HZ BOOKS 华章科技

[PACKT] PUBLISHING

Web开发技术丛书

响应式Web设计

HTML5和CSS3实践指南

[美] Benjamin LaGrone 著　　黄博文 饶勋荣 译

HTML5 and CSS3 Responsive Web Design Cookbook

- 资深Web开发工程师亲笔撰写，Amazon广泛好评，完全掌握下一代设备和浏览器技术的必备指南
- 从准备工作、实现方式和工作原理三方面，通过大量真实示例，全面而系统地讲解HTML5和CSS3响应式Web设计与开发的各种实用技术和技巧

机械工业出版社
China Machine Press

针对最新的设备和移动Web，我们如何改进和优化网站设计来更好地呈现网站内容？本书详细讲解了响应式Web设计的具体操作方法、各种实用技术和技巧。书中首先介绍如何使图片自适应页面尺寸、制作响应式菜单、嵌入响应式视频，以及利用大量响应式字体技术，接下来讲述如何使用框架创建布局，创建迷人的响应式网站，以及仅使用几行CSS代码来改进现有框架以应用于响应式网站中。本书是完全掌握下一代设备和浏览器技术的必备指南。

通过阅读本书，你将学到：

- 响应独特显示终端的响应式布局的制作原则
- 制作响应式字体，使其在所有设备上都容易阅读——在小型移动设备屏幕上轻松阅读文本
- 使用jQuery Mobile库和移动设备优先的设计理念来创建移动站点
- 发现如何获取创建、部署、测试响应式网站的一系列工具
- 学习服务器端及客户端媒介部署技术，提供可适应于任何媒介设备平台
- 利用可直接应用到现有项目中的设计和代码

[PACKT] PUBLISHING

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

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

上架指导：程序设计/Web开发

ISBN 978-7-111-47321-3

定价：39.00元

Web开发
技术丛书

响应式Web设计

HTML5和CSS3实践指南

HTML5 and CSS3 Responsive
Web Design Cookbook

[美] Benjamin LaGrone 著　黄博文 饶勋荣 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

响应式 Web 设计：HTML5 和 CSS3 实践指南 /（美）拉格罗（LaGrone, B.）著；黄博文，饶勋荣译 . —北京：机械工业出版社，2014.7
（Web 开发技术丛书）
书名原文：HTML5 and CSS3 Responsive Web Design Cookbook

ISBN 978-7-111-47321-3

I. 响… II. ① 拉… ② 黄… ③ 饶… III. ① 超文本标记语言 – 程序设计 – 指南 ② 网页制作工具 – 指南 IV. ① TP312-62 ② TP393.092-62

中国版本图书馆 CIP 数据核字（2014）第 153702 号

本书版权登记号：图字：01-2014-1837

响应式 Web 设计：HTML5 和 CSS3 实践指南

[美] Benjamin LaGrone 著

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：谢晓芳
责任校对：殷 虹
印 刷：三河市宏图印务有限公司
版 次：2014 年 8 月第 1 版第 1 次印刷
开 本：147mm × 210mm 1/32
印 张：7
书 号：ISBN 978-7-111-47321-3
定 价：39.00 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

译者序

“不是我不明白，这世界变化快。”崔健这首歌中的歌词使用在互联网领域最合适不过。只短短数年的工夫，互联网的浪潮还没过去，移动互联网的时代已经来临。人们已经习惯将越来越多的时间花在各种移动设备上。各大互联网公司先知先觉，在移动互联网领域杀得不可开交，甚至很多传统行业公司也在积极寻求自身领域与移动互联网的结合点。

终端设备种类繁多，要给所有用户群带来一致的用户体验实属不易。在这种背景下，响应式设计应运而生。响应式 Web 设计的理念是，页面的设计与开发应当根据用户行为及设备环境进行相应的响应和调整。响应式设计并不是单纯设计者的事情，它是一系列技术栈的结合。HTML5 和 CSS3 酝酿了多年终于落地，其在响应式设计中扮演着举足轻重的角色。

本书作者 Benjamin LaGrone 具有丰富的互联网设计和开发经验。本书共 7 章，系统地介绍了利用 HTML5 和 CSS3 进行响应式 Web 设计的方方面面。每一节基本都分为准备工作、实现方式、工作原理三部分，每个知识点逐步展开。并且每节所对应的方法都有详细的示例代码，可供读者参考学习。

在翻译此书的过程中，我和同事饶勋荣合作非常愉快。同时也感谢机械工业出版社的编辑对我们工作的支持。另外，还要感谢家人对我的包容和照顾。

最后，希望本书能给大家带来超凡的阅读体验。

黄博文

作者简介

Benjamin LaGrone 是一个工作和生活在得克萨斯州的 Web 开发工程师。6 岁时就在休斯顿自然科学博物馆第一次接触计算机课程，并由此开始编程之旅。他的第一个程序是“选择自己的冒险书籍”，所使用的语言是 BASIC。他至今依旧怀念编写程序需要手动添加行号的美好时光。

直到大约 30 年后，Ben 才决定将计算机作为自己的职业。由此开始，Ben 的职业生涯中包含了其个人最感兴趣的两件事：艺术和编程，即从代码中创造艺术。他最喜爱的一个项目是使用 GMaps API 实现病理学与染色体的映射，从而用于癌症研究。

Ben 长时间着迷于移动设备，他认为响应式网站设计是 Web 开发的一个新领域，同时也是最令人兴奋的特性之一，并且会持续很长一段时间。他现在工作在 SAAS，并在开发团队中担任移动响应式设计布道者。

除了将时间花费在互联网项目上，Ben 还利用业余时间建造机器人、摆弄机器、喝咖啡、冲浪，以及指导韩国国术 (Kuk Sool)。

本书的完成离不开我挚爱的妻子，还有我两个漂亮的女儿 Daphne 和 Darby 的耐心支持。谢谢你们。

审校者简介

Dale Cruse 是《HTML5 Multimedia Development》一书的作者，并且还担任过其他几本 HTML5 书籍的技术编辑。他的职业生涯开始于 1995 年，当时是美国陆军摄影记者。自从打算将 CBSNews.com 纯数字化，他为世界上一些最为知名的公司创建了网页版和移动版的用户体验平台，这其中就包括二十世纪福克斯影片公司（20th Century Fox）、布鲁明戴尔百货店（Bloomingdale's）及 MINI Cooper。现在他既是一个工作于 Allen &Gerritsen 的资深前端开发工程师，也是处于南波士顿的纽约洋基队的忠实球迷。他是个很受欢迎的演讲家，你可以关注他的推特 @dalecruse。

Ed Henderson 出生并成长于苏格兰，是一个极富生活阅历的人，他热衷于设计、构建、实现和破坏网络事物。

他并不怕弄脏双手或弄湿双脚，只要是有用或有趣的新技术，他都一直乐于接受。

Ed 拥有计算机科学学位，自己开过公司，当过自由职业人，当过上班族，做过咨询师。现在他受雇于美国加州旧金山的 POPSUGAR 公司，是一位资深软件工程师。

他对 IT 行业的各个方面都有丰富的从业经验，无论是网站还是

社交媒体应用。Ed 也为一些书籍做过评审，同时自己也写过数本书。

Ed 经常会冒出新鲜的想法。勇于尝试，并将其中一些想法付诸实践，可以看得见的想法才是 Ed 一直追求的目标。

远离疯狂的网络世界后，Ed 参加了爱丁堡马拉松比赛并从一座灯塔速降下来，由此募集了数千英镑捐款。在家乡苏格兰的橄榄球队当了 3 个赛季的队长，作为最佳得分手赢得了冠军锦标，还带领球队闯入了苏格兰锦标赛的决赛。

你可能不知道 Ed 还是 Jack Draws Anything(http://jackdrawsanything.com/) 的创始人，并且是极具声望的 .net 杂志 2011 年年度社交冠军奖项的获得者。

Ed 和他的妻子 Rose 生活在美国加利福尼亚州的科特马德拉市（距离圣弗朗西斯科只有 15 分钟车程）。他的团队成员暨好朋友 Jack、Toby 和 Noah 也生活在这里。

Ed 喜欢蛋糕、熏肉、苹果汁，也喜欢和其他人交流对自己的看法。

Rokesh Jankie 于 1998 年毕业于荷兰的莱顿大学，拥有计算机科学硕士学位。他的专业领域是算法和 NP 完全问题，关注于使用 NP 完全问题来解决相关领域难题。他曾先后工作于莱顿大学、ORTEC（位于意大利佛罗伦萨阿诺河老桥附近）和 Qualogy。在 Qualogy 公司，他使用自己的专业知识来开发产品。Qualogy 工作于 Oracle 和 Java 技术领域。使用流行的技术来交付有趣的产品，即 QAFE（请访问 www.qafe.com 获取更多信息）。

目前他工作的公司专门从事 Oracle 和 Java 技术。作为产品开发部门的负责人及 QAFE 公司的 CTO，他重点关注于 Web 程序开发的未来。在该公司，那些最新的技术（如 HTML5、Google API、AngularJS、NodeJS 及 Java）都在使用，并且和来自 Google 的优秀工程师保持密切合作，来保证这些技术能够更好地为项目服务。

他也审阅了由 Packt 出版社出版的《 HTML5 Canvas Cookbook 》

一书，及 Manning 出版社出版的《Dart in Action》一书。

我非常荣幸能够审校这本书。Savio Jose 给了我这个机会。本书的主题是关于 Web 的下一个大事件（HTML5、CSS3 及 JavaScript），作为审校者，能以这样一种方式参与感觉非常好。Web 应用程序的未来前途无限。

前　言

本书提供了新的开发工具箱，保证开发者从中获取到最新的设计和开发技能。掌握了本书中所介绍的方法之后，你就能构建响应式应用程序，并且使得 Web 项目的移动版本能够按照最新的理念进行开发和设计也是其一大优势。本书中的示例都是真实的，通过类似于对站点改进的形式，在易于理解的描述下对照示例一步一步地实践，使得读者能够理解响应式设计的精髓，并可以完成一系列设备的响应式设计优化。本书中的主题涵盖了响应式元素和媒介、响应式字体、响应式布局，使用媒介查询，学习最新的响应式框架，开发移动设备优先的 Web 应用程序，优化响应式内容，使用 JavaScript 和 jQuery 实现无侵入式的交互。每节所对应的方法都能直接通过所提供的代码实现。

本书内容

第 1 章涵盖元素创建，用于移动设备或台式机的优化。

第 2 章介绍如何使用流式字体，创建很棒的字体效果，以及使用 HTML5 画布和 CSS3 实现字体的立体特效。

第 3 章讲述如何创建可实际应用到项目中的响应式布局。你将学会如何使用视窗和媒介查询，使得 Web 站点在不同视区大小和类型

下变成响应式。

第 4 章介绍如何使用新型框架，通过最新的响应式方法和交互方式，既快速又可靠地完成响应式站点的设计和交付，以及如何将旧的静态框架转换为响应式类型的框架。

第 5 章讨论如何实现 Web 应用程序的移动版本。该章通过 jQueryMobile 优先针对移动设备优化，并针对桌面视窗进行了优化。

第 6 章介绍如何获取用以构建和测试响应式 Web 项目的各种工具，并能够正确地使用。

第 7 章介绍如何编写独立于 Web 页面的 JavaScript，以便为不同设备实现响应式交互打下基础。

准备事项

你需要一个 IDE（集成开发环境），推荐 NetBeans 或 Eclipse（有操作指南指导如何使用）。还需要图像编辑软件，如 Photoshop 或 GIMP。另外还要有一台 Web 服务器和一台本地服务器，如 Apache 或者其他本地托管应用程序，如 XAMPP 或 MAMPP。

读者对象

本书介绍的方法适用于当今所有无线网络设备，同时能够为 Web 开发者带来所期望的更快交付的革新技术，也为最新的移动网络设备提供更加直观的交互方式。

提醒或重要的事项以这种格式呈现。

提示及小窍门以这种格式呈现。

读者反馈

我们永远欢迎来自读者的反馈。请让我们知道你对本书的想法，包括喜欢的内容或可能不怎么喜欢的部分。读者反馈对我们非常重要，有助于帮助我们今后出版优秀的图书。

可以通过发送邮件到 <feedback@packtpub.com>，给我们提出最宝贵的反馈信息，只需要在邮件的主题中提到本书即可。

如果你是本书中某一个主题方面的专家，同时也有兴趣写一些东西或贡献自己的力量，就请访问作者指南页面 www.packtpub.com/authors。

客户支持

恭喜你拥有本书。我们准备了以下内容来让你的购买价值最大化。

目　录

第 1 章
响应式元素及媒介

1.1 简介

响应式 Web 设计是自我学生时代 BBS 上出现 ASCII 字符图形至今，Web 开发领域最让人感到兴奋的事情之一。HTML5、CSS3 及 jQuery 给旧网络注入了新生命，给使用你应用程序的用户带来乐趣和兴奋。本章介绍的几个方法将会帮助你构建自己的响应式 HTML 元素及媒介。

这些方法有易有难。所涉及的响应式 Web 设计的各种元素代码均能在本书中找到，因此书中所提到的所有内容都是可以实现的。每一个响应式 Web 设计的方法都有助于优化网站展示，无论你的用户使用何种尺寸的何种设备，都会从中得到令其惊叹的响应式 Web 体验。

1.2 基于宽度百分比的图像缩放

本方法依赖于客户端编码来完成对于大图像的缩放功能。客户端只需单张图像来依据浏览器的窗口大小呈现图像。如果对于客户端的网络带宽有足够的信心，确信该操作不会使得页面加载变慢，那么本方法是比较可行的。

1.2.1 准备工作

毫无疑问我们需要一张图像。使用 Google 的图像搜索来获取一张高分辨率的图像。例如，搜索 robots，将会得到 158 000 000 条记录，还不错的结果。但是想要的是一张大尺寸的图像，因此单击 Search tools 选项，然后将 Any Size 选项改为 Large。可以看到依然有 4 960 000 张图片可供选择。

该图像应该能够缩放以适配最大尺寸的可视区域。打开图像编辑软件。如果并未安装此类软件，也会有众多的免费图像编辑

软件供你选择。Gimp 便是其中之一，它是一款功能强大的开源图片编辑软件，因此下载是完全免费的。访问 http://www.gimp.org 即可获取。

1.2.2　实现方式

通过图像编辑工具打开想要编辑的图像并且将其宽度设置为 300px。保存编辑后的新图像，然后移动或者上传到你的 Web 应用所对应的资源文件夹中。

为了展示响应式特效，需要在 HTML 中嵌入图像和一些文字说明。如果没有足够的时间来阐述你的人生经历，没有关系，回到互联网并通过 Ipsum 生成器来得到一些文本样本。访问 http://www.lipsum.com 即可获取 Ipsum 文本段落。

```
<p class="text">Loremipsum dolor sit amet…</p>
<div class="img-wrap" >
     <img alt="robots image" class="responsive" src="robots.jpg" >
     <p>Loremipsum dolor sit amet</p>
</div>
```

在 CSS 文件中需要为文本段落、图像及图像包装器（wrapper）分别设置相应的类属性。文本段落设置为向左浮动，同时宽度为 60%，图像包装器（wrapper）的宽度则为 40%。

```
p.text {
     float:left;
     width:60%;
}
div.img-wrap{
     float:right;
     width:40%;
}
```

现在的布局方式为流式布局（fluid layout），但是响应式的图像效果依然不见踪影。目前的图像仍然是静态的，宽度依旧为

300px。但是当添加了下面的 CSS 配置后，一切都会改变。为图像添加一个新的类，设置 max-width 属性值为 100%。这会使得图像的宽度适应浏览器宽度的改变。接下来，将 height 属性设置为可动态变化的值。

```
img.responsive {
    max-width: 100%;
    height: auto;
}
```

到目前为止，经过优化并可以适应浏览器窗口大小改变的图片就已经呈现在读者眼前了。

下载示例代码

登录 http://www.packtpub.com 可以下载通过你的账户所购买的所有 Packt 书籍的示例源码。如果是通过其他途径购买的本书，访问 http://www.packtpub.com/support，完成注册即可通过电子邮件获取源码。

1.2.3 工作原理

设置在 CSS 中图像元素的 responsive 属性会强制该元素 100% 占据其父元素的空间。当父元素的宽度改变时，图像元素也会相应改变并填充对应的宽度。而属性 height: auto 的作用在于保证图像自身的高宽比例不会发生变化。

1.2.4 相关章节

- 1.3 节
- 1.7 节

1.3　基于 cookie 及 JavaScript 的响应式图像

使用复杂的服务器端逻辑也能实现响应式图像。有时因为一些特殊的需求，开发者往往不能够采取最简单的方法来实现特定的目标。之前提到的基于宽度百分比的方法依赖于客户端来实现对于大图像文件的缩放。而本方法则是在服务器端依据客户的请求，返回恰当大小的图像文件。如果你正为网页加载过慢导致的网站性能问题而烦恼，也许该方法能够给不堪重负的服务器负载和网络带宽带来一些改进。

1.3.1　准备工作

本条目所涉及的方法都需要在服务器端实现相应的功能。首先，服务器端需要提供 PHP 服务。其次，服务器端需要实现根据客户端的请求返回 3 张不同大小的图像的功能。

1.3.2　实现方式

下面的 JavaScript 代码比较简单，创建一个基于用户设备屏幕尺寸的 cookie。当客户端请求服务器端的图像时，运行在服务器端的 PHP 服务便会依据该屏幕参数决定返回何种大小的图像。

```
<script >
     document.cookie = "screen_dimensions=" + screen.width + "x" +
screen.height;
</script>
```

回到服务器端，在相应的 Web 资源路径下创建名为 images 的文件夹，同时新建 PHP 文件 (名为 index.php)，以下为相应的 PHP 代码：

```
<?php
 $screen_w = 0;
 $screen_h = 0;
 $img = $_SERVER['QUERY_STRING'];
```

```
if (file_exists($img)) {

  // Get screen dimensions from the cookie
  if (isset($_COOKIE['screen_dimensions'])) {
    $screen = explode('x', $_COOKIE['screen_dimensions']);
    if (count($screen)==2) {
      $screen_w = intval($screen[0]);
      $screen_h = intval($screen[1]);
    }
  }
  if ($screen_width> 0) {

    $theExt = pathinfo($img, PATHINFO_EXTENSION);

    // for Low resolution screen
    if ($screen_width<=800 {
      $output = substr_replace($img, '-low', -strlen($theExt)-1,0);
    }

    // for Medium resolution screen
    else if ($screen_width<=1024) {
      $output = substr_replace($img, '-med', -strlen($theExt)-1, 0);
    }

    // check if file exists
    if (isset($output) &&file_exists($output)) {
      $img = $output;
    }
  }

  // return the image file;
  readfile($img);
}

?>
```

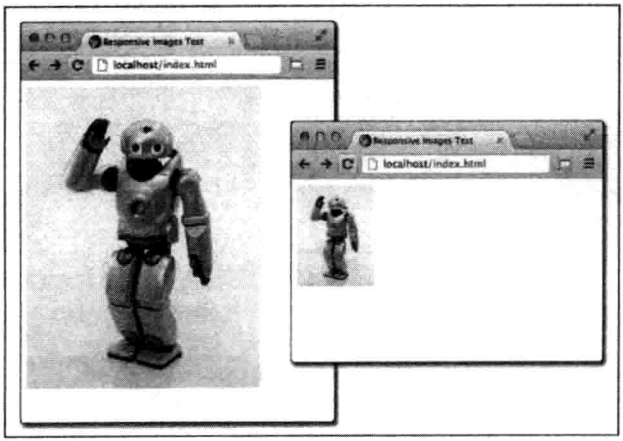
现在通过你的图像编辑软件，打开一张尺寸较大的图像，然后保存两份较小尺寸的新图像。如果原始图像为 300px，那么新图像的大小可以分别保存为 200px 和 100px。然后分别命名为 robot.png、robot-med.png 及 robot-low.png，并上传到 images 文件夹中。

最后很重要的一点，将下面的 HTML 文件存放到服务器端的文档根目录中：

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

```
        <title>Responsive Images</title>
        <meta charset="utf-8">
        <script>
   document.cookie = "screen_dimensions=" + screen.width + "x" +
screen.height;
        </script>
    </head>
    <body>
        <img alt="robot image" src="images/index.php?robot.png">
    </body>
</html>
```

可以在下面的屏幕截图中看到该方法的执行结果。



虽然该方法只能根据屏幕大小返回特定的图像，并不完全是自适应的，但是该方法在服务器端提供与 CSS 媒介查询一样的功能。可以通过 CSS 给这些图片加上样式，或者使用 JavaScript 实现动画效果。将不同的方法结合在一起使用，能够更好地提供响应式内容。

本方法中所涉及的代码最初由我聪明的同行所创建，放置在 http://www.html.it/articoli/responsive-images-con-i-cookie/ 上。

1.3.3 工作原理

首先 HTML 文件中创建 cookie 用以记录设备屏幕的尺寸。当向服务端发起获取图片的请求时，就如同 PHP 中的 include 声明一样。首先检查声明的文件是否存在，然后读取 cookie 中的屏幕尺寸，最后返回与之适配的图像资源。

1.4 使视频自适应于屏幕宽度

流媒体视频同样可以为响应式的。在 HTML5 页面中嵌入响应式视频是一件非常简单的事情。如 video 标签支持百分比的宽度设置，但是问题在于，该方案需要网站主机拥有对应的视频源。如果已经有视频源，那么一切都易如反掌。

```
<style>
video {
     max-width: 100%;
     height: auto;
}
</style>

<video width="320" height="240" controls="controls">
     <source src="movie.mp4" type="video/mp4">
     <source src="movie.ogg" type="video/ogg">
     Your browser does not support the video tag.
</video>
```

不过，使用视频托管网站（如 YouTube 或 Vimeo）比自己托管更具优势。首先，自己的托管服务器可能会受到带宽或磁盘空间的限制。其次，相比于使用自己的托管服务器，使用视频托管网站上传可用的 Web 视频的整个过程更加便捷。

1.4.1 准备工作

视频托管网站允许你在自己的页面中嵌入 iFrame 或 object 代码片段，用以在你自己的网站中展示视频。但是 iFrame 或目标

代码段不能用在 video 标签中。因此，为了实现响应式视频功能，所用的方法相对复杂一些，但是仍然不太困难。

1.4.2　实现方式

在 HTML 中将视频源代码段嵌入 div 元素中，同时设置 div 元素为相对定位，其底部内边距设定为 50% ~ 60% 。然后设定子元素，即视频 iFrame 对象为绝对定位，宽度为 100%，高度为 100%。这使得 iFrame 对象完全填充了父元素。

以下 HTML 代码展示如何通过 iFrame 标签从 Vimeo 获取视频信息：

```
<div class="video-wrap">
     <iframe src="http://player.vimeo.com/video/52948373?badge=0"
width = "800" height= "450" frameborder="0"></iframe>
</div>
```

下面的 HTML 代码展示如何使用老版本的 YouTube 标记对象：

```
<div class="video-wrap">
    <object width="800" height="450">
       <param name="movie" value="http://www.youtube.com/v/
b803LeMGkCA?version=3&amp;hl=en_US">
         </param>
         <param name="allowFullScreen" value="true"></param>
         <param name="allowscriptaccess" value="always"></param>
         <embed src="http://www.youtube.com/v/
b803LeMGkCA?version=3&amp;hl=en_US" type="application/x-shockwave-
flash" width="560" height="315" allowscriptaccess="always"
allowfullscreen="true">
          </embed>
     </object>
</div>
```

以上两段 HTML 代码均依赖于以下 CSS 配置：

```
.video-wrap {
     position:relative;
     padding-bottom: 55%;
     padding-top: 30px;
     height: 0;
     overflow:hidden;
```

```
}
.video-wrap iframe,
.video-wrap object,
.video-wrap embed {
     position:absolute;
     top:0;
     width:100%;
     height:100%;
}
```

如果不希望视频占据整个页面的宽度，可以使用 width 及 max-width 属性来限制视频宽度。接着使用另外一个 div 元素包装在 class 属性为 video-wrap 的元素的外层，并且设定 width 属性为固定值，属性 max-width 的值为 100%。

```
<div class="video-outer-wrap">
     <div class="video-wrap">
          <iframe src="http://player.vimeo.com/video/6284199?title=0&b
yline=0&portrait=0" width="800" height="450" frameborder="0">
          </iframe>
     </div>
</div>

.video-outer-wrap {
     width: 500px;
     max-width:100%;
}
```

本方法适用于目前所有的主流浏览器。

1.4.3 工作原理

该方法称为固有比率视频法（Intrinsic Ratios for Videos），由 Thierry Koblentz 在网站 A List Apart ⊖上提出。在元素中所嵌入的视频具有固有的长宽比，并且被赋予了一个绝对位置。这就使得允许视频窗口大小改变的同时，锁定视频长宽比。

⊖ http://alistapart.com/article/creating-intrinsic-ratios-for-video。——译者注

1.5　基于媒介查询的图像缩放

媒介查询是针对响应式图像的另一个有用和高度可定制的方法。这与通过设置宽度百分比来实现自适应宽度的方法并不相同。你的设计可能需要为不同的屏幕尺寸范围准备一些具体的图像宽度，而自适应宽度会打破你的设计。

1.5.1　准备工作

这种方式仅仅需要一张图像。而且在客户端浏览器调整图像，而不是在服务器端。

1.5.2　实现方式

HTML 代码相当简单，使用标准的图像标签创建一个图像元素，如下所示：

```
<img alt="robot image" src="robot.png">
```

先从一个简单的版本开始，创建一个媒介查询。该媒介查询将检测浏览器窗口的大小，如果浏览器屏幕大于 1024px，将提供一张较大的图像，为较小的浏览器窗口提供较小的图像。媒介查询首先查找媒介类型 screen，然后查找屏幕尺寸。当浏览器满足媒介查询条件时，将渲染小括号中的 CSS。

```
@media screen and ( max-width: 1024px ) {…}
@media screen and ( min-width: 1025px ) {…}
```

现在给这张图像标签添加一个类属性。该类在不同的媒介查询中的响应是不一样的，如以下代码所示：

```
<img alt="robot image" src="robot.png" class="responsive"/>
```

将 CSS 中的这个类添加到每个媒介查询中，所对应设置的尺寸都不相同。这将会使浏览器针对每个不同的尺寸窗口渲染所需

的图像大小。媒介查询可与其他 CSS 类共存。接着，添加一个独立于媒介查询的 CSS 的类属性，设置图像属性为 height:auto。这样只需添加一行 CSS 即可对两个媒介查询都起作用。

```
@media screen and ( max-width: 1024px ) {
img.responsive { width: 200px; }
}
@media screen and ( min-width: 1025px) {
img.responsive { width: 300px;}
}
img.responsive { height: auto; }
```

为了使图像能自适应于多个设备屏幕尺寸范围，可以结合 max-width 和 min-width 媒介查询。为大小介于 1024 ~ 1280px 之间的浏览器窗口指定图像尺寸，需要为 screen 添加一个 min-width 为 1024px、max-width 为 1280px 的媒介查询。

```
@media screen and ( max-width: 1024px ) {
img.responsive { width: 200px; }
}
@media screen and ( min-width:1025px ) and ( max-width: 1280px ) {
img.responsive { width: 300px; }
}
@media screen and ( min-width: 1081px ) {
img.responsive { width: 400px; }
}
img.responsive { height: auto; }
```

通过媒介查询方法便可实现针对多个不同的浏览器窗口尺寸设置对应的图像尺寸。

1.5.3 工作原理

CSS3 媒介查询在 CSS 中通过逻辑条件将浏览器依据窗口属性进行区分，并基于此来完成不同样式的渲染。该方法正是利用这一点来对不同的浏览器窗口尺寸设置一个不同的图像宽度。这提供了响应式的图像尺寸，因此你可以进行高粒度级别的控制。

1.6　基于媒介查询的动态导航栏

媒介查询不仅限于调整图像大小。可以使用媒介查询为访问者提供更加动态的网页。可以使用媒介查询显示一个基于不同屏幕尺寸的响应式菜单。

1.6.1　准备工作

为了实现一个响应式菜单系统，我们将使用两个不同的菜单，为三种不同的浏览器窗口尺寸显示一个动态菜单。

1.6.2　实现方式

对于较小的浏览器窗口，特别是对移动设备和平板电脑，创建一个简单的 select 菜单，其只占用少量的垂直空间。这个菜单为导航选项使用了 HTML 中的 form 元素，通过触发 JavaScript 代码来加载选择的新页面。

```
<div class="small-menu">
     <form>
          <select name="URL" onchange="window.location.href=this.form.
URL.options[this.form.URL.selectedIndex].value">
               <option value="blog.html">My Blog</option>
               <option value="home.html">My Home Page</option>
               <option value="tutorials.html">My Tutorials</option>
          </select>
     <form>
</div>
```

对于较大的浏览器窗口尺寸，创建一个可以通过 CSS 设置样式的简单 ul 列表元素。这个菜单对于不同的媒介查询会接收不同的布局和外观参数，并且紧跟着 select 菜单被追加到同一个页面。

```
<div class="large-menu">
     <ul>
          <li>
               <a href="blog.html">My Blog</a>
          </li>
```

```
            <li>
                <a href="home.html">My Home Page</a>
            </li>
            <li>
                <a href="tutorials.html">My Tutorials</a>
            </li>
        </ul>
</div>
```

为了使菜单变成响应式，需要为目标浏览器窗口尺寸创建一个媒介查询。如果浏览器窗口小于 800px，CSS 将只显示拥有 small-menu 类的 div 元素中的 select 表单。对于所有较大的浏览器窗口，CSS 将显示拥有 large-menu 类的 div 元素中的 ul 列表。当浏览器窗口跨越 801px 这样的临界宽度值时，页面上会呈现一个菜单切换的效果。

```
@media screen and ( max-width: 800px ) {
.small-menu { display:inline; }
.large-menu { display:none; }
}
@media screen and ( min-width: 801px ) and ( max-width: 1024px ) {
.small-menu { display:none; }.
.large-menu { display:inline; }
}
@media screen and ( min-width: 1025px ) {
.small-menu { display:none; }
.large-menu { display:inline; }
}
```

对于更大的屏幕尺寸，可以使用相同的 ul 列表，甚至可以使用相同的 HTML，但是通过媒介查询来应用新的 CSS 从而交付一个完全不同的菜单。

对于中等尺寸的菜单，使用 CSS 来显示列表项为水平列表，如下面的代码段所示。

```
.large-menu ul{
    list-style-type:none;
}
.large-menu ul li {
    display:inline;
}
```

由此得到了一个水平的菜单列表。我们希望这个导航版本出现在中等尺寸的浏览器窗口中。将其放置在范围为 801 ~ 1024px 的媒介查询中，如下面的代码段所示。

```
@media screen and ( min-width: 801px ) and (max-width: 1024px ) {
     .small-menu {
          display:none;
     }
.large-menu {
          display:inline;
     }
.large-menu ul {
          list-style-type:none;
     }
.large-menu ul li {
          display:inline;
     }
}
@media screen and (min-width: 1025px ) {
.small-menu {
          display:none;
     }
     .large-menu {
          display:inline;
     }
}
```

为了尽可能好地运用响应式导航元素，我们想要在屏幕宽度变化时，菜单列表版本能够移动到一个不同的布局位置。对于中等尺寸宽度，即 801 ~ 1024px，菜单停留在页面顶部并保持 100% 宽度。当屏幕宽度大于 1025px 时，菜单将浮动到父元素的左边。

在 801 ~ 1024px 的媒介查询中，为 large-menu 类添加一个 100% 宽度，对于 1025px 的媒介查询，为 large-menu 类添加一个 20% 宽度及 float:left 值。

为了填满这个页面，也将添加一个包装在 div 元素中的文本段落。可以再次使用 Lorem Ipsum 文本生成器来创建填充文本 (http://lipsum.com/)。在中等宽度媒介查询中设置包含该文本段落

的元素宽度为 100%。在最大的媒介查询中，设置包含该文本段落的元素宽度为 80%，将将其浮动到父元素的右边。

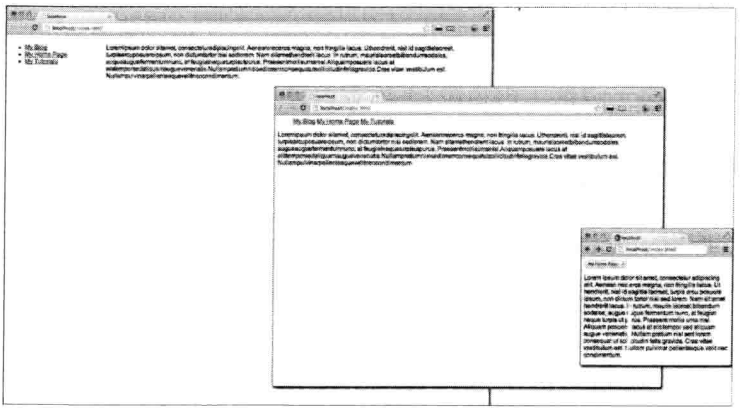
```
<div class="small-menu">
     <form>
          <select name="URL" onchange="window.location.href=this.form.
URL.options[this.form.URL.selectedIndex].value">
               <option value="blog.html">My Blog</option>
               <option value="home.html">My Home Page</option>
               <option value="tutorials.html">My Tutorials</option>
          </select>
     <form>
</div>

<div class="large-menu">
     <ul>
          <li>
               <a href="blog.html">My Blog</a>
          </li>
          <li>
               <a href="home.html">My Home Page</a>
          </li>
          <li>
               <a href="tutorials.html">My Tutorials</a>
          </li>
     </ul>
</div>

<div class="content">
     <p>Loremipsum dolor sitamet, consecteturadipiscingelit…</p>
</div>
```

样式应该与以下代码段一样。

```
<style>
@media screen and ( max-width: 800px ) {
     .small-menu {
          display: inline;
     }
     .large-menu {
          display: none;
     }
}
@media screen and ( min-width: 801px ) and ( max-width: 1024px ) {
     .small-menu {
          display: none;
     }
     .large-menu {
          display:inline;
          width: 100%;
     }
```

```
        .large-menu ul {
            list-style-type: none;
        }
        .large-menu ul li {
            display: inline;
        }
        .content: {
            width: 100%;
        }
    }
    @media screen and ( min-width: 1025px ) {
        .small-menu {
            display: none;
        }
        .large-menu {
            display: inline;
            float: left;
            width: 20%;
        }
        .content{
            float: right;
              h: 80%;
        }
    }
    </style>
```

最终结果是一个拥有三个不同版本导航菜单的页面。当发现对于每个特定的浏览器窗口尺寸都有菜单的一个优化版本时，你的观众将会感到惊讶。你可以在以下截图中看到这些神奇的导航元素。



1.6.3 工作原理

每个导航版本应用了 CSS3 的媒介查询属性来最大限度地提高菜单和内容的可用空间。在最小的窗口中（即小于 801px），导航整齐地排列在 select 表单元素中。在中等尺寸的窗口中（介于 801 ~ 1024px），导航是内联的并且横跨页面顶部，内容紧跟着导航。最后，对于最宽的浏览器窗口，导航浮动到了左边并且只占横屏宽度的 20%，而内容在宽屏浏览器窗口剩余的 80%（右边）中保持最大化。需要更多的计划和努力才能实现该技术，但是值得用它为你的用户提供尽可能最佳的视角。

1.7 基于尺寸的响应式内边距

为了衬托一个响应式宽度的图像元素，需要添加相对的内边距。如果使用静态的宽度内边距，图像内边距在较小的浏览器窗口中可能会显得过大，从而与其他附近元素相互挤压，甚至可能将图像挤出屏幕。

1.7.1 准备工作

理解盒模型属性的计算是一个好的开始。一个对象所占的总宽度是它的实际宽度加上它两边的内边距，边框以及外边距，即 2 *（外边距 + 边框 + 内边距）+ 内容的宽度 = 总宽度。

1.7.2 实现方式

假设一张图像在正常的非响应式状态下的宽度为 200px，典型的内边距可能是 8px，因此使用之前的盒模型，公式如下：

```
2 x ( 0 + 0 + 8px ) + 200px = 216px
```

为了找到内边距的百分比，使用内边距除以总宽度，即

8/216 = 0.037，舍入为 4%。

在创建响应式百分比宽度图像之前，先创建 CSS 和 HTML。给该图像添加一个内边距为 4% 的类。

```
<style>
p.text {
      float: left;
      width: 60%;
   }
div.img-wrap{
      float: right;
      margin: 0px;
      width: 38%;
   }
img.responsive {
      max-width: 100%;
      height: auto;
      padding: 4%;
   }
</style>

<p class="text">ipsum dolor sit amet, consecteturadi…</p>
<div class="img-wrap">
     <img alt="robot image" class="responsive" src="robot.png">
     <p>ipsum dolor sit amet, consecteturadipiscingelit…</p>
</div>
```

为了帮助你看到实际内边距宽度随着浏览器窗口尺寸的改变而变化，可以给图像的 CSS 添加一个背景颜色（background-color: #cccccc;）。

1.7.3　工作原理

图像内边距设置为 100% 则会紧贴其父元素的边缘。当父元素尺寸变化时，图像内边距也会相应调整。如果盒模型所对应的各项参数计算得当，布局将会成功地响应浏览器窗口的宽度变化。

1.8　基于 CSS3 按钮的进度条

和其他网站一样，你的网站也需要去迎合那些急性子的用户。

如果你的网站有一个可提交的表单，访客可能会不耐烦地多次单击“提交”按钮，因为你的页面加载新内容不够快。这可能会导致一个问题，就是表单多次提交同样的数据。

1.8.1 准备工作

为了防止这种行为，可以添加一些简单的视觉提示，告诉用户后台正在处理，请保持耐心。如果效果比较华丽，甚至可以给他们匆忙的生活增添点阳光。这个方法不需要任何图像，只需要使用 CSS 创建一个漂亮的渐变式提交按钮。本节内容很长，你可以先停下来喝杯咖啡。

1.8.2 实现方式

可以先创建一个包含几个文本框和一个提交按钮的表单。然后，为了使表单真的很酷，为文本框使用 HTML5 占位符属性。即使有占位符，这个表单还是相当单调。

注意，Internet Explorer 9 还不支持此方法。

```
<h1>My Form<h1>
<form>
     <ul>
        <li>
          <input type="text" placeholder="Enter your first name"/>
        </li>
        <li>
          <input type="text" placeholder="Enter your last name"/>
        </li>
     </ul>
<input type="submit" name="Submit" value="Submit">
</form>
```

