HZ BOOKS
华章科技

内容全面且深入，系统讲解Bootstrap的各项功能、组件、插件和扩展，深度解析Bootstrap的内核源代码

实战性强，不仅为各个知识点精心设计了辅助说明问题的小案例，而且还包含一个综合性的大案例

實戰

成林 著

Bootstrap in Action

Bootstrap 实战

机械工业出版社
China Machine Press

實戰

51CTO.com 技术成就梦想 联合策划

Bootstrap是一款简洁、直观、强悍、移动设备优先的前端开发框架，目标是让Web开发变得更迅速、更简单。自Twitter将Bootstrap开源以后，其在开源社区得到了极为广泛的关注，曾经很长一段时间内都是GitHub上最受欢迎的开源项目。Bootstrap有很多优越的特性，比如12列栅格布局、支持HTML 5标签和语法、支持响应式设计、可定制jQuery内置插件、跨设备和跨浏览器等。

如今国内外有很多流行的网站都在使用Bootstrap，但是相关的资料和图书却很少， 本书应该能算得上是最有价值的参考资料了。它根据Bootstrap的最新技术撰写，不仅全面讲解了Bootstrap的各项功能及其用法，详细讲解了它的组件、插件和扩展的相关知识，而且还分析了它的内核源代码，几乎面面俱到，广度和深度兼备。

本书主要包含以下内容：

- Bootstrap的构成组件、功能特色、开发工具、应用情况，以及它的下载和应用解析；
- Bootstrap框架解析，包含全局样式表、栅格系统、布局、响应式设计等内容；
- CSS样式优化，包含页面排版优化、表格设计优化、表单设计优化、按钮设计，以及图片和图标的设计等内容；
- 动态CSS 样式LESS的特性和使用；
- 各种Bootstrap组件的特性和使用；
- 各种Bootstrap插件的特性和使用；
- Bootstrap的扩展；
- 利用Bootstrap开发一个社区分享网站的全部流程和方法；
- Bootstrap内核源代码分析；

……

客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259
投稿热线：(010) 88379604

数字阅读：www.hzmedia.com.cn
华章网站：www.hzbook.com
网上购书：www.china-pub.com

上架指导：计算机/程序设计/Web开发
ISBN 978-7-111-44432-9
9 787111 444329 >
定价：69.00元

Bootstrap in Action

Bootstrap实战

成林 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Bootstrap 实战 / 成林著 . —北京：机械工业出版社，2013.11

ISBN 978-7-111-44432-9

I. B…　II. 成…　III. 网页制作工具　IV. TP393.092

中国版本图书馆 CIP 数据核字（2013）第 246818 号

本书由国内资深前端工程师撰写，是目前内容最为全面和深入的 Bootstrap 专著。它不仅系统讲解了 Bootstrap 的各项功能和使用方法，详细讲解了 Bootstrap 的组件、插件和扩展技术，而且深度解析了 Bootstrap 的内核源代码。本书实战性强，为各个知识点都精心设计了辅助说明问题的小案例，最后还包含一个综合性的大案例，不仅能满足读者系统学习理论知识的需求，还能满足读者充分实践的需求。

全书一共 10 章：第 1 章和第 2 章介绍了 Bootstrap 的构成组件、功能特色、开发工具、应用情况，以及它的下载和应用解析；第 3 章对 Bootstrap 框架进行了解析，包含全局样式表、栅格系统、布局、响应式设计等内容；第 4 章讲解了 CSS 样式的优化，包含页面排版优化、表格设计优化、表单设计优化、按钮设计，以及图片和图标的设计等内容；第 5~8 章则分别讲解了 CSS 动态样式 LESS 的特性和使用、各种 Bootstrap 组件的特性和使用、各种 Bootstrap 插件的特性和使用、Bootstrap 的扩展等内容；第 9 章逐步讲解了利用 Bootstrap 开发一个社区分享网站的全部流程和方法，是一个综合性的大案例；第 10 章则对 Bootstrap 的内核源代码进行了深入的分析。

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：罗词亮

三河市杨庄长鸣印刷装订厂印刷

2013 年 11 月第 1 版第 1 次印刷

186mm × 240mm · 23 印张

标准书号：ISBN 978-7-111-44432-9

定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

为什么要写这本书

Boostrap 是 Twitter 公司开发的一个基于 HTML、CSS 和 JavaScript 的技术框架，集合了最新的前端技术，为实现 Web 应用程序快速开发提供了一套前端工具包，包括布局、栅格、表格、按钮、表单、导航、提示等。使用 Bootstrap 可以构建出非常优雅的前端界面，且占用资源非常少。

Bootstrap 是目前非常流行的前端开发工具包，具有以下特色：

- 由业内权威专家开发。Twitter 是互联网领域的技术先驱，引领时代技术潮流，Twitter 前端开发团队是公认的最棒的团队之一，整个 Bootstrap 项目由经验丰富的工程师和设计师奉献。
- 拥有庞大的用户基础和实践基础，值得信赖。
- 学习和应用门槛比较低，适合各种技术水平的人。
- 跨设备、跨浏览器，为移动开发提供了平稳的开发平台。
- 提供 12 列栅格布局，使网页布局变得很简单。
- 支持响应式设计，满足不同设备的显示需求。
- 样式化的文档，使学习和参考变得直观、方便。
- 开放式代码库，使 Bootstrap 成为众多业内好手展示的舞台，也极大丰富了 Bootstrap 代码库。
- 与 jQuery 完美融合，可定制 jQuery 插件。
- 选用 LESS 构建动态样式，使动态样式开发成为可能，降低了时间成本，却可以编写更快、更灵活的 CSS 样式表。
- 支持 HTML5 标签和语法，要求在 HTML5 文档类型基础上进行设计和开发。
- 支持 CSS3 所有属性和标准，逐步改进组件以达到最终效果。

Bootstrap 始于 2011 年 8 月，至今才刚满两年，但是 Bootstrap 旋风却已刮遍了整个互联

网。各种较小的网站就不提了，国内外很多较有名的网站也采用了 Bootstrap。对于设计能力不强也没有太多时间去设计前端界面的用户来说，Bootstrap 价值巨大。Bootstrap 的目的就是帮助开发人员快速开发原型，避免经常从零开始绘制白底黑边的裸图。

Bootstrap 框架提供一级的视觉效果，且应用视觉效果是一致的，这一点其实是很难实现的。使用 Bootstrap 可以确保整个 Web 应用程序的风格完全一致，用户体验一致，操作习惯一致。如果希望整个网站的链接、按钮、提醒都有统一的视觉效果，那就应该毫不犹豫地选择 Bootstrap，此外它还可以对不同级别的提醒使用不同的颜色。

快速应用，简单而优雅，Bootstrap 会让 Web 应用程序看起来与 Windows 或 GNOME 下的程序一样，按钮一样，对话框一样，运行快速。越来越多的 Web 应用程序被直接放在桌面上运行，应用的一致性是趋势，开发人员可以把精力放在业务上，而不是 UI 设计上。

本书主要内容和特色

本书系统讲解 Bootstrap 技术的体系结构、基础知识、基本用法，以及各种深度应用。它不是一本语法书，也不是技术全能书，不会告诉读者怎么编写 HTML、CSS 和 JavaScript 代码，但它会告诉读者如何驾驭 Bootstrap，让 Bootstrap 成为你的设计宝典、你的前端开发基地。在此基础上，读者可以拓展个人插件，让设计成为一种分享和积累。

简单来说，本书具有如下几个特点（也许这些特点并不都适合你，但是只要满足一条或几条，那么阅读本书或者把本书作为参考都是很合适的）：

- 快速上手。Bootstrap 是一个比较复杂的前端技术框架，对于基础薄弱的读者来说，初步使用 Bootstrap 会面临很多困难和障碍。本书从下载 Bootstrap 框架开始，手把手地说明和演示，帮助读者快速上手，旨在教会读者正确使用 Bootstrap，并应用 Bootstrap 所提供的全部功能。
- 扎实全面。Bootstrap 是 HTML、CSS 和 JavaScript 技术框架，使用时应具备一定的基础才能够活学活用。本书知识点和技能点兼顾，系统而全面，确保读者在无基础的前提下能够轻松阅读、顺畅理解、心领神会。
- 深入解码。本书对于 Bootstrap 的剖析不仅关注知识面的系统，更强调技术的深度，特别是 Bootstrap 插件技术的开发；不仅对 Bootstrap 内部插件进行深入讲解和剖析，还对插件的设计原理、途径和方法进行解密，以帮助读者用好 Bootstrap 插件，开发自己的 Bootstrap 插件。

下面就本书主要内容进行简单说明和梳理。

第 1 章　为什么要学习 Bootstrap

该章从知识角度重点介绍 Bootstrap 框架的前世今生，以及为什么要学习 Bootstrap，如何开始学习等。

第 2 章　使用 Bootstrap 的准备

该章重点介绍如何使用 Bootstrap，从下载 Bootstrap 框架、了解 Bootstrap 工具包类型和

内部构成，到如何在页面中正确使用 Bootstrap，手把手帮你实现第一个 Bootstrap 开发示例。

第 3 章　Bootstrap 框架解析

Bootstrap 框架主要由动态 CSS 语言编写，经过 Node.js 编译后，Bootstrap 就是众多 CSS 的合集。该章就 Bootstrap 框架进行探索，以期帮助读者掌握该框架的基本实现方式和设计思路，主要包括全局样式表、栅格系统、Bootstrap 布局、响应式设计等核心内容。

第 4 章　优化 CSS 样式

CSS 本身没有可优化的，但是浏览器解析的默认标签样式却不敢恭维。Bootstrap 对常用标签样式进行优化，使其更耐看，更精致。Bootstrap 的基础 CSS（Base CSS）提供了优雅、一致的多种基础 HTML 页面要素，包括排版、表格、表单、按钮等，能够满足前端工程师的基本设计需求。

第 5 章　CSS 动态样式——LESS

LESS 是一种半自动化的动态语言，它使 CSS 具备了初步编程的能力，在 CSS 的语法基础之上，引入了变量、混合、运算和函数等特性，大大提升了 CSS 动态开发能力，降低了 CSS 的维护成本。就像它的名称所表达的意思，LESS 可以让我们用更少的代码做更多的事情。

第 6 章　使用 Bootstrap 组件

Bootstrap 作为完整的前端工具集，内建了大量强大、优雅且可重用的组件。该章详细介绍按钮（button）、导航（navigation）、标签（label）、徽章（badge）、排版（typography）、缩略图（thumbnail）、提醒（alert）、进度条（progress bar）、杂项（miscellaneous）组件的结构和基本应用。

第 7 章　使用 Bootstrap 插件

Bootstrap 自带了 13 个 JavaScript 插件，这些插件为 Bootstrap 组件赋予了生命，因此用户在学习使用 Bootstrap 组件的同时，还必须学习使用 Bootstrap 插件。该章将详细介绍这些插件的使用技法。

第 8 章　Bootstrap 扩展

虽然 Bootstrap 自带很多 JavaScript 插件，但是一些常用的控件却没有，如 Datepicker 等。该章将介绍几款比较流行的 Bootstrap 扩展插件，帮助读者认识如何设计和使用 Bootstrap 外部插件，以弥补 Bootstrap 的不足之处。

第 9 章　使用 Bootstrap 快速开发社区分享网站

该章中的示例构建的主题是一个基本单词分享的学习型网站，旨在帮助读者学习如何用 Bootstrap 制作自己的响应式社区网站。Bootstrap 是一个响应式框架，用它创建一个响应式 Web 应用程序将是一个伟大的起点。

第 10 章　Bootstrap 内核解码

该章从 Bootstrap 应用阶段上升到源码分析阶段，帮助读者掌握 Bootstrap 设计原理，为

Bootstrap 二次开发打好基础。

读者对象

本书适合以下读者阅读：

- 打算学习 DIV+CSS 的设计人员。
- 打算学习 JavaScript 的开发人员。
- 有意提升自己网站设计水平和 Web 应用程序开发能力的 Web 开发人员。
- 希望全面深入理解 Bootstrap 框架的初学者。
- 没有 UI 设计经验的程序员。
- 希望快速搭建界面的网页设计人员。

此外，本书也适合熟悉下列相关技术的读者阅读：

- PHP/ASP/JSP
- jQuery
- AJAX
- HTML/XML
- CSS
- JavaScript

对于没有计算机基础知识的初学者，以及只想快速搭建网站 UI 和交互功能的读者，阅读本书前建议先阅读 HTML、CSS 和 JavaScript 基础教程类图书。

如何阅读本书

“工欲善其事，必先利其器”，在“善其事”之前，要先检查“器”是否已经磨得足够锋利，是否能够在前进的道路上披荆斩棘。无论将来的职业发展方向是架构师、设计师、分析师、管理者，还是其他职位，只要与 Web 设计打交道，就有必要打好技术基础。本书涉及的是核心的 Bootstrap 框架技术，如果能全部理解并付诸实践，一定可以提升读者的 Web 设计和开发水平。

要用好本书，读者应当准备一些 HTML 基础知识，如果熟悉 CSS 样式，将更容易理解一些设计规则。由于 Bootstrap 是一个前端综合技术框架，因而如果读者希望系统掌握 Bootstrap，并开发自己的 Bootstrap 插件，则应该掌握 JavaScript 语言基础；如果仅仅是为了用好 Bootstrap，则可以不考虑。

资源和勘误

本书编写过程中得到了以下人员的帮助：马本连、吴建华、江淑军、李斌、李经键、郑

伟、田蜜、陆颖、王慧明、张炜、陈锐、王幼平、杨龙贵、苏震巍、崔鹏飞等，非常感谢他们。由于作者的水平有限，加之编写时间仓促，书中难免会出现一些错误或不准确的地方，恳请读者批评指正。书中的全部源文件可以从华章网站（www.hzbook.com）下载。如果您有任何意见和建议，欢迎发送邮件至邮箱 js_code@126.com，期待得到您的真诚反馈。

致谢

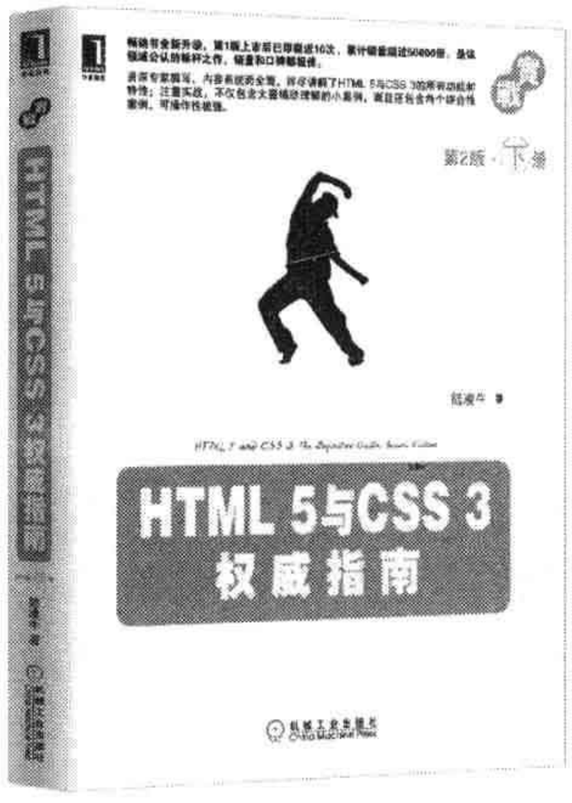
感谢机械工业出版社华章公司策划编辑杨福川在这一年多的时间中始终支持我的写作，他的鼓励和帮助让我顺利完成了本书。

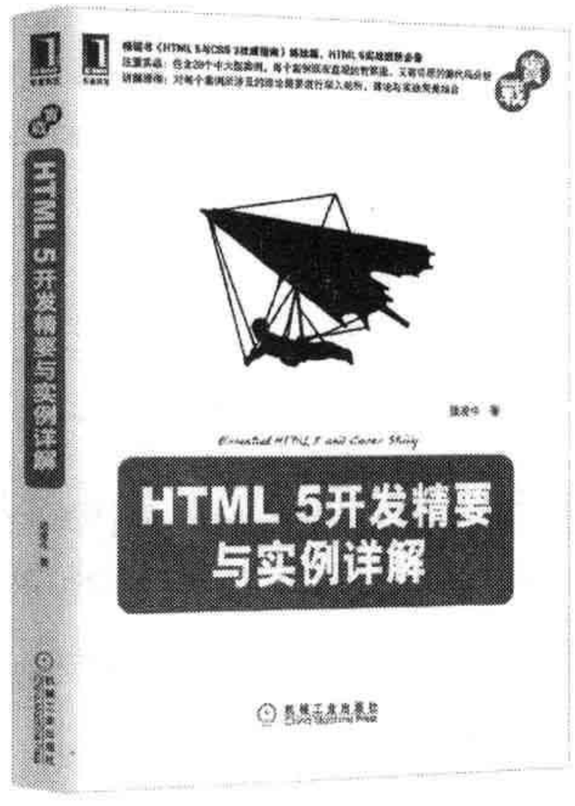
最后感谢我的父母，感谢他们的养育之恩，感谢他们时时刻刻给我信心和力量！

谨以此书献给我最爱的家人，以及众多热爱网页设计的朋友们！

成林

推荐阅读





HTML 5与CSS 3权威指南 上下册

第1版2年内印刷近10次，累计销量超过50000册，4大网上书店的读者评论超过4600条，98%以上的评论都是五星级的好评。不仅是HTML 5与CSS 3图书领域当之无愧的领头羊，而且在整个原创计算机图书领域也是佼佼者。本书已经成为HTML 5与CSS 3图书领域的一个标杆，被读者誉为“系统学习HTML 5与CSS 3技术的最佳指导参考书之一”和“Web前端工程师案头必备图书之一”。第2版首先从技术的角度结合最新的HTML 5和CSS 3标准对内容进行了更新和补充，其次从结构组织和写作方式的角度对原有的内容进行了进一步优化，使之更具价值和更便于读者阅读。

全书共29章，本书分为上下两册：上册（1~17章）全面系统地讲解了HTML 5相关的技术，以HTML 5对现有Web应用产生的变革开篇，顺序讲解了HTML 5与HTML 4的区别、HTML 5的结构、表单元素、HTML编辑API、图形绘制、History API、本地存储、离线应用、文件API、通信API、扩展的XML HttpRequest API、Web Workers、地理位置信息、多媒体相关的API、页面显示相关的API、拖放API与通知API等内容；下册（19~29章）全面系统地讲解了CSS 3相关的技术，以CSS 3的功能和模块结构开篇，顺序讲解了各种选择器及其使用、文字与字体的相关样式、盒相关样式、背景与边框相关样式、布局相关样式、变形处理、动画、颜色相关样式等内容。全书一共351个示例页面，所有代码均通过作者上机调试。下册的最后有2个综合案例，以迭代的方式详细讲解了整个案例的实现过程，可操作性极强。

HTML 5开发精要与实例详解

这是一本以综合性案例为导向并辅之以精要知识点讲解的HTML 5实战教程。内容分为两大部分：第一部分通过一系列中大型案例全方位对HTML 5的各个重要知识点进行了详细的讲解，每个案例包含案例概述、页面效果展示、案例所涉及主要知识点（精要）、源代码剖析4个部分，读者既能根据书中的步骤动手实践，又能重点学习案例中用到的核心理论知识，同时还能领会源代码的设计思路和方法；第二部分讲解了jWebSocket、RGraph、WebGL等3个重要框架和技术的详细使用方法。

全书一共12章：第1章分别用2个案例演示了如何利用HTML 5中的结构元素来构建一个博客网站和企业门户网站；第2章用2个案例讲解了表单在HTML 5中的使用；第3章用6个案例讲解了如何利用Canvas元素来绘制图形、图像和制作动画；第4章用2个案例介绍了文件APT和拖放API的使用方法；第5章用4个案例讲解了如何打造自己的网页视频播放器、网页音频播放器，以及实现视频实时回放和视频截图等多媒体功能；第6章用6个案例全面讲解了HTML 5中的本地存储技术；第7章用单点登录和获取批量数据这2个案例讲解了HTML 5中的跨文档的消息传输技术；第8章用2个案例讲解了如何利用Web Workers实现多线程处理；第9章用1个案例讲解了如何利用Geolocation API来获取地理位置信息；第10~13章分别讲解了Socket通信框架jWebSocket、统计图制作插件RGraph、三维Web开发技术WebGL的详细使用方法，并辅之以丰富的案例。

本书所有案例的源代码都是作者亲自编写并调试和运行成功的。读者可以利用这些代码进行实战练习，也可以根据需要对这些代码进行修改，以观察不同的效果，从而加深对案例代码和书中知识点的理解。

目　录

第 1 章

为什么要学习 Bootstrap

本章内容

- Bootstrap 概述
- Bootstrap 功能介绍
- Bootstrap 应用项目赏析
- Bootstrap 版本变化
- Bootstrap 开发工具和参考资源

Boostrap 是 Twitter 公司（www.twitter.com）开发的一个基于 HTML、CSS、JavaScript 的技术框架，符合 HTML、CSS 规范，且代码简洁、视觉优美。该框架设计时尚、直观、强大，可用于快速、简单地构建网页或网站。

Bootstrap 集合 CSS、HTML 和 JavaScript，使用了最新的浏览器技术，为实现快速开发提供了一套前端工具包，包括布局、栅格、表格、按钮、表单、导航和提示等。使用 Bootstrap 不仅可以构建出非常优雅的前端界面，而且占用资源非常少，使用 gzip 压缩后大小仅有 10KB。由于 Firefox、Chrome 和 Safari 等主流浏览器对 W3C 标准有着较好的支持，因此 Bootstrap 在网页跨浏览器兼容方面也有相当不错的表现。

1.1 Bootstrap 概述

Web 开发人员每天都需要与 HTML、CSS、JavaScript 打交道，然而不少人在大部分时间里都在周而复始地写模板、样式和交互效果，并没有想过如何将这些重复的工作整合在一起。Twitter 公司推出的 Bootstrap 能够帮助开发人员摆脱这种重复性的工作。下面来了解下 Bootstrap 是一项什么技术，它的来龙去脉，以及它能够帮助用户做些什么。

1.1.1 Bootstrap 的历史

早期 Twitter 提供的内部开发工具缺乏一些精致和平易近人的设计，为了应对复杂的设计需求，Twitter 前端工程师在开发网站时喜欢采用所有自己熟悉的技术，这就造成了网站维护困难、可扩展性不强、开发成本高等问题。

2010 年 6 月，为了提高内部的协调性和工作效率，Twitter 公司的几个前端开发人员自发成立了一个兴趣小组，小组早期主要围绕一些具体产品展开讨论。在不断的讨论和实践中，小组逐渐确立了一个清晰的目标，期望设计一个伟大的产品，即创建一个统一的工具包，允许任何人在 Twitter 内部使用它，并不断对其进行完善和超越。后来，这个工具包逐步演化为一个有助于建立新项目的应用系统。在它的基础上，Bootstrap 的构想产生了。

整个项目由 Mark Otto 和 Jacob Thornton 主导建立，定位为一个开放源码的前端工具包，旨在帮助设计师和开发人员快速、高效地构建通用的、最棒的东西。

他们希望通过这个工具包提供一种精致的、经典的、得到充分认可的，且使用 HTML、CSS 和 JavaScript 构建的组件，为用户建立和创建灵活的设计和丰富的插件库。

最终，Bootstrap 成为应对这些挑战的解决方案，并开始在 Twitter 内部迅速成长，被 twitter.com 广泛采用，形成了稳定版本。随着工程师对其不断的开发和完善，Bootstrap 进步显著，不仅包括基本样式，而且有了更为优雅和持久的前端设计模式。2011 年 8 月，Twitter 将其开源，开源页面地址为：http://twitter.github.com/bootstrap。

今天，Bootstrap 已发展到包括几十个组件，并已成为最流行的项目，截至 2013 年 5 月 1 日，在 GitHub（https://github.com/twbs/bootstrap/）上有超过 49554 个加星（Star）和 15213 个分支（Fork），如图 1-1 所示。当然，这个数字还在不断变化。

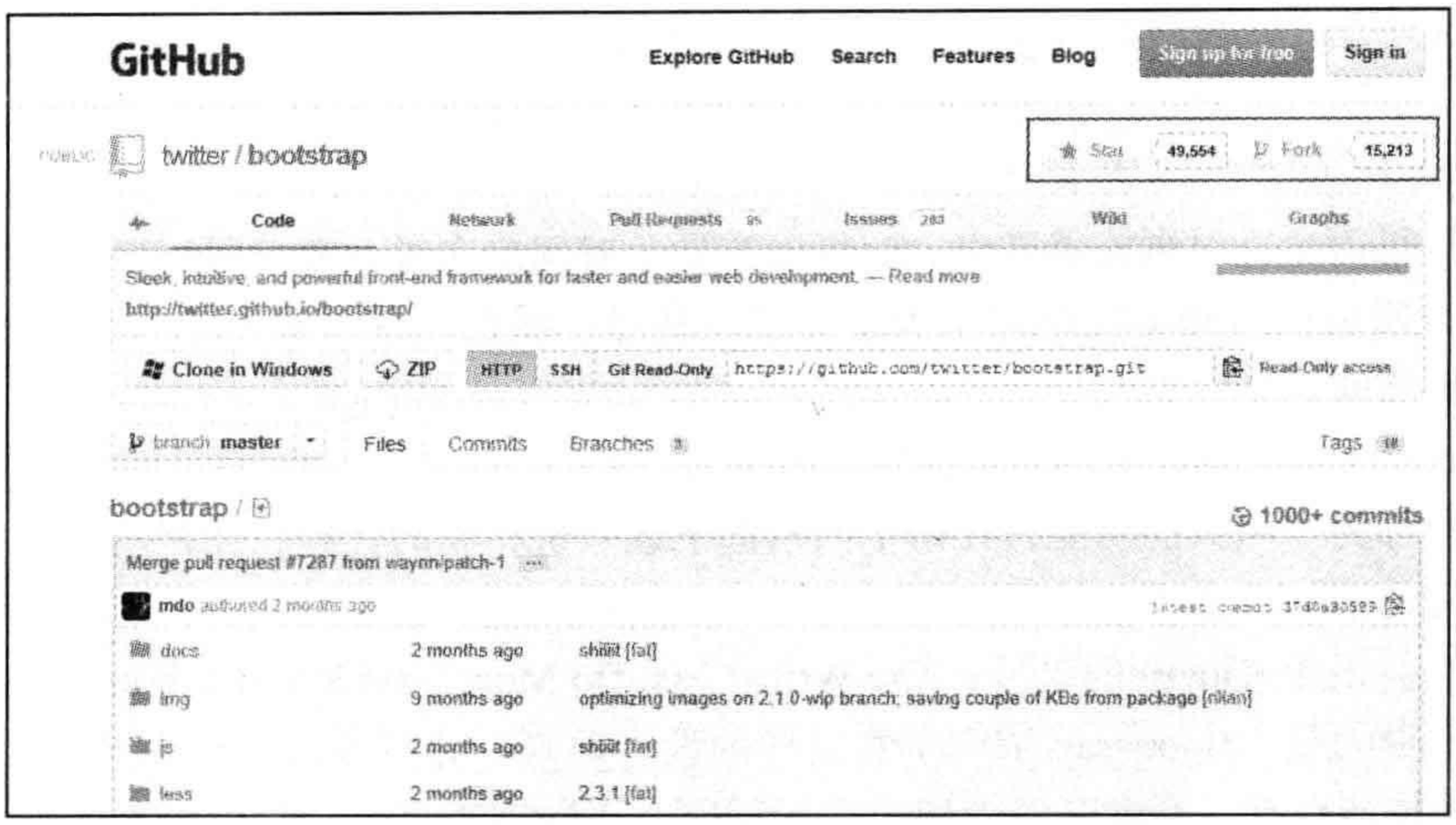


图 1-1　GitHub 开源页面

Bootstrap 的设计者、著名前端工程师 Mark Otto 这样写道：“Bootstrap 是我和 Jacob Thornton 编写的一个前端工具包，目的是帮助设计师和 Web 前端开发人员快速有效地创建结构简单、性能优良、页面精致的 Web 应用程序。它使用了最新的浏览器技术，可以提供精致的网页排版方式以及表单、按钮、表格、网格栅格化、导航等诸多元素。”

Bootstrap 的内置样式继承了 Mark Otto 简洁亮丽的设计风格，任何开发团队都能使用它提供的 HTML 模板、CSS 样式和 jQuery 组件来布署，或者重建外观漂亮的 Web 应用程序。

Bootstrap 虽然发布时间不长，但是已经非常成熟，并得到广泛推广，目前最新版本是 3.0，包括完整的 CSS 编译和非编译版本、样例模板和 JavaScript 插件。

1.1.2　选择 Bootstrap 的理由

Bootstrap 所涉及的应用范围到底有多广？ Bootstrap 是否真的很实用？关于这两方面的问题，网络上曾展开过激烈的争论。Bootstrap 对于设计能力不强，也没有太多的时间去设计前端界面的用户来说，价值是巨大的。Bootstrap 的目的是帮助开发人员快速开发原型，避免用户经常从零开始绘制白底黑边的裸图。

Bootstrap 包括几十个组件，每个组件都自然地结合了设计与开发，具有完整的实例文档，定义了真正的组件和模板。无论处在何种技术水平，也无论处在哪个工作流程中的开发者，都可以使用 Bootstrap 快速、方便地构建自己喜欢的应用程序。

Bootstrap 引入了 12 栏栅格结构的布局理念，使设计质量高、风格统一的网页变得十分容易。它包含了 HTML、CSS 和 JavaScript 三大主要部分，各部分简单说明如下。

- Bootstrap 的 HTML 是基于 HTML5 的最新前沿技术，它不同于古老陈旧的其他网页标准，灵活高效，简洁流畅。它摒弃了那些复杂而毫无意义的标签，引入了全新的 <header>、<section>、<footer>、<article>、<video> 和 <canvas> 等标签，使网页的语义性

大大增加，从此网页不再是供机器阅读的枯燥文字，而是可供人类欣赏的优美作品。在网页中插入多媒体，也因有了 <video> 和 <canvas> 而再也不需借助腐朽的 Flash 控件。

- Bootstrap 的 CSS 是使用 LESS 创建的 CSS，是新一代的动态 CSS。对设计师来说，能写得更少；对浏览器来说，解析更容易；对用户来说，阅读更轻松。直接用自然书写的四则算术和英文单词来表示宽度、高度、颜色，使得写 CSS 不再是高手才会的神秘技能。
- Bootstrap 的 JavaScript 是使用 jQuery 的 CSS，是优秀的 JavaScript，它不会使每个用户都为了相似的功能，在每个网站都下载一份相同的代码，而是用一个代码库，将常用的函数放进去，按需取用，用户的浏览器只需下载一份代码，便可在各个网站上使用。正如 jQuery 的口号：The Write Less, Do More, JavaScript Library。

难能可贵的是，Bootstrap 依旧本着这样的设计原则：并行开发、作为产品的风格指南、迎合所有的技能水平，帮助开发者解决实际问题，不断完善自己，吸引更多人选择在自己的项目中应用 Bootstrap。

Bootstrap 框架提供非常棒的视觉效果，且使用 Bootstrap 可以确保整个 Web 应用程序的风格完全一致，用户体验一致，操作习惯一致，这其实是很难的。如果希望整个网站的链接、按钮、提醒都有统一的视觉效果，那就应该毫不犹豫地选择 Bootstrap。它还可以对不同级别的提醒使用不同的颜色。

通过测试可知，市面上的主流浏览器都支持 Bootstrap 这一完整的框架解决方案，开发人员只需使用它而无需重新制作。而且这个框架专为 Web 应用程序而设计，所有元素都可以非常完美地在一起工作，很适合快速开发。

快速应用、简单而优雅，Bootstrap 会让 Web 应用程序看起来与 Windows 或 GNOME（GNU 网络对象模型环境）下的程序一样，具有一样的按钮，一样的对话框，且运行快速。随着越来越多的 Web 应用程序被直接放在桌面上运行，应用程序的一致性是一个趋势，开发人员可以把精力放在业务上，而不是 UI 设计上。

Bootstrap 始于 2011 年 8 月，至今才刚刚三年，但是在这三年里，Bootstrap 旋风已经刮遍了整个互联网。各种较小的网站就不提了，国内外很多较有名的网站也采用了 Bootstrap。例如，Name.com 已采用 Bootstrap，无论是从按钮的风格，还是从源代码中的 bootstrap_*.css 均可清晰地看出这一点；IP.cn 也采用 Bootstrap，这个可以从源代码中看出；新版的 WHMCS 已采用 Bootstrap，等等。

1.1.3 一位程序员的话

像我这样的程序员经常对设计望而却步，因为我是一名程序员，而不是设计师。拥有计算机专业学位证书的我对 Comic Sans 字体从不介意，但设计行业却极为排斥它，设计师或那些拥有美学情结的人往往对之不屑一顾。

虽然只是一名程序员，但我还是想让自己的网站看起来更加吸引人。这一方面是出于虚荣，因为这样可以让网站显得更加专业；而另一方面是出于现实的考虑，因为调查发现用户

会更加信任那些大气、设计讲究的网站。

可现实是，我一直从事编程工作，写代码就是工作的全部，我对设计并不熟悉，甚至有些畏惧。对于我这样的外行来说，设计是由很多只可意会不可言传的规则以及所谓的设计灵感混合作用而成的，知识壁垒很高。

然而，当我接触 Bootstrap 之后，我发现通过努力，我可以让网站看上去更加专业，虽然比不上设计师设计出来的效果，但对于像我这种没有设计基础的人来说已经足够了。

1.1.4　Bootstrap 构成模块

Bootstrap 构成模块从大的方面可以分为布局框架、页面排版、基本组件、jQuery 插件以及变量编译的 LESS 等部分。与前一版本相比，Bootstrap 2 增加了多个新模块。例如，布局框架中的响应式布局，页面排版中的 Icon，基本组件中的进度条，jQuery 插件也从以前的 5 种效果增加到 12 种，完全可以满足项目常用的交互效果。下面简单介绍一下 Bootstrap 中各模块的功能。

（1）页面布局

布局对于每个项目都必不可少。Bootstrap 在 960 栅格系统的基础上扩展出一套优秀的栅格布局，而在响应式布局中有更强大的功能，能让栅格布局适应各种设备。这种栅格布局使用也相当简单，只需要按照 HTML 模板应用，即可轻松构建所需的布局效果。

此外，改变模板中的类名；就能实现不同的布局风格。例如，要实现常见的固定布局，只需在 HTML 中添加 container 类名；而要实现流体布局，只需在 HTML 中添加 container-fluid 类名。

Bootstrap 还为开发者设计了 Responsive，让布局框架更为出色。开发者可以在此基础上覆盖任何样式，从而实现理想中的响应式设计。

（2）页面排版

页面排版的好坏直接影响产品风格，也就是说页面设计是不是好看。在 Bootstrap 中，页面的排版都是从全局的概念上出发，定制了主体文本、段落文本、强调文本、标题、Code 风格、按钮、表单、表格等格式。

Bootstrap 2 中添加了几个新亮点：

- 使用了 Google Prettify 插件，增强了代码的阅读体验；
- 在按钮中增加了组合、下拉、图标等效果；
- Bootstrap 在 Icon 部分采用了 Sprites 技术，为用户准备了上百种常用的 Icon 应用。

这里有必要向读者推荐 Font Awesome 项目。它是 Dave Gandy 在 Bootstrap 的基础上扩展而来的一个 Icon 主题，最大的特点在于，整套图标中没有运用任何图片。Font Awesome 项目中主要运用了 CSS3 的 @font-face 和伪元素一起实现。

（3）基本组件

基本组件是 Bootstrap 的精华之一，其中都是开发者平时需要用到的交互组件。例如，网站导航、标签页、工具条、面包屑、分页栏、提示标签、产品展示、提示信息块和进度条等。这些组件都配有 jQuery 插件，运用它们可以大幅度提高用户的交互体验，使产品不再那么呆板、无吸引力。

（4）jQuery 插件

Bootstrap 中的 jQuery 插件主要用来帮助开发者实现与用户交互的功能，Bootstrap 1 提供了 6 种常见插件。

- 模态对话框（Modal）：在 JavaScript 模板基础上自定义的一款流线型、灵活性极强的弹出蒙板效果的插件。
- 下拉项（Dropdown）：Bootstrap 中一款轻巧实用的插件，可以帮助实现下拉功能，如下拉菜单、下拉按钮、下拉工具条等。
- 滚动监听（Scrollspy）：实现滚动条位置的效果，如在导航中有多个标签，用户单击其中一个标签，滚动条会自动定位到导航中标签对应的文本位置。
- 标签页（Tab）：这个插件能够快速实现本地内容的切换，动态切换标签页对应的本地内容。
- 工具提示（Tooltip）：一款优秀的 jQuery 插件，无需加载任何图片，采用 CSS3 新技术，动态显示 data-attributes 存储的标题信息。
- 弹出提示（Popover）：在 Tooltips 的插件上扩展，用来显示一些叠加内容的提示效果，此插件需要配合 Tooltips 使用。

Bootstrap 2 在前面 6 种插件的基础上又新增加了 6 种 jQuery 插件。

- 警告框（Alert）：用来关闭警告信息块。
- 按钮（Button）：用来控制按钮的状态或更多组件功能，如复选框、单选按钮，以及载入状态条等。
- 折叠（Collapse）：一款轻巧实用的手风琴插件，可以用来制作折叠面板或菜单等效果。
- 轮播（Carousel）：实现图片播放功能的插件。
- 输入提示（Typeahead）：可以记住文本框输入的文本，下次输入时可以自动补全。
- 过渡效果（Transition）：Bootstrap 使用这个插件为一些动画效果增加了过渡，使动画效果更细腻、生动。

上面简单介绍了 Bootstrap 2 中的 jQuery 插件，至于如何使用，还需要根据 Bootstrap 所提供的文档，以及各插件的参数，具体问题具体分析。只有充分了解，才能灵活运用。

（5）动态样式语言 LESS

LESS 是动态 CSS 语言，它基于 JavaScript 引擎或者服务器端对传统的 CSS 进行动态扩展，具有更强大的功能和更好的灵活性。基于 LESS，编辑 CSS 就可以像使用编程语言一样，定义变量、嵌入声明、混合模式、运算等。

Bootstrap 中有一套编辑好的 LESS 框架，开发者可以将其应用到自己的项目中，也可以通过 less.js、Less.app 或 Node.js 等方法来编辑 LESS 文件。LESS 文件一旦编译，Bootstrap 框架就仅包含 CSS 样式，这意味着没有多余的图片、Flash 之类的元素。

（6）jQuery UI Bootstrap

jQuery UI Bootstrap 其实是从框架中衍生出来的一个 jQuery UI 主题。受到 Twitter 项目的启发，Addy Osmani 也在 Bootstrap 的基础上整理出一个 jQuery UI Bootstrap 主题。

jQuery UI Bootstrap 除了包含 Bootstrap 各个方面的功能之外，还在其基础上补充了以下

特性：动态添加标签页、日期范围选择组件、自定义文件载入框、滑动块、日期控件。

1.2　Bootstrap 功能介绍

Bootstrap 为什么能这么流行？其易用性是很重要的一个原因。Twitter 在 GitHub 上提供了方便的自定义 Bootstrap 的工具，用户可以自由地选择想要的元素和组件，并且自定义每个颜色的具体数值，然后再让它自动生成一份属于自己的、定制版的 Bootstrap。所有不需要的代码已经被去掉，以节省网络下载时间。虽然整个 Bootstrap 并不大，但是毕竟能省一点是一点。对于大流量网站来说，为每个用户节省 1KB 流量，总量也是很可观的。下面简单介绍一下 Bootstrap 功能和特色，以便更详细地了解它。

1.2.1　Bootstrap 主要特色

Bootstrap 是非常棒的前端开发工具包，它拥有以下特色。

（1）由匠人造，为匠人用

与所有前端开发人员一样，Bootstrap 团队是国际上最优秀的前端开发组织，他们乐于创造出色的 Web 应用，同时希望帮助更多同行从业者，为同行提供更高效、更简洁的产品。

（2）适应各种技术水平

Bootstrap 适应不同技术水平的从业者，无论是设计师，还是程序开发人员，不管是骨灰级别的大牛，还是刚入门槛的菜鸟。使用 Bootstrap 既能开发简单的小东西，也能构造更为复杂的应用。

（3）跨设备、跨浏览器

最初设想的 Bootstrap 只支持现代浏览器，不过新版本已经能支持所有主流浏览器，甚至包括 IE7。从 Bootstrap 2 开始，提供对平板和智能手机的支持。

（4）提供 12 列栅格布局

栅格系统不是万能的，不过在应用的核心层有一个稳定和灵活的栅格系统确实可以让开发变得更简单。可以选用内置的栅格，或是自己手写。

（5）支持响应式设计

从 Bootstrap 2 开始，提供完整的响应式特性。所有的组件都能根据分辨率和设备灵活缩放，从而提供一致性的用户体验。

（6）样式化的文档

与其他前端开发工具包不同，Bootstrap 优先设计了一个样式化的使用指南，不仅用来介绍特性，更用以展示最佳实践、应用以及代码示例。

（7）不断完善的代码库

尽管经过 gzip 压缩后，Bootstrap 只有 10KB 大小，但是它却仍是最完备的前端工具包之一，提供了几十个全功能的随时可用的组件。

（8）可定制的 jQuery 插件

任何出色的组件设计，都应该提供易用、易扩展的人机界面。Bootstrap 为此提供了定制

的 jQuery 内置插件。

（9）选用 LESS 构建动态样式

当传统的枯燥 CSS 写法止步不前时，LESS 技术横空出世。LESS 使用变量、嵌套、操作、混合编码，帮助用户花费很小的时间成本，编写更快、更灵活的 CSS。

（10）支持 HTML5

Bootstrap 支持 HTML5 标签和语法，要求建立在 HTML5 文档类型基础上进行设计和开发。

（11）支持 CSS3

Bootstrap 支持 CSS3 所有属性和标准，逐步改进组件以达到最终效果。

（12）提供开源代码

Bootstrap 全部托管于 GitHub（https://github.com/），完全开放源代码，并借助 GitHub 平台实现社区化开发和共建。

（13）由 Twitter 制造

Twitter 是互联网的技术先驱，引领时代技术潮流，Twitter 前端开发团队是公认的最棒的团队之一，整个 Bootstrap 项目由经验丰富的工程师和设计师奉献。

1.2.2 Bootstrap 主要功能

Bootstrap 的目的是提供一个便捷工具，方便快速开发项目，样式部分使用 LESS 写就，也提供了一些 jQuery 插件形式的扩展。

在样式方面，Bootstrap 提供了如下解决方案：

❑ 栅格系统

栅格系统与著名的 960 Grid 大同小异，不过 960 Grid 默认是 16 栏、940 像素宽，由于 LESS 带来的动态语言特性（变量、函数等），可以通过配置几个参数，自定义栅格。因为抛弃了对老旧浏览器的支持，可以直接用：last-child 将最后一栏的 margin-right 去掉，这也是它与 960 Grid 的区别之一。

❑ 布局

布局主要包括一个固定宽度的居中，一个可变宽度的浮动布局。

❑ 字体样式

字体风格比较明显，标题、、、<b>、<i> 以及 <address> 和 <blockquote> 等语义标签都配了一些默认样式。然后是列表，通过 .unstyled 类样式获得样式充值效果，这个方式比较实用，特别是网站是以内容为主的时候。代码块也比较朴素，借鉴了 Google Code Prettify 风格。

标签样式设计得比较贴心，风格趋于内敛，效果如图 1-2 所示。

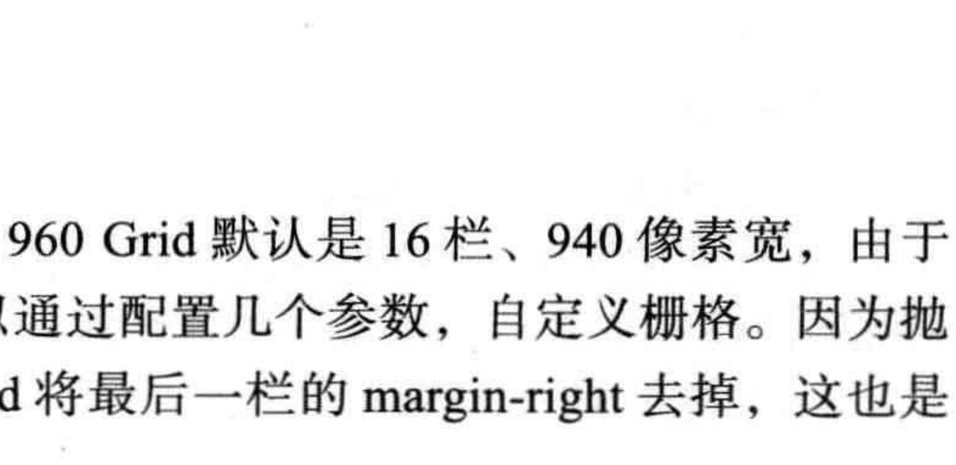


图 1-2 贴心的 label 标签

❑ 多媒体展现

多媒体列表其实也比较简单，定义了 3 种缩略图尺寸：330 像素 ×230 像素、210 像素 ×150 像素和 90 像素 ×90 像素。表格的样式也是简约风格。

❑ 表单

Bootstrap 对表单进行了比较充分的定制，风格比较鲜明，label 左浮动，圆角输入框，正确、错误的状态，表单 legend 的字号，前缀字符，输入、复选框等。文件选择比较朴素，没有提供更个性的解决方案，要想实现更个性的解决方案，需要与 JavaScript 配合设计。表单的按钮也十分细致，效果如图 1-3 所示。

按钮	class=""	描述
默认	bth	带渐变的标准灰色按钮
主要	bth bth-primary	提供额外的视觉感，可在一系列的按钮中指出主要操作
信息	bth bth-info	默认样式的替代样式
成功	bth bth-success	表示成功或积极的动作
警告	bth bth-warning	提醒应该谨慎采取这个动作
危险	bth bth-danger	表示这个动作危险或存在危险
反向	bth bth-inverse	备用的暗灰色按钮，不依赖于语义和用途
链接	bth bth-link	简化一个按钮，使它看起来像一个链接，同时保持按钮的行为

图 1-3　按钮样式

❑ 导航等

网站的全局导航栏保持一致，使用样式实现背景色渐变，固定在头部，因此不需要考虑浏览器兼容问题。此外，还实现了提示、警告、弹出对话框设计风格的统一。

以上样式的部分是 Bootstrap 框架的核心。在代码上，基本把样式重置与定制都做了，上述内容，除了比较明显的组件（如面包屑、翻页等），基本都直接用标签作选择器。

有许多风格是利用 CSS3 样式属性设计出来的，最明显的莫过于背景色渐变与圆角。因此，对于图省事、不介意早期浏览器的效果，不需要考虑 IE6 的开发者，Bootstrap 是个好选择，因为它省时省力，而且美观大方。

在样式之外，Bootstrap 还提供了十几个常用的 JavaScript 插件，如模态对话框、下拉菜单、滚动监听、标签页、工具提示等 jQuery 插件。

1.3　Bootstrap 应用项目赏析

Bootstrap 自从在 GitHub 上开源之后，迅速成为其上备受关注的项目。大量工程师踊跃为该项目贡献代码，社区惊人地活跃，代码版本进化非常快速，官方文档质量极高，可以说是非常优雅，同时涌现了许多基于 Bootstrap 建设的网站，这些网站界面清新、简洁，要素

排版利落大方。

1.3.1 Bootstrap 优秀网站

目前使用Bootstrap的著名案例有NASA和MSNBC的Breaking News。此外，很多CMS也在运用Bootstrap框架，如大家熟悉的WordPress、Drupal等。如果想了解更多Bootstrap案例，可以参考Wrapbootstrap.com。

下面介绍3个不同类型的国内网站，这些网站虽然不为人知，但分别从不同侧面展示了Bootstrap在开发中的应用效果。如果想了解更多的Bootstrap网站，可以浏览http://expo.bootcss.com/。

（1）乐窝（http://www.lewoer.com/）

乐窝是一个专为大学生和都市年轻人提供一个靠谱的、无中介的、利用地图租房和找房的工具，如图1-4所示。

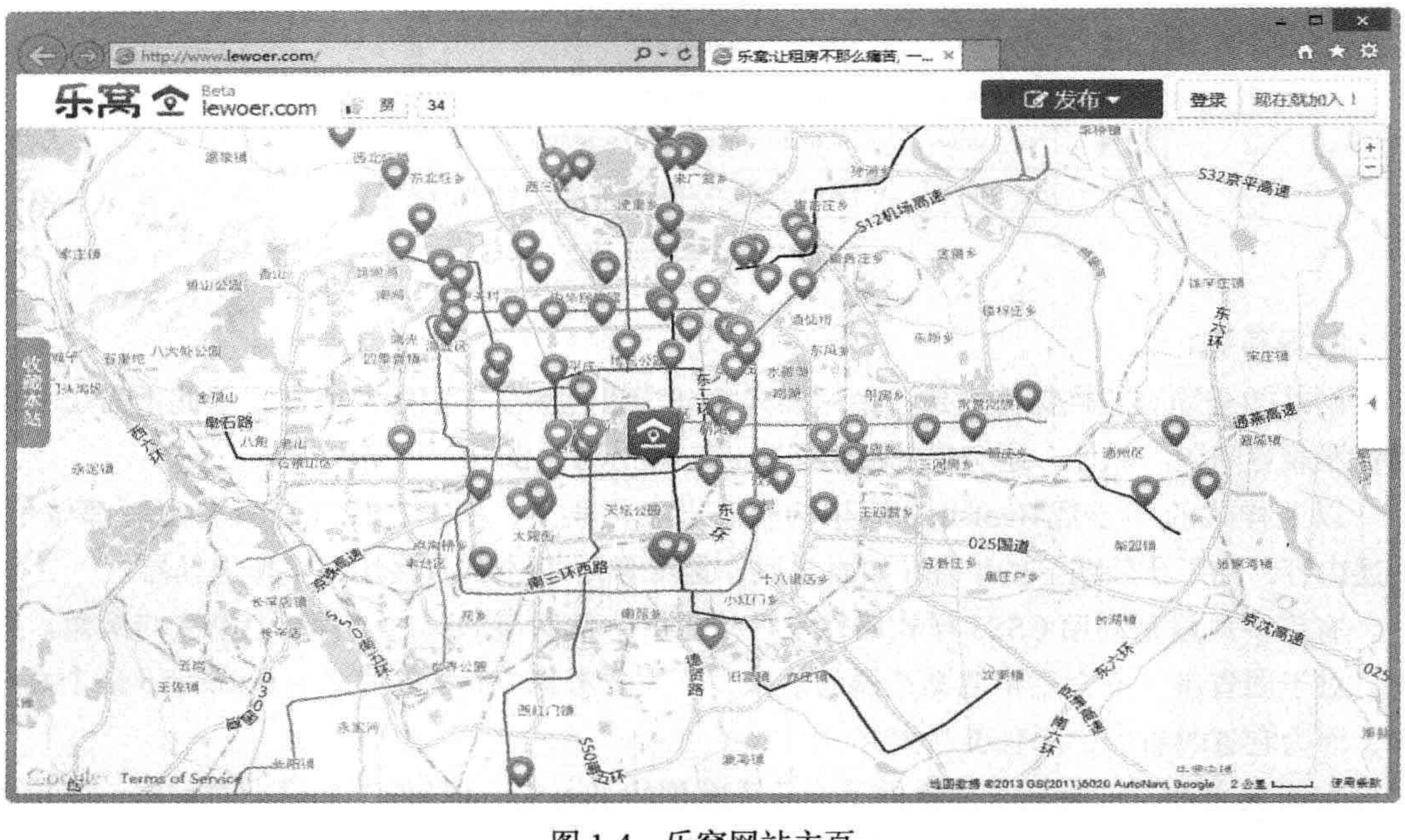


图1-4 乐窝网站主页

（2）佚站（http://www.yeahzan.com/）

佚站主要以建设展示型网站为起点，为个人、企业、组织与社会搭建网络信息平台，简单描述就是提供网站建设和网页设计的网站，主页效果如图1-5所示。

（3）翁天信（http://www.dandyweng.com/）

翁天信是一个个人网站，整个网站全部页面采用Bootstrap设计，页面整体效果大方、美观，如图1-6所示。

图 1-5　佚站网站主页

图 1-6　翁天信网站主页

1.3.2　Bootstrap 优秀插件

基于 HTML5 和 CSS3 的 Bootstrap，具有大量的诱人特性：友好的学习曲线、卓越的兼容性、响应式设计、12 列栅格布局、样式向导文档、自定义 jQuery 插件、完整的类库以及

基于 LESS 等。因此，也出现了大量以 Bootstrap 为基础的扩展技术插件，下面简单介绍几个比较著名的案例，要了解更多的案例可以访问 http://www.bootcss.com/。

（1）Sco.js（http://www.bootcss.com/p/sco.js/）

由于 Bootstrap 中大部分 JavaScript 插件是无法扩展的，因此才有了 Sco.js，它是对 Bootsrap 中 JavaScript 插件的增强实现。

（2）Chart.js（http://www.bootcss.com/p/chart.js/）

Chart.js 是一个简单、面向对象、为设计者和开发者准备的图表绘制工具库。

（3）bsie（http://www.bootcss.com/p/bsie/）

bsie 弥补了 Bootstrap 对 IE6 的不兼容。目前，bsie 能在 IE6 上支持 Bootstrap 的大部分特性。

（4）jQuery UI Bootstrap（http://www.bootcss.com/p/jquery-ui-bootstrap/）

jQuery UI Bootstrap 允许在使用 jQuery UI 控件时充分利用 Bootstrap 的样式，而且不会出现样式不统一的现象，使 Bootstrap 和 jQuery UI 可以完美融合。

（5）Flat UI（http://www.bootcss.com/p/flat-ui/）

Flat UI 是基于 Bootstrap 做的 Metro 化改造，由 Designmodo 提供。Flat UI 包含了很多 Bootstrap 提供的组件，但是外观更加漂亮。

（6）Metro UI CSS（http://www.bootcss.com/p/metro-ui-css/）

Metro UI CSS 是一套用来创建类似于 Windows 8 Metro UI 风格网站的样式。现在，Metro UI CSS 项目在 Bootstrap 的基础上被开发成一个独立的解决方案。

（7）HTML5 Boilerplate（http://www.bootcss.com/p/html5boilerplate/）

HTML5 Boilerplate 是一套专业的前端模板，用以开发快速、健壮、适应性强的 App 或网站。

（8）BootstrapEd（http://bootstraped.org/base-css.html）

BootstrapEd 基于 Bootstrap，并且优化了 Bootstrap 在中文 Web 环境中的效果。它增强了 Bootstrap 中的内置组件，增加了有价值的通用组件。

1.4 Bootstrap 版本变化

了解 Bootstrap 版本变化的过程，能够更直观地了解 Bootstrap 在 Web 开发中的地位和价值。通过其版本演变和功能变化，能够把握未来 Web 前端开发技术的发展方向。

1. Bootstrap 1

2011 年 8 月，Twitter 推出了用于快速搭建 Web 应用程序的轻量级前端开发工具 Bootstrap，该工具由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作完成。Bootstrap 是一套用于开发 Web 应用程序，符合 HTML 和 CSS 简洁但优美、规范的库。Bootstrap 由动态 CSS 语言 LESS 写成，在很多方面类似 CSS 框架 Blueprint。经过编译后，Bootstrap 就是众多 CSS 的合集。

Bootstrap 的内置样式继承了 Mark Otto 简洁亮丽的设计风格，便于开发团队快速部署一个外观尚可的 Web 应用程序。对于普遍缺乏优秀前端开发人员的创业团队来说，在某种程度上 Bootstrap 可以让他们在没有设计师的情况下完成一个 UI 较为理想的作品。Bootstrap 1 界面如图 1-7 所示。

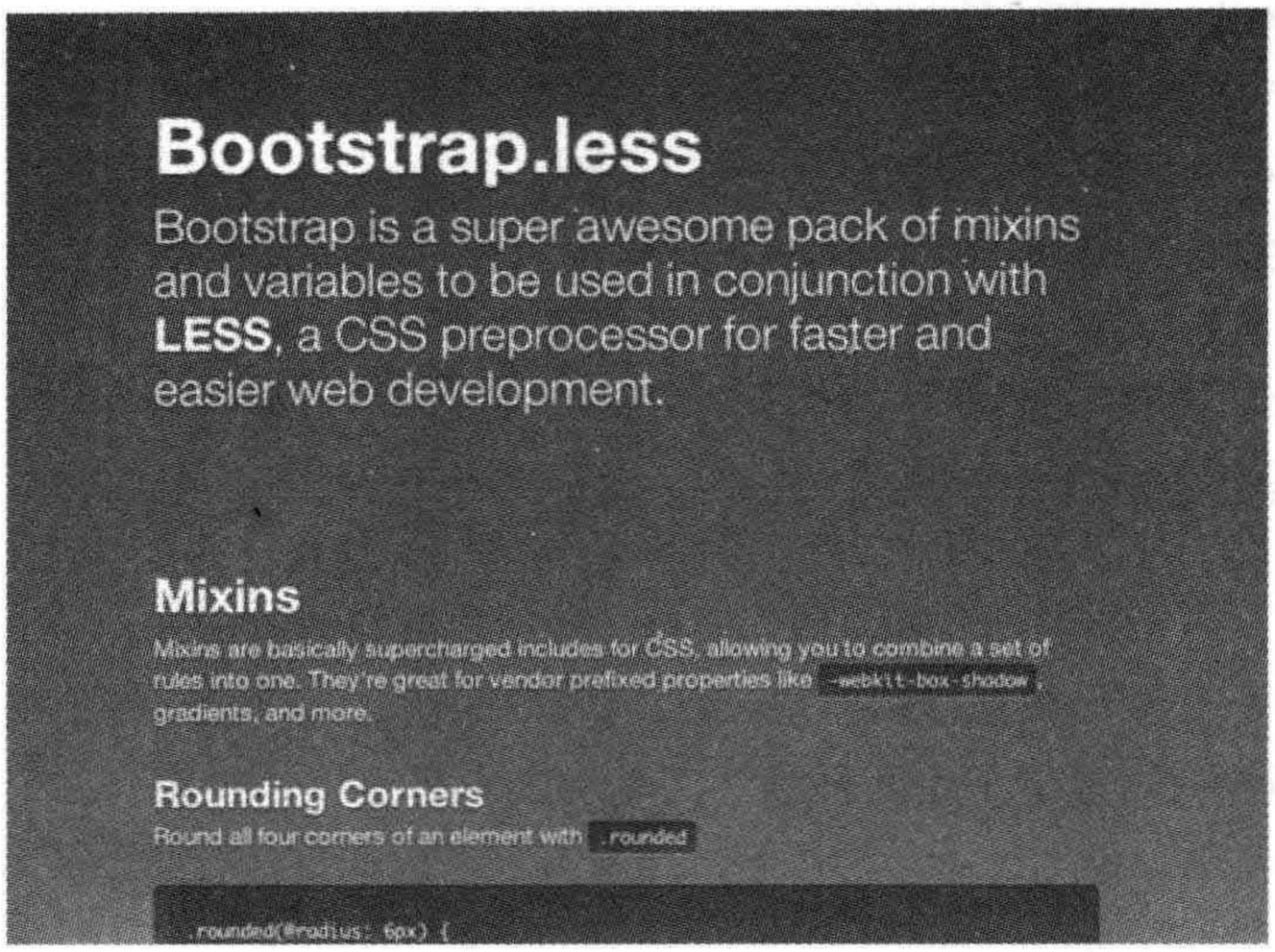


图 1-7　Bootstrap 1

2. Bootstrap 2

2012 年 1 月，Twitter 在开发者博客上公布消息，其 6 个月前发布的轻量级前端开发工具 Bootstrap 迎来重大改进，正式升级为 Bootstrap 2。和 Bootstrap 1 一样，Bootstrap 2 仍然是一个托管在 GitHub 上的开源项目。

在开发 Bootstrap 2 的过程中，Mark Otto 参考了不少来自社区的意见并借鉴了自己在 Twitter 前端重新设计过程中积累的经验。除了增加新样式外，Bootstrap 2 修改了一些网页元素的默认样式，除去了上一版本中的几十个 bug，同时完善了说明文档。BootStrap 2 在原有特性的基础上着重改进了用户的体验和交互性，比如新增加的媒体展示功能，适用于智能手机上多种屏幕规格的响应式布局，另外还新增了 12 款 jQuery 插件，可以满足 Web 页面常用的用户体验和交互功能。

Bootstrap 2 的一个重大改进是添加了响应式设计特性，Bootstrap 1 中并不支持这一特性，这让很多开发人员不满。为了提供更好地针对移动设备的响应式设计方案，Bootstrap 2 采用了更为灵活的 12 栏栅格布局。此外，它还更新了一些进度栏以及可定制的图片缩略图，并增加了一些新样式。值得关注的是，Bootstrap 是一个非常轻量级的框架，Bootstrap 2 在压缩后也只有 10KB。

Bootstrap 2 采用了更灵活也更受欢迎的 12 栏栅格布局，并以此来实现其各种布局框

架；增加了响应式设计，以适应各种移动终端的需求；完善和改进了原有样式库，并提供了更丰富的新样式，包括样式繁多的图标、漂亮易用的进度条等；改进和增加了自定义 jQuery 插件，完善文档，修复 bug，同时还提供了很多基于 Bootstrap 构建的网站样例。已经使用 Bootstrap 1.4 的开发者也不用担心，Bootstrap 专门提供了从 Bootstrap 1.4 升级到 Bootstrap 2 的手工向导可以参考。

Bootstrap 2 对现有框架进行了清晰的功能划分，主要分为脚手架（Scaffolding）、基础 CSS、构件库和 jQuery 插件库。

- Scaffolding 主要提供基于网格的各种布局，包括普通栅格系统、嵌入式栅格、固定布局、自适应布局，同时可以自定义网格和布局。Bootstrap 2 提供了响应式设计，可以通过单个文件支持各种手持设备，自适应不同的设备和屏幕变化。
- **基础 CSS** 包括各种排版样式（标题、段落、引用块、列表、内联标签等），在代码展示方面提供了基于 code 标签的内嵌代码，基于 pre 的块代码和基于 Google Prettify 的代码样式。此外，提供各种表格、表单、按钮、图标的展示方式。
- **构件库**提供了基于按钮、导航、标签、排版、警告、进度栏、图像网格等控件。
- **jQuery 插件库**则提供了十几种插件实现动态效果，例如模态对话框、下拉项、标签页、工具提示、弹出提示、轮播等，开发者可以根据自己的业务需求使用不同的插件实现各种动态效果。

应用 Bootstrap 2 的案例有 NASA（http://code.nasa.gov/，见图 1-8）和 MSNBC 的 BREAKING NEWS（http://www.breakingnews.com/，见图 1-9）。

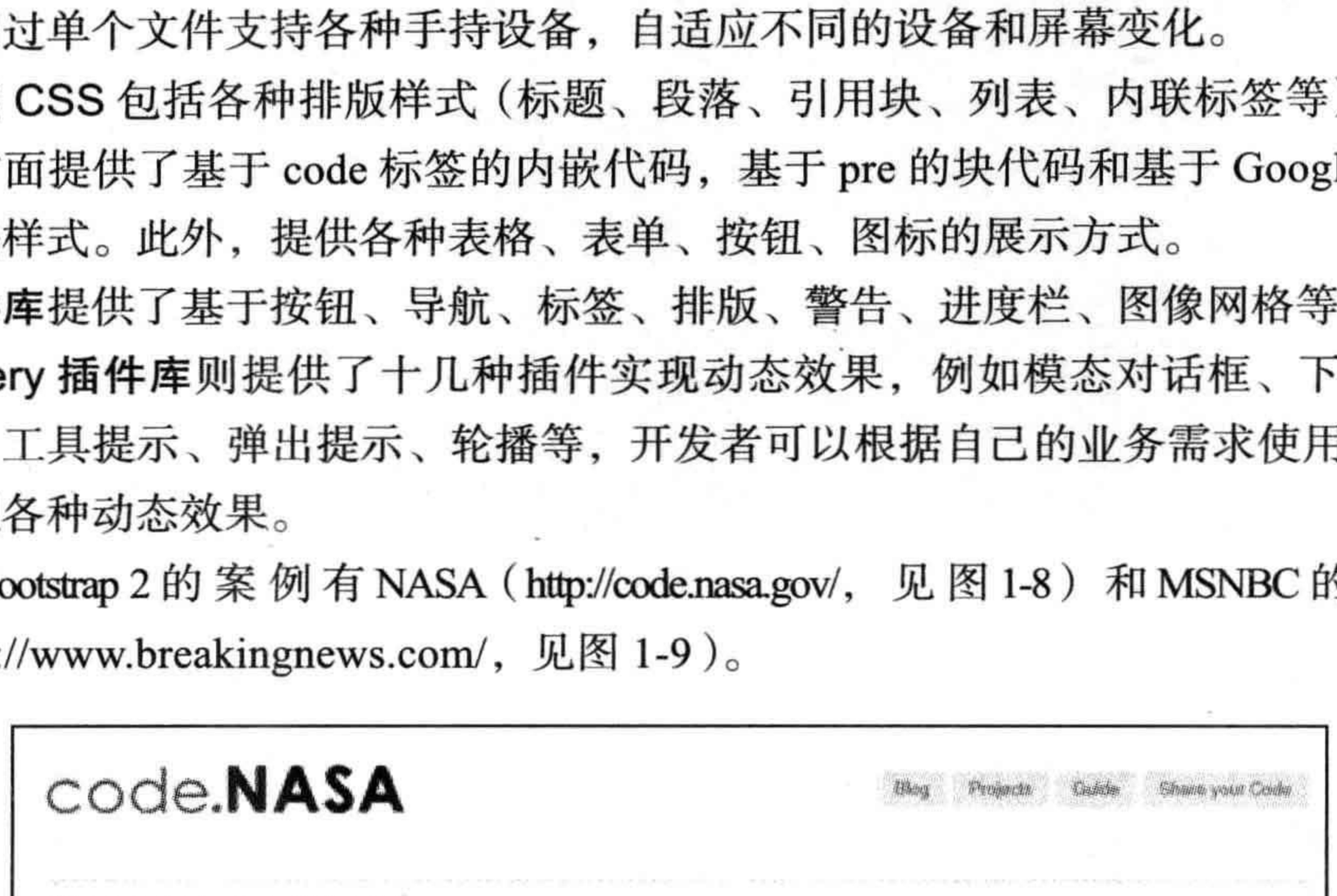


图 1-8　Bootstrap 2 应用案例（NASA）

Bootstrap 2 升级的细节请参考 http://twitter.github.io/bootstrap/upgrading.html。

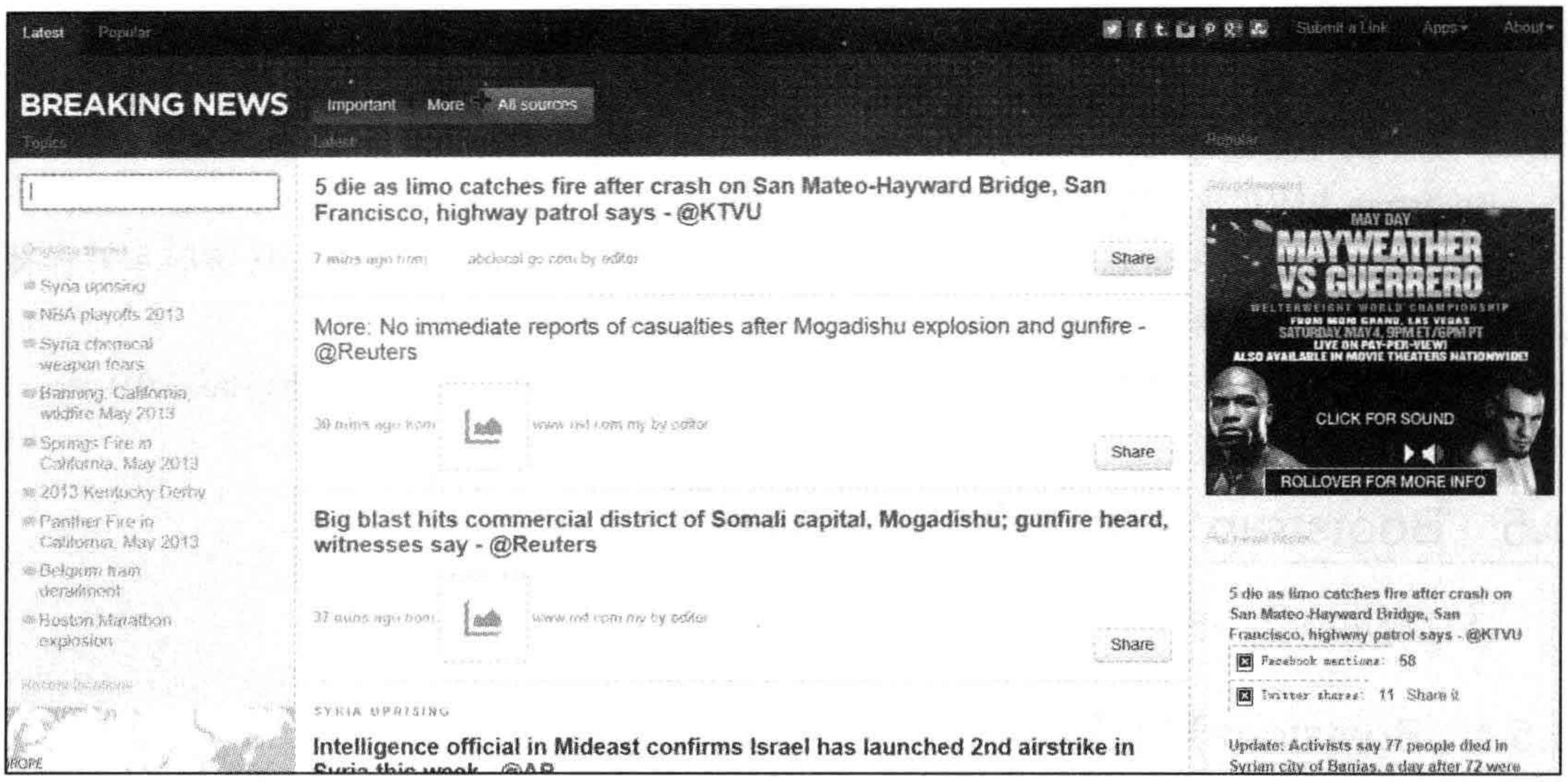


图 1-9　Bootstrap 2 应用案例（BREAKING NEWS）

3.Bootstrap 3

自 2012 年 12 月 Bootstrap 2.2.2 发布后，开发团队开始将精力放在下一个主要版本 Bootstrap 3 上。该团队在博客中透露了 Bootstrap 3 版本的开发计划（http://blog.getbootstrap.com/2012/12/10/bootstrap-3-plans/）。

Bootstrap 3 将主要致力于对之前版本的改善，将放弃一些旧代码，改善 CSS 的响应速度，并整合了社区所作出的努力。该版本的开发计划如下：

- 将 twitter/bootstrap、twitter/bootstrap-server 和 mdo/bootstrap-blog（一个私有库）迁移到 twbs。
- 更改网站 URL 为 http://getbootstrap.com，之前为跳转 URL。
- 将所有 LESS 代码（包括响应样式）编译到一个单独的 CSS 文件中。
- 完全放弃对 IE7、Firefox 3.x 的支持。
- 使用 @font-face 版本的 Glyphicons 图标，代替先前的 PNG 图标。
- 采用 MIT 许可证，代替先前的 Apache 许可证。
- 放弃 *-wip 分支样式的开发。
- 使用 tag 下载所有版本，开发工作使用更小的功能分支，并在 Bootstrap 3 发布后合并到主分支。

2013 年 3 月，Bootstrap 发布了最新的预览版本，标签为 Bootstrap 3，这是它的第三个主要发行版本。该版本的主要更新（https://github.com/twitter/bootstrap/pull/6342）与开发计划基本一致，只是还改进了响应式 CSS，集中了来自社区的贡献。

这只是预览版本，到 Bootstrap 3 正式版发布时还会有更多的改进。该版本被标签为“移动优先”，因为进行了完全重写以更好地适应手机浏览器。移动的风格可直接从库中感

受到。

其他的改变还包括转换文档到 Jekyll 替代 Mustache，重做插图，更新支持 Retina 的示例截图，重新设计图标预览，更新所有示例以演示新的改进，通过插件改进 noConflict 等。此外，Bootstrap 3 还包含一些其他重要的改进。

Bootstrap 3 包含大量新特性，这将需要很长时间来开发，不过可以先从该版本开始测试。完整的介绍内容参见 https://github.com/twitter/bootstrap/pull/6342。

读者可以从以下网址获取 Bootstrap 3 预览版本：https://github.com/twitter/bootstrap/tree/3.0.0-wip。

1.5 Bootstrap 开发工具和参考资源

下面简单介绍一下 Bootstrap 开发工具和参考资源。

1.5.1 Bootstrap 开发工具

实际上，通过任何网页设计工具都可以使用 Bootstrap，读者可以根据情况选择自己熟悉的工具。这里介绍一款专门针对 Bootstrap 的开发工具——Jetstrap，下载地址为 https://jetstrap.com/。

Jetstrap 是国外开发者为 Bootstrap 专门设计的可视化制作工具，允许其他开发者、设计师直接在网页端拖拽各个组件，制作出漂亮的网页。

通过 Jetstrap 制作出来的网页 100% 符合 Bootstrap 标准。只要设计了电脑端的页面，它就会自动适配手机端和 iPad 端（响应式设计）。这个工具有利于那些想快速搭建简单网页的用户，因为不需要去学习太多东西就可以做到。我们可以把它比作网页版的 Dreamweaver，但是使用更加简单。

另外，再介绍一个在线编辑工具：Layout It（http://www.layoutit.com/）。利用 Layout It 可以简单而又快速地搭建 Bootstrap 响应式布局，操作基本使用拖动方式来完成，而元素都是基于 Bootstrap 框架集成的，所以这工具很适合网页设计师和前端开发人员使用，快捷方便。

其英文版访问地址为 http://www.layoutit.com/build，中文版访问地址为 http://justjavac.com/tools/layoutit/。

1.5.2 Bootstrap 参考资源

Bootstrap 是个基于 HTML、CSS、JavaScript 的简洁、灵活的流行前端框架，我们可以把它想象成一个定义了很多效果的 CSS、JavaScript 的代码库，库里面已经定义好了各种组件的显示效果与动画，如栅格布局、各种按钮、表格、表单、下拉菜单、标签页、导航条等。对于初学者来说，花几个小时阅读本书，就能够快速了解其用法，然后只要按照它的使用规则使用就可以了。

- Bootstrap 中文参考：http://wrongwaycn.github.com/bootstrap/docs/index.html

- Bootstrap 英文参考：http://twitter.github.com/bootstrap/index.html
- Bootstrap 中文网：http://www.bootcss.com/

Bootstrap 对于缺乏设计基础的人是很有用处的，只需快速调用一下，就可以很容易构建出标准、精致的网站。遗憾的是，由于用到一些 CSS 新技术，它只支持现代浏览器，对 IE6 等老版本的浏览器支持效果不是很理想。不过，在 Github 上有个 bootstrap-ie 项目可以支持 IE 早期版本，感兴趣的读者可以参阅：

https://github.com/empowering-communities/Bootstrap-IE6

此外，浏览一些服务网站也会对你学习 Bootstrap 有所帮助。

- Bootstrap 版本参考手册全集（英文版：http://bootstrapdocs.com/ ；中文版：http://docs.bootcss.com/）。
- Bootstrap 应用网站导航（http://lovebootstrap.com/）
- Bootstrap 应用快速链接：提供 Bootstrap 各个版本的网络引用地址，同时提供了各种字体图标的链接地址，非常实用，网站为 http://www.bootstrapcdn.com/?v=102920122257O5。

第 2 章

使用 Bootstrap 的准备

本章内容

- 下载和定制 Bootstrap
- Bootstrap 的文件结构
- Bootstrap 应用解析
- 开发第一个 Bootstrap 示例

在上一章中，我们简单介绍了 Bootstrap 框架的来龙去脉，以及相关技术知识和话题，为了帮助读者快速入门，引导用户正确使用 Bootstrap，本章将遵循官方文档结构进一步介绍初步使用 Bootstrap 的方法和步骤，为后面系统、深入学习 Bootstrap 奠定扎实的技术基础。

2.1　下载和定制 Bootstrap

在下载 Bootstrap 之前，先确保系统中准备好了一个网页代码编辑器，本书使用 Dreamweaver 网页设计和编辑软件。另外，读者应该对自己的网页制作水平进行初步评估，评估自己是否基本掌握 HTML 和 CSS 技术，以确保能够在网页设计和开发中轻松学习和使用 Bootstrap。

2.1.1　下载 Bootstrap

Bootstrap 压缩包包含两个版本，一个是供学习使用的完全版，另一个是供应用的编译版。

1. 下载源码版 Bootstrap

访问 http://github.com/twbs/bootstrap/ 页面，下载最新版本的 Bootstrap 压缩包。在访问 GitHub 时，找到 Twitter 公司的 Bootstrap 项目页面，单击 ZIP 选项卡，即可下载保存 Bootstrap 压缩包，如图 2-1 所示。从 GitHub 直接下载到的最新版源码包括 CSS、JavaScript 的源文件，以及一份文档。

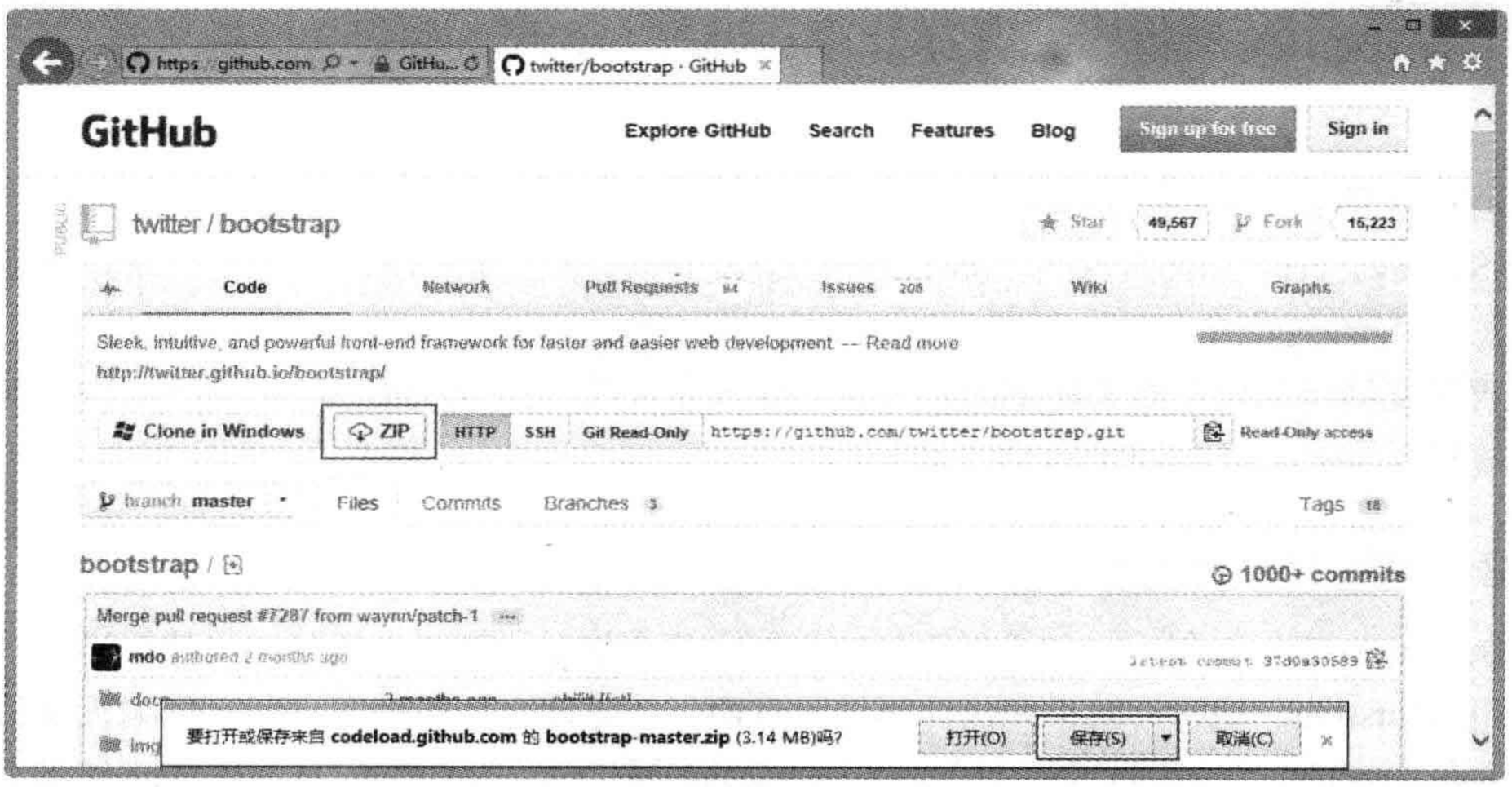


图 2-1　下载 Bootstrap 开发包

通过这种方式下载的 Bootstrap 压缩包，名称为 bootstrap-master.zip，包含 Bootstrap 库中所有的源文件以及参考文档，适合读者学习和交流使用。

2. 下载编译版 Bootstrap

如果希望快速开始，可以直接下载经过编译、压缩后的发布版。访问下面页面之一，根据下载按钮提示直接下载即可，如图 2-2 所示。

- https://github.com/twitter/bootstrap
- http://kxh.github.io/Bootstrap_doc_in_chinese/
- http://www.bootcss.com/

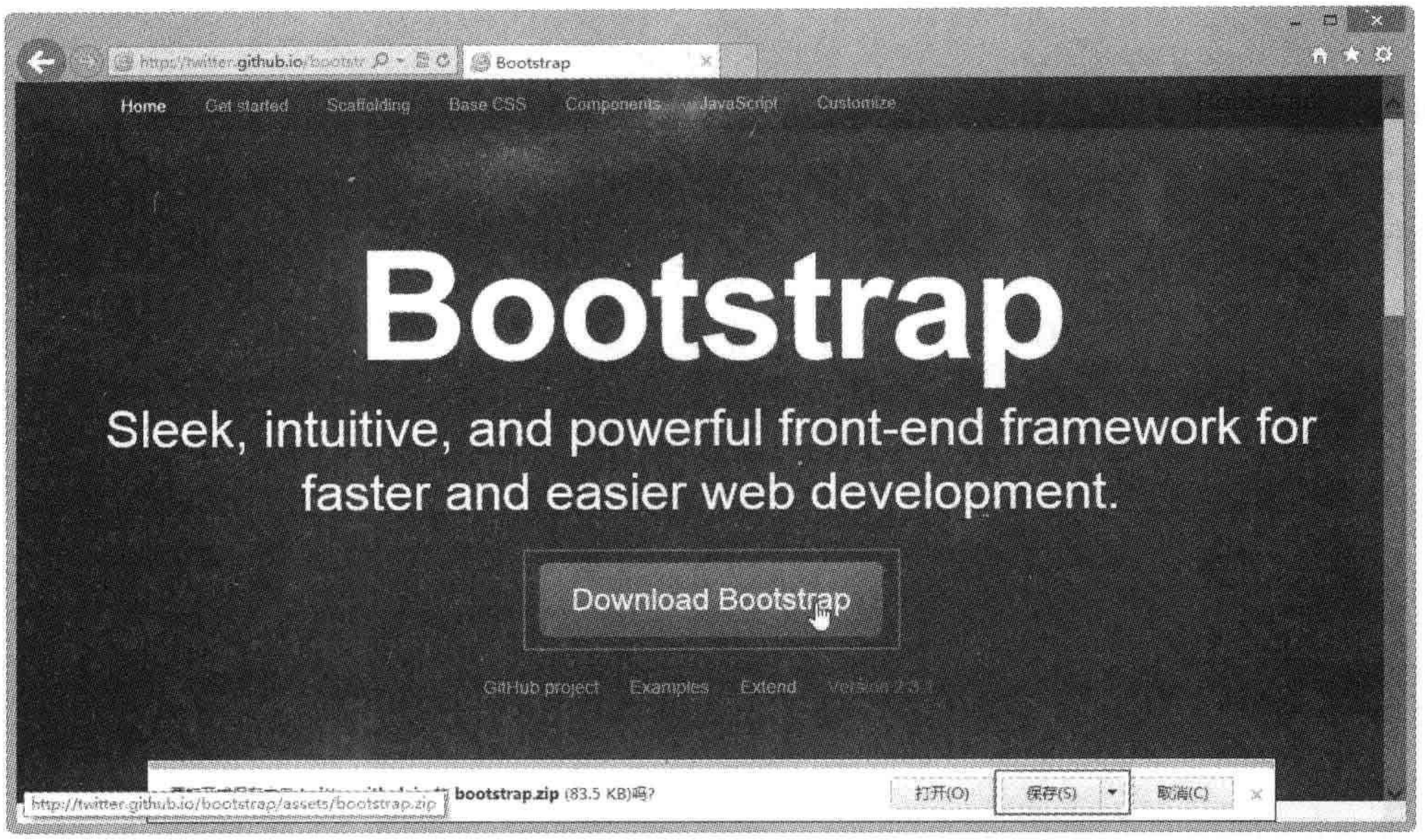


图 2-2　下载 Bootstrap 发布版

通过这种方式下载的压缩文件名为 bootstrap.zip，仅包含编译好的 Bootstrap 应用文件，如 CSS、JavaScript 和图片文件。而且所有文件已经过了压缩处理，不过文档和源码文件不包含在这个压缩包中。

直接复制压缩包中的文件到网站目录，导入相应的 CSS 文件和 JavaScript 文件，就可以在网站和页面中应用 Bootstrap 效果和插件了。

2.1.2　定制 Bootstrap

Bootstrap 库文件很大，如果仅希望应用其中几个效果或者特定插件，则建议通过定制方式使用 Bootstrap。把所有效果和插件都导入页面，一方面会增加带宽负荷，影响页面的响应速度；另一方面，众多的 CSS 类样式和 JavaScript 源代码，会与制作页面的样式和脚本发生冲突，影响解析时的执行效率和页面显示效果。

定制 Bootstrap 可以有效降低页面加载的负担和执行效率，降低潜在的源码冲突。定制的具体方法如下。

第 1 步：访问 http://twitter.github.io/bootstrap/ 页面，在顶部导航栏中单击 Customize（定制）选项卡，切换到定制页面，如图 2-3 所示。

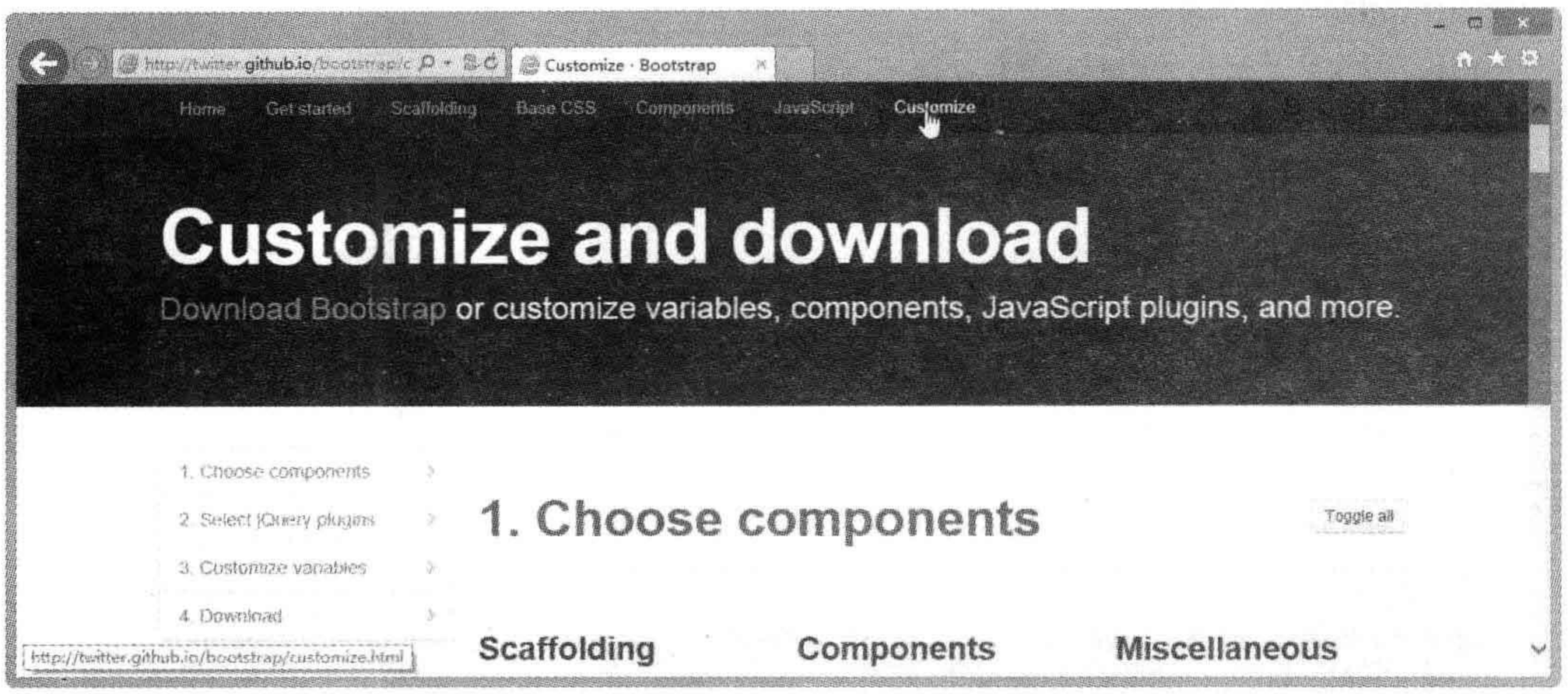


图 2-3　打开 Bootstrap 定制页面

第 2 步：选择组件。在页面左侧页内导航栏中单击“1.Choose components”（1. 选择组件）选项，切换到组件选择区，如图 2-4 所示。单击右上角的 Toggle all（切换全部）按钮，取消勾选所有选项，然后根据需要勾选组件。

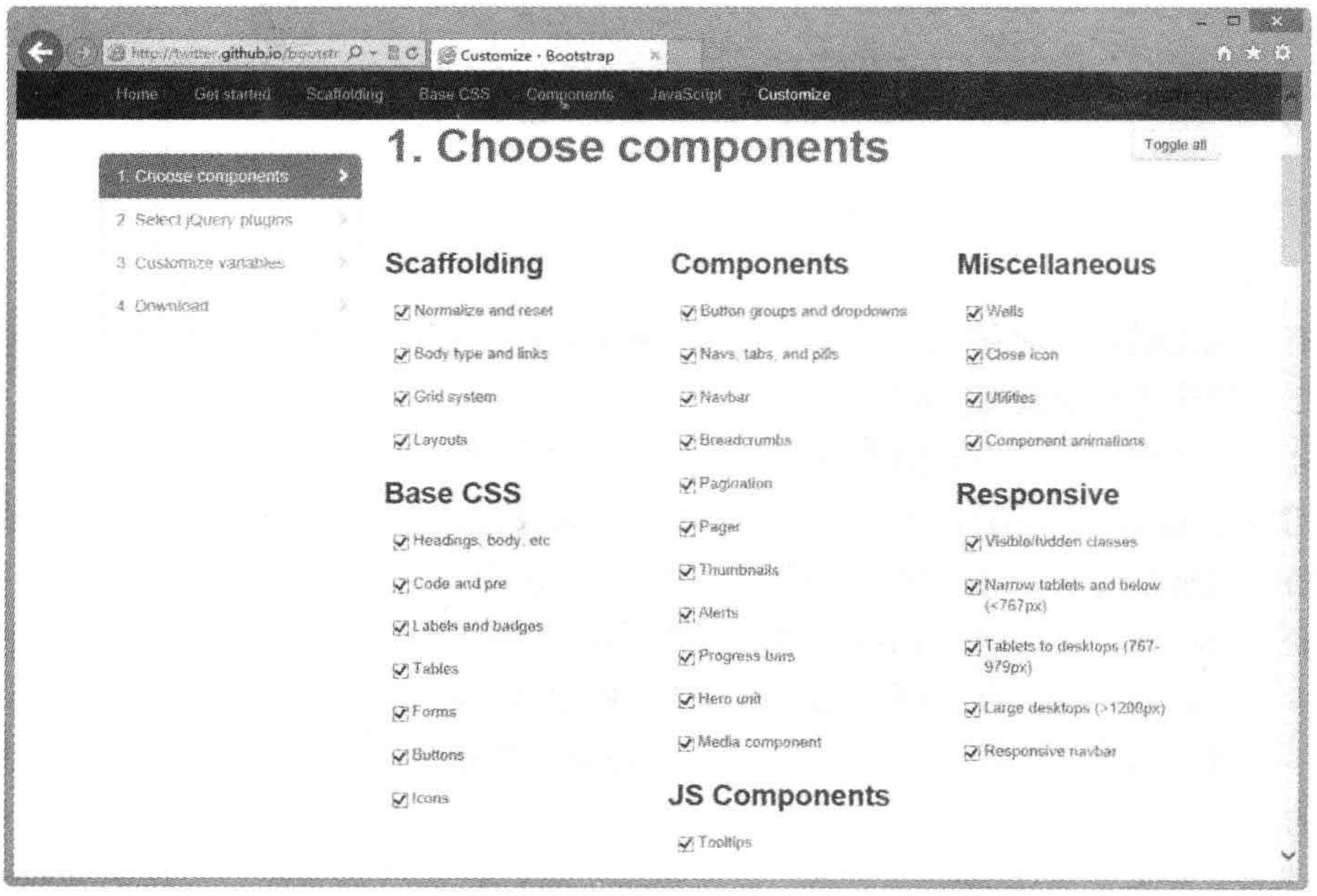


图 2-4　选择 Bootstrap 组件

组件包括如下几个部分，每部分又包含多个项目，这些部分将在后面几章中进行详细讲解。其中脚手架是用来设置页面基本样式和布局的，根据需要必须选择。

Scaffolding（脚手架）

- Base CSS（基本 CSS 样式）
- Components（组件）
- JS Components（JS 组件）
- Miscellaneous（杂项）
- Responsive（响应式交互）

第 3 步：选择 jQuery 插件。在页面左侧页内导航栏中单击“2.Select jQuery plugins”（选择 jQuery 插件）选项，切换到 jQuery 插件选择区，如图 2-5 所示。单击右上角的 Toggle all 按钮，取消勾选所有选项，然后根据需要勾选 jQuery 插件。

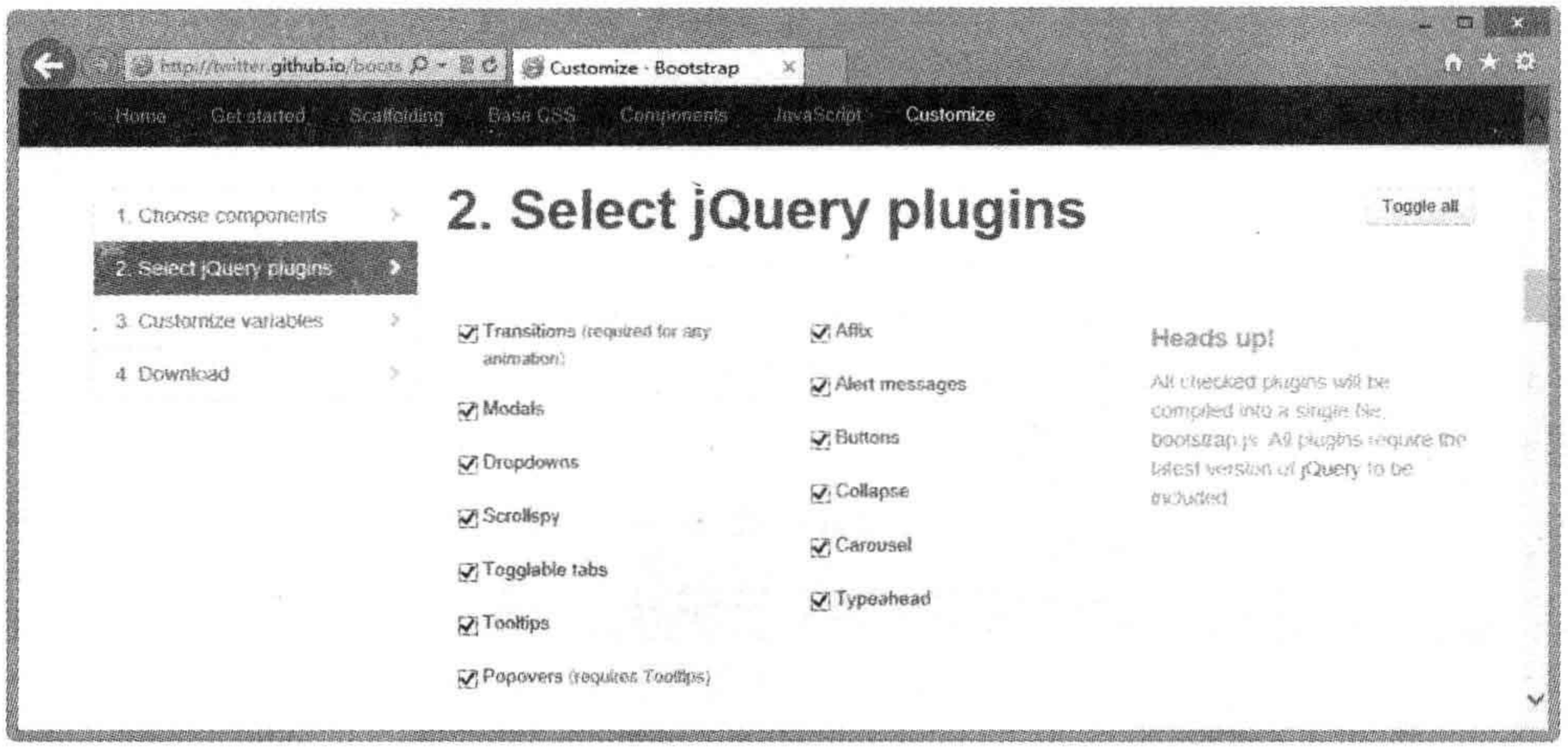


图 2-5　选择 jQuery 插件

所有被勾选的插件将被编译成一个文件 bootstrap.js。所有的插件都需要导入最新版本的 jQuery 库文件作为底层技术支撑。

第 4 步：定制变量。在页面左侧页内导航栏中单击“3.Customize variables”（定制变量）选项，切换到 LESS 变量配置区，如图 2-6 所示。如果在设置过程中需要恢复默认值，则单击右上角的 Reset to defaults（重置为默认）按钮，取消对所有 CSS 变量的设置，然后根据需要重设变量的名称。有关动态 CSS 技术的详细讲解可参阅后面章节内容。

第 5 步：打包下载。在页面左侧页内导航栏中单击“4.Download”（下载）选项，切换到下载按钮位置，如图 2-7 所示。单击“ Customize and Download”（定制并下载）按钮，下载定制后的 Bootstrap 压缩包。

下载的文件包括编译的动态 CSS、整理和压缩的 CSS 样式表，以及编译的 jQuery 插件，它们都很好地包装在一个 zip 文件中。

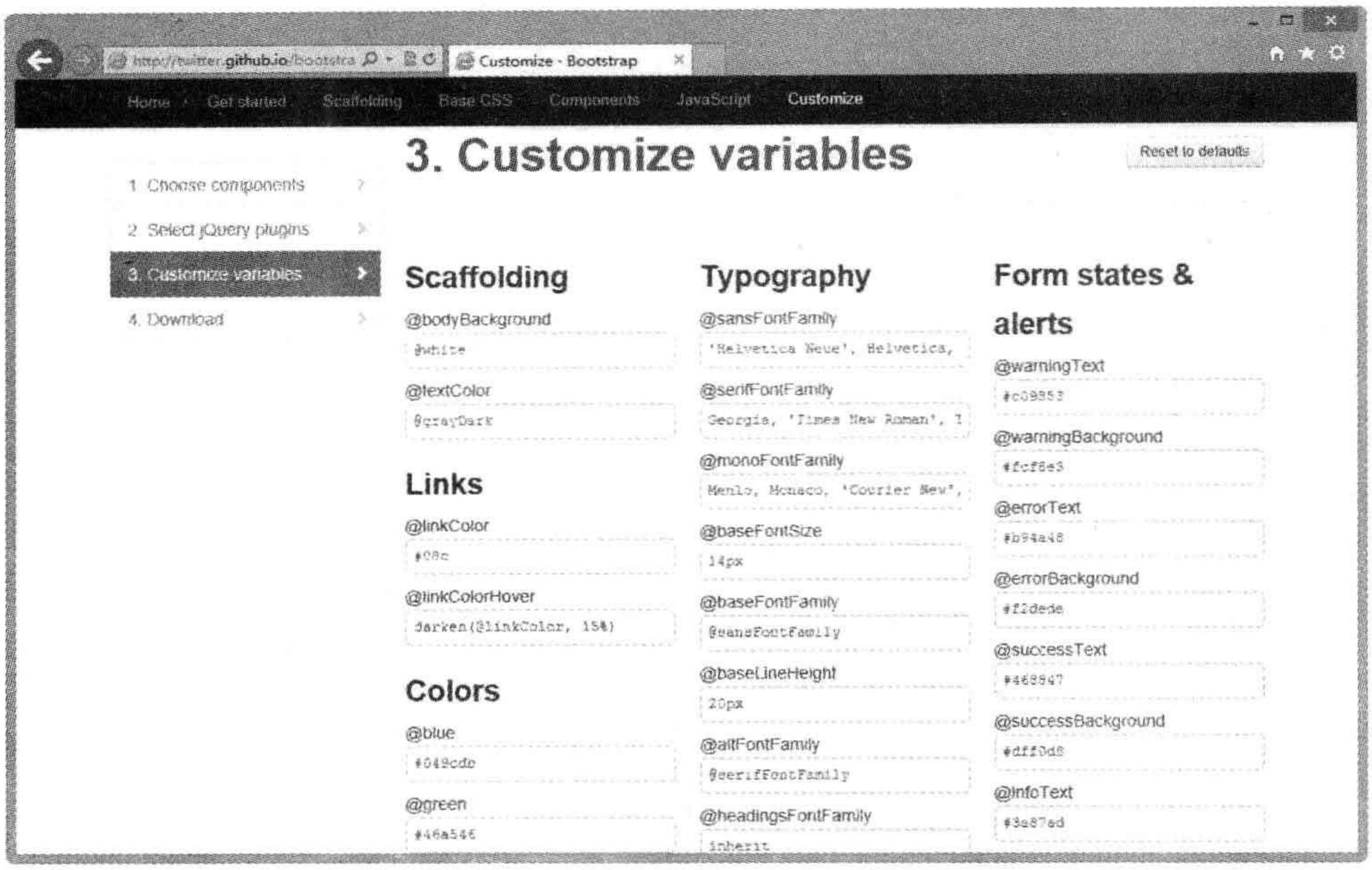


图 2-6　定制变量

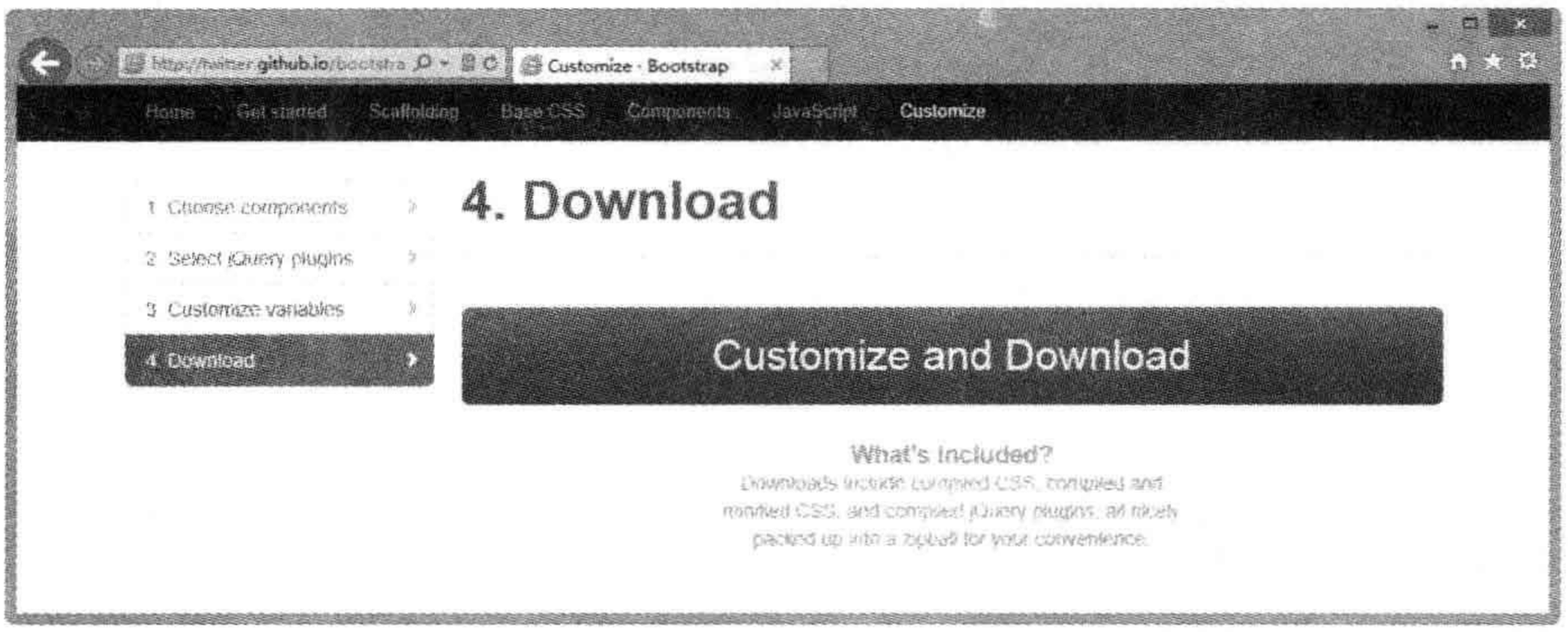


图 2-7　下载定制后的 Bootstrap 压缩包

例如，如果仅需要表格样式效果，则可以在第 2 步中仅勾选两项：“Scaffolding”（脚手架）子项中的“ Normalize and reset”和“ Base CSS”（基础 CSS 样式表）子项中的“ Tables”，如图 2-8 所示。

在第 3 步中，取消勾选所有选项，即不下载任何 jQuery 插件。在第 4 步中，保持所有动态 CSS 变量的默认值。然后，单击“ Customize and Download”按钮，下载定制后的 Bootstrap 压缩包（bootstrap.zip），该文件大小仅有 38KB，文件结构如图 2-9 所示。

如果直接下载默认编译好的压缩包，大小为 375KB，当页面不需要全部效果和交互行为时，这种做法显然就不妥了。

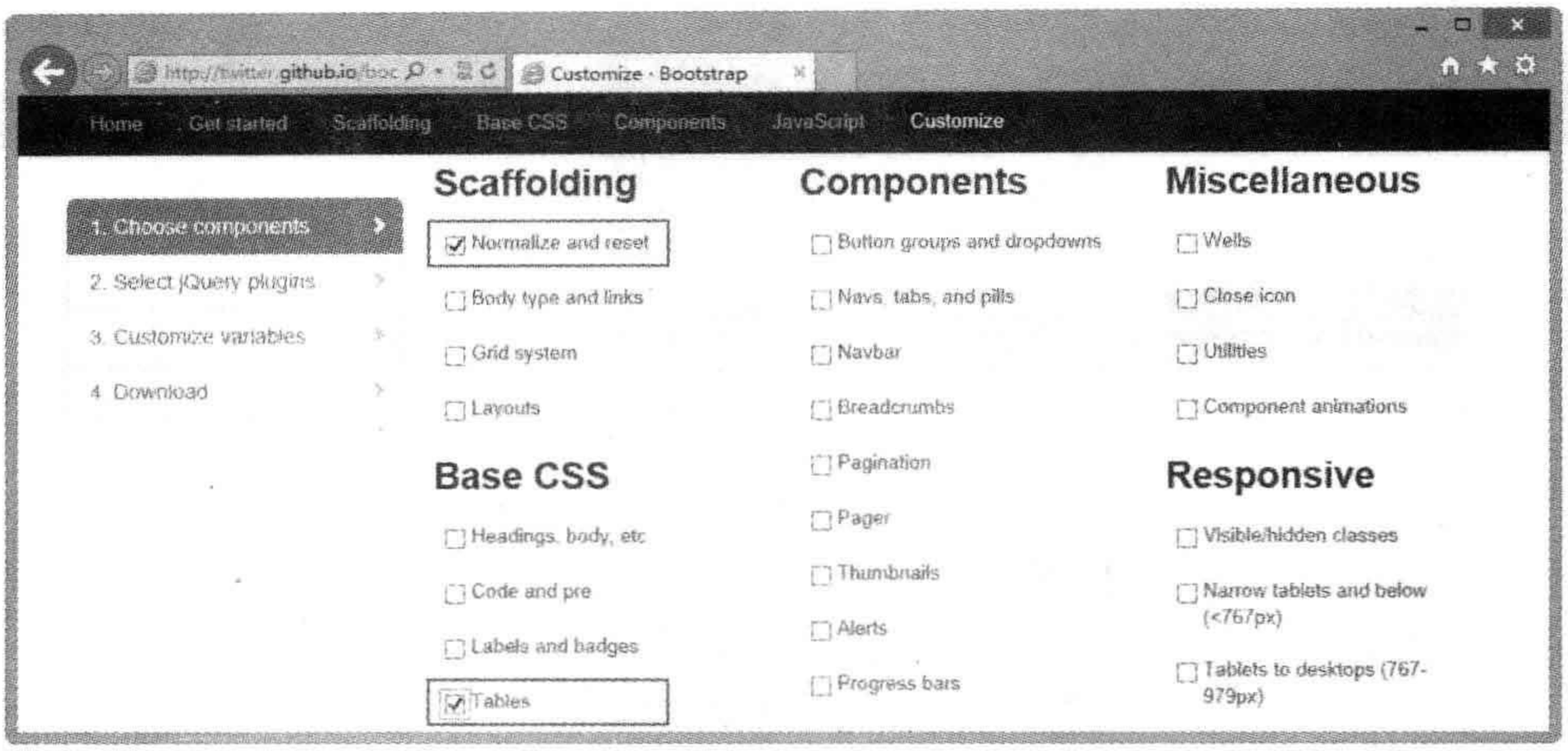


图 2-8　定制表格样式

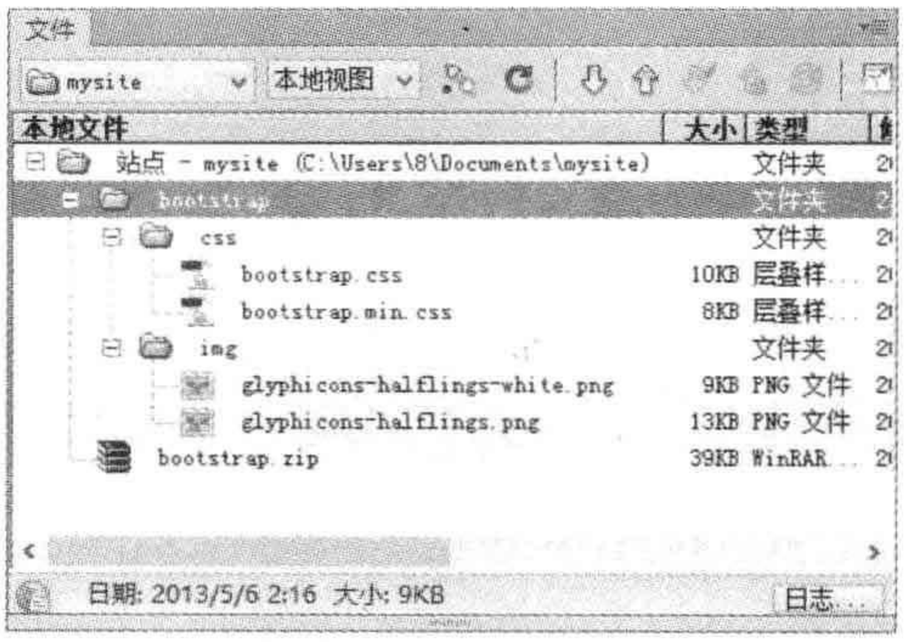


图 2-9　定制的文件结构

2.2　Bootstrap 的文件结构

下载 Bootstrap 压缩包之后，在本地进行解压，就可以看到包中包含的 Bootstrap 的文件结构，Bootstrap 提供了编译和压缩两个版本的文件，下面针对不同的下载方式进行简单说明。

2.2.1　源码版 Bootstrap 文件结构

在 2.1.1 节中，如果按照第一种方法，下载源码版 Bootstrap，则解压 bootstrap-master.zip 文件，可以看到该包中包含的所有文件，如图 2-10 所示。

在下载的压缩包中，可以看到所有文件按逻辑进行分类存储，简单说明如下。

- docs 文件夹：存储 Bootstrap 参考文档，在该文件夹中单击 index.html 文件，可以查阅相关参考资料，在 examples 子目录中可以浏览 Bootstrap 应用示例。

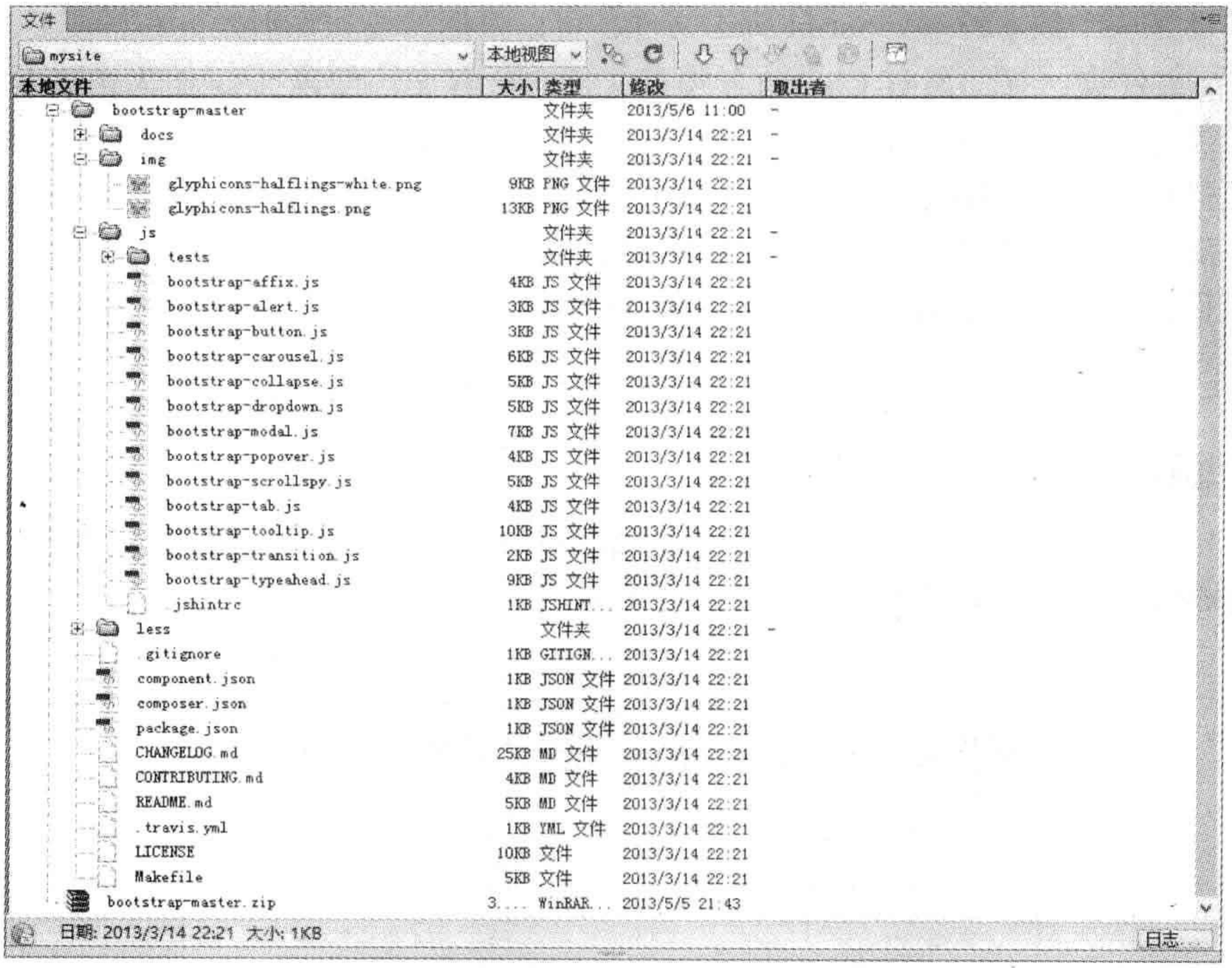


图 2-10　源码版 Bootstrap 文件结构

- img 文件夹：存储两张图片——glyphicons-halflings.png 和 glyphicons-halflings-white.png，它们通过 CSS Sprites 技术，把所有的图标整合到一个图片文件中，再利用 CSS 的 background-image、background- repeat、background-position 的组合进行背景定位，background-position 可以用数字精确地定位出背景图片的位置。利用 CSS Sprites 能很好地减少网页的 HTTP 请求，从而大大提高页面性能，这也是 CSS Sprites 最大的优点，也是其被广泛应用的主要原因。图 2-11 为 glyphicons-halflings.png 图片的效果，glyphicons-halflings-white.png 图片的效果与它是一致的，不过使用白色前景色进行设计。

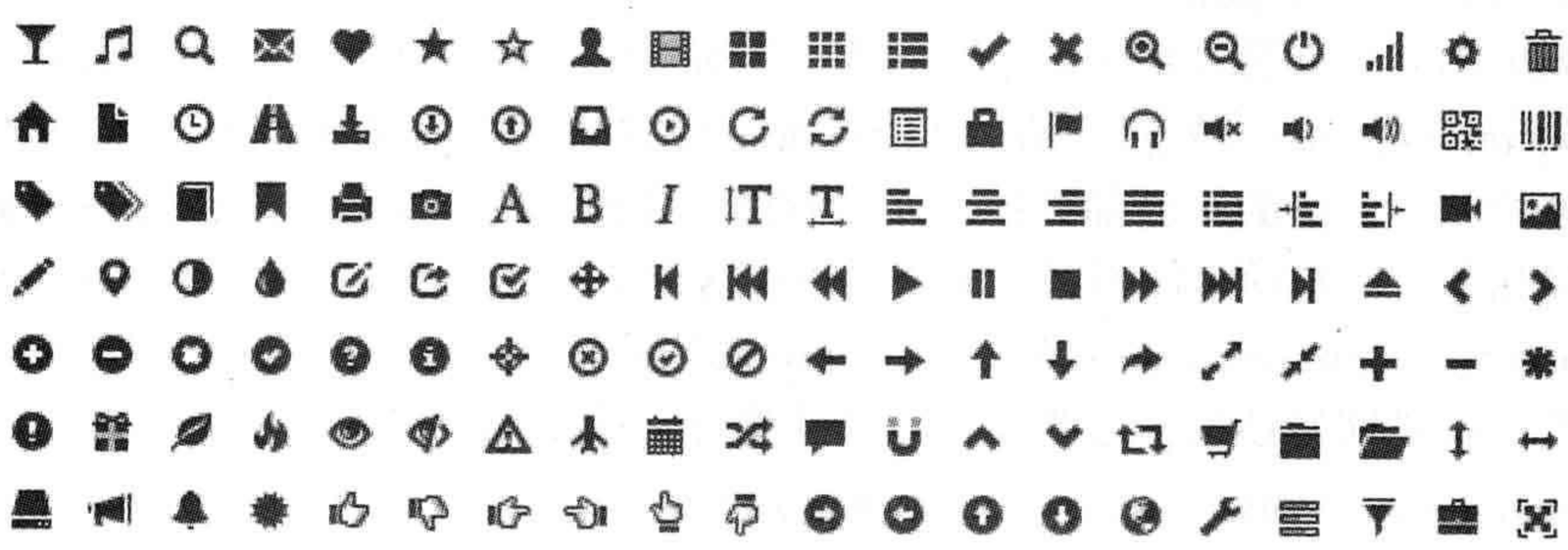


图 2-11　glyphicons-halflings.png 图片效果

- js 文件夹：存储各种 jQuery 插件和交互行为所需要的 JavaScript 脚本文件，每一个插

件都是一个独立的 JavaScript 脚本文件，可以根据需要独立引入。

- less 文件夹：存储所有 CSS 动态脚本文件，所有文件都以 .less 作为扩展名，但可以通过任何文本编辑软件打开。LESS 是动态样式表语言，需要编译才能在页面中应用，即只有 .less 文件被转换为普通的 CSS 样式表文件后才可以被浏览器正确解析。

其中最为重要的是 docs 目录下的 CSS 样式文件 .less 目录中的编译文件和 js 目录中的 jQuery 插件。bootstrap-master 目录下的 10 个文件可以不用管，它们是一些数据和服务性文件。

2.2.2 编译版 Bootstrap 文件结构

在 2.1.1 节中，如果按照第二种方法，下载编译版 Bootstrap，则解压 bootstrap.zip 文件，可以看到该包中包含的所有文件，如图 2-12 所示。

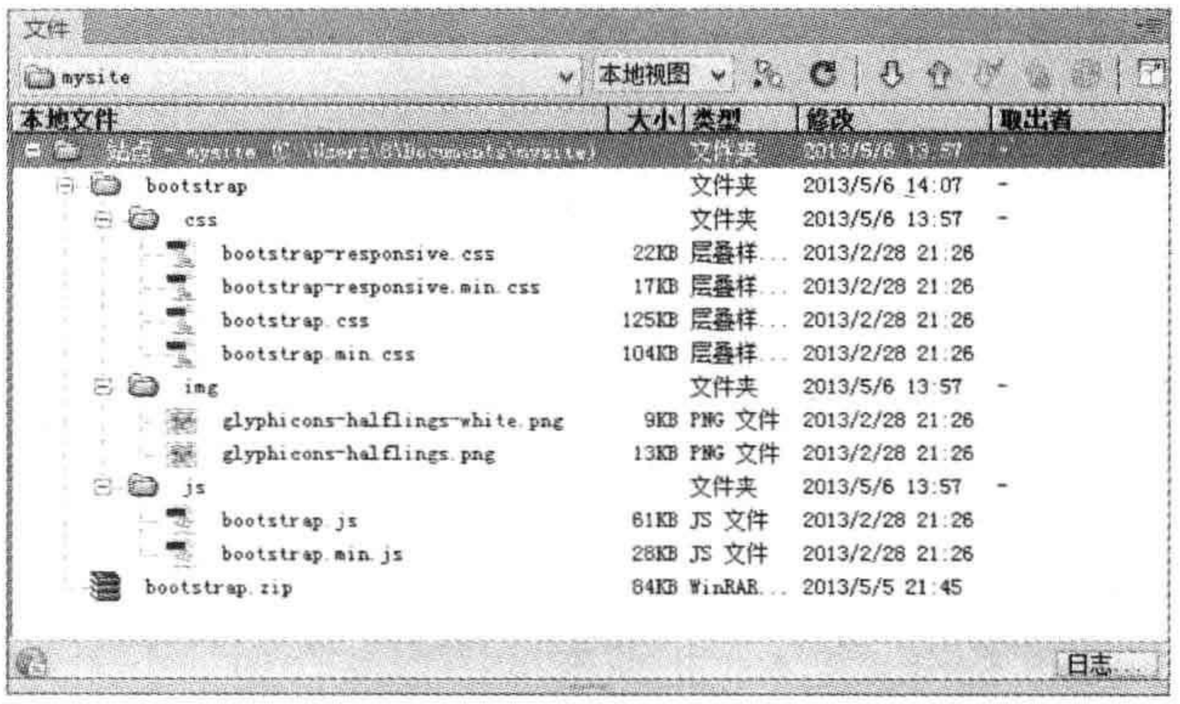


图 2-12 编译版 Bootstrap 文件结构

图 2-12 是 Bootstrap 的基本结构，编译后的文件可以快速应用于任何 Web 项目。压缩包中提供了编译版的 CSS 和 JS 文件（bootstrap.*），也同时提供了编译并压缩之后的 CSS 和 JS 文件（bootstrap.min.*）。图片文件是使用 ImageOptim 工具进行压缩的。注意，所有的 JavaScript 插件都依赖于 jQuery 库。

- bootstrap.css：完整的 Bootstrap 样式表，未经压缩过的，可供开发的时候进行调试使用。
- bootstrap.min.css：经过压缩后的 Bootstrap 样式表，内容和 bootstrap.css 完全一样，但是把中间不需要的东西都删掉了，如空格和注释，所以文件大小会比 bootstrap.css 小，可以在部署网站的时候引用，如果引用了这个文件，就没必要引用 bootstrap.css 了。
- bootstrap-responsive.css：在对 Bootstrap 框架应用了响应式布局之后所需要的 CSS 样式表，如果网站项目不需要进行响应式设计，就不需要引用这个 CSS。
- bootstrap-responsive.min.css：与 bootstrap.min.css 的作用一样，是 bootstrap-responsive.css 的压缩版。
- bootstrap.js：Bootstrap 所有 JavaScript 指令的集合，也是 Bootstrap 的灵魂，用户看到 Bootstrap 中所有的 JavaScript 效果，都是由这个文件控制的。这个文件也是一个未经压

缩的版本，供开发的时候进行调试使用。

- bootstrap.min.js：bootstrap.js 的压缩版，内容和 bootstrap.js 一样，但是文件会小很多，在部署网站的时候就可以不引用 bootstrap.js，转而引用这个文件。

2.3　Bootstrap 应用解析

把 Bootstrap 压缩包下载到本地之后，就可以安装使用了。本节不仅介绍如何正确安装 Bootstrap 工具集，同时介绍 Bootstrap 架构组成，这个架构有什么功能，能够为网页设计和开发带来哪些用处。最后，引导读者创建一个符合 Bootstrap 技术要求的标准模板，这样读者就可以利用这个模板页面上机练习了。

2.3.1　安装 Bootstrap

Bootstrap 安装大致需要以下两步。

第 1 步：安装 Bootstrap 的基本样式。样式的安装有多种方法，下面代码使用 <link> 标签调用 CSS 样式，这是一种常用的调用样式方法。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>test</title>
<link href="bootstrap/css/bootstrap.css" type="text/css">
<link href="bootstrap/css/bootstrap-responsive.css" type="text/css">
<link href="bootstrap/css/self.css" type="text/css">
</head>
<body>
</body>
</html>
```

其中 bootstrap.css 是 Bootstrap 的基本样式，bootstrap-responsive.css 是响应式布局样式，self.css 是本文档自定义样式。

注意　这里有两个关键点，其中 bootstrap.css 是 Bootstrap 框架集中的基本样式文件，只要应用 Bootstrap，就必须调用这个文件。而 bootstrap-responsive.css 则可以根据需要选择性安置，如果想让项目具有响应式布局的效果，就必须调用这个样式文件。调用必须遵循先后顺序，bootstrap-responsive.css 必须置于 bootstrap.css 之后，否则就不具有响应式布局功能。最后，self.css 是项目中的自定义样式，用来覆盖 Bootstrap 中的一些默认设置，便于开发者定制本地样式。

第 2 步：CSS 样式安装完后，就可以进入 JavaScript 调用操作。方法很简单，仅需把需要的 jQuery 插件源文件按照与上一步相似的方式加入到页面代码中。

调用 Bootstrap 的 jQuery 插件，代码如下。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>test</title>
<link href="bootstrap/css/bootstrap.css" type="text/css">
<link href="bootstrap/css/bootstrap-responsive.css" type="text/css">
<link href="bootstrap/css/self.css" type="text/css">
</head>
<body>
<!-- 文档内容 -->
<script src="http://code.jquery.com/jquery.js"></script>
<script src="bootstrap/js/bootstrap.js"></script>
</body>
</html>
```

其中 jquery.js 是 jQuery 库基础文件，bootstrap.js 是 Bootstrap 的 jQuery 插件源文件。建议将 JavaScript 脚本文件置于文档尾部，即相邻 </body> 标签的前面，不要置于 <head> 标签内。

2.3.2 Bootstrap 架构解析

Bootstrap 中的 HTML、CSS 和 JavaScript 适用于各类设备，如移动设备、平板电脑、PC 等，不过它们的功能可以概括成如下几个类别。

- 脚手架：全局性的样式文件，用于重置背景、链接样式、栅格系统等，并包含两个简单的布局结构。
- 基本 CSS 样式：常用 HTML 元素样式，如排版、代码、表格、表单、按钮样式，还包括一个非常棒的图标集——Glyphicons。
- Bootstrap 组件：常用界面组件，如标签、导航、警告、页面标题的基本样式。
- JavaScript 插件：与 Bootstrap 组件类似，这些 JavaScript 插件用来实现工具提示（Tooltip）、弹出提示（Popover）、模态对话框（Modal）等具有交互性的组件。

Bootstrap 组件库和 JavaScript 插件集共同提供了以下网页应用元素：按钮组、按钮下拉菜单、用于导航的标签、列表、导航条、标签、图标、页眉和单位、缩略图、警告对话框、进度条、模态对话框、下拉项、工具提示、弹出提示、折叠、轮播、输入提示。

后面的章节将会详细介绍这些组件的细节。

2.3.3 设计 Bootstrap 网页模板

为了把读者的注意力完全放到使用 Bootstrap 上，本节先做一下准备工作。这里创建一个基本的 HTML 模板。

1）启动 Dreamweaver，新建 HTML5 文档。在菜单栏中选择“文件”→“新建”选项，打开“新建文档”对话框。在对话框中选择“空白页”选项卡，在“页面类型”中选择 HTML，在“布局”中选择“无”，然后在“文档类型”下拉列表中选择 HTML5 选项，最后单击“创建”按钮，完成 HTML5 文档的创建过程，如图 2-13 所示。

