HZ BOOKS
华章科技

Web开发
技术丛书

图解CSS3

核心技术与案例实战

大漠 著

Illustrated CSS3
Details and Cases

- 资深Web前端专家历时两载的经验与心血之作，旨在根据最新CSS3规范撰写最权威的CSS3学习资料和备查手册
- 理论知识系统且全面，以图解的方式讲解CSS3的各项功能和特性，包含大量实战案例，直观易懂，实战性强

机械工业出版社
China Machine Press

Web 开发技术丛书

图解 CSS3
核心技术与案例实战

大 漠 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

图解 CSS3：核心技术与案例实战 / 大漠著 . —北京：机械工业出版社，2014.7
（Web 开发技术丛书）

ISBN 978-7-111-46920-9

I. 图…　II. 大…　III. 网页制作工具　IV. TP393.092

中国版本图书馆 CIP 数据核字（2014）第 116144 号

图解 CSS3：核心技术与案例实战

大　漠　著

出版发行：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：杨福川　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2014 年 7 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：32.25（含彩插 1.25 印张）
书　　号：ISBN 978-7-111-46920-9　　定　　价：79.00 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　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

Preface 前　言

为什么要写这本书

CSS3 是在 CSS2.1 基础上扩展而来，事实上，它还没有完全成熟。有些专家会告诉你，CSS3 现在还用不上，甚至几年之后都不会有成熟的规范发布。

目前为止 CSS3 还没有一套成熟的规范，其中的模块也在不断更新，特别是浏览器对 CSS3 特性的支持也在不断变化，同时没有足够的时间去学习和研究 W3C 官方文档和规范，致使我们学习 CSS3 变得更为复杂。

为什么会选择这个时候编写这样一本图书呢？原因很简单。对于希望 Web 应用开发者而言，CSS3 可以说是众望所归，这也是技术变更的硬性需求。在实际 Web 应用中新标准的采纳程度正在以令人目眩的速度不断地变更着，众多浏览器厂商也在不断加快对 CSS3 新特性的支持。在编写这本图书的过程中，我也被迫不断更新书中的浏览器支持表格。

面对自己正在使用的浏览器，大多数用户并不真正了解其具备的功能有多强大。当然，他们在浏览器自动更新后可能会发现一些细微的界面变化。但他们可能不知道，新版的浏览器对哪些 CSS3 特性有所支持。

本书的目标是帮助开发者更好地掌握 CSS3 的特性，并且将新技术运用到实际的开发当中，提高自己开发 Web 程序的水平。

本书面向的读者

❏ 有一定 CSS3 开发经验的前端工程师。

本书能帮助你系统掌握 CSS3 的各项知识，提升技术水平和业务能力。

❏ 从事 CSS3 开发的前端工程师。

由于 CSS3 涵盖的新特性非常多，在开发过程中将本书作为速查手册，提高开发效率。

❏ 前端开发爱好者。

如果还不是一名前端工程师，但是对前端开发非常感兴趣，本书也能让你对最新的 CSS 标准和规范有一个系统和全面的认识，为学习前端知识打下基础。

本书的特色

本书最大的特色就是将 CSS3 特性按模块功能分类，通过理论、图解、实战的方式向大家阐释 CSS3 每个特性功能。

❏ 内容全面、丰富、翔实。

由浅到深地讲解了 CSS3 新特性的语法、特性以及使用技巧。本书涵盖了 CSS3 众多功能模块，如 CSS3 选择器特性、边框模块、文本模块、颜色模块、UI 界面模块、CSS3 动画模块、CSS 新型盒模型以及 CSS 媒体查询、响应式设计等。

❏ 图解方式，直观易懂。

图解的方式是本写的最大特色之一，在描述每一个 CSS3 特性过程都配了生动的实战效果，甚至每一步骤都配有相应的效果图。就算是你对文字理解或者代码理解有所误差，实战效果图能辅助你更好地理解 CSS3 每个特性。

❏ 案例丰富，实战性强。

每个 CSS3 特性都配有实战体验，部分案例来自于实际开发之中。同时在每个知识点之后，还提供了综合案例。通过实践加强动手能力，更好地掌握 CSS3 中的每个知识点。

动手实践才是掌握一门新技术最有效的途径。如果能在阅读本书的过程中逐一亲手实现这些案例，那么在以后的实际开发中自然就会具有相当强的动手能力了。

本书的内容

本书包括 15 章，通过实例来演示 CSS3 模块的新特性。

第 1 章简单介绍什么是 CSS3，CSS3 的好处是什么，浏览器对 CSS3 的支持状况，以及 CSS3 带来什么新特性，并且引入渐进增强式的概念。通过对本章的学习，大家可以在一定的程度上知道一些 CSS3 的故事。

第 2 章介绍 CSS3 选择器。选择器是 CSS 中的核心部分之一，本章先阐述 CSS2 的选择器，再引入 CSS3 新增的选择器。深入介绍了 CSS3 新增选择器的功能及其实用性，还有各浏览器的兼容性。

第 3 章详细介绍 CSS3 在边框方面新增的功能特性，比如边框色、图片边框、边框圆角等，并与 CSS2 进行了对比。

第 4 章介绍 CSS3 背景功能，着重阐述了多背景、背景尺寸、背景原点方面的使用，以让

大家掌握如何使用 CSS3 背景功能的新特性。

第 5 章介绍 CSS3 文本功能。以前大家在网页制作时，只是设置文本的颜色、字体、字号等。通过对 CSS3 文本功能的学习，大家还可以运用文本阴影、文本溢出、文本换行等功能。

第 6 章介绍 CSS3 颜色特性。大家以前只有在设计软件中使用的颜色值现在都可以运用，如 RGBA、HSL、HSLA、透明度等。

第 7 章介绍 CSS3 基础盒模型与用户界面。盒模型是 CSS 的重中之重，CSS2 盒模型功能只能实现一些基本功能，对于一些特殊的功能需要借助 JavaScript 来实现。而在 CSS3 中这一点将得到很大的改善，可以通过 CSS3 来直接实现一些特殊的功能。

第 8 章介绍 CSS3 的弹性盒模型，给大家引入一种全新的布局概念，为大家的页面布局带来革命性的变化。

第 9 章介绍 CSS3 多列布局。布局在 Web 中随处可见，多列布局在 CSS2 中都是依靠 float 或者 inline-block 来实现的，而这两个属性带来的局限性也是相当大的。CSS3 多列布局将会弥补这些不足之处。

第 10 章介绍 CSS3 渐变功能。渐变效果在 Web 中也是一种常见的效果，以前靠设计师制作图片来完成，不仅增加了设计师的工作量，在页面中的效果也带来过多的局限制，扩展性也相当差。CSS3 渐变不再需要使用图片来代替这些特殊的效果。

第 11 章介绍 CSS3 变形功能。这是一个全新的功能，在 CSS2 中要实现需要借助 JavaScript。CSS3 的变形功能可以直接使用样式实现如旋转、移位、扭曲、缩放等效果。

第 12 章介绍 CSS3 过渡功能。大家在 Web 制作中，使过渡效果不再生硬，变得细腻、流畅。

第 13 章介绍 CSS3 动画功能。

第 14 章介绍 Media Query 与 Responsive 布局。随着移动设备和宽屏浏览器的普及，单一的设计不能满足 Web 页面的设计需求，此时 CSS3 的 Media Query 新特性中出现了一个新的布局概念——Responsive。本章中大家将体会到 Media Query 与 Responsive 布局的强大功能。

第 15 章介绍嵌入 Web 字体。浏览器仅限于用户在其系统上安装的字体呈现文本。CSS3 使用 @font-face 改变了这一格局。网站不再受限于少量字体，如 Arial、Verdana、Times 和 Georgia 等。

如何阅读本书

本书结构不是按层进式安排的，章节之间是按 CSS3 的模块分类，读者阅读本书时无须按照先后顺序进行，可以挑选自己喜欢的章节阅读。但如果按章节的编排顺序逐章阅读，会更

系统、更全面地学习 CSS3，从中获得最大受益。

阅读本书的案例时，尽量不要照抄书中的代码，在理解案例的设计思路基础上，自己动手开发相似功能的应用，并创造出满足自己需求的功能，举一反三。

本书中使用的约定

本书案例已在主流浏览器上进行过测试了。分别是：Firefox 12.0、Google Chrome 19.0.1084.52、Safari 5.17、Opera 11.64、IE 9。

同时在一些广泛使用的旧版本浏览器（如 IE 8）上也做了测试。很多情况下，CSS3 的效果也能体现在较低版本上，页面能保持正常阅读，而且效果也不会太差。对于每一个 CSS3 特性，将尽可能地为低版本浏览器寻求变通的备用方案，使之能兼容那些不被原生支持的浏览器。

针对每个浏览器版本，我们会标注相对应的属性在哪个版本号中开始支持。一些 CSS3 特性需要添加相应浏览器的渲染引擎的前缀才会生效，我们将会在后面的章节中依次介绍各浏览器的渲染引擎的前缀名称，以及 CSS3 特性在对应浏览器下的写法。

在阅读本书时有些约定，有必要在这里先说明。

- W3C 表示万维网联盟（World Wide Web Consortium），是制定 Web 官方标准和规范（如 CSS3）的组织。
- 初始值（即默认值）是用户不显式声明时元素所具有的属性值。需特别指明的是，属性是元素的本质，而不是用户自定义的属性。
- IE 8 及以下版本代表 IE 8、IE 7 和 IE 6。
- Webkit 引擎内核的浏览器是指 Safari（包括移动版本和桌面版本）、Google Chrome 和其他近期使用版本的 Webkit 页面渲染引擎的浏览器，其私有属性的前缀是 -webkit-。
- Gecko 引擎内核的浏览器是指 Mozilla，常指的是 Firefox 浏览器，其私有属性的前端缀是 -moz-。
- Presto 引擎内核的浏览器是指 Opera，其私有属性的前缀是 -o-。
- KHTML 引擎内核的浏览器是指 Konqueror，其私有属性的前缀是 -khtml-。
- Trident 引擎内核的浏览器是指 Internet Explorer，其私有属性的前缀是 -ms-。
- 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 系统和移动端。
- 偶尔会碰到“所有浏览器”这个说法，此时仅代表目前所有广泛使用的浏览器，而并非字面意义所涵盖的那些可能仅占零星市场份额的不知名的浏览器。
- “HTML”指 HTML 和 XHTML 这两种语言。

- “CSS”指 CSS2.1 规范，除非特别声明。
- 本书所有案例代码都是以 HTML 5 的 DTD 编写。但这仅仅表示使用短小精悍的 HTML 5 文档声明 <!DOCTYPE html>，还有更简洁的 meta 字符编码、style 和 script 标签。没有使用任何 HTML 5 的新标签，比如 section、header、nav 和 article，所以页面可以在 IE 8 及以下版本正常运行，可以在自己的页面里将其更换为喜欢的标签。所有示例也同样兼容 HTML 4.01 和 XHTML 1.0。
- 为了方便阅读，本书中的部分案例代码仅提供了 CSS 样式代码和局部 HTML 代码，所有 CSS 实例代码必须置于一个外部样式文件或 HTML 文档的 <head></head> 标签内。
- 由于 CSS3 技术还在不断的完善与更新中，建议根据本书提供的参考地址，获取有关 CSS3 最新信息与更新。

勘误和支持

由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，我特意创建了一个在线支持站点 http://www.w3cplus.com/book-comment.html。大家可以将书中的错误发布在页面的评论中，遇到任何问题，可以留言或者发送邮件到 w3cplus@hotmail.com，我将尽量提供最满意的答案。大家还可以关注微信公众账号 ednote 进入“第三极社区”微社区与广大读者和本书作者互动。书中的全部源文件可以从华章网站（http://www.hzbook.com）下载，我也会将相应的功能及时更正。期待能够得到你们真挚反馈。

致谢

首先要感谢好友林小志，是他让我鼓起勇气开始写这本书，也是他一直督促我的进度，并一直鼓励我坚持到最后。同时感谢 W3CPlus（http://www.w3cplus.com）社区的所有同学们一直以来对我的默默支持。

感谢机械工业出版社的编辑杨福川给我这样一个机会，在一年多的时间中始终支持我的写作，你的鼓励和帮助引导我能顺利完成全部书稿。同时也要感谢白宇编辑辛苦的付出，帮助我修改书中不足。

感谢我的爸爸、妈妈将我培养成人，并时时刻刻为我灌输着信心和力量！也要感谢我的弟弟，引导我进入这个行业，让我有机会从事喜欢的工作。感谢太太罗群英和儿子一直以来对我的支持，让我有一个安心写作的环境，并给我足够的信心去完成这本拙作。

谨以此书献给我最亲爱的家人、朋友以及众多热爱 W3CPlus 社区的朋友们！

HZ BOOKS
华章图书

Contents 目 录

第 1 章 Chapter 1

揭开 CSS3 的面纱

如果关注前端方面的技术，那么对 CSS 一定不会陌生，你肯定听说过 CSS3。在使用 CSS3 之前，应该对这个新一代样式表语言的来龙去脉有个基本了解。

在本章中，你将知道 CSS3 与 CSS2.1 的区别，以及当前市面上主流浏览器、移动端浏览器对 CSS3 支持的情况。对于尚不完全支持 CSS3 的浏览器，将会为大家引入一个渐进增强的概念，用一些 CSS 方法来模拟 CSS3 的实现方法。最后给大家简单介绍一些 CSS3 引入的新特性及其未来的前景。

1.1 什么是 CSS3

CSS3 并不是一门新的语言。如果接触过 CSS 就知道，CSS 是创建网页的另一个独立但并非不重要的一部分。CSS 是种层叠样式表，是一种样式语言，用来告诉浏览器如何渲染你的 Web 页面。

CSS3 是 CSS 规范的最新版本，在 CSS2.1 的基础上增加了很多强大的新功能，以帮助开发人员解决一些问题，并且不再需要非语义标签、复杂的 JavaScript 脚本以及图片，例如圆角功能、多背景、透明度、阴影等功能等。CSS2.1 是单一的规范，而 CSS3 被划分成几个模块组，每个模块组都有自己的规范。这样的好处是整个 CSS3 的规范发布不会因为部分难缠的部分而影响其他模块的推进。[⊖]

现在先来看看 CSS3 激动人心的新特性。

⊖ 更详细的信息可参见 http://www.w3.org/Style/CSS/current-work.en.html，其中介绍了 CSS3 具体划分为多少个模块组、CSS3 所有模块组目前所处的状态，以及将在什么时候发布。

1.1.1 CSS3 的新特性

CSS3 规范并不是独立的，它重复了 CSS 的部分内容，但在其基础上进行了很多的增补与修改。CSS3 与之前的几个版本相比，其变化是革命性的，虽然它的部分属性还不能够被浏览器完美的支持，但却让我们看到网页样式发展的前景，让我们更具有方向感、使命感。

CSS3 新特性非常多，这里挑选一些被浏览器支持较为完美、更具实用性的新特性。

1. 强大的 CSS3 选择器

使用过 jQuery 的人都知道，jQuery 的选择器功能强大，使用方便，CSS3 选择器和 jQuery 选择器非常类似。允许设计师通过选择器直接指定需要的 HTML 元素，而不需要在 HTML 中添加不必要的类名、ID 等。使用 CSS3 选择器，能在 Web 的制作中更完美地做到结构与表现分离，设计师能轻松地设计出简洁、轻量级的 Web 页面，并且能更好地维护和修改样式。

2. 抛弃图片的视觉效果

Web 中最常见的效果包括圆角、阴影、渐变背景、半透明、图片边框等。而这样的视觉效果在 CSS 中都是依赖于设计师制作图片或者 JavaScript 脚本来实现的。CSS3 的一些新特性可以用来创建一些特殊的视觉效果，后面的章节将为大家展现这些新特性是如何实现这些视觉效果。

3. 背景的变革

如果说 CSS 中的背景给你带来太多的限制，那么 CSS3 将带来革命性的变化。CSS3 不再局限于背景色、背景图像的运用，新特性中添加了多个新的属性值，例如 background-origin、background-clip、background-size，此外，还可以在一个元素上设置多个背景图片。这样，如果要设计比较复杂的 Web 页面效果，就不再需要使用一些多余标签来辅助实现了。举个例子，要实现 CSS 中的滑动门效果，在 CSS 中基本上要添加 2 ～ 3 个额外的标签来辅助实现，那么 CSS3 中的这些新特性能够在一个标签中完成同等的效果。

4. 盒模型变化

盒模型在 CSS 中是重中之重，CSS 中的盒模型只能实现一些基本的功能，对于一些特殊的功能需要基于 JavaScript 来实现。而在 CSS3 中这一点得到了很大的改善，设计师可以直接通过 CSS3 来实现。例如，CSS3 中的弹性盒子，这个属性将给大家引入一种全新的布局概念，能轻而易举实现各种布局，特别是在移动端的布局，它的功能更是强大。大家将在第 7 章、第 8 章见识它的神功。

5. 阴影效果

阴影主要为分两种：文本阴影（text-shadow）和盒子阴影（box-shadow）。文本阴影在 CSS 中已经存在，但没有得到广泛的运用。CSS3 延续了这个特性，并进行了新的定义，该属性提供了一种新的跨浏览器方案，使文本看起来更醒目。盒子阴影的实现在 CSS 中就有点苦不堪言，为了实现这样的效果，需要新增标签、图片，而且效果还不一定完美。CSS3

的 box-shadow 将打破这种局面，可以轻易地为任何元素添加盒子阴影。

6. 多列布局与弹性盒模型布局

CSS3 引入了几个新的模块用于更方便地创建多列布局。

“多列布局”（Multi-column Layout）模块描述了如何像报纸、杂志那样，把一个简单的区块拆分成多列，第 9 章为大家介绍这个模块的运用。“弹性盒模型布局”（Flexible Box Layout）模块能让区块在水平、垂直方向对齐，能让区块自适应屏幕大小，相对于 CSS 的浮动布局、inline-block 布局、绝对定位布局来说，它显得更加方便与灵活。缺点是，这两个模块在一些浏览器中还不被支持，但随着技术的发展革新，这一刻终将到来。

7. Web 字体和 Web Font 图标

浏览器对 Web 字体有诸多限制，Web Font 图标对于设计师来说更奢侈。CSS3 重新引入 @font-face，对于设计师来说无疑是件好事。@font-face 是链接服务器上字体的一种方式，这些嵌入的字体能变成浏览器的安全字体，不再担心用户没有这些字体而无法正常显示的问题，从此告别用图片代替特殊字体的设计时代。

8. 颜色与透明度

CSS3 颜色模块的引入，实现了制作 Web 效果时不再局限于 RGB 和十六进制两种模式。CSS3 增加了 HSL、HSLA、RGBA 几种新的颜色模式。在 Web 设计中，能轻松实现某个颜色变得再亮一点或者再暗一点。其中 HSLA 和 RGBA 还增加了透明通道，能轻松地改变任何一个元素的透明度。另外，还可以使用 opacity 属性来制作元素的透明度。从此制作透明度不再依赖图片或者 JavaScript 脚本了。

9. 圆角与边框的新法

圆角是 CSS3 中使用最多的一个属性，原因很简单：圆角比直线性更美观，而且不会与设计产生任何冲突。与 CSS 制作圆角不同之处是，CSS3 无须添加任何标签元素与图片，也不需借用任何 JavaScript 脚本，一个属性就能搞定。对于边框，在 CSS 中仅局限与边框的线型、粗细、颜色的设置，如果需要特殊的边框效果，只能使用背景图片来模仿。CSS3 的 border-image 属性使元素边框的样式变得丰富起来，还可以使用该属性实现类似 background 的效果，对边框进行扭曲、拉伸和平铺等。

10. 盒容器的变形

在 CSS 时代，让某个元素变形是一个可望而不可及的想法，为了实现这样的效果，需要写大量的 JavaScript 代码。CSS3 引进了一个变形属性，可以在 2D 或者 3D 空间里操作盒容器的位置和形状，例如旋转、扭曲、缩放或者移位。我们把这些效果称为“变形”，大家将在第 11 章体验这些新特性。

11. CSS3 过渡与动画交互效果

CSS3 的“过渡”（transition）特性能在 Web 制作中实现一些简单的动画效果，让某些效果变得更具流线性、平滑性。而 CSS3“动画”（animation）特性能够实现更复杂的样式变化，

以及一些交互效果，而不需要使用任何 Flash 或 JavaScript 脚本代码。

12. 媒体特性与 Responsive 布局

CSS3 的媒体特性可以实现一种响应式（Responsive）布局，使布局可以根据用户的显示终端或设备特征选择对应的样式文件，从而在不同的显示分辨率或设备下具有不同的布局渲染效果，特别是在移动端上的实现更是一种理想的做法。

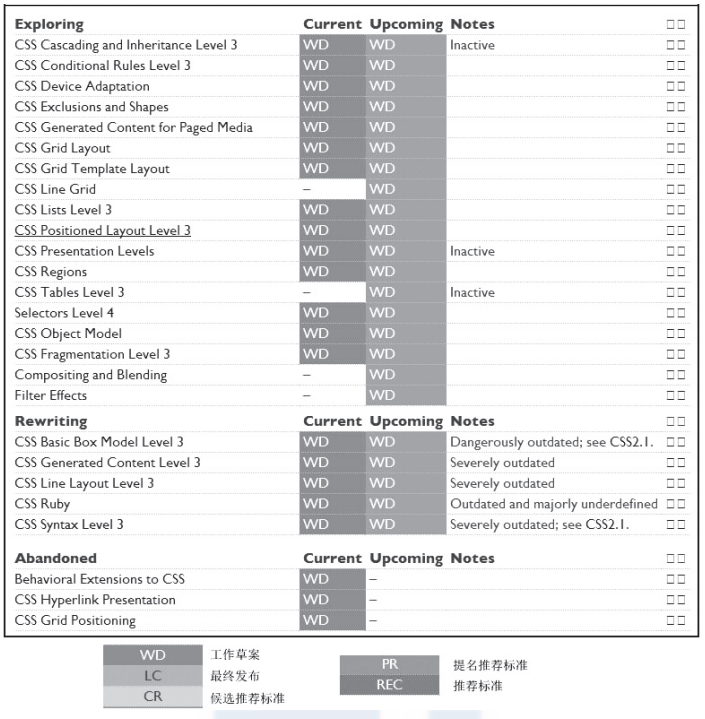
1.1.2 CSS3 的发展状况

通过对 CSS3 新特性的简单介绍，大家可能要问，这些超炫的特性什么时候才能成为标准并最终发布呢？其实 CSS3 的每一个模块都有它自己的更新（进度表）时间，如图 1-1 所示[⊖]，大家可以从这个图上看到 CSS3 的当前发展的详细进度。

Completed	**Current**	**Upcoming**	**Notes**	□□
CSS Snapshot 2010	NOTE	–	Latest stable CSS	□□
CSS Snapshot 2007	NOTE	–		□□
CSS Color Level 3	REC	REC	See Errata	□□
CSS Namespaces	REC	REC		□□
Selectors Level 3	REC	REC		□□
CSS Level 2 Revision 1	REC	REC	See Errata	□□
CSS Level 1	REC	–	Unmaintained, see Snapshot	□□
Stable	**Current**	**Upcoming**	**Notes**	□□
Media Queries	PR	REC		□□
CSS Style Attributes	CR	PR		□□
Testing	**Current**	**Upcoming**	**Notes**	□□
CSS Backgrounds and Borders Level 3	CR	PR		□□
CSS Image Values and Replaced Content Level 3	CR	PR		□□
CSS Marquee	CR	PR		□□
CSS Multi-column Layout	CR	CR		□□
CSS Speech	CR	PR		□□
CSS Mobile Profile 2.0	CR	PR	Status unknown	□□
CSS TV Profile 1.0	CR	?	Status unknown	□□
Refining	**Current**	**Upcoming**	**Notes**	□□
CSS Transforms	WD	WD		□□
CSS Transitions	WD	WD		□□
CSS Values and Units Level 3	LC	CR		□□
CSS Print Profile	LC	?	Status unknown	□□
Revising	**Current**	**Upcoming**	**Notes**	□□
CSS Animations	WD	WD		□□
CSS Flexible Box Layout	WD	WD		□□
CSS Fonts Level 3	WD	LC		□□
CSS Paged Media Level 3	LC	LC	Inactive	□□
CSS Text Level 3	WD	WD		□□
CSS Basic User Interface Level 3	CR	LC		□□
CSS Writing Modes Level 3	WD	WD		□□
CSSOM View	WD	WD		□□

图 1-1　CSS3 所有模块发展进度

⊖ http://www.w3.org/Style/CSS/current-work.en.html 上的截图。还可以到地址 http://meiert.com/en/indices/css-properties/（CSS 各属性查询表）查看各个 CSS 属性属于哪个 CSS 版本，以及各个属性对应的默认值，以便更清楚地知道哪些属性是在 CSS 基础上添加的。



Exploring	Current	Upcoming	Notes	□□
CSS Cascading and Inheritance Level 3	WD	WD	Inactive	□□
CSS Conditional Rules Level 3	WD	WD		□□
CSS Device Adaptation	WD	WD		□□
CSS Exclusions and Shapes	WD	WD		□□
CSS Generated Content for Paged Media	WD	WD		□□
CSS Grid Layout	WD	WD		□□
CSS Grid Template Layout	WD	WD		□□
CSS Line Grid	–	WD		□□
CSS Lists Level 3	WD	WD		□□
CSS Positioned Layout Level 3	WD	WD		□□
CSS Presentation Levels	WD	WD	Inactive	□□
CSS Regions	WD	WD		□□
CSS Tables Level 3	–	WD	Inactive	□□
Selectors Level 4	WD	WD		□□
CSS Object Model	WD	WD		□□
CSS Fragmentation Level 3	WD	WD		□□
Compositing and Blending	–	WD		□□
Filter Effects	–	WD		□□
Rewriting	**Current**	**Upcoming**	**Notes**	□□
CSS Basic Box Model Level 3	WD	WD	Dangerously outdated; see CSS2.1.	□□
CSS Generated Content Level 3	WD	WD	Severely outdated	□□
CSS Line Layout Level 3	WD	WD	Severely outdated	□□
CSS Ruby	WD	WD	Outdated and majorly underdefined	□□
CSS Syntax Level 3	WD	WD	Severely outdated; see CSS2.1.	□□
Abandoned	**Current**	**Upcoming**	**Notes**	□□
Behavioral Extensions to CSS	WD	–		□□
CSS Hyperlink Presentation	WD	–		□□
CSS Grid Positioning	WD	–		□□

图 1-1 （续）

Web 开发者希望在 CSS3 标准规范发布之前就能使用这些新特性，而它们的使用还受限于不同的浏览器，只有浏览器完全支持了，才能完全使用这些新特性。

目前，CSS3 还不是最终的标准，有很多浏览器支持不够完美，那么现在可以使用 CSS3 吗？

1.1.3　现在能使用 CSS3 吗

从图 1-1 中可以看出，CSS3 还在不断完善中，很多功能还处于草稿阶段，但部分模块进入了“候选推荐”状态，这说明在 Web 设计中完全可以使用这些模块。即使有一些模块还处于“工作草案状态”，也可以尝试着使用，只有不断将新的 CSS 技术运用到实际工作中，才能发现应用这些新技术所面临的真正挑战，以便 W3C 更好地完善它们，从而更好地、有效地促使它们成为真正的标准。

你应该了解哪些可用，哪些还不能使用。换句话说，在实际工作开发中可以先运用相对稳定的 CSS3 特性，并确保不会对尚不支持这些特性的浏览器造成影响。做到明智的使用，而不是盲目地滥用 CSS3 新特性。

1.1.4　使用 CSS3 有什么好处

与 CSS 相比，使用 CSS3 有什么好处呢？最明显的就是 CSS3 能让页面看起来非常炫、

非常酷，使网站设计锦上添花，但它的好处远远不止这些。在大多数情况下，使用 CSS3 不仅有利于开发与维护，还能提高网站的性能。与此同时，还能增加网站的可访问性、可用性，使网站能适配更多的设备，甚至还可以优化网站 SEO，提高网站的搜索排名结果。下面介绍 CSS3 特有的好处。

1. 减少开发与维护成本

为什么说 CSS3 能减少开发与维护的成本呢？先来看一个实例。一个圆角效果，在 CSS 中需要添加额外的 HTML 标签，使用一个或者更多图片来完成，而使用 CSS3 只需要一个标签、一个 border-radius 属性就能完成。这样，CSS3 技术能把你从绘图、切图和优化图片的工作中解救出来。

如果后续需要调整这个圆角的弧度或者圆角的颜色，使用 CSS，要从头绘图、切图才能完成，而使用 CSS3 几秒就完成这些工作。

CSS3 还能使你远离一大堆的 JavaScript 脚本代码或者 Flash，你不再需要花大把时间去写脚本或者寻找合适的脚本插件并修改以适配你的网站特效。

最后，有些 CSS3 技术还能帮你对页面进行“减肥”，让结构更加“苗条”。你不用为了达到一个效果而嵌套很多 DIV 和类名，这样能有效地提高工作效率、减少开发时间、降低开发成本。例如，制作一个重叠的背景效果，在 CSS 中需要添加 DIV 标签和类名，在不同的 DIV 中放一张背景图，现在可以使用 CSS3 的多背景和背景尺寸等新特性，在一个 DIV 标签中完成这些工作。

2. 提高页面性能

很多 CSS3 技术通过提供相同的视觉效果而成为图片的“替代品”，换句话说，在进行 Web 开发时，减少多余的标签嵌套，以及图片的使用数量，意味着用户要下载的内容将会更少，页面加载也会更快。另外，更少的图片、脚本和 Flash 文件让 Web 站点减少 HTTP 请求数，这是提升页面加载速度的最佳方法之一。而使用 CSS3 制作图形化网站无需任何图片，极大地减少 HTTP 的请求数量，并且提升页面的加载速度。当然，这取决于采用 CSS3 特性来代替什么技术，同样还要看如何使用 CSS3 特性。例如 CSS3 的动画效果，能够减少对 JavaScript 和 Flash 文件的 HTTP 请求，但可能要求浏览器执行很多的工作来完成这个动画效果的渲染，这有可能导致浏览器响应缓慢，致使用户流失。因此，在使用一些复杂的特效时，大家需要考虑清楚。不过这样的现象毕竟为数不多。其实很多 CSS3 技术能够在任何情况下都大幅提高页面的性能。就这一条足以让我们使用 CSS3。

1.2 浏览器对 CSS3 的支持状况

CSS3 给我们带来众多全新的设计体验，但有一个问题值得考虑——浏览器对 CSS3 特

性的支持情况如何？因为页面最终离不开浏览器来渲染，并不是所有浏览器都完全支持 CSS3 的特性。有时候花时间写的效果只能在特定的浏览器下有效，意味着只有部分用户能欣赏到，这样的工作变得没有什么意义。例如，使用 CSS3 制作背景仅在 WebKit 内核的浏览器下有效果。

想知道用户能够体验到哪些 CSS3 的新特性，必须了解当前浏览器对 CSS3 特性的支持程度如何。

1.2.1　经典回顾：图说浏览器大战

“浏览器大战”一词在 20 世纪末产生，网景（Netscape）与微软展开了第一次互联网大战，结果是网景以失败告终，微软荣登冠军宝座。

2004 年 11 月 Firefox 1.0 诞生，浏览器开始了历史上的第二次大战，IE 的地位受到了以 Firefox 为首的其他浏览器的挑战。2008 年 12 月 Google Chrome 的诞生，向市场投放了一颗重磅炸弹。此时的 IE 也开始了版本的升级，虽然 IE 将版本更新到了 IE 8，但面对 Firefox 和 Google Chrome 两个强劲的对手，其更新的步伐依然显得太慢，在 2010 年 IE 的市场份额跌至 50%。而后，Chrome 不断更新，其市场份额快速上升。2012 年 5 月，终于夺得浏览器的霸主之位。

这不是浏览器大战的结束，仅仅是 IE 时代在落幕而已。随着移动设备的风靡，移动版本 Safari 的市场份额在一年的时间里迅速增长。也许，第三次浏览器大战的战场并不在桌面领域，而是在移动领域。

市面上浏览器品种繁多，从而引发浏览器的市场大战，这场战争持续了近二十年，但从未有结束战争的趋势。浏览器之争提示了 Web 浏览器的影响，比如 Chrome 和 Firefox 对浏览器霸主 IE 发起的挑战，随着移动终端的出现，另一个强有力的竞争者——移动 Safari 网络浏览器也加入这场无休止的浏览器之战。下面一起来看看 monetate.com[⊖]绘制的浏览器战争图，如图 1-2 所示。

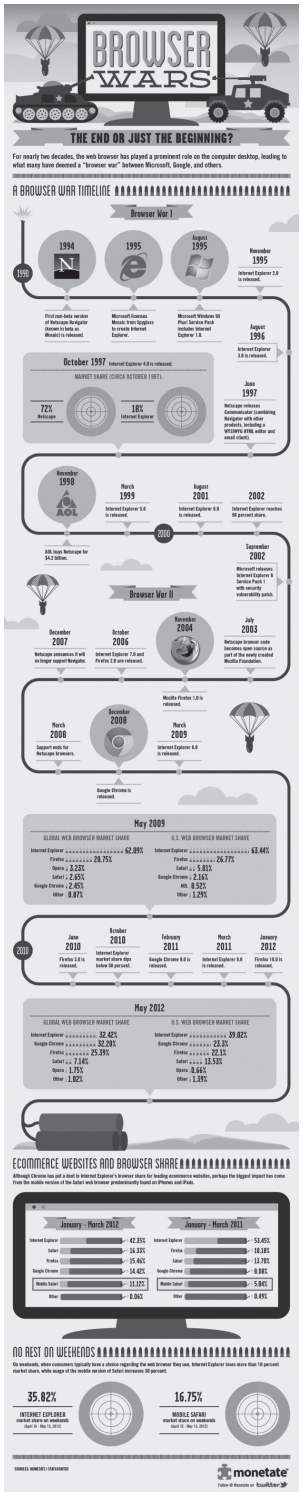


图 1-2　浏览器之争

⊖ 详情见 http://monetate.com/infographic/browser-wars-the-end-or-just-the-beginning。

1.2.2 浏览器的市场份额

图 1-2 所示只是主流浏览器的市场之争，国内还有许多国产的浏览器，例如 QQ 浏览器、奇虎 360 浏览器、移动端的 UC 浏览器等。用户在使用什么样的浏览器，这个使用率始终无法准确地掌握，因为这个概率始终都在变化，下面详细看看国内和全球浏览器的市场份额。

1. 浏览器国内市场份额

首先关注国内浏览器的市场份额，一起来看百度统计的浏览器市场份额图，如图 1-3 所示。

国内浏览器市场位列三甲的分别是 IE 6.0、奇虎 360 和 IE 7.0，三个版本的浏览器流量份额占据总市场份额的 61.1% 左右，但 IE 6 ～ 8 在国内依然处于绝对领先态势，但相比两年前，IE 浏览器在国内也步入下滑的态势，这给前端开发人员带来一丝的希望。更值得庆幸的是，360 浏览器在 5 月发起了狙击 IE 6 浏览器的活动，并开始在最新版本的 360 安全浏览器中内置了 IE 8 内核，这无疑给国内的前端工程师带来了一丝清凉。

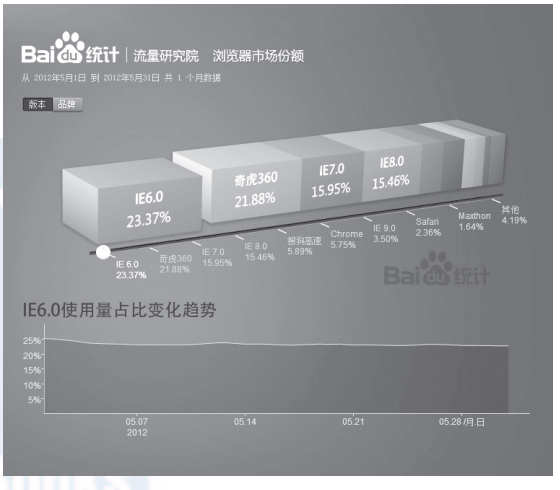


图 1-3 国内浏览器市场份额

2. 浏览器全球市场份额

2012 年 5 月可以说是浏览器厂商激烈竞争的一个月，一度报出 Google Chrome 浏览器全球份额首次超越 IE 浏览器，夺得浏览器全球霸主之位。全球浏览器市场份额发生了哪些变化呢？首先看 StatCounter 的统计数据[⊖]，如图 1-4 所示。

图 1-4 中数据显示，在 2012 年 5 月，IE 浏览器已失去了浏览器的霸主之位，被 Chrome 取代，Firefox 也在市场上位居第三。如果将其他版本的 Firefox、Chrome、Safari 和 Opera 加在一起计算，IE 所占的市场份额确实已少于这些符合标准的现代浏览器。通常，我在自己站点上发布一个新的 CSS3 技巧时，很多朋友会问：“它在 IE 浏览器上能运行吗？效果又是什么样？”根据图 1-4 的显示结构，是不是应该换一种思维，是不是询问“这个效果在 Firefox 上看起来怎么样”更有意义呢？

那么是不是可以忽略 IE 呢？其实不然，IE 虽然在全球市场份额不再是霸主，但在国内它依然是主流，特别是 IE 6 依然占有半壁江山，这也致使我们不能不考虑使用 IE 的用户群。

⊖ 想了解其他来源的统计结果，可以访问维基百科的“Usage share of web browsers”页面（http://en.wikipedia.org/wiki/Usage_share_of_web_browsers）。

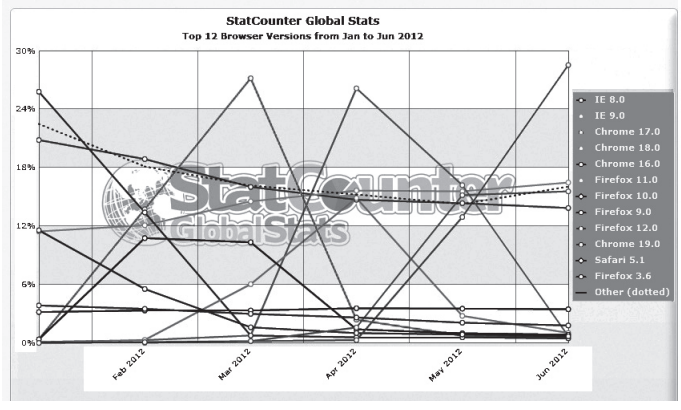


图 1-4　主流浏览器市场份额（2012 年 1 月～ 2012 年 6 月）

制作一个网站，其内容应该在任何浏览器上都是可用的，不应该忽略或抛弃某些用户。虽然制作一个令人满意的 Web 页面不是一件难事，但是为了一个渐渐消失的用户群体花费大量的时间与成本确实不是一个明智之举。

