<body>
<ul class="polaroids">
    <li> <a href="1" title="左上角为原点">  </a> </li>
    <li> <a href="2" title="右上角为原点">  </a> </li>
    <li> <a href="3" title="左下角为原点">  </a> </li>
    <li> <a href="4" title="右下角为原点">  </a> </li>
</ul>
</body>
</html>

```

8.3 CSS 过渡——transition 属性

CSS Transformation（转换）呈现的是一种变形结果，而 CSS Transition（过渡）呈现的是一种过渡，通俗点说就是一种动画转换过程，如渐显、渐弱、动画快慢等。CSS Transformation 和 CSS Transition 是两种不同的动画模型，因此 W3C 单独为动画过渡定义了单独的模块，详细信息请参阅 <http://www.w3.org/TR/css3-transitions/> 页面。

过渡可以与变形同时使用。例如，触发 :hover 或者 :focus 事件后创建动画过程，如淡出背景色、滑动一个元素以及让一个对象旋转都可以通过 CSS 转换实现。

transition 属性是一个复合属性，可以同时定义 transition-property、transition-duration、transition-timing-function、transition-delay、transition-property、transition-duration、transition-timing-function、transition-delay 子属性值。transition 属性的基本语法如表 8.3 所示。

表8.3 transition属性的基本语法

语法项目	说明
值	[< 'transition-property' > < 'transition-duration' > < 'transition-timing-function' > < 'transition-delay' > [, [< 'transition-property' > < 'transition-duration' > < 'transition-timing-function' > < 'transition-delay' >]]]*
初始值	根据各个子属性的默认值而定
适用于	所有元素，以及 :before 和 :after 伪元素
可否继承	否
百分比	N/A
媒介	交互性媒体

浏览器兼容性检测

目前 Webkit 引擎支持 -webkit-transition 私有属性，Mozilla Gecko 引擎支持 -moz-transition 私有属性，Presto 引擎支持 -o-transition 私有属性，IE 浏览器暂时不支持 transition 属性，也没有定义支持 transition 属性的私有属性。各主流浏览器对 transition 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 3.0	✗ Opera 9.6	✗ Safari 3.2	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.5	✗ Opera 10.0	✗ Safari 4.0	✗ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.7	✓ Opera 10.5		✓ Chrome 4.0.x

8.3.1 设置过渡的 CSS 属性——transition-property 属性

transition-property 属性用来定义转换动画的 CSS 属性名称，如 background-color 属性。该属性的基本语法如表 8.4 所示。

表8.4 transition-property属性的基本语法

语法项目	说明
值	none all [<IDENT>] [',' <IDENT>]*
初始值	all
适用于	所有元素，以及 :before 和 :after 伪元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

- none：表示没有元素。
- all：表示针对所有元素。
- IDENT：指定 CSS 属性列表。

浏览器兼容性请参阅 transition 属性。

例如，在下面的示例中，指定动画的属性为背景颜色。这样当鼠标经过 div 对象时，会自动从红色背景过渡到蓝色背景。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>transition-property属性</title>
<style type="text/css">
div {
    background-color:red;
    width:400px;
    height:150px;
}
```

```

    height:200px;
}
div:hover {
    background-color:blue;
    -webkit-transition-property:background-color;
    -moz-transition-property:background-color;
    -o-transition-property:background-color;
}
</style>
</head>
<body>
<div></div>
</body>
</html>

```

指定动画过渡的CSS属性

8.3.2 设置过渡的时间——transition-duration 属性

transition-duration 属性用来定义转换动画的时间长度，即从旧属性换到新属性花费的时间，单位为秒。该属性的基本语法如表 8.5 所示。

表 8.5 transition-duration 属性的基本语法

语法项目	说明
值	<time> [, <time>]*
初始值	0
适用于	所有元素，以及 :before 和 :after 伪元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

在默认情况下，动画过渡时间为 0 秒，所以当我们指定元素动画时，会看不到过渡的过程，直接看到结果。

例如，在上面的示例中，我们设置动画过渡时间为 2 秒，则当鼠标移过 div 对象时，你会看到背景色从红色逐渐过渡到蓝色，如图 8.9 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>transition-property 属性</title>
<style type="text/css">
div {
    background-color:red;
    width:400px;
    height:200px;
}
div:hover {
    background-color:blue;
    -webkit-transition-property:background-color;
    -moz-transition-property:background-color;
    -o-transition-property:background-color;
}
</style>
</head>
<body>
<div></div>
</body>
</html>

```

```

}
div:hover {
    background-color:blue;
    -webkit-transition-property:background-color;
    -moz-transition-property:background-color;
    -o-transition-property:background-color;
    -webkit-transition-duration:2s;
    -moz-transition-duration:2s;
    -o-transition-duration:2s;
}
</style>
</head>
<body>
<div></div>
</body>
</html>

```

对象默认背景色

动画过渡过程

最后显示的背景色

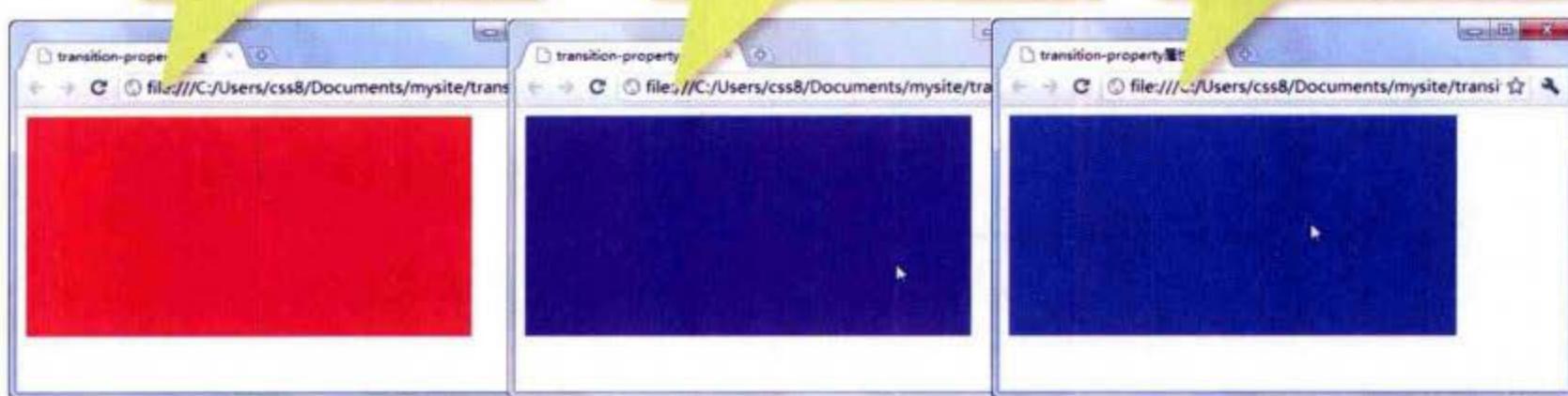


图8.9 背景色过渡的动画效果

8.3.3 设置过渡延迟时间——transition-delay 属性

transition-delay 属性用来定义过渡动画的延迟时间。该属性的基本语法如表 8.6 所示。

表8.6 transition-delay属性的基本语法

语法项目	说明
值	<time> [, <time>]*
初始值	0
适用于	所有元素，以及:before和:after伪元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

时间可以为正整数、负整数和零，非零的时候必须设置单位为 s（秒）或者 ms（毫秒）；为负数的时候，过渡的动作会从该时间点开始显示，之前的动作被截断；为正数的时候，过渡的动作会延迟触发。

例如，在上面的示例中，我们设置过渡动画推迟 2 秒钟执行，则当鼠标移过 div 对象时，会看不到任何变化，过了 2 秒钟之后，你会看到背景色从红色逐渐过渡到蓝色。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>transition-property 属性</title>
<style type="text/css">
div {
    background-color:red;
    width:400px;
    height:200px;
}
div:hover {
    background-color:blue;
    -webkit-transition-property:background-color;
    -moz-transition-property:background-color;
    -o-transition-property:background-color;
    -webkit-transition-duration:2s;
    -moz-transition-duration:2s;
    -o-transition-duration:2s;
    -webkit-transition-delay:2s;
    -moz-transition-delay:2s;
    -o-transition-delay:2s;
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

8.3.4 设置过渡效果——transition-timing-function 属性

transition-timing-function 属性用来定义过渡动画的效果。该属性的基本语法如表 8.7 所示。

表 8.7 transition-timing-function 属性的基本语法

语法项目	说明
值	ease linear ease-in ease-out ease-in-out cubic-bezier(<number>, <number>, <number>, <number>) [, ease linear ease-in ease-out ease-in-out cubic-bezier(<number>, <number>, <number>, <number>)]*
初始值	ease

(续)

语法项目	说明
适用于	所有元素，以及 :before 和 :after 伪元素
可否继承	否
百分比	N/A
媒介	交互性媒体

取值简单说明：

- ease：缓解效果，等同于 cubic-bezier(0.25, 0.1, 0.25, 1.0) 函数，即立方贝塞尔。
- linear：线性效果，等同于 cubic-bezier(0.0, 0.0, 1.0, 1.0) 函数。
- ease-in：渐显效果，等同于 cubic-bezier(0.42, 0, 1.0, 1.0) 函数。
- ease-out：渐隐效果，等同于 cubic-bezier(0, 0, 0.58, 1.0) 函数。
- ease-in-out：渐显渐隐效果，等同于 cubic-bezier(0.42, 0, 0.58, 1.0) 函数。
- cubic-bezier：特殊的立方贝塞尔曲线效果。

例如，为了让动画渐变过程更加富有立体感，可以在前面示例的基础上设置过渡效果为线性效果，代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>transition-timing-function函数</title>
<style type="text/css">
div {
    background-color:red;
    width:400px;
    height:200px;
}
div:hover {
    background-color:blue;
    -webkit-transition-property:background-color;
    -moz-transition-property:background-color;
    -o-transition-property:background-color;
    -webkit-transition-duration:5s;
    -moz-transition-duration:5s;
    -o-transition-duration:5s;
    -webkit-transition-timing-function: linear;
    -moz-transition-timing-function: linear;
    -o-transition-timing-function: linear;
}
</style>
</head>
<body>
<div></div>
</body>
```

指定动画过渡的CSS属性

指定动画过渡的时间

指定动画过渡为线性效果

</html>

CSS 3 实战体验：设计 OS X Dock (OS 系统的导航码头)

在下面这个实战案例中，我们将借助过渡动画来模拟 OS 系统桌面底部的导航码头，它类似于 Windows 桌面上的快捷图标，方便用户快速启动系统内的软件。演示效果如图 8.10 所示。



图 8.10 背景色过渡的动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3过渡综合应用</title>
<style type="text/css">
body { /* 设置页眉背景 */
    background: #030611 url(images/maelstrom.png) center -100px no-repeat;
    padding: 0;
    margin: 0;
}
ul.osx-dock { /* 定义外包含框的样式 */
    display: inline-block;
    height: 130px;
    padding: 0 40px 0 0;
    overflow: hidden;
    margin: 0 0 0 40px;
}
ul.osx-dock li { /* 定义每个图标的外框样式 */
    position: absolute;
    width: 40px;
    height: 40px;
    border-radius: 50%;
```