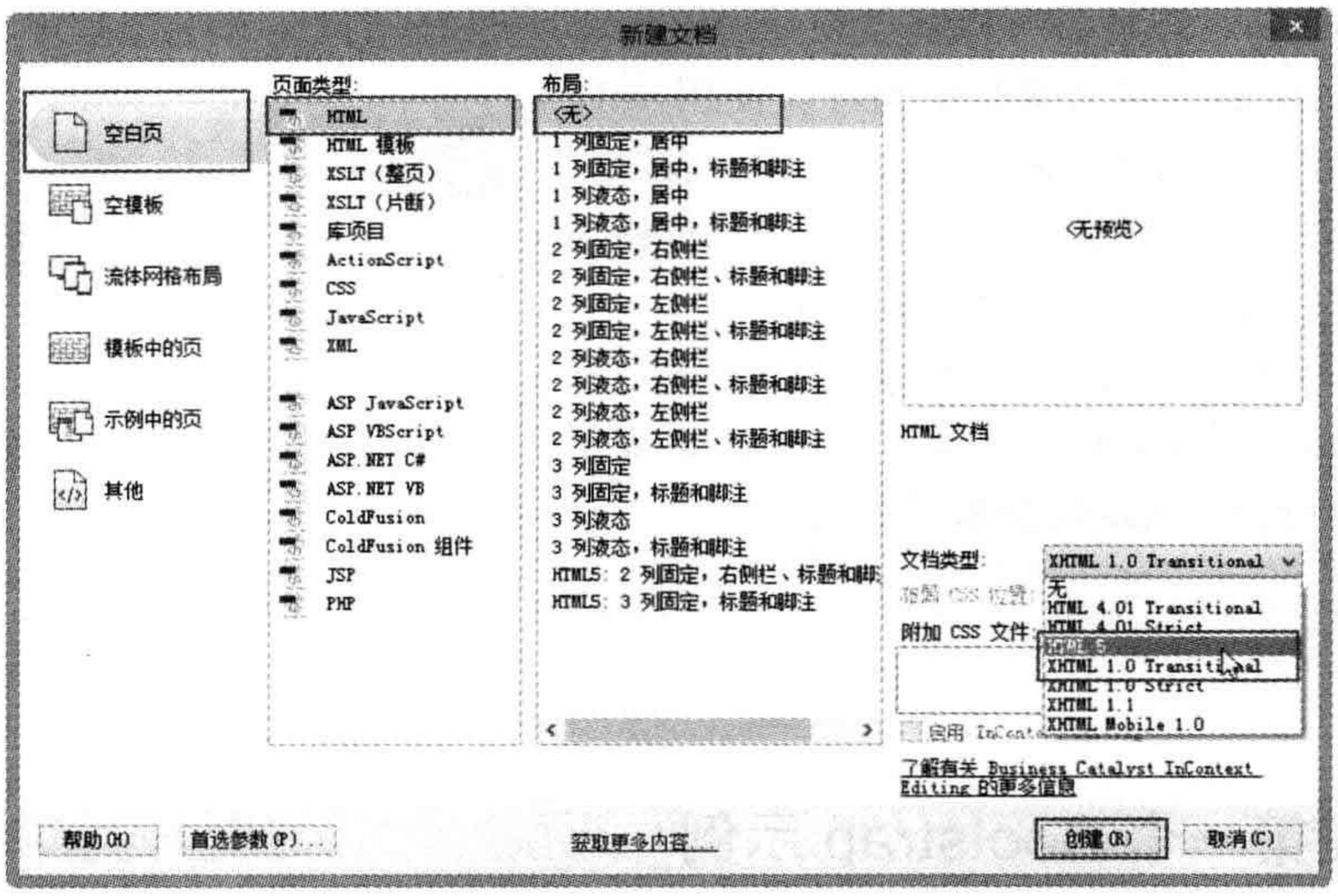


图 2-13　创建新的 HTML5 文档

2）保存为 index.html。在编辑窗口顶部的文档工具条中，设置网页标题为“Bootstrap 应用模板”。切换到代码视图，可以看到 HTML5 文档结构与 HTML4 文档结构有很大区别：代码简洁，不再严格遵循 HTML 语法规范。下面的代码展示了一个典型的 HTML 文件。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Bootstrap 应用模板 </title>
</head>
<body>
</body>
</html>
```

3）为了把页面设计为一个 Bootstrap 标准模板，需要包含相应的 CSS 和 JavaScript 文件。模板文档的详细代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Bootstrap 应用模板 </title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="bootstrap/css/bootstrap.min.css"  rel="stylesheet"  type="text/css">
</head>
<body>
    <script src="http://code.jquery.com/jquery.js"></script>
    <script src="bootstrap/js/bootstrap.min.js"></script>
</body>
</html>
```

4）设置成功，现在就可以开始使用 Bootstrap 开发任何网站和应用程序了。我们在页面中输入一行信息，与大家打个招呼。使用 <h1> 标签输出一句问候，在标签类样式中，btn 表示把 <h1> 标签定义为按钮样式，btn-success 表示按钮为成功类型样式，btn-large 表示大按钮效果。

图 2-14　设计第一个案例效果

在 <h1> 标签中包含一个 <i> 标签，用来定义图标，类 icon-user 表示用户图标，icon-white 表示图标的颜色为白色，演示效果如图 2-14 所示。

```
<h1 class="btn btn-success btn-large"><i class="icon-user icon-white"></i>
 Hello, world!</h1>
```

2.4　开发第一个 Bootstrap 示例

本节不详细分析源码文件，因为读者可以随时下载它们。这里通过几个示例介绍使用已经编译好的 Bootstrap 文件进行入门练习。

2.4.1　设计交互组件

Tabs 是页面中使用频率比较高的组件之一，要使用 Bootstrap 设计基本组件，必须满足 3 个条件：

- ❑ 正确设计最基本的 HTML 结构；
- ❑ 需要 Bootstrap 中的 jQuery 插件提供相应功能；
- ❑ 在项目中对应的 Tabs 元素上启用 Tabs 功能。

下面示例演示如何设计一个简单的 Tabs 效果，如图 2-15 所示。

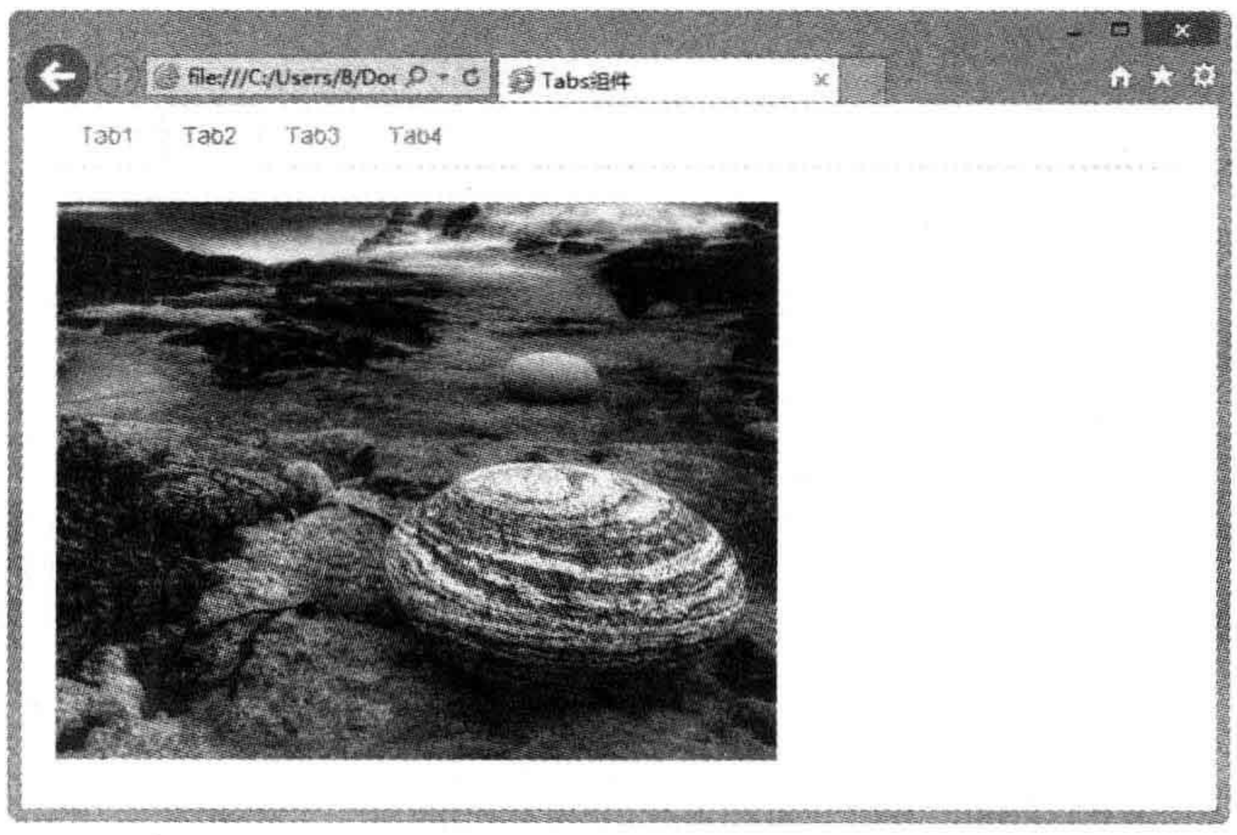


图 2-15　应用 Tabs 组件

第 1 步：利用 2.3.3 节介绍的方法完成页面基本结构创建。读者可以直接把 2.3.3 节设计的 Bootstrap 网页模板另存为 index.html，然后在页面中添加如下 Tabs 结构。

```
<ul  class="nav nav-tabs">
     <li class="active"><a href="#tab1" data-toggle="tab">Tab1</a></li>
     <li><a href="#tab2" data-toggle="tab">Tab2</a></li>
     <li><a href="#tab3" data-toggle="tab">Tab3</a></li>
     <li><a href="#tab4" data-toggle="tab">Tab4</a></li>
</ul>
<div class="tab-content">
     <div class="tab-pane active" id="tab1"><img src="images/1.png"></div>
     <div class="tab-pane" id="tab2"><img src="images/2.png"></div>
     <div class="tab-pane" id="tab3"><img src="images/3.png"></div>
     <div class="tab-pane" id="tab4"><img src="images/4.png"></div>
</div>
```

在上面结构中，类 nav 清除列表的默认样式，类 nav-tabs 定义 Tabs 标题栏。类 tab-content 定义 Tabs 组件的内容框。在内容框中，每个子框都必须包含 tab-pane 类。

通过在标题栏超链接中定义 <a> 标签的 href 属性值，该值与内容框中每个框的 id 值相对应，实现标题项与子内容框绑定，并确保一一对应。

类 active 定义活动的 Tab 项。同时，应该为标题栏中每个 <a> 标签定义 data-toggle="tab" 属性声明。

第 2 步：完成第 1 步设计工作，基本的 Tabs 组件就可以工作了。但是如果需要设计更复杂的交互行为，还需要调用 jQuery 插件。在官网下载 bootstrap-tab.js 文件，并导入到页面中，放置于 bootstrap.js 文件的后面。

```
<script src="http://code.jquery.com/jquery.js"></script>
<script src="bootstrap/js/bootstrap.min.js"></script>
<script src="bootstrap/js/bootstrap-tab.js"></script>
```

第 3 步：自定义 JavaScript 代码，调用 Tabs 组件，开启 Tabs 功能，代码如下：

```
<script type="text/JavaScript">
$(function(){
     $('.tabs a:last').tab('show')
})
</script>
```

对于其他组件，使用方法与此相近，这里不再赘述。

2.4.2　设计页面版式

本节借助 Bootstrap 布局版式设计一个完整的页面效果，页面版式模拟企业网站类型。企业网站一般都遵循基本的营销类设计格式，具有一个主消息板块和三个辅助性栏目，如图 2-16 所示。

图 2-16　设计企业网站布局版式

第 1 步：新建 HTML5 类型文档。根据 2.4.1 节介绍的方法引入 Bootstrap 库文件。

第 2 步：在 <body> 标签内完成页面基本框架的设计，代码如下。在浏览器中的预览效果如图 2-17 所示，此时 Bootstrap 对于文档并没有起到作用。

```
<div>
    <div>
        <h1> 联想控股 </h1>
        <p><img src="images/bg2.png"></p>
        <p><a href="#"> 更多 &raquo;</a></p>
    </div>
    <div>
        <div>
            <h2> 公司专题 </h2>
            <p>2012 年 12 月 2 日，联想之星创业大讲堂在常州举行，柳传志就“创业一把手的成长 "、
                “创业团队的建设”与创业者进行分享。</p>
        <p><a href="#"> 了解更多 &raquo;</a></p>
</div>
 <div>
        <h2> 特别关注 </h2>
        <p> 从靠“卖电脑”起家，到旗下集 IT、房地产、消费与现代服务、化工新材料、现代农业
            五大核心资产运营于一体，联想控股正冲刺在 2014 年 ~ 2016 年之间上市。</p>
        <p><a href="#"> 了解更多 &raquo;</a></p>
</div>
 <div>
        <h2> 我们的历史 </h2>
```

```
            <p><img src="images/bg1.png"></p>
            <p><a href="#"> 了解更多 &raquo;</a></p>
        </div>
    </div>
    <hr>
    <footer>
        <p>&copy; Company 2013</p>
    </footer>
</div>
```

图 2-17　页面初始设计效果

第 3 步：设计页面基本布局效果。在第一层 标签中引入 container 类样式，设计页面包含框宽度为 940 像素，居中显示；设计第二层中第 1 个 标签为焦点视图单元（hero-unit），通过 class 引入 hero-unit 类样式；设计第二层中第 2 个 标签为栅格布局框，通过 class 引入 row 类样式。

然后设计第三层 标签为栅格布局。分别为 布局包含框中三个 子标签引入 span4 类样式，即设计每列宽度为 228 像素。

增加布局类样式的结构代码如下，此时页面布局效果如图 2-18 所示。

```
<div class="container">
    <div class="hero-unit">
        <h1> 联想控股 </h1>
    </div>
    <div class="row">
        <div class="span4">
            <h2> 公司专题 </h2>
```

```
        </div>
        <div class="span4">
            <h2> 特别关注 </h2>
        </div>
        <div class="span4">
            <h2> 我们的历史 </h2>
        </div>
    </div>
</div>
```

图 2-18　页面初始设计效果

第 4 步：完成页面细节设计。为超链接 <a> 标签引入按钮类样式，如 <a class="btn" href="#">。

第 5 步：自定义页面样式，对 Bootstrap 布局效果进行适当修饰，主要是重写了焦点视图包含框 的样式，修改宽度，设置高度，清除边界和补白的值，使用灰色边框覆盖默认的红色边框线，增加定义相对定位，设计为定位包含框。

然后隐藏焦点视图框内的一级标题，绝对定位广告文本和导航按钮，详细代码如下：

```
<style type="text/css">
div.hero-unit { /* 重写焦点视图框样式 */
    background: url(images/bg.png) no-repeat;  /* 设计背景图 Banner 效果 */
    height: 443px;                             /* 固定高度显示，方便显示背景图 Banner*/
    width: 980px;                              /* 覆盖默认值 940 像素 */
    position: relative;                        /* 设计定位包含框，以方便内部定位 */
    padding: 0;                                /* 清除默认值 60 像素 */
    margin: 0;                                 /* 清除 margin-bottom 默认值 30 像素 */
```

```
        border-color: gray;                                    /* 重写默认值为 red 边框 */
}
div.hero-unit h1 {/* 隐藏标题 */
        display: none;
}
div.hero-unit .banner {/* 定位广告文本在左下角显示 */
        position: absolute;
        bottom: 0;
        left: 10px;
}
div.hero-unit .btn {/* 定位按钮在右下角显示 */
        position: absolute;
        bottom: 14px;
        right: 20px;
}
</style>
```

第 3 章

Bootstrap 框架解析

本章内容

- 设计全局样式表
- 栅格系统
- Bootstrap 布局
- 响应式设计

Bootstrap 框架主要是由动态 CSS 语言 LESS 编写，在很多方面类似于 Blueprint 框架（http://www.blueprintcss.org/）。经过 Node.js 编译后，Bootstrap 就是众多 CSS 的合集。Mark Otto 在 Twitter 开发官方博客表示，Bootstrap 用到了一些最新的浏览器技术，可以为开发人员提供精致的网页排版方式，这可以帮助加速项目开发，在一个完备的系统中拥有一致的设计和实现方法；开发者不需要在外观上花费过多时间，而能将精力集中于更重要的功能设计上。

Bootstrap 将改变前端开发合作方式与开发进程，任何人都可以基于 Bootstrap 建立可扩展的前端工具包，或者在它的基础上启动属于自己的框架。本章将就 Bootstrap 框架进行探索，以期帮助读者掌握该框架的基本实现方式和设计思路。

3.1　设计全局样式表

Bootstrap 为屏幕、排版和链接设置了基本的全局样式。本节只简单说明，详细介绍可以参阅 scaffolding.less 文件。

Bootstrap 使用的部分 HTML 元素和 CSS 属性需要文档类型为 HTML5 doctype，因此 <!doctype html> 文档类型必须出现在项目的每个页面开始部分。页面基本代码如下：

```
<!doctype html>
<html>
<head></head>
<body></body>
</html>
```

3.1.1　CSS 全局样式设计思路

Bootstrap 框架的核心是轻量的 CSS 基础代码库，没有一味地重置，而是注重各浏览器基础表现，降低开发难度。目前 90% 的 Reset（样式重置）都是归零的思想，但在开发过程中开发人员经常发现样式归零存在着潜在的问题。例如，在全局样式中将 strong 变成了一个普通标记，在用户可编辑内容区域的 strong 就不会有效果，用户就会很疑惑，为什么明明在编辑器中加粗了字体，但实际上却没有显示出来呢。

Bootstrap 只重置掉可能产生问题的样式（如 body、form 的默认 margin 等），保留和坚持部分浏览器的基础样式，解决部分潜在的问题，提升一些细节体验，具体说明如下。

- 移除 body 的 margin 声明。
- 设置 body 的背景颜色为白色，如 background-color: white;。
- 使用 @baseFontFamily、@baseFontSize 和 @baseLineHeight 属性作为排版的基础使用。
- 通过 @linkColor 设置全局链接颜色，且当链接处于 :hover 状态时才会显示下划线样式。

从 Bootstrap 2.0 开始，老的重置方式被 normalize.css 取代，虽然在 reset.less 文件中使用了许多 Normalize 的代码，但是它移除了一些不适合 Bootstrap 的元素。

Bootstrap 框架在设计上遵循下面三点设计原则：

- 统一的基础表现
- 更低的开发难度
- 更好的用户体验

构建好的发布文件说明如下：

css/bootstrap.css	基础样式表
css/bootstrap.min.css	压缩后的基础样式表
css/bootstrap-responsive.css	弹性布局样式表
css/bootstrap-responsive.min.css	压缩后的弹性布局样式表

在 Bootstrap 开发版压缩包中，所有样式根据功能分类存储在不同的动态样式表文件中，说明如下：

less/accordion.less	折叠插件样式
less/alerts.less	警告框插件样式
less/bootstrap.less	Bootstrap 动态样式表
less/breadcrumbs.less	面包屑组件样式
less/button-groups.less	按钮组样式
less/buttons.less	按钮样式
less/carousel.less	轮播插件样式
less/close.less	关闭按钮样式
less/code.less	代码预览格式
less/component-animations.less	组件动画样式
less/dropdowns.less	下拉项样式
less/forms.less	表单样式
less/grid.less	栅格系统
less/hero-unit.less	Hero 单元样式
less/labels-badges.less	标签徽章样式
less/layouts.less	布局样式
less/media.less	多媒体样式
less/mixins.less	混合类样式（设计特定结构下的嵌套类样式）
less/modals.less	模态对话框样式
less/navbar.less	导航条样式
less/navs.less	导航样式
less/pager.less	翻页组件样式
less/pagination.less	分页组件样式
less/popovers.less	弹出提示插件样式

less/progress-bars.less	进度条组件样式
less/reset.less	重置标签样式
less/responsive-1200px-min.less	动态响应最小 1200 像素宽度屏幕样式
less/responsive-767px-max.less	动态响应最大 767 像素宽度屏幕样式
less/responsive-768px-979px.less	动态响应 768 到 979 像素宽屏幕样式
less/responsive-navbar.less	动态响应导航条样式
less/responsive-utilities.less	动态响应公共样式
less/responsive.less	动态响应样式
less/scaffolding.less	脚手架样式（页面框架样式）
less/sprites.less	图标样式
less/tables.less	表格样式
less/thumbnails.less	缩略图组件样式
less/tooltip.less	工具提示插件动态样式
less/type.less	标签类型动态样式
less/utilities.less	公用类样式
less/variables.less	Bootstrap 动态样式变量表
less/wells.less	Wells 组件效果

有关 CSS 动态样式的详细讲解请参阅第 5 章。

3.1.2　CSS 规范和样式重用

合理而严谨的 CSS 设计，将使 CSS 代码更易于维护和重用，从而提升执行效率。一般应先规划样式，并严格区分公共样式和页面对象化样式；然后开始编码，在编码的同时进行调试、验证和代码片断的总结，而不是在所有模板都完成后才进行调试和代码整理。

1. Selector 命名规范

一般情况下，Bootstrap 选择器命名比较通用，如 .btn、.input、.table 等，这些类名都遵循对象化和语义化，符合用户使用习惯。当类名发生冲突，或者为了适应不同环境样式时，应该限定类的上下文环境。例如，针对 .btn 类型就定义了多个上下文环境，下面三个选择器分别适用于按钮、工具条、按钮组这 3 个不同的组件环境。

- button.btn
- btn-toolbar > .btn + .btn
- btn-group > .btn

同时，Bootstrap 样式类通过连字符后缀对一级类型进行细化。例如，针对 .btn 类样式，可以细分出很多子类样式。这种命名方法有两个好处：一是相同的前缀更方便类型管理；另一个是方便快速检索和应用。

.btn-large	大型按钮
.btn-small	小型按钮
.btn-mini	迷你型按钮
.btn-block	按钮块
.btn-primary	主要按钮
.btn-warning	警告按钮
.btn-danger	危险按钮
.btn-success	成功按钮
.btn-info	信息按钮
.btn-inverse	反转按钮

注意，只有在非常明确、不会影响到其他组件工作，并且其他人不会写这种命名的情况下，才让它变成全局通用的。

2. 编码规范

CSS 文件编码全部使用 UTF-8，因此建议网页编码也应设置为 UTF-8，确保编码一致性。在导入 CSS 文件时，应该明确定义 rel 和 type 声明，代码如下：

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

3. Hack 规则

Bootstrap 不使用 IE 条件注释：<!–[if IE]><![endif]–>。一般情况下，通过通用 /Hack 来解决浏览器兼容问题：

```
.all-IE{property:value\9;}
:root .IE-9{property:value\0/;}
.gte-IE-8{property:value\0;}
.lte-IE-7{*property:value;}
.IE-7{+property:value;}
.IE-6{_property:value;}
.not-IE{property//:value;}
@-moz-document url-prefix() { .firefox{property:value;} }
@media all and (-webkit-min-device-pixel-ratio:0) { .webkit{property:value;} }
@media all and (-webkit-min-device-pixel-ratio:10000),
not all and (-webkit-min-device-pixel-ratio:0) { .opera{property:value;} }
@media screen and (max-device-width: 480px) { .iphone-or-mobile-s-webkit{property:value;}}
```

当然，在自定义 CSS 样式时，强烈建议使用更优雅的 Hack 方式，那就是避免 Hack。或者在书写上，用点小技巧。例如：

```
.selector .child{property:value;} /* for ie-6 */
.selector > .child{property:value;} /* except ie-6 */
```

4. CSS3 书写规范

Bootstrap 遵循浏览器私有写法在前，标准写法在后，例如：

```
-webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
        box-sizing: border-box;
```

不强制书写顺序。但用户应该养成良好的习惯，让看代码的人更易理解。易读对于团队协作来说是非常重要的。

框架为先，细节次之。如写一个浮动容器的样式，应该先让这个容器的框架被渲染出来，先看到基本的网站框架。然后再去渲染容器里面的内容。最终呈现给用户。像 color、font、padding 之类的声明通常写在后面。

有因才有果。例如，想使用图片替换文字技术，通常要使用 text-indent。如果使用 标签： 这个文字将被图片替换 ，应该是先将 变成块级元素（display: block），再将文字隐藏（indent）。

```
.thepic{
      display: block;
      text-indent: -9999em;
}
```

3.1.3　CSS 重设

设计师们都梦想着这样一个完美世界：所有的浏览器都能够理解和适用所有 CSS 规则，并且呈现相同的视觉效果（没有兼容性问题）。但是，现实却总是恰恰相反，很多 CSS 样式在不同的浏览器中有着不同的解释和呈现。

当今流行的浏览器，如 Firefox、Opera、Internet Explorer、Chrome、Safari 等，有一些以自己的方式去理解 CSS 规范，这就会导致有的浏览器对 CSS 的解释与设计师的 CSS 定义初衷相冲突，使得网页的样子在某些浏览器下能正确按照设计师的想法显示，但有些浏览器却并没有按照设计师想要的样子显示出来，这就导致浏览器的兼容性问题。更糟的是，有的浏览器完全无视 CSS 的一些声明和属性。

正因为上述冲突和问题依然存在，所以一些设计师想到了一种避免浏览器兼容性问题的方法，那就是 CSS Reset。什么是 CSS Reset 呢？我们可以把它叫做“ CSS 重设”，也有人把它叫做“ CSS 复位”、“默认 CSS”、“ CSS 重置” 等。CSS 重设就是由于各种浏览器解释 CSS 样式的初始值有所不同，导致设计师在没有定义某个 CSS 属性时，不同的浏览器会按照自己的默认值来为没有定义的样式赋值，所以我们要先定义好一些 CSS 样式，来让所有浏览器都按照同样的规则解释 CSS，这样就能避免发生这种问题。

最简单的 CSS 重设如下：

```
* {
    padding: 0;
```

```
    margin: 0;
}
```

这个 CSS 重设将所有元素的 padding 和 margin 值都设为 0，可以避免一些浏览器在理解这两个属性默认值上发生分歧。

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

这是在上一个重设的基础上添加了对 border 属性的重设，初始值为 0，的确能避免一些问题。

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

在前两个重设的基础上添加了 outline 属性的重设，防止一些冲突。甚至还有如下的浓缩实用型 CSS 重设。


```
* {
    vertical-align: baseline;
    font-weight: inherit;
    font-family: inherit;
    font-style: inherit;
    font-size: 100%;
    outline: 0;
    padding: 0;
    margin: 0;
    border: 0;
}
```

该 CSS 重设方法出自在线资源网站 Perishable Press，这是它常用的方法。

因为这种做法具有巨大的潜在破坏性，Bootstrap 没有采用上述常规做法，进行 CSS 样式重置，而是采用针对性重置和增强型修补。

所谓针对性重置，就是设置特定上下文环境来重设样式。例如，针对通用选择器的使用，Bootstrap 仅限制在打印媒体类型中使用：

```
@media print {
 * {
   text-shadow: none !important;
   color: #000 !important; // Black prints faster: h5bp.com/s
   background: transparent !important;
   box-shadow: none !important;
```

```
  }
}
```

有针对性的重置样式：

```
button,
input,
select,
textarea {
    margin: 0;
    font-size: 100%;
    vertical-align: middle;
}
```

通过下面方法，增强 HTML5 新标签的功能，让它们拥有块状显示的布局功能。

```
article, aside, details, figcaption, figure, footer, header, hgroup, nav, section {
    display: block;
}
```

也许不同 CSS 框架和网站的重设方法不同，但是它们都有共同点，大部分都是同一个目的，就是为了让更多的浏览器能显示同样的效果。有了这些 CSS 重设作为资料和参考，也许会对后期开发有所帮助，甚至提高效率。

3.2　栅格系统

栅格系统（Grid System），也称网格系统，是一种平面设计的方法和设计风格。它运用固定的格子设计版面布局，其风格工整简洁，很受平面设计师的欢迎，已成为今日出版物设计的主流风格之一。在网页设计中，有人参考传统栅格系统的设计方法，以规则的网格阵列来指导和规范网页中的版面布局以及信息分布。

对于网页设计来说，栅格系统的使用，不仅可以让网页的信息呈现更加美观易读，更具可用性。而且，对于前端开发来说，网页将更加灵活与规范。栅格系统在现在的网页设计中应用越来越多，并出现了很多以栅格系统为设计基础的技术框架。

3.2.1　网页栅格系统的设计技法

栅格系统更多的是一种布局思想，在实际使用时，根据具体需求选用合适的技术实现即可。需要注意的是，对于栅格的技术实现来说，太灵活未必是好事，适度灵活最难得。栅格搭好了页面框架，接下来很重要的事情是往里面添加内容模块。让内容模块规范化，让页面生成工业化，对大型站点来说，这是栅格系统最有商业价值的地方。

良好的布局应该具有下面几个特性：

- 具有一定的灵活性
- 等高

❑ 基于栅格

在网页设计中，如果把网页宽度平均切分为多个网格单元，每个单元之间预留一定的空隙，此时整个页面就如同一个栅格系统，如图 3-1 所示。

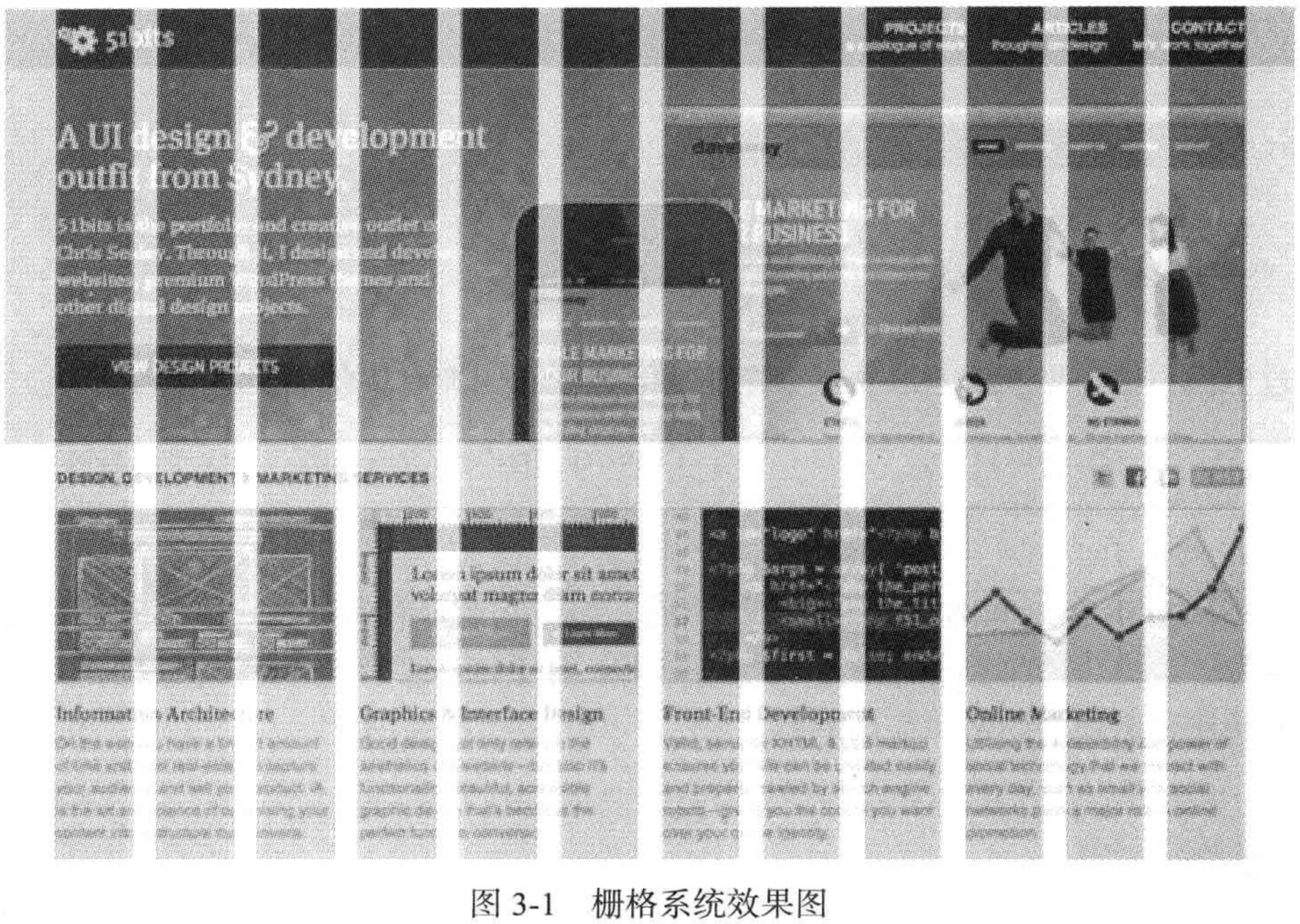


图 3-1　栅格系统效果图

如果把网页宽度设置为 W，页面分割成 n 个网格单元，每个单元的宽度设为 a，每个单元与单元之间的间隙设为 i，并把 $a+i$ 定义为 A（即一个单元的宽度），则可以定义等式：

$$W=a\times n+(n-1)\,i$$

由 $a+i=A$ 可得，$A\times n-i=W$，这个公式描述了网页布局与网页背后的栅格系统之间的某种关系，如图 3-2 所示。

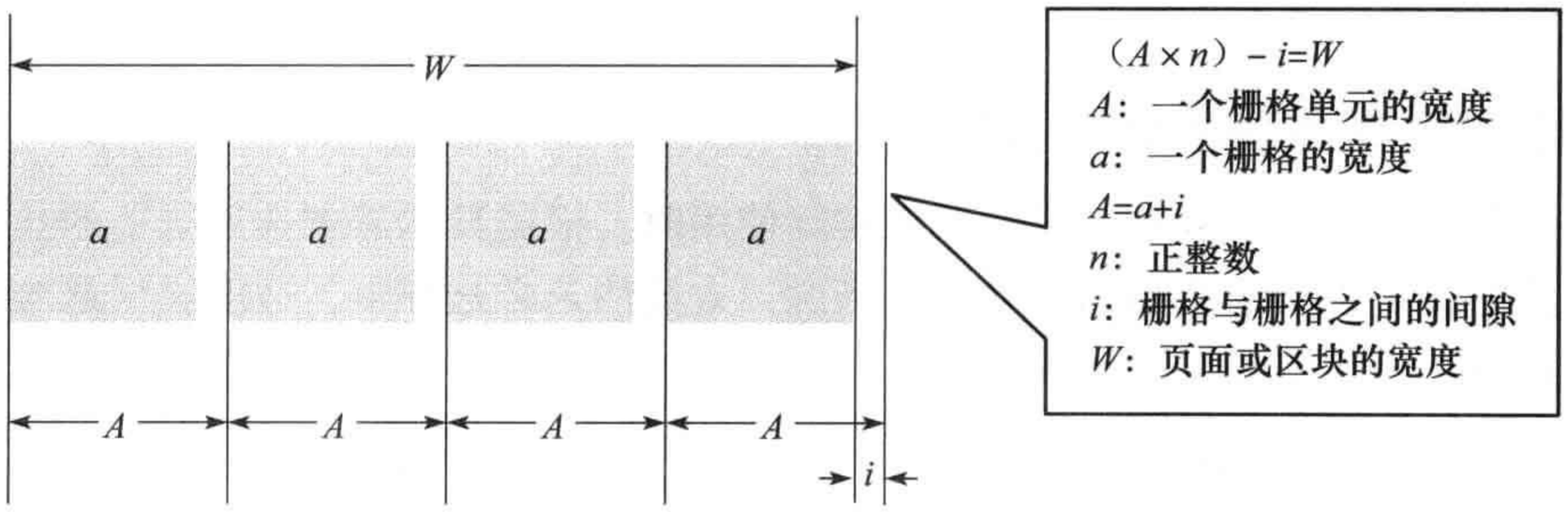


图 3-2　栅格系统示意图

例如，浏览 Yahoo! 主页（http://www.yahoo.com/），可以看到栅格系统的应用，如图 3-3 所示。

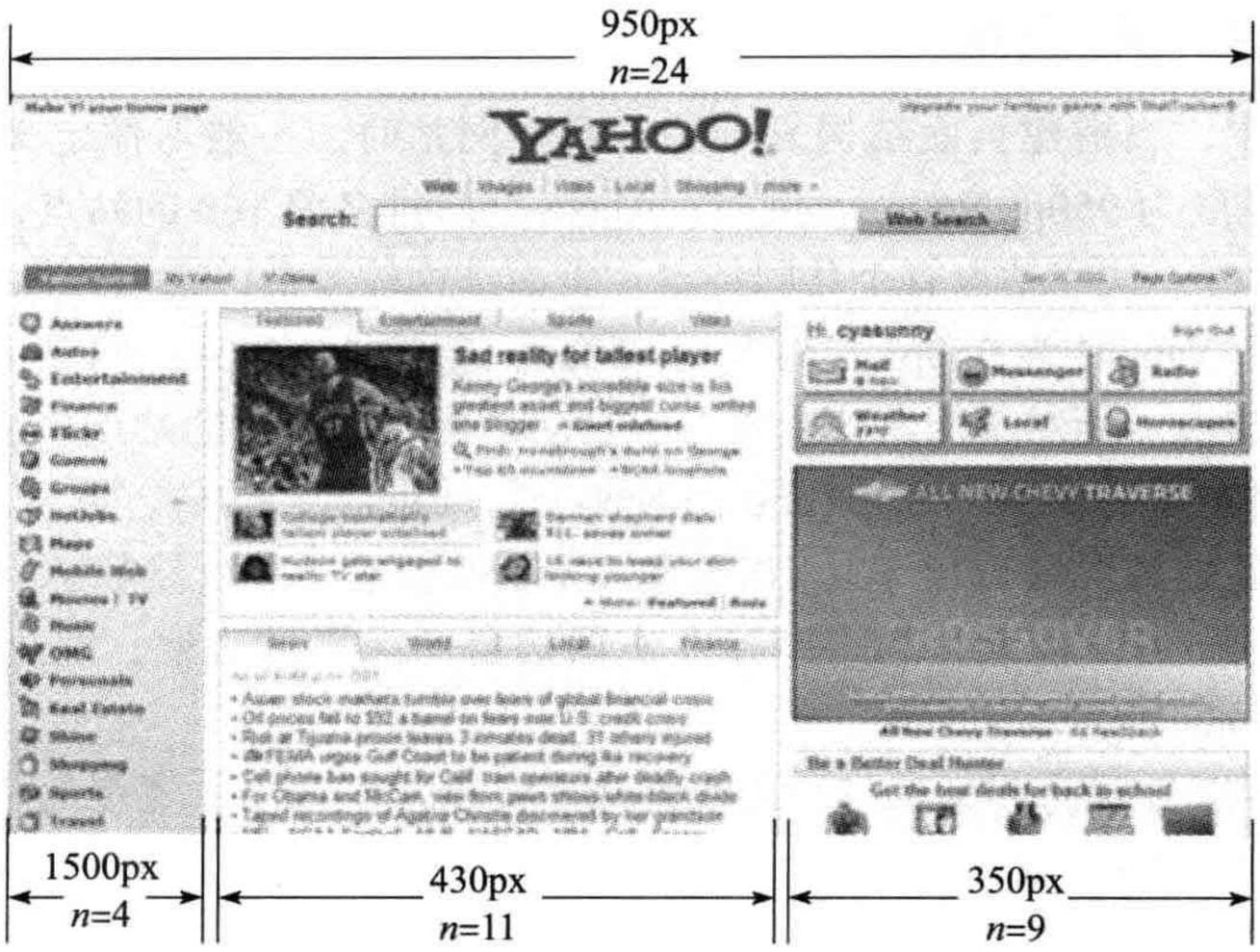


图 3-3　Yahoo! 主页的栅格系统示意图

Yahoo! 网站页面宽度为 W=950px，每个区块与区块的间隔为 i=10px。应用上面的公式，可以推出 A=40px，即 Yahoo! 主页横向版式设计采用的栅格系统为：

$$40 \times n - 10 = W$$

Yahoo! 主页的布局完全是按照栅格系统进行设计的，每个区块的宽度对应的 n 值分别为 4、11、9。因此只要保证一个横向维度的各个区块的 n 值相加等于 24，即可保证页面的宽度一定是 950px。然而，950px 的宽度也恰好就是当 n=24 的时候 W 的宽度值。由此可以得出以下的应用模式，如图 3-4 所示。

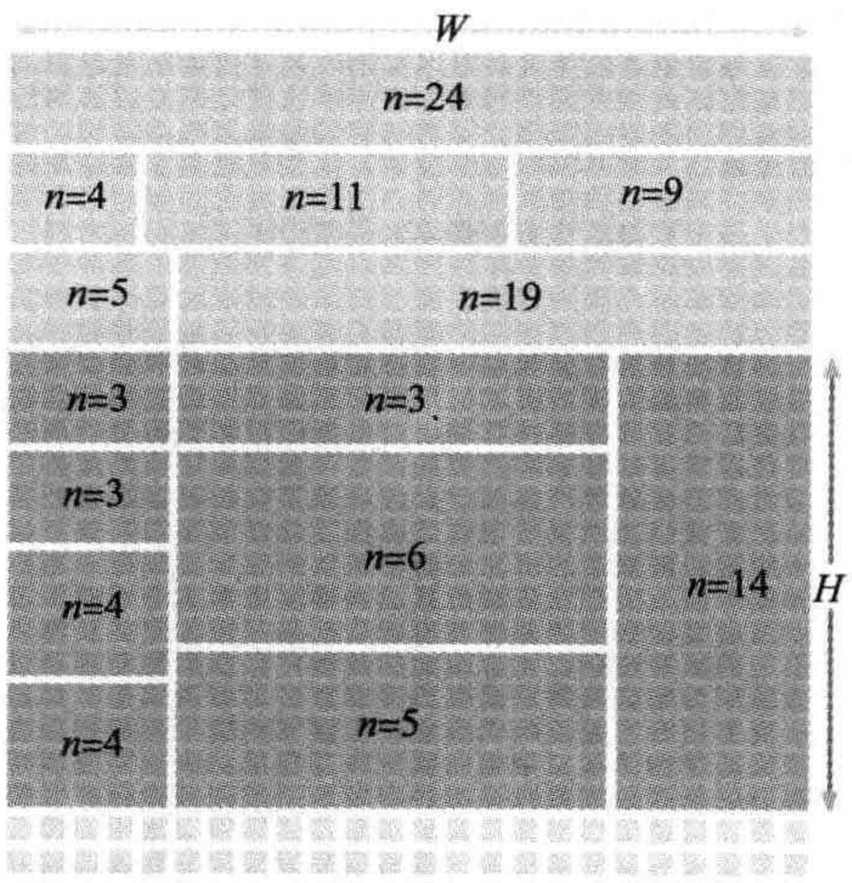


图 3-4　栅格系统与页面效果对应示意图

在栅格系统中，设计师根据需要制定不同的版式或者划分区块，这样一个栅格系统的应用就从此开始了。使用栅格系统的网页设计非常有条理，看上去也很舒服。最重要的是，它给整个网站的页面结构定义了一个标准。对于视觉设计师来说，不用再为网站的每个页面都想一个宽度或高度了。对于前端开发工程师来说，页面的布局设计将完全是规范的和可重用的，这将大大节约开发成本。对于内容编辑或广告销售来说，所有的广告都是规则的、通用的，再也不用做出一套多张不同尺寸的广告图了。

还也可以衍生出任何一种栅格系统，只需改变 A 和 i 的值（根据网站的实际情况而定）。当然，对于内容信息不确定，导致高度不确定的页面，在高度层面上就无法做到栅格。很多 CSS 库都会提供类似的栅格系统实现，如 YUI、BluePrint 等。

3.2.2 解析 960 栅格系统

在网页设计中，当搭建页面结构复杂的门户型网站时，一般习惯定义页面宽度固定宽度，同时多选择宽度为 950px/960px。例如，Alexa 全球排名前 100 的站点，它们的主页宽度多为 950px 或 960px，这些站点有个共同特点：页面结构较复杂，都可以认为是门户型网站，如 AOL 为 960px，Yahoo! 为 950px，淘宝为 950px，新浪为 950px，网易为 960px，搜狐为 950px，优酷为 960px。而对于网站功能专一、页面结构相对简单的站点来说，它们的主页宽度似乎没什么规律，如 Google、YouTube、Facebook、Flickr!、eBay、百度、新浪微博等。

大部分用户的电脑屏幕分辨率是 1024 像素 ×768 像素，在自然状态下，浏览器窗体的大小约为 974 像素 ×650 像素，如果减掉左右两边各 7 像素的边框，网页的实际大小约为 960 像素 ×650 像素。

另外，960 可以分解为 $2^6 \times 3 \times 5$，这使得 960 可以分割成以下宽度的整数倍，共 26 种：2、3、4、5、6、8、10、12、15、16、20、24、30、32、40、48、60、64、80、96、120、160、192、240、320、480。

目前绝大多数显示器都支持 1024 像素 ×768 像素及以上分辨率。为了有效地利用屏幕宽度，同时保证栅格的灵活度，可以看出 960 像素是非常合适的。这样，在目前主流显示器下，960 像素就成为网页栅格系统中的最佳宽度了。

不少设计师喜欢采用 960 固定宽度布局，当然对于结构复杂的网站来说，960 并不是万能钥匙，大部分网站没有也不需要栅格系统。Amazon 采用的是宽度自适应布局，最大限度地呈现信息。Google 更是简简单单，主题部分就一个列表。eBay 的页面也非常简洁，商品页面宽度自适应，信息自然流畅，购物很踏实。类似的站点还有很多，对于这些站点来说，宽度自适应布局更受青睐。

对于结构复杂的网站来说，可维护性和可扩展性非常重要。作为以信息展示为主的门户型网站，960 的宽度对于信息的阅读比较友善。

一个标准的栅格系统包括以下几个部分，如图 3-5 所示。

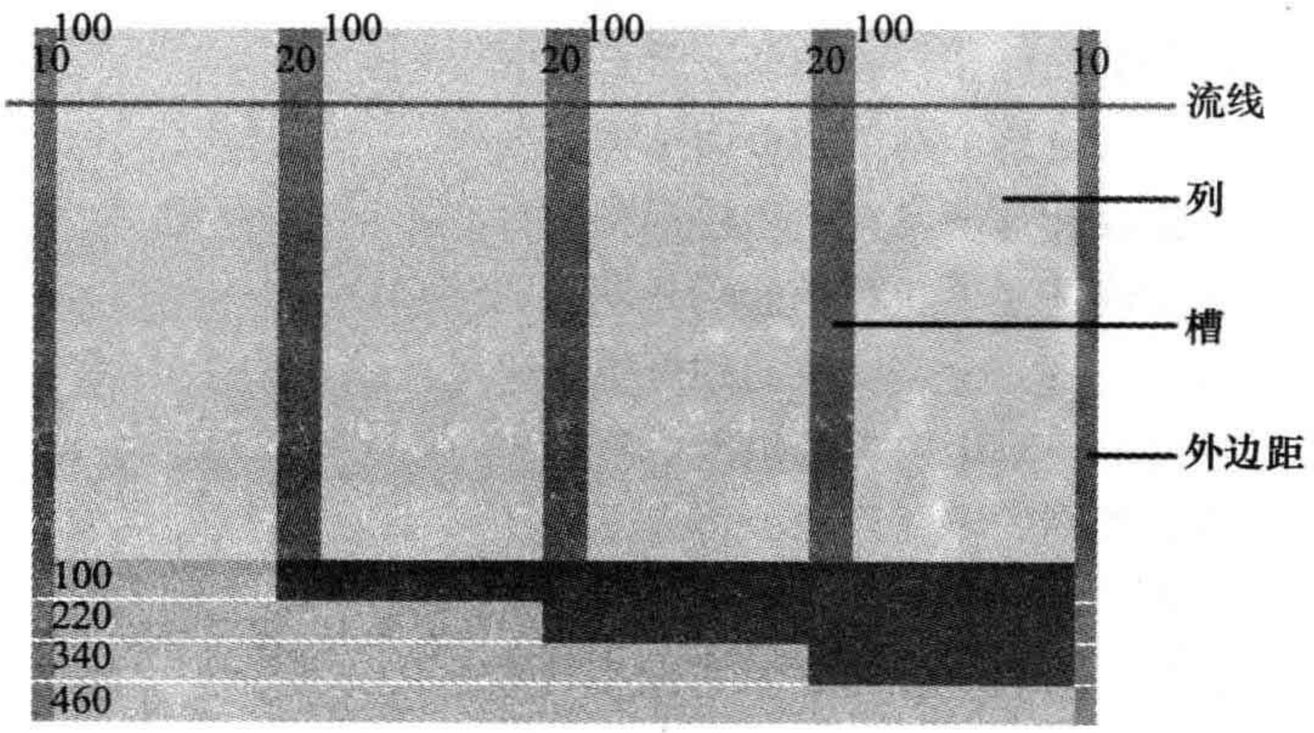


图 3-5 栅格系统构成

将流线（Flowline）的总宽度记为 W，列（Column）的宽度记为 c，槽（Gutter）的宽度记为 g，外边距（Margin）的宽度记为 m，列的个数记为 N，可以得到以下公式：

$$W=cN+g\ (N-1)\ +2m$$

一般来说，槽的宽度是外边距的两倍，上面的公式可以简化为：

$$W=cN+g\ (N-1)\ +g=(c+g)\ N$$

将 $c+g$ 记为，公式可进一步简化为：

$$W=CN$$

上面的公式就是栅格系统的基础。

在具体应用中，外边距其实是一个空白边，从视觉上看并不包含在总宽度内。不少栅格设计里习惯性地设定槽的宽度为 10px，这样外边距就是 5px. 当 W 为 960，分割成 6 列时，左右外边距各为 5px（也可以将外边距集中放在一边）。

无论外边距放在何处，如果去除外边距之后，将 W 的含义变为去除外边距的总宽度，公式变化为：

$$W=NC-g$$

将上面的公式实例化：

$$950=12\times80-10=16\times60-10=24\times40-10$$

这就形成了 960 栅格系统的 3 种常见切法，如图 3-6 ～图 3-8 所示。

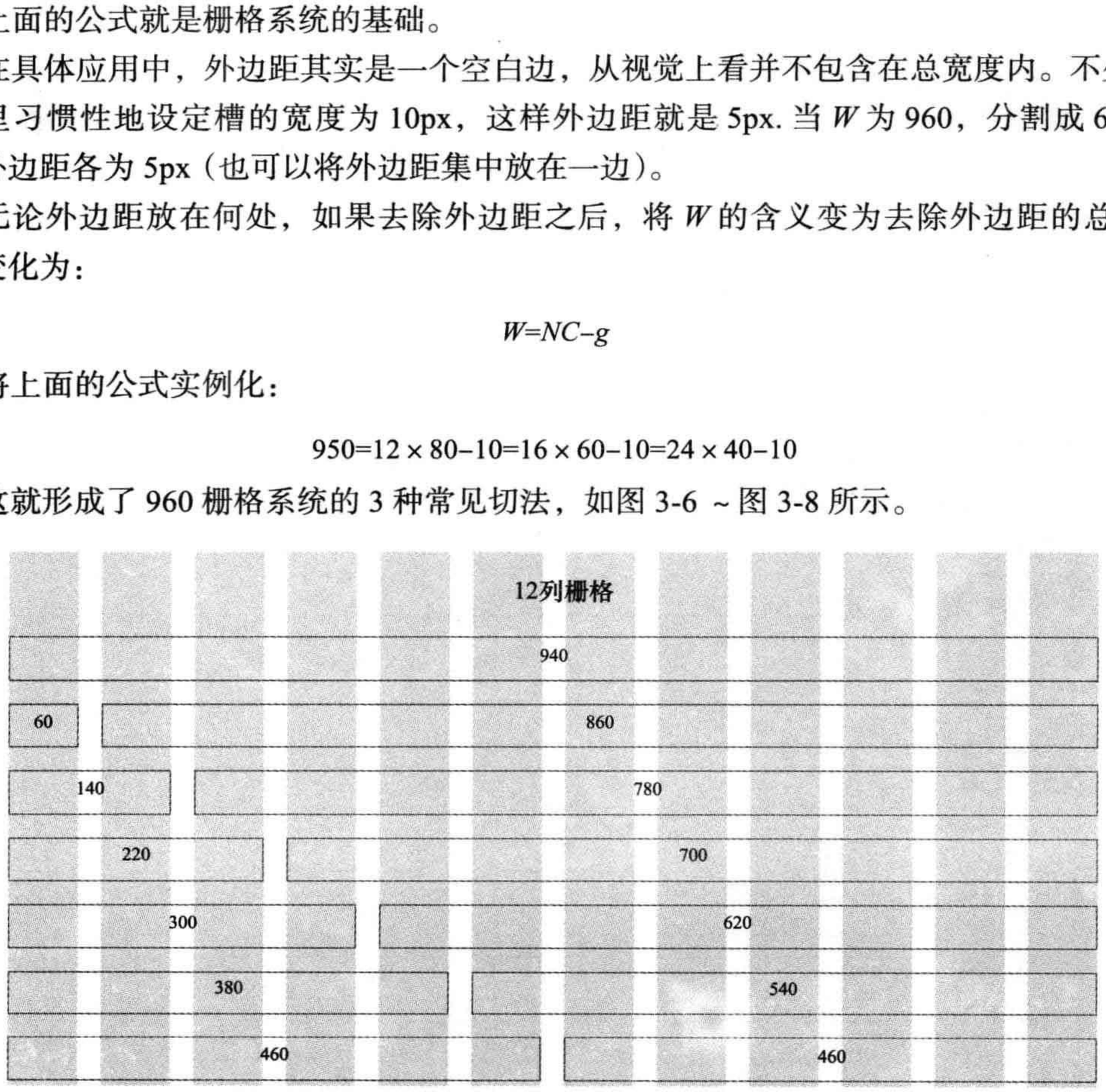


图 3-6　12 × 80 栅格系统切法

上面 3 种切法中，N 越大，灵活度越高。可以根据网页的实际复杂度来选用对应的切法。在 960Grid System 主页（http://960.gs/）中展示了 12 × 80 的应用（见图 3-9），其中左图网站地址为 http://interactionhero.com/，右图网站地址为 http://5by5.tv/。

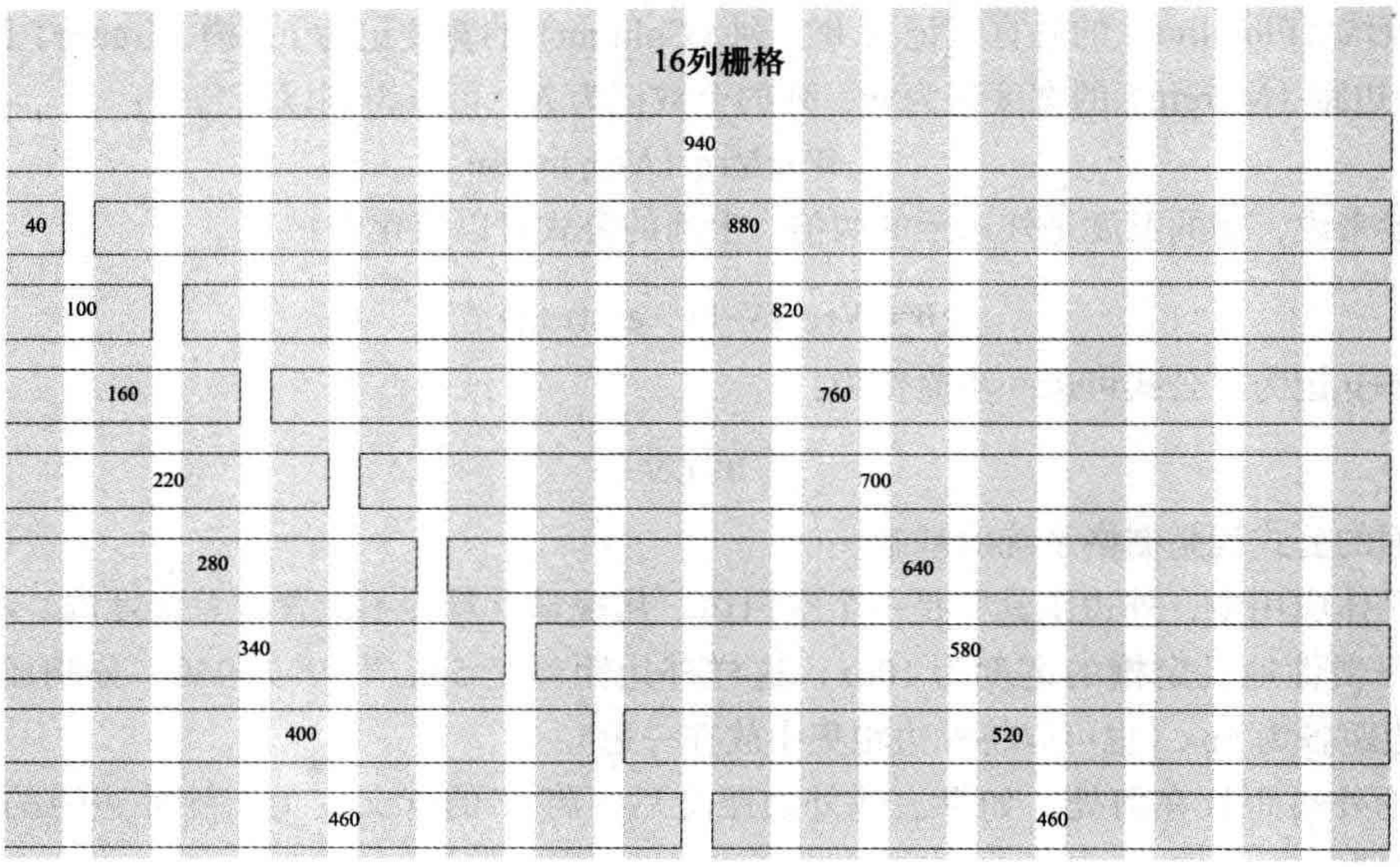


图 3-7　16 × 60 栅格系统切法

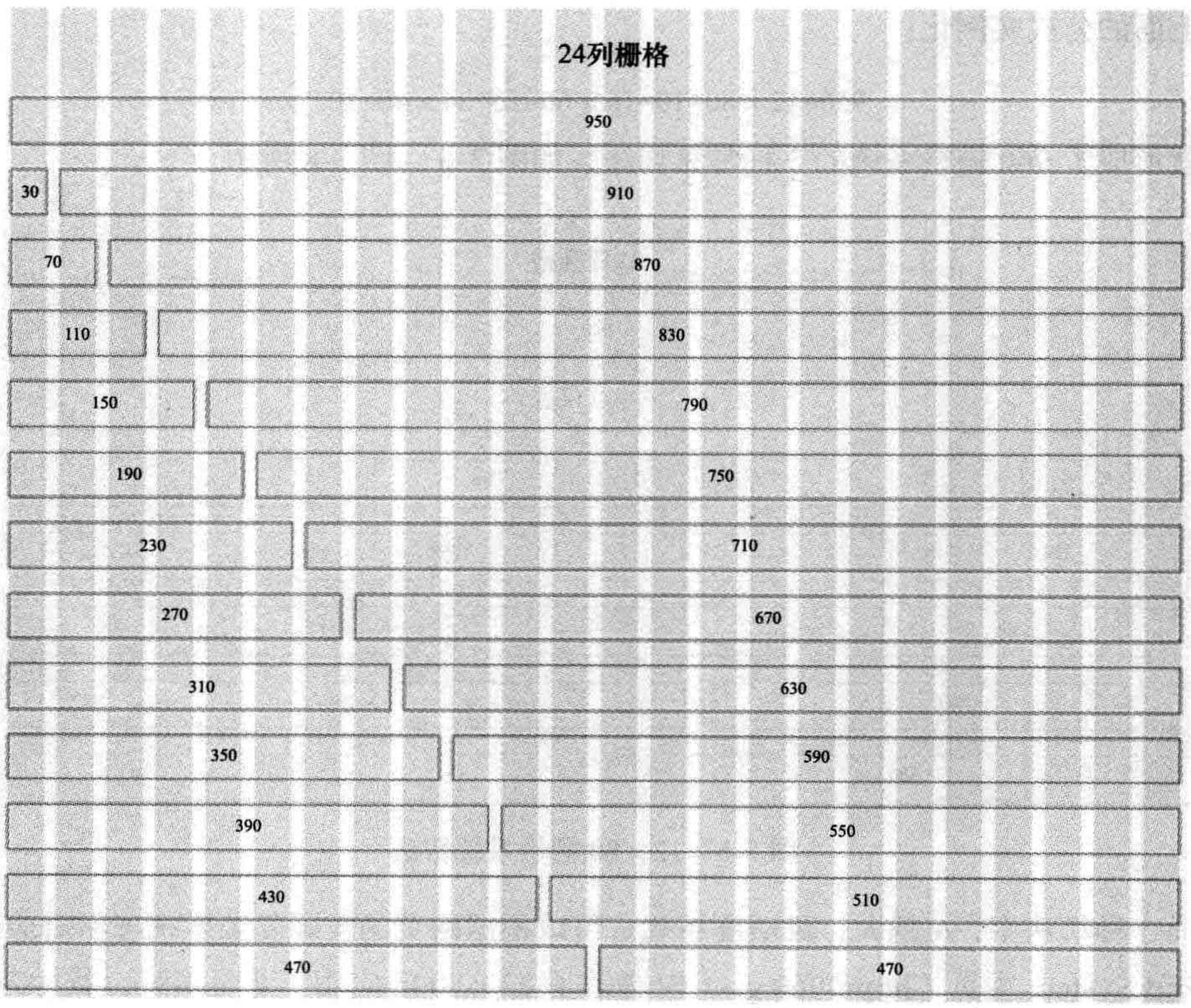


图 3-8　24 × 40 栅格系统切法

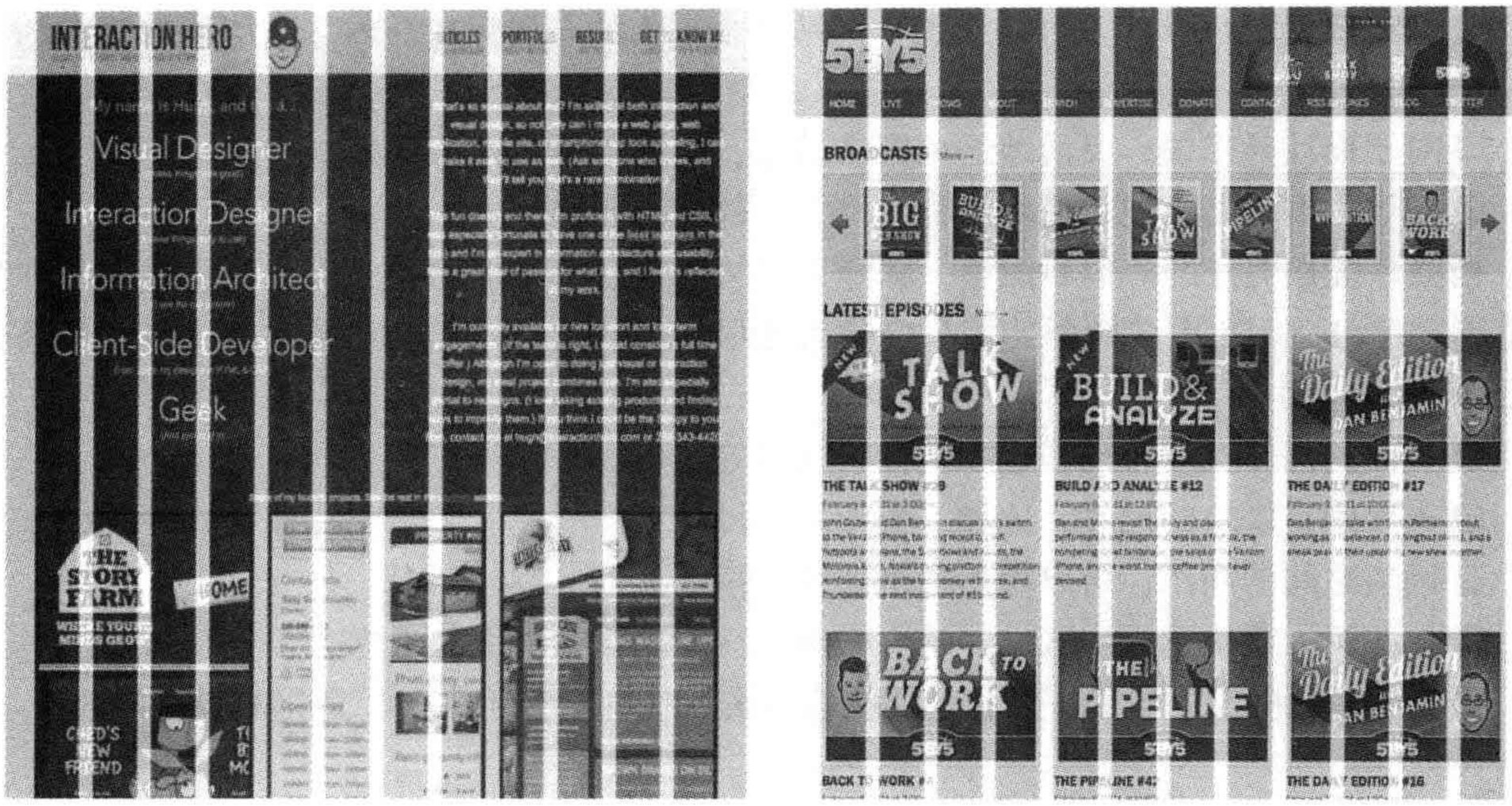


图 3-9　12 × 80 栅格系统切法实例

图 3-10 展示了 16 × 60 的应用，其中左图网站地址为 http://www.sonymusic.com/，右图网站地址为 http://fedoraproject.org/。

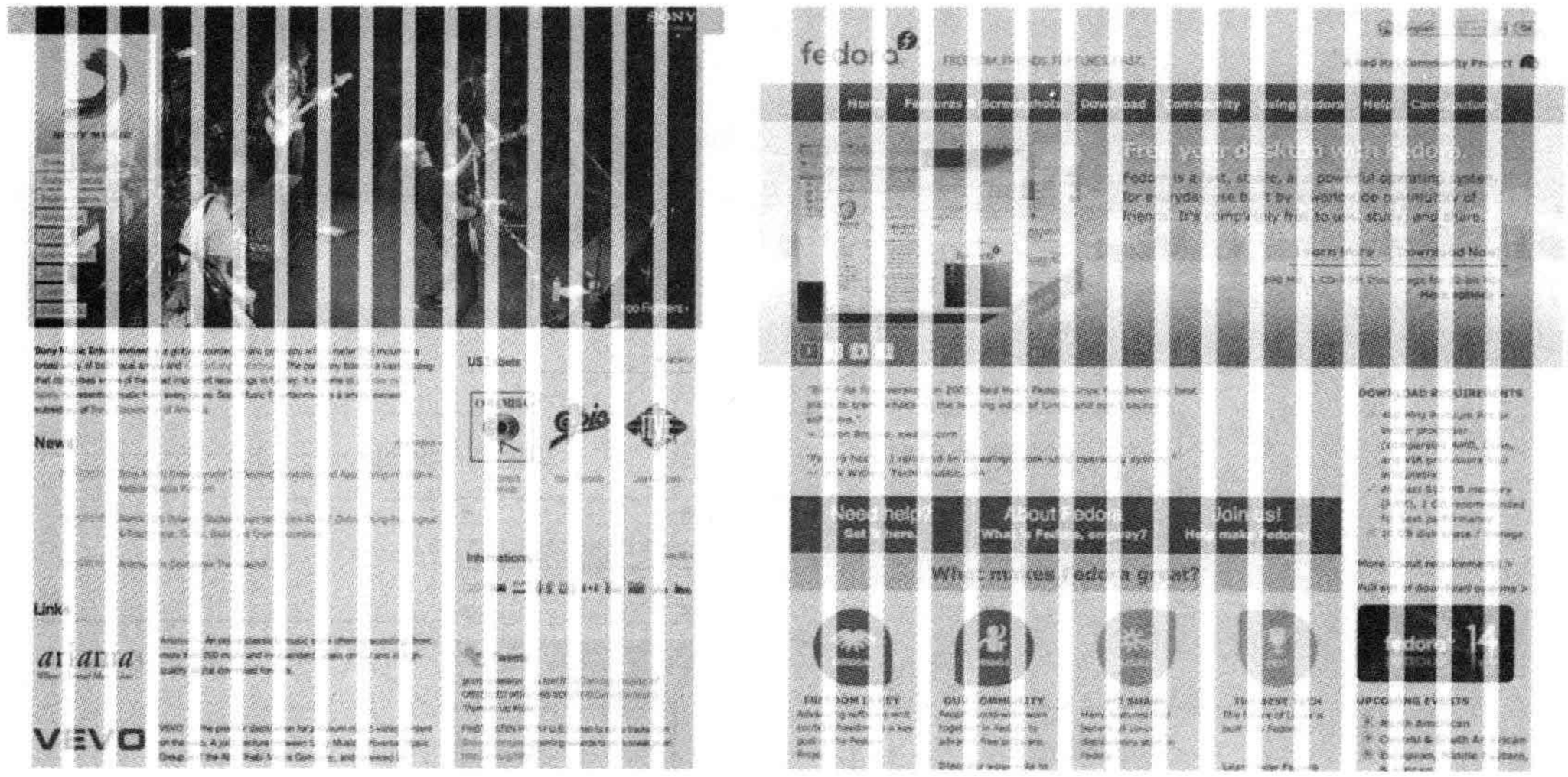


图 3-10　16 × 60 栅格系统切法实例

目前严格采用栅格系统的网站非常少，那么为什么我们还要努力学习并实践网页栅格化呢？这是因为栅格系统具有以下优势。

- 能大大提高网页的规范性。在栅格系统下，页面中所有组件的尺寸都是有规律的。这对于大型网站的开发和维护来说，能节约不少成本。

- 基于栅格进行设计，可以让整个网站各个页面的布局保持一致。这能增加页面的相似度，提升用户体验。
- 对于设计师来说，灵活地运用栅格系统，能做出很多优秀和独特的设计。
- 对于大型网站来说，使用栅格化将是一种潮流和趋势。

栅格系统是美好的。但不能够一味地追求将所有设计都栅格化，因为任何设计都有其适用范围，超出适用范围，勉强使用只会带来无尽的烦恼。对于栅格系统来说，以下场景非常适合：

- 页面的总体宽度布局，比如两栏、三栏等布局；
- 一些固定区块的尺寸，比如广告图片的尺寸；
- 区块之间的间距，可以参考栅格系统的槽宽；
- 一些可以栅格化的小区域，暗合栅格往往能简化布局上的考虑。

除了上面这些应用场景，勉强使用栅格系统，往往会束手束脚，适得其反。

3.2.3 Bootstrap 栅格系统

Blueprint（http://www.blueprintcss.org/）是个经典的 960 栅格系统，提供了完整的 CSS 框架，栅格系统是它的一部分功能。

Bootstrap默认的栅格系统为 12 列，宽度为 940px，比标准的 960 栅格系统少 20 像素，这是因为它少了一个外边距，一个外边距默认为 20 像素。因此，虽然宽度仅为 940 像素，但是网页实际宽度与 960 栅格系统相同。

Bootstrap 的栅格系统有两种，一种是固定式的（Fix），一种是流式的（Fluid），也就是可适应宽度的。固定式栅格系统每列的宽度及列与列间的间距都是固定的，列宽为 60px，列间距为 20px，如图 3-11 所示。

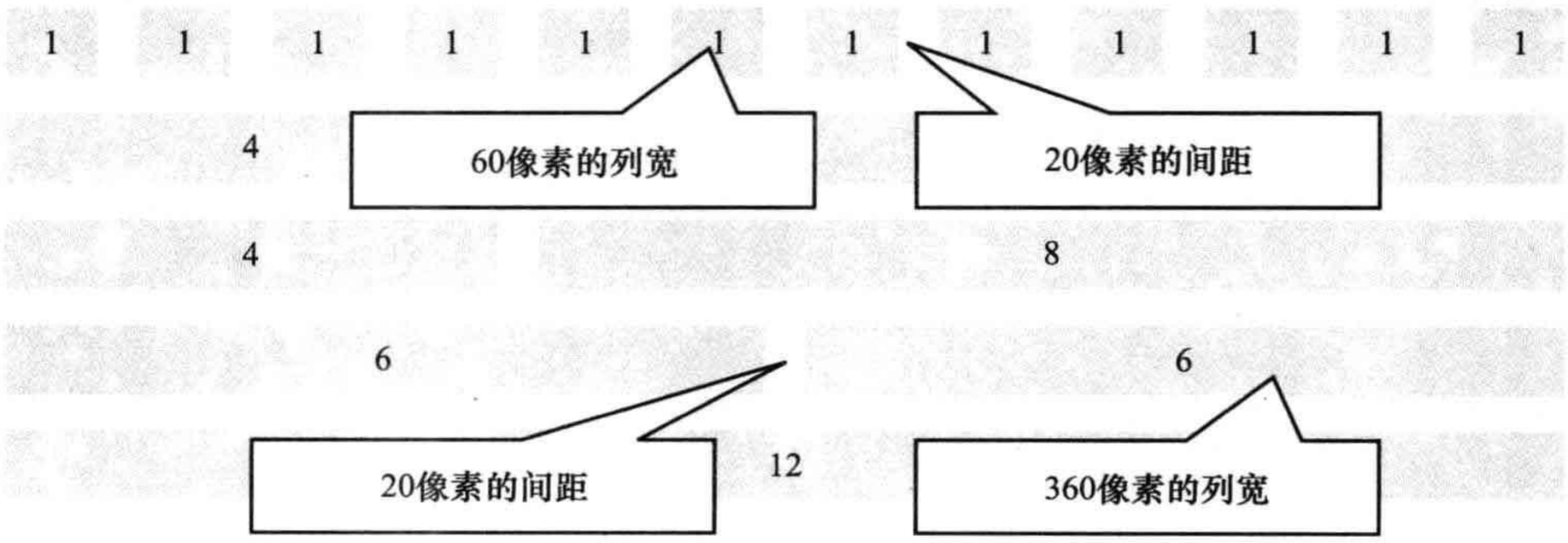


图 3-11 Bootstrap 固定宽度栅格系统

在 Bootstrap 中，为栅格系统定义了 2 类样式。容器为 .row，可在容器中加入合适数量的 .span* 列。在设计页面网格系统时，如果确定宽度为 940 像素，则应确保各列之和等于 12。具体用法如下：

```
<div class="row">
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
      <div class="span1">1</div>
</div>
```

或者

```
<div class="row">
      <div class="span3">3</div>
      <div class="span6">6</div>
      <div class="span3">3</div>
</div>
```

这种用法非常类似于 <table> 标签。 相当于 <table>， 和 分别相当于 "<td cols='3'>"、"<td cols='6'>"。注意，由于默认是 12 列的栅格，所有 .span* 列所跨越的栅格数之和最多是 12，或者等于其父容器的栅格数。

在默认情况下，其中栅格的类样式源代码如下：

```
[class*="span"] {
      float: left;
      min-height: 1px;
      margin-left: 20px;
}
```

上面是个属性选择器。定义属性名为 "class" 的值包含子串 "span"，也就是 span1、span2、span3 等。margin-left：20px 定义了每列的左边距为 20px，也就是图 3-11 中的间距 20px。

栅格系统的容器样式如下：

```
.row {
      margin-left: -20px;
      *zoom: 1;
}
.row:before,
.row:after {
      display: table;
      line-height: 0;
      content: "";
}
```

```
.row:after {
      clear: both;
}
```

.row 用来定义栅格容器，可以看到其中并没有定义宽度（width），所以其宽度就由内部栅格的总列宽决定。margin-left: –20px 定义容器的左边距为 –20px，作用是抵销第 1 列前面的 20px。然后，通过伪类选择器 .row: before 和 .row: after 在容器左右两侧各添加一个空表格，并定义表格内容为空，行高为 0，以便兼容 IE 等非标准浏览器在解析栅格系统时出现的异常，同时通过 .row：after{clear：both；} 样式清除浮动环绕问题。

```
.span12 { width: 940px; }
.span11 { width: 860px; }
.span10 { width: 780px; }
.span9 { width: 700px; }
.span8 { width: 620px; }
.span7 { width: 540px; }
.span6 { width: 460px; }
.span5 { width: 380px; }
.span4 { width: 300px; }
.span3 { width: 220px; }
.span2 { width: 140px; }
.span1 { width: 60px; }
```


span1 ~ span12 很像表格元素 <table> 中 <td> 的 cols，即合并多少列。span1 就是合并 1 列，即不合并，span2 就是合并 2 列，span3 是合并 3 列，以此类推。那为什么 span4 的宽度是 300px 呢?

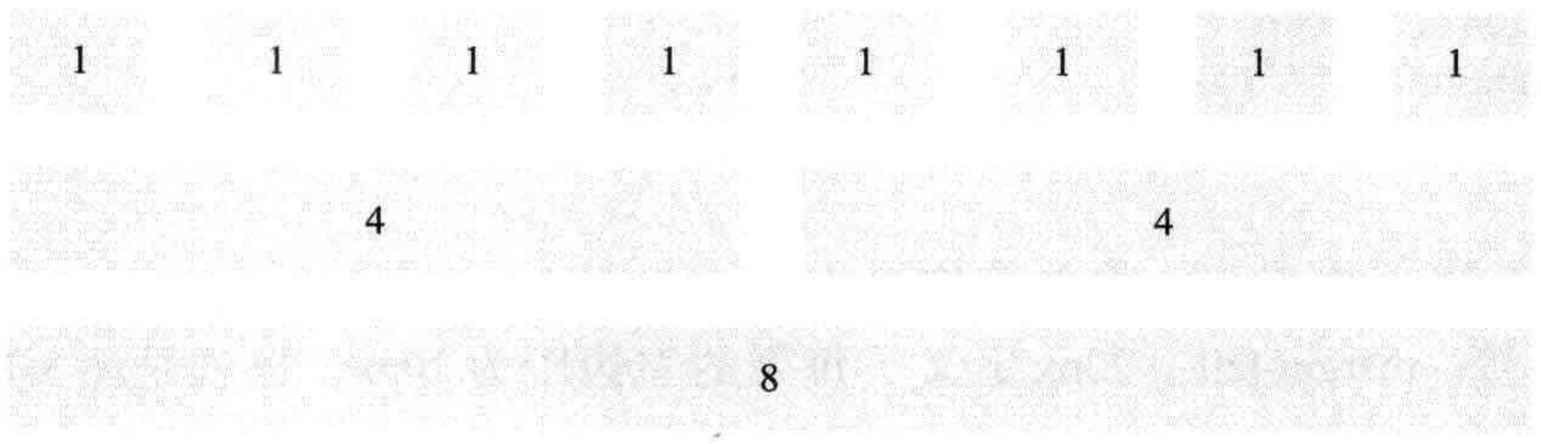


图 3-12　Bootstrap 栅格列对比

先看下 span4 样式，由于上面的属性选择器已经规定了 span1 ~ span12 的左边距都是 20px，可知 span4 自身有 20px 的左边距（margin-left: 20px）。因此对于上面图来说，span4 的宽度就等于 4 个 span1 宽度减去 20 像素，于是可以总结出这样的公式：span N 的宽度 $=N\times 60+(N-1)\times 20$。

把列向右移动可使用 .offset* 类。每个类都给列的左边距增加了指定单位的列。例如，.offset2 将 .span3 右移了 2 个列的宽度，如图 3-13 所示。

.offset* 类样式的源代码如下：

图 3-13　Bootstrap 栅格列偏移位置

```
.offset12 { margin-left: 980px; }
.offset11 { margin-left: 900px; }
.offset10 { margin-left: 820px; }
.offset9 { margin-left: 740px; }
.offset8 { margin-left: 660px; }
.offset7 { margin-left: 580px; }
.offset6 { margin-left: 500px; }
.offset5 { margin-left: 420px; }
.offset4 { margin-left: 340px; }
.offset3 { margin-left: 260px; }
.offset2 { margin-left: 180px; }
.offset1 { margin-left: 100px; }
```

.offset1 即移动 1 列，.offset2 移动 2 列。margin-left 是如何得出来的呢？以 .offset2 移动 2 列为例，图 3-13 演示了从第 2 列偏移 2 列到第 4 列。未移动前第 2 列的 margin-left 为 20px，移动后由于占了 3 个间距和 2 个列宽，最终的 margin-left 就是 180px。

注意　Bootstrap 栅格系统列宽和间距都是可以定制的，在 variables.less 文件中，可以通过下面几个变量进行定制。

```
// Default 940px grid
// -------------------------
@gridColumns: 12;
@gridColumnWidth: 60px;
@gridGutterWidth: 20px;
@gridRowWidth: (@gridColumns * @gridColumnWidth) + (@gridGutterWidth *
(@gridColumns - 1));
```

例如，在下面的示例中，利用 Bootstrap 栅格系统设计一个固定宽度的三列版式，代码如下，演示效果如图 3-14 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
```

```
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<style type="text/css">
.row div {
     background-color:gray;
     height:300px;
}
</style>
</head>
<body><div class="container">
<div class="hero-unit"></div>
<div class="row">
     <div class="span2"></div>
     <div class="span7"></div>
     <div class="span3"></div>
</div>
<footer></footer>
</div>
</body>
</html>
```

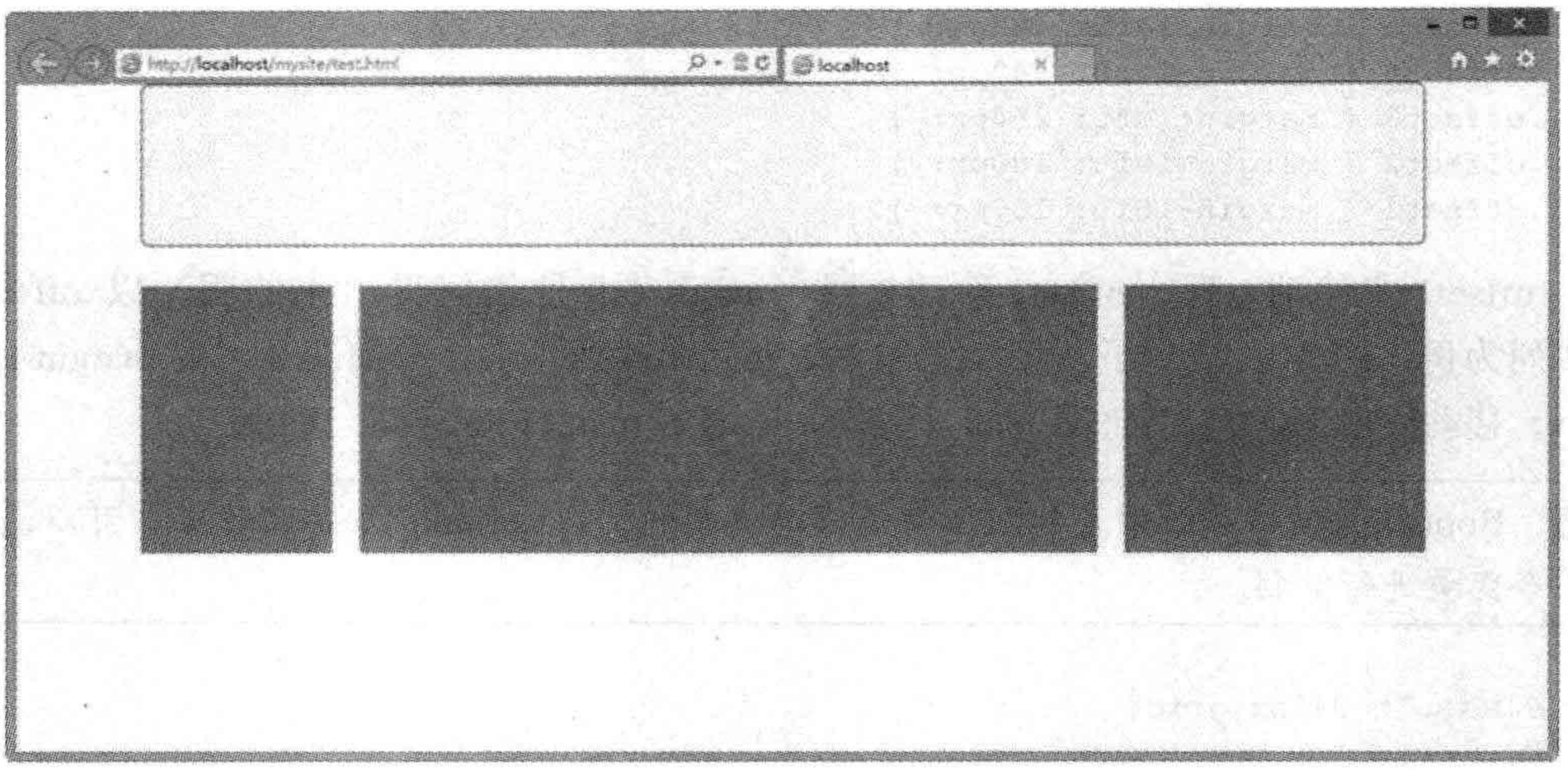


图 3-14　Bootstrap 栅格设计效果

3.2.4　响应式 Bootstrap 栅格系统

在默认情况下，Bootstrap 没有启用响应式布局特性。如果加入响应式布局 CSS 文件（bootstrap-responsive.css），栅格系统会自动根据可视窗口的宽度从 724px 到 1170px 进行动态调整。在可视窗口低于 767px 宽的情况下，列将不再固定并且会在垂直方向堆叠。

响应式（即流式）栅格与固定式栅格的区别在于，每列的宽度是按照百分比来计算外围包裹的宽度的，看一下下面的源码：

```
row-fluid {
```

```
    width: 100%;
    *zoom: 1;
}
.row-fluid:before,
.row-fluid:after {
     display: table;
     line-height: 0;
     content: "";
}
.row-fluid:after {
     clear: both;
}
```

定义容器的宽度完全填充包裹外围容器的宽度。

```
.row-fluid [class*="span"]:first-child {
     margin-left: 0;
}
```

清空第一列前面的边距，与固定式中的 margin-left: –20px 效果是一样的。

```
.row-fluid [class*="span"] {
   display: block;
   float: left;
   width: 100%;
   min-height: 30px;
   margin-left: 2.127659574468085%;
   *margin-left: 2.074468085106383%;
   -webkit-box-sizing: border-box;
      -moz-box-sizing: border-box;
           box-sizing: border-box;
}
```

这里只关注与宽度有关的代码。上面代码中 margin-left: 2.127659574468085% 以百分比设置列的左边距（margin-left），约等于 2.12%。以上面固定式中的 940px 为例来算下大约是多少像素：2.12% × 940=19.928，与 20px 很接近。

下面的代码用来设置列的宽度（width），其中百分比也是按照固定式中的原则算得的。

```
.row-fluid .span12 { width: 100%;                          *width: 99.94680851063829%;}
.row-fluid .span11 { width: 91.48936170212765%; *width: 91.43617021276594%;}
.row-fluid .span10 { width: 82.97872340425532%; *width: 82.92553191489361%;}
.row-fluid .span9 { width: 74.46808510638297%;  *width: 74.41489361702126%;}
.row-fluid .span8 { width: 65.95744680851064%;  *width: 65.90425531914893%;}
.row-fluid .span7 { width: 57.44680851063829%;  *width: 57.39361702127659%;}
.row-fluid .span6 { width: 48.93617021276595%;  *width: 48.88297872340425%;}
.row-fluid .span5 { width: 40.42553191489362%;  *width: 40.37234042553192%;}
.row-fluid .span4 { width: 31.914893617021278%  *width: 31.861702127659576%;}
.row-fluid .span3 { width: 23.404255319148934%; *width: 23.351063829787233%;}
.row-fluid .span2 { width: 14.893617021276595%; *width: 14.840425531914894%;}
.row-fluid .span1 { width: 6.382978723404255%;  *width: 6.329787234042553%;}
```

基本的流式栅格 HTML 代码与固定宽度栅格系统用法大致相同，只需要将 .row 替换

为 .row-fluid 就能让任何一行流动起来。应用于每一列的类不用改变，这样能方便地在流式与固定栅格之间切换。

```
<div class="row-fluid">
     <div class="span3">3</div>
     <div class="span6">6</div>
     <div class="span3">3</div>
</div>
```

定义流式栅格的偏移方式与固定网格系统相同：给任何想要偏移的列添加 .offset* 即可。例如，设计如下结构的效果如图 3-15 所示。

```
<div class="row-fluid">
     <div class="span4">4</div>
     <div class="span4 offset4">4 offset 4</div>
</div>
```

图 3-15　Bootstrap 流式栅格偏移效果

例如，针对 3.2.3 节的模式示例，可以把它很轻松地转换为流式布局，代码如下，演示效果如图 3-16 所示。

```
<div class="container-fluid">
     <div class="hero-unit"></div>
     <div class="row-fluid">
          <div class="span2"></div>
          <div class="span7"></div>
          <div class="span3"></div>
     </div>
     <footer></footer>
</div>
```

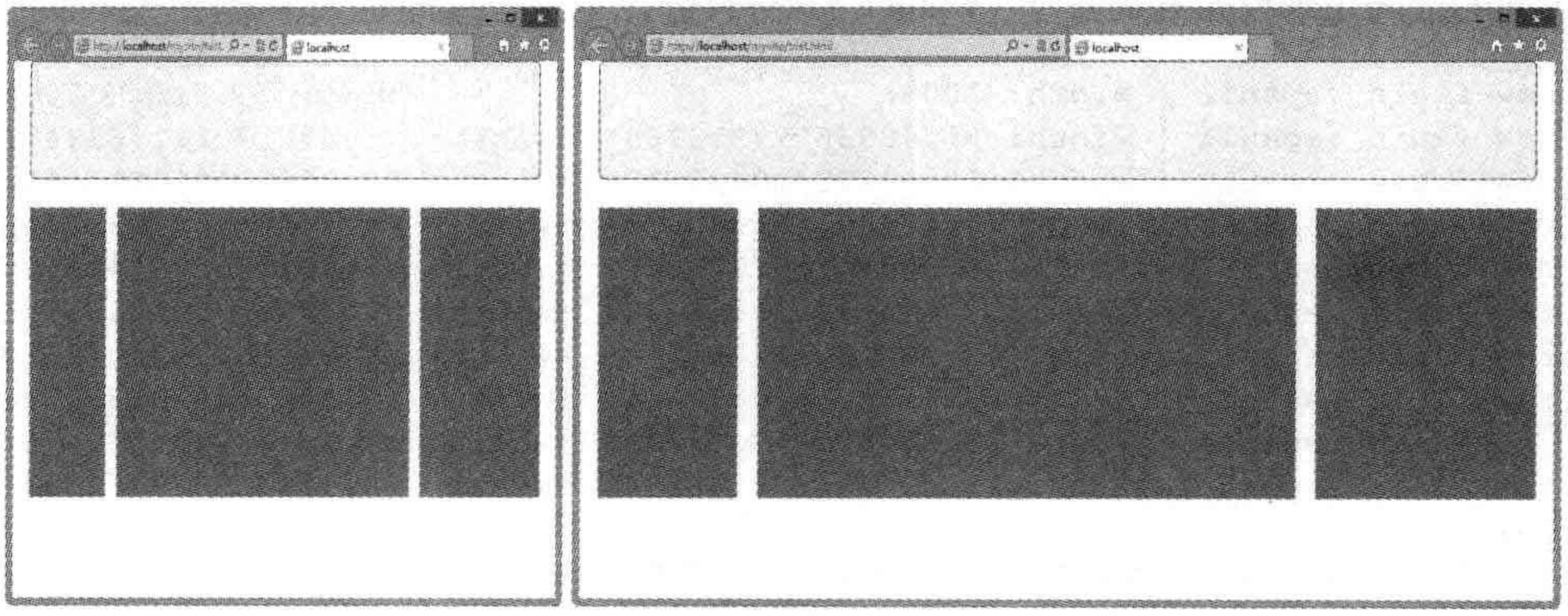


图 3-16　Bootstrap 流式栅格模板效果

不管是固定式栅格，还是流式栅格，它们都可以实现嵌套设计。但是，与固定栅格的嵌套有一点不同，流式栅格要求被嵌套的列数之和等于 12。这是因为流式栅格使用百分比来设置每列的宽度。

例如，在下面的 HTML 结构中，通过流式栅格设计三层嵌套的布局效果，如图 3-17 所示。

```
<div class="container-fluid text-center">
    <div class="row-fluid">
        <div class="span12"> Fluid 12
            <div class="row-fluid">
                <div class="span6"> Fluid 6
                    <div class="row-fluid">
                        <div class="span6">Fluid 6</div>
                        <div class="span6">Fluid 6</div>
                    </div>
                </div>
            <div class="span6">Fluid 6</div>
        </div>
    </div>
</div>
```

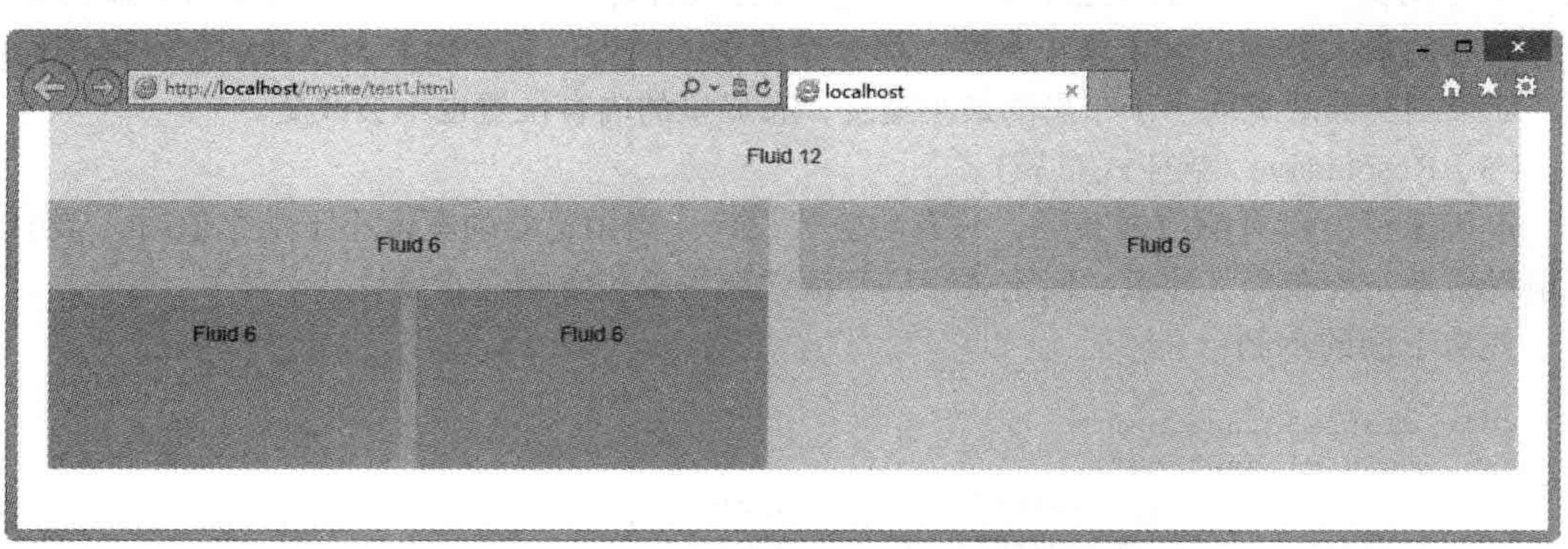


图 3-17　Bootstrap 流式栅格嵌套效果

在固定式布局里，栅格里套栅格的逻辑很简单，只需照着它们固定宽度来设计即可。在流式布局里，栅格里套栅格的逻辑就稍微复杂一点，不照固定宽度来计算，而照宽度百分比来计算。因此，在实际项目中，同样的栅格套栅格，流式布局里的栅格宽度会比固定式布局的宽一点。

在 3.2.3 节中我们介绍过，Bootstrap 允许通过修改 variables.less 的参数值来自定义栅格系统，流式栅格系统同样可以进行类似的修改。修改后并重新编译 Bootstrap 来实现自定义栅格系统。如果添加新的列，需要同时修改 grid.less，同样需要修改 responsive.less 来获得多设备兼容。

3.3 Bootstrap 布局

Bootstrap 提供两种布局方式：固定式布局和流式布局。结合 3.2 节讲解的栅格系统来说，Bootstrap 的布局实际上是在栅格外加个容器（container）。固定式布局加的是固定宽度的容器，流式布局加的是自适应（即可变）宽度的容器，这是二者的唯一区别。

提示 如果对 Bootstrap 布局不是很满意，可以参考 Less Frame work（http://lessframework.com/）提供的模板。

3.3.1 固定式布局

首先，读者需要分辨 3 个概念：

（1）Fix（固定式布局）

对于固定式布局而言，无论浏览器是放大还是缩小，内容位置都不会因受挤压而改变，而且整个布局居中。

（2）Fluid（流式布局）

流式布局要设置布局的最大宽度和最小宽度。如果浏览器的宽度大于布局的最大宽度，那么不管浏览器大小怎么调整，布局里面的内容位置都不受挤压改变。如果浏览器的宽度小于布局的最大宽度，大于最小宽度，那么布局里的内容位置会随着浏览器的大小调整而改变。如果浏览器的宽度小于布局的最小宽度，布局里的内容位置也不会改变。

（3）Responsive（响应式布局）

准确来说，没有响应式布局，只有响应式设计。当浏览器的宽度小于某个值时，整个页面呈现另外一种布局。注意，固定式布局和流式布局都包含响应式设计。

先看下面的示例：

```
<div class="container">
     <div class="row">
          <div class="span4"> span4</div>
          <div class="span8"> span8</div>
     </div>
     <div class="row">
          <div class="span4"> span4</div>
          <div class="span6"> span6</div>
          <div class="span2"> span2</div>
     </div>
</div>
```

在上面结构中， 就是上面所说的容器，其中包含两行栅格（ ）。

Bootstrap 中规定固定容器的总宽度为 940px，具体源码定义如下：

```
.container{
     width: 940px;
}
```

定义这个容器在页面中居中显示，源码如下：

```
.container {
     margin-right: auto;
     margin-left: auto;
     *zoom: 1;
}
```

这里有个技巧，为了让 在各种浏览器下产生同样的居中效果，将 margin-left 和 margin-right 的值设置为 auto，这是最简单的方式。*zoom 这个 CSS Hack 是为了兼容 IE6 和 IE7。

同时，使用了 CSS 伪元素选择器，在这个类里面清空了前后的内容，并且在后面清除浮动：

```
.container:before,
.container:after {
     display: table;
     line-height: 0;
     content: "";
}
.container:after {
     clear: both;
}
```

接着，使用 span 进行整页布局：无论是做几列的布局，一定要保证在一行内各个 span 的名字加起来正好是 12，例如，span4+span8 等于 12，也可以是一个单独的 span12，还可以是 span6+span6 或 span4+span4+span4 等。每行内的 span 加起来总和为 12，这样就可以保证完整的页面布局。

提示　虽然容器的宽度为 940px，每个 span 的宽度都是默认值，但在实际开发中用户可以适当重写。

Bootstrap 采用的是 12 列布局格式，即在页面一行之内最多可以布置 12 列。当然在局部栏目中不会使用 12 列，通常侧栏使用 2 列或 3 列，最多不会超过 4 列，主栏可以使用 6 列、7 列或 8 列。

3.3.2　流式布局

流式布局是一种自适应屏幕的设计方法，即不固定块的宽度，而是以百分比为单位来确定每一块的宽度。这种布局非常适合一次编写，然后自适应各种不同大小的屏幕，如智能手机、平板电脑等设备。

流式布局与固定式布局的结构和用法相同。例如：

```
<div class="container-fluid">
     <div class="row-fluid">
          <div class="span2"></div>
          <div class="span10"></div>
     </div>
```

```
        <div class="row-fluid">
            <div class="span2"></div>
            <div class="span10"></div>
        </div>
</div>
```

最外面包含容器是 ，定义 container-fluid 类，表明内容布局是流式布局，其主要作用是作为一个包含块来容纳流式布局内容。里面包含块的内容有 row-fluid 类（非常重要），决定是不是流式布局。然后里面的内容块代码编写与网格系统一致，依然是从 span1 到 span12，分别对应于不同的百分比。

是流式布局的容器，其源码如下：

```
.container-fluid {
    padding-right: 20px;
    padding-left: 20px;
    *zoom: 1;
}
```

容器左右各加了 20px 的内边距：

```
.container-fluid:before,
.container-fluid:after {
    display: table;
    line-height: 0;
    content: "";
}
```

清空了前后的内容，并且在后面清除了浮动：

```
.container-fluid:after {
    clear: both;
}
```

其实并非固定容器中只能配固定式栅格，流式容器只能配流式栅格，要视需要而定。例如，在下面的结构中，在流式栅格中插入了固定式布局。

```
<div class="container-fluid">
    <div class="row-fluid">
        <div class="span2"></div>
        <div class="span10"></div>
    </div>
    <div class="row">
        <div class="span2"></div>
        <div class="span10"></div>
    </div>
</div>
```

3.3.3 布局嵌套

Bootstrap 对于栅格系统中布局嵌套提供了简单的实现方式。即在需要嵌套的 span 内部，

新加入一个容器 row，在 row 内继续使用前面提到的 span。例如：

```
<div class="row">
    <div class="span12"> 嵌套的顶级
        <div class="row">
            <div class="span6"> 嵌套的 2 级 </div>
            <div class="span6"> 嵌套的 2 级 </div>
        </div>
    </div>
</div>
```

因为是嵌套，所以在 span 内加入了 row，row 内再继续进行 span，以此类推。

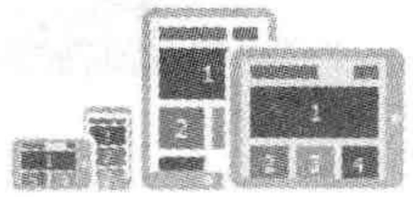
归结起来，Bootstrap 的布局其实就是容器 + 栅格系统，容器只是限制包含框的宽度，主要变化在于栅格，通过栅格的合并、偏移、嵌套来最终达到布局效果的。

3.4　响应式设计

在现代网页设计中，页面的设计与开发应该根据用户行为及设备环境（如系统平台、屏幕尺寸、屏幕定向等）进行相应的响应和调整，这种设计理念就是响应式 Web 设计。响应式设计追求的是一个网站能够兼容多个终端，而不是为每个终端做一个特定的设计版本，这样就不必为不断出现的新设备进行专门的版本设计和开发。在 Web 设计和开发领域，开发者无法跟上设备与分辨率革新的步伐，对于多数网站来说，为每种新设备、分辨率设计独立的版本是不切实际的，而响应式设计可以帮助我们避免这种情况。

3.4.1　什么是响应式设计

手机的屏幕比较小，宽度通常在 600 像素以下，PC 的屏幕宽度一般都在 1000 像素以上（目前主流宽度是 1366 像素 ×768 像素），有的还达到了 2000 像素。同样的内容，要在大小迥异的屏幕上都呈现出满意的效果，并不是一件容易的事。



很多网站的解决方法是，为不同的设备提供不同的网页，如专门提供一个移动版本，或者 iPhone/iPad 版本。这样做固然保证了效果，但是比较麻烦，同时要维护好几个版本，而且如果一个网站有多个入口，会大大增加架构设计的复杂度，如图 3-18 所示。于是，很早就有人设想，能不能实现“一次设计，普遍适用”，让同一个网页自动适应不同大小

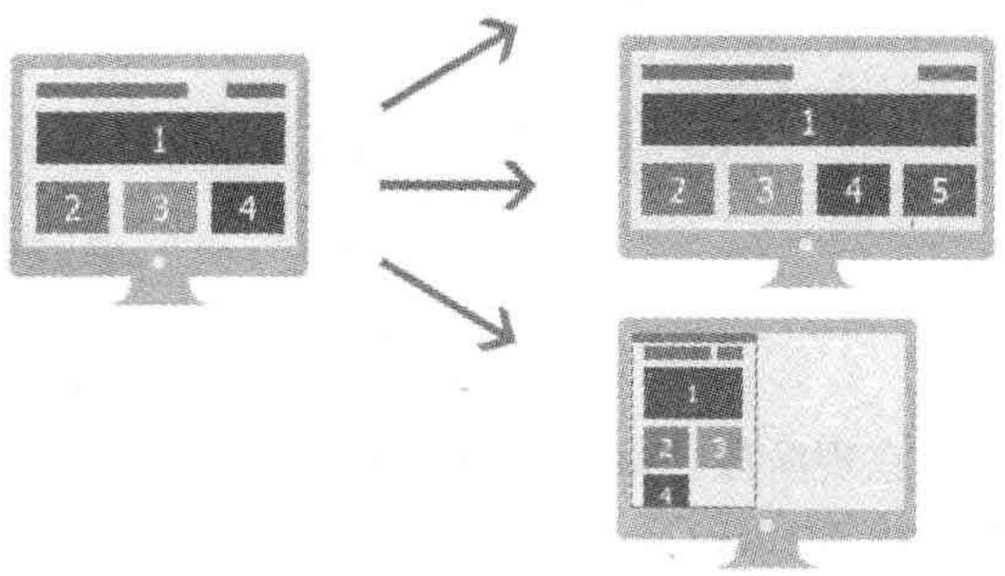


图 3-18　适应不同设备的多套网站效果

的屏幕，根据屏幕宽度，自动响应调整布局（layout）。

响应式 Web 设计是一种技术，它可以使网站适应于不同设备，这些设备可以是智能手机、平板电脑、电视机、PC 显示器、iPhone 和 Android 手机，包括横向、纵向的屏幕，如图 3-19 所示。开发人员只需要正确地实现响应式 Web 设计，网站就可以很好地适合各种设备。

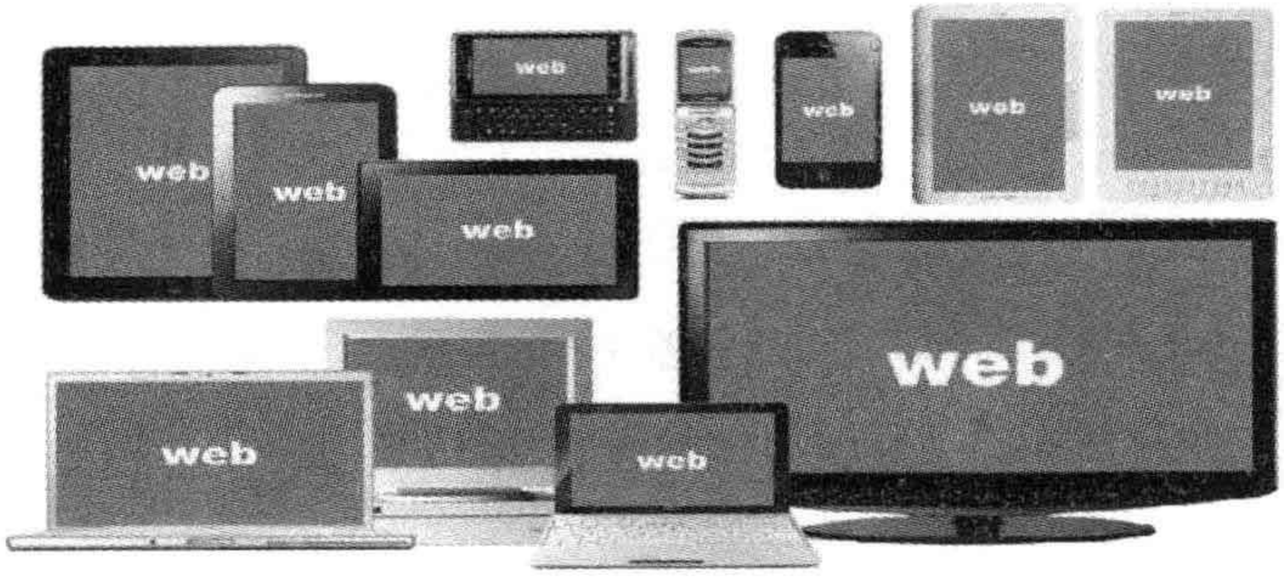


图 3-19　风采多样的屏幕设备

不同设备都有各自的屏幕分辨率、清晰度以及屏幕定向方式（如横屏、竖屏、正方形等），对于日益流行的 iPhone、iPad 及其他一些智能手机、平板电脑，用户还可以通过转动设备任意切换屏幕的定向方式，如图 3-20 所示。怎样才能做到让一种设计方案满足所有情况呢？

图 3-20　适应不同设备类型的响应式设计

在现阶段，响应式 Web 设计的实现途径包括：弹性栅格、流式布局、弹性图片显示、使用 CSS Media Query 技术等。无论用户正在使用台式机、笔记本还是平板电脑（如 iPad）、智能手机（如 iPhone）等，设计的页面都能够自动切换分辨率、图片尺寸及相关脚本功能等，以适应不同设备，即页面能够自动响应用户的设备环境。

几年前，弹性布局（flexible layout）还鲜有人提及，所谓的弹性也只体现在竖排布局以及字号等方面，图片可以轻易破坏页面结构，而且即使是弹性的元素结构，在很极端的情况下，仍会破坏布局。可见，所谓的弹性布局其实并非那样有弹性，它有时甚至不能适应台式机、笔记本的屏幕分辨率差异，更不用说手机等移动设备了。

现在，我们可以通过响应式的设计和开发思路使页面更加具有弹性。图片的尺寸可以自动调整，而页面布局也不会再破坏。无论用户是切换设备的屏幕定向方式，还是从台式机屏幕转到 iPad 上浏览，页面都会真正的富有弹性。

例如，2010 年 Ethan Marcotte 提出了“响应式 Web 设计”（Responsive Web Design）这个名词，他制作了一个范例，展示了响应式 Web 设计在页面弹性方面的特性。页面地址为 http://alistapart.com/d/responsive-web-design/ex/ex-site-flexible.html，内容是《福尔摩斯历险记》6 个主人公的头像。如果屏幕宽度大于 1300 像素，则 6 张图片并排在一行，如图 3-21 所示。

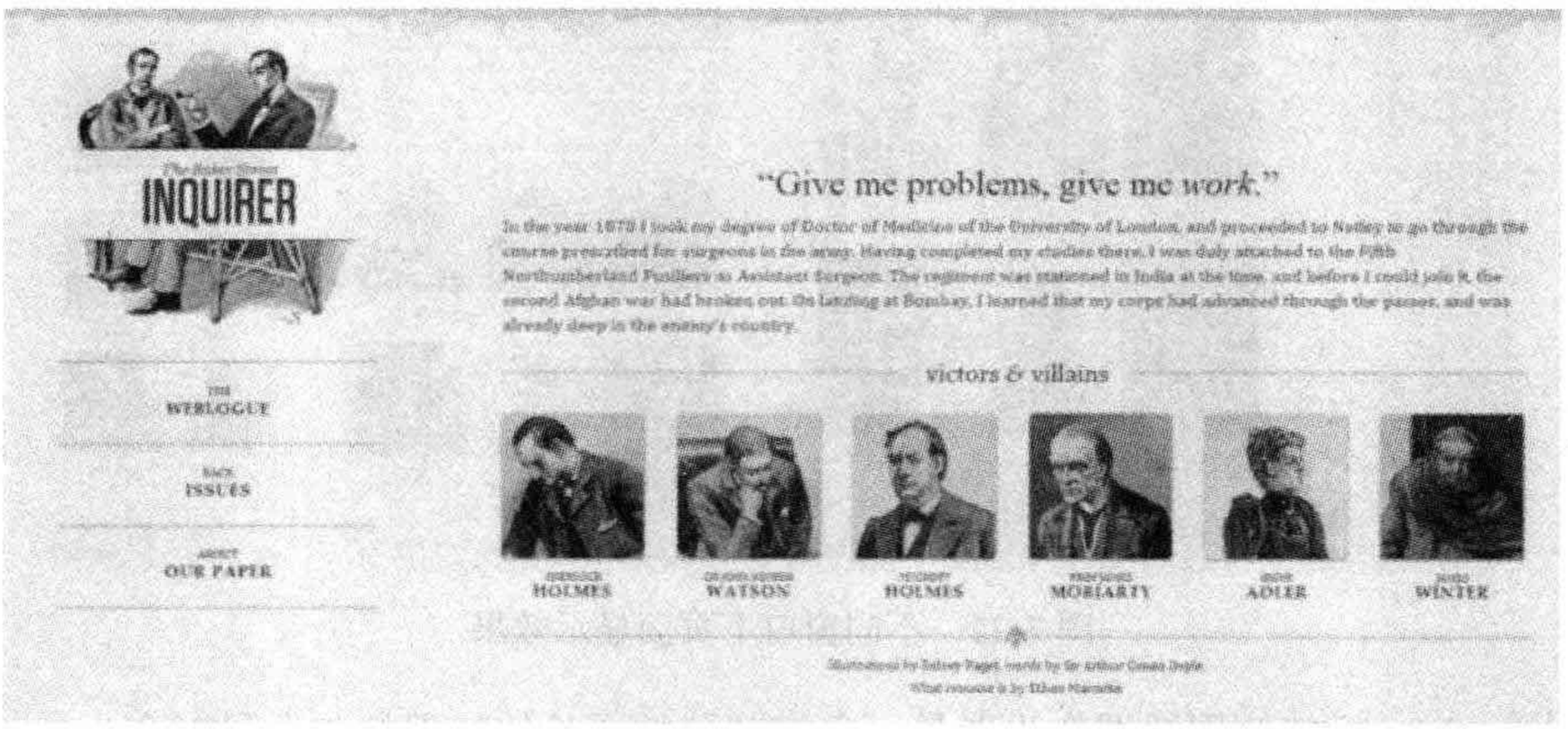


图 3-21　宽屏显示效果

如果屏幕宽度在 600 像素到 1300 像素之间，则 6 张图片分成两行，如图 3-22 左图所示。如果屏幕宽度在 400 像素到 600 像素之间，则页面排成一列，导航栏移到正文内容上面，如图 3-22 中图所示。如果屏幕宽度在 400 像素以下，则 6 张图片分成 3 行，如图 3-22 右图所示。

如果将浏览器窗口不断调小，会发现 logo 图片的文字部分始终会保持同比缩小，保证其完整可读，而不会和周围的插图一样被两边裁掉。所以整个 logo 其实包括两部分：插图作为页面标题的背景图片，会保持尺寸，但会随着布局调整而被裁切；文字部分则是一张单独的图片。

```
<h1 id="logo">
    <a href="#"><img src="site/logo.png" alt="The Baker Street Inquirer" /></a>
</h1>
```

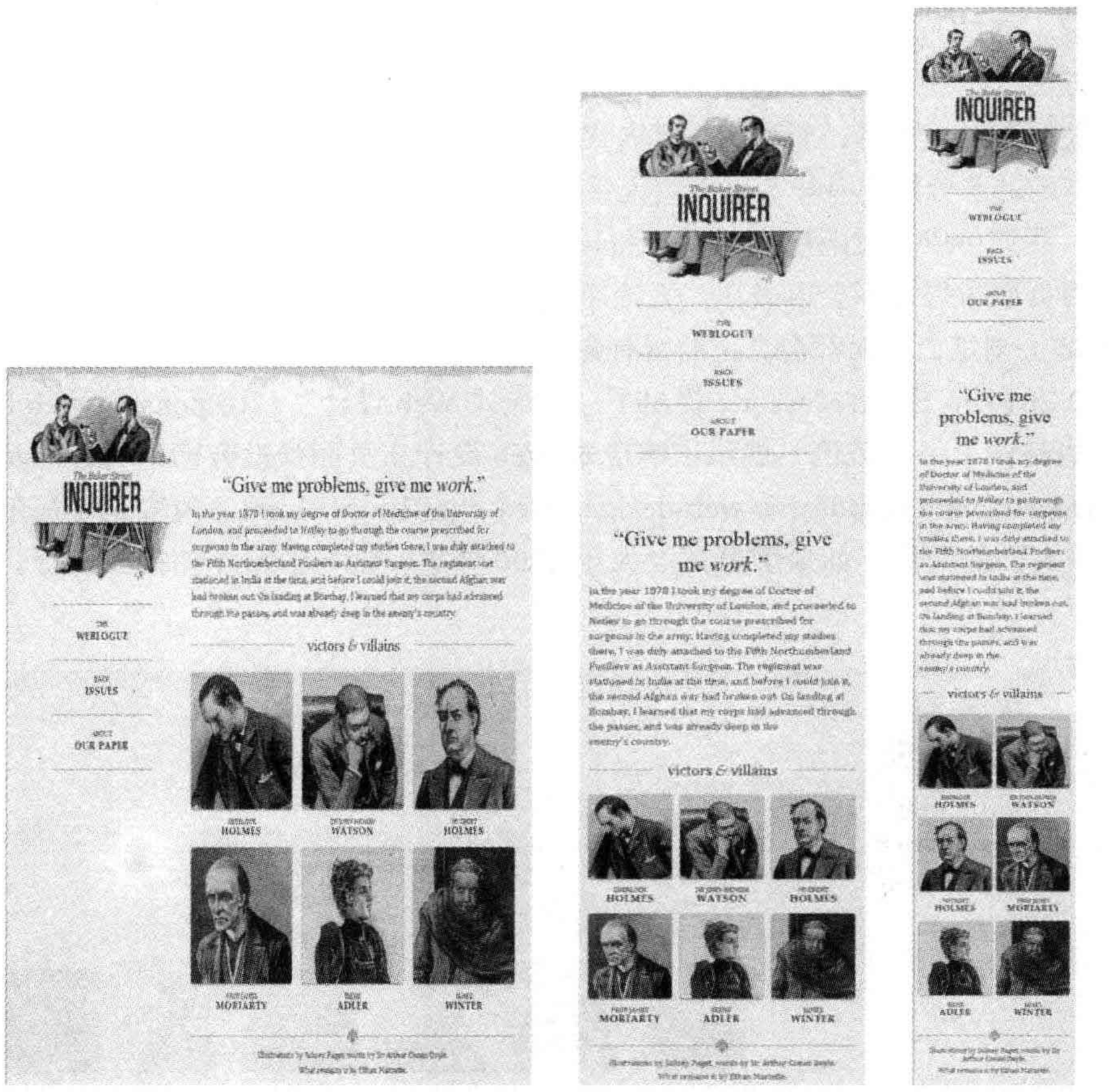


图 3-22　不同窗口下页面显示效果

其中，<h1> 标签使用插图作为背景，文字部分的图片始终保持与背景对齐。

该实例的实现方式完美地结合了流式栅格和流式图片技术，展示了响应式 Web 设计的思路，并且聪明的在正确的地方使用了正确的 HTML 标签。

mediaqueri.es 网站（http://mediaqueri.es/）提供了更多这样的例子。使用一个测试工具（http://www.benjaminkeen.com/open-source-projects/smaller-projects/responsive-design-bookmarklet/），还可以在一张网页上同时显示不同分辨率屏幕的测试效果。

3.4.2　设计响应式图片

在响应式 Web 设计的思路中，一个重要的因素是如何正确处理图片大小问题。

首先，应该设置图片具有弹性能力。弹性图片的设计思路是，无论何时，确保在图片原始宽度范围内，以最大的宽度同比完整地显示图片。我们不必在样式表中为图片设置宽度和高度，只需要让样式表在窗口尺寸发生变化时，辅助浏览器对图片进行缩放。

有很多同比缩放图片的技术，其中有不少是简单易行的，比较流行的方法是使用 CSS 的

max-width 属性。

```
img {
    max-width: 100%;
}
```

只要没有其他涉及图片宽度的样式代码覆盖上面的样式，且容器可视部分的宽度小于图片的原始宽度，那么页面上所有的图片就会以其原始宽度进行加载。上面的代码确保图片最大的宽度不会超过浏览器窗口或是其容器可视部分的宽度，所以当窗口或容器的可视部分开始变窄时，图片的最大宽度值也会相应地减小，图片本身永远不会被容器边缘隐藏或覆盖。

老版本的 IE 不支持 max-width，可以对其单独设置为：

```
img {
    width: 100%;
}
```

此外，在 Windows 平台上缩放图片时，可能出现图像失真现象。这时，可以尝试使用 IE 的专有命令：

```
img {
    -ms-interpolation-mode: bicubic;
}
```


或者，使用 Ethan Marcotte 开发的专用插件 imgSizer.js（http://unstoppablerobotninja.com/demos/resize/imgSizer.js）。

```
addLoadEvent(function() {
    var imgs = document.getElementById("content").getElementsByTagName("img");
    imgSizer.collate(imgs);
});
```

如果有条件的话，最好能够根据屏幕的不同大小，加载不同分辨率的图片。有很多方法可以做到这一条，服务器端和客户端都可以实现。

图片本身的分辨率、加载时间是另外一个需要考虑的问题。虽然通过上面的方法，可以很轻松地缩放图片，确保图片在移动设备的窗口中可以被完整浏览，但如果原始图片过大，便会显著降低图片文件的下载速度，对存储空间也会造成不必要的消耗。这个话题将在 3.4.3 节进行介绍。

要实现图片的智能响应，应该解决两个问题：自适应图片缩放尺寸，以及在小设备上自动降低图片的分辨率。

为此，Filament Group 提供了一种解决方案，这个方案的实现需要配合使用两个相关文件：rwd-images.js 和 .htaccess，读者可以在 Github 上获取（https://github.com/filamentgroup/Responsive-Images），具体使用方法可以参考 Responsive Images 的说明文档（https://github.com/filamentgroup/Responsive-Images#readme）。

Responsive Images 的设计原理是：使用 rwd-images.js 文件检测当前设备的屏幕分辨率，如果是大屏幕设备，则向页面头部区域添加 Base 标记，并将后续的图片、脚本和样式表加

载请求定向到虚拟路径 "/rwd-router"。当这些请求到达服务器端，.htaccess 文件会决定这些请求所需要的是原始图片还是小尺寸的响应式图片，并进行相应的反馈输出。对于小屏幕的移动设备，原始尺寸的大图片永远不会用到。

该技术支持大部分现代浏览器，如 IE8（及以上版本）、Safari、Chrome 和 Opera，以及这些浏览器的移动设备版本。在 Firefox 及一些旧浏览器中，仍可得到小图片的输出，但同时也会下载原始大图。

例如，使用不同的设备访问页面 http://filamentgroup.com/examples/responsive-images/ 就会发现，不同设备中所显示的图片分辨率是不同的，如图 3-23 所示。

图 3-23　不同设备下图片分辨率不同

提示　在 iPhone、iPod Touch 中，页面会被自动地同比例缩小至最适合屏幕大小的尺寸，x 轴不会产生滚动条，用户可以上下拖拽浏览全部页面，或在需要的时候放大页面的局部。这里会产生一个问题，即使使用响应式 Web 设计的方法，专门为 iPhone 输出小图片，它同样会随着整个页面一起被同比例缩小，如图 3-24 左图所示。

图 3-24　不同设备下视图下的效果

针对上面的问题，可以使用苹果公司的专有 meta 标签来解决。在页面的 <head> 部分添加以下代码：

```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

viewport 是网页默认的宽度和高度，上面这行代码的意思是，网页宽度默认等于屏幕宽度（width=device-width），原始缩放比例为 1.0（initial-scale=1），即网页初始大小占屏幕面积的 100%。更多关于 viewport meta 标签的用法，可以参考苹果公司官方文档（http://developer.apple.com/library/safari/#documentation/appleapplications/reference/safarihtmlref/Articles/MetaTags.html）。

3.4.3　设计响应式布局结构

由于网页需要根据屏幕宽度自动调整布局，我们不能使用绝对宽度的布局，也不能使用具有绝对宽度的元素。具体来说，CSS 代码不能指定像素宽度：

```
width: 940px;
```

只能指定百分比宽度：

```
width: 100%;
```

或者

```
width: auto;
```

网页字体大小也不能使用绝对大小（px），而只能使用相对大小（em）。例如：

```
body {
     font: normal 100% Helvetica, Arial, sans-serif;
}
```

上面的代码定义字体大小是页面默认大小的 100%，即 16 像素。

```
h1 {
    font-size: 1.5em;
}
```

然后，定义一级标题的大小是默认字体大小的 1.5 倍，即 24 像素（24/16=1.5）。

```
small {
      font-size: 0.875em;
}
```

定义 small 元素的字体大小是默认字体大小的 0.875 倍，即 14 像素（14/16=0.875）。

流体布局（http://alistapart.com/article/fluidgrids）是响应式设计中的一个重要方面，它要求页面中各个区块的位置都是浮动的，而不是固定不变的。

```
.main {
```

```
        float: right;
        width: 70%;
}
.leftBar {
        float: left;
        width: 25%;
}
```

Float 的优势是如果宽度太小，并列显示不下两个元素，后面的元素会自动换前面元素的下方显示，而不会出现水平方向溢出（overflow），避免了水平滚动条的出现。另外，应该尽量减少绝对定位（position: absolute）的使用。

在响应式 Web 设计中，除了图片方面，我们还应考虑页面布局结构的响应式调整。一般可以使用独立的样式表，或者使用 CSS Media Query 技术。例如，可以使用一个默认主样式表来定义页面的主要结构元素，如 #wrapper、#content、#sidebar、#nav 等的默认布局方式，以及一些全局性的样式方案。

然后可以监测页面布局随着不同的浏览环境而产生的变化，如果它们变的过窄、过短、过宽、过长，则通过一个子级样式表来继承主样式表的设定，并专门针对某些布局结构进行样式覆盖。

例如，下面的代码可以放在默认主样式表 style.css 中：

```
html, body {}
h1, h2, h3 {}
p, blockquote, pre, code, ol, ul {}
/* 结构布局元素 */
#wrapper {
    width: 80%;
    margin: 0 auto;
    background: #fff;
    padding: 20px;
}
#content {
    width: 54%;
    float: left;
    margin-right: 3%;
}
#sidebar-left {
    width: 20%;
    float: left;
    margin-right: 3%;
}
#sidebar-right {
    width: 20%;
    float: left;
}
```

下面的代码可以放在子级样式表 mobile.css 中，专门针对移动设备进行样式覆盖：

```
#wrapper {
```

```
        width: 90%;
}
#content {
        width: 100%;
}
#sidebar-left {
        width: 100%;
        clear: both;
        border-top: 1px solid #ccc;
        margin-top: 20px;
}
#sidebar-right {
        width: 100%;
        clear: both;
        border-top: 1px solid #ccc;
        margin-top: 20px;
}
```

CSS3 支持在 CSS2.1 中定义的媒体类型，同时添加了很多涉及媒体类型的功能属性，包括 max-width（最大宽度）、device-width（设备宽度）、orientation（屏幕定向：横屏或竖屏）和 color。在 CSS3 发布之后，新上市的 iPad、Android 相关设备都可以完美支持这些属性。所以，可以通过 Media Query 为新设备设置独特的样式，而忽略那些不支持 CSS3 的台式机中的旧浏览器。

例如，下面代码定义了如果页面通过屏幕呈现，非打印一类，并且屏幕宽度不超过 480px，则加载 shetland.css 样式表。

```
    <link rel="stylesheet" type="text/css" media="screen and (max-device-width:
480px)" href="shetland.css" />
```

用户可以创建多个样式表，以适应不同设备类型的宽度范围。当然，更有效率的做法是：将多个 Media Query 整合在一个样式表文件中：

```
    @media only screen  and (min-device-width: 320px)  and (max-device-width:
480px) {
        /* Styles */
    }
    @media only screen  and (min-width: 321px) {
        /* Styles */
    }
    @media only screen  and (max-width: 320px) {
        /* Styles */
    }
```

上面的代码可以兼容各种主流设备。这样整合多个 Media Query 于一个样式表文件中的方式，与通过多个 Media Query 调用不同样式表是不同的。

上面的代码受 CSS2.1 和 CSS3 支持，也可以使用 CSS3 专有的 Media Query 功能来创建响应式 Web 设计。通过 min-width 可以设置在浏览器窗口或设备屏幕宽度高于这个值的情况下，为页面指定一个特定的样式表，而 max-width 属性则反之。

例如，使用多个 Media Query 整合在单一样式表中，这样做更加高效，可以减少请求数量。

```
@media screen and (min-width: 600px) {
  .hereIsMyClass {
       width: 30%;
       float: right;
  }
}
```

上面的代码中定义的样式类只有在浏览器或屏幕宽度超过 600px 时才会有效。

```
@media screen and (max-width: 600px) {
  .aClassforSmallScreens {
       clear: both;
       font-size: 1.3em;
  }
}
```

而这段代码的作用则相反，该样式类只有在浏览器或屏幕宽度小于 600px 时才会有效。

因此，使用 min-width 和 max-width 可以同时判断设备屏幕尺寸与浏览器实际宽度。如果希望通过 Media Query 作用于某种特定的设备，但忽略在其上运行的浏览器是否由于没有最大化而尺寸与设备屏幕尺寸不一致，则可以使用 min-device-width 与 max-device-width 属性来判断设备本身的屏幕尺寸。

```
@media screen and (max-device-width: 480px) {
  .classForiPhoneDisplay {
       font-size: 1.2em;
   }
}
@media screen and (min-device-width: 768px) {
  .minimumiPadWidth {
       clear: both;
       margin-bottom: 2px solid #ccc;
   }
}
```

还有一些其他方法，可以有效使用 Media Query 锁定某些指定的设备。

对于 iPad 来说，orientation 属性很有用，它的值可以是 landscape（横屏）或 portrait（竖屏）。

```
@media screen and (orientation: landscape) {
   .iPadLandscape {
        width: 30%;
        float: right;
   }
}
@media screen and (orientation: portrait) {
   .iPadPortrait {
        clear: both;
   }
}
```

遗憾的是，这个属性目前确实只在 iPad 上有效。其他可以转屏的设备（如 iPhone），可以使用 min-device-width 和 max-device-width 来变通实现。

下面将上述属性组合使用，来锁定某个屏幕尺寸范围：

```
@media screen and (min-width: 800px) and (max-width: 1200px) {
  .classForaMediumScreen {
        background: #cc0000;
        width: 30%;
        float: right;
  }
}
```

上面的代码可以作用于浏览器窗口或屏幕宽度在 800px 至 1200px 之间的所有设备。

其实，用户仍然可以选择使用多个样式表的方式来实现 Media Query。如果从资源的组织和维护的角度出发，这样做更高效。

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
<link rel="stylesheet" media="screen and (min-width: 600px)" href="large.css" />
<link rel="stylesheet" media="print" href="print.css" />
```

读者可以根据实际情况决定使用 Media Query 的方式。例如，对于 iPad，可以将多个 Media Query 直接写在一个样式表中。因为 iPad 用户随时有可能切换屏幕定向，这种情况下，要保证页面在极短的时间内响应屏幕尺寸的调整，我们必须选择效率最高的方式。

Media Query 不是绝对唯一的解决方案，它只是一个以纯 CSS 方式实现响应式 Web 设计思路的手段。另外，还可以使用 JavaScript 来实现响应式设计。特别是当某些旧设备无法完美支持 CSS3 的 Media Query 时，它可以作为后备支援。我们可以使用专业的 JavaScript 库来帮助支持旧浏览器（如 IE 5+、Firefox 1+、Safari 2 等）支持 CSS3 的 Media Queries。使用方法很简单，下载 css3-mediaqueries.js（http://code.google.com/p/css3-mediaqueries-js/），然后在页面中调用它即可。

所有主流浏览器（包括 IE9）都支持 Media Query，对于老式浏览器（主要是 IE6、IE7、IE8）则可以考虑使用 css3-mediaqueries.js。

```
<!-[if lt IE 9]>
<script  src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-
mediaqueries.js"></script>
    <![endif]->
```

例如，以下代码演示了如何使用简单的几行 jQuery 代码来检测浏览器宽度，并为不同的情况调用不同的样式表：

```
    <script  type="text/JavaScript"  src="http://ajax.googleapis.com/ajax/libs/
jquery/1.9.1/jquery.min.js"></script>
    <script type="text/JavaScript">
    $(document).ready(function(){
        $(window).bind("resize", resizeWindow);
```

```
function resizeWindow(e){
    var newWindowWidth = $(window).width();
    if(newWindowWidth < 600){
        $("link[rel=stylesheet]").attr({href: "mobile.css"});
    }
    else if(newWindowWidth > 600){
        $("link[rel=stylesheet]").attr({href: "style.css"});
    }
});
</script>
```

类似这样的解决方案还有很多，借助 JavaScript，我们则可以实现更多的变化。

3.4.4 自适应显示 / 隐藏页面内容

对于响应式 Web 设计，同比例缩放元素尺寸及调整页面结构布局是两个重要的响应方法。但是对于页面中的文字内容信息来说，则不能简单地以同比缩小或者调整布局结构的方法进行处理。对于手机等移动设备来说，在文字内容方面，已经有了很多最佳实践方式和指导原则：简化的导航、更易聚焦的内容、以信息列表代替传统的多行文案内容等。

响应式 Web 设计的思想如下：

- 一方面要保证页面元素及布局具有足够的弹性，来兼容各类设备平台和屏幕尺寸；
- 另一方面则是增强可读性和易用性，帮助用户在任何设备环境中都能更容易地获取最重要的内容信息。

针对这个问题，我们可以使用下面这条样式代码来解决：