通过 CSS 属性先给这个按钮增添点生气。

```
input[type="submit"] {
     color: white;
     padding: 5px;
     width: 68px;
     height: 28px;
```

```
    border-radius: 5px;
    border: 1px;
    font-weight: bold;
    border: 1px groove #7A7A7A;
}
```

下图是显示效果。

如果给按钮添加一个 CSS3 渐变效果，按钮甚至会更闪耀。为实现该效果，每个浏览器渲染引擎（Opera、Internet Explorer、WebKit(Chrome 和 Safari) 及 Firefox）的 CSS 代码都不一样。可以添加任意多的渐变偏移，只须添加一个 color 阶段及相对于顶端的位置百分比。每个偏移使用逗号分隔，如下列代码段所示。

```
<style>
input[type="submit"] {
    background: -moz-linear-gradient(top, #0F97FF 0%, #97D2FF
8%,#0076D1 62%, #0076D1 63%, #005494 100%);
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#0F97FF), color-stop(8%,#97D2FF)color-stop(50%,#0076D1),
color-stop(51%,#0076D1), color-stop(100%,#005494));
    background: -webkit-linear-gradient(top, #0F97FF 0%,#97D2FF
8%,#0076D1 62%,#0076D1 63%,#005494 100%);
    background: -o-linear-gradient(top, #0F97FF 0%,#97D2FF 8%,#0076D1
62%,#0076D1 63%,#005494 100%);
    background: -ms-linear-gradient(top, #0F97FF 0%,#97D2FF
8%,#0076D1 62%,#0076D1 63%,#005494 100%);
    background: linear-gradient(to bottom, #0F97FF
0%,#97D2FF 8%,#0076D1 62%,#0076D1 63%,#005494 100%);filter:
progid:DXImageTransform.Microsoft.gradient( startColorstr='#0f97ff',
endColorstr='#005494',GradientType=0 );
}
</style>
```

效果如下图所示。

可以通过 CSS 给该按钮再添加一个 hover 效果。使用该属性，当鼠标指针移动到按钮上时，看起来就像它被按下一样。下面的 CSS 将给按钮添加黑色边框。

```
input[type="submit"]:hover {
   border: 2px groove #7A7A7A;
}
```

下面是显示效果图。

使用 CSS3 盒阴影及 jQuery 可以实现一个简单的动画，在单击 Submit 按钮后有一个跳动的光晕环绕着该按钮。使用 jQuery 创建一个事件监听器来监听按钮的 click 事件。在 click 事件中

作用于该表单按钮元素的一系列类将发生变化。脚本将动态添加 Partial-fade 类到按钮元素中。

不要忘记添加 jQuery 源码链接到 head 标签中：

```
<scriptsrc="http://code.jquery.com/jquery-latest.
js"></script>
```

然后，在表单后面插入下列脚本：

```
<script >
//Submit Glow
$('input[type="submit"]').click(function() {
$(this).addClass('partial-fade');
   $(this).animate({
      opacity: 0.1
   }, 8).animate({
       opacity: 0.9
   }, 226).animate({
       opacity: .5
   }, 86);
   setTimeout(function () {
      $('input[type="submit"]').removeClass('partial-fade');
   }, 366).animate({
       opacity: 1
   }, 86);
});
</script>
```

为了实现单击按钮后按钮的炫目特效，需要新加一个 partial-fade 类到 CSS 文件中，赋予它一个 CSS3 盒阴影属性，并且修改边框属性。

```
<style>
input[type="submit"].partial-fade {
     border-top: 1px solid #CFF !important;
     border-right: 1px solid #CCF !important;
     border-left: 1px solid #CCF !important;
     border-bottom: 1px solid #6CF !important;
     -webkit-box-shadow: 0 08px 0px #0F97FF, inset 0 0 20px rgba(37,
141, 220, 1);
     -moz-box-shadow: 0 0 8px 0px #0F97FF, inset 0 0 20px
rgba(37,141,220,1);
```

```
    box-shadow: 0 0 8px 0px #0F97FF, inset 0 0 20px rgba(37, 141,
220, 1);
}
</style>
```

现在，当 Submit 按钮被按下时有蓝光一闪而过的特效。下图显示了最终的样子。

哇！为了给该按钮实现这样一个小细节我们真是大费周章。但是这样的细节真正有助于制作一个漂亮的网站。这正是我喜欢用来给访客惊喜的细节之一。

1.8.3 工作原理

CSS3 背景渐变可以使一个漂亮的按钮在不同浏览器上外观一致。渐变是复杂的，尤其是当前每个浏览器要求各自的 CSS 代码实现渐变。可以通过手动添加百分比和颜色控制渐变转效点（breakpoint）。当事件被触发时，添加盒阴影、边框及 jQuery 使按钮产生有趣的效果。

第 2 章
响应式字体

2.1 简介

本章主要论述如何制作响应式字体。你将学到优化文本以适应不同设备的技巧，以及美化字体的方法。涉及的技术只有 CSS3 及结合了 JavaScript 的 HTML5 的画布元素。借助响应式字体，能够向文本应用一些令人兴奋的效果。

学习完本章，你将掌握一系列技术，能够用它们创作神奇的响应式站点。本章只涵盖了基础知识，但如果再结合一些创造力，你就能够制作出精彩的产品。

2.2 创建自适应的响应式字体

该技巧是一个简单的响应式字体示例。它将演示如何使用新的尺寸单元 REM。REM 的意思是根 EM（Root EM）。如果使用 EM 单元，意味着字体尺寸与根元素字体尺寸有关，而不是父元素字体。

2.2.1 准备工作

在进一步讨论之前，我们先看看这个技巧。先通过我最爱的 Ipsum 生成器 (http://ipsum.com) 生成一些用于填充的文本内容。至少生成一个段落文本并复制到剪贴板中。

2.2.2 实现方式

现在，将填充文本粘贴到 HTML 文档中并包装在一个段落标签中。设置该段落元素的类为 a，然后再复制一份并将新段落元素的类改为 b。如下面的代码片段所示：

```
<p class="a">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
<p>
```

```
<p class="b">
    ultricies ut viverra massa rutrum. Nunc pharetra, ipsum ut
    ullamcorper placerat,
<p>
```

接下来，创建 HTML 的基本 font-size 属性和静态尺寸段落的 font-size 样式，用来比较字体大小的改变，这有点像实验中的对照组：

```
html{font-size:12px;}
p.b{font-size:1rem;}
```

然后创建两个 @media 查询，一个为 orientation:portrait，另一个为 orientation:landscape。在 orientation:portrait 媒介查询中，设置“a”类段落元素的 font-size 属性值为 3rem。在 orientation:landscape 媒介查询中，设置“a”类段落的 font-size 属性值为 1rem。

```
@media screen and (orientation:portrait){
p.a{font-size:3rem;}
}
@media screen and (orientation:landscape){
p.a{font-size:1rem;}
}
```

现在，将浏览器窗口从横向模式调整到纵向模式时，会看到第一个段落的字体大小与基本字体大小的比例从 1:1 变为了 3:1。这虽然看起来非常简单，但是该方法可以多样化，基于此方法可以实现很多令人印象深刻的响应式字体效果。

2.2.3　工作原理

当浏览器发送请求时，CSS3 的 @media 查询会基于视窗宽度返回不同的样式。随着视窗大小的改变会实时地加载或建立（重建）样式。虽然不会有太多的用户在浏览网站过程中频繁改变视窗大小，但是如何使得网站更好地适应不同浏览器窗口大小往往是一项费时的工作。

2.2.4 相关章节

- 2.11 节

2.3 使用画布实现文本阴影

HTML5 为网页设计引入了一个新元素，即画布 (canvas) 元素。该元素用于在网页中使用 JavaScript 实时创建图形。

2.3.1 准备工作

画布元素会在页面上创建一个矩形区域。默认的尺寸规格为 300px 宽，150px 高。可以使用 JavaScript 指定具体大小。本方法的相关示例代码激增较快，可以在 Packt Publishing 的网站中获取全部相关代码。

2.3.2 实现方式

首先，在一个简单的 HTML 页面中创建画布元素：

```
<!DOCTYPE HTML>
<html>
     <head>

     </head>
     <body>
          <canvas id="thecanvas"></canvas>
     </body>
</html>
```

通过 JavaScript 从 DOM 中获取该画布元素。

```
var canvas = document.getElementById('thecanvas');
```

然后调用 getContext() 方法。getContext(‘2d’) 方法的返回值是一个内置的 HTML5 对象。它拥有若干用来绘制文本、形状、图像等的方法。

```
var ctx = canvas.getContext('2d');
```

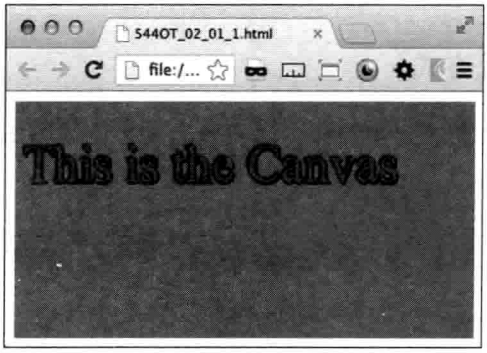
接下来使用 JavaScript 绘制文本。我们编写代码来设置该对象的水平和垂直阴影偏移量、模糊度及阴影颜色等属性。

```
ctx.shadowOffsetX = 2;
ctx.shadowOffsetY = 2;
ctx.shadowBlur = 2;
ctx.shadowColor = "rgba(0, 0, 0, 0.5)";
```

文本内容及属性可以直接通过下面的 JavaScript 代码设置，但是也可以通过 DOM 中的变量来传入。

```
ctx.font = "20px Times New Roman";
ctx.fillStyle = "Black";
ctx.fillText("This is the canvas", 5, 30);
```

回到 HTML 代码，给 body 元素添加 onload="drawCanvas();" 脚本命令。当页面加载时，JavaScript 将触发绘制事件，从而将文本及其阴影绘制到画布中，如下面的截图所示。



2.3.3　工作原理

不用太深入了解 JavaScript 的工作原理，设计者就可以编写脚本操控画布元素，从而在页面加载时绘制页面所需内容。body 元素的 onload=“drawCanvas();”命令会触发 JavaScript 的执行，把内容绘制到画布上。

2.3.4 相关章节

- 2.5 节

2.4 使用画布实现内侧阴影和外侧阴影

本方法也使用画布和 JavaScript 通过浏览器绘制文本和实现特殊效果。通过画布无法直接实现内侧渐变的阴影效果或插图效果，但是，使用 stroke 方法可以模拟文本中的内侧阴影。

2.4.1 准备工作

本方法会使用前面章节已经使用过的一段代码。可以从 Packt Publishing 的网站直接下载这段代码，它与 2.3 节中的代码是一样的。这段代码可以运行在本地电脑中，不需要任何 Web 服务器。可以从本书页面中得到所有相关代码。

2.4.2 实现方式

在开始之前，先创建一个含有画布元素的简单 HTML 页面。

```
<html>
  <head>

  </head>
  <body>
    <canvas id="thecanvas"></canvas>
  </body>
</html>
```

通过 JavaScript 从 DOM 中取得该画布元素。

```
var canvas = document.getElementById('thecanvas');
```

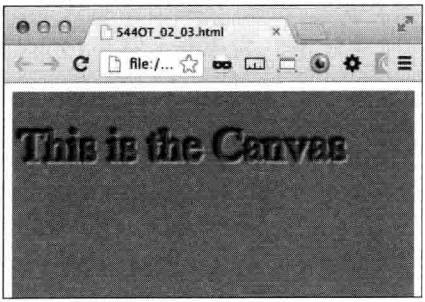
然后调用 getContext() 方法。getContext('2d') 方法的返回值是一个内置的 HTML5 对象。它拥有若干方法用来绘制文本、形状和图像等。

```
var context = canvas.getContext('2d');
```

下面这段脚本通过多种效果的组合来实现内侧阴影和外侧阴影。脚本中添加了一个投影和两个相异的轮廓。首先，添加一个阴影到左上角，背景色为黑色，并使得 context.shadowBlur 属性的值为 2。然后调用 context.fillText 方法，并给画布的 context 对象设置 context.strokeStyle 和 context.strokeText 属性。

```
context.shadowOffsetX = -1;
context.shadowOffsetY = -1;
context.shadowBlur = 2;
context.shadowColor = "#888888";
context.textAlign = "left";
context.font = "33px Times New Roman";
context.fillStyle = "#666";
context.fillText("This is the Canvas", 0, 50);
context.strokeStyle = "#555";
context.strokeText("This is the canvas", 2, 50);
context.linewidth = 2;
```

与突起的外观效果不同，该特效使得文本有一定的倾斜角度，并且有一个内发光的阴影效果。这种效果如下图所示。



2.4.3　工作原理

正如本节的开头所述，画布元素对象并没有提供能够直接实现内侧阴影效果的方法。但是可以结合 context.fillText 和 context.strokeStyle 方法来创建一个足够逼真的内侧阴影特效。

2.5 使用画布旋转文本

HTML5 中的画布元素不只是能给文本着色或者添加阴影，你也可以用它来移动或操作位于画布区域中的元素对象。在本节中，我们将旋转位于画布中的元素对象。

2.5.1 准备工作

学习本节需要掌握之前的几节。如果跳过了之前的几节也没关系，你还可以参照完整代码来学习。

2.5.2 实现方式

一旦你完成了之前技巧中的画布设置步骤，那么实现旋转的基本步骤很简单。在函数开头添加一个 rotate 方法的调用。

```
context.rotate(Math.PI/4,0,0);
```

你可能会发现位于画布上的文本向右旋转了。怎么回事？无论画布中含有什么元素，rotate 方法默认会旋转整个画布。

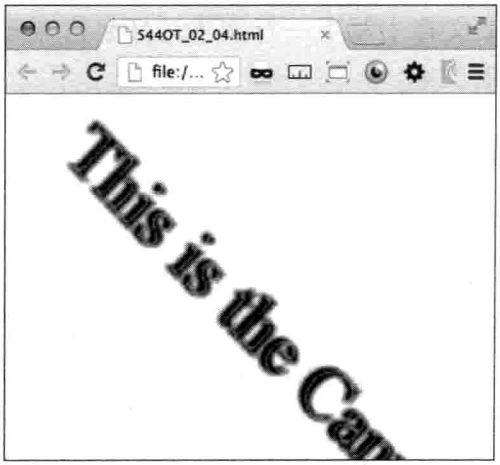
画布的默认尺寸较小，为 300px 宽，150px 高。修改画布中所含元素的尺寸属性不会影响画布尺寸，但会使画布中的元素对象失真。如果要修改画布及画布中元素的尺寸，可以通过在 JavaScript 中设置 canvas.width 和 canvas.height 属性来完成。

```
canvas.width=250;
canvas.height=250;
```

另外，因为整个画布自身被旋转了，但是文本旋转并没有围绕特定的参照点，因此文本位置需要被重新定位到正确位置。在本例子中，需要设置填充和笔画的偏移：

```
context.fillText("This is the Canvas", 140, 1);
context.strokeText("This is the Canvas ", 140, 1);
```

效果如下图所示。



2.5.3　工作原理

JavaScript 能够通过 rotate 方法来旋转整个画布以及绘制在画布中的所有元素。只是在决定使用该方法时需要预先考虑所使用的场景和进行相关设置。虽然看起来有些复杂，但这不失为在大型 web 项目中实现响应式设计的理想工具。

2.5.4　相关章节

- 2.6 节

2.6　使用 CSS3 旋转文本

CSS3 提供了一个旋转文本的简单方法。如果项目不需要使用复杂的画布元素，使用 transform:rotate 属性不失为一个易于实现的解决方案。

2.6.1 准备工作

在HTML文档中写下一行文本。打起精神来，你将使用CSS3旋转这行文本。

2.6.2 实现方式

将文本包裹在段落标签元素中：

```
<p class="rotate">I think, therefore I am</p>
```

然后，添加CSS transform属性来旋转文本。每个浏览器实现旋转的方式都不同，所以需要设置每个浏览器自身唯一的transform属性。然而，每个设置都会使用transform属性的子属性rotate，并且后面紧跟着旋转角度，如下面的代码片段所示：

```
<!DOCTYPE HTML>
<html>
     <head>
          <style>
     .rotate {
/* Chrome, Safari 3.1+*/
-webkit-transform: rotate(-90deg);
/* Firefox 3.5-15 */
-moz-transform: rotate(-90deg);
/* IE9 */
-ms-transform: rotate(-90deg);
/* Opera 10.50-12*/
-o-transform: rotate(-90deg);
/* IE */
transform: rotate(-90deg);
}
          </style>
     </head>
     <body >
          <p class="rotate">I think, therefore I am </p>
     </body>
</html>
```

2.6.3 工作原理

transform属性可以给元素应用2D或3D变换。除了旋转，

其他相应属性还可以实现元素的移动、扭曲以及透明化效果。

2.6.4　相关章节

- 2.5 节

2.7　使用 CSS3 制作 3D 文本

在前几节中，我们使用画布元素创建了投影、斜面以及内侧阴影。使用 CSS3 可以让文本真的站起来。使用 CSS3 的 text-shadow 属性，可以让文本看起来好像伸出屏幕来面对用户。

2.7.1　准备工作

如果你想直接跳过本节，你可以从 Packt Publishing 的网站上下载代码。否则，如果你想边学边做，那就一起来创建属于你自己的 3D 文本吧。下面将通过组合多个 CSS3 的阴影效果来实现 3D 特效。

2.7.2　实现方式

在 IDE 中，创建一个在 body 元素中只有一个 header 元素的 HTML 文档。在 head 标签中设置一些样式，并给 header 元素设置 color: #f0f0f0;，如下面代码片段所示：

```
<style>
    h1{ color: #f0f0f0;}
</style>
```

现在再添加 7 个 X 和 Y 方向递增或递减的 text-shadow 系列属性，如从 0px 0px0px #666 到 -6px -6px 0px #666。

```
text-shadow: 0px 0px0px #666,
-1px -1px 0px #666,
-2px -2px 0px #666,
```

```
-3px -3px 0px #666,
-4px -4px 0px #666,
-5px -5px 0px #666,
-6px -6px 0px #000,
```

现在可以看到 header 已经跨出了屏幕。好了，几乎完成了！为了确保它真的能够弹出屏幕，我们需要再加些效果。当在屏幕上创建任何 3D 对象时，给予它一致的照明和阴影是相当重要的。因为该文本向上跃起，所以需要一个阴影。

添加另外 6 个 X 和 Y 方向的 text-shadow 属性，只是坐标值为正值，颜色相对较浅（color:#ccc;）。

```
1px 1px 5px #ccc,
 2px 2px 5px #ccc,
 3px 3px 5px #ccc,
 4px 4px 5px #ccc,
 5px 5px5px #ccc,
 6px 6px 5px #ccc;
```

这个阴影效果还可以，但是看起来还是有点假。好吧，把它提高到另一个层次。我们把元素背景模糊变暗。Text-shadow 属性的第 3 个数字用以实现模糊效果，所以设置一个递增的值使其越来越模糊：0、0、1、1、2、3 和 5，如下面的代码所示。除此之外，后面的背景色也应该越来越暗：#888、#777、#666、#555、#444、#333 和 #000。

```
text-shadow:0px 0px0px #888,
-1px -1px 0px #777,
-2px -2px 1px #666,
-3px -3px 1px #555,
-4px -4px 2px #444,
-5px -5px 3px #333,
-6px -6px 4px #000,
```

现在 header 有了一个完全逼真的 3D 效果。下面的截图显示了这个特效。

2.7.3　工作原理

尽情试验和体验本方法的不同组合方式，来实现那些令人欢呼的字体效果。CSS3 将字体设计带到了一个全新水平，而之前要想达到如此深度的定制特效却总是苦难的事情。CSS3 不仅做到了，并且做得很好。

text-shadow 属性可以处理大量的阴影属性。因此，你可以把这些阴影按照与文本之间的距离依次堆叠起来。随之而来的，你的文本拥有了 3D 特效。

2.8　基于文本遮罩的文本纹理

CSS3 处理图像的强大功能使我们可以通过一张图片来给文本添加图像遮罩。换做以前，这只能在图像处理工具中创建一张带有该效果的静态图片来实现。

2.8.1　准备工作

首先需要一张图片来作为纹理遮罩。使用图片编辑软件，生成一张具有透明通道（alpha channel）的新图片。如果没有能够生成带透明通道的 PNG 格式图片的图片编辑工具，你可以从 http://

www.gimp.org 下载名叫 GIMP 的免费图片编辑工具。可以在图片的顶部采用散点画笔创建一个具有一定纹理的区域，来快速生成一张用于纹理遮罩的图片。

保留该图片的透明通道，以 PNG 格式将其保存在网站对应的 images 目录下。

2.8.2 实现方式

在你的 HTML 文件中定义一个头元素，其中包含需要添加纹理特效的文本。然后，为其添加一些文本内容：

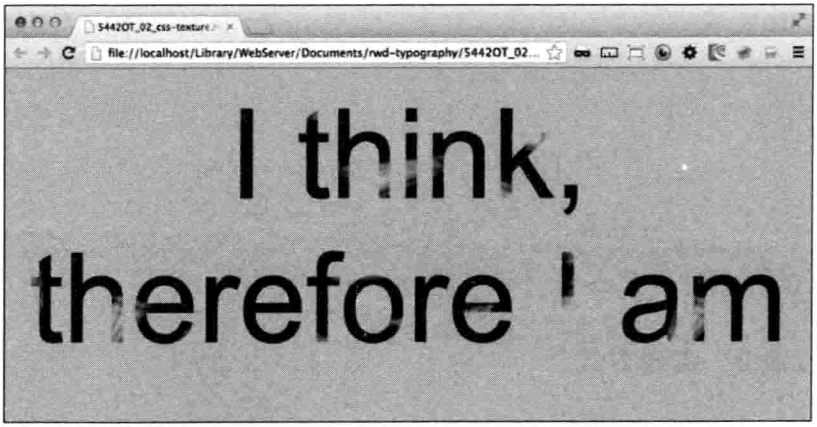
```
<h1 class="masked">I think, therefore I am</h1>
```

现在可以添加 CSS 标记了。这将包括很大的字号（为了炫耀纹理渲染的效果）、白色字体，内边距及对齐等，当然也包括图片遮罩属性。

注意不同浏览器要求的属性前缀不同。

```
h1.masked{
     font: 140px "Arial";
     color: white;
     -webkit-mask-image: url(images/mask2.png);
     -o-mask-image: url(images/mask2.png);
     -moz-mask-image: url(images/mask2.png);
     mask-image: url(images/mask2.png);
     text-shadow: 0px 0px 10px #f0f0f0;
     width: 100%;
     padding: 12% 0 12%;
     margin:0;
     text-align: center;
    }
```

CSS 特效呈现在下面的截图中。



2.8.3　工作原理

遮罩图片根据本身的透明度裁剪了元素的可视区域。使用 CSS 将遮罩图片应用于文本时，遮罩部分会被裁剪掉。该方法的实现原理与采用图像编辑软件产生的透明图层类似。

2.9　基于位置伪类的交替行样式

对于那些在以前需要恼人且复杂的方案来解决的问题，CSS3 中的位置伪类提供了极为简单的解决方案。直到最近，为了给列表或表格中的交替行添加样式，如果能够在服务器端添加一些处理逻辑，你至少可以通过计数器来遍历列表，更坏的情况是，你甚至得手动为你的行元素编号。

2.9.1　准备工作

CSS3 所提供的解决方案非常简单。首先创建带有列表元素的 HTML 文件。给列表元素指定类并不是必需的，因为你可能想将该样式应用到整个网站：

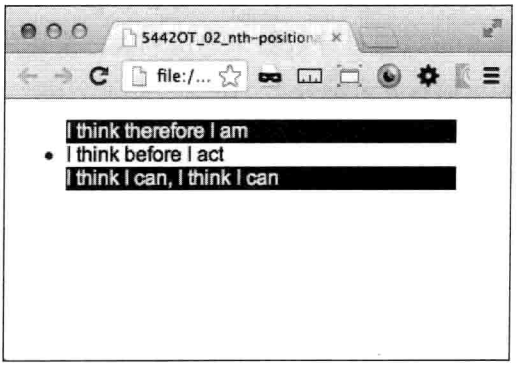
```
<ul>
    <li>
        I think, therefore I am
    </li>
    <li>
        I think before I act
    </li>
    <li>
        I think I can, I think I can
    </li>
</ul>
```

2.9.2 实现方式

为列表元素添加相应的 CSS 属性，给奇数位置的 <li> 元素设置基于位置信息的伪类。该伪类的属性包括背景色，以及能够显著与默认字体颜色模式区分的字体色。

```
  ul{
width:100px;
  }
  li:nth-of-type(odd){
background-color:#333;
color:#f0f0f0;
  }
```

该方法会自动给位于奇数位置的列表元素添加样式，十分神奇的感觉！特效呈现在下面的截图中。



可以休息一下了，一切都很简单！

2.9.3　工作原理

依据 http://www.w3.org 上的定义，伪类符号 :nth-of-type(an+b) 表示元素在同一元素层次下有 an+b-1 个兄弟节点在其之前，n 在文档树中表示零或任意正整数，除此之外，元素还得有对应的父元素。

这究竟意味着什么？这就是说，只要某元素在其父元素下拥有相似的兄弟元素，你就能够使用（*–n*+2）这样的公式来表示兄弟元素的最后两行，或者为了简便，直接使用 even 或 odd 代表奇偶子元素，然后就可以使用 CSS 来对这些元素添加样式。

2.10　基于 before 及 after 伪元素添加字符

就像在电视剧《迷离时空》（*The Twilight Zone*）中遗失的一集一样，CSS 提供的新属性赋予了使用者在内容中添加伪标记的能力。可能听起来会有点奇怪，但这种方法却有着众多的应用场景。例如一个场景是，需要在显示的时候把文本内容放进引号之中，但是同时不想在内容或者主题文件中添加这些引号，显然这样是比较明智的做法。或者你也想通过标签和 @ 标记尝试时下流行的 Twitter，例如想要在内容之前添加 # 或是 @ 符号。采用 CSS 标记方法即可完成功能，呈现出如下代码行中的内容：

```
#I think, therefore I am#
```

2.10.1　准备工作

该方法不需要任何服务器端的逻辑或者特殊技巧。你只需在本地打开页面就能看到实际的应用效果。

2.10.2 实现方式

实现该方法只需要 CSS 即可，因此需要做的只是创建 HTML 页面，在页面中对于目标内容设定 class 或 id 属性：

```
<h2 class="hashtag">I think, therefore I am</h2>
```

CSS 标记方法有一点需要理解，插入的符号使用被插入内容所设定的内边距及外边距。该方法使用 class:before 和 class:after 伪类。在 CSS 中对 before 的设定为 .class:before {content:"#";}。用你需要使用的内容替换 #。至于 after，采用 .class:after{} 替换 .class:before{}。

```
.hashtag {
    border:1px solid #ccc;
    display:block;
    width:200px;
    height:10px;
        }
.hashtag:before{
    content:"#";
        }
.hashtag:after{
    content:"#";
        }
```

2.10.3 工作原理

CSS 中的 before 及 after 伪元素分别在元素内容的前后添加设定内容。需要注意的是，添加的内容不是真正的内容或者元素，不能应用标记方法或是 JavaScript 中的事件触发器。

2.11 基于相对字体大小的按钮

在一些场景下，响应式的按钮字体也是必需的。网站的移动版就是很好的例子。当在 iPhone 中查看一个常规按钮时，往往发现其太小而不易点击。结果就是，对移动设备的疏忽，带给使用移动设备的用户非常糟糕的用户体验。

2.11.1　准备工作

本节中所介绍的方法是使用新的字体尺寸单位 REM。通过本方法，使用移动设备访问的用户能够看到更大的响应式的按钮字体。

REM 是 CSS3 引入的新字体尺寸单位。是 Root EM 的简写，意即相对于根元素的字体大小。而 EM 是相对于父元素的字体大小，因此二者是有区别的。一种使用场景是在某些元素中使用 REM，让其获得相对于整个 body 基本字体大小的值。

2.11.2　实现方式

该方法通过 @media 查询的方式为桌面设备及移动设备构建响应式的按钮。下面就是具体的实现步骤。

首先创建一个简单的 HTML 页面，含有一些基本占位符的文本信息（http://lipsum.com）和一个 submit 类型的 input。

```
<div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum
vehicula enim at dolor ultricies ut viverra massa rutrum. Nunc
pharetra, ipsum ut ullamcorper placerat,
</p>
    <input type="submit">
</div>
```

接着为刚才的 HTML 页面元素添加 CSS 属性，基本字体设置为 62.5%，同时为文本信息设置一个静态的字体大小，以此作为该方法的参照。

```
html{font-size:62.5%;}
p{font-size:1.4rem;}
```

下一步需要创建 @media 查询，以期适应移动设备及两个不同大小的桌面设备窗口。在此为桌面设备添加额外的 @media 查询的目的在于，即使无法访问移动设备，也能立即观察到响应式

的效果。

为大小为 1024px 和 1280px 的桌面设备建立 @media 查询，同样为移动设备设置两个 @media 查询，均为 max-device-width: 480px，但是为了适应横竖屏，其中一个为 orientation:landscape，另一个为 orientation:portrait。

```
@media screen and (min-width:1024px){ }
@media screen and (min-width:1280px){ }
@media screen and (max-device-width: 480px) and
(orientation:landscape){ }
@media screen and (max-device-width: 480px) and (orientation:portrait)
{ }
```

在桌面设备的两个 @media 查询中，都添加一个 input 元素；对于 min-width:1024px 的查询设置为 font-size:1rem，而对于 min-width:1280px 的查询则为 font-size:2rem。为这两个查询添加属性 width:84px; 和 padding:2%。

在为移动设备设置的 @media 查询中，同样都添加一个 input 元素。在 orientation:landscape 的查询中，设置属性 font-size:2rem; 及 width:25%;，而对于 orientation:portrait，该属性值则为 font-size:2.4rem; 和 width:30%;。

```
@media screen and (min-width:1024px){
            input{
                font-size:1rem;
                width:84px;
                padding:2%;}
        }
@media screen and (min-width:1280px){
     input{
          font-size:2rem;
          width:84px;
          padding:2%;

     }
}
@media screen and (max-device-width: 480px) and
(orientation:landscape){
     input{
          font-size:2rem;
```

```
            width:25%;
            padding:2%;
        }
}
@media screen and (max-device-width: 480px) and
(orientation:portrait){
        input{
            font-size:2.4rem;
            width:30%;
            padding:2%;
        }
}
```

现在如果通过移动设备查看该页面，就能发现 REM 字体大小单位如何相对于设置的基本字体创建新的字体。在移动设备上，可能会出现字体太小而很难辨识，同时按钮又太小而不易使用的情况。将你的移动设备从竖屏转换为横屏，马上就能发现按钮及其字体大小发生了变化。

对比移动设备版本和桌面设备版本的实现，会发现在每一个不同的设备下面，按钮的属性值均有所区别。如果将桌面设备浏览器窗口在 1024px 和 1280px 之间缩放的时候，同样会发现按钮的字体有所变化。

2.11.3　工作原理

字体大小单位 REM 创建了一个字体大小，该字体大小是相对于在 HTML 或 body 元素中的已经声明的基本字体大小而言的，如果未声明基本字体大小，则是相对于内建字体大小的。而通过 @media 查询的方式则能在不同设备和不同方向下给出实现相对字体大小的解决方案。

2.12　为字体添加阴影效果

使用 CSS3 可以轻松地为文本添加阴影效果。既可为特定的元素添加高亮特效，也可应用于 body 文本段落中达到加强内容的

视觉效果。除此之外，也能用来使文本链接更为突出。

2.12.1 准备工作

CSS3 让这些变得简单，同时也不需要复杂的设置。打开你的开发环境或是记事本开始特效之旅。当然你也可以访问从 Packet Publishing 上本书页面获取完整的代码并看看它是如何实现的。

2.12.2 实现方式

首先，创建文本段落。记住，文本内容可以通过我们所喜爱的文本内容生成工具 http://lipsum.com 得到。现在给文本信息设置一个标题：

```
<h1>I think therefore I am </h1>
<p>Lorem ipsum dolor sit amet…
</p>
```

在文本内容中插入一些链接，设置 href 属性，同时为链接添加一些描述词语：

```
<h1>I think therefore I am</h1>
<p>Morbi<a href ="#">venenatis</a>Lorem ipsum dolor sit amet…
<a href ="#">scelerisque</a> Lorem ipsum dolor sit amet…</p>
```

首先，给文本段落添加投影特效。CSS3 所提供的 dropshadow 特效就能实现。在 CSS 文件中添加 text-shadow 属性。对于 IE 浏览器，添加 filter 属性。

```
text-shadow: 1px 1px 2px #333333;
```

以上设置使得文本有一点点的阴影效果，似乎内容从页面中跳出一样。对于正文内容，该效果就已足够。但是对于链接元素来说还不够，还得添加不同层次的阴影特效，才能使链接更加突出。添加阴影的方法与前面的示例一样，只是在后面添加逗号，然后在逗号后面添加另外一个阴影特效。下面的例子给链接文本

添加了浅蓝色的阴影效果。

```
text-shadow: 0px 0px 1px blue, 1px 1px 2px #333333;
filter: dropshadow(color=blue, offx=1, offy=1);
```

现在添加一个属性来给整个页面一些亮点。通过伪悬浮动作(:hover)来实现链接的 flash 特效：

```
p.shadowa:hover{
text-shadow: 0px 0px 8px #ffff00, 2px 2px 3px #666; filter:
dropshadow(color=#ffff00, offx=1, offy=1);
}
```

该属性设置使得在文本段落中的链接在鼠标悬停时呈现出黄色辉光。该特效如下图所示。

2.12.3　工作原理

该方法是阴影特效的组合。可以通过将不同层次的阴影特效组合在一起产生逼真的 3D 效果。在此最好的方式就是应用不同的组合方式直到得到满意的 3D 特效。

2.13　基于边框半径的圆角实现

在网页设计的世界中，弧形圆角一度被看做圣杯（Holy Grail）[⊖]。它永远都是存在的，但是却很难实现。对于如何使得元素拥有弧

⊖　圣杯意指非常美好却很难得到的事物。——译者注

形圆角，设计者们往往只能在有限且拙劣的方法中选择。

2.13.1 准备工作

在 CSS3 的世界里，这些烦恼都不复存在。通过设置 border-radius 属性这样简单的方法便能给元素创建圆角特效。

2.13.2 实现方式

首先创建一个 HTML 元素。该方法对于含有边框的元素均能生效。因此可以创建一个带有边框的文本段落，然后从 http://lipsum.com 获取一些填充文本。

```
<p class="rounded"> Lorem ipsum dolor sit amet…</p>
```

接着设置 CSS 相关属性来修饰该段落元素：

```
.rounded{
        background-color:#ccc;
        width:200px;
        margin:20px;
        padding:20px;
    }
```

为了实现圆角效果，添加 CSS3 中的 border-radius 属性。在本示例中，使用的弧度半径为 5px。

```
border-radius: 5px;
-webkit-background-clip: padding-box;
background-clip: padding-box;
```

设置该属性是一种实现圆角特效的简单实用的方法，对于页面上的浮动元素特别有效。但是如果只是想对菜单元素的顶部角落添加圆角特效呢？答案是，同样容易。

首先来看看一个简单的列表：

```
<ul class="inline">
    <li class="rounded-top"><a href="#">menu 1</a></li>
    <li class="rounded-top"><a href="#">menu 2</a></li>
    <li class="rounded-top"><a href="#">menu 3</a></li>
```

```
    <li class="rounded-top"><a href="#">menu 4</a></li>
</ul>
```

然后在 CSS 中设置列表为内联显示，同时设定其内边距及外边距。

```
li.rounded-top{
    display:inline;
    background-color:#ccc;
    margin:3px;
    padding:8px;
}
```

在之前的示例中，所有的元素角都含有同样的圆角特性。为了能够让彼此之间有所区分，可以为每个特定的元素角设定一个半径。

```
border-radius: 8px 8px 1px 1px;
```

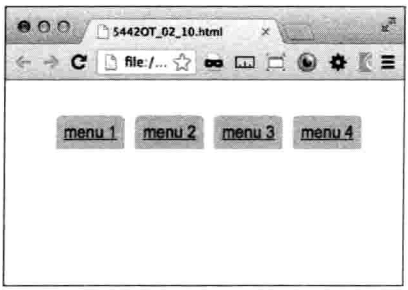
同样也可以在 CSS 中为每个元素角设置其特有的属性，这也能达到相同的效果。

```
border-top-left-radius:8px;
border-top-right-radius:8px;
border-bottom-right-radius:2px;
border-bottom-left-radius:2px;
```

更进一步，还可以设置更多的弧度半径：

```
border-top-left-radius:8px 4px;
border-top-right-radius:8px 4px;
border-bottom-right-radius:2px;
border-bottom-left-radius:2px;
```

最后的页面显示呈现在下面的截图中。



用百分比设置圆弧半径是达到同样效果的另外一种实现方式。回到本方法中的第一个示例，改变 CSS 设置，用百分比取代之前的圆弧半径值：

```
border-radius: 1%;
```

2.13.3 工作原理

属性 border-radius 提供了简易的绘制弧形元素的功能。该属性需要四个值，但是简写格式只需一个弧度值。

第 3 章
响应式布局

3.1 简介

本章中的一些方法具有一定的挑战性。实现响应式布局常常意味着困难和挑战，因而往往能够促使新的解决方案的出现。而通过使用这些响应式设计方法，你可以做得更多、更好。响应式布局为网页设计带来了全新领域中的挑战，同时也带来了新的用户体验。

3.2 基于 min-width 和 max-width 属性的响应式布局

响应式布局的很多解决方案都相当复杂，但是本节中的方法却非常简单。该方法通过在 3 个浮动元素上设置 min-width 和 max-width 属性来实现响应式布局。只要采用 CSS 中这个非常简单的响应式布局特性，就能够适应移动设备和不同尺寸的计算机屏幕。