```

display: block;
position: relative;
float: left;
width: 50px;
height: 50px;
margin: 60px 0 4px 0;
-webkit-transition: 0.15s linear;
-webkit-transition-property: -webkit-transform margin;
text-align: center;
}

ul.osx-dock li a {
    display: block;
    height: 50px;
    padding: 0 1px;
    -webkit-transition: 0.15s linear;
    -webkit-transition-property: -webkit-transform margin;
    margin: 0;
/* 设计倒影效果 */
    -webkit-box-reflect: below 2px -webkit-gradient(linear, left top, left bottom,
from(transparent), color-stop(0.45, transparent), to(rgb(255, 255, 255, 0.25)));
}
ul.osx-dock li a img { width: 48px; }
ul.osx-dock li:hover {
    margin-left: 9px;
    margin-right: 9px;
    z-index: 200;
}
ul.osx-dock li:hover a {
    -webkit-transform-origin: center bottom;
    -webkit-transform: scale(1.5);
}
ul.osx-dock li span {
    background: rgba(0, 0, 0, 0.75);
    position: absolute;
    bottom: 80px;
    margin: 0 auto;
    display: none;
    width: auto;
    font-size: 11px;
    font-weight: bold;
    padding: 3px 6px;
/* 设计圆角 */
    -webkit-border-radius: 6px;
    color: #fff;
}
ul.osx-dock li:hover span { display: block; }
div#dockContainer {
    position: fixed;
    bottom: 12px;
    height: 120px;
    padding: 50px 0 0;
    text-align: center;
/* 设计圆角 */
    -webkit-border-radius: 6px;
    -moz-border-radius: 6px;
}

```

设计线性过渡，过渡属性为-webkit-transform和margin属性

设置对象底部中心点为原点，向上放大对象1.5倍

```

width: 100%;
line-height: 1;
z-index: 100;
}
div#dockWrapper { /* 定义外包含框的样式 */
width: auto;
display: inline-block;
position: relative;
border-bottom: solid 2px rgba(255, 255, 255, .35);
line-height: 0;
}
</style>
</head>
<body>
<div id="dockContainer">
<div id="dockWrapper">
<ul class="osx-dock">
<li class="active"> <span>ZURB</span> <a href="#" title="ZURB"></a> </li>
<li> <span>LinkedIn</span> <a href="#" title="LinkedIn"></a> </li>
<li> <span>Notable</span> <a href="#" title="Notable"></a> </li>
<li> <span>last.fm</span> <a href="#" title="Last.fm"></a> </li>
<li> <span>Facebook</span> <a href="#" title="Facebook"></a> </li>
<li> <span>Google</span> <a href="#" title="Google"></a> </li>
<li> <span>Notable</span> <a href="#" title="Notable"></a> </li>
<li> <span>last.fm</span> <a href="#" title="Last.fm"></a> </li>
<li> <span>Facebook</span> <a href="#" title="Facebook"></a> </li>
<li> <span>Google</span> <a href="#" title="Google"></a> </li>
</ul>
</div>
</div>
</body>
</html>

```

8.4 CSS 动画——animation 属性

严格来讲，CSS 动画才是网页设计师最终需要的东西，虽然目前我们还无法指望 CSS 动画能够设计出类似 Flash 的动感效果或者视觉冲击力，但是 animation 属性确实能够给我们带来期望。相信不久的未来，我们能够使用纯 CSS 方法设计出轻量级的 Flash 动画效果来。

W3C 组织在 2009 年 3 月发布了 CSS Animations Module Level 3 草案，详细信息请参阅

<http://www.w3.org/TR/css3-animations/>, 在这个草案中为我们勾勒了 CSS 动画的基本实现方法和属性。

animation 属性是一个复合属性, 它包含了 animation-name、animation-duration、animation-timing-function、animation-delay、animation-iteration-count 和 animation-direction 子属性值。animation 属性的基本语法如表 8.8 所示。

表8.8 animation属性的基本语法

语法项目	说明
值	[<animation-name> <animation-duration> <animation-timing-function> <animation-delay> <animation-iteration-count> <animation-direction>] [, [<animation-name> <animation-duration> <animation-timing-function> <animation-delay> <animation-iteration-count> <animation-direction>]]*
初始值	根据各个子属性的默认值而定
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

浏览器兼容性检测

目前, 只有少数浏览器支持 CSS 动画。Safari 4(Leopard)、Chrome 3、Safari Mobile、Shira 以及其他 webkit 引擎浏览器支持 2D CSS animations 。Safari 4(Snow Leopard) 支持 3D 动画。对 2D 动画的兼容情况如下。

IE	Firefox	Opera	Safari	Chrome
✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✗ IE 7	✗ Firefox 3.0	✗ Opera 9.6	✗ Safari 3.2	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.5	✗ Opera 10.0	✓ Safari 4.0	✓ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.7	✗ Opera 10.5		✓ Chrome 4.0.x

8.4.1 设置 CSS 动画名称——animation-name 属性

animation-name 属性用来定义 CSS 动画的名称。该属性的基本语法如表 8.9 所示。

表8.9 animation-name属性的基本语法

语法项目	说明
值	none IDENT [, none IDENT]*
初始值	none
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

animation-name 属性定义了一个适用的动画列表。该属性值所设置的名字是用来方便选择动画关键帧，提供动画的属性值。如果名称不符合任何一个定义的关键帧，则该动画将不执行。此外，如果动画的名称是 none，那么就不会有动画。这可以用于覆盖任何动画。

8.4.2 设置 CSS 动画时间——animation-duration 属性

animation-duration 属性用来定义 CSS 动画的播放时间。该属性的基本语法如表 8.10 所示。

表8.10 animation-duration属性的基本语法

语法项目	说明
值	<time> [, <time>]*
初始值	0
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

该属性定义动画循环的时间，在默认情况下该属性值为 0，这意味着动画周期为 0，即不会有动画。当值为负值时，则被视为 0。

8.4.3 设置 CSS 动画播放方式——animation-timing-function 属性

animation-timing-function 属性用来定义 CSS 动画的播放方式。该属性的基本语法如表 8.11 所示。

表8.11 animation-timing-function属性的基本语法

语法项目	说明
值	ease linear ease-in ease-out ease-in-out cubic-bezier(<number>, <number>, <number>, <number>) [, ease linear ease-in ease-out ease-in-out cubic-bezier(<number>, <number>, <number>, <number>)]*
初始值	ease
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

关于这些取值说明，可以参阅 transition-timing-function 属性的取值说明。

8.4.4 设置 CSS 动画开始播放的时间——animation-delay 属性

animation-delay 属性用来定义 CSS 动画开始播放的时间，是延迟还是提前等。该属性的基本语法如表 8.12 所示。

表8.12 animation-delay属性的基本语法

语法项目	说明
值	<time> [, <time>]*
初始值	0
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

该属性定义动画的开始时间。它允许一个动画开始执行一段时间后才被应用。当动画延迟时间为 0（即默认的动画延迟时间），则意味着动画将尽快执行，否则该值指定将延迟执行的时间。

8.4.5 设置 CSS 动画播放次数——animation-iteration-count 属性

animation-iteration-count 属性用来定义 CSS 动画的播放次数。该属性的基本语法如表

8.13 所示。

表8.13 animation-iteration-count属性的基本语法

语法项目	说明
值	infinite <number> [, infinite <number>]*
初始值	1
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

该属性定义动画的循环播放次数。默认值为 1，这意味着动画将从开始到结束播放一次。infinite 表示无限次，即 CSS 动画永远重复。如果取值为非整数，将导致动画结束一个周期的一部分。如果取值为负值，则将导致在交替周期内反向播放动画。

8.4.6 设置 CSS 动画播放方向——animation-direction 属性

animation-direction 属性用来定义 CSS 动画的播放次数。该属性的基本语法如表 8.14 所示。

表8.14 animation-direction属性的基本语法

语法项目	说明
值	normal alternate [, normal alternate]*
初始值	normal
适用于	所有块状元素和内联元素
可否继承	否
百分比	N/A
媒介	可视化

取值简单说明：

该属性定义动画播放的方向，取值包括两个值，默認為 normal。当为默认值时，动画的每次循环都向前播放。另一个值是 alternate，设置该值则表示第偶数次向前播放，第奇数次向反方向播放。

CSS 3 实战体验：设计自动翻转的图片效果

在下面这个实战案例中，我们将借助 animation 属性来设计自动翻转的图片效果，该效

果模拟在2D平面中实现3D翻转，演示效果如图8.11所示。

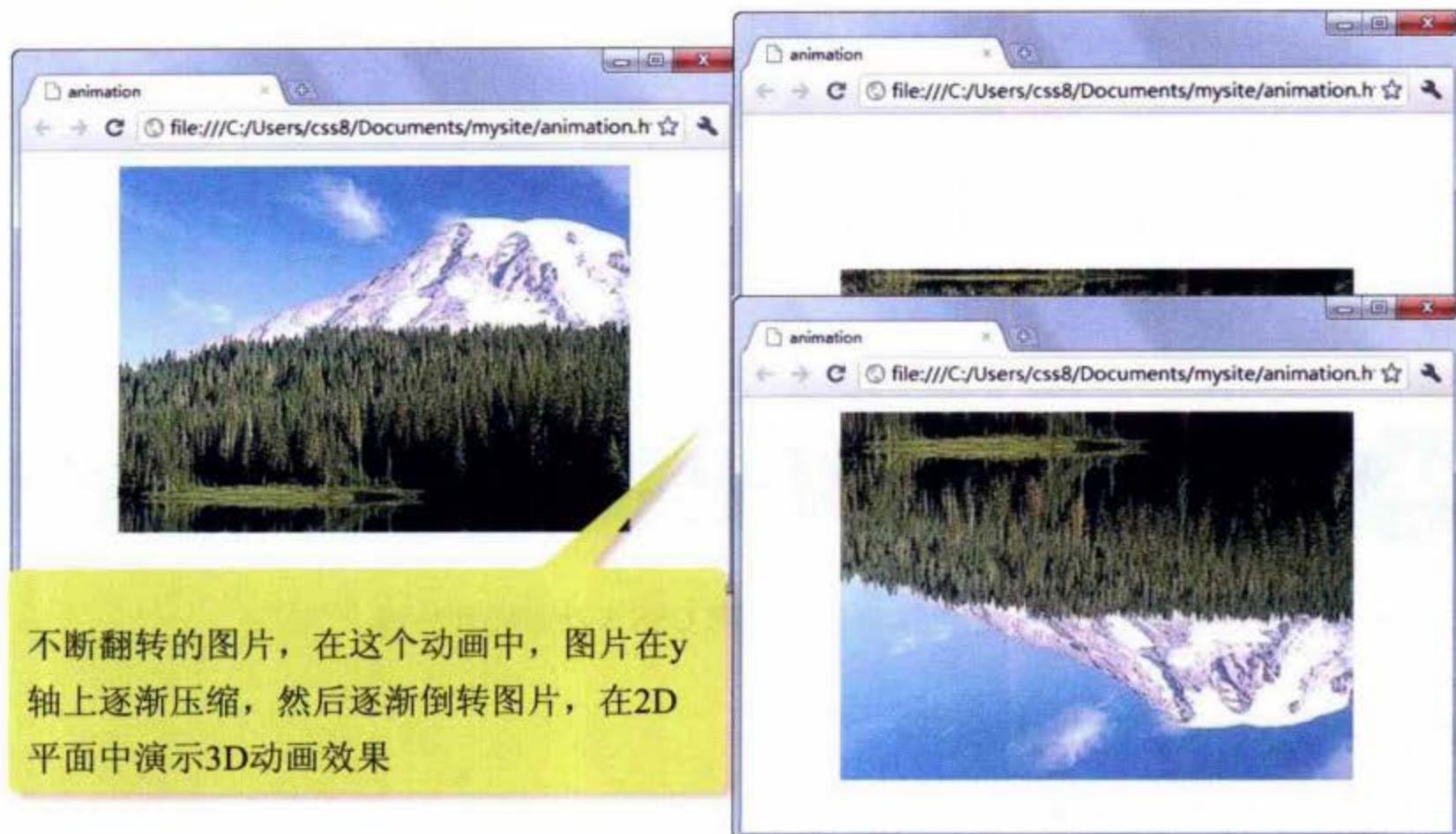


图8.11 立体翻转动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>animation</title>
<style type="text/css">
div {
    margin: 0 auto;
    width: 400px;
    height: 300px;
    background:url(images/bg1.jpg) center no-repeat;
    /* 定义3D空间 */
    -webkit-transform-style: preserve-3d;
    /* 设计沿x轴旋转、20秒线性过渡动画、无限次播放 */
    -webkit-animation-name: x-spin;          /* 定义动画名称为 x-spin */
    -webkit-animation-duration: 20s;         /* 定义动画时间为 20 秒 */
    -webkit-animation-iteration-count: infinite; /* 定义动画播放次数为无穷次 */
    -webkit-animation-timing-function: linear; /* 定义动画过渡为线性转换 */
}
/* 调用动画 */
@-webkit-keyframes x-spin {           /* 引用 x-spin 动画 */
    0% {                            /* 设置第 1 个关键帧为开始位置 */
        -webkit-transform: rotateX(0deg); /* 沿 x 轴开始旋转 */
    }
    50% {                           /* 设置第 2 个关键帧为中间位置 */
        -webkit-transform: rotateX(180deg); /* 沿 x 轴旋转 180 度 */
    }
}
```

```

}
100% {
    /* 设置第 3 个关键帧为结束位置 */
    -webkit-transform: rotateX(360deg);           /* 沿 x 轴旋转 360 度 */
}

</style>
</head>
<body>
<div></div>
</body>
</html>