```
display: none;
```

我们可以在一个针对某类小屏幕设备的样式表中使用它来隐藏掉页面中的某些块级元素，也可以使用前面的方法，通过 JavaScript 判断当前硬件屏幕规格，在小屏幕设备的情况下直接为需要隐藏的元素添加工具类 class。例如，对于手机类设备，可以隐藏掉大块的文字内容区，而只显示一个简单的导航结构，其中的导航元素可以指向详细内容页面。

注意，不要使用 visibility: hidden 的方式，因为这只能使元素在视觉上不呈现；display 属性则可帮助我们设置整块内容是否需要输出。

对初学者来说，响应式设计可能比较复杂，但是事实上它是很简单的。例如，下面示例通过简单的几步设计一个初步响应式页面效果。

第 1 步：通过 Dreamweaver 新建一个 HTML5 文档，在头部区域定义 meta 标签。大多数移动浏览器将 HTML 页面放大为宽视图（viewport）以符合屏幕分辨率。这里可以使用视图的 meta 标签来进行重置，让浏览器使用设备的宽度作为视图宽度并禁止初始的缩放。

```
<!doctype html>
<html>
<head>
```

```
<meta charset="utf-8">
<title></title>
<!-- viewport meta to reset iPhone inital scale -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
</body>
</html>
```

第 2 步：IE8 或者更早的浏览器并不支持 Media Query。可以使用 media-queries.js 或者 respond.js 来为 IE 添加 Media Query 支持。

```
<!-- css3-mediaqueries.js for IE8 or older -->
<!--[if lt IE 9]>
    <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.
     js"></script>
<![endif]-->
```

第 3 步：设计页面 HTML 结构。整个页面基本布局包括头部、内容、侧边栏和页脚等。头部为固定高度 180 像素，内容容器宽度是 600 像素，而侧边栏宽度是 300 像素，线框图如图 3-25 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<!-- viewport meta to reset iPhone inital scale -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<!-- css3-mediaqueries.js for IE8 or older -->
<!--[if lt IE 9]>
    <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.
     js"></script>
<![endif]-->
</head>
<body>
<div id="pagewrap">
     <div id="header">
          <h1>Header</h1>
          <p>Tutorial by <a href="#">Myself</a> (read <a href="#">related article</a>)</p>
</div>
<div id="content">
          <h2>Content</h2>
          <p>text</p>
</div>
<div id="sidebar">
          <h3>Sidebar</h3>
          <p>text</p>
</div>
<div id="footer">
```

```
            <h4>Footer</h4>
        </div>
    </div>
    </body>
    </html>
```

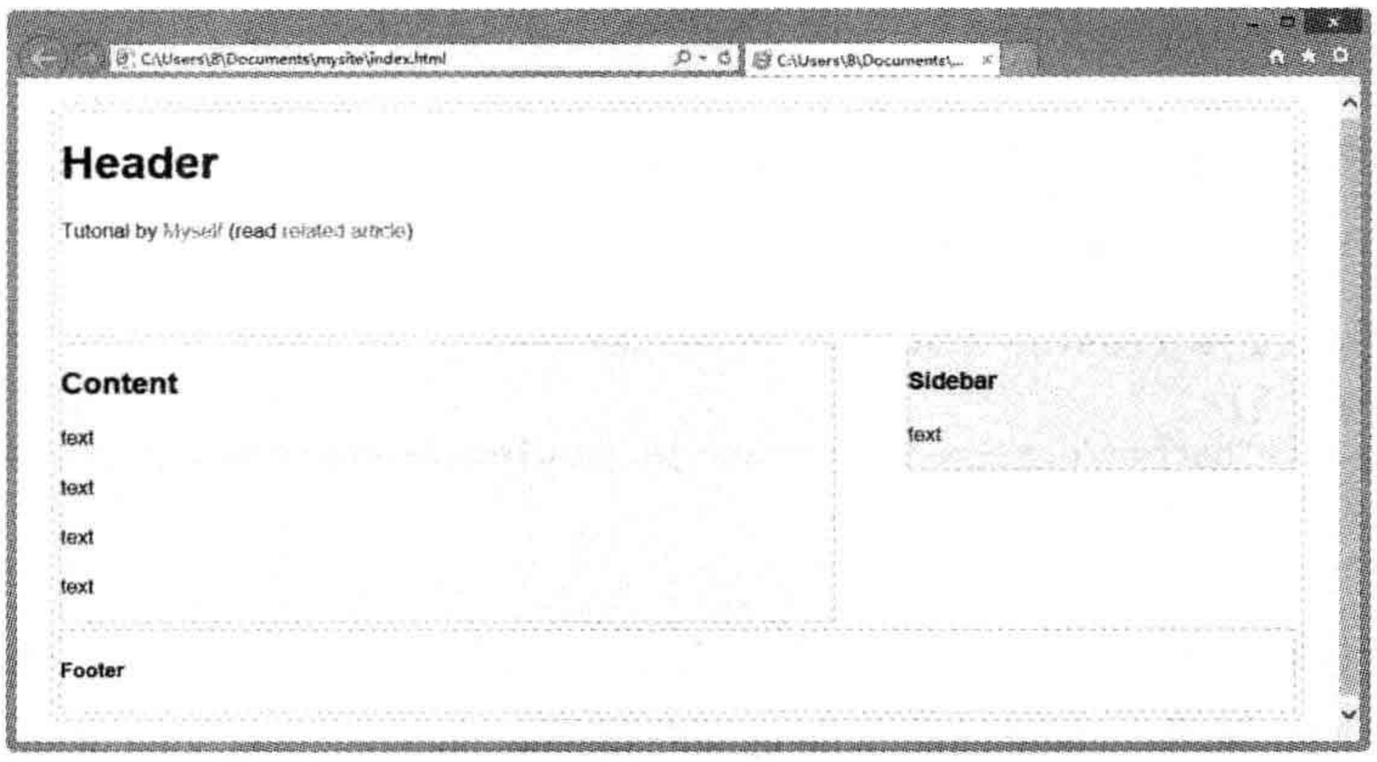


图 3-25　设计页面结构

第 4 步：使用 Media Query。CSS3 Media Query 是响应式设计的核心，它根据条件告诉浏览器如何为指定视图宽度渲染页面。

当视图宽度小于或等于 980 像素时，如下规则将会生效。基本上，会将所有的容器宽度从像素值设置为百分比以使得容器大小自适应。

```
/* for 980px or less */
@media screen and (max-width: 980px) {

    #pagewrap {
        width: 94%;
    }
    #content {
        width: 65%;
    }
    #sidebar {
        width: 30%;
    }
}
```

第 5 步：为小于或等于 700 像素的视图指定 #content 和 #sidebar 的宽度为自适应并且清除浮动，使得这些容器按全宽度显示。

```
/* for 700px or less */
@media screen and (max-width: 700px) {
    #content {
         width: auto;
```

```
        float: none;
    }
    #sidebar {
        width: auto;
        float: none;
    }
}
```

第 6 步：对于小于等于 480 像素（手机屏幕）的情况，将 #header 元素的高度设置为自适应，将 h1 的字体大小修改为 24 像素并隐藏侧边栏。

```
/* for 480px or less */
@media screen and (max-width: 480px) {
    #header {
        height: auto;
    }
    h1 {
        font-size: 24px;
    }
    #sidebar {
        display: none;
    }
}
```


第 7 步：根据个人喜好添加足够多的 Media Query。上面三段样式代码仅仅展示了 3 个 Media Query。Media Query 的目的在于为指定的视图宽度指定不同的 CSS 规则，从而来实现不同的布局。演示效果如图 3-26 所示。

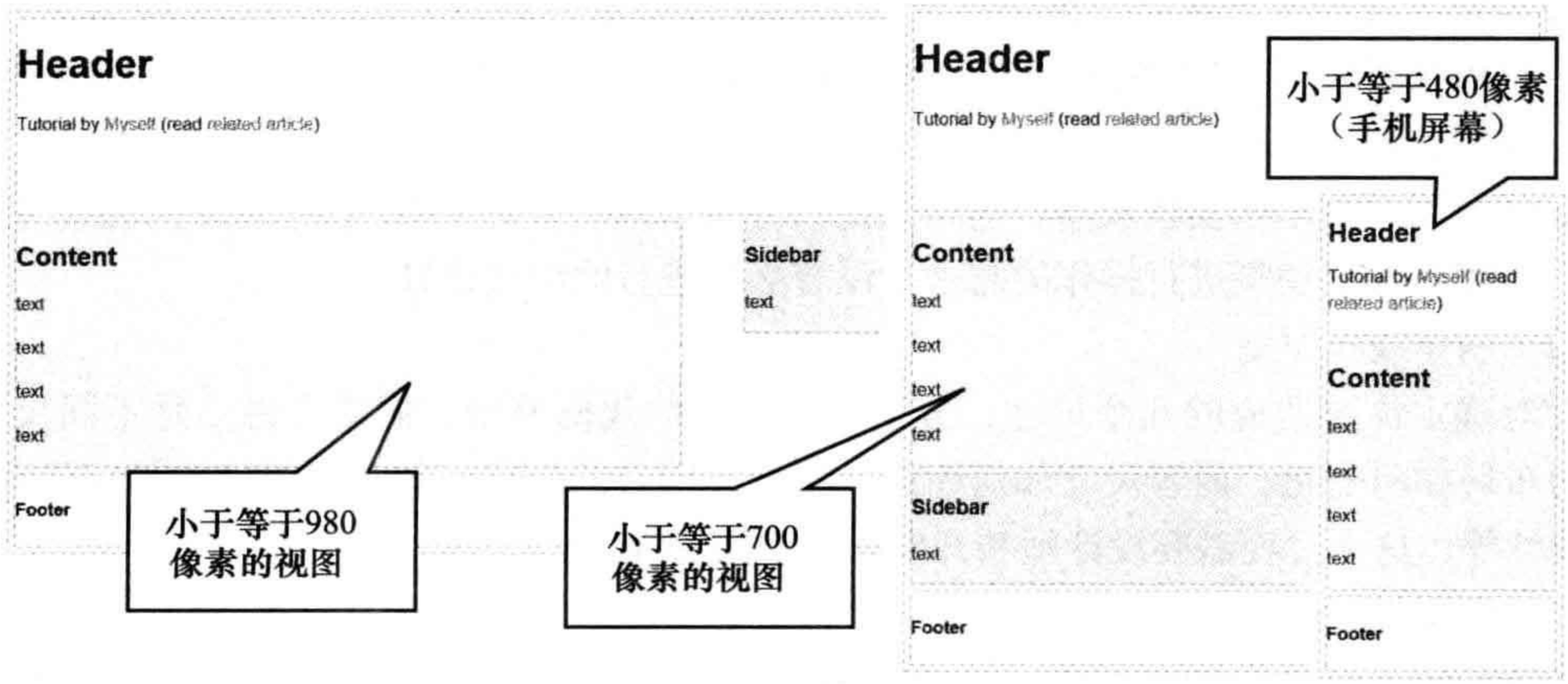


图 3-26　设计不同宽度下的视图效果

提示　触屏设备已经成为主流。虽然目前多数触屏设备（如手机）还是小屏幕，但是市场上越来越多的大屏幕设备也开始使用触屏技术。且不说 iPad 一类的平板电脑，就连一些笔记本和台式

机也加入了这一行列，如 HP Touchsmart tm2t 既使用传统的键鼠设备，同时也加入了触屏技术。

相比于传统的基于鼠标指针的互动，触屏技术显然带来了截然不同的交互方式与相应的设计规范，两者又有各自所适用的领域。所幸，要使我们的设计方案同时满足这两类设备的规范并非难事，只是有些地方需要注意。比如，触屏设备无法反映 CSS 定义的悬停行为及相应的样式，因为它没有鼠标指针的概念，手指点击就是单击行为。因此，不要让任何功能依赖于对悬停状态的触发。

一般建议设计既要有利于改进针对触屏设备的设计方式，同时又不会削弱传统键鼠设备上的用户体验。例如，放在页面右侧的导航列表可以对触屏设备的用户更加友好。因为多数人习惯用右手操作，而左手负责握持设备；这样，放在右侧的导航列表既方便右手的点击，又可以避免被握着设备的左手不小心触碰到。而这一点与键鼠设备用户的习惯完全不矛盾。

3.4.5 响应式设计流程和实战

响应式 Web 设计流程如下。

（1）确定需要兼容的设备类型、屏幕尺寸

通过用户研究，了解用户使用的设备分布情况，确定需要兼容的设备类型、屏幕尺寸。

设备类型：包括移动设备（智能手机、平板电脑等）和 PC。对于移动设备，设计和实现的时候注意增加手势的功能。

屏幕尺寸：包括各种手机屏幕的尺寸（包括横向和竖向）、各种平板电脑的尺寸（包括横向和竖向）、普通电脑屏幕和宽屏。

在设计中要注意以下问题：

- 在响应式设计页面时，确定页面适用的尺寸范围。例如，1688 搜索结果页面，跨度可以从手机到宽屏电脑，而 1688 首页由于结构过于复杂，想直接迁移到手机上不太现实，不如重新设计一个手机版的首页。
- 结合用户需求和实现成本，对适用的尺寸进行取舍。如一些功能操作的页面，用户一般没有在移动端进行操作的需求，没有必要进行响应式设计。

（2）制作线框原型

针对确定需要适应的几个尺寸，分别制作不同的线框原型，需要考虑清楚不同尺寸下，页面的布局如何变化，内容尺寸如何缩放，功能、内容的删减，甚至针对特殊的环境作特殊化的设计等。这个过程需要设计师和开发人员密切沟通。

（3）测试线框原型

将图片导入到相应的设备进行一些简单的测试，可以尽早发现可访问性、可读性等方面的问题。

（4）视觉设计

由于移动设备的屏幕像素密度与传统电脑屏幕不一样，在设计的时候需要保证内容文字的可读性、控件可点击区域的面积等。

（5）脚本实现

与传统的 Web 开发相比，响应式设计的页面由于页面布局、内容尺寸发生了变化，最终的成品更有可能与设计稿出入较大，需要开发人员和设计师多沟通。

例如，在下面示例中将页面父级容器宽度设置为固定的 980px，对于桌面浏览环境，该宽度适用于任何宽于 1024 像素的分辨率。通过 Media Query 来监测那些宽度小于 980px 的设备分辨率，并将页面的宽度设置由固定方式改为流式版式，布局元素的宽度随着浏览器窗口的尺寸变化进行调整。当可视部分的宽度进一步减小到 650px 以下时，主要内容部分的容器宽度会增大至全屏，而侧边栏将被置于主内容部分的下方，整个页面变为单栏布局。演示效果如图 3-27 所示。

图 3-27　设计不同宽度下的视图效果

在本示例中，主要应用了下面几个技术和技法。

- Media Query JavaScript。对于那些尚不支持 Media Query 的浏览器，在页面中调用 css3-mediaqueries.js。
- 使用 CSS Media Query 实现自适应页面设计，使用 CSS 根据分辨率宽度的变化来调整页面布局结构。
- 设计弹性图片和多媒体。通过 max-width: 100% 和 height: auto 实现图片的弹性化。通过 width: 100% 和 height: auto 实现内嵌元素的弹性化。
- 字号自动调整的问题，通过 -webkit-text-size-adjust:none 禁用 iPhone 中 Safari 的字号自动调整。

第 1 步：新建 HTML5 类型文档，编写 HTML 代码。使用 HTML5 标签来更加语义化地实现这些结构，包括页头、主要内容部分、侧边栏和页脚。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title> 无标题文档 </title>
</head>
<body>
<div id="pagewrap">
     <header id="header">
          <hgroup>
               <h1 id="site-logo">Demo</h1>
               <h2 id="site-description">Site Description</h2>
          </hgroup>
          <nav>
               <ul id="main-nav">
                    <li><a href="#">Home</a></li>
               </ul>
          </nav>
          <form id="searchform">
               <input type="search">
          </form>
     </header>
     <div id="content">
          <article class="post"> blog post </article>
     </div>
     <aside id="sidebar">
          <section class="widget"> widget </section>
     </aside>
     <footer id="footer"> footer </footer>
</div>
</body>
</html>
```

第 2 步：IE 是永恒的话题，对于 HTML5 标签，IE9 之前的版本无法提供支持。目前的最佳解决方案仍是通过 html5.js 来帮助这些旧版本的 IE 浏览器创建 HTML5 元素节点。因此，这里添加如下兼容技法，调用该 JS 文件。

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

第 3 步：设计 HTML5 块级元素样式。首先仍是浏览器兼容问题，虽然经过上一步努力已经可以在低版本的 IE 中创建 HTML5 元素节点，但还是需要在样式方面做些工作，将这些新元素声明为块级样式。

```
article,aside,details,figcaption,figure,footer,header,hgroup,menu,nav,section {
        display: block;
}
```

第 4 步：设计主要结构的 CSS 样式。这里将忽略细节样式设计，而将注意力集中在整体布局上。整体设计在默认情况下页面容器的固定宽度为 980 像素，页头部分（header）的固定高度为 160 像素，主要内容部分（content）的宽度为 600 像素，左浮动。侧边栏（sidebar）右

浮动，宽度为 280 像素。

```
<style type="text/css">
#pagewrap {
     width: 980px;
     margin: 0 auto;
}
#header { height: 160px; }
#content {
     width: 600px;
     float: left;
}
#sidebar {
     width: 280px;
     float: right;
}
#footer { clear: both; }
</style>
```

第 5 步：初步完成了页面结构的 HTML 和默认结构样式，当然，具体页面细节样式就不再烦琐，读者可以参考本节的示例源代码。

此时预览页面效果，由于还没有做任何 Media Query 方面的工作，页面还不能随着浏览器尺寸的变化而改变布局。在页面中调用 css3-mediaqueries.js 文件，解决 IE8 及其以前版本支持 CSS3 Media Query。

```
<!--[if lt IE 9]>
     <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.
      js"></script>
<![endif]-->
```

第 6 步：创建 CSS 样式表，并在页面中调用：

```
<link href="media-queries.css" rel="stylesheet" type="text/css">
```

第 7 步：借助 Media Query 技术设计响应式布局。

当浏览器可视部分宽度大于 650px、小于 980px 时（流式布局），将 pagewrap 的宽度设置为 95%，将 content 的宽度设置为 60%，将 sidebar 的宽度设置为 30%。

```
@media screen and (max-width: 980px) {
   #pagewrap { width: 95%; }
   #content {
       width: 60%;
       padding: 3% 4%;
   }
   #sidebar { width: 30%; }
   #sidebar .widget {
       padding: 8% 7%;
       margin-bottom: 10px;
   }
}
```

第 8 步：当浏览器可视部分宽度小于 650px 时（单栏布局），将 header 的高度设置为 auto；将 searchform 绝对定位在 top: 5px 的位置；将 main-nav、site-logo、site-description 的定位设置为 static；将 content 的宽度设置为 auto（主要内容部分的宽度将扩展至满屏），并取消 float 设置；将 sidebar 的宽度设置为 100%，并取消 float 设置。

```
@media screen and (max-width: 650px) {
   #header { height: auto; }
   #searchform {
        position: absolute;
        top: 5px;
        right: 0;
   }
   #main-nav { position: static; }
   #site-logo {
        margin: 15px 100px 5px 0;
        position: static;
   }
   #site-description {
        margin: 0 0 15px;
        position: static;
   }
   #content {
        width: auto;
        float: none;
        margin: 20px 0;
   }
   #sidebar {
        width: 100%;
        float: none;
        margin: 0;
   }
}
```

第 9 步：480px 是 iPhone 横屏时的宽度，当浏览器可视部分的宽度小于该数值时，禁用 HTML 节点的字号自动调整。默认情况下，iPhone 会将过小的字号放大，这里可以通过 -webkit-text-size-adjust 属性进行调整。将 main-nav 中的字号设置为 90%。

```
@media screen and (max-width: 480px) {
   html {
        -webkit-text-size-adjust: none;
   }
   #main-nav a {
        font-size: 90%;
        padding: 10px 8px;
   }
}
```

第 10 步：设计弹性图片。为图片设置 max-width: 100% 和 height: auto，实现其弹性化。对于 IE，仍然需要做一点额外的工作。

```
img {
```

```
        max-width: 100%;
        height: auto;
         width: auto\9; /* ie8 */
    }
```

第 11 步：设计弹性内嵌视频。对于视频也需要做 max-width: 100% 的设置，但是 Safari 对 embed 的该属性支持不是很好；所以使用以 width: 100% 来代替。

```
.video embed,.video object,.video iframe {
        width: 100%;
        height: auto;
        min-height: 300px;
}
```

第 12 步：iPhone 中的初始化缩放。在默认情况下，iPhone 中的 Safari 浏览器会对页面进行自动缩放，以适应屏幕尺寸。这里可以使用以下的 meta 设置，将设备的默认宽度作为页面在 Safari 的可视部分宽度，并禁止初始化缩放。

```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

3.4.6　响应式 Bootstrap

Bootstrap 完全支持响应式设计，在默认情况下 Bootstrap 是没有引入响应式特性的，因为不是任何情况都需要使用到，建议在需要使用的时候才启用它。

1. 响应式 Bootstrap 概述

响应式 Bootstrap 通过 Media Query 技术实现，相关的响应式 CSS 样式存放在 bootstrap-responsive.css 文件中，Media Query 允许在一些条件基础上自定义 CSS，不过主要是通过 min-width 和 max-width 进行设计。主要包含的定义项目：

- 修改栅格系统中列的宽度。
- 根据需要，用堆叠元素代替浮动。
- 调整标题和文本的大小以适合各种设备。

当然，Media Query 技术不是万能的，对于大型商业项目，建议选用 JavaScript 技术来解决响应式设计，可以考虑使用专门的代码库，而不是构筑在 Media Query 技术之上。

Bootstrap 支持的几个 Media Query 都放在 bootstrap-responsive.css 文件中，调用该文件可以使页面能够灵活适应不同设备和屏幕分辨率，简单说明如表 3-1 所示。

表 3-1　Bootstrap 支持的 Media Query 分类

类　型	布局宽度	列　宽	间隙宽度
大屏幕	大于或等于 1200px	70px	30px
默认	大于或等于 980px	60px	20px
平板电脑	大于或等于 768px	42px	20px
手机到平板电脑	小于或等于 767px	流式列，无固定宽度	
手机	小于或等于 480px	流式列，无固定宽度	

具体样式结构如下：

```
/* 大屏幕 */
@media (min-width: 1200px) {
}
/* 平板电脑和小屏电脑之间的分辨率 */
@media (min-width: 768px) and (max-width: 979px) {
}
/* 横向放置的手机和竖向放置的平板电脑之间的分辨率 */
@media (max-width: 767px) {
}
/* 横向放置的手机及分辨率更小的设备 */
@media (max-width: 480px) {
}
```

2. 使用响应式 Bootstrap

使用响应式 Bootstrap 的具体方法和步骤如下。

第 1 步：在需要使用响应式设计的页面中启用响应式特性，具体方法如下。

❑ 在文档头部区域内添加合适的 meta 标签。

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

❑ 引入 Bootstrap 响应式样式表，即可启用响应式 CSS。

```
<link href="bootstrap/css/bootstrap-responsive.css"rel="stylesheet" "type="text/css">
```

提示 如果已经在定制页面编译好一个 Bootstrap，那么只需添加一个 meta 标签。

第 2 步：添加响应式布局辅助类样式。具体说明如表 3-2 所示。

表 3-2 响应式布局辅助类样式

类（Class）	手机（767px 及以下）	平板电脑（979px 到 768px）	台式机（默认）
.visible-phone	显示	隐藏	隐藏
.visible-tablet	隐藏	显示	隐藏
.visible-desktop	隐藏	隐藏	显示
.hidden-phone	隐藏	显示	显示
.hidden-tablet	显示	隐藏	显示
.hidden-desktop	显示	显示	隐藏

提示 这些类样式不支持 table 元素，在有限的情况下使用 Bootstrap 响应式设计类样式，避免创建同一个站点的不同版本。当这些类样式能对每种设备的展示做有益的补充时才使用。

第 3 步：复制 3.5.4 节的案例，在样式表中删除 Media Query 部分的全部样式声明，同时清理掉结构部分的 #content、#sidebar 和 #footer 样式。同时在内部样式表上面引入 Bootstrap 响应式设计样式表。

```
<link href="bootstrap/css/bootstrap-responsive.css"rel="stylesheet" "type="
 text/css">
```

第 4 步：在 HTML 结构中，为 和 栏目包裹一层流式布局框，定义该标签的类样式为 class="row-fluid"。同时为 添加 class="span8" 类样式，为 添加 class="span4" 类样式，并添加 .hidden-phone 类样式，设计在移动设备的小屏幕中隐藏侧栏显示。

```
<div class="row-fluid">
     <div id="content" class="span8">
          <h2>Content</h2>
          <p>text</p>
     </div>
     <div id="sidebar" class="span4 hidden-phone">
          <h3>Sidebar</h3>
          <p>text</p>
     </div>
</div>
```

第 5 步：在浏览器中预览，逐步调整浏览器窗口，会发现 Bootstrap 能够自动调整页面的版式布局，如图 3-28 所示。

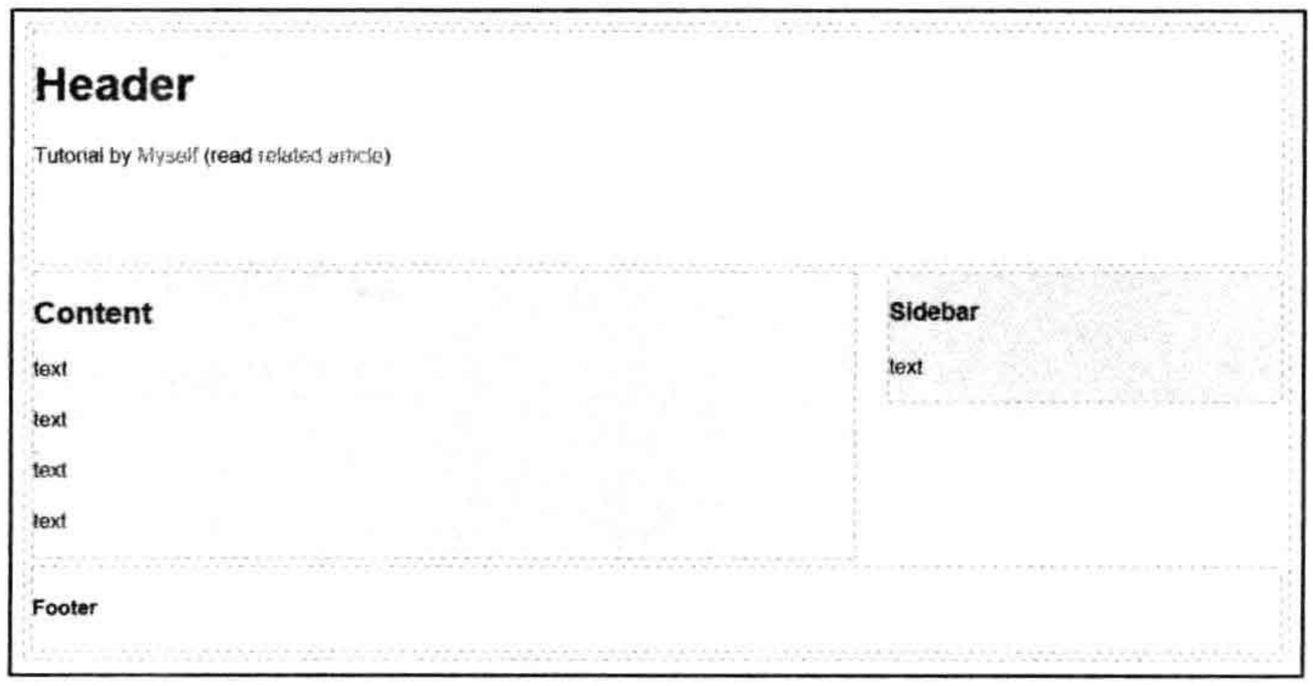


图 3-28　应用 Bootstrap 响应式设计效果

第 4 章

优化 CSS 样式

本章内容

- 页面排版优化
- 表格优化设计
- 表单优化设计
- 按钮设计
- 图片和图标设计

Bootstrap 支持所有主流浏览器，提供大量交互式组件以及常用的 jQuery 插件。使用 Bootstrap 可以设计各种形式的 Web 界面。不过，Bootstrap 核心依然是一个 CSS 框架，Bootstrap 的基础 CSS（Base CSS）提供了优雅、一致的多种基础 HTML 页面要素，包括排版、表格、表单、按钮等，能够满足前端工程师的基本设计需求。为了充分利用 Bootstrap 的强大功能，读者有必要了解其对 CSS 进行设计和优化的细节。

4.1　页面排版优化

文字是页面信息传递的主要载体，虽然使用图像、动画或视频等多媒体信息可以表情达意，但是文字所传递的信息是最准确也是最丰富的。字体和文字样式与传统印刷排版的样式相似，如字体类型、大小和颜色，段落文本的版式和样式，这些效果在网页中都可以通过 CSS 来实现。Bootstrap 通过重写标签默认样式，实现对页面版式的优化，以适应当前网页信息呈现的流行趋势。

4.1.1　标题和字体风格

1. 标题

在 Bootstrap 中，HTML 定义的所有标题标签都是可用的，从 <h1> 到 <h6>。图 4-1 比较了标题标签默认样式与 Bootstrap 样式风格。

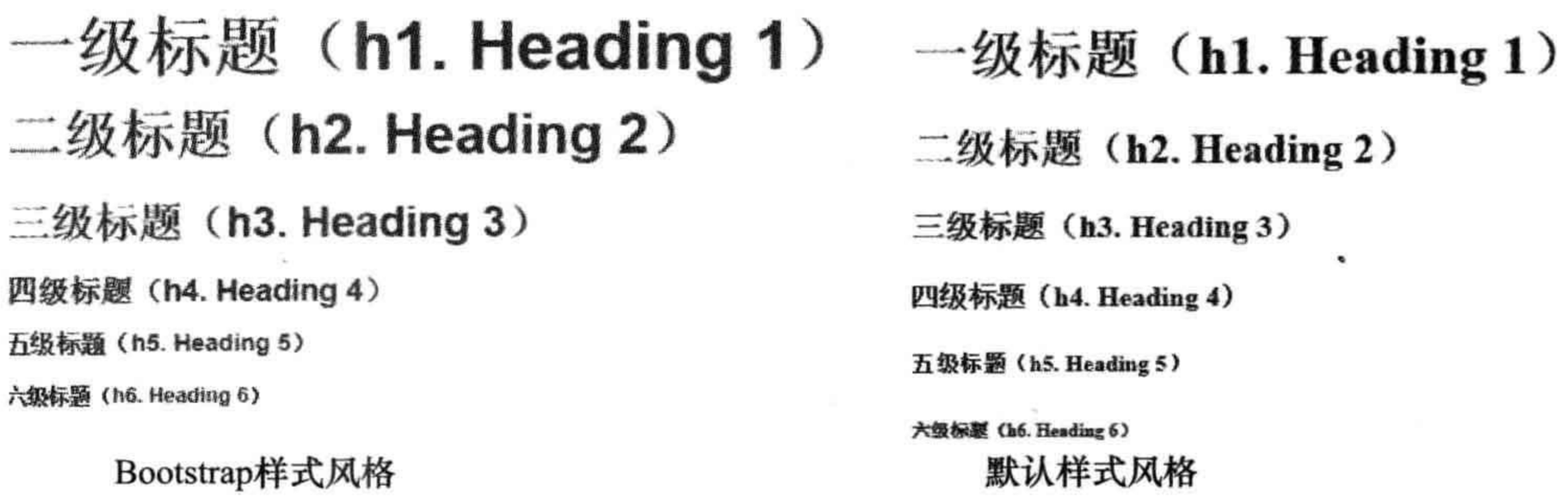


图 4-1　标题标签样式风格比较

通过比较可以发现，Bootstrap 标题样式进行了以下显著的优化重置。

- 重设上下边界为固定值，默认为一个行高距离，优化后统一为上下各为 10 像素，且不分标题级别，全部统一样式。
- 固定一级到三级标题行高为 40 像素，四级到六级标题行高为 20 像素，避免行高因标题字体大小而变化，同时也避免不同级别标题行高不一致，影响版式风格统一。
- 固定不同级别标题字体大小，一级为 38.5px，二级为 31.5px，三级为 24.5px，四级为 17.5px，五级为 14px，六级为 11.9px。
- 启用了 CSS3 中的 text-rendering 属性，优化渲染标题文字。text-rendering 属性主要用来告诉渲染引擎（Rendering Engine）渲染文字的时候如何来优化，浏览器根据这个属

性来权衡速度、易读性、几何精度等。

随着越来越多的网站开始用 @font-face 来渲染文字，易读性开始受到关注了。小号字体上，文字更容易出现。由于目前还没有 CSS 属性控制显示在线字体的微妙细节，可以利用 text-rendering 来启用 kerning 和 ligatures。Gecko 和 WebKit 浏览器处理这个属性的方式很不一样。前者默认启用这个特性，而后者需要将其设置为 optimizeLegibility。

提示 目前暂时只有 Gecko（Firefox）和 WebKit（Safari、Chrome）类型浏览器支持这个属性。

text-rendering 属性可取 4 个值，简单说明如下。

- auto：浏览器为速度、易读性、几何精度等自动优化来绘制文本。

在实践中，Gecko 桌面浏览器（Firefox）如果字体大小为 20px 或者更大，会使用 optimizeLegibility；否则对于较小的文本使用 optimizeSpeed。

- optimizeSpeed：绘制文本时速度优先，会禁用字距调整和连字。
- optimizeLegibility：绘制文本时易读性优先，会启用字距调整和连字。
- geometricPrecision：绘制文本时几何精度优先，暂时和 optimizeLegibility 相同。

明显的效果是 optimizeLegibility 在文字较小（20px 以下）的时候，在一些特殊字体设定下（如微软的 Calibri、Candara 等）启用了连字（如 ff、fi、fl）规则。

提示 另外，Bootstrap 提供了一套 small 标题样式，只要在标题文本外包裹一层 <small> 标签即可，用法代码如下，演示效果如图 4-2 所示。

```
<h1><small>一级标题（h1. Heading 1）</small></h1>
<h2><small>二级标题（h2. Heading 2）</small></h2>
<h3><small>三级标题（h3. Heading 3）</small></h3>
<h4><small>四级标题（h4. Heading 4）</small></h4>
<h5><small>五级标题（h5. Heading 5）</small></h5>
<h6><small>六级标题（h6. Heading 6）</small></h6>
```

图 4-2 small 标题风格比较

small 标题风格取消了字体粗体样式（font-weight: normal;），设置字体颜色为浅灰色（color: #999999;），行高为 1（line-height: 1;），即一个字体大小。修改一级标题的 small 风格大小为 24.5px，二级为 17.5px，三级为 14px，四级也为 14px，五级和六级保持默认大小。

2. 正文文本

Bootstrap 定义的全局 font-size 为 14px，line-height 为 20px，color 为 #333333，background-color 为 #ffffff。这些样式应用到了 <body> 标签和所有的段落上。

默认情况下，<p> 标签上下外边距（margin）保持一个行高的高度。Bootstrap 定义 <p> 标签为 1/2 行高（10px）的底部外边距属性。

```
p { margin: 0 0 10px;}
```

例如，在下面的文本中，使用 Bootstrap 定义段落文本样式，同时与默认浏览器样式风格进行对比，如图 4-3 所示。可以发现，Bootstrap 风格的段落文本显得更加清秀、温润，字距看起来更柔和，而不像默认字距那样突兀。

```
<h1> 出塞 </h1>
<h2><small>[ 王之涣 ]</small></h2>
<p> 黄河远上白云间，一片孤城万仞山。</p>
<p>Where a yellow river climbs to the white clouds,
Near the one city-wall among ten-thousand-foot mountains,</p>
<p> 羌笛何须怨杨柳，春风不度玉门关。</p>
<p>A Tartar under the willows is lamenting on his flute
That spring never blows to him through the Jade Pass.</p>
```


图 4-3　段落标签样式风格比较

通过添加 lead 类样式，可以定义段落突出显示，如图 4-4 所示。被突出的段落文本字号被放大，字距和行高也都被显著放大。

图 4-4　突出段落文本样式

提示 用户可以用 LESS 构建正文字体大小、行高和边距等基本属性。在 variables.less 文件中，定义的两个 LESS 变量决定了排版尺寸，如图 4-5 所示。

- @baseFontSize：该变量定义了全局 font-size 基准。
- @baseLineHeight：该变量是 line-height 基准。

```
// Typography
// -------------------------
@sansFontFamily:        "Helvetica Neue", Helvetica, Arial, sans-serif;
@serifFontFamily:       Georgia, "Times New Roman", Times, serif;
@monoFontFamily:        Monaco, Menlo, Consolas, "Courier New", monospace;

@baseFontSize:          14px;
@baseFontFamily:        @sansFontFamily;
@baseLineHeight:        20px;
@altFontFamily:         @serifFontFamily;
```

图 4-5 预定义页面基准字体大小和行高

使用这些变量和一些简单的公式，能够计算出其他所有页面元素的 margin、padding 和 line-height。自定义这些变量即可改变 Bootstrap 的默认样式。

4.1.2 文本强调风格

1. 强调类文本

为了说明正文部分的某些字词、句子的重要性，应该通过 HTML 强调代码来标识它们，将其与其他字词或者句子区分开来。

HTML 默认定义 标签为强调语义，一般被 标签包括了的字词或句子在网页中表现为斜体样式。另外，HTML 还定义了两个重点强调的标签：<b> 和 ，使用它们，被重点强调标签包括的字词或句子在网页中表现为粗体。<b> 主要侧重于表现视觉上的强调，而 则是指语意上的强调。

建立在这些强调语义的标签之上，Bootstrap 定义了一套强调类，这些表示强调的工具类通过颜色来表示强调，具体说明如下。

- .muted：提示，浅灰色。
- .text-warning：警告，黄色。
- .text-error：错误，红色。
- .text-info：通知信息，浅蓝色。
- .text-success：成功，浅绿色。

例如，在文档中输入下面段落文本，并使用 Bootstrap 强调类提示工具标识文本的性质，代码如下，演示效果如图 4-6 所示。

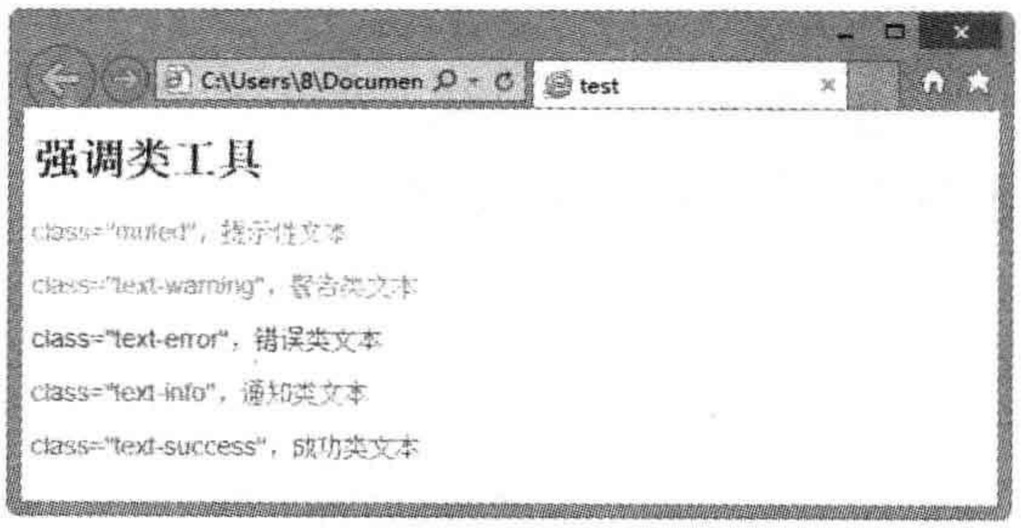


图 4-6 强调类文本效果

```
<h3> 强调类工具 </h3>
<p class="muted">class="muted", 提示性文本 </p>
<p class="text-warning">class="text-warning", 警告类文本 </p>
<p class="text-error">class="text-error", 错误类文本 </p>
```

```
<p class="text-info">class="text-info", 通知类文本 </p>
<p class="text-success">class="text-success", 成功类文本 </p>
```

Bootstrap 定义 <small> 标签用来标识不需要强调的文本。<small> 标签将缩减文本大小到 85%，样式如下：

```
small {
  font-size: 85%;
}
```

对于不需要强调的行内文本（inline）或者块状文本（block），建议使用 <small> 标签。

提示　以上介绍的几个强调语义标签表达的文本逻辑性权重很大，所以其配合表现的字词句段在搜索引擎的文本分析中占有极大的评估优势，这也就不难理解为什么很多的 SEO 对它们十分重视了。网页设计师在编写代码的时候应当尽量使用丰富多样的标签，让网页的文本信息具有逻辑性，从而让搜索引擎能更好地读“懂”网页中的信息。对于搜索引擎优化人员来说，在这些标签中部署关键词是一个不错的选择。

---

2. 加粗和斜体文本

使用 font-weight 属性可以定义字体粗细。用增加 font-weight 值的方式加粗强调一段文本。

```
strong {
  font-weight: bold;
}
```

font-weight 属性取值比较特殊，默认值 normal 表示正常的字体，相当于取值为 400。bold 关键字表示粗体，相当于取值为 700，或者使用 标签定义的字体效果。bolder(较粗）和 lighter（较细）是相对于 normal 字体粗细而言的。

例如，下面的代码使用 标签定义了一行强调文本。

```
<p><strong> 加粗强调文本 </strong></p>
```

使用 font-style 属性可以定义字体倾斜效果，其默认值 normal，表示正常的字体，italic 表示斜体，oblique 表示倾斜的字体。

```
em {
  font-style: italic;
}
```

斜体与加粗一样，都可以用来强调文本。 标签负责定义斜体强调效果，与 标签对应使用。

```
<p><em> 斜体强调文本 </em></p>
```

提示　在 HTML5 中可任意使用 <b> 和 <i> 标签定义强调文本，默认情况下，<b> 标签会加粗文本显示，而 <i> 标签会让文本斜体显示。<b> 是为了高亮词或短语而不会赋予重要含义，<i> 主要被用来表示发言、技术术语等。

4.1.3 文本对齐风格

在传统布局中，一般使用 HTML 的 align 属性来定义对象水平对齐，这种用法在过渡型文档类型中依然可以使用。CSS 使用 text-align 属性来定义文本的水平对齐方式，该属性取值包括 4 个：其中 left 为默认值，表示左对齐；right 表示右对齐；center 表示居中对齐；justify 表示两端对齐。

为了简化操作，方便使用，Bootstrap 定义了 3 个对齐类样式：

```
.text-left { text-align: left;}
.text-right { text-align: right;}
.text-center { text-align: center;}
```

它们分别用来表示文本左对齐、右对齐和居中对齐。例如，下面 3 行代码分别定义文本左对齐、居中对齐和右对齐效果。

```
<p class="text-left"> 文本左对齐 </p>
<p class="text-center"> 文本居中对齐 </p>
<p class="text-right"> 文本右对齐 </p>
```

4.1.4 缩略语风格

当鼠标悬停在缩写语上时会显示完整内容，Bootstrap 实现了对 HTML 中 <abbr> 标签的增强样式。

缩略语标签应该带有 title 属性，其外观表现为带有较浅的虚线框，鼠标移至上面时会变成带有“问号”的指针。

```
abbr[title],
abbr[data-original-title] {
  cursor: help;
  border-bottom: 1px dotted #999999;
}
```

例如，下面的代码在浏览器中的预览效果如图 4-7 所示。如想看 CSS 完整的内容，可以把鼠标悬停在缩略语上，即可显示提示性文本。但是，<abbr> 需要包含 title 属性。

```
<p><abbr title="Cascading Style Sheets">CSS</abbr> 是英语层叠样式表单的缩写，它是一种
   用来表现 HTML 或 XML 等文件样式的计算机语言。
</p>
```

另外，Bootstrap 为 <abbr> 标签添加了一个 initialism 类，使用该类可以设计更小一些的字号，字体大小缩小 10%，同时设置字母全部大写显示，定义的样式如下。

```
abbr.initialism {
  font-size: 90%;
  text-transform: uppercase;
}
```

例如，针对上面示例，为 <abbr> 标签添加 initialism 类，则预览效果如图 4-8 所示，缩略词被缩小，同时全部大写显示。

```
<p><abbr class="initialism" title="Cascading Style Sheets">CSS</abbr>是英语层叠样
   式表单的缩写，它是一种用来表现 HTML 或 XML 等文件样式的计算机语言。
</p>
```

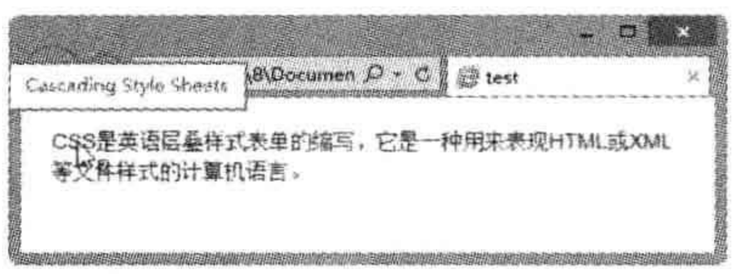


图 4-7　缩略语效果

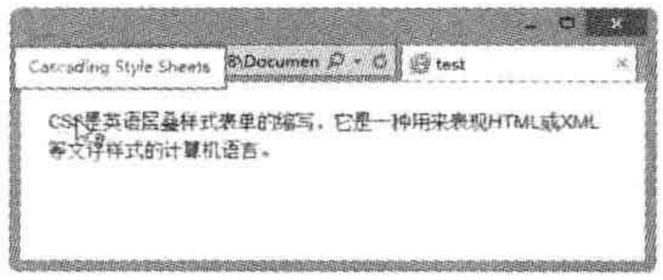


图 4-8　为缩略语应用 initialism 类效果

4.1.5　地址风格

<address> 标签可定义地址（如电子邮件地址）。一般使用它来定义地址、签名或者文档的作者身份。不论创建的文档是简短扼要还是冗长完整，都应该确保每个文档都附加了一个地址，这样做不仅为读者提供了反馈的渠道，还可以增加文档的可信度。

Bootstrap 优化了 <address> 标签样式，让联系信息以最接近日常使用的格式呈现。定义 <address> 标签以块状显示，设置底部外边距为 20 像素，清理默认的字体样式，设置行高为 20 像素。样式代码如下：

```
address {
  display: block;
  margin-bottom: 20px;
  font-style: normal;
  line-height: 20px;
}
```

在 <address> 标签内部，如果在每行结尾添加
 标签可以保留需要的样式。例如，下面的代码通过 <address> 标签定义一组客户联系信息，并通过
 标签实现联系信息多行显示，如图 4-9 所示。

```
<address>
    <a href="mailto:yidingny@163.com "> 一定能赢信箱 </a><br />
    一定能赢投资有限公司 <br />
    北京东城区 689 号 <br />
</address>
```

4.1.6　引用风格

<blockquote> 标签定义摘自另一个源的块引用。在 <blockquote> 与 </blockquote> 之间的所有文本都会从常规文本中分离出来，经常会在左、右两边进行缩进，而且有时会使用斜体。也就是说，块引用拥有它们自己的空间。

如果标识提示、注释等简短的引用，建议使用 <q> 标签进行设计。如果直接引用，建议使用 <p> 标签。

Bootstrap 优化了 <blockquote> 标签样式，重新定义了 padding 和 margin 属性值，清除了左右缩进样式，设计底部外边距为 20 像素，通过在左侧添加灰色粗边框线，设计一种引用效果，具体样式代码如下：

```
blockquote {
  padding: 0 0 0 15px;
  margin: 0 0 20px;
  border-left: 5px solid #eeeeee;
}
```

例如，在下面示例代码中，通过 <blockquote> 标签引用一段文本介绍，演示效果如图 4-10 所示。

```
<blockquote cite="http://www.w3school.com.cn/">
    <p> 全球最大的中文 Web 技术教程。在 w3school，你可以找到你所需要的所有的网站建设教程。
       从基础的 HTML 到 XHTML，乃至进阶的 XML、SQL、数据库、多媒体和 WAP。</p>
</blockquote>
```

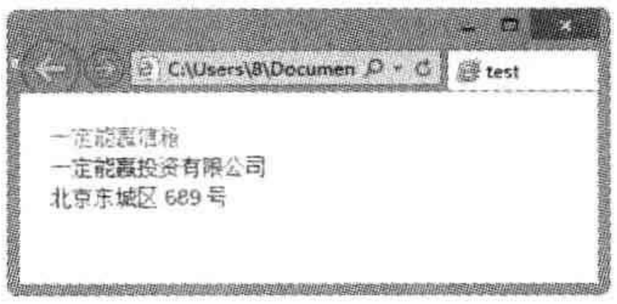



图 4-9　地址信息样式优化效果

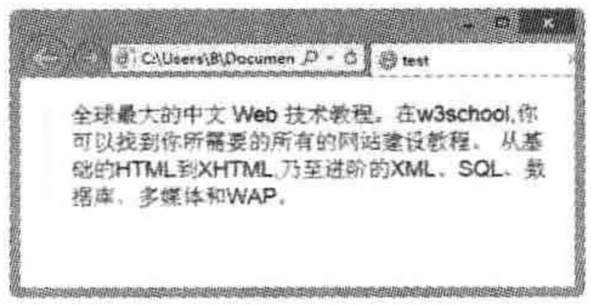


图 4-10　引用文本样式优化效果

也可以命名来源，通过添加 <small> 标签来注明引用来源，来源名称可以放在 <cite> 标签里面。例如，针对上面示例代码，可以进行如下修改，预览效果如图 4-11 所示。

```
<blockquote>
   <p> 全球最大的中文 Web 技术教程。在 w3school，你可以找到你所需要的所有的网站建设教程。从
      基础的 HTML 到 XHTML，乃至进阶的 XML、SQL、数据库、多媒体和 WAP。</p>
    <small> 来源于 <cite title="http://www.w3school.com.cn/"><a href="http://www.w3school.
     com.cn/" target="_blank">W3School</a></cite></small>
</blockquote>
```

还可以使用 pull-right 类，展示另一种样式风格：让引用展现出向右侧移动、对齐的效果。例如，为上面示例中 <blockquote> 添加 pull-right 类，则引用效果如图 4-12 所示。

```
<blockquote class="pull-right">
    <p> 全球最大的中文 Web 技术教程。在 w3school，你可以找到你所需要的所有的网站建设教程。
       从基础的 HTML 到 XHTML，乃至进阶的 XML、SQL、数据库、多媒体和 WAP。</p>
    <small> 来源于 <cite title="http://www.w3school.com.cn/"><a href="http://www.
      w3school.com.cn/" target="_blank">W3School</a></cite></small>
</blockquote>
```

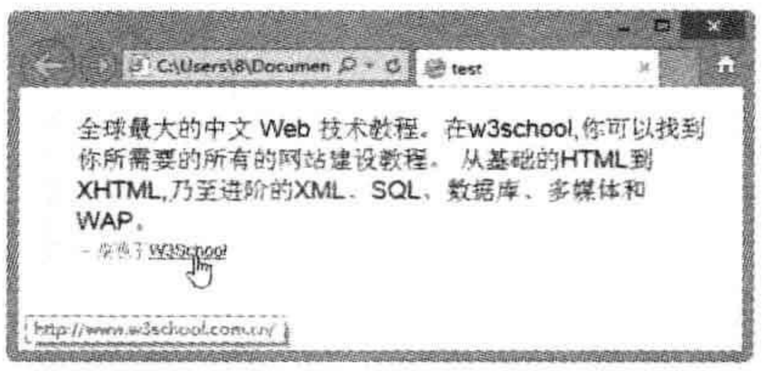


图 4-11　使用 <small> 标明引用源效果

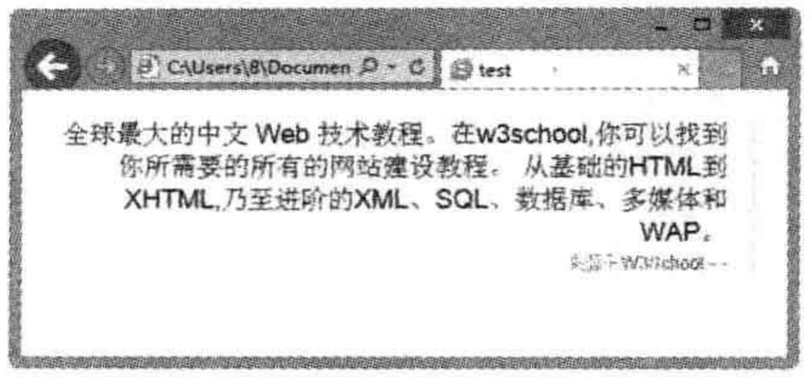


图 4-12　为引用应用 pull-right 类样式效果

4.1.7　列表风格

在 HTML 文档中，列表结构可以分为两种类型：有序列表和无序列表。无序列表使用项目符号来标识列表，而有序列表则使用编号来标识列表的项目顺序。具体使用标签说明如下：

- ...：标识无序列表。
- ...：标识有序列表。
- ...：标识列表项目。

列表样式在默认状态下，呈现缩进显示，并带有列表项目符号。Bootstrap 定义了 unstyled 类样式，使用它可以移除默认的 list-style 样式，清理左侧填充，并允许对直接子节点列表项呈现默认样式。

例如，在下面的嵌套列表结构中，为外层 <ul> 标签引用 unstyled 类样式，则预览效果如图 4-13 所示。

```
<ul class="unstyled">
    <li> 首页 </li>
    <li> 二手车 </li>
    <li> 二手市场
        <ul>
            <li> 二手电脑 / 配件、笔记本 </li>
            <li> 数码产品、数码相机 </li>
            <li> 二手手机、手机号码 </li>
            <li> 二手家电、二手家具 </li>
        </ul>
    </li>
    <li> 二手房 </li>
</ul>
```

如果让列表项目水平分布，则需要定义行内列表，使用 display: inline-block; 让列表项水平排列。为此，Bootstrap 定义了 inline 类样式，同时设置每项都有少量的内补（padding）。

```
ul.inline,
ol.inline {
   margin-left: 0;
   list-style: none;
}
ul.inline > li,
ol.inline > li {
```

```
    display: inline-block;
    *display: inline;
    padding-right: 5px;
    padding-left: 5px;
    *zoom: 1;
}
```

其中，*display: inline; 和 *zoom: 1; 声明是为了解决 IE 早期版本不兼容 display: inline-block; 的问题。例如，针对上面的列表结构，为外层 <ul> 标签引用 inline 类，同时清理掉嵌套的内层列表结构，则在浏览器中预览效果如图 4-14 所示。

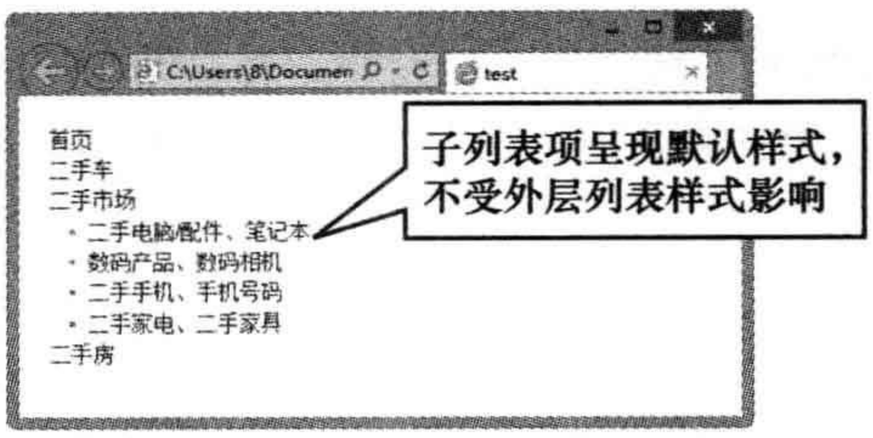


图 4-13　为列表样式引用 unstyled 类样式

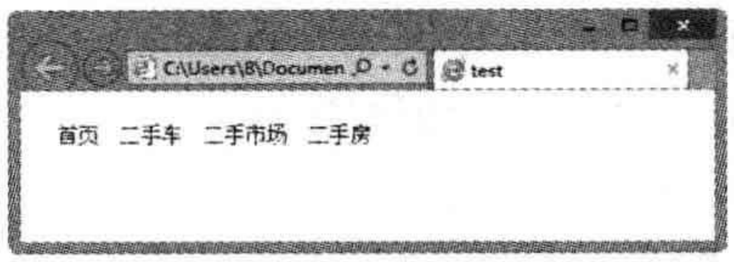


图 4-14　为列表样式引用 inline 类样式

定义列表是一种特殊的结构，它包括词条和解释两块内容。包含的标签说明如下：

- <dl>...</dl>：标识定义列表。
- <dt>...</dt>：标识词条。
- <dd>...</dd>：标识解释。

Bootstrap 优化了定义列表样式，加粗显示词条（<dt>），重设了定义列表缩进和间距，使定义列表默认样式看起来更实用，如图 4-15 所示。

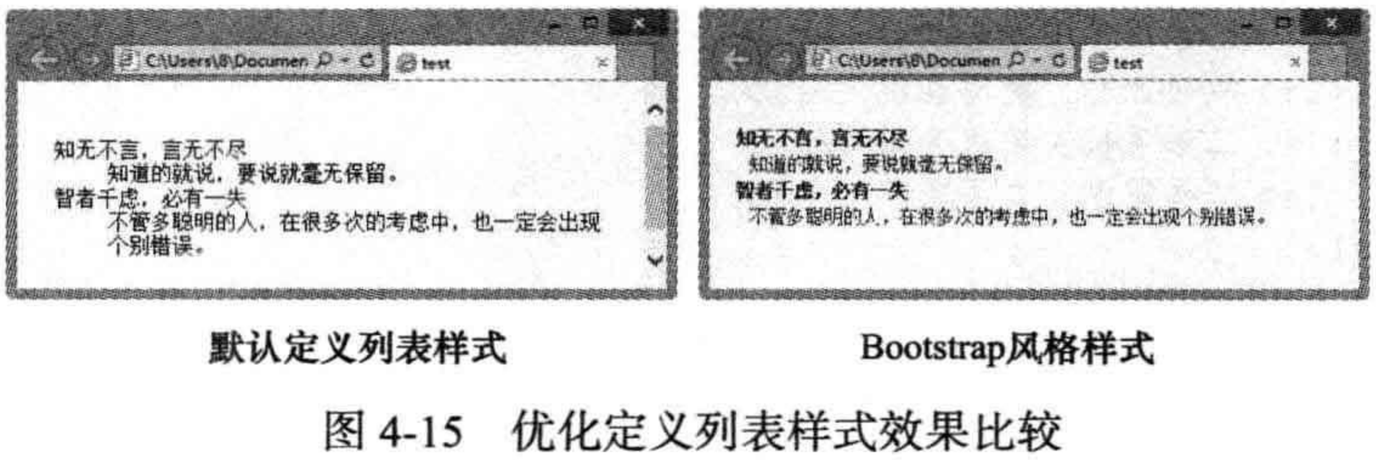


图 4-15　优化定义列表样式效果比较

通过 dl-horizontal 类样式，可以让解释与词条并列显示。例如，在下面定义列表结构中，为 <dl> 标签引用 dl-horizontal 类样式，则显示效果如图 4-16 所示。

```
<dl class="dl-horizontal">
    <dt> 知无不言，言无不尽 </dt>
    <dd> 知道的就说，要说就毫无保留。</dd>
    <dt> 智者千虑，必有一失 </dt>
    <dd> 不管多聪明的人，在很多次的考虑中，也一定会出现个别错误。</dd>
</dl>
```

提示　通过引入 text-overflow 类样式，会将水平定义列表过长而无法在左栏中完全显示的列名截掉一部分。而在较窄的视口（宽度）中，会改变成以垂直形式显示，来适应当前屏幕。

4.1.8　代码风格

1. 行内代码

<code> 标签用于表示计算机源代码或者其他机器可以阅读的文本内容。开发人员习惯了编写源代码时文本表示的特殊样式，包含在该标签内的文本将用等宽、类似电传打字机样式的字体显示出来，对于大多数程序员来说，这是十分熟悉的。

虽然 <code> 标签通常只是把文本变成等宽字体，但它暗示着这段文本是源程序代码。将来的浏览器有可能会加入其他显示效果。例如，程序员的浏览器可能会寻找 <code> 片段，并执行某些额外的文本格式化处理，如循环和条件判断语句的特殊缩进等。

Bootstrap 优化了 <code> 标签默认样式效果，定义行内代码呈现：灰色背景、灰色边框和红色字体效果。

```
code {
  padding: 2px 4px;
  color: #d14;
  white-space: nowrap;
  background-color: #f7f7f9;
  border: 1px solid #e1e1e8;
}
```

例如，下面段落文本中，通过 <code> 标签标识代码文本 “ $i=1;”，预览效果如图 4-17 所示。

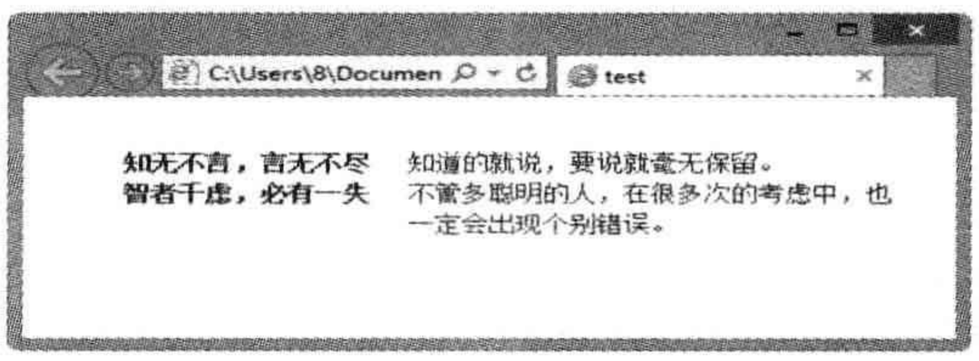


图 4-16　应用 dl-horizontal 类样式效果

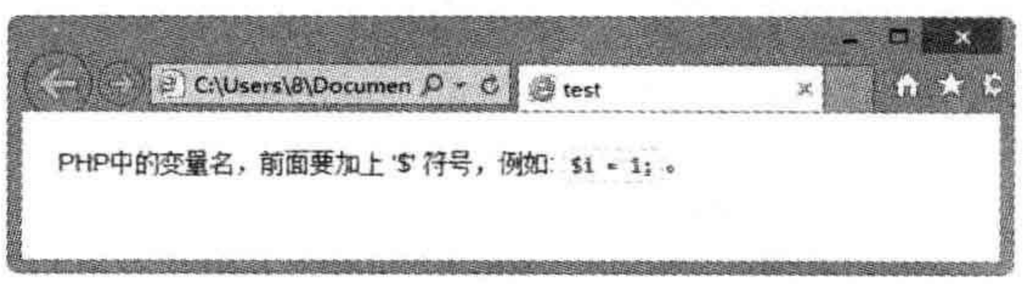


图 4-17　优化后 <code> 标签样式效果

```
<p>PHP 中的变量名，前面要加上 '$' 符号，例如：<code>$i=1;</code>。</p>
```

提示　如果只是希望使用等宽字体的效果，建议使用 <tt> 标签。如果想要在严格限制为等宽字体格式的文本中显示编程代码，建议使用 <pre> 标签。

2. 代码块

<pre> 标签可定义预格式化的文本。被包围在 <pre> 标签中的文本通常会保留空格和换

行符。而文本也会呈现为等宽字体效果。

注意，可能导致段落断开的标签（如标题、<p> 和 <address> 标签）绝不能包含在 <pre> 所定义的块里。<pre> 标签中允许的文本可以包括物理样式和基于内容的样式变化，还有链接、图像和水平分隔线。当把其他标签（如 <a> 标签）放到 <pre> 块中时，就像放在 HTML 文档的其他部分中一样即可。

Bootstrap 优化了 <pre> 标签默认样式效果，定义代码块呈现：灰色背景、灰色圆角边框和深色字体效果。具体优化样式代码如下：

```
pre {
  display: block;
  padding: 9.5px;
  margin: 0 0 10px;
  font-size: 13px;
  line-height: 20px;
  word-break: break-all;
  word-wrap: break-word;
  white-space: pre;
  white-space: pre-wrap;
  background-color: #f5f5f5;
  border: 1px solid #ccc;
  border: 1px solid rgba(0, 0, 0, 0.15);
  -webkit-border-radius: 4px;
     -moz-border-radius: 4px;
          border-radius: 4px;
}
```

例如，下面的段落文本中，通过 <pre> 标签定义代码段，预览效果如图 4-18 所示。

```
<h2>Bootstrap 网页模板 </h2>
<pre>
&lt;!doctype  html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta  charset=&quot;utf-8&quot;&gt;
&lt;title&gt;Bootstrap 应用模板 &lt;/title&gt;
&lt;meta  name=&quot;viewport&quot; content=&quot;width=device-width,  initial-
 scale=1.0&quot;&gt;
&lt;link  href=&quot;bootstrap/css/bootstrap.min.css&quot;
 rel=&quot;stylesheet&quot;  type=&quot;text/css&quot;&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;!-- 网页内容 --&gt;
&lt;script  src=&quot;http://code.jquery.com/jquery.js&quot;&gt;&lt;/script&gt;
&lt;script  src=&quot;bootstrap/js/bootstrap.min.js&quot;&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>
```

图 4-18　优化后 <pre> 标签样式效果

提示　使用 <pre> 标识多行代码时，为了能够正确展示，务必将代码中的任何尖括号进行转义。在 <pre> 标签里，Tab 键会被算进去，所以要保持代码尽可能地靠左侧。

使用 pre-scrollable 样式类，可以把该区域设置成最大高度为 350px，并带有一个 Y 轴滚动条。例如，在上面的示例代码中，为 <pre> 标签引用样式类（<pre class="pre-scrollable">），则在浏览器中预览效果如图 4-19 所示。

图 4-19　优化后 <pre> 标签 pre-scrollable 样式类效果

4.2 表格优化设计

表格拥有特殊的结构和布局模型，能够简明、直观地描述数据间的逻辑关系，如果借助CSS 设计表格样式，可以帮助用户在阅读数据时更了然、更轻松。Bootstrap 优化了表格在数据呈现上的风格，并添加了很多表格专用样式类。

4.2.1 优化表格结构

表格结构包含众多标签，这些标签各司其职，负责二维数据的平面呈现。Bootstrap 优化了表格结构标签，仅支持下面表格标签的应用，及其样式优化设计。

- <table>：定义表格容器，构建表格数据的框架。
- <thead>：定义表头容器。
- <tbody>：定义表格内容容器。
- <tr>：定义数据行结构。
- <td>：定义表格数据的单元，即定义单元格。
- <th>：定义每列（或行，取决于放置的位置）所对应的标签（label），即定义列标题或行标题。
- <caption>：定义表格标题，用于对表格进行描述或总结，这对屏幕阅读器特别有用。

其他表格标签（如 <tfooter>、<colgroup> 和 <col> 标签）依然可以继续使用，但是 Bootstrap 不再提供样式优化。

Bootstrap 支持的表格标签在表格结构中的顺序和位置如下所示：

```
<table>
    <caption>...</caption>
    <thead>
        <tr>
            <th>...</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>...</td>
        </tr>
    </tbody>
</table>
```

4.2.2 默认风格

Bootstrap 优化了 <table> 标签的表现效果，通过 border-collapse: collapse; 声明，定义表格单线显示；通过 border-spacing: 0; 声明，清除表格内边距；通过 max-width: 100%; 声明，定义表格 100% 宽度显示。

同时，为 <table> 标签定义了一个基本样式类 table，引用该样式类，将会为表格 <table> 标签增加基本样式：很少的内补（padding）空间、灰色的细水平分隔线。

```
.table {
    width: 100%;
    margin-bottom: 20px;
}
.table th,
.table td {
    padding: 8px;
    line-height: 20px;
    text-align: left;
    vertical-align: top;
    border-top: 1px solid #dddddd;
}
```

例如，在下面示例中，直接为表格引用 table 样式类，则表格呈现效果如图 4-20 所示。

```
<h2> 华语九天榜 </h2>
<table class="table">
      <tr><th> 排名 </th> <th> 歌曲名 </th><th> 歌手名 </th></tr>
      <tr><td>1</td><td> 我，一个人 </td><td> 付辛博 </td> </tr>
      <tr><td>2</td><td> 他们 </td><td> 张惠妹 </td></tr>
      <tr><td>3</td><td> 伤不起 </td><td> 郁可唯 </td></tr>
      <tr><td>4</td><td> 如果有如果 </td><td> 邓福如 </td></tr>
      <tr><td>5</td><td> 狂想曲 </td><td> 萧敬腾 </td></tr>
      <tr><td>6</td><td> 越来越想爱上你 </td><td>SD5 行堂 </td></tr>
      <tr><td>7</td><td> 妈妈咪呀 !</td><td> 张靓颖 </td></tr>
      <tr><td>8</td><td> 除下吊带前 </td><td> 薛凯琪 </td></tr>
      <tr><td>9</td><td> 如梦令（电影《大武生》主题曲）</td><td> 韩庚 </td></tr>
      <tr><td>10</td><td> 因为爱情 </td><td> 陈奕迅 </td></tr>
</table>
```

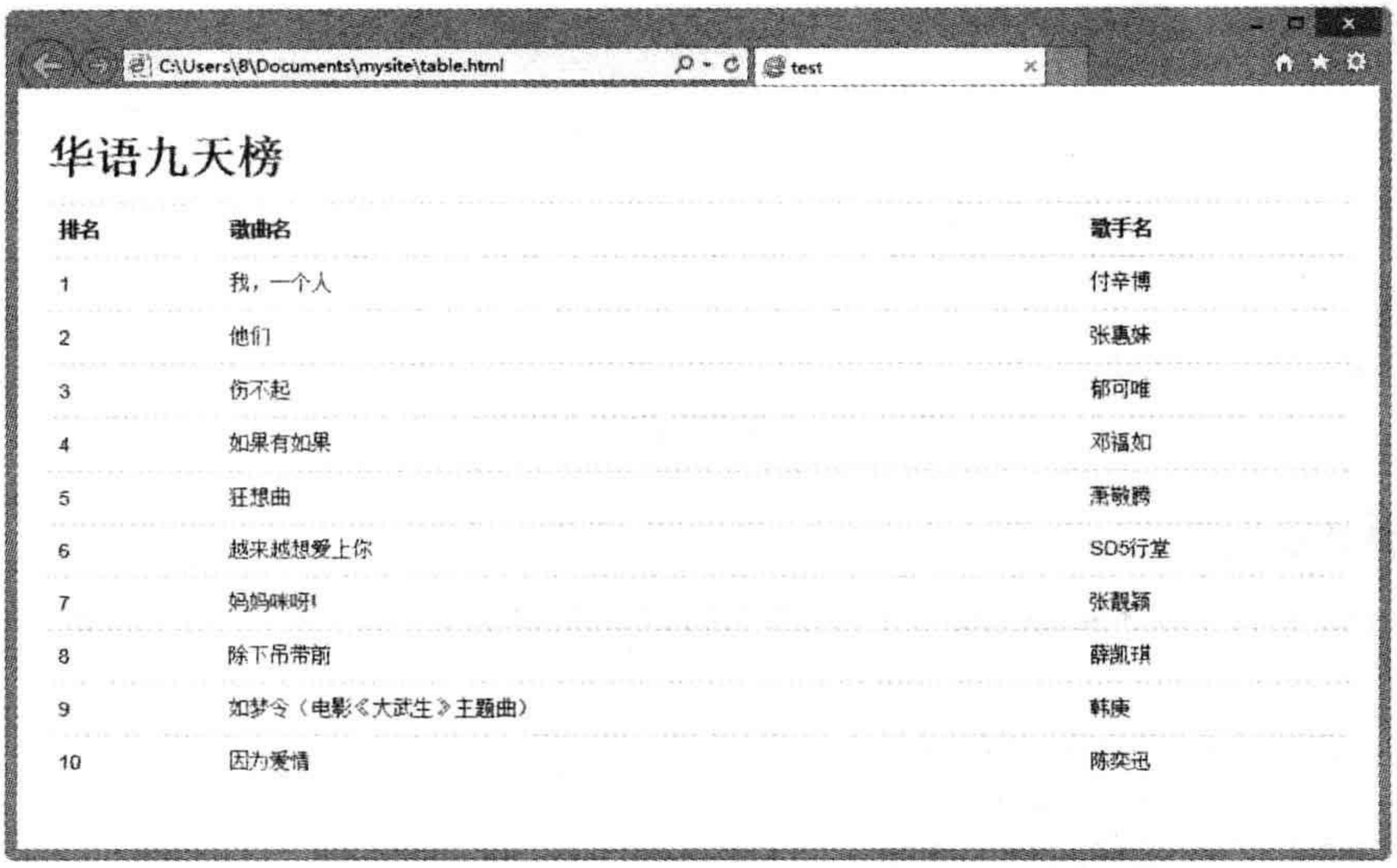


图 4-20　优化后的 <table> 标签风格

4.2.3 表格个性风格

除了基本的 table 样式类外，Bootstrap 补充了多种表格风格样式类，下面进行简单说明。

1. 斑马纹风格

Bootstrap 定义了 table-striped 类样式，设计斑马纹样式，即实现表格数据行隔行换色效果。

```
.table-striped tbody > tr:nth-child(odd) > td,
.table-striped tbody > tr:nth-child(odd) > th {
   background-color: #f9f9f9;
}
```

上面样式在 <tbody> 内，通过 CSS 的：nth-child 选择器为表格中的行添加奇数行加背景色样式，由于 IE8 及其以下版本浏览器不支持：nth-child 选择器，因而在应用时，要考虑兼容性的处理方法。

例如，针对上面的表格结构，为 <table> 标签引用 table-striped 类样式（<table class="table table-striped">），则表格显示效果如图 4-21 所示。

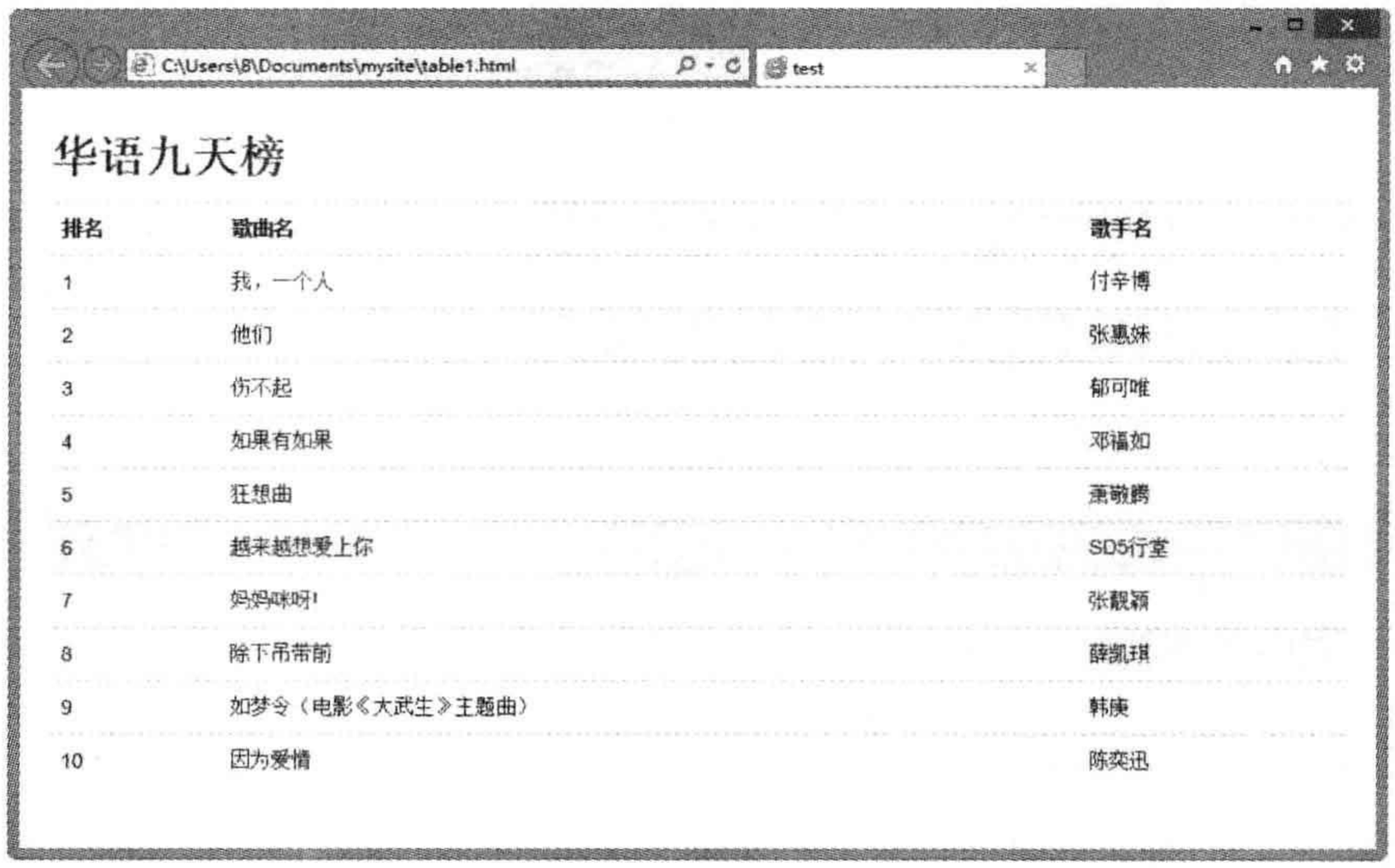


图 4-21 表格的斑马纹风格

2. 圆角边框风格

Bootstrap 通过 table-bordered 类设计圆角边框表格样式，该样式类的代码如下：

```
.table-bordered {
   border: 1px solid #dddddd;
   border-collapse: separate;
   *border-collapse: collapse;
   border-left: 0;
   -webkit-border-radius: 4px;
```

```
        -moz-border-radius: 4px;
             border-radius: 4px;
}
```

例如，针对上面的表格结构，为 <table> 标签引用 table-bordered 类样式（<table class="table table-striped table-bordered">），则表格显示效果如图 4-22 所示。

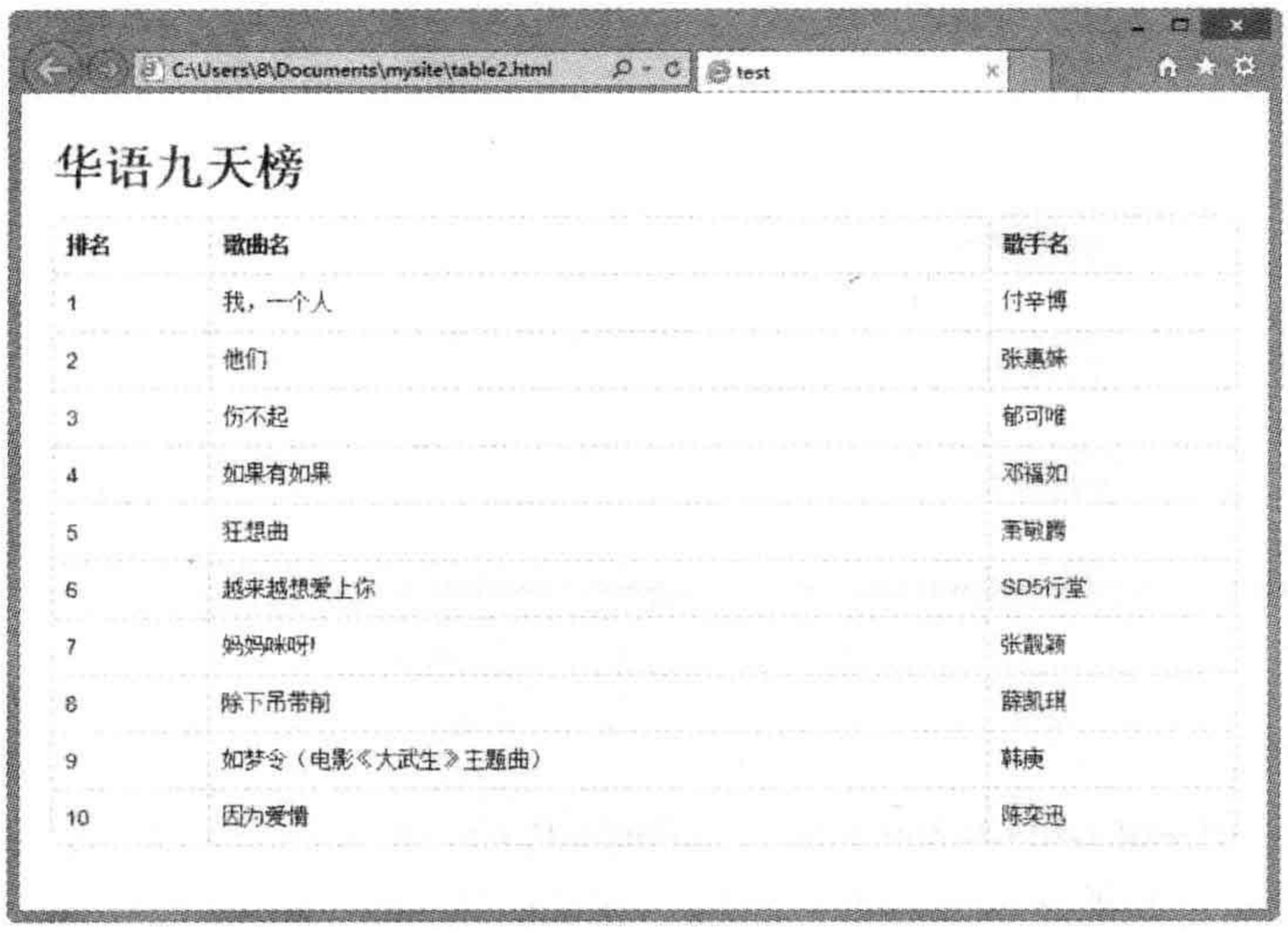


图 4-22　表格的圆角边框风格

3. 鼠标悬停风格

Bootstrap 通过 table-hover 类设计为 <tbody> 标签中的每一行赋予鼠标悬停样式，该样式类的代码如下：

```
.table-hover tbody tr:hover > td,
.table-hover tbody tr:hover > th {
   background-color: #f5f5f5;
}
```

例如，针对上面的表格结构，为 <table> 标签引用 table-hover 类样式（<table class="table table-striped table-bordered table-hover">），则表格显示效果如图 4-23 所示。鼠标经过数据行时，该行背景色显示效果与斑马纹背景效果相同。

4. 紧凑单元格风格

Bootstrap 通过 table-condensed 类设计为 <table> 标签中的每个单元格的内补（padding）减半，从而设计紧凑型表格样式。该样式类的代码如下：

```
.table-condensed th,
.table-condensed td {
   padding: 4px 5px;
}
```

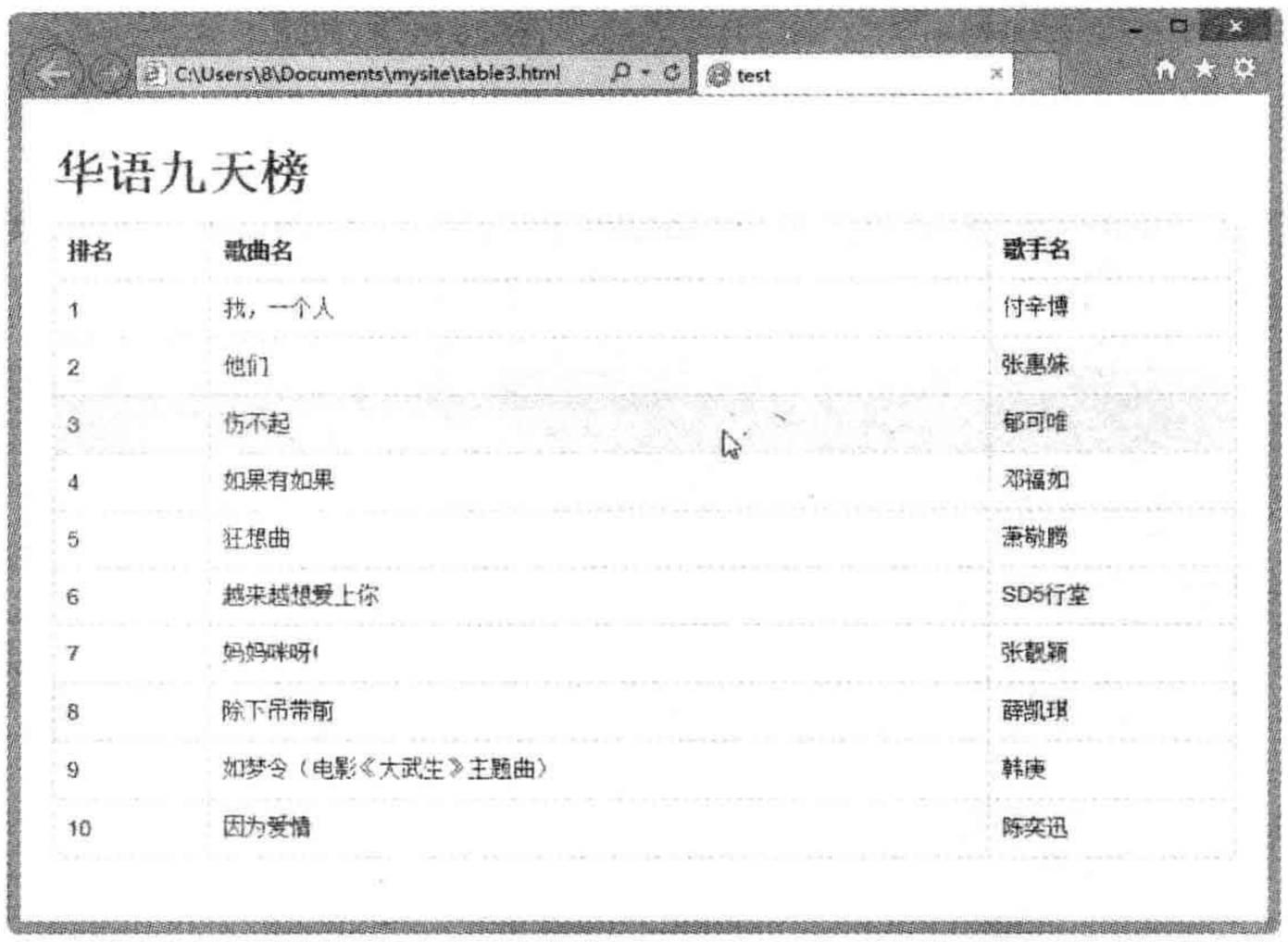


图 4-23　表格的鼠标悬停风格

例如，针对上面的表格结构，为 <table> 标签引用 table-hover 类样式（<table class="table table-striped table-bordered table-hover table-condensed">），则表格显示效果如图 4-24 所示。此时表格显得非常紧凑，如果显示大容量数据时，建议引用 table-hover 类样式。

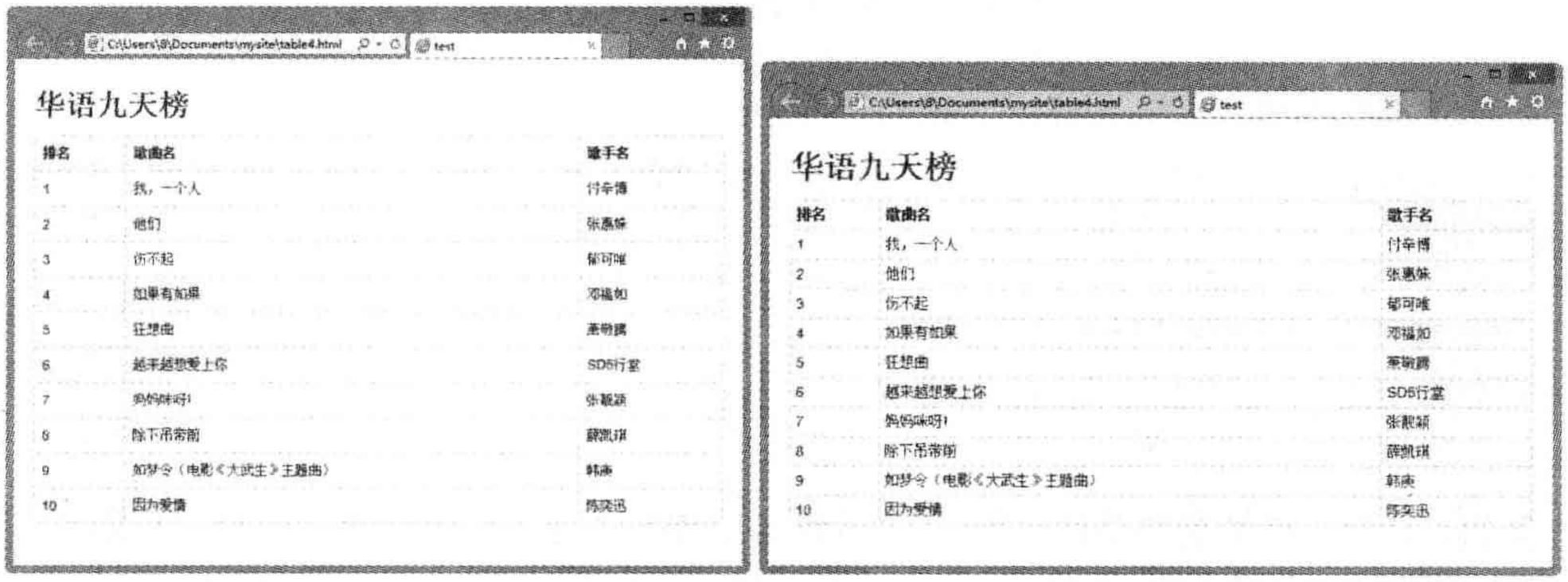


图 4-24　设计紧凑型表格风格

4.2.4　表格行风格

Bootstrap 为表格行增加了多个情景类样式，通过选择不同的情景类为表格添加特殊背景颜色。表格行情景类说明如下。

- success：表示成功或积极的行为。
- error：表示一个危险或存有潜在危险的行为。

- warning：表示警告，可能需要注意。
- info：作为默认样式的一个替代样式。

例如，在下面示例中，分别为每行引用不同的情景类样式，代码如下，显示效果如图 4-25 所示。

```
<h2> 表格情景类样式 </h2>
<table class="table table-bordered table-hover">
    <tr> <th> 样式类 </th><th> 说明 </th></tr>
    <tr class="success"><td>success</td><td> 表示成功或积极的行为。</td> </tr>
    <tr class="error"><td>error</td><td> 表示一个危险或存有潜在危险的行为。</td></tr>
    <tr class="warning"><td>warning</td><td> 表示警告，可能需要注意。</td></tr>
    <tr class="info"><td>info</td><td> 作为一个默认样式的一个替代样式。</td></tr>
</table>
```

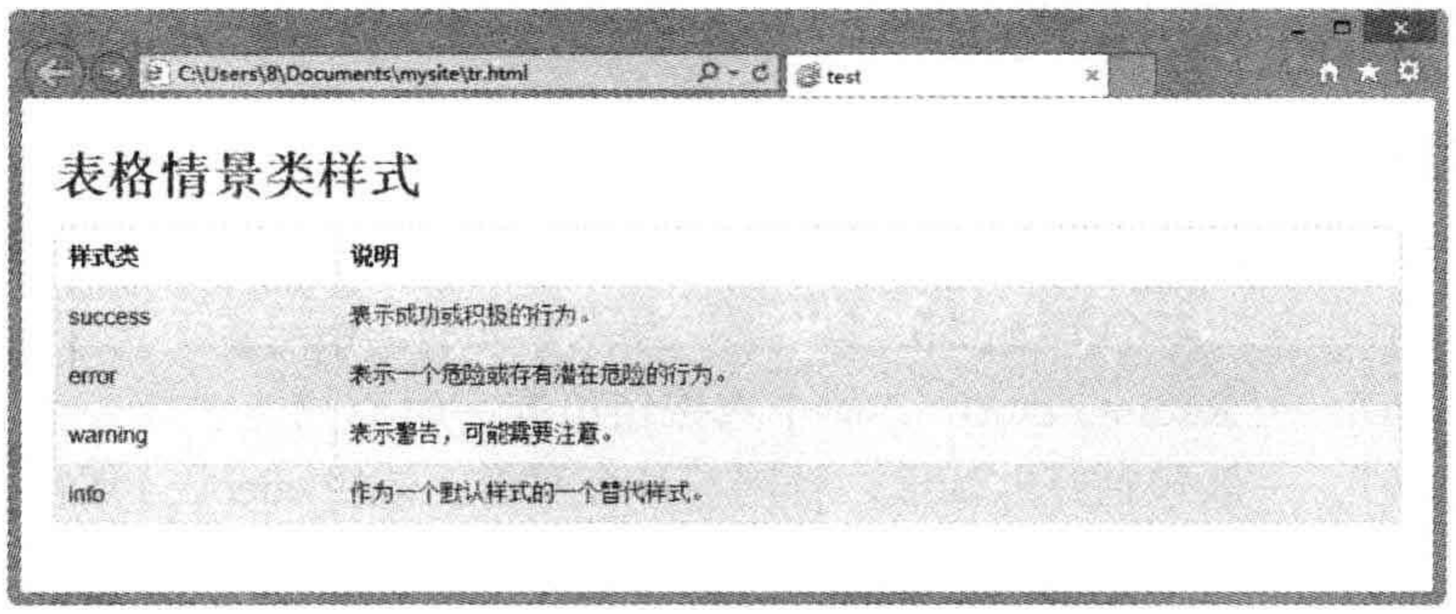


图 4-25　引用情景类的表格风格效果

提示　如果同时为表格引用情景类样式和鼠标悬停类样式之后，Bootstrap 重新定义了一套情景类鼠标悬停样式，该套类样式坚持在情景类样式基础上适当加深背景色效果。

4.3　表单优化设计

良好的表单设计能够方便用户快速沟通。表单主要包含表单域、输入框、下拉框、单选框、多选框和按钮等控件，每个表单控件在交互中所起到的作用也是各不相同。了解不同表单控件在浏览器中所具备的特殊性，以及 Bootstrap 对其控制的能力，就能更加清楚如何恰当地选用表单对象并进行美化。

4.3.1　Bootstrap 支持的表单控件

Bootstrap 支持所有的标准表单控件，同时对不同表单标签进行优化和扩展。下面进行简单的说明。

1. 输入框（Input）

Bootstrap 支持大部分常用输入型表单控件，包括所有 HTML5 支持的控件，如 text、password、datetime、datetime-local、date、month、time、week、number、email、url、search、tel 和 color。使用这些表单控件时，必须指明 type 属性值。

例如，下面这行代码定义了一个文本输入框。

```
<input type="text"placeholder=“文本框默认值 ">
```

2. 文本区域（Textarea）

对于多行文本框，可使用文本区域，该表单控件支持多行文本。可根据需要设置 rows 属性，来定义多行文本框显示的行数。

例如，下面这行代码定义了一个 3 行文本区域。

```
<textarea rows="3"></textarea>
```

3. 单元按钮和复选框

单选按钮（<input type="radio">）是一个圆形的选择框。当选中单选按钮时，圆形按钮的中心会出现一个圆点。多个单选按钮可以合并为一个单选按钮组，单选按钮组中的 name 值必须相同，如 name="RadioGroup1"，即单选按钮组同一时刻只能选择一个。单选按钮组的作用是“多选一”，一般包括有默认值，否则不符合逻辑。使用 checked 属性可以定义选中的按钮。

复选框（<input type="checkbox">）允许同时选择多个，每个复选框都是一个独立的对象，且必须有一个唯一的名称（name）。它的外观是一个矩形框，当选中某项时，矩形框里会出现小对号。

与单选按钮（radio）一样，使用 checked 属性表示选中状态。与 readonly 属性类似，checked 属性也是一个布尔型属性。

例如，下面分别设计一个复选框和两个单选按钮，并通过 name 属性，把两个单选按钮捆绑在一起，默认状态下，它们会以垂直顺序进行排列，如图 4-26 所示。

```
<label class="checkbox">
  <input type="checkbox"value="">复选框
</label>
<label class="radio">
  <input type="radio"name="optionsRadios"id="optionsRadios1"value="option1"checked>男
</label>
<label class="radio">
  <input type="radio"name="optionsRadios"id="optionsRadios2"value="option2">女
</label>
```

如果为复选框或单选按钮控件引用 inline 类，会使它们排列在同一行。例如，针对上面的示例，在代码中为 <label> 标签引入 inline 类样式，则显示效果如图 4-27 所示。

```
<label class="checkbox inline">
  <input type="checkbox" value="">复选框
</label>
<label class="radio inline">
<input type="radio"name="optionsRadios"id="optionsRadios1"value="option1"checked>男
</label>
<label class="radio inline">
  <input type="radio"name="optionsRadios"id="optionsRadios2"value="option2">女
</label>
```

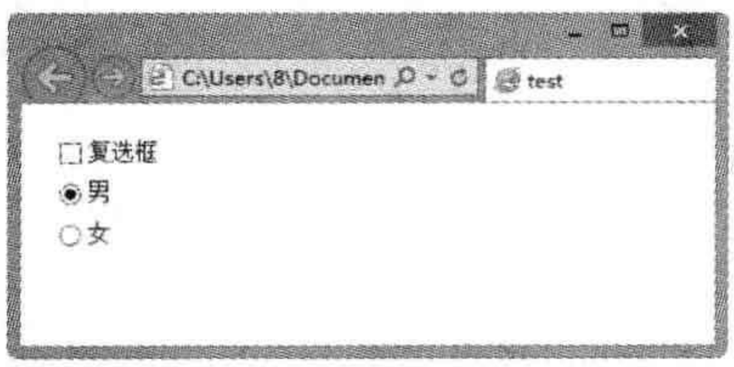


图 4-26　默认单选按钮和复选框样式

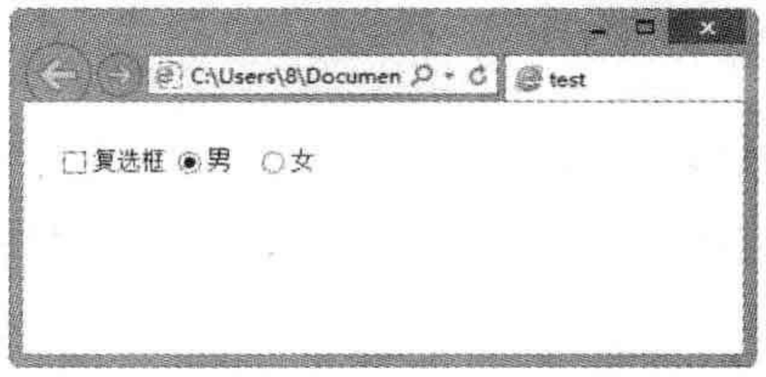


图 4-27　水平显示复选框和单选按钮样式

4. 下拉框

<select> 标签与 <option> 标签可配合使用，用来设计下拉菜单或者列表框，<select> 标签可以包含任意数量的 <option> 标签或 <optgroup> 标签。<optgroup> 标签负责对 <option> 标签进行分组，即多个 <option> 标签放到一个 <optgroup> 标签内。

注意，<optgroup> 标签中的内容不能选择，它的值也不会提交给服务器。<optgroup> 标签用于在一个层叠式选择菜单中为选项分类，label 属性是必不可少的，在可视化浏览器中，它的值将会是一个不可选的伪标题。

<select> 标签同时定义菜单和列表。菜单和列表的区别如下：

- 菜单是节省空间的方式，正常状态下只能看到一个选项，单击下拉按钮打开菜单后才能看到全部选项，即默认设置是菜单形式；
- 列表显示一定数量的选项。如果超出了这个数量，则会出现滚动条，浏览者可以通过拖动滚动条来查看选项并选择。

例如，在下面的示例中，“您来自哪个城市”针对省份的不同进而更快地选择城市，通过 <optgroup> 标签将数据进行分组，可以更快地找到所要选择的选项，使用 selected 属性默认设置选中“青岛”（见图 4-28）。如果没有定义该属性，则“您来自哪个城市”的值将为第一个选项——“潍坊”。

```
<p>
您来自哪个城市:
<select name=" 选择城市 ">
      <optgroup label=" 山东省 ">
          <option value=" 潍坊 "> 潍坊 </option>
          <option value=" 青岛 " selected="selected"> 青岛 </option>
      </optgroup>
```

```
        <optgroup label="山西省">
            <option value="太原">太原</option>
            <option value="榆次">榆次</option>
        </optgroup>
</select>
</p>
```

<select>标签中，通过设置 size 属性定义下拉菜单中显示的项目数目，<optgroup>标签的项目计算在其中。它的作用与输入域中的是不同的，在输入域中的代表的是默认值。在<select>中设置 size="3"，则下拉菜单将不止显示“潍坊”一个值，而是显示“山东省”、“潍坊”及“青岛”三个值。

通过设置 multiple 属性定义下拉菜单可以多选。例如，设置 multiple="multiple"，则按住 Shift 键，在下拉菜单中单击可以同时选择多个项目值，如可以同时选中“潍坊”和“青岛”两个值。

4.3.2 Bootstrap 扩展的表单组件

Bootstrap 支持现有的表单控件，同时也定义了一些有用的表单组件。下面进行简单的说明。

1. 缀饰文本框

通过 add-on 类和 input 进行组合设计，可以在任何文本输入框之前或之后添加文本或按钮。例如，在下面的代码中，分别为文本框绑定 E-mail 前缀和补加两位小数位后缀，演示效果如图 4-29 所示。

```
<div class="input-prepend">
     <span class="add-on">E-mail</span>
     <input class="span2"id="prependedInput"type="text"placeholder="xxx@xx.xx">
</div>
<div class="input-append">
     <input class="span2" id="appendedInput" type="text">
     <span class="add-on">.00</span>
</div>
```

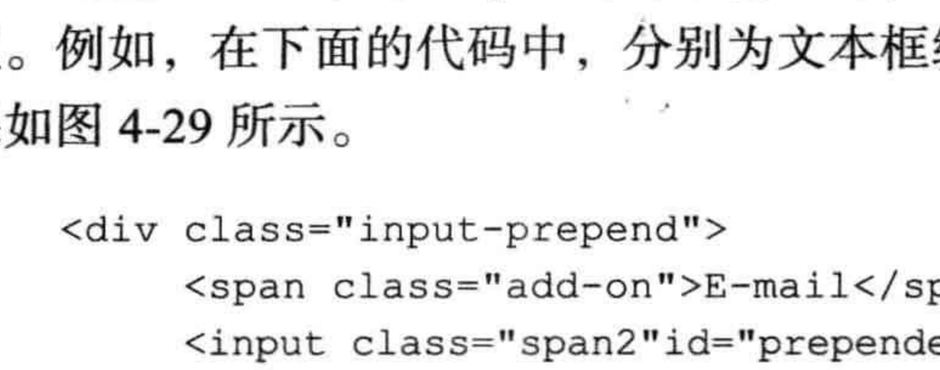


图 4-28 默认下拉列表框样式

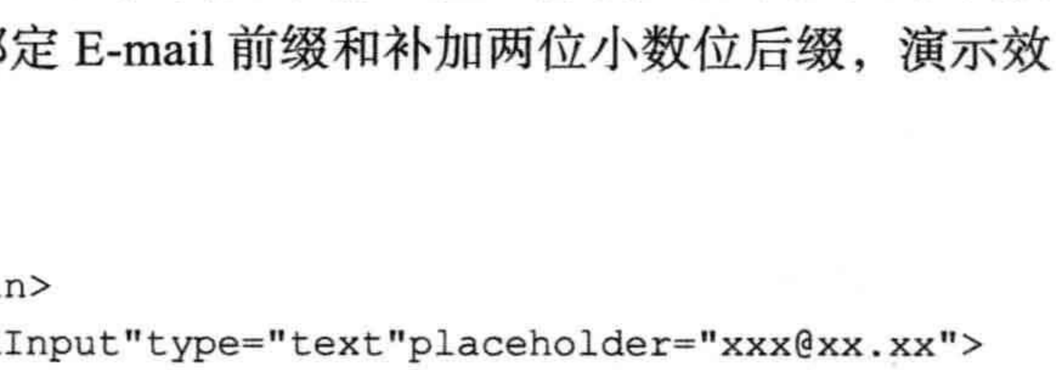


图 4-29 分别绑定前缀和后缀文本框的效果

该组件不支持 select 控件。同时，在使用该组件时，建议使用外包含框 或者 标明组件的形式，即是前缀文本框还是后缀文本框。

也可以同时使用两个类，将两个 .add-on 分别放在输入框的前面和后面。例如，下面文本框绑定了电子邮箱的后缀和提示标签，预览效果如图 4-30 所示。

```
<div class="input-prepend input-append">
     <span class="add-on">E-mail</span>
     <input class="span2" id="prependedInput" type="text" placeholder=" 邮箱地址 ">
     <span class="add-on">@163.com</span>
</div>
```

提示　当同时为一个文本框绑定前缀和后缀文本时，应该在包含框中引入 input-prepend 和 input-append 类样式。

2. 按钮文本框

使用 btn 类可以把表单按钮与输入文本框捆绑在一起，定制按钮文本框组件，按钮可以放在文本框的前面或者后面。

例如，在下面的示例中，分别为文本框绑定 3 个按钮，置于其前后，使用 btn 类样式定制按钮风格，使用 包含框把它们捆绑在一起，预览效果如图 4-31 所示。

```
<div class="input-prepend input-append">
     <input type="button" class="btn" value=" 用户名 " />
     <input type="text" />
     <button class="btn"> 注册 </button>
</div>
```

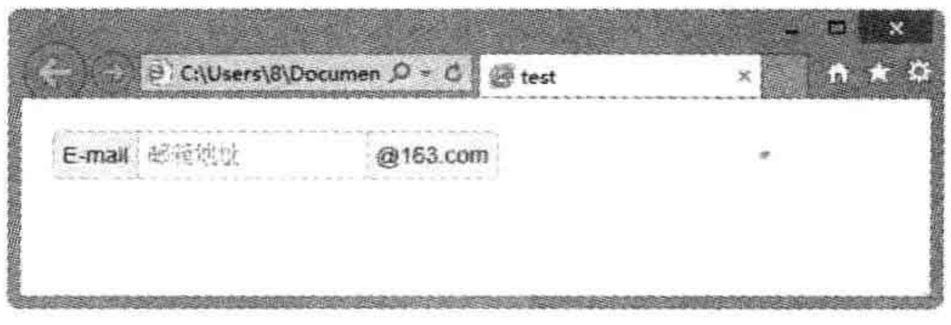


图 4-30　同时绑定前缀和后缀文本框的效果

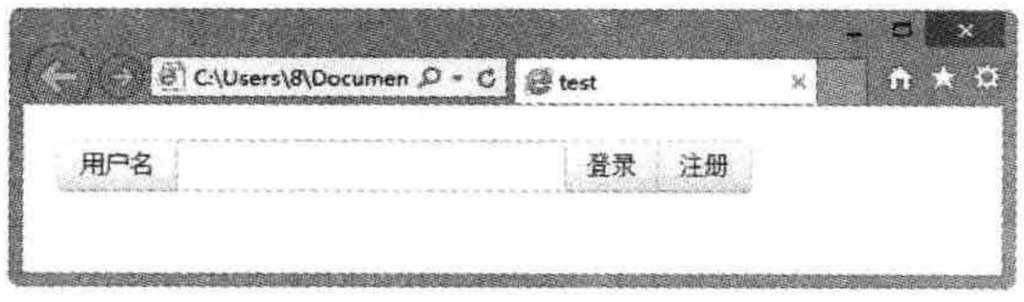


图 4-31　绑定的按钮文本框组件效果

3. 按钮下拉菜单

通过 btn-group 类可以定义按钮组，使用 dropdown-menu 类可以定义下拉菜单样式，如果把它们结合在一起，就可以定制按钮式下拉菜单。

例如，在下面的示例中为文本框绑定前缀按钮和后缀按钮，后缀按钮通过 包含框进行捆绑，在其中添加了一个下拉列表框 <ul class="dropdown-menu">，演示效果如图 4-32 所示。

```
<div class="input-prepend input-append">
     <input type="button" class="btn" value="Email" />
     <input type="text" />
     <div class="btn-group">
```

```
        <button class="btn" data-toggle="dropdown">@163.com</button>
        <ul class="dropdown-menu">
            <li><a href="#">@126.com</a></li>
            <li><a href="#">@sohu.com</a></li>
            <li><a href="#">@qq.com</a></li>
            <li><a href="#">@263.net</a></li>
        </ul>
    </div>
    <button class="btn">登录</button>
</div>
```

在设计按钮下拉菜单样式时，应该在页面中导入 bootstrap.js 文件，以实现下拉菜单的显隐效果。通过为 <button> 按钮设置 data-toggle 属性值为 dropdown，以激活按钮的下拉响应事件。

4. 分段按钮下拉菜单

通过添加辅助标签，可以设计分段式按钮下拉菜单样式。例如，在上面的代码中，通过添加一个空按钮，然后在其中插入一个按钮图标（），修改代码如下，此时预览效果如图 4-33 所示。

```
<div class="input-prepend input-append">
    <input type="button" class="btn" value="Email" />
    <input type="text" />
    <div class="btn-group">
        <button class="btn">@163.com</button>
        <button class="btn" data-toggle="dropdown" tabindex="-1">
            <span class="caret"></span>
        </button>
        <ul class="dropdown-menu">
            <li><a href="#">@126.com</a></li>
            <li><a href="#">@sohu.com</a></li>
            <li><a href="#">@qq.com</a></li>
            <li><a href="#">@263.net</a></li>
        </ul>
    </div>
    <button class="btn">登录</button>
</div>
```

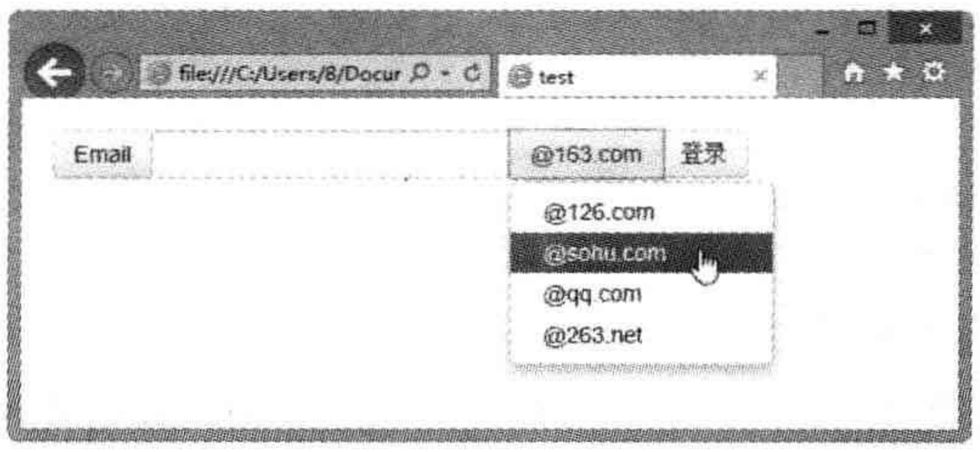


图 4-32　按钮下拉菜单样式

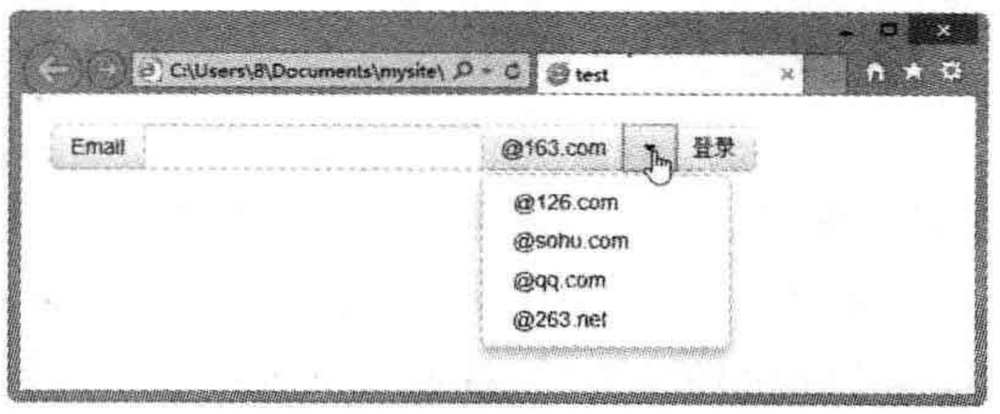


图 4-33　设计分段式下拉菜单

5. 搜索框

为 <form> 标签引入 form-search 类，可以设计搜索框样式。例如，在下面的 <form

class="form-search"> 表单框中，插入一个文本框，然后使用 包含框把它与一个按钮绑定在一起，则显示效果如图 4-34 所示。

```
<form class="form-search">
    <div class="input-append">
        <input type="text" class="span3">
        <button type="submit" class="btn">快速搜索</button>
    </div>
</form>
```

如果为文本框引入 search-query 类，则可以让文本框与按钮保持相同的外观风格（<input type="text" class="span3 search-query">），如图 4-35 所示。

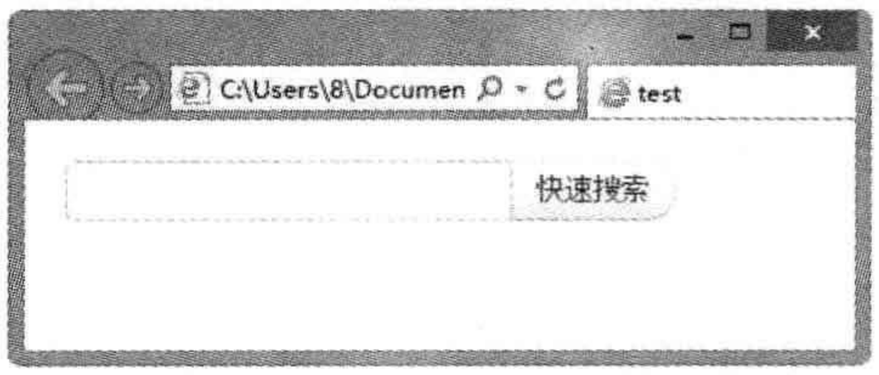


图 4-34　搜索框样式

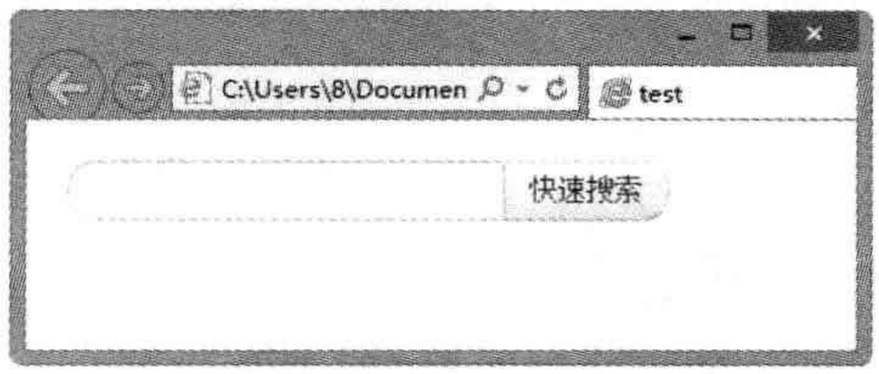


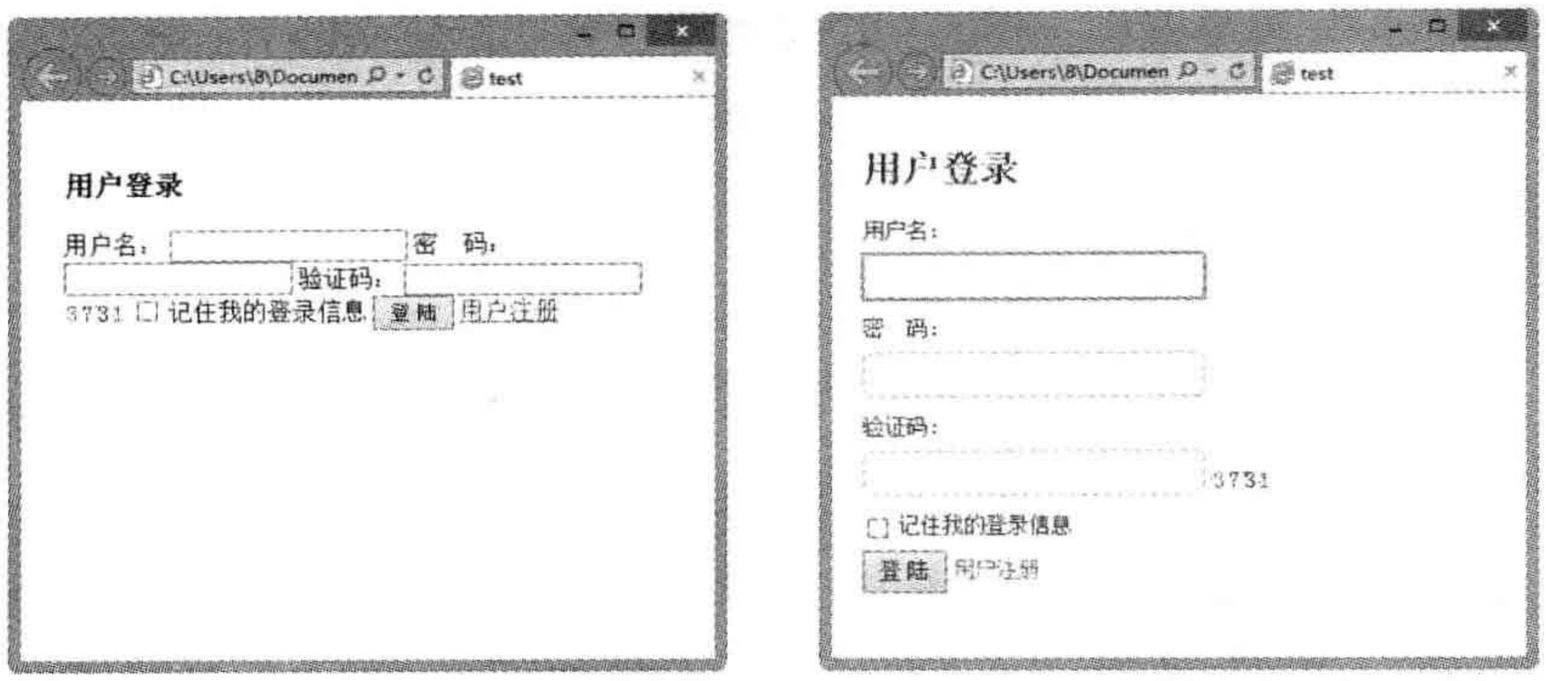
图 4-35　优化搜索框样式

4.3.3　默认风格

Bootstrap 根据表单设计流行趋势，优化了表单对象的默认风格。不用引入任何样式类，也不用修改表单的结构。例如，设计一个简单的登录表单结构，代码如下，则 Bootstrap 风格的表单样式如图 4-36 右图所示。

```
<h3>用户登录</h3>
<form method="post" action="">
    <label for="userName">用户名：</label>
    <input type="text" id="userName" />
    <label for="userPsw">密　码：</label>
    <input type="password" id="userPsw" />
    <label for="validate">验证码：</label>
    <input type="text" id="validate" />
    <img src="images/getcode.jpg" alt="验证码：3731" />
    <label for="keepLogin">
    <input type="checkbox" id="keepLogin" />
    记住我的登录信息</label>
    <button type="submit" class="btn_login">登 陆</button>
    <a href="#" class="reg">用户注册</a>
</form>
```

Bootstrap 默认所有表单控件垂直分布，设计标签 <label> 左侧对齐并在控件之上，添加手形光标样式；对文本框的样式进行提升：圆角、灰边、激活时蓝色晕边。通过 padding 和 margin 属性改善表单控件的内补白和边距，让整个表单结构看起来更大气。



浏览器默认表单样式　　Bootstrap 风格的表单样式

图 4-36　经过优化的表单样式前后对比效果

4.3.4　布局风格

Bootstrap 从三个方面完善了表单的布局特性，具体说明如下。

1. 圆角搜索框

圆角搜索框风格主要通过两个类样式实现：

- form-search：为 <form> 标签设置表单类型，即搜索表单框。
- search-query：为文本框 <input> 标签设置圆角样式。

简单代码如下，预览效果如图 4-37 所示。

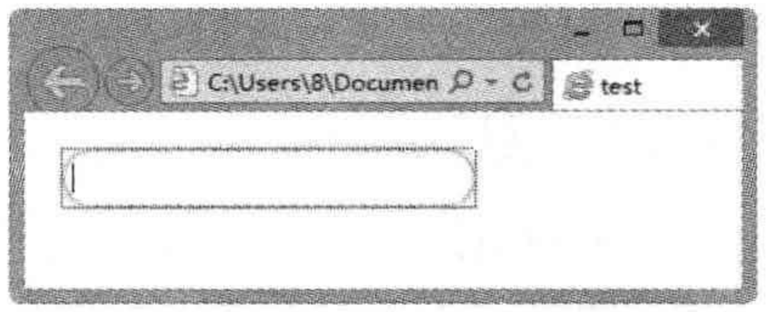


图 4-37　搜索文本框默认样式

```
<form class="form-search">
    <input type="text" class=" search-query">
</form>
```

2. 行内布局

通过为 <form> 标签引入 form-inline 类，可以设计整个表格结构以行内显示。例如，在前面介绍的表格结构中，清理掉表单中所有 标签，为 <form> 标签设置 class="form-inline"，则所有表单控件都将在行内显示，如图 4-38 所示，得到一个压缩型排列的表单，其中 <label> 文本左侧对齐、控件 display 为 inline-block 类型。

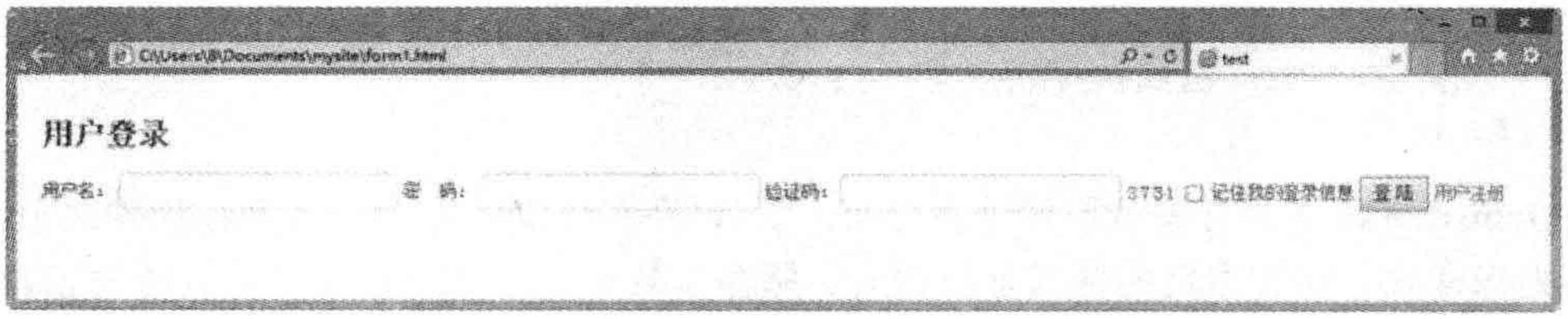


图 4-38　行内显示的表单控件

3. 水平布局

要设计水平布局效果，需要对表单结构进行重新设计。重构表单模板代码如下：

```
<form class=" form-horizontal">
    <div class="control-group">
        <div class="control-label">
            <!-- 标签 -->
        </div>
        <div class="controls">
            <!-- 表单控件 -->
        </div>
    </div>
</form>
```

上面的模板结构说明如下：

- 为表单框（<form> 标签）添加 form-horizontal 类；
- 将标签文本（<label>）和控件包裹在 control-group 类容器中；
- 将标签文本（<label>）包裹在 control-label 类子容器中；
- 将控件包裹在 controls 类子容器中。

例如，针对上面的表单案例，重构后的表单结构代码如下，在浏览器中预览表单水平布局效果，如图 4-39 所示。

```
<h3>用户登录</h3>
<form method="post" action="" class=" form-horizontal">
    <div class="control-group">
        <div class="control-label">
            <label for="userName">用户名：</label>
        </div>
        <div class="controls">
            <input type="text" id="userName" />
        </div>
    </div>
    <div class="control-group">
        <div class="control-label">
            <label for="userPsw">密　码：</label>
        </div>
        <div class="controls">
            <input type="password" id="userPsw" />
        </div>
    </div>
    ...
</form>
```

4.3.5　外观风格

Bootstrap 通过各种样式类，为用户提供了更多定制表单样式的途径和方法。下面进行简

单说明。

1. 定制大小

Bootstrap 提供了两种定制表单控件大小的途径：相对大小和网格大小。

1）相对大小是一组与关键字相关联的类，如下所示。

```
.input-mini { width: 60px; }
.input-small { width: 90px; }
.input-medium { width: 150px; }
.input-large { width: 210px; }
.input-xlarge { width: 270px; }
.input-xxlarge { width: 530px; }
```

分别在文本框中引用这些样式类，则可以直观地比较它们的大小，如图 4-40 所示。

```
<label><input class="input-mini" type="text" placeholder=".input-mini"></label>
<label><input class="input-small" type="text" placeholder=".input-small"></label>
<label><input class="input-medium" type="text" placeholder=".input-medium"></label>
<label><input class="input-large" type="text" placeholder=".input-large"></label>
<label><input class="input-xlarge" type="text" placeholder=".input-xlarge"></label>
<label><input class="input-xxlarge" type="text" placeholder=".input-xxlarge"></label>
```


图 4-39　水平布局的表单控件

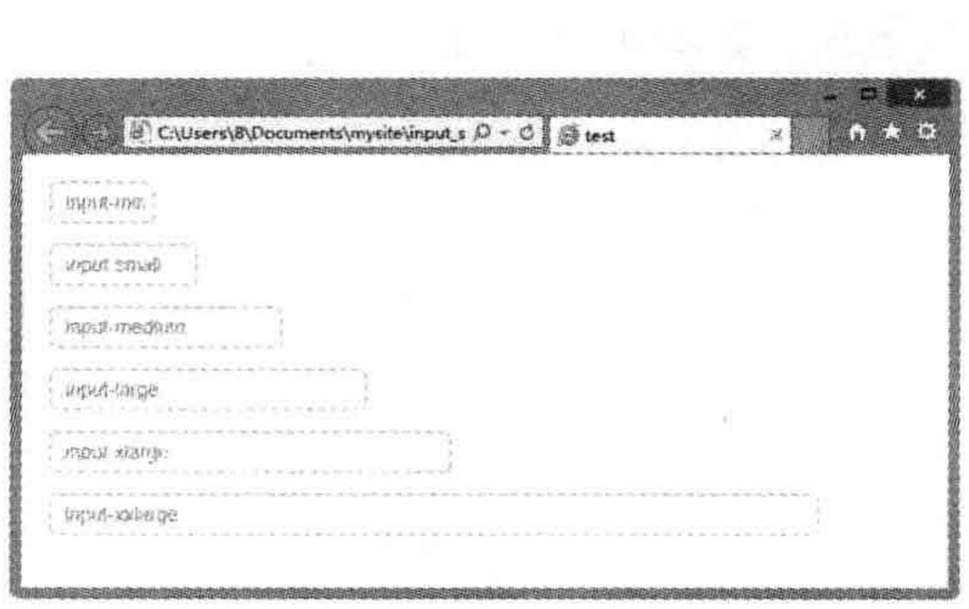


图 4-40　设计表单控件相对大小

2）栅格大小是一组与栅格布局相关的控制大小的类，代码如下：

```
input.span3, textarea.span3, .uneditable-input.span3 { width: 206px; }
input.span2, textarea.span2, .uneditable-input.span2 { width: 126px; }
input.span1, textarea.span1, .uneditable-input.span1 { width: 46px; }
```

例如，针对上面的文本框，修改为引用不同级别的栅格单元宽度类，则演示效果如图 4-41 所示。

如果设计表单控件 100% 宽度显示，则可以使用 input-block-level 类把表单对象转换为块状显示，该类主要作用于 <input> 和 <textarea> 表单控件。

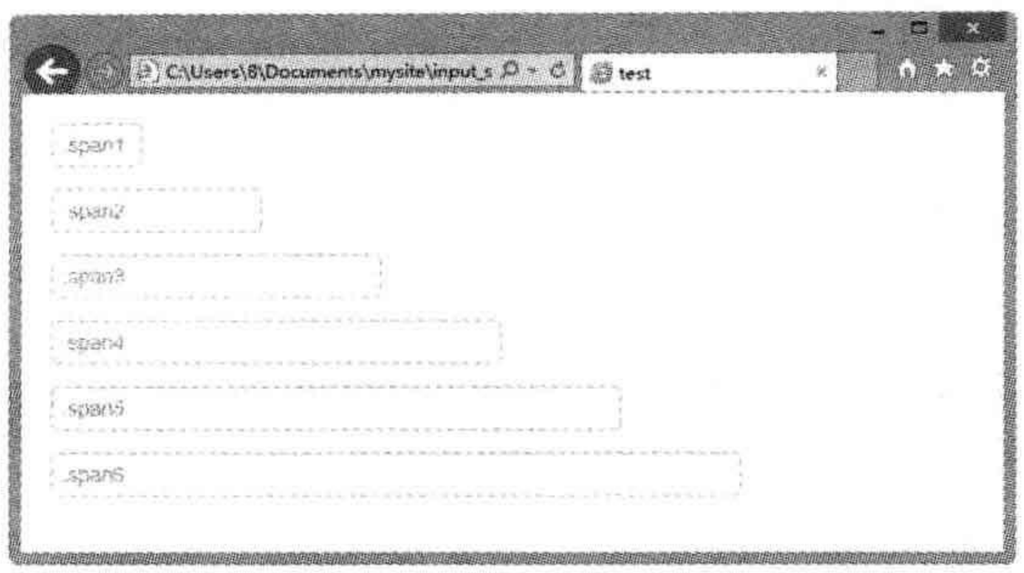


图 4-41　设计表单控件栅格大小

例如，下面的代码定义文本框为 100% 宽度的块状显示，如图 4-42 所示。

```
<input class="input-block-level" type="text" placeholder=".input-block-level">
```

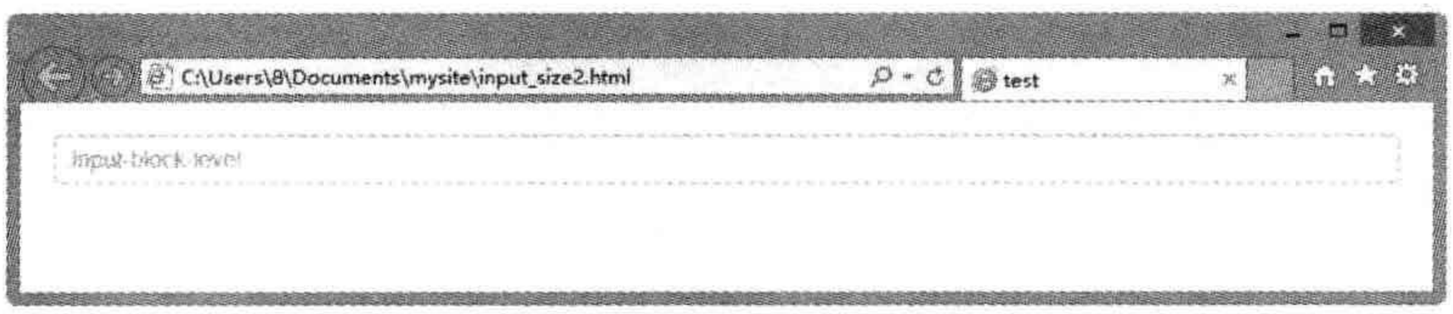


图 4-42　设计表单控件为块状显示

如果想让每行显示多个文本框，则可以使用 controls-row 类为输入框增加合适的间距。通过浮动让输入框之间缩减一些空白，设置适当的边距，并清除浮动。例如，设计两个并列显示的文本框，可以将它们包裹在 controls-row 容器中，代码如下，预览效果如图 4-43 所示。

```
<div class="controls-row">
     <input class="span1" type="text" placeholder=".span1">
     <input class="span6" type="text" placeholder=".span6">
</div>
```