正如前面所讲，CSS3 对网站意义何在，应该是由用户群体来确定，而不是由浏览器的市场份额所决定。换而言之，除非网站统计结果与这个结果有很大的出入，否则就不应该继续认为非 IE 用户仅仅是个不需要特别关注的边缘化群体。在非 IE 浏览器与 IE 浏览器上花费的时间同样重要。而 CSS3 能很容易地让网站在非 IE 浏览器下更棒，而且少数情况之下这些 CSS3 属性也适合 IE 浏览器。

1.2.3　主流浏览器对 CSS3 支持状况

幸运的是，CSS3 特性大部分都已经有了很好的浏览器支持度（后面在讲每个 CSS3 特性之时，会列出各浏览器对其支持情况）。各大主流浏览器对 CSS3 的支持越来越完善，曾经让多少前端开发人员心碎的 IE 也开始挺进 CSS3 标准行列。当然，即使 CSS3 标准制定完成，现代浏览器要普及到大部分用户也是一个相当漫长的过程。如果现在就要使用 CSS3 来美化你的站点，有必要对各大主流浏览器对其新技术的支持情况有一个全面的了解。

本节分别在 Mac 和 Windows 两个平台介绍 Chrome、Firefox、Safari、Opera 和 IE 五大主流浏览器对 CSS3 新特性和 CSS3 选择器的支持情况。

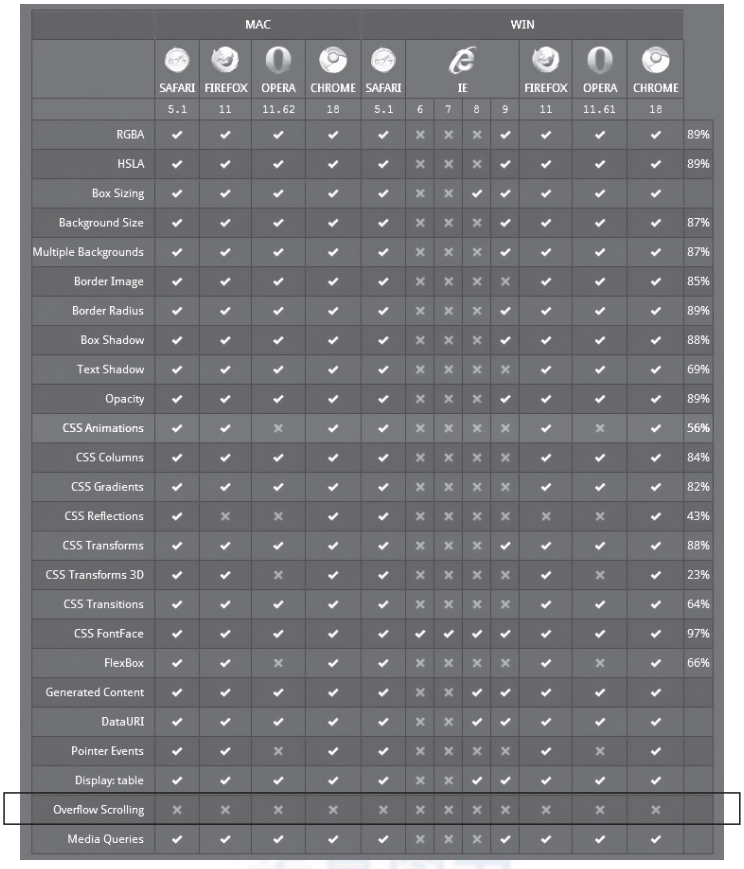
1. 主流浏览器对 CSS3 属性的支持状况

图 1-5 所示是 findmebyIP[⊖]为五大主流浏览器对 CSS3 属性支持状况的统计图。

从图中可以看出，CSS3 中的 Overflow Scrolling 属性还没有浏览器支持，其他属性在 Mac 系统下，Chrome、Safari 都支持，其次支持较好的是 Firefox 和 Opera。而在 Windows 系统下，WebKit 内核浏览器表现的非常优秀，其次是 Firefox 和 Opera。同时 IE 也迎头追

⊖　详情见 http://fmbip.com/litmus。

赶，在 IE 9 中支持部分 CSS3 特性。据说，IE 10 将会更完美地支持 CSS3。



	MAC				WIN								
	SAFARI	FIREFOX	OPERA	CHROME	SAFARI	IE				FIREFOX	OPERA	CHROME	
	5.1	11	11.62	18	5.1	6	7	8	9	11	11.61	18	
RGBA	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
HSLA	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
Box Sizing	✔	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	
Background Size	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	87%
Multiple Backgrounds	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	87%
Border Image	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	85%
Border Radius	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
Box Shadow	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	88%
Text Shadow	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	69%
Opacity	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
CSS Animations	✔	✔	✘	✔	✔	✘	✘	✘	✘	✔	✘	✔	56%
CSS Columns	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	84%
CSS Gradients	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	82%
CSS Reflections	✔	✘	✘	✔	✔	✘	✘	✘	✘	✘	✘	✔	43%
CSS Transforms	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	88%
CSS Transforms 3D	✔	✔	✘	✔	✔	✘	✘	✘	✘	✔	✘	✔	23%
CSS Transitions	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	64%
CSS FontFace	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	97%
FlexBox	✔	✔	✘	✔	✔	✘	✘	✘	✘	✔	✘	✔	66%
Generated Content	✔	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	
DataURI	✔	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	
Pointer Events	✔	✔	✘	✔	✔	✘	✘	✘	✘	✔	✘	✔	
Display: table	✔	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	
Overflow Scrolling	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	
Media Queries	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	

图 1-5　主流浏览器对 CSS3 属性支持状况

2. 主浏览浏览器对 CSS3 选择器的支持状况

图 1-6 所示是 findmebyIP[⊖]为五大主流浏览器对 CSS3 选择器的支持状况统计图。值得高兴的是，图中除了 IE 6 ～ 8 有“ × ”之外，其他浏览器都是“√”，全部支持 CSS3 选择器。

从图 1-5 和图 1-6 中可以清楚地知道，无论是在 Mac 系统下还是在 Windows 系统下，Google Chrome 和 Safari 对 CSS3 特性的支持度是最好的，而 IE 系列是最差的，特别是 IE 6 ～ 8。原因很简单，IE 6 ～ 8 发布于 CSS3 完善之前。

差别各异的浏览器致使页面在不同的浏览器之中渲染并不一致。特别是在当今这个信息发达的时代，设备、屏幕、浏览器的形态越来越丰富，人们的习惯设置也不尽相同，因此想再创造一个在任何地方都表现一致的页面就更加的不可能。只要你关注如何提供实用、易用、好用的页面，这点表面上的差异就显得不重要。而这种想法就是接下来要介绍的“渐进增强”。

⊖ 详情见 http://fmbip.com/litmus。

	MAC				WIN								
	SAFARI	FIREFOX	OPERA	CHROME	SAFARI	IE				FIREFOX	OPERA	CHROME	
	5.1	11	11.62	18	5.1	6	7	8	9	11	11.61	18	
Begins with	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	98%
Ends with	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	98%
Matches	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	98%
Root	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
nth-child	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
nth-last-child	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
nth-of-type	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
nth-last-of-type	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
last-child	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
first-of-type	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
last-of-type	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
only-child	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
only-of-type	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
empty	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
target	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	88%
enabled	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
disabled	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
checked	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
not	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✔	✔	89%
General Sibling	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	98%

图 1-6　主流浏览器对 CSS3 选择器的支持状况

1.3　渐进增强

第一次听到“渐进增强”（Progressive Enhancement）一词是在前端重构交流会上。渐进增强并不是一种技术，而是一种开发的方式，更是一种 Web 设计理念。首先思考一个问题：“网站是否需要在每个浏览器中看起来都一样？”带着这个问题来看渐进增强。

1.3.1　渐进增强与优雅降级

正如前面所言，渐进增强是一种开发方式，更是一种设计理念。在编写 Web 页面时，首先保证最核心的功能实现，让任何低端的浏览器都能看到站点内容，然后考虑使用高级但非必要的 CSS 和 JavaScript 等增强功能，为当前和未来的浏览器提供更好的支持，给用户带来更好的体验。

在设计的时候，先考虑低端设备用户能否看到所有内容，然后在此基础之上为高端用户进行设计。不仅为高端设备用户提供一个完美的应用，也要为不同性能级别设备的用户设计不同级别的不那么完美的应用。这称为“优雅降级”。

目前而言，虽对“渐进增强”有所了解的人很多，但是要说普及或深入还远远没到时候。在大家平时的设计思维中有一种极强的固定思维，也就是想让网站在各个浏览器下表现一致。这种出发点本身并没有什么问题，但是这样会让领先的浏览器的优势无法充分显示出来。

因此，从今天开始要改变制作 Web 站点的思维，让网站能优雅降级，目标应该是——向尽可能多的用户提供尽可能优质的用户体验。这跟用户访问网站使用方式无关，无论通过 iPhone、高端的桌面系统、Kindle，还是阅读器，用户都能得到尽可能独特且完美的体验。

1.3.2 渐进增强的优点

“向尽可能多的用户提供尽可能优质的用户体验”这一目标听起来相当不错。有的人制作 Web 站点时报怨，IE 怎样才具有 CSS3 的效果。诚然，我们不使用渐进增强也可以实现，如为一些旧浏览器提供一套兼容方案，确保页面与现代浏览器保持一致。简单来说就是在支持 CSS3 的浏览器中使用 CSS3 特性，在不支持的浏览器中另写一套样式来模拟 CSS3 效果，实现让网站在所有浏览器都一样。

可以说，通过这种方法只是让低版本的浏览器渲染页面更好看一点，并没有得到实质性的提高。

因此，如果网站始终无法做到一模一样，为什么不使用 CSS3 技术使它在现代浏览器上看起来更靓丽呢？当然，某些 CSS3 特性在不支持的浏览器中是“无法模拟”的，但使用渐进增强，就无须为了让网站适合所有人而放弃这些技术。

CSS 的渐进增强有别于 CSS 的 hack。hack 是浏览器厂商的一种手法，用来增强自己的竞争，而渐进增强起到锦上添花之效。所以前者应该尽量避免使用，而后者应该适当使用。

就样式而言，渐进增强的对象是一些现代浏览器，渐进增强的一些属性主要是 CSS3 的一些特性，或是 IE 低级版本不支持的一些属性，或是其他一些特殊情况。淘宝网上的一个例子如图 1-7 所示。

这里采用了 CSS3 的旋转特性 transform，鼠标移上去时，三角会实现旋转的动画效果，并改变方向。之前，要实现图 1-7 所示的旋转效果，需要一大串 JavaScript 脚本。而使用 transform 仅仅需要几行代码。

```
#site-nave .menu:hover .menu-hd {
  -webkit-transform: rotate(180deg);
  -webkit-transform-origin-x:50%;
  -webkit-transform-origin-y:30%;
  -moz-transform: rotate(180deg);
  -moz-transform-origin-x:50%;
  -moz-transform-origin-y:30%;
  -o-transform: rotate(180deg);
  -o-transform-origin-x:50%;
  -o-transform-origin-y:30%;
```

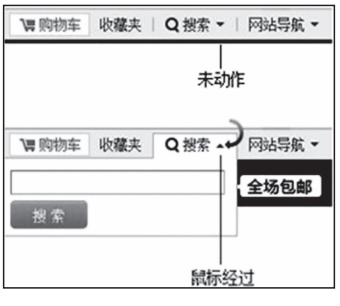


图 1-7 渐进增强制作旋转三角

```
    -ms-transform: rotate(180deg);
    -ms-transform-origin-x:50%;
    -ms-transform-origin-y:30%;
    transform:rotate(180deg);
    transform-origin-x: 50%;
    transform-origin-y: 30%;
}
```

采用渐进增强能给现代浏览器用户一个更好的体验，在不支持 CSS3 的 IE 浏览器也能正常使用，只不过体验会大打折扣。

以上只是一个简单的例子，本书后面还会介绍 CSS3 渐进增强的案例，例如 text-shadow 文字阴影、border-radius 圆角属性、box-shadow 盒阴影属性、CSS3 渐变背景和 CSS3 选择器等。

1.4　CSS3 的现状及未来

了解到使用 CSS3 会带来很多好处，哪些网站在使用 CSS3？这很容易，打开计算机随处可见。既然这样，就一起来看几个网站吧。

1.4.1　谁在使用 CSS3

首先看看新浪微博[⊖]，其中最明显的是圆角的应用，在发表评论的地方还使用了内阴影属性，如图 1-8 所示。

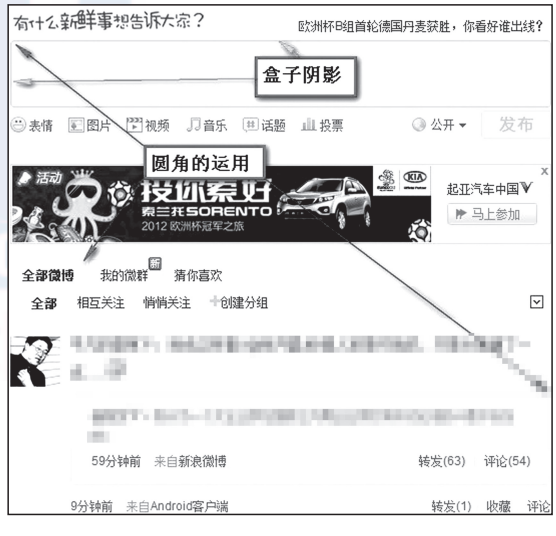


图 1-8　新浪微博

Google 的 UI 也使用了大量 CSS3 特性，看主版面的 Button 效果，这个按钮使用了 CSS3 的阴影、圆角和渐变三种属性，如图 1-9 所示。

图 1-9　Google 的按钮效果

⊖ 网页地址为 http://weibo.com/311290005。

接下来介绍的 Twitter 网站（http://twitter.com）可以说把 CSS3 运用得出神入化，登录界面如图 1-10 所示。

图 1-10　Twitter 登录界面

整个界面都是使用 CSS3 完成的，没有使用任何图片、背景渐变、圆角、文本框发光效果等。这些都是 CSS3 特性的一气呵成，这不得不让我们为 CSS3 强大的特性折服。

国内外使用 CSS3 特性制作网站的案例越来越多，特别是一些优秀的个人站点，更是运用得出神入化。

1.4.2　CSS3 的未来

CSS3 无疑对 Web 前端开发带来质的飞跃。虽然目前 CSS3 还没有完全普及，而且浏览器也不完全支持，但对于我们积极地去学习和实践并不矛盾，掌握和学习 CSS3 将是大势所趋。CSS3 将是引导我们进入编写网页精彩世界的先驱技术。开发人员能够更轻松地创建功能强大、易于维护网站。

随着旧版浏览器所占市场份额逐渐减少，学习 CSS3 技术将更有价值。这是作为一位优秀前端开发人员所必须掌握的技术之一，也是前端开发人员的大势所趋。

当然，学习一门新技术不能跟风，需要理性思考，这种理性思考并不表示对新技术的畏缩，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，才能更好地驾驭 CSS3。

1.5　本章小结

本章主要介绍了什么是 CSS3、CSS3 的发展状况、新特性，以及浏览器对 CSS3 的支持状况；同时，引进了渐进增强的设计理念。通过本章的学习，可以对 CSS3 有一个初步的了解。学习 CSS3 的好处有很多，它能让你始终处于 Web 设计的前沿，增加你的职业技能和竞争力，还能帮助你缩短与顶级设计师或开发者的距离。马上开始使用，然后不断磨砺你的技巧并在职业的道路上不断前进。

第 2 章 Chapter 2

CSS3 选择器

W3C 在 CSS3 的工作草案中把选择器独立出来成为一个模块[⊖]。实际上，选择器是 CSS 知识中的重要部分之一，也是 CSS 的根基。利用 CSS 选择器能不改动 HTML 结构，通过添加不同的 CSS 规则得到不同样式的网页。

2.1 认识 CSS 选择器

要使某个样式应用于特定的 HTML 元素，首先需要找到该元素。在 CSS 中，执行这一任务的表现规则称为 CSS 选择器。它为获取目标元素之后施加样式提供了极大的灵活性。实际上，CSS2.1 已经为大家提供了很多常用的选择器，基本能够满足 Web 设计师常规的设计需求。

2.1.1 CSS3 选择器的优势

既然 CSS 选择器能满足 Web 常规的设计需求，CSS3 选择器有什么优势呢？ CSS3 选择器不但支持所有 CSS 选择器，同时新增了独有的选择器，对拥有一定 CSS 基础的开发人员来说，学习 CSS3 选择器是件非常容易的事。

CSS3 选择器在常规选择器的基础上新增了属性选择器、伪类选择器、过滤选择器。可以帮助您在开发中减少对 HTML 类名或 ID 名的依赖，以及对 HTML 元素的结构依赖，使编写代码更加简单轻松。如果学习过 jQuery 选择器，学习 CSS3 选择器会更容易，因为 CSS3 选择器在某些方面和 jQuery 选择器是完全一样的，唯一遗憾的是部分旧版本浏览器

⊖ 详情参考 http://www.w3.org/TR/css3-selectors。

并不支持 CSS3 新增的部分选择器。下面一起来体验 CSS3 选择器。

2.1.2 CSS3 选择器分类

根据所获取页面中元素的不同，把 CSS3 选择器分为五大类：基本选择器、层次选择器、伪类选择器、伪元素和属性选择器。其中，伪类选择器又分为六种：动态伪类选择器、目标伪类选择器、语言伪类、UI 元素状态伪类选择器，结构伪类选择器和否定伪类选择器，如图 2-1 所示。

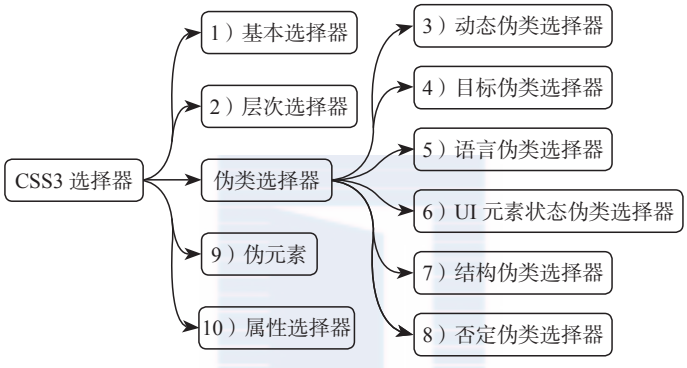


图 2-1 CSS3 选择器分类

下面分别介绍这十种选择器。

2.2 基本选择器

基本选择器是 CSS 中使用最频繁、最基础，也是 CSS 中最早定义的选择器，这部分选择器在 CSS1 中就定义了，为了便于初学者温故而知新，不妨回顾 CSS 的基础选择器。

2.2.1 基本选择器语法

通过基本选择器可以确定 HTML 树形结构中大多数的 DOM 元素节点。其详细说明如表 2-1 所示。

表 2-1 基本选择器语法

选择器	类型	功能描述
*	通配选择器	选择文档中所有的 HTML 元素
E	元素选择器	选择指定的类型的 HTML 元素
#id	ID 选择器	选择指定 ID 属性值为“id”的任意类型元素
.class	类选择器	选择指定 class 属性值为“class”的任意类型的任意多个元素
selector1,selectorN	群组选择器	将每一个选择器匹配的元素集合并

2.2.2　浏览器兼容性

浏览器兼容性如表 2-2 所示。从表中可以看出，浏览器对基本选择器都是一路绿灯通行，大家可以放心使用。

表 2-2　基本选择器的浏览器兼容性

选择器					
*	√	√	√	√	√
E	√	√	√	√	√
#id	√	√	√	√	√
.class	√	√	√	√	√
selector1,selectorN	√	√	√	√	√

2.2.3　实战体验：使用基本选择器

下面通过示例介绍各种基本选择器在页面中的使用方法。

页面中有一个列表，其中第一个和最后一个设置了 ID 属性，其中部分列表项设置了 class 类名，通过基本选择器来改变元素的样式风格。

新创建一个 HTML 文件 2-1.html，加入以下代码。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>使用 CSS3 基本选择器 </title>
  <style type="text/css">
    *{margin: 0;padding:0;}
    .clearfix:after,.clearfix:before{display:table;content:""}
    .clearfix:after{clear:both;overflow:hidden}
    .demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
    li {list-style:none outside none; float: left; height: 20px;
        line-height: 20px; width: 20px;border-radius: 10px;
        text-align: center; background: #f36; color: green; margin-right: 5px; }
  </style>
</head>
<body>
  <ul class="clearfix demo">
    <li class="first" id="first">1</li>
    <li class="active">2</li>
    <li class="important item">3</li>
    <li class="important">4</li>
    <li class="item">5</li>
    <li>6</li>
    <li>7</li>
    <li>8</li>
    <li>9</li>
    <li class="last" id="last">10</li>
  </ul>
</body>
</html>
```

上面代码使用了基本选择器，首先看看页面的初步效果（背景为粉红色），如图 2-2 所示。下面通过图解的方法说明 CSS3 基本选择器的使用方法。

☆图 2-2　页面初步效果㊀

2.2.4　通配选择器

通配选择器（*）用来选择所有元素，当然也可以选择某个元素下的所有元素。如：

```
*｛margin: 0;padding:0｝
```

上面一行代码大家在 Reset 样式文件中经常看到，表示所有元素的 margin 和 padding 都设置为 0。为了更好地说明问题，通过 CSS3 选择器中的通配选择器来改变列表中所有子项的背景色设置为 orange。

```
    *{margin: 0;padding:0;}
    .clearfix:after,.clearfix:before{display:table;content:""}
    .clearfix:after{clear:both;overflow:hidden}
    .demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
    li {list-style:none outside none; float: left; height: 20px;
        line-height: 20px; width: 20px;border-radius: 10px; text-align: center;
background: #f36; color: green; margin-right: 5px; }
    .demo * {background: orange}
```

此时元素类名为 demo 下的所有元素都将背景色设置为橙色，如图 2-3 所示。

☆图 2-3　通配选择器使用效果

2.2.5　元素选择器

元素选择器（E）是 CSS 选择器中最常见、最基本的选择器。文档的元素包括 html、body、p、iv 等，如示例中 ul、li 也属于 HTML 元素。接下来通过 ul 元素选择器改变整个列表的背景色。

```
    *{margin: 0;padding:0;}
    .clearfix:after,.clearfix:before{display:table;content:""}
    .clearfix:after{clear:both;overflow:hidden}
    .demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
    li {list-style:none outside none; float: left; height: 20px; line-height: 20px;
            width: 20px;border-radius: 10px; text-align: center; background: #f36;
            color: green; margin-right: 5px; }
    .demo *{background: orange}
    ul {background:grey}
```

此时列表 ul 的背景色将变成灰色，如图 2-4 所示。

☆图 2-4　元素选择器使用效果

2.2.6　ID 选择器

在使用 ID 选择器（#id）之前，需要在 HTML 文档中给对应的元素设置 id 属性并设置

㊀ 有☆号的图示附有彩色图，后同。

其值，才能找到对应的元素。ID 选择器具有唯一性，在一个页面中不会同时出现 id 相同的属性值。在 CSS 样式中使用 id 选择器时，需要在 id 属性值的前面加上“#”号，如（#id）。在下面这个示例中，可以轻松地看到列表中的第一项和最后一项分别定义了一个 id，其值分别为“ first”和“ last”。使用这两个 id 值来改变列表项中第一个和最后一个列表项的背景色和前景色，代码如下。

```
*{margin: 0;padding:0;}
.clearfix:after,.clearfix:before{display:table;content:""}
.clearfix:after{clear:both;overflow:hidden}
.demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
 li {list-style:none outside none; float: left; height: 20px; line-height: 20px;
          width: 20px;border-radius: 10px; text-align: center; background: #f36;
          color: green; margin-right: 5px; }
.demo *{background: orange}
ul {background:grey}
#first{background:lime;color:#000}
#last {background:#000;color:lime}
```

页面效果如图 2-5 所示。

☆图 2-5　ID 选择器使用效果

2.2.7　类选择器

类选择器（.class）是以独立于文档元素的方式来指定元素样式。使用方法与 ID 选择器极其相似，首先在 HTML 给需要的元素定义 class 属性，并为其设置属性值。其与 ID 选择器有一个很大不同之处。“**类选择器在一个页面中可以有多个相同的类名，而 ID 选择器其 ID 值在整个页面中是唯一的一个**”。同样，看看如何通过类选择器来改变元素的样式。

```
*{margin: 0;padding:0;}
.clearfix:after,.clearfix:before{display:table;content:""}
.clearfix:after{clear:both;overflow:hidden}
.demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
li {list-style:none outside none; float: left; height: 20px;
    line-height: 20px; width: 20px;border-radius: 10px;
    text-align: center; background: #f36; color: green;
     margin-right: 5px; }
.demo *{background: orange}
ul {background:grey}
#first{background:lime;color:#000}
#last {background:#000;color:lime}
.item {background: green;color: #fff;font-weight:bold}
```

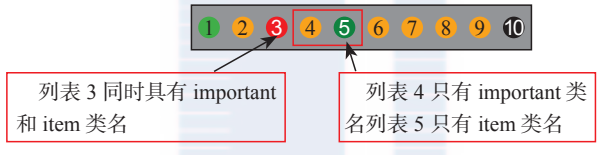
页面效果就变成图 2-6 所示的风格了。

上面是类选择器的简单使用，其实类选择器还有一种使用方法，就是多类选择器。通过两个或两个以上类选择器合并，来定义有别于一个类名的元素效果。

☆图 2-6　类选择器使用效果

```
*{margin: 0;padding:0;}
.clearfix:after,.clearfix:before{display:table;content:""}
.clearfix:after{clear:both;overflow:hidden}
.demo { width: 250px; border: 1px solid #ccc; padding: 10px;margin: 20px auto}
li {list-style:none outside none; float: left; height: 20px;
    line-height: 20px; width: 20px;border-radius: 10px;
    text-align: center; background: #f36; color: green;
    margin-right: 5px; }
.demo *{background: orange}
ul {background:grey}
#first{background:lime;color:#000}
#last {background:#000;color:lime}
.item {background: green;color: #fff;font-weight:bold}
.item.important {background:red;}
```

使用多类选择器时，大家需要注意，如果一个多类选择器包含的类名中其中有一个不存在，这个选择器将无法找到相匹配的元素，正如上面的代码，其只能匹配 li 元素同时具有“item”和“important”的元素，而只有其中任何一个类名都将无法匹配，如图 2-7 所示。



☆图 2-7　多类名选择器使用效果

IE 6 选择器并不支持多类名选择器，其将以末尾类名为准，大家使用时千万小心。

由于类名在一个 HTML 文档中可以同时存在于不同的元素标签上。换句话说，在一个 HTML 文档中，div 可以有类名“block”，ul 也可以有类名“block”，但有时在 Web 的页面开发中，仅需要对 ul 为“block”定义样式，此时仅采用类名选择器并不能达到需要的效果，其实 CSS 选择器还支持带有标签的类名选择器“ul.block”。

```
ul.block{background:#ccc;}
```

上面的代码只匹配 class 属性包含“block”的所有 ul 元素，但其他任何类型的元素都不匹配，包括有“blcok”类名的元素。简而言之，“ul.block”只会匹配 ul 元素，并且有一个类名“block”。不符合这两个条件的任何一个都不能与选择器匹配。

2.2.8　群组选择器

群组选择器（selector1,selectorN）是将具有相同样式的元素分组在一起，每个选择器之间用逗号（,）隔开，例如“selector1,selector2,…,selectorN”。这个逗号告诉浏览器，规则中包含多个不同的选择器，省去逗号就成了后代选择器，这一点大家在使用中千万要小心。

注意　“selector1,selectorN”是群组选择器，表示选择匹配为 selector1 和 selectorN 的所有元素；“selector1 selectorN”是后代选择器（后面介绍），表示选择器 selectorN 所有元素为 selector1 的后代元素。

2.3　层次选择器

层次选择器通过 HTML 的 DOM 元素间的层次关系获取元素，其主要的层次关系包括后代、父子、相邻兄弟和通用兄弟几种关系，通过其中某类关系可以方便快捷地选定需要的元素。

2.3.1　层次选择器语法

层次选择器是一个非常好的选择器，也是大家常用的选择器，其详细说明如表 2-3 所示。

表 2-3　层次选择器语法

选择器	类型	功能描述
E　F	后代选择器（包含选择器）	选择匹配的 F 元素，且匹配的 F 元素被包含在匹配的 E 元素内
E > F	子选择器	选择匹配的 F 元素，且匹配的 F 元素是所匹配的 E 元素的子元素
E + F	相邻兄弟选择器	选择匹配的 F 元素，且匹配的 F 元素紧位于匹配的 E 元素后面
E ～ F	通用选择器	选择匹配的 F 元素，且位于匹配的 E 元素后的所有匹配的 F 元素

2.3.2　浏览器兼容性

浏览器的兼容性如表 2-4 所示。从表中可以看出，子选择器、相邻兄弟选择器和通用兄弟选择器要 IE 7 以及其以上版本才支持，随着 IE 6 的慢慢消失，层次选择可以放心使用。

表 2-4　层次选择器的浏览器兼容性

选择器					
E　F	√	√	√	√	√
E > F	7 + √	√	√	√	√
E + F	7 + √	√	√	√	√
E ～ F	7 + √	√	√	√	√

2.3.3　实战体验：使用层次选择器选择元素

在层次选择器中，后代选择器与子选择器是比较常用的，而对于相邻兄弟选择器和通用兄弟选择器而言，平时大家并不常使用，特别是 CSS3 选择器中新增的通用兄弟选择器。下面通过示例来演示各种层次选择器在页面中如何选择 HTML 的 DOM 元素的方法。

页面中有 10 个 div 元素，其中第四个 div 中包含了 2 个 div，另外第七个 div 包含了第八个，而第八个 div 又包含了第九个 div，并且第九个 div 包含了第十个 div。下面通过层次

选择器来改变 div 的样式风格。

新创建一个 HTML 文件 2-2.html，代码如下。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>使用 CSS3 层次选择器 </title>
  <style type="text/css">
    *{margin: 0;padding:0;}
    body {width: 300px;margin: 0 auto;}
    div{margin:5px;padding:5px;border: 1px solid #ccc;}
  </style>
</head>
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4
    <div>5</div>
    <div>6</div>
  </div>
  <div>7
    <div>8
      <div>9
        <div>10</div>
      </div>
    </div>
  </div>
</body>
</html>
```


在具体使用层选择器之前，先来看看页面的初步效果，如图 2-8 所示。

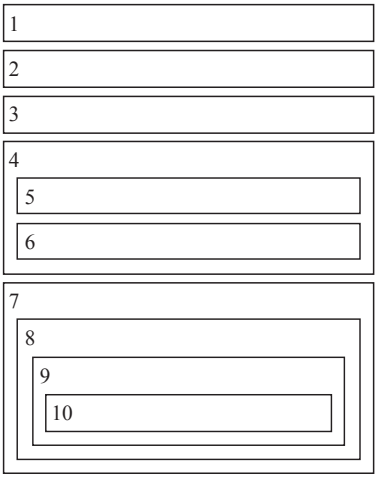


图 2-8　页面初步效果

为了更好地理清这些 div 的层次关系，可以先将示例中的 body 部分画一个 DOM 树形草图，如图 2-9 所示。

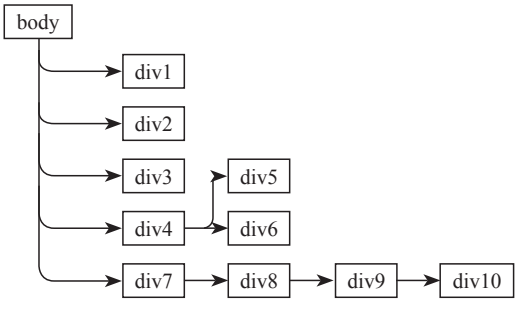


图 2-9　body 的树形结构

2.3.4　后代选择器

后代选择器（E F）也称为包含选择器，作用就是可以选择某元素的后代元素。例如“E F”，E 为祖先元素，F 为后代元素，表达的意思就是选择 E 元素的所有后代 F 元素。这里的 F 元素不管是 E 元素的子元素、孙辈元素或者更深层次的关系，都将被选中。换句话说，不论 F 在 E 中有多少层级关系，F 元素都将被选中。接下来使用后代选择器来改变其背景色。

```
*{margin: 0;padding:0;}
body {width: 300px;margin: 0 auto;}
div{margin:5px;padding:5px;border: 1px solid #ccc;}
div div {background: orange;}
```

一起来看使用后代选择器改变了哪几个 div 的背景色，如图 2-10 所示。

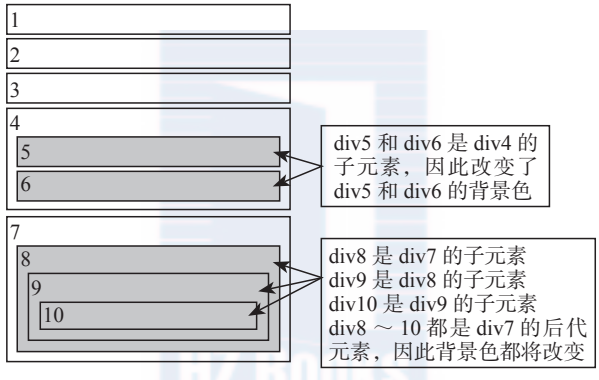


图 2-10　使用后代选择器的效果

注意　后代选择器两选择符之间必须以空格隔开，中间不能有任何其他符号插入。

2.3.5　子选择器

子选择器（E > F）只能选择某元素的子元素，其中 E 为父元素，而 F 为子元素，其中 E>F 表示选择了 E 元素下所有子元素 F。这与后代选择器（E　F）不一样，在后代选择器中 F 是 E 的后代元素，而在 E>F 中 F 仅仅是 E 的子元素而已。接下来通过例子，选择器改变 body 下的子元素 div 的背景色。

```
*{margin: 0;padding:0;}
body {width: 300px;margin: 0 auto;}
div{margin:5px;padding:5px;border: 1px solid #ccc;}
div div {background: orange;}
body > div {background: green;}
```

通过上面子选择器的运用，你能猜出 body 的 10 个 div 中，哪几个背景色将变成绿色

呢？来看图 2-11 的效果。

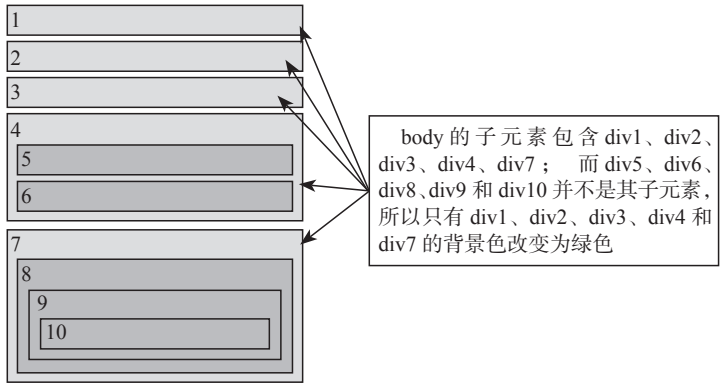


图 2-11　使用子选择器的效果

2.3.6　相邻兄弟选择器

相邻兄弟选择器（E + F）可以选择紧接在另一个元素后的元素，它们具有一个相同的父元素。换句话说，E 和 F 是同辈元素，F 元素在 E 元素后面，并且相邻，这样就可以使用相邻兄弟选择器来选择 F 元素。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>使用 CSS3 层次选择器 </title>
  <style type="text/css">
    *{margin: 0;padding:0;}
    body {width: 300px;margin: 0 auto;}
    div{margin:5px;padding:5px;border: 1px solid #ccc;}
    div div {background: orange;}
    body > div {background: green;}
    .active + div {background: lime;}
  </style>
</head>
<body>
  <div  class="active">1</div>
  <!-- 为了说明相邻兄弟选择器，在此处添加一个类名 active -->
  <div>2</div>
  <div>3</div>
  <div>4
    <div>5</div>
    <div>6</div>
  </div>
  <div>7
    <div>8
```

```
        <div>9
          <div>10</div>
        </div>
      </div>
    </div>
  </body>
  </html>
```

此时第二个 div 的背景色将变成“ lime”色，如图 2-12 所示。

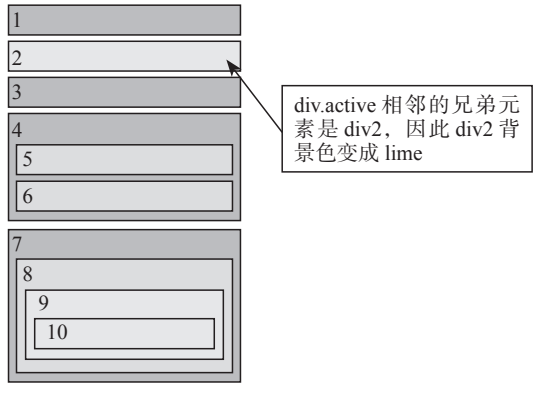


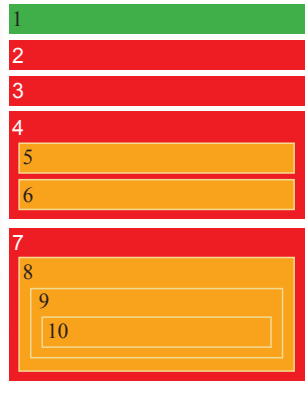
图 2-12　使用相邻兄弟选择器效果

2.3.7　通用兄弟选择器

通用兄弟选择器（E ～ F）是 CSS3 新增加的，用于选择某元素后面的所有兄弟元素，它们和相邻兄弟选择器类似，需要在同一个父元素之中。也就是说，E 和 F 元素都是同辈元素，并且 F 元素在 E 元素之后，E ～ F 将选中 E 元素后面的所有 F 元素。如下面的代码所示。

```
    *{margin: 0;padding:0;}
    body {width: 300px;margin: 0 auto;}
    div{margin:5px;padding:5px;border:
1px solid #ccc;}
    div div {background: orange;}
    body > div {background: green;}
    .active + div {background: lime;}
    .active ～ div {background: red;}
```

这样，只要是 div.active 的兄弟元素 div，并且在 div.active 之后，其背景色将变成红色，如图 2-13 所示。



☆图 2-13　使用通用兄弟选择器效果

注意　通用兄弟选择器选中的是与 E 元素相邻的后面兄弟元素 F，其选中的是一个或多个元素；而相邻兄弟选择器选中的仅是与 E 元素相邻并且紧挨的兄弟元素 F，其选中的仅是一个元素。

2.4　动态伪类选择器

伪类选择器对于大家来说最熟悉的莫过于“ :link”、“ :visited”、“ :hover”、“ :active”，因为这些是大家平时常用到的伪类选择器。而 CSS3 的伪类选择器可以分成六种：动态伪类选择器、目标伪类选择器、语言伪类选择器、UI 状态伪类选择器、结构伪类选择器和否定

伪类选择器。

伪类选择器语法书写时和其他的 CSS 选择器写法有所区别，都以冒号（：）开头。例如：

