

目录

1
章

搭建网页开发环境

015

客户与主机	016
搭建 Web 客户测试环境	017
选择与安装网页制作工具	023
搭建 Web 主机环境	025
选择内容管理系统 (CMS)	029

2
章

HTML5、CSS3、jQuery 基础

030

body 区块(section) 标题

HTML 5 文档由不同的区块构成。

HTML 5 中新增的区块元素

- section
- article
- nav
- aside

中国网页浏览器的占有率

1. Internet Explorer
2. Firefox
3. Mobile Safari
4. Chrome

超文本标记语言，HTML5 基础

031

创建 HTML5 基本文档	031
---------------	-----

编写正文主体：列表	035
-----------	-----

HTML5 大纲算法 (Outliner) 与 Section 元素	036
--------------------------------------	-----

层叠样式表，CSS3 基础

039

控制正文主体 (body) 样式	039
--------------------	-----

控制标题 (h1, h2) 样式	041
--------------------	-----

控制网页背景与字体	042
-----------	-----

脚本语言 JavaScript 基础

045

JavaScript Library，jQuery 基础

058

设计网页动态横幅广告 (Banner)

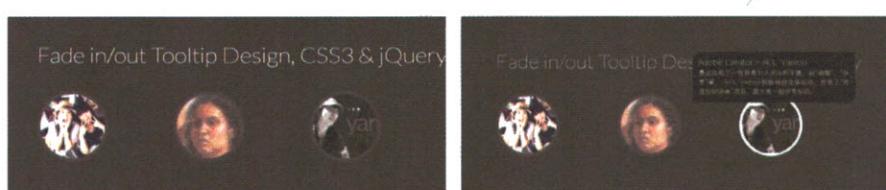
063



设计效果预览	064
设计网页布局	065
编写 HTML5 代码	066
输入单击 Banner 时要链接的网站	066
向 Banner 中添加图片与文字	066
使用 class 属性标识各元素	067
编写 CSS3 样式表	069
控制 body 样式	069
控制 Banner 样式	069
向 Banner 插入背景图片	070
设置横幅广告的 Logo 图像	071
向 Banner 上的文字应用字体	073
设置 Banner 文字的位置与颜色	074
设计鼠标指针未移动到 Banner 上的 Banner	075
使用 jQuery 播放声音	081
浏览器兼容性检查	085

设计气泡悬浮框

087



设计效果预览	088
设计网页布局	089
编写 HTML5 代码	090

编写 CSS3 样式表	093
控制 body 样式	093
设置标题字体样式	093
控制图片样式	095
控制气泡悬浮框 1：基本样式	099
控制气泡悬浮框 2：位置	100
控制气泡悬浮框 3：添加圆角与尾巴	102
控制气泡悬浮框 4：制作 Transition 动画	104
使用 jQuery 实现淡入 / 淡出效果	109
检查是否支持 CSS3 的 transition 属性	109
使用 jQuery 实现淡入 / 淡出效果	111

5

章

制作 Lava Lamp 滑动导航条

117

Lava Lamp Style Navigation Design

本例 Lava Lamp 风格的滑动导航条设计灵感源于台灯 (Lava Lamp)。
台灯的玻璃瓶体内含有一种特殊的水溶液和蜡烛固体，蜡烛加热后融化，蜡烛融化后的液体温度高，底座接触热熔化变轻，便会往上上升，到了新位置冷却又慢慢下降，不往复。其独特的功能和独特的视觉效果令人心旷神怡，多姿多彩的状态时而灿烂时而暗淡，对热爱火山爆发的朋友。

Home Navigation Lecture Open Lecture DVD WebBlog Contact

设计效果预览	118
设计网页布局	119
编写 HTML 5 代码	120
编写 CSS3 样式表	123
控制 body 样式	123
控制标题样式：应用英文网页字体	123
控制导航条边框样式	124
控制导航条菜单项样式	126
控制导航条样式：渐变	128
指定默认激活的菜单项	130
编写 jQuery 脚本代码	131



声明变量与引用文档对象	132
控制 lava 样式	135
设置 lava 的位置	136
向 lava 应用渐变	138
调整 lava 的位置	138
事件监听与处理	140
创建命名空间 (Namespace)	144
制作插件	145

6 章

制作气泡动画按钮

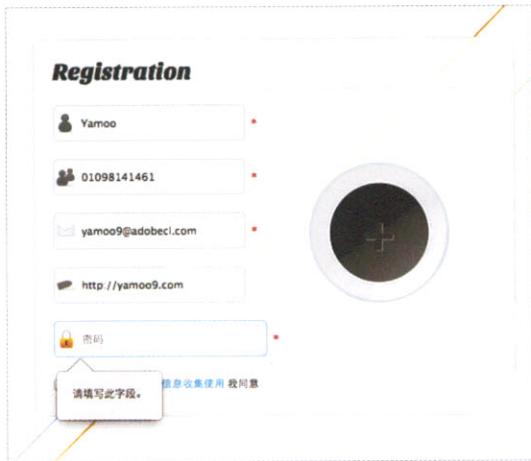
149



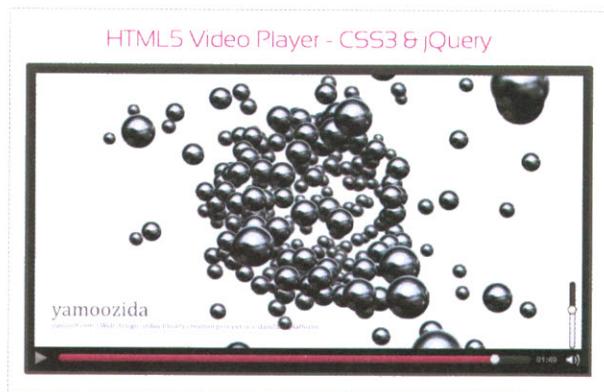
设计效果预览	150
网页布局设计	151
编写 HTML5 代码	152
编写 CSS3 样式表	155
控制 body 样式	155
向 body 添加彩虹渐变	155
从 CSS 代码中提取重复代码，使用 LESS 重写	159
创建 LESS 函数	161
控制 buttons_wrap 样式	162
向 jQuery 按钮添加声音插件	175



设计效果预览	184
设计网页布局	185
编写 HTML5 代码	186
兼容 IE 8 之前版本的代码	186
应用 Web 字体与调用外部文件	188
编写网页结构代码	189
编写 CSS3 与 LESS 样式代码	194
控制基本样式	194
向整个选项卡区域与标题指定样式	196
控制选项卡样式	196
选项卡区域样式控制 1：不支持 JavaScript 脚本的情形	199
选项卡区域样式控制 2：支持 JavaScript 脚本的情形	200
编写 jQuery 插件	201
编写使用插件的脚本文件	201
编写插件	202



设计效果预览	210
设计网页布局	211
编写 HTML5 表格代码	212
编写基本结构	212
编写表单代码	213
编写 CSS3 与 LESS 样式代码	217
编写基本样式控制代码	218
为整个表单区域（容器元素）编写样式代码	218
设置标题样式	222
控制表单元素样式 1：标签与输入文本框	223
控制表单元素样式 2：用户使用条款与信息收集	224
控制表单元素样式 3：向输入文本框左侧添加图标	224
控制表单元素样式 4：提交按钮	226
使用 jQuery 编写脚本代码	229



设计效果预览	234
设计网页布局	235
编写 HTML5 视频播放器	236
编写基本的 HTML 代码	236
编写 JavaScript 脚本自动生成的代码	240
编写 jQuery 视频播放插件	242
调用插件 & 设置初始值	243
建视频播放器容器元素	244
编写视频控件动态生成代码	247
设置视频控件按钮	249
设置视频播放进度条	251
制作播放进度条的滑块	252
编写播放条状态更新函数	254
编写声音控制滑块	256
为声音调节按钮编写函数	256
编写 CSS3 与 LESS 样式代码	259
基本样式控制	260
为视频容器编写样式代码	262

视频控件样式 1：整体	263
视频控件样式 2：播放按钮	264
视频控件样式 3：播放进度条	265
视频控件样式 4：计时器	268
视频控件样式 5：声音控件	268
视频控件样式 6：声音调节条	269
设置播放控制条到视频内部，并改变颜色	272

10
章

制作设备感应型网页

277



设计效果预览	278
设计网页布局	279
编写 HTML5 网页代码	280
header (logo 与导航) 与 header_bar	281
contents 区域	282
编写 CSS3 与 LESS 样式代码	285

设置基本环境（下载 normalize.css）	285
编写 response.css 框架	286
桌面显示样式 1：基本样式	290
桌面显示样式 2：header	291
桌面显示样式 3：header bar	297
桌面显示样式 4：contents	299
桌面显示样式 5：footer	302
桌面显示样式 6：设置拖选区域的背景色与文字颜色	303
为平板电脑编写显示样式	304
为智能手机编写显示样式	305
使用 JavaScript 编写固定的导航菜单插件	307

1 章

搭建网页开发环境

从现在开始，你就要做好两个准备，随时准备用大脑思考，用双手敲击键盘。请牢记：键盘敲得越多，你学到的也越多。但在动手敲键盘写网页代码之前，我们需要先搭建代码测试环境。在本章中，我们将一起学习的内容有：搭建网页开发环境、选择并安装网页编写工具，以及使用内容管理系统（CMS，Contents Management System）快速制作网页的方法。你的心是否已经开始怦怦乱跳了呢？好，下面让我们一起开始学习之旅。

客户与主机

客户(Client)指的是什么呢？不就是顾客、委托人的意思吗？是的，是这样的。客户就是指顾客、委托人。那么，互联网上的客户又是指谁呢？它是指通过网页浏览器使用 Web 服务的人。我们每天都在使用 Web 服务，我们就是客户。在网络环境中，客户就是指使用 Web 服务的人。

主机(Host)是指什么呢？翻一下字典，可以看到其解释为“主人”或“广播电视的节目主持人”。在互联网上，主机(Host)更像是一个广播电视节目的主持人，它通过 Web 向外提供服务内容。

事实上，我们可以把互联网形象地比作“线上交易”。我们在线下交易的是有形的商品，而在互联网上交易的则是数据信息(Data)，包括我们熟知的图片、文字、视频和音乐等。我们使用一幅简图来描述这种“交易”关系，如右图所示。

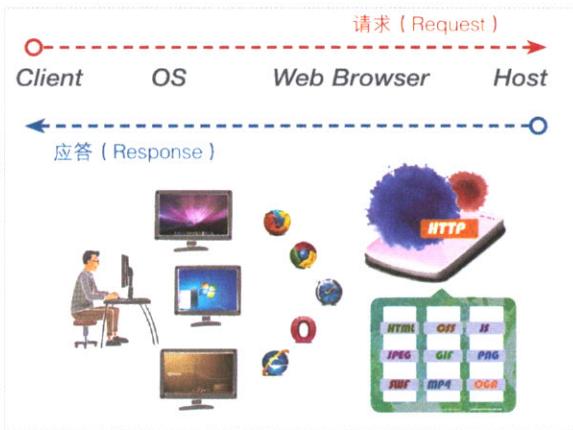


如右图所示，首先客户向主机发送数据请求(Request)，主机接收到请求后，将客户请求的数据返回给客户。图中还有 OS 与 Web Browser 两个部分，它们又是指什么呢？

OS 指计算机操作系统(Operating System)，它为人们使用计算机提供了交互界面，是一款庞大的计算机管理控制程序。目前常见的操作系统有 Windows XP、Windows 7，以及 Mac OS 等。

网页浏览器(Web Browser)是一款安装于 OS 中的应用程序，用于向用户显示网页服务器或文件系统中的 HTML 文件内容，并允许用户与这些文件进行交互。其中最常见的，又最具代表性的网页浏览器有 Microsoft Internet Explorer、Apple Safari、Google Chrome、Mozilla Firefox 和 Opera 等。

整个使用过程，我们可以简单地归纳一下：首先用户启动电脑，运行其中的操作系统，而后运行网页浏览器，在浏览器的地址栏中，输入主机的网络地址后，即可连接到指定的主机上，使用其提供的网络服务。在使用网络时，大部分用户通常先使用搜索引擎（像百度、搜狗、谷歌和雅虎等）查找到所需信息的片段，然后连接到相应的主机上，获取完整的信息。整个过程我们可以使用一张简图来描述一下，如右图所示。



搭建 Web 客户测试环境

在我们提供的众多 Web 服务中，最重要的是网站内容。对于网站内容的设计与显示，在制作网页之前肯定已经有了相关规划。我们是网站内容的提供者，也是获取内容的客户。我们要站在使用者的立场上去审视设计，去阅读、使用内容，这样设计规划出的内容才能更容易地被客户所接受，用户才能更好地使用这些内容。好的内容和好的设计是网站设计第一步。

下面我们首先搭建 Web 客户环境，以方便我们查看网站内容。目前，绝大部分人使用的网页浏览器有 Internet Explorer、Apple Safari、Google Chrome、Mozilla Firefox 和 Opera。

各位可以去相应的浏览器网站下载安装文件，再安装使用。各浏览器的下载地址如下所示。



Firefox <http://www.firefox.com.cn/download/>



Internet Explorer <http://windows.microsoft.com/zh-cn/internet-explorer/downloads/ie>



Google Chrome <http://www.google.cn/intl/zh-CN/chrome/>



Apple Safari <http://www.apple.com.cn/safari/>



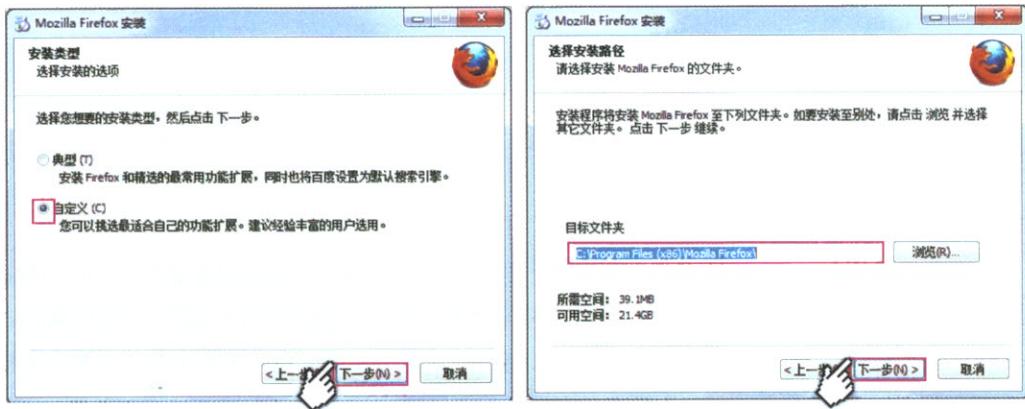
Opera <http://www.opera.com>

下面我们以下载并安装 Firefox 浏览器为例，跟大家一起学习如何下载并安装浏览器软件。如果你已经安装好了 Firefox 浏览器，可跳过本部分内容，直接阅读后续内容。

- 01 访问 Firefox 官方网站 (<http://www.firefox.com.cn/download/>)，下载 Firefox 浏览器安装包。下载完成后，双击 Firefox-latest.exe 可执行文件，弹出 Firefox 安装向导对话框，单击“下一步”按钮。



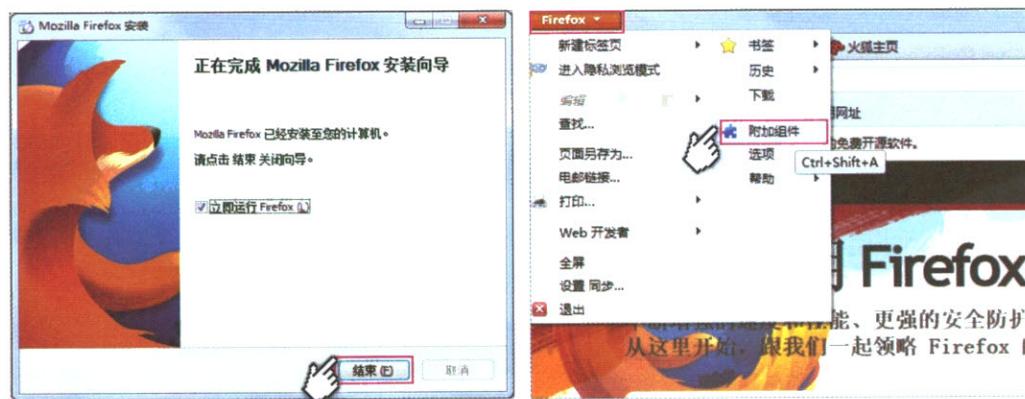
02 在“安装类型”对话框中，点选“自定义”后，单击“下一步”按钮。在“选择安装路径”对话框中，选择“目标文件夹”后，单击“下一步”按钮。



03 在“设置快捷方式”对话框中，取消“在我的桌面”复选框，单击“下一步”按钮。在“概述”对话框中，取消“让 Firefox 作为我的默认浏览器”复选框，然后单击“下一步”按钮，开始执行安装。请注意，若你想把 Firefox 作为默认浏览器，请点选“让 Firefox 作为我的默认浏览器”复选框。



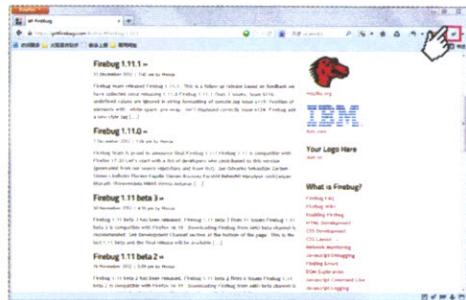
04 安装完成后，单击“结束”按钮，运行 Firefox 浏览器。在 Firefox 浏览器中，点击左上角的 Firefox 按钮，在弹出的菜单中，单击“附加组件”菜单。



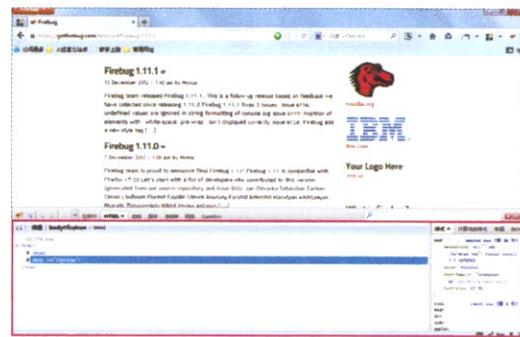
05 在“附加组件管理器”标签页中，在搜索栏中，输入“firebug”，单击Enter按钮，进行搜索。在搜索列表中，单击Firebug右侧的“安装”按钮，进行下载安装。安装完成后，重新启动Firefox浏览器。



06 重启Firefox浏览器后，在地址栏右侧出现Firebug图标（），单击该图标，启动Firebug插件。



07 启用Firebug插件后，在Firefox底部出现Firebug窗口。至此安装设置Firefox及其Firebug插件的工作全部完成。



其他浏览器，像Internet Explorer、Apple Safari、Google Chrome和Opera等，安装设置方法与Firefox是类似的，在此不再赘述。安装完浏览器之后，就能测试用户的网页浏览器环境了吗？NO，还有一个问题需要解决，那就是老版本浏览器的问题。

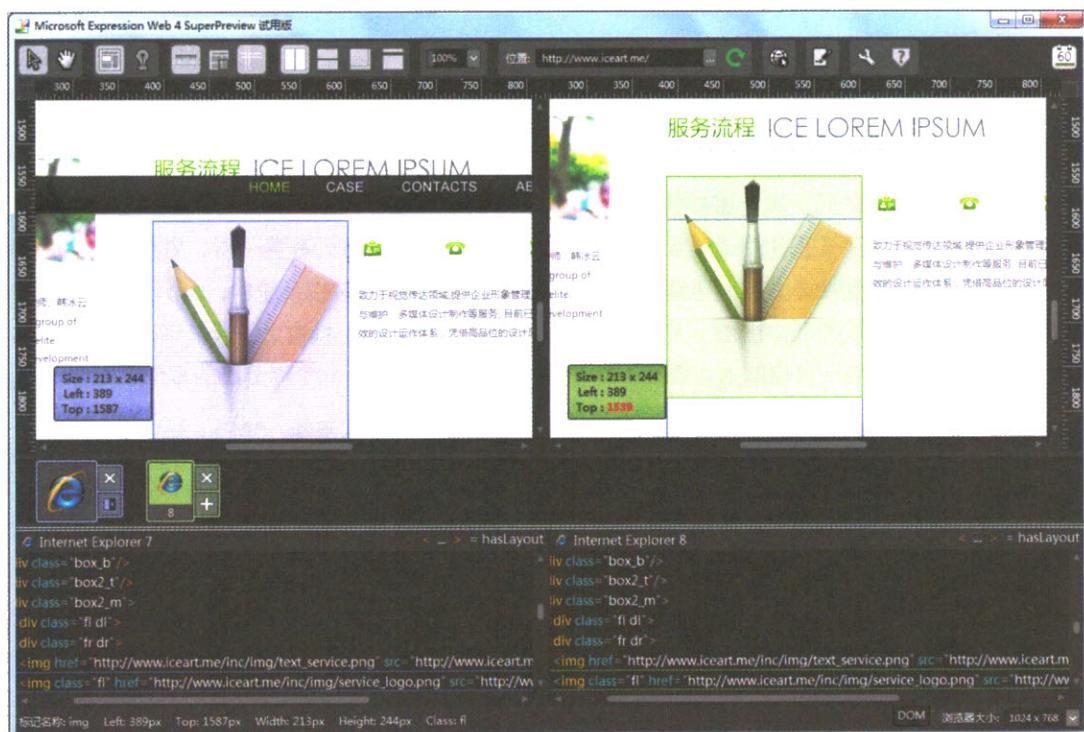
前面，我们介绍的网页浏览器都是最新版本的，但是并不是所有的用户安装的浏览器都是最新版本的。所以，我们还要搭建测试老版本浏览器的环境。就像其他软件程序一样，安装时，软件一般都是提示你安装最新版本，并且老版本与新版本往往无法同时安装在系统中，Web浏览器也是一样的。你即使安装微软的IE浏览器，它也只允许你安装一个版本的。换言之，我们可以把不同种类的网页浏览器安装到同一台电脑上，但是不能在同一台电脑上安装同一个浏览器的多个版本。

解决这个问题的方法多种多样，其中最简单又最有效率的方法有两种，我们一起学习一下。

第一种方法，使用my-debugbar.com公司提供的IE Tester，它是一个免费的Web浏览器调试工具，帮助我们解决了在一个OS中无法安装多个IE版本的问题。该工具可以模拟IE 6/7/8/9/10（IE 9/10版本测试不能在Windows XP中进行，只能在Windows 7中进行）不同版本运行环境，用于调试Web源代码的运行效果。此外，还提供Debug bar，它是一个功能强大的IE插件，从不同角度剖析Web页面内部的细节层次，灵活使用该插件，可以帮助开发者大大减少网页设计中发生的错误，给Web开发者带来很大的便利。



微软也发布了一款免费的网页调试工具 Super Preview，自带有很多元素查看工具，如箭头、移动和辅助线等。在 Super Preview 中，开发者可以同时浏览网页在各版本 IE 中的显示效果。



另一种方法是使用在线的 Web 浏览器测试服务。前面介绍过的 IE Tester 是调试工具，需要安装在本机中使用，而在线 Web 浏览器测试是一种 Web 服务，它根据 Web 浏览器、OS 提供更多的测试环境。

使用这些 Web 浏览器测试服务，一般需要先注册为会员，这多少有些繁琐，而且与安装在本地的测试工具相比，其速度要慢得多。

The screenshot displays three different web-based browser testing platforms:

- Browser Sandbox** (<http://spoon.net/Browsers>): A service that allows you to run any browser instantly from the web in an isolated virtual environment. It features a grid of browser icons for various versions of Chrome, Firefox, Opera, and mobile browsers like Firefox Mobile and Opera Mini. An "ABOUT" section explains the service's purpose.
- Adobe Browser Lab** (<http://is.gd/twbB6Y>): A service provided by Adobe that lets you test your website across multiple browsers and operating systems. It shows screenshots of Internet Explorer 9.0 on Windows and Safari 5.1 on OS X.
- BrowserShots** (<http://browsershots.org>): A service that provides screenshots of websites across a wide range of browsers and operating systems. The interface includes a search bar, a list of recent URLs, and a large table where users can select specific browser and OS combinations to generate a screenshot.