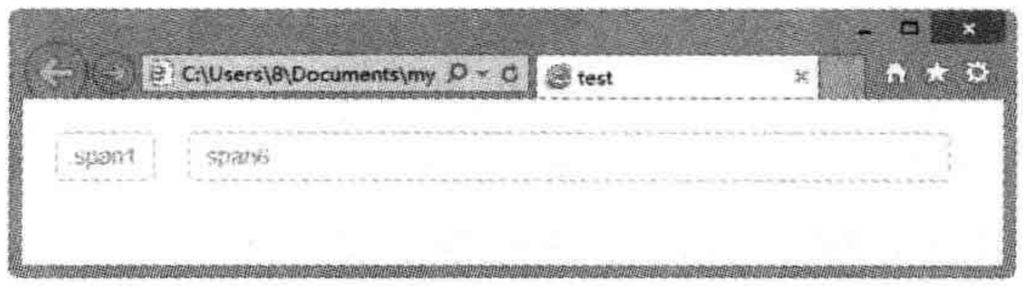


图 4-43　设计并列显示的表单控件

2. 定制不可编辑的样式控件

通过设置 disabled 属性，可以设计不可编辑的表单控件。Bootstrap 允许对于在表单中呈现不可编辑的数据使用任何标签来表示，不需要使用实际的表单控件，只需引入 uneditable-input 类即可。

例如，分别为 <input> 和 标签引入 span4 和 uneditable-input 类，则预览效果如图 4-44 所示。

```
<input class="span4 uneditable-input" disabled type="text" placeholder=".span4">
<span class="span4 uneditable-input">默认值 </span>
```

图 4-44　设计不可编辑的表单控件

注意，uneditable-input 类不会真正让文本框不可编辑，仅是模拟表单控件不可编辑样式。

3. 定制帮助文本

帮助文本可以与表单控件同行显示，也可以换行显示。

为提示文本框引入 help-inline 类样式，则提示文本与控件会同行显示，效果如图 4-45 所示。

```
<input type="text"><span class="help-inline">行内解释文本 </span>
```

为提示文本框引入 help-block 类样式，则提示文本与控件分行显示，效果如图 4-46 所示。

```
<input type="text"><span class="help-block">块解释文本 </span>
```

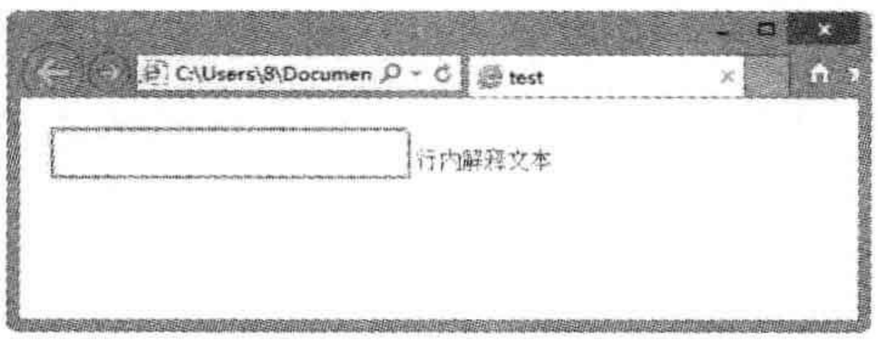


图 4-45　设计行内显示的帮助文本

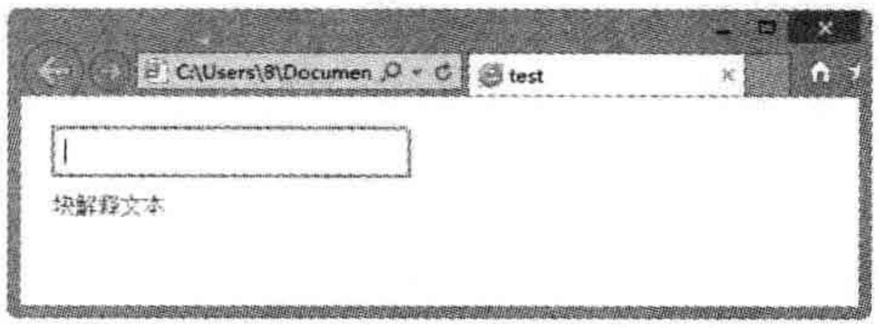


图 4-46　设计块状显示的帮助文本

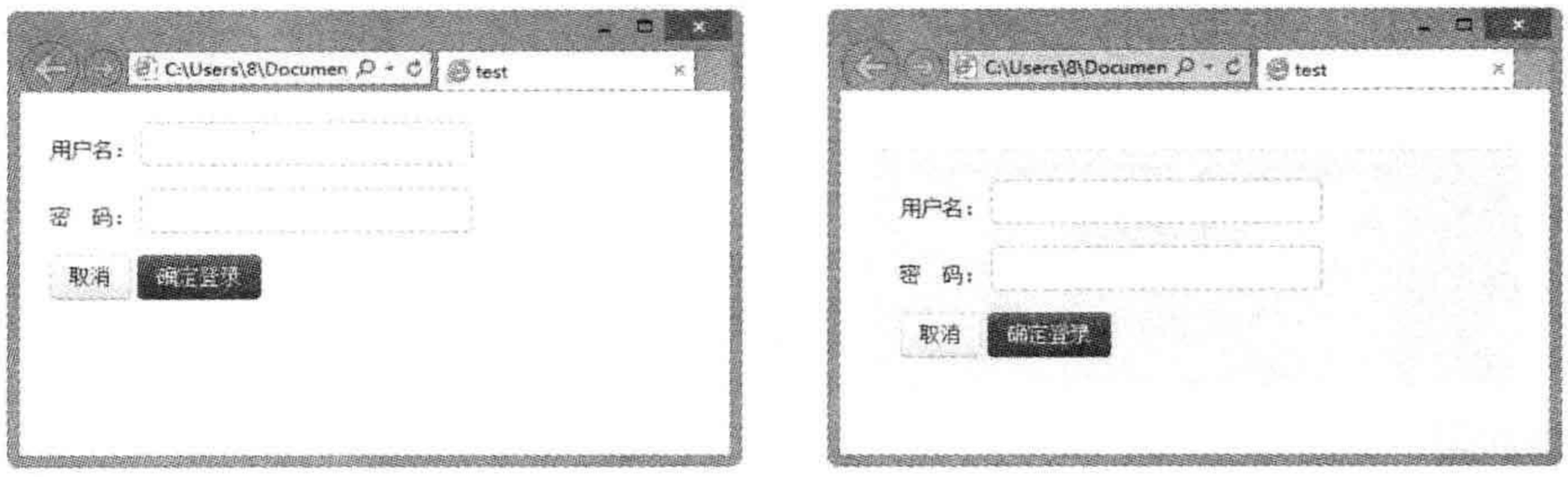
4. 定制表单行为

Bootstrap 定义了 form-actions 类，该类专门负责控制表单整体样式，以便更好地显示表单效果。它主要针对在复杂的混合版式中，可能出现的浮动和流动版块相互环绕的问题进行设计，以清除浮动，并为表单定义独特的版式：灰背景、灰边框、缩进显示，以及增大上下边距。

例如，下面的示例为 <form> 标签引入了 form-actions 类，可以明显看到表单样式的变化，如图 4-47 所示。

```
<form class="form-actions">
    <label>用户名:
        <input type="text" id="userName" />
    </label>
    <label>密　码:
```

```
        <input type="password" id="userPsw" />
    </label>
    <button type="button" class="btn">取消</button>
    <button type="submit" class="btn btn-primary">确定登录</button>
</form>
```



引用前　　引用后

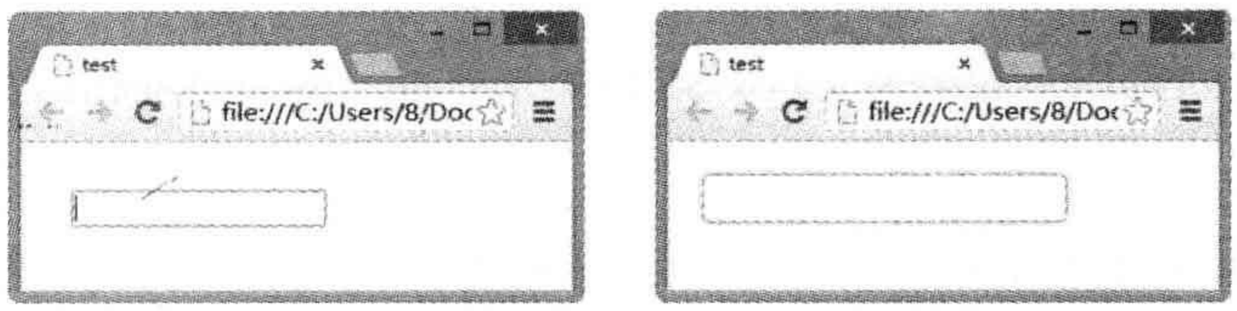
图 4-47　设计表单行为样式

4.3.6　状态风格

表单控件存在多种状态：读写、焦点、禁止、有效、验证等。针对不同的状态提供不同的样式类，以方便用户更加轻松地辨析，这也是 Bootstrap 改进表单样式的一项重要工作。

1. 焦点状态

当表单控件获取焦点后，默认会呈现 outline 样式，而 Bootstrap 优化了这种样式，默认清除 outline 样式，增加了 box-shadow 样式，如图 4-48 所示。



Chrome 浏览器焦点风格　　Bootstrap 焦点风格

图 4-48　获取焦点时样式的比较

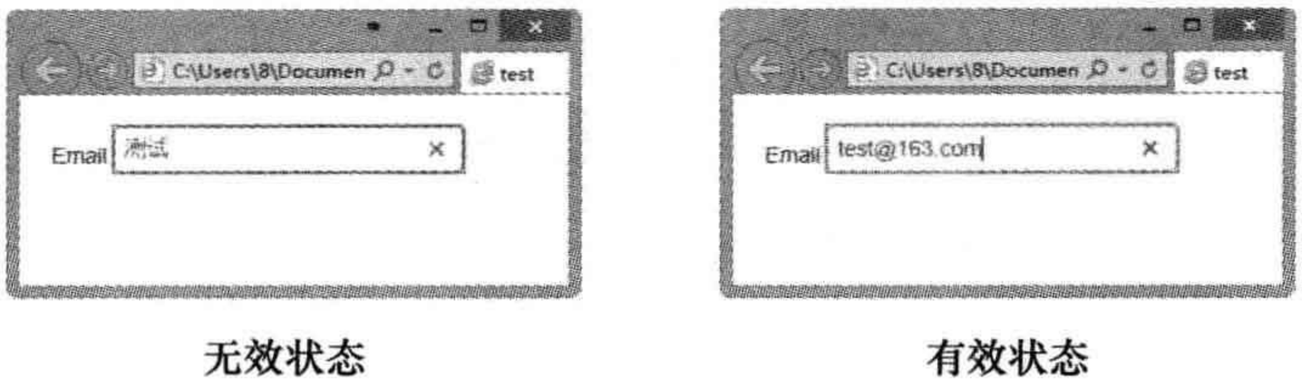
注意，不同浏览器对于表单控件的焦点样式有各自不同的解析，呈现的效果也会有所不同。

2. 无效状态

Bootstrap 重设了输入文本框的：invalid 样式，为它指定 type 和 required 属性就可以激活无效状态，用户也可以配合 pattern 属性，设置输入模式。

例如，针对下面的文本框，添加 required 属性激活无效状态。这样当获取焦点后，文本框会呈现红色边框和字体颜色，而当输入有效文本值后，会显示有效状态样式，如图 4-49 所示。

```
<label>Email
<input class="span3" type="email" required>
</label>
```



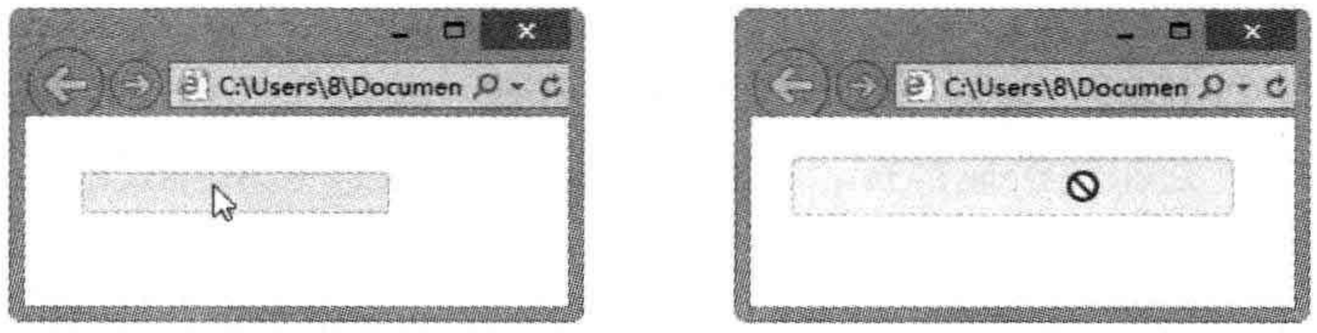
无效状态　　有效状态

图 4-49　激活无效状态

3. 禁用状态

Bootstrap 改善了输入框的禁用状态，设置光标样式为禁用状态，能够更有效地提醒用户，如图 4-50 所示。

```
<input class="span3" type="text" disabled>
```



默认样式　　Bootstrap 风格

图 4-50　禁用状态

注意，由于 IE9 及其以下版本不支持 CSS 伪类选择器，因而不能在这些浏览器中使用。

4. 验证状态

Bootstrap 增设了验证状态类样式，主要包括 error（错误）、warning（警告）、info（通知）和 success（成功）信息的样式。

在使用时，用户只需要为控件组包含框添加验证状态类样式即可，应用模式如下：

```
<div class="control-group warning">
     <label class="control-label" for="user">用户名 </label>
     <div class="controls">
          <input type="text" id="user">
          <span class="help-inline">非法的用户名 </span>
     </div>
</div>
```

上面的模式中，验证状态样式会影响到 label 文本，使用下面的简洁模式，不会对 label 文本产生影响。

```
<div class="control-group warning">
    <label>用户名
        <input type="text" id="user">
        <span class="help-inline">非法的用户名</span>
    </label>
</div>
```

例如，分别设置 4 个表单组结构，并引入 4 不同的验证状态样式类，比较效果如图 4-51 所示。

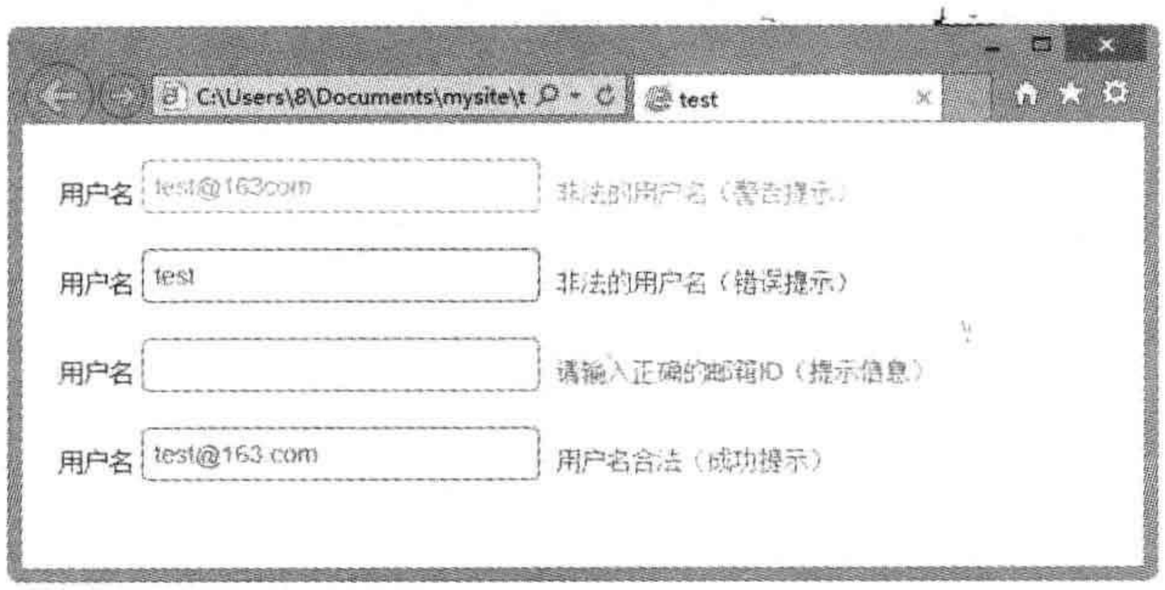


图 4-51　验证状态效果比较

4.4　按钮设计

在页面中添加有立体感、水晶感、动态化的按钮效果会让网页看起来更富有吸引力。传统做法是采用背景图像来模拟，随着 CSS3 表现力增强，纯 CSS 立体按钮样式也逐渐多起来。借助边框和背景样式的变化（主要是颜色的深浅变化）来模拟凹凸变化的效果，然后借助 CSS3 中的圆角、渐变、透明等表现属性，就可以打造出更具专业水准的按钮效果。

4.4.1　默认风格

Bootstrap 专门定制了 btn 样式类，应用该类可以设计出专业水平的按钮效果。该类样式设计复杂，综合应用了 CSS3 中大部分特效，如文本阴影（text-shadow）、渐变背景色（background-image）、边框半透明（border:）、圆角（border-radius）、元素阴影（box-shadow）等。

难能可贵的是，Bootstrap 从专业视角搭配各种设置参数，使按钮视觉效果看起来更佳，同时考虑到不同浏览器的解析差异，进行了比较安全的兼容性处理，使按钮效果在不同浏览器中所呈现的效果基本相同。该样式的完整代码如下：

```
.btn {
    /* 行内块显示，IE 不支持，则定义为行内显示 */
    display: inline-block;
    *display: inline;
    /* 通过补白，让按钮看起来更饱满 */
    padding: 4px 12px;
    /* 清除边界的干扰，特别是 IE 的怪异模式 */
```

```
margin-bottom: 0;
*margin-left: .3em;
/* 固定字体大小和行高，稳妥设计 */
font-size: 14px;
line-height: 20px;
/* 深灰色更耐看 */
color: #333333;
text-align: center;
/* 给文本添加半透明的灰色阴影，营造淡淡的雕琢效果 */
text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);
vertical-align: middle;
cursor: pointer;
/* 背景色是经典的灰白色 */
background-color: #f5f5f5;
*background-color: #e6e6e6;
/* 线性渐变，从白色到经典灰色，让背景看起来更温润 */
background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));
background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);
background-image: linear-gradient(to bottom, #ffffff, #e6e6e6);
background-repeat: repeat-x;
/* 边框是浅灰色，稍稍暗于背景色，边框效果更醒目 */
border: 1px solid #cccccc;
*border: 0;
border-color: #e6e6e6 #e6e6e6 #bfbfbf;
/* 调整各边边框色彩透明度，营造立体效果 */
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
border-bottom-color: #b3b3b3;
/* 圆角边框，让按钮更耐看 */
-webkit-border-radius: 4px;
   -moz-border-radius: 4px;
        border-radius: 4px;
/*IE 背景渐变 */
 filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffffffff',
  endColorstr='#ffe6e6e6', GradientType=0);
  filter: progid:DXImageTransform.Microsoft.gradient(enabled=false);
/* 让 IE 具有块状显示特性 */
*zoom: 1;
   /* 设计按钮有淡淡的内阴影效果 */
   -webkit-box-shadow:inset 0 1px 0 rgba(255,255,255,0.2),0 1px 2px rgba(0,0, 0,0.05);
      -moz-box-shadow:inset 0 1px 0 rgba(255,255,255,0.2),0 1px 2px rgba(0,0,0, 0.05);
          box-shadow:inset 0 1px 0 rgba(255,255,255,0.2),0 1px 2px rgba(0,0,0, 0.05);
}
```

任何引用 btn 样式类的页面元素都会显示按钮样式。不过，btn 样式类通常应用于 <a> 和 <button> 标签，因为一方面这两种标签拥有更好的表现力，另一方面为 <a> 和 <button> 标签设计按钮效果，也符合 HTML 结构的语义化要求。

例如，下面分别为页面中的 <a>、<button>、<input type="button">、<input type=" submit">

标签引入 btn 样式类，则页面显示效果是一样的，如图 4-52 所示。

```
<a class="btn" href="">超级链接（a）</a>
<button class="btn">按钮标签（button）</button>
<input class="btn" type="button" value="按钮标签（input）">
<input class="btn" type="submit" value="提交按钮（input）">
```

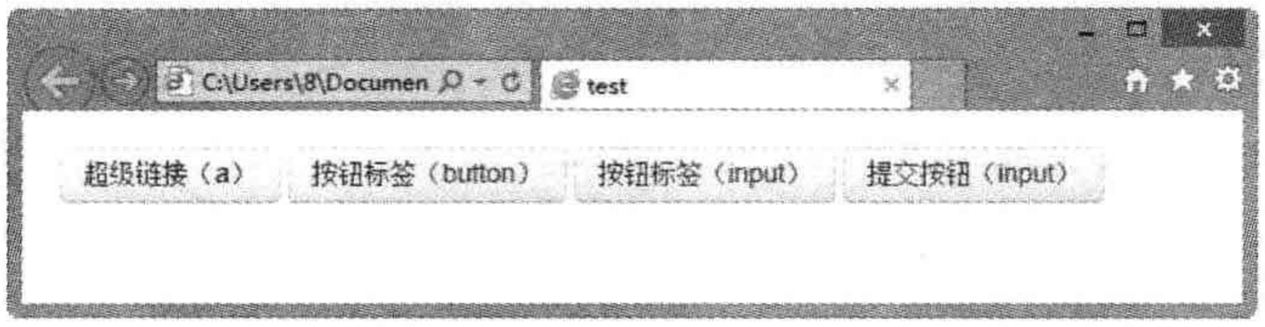


图 4-52　设计按钮效果

提示　最佳实践是，根据使用环境尝试选用合适的标签，以确保渲染的效果在各个浏览器中基本一致。如果正使用 input，那么设计按钮效果就应该使用 <input type="submit"> 而不是 <a>，这样它们的表现是一致的。

4.4.2　定制风格

Bootstrap 提供了多种按钮风格，以供用户自由选用。它为 btn 附加了一组情景样式类，方便在不同环境中改变按钮的色彩，简单说明如下。

- btn-primary：主要，通过醒目的视觉变化（亮蓝色），提示浏览者当前按钮在一系列的按钮中为主要操作。
- btn-info：信息，通过舒适的色彩设计（浅蓝色），调节按钮默认的灰色视觉效果，可以用来替换默认按钮样式。
- btn-success：成功，通过亮绿色，表示成功或积极的动作。
- btn-warning：警告，通过通用黄色，提醒应该谨慎采取这个动作。
- btn-danger：危险，通过通用红色，提醒当前操作可能会存在危险。
- btn-inverse：反向，通过黑色背景，表示特殊用途，一般不依赖于语义和用途。
- btn-link：链接，把按钮转换为链接样式，简化按钮，使它看起来像一个链接。

例如，在下面的代码中分别引用这些附加按钮样式，则显示效果如图 4-53 所示。

```
<button class="btn">默认</button>
<button class="btn btn-info">信息</button>
<button class="btn btn-primary">主要</button>
<button class="btn btn-success">成功</button>
<button class="btn btn-warning">警告</button>
<button class="btn btn-danger">危险</button>
<button class="btn btn-inverse">反向</button>
<button class="btn btn-link">链接</button>
```

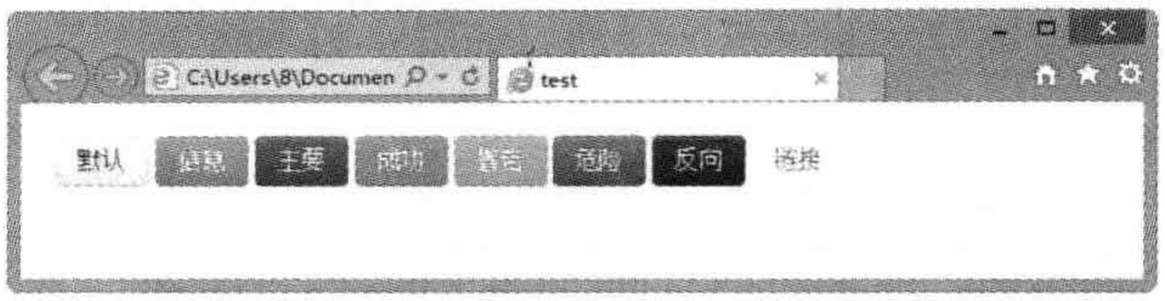


图 4-53　设计多种风格的按钮

注意，这些附加样式类必须与 btn 捆绑使用，否则按钮效果就会失真。

Bootstrap 还提供了三个定制相对大小的按钮样式类，使用它们可以酌情调整按钮的大小，简单说明如下。

- btn-large：大号按钮。
- btn-small：小号按钮。
- btn-mini：迷你号按钮。

如果把这三个样式类与按钮默认样式进行比较，则效果如图 4-54 所示。

```
<button class="btn btn-info btn-large">大号按钮</button>
<button class="btn btn-info">默认大小</button>
<button class="btn btn-info btn-small">小号按钮</button>
<button class="btn btn-info btn-mini">迷你按钮</button>
```

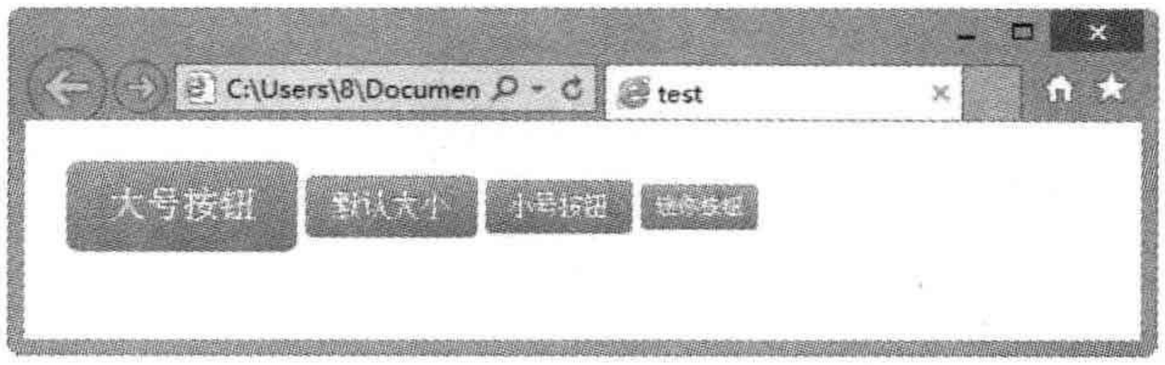



图 4-54　按钮大小比较

Bootstrap 定义 btn-block 样式类，用来把按钮转换为块级元素，此时按钮会填充整个包含框。例如，下面代码会把按钮放大，填充一行，这样更适合用户快速操作，如图 4-55 所示。

```
<button class="btn btn-info btn-large btn-block">登录微博</button>
```

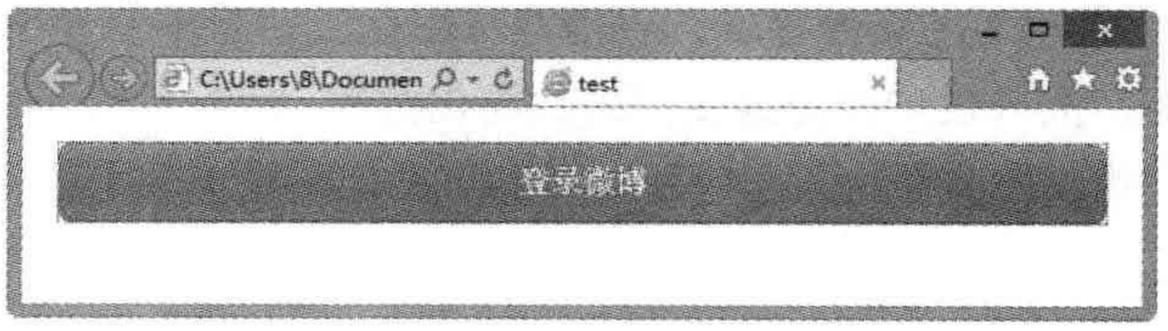


图 4-55　设计按钮块显示

4.4.3　状态风格

按钮只有一个禁用状态，当按钮被禁用时，颜色将会变浅，降低 50%，同时按钮的交互

样式被禁用，当光标移到按钮上时，按钮样式不再发生变化。这种不可用状态通过 disabled 样式类实现。

例如，下面分别对默认按钮和其他风格按钮应用 disabled 样式类，则显示效果如图 4-56 所示。

```
<a href="#"class="btn btn-large btn-primary disabled">大号链接 </a>
<a href="#"class="btn disabled">默认链接 </a>
```

注意，disabled 样式类只能够禁用 CSS 交互样式，但是无法禁用按钮的默认行为，如果要禁用默认行为，还需要用 JavaScript 脚本来控制。

HTML 表单控件包含一个 disabled 属性，使用该属性可以禁用按钮行为。Bootstrap 因此为包含该属性的控件统一了不可用样式，使其效果与 disabled 样式类保持一致。例如，下面两种用法都可以实现相同的不可用状态，如图 4-57 所示。

```
<button type="button" class="btn btn-large" disabled="disabled">属性禁用 </button>
<button type="button" class="btn btn-large disabled">类样式禁用 </button>
```

图 4-56　按钮禁用状态

图 4-57　按钮禁用样式比较

但是，disabled 样式类无法禁用按钮的默认交互行为，所以建议用户在使用时，同时引用这两种用法，代码如下：

```
<button type="button" class="btn btn-large disabled" disabled="disabled">按钮禁
用 </button>
```

4.5　图片和图标设计

Bootstrap 对图片的风格改进很少，其实也无法改进，主要是提供了圆角和描边样式。但是它对于图标的增强是一大亮点，利用 Bootstrap 庞大的图标类集，可以极大地改善网页设计中的装饰和提醒功能。

4.5.1　图片风格

当图片定义超链接时，会呈现默认的边框样式，Bootstrap 关闭了这种边框样式。同时，还为图像定义了三个特殊风格的样式类，分别用来设计圆角图片、圆形图片和镶边图片三种特效风格，简单说明如下。

- img-rounded：设计圆角图片。
- img-circle：设计圆形图片。

❑ img-polaroid：设计镶边图片。

例如，在下面的示例中，为同一幅图片分别应用上述三种类样式，代码如下，演示效果如图 4-58 所示。

```
<div class="row-fluid">
    <div class="text-center span3"> <img src="images/bg1.jpg">
        <h3> 正常效果 </h3>
    </div>
    <div class="text-center span3"> <img src="images/bg1.jpg" class="img-rounded"
     title=" 圆角图片 ">
        <h3> 圆角效果 </h3>
    </div>
    <div class="text-center span3"> <img src="images/bg1.jpg" class="img-circle"
     title=" 圆形图片 ">
        <h3> 圆形效果 </h3>
    </div>
    <div class="text-center span3"> <img src="images/bg1.jpg" class="img-polaroid"
     title=" 镶边图片 ">
        <h3> 镶边效果 </h3>
    </div>
</div>
```

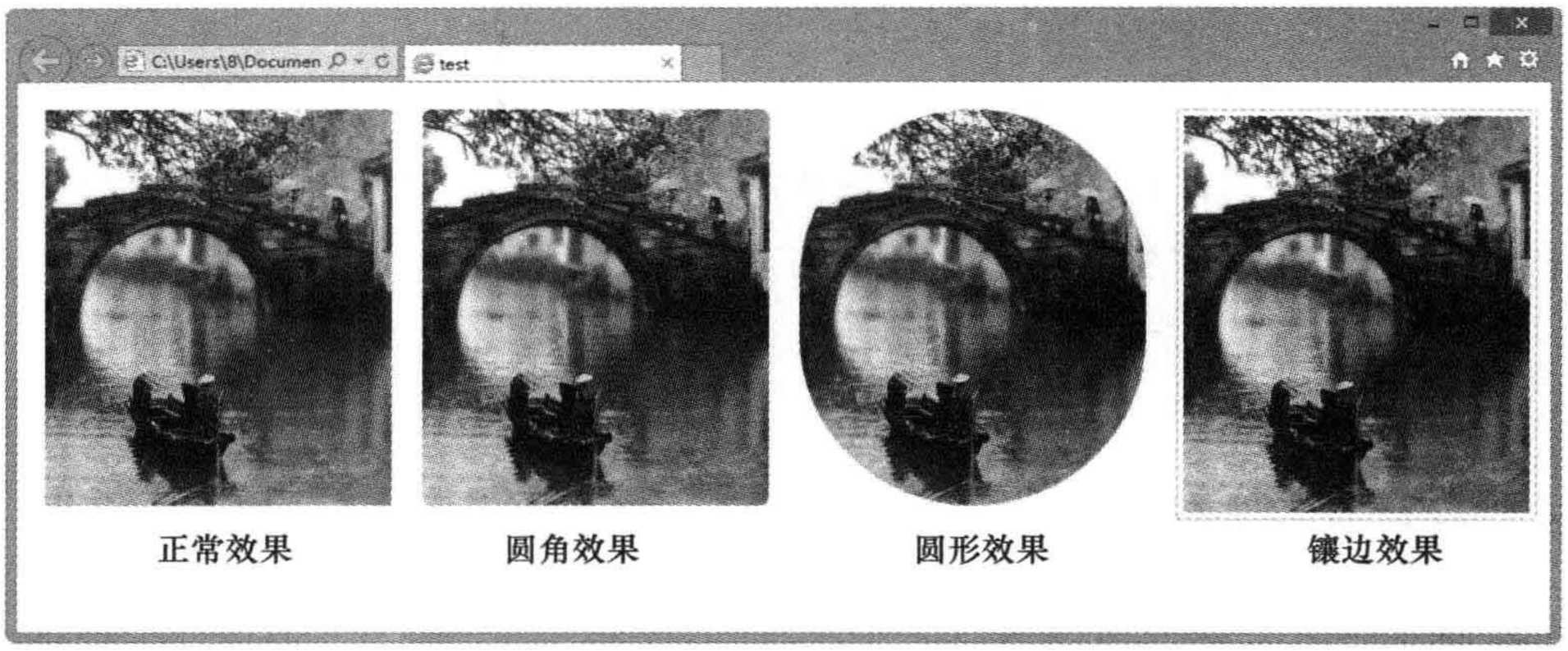



图 4-58　设计图片的四种风格效果

注意，Bootstrap 统一了页面中所有图片的流式显示效果，当页面或者栏目宽度发生变化时，图片的大小也会随之改变。例如，针对上面示例的效果，逐步改变窗口的大小，会发现图片的大小也随之调整，如图 4-59 所示。

4.5.2　图标风格

Bootstrap 设计了一套图标集，以便在网页中嵌入使用各种图标，它们被排列在一幅图片中，如图 4-60 所示。

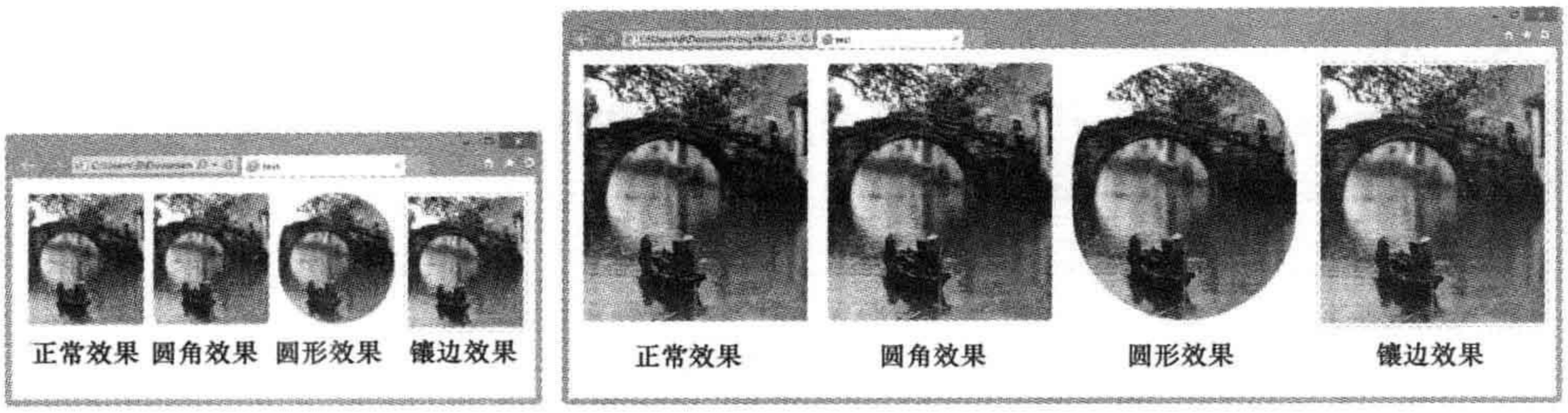


图 4-59　图片的流式变化效果

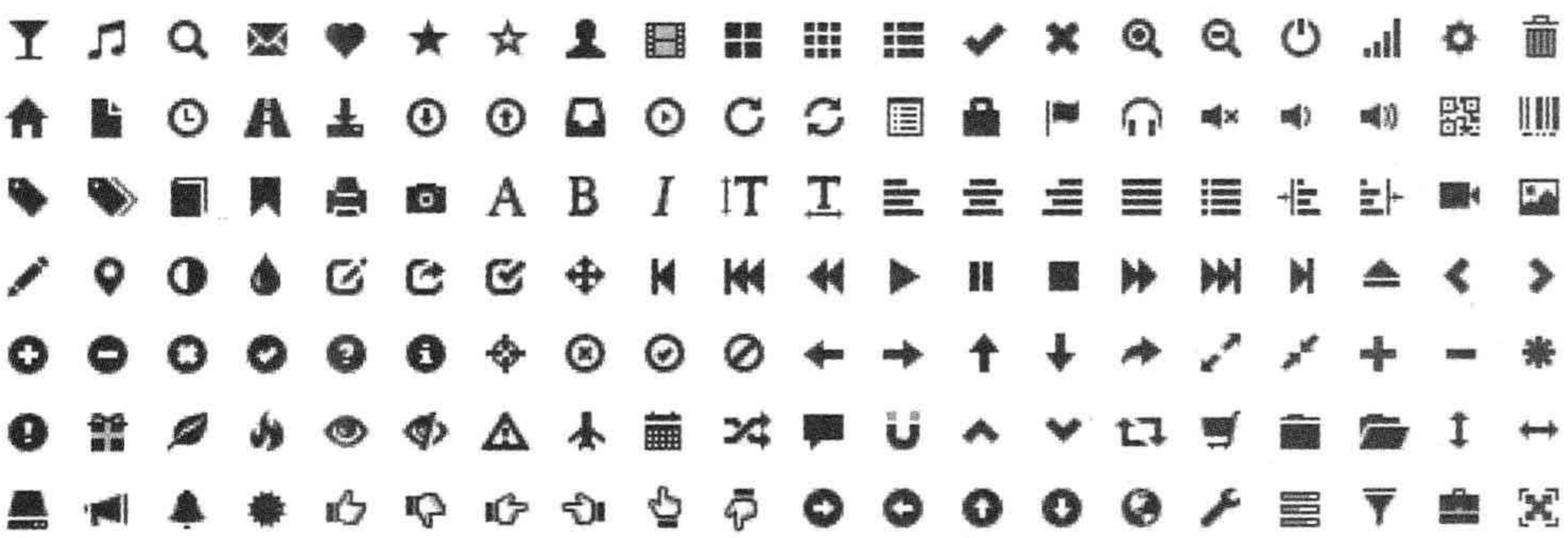


图 4-60　图标集合

使用时，只需要在标签中引入相应的图标样式类即可。建议使用 <i> 标签定义图标，然后将 <i> 标签放到任何需要的地方。

例如，下面的标签中都可以插入一个搜索图标，效果如图 4-61 所示。

```
<i class="icon-search"></i>
<span class="icon-search"></span>
<div class="icon-search"></div>
```

要使用反色图标，只需增加一个额外的类 icon-white。利用它可以设计图标的交互状态，如在导航条和下拉菜单中设计悬停和活动时候的状态效果。例如，分别为上面的标签引入 icon-white 样式类，即可设计如图 4-62 所示的效果。

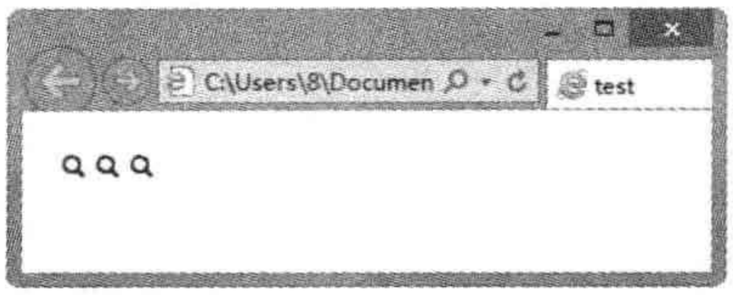


图 4-61　插入图标

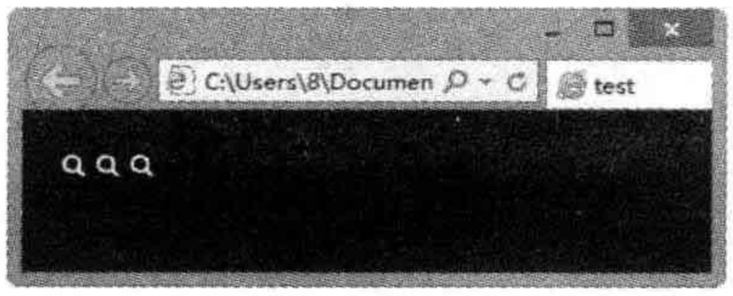


图 4-62　插入图标

提示　当在文本边插入图标时，应该在图标标签后面添加一个空格，这样能确保留有合适的间距。图标在导航、按钮和表单中广泛使用，以提升界面的体验，下面简单举例说明。

1. 按钮图标

使用图标标识按钮功能，往往会比文字更加形象和直观。例如，下面 5 个按钮分别使用图标模拟播放器按钮，如图 4-63 所示。

```
<div class="btn-toolbar">
    <div class="btn-group">
        <a class="btn" href="#"><i class="icon-play"></i></a>
        <a class="btn" href="#"><i class="icon-pause"></i></a>
        <a class="btn" href="#"><i class="icon-stop"></i></a>
        <a class="btn" href="#"><i class="icon-backward"></i></a>
        <a class="btn" href="#"><i class="icon-forward"></i></a>
    </div>
</div>
```

2. 导航图标

在导航中插入图标，导航栏会更生动。例如，下面的代码设计了一个简单的导航效果，如图 4-64 所示。

```
<ul class="nav nav-pills">
  <li class="active"><a href="#"><i class="icon-home icon-white"></i> 首页 </a></li>
  <li><a href="#"><i class="icon-book"></i> 资料 </a></li>
  <li><a href="#"><i class="icon-pencil"></i> 写日志 </a></li>
  <li><a href="#"><i class=" icon-film"></i> 视频 </a></li>
</ul>
```

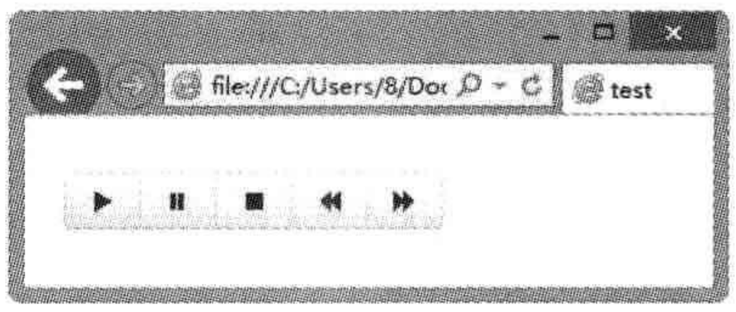


图 4-63　按钮图标效果

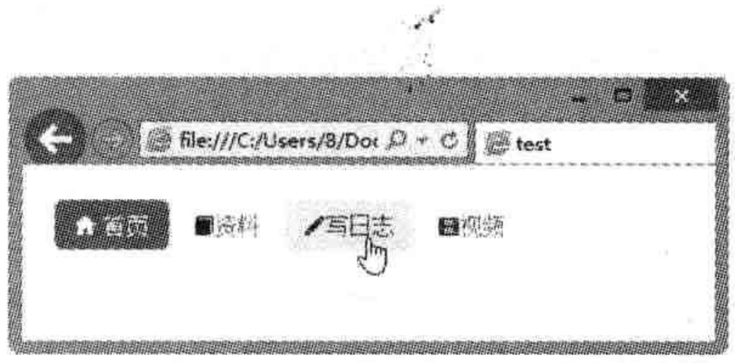


图 4-64　导航按钮图标

3. 表单图标

在表单控件中添加图标，会让表单对象更直观，甚至不需要 title 标签说明，如图 4-65 所示。

```
<div class="control-group">
    <label class="control-label"for="inputIcon"> 邮箱 </label>
    <div class="controls">
        <div class="input-prepend"><span class="add-on"><i class="icon-envelope"></i></span>
            <input class="span2"id="email"type="text">
        </div>
    </div>
    <label class="control-label" for="inputIcon"> 密码 </label>
    <div class="controls">
        <div class="input-prepend"><span class="add-on"><i class="icon-lock"></i></span>
            <input class="span2" id="pass" type="text">
        </div>
```

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

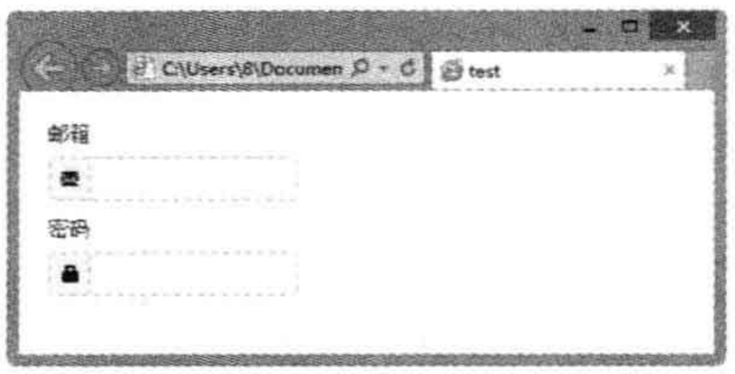


图 4-65　表单图标

Bootstrap 共定义了 140 个图标，这些图标名称以 icon- 作为前缀，后面是图标名称（语义名称），具体说明如图 4-66 所示。

图 4-66　Bootstrap 图标集合及其说明

提示　Bootstrap 3 版本开始，计划使用字体类型来设计图标，即使用 @font-face 版本的 Glyphicons 图标代替现在的 PNG 图标，使图标设计具有更大的灵活性和应用能力。

另外，也可以使用第三方专为 Bootstrap 设计的图标字体，如 Font Awesome。访问 http://www.bootcss.com/p/font-awesome/ 下载 Font Awesome 3，在页面中引入相应的 CSS 文件，即可快速使用，感兴趣的读者可以访问官网了解更详细的介绍。

第5章

CSS 动态样式——LESS

本章内容

- 为什么要使用 LESS
- 如何使用 LESS
- LESS 包含哪些内容
- LESS 动态语法
- Bootstrap 与 LESS 结合

在网页设计中，CSS 与 HTML、JavaScript 并列为 Web 前端开发的三种基础性语言。其中 CSS 负责网页的表现，HTML 负责网页的结构，JavaScript 负责网页的交互行为。

与 JavaScript 不同的是，HTML 和 CSS 都是标识语言。LESS 扩展了 CSS 语言功能，它在 CSS 的语法基础之上，引入了变量、混合（mixin）、运算和函数等特性，大大提升了 CSS 动态开发能力，降低了 CSS 的维护成本，就像它的名称所暗示的，LESS 可以让我们用更少的代码做更多的事情。

5.1　为什么要使用 LESS

LESS 提供了多种能将写好的代码平滑地转化成标准 CSS 代码的方式。在很多流行的框架和工具中已经能经常看到 LESS 的身影，如 Twitter 的 Bootstrap 库就使用了 LESS。那么，LESS 从何而来，它与 CSS、SASS 等样式表语言又有何区别呢?

5.1.1　LESS 概述

LESS 是 Alexis Sellier 受 SASS 的影响创建的开源项目。当时，SASS 采用了缩进作为分隔符来区分代码块，而不是 CSS 中广为使用的括号。为了让 CSS 现有用户使用起来更为方便，Alexis 开发了 LESS，并提供了类似的功能。

最初，LESS 的解释器是由 Ruby 编写的，后来才转而采用了 JavaScript。LESS 源代码既可以运行在客户端，也可以运行在服务器端。在客户端，只要在同一页面引用 LESS 代码和相应的 JavaScript 解释器即可；在服务器端，LESS 可以运行在 Node.js 上，也可以运行在 Rhino 这样的 JavaScript 引擎上。

简单来说，LESS 就是在网页设计的时候，可以更方便地编写 CSS 的工具。LESS 官网是这样描述的：LESS 使用动态行为（如变量、混合、运算、函数）拓展 CSS。LESS 能够运行在客户端（IE6+、Webkit、Firefox），也可以借助 Node.js 运行在服务器端。

也就是说，借助 LESS 语法，使用变量、混合、运算和函数等特性，再借助编译工具转换后，就可以将 LESS 代码转换成一般的 CSS 代码，从而可以设计更加有弹性的 CSS 样式。

为什么设计 CSS 需要更加有弹性呢？ CSS 不就是一个纯文本文档吗？写完就行了，需要编译那么复杂吗?

是的，CSS 原本只是一个描述样式的纯文本文件，用户也许觉得使用 Dreamweaver 等网页编辑工具设计完成就行了。但是，网页设计往往不是一次性的工作，可能需要经常去修改它。

此外，就算是在刚开始设计的阶段，LESS 也可以提供很多方便的工具：传统写 5 行的样式声明，现在只要写 1 行；过去改一个颜色值要改多个地方，很容易遗漏，现在用户只要改一个地方，不会有遗漏，由此可见 LESS 的效率之高。

5.1.2　LESS 的优势

CSS 是一门历史悠久的标识语言，与 HTML 一起，被广泛应用于 Web 应用和设计开发

中。HTML 主要负责文档结构的定义，CSS 负责文档表现样式的定义。作为一门标识语言，CSS 语法相对简单，对使用者的要求较低，但同时也带来一些问题：CSS 需要书写大量看似没有逻辑的代码，不方便维护及扩展，不利于重用，非前端开发人员，往往会因为缺少 CSS 编写经验而很难写出组织良好且易于维护的 CSS 代码。造成这些困难的很大原因源于 CSS 是一门非程序设计语言，没有变量、函数、作用域等概念。

LESS 为 Web 开发者带来了福音，它在 CSS 的语法基础之上引入了变量、混合、运算以及函数等特性。LESS 的目标是简化 CSS 使用，降低 CSS 维护成本，让 CSS 可编程，让代码更加优雅，以更少的 CSS 代码做更多的事。

一直以来，大家都习惯于使用 JavaScript 扩展，还没有使用过 CSS 的扩展。LESS 最早是一个 Ruby 的 gem，实现 CSS 具有动态语言的特性，这些特性包括变量、运算和嵌套规则等。其实 LESS 真正的作用是将使用高级特性的 CSS 转换成标准的 CSS。这些都是在 Web 客户端发起请求时通过 HTTP 请求来完成的，也可以是编辑时就完成的。此外，LESS 可以配置成自动最小化所生成的 CSS 文件，不仅节省了带宽，并且使最终用户体验更上一层楼。

LESS 具有以下明显的优点：

1）需要编写的代码量明显变少了。

2）CSS 管理更加容易了，在需要更换网站样式风格时尤其如此，此时如果直接重写这些样式，工作量将非常浩大，但是用 LESS 就很简单，改个全局配置就可以了。

3）LESS 学习成本不是很高，与 CSS 规则完全融合，如果用户熟悉 CSS 的话，那么只需要简单的学习，就能够快速驾驭 LESS。

4）使用 LESS 实现配色变得非常容易。在传统设计中，用户需要借助配色工具或者图像编辑软件来实现色彩的搭配，不但效果不是很理想，而且还需要投入很大的精力。

例如，在设计的时候，常常需要反复试验哪种颜色比较适合。虽然知道主色调要用原色，但是根据设计意图，可能需要深一点、浅一点、亮一点、暗一点，或者再带一点黄色，再带一点棕色，如此等等。如果每试一次都要改全套的样式，那么这个工作量是非常大的，而且也不能够精确量化。现在，利用 LESS 的变量，结合颜色函数，所有配色问题就变得容易多了。在网站改版中也是如此。

5）兼容 CSS3。很多 CSS3 语法目前还需要为各个浏览器写特别的语法，如圆角、盒子阴影、变形、过渡等，如果把这些代码使用 LESS 先封装起来，使用时就会省事很多。

6）与 CSS 能够很好地融合使用。在 LESS 代码中可以融入 CSS 代码，在 CSS 代码中可以插入 LESS 语法。因此，对于用户来说，有了 LESS，同样还需要继续学习 CSS，还是需要会写 CSS，LESS 只是帮我们省下一些工夫，大部分样式还是要知道 CSS 是如何实现的，否则就不知道如何把 LESS 编译成为 CSS。

5.1.3 LESS 参考和工具

下面列出学习 LESS 需要参考的网站和辅助工具。

1. 参考网站

- LESS 官方网站：http://lesscss.org/
- LESS 中文网站：http://www.lesscss.net/
- LESS 中文参考：http://www.bootcss.com/lesscss.html

2. 在线编译工具

下面两个在线编译工具可以快速把 LESS 源代码编译为 CSS 源代码，用户不需要在本地配置服务器环境，或者安装本地编译工具。

- 开源中国提供的在线 LESS CSS 编译器：http://www.ostools.net/less
- LESS 在线提供的 LESS CSS 在线编译工具：http://less.cnodejs.net/#

3. 本地编译工具

CodeKit（http://incident57.com/less/）

CodeKit 是非官方的 Mac 应用，可以编译 LESS、SASS、Stylus 和 CoffeeScript，是 Web 前端的全能工具，能够提供强大的 LESS 编译功能。对于 Mac 用户来说，它就是 less.app。less.app 是一个第三方提供的工具，使用起来十分方便，在图 5-1 所示的界面中添加 LESS 文件所在的目录，此工具就会在右侧列出目录中包含的所有 LESS 文件。

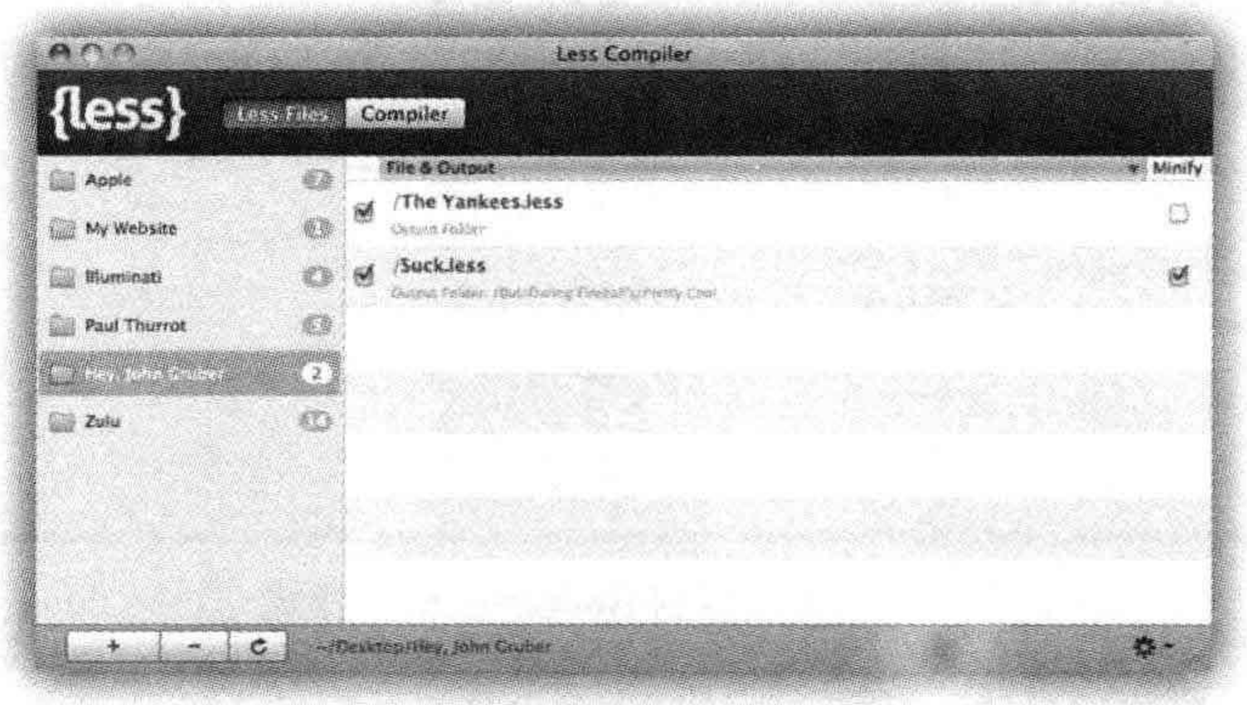


图 5-1　less.app 应用开发界面

应用该工具之后，就不用再考虑如何把 LESS 文件编译成 CSS 文件了，因为它会在每次修改完保存 LESS 文件时执行编译，自动生成 CSS 文件。这样就可以随时查看 LESS 代码的最终效果，检查目标 CSS 是否符合需要了。

- SimpLess（http://wearekiss.com/simpless）

SimpLess 是一款免费的离线 LESS 代码编译器，支持跨平台使用，可在 Mac、Windows 和 Linux 平台上使用，能够自动检测代码变化并编译它。可以拖拽 LESS 文件进行编译，而且其代码开源，代码托管在 GitHub 上（https://github.com/Paratron/SimpLESS）。

- WinLess（http://winless.org/）

WinLess 是 Windows 下 less.js 图形用户界面（GUI）开发工具，是 Windows 下的 Web 开

发必备工具。

- Crunch（http://crunchapp.net/）

  Crunch 是用 Abode Air 构造的界面优美的 LESS 编辑器和编译器。

5.2 如何使用 LESS

LESS 可以直接在客户端使用，也可以在服务器端使用。在实际项目开发中，建议使用第三种方式，提前将 LESS 文件编译成静态的 CSS 文件，然后在 HTML 文档中应用。例如，在 Bootstrap 框架中，通过在线定制 LESS，提前生成 CSS 文件，以提高页面响应速度。访问 http://www.bootcss.com/customize.html，快速定制 Bootstrap 框架样式，如图 5-2 所示。

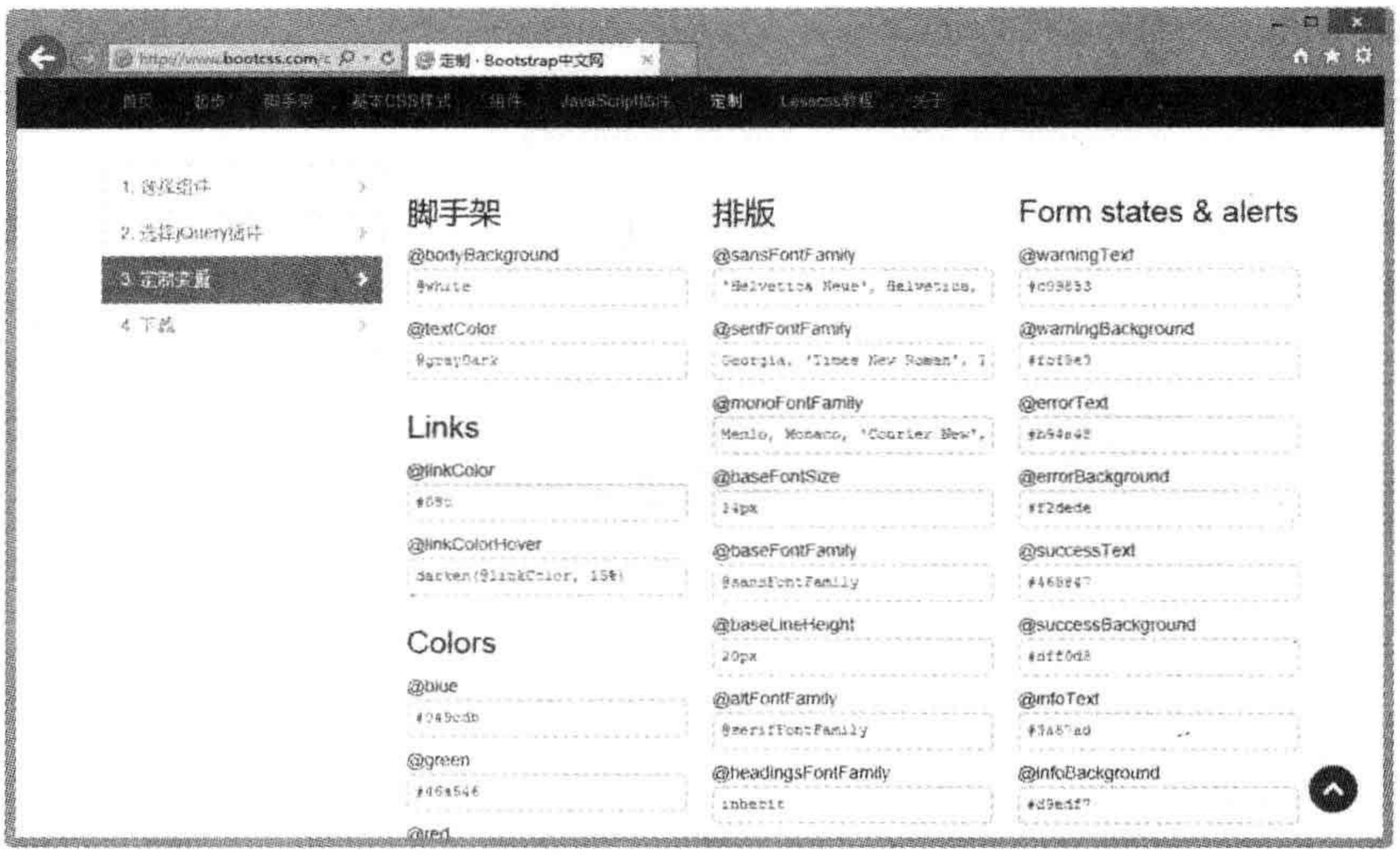


图 5-2 在线定制 Bootstrap 框架样式

如果读者希望直接使用 LESS，则可以在客户端或者服务器端直接编译应用，详细说明如下。

5.2.1 在客户端使用 LESS

在客户端使用 LESS，只需要下载 less.js 文件，然后在需要使用 LESS 源文件的 HTML 文档中导入即可。下面通过示例来演示如何在客户端正确使用 LESS。

第 1 步：下载 less.js 文件。可访问 http://lesscss.org 或 http://lesscss.googlecode.com/ 下载最新版本（1.3.0 版）的 less.js 文件。（访问 https://github.com/cloudhead/less.js 可以下载未压缩的 Master 版本。）

第 2 步：创建 LESS 文件，LESS 文件与 CSS 文件结构相近，不同点是 LESS 源文件多了编程语言的变量、函数等特性。可以把 CSS 文件改为 LESS 文件，只需要把 CSS 文件的后缀名改为 .less 即可。

第 3 步：新建 HTML5 文档，在文档头部区域加入以下代码：

```
<link type="text/less"rel="stylesheet/less"rev="stylesheet/less"href="style.less"/>
<script language="JavaScript"type="text/JavaScript"src="less1.0.33.min.js"></script>
```

可以看到，LESS 源文件的导入方法与标准 CSS 文件导入方式一样，实际上 LESS 文件就是样式表文件。需要注意的是，在导入 LESS 文件时，需要设置 rel 属性值为“ stylesheet/less”，指定与本文档关联的外部文件类型为层叠式样式；或者设置 type 属性值为“ text/less”，指定导入的外部文件 MIME 类型为 LESS。

提示　<link> 标签的 type 属性必须是“ text/less”，在导入外部 CSS 文件时，习惯将其设置为“text/css”，这样容易引发歧义，而不能够正确导入外部 LESS 文件。

在导入外部 LESS 源文件之后，应再导入 less.js，该 JavaScript 文件作为 LESS 源文件的解析器，负责把 LESS 文件编译为 CSS 文件，并插入到当前文档中。

第 4 步：设计 HTML 文档结构，代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<link type="text/less" rel="stylesheet/less" rev="stylesheet/less" href="style.
  less" />
<script language="JavaScript" type="text/JavaScript" src="less1.0.33.min.js"></
  script>
</head>
<body>
    <div class="div1">盒子 1</div>
    <div class="div2">盒子 2</div>
    <div class="div3">盒子 3</div>
    <div class="div4">盒子 4
        <div>
            <span>行内元素 1</span>
        </div>
        <span>行内元素 2</span>
    </div>
    <div class="div5">盒子 5</div>
    <div class="div6">盒子 6</div>
</body>
</html>
```

第 5 步：设计 LESS 动态样式。

新建文本文件，另存为 style.less，注意扩展名为 .less。然后输入下面动态样式代码，读者可以直接复制本书实例源代码，暂时先不考虑源代码的具体意义和功能。

下面就几点进行简单说明，读者可以先睹为快，也可以跳过。后面的章节会详细讲解 LESS 语言语法。

1）定义变量 @colors，用来存储颜色值。

```
@colors: #333;
.div1{
      color: @colors;
      font-weight: bold;
      background-color: #CCC;
}
```

这样，当需要修改颜色的时候只需修改变量的值即可。

2）样式内嵌，快速重用样式代码。

```
.div3{
     border: #222 solid 1px;
     .div1
}
```

这样就可以直接在 .div3 样式中嵌入 .div1 样式类，而无须复制代码了。

3）嵌套规则

```
@fonts: 12px;
.div4{
     border: #333 solid 1px;
     padding: 10px;
     div{
          background-color: red;
          span{
              color: red;
          }
      }
      span{
          background-color: @colors;
          font-size: @fonts * 2;
     }
}
```

这样可以使样式的名称更为简短，并且修改的时候更容易查找，后期维护也比较方便，因为样式按标签的层级结构关系进行嵌套，样式富有层次感。

4）样式运算，在样式声明中可以传递简单的 JavaScript 表达式，实现灵活的赋值。

```
span{
     background-color: @colors;
     font-size: @fonts * 2;
     color: @colors;
}
```

样式运算增强了动态样式的灵活性，用户可以在样式表中添加表达式，以实现智能计算。

5）样式传参

```
.div5(@widths: 5px){
      color: red;
```

```
        border-style: solid;
        border-color: @colors;
        border-width: @widths;
}
.div6{
    .div5(10px);
}
```

可以把样式作为函数使用，然后通过参数动态修改样式内部值，这样，一个样式就可以在样式表中多处调用。

第 6 步：浏览动态 CSS 设计的文档效果。在本地使用 Firefox、Opera、Safari 浏览器预览文档，显示效果如图 5-3 所示。

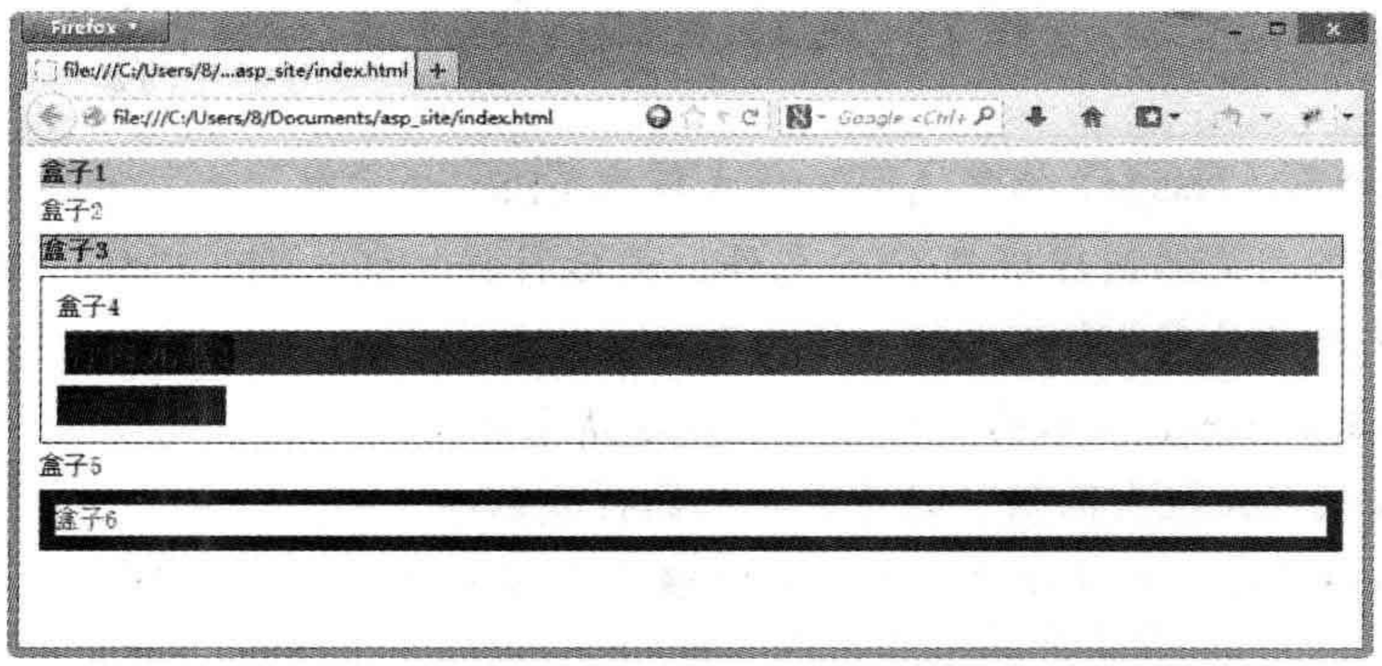


图 5-3　浏览动态 CSS 样式效果

需要说明的是，在浏览动态 CSS 样式的文档时，由于安全限制的原因，IE 和 Chrome 浏览器无法查看到 LESS 的效果，可能会提示 HTTP 404 错误。如果出现这种状况，建议使用服务器进行访问。例如，在本地启动 IIS 虚拟服务器服务，同时在 IIS 中添加 MIME 类型，如图 5-4 所示。

图 5-4　在 IIS 中添加 .less 文件类型的 MIME 信息

通过上面的配置，就可以在IE和Chrome浏览器中通过向服务器（http://localhost/mysite/index.html）发出请求，访问当前页面，效果如图5-5所示。

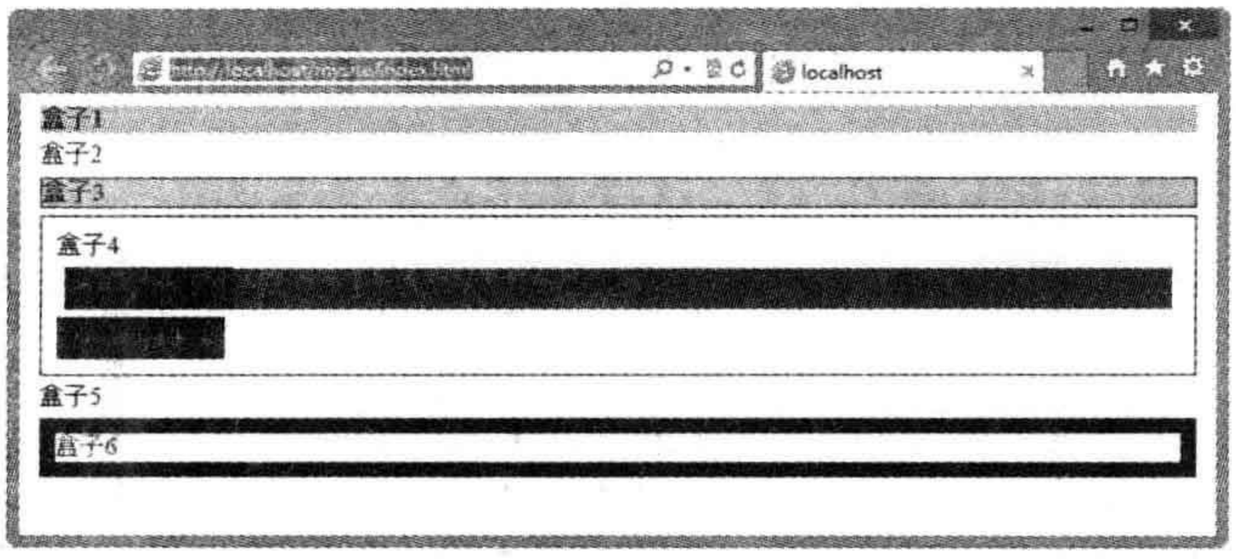


图5-5　通过服务器访问当前页面

提示　LESS编译器（less.js）使用AJAX技术获取LESS源文件，然后根据该文件中所定义的规则，生成最终浏览器能理解的CSS文件，最后再将其插入到HTML代码中。因此，使用<link>标签导入的外部LESS文件必须位于less.js文件前。

基于LESS JavaScript版本的实现原理，每次请求都需要通过JavaScript动态生成原始的CSS，如果CSS代码比较大的话，对于客户端的性能影响比较大，所以在商业应用时，这种方法的实用性不是很强，应该避免。可以通过服务器预编译来提高效率。

5.2.2　在服务器端使用LESS

LESS在服务器端的使用主要是借助于LESS的编译器，将LESS源文件编译成最终的CSS文件。目前常用的方式是利用node包管理器（npm）安装LESS，安装成功后就可以在node环境中对LESS源文件进行编译。

1. 搭建Node.js Windows环境

在Windows环境下下载Node.js（稳定版），直接安装。

Node.js参考文档：http://www.ostools.net/apidocs/apidoc?api=nodejs%2Fapi

Node.js下载地址：http://nodejs.org/download/

2. 搭建Node.js Linux环境

以Windows 7为开发环境，配合Vbox虚拟机、Ubuntu搭建Node.js编译环境，然后通过securecrt远程连接到虚拟机进行开发。如果securecrt远程连接被拒绝，则需要sudo apt-get install openssh-server。具体步骤如下。

第1步：安装开发包。

1）安装Python 2.6版或者更高（Ubuntu默认都已安装，可以在terminal中使用python-v命令查看Python版本）。

2）安装其他依赖包：sudo apt-get install g++ curl libssl-dev apache2-utils。

3）安装 git 工具：sudo apt-get install git。

第 2 步：获取源码。

```
git clone git://github.com/joyent/node.git
```

第 3 步：指定编译版本。

首先，进入存放下载源码的文件夹：

```
cd node
```

然后，指定迁出版本：

```
git checkout v0.6.12（版本的选择，遵循稳定原则）
```

最后，指定路径，编译执行：

```
mkdir ~/local
./configure -prefix=$HOME/local/node
make
make install
echo 'export PATH=$HOME/local/node/bin:$PATH' >> ~/.profile
echo 'export NODE_PATH=$HOME/local/node:$HOME/local/node/lib/node_modules' >> ~/.profile
source ~/.profile
```

第 4 步：设置环境变量。

如果想重启后还能继续直接使用 node 命令，那么需要设置环境变量。使用命令 sudo gedit/etc/profile 打开配置文件，在文件最后添加如下两行：

```
export PATH="$HOME/local/node/bin:$PATH"
export NODE_PATH="$HOME/local/node:$HOME/local/node/lib/node_modules"
```

保存后重启系统使设置生效。

第 5 步：安装 npm。

```
curl https://npmjs.org/install.sh | sh
```

根据需要安装相应的包，如 express：

```
npm install express -gd
```

-g 代表安装到 NODE_PATH 的 lib 里面，而 -d 代表把相依性套件也一起安装。如果没有 -g 的话会安装到目前所在的目录（会建立一个 node_modules 的文件夹）。

第 6 步：通过 npm 按需安装文件包。

以一个练习 Demo（https://github.com/cmarin/MongoDB-Node-Express-Blog）进行说明。这个 Demo 需要安装的依赖包已经标明，按照命令操作即可。

首先，cd 到自己的工作目录 git clone git://github.com/cmarin/MongoDB-Node-Express-Blog.git 获取源码。

然后安装数据库，直接在命令行里输入 sudo apt-get install mongodb 即可，安装完成后测试方法，在终端命令行中输入：

```
mongo
db.foo.save({a:1})
db.foo.findOne()
```

继续输入：

```
npm install express
npm install express-messages
npm install ejs npm install sass
npm install mongoose

Then cd into the directory and run: node app.js
```

此时会看到终端的 log 提示：

```
You can debug your app with http://localhost:3000
```

表明已经安装成功。

此时通过 ip 就可以访问了，如 http://192.168.1.106:3000/。

3. 安装注意事项

有时候由于异常关机 MongoDB 数据库会被锁住，提示“ Error: couldn't connect to server 127.0.0.1}”。解决方案是：

```
sudo rm /var/lib/mongodb/mongod.lock
sudo chown -R mongodb:mongodb /var/lib/mongodb/
sudo -u mongodb mongod -f /etc/mongodb.conf --repair
sudo service mongodb start
```

- 在 Node.js 开发阶段，如果用 node xxx.js 来运行不能时时检测 .js 文件的变化，这样调试起来就很麻烦，所以需要安装一个开发调试脚本。

```
npm install -g node-dev
```

使用 node-dev app.js 来调试。

- 如果需要在 nodeserver 之前再搭建一层 nginx，基础配置步骤如下：

1）安装 pcre。

```
sudo apt-get install libpcre3-dev
```

2）安装 zlib。

```
$ tar xzf zlib-1.2.3.tar.gz
$ cd zlib-1.2.3/
$ ./configure
$ make
$ sudo make install
```

3）nginx-1.0.8$./configure--prefix=<所要安装的 path，如安装在 /home/ 机器名 /work/nginx>

```
nginx-1.0.8$ make
nginx-1.0.8$ sudo make install
```

4）检查配置是否正确。

```
root@ubuntu:/home/# /usr/local/nginx/sbin/nginx -t
```

根据上面不同的 --prefix 需要找不同的路径。此处是默认安装不指定 prefix 的默认目录。

```
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

5）启动 nginx。

```
root@ubuntu:/home/# /usr/local/nginx/sbin/nginx
root@ubuntu:/home/# ps -ef | grep nginx
root     1436     1  004:58 ?   00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nobody     1437  1436  004:58 ?   00:00:00 nginx: worker process
root       1439  1413  004:58 pts/0  00:00:00 grep --color=auto nginx
```

6）测试。

```
root@ubuntu:/home/# curl http://localhost
<html>
<head>
<title>Welcome to nginx!</title>
</head>
<body bgcolor="white" text="black">
    <center><h1>Welcome to nginx!</h1></center>
</body>
</html>
```

或者

```
root@ubuntu:/home/xiong# ifconfig
eth0 Link encap:Ethernet  HWaddr 00:0c:29:7b:80:c2
inet addr:192.168.130.131  Bcast:192.168.130.255  Mask:255.255.255.0
```

浏览器输入 http://192.168.130.131/ 关闭 nginx：

```
nginx -s stop     快速关闭 Nginx  可能不保存相关信息，并迅速终止 web 服务。
nginx -s quit     平稳关闭 Nginx，保存相关信息，有安排地结束 web 服务。
```

重启 nginx：

```
nginx -s reload     因改变了 Nginx 相关配置，需要重新加载配置而重载。
nginx -s reopen     重新打开日志文件。
```

至此 nginx 基本配置完毕，可以启动了。简单放一个基础的 nginx.conf 文件：

```
#user  nobody;
worker_processes  1;
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;
error_log  logs/error.log  debug; # 第一处  修改开启日志
pid        logs/nginx.pid;
events {
    worker_connections  1024;
}
http {
     include       mime.types;
     default_type  application/octet-stream;
     # 第二处 开启日志格式化
     log_format main '$remote_addr-$remote_user [$time_local] "$request"''$status
      $body_bytes_sent "$http_referer"'  '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  logs/access.log  main;
    sendfile        on;
    #tcp_nopush     on;
    #keepalive_timeout  0;
    keepalive_timeout  65;
    #gzip  on;
    server {
    listen       9199; # 第三处 修改默认 80 端口，不然启动 nginx 需要 root 权限
    server_name  localhost;
    #charset koi8-r;
    #access_log  logs/host.access.log  main;
    location /xxx {
        root   html;
        index  index.html index.htm;
    }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }
    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    location / {
        proxy_pass   http://127.0.0.1:3001;  # 第四处 所有 9199 端口下直接 proxy_pass 到其他
          服务端
    }
    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
```

```
#    fastcgi_index  index.php;
#    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
#    include        fastcgi_params;
#}
# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#    deny  all;
#}
}
# another virtual host using mix of IP-, name-, and port-based configuration
#
#server {
#    listen       8000;
#    listen       somename:8080;
#    server_name  somename  alias  another.alias;
#    location / {
#        root   html;
#        index  index.html index.htm;
#    }
#}
# HTTPS server
#
#server {
#    listen       443;
#    server_name  localhost;
#    ssl                  on;
#    ssl_certificate      cert.pem;
#    ssl_certificate_key  cert.key;
#    ssl_session_timeout  5m;
#    ssl_protocols  SSLv2 SSLv3 TLSv1;
#    ssl_ciphers  HIGH:!aNULL:!MD5;
#    ssl_prefer_server_ciphers   on;
#    location / {
#        root   html;
#        index  index.html index.htm;
#    }
#}
}
```

4. 使用 LESS

添加 less 组件命令如下：

```
node npm install less
```

编译 less 文件命令如下：

```
lessc styles.less styles.css
```

5.3 LESS 包含哪些内容

LESS 包含一套自定义的语法和一个解析器，用户可根据这些语法自定义样式规则，而这些规则通过解析器，编译生成对应的 CSS 文件。LESS 没有裁剪 CSS 原有的特性，更不会取代 CSS，而是在现有 CSS 语法的基础上，为 CSS 加入编程语言的特性。

5.3.1 LESS 基本特性

LESS 是一种动态样式语言，拥有四大特性：变量、混合、嵌套和运算。下面进行简单的介绍。

1. 变量

变量是 LESS 里面最重要的特性。在设计的时候，常常会在很多个地方使用相同的颜色值（或是用很相近的颜色）来营造整体的感觉，如 h1、h2、h3、button、ink hover color 等。

在以往设计 CSS 的时候，可能需要在这些标签样式里面做各自的设定。但现在使用变量，可以在最上面声明一个 base color，然后在其他地方反复使用这个 base color。这样，网站在进行设计上的调整时，就可以省下很多时间。

通过 @ 的方式声明变量，甚至可以通过算法操作对变量进行运算，这样有利于对全局变量的重复运用，实现全局样式动态配置。在网站开发中，可以在样式表最前面定义好全局样式，通过变量存储需要动态配置的样式值，然后在属性中直接传递变量，而不是具体的值，这样如果需要改动样式，只需要为变量赋值即可。例如，在下面的代码中，通过全局变量 @color 存储网页基本字体色，在具体样式声明中，直接传递变量 @color; 即可。

```
// LESS
@color: #4D926F;
#header { color: @color;}
h2 { color: @color;}
/* 转换为 CSS */
#header { color: #4D926F;}
h2 { color: #4D926F;}
```

2. 混合

混合（mixin）可以重复利用某些样式的宣告，用户可以在 A 样式里面包含另一个 B 样式，B 中所有的类样式都会继承 A 样式中的设定。

这个最好用的地方就在于简化一些比较不好写的 CSS，如 CSS3 的圆角设定，目前因为浏览器的语法尚未统一，可以按下面的示例代码进行编写。通过定义参数来达到值传递的目的，类似函数，可以将一些 Hack 整理成统一的函数来达到牵一发而动全身的目的。

混合特性类似于编程语言中的继承。设计好一个样式类，然后在其他样式中直接混合这个样式类，实现样式的继承重用。就像函数一样调用，并且可以传递参数，功能非常强大、实用。

例如，下面先定义一个样式类 rounded-corners，并为它定义一个参数 @radius，默认值为 5 像素。然后分别在不同样式中调用这个样式类，在 #header 中使用默认值，而在 #footer 中重新传递一个新参数。

```
// LESS
.rounded-corners (@radius: 5px) {
     border-radius: @radius;
     -webkit-border-radius: @radius;
      -moz-border-radius: @radius;
}
#header { .rounded-corners;}
#footer { .rounded-corners(10px);}
/* 转换为 CSS */
#header {
    border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
}
#footer {
    border-radius: 10px;
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
}
```

3. 嵌套

CSS selector 里面有一个重要的用法叫做后代选择器。这样的写法虽然很好理解，但要对于编写的人或是要修改的时候，就不是那么方便了，因为相关的样式可能散布在 CSS 文件的任何地方。在 LESS 里面，使用嵌套可以避免这种问题。

嵌套规则可以更容易设计模块化 CSS、精简代码，让 CSS 的层级更加具有归属感，甚至可以用 & 符号来声明伪类属性。这样就不用编写很长的复合选择器，而且这种嵌套结构更容易了解样式之间的关系。

```
// LESS
#header {
    h1 {
        font-size: 26px;
        font-weight: bold;
    }
    p { font-size: 12px;
         a { text-decoration: none;
               &:hover { border-width: 1px }
         }
    }
}
/* 转换为 CSS */
#header h1 {
    font-size: 26px;
```

```
    font-weight: bold;
}
#header p { font-size: 12px;}
#header p a { text-decoration: none;}
#header p a:hover { border-width: 1px;}
```

4. 运算

运算是动态的核心，通过嵌入 JavaScript 表达式，以及使用简单的四则运算符，可以模仿 JavaScript 进行计算，可以通过变量的 +、-、*、/ 运算操作来达到一些运算的目的。想象一下：一个容器，包含三个等量大小的子容器，那么只需要做简单的除法运算即可，或许更复杂运算，可以更直接体现 LESS 的优势。如获取浏览器窗口尺寸，设计递增、递减样式，设置动态宽度等。

例如，在下面的代码中，通过简单的运算，就可以获取颜色递增、递减、值的有规律变化等。

```
// LESS
@the-border: 1px;
@base-color: #111;
@red:       #842210;
#header {
    color: @base-color * 3;
    border-left: @the-border;
    border-right: @the-border * 2;
}
#footer {
    color: @base-color + #003300;
    border-color: desaturate(@red, 10%);
}
/* 转换为 CSS */
#header {
    color: #333;
    border-left: 1px;
    border-right: 2px;
}
#footer {
    color: #114411;
    border-color: #7d2717;
}
```

LESS 提供了丰富的颜色和数学函数，没有你想不到的。例如：

```
@base: #f04615;
.class {
     color: saturate(@base, 5%);
     background-color: lighten(spin(@base, 8), 25%);
}
```

这里都是 LESS 的简单用法，当你完全掌握 LESS 语法之后，你会发现可以用 LESS 做

更多 JavaScript 的工作。

5.3.2　LESS 主要功能

在语法上 LESS 主要包含下面几个功能。

❑ 混合（mixin），class 中的 class。

mixin 可以译成“混合”，也可以译成“混入”。其主要意思是将一个定义好的 class A 引入到 class B 中，从而简单实现 class B 继承 class A 的所有属性。

❑ 参数混合（parametric），可以像函数一样传递参数的 class。

参数混合像函数一样在 class A 中定义一个参数的默认值或者参数属性集合，还可以是 @arguments 变量，然后将定义好的 class A 引入 class B 中。

❑ 嵌套规则（nested rule），class 中嵌套 class，从而减少重复的代码。

嵌套规则指的是在一个选择器中嵌套另一个选择器，用来实现样式继承，从而减少代码量，并且增加代码的可读性。

❑ 运算（operation），CSS 中的数学计算。

运算指在 CSS 中使用加、减、乘、除进行数学运算，主要运用于属性值和颜色的运算，可以轻松实现属性值之间的复杂关系。

❑ 颜色功能（color function），可以编辑你的颜色。

颜色功能是对颜色的函数运算，颜色会先被转化成 HSL 色彩空间，然后在通道级别操作。

❑ 命名空间（namespace），样式分组，从而方便被调用。

命名空间将一些变量或者混合模块打包封装，更好地组织 CSS 和属性集的重复使用。

❑ 作用域（scope），局部修改样式。

作用域先从本地查找变量或者混合模块，如果没有找到的话就会去父级作用域中查找，直到找到为止，这一点和编程语言的作用域非常相似。

❑ JavaScript 表达式（JavaScript evaluation），在 CSS 样式中使用 JavaScript 表达式赋值。

上面 8 条是 LESS 中很重要的概念，只有把上面的概念理解清楚了，才能更好地学习 LESS。

5.3.3　LESS 和 SASS

LESS 和 SASS 都是动态样式开发工具，可以帮助开发者写出重用性更优秀的 CSS 文件。

SASS 由 Haml 的团队开发，它采用了 Haml 的设计思想，使用缩进而不是括号这样的分隔符来定义代码块或者内嵌级别。LESS 是受 SASS 启发而开发的工具，其开发者列出了如下开发理由。

❑ 为什么要开发一个 SASS 的替代品呢？原因很简单：首先是语法。SASS 的一个关键特性是缩进式的语法，这种语法可以产生柱式外观的代码，但是用户需要花费时间学习一门新的语法以及重新构建现在的样式表。

❑ LESS 给 CSS 带来了很多特性，使得 LESS 能够和 CSS 无缝地紧密结合在一起。因

此，用户可以平滑地由 CSS 迁移到 LESS，如果只是对使用变量或者操作感兴趣的话，不需要重新学习一整门全新的语言。

- LESS 的解析器是使用 TreeTop 编写的，TreeTop 是一个用 Ruby 编写的 PEG 解析器的生成器（LESS TreeTop 语法）。
- LESS 和 SASS 工具（编译器和 API）能够作为 gem 安装，使用命令行工具进行编译，但是也可以在 Ruby 代码中使用。

SASS 看起来在提供的特性上占有优势，但是 LESS 能够让开发者平滑地从现存 CSS 文件过渡到 LESS，而不需要像 SASS 那样需要将 CSS 文件转换成 SASS 格式。SASS 的维护者 Nathan Weizenbaum 在一篇对比 LESS 和 SASS 的博文中提到，未来 SASS 将会提供括号，而不是像 CSS 或者 LESS 那样的缩进。现在，SASS 已经有了两套语法规则：一套依旧是用缩进作为分隔符来区分代码块的；另一套规则和 CSS 一样，采用大括弧作为分隔符。后一种语法规则又名 SCSS，在 SASS 3 之后的版本都支持这种语法规则。SCSS 和 LESS 已经越来越像了。

不过，LESS 和 SASS 的方法基本类似，两种语言给 CSS 添加的特性都是相似的，它们之间的详细对比可以参考 https://gist.github.com/chriseppstein/674726，简单说明如下。

（1）变量

LESS 中的 @name 和 SASS 中的 !name 都是变量。可以给变量赋值，然后在代码中使用它们。

（2）样式内嵌

将选择器嵌入到其他样式中，取消了一些高级选择器嵌套。LESS 和 SASS 都将这个简洁的特性扩展到了 CSS。

（3）混合类型

允许开发者抽象出声明的共同点，然后命名并且加入到选择器中。熟悉 Ruby 混合类型的开发者会了解混合类型在 CSS 中的应用。SASS 也允许将混合类型作为参数，使得混合类型的应用更加灵活。

（4）表达式

LESS 和 SASS 都支持简单的算术运算，如加法、减法等。将这个特性和变量结合起来，会使得 CSS 变得更加灵活。这两个工具需要保证运算的正确性。

LESS 和 SASS 之间的主要区别是它们的实现方式不同。LESS 基于 JavaScript 运行，因此 LESS 可以在客户端处理。SASS 是基于 Ruby 的，是在服务器端处理的。很多开发者不选择 LESS，是因为 LESS 将修改过的 CSS 输出到浏览器需要依赖于 JavaScript 引擎，而 JavaScript 引擎需要额外的时间来处理代码。关于这个有很多种方式。一种方式是只在开发环节使用 LESS，开发完成后，就复制、粘贴 LESS 输出到一个压缩器，然后到一个单独的 CSS 文件来替代 LESS 文件。另一种方式是使用 LESS.app 来编译和压缩 LESS 文件。两种方式都将是最小化样式输出，从而避免由于用户的浏览器不支持 JavaScript 而可能引起的任何问题。

5.4　LESS 动态语法

LESS 是 CSS 的一种扩展形式，它没有删减 CSS 规则和功能，严格遵循 CSS 基本语法规则，只是在现有的 CSS 语法基础之上，添加了很多额外的功能，所以学习 LESS 是一件轻而易举的事情。

5.4.1　变量

对于程序开发人员来说，变量应该是最熟悉不过的概念了。如果重复使用一个信息（如网页颜色 color），将它设置为一个变量，就可以在代码中重复引用。使用这种方式可保证网页色彩设计的一致性，并减少因需要修改颜色值而在大量样式代码中查找、复制和粘贴等烦琐的工作。

LESS 规定变量名以 @ 为前缀，语法格式与样式声明相同。例如下面代码中声明一个变量 @blue，其值为 #0000ff，然后把这个变量传递给样式 #header 中的 color 属性。

```
@blue: #0000ff;
#header { color: @blue; }
```

输出：

```
#header { color: #0000ff; }
```

还可以用变量名定义为变量，例如：

```
@blue: #0000ff;
@b: 'blue';
#header { color: @@b; }
```

输出：

```
#header { color: #0000ff; }
```

注意，LESS 变量为完全常量，所以只能定义一次。

在样式声明中，变量可以参与简单的运算，如针对上面的变量 @blue，可以使用加减颜色值，从而得到需要的颜色。例如：

```
@blue: #00c;                    /* 定义蓝色变量 */
@light_blue: @blue + #333;      /* 定义浅蓝色变量 */
@dark_blue: @blue - #333;       /* 定义深蓝变量 */
```

将上面 3 个样式分别应用到 3 个 标签的背景上，就可以看到加上、减掉十六进制的颜色值和原始的蓝色形成的渐变效果，即从 @light_blue 到 @blue，再到 @dark_blue 的渐变效果，如图 5-6 所示。

```
.bg1 {background: @light_blue;}
.bg2 {background: @blue;}
.bg3 {background: @dark_blue;}
```

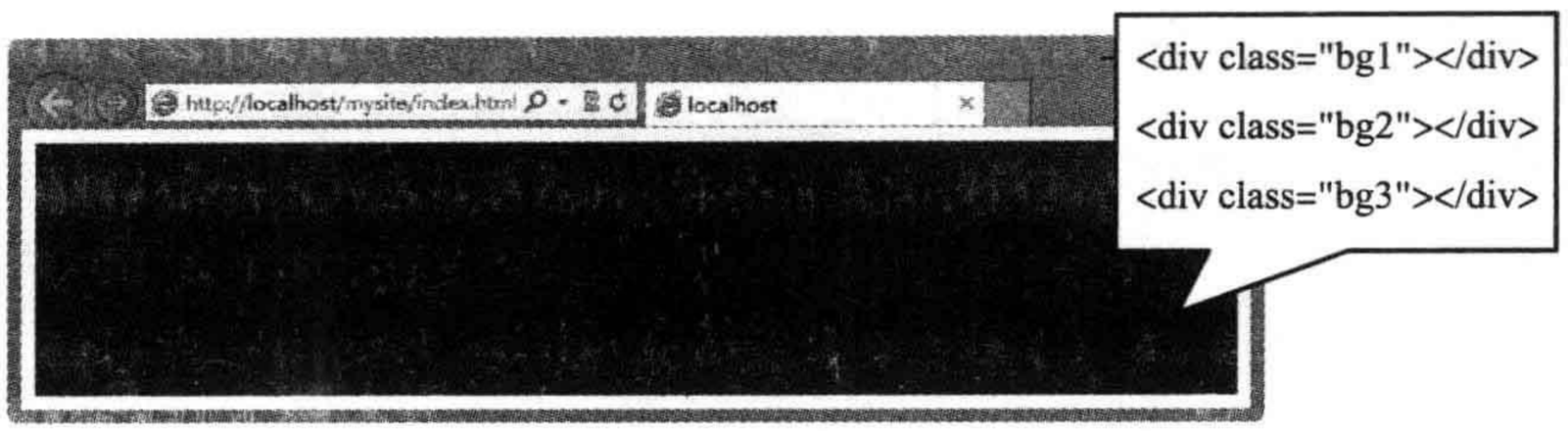


图 5-6　设计渐变样式

5.4.2　混合

有时候需要创建公共样式，然后在其他样式中重复利用，以此来模拟编程语言中的继承机制。当然，用户可以在一个 HTML 文档中引用多个 Class 样式类，来实现这种继承机制，但是使用 LESS，这些操作在样式表中就可以快速实现。

例如，定义一个样式类 border，然后在其他样式中调用该类，这样就可以在两个元素中分别添加类 .bordered，得到同样的效果，且仅仅在样式表中就完成了。

```
.border { border-top: 1px dotted #333;}
article.post {
      background: #eee;
       .border;
}
ul.menu {
      background: #ccc;
      .border;
}
```

输出：

```
article.post {
    background: #eee;
    border-top: 1px dotted #333;
}
ul.menu {
    background: #ccc;
    border-top: 1px dotted #333;
}
```

article.post 和 ul.menu 样式可以共享同一个边框样式，实现样式继承的功能，其中 .border 样式类相当于继承的类，而 article.post 和 ul.menu 样式相当于两个子类或实例。

5.4.3　参数混合

在 LESS 中，可以像声明函数一样定义一个带参数的样式，你可以把这个样式想象为一个 JavaScript 函数。例如，下面定义一个样式类 border-radius，通过小括号运算符包含一个参数变量，并把这个参数变量传递给样式中声明的属性。

```
.border-radius (@radius) {
    border-radius: @radius;
    -moz-border-radius: @radius;
    -webkit-border-radius: @radius;
}
```

然后，就可以在其他样式中调用这个类函数：

```
#header {
    .border-radius(4px);
}
.button {
    .border-radius(6px);
}
```

也可以给参数设置默认值，例如，设置参量 @radius 的默认值为 5px。

```
.border-radius (@radius: 5px) {
     border-radius: @radius;
     -moz-border-radius: @radius;
     -webkit-border-radius: @radius;
}
```

这样在调用该类函数时，可以不用传递参数，此时就会以默认值 5px 进行传递。

```
#header {
    .border-radius;
}
```

模仿 JavaScript 函数，可以定义不带参数的样式函数，如果想隐藏这个样式，不让它暴露到 CSS 中去，但是又想在其他样式中引用这个样式，会发现这个方法非常的好用。它与简单的混合方法没有什么区别。例如，定义一个文本换行处理的样式类，然后在不同样式中，直接调用这个样式函数就可以了。

```
.wrap () {
    text-wrap: wrap;
    white-space: pre-wrap;
    white-space: -moz-pre-wrap;
    word-wrap: break-word;
}
pre { .wrap }
```

输出：

```
pre {
     text-wrap: wrap;
     white-space: pre-wrap;
     white-space: -moz-pre-wrap;
     word-wrap: break-word;
}
```

参数混合对于那些 CSS 中看似多余的代码非常有用。最好、最有用的例子就是 CSS2 到 CSS3 中各浏览器的私有属性前缀，通过这种样式函数把各种私有属性都包装在一个标准样式类中，应用时直接引用这个标准样式类，而不用再考虑各种浏览器的私有属性。例如，上面的 .border-radius 和 .wrap 样式类就是一个很好的演示。.border-radius 有个默认的 5px 圆角，应用时可以使用任何圆角属性值，如 .border-radius（10px）将会生成半径为 10px 的圆角，此时不用再考虑为各个浏览器设置私有属性值。

需要注意的是，在函数型样式中，有一个特殊的默认变量 @arguments，它类似于 JavaScript 中的 Arguments，该变量包含了所有传递给样式函数的参数。如果不想单独处理每一个参数，就可以像这样写，使用 @arguments 进行快速传递。

例如，定义一个多参数样式函数，在这个样式中定义了盒子的阴影特效，共包含了 4 个参数，并分别设置了每个参数的默认值。在样式声明中，使用参数变量 @arguments 把这 4 个参数快速传递给每个属性。

```
.box-shadow (@x: 0, @y: 0, @blur: 1px, @color: #000) {
    box-shadow: @arguments;
    -moz-box-shadow: @arguments;
    -webkit-box-shadow: @arguments;
}
```

然后，在动态样式中定义三个类样式，分别调用这个样式函数：在第一个类样式中直接传递前面两个参数，后面两个保持默认值；第二个和第三个类样式分别传递 4 个参数，代码如下：

```
.box-shadow1 {.box-shadow(2px, 5px);}
.box-shadow2 {.box-shadow(5px, 5px, 5px, red);}
.box-shadow3 {.box-shadow(5px, -5px, 10px, green);}
```

最后，在 HTML 中为每个 标签应用一个类样式，则效果如图 5-7 所示。

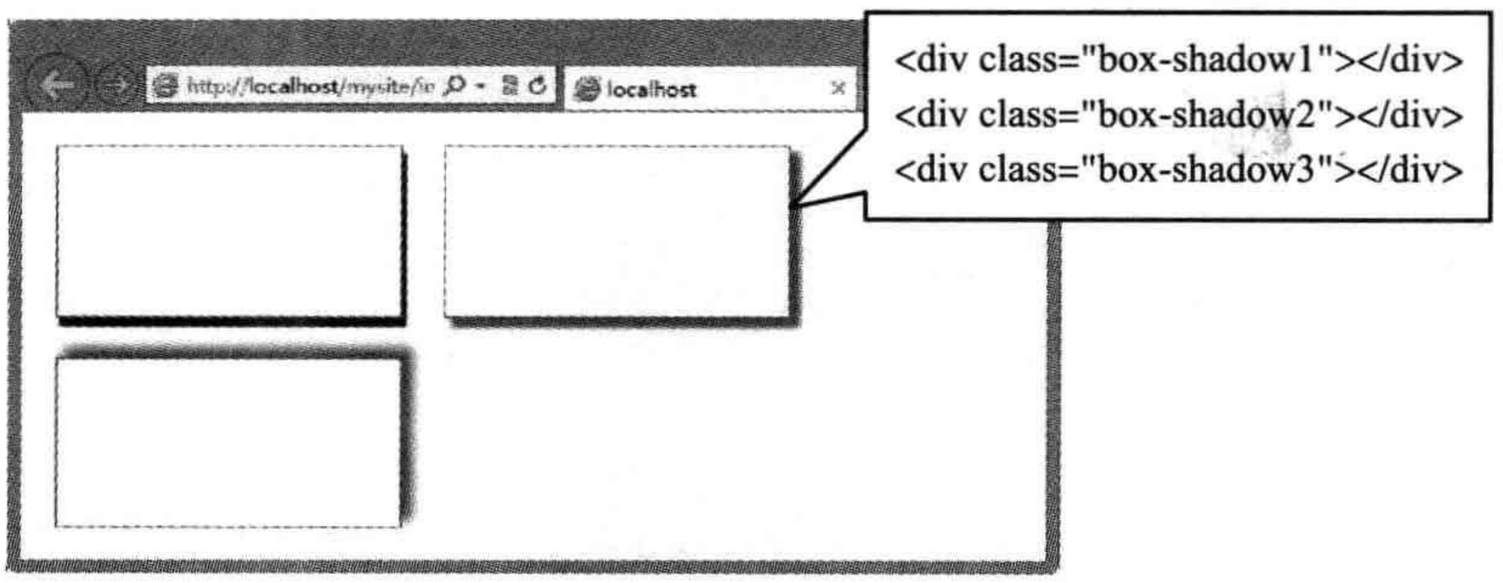


图 5-7　使用 @arguments 变量

5.4.4　模式匹配

上面两节详细讲解了 LESS 的混合（mixin）特性，使用它能够定义多个属性，然后轻松

地在多个样式中重用。甚至在定义混合时加入参数，使这些属性能根据调用参数不同而生成不同的属性。那么，让我们更进一步，来了解一下 LESS 对混合的更高级支持：模式匹配和条件表达式。

回顾一下 5.4.3 节介绍的带参数的混合方式：在混合中可以定义参数，同时也可以为这个参数指定一个默认值，在调用这个混合时如果指定了参数，LESS 就会用指定值来替换；如果不指定参数调用，就会用默认值替换。现在，考虑不仅仅通过参数值来更改最终结果，而是通过传入不同的参数或者参数个数来匹配不同的混合。

例如，下面的代码利用不同的参数个数来匹配不同的混合。

```
.mixin (@a) {
    color: @a;
    width: 10px;
}
.mixin (@a, @b) {
    color: fade(@a, @b);
}

.header{
    .mixin(red);
}
.footer{
    .mixin(blue, 50%);
}
```

使用不同的参数个数调用后，所生成的 CSS 代码如下：

```
.header {
    color: #ff0000;
    width: 10px;
}
.footer {
    color: rgba(0, 0, 255, 0.5);
}
```

这类似于 Java 语言中的方法调用，LESS 可以根据调用参数的个数来选择正确的混合来带入。上面示例中参数都是由变量构成的，其实在 LESS 中定义参数是可以使用常量的，此时模式匹配时匹配的方式也会发生相应的变化。例如，下面使用常量参数来控制混合的模式匹配。

```
.mixin (dark, @color) {
    color: darken(@color, 10%);
}
.mixin (light, @color) {
    color: lighten(@color, 10%);
}
.mixin (@zzz, @color) {
    display: block;
```

```
    weight: @zzz;
}
.header{
    .mixin(dark, red);
}
.footer{
    .mixin(light, blue);
}
.body{
    .mixin(none, blue);
}
```

通过常量参数生成的 CSS 代码如下：

```
.header {
    color: #cc0000;
    display: block;
    weight: dark;
}
.footer {
    color: #3333ff;
    display: block;
    weight: light;
}
.body {
    display: block;
    weight: none;
}
```

通过这个示例可以看出，当定义的是变量参数时，因为 LESS 中对变量并没有类型的概念，所以它只会根据参数的个数来选择相应的混合进行替换。而定义常量参数就不同了，这时候不仅参数的个数要匹配，而且常量参数的值和调用时的值也要一样才会匹配。值得注意的是，在 body 样式类中，调用时指定的第一个参数 none 并不能匹配上前两个混合，而第三个混合 .mixin（@zzz，@color）就不同了，由于它的两个参数都是变量，它接受任何值，因而它对三个调用都能匹配成功，于是在最终的 CSS 代码中看到每次调用的结果中都包含了第三个混合的属性。

最后，再来分析增加一个无参的混合和一个常量参数的混合，最终的匹配结果会发生什么变化。例如，在下面的代码中，分别设计无参和常量参数的模式匹配。

```
.border-radius (@radius: 3px) {
    border-radius: @radius;
    -moz-border-radius: @radius;
    -webkit-border-radius: @radius;
}
.border-radius (7px) {
    border-radius: 7px;
    -moz-border-radius: 7px;
}
```

```
.border-radius () {
     border-radius: 4px;
     -moz-border-radius: 4px;
     -webkit-border-radius: 4px;
}
.button {
     .border-radius(6px);
}
.button2{
     .border-radius(7px);
}
.button3{
    .border-radius();
}
```

加入了无参混合后生成的 CSS 代码如下：

```
.button {
     border-radius: 6px;
     -moz-border-radius: 6px;
     -webkit-border-radius: 6px;
     border-radius: 4px;
     -moz-border-radius: 4px;
     -webkit-border-radius: 4px;
}
.button2{
     border-radius: 7px;
     -moz-border-radius: 7px;
     -webkit-border-radius: 7px;
     border-radius: 7px;
     -moz-border-radius: 7px;
     border-radius: 4px;
     -moz-border-radius: 4px;
     -webkit-border-radius: 4px;
}
.button3{
     border-radius: 3px;
     -moz-border-radius: 3px;
     -webkit-border-radius: 3px;
     border-radius: 4px;
     -moz-border-radius: 4px;
     -webkit-border-radius: 4px;
}
```

生成的结果可能会出乎意料，无参的混合能够匹配任何调用，而常量参数非常严格，必须保证参数的值（7px）和调用的值（7px）一致才会匹配。

5.4.5　条件表达式

模式匹配提供了多项选择的设计思路，用户能根据不同的需求来匹配不同的混合，但更

进一步的就是利用条件表达式来更加准确、更加严格地限制混合的匹配，实现的方式就是利用 when 这个关键字。

当希望根据表达式进行匹配，而非根据值和参数匹配时，条件表达式就显得非常有用。如果读者熟悉函数式编程，那么就能够理解。为了尽可能保留 CSS 的可声明性，LESS 通过导引混合，而非 IF/Else 语句实现条件判断，因为前者已在 @Media Query 特性中被定义。

例如，下面代码使用 when 关键字定义两个混合。

利用条件表达式来控制模式匹配：

```
.mixin (@a) when (@a >= 10) {
    background-color: black;
}
.mixin (@a) when (@a < 10) {
    background-color: white;
}
.class1{ .mixin(12) }
.class2{ .mixin(6) }
```

条件表达式生成的 CSS 代码：

```
.class1{
     background-color: black;
}
.class2{
     background-color: white;
}
```

导引中可用的全部比较运算有 >、>=、=、=<、<。此外，关键字 true 只表示布尔真值，下面两个混合是相同的：

```
.truth (@a) when (@a) { }
.truth (@a) when (@a = true) { }
```

注意，除去关键字 true 以外的值都被视为布尔假：

```
.class {
    .truth(40); // 将不会匹配上面任何一个混合
}
```

导引序列使用逗号（,）分割，当且仅当所有条件都符合时，才会被视为匹配成功。

```
.mixin (@a) when (@a > 10), (@a < -10) { }
```

导引可以无参数，也可以对参数进行比较运算：

```
@media: mobile;
.mixin (@a) when (@media = mobile) {  }
.mixin (@a) when (@media = desktop) {  }
.max (@a, @b) when (@a > @b) { width: @a }
.max (@a, @b) when (@a < @b) { width: @b }
```

最后，如果想基于值的类型进行匹配，可以使用 is* 函数：

```
.mixin (@a, @b: 0) when (isnumber(@b)) { }
.mixin (@a, @b: black) when (iscolor(@b)) { }
```

下面就是常见的检测函数。

- iscolor：是否为颜色值。
- isnumber：是否为数值。
- isstring：是否为字符串。
- iskeyword：是否为关键字。
- isurl：是否为 URL 字符串。

例如，在条件表达式中支持的类型检查函数：

```
.mixin (@a) when (iscolor(@a)) {
    background-color: black;
}
.mixin (@a) when (isnumber(@a)) {
    background-color: white;
}
.class1{ .mixin(red) }
.class2{ .mixin(6) }
```

类型检查匹配后生成的 CSS 代码：

```
.class1{
    background-color: black;
}
.class2{
    background-color: white;
}
```

如果想判断一个值是纯数字还是某个单位量，可以使用下列函数。

- ispixel：是否为像素单位。
- ispercentage：是否为百分比。
- isem：是否为 em 单位。

另外，LESS 条件表达式支持 AND、OR 和 NOT 来组合条件表达式，这样可以组织成更为强大的条件表达式。注意，OR 在 LESS 中并不是 or 关键字，而是用逗号（,）来表示 or 的逻辑关系。

```
.smaller(@a, @b) when (@a > @b) {
    background-color: black;
}
.math (@a) when (@a > 10) and (@a < 20) {
    background-color: red;
}
.math (@a) when (@a < 10),(@a > 20) {
```

```
    background-color: blue;
}
.math (@a) when not (@a = 10)  {
    background-color: yellow;
}
.math (@a) when (@a = 10)  {
    background-color: green;
}
.testSmall {.smaller(30, 10) }
.testMath1{.math(15)}
.testMath2{.math(7)}
.testMath3{.math(10)}
```

生成的 CSS 代码如下：

```
.testSmall {
    background-color: black;
}
.testMath1{
    background-color: red;
    background-color: yellow;
}
.testMath2{
    background-color: blue;
    background-color: yellow;
}
.testMath3{
    background-color: green;
}
```

5.4.6 嵌套规则

在 CSS 中，复合选择器中的 Class 和 ID 嵌套是避免样式干扰或者被别的样式干扰的唯一方式。但是这种复合选择器会变得很乱。使用类似于 "#site-body. post. post-header h2" 的选择器毫无用处，而且还会占用大量不必要的空间。使用 LESS，可以嵌套 ID、Class 以及标签等选择符。例如，针对 "#site-body. post. post-header h2" 选择器，可以使用 LESS 的嵌套规则进行优化：

```
#site-body { …
    .post { …
        .post-header { …
            h2 { … }
            a { …
                &:visited { … }
                &:hover { … }
            }
        }
    }
}
```

上面的代码最终效果和上面的一大串选择器的效果一样，但是更容易阅读和理解，而且它占用很少的空间。也可以通过 & 关键字来引用标签样式到自己的伪元素上，这个功能类似于 JavaScript 函数中的 this 关键字。

LESS 允许以嵌套的方式编写层叠样式，如下面的多个样式代码。

```
#header { color: black; }
#header .navigation {
    font-size: 12px;
}
#header .logo {
    width: 300px;
}
#header .logo:hover {
    text-decoration: none;
}
```

在 LESS 中可以这样写：

```
#header {
    color: black;
    .navigation {
        font-size: 12px;
    }
    .logo {
        width: 300px;
        &:hover { text-decoration: none }
    }
}
```

或者这样写：

```
#header            { color: black;
    .navigation    { font-size: 12px }
    .logo          { width: 300px;
        &:hover    { text-decoration: none }
    }
}
```

代码更简洁了，而且与 HTML 文档结构保持一致。

注意 & 符号的使用。如果想写串联选择器，而不是写后代选择器，就可以用到 &，这对伪类选择器 (如 :hover 和 :focus) 尤其有用。例如：

```
.bordered {
    &.float {
        float: left;
    }
    .top {
        margin: 5px;
    }
}
```

转换为 CSS 如下：

```
.bordered.float {
     float: left;
}
.bordered .top {
     margin: 5px;
}
```

5.4.7 运算

LESS 允许任何数字、颜色或者变量参与运算，例如：

```
@base: 5%;
@filler: @base * 2;
@other: @base + @filler;
color: #888 / 4;
background-color: @base-color + #111;
height: 100% / 2 + @filler;
```

LESS 运算能够自动分辨出颜色和单位。例如，单位运算

```
@var: 1px + 5;
```

会输出 6px。

LESS 也允许使用括号，例如：

```
width: (@var + 5) * 2;
```

且可以在复合属性中进行运算：

```
border: (@width * 2) solid black;
```

注意，LESS 运算符、运算法则和运算顺序完全遵循 JavaScript 语言规则。

5.4.8 颜色函数

LESS 提供了一系列的颜色运算函数，在运算时颜色会先被转化成 HSL 色彩空间，然后在通道级别进行操作。颜色函数说明如表 5-1 所示。

表 5-1 常用颜色函数

颜色函数	说 明
lighten(@color,10%)	返回颜色比 @color 颜色亮 10%
darken(@color,10%);	返回颜色比 @color 颜色暗 10%
saturate(@color,10%);	返回颜色比 @color 颜色饱和度高 10%
desaturate(@color,10%);	返回颜色比 @color 颜色饱和度低 10%
fadein(@color,10%);	返回颜色比 @color 颜色不透明度高 10%

（续）

颜色函数	说　明
fadeout(@color,10%);	返回颜色比 @color 颜色不透明度低 10%
fade(@color,50%);	返回颜色是 @color 颜色透明度的 50%
spin(@color,10);	返回颜色在 @color 颜色基础上增加 10 度色调
spin(@color,-10);	返回颜色在 @color 颜色基础上减少 10 度色调
mix(@color1,@color2);	返回 @color1 和 @color2 的混合色

颜色函数使用起来比较简单。例如，下面的代码定义一个颜色变量，并赋值为 #f04615，然后在类样式中，调用 saturate() 函数，把 #f04615 的饱和度提高 5%，然后再设置字体颜色。使用 spin() 函数增加 #f04615 色调 8 度，再使用 lighten() 函数把颜色亮度降低 25%。

```
@base: #f04615;
.class {
      color: saturate(@base, 5%);
      background-color: lighten(spin(@base, 8), 25%);
}
```

用户也可以提取颜色信息，具体方法说明如表 5-2 所示。

表 5-2　提取颜色信息函数

颜色函数	说　明
hue(@color);	提取颜色 @color 的色调值
saturation(@color);	提取颜色 @color 的饱和度值
lightness(@color);	提取颜色 @color 的亮度值

如果想在一种颜色的通道上创建另一种颜色，这些函数就显得非常好用。例如，下面的代码 @new 将会保持 @old 的色调，但是具有不同的饱和度和亮度。

```
@new: hsl(hue(@old), 45%, 90%);
```

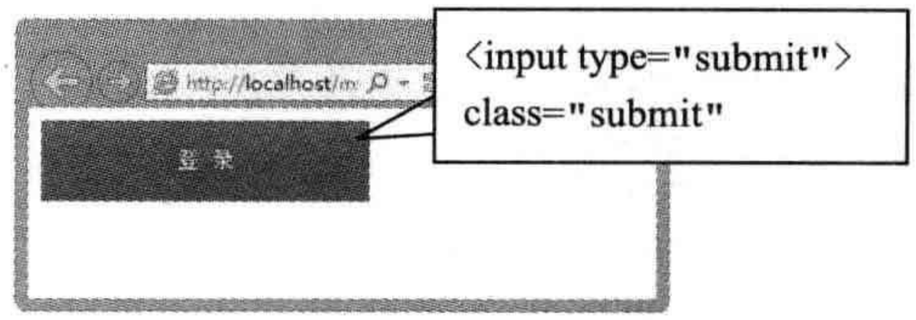


图 5-8　使用颜色函数设计渐变效果按钮

例如，下面的示例设计一个调色板。首先在样式中使用一个标准的蓝色风格，然后使用这个颜色在一个表单中制作一个渐变的提交按钮，演示效果如图 5-8 所示。

具体实现代码如下：

```
@blue: #369;
.submit {
    padding: 5px 10px;
    border: 1px solid @blue;
    background: -moz-linear-gradient(top, lighten(@blue, 10%), @blue 100%); /*Moz*/
    background: -webkit-gradient(linear, center top, center bottom, from(lighten(@
blue, 10%)), color-stop(100%, @blue)); /*Webkit*/
```

```
    background: -o-linear-gradient(top, lighten(@blue, 10%) 0%, @blue 100%); /*Opera*/
    background: -ms-linear-gradient(top, lighten(@blue, 10%) 0%, @blue 100%); /*IE 10+*/
    background: linear-gradient(top, lighten(@blue, 10%) 0%, @blue 100%); /*W3C*/
    color: #fff;
    text-shadow: 0 -1px 1px rgba(0,0,0,0.4);
    height:50px;
    width:200px;
}
```

lighten 函数很明显就是用百分比值来减轻颜色，在上面的代码中，它将在蓝色的基础上减少 10%。这种方法只需要简单地改变基础颜色就可以修改渐变元素或者其他元素的颜色。这对于制作主题模板来说是非常有用的。而且，如果使用参数函数，还可以更简单地应用到一些声明中，设计更精致的色彩效果，如 .linear-gradient(lighten(@blue), @blue, 100%),。

5.4.9 数学函数

LESS 提供了一组数学函数，使用它们可以很方便地处理一些数字类型的值计算，说明如表 5-3 所示。

表 5-3 常用数学函数

数学函数	说 明
round(@number)	对数值变量 @number 进行四舍五入计算
ceil(@number)	对数值变量 @number 进行上舍入计算
floor(@number)	对数值变量 @number 进行下舍入计算
percentage(@number)	对数值变量 @number 进行百分比转换

例如：

```
round(1.67);      // 返回 2
ceil(2.4);        //返回 3
floor(2.6);       //返回 2
percentage(0.5);  //返回 50%
```

5.4.10 作用域

LESS 变量、混合等特性其实在很大程度上避免了传统 CSS 中的大量代码冗余。变量能够避免一个属性多次重复，混合能够避免属性集的重复。而且变量和混合使用起来更加灵活，维护起来也方便了许多，只要修改一处定义，而无需修改每个引用的地方。现在，让我们更进一步，当定义好了变量和混合之后，怎么能更好地控制和运用它们呢，怎么避免与其他地方定义的变量和混合产生冲突？一个显而易见的想法是，像其他语言一样引入命名空间和作用域。

作用域是程序中的一个标准，LESS 中也是。在样式表的 root 级声明的变量在整个文档中都是可以调用的。然而，如果在一个选择器（如 ID 或 Class）中重新定义了这个变量，那

么它就只能在这个选择器中用了，而且取重新定义后的新值。

首先，来看一个 LESS 变量作用域的示例：

```
@var: red;
#page {
    @var: white;
    #header {
        color: @var;
    }
}
#footer {
    color: @var;
}
```

在上面的示例中可以看到，header 中的 @var 会首先在当前作用域寻找，然后再逐层往父作用域中寻找，一直到顶层的全局作用域中为止。所以，header 的 @var 在父作用域中找到之后就停止了寻找，最终的值为 white。而 footer 中的 @var 在当前作用域没找到定义之后就寻找到了全局作用域，最终的结果是全局作用域中的定义值 red。

上面的代码生成如下 CSS 代码：

```
#page #header {
    color: #ffffff;  // white
}
#footer {
    color: #ff0000;  // red
}
```

5.4.11　命名空间

了解了作用域之后，再来了解命名空间。我们可以用命名空间把变量和混合封装起来，避免和其他地方的定义冲突，引用起来也十分方便，只要在前面加上相应的命名空间就可以了。例如：

```
@var-color: white;
#bundle {
    @var-color: black;
    .button () {
        display: block;
        border: 1pxsolidblack;
        background-color: @var-color;
    }
    .tab() { color: red}
    .citation() { color: black}
    .oops {weight: 10px}
}
#header {
    color: @var-color;
```

```
        #bundle > .button;
        #bundle > .oops;
    }
```

这里可以看出，利用嵌套规则在 #bundle 中建立了一个命名空间，在里面封装的变量以及属性集合都不会暴露到外部空间中。例如 .tab()、.citation() 都没有暴露在最终的 CSS 代码中。

值得注意的是，.oops 被暴露在了最终的 CSS 代码中，这种结果可能并不是我们想要的。其实同样的例子可以在混合示例中发现，即无参的混合 .tab() 是和普通的属性集 .oops 不同的。无参的混合不会暴露在最终的 CSS 代码中，而普通的属性集则会出现。在定义命名空间和混合时，要小心处理这样的差别，避免带来潜在的问题。

上面的命名空间示例最后生成的 CSS 代码如下：

```
#bundle .oops {
    weight: 10px;
}
#header {
    color: #ffffff;
    display: block;
    border: 1pxsolidblack;
    background-color: #000000;
    weight: 10px;
}
```

有时候，为了更好地组织 CSS，或者为了更好地封装，将一些变量或者混合模块打包起来。例如，在下面的代码中，在 #bundle 中定义一些属性集之后就可以重复使用。

```
#bundle {
    .button () {
        display: block;
        border: 1px solid black;
        background-color: grey;
        &:hover { background-color: white }
    }
}
```

这样只需要在 #header a 中按如下方式引入 .button：

```
#header a {
    color: orange;
    #bundle > .button;
}
```

5.4.12 注释

LESS 允许以下两种注释写法。

（1）标准的 CSS 注释

```
/* comment */
.class { color: black }
```

这种方法是有效的，而且能够通过处理并正确输出。

（2）行注释

LESS 支持 C 语法中的双斜线注释，但是编译成 CSS 的时候会自动过滤掉。

```
// comment
.class { color: white }
```

这种方法可以使用，但是不能够通过处理，也不能输出到最终的 CSS 样式表中。

5.4.13　导入

LESS 支持符合标准的导入方法，利用 @import 命令可以导入外部 LESS 文件或 CSS 文件。

例如，用户可以在 main.less 文件中通过下面方法导入外部 .less 文件，.less 后缀可带可不带：

```
@import "lib.less";
@import "lib";
```

如果想导入一个 CSS 文件，而且不希望 LESS 对它进行编译，只需使用 .css 后缀：

```
@import "lib.css";
```

这样 LESS 就会跳过，不进行任何处理。

5.4.14　字符串插值

变量可以使用类似 Ruby 和 PHP 的方式嵌入到字符串中，方法是使用 @{name} 结构，通过 @{name} 来调用变量的值。例如：

```
@base_url = 'http://coding.smashingmagazine.com';
background-image: url("@{base_url}/images/background.png");
```

5.4.15　转义字符

有时候，当需要引入无效的 CSS 语法或者 LESS 不能识别的字符时（通常是一些 IE 的 Hack）。要避免 LESS 抛出异常，并避免 LESS 编译。此时，可以在字符串前加上一个 ~，并将需要转义的字符串用 "" 包含起来，表示避免编译该行字符串，例如：

```
.class {
    filter: ~"progid:DXImageTransform.Microsoft.Alpha(opacity=20)";
}
```

最后输出的 CSS 代码如下：

```
.class {
     filter: progid:DXImageTransform.Microsoft.Alpha(opacity=20);
}
```

5.4.16 JavaScript 表达式

JavaScript 表达式可以在 LESS 文件中使用，通过反引号的方式使用：

```
@var: `"hello".toUpperCase() + '!'`;
```

输出：

```
@var: "HELLO!";
```

注意，也可以同时使用字符串插值和避免编译：

```
@str: "hello";
@var: ~'"@{str}".toUpperCase() + '!'';
```

输出：

```
@var: HELLO!;
```


也可以访问 JavaScript 环境：

```
@height: 'document.body.clientHeight';
```

如果想将一个 JavaScript 字符串解析成十六进制的颜色值，可以使用颜色函数：

```
@color: color('window.colors.baseColor');
@darkcolor: darken(@color, 10%);
```

在 CSS 中使用 JavaScript 表达式无疑是十分令人兴奋的。在下面的示例中通过使用 JavaScript 表达式，实现各种复杂的运算，同时可以看到 JavaScript 表达式可以运用于数组操作当中。其实 LESS 的 JavaScript 表达式还支持其他一些方式，不过目前尚未公布出来。

```
.eval {
   js: '1+ 1';
   js: '(1+ 1== 2? true : false)';
   js: '"hello".toUpperCase() + '!'';
   title: 'process.title';
}
.scope {
    @foo: 42;
    var: 'this.foo.toJS()';
}
.escape-interpol {
    @world: "world";
    width: ~'"hello"+ " "+ @{world}';
```

```
}
.arrays {
        @ary:  1, 2, 3;
        @ary2:1  2  3;
        ary: '@{ary}.join(', ')';
        ary: '@{ary2}.join(', ')';
}
```

上面的 JavaScript 表达式生成的 CSS 代码如下：

```
.eval {
     js: 2;
     js: true;
     js: "HELLO!";
     title: "/Users/Admin/Downloads/LESS/Less.app/Contents/Resources/engines/bin/node";
}
.scope {
     var: 42;
}
.escape-interpol {
     width: hello world;
}
.arrays {
     ary: "1, 2, 3";
}
```

在 eval 中可以使用 JavaScript 数字运算、布尔表达式，对字符串进行大小写转换、串联字符串等操作。甚至最后能够获取到 JavaScript 的运行环境。同样可以看到 LESS 的作用域和变量也在 JavaScript 表达式中使用。

5.5　Bootstrap 与 LESS 结合

LESS 比较适合于重用性比较高的 Web 开发，或者网页设计代码，如果是换肤的功能，使用 LESS 再合适不过了。当然也有很多工程师喜欢 LESS 的代码风格。因此，它的可维护性、可扩展性和易读性都会是广大用户选择它的原因。

众所周知，LESS 是一种动态样式语言，Bootstrap 基于 LESS 进行编写，主要是为了继承 LESS 的优势，如变量、继承、运算和函数。

5.5.1　基于 LESS 的 Bootstrap

Bootstrap 的核心是基于 LESS 框架构建的 CSS，因此学习 Bootstrap 就必须关注 LESS。

LESS 是一个动态 CSS 语言框架，LESS 扩展了 CSS 的动态特性，相对于传统的 CSS，LESS 提供了更为强大的功能和灵活性。基于 LESS，我们可以在编写 CSS 时使用嵌入式申明、变量、混合模式、运算和颜色编辑功能函数等。无论是在灵活性、扩展性还是在可维护性上，LESS 都让 CSS 开发有了大幅提升，动态性增强了。

使用了 LESS 的 Bootstrap 具备如下优点：

- Bootstrap 实现起来依旧很简单，使用也很简单，把 Bootstrap.css 拖入页面中即可。编译 LESS 文件可以使用 less.js、Less.app 或 Node.js 等多种方案实现。
- 一旦编译，Bootstrap 框架仅包含 CSS 文件，这意味着没有多余的图片、Flash 或 JavaScript，只有用于 Web 应用程序开发的简洁而强大的 CSS 样式。

5.5.2 Bootstrap 变量

Bootstrap 变量全部部署在 variables.less 文件中（bootstrap-master/less），另外如果在网上定制下载 Bootstrap，可以在官网直接配置这些变量，下载之后的 Bootstrap 组件动态样式将被编译为 CSS 源代码，如图 5-9 所示。

图 5-9 定制变量

下面分类介绍一下 Bootstrap 变量。

1. 基础设置

页面基本属性设置主要包括页面基本前景色和背景色，以及超链接默认颜色，如表 5-4 所示。

表 5-4 页面基本属性

变量名称	默 认 值	说 明
@bodyBackground	@white	页面背景色，默认值为白色
@textColor	@grayDark	默认的文字颜色，默认值为深灰色
@linkColor	#08c	默认的链接颜色，默认值为浅蓝色
@linkColorHover	darken(@linkColor, 15%)	默认链接 hover 样式，默认值为深蓝色

2. 栅格设置

页面栅格基本属性设置主要包括栅格数、栅格宽度、间距，以及流动栅格宽度和间距，如表 5-5 所示。

表 5-5　栅格属性

变量名称	默认值	说　明
@gridColumns	12	网页默认栅格数
@gridColumnWidth	60px	默认每个栅格的宽度
@gridGutterWidth	20px	默认栅格之间的距离
@fluidGridColumnWidth	6.382978723%	默认流动布局中每个栅格的宽度
@fluidGridGutterWidth	2.127659574%	默认流动布局中栅格之间的距离

3. 字体设置

页面字体基本属性设置主要包括字体类型、字体大小、行高、标题字体样式等，如表 5-6 所示。

表 5-6　字体属性

变量名称	默认值	说　明
@sansFontFamily	"Helvetica Neue", Helvetica, Arial, sans-serif	Sans 类型字体默认集合
@serifFontFamily	Georgia, "Times New Roman", Times,serif	Serif 类型字体默认集合
@monoFontFamily	Menlo, Monaco, "Courier New", monospace	Mono 类型字体默认
@baseFontSize	13px	网页字体默认大小，以像素为单位
@baseFontFamily	@sansFontFamily	网页默认字体类型
@baseLineHeight	18px	默认行高，以像素为单位
@altFontFamily	@serifFontFamily	Alt 提示字体类型，在 3.0 版本中将被移除
@headingsFontFamily	inherit	标题字体类型，继承浏览器默认字体类型
@headingsFontWeight	bold	标题字体样式，默认为加粗显示
@headingsColor	inherit	标题字体颜色，继承浏览器默认字体颜色