```
E:pseudo-class {property:value}
```

其中 E 为 HTML 中的元素；pseudo-class 是 CSS 的伪类选择器名称；property 是 CSS 的属性；value 为 CSS 属性值。

CSS3 伪类选择器有什么功能？选定元素能带来什么便利？带着这些问题，依次学习 CSS3 伪类选择器的使用方法，首先是动态伪类选择器。

2.4.1 动态伪类选择器语法

动态伪类早在 CSS 中就有，其并不是 CSS3 才有的，动态伪类并不存在于 HTML 中，只有当用户和网站交互的时候才能体现出来。动态伪类包含两种，第一种是在链接中常看到的锚点伪类，另一种为用户行为伪类。其详细说明如表 2-5 所示。

表 2-5 动态伪类选择器语法

选择器	类型	功能描述
E:link	链接伪类选择器	选择匹配的 E 元素，而且匹配元素被定义了超链接并未被访问过。常用于链接锚点上
E:visited	链接伪类选择器	选择匹配的 E 元素，而且匹配元素被定义了超链接并已被访问过。常用于链接锚点上
E:active	用户行为伪类选择器	选择匹配的 E 元素，且匹配元素被激活。常用于锚点与按钮上
E:hover	用户行为伪类选择器	选择匹配的 E 元素，且用户鼠标在停留在元素 E 上。IE 6 及以下浏览器仅支持 a:hover
E:focus	用户行为伪类选择器	选择匹配的 E 元素，且匹配的元素获得焦点

锚点伪类的设置必须遵守一个“爱恨原则”LoVe/HAte，也就是“link-visited-hover-active”。另外，在 IE 6、IE 7(Q)、IE 8(Q) 中，a:hover、a:active 和 a:visited 并没有按照规范描述的算法来计算它们的针对性，而是根据链接的实际状态来决定使用哪个规则集里的声明。它们三个的针对性比 a:link 强，详细参阅 http://www.w3help.org/zh-cn/causes/RS3005。

2.4.2 浏览器兼容性

浏览器兼容性如表 2-6 所示。

表 2-6 动态伪类选择器浏览器兼容性

选择器					
E:link	√	√	√	√	√
E:visited	√	√	√	√	√

（续）

选择器					
E:active	8 + √	√	√	√	√
E:hover	√	√	√	√	√
E:focus	8 + √	√	√	√	√

注意　E:hover 在 IE 6 浏览器中仅支持链接锚点 a:hover。

2.4.3　实战体验：美化按钮

在众多网站上按钮在不同状态下效果不一，用以增强用户体验，这也是一种非常好的设计体验与细节。实现并不复杂，下面一起来看 Twitter 的 Bootstrap[⊖]如何美化按钮。

实现页面中按钮在不同行为状态下的样式风格，常见的行为状态如默认状态、悬浮状态、用户点击时状态和按钮获得焦点状态。

通过 Twitter 的 Bootstrap 制作的按钮效果如图 2-14 所示。

根据用户的行为不同，按钮效果可以分为：默认状态、悬浮状态、点击时状态、焦点状态和点击后状态，可以按照 CSS3 的动态伪选择器，在不同状态下给按钮赋予不同的样式风格。

根据这一设计思路，可以轻松美化按钮，看下面的示例代码，演示的效果如图 2-14 所示。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>使用动态伪类选择器美化按钮</title>
  <style type="text/css">
    .download-info {
      text-align: center;
    }
    /* 默认状态下的按钮效果 */
    .btn {
      background-color: #0074cc;
      *background-color: #0055cc;
      /*CSS3 渐变制作背景图片 */
      background-image: -ms-linear-gradient(top, #0088cc, #0055cc);
      background-image: -webkit-gradient(linear, 0 0, 0 100%,
                                         from(#0088cc), to(#0055cc));
      background-image: -webkit-linear-gradient(top, #0088cc, #0055cc);
      background-image: -o-linear-gradient(top, #0088cc, #0055cc);
```

☆图 2-14　美化按钮效果

⊖ 参考 http://twitter.github.com/bootstrap/。

```
  background-image: -moz-linear-gradient(top, #0088cc, #0055cc);
  background-image: linear-gradient(top, #0088cc, #0055cc);
  background-repeat: repeat-x;
  display: inline-block;
  *display: inline;
  border: 1px solid #cccccc;
  *border: 0;
  border-color: #ccc;
  /*CSS3 的色彩模块 */
  border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
  border-radius: 6px;
  color: #ffffff;
  cursor: pointer;
  font-size: 20px;
  font-weight: normal;
  filter: progid:dximagetransform.microsoft.gradient
              (startColorstr='#0088cc', endColorstr='#0055cc', GradientType=0);
  filter: progid:dximagetransform.microsoft.gradient(enabled=false);
  line-height: normal;
  padding: 14px 24px;
  text-align: center;
  /*CSS3 文字阴影特性 */
  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
  text-decoration: none;
  vertical-align: middle;
  *zoom: 1;
}
/* 悬浮状态下按钮效果 */
.btn:hover {
  background-position: 0 -15px;
  background-color: #0055cc;
  *background-color: #004ab3;
  color: #ffffff;
  text-decoration: none;
  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
  /*CSS3 动画效果 */
  -webkit-transition: background-position 0.1s linear;
  -moz-transition: background-position 0.1s linear;
  -ms-transition: background-position 0.1s linear;
  -o-transition: background-position 0.1s linear;
  transition: background-position 0.1s linear;
}
/* 点击时按钮效果 */
.btn:active {
  background-color: #0055cc;
  *background-color: #004ab3;
  background-color: #004099 \9;
  background-image: none;
  outline: 0;
  /*CSS3 盒子阴影特性 */
  box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.15),
```

```
                              0 1px 2px rgba(0, 0, 0, 0.05);
      color: rgba(255, 255, 255, 0.75);
    }
    /* 获得焦点按钮效果 */
    .btn:focus {
      outline: thin dotted #333;
      outline: 5px auto -webkit-focus-ring-color;
      outline-offset: -2px;
    }
  </style>
</head>
<body>
  <div class="download-info">
    <a href="#" class="btn">View project on GitHub</a>
  </div>
</body>
</html>
```

这个美化按钮实例涉及一些 CSS3 的特性，此时不懂不要害怕，本书后续章节中会为大家逐一介绍。在此例中只需要知道哪些属性是 CSS3 的特性就足够了。同时实例中采用了“渐进增强，优雅降级”，在不支持 CSS3 的浏览器中，同样可以看到按钮在不同状态下的效果，只是失去了渐变、阴影等效果，但并不影响网站的功能与用户的体验，IE 8 下的效果如图 2-15 所示。

☆图 2-15　IE 8 下的按钮效果

2.5　目标伪类选择器

目标伪类选择器“ :target”是众多实用的 CSS3 特性中的一个，用来匹配文档（页面）的 URI[⊖]中某个标志符的目标元素。具体来说，URI 中的标志符通常会包含一个井号（#），后面带有一个标志符名称，例如“#contact”“ :target”就是用来匹配 ID 为“contact”的元素的。换种说法，在 Web 页面中，一些 URL 拥有片段标识符，它由一个井号（#）后跟一个锚点或元素 ID 组合而成，可以链接到页面的某个特定元素。“ :target”伪类选择器选取链接的目标元素，然后供定义样式。

2.5.1　目标伪类选择器语法

目标伪类选择器的语法如表 2-7 所示。

表 2-7　目标伪类选择器语法

选择器	功能描述
E:target	选择匹配 E 的所有元素，且匹配元素被相关 URL 指向

注意　目标伪类选择器是动态选择器，只有存在 URL 指向该匹配元素时，样式效果才会生效。

⊖ 参考 http://en.wikipedia.org/wiki/Uniform_Resource_Identifier#cite_note-1。

2.5.2 浏览器兼容性

浏览器兼容性如表 2-8 所示。

表 2-8 目标伪类选择器浏览器兼容性

选择器					
E:target	9 + √	√	√	9.6 + √	√

从表 2-8 可知，目标伪类选择器在 IE 8 及之前版本不被支持，但 IE 用户点击目录里的链接仍将跳转到相应的标题，只是标题不会高亮显示。

提示 在包含更多内容的页面中，高亮显示效果的确能给用户带来极好的体验。如果页面效果需要兼容 IE 低版本浏览器，就要用到 JavaScript。这里有一些资源供大家参考。

- “Suckerfish :target”，作者：Patrick Griffiths 和 Dan Webb，网址如下。
  http://www.htmldog.com/articles/suckerfish/target
- “Improving the usability of within-page links ”，作者：Bruce Lawson，网址如下。
  http://dev.opera.com/articles/view/improving-the-usability-of-within-page-l/

2.5.3 实战体验：制作手风琴效果

以前制作手风琴效果（Accordion）需要依赖 JavaScript 脚本。CSS3 的目标伪选择器可不使用任何 JavaScript 代码实现手风琴效果。

页面中有三个区块，默认状态只显示三个区块的标题，点击其中一个标题时，其对应的内容就会显示；点击另一个标题时，对应区块内容将显示，而前一块内容将隐藏。目标伪类选择器制作的页面效果如图 2-16 所示。

通过目标伪类选择器“E:target”，显示和隐藏不同栏目的内容，从而实现手风琴效果。

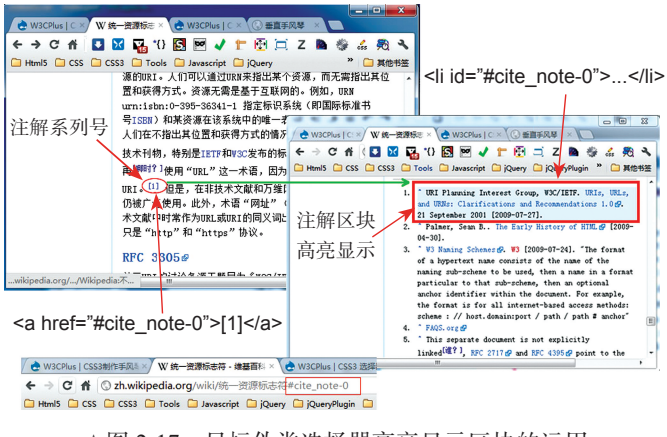
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title> 垂直手风琴 </title>
  <style type="text/css">
.accordionMenu {
  background: #fff;
  color:#424242;
  font: 12px Arial, Verdana, sans-serif;
  margin:0 auto;
  padding: 10px;
  width: 500px;
}
.accordionMenu h2 {
  margin:5px 0;
```

图 2-16 手风琴效果

```
  padding:0;
  position: relative;
}
.accordionMenu h2:before {/* 制作向下三角效果 */
  border: 5px solid #fff;
  border-color: #fff transparent transparent;
  content:"";
  height: 0;
  position:absolute;
  right: 10px;
  top: 15px;
  width: 0;
}
.accordionMenu h2 a {/* 制作手风琴标题栏效果 */
  background: #8f8f8f;
  background: -moz-linear-gradient( top, #cecece, #8f8f8f);
  background: -webkit-gradient(linear, left top, left bottom,
               from(#cecece), to(#8f8f8f));
  background: -webkit-linear-gradient( top, #cecece, #8f8f8f);
  background: -o-linear-gradient( top, #cecece, #8f8f8f);
  background: linear-gradient( top, #cecece, #8f8f8f);
  border-radius: 5px;
  color:#424242;
  display: block;
  font-size: 13px;
  font-weight: normal;
  margin:0;
  padding:10px 10px;
  text-shadow: 2px 2px 2px #aeaeae;
  text-decoration:none;
}
.accordionMenu :target h2 a, /* 目标标题的效果 */
.accordionMenu h2 a:focus,
.accordionMenu h2 a:hover,
.accordionMenu h2 a:active {
  background: #2288dd;
  background: -moz-linear-gradient( top, #6bb2ff, #2288dd);
  background: -webkit-gradient(linear, left top, left bottom,
             from(#6bb2ff), to(#2288dd));
  background: -webkit-linear-gradient( top, #6bb2ff, #2288dd);
  background: -o-linear-gradient( top, #6bb2ff, #2288dd);
  background: linear-gradient( top, #6bb2ff, #2288dd);
  color:#FFF;
}
.accordionMenu p {/* 标题栏对应的内容 */
  margin:0;
  height: 0;/* 默认栏目内容高度为 0，达到隐藏效果 */
  overflow: hidden;
  padding:0 10px;
  -moz-transition: height 0.5s ease-in;
  -webkit-transition: height 0.5s ease-in;
```

```
  -o-transition: height 0.5s ease-in;
  transition: height 0.5s ease-in;
}
/* 这部分是显示内容的关键代码 */
.accordionMenu :target p {/* 展开对应目标内容 */
  height:100px;/* 显示对应目标栏内容 */
  overflow: auto;
}
.accordionMenu :target h2:before {/* 展开时标题三角效果 */
  border-color: transparent transparent transparent #fff;
}
  </style>
</head>
<body>
  <div class="accordionMenu">
        <div class="menuSection" id="brand">
      <h2><a href="#brand">Brand</a></h2>
      <p>Lorem ipsum dolor...</p>
        </div>
        <div class="menuSection" id="promotion">
            <h2><a href="#promotion">Promotion</a></h2>
            <p>Lorem ipsum dolor sit amet...</p>
        </div>
        <div class="menuSection" id="event">
            <h2><a href="#event">Event</a></h2>
            <p>Lorem ipsum dolor sit amet...</p>
        </div>
  </div>
</body>
</html>
```

维基百科的官网[⊖]上就运用了目标伪类选择器来高亮显示脚注，如图 2-17 所示。



☆图 2-17　目标伪类选择器高亮显示区块的运用

⊖ 网址为 http://zh.wikipedia.org/wiki/ 统一资源标志符 #cite_note-0。

点击注解的上标链接时，其对应的注解内容区块就会高亮显示，以便用户在众多内容中找到对应的注解内容，方便用户阅读，而实现此功能仅一行代码就完成了。

```
ol.references > li:target, sup.reference:target, cite:target {
  background-color: #def;
}
```

除了能制作手风琴效果、高亮显示脚注之外，目标伪类选择器还可以用在以下场景，如表 2-9 所示。

表 2-9 “:target”应用场景

效果	地址
高亮显示区块	http://www.red-team-design.com/get-to-know-your-css3-target-pseudo-class
从相互层叠的盒容器或图片中突出显示其中一项	http://virtuelvis.com/ gallery/ css3/ target/interface.html
tabs 效果	http://css-tricks.com/css3-tabs/
幻灯片效果	http://designmodo.com/slider-css3/
灯箱效果	http://www.decodize.com/demos/CSS3-target-pseudo-class/gallery.html
相册效果	http://www.ie7nomore.com/fun/scroll/

其中几项效果使用 JavaScript 制作效果会更好，因为纯 CSS 的版本可能存在潜在的易用性和可用性问题。

2.6　语言伪类选择器

使用语言伪类选择器来匹配使用语言的元素是非常有用的，特别是用于多语言版本的网站，其作用更是明显。可以使用他来根据不同语言版本设置页面的字体风格。

2.6.1　语言伪类选择器语法

语言伪类选择器是根据元素的语言编码匹配元素。这种语言信息必须包含在文档中，或者与文档关联，不能从 CSS 指定。为文档指定语言，有两种方法可以表示。如果使用 HTML 5，直接可以设置文档的语言。例如：

```
<!DOCTYPE HTML>
<html lang="en-US">
```

另一种方法就是手工在文档中指定 lang 属性，并设置对应的语言值。例如：

```
<body lang="fr">
```

语言伪类选择器允许为不同的语言定义特殊的规则，这在多语言版本的网站用起来是

特别的方便。

E:lang(language) 表示选择匹配 E 的所有元素，且匹配元素指定了 lang 属性，而且其值为 language。

2.6.2 浏览器兼容性

浏览器兼容性如表 2-10 所示。

表 2-10 语言伪类选择器的浏览器兼容性

选择器					
E:target	8 + √	√	√	9.2 + √	√

语言伪类选择器在 IE 7 及以下版本中还不被支持，对于追求完美的设计师来说，又有点畏惧，不敢使用。其实也不是没有办法，可以为不同的浏览器（IE 6 和 IE 7）采用不同的方法来实现。

- 对于 IE 6 浏览器，给引文元素在不同版本的时候设置不同的类名，例在英文版本下设置类名“.en”，而在法文版本下设置类名为“.fr”，就可以通过类名给引文定义不同的样式。
- 对于 IE 7 浏览器，也可以采用 IE 6 浏览器的方法。如果不考虑 IE 6 浏览器，可以使用属性选择器中的“E[foo|="en"]”选择器为不同语言版本的引文设置不同样式。

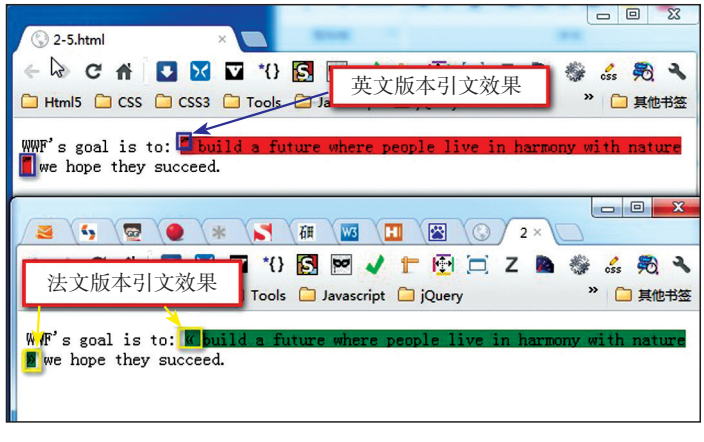
2.6.3 实战体验：定制不同语言版本引文风格

如果网站是一个多语言版本，使用语言伪类选择器为特定的语言定义不同样式是非常完美的。例如，多语言版本有一段这样的引文，如图 2-18 所示。

WWF's goal is to: " build a future where people live in harmony with nature " we hope they succeed.

图 2-18 引文示例

在多语言的网站中，改变引文的不同样式，例如网站还有一个法语语言版本，使用 « ...» 替代是不是比引号（“...”）更适合其语言版本呢？同时，为了突出引文的重要性，可以在不同的语言版本中给引文设置不同的背景颜色。最后的效果如图 2-19 所示。



☆图 2-19 多语言版本引文的效果

在不增加任何代码或手工修改代码达到图 2-19 所示的引文效果，使用语言伪类选择器是一个不错的方法，示例代码如下所示。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>语言伪类选择器运用 </title>
</head>
<body>
<p>WWF's goal is to:
<q cite="http://www.wwf.org">
build a future where people live in harmony with nature
</q> we hope they succeed.</p>
</body>
</html>
```

为 <html> 标签设置一个“ lang=en-US”属性，这是默认英文版本的时候。当网站转译到法语版本，此时 <html> 标签中的 lang 属性值动态变成“ fr”，如下所示。

```
<!DOCTYPE HTML>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>语言伪类选择器运用 </title>
</head>
<body>
<p>WWF's goal is to:
<q cite="http://www.wwf.org">
build a future where people live in harmony with nature
</q> we hope they succeed.</p>
</body>
</html>
```

也可以简单地通过目标伪类来实现。

```
/* 英文 (en-US) 版本的引文 q 效果 */
:lang(en) {
  quotes:'"' '"';
}
:lang(en) q {background: red;}
/* 法文 (fr) 版本的引文 q 效果 */
:lang(fr) {
  quotes:'«' '»';
}
:lang(fr) q {background: green;}
```

提示　大家也可以通过这种方法为不同语言版本的网站相关元素设置不同的样式，例如改变网站面页面的字号、设置不同的背景图片等。

2.7 UI 元素状态伪类选择器

UI 元素状态伪类选择器也是 CSS3 选择器模块组中的一部分，主要用于 form 表单元素上，以提高网页的人机交互、操作逻辑以及页面的整体美观，使表单页面更具个性与品位，而且使用户操作页面表单更便利和简单。

2.7.1 UI 元素状态伪类选择器语法

UI 元素的状态一般包括：启用、禁用、选中、未选中、获得焦点、失去焦点、锁定和待机等。在 HTML 元素中有可用和不可用状态，例如表单中的文本输入框；HTML 元素中还有选中和未选中状态，例如表单中的复选按钮和单选按钮。这几种状态都是 CSS3 选择器中常用的状态伪类选择器，详细说明如表 2-11 所示。

表 2-11 UI 元素状态伪选择器语法

选择器	类型	功能描述
E:checked	选中状态伪类选择器	匹配选中的复选按钮或单选按钮表单元素
E:enabled	启用状态伪类选择器	匹配所有启用的表单元素
E:disabled	不可用状态伪类选择器	匹配所有禁用的表单元素

2.7.2 浏览器兼容性

浏览器兼容性如表 2-12 所示。

表 2-12 UI 元素状态伪类选择器浏览器兼容性

选择器					
E:checked	9 + √	√	√	√	√
E:enabled	9 + √	√	√	√	√
E:disabled	9 + √	√	√	√	√

除了 IE 浏览器外，各主流浏览器对 UI 元素状态选择器的支持都非常好，但 IE 9 也开始全面支持这些 UI 元素状态伪类选择器（如表 2-12 所示）。因此，考虑到 IE 6 ～ 8 是国内用户数最多的浏览器，使用 UI 元素状态伪类选择器需使用特别的方法来处理。

例如使用 JavaScript 库，选用内置已兼容了 UI 元素状态伪类选择器的 JavaScript 库或框架，然后在代码中引入它们并完成想要的效果。由 Keith Clark 编写的 Selectivizr 脚本[⊖]是一个不错的选择。先将该脚本直接引入到页面中，再从 7 个 JavaScript 库中选择一个引入，UI 元素状态伪类选择器就能够在 IE 上工作了。

除了使用 JavaScript 库外，还有一个比较原始而古老的做法，就是在不支持 UI 元素状态伪类选择器的 IE 浏览器下使用类名来处理。例如禁用的按钮效果，可以先在 HTML 标签中添加一个类名“disabled”，就可以在样式中添加样式。

⊖ 网址为 http://selectivizr.com。

```
.btn.disabled,/* 等效于 .btn:disabled，用于兼容 IE 低版本浏览器 */
.btn:disabled {
  …
}
```

2.7.3　实战体验：Bootstrap 的表单元素 UI 状态

UI 元素状态伪类选择器目前主要针对应用在表单元素上，让表单更可用、易用和好用，同时让表单设计更漂亮。这里介绍 Twitter 的表单元素状态是如何来控制的。

Bootstrap 中部分表单元素状态的效果如图 2-20 所示。

图 2-20　Bootstrap 表单元素状态部分效果

图中展示了表单元素中常用的几种 UI 状态效果，接着来看其实现代码。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title></title>
  <style type="text/css">
/* 表单基本样式，请查看对应示例代码 */
/* 表单元素获得焦点效果 */
textarea:focus,
input[type="text"]:focus,
input[type="password"]:focus{
  border-color: rgba(82, 168, 236, 0.8);
  outline: 0;
  outline: thin dotted \9;
  box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075),
                0 0 8px rgba(82, 168, 236, 0.6);
}
/* 表单中下拉选择框、文件控件、单选按钮、复选按钮得到焦点时效果 */
select:focus,
```

```
input[type="file"]:focus,
input[type="radio"]:focus,
input[type="checkbox"]:focus {
  outline: thin dotted #333;
  outline: 5px auto -webkit-focus-ring-color;
  outline-offset: -2px;
}
/* 禁用的 input、select、textarea 表单元素效果 */
input[disabled],  /* 等效于 input:disabled*/
select[disabled], /* 等效于 select:disabled*/
textarea[disabled]/* 等效于 textarea:disabled*/
{
  cursor: not-allowed;
  background-color: #eeeeee;
  border-color: #ddd;
}
/* 禁用的单选按钮和复选按钮效果 */
input[type="radio"][disabled],   /* 等效于 input[type="radio"]:disabled*/
input[type="checkbox"][disabled] /* 等效于 input[type="checkbox"]:disabled*/
{
  background-color: transparent;
}
.control-group.warning > label,
.control-group.warning .help-block,
.control-group.warning .help-inline {
  color: #c09853;
}
/* 表单警告状态下效果 */
.control-group.warning .checkbox,
.control-group.warning .radio,
.control-group.warning input,
.control-group.warning select,
.control-group.warning textarea {
  color: #c09853;
  border-color: #c09853;
}
/* 表单警告状态下并获得焦点下效果 */
.control-group.warning .checkbox:focus,
.control-group.warning .radio:focus,
.control-group.warning input:focus,
.control-group.warning select:focus,
.control-group.warning textarea:focus {
  border-color: #a47e3c;
  box-shadow: 0 0 6px #dbc59e;
}
.control-group.error > label,
.control-group.error .help-block,
.control-group.error .help-inline {
  color: #b94a48;
}
/* 表单错误状态下效果 */
```

```
.control-group.error .checkbox,
.control-group.error .radio,
.control-group.error input,
.control-group.error select,
.control-group.error textarea {
  color: #b94a48;
  border-color: #b94a48;
}
/* 表单错误状态并获取焦点时效果 */
.control-group.error .checkbox:focus,
.control-group.error .radio:focus,
.control-group.error input:focus,
.control-group.error select:focus,
.control-group.error textarea:focus {
  border-color: #953b39;
  box-shadow: 0 0 6px #d59392;
}
.control-group.success > label,
.control-group.success .help-block,
.control-group.success .help-inline {
  color: #468847;
}
/* 表单成功状态下效果 */
.control-group.success .checkbox,
.control-group.success .radio,
.control-group.success input,
.control-group.success select,
.control-group.success textarea {
  color: #468847;
  border-color: #468847;
}
/* 表单成功状态于并获得焦点下效果 */
.control-group.success .checkbox:focus,
.control-group.success .radio:focus,
.control-group.success input:focus,
.control-group.success select:focus,
.control-group.success textarea:focus {
  border-color: #356635;
  box-shadow: 0 0 6px #7aba7b;
}
.form-actions {
  padding: 17px 20px 18px;
  margin-top: 18px;
  margin-bottom: 18px;
  background-color: #f5f5f5;
  border-top: 1px solid #e5e5e5;
  *zoom: 1;
}
.form-actions:before,
.form-actions:after {
  display: table;
```

```
  content: "";
}
.form-actions:after {
  clear: both;
}
/* 按钮基本样式 */
.btn {
  display: inline-block;
  *display: inline;
  /* 由于篇幅，基本样式在此省略，详细请查阅随书代码 */
}
.btn:hover,
.btn:active,
.btn.active,
.btn.disabled,/* 按钮禁用下效果，等效于 .btn:disabled*/
.btn[disabled] {
  background-color: #e6e6e6;
  *background-color: #d9d9d9;
}
.btn:active,
.btn.active {
  background-color: #cccccc \9;
}

.btn:first-child {
  *margin-left: 0;
}
.btn:hover {
  color: #333333;
  text-decoration: none;
  background-color: #e6e6e6;
  *background-color: #d9d9d9;
  background-position: 0 -15px;
  -webkit-transition: background-position 0.1s linear;
     -moz-transition: background-position 0.1s linear;
      -ms-transition: background-position 0.1s linear;
       -o-transition: background-position 0.1s linear;
          transition: background-position 0.1s linear;
}
.btn:focus {
  outline: thin dotted #333;
  outline: 5px auto -webkit-focus-ring-color;
  outline-offset: -2px;
}
.btn.active,
.btn:active {
  background-color: #e6e6e6;
  background-color: #d9d9d9 \9;
  background-image: none;
  outline: 0;
  box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.15),
```

```
                0 1px 2px rgba(0, 0, 0, 0.05);
}
/* 表单按钮禁用状态下效果 */
.btn.disabled,/* 等效于 .btn:disabled*/
.btn[disabled] {
  cursor: default;
  background-color: #e6e6e6;
  background-image: none;
  opacity: 0.65;
  filter: alpha(opacity=65);
  box-shadow: none;
}
...// 省略部分代码
```


注意　以上样式代码摘选于 Bootstrap 中的表单元素样式（详细内容参见 http://twitter.github.com/ bootstrap/base-css.html#forms）。

上面案例主要展示的是表单元素得到焦点和禁用两种状态使用方法，在使用 UI 状态选择器时特别注意，HTML 结构中要存在这种状态，例如禁用的输入框，需要在 HTML 的对应元素上添加禁用属性。

```
<input class="input-xlarge disabled" id="disabledInput"
        type="text" placeholder="Disabled input here..." disabled="">
```

2.8　结构伪类选择器

伪类可以将一段并不存在的 HTML 当作独立元素来定位，或是找到无法使用其他简单选择器就能定位到的切实存在的元素。因此 CSS3 给伪类选择器引入一种“结构伪类选择器”。这种选择器可以根据元素在文档树中的某些特性（如相对位置）定位到它们。也就是说，通过文档树结构的相互关系来匹配特定的元素，从而减少 HTML 文档对 ID 或类名的定义，帮助你保持代码干净和整洁。

2.8.1　重温 HTML 的 DOM 树

所有的结构伪类都是基于 HTML 文档树的，也称做文档对象模型（DOM），下面简单回顾一下这方面的知识。文档树（Document Tree）是 HTML 页面的层级结构。它由元素、属性和文本组成，它们都是一个节点（Node），就像公司的组织结构图一样。下面看一个简单的 HTML 文档。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
```

```
  <title>HTML DOM 树型结构图 </title>
</head>
<body>
  <div>
    <ul>
      <li>one</li>
      <li>two</li>
      <li>three</li>
    </ul>
    <div>abc</div>
    <p>para</p>
    <div>def</div>
    <p>para</p>
    <b>ghi</b>
  </div>
</body>
</html>
```

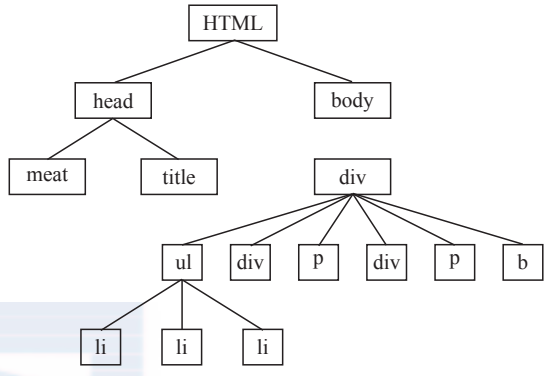


图 2-21　HTML DOM 树型结构

依据上面的 HTML 文档，可以绘制一个清晰的 DOM 结构树，如图 2-21 所示。

图 2-21 所示的文档树包含有多个层级是因为 HTML 元素间的相互嵌套。其中把直接嵌入其他元素的元素被称做那些元素的子元素（Child），例如结构图中的 li 就是 ul 的子元素；而随着嵌套的继续深入，它们也就成了后代元素（Descendant），同样，结构图中的 li 元素就是 body 元素的后代元素。那些外部元素称为父元素（Parent）（一层之上），例如结构图中的 ul 元素就是 li 元素的父元素；有些外部元素称做祖先元素（Ancestor）（两层或以上），例如结构图中的 body 就是 li 元素的祖先元素。另外位于相同嵌套层级的元素称为兄弟元素（具有同一父元素节点），例如结构图中的两个段落 P 元素就是兄弟元素，因为它们具有同一个父元素 div。在 HTML 文档中，一个元素可以同时拥有以上部分甚至所有称谓，正如家谱中的某个成员一样，总的来说这些称谓都是用来描述一个元素与另一个元素的关系。

2.8.2　结构伪类选择器语法

通过回顾 HTML 的 DOM 树型结构，清楚了元素之间的关系术语，现在看看有哪些可以建立元素间关系的方法，表 2-13 列出所有结构伪选择器的详细说明。

表 2-13　结构伪类选择器使用语法

选择器	功能描述
E:first-child	作为父元素的第一个子元素的元素 E。与 E:nth-child(1) 等同
E:last-child	作为父元素的最后一个子元素的元素 E。与 E:nth-last-child(1) 等同
E:root	选择匹配元素 E 所在文档的根元素。在 HTML 文档中，根元素始终是 html，此时该选择器与 html 类型选择器匹配的内容相同
E F:nth-child(n)	选择父元素 E 的第 n 个子元素 F。其中 n 可以是整数（1、2、3）、关键字（even、odd）、可以是公式（2n+1、−n+5），而且 n 值起始值为 1，而不是 0

（续）

选择器	功能描述
E F:nth-last-child(n)	选择元素 E 的倒数第 n 个子元素 F。此选择器与 E F:nth-child(n) 选择器计算顺序刚好相反，但使用方法都是一样的，其中 :nth-last-child(1) 始终匹配的是最后一个元素，与 :last-child 等同
E:nth-of-type(n)	选择父元素内具有指定类型的第 n 个 E 元素
E:nth-last-of-type(n)	选择父元素内具有指定类型的倒数第 n 个 E 元素
E:first-of-type	选择父元素内具有指定类型 的第一个 E 元素，与 E:nth-of-type(1) 等同
E:last-of-type	选择父元素内具有指定类型的最后一个 E 元素，与 E:nth-last-of-type(1) 等同
E:only-child	选择父元素只包含一个子元素，且该子元素匹配 E 元素
E:only-of-type	选择父元素只包含一个同类型的子元素，且该子元素匹配 E 元素
E:empty	选择没有子元素的元素，而且该元素也不包含任何文本节点

表 2-13 中，只有“ :first-child”属于 CSS2.1，此外其他的结构伪类选择器都是 CSS3 的新特性，为我们提供精确定位到元素的新方式。表中只是告诉我们 CSS3 结构伪类选择器的概念性的知识，有时候看起来不太好理解，针对图 2-21 的 HTML DOM 树型结构，将 CSS3 结构伪类选择器结合起来理解，如图 2-22 所示。

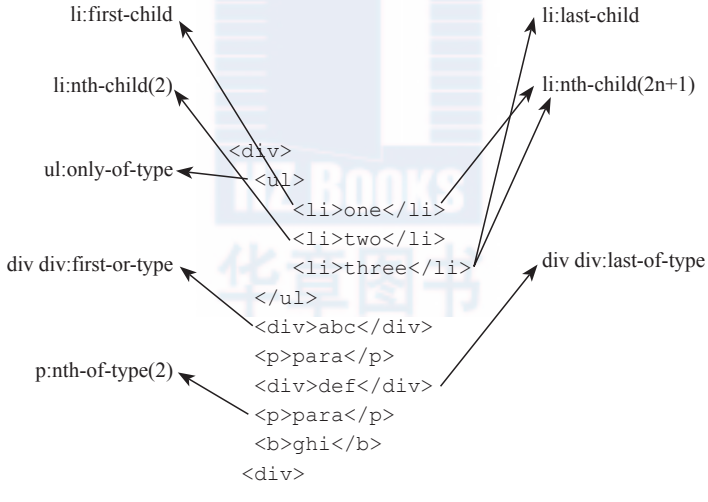


图 2-22　CSS3 结构伪类选择器

2.8.3　浏览器兼容性

CSS3 结构伪类选择器在主流浏览器下运行都非常的完美，只是在 IE 9 以下版本的浏览器中无法正常运行，浏览器兼容性如表 2-14 所示。

在本例中颜色交替只是一个很小的视觉增强，IE 8 及之前版本的用户看不到也无妨，而对于圆角、渐变之类的效果，IE 将直接忽略它们，使用直角或纯色显示，看起来没有任何异常之处。

表 2-14　结构伪类选择器浏览器兼容性

选择器					
E:first-child	9 + √	√	√	√	√
E:last-child	9 + √	√	√	√	√
E:root	9 + √	√	√	√	√
E F:nth-child(n)	9 + √	√	√	√	√
E F:nth-last-child(n)	9 + √	√	√	√	√
E:nth-of-type(n)	9 + √	√	√	√	√
E:nth-last-of-type(n)	9 + √	√	√	√	√
E:first-of-type	9 + √	√	√	√	√
E:last-of-type	9 + √	√	√	√	√
E:only-child	9 + √	√	√	√	√
E:only-of-type	9 + √	√	√	√	√
E:empty	9 + √	√	√	√	√

想为 IE 8 及之前的版本提供一个解决方案，需要用到 JavaScript 脚本。例如要实现例中的 Zebra 表格效果，网上有大量的脚本可以帮你。最适合的脚本还是取决于项目本身，可以通过 Google 搜索“Zebra Stripe JavaScript”、“Zebra Stripe PHP”或者其他更符合需求的关键词。

还可以使用一个脚本为 IE 增加高级选择器的支持，然后使用选择器实现需要的效果。例如 jQuery 脚本库，还有 Dean Edwards 的 ie9.js，此外还有前面介绍的 Selectivizr 脚本[⊖]配合其他 JavaScript 脚本库一起使用。

2.8.4　结构伪类选择器中的 n 是什么

在结构伪类选择器中，有 4 个伪类选择器接受参数 n。

- :nth-child(n)
- :nth-last-child(n)
- :nth-of-type(n)
- :nth-last-of-type(n)

在实际应用中，这个参数 n 可以是整数（1、2、3、4）、关键字（odd、even），还可以是公式（2n+1、−n+5），但参数 n 的起始值始终是 1，而不是 0。换句话说，当参数 n 的值为 0 时，选择器将选择不到任何匹配的元素。

根据上面所述，将结构伪类选择器中的参数按常用的情况划分为七种情形。

1. 参数 n 为具体的数值

这个数值可以是任何大于 0 的正整数，例如“:nth-child(3)”将选择一个系列中的第 3

⊖ 参考 http://selectivizr.com。

个元素。

2. 参数 n 为表达式“n*length”

选择 n 的倍数，其中 n 从 0 开始计算，而 length 为大于 0 的整数。当 length 为整数 1 时，将选择整个系列中的所有元素，直到元素耗尽无法选择为止。因为 length 在实际运用中常为大于 1 的正数，表达式才具有实际意义，例如“:nth-child(2n)”。

❑ n=0 时，2 × 0=0，不选中任何元素；

❑ n=1 时，2 × 1=2，选中系列中的第 2 个元素；

❑ n=2 时，2 × 2=4，选中系列中的第 4 个元素；

❑ n=3 时，2 × 3=6，选中系列中的第 6 个元素；

……

以此类推，直到元素耗尽无法选择为止。

3. 参数 n 为表达式“n+length”