```

8.5 CSS 3 动画综合实战

本节将通过几个综合案例帮助读者快速掌握 CSS 3 动画设计技能。

8.5.1 设计动态立体盒子

这里我们没有讲解 CSS 3 3D 建模的概念，因为目前浏览器不支持 3D 建模，所以我们也暂时省略。不过，我们仍然可以使用 CSS 3 的 2D 转换功能设计 3D 效果。

注意，使用新发布的（目前只有 Webkit 最新版本的浏览器支持）3D 变换，可以快速设计立体、旋转立方体等效果。

在本案例中，我们没有使用 JavaScript、图像或 SVG 等技术，仅仅利用前面章节中介绍的各种 2D 转换属性进行设计，即采用变换属性中的倾斜，然后旋转定义边框的长方形，通过把三个旋转立方体面结合起来，形成一个三维对象。

本案例兼容 Webkit 和 Gecko 的浏览器，其中 Firefox 3.5+、Safari 3.2+，以及 Chrome 3.0+ 都可以很好地支持。案例演示效果如图 8.12 所示。

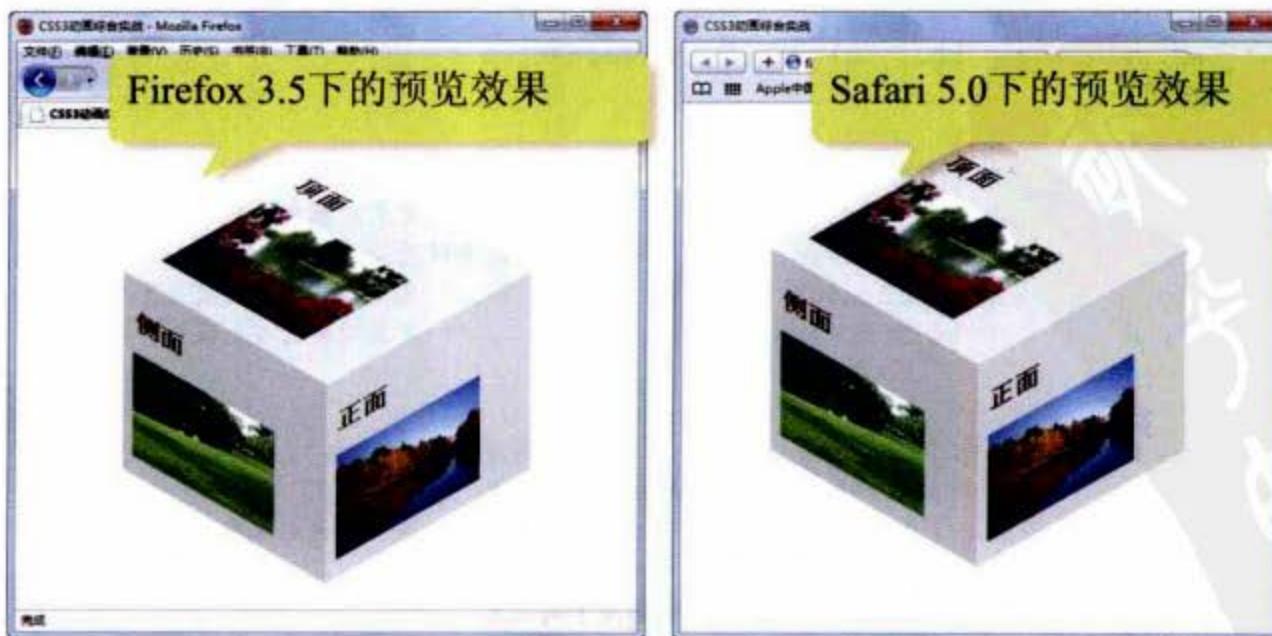




图8.12 3D立体、动态方形盒

完整案例代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3动画综合实战</title>
<style type="text/css">
.cube { /* 固定定位盒子包含框 */
    position: fixed;
    left: 50%;
    top: 12px;
}
.cube p {
    line-height: 14px;
    font-size: 12px;
}
.cube h2 { font-weight: bold; }
.cube.one {
    top: 200px;
    left: 50%;
    margin-left: -200px;
}
.rightFace, .leftFace, .topFace div { /* 统一盒子三立面的尺寸 */
    padding: 10px;
    width: 180px;
    height: 180px;
}
/* 绝对定位盒子三立面 */
.rightFace, .leftFace, .topFace { position: absolute; }
/* 统一盒子三立面的背景色 */

```

```

.cube:hover .rightFace, .cube:hover .leftFace, .cube:hover .topFace div { background-color: #ffc; }
/* 统一鼠标经过时，盒子三立面的背景色 */
.cube:hover .rightFace:hover, .cube:hover .leftFace:hover, .cube:hover .topFace:hover div { background-color: #ffa; }
.leftFace /* 变形左侧面 */
-webkit-transform: skew(0deg, 30deg);
-moz-transform: skew(0deg, 30deg);
background-color: #ccc;
}
.rightFace /* 变形正面 */
-webkit-transform: skew(0deg, -30deg);
-moz-transform: skew(0deg, -30deg);
background-color: #ddd;
left: 200px;
}
.topFace div /* 变形顶侧面 */
-webkit-transform: skew(0deg, -30deg) scale(1, 1.16);
-moz-transform: skew(0deg, -30deg) scale(1, 1.16);
background-color: #eee;
font-size: 0.862em;
}
.topFace /* 旋转顶侧面 */
-webkit-transform: rotate(60deg);
-moz-transform: rotate(60deg);
top: -158px;
left: 100px;
}
/* 仅兼容Webkit类型浏览器的移动动画 */
.cube { -webkit-transition: -webkit-transform 1s linear; }
.cube:hover { -webkit-transform: translate(202px, 115px); }
</style>
</head>
<body>
<div class="cube one">
    <div class="topFace">
        <div>
            <h2>顶面</h2>
            <p></p>
        </div>
    </div>
    <div class="leftFace">
        <h2>侧面</h2>
        <p></p>
    </div>
    <div class="rightFace">
        <h2>正面</h2>
        <p></p>
    </div>
</div>
</body>
</html>

```

左侧面y轴倾斜30度

右侧面y轴倾斜30度，并翻转

顶侧面y轴倾斜30度，并翻转，同时放大显示

顶侧面逆时针旋转60度

8.5.2 设计 CSS 3 手风琴式折叠面板

折叠面板是设计师经常要做的页面小部件，由于它比较实用，所以在网页上会经常看到。尽管样式各异，但它们的设计思路和实现方法都基本相似，即利用 JavaScript 脚本动态控制每个选项卡盒子的伸缩，实现动态显示和隐藏，从而实现鼠标操作的折叠面板效果。

现在使用 CSS 3 的目标伪类 (:target) 就可以轻松实现折叠面板效果，而不需要使用 JavaScript 脚本。如果配合 Webkit 的过渡效果，就可以设计出手风琴式的折叠动画效果，如图 8.13 所示。

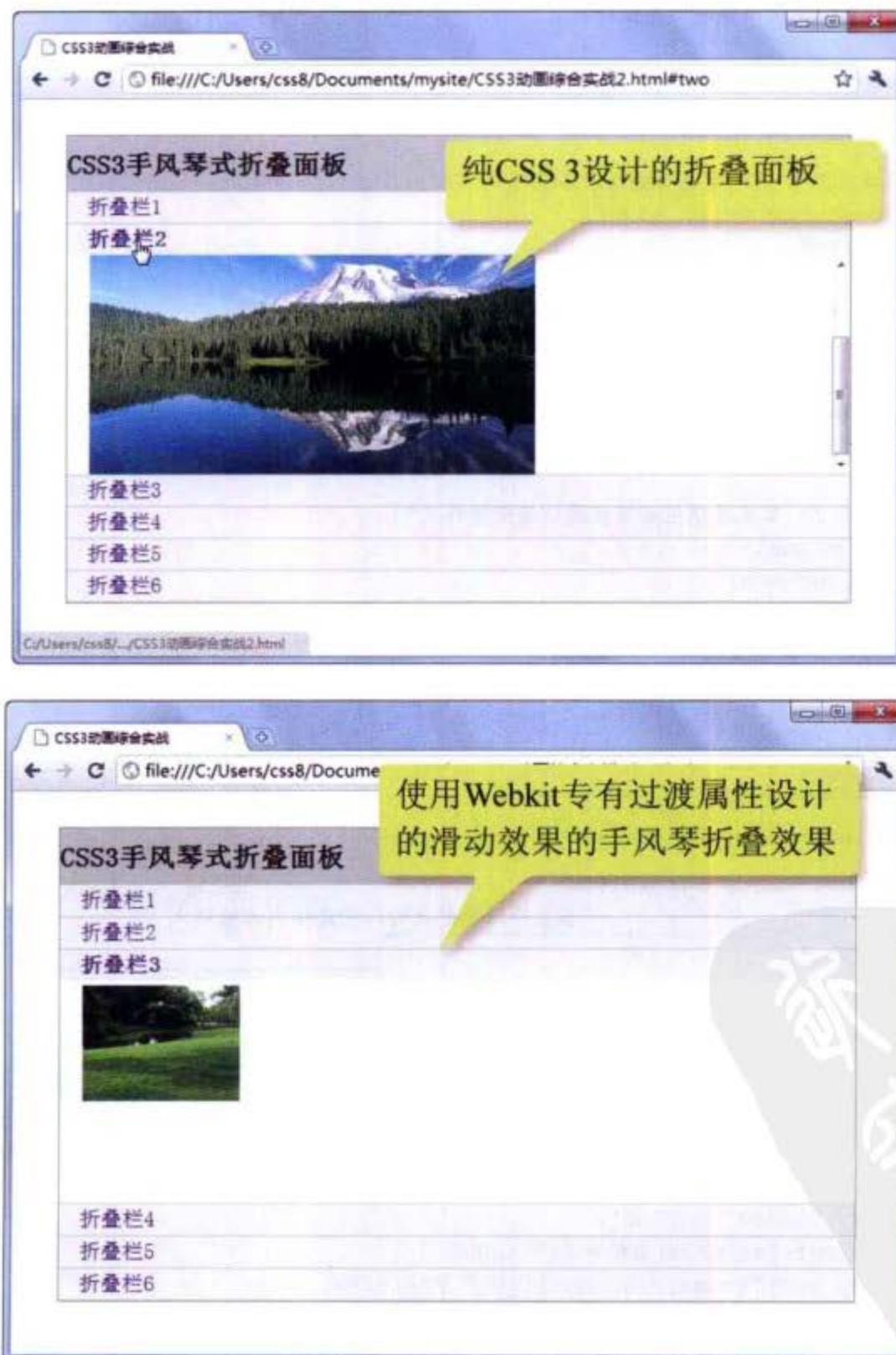


图 8.13 折叠面板

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml11/
DTD/xhtml11-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3动画综合实战</title>
<style type="text/css">
.accordion { /* 定义折叠框外框样式 */
    background: #eee;
    border: 1px solid #999;
    margin: 2em;
}
.accordion h2 { /* 定义折叠框标题栏样式 */
    margin: 0;
    padding: 12px 0;
    background:#CCC
}
.accordion .section { /* 定义折叠框内容框样式 */
    border-bottom: 1px solid #ccc;
    background: #fff;
}
.accordion h3 { /* 定义折叠框选项标题栏样式 */
    margin:0;
    padding:0;
    background: #eee;
    padding:3px 1em;
}
.accordion h3 a { /* 定义折叠框选项标题栏超链接样式 */
    font-weight: normal;
    text-decoration:none;
}
.accordion :target h3 a { font-weight: bold; } /* 当获得目标焦点时，粗体显示选项标题栏文字 */
.accordion h3 + div { /* 选项栏标题对应的选项子框样式 */
    height: 0;
    padding:0 1em;
    overflow: hidden;
    -webkit-transition: height 0.3s ease-in;
}
.accordion h3 + div img { margin:4px; }
.accordion :target h3 + div { /* 当获得目标焦点时，子选项内容框样式 */
    height: 200px;
    overflow:auto;
}
</style>
</head>
<body>
<div class="accordion">
    <h2>CSS3手风琴式折叠面板</h2>
    <div id="one" class="section">
        <h3> <a href="#one">折叠栏1</a> </h3>
        <div></div>
    </div>
    <div id="two" class="section">
        <h3> <a href="#two">折叠栏2</a> </h3>
        <div></div>
    </div>
</div>