3.2.1 准备工作

在极小视窗的限制下，把浮动元素的多个列合并显示为一个列并不是什么新鲜的玩意，在很多年前就已经成为 CSS1 的一个标准属性。但是在移动设备普及之前，该属性一直被认为没有多大的作用。接下来我们使用这个老旧的状态属性，结合一些新的 CSS 属性来实现响应式布局。

3.2.2 实现方式

创建一个简单的根元素为 article 的 HTML 页面，其中包含一个 h1 的头元素及三个 div 元素。第一个 div 元素包含一张图片，第二个和第三个 div 元素包含一些文本内容。给这三个元素都设

置 float 类，然后再分别设置其 ID 为 one、two 及 three：

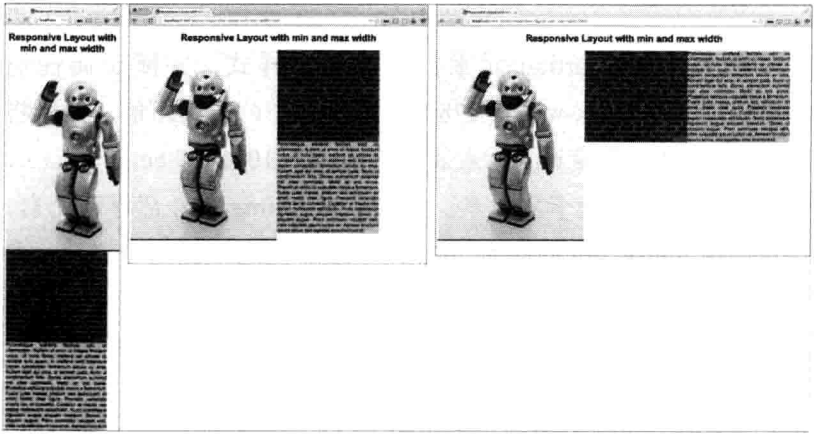
```
<article>
     <h1>Responsive Layout with min and max width</h1>

     <div class="one float">
        <img src="images/robot.png">
     </div>

     <div class ="two float">Pellentesqueeleifendfacilisisodio ac
     ullamcorper. Nullamutenimutmassatinciduntluctus...
     </div>

     <div class="three float">Pellentesqueeleifendfacilisisodio ac
     ullamcorper. Nullamutenimutmassatinciduntluctus. Utnullalibero, …
     </div>
</article>
```

接下来，为 .article 元素创建对应的样式，并赋予属性值 width: 100%;、max-width: 1280px; 以及自动的侧边距值。然后将 h1 标题居中。设置 img 元素属性为 width: 100% 和 height: auto;，使得该元素自适应于父元素。而对于含有 img 的外部浮动元素，设定属性 min-width 的值为 500px。当然，也可以给每个浮动元素设置不同的背景色，从而让它们更容易辨识。但是对于响应式布局来说，这不是必需的。对于已经设置过 .float 类的浮动元素，可以添加 max-width: 350px 的属性值，并同时设置向左浮动。为了进一步看得更清楚，设置文本为两端对齐。

```
<style>
article{
     width: 100%;
     max-width: 1280px;
     margin: 0 auto;
}
h1 {text-align:center;}
img {
     width: 100%;
     height: auto;
}
.one {
     background-color: #333;
```

```
        min-width: 500px;
    }
    .two {background-color:#666}
    .three {background-color:#ccc}
    .float {
        max-width: 350px;
        float: left;
        text-align: justify;
    }
    </style>
```

当这一切都完成以后，现在你只需要在浏览器中打开 HTML 页面，接下来会看到页面布局非常平滑地从三列变为两列，最终只有单列的布局方式。具体的效果见下面的截图。



3.2.3 工作原理

施加于列元素之上的 max-width 属性使得列的宽度不是固定的，但是同时限定了最大值。相对于设置静态的宽度，该方法赋予了列布局更好的灵活性。图片所在列利用 min-width 属性，能够响应父元素宽度的变化，比如增加或收缩宽度。最终，通过 float 属性，整个页面能够从三列的布局方式平滑地过渡为单列布局方式；一旦没有足够的空间并列容纳 float 元，最后一个列元素

将会从列布局中移除，并放置在新的一行中。

3.3　基于相对内边距的布局控制

本节将为博客页面设计一个简单的布局，它需要有评论博客及回复评论的功能。而采用相对内边距的方法就能实现该布局。你可能会说："那不可能！你怎么可能仅仅通过内边距来控制页面的布局呢？"那就来看看如何做到吧。

3.3.1　准备工作

当然，相对于静态的 HTML 页面来说，博客页面本身具有很多动态的特性。因此对于你的博客软件来说，本节对博客评论模板部分的设计有一定的帮助。事实上，本方法非常简单，同时又非常高效。现在，从 Ipsum 上面获取一些文本内容，做好为本方法折服的准备。

3.3.2　实现方式

第一步，创建一个简单的具有博客风格的页面，并在 div 元素中嵌入评论相关内容。在页面的主体中，创建类为 content 的 div，用于包含所有页面内容。首先添加一个 h1 标题，然后是一段 Ipsum 生成的文本内容，紧接着就是一个 .comments 元素。而内嵌的评论布局就位于 .comments 元素中。

```
<div class="content">
     <header>Control your layout with relative padding</header>
     <p>
Pellent esque eleifend facilis isodio ac ullam corper. Null amuten
imut massat incident luctus. Utnull alibero, el eifend vel ultrices
at, volut patquis quam...</p>
     <div class="comments">
          <h2>Comments</h2> No 2 x h1
     </div>
</div>
```

在 .comments 的 div 元素中，添加第一条评论。然后，在评论的里面，紧接着评论的闭合标签为第一条评论添加新的评论：

```
<aside>
     <h1>Comments</h1>
     <div class="comment">
          <p>
Pellent esque eleifend facilis isodio ac ullam corper. Null amuten
imut massat incident luctus. Utnull alibero, et...
          </p>
          <div class="comment">
               <p>
Pellent esque eleifend facilis isodio ac ullam corper. Null amuteni
mut massat incident luctus. Ut null alibero, el eifend vel ultrices
at, volut patquis quam...
               </p>
          </div>
     </div>
</aside>
```

按照同样的方式继续下去，就能一直为评论发表评论了。同理也可在父 div 元素之外添加评论内容，这样所评论的对象就是父元素的父元素所代表的评论内容了。以此类推，可以评论博客内容本身：

```
<aside>
    <h1>Comments</h1>
      <div class="comment">
        <p>
          Pellent esque el eifend facilis isodio ac ullam corper..
        </p>

      <div class="comment">
        <p>
           Null amuten imut massat incident luctus....
        </p>

      <div class="comment">
        <p>
          Ut null alibero, el eifend velul trices at, volut pat quis
quam...
        </p>
      </div>
       </div>
      </div>
```

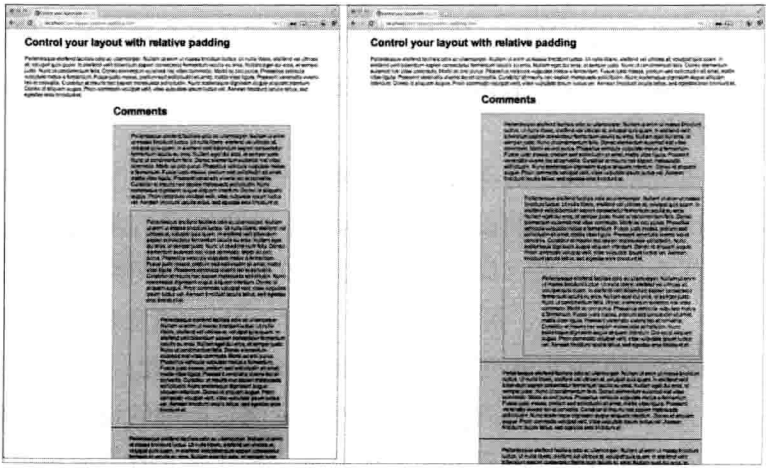
```
    <div class="comment">
        <p>
          Null ameget dui eros, et semper justo. Nun cut condi mentum
felis...
        </p>
      </div>
    </div>

</aside>
```

最终的结果是，只通过最简单的相对内边距方法，就能完成多层次的评论布局，而且得到的布局效果还非常漂亮。

通过 CSS 来完成这样的工作出奇容易。添加一些简单的类：.content、.comments 及 .comment。在 content 类里面再添加一些侧边距，同时 comment 里面设置的左侧内边距稍微多一些。

```
.content {padding:0 5% 0 5%;}
aside {padding:0 10% 0 20%}
.comment {padding:0 0 0 10%}
```

布局效果呈现在下面的截图中。



3.3.3　工作原理

随着页面宽度的变化，相对内边距属性会自适应地调整自身宽度值。

3.4 为 CSS 添加媒介查询

在本节中，我们会一起探究媒介查询的强大之处：渲染的页面能够自适应于所有的设备和所有的可能场景。好吧，我承认有点夸大其词。但是接下来我们确实会创建一个简单的 Web 页面，该页面可适用于不同窗口大小的浏览器、不同的设备以及其他可能的呈现方式。

3.4.1 准备工作

仅仅为了学习本节内容，你就得出去购买在此处所列的每一种设备，例如一台高清电视、一部智能手机、一部功能手机，至少还要有一台打印机。什么，没门？好吧，出于为你减少花费的考虑，我还只是让你买一些最有代表性的设备。这说明在每种设备上面都进行测试媒介查询是不太现实的，因为设备的种类太多了。其实在现实生活中，绝大多数的设备都是不会用到的，同时你也不会在意。接下来我们只是关注于那些平时会经常使用到的媒介查询场景。

这里先跳过那些对你来说不必要的设备。如果你发现需要在那些平时不怎么常见的设备中展现页面，也能非常容易地获取到相关的媒介查询信息。你永远不知道什么时候需要用哪些稀奇古怪的设备！如果需要，可访问 WC3 以获得详尽的信息和描述，网址为 http://www.w3.org/TR/css3-mediaqueries/。在此，就不去关注那些无关问题了，只列举几种具有特定颜色限制的设备：单色显示终端、打印机、电视以及手持设备。此处媒介查询应用得最多的要数 screen 和 print 设置。

3.4.2　实现方式

创建一个 HTML 页面，其中包含一个 h1 的标题和一个 div 元素，div 元素中包含一个 image 元素以及一段文本。如果没有合适的文本内容，可以用 Ipsum 生成一些。页面的源码如下所示：

```
<body>
     <h1>Add Media Query to your CSS</h1>
          <div class="wrap">
               <img src="images/robot.png"/>
Pellent esque el eifend facilisis odio ac ullam corper. Nullam ut enim
ut massa tincidunt luctus…
          </div>
</body>
```

接下来，就需要创建相关的媒介查询了。在下面所述的列表中，会对每一条设置进行一些简短的说明：

```
@media print{...}
```

当 Web 页面处于打印状态时，就会启用该媒介查询。可以通过选择菜单 File | Print 然后查看打印预览进行验证。该媒介查询对于那些需要打印出来的 Web 页面是非常有用的。当然也可以修改或删除一些样式，确保打印版本的页面尽可能简洁。

```
@media (orientation: portrait){...}
```

这条配置信息作用于所有的竖屏显示设备。可以使得移动设备在横竖屏切换的时候呈现出不同的显示效果。有一点需要注意，该设置对于桌面设备同样适用，除非你限定该媒介查询只在较小的屏幕或设备上起作用。媒介查询的另一个方向值就是横屏。

```
@media (height:500px){...}
```

基于 height 及 width 的媒介查询允许设置针对特定大小屏幕的样式。

```
@media (device-width:500px){...}
```

上面的媒介查询会给所有页面施加同样的样式，无论浏览器窗口大小是多少，也就是说，以指定的大小在设备上渲染页面。

```
@media screen and (device-aspect-ratio: 16/9) {...}
```

此处的媒介查询用于对 16/9 比率的设备窗口（非 print 类型）进行样式设置。

```
@media tv {...}
```

该设定选项适用于采用电视作为视频输出的设备。

```
@media screen and (max-width:960px){...}
@media screen and (min-width:961px) and (max-width:1280px){...}
@media screen and (min-width:1281px) and (max-width:1336px){...}
@media screen and (min-width:1336px){...}
```

在媒介查询中，min-width 和 max-width 是最为有用的两个。基于该媒介查询能为任何窗口大小的设备设置响应式样式，这其中也包括那些屏幕很小的移动设备。首先设置最小（即移动）设备的相关样式，接下来设置那些最常见大小屏幕的相关样式，最后通过 min-width 来适配最大尺寸的屏幕。

创建了与项目显示设备相关的媒介查询后，通过给媒介查询设定不同的属性值进行样式的配置：

```
@media tv {
    body {color: blue;}
    h1 {
        font-weight: bold;
        font-size: 140%;
    }
    img {
        float: left;
        width: 20%;
        border: 2px solid #ccc;
        padding: 2%;
        margin: 2%;
    }
    p {
        width: 62%;
        float: right;
```

```
            font-size: 110%;
            padding: 2%;
        }
}
@media screen and (max-width: 960px) {
        body {color: #000;}
        h1 {
            font-weight: bold;
            font-size: 120%;
        }
        img {
            float: right;
            width: 20%;
            border: 2px solid #ccc;
            padding: 1%;
            margin: 1%;
        }
        P {
            width: 80%;
            float: left;
            font-size: 60%;
        }
}
@media screen and (min-width:961px) and (max-width:1280px) {
        body {color: #000000;}
        h1 {
            font-weight: bold;
            font-size: 120%;
        }
        img {
            float: right;
            width: 20%;
            border: 2px solid #ccc;
            padding: 1%;
            margin: 1%;
        }
        P {
            width: 76%;
            float: left;
            font-size: 60%;
        }
}
@media screen and (min-width: 1281px) {
        body {color: #000000;}
        h1 {
            font-weight: bold;
            font-size: 120%;
        }
```

```
        img {
            float: right;
            width: 20%;
            border: 2px solid #ccc;
            padding: 1%;
            margin: 1%;
        }
        p {
            width: 70%;
            float: left;
            font-size: 100%;
        }
}
```

页面的最终版本呈现在下面的截图中。

3.4.3 工作原理

经过这些不同的设置后，就能够发现在不同的设备中，页面呈现出了不同的样式效果。当然，在你的站点中也可以通过组合不同的配置来实现更加多样化的响应式特效。

3.5 基于媒介查询创建响应式宽度布局

在本节中，我们会尝试让一个简单的响应式宽度布局自适应于不同宽度的屏幕。这样的布局方式可作为一个初始模板应用到个人博客或新闻杂志页面。同时这些页面允许读者评论页面内容或者评论他人的评论。当然对于那些喜欢挑起网络舌战的“刺儿头”，这也会是一个他们所喜欢的主题。开个玩笑，请别当真！

3.5.1　准备工作

本节中的模板可用于动态的 CMS(内容管理系统) 或是博客软件中，可别小看它只是一个纯的 HTML 页面。绝大多数模板主题的工作方式与 HTML 的渲染方式是一样的。绝大多数情况下，你只需要使用模板标签替换文本以及静态导航栏就可以应用模板。本节也需要一些填充文本用以展示这些效果。一如之前的章节中所提到的方法一样，没有合适的文本内容，就让随时准备好的 Ipsum 生成器帮我们生成一些文本。

3.5.2　实现方式

开始之前，新建一个简单的 Web 页面，在元素 style 中创建一些媒介查询。你可以使用外部的样式表，但为了演示的简便，本节和本书其他一些章节都会将 CSS 样式放置在 header 元素中的 <style>...</style> 区域。媒介查询包括以下这些标准视图断点的屏幕尺寸：960、1024 及 1280。

```
<style>
@media screen and (max-width: 960px) {…}
@media screen and (min-width: 961px) and (max-width: 1024px) {…}
@media screen and (min-width: 1025px) and (max-width: 1280px) {…}
@media screen and (min-width: 1281px) {…}
</style>
```

此处第一条媒介查询对于所有屏幕宽度小于 960px 的设备生效。第二条作用于屏幕宽度为 961 ~ 1024px 的设备，第三条的作用范围是 1025 ~ 1280px，而最后一条媒介查询的设置会应用到所有屏幕宽度大于 1281px 的设备。对于每一条媒介查询，都需要编写相应的 CSS 以期得到不同的页面布局。当然也会有一些 CSS 布局设置不在媒介查询中，而是与其他呈现样式放在一起，但是绝大多数的布局设置还是需要定义在媒介查询中。

接下来就是创建 HTML 页面的布局了。基本上来说，总是会从那些最基本的 div 元素 nav、content 及 comments 开始：

```
<body>
  <nav></nav>
  <div class="content"></div>
  <aside class="comments"></aside>
</body>
```

然后给页面添加一些文本内容。这有助于更好地展示我们的布局。

在 nav 元素中添加一个无序的列表作为示例菜单链接。它将会成为布局中的响应式菜单。当页面位于最小宽度的状态下时，就会转换为垂直菜单。当页面宽度介于 961px 与 1280px 之间时，菜单则会以水平的布局方式显示在最上方。对于那些更宽的显示设备，我们期待菜单会位于页面的左侧并以垂直的方式呈现给用户。

在前面两个媒介查询中，content 和 comments 元素均会向左侧浮动，只是比例不同。在 960px 的设置中，这些元素的宽度均为 90%。对于更大的屏幕宽度，content 和 comments 元素则分别设置为 60% 和 20%。

```
@media screen and (max-width: 960px) {
     .content {width: 90%;}
     .comments {width: 90%;}
}
@media screen and (min-width: 961px) and (max-width: 1280px) {
     .nav ul li {display: inline-block;}
     .content {width: 60%;}
     .comments {width: 20%;}
@media screen and (min-width: 1281px) {
     .content {width: 60%;}
     .comments {width: 20%;}
}
```

为了确保菜单能够呈现在大尺寸屏幕的左侧，首先通过定位创建一个三列布局。在 min-width:1281px 媒介查询中，添加 .nav 元素并设置绝对定位和宽度样式：

```
.nav{
     position: absolute;
     top: 20px;
     left: 0px;
     width:144px;
}
```

至此，创建一个响应式的布局需要的所有步骤基本都已完成。为了使效果更好，下面再给布局添加一些内边距填充。在另外一条媒介查询中添加 .nav、.content 及 .comments 元素，并设置相应的内边距。请参考下面所列的 CSS。媒介查询 min-width:1281px 的 .nav 元素没有设置内边距，同时 .content 和 .comments 元素的内边距也被进一步缩减以满足垂直菜单的需求。

```
@media screen and (max-width: 960px){
     .nav {padding: 1% 5%;}
     .content,.comments {padding: 1% 5%;}
     .content {width: 90%;}
}
@media screen and (min-width: 961px) and (max-width: 1280px){
     .nav {padding: 1% 5%;}
     .nav ul li {display: inline;}
     .content,.comments {padding: 1% 5%;}
     .content {width: 60%;}
}
@media screen and (min-width: 1281px){
     .nav {
          position: absolute;
          top: 20px;
          left: 0px;
          width: 144px;
     }
     .content,.comments {padding: 1% 1% 1% 0;}
     .content{
          width: 60%;
          margin-left: 144px;
     }
}
```

当然你可以给菜单设置任何你喜欢的样式。在这里只是简单地给 li 元素添加一些外边距。在媒介查询之外设置元素样式，.nav ul li{margin: 2px 10px;}。

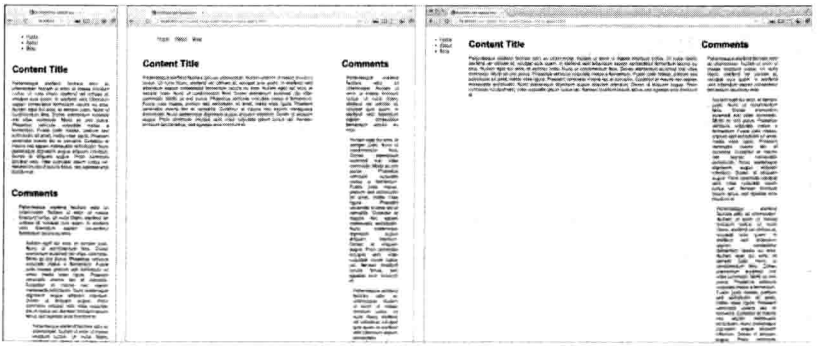
最后，在 .content 元素中粘贴填充文本充当正文及评论。在元素内也添加了 header 及 paragraph 标签。这样就看起来和真正的评论差不多了。

记住，该布局要允许内嵌评论和发表评论的评论。因此页面中的相关评论需要形成具有继承关系的层次结构，而同时要做到在不同的浏览器下都有不错的效果。基于此，内边距是必不可少的。给 .comment 元素添加静态的内边距是无法达到我们想要的效果的，而应给每一个媒介查询中的 .comments 元素设置一个相对的内边距，这样当浏览器窗口变小后，内边距所占宽度也随之缩小：对于 max-width:960px 的媒介查询设置元素宽度为 90%，而其他大于 960px 的媒介查询设置为 20%。在媒介查询之外，设置 .comment 元素 padding-left: 8%，并且设置 .content 和 .comments 元素为左浮动。还可以设置 text-align:justify，使文本内容看起来更像一个整体。

```
@media screen and (max-width: 960px) {
     .nav {padding: 1% 5%;}
     .content,.comments {padding: 1% 5%;}
     .content {width: 90%;}
     .comments {width: 90%;}
}
@media screen and (min-width: 961px) and (max-width: 1280px) {
     .nav {padding: 1% 5%;}
     .nav ul li {display: inline;}
     .content,.comments {padding: 1% 5%;}
     .content {width: 60%;}
     .comments {width: 20%;}
}
@media screen and (min-width: 1281px) {
     .nav {
          position: absolute;
          top: 20px;
          left: 0;
          width: 144px;
     }
    .content,.comments {padding:1% 1% 1% 0}
    .content {
```

```
            width: 60%;
            margin-left: 144px;
        }
        .comments { width: 20%;}
}
.content,.comments {
        float: left;
        text-align: justify;
}
.nav ul li {margin: 2px 10px;}
.comment {padding-left: 8%;}
```

这里的 CSS 设置使页面中评论的内边距能自适应浏览器窗口大小的变化。最终页面中的评论部分呈现出父级与子级的层次，且能够在不同大小的浏览器中保持一致的样式和布局。这些代码的特效就展示在下面的截图中。



3.5.3　工作原理

本节的响应式布局采用的技术和之前的有点不一样。首先，媒介查询提供了虽有限但实用的技术，来实现针对不同浏览器窗口大小的布局效果。其次，那些流式及浮动的元素根据各自的内边距比率自适应于变化中的新布局。最后，采用百分比的流式内边距使得在屏幕大小变化造成布局改变的情况下，始终保持一致的内边距比率。

3.6 基于媒介查询改变图片大小

在本节中，你将学到如何通过 CSS 媒介查询调整图片的大小。很多情况下都会用到这个技术，特别是当你只下载了一张图片，但是又想在不同尺寸的响应式布局中使用它时。

3.6.1 准备工作

在客户端调整图片尺寸是个好方法。但是需要注意的是不能滥用该方法，否则当客户端下载的图片尺寸较大时，会导致调整图片大小变成浏览器的负担。当然也有更好的方法，第 1 章中的相关方法就是如此。

3.6.2 实现方式

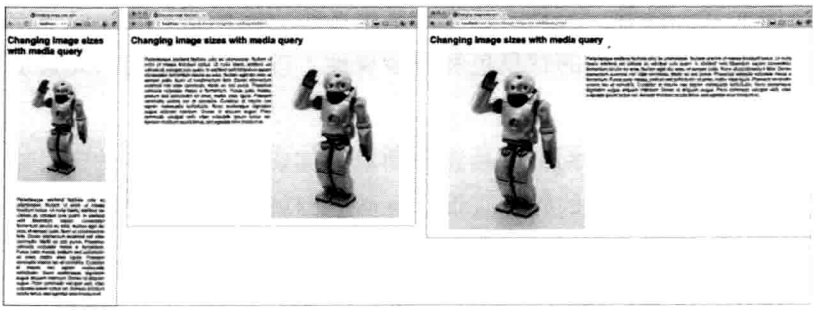
在此需要创建 HTML 页面，页面中包含一个 h1 标题，一个 wrap 元素，并且在 wrap 元素内含有一张图片和一段文本。实际上，这些页面元素对于通过图片查询的方式调整图片大小来说都不是必需的，这样做的目的只是为了更好地演示使用媒介查询调整图像大小的效果。

接下来，为最常见的浏览器窗口的尺寸断点创建媒介查询：960px、1024px、1280px、1366px、1440px 和 1680px。在每个媒介查询中，都需要设置对应元素的样式。在此处的示例中，会创建针对 960px 和 1280px 的媒介查询：

```
@media screen and (max-width: 960px){
     .wrap {padding:0 5%; width: 90%;}
     .wrap img {
          width: 90%;
          height: auto;
          padding:5%;
     }
     .wrap p {
          width: 90%;
          padding: 5%;
```

```
            text-align: justify;
        }
}
@media screen and (min-width: 961px) and (max-width: 1280px) {
        .wrap {
            padding: 0 5%;
            width: 90%;
        }
        .wrap img {
            width: 50%;
            height: auto;
            max-width: 600px;
            float: right;
            }
        .wrap p {
            width: 50%;
            text-align: justify;
            float: left;
        }
}
@media screen and (min-width:1281px) {
        .wrap {
            padding: 0 5%;
            width: 90%;
         }
        .wrap img {
            width: 40%;
            height: auto;
            max-width: 500px;
            float: left;
         }
        .wrap p {
            width: 60%;
            text-align: justify;
            float: right;
         }
}
```

现在可以通过改变浏览器的窗口大小观察本方法是如何工作的。效果呈现在下面的截图中。



3.6.3 工作原理

浏览器根据自身窗口大小应用对应的媒介查询中的相关设置，使元素能够呈现出不同的 width 和 height 属性。这样，在不同尺寸的设备窗口中都得到合适大小的图片。但是如果原始图片非常大，在服务器端改变图片尺寸会是一个好的备选方案。

3.7 基于媒介查询隐藏元素

本节会展示一些有用的小技巧，即根据浏览器窗口的大小，通过媒介查询使页面上的某些元素从屏幕中消失。隐藏元素的方法有很多，本节会探究其中的三个。

3.7.1 准备工作

本节中介绍的方法可应用于很多场景。一个常见的用例就是当页面在较小屏幕的设备上渲染时，菜单会被及时地隐藏。可以利用这点来改变关注的内容及相关区域的显示方式。使用该方法时，你可充分发挥想象力来实现各种效果。

3.7.2 实现方式

需要创建一个简单的页面用于效果展示。在本例中，首先给页面添加了一个 h1 的头，一张图片，然后是两个含有文本信息的页面元素。接着，需要给这些元素添加相应的样式。此处对每一个元素均设置不同的背景色和宽度属性，以便直观地展示元素从页面上消失的效果。

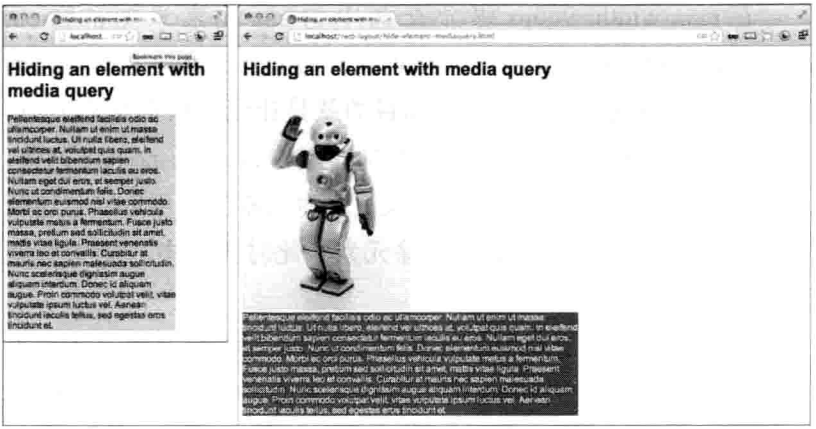
接下来针对不同的屏幕视图断点添加媒介查询。本例中，添加的是针对 960px 的媒介查询。下面我们就将通过媒介查询，看看使得元素消失的一些方法。

在 max-width: 960px 媒介查询中为 img 元素添加 position: absolute 和 left: 5000px 样式。该样式会将 img 元素向左移动远远超出屏幕设备的可见范围，实际目的就是让元素从页面消失掉。同时在该媒介查询中设置 .bar 元素的样式为 display: none。这样，元素的位置并没有变化，只是渲染页面时元素为不可见。上述两种方法都能有效地从页面中移除元素，此时页面中只剩下标题和 .foo 元素。

在第二个媒介查询中，会尝试用另外一种方式从屏幕上移除指定元素。首先，在媒介查询中为 .foo 元素设置一个值为 5000px 的左外边距。这也就使得元素从屏幕上消失了，但需要注意的是，紧接着 .foo 元素的 .bar 元素填补了原来 .foo 元素所占的空间，而在 .foo 元素本来的位置留下了一片明显的空白。然后设置 .foo 元素为左浮动，就会发现白色区域消失了。具体实现参见下面的代码：

```
.foo {
     background-color: #ccc;
     width: 300px;
}
.bar {
     background-color: blue;
     width: 600px;
     color: white;
}
@media screen and (max-width: 960px) {
     img {
          position: absolute;
          left: 5000px;
     }
     .bar {display: none;}
}
@media screen and (min-width: 961px) {
     .foo {
          float: left;
          margin-left: -5000px;
     }
}
```

恭喜你！可以在浏览器中打开该页面看看效果是否如下图所示。



3.7.3 工作原理

无论是绝对定位还是浮动都没有高度属性，因此一旦应用于某个元素，均不会占用任何垂直的空间区域。利用这个有用的小窍门可以很好地在页面上移除相应元素。当然通过浮动元素来布局时也会导致一些小问题。这些问题可以为元素添加 clear:both 属性来解决。

3.8 创建平滑过渡的响应式布局

本节将引导大家创建一个涉及多区域的响应式前端页面。页面中包括许多元素，而这些元素的响应方式又各不相同，最终生成给人印象深刻的页面布局，并能够提供出色的用户体验。我在之前一个项目的启动阶段应用了本方法，发现非常有意思。后来做了进一步的研究，通过本节分享给大家。

3.8.1 准备工作

本方法非常适用于信息量非常大的站点首页。假如你已经准

备了一定的内容，该方法对于登录页面也同样适用。也就是说，只需对本方法稍加修改就可适应于单一内容的页面。如果你碰巧刚刚开始设计你的网站，可以从 http://lipsum.com 获取一些现成的文本信息。本节中的示例文本就是从该网站获取的。

3.8.2　实现方式

站点内容被划分为三个不同的部分，包括两个页面元素及一个页脚。而这两个页面元素布局有时为垂直排列，有时又为左右浮动排列，这取决于页面的宽度。这些元素本身又被划分为更小的元素。因此，开始先创建一个基础页面，该页面含有一个顶层包装元素、一个中层包装元素和一个页脚：

```
<body>
  <header>...</header>
  <div class="content" role="main">...</div>
  <footer>...</footer>
</body>
```

接下来，为这些元素设置 CSS 属性。在下面的媒介查询中添加一些基本的 CSS 设置：

```
body{
     margin: 0;
     padding: 0;
}
footer {width: 100%;}
.clear {clear: both;}
@media screen and (max-width: 1280px) {
     header, .content {width: 100%;}
}
@media screen and (min-width: 1281px) {
     header {
          float: left;
          width: 60%;
     }
     .content {
          float: right;
          width: 40%;
     }
}
```

在此布局中，当页面宽度小于 1280px 时，header 和 .content 均会占据 100% 的页面宽度。当页面宽度大于 1280px，它们分别占据页面宽度的 60% 和 40%，而浮动方式分别为 left 和 right。

接下来，需要创建一些菜单。这里的示例将通过媒介查询来隐藏和显示两个不同的菜单。所以，首先创建所需的两个菜单，然后设置 CSS 以便在不同的屏幕上呈现最合适的菜单。最小尺寸版本将会是一个多选的下拉框式的菜单，而较大尺寸版本的菜单则包含两个内联的列表。HTML 页面中的顶层包装元素代码如下所示：

```
<header>
    <nav>
        <div class="menu small-menu">
            <img src="images/robot-low.png">
            <form>
               <select name="URL" onchange='window.location.
href=this.form.URL.options[this.form.URL.selectedIndex].value'>
                <option value="blog.html">Page 1</option>
                <option value="home.html">Home Page</option>
                <option value="tutorials.html">Tutorials</option>
               </select>
            </form>
        </div>

        <div class="menu large-menu">
            <div class="top-menu">
                <nav>
                  <ul>
                   <li><a href="login.html">Log In</a></li>
                   <li><a href="account.html">My Account</a></li>
                  </ul>
                </nav>
            </div>
        <div class="bottom-menu"> these should be classes so they can
be reused. Plus the names are too specific.
            <nav>
              <a href="#" class="logo">
                 <img src="images/robot-low.png">
              </a>
              <ul>
                 <li><a href="blog.html">Page 1</a></li>
                 <li><a href="home.html">Home Page</a></li>
```

```
                <li><a href="tutorials.html">Tutorials</a></li>
                <li> <a href="news.html">News</a> </li>
            </ul>
        </nav>
    </div>
</div>
</nav>
</header>
```

为 header 元素添加以下 CSS 样式：

```
nav .small-menu img{
    width:9%;
    height:auto;
    float:left;
    padding:0 2%;
}
nav .small-menu select {
    margin: 3%;
    width: 80%;
}
```

接下来通过添加相应的媒介查询来呈现两种不同菜单的效果。添加的媒介查询可以实现在较小的浏览器窗口中显示下拉式的菜单，而在较大的浏览器窗口中显示内联列表菜单。使用 display 属性来实现菜单的显示和隐藏。

```
@media screen and (max-width: 600px) {
    nav .small-menu {display: inline;}
    nav .large-menu {display: none;}
}
@media screen and (min-width: 601px) {
    nav .small-menu {display: none;}
    nav .large-menu {display: inline;}
}
```

在菜单元素下面、结束标签 </header> 之前，添加一张用于展示的高质量图片。为了防止该图片所占空间被浪费，在图片中间放置一个搜索框。搜索框位于图片中间，并能响应屏幕尺寸的变化而自动调节。代码如下：

```
<div class="img-search">classes
  <div class="search">
```

```
        <form>
          <input type="text" placeholder="Find a Robot">
          <input value="Search" class="search-input" type="submit">
        </form>
    </div>
    <img class="main-img" src='images/robot-wide.png'>
</div>
```

当然所有的特效都藏在 CSS 中。我们会使用一些小技巧使得搜索框悬停在同一位置。首先，设置外层 div 元素的宽度为 100%，然后设置 search 元素为绝对定位，并在不同的媒介查询下分别设置其他几个属性。这样的组合使得搜索框能够始终悬浮在 img 区域的中间位置。下面所示的 CSS 代码仅仅只是那些新增的样式，而不包括之前已经存在的属性样式。如果每次都将整个 CSS 代码展开，就有点太冗余了。最后，我会将最终的 CSS 代码呈现出来。

```
.img-search {width: 100%;}
.search {position: absolute; }
.top-menu {
     height: 33px;
     background-color: #ccc;
}
.logo img {height: 87px; float: left;}
.top-menu nav li {display: inline-block;}
.large-menu ul {margin: 0 5px;}
.large-menu li {display: inline;}

@media screen and (max-width: 600px) {
     .search {
          margin-top: 87px;
          left: 22%;}
   }
@media screen and (min-width: 601px) and (max-width: 1280px) {
     .search {
          margin-top: 144px;
          left: 40%;
     }
}
@media screen and (min-width: 1281px) {
     .search {
          margin-top: 144px;
```

```
            left: 22%;
        }
    }
```

这里的 .img-search 图像元素会 100% 动态占据页面宽度，而高度则为自动。也就是说，它是用来作为大图片的搜索框区域。

对于下一个元素 .flip-tab，设定 100% 的宽度值及任意的高度或者其他你想要的属性。同样此处的 CSS 代码也只是新增的样式：

```
<div class="flip-tab"><h3>Look Down Here</h3></div>

.flip-tab {width: 100%; height: 54px; text-align: center;}
```

下一个元素 .teasers 得到属性值 max-width: 1280px，因为其父元素 top-wrap 的宽度最大为 1280px，因此该元素自动地 100% 占据了父元素的宽度。该元素含有三个向左浮动的 .teaser 元素。在以 600px 为分界点的媒介查询条目中，这些 .teaser 元素会得到两种不同的样式。

```
<div class="teasers">
    <div class="teaser teaser1">
        <h3>The First Law of Robotics</h3>
            <p>
                Lorem ipsum dolor sit amet,..
            </p>
    </div>
    <div class="teaser teaser2">
        <h3>The First Law of Robotics</h3>
            <p>
                Lorem ipsum dolor sit amet,..
            </p>
    </div>
    <div class="teaser teaser3">
        <h3>The First Law of Robotics</h3>
            <p>
                Lorem ipsum dolor sit amet,..
            </p>
    </div>
</div>
.teasers {max-width: 1280px;}
.teaser {float: left;}
```

```
@media screen and (max-width: 600px) {
     .teaser {width: 100%;}
}
@media screen and (min-width: 601px) {
     .teaser {
          width: 32%;
          min-width: 144px;
     }
}
```