小知识 IE 6 与 Web 浏览器的使用率

在所有老版本浏览器中，最让人头疼的是 IE 6。它是微软在 2001 年随 Windows XP 一起发布的，至今已有 10 多年，几乎不支持所有新技术，但 IE 6 仍然广受欢迎，在浏览器市场中仍然占有不少的份额。事实上，IE 6 的安全性、性能和功能已经明显不能满足现在网络的需要，并且微软公司已经停止给 IE 6 安装补丁了。甚至有些网站还为 IE 6 举行葬礼，希望它尽快消失。

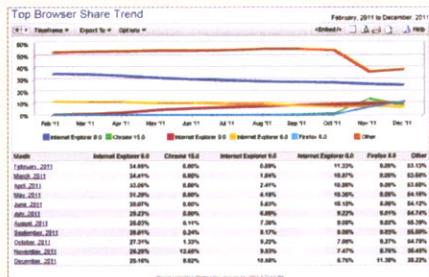


IE 6 葬礼网页 (<http://IE6funeral.com>)

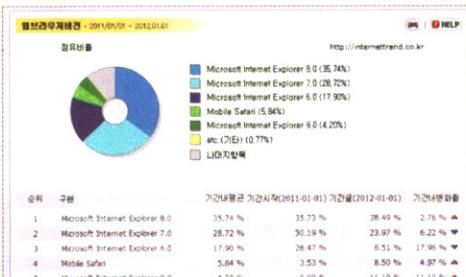


希望 IE 6 占有率为 0% 的网站
<http://www.theie6countdown.com>

尽管如此，还是有很多人在使用 IE 6，其市场占有不容小觑。据统计，截止 2011 年 1 月，其市场占有率为 11%，到 12 月份，占有率达到 6.75%，但仍然超过了 6%。韩国国内 IE 6 的情况也大致类似，2011 年 1 月约占 24.67%，到 12 月份下降到 8.51%，仍超过 8% 的占有率。



(<http://is.gd/PsAwZt>)



(<http://trend.logger.co.kr>)

微软公司为了尽快淘汰 IE 6，采用了强制升级的方式，对用户使用的老版本浏览器进行升级，升级内容如下：

Windows XP: IE6~7 → IE 8

Windows Vista, Windows 7: IE8 → IE 9

选择与安装网页制作工具



编写 Web 文档时需要使用某种制作工具。制作工具多种多样，主要分为 Windows 与 Mac 两大类，具体种类如下表所示。如果你有用习惯的工具，就不需要进行更换了，继续使用就好。

下表列出了一些常用的网页制作工具，并给出了官方地址，请各位参考。

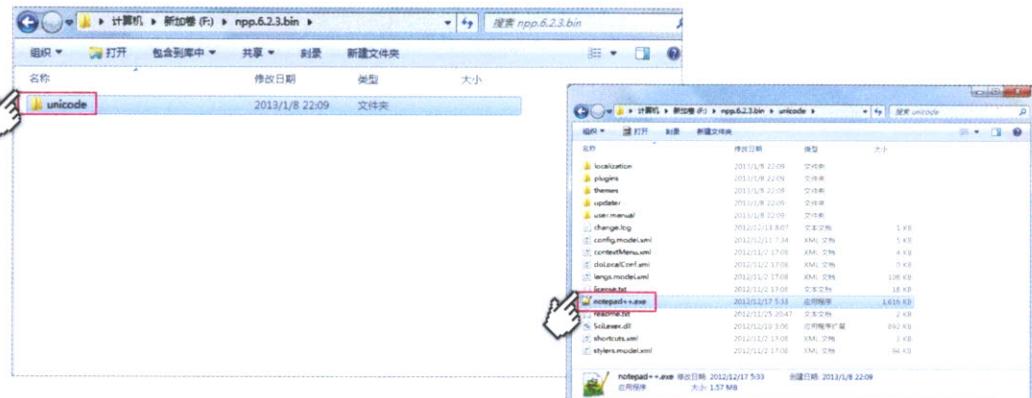
操作系统	网页制作工具	下载地址
Windows	Notepad++	http://notepad-plus-plus.org
	Editplus	http://www.editplus.com
	Expression Web	http://www.expression-web-tutorials.com
	Visual Web Developer	http://www.microsoft.com/express/Web
Windows/Mac	Dreamweaver	http://www.adobe.com/cn/products/dreamweaver.html
Mac	CODA	http://www.panic.com/coda
	Espresso	http://macrabbit.com/espresso/features/edit
	Textmate	http://macromates.com
	BBEdit	http://www.barebones.com/products/bbedit

下面我们以 Notepad++ 为例，介绍下载安装方法，并对使用环境进行一些设置。Notepad++ 是一款 Windows 环境下免费开源的代码编辑器，小巧而方便。如果你已经有上手的编辑工具，你可以继续使用它，而不必再使用 Notepad++。

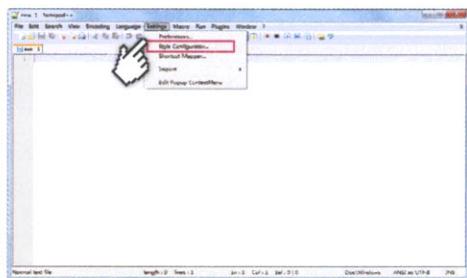
01 访问 Notepad++ 官方网站 (<http://notepad-plus-plus.org>)，单击 Download 按钮，再单击需要的版本，选择 zip package 进行下载。



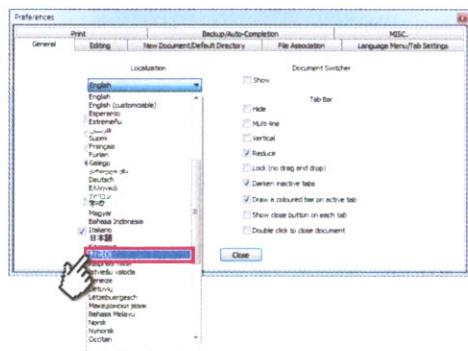
02 下载完成后，解压缩 Zip 文件，双击进入 unicode 文件夹中。而后双击 notepad++.exe，直接运行 Notepad++。



03 在菜单栏中，依次单击 Settings—Preferences 菜单，打开 Preferences 设置窗口。



04 在 Preferences 设置窗口中，设置 Localization 为“中文简体”。



05 然后在“标签栏”区域中，点选“标签显示关闭按钮”。



06 单击“新建”选项卡，在“编码”区域中，点选“UTF-8(无 BOM)”，再点选“应用于打开 ANSI 文件”复选框。在“默认语言”中，选择“HTML”，单击“关闭”按钮，完成首选项设置。至此，Notepad++ 设置完成。



搭建 Web 主机环境

接下来，我们将搭建 Web 主机环境，以便客户进行连接访问。如前所言，我们所说的客户机与主机都是指的计算机，只是客户机提出请求；而主机则对外提供服务。我们可以认为主机就是一台对外提供服务的计算机。

在建造网站时，我们需要一块存放网站的空间，常常使用虚拟主机。虚拟主机可在网络服务器上划分出一定的磁盘空间供用户放置站点、应用组件等，并提供必要的站点功能、数据存放和传输功能。在搜索引擎中，搜索“虚拟主机”，可以搜到许多提供虚拟主机服务的公司。

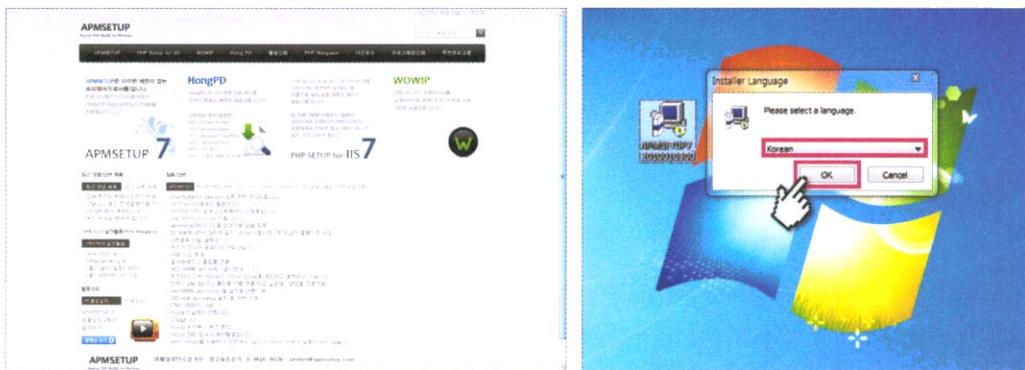
如果只是建设个人网站，可以申请使用网上提供的免费的虚拟主机服务。国内外有很多公司对外提供免费的虚拟主机服务，可以先通过搜索引擎搜索提供免费虚拟主机服务的公司，而后查看具体信息，选择适合自己使用的即可。

向提供免费虚拟主机服务的公司提出申请后，若审核通过，他们将向你提供一块免费空间，供你存放网站文档。在学习本书过程中，你需要使用 FTP (File Transfer Protocol) 是 TCP/IP 网络上两台计算机传送文件的协议，FTP 是在 TCP/IP 网络和 Internet 上最早使用的协议之一，它属于网络协议组的应用层。FTP 客户机可以给服务器发出命令来下载文件，上载文件，创建或改变服务器上的目录。) 软件不断上传更新网页文档，以查看文档更改后的效果。这操作起来多少有些麻烦，为了解决这一问题，我们可以把本机虚拟成一台提供服务的主机，构建 Web 服务器环境。

构建 Web 服务器环境时需要安装 Apache Web Server、PHP 和 MySQL。若分别安装这 3 个软件，安装过程多少有些繁琐，并且与本书所讨论的内容有些远。在这里我们使用一个更方便、更简单的 APMSETUP，它是一个免费软件，使用不受任何限制，并且支持 Windows 环境。



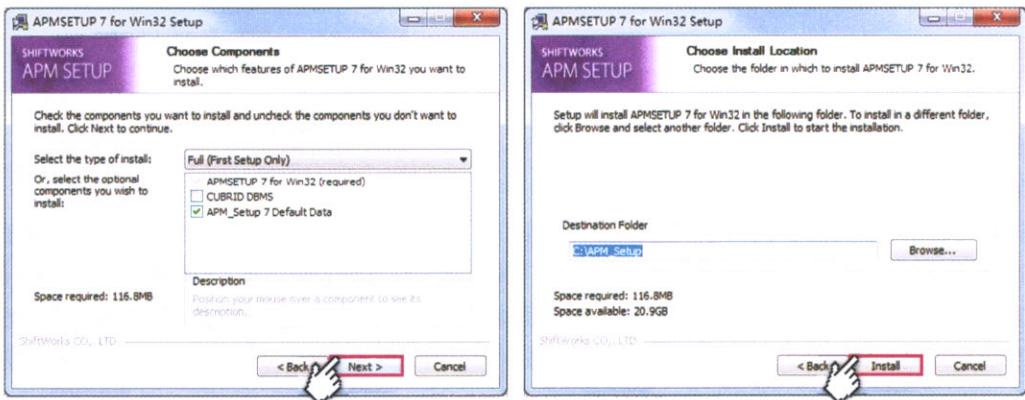
01 访问 APMSETUP 网站 (<http://apmsetup.com>)，下载 APMSETUP。下载完成后，双击 .exe 可执行文件，开始安装。



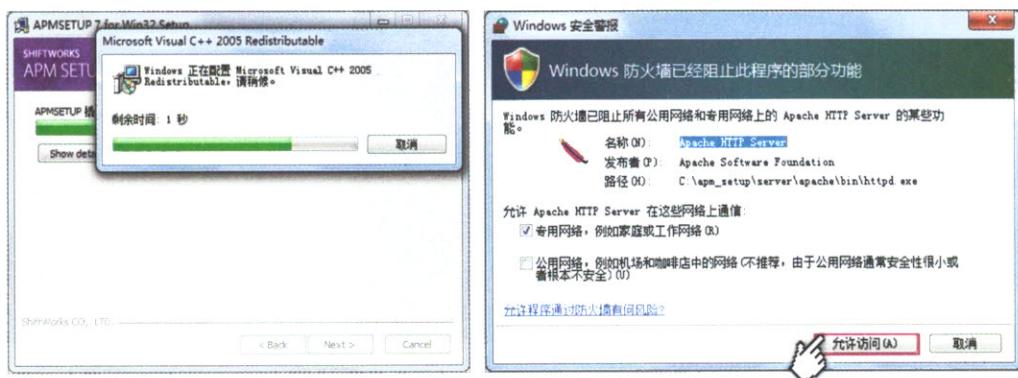
02 弹出安装欢迎界面，单击“Next”按钮。在用户协议窗口中，单击“I Agree”按钮。



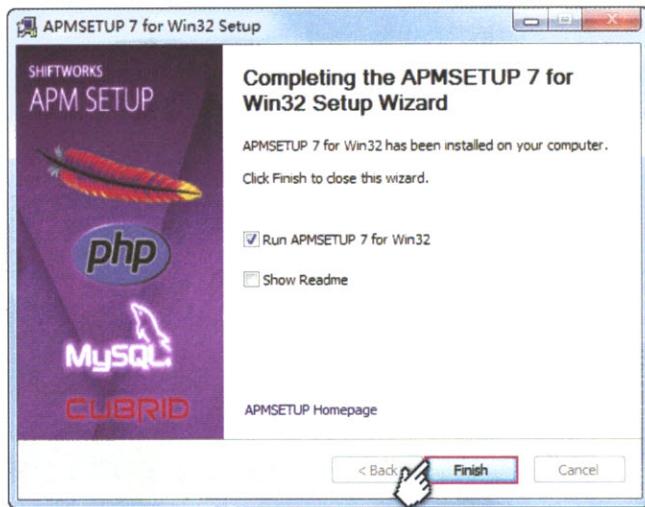
03 在组件选择窗口中，单击“Next”按钮。在选择安装目录窗口中，指定安装目录后，单击“Install”按钮。



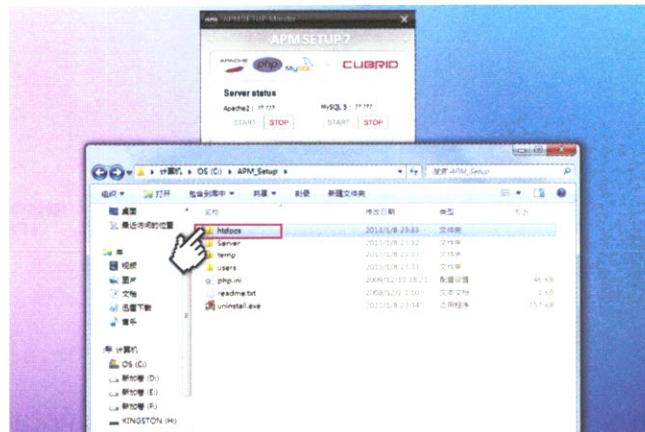
04 在安装过程中会显示Microsoft Visual C++ 2005 Redistributable设置画面。安装完成后，弹出Windows安全警报，询问是否允许Apache HTTP Server访问网络，单击“允许访问”按钮。



05 最后，弹出安装完成窗口，单击“Finish”按钮，完成安装。



06 进入 APMSETUP 安装目录下，会看到一个 APM_Setup 文件夹，双击进入该文件夹，将看到 htdocs 文件夹。在本书内容学习中，所有示例将放入该文件夹进行测试。



07 打开浏览器，在地址栏中输入“localhost”，按 Enter 键，将会看到 APMSETUP 安装成功的页面，表示 APMSETUP 安装成功。



以上，我们一起学习了如何在 Windows 环境下安装 APMSETUP 的整个过程。但是有些读者使用的可能是苹果的 Mac 电脑。我们可以使用 MAMP 软件，帮助在 Mac 中搭建 Web 服务器环境。类似于 APMSETUP，MAMP 内含 Apache 服务器、PHP 安装套件和 MySQL 安装套件，使用方法也非常简单。进入 MAMP.info 网站 (<http://www.mamp.info>) 下载 MAMP 软件后安装即可。

MAMP & MAMP PRO
manage your websites locally

10.6 READY

Support

Home MAMP MAMP PRO Downloads Documentation Blog Forum Bugtracker

MAMP: One-click-solution for setting up your personal webserver

MAMP PRO: Configure an unlimited number of Virtual Hosts, DynDNS, E-Mail, ...

Download now Buy now

选择内容管理系统 (CMS)

内容管理系统 (CMS, Contents Management System) 是一个广泛的称呼，从一般的博客程序、新闻发布程序，到综合性的网站管理程序都可以称为内容管理系统。它具有许多基于模板的优秀设计，可以帮助我们加快网站开发速度和减少开发的成本。我们之所以使用 CMS 内容管理系统是因为网站需要不断更新，不论是网站的设计，还是网站的内容，都需要经常更新，那种认为网站上线就结束的看法是错误的。网站上线后，网站的设计也需要不断维持、维护。对于需要不断更新的网站，网页设计师每次都要手工进行编写修改，这十分困难。使用 CMS 内容管理系统后，可以轻松地更新网站设计，以及网站的内容，这正是 CMS 内容管理系统的便利之处。

若把手工敲代码编写网站看作手工作业，那么使用 CMS 内容管理系统就是机器作业。换言之，使用 CMS 内容管理系统，能够帮助我们更快、更高效地制作网站。

国内外出现了很多非常优秀的 CMS 内容管理系统，其中最具代表性的是 WordPress，它在全世界范围内受到广泛的关注与喜爱。

The screenshot shows the WordPress.org download page for WordPress 3.5. At the top, there's a dark header with the WordPress logo and "China 简体中文". Below it is a navigation bar with links for 首页, 论坛, 文档, 安全, 团队, 等待审核. The main content area has a "欢迎" (Welcome) heading and a brief introduction about WordPress. To the right, there's a "控制板" (Dashboard) sidebar with stats: 65 文章, 2 页面, 5 分类目录, 66 标签. Below that is a note about the theme "Twenty Eleven, 正在使用 8 个小工具". At the bottom, there are download links for "下载 WordPress 3.5 .zip – 6.3 MB" and "下载 tar.gz – 5.8 MB". On the left, there are sections for "下载和安装" (Download and Install) and "服务器环境要求" (Server Environment Requirements), along with a "资源" (Resources) sidebar.