选择大于或等于 length 的元素，例如“:nth-child(n+3)”。

❑ n=0 时，0+3=3，选中系列中的第 3 个元素；

❑ n=1 时，1+3=4，选中系列中的第 4 个元素；

❑ n=2 时，2+3=5，选中系列中的第 5 个元素；

❑ n=3 时，3+3=6，选中系列中的第 6 个元素；

……

以此类推，直到元素耗尽无法选择为止。

4. 参数 n 为表达式“−n+length”

选择小于或等于 length 的元素，例如“:nth-child(−n+3)”。

❑ n=0 时，−0+3=3，选中系列中的第 3 个元素；

❑ n=1 时，−1+3=2，选中系列中的第 2 个元素；

❑ n=2 时，−2+3=1，选中系列中的第 1 个元素；

❑ n=3 时，−3+3=0，不选择任何元素；

❑ n=4 时，−4+3=−1，不选择任何元素；

……

以此类推，当值小于或等于 0 时将不选择任何元素。

5. 参数 n 为表达式“n*length+b”

其中 b 是您想设置的偏移值，其表示隔 length 个元素选中第 n*length+b 个元素，例如“:nth-child(2n+1)”。

❑ n=0 时，2 × 0+1=1，选中系列中的第 1 个元素；

❑ n=1 时，2 × 1+1=3，选中系列中的第 3 个元素；

❑ n=2 时，2 × 2+1=5，选中系列中的第 5 个元素；

❑ n=3 时，3 × 2+1=7，选中系列中的第 7 个元素；

……

以此类推，直到元素耗尽无法选择为止。

6. 参数 n 为关键词“odd”

选择系列中的奇数（1、3、5、7）元素，其效果等同于“ :nth-child(2n−1)”和“ :nth-child(2n+1)”。

7. 参数 n 为关键词“even”

选择系列中的偶数（2、4、6、8）元素，其效果等同于“ :nth-child(2n)”。Sitepoint.com 也制作了一个关于“:nth-child(n)”的参考指南㊀供大家对照参考，如表 2-15 所示。

表 2-15　结构伪类表达式的计算列表

n	2n+1	4n+1	4n+4	4n	5n−2	−n+3
0	1	1	4	—	—	3
1	3	5	8	4	3	2
2	5	9	12	8	8	1
3	7	13	16	12	13	—
4	9	17	20	16	18	—
5	11	21	24	20	23	—

以上几种情形也适用于“:nth-last-child(n)”、“:nth-of-type(n)”和“:nth-last-of-type(n)”三种结构伪类选择器。

尽管上面公式算出来的数值（1、3、5、7）也可以手动给系列元素中的第 1、3、5、7 元素添加对应类名，但这样做不仅耗时费力，还容易遗忘，代码不整洁使维护过程非常痛苦。如果想在已经存在的项目中插入另外一个，不得不对插入处之后的项目全部重新定义新的类名，因为插入后新的序号被打乱。而用 CSS3 的结构伪类选择器“ :nth-child(n)”代替类名，跟踪元素系列的序号变化并自动匹配将会更为准确、高效，而且维护非常方便。

或许会觉得这样的数学计算太麻烦，从而产生对“nth-child(n)”系列结构伪类选择器的抵触。大家不用担心，线上有一些不错的工具，可以通过更改数值为即时查看它是如何影响页面样式的，这将有助于更好地了解“:nth-child(n)”系列结构伪类选择器的作用，例如：

❑ Lea Verou 制作的工具㊁。

❑ Chris Coyier 制作的工具㊂。

❑ Neal Grosskopf 制作的工具㊃。

㊀ 网址为 http://reference.sitepoint.com/css/ understandingnthchildexpressions。

㊁ http://lea.verou.me/demos/nth.html，笔者比较喜欢的一个工具。

㊂ http://css-tricks.com/examples/nth-child-tester/，比较基础，只展示了“:nth-child(n)”的使用效果。

㊃ http://www.nealgrosskopf.com/tech/resources/80/。

2.8.5　结构伪类选择器的使用方法详解

结构伪类选择器是 CSS3 选择器最具特色的一部分内容，同时也是 CSS3 选择器中最出色的一部分内容。通过前面的内容，大家对 CSS3 的结构伪类选择器有了初步的了解，但是完全理解它们，还是需要一定的实例，下面分别介绍 CSS3 的结构伪类选择器的具体使用方法。

为了更好地实例化，先创建一个简单的列表结构，并附上一些简单的样式。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 结构伪选择器的使用 </title>
  <style type="text/css">
    *{
      margin: 0;
      padding: 0;
    }
    ul {
      margin: 50px auto;
      width: 400px;
      list-style: none outside none;
    }
    li {
      display:inline-block;
      margin: 5px;
      padding: 5px;
      width:50px;
      height: 50px;
      font: bold 30px/50px arial;
      background: #000;
      color: #fff;
      border-radius: 50px;
      text-align: center;
    }
  </style>
</head>
<body>
  <ul>
    <li>1</li>
    <li>2</li>
    …
    <li>20</li>
  </ul>
</body>
</html>
```


页面初始效果如图 2-23 所示。

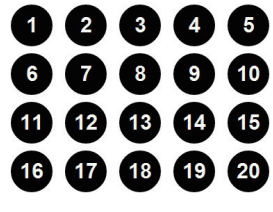


图 2-23　页面初始效果

1. :fist-child 的使用

“:first-child”允许定位某个元素的第一个子元素，例如想改变列表中的第一个“li”的

背景色，代码如下。

```
ul>li:first-child {
  background-color: green;
}
```

在没有这个选择器之前，需要在列表中的第一个“ li”加上一个类名，例如“ first”，然后给添加对应的样式。

```
ul>li.first {
  background-color: green;
}
```

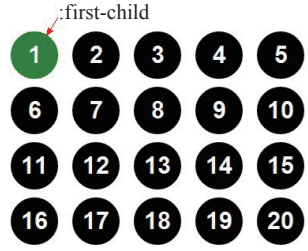


图 2-24　:first-child 效果

其实这两种最终效果是一样的，如图 2-24 所示。后面这种需要在 html 标签中增加一个额外的 class 类名，只是一个地方还好处理，如果是多处都具有这样的效果，给 html 添加类名的方法其弊端明显可见。

在实际项目中这样的运用也是常有的事，例如博客侧边栏的标题“ h2”顶部都有一个“ margin”，用来区分标题和它们前面区块的内容，但是第一个标题“ h2”不需要顶部“margin”值，就可以使用下面的代码。

```
.aside > h2 {
  margin-top: 15px;
}
.aside>h2:first-child {
  margin-top: 0;
}
```

上面看到的是使用“ :first-child”来移除标题顶部的间距，当然也可以用来移除元素底部的间距，此时在不支持“ :first-child”的浏览器中，布局并不会因此而破坏掉，它只会看起来有些不同（顶部或底部间距没清除）。但是，如果使用“ :first-child”来移除一个浮动元素的左边距或右边距，在不支持“:first-child”的浏览器中，布局将会被破坏掉。

2. :last-child 的使用

“ :last-child”与“ :first-child”作用类似，不同的是“ :last-child”选择的是元素最后一个子元素。就拿上面的例子来说，改变列表中最后一个“ li”的背景色，使用这个选择器如下所示。

```
ul>li:last-child {
  background-color: blue;
}
```

和“ :first-child”一样，以前为了实现上面的效果，都是在列表中的最后一个“ li”添加一个类名“ last”，并在此类添加样式，其最终效果都是一样的，如图 2-25 所示。

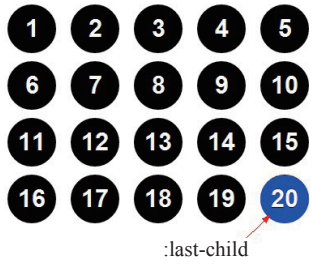


图 2-25　:last-child 效果

在实际的 Web 项目中，“:last-child”的用处也非常广泛。例如在一个导航条中每个导航项都有一个右边框效果，但最后一个不想要这个右边框，此时就可以使用“:last-child”。

```
#nav > li {
  …
  border-right: 1px solid #ccc;
}
#nav > li:last-child {
  border-right: none;
}
```

另一个较常见的就是在博客制作中，假设“post”中最后一段不需要底部“margin”值，代码如下。

```
.post > p:last-child {
  margin-bottom: 0;
}
```

3. :nth-child 的使用

“:nth-child()”用来定位某个父元素的一个或多个特定的子元素。“:nth-child()”可以接受参数 n，而且 n 可以是数值，也可以是表达式和关键词。有关于“n”是什么？大家可以参考“结构伪类选择器中的 n 是什么”。在这一节中，通过实例加深理解“:nth-child(n)”。

(1) :nth-child(3)

参数 n 是一个具体的整数值，例如“:nth-child(3)”表示选择某元素下的第 3 个子元素，（这里的整数 3 可以根据自己的需要来定义）。接上面的实例，如果需要改变列表项中第 3 个“li”元素的背景色，就可以直接这样使用。

```
ul>li:nth-child(3){
  background-color: yellow;
}
```

这样一来，列表中的第 3 个 li 的背景就变成黄色了，如图 2-26 所示。

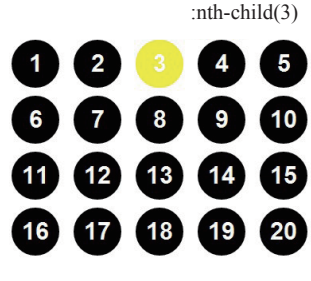


图 2-26　:nth-child(3) 效果

上面仅是列表中存在 li 一种子元素，但是如果列表中第 3 个 li 之前还有其他的子元素，例如 DIV 元素（当然这样写 HTML 是一种不规范的写法，此处仅用来说明问题，不提倡这样去写 HTML 结构），“ul>li:nth-child(3)”还会选中列表中的第 3 个 li 元素吗？先不做任何回答，改变一下实例的代码。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title></title>
```

```
  <style type="text/css">
    ul {
      list-style: none outside none;
      padding: 10px;
      background: green;
      width: 400px;
    }
    li {margin-bottom: 10px;border: 1px solid orange;}
    div {margin-bottom: 10px;border: 1px solid blue;}
  </style>
</head>
<body>
  <ul>
    <li>1</li>
    <li>2</li>
    <div>div</div>
    <div>div</div>
    <li>3</li>
    …
    <li>10</li>
  </ul>
</body>
</html>
```

页面的初步效果如图 2-27 所示。

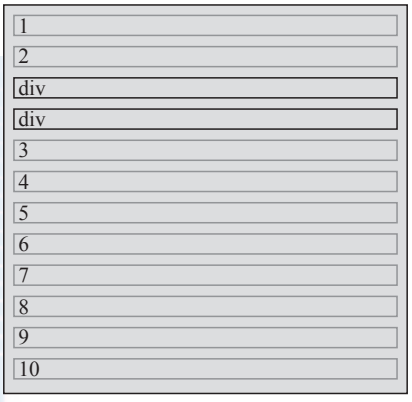


图 2-27　页面初步效果

同样，加上以下代码。

```
ul>li:nth-child(3){
  background-color: yellow;
}
```

保存上面样式后刷新浏览器，并没有得到想要的效果，列表中的第 3 个 li 背景色没有为此改变。因为在 ul 里面有其他类型的元素（不是 li），它也会算作是列表的子元素。就上面的实例，li:nth-child(3) 并不存在，列表的第 3 个子元素是 div 而不是 li，此时如果还想改变第 3 个列表的背景色，就必须改变“:nth-child()”中的值（或者使用 :nth-of-type()）。拿这个实例来说，第 3 个 li 是列表中的第 5 个子元素，将上面的代码改为以下形式。

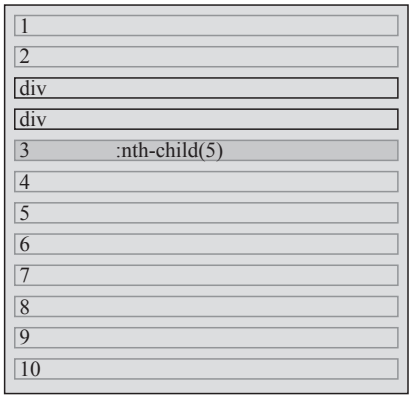


图 2-28　:nth-child(5) 改变第 3 个 li 背景效果

```
ul>li:nth-child(5){
  background-color:yellow;
}
```

接下来用效果来证明是不是需要的效果，如图 2-28 所示。

通过上面两个实例的对比，大家清楚“ul>li:nth-child(3)”表达的并不是一定选择列表 ul 元素中的第 3 个

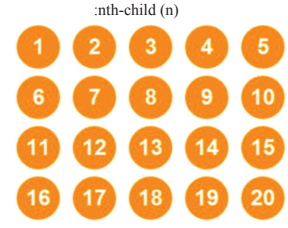
子元素 li，仅有列表 ul 中第 3 个 li 元素前不存在其他的元素，命题才有意义，否则就会造成开始的问题，不会改变列表第 3 个 li 元素的背景色。

(2) :nth-child(n)

参数 n 是一个简单的表达式，取值从 0 开始计算的，到什么时候结束不知道，如果在实际应用中直接这样使用，将会选中父元素中的所有子元素。接着前面列表的案例，在代码中加入以下代码。

```
ul>li:nth-child(n){
  background-color: orange;
}
```

效果如图 2-29 所示。



☆图 2-29　:nth-child(n) 效果

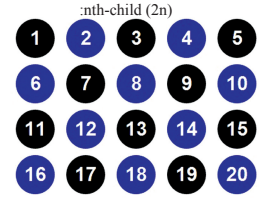
注意　“:nth-child(n)”中参数只能是 n，不可以使用其他的字母代替，不然会没有任何效果。

(3) :nth-child(2n)

该选择器是“:nth-child(n)”的一种变身，可以选择 n 的 2 倍数，当然其中“2”可以换成需要的数字。

```
ul>li:nth-child(2n){
  background-color: blue;
}
```

这样列表中的偶数项都将被选中，其效果如图 2-30 所示。



☆图 2-30　:nth-child(2n) 效果

这个时候大家有没有想到关键词“even”呢？是的，“:nth-child(2n)”和使用关键词“:nth-child(even)”得到的效果是一样的。它们在实际项目中用来制作一些突出效果是非常方便的。例如制作斑马线表格，给偶数行设置一个不同的背景色。

```
table tr:nth-child(even){...}
```

或者：

```
table tr:nth-child(2n){...}
```

(4) :nth-child(2n+1)

该选择器是在“:nth-child(2n)”基础上演变过来的，前面说了“:nth-child(2n)”和“:nth-child(even)”是选择偶数，就可以在其基础上加 1 或减 1，将偶数变成奇数。例如：

```
ul>li:nth-child(2n+1){
  background-color: blue;
}
```

这样列表中奇数列背景色都将变成蓝色，如图 2-31 所示。

“:nth-child(2n+1)”和“:nth-child(2n−1)”选择的是父元素中排序为奇数的子元素，同时选择奇数还可以直接使用关键词“odd”来实现（“:nth-child(odd)”）。换句话说，“:nth-child(odd)”、“:nth-child(2n+1)”和“:nth-child(2n−1)”三个选择器定位的元素是完全相同的。

使用奇数选择器制作斑马线表格是很常见的，不同之处是改变的是表格奇数行的背景色，当然也可以同时使用偶数和奇数，制作一个靓丽的斑马线表格。

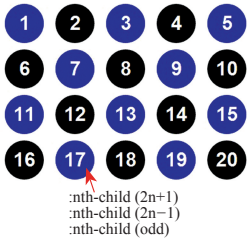
```
table tr:nth-child(odd) {...}/* 设置表格奇数行样式 */
table tr:nth-child(even){...}/* 设置表格偶数行样式 */
```

(5) :nth-child(n+5)

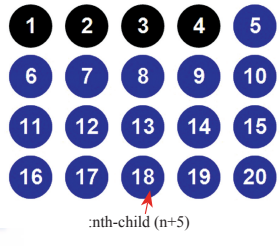
这个选择器从父元素中的第 5 个子元素开始选择，这里的数字可以自己定义。例如：

```
ul>li:nth-child(n+5){
  background-color: blue;
}
```

此时列表中第 5 个 li 元素开始直到最后一个 li 元素，其背景都将变成蓝色，如图 2-32 所示。



☆图 2-31　:nth-child(2n+1) 效果



☆图 2-32　:nth-child(n+5) 效果

例如在博文有一个“今日焦点”区块，想让今日点击量落后于第三的文章序列号标注红色，此时这个选择器就非常的方便了，如图 2-33 所示。

只要将选择器中的数字变成需要的即可。

```
.post-hot-today li:nth-child(n+4){color:red;}
```

(6) :nth-child(−n+5)

该选择器刚好和“:nth-child(n+5)”相反，选择父元素中第 1 个到第 5 个子元素，如果不太清楚，只要把两个表达式的计算值对比就非常清楚了。同样通过“:nth-child(−n+5)”来选择列表中前 5 个子元素 li。

图 2-33　点击量未进前三的序列号标注红色

```
ul>li:nth-child(-n+5){
  background-color: blue;
}
```

上面代码刚好与图 2-32 效果相反，如图 2-34 所示。

同样，想在博客“热门博文”区块内把排在前三的文章进行特殊标注，如图 2-35 所示。

选择器的使用如下所示。

```
.post-hot li:nth-child(-n+3){...}
```

(7) :nth-child(4n+1)

选择器实现某父元素的子元素隔几选一的效果，例如“:nth-child(4n+1)”实现的就是隔 3 个子元素选中一个子元素，直到所有元素耗尽为止。其中“4n+1”可以根据自己的需求进行修改。在前面实例基础上，使用“:nth-child(4n+1)”来改变 li 的效果色为蓝色。

```
ul > li:nth-child(4n+1){
  background-color: blue;
}
```

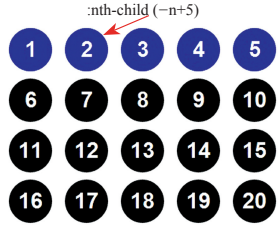
此时整个列表项的第 1、5、9、13、17 项背景色变成蓝色，如图 2-36 所示。

上面展示的是“:nth-child”结构伪类选择器的几种使用方法，大家在实际应用中根据需求使用不同的表达式，从而满足需求。

4. :nth-last-child 的使用

“:nth-last-child”和“:nth-child”相似，只是这里多了一个“last”，作用和“:nth-child”有所区别。从某父元素的最一个子元素开始计算来选择特定的元素。接着上面的实例代码，来看“:nth-last-child”选择器的实例。

```
ul>li:nth-last-child(4){
  background-color: blue;
}
```



☆图 2-34　:nth-child(−n+5) 效果

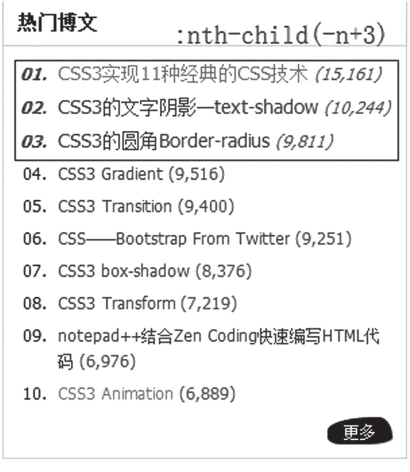
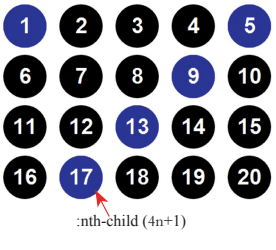


图 2-35　热点博文排在前三的特殊效果

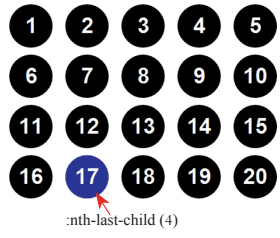


☆图 2-36　:nth-child(4n+1) 效果

上面的代码选择了 ul 列表中倒数第 4 个 li 元素，如图 2-37 所示。

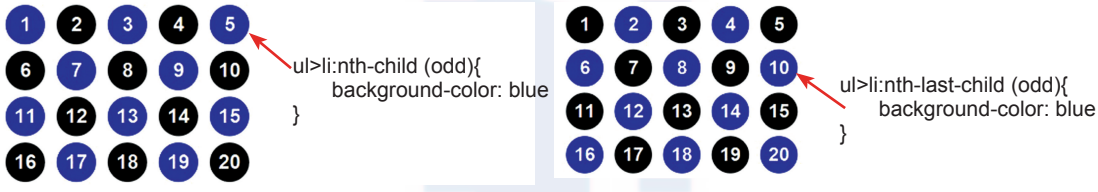
“:nth-last-child”可以像“:nth-child”一样使用表达式、关键词等，不同的是，“:nth-last-child”起始位置是从父元素的最后一个子元素开始计算。注意“:nth-last-child”使用关

键词 odd、even，由于起始位置不同，“ :nth-last-child”使用关键词 odd 和 even 所得到的顺序刚好相反。

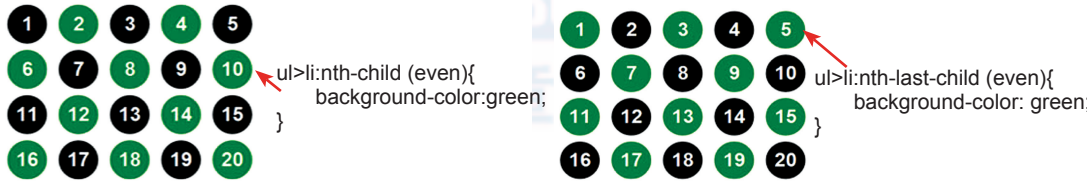


☆图 2-37　:nth-last-child(4) 效果

简单来说，使用“ :nth-child(odd)”选择的是奇数项，而使用“ :nth-last-child(odd)”选择的却是偶数项，如图 2-38 所示。同样，使用“ :nth-child(even)”选择的是偶数项，而使用“ :nth-last-child(even)”选择的是奇数项，如图 2-39 所示。



☆图 2-38　:nth-child(odd) 与 :nth-last-child(odd) 对比



☆图 2-39　:nth-child(even) 和 :nth-last-child(even) 对比

可知，“ :nth-child(odd)”与“ :nth-last-child(even)”选择的元素是相同的，而“ :nth-child(even)”和“ :nth-last-child(odd)”选择的元素相同。

扩展阅读　“:nth-child”与“:nth-last-child”的区别

“ :nth-child”和“ :nth-last-child”两个选择器的使用方法和所起的作用是一样的，用来选择某父元素中的特定子元素，同时所有的子元素不分类型，而且所有出现的子元素都会按文档流的先后顺序来排序。同时它们之间有一个很明显的区别。

- “ :nth-child”选择器选择的子元素是从第一个子元素开始算起；
- “ :nth-last-child”选择器选择的子元素却是从最后一个子元素开始算起。

5. :nth-of-type 的使用

“:nth-of-type”和“:nth-child”类似，不同的是它只计算父元素中指定的某种类型的子元素。当某个元素中的子元素不单单是同一种类型的子元素这，使用“ :nth-of-type”选择器来定位于父元素中某种类型的子元素是非常的方便和有用。正如前面看到的实例，当 ul 列表中不仅仅有子元素 li，而且还有别的子元素是 DIV，使用 ul>li:nth-child（3）来选择第 3 个 li 元素时，无法选择，如图 2-27 所示。通过改变选择器“ :nth-child”的变量参数值为 5，才可以选中列表中的第 3 个子元素 li，如图 2-28 所示。但是使用“ :nth-of-type”，就不用改变其变量参数值。

```
ul>li:nth-of-type(3){
  background-color:orange;
}
```

效果如图 2-40 所示。

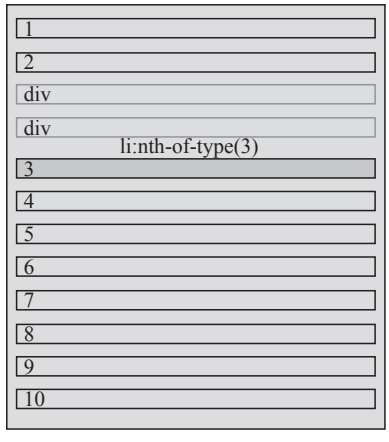


图 2-40　:nth-of-type(3) 效果

同“ :nth-child”的具体使用方法一样，“ :nth-of-type”也具有参数设置，这个参数也是 n，它可以是具体的整数值，也可以是表达式和关键词。

在 Web 应用中，“ :nth-of-type”在以下场景中可以使用。

- ❏ 营造一种有随意感的界面，例如改变每张图片的旋转角度；
- ❏ 使文章中的图片交替着向左向右浮动；
- ❏ 为一篇文章的头一段设置不同的样式，例如首字下沉；
- ❏ 为一个定义列表的条上使用交替样式；
- ❏ 制作图表。

此外，还有更多的使用场景。一起来看看其中几个场景下的具体使用。

在 Web 页面的设计中，常常给某种类型元素设置一些特殊的样式，以达到突出的目的。例如，将一篇文章第一个段落的文字设置大一些，以前通常是在第一个段落 P 添加一个类名“ first”，然后设置其字号比别的段落大一些。使用“ :nth-of-type”选择器，就能轻松的改变文章中第一个段落的字号。

```
.node p:nth-of-type(1){
  font-size: 1.5em;
}
```

也许想在一篇文章中把奇数图片左对齐，偶数图片右对齐，如图 2-41 所示。

首先想到给文章中的图片加一个 left 或 right 类名，然后分别给这两个不同的类名定义不同的对齐方式。这样的做法毫无疑问是正确的，但现在说最佳的做法是使用“ :nth-of-type”来实现。

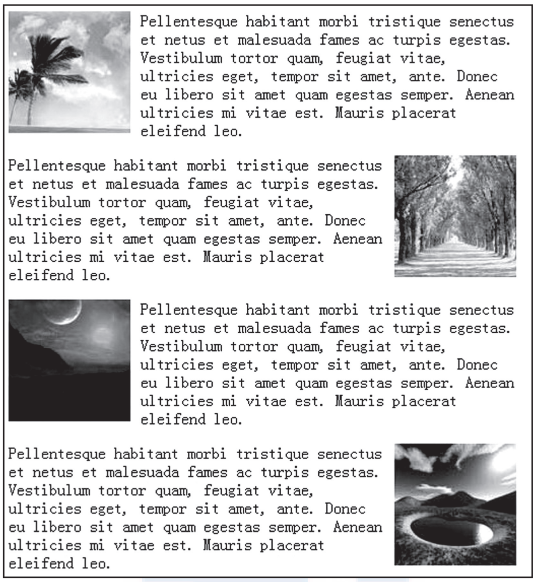


图 2-41　图片对齐效果

```
.article img:nth-of-type(odd){
  float: left;
  margin-right: 10px;
}
.article img:nth-of-type(even){
  float: right;
  margin-left: 10px;
}
```

它在 IE 8 及以下浏览器是不被支持的，但可以借助 JavaScript 库来实现，例如 Keith Clark 编写的 Selectivizr 脚本[1]，或者直接无视不支持的浏览器，例如 Simon Foster[2]就使用了“:nth-of-type”为它收集的 45RPM 唱片制作的一份漂亮的图表，采用“:nth-of-type”为每种不同的流派设置一个不同的背景图片。下面是 Simon Foster 站上截取的一段代码。

```
ul#genre li:nth-of-type(1) {
   width:32.9%;
  background:url(images/orangenoise.jpg);
}
ul#genre li:nth-of-type(2) {
   width:15.2%;
  background:url(images/bluenoise.jpg);
}
ul#genre li:nth-of-type(3) {
  width:13.1%;
```

[1] 详见 http://selectivizr.com。

[2] 详见 http://www.fortherecord.simonfosterdesign.com/。

```
  background:url(images/greennoise.jpg);
}
```

图 2-42 是从站点截取的效果。

图 2-42 “For The Record”站点使用 :nth-of-type 的效果

扩展阅读 “:nth-child”和“:nth-of-type”的区别

“:nth-child”和“:nth-of-type”两者之间的区别对于初学者来说很是头痛的问题，为了更好地帮助大家区分使用方法，特意在此加以区分。首先创建一个简单的 HTML 结构。

```
<div class="post">
  <p> 我是文章中的第一个段落 </p>
  <p> 我是文章中的第二个段落 </p><!--  == 我要变成红色的字 == -->
</div>
```

接下来，使用“:nth-child”和“:nth-last-child”选择段落并改变其文字颜色。

```
.post>p:nth-child(2){color:red;}
.post>p:nth-of-type(2){color:red;}
```

上面的代码都把“post”中的第二段文字变成大红色，是不是代表这两个选择器就是一样的呢？其实不然。“:nth-child”仅从字面上来解释，其包含了两层意思。首先是一个段落元素，而且这个段落是父元素“DIV”的第二个子元素；而“:nth-of-type”从字面上解释是“选择父元素 DIV 的段落二”。

上面一段话看来很晕，有没有更好的方法来区分它们呢？有的，把上面的 HTML 结构改变一下，在两个段落前加上一个标题“h1”。

```
<div class="post">
```

```
  <h1> 我是标题一 </h1>
  <p> 我是文章中的第一个段落 </p>
  <p> 我是文章中的第二个段落 </p> <!--  == 我要变成红色的字 == -->
</div>
```

前面的样式不变，但结构却完全不同了。“p:nth-child(2)”并没有选择段落二，而是选择了段落一，从而也就没有达到需要的效果。

```
.post>p:nth-child(2){color: red;} /* 第一个段落变成红色，不是我们需要的效果 */
```

“p:nth-child(2)”选错了段落，但“p:nth-of-type(2)”却工作正常，选择的还是段落二，实现想要的效果。

```
.post>p:nth-of-type(2){color:red;}/* 改变段落二文字色为红色，是我们需要的效果 */
```

如果在“h1”标题后面添加一个“h2”标题，此时“p:nth-child(2)”将无法选择任何元素，因为此时“DIV”的第二个子元素并不是段落一“P”，所以无法选择任何元素。但“p:nth-of-type(2)”依然能正常工作，因为选择的始终是“DIV”中的段落二“P”。

大家只需要记住一点：“:nth-child”选择的是某父元素的子元素，这个子元素并没有指定确切的类型，同时满足两个条件时方能有效果，其一是子元素，其二此子元素刚好处在那个位置；“:nth-of-type”选择的是某父元素的子元素，而且这个子元素是指定类型。

“:nth-child”虽然常见，但却脆弱，正如前面的示例所示，随时被其他子元素给挤出选择的范围。而“:nth-of-type”不常见，但在选择某种类型的子元素时，更稳定，更可靠。

6. :nth-last-of-type 的使用

“:nth-last-of-type”和“:nth-of-type”一样，用来选择父元素中指定的某种子元素类型，但它的起始方向是从最后一个子元素开始，而且使用方法可以像前面提到的“:nth-last-child”一样使用。

例如选择一篇文章的最后一个段落，会很快地想到“:last-child”和“:nth-last-child(1)”这两个选择器，但是文章中并不常常都是以段落来结束，这个时候“:nth-last-of-type”就能快速、准确定位到最后一个段落。

```
.article p:nth-last-of-type(1){...}
```

也可以更加的聪明一些，在一个块级选择器中**结合多种这样的伪类选择器**。举个例子，让文章中除了第一个和最后一个的所有图片左浮动，这个时候多种伪类的选择器就非常的实用。

```
.article img:nth-of-type(n+2):nth-last-of-type(n+2){
  float:left
}
```

在这个组合型选择器中，第一部分“:nth-of-type(n+2)”从第 2 个图片开始定位，选择了文章中所有图片，除了第一张；第二部分“:nth-last-of-type(n+2)”从文章中倒数第 2 张

图片开始定位，选择了文章中所有图片，除了最后一张。因为这两个选择器并非互相排斥的，可以同时使用它们，这样就可以立刻排除第一张和最后一张图片，达到想要的选择效果。

扩展阅读　“:nth-of-type”和“:nth-last-of-type”的区别

“:nth-of-type”和“:nth-last-of-type”都是结构伪类选择器，它们的作用也是一样的，都是用来选择某父元素中指定类型的子元素，区别就是，“:nth-of-type”选择的某类型子元素是从前往后排序计算，而“:nth-last-of-type”选择的某类型子元素是从后向前排序计算。

7. :first-of-type 和 :last-of-type 的使用

“:first-of-type”和“:last-of-type”这两个选择器类似于“:first-child”和“:last-child”，不同之处就是指定了元素的类型。换句话说，“:first-of-type”是用来定位一个父元素下的某个类型的第一个子元素；而“:last-of-type”用来定位一个父元素下的某个类型的最后一个子元素。

通过前面的学习了解到，“:nth-of-type(1)”用来选择父元素指定类型第一个元素，其实可以使用“:fist-of-type”来代替。同样，可以使用“:last-of-type”来代替“:nth-last-of-type(1)”。

8. :only-child 的使用

“:only-child”表示一个元素是它父元素的唯一子元素。换句话说，匹配元素的父元素中仅有一个子元素。来看一个简单的例子，在 post 列表中，有的只有一个段落，有的不只一个段落。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>:only-child 的使用 </title>
  <style type="text/css">
    .post {
      width: 300px;
      margin: 20px auto;
      padding: 5px;
      border: 1px solid #ccc;
    }
    p {
      background: green;
      color: #fff;
      border: 1px solid orange;
      padding: 5px;
    }
  </style>
</head>
```

```
<body>
  <div class="post">
    <p> 我是第一个段落 </p>
    <p> 我是第二个段落 </p>
  </div>
  <div class="post">
    <p> 我就一个段落 </p>
  </div>
</body>
</html>
```

上面的实例中运用了一些初步样式，其初步页面效果如图 2-43 所示。

下面是关键时候了，使用“:only-child”看看两个 post 中的段落 p 会有什么样的变化。

```
.post>p:only-child {
  border-width:2px;
  background-color:#000;
}
```

使用“:only-child”后的效果如图 2-44 所示。

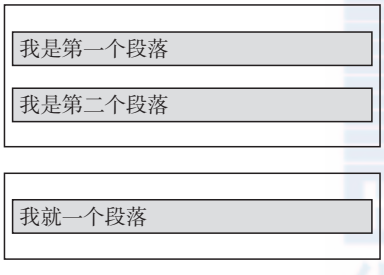


图 2-43　页面初步效果

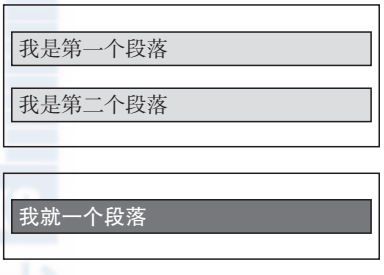


图 2-44　:only-child 器使用效果

很明显的两个列表中只有一个段落 p 的改变了样式，也就是说“ :only-child”仅能匹配父元素中一个子元素，而且这个子元素是唯一的。

9. :only-of-type 的使用

“ :only-of-type”用来选择一个元素是它的父元素的唯一一个相同类型的子元素。这样说或许不太好理解。换一种说法，“ :only-of-type”表示一个元素有很多个子元素，而其中只有一个子元素是唯一的，使用“:only-of-type”就可以选中这个唯一类型子元素。

Devsnippet[⊖]制作了一个 demo，只有一张图片与一个容器中有多张图片的不同样式风格，下面代码是从 demo 中截取的。

```
div > img {
   background:#DCE8F5 none repeat scroll 0 0;
   border:3px solid #96C8E5;
   float:left;
```

⊖ 链接为 http://devsnippets.com/article/5-advanced-css-pseudo-class.html。

```
    margin:5px;
    padding:5px;
}
div > img:only-of-type {
  border:3px solid #E71F58;
  float:none;
  margin:10px;
  padding:10px;
}
```

从代码中可以明显的得知，div 中的图片左浮动，但 div 中仅有一图片类型时，此图片样式不浮动，而且还将改变对应的外边距和内边距值，效果如图 2-45 所示。

图 2-45　Devsnippet 实例效果

10. :empty 的使用

“:empty”用来选择没有任何内容的元素，这里“没有任何内容”指的是一点内容都没有，哪怕是一个空格。这个选择器用来处理动态输出内容方面非常方便。例如想高亮提示用户搜索出来的结果为空时，就可以这样使用。