这些就是 header 元素所需要的全部内容。下面轮到 content 元素了，content 将会把所包装的内容浮动到布局的右侧。在 content 元素中，是两个列式的布局，所占宽度比例为 60/40，当父元素的宽度不够时，每个列所占比例均为 100%。在以 1280px 为分界点的媒介查询中，content 元素同样被设置不同的样式。到目前为止，这些元素都只含有一些简单的示例内容。如果想真正地发布这些页面，你可以添加自己喜欢的内容：

```
<div class="content" role="main">
     <div class="contact-us">

          <div class="form-wrap">
               <legend>Find a Robot</legend>

               <form>
                    <input type="text" placeholder="Robot Search">
                    <input value="Search" class="search-input"
type="submit">
               </form>
          </div>
               <h4>Search or Like Us Locally</h4>
          <ul class="local-like">                    <li><a
href="/search/SanFranciso">San Francisco</a><a href="/like/
SanFrancisco">Like</a></li>
               <li><a href="/search/LosAngeles">Los Angeles</a><a
href="/like/LosAngeles">Like</a></li>
               <li><a href="/search/Austin">Austin</a><a href="/lil
Austin">Like</a></li>
              <li><a href="/search/Houston">Houston</a><a href="/like/
Houston">Like</a></li>          </ul>
     </div>
     <divclass="cities"> really?
          <p>Loremipsumdolor sitamet, consecteturadipiscingelit.
Nunc non felisutmetusvestibulumcondimentumuteueros.Nam id ipsumnibh.
```

```
Praesent sit ametvelit...
          </p>
     </div>

</div>
```

整个 CSS 稍微有点复杂，但可以通过网络获取完整版本。可以看到，这些元素的布局是一个 Z 字形，但是针对不同断点的显示都经过了优化。

```
.contact-us {float: left;}
.cities {float: left;}
@media screen and (max-width: 600px) {
     .contact-us {width: 100%;}
     .cities {width: 100%;}
}
@media screen and (min-width: 601px) and (max-width: 1280px) {
     .contact-us {width: 40%;}
     .cities {width: 60%;}
}
@media screen and (min-width: 1281px) and (max-width: 1366px) {
     .contact-us {width: 100%;}
     .cities {width: 100%;}
}
@media screen and (min-width: 1367px) {
     .contact-us {width: 40%;}
     .cities {width: 60%;}
}
```

最后轮到了页脚！（页面的最底部！）页脚内容全部位于占 100% 宽度的 <footer> 元素中，其中包括一个包装元素 footer-wrap，其宽度同样也为 100%，max-width 为 1280px，除此之外，还有动态的边框和 inline-block 的显示样式。在 footer-wrap 的内部是三个始终含有 display:inline-block 属性的元素。当屏幕显示尺寸较小时，每一个元素的宽度均为 100%，否则它们的宽度都是 33%，并向左浮动，且最小宽度值为 144px：

```
<footer>
     <div class="footer-wrap">
          <div class="footer-1 footer-third">
               <ul>
```

```
                <li><span class=""><a href="#">FaceBook</a></span></li>
                <li><span class=""><a href="#">Google +</a></span></li>
                <li><span class=""><a href="#">Twitter</a></span></li>
                </ul>
            </div>
            <div class="footer-2 footer-third">
                <ul>
                  <li><span class=""><a href="#">Link1</a></span></li>
                  <li><span class=""><a href="#">Privacy Policy</a></
                  span></li>
                  <li><span class=""><a href="#">Terms of Use</a></
                  span></li>
                </ul>
            </div>
            <div class="footer-3 footer-third">
                <ul>
                   <li><span class=""><a href="#">Link1</a></span></li>
                   <li><span class=""><a href="#">Link2</a></span></li>
                   <li><span class=""><a href="#">Link3</a></span></li>
                </ul>
            </div>
    </div>
</footer>

.footer-wrap{
     width: 100%;
     max-width: 1280px;
     margin :0 10%;
     display: inline-block;
}
.footer-third {display: inline-block;}

@media screen and (max-width: 600px) {
     .footer-third {width :100%;}
}
@media screen and (min-width: 601px{
     .footer-third {
          float: left;
          width: 33%;
             min-width: 144px;
     }
}
```

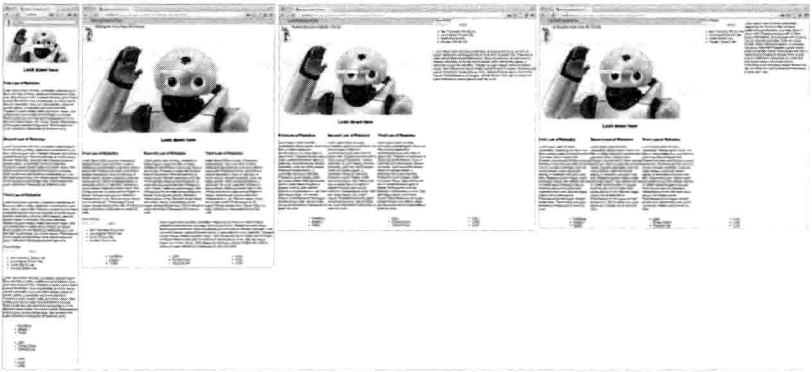
兑现我之前的承诺，将完整的 CSS 代码列在下面：

```
body{margin:0;padding:0;}
.img-search {width: 100%}
.search {position:absolute;}
nav .small-menu img{width:9%;height:auto;float:left;padding:0 2%;}
nav .small-menu select {margin: 3%; width: 80%;}
.main-img {width: 100%; height: auto;}
.top-menu {height: 33px; background-color: #ccc;}
.top-menu nav li {display: inline-block;}
```

```
.logo img {height: 87px; float: left;}
.large-menu ul {margin: 0 5px;}
.large-menu li {display: inline;}

.flip-tab {width: 100%; height: 54px; text-align: center;}
.teasers {max-width: 1280px;}
.teaser {float:left;}
.contact-us {float:left;}
.cities {float:left;}

footer {width:100%}
.footer-wrap {width: 100%; max-width: 1280px; margin: 0 10%; display:
inline-block;}
.footer-third {display:inline-block;}


@media screen and (max-width: 600px) {
 nav .small-menu {display: inline}
 nav .large-menu {display: none}
 .search {margin-top: 87px; left: 22%;}
 .teaser {width: 100%}
 .contact-us {width: 100%;}
 .cities {width: 100%}
 .footer-third {width: 100%}
}
@media screen and (min-width: 601px) and (max-width: 1280px){
     .search {margin-top: 144px; left: 40%}
     .contact-us {width: 40%;}
     .cities {width: 60%}
}
@media screen and (min-width: 601px) {
 nav .small-menu{display: none}
 nav .large-menu{display: inline}
 .teaser {width: 32%; min-width: 144px;}
 .footer-third {float: left; width: 33%; min-width: 144px;}
}
@media screen and (max-width: 1280px) {
 header, .content {width: 100%;}
}
@media screen and (min-width: 1281px) {
 header {float: left; width: 60%;}
 .content {float: right; width: 40%;}
 .search {margin-top: 144px; left:22%;}
}
@media screen and (min-width: 1281px) and (max-width: 1366px){
 .contact-us {width: 100%}
 .cities {width:100%}
 }
```

```
@media screen and (min-width: 1367px) {
.contact-us {width: 40%}
.cities {width: 60%}
}
```

本方法有点长且晦涩难懂，非常感谢你能坚持到这里！特效呈现在下面的截图中，你可以拿它和你自己的页面作一个对比。



3.8.3 工作原理

本节将 CSS 与媒介查询结合在一起，使得页脚在所有的尺寸屏幕中都能居中显示，同时在移动设备上又能够收为一列。

在网页开发方法论中，响应式布局绝对是一个产生惊喜的新领域。响应式设计使得设计者和开发者能够构建适用于不同设备，尤其是移动设备的网页，而省去了开发原生 App 的花费。也许现在还不到时候，但是很快你就会看到越来越多的公司会尝试通过响应式的方式来重新设计他们的站点。

3.8.4 更多内容

只是通过 CSS 实现这个非常简单的响应式方法是不够的，我会要求你再进一步，尝试去消除在 nav 元素中出现的重复菜单选项。参考一下 5.10 节，在移动浏览器上使用 jQuery 方法替换

<select> 元素中的大型菜单选项。这样能够防止因为菜单含有重复内容而遭受搜索引擎惩罚的潜在问题。

首先，剪切 smallMenu 元素及其子元素，然后将其粘贴在 header 中的某个位置或是 body 的最顶部，在 <script></script> 的作用域内定义一个 smallMenu 变量。

```
var smallMenu = '<div class="menu small-menu">…</div>'
```

接下来添加一些新的脚本，调用后能够移除 large-menu 元素，同时在 nav 元素下附加 smallMenu 元素变量。

```
$(document).ready(function() {
    $('.large-menu').remove();
    $('nav').append(smallMenu);
});
```

现在，如果通过移动设备加载页面，脚本生效后会使用缩减版本的导航栏替换原始版本，同时不用担心 SEO 问题。

第 4 章
使用响应式框架

4.1　简介

在布局设计和开发中布局框架越来越有用，使用得也越来越广泛。很多 Web 开发工程师已经发现，借助于框架来实现自己的设计能够大幅度提高产品开发效率。

目前网络上存在不少好用的框架。但刚开始最好多花些时间挑选契合自己的设计的框架，否则一款不适合的框架会产生事倍功半的效果。至少，我认为要先这样做。在实践中，我发现学习和使用框架能够让我更专注于项目中自己感兴趣的部分，并帮助我更快地完成这个项目。本质上说，在项目中使用框架可能会导致最终项目像这个框架。有时这并不是个严重的问题，至少它供你支配，帮助你又快又好地开发网站。当前有很多框架可供使用，有些只是骨架，需要花费更多的时间进行设计和开发，但是你对最终产品有更多控制权。而有些框架则提供了更多的功能，但是这些框架会限制你的设计，除非整个重新设计，否则很难摆脱它的束缚。

那么，哪款框架最适合你呢？答案当然是最契合当前项目需求的那款。我的建议是了解并尝试本章中介绍的框架，从而选择出你所需要的框架。

4.2　使用流式 960 网格布局

960 网格系统已经存在了一段时间，并且已经证明可以被快速应用到新项目中进行使用。该框架很简单易学，只需经历快速的学习曲线后就可以上手了。

960 网格系统的唯一问题是它不是响应式的。事实上，它更像一个表格，其列跨越固定宽度的表头。它在 960px 宽的窗口中布局最完美，这就是它的特点，而你不得不忍受良好的布局只能

呈现在一个特定窗口尺寸中。那为什么还要在一本关于响应式设计的书中讨论这个 960 网格框架呢？答案是一些人很喜欢使用这个网格系统，他们决定解决这个只能适应单一窗口尺寸的问题。

4.2.1 准备工作

有很多好的方案可以解决这个问题，希望你能在本章中发现它们。慢慢来，本节中会介绍这些方法中最简单的一个。实际上，简单的响应式 960 网格系统可以描述为一个流式网格。它使用了百分比宽度、左浮动元素替代了固定宽度的网格元素。该版本大多数情况都能良好工作，但是当列变得很狭窄时，阅读会变得很困难。可以使用额外的 CSS 来轻松解决这个问题。

我们最终希望页面能够根据屏幕改变而自适应，而这需要对网格更细粒度的控制。

首先，从 http://www.designinfluences.com/fluid960gs/ 找到流式 960 网格系统。然后下载并解压缩存档文件。从存档文件的 CSS 文件夹中将 grid.css 文件复制到项目的 CSS 文件夹中。接下来，在项目的 CSS 目录中创建一个名为 responsive.css 的新 CSS 文件。稍后将会使用该文件。

4.2.2 实现方式

在 IDE 中创建一个新的 HTML 文件。添加对 grid.css 及 responsive.css 样式文件的引用链接。

```
<link rel="stylesheet" href="css/grid.css" media="screen" />
<link rel="stylesheet" href="css/responsive.css" media="screen" />
```

接下来需要在 HTML body 元素中添加一些内容。为了使流式 960 网格能够工作，需要添加一个 div 元素，为其添加一个类用于定义拥有的列数量。本节中使用 containter_16 类，总共有 16 个可

用的列。当然也可以分配 containter_12 类给该 div 元素，这样可用列的数量就会变为 12。

在 container_16 元素中，先为标题创建一个容器，即创建一个新的 div 元素并为其分配 grid_16 类。你可能会猜 grid_16 类占据 container_16 元素的全部宽度。这是个相当不错的猜测，你对了 98%。它实际上占据了 98% 的宽度，总宽度是全部 16 列再加 2% 的外边距。如果你改用 grid_11 类，它将占据 11 列，即 66.75% 宽度，另外有 2% 宽度用于外边距。

我们再添加一个 div 元素来创建一个新行，这次设置 class 值 clear。这个 div 元素的效果类似于在键盘上按下回车键，或者在一些编程语言中的换行符 (\n)。在行间添加 clear 元素可以对不同的行进行分隔，因为 clear 元素的位置被设置为 left:float 属性，并不占用垂直空间。

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

也可以用一个简单的换行标签来实现相同的效果，如下所示：

```
<br class="clear">
```

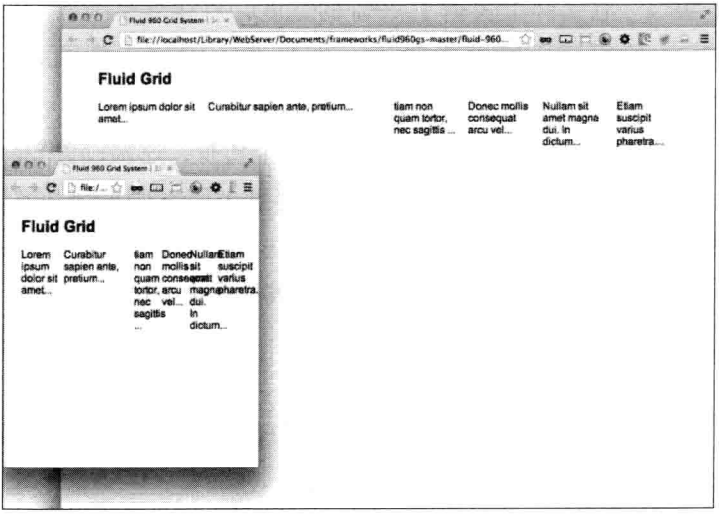
只需在每行之间添加 clear div 元素或换行标签就可实现换行效果。

现在把注意力放在内容上。在 clear 元素下添加 6 个新的 div 元素。给第一个元素分配 class 值 grid_3，第二个元素分配 grid_5，其余的分配 grid_2。顺序无所谓，直到 grid_* 数字总和为 16。在这些 div 元素中添加一些 Insum 填充文本（http://lipsum.com）。代码如下：

```
<div class="container_16">
<div class="grid_16">
<h2>Fluid Grid</h2>
</div>
<div class="clear"></div>
<div class="grid_3">Loremipsum dolor sit amet...</div>
```

```
<div class="grid_5">Curabitursapien ante, pretium...</div>
<div class="grid_2">tiam quam tortor, necsagittis ...</div>
<div class="grid_2">Donecmollisconsequatarcuvel...</div>
<div class="grid_2">Nullam sit amet magna dui. In dictum...</div>
<div class="grid_2">Etiamsuscipitvariuspharetra...</div>
</div>
```

在下面的截图中你可以看到流式网格如何适应较小的视窗。



下一步是设置 CSS，给流式布局添加一些响应式特性。在 IDE 中打开 responsive.css 文件进行编辑。添加媒介查询来覆盖 1024px、600px、420px 这几个屏幕断点，代码如下：

```
@media screen and (max-width:420px){...}
@media screen and (max-width:600px) and (min-width:421px){...}
@media screen and (max-width:1024px) and (min-width:601px){...}
```

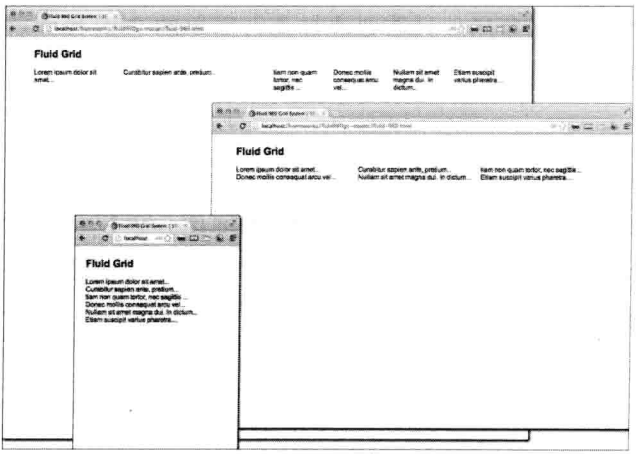
我们的目的是使用新的 CSS 来重写流式网格的样式，为内容元素创建一些更友好的新断点。对于较窄的宽度要设置更大的百分比宽度或一个固定宽度。添加一个新的类 .break-column 到媒介查询中来重写现有的样式。

接下来在 max-width:420px 媒介查询中，为 .break-column 类

添加样式 min-width:360px。然后在 max-width:600px 及 min-width:421px 的媒介查询中，为 .grid_2.break-column、.grid_3.break-column 和 .grid_5.break-column 类设置属性值 width:48%。然后在 max-width:1024px 媒介查询中，添加属性为 width:30% 的类，并紧跟着一个 !important 的重载（确保将其插入在分号前），如下面的代码所示：

```
@media screen and (max-width:420px){
   .break-column{min-width:360px;}
}
@media screen and (min-width:421px) and (max-width:600px){
   .grid_2.break-column, .grid_3.break-column, .grid_5.break-
column{width:48%;}
}
@media screen and (max-width:1024px) and (min-width:601px){
   .break-column{width:30% !important;}
}
```

创建响应式的流式网格只差最后一步了！再次打开 HTML 文件，给 6 个 div 元素都添加值为 break-column 的类。然后本方法就大功告成。你可以刷新浏览器或打开 HTML 查看效果。当缩小浏览器窗口或在移动设备上查看该 HTML 文件时，将会看到针对较小视图进行了优化的响应式布局。效果参见以下截图。



4.2.3 工作原理

如果在浏览器中打开使用了非响应式流式网格的 HTML 文件，当浏览器窗口或设备变小时，页面中的 6 列会保持相同比例。当在小窗口或移动设备中查看时，将会显示 6 个可读性不好的窄列。

通过添加媒介查询来重载 div 元素的样式属性，从而得到响应式效果。本节总共演示了三种方法来实现重载。第一种，使用 min-width 方法重载百分比宽度；第二种，在 grid.css 文件之后加载 responsive.css 文件，并且 responsive.css 中的 CSS 使用了显式命名空间（.grid_2.break-column、.grid_3.break-column 和 .grid_5.break-column），这重载了 grid.css 文件中声明的流式宽度；最后一种，使用 !important 声明提升重载等级。

4.3 使用 Blueprint 网格布局

Blueprint CSS 框架是另一款广受欢迎的静态 CSS 网格系统。在某些情况下可能需要将静态的 Blueprint CSS 网格框架定制为自己需要的响应式 Blueprint 框架。该 Blueprint CSS 框架很容易修改为一个响应式布局。只需要添加几行简单的 CSS 代码就能拥有一个响应式框架。

4.3.1 准备工作

首先需要获取这个 Blueprint CSS 框架。可以从 http://www.blueprintcss.org/ 下载。这个框架的工作方式与其他静态的 CSS 网格框架很相似。

4.3.2　实现方式

下载完毕以后，解压文件并将 blueprint 文件夹复制到项目中的 css 目录。接下来将创建一个 HTML 文件来使用该框架。在 IDE 中新建一个 HTML 文件，在 body 元素中添加一个标题和一个 hr 元素。你可能会问："hr 元素是什么？"它是一条水平线，在这里用作主题分隔符。

在 HTML 的早前版本中，hr 是一条水平线。这条水平线跨越整个页面，起到了分隔符的作用。在 HTML5 中它被升级为主题分隔符。那与之前有什么不同呢？本身而言还是一条横跨页面的水平线，但是过去它主要用于定义布局，现在则用于强调主题或内容的变化。

然而，在 Blueprint CSS 框架中，hr 元素专门用于建立一个新行。让我们继续手头的工作。

在 hr 元素以后，可以新起一行内容。首先给第一行创建一个三列布局。然后插入一些通过 Ipsum（http://Ipsum.com）生成的文本到这三个 div 元素中。给每个 div 元素分配一个类，其值取决于你想让该元素跨多少列，这有点像 960 网格框架中的表格 colspan。在此总列数是 22，前三列所对应的类分别是 span-7、span-8 及 span-7。最后再加上一个主题分隔符：

```
<h1>Blueprint CSS Framework Responsive<h2>
<hr>
  <div class="span-7">Loremipsum dolor sit amet,
   consecteturadipiscingelit...</div>
  <div class="span-8">Etiamegettortorlectus, et
   variusnibh...</div>
  <div class="span-7">Duis sit
   ametfelislobortisfeliscommodolacinia...</div>
<hr>
```

在第二行中，添加两个宽列。即添加两个 div 元素，分别设

置类为 span-15 和 span-17。在左边的 div 元素中插入一段使用 Ipsum 生成的文本段落及一张图片。右边的 div 元素中添加一个文本短语的无序列表。最后再添加一条水平线来结束该行：

```
<hr />
<div class="span-15">
    <img src="test.jpg" class="top pull-1 left" alt="test">
    <p>Loremipsum dolor sit amet, consectetueradipiscingelit...</p>
</div>
<div class="span-7">
<ul>
<li>Loremipsum dolor sit amet, consectetueradipiscingelit...</li>
<li>Loremipsum dolor sit amet, consectetueradipiscingelit...</li>
<li>Loremipsum dolor sit amet, consectetueradipiscingelit...</li>
</ul>
</div>
<hr />
```

这是本节需要使用的大部分 HTML。如果想要添加更多的内容，可以参考下载的文件中 tests 目录下的 sample.html 文件。

在 HTML 的 header 标签中添加对 css/Blueprint/ 目录中存放的 Blueprint CSS 框架的样式表链接。

接下来添加自定义的样式表来使这个框架变为一个响应式框架。在 header 标签中添加对新样式文件 responsive.css 的链接。如果还未创建该样式文件，请立即创建一个：

```
<link rel="stylesheet" href="css/responsive.css"  >
```

打开 responsive.css 样式表，为最小及次小的屏幕断点创建媒介查询。媒介查询的断点为 600px 和 1024px，如下代码所示：

```
@media screen and (max-width:600px) {...}
@media screen and (min-width:601px) and (max-width:1024px) {...}
```

在媒介查询内要使用一个叫做属性选择器的 CSS 特性，其功能有点像通配符 *。为了能够将样式应用到 Blue CSS 网格所有含有 span 类的列（如 span-1,span-2,span3,…），应该这样写：div[class*='span']{...}。这是个很实用的技巧来重载 CSS 网格的样

式，使其变为响应式。

在 600px 媒介查询中，使用属性选择器添加 CSS，设置宽度为 90%。当浏览器窗口宽度小于 600px 时，所有 span 元素宽度将会自动扩展到 100%。同样，在 1024px 媒介查询中设置宽度为 42%。如果你认为像 100%、50% 这样经过四舍五入的百分比数字效果更好，事实会证明你是错的。请记住 Blueprint CSS 框会自动添加内边距。

```
@media screen and (max-width:600px){
   div[class*='span-']{width:90%;}
}
@media screen and (min-width:601px) and (max-width:1024px){
   div[class*='span-']{width:42%;}
}
```

使用浏览器打开该 HTML 文件或者刷新屏幕，将看到当你改变了浏览器宽度时，span 元素会自动调整为新的宽度。

你可能已经注意到了如果屏幕宽度达到 1024px 断点，第二行会留下很多空白。现在来解决这个问题。复制 1024px 媒介查询中的属性选择器样式代码并粘贴到源 CSS 样式代码下方。给属性选择器追加一个 .wide 类，设置宽度为 90%。

在 HTML 文件中给紧跟着第二个主题分隔符（hr 元素）的第一个 span 元素（包含一张图片的那个）添加类 wide。

这个功能兼容大多数的现代浏览器，但是在老版本的浏览器中还不行。还需要多添加一些 CSS 配置来使其兼容更多的浏览器。在 responsive.css 文件的第一行添加类 .container，设置 width 属性为 960px。然后在每个媒介查询中都添加该类，但是 width 属性为 100%。

```
.container{width:960px}
@media screen and (max-width:600px){
     div[class*='span-']{width:90%;}
     .container{width:100%}
}
```

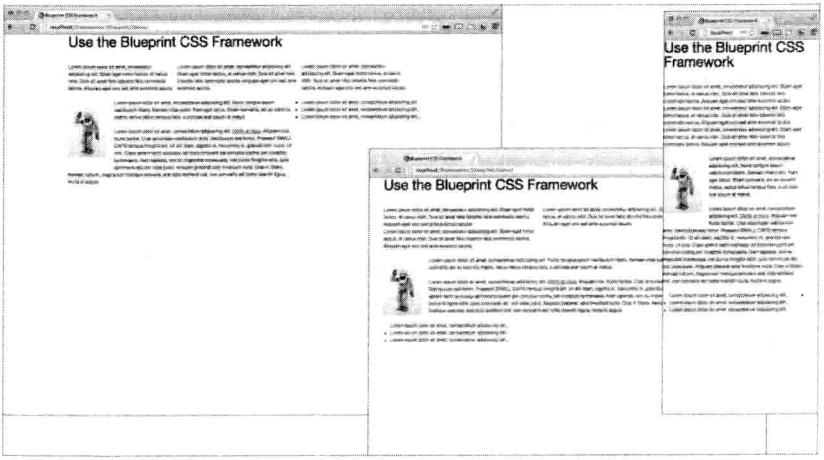
```
@media screen and (min-width:601px) and (max-width:1024px){
     div[class*='span-']{width:42%;}
     div[class*='span-'].wide{width:90%;}
     .container{width:100%}
}
```

这将兼容不支持媒介查询的老版本浏览器。

为了让效果更佳，可以给 span 元素添加一些 CSS3 渐变效果。这会给受影响的 span 元素宽度赋予平滑的渐变动画效果。将以下 CSS 添加在媒介查询外。

```
div[class*='span-']{

-moz-transition: width 0.1s; /* Firefox 4 */
-webkit-transition: width 0.1s; /* Safari and Chrome */
-o-transition: width 0.1s; /* Opera */
transition: width 0.1s;

}
```

可以在每个媒介查询中添加类似于这样好玩的设计来达到更炫目的响应式效果。以下截图显示了新的响应式 Blueprint 框架效果。



4.3.3　工作原理

为了让 Blueprint CSS 框架具备响应式效果，首先将其容器宽度由固定宽度修改为流式最大宽度，然后设置一些媒介查询断点。本节的核心部分是使用属性选择器在 CSS 中利用通配符批量修改 span 属性，从而避免了逐个修改的不便。

4.4　基于三分法的流式布局

三分法（Rule of thirds）是一种设计方法论，即一个布局或图片如果从水平方向或垂直方向被划分为三个区域，将会变得更吸引人。就像与互网有关的其他定律一样，关于三分法也有无尽的讨论和争议。本书的目的只是如何利用此定律。

搜索基于三分法实现响应式及流式布局，没有任何结果，至少我看到的是这样。然而，其实有一个出色的基于三分法的静态框架，它的名字是 Golden Grid。

4.4.1　准备工作

搜索关键词 Golden Grid，第一个结果应该是 http://code.google.com/p/the-golden-grid/。单击页面上方的导航栏，到达下载页面并下载最新版本。

4.4.2　实现方式

解压缩后可以看到在 css 目录下有个 golden-base 目录。复制 golden-base 目录中的 golden.css 文件到自己项目的开发目录。我们将使用这个 CSS 文件作为布局的基础框架。

创建一个新的 HTML 文件，添加对 golden.css 样式表的链接。

```
<link rel="stylesheet" href="CSS/golden.css" media="screen,
projection">
```

打开 golden.css 文件，编辑其中的 .main 类属性，修改 width:970px 为 max-width:970px。这将破坏静态页面模板，并且允许页面最外部容器随着浏览器窗口收缩而自动调整宽度。

可以打开 golden.css 样式表，看下它的工作原理。其实非常简单，里面有三条竖线，每条将页面布局一分为二，然后再一分为二。span 类元素的宽度起始值为 70px，每次递增 80px，直到占满属性 width:950px;。给元素分配 width 属性时，分配类为起始字母 g 加上宽度和 10px 的外边距。同时还要赋予元素 float:left; 及 display:inline; 样式，因为这些元素都是左浮动内联元素，当它们超出水平空间时，将会另起一个新行。因为这些元素是左浮动的，所以默认是左对齐的。你可以在元素之前放置一个空元素来将该元素移动到右边，或者使用框架的 .margin 类在该元素之前设置一个左边距。

外边距有点像网格 span 的宽度属性，每次递增 80px，与网格 span 的唯一不同是外边距从 90px 开始增长，而不是 70px。之所以会有这个不同点，是因为考虑了元素的 margin-left:10px 属性。

框架中的元素按行排列，就像本章中的其他框架一样，它也在新行开始之前使用一个元素来做分隔。该框架使用具有 clear:both 属性的 div 元素来实现。

回到 HTML 文件来，使用三分法创建一个响应式布局。首先来创建一个静态布局。创建一个标题（H1）并赋予其样式 width: 100%，然后添加 3 个 div 元素来创建 3 个新行。

```
<body>
<div class="g960"><h1>Golden Grid CSS Layout</h1></div>
   <div class="clear"></div>
   <div class="clear"></div>
```

```
    <div class="clear"></div>
</body>
```

在第一个 div 分隔元素后，添加一个 div 元素并设置类值为 g960，然后插入一张大图片用来创建响应式特效。可以回顾 1.2 节来实现响应式图片。

```
<div class="clear"></div>
<div class="g960">
<img src="robot-large.png" class="resp" alt="robot picture"/>
</div>
<div class="clear"></div>
```

在下一行，插入 6 个 div 元素，每个设置类 g160。在每个 div 元素中插入一段由 Ipsum 生成的文本。如果想要更翔实的示例，可以将 g160 元素中的其中一个类改为 g80(80px 宽)。记得给该元素也要添加类 m180，用于设置外边距，如下所示：

```
<div class="clear"></div>
<div class="g160"><p>Loremipsum dolor sit amet...</p></div>
<div class="g160"><p>Loremipsum dolor sit amet...</p></div>
<div class="g160"><p>Loremipsum dolor sit amet...</p></div>
<div class="g160"><p>Loremipsum dolor sit amet...</p></div>
<div class="g80 m180"><p>Loremipsum dolor sit amet...</p></div>
<div class="g160"><p>Loremipsum dolor sit amet...</p></div>
<div class="clear"></div>
```

这些 HTML 足以清晰地演示本方法的工作原理。接下来通过添加一些自定义 CSS 来实现响应式设计。

在项目的 css 目录新加一个 CSS 文件 responsive.css，在 HTML 文件 head 标签中添加对该样式表的链接。

```
<link rel="stylesheet" href="CSS/responsive.css" media="screen,
projection">
```

现在将添加一些 CSS 属性来实现响应式设计。首先处理那些大图片。需要做到当浏览器缩小时图片随之缩小。

```
.resp{
    width:100%;
```

```
    height:auto;
}
```

接下来，给 600px 及 1024px 这两个断点添加媒介查询，600px 是针对移动设备，而 1024px 是针对平板设备。还可以针对大屏幕设备添加更多媒介查询，但是这里我们只涵盖基本情况。

```
@media screen and (max-width:600px){...}
@media screen and (min-width:601px) and (max-width:1024px){...}
```

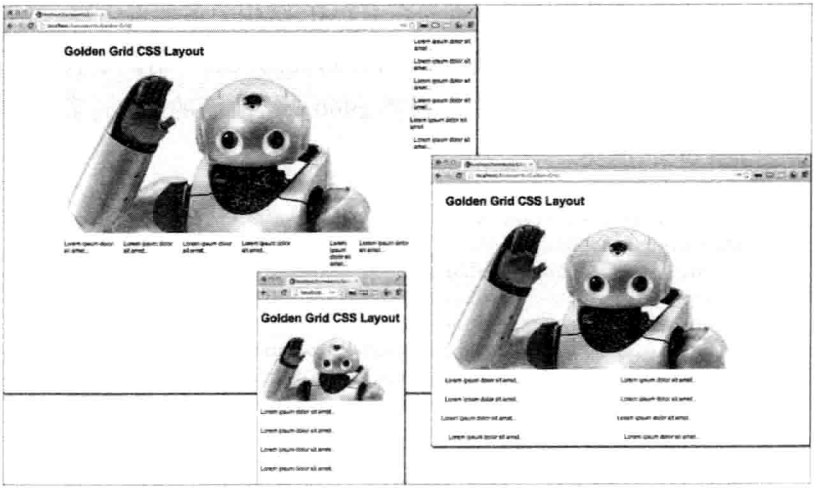
对于宽度小于 600px 的屏幕，需要做到所有 div 元素默认占满整个屏幕宽度。不要忘记有 left-margin 属性的类，在该场景下将其设置为 0。为了使新的 CSS 代码更简洁，可以使用 CSS 属性选择器来通配选择所有网格类。新加样式 div[class*='g']{...}，设置宽度为 90%，再加样式 div[class*='ml'] {...} 并设置属性 margin-left 为 0。

```
@media screen and (max-width:600px){
   div[class*='g']{width:96%;}
   div[class*='ml']{margin-left:0;}
}
```

如果屏幕范围为 600 ~ 1024px，设置相同的样式，不同的是网格类的宽度为 48%。对于该媒介查询，我们不想让每个元素都缩放到屏幕的一半。因为这将抹杀响应式框架效果。所以在属性选择器之后，添加类 .wide 并设置宽度为 96%。在 HTML 中，给标题及图片父元素（即拥有类 g960 的元素）添加类 wide。

```
div[class*='g'].wide{width:96%;}
```

下面的截图展示了 Golden Grid 的效果。



4.4.3　工作原理

属性选择器是一个非常简洁实用的小技巧，可以使一成不变的框架的有限列宽变为横跨整个屏幕宽度。再结合自定义的媒介查询，使得 HTML 适配较小屏幕，就可以得到一个适应所有尺寸的响应式的、可视化的、出色的布局。同样的技术也可应用到其他框架中。

4.4.4　更多内容

让我们再扩展一下该框架。到目前为止，通过本章所涉及的方法已经可以让很多静态框架运行在移动设备上。接下来做个实验，让 Golden Grid 在大屏幕上显示得更好看。给 1280px 断点添加一个新的媒介查询。

```
@media screen and (min-width:1280px){...}
```

作为本方法的扩展部分，本节将使用属性选择器的高级功能。看到 CSS 中的基础逻辑你可能会有些不安，但请保持耐心，你将

掌握一些非常有用的技巧。首先在 HTML 结构中多加点内容。

复制 HTML 文件中的最后一行并追加到最后一行的后面。给新行添加一个父 div 元素并设置类 g960。给前面的 div 元素添加值为 last clear 的类。

```
<div class="last clear"></div>
<div class="g960">
   <div class="g160"><p>Loremipsum dolor sit amet...</p></div>
   <div class="g160"><p>Loremipsum dolor sit amet...</p></div>
   <div class="g160"><p>Loremipsum dolor sit amet...</p></div>
   <div class="g160"><p>Loremipsum dolor sit amet...</p></div>
   <div class="g80 ml80"><p>Loremipsum dolor sit amet...</p></div>
   <div class="g160"><p>Loremipsum dolor sit amet...</p></div>
</div>
```

回到 CSS 文件中，属性选择器目前支持更多条件选择，如父元素、子元素及优先级。现在使用这些特性来应用 CSS 属性到 .last div 元素之后的元素。实现该功能需要符号 ~。该语法如下所示：

```
DIV.preceding~DIV.following
```

我们想让元素在屏幕大于 1280px 时变为右边的一列，从而最大化可视区域。

```
div.last~div[class*='g']{position:absolute;right:0;top:0;width:14%;m
ax-width:226px;}
```

接下来，我们希望其所有子元素排成一排并占满所有可用空间，并使用类 ml 移除所有外边距。该语法与之前的语法很相似，但是符号为 >，写法为 DIV.parent>DIV.child。

```
div.last~div[class*='g']>div[class*='g']{display:block;float:none;wid
th:100%;}
div.last~div[class*='g']>div[class*='ml']{margin-left:0;}
```

我们也需要防止 g960 网格元素受 max-width:1024px 媒介查询中的通配符影响。给 .lost div 元素之后的网格元素使用相同的

属性选择器，设置宽度为 100%，代码如下：

```
div.last~div[class*='g']{width:100%}
```

现在刷新浏览器并让窗口尺寸大于 1280px。你将看到最后一行移动到了侧边栏位置。谁说框架太过僵化很难变成响应式的呢？

目前你已经知道旧版浏览器并不支持媒介查询，由于我们要关心所有用户的感受，因此要给旧版浏览器的坚定支持者多一些关爱。复制 1280px 媒介查询中的 CSS，并添加到只会在 IE9 之前版本中使用的样式表中。然后在 header 标签中给该样式表添加一个条件链接：

```
<!--[if lt IE 9]>
  <link rel="stylesheet" type="text/css" href="IE8.css" />
<![endif]-->
```

这将解决对老版浏览器的兼容问题，而你的网站在老版本浏览器中仍然能正常显示。

4.5　响应式 960 网格框架——Gumby

Gumby 框架是在老版且可靠的静态 960 网格框架的基础上开发出的响应式框架，由 Digital Surgeons 公司的员工开发。Gumby 自身有很多更新，包括添加了一些非常出色的特性。它拥有太多的功能，我们没时间一一介绍，本节只关注改进的布局结构。

4.5.1　准备工作

先访问 Gumby 960 响应式框架的网站 gumbyframework.com/。当浏览该网站时，你会看到新版框架相关特性的实际应用。其布局能够很好地展现在 767px 的移动设备上，同时将菜单变换为更加实用的移动导航栏。该框架包含多个实用的 UI 元素，需要花时间来熟悉对它们的使用。

单击导航栏上高亮的 Download Gumby 2 按钮来获取 Gumby 主版本文件。压缩包中有用来帮助设计布局的 Photoshop 文件、CSS 框架文件、JavaScript 文件、图片文件以及示例 HTML 文件。可以通过 demo.html 文件来学习如何使用该框架。

把这些文件放到一边，我们先创建一个 HTML 页面。

4.5.2 实现方式

在 HTML 编辑器中创建一个新的 HTML 文件。该框架提供了一个快捷的方法来导入自定义的 CSS 脚本，即在 HTML 中添加对 css/imports.css 文件的链接。该文件导入了不同的样式表。这是很实用的技巧，万一你以后需要修改或添加样式表，只需管理这个 CSS 文件即可。

```
<link rel="stylesheet" href="css/imports.css">
```