4. 表格设置

设置表格背景色、边框颜色、表格鼠标交互背景色等，如表 5-7 所示。

表 5-7　表格属性

变量名称	默认值	说　明
@tableBackground	transparent	表格默认背景色，默认值为透明
@tableBackgroundAccent	#f9f9f9	表格默认强调背景色，默认为浅白色
@tableBackgroundHover	#f5f5f5	表格默认鼠标经过行背景色，默认为浅灰色
@tableBorder	#ddd	表格边框颜色，默认为较浅灰色

5. 冷色调设置

统一设计常用冷色调色系标准值，如表 5-8 所示。

表 5-8 冷色调属性

变量名称	默认值	说明
@black	#000	黑色
@grayDarker	#222	较深灰色
@grayDark	#333	深灰色
@gray	#555	灰色
@grayLight	#999	浅灰色
@grayLighter	#eee	较浅灰色
@white	#fff	白色

6. 暖色调设置

统一设计常用暖色调色系标准值，如表 5-9 所示。

表 5-9 暖色调属性

变量名称	默认值	说明
@blue	#049cdb	蓝色
@green	#46a546	绿色
@red	#9d261d	红色
@yellow	#ffc40d	黄色
@orange	#f89406	橙色
@pink	#c3325f	粉色
@purple	#7a43b6	紫色

7. 按钮设置

统一设计按钮组件的默认样式，如表 5-10 所示。

表 5-10 按钮样式属性

变量名称	默认值	说明
@btnBackground	@white	按钮背景色，白色
@btnBackgroundHighlight	darken(@white,10%)	按钮高亮背景色，白色加深 10%，浅灰色
@btnBorder	darken(@white,20%)	按钮边框色，白色加深 20%，灰色
@btnPrimaryBackground	@linkColor	重要按钮背景色，亮蓝色
@btnPrimaryBackgroundHighlight	spin(@btnPrimaryBackground,15%)	重要按钮高亮背景色，深蓝色
@btnInfoBackground	#5bc0de	信息按钮背景色，浅蓝色
@btnInfoBackgroundHighlight	#2f96b4	信息按钮高亮背景色，灰蓝色
@btnSuccessBackground	#62c462	成功按钮背景色，亮绿色

（续）

变量名称	默 认 值	说　明
@btnSuccessBackgroundHighlight	#51a351	成功按钮高亮背景色，深绿色
@btnWarningBackground	lighten(@orange,15%)	警告按钮背景色，浅橙色
@btnWarningBackgroundHighlight	@orange	警告按钮高亮背景色，橙色
@btnDangerBackground	#ee5f5b	危险按钮背景色，浅红色
@btnDangerBackgroundHighlight	#bd362f	危险按钮高亮背景色，深红色
@btnInverseBackground	@gray	逆反按钮背景色，灰色
@btnInverseBackgroundHighlight	@grayDarker	逆反按钮高亮背景色，深灰色

8. 表单设置

统一设计表单组件的默认样式，如表 5-11 所示。

表 5-11　表单样式属性

变量名称	默 认 值	说　明
@placeholderText	@grayLight	文本框默认文本颜色，浅灰色
@inputBackground	@white	输入表单背景色，白色
@inputBorder	#ccc	输入表单边框色，浅灰色
@inputBorderRadius	3px	输入表单控件圆角弧度，3 像素
@inputDisabledBackground	@grayLighter	无效表单控件背景色，较浅灰色
@formActionsBackground	#f5f5f5	激活表单控件背景色，浅白色

表单提示信息背景及文字颜色设置表，如表 5-12 所示。

表 5-12　表单提示背景及文字颜色

变量名称	默 认 值	说　明
@warningText	#c09853	警告文本颜色，土黄色
@warningBackground	#f3edd2	警告文本背景色，浅黄色
@errorText	#b94a48	错误文本颜色，土红色
@errorBackground	#f2dede	错误文本背景色，浅红色
@successText	#468847	成功文本颜色，墨绿色
@successBackground	#dff0d8	成功文本背景色，浅绿色
@infoText	#3a87ad	信息文本颜色，浅蓝色
@infoBackground	#d9edf7	信息文本背景色，蓝白色

9. 导航栏设置

统一设计导航栏的默认样式，如表 5-13 所示。

表 5-13 导航栏样式属性

变量名称	默认值	说明
@navbarHeight	40px	导航栏默认高度，40 像素
@navbarBackground	@grayDarker	导航栏背景色，较深灰色
@navbarBackgroundHighlight	@grayDark	导航栏高亮背景色，深灰色
@navbarText	@grayLight	导航栏文本颜色，浅灰色
@navbarLinkColor	@grayLight	导航栏链接文本颜色，浅灰色
@navbarLinkColorHover	@white	鼠标经过导航栏链接文本颜色，白色
@navbarLinkColorActive	@navbarLinkColorHover	导航链接文本被激活时文本颜色，白色
@navbarLinkBackgroundHover	transparent	鼠标经过导航栏链接背景颜色，透明
@navbarLinkBackgroundActive	@navbarBackground	导航文本链接文本背景色，透明
@navbarSearchBackground	lighten(@navbarBackground, 25%)	导航栏搜索文本框背景色，在导航背景色基础上变亮 25%
@navbarSearchBackgroundFocus	@white	导航栏搜索文本框获取焦点时背景色，白色
@navbarSearchBorder	darken(@navbarSearchBackground, 30%)	导航栏搜索文本框边框颜色，在背景色基础加深 30%
@navbarSearchPlaceholderColor	#ccc	导航栏搜索文本框默认显示文本颜色，浅灰色
@navbarBrandColor	@navbarLinkColor	导航栏标识文本颜色，同链接文本颜色

10. 下拉设置

统一设计下拉组件的默认样式，如表 5-14 所示。

表 5-14 下拉样式属性

变量名称	默认值	说明
@dropdownBackground	@white	下拉背景色，白色
@dropdownBorder	rgba(0,0,0,.2)	下拉边框颜色，不透明度为 0.2 的黑色
@dropdownLinkColor	@grayDark	下拉链接文本颜色，深灰色
@dropdownLinkColorHover	@white	下拉链接经过时文本颜色，白色
@dropdownLinkBackgroundHover	@linkColor	下拉链接经过时背景色，同链接颜色

11. Hero 区域设置

统一设计 Hero 区域的默认样式，如表 5-15 所示。

表 5-15 Hero 样式属性

变量名称	默认值	说明
@heroUnitBackground	@grayLighter	Hero 区域背景色，较浅灰色
@heroUnitHeadingColor	inherit	Hero 区域标题文本颜色，继承
@heroUnitLeadColor	inherit	Hero 区域导航文本颜色，继承

5.5.3　Bootstrap 混合

Bootstrap 混合主要是将一段需要进行合并的样式通过 LESS 的一种声明方式写到一起，它可以方便地被其他样式调用，从而达到可重用的目的，例如：

```
.element {
    .clearfix();
}
```

也可以设计带参数的混合，这种混合和基础混合比较类似，它增加接受参数的功能，当然如果不传任何参数，它会提供一个默认值，例如：

```
.element {
    .border-radius(4px);
}
```

在 Bootstrap 中，所有的混合都存储在 mixins.less 文件中，如果需要增加混合可以直接集成到 utilities.less 中，方便调用。

Bootstrap 包含的混合说明如下。

1. 常用混合

汇总在布局设计中常用的混合用法，如表 5-16 所示。

表 5-16　常用混合说明

混合名称	参 数 值	用　途
.clearfix()	none	清除浮动
.tab-focus()	none	自动聚焦
.center-block()	none	居中，相当于 margin: auto
ie7-inline-block()	none	让 IE6、IE7 支持 display: inline-block
.size()	@height @width	设置容器宽高
.square()	@size	设置该容器为正方形，参数为边长
.opacity()	@opacity	设置容器透明度

2. 表单混合

表单混合如表 5-17 所示。

表 5-17　表单混合说明

混合名称	参 数 值	用　途
.placeholder()	@color: @placeholderText	设置输入框的默认文本颜色和默认文本内容

3. 字体混合

字体混合如表 5-18 所示。

表 5-18 字体混合说明

混合名称	参 数 值	用 途
#font>#family>.serif()	none	设置这个元素的字体为有衬线字体
#font>#family>.sans-serif()	none	设置这个元素的字体为无衬线字体
#font>#family>.monospace()	none	设置这个元素的字体为等宽字体
#font>.shorthand()	@size:@baseFontSize, @weight:normal,@lineHeight: @baseLineHeight	简单地设置字体的大小粗细等
#font>.serif()	@size:@baseFontSize, @weight:normal,@lineHeight: @baseLineHeight	设置该字体为有衬线字体，并设置字体的大小粗细等
#font>.sans-serif()	@size:@baseFontSize, @weight:normal,@lineHeight: @baseLineHeight	设置该字体为无衬线字体，并设置字体的大小粗细等
#font>.monospace()	@size:@baseFontSize, @weight:normal,@lineHeight: @baseLineHeight	设置该字体为等宽字体，并设置字体的大小粗细等

4. 栅格系统

相关的 LESS 代码在 mixins.less 中，具体说明如表 5-19 所示。

表 5-19 栅格系统

混合名称	参 数 值	用 途
.container-fixed()	none	指定该容器为居中
#grid>.core()	@gridColumnWidth, @gridGutterWidth	初始化栅格系统，参数传递分别为栅格的列宽和栅格之间的距离
#grid>.fluid()	@fluidGridColumnWidth, @fluidGridGutterWidth	初始化栅格系统，参数传递分别为每个栅格的所占栅格总宽度的百分比和栅格之间的距离所占栅格总宽度的百分比
#grid>.input()	@gridColumnWidth, @gridGutterWidth	生成 input 相关元素的栅格布局，参数传递分别为栅格的列宽和栅格之间的距离
.makeColumn	@columns:1, @offset:0	在栅格系统中初始化一个占几列的 div 容器，columns 为该容器跨域的列数，offset 为该容器的左偏移

5. CSS3 属性

针对 CSS3 属性中存在的兼容性问题，提供了完整的兼容性解决方案，具体说明如表 5-20 所示。

表 5-20　CSS3 属性

混合名称	参　数　值	用　　途
.border-radius()	@radius	CSS3 圆角，参数为圆角像素
.box-shadow()	@shadow	CSS3 阴影
.transition()	@transition	CSS3 动画（如 all.2s linear）
.rotate()	@degrees	旋转一个元素，参数为旋转的度数
.scale()	@ratio	缩放元素，参数为缩放后和元尺寸的比例
.translate()	@x, @y	在平面上移动元素，参数分别对应相对于 x 轴和 y 轴的移动距离
.background-clip()	@clip	背景裁剪，传入 clip，clip 选择 border|padding|content
.background-size()	@size	通过 CSS3 来控制背景图片的尺寸
.box-sizing()	@boxmodel	改变容器的盒模型，例如我们可以改变类似 input button 的盒模型为传统的 IE 模型，即设置第一个参数为 border-box 就可以达到所有浏览器兼容的目的
.user-select()	@select	用来控制内容的可选择性
.backface-visibility()	@visibility: visible	CSS3 动画效果是否隐藏内容的背面
.resizable()	@direction: both	让元素可以进行向右和向下的拉伸缩放
.content-columns()	@columnCount, @columnGap: @gridGutterWidth	让容器具有 CSS3 的属性 content-count 和 column-gap，第一个参数为列数，第二个参数为列数之间的间距

6. 背景和渐变

下面是 Bootstrap 为背景和渐变提供的完整的兼容性解决方案，具体说明如表 5-21 所示。

表 5-21　背景和渐变混合

混合名称	参　数　值	用　　途
#translucent>.background()	@color: @white, @alpha:1	给元素半透明的背景色，第一个参数为背景色，第二个参数为透明度
#translucent>.border()	@color: @white, @alpha:1	给元素半透明的边框色，第一个参数为边框颜色，第二个参数为透明度
#gradient>.vertical()	@startColor, @endColor	让一个容器自上而下颜色渐变，兼容任何浏览器，第一个参数是开始的颜色，第二个参数是结束的颜色
#gradient>.horizontal()	@startColor, @endColor	让一个容器自左而右颜色渐变，兼容任何浏览器，第一个参数是开始的颜色，第二个参数是结束的颜色
#gradient>.directional()	@startColor, @endColor, @deg	让一个容器按照一个角度进行渐变，第一个参数是开始的颜色，第二个参数是结束的颜色，第三个参数是进行渐变的角度

（续）

混合名称	参 数 值	用 途
#gradient>.vertical-three-colors()	@startColor, @midColor, @colorStop, @endColor	让一个容器按照开始颜色和中间颜色以及结束颜色进行渐变，第一个参数是开始的颜色，第二个参数是中间的颜色，第三个参数是中间渐变结束的位置，最后一个参数是渐变结束的颜色
#gradient>.radial()	@innerColor, @outerColor	让一个容器放射性渐变，第一个参数是容器中心的颜色，第二个参数是容器最外层的颜色
#gradient>.striped()	@color, @angle	条纹渐变，第一个参数为渐变的颜色，第二个参数为渐变的角度
#gradientBar()	@primaryColor, @secondaryColor	主要用于按钮和提示框的垂直颜色渐变

第6章

使用 Bootstrap 组件

本章内容

- 下拉菜单
- 按钮组
- 按钮式下拉菜单
- 导航
- 导航条
- 面包屑和分页
- 标签与徽章
- 缩略图
- 警告框
- 进度条
- 媒体
- 版式
- 其他组件

Bootstrap 是基于 HTML、CSS 和 JavaScript 的简洁、灵活的流行前端框架及交互组件集，使用它开发人员能够轻松搭建出清爽风格的界面，实现良好的交互效果。Bootstrap 的口号是：Designed for everyone，everywhere（适用于任何场景，适用于每一个人）。

Bootstrap 作为完整的前端工具集，内建了大量的强大、优雅的可重用组件，包括按钮（Button）、导航（Navigation）、标签（Label）、徽章（Badge）、排版（Typography）、缩略图（thumbnail）、提醒（Alert）、进度条（progress bar）、杂项（Miscellaneous）。本章将详细介绍这些组件的结构和基本应用。

6.1 下拉菜单

下拉菜单是网页上最常见的效果之一，用鼠标轻轻一点或是移过去，就会出现一个更加详细的菜单。它不仅节省了网页排版空间、使网页布局简洁有序，而且一个新颖美观的下拉菜单更是为网页增色不少。

制作下拉菜单的方法多种多样。常规方法是 HTML+CSS+JavaScript 实现，也有 HTML+CSS 设计的纯 CSS 下拉菜单。Bootstrap 定义了一套完整的下拉菜单组件，配合其他元素可以设计形式多样的导航菜单效果。

6.1.1 快速体验下拉菜单

在系统介绍 Bootstrap 下拉菜单组件之前，我们先看一个完整示例的实现过程，以便获得一个总体的印象，快速掌握 Bootstrap 下拉菜单组件的设计。

第 1 步：新建 HTML5 文档。Bootstrap 使用的某些 HTML 元素和 CSS 属性需要文档类型为 HTML5 doctype。因此这一文档类型必须出现在项目每个页面的开始部分：

```
<!doctype html>
<html>
</html>
```

第 2 步：在页面头部区域引入下面的文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

- bootstrap.css：Bootstrap 样式库，这是必不可少的。
- jquery.js：引入 jQuery，Bootstrap 插件是 jQuery 插件，依赖于 jQuery 技术库。
- bootstrap-dropdown.js：下拉菜单插件。

第 3 步：设计下拉菜单 HTML 结构。

```
<div class="dropdown">
    <a href="#" class="btn btn-large">激活按钮 <i class="icon-arrow-down"></i></a>
    <ul class="dropdown-menu">
      <li><a href="#">菜单项 1</a></li>
```

```
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
</div>
```

上面的代码创建了一个下拉菜单，其中包括一个激活元件 <a> 标签，以及多个下拉菜单列表项。在下拉包含框中，引入 dropdown 类，定义当前框为下拉菜单框。然后在下拉列表框中引入 dropdown-menu 类，定义下拉菜单条面板。

第 4 步：上面的代码只是定义了下拉菜单的样式，要想使其真正成为下拉菜单，还必须激活下拉菜单，这有两种方法：定义 data 属性，或者使用 JavaScript 脚本直接调用。

方法一，使用 data 属性触发。只需要在激活元件上设置 data-toggle="dropdown" 即可，代码如下：

```
<div class="dropdown">
    <a href="#" class="btn btn-large" data-toggle="dropdown" >激活按钮 <i class="icon-
       arrow-down"></i></a>
    <ul class="dropdown-menu">
       <li><a href="#">菜单项 1</a></li>
       <li><a href="#">菜单项 2</a></li>
       <li><a href="#">菜单项 3</a></li>
    </ul>
</div>
```

此时，在浏览器中预览，可以看到仅显示按钮，单击按钮即可显示下拉菜单，如图 6-1 所示。

方法二，调用 JavaScript 脚本，通过 JavaScript 代码触发下拉菜单。为 <a> 定义一个 ID，以方便 JavaScript 抓取激活元素对象，当然也可以设置其他属性，如 class 属性。然后为该控制元件绑定 dropdown() 构造函数。代码如下：

图 6-1　设计简单的下拉菜单效果

```
<script type="text/JavaScript">
$(function(){
      $('#mydrop').dropdown();
})
</script>
<div class="dropdown">
    <a href="#" class="btn btn-large" id="mydrop">激活按钮 <i class="icon-arrow-
       down"></i></a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
</div>
```

通过 JavaScript 激活的下拉菜单不能够在显示和隐藏之间切换。在单击按钮后，显示下

拉菜单，再次单击时，不会隐藏下拉菜单，只有执行选择或者在其他区域单击之后，下拉菜单才能够隐藏，执行效果如图 6-2 所示。

图 6-2　设计简单的下拉菜单效果

第 5 步：创建简单的导航栏。下面利用 Bootstrap 提供的样式创建一个简单的导航栏。首先，使用 <ul class="nav nav-pills"> 创建一个水平显示的导航条，在导航条中包含三个下拉菜单项目。其中 nav 定义导航菜单框，nav-pills 定义水平显示的导航条，dropdown 定义下拉菜单包含框。

```
<ul class="nav nav-pills">
    <li class="dropdown"></li>
    <li class="dropdown"></li>
    <li class="dropdown"></li>
</ul>
```

然后，利用第 4 步中的方法一设计下拉菜单结构，并激活交互行为。

```
<ul class="nav nav-pills">
    <li class="dropdown">
        <a class="dropdown-toggle" data-toggle="dropdown" href="#"> 菜单项目 1 <b
           class="caret"></b></a>
        <ul id="menu1" class="dropdown-menu">
            <li><a href="#"> 子菜单栏 </a></li>
        </ul>
    </li>
    <li class="dropdown">
        <a class="dropdown-toggle" data-toggle="dropdown" href="#"> 菜单项目 2 <b
          class="caret"></b></a>
       <ul id="menu2" class="dropdown-menu">
            <li><a href="#"> 动作 1</a></li>
            <li><a href="#"> 动作 2</a></li>
            <li class="divider"></li>
            <li><a href="#"> 动作 3</a></li>
       </ul>
    </li>
    <li class="dropdown">
        <a class="dropdown-toggle" data-toggle="dropdown" href="#"> 菜单项目 3 <b
           class="caret"></b></a>
        <ul id="menu3" class="dropdown-menu">
            <li><a href="#"> 其他 </a></li>
        </ul>
    </li>
</ul>
```

其中 dropdown-toggle 定义下拉菜单的激活元素，data-toggle="dropdown" 属性激活下拉菜单交互行为，dropdown-menu 定义子菜单包含框，divider 定义子菜单中的分隔条。

此时，在浏览器中预览，显示效果如图 6-3 所示。

图 6-3　设计包含下拉菜单的导航条

6.1.2　设计下拉菜单

Bootstrap 为下拉菜单设计了一套严谨的结构，规定下拉菜单组件必须包含在 dropdown 类容器中，该容器包含下拉菜单的触发器（触发元素）和下拉菜单，下拉菜单必须包含在 dropdown-menu 容器中。基本结构如下：

```
<div class="dropdown">
    <a href="#" >激活元素 </a>
    <ul class="dropdown-menu">
    </ul>
</div>
```

如果下拉菜单组件不包含在 dropdown 类容器中，则可以使用声明为 position:relative ；的页面元素，例如：

```
<div style="position:relative;">
    <button> 激活元素 </button>
    <div class="dropdown-menu">
    </div>
</div>
```

在默认情况下，下拉菜单是隐藏显示的，如果显示查看下拉菜单效果，则必须在页面头部区域导入 3 个文件，缺一不可。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

然后，在触发元素中定义 data-toggle="dropdown" 属性，激活下拉菜单的交互行为，以方便查看下拉菜单效果。

```
<div style="position:relative;">
    <button data-toggle="dropdown"> 激活元素 </button>
    <div class="dropdown-menu">
    </div>
</div>
```

在下拉菜单容器（dropdown-menu）中可以包含任何元素和内容。但是，作为下拉菜单标

准结构，一般建议使用列表结构，并为每个列表项定义超级链接。下拉菜单的标准结构如下：

```
<div class="dropdown">
    <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 菜单项 1</a></li>
        <li><a href="#"> 菜单项 2</a></li>
        <li><a href="#"> 菜单项 3</a></li>
    </ul>
</div>
```

Bootstrap 为这套标准的下拉菜单结构打造了一套标准的样式效果，让下拉菜单看起来大气、精致，如图 6-4 所示。整个下拉菜单带有阴影，包含圆角，列表项目缩进显示，字体大气，项目间隔紧凑。

6.1.3 设计多级下拉菜单

使用 dropdown-submenu 类可以设计二级菜单，用法很简单：在现有下拉菜单结构中，向任意的 <li> 标签添加 dropdown-submenu 类，即可自动赋予一个二级菜单。例如，在下面的下拉菜单中，为第 3 个菜单项 <li> 标签添加 class="dropdown-submenu"，然后在该 <li> 标签中再嵌套一个下拉菜单容器 <ul class="dropdown-menu">，代码如下：

```

<div class="dropdown"> <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 菜单项 1</a></li>
        <li><a href="#"> 菜单项 2</a></li>
        <li class="dropdown-submenu"><a href="#"> 菜单项 3</a>
            <ul class="dropdown-menu">
                <li><a href="#"> 菜单项 3-1</a></li>
                <li><a href="#"> 菜单项 3-2</a></li>
                <li><a href="#"> 菜单项 3-3</a></li>
            </ul>
        </li>
    </ul>
</div>
```

通过添加一些简单的标记，就可以为下拉菜单增加一个二级菜单。这个二级菜单会在鼠标经过时自动展现，效果类似于 OS X，如图 6-5 所示。

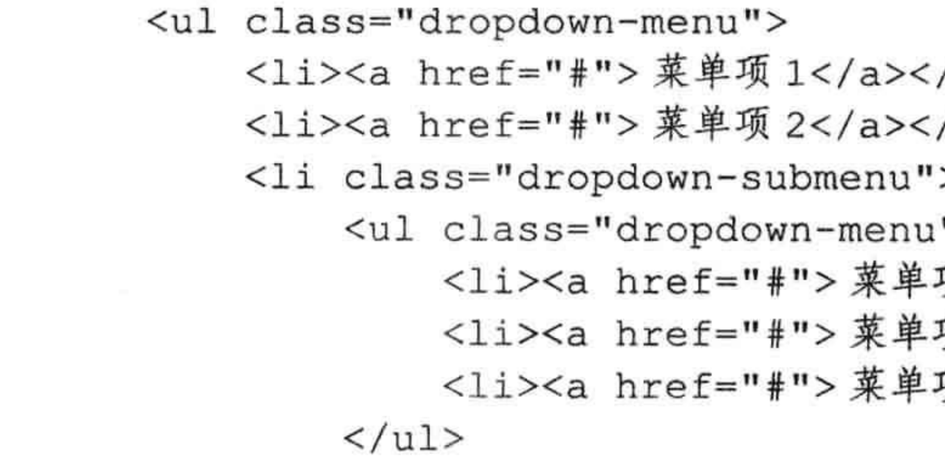


图 6-4 Bootstrap 设计的下拉菜单标准样式

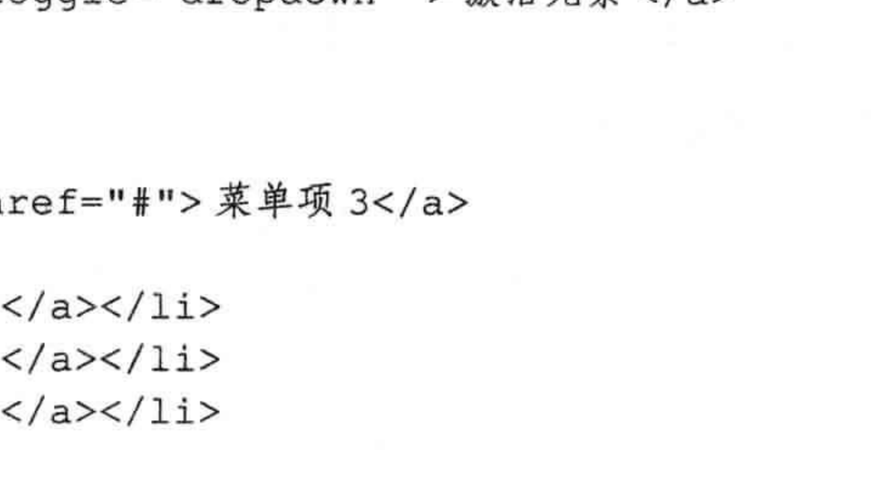


图 6-5 设计二级下拉菜单

以此类推，只要为任何级别的菜单项上引入 dropdown-submenu 类，就可以设计下级下拉菜单，这种多级菜单不需要任何触发器，鼠标经过时会自动展示下级菜单。例如，针对上面的二级菜单结构，可以拓展出 4 级菜单效果，代码如下，演示效果如图 6-6 所示。

```
<div class="dropdown"> <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 菜单项 1</a></li>
        <li><a href="#"> 菜单项 2</a></li>
        <li class="dropdown-submenu"><a href="#"> 菜单项 3</a>
            <ul class="dropdown-menu">
                <li><a href="#"> 菜单项 3-1</a></li>
                <li><a href="#"> 菜单项 3-2</a></li>
                <li class="dropdown-submenu"><a href="#"> 菜单项 3-3</a>
                    <ul class="dropdown-menu">
                        <li><a href="#"> 菜单项 3-3-1</a></li>
                        <li><a href="#"> 菜单项 3-3-2</a></li>
                        <li class="dropdown-submenu"><a href="#"> 菜单项 3-3-3</a>
                            <ul class="dropdown-menu">
                                <li><a href="#"> 菜单项 3-3-3-1</a></li>
                                <li><a href="#"> 菜单项 3-3-3-2</a></li>
                                <li><a href="#"> 菜单项 3-3-3-3</a> </li>
                            </ul>
                        </li>
                    </ul>
                </li>
            </ul>
        </li>
    </ul>
</div>
```

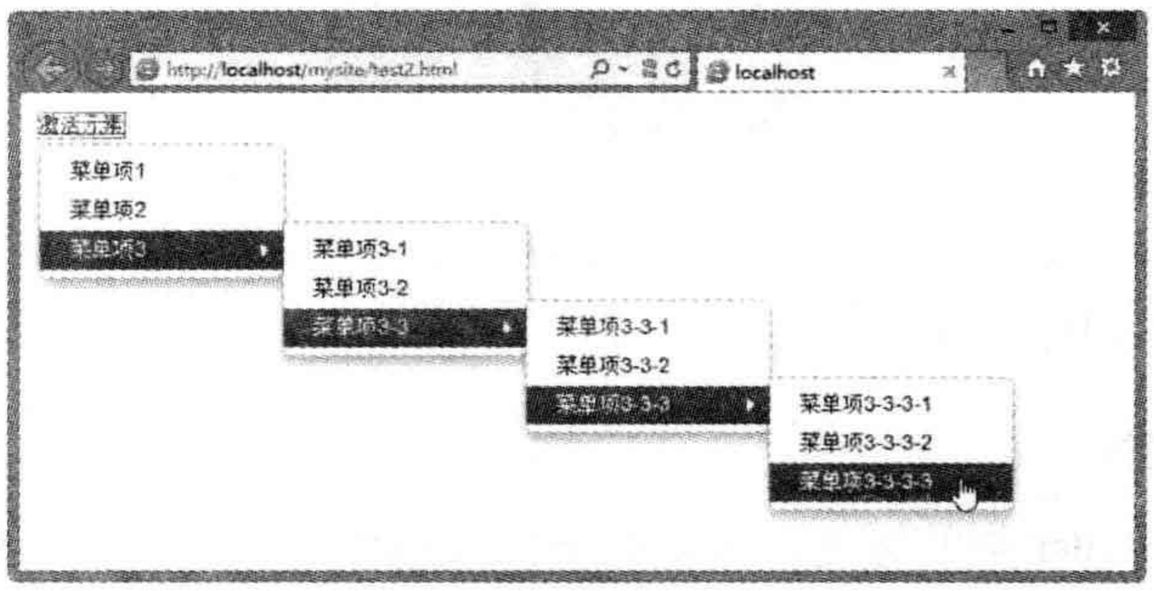


图 6-6　设计四级下拉菜单

6.1.4　设置下拉菜单选项

Bootstrap 为下拉菜单组件设置了一些可选项，以方便用户进行控制，简单说明如下。

1. 右对齐菜单

在默认状态下，下拉菜单是左对齐显示的。如果在 dropdown-menu 容器中添加 pull-right

类，即可设计右对齐下拉菜单；如果设置 pull-left 类，则可以实现左对齐效果。例如：

```
<div class="dropdown">
    <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu pull-right">
        <li><a href="#"> 菜单项 1</a></li>
        <li><a href="#"> 菜单项 2</a></li>
        <li><a href="#"> 菜单项 3</a></li>
    </ul>
</div>
```

上面的代码在浏览器中的预览效果如图 6-7 所示。

2. 禁用列表项

为下拉菜单中的 <li> 标签添加 disabled 类，可以禁用菜单项中链接。例如，在下拉菜单中为第二菜单项添加 class="disabled"，则在浏览器中的预览效果如图 6-8 所示。

```
<div class="dropdown">
    <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 菜单项 1</a></li>
        <li class="disabled"><a href="#"> 菜单项 2</a></li>
        <li><a href="#"> 菜单项 3</a></li>
    </ul>
</div>
```

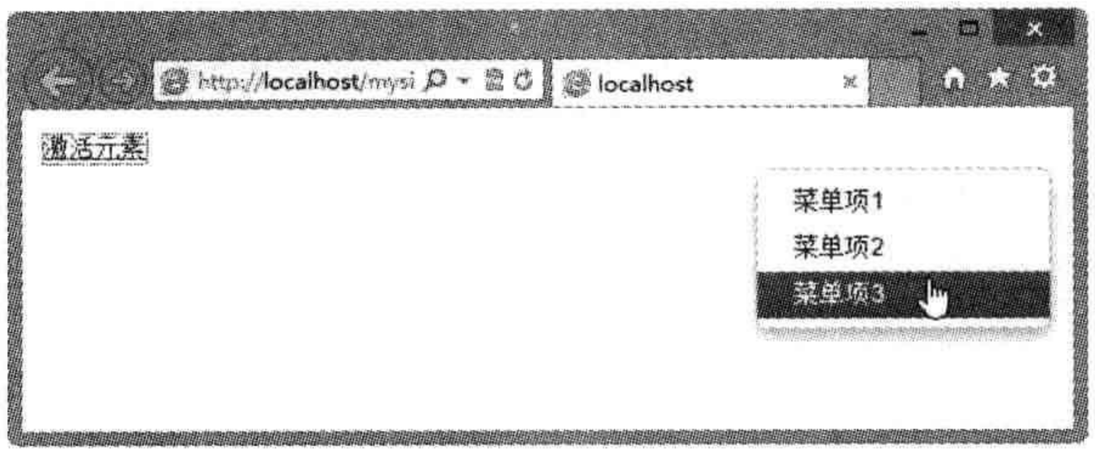


图 6-7　设计右对齐下拉菜单

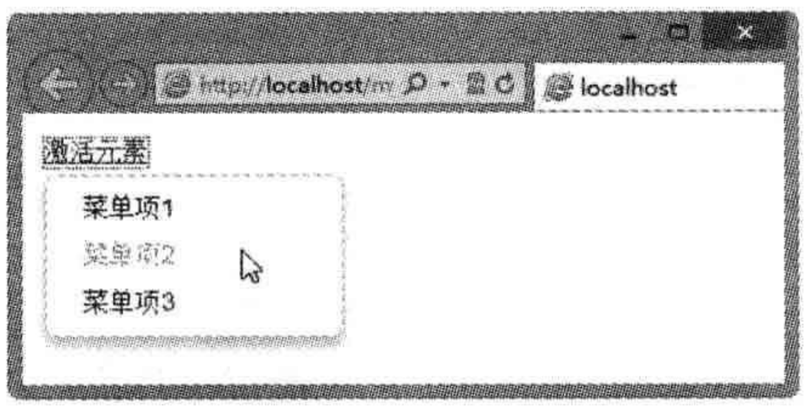


图 6-8　设计禁用菜单项

3. 设计菜单分隔线

通过添加包含 divider 类的标签，可以在下拉菜单中插入一条分隔线。例如，在上面的示例中，为第二与第三个菜单项插入一条分隔线（<li class="divider">），显示效果如图 6-9 所示。

```
<div class="dropdown">
    <a href="#" data-toggle="dropdown" > 激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 菜单项 1</a></li>
        <li><a href="#"> 菜单项 2</a></li>
        <li class="divider"></li>
```

```
            <li><a href="#">菜单项 3</a></li>
        </ul>
    </div>
```

4. 设计向上弹出菜单

把下拉菜单组件包含框改为 dropup，即可让下拉菜单向上弹出。例如，向下面二级下拉菜单外包含框添加 dropup 类（ ），代码如下，则显示效果如图 6-10 所示。

```
<div class="dropup"> <a href="#" data-toggle="dropdown" >激活元素 </a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li class="dropdown-submenu"><a href="#">菜单项 3</a>
            <ul class="dropdown-menu">
                <li><a href="#">菜单项 3-1</a></li>
                <li><a href="#">菜单项 3-2</a></li>
                <li><a href="#">菜单项 3-3</a></li>
            </ul>
        </li>
    </ul>
</div>
```


图 6-9　设计菜单项分隔线

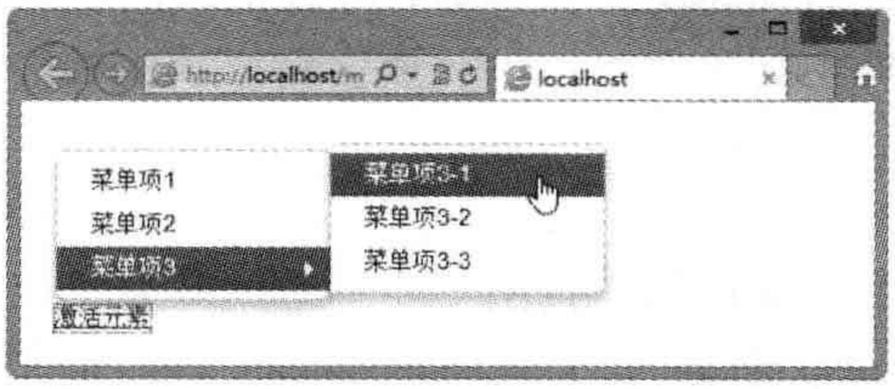


图 6-10　设计向上弹出下拉菜单

5. 设计向左弹出菜单

上面介绍过使用 pull-right 可以设计右对齐下拉菜单，反之使用 pull-left 可以设计左对齐菜单。如果为下级菜单添加 pull-left，则可以让它向左弹出显示。例如，为 <li class="dropdown-submenu"> 添加 pull-left，即可设计向左弹出菜单，显示效果如图 6-11 所示。

```
<div class="dropdown"> <a href="#" data-toggle="dropdown" >激活元素 </a>
    <ul class="dropdown-menu ">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li class="dropdown-submenu  pull-left"><a href="#">菜单项 3</a>
            <ul class="dropdown-menu">
                <li><a href="#">菜单项 3-1</a></li>
                <li><a href="#">菜单项 3-2</a></li>
```

```
                <li><a href="#">菜单项 3-3</a></li>
            </ul>
        </li>
    </ul>
</div>
```

图 6-11　设计向左弹出下拉菜单

6.2　按钮组

Bootstrap 非常重视按钮效果的设计，不仅提供了大量图标按钮，还精心打造了 btn 类按钮风格。通过对按钮分组管理，可以实现在 Web 中设计各种快捷操作风格，同时与下拉菜单等组件组合使用，能够设计各种精致的按钮导航版式，从而获得更多类似于工具条的功能，使组件中的按钮可以组合成多种样式，如按钮组（button group）和按钮式下拉菜单（button drown menu）。

6.2.1　设计按钮组

顾名思义，按钮组就是将多个按钮集合成一个页面组件。只需要使用 btn-group 类和一系列的 <a> 或者 <button> 标签，就可以轻易地生成一个按钮组或按钮工具条。

例如，在下面的示例代码中，带有 btn 类的多个标签被包含在 btn-group 中：

```
<div class=" btn-group">
    <p class="btn">按钮 1(p)</p>
    <li class="btn">按钮 2(li)</li>
    <a class="btn">按钮 3(a)</a>
    <span class="btn">按钮 4(span)</span>
</div>
```

上面的代码使用不同的标签定义了 4 个按钮，然后包含在 标签中，预览效果如图 6-12 所示。

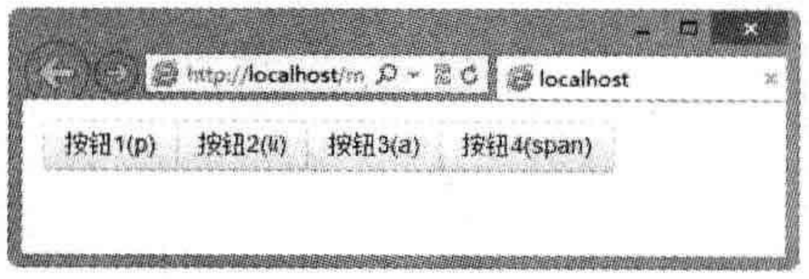


图 6-12　使用不同标签定义的 4 个按钮

注意，在设计按钮组时不需要导入 jquery.js 等脚本文件，因为在默认状态下，按钮不需要执行交互行为。但是必须导入 bootstrap.css 样式表。

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

注意

1）在单一的按钮组中不要混合使用 <a> 和 <button> 标签，仅用其中的一个。

2）同一按钮组最好使用单一色。

3）在使用图标的时候要确保引用位置正确。

6.2.2　设计按钮导航条

要将多个按钮组（btn-group）包含在一个 btn-toolbar 中，可以设计一个按钮工具条，以此设计一个更复杂的按钮组件。例如，下面的代码设计三组按钮组，然后把它们包含在 框中，得到一个分页导航条，演示效果如图 6-13 所示。

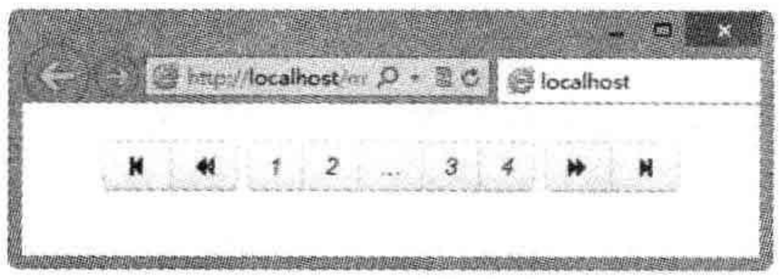


图 6-13　设计按钮导航条

```
<div class="btn-toolbar text-center">
    <div class=" btn-group">
        <i class="btn"><i class="icon-step-backward"></i></i>
        <i class="btn"><i class="icon-backward"></i></i>
    </div>
    <div class=" btn-group">
        <i class="btn">1</i>
        <i class="btn">2</i>
        <i class="btn">...</i>
        <i class="btn">3</i>
        <i class="btn">4</i>
    </div>
    <div class=" btn-group">
        <i class="btn"><i class="icon-forward"></i></i>
        <i class="btn"><i class="icon-step-forward"></i></i>
    </div>
</div>
```

注意，按钮必须包含在 btn-group 中，然后才能放入 btn-toolbar 中，只有这样才能正确渲染整个按钮导航条。

6.2.3　设计按钮布局

通过添加 btn-group-vertical 样式类，可以设计垂直分布的按钮组。例如，对于上面的代码，把 改为 ，或者直接添加 btn-group-vertical，即可设计成垂直按钮组效果，如图 6-14 所示。

```
<div class="btn-group-vertical">
    <p class="btn"> 按钮 1(p)</p>
```

```
    <li class="btn">按钮 2(li)</li>
    <a class="btn">按钮 3(a)</a>
    <span class="btn">按钮 4(span)</span>
</div>
```

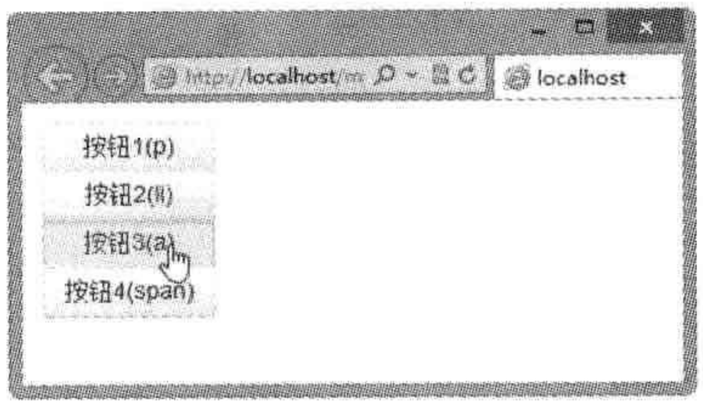


图 6-14　设计垂直分布的导航按钮

6.3　按钮式下拉菜单

Bootstrap 支持把按钮和下拉菜单捆绑在一起，形成按钮式下拉菜单。此时，可以将按钮包含在 btn-group 框中，并为其添加适当的菜单标签，即可让此按钮触发下拉菜单。

6.3.1　设计按钮式下拉菜单

例如，在下面的代码中，为第一个按钮绑定下拉菜单，通过 data-toggle="dropdown" 触发下拉菜单交互显现，此时在浏览器中预览效果如图 6-15 所示。

```
<div class="btn-group">
     <a class="btn" href="#" data-toggle="dropdown">按钮式下拉菜单 <i class="caret">
        </i> </a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
    <a class="btn" href="#">按钮 </a>
</div>
```

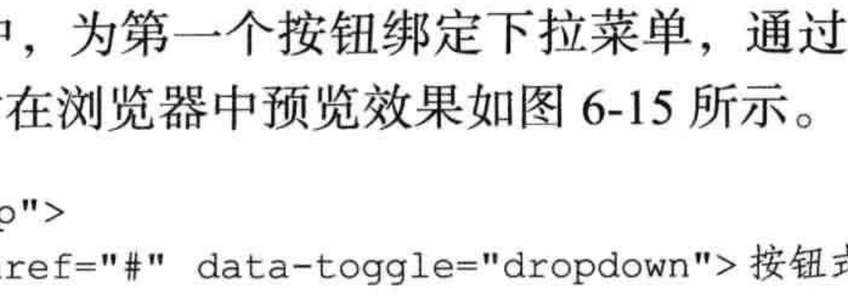


图 6-15　设计按钮式下拉菜单

注意，在设计按钮式下拉菜单时，应该引入 JavaScript：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
```

按钮式下拉菜单需要和 Bootstrap 下拉菜单插件（bootstrap-dropdown.js）配合使用。

在某些情况下，如果下拉菜单超出可视范围，则用户需要自己手工解决这一问题或者修改 JavaScript。

6.3.2　设计分隔样式

通过调整按钮式下拉菜单的 HTML 结构，可以设计按钮与向下指示图标分隔的效果。例如，在上面的代码中，把向下箭头单独包含在一个按钮标签中，同时定义 data-toggle="dropdown"，则预览效果如图 6-16 所示。

```
<div class="btn-group">
    <a class="btn" href="#">按钮式下拉菜单 </a>
    <a class="btn" href="#" data-toggle="dropdown"><i class="caret"></i></a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
    <a class="btn" href="#">按钮 </a>
</div>
```

按钮式下拉菜单可以兼容所有尺寸的按钮，当使用 btn-large、btn-small、btn-mini 分别设计按钮大小时，下拉菜单能够自动调整显示的位置，以便紧贴按钮下沿显示，如图 6-17 所示。

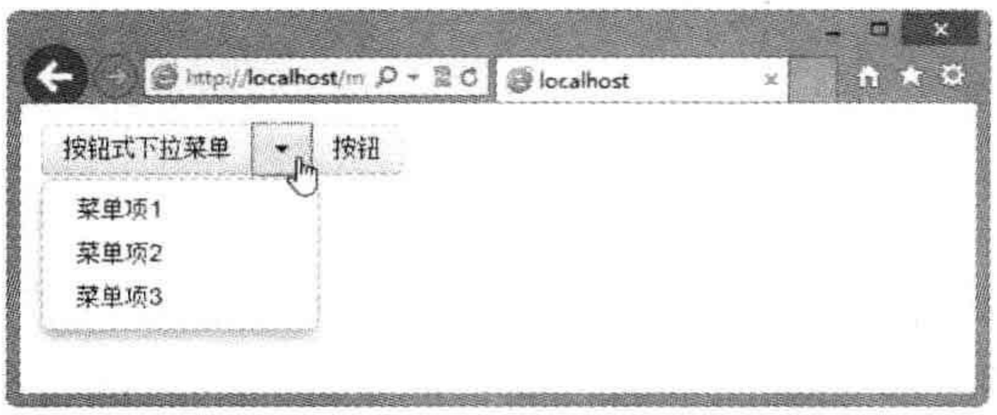


图 6-16　设计按钮式下拉菜单

图 6-17　设计大型按钮的下拉菜单

6.3.3　设计按钮式下拉菜单布局

只需要为按钮式下拉菜单包含框添加 dropup 类，也就是说为 dropdown-menu 的直接父节点添加 dropup 类，即可实现向上弹出式菜单。当向上弹出下拉菜单时，caret 将会自动翻转，菜单的位置也会变为由下到上，而不是由上到下了。例如，针对上面的示例代码，在 按钮组包含框中添加 dropup，则可以得到如图 6-18 所示的向上弹出式菜单效果。

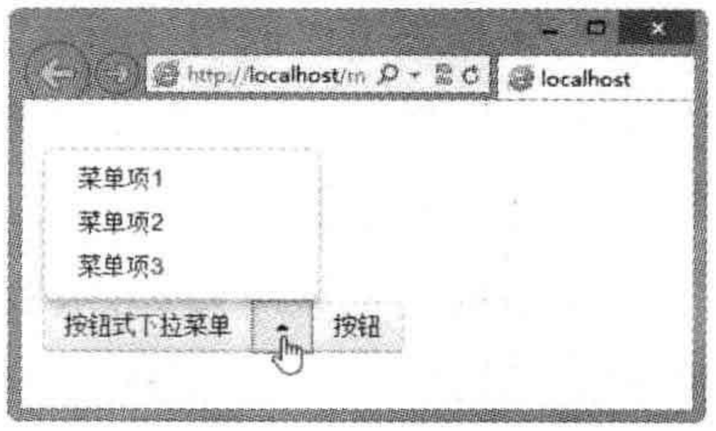


图 6-18　设计向上弹出的按钮式下拉菜单

```
<div class="btn-group dropup">
    <a class="btn" href="#">按钮式下拉菜单 </a>
    <a class="btn" href="#" data-toggle="dropdown"><i class="caret"></i></a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
    <a class="btn" href="#">按钮 </a>
</div>
```

6.4 导航

Bootstrap 提供灵活、多样的的导航组件，允许使用相同的标签、不同的类，实现不同风格的导航样式，具有非常高的可定制性。所有的导航组件，包括标签页、pills、导航列表标签，都必须使用 nav 类实现基础的导航效果。除了常见的导航，还可以利用 nav-stacked 类实现堆叠式导航版式。在第 5 章中曾经介绍过按钮组的结构和样式，下面详细介绍其他几种导航结构和样式。

6.4.1 设计导航组件

所有的导航组件都具有相同的结构，并共用一个样式类 nav。基本结构代码如下：

```
<ul class="nav">
    <li class="active"><a href="#">首页 </a></li>
    <li><a href="#">导航标题 1</a></li>
    <li><a href="#">导航标题 2</a></li>
</ul>
```

在网页中大部分信息都是通过列表形式显示的，如分类、列表、菜单、排行等，除了网页正文的段落文本和标题文本外，其他信息都需要列表结构进行组织和管理。列表结构是标准结构中最核心的组件之一，不管从语义角度分析，还是从表现层控制角度分析，使用列表结构来显示信息都是最佳选择，因此 Bootstrap 导航组件也是以列表结构为基础进行设计的。

HTML 提供三种列表结构：无序列表（ul）、有序列表（ol）和自定义列表（dl）。无序列表和有序列表可以通用，而自定义列表还包含了一个项目标题选项。Bootstrap 支持使用无序列表和有序列表来定义导航结构，但是暂时不支持自定义列表。

1. 设计标签页

为导航结构添加 nav-tabs 样式类，就可以设计标签页（Tab，或称选项卡）。例如，在上面的列表结构中，为 <ul class="nav"> 添加 nav-tabs 样式类，则结构呈现效果如图 6-19 所示。

```
<ul class="nav nav-tabs">
    <li class="active"><a href="#">首页 </a></li>
    <li><a href="#">导航标题 1</a></li>
```

```
    <li><a href="#"> 导航标题 2</a></li>
</ul>
```

2. 设计 pills 导航

为导航结构添加 nav-pills 样式类，就可以设计 pills（胶囊式）导航。例如，在上面的列表结构中，为 <ul class="nav"> 添加 nav-pills 样式类，则结构呈现效果如图 6-20 所示。

```
<ul class="nav nav-pills">
    <li class="active"><a href="#"> 首页 </a></li>
    <li><a href="#"> 导航标题 1</a></li>
    <li><a href="#"> 导航标题 2</a></li>
</ul>
```

图 6-19　设计标签页效果

图 6-20　设计 pills 导航效果

6.4.2　设置导航选项

导航结构是固定的，但是导航样式可以控制。Bootstrap 提供了多个设置选项，方便对导航进行适当控制，用户也可以通过手工方式修改 CSS 样式代码，实现更高级别的导航效果改造。

1. 设计导航对齐方式

6.1 节介绍过使用 pull-right 定义下拉子菜单向右浮动。实际上，pull-left 和 pull-right 是 Bootstrap 提供的两个通用工具类，分别定义向左对齐和向右对齐。这两个样式类简单定义了元素向左或者向右浮动。

```
.pull-right {
    float: right;
}
.pull-left {
    float: left;
}
```

在导航结构中，我们同样可以使用 pull-left 和 pull-right 工具类来对齐导航链接。例如，在下面的标签页导航中，通过添加 pull-right 类，让整个导航在页面或者包含框右侧显示，效果如图 6-21 所示。

```
<ul class="nav nav-tabs pull-right">
    <li class="active"><a href="#"> 首页 </a></li>
    <li><a href="#"> 微客 </a></li>
    <li><a href="#"> 微博 </a></li>
</ul>
```

图 6-21　设计右对齐标签页效果

同样，使用 pull-left 可以让导航结构向左对齐，不过该结构默认为左对齐，所以可以不用该样式类。

2. 设计禁用项

disabled 也是一个通用工具类，用于定义不可用状态的样式效果，但是 Bootstrap 针对不同的组件进行了个性化重写。例如，针对导航结构来说，不可用选项效果以浅灰色字体表示，同时当鼠标经过或者激活时，样式不会发生变化，其样式代码如下：

```
.nav > .disabled > a {
    color: #999999;
}
.nav > .disabled > a:hover,
.nav > .disabled > a:focus {
    text-decoration: none;
    cursor: default;
    background-color: transparent;
}
```

例如，在下面结构中，为标签页中第二个选项添加 disabled 样式类，设计该项为不可用状态，则效果如图 6-22 所示。

```
<ul class="nav nav-tabs">
    <li class="active"><a href="#">首页</a></li>
    <li class="disabled"><a href="#">微客</a></li>
    <li><a href="#">微博</a></li>
</ul>
```

注意，当为导航组件添加 disabled 样式类时，均可以设置超链接变灰，并使其失去鼠标悬停效果。但是 disabled 只是一个样式类，不能够控制行为，链接仍然是可以单击的，除非将超链接的 href 属性去除，或者通过 JavaScript 脚本阻止用户单击链接。

3. 设计堆叠效果

导航组件在默认状态下是水平显示的，如果添加一个 nav-stacked 样式类即可让组件以堆叠式进行排列，即恢复列表结构的默认垂直方式显示。nav-stacked 样式类实际上就是清除列表项的浮动显示。

```
.nav-stacked > li {
    float: none;
}
```

例如，为标签页结构添加 nav-stacked 样式类，则显示效果如图 6-23 所示。

```
<ul class="nav nav-tabs nav-stacked">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
    <li><a href="#">微博</a></li>
</ul>
```

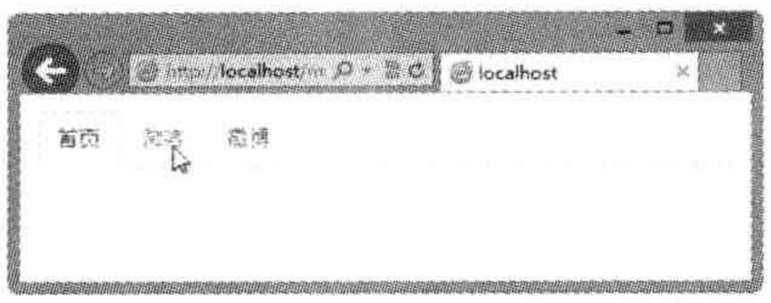


图 6-22　设计不可用状态

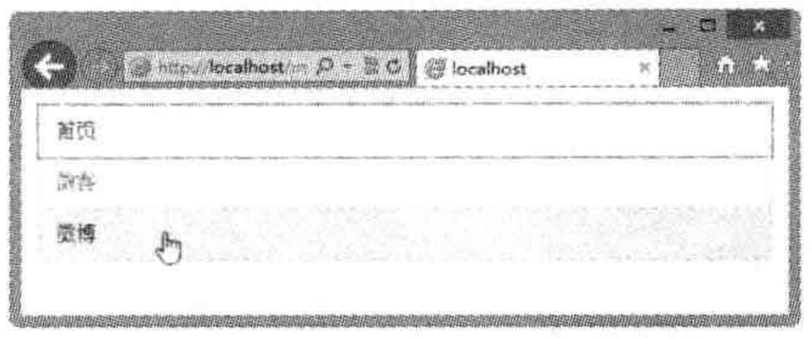


图 6-23　设计堆叠式标签页效果

同样，针对 pills 导航结构添加 nav-stacked 样式类（<ul class="nav nav-pills nav-stacked">），则显示效果如图 6-24 所示。

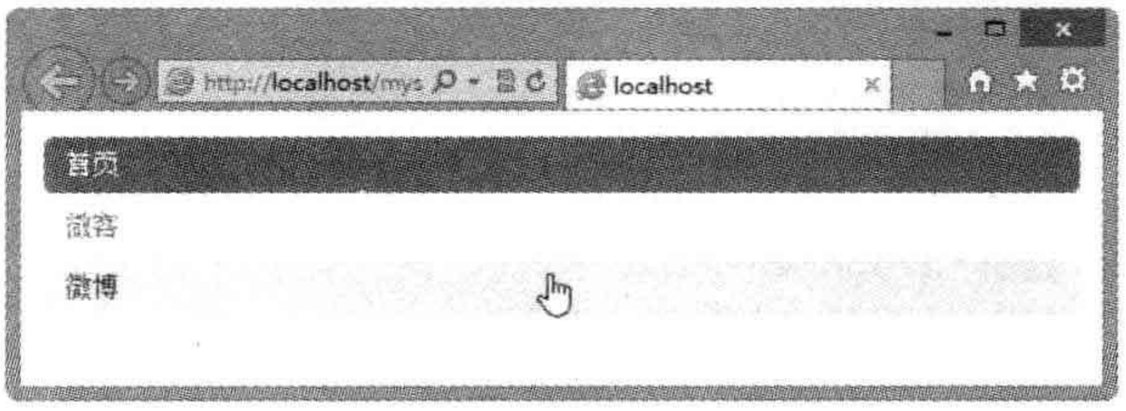


图 6-24　设计堆叠式 pills 导航效果

6.4.3　绑定导航和下拉菜单

下拉菜单（dropdown）是一个独立的组件，它可以与页面中任何元素（如按钮、导航等）捆绑使用。我们将一段 HTML 代码和下拉菜单捆绑在一起，然后借助下拉菜单的 JavaScript 插件，即可设计一个导航菜单。

在操作之前，应该先导入下拉菜单的 JavaScript 插件，同时导入 jQuery 库文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
```

1. 设计标签页下拉菜单

在标签页选项中，包含一个下拉菜单结构，然后为标签项添加 dropdown 类，为下拉菜单结构添加 dropdown-menu。最后，在标签项的超链接中绑定激活属性 data-toggle="dropdown"，整个效果就设计完毕。

例如，针对上面示例中的标签结构，为第三个标签项添加一个下拉菜单，并添加一个向下箭头进行标识（），效果如图 6-25 所示。

```
<ul class="nav nav-tabs">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
   <li class="dropdown"><a data-toggle="dropdown" href="#">微博  <b class="
     caret"></b></a>
          <ul class="dropdown-menu">
               <li><a href="#">登录</a></li>
```

```
                <li><a href="#">注册</a></li>
                <li><a href="#">退出</a></li>
            </ul>
        </li>
    </ul>
```

2. 设计 pills 下拉菜单

同样，针对 pills 导航结构，也可以进行相同的操作，设计一个 pills 下拉菜单。例如，把上面的示例稍加修改，把标签页换成 pills 导航，则效果如图 6-26 所示。

```
<ul class="nav nav-pills">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
    <li class="dropdown"><a data-toggle="dropdown" href="#">微博  <b class=
      "caret"></b></a>
          <ul class="dropdown-menu">
           <li><a href="#">登录</a></li>
           <li><a href="#">注册</a></li>
           <li><a href="#">退出</a></li>
         </ul>
    </li>
</ul>
```

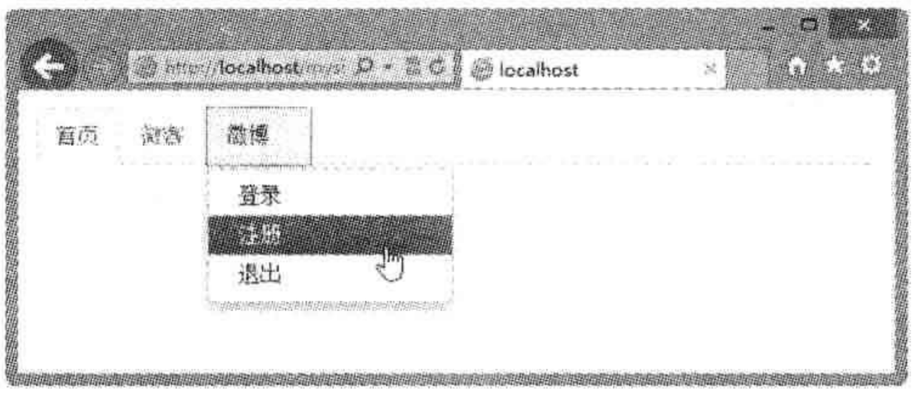



图 6-25　设计标签页下拉菜单效果

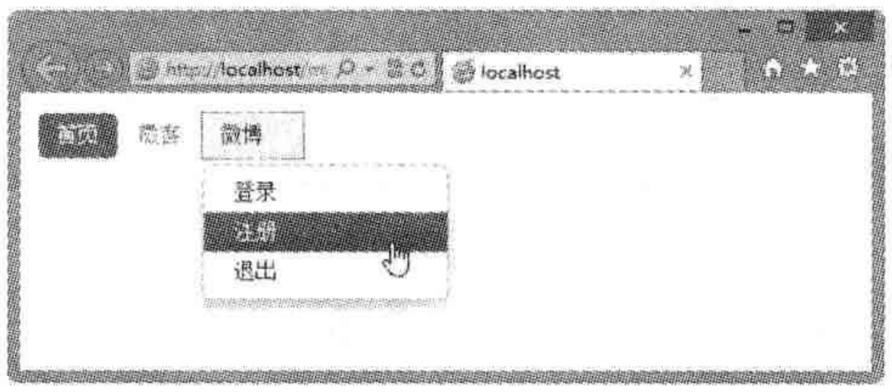


图 6-26　设计 pills 下拉菜单效果

6.4.4　设计导航列表

导航列表是一种简洁式导航样式，与标签页、pills 下拉菜单具有相同列表结构，只要引入 nav-list 样式类即可。例如，针对标签页中的示例结构，把 <ul class="nav nav-tabs"> 修改为 <ul class="nav nav-list">，即可定义一个导航列表，效果如图 6-27 所示。

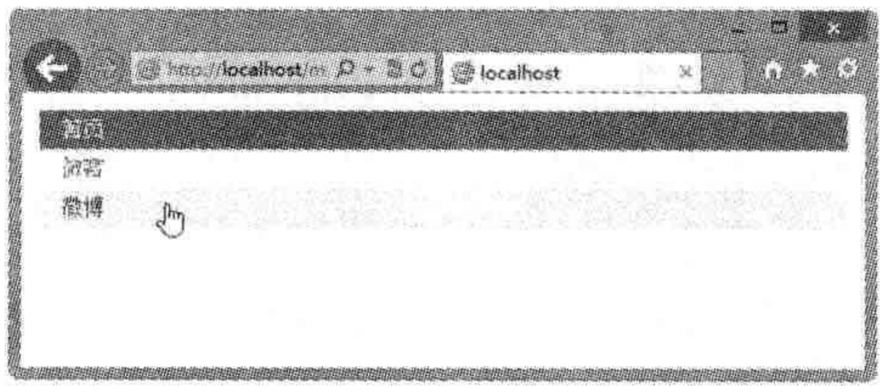


图 6-27　设计导航列表效果

```
<ul class="nav nav-list">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
    <li><a href="#">微博</a></li>
</ul>
```

导航列表经常被用在侧栏位置，效果类似于 OS X 中的 Finder。

导航列表可以相互嵌套，用来设计多级导航列表结构。其用法与下拉菜单相似，只需为嵌套的列表结构添加 nav 和 nav-list 样式类即可，效果如图 6-28 所示。

```
<ul class="nav nav-list">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
    <li><a href="#">微博</a>
        <ul class="nav nav-list">
            <li><a href="#">登录</a></li>
            <li><a href="#">注册</a></li>
            <li><a href="#">退出</a></li>
        </ul>
    </li>
</ul>
```

在导航列表中通过添加 divider 样式类，可以为列表结构插入一条分隔线。例如，在下面的结构中为第二和第三列表项插入一条分隔线，则效果如图 6-29 所示。

```
<ul class="nav nav-list">
    <li class="active"><a href="#">首页</a></li>
    <li><a href="#">微客</a></li>
    <li class="divider"></li>
    <li><a href="#">微博</a></li>
</ul>
```

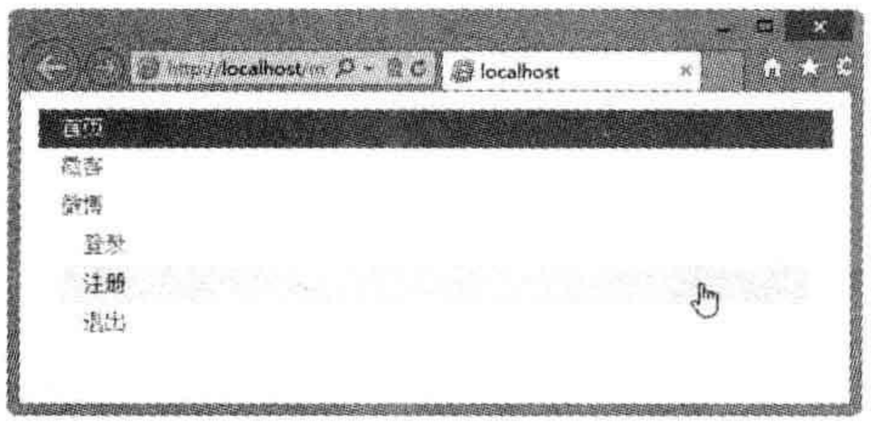


图 6-28　设计导航列表嵌套效果

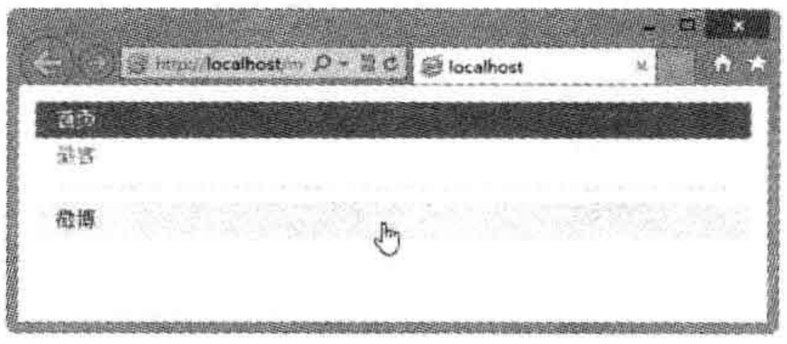


图 6-29　设计导航列表的分隔线效果

6.4.5　激活标签页

激活标签页就是让标签页每个 Tab 项能够自由切换，并能够控制 Tab 项目对应内容框的显示和隐藏。具体方法如下。

第 1 步：需要用到 jQuery 插件的支持，并导入 bootstrap-tab.js 文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-tab.js"></script>
```

第 2 步：在标签页结构基础上，添加内容包含框，通过 tab-content 定义包含框为标签页的内容显示框。在内容包含框中插入与标签页结构对应的多个子内容框，并使用 tab-pane 进

行定义。

第 3 步：为每个内容框定义 id 值，并在标签列表项中为每个超链接绑定锚链接。

第 4 步：为每个标签项超链接定义 data-toggle="tab" 属性，激活标签页的交互行为。完整的代码如下：

```
<div>
    <ul class="nav nav-tabs">
        <li class="active"><a href="#tab1" data-toggle="tab">首页</a></li>
        <li><a href="#tab2" data-toggle="tab">微客</a></li>
        <li><a href="#tab3" data-toggle="tab">微博</a></li>
    </ul>
    <div class="tab-content">
        <div class="tab-pane active" id="tab1">首页内容框</div>
        <div class="tab-pane" id="tab2">微客内容框</div>
        <div class="tab-pane" id="tab3">微博内容框</div>
    </div>
</div>
```

第 5 步：在浏览器中预览，则显示效果如图 6-30 所示。

如果要设计标签页淡入效果，只需为每个标签页选项 tab-pane 添加 fade 类即可。例如，在上面的示例中，分别为每个 添加 fade 类，则可以看到当切换 Tab 选项时，会有淡入的效果，如图 6-31 所示。


```
<div class="tabbable">
    <ul class="nav nav-tabs">
        <li class="active"><a href="#tab1" data-toggle="tab">首页</a></li>
        <li><a href="#tab2" data-toggle="tab">微客</a></li>
        <li><a href="#tab3" data-toggle="tab">微博</a></li>
    </ul>
    <div class="tab-content">
        <div class="tab-pane active" id="tab1"><img src="images/1.jpg"></div>
        <div class="tab-pane fade" id="tab2"><img src="images/2.jpg"></div>
        <div class="tab-pane fade" id="tab3"><img src="images/3.jpg"></div>
    </div>
</div>
```

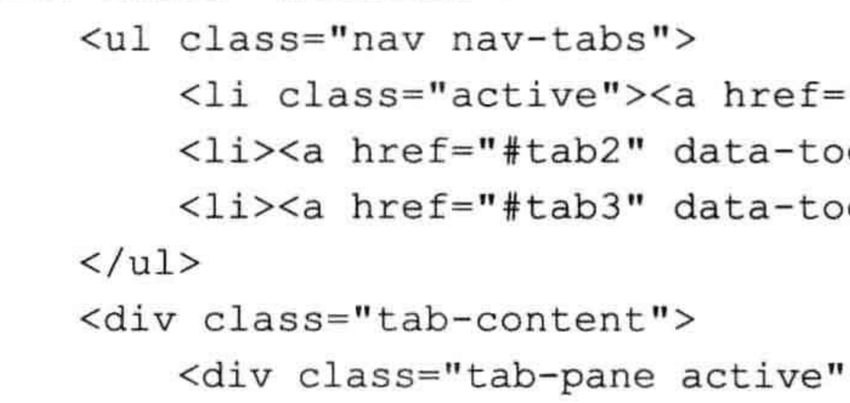


图 6-30　激活标签页交互效果

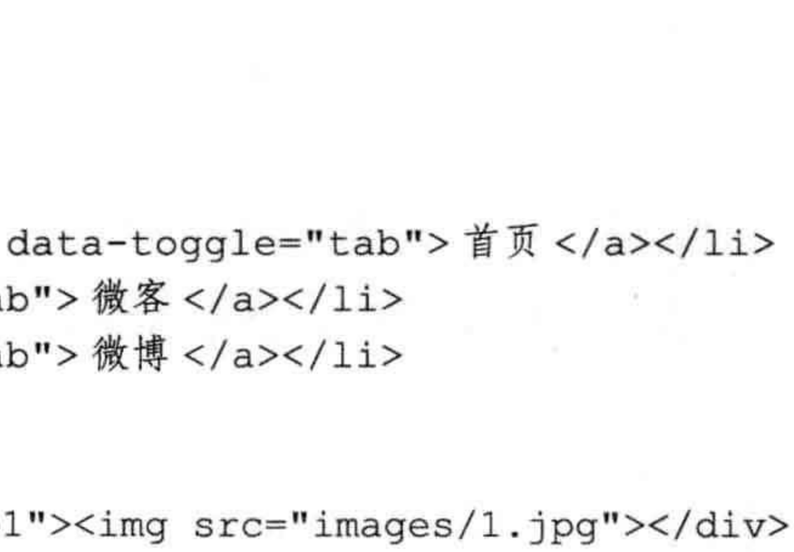


图 6-31　设计淡入交互效果

6.4.6　设计标签页布局

Bootstrap 整合了 4 种标签页布局样式：top（默认）、right、bottom 和 left。要设计标签页布局，向标签页组件包含框中添加标签页布局类即可。

例如，在标签页组件包含框中添加 tabs-left 类，可以设计 Tab 标题栏左侧显示，效果如图 6-32 所示。

```
<div class="tabbable tabs-left">
    <ul class="nav nav-tabs">
        <li class="active"><a href="#tab1" data-toggle="tab">首页</a></li>
        <li><a href="#tab2" data-toggle="tab">微客</a></li>
        <li><a href="#tab3" data-toggle="tab">微博</a></li>
    </ul>
    <div class="tab-content">
        <div class="tab-pane active" id="tab1"><img src="images/1.jpg"></div>
        <div class="tab-pane fade" id="tab2"><img src="images/2.jpg"></div>
        <div class="tab-pane fade" id="tab3"><img src="images/3.jpg"></div>
    </div>
</div>
```

提示，tabbable 类主要用于兼容 IE 早期版本，促使标签页包含框具有布局特性，防止布局发生环绕和错位现象，其样式代码如下：

```
.tabbable {
    *zoom: 1;
}
.tabbable:before,
.tabbable:after {
    display: table;
    line-height: 0;
    content: "";
}
.tabbable:after {
    clear: both;
}
```

如果为标签页包含框添加 tabs-right 样式类（ ），则显示效果如图 6-33 所示。

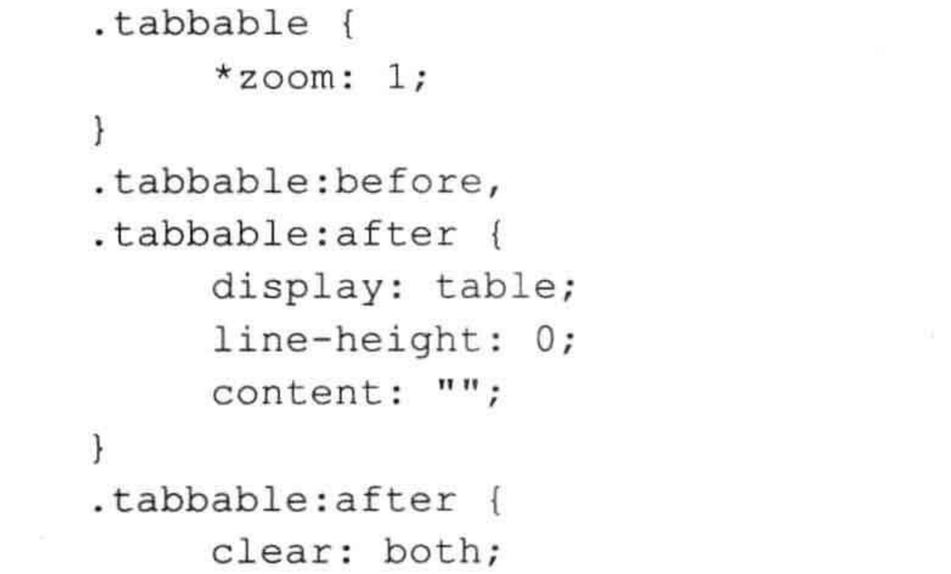


图 6-32　设计标签页标题栏左侧显示

图 6-33　设计标签页标题栏右侧显示

如果为标签页包含框添加 tabs-below 样式类（ ），同时调整 <ul class="nav nav-tabs"> 和 之间的结构顺序，则显示效果如图 6-34 所示。

```
<div class="tabbable tabs-below">
    <div class="tab-content">
        <div class="tab-pane active" id="tab1"><img src="images/1.jpg"></div>
        <div class="tab-pane fade" id="tab2"><img src="images/2.jpg"></div>
        <div class="tab-pane fade" id="tab3"><img src="images/3.jpg"></div>
    </div>
    <ul class="nav nav-tabs">
        <li class="active"><a href="#tab1" data-toggle="tab">首页</a></li>
        <li><a href="#tab2" data-toggle="tab">微客</a></li>
        <li><a href="#tab3" data-toggle="tab">微博</a></li>
    </ul>
</div>
```

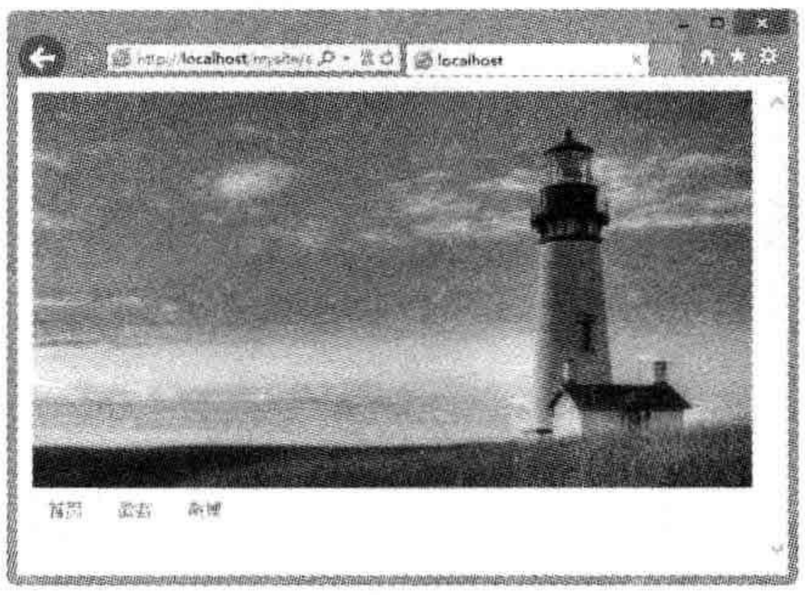


图 6-34　设计标签页标题栏底部显示

6.5　导航条

对导航组件进行适当包装，即可设计导航条。导航条是网页设计中不可缺少的部分，是整个网站的控制中枢，在每个页面都会看见它，因此如何设计导航就成为网页设计中很关键的一步。它可以帮助用户很方便地访问所需的内容，是浏览网站时从一个页面转到另一个页面的快速通道。

6.5.1　设计导航条

导航条是一个长条形区块，其中可以包含导航或按钮，以方便用户执行导航操作。Bootstrap 配合使用 navbar 和 navbar-inner 类定义导航条包含框，此时的导航条是一个空白区域，效果如图 6-35 所示。

```
<div class="navbar">
    <div class="navbar-inner">
    </div>
</div>
```

图 6-35　设计空的导航条

在导航条中包含导航结构，就可以设计更实用的导航条效果，如图 6-36 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <ul class="nav">
            <li class="active"><a href="#"> 首页 </a></li>
            <li><a href="#"> 导航标题 1</a></li>
            <li><a href="#"> 导航标题 2</a></li>
        </ul>
     </div>
</div>
```

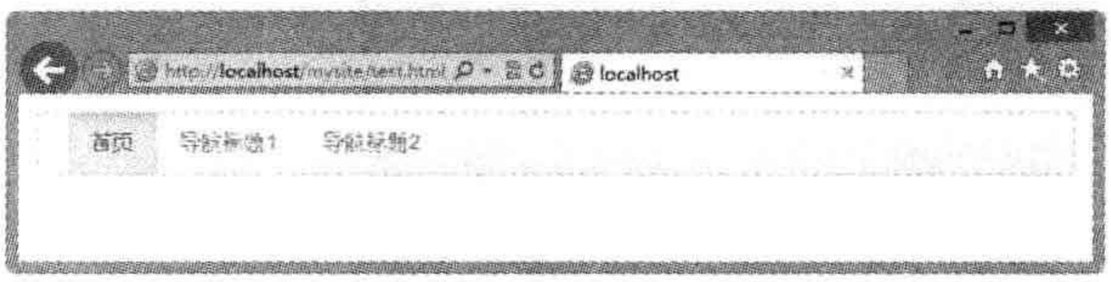


图 6-36　设计导航条效果

在默认情况下，导航条是静态的（static），不是定位显示的（fixed、absolute）。

完整的导航条一般包含项目（或网站）名称和导航项。项目名称使用 brand 样式类进行设计，一般位于导航条的左侧。例如，在下面代码中通过 <a class="brand"> 为导航条添加一个网站标识名称，效果如图 6-37 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <a class="brand" href="#"> 网站名称 </a>
        <ul class="nav">
            <li class="active"><a href="#"> 首页 </a></li>
            <li><a href="#"> 导航标题 1</a></li>
            <li><a href="#"> 导航标题 2</a></li>
        </ul>
    </div>
</div>
```

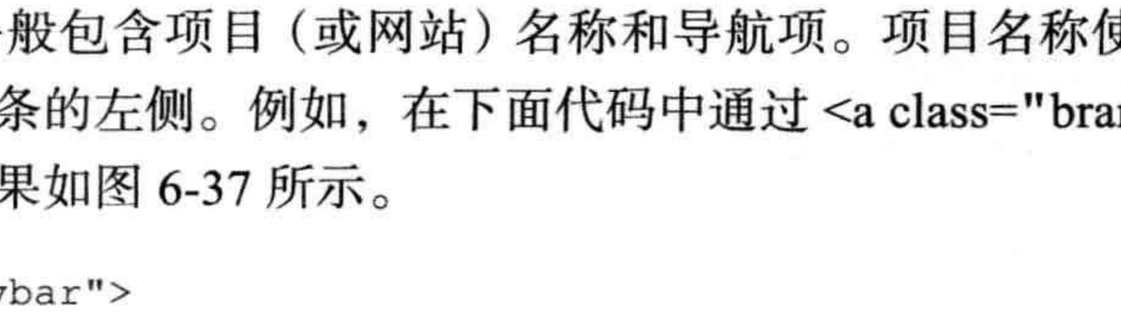


图 6-37　设计导航条标题效果

如果将导航条放入 container 包含框中，可以限制导航条的宽度，默认为 960 像素。

使用 divider-vertical 可以为导航条添加项目分隔线，其用法与 divider 样式类的用法是相同的。例如，针对上面的示例结构，为其插入两条分隔线，用来分隔每个项目，效果如图 6-38 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <a class="brand" href="#">网站名称</a>
        <ul class="nav">
            <li class="active"><a href="#">首页</a></li>
            <li class="divider-vertical"></li>
            <li><a href="#">导航标题 1</a></li>
            <li class="divider-vertical"></li>
            <li><a href="#">导航标题 2</a></li>
        </ul>
    </div>
</div>
```

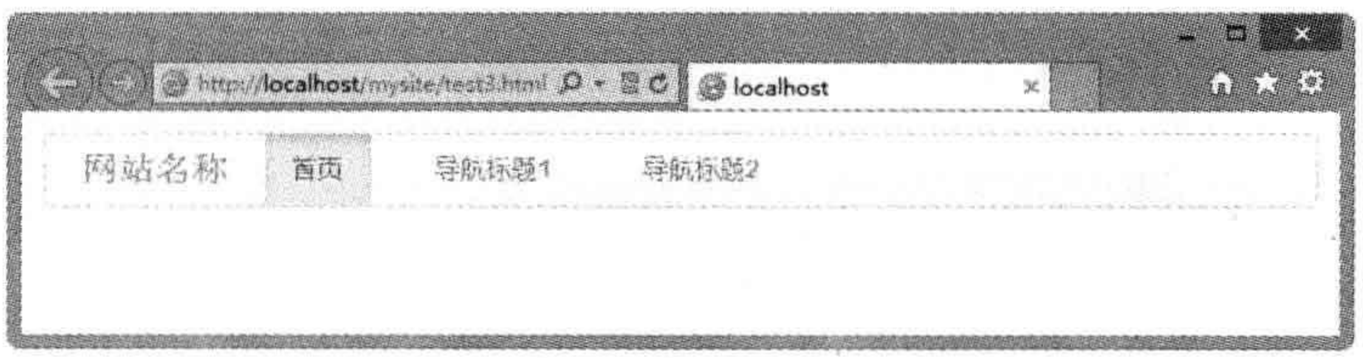


图 6-38　设计导航条分隔线效果

6.5.2　绑定表单和下拉菜单

在 Bootstrap 中，导航条被视为一个容器，可以包含导航组件，也可以包含表单或者下拉菜单。使用比较灵活、方便。

1. 设计表单导航条

如果希望在导航条中放置一个表单，需要为表单框添加 navbar-form 样式类，同时设置对齐方式（如 pull-left 或 pull-right）。

例如，设计一个提交表单，并把它放置于导航条中，为 <form> 表单框添加 navbar-form，并通过 pull-left 让表单左对齐，演示效果如图 6-39 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <form class="navbar-form pull-left">
            <input type="text" class="span3">
            <input type="text" class="btn span1" value="提交" />
        </form>
    </div>
</div>
```

如果使用 pull-right（<form class="navbar-form pull-right">），则可以让表单显示在导航条的右侧，如图 6-40 所示。

图 6-39　设计表单导航效果

图 6-40　设计表单导航右对齐效果

在导航条内为 <form> 添加 navbar-search，并为输入文本框添加 search-query，即可获得一个搜索表单。代码如下，演示效果如图 6-41 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <form class="navbar-search form-search pull-left">
            <div class="input-append">
                <input type="text" class="search-query span3">
                <input type="text" class="btn span1" value="提交" />
            </div>
        </form>
    </div>
</div>
```

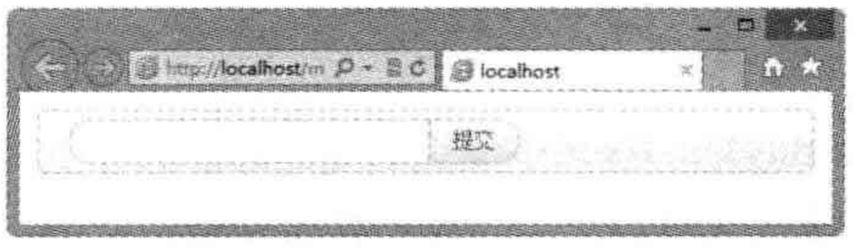



图 6-41　设计搜索表单导航条

提示　使用 pull-left 或 .pull-right 工具类可以对导航条内的链接、搜索表单或文本进行左右对齐操作，每个 CSS 类都会指定浮动的方向。

2. 设计下拉菜单导航条

在操作之前，应该先导入下拉菜单 JavaScript 插件，同时导入 jQuery 库文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
```

然后在导航条包含框中添加下拉菜单结构，设计一个简单的导航条下拉菜单，代码如下，演示效果如图 6-42 所示。

```
<div class="navbar">
    <div class="navbar-inner">
        <ul class="nav">
            <li class="dropdown"><a data-toggle="dropdown" href="#">微博 <b class=
             "caret"></b></a>
```

```
                <ul class="dropdown-menu">
                    <li><a href="#">登录</a></li>
                    <li><a href="#">注册</a></li>
                    <li><a href="#">退出</a></li>
                </ul>
            </li>
        </ul>
    </div>
</div>
```

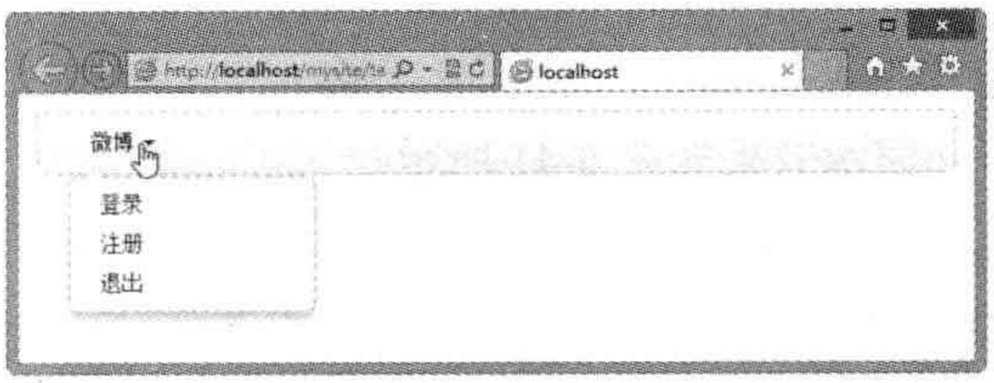


图 6-42　设计下拉菜单导航条效果

6.5.3　导航条布局

导航条可以在页面中进行固定布局，如固定显示在浏览器窗口的顶部或者底部，也可以改变导航条的样式风格，或者设计响应式导航条。

1. 置顶导航条

为导航条外包含框添加 navbar-fixed-top 类，就可以让导航条置顶显示。注意，为了确保导航条不覆盖其他页面内容，建议给 <body> 增加 40px 的 padding-top（内补）。一定要在 Bootstrap 核心 CSS（即 bootstrap.css）文件之后，响应式 CSS（bootstrap-responsive.css）文件之前添加该样式。

例如，在下面的示例中为页面插入一个置顶导航条，并定义 body 顶部补白为 40 像素，整个页面的完整代码如下，页面浏览效果如图 6-43 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<style type="text/css">
body { padding-top: 40px; }
</style>
</head>
<body>
<div class="navbar navbar-fixed-top">
    <div class="navbar-inner">
        <a class="brand" href="#">置顶导航条</a>
        <form class="navbar-form pull-left">
```

```
            <input type="text" class="span3">
            <input type="text" class="btn span1" value="提交" />
        </form>
    </div>
</div>
<div style="height:2000px; border:solid 1px red; margin:6px;"></div>
</body>
</html>
```

图 6-43　设计置顶导航条效果

2. 置底导航条

同样，如果为导航条外包含框添加 navbar-fixed-bottom 样式类，则可以让导航条置底显示。此时，也应该为 <body> 标签定义底部补白为 40 像素，以避免导航条遮盖住网页正文内容，演示效果如图 6-44 所示。

```
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<style type="text/css">
body { padding-bottom: 40px; }
</style>
<div class="navbar navbar-fixed-bottom">
    <div class="navbar-inner">
        <a class="brand" href="#">置底导航条</a>
        <form class="navbar-form pull-left">
            <input type="text" class="span3">
            <input type="text" class="btn span1" value="提交" />
        </form>
    </div>
</div>
<div style="height:2000px; border:solid 1px red; margin:6px;"></div>
```

图 6-44　设计置底导航条效果

3. 设计导航条反色效果

通过为导航条外包含框添加 navbar-inverse 样式类，可以设计反色效果的导航条。例如，在搜索文本框的导航条外框中添加 navbar-inverse 类样式，则预览效果如图 6-45 所示。

```
<div class="navbar navbar-inverse">
    <div class="navbar-inner">
        <a class="brand" href="#">导航条</a>
        <form class="navbar-search form-search pull-left">
            <div class="input-append">
                <input type="text" class="search-query span3">
                <input type="text" class="btn span1" value="搜索" />
            </div>
        </form>
    </div>
</div>
```

图 6-45　设计反色效果导航条

4. 设计响应式导航条

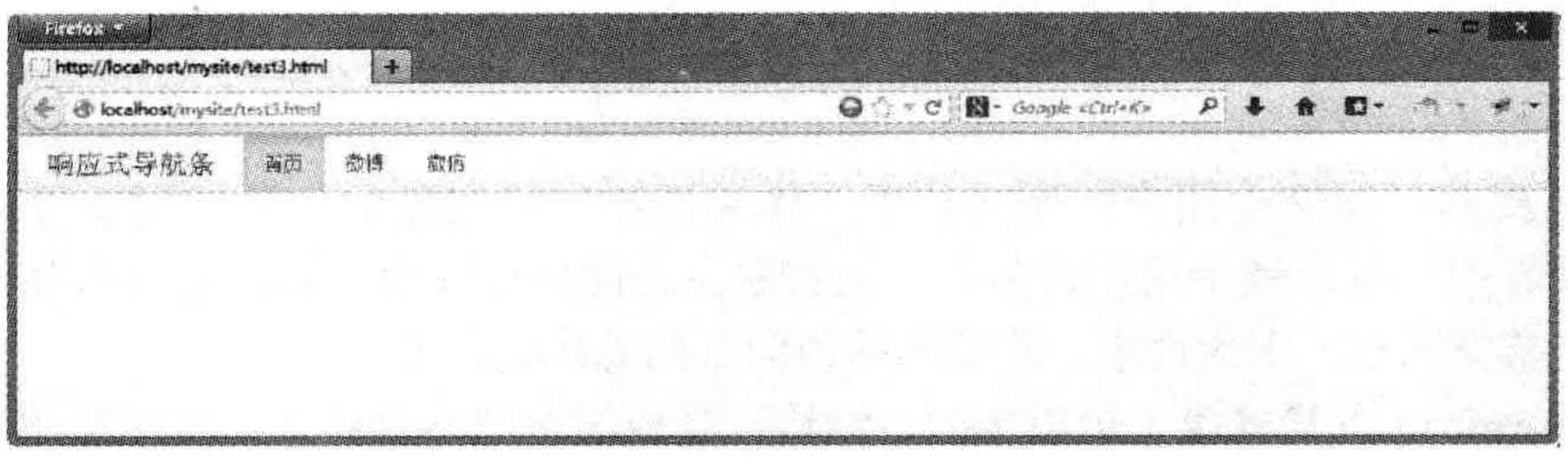
响应式导航条能够根据窗口宽度自动调整导航条的显示状态。方法是为需要自动响应的导航框添加 nav-collapse 和 collapse 样式类，然后添加一个按钮，定义 btn-navbar 样式类，并为该按钮设置 data-toggle="collapse" 属性，激活响应式交互，同时使用 data-target=".navbar-responsive-collapse" 属性绑定与导航框之间的响应联系。

例如，在下面的示例中，首先导入响应式交互空间插件 bootstrap-collapse.js，同时导入响应式样式表 bootstrap-responsive.css。在页面结构中，沿用上一小节的示例部分，并添加其他导航选项。在浏览器中预览，然后不断调整浏览器窗口的宽度，则显示效果如图 6-46 所示。

bootstrap-responsive.css

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap-
  responsive.css">
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-collapse.js">
  </script>
```

```
</head>
<body>
<div class="navbar">
    <div class="navbar-inner">
        <a class="btn btn-navbar" data-toggle="collapse" data-target=".navbar-
          responsive-collapse">
           <span class="icon-bar"></span>
           <span class="icon-bar"></span>
          </a>
          <a class="brand" href="#">响应式导航条</a>
          <ul class="nav nav-collapse collapse navbar-responsive-collapse">
             <li class="active"><a href="#">首页</a></li>
             <li><a href="#">微博</a></li>
             <li><a href="#">微信</a></li>
          </ul>
    </div>
</div>
</body>
</html>
```



宽屏下显示效果

窄屏下显示效果

图 6-46　设计响应式导航条效果

6.6　面包屑和分页

当在网站内多个页面之间进行有序切换时，使用面包屑（breadcrumb）和分页组件是个不错的选择。面包屑组件类似于树权分支导航，从网站首页逐级导航到详细页，类似效果如图 6-47 所示。分页组件类似于标签页，可以快速在多页之间来回切换，类似效果如

图 6-48 所示。

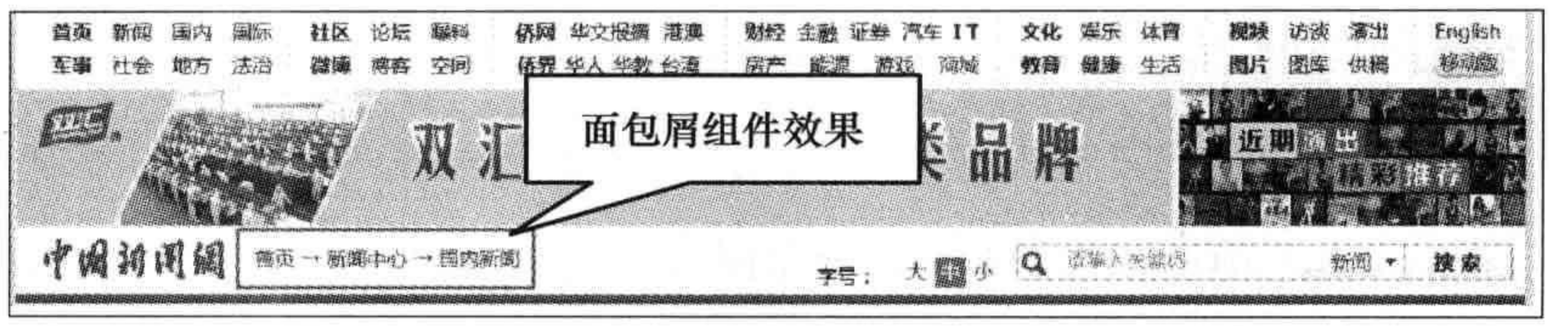


图 6-47　面包屑组件效果

图 6-48　分页组件效果

6.6.1　设计面包屑

面包屑揭示了网站中用户的所在位置。作为用户寻找路径的一种辅助手段，面包屑能方便定位和导航，可以减少用户返回上一级页面所需的操作次数。同时它具有临时性、动态性，占用屏幕空间小、干扰性小，降低网站访问者的总体跳出率。

使用 breadcrumb 样式类，可以把列表结构设计成面包屑导航样式。其用法比较简单，如在下面的示例中直接为 <ul> 标签添加 breadcrumb 类，然后使用 标签插入一个分隔符，即可设计出面包屑组件效果，如图 6-49 所示。

```
<ul class="breadcrumb">
    <li><a href="#"> 首页 </a> <span class="divider">→ </span></li>
    <li><a href="#"> 新闻频道 </a> <span class="divider">→ </span></li>
    <li><a href="#"> 国内新闻 </a> <span class="divider">→ </span></li>
    <li class="active"> 新闻详细页 </li>
</ul>
```

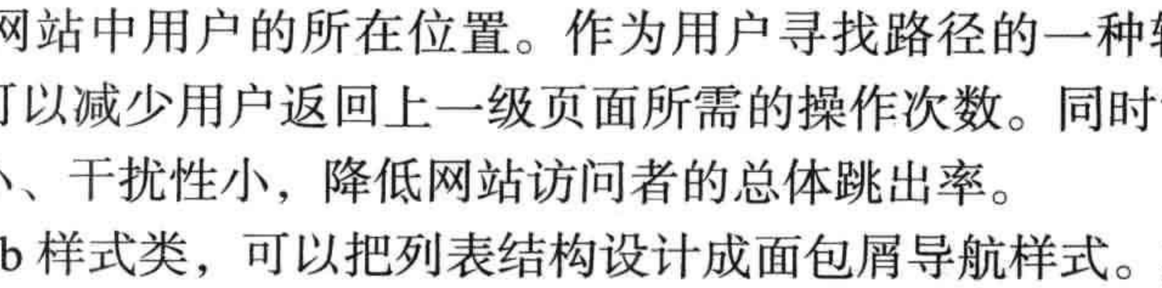


图 6-49　设计面包屑组件效果

面包屑是作为辅助和补充的导航方式，它能让用户知道在网站或应用中所处的位置并能方便地回到原来的地点。很多著名的互联网公司在建站之初就采用了面包屑导航做为网站产品线的“标准配置”，现在面包屑被越来越多的行业网站所认可及采用。其设计形式有三种。

（1）基于用户所在的层级位置

基于位置的面包屑用于告知用户在当前网站中所在的结构层级，常用在具有多级导航中，如图 6-47 所示。

（2）基于产品的属性

这种面包屑常出现在具有大量类别产品和服务的网站中，如电子商务、购物网等，如图 6-50 所示。

（3）基于用户的足迹

显示用户浏览的轨迹，面包屑之间没有明显的层级关系，只是展示用户从哪个级别过来的。这种面包屑在一级导航方案不明确的网站适合，其他情况不建议采用。

当用户从别处链接到网页，或者从搜索引擎查找到网页，则面包屑的存在能帮助用户快速了解当前的层级位置，并引导用户查看网站的其余部分，减少了看完直接跳走的用户数量。

6.6.2　设计分页组件

页码（pagination）也是非常常用的页面要素，Bootstrap 提供两种风格的页码组件。一个是多页面导航，用于多个页码的跳转，它具有极简主义风格的分页提示，能够很好地应用在结果搜索页面；另一种则是翻页，是轻量级组件，可以快速翻动上下页，适用于个人博客或者杂志。

使用 pagination 类可以设计标准的分页组件样式。例如，下面代码中使用 标签作为分页组件的包含框，包含列表结构框，演示效果如图 6-51 所示。

```
<div class="pagination">
    <ul>
        <li><a href="#">Prev</a></li>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">4</a></li>
        <li><a href="#">5</a></li>
        <li><a href="#">Next</a></li>
    </ul>
</div>
```

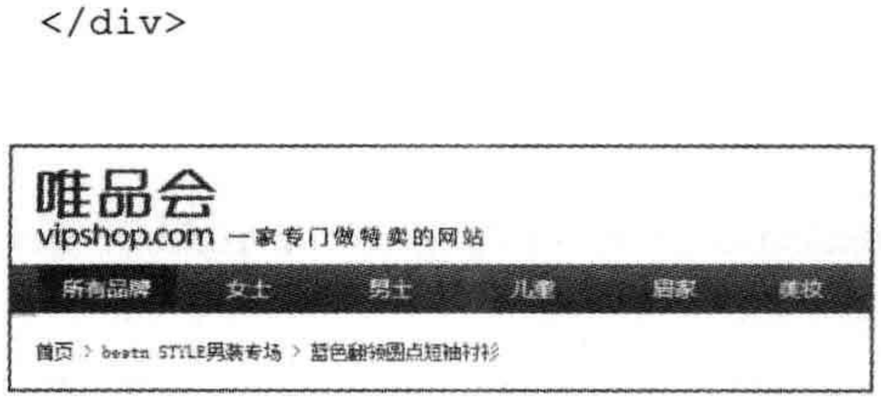


图 6-50　根据产品属性设计的面包屑导航效果

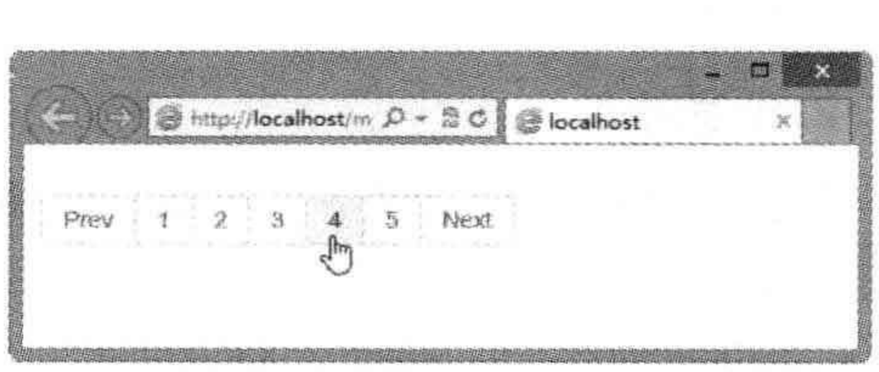


图 6-51　设计标准分页组件效果

注意　pagination 类只能够应用于列表框外包含框上。把 pagination 类直接添加到 <ul> 标签上（<ul class="pagination">）是无效的，此时的效果如图 6-52 所示。

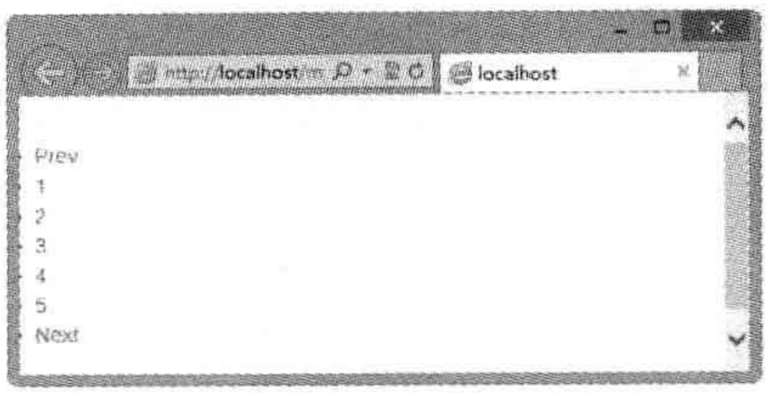


图 6-52　无效的 pagination 类引用

标准分组组件样式是一种简单的分页方式，适合 App 和搜索结果的展示。分页中的每一块都非常大，不易弄错，而且很容易扩展，并具有非常大的单击区域。

6.6.3　设置分页选项

分页组件提供了多个配置选项，以便根据页面布局效果对分组样式进行调整，主要包括分页按钮大小、分页组件对齐方式、激活或禁用按钮等。

1. 设置大小

分页组件按钮是可以调整大小的，Bootstrap 提供了 3 套尺寸供用户选择：

- pagination-large：大号分页按钮样式。
- pagination-small：小号分页按钮样式。
- pagination-mini：迷你型分页按钮样式。

例如，下面分别使用这三个样式类设置 3 套分页组件效果，效果比较如图 6-53 所示。

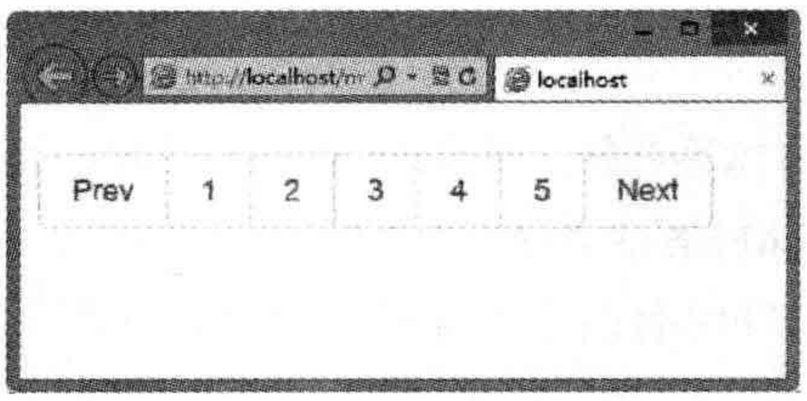
```
<div class="pagination pagination-large">
    <ul>
        <li><a href="#">Prev</a></li>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">4</a></li>
        <li><a href="#">5</a></li>
        <li><a href="#">Next</a></li>
    </ul>
</div>
```

2. 设置对齐

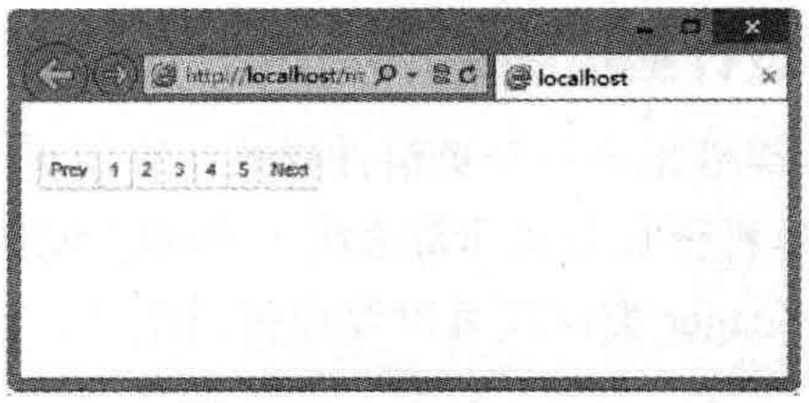
分页组件默认是左对齐，可以分别使用 pagination-centered 和 pagination-right 设置分页组件居中对齐和右对齐（见图 6-54）。例如，分别在上面示例中为

```
<div class="pagination pagination-large pagination-centered">
    <ul>
        <li><a href="#">Prev</a></li>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">4</a></li>
```

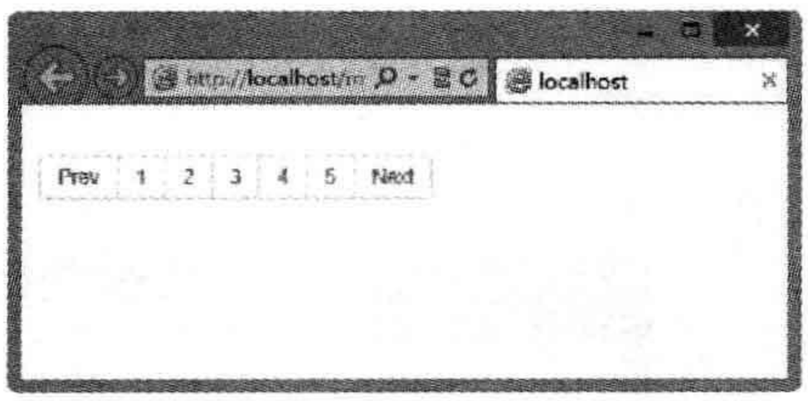
```
        <li><a href="#">5</a></li>
        <li><a href="#">Next</a></li>
    </ul>
</div>
```



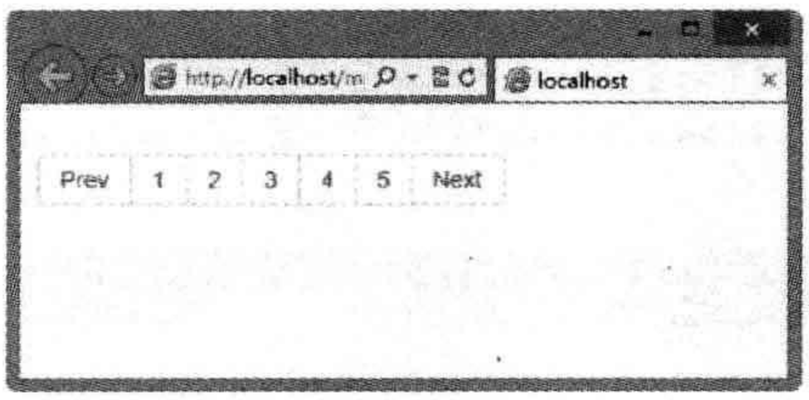
大号（pagination-large）



迷你型（pagination-mini）

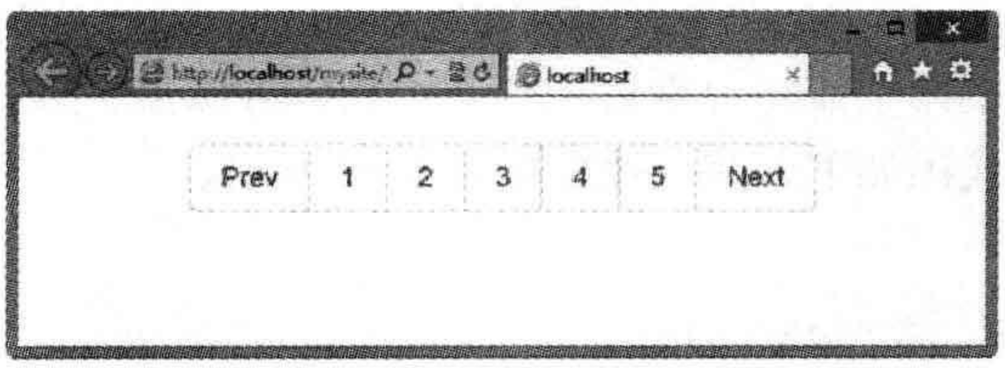


小号（pagination-small）

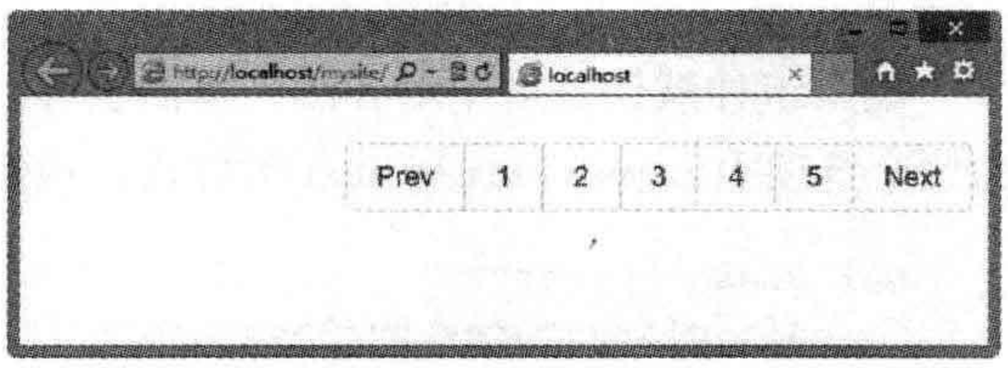


默认大小

图 6-53　分页按钮效果比较



居中（pagination-centered）



右对齐（pagination-right）

图 6-54　分页按钮对齐方式比较

3. 设置激活和禁用

在分页组件中，可以根据不同情况定制链接，如使用 disabled 样式类标明链接不可单击，而使用 active 标明当前页。

例如，针对上面的分页代码，分别为第 2 项引入 disabled 样式类，为第 4 项引入 antive 类演示，演示效果如图 6-55 所示。

```
<div class="pagination pagination-large pagination-right">
    <ul>
        <li><a href="#">Prev</a></li>
        <li class="disabled"><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li class="antive"><a href="#">3</a></li>
        <li><a href="#">4</a></li>
```

```
            <li><a href="#">5</a></li>
            <li><a href="#">Next</a></li>
        </ul>
    </div>
```

6.6.4 设计翻页组件

翻页组件是另类分页组件样式，它用更少的标签和样式来创建简单的“前一页”和“后一页”。这种分页方式非常适用于简单的网站，如博客或者杂志网站。

使用 pager 类样式可以设计翻页组件，该组件仅有两个列表项。例如，使用下面的代码可以快速设计一个翻页效果，如图 6-56 所示。

```
<ul class="pager">
    <li><a href="#">上一页</a></li>
    <li><a href="#">下一页</a></li>
</ul>
```

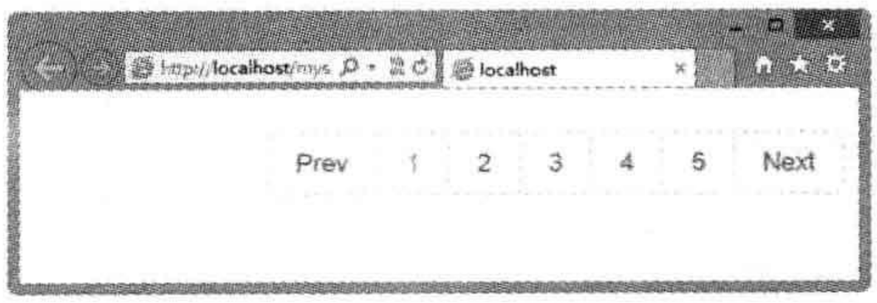


图 6-55 设计分页按钮禁用和激活状态

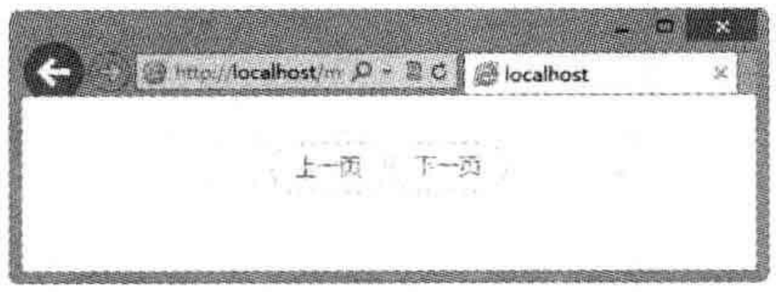


图 6-56 设计翻页效果

翻页组件默认是居中对齐的，当然也可以把两个按钮分别置于两侧。具体方法是：分别为两个选项引入 previous 和 next 类样式，示例代码如下，演示效果如图 6-57 所示。

```
<ul class="pager">
    <li class="previous"><a href="#">上一页</a></li>
    <li class="next"><a href="#">下一页</a></li>
</ul>
```

```
<ul class="pager">
    <li class="previous disabled"><a href="#">上一页</a></li>
    <li class="next"><a href="#">下一页</a></li>
</ul>
```

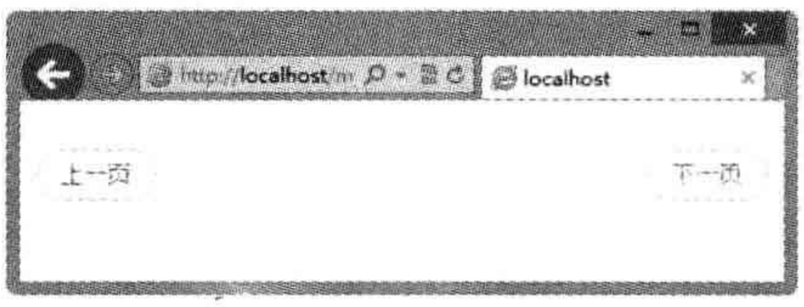


图 6-57 设计翻页按钮两端对齐效果

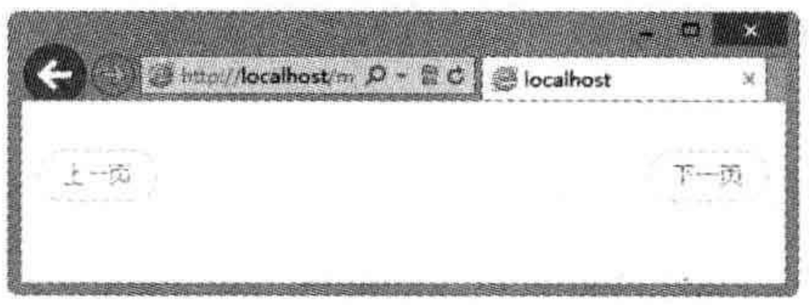


图 6-58 禁用翻页按钮效果

6.7　标签与徽章

标签和徽章可以作为页面装饰性或者提示性短语被广泛应用，在传统设计中标签和徽章主要通过图标来设计，如图 6-59 所示。

图 6-59　标签和徽章效果

标签是一个很好用的页面小要素，Bootstrap 具有多种颜色标签，表达不同的页面信息。只需要简单使用 label 样式类即可。徽章是细小而简单的组件，用于指示或者计算某种类别的要素，在 E-mail 客户端很常见，实际上在一些签到式的网站（LBS）上也常常用到。

与按钮的 btn 一样，Bootstrap 为标签和徽章设计了两套样式风格。标签样式通过 label 样式类实现，而徽章通过 badge 样式类实现。两套风格的基本外观相同，都是通过同一个样式实现：以行内块状显示，字体比较小，为 11.844 像素，字体加粗显示，文本适当添加一点阴影效果，字体颜色为白色，背景色为灰色。

```
.label,
.badge {
    display: inline-block;
    padding: 2px 4px;
    font-size: 11.844px;
    font-weight: bold;
    line-height: 14px;
    color: #ffffff;
    text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
    white-space: nowrap;
    vertical-align: baseline;
    background-color: #999999;
}
```

当然，两套风格也存在差异，标签呈现圆角矩形外观，而徽章呈现椭圆形外观。这种差异主要是通过 border-radius 属性实现的，通过设置不同的圆角值，达到区分外观的效果。

```
.label {
    -webkit-border-radius: 3px;
       -moz-border-radius: 3px;
            border-radius: 3px;
```

```
}
.badge {
    padding-right: 9px;
    padding-left: 9px;
    -webkit-border-radius: 9px;
       -moz-border-radius: 9px;
            border-radius: 9px;
}
```

例如，在 标签中，分别引入标签和徽章样式，则比较效果如图 6-60 所示。

```
<span class="label">标签样式</span>
<span class="badge">徽章样式</span>
```

在标签样式中，Bootstrap 提供了一套可选样式方案，说明如下。

- label-important：重要，通过醒目的视觉变化（深红色），提示浏览者注意阅读。
- label-info：信息，通过舒适的色彩设计（浅蓝色），调节默认的灰色视觉效果，可以用来替换默认按钮样式。
- label-success：成功，通过积极的亮绿色，表示成功或积极的动作。
- label-warning：警告，通过通用黄色，提醒应该谨慎操作。
- label-inverse：反向，通过黑色背景，表示特殊用途，一般不依赖于语义和用途。

例如，在下面的代码中分别引用这些附加标签样式，显示效果如图 6-61 所示。

```
<span class="label">默认</span>
<span class="label label-info">信息</span>
<span class="label label-important">重要</span>
<span class="label label-success">成功</span>
<span class="label label-warning">警告</span>
<span class="label label-inverse">反向</span>
```

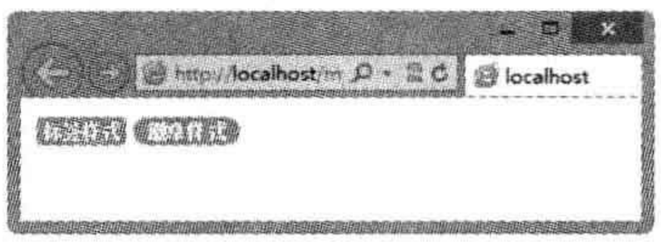


图 6-60　标签和徽章样式比较

图 6-61　设计标签多种风格

Bootstrap 为徽章也提供了一套可选样式方案，说明如下。

- badge-important：重要，通过醒目的视觉变化（深红色），提示浏览者注意阅读。
- badge-info ：信息，通过舒适的色彩设计（浅蓝色），调节默认的灰色视觉效果，可以用来替换默认按钮样式。
- badge-success：成功，通过积极的亮绿色，表示成功或积极的动作。
- badgel-warning：警告，通过通用黄色，提醒应该谨慎操作。
- badge-inverse：反向，通过黑色背景，表示特殊用途，一般不依赖于语义和用途。

例如，在下面的代码中分别引用这些附加徽章样式，显示效果如图 6-62 所示。

```
<span class="badge">默认</span>
<span class="badge badge-info">信息</span>
<span class="badge badge-important">重要</span>
<span class="badge badge-success">成功</span>
<span class="badge badge-warning">警告</span>
<span class="badge badge-inverse">反向</span>
```

图 6-62　设计徽章多种风格

注意　当标签和徽章元素内不包含任何文本，则 Bootstrap 会隐藏这些标签和徽章元素，以实现标签和徽章的折叠。

6.8　缩略图

缩略图（thumbnail）可以作为图片、视频、文字的网格结构展示。实现默认形式的 Bootstrap 缩略图，只需要简单的 thumbnails 标签。Thumbnails 多应用于图片、视频的搜索结果等页面，还可以链接到其他页面。它具有很好的可定制性，可以将文章片段、按钮等标签融入缩略图，同时可以混合与匹配不同大小的缩略图。

6.8.1　关于图像占位符

Bootstrap 支持图像占位符功能，在第一个版本中使用 Placehold（http://placehold.it/）工具来设计，类似于如下代码，效果如图 6-63 所示。

```
<img src="http://placehold.it/150x350">
<img src="http://placehold.it/250x150">
```

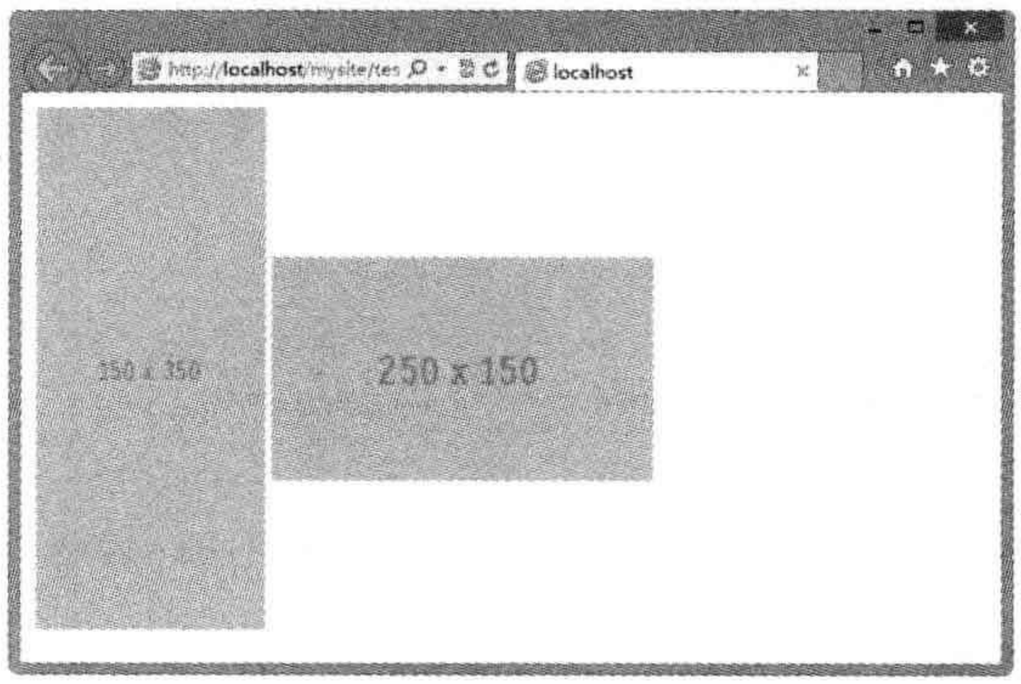


图 6-63　使用 Placehold 设计占位符

当 Bootstrap 2.2.2 发布之后，针对视网膜屏幕的资源，Bootstrap 将 Placehold.it 更换为 Holder.js（http://imsky.github.io/holder/）。Holder.js 也是一个针对客户端和视网膜屏幕的图像占位符工具，下载地址为 https://github.com/imsky/holder。

Holder 可以直接在客户端渲染图片的占位，支持在线和离线，提供一个链式 API 对图像占位进行样式处理。例如，在下面的示例中，先导入 holder.js 工具脚本文件，然后在图像中添加 data-src 自定义属性，设置 holder.js 的 URL 值，同时可设置图像的占位大小，如图 4-64 所示。

```
<script language="JavaScript" type="text/JavaScript" src="js/holder.js"></script>
<img src="images/1.jpg" data-src="holder.js/150x350" alt="">
<img src="images/2.jpg" data-src="holder.js/250x150" alt="">
```

图 6-64　使用 holder.js 设计占位符

6.8.2　设计缩略图

缩略图能够给图片、视频、文本等加入栅格功能，在 Bootstrap 1.4 版之前缩略图被叫做 media-grid，它很适合将图片、视频、图片搜索结果、商品列表等展示为栅格样式，这些对象可以是链接或纯粹的内容。

组成缩略图的标签很简单：使用 <ul> 标签包裹任意数量的 <li> 即可。它同样很灵活，只需添加少量标记即可包裹需要展示的任何内容。

首先，需要在 <ul> 标签中引入 thumbnails 样式类，指定当前列表框为缩略图集，然后在 <a> 标签中引入 thumbnail 类，设置当前超链接包含图像为缩略图效果。

例如，在下面的代码中，使用 thumbnails 为列表结构设计 4 个缩略图集合，效果如图 6-65 所示。

```
<ul class="thumbnails">
    <li class="span1">
        <a href="#" class="thumbnail"><img src="images/1.jpg"></a>
    </li>
    <li class="span2">
        <a href="#" class="thumbnail"><img src="images/2.jpg"></a>
    </li>
    <li class="span3">
        <a href="#" class="thumbnail"><img src="images/3.jpg"></a>
    </li>
```

```
    <li class="span4">
        <a href="#" class="thumbnail"><img src="images/4.jpg"></a>
    </li>
</ul>
```

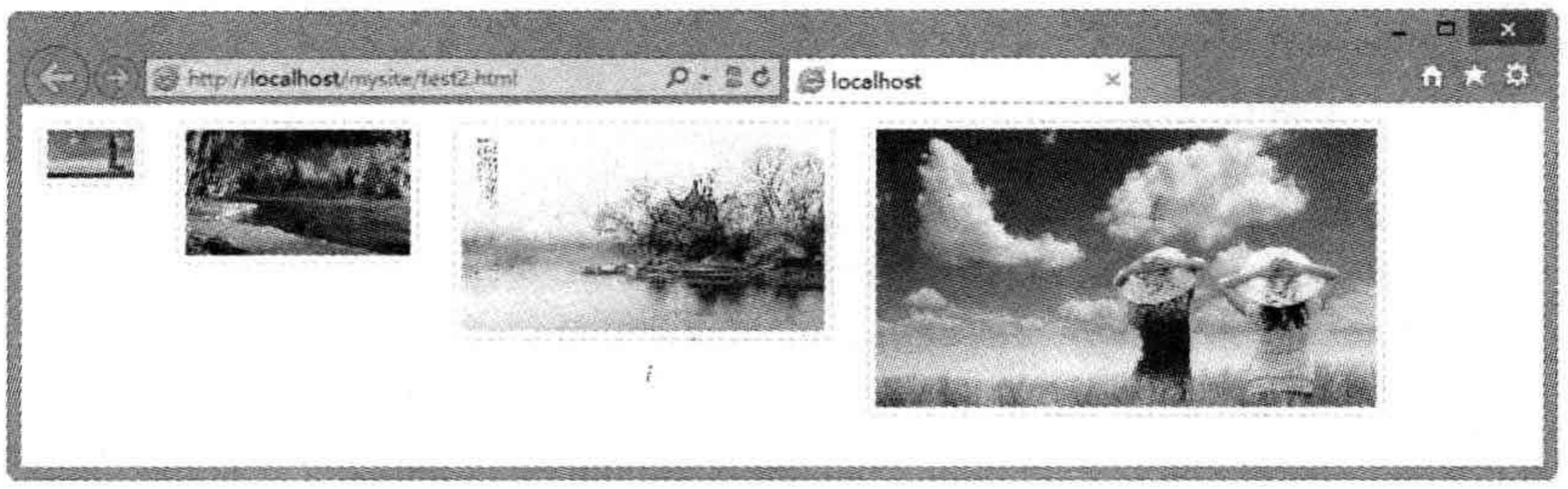


图 6-65　设计缩略图效果

在设计缩略图大小时，建议使用栅格中的列尺寸。例如，在上面的代码中缩略图组件使用现有的栅格系统中的类 span1 或 span3 等，用以控制缩略图的尺寸。

针对上面的示例，还可以使用图像占位符替换图像，以设计缩略图占位标识，演示效果如图 6-66 所示。

```
<ul class="thumbnails">
    <li class="span1">
        <a href="#" class="thumbnail"><img src="images/1.jpg" data-src="holder.
          js/260x180"></a>
    </li>
    <li class="span2">
        <a href="#" class="thumbnail"><img src="images/2.jpg" data-src="holder.
          js/260x180"></a>
    </li>
    <li class="span3">
        <a href="#" class="thumbnail"><img src="images/3.jpg" data-src="holder.
          js/260x180"></a>
    </li>
    <li class="span4">
        <a href="#" class="thumbnail"><img src="images/4.jpg" data-src="holder.
          js/260x180"></a>
    </li>
</ul>
```

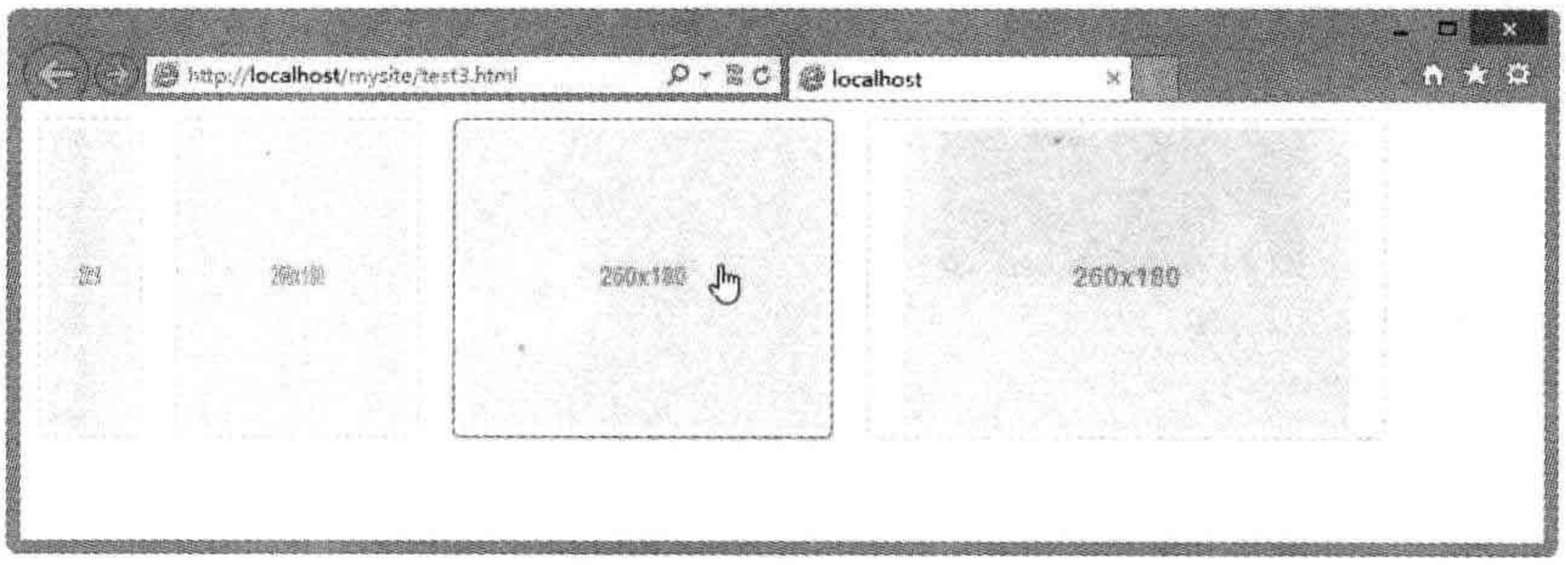


图 6-66　设计缩略图占位符效果

在缩略图中自定义 HTML 内容，标签的变化不大。为了放入块级内容，可以把 <a> 替换成 。例如，在下面示例中模拟演示了商品导购中的缩略图应用，效果如图 6-67 所示。