```
#results:empty{background-color:#fcc;}
```

2.8.6　实战体验：CSS3 美化表格

对于数量大的表格，普通的设计极易影响用户的阅读体验。例如长时间地阅读数据量大的表格容易引起视觉的疲劳，会诱发错行误读等问题。因此，Web 设计师要设计一个表格，不仅需要考虑表格的外观，而且要提高用户的体验。

Zebra 是经典的数据表格设计样式，主要从易用性的角度来考虑，以提高用户浏览数据的速度和准确度。传统的做法是在表格的行中分奇数和偶数，为相应的行添加类名，然后通过类名来控制表格单元格的背景色。当然，这种设计给前端工作人员带来很多不便之处，例如动态插入行，就需要重新为行设置类名，也给维护带来很多困难。

然而采用 CSS3 结构伪类选择器后，一切都是那么简单而轻松。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 美化表格 </title>
  <style type="text/css">
*{margin: 0;padding: 0;}
body {
  padding: 40px 100px;
}
.demo {
```

```
  width: 600px;
  margin: 40px auto;
  font-family: 'trebuchet MS', 'Lucida sans', Arial;
  font-size: 14px;
  color: #444;
}
/* 表格的默认设置 */
table {
  *border-collapse: collapse; /* IE 7 and lower */
  border-spacing: 0;
  width: 100%;
}

/*======== 制作圆角表格 ========*/
.bordered {
  border: solid #ccc 1px;     /* 给表格添加边框 */
  border-radius: 6px;         /* 设置表格圆角 */
  box-shadow: 0 1px 1px #ccc;/* 表格阴影设置 */
}
.bordered tr {
  -o-transition: all 0.1s ease-in-out;
  -webkit-transition: all 0.1s ease-in-out;
  -moz-transition: all 0.1s ease-in-out;
  -ms-transition: all 0.1s ease-in-out;
  transition: all 0.1s ease-in-out;
}
.bordered .highlight,
.bordered tr:hover {
  background: #fbf8e9;/* 表格行的悬浮状态效果 */
}
.bordered td,
.bordered th {
  border-left: 1px solid #ccc;
  border-top: 1px solid #ccc;
  padding: 10px;
  text-align: left;
}
.bordered th {
  /* 表格表头添加渐变背景色 */
  background-color: #dce9f9;
  background-image: -webkit-gradient
       (linear, left top, left bottom, from(#ebf3fc), to(#dce9f9));
  background-image: -webkit-linear-gradient(top, #ebf3fc, #dce9f9);
  background-image: -moz-linear-gradient(top, #ebf3fc, #dce9f9);
  background-image: -ms-linear-gradient(top, #ebf3fc, #dce9f9);
  background-image: -o-linear-gradient(top, #ebf3fc, #dce9f9);
  background-image: linear-gradient(top, #ebf3fc, #dce9f9);
  filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0,
           startColorstr=#ebf3fc, endColorstr=#dce9f9);
  -ms-filter: "progid:DXImageTransform.Microsoft.gradient (GradientType=0,
           startColorstr=#ebf3fc, endColorstr=#dce9f9)";
```

```
  box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;/* 表格表头设置内阴影 */
  border-top: none;
  text-shadow: 0 1px 0 rgba(255,255,255,.5);/* 表格表头设置文字阴影 */
}
/* 使用 :first-child 去除表格每行的第一个单元格的左边框 */
.bordered td:first-child,
.bordered th:first-child {
  border-left: none;
}
/* 使用 :first-child 设置表格表头第一个单元格仅左上角为圆角 */
.bordered th:first-child {
  border-radius: 6px 0 0 0;
}
/* 使用 :last-child 设置表格表头最后一个单元格仅右上角为圆角 */
.bordered th:last-child {
  border-radius: 0 6px 0 0;
}
/* 使用 :first-child 和 :last-child 设置表格最后一行的第一个单元格左下角为圆角 */
.bordered tr:last-child td:first-child {
  border-radius: 0 0 0 6px;
}
/* 使用 :last-child 设置表格最后一行的最后一个单元格右上角为圆角 */
.bordered tr:last-child td:last-child {
  border-radius: 0 0 6px 0;
}

/*======= 制作 Zebra 表格（斑马线表格）效果 ==========*/
.zebra td,
.zebra th {
  padding: 10px;
  border-bottom: 1px solid #f2f2f2;
}
/* 使用 :nth-child(even) 给表格的奇数行添加背景和阴影效果 */
.zebra .alternate,
.zebra tbody tr:nth-child(even) {
  background: #f5f5f5;
  box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
}
.zebra th {
  text-align: left;
  text-shadow: 0 1px 0 rgba(255,255,255,.5);
  border-bottom: 1px solid #ccc;
  background-color: #eee;
  background-image: -webkit-gradient(linear,
            left top, left bottom, from(#f5f5f5), to(#eee));
  background-image: -webkit-linear-gradient(top, #f5f5f5, #eee);
  background-image: -moz-linear-gradient(top, #f5f5f5, #eee);
  background-image: -ms-linear-gradient(top, #f5f5f5, #eee);
  background-image: -o-linear-gradient(top, #f5f5f5, #eee);
  background-image: linear-gradient(top, #f5f5f5, #eee);
  filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0,
```

```
            startColorstr=#f5f5f5, endColorstr=#eeeeee);
  -ms-filter: "progid:DXImageTransform.Microsoft.gradient (GradientType=0,
            startColorstr=#f5f5f5, endColorstr=#eeeeee)";
}
/* 使用 :first-child 设置表格表头第一个单元格左上角为圆角 */
.zebra th:first-child {
  border-radius: 6px 0 0 0;
}
/* 使用 :last-child 设置表格表头最后一个单元格右上角为圆角 */
.zebra th:last-child {
  border-radius: 0 6px 0 0;
}
.zebra tfoot td {
  border-bottom: 0;
  border-top: 1px solid #fff;
  background-color: #f1f1f1;
}
/* 使用 :first-child 设置表格脚部第一个单元格左下角为圆角 */
.zebra tfoot td:first-child {
  border-radius: 0 0 0 6px;
}
/* 使用 :last-child 设置表格脚部最后一个单元格右下角为圆角 */
.zebra tfoot td:last-child {
  border-radius: 0 0 6px 0;
}
  </style>
</head>
<body>
…
</body>
</html>
```

通过上面的代码，在现代浏览器中就能看到两个美化后的表格，如图 2-46 所示。

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966

图 2-46　CSS3 美化表格

图 2-46 的表格效果是不是很清晰。我们需要知道这样的表格是怎么制作出来的，下面就一起来细化以上代码。

以前表格的圆角效果是通过图片来模拟，制作相当麻烦。有了 CSS3 后，这些都变得那么容易，只要使用 border-radius 就可以（这个属性后面章节会详细介绍）。制作表格圆角时还有一个技巧，在制作表格圆角效果之前，有必要先完成一步。border-collapse 的默认值是 separate，需要将其设置为 0，如下所示。

```
table{
  *border-collapse: collapse;/*IE 7 and lower*/
   border-spacing:0;
}
```

接下来一起看表格圆角实现的代码。

```
/*== 整个表格设置了边框，并设置了圆角 ==*/
.bordered {
  border: solid #ccc 1px;
  border-radius: 6px;
}
/*== 表格头部第一个 th 需要设置一个左上角圆角 ==*/
.bordered th:first-child {
  border-radius: 6px 0 0 0;
}
/*== 表格头部最后一个 th 需要设置一个右上角圆角 ==*/
.bordered th:last-child {
  border-radius: 0 6px 0 0;
}
/*== 表格最后一行的第一个 td 需要设置一个左下角圆角 ==*/
.bordered tr:last-child td:first-child {
  border-radius: 0 0 0 6px;
}
/*== 表格最后一行的最后一个 td 需要设置一个右下角圆角 ==*/
.bordered tr:last-child td:last-child {
  border-radius: 0 0 6px 0;
}
```

在 table 中设置一个边框，为了让表格具有圆角效果，需要在表格四个角的单元格上分别设置圆角效果，并且其圆角的半径弧度与表格的圆角半径弧度大小一样。反之，如果在 table 上没有设置边框，只需要在表格四个角的单元格设置圆角，就能实现圆角效果，例如 Zebra 表格。

```
/*== 表格头部第一个 th 需要设置一个左上角圆角 ==*/
.zebra th:first-child {
  border-radius: 6px 0 0 0;
}
/*== 表格头部最后一个 th 需要设置一个右上角圆角 ==*/
.zebra th:last-child {
```

```
  border-radius: 0 6px 0 0;
}
/*== 表格最后一行的第一个 td 需要设置一个左下角圆角 ==*/
.zebra tfoot td:first-child {
  border-radius: 0 0 0 6px;
}
/*== 表格最后一行的最后一个 td 需要设置一个右下角圆角 ==*/
.zebra tfoot td:last-child {
  border-radius: 0 0 6px 0;
}
```

例中的表格除了使用了 CSS3 的圆角效果之外，还使用了 box-shadow 制作表格的阴影效果，使用 text-shadow 制作表格表头的文字阴影效果，使用 transition 制作 hover 悬浮高亮的过渡效果，以及使用 gradient 制作表头的渐变效果。这些属性还不清楚如何使用，大家不必太担心，本书后续章节会一一介绍。

2.9 否定伪类选择器

否定选择器“ :not()”是 CSS3 的新选择器，类似 jQuery 中的“ :not()”选择器，主要用来定位不匹配该选择器的元素。

2.9.1 否定伪类选择器语法

“ :not()”是一个非常有用的选择器，可以起到过滤内容的作用。语法如表 2-16 所示。

表 2-16 否定伪类选择器的使用语法

选择器	功能描述
E:not(F)	匹配所有除元素 F 外的 E 元素

否定选择器作用非常大，例如以下选择器表示选择页面中所有元素，除了“ footer”元素之外。

```
:not(footer){...}
```

有时候常在表单元素中使用，举个实例，给表单中所有 input 定义样式，除了 submit 按钮之外，此时就可以使用否定选择器。

```
input:not([type=submit]){...}
```

类似这样的选择器在移动端使用也是常见的，例如在 Web 移动页面中，给表单中的 input 定义样式，除了单选择按钮之外，代码如下所示。

```
fieldset input:not([type=radio] {
  margin:0;
  width:265px;
  font-size: 18px;
  border-radius: 0;
  border-bottom: 0;
```

```
  border-color: #999;
  padding: 8px 10px;
}
```

2.9.2　浏览器兼容性

浏览器兼容性如表 2-17 所示。

表 2-17　否定伪类选择器的兼容性

选择器					
E:not(F)	9 + √	√	√	√	√

2.9.3　实战体验：改变图片效果

本节的实例是通过否定选择器来改变图片墙中图片，用来区分悬浮状态下的效果。

这个实例中采用两种技术，一种是图片的过滤效果（CSS3 中的新特性，现在仅 Webkit 内核浏览器支持，本例子中不详细介绍），第二种技术就是前面介绍的否定选择器“:not”。

当鼠标悬浮在整个图片墙上时，所有图片通过自定义的过滤特性，变成黑白模糊的效果，当鼠标移动到一张图片上时，图片恢复到默认效果，而其他图片保持黑白模糊效果，如图 2-47 所示。

图 2-47　否定选择器制作图片墙

制作原理非常简单，接下来一起来看看如何实现。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>:not 选择器使用 </title>
  <style type="text/css">
    *{
      margin: 0;
      padding: 0;
    }
    ul {
      width: 690px;
      margin: 20px auto;
      letter-spacing: -4px;
      word-spacing: -4px;
```

```
      font-size: 0;
    }
    li {
      font-size: 16px;
      letter-spacing: normal;
      word-spacing: normal;
      display:inline-block;
      *display: inline;
      zoom:1;
      list-style: none outside none;
      margin: 5px;
    }
    img {
      width: 128px;
      height: 128px;
    }
    .gallery li:nth-child(2){
      -webkit-filter:grayscale(1);
    }
    .gallery li:nth-child(3){
      -webkit-filter:sepia(1);
    }
    .gallery li:nth-child(4){
      -webkit-filter:saturate(0.5);
    }
    .gallery li:nth-child(5){
      -webkit-filter:hue-rotate(90deg);
    }
    .gallery li:nth-child(6){
      -webkit-filter:invert(1);
    }
    .gallery li:nth-child(7){
      -webkit-filter:opacity(0.2);
    }
    .gallery li:nth-child(8){
      -webkit-filter:blur(3px);
    }
    .gallery li:nth-child(9){
      -webkit-filter:drop-shadow(5px 5px 5px #ccc);
    }
    .gallery li:nth-child(10){
       -webkit-filter: saturate(6) hue-rotate(361deg) brightness(.09);
    }
    .gallery:hover li:not(:hover){
       -webkit-filter: blur(2px) grayscale(1);
      opacity: .7;
    }
  </style>
</head>
<body>
   <ul class="gallery">
```

```
    <li>
      <img alt="" src="http://www.w3cplus.com/sites/default/files/filter.jpg" />
    </li>
    ...
    <li>
       <img alt="" src="http://www.w3cplus.com/sites/default/files/filter.jpg" />
    </li>
  </ul>
</body>
</html>
```

整个案例中，通过“:nth-child()”给每个图片设置 filter 效果，关键部分是使用“:not()”过滤了除悬浮状态“(:hover)”的图片，其他都变成黑白模糊透明度为 70% 的效果。

```
.gallery:hover li:not(:hover){
  -webkit-filter: blur(2px) grayscale(1);
  opacity: .7;
}
```

不过在写本书时，仅有 Webkit 内核的浏览器支持 filter 属性。当浏览器不支持 filter，但支持“ :not()”，能看到除悬浮状态图片以外的所有图片透明度变成 70% ；如果对否定选择器也不支持，将看不到任何的效果。

2.10　伪元素

除了伪类，CSS3 还支持访问伪元素。伪元素可用于定位文档中包含的文本，但无法在文档树中定位。伪类一般反映无法在 CSS 中轻松或可靠地检测到的某个元素属性或状态；另一方面，伪元素表示 DOM 外部的某种文档结构。

伪元素其实在 CSS 中一直存在，大家平时看到的有“ :first-line”、“ :first-letter”、“ :before”和“ :after”。 CSS3 中对伪元素进行了一定的调整，在以前的基础上增加一个冒号，也就相应的变成了“ ::first-letter”、“ ::first-line”、“ ::before”和“ ::after”，另外伪元素还增加了一个“::selection”。

或许大家会问，为什么要使用两个冒号？对于 IE 6 ～ 8，仅支持单冒号表示法，而现代浏览器同时支持这两种表示法。另外一个区别是，双冒号与单冒号在 CSS3 中主要用来区分伪类和伪元素。到目前来说，这两种方式都是被浏览器接受的。接下来简单介绍一下伪类元素的作用。

2.10.1　伪元素 ::first-letter

“ ::first-letter”用来选择文本块的第一个字母，除非在同一行中包含一些其他元素。“ ::first-letter”通常用于给文本元素添加排版细节，例如下沉字母或首字母，下面的代码是

如何使用“::first-letter”创建首字下沉。

```
p:first-child::first-letter {
  float: left;
  color: #903;
  padding: 4px 8px 0 3px;
  font:75px/60px  Georgia;
}
```

效果如图 2-48 所示。

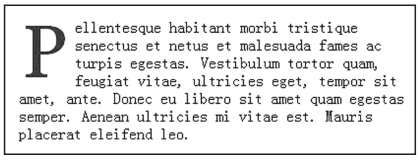


图 2-48　::first-letter 制作首字下沉效果

2.10.2　伪元素 ::first-line

“::first-line”的使用和“::first-letter”类似，也常用于文本排版方面，只不过“::first-line”用来匹配元素的第一行文本，可以应用一些特殊的样式，给文本添加一些细微的区别。“::first-line”将匹配 block、inline-block、table-caption、table-cell 等级别元素的第一行，来看一个简单的例子。

```
p:last-child::first-line {
  font: italic 16px/18px Georgia;
  color: #903;
}
```

上面的代码意味着最后一个段落的第一行文字显示为红色斜体，如图所示 2-49 所示。

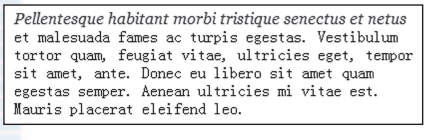


图 2-49　::frist-line 制作首行文字效果

2.10.3　伪元素 ::before 和 ::after

对于“::before”和“::after”来说，大家并不多见，但“:before”和“:after”，或许不会陌生，因为清除浮动就使用这两个伪类。

“::before”和“::after”不是指存在于标记中的内容，而是可以插入额外内容的位置。尽管生成的内容不会成为 DOM 的一部分，但它同样可以设置样式。

要为伪元素生成内容，还需要配合 content 属性。例如，假设在页面上所有外部链接之后的括号中附加它们所指向的 URL，无须将 URL 硬编码到标记中，可以结合使用一个属性选择器和“::after”伪元素。

```
a[herf^=http]::after {
  content:"(" attr(herf) ")";
}
```

如今在 CSS3 中使用“::before”和“::after”越来越多见，例如给链接添加 ICON 的效果。Font Awesome[⊖]站点制作的 ICON 就使用伪元素“::before”和“::after”，下面截取部

⊖　网址为 http://fortawesome.github.com/Font-Awesome。

分代码。

```
/*  Font Awesome styles
    ------------------------------------------------------- */
[class^="icon-"]:before,
[class*=" icon-"]:before {
  font-family: FontAwesome;
  font-weight: normal;
  font-style: normal;
  display: inline-block;
  text-decoration: inherit;
}
a [class^="icon-"],
a [class*=" icon-"] {
  display: inline-block;
  text-decoration: inherit;
}
/* makes the font 33% larger relative to the icon container */
.icon-large:before {
  vertical-align: middle;
  font-size: 1.3333333333333333em;
}
.btn [class^="icon-"],
.nav-tabs [class^="icon-"],
.btn [class*=" icon-"],
.nav-tabs [class*=" icon-"] {
  /* keeps button heights with and without icons the same */

  line-height: .9em;
}
li [class^="icon-"],
li [class*=" icon-"] {
  display: inline-block;
  width: 1.25em;
  text-align: center;
}
li .icon-large:before,
li .icon-large:before {
  /* 1.5 increased font size for icon-large * 1.25 width */

  width: 1.875em;
}
ul.icons {
  list-style-type: none;
  margin-left: 2em;
  text-indent: -0.8em;
}
ul.icons li [class^="icon-"],
ul.icons li [class*=" icon-"] {
  width: .8em;
}
```

```
ul.icons li .icon-large:before,
ul.icons li .icon-large:before {
  /* 1.5 increased font size for icon-large * 1.25 width */
  vertical-align: initial;
}
.icon-bullhorn::before {
  content: "\f0a1";
}
.icon-beaker::before {
content: "\f0c3";
}
```

效果如图 2-50 所示。

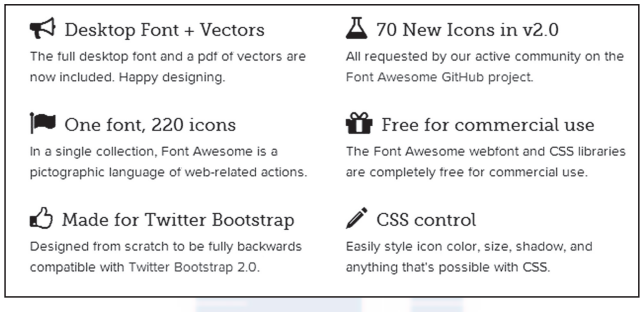


图 2-50　Font Awesome 制作 ICON

注意　图 2-50 效果还需要配合 CSS3 的 @font-face 特性，后续章节会进行详细介绍。

2.10.4　伪元素 ::selection

“::selection”是用来匹配突出显示的文本。浏览器默认情况下，选择网站文本是深蓝的背景，白色的字体，如图 2-51 所示。

图 2-51　浏览器默认突出文本的效果

有的设计需要一个与众不同的效果，此时“::selection”就非常实用。不过浏览器对“::selection”支持并不完美，在整个 IE 系列中仅有 IE 9 支持，Firefox 也需要加上其私有属性“-moz”。不过值得庆幸的是，Webkit 内核浏览器支持，其正确的使用如下。

```
/*Webkit,Opera9.5+,IE 9+*/
::selection {
  background: 颜色值;
    color: 颜色值;
```

```
}
/*Mozilla Firefox*/
::-moz-selection {
  background: 颜色值;
    color: 颜色值;
}
```

一起来看看 W3cplus[12] 上的使用。

```
::selection {
  background: red;
  color: #fff;
}
::-moz-selection {
  background: red;
  color: #fff;
}
```

此时选择文本时，背景是大红色，前景色是白色，如图 2-52 所示。

基于JQuery实现的当当品牌选择器的功能的实现功能描述：1、默认状态下品牌只显示其中的一部分，但是实事上所有的品牌的名称和莲姐

图 2-52　::selection 伪元素突出文本效果

注意　伪元素 ::selection 仅接受两个属性，一个是 background，另一个是 color。

2.11　属性选择器

在 HTML 中，通过各种各样的属性可以给元素增加很多附加的信息。例如，通过 id 属性可以将不同 DIV 元素进行区分。CSS2 中引入了一些属性选择器，这些选择器可基于元素的属性来匹配元素，而 CSS3 在 CSS2 的基础上扩展了这些属性选择器，支持基于模式匹配来定位元素。

2.11.1　属性选择器语法

CSS3 在 CSS2 的基础上新增了 3 个属性选择器，可以帮助大家对标签进行过滤，也能非常容易帮助大家在众多标签中定位自己需要的 HTML 标签，下面通过表 2-18 详细介绍 CSS3 的属性选择器的使用。

表 2-18　CSS3 属性选择器列表

选择器	功能描述
E[attr]	选择匹配具有属性 attr 的 E 元素。其中 E 可以省略，表示选择定义了 attr 属性的任意类型元素

（续）

选择器	功能描述
E[attr=val]	选择匹配具有属性 attr 的 E 元素，并且 attr 的属性值为 val（其中 val 区分大小写），同样 E 元素省略时表示选择定义了 attr 属性值为 val 的任意类型元素
E[attr|=val]	选择匹配 E 元素，且 E 元素定义了属性 attr，attr 属性值是一个具有 val 或者以 val- 开始的属性值。常用于 lang 属性（例如 lang=”en-us”）。例如 p[lang=en] 将匹配定义为英语的任何段落，无论是英式英语还是美式英语
E[attr~=val]	选择匹配 E 元素，且 E 元素定义了属性 attr，attr 属性值具有多个空格分隔的值，其中一个值等于 val。例如，.info[title~=more] 将匹配元素具有类名 info，而且这个元素设置了一个属性 title，同时 title 属性值以包含了“ more”的任何元素，例如 click me
E[attr*=val]	选择匹配元素 E，且 E 元素定义了属性 attr，其属性值任意位置包含了“ val”。换句话说，字符串 val 与属性值中的任意位置相匹配
E[attr^=val]	选择匹配元素 E，且 E 元素定义了属性 attr，其属性值以 val 开头的任何字符串
E[attr$=val]	选择匹配元素 E，且 E 元素定义了属性 attr，其属性值以 val 结尾的任何字符串。刚好与 E[attr^=val] 相反

CSS3 遵循了惯用的编码规则，通配符的使用提高了样式表的书写效率，也使 CSS3 的属性选择器更符合编码习惯。CSS3 中常见的通配符如表 2-19 所示。

表 2-19　CSS3 中常见的通配符

通配符	功能描述	示例
^	匹配起始符	span[class^=span] 表示选择以类名以“span”开头的所有 span 元素
$	匹配终止符	a[href$=pdf] 表示选择以”pdf”结尾的 href 属性的所有 a 元素
*	匹配任意字符	a[title*=site] 匹配 a 元素，而且 a 元素的 title 属性值中任意位置有“site”字符的任何字符串

2.11.2　浏览器兼容性

属性选择器在浏览器兼容性方面表现还是可以的，如表 2-20 所示。

表 2-20　属性选择器的浏览器兼容性

选择器					
E[attr]	7 + √	√	√	√	√
E[attr=val]	7 + √	√	√	√	√
E[attr|=val]	7 + √	√	√	√	√
E[attr~=val]	7 + √	√	√	√	√
E[attr*=val]	7 + √	√	√	√	√
E[attr^=val]	7 + √	√	√	√	√
E[attr$=val]	7 + √	√	√	√	√

2.11.3　属性选择器的使用方法详解

前面了解了 CSS3 属性选择器的使用规则以及浏览器兼容性，接下来通过一些简单的示例来体验一下 CSS3 属性选择器的强大功能。

先创建一个简单的 HTML 结构，并付上一些默认样式。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 属性选择器的使用 </title>
  <style type="text/css">
    .demo {
      width: 300px;
  border: 1px solid #ccc;
  padding: 10px;
      overflow: hidden;
      margin: 20px auto;
  }
  .demo a {
  float: left;
  display: block;
  height: 50px;
  width: 50px;
  border-radius: 10px;
  text-align: center;
  background: #aac;
  color: blue;
     font: bold 20px/50px Arial;
  margin-right: 5px;
  text-decoration: none;
     margin: 5px;
  }
  </style>
</head>
<body>
  <div class="demo">
  <a href="http://www.w3cplus.com" target="_blank"
           class="links item first" id="first" title="w3cplus">1</a>
  <a href="" class="links active item" title="test
           website" target="_blank" lang="zh">2</a>
  <a href="sites/file/test.html" class="links item"
           title="this is a link" lang="zh-cn">3</a>
  <a href="sites/file/test.png" class="links item"
           target="_balnk" lang="zh-tw">4</a>
  <a href="sites/file/image.jpg" class="links item"
           title="zh-cn">5</a>
  <a href="mailto:w3cplus@hotmail" class="links item"
           title="website link" lang="zh">6</a>
```

```
  <a href="/a.pdf" class="links item"
          title="open the website" lang="cn">7</a>
  <a href="/abc.pdf" class="links item"
          title="close the website" lang="en-zh">8</a>
  <a href="abcdef.doc" class="links item"
          title="http://www.sina.com">9</a>
  <a href="abd.doc" class="linksitem last" id="last">10</a>
  </div>
</body>
</html>
```

基本样式的效果如图 2-53 所示。

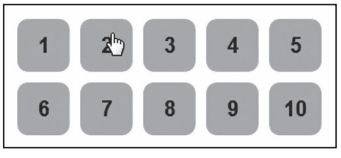


图 2-53　CSS3 属性选择器使用默认效果

接下来使用 CSS3 属性选择器进行个性化修改。

1. E[attr] 属性选择器

这个属性选择器是最简单的一种。用来选择有某个属性的元素，而不论这个属性值是什么。例如，选择所有带有 ID 属性的 a 元素，并且改变了匹配元素的背景色，如下所示。

```
a[id]{background-color:yellow;}
```

对照上面的 HTML 结构不难发现，只有第一个和最后一个链接元素 a 使用了 ID 属性，所以上面的代码选择了两个 a 元素，效果如图 2-54 所示。

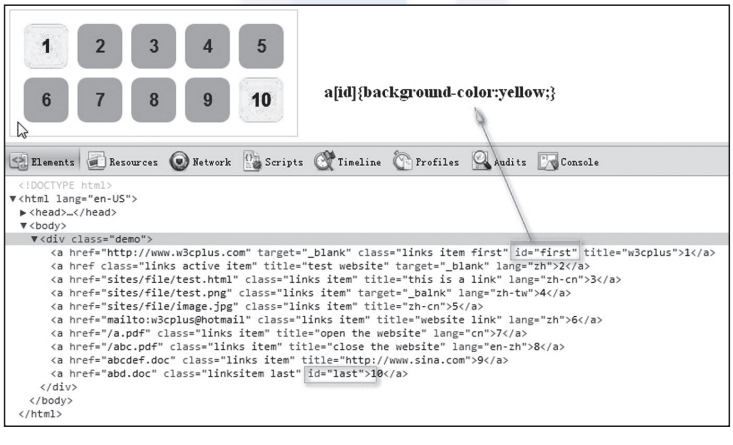


图 2-54　E[attr] 属性选择器的效果

也可以使用多属性进行选择元素。例如：

```
a[id][title]{background-color: red;}
```

上面代码表示元素 a 只要同时定义了 id 和 title 属性，都将匹配，其背景将变成红色，效果如图 2-55 所示。

2. E[attr=val] 属性选择器

E[attr=val] 是指元素 E 设置了属性 attr，并且其属性值为“ val”，而 E[attr] 属性选择器

只选择了属性为 attr 的元素 E，并没有明确的设置 attr 的属性值，这也是这两种选择器的最大区别之处。在众多元素之中缩小选择范围，能进一步精确选择自己需要的元素。在前面的实例基础上进行一下简单的修改。

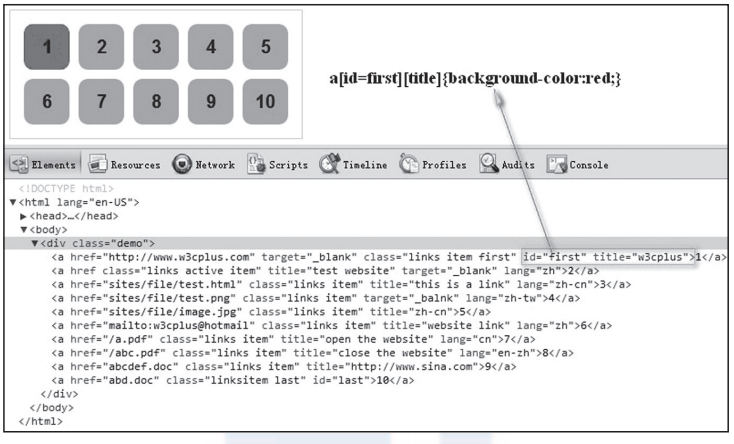


图 2-55　E[attr1][attr2] 多属性选择器使用效果

```
a[id=first]{background-color:red;}
```

此处在 id 属性的基础上指定了相应的 val 值为 first，选中所有 a 元素中设置“ id=first”的链接，并将其背景色设置为 red，效果如图 2-56 所示。

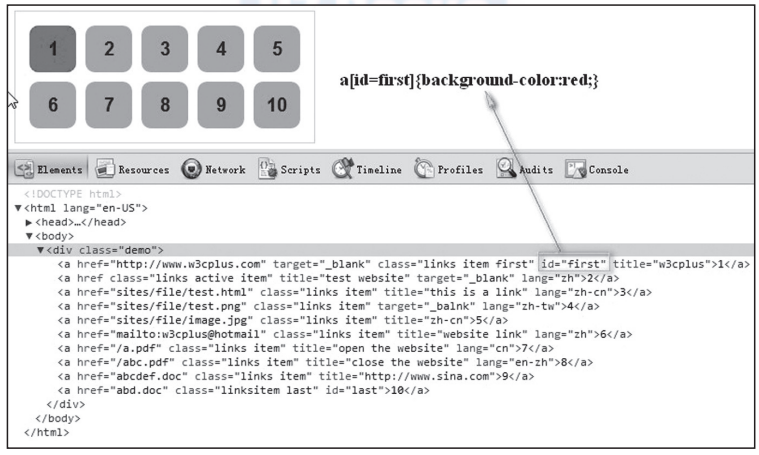


图 2-56　E[attr=val] 属性选择器的使用效果

注意　E[attr=val] 选择器中，属性和属性值必须完全匹配，特别对于属性值是词列表的形式，例如，

```
<a href="#" class="links  item"></a>
```

其中 a[class="links"]{...} 是找不到匹配元素，只有 a[class="links item"]{...} 才匹配。

3. E[attr|=val] 属性选择器

首先提醒大家，attr 后面的是一条竖线，而不是字母。E[attr|=val] 为特定属性选择器，选择 attr 属性值等于“ val”，或以“ val-”开头的所有字符串属性的元素，一起来看看这个属性选择器的使用。

```
a[lang|=zh]{background-color: yellow;}
```

上面的代码会选中所有设置了 lang 属性，并且其属性值是以“zh”或“zh-”开头的所有 a 链接元素，其效果如图 2-57 所示。

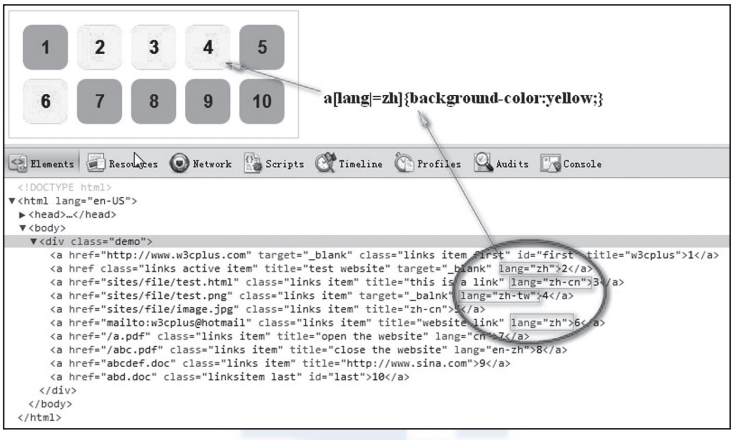


图 2-57　E[attr|=val] 选择器的使用效果

例如 Web 页面中有很多图片，图片文件名都以“ figure-1”、“ figure-2”的方式来命名，使用 E[attr|=val] 选择器来选择这些图片就很方便了。

4. E[attr~=val] 属性选择器

如果想根据元素属性值中的词列表的某个词来匹配需要的元素，就可以使用这个属性选择器。这种属性选择器匹配的元素某个属性具有一个或多个属性值，多个属性值之间使用空格隔开，当元素属性值中有一个属性值与选择器的 val 相匹配，就可以选中该元素。而前面介绍的 E[attr=val] 属性选择器是属性值要完全匹配才会被选中，它们两者区别就是“~”符号。一起来看一个简单的示例。

```
a[title~=website]{background-color:yellow;}
```

效果如图 2-58 所示。

图 2-58 很清楚地告诉我们，只要 a 链接元素设置有 title 属性，并且这个 title 属性的值包含了 website 属性值，都将匹配。如果不小心将“~”省略，结果将找不到匹配的元素。

5. E[attr*=val] 属性选择器

这个属性选择器使用了通配符，将匹配元素设置了 attr 属性，而且 attr 属性值中包含

“val”字符串。也就是说，只要所选择的属性中有 val 字符串，都将被匹配。例如：

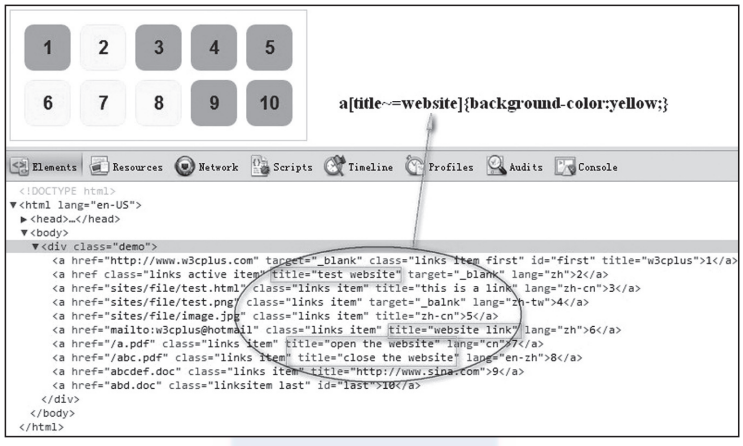


图 2-58　E[attr~=val] 选择器的使用效果

```
a[class*=links]{background-color:yellow;}
```

这个示例将选中所有的 a 元素，因为示例中的 a 元素“ class”的属性值都含有字符串“links”，不管是“links”还是“linksitem”，其效果如图 2-59 所示。

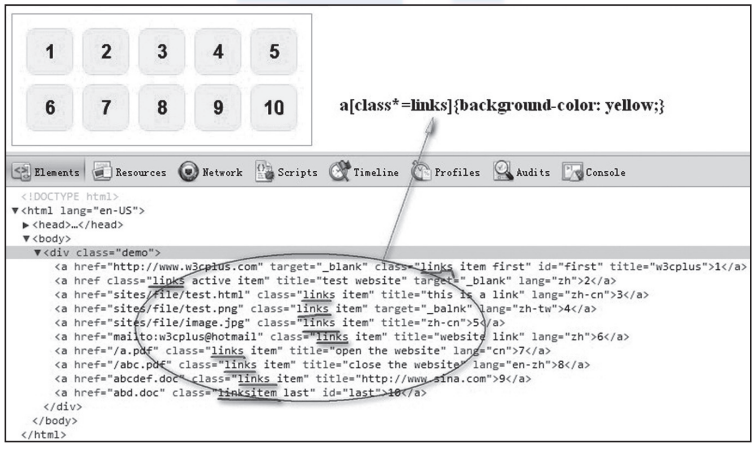


图 2-59　E[attr*=val] 属性选择器的使用效果

E[attr*=val] 与 E[attr~=val] 很容易混淆，它们的区别是：E[attr*=val] 匹配的是元素属性值中只要包含“ val”字符串就可以，而 E[attr~=val] 匹配的是元素属性值中要包含“ val”，并不仅是字符串。例如，a[title~=links] 属性值中的 links 是一个单词，而 a[title*=links] 中的 links 不一定是一个单词，就如上面的示例中，可以是“ linksitem”。换句话来说，“”元素只有 a[title*=links] 匹配，但是“<a title='links

item'>”元素，a[title~=links] 和 a[title*=links] 都匹配。

6. E[attr^=val] 属性选择器

指选择 attr 属性值以“ val”开头的所有元素。换句话说，选择的属性，其对应的属性值是以“val”开始的，一起来看一个简单的示例。

```
a[href^=http]{background-color:yellow;}
```

这个选择器表示选择 href 属性，以“ http”开头的所有 a 元素，即只要 a 元素中的 href 属性值是以“http”开头的元素都会选中，如图 2-60 所示。

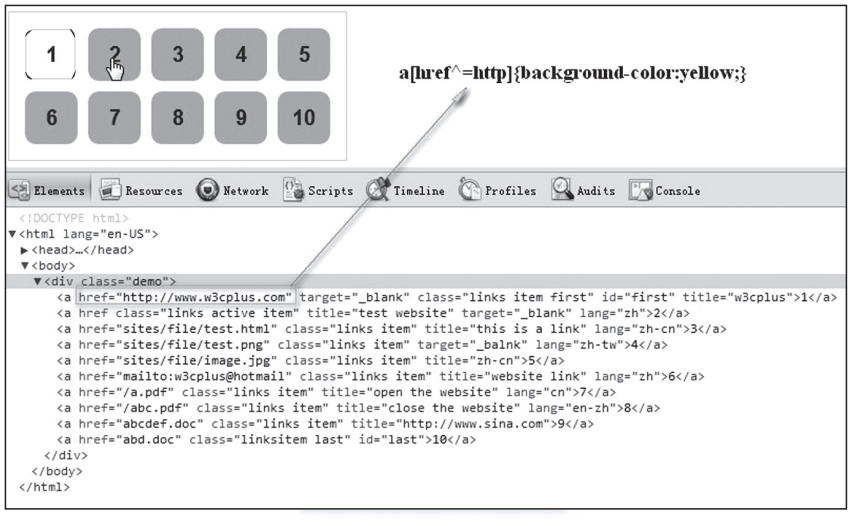


图 2-60　E[attr^=val] 属性选择器的使用效果

前面有一个属性选择器 E[attr|=val] 匹配的是以“ val”或者“ val-”开头的元素，而 E[attr^=val] 匹配的是以“ val”开头的元素，这两个选择器有时使用的效果非常相似，仅仅区别在于 E[attr|val] 还匹配以“val-”开头的元素，其中“-”是不可或缺的。

7. E[attr$=val] 属性选择器

这个属性选择器刚好与 E[attr^=val] 相反，表示选择 attr 属性值以“ val”结尾的所有元素。运用在一些特殊的链接加背景图很方便，例如给 PDF、PNG、DOC 等不同文件加上不同的 ICON 图标就可以使用这个属性。下面通过给“png”值结尾的 a 元素改变背景色。

```
a[href$=png]{background-color:yellow;}
```

运用上面代码的效果如图 2-61 所示。

通过一些简单的示例，向大家详细介绍了 CSS3 属性选择器的使用方法，以及效果，接下来通过实战来学习 CSS3 属性选择器在 Web 中是如何发挥作用。

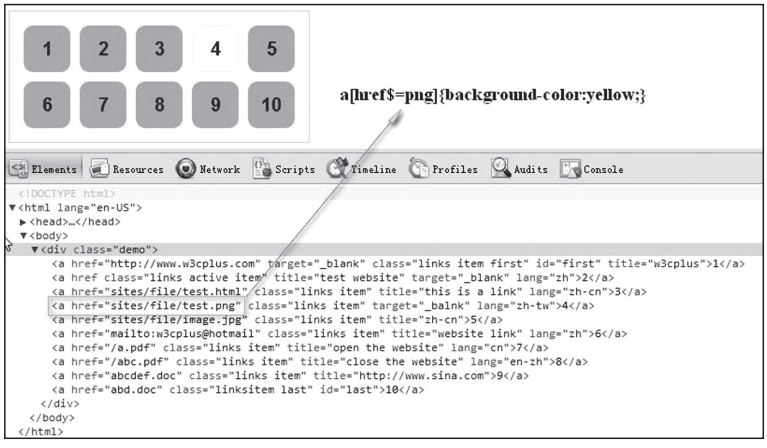


图 2-61　E[attr$=val] 属性选择器的使用效果

2.11.4　实战体验：创建个性化链接样式

对于 Web 设计人员来说碰到的链接有很多种，例如文字、图片等。

例如新浪爱问共享资料库[⊖]，下载文档列表的链接前面会显示一个文档类型图标（如图 2-62 所示），这样的设计给用户带来一种超好的体验。另外，有些 Web 网站对于本地链接和外部链接也会使用不同的图标来区分（如图 2-63 所示）。CSS3 的属性选择器此时就显得特别的有用，通过它们可以很容易实现这些效果。