```

定义过渡对象为高度，过渡时间为0.3秒，渐显显示

当获取目标之后，高度为300像素

```

</div>
<div id="three" class="section">
    <h3> <a href="#three">折叠栏3</a> </h3>
    <div></div>
</div>
<div id="four" class="section large">
    <h3> <a href="#four">折叠栏4</a> </h3>
    <div></div>
</div>
<div id="five" class="section">
    <h3> <a href="#five">折叠栏5</a> </h3>
    <div></div>
</div>
<div id="six" class="section">
    <h3> <a href="#six">折叠栏6</a> </h3>
    <div></div>
</div>
</div>
</body>
</html>

```

8.5.3 设计能够旋转背景的易拉罐

在下面这个案例中，我们将另辟蹊径，通过最原始的方法来设计三维立体图。用过 CSS 的读者可能熟悉如何使用纯 CSS 方法设计圆角（在 CSS 2 环境下），本案例的立体易拉罐也是利用多标签模拟像素进行堆砌的，如图 8.14 所示。

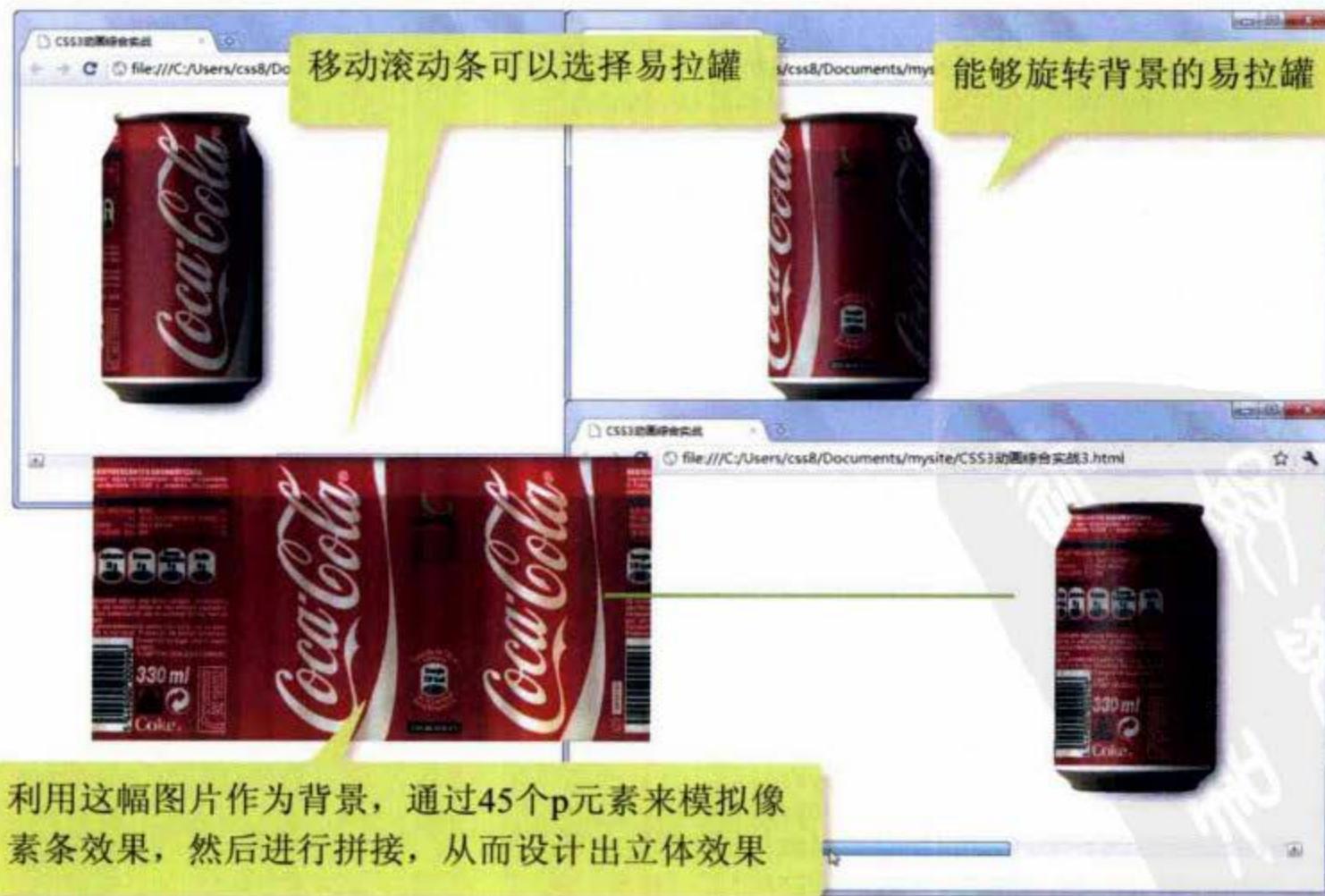


图 8.14 能够旋转的易拉罐

本案例综合使用 CSS 1 的 background-image、background-attachment、background-position 属性，设计二维位移的立体效果，同时当滚动窗口时，位移图将适用于不同部位的纹理（背景图像），从而模拟出旋转的易拉罐效果。

虽然使用 CSS 3 可以设计出三维立体效果，但是我们希望通过本例让读者明白：三维立体效果其实并不高深，只要掌握光线的明暗以及透视原理，在二维平面中使用很多方法都能够呈现三维效果。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3动画综合实战</title>
<style type="text/css">
#coke { /* 易拉罐外框样式 */
    width: 760px;
    height: 400px;
    overflow: auto;
}
img { /* 把易拉罐内胆嵌在底部 */
    border: 0;
    margin-left: -172px;
}
a { /* 把超链接设置为块显示 */
    display: block;
    padding-top: 19px;
    width: 194px;
}
a:hover img { /* 鼠标经过时显示图标 */
    background-image: url(images/coke-title.png);
    background-repeat: no-repeat;
    background-position: 15px 100px;
}
div div { /* 加一个内框 */
    padding-left: 500px;
    width: 760px;
}
p { /* 定义每个单元的样式 */
    margin: 0;
    padding: 0;
    float: left;
    height: 336px;
    width: 1px;
    background-image: url(images/coke-label.jpg); /* 添加背景图像 */
    background-attachment: fixed; /* 固定背景显示 */
    background-repeat: repeat-x; /* 水平平铺 */
}
/* 以下样式列表为设计每列背景图像，通过多条组合拼装出3D效果的易拉罐 */
#x1 { background-position: 5px 30px; }
#x2 { background-position: 0px 30px; }
#x3 { background-position: -3px 30px; }
#x4 { background-position: -6px 30px; }
#x5 { background-position: -8px 30px; }
#x6 { background-position: -10px 30px; }
```

```
#x7 { background-position: -12px 30px; }
#x8 { background-position: -14px 30px; }
#x9 { background-position: -15px 30px; }
#x10 { background-position: -16px 30px; }
#x11 { background-position: -17px 30px; }
#x12 { background-position: -18px 30px; }
#x13 { background-position: -19px 30px; }
#x14 { background-position: -20px 30px; }
#x15 { background-position: -21px 30px; }
#x16 { background-position: -22px 30px; width: 2px; }
#x17 { background-position: -23px 30px; }
#x18 { background-position: -24px 30px; width: 2px; }
#x19 { background-position: -25px 30px; width: 2px; }
#x20 { background-position: -26px 30px; width: 2px; }
#x21 { background-position: -27px 30px; width: 2px; }
#x22 { background-position: -28px 30px; width: 3px; }
#x23 { background-position: -29px 30px; width: 3px; }
#x24 { background-position: -30px 30px; width: 4px; }
#x25 { background-position: -31px 30px; width: 5px; }
#x26 { background-position: -32px 30px; width: 7px; }
#x27 { background-position: -33px 30px; width: 12px; }
#x28 { background-position: -34px 30px; width: 55px; }
#x29 { background-position: -35px 30px; width: 11px; }
#x30 { background-position: -36px 30px; width: 6px; }
#x31 { background-position: -37px 30px; width: 5px; }
#x32 { background-position: -38px 30px; width: 4px; }
#x33 { background-position: -39px 30px; width: 3px; }
#x34 { background-position: -40px 30px; width: 2px; }
#x35 { background-position: -41px 30px; width: 3px; }
#x36 { background-position: -42px 30px; width: 2px; }
#x37 { background-position: -43px 30px; width: 2px; }
#x38 { background-position: -44px 30px; }
#x39 { background-position: -45px 30px; width: 2px; }
#x40 { background-position: -46px 30px; }
#x41 { background-position: -47px 30px; }
#x42 { background-position: -48px 30px; }
#x43 { background-position: -49px 30px; }
#x44 { background-position: -50px 30px; }
#x45 { background-position: -51px 30px; }
#x46 { background-position: -52px 30px; }
#x47 { background-position: -53px 30px; }
#x48 { background-position: -54px 30px; }
#x49 { background-position: -56px 30px; }
#x50 { background-position: -58px 30px; }
#x51 { background-position: -60px 30px; }
#x52 { background-position: -62px 30px; }
#x53 { background-position: -65px 30px; }
#x54 { background-position: -68px 30px; }
#x55 { background-position: -74px 30px; }
</style>
</head>
<body>
<div id="coke">
    <div id="y">
        <p id="x1"></p><p id="x2"></p><p id="x3"></p><p id="x4"></p><p id="x5"></p>
        <p id="x6"></p><p id="x7"></p><p id="x8"></p><p id="x9"></p><p id="x10"></p>
        <p id="x11"></p><p id="x12"></p><p id="x13"></p><p id="x14"></p><p id="x15"></p>
        <p id="x16"></p><p id="x17"></p><p id="x18"></p><p id="x19"></p><p id="x20"></p>
```