2 章

HTML5、CSS3、jQuery 基础

在本章中我们将介绍网页设计的基本知识，分为 HTML5、CSS3、jQuery 三部分进行讲解，讲解中辅以实例说明。在实例学习过程中，我们要关注 HTML5、CSS3、jQuery 分别有什么作用，它们之间是如何联系的。如果之前你学习过 HTML5、CSS3、jQuery，并且对它们已有相当的了解，那么你可以跳过本章内容。当然如果你是初次接触它们，请认真学习本章内容，本章内容是后续章节的基础。

超文本标记语言， HTML5 基础



HTML 全称为 Hypertext Markup Language，即超文本标记语言，这种语言的语法简洁、结构简单，只要尝试编写几次，就能快速地理解。下面我们一边编写 HTML5 代码，一边学习它。

本书在讲解 HTML 时只讲解 HTML5，如果你想了解学习 HTML 4.01 与 XHTML 1.0 的相关内容，请参考其他书籍。

创建 HTML5 基本文档

HTML 最快的学习方法是边敲代码边学习。当然通过用眼睛看书理解学习相关内容也很重要，但更重要的是手头上要熟悉它。现在，请打开最常用的网页代码编辑器（笔者习惯使用的编辑器是 Espresso，你不必一定要使用 Espresso，只要打开你常用的编辑器即可）。打开编辑器后，新建文档，而后依次单击“文件 – 另存为”菜单，将新建文档保存为 index.html 文档。保存文档时，你可以随意为文档取其他名称，但扩展名一定要是 .html 或 .htm。然后在文档中输入如下代码。

The screenshot shows the Espresso code editor interface. The top menu bar includes Project, Settings, Preview, Action, New Tag, Un/Comment, and Tools. The left sidebar has sections for WORKSPACE, PROJECT FILES (with index.html selected), and PUBLISH. The main workspace displays the following HTML code:

```
<html>
  <head>
    <title>HTML5 基本文档</title>
  </head>
  <body>
  </body>
</html>
```

当使用网页浏览器打开 HTML 文档时，浏览器就会解析 HTML 文档。HTML 是一种标记语言，提供了一套标记标签来描述网页。在 HTML 文档中，标签是指由尖括号 (`<>`) 包围的关键词。在上面的代码中，`<html>`、`<head>`、`<title>` 和 `<body>` 就是 HTML 文档的标签。浏览器在解析 HTML 文档时通过 HTML 标签来识别文档的结构，并以网页的形式显示它们。`<html>`、`<head>` 和 `<body>` 3 个标签分别表示整个网页文档、文档头和文档主体，`<title>` 标签表示文档的标题。

HTML 标签通常是成对出现的，标签对中的第一个标签是开始标签，第二个标签是结束标签。开始标签也被称为开放标签，用尖括号 `<>` 表示；结束标签也叫闭合标签，用带斜杠的尖括号 `</>` 表示。不同的 HTML 标签可以相互嵌套，比如在上面的代码中，`<head>`、`<title>` 和 `<body>` 3 个标签就嵌套在 `<html>` 与 `</html>` 之间。

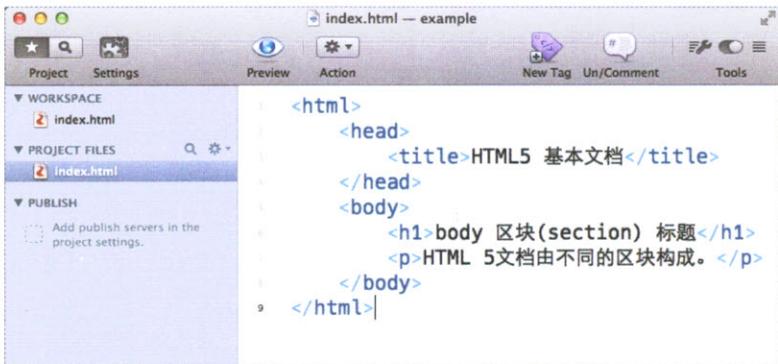
HTML 文档由一系列的标签组成，各个标签具有不同的含义。在上面的代码中，我们使用了 HTML 最常用的标签，它们的含义如下所示。

<code><html></code>	开始 html 文档
<code><head></code>	文档头开始
<code><title> HTML 基本文档 </title></code>	文档标题
<code></head></code>	文档头结束
<code><body></code>	文档主体开始，本部分包含用户看到的主要内容
<code></body></code>	文档主体结束
<code></html></code>	html 文档结束

仔细分析代码，可以了解把握 HTML 文档的组成结构。接下来，我们继续编写代码，为文档主体添加标题与段落。注意在为文档主体添加标题与段落时，要添加在 `<body>` 与 `</body>` 之间。文档主体的标题使用 `<h1>` 标签进行标记，而主体段落则使用 `<p>` 标签进行标记。如下所示，在 `<body>` 与 `</body>` 之间添加两行代码，为文档主体添加标题与段落。

<code><html></code>	
<code><head></code>	
<code><title> HTML5 基本文档 </title></code>	
<code></head></code>	
<code><body></code>	
<code><h1> body 部分 (section) 标题 </h1></code>	文档主体的标题
<code><p> HTML5 文档由不同区块组成。</p></code>	文档主体的段落
<code></body></code>	
<code></html></code>	

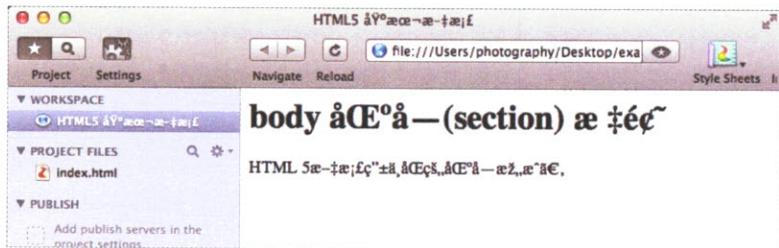
然后，我们启动浏览器，查看一下代码效果。在编辑器中，编写完代码后，按 `Ctrl + S` 组合键，保存 `index.html` 文档。



浏览器能否正常解析我们刚刚编写的代码呢？运行网页浏览器，打开保存好的 index.html 文档，显示下图所示的结果。



在使用浏览器打开我们编写的代码时，有时网页能够正常显示，有时会出现网页乱码的情形，如下图所示。网页乱码是编写网页时经常碰到的问题，必须解决这个问题，网页才能正常地显示给访问的用户。那么，是什么原因引起了网页乱码呢？或者说，网页乱码是如何产生的呢？仔细分析乱码网页，发现页面中的英文能正常显示，但中文出现了乱码现象。由此推断，网页乱码可能是由编码错误引起的。就像看电影必须要有解码器一样，浏览器在解读语言文字时也需要类似于解码器的东西。



事实上，网页乱码问题的解决方法非常简单，只需在文档头中添加一行代码就可以了，即在 <head></head> 标签之间添加一个 meta 标签，并指定字符集属性为 utf-8。

```
<html>
  <head>
    <meta charset="utf-8">          设置html文档编码为UTF-8
    <title> HTML5基本文档 </title>
  </head>
  <body>
    <h1> body区块(section)标题 </h1>
    <p> HTML 5文档由不同的区块构成。 </p>
  </body>
</html>
```

在 HTML 代码中添加完 meta 标签后，保存 HTML 文档，按 F5 键刷新浏览器，再次查看网页显示效果，可以发现网页已经能够正常显示了。UTF-8 是 UNICODE 的一种变长字符编码，又称为万国码，使用该字符编码，可以在同一页面上显示中文简体、繁体以及其他语言（如日文、韩文）。若设置了 meta 字符集后网页仍然乱码，请检查代码编辑器的编码是否已经设置为 UTF-8。



小知识 meta 标签与空标签

细心的朋友会发现 meta 标签与其他 HTML 标签有所不同，它只有开始标签，而没有结束标签。在 HTML 文档中，标签通常是成对出现的，但有一些标签仅有开始标签，这类标签我们称为空标签。meta 就是一个空标签，它向浏览器提供相关网页的元信息，比如文档作者、文档格式和文档编码方式等。

在 HTML 文档中，成对的标签之间会包含相关内容，而 meta 这类空标签没有结束标签，它不包含任何内容，但这些标签往往都带有属性，并且这些属性往往都带有属性值，通过“属性 = 属性值”的形式提供网页相关信息。空标签不包含内容，不需要配对使用，书写时除了采用常规写法外，还可以使用自关闭写法，如 <meta />，常见的空标签包括 img、area、br、hr 和 input 等。

到这里，细心的朋友可能会问：我们刚刚编写的网页文档到底是 HTML5 文档呢，还是普通的 HTML 文档呢？从网页浏览器角度看，我们刚刚编写的网页文档只是一个普通的 HTML 文档，浏览器无法知道它是 HTML5 文档，除非我们通过在网页中插入某行代码告诉浏览器它是一个 HTML5 文档。通常，我们在 HTML 文档的最顶部插入 <!DOCTYPE html> 一行代码，向浏览器声明该文档是 HTML5 文档，浏览器会按 HTML5 文档格式进行解析，并显示在显示器上。

```
<!DOCTYPE html>
```

告知浏览器该文档是HTML5文档

```
<html>
  <head>
    <meta charset="utf-8">
    <title> HTML5基本文档 </title>
  </head>
```

在 <html> 标签中有一个 lang 属性，该属性用于指定文档所使用的语言。比如，我们指定 lang 属性值为 zh，表明该网页使用中文。

```
<!DOCTYPE html>
```

该文档中主要使用的语言为中文

```
<html lang="zh">
  <head>
    <meta charset="utf-8">
    <title> HTML5基本文档 </title>
  </head>
```

<html> 标签的 lang 属性用于通知浏览器当前网页使用哪种语言，以便浏览器在载入文档时采用相应的字符集进行解析。lang 有多个属性值，比如 en 表示英文，en-US 表示美国英语，ja 表示日语。几种常用的语言代码如下表所示，关于各国语言代码的更多内容，请访问 ISO 2 Letter Language Codes 查看。至此，一个基本的 HTML5 文档编写完成。

语言	代码	语言	代码
Korean	KO	English, American	EN
Japanese	JA	French	FR
Chinese	ZH	German	DE

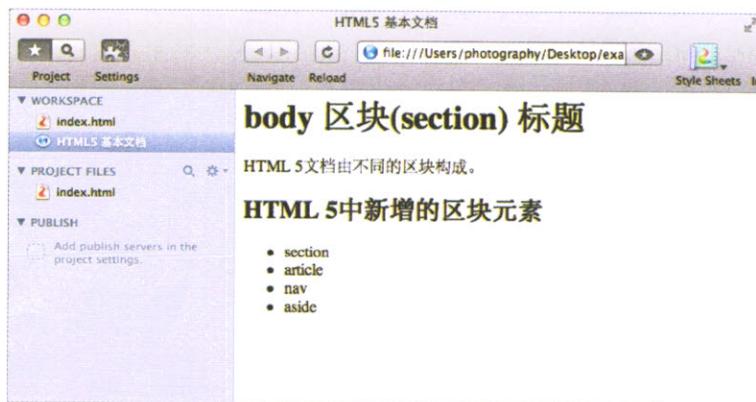
编写正文主体：列表

接下来，我们开始着手向正文主体中添加列表（Lists）。列表是网页设计中常用的元素，常用于显示信息，其结构整齐直观，便于用户理解信息内容。列表分为有序列表与无序列表两种，在有序列表中的列表项有先后顺序，而在无序列表中的列表项没有先后顺序的要求。

如下所示，我们向正文主体插入黄色的 HTML 代码，在代码中使用了无序列表显示各个项目，其列表标签使用 ，子列表项标签采用 ，即各个列表项包含在 之间。

```
<p> HTML5 文档由不同的区块构成。</p>
<h2> HTML5 中新增的区块元素 </h2>
<ul>
    <li> section </li>
    <li> article </li>
    <li> nav </li>
    <li> aside </li>
</ul>
</body>
```

向网页中添加完代码后，再次保存，然后打开浏览器查看显示效果，如右图所示。在无序列表中，各个列表项采用箭头或圆点标出。



与无序列表不同，有序列表使用 标签进行标记，子列表项使用 表示，各列表项内容包含在 标签之间。

```

<h2>中国网页浏览器的占有率</h2>
<ol>
    <li>Internet Explorer</li>
    <li>Firefox</li>
    <li>Mobile Safari</li>
    <li>Chrome</li>
</ol>
</body>

```

如上所示，添加完代码后保存网页文档，打开浏览器，查看网页，可以发现在有序列表的各个子列表项前使用了数字序号 1、2、3、4，这是与无序列表的不同之处。

由于我们尚未为刚刚编写的 HTML 文档添加样式，因此在浏览器中查看时会感觉网页有些单调、乏味。但 HTML 文档的内部结构十分清晰、明了，浏览器与搜索引擎能够准确地理解文档的结构与语义。

HTML5 大纲算法（Outliner）与 Section 元素

从浏览器的显示效果看，我们编写的 HTML5 代码与使用 HTML 4.01 或 XHTML 1.0 编写的代码没什么不同，但实际上它们的内在结构是不同的。在 HTML5 中存在着大纲算法，它允许用户从一个 Web 页面生成一个信息结构目录，以便用户快速浏览网页，定位所需内容，类似于书籍、PDF 或帮助文档的目录。

既然这种信息结构目录不显示出来，那么如何查看它呢？我们可以访问 HTML 5 Outliner 网页服务页面 (<http://gsnedders.html5.org/outliner>)，复制 HTML 源代码后，粘贴到网页的文本框中，然后单击 Outline this！按钮。

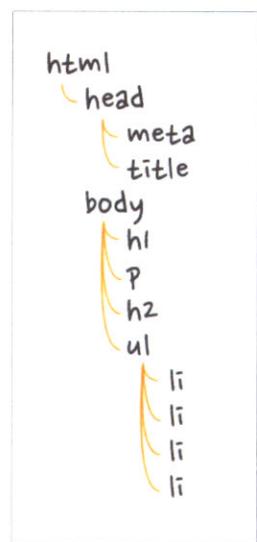
单击 Outline this！按钮后，即可在浏览器中显示出网页的信息结构目录，如右图所示。或许你现在仍然迷惑不解，没关系，我们继续详细地讲解一下其中的含义。

section 区域标题，使用 h2 标签
有序列表开始
包含有序的子列表项

有序列表结束



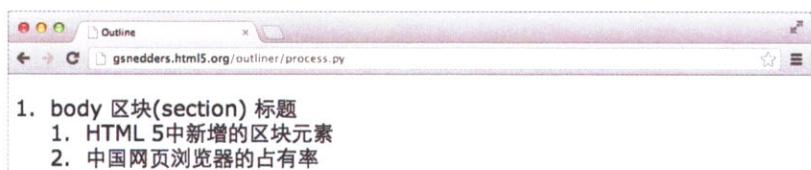
1. body 区块(section) 标题
2. HTML 5 中新增的区块元素
3. 中国网页浏览器的占有率



在 HTML 4.0 与 XHTML 1.0 标准下，我们编写的 HTML 文档会被解析成文档树，文档树是网页文档组织的基本结构，如左图所示。

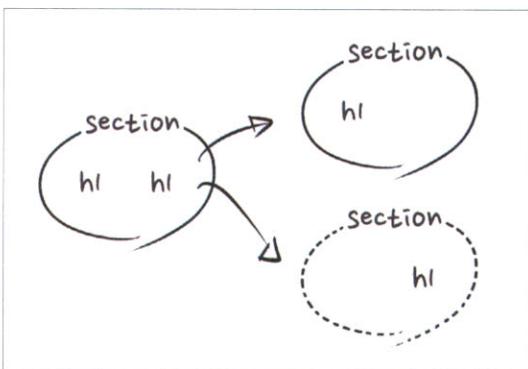
在我们编写的 HTML 代码中，`html` 是文档树的根节点，其下有两个子节点 `head` 与 `body`，在子节点 `head` 与 `body` 下又分别包含各自子节点与孙子节点，最终形成一个结构化的文档。我们将这种关系模型称为文档对象模型（DOM，Document Object Model），它用于导航、访问结构化文档的节点。再次看一下源代码，`body` 节点下包含 `<h1>`、`<h2>`、`<p>` 和 `` 4 个子节点，它们是平行的节点，浏览器将它们视作兄弟节点。

而在解析 HTML5 文档时，浏览器不会将 `<h1>`、`<h2>` 和 `` 视作兄弟节点。再次查看我们编写的 HTML 文档的大纲页面，可以看到“`body` 区块（`section`）标题”由 `<h1>` 标签包含，而“HTML5 中新增的区块元素”与“中国网页浏览器的占有率”分别被包含在 `<h2>` 标签中。但是，`<h1>` 与 `<h2>` 两个标签不是兄弟关系，`<h1>` 比 `<h2>` 拥有更高的级别。



为了更好地表达 HTML 的文档结构、文档语义，HTML5 新增了大量标签，这些标签更好地丰富了 HTML 文档的语义。这些新增的文档结构标签包括 `section`、`article`、`nav` 和 `aside` 等。各个标签的用途明确，要正确使用它们，各标签的用途如下表所示。

新增标签	用法
<code>section</code>	表示页面中的一个内容区块，比如章节、页眉、页脚或页面中的其他部分
<code>article</code>	用于代表页面上独立、完整的一篇“文章”，其内容可以是一个帖子、一篇 Blog 文章、一篇短文或一条完整的回复等
<code>nav</code>	用于定义页面上的导航链接部分
<code>aside</code>	用于定义当前页面或文章的附属信息，通常用作文章的侧栏



在使用 `section` 标签时，通常建议其内部仅包含一个标题。如果 `section` 内部包含了两个以上标题，则这些标题会被自动分割成几个并列或嵌套的 `section`。那么在什么情况下分割成并列的 `section`，在什么情况下会生成子 `section` 呢？这要根据标题级别的不同而不同。比如，在一个 `section` 中有两个标题，如果这两个标题拥有相同级别，则它们会被分割到并列的两个 `section` 中，如左图所示。

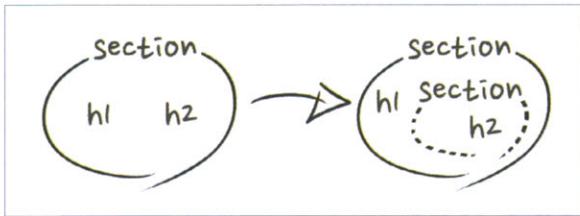
当然，这种分割是肉眼看不见的，但是浏览器会正确解析这种自动生成的结构。

```
<section>
  <h1> 区块(section)标题 </h1>
</section>
<section>
  <h1> 区块(section)标题 </h1>
</section>
```

在文档结构中可见的 section
当包含的标题（同级别标题）大于2个时，
就会生成隐藏的 section 结构
并且分离出一个并列的 section

若同一个 section 中的两个标题拥有不同的级别的标题会被划分到一个新的 section 中，并且将新生成的 section 作为一个子 section。

同样，这个过程用肉眼也是看不见的。

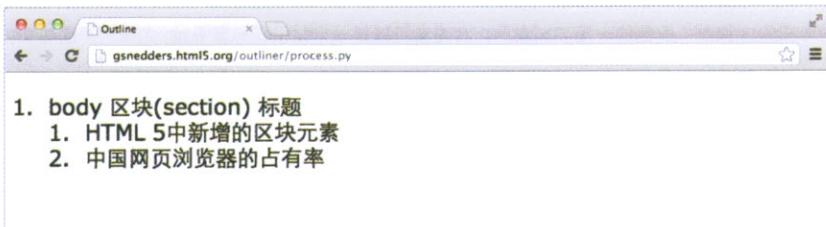


```
<section>
  <h1> 区块(section)标题 </h1>
<section>
  <h2> 区块(section)标题 </h2>
</section>
</section>
```

当一个显式 section 包含2个以上的标题时（2个标题有不同的级别），
自动生成一个隐式的 section

下级 section 在原 section 内部生成

再次返回到我们编写的代码中，在同一个 section 内部存在3个标题，其中第1个标题为 h1 级别，第2、第3个标题是 h2 级别。所以，第2、第3个 h2 级别的标题被划分到一个新的 section 中，并且新生成的 section 被作为一个子 section 看待，最终呈现出如下页面。



由于这些 section 的划分与创建过程是不可见的，因此感觉有些难以理解。不过，如果你先掌握其中的原理，慢慢熟悉就好了。我们刚刚了解学习了 HTML5 的大纲结构，理解了它，再学习其他内容就会容易得多。

小知识 Zen-Coding

Zen-Coding 是一款用于快速开发 HTML 与 CSS 代码的插件，可以安装到多种软件中使用。

层叠样式表，CSS3 基础

现在，终于迎来了令人心动的时刻，我们要对编写的 HTML 文档添加风格样式，对网页进行美化修饰。好，让我们一起开始学习 CSS3。

首先在 HTML 文档的路径下创建一个 css 文件夹，而后在 css 文件夹中创建一个名称为 style.css 的样式表文件，然后输入如下代码。

```
@charset "utf-8";
```

在 CSS 中使用 @charset 语句指定编码格式为 utf-8

在 style.css 文件的开头部分，我们使用 @ 命令，设置字符集为 utf-8，以便在 index.html 文档中调用该样式文件。如果样式文件与 HTML 文档编码不同，就可能会造成部分文本无法正常显示的现象。所以，建议你在 HTML 文档与样式表文件中采用相同的字符集编码，避免出现乱码问题。

控制正文主体（body）样式

对象 {
属性：值；
}

CSS 样式表的编写规则十分简单，遵循“先选对象后修饰”的原则，在进行样式控制之前，要先选择修饰的对象，这个对象，我们称之为“选择器”（ Selector ）。比如，我们要修饰 HTML 文档中的 body 部分，它就是一个选择器，在样式表中输入选择器 “body” ，如下所示。

body

选择要修饰的 <body> 元素，它被称为“选择器”

不同选择器使用的语句有所不同，为了将它们区分开来，我们要在各个选择器后输入一对大括号（ {} ），大括号内的样式控制语句仅对该选择器起作用。

body {

左大括号表示开始编写样式语句

大括号内为样式声明语句

右大括号表示样式声明结束

接下来，我们开始在大括号中输入风格样式控制语句，它由属性与属性值组成，不同属性之间使用分号分隔。首先，我们控制文本字体的大小，控制文本字体大小的属性（Property）为 font-size，设置其值为 75%，75% 是属性值（Value）。在大部分浏览器中，默认字体大小为 16px，此处设置为 75% 后，实际的字体大小应为 12px ($16\text{px} \times 0.75$)。在属性与属性值之间，使用冒号（:）进行区分，在语句末尾使用分号（;）表示该语句结束。

```
body {  
    font-size: 75%;      设置字号为75% (12px), 浏览器的默认字号为16px  
}
```