图 2-62　显示类型图标的文档下载链接

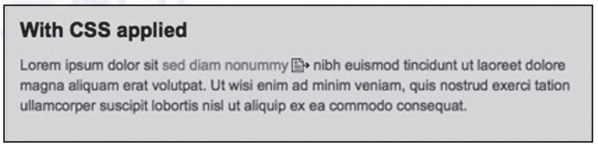


图 2-63　Web 页面的外部链接

本节示例的效果如图 2-64 所示，当然此效果只是部分 CSS3 属性选择器的运用。

图 2-62 和图 2-63 的链接效果，使用属性选择器匹配 a 元素中的 href 属性值。如果 a 元素的 href 属性值是以“http://”开头，将这些链接划分到外部链接一类中，给其定义一个样式，对于下载文档的类型可以根据其文件的扩展名来设置不同的图标。例如，链接为 Word 文件时，后面显示 Word 文档图标；链接为 PDF 文件时，后面显示 PDF 文档图标。

根据这一思路，可以编写一个简化的示例代码，如下所示。

⊖ 链接是 http://ishare.iask.sina.com.cn/。

图 2-64　创建个性化链接

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title> 创建个性化链接样式 </title>
  <style type="text/css">
    a {
      font-size: 16px;
      text-decoration: none;
      padding-right: 20px;
      display: inline-block;
      margin-bottom: 5px;
      background: url(a.png) no-repeat 100% -327px;
    }
    /* 匹配所有外部链接 */
    a[href^="http://"]{
      background-position: 100% -409px;
    }
    /* 匹配首页 */
    a[href="home"]{
      background-position: 100% -382px;
    }
    /* 匹配 .doc 文档 */
    a[href$="doc"]{
      background-position: 100% 0;
    }
    /* 匹配 .ppt 文档 */
    a[href$="ppt"]{
      background-position:100% -50px;
    }
    /* 匹配 .xls 文档 */
    a[href$="xls"]{
```

```
      background-position:100% -75px;
    }
    /* 匹配 .html 文档 */
    a[href$="html"]{
      background-position:100% -100px;
    }
    /* 匹配 .pdf 文档 */
    a[href$="pdf"]{
      background-position:100% -125px;
    }
    /* 匹配 .jpg,.gif,.png 图片 */
    a[href$="jpg"],
    a[href$="gif"],
    a[href$="png"]{
      background-position:100% -172px;
    }
    /* 匹配 .swf 文档 */
    a[href$="swf"]{
      background-position:100% -193px;
    }
    /* 匹配 .zip,.rar 压缩文档 */
    a[href$="zip"],
    a[href$="rar"]{
      background-position:100% -217px;
    }
    /* 匹配 .mp3 文档 */
    a[href$="mp3"]{
      background-position:100% -245px;
    }
    /* 匹配 .exe 安装文件 */
    a[href$="exe"]{
      background-position:100% -273px;
    }
    /* 匹配 .txt 文本文档 */
    a[href$="txt"]{
      background-position:100% -298px;
    }
    /* 匹配邮箱地址 */
    a[href^="mailto"]{
      background-position:100% -350px;
    }
  </style>
</head>
<body>
  <ul>
    <li><a href="http://www.w3cplus.com">w3cplus.com</a></li>
    <li><a href="home">home</a></li>
    <li><a href="a.doc">Word 文档 </a></li>
    <li><a href="b.ppt">Powerpoint 文档 </a></li>
    <li><a href="c.xls">Excel 文档 </a></li>
    <li><a href="d.html">HTML 文档 </a></li>
```

```
    <li><a href="e.pdf">PDF 文档 </a></li>
    <li><a href="f.jpg">JPG 图片文档 </a></li>
    <li><a href="f.gif">GIF 图片文档 </a></li>
    <li><a href="f.png">PNG 图片文档 </a></li>
    <li><a href="g.swf">Flash 文档 </a></li>
    <li><a href="h.zip">ZIP 压缩文档 </a></li>
    <li><a href="h.rar">RAR 压缩文档 </a></li>
    <li><a href="j.mp3">MP3 文档 </a></li>
    <li><a href="h.exe">EXE 安装文件 </a></li>
    <li><a href="a.txt">TXT 文本文档 </a></li>
    <li><a href="mailto:w3cplus@hotmial.com">w3cplus@hotmail.com</a></li>
    <li><a href="#"> 本地链接 </a></li>
  </ul>
</body>
</html>
```

由于 IE 6 浏览器不被支持 CSS3 属性选择器，为了实现图 2-64 效果，早期的 Web 设计师们会借助于 JavaScript 脚本，或者通过古老的办法给不同文档类型元素添加不同的类名，例如给外部链接添加类名“external”，给 Word 文档添加类名“word”。

随着 IE 6 逐渐接近死亡，可以大胆使用 CSS3 属性选择器来制作，或者你是一个完美的追求者，可以考虑优雅降级方案来处理。

2.12 本章小结

本章主要向大家介绍了 CSS3 核心部分中的选择器。首先介绍 CSS3 选择器的优势，然后分别详细介绍了基本选择器、层次选择器、伪类选择器、属性选择器。希望大家学习之后对 CSS3 选择器有一个更详细的了解，为后面的学习打下坚实的基础。

第 3 章 Chapter 3

CSS3 边框

提到边框，大家首先想到的是 CSS 的 border 属性。不错，border 属性是 CSS 中盒模型基础属性之一。在 CSS3 中，关于边框的运用会有什么样的不同之处呢？又将如何使用呢？本章我们带着这些问题开始我们的 CSS3 边框之旅。

3.1 CSS3 边框简介

border 属性在 CSS1 中就已经定义了，使用它可以设置元素的边框风格，例如设置不同的边框颜色以及粗细。在详细介绍 CSS3 边框运用之前，先简单回顾边框属性。

3.1.1 边框的基本属性

CSS1 和 CSS2 中的边框属性其实很简单，其主要包括三个类型值。

- border-width：设置元素边框的粗细。
- border-color：设置元素边框的颜色。
- border-style：设置元素边框的类型。

在实际中可以将上面三个属性合并在一起，其缩写的语法：

```
border: border-width  border-style  border-color;
```

缩写后的每个属性之间使用空格隔开，而且它们之间没有先后顺序，可这里三个值中唯一需要的值是“border-style”，因此，要是这样写边框样式将会没有任何效果。

```
.elm {border:3px red}
```

此时浏览器将“ border-style”解析成为“ none”。要是这样设置，这个时候元素的边框是实线，粗线将是其默认值。

```
.elm{border:solid}
```

边框 border-width 的默认值是“ medium”（大约等于 3 ～ 4px）；border-color 的默认色就是字体的颜色。

在 Web 实际制作之中，时常只为了方便使用，CSS 中的 border 可以给不同的边设置不同的风格，其也遵守“TRBL”原则（Top/Right/Bottom/Left），例如单独写边框类型。

```
border-top-style:/* 设置元素顶部边框类型 */
border-right-style:/* 设置元素右边边框类型 */
border-bottom-style:/* 设置元素底部边框类型 */
border-left-style:/* 设置元素左边边框类型 */
```

上面是边框类型的扩展写法，同样的道理，border-color 和 border-width 也可以像上面一样使用。除了上面的写法之外，还有一种简写形式。

```
border-style: solid;
/* 一个值时，表示四条边都 solid 类型 */
border-style: solid dotted;
/* 两个值时，第一个值表示元素上下边框类型，第二值表示左右边框类型 */
border-style:solid dotted dashed;
/* 三个值时，第一个值表示元素顶边的类型，第二个值表示左右边框类型，第三个值表示底部边框类型 */
border-style: solid dotted dashed inset;
/* 四个值时，第一个值表示元素顶边的类型，第二个值表示元素右边框类型，第三个值表示元素底边的类型，第四个值表示元素左边框类型 */
```

同样的原理，border-color 和 border-width 具有一样的使用方法。如果只需要设置元素某部分具有边框效果，我们可以合成起来。

```
li {
  border: 1px solid red;
  border-width: 1px 0;
}
```

仅两行代码就表达出元素 li 顶部和底部都有一条 1px 的红色实线。这样方便维护代码，并且提升 CSS 性能。

3.1.2 边框的类型

CSS 中使用 border-style 为元素 border 定义边框类型，常见的有实线“ solid”、虚线“ dashed”、点状线“ dotted”等。下面一起看看 CSS 中 border-style 的几种类型效果，如表 3-1 所示。

表 3-1　border-style 值列表

属性值	功能描述	示例代码	效果
none	定义无边框	.elm {border:none;}	none
hidden	与“none”相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突	.elm{border:hidden;}	hidden
dotted	定义点状边框	.elm{border:3px dotted red ;}	dotted
dashed	定义虚线边框	.elm{border:3px dashed red;}	dashed
solid	定义实线边框	.elm{border:3px solid red;}	solid
double	定义双线。双线的宽度等于 border-width 的值	.elm{border:3px double red;}	double
groove	定义 3D 凹槽边框，其效果取决于 border-color 的值	.elm{border:3px groove red;}	groove
ridge	定义 3D 垄状边框，其效果取决于 border-color 的值	.elm{border:3px ridge red;}	ridge
inset	定义 3D inset 边框，其效果取决于 border-color 的值	.elm{border:8px inset red;}	inset
outset	定义 3D outset 边框，其效果取决于 border-color 的值	.elm{border:8px outset red;}	outset
inherit	规定应该从父元素继承边框样式。部分浏览器不支持这个属性值		

上面 11 个值在各浏览器下呈现的效果均有差异，其中最不可预测的边框样式是 double。它定义为两条线的宽度加上这两条线之间的空间等于 border-width 值。而 dotted、dashed、outset 和 inset 在不同的浏览器下也无法保证一致，如图 3-1 所示。

图 3-1　border-style 各浏览器渲染效果

注意 图 3-1 中 IE 7 和 IE 8 未使用原生 IE 测试，而是使用了 IE 9 自带的 IE 7、IE 8 进行的测试。

3.1.3 谁在使用 CSS3 边框

CSS3 增强的边框样式具有强大的生命力，灵活使用这些属性可以设计很多优美精巧的 UI 界面效果。这些属性谁在使用呢?

- border-color 受制于浏览器兼容性，至今在项目中使用该属性的项目几乎不存在。
- border-image 浏览器的支持度强一些，但运用在项目中仅存在一些前端爱好者的 blog 中。
- border-radius 得到浏览器的强大支持，在互联网上随处可见。
- box-shadow，目前在 Web 页面上 CSS3 的盒子阴影特性应用非常广泛。

3.2 CSS3 边框颜色属性

border-color 属性早在 CSS1 中就已经定义了。不过，CSS3 增强了这个属性的功能，使用它可以为元素边框设置更多的颜色，从而方便 Web 设计人员设计出更为绚丽的边框效果，例如渐变的边框效果、多颜色的边框效果等。

3.2.1 border-color 属性的语法及参数

border-color 的语法看上去和 CSS1 中的完全相同，但为了避免与 border-color 属性的原功能（也就是在 CSS1 中的定义边框颜色功能）发生冲突，CSS3 在这个属性上做出一定的修改。语法如下。

```
border-color: [ <color> | transparent ]{1,4} | inherit
```

换句话说，如果使用 border-color 这种缩写语法，将不会有任何效果，必须将这个 border-color 标准写法拆分成四个边框，使用多颜色才会有效果。

- border-top-colors: [<color> | transparent]{1,4} | inherit;
- border-right-colors: [<color> | transparent]{1,4} | inherit;
- border-bottom-colors: [<color> | transparent]{1,4} | inherit；
- border-left-colors: [<color> | transparent]{1,4} | inherit;

注意 以上四个属性中 color 是复数 colors，如果在书写过程中要是少了 (s) 字母是错误的，没有任何反应。

由于 CSS3 的 border-color 属性还没成为标准规范，为了让不同浏览器能渲染正常，有

必要加上前缀，如表 3-2 所示。

表 3-2　不同浏览器前缀

浏览器分类	浏览器	私有属性的前端缀
Gecko 引擎内核的浏览器	Mozilla（常指 Firefox 浏览器）	-moz-
Presto 引擎内核的浏览器	Opera	-o-
KHTML 引擎内核的浏览器	Konqueror	-khtml-
Trident 引擎内核的浏览器	Internet Explorer	-ms-

有些情况中，为了使用 CSS3 属性，可能必须添加 4 行代码或者更多行。例如，为了让 border-color 在 Firefox 浏览器下正常，需要加上前缀“-moz-”。

```
-moz-border-top-colors: #111 #222 #333 #444 #555;
-moz-border-right-colors: #111 #222 #333 #444 #555;
-moz-border-bottom-colors: #111 #222 #333 #444 #555;
-moz-border-left-colors: #111 #222 #333 #444 #555;
```

现在的规范还不是最终版本，在执行中还会有一些漏洞。因此，执行这些功能时，使用供应商前缀来提供数值，并且使用无前缀声明来提供每个属性的永久版本。当规范成为最终版本且经过完善后，浏览器前缀将被取消。例如：

```
-moz-border-top-colors:          #111 #222 #333 #444 #555;
-moz-border-right-colors:        #111 #222 #333 #444 #555;
-moz-border-bottom-colors:       #111 #222 #333 #444 #555;
-moz-border-left-colors:         #111 #222 #333 #444 #555;

-webkit-border-top-colors:       #111 #222 #333 #444 #555;
-webkit-border-right-colors:     #111 #222 #333 #444 #555;
-webkit-border-bottom-colors:    #111 #222 #333 #444 #555;
-webkit-border-left-colors:      #111 #222 #333 #444 #555;

-ms-border-top-colors:           #111 #222 #333 #444 #555;
-ms-border-right-colors:         #111 #222 #333 #444 #555;
-ms-border-bottom-colors:        #111 #222 #333 #444 #555;
-ms-border-left-colors:          #111 #222 #333 #444 #555;

-o-border-top-colors:            #111 #222 #333 #444 #555;
-o-border-right-colors:          #111 #222 #333 #444 #555;
-o-border-bottom-colors:         #111 #222 #333 #444 #555;
-o-border-left-colors:           #111 #222 #333 #444 #555;

border-top-colors:               #111 #222 #333 #444 #555;
border-right-colors:             #111 #222 #333 #444 #555;
border-bottom-colors:            #111 #222 #333 #444 #555;
border-left-colors:              #111 #222 #333 #444 #555;
```

即使用这些前缀来维护代码似乎需要很多工作，现在使用 CSS3，仍然是利大于弊。虽然需要改变一些前缀属性来修改设计元素，相对于通过图形软件更改背景图像或处理那些

其他标记和 hack 脚本，维护基于 CSS3 的设计要容易一些。

Lea Verou 制作了一个插件 -prefix-free[⊖]，使用这个插件，大家在平时的开发中就可以略去浏览器的前缀，从而节约大家的开发时间与维护成本。

注意 如无特别注明，后面涉及的 CSS3 属性，都需加上各浏览器前缀。

border-color 属性的参数其实很简单，就是颜色值 < color>，可以为任意合法的颜色值或者颜色值列表。当 border-color 只设置一个颜色值时，效果和 CSS1 中的 border-color 效果一样。只有设置了 n 个颜色，并且边框宽度也为 n 像素，就可以使用 CSS3 的 border-color 属性设置 n 个颜色，每种颜色显示 1 像素的宽度，如果宽度值大于颜色数量的值，最后一种颜色用于显示剩下的宽度。例如，元素的边框设置为 20px，而元素的边框颜色只设置了 10 个，剩下的 10px 宽度都将显示最后一种颜色，如图 3-2 所示。

图 3-2 border-color 颜色值设置效果

3.2.2 浏览器兼容性

CSS3 的“ border-color”属性浏览器兼容性虽然功能强大，但到写这本书的时候为止，仅有 Firefox 3.0 以及其以上的版本支持，而且还不是标准写法（如表 3-3 所示）。

表 3-3 border-color 浏览器兼容性

属性名					
border-color	×	3.0 + √	×	×	×

CSS3 的 border-color 能帮 Web 设计师制作渐变、内阴影、外阴影等绚丽的边框效果，但是目前为止，仅有 Firefox 3.0 以及其以上版本的浏览器支持，而且还不是标准语法，仅是 Firefox 浏览器自己一个扩展性写法。因此这个属性的使用性并不是很强，大家在实际使用中需要注意。

3.2.3 border-color 属性的优势

在 CSS2 中实现多颜色的边框效果，无外乎两种方法，其一通过添加额外的标签，在每

⊖ 详见 http://leaverou.github.com/prefixfree/。

个标签上设置不同的颜色，其二就是通过背景图片来制作。这两种方法和 CSS3 的 border-color 相比都略显弊端，第一种多了标签，使结构冗余，第二种方法背景图片不好维护，特别是在改变边框颜色之时更是麻烦。

有一篇博文[⊖]介绍了使用一个 HTML 标签元素，通过“::before”和“::after”伪元素使用背景色和边框颜色来模拟一个六色边框的效果，这种方法对多颜色受到限制，最多只能制作六种颜色。

注意　除了这些方法，CSS3 中还有 box-shadow 也能实现，详细参考后面介绍 box-shadow 的章节。

3.2.4　实战体验：立体渐变边框效果

在这个案例中，使用 CSS3 的 border-color 属性制作一个渐变立体效果的边框，如图 3-3 所示。

制作这个效果的设计思路很简单，使用 border-color 属性，将颜色从浅到深依次叠加起来，营造出一种立体渐变的边框效果。

图 3-3　border-color 制作立体渐变边框（Firefox）

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>border-color 制作立体渐变边框效果 </title>
  <style type="text/css">
    div{
      width: 200px;
      height: 100px;
      border: 10px solid transparent;
      border-radius: 15px 0 15px 0;
      -moz-border-top-colors:#a0a #909 #808 #707 #606 #505 #404 #303;
      -moz-border-right-colors:#a0a #909 #808 #707 #606 #505 #404 #303;
      -moz-border-bottom-colors:#a0a #909 #808 #707 #606 #505 #404 #303;
      -moz-border-left-colors:#a0a #909 #808 #707 #606 #505 #404 #303;
    }
  </style>
</head>
<body>
  <div></div>
</body>
</html>
```

3.3　CSS3 图片边框属性

border-image 效果在 CSS2 中，只有使用背景图片来制作，而且制作过程非常复杂，做

⊖ 地址为 http://nicolasgallagher.com/multiple-backgrounds-and-borders-with-css2/。

完后也很难维护。如今 CSS3 中增添了一个图片边框的属性，能够模拟出 background-image 属性的功能，功能比 background-image 强大，我们可以使用 border-image 属性给任何元素（除 border-collapse 属性值为 collapse 的 table 元素之外）设置图片效果边框，还可以使用这个来制作圆角按钮效果、渐变的 Tabs 效果等。

3.3.1 border-image 属性的语法及参数

为了能更好地学习和理解 border-image 这个属性，还是从其最基本的语法入手。

```
border-image: none | <image> [<number> | <percentage>] {1,4} [/ <broder-width> {1,4}] ?[stretch | repeat | round] {0,2}
```

接下来就给大家说说这些参数的含义与使用方法。

- none：默认值，表示边框无背景图片。
- <image>：设置背景图片，这跟 background-image 一样，可以使用绝对或相对的 URL 地址，来指定边框的背景图片。
- <number>：number 是一个数值，用来设置边框或者边框背景图片的大小，其单位是像素（px），可以使用 1 ～ 4 个值，表示 4 个方位的值，大家可以参考 border-width 设置方式。
- <percentage>：percentage 也是用来设置边框或者边框背景图片的大小，跟 number 不同之处是，percentage 使用的是百分比。
- stretch、repeat、round：这三个属性参数是用来设置边框背景图片的铺放方式，类似于 background-position，其中 stretch 会拉伸边框背景图片、repeat 是会重复边框背景图片、round 是平铺边框背景图片，其中 stretch 为默认值。

border-image 和 background-image 之间有一些类似之处，包括图片的引用和排列方式等。

3.3.2 border-image 属性使用方法

为了更好地理解，暂时把 border-image 在语法的表达形式进行属性的分解阐述（实际应用中是不能分解的，此处只是用来帮助大家更好地理解 border-image 属性）。

- 引入背景图片：border-image-source。
- 切割引入背景图片：border-image-slice。
- 边框图片的宽度：border-image-width。
- 边框背景图片的排列方式：border-image-repeat。

接下来重点学习 border-image 拆分出来的四个属性。

1. border-image-source

语法：

```
border-image-source: url(image url);
/*image url 可是以边框图片的相对地址，也可以是绝对地址 */
```

border-image-source 跟 CSS2 中的 background-image 属性相似，也是通过 url() 来调用背景图片，图片的路径可以是相对地址，也可以是绝对地址，当然不想使用背景图片也可以设置为“border-image:none”；其默认值就是 none。

2. border-image-slice

语法：

```
border-image-slice:[<number> | <percentage>] {1,4} && fill ?
```

border-image-slice 是用来分解引入进来的背景图片，这个参数相对来说比较复杂和特别，主要表现在以下几点。

1）取值支持 <number> | <percentage>。其中 number 是没有单位的，因为其默认的单位就是像素。除了直接用 number 来设置外，还可以使用百分比值来表示，即相对于边框背景图片而言的，例如边框图片的大小是 300px × 240px，取百分比为 25%，30%，15%，20%，它们实际对应的效果就是剪切了图片的 60px，90px，36px，60px 的四边大小，如图 3-4 所示。

border-image-slice 中的 number 或者 percentage 都可以取 1 ～ 4 个值，这个类似于 CSS2 中的 border-width 的取值方式，也遵从 top、right、bottom、left 的规则，如果对这个不太清楚可以参考 CSS2 中的 border-width 或者 padding、margin 等属性的使用方法。

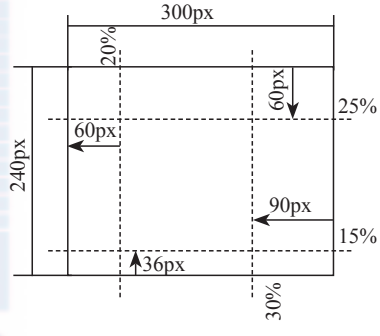


图 3-4　border-image-slice 百分比取值示意图

Fill 从字面上说就是填充的意思，如果使用这个关键字时，图片边界的中间部分将保留下来。默认情况下为空。

2）剪切的特性（slice）。在 border-image 中 slice 是一个关键部分，也是让人难以理解的部分。如果理解 CSS3 的 clip 属性，再理解 border-image-slice 相对会轻松一些。border-image-slice 虽然表面上说不是剪切，但在实际应用中它就是一种纯粹的剪切，它把通过 border-image-source 取到的边框背景图片切成九份，再像 background-image 一样重新布置。

来看一个示例[⊖]，其剪切的效果如图 3-5 所示。对应的代码如下所示。

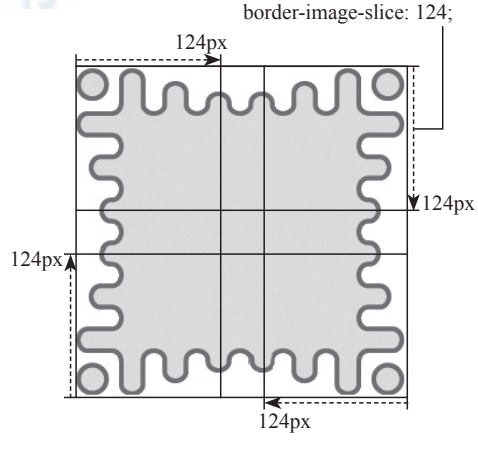


图 3-5　border-image-slice 以 124px 位置切图

```
div {
  border: 12px double green;
```

⊖ 选自 W3C 官网（http://www.w3.org/TR/css3-background/#border-images）。

```
  -moz-border-image: url(../image/border.png) 124;
  -webkit-border-image: url(../image/border.png) 124;
  -o-border-image: url(../image/border.png) 124;
  border-image: url(../image/border.png) 124;
}
```

上面的示意中，它在距边框背景图的 top、right、bottom、left 四边的 124px 分别切了一刀，这样一来就把背景图切成了九个部分，称为“九宫格”。“九宫图”在本文专指由九个方格形成的矩形布局图，正如图 3-5 所示。这样就应用这个“九宫格”来帮助我们了解 border-image 的绘制原理。图 3-6 是来自 W3C 官网的 border-image 背景图，也是一张重要的示意图，因为这张图刚好具有我们所说的“九宫格”（27x3）x(27x3)。

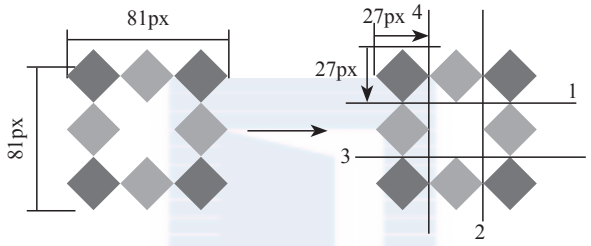


图 3-6　九宫格示意图

图 3-6 中，1、2、3、4 四条蓝色切割线分别在距边框背景图片的 27px 位置切了四刀，将 border-image 背景图片分成九部分。八个边块 border-top-image、border-right-image、border-bottom-image、border-left-image、border-top-right-image、border-bottom-right-image、border-bottom-left-image、border-top-left-image 和最中间的内容区域，如果元素的 border-width 刚好是 27px，则上面所说的九部分正好如图 3-7 所示的对应位置。

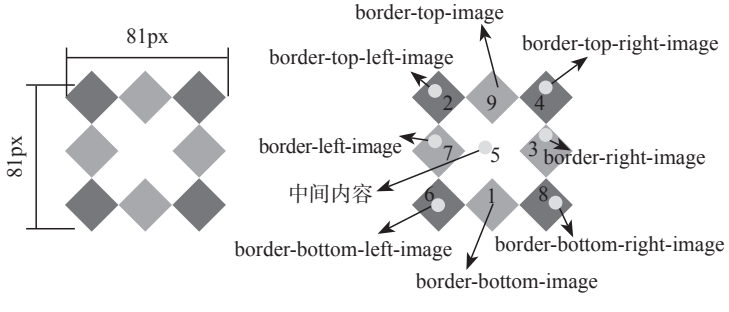


图 3-7　九宫图对应区域

图 3-7 所示的右边图片中，border-top-right-image，border-bottom-right-image，border-bottom-left-image，border-top-left-image 四个边角部分，在 border-image 中是没有任何展示效果的，把这四个部分（图 3-7 中对应的 2、4、6、8 部分）称做盲区；而对应的 border-top-image、border-right-image、border-bottom-image、border-left-image 四个区域在 border-

image 中是属于展示效果区域（图 3-7 中对应的 1、3、7、9 区域）。

其中上下区域 border-top-image 和 border-bottom-image 区域受到水平方向效果影响。如果是 repeat 则此区域图片会水平重复；如果是 round 则会水平平铺；如果是 stretch 则被水平拉伸，针对这个我们使用案例演示背景图片剪切的方法以及其对应的效果。

假设有一个元素边框背景定义了一个背景图片为 border.png，然后分别距离边框背景图片顶边（top）、右边（right）、底边 (bottom) 和左边（left）的 27px 处切一刀，此时 border-image-slice 的属性值为 (27,27,27,27)，由于四边的值相同，该属性可以简写为 border-image-slice:27。接下来来看几种不同的切片处理效果。

（1）水平效果

1）水平 round 效果。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>border-image 的水平 round 效果 </title>
  <style type="text/css">
    .border-image {
      width: 150px;
      height: 100px;
      border: 27px solid;
      -webkit-border-image: url(border.png) 27 round stretch;
      -moz-border-image: url(border.png) 27 round stretch;
      -o-border-image: url(border.png) 27 round stretch;
      -ms-border-image: url(border.png) 27 round stretch;
      border-image: url(border.png) 27 round stretch;
    }
  </style>
</head>
<body>
  <div class="border-image"></div>
</body>
</html>
```

对应的效果如图 3-8 所示。

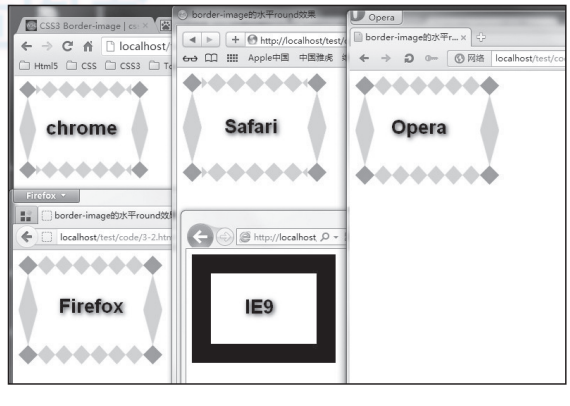


图 3-8　各浏览器下水平 round 演示效果

2）水平 repeat 效果。

```
.border-image {
      width: 150px;
      height: 100px;
      border: 27px solid;
      -webkit-border-image: url(border.png) 27 repeat stretch;
      -moz-border-image: url(border.png) 27 repeat stretch;
      -o-border-imag`e: url(border.png) 27 repeat stretch;
      -ms-border-image: url(border.png) 27 repeat stretch;
      border-image: url(border.png) 27 repeat stretch;
}
```

对应的效果如图 3-9 所示。

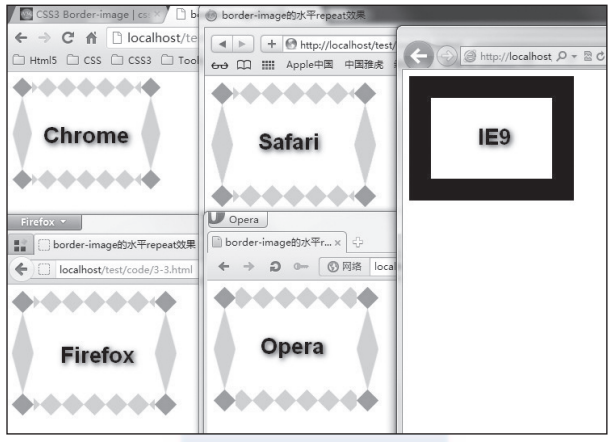


图 3-9　各浏览器下水平 repeat 演示效果

3）水平 stretch 效果。

```
.border-image {
      width: 150px;
      height: 100px;
      border: 27px solid;
      -webkit-border-image: url(border.png) 27 stretch stretch;
      -moz-border-image: url(border.png) 27 stretch stretch;
      -o-border-image: url(border.png) 27 stretch stretch;
      -ms-border-image: url(border.png) 27 stretch stretch;
      border-image: url(border.png) 27 stretch stretch;
    }
```

对应的效果如图 3-10 所示。

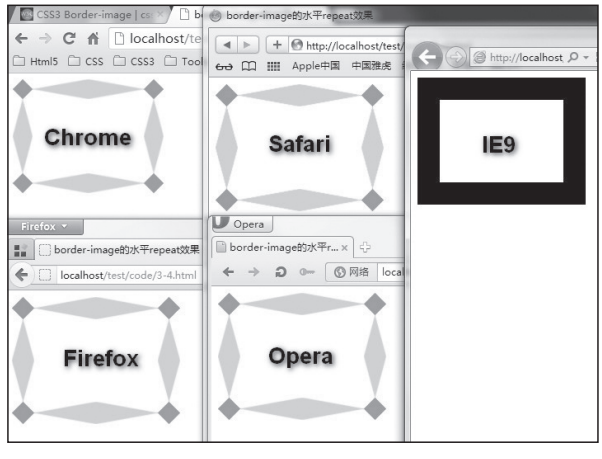


图 3-10　各浏览器下水平 stretch 演示效果

以上分别演示了 border-top-image、border-bottom-image 的 round、repeat 和 stretch 三种效果，从各浏览器下的效果对比图，很容易看出 border-image 在各浏览器下渲染的效果并不一致。在此，只想通过这几个效果来告诉大家 border-top-image 和 border-bottom-image 作用方向，以及对应的 round、repeat 和 stretch 各自会产生何种效果。

（2）垂直效果

通过前面学习，了解了 border-top-image 和 border-bottom-image 作用区域仅在水平方向，并不会影响垂直方向的效果。由此可以想象，border-right-image 和 border-left-image，只会作用在垂直方向，而且其同样具有 round、repeat、stretch 三种效果。接下来为了验证我们的猜想，一起看看 border-image 垂直方向的作用效果。

1）垂直 round 效果。

```
.border-image {
      width: 150px;
      height: 100px;
      border: 27px solid;
      -webkit-border-image: url(border.png) 27 stretch round;
      -moz-border-image: url(border.png) 27 stretch round;
      -o-border-image: url(border.png) 27 stretch round;
      -ms-border-image: url(border.png) 27 stretch round;
      border-image: url(border.png) 27 stretch round;
    }
```

对应的效果如图 3-11 所示。

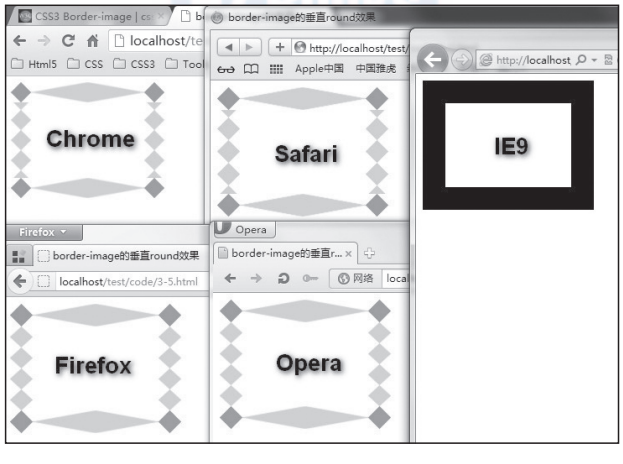


图 3-11　各浏览器下垂直 round 的演示效果

2）垂直 repeat 效果。

```
.border-image {
      width: 150px;
      height: 100px;
```

```
    border: 27px solid;
    -webkit-border-image: url(border.png) 27 stretch repeat;
    -moz-border-image: url(border.png) 27 stretch repeat;
    -o-border-image: url(border.png) 27 stretch repeat;
    -ms-border-image: url(border.png) 27 stretch repeat;
    border-image: url(border.png) 27 stretch repeat;
}
```

对应的效果如图 3-12 所示。

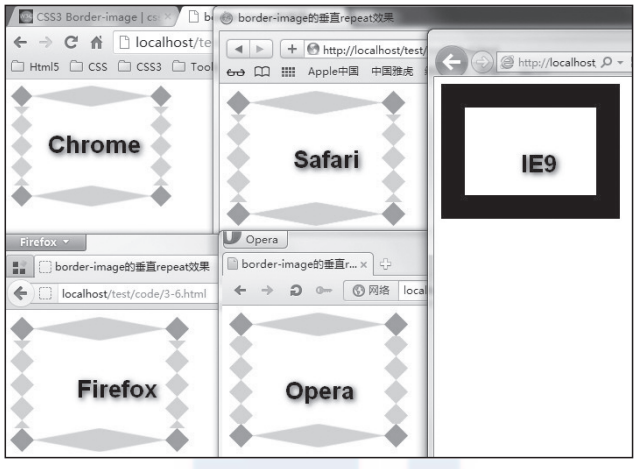


图 3-12　各浏览器下垂直 repeat 的演示效果

其实在演示水平 stretch 效果时也设置了垂直方向的 stretch，前面已说过，border-image-repeat 的默认值是 stretch，因此就算不设置任何值，都将用 stretch 来渲染，具体效果如图 3-10 所示。

通过上面几个示例比较，repeat 属性是边框中间向两端不断平铺，在平铺的过程中保持边框背景图片切片的大小，这样就造成了图示中的两端边缘处有被切的现象。而 round 却会对边框背景图的切片进行压缩（或伸缩）来适应边框宽度大小，进行排列，使其正好显示在区域内。stretch 有点特殊，只会把相应的切片进行拉伸，适应边框大小。

注意　在 Webkit 内核的浏览器下（Chrome、Safari），repeat 和 round 两者效果无区别。

3. border-image-width

语法：

```
boder-image-width: [<length> | <percentage> | <number> | auto] {1,4}
```

用来设置边框背景图片的显示大小，其实也可以理解为 border-width。虽然 W3C 定义了 border-image-width 属性，但各浏览器还是将其视为 border-width 来用，也就是说它和 border-width 的使用方法是一样的。

4. border-image-repeat

语法：

```
border-image-repeat: [stretch | repeat | round] {1,2}
```

用来指定边框背景图片的排列方式，其默认值为 stretch。这个属性设置参数和其他的不一样，border-image-repeat 不遵循 top、right 、bottom、left 的方位原则，它只接受两个（或一个）参数值，第一个值表示水平方向的排列方式，第二个值表示垂直方向的排列方式。当只取一个值时，表示水平和垂直方向的排列方式相同。如果你不显式设置任何值时，水平和垂直都会以其默认值 stretch 方式来进行排列。

为了能让大家更好地理解 border-image-repeat 的使用，下面将结合 border-image-slice 一起来看看 round、repeat 和 stretch 的实现原理。

在上面的示例中，使用一个 81px × 81px 的背景图片 border.png，分别在背景图片的顶边、右边、底边和左边的第 27px 处切了四刀，分成九个部分，每个方块的高和宽都是 27px × 27px。其中有四个部分是盲区，不管什么排列方式，这四个区都不变（border-top-right-image、border-bottom-right-image、border-bottom-left-image、border-top-left-image），而 border-top-image 和 border-bottom-image 两部分随着排列方式不同而效果不同，只限于水平方向的排列变化；另外两个 border-right-image 和 border-left-image 只是在垂直方向进行排列；最后中间部分同时在水平和垂直方向平铺，如图 3-13 所示。

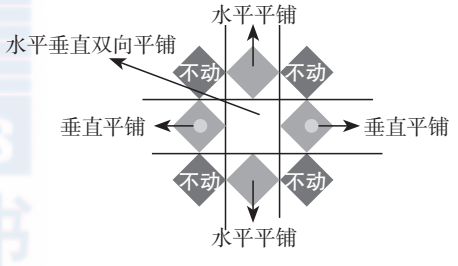


图 3-13　九宫图

前面把 border-image 像 background 一样分解介绍了其相关知识点，但在实际应用中，border-image 各属性必须写在一起，不能分解。下面给大家提供一个正确的速记法。