```

<p id="x21"></p><p id="x22"></p><p id="x23"></p><p id="x24"></p><p id="x25"></p>
<p id="x26"></p><p id="x27"></p><p id="x28"></p><p id="x29"></p><p id="x30"></p>
<p id="x31"></p><p id="x32"></p><p id="x33"></p><p id="x34"></p><p id="x35"></p>
<p id="x36"></p><p id="x37"></p><p id="x38"></p><p id="x39"></p><p id="x40"></p>
<p id="x41"></p><p id="x42"></p><p id="x43"></p><p id="x44"></p><p id="x45"></p>
<p id="x46"></p><p id="x47"></p><p id="x48"></p><p id="x49"></p><p id="x50"></p>
<p id="x51"></p><p id="x52"></p><p id="x53"></p><p id="x54"></p><p id="x55"></p>
<a href="#"></a>
</div>
</div>
</body>
</html>

```

8.5.4 设计旋转出仓的光盘动画效果

在下面这个案例中，我们将模拟光盘旋转出仓的动画效果，如图 8.15 所示。考虑到代码的简洁性，本案例暂时兼容 Webkit 类型的浏览器。

本案例采用一个标准的专辑封面、一点点 HTML 代码和一些 CSS 3 的过渡和转变，创造一个炫目的光盘滑动出仓的动画效果。

实现难点：旋转光盘的动画，这个效果通过旋转转换实现；光盘出仓的动画，这个效果主要通过移动转换实现；通过渐变和阴影设计光盘效果以及局部细节。



图 8.15 滑动门动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3动画综合实战</title>
<style type="text/css">
body { background: #fff url(images/wood-oak.jpg); }
ul.tunes { margin-left: -20px; } /* 设计外框向左移动 */
ul.tunes li { /* 定义每个光盘项的样式 */
    position: relative; /* 包含块 */
    width: 200px;
    height: 190px;
    float: left;
    margin-left: 20px;
}
ul.tunes li div.album { /* 音乐盒外框 */
    margin: 0 0 48px 0;
    display: inline;
    width: 200px;
    height: 120px;
    position: absolute;
    text-decoration: none;
    -webkit-transition: all .15s linear; /* 线性动画，0.15秒，针对所包含的所有元素 */
    color: #333;
    left: 0px;
    top: 0px;
}
ul.tunes img {
    width: 120px;
    position: relative;
    z-index: 30;
    float: left;
    -webkit-box-shadow: 0 3px 6px rgba(0, 0, 0, .5); /* 音乐盒阴影 */
    -webkit-border-radius: 2px; /* 音乐盒圆角 */
}
ul.tunes li div.album div {
    display: block;
    opacity: .95;
    text-align: center;
    -webkit-transition: all .25s linear; /* 线性动画，0.25秒，针对所包含的所有元素 */
    clear: left;
    width: 120px;
}
ul.tunes li div.album div h5 { text-align: center; }
ul.tunes li div.album:hover div { opacity: 1; } /* 鼠标经过时外框的不透明度 */
ul.tunes li div.album span.vinyl { /* 光驱样式 */
    width: 116px;
    height: 116px;
    z-index: 1;
    display: block;
    -webkit-transition: all .25s linear; /* 线性动画，0.25秒，针对所包含的所有元素 */
    position: absolute;
    top: 2px;
    left: 2px;
    margin-left: 16px;
}
```

```

ul.tunes li div.album span.vinyl div { /* 光驱内框样式 */
    position: absolute;
    top: 2px;
    left: 2px;
    display: block;
    z-index: 10;
    width: 112px;
    height: 112px;
    -webkit-border-radius: 59px;           /* 圆角 */
    -moz-border-radius: 59px;             /* 圆角 */
    -webkit-box-shadow: 0 0 6px rgba(0, 0, 0, .5); /* 阴影 */
    -webkit-transition: all .25s linear; /* 线性动画，0.25秒，针对所包含的所有元素 */
    overflow: hidden; /* 禁止超出外框 */
    border: solid 1px black;
    /* 设计光驱效果，通过直线渐变和径向渐变设计光泽效果 */
    background:
        -webkit-gradient( linear, left top, left bottom, from(transparent), color-stop(0.1,
        transparent), color-stop(0.5, rgba(255, 255, 255, 0.25)), color-stop(0.9, transparent),
        to(transparent)), -webkit-gradient( radial, 56 56, 10, 56 56, 114, from(transparent), color-
        stop(0.01, transparent), color-stop(0.021, rgba(0, 0, 0, 1)), color-stop(0.09, rgba(0, 0, 0, 1)),
        color-stop(0.1, rgba(28, 28, 28, 1)), to(rgba(28, 28, 28, 1)));
        border-top: 1px solid rgba(255, 255, 255, .25);
}
/* 鼠标经过时，设计出仓动画效果 */
ul.tunes li div.album:hover span.vinyl { -webkit-transform: translateX(60px); }
ul.tunes li div.album:hover span.vinyl div { /* 在出仓过程中旋转内框，营造立体动画效果 */
    -webkit-transform: rotate(120deg);
    border-top: 0;
    border-left: 1px solid rgba(255, 255, 255, .25);
}
ul.tunes li span.gloss { /* 设计掩饰物 */
    display: block;
    position: absolute;
    top: 0;
    left: 0;
    width: 120px;
    height: 120px;
    background: url(images/sheen3.png) no-repeat;
    z-index: 100;
}
ul.tunes li div.album ul.actions { /* 设计音乐盒被激活时的样式 */
    display: block;
    position: absolute;
    width: 60px;
    /* 圆角效果 */
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    left: 70px;
    top: 0px;
    height: 114px;
    z-index: 20;
    -webkit-transition: all 0.25s linear; /* 线性动画，0.25秒，针对所包含的所有元素 */
}
/* 激活时光盒出仓动画 */
ul.tunes li div.album:hover ul.actions { -webkit-transform: translateX(60px); }
ul.tunes li div.album ul.actions li { /* 激活时光盒上按钮样式 */
    display: block;
}

```

```
position: absolute;
height: 20px;
width: 20px;
left: 10px;
top: 22px;
/* 设计凹凸效果 */
background: -webkit-gradient(linear, left top, left bottom, from(black), to(#333));
/* 设计圆角 */
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
-webkit-box-shadow: 0 1px 0 rgba(255, 255, 255, .15); /* 添加阴影 */
}
ul.tunes li div.album ul.actions li:hover { background: -webkit-gradient(linear, left top, left bottom, from(#333), to(black)); } /* 被激活时改变光盘上按钮的凹凸效果 */
ul.tunes li div.album ul.actions li.info {
    top: 48px;
    left: 19px;
}
ul.tunes li div.album ul.actions li a {
    display: block;
    width: 20px;
    height: 20px;
}
ul.tunes li div.album ul.actions li.play-pause a { background: url(images/play-button.png) no-repeat center center; } /* 播放按钮 */
ul.tunes li div.album ul.actions li.info a { background: url(images/info.png) no-repeat center center; } /* 信息按钮 */
ul.tunes li { text-shadow: 0 2px 3px rgba(0, 0, 0, .75); }
ul.tunes h5 {
    padding-top: 8px;
    color: #fff;
}
ul.tunes small {
    color: #fff;
    opacity: .75;
}

```

</style>

</head>

<body>

<ul class="tunes">

<div class="album">

<div></div>

<ul class="actions">

<li class="play-pause">

<li class="info">

<div>

<h5>Michael Jackson</h5>

<small>Thriller</small></div>

</div>

<div class="album">


```
<div></div>
</span>
<ul class="actions">
    <li class="play-pause"><a href=""></a></li>
    <li class="info"><a href=""></a></li>
</ul>
<div>
    <h5>Michael Jackson</h5>
    <small>Bad</small> </div>
    <span class="gloss"></span></div>
</li>
<li>
    <div class="album"> <a href=""></a> <span class="vinyl">
        <div></div>
        </span>
        <ul class="actions">
            <li class="play-pause"><a href=""></a></li>
            <li class="info"><a href=""></a></li>
        </ul>
        <div>
            <h5>Michael Jackson</h5>
            <small>Off the Wall</small></div>
            <span class="gloss"></span></div>
        </li>
        <li>
            <div class="album"> <a href=""></a>
<span class="vinyl">
            <div></div>
            </span>
            <ul class="actions">
                <li class="play-pause"><a href=""></a></li>
                <li class="info"><a href=""></a></li>
            </ul>
            <div>
                <h5>Jay-Z</h5>
                <small>The Blueprint 3</small></div>
                <span class="gloss"></span></div>
            </li>
            ...
            <li>
                <div class="album"> <a href=""></a>
<span class="vinyl">
                    <div></div>
                    </span>
                    <ul class="actions">
                        <li class="play-pause"><a href=""></a></li>
                        <li class="info"><a href=""></a></li>
                    </ul>
                    <div>
                        <h5>Muse</h5>
                        <small>The Resistance</small></div>
                        <span class="gloss"></span></div>
                    </li>
                </ul>
            </body>
        </html>
```

CSS 3 新增的其他功能

应该说，CSS 3 目前还处于生命的幼年时期，大量的新增功能还在不断孕育和完善中，所以本书也无法穷尽 CSS 3 的全部功能。在 W3C 官方网站 (<http://www.w3.org/Style/CSS/>) 中，你能够感受到各个项目组正在忙碌工作着，新的工作草案不断被推出，新的模块随时被列上日程表。此外，即便是 W3C 制订了各种标准草案，这些标准能否获得各主流浏览器的支持，也将是一个漫长的过程。本章将就 CSS 3 中另外几个重要的新增功能进行介绍。其他细节限于篇幅就不再详细讲解，感兴趣的读者可以访问 W3C 官方网站查询。

The screenshot shows the W3C CSS Working Group website at <http://www.w3.org/Style/CSS/>. A yellow callout box highlights the top news section, which lists several recent milestones and events. Another yellow callout box highlights the 'Current work' section, which displays a table of specifications with their current status.

News Section (Yellow Callout):

- 2010-10-28 Proposed Recommendation: CSS Color Module Level 3
- 2010-10-12 Candidate Recommendation: CSS Style Attributes
- 2010-10-09 Daniel Glazman of Disruptive Innovations announced "milestone 1" WYSIWYG Web editor with support for HTML, MathML, SVG and CSS (full ie Windows, Mac; Open Source)
- 2010-10-07 New style! The CSS pages have a new style sheet and an improved structure.
- 2010-10-05 Working Draft: CSS Text Level 3
- 2010-09-10 WWW 2011, the 20th World Wide Web Conference, in Hyderabad, India, March 28 to April 1, 2011
- 2010-09-10 WWW 2012, the 21st World Wide Web Conference, in Lyon, France, April 16 to 20, 2012.
- 2010-07-27 Candidate Recommendation: Media Queries

Current Work Section (Yellow Callout):

High Priority	Current	Upcoming
CSS Level 2 Revision 1	CR	PR
Selectors	PR	REC
CSS Mobile Profile 2.0	CR	PR
CSS Marquee	CR	PR

9.1 引用外部字体类型——@font-face 规则

@font-face 规则在 CSS 3 规范中属于字体模块 (<http://www.w3.org/TR/css3-fonts/#font-face>)，该规则的推出对于网页设计来说将是一个革命性的进步。在传统设计中，设计师不敢使用各种艺术字体类型，甚至是常规字体也需慎重使用，因为设计师必须考虑每位浏览者的系统中是否安装了所用的字体。有了 @font-face 规则，这个顾虑就可以放下了：只要在互联网上指定一种字体类型源，而不管用户电脑是否安装该字体，设计的网页都能够正确显示。

用较专业的话来讲，@font-face 能够加载服务器端的字体文件，让客户端浏览器显示客户端没有安装的字体。如果没有 @font-face 规则，浏览器只能够在客户端系统中寻找指定字体，这就给网页设计带来了很多限制，妨碍了设计师的创意设计，也就无法展现丰富多彩的字体艺术。

9.1.1 @font-face 规则的用法

@font-face 规则的语法格式如下：

```
@font-face { <font-description> }
```

@font-face 规则的选择符是固定的，用来引用服务器端的字体文件。

<font-description> 是一个属性名值对，格式类似如下样式：

```
descriptor: value;  
descriptor: value;  
descriptor: value;  
descriptor: value;  
[...]  
descriptor: value;
```