如下是导入的 CSS 样式表的示例：

```
@import url('gumby.hybrid.css');
@import url('ui.css');
@import url('style.css');
@import url('text.css');
```

请不要忘记在页面底部 body 结束标签之前直接加入 jQuery 库的链接，以及 JavaScript 文件：gumby.min.js、plugins.js 及 main.js。稍后将会使用到这些文件。

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.
min.js"></script>
<script src="js/libs/gumby.min.js"></script>
<script src="js/plugins.js"></script>
<script src="js/main.js"></script>
</body>
```

现在基本框架已经完成，接下来编写一些 HTML 代码。Gumby 响应式网格框架支持 12 列或 16 列布局。我们先创建一个

基于 12 列的布局，然后再在同一个页面追加一个 16 列布局的区域。

在 HTML 主体中添加一个 div 元素，赋予类 container。对于 container 类元素，默认的布局是 12 列。接下来，在这个 div 元素中添加一个含有 row 类的 div 元素。row 类元素会占据整个 12 列的宽度。在每一行中，有 12 列可以使用，用以容纳 div 元素内容。

在拥有 row 类的元素中添加三个 div 元素，分别设置类为 four columns、three columns 及 five columns。column 类前可使用任何你实际需要的数字，只要数字之和等于 12 即可。该类中的数字决定该元素占用多少个列宽。在每个元素中添加一些 Ipsum 填充文本（http://ipsum.com），用于演示布局效果。

```
<div class="container">
        <div class="row">
<div class="four columns"><p>Loremipsum dolor sit amet,
consecteturadipiscingelit. …</p></div>
<div class="three columns"><p>Loremipsum dolor sit amet,
consecteturadipiscingelit. …</p></div>
<div class="five columns"><p>Loremipsum dolor sit amet,
consecteturadipiscingelit. …</p></div>
        </div>
</div>
```

现在在浏览器中打开该页面看看它的外观。测试它的响应式特性，查看它在较小的屏幕中的表现。columns 类的样式如下所示：

```
.column, .columns {
margin-left: 2.127663%;
float: left;
min-height: 1px;
position: relative;
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
box-sizing: border-box;
}
```

class number 样式如下所示：

```
.row .two.columns {
width: 14.893641%;
}
.row .three.columns {
width: 23.404293%;
}
.row .four.columns {
width: 31.914945%;
}
.row .five.columns {
width: 40.425597%;
}
....
And so on.
```

正如你所见，columns 类设置元素位置为相对位置，并浮动元素到左边，还包含内边距及其他样式设置。

接下来添加另一个拥有 row 类的 div 元素。在该元素内，添加 6 个较小的 div 元素构成一行。每个 div 元素都设置类 two 和 columns。这样每行会占据 12 列宽。在每个元素中添加一小段文本。

```
<div class="row">
<div class="two columns"><p>Loremipsum dolor sit amet...</p></div>
<div class="two columns"><p>Cum sociisnatoquepenatibus et...</p></div>
<div class="two columns"><p>eufacilisis sem. Phasellus...</p></div>
<div class="two columns"><p>Loremipsum dolor sit amet...</p></div>
<div class="two columns"><p>Cum sociisnatoquepenatibus et...</p></div>
<div class="two columns"><p>eufacilisis sem. Phasellus...</p></div>
</div>
```

在浏览器中你能看到 6 个元素中的内容具有很好的布局。当浏览器窗口缩小时，6 个元素中的内容会自适应为 100% 宽度。

目前为止，如果所有元素在屏幕上都是左对齐的，那么该网格框架的布局非常有序。然而，现实情况不全是这样，总有一些情况需要将内容居中、右对齐或者任意排列。不用担心，Gumby 960 响应式框架已经考虑到了这些情况。接下来再添加一些行来演示如何实现这些效果。

在新一行中我们将创建两个 div 元素，达到一个左对齐、一个右对齐的效果。先创建一个 div 元素，在里面再创建两个 div 元素。两个 div 元素中第一个将会在屏幕的左边，类为 two 和 columns。在这两个类的作用下，元素会浮动到左边并占据两列宽度。我们想让另外一个元素只占据 6 列，设置其类为 six 和 columns。但是并不想让这个元素浮动到左边，而是要和第一个 div 元素间有些空间。那些含有 left-margin 百分比宽度的类能实现这点。在本示例中，需要该元素向右移动 4 个列宽。可以通过添加类 push four 来实现。

```
<div class="row">
<div class="two columns"><p>Loremipsum dolor sit amet...</p></div>
<div class="six columns push_four"><p>Consecteturadipiscingeli...</p>/
div>
</div>
```

下列是类 push four 的 CSS 源码：

```
.row .push_four {
margin-left: 36.170271%;
}
```

有一个特殊的类可以使元素内容实现“居中”效果。这里将居中打上引号是因为这并不是真正的居中，而是伪居中。Gumby 网格框架并没有使用 text-align:center 或 float:center 等属性，而是采用了一个智能的 left-marign 系统。居中且含有 six column 的 div 元素的 CSS 代码如下：

```
.row .six.centered {
margin-left: 25.531956%;
}
```

这遵循了 number 类中使用到的相同模式，一个居中的 five column 行拥有较大的左外边距：margin-left: 29.787282%。

最后在结束本节前，让我们使用这个框架创建一个响应式菜

单。多花点时间来演示如何使用该框架提供的一些响应式 UI 元素是很值得的。

在此只需要使用 HTML 来构建这个菜单，因为框架已经帮我们完成了 CSS 代码。在顶部的 container div 元素里添加一个 row div 元素。在 row div 元素中添加一个 nav 元素，设置 id 值为“prettynav”，同时设置类为 pretty navbarclearfix。接下来在 nav 元素里添加一个 a href 标签，link 值为 #，标签的类值为 toggle，添加一个 data-for 值为 #prettynav>ul 的元素标签。给 a href 中添加一张存放在 img 目录中的图片 img/icon_nav_toggle.gif。

```
<div class="row">
<nav class="pretty navbarclearfix" id="prettynav">
<a href="#" class="toggle" data-for="#prettynav&gt; ul"><img src="img/
icon_nav_toggle.gif"></a>
</nav>
</div>
```

该导航菜单在移动设备上会被隐藏，而 a href 元素则以按钮的方式来显示导航菜单。

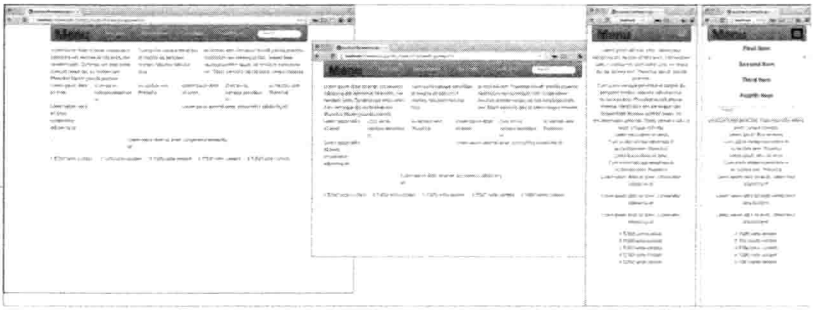
在 a href 元素后面添加一个无序的列表（ul），列表中的条目（li）是导航链接：

```
<ul>
  <li><a href="#">First Item</a></li>
  <li><a href="#">Second Item</a></li>
  <li><a href="#">Third Item</a></li>
  <li><a href="#">Fourth Item</a></li>
</ul>
```

这已经是令人兴奋的响应式菜单系统，但我们仍能做得更好。可以给菜单中的每个条目添加子菜单。还可以使用一个拥有 dropdown 类的 div 元素来实现子菜单。在该 div 元素中，添加带有条目的 ul 元素，就像父菜单中的一样。该子菜单会自动转变为隐藏的子菜单。

```
<li>
<a href="#">Second Item</a>
<div class="dropdown">
<ul>
<li><a href="#">Dropdown item</a></li>
<li><a href="#">Dropdown item</a></li>
</ul>
</div>
</li>
```

下面的截图演示了 Gumby 框架。



4.5.3　工作原理

Gumby 960 网格框架是一个简洁易用的元素布局框架。并不需要对它的工作原理了解太多就可使用它。先学会如何对 div 元素设置类来使其能在该框架下工作，然后就可以自由发挥了。学习并了解如何使用框架中提供的 UI 元素需要花些时间，但是绝对值得。

4.6　易上手的 Bootstrap 框架

Bootstrap 框架（前身为 Twitter Bootstrap 框架）能够从众多框架中脱颖而出，因为它是完全响应式的。你可以将它作为静态框架，也可以使用其提供的附加文件快速部署一个完全响应式的站点。如果需要快速开发网站，它会是你的得力助手，你只需要作最少的设计调整来适应该框架的标准。

使用 Bootstrap Framework 作为关键字搜索，点击链接 http://twitter.github.com/bootstrap/，然后点击页面上的大按钮 Download Bootstrap 就可得到 BootStrap 框架[⊖]。下载的文件包括 CSS 文件、图片及 JavaScript 文件，但没有文档。官网上有很多优秀的文档，并且与文档中的示例源码保持一致。本节将引领你使用 Bootstrap 框架。

4.6.1 准备工作

Bootstrap 框架使用起来非常简单。几分钟之内就可以设置好一个模板。话虽这么说，具体还要动手做。先创建一个新的 HTML 文件，在 header 标签中加入对 Bootstrap 的 CSS 文件的引用，以便我们及时查看进展：

```
<link href="css/bootstrap.css" rel="stylesheet" media="screen">
<link href="css/bootstrap-responsive.css" rel="stylesheet"
media="screen">
```

先从创建一个拥有顶部导航栏和少量内容的简单页面做起。导航栏会根据不同屏幕宽度优化其显示。导航栏 div 元素需要使用几个类来实现需求。它们是 navbarnavbar-inverse navbar-fixed-top。在该 div 元素里，添加一个拥有 container 类的 div 元素。该元素中有一个用于显示在移动版本中的按钮图片。当点击它时，会显示移动版本的菜单。该菜单在移动版和桌面版中都能优化显示。是不是很酷？

以下是菜单的示例代码：

```
<div class="navbarnavbar-inverse navbar-fixed-top">
     <div class="navbar-inner">
       <div class="container">
```

⊖ 2014 年 4 月，网址为 https://github.com/twbs/bootstrap，页面布局也有变动，但方法是相似的。——编辑注

```
        <a class="btnbtn-navbar" data-toggle="collapse" data-
target=".nav-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </a>
        <a class="brand" href="#">Project name</a>
        <div class="nav-collapse collapse">
          <ul class="nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#contact">Contact</a></li>
          </ul>
        </div><!--/.nav-collapse -->
      </div>
    </div>
  </div>
```

然后在 header 标签中添加对 jQuery 库的链接。

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

在 HTML 的底部，body 的结束标签之前，加上对 js/bootstrap.js 文件的链接。

```
<script src="js/bootstrap.js"></script>
```

最后，将 JS 文件复制到网站根目录。

现在可以检查这个响应式导航了。

怎么样？很不错吧。是不是对 Bootstrap 框架激动不已？来继续实现一些响应式内容布局。接下来将会使用 Bootstrap 建立一个营销网站。

首先添加一个有 container 类的 div 元素。如果回头看看菜单元素，就会发现 container 类是一个用于控制内部子元素的响应式宽度的可复用布局元素。在 container 元素中添加一个新的 div 元素，并设置类 hero-unit。同时在此 div 元素中添加一些用于在屏幕中显示的大型广告牌风格的内容。

```
<div class="container">
<div class="hero-unit">
<h1>Hello World</h1>
<p>Loremipsum dolor sit amet...</p>
</div>
</div>
```

刷新浏览器并试着调整浏览器大小。毫不费力就能让一切看起来都不错。下面我们想添加几列摘要文本。这样，页面看起来就像一个不错的登录页了。能够实现这些是不是很高兴?

Bootstrap 框架使用一个拥有 row 类的 div 元素来勾勒出列宽。所以新建一个拥有 row 类的 div 元素来创建一个新行。在这行中你有 12 列可以装载内容。在本节中，为了简单起见，在 row div 元素中插入三个 div 元素，每个都赋予 span4 类。在每个 span4 元素中，插入一个二级标题及一段 Ipsum（http://lipsum.com）填充文本段落。

```
<div class="row">
<div class="span4">
<h2>Header</h2>
<p>Loremipsum dolor sit amet, consecteturadipiscingelit...</p>
</div>
<div class="span4">
<h2>Header</h2>
<p>Loremipsum dolor sit amet, consecteturadipiscingelit...</p>
</div>
<div class="span4">
<h2>Header</h2>
<p>Loremipsum dolor sit amet, consecteturadipiscingelit.....</p>
</div>
```

打开浏览器或刷新窗口来查看实时的布局效果。新行被分为三列，当在移动设备浏览器上或者较小宽度的窗口上显示时会缩为一列。

你可以复制整个 row 类元素及里面的 HTML 代码，然后粘贴成一个新行，这样会看起来更美观。

现在已经有了一个漂亮的页面呈现在我们的面前，这并没有耗费多少精力，让我们给该页面再提升一个层次。接下来的部分是体现 Bootstrap 框架灵活性的一个极好的示例。下面需要给页面添加一个侧边导航栏。

在第二个 container 类元素中，将 hero-unit 和 row 元素使用一个新的 div 元素包裹起来，并赋予该 div 元素类 span9。然后在该 div 元素前新插入一个拥有类 span3 的 div 元素。这将自动适应页面布局的变化。接下来我们将在新插入的 div 元素中快速创建一个菜单。

在 span3 div 元素中添加一个新的 div 元素，并赋予类 well 和 sidebar-nav。这些设置赋予侧边栏导航好看的样式。现在添加一个无序的列表（ul）作为菜单列表，赋予类 nav 及 nav-list。你可以给菜单列表赋予类 nav-header，来使其变为列表标题。在每个列表项中添加一个 href 链接作为导航：

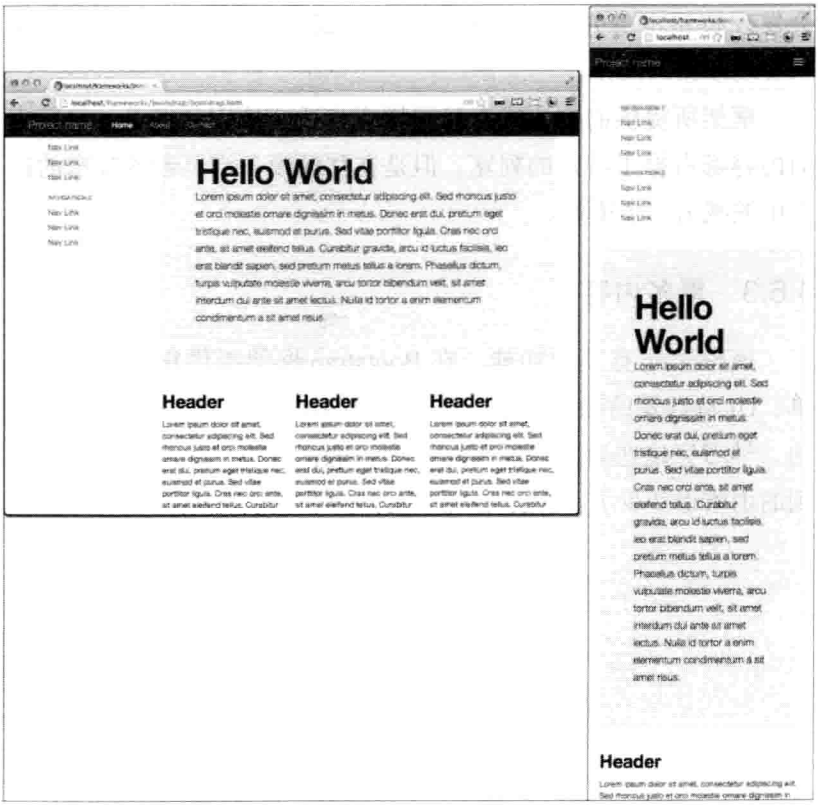
```
<div class="well sidebar-nav">
     <ul class="navnav-list">
          <li class="nav-header">Navigation 1</li>
          <li><a href="#">Nav Link</a></li>
          <li><a href="#">Nav Link</a></li>
          <li><a href="#">Nav Link</a></li>
          <li class="nav-header">Navigation 2</li>
          <li><a href="#">Nav Link</a></li>
          <li><a href="#">Nav Link</a></li>
          <li><a href="#">Nav Link</a></li>
     </ul>
</div>
```

就快完工了。将这两个新的 span* 元素包裹在另一个 div 元素中，并赋予该 div 元素类 row 或 row-fluid。最后，将包含摘要内容的 div 元素的类 row 修改为 row-fluid。

```
<div class="container">
          <div class="row">
            <div class="span3">
```

```
            <div class="well sidebar-nav">
               <ul class="navnav-list">
                 <li class="nav-header">Navigation 1</li>
                 <li><a href="#">Nav Link</a></li>
                 <li><a href="#">Nav Link</a></li>
                 <li><a href="#">Nav Link</a></li>
                 <li class="nav-header">Navigation 2</li>
                 <li><a href="#">Nav Link</a></li>
                      <li><a href="#">Nav Link</a></li>
                      <li><a href="#">Nav Link</a></li>
                  </ul>
              </div>

          </div>
          <div class="span9">
          <div class="hero-unit">
<h1>Hello World</h1>
     <p>Loremipsum dolor sit amet, consecteturadipiscingelit...</p>
       </div>
         <div class="row-fluid">
           <div class="span4">
             <h2>Header</h2>
             <p>Loremipsum dolor sit amet, consectetur adipiscing
elit...</p>
           </div>
           <div class="span4">
             <h2>Header</h2>
             <p>Loremipsum dolor sit amet,
consecteturadipiscingelit...</p>
           </div>
         </div>
       </div>
    </div>

</div>
```

恭喜你，全部搞定！现在你已经拥有了坚实的基础来实现专业级的响应式布局与设计。你只需对该示例作少量的修改便可得到一个成品。下列截图显示了通过 Bootstrap 框架实现的页面效果。



4.6.2　工作原理

是不是看起来很神奇？ Bootstrap 框架是我使用过的所有框架中最简单最好的一个。一旦你通过本节及 Bootstrap 框架的文档熟悉了这些类及布局，那么就能非常容易地快速开发项目。

接下来讨论一些具体内容。首先是响应式菜单，在 container 元素中顶部 div 元素的类是 button，其只会显示在移动设备版本中，该元素的目的是当被单击时，显露出移动样式的 div 菜单元素 nav-collapse。

这为创建可用的、优雅的响应式菜单开了个好头。然而，你

会发现按钮本身并不起作用，因为我们需要添加一些 JavaScript 来实现按钮的操作。

框架所提供的响应式布局同样会自适应使用的场景。每行中的内容都占据了特定的列宽，但是在移动设备浏览器或较窄的窗口中缩减为一个单列。

4.6.3 更多内容

该框架还有其他功能。在 Bootstrap 框架提供很多元素、菜单、UI 特性及的动画特效。花一些时间进行深入学习是非常值得的。学习完这些后，我发现可以更快地完成工作，与此同时所碰到的困难也变少了。

第 5 章
设计移动设备优先的 Web 应用

5.1 简介

本章将会关注移动设备优先的响应式设计。注意，这里的意思是先为移动设备设计站点，然后再去适配那些有差异甚至截然不同的桌面设备。本章的一些方法可能会涉及 jQuery Mobile，jQuery Mobile 是一个用于开发移动 UI 元素和微件（widget）的免费开源库。除此之外，还会编写一些客户端脚本，来处理站点在移动设备上的独特显示效果。

5.2 使用 Safari 开发者工具的用户代理设置

为了开发移动设备优先的应用，一般来说需要在本地部署并测试相关的特性。到目前为止，那些涉及响应式页面设计的方法大多依赖于媒介查询，并根据客户端浏览器窗口大小进行页面优化及决定页面的布局方式。但这并不是开发能够工作于移动设备上的站点的唯一方法，嗅探用户代理就是另一个方法。

或许你已经知道什么是用户代理，但是在此假设你并不知道。不然的话，如果什么都知道的话那就没有必要购买本书了，难道不是吗？用户代理信息位于 HTTP 请求的头信息中，用户代理可以标识发起请求的客户端的具体信息。其中包括处理器、操作系统版本、浏览器、渲染引擎、IP 地址及其他标识信息。

依据产品的需要或是开发者的喜好，一些站点为了响应式的设计和用户体验，根据不同的移动设备类型或用户代理数据设计了不同的显示样式。本方法需要服务器或客户端具有一定的智能，也就是说能够读取用户代理信息并能解析出其中的内容，然后再根据不同的场景渲染对应的样式。

因此假设现在已经开发了一个新的 Web 应用，当读取到客户

端请求中的用户代理是移动设备时，需要加载基于移动设备的模板。但是现在想要在不启动服务器的情况下进行测试，最好的方式就是使用 Safari 中的用户代理配置特性。

使用 Safari 浏览器的用户代理设置特性是双赢的，因为其不仅能模拟 iOS 设备上的 Safari 浏览器用户代理信息，也能模拟 Android 平台上的浏览器用户代理信息。因此你需要平静地面对，因为 Android 类型的用户代理信息实际上还是通过设置 Mobile Safari 得到的，而这些都是为了让生活变得更简单。这些特性是不是很棒呢？

清楚地列出需要为哪些浏览器和用户代理进行测试是一项好实践。

5.2.1　准备工作

在苹果电脑上 Safari 是内置的，因此苹果电脑的用户现在已经在准备过程中领先了。但是还是得等等，以便 Windows 用户赶上进度。

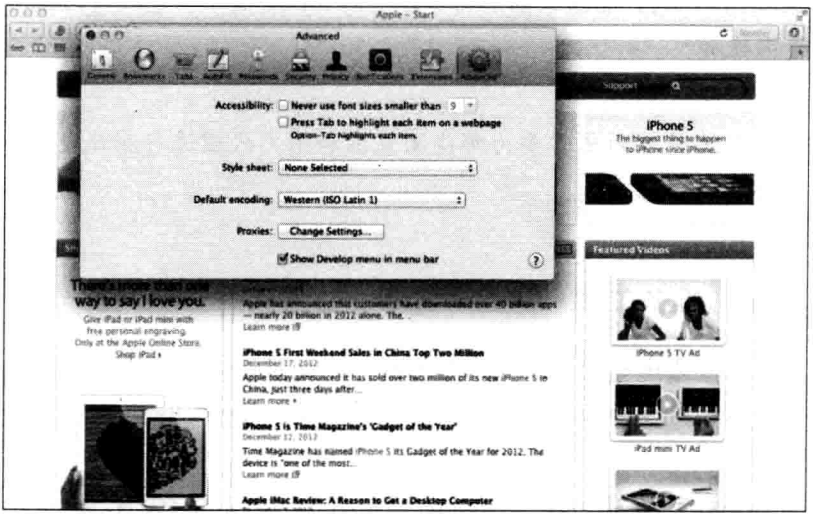
看起来苹果公司不太可能继续开发 Windows 版本的 Safari 浏览器。事实上，搜索关键词 Safari Windows，第一条搜索结果的链接并不是 Safari 主页，而是指向一个苹果产品的支持页面，其中含有最新的 Safari Windows 版本 5.1.7，而不是 Safari 自身的最新版本（版本 6[⊖]）。但这并不影响本节的目的，因此我们继续。

⊖　在作者撰写本书时，Safari 的最新版本为 Safari 6。——译者注

5.2.2 实现方式

首先，打开 Safari 浏览器；现在需要访问展示用户代理的作用和示例的站点来了解相关信息。输入网址 http://whatsmyuseragent.com，在该页面你将会了解用户代理的相关细节信息。

在 Safari 界面中，选择 Safari | Preferences，或是按下组合键 Command+ ,。在 Advanced 标签下面，勾选 Show Develop menu in menu bar 复选框，如下图所示。



现在菜单栏中多出了 Develop 菜单选项。单击菜单并选择 user agent，就能看到含有不同用户代理选项的子菜单。菜单中很多选项都是很有用的，但对于本节最相关的是 Safari iOS 5.1 - iPhone 和 Safari iOS 5.1 – iPad（基于不同版本的 Safari，菜单中的选项极有可能不是 5.1 版本），具体截图如下。

选择其中一个 iOS 版本后，页面会自动刷新。现在就能在页面上看到最新的用户代理信息，结果如下图所示。

5.2.3　工作原理

这表面上看起来很简单，实际上背后发生了一系列的操作。浏览器在与服务器进行交互时，附带了客户端电脑和浏览器的相

关信息，使得服务器根据客户端的差异返回不同的页面。在服务端，可以构建相应的逻辑来为移动设备终端生成特殊的样式、模板、脚本，甚至完全不同的页面内容。

5.3 通过 Chrome 插件设置用户代理

Chrome 浏览器具有种类繁多的插件，几乎可以实现任何目的。下面，来看看一个用于设置用户代理信息的插件。

为什么需要“篡改”用户代理信息？难道那不是欺诈吗？好吧，我承认是欺诈，但这只是为了结束这场辩解。除此之外，该行为不会造成任何损害，服务器即使发现客户端的浏览器发送了假的用户代理信息，也不会感到被欺骗和伤害。修改客户端用户代理信息只是为了让服务器坚信桌面浏览器确实是移动设备的浏览器。如果服务器端认为你在使用移动设备的浏览器，就会构建移动版本的页面，这就是我们想要的结果。

5.3.1 准备工作

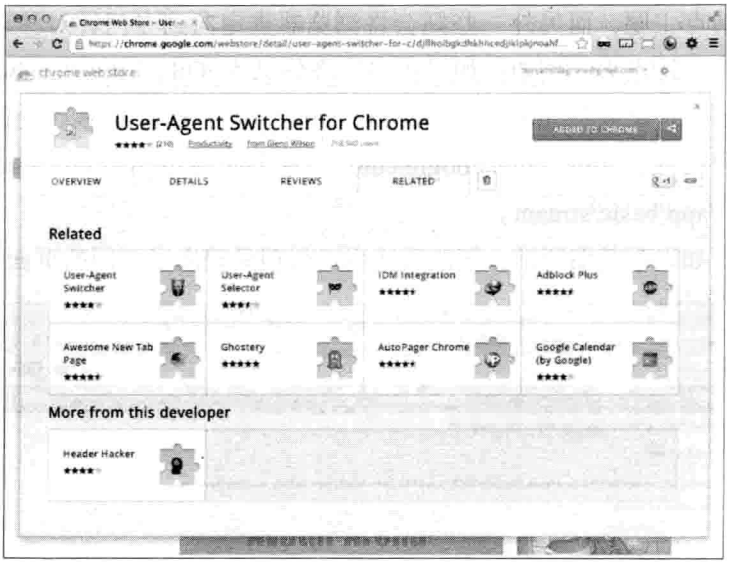
在此需要找到一种能够在不同的用户代理信息之间切换的方法，同时方法本身也得非常简单。事实上，最理想的莫过于浏览器上面拥有一个按钮，点击以后就能自动切换。那么在哪里能够找到这个功能强大的新奇玩意呢？答案是 Chrome 网络商店（Chrome Web Store）。

我试过几个能够切换用户代理的 Chrome 浏览器插件，其中有一个已经成为我的响应式工具集中的一员。Chrome 的用户代理转换器 (User-Agent Switcher) 使得你可以在一个非常全面的用户代理信息列表中切换所需的代理配置。获取它的一个较为简便的方式为搜索关键词 Google UA Spoofer。

5.3.2　实现方式

搜索结果的第一条应该就是指向 Chrome 网络商店中用户代理转换器的链接。如果确实如此，进入相应页面然后单击 ADD TO CHROME 按钮。安装该插件仅此一步。同时它的使用也非常简单。

现在看看浏览器顶端区域，在地址栏的右侧，应该能找到一个形如小面具的新图标。当单击这个图标时，就会出现不同浏览器的菜单，然后每一个子菜单中都包含相应的可用版本。我已经做过测试，非常简单。下面的截图可以证明。



5.3.3　工作原理

Chrome 浏览器用户代理信息篡改插件首先拦截浏览器正常请求头中携带的用户代理信息，然后用篡改过的用户代理信息进行替换。我们到目前为止，只是探讨了如何使用用户代理切换插件。至于如何设计站点，并能够处理请求中的不同代理信息就是完完

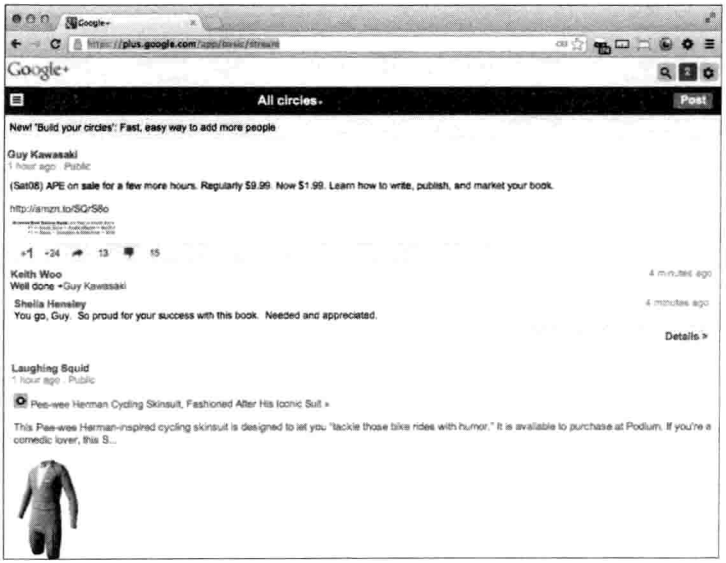
全全的另一个主题了。

为了能够立即看到效果，访问页面 http://whatsmyuseragent.com/，然后将代理从 iOS 切换到 iPhone。你将会看到用户代理信息变为了 iPhone。多试些不同的选项，看看那些你最喜欢的网站是如何应对这些不同的用户代理信息的。

5.3.4 更多内容

访问那些 Web 中最受欢迎的站点，就能观察到它们是如何处理不同的用户代理信息的。这些站点中，其中一些会采用不同的站点主题，而另外一些则会将浏览器重定向到专门针对移动设备的子站点去。例如：如果用户代理信息为 iOS，http://facebook.com 会重定向到 http://m.facebook.com/?_rdr；而针对移动设备的用户代理，https://plus.google.com/ 会重定向到 https://plus.google.com/app/basic/stream。

如以下截图所示，篡改后的用户代理呈现出了不同的页面。



5.4　使用插件调整浏览器窗口大小

我得坦率地告诉你，本节是关于如何安装和使用我正在使用的调整窗口大小的插件。如果你有更好的选择，请告诉我。在经过搜索和尝试后，我的选择是名为“Window Resizer”的插件。

除了在目标设备上测试媒介查询之外，通过插件调整窗口大小也不失为一个精确的方法。然而对于你的响应式站点而言，这也只是部分测试。在站点上线之前，确保进行了模拟器测试和真实设备测试是相当重要的。没有什么事情比眼看着刚上线的网站宕掉或崩溃更糟糕的了。

5.4.1　准备工作

使用 Google 这个好伙伴，搜索关键词 Window Resizer。搜索结果的第一条应该就是 Chrome 网络商店中的 Window Resizer 插件。那就像是黑夜中的灯塔！插件的评价是五星，并且还是免费的，那何不赶快单击安装的链接呢？

5.4.2　实现方式

如果你完成上述步骤的话，现在应该位于 Chrome 网络商店的安装页面。你将会看到一个醒目、大气且内敛、上面写着 + ADD TO CHROME 的深蓝色按钮。你是如此迫切地想要点击它。想象着浏览器窗口大小有多少种可能性；想象着当你拖拽浏览器的边角，并尝试猜测当前窗口尺寸大小的痛苦。忘记那一切吧，现在就单击按钮！

浏览器中出现进度条表明安装正在进行。最后，蓝色按钮变为绿色，安装完成。

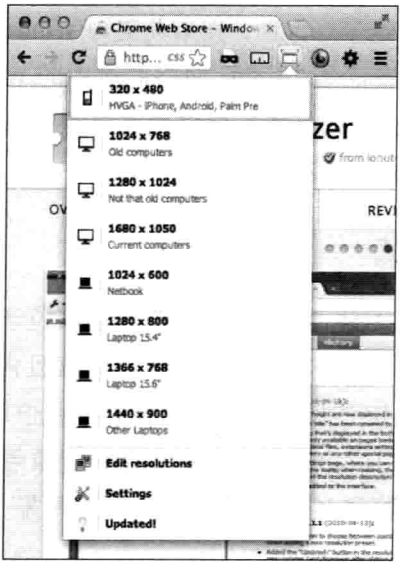
在浏览器中，可以发现地址栏右侧区域多出了一个看起来像

小型浏览器的新图标。好奇心驱使你想知道这个新插件到底能干些什么。

实际上这是测试站点在不同条件下的媒介查询及响应式页面的完美方式，效果仅次于直接使用目标设备进行测试。

5.4.3 工作原理

该按钮用于测试基于像素级别的响应式设计。当单击按钮后，会看到不同浏览器窗口尺寸的列表。每一个选项都是像素级匹配并保证浏览器的窗口尺寸大小与你想要的完全一致。该插件替你完成了之前需要猜测窗口大小并度量尺寸的工作，你所要做的仅仅只是点击一下按钮使之生效，如下图所示。



5.5 学习视窗及其相关选项

如果视窗只有一个目的，那就是操控移动浏览器窗口。视窗

对于如何在移动设备上的浏览器中渲染页面起绝对性作用。

5.5.1　准备工作

如果使用的是苹果电脑，通过下载 Xcode 就能获取 iOS 模拟器。它是 Xcode 安装包的一部分。我经常通过 Spotlight 来打开 iOS 模拟器。按下 Command 和空格键，Spotlight 的搜索框就会出现在屏幕的右上角位置。输入关键词 iOS Simulator，iOS 模拟器就会出现在结果列表中。点击即可启动 iOS 模拟器。

5.5.2　实现方式

打开一个之前章节中所做过的项目页面。我倾向于打开 3.5 节中的 resp-width-layout-media-query.html 页面。

要在 Windows 系统中使用 iOS 模拟器，就得去网上查找了。在搜索后，我发现 http://iphone4simulator.com/ 提供的模拟器比较不错，另一个不错的模拟器是 http://iphonetester.com/ 提供的。为了使用这些模拟器，需要将对应的页面上传到指定的服务器，然后才能在模拟器中看到效果。模拟器并不能读取本地硬盘上的文件，除非你在本地运行一个服务器。

为了便于对比，先在桌面浏览器中打开页面。然后在 iPhone 模拟器中输入页面 URL，当发现页面看起来和在桌面浏览器中一模一样的时候，你会感到震惊。在早期开发响应式站点时，我也遇到过同样的困难，页面并不总是按照我所期望的方式渲染。问题就在于移动浏览器并不知道你想要多大尺寸的窗口。它足够智能，但绝非聪明。和所有的软件一样，你得告诉它如何去做。现在深吸一口气，让我们一起来修复这个问题。问题页面呈现在下面的截图中。

通过配置视窗，使得移动设备上的浏览器知道它该做什么。首先添加一个视窗 <meta> 标签：

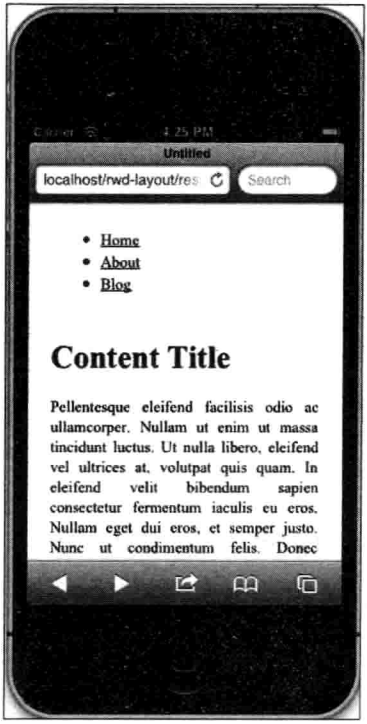
```
<meta name="viewport">
```

在继续进行下一步之前，有一个注意事项需要告知。如果并不打算为移动设备设计页面，请不要添加视窗 <meta> 标签。因为它可能会导致在页面渲染时出现意料之外的结果。例如呈现在你面前的可能只是页面的一小部分，但你又无法通过滚动条来浏览整个页面。

接下来看看视窗属性的相关选项。首先是宽度，我是 K.I.S.S.(keep it short and simple) 原则的忠实拥趸。除非有特别的需求来指定特定的视窗宽度，否则一律使用设备自身的屏幕宽度作为视窗的宽度值。通过这样的设定，浏览器会尝试获取设备的屏幕宽度，并将页面宽度设置为该值。而设定一个具体的宽度值，如 1000px，或许在 iPad 上面看起来很不错，但是对于手机来说就显得太大了，同时所有宽度小于 1000px 的媒介查询都会失效。

```
<meta name="viewport" content="width=device-width">
```

完成这些修改，再次在 iOS 模拟器的浏览器中打开这个页面后，就会发现问题已经得到修复，效果如下图所示。



下面来看看页面缩放。这里假设你并没有任何古怪的需求，例如想要初始页面比例不为 1。添加视窗 <meta> 标签，初始比例设定为 1。

之前提到过不做任何奇怪的需求和假设，但这里为了演示缩放效果，把初始值改为 2，然后刷新页面。

接下来再改为 0.4。记住这里只是为了展示不同的缩放效果。再次刷新页面，在竖屏情况下，会发现页面使用的是适用于较小页面宽度的媒介查询。现在把模拟器换回横屏模式。这时更大宽度值的媒介查询设定被触发。这是一个非常有意思的尝试，现在把页面比例初始值改回 1。

最后，如果想要用户通过多点触控实现页面的放大和缩小功

能，通过设置 meta 属性 maximum-scale 来限定所允许页面缩放的阈值。设置最大比例为 1 就能够禁止页面的缩放。