```
    border-image:<border-image-source> || <border-image-slice> [/<border-image-width>
] || <border-image-repeat>
```

3.3.3　浏览器兼容性

border-image 是 CSS3 新增的核心属性之一，也是一个非常实用的属性。随着主流浏览器的全面支持，这个属性会更实用。目前使用 border-image 属性，还是需要带上浏览器的私有属性，如表 3-4 所示。

表 3-4　各浏览器对 border-image 的私有属性

	Mozilla Gecko	Webkit	Presto	Konqueror	Internet Explorer
border-image	-moz-	-webkit-	-o-	-khtml-	-ms-

目前 IE 系列并不支持，也没有定义 -ms-border-image 的私有属性，其他各主流浏览器的支持情况如表 3-5 所示。

表 3-5　border-image 的浏览器兼容性

属性名					
border-image	×	3.5 + √	3.0 + √	10.5 + √	1.0 + √

3.3.4　border-image 属性的优势

border-image 功能强大，但受限于浏览器的支持度，其使用还是受到很大的限制。但相信这个功能将会在未来的 Web 应用中得到广泛的运用，尽展个人的魅力。

以前，给某个元素添加图片边框效果，唯一的办法就是使用背景图片。如果知道元素的尺寸会简单点，使用滑动门技术就可以实现，如果元素的尺寸不定，也就是说元素宽度、高度都自适应，单独使用背景图片还很难实现。在这种情况之下就需要添加很多空标签，使用九宫格来填充背景。使用 border-image 就轻松多了，只需要一张背景图片可以让某个元素实现图片边框的效果，或者其他效果，如圆角效果、阴影效果等。这样大大提高了开发效率，降低了开发成本。

3.3.5　实战体验：按钮圆角阴影效果

border-image 是 CSS3 中很实用的属性，接下来通过几个小案例帮助读者拓展自己的设计灵感，在实际中灵活运用这个属性。

制作按钮有很多种方法，但制作自适应宽度的圆角按钮还是很头痛的。早期使用四个圆角分别定位到按钮的四个角，接着有人使用一张圆角背景图片，通过滑动门技术来制作圆角按钮。随着 border-radius 的出现，很多情况下使用这个属性制作圆角按钮。本节介绍用 border-image 制作按钮的案例。

首先需要一张图片，当做 border-image 的背景图片，如图 3-14 所示。

接下来一起来看案例的实现代码。

图 3-14　制作按钮的背景图片

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 的 border-image 制作按钮 </title>
  <style type="text/css">
.border-image-btn {
  display: inline-block;
  border: 18px solid green;
  border-width: 0 18px;
    border-image: url("button_sprite.png") 0 18 50 18;
  -webkit-border-image: url("button_sprite.png") 0 18 50 18;
```

```
    -moz-border-image: url("button_sprite.png") 0 18 50 18;
    -o-border-image: url("button_sprite.png") 0 18 50 18;
    -ms-border-image: url("button_sprite.png") 0 18 50 18;

    padding: 13px 10px 17px;
    font-size: 16px;
    color: #fff;
    font-weight: bold;
    text-decoration:none;
    line-height: 15px;
    margin: 10px;
  }

  .border-image-btn:hover {
    border-image: url("button_sprite.png") 50 18 0 18;
    -webkit-border-image: url("button_sprite.png") 50 18 0 18;
    -moz-border-image: url("button_sprite.png") 50 18 0 18;
    -o-border-image: url("button_sprite.png") 50 18 0 18;
    -ms-border-image: url("button_sprite.png") 50 18 0 18;

    color: #000;
    border-color: yellow;
    text-decoration: none;
  }
    </style>
  </head>
  <body>
    <a href="#" class="border-image-btn">Click Me!</a>
    <a href="#" class="border-image-btn">用力点击我吧！</a>
    <a href="#" class="border-image-btn">CSS3 Border-image Button</a>
  </body>
  </html>
```

各浏览器下 border-image 制作按钮效果如图 3-15 所示。

从图 3-15 所示的效果中可以看出，border-image 在现代浏览器下得到较好的支持，唯有 IE 系列不支持（希望 IE 10 能支持）。接下来简单说一下原理。

这个简单的案例中，首先采用了一张 40px × 100px 的图片精灵（如图 3-14 所示）作为元素的边框背景

☆图 3-15　各浏览器下 border-image 制作按钮效果

图像，然后在距图片顶边 0px 处切第一刀，在距图片右边 18px 处切第二刀，在距图片底边 50px 处切第三刀，在距图片左边 18px 处切第四刀，从而组成九宫格。接下来利用 border-image 的拉伸属性，实现 border-image 制作按钮的默认效果。按钮悬浮状态下，采用相同的办法，只是改变切片的位置来达到一样的效果。这里还有关键一步，按钮的边框宽度只有左右，如果上下也要设置边框宽度，上面的切图就无法达到所需的效果。感兴趣的同学不妨一试。

在 Chrome 浏览器下 border-image 的标准写法写在最后，会造成不可预计的错误效果。

使用 border-image 除了可以制作上面的按钮效果之外，还可以制作 tabs 效果，其原理是一样的，此处不再做过多的重复阐述。但有一个关键处就是 border-image 的背景图片源要制作好，这里采用的背景图像如图 3-16 所示。

图 3-16　border-image 制作 tabs 效果的背景图

一起看看示例代码。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 的 border-image 制作 tabs 效果 </title>
  <style type="text/css">
  .tabs-box {
     border-bottom: 3px solid #9eaab6;
     margin: 0;
     padding: 0;
     overflow: hidden;
     zoom: 1;
  }
  .tabs-box li {
    float: left;
    display: inline;
    margin: 0 12px 0 0;
    list-style: none outside none;
    border: 1px solid #9EAAB6;
    padding: 5px;
    border-image: url("border-image-tab.png") 0 5 0 5;
    -moz-border-image: url("border-image-tab.png") 0 5 0 5;
    -webkit-border-image: url("border-image-tab.png") 0 5 0 5;
    -o-border-image: url("border-image-tab.png") 0 5 0 5;
    -ms-border-image: url("border-image-tab.png") 0 5 0 5;
    border-width: 0 5px;
    text-align: center;
    text-shadow: 0 -1px 0 rgba(0,0,0,0.8);
    color: rgba(0, 125, 200, 0.3);
  }
  </style>
</head>
```

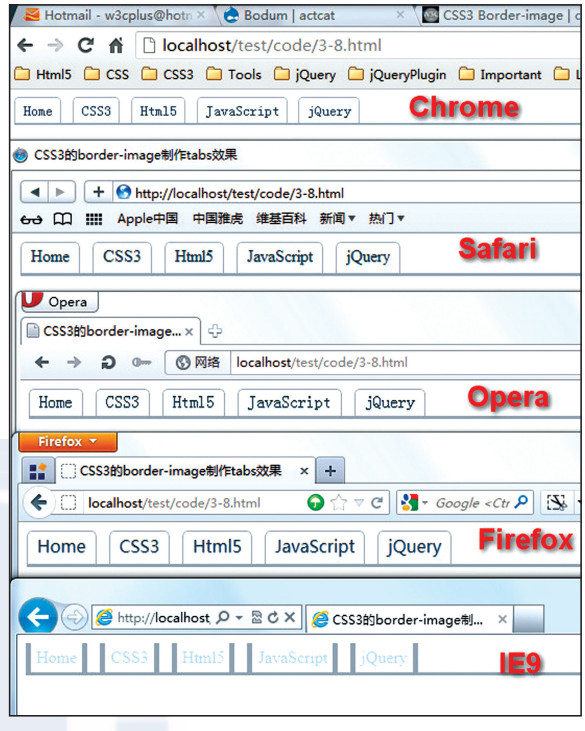
```
<body>
  <ul class="tabs-box">
    <li>Home</li>
    <li>CSS3</li>
    <li>Html5</li>
    <li>JavaScript</li>
    <li>jQuery</li>
  </ul>
</body>
</html>
```

示例效果如图 3-17 所示。

使用 border-image 制作 tabs 效果是不是比滑动制作要方便快捷。不过使用 border-image 制作需要掌握三点：1）源图片制作恰当；2）动刀切边框背景图片合理；3）边框宽度配合到位。

比如此例的切图如图 3-18 所示。

接下来看使用 border-image 制作圆角与阴影的示例。和前面两个示例一样，需要制作好的边框背景图，如图 3-19 所示。



☆图 3-17　各浏览器下 border-image 制作 tabs 效果

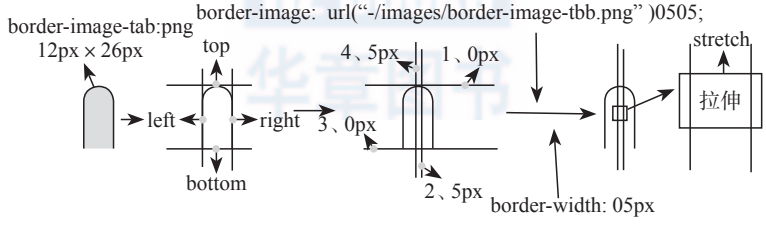


图 3-18　border-image 切图

图 3-19　制作圆角和阴影的边框单背景图

示例代码如下。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 的 border-image 制作圆角和阴影效果 </title>
  <style type="text/css">
```

```
  .border-image-drop-boxshadow {
   width: 150px;
   height: 50px;
   padding: 10px;
   margin: 10px;
   border: 1px solid #ccc;
   border-width: 7px 7px 16px;
   border-image: url("border-image-box-shadow.png") 7 7 16 7;
   -moz-border-image: url("border-image-box-shadow.png") 7 7 16 7;
   -webkit-border-image: url("border-image-box-shadow.png") 7 7 16 7;
   -o-border-image: url("border-image-box-shadow.png") 7 7 16 7;
   -ms-border-image: url("border-image-box-shadow.png") 7 7 16 7;
 }
 .box2{
  width: 200px;
  height: 100px;
 }
  </style>
</head>
<body>
  <div class="border-image-drop-boxshadow box1"> 小框 </div>
  <div class="border-image-drop-boxshadow box2"> 大框 </div>
</body>
</html>
```

效果如图 3-20 所示。

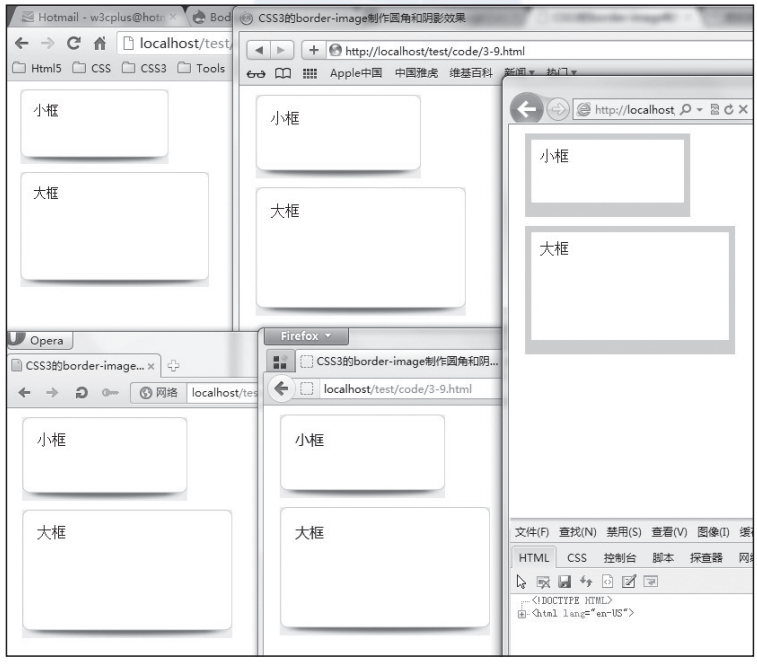


图 3-20　各浏览器下 border-image 制作圆角与阴影效果

以上几个案例可以体现 border-image 在实际应用中非常灵活，可以根据不同需求设计不同的边框背景图，设置不同 border-image-slice 属性值，从而设计各种各样的特殊边框样式，如带有纹理的相框、带有花边的边框等，大家不妨亲自体验一下。

3.4　CSS3 圆角边框属性

在 Web 页面上圆角效果很常见。圆角给页面增添曲线之美，让页面不那么生硬，但是为了设计圆角效果，Web 设计师们要花费更多的时间与精力。

3.4.1　border-radius 属性的语法及参数

CSS3 中专门针对元素的圆角效果增加了一个圆角属性 border-radius。Web 设计师不会为 Web 页面中的圆角效果纠结了。

语法：

```
border-radius: none | <length> {1,4}[/<length>{1,4}] ?
```

border-radius 是一种缩写方法。如果反斜杠符号“/”存在，“/”前面的值是设置元素圆角的水平方向半径，“/”后面的值是设置元素圆角的垂直方向的半径；如果没有“/”，则元素圆角的水平和垂直方向的半径值相等。另外四个值是按照 top-left、top-right、bottom-right 和 bottom-left 顺序来设置的，其主要会有以下四种情形出现。

1）border-radius:<length>{1} 设置一个值，top-left、top-right 、bottom-right 和 bottom-left 四个值相等，也就是元素四个圆角效果一样。

2）border-radius:<length>{2} 设置两个值，top-left 等于 bottom-right，并且取第一个值；top-right 等于 bottom-left，并且取第二值。也就是元素的左上角和右下角取第一个值，右上角和左下角取第二个值。

3）border-radius:<length>{3} 设置三个值，第一个值设置 top-left，第二个值设置 top-right 和 bottom-left，第三个值设置 bottom-right。

4）border-radius:<length>{4} 元素四个圆角取不同的值，第一个值设置 top-left，第二个值设置 top-right，第三个值设置 bottom-right，最后一个值设置 bottom-left。

border-radius 的属性参数非常简单，主要包含两个值。

❑ 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。不可以是负值。

如果要重置元素没有圆角，取值 none 并无效果，需要将元素的 border-radius 取值为 0。

border-radius 和 border 属性一样，可以将各个角单独拆分出来。这样 border-radius 就

派生出另外四个子属性，而且它们都是先 Y 轴再 X 轴。

❑ border-top-left-radius：<length>/<length>；定义元素左上角圆角。

❑ border-top-right-radius：<length>/<length>；定义元素右上角圆角。

❑ border-bottom-right –radius ：<length>/<length>；定义元素右下角圆角。

❑ border-bottom-left-radius：<length>/<length>；定义元素左下角圆角。

上面四个子属性取值和 border-radius 是一样的，只不过水平和垂直方向仅一个值，“ /”前面的值为水平方向半径，后面的值为垂直方向半径。如果第二个值省略，元素水平和垂直方向半径，其实就是以“ <length>”为半径的四分之一圆。如果任意一个值为“0”，这个角就不是圆角。

由于各浏览器厂商对 border-radius 子属性解析不一致，造成了各浏览器下的 border-radius 属性的派生子属性写法有所区别。

1）Gecko 内核浏览器（Firefox、Flock 等）。

```
-moz-border-radius-topleft:<length>/<length>;      左上角圆角
-moz-border-radius-topright:<length>/<length>;     右上角圆角
-moz-border-radius-bottomright:<length>/<length>; 右下角圆角
-moz-border-radius-bottomleft:<length>/<length>;   左下角圆角
```

2）Webkit 内核浏览器（Chrome、Safari 等）。

```
-webkit-border-top-left-radius: <length>/<length>;       左上角圆角
-webkit-border-top-right-radius: <length>/<length>;      右上角圆角
-webkit-border-bottom-right-radius: <length>/<length>;   右下角圆角
-webkit-border-bottom-left-radius: <length>/<length>;    左下角圆角
```

3）Presto 和 Trident 内核浏览器（Opera、IE 9+ 等）。

```
border-top-left-radius: <length>/<length>;       左上角圆角
border-top-right-radius: <length>/<length>;      右上角圆角
border-bottom-right-radius: <length>/<length>;   右下角圆角
border-bottom-left-radius: <length>/<length>;    左下角圆角
```

border-radius 派生的子属性虽然方便为元素设置指定角的圆角，但为了兼容各浏览器的新老版本写法，不得为样式增加额外的代码。

```
/*Firefox 浏览器 */
-moz-border-radius-topleft: <length>/<length>;           右上角圆角
-moz-border-radius-topright: <length>/<length>;          右上角圆角
-moz-border-radius-bottomright: <length>/<length>;       右下角圆角
-moz-border-radius-bottomleft: <length>/<length>;        左下角圆角
/*Chrome 和 Safari 浏览器 */
-webkit-border-top-left-radius:  <length>/<length>;      左上角圆角
-webkit-border-top-right-radius:  <length>/<length>;     右上角圆角
-webkit-border-bottom-right-radius: <length>/<length>;   右下角圆角
-webkit-border-bottom-left-radius:  <length>/<length>;   左下角圆角
/*Opera、IE 9+、W3C 标准写法 */
```

```
border-top-left-radius: <length>/<length>;        左上角圆角
border-top-right-radius: <length>/<length>;       右上角圆角
border-bottom-right-radius: <length>/<length>;    右下角圆角
border-bottom-left-radius: <length>/<length>;     左下角圆角
```

这样给元素设置单个圆角效果是件非常痛苦的事情，而且难以维护，也容易出错。其实给元素设置单个圆角效果，完全可以借助 border-radius 属性的标准写法，只是需要将其他顶边的圆角半径值设置为 0。例如，只要给元素左上角设置圆角效果。

```
border-radius: 5px 0 0 0;/* 左上角设置圆角 */
```

3.4.2　border-radius 属性使用方法

前面了解了 border-radius 的语法，同时知道 border-radius 属性可以包含两个参数值，第一个是水平圆角半径值，第二个是垂直圆角半径值，而且两个参数值使用“/”分开，如图 3-21 所示[⊖]。

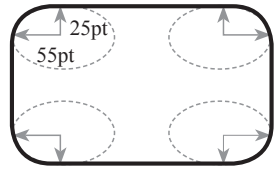


图 3-21　border-radius 的圆角

1. 水平和垂直半径一样

通过一些简单的示例代码进一步加强对 border-radius 的理解。

1）border-radius 只设置一个值。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>CSS3 的 border-radius 制作圆角 </title>
  <style type="text/css">
    .border-radius {
      width: 250px;
      height: 100px;
      border: 10px solid orange;
      border-radius: 10px;
    }
  </style>
</head>
<body>
  <div class="border-radius"></div>
</body>
</html>
```

此时，元素四个角都具有圆角，而且圆半径值一样，效果也一样，如图 3-22 所示。

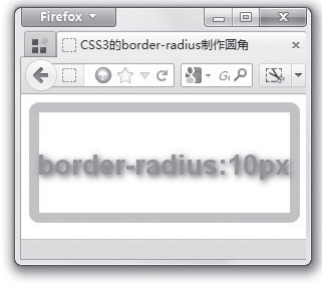


图 3-22　border-radius 取一个值时圆角边框效果

2）border-radius 设置两个值。

```
.border-radius {
      width: 350px;
```

⊖ 图 3-21 来自于 http://www.w3.org/TR/css3-background/#border-radius。

```
        height: 100px;
        border: 10px solid orange;
        border-radius: 10px  30px;
    }
```

此时 top-left 等于 bottom-left 并且它们取第一个值 10px；top-right 等于 bottom-left 并且取第二个值 30px。即元素左上角和右下角圆角相同，其圆角半径值为 10px，而元素右上角和左下角圆角相同，其圆角半径值为 30px，如图 3-23 所示。

3）border-radius 设置三个值。

```
.border-radius {
        width: 350px;
        height: 100px;
        border: 10px solid orange;
        border-radius: 10px  50px  30px;
 }
```

此时 top-left 取第一个值 10px，top-right 和 bottom-left 圆角效果一样，取第二个值 50px，bottom-right 取第三个值 30px，如图 3-24 所示。

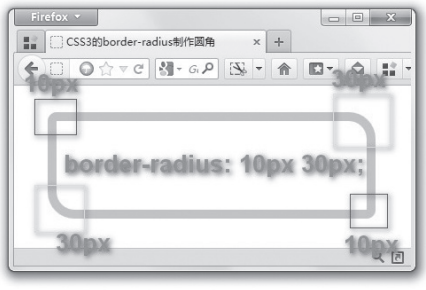


图 3-23　border-radius 取两个值的圆角边框效果

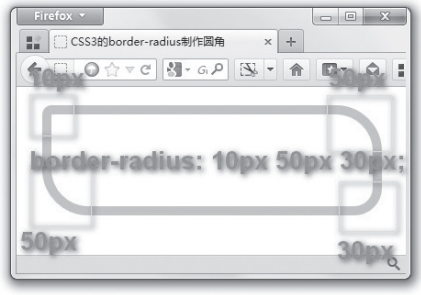


图 3-24　border-radius 取三个值时圆角边框效果

4）border-radius 设置四个值。

```
.border-radius {
        width: 350px;
        height: 100px;
        border: 10px solid orange;
        border-radius: 10px  20px  30px  40px;
 }
```

此时，左上角取第一个参数值 10px，右上角取第二个参数值 20px，右下角取第三个参数值 30px，左下角取第四个参数值 40px；如果四个值不相同时，意味着元素的四个顶角的圆角效果都不一样，如图 3-25 所示。

2. 单独设置水平和垂直半径值

上面展示的是 border-radius 设置圆角的水平和垂直半径都是一样的，不过前面介绍过，

border-radius 设置圆角时，可以把圆角的水平和垂直半径值单独设置，此时就需要使用以“/”来区别。“/”前面的表示圆角的水平半径，而“/”后面的值表示圆角的垂直半径。一起来看一个简单的实例，设置不规则圆角边框。

```
.border-radius {
      width: 350px;
      height: 100px;
      border: 10px solid orange;
      border-radius: 60px  40px  30px  20px / 30px 20px 10px 5px;
 }
```

border-radius 设置水平 / 垂直两个半径参数时，元素的每个角不是四分之一圆角，得到的圆角效果是不规则的。元素左上角的是一个水平半径为 60px，垂直半径为 30px 的不规则圆角；右上角是一个水平半径为 40px，垂直半径为 20px 的不规则圆角；右下角是一个水平半径为 30px，垂直半径为 10px 的不规则圆角；左下角是一个水平半径为 20px，垂直半径为 5px 的不规则圆角，如图 3-26 所示。

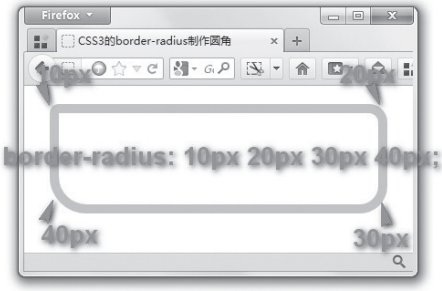


图 3-25　border-radius 设置四个值的圆角边框效果

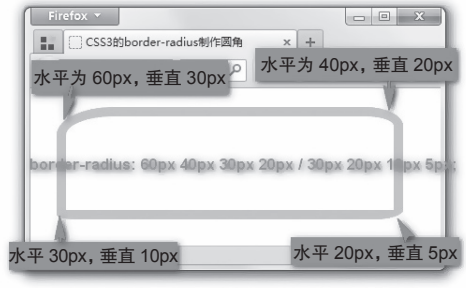


图 3-26　border-radius 设置不规则圆角边框效果

其实 border-radius 的水平和垂直半径也遵循前面介绍的规则，可以设置 1 ～ 4 个值的集合，同时它们分别遵循 CSS 赋值规则。但分开设置元素各个顶角的圆角的水平和垂直半径圆角效果时，不需要“/”，如下所示。

```
border-top-left-radius: 10px  50px;
border-top-right-radius: 20px 60px;
border-bottom-left-radius: 20px   60px;
border-bottom-right-radius: 30px  50px;
```

如果加上反而是一种错误的写法。

```
border-top-left-radius: 10px / 50px;
border-top-right-radius: 20px / 60px;
border-bottom-left-radius: 20px  /  60px;
border-bottom-right-radius: 30px  / 50px;
```

3. 制作单个圆角边框

使用 border-radius 可以给元素设置圆角边框，还可以使用 border-radius 的派生子属性来定义元素的圆角边框效果。

要给元素设置单个圆角效果，不一定需要使用 border-radius 派生的子属性，完全可以使用下面的方法来替代。

```
border-radius: 50px 0 0 0;
```

上面的代码就是给元素设置了一个左上角圆角边框效果，其效果等同于：

```
-moz-border-radius-topleft: 50px;
-webkit-border-top-left-radius: 50px;
border-top-left-radius: 50px;
```

这两种方法制作出来的效果都是一样的，如图 3-27 所示。

4. 特殊应用

前面所了解的都是 border-radius 一些常见的运用，其实 border-radius 还有几个特殊的应用。

1）border-radius 还有一个内半径和外半径的区别，元素边框值较大时，效果就很明显。当 border-radius 半径值小于或等于 border 的厚度时，元素边框内部就不具有圆角效果。如：

```
.border-radius {
      width: 350px;
      height: 100px;
      border: 30px solid orange;
      border-radius: 30px;
 }
```

效果如图 3-28 所示。

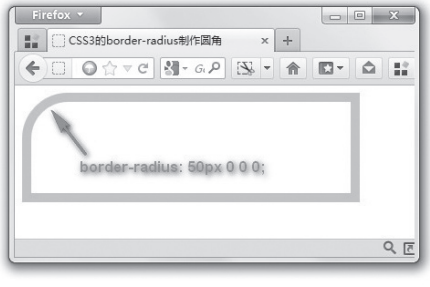


图 3-27　border-radius 制作单个圆角边框效果

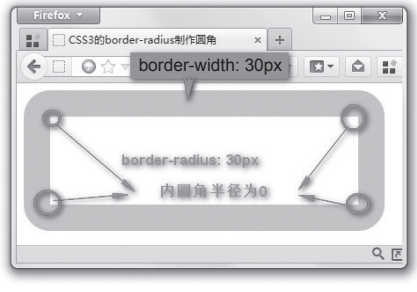


图 3-28　圆角半径等于边框厚度时效果

在这个基础上，把圆角半径值调大些，比边框值大。

```
.border-radius {
      width: 350px;
```

```
    height: 100px;
    border: 30px solid orange;
   border-radius: 35px;
  }
```

这个时候内圆角就出来了，如图 3-28 所示。

为何当 border-radius 的半径小于或等于元素的边框厚度时，内部是直角效果？因为 border-radius 内边半径（内径）等于外边半径（外径）减去对应的边框宽度。

- ❏ border-radius 半径值与 border-width 值等于或小于 0 时，元素内角为直角，如图 3-28 所示。
- ❏ border-radius 半径值与 border-width 值大于 0 时，元素内角具有圆角效果，其圆角半径为它们的差值。差值越大，圆角幅度也大，反之圆角幅度也小，如图 3-29 所示。

2）第二种特殊应用是，元素相邻边有不同的宽度，这个角将会从宽的边平滑过渡到窄的一边，其中一条边甚至可以是 0，元素相邻转角是由大向小转。例如：

```
.border-radius {
    width: 350px;
    height: 100px;
    border: 30px solid orange;
    border-width: 20px 5px 30px 60px;
   border-radius: 100px;
  }
```

效果如图 3-30 所示。

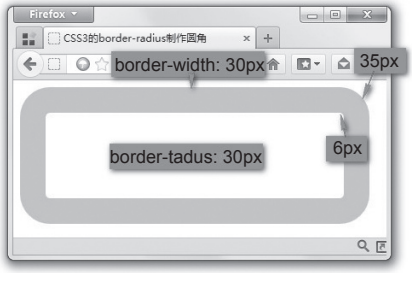


图 3-29　圆角半径大于边框厚度时效果

图 3-30　元素相邻转角是由大向小转效果

当元素相邻两条边颜色和线条样式不同时，两条相邻边颜色和样式转变的中心点是在一个和两边宽度成正比的角上。如果两条边宽度相同，这个临界点应该在一个 45 度角上，如果一条边是另外一条边的 2 倍，这个临界点就在一个 30 度角上。界定这个转变的线就是连接在内外曲线上的两个点的直线。一起来看个示例，给元素四边设置不同的颜色和宽度。

```
.border-radius {
    width: 350px;
    height: 100px;
```

```
        border: 30px solid orange;
        border-width: 35px 35px 60px 30px;
        border-color: orange red green blue;
      border-radius: 80px;
}
```

效果如图 3-31 所示。

5. 图片应用圆角

border-radius 能应用在各个元素中，但在 img 和 table 应用时会有点差别的，首先看图片上应用 border-radius 时的情况。在 img 上应有用 border-radius 到目前只有在 Webkit 内核浏览器不能对图片进行剪切，来看一个图片应用圆角的实例。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title> 图片上的圆角应用 </title>
  <style type="text/css">
    img {
      border: 5px solid red;
      border-radius: 10px;
    }
  </style>
</head>
<body>
  <img src="border.jpg" alt=""  width="150"  height="150" />
</body>
</html>
```

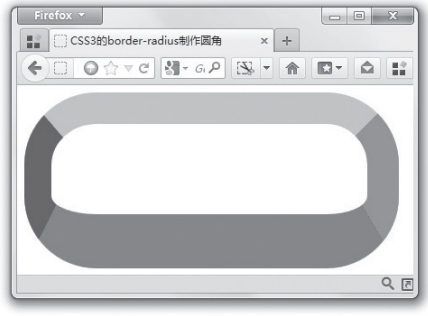


图 3-31　border-radius 临界分界点效果

各浏览器下图片圆角效果如图 3-32 所示。

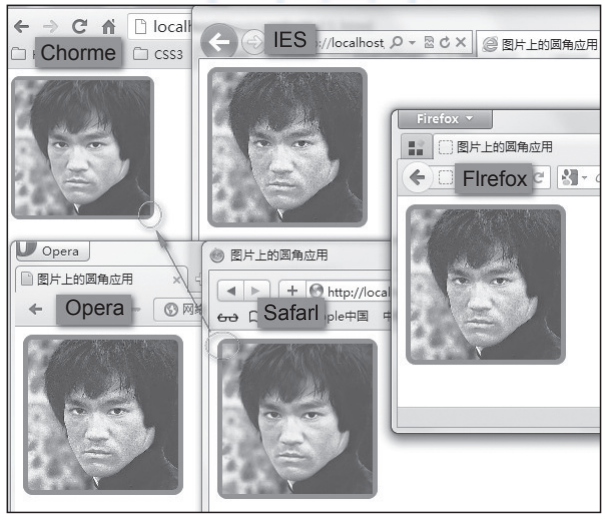


图 3-32　各浏览器下图片圆角效果

正如图 3-32 效果所示，图片在 Webkit 内核浏览器（例如 Chrome、Safari）下根本没有圆角效果，图片不会被圆角剪切。如果需要让浏览器达到一致效果，可以把图片转换成元素的背景图片，然后再给元素定义圆角效果。这时需要借助 jQuery 来实现。例如：

```
$("img").load(function(){
$(this).wrap(function(){
return '<span class="' + $(this).attr('class') + '" style="background:url(' +
      $(this).attr('src') + ') no-repeat center center; width: ' +
      $(this).width() + 'px; height: ' + $(this).height() + 'px;"></span>';
});
(this).css("opacity","0");
});
```

详细的解决方案可以参考 http://www.w3cplus.com/css3/jquery-css3-rounded-image。

6. 表格应用圆角

另外表格元素 table 使用 border-radius 是不一样的，当表格样式属性 border-collapse 是 collapse 时，表格不能正常显示，只有 border-collapse 属性值为 separate 时，表格圆角才能正常显示。例如：

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>表格的圆角应用 </title>
  <style type="text/css">
    table {
      margin: 10px;
      border:5px solid orange;
      border-radius: 10px;
    }
    .table1 {
      border-collapse: collapse;
    }
    .table2 {
      border-collapse: separate;
    }
  </style>
</head>
<body>
  <table class="table1">
    <tr>
      <td>border-collapse:collapse</td>
    </tr>
  </table>
  <table class="table2">
    <tr>
      <td>border-collapse:separate</td>
    </tr>
```

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

效果如图 3-33 所示。

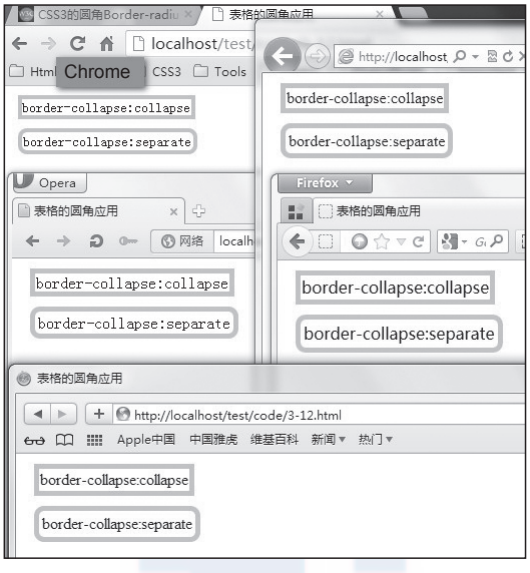


图 3-33　各浏览器下表格圆角边框效果

3.4.3　浏览器兼容性

目前，border-radius 属性除了 IE 老版本之外的浏览器都得到了较好的支持，如 IE 9+、Firefox 4+、Chrome 5+、Safari 5+、Opera 10.5+ 版本都支持 border-radius 的标准写法。如果需要支持一些老版本，还要添加各浏览器的私有前缀，如 Firefox 3、Chrome 4、Safari 3.1 ～ 4。而 IE 8 及其以下版本的浏览器不支持 border-radius 属性，如表 3-6 所示。

表 3-6　border-radius 浏览器兼容性

属性名					
border-radius	9 + √	3.0 + √	1.0 + √	10.5 + √	3.0 + √

CSS3 的 border-radius 属性目前在 Web 中的使用随处可见，特别是国外的 Web 运用上，国内很多 Web 设计师也逐渐在使用。在 IE 低版本浏览器下，Web 设计师可以采用以下方案来处理兼容性。

- ❏ 使用第三方插件，例如 IE -css3.js、PIE 或者其他 JavaScript 脚本插件。
- ❏ 采用渐近增强，在不支持 border-radius 属性的浏览器采用另一套样式，也就是 CSS2 中的图片实现圆角方法优雅降级，在不支持 border-radius 的浏览器默认显示直角。

3.4.4　border-radius 属性的优势

使用 CSS 来实现宽度固定的圆角效果，采用背景图片配合滑动门技术实现还是一种不错的方法。但是，如果想要一个宽度不固定的元素就变得复杂了。宽度不定，就意味着这个元素在水平和垂直方向都能灵活地变化。实现元素四个圆角效果，就需要制作四个圆角背景图片，并且进行合理的放置，这样还需要添加四个额外的标签来辅助完成。当然还可以制作两个超大、超宽的背景图片，这个方法虽然减少了两张背景图片，以及四个 HTML 标签，但另一个问题又随之产生，图片尺寸过大增加了图片载入的难度，直接影响了网站的性能。如果使用 CSS3 的 border-radius 属性制作圆角，就不需考虑元素是否可以自由扩展，同时也不需要为了实现圆角制作不同的圆角背景图片，从而获得了极大的灵活性、维护性。

使用 CSS3 的 border-radius 属性来代替 CSS 之前使用图片制作圆角，在部分不支持 border-radius 的浏览器上，牺牲了一点效果的一致性，但这不是问题。我们所做的是一种渐进增强，一种优雅降级，即使用圆角的元素在不支持 CSS3 的 border-radius 属性的浏览器下完全有效果且易读，只不过在支持的浏览器下，使用 border-radius 的元素会更美观，视觉效果更细腻圆润。你或者客户希望在所有浏览器下能达到一样的圆角效果，可以考虑这样的实现方法：“在支持的浏览器下使用 CSS3 的 border-radius 属性，而在不支持的浏览器下，可以考虑图片，或者第三方插件的方法来实现”。

3.4.5　实战体验：制作特殊图形

border-radius 属性除了可以实现元素的圆角效果，还可以制作一些特殊的图形效果，如圆形、半圆形、扇形、椭圆形和不规则的圆角图形等。

1. 圆形

border-radius 制作圆角有两点技巧。

❑ 元素的宽度和高度相同。

❑ 圆角的半径值为元素宽度或宽度的一半或者直接设置圆角半径值为 50%。

不过早期的 Webkit 内核浏览器不支持百分比值。例如：

```
div {
      width: 100px;
      height: 100px;
      background-color: orange;
      border-radius: 50%;
    }
```

效果如图 3-34 所示。

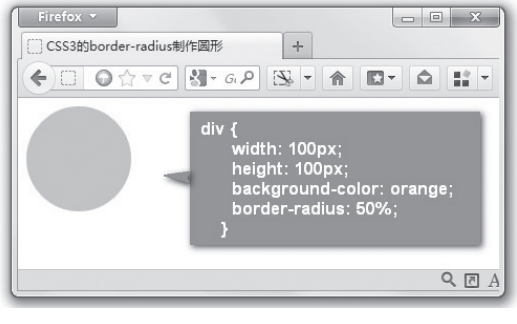


图 3-34　border-radius 制作圆形

2. 半圆

border-radius 制作半圆与制作圆形的方法是一样的，只是元素的宽度与圆角方位要配

合一致，不同的宽度和高度比例，以及圆角方位，可以制作上半圆、下半圆、左半圆和右半圆效果。例如：