属性及其取值说明如下。

- font-family：设置文本的字体名称。
- font-style：设置文本样式。
- font-variant：设置文本是否大小写。
- font-weight：设置文本的粗细。
- font-stretch：设置文本是否横向拉伸变形。
- font-size：设置文本字体大小。
- src：设置自定义字体的相对路径或者绝对路径。注意，该属性只能在 @font-face 规则里使用。

● 浏览器兼容性检测

实际上，微软的IE 5已经开始支持该属性，但是只支持微软自有的.eot（Embedded Open Type）字体格式，而其他浏览器直到现在都不支持这一字体格式。不过，从Safari 3.1开始，网页重构工程师已经可以设置.ttf（TrueType）和.otf（OpenType）两种字体作为自定义字体了。

借助兼容方式，各主流浏览器对@font-face规则的支持情况如下。

.eot 格式

✓ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 1.0.x
✓ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 3.0.x
✓ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 4.0.x
✓ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 8.0.x

.ttf 格式

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✓ Safari 3.0	✗ Chrome 2.0
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✓ Safari 4.0	✓ Chrome 2.0.162
✗ IE 8	✗ Firefox 3.0	✓ Opera 10.0		✓ Chrome 3.0.x
✗ IE 9	✓ Firefox 3.5	✓ Opera 10.5		✓ Chrome 4.0.x

实战体验：设计艺术字体

下面我们看一个简单的示例，力图通过简洁的代码让读者学会使用@font-face规则。示例代码如下，演示效果如图9.1所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>@font-face规则</title>
<style type="text/css">
/* 引入外部字体文件 */
@font-face {
    font-family: "lexograph";
    /* 兼容IE */
}

```

```

src: url(http://randsco.com/fonts/lexograph.eot);
/* 兼容非IE */
src: local("Lexographer"), url(http://randsco.com/fonts/lexograph.ttf)
format("truetype");
}
h1 {
font-family: lexograph, verdana, sans-serif; /* 设置引入字体文件中的 lexograph 字体类型 */
font-size:4em;
}
</style>
</head>
<h1>CSS3 @font-face</h1>
<body>
</body>
</html>

```



图 9.1 应用@font-face 规则

注意，嵌入外部字体的做法对于中文网站来说不大适用，因为一个中文字体文件小的有几个 MB、大的有十几个 MB，这么大的字体文件，其下载过程让人无法忍受，同时服务器也不能接受如此频繁的申请下载。所以对中文来说，如果只是想标题使用特殊字体，最好设计成图片。由于英文字体文件只有几十 KB，与一张图片的大小差不多，如果有大量的文字需要使用该字体，存储、带宽方面就划算多了。

9.1.2 关于开放字体格式

@font-face 规则虽然让网页设计师很兴奋，但是也存在着诸多问题，于是，在 Web 设计界诞生了 Web 开放字体格式。

Web 开放字体格式由 Mozilla 发起，旨在让设计者在 Web 中使用自己的个性字体。Web 开放字体格式允许设计者像链接图片那样链接自己的字体文件，并使用该字体显示文字。@font-face 允许浏览器下载指定的字体，但并没有指明字体的格式，使用 @font-face 可能会在不同浏览器上产生不同的效果。著名博客站点 Boing Boing 就曾使用 @font-face 规则对站点进行重建，结果以失败告终。Web 开放字体格式正是为了解决 @font-face 存在的问题而出现的。

Web 开放字体格式的另一个好处是它本身是经过压缩的，这样就避免了下载巨大字体文件的问题。Web 开放字体格式目前已经部署到 Firefox 3.6，Opera 和微软都表示将提供支持，该格式目前已经提交到 W3C，以期成为标准。不过，微软的 IE 9 还不支持该格式，但微软是该标准的提议者之一，因此，我们可以期待 IE 9 最终发布时会包含对该功能的支持。Chrome 也支持 Web 开放字体格式，目前仅有苹果的 Safari 未标明支持该技术的浏览器。

除了浏览器的支持外，另一个重要的问题是字体设计商的支持。因为 Web 开放字体格式并不安全，字体商有可能不愿对之提供支持。不过，包括 Adobe、House Industries、Hoefler Frere-Jones 以及 ITC、Linotype 在内的字体商都表示支持该格式。

尽管 Web 开放字体格式并不能解决 Web 字体的所有问题，但它是朝着正确的方向迈进了一大步，随着 Firefox、Opera、Chrome 等浏览器对该技术的支持，设计者有望很快在网页设计中任意使用自己喜欢的字体。

9.2 定义 CSS 设备类型——Media Queries

移动时代是任何网页设计师与开发者都不能忽视的一个时代，总有一天，我们设计的页面会在电脑大屏幕或者移动小屏幕上显示。如何让同一个网站同时适应完全不同尺寸的屏幕，这是很久以来都没有完美解决方案的问题，直到有了 CSS3。

在 CSS 2.1 版本时，我们曾经为网站设计不同的 CSS 样式表文件，如打印样式表文件、手机样式表文件、电脑样式表文件等，这种通过在 CSS 样式表中指定设备类型的做法，显然比较低级。

- 设置外部样式表文件的设备类型。

```
<link href="csss.css" rel="stylesheet" type="text/css" media="handheld" />
```

- 设置内部样式表文件的设备类型。

```
<style type="text/css" media="screen">  
...  
</style>
```

为了解决这个问题，CSS 3 提出了 Media Queries（媒体查询）这个新概念，它比 CSS 2 的 Media Type（媒体类型）更加实用，事实上，CSS 2 的 Media Type 并不曾被多少设备支持过。而 CSS 3 的 Media Queries 可以帮助设计师获取以下数据。

- 浏览器窗口的宽和高。
- 设备的宽和高。
- 设备的手持方向，横向还是纵向。

□ 分辨率。

注意，W3C 于 2010 年 7 月推荐了 Media Queries 标准块，详细信息请参阅 <http://www.w3.org/TR/css3-mediaqueries/>。

Media Queries 允许添加表达式用以确定媒体的情况，以此来应用不同的样式表。换句话说，它允许我们在不改变内容的情况下改变页面的布局，以精确适应不同的设备，以此加强体验。这种设计灵感来源于 IE 的条件语句，不过 Media Queries 的功能更加强大。

```
<!--[if IE]>
    根据IE条件表达式决定在不同IE版本中可被解析的代码，如CSS样式、JavaScript脚本、HTML结构
<![endif]-->
```

如果用户有一个支持 Media Queries 的设备，我们就可以为该设备编写专门的 CSS，让网站适应这个设备的屏幕显示。所以，Media Queries 和 CSS 优化没有关系，甚至是矛盾的。

9.2.1 @media 规则的用法

@media 规则的语法格式如下：

```
@media: <sMedia> { sRules }
```

参数简单说明：

■ <sMedia>：指定设备名称。CSS 设备类型包括如下这些。

- all：用于所有设备类型。
- aural：用于语音和音乐合成器。
- braille：用于触觉反馈设备。
- embossed：用于凸点字符（盲文）印刷设备。
- handheld：用于小型或手提设备。
- print：用于打印机。
- projection：用于投影图像，如幻灯片。
- screen：用于计算机显示器。
- tty：用于使用固定间距字符格的设备，如电传打字机和终端。
- tv：用于电视类设备。

■ {sRules}：定义样式表。

通过判断设备（对象）的类型来实现不同的 CSS 样式呈现，这样就可以使 CSS 更精确作用于不同的设备类型，以及同一设备的不同条件（分辨率、色数等）。详细用法如下。

```
media_query: [only | not]? <media_type> [ and <expression> ]*
expression: ( <media_feature> [: <value>]? )
media_type: all | aural | braille | handheld | print | projection | screen | tty | tv |
embossed
media_feature: width | min-width | max-width
```

```

| height | min-height | max-height
| device-width | min-device-width | max-device-width
| device-height | min-device-height | max-device-height
| device-aspect-ratio | min-device-aspect-ratio | max-device-aspect-ratio
| color | min-color | max-color
| color-index | min-color-index | max-color-index
| monochrome | min-monochrome | max-monochrome
| resolution | min-resolution | max-resolution
| scan | grid

```

在 @media 规则中，主要参数说明如下。

- **media_query**：设置逻辑关键字，如 and（逻辑与）、not(排除某种设备)、only(限定某种设备)等。
- **media_type**：设置设备类型，在 CSS 2.1 中已经定义了上面介绍的 10 种媒体类型。
- **media_feature**：定义媒体特性，该特性放置在一对圆括号中，如 (min-width: 400px)。完整的特性说明请参阅表 9.1。

表9.1 Media Queries媒体特性说明

媒体特性	值	可用媒体类型	接受min/max
width	length	visual、tactile	yes
height	length	visual、tactile	yes
device-width	length	visual、tactile	yes
device-height	length	visual、tactile	yes
orientation	portrait landscape	bitmap	no
aspect-ratio	ratio	bitmap	yes
device-aspect-ratio	ratio	bitmap	yes
color	integer	visual	yes
color-index	integer	visual	yes
monochrome	integer	visual	yes
resolution	resolution	bitmap	yes
scan	progressive interlace	tv	no
grid	integer	visual、tactile	no

媒体特性共 13 种，可以说是一个类似 CSS 属性的集合。但和 CSS 属性不同的是，媒体特性只接受单个的逻辑表达式作为其值，或者没有值，并且其中的大部分接受 min/max 的前缀，用来表示大于等于 / 小于等于的逻辑，以此避免使用 < 和 > 这些字符。

浏览器兼容性检测

各主流浏览器对 @media 规则的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.6	✗ Safari 3.0	✗ Chrome 2.0
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.63	✗ Safari 4.0	✗ Chrome 2.0
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

实战体验：为不同设备设计不同的盒子框样式

下面我们看一个简单的示例，力图通过简洁的代码让读者学会使用 @media 规则。示例代码如下，演示效果如图 9.2 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>@font-face 规则</title>
<style type="text/css">
.wrapper {
    border: solid 1px #666;
    padding: 5px 10px;
    margin: 40px;
}
.viewing-area span {
    color: #666;
    display: none;
}
/* max-width: 如果视图窗口的宽度小于 600 像素，则该盒子将变成品红色 */
@media screen and (max-width: 600px) {
    .one {
        background: #F9C;
    }
    span.lt600 {
        display: inline-block;
    }
}
/* min-width: 如果视图窗口的宽度大于 900 像素，则该盒子将变成橙色 */
@media screen and (min-width: 900px) {
    .two {
        background: #F90;
    }
    span.gt900 {
        display: inline-block;
    }
}
/* min-width & max-width: 如果视图窗口的宽度小于 600 像素，则该盒子将变成品红色 */
@media screen and (min-width: 600px) and (max-width: 900px) {
    .three {
        background: #9CF;
    }
}
```

```



```



图9.2 应用@media规则

9.2.2 使用 Media Queries 链接外部 CSS 文件

对于单页样式设计来说，直接在 CSS 内部样式表中插入移动设备样式代码分支是很方便的，但对于大型网站来说，这种做法就非常不科学了。不过，我们可以使用 Media Queries 链接外部 CSS 样式表文件，以便在独立的样式表文件中完全自由地为小设备编写 CSS 代码。具体用法如下。

```
<link rel="stylesheet" type="text/css" href="small-device.css"
      media="only screen and (max-device-width: 480px)" />
```

通过在 `<link>` 标签中设置 `media` 属性，添加 Media Queries 规则，此时 `media` 属性值的语法格式遵循 `@media` 规则的用法。这样 `media` 属性类似于标签的 `style` 属性，即在标签内添加样式属性。

一个 Media Query 语句包含一种媒体类型，如果媒体类型没有指定，那么就是默认类型 `all`，例如：

```
<link rel="stylesheet" type="text/css" href="example.css"
      media="(max-width: 600px)">
```

一个 Media Query 包含 0 到多个表达式，表达式又包含 0 到多个关键字，以及一种媒体特性，例如：

```
<link rel="stylesheet" type="text/css" href="example.css"
      media="handheld and (min-width:20em) and (max-width:50em)">
```