```
<ul class="thumbnails">
    <li class="span5">
        <div class="thumbnail">
            <a href="#"  class="thumbnail"><img src="images/11.jpg"></a>
            <h3> 卡帕 Kappa 女鞋专场 </h3>
            <p><span class="label label-info"> 剩余 </span>4 天 12 时 30 分 19 秒 </p>
            <span class="btn btn-success" > 品牌介绍 </span>
        </div>
    </li>
    <li class="span5">
        <div class="thumbnail">
            <a href="#"  class="thumbnail"><img src="images/12.jpg"></a>
            <h3> 韩都衣舍 HSTYLE 女上装专场 </h3>
            <p><span class="label label-info"> 剩余 </span>4 天 12 时 28 分 48 秒 </p>
            <span class="btn btn-success" > 品牌介绍 </span>
        </div>
    </li>
    <li class="span5">
        <div class="thumbnail">
            <a href="#"  class="thumbnail"><img src="images/13.jpg"></a>
            <h3>HAZZYS 男装专场 </h3>
            <p><span class="label label-info"> 剩余 </span>4 天 12 时 28 分 48 秒 </p>
            <span class="btn btn-success" > 品牌介绍 </span>
        </div>
    </li>
</ul>
```

图 6-67　设计缩略图页面应用效果

6.9　警告框

Bootstrap 2 重写了提示框的基础类，使用 alert 简化了原有的 alert-message 类。为了使组件更为简洁、实用，Bootstrap 去除了不同的提示块区域的外观，内容具有更多补白，同时可以显示更多文字。

6.9.1　设计警告框

使用 alert 类可以设计提示框组件，效果类似于 IE 浏览器的提示框，如图 6-68 所示。

图 6-68　IE 浏览器的警告框效果

例如，在下面的代码中为 标签引入 alert 类，并在警告框包含框中包含一个关闭按钮和一条提示信息，效果如图 6-69 所示。

```
<div class="alert">
    <button type="button" class="close">&times;</button>
    <strong>警告！ </strong> 确定要删除当前信息?
</div>
```

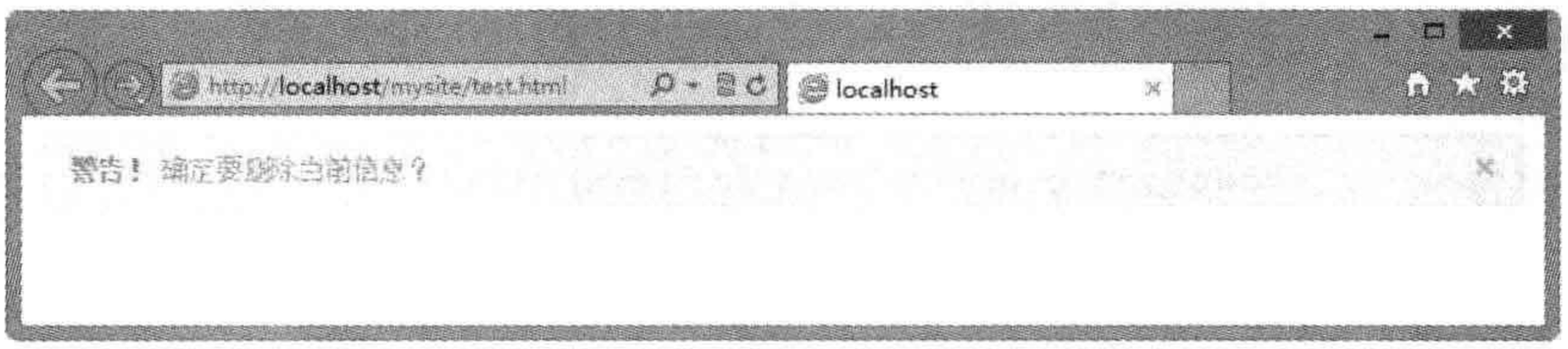


图 6-69　设计默认样式的警告框效果

通过添加其他类，可以改变警告框的语义，简单说明如下。

- alert-error：错误，浅红色背景，提示错误性信息。
- alert-danger：危险，浅红色背景，提示危险性操作。
- alert-success：成功，浅绿色背景，提示成功性操作，或者正确信息。
- alert-info：信息，浅蓝色背景，提示一般性信息。

例如，在下面的示例中，分别引用了警告框组件的不同类型的提示类，演示效果如图 6-70 所示。

```
<div class="alert alert-danger">
   <button type="button" class="close" data-dismiss="alert">&times;</button>
   <strong> 危险 </strong>
</div>
<div class="alert alert-error">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong> 错误 </strong>
</div>
<div class="alert alert-success">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong> 成功 </strong>
</div>
<div class="alert alert-info">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong> 信息 </strong>
</div>
```

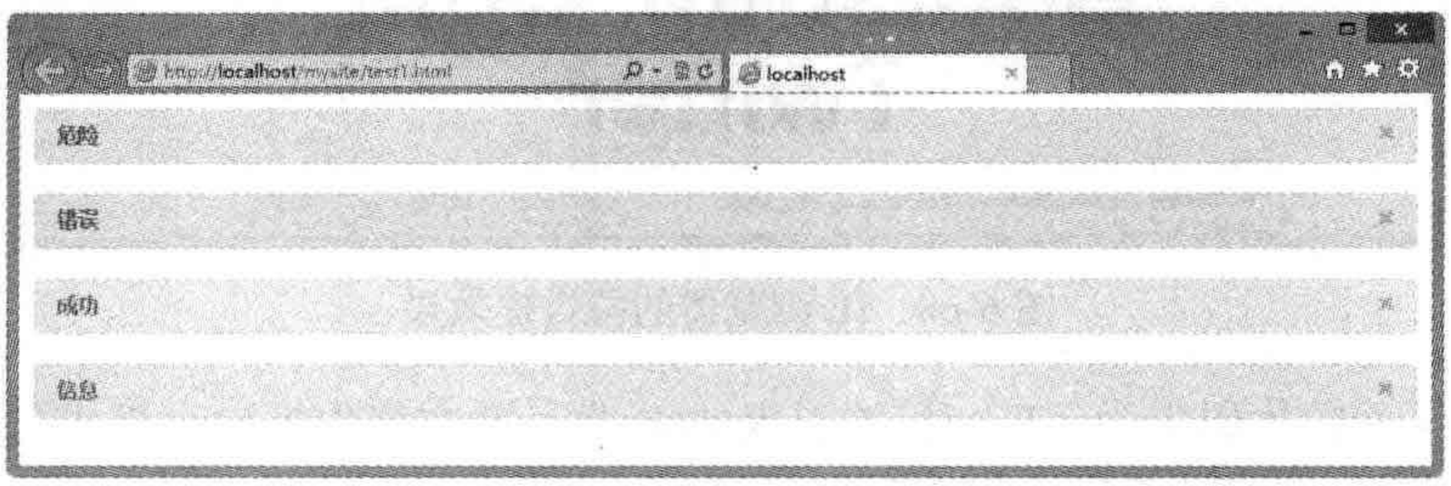


图 6-70　设计不同类型的警告框效果

如果提示信息比较多，需要占用很多区域，则可以通过添加 alert-block 类，以增加警告框上下方向的补白，让警告框看起来更大方。例如，为长提示信息条添加一个 alert-block 类，增加上下补白，避免警告框看起来很狭窄，如图 6-71 所示。

```
<div class="alert alert-info alert-block">
    <button type="button" class="close">&times;</button>
    <strong> 提示 </strong> 百度最近在香格里拉召开联盟大会，李彦宏发表了“中国互联网正在加速
      淘汰中国的传统产业，所有的互联网从业者应该推进这种进步”的号召。中国互联网一直分为娱乐和
      商业两条主线发展，前者讲究“得屌丝者得天下”，李彦宏是后者的领军人物之一。联盟大会剑指传统
      行业，行者解读为这是一篇向传统行业发起挑战的檄文。十二字诀“变局在即、舍我其谁、天下归君”。
</div>
```

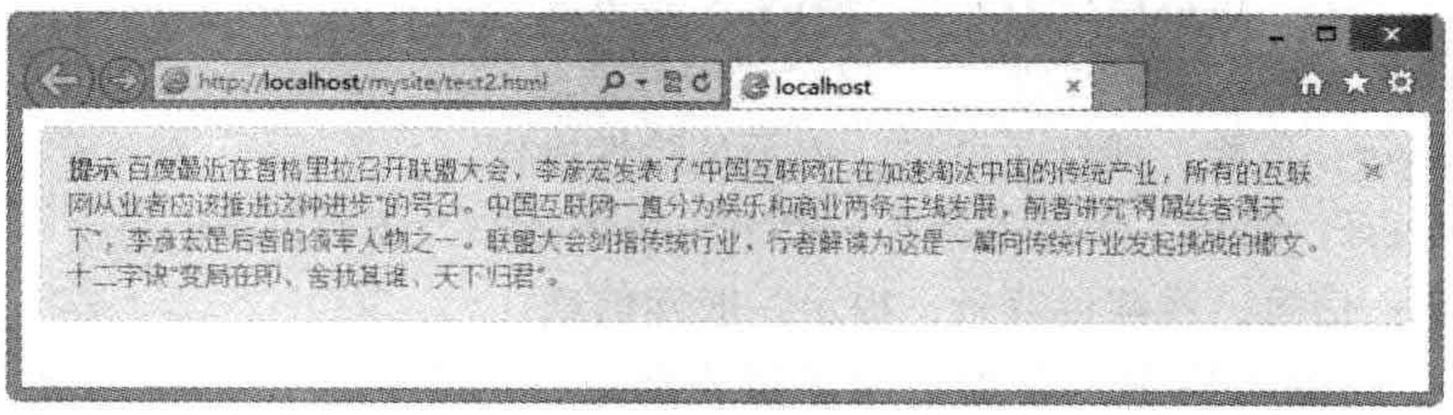


图 6-71　设计宽幅警告框效果

6.9.2　添加关闭按钮

通过为警告框添加关闭按钮，可以关闭警告框。在上一节示例中，已经添加了一个关闭按钮，但是该按钮不具备交互功能。

在 Safari 和 Opera 浏览器移动版上，当使用 <a> 标签关闭警告框时，除了添加 data-dismiss="alert" 属性外，还需要包含 href="#" 属性。

```
<div class="alert">
    <a href="#" class="close" data-dismiss="alert">&times;</a>
    <strong>警告！ </strong> 确定要删除当前信息?
</div>
```

或者使用带有 data 属性的 <button> 标签。当使用 <button> 时，必须包含 type="button" 属性，否则将无法执行提交动作。

```
<div class="alert">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong>警告！ </strong> 确定要删除当前信息?
</div>
```

如果希望通过 JavaScript 代码快速关闭警告框，则可以使用 Bootstrap 定义的警告框 jQuery 插件，然后与 jQuery 框架一起引入到页面中即可。完整示例代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-alert.js"></script>
</head>
<body>
<div class="alert">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong>警告！ </strong> 确定要删除当前信息?
</div>
</body>
</html>
```

6.10　进度条

进度条也是不可缺少的页面要素，在重定向、载入、跳转等页面动作中都经常用到，以提示当前正在执行的状态。Bootstrap 提供多种漂亮、简单、多种颜色的进度条。不过其中条纹和动画效果的进度条不支持 IE 浏览器，因为它使用了 CSS3 的渐变（Gradients）、透明度（Transitions）、动画效果（Animations）来实现它们的效果。IE7 ~ IE9 和旧版的 Firefox 都不

支持这些特性，所以在实现进度条的时候请注意浏览器支持程度。

6.10.1 设计进度条

进度条一般由嵌套的两层结构标签构成，外层标签引入 progress 类，用来设计进度槽，内层标签引入 bar 类，用来设计进度条。基本结构如下：

```
<div class="progress">
    <div class="bar" style="width:50%;"></div>
</div>
```

进度条默认样式是带有垂直渐变的进度条，进度槽显示为灰色，如图 6-72 所示。

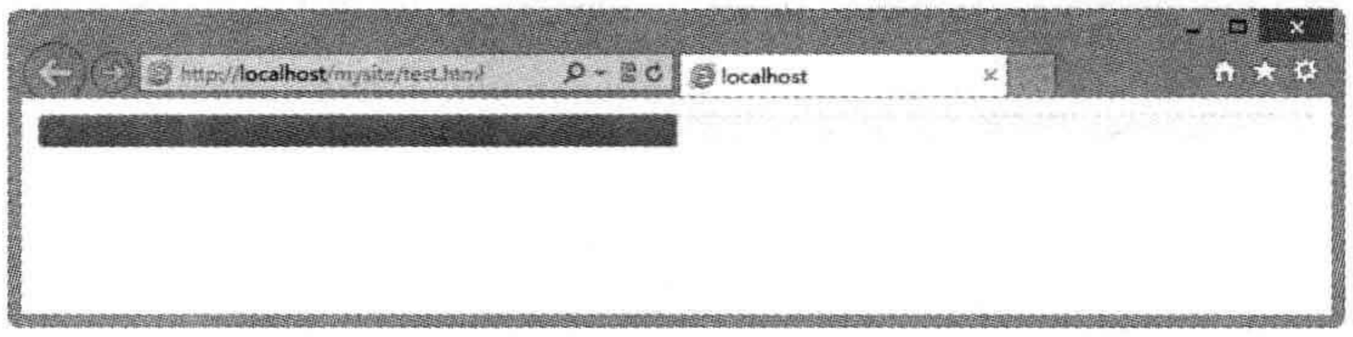


图 6-72 默认进度条样式效果

1. 设计条纹样式

条纹进度样式是通过 progress-striped 类实现的，它使用渐变创建的一个条纹效果的进度条，效果如图 6-73 所示，不支持 IE7、IE8。

```
<div class="progress progress-striped">
    <div class="bar" style="width:60%;"></div>
</div>
```

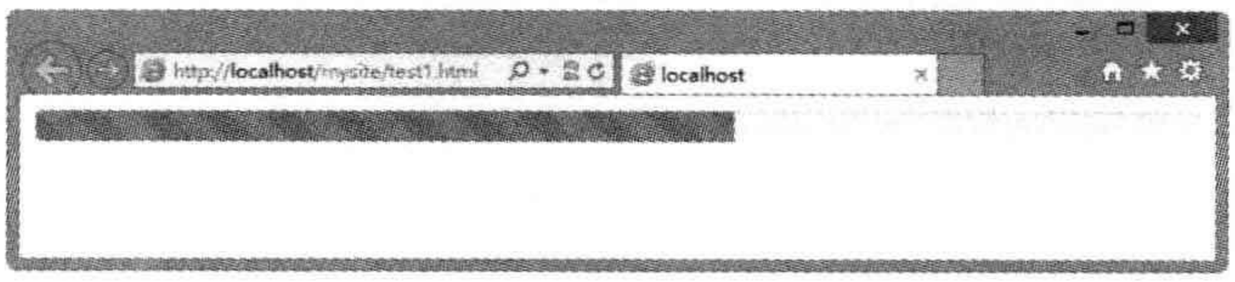


图 6-73 斑马条纹进度条样式效果

2. 设计动态条纹样式

如果为 progress-striped 添加 active 类样式，即可创建一个从右向左变化的条纹样式，效果如图 6-74 所示。IE 全系列都不支持此效果。

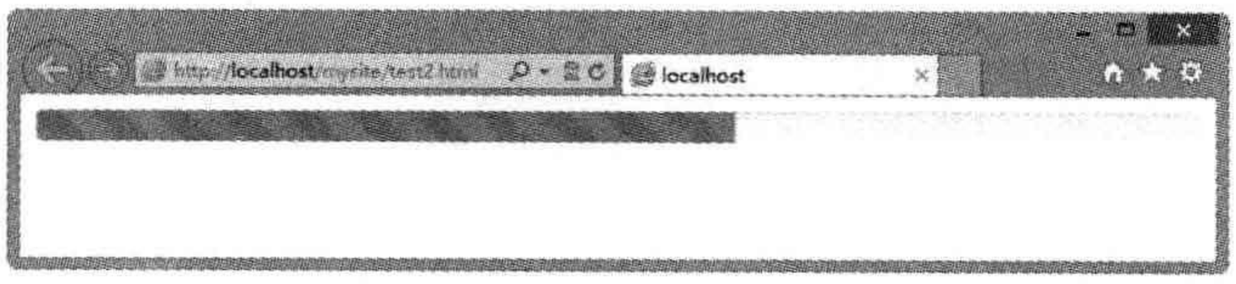


图 6-74 动态进度条样式效果

6.10.2　设置个性进度条

与警告框一样，进度条也允许通过添加其他类，改变进度条的背景效果，简单说明如下。

- progress-info：浅蓝色背景。
- progress-success：浅绿色背景。
- progress-warning：浅黄色背景。
- progress-danger：浅红色背景。

例如，在下面的示例中，分别引用进度条组件的不同类型的提示类，演示效果如图 6-75 所示。

```
<div class="progress progress-info">
    <div class="bar" style="width:40%;"></div>
</div>
<div class="progress progress-success">
    <div class="bar" style="width:50%;"></div>
</div>
<div class="progress progress-warning">
    <div class="bar" style="width:60%;"></div>
</div>
<div class="progress progress-danger" >
    <div class="bar" style="width:70%;"></div>
</div>
```

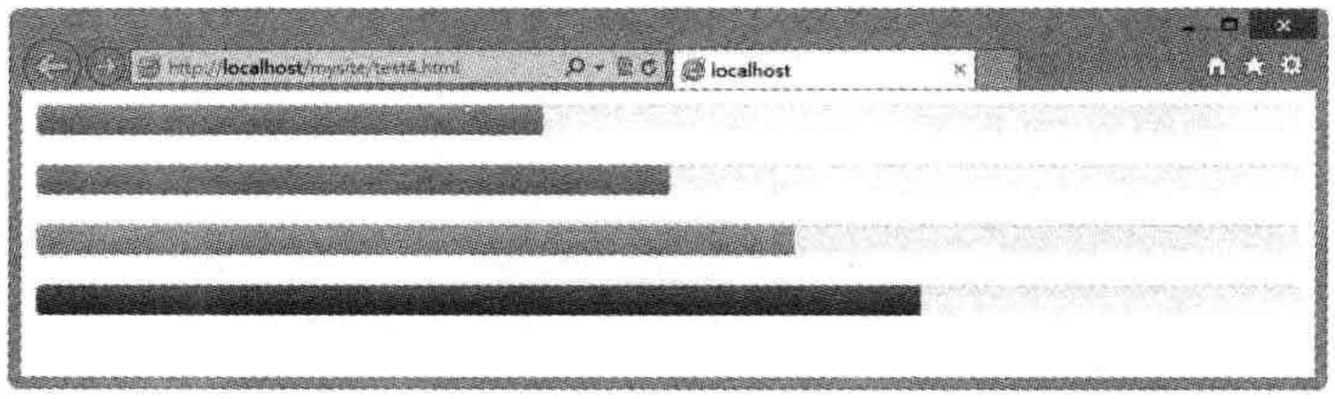



图 6-75　设计不同背景效果的进度条

如果为这些彩色进度条引入 progress-striped 样式类，可以设计彩色条纹效果，如图 6-76 所示。

```
<div class="progress progress-info progress-striped">
    <div class="bar" style="width:40%;"></div>
</div>
<div class="progress progress-success progress-striped">
    <div class="bar" style="width:50%;"></div>
</div>
<div class="progress progress-warning progress-striped">
    <div class="bar" style="width:60%;"></div>
</div>
<div class="progress progress-danger progress-striped" >
    <div class="bar" style="width:70%;"></div>
</div>
```

也可以把多个进度条放置于同一个进度槽中，设计一种堆叠样式，此时可以分别为每个进度条设计不同的背景样式，演示效果如图 6-77 所示。

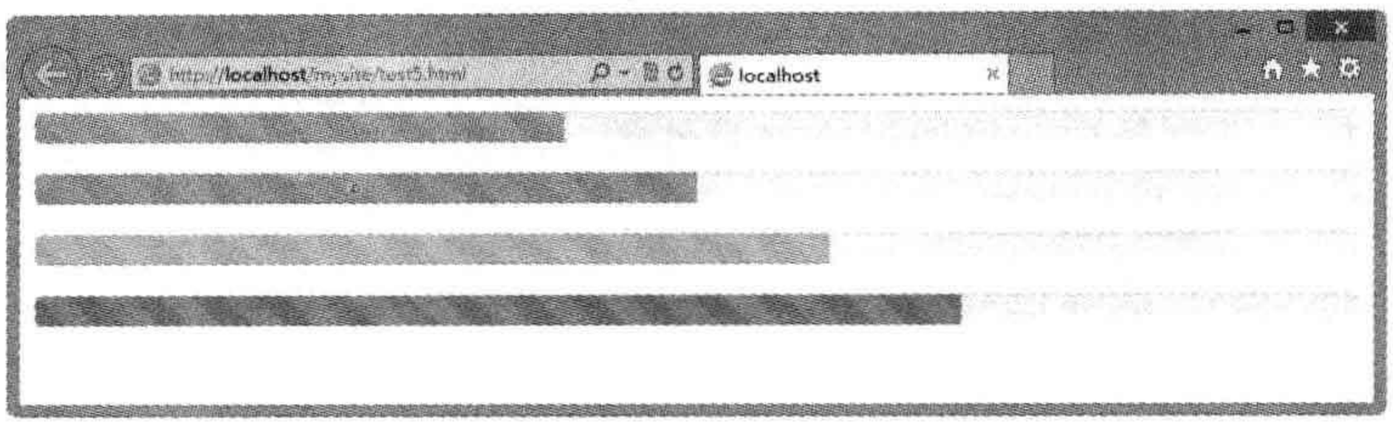


图 6-76　设计不同背景效果的条纹进度条

```
<div class="progress">
    <div class="bar bar-info" style="width:10%;"></div>
    <div class="bar bar-success" style="width:20%;"></div>
    <div class="bar bar-warning" style="width:30%;"></div>
    <div class="bar bar-danger" style="width:40%;"></div>
</div>
```

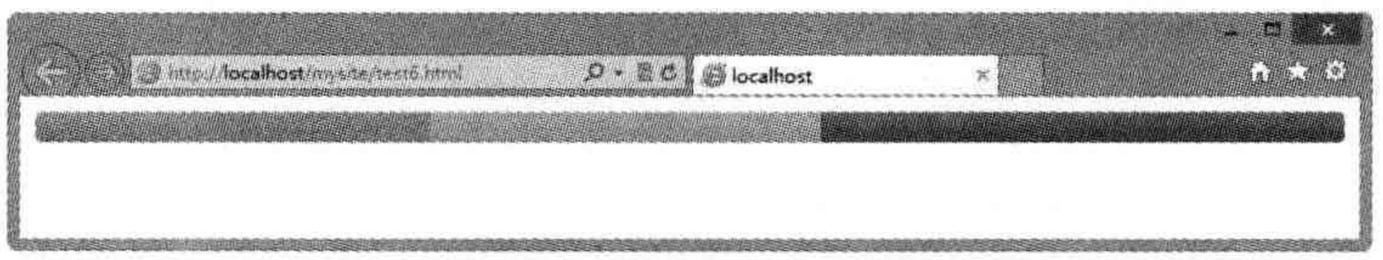


图 6-77　设计进度条堆叠效果

如果为进度条外框引入 progress-striped，则可以设计条纹堆叠样式（ ），如图 6-78 所示。

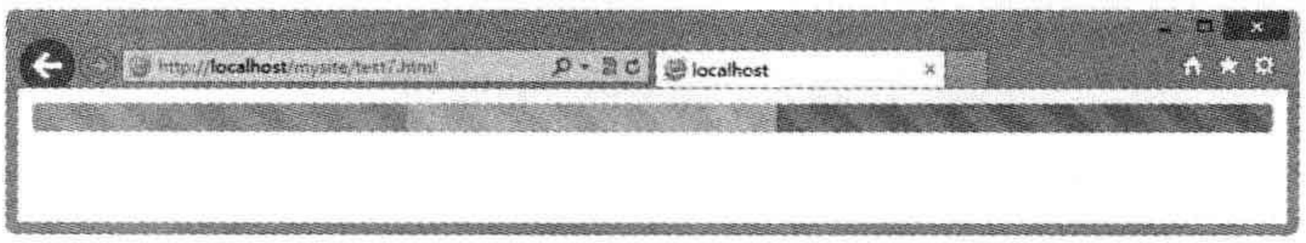


图 6-78　设计进度条条纹堆叠效果

6.11　媒体

媒体对象是一类特殊版式的区块样式，用来设计图文混排或者多媒体与文本混排的效果。作为抽象的结构样式，媒体对象可用于构建不同类型的组件，设计具有在文本内容的左或右对齐的图片。

6.11.1　媒体版式

在默认情况下，媒体对象组件的默认样式是在内容区域的左侧或右侧浮动一个媒体对象，如图片、视频、音频、Flash 动画等。

构件媒体对象组件需要 3 个类样式，具体说明如下：

- media：创建媒体对象组件包含框。

- media-object：定义媒体对象，如图片、视频、音频、Flash 动画。
- media-body：定义媒体对象的正文区域。在该区域可以使用 media-heading 定义媒体对象组件的正文标题。

例如，下面设计一个科技新闻报道，其中媒体对象是新闻焦点图，具体代码如下，演示效果如图 6-79 所示。

```
<div class="media"> <a class="pull-left" href="#"> <img class="media-object"
src="images/nayun.png"> </a>
   <div class="media-body">
      <h2 class="media-heading">马云：金融业要服务 80% 以前没有被服务好的客户，需要新的
       思想与技术 </h2>
      <div class="media">
         <p> 在今天举行的“外滩国际金融峰会”上，复星老板郭广昌在致辞里说，他希望问同来开会
             的马云：互联网金融和金融互联网，到底给我们带来哪些机会、带来哪些挑战。</p>
         <p> 这个问题太大，马主席肯定也不会现场具体做答。在事情没做之前就大声嚷嚷、指点大家
             应该怎么去做的，是老师，不是商人。马云当过老师，但归根结底是商人。所以他的公开
             讲话经常是这样的：给世界提出问题，但并不会给出答案——哪怕他其实已在暗中埋头答题。</p>
         <p> 他做过的电商是这样，眼下正在做的物流骨干网是这样（你能清晰知道菜鸟具体在做哪些
             工作吗？不知道），阿里参与的金融领域诸多事宜也是这样。</p>
         <p> 郭广昌们得不到具体的答案不要紧，这并不妨碍我们去听下马云对金融世界提出的问题：
             “如何能服务好剩下的那 80% 的客户？”马云认为，凭着互联网思想与技术，这个在传统金
             融业里难以求解的问题是可以求解出来的。</p>
       </div>
      </div>
</div>
```


图 6-79　设计媒体对象组件效果

6.11.2　媒体列表

Bootstrap 为媒体对象提供了列表结构，通过引入 media-list 样式类，可以设计媒体对象列表效果。在媒体对象列表结构中，每个列表项目都是一个独立的媒体对象组件，此时用户可以套用上面示例的结构。媒体对象列表在评论或文章列表页面中应用比较广泛，也比较实用。

例如，下面的示例演示了如何使用媒体对象定义列表信息，该示例是列表页典型样式，效果如图 6-80 所示。而 6.11.1 节示例是媒体对象的详细页。

```
<ul class="media-list">
    <li class="media"> <a class="pull-left" href="#"> <img class="media-object"
     src="images/31.jpg"> </a>
        <div class="media-body">
            <h4 class="media-heading">信息图表：IT 领域的发展和工作的变迁 </h4>
            <div class="media">
                <p> 过去几十年在科技领域发生了翻天覆地的变化，虎嗅编译了这张微软服务器和云计算
                    博客制作的信息图表，让我们来看一下自 IT 业诞生以来，这个行业都发生了些什么。</p>
                <p>Microsoft 2013-06-02 17:38</p>
            </div>
        </div>
    </li>
    <li class="media"> <a class="pull-left" href="#"> <img class="media-object"
     src="images/32.jpg"> </a>
        <div class="media-body">
            <h4 class="media-heading">交通银行董事长牛锡明这样回应马云与郭广昌 </h4>
            <div class="media">
                <p> 他指出：第一，互联网金融的发展，怎样符合监管，这个问题需要解决。第二个，
                    就是互联网金融的自我约束和自我风险控制机制，应该如何建立。</p>
                <p> 虎嗅 2013-06-02 16:34</p>
            </div>
        </div>
    </li>
</ul>
```


图 6-80　设计媒体对象列表效果

6.12　版式

第 4 章曾经讲解到一些排版技法，本节主要介绍到两个排版组件：Hero-unit 和 Page-header。Hero-unit 是一个轻量级的可扩展组件，主要用于市场推广网站显示大量的要素，而 Page-header 则用于简单地为页面段落的头部设置一个合适的空间和排版形式。

6.12.1　Hero 区块

Hero 区块是一个轻量、灵活的用于展示网站重点内容的组件，适合应用于营销类或内容类网站。设计 Hero 区块使用 hero-unit 类，代码如下，效果如图 6-81 所示。

```
<div class="hero-unit">
</div>
```

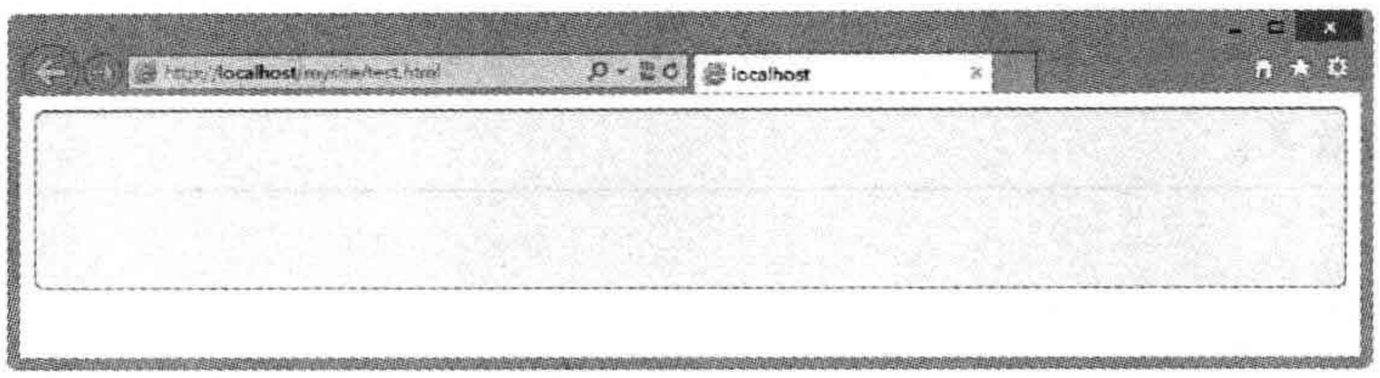


图 6-81　设计 Hero 区块

在 Hero 区块可以添加标题、说明性文本、导航按钮等，效果如图 6-82 所示。

```
<div class="hero-unit">
    <h1>Hero 标题 </h1>
    <p> 说明性文字 </p>
    <p><a class="btn btn-primary btn-large"> 更多 </a></p>
</div>
```

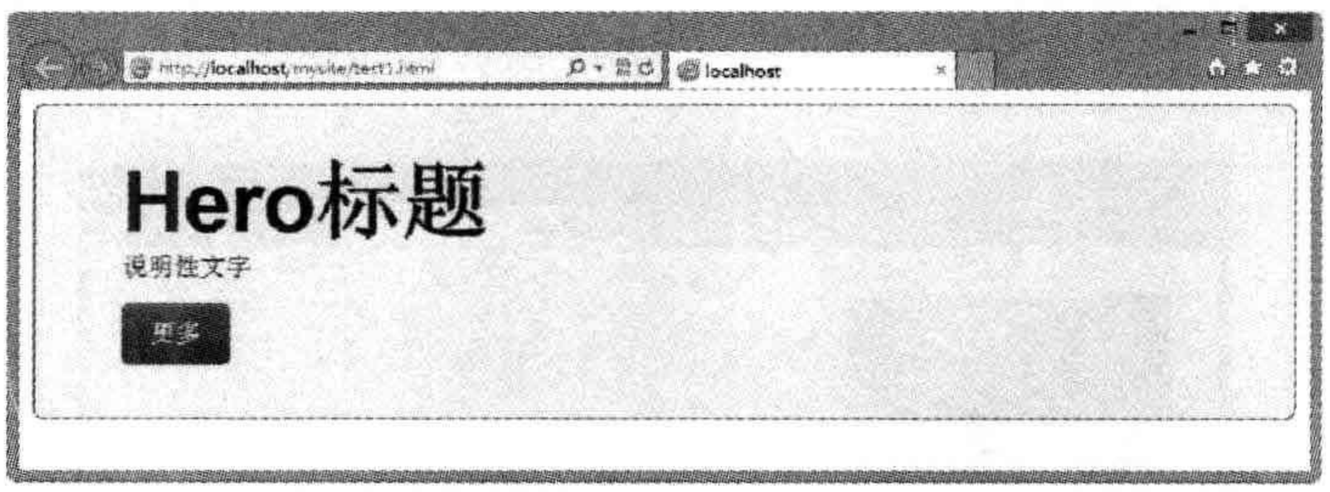


图 6-82　设计 Hero 区块效果

6.12.2　页面标题

Page-header 可以设计网页标题，它相当于一个标题框，可以给 <h1> 标签套上一个包含框，这样就可以为其增加间隔并从页面中分离出来，也可以在 <h1> 标签里增加 <small> 标签。演示效果如图 6-83 所示。

```
<div class="hero-unit">
    <div class="page-header">
    <h1> 网页标题 <small> 附加信息 </small></h1>
    </div>
</div>
```

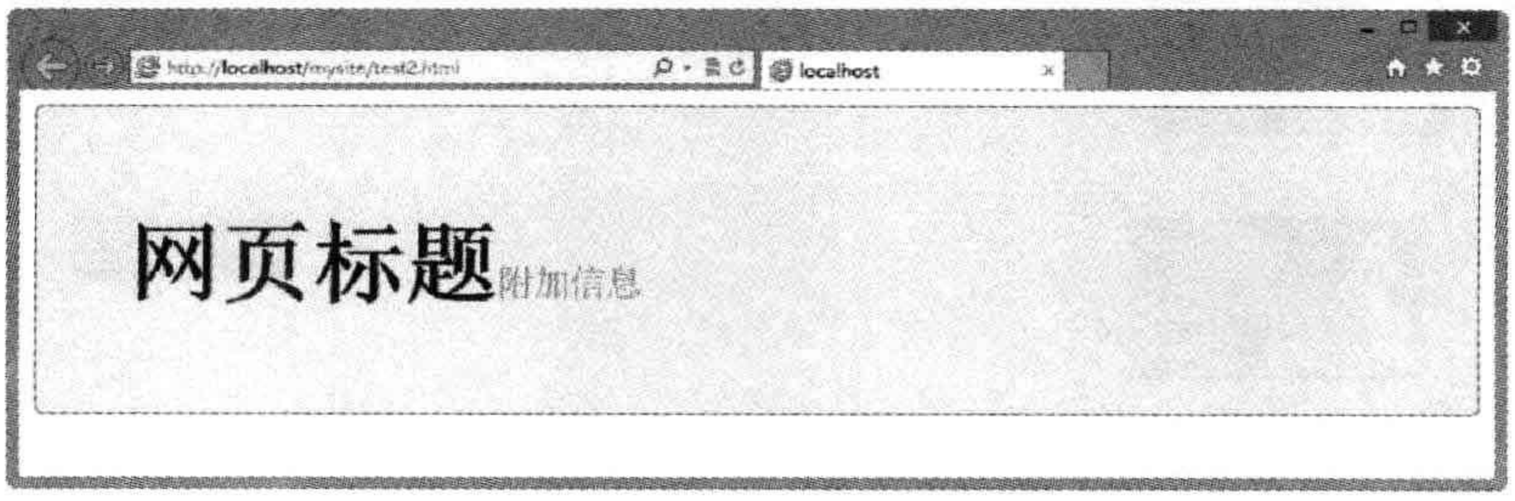


图 6-83　设计网页标题效果

提示 网页标题效果仅是一个范例，在实际应用中应该酌情添加额外的样式，以便设计出需要的统一的页面效果。

6.13 其他组件

Bootstrap 提供了很多小组件，这些组件在页面中都具有特定的角色和作用，下面分别进行介绍。

6.13.1 Well

Well 是一个小组件，用来设计一个内嵌容器，能够很好地包含指定对象或者页面内容。例如，把一幅图像包含在 Well 容器中，效果如图 6-84 所示。

```
<div class="well"><img src="images/4.jpg" style="width:200px;"> </div>
```

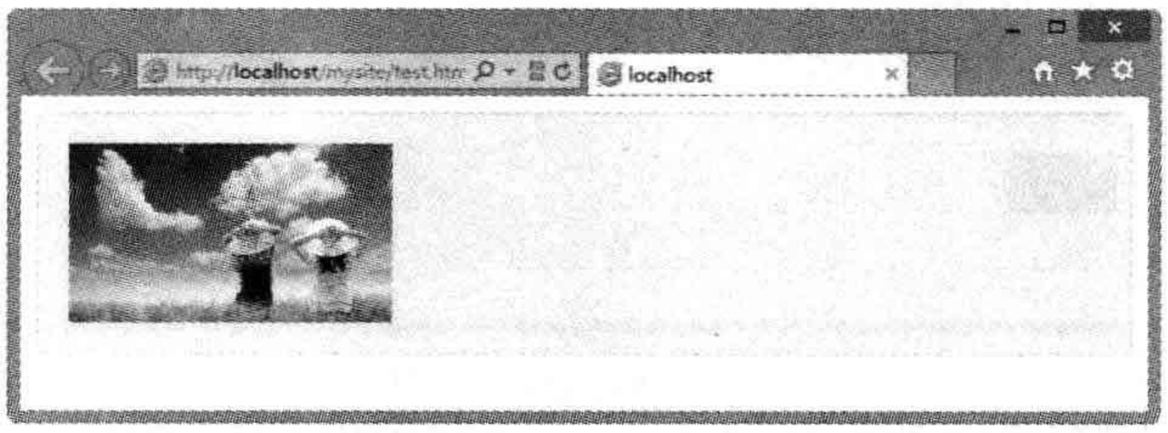


图 6-84 设计 Well 容器

通过在 Well 中添加 well-large 或者 well-small 可以调整 Well 容器的补白空间大小和圆角大小，比较效果如图 6-85 所示。

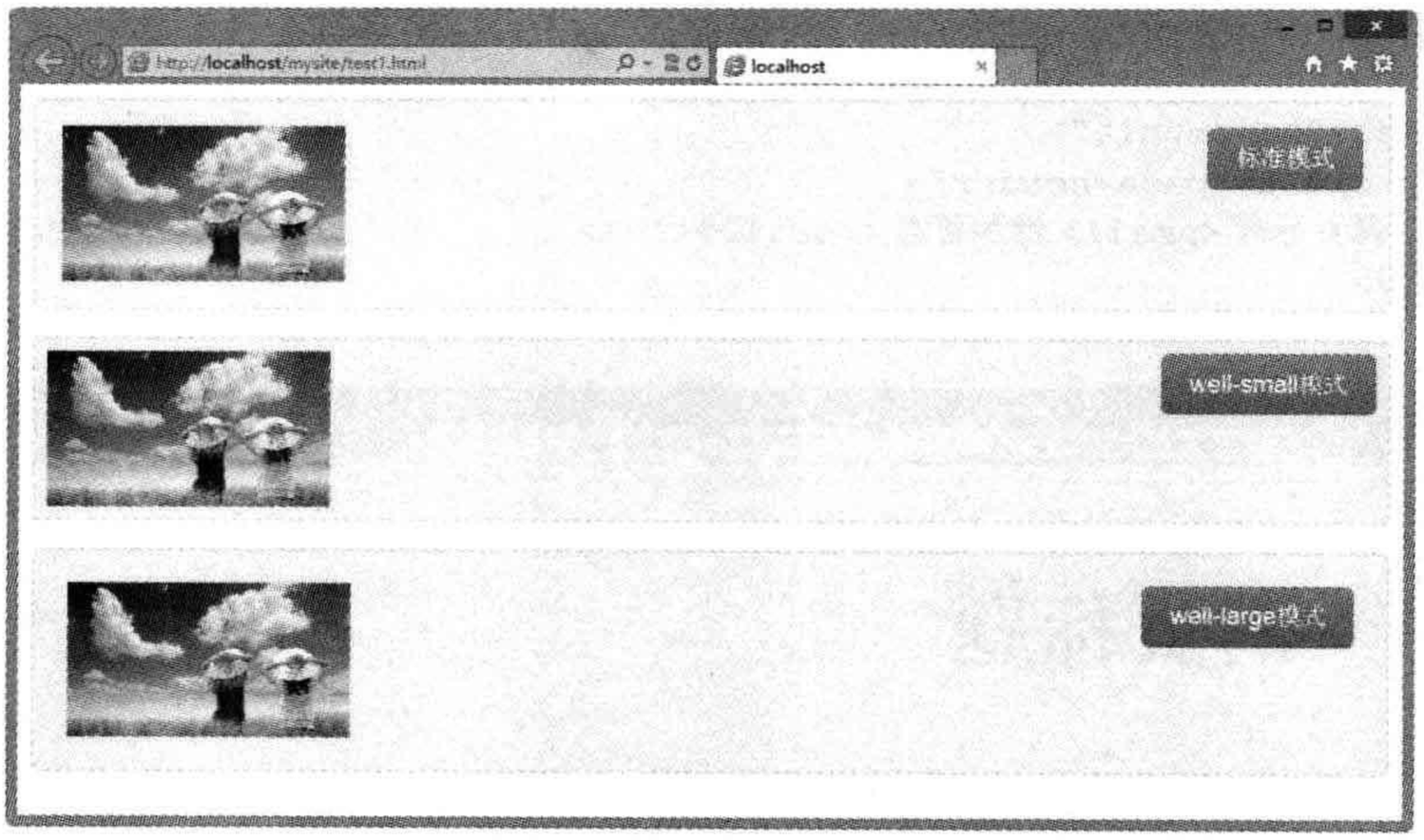


图 6-85 设计 Well 容器大小

```
<div class="well"><img src="images/4.jpg" style="width:200px;"><span class="btn
 btn-large btn-success pull-right">标准模式</span></div>
<div class="well well-small"><img src="images/4.jpg" style="width:200px;"><span
 class="btn btn-large btn-success pull-right">well-small 模式</span></div>
<div class="well well-large"><img src="images/4.jpg" style="width:200px;"><span
 class="btn btn-large btn-success pull-right">well-large 模式</span></div>
```

6.13.2　关闭图标

在警告框组件中，曾经介绍过关闭图标的应用，使用 close 样式类可以设计关闭图标按钮。该按钮主要应用于关闭对话框或警告框，效果如图 6-86 所示。

```
<button class="close">&times;</button>
```

如果选择使用连接标签的话，在 iOS 设备上需要一个 href="#" 配合单击事件。

```
<a class="close" href="#">&times;</a>
```

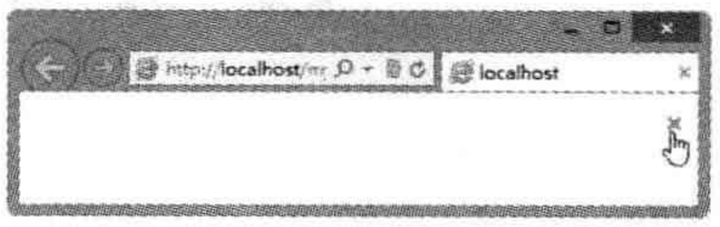


图 6-86　设计关闭图标按钮

6.13.3　辅助类

Bootstrap 提供大量公共样式类，用于小屏幕或调整行为的简单、功能单一样式类。这些样式类有的用于辅助设计，有的作为工具供用户使用，在前面几章中曾经详细介绍过一些，下面介绍 4 个常用的辅助类。

- pull-left：让页面元素向左浮动。
- pull-right：让页面元素向右浮动。
- muted：改变页面元素的颜色为 #999。
- clearfix：为任意页面元素清除浮动。

第 7 章

使用 Bootstrap 插件

本章内容

- JavaScript 插件开发概述
- 模态对话框
- 下拉项
- 滚动监听
- 标签页
- 工具提示
- 弹出提示
- 警告框
- 按钮
- 折叠
- 轮播
- 输入提示
- 附加导航

Bootstrap 包含了丰富的 Web 组件，如下拉菜单、按钮组、按钮下拉菜单、导航、导航条、面包屑、分页、排版、缩略图、对话框、进度条和媒体对象等。灵活利用这些内置组件，可以快速搭建外观漂亮、功能完备的网站。当然，组件还仅是静态对象，如果要让这些组件动起来，还需要 JavaScript 插件配合。简单地说，组件是框架和样式，而插件是交互行为和动态效果。

Bootstrap 自带了 13 个 JavaScript 插件，这些插件为 Bootstrap 组件赋予了生命，因此用户在学习使用组件的同时，还必须学习 Bootstrap 插件的使用。

7.1　JavaScript 插件开发概述

Bootstrap 内置了 13 种 JavaScript 插件，这些插件都建立在 jQuery 框架基础上，完全遵循 jQuery 使用规范和习惯，因为 Bootstrap 的插件实际上也是标准的 jQuery 插件。

这些插件都可以单独导入到页面中，不会相互影响，也可以一次性导入。注意，bootstrap.js 和 bootstrap.min.js 文件将所有插件包含在一个文件中了，前者是未压缩版，后者是压缩版。

下面列出了各个插件及对应的 js 文件：

- 过渡效果：bootstrap-transition.js
- 模态对话框：bootstrap-modal.js
- 下拉项：bootstrap-dropdown.js
- 滚动监听：bootstrap-scrollspy.js
- 标签页：bootstrap-tab.js
- 工具提示：bootstrap-tooltip.js
- 弹出提示：bootstrap-popover.js
- 警告框：bootstrap-alert.js
- 按钮：bootstrap-button.js
- 折叠：bootstrap-collapse.js
- 轮播：bootstrap-carousel.js
- 输入提示：bootstrap-typeahead.js
- 附加导航：bootstrap-affix.js

7.1.1　使用 Bootstrap 插件

Bootstrap 提供了两种调用插件的方法，一种是 data 定义法，另一种是 JavaScript 调用法。

1. data 调用

对于没有 JavaScript 基础的用户来说，这种方法是最实用、最方便的。我们只需要在页面中目标元素上定义 data 属性，就可以启用插件，且不用写一行 JavaScript 代码。这是 Bootstrap 中的一等 API，建议大家首选这种方式。

例如，要激活下拉菜单行为，只需要为控制对象定义 data-toggle 属性，设置属性值为"dropdown"，即可激活下拉菜单插件。

```
<a href="#" class="btn" data-toggle="dropdown">按钮</a>
```

data-toggle 是 Bootstrap 激活特定插件的专用自定义属性，它的值为对应插件的字符串名称。另外，大部分 Bootstrap 插件还需要 data-target 属性配合使用。data-target 也是一个 Bootstrap 自定义属性，用来指定控制对象，该属性值为一个 jQuery 选择符。

例如，在调用模态对话框时，除了定义 data-toggle="modal" 激活对话框插件，还应该使用 data-target="#myModal" 属性绑定对话框，告诉 Bootstrap 插件应该显示那个页面元素，"#myModal" 属性值匹配页面中对话框包含框 。

```
<button data-toggle="modal" data-target="#myModal" class="btn">打开对话框</button>
<div id="myModal" class="modal hide fade">模态对话框</div>
```

不同的插件可能还会支持其他 data 属性，具体请参阅后面章节说明。

注意 在某些特殊情况下，可能需要禁用这种默认动作。Bootstrap 提供了禁用 data 属性 API 的方式，通过解除绑定在 body 上的被命名为 "data-api" 的事件即可实现。代码如下：

```
$('body').off('.data-api')
```

还可以解除特定插件的事件绑定，只要将插件名和 data-api 链接在一起作为参数使用。代码如下：

```
$('body').off('.alert.data-api')
```

---

2. JavaScript 调用

除了使用 data 方法调用外，Bootstrap 插件也支持 JavaScript 调用。所有插件都可以单独或链式调用，与 jQuery 插件用法相同。

例如，针对上面的 data 调用示例，使用 JavaScript 调用的方法如下：

```
//显示下拉菜单
$(".btn").dropdown();
//显示模态对话框
$(".btn").click(function(){
      $("#myModal").modal();
})
```

当调用方法没有传递任何参数时，Bootstrap 将使用默认参数初始化此插件。

Bootstrap 插件定义的所有方法都可以接受一个可选的参数对象。例如，下面用法可以在打开模态对话框时取消遮罩层和快捷键控制。

```
$(".btn").click(function(){
      $("#myModal").modal({
          backdrop:false,
```

```
            keyboard:false
    });
})
```

Bootstrap 插件方法也可以接收特定意义的字符串。例如，下面代码将隐藏显示的模态对话框。

```
$("#myModal").modal('hide')
```

Bootstrap 插件允许使用 Constructor 属性访问插件构造函数：

```
$.fn.modal.Constructor
```

Bootstrap 插件也允许使用 data() 方法访问插件实例：

```
$('[rel= modal]').data('modal')
```

Bootstrap 插件支持与其他 UI 框架友好共存的方式。如果页面中发生命名空间冲突，可以通过调用插件的 noConflict() 方法恢复其原始值。例如，在同一个页面中导入了多个框架，它们都定义了 modal() 方法，这样就会发生冲突，出现相互覆盖问题。此时，我们可以使用下面的方法恢复 Bootstrap 的模态对话框插件。

```
var bootstrapmodal = $.fn.modal.noConflict()
$.fn.bootstrapmodal = bootstrapmodal
```

大部分 Bootstrap 插件都支持自定义事件。这些事件以英语动词原型和过去分词来表示。动词原形形式的（如 show）在事件执行之前触发，过去分词形式的（如 shown）在动作执行后触发。

7.1.2　过渡效果

Bootstrap 支持简单的过渡效果，需要在使用插件过程中同时导入 bootstrap-transition.js 文件，如果导入的是编译（或压缩）之后的 bootstrap.js 文件，就不再需要导入 bootstrap-transition.js 文件了，因为 bootstrap.js 文件已经包含了过渡效果。

bootstrap-transition.js 文件为 Bootstrap 其他 JavaScript 插件提供一个通用的特性检测。由于 CSS3 的限制，它提供的特效很有限，最常用的就是 fade。该文件的源代码如下：

```
!function ($) {
  "use strict";                                          // ecma262v5 引入的严格模式
  $(function () {
    $.support.transition = (function () {
      var transitionEnd = (function () {
        var el = document.createElement('bootstrap')     //创建一个自定义标签做测试
          , transEndEventNames = {                       //用于检测 CSS3 transition
                                                         //结束时的回调名
            'WebkitTransition'      : 'webkitTransitionEnd'
          , 'MozTransition'         : 'transitionend'
```

```
      , 'OTransition'              : 'oTransitionEnd otransitionend'
                                                    // Opera 支持效果不是很好
      , 'transition'               : 'transitionend'
      }
    , name
  for (name in transEndEventNames){
    if (el.style[name] !== undefined) {
      return transEndEventNames[name]
    }
  }
 }())
 return transitionEnd && {
   end: transitionEnd
 }
 })()
 })
}(window.jQuery);
```

Bootstrap 过渡效果主要应用于下面插件示例中：

- 具有幻灯片或淡入效果的对话框；
- 具有淡出效果的标签页；
- 具有淡出效果的警告框；
- 具有幻灯片效果的轮播。

7.2 模态对话框

JavaScript 支持 3 种内置对话框：window.alert()、window.confirm() 和 window.prompt()。当弹出一个对话框时，脚本执行将被中断，浏览器会暂停页面解析和任何交互，除非关闭对话框结束操作。上述三个方法中，alert() 能够创建一个简单的提示对话框，prompt() 方法可以接收用户输入的信息，confirm() 方法允许用户执行选择性操作。

JavaScript 提供的这三个对话框界面都比较简陋，功能单一，无法实用 Web 开发需要，为此 Bootstrap 重新打造了一个对话框插件，它提供了简洁、灵活的调用形式和样式，并具有精简的功能和友好的默认行为。

7.2.1 设计对话框

Bootstrap 对话框需要 bootstrap-modal.js 支持，因此在设计之前导入下面两个脚本文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-modal.js"></script>
```

或者直接导入 Bootstrap 综合脚本文件（bootstrap.js）：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
```

当然，Bootstrap 样式表文件也是必须加载的，它是 Bootstrap 框架的技术基础。

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

完成页面框架技术的初始化操作之后，就可以在页面中设计对话框文档结构，并为页面特定对象绑定触发行为，以便打开对话框。完整的页面代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
</head>
<body>
<a href="#myModal" class="btn" data-toggle="modal">弹出对话框</a>
<div id="myModal" class="modal hide fade">
     <h1>弹出对话框</h1>
     <p>这是弹出的对话框吗？</p>
</div>
</body>
</html>
```


在浏览器中预览该文档，单击“弹出对话框”按钮，将会看到如图 7-1 所示的弹出对话框。在对话框外面单击，即可自动关闭对话框，恢复页面的初始状态。

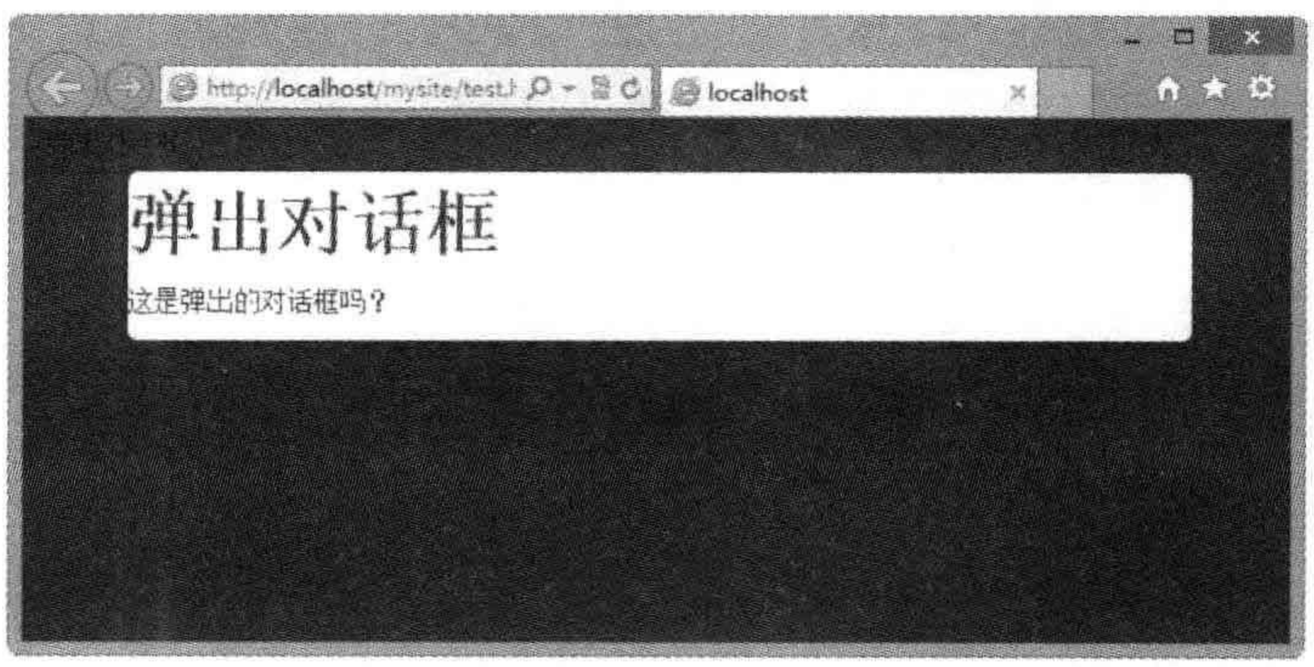


图 7-1　简单的弹出对话框

使用 modal 样式类可以定义弹出对话框的外框，如果引用 modal-header、modal-body 和 modal-footer 三个类样式，可以定义弹出对话框的标题区、主体区和脚注区。例如，针对上面的示例代码，为对话框增加结构设计，则效果如图 7-2 所示。

```
<a href="#myModal" class="btn" data-toggle="modal">弹出对话框</a>
<div id="myModal" class="modal hide fade">
     <div class="modal-header">
          <button class="close" data-dismiss="modal">×</button>
          <h3>标题</h3>
```

```
        </div>
        <div class="modal-body">
            <p> 正文 </p>
        </div>
        <div class="modal-footer">
            <button class="btn btn-info" data-dismiss="modal"> 关闭 </button>
        </div>
    </div>
```

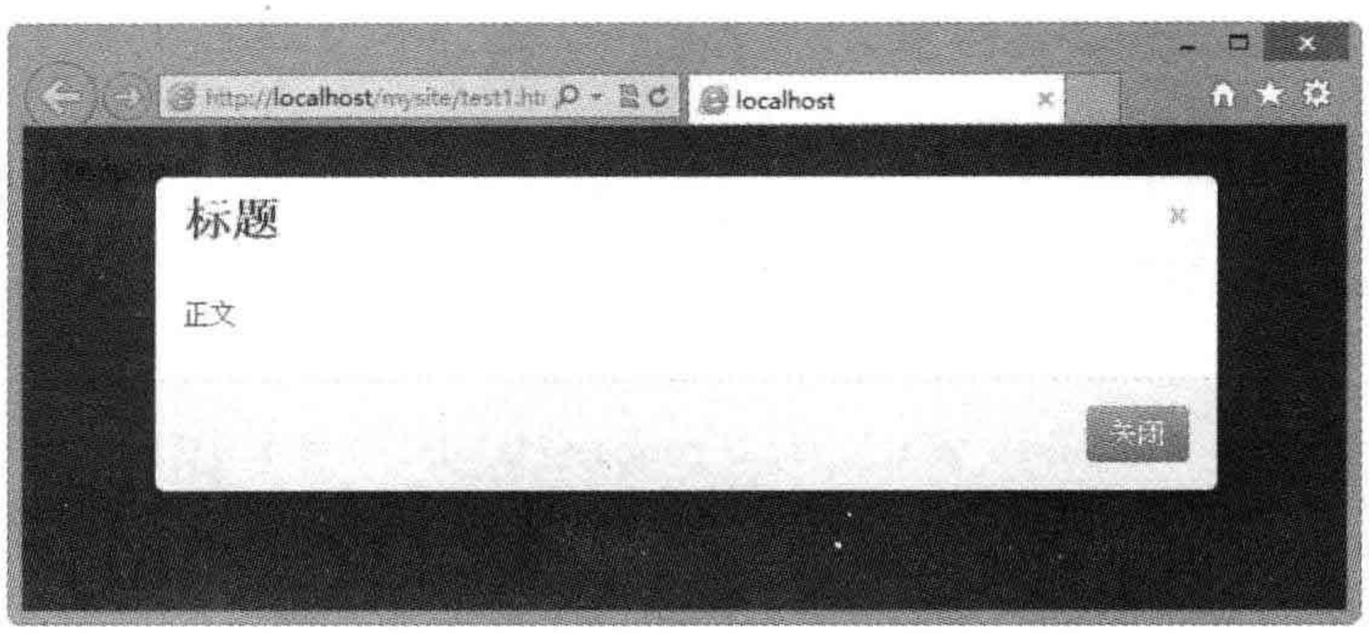


图 7-2　设计标准的弹出对话框样式

标准对话框中包含两个关闭按钮，一个是对话框右上角的关闭图标，另一个是页脚区域的关闭按钮。这两个关闭对话框的标签通过自定义属性 data-dismiss 触发对话框关闭行为，data-dismiss 属性值指定了要关闭的对话框组件。

打开对话框的行为通过 <a> 标签来实现，其中 href 属性通过锚记与对话框（ 标签）建立绑定关系，然后通过自定义属性 data-toggle 激活对话框显示行为，data-toggle 属性值指定了要打开对话框的组件。

7.2.2　调用对话框

调用对话框的方法有两种，下面简单进行介绍。

1. HTML 属性调用

在 7.1 节的示例中，我们看到了使用 data 属性调用对话框的一般方法。通过 data 属性，无须编写 JavaScript 脚本即可创建对话框。

定义激活元素时，必须注意两点：第一，使用 data-toggle 属性定义激活插件的类型，对于对话框插件来说，即设置为 data-toggle="modal"；第二，设置具体打开的目标对象。

当激活元素为按钮或者其他元素时，可以设置自定义属性 data-target 为对话框包含框的 ID 值，以绑定目标对象，指向某个将要被启动的对话框。

```
<button data-toggle="modal" data-target="#myModal" class="btn"> 打开对话框 </button>
```

当激活元素为超链接元素时，可以直接在 href 属性上设置对话框包含框的 ID 值，以锚点的形式绑定目标对象，以指向某个将要被启动的对话框。

```
<a href="#myModal" data-toggle="modal" class="btn">打开对话框</a>
```

2. JavaScript 调用

如果需要设计更复杂的对话框调用，则建议使用 JavaScript 调用。其实，JavaScript 调用也比较简单，直接使用 modal() 构造函数即可，其用法与 jQuery 插件用法高度一致。

例如，针对 7.2.1 节示例，为超链接 <a> 标签绑定 click 事件，当单击该按钮时，为对话框调用 modal() 构造函数。

```
<script type="text/JavaScript">
$(function(){
      $(".btn").click(function(){
            $("#myModal").modal();
      })
})
</script>
```

modal() 构造函数可以传递一个配置对象，该对象包含的配置属性说明如表 7-1 所示。

表 7-1　modal() 配置参数

名　称	类　型	默认值	描　述
backdrop	boolean	true	是否显示背景遮罩层，同时设置单击对话框其他区域是否关闭对话框 默认值为 true，表示显示遮罩层，当单击遮罩层时，会自动隐藏对话框和遮罩层
keyboard	boolean	true	是否允许 Esc 键关闭对话框，默认值为 true，表示允许使用键盘上的 Esc 键关闭对话框，当按 Esc 键时，快速关闭对话框
show	boolean	true	在初始状态是否显示对话框，默认值为 true，表示显示对话框
remote	path	false	设置一个远程 URL，Bootstrap 将利用 jQuery 加载该链接页面，并把响应的数据添加到对话框的 modal-body 包含框中。也可以使用下面方式加载远程数据： <a data-toggle="modal" href="remote.html" data-target="#modal">click me

注意，如果使用 HTML 属性调用对话框时，上面的选项也可以通过 data 属性传递给组件。对于 data 属性，将选项名称附着于 data- 字符串之后，类似于 data-backdrop=""。

例如，下面代码可以打开对话框，但是不显示遮罩层，同时取消了 Esc 键关闭对话框的操作，显示效果如图 7-3 所示。

```
<script type="text/JavaScript">
$(function(){
      $(".btn").click(function(){
            $("#myModal").modal({
                  backdrop:false,
                  keyboard:false
```

```
            });
        })
    })
    </script>
```

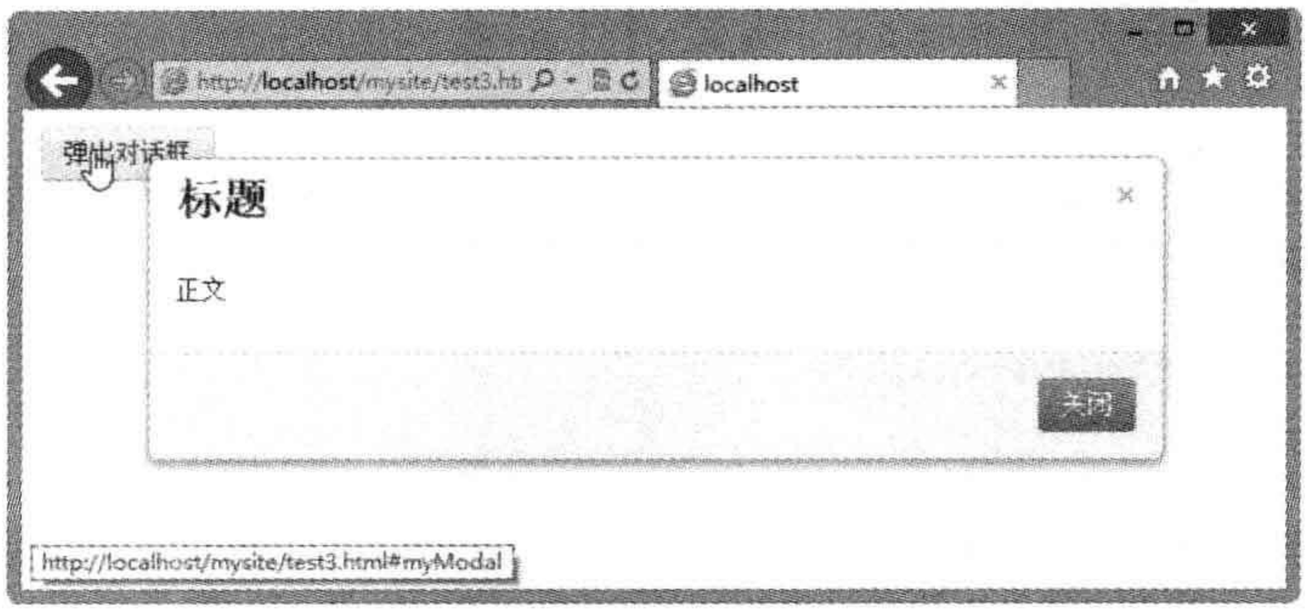


图 7-3　关闭遮罩层效果

针对上面代码传递的参数，在 HTML5 文档中可以使用如下方式实现相同的设计效果。

```
<button data-toggle="modal" data-backdrop="false" data-keyboard="false"  data-
   target="#myModal" class="btn">打开对话框</button>
```


7.2.3　应用对话框

modal() 构造函数也可以接收特定字符串参数，以方便手动控制对话框显示或者隐藏。简单说明如下：

- modal('toggle')：手动打开或隐藏一个对话框。
- modal('show')：手动打开一个对话框。
- modal('hide')：手动隐藏一个对话框。

例如，在页面初始化时，隐藏对话框的遮罩层，并显示对话框，然后使用按钮调用 modal() 方法，传递参数值为 toggle，则当单击该按钮时，可以显示或者隐藏对话框，如图 7-4 所示。

```
<script type="text/JavaScript">
$(function(){
     $("#myModal").modal({
          backdrop:false,
          show:true
     });
     $(".btn").click(function(){
          $("#myModal").modal("toggle");
   })
})
</script>
```

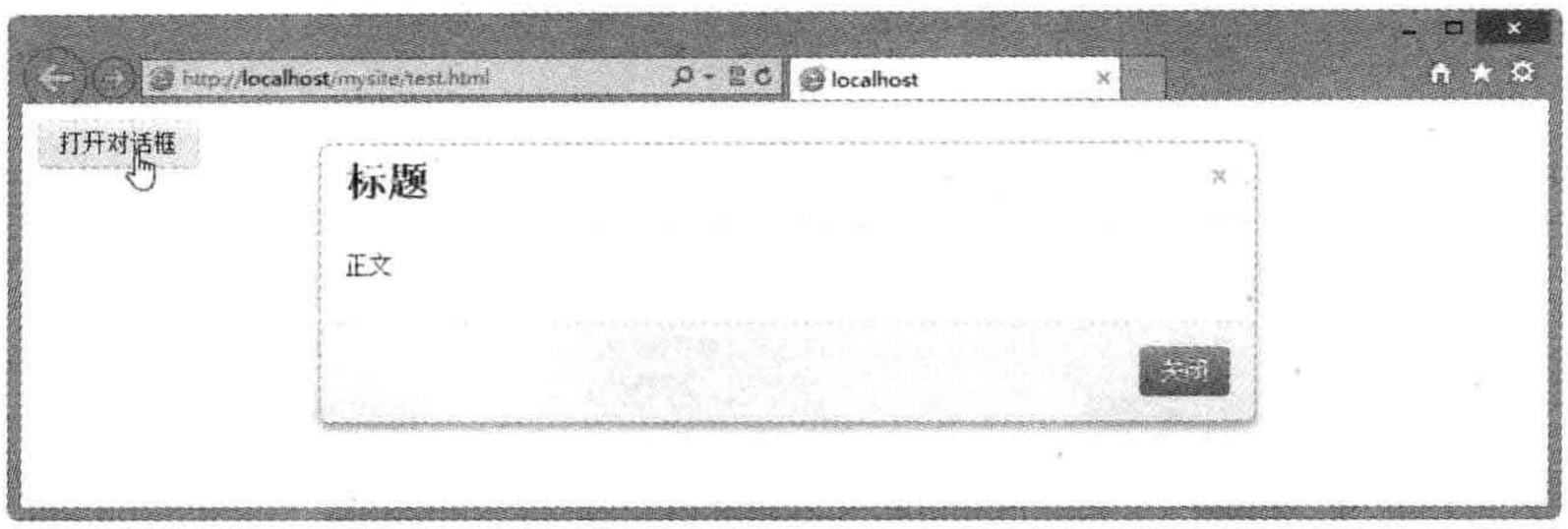


图 7-4　手动控制对话框显示或隐藏

Bootstrap 为对话框插件定义了 4 个事件，用来响应特定操作阶段的用户行为，说明如表 7-2 所示。

表 7-2　Modal 事件

事　件	描　述
show	当调用显示对话框的方法时会触发该事件
shown	当对话框显示完毕后触发该事件
hide	当调用隐藏对话框的方法时会触发该事件
hidden	当对话框隐藏完毕后触发该事件

例如，在下面的示例中，为当前对话框绑定 4 个监听事件，分别是 show、shown、hide 和 hidden，然后初始化对话框为显示状态，并隐藏遮罩层，为按钮调用 modal（"toggle"）方法。当对话框初始化显示以及单击按钮隐藏对话框的过程中，我们可以看到 4 个事件的执行顺序和发生节点，如图 7-5 所示。

```
<script type="text/JavaScript">
$(function(){
    $("#myModal").on("show",function(){
        alert(" 对话框开始打开 ");
    })
    $("#myModal").on("shown",function(){
        alert(" 对话框已经打开 ");
    })
    $("#myModal").on("hide",function(){
        alert(" 对话框开始关闭 ");
})
$("#myModal").on("hidden",function(){
    alert(" 对话框已经关闭 ");
})
$("#myModal").modal({
    backdrop:false,
    show:true
});
$(".btn").click(function(){
     $("#myModal").modal("toggle");
```

```
        })
    })
    </script>
```

图 7-5　通过事件监听对话框的显示和隐藏过程

7.3　下拉项

在第6章中我们详细讲解了下拉项组件的构成和应用。Bootstrap 通过 bootstrap-dropdown.js 脚本文件支持下拉项插件。因此，在使用之前应该导入 jquery.js 和 bootstrap-dropdown.js 插件支持文件：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-dropdown.js"></script>
```

或者直接导入 jquery.js 和 bootstrap.js 文件：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
```

调用下拉项的方法有两种，简单介绍如下。

1. HTML 属性调用

在超链接或按钮上添加 data-toggle="dropdown" 属性，即可激活下拉项交互行为。例如，在下面的示例中为 dropdown 中的按钮 <a> 标签添加 data-toggle="dropdown" 属性，即可激活下拉项，如图 7-6 所示。

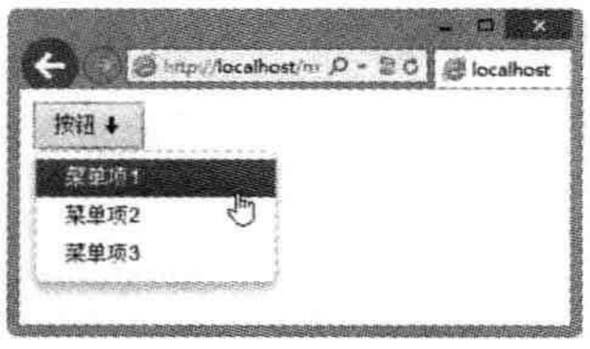


图 7-6　通过 data 属性激活下拉项

```
<div class="dropdown">
    <a href="#" class="btn" data-toggle="dropdown">按钮 <i class="icon-arrow-down">
      </i></a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
```

```
        </ul>
    </div>
```

注意　为了保证超链接 <a> 标签的 URL 符合规范，建议使用 data-target 属性代替 href="#"，而 href 属性用来执行链接操作。

```
    <div class="dropdown">
        <a href="/" class="btn" data-toggle="dropdown" data-target="#" >按钮 <iclass=
         "icon-arrow-down"></i></a>
         <ul class="dropdown-menu">
             <li><a href="#">…</a></li>
         </ul>
    </div>
```

2. JavaScript 调用

通过 dropdown() 构造函数可以直接调用下拉项。针对上面的示例，为激活按钮绑定 dropdown() 方法，具体用法如下：

```
<script type="text/JavaScript">
$(function(){
    $(".btn").dropdown();
})
</script>
<div class="dropdown">
    <a class="btn">按钮 <i class="icon-arrow-down"></i></a>
    <ul class="dropdown-menu">
        <li><a href="#">菜单项 1</a></li>
        <li><a href="#">菜单项 2</a></li>
        <li><a href="#">菜单项 3</a></li>
    </ul>
</div>
```

使用 JavaScript 调用 dropdown() 方法后，单击激活按钮，会弹出下拉项，但是再次单击激活按钮，就不再收起下拉项。

当下拉项隐藏时，调用 dropdown('toggle') 方法可以显示下拉项，反之，如果下拉项显示时，调用 dropdown('toggle') 方法可以隐藏下拉项。

```
$(function(){
    $(".btn").dropdown('toggle')
})
```

7.4　滚动监听

滚动监听是 Bootstrap 提供的非常实用的 JavaScript 插件，被广泛应用到 Web 开发中。例如，在天猫商城主页（http://www.tmall.com/）中，当用户滚动滚动条时，页面左侧的导航

条能够自动根据滚动的位置展开对应的导航菜单项明细，如图 7-7 所示。

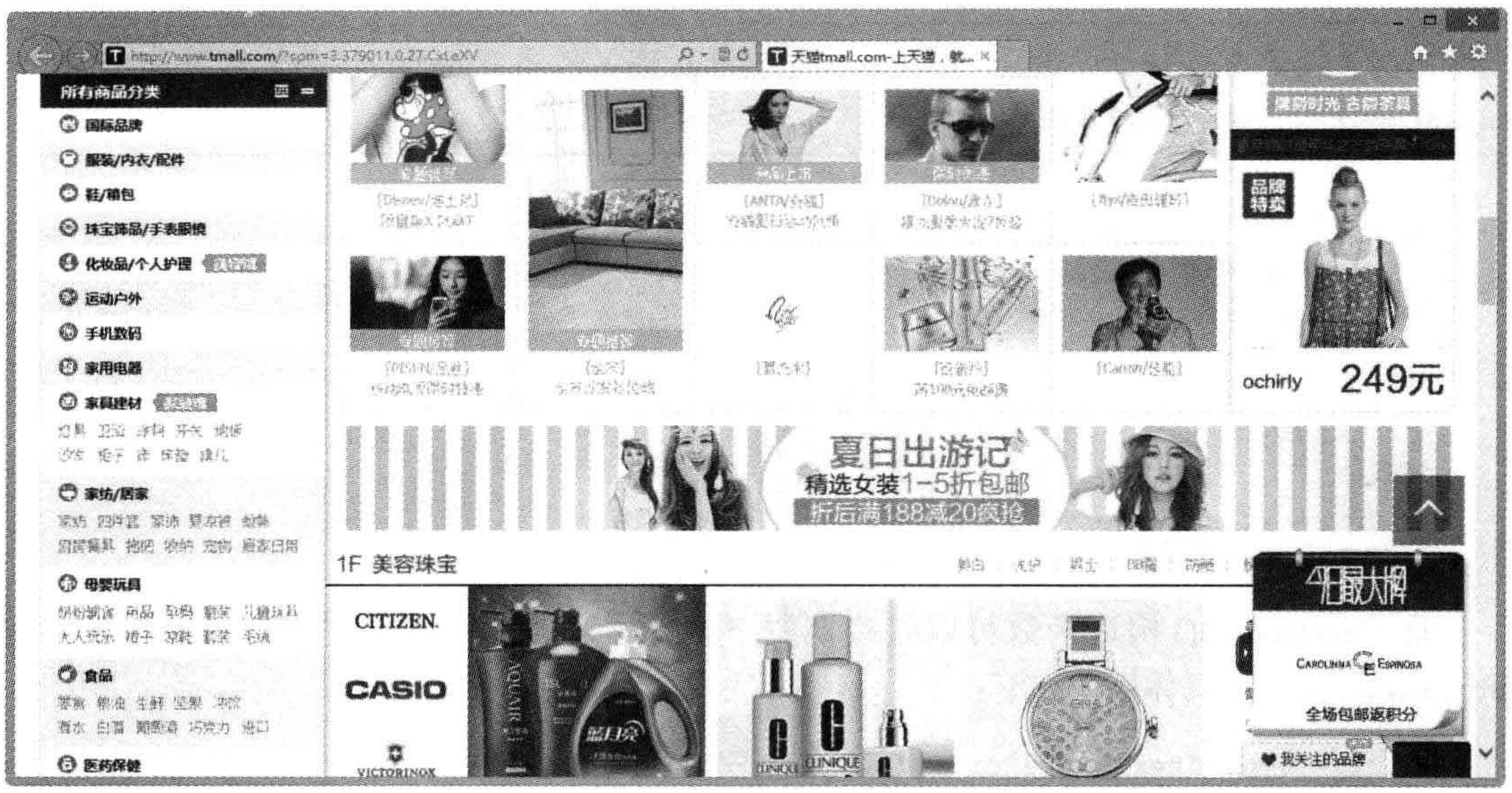


图 7-7　天猫主页滚动监听效果

7.4.1　使用滚动监听插件

Bootstrap 的 ScrollSpy（滚动监听）插件能够根据滚动的位置，自动更新导航条中相应的导航项。实现滚动监听的操作如下。

第 1 步：使用滚动监听插件之前，应在页面中导入 bootstrap-scrollspy.js 脚本文件或 bootstrap.js 脚本文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-scrollspy.js"></script>
```

第 2 步：设计导航条，在导航条中包含一个下拉项。分别为导航条列表项和下拉项设计锚点链接，锚记分别为 "#1"、"#2"、"#3"、"#4"、"#5"。同时为导航条外框定义一个 ID 值（id="menu"），以方便滚动监听控制。

```
<div id="menu" class="navbar navbar-fixed-top">
    <div class="navbar-inner">
        <ul class="nav">
            <li><a href="#1">列表 1</a></li>
            <li><a href="#2">列表 2</a></li>
            <li class="dropdown"> <a href="#" data-toggle="dropdown">下拉列表 <b
              class="caret"></b></a>
                <ul class="dropdown-menu">
                    <li><a href="#3">列表 3</a></li>
```

```
                    <li><a href="#4">列表 4</a></li>
                    <li class="divider"></li>
                    <li><a href="#5">列表 5</a></li>
                </ul>
            </li>
        </ul>
    </div>
</div>
```

第 3 步：设计监听对象。这里设计一个包含框，其中存放多个子内容框，代码如下。在内容框中，为每个标题设置锚点位置，即为每个 <h3> 标签定义 ID 值，对应值分别为 1、2、3、4、5。

```
<div class=" scrollspy">
    <h3 id="1">列表 1</h3>
    <p><img src="images/1.jpg"></p>
    <h3 id="2">列表 2</h3>
    <p><img src="images/2.jpg"></p>
    <h3 id="3">列表 3</h3>
    <p><img src="images/3.jpg"></p>
    <h3 id="4">列表 4</h3>
    <p><img src="images/4.jpg"></p>
    <h3 id="5">列表 5</h3>
    <p><img src="images/5.jpg"></p>
</div>
```

第 4 步：为监听对象（ ）定义样式类，设计该包含框为固定大小，并显示滚动条。

```
.scrollspy {
    width: 520px;
    height: 300px;
    overflow: scroll;
}
.scrollspy-example img { width: 500px; }
```

第 5 步：为监听对象设置被监听的 Data 属性：data-spy="scroll"，指定监听的导航条：data-target="#menu"，定义监听过程中滚动条偏移位置：data-spy="scroll" data-offset="30"。完成代码如下：

```
<div data-spy="scroll" data-target="#menu" data-offset="30" class="scrollspy">
</div>
```

第 6 步：在浏览器中预览，则可以看到当滚动 的滚动条时，导航条会实时监听并更新当前被激活的菜单项，效果如图 7-8 所示。

通过滚动监听插件，也可以为页面绑定监听行为，实现对页面滚动的监听响应。例如，针对上面的示例，为 <body> 标签建立监听行为：

```
<body data-spy="scroll" data-target="#navbar" data-offset="0">
```

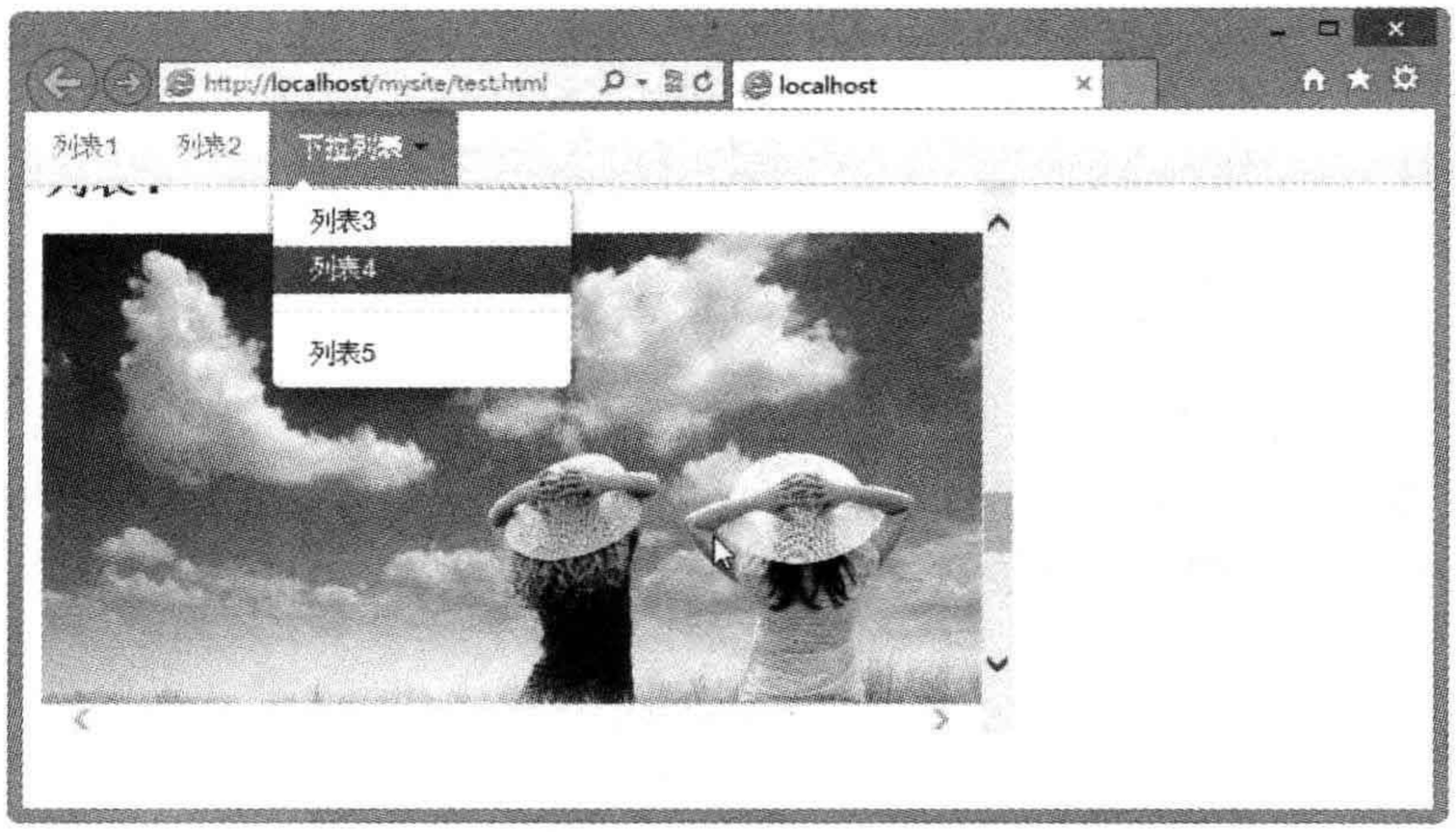


图 7-8　导航条自动监听滚动条的变化

然后，清理掉原来页面中的 包含框及其样式。清理掉导航条结构，重新设计导航结构，定义导航外包含框的 ID 值为 navbar。

```
<div id="navbar">
    <ul class="nav nav-pills nav-stacked">
        <li><a href="#1">列表 1</a></li>
        <li><a href="#2">列表 2</a></li>
        <li class="dropdown"> <a href="#" data-toggle="dropdown">下拉列表 <b class = "caret">
            </b></a><ul class="dropdown-menu">
                <li><a href="#3">列表 3</a></li>
                <li><a href="#4">列表 4</a></li>
                <li><a href="#5">列表 5</a></li>
            </ul>
        </li>
    </ul>
</div>
```

同时，在样式表中定义导航包含框，让其固定在浏览器窗口右上角位置，并定义宽度和背景色，样式代码如下：

```
#navbar {
    top: 50px;
    right: 10px;
    position: fixed;
    width: 200px;
    background-color: #FFF;
}
```

最后在浏览器中预览，滚动页面会发现导航列表会自动进行监听，并显示活动的菜单项，效果如图 7-9 所示。

图 7-9　导航列表自动监听页面滚动

7.4.2　控制滚动监听

Bootstrap 支持 HTML 和 JavaScript 两种方法调用滚动监听插件，简单说明如下。

（1）通过 data 属性调用滚动监听

在页面中为被监听的元素定义 data-spy="scroll" 属性，即可激活 Bootstrap 滚动监听插件。如果要监听浏览器窗口的内容滚动，则可以为 <body> 标签添加 data-spy="scroll" 属性。

```
<body data-spy="scroll">
```

然后，使用 data-target=" 目标对象 " 定义监听的导航结构，例如，为 body 元素定义 data-target="#navbar"，则 ID 值为 navbar 的导航框就拥有了监听页面滚动的行为。

```
<body data-spy="scroll" data-target="#navbar" >
```

（2）通过 JavaScript 调用滚动监听

直接为被监听的对象绑定 scrollspy() 方法即可。例如，针对 7.4.1 节介绍的第二个示例，我们可以使用 JavaScript 来快速为 <body> 标签绑定滚动监听行为。

```
<script type="text/JavaScript">
$(function(){
     $("body").scrollspy();
})
</script>
```

注意　在设计滚动监听时，必须为导航条中的链接指定相应的目标 ID。例如， 列表 1 必须与页面中类似 <h3　id="1"> 列表 1</h3> 的标签相呼应，即要为导航条设计好页内锚点。

---

scrollspy() 构造函数能够接收一个参数对象，在其中可以设置滚动偏移的值，当该属性为正值时，则滚动条向上偏移，为负值时将向下偏移。例如，如果使用下面的代码调用页面的滚动监听行为，则在浏览器中预览时，会发现在滚动条还没有滚出第一个标题内容区时，导航焦点已经切换到第二个列表项了，如图 7-10 所示，这是因为上面的代码调整了滚动监听的偏移位置，出现了错位现象。

```
<script type="text/JavaScript">
$(function(){
     $("body").scrollspy({
          offset: 200
     });
})
</script>
```

图 7-10　导航列表自动监听页面滚动

对于 Bootstrap 的插件来说，所有参数都可以通过 data 属性或 JavaScript 传递。对于 data 属性，将参数名附着到 data- 后面。例如，针对上面的 offset 配置参数，可以在 HTML 中通过 data-offset="" 进行相同的配置。offset 能够调整滚动定位的偏移位置，取值为数字，单位为像素，默认值为 10 像素。

滚动监听插件定义了一个事件：activate。该事件在当一个新的导航项目被激活时触发。例如，下面的示例建立在上面示例基础上，利用 activate 事件跟踪当前菜单项，判断如果当前项目为下拉菜单的包含框，即下拉菜单的父元素（<li class="dropdown">），则展开下拉菜单，否则收起下拉菜单。主要控制脚本如下，演示效果如图 7-11 所示。

```
<script type="text/JavaScript">
$(function(){
     $("body").scrollspy({
          offset: 0
     });
```

```
        $("body").on("activate",function(e){
                if(e.target && $(e.target).hasClass("dropdown")){
                        $(e.target).children("ul.dropdown-menu").css("display","block");
                }
                else{
                        $(e.target).parent().find("ul.dropdown-menu").css("display","none");
                }
        });
})
</script>
```

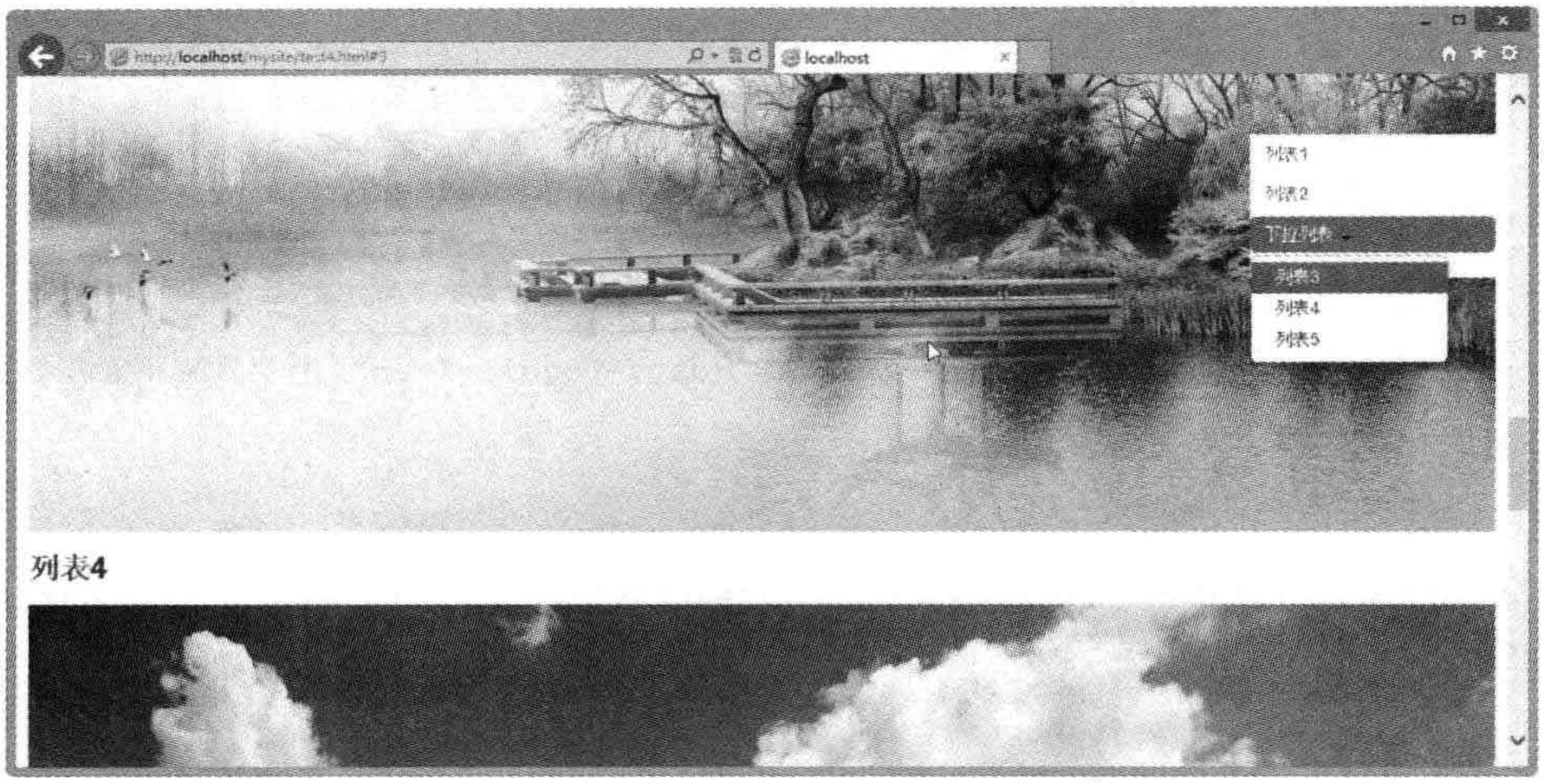


图 7-11　自动展开下拉菜单效果

滚动监听插件还定义了一个方法：scrollspy('refresh')。当滚动监听所作用的 DOM 有增删页面元素的操作时，需要调用下面的 refresh 方法：

```
$('[data-spy="scroll"]').each(function () {
        var $spy = $(this).scrollspy('refresh')
});
```

7.5　标签页

标签页插件需要 bootstrap-tab.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-tab.js 文件，或者 bootstrap.js 脚本文件。在使用标签页插件之前，建议读者先复习一下第 6 章介绍的标签页组件的基本 HTML 结构和使用。

7.5.1　使用标签页插件

使用标签页插件比较简单，首先在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
```

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

然后，设计标签页组件结构，在设计 HTML 结构时，应该注意两个问题：第一，导航区内每个超链接的链接定义为锚点链接，锚点值指向对应的标签内容框的 ID 值；第二，导航内容区域，需要使用 tab-content 类定义外包含框，使用 tab-pane 类定义每个 Tab 内容框。

最后，在导航区域内为每个超链接定义 data-toggle="tab"，激活标签页插件。对于下拉菜单选项，也可以通过该属性激活它们对应的行为。完整结构代码如下：

```
<div class="tabbable">
    <ul class="nav nav-tabs">
        <li class="active"><a href="#tab1" data-toggle="tab">超值特惠</a></li>
        <li><a href="#tab2" data-toggle="tab">当季推荐</a></li>
        <li><a href="#tab3" data-toggle="tab">潮流搭配</a></li>
        <li class="dropdown"><a href="#" class="dropdown-toggle" data-toggle=
         "dropdown">更多选择 <b class="caret"></b></a>
            <ul class="dropdown-menu">
                <li><a href="#tab4" data-toggle="tab">新品速递</a></li>
                <li><a href="#tab5" data-toggle="tab">特卖商品</a></li>
            </ul>
        </li>
    </ul>
    <div class="tab-content">
        <div class="tab-pane active" id="tab1"><img src="images/1.png"></div>
        <div class="tab-pane fade" id="tab2"><img src="images/2.png"></div>
        <div class="tab-pane fade" id="tab3"><img src="images/3.png"></div>
        <div class="tab-pane fade" id="tab4"><img src="images/4.png"></div>
        <div class="tab-pane fade" id="tab5"><img src="images/5.png"></div>
    </div>
</div>
```

保存页面，在浏览器中预览，效果如图 7-12 所示。

图 7-12　标签页演示效果

7.5.2　控制标签页插件

调用标签页插件的方法也有两种。一种方法是通过 HTML 的 data 属性来激活，此时不需要编写任何 JavaScript 脚本，只需在导航标签或者导航 pill 的超链接中添加 data-toggle="tab" 或者 data-toggle="pill" 属性即可。同时，确保为导航包含框添加 nav 和 nav-tabs 样式类。具体示例可以参阅 7.5.1 节的示例代码。

另一种方法是通过 JavaScript 脚本直接调用，调用方法是在每个超链接的单击事件中调用 tab('show') 方法，显示对应的标签内容框。例如，针对上面的示例代码，清理掉每个超链接的 data-toggle="tab"，然后编写如下脚本：

```
<script type="text/JavaScript">
$(function(){
     $(“.nav-tabs a").click(function(e){
          e.preventDefault();
          $(this).tab('show');
     });
})
</script>
```

其中 e.preventDefault(); 阻止超链接的默认行为，$(this).tab('show'); 显示当前标签页对应的内容框内容。

因此，根据上面的实现方法，用户还可以设计单独控制按钮，专门显示特定 Tab 项的内容框。

```
$('.nav-tabs a[href="#profile"]').tab('show');          // 显示 ID 名为 profile 的项目
$('.nav-tabs a:first').tab('show');                     // 显示第一个 Tab 选项

$('.nav-tabs a:last').tab('show');                      // 显示最后一个 Tab 选项
$('.nav-tabs li:eq(2) a').tab('show');                  // 显示第 3 个 Tab 选项
```

标签页插件包含两个事件，简单说明如下。

- show：在一个标签选项被显示前将触发。通过 event.target 和 event.relatedTarget 可以获取当前触发的 Tab 标签和前面一个被激活的 Tab 标签。
- shown：在一个标签选项被显示之后触发。通过 event.target 和 event.relatedTarget 可以获取当前触发的 Tab 标签和前一个被激活的 Tab 标签。

例如，针对 7.5.1 节的示例，分别为当前标签页绑定 show 和 shown 事件，并实时跟踪当前 Tab 选项的地址信息和前一个 Tab 选项的地址信息，然后把这些信息显示在页面中，效果如图 7-13 所示。

```
<script type="text/JavaScript">
$(function(){
     $(".nav-tabs a").click(function(e){
          e.preventDefault();
          $(this).tab('show');
```

```
        });
        $(".nav-tabs a").on("show",function(e){
              $("div#info").html(" 前一个 Tab 选项目标: " + e.relatedTarget);
        });
        $(".nav-tabs a").on("show",function(e){
              $("div#info1").html(" 当前 Tab 选项目标: " + e.target);
        });
    })
</script>

<div class="tabbable">
      <ul class="nav nav-tabs">
      </ul>
      <div class="tab-content">
      </div>
</div>
<div id="info"></div>
<div id="info1"></div>
```

图 7-13　动态跟踪标签页选项的切换过程

7.6　工具提示

在网页设计中，经常需要提示用户某些功能或相关消息，而 HTML 自带的 title 属性比较简陋：样式单调、功能单一、用法呆板，经常需要自己去开发相关的工具提示条功能。Bootstrap 定义一个工具提示插件，功能比较完善，使用简单、灵活。

工具提示插件需要 bootstrap-tooltip.js 文件支持，因此在使用该插件之前，应该导入

jquery.js 和 bootstrap-tooltip.js 文件，或者 bootstrap.js 脚本文件。

7.6.1　使用工具提示插件

使用工具提示插件比较简单，首先在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-tooltip.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

然后，在页面中设计一个超链接，定义 title 属性，设置工具提示文本信息，代码如下：

```
<a href="http://www.baidu.com/" title="百度一下，你就知道 "><img src="images/logo.
 gif"></a>
```

出于性能方面的考虑，Bootstrap 没有支持工具提示插件通过 data 属性激活，因此用户必须手动调用。调用通过 tooltip() 方法来实现，代码如下：

```
<script type="text/JavaScript">
$(function(){
     $('a').mouseover(function() {
            $(this).tooltip('show');
     })
});
</script>
```

在浏览器中预览，显示效果如图 7-14 所示。

通过 data-placement="" 属性可以设置提示信息的显示位置，取值包括 top、right、bottom、left。例如，在下面的代码中分别使用 data-placement 属性定义工具提示信息显示在顶部、右侧、底部和左侧，显示效果如图 7-15 所示。

```
<script type="text/JavaScript">
$(function(){
     $('a').mouseover(function() {
            $(this).tooltip('show');
     })
});
</script>

<a href="http://www.baidu.com/" data-placement="top" title="百度一下，你就知道 "><img src=
 "images/logo.gif"></a>
<a href="http://www.baidu.com/" data-placement="right" title="百度一下，你就知道 "><img src=
 "images/logo.gif"></a>
<a href="http://www.baidu.com/" data-placement="bottom" title="百度一下，你就知道 "><img src=
 "images/logo.gif"></a>
<a href="http://www.baidu.com/" data-placement="left" title="百度一下，你就知道 "><img src=
 "images/logo.gif"></a>
```

图 7-14　工具提示演示效果

图 7-15　设置工具提示显示位置

注意　当为文本框输入组添加工具提示功能时，建议设置 container 包含框，以避免不必要的副作用。

7.6.2　控制工具提示插件

在使用工具提示插件时，可以通过 JavaScript 触发，核心代码如下：

```
$('#example').tooltip(options)
```

其中 $（'#example'）表示匹配的页面元素，options 是一个参数对象，可以配置工具提示的相关设置属性，说明如表 7-3 所示。

表 7-3　Tooltip 配置参数 options 属性

名　称	类　型	默认值	描　述
animation	boolean	true	是否应用 CSS 淡入淡出过渡特效显示工具提示
html	boolean	false	是否插入 HTML 字符串，如果设置 false，则使用 jQuery 的 text() 方法插入文本，就不用担心 XSS 攻击
placement	string | function	'top'	设置提示的位置，取值有 top、bottom 、left、right
selector	string	false	设置一个选择器字符串，则具体提示针对选择器匹配的目标进行显示
title	string | function	''	如果 title 属性不存在，则需要显示的提示文本
trigger	string	'hover focus'	设置工具提示的触发方式，包括单击（click）、鼠标经过（hover）、获取焦点（focus）或者手动（manual），可以指定多种方式，多种方式之间通过空格进行分隔
delay	number | object	0	延迟显示和隐藏工具提示，不适用于手动触发类型。如果提供一个数值，则表示隐藏或者显示的延迟时间；如果是一个对象结构，可以这样进行设置：{ show: 500, hide: 100 }，它分别表示显示和隐藏的延迟时间
container	string | false	false	是否追加一个特定的元素容器提示：body

可以通过 data 属性或 JavaScript 传递参数。对于 data 属性，将参数名附着到 data- 后面

即可，如 data-animation=""。也可以针对单个工具提示指定单独的 data 属性。

```
<a href="#" data-toggle="tooltip" title="first tooltip">hover over me</a>
```

工具提示插件拥有多个实用方法，说明如下。

- .tooltip('show')：弹出某个页面元素的工具提示。
- .tooltip('hide')：隐藏某个页面元素的工具提示。
- .tooltip('toggle')：打开或隐藏某个页面元素的工具提示。
- .tooltip('destroy')：隐藏并销毁某个页面元素的工具提示。

例如，在下面的示例中通过设置工具提示的参数，让提示信息以 HTML 文本格式显示一幅图片，同时延迟一秒钟显示，并推迟半秒钟隐藏，效果如图 7-16 所示。

```
<script type="text/JavaScript">
$(function(){
    $('a').tooltip({
            html:true,
            title:"<img src='images/logo.gif' />",
            placement:"right",
            delay: { show: 1000, hide: 500 }
    })
});
</script>

<a href="http://www.baidu.com/">百度</a>
```

图 7-16　自定义工具提示显示信息

7.7　弹出提示

弹出提示是工具提示的子类，建立在工具提示插件基础上进行拓展。它比工具提示插件多了一个 content 参数，除了一些默认值不同外，用法基本相同。与工具提示插件一样，弹出提示也没有自定义事件。

弹出提示插件需要 bootstrap-popover.js 文件支持，因此在使用该插件之前，应该导入

jquery.js 和 bootstrap-popover.js 文件，或者 bootstrap.js 脚本文件。

7.7.1 使用弹出提示插件

使用弹出提示插件比较简单，首先在页面头部位置导入插件所需要的脚本文件和样式表文件：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-popover.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

然后，在页面中设计一个超链接，定义 title 属性，设置弹出提示标题信息，定义 data-content 属性，设置弹出提示的正文内容，代码如下：

```
<a href="#" class="btn btn-large btn-success"  title=" 弹出提示标题 " data-content=
 " 这里将显示弹出提示的正文内容 "> 单击查看效果 </a>
```

出于性能方面的考虑，Bootstrap 没有支持弹出提示插件通过 data 属性激活，因此用户必须手动调用。调用的方法是通过 popover() 方法实现，代码如下：

```
$(function(){
    $('a').popover();
});
```

在浏览器中预览，显示效果如图 7-17 所示。

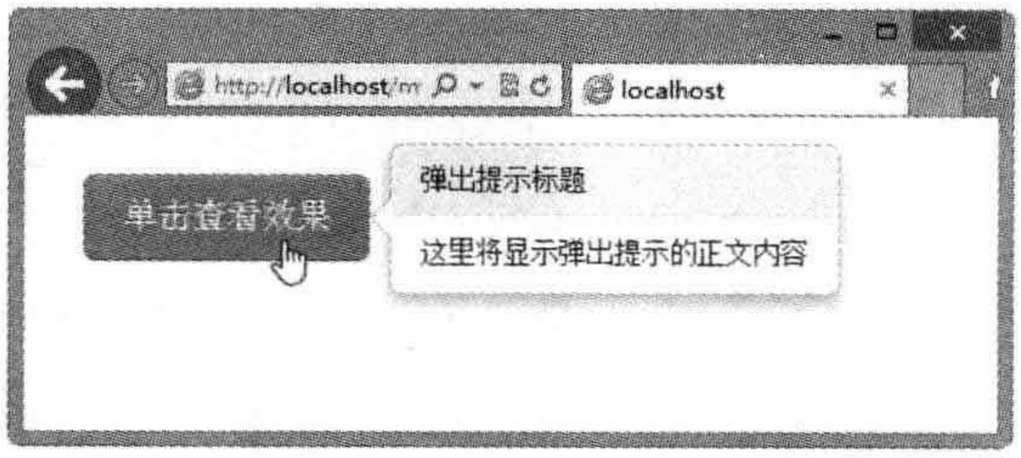


图 7-17　弹出提示演示效果

与工具提示默认显示位置不同，弹出提示默认显示位置在目标对象的右侧。通过 data-placement="" 属性可以设置提示信息的显示位置，取值包括 top、right、bottom、left。例如，在下面代码中分别使用 data-placement 属性定义工具提示信息显示在顶部、右侧、底部和左侧，显示效果如图 7-18 所示。

```
$(function(){
    $('a').popover();
});
</script>

<a href="#" class="btn btn-large btn-success"  title=" 弹出提示标题 " data-content=
 " 这里将显示弹出提示的正文内容 "> 单击查看效果 </a>
<a href="#" class="btn btn-large btn-success" data-placement="top"  title=" 弹出
 提示标题 " data-content=" 这里将显示弹出提示的正文内容 "> 单击查看效果 </a>
<a href="#" class="btn btn-large btn-success" data-placement="bottom" title=" 弹
 出提示标题 " data-content=" 这里将显示弹出提示的正文内容 "> 单击查看效果 </a>
<a href="#" class="btn btn-large btn-success" data-placement="left" title=" 弹出
 提示标题 " data-content=" 这里将显示弹出提示的正文内容 "> 单击查看效果 </a>
```

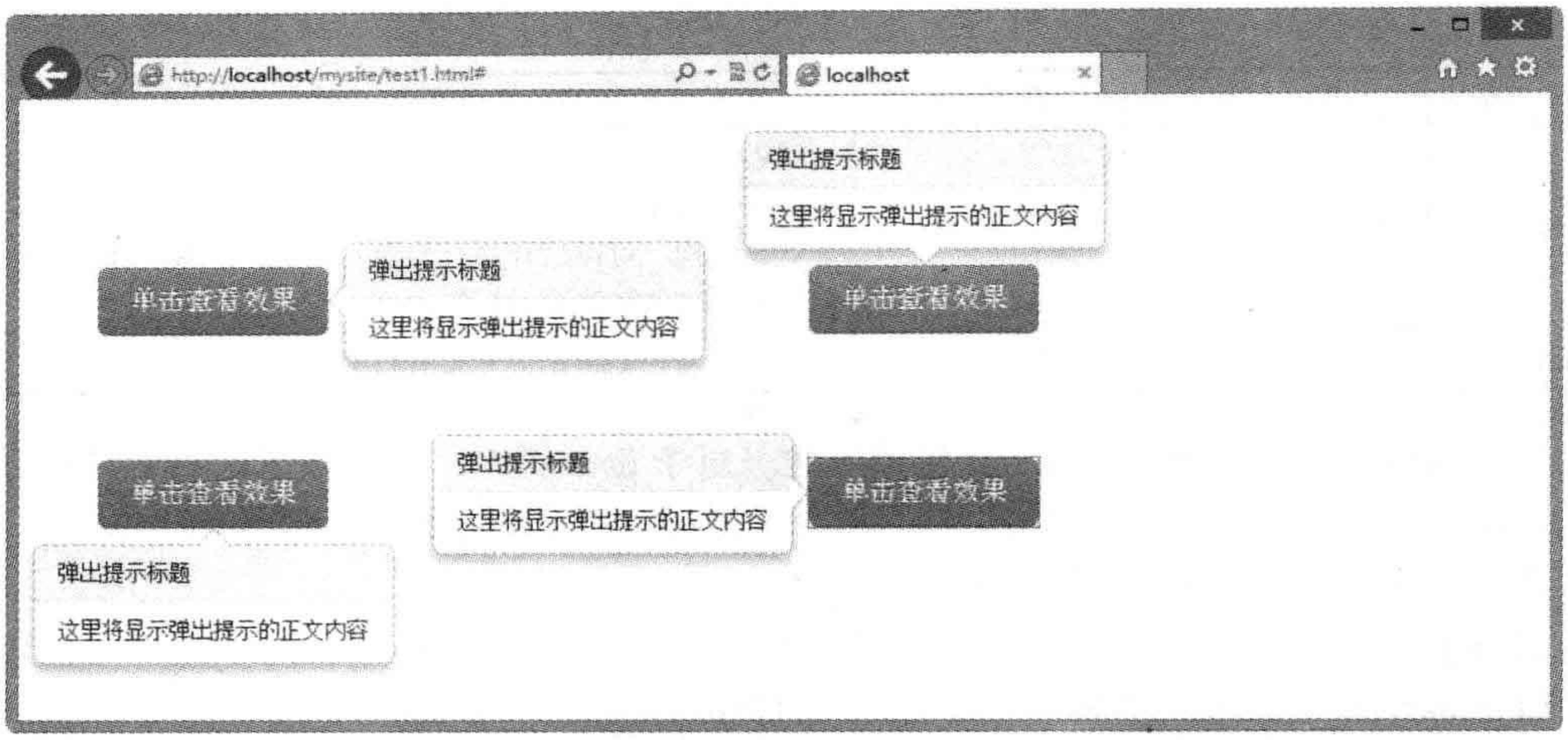


图 7-18　设置弹出提示显示位置

注意　当为文本框输入组添加工具提示功能时，建议设置 container 包含框，以避免不必要的副作用。

7.7.2　控制弹出提示插件

在使用弹出提示插件时，可以通过 JavaScript 触发，核心代码如下：

```
$('#example').popover(options)
```

其中 $（'#example'）表示匹配的页面元素，options 是参数对象，可以配置弹出提示的相关设置属性，说明如表 7-4 所示。

表 7-4　Popover 配置参数 options 属性

名　称	类　型	默认值	描　述
animation	boolean	true	是否应用 CSS 淡入淡出过渡特效显示工具提示
html	boolean	false	是否插入 HTML 字符串，如果设置为 false，则使用 jQuery 的 text() 方法插入文本，就不用担心 XSS 攻击
placement	string | function	'top'	设置提示的位置，取值有 top、bottom、left、right
selector	string	false	设置一个选择器字符串，则具体提示针对选择器匹配的目标进行显示
title	string | function	''	如果 title 属性不存在，则需要显示的提示标题文本
content	string | function	''	如果 data-content 属性不存在，则需要显示的提示正文文本
trigger	string	'hover focus'	设置工具提示的触发方式，包括单击（click）、鼠标经过（hover）、获取焦点（focus）或者手动（manual），可以指定多种方式，多种方式之间通过空格进行分隔

（续）

名　称	类　型	默认值	描　述
delay	number | object	0	延迟显示和隐藏工具提示，不适用于手动触发类型。如果提供一个数值，则表示隐藏或者显示的延迟时间；如果是一个对象结构，可以这样进行设置：{show: 500, hide: 100}，它分别表示显示和隐藏的延迟时间
container	string | false	false	是否追加一个特定的元素容器提示：body

可以通过 data 属性或 JavaScript 传递参数。对于 data 属性，将参数名附着到 data- 后面即可，如 data-animation=""。也可以针对单个弹出提示指定单独的 data 属性。

弹出提示插件拥有多个实用方法，说明如下。

- .popover('show')：显示某个页面元素的弹出提示。
- .popover('hide')：隐藏某个页面元素的弹出提示。
- .popover('toggle')：打开或隐藏某个页面元素的弹出提示。
- .popover('destroy')：隐藏并销毁某个页面元素的弹出提示。

例如，在下面的示例中通过设置弹出提示的参数，模拟工具提示效果，让弹出信息以 HTML 文本格式显示一幅图片，同时延迟一秒钟显示，并推迟半秒钟隐藏，效果如图 7-19 所示。

```
<script type="text/JavaScript">
$(function(){
     $('a').popover({
              html:true,
              title:"模拟工具提示功能",
              content:"<img src='images/logo.gif' />",
              placement:"bottom",
              trigger:"hover",
              delay: { show: 1000, hide: 500 }
     })
});
</script>

<a class="btn btn-large btn-success" href="http://www.baidu.com/">百度</a>
```

图 7-19　自定义弹出提示显示信息

7.8　警告框

在第 6 章中我们曾经介绍过警告框组件的结构和样式，本节将对警告框插件的相关行为和脚本设计进行详细讲解。

警告框插件需要 bootstrap-alert.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-alert.js 文件，或者 bootstrap.js 脚本文件。在使用提示框插件之前，应在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

然后，设计一个警告框包含框，并添加一个关闭按钮，代码如下：

```
<div class="alert fade in">
     <button type="button" class="close" data-dismiss="alert">&times;</button>
     <strong>警告框标题 </strong> 说明文字。
</div>
```

预览效果如图 7-20 所示。

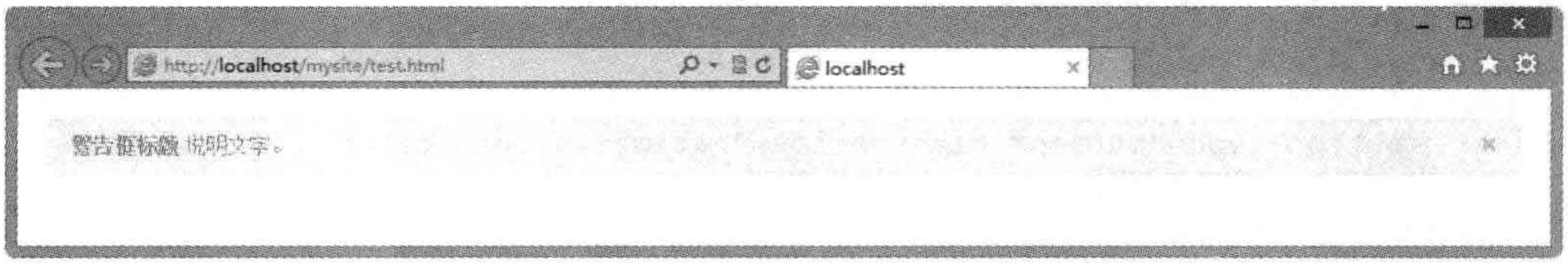


图 7-20　警告框插件效果

警告框插件基本功能在 CSS 上，JavaScript 部分就是一个关闭事件。这是框架提供的关闭事件，通过 JavaScript 为某个警告框添加关闭功能：

```
$(".alert").alert("close")
```

如果不使用 JavaScript，则在 HTML 标签中定义属性即可，仅需将 data-dismiss="alert" 添加到关闭按钮上，即可自动赋予某个警告框关闭的功能。

```
<a class="close" data-dismiss="alert" href="#">&times;</a>
```

alert() 方法赋予所有警告框以关闭功能。如果希望警告框在关闭时带有动画效果，则应该确保 .fade 和 .in 类已经应用到这些提示包含框上，可以参见上面的示例代码。

例如，针对上面的示例，可以使用 JavaScript 脚本来控制警告框关闭控制。

```
<script type="text/JavaScript">
$(function(){
     $(".close").click(function(){
          $(this).alert("close");
     })
});
```

```
</script>
<div class="alert fade in">
     <button type="button" class="close">&times;</button>
     <strong>警告框标题</strong> 说明文字。
</div>
```

警告框插件支持两个事件，说明如下。

- close：在关闭警告框之前触发。
- closed：在关闭警告框之后触发。

例如，在下面的示例中，我们将警告框插件与弹出对话框插件捆绑使用，当关闭警告框之前，将弹出一个对话框进行提示，演示效果如图 7-21 所示。

```
<script type="text/JavaScript">
$(function(){
     $(".close").click(function(){
          $(this).alert("close");
     })
     $(".alert").on("close",function(e){
          $("#myModal").modal();
     })
});
</script>

<div class="alert fade in">
     <button type="button" class="close">&times;</button>
     <strong>警告框标题</strong> 说明文字。
</div>
<div id="myModal" class="modal hide fade">
     <div class="modal-header">
          <h3>提示</h3>
     </div>
     <div class="modal-body">
          <p>确定要关闭警告框信息？</p>
     </div>
     <div class="modal-footer">
          <button class="btn btn-info" data-dismiss="modal">关闭</button>
     </div>
</div>
```

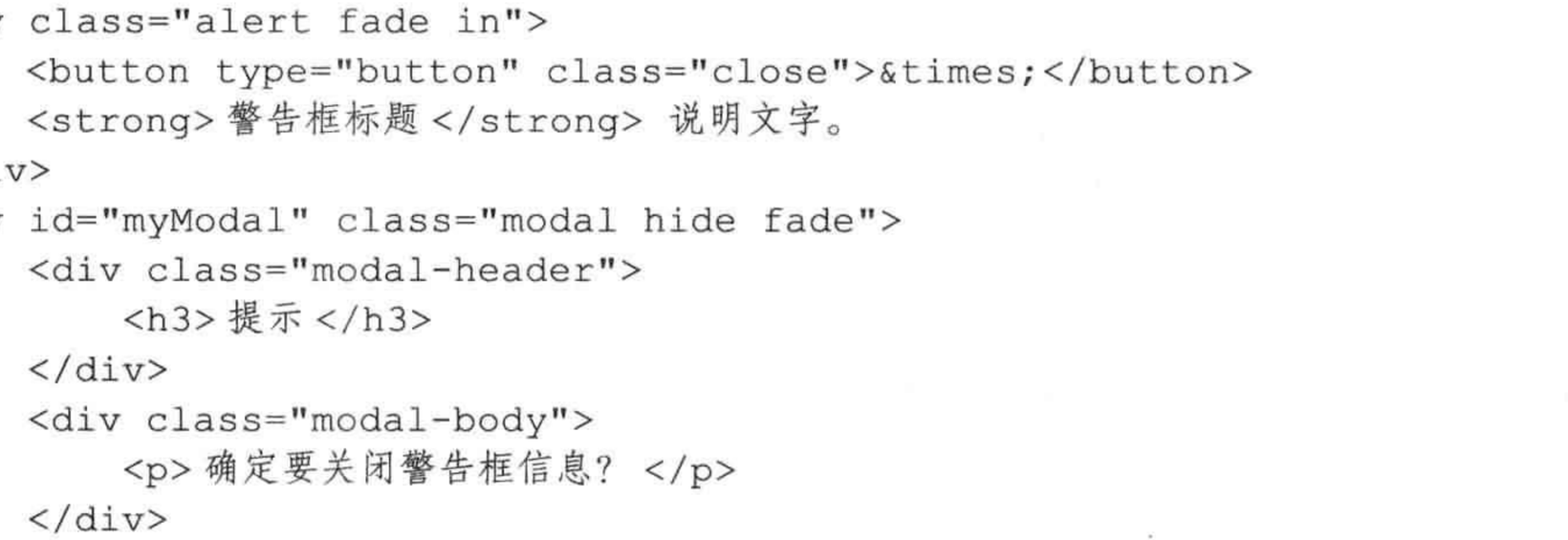


图 7-21　把警告框和弹出对话框插件捆绑应用

7.9　按钮

按钮是一类常用组件，Bootstrap 把按钮设计得非常智能，只有一个调用接口，使用很方便。按钮在 Bootstrap 中扮演很多角色，它可以实现很多功能，如为工具条（toolbar）之类的组件赋予控制按钮的状态，或者创建一组按钮的功能。

按钮插件需要 bootstrap-button.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-button.js 文件，或者 bootstrap.js 脚本文件。在使用按钮插件之前，应在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

一般情况下，要激活按钮的行为，可以使用 data-toggle 属性，用法如下：

```
<button type="button" class="btn btn-primary" data-toggle="button"> 激活按钮 </button>
```

1. 加载状态

通过按钮可以设计状态提示，当单击按钮时，会显示 loading 状态信息。例如，在下面的代码中设计一个按钮，当单击按钮时将调用 button（"loading"）方法，触发按钮的加载状态，效果如图 7-22 所示。

图 7-22　加载状态效果

```
<script type="text/JavaScript">
$(function(){
     $(".btn").click(function(){
                    $(this).button("loading");
     })
});
</script>

<button data-loading-text=" 正在加载 ..." class="btn btn-primary"> 加载 </button>
```

在上面的代码中，通过 data-loading-text 属性定义加载的信息文本，通过 button（"loading"）方法激活按钮的加载状态行为。

2. 模拟复选框组

复选框组是一组复选框，可以实现多选操作。使用按钮组模拟复选框组，能够设计更具个性的复选框样式。例如，在下面代码中设计 3 个按钮组成一组，然后通过 data-toggle="buttons-checkbox" 属性把它们定义为复选框组，单击的按

图 7-23　勾选的按钮将加深背景色显示

钮将显示深色的背景，再次单击将会显示浅色背景效果，效果如图 7-23 所示。

```
<div class="btn-group" data-toggle="buttons-checkbox">
    <button type="button" class="btn btn-primary">语文</button>
    <button type="button" class="btn btn-primary">数学</button>
    <button type="button" class="btn btn-primary">英语</button>
</div>
```

3. 模拟单选按钮组

单选按钮组是一组单选按钮，可以实现单选操作。使用按钮组模拟单选按钮组，能够设计更具个性的单选按钮样式。例如，在下面的代码中设计两个按钮组成一组，然后通过 data-toggle="buttons-radio" 属性把它们定义为单选按钮组，单击的按钮将显示深色的背景，这样可以在两个按钮之间进行切换，但不能够同时选中或者不选中，效果如图 7-24 所示。

图 7-24　设计的单选按钮组效果

```
<div class="btn-group" data-toggle="buttons-radio">
    <button type="button" class="btn btn-primary">男</button>
    <button type="button" class="btn btn-primary">女</button>
</div>
```

4. 重置和完成状态

在按钮处于加载状态或者其他行为状态中时，可以使用 data-reset-text 和 data-complete-text 属性设置重置和完成提示信息，然后通过 button（'reset'）和 button（'complete'）方法来激活它们。

例如，在下面的示例中为按钮设计 3 个状态：一个是加载状态（data-loading-text），一个是完成状态（data-complete-text），另一个是重置状态（data-reset-text），然后设计当单击按钮时触发加载状态，延迟 2 秒钟之后显示加载完成状态提示，再过 2 秒钟之后则显示重置按钮状态，效果如图 7-25 所示。

```
<script type="text/JavaScript">
$(function(){
    $(".btn").click(function(){
        $(this).button("loading");
        setTimeout((function(_this){
            return function(){
                $(_this).button("complete");
            }
        })(this), 2000);
        setTimeout((function(_this){
            return function(){
                $(_this).button("reset");
            }
```

```
            })(this), 4000);
    })
});
</script>

<button  class="btn btn-primary" data-loading-text=" 正在加载 ..." data-reset-text="
   重新加载 " data-complete-text=" 完成加载 "> 加载 </button>
```

图 7-25　设计按钮在 4 种状态中切换

需要注意的是，Bootstrap 提供了比较灵活的用法，对于按钮的状态设置，可以自定义 data 属性，并在调用传递自定义的 data 字符串进行激活。例如，针对上面的示例，我们也可以按如下代码进行设计，重写 loading、complete 和 reset 状态的字符串表示，则演示效果依然相同，唯一的不同是 loading 属性拥有内置样式。

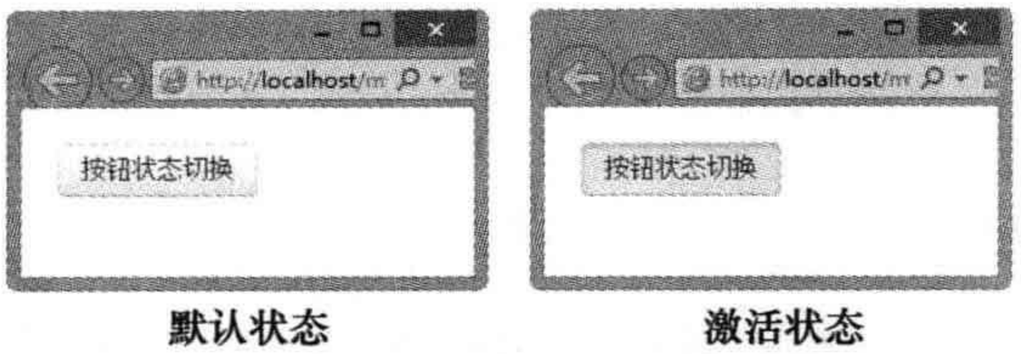
```
<script type="text/JavaScript">
$(function(){
    $(".btn").click(function(){
        $(this).button("a");
        setTimeout((function(_this){
            return function(){
                $(_this).button("b");
            }
        })(this), 2000);
        setTimeout((function(_this){
            return function(){
                $(_this).button("c");
            }
        })(this), 4000);
    })
});
</script>

<button  class="btn btn-primary" data-a-text=" 正在加载 ..." data-c-text=" 重新加载 "
 data-b-text=" 完成加载 "> 加载 </button>
```

5. 状态切换

使用 data-toggle 属性可以激活按钮的行为状态，实现在激活和未激活之间进行状态切换。例如，下面代码可以激活按钮行为特性，单击时将会激活按钮，再次单击将会让按钮恢复为默认状态。

```
<button type="button" class="btn" data-toggle="button" > 按钮状态切换 </button>
```

也可以使用 JavaScript 脚本来激活按钮的行为状态。例如，针对上面的这行代码，转换为 JavaScript 脚本形式激活，则代码如下，效果如图 7-26 所示。



默认状态　　激活状态

图 7-26　设计按钮激活行为

```
<script type="text/JavaScript">
$(function(){
    $(".btn").click(function(){
        $(this).button("toggle");
    })
});
</script>
<button type="button" class="btn" >按钮状态切换</button>
```

7.10　折叠

当列表菜单项目特别多的时候，使用类似于手风琴的折叠菜单是个不错的选择。折叠菜单利于组织菜单项、节约网页空间和方便用户单击浏览。纯 CSS 也可以实现悬浮效果的折叠菜单，但是结合 JavaScript 能得到更好的特效和控制能力。当然，切记不要单纯地为了花哨的特效而使用 JavaScript 折叠菜单。

折叠插件需要 bootstrap-collapse.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-collapse.js 文件，或者 bootstrap.js 脚本文件。折叠和导航插件具有基本的样式，并相互提供灵活的支持。

7.10.1　使用折叠插件

折叠插件具有复杂的结构，但是调用比较简单，可以通过 data 属性调用，也可以通过 JavaScript 脚本调用。下面通过示例介绍折叠插件的一般用法。

第 1 步：在使用折叠插件之前，应在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-collapse.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

第 2 步：设计折叠包含框，定义 accordion 类样式，设计 ID 值，该值将作为 data-parent 属性的引用，以确保当前折叠插件中只有一个选项能够打开。在折叠外包含框内设计 3 个子容器，引入 accordion-group 类名，此时的折叠外壳效果如图 7-27 所示。

```
<div class="accordion" id="box">
    <div class="accordion-group"></div>
    <div class="accordion-group"></div>
    <div class="accordion-group"></div>
</div>
```

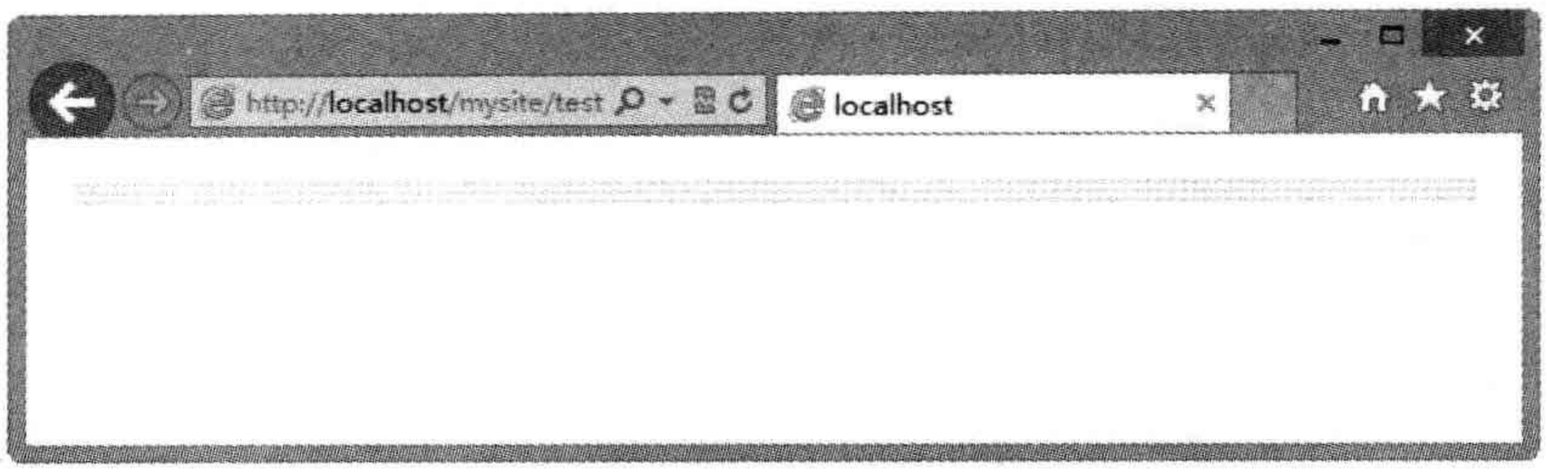


图 7-27　设计折叠外壳效果

第 3 步：设计折叠的选项面板。每个面板包含两部分：第一部分是标题部分（ ），在该子框中可以添加导航标题；第二部分是内容主体部分（ ）。一个完整的折叠单元结构如下：

```
<div class="accordion-group">
    <div class="accordion-heading"><a class="accordion-toggle" href="#">财经 / 生活
     </a> </div>
    <div class="accordion-body collapse in">
        <div class="accordion-inner"><img src="images/1.png"></div>
    </div>
</div>
```

第 4 步：为了把标题和内容框捆绑在一起，可以通过锚链接的方法，把 和 连在一起，代码如下：

```
<div class="accordion-group">
   <div class="accordion-heading"><a class="accordion-toggle"  href="#1">财经 / 生活
    </a> </div>
   <div id="1" class="accordion-body collapse in">
       <div class="accordion-inner"><img src="images/1.png"></div>
   </div>
</div>
```

第 5 步：激活折叠交互行为。为标题区块的超链接定义 data-toggle="collapse" 激活折叠交互行为。同时，通过定义 data-parent="#box" 属性，设置折叠的包含框，以便在该框内只能够显示一个单元项目。

```
<div class="accordion-heading"><a class="accordion-toggle" data-toggle="collapse"
 data-parent="#box" href="#1">财经 / 生活 </a> </div>
```

第 6 步：完成折叠整个结构的设计，完整的结构代码如下，预览效果如图 7-28 所示。

```
<div class="accordion" id="box">
    <div class="accordion-group">
        <div class="accordion-heading"><a class="accordion-toggle" data-toggle=
         "collapse" data-parent="#box" href="#1">财经 / 生活 </a> </div>
        <div id="1" class="accordion-body collapse in">
```