```
.semicircle {
      background-color: orange;
      margin: 30px;
    }
    .top {
      width: 100px;/* 宽度为高度的 2 倍 */
      height: 50px;
      border-radius: 50px 50px 0 0;/* 圆角半径为高度的值 */
    }
    .right {
      height: 100px;/* 高度为宽度的 2 倍 */
      width: 50px;
      border-radius: 0 50px 50px 0;/* 圆角半径为宽度的值 */
    }
    .bottom {
      width: 100px;/* 宽度为高度的 2 倍 */
      height: 50px;
      border-radius: 0 0 50px 50px;/* 圆角半径为高度的值 */
    }
    .left {
      width: 50px;
      height: 100px;/* 高度为宽度的 2 倍 */
      border-radius: 50px 0 0 50px;/* 圆角半径为宽度的值 */
    }
```

效果如图 3-35 所示。

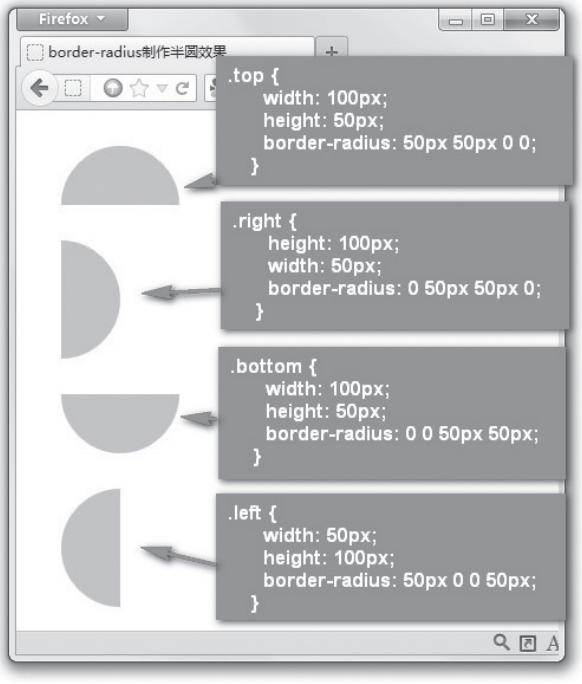


图 3-35　border-radius 制作半圆形

border-radius 制作半圆有两个小技巧：

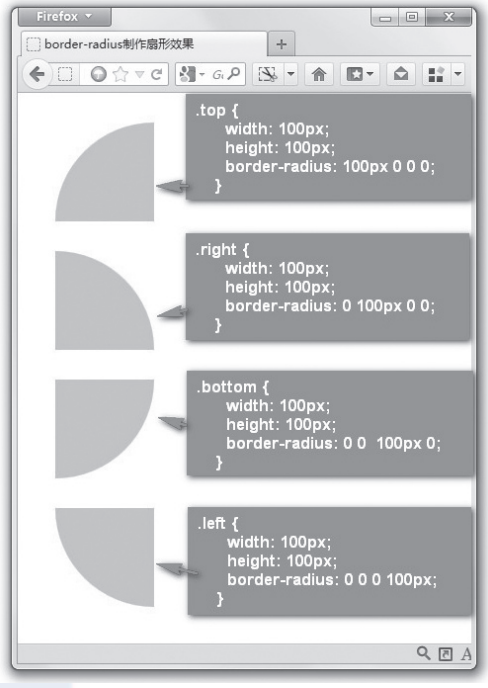
- ❑ 制作上半圆或下半圆，元素的宽度值是高度值的 2 倍，而且圆角半径值为元素的高度值；
- ❑ 制作左半圆或右半圆，元素的高度值是宽度值的 2 倍，而且圆角半径值为元素的宽度值。

3. 扇形

border-radius 制作扇形，其实就是使用 border-radius 属性制作四分之一圆形。遵循“三同，一不同”原则，其中“三同”是指元素的宽度、高度和圆角半径值相同，而“一不同”指的是圆角位置不同。根据圆角取值位

置不一样，可以分左上、右上、右下和左下四种扇形效果。例如：

```
.quarterCircle {
      background-color: orange;
      margin: 30px;
    }
    .top {
      width: 100px;
      height: 100px;
      border-radius: 100px 0 0 0;
    }
    .right {
      width: 100px;
      height: 100px;
      border-radius: 0 100px 0 0;
    }
    .bottom {
      width: 100px;
      height: 100px;
      border-radius: 0 0  100px 0;
    }
    .left {
      width: 100px;
      height: 100px;
      border-radius: 0 0 0 100px;
    }
```



效果如图 3-36 所示。

图 3-36　border-radius 制作四分之一圆

4. 椭圆

椭圆其实就是一个圆形受到挤压而成的一种形状，border-radius 制作椭圆也非常方便，只受限于元素的宽度或高度，然后就是圆角半径，制作椭圆的圆角半径和其他图形有所不一样，需要设置圆角的水平和垂直方向的半径值。椭圆有两种，一种是水平的，另外一种是垂直的。它们之间的差别只是方向性的区别，其制作方法是一样的。

制作水平椭圆，元素宽度是高度的 2 倍，而且 border-radius 的水平半径等于元素宽度，垂直半径等于元素高度；而垂直椭圆刚好与水平椭圆的参数相反。例如：

```
.oval {
      background-color: orange;
      margin: 30px;
    }
    .hov {
      width: 100px;
      height: 50px;
      border-radius: 100px / 50px;
    }
    .ver {
      width: 50px;
      height: 100px;
```

```
        border-radius: 50px  / 100px;
    }
```

效果如图 3-37 所示。

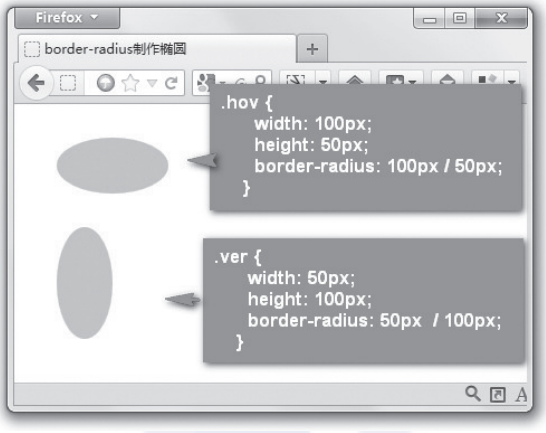


图 3-37　border-radius 制作椭圆

上面几个小案例都是使用 border-radius 配合元素的其他属性实现不同的图形效果，也可以使用 border-radius 制作更多的图形效果（或者说圆角效果）来适合项目需求。

3.5　CSS3 盒子阴影属性

box-shadow 也是 CSS3 新增的一个重要属性，用来定义元素的盒子阴影。本节主要介绍 CSS3 的 box-shadow 的属性以及如何使用。

3.5.1　box-shadow 属性的语法及参数

在具体学习 box-shadow 使用方法之前，我们必须先知道 box-shadow 使用的语法规则。

```
box-shadow:none | [ <length> <length> <length>?<length>? || <color> ] [ ,
<length> <length> <length>? <length>?|| <color> ]+
```

上面的语法规则可以简写如下：

```
box-shadow:none | [inset  x-offset y-offset blur-radius spread-radius color],
[inset  x-offset y-offset blur-radius spread-radius color]
```

box-shadow 属性可以使用一个或多个投影，如果使用多个投影时必须使用逗号“,”隔开。

其实 box-shadow 属性很简单，可以为其设置以下参数。

- none：默认值，元素没有任何阴影效果。
- inset ：阴影类型，可选值。如果不设置，其默认的投影方式是外阴影；如果取其唯一值“inset”，就是给元素设置内阴影。
- x-offset：阴影水平偏移量，其值可以是正负值。如果取正值，则阴影在元素的右边，反之取负值，阴影在元素的左边。
- y-offset：阴影垂直偏移量，其值可以是正负值。如果取正值，则阴影在元素的底部，反之取负值，阴影在元素的顶部。
- blur-radius ：阴影模糊半径，可选参数。其值只能是正值，如果取值为“0”时，表示阴影不具有模糊效果，如果取值越大，阴影的边缘就越模糊。
- spread-radius ：阴影扩展半径，可选参数。其值可以是正负值，如果取值为正值，则整个阴影都延展扩大，反之取值为负值，则整个阴影都缩小。
- color ：阴影颜色，可选参数，如果不设定任何颜色时，浏览器会取默认色，但各浏览器默认色不一样，特别是在 Webkit 内核下的浏览器将无色，也就是透明，建议不要省略这个参数。

3.5.2　box-shadow 属性使用方法

和 PSD 软件制作图片相比，box-shadow 修改元素的阴影效果要方便得多，因为 box-shadow 可以修改六个参数，得到不同的效果。下面结合一些简单的案例来对 box-shadow 属性进行演示说明。

1. 单边阴影效果

定义元素的单边阴影效果和调协 border 的单边边框颜色是相似的，例如：

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>box-shadow 设置单边阴影效果 </title>
  <style type="text/css">
    .box-shadow {
      width: 200px;
      height: 100px;
      border-radius: 5px;
      border: 1px solid #ccc;
      margin: 20px;
    }
    .top {
      box-shadow: 0 -2px 0 red;
    }
    .right {
      box-shadow: 2px 0 0 green;
    }
```

```
      .bottom {
        box-shadow: 0 2px 0 blue;
      }
      .left {
        box-shadow: -2px 0 0 orange;
      }
    </style>
  </head>
  <body>
    <div class="box-shadow top"></div>
    <div class="box-shadow right"></div>
    <div class="box-shadow bottom"></div>
    <div class="box-shadow left"></div>
  </body>
  </html>
```

效果如图 3-38 所示。

这个案例中，使用 box-shadow 给元素设置了顶边、右边、底边和左边的单边阴影效果。主要通过 box-shadow 的水平和垂直阴影的偏移量来实现，其中 x-offset 为正值时，生成右边阴影，反之为负值时，生成左边阴影；y-offset 为正值时，生成底部阴影，反之为负值时生成顶部阴影。此例中是一个单边实影投影效果（阴影模糊半径为 0），但是如果阴影的模糊半径不是 0，上面的方法还能不能实现单边阴影效果呢？不急着来回答，在上面的实例中添加一个模糊半径，例如：

```
.top {
      box-shadow: 0 -2px 5px red;
    }
    .right {
      box-shadow: 2px 0 5px green;
    }
    .bottom {
      box-shadow: 0 2px 5px blue;
    }
    .left {
      box-shadow: -2px 0 5px orange;
    }
```

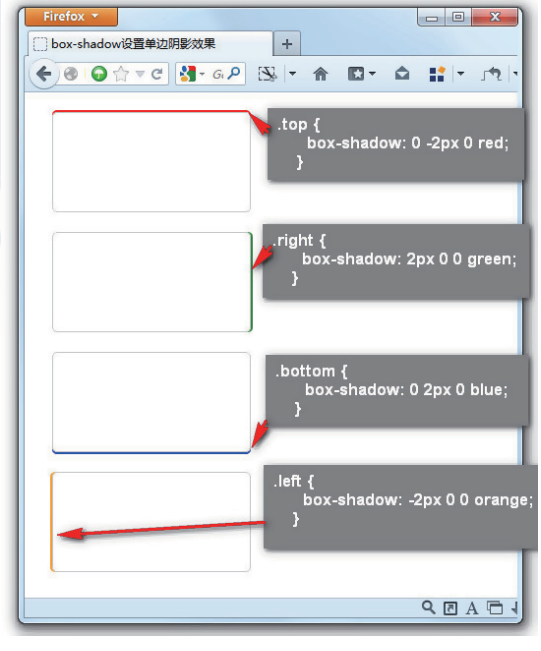


图 3-38　box-shadow 单边阴影效果

图 3-39 说明，这个效果并不是理想的单边阴影效果，当 box-shadow 添加了 5px 阴影模糊半径后，阴影不再是实影投影，阴影清晰度向外扩散，更具阴影的效果。但造成了另一个问题，给元素其他三个边加上淡淡的阴影效果，可这并不是设计需要的效果。

那究竟要怎么做呢？此时，box-shadow 属性中的阴影扩展半径（spread-radius）会是一

个很关键的属性，要实现单边阴影效果，必须配上这个属性（除单边实影之外）。

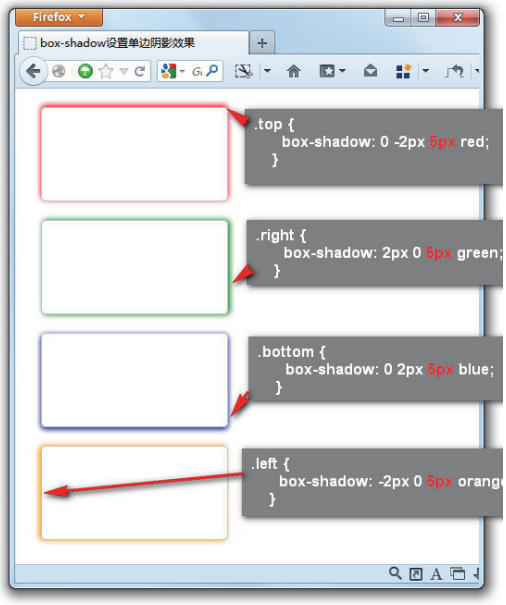


图 3-39　有模糊值的单边阴影效果

```
.top {
      box-shadow: 0 -4px 5px -3px red;
    }
    .right {
      box-shadow: 4px 0 5px -3px green;
    }
    .bottom {
      box-shadow: 0 4px 5px -3px blue;
    }
    .left {
      box-shadow: -4px 0 5px -3px orange;
    }
```

上面的代码调整了阴影的位移量，新增了 box-shadow 的扩展半径，最终效果如图 3-40 所示。

图 3-40　box-shadow 制作单边阴影效果

注意　各浏览器下显示效果略有细节差别。

2. 四边相同阴影效果

box-shadow 给元素设置相同的四边阴影效果，其实分为两种，在这里先看第一种。

（1）只设置阴影模糊半径和阴影颜色

只设置阴影模糊半径和阴影颜色。例如：

```
.box-shadow{
      width: 200px;
      height: 100px;
      border-radius: 10px;
      border: 1px solid #ccc;
      margin: 20px;
      box-shadow: 0 0 10px #06c;
    }
```

效果如图 3-41 所示。

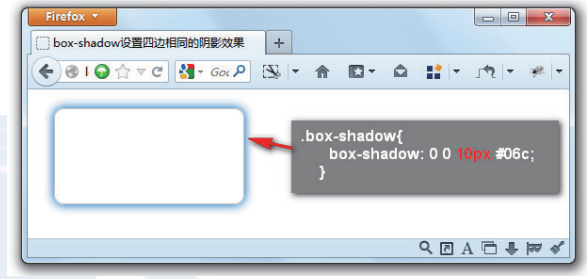


图 3-41　box-shadow 设置四边相同阴影

在这个示例基础上，添加 box-shadow 扩展半径还可以控制阴影深度，如果取正值将加深阴影的深度，如果取负值可以向内压缩阴影，直到扩展半径等于模糊半径时，阴影会完全消失。例如：

```
.box-shadow{
      width: 200px;
      height: 100px;
      border-radius: 10px;
      border: 1px solid #ccc;
      margin: 20px;
      box-shadow: 0 0 10px 10px #06c;
    }
```

效果如图 3-42 所示。

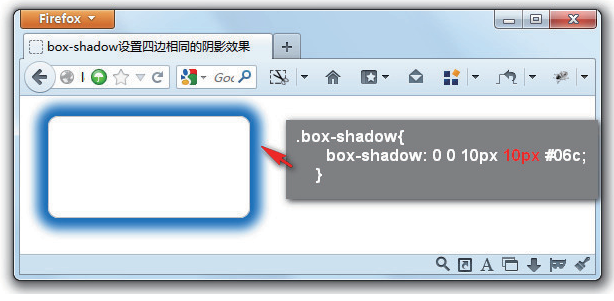


图 3-42　box-shadow 扩展半径加强阴影效果

接下来，将扩展半径改成“−10px”，此时将看不到任何阴影效果。

```
.box-shadow{
      width: 200px;
      height: 100px;
      border-radius: 10px;
      border: 1px solid #ccc;
      margin: 20px;
      box-shadow: 0 0 10px  -10px #06c;
    }
```

效果如图 3-43 所示。

（2）只设置扩展半径和阴影颜色

另外一种设置元素四边相同阴影效果，是设置扩展半径和阴影颜色，先来看一个简单的示例。

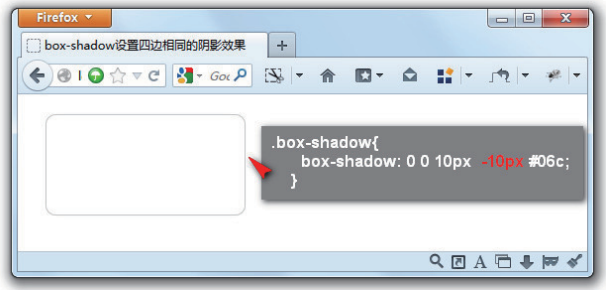


图 3-43　无阴影效果

```
.box-shadow{
      width: 200px;
      height: 100px;
      border-radius: 10px;
      border: 1px solid #ccc;
      margin: 20px;
      box-shadow: 0 0  0 10px #06c;
    }
```

效果如图 3-44 所示。

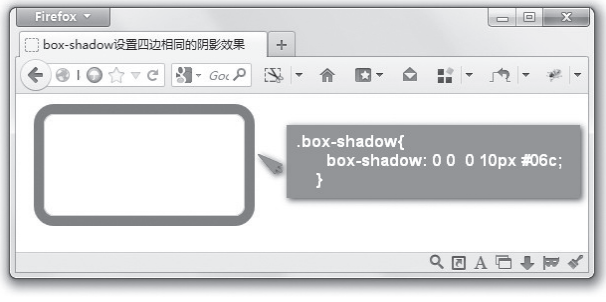


图 3-44　四边相同阴影效果

从图 3-44 可知道，box-shadow 制作的阴影效果和元素设置“10px”实线边框一样。

```
border:10px solid  #06c;
```

如此一来，可以利用 box-shadow 扩展半径制作类似于边框的效果，但实质上并非边框，因为 box-shadow 并不是盒模型中的元素，不会计算到内容宽度。具体来看一个 box-shadow 与 border 的对比示例。

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>border 与 box-shadow</title>
```

```
  <style type="text/css">
   .box {
    width: 200px;
    height: 100px;
    text-align: center;
    line-height: 100px;
    float: left;
    margin: 30px;
   }
   .border{
    border: 10px solid red;
   }
   .box-shadow {
    box-shadow: 0 0 0 10px red;
   }
  </style>
</head>
<body>
  <div class="box border">Border</div>
  <div class="box box-shadow">Box-shadow</div>
</body>
</html>
```

效果如图 3-45 所示。

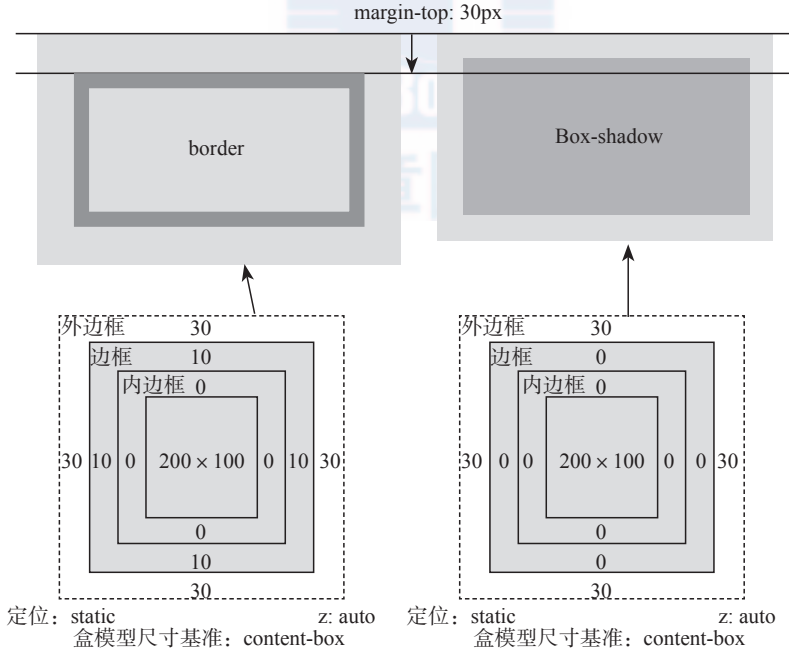


图 3-45　border 与 box-shadow 制作边框效果对比

图 3-45 证实了 box-shadow 不会影响页面的任何布局。div.border 元素的边框被计算了

宽度，但 div.box-shadow 的阴影被浏览器忽略不计，所以借助 box-shadow 属性的这个特性，border-shadow 用来模拟元素的边框效果可以自由地使用，但必须注意其层级关系。

W3C 标准规范中描述了 box-shadow 的工作方式，直观告诉我们 box-shadow 在元素盒模型中的层次关系，如图 3-46 所示。

图 3-46 告诉我们很多信息，比如说 border-radius 圆角、阴影扩展、阴影模糊以及 padding 是如何影响对象的阴影的。非零值的 border-radius 会以相同的作用影响阴影的外形，但 border-image 不会影响对象阴影的任何外形；对象阴影同盒模型的层次一样，外阴影会在对象背景之上，内阴影会在边框之下，背景之上。所以整个层级就是：边框在内阴影之上，内阴影在背景图片之上，背景图片在背景色之上，背景色在外阴影之上。

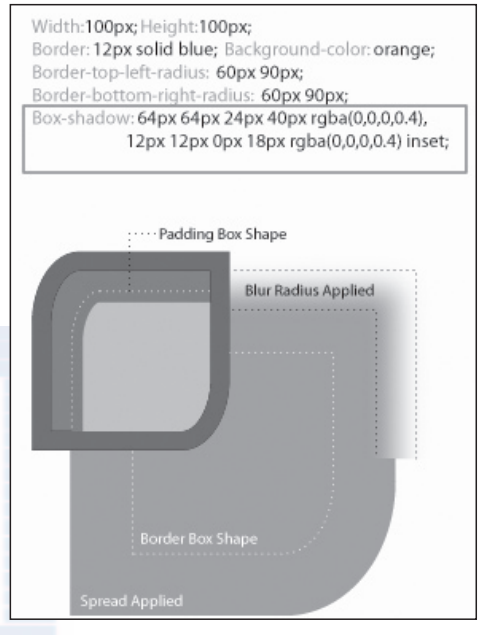


图 3-46　box-shadow 的工作方式

3. 内阴影

前几种都是外阴影的使用方法，其实使用 inset 属性值可以改变元素的阴影类型，将元素的默认外阴影重置为内阴影类型。例如：

```
.box-shadow {
      width: 200px;
      height: 100px;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-shadow: inset 3px 3px 10px #06c;
    }
```

效果如图 3-47 所示。

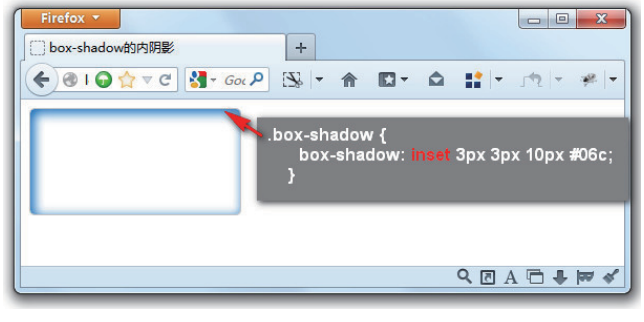
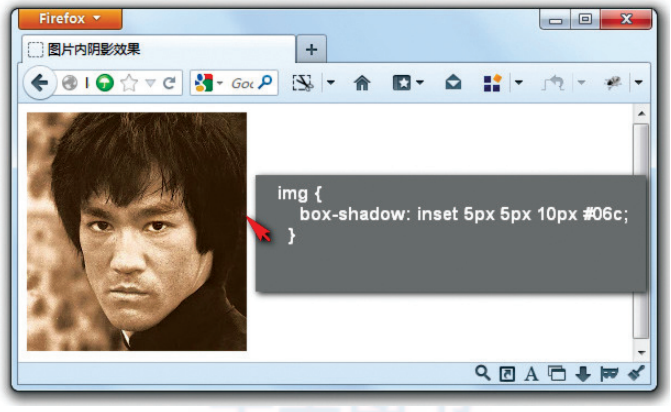


图 3-47　box-shadow 制作内阴影

不过 box-shadow 的内阴影使用在图片“img”元素上是没有任何效果的，例如：

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title> 图片内阴影效果 </title>
  <style type="text/css">
    img {
      box-shadow: inset 5px 5px 10px #06c;
    }
  </style>
</head>
<body>
  <img src="border.jpg" alt="" width="200" />
</body>
</html>
```

效果如图 3-48 所示。



☆图 3-48　box-shadow 在 img 上的内阴影无效果

图 3-48 的效果再次证实了 box-shadow 的 inset 内阴影直接运用在 img 上没有任何效果，但在实际 Web 项目中难免在图片上添加内阴影的效果。记得将 border-radius 运用在 img 上时，Webkit 内核浏览器也无效果，最后在 img 外添加一个容器标签，并将 img 转换成外容器的背景图片，将 border-radius 运用在外容器上才有圆角效果的。借助这个思路，也在 img 标签外添加一个容器，例如“div”标签，但这里不将 img 转换成 div 标签的背景，只是将 box-shadow 的内阴影使用在 div 标签上，例如：

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title> 图片内阴影效果 </title>
  <style type="text/css">
    .box-shadow {
```

```
      display: inline-block;/* 这个很重要 */
      box-shadow: inset 5px 5px 10px #06c;
    }
    img {
      position:relative;/* 这个很重要 */
      z-index: -1;/* 这个很重要 */
      vertical-align: top;
    }
  </style>
</head>
<body>
<div class="box-shadow">
  <img src="border.jpg" alt="" width="200" />
 </div>
</body>
</html>
```

此时 img 就具有内阴影效果了，如图 3-49 所示。

☆图 3-49　图片内阴影效果

也可以像 border-radius 制作图片圆角的方法，将图片转为容器 div 的背景图，也能实现图 3-49 的效果，但是会使用 JavaScript 脚本。对于不懂脚本的 Web 设计师来说，还是蛮头痛的。具体的操作方法可以参考 border-radius 一节。

4. 多层阴影

前几种都是单阴影效果的使用，其实 box-shadow 可以多层阴影同时使用，每层阴影之间使用逗号“,”隔开。而每层阴影的使用方法都和前面一样，例如：

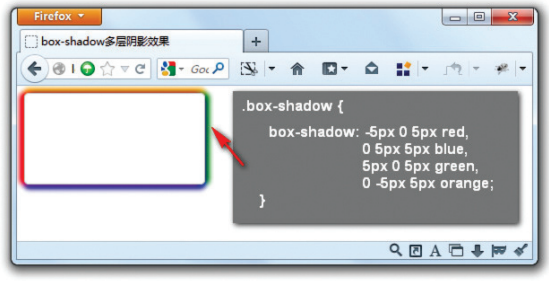
```
.box-shadow {
      width: 200px;
      height: 100px;
      border: 1px solid #ccc;
      border-radius: 5px;
```

```
    box-shadow: -5px 0 5px red, 0 5px 5px blue, 5px 0 5px green, 0 -5px 5px orange;
}
```

效果如图 3-50 所示。

制作多层阴影效果时，不设置模糊半径，只设置扩展半径，并配合多个阴影颜色，还可以制作多色边框效果，代替 border-color 属性制作多色边框效果，例如：

```
.box-shadow {
    width: 200px;
    height: 100px;
    border: 1px solid #ccc;
    margin: 30px;
    box-shadow: 0 0 0 1px red,
                0 0 0 5px blue,
                0 0 0 8px green,
                0 0 0 12px yellow,
                0 0 0 16px orange,
                0 0 0 20px #06c,
                0 0 0 24px lime;
}
```



☆图 3-50　box-shadow 多层阴影效果

效果如图 3-51 所示。

使用 box-shadow 制作多色边框效果，需要注意模仿 border 的宽度。前面介绍过 box-shadow 的工作模式，在计算宽度时需要减去前面阴影的值，才是显示的颜色宽度。

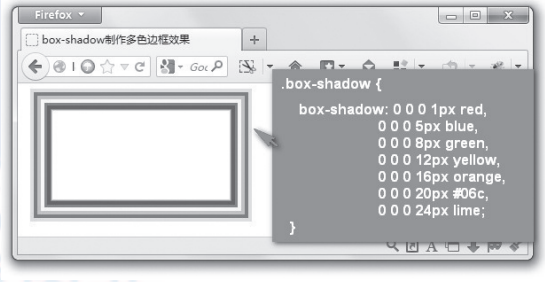


图 3-51　box-shadow 制作多色边框效果

在使用多层级 box-shadow 时，还需要特别注意阴影的顺序，最先写的阴影将显示在最顶层，如上面的示例，先定义 1px 红色阴影，再定义 5px 蓝色阴影，接着是 8px 绿色阴影，以此类推。显示结果就是红色在蓝色上面，蓝色在绿色上面，绿色在黄色上面，以此类推。但是，如果最前面的阴影太大，顶层的阴影就会遮盖底部的阴影。例如，上例中将最底层的 24px 的 lime 阴影放到最前面，效果就完全不一样了。

```
.box-shadow {
    width: 200px;
    height: 100px;
    border: 1px solid #ccc;
    margin: 30px;
    box-shadow: 0 0 0 24px lime,
                0 0 0 1px red,
                0 0 0 5px blue,
                0 0 0 8px green,
                0 0 0 12px yellow,
                0 0 0 16px orange,
```

```
                0 0 0 20px #06c;
}
```

此时后面的阴影都被第一个阴影遮盖了，如图 3-52 所示。

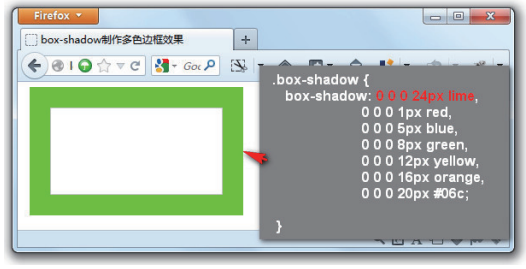


图 3-52　被遮盖的阴影效果

3.5.3　浏览器兼容性

目前 box-shadow 属性得到很好的支持，IE 8 及以前版本的浏览器不支持 box-shadow 属性。在现代浏览器的新版本中无须加各浏览器的前缀，不过要向前兼容，Firefox 3.5 ～ 3.6 下需要添加“-moz-”，Chrome 4 ～ 9 和 Safari 3.1 ～ 5.0 浏览器中添加“-webkit”。借助兼容方式，各主流浏览器对 box-shadow 属性的支持情况如表 3-7 所示。

表 3-7　box-shadow 的浏览器兼容表

属性名					
box-shadow	9 + √	3.5 + √	2.0 + √	10.5 + √	4.0 + √

虽然 IE 低版本不支持这个属性，但目前 box-shadow 在实际项目中运用越来越普遍。因为 box-shadow 实现阴影比使用背景图片的方法方便，同时能为 Web 前端设计师减少很多时间，维护也方便。

要兼容 IE 低版本，可以使用 IE 的滤镜来模拟实现。

```
filter: progid:DXImageTransform.Microsoft.Shadow(color='颜色值',
        Direction=阴影角度（数值）, Strength=阴影半径（数值））;
```

其中“DropShadow”（盒状阴影）和“Shadow”（阴影）两个滤镜正是为实现阴影而设，另外“Glow”（发光）滤镜则用于在盒容器四周实现发光阴影。但这些滤镜可设置的参数并不像 box-shadow 属性那样提供诸多的自定义参数，我不认为这些滤镜能够实现前面示例中所需的效果。当然除了 IE 滤镜之外，同样可以采用前面说的 PIE 和 IE -CSS3 脚本来实现 IE 下的阴影效果。

现代浏览器使用 box-shadow 来制作阴影，而不支持 box-shadow 的浏览器让它不显示阴影，如果非要完美兼容，不妨考虑在不支持 box-shadow 的浏览器中使用背景图片来模仿阴影。

3.5.4 box-shadow 属性的优势

从实现盒子阴影来说，box-shadow 是最方便的，不管是使用背景图片，还是使用滤镜或者说 JavaScript 脚本，都无法与 box-shadow 属性相比。

- box-shadow 具有多个属性参数可选，能制作出圆润平滑的阴影效果。
- 代码维护方便，可以随时更改参数来实现效果的更新。

3.5.5 实战体验：制作 3D 搜索表单

为了方便读者理解，接下来介绍一个 box-shadow 案例——制作 3D 搜索表单。

在这个案例中，除了使用 box-shadow 之外，还使用 border-radius 制作圆角，并涉及 text-shadow 制作文本阴影，以及 gradient 制作渐变背景图片，关于这两项技术，请参阅后面章节。整个案例的效果如图 3-53 所示。

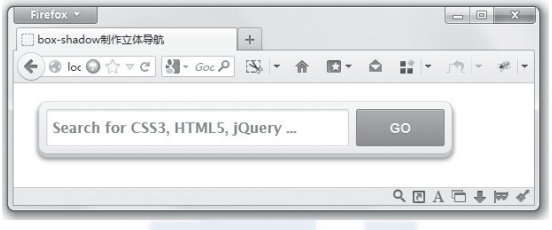


图 3-53　Box-shadow 制作 3D 搜索表单

从 box-shadow 多层级阴影特性出发，给表单容器设置多个同方向阴影效果，并且配合圆角属性 border-radius 来描绘圆角线框，同时使用渐变属性制作渐变的背景图片等，结合 CSS3 的多种效果，从而构建出这个 3D 搜索表单。

1. 构建 3D 表单的结构

整个表单结构很简单，代码如下所示。

```
<!—表单结构 -->
<form id="formWrapper">
      <div class="formFiled clearfix">
<!-- 搜索表单的输入框 -->
<input type="text" required="" placeholder="Search for CSS3, HTML5, jQuery ..."
class="search">
<!-- 搜索按钮 -->
<input type="submit" class="btn submit" value="go">
       </div>
   </form>
```

整个结构使用一个“ form”元素，并且应用一个“ div.formFiled”容器来包裹“ input.search”的输入框和一个搜索按钮“input.btn”，如图 3-54 所示。

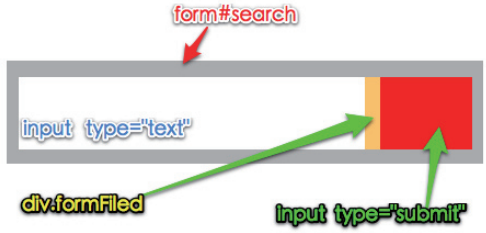


图 3-54　搜索表单结构

2. 设计表单容器的 3D 立体效果

使用 box-shadow 的多层阴影特性给表单元素设计 3D 立体效果，样式代码如下所示。

```
#formWrapper {
      width: 450px;/* 设置搜索表单的宽度 */
      padding: 8px;
      margin: 20px;
      overflow: hidden;/* 清除浮动 */
      /* 设置表单的边框效果 */
      border-width: 1px;
      border-style: solid;
      border-color: #dedede #bababa #aaa #bababa;
      /* 最为关键的代码，设置表单 3D 立体效果 */
      box-shadow: 0 3px 3px rgba(255,255,255,.1),
                  0 3px 0 #bbb, 0 4px 0 #aaa,
                  0 5px 5px #444;
      /* 设置圆角效果 */
  border-radius: 10px;
     /* 使用渐变制作表单的渐变背景图片 */
     background-color: #f6f6f6;
     background-image: -webkit-gradient(linear, left top,
                       left bottom, from(#f6f6f6), to(#eae8e8));
     background-image: -webkit-linear-gradient(top, #f6f6f6, #eae8e8);
     background-image: -moz-linear-gradient(top, #f6f6f6, #eae8e8);
     background-image: -ms-linear-gradient(top, #f6f6f6, #eae8e8);
     background-image: -o-linear-gradient(top, #f6f6f6, #eae8e8);
     background-image: linear-gradient(top, #f6f6f6, #eae8e8);
     }
```

3. 制作表单输入框的搜索按钮效果

接下来使用 box-shadow 制作 3D 立体效果，为了使表单更漂亮，将输入框和搜索按钮进行美化，代码如下所示。

```
/* 输入框样式效果 */
#formWrapper .search {
      width: 330px;
      height: 20px;
      padding: 10px 5px;
```

```
    float: left;
    font: bold 16px 'lucida sans', 'trebuchet MS', 'Tahoma';
    border: 1px solid #ccc;
    box-shadow: 0 1px 1px #ddd inset, 0 1px 0 #fff;/* 多阴影效果 */
    border-radius: 3px;
    }
/* 输入框得到焦点时样式 */
    #formWrapper .search:focus {
    outline: 0;
    border-color: #aaa;
    box-shadow: 0 1px 1px #bbb inset;
    }
    #formWrapper .search::-webkit-input-placeholder,
    #formWrapper .search:-moz-placeholder,
    #formWrapper .search:-ms-input-placeholder {
      color: #999;
      font-weight: normal;
    }
    /* 搜索按钮效果 */
    #formWrapper .btn {
    float: right;
    border: 1px solid #00748f;
    height: 42px;
    width: 100px;
    padding: 0;
    cursor: pointer;
    font: bold 15px Arial, Helvetica;
    color: #fafafa;
    text-transform: uppercase;
    background-color: #0483a0;
    background-image: -webkit-gradient(linear, left top,
                      left bottom, from(#31b2c3), to(#0483a0));
    background-image: -webkit-linear-gradient(top, #31b2c3, #0483a0);
    background-image: -moz-linear-gradient(top, #31b2c3, #0483a0);
    background-image: -ms-linear-gradient(top, #31b2c3, #0483a0);
    background-image: -o-linear-gradient(top, #31b2c3, #0483a0);
    background-image: linear-gradient(top, #31b2c3, #0483a0);
    border-radius: 3px;
    text-shadow: 0 1px 0 rgba(0, 0 ,0, .3);
    box-shadow: 0 1px 0 rgba(255, 255, 255, 0.3) inset, 0 1px 0 #fff;
    }
/* 按钮悬浮状态和焦点状态下效果 */
      #formWrapper .btn:hover,
      #formWrapper .btn:focus {
    background-color: #31b2c3;
    background-image: -webkit-gradient(linear, left top,
                left bottom, from(#0483a0), to(#31b2c3));
    background-image: -webkit-linear-gradient(top, #0483a0, #31b2c3);
    background-image: -moz-linear-gradient(top, #0483a0, #31b2c3);
    background-image: -ms-linear-gradient(top, #0483a0, #31b2c3);
    background-image: -o-linear-gradient(top, #0483a0, #31b2c3);
```

```
    background-image: linear-gradient(top, #0483a0, #31b2c3);
    }
    /* 按钮点击时效果 */
      #formWrapper .btn:active {
    outline: 0;
    box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
    }
/*firefox 下按钮去除焦点线 */
     #formWrapper::-moz-focus-inner {
    border: 0;
    }
```

制作的 3D 立体搜索表单效果如图 3-53 所示。当然这只是一个简单的案例，box-shadow 还可以制作更多的效果，例如双层边框效果、发光效果、立体按钮等。大家还可以发挥自己的想象力，创造出更多的有创意的、有吸引力的 UI 效果。

3.6　本章小结

本章主要介绍 CSS3 新增的边框特性，首先从 CSS 的 border 属性着手切入，分别介绍了 CSS3 新增边框特性，border-color、border-image、border-radius 以及 box-shadow。详细介绍了每个特性的语法规则，并且结合一些简单的案例，以图解的方式介绍了这些特性的具体使用方法以及在 IE 下相应的兼容和处理方式。