接着，我们控制文本行间距，控制文本行间距的属性为 line-height，设置其值为 1.5。在网页浏览器中默认的行间距约为 1.2，默认字体大小为 16px，所以实际行间距为 $16\text{px} \times 1.2$ ，即 19px 左右。这里我们设置行间距为 1.5，字体大小设为 12px，所以实际的行间距为 $12\text{px} \times 1.5$ ，即 18px。

```
body {  
    font-size: 75%;  
    line-height: 1.5;    设置行间距 (行高) 为字体尺寸的1.5倍 (12px*1.5=18px)  
}
```

综上所述，编写的 CSS 代码如右图所示。编写好之后，保存一下。

再次打开 index.html 文件，在 <head> 与 </head> 之间插入一行代码，使其可以调用 style.css 样式表文件。



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

使用<link />调用css/style.css

运行网页浏览器，打开 index.html 文件，查看网页页面效果，可以发现文本的大小与行间距都被改变了，样式表对网页起到了很好的控制作用。



控制标题 (h1, h2) 样式

下面我们开始为标题设置字体颜色，首先我们确定选择器，标题 h1 与 h2 就是要控制的选择器，它们是 HTML 标签。

```
<h1>body 区块(section) 标题</h1>
<p>HTML 5文档由不同的区块构成。</p>

<h2>HTML 5中新增的区块元素</h2>
<ul>
    <li> section </li>
    <li> article </li>
    <li> nav </li>
    <li> aside </li>
</ul>

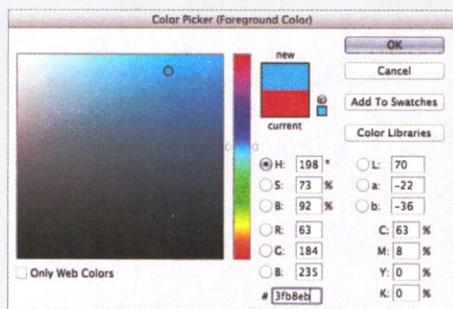
<h2> 中国网页浏览器的占有率 </h2>
```

```
h1, h2 {           选择要修饰的对象(h1、h2)
    color: #3fb8eb;  设置color属性为十六进制颜色值#RRGGBB，指定标题字体颜色
}
```

若要向多个 HTML 标签应用相同样式，就应在大括号前罗列出要应用的所有标签，并且使用逗号 (,) 将它们分隔开来，我们将这些选择器称为“选择器组”(Grouping)。大括号中定义的风格样式将会对前面列出的所有选择器都起作用。在大括号中，我们指定标题的颜色为 #3fb8eb，该颜色值为十六进制数，以 # 开始，6 位数字分为 3 组，每组两位，依次表示红、绿、蓝 3 种颜色的强度，即 #RRGGBB。十六进制数共包含 0~9 和 a~f 共 16 个数。颜色值越接近 0，颜色就越接近黑色；颜色值越接近 f，颜色就越接近白色。比如，#000000 表示黑色，#ffffff 表示白色，#808080 表示中性灰。

小知识 查找颜色值

若使用 Photoshop 或 Dreamweaver 等工具，查找颜色值就会非常容易。在不使用这些软件时，可以通过访问 HTML Color Codes 网站 (<http://html-color-codes.info/chinese/>)，获取所需的颜色值。



使用 Photoshop 颜色拾取器获取颜色值



使用 HTML Color Codes 网站获取颜色值

控制网页背景与字体

下面我们开始为网页添加背景图片。首先在包含 html 文件与 css 文件夹的目录下创建一个 images 文件夹，而后在其中添加一张背景图片 bg.jpg。

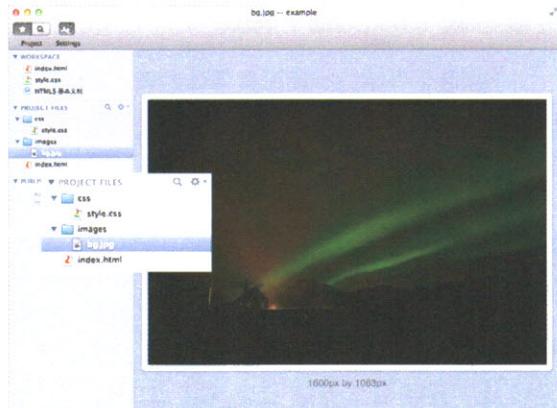
在 body 选择器的大括号中，添加背景图片属性 background-image，设置其属性值为 url('.. /images/bg.jpg')。url 指背景图片的路径。我们目前正在编辑 style.css 文件，该文件在 css 文件夹下，若想访问 images 文件夹中的图片，需要先跳出当前目录，返回到上一级目录中，因此我们使用“..”，它表示上一级目录。若想访问当前目录中的文件，则使用“.”，它表示当前目录。我们在跳出 css 文件夹，返回到上一级目录之后，再访问 images 文件夹中的 bg.jpg 图片，所以指定图片路径为“.. /images/bg.jpg”。在后面的学习中，我们会不断涉及文件路径的问题，所以请各位务必弄清楚设置路径的方法。向 css 的 body 选择器中添加插入网页图片的代码如下所示。

```
body {  
    font-size: 75%;  
    line-height: 1.5;  
    background-image: url('../images/bg.jpg');  
}
```

向 css 中添加插入网页图片的代码之后，保存它，按 F5 键，刷新网页，可以发现网页已经有了背景图片，标题 h1 与 h2 的文本颜色为浅蓝色。其他未指定颜色的文本默认颜色为黑色，并且因为背景是深色的，所以它们被淹没于背景的黑色之中，就像隐藏了一样。

接下来，设置正文字体颜色为浅灰色，在 body{} 中添加如下代码。

```
body {  
    font-size: 75%;  
    line-height: 1.5;  
    background: url('../images/bg.jpg');  
    color: #e6e6e6;  
}
```



为body设置背景图片



设置body所有的字体颜色为浅灰色

保存 CSS 样式表，单击 F5 键，刷新网页，在暗背景上能够清晰地看到亮灰色的文字。但是因为文本使用的是衬线字体（Serif），所以可读性不佳。与印刷物不同，网页多使用无衬线字体（Sans-Serif），以便提高网页文本的可读性。接着，我们将正文文本字体设置为无衬线字体（Sans-Serif）。

在 CSS 中 font-family 属性用于设置字体种类，我们将其设置为黑体，黑体属于无衬线字体（Sans-Serif）。

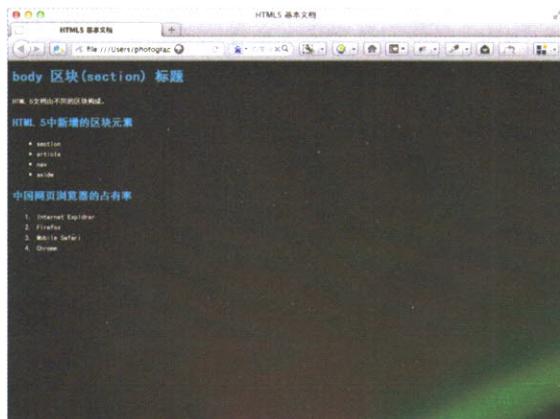
```
body {  
    font-family: 黑体, Dotum, Sans-Serif;  
    font-size: 75%;  
    line-height: 1.5;  
    background: url('../images/bg.jpg');  
    color: #ececfc;  
}
```



为 body 设置字体(font)

再次保存 CSS 样式表，返回浏览器中，按 F5 键，刷新网页，可以看到网页字体已经被更改为黑体了。仔细观察页面，可以发现由于背景图片比浏览器窗口的尺寸大，使得背景图片看上去就像被剪裁过一样。CSS3 提供了一个新的属性 background-size，用于解决这一问题。通过使用 background-size，可以根据浏览器的窗口尺寸，设置背景图片的大小，使得背景图片恰好能够填充满整个浏览器窗口。

如前所言，background-size 是 CSS3 新增的一个属性，它允许我们随心所欲地控制背景图片的尺寸。使用时，它的值可以是像素值，也可以是百分比或 auto，还可以是 cover 和 contain。若设置为 cover，则保持图片本身的宽高比例，将图片缩放到正好完全覆盖定义背景的区域；若设置为 contain，则保持图片本身的宽高比例，将图片缩放到宽度或高度正好适应定义背景的区域。



```
body {  
    font-family: 黑体, Dotum, Sans-Serif;  
    font-size: 75%;  
    line-height: 1.5;  
    background: url('../images/bg.jpg');
```

```
background-size: cover;  
color: #ececce;  
}
```

设置body元素的背景图片尺寸为cover

在body{}中添加完控制背景图片尺寸的属性后，保存CSS样式表。然后返回到浏览器中，按F5键，刷新网页，可以发现背景图片已经完全覆盖浏览器窗口，并且没有出现被裁剪的现象。关于background-size属性更多的用法，请参考本书第10章中的相关内容。



如上所言，background-size是CSS3新增的属性，非常实用。除此之外，CSS3还增加了其他许多属性，如果感兴趣，可以翻看专门的书籍进行学习。在使用这些新增的属性时，要考虑老版本的浏览器是否支持它们。每种浏览器支持的属性也有所不同，我们在编写CSS样式表时要充分考虑到这个问题，并采取相应的应对措施。关于各种浏览器对CSS3新增属性的支持情况，请访问standardista CSS3网站(<http://wp.me/P0Zpz-1J>)进行确认。



在standardista CSS3网站中查看各浏览器对CSS3新增属性的支持情况

至此，我们已经使用CSS样式表设置了字体、字号，以及字体颜色，还向网页中插入了背景图片，对CSS样式表的使用方法有了大致的了解。CSS样式表最基本的功能就是对网页进行美化修饰。在后面的学习中，我们会通过更多的实例，进一步学习有关CSS样式表的知识。

小知识 了解 Snippet Code

Snippet Code常用来编写通用代码或记录不常用的代码，在需要的时候，单击Tab Trigger键，即可快速地调出相应代码，方便代码编写。

脚本语言 JavaScript 基础

JavaScript 是一种基于客户端浏览器的，基于对象、事件驱动的脚本语言，它是动态解释执行的。它比 HTML 与 CSS 要难学一些，如果之前你从未接触过任何编程语言，学起来会有些吃力。JavaScript 包含很多内容，市面上有很多图书专门讲解它，如果你想掌握它，建议你购买专业图书进行学习。在本部分，我们只简单地介绍 JavaScript 的一些基本的知识及常用的使用方法。

像 CSS 一样，JavaScript 也访问 HTML 文档中的元素，通过操作文档中的对象，动态修改 HTML 页面内容，创建、删除 HTML 页面元素，修改 HTML 页面元素的内容、外观、位置和大小等。在前面的学习中，我们知道 CSS 使用选择器（Selector）来访问文档中的元素对象，而 JavaScript 则使用文档对象模型（DOM）来访问文档对象。在正式开始学习 JavaScript 之前，首先在 index.html 文件所在的目录下，创建一个 js 文件夹，而后在该文件夹下创建 script.js 文件。

如同在 index.html 中调用 style.css 文件一样，我们也需要在网页中调用 script.js 脚本文件，它才能发挥作用。在编辑器中打开 index.html 文件，而后在 </body> 标签之前添加 <script></script> 标签，调用 script.js 脚本文件。

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

使用<script>调用js/script.js

在网页中添加完脚本调用代码后，进入 js 文件夹，在编辑器中打开 script.js 文件，输入如下一行代码。

```
// 选择h1元素  
document.getElementsByTagName('h1');
```

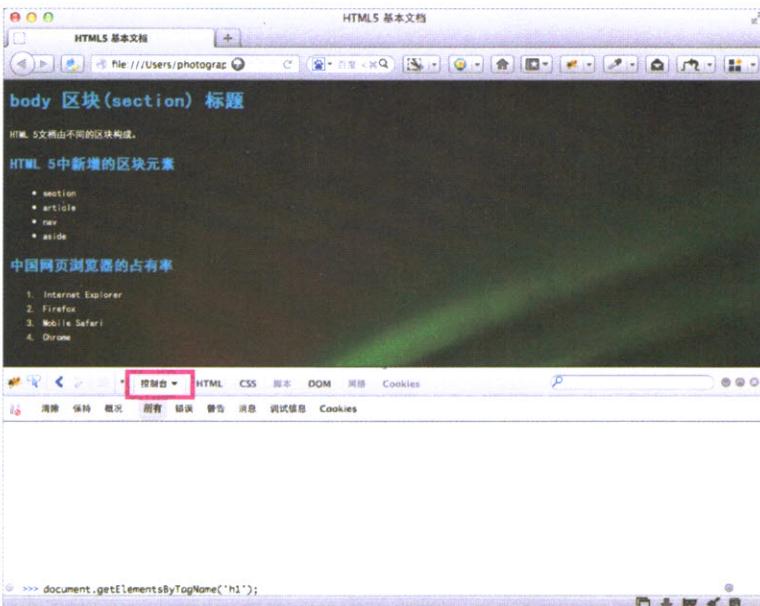
在文档中查找并返回h1

在上述代码中,document 代表 html 文档。若想获得 HTML 文档中的某个元素,首先要使用 document 对象,而后使用点操作符(.) 调用该对象的 getElementsByTagName() 方法,通过标签名称获取文档中指定的元素。

getElementsByTagName() 是一个方法,它通过查找整个网页文档的 HTML 元素,返回指定名称的元素集合。在 JavaScript 中把执行某个动作的语句称为“方法”(Method)。方法是一种函数,函数就是由事件驱动的或者当它被调用时执行的可重复使用的代码块。当某个函数被某对象所含有时,我们把该函数称为该对象的方法。如果你之前从未接触过编程知识,你可能不太理解这句话。现在不理解没关系,继续跟着学习,慢慢你就会理解它。

在方法名称后紧跟着一个小括号,该小括号内可以不放任何东西,也可以放入某个类型的参数,将外部参数传入方法,并由方法进行使用。getElementsByTagName() 方法要求传入一个字符串(String)参数,以便使用该参数查找指定的 HTML 元素。在 JavaScript 中,字符串需要使用单引号或双引号引起,这是 JavaScript 的规则之一,希望大家养成遵循规则的好习惯。这里,我们使用单引号将 h1 引起来,将其作为参数传入方法之中。

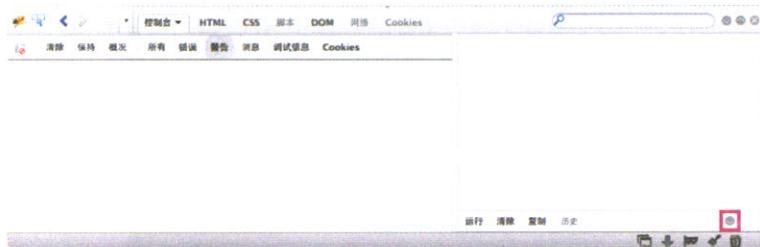
接下来,我们要在 Firebug 中查看 JavaScript 代码。首先打开 Firefox 浏览器,按 F12 键,运行 Firebug 插件,在浏览器的下半部分打开 Firebug 窗口,单击“控制台”,打开控制台窗口。



小知识

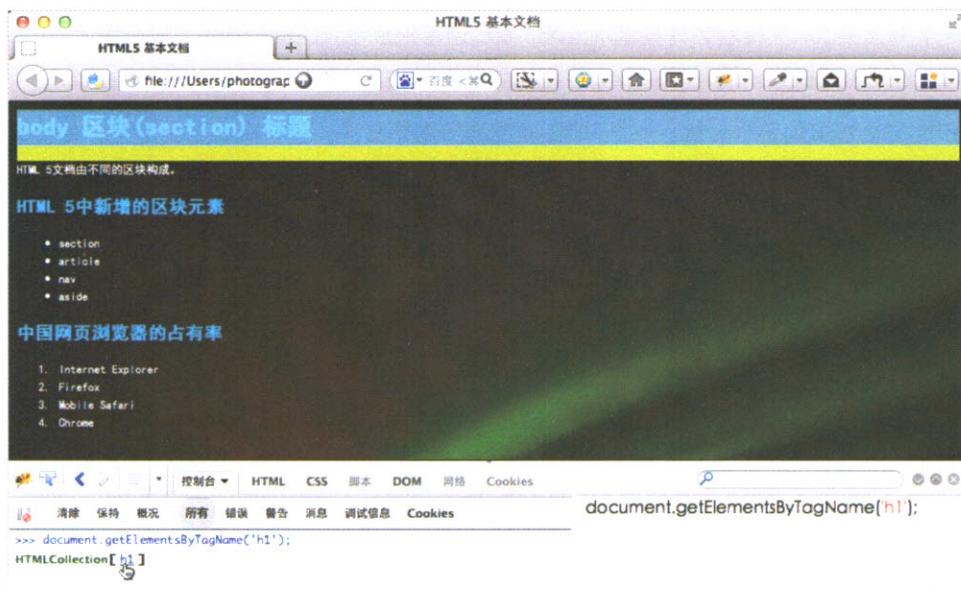
在 Firebug 窗口中,单击“控制台”,若显示“控制台面板已被禁用”信息,可以单击“启用”,启用控制台面板。

在需要输入多行代码时，仅允许进行单行输入的命令行，使用起来非常不方便。在控制台窗口的右下角区域中，单击三角形标志按钮（），打开允许多行输入的区域窗口。



在右侧的命令输入窗格中，输入 `document.getElementsByTagName('h1')` 命令，而后单击窗格底部的“运行”按钮，或按 `Ctrl+Enter` 键，在左侧窗口中显示出命令执行的结果。关于 JavaScript 的所有内容，我们都会在 Firebug 中进行运行调试，所以请记住上面使用的方法。

如下图所示，运行命令后，得到的执行结果为 `[h1]`，其中中括号 `[]` 表示 `document.getElementsByTagName('h1')` 方法返回的是指定名称的元素集合，即返回 `h1` 元素的集合。从方法本身名称的含义来看，该方法通过指定的标签名称获取一组元素，其返回的是 `h1` 元素集合。结合 HTML 文档，该方法的功能是查找整个 HTML 文档，获取其中所有的 `h1` 元素。因此在 Firebug 左侧的窗口中，得到的结果带有中括号 `[]`。



从获取的结果来看，该集合中仅仅包含一个 `h1` 元素。当然是这样的，因为我们编写的 `index.html` 文档中仅含有一个 `h1` 元素。接下来，我们修改一下标签名称，查找标签为 `li` 的元素。

```
// 选择元素  
document.getElementsByTagName('li');
```

在文档中查找并返回元素集合



在命令窗格中输入获取 li 元素的命令后，单击“运行”按钮。在左侧的结果显示窗口中，可以看到从 HTML 文档中查找到的多个 li 元素。从运行结果看，该方法查找了整个 HTML 文档，获得所有的 li 元素，并以集合的形式返回这些元素。查找到的 li 元素既有 ul 中的，也有 ol 中的。移动鼠标指针到其中一个 li 元素上，文档就会将其显示出来。我们通常把查找到的集合称为“节点集合”（集合类似于数组，但它不是数组，而是包含多个节点的节点列表 Node List）。

节点用于区分文档中的各个对象，一个 HTML 文档拥有多个节点，但只有一个根节点，其他节点都是根节点的子节点或孙子节点，我们可以把这些节点大致划分为元素节点（element node）、属性节点（attribute node）和文本节点（text node）三部分。

在我们编写的 HTML 文档中，`<h1>`、`` 和 `<p>` 等都属于元素节点，即 HTML 文档中的各个标签都是元素节点。属性节点是指元素节点包含的属性，比如在 `<html lang="zh">` 中，`html` 是元素节点，属性 `lang` 是属性节点。

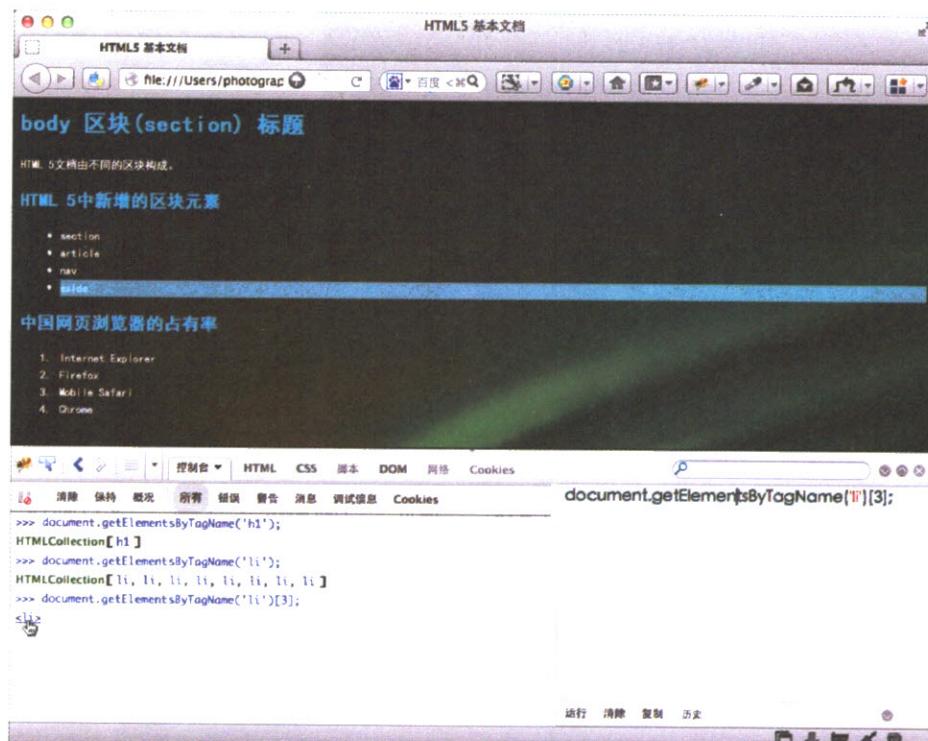
最后，文本节点是指元素节点所包含的文字，比如在 “`<h2> 中国网页浏览器的占有率 </h2>`” 中，`h2` 是元素节点，无属性节点，元素节点 `h2` 包含的 “中国网页浏览器的占有率” 文本是文本节点。