```
            <div class="accordion-inner"><img src="images/1.png"></div>
        </div>
    </div>
    <div class="accordion-group">
        <div class="accordion-heading"><a class="accordion-toggle" data-toggle="
         collapse" data-parent="#box" href="#2">搞笑 / 广告 </a> </div>
        <div id="2" class="accordion-body collapse">
            <div class="accordion-inner"><img src="images/2.png"></div>
        </div>
    </div>
    <div class="accordion-group">
        <div class="accordion-heading"><a class="accordion-toggle" data-toggle="
         collapse" data-parent="#box" href="#3">特色 / 旅游 </a> </div>
        <div id="3" class="accordion-body collapse">
            <div class="accordion-inner"><img src="images/3.png"></div>
        </div>
    </div>
</div>
```

图 7-28　设计折叠演示效果

需要说明的是，Bootstrap 的折叠只能以垂直形式出现，利用 data-toggle="collapse" 属性可以设计折叠面板。例如，在下面的代码中，为按钮定义了 data-toggle="collapse" 属性，同时使用 data-target="#box" 属性把当前按钮与一个面板捆绑在一起，当单击按钮时，能够自动隐藏或者显示面板，代码如下，演示效果如图 7-29 所示。

```
<a class="btn btn-primary" data-toggle="collapse" data-target="#box">折叠面板 </a>
<div id="box" class="in">
    <div ><img src="images/1.png"></div>
</div>
```

图 7-29　设计折叠面板效果

7.10.2　控制折叠插件

调用折叠插件的方法有两种。一种是通过 data 属性实现，为控制标签添加 data-toggle="collapse" 属性，同时设置 data-target 属性，绑定控制标签要控制的包含框即可。如果使用超链接，则不用 data-target 属性，直接在 href 属性中定义目标锚点即可。

对于折叠插件来说，由于折叠插件结构复杂，因而还应该使用 data-parent 属性设置折叠的外包含框，以方便 Bootstrap 监控整个折叠组件的内部交互行为，确保在某个时间内只能够显示一个子项目。

除了 data 属性调用外，还可以使用 JavaScript 脚本形式进行调用，调用方法如下：

```
$(".collapse").collapse()
```

collapse() 方法可以包含一个配置对象，该对象包含以下两个配置参数。

- parent：设置折叠包含框，类型为选择器，默认值为 false。如果指定的父元素下包含多个折叠项目，则在同一时刻只能够显示一个项目，效果类似于传统的折叠行为。
- toggle：是否切换可折叠元素调用，布尔值，默认值为 true。

例如，针对上面的示例重新优化了 HTML 结构设计，清理掉所有的 data 自定义属性以及锚点绑定，仅保留折叠组件的类样式，HTML 结构代码如下：

```
<div class="accordion" id="box">
     <div class="accordion-group">
          <div class="accordion-heading"><a class="accordion-toggle" href="#">
           财经 / 生活 </a> </div>
           <div class="accordion-body collapse in">
                <div class="accordion-inner"><img src="images/1.png"></div>
           </div>
     </div>
     <div class="accordion-group">
        <div class="accordion-heading"><a class="accordion-toggle" href="#">
           搞笑 / 广告 </a> </div>
           <div class="accordion-body collapse">
```

```
            <div class="accordion-inner"><img src="images/2.png"></div>
        </div>
</div>
<div class="accordion-group">
    <div class="accordion-heading"><a class="accordion-toggle" href="#">
     特色 / 旅游 </a> </div>
      <div class="accordion-body collapse">
            <div class="accordion-inner"><img src="images/3.png"></div>
      </div>
</div>
```

然后，自带 JavaScript 脚本中输入下面的代码，即可实现折叠的交互行为和动态效果，如图 7-30 所示。

```
<script type="text/JavaScript">
$(function(){
    $(".accordion-body").collapse({
          parent:"#box",
          toggle:true
    });
    $(".accordion-toggle").click(function(){
          $(this).parent().next().collapse("toggle");
    })
});
</script>
```

图 7-30　JavaScript 脚本控制折叠效果

在上面的脚本中，首先，为所有的 调用 .collapse()，并通过配置参数设置折叠外包含框的 ID 值，同时设置 toggle 为 true，打开折叠交互切换效果。最后，为每个超链接 <a> 标签定义一个 click 事件，调用 collapse（"toggle"）方法激活折叠插件行为。

Bootstrap 为折叠插件定义了 3 个特定方法，调用它们可以实现特定的行为效果。

- .collapse('toggle')：切换一个可折叠元素，显示或者隐藏该元素。
- .collapse('show')：显示一个可折叠元素。
- .collapse('hide')：隐藏一个可折叠元素。

Bootstrap 还为折叠组件提供了一组事件，通过这些事件，可以监听用户的动作和折叠组件的状态。简单说明如下。

- show：在触发打开动作时立刻触发。
- shown：在折叠组件完全打开后触发（过渡效果完成后）。
- hide：在用户触发折叠动作时立刻触发。
- hidden：在折叠组件完全折叠后触发（过渡效果完成后）。

注意，show 和 hide 是监听动作的，shown 和 hidden 是监听状态的。

例如，针对上面的示例结构，添加如下 JavaScript 脚本，为折叠元素绑定两个事件，当显示折叠元素时，把它的背景色改为绿色，当隐藏折叠元素时，则取消背景色，演示效果如图 7-31 所示。

```
<script type="text/JavaScript">
$(function(){
    $(".accordion-body").collapse({
          parent:"#box",
          toggle:true
    });
    $(".accordion-body").on("show",function(e){
          $(e.target).css("background-color","green");
    });
    $(".accordion-body").on("hide",function(e){
          $(e.target).css("background-color","transparent");
    });
    $(".accordion-toggle").click(function(){
          $(this).parent().next().collapse("toggle");
    })
});
</script>
```

图 7-31　为折叠插件绑定事件

7.11 轮播

轮播是灯箱广告的一种样式，也是图片展示的一种方式。Bootstrap 提供的轮播插件比较简单，只能自动从左到右，而且与其他组件不一样，需要自己手动初始化一下，不过能够满足常规图片展示需要。

轮播插件需要 bootstrap-carousel.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-carousel.js 文件，或者 bootstrap.js 脚本文件。

7.11.1 使用轮播插件

轮播插件的结构也比较复杂，与折叠插件一样需要多层嵌套，并应用多个样式类才行。下面通过示例介绍轮播插件的一般用法。

第 1 步：在使用轮播插件之前，应在页面头部位置导入插件所需要的脚本文件和样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap-carousel.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

第 2 步：设计轮播包含框，定义 carousel 类样式，设计 ID 值，该值将作为 data-parent 属性的引用，以确保当前轮播插件中各种控制图标和按钮进行定位。在轮播外包含框内设计两个子容器，用来设计轮播标识图标框和轮播信息框，此时的折叠外壳效果如图 7-32 所示。


```
<div id="box" class="carousel slide">
    <ol class="carousel-indicators">
        <li data-target="#box" data-slide-to="0" class="active"></li>
        <li data-target="#box" data-slide-to="1"></li>
        <li data-target="#box" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner"> </div>
    <a class="left carousel-control" href="#box" data-slide="prev">&lsaquo;</a>
    <a class="right carousel-control" href="#box" data-slide="next">&rsaquo;</a>

</div>
```

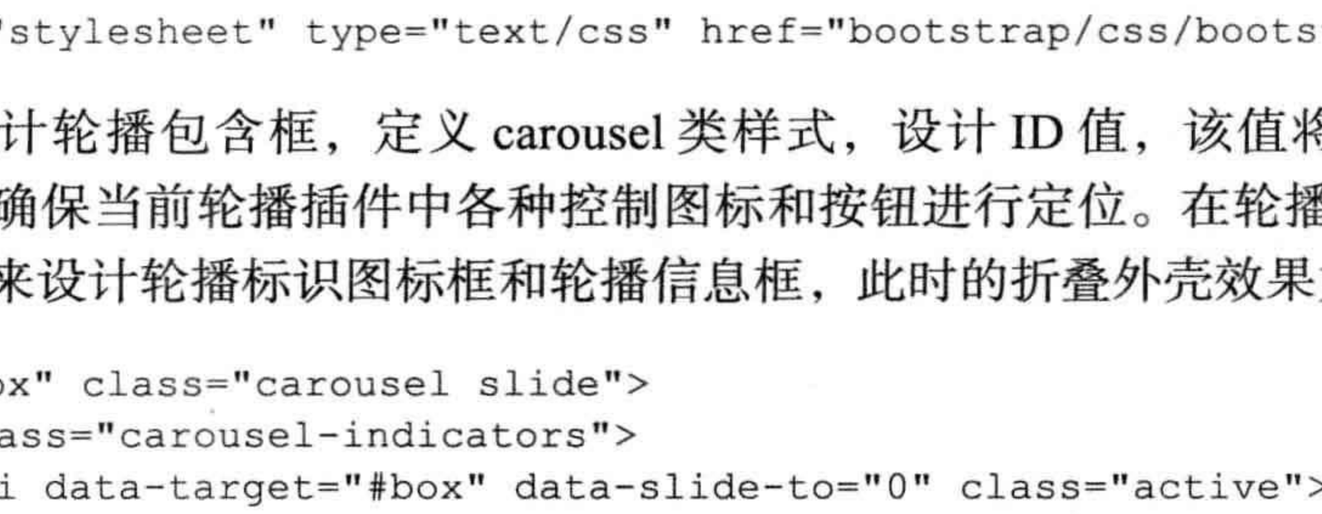


图 7-32 设计轮播外壳效果

<ol class="carousel-indicators"> 包含框定义了 3 个指示图标，显示当前图片的播放顺序，在这个列表结构中，使用 data-target="#box" 指定目标包含容器为 ，使用 data-slide-to="0" 定义播放顺序的下标。

 包含框准备放置要轮播的图片和说明文字，具体说明请参考下一步。在 轮播框最后面插入两个控制按钮：使用 carousel-control 定义按钮样式，left 和 right 定义按钮靠齐位置；使用 data-slide 定义按钮控制的行为方式，data-slide="prev" 表示向左滑动，data-slide="next" 表示向右滑动。

第 3 步：设计轮播的选项面板。每个轮播项目都包含在 子框中，每个项目包含两部分：第一部分是图片；第二部分是图片描述（ ）。一个完整的轮播项目结构如下：

```
<div class="item">
    <img src="images/1.jpg" alt="">
    <div class="carousel-caption">
        <h4> 标题 </h4>
        <p> 描述文本 </p>
    </div>
</div>
```

第 4 步：完成轮播整个结构的设计，完整的结构代码如下，则预览效果如图 7-33 所示。可以通过轮播外框的 CSS 样式控制轮播的显示空间。

```
<div id="box" class="carousel slide">
    <ol class="carousel-indicators">
        <li data-target="#box" data-slide-to="0" class="active"></li>
        <li data-target="#box" data-slide-to="1"></li>
        <li data-target="#box" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner">
        <div class="item active">
            <img src="images/1.jpg" alt="">
            <div class="carousel-caption">
                <h4> 宇宙 </h4>
                <p> 宇宙 (Universe) 是由空间、时间、物质和能量所构成的统一体。是一切空
                    间和时间的综合。一般理解的宇宙指我们所存在的一个时空连续系统，包括其
                    间的所有物质、能量和事件。根据大爆炸宇宙模型推算，宇宙年龄大约
                    138.2 亿年。</p>
            </div>
        </div>
        <div class="item">
            <img src="images/2.jpg" alt="">
            <div class="carousel-caption">
                <h4> 图片标题 2</h4>
                <p> 描述文本 2</p>
            </div>
    </div>
    <div class="item">
        <img src="images/3.jpg" alt="">
        <div class="carousel-caption">
            <h4> 图片标题 3</h4>
            <p> 描述文本 3</p>
        </div>
    </div>
```

```
        </div>
        <a class="left carousel-control" href="#box" data-slide="prev">&lsaquo;</a>
        <a class="right carousel-control" href="#box" data-slide="next">&rsaquo;</a>
    </div>
```

图 7-33　设计轮播演示效果

在默认状态下轮播是不会自动播放的，如果要轮播自动播放，需要使用 JavaScript 脚本进行激活。

7.11.2　控制轮播插件

调用轮播插件的方法也有两种，简单说明如下。

一种是通过 data 属性实现，data 属性可以很容易地控制轮播的位置。其中使用 data-slide 属性可以改变当前帧，该属性取值包括 prev、next，prev 表示向后滚动，next 表示向前滚动。另外，使用 data-slide-to 属性可以传递某个帧的下标，如 data-slide-to="2"，这样就可以直接跳转到这个指定的帧（下标从 0 开始计算）。

```
<div id="box" class="carousel slide">
    <ol class="carousel-indicators">
        <li data-target="#box" data-slide-to="0" class="active"></li>
        <li data-target="#box" data-slide-to="1"></li>
        <li data-target="#box" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner"> </div>
```

```
    <a class="left carousel-control" href="#box" data-slide="prev">&lsaquo;</a>
    <a class="right carousel-control" href="#box" data-slide="next">&rsaquo;</a>
</div>
```

另一种是通过 JavaScript 实现。JavaScript 调用轮播其实很简单，只需在脚本中调用 carousel() 方法即可。例如，继续以上面的示例为基础，介绍如何快速调用 JavaScript 脚本来启动轮播动画效果。

首先，先清理掉自定义的 data 属性，保留轮播组件的基本结构和样式类。

```
<div id="box" class="carousel slide">
    <ol class="carousel-indicators">
        <li class="active"></li>
        <li></li>
        <li></li>
    </ol>
    <div class="carousel-inner">
        <div class="item active">
            <img src="images/1.jpg" alt="">
            <div class="carousel-caption">
                <h4> 宇宙 </h4>
                <p> 宇宙 (Universe) 是由空间、时间、物质和能量所构成的统一体。是一切空间和时间的综合。一般理解的宇宙指我们所存在的一个时空连续系统，包括其间的所有物质、能量和事件。根据大爆炸宇宙模型推算，宇宙年龄大约 138.2 亿年。</p>
            </div>
        </div>
        <div class="item">
            <img src="images/2.jpg" alt="">
            <div class="carousel-caption">
                <h4> 图片标题 2</h4>
                <p> 描述文本 2</p>
            </div>
        </div>
        <div class="item">
            <img src="images/3.jpg" alt="">
            <div class="carousel-caption">
                <h4> 图片标题 3</h4>
                <p> 描述文本 3</p>
            </div>
        </div>
        <a class="left carousel-control" href="#">&lsaquo;</a>
        <a class="right carousel-control" href="#">&rsaquo;</a>
</div>
```

然后，在脚本中调用 carousel() 方法，并设置参数值为 'cycle'，表示循环播放图片。

```
<script type="text/JavaScript">
$(function(){
    $('.carousel').carousel('cycle')
});
</script>
```

carousel() 方法包含两个配置参数，简单说明如表 7-5 所示。

表 7-5 carousel() 方法配置参数 options 属性

名　称	类　型	默认值	描　述
interval	number	5000	在自动轮播过程中，展示每帧所停留的时间。如果是 false，轮播不会自动启动
pause	string	"hover"	当鼠标在轮播区域内时暂停循环，在区域外时则继续循环

上述参数可以通过 data 属性或 JavaScript 传递。对于 data 属性，将参数名称附着到 data- 之后，如 data-interval=""。例如，在下面的脚本中，定义轮播速度为 1 秒钟，然后以快速方式播放图片。

```
<script type="text/JavaScript">
$(function(){
     $('.carousel').carousel({
          interval:1000
     })
     $('.carousel').carousel('cycle')
});
</script>
```

实际上，当我们配置 carousel() 方法的配置参数之后，轮播插件就能够自动播放动画，因此也可以不用再次调用 $('.carousel').carousel('cycle') 方法。

.carousel() 方法还包含多种特殊调用，简单说明如下。

- .carousel('cycle')：从左向右循环播放。
- .carousel('pause')：停止循环播放。
- .carousel(number)：循环到指定帧，下标从 0 开始，类似数组。
- .carousel('prev')：返回到上一帧。
- .carousel('next')：下一帧。

例如，在上面的示例中通过 JavaScript 调用轮播动画播放，但是两侧的两个导航按钮还无法正确工作，下面通过 carousel（'prev'）和 carousel（'next'）方法让它们实现交互，代码如下：

```
<script type="text/JavaScript">
$(function(){
     $('.carousel').carousel({
          interval:1000
     });
     $("#box a.left").click(function(){
          $('.carousel').carousel("prev");
     })
     $("#box a.right").click(function(){
          $('.carousel').carousel("next");
     })
});
</script>
```

Bootstrap 在轮播插件中定义了两个事件，简单说明如下。

- slide：当 slide 实例方法被调用时，该事件会被立即触发。
- slid：当切换完一帧后触发。

例如，以上面的示例为例，设计当图片滑动过程时让轮播组件外框显示高亮边框线，滑过之后恢复默认效果，代码如下，演示效果如图 7-34 所示。

```
<script type="text/JavaScript">
$(function(){
    $('.carousel').carousel({
        interval:3000
    });
    $('.carousel').on("slide",function(e){
        e.target.style.border = "solid 2px red"
    });
    $('.carousel').on("slid",function(e){
        e.target.style.border = "solid 2px white"
    );
    $("#box a.left").click(function(){
        $('.carousel').carousel("prev");
    })
    $("#box a.right").click(function(){
        $('.carousel').carousel("next");
    })
});
</script>
```

图 7-34　设计轮播演示事件触发效果

7.12 输入提示

输入提示是一个表单输入辅助类型的插件，简单、易于扩展，可迅速地为表单中的文本输入框创建优雅的输入提示。

该插件需要 bootstrap-typeahead.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-typeahead.js 文件，或者 bootstrap.js 脚本文件。下面先看一个示例，了解输入提示插件的简单用法。

首先，在页面头部区域导入 Bootstrap 框文件，包括 JavaScript 脚本文件和 CSS 样式表文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

然后，在页面设计一个简单的文本输入框。在文本框中通过 data-provide="typeahead" 属性激活输入提示插件功能，使用 data-items="4" 属性设置每次最多提示的条数，使用 data-source 属性设置提示的数据源，以数组形式进行传递。

```
    <input type="text" class="span3" data-provide="typeahead" data-items="4" data-
source='[
    "CSS: Cascading Style Sheets, 层叠格式表 ",
    "CGI(Common Gateway Interface, 通用网关接口 )",
    "DCD: Document Content Description for XML: XML 文件内容描述 ",
    "DTD: Document Type Definition, 文件类型定义 ",
    "HTML(HyperText Markup Language, 超文本标记语言 )",
    "JVM: Java Virtual Machine, Java 虚拟机 ",
    "SGML: Standard Generalized Markup Language, 标准通用标记语言 ",
    "XML: Extensible Markup Language( 可扩展标记语言 )",
    "XSL: Extensible Style Sheet Language( 可扩展设计语言 )",
    "SMTP(Simple Mail Transfer Protocol, 简单邮件传输协议 )",
    "VPN: virtual private network, 虚拟局域网 "
    ]'>
```

最后，在浏览器中预览会发现，当输入字母 c 时，会立即弹出对应的提示性词条，效果如图 7-35 所示。

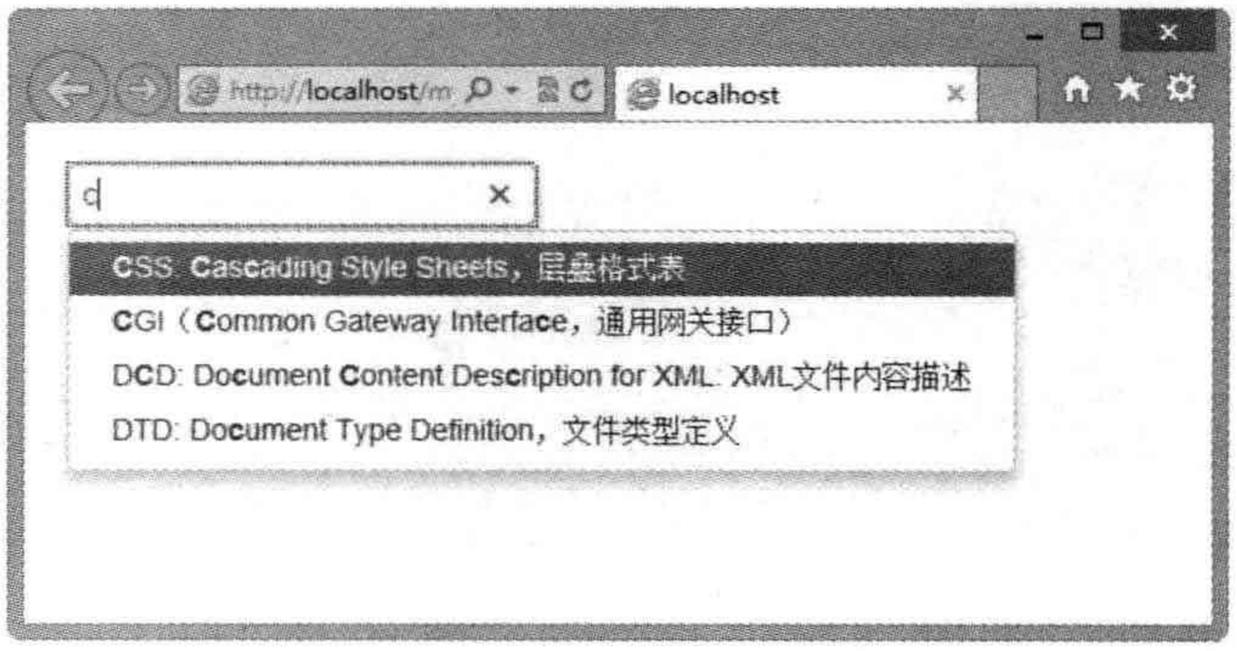


图 7-35 设计输入提示效果

输入提示插件其实就是其他 UI 库的自动完成，只要引入 JavaScript 就能用了。要求目标文本域至少有两个属性：data-provide="typeahead" 和 data-source。data-source 是一个经过 unescapeHTML 的字符串数组，不过还是建议使用 JavaScript 初始化后，动态更新 source 属性。

输入提示插件支持两种方法调用：一是通过添加 data 属性给页面元素注册输入提示功能，可参考上面的示例；二是通过 JavaScript 手动调用输入提示：

```
$('.typeahead').typeahead()
```

typeahead() 方法包含一个配置参数对象，选项说明如表 7-6 所示。

表 7-6　typeahead() 方法配置参数 options 属性

名　称	类　型	默认值	描　述
source	array, function	[ ]	用于查询的数据源。可以是一个字符串数组或是一个函数。函数会接收到两个参数，分别是输入域中的 query 值和 process 回调函数。函数可能会被同步调用，直接返回数据源；或者异步调用，通过 process 回调函数的唯一一个参数
items	number	8	下拉菜单中显示的最大条目数
minLength	number	1	触发提示所需的最小字符个数
matcher	function	case insensitive	该函数用于决定某个查询是否匹配某个条目。它接受唯一一个参数 item，表示当前需要测试的条目。使用 this.query 引用当前查询字符串。如果匹配查询，就返回一个布尔值 true
sorter	function	exact match, case sensitive, case insensitive	该函数用来排序提示项。它接受唯一一个参数 item，并且其变量范围在 typeahead 实例内。使用 this.query 引用当前查询字符串
updater	function	returns selected item	此方法用于返回选中的条目。它接受唯一一个参数 item，并且其变量范围在 typeahead 实例内
highlighter	function	highlights all default matches	该函数用来高亮自动完成的结果。它接受唯一一个参数 item，并且变量范围在 typeahead 实例内。应该返回 html

上述参数可以通过 data 属性或 JavaScript 传递。对于 data 属性，将参数名称附着到 data- 之后，如 data-source=""。

例如，针对上面的示例，通过 JavaScript 方式激活输入提示的代码如下：

```
<script type="text/JavaScript">
$(function(){
     var a = [
          "CSS: Cascading Style Sheets, 层叠格式表 ",
          "CGI(Common Gateway Interface, 通用网关接口 )",
          "DCD: Document Content Description for XML: XML 文件内容描述 ",
          "DTD: Document Type Definition, 文件类型定义 ",
          "HTML(HyperText Markup Language, 超文本标记语言 )",
```

```
            "JVM: Java Virtual Machine, Java虚拟机",
            "SGML: Standard Generalized Markup Language, 标准通用标记语言 ",
            "XML: Extensible Markup Language(可扩展标记语言)",
            "XSL: Extensible Style Sheet Language(可扩展设计语言)",
            "SMTP(Simple Mail Transfer Protocol, 简单邮件传输协议)",
            "VPN: virtual private network, 虚拟局域网 "
    ]
    $("input").typeahead({
            items:4,
            source:a
        })
    });
    </script>

    <input type="text" class="span3" >
```

7.13 附加导航

附加导航（affix）是 Bootstrap 2.1 新增的插件。这个插件其实很简单，就是通过为特定元素添加或移除 affix 类，实现在窗口中固定或不固定显示。当页面加载完毕时，附加导航插件会搜索页面上所有定义了 data-spy="affix" 的元素，然后找其 data-offset-top 或 data-offset-bottom 属性，即离页面顶部或者底部少于多少像素就会放弃固定，即移出 affix 类；当滚动条滚动页面超出这个偏移距离时，目标元素就会固定在窗口中指定位置不再滚动。

该插件需要 bootstrap-affix.js 文件支持，因此在使用该插件之前，应该导入 jquery.js 和 bootstrap-affix.js 文件，或者 bootstrap.js 脚本文件。下面先看一个示例，了解附加导航插件的简单用法。

第 1 步：在页面头部区域导入 Bootstrap 框文件，包括 JavaScript 脚本文件和 CSS 样式表文件。考虑到本示例不仅用到附加导航插件，还要使用滚动监听插件，因此建议直接导入 bootstrap.js 文件，而不是专项脚本文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/ bootstrap.js"></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

第 2 步：设计一个完整的页面结构，顶部是 Hero 单元（ ），用来显示网页标题信息。主体部分是一个流式布局（ ），左侧是一个导航菜单（<div class="span3"id="menu">），右侧是主体信息栏，<body> 标签内包含的代码如下，预览效果如图 7-36 所示。

```
<div class="hero-unit">
    <h1 class="">网页标题</h1>
    <a class="btn btn-large btn-success">更多</a>
</div>
<div class="row-fluid">
```

```
<div class="span3" id="menu" >
    <ul class="nav nav-list ">
        <li><a href="#1"><i class="icon-chevron-right"></i> 列表 1</a></li>
        <li><a href="#2"><i class="icon-chevron-right"></i> 列表 2</a></li>
        <li><a href="#3"><i class="icon-chevron-right"></i> 列表 3</a></li>
        <li><a href="#4"><i class="icon-chevron-right"></i> 列表 4</a></li>
        <li><a href="#5"><i class="icon-chevron-right"></i> 列表 5</a></li>
        <li><a href="#6"><i class="icon-chevron-right"></i> 列表 6</a></li>
        <li><a href="#7"><i class="icon-chevron-right"></i> 列表 7</a></li>
        <li><a href="#8"><i class="icon-chevron-right"></i> 列表 8</a></li>
    </ul>
</div>
<div class="span9">
    <fieldset id="1">
        <legend> 列表 1</legend>
        <div class="fieldset-content">
            <p><img src="images/1.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="2">
        <legend> 列表 2</legend>
        <div class="fieldset-content">
            <p> <img src="images/2.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="3">
        <legend> 列表 3</legend>
        <div class="fieldset-content">
            <p><img src="images/3.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="4">
        <legend> 列表 4</legend>
        <div class="fieldset-content">
            <p><img src="images/4.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="5">
        <legend> 列表 5</legend>
        <div class="fieldset-content">
            <p><img src="images/5.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="6">
        <legend> 列表 6</legend>
        <div class="fieldset-content">
            <p><img src="images/6.jpg"></p>
        </div>
    </fieldset>
    <fieldset id="7">
        <legend> 列表 7</legend>
        <div class="fieldset-content">
            <p><img src="images/7.jpg"></p>
        </div>
```

```
            </fieldset>
            <fieldset id="8">
                <legend>列表 8</legend>
                <div class="fieldset-content">
                    <p><img src="images/8.jpg"></p>
                </div>
            </fieldset>
        </div>
    </div>
```

图 7-36　设计的页面效果

第 3 步：设计滚动监听。在 <body> 标签上定义 data-spy="scroll" 属性，启动滚动监听插件，同时定义 data-target="#menu" 属性，设置监听对象为附加导航列表框。

```
<body  data-spy="scroll" data-target="#menu">
```

第 4 步：设计附加导航。附加导航是一种导航智能定位插件，它能够根据滚动条的位置，确实是执行定位显示，还是随文档流自然流动显示。在导航列表框上（<ul class="nav nav-list ">）定义 data-spy="affix"，启动附加导航插件，同时设置 data-offset-top="280" 属性，设置智能监控条件为距离顶部偏移位置为 280 像素，即当滚动条向下滚动，导航列表与顶部偏移距离开始大于 280 像素时，导航列表被固定在窗口左上角位置显示，否则允许随文档自由滚动。演示效果如图 7-37 所示。

```
<ul class="nav nav-list "  data-spy="affix" data-offset-top="280">
```

第 5 步：在页面自定义样式表中定义附加导航距离顶部位置为 20 像素，作为固定位置显示的精确值。

```
.nav-list { top:20px;}
```

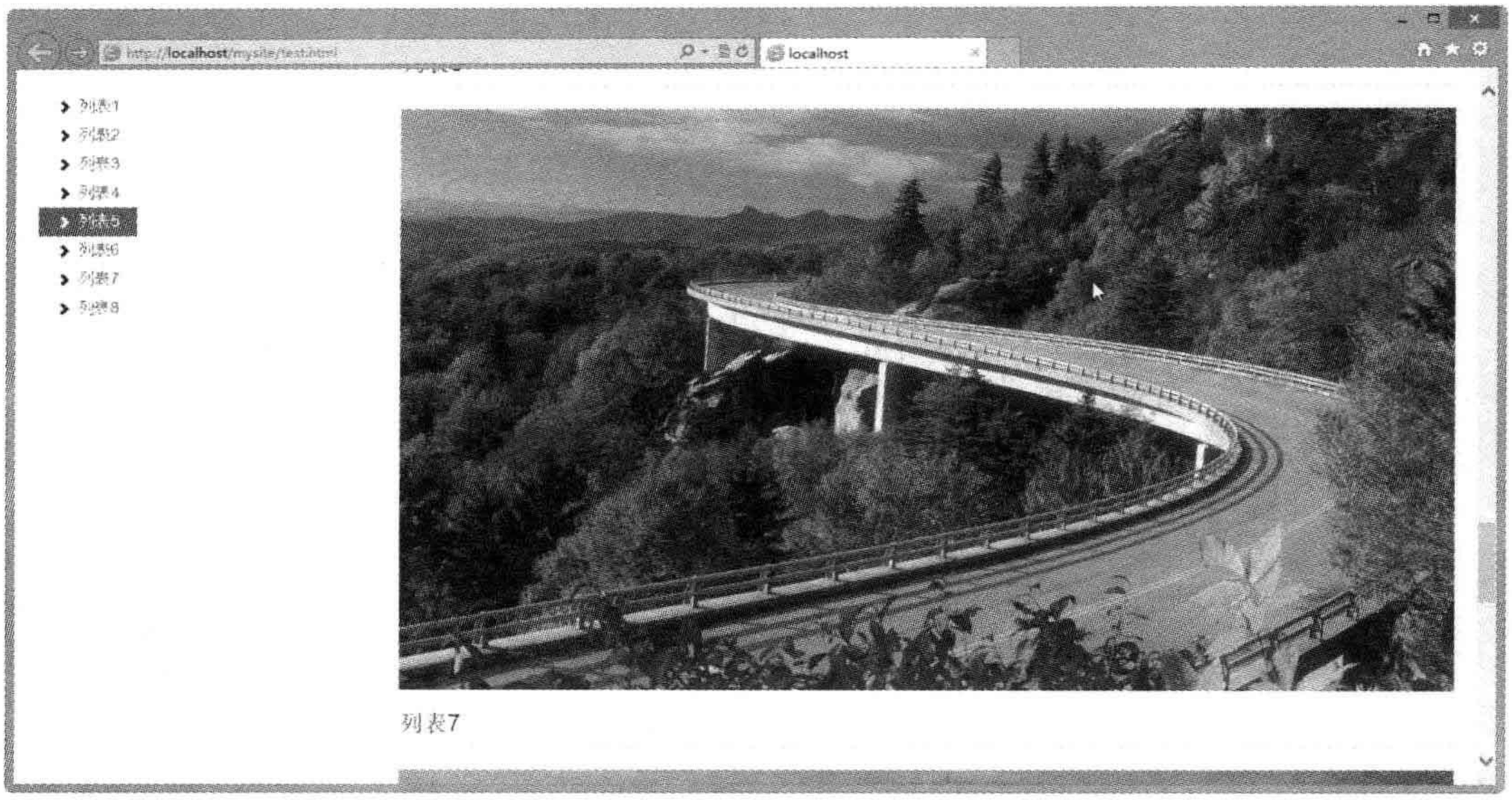


图 7-37　设计附加导航演示效果

附加导航插件也支持两种调用方式。

（1）通过 data 属性调用

只需添加 data-spy="affix" 到任意需要监听的页面元素上，就可以很容易地将其变为附加导航。然后使用 data-offset-top 偏移量来控制其位置。

```
<div data-spy="affix" data-offset-top="200"></div>
```

也可以通过 data-offset-bottom 偏移量来控制其位置，此时将根据目标元素距离底部的偏移位置进行计算。

（2）通过 JavaScript 调用 affix() 方法

例如，针对上面的示例可以按如下方法进行调用：

```
<script type="text/JavaScript">
$(function(){
    $(".nav-list").affix({
        offset:{top:280}
    })
});
</script>
```

affix() 方法包含一个配置参数 offset，用来确定智能定位的偏移位置，该属性值可以是数字、函数或者对象。当传递一个数值时，它将同时作用于 data-offset-top 和 data-offset-bottom 两个属性上，此时附加导航同时监听目标元素与页面顶部和底部的偏移距离。通过对象形式赋值，格式类似于 {top:280, bottom:120}。如果需要动态提供智能定位，可以使用函数作为值进行传递，在函数体内使用条件语句和循环语句，实现更复杂的智能监控。

第 8 章

Bootstrap 扩展

本章内容

- 针对 IE6、IE7 的 Bootstrap 扩展
- Bootstrap Metro
- 颜色选择器
- 日期选择器
- jQuery UI Bootstrap

Bootstrap 集合了 CSS、HTML 和 JavaScript，使用了最新的浏览器技术，为快速 Web 开发提供了一套前端工具包，包括布局、栅格、表格、按钮、表单、导航、提示等。使用 Bootstrap 可以构建出非常优雅的前端界面，而且占用资源非常少。虽然 Bootstrap 自带很多 JavaScript 插件，但是一些常用的控件却没有，如 Datepicker 等。好在 Bootstrap 提供了友好的扩展接口，允许开发人员扩展 Bootstrap。本章将介绍几款比较流行的 Bootstrap 扩展插件，帮助读者认识如何设计和使用 Bootstrap 外部插件，以弥补 Bootstrap 的不足之处。

8.1　针对 IE6、IE7 的 Bootstrap 扩展

Bootstrap 不支持 IE6、IE7 等 IE 浏览器的早期版本，但是 IE6 和 IE7 等早期浏览器在国内用户中占据重要市场份额，一般商业项目都会尊重市场，要求兼容 IE 早期版本，因此如何让 Bootstrap 兼容早期 IE 浏览器，是一件很重要的事情。

Bsie 是一个专为 Bootstrap 提供 IE6 兼容能力的插件，针对 IE6 问题提出了一种兼容性扩展方案，并以插件的形式进行发布。读者可以访问其官方网站（http://ddouble.github.io/bsie/）下载该插件。

目前，Bsie 能在 IE6 上支持 Bootstrap 2 的大部分常用特性，其支持的组件和特性说明见表 8-1。

表 8-1　Bsie 支持的 Bootstrap 特性说明

组　　件	支持特性
grid	fixed, fluid
navbar	top, fixed
nav	list, tabs, pills
dropdown	dropdown（两级）
buttons	button, group color, size, dropdown-button（不可用状态不是动态的）
form	default, horizontal, inline, all controls, validation state
tables	hover
pagination	all
labels	all
badges	all
code	all
modal	most
tooltip	all
popover	all
alert	all
typeahead	all
progressbar	most
media	all
wells	all
hero unit	all
icons	all

8.1.1 使用 Bsie 插件

使用 Bsie 扩展比较简单，只需要导入对应的 CSS 样式表和 JavaScript 脚本文件即可，具体操作步骤如下。

第 1 步：在文档头部区域导入 Bsie 样式表文件。

```
<!-- Bootstrap CSS 文件 -->
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.min.css">
<!--[if lte IE 6]>
<!-- Bsie CSS 补丁文件 -->
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap-ie6.css">
<!-- Bsie 额外的 CSS 补丁文件 -->
<link rel="stylesheet" type="text/css" href="bootstrap/css/ie.css">
<![endif]-->
```

第 2 步：在文档结尾位置导入 Bsie 脚本文件，也可以在文档头部导入，不过还是建议在文档尾部导入。

```
<!-- jQuery 1.7.2 or higher -->
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<!-- 可选 Bootstrap JavaScript library -->
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<!--[if lte IE 6]>
<!-- Bsie JavaScript 补丁文件，它仅能够在 IE6 中被解析 -->
<script type="text/JavaScript" src="js/bootstrap-ie.js"></script>
<![endif]-->
```

完成上面两步操作，就可以遵循 Bootstrap 组件的使用方法进行设计了。例如，针对第 7 章的工具提示插件，如果没有 Bsie 的支持，则在 IE6 中预览效果如图 8-1 所示。

如果按上面的操作方法导入 Bsie 补丁文件（bootstrap-ie6.css、ie.css、bootstrap-ie.js），那么在 IE6 下重新预览，则效果如图 8-2 所示。

图 8-1　工具提示插件在 IE6 下的默认效果

图 8-2　工具提示插件被修补后的效果

注意　当使用 AJAX 技术或其他方法动态添加 HTML 元素或内容，对这些动态对象应用 Bootstrap 组件时，必须使用如下方法调用 Bsie 补丁进行兼容：

```
if ($.isFunction($.bootstrapIE6)) $.bootstrapIE6(el);
```

目前，IE6 不支持嵌套的标签控制，因为 IE6 不支持 CSS 子元素选择器。因此，对于标签选择器等组件的应用还无法提供完善的解决方案。

8.1.2　手动修补 Bsie

Bsie 不能够解决 IE6 所有的不兼容问题，很多个别问题还需要用户手动解决，这虽然很痛苦，但是如果掌握了基本方法和途径，动手也不是那么困难。

1. 基本解决途径

除了使用 IE 条件单独为 IE6、IE7 等早期版本浏览器设计样式外，对于一些小的补丁，可以使用如下方法。

针对 IE6 的声明，只需在属性前面添加下划线前缀即可，例如：

```
_zoom:1;
```

针对 IE6 和 IE7 的声明，只需在属性前面添加星号前缀即可，例如：

```
*zoom:1;
```

2. 主要问题及修复方法

（1）问题 1：让 IE6 拥有布局特性

默认情况下，IE6 不支持所有行内元素都拥有布局特性，缺失布局特性的元素无法控制它的尺寸（width 和 height），因此可以使用如下方法让当前元素拥有布局特性，且这种方法没有污染性，不会影响其他浏览器的正常解析，因为 zoom 是 IE 的私有属性。

```
.container{
    zoom:1;
}
```

当然，用户也可以使用下面的声明让元素拥有布局特性，即触发 hasLayout：

```
position:                   absolute
float:                      left | right
display:                    inline-block
width:                      except 'auto'
height:                     except 'auto'
zoom:                       except 'normal'
overflow:                   hidden | scroll | auto
overflow-x/-y:              hidden | scroll | auto
position:                   fixed
min-width:                  any value
max-width:                  except 'none'
min-height:                 any value
max-height:                 except 'none'
writing-mode:               tb-rl   /* only for MS */
```

而下面的声明将会让元素失去布局特性，即清除 hasLayout：

```
width:                      auto;
```

```
height:                 auto;
max-width:              none;    /* IE7 */
max-height:             none;    /* IE7 */
position:               static;
float:                  none;
overflow:               visible;
zoom:                   normal;
writing-mode:           lr-t;
```

例如，在下面的样式中，元素不会被设置为 hasLayout=false：

```
.element {
    display:inline-block;
    display:inline;
}
```

（2）问题 2：让 IE6 支持行内块状显示

IE6 不支持 inline-block 属性值，因此要设置行内元素块状显示，以方便定义它们的大小，则可以使用下面的方法进行兼容：

```
.container {
    zoom:1;
    display:inline;
}
```

（3）问题 3：让 IE6 支持透明色

```
.element{
      border-color:pink/* 很少使用的颜色 */;
      filter:chroma(color:pink);
}
```

注意，这种方法容易导致绝对定位元素在相对定位包含框中消失。

（4）问题 4：网页背景色

```
body { /* 这种方法很容易失败，因为整个页面并不是显示灰色背景 */
    background-color: gray;
}
* html { /* 使用这种方法进行修补，可以让整个页面背景都显示为灰色 */
    background-color: gray;
}
```

（5）问题 5：下拉菜单

在 IE6、IE7 中使用下拉菜单时，应该为 ul.dropdown-menu 样式添加如下声明，即显式定义下拉菜单的宽度，具体宽度值应该根据实例而定。

```
*width:180px;
```

8.2 Bootstrap Metro

Metro UI 是一种界面设计技术，是 Windows 8 的主要界面显示风格。Metro 界面与 iOS、

Android 界面风格完全不同，iOS、Android 界面都是以应用为主要呈现对象，而 Metro 界面强调的是信息本身，而不是冗余的界面元素。同时在视觉效果方面，Metro 界面有助于形成一种身临其境的感觉。

基于 Bootstrap 框架的 Metro UI 插件很多，风格也比较相似。下面介绍几款同类 UI 插件，读者可以根据网站项目或者 Web 应用程序需要，选用其中的 Windows 8 Metro 风格，这些流行的 Metro UI 风格的 Bootstrap 主题和模板一定能够帮助用户快速设计风格鲜明的网站效果。

1. BootMetro

BootMetro 是基于 Bootstrap 的简单灵活的 HTML、CSS 和 JavaScript 框架，提供了导航、面板、表单、表格、图标等组件，效果如图 8-3 所示。

预览页面：http://aozora.github.io/bootmetro/hub.html。

下载页面：http://aozora.github.io/bootmetro/。

图 8-3　BootMetro UI

2. Bootswatch

Bootswatch 也是基于 Bootstrap 的开发框架，提供了按钮、表单、表格、标签、进度条等组件，效果如图 8-4 所示。

预览和下载页面：http://bootswatch.com/cosmo/。

3. Metro UI CSS

Metro UI CSS 是一个样式包，用于创建 Windows 8 Metro 风格的 UI，效果比较漂亮，如图 8-5 所示。

预览页面：http://metroui.org.ua/。

下载页面：https://github.com/olton/Metro-UI-CSS/archive/master.zip。

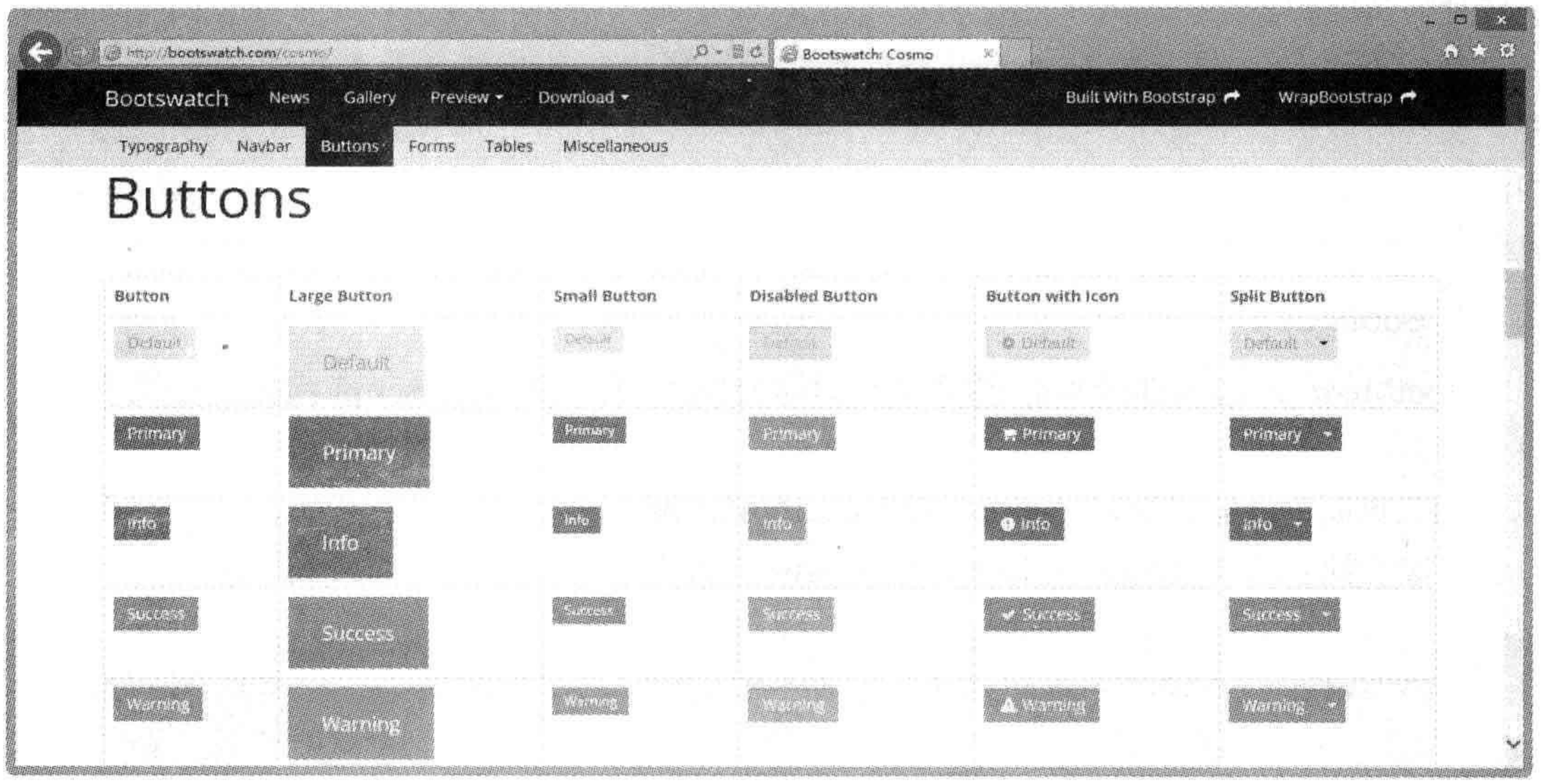


图 8-4　Bootswatch UI

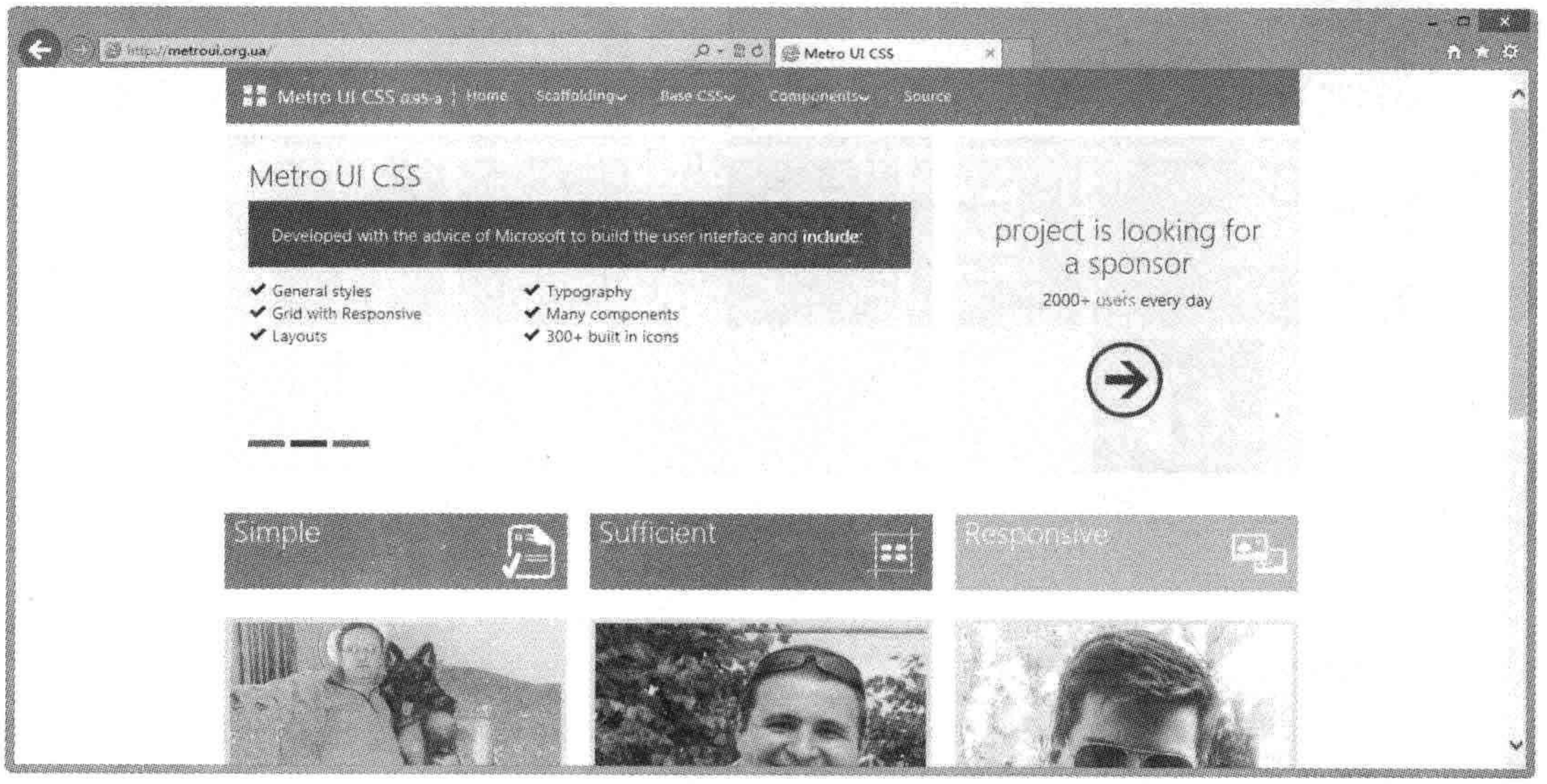


图 8-5　Metro UI CSS

4. MelonHTML5

MelonHTML5 是基于 HTML5、CSS3 和 JavaScript 的 Metro UI 框架，可以通过配置参数满足实际的需要，如图 8-6 所示。

预览页面：http://www.melonhtml5.com/#。

下载页面：http://codecanyon.net/item/melonhtml5-metro-ui/2986068。

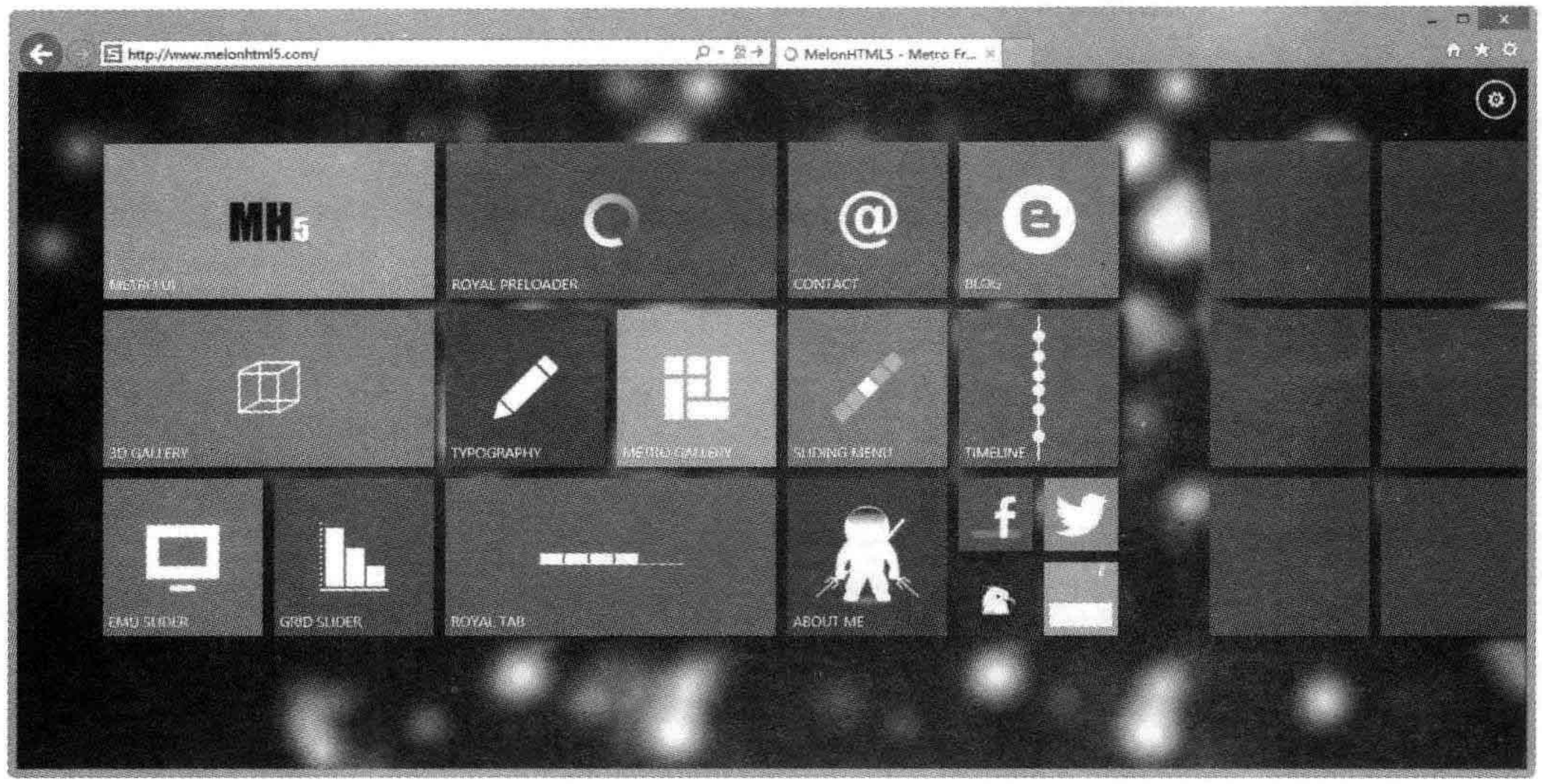


图 8-6　MelonHTML5

5. Metro Mania

Metro Mania 是专业的、响应式的 Bootstrap Metro 风格主题，有 6 套颜色可供选择，如图 8-7 所示。

预览和下载页面：http://responsivewebinc.com/premium/metro/。

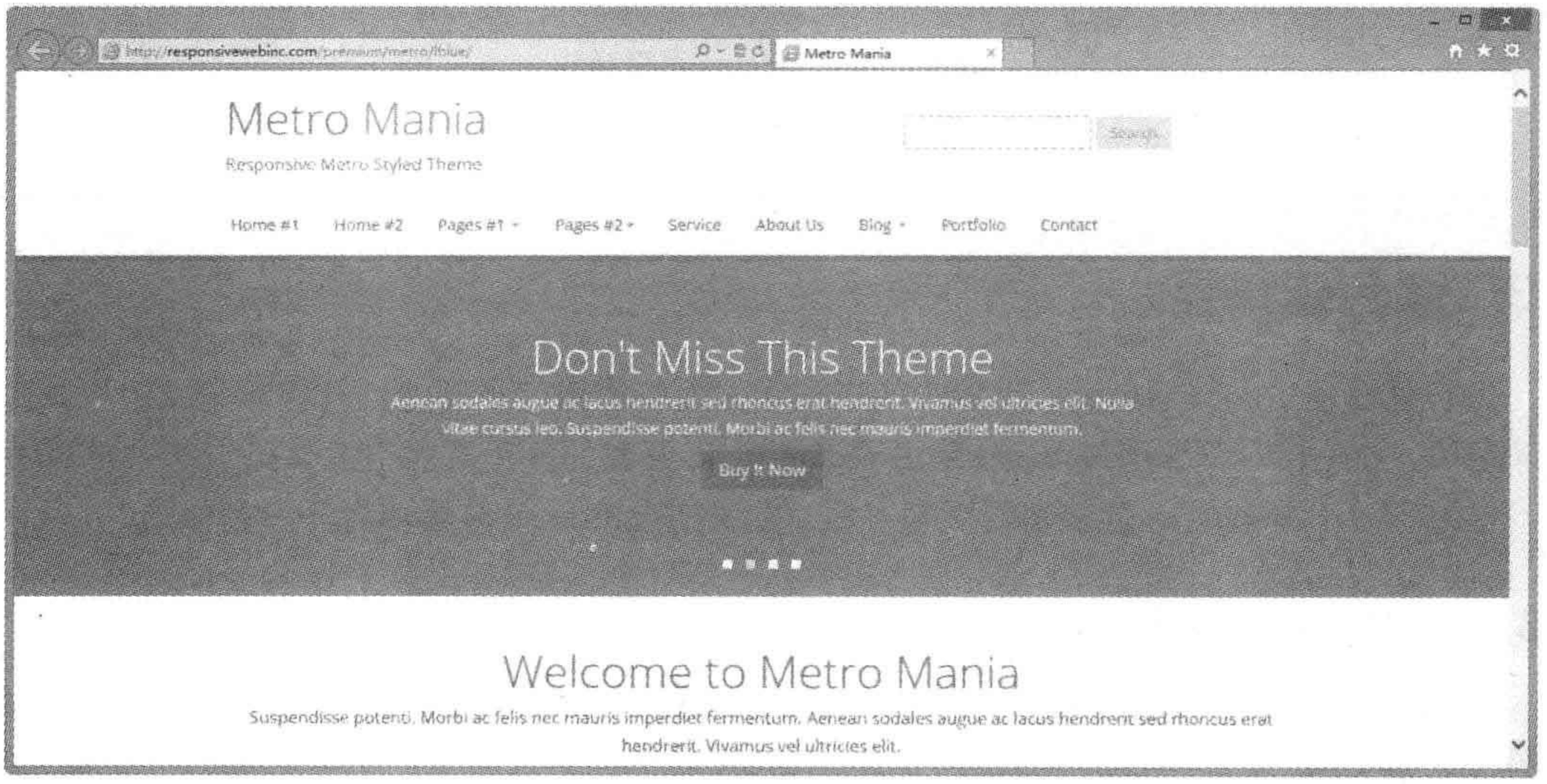


图 8-7　Metro Mania

下面以 BootMetro 插件为例，演示如何应用 Bootstrap Metro 效果。

第 1 步：下载 BootMetro 插件压缩包，解压之后在页面中引入 bootmetro.css 样式表文件，并且在其前面引入 bootstrap.css 样式表文件。

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

第 2 步：利用 BootMetro 设计表格样式。此时只需要在表格标签（<table>）中引入对应样式类即可。

```
<table class="table table-condensed ">
    <thead>
        <tr><th> 列标题 1</th><th> 列标题 2</th> <th> 列标题 3</th></tr>
    </thead>
    <tbody>
        <tr><td> 数据 11</td><td> 数据 12</td><td> 数据 13</td></tr>
        <tr><td> 数据 21</td><td> 数据 22</td><td> 数据 23</td></tr>
    </tbody>
</table>
```

第 3 步：在浏览器中预览效果，如图 8-8 所示。如果改变表格风格，只需要换一种样式类即可，如把 table-condensed 替换为 table-bordered，即把水平数据显示风格切换为垂直数据显示风格。

图 8-8　BootMetro 表格样式

其他网页风格设计可以参阅官方网站提供的效果预览页面。

8.3　颜色选择器

颜色选择器（color picker）是 Web 应用程序中比较实用的小插件，它能够帮助用户快速设置颜色值。Colorpicker for Bootstrap 是由 Stefan Petre 开发的颜色选择器插件，它基于 Bootstrap 技术框架，沿用 Bootstrap 的使用规则和习惯，非常适合 Bootstrap 学习者学习和使用。

8.3.1　使用颜色选择器

颜色选择器能够为文本输入框或其他任意元素添加颜色选择功能，作为 Bootstrap 的一个组件，它支持 HEX、RGB、RGBA、HSL、HSLA 等多种格式。下面结合示例进行简单介绍。

第 1 步：下载颜色选择器插件。访问 http://www.eyecon.ro/bootstrap-colorpicker/ 页面下

载 Colorpicker for Bootstrap（以下简称 Colorpicker）。

第 2 步：在页面头部导入 jQuery 和 Colorpicker 脚本文件（bootstrap-colorpicker.js），同时导入 Bootstrap 和 Colorpicker 样式表文件，代码如下：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js" ></script>
<script type="text/JavaScript" src="colorpicker/js/bootstrap-colorpicker.js" ></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="colorpicker/img/colorpicker.css">
```

注意　Colorpicker 插件的样式表文件中需要用到几个图片背景文件，但是在解压后直接引用时会出现异常，因为样式表文件 colorpicker.css 所使用的路径是根路径，在应用时样式表可能会找不到图像源文件，建议读者手动修改样式表中的引用路径，局部样式代码如下：

```
.colorpicker-saturation {
    width: 100px;
    height: 100px;
    background-image: url(saturation.png);
    cursor: crosshair;
    float: left;
}
```


第 3 步：设计一个颜色选择器包含框（ ），通过 data-color-format="rgb" 属性设置颜色格式为 RGB，通过 data-color="rgb（255，146，180）" 属性设置默认颜色值。在颜色选择器包含框中插入一个文本框，同时绑定一个触发按钮（）。

```
<div class="input-append color" data-color="rgb(255, 146, 180)" data-color-format="rgb">
    <input type="text" class="span2" value="" >
    <span class="add-on"><i style="background-color: rgb(255, 146, 180)"></i></span>
</div>
```

第 4 步：设计脚本，激活颜色选择器。在浏览器中预览，当单击颜色按钮时，会弹出一个颜色选择器面板，在其中选择一种颜色，该颜色值会被自动转换为 RGB 格式，并显示在文本框中，效果如图 8-9 所示。

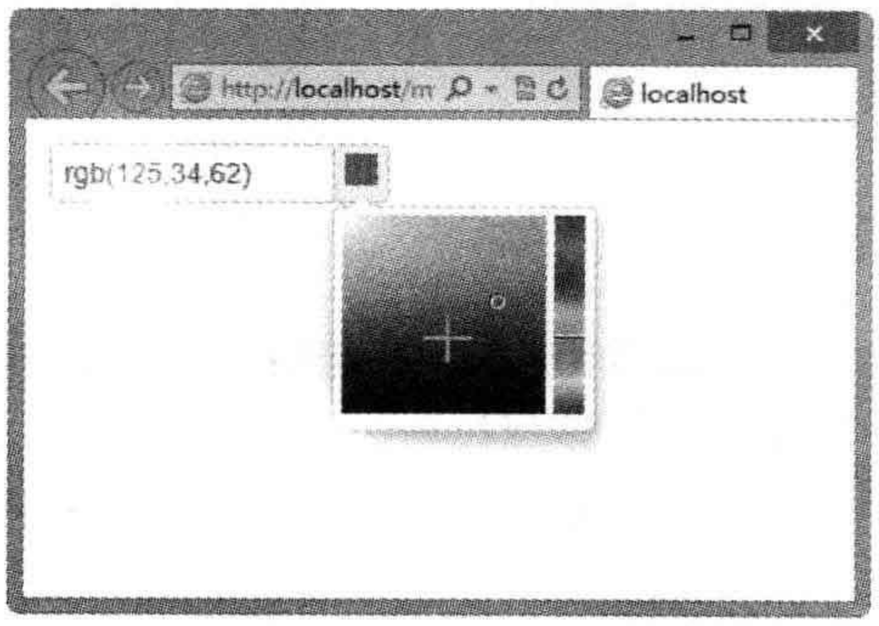


图 8-9　颜色选择器效果

```
<script type="text/JavaScript">
$(function(){
    $('.color').colorpicker();
});
</script>
```

8.3.2　配置颜色选择器

颜色选择器插件支持 JavaScript 调用方法，没有提供 data 属性调用方法。使用 colorpicker() 可以调用颜色选择器面板，同时可以附加配置参数。

“使用 colorpicker() 方法的参数对象可以设置 format 属性，该属性值为字符串，对应不同的颜色格式，默认为 'hex'，还可以设置为 rgb 或 rgba。例如，针对 8.3.1 节的示例，我们也可以通过脚本方法定义颜色选择器，代码如下：

```
<script type="text/JavaScript">
$(function(){
    $('.color').colorpicker({
        format:"hex"
    })
});
</script>
<div class="input-append color" data-color="rgb(255, 146, 180)">
    <input type="text" class="span2" value="" >
    <span class="add-on"><i></i></span>
</div>
```

data-color 属性是一个不可或缺的属性，使用它可以绑定文本框，用来设置颜色选择器面板。

如果直接为文本框设计颜色选择器，用法就非常简单，不需要过多设置。例如，在下面的代码中直接在文本框上调用 colorpicker() 方法即可，演示效果如图 8-10 所示。

```
<script type="text/JavaScript">
$(function(){
    $('input').colorpicker();
});
</script>

<input type="text" class="span2" value="" >
```

图 8-10　快速调用颜色选择器

如果设置 input 为只读，则在文本框上调用颜色选择器依然有效，演示效果如图 8-11 所示。

```
<script type="text/JavaScript">
$(function(){
    $('input').colorpicker();
});
</script>
<input type="text" class="span2" value=""
readonly>
```

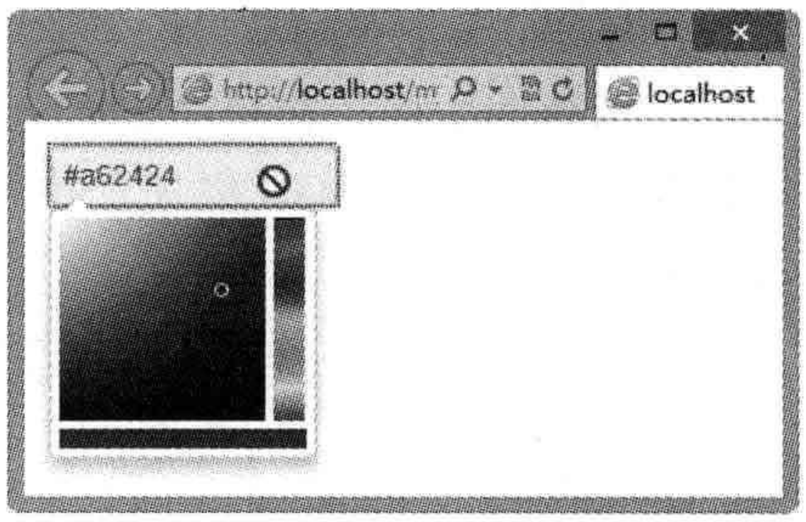


图 8-11　只读状态调用颜色选择器

如果希望文本框在只读或不可用状态下仅能够接收值，不能够直接触发，则建议使用颜色选择器包含框进行按钮调用，代码如下，演示效果如图 8-12 所示。

```
<script type="text/JavaScript">
$(function(){
    $('.color').colorpicker();
});
```

```
</script>

<div class="input-append color" data-color="rgb(255, 146, 180)">
    <input type="text" class="span2" value=""  readonly disabled>
    <span class="add-on"><i></i></span>
</div>
```

colorpicker() 方法除了接收一个参数对象（.colorpicker（options）），以便初始化颜色选择器配置外，还支持几个专用方法，用来显示或者隐藏颜色选择器。

- .colorpicker('show')：显示颜色选择器。
- .colorpicker('hide')：隐藏颜色选择器。
- .colorpicker('place')：更新颜色选择器相对于元素的位置。
- .colorpicker('setValue', value)：为颜色选择器设置一个新值，该方法可以触发 changeColor 事件。

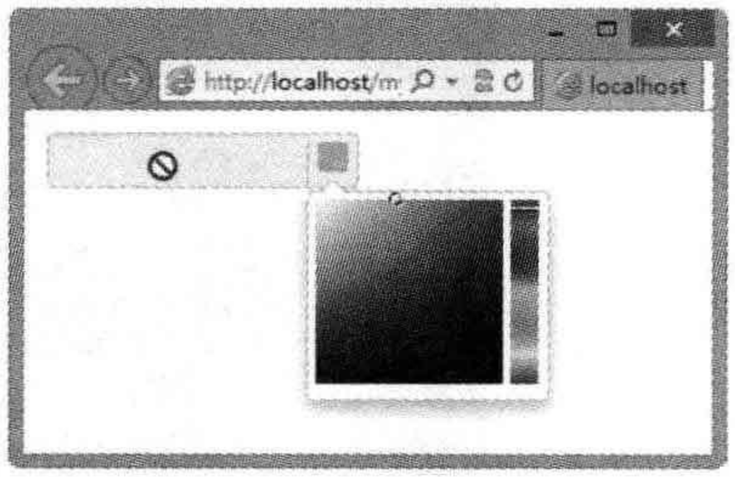


图 8-12　通过按钮调用颜色选择器

Colorpicker 支持事件处理，用来公开操作色彩，主要包括 3 个事件，说明如下。

- show：当颜色选择器显示时触发该事件。
- hide：当颜色选择器隐藏时触发该事件。
- changeColor：当颜色值发生改变时触发该事件。

每一个触发事件都有一个内部的颜色选择器对象（Color）。该对象提供了几个非常有用的方法，利用这些方法能够完成各种操作。

- .setColor(value)：设置颜色值，该值将被解析并转换为特定格式。
- .setHue(value)：设置色调，取值范围为 0 ~ 1。
- .setSaturation(value)：设置饱和度，取值范围为 0 ~ 1。
- .setLightness(value)：设置亮度值，取值范围为 0 ~ 1。
- .setAlpha(value)：设置不透明度值，取值范围为 0 ~ 1。
- .toRGB()：返回 RGB 颜色值，以散列表格式返回（red、green、blue 和 alpha）。
- .toHex()：返回 HEX 颜色值，以字符串形式返回。
- .toHSL()：返回 HSL 颜色值，以散列表格式返回。

例如，下面的示例演示了改变颜色选择器的值，将该值应用到 body 背景上的效果，如图 8-13 所示。

```
<script type="text/JavaScript">
$(function(){
    $('.color').colorpicker({
        format:"hex"
    }).on("changeColor",function(e){
        $('body')[0].style.backgroundColor = e.color.toHex();
    });
```

```
});
</script>

<div class="input-append color" data-color="rgb(255, 146, 180)">
    <input type="text" class="span2" value="">
    <span class="add-on"><i></i></span>
</div>
```

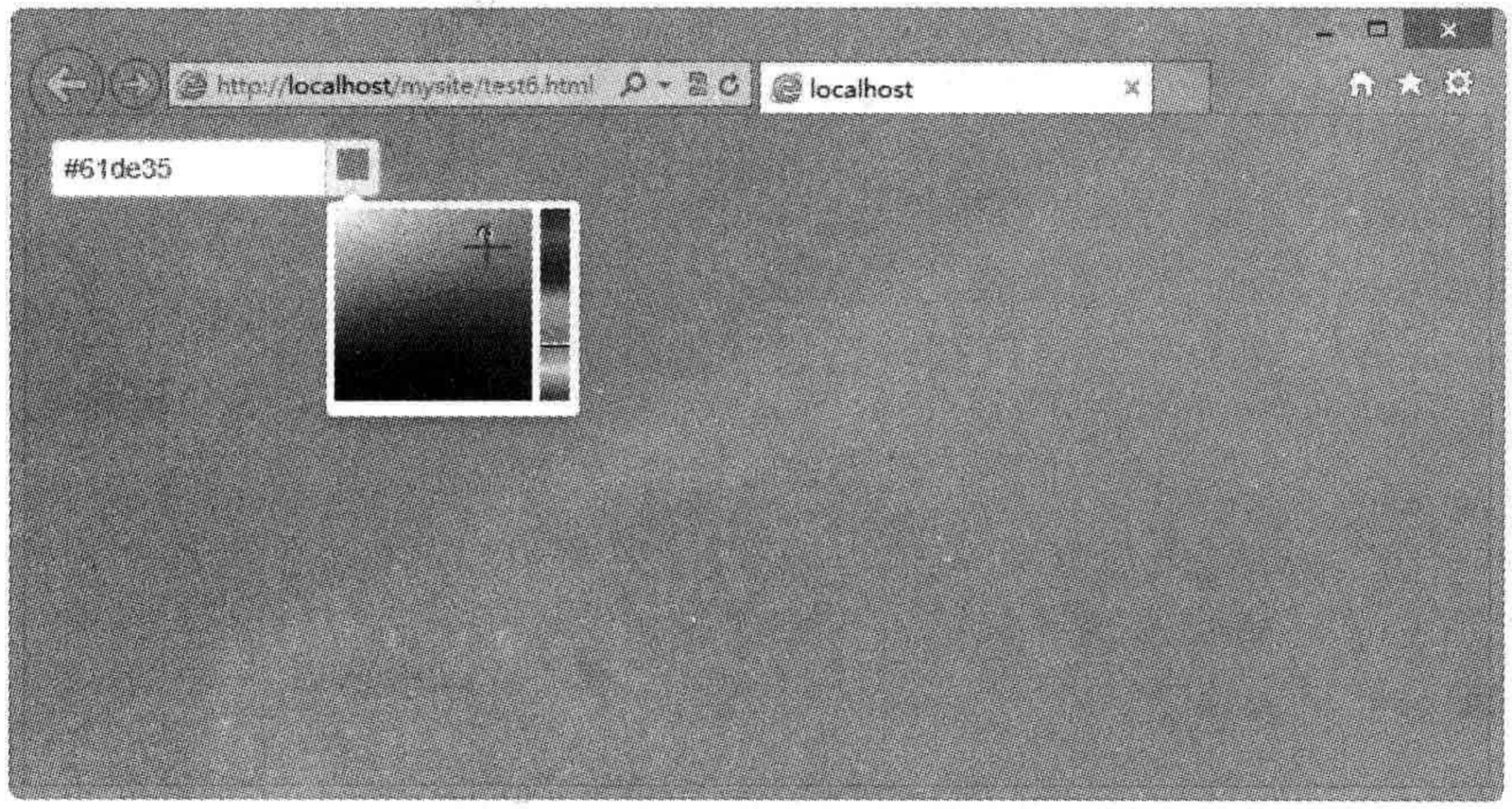


图 8-13　通过颜色选择器改变页面背景色

8.4　日期选择器

日期选择器（date picker）也是 Web 应用程序中比较实用的小插件，它能够帮助用户快速设置日期值。Date picker for Bootstrap 是由 Stefan Petre 开发的日期选择器插件，它基于 Bootstrap 技术框架，沿用 Bootstrap 的使用规则和习惯，非常适合 Bootstrap 学习者学习和使用。

Datepicker for Bootstrap 允许为文本框或其他任意元素添加日期选择功能，支持格式包括 DD、D、MM、M、YYYY、YY，其中 D 表示日期值，M 表示月份值，Y 表示年份值，年月日值之间用连字符（-）或斜线（/）分隔。

8.4.1　使用日期选择器

日期选择器能够为文本输入框或其他任意元素添加日期选择功能，作为 Bootstrap 的一个组件，其功能比较灵活，能够显示不同格式的日期以及日期范围。下面结合示例进行简单介绍。

第 1 步：下载日期选择器插件。访问 http://www.eyecon.ro/bootstrap-datepicker/ 页面下载 Datepicker for Bootstrap（以下简称 "Datepicker"）。

第 2 步：在页面头部导入 jQuery 和 Datepicker 脚本文件（bootstrap-datepicker.js），同时

导入 Bootstrap 和 Datepicker 样式表文件，代码如下：

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js" ></script>
<script type="text/JavaScript" src="datepicker/js/bootstrap-datepicker.js" ></script>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="datepicker/css/datepicker.css">
```

第 3 步：设计一个日期选择器文本框。定义 value 属性，设置值为当前日期。

```
<input type="text" class="span2" value="06/15/2013" id="date" >
```

第 4 步：设计脚本，激活日期选择器。在浏览器中预览，当单击文本框时，会弹出一个日期选择器面板，在其中选择一个日期，该日期值会自动显示在文本框中，效果如图 8-14 所示。

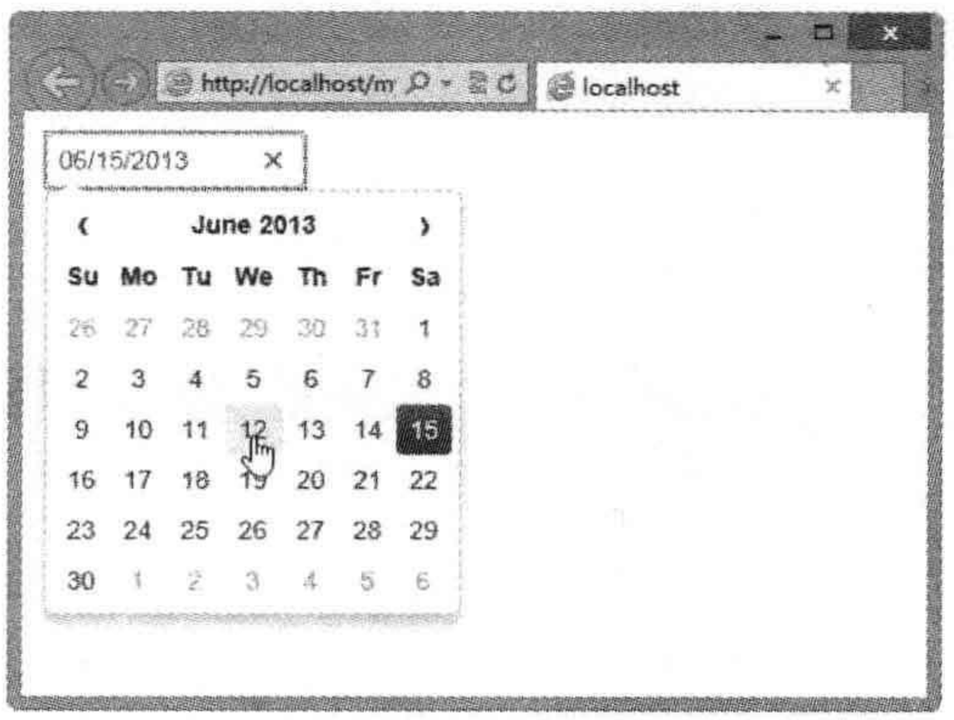


图 8-14　日期选择器效果

```
<script type="text/JavaScript">
$(function(){
    $('#date').datepicker();
});
</script>
```

8.4.2　配置日期选择器

日期选择器插件与 Bootstrap 其他插件用法相同，datepicker() 方法可以包含一个参数对象，用来设置日期选择器的显示样式和执行功能，具体说明如表 8-2 所示。

表 8-2　datepicker() 方法配置参数

名　称	类　型	默认值	描　述
format	string	'mm/dd/yyyy'	设置日期格式，包含这些组合：d、dd、m、mm、yy、yyy，其中 d 表示一位日期值，dd 表示两位日期值，m 表示一位月份值，mm 表示两位月份值，yy 表示两位年份值，yyyy 表示四位年份值
weekStart	integer	0	设置每周的开始值，0 表示周日，6 表示周六
viewMode	string|integer	0 = 'days'	设置开始视图模式，取值包括 0='days'、1='months'、2='years'
minViewMode	string|integer	0 = 'days'	设置视图限制模式，最小显示的级别值，取值包括 0='days'、1='months'、2='years'

例如，在下面的示例中通过 format 配置参数设置日期的格式为 "mm-dd-yyyy"，显示效果如图 8-15 所示。

```
<script type="text/JavaScript">
$(function(){
    $('#date').datepicker({
        format:"mm-dd-yyyy"
    });
});
```

```
</script>

<input type="text" class="span2" value="06/13/2013" id="date" >
```

除了使用 JavaScript 脚本配置日期选择器外，也可以使用 data 属性设置日期格式。例如，对于上面的示例，可以使用下面的代码进行设置：

```
<input type="text" class="span2" value="06/13/2013" data-date-format="mm-dd-
 yyyy" id="date" >
```

日期格式的设置可以任意组合，例如，我们也可以设置为年日月的顺序进行显示，代码如下，预览效果如图 8-16 所示。

```
<input type="text" class="span2" value="2013/13/06" data-date-format="yyyy-dd-
 mm" id="date" >
```

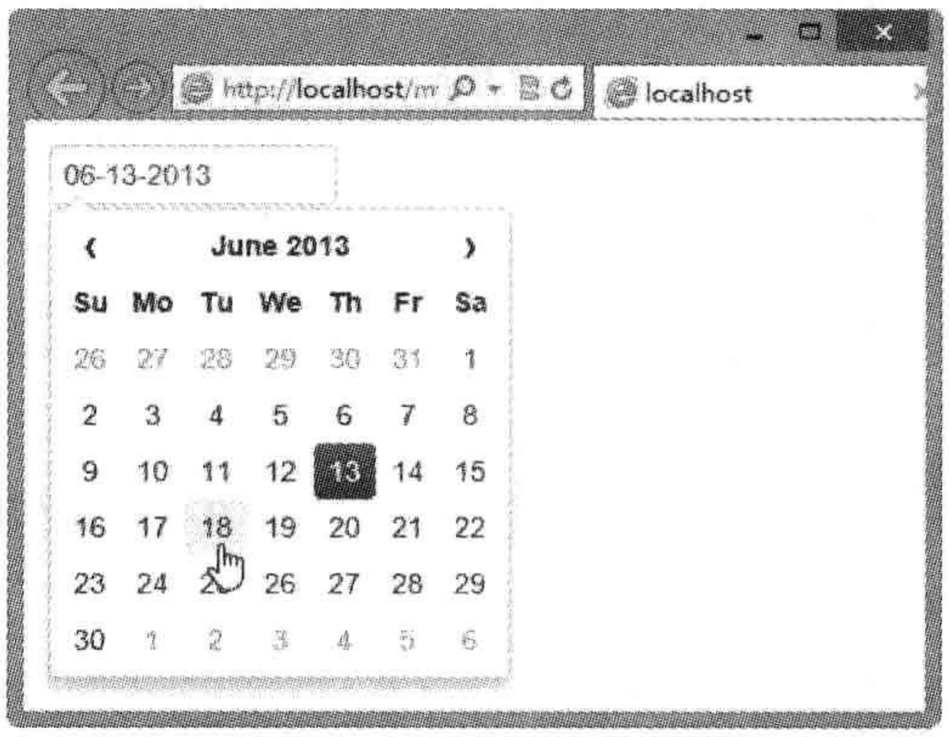


图 8-15　设置日期格式

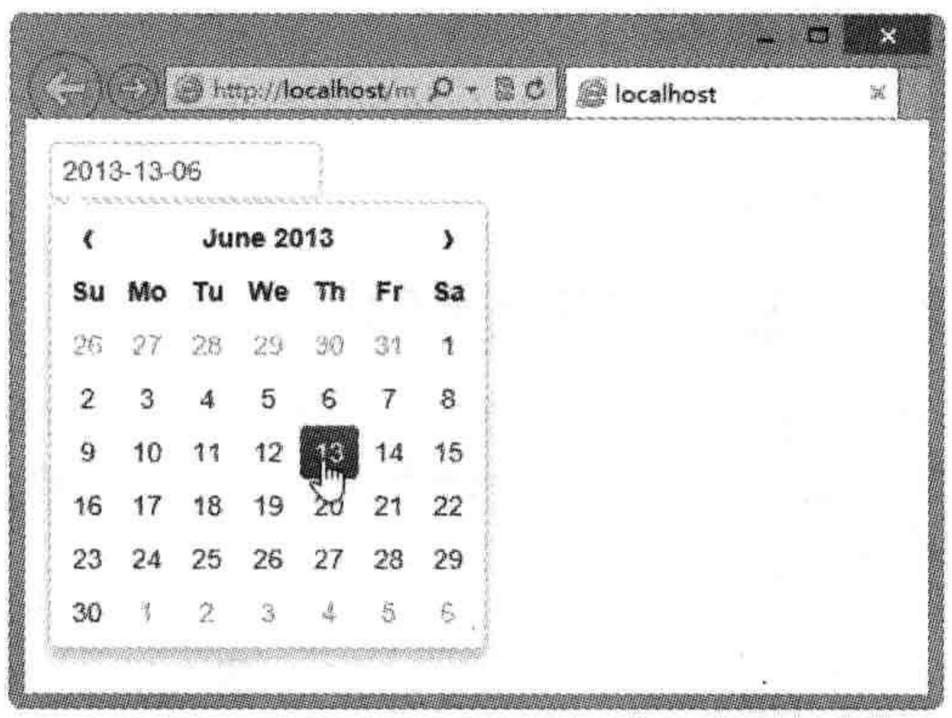


图 8-16　自定义日期格式显示

也可以为日期组件绑定一个触发按钮，按钮与文本框通过一个包含框捆绑在一起，具体格式代码如下：

```
<div class="input-append date" id="date" data-date="31-01-2013" data-date-
 format="dd-mm-yyyy">
    <input class="span2" size="16" type="text" value="31-01-2013" readonly>
    <span class="add-on"><i class="icon-calendar"></i></span>
</div>
```

设计一个日期选择器包含框（ ），通过 data-date-format="dd-mm-yyyy" 属性设置日期格式为“日 – 月 – 年”，通过 data-date="31-01-2013" 属性设置默认当前日期值。在日期选择器包含框中插入一个文本框，同时绑定一个触发按钮（）。

然后，通过脚本调用日期选择器，代码如下，预览效果如图 8-17 所示。

```
<script type="text/JavaScript">
```

```
$(function(){
    $('#date').datepicker();
});
</script>
```

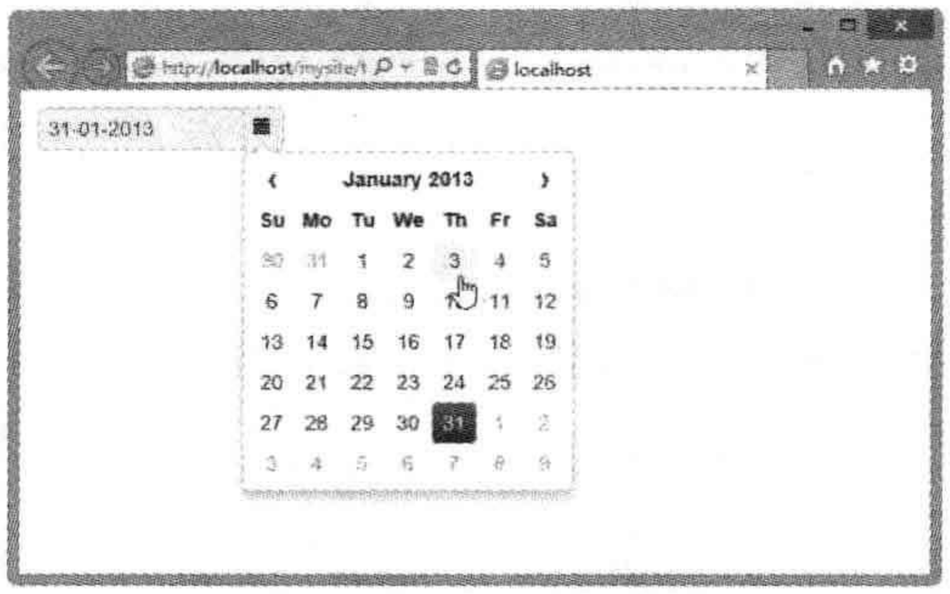


图 8-17　自定义日期格式显示

使用 viewMode 属性可以设置日期选择器面板的视图模式，即日期选择器面板最初显示的视图，默认为日期选择器视图，下面的代码可以设置视图开始模式为年，脚本如下，也可以传递"years"字符串，即 viewMode:"years"。

```
<script type="text/JavaScript">
$(function(){
    $('#date').datepicker({
        viewMode:2
    });
});
</script>
```

在页面中预览，首先看到一个年份选择视图，从中单击选择一个年份，然后显示第二个视图，选择月份，最后在第三个视图中选择日期，效果如图 8-18 所示。

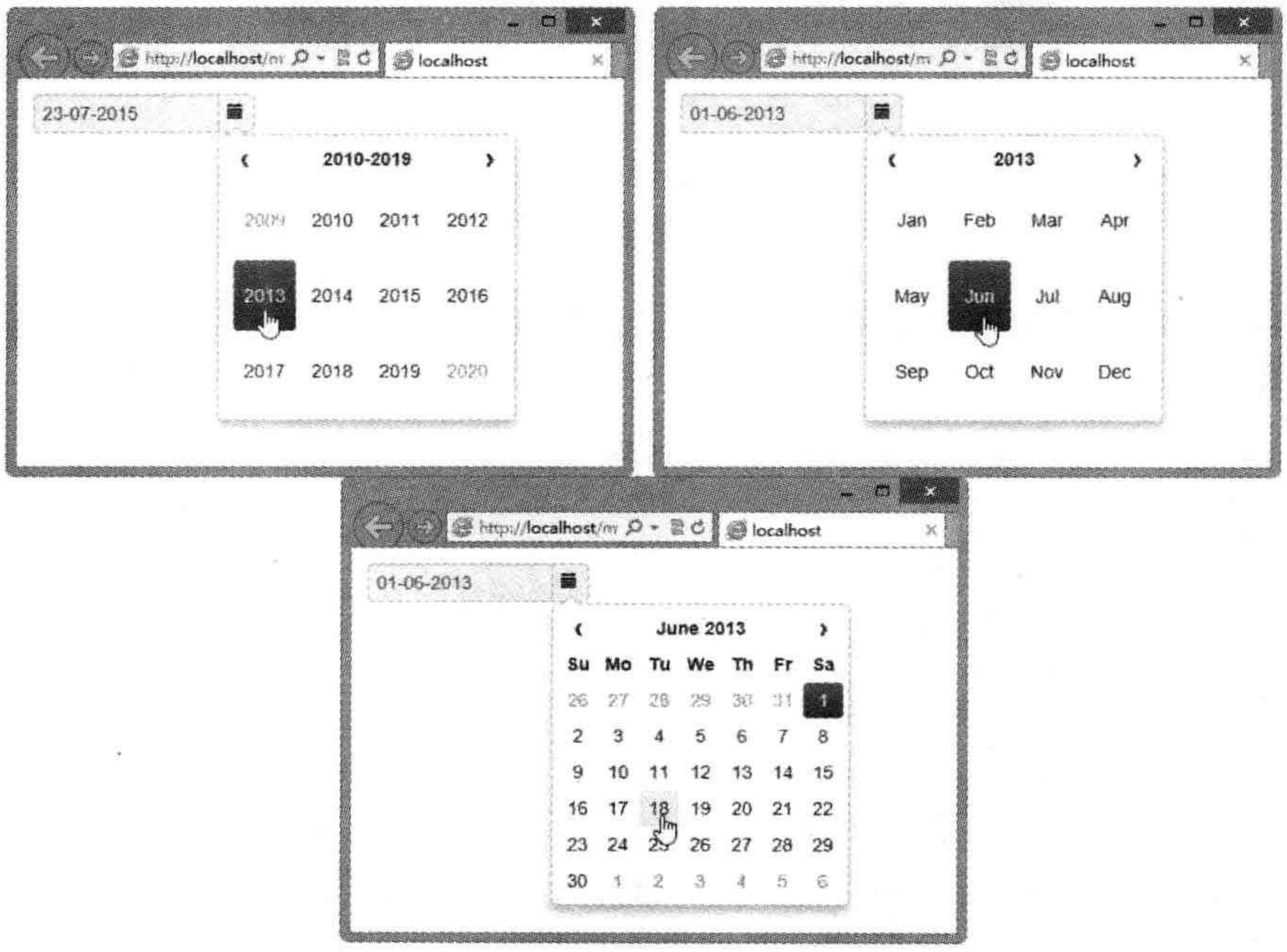


图 8-18　起始年份视图模式

上面的示例也可以直接通过 data 属性进行设置（data-date-viewmode="years"），代码如下：

```
<div class="input-append date" id="date" data-date="01-06-2013" data-date-
 format="dd-mm-yyyy" data-date-viewmode="years">
    <input class="span2" size="16" type="text" value="01-06-2013" readonly>
    <span class="add-on"><i class="icon-calendar"></i></span>
</div>
```

通过 minViewMode 属性可以设置日期选择器视图的最小视图模式，例如，设置日期选择器视图仅显示年和月，则代码如下，预览效果如图 8-19 所示。

```
<script type="text/JavaScript">
$(function(){
     $('#date').datepicker({
          viewMode:"years",
          minViewMode: "months"
     });
});
</script>
```

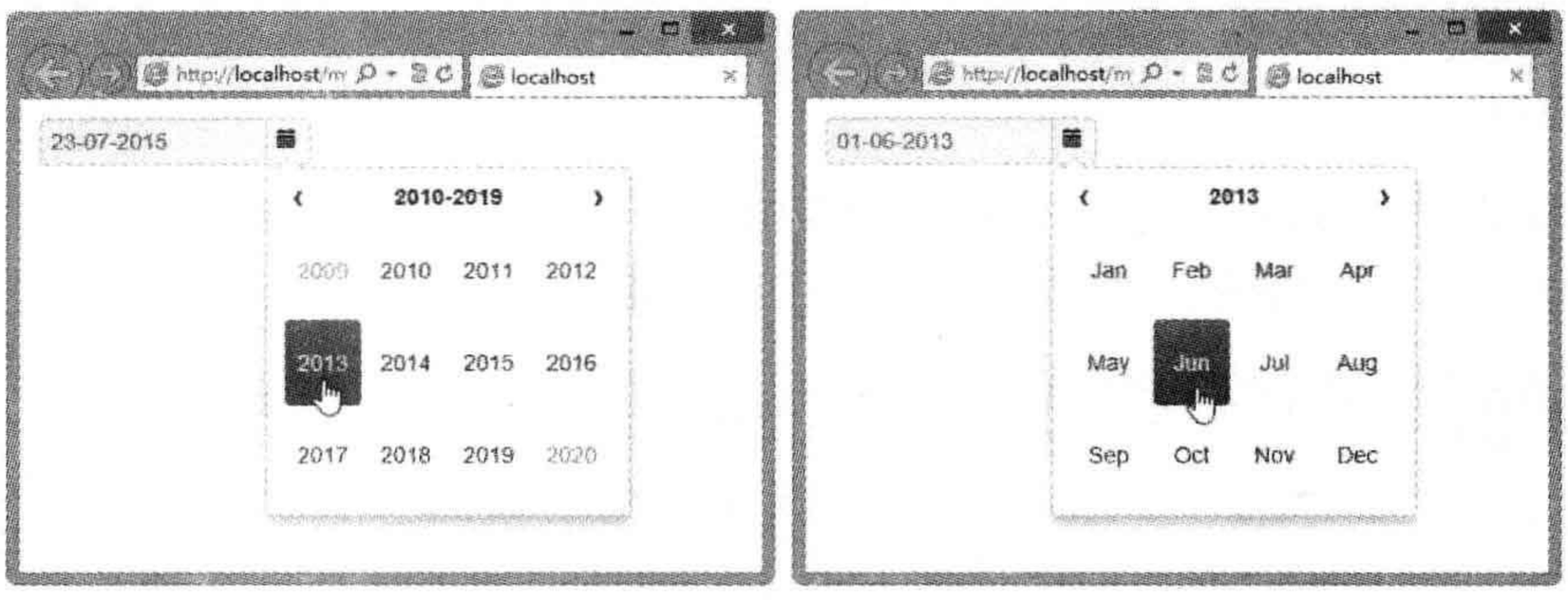


图 8-19　只能够选择年份和月份视图

也可以直接使用 data 属性定义该视图模式。例如，下面的代码定义日期选择器只能够显示月份视图。

```
<div class="input-append date" id="date" data-date="01-06-2013" data-date-
 format="dd-mm-yyyy" data-date-viewmode="months" data-date-minviewmode="months">
    <input class="span2" size="16" type="text" value="01-06-2013" readonly>
    <span class="add-on"><i class="icon-calendar"></i></span>
</div>
```

该插件也支持下面几个特殊方法，简单说明如下。

- .datepicker('show')：显示日期选择器面板。
- .datepicker('hide')：隐藏日期选择器面板。
- .datepicker('place')：更新日期选择器面板相对于元素的位置。
- .datepicker('setValue', value)：为日期选择器设置一个新日期值，该值应该是特定格式的日期字符串或者 Date 对象。

Datepicker 插件还提供几个事件，用来对日期选择执行特殊处理，简单说明如下。

- show：当日期选择器面板显示时触发该事件。
- hide：当日期选择器面板隐藏时触发该事件。
- changeDate：当改变日期值时触发该事件。
- onRender：在日期选择器面板中当一个日期值被选中高亮显示时，将触发该事件。同时该事件处理函数应该返回字符串 'disabled'，用来设置被选择日期及其前面的日期不可选用。

例如，在下面的示例中设计两个按钮，用来设计起始日期值和结束日期值。当设置日期值时，在 changeDate 事件处理函数中判断起始日期值和终点日期值，如果起始日期值大于终点日期值，则显示错误提示信息，如图 8-20 所示。

```
<button class="btn small" id="start" data-date-format="yyyy-mm-dd" data-date="2013-05-20">起始
 日期 </button>
<span id="startDate">2013-05-20</span><br>
<button class="btn small" id="end" data-date-format="yyyy-mm-dd" data-
 date="2013-06-20"> 结束日期 </button>
<span id="endDate">2013-06-20</span><br>
<div class="alert alert-error" id="alert"></div>

<script type="text/JavaScript">
$(function(){
     var startDate = new Date(2013,5,20);
     var endDate = new Date(2013,6,25);
     $('#alert').hide();
     $('#start').datepicker().on('changeDate', function(e){
          if (e.date.valueOf() > endDate.valueOf()){
               $('#alert').show().text(' 起始日期不能够大于终点日期 ');
          } else {
               $('#alert').hide();
               startDate = new Date(e.date);
               $('#startDate').text($('#start').data('date'));
          }
          $('#start').datepicker('hide');
     });
     $('#end').datepicker().on('changeDate', function(e){
          if (e.date.valueOf() < startDate.valueOf()){
               $('#alert').show().text(' 终点日期值不能够小于起始日期值 ');
          } else {
               $('#alert').hide();
               endDate = new Date(e.date);
               $('#endDate').text($('#end').data('date'));
          }
          $('#end').datepicker('hide');
     });
});
</script>
```

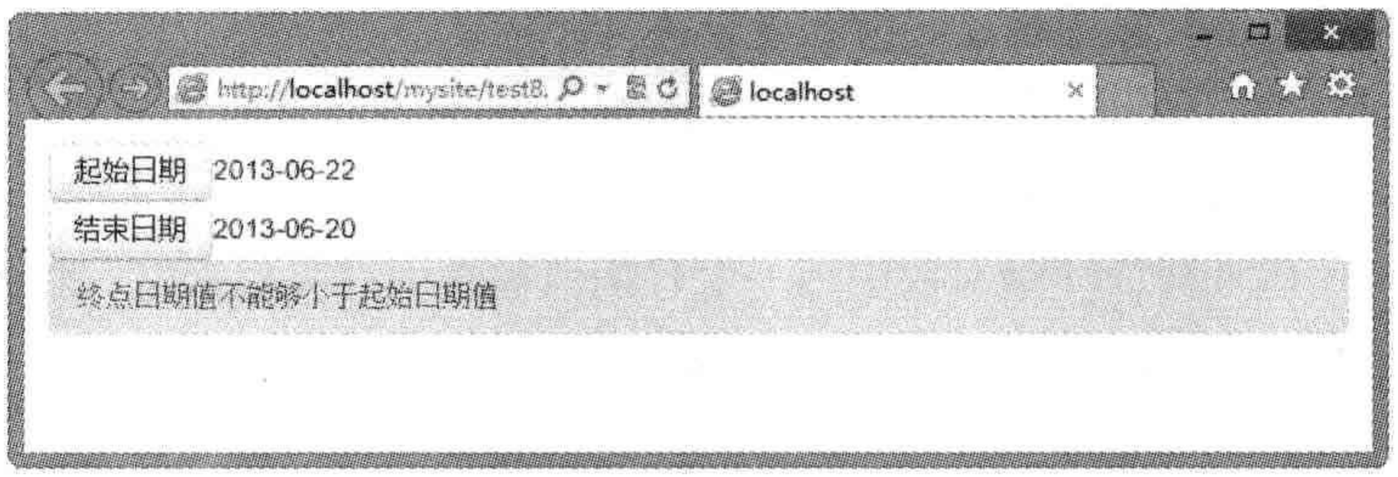


图 8-20 设置有效日期范围

通过 onRender 事件处理函数，也可以设置日期范围，并把范围外的日期显示为不可用状态。例如，在下面的代码中，设置当前日期的值，当选择起始日期之后，则在终点日期面板中把起始日期及其以前的日期全部设置为不可用状态，演示效果如图 8-21 所示。

```
<button class="btn small" id="start" data-date-format="yyyy-mm-dd" data-
 date="2013-06-17">起始日期</button><br>
<button class="btn small" id="end" data-date-format="yyyy-mm-dd" data-
 date="2013-06-27">结束日期</button>

<script type="text/JavaScript">
$(function(){
    var nowTemp = new Date();
    var now = new Date(nowTemp.getFullYear(), nowTemp.getMonth(), nowTemp.
     getDate(),0, 0, 0, 0);
    var checkin = $('#start').datepicker({
        onRender: function(date) {
            return date.valueOf() < now.valueOf() ? 'disabled': '';
        }
    }).on('changeDate', function(e) {
        if (e.date.valueOf() > checkout.date.valueOf()) {
            var newDate = new Date(ev.date)
            newDate.setDate(newDate.getDate() + 1);
            checkout.setValue(newDate);
        }
        checkin.hide();
        $('#end')[0].focus();
    }).data('datepicker');
    var checkout = $('#end').datepicker({
        onRender: function(date) {
            return date.valueOf() <= checkin.date.valueOf() ? 'disabled': '';
        }
    }).on('changeDate', function(e) {
        checkout.hide();
    }).data('datepicker');
});
</script>
```

图 8-21　设置有效日期范围

8.5　jQuery UI Bootstrap

jQuery UI Bootstrap 是一个简洁而优美的 jQuery UI 主题，它受到 Bootstrap 的启发，将其特性引入 jQuery UI 组件中。jQuery UI Bootstrap 提供了漂亮精致的网页排版方式以及导航、表单、按钮等网页设计中常用的元素，并且符合 HTML 和 CSS 标准规范。用户可以在这个主题中使用 Bootstrap 的全部组件，且完美兼容 Twitter Bootstrap。

jQuery UI Bootstrap 提供了在 jQuery UI 中集成 Twitter 的 Bootstrap 框架的功能。使用这个主题，不仅可以使用 Bootstrap 主题窗口微件（widget），而且还兼容 Twitter Bootstrap 的各个方面。即使没有 UI 设计师，也可以制作出很漂亮的 Web 应用程序。

项目主页：http://addyosmani.github.com/jquery-ui-bootstrap/。

中文主页：http://www.bootcss.com/p/jquery-ui-bootstrap/。

下载地址：https://github.com/addyosmani/jquery-ui-bootstrap/。

使用 jQuery UI Bootstrap 的方法如下。

第 1 步：下载 jQuery UI Bootstrap 压缩包，解压到本地站点中，重命名为 jquery-ui-bootstrap。

第 2 步：新建 HTML5 文档，在文档头部区域导入 jQuery 和 jQuery UI 脚本文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="jquery-ui-bootstrap/js/jquery-ui-1.9.2.custom.
 min.js"></script>
```

第 3 步：导入 jQuery UI 样式表文件。

```
<link type="text/css" href="jquery-ui-bootstrap/css/custom-theme/jquery-ui-
 1.9.2.custom.css" rel="stylesheet" />
```

第 4 步：如果需要，还应该导入第三方插件的脚本文件或样式表文件（没有特别说明可以忽略该步）。

第 5 步：构建 HTML 结构。jQuery UI Bootstrap 也采用 Bootstrap 方式，通过样式类来设计对象样式和风格，不过 jQuery UI Bootstrap 样式类的名称与 Bootstrap 不同，主要体现在前缀上。例如，定义不同样式的按钮结构，其中第一个按钮表示主要按钮，第二个按钮表示成功按钮，第三个按钮表示危险按钮，第四个和第五个为普通按钮。

```
<div id="buttons">
    <button class="ui-button-primary">Primary</button>
    <button class="ui-button-success">Success</button>
    <button class="ui-button-error">Danger</button>
    <a class="button">Anchor</a>
    <input type="submit" class="button" value="Submit"/>
</div>
```

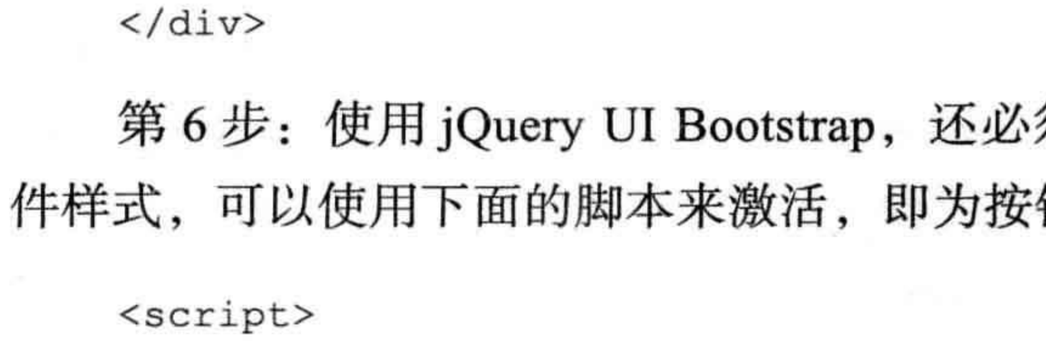
如果把上面的结构转换为 Bootstrap 方式，则可以使用下面的样式类。

```
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
<div id="buttons">
    <button class="btn btn-primary">Primary</button>
    <button class="btn btn-success">Success</button>
    <button class="btn btn-danger">Danger</button>
    <a class="btn">Anchor</a>
    <input type="submit" class="btn" value="Submit"/>
</div>
```

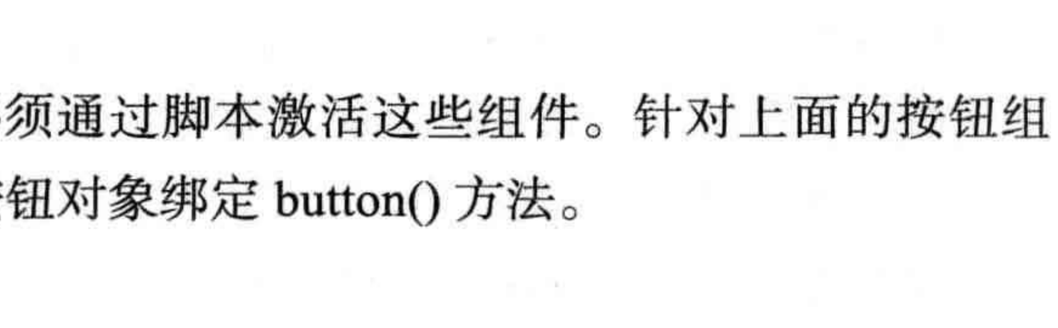
第 6 步：使用 jQuery UI Bootstrap，还必须通过脚本激活这些组件。针对上面的按钮组件样式，可以使用下面的脚本来激活，即为按钮对象绑定 button() 方法。

```
<script>
$(function(){
    $("button").button();
    $(".button").button();
})
</script>
```

第 7 步：在浏览器中预览，显示效果如图 8-22 左图所示，而对应的 Bootstrap 按钮效果如图 8-22 右图所示。



jQuery UI Bootstrap效果



Bootstrap效果

图 8-22　设计 jQuery UI Bootstrap 按钮效果

下面简单介绍 jQuery UI Bootstrap 的几个主要组件。

1. 按钮组

（1）单选按钮组

HTML 结构：

```
<div id="radioset">
    <input type="radio" id="radio1" name="radio" />
    <label for="radio1">Choice 1</label>
    <input type="radio" id="radio2" name="radio" checked="checked" />
    <label for="radio2">Choice 2</label>
    <input type="radio" id="radio3" name="radio" />
    <label for="radio3">Choice 3</label>
</div>
```

激活方法：

```
$('#radioset').buttonset();
```

（2）多选按钮组

HTML 结构：

```
<div id="format">
    <input type="checkbox" id="check1" />
    <label for="check1">B</label>
    <input type="checkbox" id="check2" />
    <label for="check2">I</label>
    <input type="checkbox" id="check3" />
    <label for="check3">U</label>
</div>
```

激活方法：

```
$("#format").buttonset();
```

单选按钮和多选按钮组效果比较如图 8-23 所示。

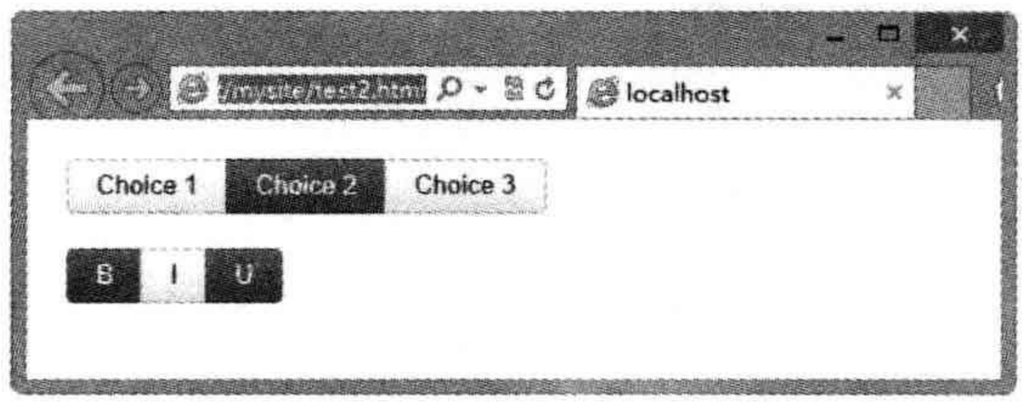


图 8-23　单选按钮组和多选按钮组

2. 工具条

HTML 结构如下，其中 .ui-toolbar 定义工具条框，其中包含 1 个复选框按钮以及 3 个按钮组，如图 8-24 所示。

```
<div class="ui-toolbar">
    <input type="checkbox" id="check" />
    <label for="check">复选框</label>
    <span id="buttonset">
        <input type="radio" id="repeat0" name="repeat" checked="checked" />
        <label for="repeat0">无</label>
        <input type="radio" id="repeat1" name="repeat" />
        <label for="repeat1">有</label>
        <input type="radio" id="repeatall" name="repeat" />
```

```
            <label for="repeatall">全部</label>
        </span> </div>
</body>
```

激活方法：

```
$(function(){
      $("#check").button();
      $("#buttonset").buttonset();
})
```

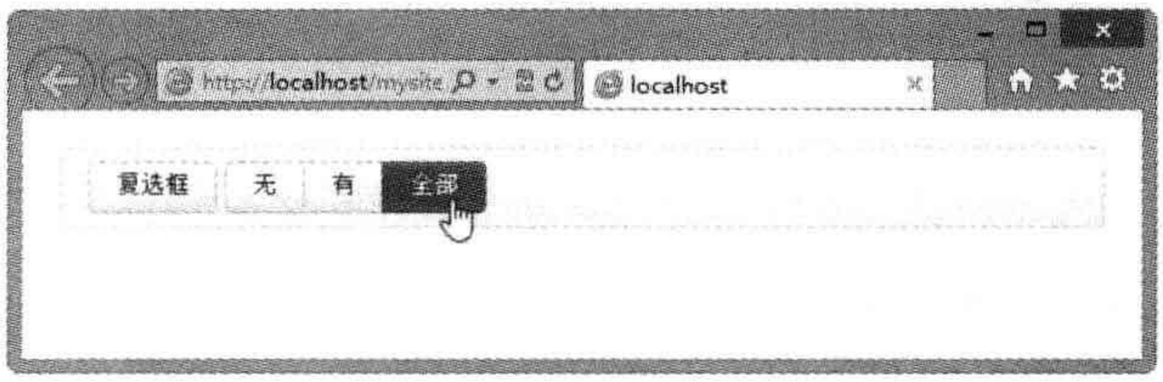


图 8-24　设计工具条效果

3. 折叠

HTML 结构如下，jQuery UI Bootstrap 的折叠结构没有 Bootstrap 的折叠结构复杂，只需要在一个容器中设置 3 个子容器，每个子容器中包含一个标题框和内容框，演示效果如图 8-25 所示。

```
<div id="box">
    <div>
        <h2><a href="#">标题 1</a></h2>
        <div>内容框 1</div>
    </div>
    <div>
        <h2><a href="#">标题 2</a></h2>
        <div>内容框 2</div>
    </div>
    <div>
        <h2><a href="#">标题 3</a></h2>
        <div>内容框 3</div>
    </div>
</div>
```

激活方法如下，jQuery UI Bootstrap 折叠组件需要 JavaScript 脚本配合，并在激活方法中设置折叠的标题块。

```
$(function(){
      $("#box").accordion({
            header: "h2"
      });
})
```

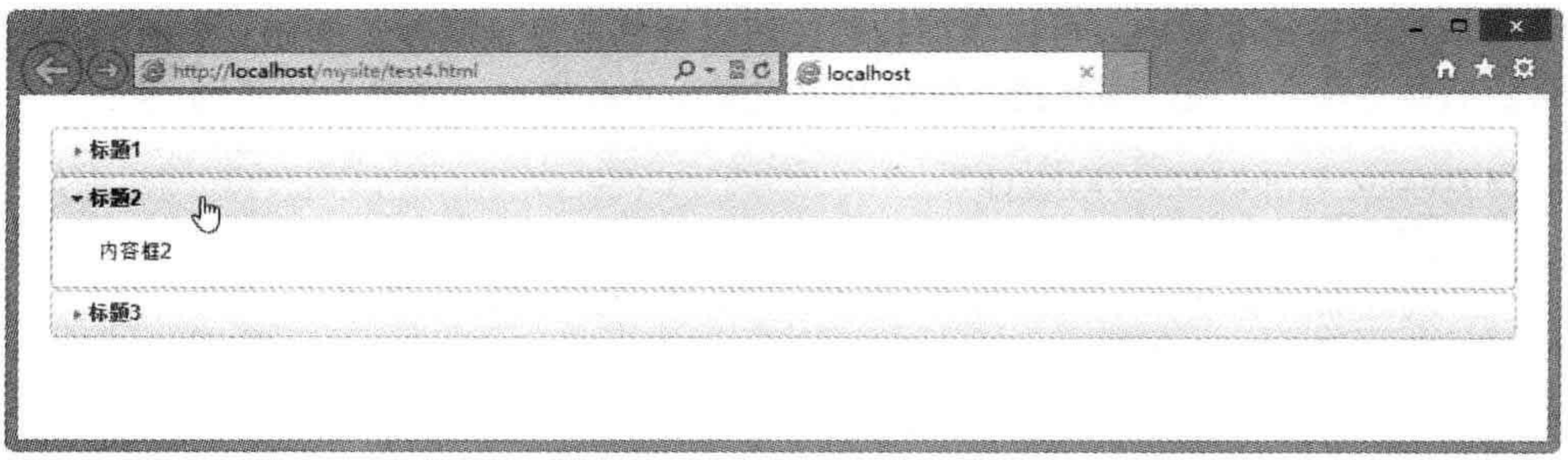


图 8-25　设计折叠组件效果

4. 对话框

设计控制按钮和对话框结构的代码如下，对话框的标题通过 title 属性定义，对话框包含的内容通过包含的 <p> 标签定义。

```
<button id="dialog_link" class="ui-button-primary">打开对话框 </button>
<div id="dialog_simple" title=" 对话框标题 ">
     <p> 对话框内容 </p>
</div>
```

在脚本中初始化对话框，设置 autoOpen:false 即初始化隐藏对话框。然后，当单击按钮时，调用 .dialog（'open'）方法打开对话框，演示效果如图 8-26 所示。

```
<script>
$(function(){
     $('#dialog_simple').dialog({
          autoOpen: false
     });
     $('#dialog_link').button().click(function () {
          $('#dialog_simple').dialog('open');
          return false;
     });
})
</scr ipt>
```

图 8-26　设计简单的对话框效果

在对话框初始化配置中，可以添加控制按钮以及其他显示属性。例如，下面的代码能够设置对话框显示两个按钮，同时初始化对话框宽度为 600 像素，效果如图 8-27 所示。

```
$('#dialog_simple').dialog({
        autoOpen: false,
        width: 600,
        buttons: {
            "Ok": function () {
                $(this).dialog("close");
            },
            "Cancel": function () {
                $(this).dialog("close");
            }
        }
});
```

图 8-27　设计复杂的对话框效果

如果在配置参数中添加 modal:true 声明，则可以打开模态对话框，弹出对话框后，页面其他内容将不再允许操作，效果如图 8-28 所示。

```
$('#dialog_simple').dialog({
        autoOpen: false,
        modal: true
});
```

图 8-28　打开模态对话框

5. Tab 标签页

❑ 设计 Tab 标签页结构，只需要在 Tab 标签列表中通过锚链接把每个标题项目绑定到对应的 Tab 内容框上。

```
<div id="tabs">
    <ul>
        <li><a href="#tabs-a">Tab1</a></li>
        <li><a href="#tabs-b">Tab2</a></li>
        <li><a href="#tabs-c">Tab3</a></li>
    </ul>
    <div id="tabs-a">Tab1 内容框 </div>
    <div id="tabs-b">Tab2 内容框 </div>
    <div id="tabs-c">Tab3 内容框 </div>
</div>
```

激活方法如下，效果如图 8-29 所示。

```
$(function(){
     $('#tabs').tabs();
})
```

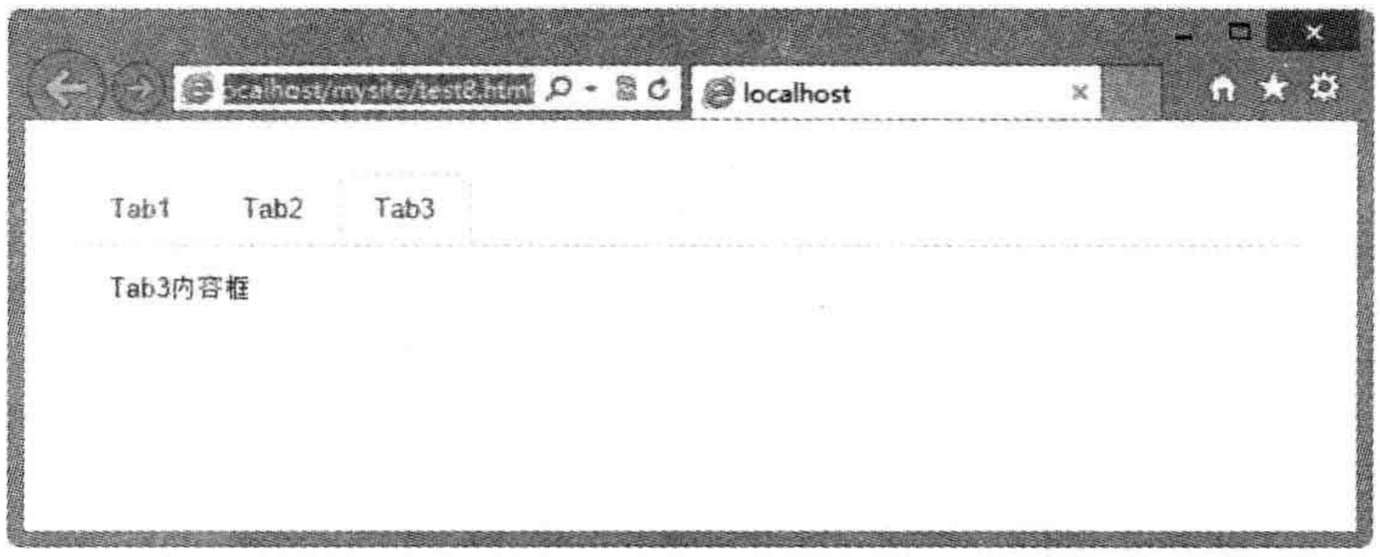


图 8-29　设计标签页

6. 日期选择器

启动日期选择器的代码如下，演示效果如图 8-30 所示。

```
<script>
$(function(){
     $('#datepicker').datepicker({
          inline: true
     });
})
</script>
<div id="datepicker"></div>
```

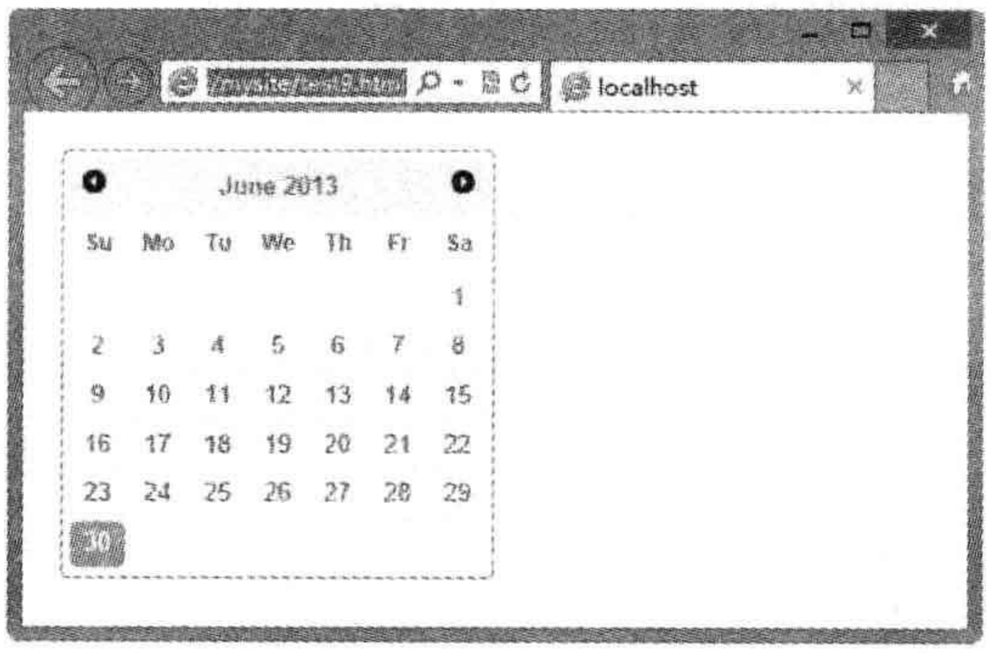


图 8-30　设计日期选择器

第 9 章

使用 Bootstrap 快速开发社区分享网站

本章内容

- 准备工作
- 首页设计
- 阅读页设计
- 小组页设计
- 打卡页设计
- 词根页设计

在前面几章中，我们介绍了什么是 Bootstrap，并系统讲解了 Bootstrap 框架、样式、组件、插件等知识和应用，本章将学习如何用 Bootstrap 制作响应式社区网站。Bootstrap 是一个响应式框架，用它创建一个响应式的 Web 应用程序将是一个不错的起点。

互联网中各种 Web 应用程序千奇百怪，如雨后春笋般不断闯入人们的视野。把一种朦胧的想法快速设计成页面，呈现给广大用户，是广大草根创业者必须面对的实际问题。事实上，我们经常会冒出各种新奇的想法和创业冲动，但是苦于技术门槛和网站开发的成本，很多时候这些想法和冲动会自生自灭，半途而废。现在，借助 Bootstrap 这个成熟的响应式框架，用户可以快速把自己的想法变成网页。

本章将要构建的主题是基于一个基本单词分享的学习型网站，对于本示例，我们将为其页面及功能创建出一系统模板：

- 优雅、成熟的首页
- 阅读页面
- 小组圈
- 打卡创意
- 词根应用

9.1　准备工作

使用 Bootstrap 开发网站，读者应该适当做些准备工作，特别是 Bootstrap 定制工作不能够忽视，当然网站本身的策划、网站配色、网站技术和资料的准备等都是不可或缺的环节，下面就几个主要问题进行说明。

9.1.1　定制 Bootstrap

定制 Bootstrap 是一件很重要的工作，我们在第 2 章中作过说明，下面结合本章示例再进一步详细说明。

第 1 步：定制网站配色系统。每个网站都有自己的色彩风格，这在设计之初就应该基本确定。例如，对于本示例来说，主色调为灰色和灰绿色。因此，这里需要定制几个基本颜色：

- 网页背景色：#E4E4E4（浅灰色）
- 网页前景色（字体颜色）：#333333（深灰色）
- 超链接颜色：#56A590（灰绿色）
- 鼠标经过颜色：#209E85

其他颜色不是很重要，可以忽略，但是上面四个基本颜色必须定制。如果忘记定制，而是直接引用 Bootstrap 框架的默认样式，则应该在本地样式表中重置这些基本样式：

```
body {
    color: #333333;
    background-color: #E4E4E4;
}
a { color: #56A590; }
a:hover, a:focus { color: #209E85; }
```

定制工作可以在 http://twitter.github.io/bootstrap/customize.html 页面完成，如图 9-1 所示。

第 2 步：定制栅格系统。Bootstrap 默认栅格系统：网页宽度 940 像素，12 格，每格宽度 60 像素，栅格间隔 20 像素。例如，设计网站宽度为 950 像素，栅格数为 24，每个栅格宽 30 像素，栅格间隔为 10 像素，如图 9-2 所示。

图 9-1　定制网站色彩

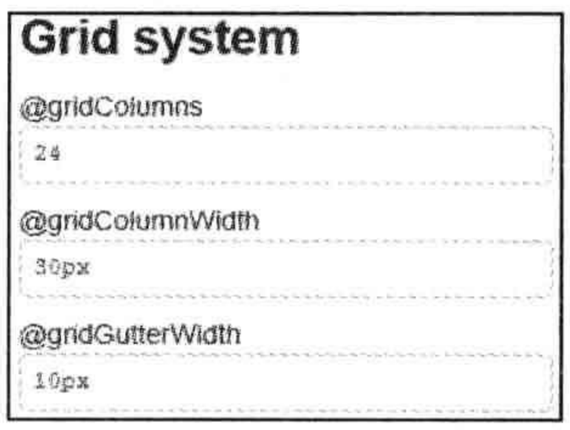


图 9-2　定制栅格系统

第 3 步：除了页面基本色彩问题、栅格系统，读者也可以根据需要定制各种组件的基本样式或者标签基本样式。不过对于这些非基本样式，建议在本地样式表中重置覆盖。

9.1.2　初始化 Bootstrap

建议应用 Bootstrap 框架的页面为 HTML5 文档类型，同时在页面头部区域导入框架基本样式表文件和脚本文件。

```
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<link href="bootstrap/css/bootstrap.css" rel="stylesheet" type="text/css" />
```

必须导入 jQuery 框架文件、Bootstrap 样式表文件，如果在页面中使用多个插件，则可以导入 bootstrap.js，如果仅需要特定插件，可以仅导入该特定的插件文件，具体文件可以查看未压缩的 Bootstrap 压缩包。