逗号 (,) 用来表示并列，或表示“或者”。例如，下面的代码表示该 CSS 样式表应用于宽度小于 20em 的手持设备，或者宽度小于 30em 的屏幕设备中。

```
<link rel="stylesheet" type="text/css" href="example.css"
      media="handheld and (max-width:20em), screen and (max-width:30em)">
```

`not` 关键字用来排除符合表达式的设备，例如：

```
<link rel="stylesheet" type="text/css" href="example.css"
      media="not screen and (color)">
```

例如，再看下面这个示例。

```
<link rel="stylesheet" type="text/css" href="styleA.css"
      media="screen and (min-width: 800px)">
<link rel="stylesheet" type="text/css" href="styleB.css"
      media="screen and (min-width: 600px) and (max-width: 800px)">
<link rel="stylesheet" type="text/css" href="styleC.css"
      media="screen and (max-width: 600px)">
```

上面将设备分成 3 种，分别是：宽度大于 800px 时，应用 styleA；宽度在 600px 到 800px 之间时应用 styleB；宽度小于 600px 时应用 styleC。这其中也包括一个 CSS 覆盖的问题，即当宽度正好等于 800px 时，该应用哪个样式？答案是 styleB，因为前两条表达式都成立，后者覆盖了前者。所以说，上面的例子虽然能工作，但是不准确。这个例子应该这样写：

```
<link rel="stylesheet" type="text/css" href="styleA.css"
      media="screen">
<link rel="stylesheet" type="text/css" href="styleB.css"
      media="screen and (max-width: 800px)">
<link rel="stylesheet" type="text/css" href="styleC.css"
      media="screen and (max-width: 600px)">
```

并非所有的浏览器都支持 Media Queries，那么这些浏览器是怎么看待 Media Queries 的呢？

Media Queries 是 CSS 3 对 Media Type 的一个扩展，所以不支持 Media Queries 的浏览器，应该仍然要识别 Media Type，但是 IE 只是简单地忽略了样式。only 关键字可能显得有些多余，对支持 Media Queries 的浏览器来说确实是这样，因为加不加 only 没什么影响。only 的作用很多时候是用来对那些不支持 Media Queries 但是却读取 Media Type 的设备隐藏样式表的。例如：

```
<link rel="stylesheet" type="text/css" href="example.css"
      media="only screen and (color)">
```

支持 Media Queries 的设备会正确应用样式；不支持 Media Queries，但能正确读取 Media Type 的设备，由于先读取到 only 而不是 screen，所以将忽略这个样式；而不支持 Media Queries 的 IE，不论是否有 only，都会忽略样式。

9.2.3 测试 Media Queries

要在不同设备上测试 Media Queries 并非易事，用户需要准备各种设备，还要将代码上传到某个主机进行访问测试。不过，这里推荐一个网址 (<http://protofluid.com/>)，它能够提供在线测试服务。该服务允许提供要测试的网站 URL，或者本机上的 URL，然后，模拟 iPhone 等移动设备显示网页设计，用户也可以填写自己的窗口尺寸，来模拟特定的设备。

9.3 定义投影——CSS Reflections

顾名思义，CSS Reflections 就是使用 CSS 设计投影效果，这种效果在传统设计中只能够通过 Photoshop 等图像编辑软件事先设计好，然后再导入网页中。CSS Reflections 简化了这种操作，它允许设计师直接使用 CSS 通过代码进行控制，编写一行代码就可以，很容易完成，并提供简单而有效的效果。例如，给一个图片库设计投影效果，从而设计出超酷的视觉效果。

浏览器兼容性检测

目前，CSS Reflections 仅获得 Webkit 引擎的支持，所以我们只能够在谷歌 Chrome 和

Safari 浏览器中进行测试。W3C 还没有推出 CSS Reflection 标准草案。不过，我们相信不久的将来，该标准模块会制定出来。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 2.0.x
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.6	✗ Safari 4.0	✗ Chrome 2.0.x
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

-webkit-box-reflect 的基本用法

Webkit 引擎定义了 -webkit-box-reflect 属性，该属性能够实现投影效果，具体语法格式如下。

```
-webkit-box-reflect: <direction> <offset> <mask-box-image>
```

属性取值说明如下。

- <direction>：定义反射方向，取值包括 above、below、left 和 right。
- <offset>：定义反射偏移的距离，取值包括数值或者百分比，其中百分比是根据对象的尺寸进行确定，如果省略该参数值，则默认为 0。
- <mask-box-image>：定义遮罩图像，该图像将覆盖投影区域。如果省略该参数值，则默认为无遮罩图像。也可以设置渐变色或者纯色覆盖。

当对象源发生变化时，投影能够自动更新，当鼠标经过对象上时，也能够在投影中看到鼠标效果。如果该属性应用到 <video> 标签上，还可以看到视频以投影效果进行播放。

记住，投影的规模和反射偏移不影响页面的布局。

CSS 实战体验：应用 CSS Reflections

```
<style type="text/css">
body {background:rgb(51, 52, 154)
      url(images/reflectionsProjections-bg.jpg)
      no-repeat 0 0; }

img {
    height:400px;
    border:5px solid white;
    -webkit-box-reflect:below;
}
</style>



<style type="text/css">
```



```

body { background:rgb(51, 52, 154)
       url(images/reflectionsProjections-
bg.jpg) no-repeat 0 0; }

img {
    height:400px;
    border:5px solid white;
    -webkit-box-reflect:below 10px;
}

</style>

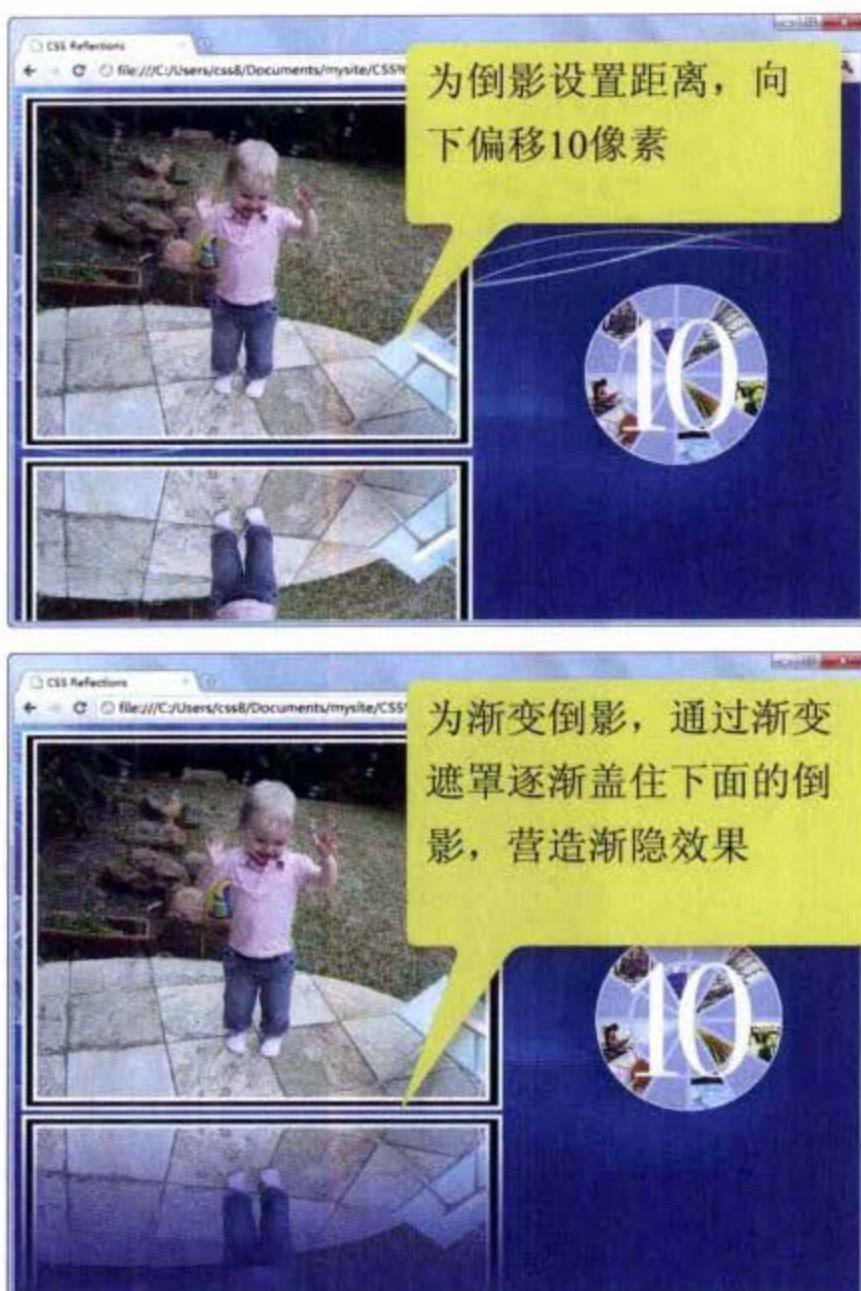


<style type="text/css">
body { background:rgb(51, 52, 154)
       url(images/reflectionsProjections-
bg.jpg) no-repeat 0 0; }

img {
    height:400px;
    border:5px solid white;
    -webkit-box-reflect:below 5px
    -webkit-gradient(linear, left top, left
bottom, from(transparent), color-stop(0.5,
transparent), to(white));
}

</style>

```



当然，并不仅仅是图片对象可以设计倒影，在网页中任何对象都可以应用CSS Reflections。例如，下面的示例演示了如何为文本设置倒影，效果如图9.3所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS Reflections</title>
<style type="text/css">
body { background:rgb(51, 52, 154) url(images/reflectionsProjections-bg.jpg) no-repeat 0 0; }

div {
    border:2px solid white;
    color:#9C6;
    -webkit-box-reflect:below 5px;
}

h1{text-align:center;}
p{text-indent:2em;}
</style>
</head>

```

```

<body>
<div>
    <h1>倒影</h1>
    <p> 一个人总有一条可以亲近的河（或者江）。当我走近一条河，不像走向母亲，看到自己作为女儿，作为一个女人，所有的光荣和悲哀；也不像走向父亲，看到作为一种囚禁，人性的硬度和固执，仰慕和失望。 </p>
    <p> 当我走向江堤大道时，是想去江边田野里拍几张有阳光余韵的相片。桂香浮动的天空开始收敛下午升起的亮色，很快走向一日的夜。这是我回想起来的感觉。感觉往往是真实的，对于白日温度或者色彩的落差，感觉往往也特别清晰。下午乍见雨后初晴的阳光时，心完全是盲目的，而视觉是明亮的。我抬头看天，喃喃地说：太阳多好。而此刻我感觉到夜色降临，视力变得更加微弱：香喷喷的桂树，三三两两散步的人，都仿佛是梦中之景，从我眼前云一般飘走了。我的目光一直依附在小儿跃动的身体上。我的手以朋友的姿态在他肩上放了一下，给他举手轻轻一拂：像不愿意惹尘埃。我心一颤。 </p>
</div>
</body>
</html>

```



图 9.3 为文本应用倒影效果

我们还可以为视频等多媒体界面设计倒影，这样能够设计出非常动感的视觉效果。例如，在下面的案例中使用 CSS Reflections 为 `<object>` 标签应用倒影，效果如图 9.4 所示。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS Reflections</title>
<style type="text/css">
body { background:rgb(51, 52, 154) url(images/reflectionsProjections-bg.jpg) no-repeat 0 0; }

object {
    border:2px solid white;
    color:#9C6;
    -webkit-box-reflect:below 5px;
}
</style>