让我们返回去再看一看集合。由于 `getElementsByName()` 方法返回的是多个 HTML 元素，因此我们也把它返回的集合称为元素节点集合。既然集合中包含着元素节点，那么元素节点包含的文本节点也能被获取吗？是的，可以。但是在当前的状态下，无法获取，在访问某些个别元素时，需要获取其所含的文字，而当前的集合中包含的是元素。

若想从集合中获取某个元素，应该怎么办呢？方法很简单，只要在集合后使用中括号，并在中括号中指定元素的序号，即可获取指定位置的元素，假如我们要获取集合中的第 4 个元素，如下所示。

```
// 选择li元素集合中的第4个元素  
document.getElementsByTagName('li')[3];
```

查找li集合，并返回集合中的第4个元素



从上述命令看，在获取指定的元素时，首先在`getElementsByTagName('li')`方法后添加一个中括号，而后在中括号内指定一个数字，即可获取该数字所对应的元素。如果你理解了集合，就很容易理解这种从集合获取元素的方法。在命令语句中，[]中的数字3表示什么意思呢？为什么放入的数字是3，而得到的是第4个元素呢？是不是搞错了呢？

不是的。在编程中数组或列表中的元素编号是从0开始的，所以数字编号3对应的是第4个元素。但在CSS编写中不遵循这一原则，它是从数字1开始编号的。记住这种不同，防止发生混淆。

我们已经学会如何从集合中获取指定的元素，那么我们如何获取元素所包含的文本节点呢？比如我们要获取`<h1>`元素包含的文本节点，应该怎么办呢？再次返回到Firebug的控制台窗口中，修改要获取的元素，如下所示。

```
// 选择h1元素集合中的第1个元素  
document.getElementsByTagName('h1')[0];
```

查找h1集合，并返回集合中的第1个元素



在我们编写的 HTML 文档中只含有一个 h1 元素，即在 getElementsByTagName() 方法返回的集合中仅有一个 h1 元素，其数字编号为 0，所以我们使用 getElementsByTagName('h1')[0] 来获取它。在获得了 h1 元素后，如何获得它包含的文本节点呢？

获得文本节点的第 1 个方法是使用 firstChild，使用它获取 <h1> 元素的第一个子节点。由于 <h1> 元素只包含文本，因此 firstChild 指的就是文本节点。

```
// 访问第1个h1元素的第一个子节点
document.getElementsByTagName('h1')[0].firstChild;
```



在控制台左侧窗口中，查看命令语句的执行结果：“<TextNode textContent="body 区块(section) 标题">”。从执行结果可以看出，获取的的确是文本节点。接下来，再从文本节点的 textContent 属性获取属性值，即可得到文本。在文档对象模型中，使用 nodeValue 来获取节点值。

```
// 访问第1个h1元素的第1个子节点的值  
document.getElementsByTagName('h1')[0].firstChild.nodeValue;
```



再次执行命令语句，在控制台左侧的窗口中，可以看到获取的节点值。简单地整理一下，获取节点值的过程：首先通过标签名称，获取同名元素的集合，再在集合中获得指定的元素，而后访问元素的节点，最后获得节点值。

上面我们刚刚学习了如何获取某个元素节点值的方法，下面我们将尝试操作元素的节点值，这里我们将修改元素的节点值。与访问元素节点值一样，修改元素节点值的方法也非常简单，比如我们修改 `<h1>` 元素的文本内容。

```
// 访问第1个h1元素的第1个子节点的值  
document.getElementsByTagName('h1')[0].firstChild.nodeValue = '进行文档操作 ^~^'
```



如上所示，在修改元素的节点值时，我们在节点值右侧使用 “=”，并在 “=” 右侧给出修改值。在右侧命令窗口中，输入命令语句，单击“运行”按钮后，即可修改指定元素的节点值。在这里，我们要注意理解 “=” 这个符号。在数学中，= 是等号，它表示左右两部分是相等的，但是在程序

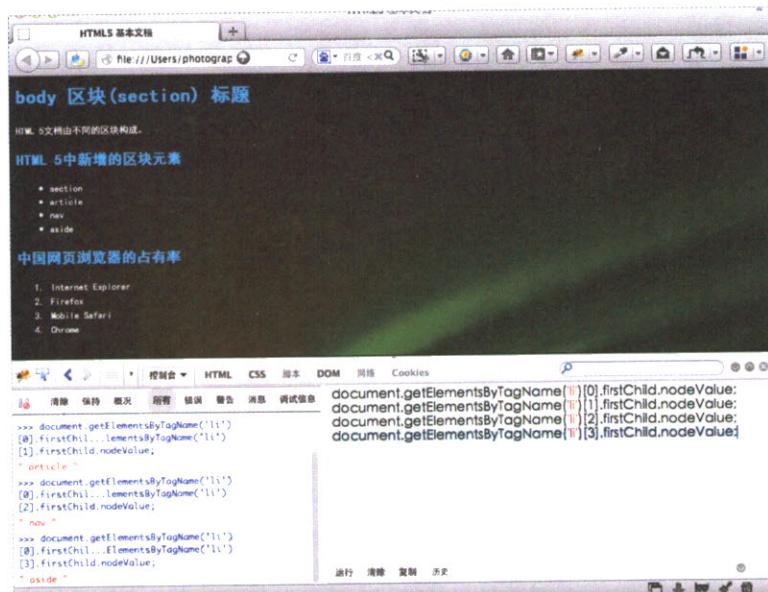
语言中，`=`是一个赋值运算符，它将右侧表达式的值赋给左侧变量。在上面的命令语句中，`=`用于将右侧的字符串赋给左侧文本节点的`nodeValue`。

接下来，我们继续使用上面的方法，尝试获取HTML文档中`li`的文本节点值。在我们编写的HTML文档中仅有一个`h1`元素，而`li`元素有多个，在这种情况下，我们该如何查找指定`li`元素的文本节点值呢？我们可以像下面这样，一个个地进行查找。

```
// 访问第1个li元素的第一个子节点的值  
document.getElementsByTagName('li')[0].firstChild.nodeValue;
```



如下图所示，在命令窗口中，输入命令语句，在左侧窗口中显示执行结果“`section`”，它是第一个`li`元素的文本节点值。我们可以使用相同的命令语句，获取其他`li`元素的文本节点值，只要修改`[]`中的元素编号即可，如下所示。



但是每次都敲入这些类似的代码让人十分劳累、厌烦，甚至会让人精神恍惚、发呆。那么有没有方法可以减少相似代码的输入呢？有的，在编写需要反复执行的代码片段时，我们常常会采用一些循环语句。下面我们将采用 JavaScript 语言中的循环语句来实现对元素集合的遍历功能。

在 JavaScript 中支持多种循环语句，其中最常用的是 for 语句与 while 语句。在这里，我们将使用 while 语句对相同语句进行循环。

换言之，通过使用 while 语句，对相同的语句进行循环，减少不必要的重复代码。在使用 while 语句之前，我们要再谈一谈节点集合。

通过前面的学习，我们已经知道节点集合是多个相同元素的集合。如果文档中存在多个要查找的元素，那么查找后的结果就是这些元素组成的集合。在代码编写过程中，我们往往需要知道节点集合中包含的元素个数。那么我们如何知道节点集合中包含的元素个数呢？在前面我们提到过，节点集合类似于数组，我们知道 JavaScript 数组提供了 length 属性，通过该属性，即可获知数组内部包含的元素个数。同样，节点集合也提供了该属性，以便我们获取节点集合中包含的元素个数。若想获取 li 节点集合中的元素个数，就需要使用节点集合的 length 属性，如下所示。

```
// 获取li元素集合包含的元素个数  
document.getElementsByTagName('li').length;
```



在命令语句窗口中，输入节点集合元素个数查询语句，单击“运行”按钮，在左侧窗口中即显示出元素个数 8，表明在 HTML 文档中共有 8 个 li 元素。在 HTML 文档中的确有 8 个 li 元素。那么，我们在使用循环语句访问这些元素时，需要循环 8 次，才能访问完全部 8 个元素。

下面让我们一起看一下 while 循环的语法格式，如下所示，它是 while 循环最基本的语法格式。首先是 while 关键字，表示后面是一个 while 循环。然后紧跟着一个小括号 ()，该小括号内包含逻辑表达式，这个逻辑表达式是循环条件，只有真 (true) 与假 (false) 两个取值。当逻辑表达式的值为真 (true) 时，花括号 ({}) 内的循环体就被执行；当逻辑表达式的值为假 (false) 时，结束循环。花括号 ({}) 包含循环体，当循环体只有 1 行语句时，包裹循环体的花括号可以省略。

```
// JavaScript中的while语句  
while (循环条件) {  
    // 循环代码  
}
```

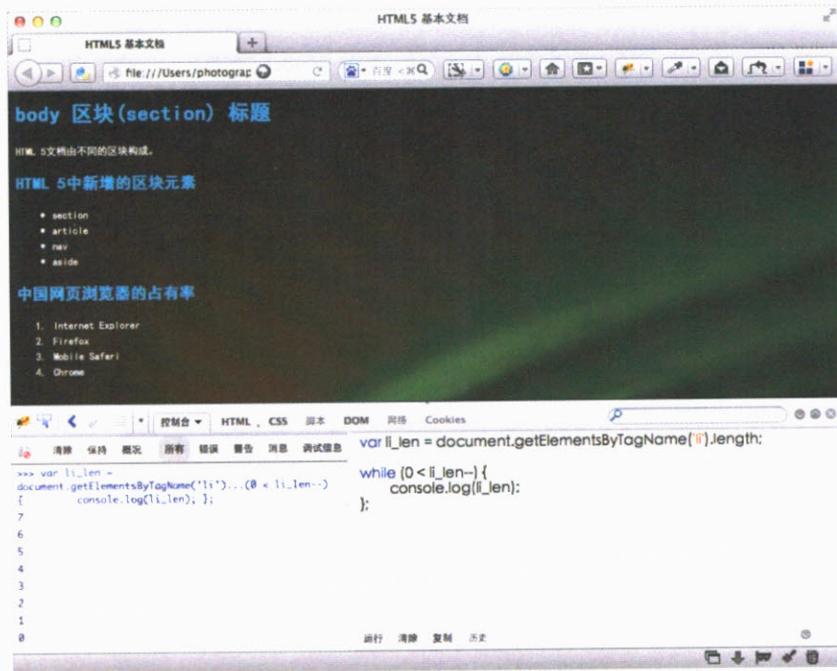
在了解了 while 语句的语法格式后，下面我们开始编写具体的代码。首先我们要获取节点集合的元素个数，而后将其保存到一个变量（Variable）中。在 JavaScript 中定义一个变量非常简单，既可以采用隐式定义，也可以采用显式定义。这里，我们采用显式定义的方式，这种方式采用 var 关键字声明变量。如下所示，我们采用 var 关键字声明了一个变量 li_len，并使用它来保存 li 元素集合中的元素个数。

```
// 声明变量li_len，保存li集合的长度  
var li_len = document.getElementsByTagName('li').length;
```



什么是变量呢？在数学中，变量是指取值不断变化的量，而在编程中变量是指保存数据（Data）的空间。为什么不直接使用数字，而是先把数字保存到变量中呢？似乎多此一举。不是这样的，当我们把数字保存到变量之中后，再次使用就非常方便了。为了便于理解，举个小例子，我们要数一个年级的学生总数，一一数完之后，是 53 名。过了一段时间后，你可能会忘记这个数字，不得不重新点数学生的人数，非常费力，又不方便。事实上，我们完全可以把学生的人数记录在便笺上，贴在考勤表上。这样，就不用再次点数学生人数了，直接看便笺上的人数就行了。变量就相当于便笺纸，用来记录某个值，方便我们下次使用。

`li_len` 是变量名称，它里面保存着数字 8。关于变量名称，你可以根据需要指定其他任意名称，但是建议在为变量起名时起一个具有特定含义的名称，以方便记忆使用。现在，在 `li_len` 变量中存储着数字 8，需要时，我们可以直接使用这个变量。接下来，我们要使用 `while` 循环编写循环语句，将 `li_len` 中的值打印出来。



```
var li_len = document.getElementsByTagName('li').length;
while (0 < li_len--) {
    console.log(li_len);
}
```

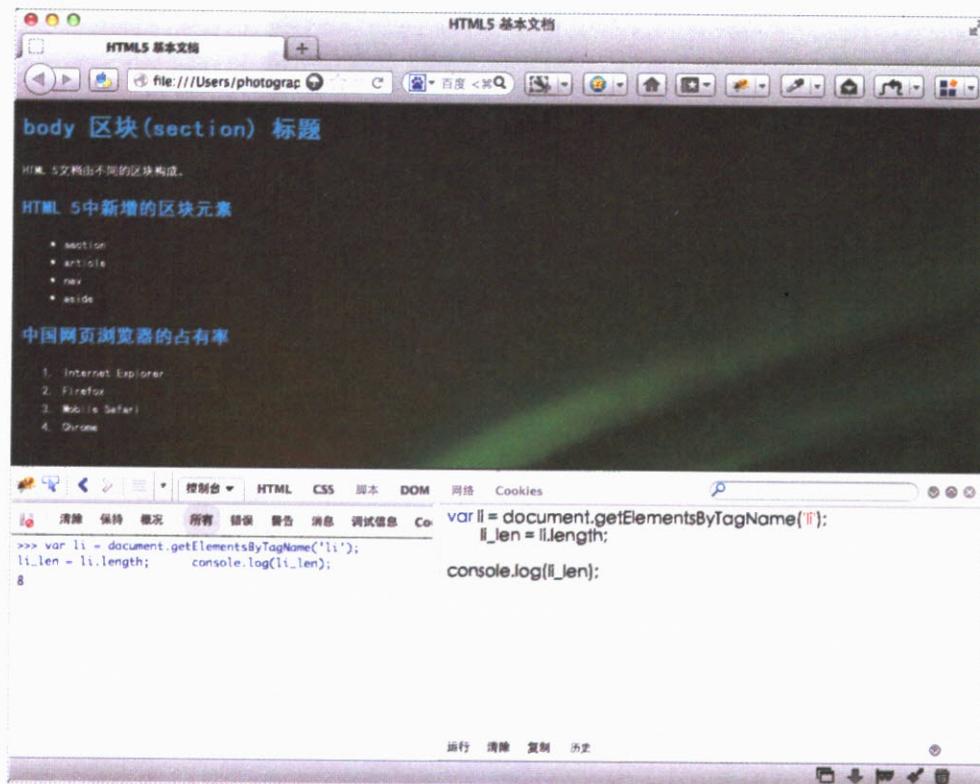
如代码所示，在 `while` 语句小括号中的逻辑表达式为 “`0 < li_len--`” ，它表示什么意思呢？下面让我们一起分析一下。

该逻辑表达式表达的含义为：每次执行一次循环，`li_len` 值减 1，循环不断执行，直到 `li_len` 小于 0。`li_len` 初始值为 8，逻辑表达式为真，随即 `li_len` 值减 1，变为 7，再执行循环语句 `console.log(li_len)`，将 `li_len` 值打印在控制台中。`log()` 方法是 `console` 对象的一个方法，用于向控制台输出指定内容。在控制台左侧窗口中显示的是循环语句输出的 `li_len` 变量值。当 `li_len` 变量值小于或等于 0 时，循环就被终止。在循环执行过程中，变量 `li_len` 的值不断减 1，直到其值为 0 时，循环条件不再成立，花括号 {} 内的循环体将不会再执行。

在使用 `while` 循环语句时，需要注意的是，循环条件不能恒为真 (`true`)，即一定要让循环条件有为假 (`false`) 的时候，否则循环将陷入死循环中，永远无法结束，这可能会导致浏览器崩溃。为了避免出现这个问题，在写循环条件时一定要注意。

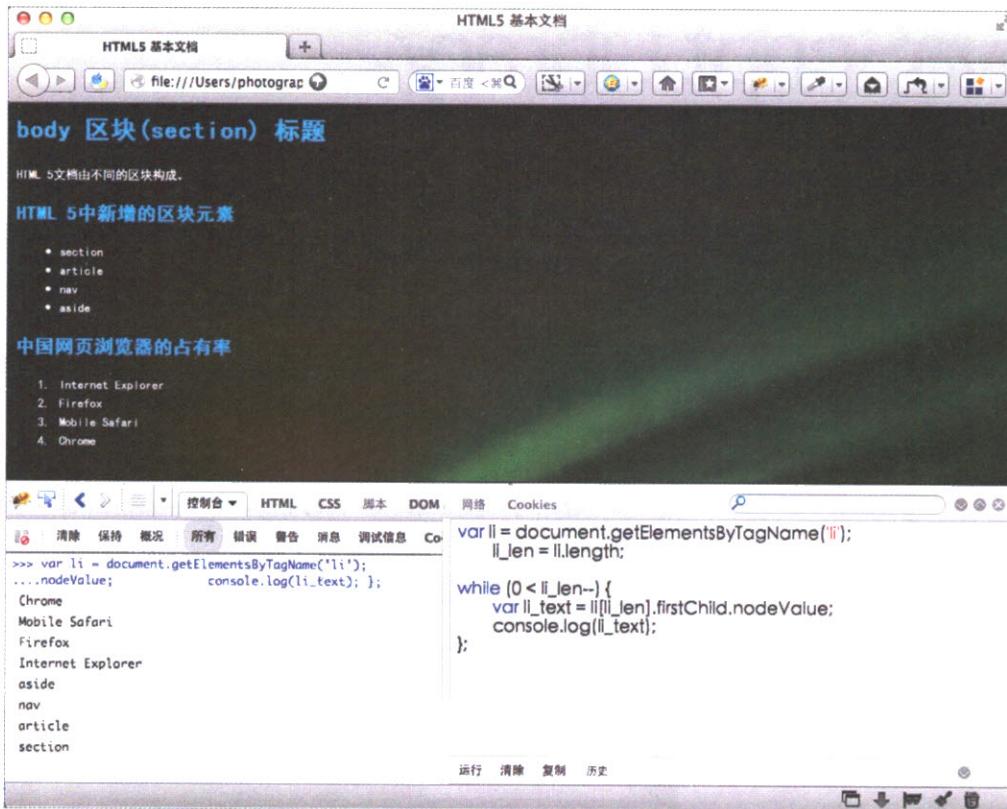
在前面的学习中，我们已经知道调用 `getElementsByName()` 方法将返回一个元素集合。下面我们将定义一个新变量 `li`，并使用它来接收 `getElementsByName()` 方法返回的元素集合，获取元素集合包含的元素个数后，将元素个数打印在控制台中。

```
var li = document.getElementsByTagName('li');
li_len = li.length;
```



如上图代码所示，首先定义一个 `li` 变量，并将 `getElementsByName()` 方法返回的元素集合保存到其中。而后再定义一个 `li_len` 变量，保存元素集合中所包含的元素个数，即 `li_len=li.length`，注意定义的两个变量之间要使用逗号（`,`）进行分隔。从前面的分析可知，现在 `li_len` 变量中保存着数字 8。

像这样，我们将某个值保存到某个变量中，以后再使用时会更加方便，因为变量的名称简短，并且非常好记。在使用某个值时，我们只需要使用相应语句查询一次，而后将查询到的值保存到变量中，再次使用时，直接引用该变量即可，不需要再次进行查询，节省了时间，也提高了程序的性能。在获取并保存了元素集合的元素个数后，使用 `console` 对象的 `log()` 方法，将 `li_len` 变量保存的值输出到控制台窗口中，输入的元素个数为 8，如上图所示。接下来，我们将使用 `while` 循环语句将所有文本节点的值输出到控制台窗口中。



```

var li = document.getElementsByTagName('li'),
    li_len = li.length;
while (0 < li_len--) {
    var li_text = li[li_len].firstChild.nodeValue;
    console.log(li_text);
}

```

首先新定义一个变量 `li_text`，用于保存指定 `li` 元素的文本节点值。关于获取文本节点值的方法，在前面我们已经介绍过，在此不再赘述。在获得文本节点值并保存到 `li_text` 变量中后，调用 `console` 对象的 `log()` 方法，将 `li_text` 变量中保存的文本节点值输出到控制台中。

如上图所示，在控制台左侧的窗口中，可以看到输出的文本节点值。如果执行命令语句时发生错误，请查看命令语句是否存在错误，这些错误一般是由输入错误引起的，请认真检查输入的语句。

到此为止，我们学习了 JavaScript 变量、基本语法、`while` 循环语句、数组、节点、访问节点集合中的某个对象，以及使用 `getElementsByTagName()` 方法（文档对象模型的 API 之一）获取元素节点集合的方法。这些内容相对简单，但都是非常基础的内容，希望各位认真学习，将这些内容弄清楚，理解好，这些知识在后续的学习中非常重要。

* 如前所言，JavaScript 是一种较为复杂的脚本语言，包含条件语句、函数、对象、事件和事件处理注册等多种内容，建议大家购买专门介绍 JavaScript 的书籍进行系统学习。

JavaScript Library, jQuery 基础

jQuery 是继 prototype 之后又一套 JavaScript 脚本库，其宗旨是“写更少的代码，做更多的事情”。它是轻量级的 js 库，兼容 CSS3，还兼容各种浏览器。jQuery 的语法直观，使用简单，使用它将极大提高编写 JavaScript 代码的效率，让编写的代码更加优雅、更加健壮。由于它解决了 JavaScript 在不同浏览器中的兼容性问题，因此越来越多的开发人员开始使用它，相信在将来，它仍然会是最受欢迎的 JavaScript 脚本库。

若要使用 jQuery，首先要访问 <http://jquery.com> 网站下载最新版本的 jQuery。进入官方网站后，单击右侧的“Download (jQuery)”按钮，进行下载。

下载完 jQuery 文件后，需要将其放入指定的文件夹中，以便在编写 JavaScript 代码时使用它。在本部分示例中，我们将 jquery.min.js 文件放入 js 文件夹中。（在本书中，我们在使用 jQuery 时都删除了版本部分。若出现未删除的情况，在编写调用相应文件的代码时，都要写上包含版本的名称。）

The screenshot shows the official jQuery website at jquery.com. At the top, there's a navigation bar with links for "jQuery", "Plugins", "UI", "Meetups", "Forum", "Blog", "About", and "Donate". Below the header, there's a large "GRAB THE LATEST VERSION!" section with a "Download" button. To the left, there's a "WHO'S USING JQUERY?" section with a search bar and a list of websites. On the right, there's a "LEARN JQUERY NOW!" section with a "RUN CODE" button, and a "JQUERY RESOURCES" section with links to "Getting Started With jQuery", "How jQuery Works", "Tutorials", "Using jQuery with other libraries", and "jQuery Documentation". There's also a "Developer Resources" section with links to "Mailing List", "Source code / Git", "Plugin Authoring", and "Submit a New Bug Report".