```
maximum-scale=1
```

5.5.3 工作原理

视窗 <meta> 标签最开始是苹果的 Safari 浏览器所引入的，随后其他的浏览器也都开始支持该特性。该标签用于定义渲染页面的宽度。当浏览器发现视窗 <meta> 标签设置了宽度属性，就会按照所设定的宽度加载页面，并结合初始的页面比例属性进行页面的渲染。

5.6 为 jQuery Mobile 添加标签

本节会涉及响应式设计中的一个新领域，那就是移动设备优先 (mobile-first)。移动设备优先，简而言之就是先为站点设计移动版本，然后才去考虑适配桌面浏览器。但是，这并不是说设计的站点只有移动版本，仅仅只是优先考虑移动设备上面的布局和样式。

移动设备优先或许需要重新审视你的设计，至少也得换个角度进行设计。改变难道不好吗？如果不通过尝试新方法，我们如何提高设计的技能？进化论不是说只有适应环境改变的个体才能生存吗？

让我们来开放思维，尝试一些移动设备优先的设计。

5.6.1 准备工作

首先，访问 jQuery Mobile 的网站，网址为 http://jquerymobile.com。如果你像我一样懒惰，也可以搜索关键词 jQuery Mobile。

假如你不想手动搜索，在此我也会提供给你直接的搜索链接。直接搜索的链接地址为 http://lmgtfy.com/?q=jquery+mobile&l=1。这里还有更加简洁的地址 http://bit.ly/TMpuB8。

现在可以下载库文件，如果想本地托管相关资源的话（当然本地托管的方式好处很多）；但在本节中，为了更快捷地进行演示，采用的是非本地托管的形式。

jQuery Mobile 的网站上有非常丰富的文档和示例。网站甚至提供了一个下载器，可以只下载对你的移动应用来说必需的库，从而减少库文件大小。

5.6.2　实现方式

首先在 IDE 中创建一个新的 HTML 页面。在头元素中添加视窗 <meta> 标签：

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

接下来，引入指向 jQuery Mobile CSS 和 JavaScript 文件的链接。

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/
jquery.mobile-1.2.0.min.css" />
<script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
<script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
```

现在是教学时间，花几分钟来说说样式表是值得的。在之前的代码中，曾经引入过远端托管的 jQuery CSS。我建议你尽量不要用它（如果你打算在本地托管该文件的话），同时将新添加的自定义 CSS 配置都存放到同一个样式表中。除此之外，如果想要对 jQuery 的 CSS 作修改，添加另外一个 CSS 文件并显式地重载命名空间，或使用 !important 重载。文件名可以类似于 jQuery-mobile-changes.css。虽然我并不认为你需要修改 jQuery CSS 的文件名，但万一要修改，上面的方式不失为一个好的处理方式。这样做的

原因在于当有新版本的 jQuery 发布时，无需中断网站服务即可完成升级。

头元素已经基本完成。下面再给页面添加一些基本内容。首先，在页面中放置一个 元素：

```
<body>
    <div>

    </div>
</body>
```

jQuery Mobile 有一个很棒的特性，那就是使用标签。标签可以添加到 HTML 元素中，但并不是用来渲染页面。这样做的好处是能够复用为桌面浏览器所设置的页面模板，而要做的只是在页面中添加 jQuery Mobile 的脚本和样式。接下来，为 元素添加一些标签，使得 jQuery Mobile 能够作用于该页面。给该元素添加 data-role="page" 标签。

```
<div data-role="page">
```

现在构建一个文本示例页面用来展示效果。

在 元素中添加一个新的 h1 标题。同时在 元素上添加 data-role="header" 属性。然后，在浏览器中打开页面，来看看 jQuery Mobile 的样式效果。

```
<div data-role="header">
   <h1>Adding tags for jQuery Mobile</h1>
</div>
```

这是一个好的开始，下面继续看看更多利用 jQuery Mobile 进行页面结构设计的例子。

对于桌面版本的页面，同样可以给元素赋予 ID 和 class 属性。

接下来，添加内容。在 元素中添加一个文本段落。并给 元素赋予 HTML5 的数据属性 data-role:"content"。

```
<div data-role="content">
  <p>
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean
commodo ligula eget dolor. Aenean massa....
  </p>
</div>
```

使用同样的方式添加一个页脚。在 元素中放置一个 h4 标签，内容同样为一些简单的文本。然后给 元素设置 data-role="footer" 属性：

```
<div data-role="footer">
  <h4>The Footer</h4>
</div>
```

以上就是本节的全部示例。jQuery Mobile 官网上面有非常好的文档和示例，学习它们来更好地利用 jQuery Mobile 框架搭建网站。在本章中，还会学习更多 jQuery Mobile 的技巧。现在就去找找看吧。下图就是通过 jQuery Mobile 渲染的页面效果。

5.6.3 工作原理

jQuery Mobile 利用 HTML5 的数据属性来触发脚本，从而作用于所标记的元素或者组件。如果元素含有特定数据属性，脚本就会自动触发，从而渲染出特定的效果。

5.7 基于 jQuery Mobile 添加子页面

jQuery Mobile 有一个非常棒的特性，通过它可以将较大的 HTML 页面拆分为更小、更易阅读的几个部分。试想页面中有很多的内容，但是又不太希望用户通过不断地滚动鼠标进行浏览。可以考虑使用 jQuery Mobile 的多模板结构。网页在移动设备上的用户体验和桌面电脑是截然不同的。在早期的网站里面，经常听到说“内容为王”；现在的移动网页只有有限的页面空间，部分正常的网页内容也可能显得冗余了。因此在每一个页面中需要显示的内容也是有必要进行一些限制的。在本节中，我们将会使用 jQuery Mobile 来拆分一个大型网页，把那些内容都转换成小的易于阅读的部分。

5.7.1 准备工作

在前一节中，通过 jQuery Mobile 的标签创建了一个简单的页面。现在继续复用之前的页面文件，将其另存为一个新的页面。到此本方法所需要的准备都完成了。

5.7.2 实现方式

给 元素 (含有属性 data-role 的页面元素) 添加值为 p1 的 ID。这将帮助 jQuery 在多个元素间进行识别和转换。

```
<div data-role="page" id="p1">
```

在多页面切换的过程中，刚才创建的元素会被 jQuery Mobile 当做第一个页面。下面创建其他的页面。在 <body> 结束标签之前创建新的 标签对。像之前的 一样，赋予属性值 data-role="page"，并同时设置其 ID 为 p2。

```
<div data-role="page" id="p2">
```

和之前含有属性 data-role="page" 的 元素一样，新加的页面也需要相应的 data-role="header"、data-role="content" 及 data-role="footer"。为了简便起见，你也可以直接将前面的元素内容复制到新创建的 "p2" 元素中。

```
<div data-role="page" id="p2">
  <div data-role="header">
    <h1>The second page</h1>
  </div>
  <div data-role="content">
    <p> Lorem ipsum dolor sit amet...</p>
  </div>
  <div data-role="footer">
    <h4>The Footer</h4>
  </div>
</div>
```

本示例基本快完成了，现在只需要将这些页面组合在一起即可。在 p1 页面内 元素最后，添加 href 标签使其链接到 p2 页面：

```
<a href="#p2">Page 2</a>
```

在 p2 页面含有 data-role="content" 属性的 元素中，添加另外一个链接，链接所对应的是第一个页面的 ID：

```
<a href="#p1">Back to Page 1</a>
```

保存好页面并打开，呈现在你面前的是非常漂亮并类似于内置控件样式的移动站点。点击页面上的链接，会在不同的页面间流畅地切换。当然，移动设备上的返回键在该场景下同样适用。

这样的设计是非常有意义的，因为它使得站点与移动应用保持一致的用户体验。第一个页面呈现在下面的截图中。

下面的截图是第二个页面的效果图。

5.7.3　工作原理

jQuery Mobile 拥有将单 HTML 页面内容加载为多页的特性，使得看起来就像是多个页面或是子页面一样。只需是在页面间添加 HREF="#page" 的链接就可以链接多个页面。点击这些链接以后，jQuery Mobile 会根据 ID 属性去定位相应的内部页面，然后将页面渲染到视窗中。

5.8　基于 jQuery Mobile 制作列表元素

在开始学习本节之前，有一点要声明：我喜欢那些无序的列表，对于那些为了测试功能而随意填充的表格感到深深地厌恶。实际上，在我的同事那里，我得到了“表格破坏者”的名号。在 HTML 中有一些事情是无法通过设计良好的原生列表来完成的，这也正是为什么我倾向于使用 jQuery Mobile 来处理列表。jQuery Mobile 列表，在我看来，证明了为何列表可以作为用以展示数据、菜单、导航栏等页面内容的更好选择。我对无序列表的异常痴迷就说到这里，现在该看看 jQuery Mobile 列表的实现方式了。

5.8.1　准备工作

想想过去你已经把多少非常糟糕的表格和用糟糕代码编写的东西发布在互联网上了。这些都是对你过去的“罪行”的训诫。现在该结束这些了，开始使用 jQuery Mobile 列表吧！

5.8.2　实现方式

创建一个新的页面，再添加一些 jQuery Mobile 所需要的头信息。包括视窗 <meta> 标签，以及指向 jQuery Mobile 样式表、jQuery JavaScript 文件和 jQuery Mobile 的 JavaScript 链接。可以

在本地托管这些文件，也能通过 http://code.jquery.com 来链接这些资源。

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.0-
beta.1/jquery.mobile-1.3.0-beta.1.min.css" />
<script src="http://code.jquery.com/jquery-1.9.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.0-beta.1/jquery.mobile-
1.3.0-beta.1.min.js"></script>
```

接着创建一个带有 data-role="page" 属性的 元素。data-role="page" 是一个 HTML5 属性，jQuery Mobile 通过它来生成样式、元素和组件。

```
<div data-role="page"></div>
```

在 元素里面，创建一个你所喜欢的机器人无序列表。

```
<ul>
  <li>Hero 1</li>
  <li>Bender</li>
  <li>Optimus Prime</li>
  <li>Soundwave</li>
  <li>Wall-E</li>
  <li>Maximillian</li>
  <li>R2-D2</li>
  <li>GORT</li>
  <li>Cat Quadcopter</li>
  <li>Robocop</li>
  <li>The Maschinenmensch</li>
</ul>
```

现在还不是打开这个页面的时候。因为现在打开所看到的样式和那些现有列表是一致的。如果桌面浏览器版本的 CSS 是设置在单独的配置文件中的话，就可以在那里设置列表的样式。

给无序列表添加 data-role="listview" 属性。现在打开页面就会发现一个含有特定样式的机器人列表。

下面继续我们的探索。因为需要处理列表，并且我们又喜欢使用列表，所以就看看 jQuery Mobile 究竟能使用列表做些什么。

添加另外一个属性 data-inset="true"。可以观察到列表有一个非常炫的外边框，因此列表中每一个元素都不会跑到屏幕之外了。

在有些情况下，列表很长且包含很多元素，例如一个很酷的机器人列表，你可不想以滚动屏幕这样非常不酷的方式来查找并选择一个很酷的机器人。对此场景，jQuery Mobile 有一个内建的解决方案——元素过滤。可以通过添加 data-filter="true" 属性来使其生效。现在刷新移动版的浏览器，会发现在列表元素的顶部出现了一个输入框，用以输入查询关键词来过滤元素。该搜索组件采用客户端逻辑实现搜索和筛选列表元素。再也不用担心滚动列表到最底部去查找那些很酷的机器人了。

接下来看看更进一步的应用。如果需要筛选的数据并没有显示在页面上怎么办？比如机器人的生产厂家。这时需要添加 data-filtertext="" 属性到每一个列表中的元素上。看起来像下面这样：

```
<li data-filtertext="Mom's Robots"><a href="#">Bender</a></li>
<li data-filtertext="Hasbro"><a href="#">Optimus Prime</a></li>
```

下图展示了相关效果。

通过设置不同的主题属性值，列表可以呈现出不同的样式风格。试着添加 data-theme="a" 到该无序列表上。再试试从字母 b 到 f，可以看到每一个不同的属性值都会给列表赋予不同的样式。

下面的代码展示的是一个无序列表，设置了到目前为止所讨论的几种不同的属性值。代码之后的截图展示的是相同列表不同的主题风格。

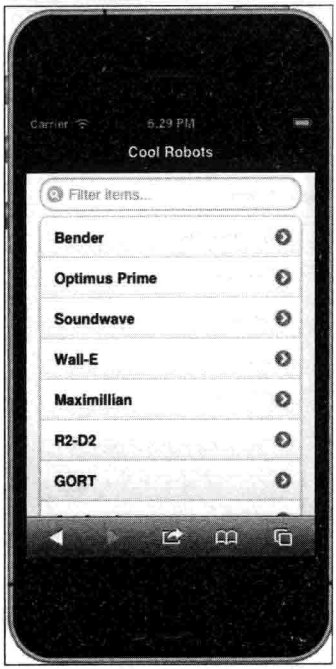
```
<ul data-role="listview" data-inset="true" data-filter="true" data-
theme="g">
```

接着再来看看将这些列表元素变成链接会发生什么。给每一个元素都添加 href 标签。

```
<li><a href="#">Bender</a></li>
```

当页面刷新后，jQuery Mobile 会给所有的列表元素都添加一个图标，来表明每一个元素都是可以点击的链接。但是因为链接的 href 属性值为 #，所以并不会真正地加载新页面。本例的效果

如下面的截图所示。



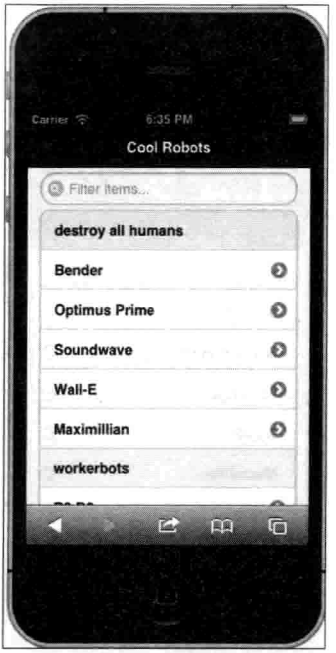
现在将列表分成两个不同的组，“终结者机器人”组和“工作机器人”组。在第一个组的最上面额外添加一个含有属性 data-role="list-divider" 的列表项。

```
<li data-role="list-divider">destroy all humans</li>
```

给另外一个组也添加同样属性的列表项。

```
<li data-role="list-divider">workerbot</li>
```

效果呈现在下面的截图中。



有时候将机器人按照不同的组进行区分是有必要的。不过在这之前，首先需要将列表元素变成嵌套类型。为刚刚创建的 list-divider 元素添加一个 ul 元素，然后剪切机器人列表的前半部分 li 数据并粘贴到 ul 元素中去。

```
<li data-role="list-divider">destroy all humans
  <ul>
    <li><a href="#">Bender</a></li>
    <li><a href="#">Optimus Prime</a></li>
    <li><a href="#">Soundwave</a></li>
    <li><a href="#">Wall-E</a></li>
    <li><a href="#">Maximillian</a></li>
  </ul>
</li>
```

对列表的第二部分也进行同样的操作。接着刷新页面来看看最新的效果。看看是否和下面的截图一样。

在父列表元素中添加一个 h3 头元素标题，甚至可以在其中添加一些描述信息。这使得列表变得越来越酷了，看看下面的截图。

最后再给列表添加一个终极特性。在 jQuery Mobile 中有一个很棒的部件用以处理可折叠的列表元素。接下来会改变 ul 和 li 列表元素的属性值。首先，外层的 ul 元素包含属性 data-role="collapsible-set"、data-theme="b" 及 data-content-theme="d"。

```
<ul data-role="collapsible-set" data-theme="b" data-content-theme="d">
```

给这个 ul 元素的两个直接 li 子元素设置 data-role="collapsible" 属性。

```
<li data-role="collapsible"><h2>workerbots</h2><p>...<p>
```

给可折叠 li 元素的子元素 ul 设置属性 data-role="listview" 和 data-filter="true"。

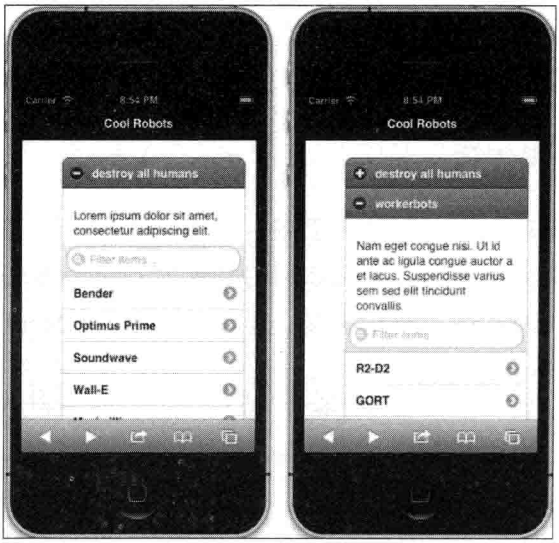
```
<ul data-role="listview" data-filter="true">
```

得到无序列表的完整版本如下所示：

```
<ul data-role="collapsible-set" data-theme="b" data-content-theme="d">
  <li data-role="collapsible">
    <h2>destroy all humans</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer consectetur quam in nulla malesuada congue volutpat mi
molestie. Quisque faucibus, nisi ut malesuada volutpat</p>
    <ul data-role="listview" data-filter="true">
      <li><a href="#">Bender</a></li>
      <li><a href="#">Optimus Prime</a></li>
      <li><a href="#">Soundwave</a></li>
      <li><a href="#">Wall-E</a></li>
      <li><a href="#">Maximillian</a></li>
    </ul>
  </li>
  <li data-role="collapsible" >
    <h3>workerbots</h3>
    <p>Nam eget congue nisi. Ut id ante ac ligula congue auctor a et
lacus. Suspendisse varius sem sed elit tincidunt convallis.</p>
    <ul data-role="listview" data-filter="true">
      <li><a href="#">R2-D2</a></li>
      <li><a href="#">GORT</a></li>
      <li><a href="#">Cat Quadcopter</a></li>
      <li><a href="#">Robocop</a></li>
      <li><a href="#">The Maschinenmensch</a></li>
```

```
    </ul>
  </li>
</ul>
```

最终的列表效果如下图所示。



5.8.3　工作原理

这真是太棒了。除了需要思考列表内容之外，并不需要做太多其他的事情。纯表格绝不可能完成这些特性。只要在元素中添加 HTML5 的数据属性，jQuery Mobile 就会帮你完成那些繁杂的处理逻辑，使得列表看起来是一个井然有序并有着移动设备样式的应用。jQuery Mobile 利用这些数据属性 (不会对页面的布局和样式产生任何影响) 为移动版本的页面重写 HTML 和 CSS。

5.9　基于 jQuery Mobile 开发具有移动设备外观的按钮

本节将会关注于创建按钮元素。在站点的设计中，创建一个

按钮看起来毫不起眼，但是对于一个 Web 应用来说，按钮对于站点的可用性来说却非常重要。

jQuery Mobile 提供了一系列非常棒的按钮可供选择，同时这些按钮都非常易于使用。这些按钮元素和 jQuery Mobile 中其他的组件一样好用。另外，将一个链接变成按钮和将一个 form input 元素变成按钮都非常容易。

5.9.1 准备工作

在 IDE 或文本编辑器中新建一个 HTML 页面，然后添加一些必需的头元素标签。首先添加的是视窗 <meta> 标签，接着是 jQuery Mobile CSS 文件的链接，还有 jQuery 及 jQuery Mobile 的 JavaScript 库文件链接。

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/
jquery.mobile-1.2.0.min.css" />
<script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
<script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
```

在 HTML 的 <body> 元素中，新加一个 元素并设置 HTML5 的 data-role="page" 属性。在这个 元素中再添加一个含有 data-role="header" 属性的 元素，在这个内部的 元素中新加一个 h1 头元素。在这个头元素最后，添加一个含有 data-role="content" 属性的 元素。下面就是代码片段：

```
<div data-role="page">
  <div data-role="header"><h1>There be buttons</h1></div>
  <div data-role="content">...</div>
</div>
```

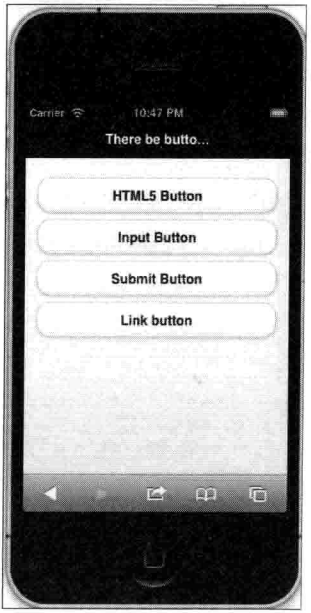
5.9.2 实现方式

先对比一下创建一个基本按钮的几种不同方式。首先就是

HTML5 的 <button> 元素，然后是 <input> 表单元素的 button 和 submit 类型，最后是一个含有 href 属性的链接按钮。在 元素中依次添加这些元素。

```
<button>HTML5 Button</button>

<input type="button" value="Input Button" />

<input type="submit" value="Submit Button" />

<a href="#" data-role="button">Link button</a>
```

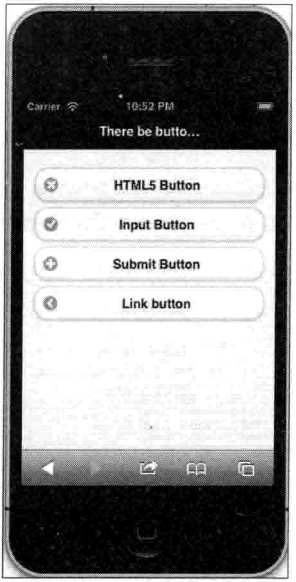
打开刚创建的新页面。会发现这四个按钮看起来都差不多（按钮上的文本除外）。也就是说这些方法生成按钮的方式都是一样的。这给人留下深刻印象，非移动版本的模板文件一般需要特定类型的 submit 元素（虽然并不是真正的移动优先，但不存在完美的方案）。看看下面的截图。



现在继续来看看如何通过 jQuery Mobile 给按钮添加图标。这很简单，一步到位：添加 HTML5 的数据属性 data-icon。在第一个按钮上添加 data-icon="delete" 属性；第二个按钮添加 data-icon="check" 属性；新加 data-icon="plus" 属性在第三个按钮上；最后一个按钮添加 data-icon="arrow-l" 属性。jQuery Mobile 拥有一系列可供使用的图标，可以在相关的文档中找到它们。

```
<button data-icon="delete">HTML5 Button</button>

<input type="button" value="Input Button" data-icon="check" />

<input type="submit" value="Submit Button" data-icon="plus"/>

<a href="#" data-role="button" data-icon="arrow-l">Link button</a>
```

新按钮的样式呈现在下面的截图中。



可以通过设置 data-mini="true" 属性来让按钮变得小一些，也能通过 data-iconpos 属性来设置图标相对于按钮的显示方位：上、

下、左、右。除此之外，也可以通过设置 data-iconpos="notext" 属性来让按钮只显示图标。效果如下图所示。

在默认情况下，jQuery Mobile 的按钮会占据整个屏幕的宽度。可以通过添加 data-inline="true" 属性来对此进行修改。

```
<button data-icon="delete" data-mini="true" data-inline="true">HTML5
Button</button>

<input type="button" value="Input Button" data-icon="check" data-
iconpos="right" data-inline="true"/>

<input type="submit" value="Submit Button" data-icon="plus" data-
iconpos="top" data-inline="true"/>

<a href="#" data-role="button" data-icon="arrow-l" data-
iconpos="notext" data-inline="true">Link button</a>
```

可以在下面的截图中看到运行的效果，虽然有点杂乱。

设置 data-inline 属性的相应元素都会变成内联元素，和那些以内联方式显示的列表元素显示效果相似。到此为止，所有要做的都基本完成了，但是还有一些有意思的东西值得继续探索，例如对按钮进行分组也是很容易实现的。移除之前区域中添加的 data-inline="true" 属性。接着在这些按钮元素的外部包装一个 元素，并设置 data-role="controlgroup" 属性。

```
<div data-role="controlgroup">

  <button data-icon="delete" data-mini="true" >HTML5 Button</button>
  <input type="button" value="Input Button" data-icon="check" data-
iconpos="right"/>
  <input type="submit" value="Submit Button" data-icon="plus" data-
iconpos="top" />
  <a href="#" data-role="button" data-icon="arrow-l" data-
iconpos="notext" >Link button</a>

</div>
```

现在距离构建一个极具创造力的漂亮的按钮分组已经不太远了。继续为这个按钮分组添加更多一些特效。如果为 "controlgroup" 元素添加 data-type="horizontal" 属性，将会使得整个布局变得一团糟，需要进行处理。一个较好的处理方式就是设置 data-iconpos 属性的值为 "notext"。

最后，就像在之前几节中所看到的，可以通过 data-theme 属性来为按钮添加一些色彩。可以给每一个按钮添加不同的 data-theme 属性（a,b,c,e）来快速实现配色。在这里有意地跳过 d，是因为它的效果看起来和 c 差不多。下面的截图就是所有的效果。

5.9.3　工作原理

要使用本方法，真正需要了解的是那些属性标签。jQuery Mobile 正是通过这些属性将那些 HTML 元素转换成移动设备样式的按钮。这一切都是 jQuery Mobile 自动帮你实现的，只要设置了正确的属性值，无论采用哪种方式创建 submit 按钮，按钮都会正常工作。jQuery Mobile 会触发 HTML5 属性上特定的事件来添加

指定样式到渲染的页面。

5.10 仅通过媒介查询为移动设备设置移动版本的样式表

在本节中，我们将会尝试为那些移动浏览器设置独有的样式表文件。除开 JavaScript，在客户端就没有其他方法来判断用户代理信息并选择适当的逻辑来为移动浏览器渲染页面。最接近 K.I.S.S. 原则的莫过于使用媒介查询来实现渲染移动设备的特定样式。

当然，在 JavaScript 中可以有很多种方式来获取用户代理信息，这会在后面的章节中探讨。现在要做的就是编写一个杀手级媒介查询来锁定移动浏览器，然后设置特定的 CSS。在前面的几节中，媒介查询都是放置于样式表中的。但是在本节中有所不同，媒介查询会位于 HTML 的头部链接中。有所改变是好事，不必过分担心。将媒介查询放置在 HTML 指向 CSS 文件链接中的原因是，只在一些特定的情况下才会使用那些 CSS 中的样式。如果要进行移动设备优先的设计或使用类似于 jQuery Mobile 这样的技术时，本方法可以大显神威。

5.10.1 准备工作

打开 IDE 并新建一个 HTML 页面。记得添加一个视窗 <meta> 标签。如果愿意，也可以给 HTML 的正文中添加一些文本信息。

5.10.2 实现方式

在 HTML 页面的 <body> 标签中，添加两个文本段落。并赋予每个段落不同的 class 属性（class="a" 和 class="b"）。对于本节中所要展示的媒介查询来说，这些已经足够。

```
<p class="a">Lorem ipsum dolor sit amet, consectetur adipiscing
elit.</p>
<p class="b">Nulla ante tortor, rutrum eu sollicitudin eget, vehicula
quis sem. Nullam cursus placerat luctus.</p>
```

再回到 <head> 标签。首先添加一个视窗 <meta> 标签，其中包含属性 "width=device-width"。然后为字体添加一些简单的样式（font-size: 100%）。

```
<style>
  html{font-size:100%}
</style>
```

下面通过媒介查询添加指向移动版本 CSS 样式表的链接。样式表链接元素包含 rel="stylesheet" 及样式文件路径。还需要包含启用所链接样式文件的条件。例如添加一个对于 screen 和 max-device-width=320px 的媒介查询。CSS 的链接应该看起来是下面这样的：

```
<link rel="stylesheet" media="screen and (max-device-width:320px)"
href="mobile.css" />
```

在 HTML 页面中不需要再做什么了，在与页面同一目录下创建一个 CSS 文件，并且命名为 mobile.css。打开然后编辑该文件。对于本节来说不需要特别多的配置，一行就已足够。对 class 为 b 的段落设置值为 2rem 的字体大小。REM 的意思是相对 EM，也就是说相对于根字体大小（这里再提一次，以防你跳过了之前关于响应式字体的一节）。

```
p.b{font-size:2rem}
```

现在看看效果如何。在浏览器中打开这个 HTML 页面，然后在移动设备模拟器中也打开这个页面。对比就能发现在移动设备上 class 为 b 的段落的字体大小不同于桌面浏览器中的字体大小。下图就是本方法所呈现的效果。

5.10.3 工作原理

本方法中的媒介查询会在屏幕分辨率小于或等于 320px 的时候生效。分辨率高于 320px 的设备均会忽略这个 CSS 文件链接（但是 CSS 文件还是会被下载）。基于同样的原理，也可以为其他的设备编写相应的媒介查询。

5.11 仅为移动设备添加 JavaScript 功能特效

在前一节中，我们编写了位于样式表链接中的媒介查询。这对于移动设备优先的响应式网页开发来说非常有用。但是如果是为移动设备编写的 JavaScript 代码，例如 jQuery Mobile，就没有必要让桌面的客户端下载这些文件。接下来我们会通过一些 JavaScript 代码来检测移动设备的屏幕尺寸，只给移动设备部署 jQuery Mobile，而不会为桌面设备部署。

5.11.1　准备工作

对于移动设备优先的设计理念来说，诸如 jQuery Mobile 这样的技术是协助实现服务器端逻辑的实用工具。结合服务器端的逻辑才能最大化这些技术的作用。如果你不幸无法访问服务器端的逻辑，可在客户端使用一些技巧进行模拟。

5.11.2　实现方式

如果你没有看过之前关于 jQuery Mobile 的章节，现在需要回过头去了解一下。我们将复用之前某个章节的代码。

打开在前一节中使用了 jQuery Mobile 的一个 HTML 页面。你可以使用 5.9 节中的文件。如果在这一节中你已经开发了一个具有移动设备样式外观的按钮，那就从这个页面开始吧。

之前在该页面中，jQuery Mobile 将那些老旧的无趣的 HTML 按钮转换成非常炫的 jQuery Mobile 按钮。我们所要做的只是在元素中添加 HTML5 的数据属性，剩下的交给 jQuery Mobile 就行了。那如果只是想在移动设备上实现这些效果呢？

如果想只是通过客户端的逻辑代码实现该功能，可能会碰到一些麻烦。现在首要任务就是需要判断是不是移动设备。一种方法就是查询 DOM 元素的用户代理。我在以前看到过有人这样做，但这种方法逻辑非常复杂，非常容易产生 bug。一个替代方案是检测设备的屏幕尺寸大小。绝大多数移动设备视窗都小于 600px，就目前来说，在开发移动应用的过程中使用这个尺寸作为最大尺寸不会有太大的问题。

现在编写一些脚本用以从 DOM 元素中获取屏幕宽度，如果宽度小于 600px，则下载 jQuery Mobile 文件。首先使用 jQuery，使得页面加载后触发特定函数。

```
$(document).ready(function(){
  //
});
```

在函数的内部，编写条件判断语句。如果屏幕尺寸小于600px，实现特定的逻辑。

```
$(document).ready(function(){
  if (window.screen.width < 600){
    //Do something!
  };
});
```

这是一个好的开始，但是还要指定具体的逻辑实现。这里要做的就是下载并运行 jQuery Mobile 脚本。一个好的方式就是调用 jQuery 的 $.getScript() 函数。因此在满足 if 条件的分支中调用该函数，参数是 jQuery Mobile 源文件的 URL。

```
$(document).ready(function(){
  if (window.screen.width < 600){
    $.getScript("http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js");
  };
});
```

现在需要做的就是在移动设备模拟器中打开页面。

5.11.3 工作原理

如果模拟器的虚拟设备宽度值满足要求，那么在 HTML 页面中看到的是通过 jQuery Mobile 渲染的移动设备版本。在桌面浏览器中，无论浏览器窗口大小是多少，都不会加载 jQuery Mobile 相关文件。

jQuery 的 $.getScript() 函数用于加载外部脚本到本页面。使用方法如本节所示，可以在一些特定的条件下加载外部的 JavaScript，并且在成功加载 JavaScript 后进一步执行其中的函数。

第 6 章
优化响应式内容

6.1 简介

本章中的方法涵盖了广泛的主题。在本章中并不需要写代码，但是这些方法可以为你的程序建立功能强大的保护伞。本章会讨论一些用于开发和测试代码的工具，以保证编写的代码会以期望的方式运行。虽然你有可能对该主题提不起兴趣，但是在设计和开发中这些方法对于提升你的技能大有裨益。再充足的信心也不能保证前端开发工程师不犯错，并且随着项目变得越来越复杂，会有更多的东西导致错误。请了解这些方法并尝试使用这些工具，它们将让你工作更轻松，更不易出错。

6.2 使用 IE 开发者工具进行响应式测试

一个出色的响应式设计需要包括对所有主流浏览器的优化设计，这毫无疑问是响应式设计的一大卖点。IE 浏览器的未来版本可能无法支持 HTML5 及 CSS3 中的许多特性，甚至目前支持的一些特性也时常会出问题，这一点 IE 浏览器根本无法掩饰。更让人无法接受的是，IE7、IE8、IE9 的行为各不相同，导致不计其数的用户不能或不想升级他们的 IE 浏览器版本。还有一个问题是相当数量的公司投资的网络软件只能运行在低版本的 IE 浏览器上。其他某些浏览器已经解决了版本更新的问题，比如 Chrome 和 Firefox。IE 浏览器的开发团队确实需要迎头赶上。然而，你肯定希望自己的产品不管在哪个浏览器上都能良好地工作，你需要承担起保证站点在不同的浏览器上兼容的职责。

6.2.1 准备工作

向你的客户和设计师了解当前项目中你需要对 IE 浏览器的用户提供哪种级别的支持。有数种潜在的策略来支持低版本 IE 浏览

器。你需要了解支持低版本 IE 浏览器需要多少额外的工作，花费是多少，谁将为此买单。最终你要考虑的是当你给客户演示他们的新版网站时，客户会不会抱怨说这在他们最爱的低版本浏览器中根本不工作。

第一个问题是：使用 IE 浏览器的 F12 开发者工具都能干什么？答案是：可以在 IE 浏览器中显示源代码并进行调试，并且方便地在不同版本的浏览器之间切换来验证网站行为。

6.2.2　实现方式

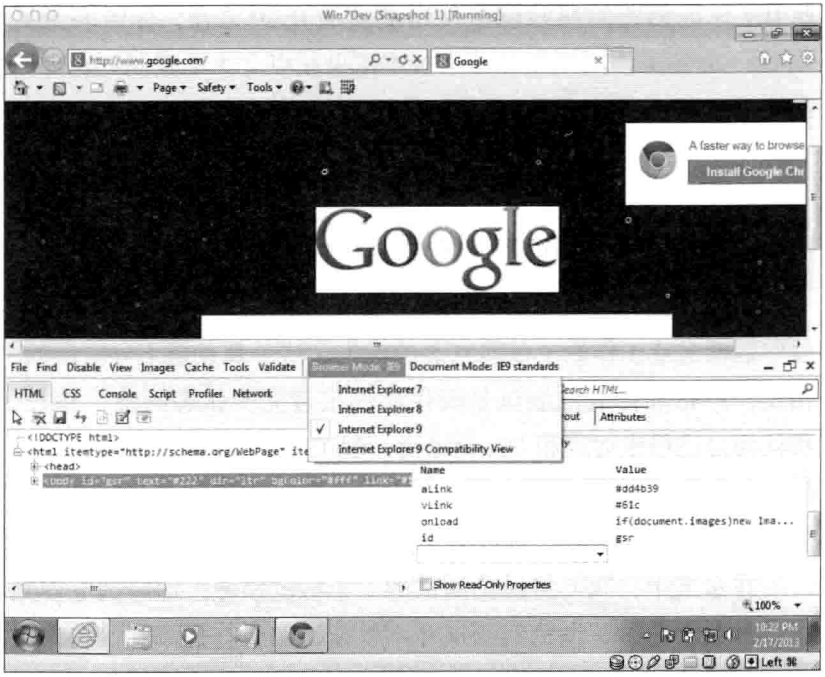
如果你的电脑不是 Windows 操作系统，你将不能使用原生的 IE 浏览器 F12 开发者工具。这并不意味着你可以不在 IE 下做测试就幻想让网站正常工作。有大量的网页和插件承诺它们可以精确地模拟 IE 多个版本中的奇怪行为，但我尝试了许多工具发现，没有一款比原生的 IE 开发者工具出色。经历了许多的失败和挫折后，我发现测试 IE 浏览器唯一可靠的方法是使用虚拟化技术，而不是专门买几台电脑用作测试。我在虚拟机里创建了一些 Windows 系统的实例，安装了不同版本的 IE 浏览器，这是唯一可靠的方法来测试 IE 浏览器的兼容性。如果你想从头学习虚拟化技术，请参考 6.5 节。

请启动 Windows 机器并将 IE 浏览器升级到最新版本，然后了解开发者工具能做些什么。按下键盘上的 F12 键或者单击工具栏上的右上角的齿轮按钮来显示开发者工具。请看下面截图的演示。

第一个有用的功能就是可以单击指针图标并移动鼠标到浏览器窗口中表现异常的元素上。当鼠标移动时，你将看到鼠标指针下的元素周围有一个白色的边框。当白色边框圈住了想检查的元素时，单击鼠标，HTML 面板会将该元素的 HTML 代码显示在左边的窗口中，而该元素的 CSS 样式会显示在右边的窗口中。在 CSS 面板中可以编辑每个元素的 CSS 属性树。

如果想添加 CSS 属性，单击 Attributes 按钮。向下滑动到页面的底部，可以添加新属性的名称和属性值。你可以使用这两个工具来测试不同 CSS 属性的表现或调试 IE 浏览器的一些奇怪行为。

另一个有用的工具是 Browser Mode 选项菜单。该工具可用于不同 IE 浏览器版本间的切换。它既可实时检查工作正确与否，也可以测试指定 IE 版本下的样式。请看下面的截图。



6.2.3　工作原理

根据 MSDN 上的描述，开发者工具描绘了文档对象模型（DOM）在实际中解释页面的方式，而不是所编写的代码。

6.2.4　更多内容

如果你设计的网站作为内部软件或者作为同一个域中的内部网站来访问时，可能会掉入一个陷阱中，那就是 IE 浏览器将使用 IE7 兼容视图作为默认的渲染视图。

兼容模式是 IE8 中引入的新特性，该特性可以使依照旧标准开发的网站仍可运行在新版浏览器中。

通常浏览器会使用兼容模式渲染内部网站。为了确保依

照 IE7 标准创建的网站能够使用最新的 IE 浏览器，你需要设置 <meta> 标签来使用期望的渲染版本渲染站点。为了强制浏览器始终使用最新的渲染引擎，需要指定下面的 <meta> 标签。