```

```
<script src="Scripts/swfobject_modified.js" type="text/javascript"></script>
</head>
<body>
<object id="FlashID" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="640" height="400">
    <param name="movie" value="images/flash15.swf" />
    <param name="quality" value="high" />
    <param name="wmode" value="opaque" />
    <param name="swfversion" value="6.0.65.0" />
    <!-- 此 param 标签提示使用 Flash Player 6.0 r65 和更高版本的用户下载最新版本的 Flash Player。如
果您不想让用户看到该提示，请将其删除。 -->
    <param name="expressinstall" value="Scripts/expressInstall.swf" />
    <!-- 下一个对象标签用于非 IE 浏览器。所以使用 IECC 将其从 IE 隐藏。 -->
    <!--[if !IE]>-->
        <object type="application/x-shockwave-flash" data="images/flash15.swf" width="640"
height="400">
            <!--<![endif]-->
            <param name="quality" value="high" />
            <param name="wmode" value="opaque" />
            <param name="swfversion" value="6.0.65.0" />
            <param name="expressinstall" value="Scripts/expressInstall.swf" />
            <!-- 浏览器将以下替代内容显示给使用 Flash Player 6.0 和更低版本的用户。 -->
            <div>
                <h4>此页面上的内容需要较新版本的 Adobe Flash Player。</h4>
                <p><a href="http://www.adobe.com/go/getflashplayer"></a></p>
            </div>
            <!--[if !IE]>-->
        </object>
        <!--<![endif]-->
    </object>
<script type="text/javascript">
swfobject.registerObject("FlashID");
</script>
</body>
</html>
```

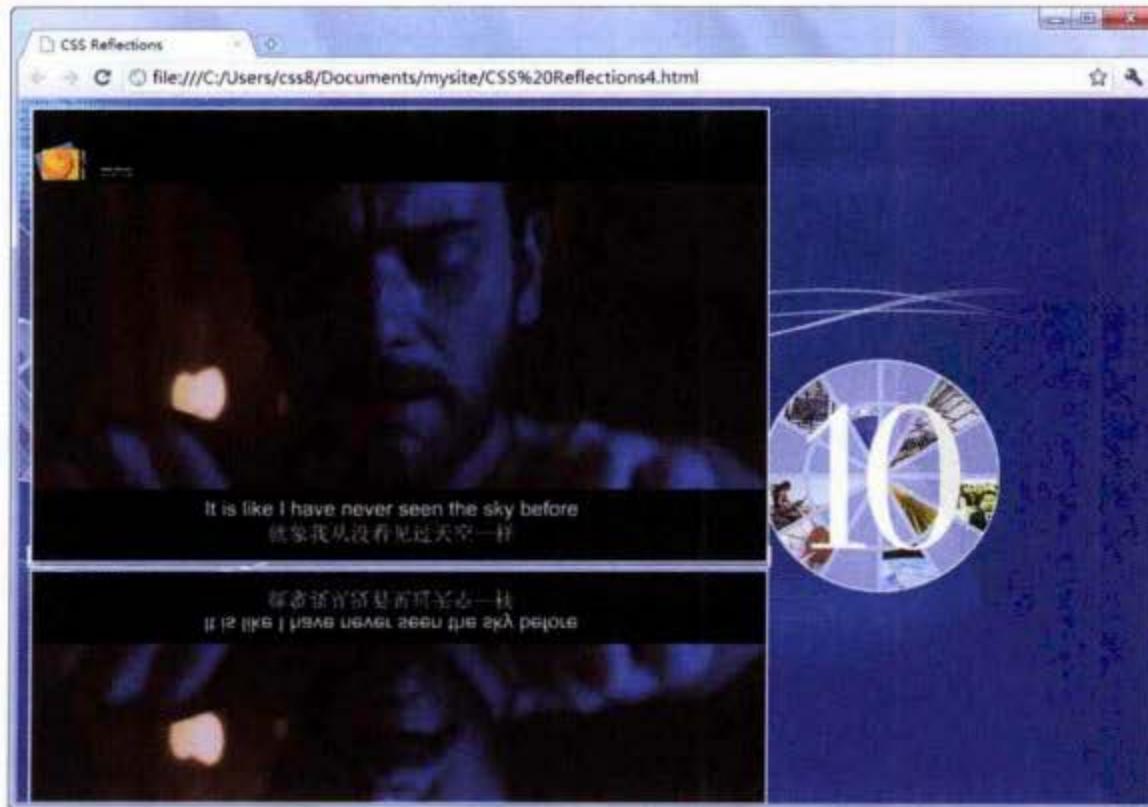


图9.4 为多媒体视频应用倒影效果

9.4 定义语音样式——CSS 3 Speech

在 CSS 2 版本时，W3C 就添加了对听觉媒体类型的支持。这项支持用来定义在听觉设备中合成语音样式。当 CSS 2.1 草案发布后，新标准又提出了 media="speech" 的特性，但是没有明确定义可以应用的属性，所以听觉媒体类型没有被认可。

2004 年 12 月，CSS 3 新推出了 CSS 3 Speech Module 语音模块，详细信息请参阅 <http://www.w3.org/TR/css3-speech/>。CSS 3 的 Speech 模块移除了一些老的属性，添加了新的属性，这些都分配在 speech 媒体类型中。

浏览器兼容性检测

目前，Opera 是实现最多 CSS 3 Speech 属性的浏览器。该浏览器通过 -xv- 前缀加以实现。各浏览器对 Speech 属性的支持情况如下。

✗ IE 6	✗ Firefox 1.5	✗ Opera 9.0	✗ Safari 3.0	✗ Chrome 2.0
✗ IE 7	✗ Firefox 2.0	✗ Opera 9.63	✗ Safari 4.0	✗ Chrome 2.0.162
✗ IE 8	✗ Firefox 3.0	✗ Opera 10.0		✗ Chrome 3.0.x
✗ IE 9	✗ Firefox 3.5	✗ Opera 10.5		✗ Chrome 4.0.x

CSS 3 Speech 的基本用法

CSS 3 Speech 定义了众多属性，详细说明如下所示。

- voice-volume：设置音量。该属性取值包括 <number>、<percentage>、silent | x-soft、soft、medium、loud、x-loud、inherit，默认值为 medium。
- voice-balance：设置声音平衡。该属性取值包括 <number>、left、center、right、leftwards、rightwards、inherit，默认值为 center。
- speak：设置阅读类型。该属性取值包括 none、normal、spell-out、digits、literal-punctuation、no-punctuation、inherit，默认值为 normal。
- pause-before, pause-after：设置暂停时的效果。该属性取值包括 <time>、none、x-weak、weak、medium、strong、x-strong、inherit，默认值为 implementation dependent。
- pause：设置暂停。该属性取值包括 [<"pause-before">||<"pause-after">]、inherit，默认值为 implementation dependent。

- rest-before, rest-after : 设置休止时的效果。该属性取值包括 <time>、none、x-weak、weak、medium、strong、x-strong、inherit，默认值为 implementation dependent。
- rest : 设置休止。该属性取值包括 [<"rest-before">||<"rest-after">]、inherit，默认值为 implementation dependent。
- cue-before, cue-after : 设置提示时的效果。该属性取值包括 <uri> [<number>、<percentage>、silent、x-soft、soft、medium、loud、x-loud]、none、inherit，默认值为 none。
- cue : 设置提示。该属性取值包括 [<"cue-before">||<"cue-after">]、inherit，无默认值。
- mark-before, mark-after : 设置标注时的效果。该属性取值为 <string>，默认值为 none。
- mark : 设置标注。该属性取值包括 [<"mark-before">||<"mark-after">]，无默认值。
- voice-family : 设置语系。该属性取值包括 [[<specific-voice>、<age>] <generic-voice>] [<number>],[]* [<specific-voice>、<age>]<generic-voice>][<number>]、inherit，默认值为 implementation dependent。
- voice-rate : 设置比率。该属性取值包括 <percentage>、x-slow、slow、medium、fast、x-fast、inherit，默认值为 implementation dependent。
- voice-pitch : 设置音调。该属性取值包括 <number>、<percentage>、x-low、low、medium、high、x-high、inherit，默认值为 medium。
- voice-pitch-range : 设置音调范围。该属性取值包括 <number>、x-low、low、medium、high、x-high、inherit，默认值为 implementation dependent。
- voice-stress : 设置重音。该属性取值包括 strong、moderate、none、reduced、inherit，默认值为 moderate。
- voice-duration : 设置音乐持续时间。该属性取值为 <time>，默认值为 implementation dependent。
- phonemes : 设置音位。该属性取值为 <string>，默认值为 implementation dependent。

实战体验：体验 CSS 3 Speech 应用

下面我们看一个简单的示例，体验 CSS 3 Speech 的使用。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS3 Speech</title>
<style type="text/css">
#speech-sample {
    voice-family: male;
    border: 1px solid #666;
    padding: 5px;
    font-size: 14px;
}
```

```
}

#voice-volume {
    -xv-voice-volume: x-soft;
    -xv-voice-balance: right;
}

#voice-balance { -xv-voice-balance: left; }
#speech-cue { cue-after: url(images/ding.wav); }
#voice-rate { -xv-voice-rate: x-slow; }
#voice-family { voice-family: female; }
#voice-pitch { -xv-voice-pitch: x-low; }
#speech-speak { speak: spell-out; }
</style>
</head>

<body>
<div>
    <div id="speech-sample">这是一个示例，演示CSS3 Speech的应用，包括
        <span id="voice-volume">voice-volume、</span>
        <span id="voice-balance">voice-balance、</span>
        <span id="voice-rate">voice-rate、</span>
        <span id="voice-family">voice-family、</span>
        <span id="voice-pitch">voice-pitch、</span>
        <span id="speech-speak">speak</span> 和
        <span id="speech-cue">cue-after。</span>
    </div>
</div>
</body>
</html>
```



虽然CSS 3的标准制定工作早在10年前就开始了，但直到近一两年才逐渐开始得到各主流浏览器的支持。与CSS 2.x相比，CSS 3在各方面都有非常大的改进，不仅功能更强大，而且也更便于Web前端工作者使用。如果你是一位前端工作者，或将来打算进入前端领域，建议你从现在就开始学习CSS 3。本书不仅全面而系统地讲解了CSS 3的所有新功能和新特性，而且还精心设计了数十个经典的案例，实战性极强。本书将是当下深入而系统学习CSS 3的最佳选择！

——CSS 3研究小组

HTML 5和CSS 3是Web开发领域当下最热门的话题之一，二者必定会掀起一场革命，未来的Web世界将由它们来主宰。本书可谓极具前瞻性，而且出版时机也恰到好处，应该能为CSS 3在国内的发展和普及做出一定的贡献。它不仅在与CSS 2.x进行对比的基础上全面讲解了CSS 3的方方面面，而且还包含大量的实战案例和最佳实践。此外，本书全彩印刷，装帧和版式设计都非常精美，不仅有学习价值，而且还有收藏价值。强烈推荐。

——CSS开发者社区

随着各种新型Web应用的出现，以及用户对用户体验的要求的不断提高，各大主流浏览器都开始将CSS 3作为一种事实标准的解决方案，相应地，CSS 3自然而然也将成为Web前端工作者的必修课之一。如果你是一位有远见的前端工作者，那就从现在开始学习和实践CSS 3吧，相信这本书会给你惊喜。

——HTML51 (www.html51.com)

作者简介

成林 资深Web前端工程师，从事Web前端工作多年，精通CSS、HTML、JavaScript、jQuery和Ajax等Web前端技术，在实践中积累了大量的经验。推崇Web技术标准，曾经在多所高等院校和一些线下技术沙龙主讲Web标准和规范相关的课程，曾经还参与过W3C组织的标准化文档的中文编译工作。近两年来，集中精力研究和实践CSS 3和HTML 5前沿技术，是国内该领域的先驱者之一，已经有较为深入的认识和丰富的实践经验。

上架指导：计算机 / 程序设计 / Web开发

ISBN 978-7-111-34155-0

9 787111 341550



客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 <http://www.hzbook.com>

网上购书：www.china-pub.com