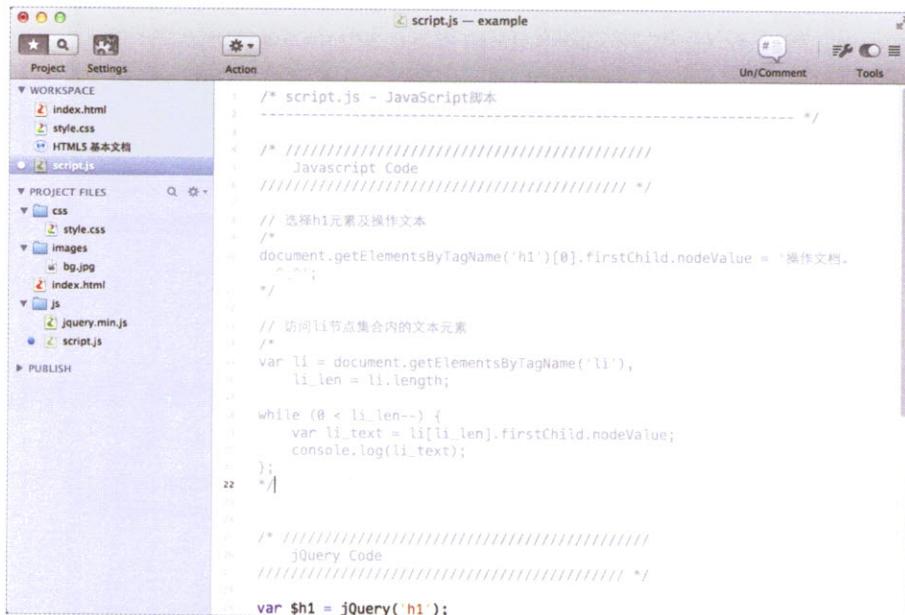
The screenshot shows the Eclipse IDE interface with a Java workspace titled "jquery.min.js - work". The left sidebar shows the project structure with files like index.html, style.css, script.js, and jquery.min.js. The main editor area displays the contents of jquery.min.js, which is a highly compressed and minified version of the jQuery library. The code includes various utility functions, event handlers, and DOM manipulation methods.

```
(function(a,b){function cv(a){return f.isWindow(a)?a:a.nodeType==9&a.defaultView||a.parentWindow:!1}function cs(a){if(!cg[a])var b=c.body,d=f(<a>).appendTo(b),e=d.css("display");d.remove();if(e=="none")e=""})(ch||ch=c.createElement("iframe"),ch.frameBorder=ch.width=ch.height=0),b.appendChild(ch);if(!ci||ch.createElement)ci=(ch.contentWindow||ch.contentDocument).document,ci.write((c.compatMode=="CSS1Compat"?<!doctype html>:"<html><body>"),ci.close();d.ci.createElement(a),ci.body.appendChild(d),e=f.css(d,"display"),b.removeChild(ch))cg[a]=e}return cg[a];function cr(a,b){var c={};f.each(cm.concat.apply([],cm.slice(0,b)),function(){f.extend(a,b)});return a}f.extend(a,b)})(jQuery);
```

如何在 HTML 文档中调用 jquery.min.js 脚本库呢？很简单，只要打开 index.html 网页文档，在 </head> 标签前添加一行引用代码即可，如下所示。

```
<script src="js/jquery.min.js"></script>  
</head>
```

在前面，我们已经编写了 JavaScript 代码，下面我们将使用 jQuery 代码来实现相同的功能。首先打开 script.js 脚本文件，由于在 HTML 文档的 </body> 标签之前已经调用了 script.js 脚本文件，因此不需要再添加调用 script.js 脚本的语句了。在打开的 script.js 脚本文件中，先使用注释（单行注释 //，多行注释 /* */）将之前编写的脚本注释掉，如下图所示。



如上图所示，当使用注释将前面编写的 JavaScript 脚本注释掉后，浏览器就会自动忽略这些脚本，因为浏览器中的 JavaScript 不会解析这些注释。当然，即使去掉注释符号（// 或 /* */），浏览器也依然会解析这些脚本。

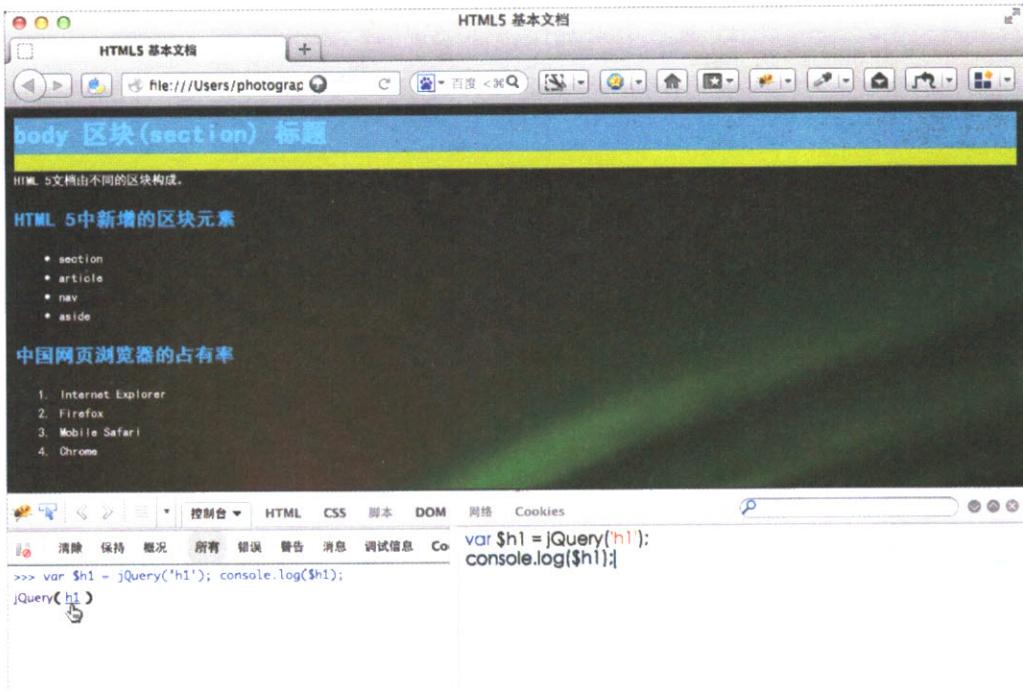
下面让我们开始编写 jQuery 代码。首先获取 h1 元素，访问其文本节点，并修改文本节点内容。我们要为 jQuery 对象指定要获取的对象，可以使用 getElementsByTagName() 方法将查找到的文档对象传到 jQuery 对象内，也可以直接使用 CSS 选择器，大多数时候会使用 CSS 选择器。

这里，我们要查找的对象是 h1，因此要把 h1 对象以文字串的形式传给 jQuery 对象，而后定义一个变量，用于保存查询的结果。在定义变量时，在变量名称前使用 \$ 符号，将其内含有的 jQuery 对象更直观地表示出来。

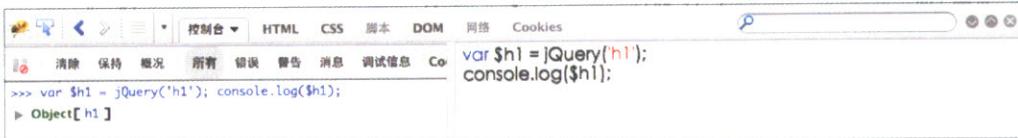
```
var $h1 = jQuery('h1');
```

变量\$h1保存含有文字串h1的jQuery对象

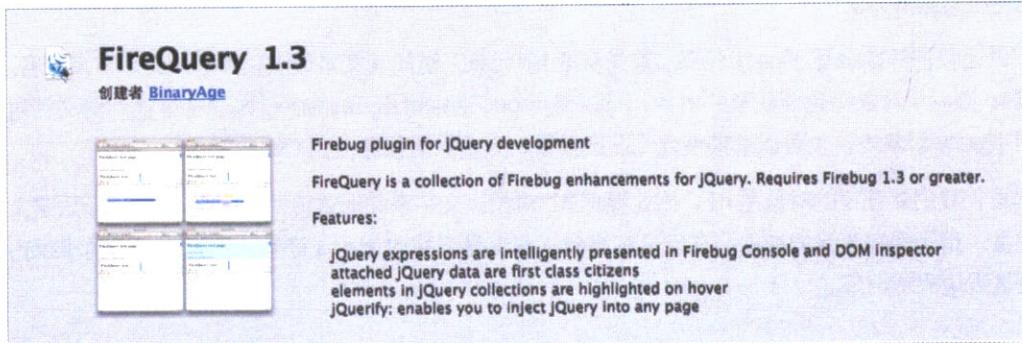
查看 Firebug 的控制台窗口，可以看到在 jQuery 对象的 () 中含有 h1 文档对象，并且 h1 显示为蓝色。移动鼠标指针到 h1 上，在 HTML 页面中可以看到它指向网页中的 <h1> 元素。



在 Firebug 的控制台窗口中，命令的执行结果可能不是 `jQuery(h1)` 形式，有可能是下图的形式，请不要慌张，这是正常的。



若想在控制台窗口中将查询结果显示为 `jQuery()` 的形式，就需要在 Firefox 中安装 FireQuery 插件。与安装 Firebug 类似，首先在 Firefox 的附件组件中进行搜索。搜索到之后，单击右侧的“安装”按钮，Firefox 会自动下载并安装该插件。



安装完 FireQuery 插件之后，重启 Firefox 浏览器，再次查看执行结果，可以发现查询结果以 `jQuery()` 形式出现了。



接下来，我们将更改 h1 元素的文本节点内容。在前面编写的 JavaScript 脚本中，我们通过为 firstChild.nodeValue 重新赋值的方式，更改了文本节点内容，这种方式不够直观，且较难理解。而使用 jQuery 代码则较为直观得多，理解起来也更容易。jQuery 对象拥有一个 text() 方法，它接受一个字符串参数，并且使用传入的参数替换原来的文本。首先使用点操作符，调用含有 h1 元素的 jQuery 对象的 text() 方法，而后传入字符串参数，该字符串就是修改后的文本，如下所示。

```
var $h1 = jQuery('h1');
$h1.text(使用jQuery更改内容);
```

更改 jQuery(h1) 对象的文本

在 Firebug 右侧的命令窗口中，输入命令语句，单击“运行”按钮。在浏览器中查看 HTML 页面，可以发现 h1 的内容发生了改变。比较我们前面编写的代码，你会发现同样的功能使用 jQuery 代码编写会更简洁、更直观、更容易理解，这正是 jQuery 的迷人之处。

接下来，我们将 HTML 文档中的 li 元素放入 jQuery 对象之中，即 jQuery(li)，这会查找 HTML 文档中的所有 li 元素，并将查找到的 li 元素包含到 jQuery 对象中。同时，我们又新定义了一个 \$lis 变量，用来保存 jQuery 对象。



```
var $lis = jQuery(li);
```

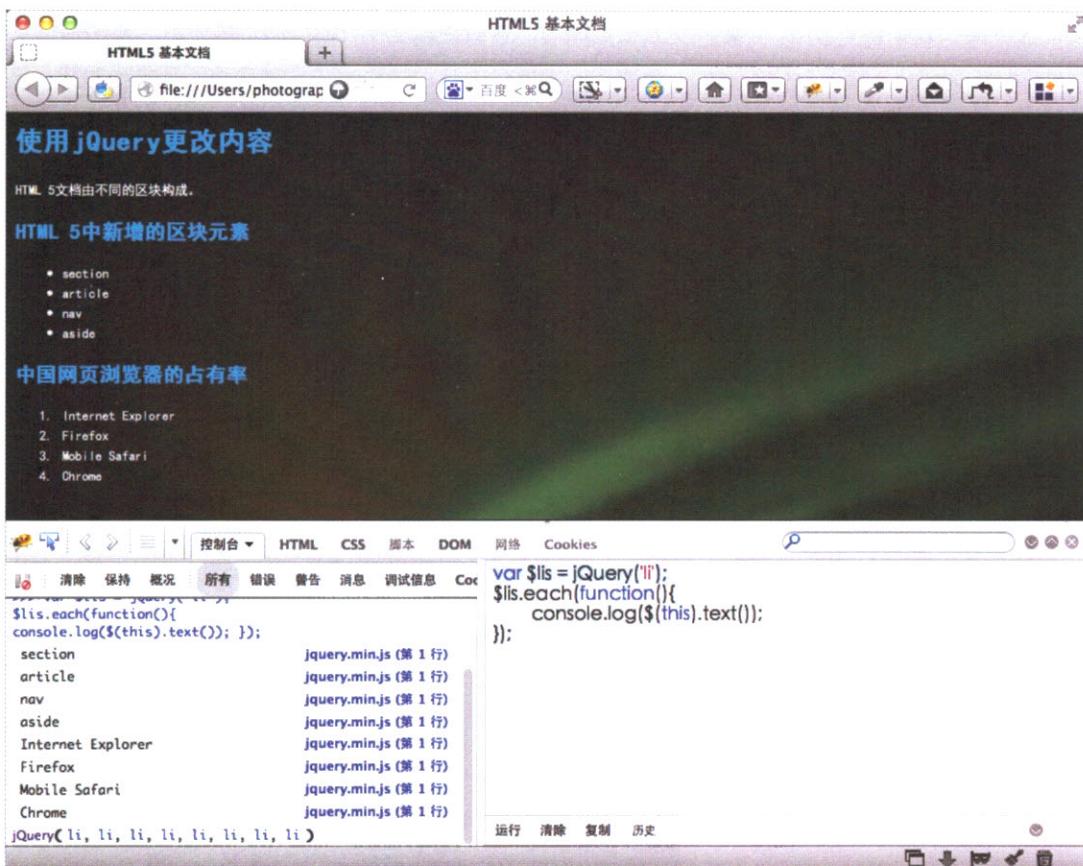
变量 \$lis 中保存着 jQuery 对象

然后，我们使用 jQuery 的循环语句遍历 li 元素的文本内容。在前面编写的 JavaScript 中，我们使用 while 循环语句来遍历 li 元素。在使用 jQuery 编写代码时，可以使用其提供的 each 语句。使用 jQuery 编写代码如下所示。

```
$lis.each(function() {
  console.log($(this).text());
});
```

将 jQuery(li) 对象保存到 \$lis 变量中，使用 each 语句遍历 \$lis 中的各个对象
在控制台窗口中显示 \$lis 变量中包含的各对象的文本值
结束循环

在 Firebug 命令窗口中，输入命令语句，单击“运行”按钮，在左侧窗口中可以看到 li 文本的内容被正常输出出来。与使用 JavaScript 相比，使用 jQuery 编写脚本，更简单、更简洁，也更有效率。在后面的学习中，我们将继续使用 jQuery 编写脚本，希望大家好好学习它，并尽快使用起来。



在本章中，我们简单地学习了有关 HTML、CSS 和 JavaScript 的知识，它们是制作网页必须的知识，同时也简单地介绍了 jQuery 这一强大的脚本库。但是仅了解学习本书中所介绍的相关内容，是远远不够的，建议大家购买专业书籍进行系统学习。

3 章

设计网页动态横幅广告 (Banner)

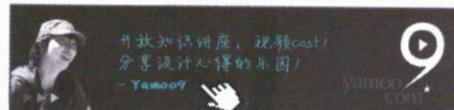
横幅广告是网络广告的常见形式，一般位于网页的醒目位置上，当用户单击这些横幅广告时，通常可以链接到相应的广告页面。设计横幅广告时，要力求简单明了，能够体现出主要的中心主旨，鲜明、形象地表达出最主要的广告意图。横幅广告可以是静态图形，也可以是动态图像。一般而言，与静态横幅广告相比，动态横幅广告更醒目，更能吸引观众的注意力。当然这是在恰当合适使用的前提下，如果使用不当，它也能引起观看者的反感，甚至厌恶，从而影响到实际的广告效果。设计横幅广告时，究竟是采用静态形式还是动态形式，取决于哪种形式能够把要表达的信息准确、快速地传递给观看者。在本章中，我们将一起尝试设计一个动态横幅广告，向各位详细地讲解设计动态横幅广告的相关知识。

设计效果预览

transition 是 CSS3 新增的动画属性，它允许 CSS 的属性值在一定的时间区间内平滑地过渡，从而产生过渡的动画效果。在本章中，我们将使用这个新添加的属性来设计动态横幅广告。学习本章内容，需要各位具备 HTML5、CSS3 和 jQuery 的基本知识，如果你尚未掌握这些基础知识，请返回到第 2 章中进行学习。在这里，我们假定你已经掌握了这些基础知识，下面我们将应用这些基础知识以及 CSS3 新添加的 transition 动画属性，学习制作动态横幅广告。下面两幅图是已经制作好的横幅广告，左图是鼠标指针移动到横幅广告之前的效果，右图是鼠标指针移动到横幅广告上的效果。

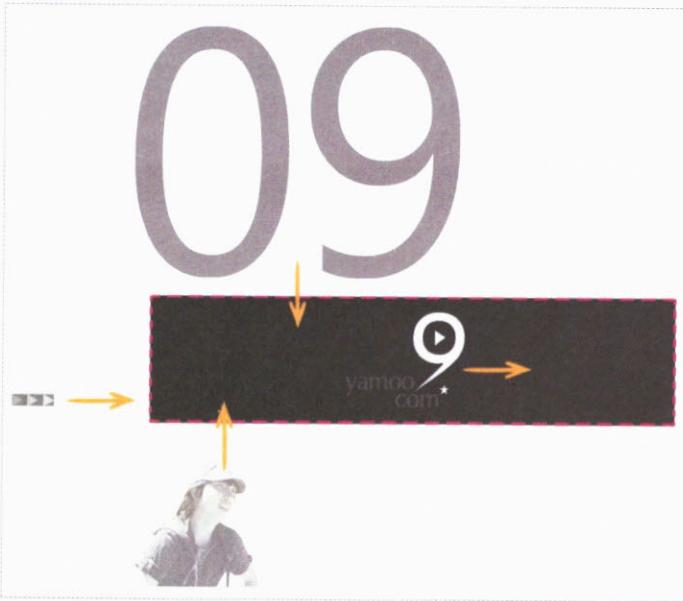


鼠标指针移动到横幅广告之前



鼠标指针移动到横幅广告上

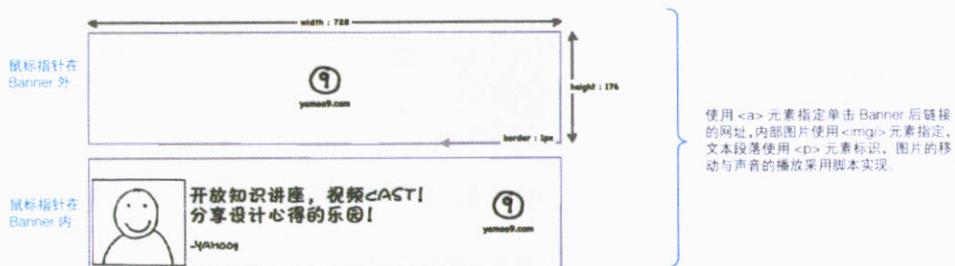
在这里，需要注意的不是在鼠标指针移动到横幅广告前后两幅图片的切换，而是人物图片自外向里进入，位于中心部位的 Logo 元素向右滑动，以及文本显现的动画。在本章中，我们将使用 CSS3 的 transition 属性来实现上述这些动画效果。