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
```

6.3 浏览器测试——使用插件

测试在任何开发流程中都是非常重要的。有人认为测试是工程质量低劣或工作处于危险状态的标志，这个观点是十分错误的。相反，严格和彻底的测试是确保软件接近完美状态的唯一途径。我认为自己很幸运能和 QA 测试者一起工作。QA 测试者的工作是测试开发团队开发出的产品。相比于之前测试工作都是我自己完成的（在以前的时候），现在真是相当地奢侈。

在本节中，我们将讨论测试的一个特定领域，即跨浏览器测试。不久之前，跨浏览器测试并不复杂，但很有挑战性。在移动设备上测试 Web 项目的想法并不常见。你不能简单地期望 Web 项目在移动设备上的表现和在桌面浏览器中的表现相近，甚至完全一样。虚拟机中能够安装的桌面设备数量会限制你能够测试的设备数量。虚拟环境中的工具也是有限的，在此条件下通常是只安装旧版本浏览器。还记得那些拒绝使用 IE6 以上版本的顽固的家伙吗？

跨浏览器测试的一个简单方法是拿出你的信用卡，把所有需要支持运行的设备各买一个。但我从没看到有任何人做到了这点，只有我给孩子讲的个别童话中有类似的故事。这并不是为钱而工作的人们愿意采用的解决方案。这促使了网络上一批收费或者免费的跨浏览器测试工具的出现。

6.3.1 准备工作

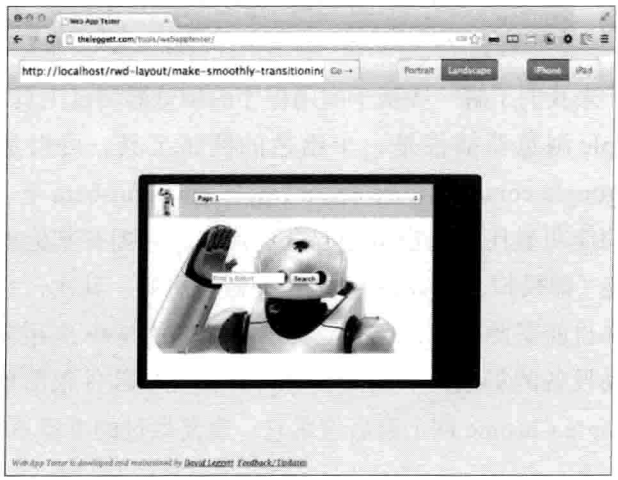
如果你认为这个方法代价太昂贵，先平静下来。不需要去市

场上把所有新的移动设备都买一个，有大量的模拟器能满足你的大多数需求。

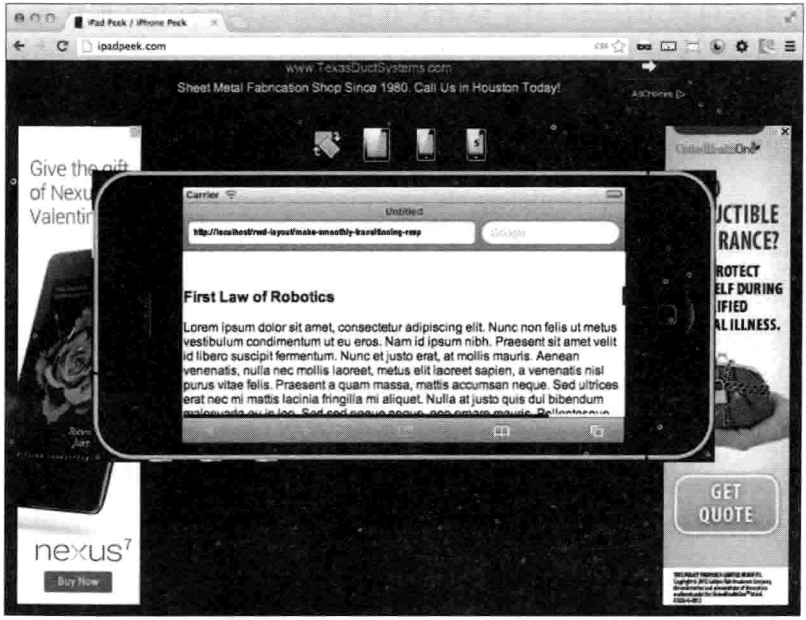
6.3.2　实现方式

我已经从网上找到了一系列免费的工具来用于测试。接下来就来尝试使用这些工具。这需要在浏览器标签中打开之前任意一节中做的响应式网页设计（RWD）项目文件。你需要在每个模拟器的浏览器地址栏中打开该文件。如果你手边没有合适的文件，可以到 Packt 网站下载一个，然后在模拟器中打开。

首先访问 http://theleggett.com/tools/webapptester 来得到一个在线浏览器模拟器。在这里可以通过 iOS 设备的 Web 模拟器测试你的 RWD 网站。该模拟器能读取本地文件，还可以切换横屏模式或竖屏模式，以及选择 iPhone 或者 iPad 版本。它足够简单，无须安装任何复杂的程序或插件就可以使用。如果你紧急需要某样特性，并希望能快速验证，而不希望安装任何东西，那么这个在线模拟器非常适合这种情形。你可以在下面的截图中看到正在运行的模拟器。



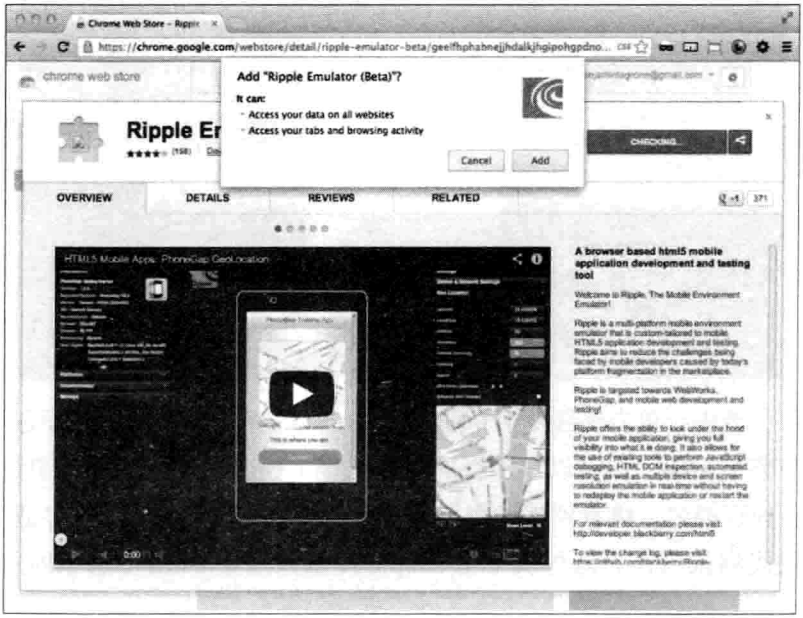
http://ipadpeek.com 上有另一个好用的基于 Web 的 iOS 模拟器。该模拟器也支持横屏模式或竖屏模式，iPad 版本及 iPhone 版本 (包括 iPhone5)。它也能访问本地服务器。我不停提及这点是因为有很多其他的模拟器没有这个功能，甚至包括一些商业模拟器。下面的截图显示了这个基于 Web 的模拟器。



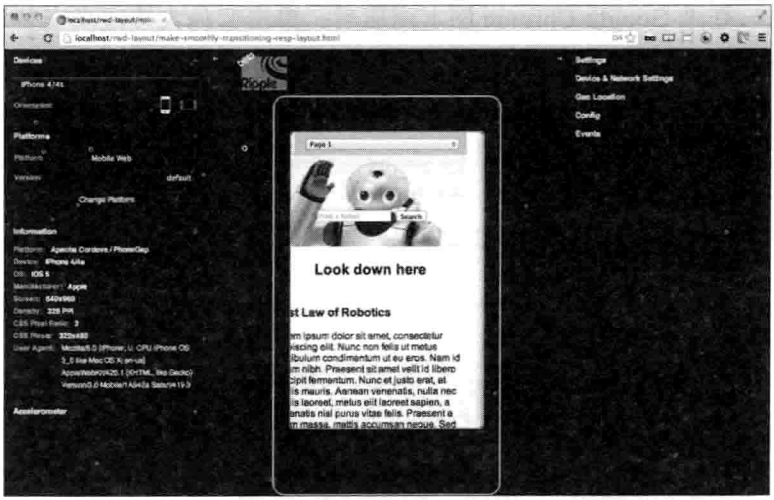
接下来我们了解一些基于应用程序的浏览器测试工具。

Ripple 浏览器插件是一个出色的测试工具。可以从 https://chrome.google.com/webstore/detail/ripple-emulator-beta 下载。该模拟器功能明显比其他模拟器齐全。首先，它拥有其他模拟器拥有的功能（即模拟 iOS 设备），但是却做得更好。其次，它的功能确实比你目前需要的多，但它能帮助测试你的 Web 应用程序与将来的移动设备的集成。下载安装 Ripple 浏览器插件很简单，只需要在 Google Chrome 网上商店检索它。重复做过的事情不难。

登录 Google Chrome 网上商店后，单击蓝色大按钮来安装该插件，参见以下截图。



一旦安装完毕，你将看到一个有蓝色波纹的浏览器按钮出现在 Chrome 浏览器的地址栏旁边。使用浏览器访问你的响应式网站，然后点单击 Ripple plugin 按钮，当菜单弹出来询问你是否启用 Ripple 插件时，选择 Enable 选项。浏览器窗口中的内容会被显示到设备虚拟器中，该虚拟器展示了当前网页的移动版本。另外，你可能会注意到有大量设置选项和工具的菜单栏。让我们来尝试使用其中一些工具。大多数工具的功能并不在本节讨论的范围之内，但你仍需充分利用这些工具，它们能在你开发更先进的移动网站时派上用场。在接下来的截图中你能看到 Ripple 插件众多的设置选项。



首先，单击屏幕左上角的菜单来显示一系列不同的移动设备。在菜单下面，你可以选择横屏模式或竖屏模式。当在不同模拟设备间切换时，将看到信息面板实时更新当前模拟设备的技术参数。完成测试后，只需再次单击 Ripple 按钮并选择 Disable 选项即可关闭模拟器。

该模拟器中有大量出色的工具，不过大部分都不在本书讨论的范围之内。你可以花些时间来探索这些工具，以便应用于以后的移动 Web 项目中。接下来了解下另一个浏览器测试工具。

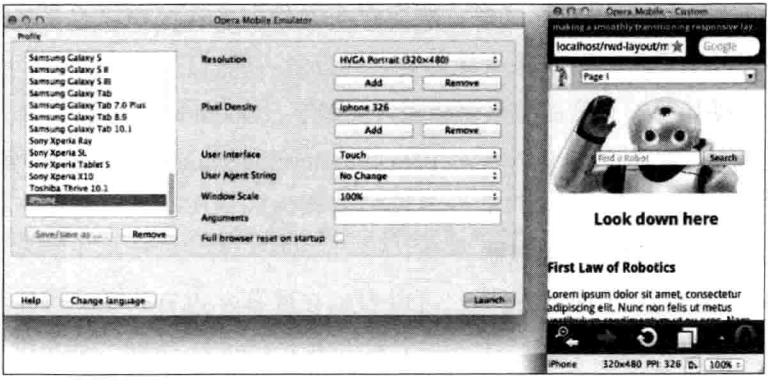
Opera 移动设备模拟器的地址是 http://www.opera.com/developer/tools/mobile。当第一次看到 Opera 浏览器时，我差点就忽视了它，因为它的名字叫歌剧（Opera）。我曾一度忽略它的存在，尽管它是一个相当严谨的浏览器项目。对于移动设备上的浏览器来说，它其实是一个非常好的选择。很高兴最终我尝试使用了它。我惊奇地发现它有许多选项，可以用来模拟相当数量的设备。它是一个出色的移动设备浏览器测试工具，可用于测试多种安卓设备。注意，在此说明安卓设备是因为它只能测试安卓设备。

它虽然只能测试安卓设备，然而它允许你创建和保存自定义的屏幕尺寸和配置。接下来直接安装它并设置一些自定义的屏幕尺寸。

使用你最喜欢的搜索引擎输入关键字：Opera Mobile Emulator。这将引导你到 http://www.opera.com/developer/tools/mobile/ 页面，在该页面下载 Opera 移动设备模拟器到你的操作系统。完成下载和安装后，启动该模拟器。

当应用程序加载完毕，可以看到在屏幕左边有一系列预定义的设置可供选择。选择任意一个然后单击 Launch 按钮。请参见下列截图的演示。

也可以创建自定义的设备参数并保存起来。由于默认没有 iPhone 设备，可以定义一个 iPhone 设备的屏幕。从 Profile 列表中选择 Custom 选项。接下来在 Resolution 下拉菜单中选择分辨率 320 × 480。然后在 Pixel Density 下拉菜单中单击 Add 按钮，输入 326。最后单击 Launch 按钮。也可以单击 Save 或 Save As 按钮来保存设置。iPhone 4 的屏幕尺寸为 640 × 960，iPhone 5 的屏幕尺寸为 640 x 1136。请参见下列截图中的设置。



Opera 移动设备浏览器的一个重要特性是可以在电脑上使用它来调试代码。要使用这个工具，需要从 www.opera.com 下载和安装针对桌面设备的 Opera 浏览器。安装完毕后，打开该浏览器，选择 Menu 下的 Tools | Advanced | Opera Dragonfly。在 OperaDragonfly 窗口的右边，找到并单击 Remote debug configuration 按钮，再单击 Apply 按钮。然后在移动设备浏览器模拟终端的地址栏中输入 opera:debug 然后单击 Connect 按钮。这样就可以调试代码了。

6.4 开发环境——使用免费 IDE

本书中我们经常提及使用 IDE（集成开发环境）来开发代码。IDE 是开发者用于编写和管理代码的工具集。有很多收费的和免费的 IDE 可以用于开发良好的代码。有很多因素会影响你对 IDE 的选择，费用是其中最重要的因素。Visual Studio 要花费好几百美元，还不算额外的自动提示功能的插件费用。只要有人愿意付费使用昂贵的 IDE，那么就说明这个 IDE 是出色的。

6.4.1 准备工作

在本节中我们选择更简单、便宜的路线，即安装一个出色的

免费 IDE。有那么几年，我工作中的角色是科学家，由于 90% 以上的科学家都喜欢 NetBeans，你可能猜测我会使用 NetBeans。我可以告诉你，推测的正确率大概是 90%。

你可能认为一个高级的记事本足以作为构建应用程序的工具。这可能是对的，记事本对于某些应用已经足够。但是使用开发环境更适合大型应用程序，因为其提供很多特性，比如先进的项目组织、自动提示及各种社区开发的针对不同类型项目或特定功能的插件。

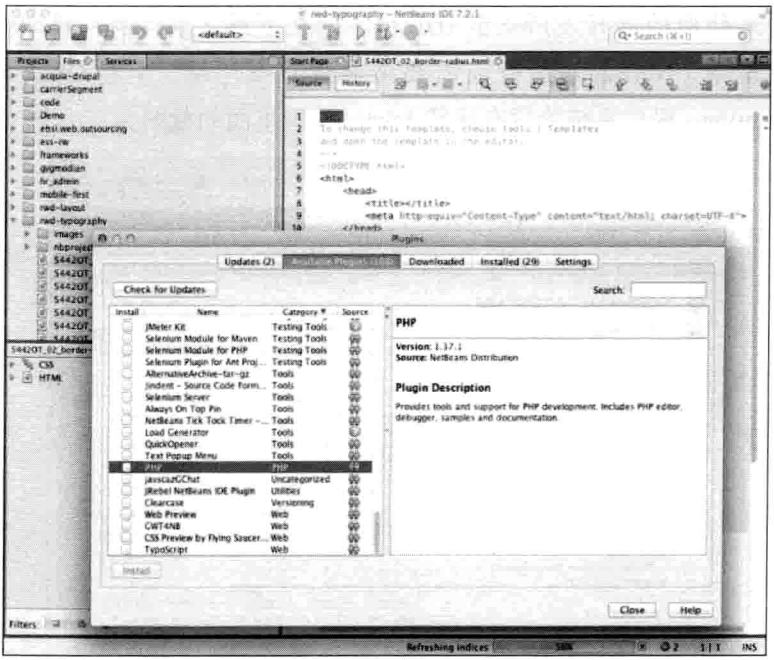
6.4.2　实现方式

可以直接访问 www.netbeans.org 网站，并直接单击橙色的 Download 大按钮来获取 NetBeans。在下载页面中有一系列 NetBeans 的下载选项。你可以选择 PHP 选项，或者“所有”选项来获取前端开发所需的 IDE 包。但在下载之前，有一件事需要澄清，NetBeans 运行前需要预加载 Java 环境，但是 OS X 和 Windows 操作系统均没有预装 Java。请看下面的截图。

如果你已经安装了 Java 开发者工具包，那么就可以继续下载和安装 NetBeans。否则就要去 Java JDK 官网 http://www.oracle.com/technetwork/java/javase/downloads/index.html（如果该 URL 不可用，则搜索 Java JDK 关键字进行下载）。在这里可以下载包含了 JDK 的 NetBeans 最新的稳定版发布包。该文件较大，可以在开始下载后去喝杯咖啡。

解压缩下载的文件包后，安装程序会自动安装 JDK 和 IDE。

安装完毕后启动 NetBeans。在 IDE 的左边面板可以看到文件及项目预览。如果看不到，并且无法打开任何项目文件，这是因为没有激活 Web 开发插件。打开 Tools 菜单并选择 Plugins 选项。在 Available Plugins 列表中选择 PHP 插件并激活，IDE 会要求重启。重启之后，将会在 IDE 的左边看到 Projects 与 File 面板，如下面的截图所示。



6.4.3　工作原理

NetBeans 是基于 Java 构建的集成开发环境，因此需要 JDK 来运行。它是个基础 IDE，可以下载和安装指定项目需要的插件。另外，它是开源的，人们可以开发更多的好用的插件。关于测试、自动提示、编程语言特性及其他方面的插件一直都有人在持续开发，所以尽量勇敢地尝试这些插件来看看是否有助于你的开发工作。

6.5　虚拟化——下载 VirtualBox

虚拟化工具是开发者工具箱中的重要工具。其可用在开发过程中的不同阶段。本节主要讲述在测试阶段的使用。但首先我想说明在测试阶段之前你也可以使用它。可以用你喜欢的操作系统和工具集来设置虚拟机，这就像建立了一个商店，里面支持不同操作系统和工具集的搭配。例如，你想使用 Visual Studio 但是不想使用 Windows 操作系统，你可以启动一个虚拟机，在虚拟机里面安装相应系统和工具来开发程序。你也可以在虚拟机中设置一个 LAMP 栈。

虚拟化是资源密集型计算任务。如果运行的虚拟机里包含一个 IDE、一个 Web 服务器和一个远程桌面连接，它并不会拖垮你的系统，但会使你的系统变慢。我建议如果需要加载多个虚拟机，首先请扩展机器内存。

6.5.1　准备工作

在设置一个新的虚拟机之前，先理解我们这样做背后的原因是什么。第一个原因是 IE 浏览器。还需多说吗？我总是对其“缺乏”知识和经验。每当设计者们需要将漂亮网站运行在 IE 浏览器

的各个版本时，总会充斥着漫天的集体抱怨。只在 IE9 中能正常工作是不够的，我们经常被要求在 IE8 中也能给用户带来较好的体验。

什么造就了 Web 开发者的这种现实？因为人们对升级浏览器版本并不着急，企业甚至情况更糟。安装 Google 分析工具来监视访问自己网页的浏览器类型，可以得到用户使用过时的浏览器的比例图。你可能惊奇地发现 20% 的访问量是来自 IE7，你需要做的就是继续提供相应的支持。在一台电脑上不能同时运行 IE7 和 IE9，所以要使用虚拟化来解决这个问题。

可以使用虚拟化技术来测试你的网站，确保它能在老版 IE 浏览器中优化显示，至少不那么差劲，或者确保网站在移动设备中是响应式的。可以为每个需要测试的浏览器创建一个虚拟机。本节的剩余内容将讲解如何创建一个虚拟机。

6.5.2 实现方式

VirtualBox 是 Oracle 公司开发的一款免费软件。当然也有其他的虚拟化软件，如 VMware，不过它不是免费的。可以在 www.VirtualBox.org 网站的下载页面下载 VirtualBox 软件。

VirtualBox 的安装流程非常简单。如果是 OS X 系统的话，解压缩文件然后拖拽到 Applications 文件夹即可。对于 Windows 操作系统，有好几个选项。为了不引起不必要的麻烦，直接选择可以正常工作的默认选项。OS X 系统和 Windows 系统都会将虚拟机目录建立在当前用户的根目录下。

接下来需要使用操作系统安装盘或镜像文件（ISO）来为虚拟机安装你想要的操作系统。如果你已经有了操作系统安装软件，则单击 Oracle VM VirtualBox Manager 左上角的 New 按钮，这将弹出一个叫做 New Virtual Machine Wizard 的窗口，请看下面的截图。

你需要在接下来的屏幕中输入一个名称及操作系统类型。然后选择分配给虚拟机的内存大小。推荐的最小内存是 192MB。然后会询问你需要创建一个新的磁盘还是使用现有的磁盘。如果是从磁盘或镜像文件安装操作系统，请选择 Create new hard disk。接下来的界面中使用默认选中的值，选择 VDI（VirtualBox 磁盘镜像），最后选择 Dynamically Allocated。

然后系统会询问存放虚拟镜像的目录名以及虚拟磁盘的大小。默认大小是 10GB。接下来是摘要界面，在继续操作之前你可以确认之前的选择信息。到目前为止，还只是相当于一台没有安装操作系统的新电脑。

我们需要启动虚拟机并为其安装 Windows 系统。选择新建的虚拟机，启动后会初始化 First Run Wizard（第一次运行向导）。它将提示你选择安装媒介，可以选择磁盘或 ISO 镜像文件。选择完毕后进入摘要页面，然后就是安装进程。安装非常快，因为这是虚拟驱动器。我们将跳过安装 Windows 桌面操作系统软件的过程，这里没什么神秘的最佳实践，只需保留默认值继续前进。

在写以上这段文字时，我的虚拟机就完成了操作系统的安装。

我说过很快的吧？一旦系统启动起来，就可以使用默认的浏览器版本或升级版本，这取决于你项目的需要。我推荐在不同的虚拟机上安装 IE9、IE8 甚至 IE7。一切准备妥当，就拥有了一个简洁、良好、可运行的 Windows XP 操作系统，请看下面的截图。

现在虚拟机中已经安装了操作系统，可以打开浏览器输入想测试的主机的 IP 地址。如果本地服务器已经运行起来，并且 VirtualBox 网络设置一切正常，那么将在浏览器中看到本地 Web 服务中的页面。

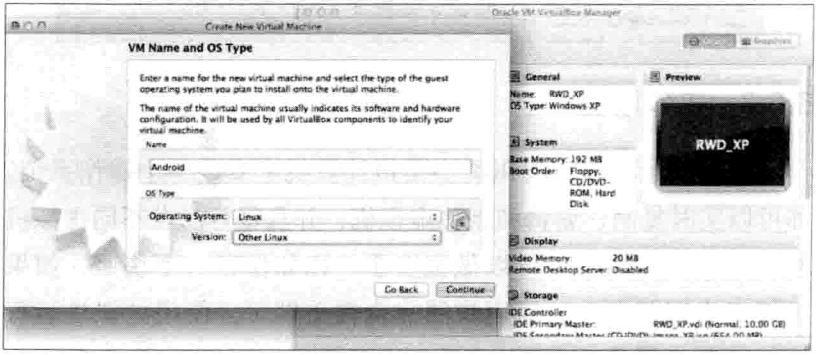
使用虚拟机进行浏览器兼容性测试可以确保你的网站在所有桌面用户前都能良好工作，甚至包括使用 IE7 的用户。

你无须维护 Chrome 或 Firefox 浏览器的多个版本，因为它们都可以自动升级。过时的 Firefox 版本已经成为过去式。

使用虚拟机进行测试可完全覆盖所有桌面设备。在进入下一章之前，先了解下如何使用 VirtualBox 来测试移动设备。

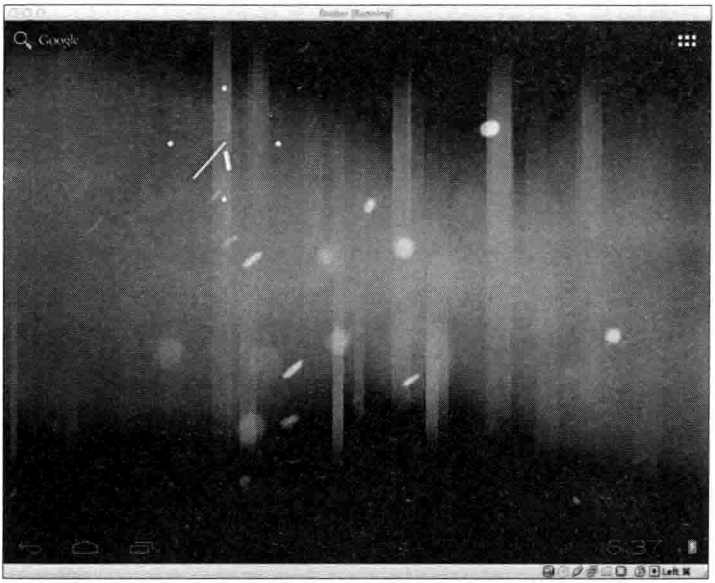
目前网络上已经有安装了安卓系统的虚拟机可供下载。在 http://www.android-x86.org/download 页面有些可供下载的资源。搜索关键字 Android-v4.7z，可以得到下载链接 http://www.vmlite.com/index.php?option=com_kunena&func=view&catid=9&id=8838。它提供了一个链接 http://www.vmlite.com/vmlite/VMLite-Android-v4.0.4.7z 来下载 Android-v4.7z，下载该文件到硬盘并解压缩。

让我们看看当使用 VirutalBox 打开其中一个安卓镜像文件会发生什么。下载完安卓镜像文件后，设置一个新虚拟机。当选择操作系统类型时，从操作系统列表中选择 Linux，在 Version 列表中选择 Other Linux。请看下面演示的截图。



在 Virtual Hard Disk 界面，选择 Use existing hard disk 选项，然后在选择对话框中选取解压后的安卓镜像文件所在的驱动器目录。目录里有个 *.vmdk 文件，选择它并加载到新的虚拟机中，然后单击 Continue 按钮。

经过 Summary 页面之后，安卓模拟器就会被完全设置起来。现在可以在一个真实的安卓模拟器中测试程序了。请看下面的截图。



6.5.3 工作原理

虚拟机允许你在通用类型模拟计算机上安装一个操作系统。你可以实时复制、编辑和删除虚拟机，并且很容易在不同虚拟机间切换。在虚拟机里可以做很多事，比如保存一个快照，如果虚拟机出现问题，只需用快照完全覆盖即可。使用虚拟机运行 Apache 服务器是一项最佳实践，你无须担心你的真实操作系统。

6.6 在 Chrome 中使用浏览器缩放工具

试想你从左到右一次次地拖动浏览器窗口的边角，当浏览器尺寸达到媒介查询的断点时，使用自己最好的可视系统来估算网站是否针对当前尺寸最优化的显示。有个小问题是不知道在哪里会达到断点，因为你得不到当前浏览器的真实尺寸，并且没有信赖的方法将浏览器设置到希望的尺寸。不停拖动浏览器看起来很

蠢是不是？坐在你背后的同事也这样认为。

其实有更好的方式。现在你可以在你同事面前停止拖拽浏览器窗口的滑稽动作。

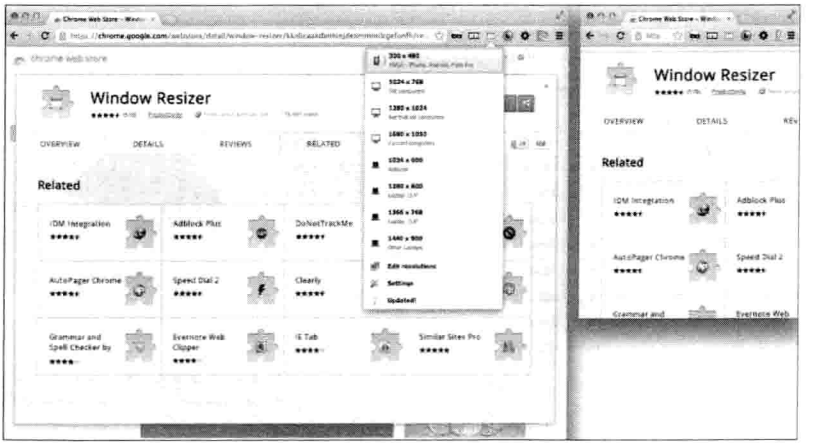
6.6.1　准备工作

有些网站可以调整浏览器窗口到最常见的一些断点所对应的尺寸。然而，这些站点很难找到，并且也不是很可靠。最好的选择是安装出色的浏览器缩放插件。

6.6.2　实现方式

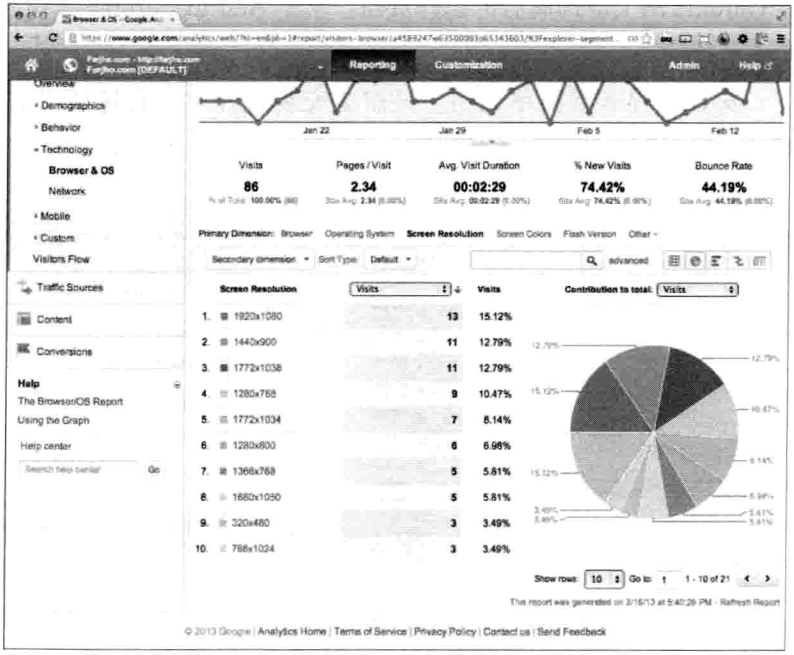
我发现最好的解决方案是 Chrome 窗口缩放插件。在你最喜欢的搜索引擎中搜索 Window Resizer 关键字，然后点击链接会到达 Chrome 网上应用商店的插件页面。单击 Add to Chrome 按钮来安装该插件。

安装过程非常简单容易。只需在安装过程中一直选择“是”即可。请看下面截图中运行的缩放工具。



一旦安装完成，就可以在靠近 Chrome 浏览器地址栏的地方看到世界上最小的浏览器文本。开个玩笑，其实它是个图标。当单击它时，会显示不同浏览器尺寸的下拉列表。这些尺寸基本上是互联网上最常见的一些屏幕尺寸。

如果为你的 Web 项目安装了分析工具，如 Google 分析工具，可以得到一个访问者统计图。正如本节中关心的一样，你可能想知道他们使用的浏览器尺寸。跳到 Audience 标签页展开 Technology 元素来显示 Browser & OS 链接，你将看到用户浏览器的统计分析。在这个页面，将 Primary Dimension: 改为 Screen Resolution。现在可以看到访问网站的用户常用的桌面屏幕尺寸。这个工具能给你在开发网站时针对屏幕尺寸的设计带来一些启示。请看下面的截图。



智能数据分析能提供用户的屏幕尺寸统计数据，但请记住，用户经常只是用部分桌面屏幕来放置浏览器窗口。

回到浏览器缩放插件来，为你的项目尝试一些内置的尺寸来看它表现如何。在响应式设计工具箱中这是很有用的测试工具。

你会看到下拉列表中有个叫做 Edit resolutions 的选项来设置自定义的尺寸。可以将在网站分析中发现的屏幕尺寸添加到这个工具中。基于我网站的访问分析报告，我会添加 1920 × 1080、960 × 1080、1772 × 1038 及 886 × 1038。请看下面的截图中的演示。

6.6.3　工作原理

这些非常有用的工具插件能模拟不同的屏幕分辨率来设置浏览器尺寸。不仅如此，甚至一些优秀的软件有时会发挥不可思议的作用。借助统计分析工具，可以针对网站访问者的屏幕尺寸进行具体的设计优化。

第 7 章
非侵入式 JavaScript

7.1　简介

非侵入式 JavaScript[⊖]的理念非常契合响应式设计的思想。通过展示层与逻辑控制层的分离，所构建的站点具有很高的灵活性。由于移动设备输入方式多种多样，与之绑定的事件也是千差万别的。例如对于桌面浏览器使用普通的 JavaScript 函数绑定事件，而移动版本则会采用 jQuery Mobile 的事件绑定；与此同时，还得使用相同的页面布局模板。以上的需求按照非侵入式 JavaScript 设计理念实现，只是小菜一碟。

7.2　基于非侵入式 JavaScript 编写“Hello World”

交互是响应式设计中一个非常重要的方面。正如桌面设备和移动设备有着截然不同的用户界面，幻想着同一段 JavaScript 的交互代码在所有的设备上都适用是不现实的。其中一个例证就是鼠标悬停事件监听器，即 .mouseover()。触摸屏设备并没有鼠标，因此任何 .mouseover() 事件都有可能触发与 .click() 事件所绑定的函数。对此，解决方案就是在页面中完全移除相关交互脚本。

7.2.1　准备工作

本方法是关于“非侵入式 JavaScript”的。在此，为了避免 HTML 页面中内嵌用于事件处理的脚本，可以创建一个额外的 JavaScript 文件，在该文件中设定一系列的事件监听器来绑定相关事件用以处理用户交互。

⊖ 非侵入式 JavaScript(unobtrusive JavaScript) 是指将 JavaScript 从 HTML 页面结构中抽离的一种设计理念，参见 http://en.wikipedia.org/wiki/Unobtrusive_JavaScript。——译者注

7.2.2 实现方式

本节将从一个简单的示例开始，创建只含有一个按钮和一个警告框的页面文件。许多的 JavaScript 程序都是从测试开始的。首先创建一个事件监听器，然后通过调试的方式来验证警告框弹出逻辑，在本质上，这也是一个测试。现在就来创建这个带 submit 按钮的 HTML 页面。

```
<body>
<input type="submit">
</body>
```

目前这只是一个很简单的实现。现在得到的仅仅是一个基本的 submit 按钮，甚至还不能点击进行提交。下面需要一步步地使按钮变得有实际意义。首先给按钮添加一些自定义的文本，至少在页面加载完毕后能够看到这些文本。在按钮上面添加 value="Say Hello"。对位于 body 中的元素来说，这就已经足够了，接下来在头元素中再添加一个 script 标签：

```
<script></script>
```

在脚本标签里面，需要添加一个事件来触发 JavaScript 执行。页面加载完毕后，下面脚本 $(document).ready(function(){...}); 中的 function 函数体得以运行：

```
$(document).ready(function(){
//do something here
};
```

在函数体内部，将注释 //do something 替换为一个监听器来监听 :submit 按钮的点击事件。而点击该按钮后会触发相应函数，通过特定的方式将 Hello World 呈现在屏幕上：

```
$(":submit").click(function() {
//write "Hello World"
});
```

就目前来说，页面加载完成后，上面创建的 JavaScript 代码就会加载执行，从而监听页面中按钮的点击事件。当 click 事件发生时，对应的函数就会被执行，只是现在函数体仍然是空的。接下来需要做的就是在函数体中实现在页面中显示“Hello World”文本的功能。

在函数体内部，需要在 :submit 按钮所在的父元素中增添“Hello World”文本。因为触发的事件所对应的是 :submit 按钮对象，所以可以通过 jQuery 的 $(this) 来获取该对象的引用。然后通过 jQuery 的 .append() 方法给页面添加“Hello World”文本：

```
$(this).parent().append("Hello World");
```

上面的 jQuery 代码会在 HTML 的 body 元素最后附加“Hello World”文本内容。为了控制所添加的文本位置，将按钮包装在一个父 div 元素中。

在浏览器中打开这个 HTML 页面，测试一下按钮的功能。如果单击按钮后，Hello World 文本并没有出现在按钮下方，那一定是哪里出错了。按照前面所述的步骤来检查，看看到底是哪步出现了问题。

在继续本节之前，我们并不满足于所添加的文本是没有任何样式的。我们希望稍后能够改变这点，给该文本包装在一个段落标签中，并设定 ID 属性，值为 helloWorld。

到此为止，需要的基本功能都已实现：点击按钮然后显示 Hello World。这很不错，但还不够，因为我们总能做得更好，不是吗？

在处理点击 .click() 事件的函数外，新加一个 foo 变量用以存储字符串 Hello World。然后用变量 foo 替换 .append(...) 函数调用的参数 Hello World 文本。用变量替换硬编码的字符串使得可读性

更高，这只是改善函数的一小步。保存并刷新页面，并验证所有功能都依旧工作。

在 body 标签里面，新增定制化页面功能：新加一个 input 元素，将 input 中的输入内容作为参数传给执行脚本。现在给 HTML 添加一个 input 元素，赋予属性 id="bar" 和 placeholder="Enter your name"。

为了能够从输入框中接收到文本内容，在函数体内部添加新的变量 bar。变量 bar 的值为输入框的内容：

```
var bar = $('input').val();
```

接着，对 .append() 方法参数内容作一些修改，使其包含变量 foo、bar 以及一些新的文本内容，所有这些内容均包装在一个含有样式的元素中：