如果需要设计响应式页面，则还应该导入 bootstrap-responsive.css 文件，并把它置于 bootstrap.css 文件之后。具体代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title> </title>
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js"></script>
<script type="text/JavaScript" src="bootstrap/js/bootstrap.js"></script>
<link href="bootstrap/css/bootstrap.css" rel="stylesheet" type="text/css" />
<link href="bootstrap/css/bootstrap-responsive.css" rel="stylesheet"
 type="text/css" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
</head>
<body>
</body>
</html>
```

应用 Bootstrap 插件时，推荐使用 data 属性激活，不建议使用 JavaScript 脚本激活，当然部分必须用 JavaScript 脚本激活的插件除外。各种插件的激活方法请参阅第 7 章的详细说明。

如果需要配置插件参数，则可以使用 data 方法设置，也可以使用 JavaScript 构造函数进行配置，具体方法可以根据需要进行选择。

完成页面初始化设置后，下面的工作就是根据页面设计草图，分别引用 Bootstrap 栅格系统完成页面版式设计，然后使用 Bootstrap 组件、插件设计各个具体模块的样式和交互效果。

最后，读者可以使用本地样式表文件对 Bootstrap 样式进行重置和修补，也可以在页面中引入其他技术框架，实现技术混合开发。

9.2　首页设计

在全面考虑好网站的栏目、结构、风格和配色等基本问题之后，就可以动手制作首页了。在网站开发中，首页设计和制作将占整个制作时间的 40%。首页设计是一个网站成功与否的关键，往往看到第一页就会对整个网站有一个整体的感觉，是不是能够让浏览者继续单击进入，是否能够吸引浏览者留在站点上，全凭首页设计的功力。

9.2.1　设计思路

首页类似书的封面，但又不同于封面设计。封面式网页没有具体内容，只放一个 logo 供单击进入，或者只有简单的图形菜单。除非设计艺术类站点，或者确信内容独特可以吸引浏览者单击进入站点，否则封面式首页并不会给站点带来什么好处。从根本上说，首页就是全站内容的目录，是一个索引。但只是罗列目录显然是不够的，那么如何才能设计好首页呢？以下是一般的步骤。

（1）确定首页的功能模块

首页的内容模块是指在首页上实现的主要内容和功能。一般的站点都需要这样一些模

块：网站名称（logo）、广告条（banner）、主菜单（menu）、新闻（news）、搜索（search）、链接（links）、版权（copyright）等。选择哪些模块，实现哪些功能，是否需要添加其他模块，都是首页设计首先需要确定的。

本示例首页主要包括如下几个功能模块：网站名称和标识、主菜单、灯箱广告、宣传性新闻、链接（links）和版权（copyright）。

（2）设计首页的版面

在功能模块确定后，开始设计首页的版面。就像搭积木一样，每个模块是一个单位积木，如何拼搭出一座漂亮的房子，就需要创意和想象力了。

设计版面的最好方法是：找一张白纸，一支笔，先将理想中的草图勾勒出来，然后再用网页制作软件实现。例如，在设计本案例之前，初步勾勒一幅草图，如图 9-3 所示。整个网站包含四部分：头部标题区、主体灯箱广告位、新闻展示位以及网站相关链接和版权信息区域。根据草图设计整个页面框架模块，如图 9-4 所示。

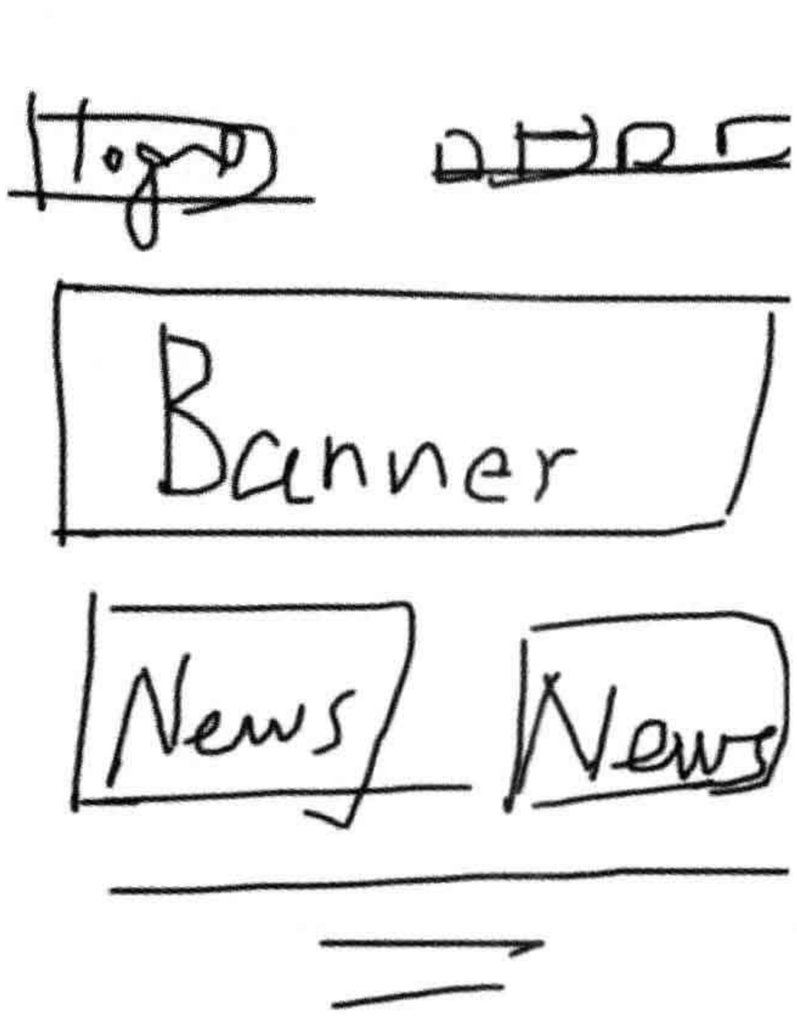


图 9-3　网站首页设计草图

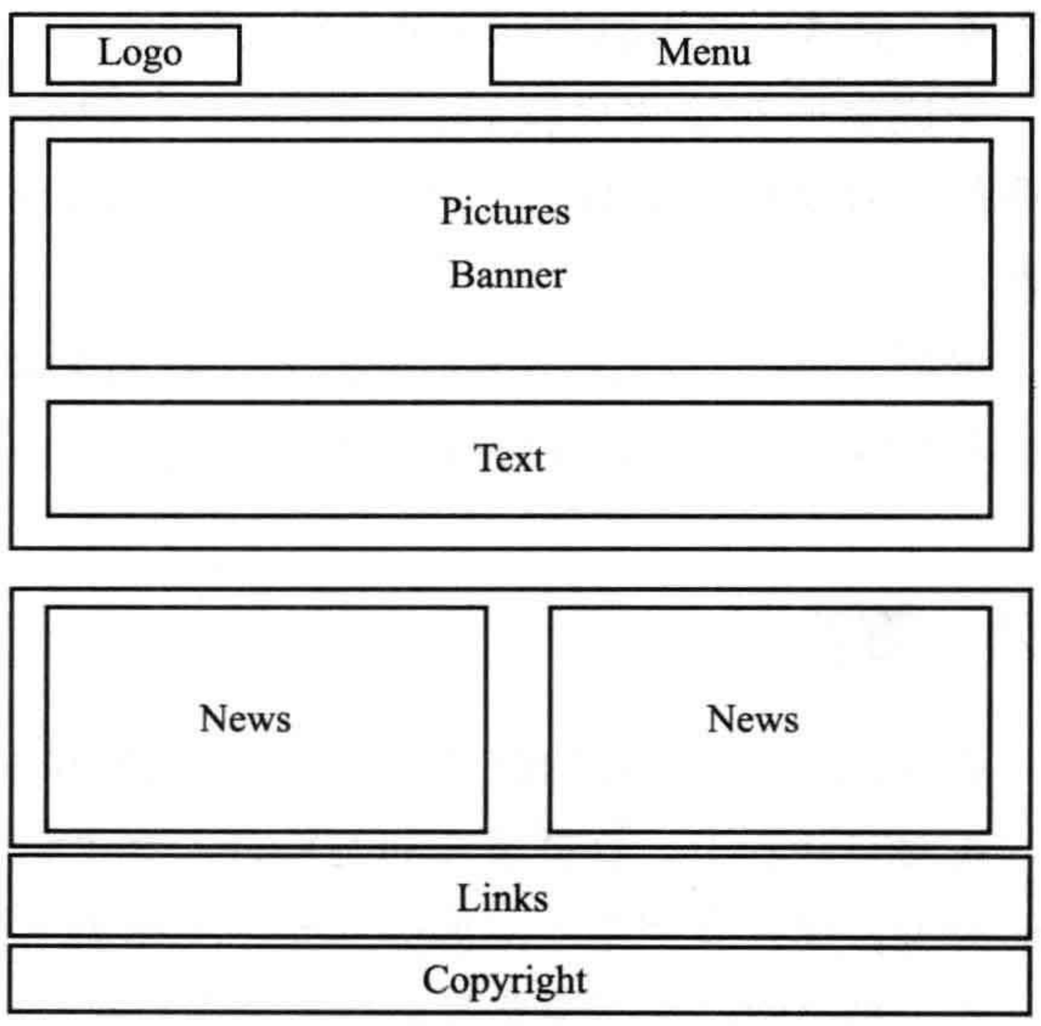


图 9-4　网站首页设计模块图

如有需要，建议读者使用 Photoshop 绘制效果图，然后通过切片输出。由于本站点使用 Bootstrap 框架进行设计，布局采用 Bootstrap 的 940 栅格系统，样式借用 Bootstrap 模板组件，只需要确定整个页面的结构和配色即可，所以整个页面不需要使用 Photoshop 输出效果图，整个首页设计效果如图 9-5 所示。

（3）处理技术上的细节

完成页面的规划、设计工作后，后期的制作过程就是一些具体的技术活了。例如，制作的首页如何能在不同分辨率下保持不变形，如何能在不同浏览器下看起来都不太丑陋，如何设置字体和链接颜色等。

图 9-5　首页设计效果图

9.2.2　设计结构

首页设计是整个网站设计的难点和关键，借助 Bootstrap 栅格系统，会更加轻松地完成整个站点的设计制作。

首先，启动 index.html 首页文档，在 <body> 标签内设计一个 Bootstrap 栅格基本结构， 包含框设置页面宽度为 940 像素，设置页面居中显示； 设计每行版块，其中包含两个同等宽度的列，每列宽度为 470 像素。

```
<div class="container">
```

```
        <div class="row">
            <div class="span6"></div>
            <div class="span6"></div>
        </div>
    </div>
```

然后，在这个框架基础上添加添加 3 行版块：标题栏、整版灯箱广告和版权信息区。考虑到这 3 行内容块不需要多列布局，因此不再需要 row 控制。

```
<div class="container">
    <div id="page-head" class="clearfix">
        <div id="logo"></div>
        <div id="page-head-right"></div>
    </div>
    <div id="carousel_box" class="carousel slide">
        <ol class="carousel-indicators"> </ol>
        <div class="carousel-inner"></div>
    </div>
    <div class="row">
        <div class="span6"></div>
        <div class="span6"></div>
    </div>
    <footer>
        <div class="row"></div>
        <div class="copyright"></div>
    </footer>
</div>
```

最后，根据页面三级结构细化文本内容，如列表信息、分类链接、标题和段落文本等，对于细节结构，主要用到 、<li>、<h2>、<p> 标签。具体细节结构请参阅本书源代码。

9.2.3 设计主菜单和按钮

标题栏包含 3 个部件：logo、主菜单和用户交互按钮，如图 9-4 所示。为了避免标题栏与下面一行栏目发生重叠，在标题栏包含框中引入 clearfix 类，清除下一行浮动错位问题。使用 pull-left 类把网站 logo 推到左侧显示，使用 pull-right 类把登录和注册按钮推到页面右侧显示。

设置页面主菜单栏右侧浮动显示，并通过相对定位方式调整上下左右的偏移位置，效果如图 9-6 所示。

```
<div id="page-head" class="clearfix">
    <div class="pull-left"> <a href="#"><img src="images/landing_page_logo.
     png" /></a> </div>
    <div id="page-head-right">
       <div id="main-login" class="pull-right"> <a id="" class="twi-btn-left"
        name="login" href="#"><span> 登录 </span></a><a id="" class="twi-btn-right"
        name="register" href="#"><span> 注册 </span></a> </div>
       <div id="main-menu"> <a href="#"> 阅读 </a> <a href="#"> 词根 </a> <a href="#">
```

```
        词汇派生 </a> <a href="#"> 周刊 </a> <a href="#"> 小组 </a> <a href="#"> 打卡日记
        </a> <a href="#"> 手机客户端 </a> <a href="#"> 关注我们 </a> </div>
    </div>
</div>
```

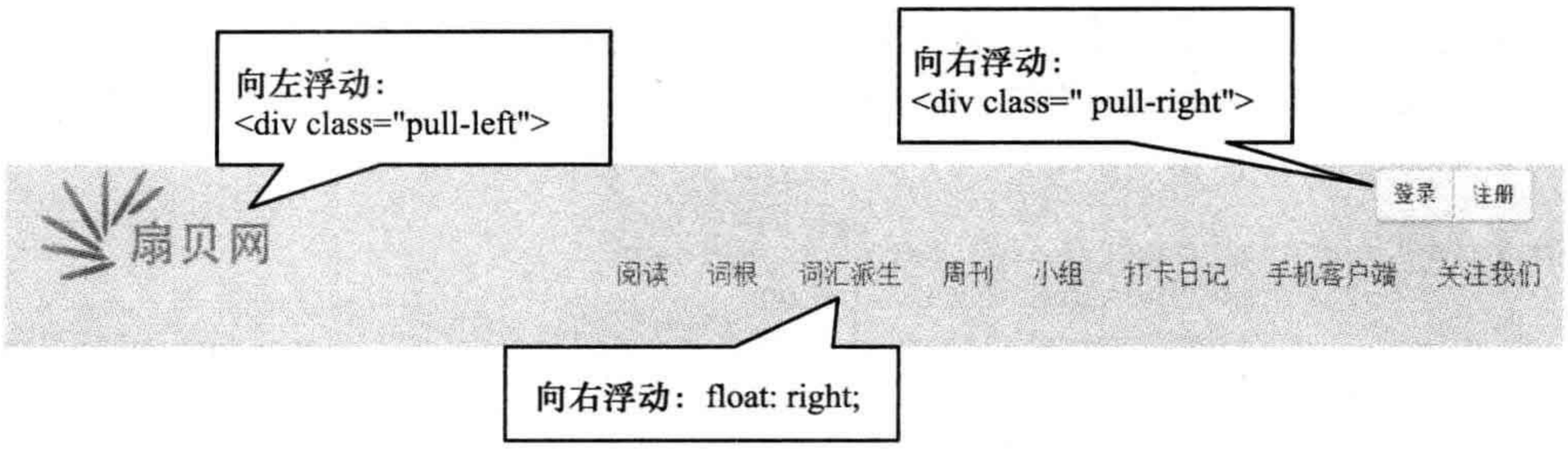


图 9-6　标题栏构成效果

下面我们重点看一下按钮样式。Bootstrap 提供了按钮基本样式（btn）以及系列风格（如 btn-success）。在本示例中考虑到页面配色需要，我们重新自定义了 btn 样式，并命名为 twi-btn-left 和 twi-btn-right。这里运用了背景图片切换模式来模拟 Bootstrap 风格按钮样式，确保与页面色彩保持一致。代码如下：

```
/* 登录按钮样式 */
.twi-btn-left { background: url(../images/icons.png) no-repeat 0 -64px; width:
 54px; height: 40px; display: inline-block; text-align: center; }
.twi-btn-left: hover { background: url(../images/icons.png) no-repeat 0px -144px; }
.twi-btn-left: hover span { color: white; }
.twi-btn-left span { line-height: 40px; text-indent: 6px; color: #555555;
 display: inline-block; width: 32px; height: 32px; cursor: pointer; }
/* 注册按钮样式 */
.twi-btn-right { background: url(../images/icons.png) no-repeat -54px -64px;
 width: 54px; height: 40px; display: inline-block; text-align: center; }
.twi-btn-right: hover, .twi-btn-right.active { background: url(../images/icons.
 png) no-repeat -54px -104px; }
.twi-btn-right: hover span, .twi-btn-right.active span { color: white; }
.twi-btn-right span { text-indent: -6px; color: #555555; display: inline-block;
 line-height: 40px; width: 32px; height: 32px; cursor: pointer; }
```

设计思路：事先做好按钮图，包含默认效果和鼠标经过效果两个小图，可以把它们合成在一张大图中，如图 9-7 所示。这种技巧就是 CSSSprites，它借助 CSS 背景定位技术实现背景图的切换显示。交互行为的触发机制借助鼠标伪类样式来实现，即正常状态样式和 :hover 伪类状态样式。

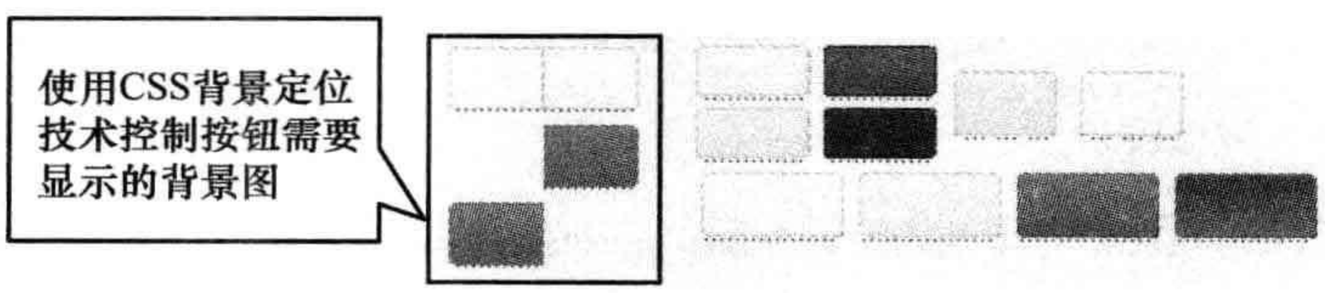


图 9-7　设计的按钮背景合成图（部分）

9.2.4 设计轮播广告位

Bootstrap 轮播插件结构比较固定，完整的结构代码如下。轮播包含框需要指明 ID 值和 carousel、slide 类。框内包含三部分组件：标签框（carousel-indicators）、图文内容框（carousel-inner）和左右导航按钮（carousel-control）。通过 data-target="#carousel_box" 属性启动轮播，使用 data-slide-to="0"、data-slide="prev"、data-slide="next" 定义交互按钮的行为。

```
<div id="carousel_box" class="carousel slide">
        <ol class="carousel-indicators">
            <li data-target="#carousel_box" data-slide-to="0" class="active"></li>
        </ol>
        <div class="carousel-inner">
            <div class="item active"></div>
        </div>
        <a class="left carousel-control" href="#carousel_box" data-slide="prev">&lsaquo;</a>
        <a class="right carousel-control" href="#carousel_box" data-slide="next">&rsaquo;</a>
 </div>
```

在轮播基本结构基础上，我们来设计本示例首页的轮播广告位结构。考虑到设计需要，在图文内容框（carousel-inner）中包裹了多层内嵌结构。标签框通过有序列表结构定义（<ol class="carousel-indicators">）。图文内容框（carousel-inner）中每个图文项目使用 定义，在该项目框中使用 定义轮播图的标签文字框，并借助 Bootstrap 栅格系统设计两列布局版式。

左右导航按钮使用 carousel-control 来设计，并通过 left 和 right 定义左右导航箭头样式，通过 href="#carousel_box" 绑定轮播空间的目标框，使用 data-slide="prev" 和 data-slide="next" 激活轮播行为。整个轮播广告的结构如下：

```
<a class="left carousel-control" href="#carousel_box" data-slide="prev">
<div id="carousel_box" class="carousel slide">
        <ol class="carousel-indicators">
            <li data-target="#carousel_box" data-slide-to="0" class="active"></li>
            <li data-target="#carousel_box" data-slide-to="1"></li>
            <li data-target="#carousel_box" data-slide-to="2"></li>
            <li data-target="#carousel_box" data-slide-to="3"></li>
        </ol>
        <div class="carousel-inner">
      <div class="item active"> <a href="#"><img src="images/reader.jpg"
       alt="扇贝阅读"/></a>
          <div class="carousel-caption">
              <div class="slide-col2">
                  <h2>扇贝阅读</h2>
                  <div class="row"></div>
              </div>
          </div>
</div>
<div class="item"> <a href="#"><img src="images/apps.jpg" alt="Slide-3"></a>
    <div class="carousel-caption">
        <div class="slide-col2">
```

```
                    <h2>随时随地都可学习</h2>
                    <div class="row"></div>
                </div>
            </div>
        </div>
        <div class="item"> <a href="#"><img src="images/market.jpg" alt="Slide-4" /></a>
            <div class="carousel-caption">
                <div class="slide-col2">
                    <h2>丰富的学习资料</h2>
                    <div class="row"></div>
                </div>
            </div>
        </div>
        <div class="item"> <a href="#"><img src="images/progress.jpg" /></a>
            <div class="carousel-caption">
                <div class="slide-col2">
                    <h2>进步看得见</h2>
                    <div class="row"></div>
                </div>
            </div>
        </div>
    </div>
    <a class="left carousel-control" href="#carousel_box" data-slide="prev">
&lsaquo;</a> <a class="right carousel-control" href="#carousel_box" data-
slide="next">&rsaquo;</a>
</div>
```

在默认状态下，本页面的轮播效果如图 9-8 所示。

图 9-8　轮播组件默认样式

考虑到这种样式与本示例的设计风格发生冲突，下面来分析如何自定义轮播插件的样式，让导航标签和说明文字显示在图片的底部，并修改其显示效果以与页面整体效果保持协调。在本页 CSS 本地样式表中定义如下样式，设计如图 9-9 所示的效果。

图 9-9　自定义轮播组件样式

首先，显式定义轮播包含框的高度，以便留出多余的空间显示文本和标签。

```
#carousel_box {
    height: 770px;
}
```

然后，重新定义 <ol class="carousel-indicators"> 和 的定位位置，同时修改文本包含框的背景色，使其与页面背景色融合。轮播插件默认状态下设置 <ol class="carousel-indicators"> 和 包含框为绝对定位，所以这里仅需要修改 left 和 top 属性值即可。

```
.carousel-indicators {
    top: 530px;
    left: 500px;
}
.carousel-caption {
    top: 580px;
    background: #E4E4E4;
}
```

最后，设计细节样式：取消轮播包含框的 overflow 限制，让超出的区域能够可见显示；重新定义文本标签样式，如行高和字体颜色；重新设计导航标签样式，设置当前指示标签背

景色为绿色，与页面主色调保持一致。

```
.carousel-inner { overflow: visible; }
.carousel-caption h4, .carousel-caption p {
     line-height: 20px;
     color: #333;
}
.carousel-indicators li { background-color: #D4D5D6; }
.carousel-indicators .active { background-color: #209E85; }
```

9.2.5　设计新闻区和版权区版式

新闻区使用嵌套的栅格系统设计。外层栅格分为两列，各占一半，总宽度为 940 像素。

```
<div class="row">
     <div class="span6"></div>
     <div class="span6"></div>
</div>
```

每列内又包含多行，第一行为标题行，第二行为栅格系统行，其内部包含两列，代码如下。以此方式可以在每列中设计多行版式样式，效果如图 9-10 所示。

```
<div class="page-header">
     <h3 class="title"> <span>老师的话 </span> </h3>
</div>
<div class="row" id="teacher-testmonials">
     <div class="span potrait"></div>
     <div id="teacher-testmonial-area" class="span4"> </div>
</div>
```

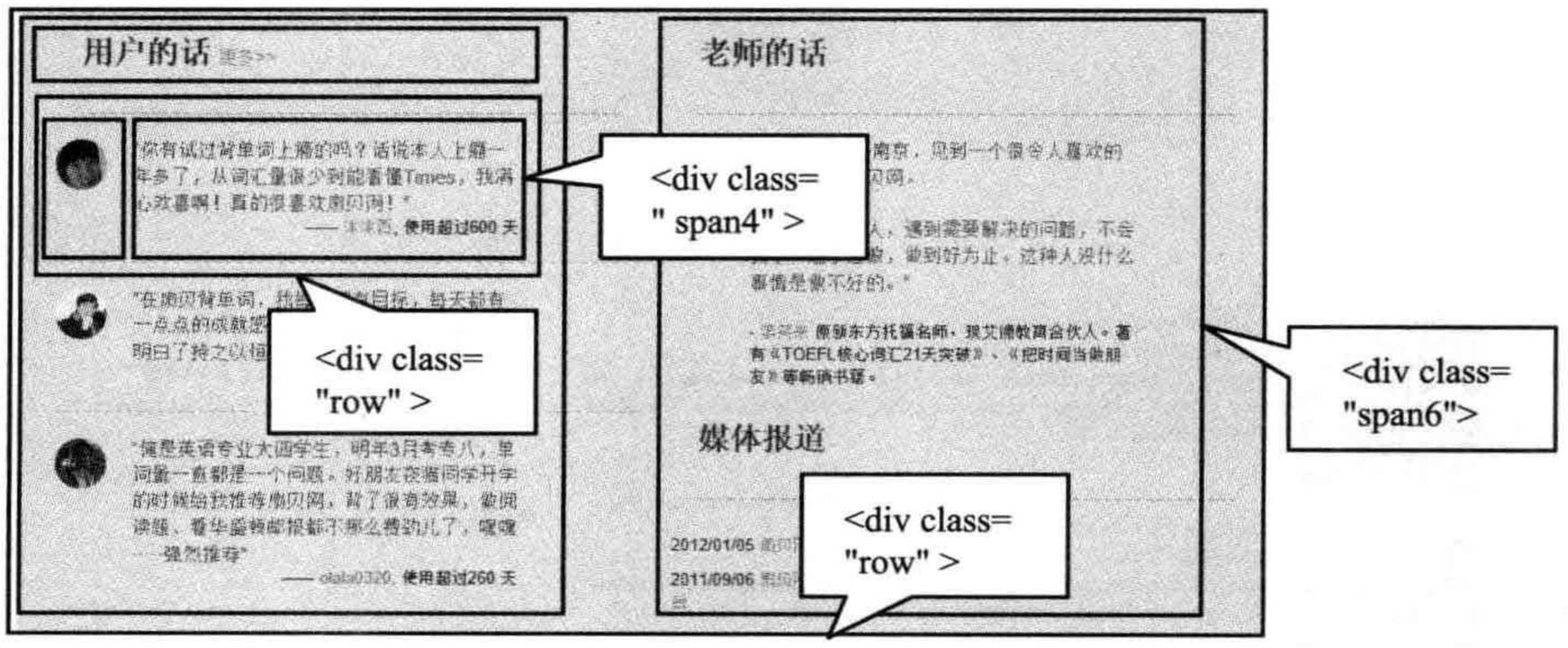


图 9-10　新闻区栅格嵌套版式

版权区版式比较简单，是一个 4 列版式的栅格布局，每列 220 像素，如图 9-11 所示。

```
<footer>
    <div class="row">
        <div class="span3"></div>
```

```
            <div class="span3"></div>
            <div class="span3"> </div>
            <div class="span3"></div>
        </div>
        <div class="copyright"> </div>
    </footer>
```

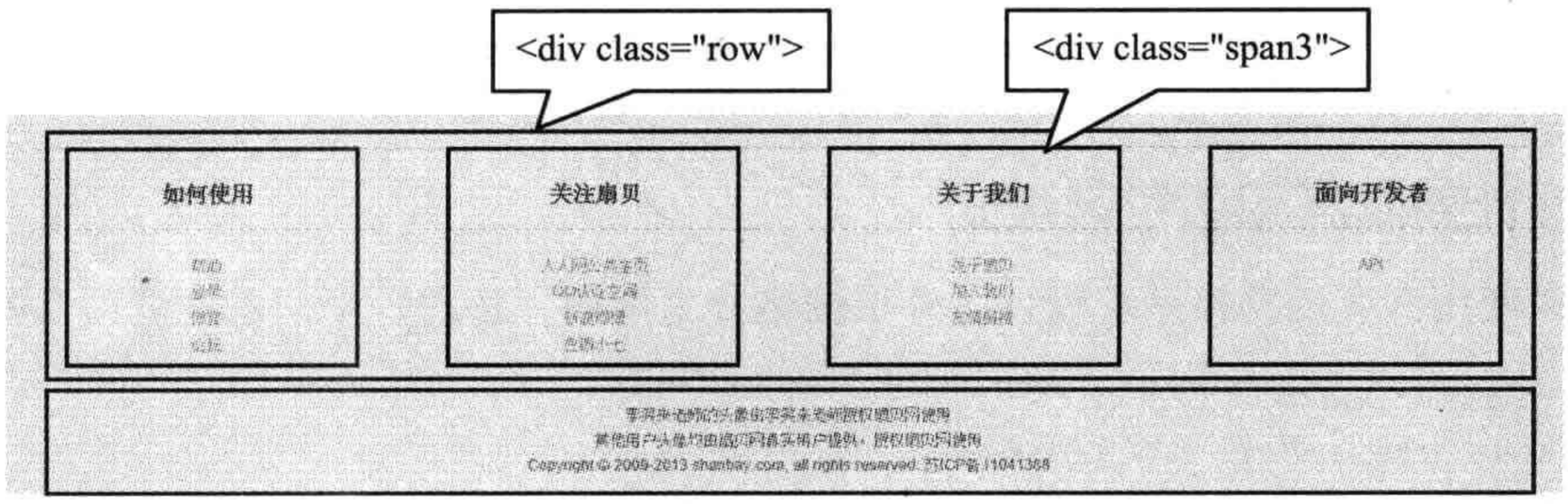


图 9-11　版权区栅格版式

9.3　阅读页设计

阅读页实际上就是一个新闻页面，该页面主要呈现各种英文新闻，列表显示世界主要英文报刊源。页面整体结构比较简单，使用 Bootstrap 栅格系统设计两列版式，主列显示新闻条目，侧列显示辅助性内容和报刊源列表，页面效果如图 9-12 所示。

图 9-12　阅读页面设计效果

9.3.1　设计响应式主菜单

新建 HTML 5 文档，保存为 news.html，在页面头部区域完成 Bootstrap 框架的导入工作，包括 jquery.js、bootstrap.js、bootstrap.css、bootstrap-responsive.css 4 个基础文件。具体设计过程如下。

第 1 步：设计导航条框架结构。导航条框架由两层嵌套包含框构成。使用 container 固定导航条宽度并居中显示。自定义 topbox 类调节导航条上下间距。

```
<div class="navbar container topbox">
    <div class="navbar-inner">
    </div>
</div>
```

第 2 步：设计响应式收缩展开按钮。在导航条内嵌入如下结构，btn 和 btn-navbar 组合定义导航条按钮组件，data-toggle="collapse" 触发交互式行为，单击该按钮能够展开收缩的导航条，data-target=".nav-collapse" 指定需要收缩的导航条内容框。如图 9-13 所示，当浏览器窗口变窄时，主菜单将自动收缩，单击该展开收缩按钮，可以显示所有的导航项目。

```
<a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
     <span class="icon-bar"></span>
     <span class="icon-bar"></span>
     <span class="icon-bar"></span>
</a>
```

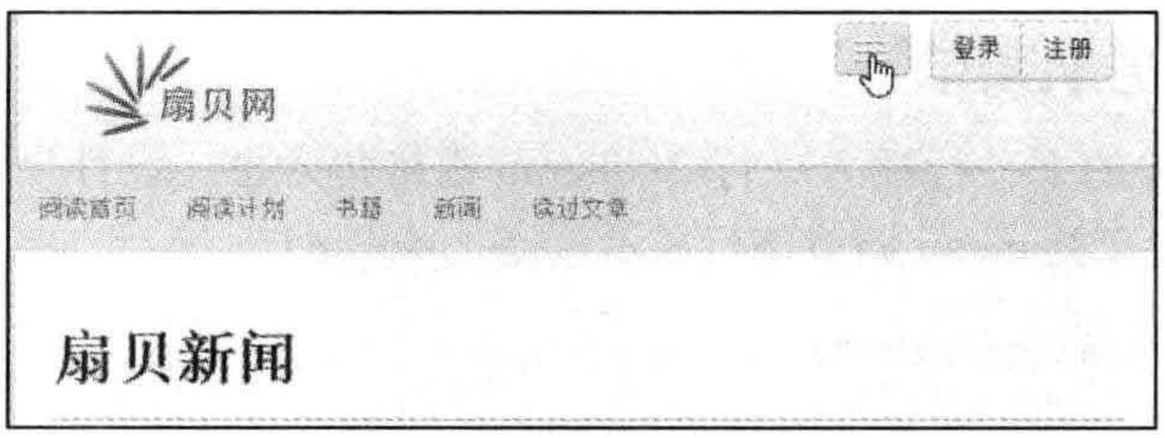


图 9-13　收缩后的导航条效果

第 3 步：设计导航条图标。使用 brand 类设计导航条标志，在该超链接中嵌入网站 logo。

```
<a class="brand" href="#"><img src="images/logo2.png" /></a>
```

第 4 步：在导航条中嵌入登录和注册按钮组。该组件设计请参阅 9.2.3 节，具体结构和样式不再赘述。

第 5 步：设计下拉菜单组件。使用 nav-collapse 设计下拉菜单外框，使用 nav 设计下拉菜单内框，使用 dropdown 设计下拉菜单项，使用 data-toggle="dropdown" 激活下拉菜单行为。在下拉菜单外框中使用 navbar-search 设计导航条搜索框。整个下拉菜单组件效果如图 9-14 所示。

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

```
<li class="dropdown"> <a href="#"
      class="dropdown-toggle"
      data-toggle="dropdown"> 背单词 <b class="caret"></b> </a>
    <ul class="dropdown-menu">
        <li><a href="#"> 单词书 </a></li>
        <li><a href="#"> 单词量测试 </a></li>
    </ul>
</li>
        <li class="dropdown">……</li>
        <li> <a class="main-menu" href="#"> 阅读 </a> </li>
        <li> <a class="main-menu" href="#"> 市场 </a> </li>
        <li class="dropdown">……</li>
    </ul>
    <form class="navbar-search pull-left" action="">
        <input type="text" class="search-query span2" placeholder=" 查词 ">
        <span class="search-icon"></span>
    </form>
</div>
```

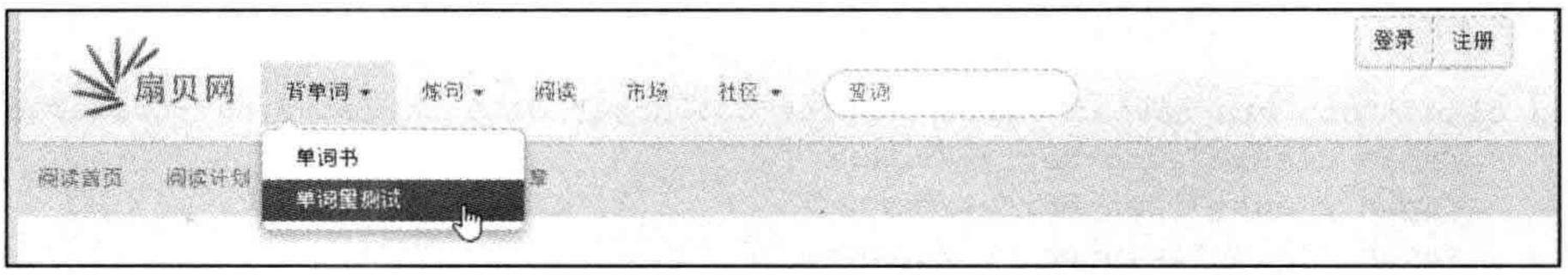


图 9-14　设计的下拉菜单效果

9.3.2　设计附加导航菜单

首先，在主菜单的下一行插入一行 Bootstrap 栅格包含框，设计导航菜单固定宽度，并居中显示。借助自定义类 menu 调节该行显示位置（margin）。

```
<div class="container menu">
</div>
```

然后，在 中插入一个列表结构，引入 nav 和 nav-pills 样式类，设置 pill 胶囊式菜单样式。为菜单框定义 id 属性（id="menu"），以便应用附加导航插件（借助 id 来控制该菜单对象）。

```
<ul class="nav nav-pills"  id="menu">
    <li> <a href="#"> 阅读首页 </a> </li>
    <li> <a href="#"> 阅读计划 </a> </li>
    <li> <a href="#"> 书籍 </a> </li>
    <li> <a href="#"> 新闻 </a> </li>
    <li> <a href="#"> 读过文章 </a> </li>
</ul>
```

最后，在页面初始化脚本中写入下面的代码，调用附加导航插件。在 affix() 构造函数中传入 offset:{top:86} 参数，设置菜单距离窗口顶部偏移距离小于 86 像素时，将激活附加导航插件。

```
<script type="text/JavaScript">
$(function(){
    $("#menu").affix({
          offset:{top:86}
   })
});
</script>
```

在浏览器中预览效果如图 9-15 所示，当滚动滚动条时，导航菜单会被固定在窗口顶部，而不是随滚动条的滚动而消失。

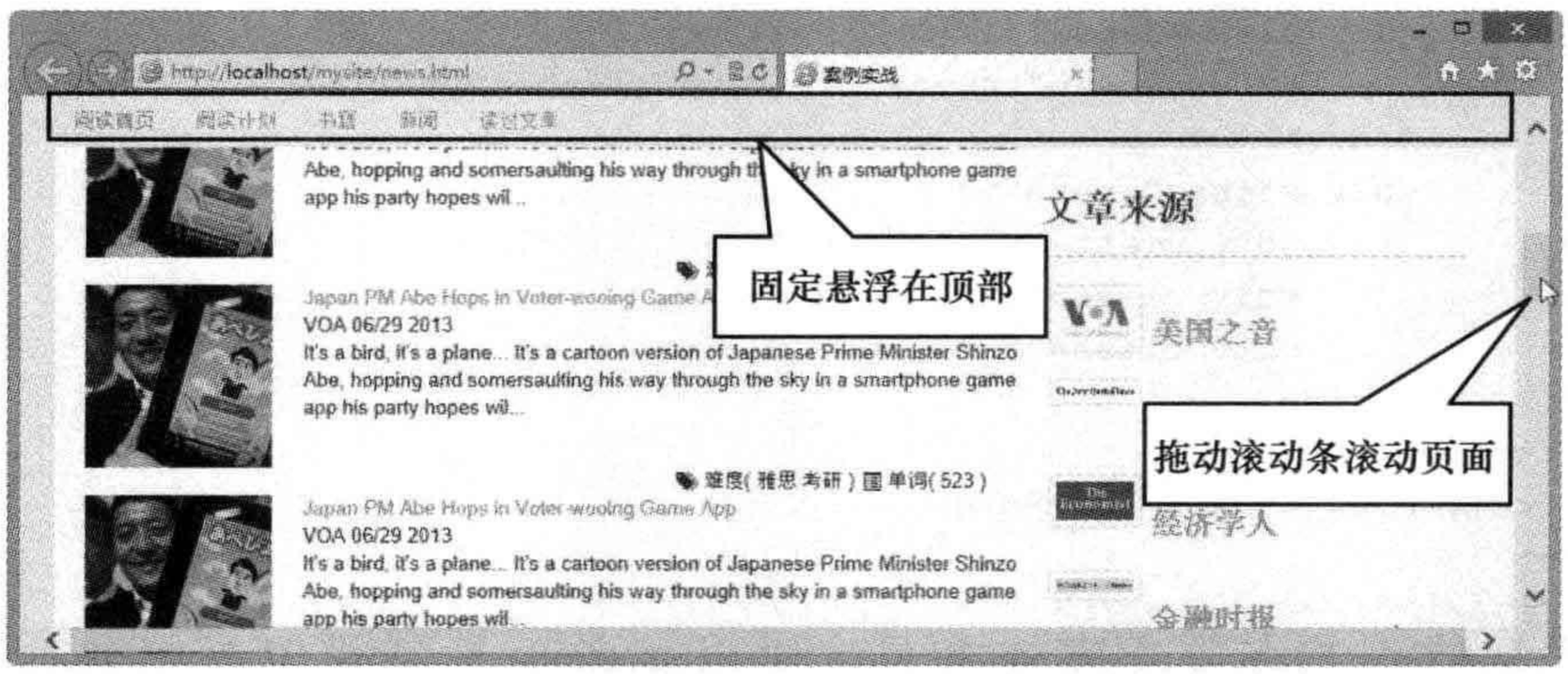


图 9-15　附加导航菜单演示效果

9.3.3　设计页面版式

除了主菜单、附加导航菜单和版权区外，整个页面主体区域使用 Bootstrap 栅格系统完成版式设计。使用 container 设置页面宽度并居中，row 设置行，span8 设置主栏，span4 设置侧栏。

```
<div class="container main-body">
    <div class="row">
        <div class="span8"></div>
        <div class="span4"></div>
    </div>
</div>
```

主栏包括标题区和文章列表，标题区使用 <section> 标签定义，文章列表使用 定义。

```
<div class="span8">
    <section>
        <div class="page-header">
            <h2> 扇贝新闻 </h2>
        </div>
    </section>
    <div class="articles">
```

```
            <div class="article"></div>
            <div class="article"></div>
            ...
        </div>
    </div>
```

每篇文章列表包含两列：左列为 span2，显示新闻图；右列为 span6，显示新闻信息，如标题、信息、内容提要等。在 下一行，设置辅助提示信息。

```
<div class="article">
    <div class="row">
        <div class="span2">
            <div class="thumnail"> <img src="images/787.jpg"/> </div>
        </div>
        <div class="span6">
            <div class="title"> </div>
            <div class="info"></div>
            <div class="summary"> </div>
        </div>
    </div>
    <div class="data row"></div>
</div>
```

侧栏版式结构与左侧基本相同，使用<section>标签设置标签块，使用 设置新闻源列表框，使用 设置新闻源项目，同时添加 row 设计栅格版式，左栏 span1 显示图标，右栏 span3 显示文字。注意，左右栏宽度总和等于 span4 宽度。

```
<div class="span4 width5">
    <section>
        <div class="page-header">
            <h3> 文章来源 </h3>
        </div>
    </section>
    <div class="sources">
        <div id="sources">
            <div class="sources">
                <div class="source row">
                    <div class="span1">
                        <div class="thumbnail"><img src="images/voa.png"></div>
                    </div>
                    <div class="span3">
                        <h3><a href="#"> 美国之音 </a></h3>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

页面版式设计效果如图 9-16 所示。

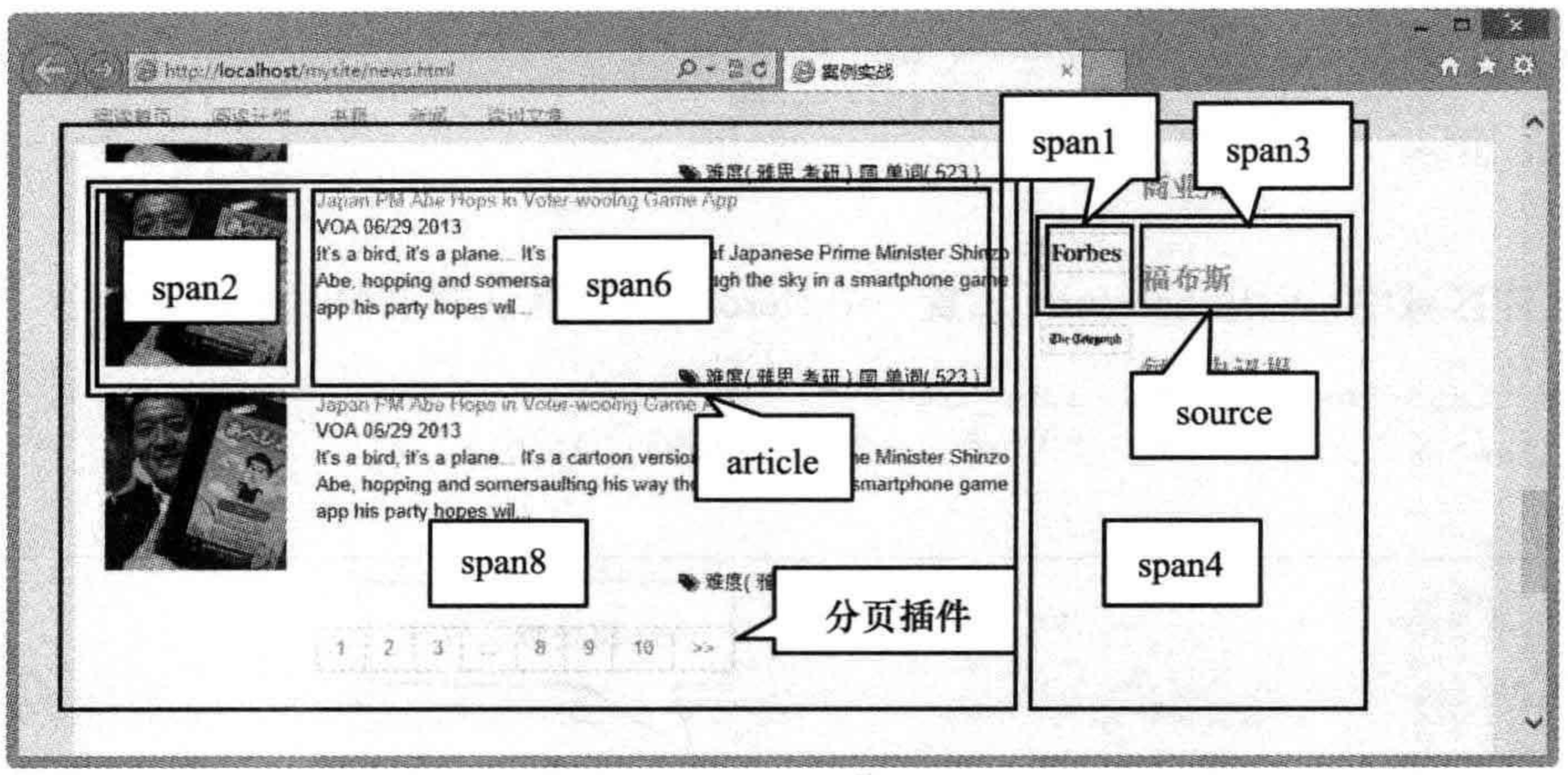


图 9-16　新闻阅读页版式设计效果

9.4　小组页设计

小组页（team.html）借用了新闻阅读页面的主菜单、附加导航菜单和页脚模块的 HTML 结构和代码，这里不再说明，后面的几个页面都共享了这些通用模块，整个页面效果如图 9-17 所示。

图 9-17　小组页面设计效果

首先，构建 Bootstrap 栅格基本结构，该结构与新闻阅读页主体结构相同。然后借助自定义类 bgfff 为主体区域定义白色背景，使用 team_box 和 width4 自定义类设置主栏和侧栏的偏移位置，以及其他显示属性。

```
<div class="container bgfff">
    <div class="row">
        <div class="span8 team_box"></div>
        <div class="span4 width4"></div>
    </div>
</div>
```

在主体区域中，引用 hero-unit 设置一个 Hero 模块，默认效果如图 9-18 所示。

```
<div class="hero-unit header_top">
    <h1> 扇贝小组 <small> 加入小组，一起学习 </small></h1>
</div>
```

图 9-18　Hero 模块默认效果

在本地样式表文件中添加下面两条样式，清除 Hero 默认的红色边框线，让文本居中显示，重置 Padding 距离，添加修饰性背景图，演示效果如图 9-19 所示。

```
.header_top {
    border: none;
    text-align: center;
    padding: 40px;
    background: #eeeeee no-repeat right -20px url(../images/team2.png);
}
.header_top small {
    display: block;
    font-size: 16px;
    margin-top: 1em;
}
```

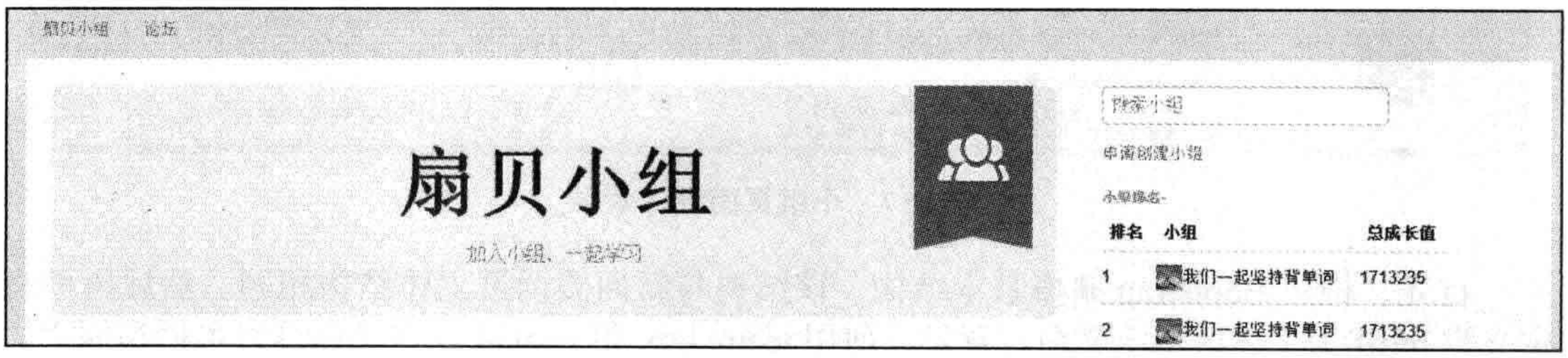


图 9-19　重设的 Hero 模块效果

在 Hero 组件下面是推荐小组和新建小组栏目，这两个模块的结构相同，主要包括栏目标题、小组名称、组长、创建时间以及新行的小组明细。在小组明细行中，使用栅格系统构建一个两列布局版式：左列是小组图标（ ），右列是小组信息（ ）。小组信息列又嵌套一层栅格，左侧是文字信息（ ），右列是提交按钮（ ）。注意，考虑到间距调节，这里特意设置 span5 和 span1 小于 span7，版权效果如图 9-20 所示。

```
<h3 class="team-header"> </h3>
<div class="team-title"></div>
<div class="row">
    <div class="span1"> </div>
    <div class="span7">
        <div class="team-stat"></div>
        <div class="row">
            <div class="span5">
                <h5> </h5>
                <div></div>
            </div>
            <div class="span1"></div>
        </div>
    </div>
</div>
```

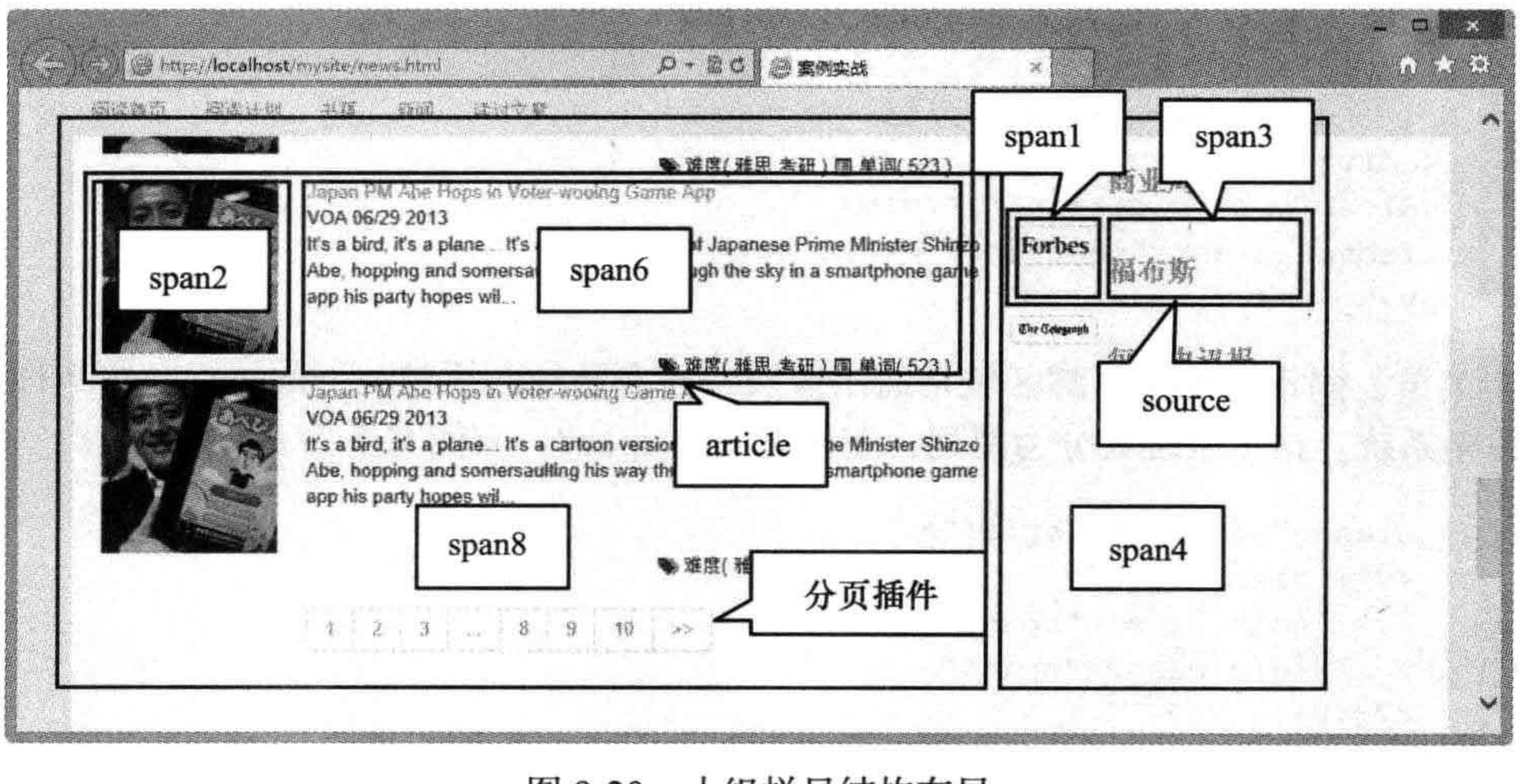



图 9-20　小组栏目结构布局

引入 pagination，在新建小组栏目底部插入分页组件：

```
<div class="span8">
    <div class="pagination">
        <ul>
            <li class="active"><a href="#" rel="page">1</a></li>
            <li><a href="#" rel="page">2</a></li>
            <li><a href="#" rel="page">3</a></li>
            <li><a href="">...</a></li>
```

```
                <li><a href="#" rel="page">123</a></li>
                <li><a href="#" rel="page">124</a></li>
                <li><a href="#" rel="page">125</a></li>
                <li><a href="#" rel="page">&gt;&gt;</a></li>
            </ul>
        </div>
</div>
```

页面右边栏布局思路与左侧相同，这里不再赘述。

9.5 打卡页设计

打卡页类似于日记页，具有流水账版式效果，因此在设计中套用 Bootstrap 栅格系统，同时引用滚动监听和附加导航插件来完善长页面的用户体验问题。

9.5.1 设计页面栅格系统

新建 HTML5 文档，保存为 top.html。先在页面设置 4 行，分别是主菜单（ ）、固定导航（<ul class="nav">）、主体内容（ ）、页脚区域（<footer>）。

```
<body>
    <div class="navbar container"></div>
    <div class="container">
        <ul class="nav nav-pills dropdown"></ul>
    </div>
    <div class="container"></div>
    <footer class="container"></footer>
</body>
```

主菜单、固定导航和页脚区域是通用模块，前面已经介绍过，下面重点介绍页面主体部分的栅格系统。这个系统共分为两列，左侧宽度为 span8，右侧宽度为 span4。

```
<div class="container bgfff">
    <div class="row">
        <div class="span8"> </div>
        <div class="span4"> </div>
    </div>
```

左侧栏目采用通用的结构，其中栏目标题结构如下：

```
<div class="page-header" id="1">
    <h3> 榜样打卡 </h3>
</div>
```

通用模块的单元结构如下，嵌套子栅格系统，其包含两列结构，左侧宽度为 span1，右侧宽度为 span6，它们的宽度之和小于 span8，这是因为在 包含框中通过 margin 和 padding 调整栏目间距，为了避免错位显示，让两列宽度之和小于包含框 <div

class="span8">。读者也可以在本地样式表中重置 span6 的宽度，以避免因调整 margin 和 padding 而导致错位问题。

```
<div class="row row_card">
     <div class="span1">
          <div class="avatar"> </div>
     </div>
     <div class="span6">
          <div class="info"></div>
          <div class="note"> </div>
          <div> </div>
          <div class="btn-group span2 pull-right"> <a class="btn btn-mini drop
           down-toggle"  href="#"  data-toggle="dropdown">分享<span class="caret"></span></a>
              <ul class="dropdown-menu">
                  <li class=""><a href="#">暂无 </a></li>
              </ul>
        </div>
     </div>
</div>
```

右侧栏目是一个简单的导航菜单，使用 nav 设计导航组件，使用 nav-tabs 和 nav-stacked 设计堆叠式标签页样式，页面效果如图 9-21 所示。

```
<div class="span4">
     <div class="nav-box ">
          <div class="page-header">
            <h3>打卡导航 </h3>
          </div>
          <ul class="nav nav-tabs nav-stacked">
            <li><a href="#1">榜样打卡 </a></li>
            <li><a href="#2">最新打卡 </a></li>
            <li><a href="#3">最受欢迎打卡 </a></li>
         </ul>
    </div>
</div>
```

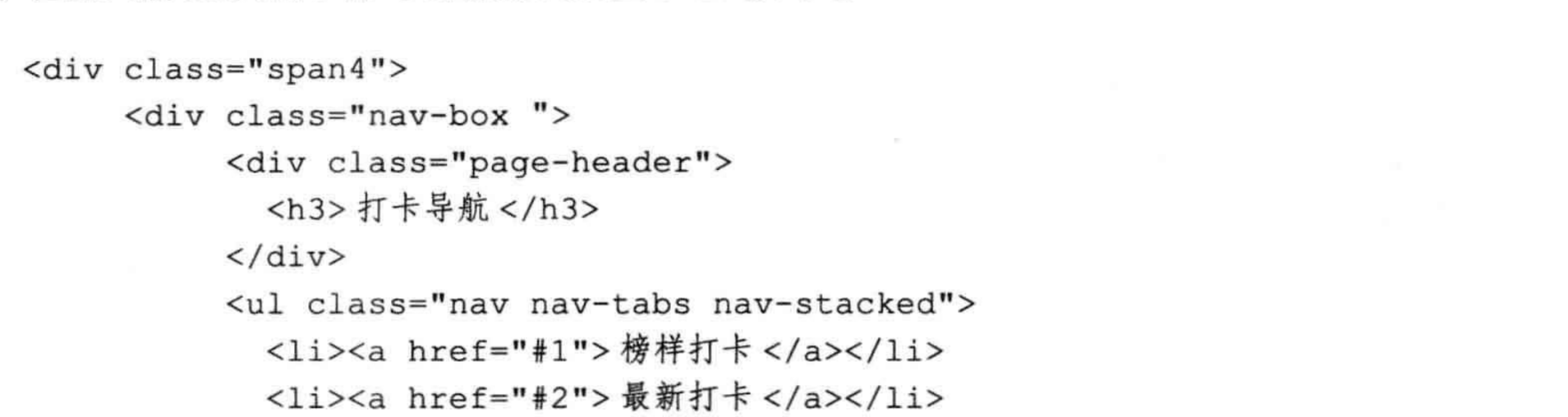


图 9-21　页面版式和导航菜单

9.5.2 设计滚动监听和附加导航

滚动监听是一个非常实用的 Bootstrap 插件，当页面很长时，通过滚动监听可以实现实时导航和跟踪。

首先，在 <body> 标签中定义 data-spy="scroll" 属性，启动全页面滚动监听，设置 data-target=".menu" 属性定义滚动监听对象是包含了 menu 类的菜单。

```
<body   data-spy="scroll" data-target=".menu">
```

在固定导航菜单外面包含一层嵌套结构，定义 class 为 menu。

```
<div class="container menu">
    <ul class="nav nav-pills dropdown" id="menu">
        ......
    </ul>
</div>
```

在右侧标签页导航中也包含一层结构，添加 class 的值为 menu。

```
<div class="nav-box span3  menu">
     <ul class="nav nav-tabs nav-stacked">
         ......
     </ul>
</div>
```

这样当滚动浏览器窗口内的滚动条时，将会自动侦测并准确定位被激活的菜单项。由于在页面滚动过程中导航菜单也会自动滚动，因此在调用附加导航插件，设计菜单滚出页面时，把它们固定在页面中固定位置。

```
<script type="text/JavaScript">
$(function(){
   $("#menu").affix({
        offset:{top:86}
   })
   $(".nav-box").affix({
        offset:{top:40}
   })
});
</script>
```

最后，还需要在本地样式表文件中添加如下两条样式，以固定导航菜单和标签页在页面中的显示位置，避免在滚动过程中上下晃动，同时设置导航菜单背景色为灰色，避免透明背景带来的上下文字重叠问题，并定义其宽度为 100%，避免宽度固定后自动收缩显示。

```
#menu {
    background-color: #E4E4E4;
    width: 100%;
    top: 0;
}
.nav-box { top: 40px; }
```

在浏览器中预览页面，当滚动滚动条时，会发现滚动监听和附加导航协同工作，极大地提高了用户体验，效果如图 9-22 所示。

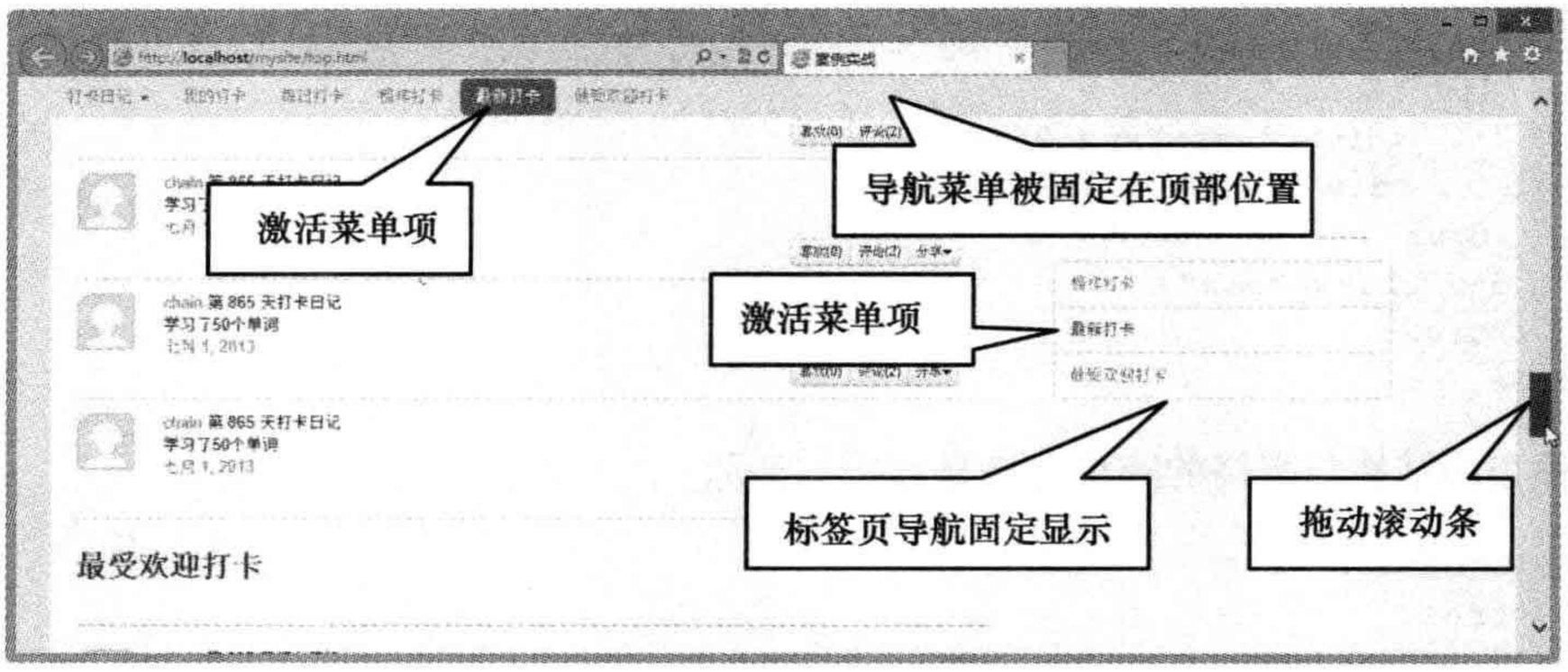


图 9-22　页面版式和导航菜单

9.6　词根页设计

完成前几页主页面设计工作之后，后面的工作变得容易起来，因为其他页面可以快速仿制，结构和样式基本保持一致。词根页（market.html）保留基本的设计风格和布局思路，为了应用 Bootstrap 各种组件，这里简单介绍几个技术细节。

设计主菜单固定显示。如果希望主菜单能够固定在窗口顶部显示，可以引用 navbar-fixed-top 来实现。添加的代码如下，演示效果如图 9-23 所示。

```
<div class="navbar navbar-fixed-top">
     <div class="navbar-inner">
          <div class="container">
          </div>
     </div>
</div>
```

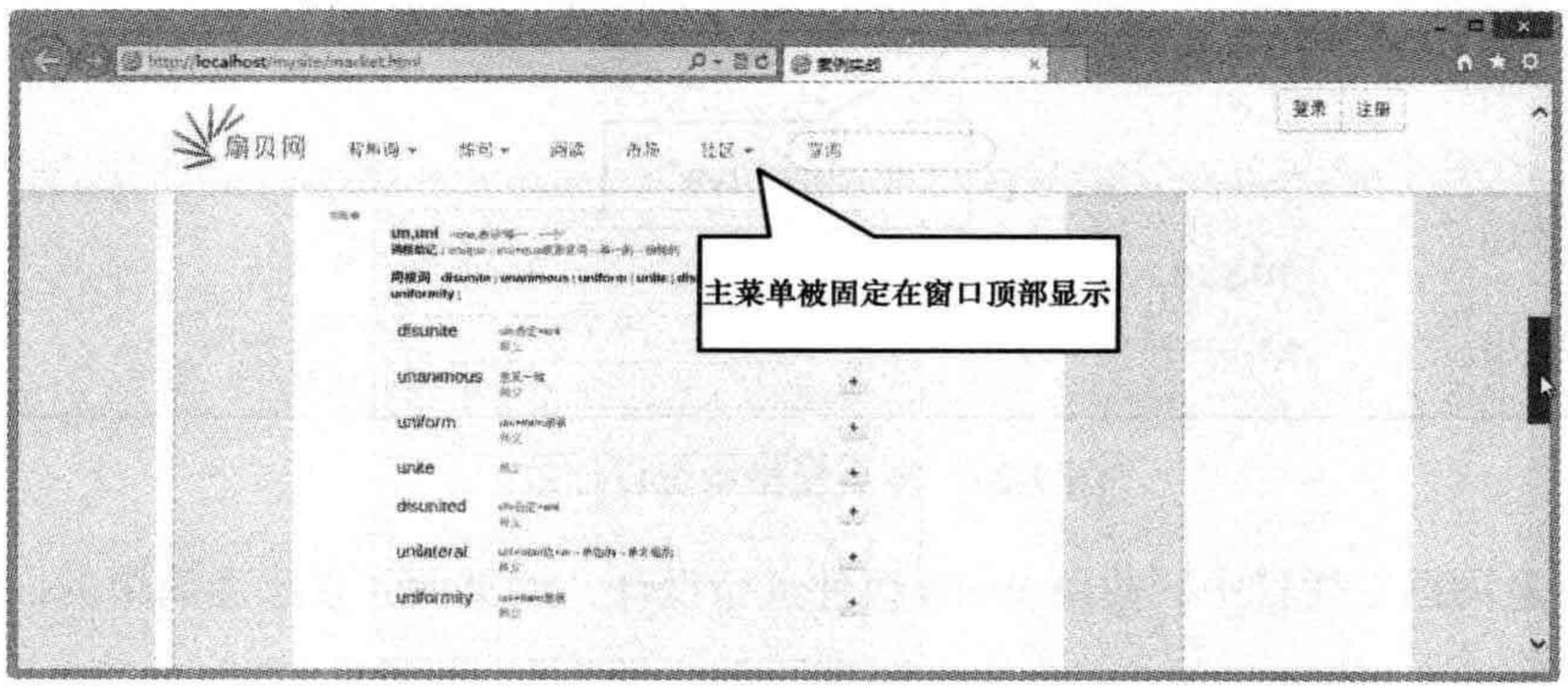


图 9-23　设计主菜单固顶显示

页面主体区域通过 Bootstrap 栅格系统进行设计，分为两行两列：第一行为两列显示，第二行为一列显示，结构如下：

```
<div class="container main-body market">
    <div class="row">
        <div class="span8"></div>
        <div class="span3"></div>
    </div>
    <div class="row">
    </div>
</div>
```

这种布局结构与表格的跨单元格显示结构类似：

```
<table border="1">
    <tr>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td colspan="2"></td>
    </tr>
</table>
```

显示效果如图 9-24 所示。

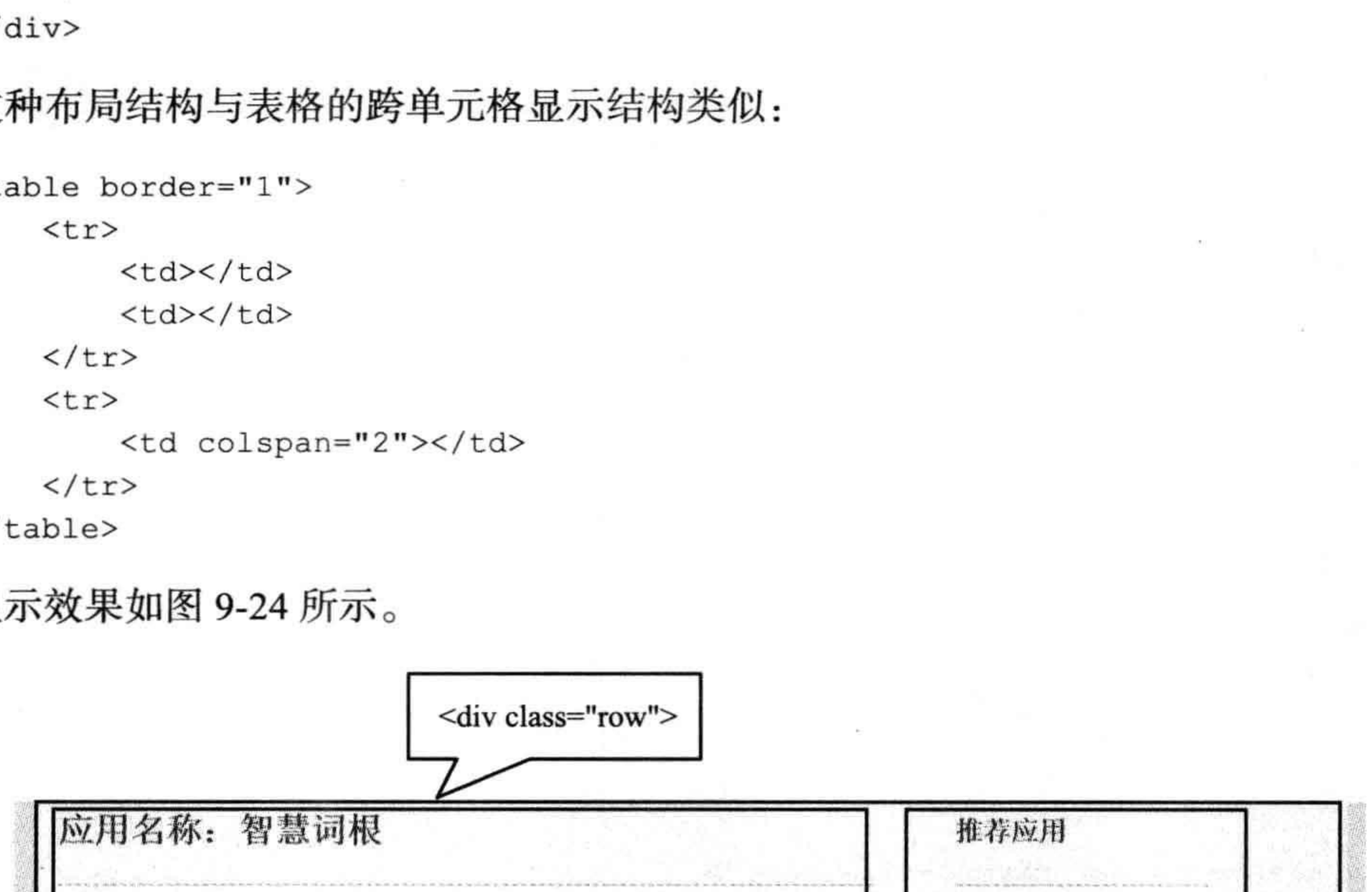


图 9-24　本页栅格系统设计效果

在“智慧词根”栏目中，使用 Wells 组件进行设计，给页面元素添加简单的嵌入效果。

```
<div class="well">
    <p> <b> 应用介绍：</b>
```

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

“推荐应用”栏目使用了一个堆叠式标签页组件进行设计，效果如图 9-24 所示。

```
<ul class="nav nav-stacked nav-tabs">
    <li class=""><a href="#"> 单词配图 </a></li>
    <li class=""><a href="#"> 音节划分 </a></li>
    <li class="active"><a href="#"> 智慧词根 </a></li>
    <li class=""><a href="#"> 派生词 </a></li>
</ul>
```

在第二行结构中，使用细线表格风格进行设计，并通过 span9 设计表格显示的宽度。

```
<table class="table table-bordered span9"
</table>
```

然后把表格包含在 框内，把表格与 Wells 组件包装在一起，这样就可以得到如图 9-25 所示的效果。

图 9-25　设计表格效果

到此，本示例设计基本完成。当然，随着网站规模的扩大，可能还有更多的主页面需要设计，本示例只展示了如何应用 Bootstrap 框架来快速设计一个成熟的、富有弹性的页面。本示例应用了 Bootstrap 的大部分组件、实用插件，以及栅格布局系统。通过该示例的学习，相信读者会对 Bootstrap 有更深的理解和认识。

第 10 章

Bootstrap 内核解码

本章内容

- 定义 jQuery 插件
- Bootstrap 设计思想
- Bootstrap 框架解析
- Bootstrap 内核解疑

通过前面的系统学习，相信读者已经能够熟练使用 Bootstrap，并体验到 Bootstrap 的强悍之处。Bootstrap 所提供的类型如此庞大，并可以轻松使用，以便更随心所欲地设计页面。即便如此，如果一些代码遵循一定的使用模式，并在开发中反复应用，那么建议读者不妨把这些代码添加到 Bootstrap 扩展中，以便扩展自己的代码库，当然前提是读者了解 Bootstrap 源代码，并能把握其设计思路和内核结构。

本章将从 Bootstrap 应用阶段上升到源码分析阶段，帮助读者掌握 Bootstrap 设计原理，为 Bootstrap 二次开发打好基础。

10.1　定义 jQuery 插件

jQuery 允许开发人员自定义 jQuery 扩展功能，并提供了友好的接口。Bootstrap 框架也是 jQuery 的一个扩展插件，因此也应该遵循 jQuery 插件的基本设计原则，并保持相同用法。

10.1.1　jQuery 插件形式

开发 jQuery 插件包括三种形式。

（1）jQuery 方法

这种形式的插件是把一些常用或者重复使用的功能定义为方法，然后绑定到 jQuery 对象上，从而成为 jQuery 对象的一个扩展方法。对 jQuery 包装集进行操作的方法即所谓的 jQuery 命令。

大部分 jQuery 插件都是这种形式的插件，这种插件是将对象方法封装起来，用于对通过 jQuery 选择器获取的 jQuery 对象进行操作，从而发挥 jQuery 强大的选择器优势。有很多 jQuery 内部方法也是通过在 jQuery 脚本内部通过这种形式插入到 jQuery 框架中，如 parent()、appendTo()、addClass() 等方法。

Bootstrap 插件都是这种形式的插件，在使用时可以直接在 jQuery 对象上调用 Bootstrap 方法。

（2）工具函数

在 $（jQuery 的别名）上直接定义实用工具函数，把自定义的工具函数独立附加到 jQuery 命名空间下，从而作为 jQuery 作用域下的公共函数使用。

例如，jQuery 的 ajax() 方法就是利用这种途径内部定义的全局函数。由于全局函数没有绑定到 jQuery 对象上，故不能够在选择器获取的 jQuery 对象上直接调用，需要通过 jQuery.fn() 或者 $.fn() 方式进行引用。

（3）自定义选择器

jQuery 提供了强大的选择器，当然用户也可以自定义选择器，以满足特定环境下选择元素的需要。

10.1.2　jQuery 插件规范

jQuery 开发团队制定了 jQuery 插件通用规则，为用户创建一个通用而可信的环境。因

此，建议读者在自定义插件之前阅读并遵守这些规则，确保自定义插件与其他代码融合。遵守这些规则非常重要，它不仅保证插件代码的统一性，还能增加插件的成功几率。

1. 命名规则

```
jquery. plug-in_name.js
```

其中 plug-in_name 表示插件的名称，在这个文件中，所有全局函数都应该包含在名为 plug-in_name 的对象中。如果插件只有一个函数，则可以考虑使用 jQuery.plug-in_name() 形式。

插件中的对象方法可以灵活命名，但是应保持相同的命名风格。如果定义多个方法，建议在方法名前添加插件名前缀，以保持清晰。不建议使用过于简短的名称、语义含糊的缩写名，或者公共方法名，如 set()、get() 等，这样很容易与外界的方法混淆。

Bootstrap 插件由于是在 Bootstrap 框架下进行定义，因而在插件命名时，没有严格遵循 jQuery 插件命名规则，而是遵循自己的一套规则，以 bootstrap 为前缀，后面附加功能名称，如 bootstrap-affix.js、bootstrap-alert.js 和 bootstrap-button.js 等。

2. 编码规则

1）所有新方法都附加到 jQuery.fn 对象上。

2）所有新功能都附加到 jQuery 对象上。

3. this 指代

jQuery 插件内的 this 应该引用 jQuery 对象。

让所有插件在引用 this 时，知道从 jQuery 接收到哪个对象。所有 jQuery 方法都是在一个 jQuery 对象的环境中调用的，因此函数体中 this 关键字总是指向该函数的上下文，即 this 此时是一个包含多个 DOM 元素的伪数组（Object 对象）。

但是，在插件函数内部方法中，this 不再指代当前 jQuery 对象，而是 jQuery 对象中包含的每一个 DOM 元素。例如，在 Bootstrap 插件 bootstrap-alert.js 中，外层的 this 指代匹配所有 DOM 元素的 jQuery 对象，而在 each() 函数内，this 指代的是每个匹配的 DOM 元素，因此在使用时还必须把 this 包装为 jQuery 对象（var $this=$（this））才能够正确使用。

```
$.fn.alert = function (option) {
    return this.each(function () {
        var $this = $(this),
          data = $this.data('alert')
        if (!data) $this.data('alert', (data = new Alert(this)))
        if (typeof option == 'string') data[option].call($this)
    })
}
```

4. 迭代元素

使用 this.each() 迭代匹配的元素。

插件应该调用 this.each() 来迭代所有匹配的元素，然后依次操作每个 DOM 元素。在

this.each() 方法体内，this 就不再引用 jQuery 对象，而是引用当前匹配的 DOM 元素对象。

5. 返回 jQuery 对象

插件应该有返回值，除了特定需求外，所有方法都必须返回 jQuery 对象。

一般都应该返回当前上下文环境中的 jQuery 对象，即 this 关键字引用的数组。通过这种方式，可以保持 jQuery 框架内方法的连续行为，即链式语法。如果破坏这种规则，就会给用户开发带来诸多不便。

如果匹配的对象集合被修改，则应该通过调用 pushStack() 方法创建新的 jQuery 对象，并返回这个新对象，如果返回值不是 jQuery 对象，则应该明确说明。

6. 语法严谨

插件中定义的所有方法或函数，末尾都必须加上分号（即；），以方便代码压缩。

7. 区别 jQuery 和 $

在插件代码中总是使用“jQuery”，而不是“$”。

$ 并不总是等于 jQuery，这一点很重要。对于 var JQ=jQuery.noConflict(); 函数，如果将 jQuery 替换为 $ 别名，那么就会引发错误。另外，其他 JavaScript 框架也可能使用 $ 别名。

在复杂的插件中，如果全部使用“jQuery”代替“$”，又会让人难以接受这种复杂的写法，为了解决这个问题，建议使用如下插件模式：

```
(function($){
    //在插件包中使用 $ 代替 jQuery
})(jQuery);
```

这个包装函数接受一个参数，该参数传递的是 jQuery 全局对象，由于参数被命名为 $，因此在函数体内就可以安全使用 $ 别名，而不用担心命名冲突。

10.1.3　jQuery 插件封装

封装 jQuery 插件的第一步是定义一个独立域，代码如下所示。

```
(function($){
    //自定义插件代码
})(jQuery)      //封装插件
```

确定创建插件类型，选择创建方式。例如，创建一个设置元素字体颜色的插件，则应该创建 jQuery 对象方法。考虑到 jQuery 提供了插件扩展方法 extend()，调用该方法定义插件会更为规范。

```
(function($){
    $.extend($.fn,{        //jQuery 对象方法扩展
        //函数列表
    })
})(jQuery)                 //封装插件
```

一般插件都会接受参数，用来控制插件的行为。根据 jQuery 设计习惯，可以把所有参数以列表形式封装在选项对象中。例如，对于设置元素字体颜色的插件，应该允许用户设置字体颜色，同时还应考虑如果用户没有设置颜色，则应确保使用默认色进行设置。实现代码如下。

```
(function($){
    $.extend($.fn,{
        color: function(options){                    //自定义插件名称
            var options = $.extend({                 //参数选项对象处理
                bcolor: "white",                     //背景色默认值
                fcolor: "black"                      //前景色默认值
            },options);
            return this.each(function(){             //返回匹配的 jQuery 对象
                $(this).css("color", options.fcolor);
                                                     //遍历设置每个 DOM 元素字体颜色
                $(this).css("backgroundColor", options.bcolor);
                                                     //遍历设置每个 DOM 元素背景颜色
            })
        }
    })
})(jQuery);                                          //封装插件
```

完成插件封装之后，应该测试一下自定义的 color() 方法，演示效果如图 10-1 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<script type="text/JavaScript" src="bootstrap/js/jquery-1.9.1.js" ></script>
<script type="text/JavaScript" >
(function($){
    $.extend($.fn,{
        color: function(options){
            var options = $.extend({
                bcolor: "white",
                fcolor: "black"
            },options);
            return this.each(function(){
                $(this).css("color", options.fcolor);
                $(this).css("backgroundColor", options.bcolor);
            })
        }
    })
})(jQuery);                                          //封装插件
$(function(){
    $("p").color({
            bcolor: "blue",
            fcolor: "red"
    });
```

```
})
</script>
</head>
<body>
<p> 段落文本 1</p>
<p> 段落文本 2</p>
<p> 段落文本 3</p>
<p> 段落文本 4</p>
<p> 段落文本 5</p>
<p> 段落文本 6</p>
</body>
</html>
```

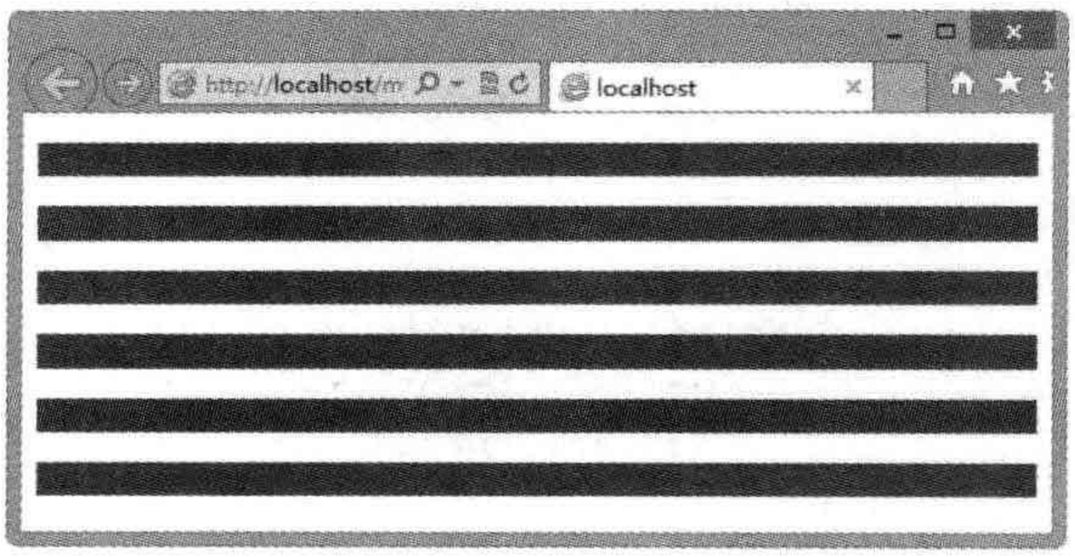


图 10-1　封装 jQuery 插件

10.1.4　jQuery 插件优化

如果要发布自定义 jQuery 插件，用户应该确保插件的开放性和封闭性，其他用户能够使用，同时还应避免冲突或者破坏。

1. 允许定义默认设置

插件一般都需要用户设置参数，这个参数是灵活应用插件的体现，当然开发人员也应该考虑到用户不设置参数时该如何设置默认值，避免程序出错。同时，可以考虑开放插件的默认设置，这对于插件使用者来说，会更容易使用和修改。为了探讨这个主题，下面以示例的形式进行讲解。

例如，继续以上面的代码为例进行说明，把其中的参数默认值作为 $.fn.color 对象的属性单独进行设计，然后借助 jQuery.extend() 覆盖原来的参数选项即可。在 color() 函数中，$.extend() 方法能够使用参数 options 覆盖默认的 defaults 属性值，如果没有设置 options 参数值，则使用 defaults 属性值。由于 defaults 属性是单独定义，可以在页面中预设前景色和背景色，然后就可以多次调用 color() 方法，完整的示例代码如下，演示效果如图 10-2 所示。

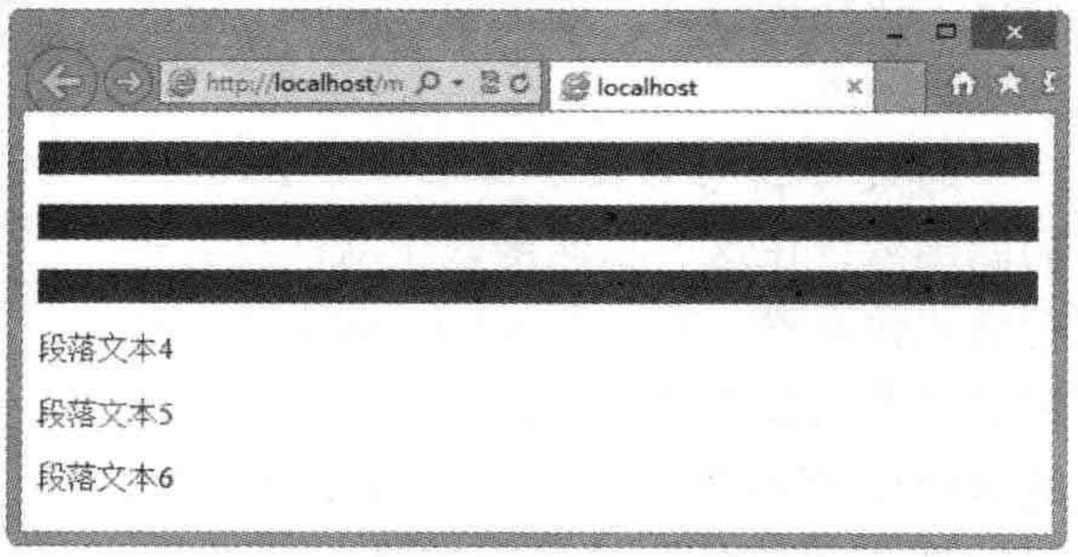


图 10-2　动态定义默认值

```
<script type="text/JavaScript" >
(function($){
    $.extend($.fn,{
        color: function(options){
            var options = $.extend({}, $.fn.color.defaults, options); //覆盖原来参数
            return this.each(function(){
                $(this).css("color", options.fcolor);
                $(this).css("backgroundColor", options.bcolor);
            })
        }
```

```
    })
    $.fn.color.defaults = {                          //独立设置 $.fn.color 对象的默认参数值
          bcolor: "white",
          fcolor: "black"
    }
})(jQuery);
$(function(){
     $.fn.color.defaults = {                         //预设默认的前景色和背景色
          bcolor: "red",
          fcolor: "blue"
      }
    $("p").color();                                  //设置默认色
    $("p:gt(2)").color({bcolor:"#fff"});             //设置默认色，同时覆盖背景色为白色
})
</script>
```

通过这种开发插件默认参数的做法，用户不再需要重复定义参数，这样就可以节省开发时间。

2. 让插件具有功能扩展性

实际上在封装插件时，开发人员是无法确保把所有功能都封装进去，也没有办法预知用户的所有需求，此时最友好的方式就是让用户自己设计或者添加部分功能，从而使该插件满足不同用户的不同需求。

继续以 10.1.3 节的示例为基础，为 color() 命令添加一个格式化的扩展功能，这样用户在设置颜色的同时，还可以根据需要适当进行格式化功能设计，如加粗、斜体、放大等功能操作。

通过开放的方式定义了一个 format() 功能函数，在这个功能函数中默认没有进行格式化设置，然后在 color() 函数体内利用这个开放性功能函数格式化当前元素内的 HTML 字符串。最后调用 color() 命令，同时在调用前分别扩展它的格式化功能，演示效果如图 10-3 所示。

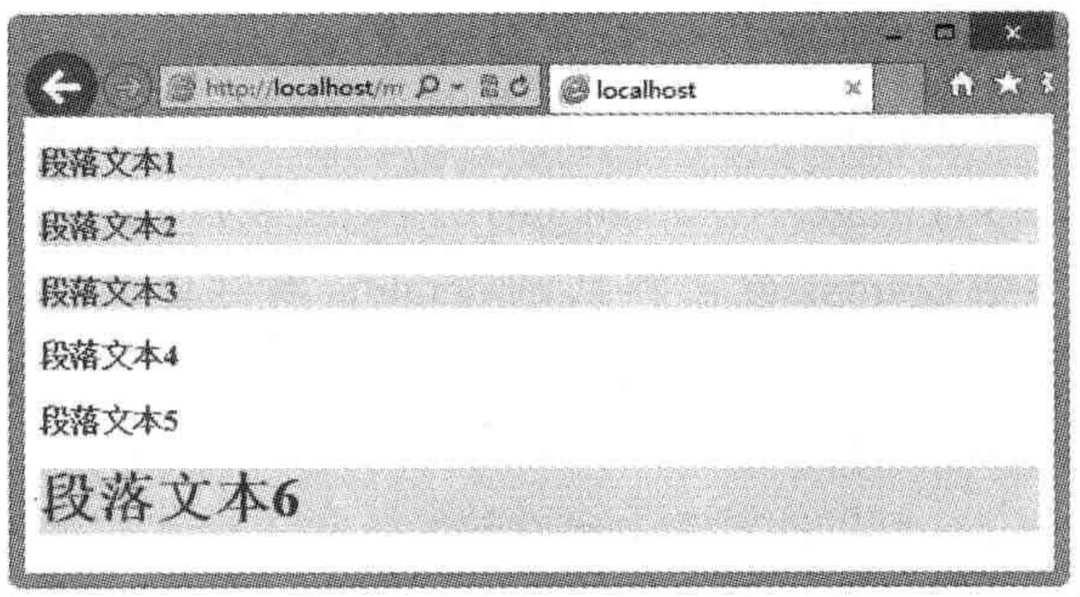


图 10-3　开放 color() 插件

```
<script type="text/JavaScript" >
(function($){
     $.extend($.fn,{
          color: function(options){
              var options = $.extend({}, $.fn.color.defaults, options);
                                                     //覆盖原来参数
              return this.each(function(){
                $(this).css("color", options.fcolor);
                $(this).css("backgroundColor", options.bcolor);
```

```
            var _html = $(this).html();                    //获取当前元素包含的 HTML 字符串
            _html = $.fn.color.format(_html);              //调用格式化功能函数对其进行格式化
            $(this).html(_html);                           //使用格式化的 HTML 字符串重写
                                                           //当前元素内容
        })
    }
})
$.fn.color.defaults = {                                    //独立设置 $.fn.color 对象的
                                                           //默认参数值
    bcolor: "white",
    fcolor: "black"
}
$.fn.color.format = function(str){                         //开放的功能函数
    return str;
};
})(jQuery);
$(function(){
    $.fn.color.defaults = {                                //预设默认的前景色和背景色
        bcolor: "#eea",
        fcolor: "red"
    }
    $.fn.color.format = function(str){                     //扩展 color() 插件的功能，使
                                                           //内部文本加粗显示
        return "<strong>" + str + "</strong>";
    }
    $("p").color();
    $("p:gt(2)").color({bcolor:"#fff"});
    $.fn.color.format = function(str){                     //扩展 color() 插件的功能，使
                                                           //内部文本放大显示
        return "<span style='font-size:30px;'>" + str + "</span>";
    }
    $("p:gt(4)").color();
})
</script>
```

允许用户自定义功能设置，以覆盖插件默认的功能，从而方便其他用户以当前插件为基础进一步扩写插件。

3. 让插件具有隐私功能

在设计插件时必须考虑插件暴露和隐私的关系，该暴露的功能就应该完全开放，不该暴露的属性或者成员就应该保护它的隐私。一旦被暴露，就需要铭记保持任何对于参数或者语义的改动也许会破坏向后的兼容性。如果不能确定应该暴露特定的函数，那么就必须考虑如何进行保护的问题。

如果插件包含很多函数，那么要想在设计时希望这么多函数不搅乱命名空间，也不会被完全暴露，唯一的方法就是使用闭包。为了创建闭包，可以将整个插件封装在一个函数中即可。

继续以 10.1.3 节示例进行讲解，为了验证用户在调用 color() 方法时所传递参数的合法性，不妨在插件中定义一个参数验证函数，但是该验证函数是不允许外界侵入或者访问的，此时可以借助闭包把它隐藏起来，只允许在插件内部进行访问。下面的示例中定义了一个验

证函数，这个函数作为 color() 插件的私有函数，是不允许外界访问和修改的。用它来验证用户传递的参数是否合法，对于非法参数设置，则忽略该方法调用，但是不会抛出异常。

```
<script type="text/JavaScript" >
(function($){
    $.extend($.fn,{
        color: function(options){
            if(!filter(options))          //调用隐私方法验证参数，不合法则返回
                return this;
            var options = $.extend({}, $.fn.color.defaults, options);
            return this.each(function(){
                $(this).css("color", options.fcolor);
                $(this).css("backgroundColor", options.bcolor);
                var _html = $(this).html();
                _html = $.fn.color.format(_html);
                $(this).html(_html);
            })
        }
});
$.fn.color.defaults = {                   //独立设置 $.fn.color 对象的默认参数值
    bcolor: "white",
    fcolor: "black"
}
$.fn.color.format = function(str){  //开放的功能函数
        return str;
    };
    function filter(options){             //定义隐私函数，外界无法访问
                                          //如果参数不存在，或者存在且为对象，则返回
                                          //true，否则返回 false
        return !options || (options && typeof options === "object")?true : false;
    }
})(jQuery);
$(function(){
    $("p").color("#fff");
})
</script>
```

4. 让插件具有非破坏性

在特定情况下，jQuery 对象方法可能会修改 jQuery 对象匹配的 DOM 元素，这时就有可能破坏方法返回值的一致性。为了遵循 jQuery 框架的核心设计理念，应该避免修改 jQuery 对象。

定义一个 jQuery 对象方法 parent()，获取 jQuery 匹配的所有 DOM 元素的父元素。该方法通过遍历所有匹配元素，获取每个 DOM 元素的父元素，并把这些父元素存储到一个临时数组中，通过过滤、打包，最后返回。

例如，利用自定义的插件 parent() 为所有 p 元素的父元素添加一个边框，示例代码如下所示。

```
<script type="text/JavaScript" >
//自定义插件，扩展 jQuery 对象方法，获取所有匹配元素的父元素
```

```
(function($){
    $.extend($.fn,{
        parent: function(options){
            var arr = [];
            $.each(this, function(index, value){          //遍历匹配的 DOM 元素
                arr.push(value.parentNode);               //把匹配元素的父元素推入临时数组
            });
            arr = $.unique(arr);                          //把临时数组中过滤重复的元素
             this.length = 0;
            Array.prototype.push.apply( this, arr );
                                                          //把变量 arr 打包为伪数组类型返回
            return this;
        }
    })
})(jQuery);
$(function(){
    var $p = $("p");
    $p.parent().css("border","solid 1px red");
})
</script>
<style type="text/css">
div.big { width: 400px; height: 400px; }
div.small { width: 200px; height: 200px; }
</style>

<div class="big">
    <p> </p>
    <div class="small">
        <p> </p>
    </div>
</div>
```

如果在设置了父元素的边框后，希望把 jQuery 对象匹配的所有元素隐藏起来，则可以添加下面的代码，在浏览器中预览就会发现 div 元素也被隐藏起来了。

```
$(function(){
    var $p = $("p");
    $p.parent().css("border","solid 1px red");
    $p.hide();
})
```

也就是说，上面的代码中 $p 变量已经被修改，它指向的不再是当前 jQuery 对象，而是 jQuery 对象匹配元素的父元素，因此为 $p 调用 hide() 方法，就会隐藏 div 元素，而不是 p 元素。

这是破坏性操作的一种表现，如果要避免此类行为，建议采用非破坏性操作。例如，在本例中可以使用 pushStack() 方法创建一个新的 jQuery 对象，而不是修改 this 所引用的 jQuery 对象。同时 pushStack() 方法还允许调用 end() 方法操作新创建的 jQuery 对象方法。把上面示例的 jQuery 对象方法进行优化。

```
(function($){
    $.extend($.fn,{
```

```
        parent: function(options){
            var arr = [];
            $.each(this, function(index, value){
                arr.push(value.parentNode);
            });
            arr = $.unique(arr);
            return this.pushStack(arr);
                        //返回新创建的 jQuery 对象，而不是修改后的当前 jQuery 对象
        }
    })
})(jQuery);
```

这时，如果继续执行上面的演示示例操作，则可以看到 div 元素边框样式被定义为红色实现，同时也隐藏了其包含的 p 元素。读者也可以采用连续行为进行编写，其中 end() 方法能够恢复被破坏的 jQuery 对象，也就是说，parent() 方法返回的是当前元素的父元素的集合。现在调用 end() 方法之后，又恢复到最初的当前元素集合，此时可以继续调用方法作用于原来的 jQuery 对象了。

```
$(function(){
    var $p = $("p");
    $p.parent().css("border","solid 1px red").end().hide();
})
```

10.2 Bootstrap 设计思想

Bootstrap 是一个前端开发工具集，实际上它更多的是一个 CSS 框架，提供了一套易用、优雅、灵活、可扩展的样式库，内容包含了构架基本 Web 应用程序所需的组件，思路清晰，样式精美，值得前端开发人员学习和借鉴。基于 Bootstrap 设计的页面界面清新、简洁，要素排版利落大方。

10.2.1 类型化

类型化设计是样式优化最有效的途径之一。Bootstrap 主要使用类（Class）选择器定义样式库，适当通过标签选择器进行样式重设，以及类型个性化设计的限制。类型化设计也是抽象化编程思想的应用。当所有类汇集在一起，通过一定的逻辑组织在一起时，就形成了类库。

在 bootstrap.css 文档中，用户会发现样式类与标签样式构成了全部样式，但是 Bootstrap 并没有大量重置标签默认样式，仅对个别浏览器解析存在差异的标签进行样式统一，同时对于一些标签缺陷样式进行修补，以实现标准化视觉设计要求。

大量地使用样式类，尽量避免破坏标签默认样式，这是 Bootstrap 的设计准则。例如，针对表格样式，Bootstrap 没有直接对 table 元素进行重置，而是通过 .table 类对表格样式标准化，然后在 .table 类下面又发展很多子类。

```
.table {
    width: 100%;
```

```
    margin-bottom: 20px;
}
.table th,
.table td {}
.table th {}
.table thead th {}
.table tbody + tbody {}
.table .table {}
```

这种简单的 CSS 类功能增强了 Bootstrap 的灵活性，使其在网页中广泛应用，为 CSS 样式的抽象性提供了一种选择。

是不是定义样式类就很简单呢？不是，样式类的定义方法很简单，但是设计好一套比较实用的类库就很不容易了。下面几点是 Bootstrap 在设计时遵循的基本原则。

第一，CSS 的类应体现最小化效果设计原则。这样就能够更灵活地应用样式类。例如，在设计栅格系统时，在列类型中仅指定宽度属性，通用属性通过属性选择器定义，这样设计的样式就比较灵活。

```
.span1 {
    width: 60px;
}
[class*="span"] {
    float: left;
    min-height: 1px;
    margin-left: 20px;
}
```

而在流式栅格系统中，通过子类定义列类型，实现样式的实用性。

```
.row-fluid .span1 {
    width: 6.382978723404255%;
   *width: 6.329787234042553%;
}
```

通过这种方法，用户可以在一个对象中引用多个样式类，它们的位置顺序不会对样式产生影响。

当然对于这个问题也不是绝对的，所谓最小化效果单元就是一个样式类中，如果几个声明被分开之后，没有被重复利用的价值，就不应该再分开定义。例如，看看下面这个文本隐藏类：

```
.hide-text {
    font: 0/0 a;
    color: transparent;
    text-shadow: none;
    background-color: transparent;
    border: 0;
}
```

对于这个隐藏类，其中定义了 5 个属性，这些属性都是针对隐藏文本来设计的，此时就

不能够把它们拆分为 5 个小类。一方面，这 5 个属性都是针对同一个样式类的效果进行定义的，拆分之后没有意义。另一方面，这些被拆分的小类实用价值不高，没有必要为此定义多个小类。

第二，CSS 类型体现通用性。所谓通用性是指具备广泛的应用价值。定义类时除了上面讲的应该尽可能定义小的样式单元，还应该保证所定义的类具有代表性。例如，下面的类型是页面中经常用到的基本样式，因此可以把它们独立出来进行设计。

```
.pull-right { float: right; }
.pull-left { float: left; }
.hide { display: none; }
.show { display: block; }
.invisible { visibility: hidden; }
```

第三，当定义 CSS 类库时，要遵循规律，如在命名样式类时要有规律，这样在使用和参阅时也可以快速浏览。例如，有关按钮样式类的定义中，通过 btn 前缀，把所有与按钮相关的样式统一起来，这样能够方便使用，也方便管理。

```
.btn{}
.btn-large {}
.btn-small {}
.btn-mini {}
.btn-block {}
.btn-primary{}
.btn-warning{}
.btn-danger{}
.btn-success{}
.btn-info{}
.btn-inverse{}
```

10.2.2 松散与耦合处理

Bootstrap 样式库具有很强的模块化设计特性，没有系统性，代码非常松散，但是 Bootstrap 利用应用体系把这些松散的样式码收拢在一起，避免 CSS 代码冗余。注意，Bootstrap 类样式的体系主要通过 LESS 来设计，用户可以通过 LESS 源代码了解其体系设计思路。

1. 布局类

Bootstrap 设计了一套栅格系统，该栅格系统历经锤炼，在 Web 应用社区化发展中具有不可小视的价值，并在同类栅格系统中脱颖而出。虽然网页布局中一般都是针对具体的模块进行定义，能够提炼的样式类没有很大的实用价值，但是 Bootstrap 栅格系统提供了经典的视觉效果和标准的 Web 设计标准，因此这套栅格系统仍然具有很高的参考价值，建议广大用户采用。

整个栅格系统分为两类：固定式和流式。从结构上来分，包括布局框、布局行和布局列，与表格结构具有某种相似性。

```
.container {/* 栅格包含框 */
     width: 940px;
}
.row {/* 栅格行 */
     margin-left: -20px;
    *zoom: 1;
}
[class*="span"] {/* 栅格列 */
     float: left;
     min-height: 1px;
     margin-left: 20px;
}
```

然后，Bootstrap 在这三个基础样式上拓展了大量的关联样式类，在此就不再一一罗列。

2. 功能类

Bootstrap 根据页面对象的功能分门别类地设计多种样式，如按钮类、表格类、表单类、版式、文本代码、图片等。下面简单地列出按钮类、表格类的祖样式类。

```
.btn {

     display: inline-block;
     *display: inline;
     padding: 4px 12px;
     margin-bottom: 0;
     *margin-left: .3em;
     ......
}
.table {
     width: 100%;
     margin-bottom: 20px;
}
```


类似的功能类还有很多，读者可以查看 bootstrap.css 源代码。

3. 基本类

这类样式主要是针对 CSS 中通用组件进行定义。例如，下面的样式类都是导航相关的组件样式，这个基本组件在页面中经常出现，通过样式类的形式进行定制，这样能够保证在一个页面中多次重复引用。

```
.nav {
   margin-bottom: 20px;
   margin-left: 0;
   list-style: none;
}
.nav > li > a {display: block;}
.nav > li > a: hover,
.nav > li > a: focus {
   text-decoration: none;
   background-color: #eeeeee;
}
```

```
.nav > li > a > img { max-width: none;}
.nav > .pull-right { float: right;}
```

4. 特定样式类

关注 CSS 中一些特定样式进行类化，如浮动、显隐、可见性、定位等。

```
.pull-right { float: right; }
.pull-left { float: left; }
.hide { display: none; }
.show { display: block; }
.invisible { visibility: hidden; }
.affix { position: fixed; }
```

10.2.3 继承和可扩展性

CSS 的继承性体现在结构关系上，且与属性本身存在很大联系，这与编程语言中的继承性有很大的不同。CSS 所定义的 100 多种属性中，只有一部分具有继承性，而不是全部都具有继承性。具体来讲，拥有继承性的属性包括如下几大类：

- 字体属性
- 文本属性（个别属性不支持继承）
- 表格属性（个别属性不支持继承）
- 列表属性
- 打印属性（部分属性支持继承）
- 声音属性（部分属性支持继承）
- 另外鼠标样式也具有继承性

而盒模型、布局、定位、背景、轮廓和内容等类属性都不具备继承性。

CSS 继承的结构性主要体现在内部结构会自动继承外部结构的可继承属性。因此，当希望统一整个 CSS 的字体、字号、字体颜色、行高等基本样式时，不妨在 html 和 body 元素中进行定义，然后通过继承性实现网页字体样式的统一。

```
html {
     font-size: 100%;
     -webkit-text-size-adjust: 100%;
     -ms-text-size-adjust: 100%;
}
body {
   margin: 0;
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    font-size: 14px;
    line-height: 20px;
    color: #333333;
    background-color: #ffffff;
}
```

如果通过继承性实现网页样式的统一，同时又希望设计个性化组件样式，可以有两种选

择方法：

- ❑ 为对象或组件单独定义样式；
- ❑ 通过专用类进行设计。

例如，在超链接样式中，页面统一为深灰色、下划线效果，但是在标签组件中通过重定义，让其文本呈现高亮显示，则主要代码如下。

```
a {
    color: #0088cc;
    text-decoration: none;
}
a:hover,
a:focus {
    color: #005580;
    text-decoration: underline;
}
a.label:hover,
a.label:focus,
a.badge:hover,
a.badge:focus {
    color: #ffffff;
    text-decoration: none;
    cursor: pointer;
}
```


由于 CSS 继承性，所有超链接的字体显示同样的效果，但是由于标签样式的不同，所设置的字体颜色显示效果截然不同。

如果仅就特定组件来说，直接在组件结构中进行定义会方便许多，但是对于 CSS 样式表来说，使用专用类来弥补 CSS 继承性是最佳选择。因为在一个网站中可能会有多处引用，通过类的方式提炼这个样式，就能够达到最优化应用。

类似这样的问题还有很多，例如通用行高与个别栏目的特殊行高，网页默认字体大小与特定栏目的特殊字体大小等。

Bootstrap 在设计思路上一般遵循以下步骤。

首先，使用 CSS 继承性来统一 CSS 样式表中的基本样式。

然后，对于特定对象、组件所需要的特殊样式，可以通过重定义的方法来修正继承所带来的问题。

最后，如果某种特殊样式使用比较普遍，通过定义特殊类的方式实现修正；如果这种特殊样式使用不是很普遍，则直接合并到组件样式中，避免类的泛滥。

CSS 是与 HTML 紧密相关的，CSS 的每一个样式都与 HTML 元素或者对象相对应，而 HTML 可以调用多个样式类。一个 CSS 样式类可以根据 HTML 代码来进行复合定义。一个 HTML 标签也可以复合调用多个样式类。因此，CSS 样式定义的复杂性与关联的 HTML 是密不可分的。

这种复杂性给 CSS 样式的应用带来很大挑战，如何在复杂的 HTML 结构中提炼出更精确的 CSS 包含样式呢?

在网站开发中，灵活使用 CSS 继承性和包含关系，可以设计出更具灵活性的样式。在设计多层包含关系的选择符时，一定要注意 CSS 优先级，当样式发生层叠时，要保证 CSS 样式在解析时不至于出现各种异常效果。

在包含选择符中，空格作为 HTML 结构的路径，明确表明网页结构的父子级别关系。而没有空格（如 div.hide）则表示同一级路径，所以没有空格表示 HTML 类自身所代表的元素，不过它要比单独指定元素的名称的优先级要高。很多时候，用户可以利用这种关系来提高元素的优先级。

10.3 Bootstrap 框架解析

10.2 节针对 Bootstrap 的 CSS 框架设计思路进行梳理，本节将重点解析 Bootstrap 插件的脚本。Bootstrap 插件的编写规范很值得我们来学习，尤其是 OOP 思想和其他设计模式在插件开发过程中的体现，易维护性和可扩展性等也是我们应该考虑的因素，当然个中细节还需要读者慢慢体会。

Bootstrap 内置插件也是很松散的，每个插件都是一段独立的代码和一个封闭的作用域，互不干扰。从结构上分析，Bootstrap 与 jQuery 插件的代码结构截然不同。当然，这种松散的插件代码设计思路，并非表明 Bootstrap 在设计插件时很随意，Bootstrap 所有插件都必须遵循相同的设计原则和规范。下面结合按钮插件的源代码进行分析。虽然 bootstrap-button.js 是 Bootstrap 中最简单的插件了，但是麻雀虽小，五脏俱全，它反映了 Bootstrap 插件编写的一些基本规范，笔者希望能抛砖引玉，让读者举一反三。

10.3.1 源码结构

bootstrap-button.js 插件基于 jQuery 的扩展，是为 HTML 原生的 button 按钮扩展一些简单功能的插件，读者应该比较熟悉按钮插件，只要向 <button> 标签添加一些额外的 data 属性，就能实现当单击按钮时出现 loading 文字以及模拟复选和单选等功能。

bootstrap-button.js 源码的主体结构如下，具体功能代码请参阅 bootstrap-button.js 源文件。

```
!function ($) {
    "use strict"; // jshint ;_;
/* BUTTON PUBLIC CLASS DEFINITION
 * ============================== */
 var Button = function (element, options) {/*some code*/}
 Button.prototype.setState = function (state) {/*some code*/}
 Button.prototype.toggle = function () {/*some code*/}
 /* BUTTON PLUGIN DEFINITION
 * ======================== */
var old = $.fn.button
$.fn.button = function (option) {return this.each(function () {/*some code*/})}
$.fn.button.defaults = {loadingText: 'loading...'}
$.fn.button.Constructor = Button
```

```
    /* BUTTON NO CONFLICT
     * ================== */
     $.fn.button.noConflict = function () {$.fn.button = old;return this;}
    /* BUTTON DATA-API
     * =============== */
   $(document).on('click.button.data-api', '[data-toggle^=button]', function (e)
    {/*some code*/})
}(window.jQuery);
```

与 jQuery 插件一样，Bootstrap 定义一个匿名函数，并将 jQuery 作为函数参数传递进来执行。这样就可以在闭包中定义私有函数而不破坏全局的命名空间，而把 JavaScript 插件写在一个相对封闭的空间，并开放可以增加扩展的地方，将不可以修改的地方定义成私有成员属性或方法，以遵循“开闭原则”。

```
!function($){
     // some code
}(window.jQuery)
```

其中，!function(){}() 是匿名函数的一种写法，它与（function(){}）() 的写法区别不大，类似的还有 +function(){}()、-function(){}()、~function(){}() 等，只是返回值不同而已。

匿名函数内的代码构成包括以下四部分。

- PUBLIC CLASS DEFINITION：类定义，定义了插件构造方法类及方法。
- PLUGIN DEFINITION：插件定义，上面只是定义了插件的类，这里才是实现插件的地方。
- PLUGIN NOCONFLICT：插件命名冲突解决。
- DATA API：数据接口。

10.3.2　类定义

插件类通过下面的构造方法来实现：

```
var Button = function (element, options) {
      this.$element = $(element)
      this.options = $.extend({}, $.fn.button.defaults, options)
}
```

这种设计方法体现了 JavaScript 的 OOP 思想，定义一个类的构造方法，然后再定义类的方法（属性），这样 new 出来的对象（类的具体实现）就可以调用类的公共方法和访问类的公共属性了。这里在 Button 函数体内部定义的属性和方法可以看做是类的私有属性和方法，为 Button.prototype 对象定义的属性和方法都可以看做是类的公共属性和方法。这个类封装了插件对象初始化所需的方法和属性。

这样通过下面的方法就可以定义了一个 Button 类型的 btn 对象，这里的 this 就是 btn 对象本身。

```
var btn=new Button(element, options);
```

Button（element，options）方法接受两个参数：element 和 options。

- element：与插件相关联的 DOM 元素，通过代码 this.$element = $(element)，将 element 封装成为一个 jQuery 对象 $element，并由 this(btn) 对象的 $element 属性引用。
- options：插件的一些设置选项（参数配置对象），这里简单说一下代码：

```
$.extend(target [, object1] [, objectN])
```

这个是 jQuery 工具函数，它的作用是将 object1，...objectN 对象合并到 target 对象中，这是编写 jQuery 插件中经常用到的方法，通过代码

```
this.options = $.extend({}, $.fn.button.defaults, options)
```

就可以实现用用户自定义的 options 覆盖插件的默认 options：$.fn.button.defaults，合并到一个空对象 {} 中，并由 this（btn）对象的 options 属性引用。

通过构造方法，btn 的方法 setState、toggle 就可以调用 btn 的 $element 和 options 属性了。

下面再来分析类的方法定义。

1. setState 方法

```
Button.prototype.setState = function (state) {
    var d = 'disabled'
       , $el = this.$element
       , data = $el.data()
       , val = $el.is('input') ? 'val' : 'html'
       state = state + 'Text'
  data.resetText || $el.data('resetText', $el[val]())
  $el[val](data[state] || this.options[state])
  // push to event loop to allow forms to submit
  setTimeout(function () {
    state == 'loadingText' ?
       $el.addClass(d).attr(d, d) :
       $el.removeClass(d).removeAttr(d)
  }, 0)
}
```

setState（state）方法的作用是为 $element 添加 'loading...'，loading... 是 $.fn.button.defaults 属性 loadingText 默认设置。这里简单说几点。

```
val = $el.is('input') ? 'val' : 'html'
```

上面的代码是为了兼容 <button>Submit</button> 和 <input type="button" value="submit"> 两种写法。

```
data.resetText || $el.data('resetText', $el[val]())
```

上面也是一个小技巧，|| 是逻辑或，意即 || 左边的表达式为 true 则不执行 || 右边的表达式，为 false 则执行 || 右边的表达式，等价于

```
if(!data.resetText){
      $el.data('resetText', $el[val]());
}
```

2. toggle 方法

```
Button.prototype.toggle = function () {
    var $parent = this.$element.closest('[data-toggle="buttons-radio"]')
    $parent && $parent.find('.active').removeClass('active')
    this.$element.toggleClass('active')
}
```

toggle() 方法的作用是通过为 button 添加 'active' 的 class 来添加“已选中”的 CSS 样式。其中代码

```
$parent && $parent.find('.active').removeClass('active')
```

与前面逻辑或的例子相似，&& 是逻辑与，意即 && 左边的表达式为 false 则不执行 && 右边的表达式，为 true 则执行 && 右边的表达式，等价于

```
if($parent){
    $parent.find('.active').removeClass('active');
}
```

定义好插件的类只是完成了对插件的抽象，即使用属性和方法来描述这个插件，但是尚未完成插件的具体实现，所以还要通过定义 jQuery 级插件对象来实现。

10.3.3　插件定义

1. 插件的 jQuery 对象级定义

插件的 jQuery 对象级定义代码如下：

```
$.fn.button = function (option) {
    return this.each(function () {
        var $this = $(this)
        , data = $this.data('button')
        , options = typeof option == 'object' && option
        if (!data) $this.data('button', (data = new Button(this, options)))
        if (option == 'toggle') data.toggle()
        else if (option) data.setState(option)
    })
}
```

首先，$.fn.button=function(){} 是在 $.fn 对象（插件的命名空间）下添加了 button 属性，这样在使用时就可以通过 $（selector）.button() 来调用插件了。

为什么在 $.fn 中添加方法，$（selector）就能直接调用该方法了呢?

实际上，读者在分析 jQuery 源码时，会发现这样的架构：

```
var jQuery = function( selector, context ) {
    // The jQuery object is actually just the init constructor 'enhanced'
    return new jQuery.fn.init( selector, context, rootjQuery );
},
```

```
// some code
jQuery.fn = jQuery.prototype = {/*some code*/}
jQuery.fn.init.prototype = jQuery.fn;
```

每次写 $（selector）实际上就是调用了 jQuery（selector）函数一次（$ 是 jQuery 的别名），都会返回一个 jQuery.fn.init 类型的对象，每写一次 $（selector）都会生成一个不同的 jQuery 对象。

jQuery 实例上是一个类（构造方法），jQuery.fn.init 也是一个类（构造方法）。jQuery.fn 正是 jQuery.prototype，jQuery.fn.init.prototype 也正是 jQuery.fn，所以添加到 jQuery.fn 的方法相当于被添加到了 jQuery.fn.init 类下面。$（selector）实质上是一个 new 的 jQuery.fn.init 类型的对象，理所当然地也就可以调用 jQuery.fn.init.prototype 下的方法，也就是 jQuery.fn 下的方法。

看起来似乎很绕，感觉只需要一个 new 就可以实现的对象为什么绕了这么大个圈子？其实一点都不绕，这里绕了这么多，主要是为了避免 jQuery 内部逻辑混乱，实现上下不同层级作用域能够相互访问，并不会相互干扰。

下面这段代码也需要注意：

```
return this.each(function () {
    var $this = $(this)
    // some code
})
```

通过 jQuery.each 方法遍历 $（selector）的所有 DOM 元素，然后再通过 $（this）将每个遍历到的 DOM 元素封装为单一的 jQuery 对象，其作用是：对于 $（selector）得到的结果集，通过形如 $（selector）.attr（'class'）方法得到的是单个结果（第一个匹配的 DOM 元素的 class 属性），而不是一组结果，通过 $（selector）.attr（'class'，'active'）更会将全部的 class 设置为 active 类。所以需要将 $（selector）的结果集逐一封装成 $ 对象再去 get 或者 set 属性，这样才是严谨的做法。

```
if (!data) $this.data('button', (data = new Button(this, options)))
```

这里是真正用到 data=new Button（this，options）；的地方，整个 $.fn.button 做的最主要的事情就是将每个匹配的 DOM 元素的 data-button 属性引用 new Button（this，options）对象。其次，通过判断 option 来调用 toggle 方法，还是调用 setState 方法。至此，整个插件才算是基本实现。

2. 插件的默认设置定义

```
$.fn.button.defaults = {
    loadingText: 'loading...'
}
```

将插件的默认设置设计为 $.fn.button 的 defaults 属性，其优点就是给用户修改插件的一些默认设置提供了通道，用户只需设置 $.fn.button.defaults={/*some code*/} 就改变了插件的

默认配置。也就是说，插件对扩展是开放的。

3. 插件的构造器

```
$.fn.button.Constructor = Button
```

开放了插件的构造方法类作为 $.fn.button 的 Constructor 属性，使得用户可以读取插件的构造方法类。

10.3.4　命名冲突解决

解决的方法与 jQuery 相同，用法类似于 $.noConflict，释放 $.fn.button 的控制权，并重新为 $.fn.button 声明一个名称，旨在解决插件名称和其他插件有冲突的情况。详细代码如下：

```
$.fn.button.noConflict = function () {
    $.fn.button = old
    return this
}
```

10.3.5　数据接口

Bootstrap 的所有插件都提供了 Data 接口，通过这种方法可以通过 HTML 标签属性来激活插件的脚本行为，避免用户编写 JavaScript 代码。

在 Bootstrap 按钮插件中，我们可以看到，通过此方法向所有带有 data-toggle 的 <button> 标签绑定 click 事件，注意这里用了事件委托的写法。

```
<button data-toggle="button">Click Me</button>
```

在插件源码中，我们可以看到下面的代码：

```
$(document).on('click.button.data-api', '[data-toggle^=button]', function (e) {
    var $btn = $(e.target)
    if (!$btn.hasClass('btn')) $btn = $btn.closest('.btn')
    $btn.button('toggle')
})
```

上面的代码通过委托的方式为页面中所有定义了 data-toggle=button 的属性定义事件，这样就不用再写 $(selector).button(options) 来初始化插件，只要页面加载，插件就自动完成初始化了。甚至 options 的某些属性都可以写在 data- 属性中，但是 $.fn.button.defaults 设置的默认属性可能会无效。

10.4　Bootstrap 内核解疑

Bootstrap 内置 13 个常用插件，同时允许开发人员自定义扩展 Bootstrap 插件。实际上，Bootstrap 插件是 jQuery 插件的一种特殊形式，在保留 jQuery 大部分语法特色基础上，进行了优化和扩展。

10.4.1 封装形式

Bootstrap 插件的封装形式与 jQuery 插件封装结构基本相同。Bootstrap 插件的封装结构如下：

```
!function ($) {
    // code
}(window.jQuery)
```

当页面初始化完成之后，上面的代码将被立即执行，并构建一个独立的作用域，这样能够有效保护内部私有变量，避免与外部变量发生冲突，同时也具有代码优化作用。该封装形式中，$ 与 window.jQuery 引用了同一个地址，都指向 jQuery 构造器函数，但是 $ 是在该匿名函数的活动对象内，而 window.jQuery 是在全局作用域中的静态对象。

不过，与 jQuery 插件结构不同的是：Bootstrap 插件在匿名函数前补加一个！运算符。那么在 function 之前加上感叹号（！）有什么意义呢?

例如，执行下面一行代码，在控制台运行后返回值为 true。为什么是 true 这很容易理解，因为这个匿名函数没有返回值，默认返回值是 undefined，求反的结果自然就是 true。

```
!function(){alert(1)}()          // true
```

当然，问题并不在于结果值，而是在于为什么求反操作能够让一个匿名函数的自动调用变得合法。

在一般开发中，常用添加小括号来调用匿名函数：

```
(function(){alert(1)})()
```

或

```
(function(){alert(1)}())
```

上述两者虽然括号的位置不同，但是效果完全一样。

无论是小括号还是感叹号，让整个语句合法做的事情只有一件，就是让一个函数声明语句变成了一个表达式。

```
function a(){alert(1)}
```

上面这行代码是一个函数声明，如果在声明后直接加上括号调用，解析器自然不会理解而报错：

```
function a(){alert(1)}();                    // SyntaxError: unexpected_token
```

因为这样的代码混淆了函数声明和函数调用，以这种方式声明的函数 f，就应该以 f()；的形式调用。

但是，小括号则不同，它将一个函数声明转化成了一个表达式，解析器不再以函数声明的方式处理函数 f，而是作为一个函数表达式处理，也因此只有在程序执行到函数 f 时，它才能被访问。

所以，任何消除函数声明和函数表达式间歧义的方法，都可以被解析器正确识别。例如：

```
var i = function(){return 1}();                    // undefined
1 && function(){return true}();                    // true
1, function(){alert(1)}();                         // undefined
```

赋值、逻辑，甚至是逗号，各种操作符都可以告诉解析器，这个不是函数声明，而是函数表达式。并且，对函数的一元运算可以算得上是消除歧义最快的方式，感叹号只是其中之一，如果不在乎返回值，下面的一元运算都是有效的：

```
!function(){alert(1)}() ;                          // true
+function(){alert(1)}() ;                          // NaN
-function(){alert(1)}() ;                          // NaN
~function(){alert(1)}();                           // -1
```

甚至下面这些关键字，都能很好的工作：

```
void function(){alert(1)}() ;                      // undefined
new function(){alert(1)}() ;                       // Object
delete function(){alert(1)}() ;                    // true
```

最后，小括号做的事情也是一样的，消除歧义才是它真正的工作，而不是把函数作为一个整体，所以无论小括号括在声明上还是把整个函数都括在里面，都是合法的：

```
(function(){alert(1)})();                          // undefined
(function(){alert(1)}()) ;                         // undefined
```

最后分析一下上面不同表达式的执行性能。在 http://jsperf.com/ 上建立一个测试，可以用不同浏览器访问，运行测试查看结果。

通过测试可以发现：new 方法永远最慢，其他表达式的差距不大，但有一点可以肯定的是，感叹号并非最理想的选择。而小括号在测试里表现始终很快，在大多数情况下比感叹号更快，所以它是最优的。加减号在 C hrome 浏览器中表现惊人，而且在其他浏览器下也普遍很快，相比感叹号效果更好。

但是 Bootstrap 插件为什么选择感叹号呢？这可能只是一种习惯问题，它们之间的优劣完全可以忽略。一旦习惯了一种代码风格，那么这种约定会使得程序从混乱变得可读。如果习惯了感叹号，它比括号有更好的可读性。用户不用在阅读时留意括号的匹配，也不用在编写时留心不要遗忘括号。

10.4.2　启用严格模式

在 Bootstrap 插件的第一行代码中都会出现这样一句代码：

```
"use strict";
```

它表示什么含义，在插件中起到什么作用呢？

一直以来，JavaScript 松散灵活的语法饱受争议。在 ECMAScript 5.0 版本中开始引入严

格模式（strict mode），使 JavaScript 解释器可以采用严格的语法来解析代码，以帮助开发人员发现常见和不易发现的错误。Firefox 是最早支持严格模式的浏览器，IE6、IE7、IE8、IE9 均不支持严格模式，IE10 开始支持严格模式，其他最新版本的主流浏览器都支持严格模式。

1. 认识严格模式

严格模式为 JavaScript 引入了很多变化，主要包括两类：明显的和细微的。细微改进的目标是修复当前 JavaScript 中的一些细节问题，下面主要介绍严格模式引入的明显变化。

（1）去除 with 关键词

严格模式中去除了 with 语句，包含 with 语句的代码在严格模式中会抛出异常。所以使用严格模式的第一步是，确保代码中没有使用 with。

```
//在严格模式中以下 JavaScript 代码会抛出错误
with (location) {
     alert(href);
}
```

（2）防止意外为全局变量赋值

局部变量在赋值前必须先声明。在启用严格模式之前，为一个未声明的局部变量复制时会自动创建一个同名全局变量。这是 JavaScript 程序中最容易出现的错误之一，在严格模式中尝试这样做时会有显性的异常抛出。

```
// 严格模式下会抛出异常
(function() {
     someUndeclaredVar = "foo";
}());
```

（3）函数中的 this 不再默认指向全局

严格模式中另一个重要的变化是函数中未被定义或为空（null or undefined）的 this 不再默认指向全局环境（global）。这会造成一些依赖函数中默认 this 行为的代码执行出错。例如：

```
window.color = "red";
function sayColor() {
     alert(this.color);
}
//在 strict 模式中会报错，如果不在严格模式中则提示 "red"
sayColor();
//在 strict 模式中会报错，如果不在严格模式中则提示 "red"
sayColor.call(null);
```

this 在被赋值之前会一直保持为 undefined，这意味着当一个构造函数在执行时，如果之前没有明确的 new 关键词，将会抛出异常。

```
function Person(name) {
     this.name = name;

}
//在严格模式中会报错
var me = Person("Nicholas");
```

在上面的代码中，Person 构造函数在运行时因为之前没有使用 new 调用，函数中的 this 会保留为 undefined，而由于不能为 undefined 设置属性，上面的代码会抛出错误。在非严格模式环境中，没有被复制的 this 会默认指向 window 全局变量，运行的结果将是意外地为 window 全局变量设置 name 属性。

（4）防止重名

当编写大量代码时，对象属性和函数参数很容易不小心被设置成一个重复的名字。严格模式在这种情况下会显性地抛出错误。

```
//重复的参量名，在严格模式下会报错
function doSomething(value1, value2, value1) {
    //code
}
//重复的对象属性名，在严格模式下会报错
var object = {
    foo: "bar",
    foo: "baz"
};
```

以上代码在严格模式中都会被认为有语法错误而在执行前都能得到提示。

（5）安全的 eval

虽然 eval 语句最终没有被移除，但在严格模式中仍然对它进行了一些改进。最大的改变是在 eval() 中执行的变量和函数声明不会直接在当前作用域中创建相应变量或函数。例如：

```
(function() {
    eval("var x = 10;");
    // 非严格模式中 ,alert 10
    // 严格模式中则因 x 未被定义而抛出异常 ,
    alert(x);
}());
```

任何在 eval() 执行过程中创建的变量或者函数保留在 eval() 中，但能明确地从 eval() 语句的返回值来获取 eval() 中的执行结果，例如：

```
(function() {
    var result = eval("var x = 10, y = 20; x + y");
    // 在严格或非严格模式中都能正确的运行余下的语句 (result 为 30)
    alert(result);
}());
```

（6）对只读属性修改时抛出异常

ECMAScript 5 中还引入为对象的特定属性设为只读，或让整个对象不可修改的能力。但在非严格模式中，尝试修改一个只读属性只会失败，但不会报错。在一些浏览器原生 API 中，会遇到这种情况。严格模式会在这种情况下明确的抛出异常，提醒修改这个属性是不被允许的。

```
var person = {};
Object.defineProperty(person, "name" {
```

```
        writable: false,
        value: "Nicholas"
    });
    // 在非严格模式时，沉默的失败，在严格模式则抛出异常
    person.name = "John";
```

在上面的示例中，name 属性被设为只读，非严格模式中执行对 name 属性的修改不会引发报错，但修改不会成功，但严格模式则会明确抛出异常。

此外，在代码中不能使用一些扩展的保留字：

implements、interface、let、package、private、public、static、yield

不能声明或重写 eval 和 arguments 两个标识符。不能使用 delete 删除显式声明的标识符，名称或具名函数。

2. 启用严格模式

开启严格模式很简单，只要使用字符串序列，在代码的开头加入下面这行代码，则其中运行的所有代码都必然是严格模式下的。

```
"use strict";
```

虽然看上去上面的代码仅仅只是未赋予某个变量的字符串，但它实际上起到指示 JavaScript 引擎切换到严格模式的作用，不支持严格模式的浏览器会忽略以上代码，不会对后续的执行产生任何影响，这样很巧妙地兼容了那些不支持严格模式的浏览器，也不会报错。如果在语法检测时发现语法问题，则整个代码块失效，并导致一个语法异常；如果在运行期出现了违反严格模式的代码，则抛出执行异常。

加入严格模式字符串序列的位置有如下几种情况，同时可以开启相应代码块中的严格模式：

- 在全局代码的开始处加入。
- 在 eval 代码开始处加入。
- 在函数声明代码开始处加入。
- 在 new Function() 所传入的 body 参数块开始加入。

虽然可以把这个指令作用到全局或某个函数中，建议用户不要在全局环境下启用严格模式。

例如：

```
// 请不要这么使用
"use strict";
function doSomething() {
    // 这部分代码会运行于严格模式
}
function doSomethingElse() {
    // 这部分代码也会运行于严格模式
}
```

虽然上面的代码看起来没有问题，但当在页面中引入大量的外部脚本文件时，这样使用严格模式会让页面面临由于第三方代码没有遵循严格模式，而引发的大量问题。

因此，最好把开启严格模式的指令作用于函数中。例如：

```
function doSomething() {
      "use strict";
      //这个函数中的代码将会运行于严格模式
}
function doSomethingElse() {
      //这个函数中代码不会运行于严格模式
}
```

如果希望严格模式在多个函数中开启，可以使用立即执行函数表达式：

```
(function() {
      "use strict";
      function doSomething() {
            //这个函数运行于严格模式
      }
      function doSomethingElse() {
            //这个函数同样运行于严格模式
      }
}());
```

Bootstrap 插件都采用这种方式启用严格模式，只在当前插件的代码作用域中执行严格模式，不会干扰页面其他代码模式。

10.4.3　插件中的 this

每当 JavaScript 函数被创建时，一个叫做 this 的关键字也随之被创建，它指向操作当前函数的对象。因此 this 仅在函数作用域里是可用的，并指向包含这个函数的对象，而不是这个函数本身。在特殊情况下，使用 new 运算符创建实例，或者使用 call 或 apply 调用函数，会主动改变 this 所指向的对象。

注意，与 argument 和函数参数不同，this 是一个关键字，而不是属性。this 的行为类似于变量，但是用户是无法随意赋值的。

由于晚绑定的特性，JavaScript 中的 this 总是让人迷惑，它可以是全局对象、当前对象或者特定对象，如何确定 this 的值？

确定 this 的值，只能够基于函数被调用时的运行上下文。因此，读者需要明白 this 的值会随着调用函数上下文的不同而改变。

注意，除了 this 以外的所有变量以及参数都会遵循词法作用域，但是嵌套函数里的 this 指向全局对象。

下面简单介绍一下 this 指代的对象。

（1）全局代码中的 this

Window 全局范围内的 this 将会指向全局对象，在浏览器中表示 window 对象。如下面代码将指代 window 对象。

```
alert(this)
```

（2）作为单纯的函数调用

直接调用函数时，不管调用位置（函数内或者函数外），this 都指向全局对象 window。例如：

```
function foo(x) {
     this.x = x;
}
foo(2);
alert(x);  // 2，全局变量 x 的值
```

但是在严格模式下，这种用法是不允许的，返回值将是 undefined。

（3）作为对象的方法调用

当函数作为对象的方法被调用时，函数内的 this 指向当前调用对象。例如：

```
var n = "true";
var p = {
     n: "false",
     m: function(){
          console.log(this.n);
     }
}
p.m();//"false"
```

在上面的代码中，this 指向 p 对象，即当前对象，而不是 window。

（4）作为构造函数

在构造函数中，this 指向新创建的实例对象。

```
function F(x) {
     this.x = x;
}
var f = new F(2);
```

函数内部的 this 指向新创建的对象 f。

（5）内部函数

在内部函数中，this 没有按预期绑定到外层函数对象上，而是绑定到了全局对象上。这是 JavaScript 语言的设计缺陷，因为没有人想让内部函数中的 this 指向全局对象。

```
var n = "true";
var p = {
     n: "false",
     m: function(){
          var say = function() {
               console.log(this.n);
          };
          say();
     }
}
p.m(); // "true"
```

一般处理方式是将 this 作为变量保存下来，默认约定为 that 或者 self：

```
var n = "true";
var p = {
     n: "false",
     m: function(){
          var that = this;
          var say = function() {
               console.log(that.name);
          };
          say();
     }
}
p.m(); //"false"
```

（6）使用 call 和 apply 设置 this

call 和 apply 都能够将被调用函数绑定到指定对象上，自然此时的 this 会被显式地设置为第一个参数。例如，针对上面的代码，可以通过下面的方法实现把 m() 方法中的 this 绑定到 p 对象上。

```
p.m.call(p);
```

apply 和 call 类似，只是第二个参数是通过一个数组传入，而不是分开传入。

在 Bootstrap 插件中，也包含两种不同对象的 this 应用。例如，针对按钮插件来说，在插件函数第 2 行代码中，这里的 this 指向 jQuery 对象，调用 jQuery 对象的 each() 方法时，传递给它的参数函数中也包含一个 this 关键字，这个关键字指向 jQuery 对象中每个 DOM 元素。

```
$.fn.button = function (option) {
     return this.each(function () {
          var $this = $(this)
          , data = $this.data('button')
          , options = typeof option == 'object' && option
          if (!data) $this.data('button', (data = new Button(this, options)))
          if (option == 'toggle') data.toggle()
          else if (option) data.setState(option)
     })
}
```