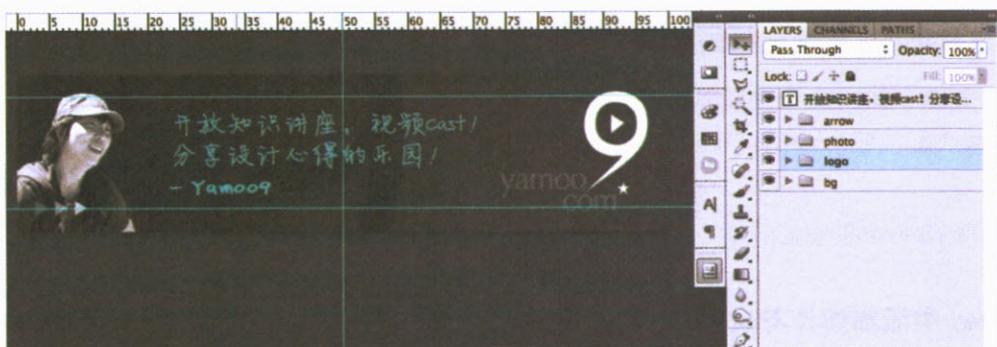
CHAPTER 3

设计网页布局

首先使用 Balsamiq Mockups 软件，将脑海中网页的布局描绘出来。



接下来，我们使用 Photoshop 等图形图像工具把上面的设计布局用实际的图片、文字表现出来。当然，你也可以不用 Photoshop，而用其他的图形图像处理工具，只要你觉得顺手即可。使用图形图像处理工具将设计布局草图表现出来，这是个非常重要的过程，通过它我们可以轻松地把握制作的整个过程，在脑子中思考所需要的技术，确定 Banner 的大小，要传递给观看者的文字信息，以及 Logo 与背景图片等。



作为一个网页设计师，设计精美的图像是最基本的要求，此外还要考虑技术方面的问题，思考用哪种技术设计出最符合要求的效果。制作图像时，如果不理会技术，则会难以通过技术来实现它。即使有相应的技术，如果开发者或设计师不懂这项技术，同样也无法实现。这个时候，如果你选用熟悉的技术，则可能无法实现预期的效果。因此，一个优秀的设计师应该不断关注学习新技术，并把这些新技术应用到自己的设计中，最终获取预期的设计效果。

编写 HTML5 代码

首先下载例题源文件，解压缩后，在文件夹 \ex_03\Begin 中存在一个名为 banner-design.html 的 HTML 文件，使用网页编辑器打开它。在打开的 HTML5 文档中，可以看到 HTML5 DTD 与基本的 html 语句。在 <head> 部分调用了外部 3 个文件（css/banner.css、js/jquery.min.js、js/banner.js）。接下来，我们将在 <body> 中输入 HTML5 代码。



小知识 ↗ UTF-8

在使用网页编辑器打开 banner-design.html 文档后，可能会出现乱码的情形，原因是文字编码没有设置为 UTF-8 格式，这时需要将网页编辑器的文字编码设置为 UTF-8 格式。如果使用的是 Notepad++ 编辑器，则可以在菜单栏中依次单击“格式 – 以 UTF-8 格式编码”，更改文字编码。若使用的是 CODA 编辑器，则请依次单击“Text-Encoding – Unicode(UTF-8)”，更改文字编码。

输入单击 Banner 时要链接的网站

<a> 元素用于定义锚，其最重要的属性是 href 属性，使用该属性可以指定链接的目标网站。在 <body> 中输入如下代码，并通过 href 属性指定链接网站。

```
<a href="http://yamoo9.com"></a>
```

向 Banner 中添加图片与文字

在 HTML5 之前的版本中，<a> 元素被定义为内联元素（inline element）。内联元素一般都是基于语义级（semantic）的基本元素，只能容纳文本或者其他内联元素。而在 HTML5 中，<a> 被定义为可以包含块元素（block element）的元素。在 <a> 内部中插入 元素，通过指定 src 属性插入 images/banner-logo.png 图片，设置 alt 属性来指定图像的替代文本，height 与 width 分别用来设定图像的高度与宽度。然后使用 <p> 标签向 Banner 中添加广告文本，并使用
 插入简单的换行符，对文本进行换行。最后使用 标签对重要的文本进行强调，但通过使用 css 样式表可以取得更丰富的效果。

```

<a href="http://yamoo9.com">
    
    <p>开放知识讲座，视频cast！<br/> 分享设计心得的乐园！<br/>
        <strong>- Yamoo9</strong></p>
</a>

```

小知识 内联元素（inline element）与块元素（block element）

在 HTML 文档中元素大致可以分为内联元素（inline element）与块元素（block element）两种。块元素可以包含其他块元素或内联元素，如 h1~h6、div、p、ul 和 ol 等。内联元素位于块元素内部，它不能包含块元素，如 a、img、strong、em 和 span 等。但需要特别注意的是，从 HTML5 开始，内联元素 `<a>` 可以包含块元素了，即在 `<a>` 元素内部可以出现标题（h1~h6）、段落（p）和列表（ul、ol）等块元素。但是在 `<a>` 内部仍然不能包含支持点击的交互元素，像 `<a>`、`<button>` 等。

strong 元素与 bold 元素

当使用 `strong` 元素或 `bold` 元素对文本进行标识时，文本都会以粗体进行显示，仅从视觉效果上无法区分出两种元素有什么不同，但它们的含义是不同的。`bold` 元素仅从视觉效果上对指定的文本进行凸显强调，而 `strong` 元素不仅从视觉效果上凸显强调指定的文本，而且它所包含的文本内容也比其他文本更重要。如果某些文本需要从意义上进行凸显强调，建议使用 `strong` 元素。

使用 class 属性标识各元素

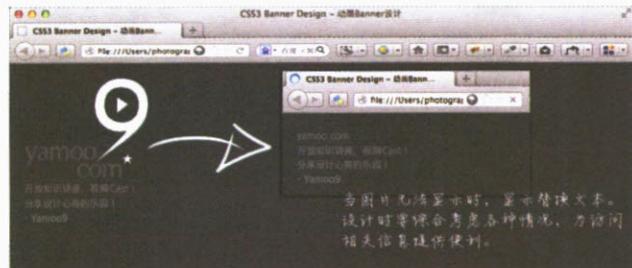
在 HTML 文档中可以为元素指定 `class` 属性，以将其与其他元素区分开来，便于使用 CSS 样式表分别控制它们的样式。在本实例中，我们将分别为 `<a>`、``、`<p>` 元素指定 `class` 属性：`banner`、`banner-logo`、`banner-desc`，以便在 CSS 中通过 `(banner, banner-logo, banner-desc).class` 区分各元素并分别指定不同的样式。

```

<a class="banner" href="http://yamoo9.com">
    
    <p class="banner-desc">开放知识讲座，视频cast！<br/> 分享设计心得的乐园！<br/>
        <strong>- Yamoo9</strong></p>
</a>

```

至此，网页横幅广告的 HTML5 代码编写完成，按 `Ctrl+S` 键保存，而后在浏览器中打开它，查看显示效果是否正常。在主机环境（关于主机环境搭建的内容，请参考第 1 章第 27 页中的相关内容）下若图片无法显示，图片的替代文本就会显示出来。为了应对图片无法正常显示的情况，最好养成提供图片替代文本的习惯。



→ 小知识 ← QuickJava

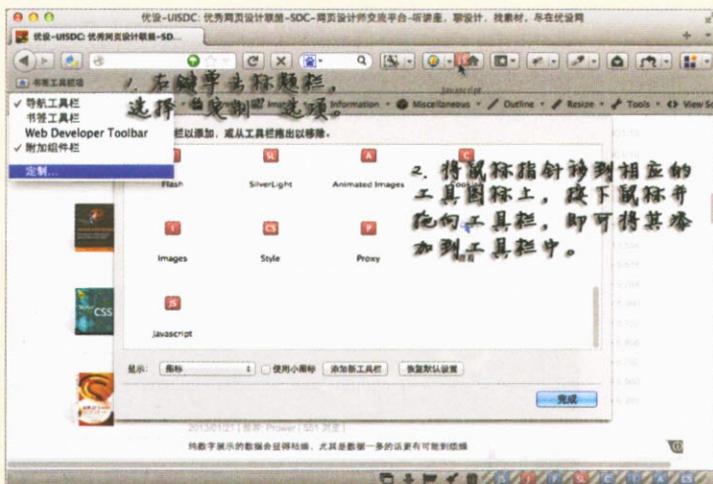
QuickJava 是 Firefox 浏览器的一个附加组件，安装好该组件后，可以在 Firefox 浏览器的状态栏或工具栏中快速启用和禁用 images、JavaScript、CSS 和 Flash 等，而不需要打开选项对话框。

在 Firefox 浏览器中安装 QuickJava 组件的步骤如下。

- ① 在 Firefox 浏览器左上角，单击 Firefox 按钮，在弹出的菜单中，单击“附加组件”，打开“附加组件管理器”窗口。
- ② 在位于“附加组件管理器”窗口右上角的搜索框中，输入“quickjava”，单击“Enter”按键进行搜索。
- ③ 在“可用附加组件”列表中，单击“QuickJava”组件右侧的“安装”按钮进行下载安装。



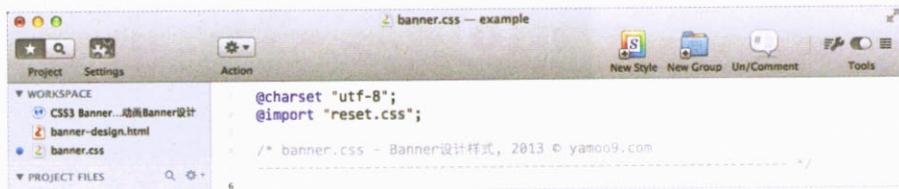
安装完毕后，重启 Firefox 浏览器，在浏览器窗口的右下角显示出 JS、J、F、SL、I、CSS、P 等开关按钮。当按钮是蓝色时处于开启状态，单击一下蓝色按钮即变为红色，由开启状态转换为关闭状态。再单击一下红色的按钮，红色变为蓝色，状态由关闭转换为开启。通过单击按钮，即可开关相应的项目，使用起来非常方便。若开关按钮未显示出来，请单击浏览器左上角的 Firefox 按钮，在弹出菜单中，依次选择“选项 - 附加组件栏”，即可将它们显示出来。若想把这些开关按钮添加到工具栏中，① 移动鼠标指针到浏览器的标题栏，单击鼠标右键，在弹出的菜单中选择“定制”，打开“定制工具栏”对话框，② 选择需要添加的项目，按住鼠标左键，将其拖动到工具栏中，即可添加完毕。



编写 CSS3 样式表



首先进入 ex03\Begin\css 文件夹，打开名为“banner.css”的样式表文件。在“banner.css”的样式表中仅有两行语句，@charset 语句用于设置网页的字符编码，@import 语句用于将外部的 reset.css 样式表文件引入到 banner.css 文件中（关于 reset.css 样式表的详细内容，请参考第 6 章的相关内容）。



控制 body 样式

首先设置 Banner 背景颜色（background-color）为深灰色，设置内边距（padding）为 20px。

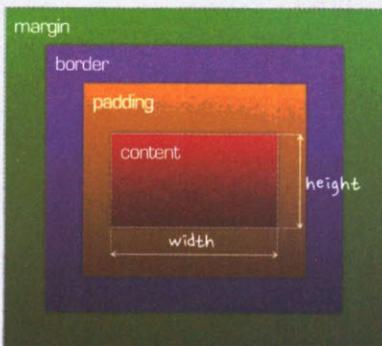
```
body {  
    padding: 20px;           设置内部空白为20px  
    background-color: #333333;  设置背景色为深灰色  
}
```

控制 Banner 样式

下面开始控制 Banner 样式，整个 Banner 由①banner 区块（banner）、②banner 的 Logo 图片（banner-logo）和③banner 文本（banner-desc）三部分组成。首先控制 banner 区块的样式，整个 banner 区块由[元素定义，在 CSS 中我们使用 a.banner 来标识它。a.banner 是类选择器，其中定义的 CSS 样式仅对 class 为 banner 的\[元素起作用。在这里，我们将设置\\[元素的 display、width、height 和 border 这 4 个属性，用来控制 Banner 的显示方式、尺寸和边框。\\]\\(#\\)\]\(#\)](#)

```
a.banner {  
    display: block;          将a.banner转换为块元素  
    width: 728px;            Banner宽度  
    height: 176px;           Banner高度  
    border: 1px solid #555;   边框粗细为1px，边框样式为实线，边框颜色为中灰  
}
```

小知识 CSS 盒状模型



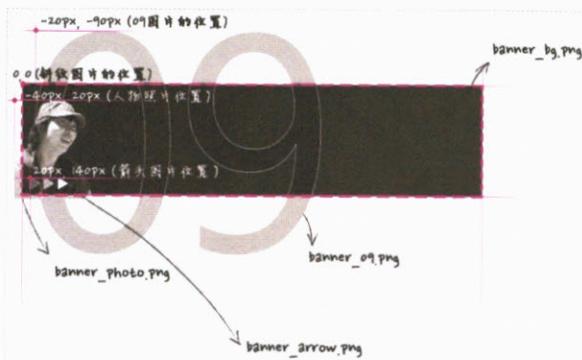
CSS 盒状模型是大多数 CSS 布局与定位的基础，它以包含 4 个成分的有界限的盒子来描述网页中的每个元素。盒状模型示意图如左图所示，盒子的尺寸 = 内容尺寸 (width+height) + 内边距 (padding) + 边框粗细 (border) + 外边距 (margin)。

Banner 样式控制代码输入完毕后，按 Ctrl+S 键保存，按 F5 键刷新页面，查看样式控制效果，如右图所示。



向 Banner 插入背景图片

在 CSS3 中，我们可以为一个元素设置多个背景图片，而后为这些图片指定不同的位置，从而使图片呈现出多种排列形式。首先进入 Begin\images 文件夹，该文件夹含有 banner-bg.png、banner-09.png、banner-photo.png、banner-arrow.png 等 4 张图片，这 4 张图片将被用作 Banner 的背景。这里，需要注意的是，各种背景图片的位置。请看右面的示意图，弄清楚图片放置的位置以及如何放置这些图片。



下面我们开始编写代码部分。在代码编写过程中，将使用 background 简写属性把 4 张图片添加到背景中。使用 url() 指定要插入的图片的路径，若要插入多张图片，各张图片之间要使用逗号 (,) 分隔开来。no-repeat 设定图像不重复，然后根据上面的示意图为各张图片指定位置。各张图片按照代码编写的顺序从前往后排，即最后插入的图片位于最底层。

```
background:  
url(..../images/banner-arrow.png) no-repeat 20px 140px,  
url(..../images/banner-photo.png) no-repeat -40px 20px,  
url(..../images/banner-09.png) no-repeat -20px -90px,  
url(..../images/banner-bg.png) no-repeat 0 0;
```

开始 background 的语句简写属性
添加箭头图片，并调整位置
添加图片并调整位置
添加 09 图片，调整位置
添加图片，并调整位置

→ 小知识 ← background 简写属性

在 CSS 中有多个属性用于设置背景样式，其中 `background-color` 用于设置背景颜色，`background-image` 指定要使用的背景图像，`background-repeat` 设置是否以及如何重复背景图像，`background-position` 属性设置背景图像的起始位置，`background-attachment` 属性设置背景图像是否固定或者随着页面的其余部分滚动。在使用这些属性设置背景样式时，既可以分别使用单个属性，也可以使用 `background` 简写属性，在一个声明中设置所有背景属性。建议使用 `background` 简写属性，而不是分别使用单个属性，因为这个属性在较老的浏览器中能够得到更好的支持，而且需要键入的字母也更少。

在使用 `background` 简写属性时，可以按照如下顺序依次设置各个属性，也可以不设置其中的某个值，CSS 会自动将其设置为默认值。比如，当不设置背景颜色时，背景颜色会被自动设置为透明（`transparent`）。

<code>background:</code>	<code>background-color</code>	<code>background-image</code>	<code>background-repeat</code>
	背景色	背景图像	是否以及如何重复图像
	<code>background-attachment</code>	<code>background-position</code>	
	是否固定	图像位置	

代码编写完成后，按 `Ctrl+S` 进行保存，而后在浏览器中按 `F5` 键刷新网页，查看设置的背景样式是否正常。



设置横幅广告的 Logo 图像

当前 Banner 的背景图像与 Logo 图像重叠在一起，看上去十分不美观，并且也不符合网页的设计要求。下面我们要把 Banner 的 Logo 图像移动到右侧位置上，Logo 图像的类名为 `.banner-logo`，在 CSS 中它被称为 Logo 图像的类选择器，在 CSS 中使用该选择器即可对 Logo 图像的样式进行控制。首先使用 `position` 属性指定 Logo 图像为绝对定位，而后分别指定 `top` 与 `left` 值，确定相对于参考元素的位置。

```
a.banner .banner-logo {  
    position: absolute;          绝对定位  
    top: 20px;                  上端距离基准元素为20px  
    left: 540px;                左端距离基准元素为540px  
}
```

那么，在这里我们的参考元素是什么呢？事实上，我们并未明确指定参考元素，此时默认网页的左上角为参考位置。由于在 CSS 中为 Logo 图像设置 `top` 为 `20px`，`left` 为 `540px`，所以 Logo 图像距离网页左侧为 `540px`，距离网页顶端为 `20px`，如右图所示。



默认参考位置为网页的左上角

下面我们更改一下定位的参考元素，将其改为整个 Banner (a.banner)，在 a.banner 样式控制代码中添加一行代码：position: relative;。这样设置之后，.banner-logo 定位时就以 a.banner 对象的左上角为基准，向右、向下分别移动 540px 与 20px。

```
a.banner {  
    position: relative;           设置为相对定位，作为.banner_logo的基准对象  
    display: block;  
}
```

保存代码，按 F5 键刷新网页，可以看到 Logo 图像移动到指定的位置上，如右图所示。



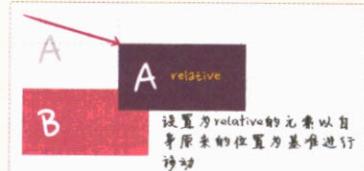
设置参考元素为 Banner

→ 小知识 ← 绝对定位、相对定位、固定定位

定位是实现 CSS 布局控制中最重要、最常用的功能，也是网页设计的难点。如果不理解定位，那么可能实现不了想要的效果，或者实现的结果与预期的要求不同。CSS 使用 position 属性来指定元素的定位类型，该属性有 4 种不同类型的定位供选择使用，分别是 static（默认定位）、relative（相对定位）、absolute（绝对定位）和 fixed（固定定位），其中 static 是所有元素默认的定位设置，遵循基本的定位规定，没有特别的设定。

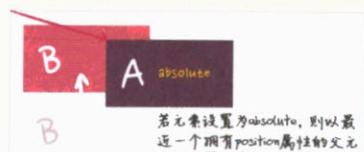
relative（相对定位）

设置为相对定位的元素不脱离文档流，参考自身位置通过 top、bottom、left 和 right 进行定位，元素仍然保持其未定位前的形状，它原本所占的空间仍然保留，因此采用相对定位的元素有可能覆盖其他元素。比如，设置 A 元素的 position 属性为 relative（相对定位），该元素将以自身原来的位置为基准进行移动，基准为 A 元素框的边角，如设置了 top、right 值，该元素将以本身的右上角为基准根据 top、right 值进行相应移动。



absolute（绝对定位）与 fixed（固定定位）

与 relative（相对定位）不同，设置为 absolute（绝对定位）与 fixed（固定定位）元素脱离了文档流，元素原先在正常文档流中所占的空间会关闭，就像该元素原来不存在一样。absolute（绝对定位）与 fixed（固定定位）看上去非常类似，但实际上它们之间是有明显差别的。absolute（绝对定位）元素的位置相对于最近的已定位的祖先元素，若元素没有已定位的祖先元素，那么它的位置相对于最初包含块。而 fixed（固定定位）则是相对于浏览器窗口的，即将设置为 fixed（固定定位）的元素固定在浏览器的某个位置上，即使拖动浏览器的滚动条，该元素的位置也不会改变。



如上图，若 A 元素被设置为 absolute（绝对定位），定位时则以最近的已定位的祖先元素为基准。需要注意的是，A 元素原先所占的空间会关闭，并被其他元素（B）占据。



如上图，若 A 元素被设置为 fixed（固定定位），其定位时将以浏览器窗口为基准，被固定在某个指定的位置上，即使拖动窗口滚动条，它的位置也不会改变。与 absolute（绝对定位）一样，定位时它原来所占的空间也会被关闭。

向 Banner 上的文字应用字体

Banner 上文字的类名为 banner-desc。在 CSS 中，我们使用 .banner-desc 选择器为 Banner 上的文字指定样式。在这里，我们使用 font 简写属性为文字设置字体大小 (font-size) 、行间距 (line-height) 和字体系列 (font-family) 等。

```
a.banner .banner-desc {  
    font: 39px/0.9 "NanumPenWeb", "方正静蕾简体", "Nanum Pen Script";    设置网页字体文件  
}
```

↔ 小知识 ↔ font 简写属性

在 CSS 中用于设置字体样式的属性有字体系列 (font-family) 、字体大小 (font-size) 、行间距 (line-height) 、字体风格 (font-style) 和字体粗细 (font-weight) 。此外，还有一个名为 font-variant 的属性，它用来设置小型大写字母的字体显示文本，仅用于英文字体中。与 background 简写属性不同，编写 font 简写属性需要遵循以下几条规则。

① 至少要指定字体大小与字体系列（必须）。

例) font: 12px "Times New Roman";

② 行间距添加到字体大小之后，并且使用斜杠 (/) 分割（可选）。

例) font: 12px/1.5 "Times New Roman";

③ 设置字体粗细、字体风格和字体变体属性时，要添加到字体大小之前（可选）。

例) font: bold italic small-caps 12px/1.5 "Times New Roman";

在这里，我们将 Banner 上的文字设置为 “Nanum Pen” 字体。由于 “Nanum Pen” 字体不是电脑默认的字体，因此这种字体仅在安装了该字体的电脑上才能正常显示出来。在网页设计中，如果使用了电脑非默认的字体，所使用的字体效果就无法正常显示出来，这就大大背离了设计的初衷。为了解决这个问题，设计师们常常会使用 Photoshop 等图像处理软件把要使用的字体做成图片，而后插入到网页中，这是一种非常好的处理方法。但从 CSS3 开始，我们可以使用 @font-face 技术将自定义的 Web 字体嵌入到网页中，使未安装该字体的电脑也能正常显示它。

若想使用 CSS3 的这种 Web 字体技术需要经过几个略显繁琐的过程，因为目前各种浏览器所支持的字体格式有所不同，比如，*.eot 文件仅有 IE 浏览器支持，它是 MS 专用的压缩字库，用来解决在网页中嵌入特殊字体的难题。因此，我们往往需要准备多种格式的文件，以匹配各种不同的浏览器。

在 CSS3 中使用 Web 字体技术，需要先变换格式之后再进行调用。在这里我们使用一种更加简便的方法，即使用提供 Web 字体服务的 mobilis 公司的 Web 字体。首先进入 mobilis 公司的 Web 字体页面 (<http://api.mobilis.co.kr/webfonts>)，该页面中记述了 mobilis 公司 Web 字体的使用方法。

모빌리스 웹폰트 서비스

그림이 아닌 진짜 폰트를 사용해 보세요!!!