```
$(this).parent().append("<div class='newText'>" + bar + " says " + foo
+ "!</div>");
```

现在刷新页面，可以发现新添加的文本框。试试输入你自己的名字，然后看看有什么惊喜。

很棒的实现，但是仍然不够。现在该花些时间来做一些扫尾工作。那么看看应该避免的场景：提交空字符串以及多次添加 Hello World 内容。

首先需要小心的是在输入框中提交空字符串。在将输入框中的内容显示到 HTML 页面上之前添加一个 if 条件判断，用以检查输入框中内容是否为空。在获取到输入框中内容后，再添加处理输入框内容不为空的逻辑处理，即之前的 append 方法。同样需要通过 else 处理输入框为空的情况。在该逻辑分支中，一样调用 .append() 方法，内容则为提醒用户在输入框中输入文本内容。

```
var bar = $('input').val();
if (bar != ""){
```

```
$(this).parent().append("<div class='newText'>" + bar + " says " + foo
+ "!</div>");
} else {
$(this).parent().append("Please enter a your name!")
};
```

上面的代码添加了验证逻辑，当用户单击 submit 按钮而且输入框内容为空的时候，提醒用户需要输入用户名。现在还剩两个扫尾工作需要完成，再多花几分钟来完成它们吧。

首先，新添加的文本内容应该在每次处理后重置。因此在 if 条件分支的最前面，新添加一行代码，用以移除之前添加的 .newText 元素。

```
$(".newText").remove();
```

最后，在 if 条件分支的尾部，通过 .val() 方法重置输入框的内容为空。当然文本输入框需要一个 ID 属性用以定位，以便重置内容。

```
$('input#fooBar').val("");
```

工作完成！某种程度上来说有一点过度设计了，但是最终完成了一个非常棒的 Hello World 网页。

7.2.3　工作原理

非侵入式 JavaScript 的工作方式为，在页面加载后执行绑定的脚本，并通过事件监听器监听页面上发生的特定事件。本方法可能会要求你调整书写代码的方式，但是将显示层与逻辑交互层分离的好处是显而易见的。

7.3　基于事件监听器创建发光效果的“提交”按钮

绝大多数页面设计方法都会忽视表单的作用，响应式设计更

是如此。一般的非事务性页面都不会含有表单，比如简单的联系我们页面，因此表单设计往往都是事后才考虑的。但是，在电子商务和软件即服务领域，在站点与用户的交互过程中，表单是最为重要的元素。对于响应式设计来说，其不仅仅只是响应式的布局和图像，也包括考虑周到的用户交互。本方法所要处理的场景就是，用户填写表单完毕，正准备提交表单数据的时候。

我们经常会看到这样的场景，一个用户单击提交按钮，但是页面没有任何的变化，似乎什么都没发生（实际上表单信息已经通过 post 请求发送出去），因此用户会尝试多次单击该按钮。在简单的联系我们场景中，这会导致重复生成多个不必要的电子邮件提交请求。但是对于那些提交表单后有长长的事务需要处理的场景来说，多次反复提交可能会使得整个处理过程出现问题。

站在用户的角度，如果单击提交按钮后感觉什么都没有发生，用户可能会认为哪里出错了或者是站点失效了，而这导致的结果就是用户放弃继续操作，对站点的信任度大打折扣。为了防止给用户造成这样的体验，可以采取的方法有很多。其中之一就是提供可视化的暗示，使得用户明白提交按钮点击操作完成，只需等待服务器响应。考虑到事务型请求的处理需要一定的时间，如果预计该操作耗时较长，务必确保用户对此是了解的。用户通常期望因特网上的一切操作都是即时响应的，如果不是的话一定是出了问题。

7.3.1 准备工作

在 7.2 节中，已经创建了带有提交功能的按钮，本节会利用这个按钮。如果现在找不到相关代码，可以在 Packt Publishing 网站（http://www.packtpub.com/）上下载完整版本。

7.3.2　实现方式

首先将 .click() 事件处理函数中的内容提取出来。函数体内的所有内容都剪切后并粘贴到 $(document).ready(function() {...}); 外面。将所有这些内容封装成为另外一个函数，并在 .click() 函数体中调用该新函数。该函数调用需要参数 foo 和 $(this) 对象通过 .attr() 方法获取的 ID 属性值。所以在调用时传入所获取的两个参数值。最后再给输入框添加一个 ID 属性。代码如下所示：

```
$(document).ready(function(){
     var foo = "hello world ";
     $(":submit").click(function(){
          formAction(foo,$(this).attr("id"));
     });
});

function formAction(foo,id){
     var bar = $('input').val();
     if (bar != ""){
          $(".newText").remove();
          $("#" + id).parent().append("<div class='newText'>" +
          bar + " says " + foo + "!</div>");
          $('input#fooBar').val("");
      } else {
          $(".newText").remove();
          $("#" + id).parent().append("<div class='newText'>
          Please enter a your name!</div>");
     };
};
```

首先需要做的就是，从函数中剪切变量 bar，然后粘贴到 .click() 事件的监听函数中。也就是在每一次点击事件发生时均会生成新的变量值。现在需要写一些新的函数了，可以添加一个名为 buttonAnimate() 的函数。修改 .click() 事件监听器中的函数逻辑，添加 buttonAnimate() 的调用，并将其放置在调用函数 formAction() 后。调用函数 buttonAnimate() 的时候，需要变量 bar 作为参数。最后，在声明并调用 formAction() 函数时，添加变量

bar。这里最主要的变化在于，触发点击事件后将输入框的值作为参数传递给函数 buttonAnimate() 和 formAction()。

把已经实现的功能放在一边，现在编写一个新函数来给按钮添加动态效果。是时候休息片刻了。下面暂时将目光从这些 JavaScript 代码转移到 CSS 上面来。

为项目新增一个样式表文件，在样式表文件中添加两个类：.valid 和 .invalid，它们作用于按钮并使其呈现不同的状态：valid 和 invalid。当表单中输入了文本后单击提交按钮的状态为合法，而表单中无任何输入就单击按钮则视为非法。

```
.valid{...}
.invalid{...}
```

在合法状态下，输入框中的文本信息已经通过表单提交。下面就利用 CSS 为按钮增添一些代表合法状态的特效。按钮已经处于激活状态，表明提交已经完成，现在按钮呈现的效果包括边框、阴影、文本阴影、背景色、文本颜色以及圆角，这些应该足够代表表单提交成功后的状态。

```
.valid{
    border:2px solid #000;
    -webkit-box-shadow: 1px 1px 5px 3px #0000ff;
    box-shadow: 1px 1px 5px 3px #0000ff;
    text-shadow: 1px 1px 1px #666666;
    filter: dropshadow(color=#666666, offx=1, offy=1);
    background-color:rgb(150, 150, 255);
    color:#ffffff;
    -webkit-border-radius: 5px;
    border-radius: 5px;
}
```

如果没有任何有效数据就提交表单，也就是非法状态，同样需要特定的 CSS 样式来进行指示。在这种情况下，需要能够在视觉上直观的让用户感知到错误的发生。橘色和红色是在该场景下最能够代表出现错误的颜色。当然，也可以再通过 CSS 给按钮设

置模糊的渐变效果。

```
.invalid{
    border:2px solid #ffff00;
    -webkit-box-shadow: 1px 1px 5px 3px rgb(255, 0, 0);
    box-shadow: 1px 1px 5px 3px rgb(255, 0, 0);
    background-color:rgb(255, 133, 0);
    color:#ffffff; -webkit-border-radius:
    5px; border-radius: 5px;
    -webkit-filter: grayscale(0.1) blur(1px);
    -webkit-transition: border 0.2s ease;
    -moz-transition: border 0.2s ease;
    -ms-transition: border 0.2s ease;
    -o-transition: border 0.2s ease;
    transition: border 0.2s ease;
    text-shadow: 1px 1px 1px #666666;
    filter: dropshadow(color=#666666, offx=1, offy=1);
}
```

本方法所需要的 CSS 都已经编写完毕。接下来，需要编写 JavaScript 将按钮的不同状态和样式联系起来。再来看看之前所创建接收参数 bar、名为 buttonAnimate() 的空函数，现在可以动手实现它了。在函数体中，添加条件语句 if 用以判断 bar 是否为空字符串。如果不是空字符串，则为提交按钮设置名为 valid 的类，反之设置的类为 invalid。添加 invalid 的类的目的就是警示用户出现了错误的状态，需要进行处理。

```
if(bar!= ""){
    $(":submit").addClass("valid");
} else {
    $(":submit").addClass("invalid");
};
```

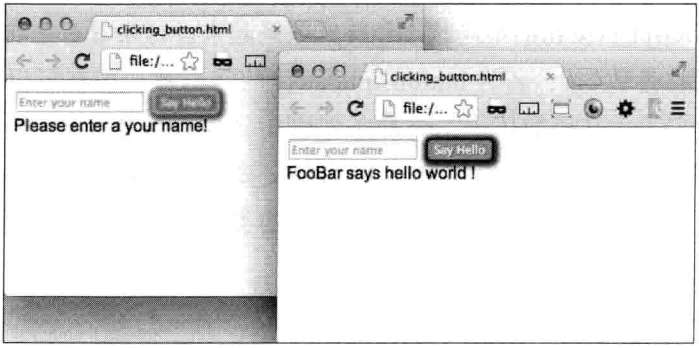
处理非法状态，也就是说，当焦点再次回到输入框后，按钮应该重置到最初的状态。从技术层面来看，需要将新添加的类从按钮上移除。代码如下所示：

```
$('input#fooBar').focus(function(){
    $(":submit").removeClass('invalid')
});
```

最后还有一点需要完成，那就是在 if 以及 else 分支的开始移除之前的类状态。对提交按钮使用 .removeClass() 方法移除与需要添加的类效果相反的类。

```
function buttonAnimate(bar){
     if(bar!= ""){
          $(":submit").removeClass("invalid");
          $(":submit").addClass("valid");
     } else {
          $(":submit").removeClass("valid");
          $(":submit").addClass("invalid");
          $('input#fooBar').focus(function(){
               $(":submit").removeClass('invalid')
          });
     };
};
```

刷新页面，看看效果如何。你所看到的效果应该如下图所示。



7.3.3 工作原理

就开发 Web 应用来说，jQuery 是非常棒的工具，因为它可以通过很少的代码在很短的时间里完成特别繁重的任务。在这个工具出现以前，实现这些功能需要花费更多的时间和精力。而现在，库函数提供了诸如从表单中读取值、给 HTML 追加内容、控制元素的 CSS 类等通用功能。你需要做的只是实现一些 jQuery 方法和配置 CSS，剩下的就都交给 jQuery 来帮你完成。

7.4　制作鼠标悬停后的按钮突出效果

对于响应式设计来说，如果所交付的项目已经非常不错，那么想要超出需求更进一步则是一个很大的挑战。而在页面设计中，按钮元素就是一个很好的机会，能够实现超出预期的效果，并使用户感到惊讶。如果现在还没有实现任何超预期的效果，那就试试 :hover 选择器。在本节中，我们来为按钮实现非常酷的鼠标悬停特效。

7.4.1　准备工作

首先需要了解超出预期交付的误区。实现比需求更多的功能是努力的方向，但是实现那些用户可能不会用到的功能是没有任何意义的。相反可能会导致一个本身非常成功的项目走向失败。

在之前的一节中，我们创建过一个表单，当表单中的按钮被单击后会有相应的动态效果。本节中，还需要复用之前的页面代码。当然你也可以选择从网上下载这些代码。

当然，创建一个含有表单和按钮的页面也不是太难的事情。

7.4.2　实现方式

页面中需要含有两个表单元素，一个输入框和一个提交按钮。正如之前所说，这些在前面都已经实现，当然重新构建这些元素也行。之前编写的 JavaScript 代码需要处理新的用户交互事件，但这并不是必需的。元素 input 的 id 属性为 fooBar，按钮的 id 为 submit。

```
<input id="fooBar" type="text" placeholder="Enter your name">
<input id="submit" type="submit" value="Say Hello">
```

我们需要给默认的按钮增添更多的样式。在 CSS 中为

input#submit 元素配置样式信息，包括蓝色的背景色、白色的字体、8 像素的圆角半径、14 像素的字体大小以及分别为 5 像素和 8 像素的内外边距。这些配置可以通过以下代码做到：

```
input#submit{
     background-color:blue;
     color:white;
     border-radius:8px;
     font-size:14px;
     padding:5px 8px;
}
```

到目前为止，按钮的默认外观已经设计完毕，现在该考虑交互式设计了。在此，我们习惯于通过监听 .mouseover() 事件来取代之前 CSS 的 :hover 选择器。就我个人来说，喜欢监听 form 元素的交互事件，用以判断是否有文本输入。如果存在文本内容，需要一个可视化的暗示表明表单信息有效并可以提交；反之，需要通过强烈的视觉暗示告知用户停止现有操作并检查表单信息的正确性。

首先，如果表单内容有效且能够提交，按钮在鼠标指向的时候变得更加突出并呈现出绿色的效果。在 CSS 中采用覆写 !important 的方式引入绿色的背景色、盒阴影以及文本阴影效果。下面的 CSS 代码片段即为相关设置：

```
.buttonLight{
     background-color:green !important;
     -webkit-box-shadow: 1px 1px 2px 1px green;
     box-shadow: 1px 1px 2px 1px green;
     text-shadow: 1px 1px 2px #666666;
     filter: dropshadow(color=#666666, offx=1, offy=1);
          }
```

其次，如果表单输入内容为空，按钮在鼠标指向其上时将会变红。同理覆写 !important 的方式引入红色的背景色，还有插图效果，以及使用文本阴影来使文本变得模糊。

```
.redButtonLight{
    background-color:red !important;
    -webkit-box-shadow:inset 1px 1px 3px 2px #663535;
    box-shadow:inset 1px 1px 3px 2px #663535;
    text-shadow: 0px 0px 2px #fff;
    filter: dropshadow(color=#fff, offx=0, offy=0);
}
```

CSS 的相关设置都已经完成了。现在需要实现元素的交互特性了。在页面的头元素中，添加标签对 <script>。接着，创建 (document).ready 事件的监听器：

```
$(document).ready(function(){
    //do some things here
});
```

上面的步骤只是开始，还有许多事情需要做。现在就来实现交互特性。在 (document).ready 事件监听函数中，添加事件 .mouseover() 的监听器以及事件 .mouseout() 的监听器。.mouseover() 事件监听器可以处理鼠标在元素上悬停时的场景，同时还将在元素上添加之前创建的 CSS 特效使得按钮呈现动态特性。而 .mouseout() 监听器完成鼠标从元素上移出的相关功能，并将移除之前在 .mouseover() 监听函数中所添加的相关特效。

```
$(document).ready(function(){
    $("#submit").mouseover(function(){
        //do something
    });
    $("#submit").mouseout(function(){
        //do something else
    });
});
```

下面首先实现 .mouseover() 事件的监听函数。就其基本功能而言，有两个功能需要实现。其一，查询表单中 input 元素中的值；其二，依据该值来改变 submit 按钮的样式。对于第一个功能点——查询输入框的值，代码看起来可能像这样：

```
if($('input').val()!="")
    //do something
} else {
    //do something else
}
```

当表单的值不为空时，满足第一个条件分支，需要创建一些新的变量，classtoAdd = "buttonLight" 和 paddingAdd = "5px 8px 5px 9px"。对于另外一个分支，当表单中的值为空时，也会创建同样的变量，classtoAdd = "redButtonLight" 及 paddingAdd = "5px 9px 5px 7px"。在接下来的函数实现中，这些变量将会赋值为按钮的样式。

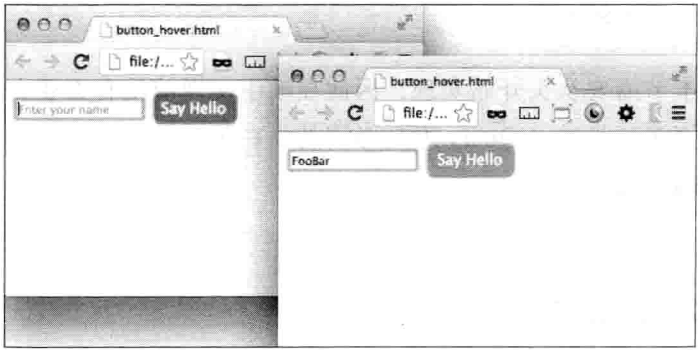
下面的函数实现将会通过 .animate() 方法动态改变按钮的透明度和边距，同时将参数 classtoAdd 值作为样式类添加到按钮上。动态效果应该迅速完成，这里的值设置为 100 毫秒。

```
$("#submit").animate({opacity: 0.7, padding: paddingAdd},
100, function(){
    $("#submit").addClass(classtoAdd);
});
```

对 .mouseover() 事件处理所需操作都已经全部实现，下面要做的是确保事件 .mouseout() 处理逻辑的正确性。同样，动态改变按钮的 position 和 padding 属性，但是整个过程持续的时间将会更长，然后移除相关 class 属性。

```
$("#submit").mouseout(function(){
    $("#submit").animate({opacity: 1, padding :"5px 8px"},
    300, function(){
        $("#submit").removeClass(classtoAdd);
    });
});
```

全部实现就是这些。现在打开页面来看看按钮交互效果，如下图所示。



7.4.3　工作原理

本节中使用了事件监听器，而不是简单的 CSS:hover 选择器，原因在于事件监听能够实现更多的逻辑，而不受 CSS 配置限制，如可以判断表单中 input 元素的值是否为空。基于页面中表单值的状态，就能使用脚本为按钮设置不同的 CSS 样式。这在一定程度上丰富了客户端的逻辑，提供了互动性更好应用程序。

7.5　基于非侵入式 jQuery 改变页面元素大小

本节的意图是构建一个智能的图片元素处理器。通过它可以使相应的图片元素随着屏幕大小的改变而改变。而使用非侵入式 jQuery 就能做到这点。

在之前的某一节中，通过 PHP 服务端逻辑实现了图片缩放的功能。本节也希望达到同样的效果，但是不同的是这次需要客户端来实现，因此可以算得上移动设备优先的响应式设计。

对于移动优先的响应式设计来说，本方法会是一个好的实现方式。例如，如果在页面加载的过程中想要按比例显示图片，或者屏幕太大，需要显示更大尺寸的图片。非侵入式所代表的含义就是采用脚本能够很容易地为图片添加 class 属性，而不用改变页面本身。

7.5.1 准备工作

本节需要从头开始实现，因此在动手实现前没有必要去下载任何资源。当然，jQuery 库是必需的。在页面的头元素中，引入在线的 jQuery 库：

```
<script src="">
```

7.5.2 实现方式

在页面的头元素中设置好 jQuery 库文件的路径后，需要再添加一个脚本元素。在 <script> 标签中，快速添加一些事件监听器和一个实现元素缩放的函数。

在 HTML 页面的主体中，新增一个 div 元素用以包装页面中所有的子元素，并设置其类为 wrap。在该 div 元素中，再添加两个子 div 元素。

在其中一个子 div 元素里插入一张图片。在此，需要准备两个版本的图片用于页面的展示，因此打开图片编辑软件（如果没有安装，可以访问 www.gimp.com 下载）创建一大一小两个版本的图片，图片的内容不限。

分别将两个图片命名为 imagename-small 和 imagename-large。这里我所创建的图片为 robot-small.png 和 robot-large.png。将小图片添加成一个图片元素，然后设置类为 scalable。

```
<img src="robot-small.png" class="scalable" />
```

所需的 HTML 雏形已经完成了，现在需要增加一些简单的 CSS 布局和样式。在头元素中添加 <style> 标签。在标签中，为 div.wrap 元素设置样式，75% 的宽度，同时向左浮动其第一个子元素并设置 50% 的宽度。对于第二个子元素正好相反。可以给两个元素分别设置不同颜色的背景色，以便于更好地观察。最后，

对于元素 img，设置其宽度为 100%，高度为 auto。下面是对应的 CSS 配置：

```
div.wrap{width:75%;}
div.wrap div:first-child{float:left;width:50%;background-color:#ccc;}
div.wrap div:nth-child(2){float:right;width:50%;background-
color:#666;}
div.wrap div img{width:100%;height:auto;}
```

现在，页面的布局也已经准备就绪了，该到编写 JavaScript 的时候了。最重要的函数就是工具函数，用于将旧图片替换为新创建的图片。该函数会在一个独立函数中调用，同时以需要显示的图片作为参数传入。

```
function replaceImage(size){...}
```

在该函数里面，首先需要注意的是判断传入的参数代表大图片还是小图片。创建一个简单的 if 和 else 条件控制语句来实现。

```
if (size == 'small') {…} else {…};
```

如果传入的参数是 small，函数将会使用小图片来替换页面中的图片元素。为了防止不必要的替换，例如使用小图片替换小图片，可以在 src 属性中通过 .indexOf() 方法添加另外的 if 条件语句来验证所含 img 元素的 scalableclass 是否为 large。如果 .indexOf() 方法找到了相应的值，将会返回对应的 img.scalable 对象所在的索引位置。如果索引值大于 1，则满足 if 条件，执行对应的处理逻辑。

```
if($("img.scalable").attr("src").indexOf('large')>1){…}
```

这时会创建一个新的 newImageReplace 变量，变量的值将会作为 src 属性。获取 img.scalable 对象的 src 属性值，用 -small. 替换 -large.（这里结尾是句点形式以防原始的图片可能包含 -large.），然后将新值赋给变量 newImageReplace。

```
var newImageReplace = $("img.scalable").attr("src").replace("-large.",
"-small.");
```

接下来，通过 .attr() 方法来改变 img.scalable 对象的 src 属性，而新值即为之前创建的 "robot-small.png"。

```
$("img.scalable").attr({src:newImageReplace});
```

上面所列的都是 if 条件分支的处理逻辑，剩下需要处理的是 else 逻辑分支。对于父元素上处理的 else 条件分支，如果参数 size 不为 small，函数所做的事情正好和之前所编写的逻辑相反。通过 .indexOf() 方法检查 small 图片是否已经显示，如果是的话，将其变成 large 图片。

```
} else {
     if($("img.scalable").attr("src").indexOf('small')>1){
          var newImageReplace =    $("img.scalable").attr("src").
          replace("-small.", "-large.");
          $("img.scalable").attr({src:newImageReplace});
     };
};
```

现在最重要的功能都已经完成。接着需要创建能够触发这些逻辑处理的函数，当然需要特定的条件作为参数。函数需要足够智能，能够判断屏幕的宽度，因而称之为 measureWindow()。在函数里面，将获取到的屏幕宽度赋予变量 getWindowWidth。如果屏幕宽度太小，例如小于 600px，则需要加载小图片，这时需要调用函数 replaceImage()，参数为 small。如果屏幕宽度大于 600px，参数则为 large。

```
function measureWindow(){
     var getWindowWidth = $(window).width();
     if (getWindowWidth < 600){
          replaceImage("small");
     } else {
          replaceImage("large");
     };
};
```

上面的函数首先会判断屏幕的宽度，然后再根据得到的结果替换图片，当然该函数本身也需要被调用才能生效。但是函数不可能主动调用自己，同样也不可能持续不断地获取屏幕宽度。在目前的情况下，只需要在两个场景下触发函数。第一个场景是，页面加载时，当判断出屏幕很大时，需要使用高分辨率的图片替换默认的小分辨率图片。对于该场景，调用方式如下：

```
$(document).ready(function(){
    measureWindow();
});
```

第二个场景是，窗口大小被人为修改时触发函数。可以通过 .resize() 事件监听器做到这点。

```
$(window).resize(function(){
        measureWindow();
});
```

现在所有的工作都已经完成了，看起来非常简单。打开页面，将页面的大小调整为 600px 以下，通过检查器或调试器来看看图片的 src 属性是否改变。当然基于该方法，也可以设置不同的屏幕大小阈值，这一切都取决于你的需求。

7.5.3　工作原理

本小节采用非侵入式 JavaScript，是客户端处理响应式图片的好例子。当屏幕大小改变后，自动获取新的宽度值，然后根据最新的宽度来更新图片显示。

7.6　基于非侵入式 JavaScript 的密码遮罩

遮罩密码最为常见的方式为，创建一个 input 元素，并设置其类型为 password。对于桌面浏览器来说这是最好的实现。但是在

移动设备上则不一样，在触屏上输入密码出错的概率非常大。因为无法看到所输入的密码内容，所以这些错误常常发现不了。iOS 工程师已经发现这个问题并很好地解决了。方法就是创建一个新的 input 组件，当输入的密码字符变成 * 之前或是输入下一个字符之前，会有一小段时间可见。

本方法所要做的就是，模拟这种处理方式，创建一个自定义的密码 input 元素。

该表单元素也能用以遮罩其他表单项输入。但是需要你确保理解隐藏域中的表单隐式包含需要转换的密码。如果没有指定，这些密码默认是不会进行加密的。遮罩只是防止密码在页面上明文可见。

7.6.1 准备工作

学习本节不需要任何本地文件或资源。唯一要做的就是在头文件中引入 jQuery 的链接。这样就能通过 jQuery 扩展和实现更多功能。

```
<script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
```

7.6.2 实现方式

首先，在 HTML 的主体中创建两个 input 元素。第一个元素含有 type 和 ID 属性，正常的密码输入方式。在表单提交操作中会使用该输入框中经过加密的内容，但是输入框本身是根本不可见的。第二个输入框的 ID 为 altDisplay，但是会被设置成 disabled，因此用户不可能在该输入框中输入任何内容。该输入框会在隐藏的输入框之上显示，使得用户感觉输入框和显示框是同一个。然后，设置一些样式使得密码输入框真正隐藏起来。

所需的 HTML 代码都已经完成了，当然你也可以添加更多的表单元素。

在页面头部，添加包含 JavaScript 的 <script> 标签对，在其中调用 jQuery 的 $(document).ready 函数。然后添加一个监听器用于监听 #password 输入框的 .keyup() 事件。当输入框中输入字符后，相应的事件就会被触发。

但是在真正实现本方法之前会碰到一个棘手的问题。首先，并非所有触发事件的输入都是字符，例如还有 Shift、Tab 和其他的功能键，甚至也包括 Delete 键。每一个按键都能转换成一个数值表示，在控制台通过 e.which 就能看到。然后根据这些数值编写逻辑判断，使得这些非字符型的输入事件不会触发对应的事件。

首先在 if 语句中添加一系列逻辑判断，确保触发事件的输入都是真正的字符。在满足条件后，还需要另外的 if 来检测用户是否键入 Delete (8)。如果不是，继续处理正常的字符输入操作流程，否则还得添加处理 delete keyup 事件的功能（稍后实现）。

```
$(document).ready(function(){
    $("#password").keyup(function(e){
        if (e.which!=16 && e.which!=27 && e.which!=91 &&
        e.which!=18 && e.which!=17 && e.which!=20 ){
            if (e.which!=8){
                //do something for the character key
            }else{
                //Do something for the delete key
            }};
        });
    });
```

当所输入的字符通过验证并成功触发了 keyup 事件后，就会将当前输入值存储到变量 altDisplayVal 和 passwordVal 中。在 #altDisplay 输入框中所显示的值通过正则表达式全部转换为 *，并存储在变量 regAltDisplayVal 中。而在 #password 输入框中显示

的最后一位字符会通过 .charAt() 方法剔除，然后将剩下的值赋予一个新的变量。最后这两个新变量的值会被追加在一起，使其成为输入框 #altDisplay 的显示内容。

```
var altDisplayVal = $("#altDisplay").val();
var passwordVal = $("#password").val();
var regAltDisplayVal = altDisplayVal.replace(/./g,"*");
var passwordValLastLetter = passwordVal.charAt( passwordVal.length-1
);
$("#altDisplay").val(regAltDisplayVal + passwordValLastLetter);
```

上面的代码对于 keyup 事件进行了处理，接下来需要编写处理删除键的逻辑。删除键与其他按键的区别在于它会从字符串的最后移除一位字符。为了处理删除事件，先从 #password 输入框中通过 .charAt() 得到最后一位字符信息，然后将其存储在变量 delLast 中。

然后使用 .slice() 方法截取字符串中除了最后一位的所有信息，存储在变量 delTxt 中。采用正则表达式将所有字符都转换为 *，并存储在变量 regDelTxt 中。最后，将变量 regDelTxt 和 delLast 中的内容合并，显示在输入框 #altDisplay 中。

```
var delLast = this.value.charAt(this.value.length-1);
var delTxt = this.value.slice(0,this.value.length-1);
var regDelTxt = delTxt.replace(/./g,"*");
$("#altDisplay").val(regDelTxt + delLast);
```

这些只是完成了 JavaScript 的处理逻辑。现在可以打开页面来看看两个输入框的效果。在第一个输入框中输入一些内容，该内容相对应地在第二个输入框中显示为 *。可是问题在于页面含有两个输入框元素，似乎和 iOS 样式的密码控件不一样。为了真正做到逼真，需要使 #altDisplay 覆盖在 #password 之上，使得 #password 变得不可见。通过 CSS 就能做到，如下面的设置：

```
div input:first-child{
     position: relative;
     left: 131px;
     background: transparent;
     color: transparent;
}
```

配置完上述的 CSS 后，刷新页面，将会只看到一个输入框。在其中输入的内容会自动转换成星号。

7.6.3　工作原理

本方法实际上并没有改变提交的内容。只是将所需要提交的密码隐藏了起来，然后将隐藏域的密码内容转换为星号字符。本方法可以很好地模拟 iOS 密码输入效果。

7.7　基于事件监听器实现图像阴影的动态效果

这是本书的最后一节，应该有点意思才行。本节将会实现一个响应式的图像，就像在第 1 章中所做的一样，随着鼠标的移动，通过 jQuery 的事件监听器和 CSS3 来实现动态的阴影效果。

本方法非常简单，但是工作的方式仍然为响应式的。图片会随着页面宽度的改变而改变，只是在鼠标移动时需要通过 jQuery 来判断图片的方位和鼠标位置。

7.7.1　准备工作

本节需要使用 jQuery。在新的页面中添加 jQuery 库文件的链接。完成这点后，就可以开始实现了。

```
<script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
```

7.7.2 实现方式

首先，在 HTML 页面的主体中创建一个类为 wrap 的 div 元素。在 div 元素中，添加一个类为 topRight 的图片。接下来就是 CSS 文件。

```
<div class="wrap">
    <img class="topRight" src="robot-small.png"/>
</div>
```

下面在 CSS 中添加一些配置信息。首先，为主体添加 text-align: center 样式。然后，为类为 .wrap 的 div 元素设置 30% 及自动水平扩展的宽度。下面的代码片段就是相关设置：

```
body{text-align:center;}
.wrap{
    width:30%;
    margin:0 auto;
}
.wrap img{
    width:100%;
    height:auto;
    margin:80px 1%;
    border-radius:50%;
    -webkit-border-radius:50%;
    border:1px dotted #666;
}
```

CSS 下面需要做的是根据鼠标的位置通过 jQuery 来变化图片中的类属性。这些类都含有 box-shadow 不同角度的阴影效果。而这些类的名称也分别为 topLeft、topRight、bottomLeft 及 bottomRight。所有的类都设置阴影的偏离值为 5 像素，延伸 2 像素，并且有 2 像素的模糊圆角半径。

```
img.topLeft{
    border-top: 5px solid #666;
    border-right:5px solid #999;
    border-bottom: 5px solid #999;
    border-left:5px solid #666;
    -webkit-box-shadow: -5px -5px 2px 2px #666;
```

```
        box-shadow: -5px -5px 2px 2px #666;
}
img.topRight{
        border-top: 5px solid #666;
        border-right:5px solid #666;
        border-bottom: 5px solid #999;
        border-left:5px solid #999;
        -webkit-box-shadow: 5px -5px 2px 2px #666;
        box-shadow: 5px -5px 2px 2px #666;
}
img.bottomLeft{
        border-top: 5px solid #999;
        border-right:5px solid #999;
        border-bottom: 5px solid #666;
        border-left:5px solid #666;
        -webkit-box-shadow: -5px 5px 2px 2px #666;
        box-shadow: -5px 5px 2px 2px #666;
}
img.bottomRight{
        border-top: 5px solid #999;
        border-right:5px solid #666;
        border-bottom: 5px solid #666;
        border-left:5px solid #999;
        -webkit-box-shadow: 5px 5px 2px 2px #666;
        box-shadow: 5px 5px 2px 2px #666;
}
```

到目前为止所完成的工作已经很出色了。现在是时候来完成 JavaScript 的工作了。在 script 标签对中，创建标准的 $(document).ready 事件监听函数。然后针对 body 添加一个 .mousemove() 事件监听器。在其中创建两个变量 imgHorz 和 imgVert，分别用于记录 .wrap img 的 div 元素的水平和垂直位置。

```
$("body").mousemove(function(e){
        var imgHorz = ($(".wrap img").offset().left);
        var imgVert = ($(".wrap img").offset().top);
});
```

在这些变量被创建后，需要依据变量所对应的位置构建一些条件，用以比较在事件触发时鼠标所处位置是否满足条件。如果为真，在为图像添加类之前就需要移除所有原有 CSS 的类。

```
if(e.pageX < imgHorz && e.pageY < imgVert){
     $(".wrap img").removeClass();
     $(".wrap img").addClass("bottomRight");
};
```

现在还需要添加其他三个 else/if 条件判断用以处理其他的类样式。下面的代码片段展示了所有四个条件判断及其处理逻辑：

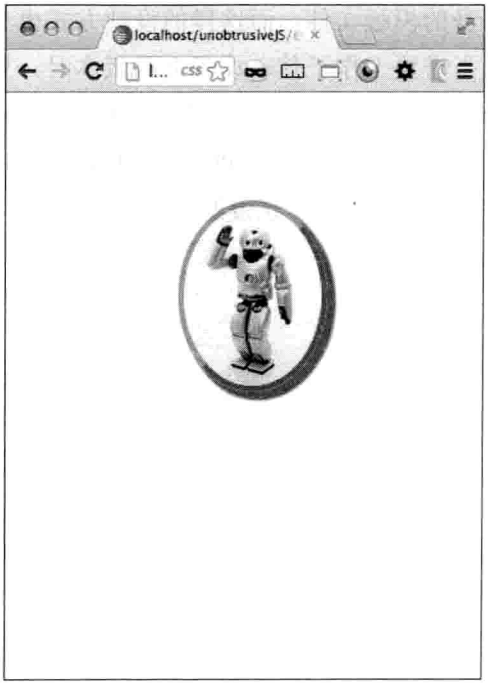
```
if(e.pageX < imgHorz && e.pageY < imgVert){
     $(".wrap img").removeClass();
     $(".wrap img").addClass("bottomRight");
} else if (e.pageX > imgHorz && e.pageY < imgVert) {
     $(".wrap img").removeClass();
     $(".wrap img").addClass("bottomLeft");
} else if(e.pageX > imgHorz && e.pageY > imgVert) {
     $(".wrap img").removeClass();
     $(".wrap img").addClass("topLeft");
} else if(e.pageX < imgHorz && e.pageY > imgVert) {
     $(".wrap img").removeClass();
     $(".wrap img").addClass("topRight");
};
```

这就是 JavaScript 的全部实现。

最后，还需要在不同 CSS 样式之间动态转换效果。为了不增添 JavaScript 代码，需要为 .wrap img 元素添加 CSS 转换（不同的浏览器所对应的转换指令不尽相同）。

```
-webkit-transition: all .5s linear;
-o-transition: all .5s linear;
-moz-transition: all .5s linear;
-ms-transition: all .5s linear;
-khtml-transition: all .5s linear;
transition: all .5s linear;
```

这是相对简单的实现方法，所得到的效果就是当鼠标移动时，在图像元素周围会有一个有趣的阴影特效。下面的截图就是对本方法的呈现。



7.7.3　工作原理

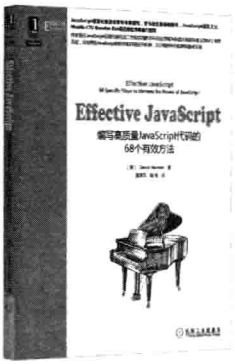
本方法在监听到鼠标移动的 .mousemove() 事件时，探测图片以及鼠标的位置。得到的结果就是随着鼠标的移动，新的阴影效果就会加载在页面之上。现在需要思考一个很重要的问题，对于那些非桌面设备（如移动设备），什么样的事件才能更适合本方法所处的场景。因为移动设备并没有鼠标，所以 .mousemove() 事件永远不会被触发。对于这一点，就需要参考第 5 章来温习一下如何通过诸如 jQuery Mobile 这样的 JavaScript 库完成相应设计。

这里只是通过非侵入式 JavaScript 来构建了一些非常简单的 UI 交互页面。但是我希望这些脚本并不只是可以应用到项目的示例代码，而是想展示如何更好地编写 JavaScript 代码，使得不用

依赖于页面本身的模板文件。而这样的设计方式也契合了响应式设计的思想，当需要工作于移动设备上的版本时，只需要编写所对应的 JavaScript 即可。继续探索下去，非侵入式 JavaScript 能够帮助你构建有更好响应和更流畅转换效果的 Web 应用。

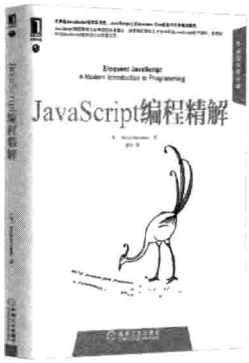
祝愿你的生活快乐幸福，亲爱的朋友！

推荐阅读



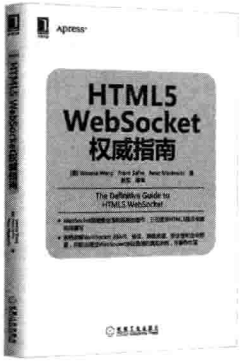
Effective JavaScript: 编写高质量JavaScript代码的68个有效方法

作者：David Herman ISBN：978-7-111-44623-1 定价：49.00元



JavaScript编程精解

作者：Marijn Haverbeke ISBN：978-7-111-39665-9 定价：49.00元



HTML5 WebSocket权威指南

作者：Vanessa Wang 等 ISBN：978-7-111-45641-4 定价：49.00元

构建实时Web应用：基于HTML5 WebSocket、PHP和jQuery

作者：Jason Lengstorf 等 ISBN：978-7-111-43983-7 定价：69.00元

Web前端开发&设计经典

框架篇