웹페이지 디자인에서 타이포그래피가 중요해 지면서, 다양한 폰트를 이용하고자 하는 욕도 커지고 있습니다. 그러나 웹브라우저마다 다른 폰트 방식과 한글폰트 파일의 크기로 인해 웹페이지에서 원활한 한글폰트 이용이 어려웠습니다. 이로 인해 웹페이지 디자이너나 한글폰트가 아닌 한글을 이용해 웹페이지에서 타이포그래피를 구현하는 것이 일반적인 현상이 되었습니다.

모빌리스 웹폰트 서비스는 웹페이지 디자이너뿐만이 아니고 개발자들도 블로그나 웹페이지에 그림이 아닌 진짜 폰트를 쉽게 이용할 수 있도록 제공합니다. 무거운 한글 웹폰트를 기법에 전송해 주고, 웹페이지 방문자와 웹브라우저가 무엇이든 관계없이 동일한 웹폰트를 방문객에게 보여줍니다. 무료로 사용할 수 있는 아래 웹폰트 목록에서 마음에 드는 웹폰트를 선택하면, 어떻게 웹폰트를 사용할 수 있는지 자세히 설명해 드립니다.

모빌리스 웹폰트 서비스는 개인 또는 비영리조직에 한해서 자유롭게 이용할 수 있습니다. 단, 모든 기체는 각 폰트 저작권자와 라이선스 정책에 따릅니다. 기업 및 상업용 사이트의 경우 marketing@mobilis.co.kr로 문의 바랍니다.

웹개발자들을 위해 jQuery 플러그인으로 작성된 정교한 웹폰트 라이브러리도 함께 제공합니다.

[나눔글꼴](#) [온글꼴 1](#) [온글꼴 2](#) [백목글꼴](#) [혁시글꼴](#) [온진글꼴](#)

小知识

目前 mobilis 公司提供的 Web 字体是免费开放字体，使用时不会引发著作权等问题，但仅针对个人或非盈利组织免费。大多数专业的字体制作公司提供的 Web 字体都是有偿服务，使用时需要支付一定的费用。

浏览器转到 NanumPenWeb 字体的详细信息页面后，向下拖动浏览器的滚动条，在“Web 字体使用方法”中，拖选字体调用代码，然后复制。

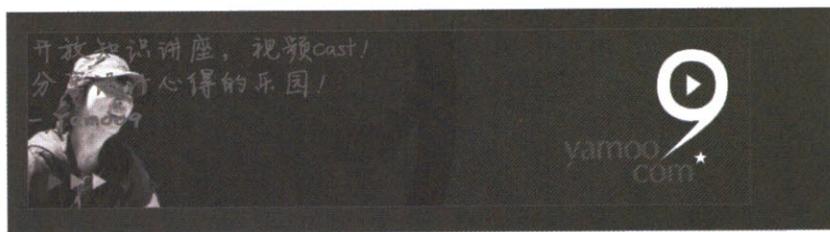
웹 폰트 사용 방법

웹브라우저에 관계없이 여러분의 웹페이지에 한글 폰트를 사용하려면, 먼저 HTML 문서에서 <head> 태그 안에 아래와 같이 모빌리스 웹폰트 API를 호출하는 코드를 넣습니다.

```
<head>
  <link href='http://api.mobilis.co.kr/webfonts/css/?fontface=NanumPenWeb'
        rel='stylesheet' type='text/css' />
  ...
</head>
```

在网页编写工具中，打开 banner_design.html 文档，将上面复制的代码插入到 <title> 标签之下，而后保存文档。在浏览器中打开它，可以看到 Banner 的文本已经被应用了 NanumPenWeb 字体，表明浏览器正常调用 mobilis 公司提供的 Web 字体服务。

```
<title>CSS3 Banner Design - 动画Banner设计</title>
<link href='http://api.mobilis.co.kr/webfonts/css/?fontface=NanumPenWeb' rel='stylesheet' />
```

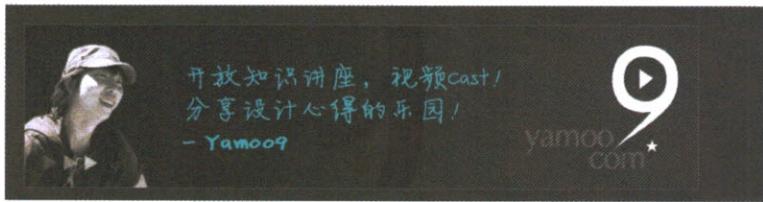


设置 Banner 文字的位置与颜色

接下来，我们要把 Banner 文字移动到人物图片右侧，并且将字体颜色更改为亮青色。

```
a.banner .banner-desc {
  position: absolute;
  top: 35px;
  left: 170px;
  font: 39px/0.9 "NanumPenWeb", "方正静蕾简体", "Nanum Pen Script";
  color: #4ec1cd;
}
```

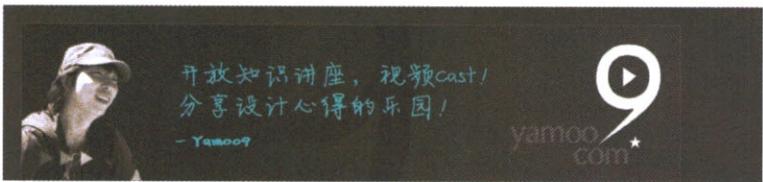
绝对定位
从基准对象向下移动39px
从基准对象向右移动170px
设置字体颜色



最后, 将“- yamoog.com”的字号设置为 23px。由于在 html 文档中“- yamoog.com”被包含在 **标签中, 所以在 CSS 样式表中使用 a.banner.banner-desc strong 选择器控制它的字体样式。**

```
a.banner .banner-desc strong {  
    font-size: 23px;           设置字体大小  
}
```

CSS 代码编写完成后保存, 按 F5 键刷新页面, 可以看到“- yamoog.com”字体变小了。接下来, 开始设计鼠标指针移动到 Banner 上之前与之后(或按 Tab 键使 Banner 失焦或获得焦点)的 Banner。



设计鼠标指针未移动到 Banner 上的 Banner

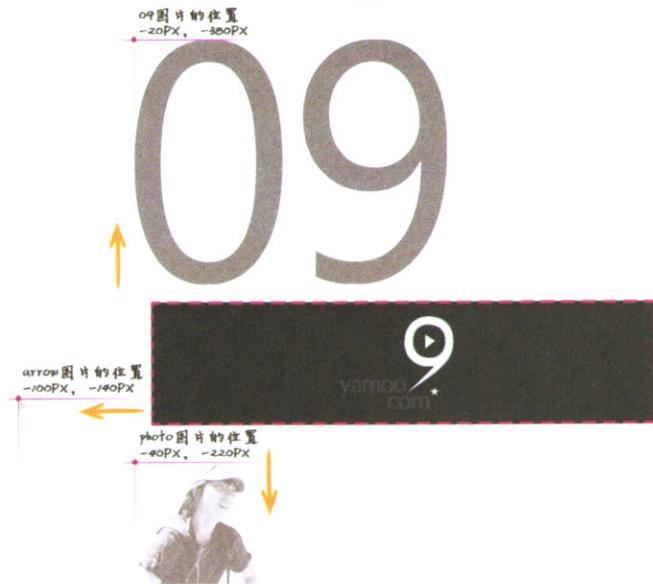
首先制作鼠标指针移动到 Banner 上之前的状态。CSS 的 :hover 伪类在鼠标指针移动到某个元素上时向该元素添加某些样式; :focus 伪类用于在元素获得焦点(按键盘上的 Tab 键)时向元素添加某些样式。在这里, 我们将使用 :hover 与 :focus 伪类制作鼠标指针移动到 Banner 上之前与之后的 Banner。

小知识 伪类

在 CSS 中通过为元素指定 id 或 class 来标识特定元素。若想在 CSS 中使用元素的标识来指定元素样式, 需要在 HTML 文档中先为相应的元素设定 id 或 class 属性值(比如: <p class="memo">…</p>)。CSS 伪类是选择符的螺栓, 用来指定一个或者与其相关的选择符的状态, 它们的形式是 selector:pseudo class{property:value}, 注意要使用半角英文冒号(:)来隔开选择符与伪类。在 CSS 中存在多种伪类, 其中与超链接有关的伪类如下表所示。

种类	说明
:link	向未被访问的链接添加样式
:visited	向已被访问的链接添加样式
:hover	当鼠标指针悬浮在元素上方时, 向元素添加样式
:active	向被激活的元素添加样式
:focus	向获得焦点(比如, 被点击或键盘输入等)的元素添加样式

我们把鼠标指针移动到元素之前的状态，称为“Out”，而把鼠标指针置于元素之上的状态，称为“Over”。在 Out 状态下，横幅广告（Banner）上的人物图片、数字 09 和箭头都消失不见了，Banner 上的文字也不可视，并且 Logo 处于 Banner 的中间位置。接下来，我们要根据移动进行设计，在 Out 状态下处于不可视的图片、箭头和数字 09 的方向要准确地进行设定才行。



根据上图各个元素的位置，编写如下代码。

```
a.banner {  
    position: relative;  
    display: block;  
    width: 728px;  
    height: 176px;  
    border: 1px solid #555;  
    background:  
        url(..../images/banner-arrow.png) no-repeat -100px 140px,  
        url(..../images/banner-photo.png) no-repeat -40px 220px,  
        url(..../images/banner-09.png) no-repeat -20px -380px,  
        url(..../images/banner-bg.png) no-repeat 0 0;  
}  
  
a.banner:hover, a.banner:focus {  
    background-position:  
        20px 140px,  
        -40px 20px,  
        -20px -90px,  
        0 0;  
}
```

画面上的各个图片改变位置隐藏
根据移动改变方向
使图片在画面上不可见
当鼠标指针移上或获得焦点时，更改背景位置

代码编写完成后保存，在浏览器中查看效果，可以看到背景图片中仅有 banner-bg.png，其余图片都处于不可见状态。当进入 Over 状态下时，必须重新设置各个图片。当鼠标指针移动到 Banner 之上，或者按 Tab 键使 Banner 获得焦点时，一切动作正常。



下面我们开始向 Banner 设置 Logo，并设置 Banner 文本。首先设置 Banner 文本的“不透明度”(Opacity) 为 0，使其处于隐藏不可见状态。

```
a.banner .banner-desc {  
    opacity: 0;          // 设置不透明度为0，使其不可见  
    position: absolute;  
}
```

当鼠标指针移动到 Banner 之上或获得焦点时，Banner 文本要显示出来，所以我们将其“不透明度”(Opacity) 设置为 1，如下所示。

```
a.banner:hover .banner-desc,  
a.banner:focus .banner-desc {  
    opacity: 1;          // 设置不透明度为1，当鼠标指针移到Banner上或获得焦点时显示  
}  
}
```

保存代码，在浏览器中查看效果，可以看到 Banner 上的文本消失不见了。



接下来，我们要把 Logo 设置到横幅广告的中间位置上。修改 .banner-logo 代码，插入 left:270px 一行，使 Logo 位于 Banner 的中间位置。这里，仅设置 X 轴（水平方向）即可。

```
a.banner .banner-logo {  
    position: absolute;  
    top: 20px;  
    left: 270px;        // 使Logo位于Banner的中央  
}
```

当鼠标指针移动到 .banner 之上或 .banner 获得焦点后，.banner-logo 要移动到横幅广告的右侧，在 CSS 中编写如下代码：

```
a.banner:hover .banner-logo,  
a.banner:focus .banner-logo {  
    left: 540px;                                设置Logo的left  
}  
...
```

代码编写完成后保存，在浏览器中查看效果，可以发现 Logo 移动到了 Banner 的中间位置上。至此，鼠标分别在 Out 与 Over 状态下的设计工作完成。



下面我们开始做一个有趣的效果，使元素在移动时画面能够平滑地进行切换。要制作这种效果，需要使用 CSS3 提供的 transition 属性，该属性可以控制 HTML 组件的某个属性发生改变时会经历一段时间、以平滑渐变的方式发生改变。

首先要应用 transition 属性的是背景图像，当从 Out 状态转换为 Over 状态时，各个背景图像要以平滑的方式进行移动。在 a.banner 选择器中，在 background 属性下添加 transition 属性，并设置其值为 background-position .2s ease-in .2s，代码如下：

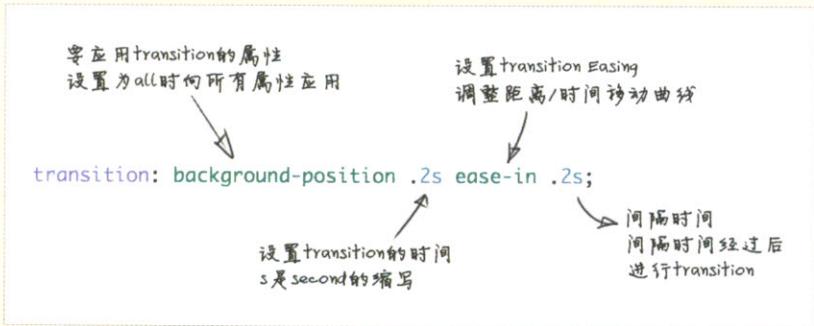
```
a.banner {  
    background:  
        url(..../images/banner-arrow.png) no-repeat -100px 140px,  
        url(..../images/banner-photo.png) no-repeat -40px 220px,  
        url(..../images/banner-09.png) no-repeat -20px -380px,  
        url(..../images/banner-bg.png) no-repeat 0 0;  
    transition: background-position .2s ease-in .2s;      在0.2秒后，平滑切换画面0.2秒  
}
```

小知识 Transition 动画

CSS3 提供了对 Transition 动画的支持，控制 HTML 组件的某个属性发生改变时会经历一段时间，以平滑渐变的方式发生改变。Transition 动画通过 transition 属性来指定，transition 属性值包含如下 4 个部分。

属性	说明
transition-property	指定对 HTML 元素的哪个 CSS 属性进行平滑渐变处理。若为所有属性，则应使用 all 指定
transition-duration	指定属性平滑渐变的持续时间，默认值为 0，设置时要用正数
transition-timing-function	指定平滑渐变过程中使用的效果，预设有 linear、ease、ease-in、ease-out 和 ease-in-out 等
Transition-delay	指定动画开始执行的时间，即指定经过多少时间延迟后才开始执行平滑渐变

与 background 或 font 一样，transition 也支持属性连写，若省略其中的某个属性，则自动应用默认值。



上述语句的含义为：当经过 0.2 秒之后，背景位置（background-position）开始进行 ease-in 平滑渐变，持续时间为 0.2 秒。

关于平滑渐变过程中使用的效果，即平滑渐变如何移动，预设主要有 4 种取值，如下表所示。

easing 属性值	曲线
Linear 线性速度，动画从开始到结束速度保持不变	
Ease-in 动画开始时速度较慢，然后速度加快	
Ease-out 动画开始时速度很快，然后速度减慢，与 ease-in 相反	
Ease-in-out 动画开始时速度较慢，然后速度加快，达到最大速度后再减慢速度	

代码编写完成后保存，在浏览器中查看效果，发现网页没有任何变化。原因是 CSS3 Transition 模块规范尚未完成，目前各主流浏览器暂未支持 transition 属性。为了在浏览器中测试这种新模块，各浏览器厂商分别提供了相应的前缀，比如 Firefox 浏览器的前缀为 “-moz-”，只有在 transition 属性前添加上这一前缀，Firefox 浏览器才能识别并使之正常工作。

同样，其他浏览器也分别提供了不同的前缀，在实际开发中我们需要分别添加这些浏览器厂商的前缀。随着浏览器版本的升级，当它们开始支持 transition 属性时，就不需要再添加这些前缀了。但目前为止，在使用 transition 属性时，我们仍然需要添加这些前缀，各浏览器需要添加的前缀如下表所示。

前缀	浏览器
-moz-	Mozilla 系列浏览器 (Firefox)
-webkit-	基于 Webkit 的浏览器 (safari、Chrome)
-o-	Opera 浏览器
-ms-	Internet Explorer 浏览器
-khtml-	Konqueror 浏览器

在 transition 代码下插入如下代码，以在 Mozilla Firefox 浏览器中实现 transition 属性。

```
transition: background-position .2s ease-in .2s;
-moz-transition: background-position .2s ease-in .2s;      在transition属性前添加前缀
}
```

若想在 Safari、Chrome、Opera 和 Internet Explorer 浏览器中使 transition 属性动作正常，需要添加如下代码。在实际开发中为了使 transition 属性在各个浏览器中都能工作正常，我们需要分别为各种浏览器添加前缀。虽然这可能会使代码显得有些混乱，但这也是没办法的事，为了让 transition 属性在各浏览器中正常工作，我们必须这样做。当各家浏览器开始正式支持 transition 属性时，我们就不再需要重复这些类似的代码了，但到目前为止，我们不得不一一手动添加这些类似的代码。

接着，我们开始向 Banner 的 Logo 与文本添加 Transition 动画。首先向 .banner-logo 添加 transition 效果。

```
transition: background-position .2s ease-in .2s;
-webkit-transition: background-position .2s ease-in .2s;
-moz-transition: background-position .2s ease-in .2s;
-o-transition: background-position .2s ease-in .2s;
-ms-transition: background-position .2s ease-in .2s;
```

```
a.banner .banner-logo {
    position: absolute;
    top: 20px;
    left: 270px;
    transition: all .4s ease-out .3s;
    -webkit-transition: all .4s ease-out .3s;
    -moz-transition: all .4s ease-out .3s;
    -o-transition: all .4s ease-out .3s;
    -ms-transition: all .4s ease-out .3s;
}
a.banner:hover .banner-logo,
a.banner:focus .banner-logo {
    left: 540px;
}
```

```
a.banner .banner-desc {
    opacity: 0;
    position: absolute;
    top: 39px;
    left: 170px;
    font: 32px/1.1 "NanumPenWeb", "方正静善简体", "Nanum Pen Script";
    color: #4e4c1d;
    transition: all .4s ease-out .3s;
    -webkit-transition: all .4s ease-out .3s;
    -moz-transition: all .4s ease-out .3s;
    -o-transition: all .4s ease-out .3s;
    -ms-transition: all .4s ease-out .3s;
}
a.banner:hover .banner-desc,
a.banner:focus .banner-desc {
    opacity: 1;
}
```

在设置 transition 属性时，虽然我们可以把 transition-property 指定为 left 属性，但这里我们将其设置为 all，方便以后有其他元素添加进来时应用 transition 效果。上述 transition 语句含义为：在经过 0.3 秒之后，所有属性平滑渐变 0.4 秒钟。最后，我们向 .banner-desc，即广告文本应用 transition 动画，使广告文本在经过 0.3 秒后开始平滑渐变，持续时间为 0.4 秒。

关于延迟时间与持续时间的设置，各位可以根据自己的情况设置合适的数值，不必非得与实例中的数值一样。在网页设计中没有一成不变的金律，我们要不断地尝试，不断地积累经验，要相信自己的感觉，相信自己的经验。

使用 jQuery 播放声音

到目前为止，我们制作 Banner 动画时并未使用过 jQuery。下面我们将使用 jQuery 为 Banner 添加声音，当鼠标指针移动到 Banner 上之后就会播放指定的声音，在使用 Flash 制作网页时常常会使用这种效果。那么使用 jQuery 如何实现这种效果呢？首先打开 js 文件夹中的 banner.js 文件，并添加如下代码。

```
;(function($){  
    $(function(){  
        var banner_audio = new Audio('media/banner_sound.webm');  
        banner_audio.load();  
        banner_audio.play();  
    });  
})(jQuery);
```

JavaScript自运行函数
通过jQuery ready()当文档准备好后运行
使用指定的音源创建 Audio对象，而后保存到banner_audio变量中
加载banner_audio
播放banner_audio
在内部使用\$代替jQuery

小知识 自执行函数

在 JavaScript 中不需要调用而自动执行的函数，被称为自执行函数。.

new Audio() 语句

在 HTML 5 中新增了 `<audio>` 和 `<video>` 两个元素，通过这两个元素可以在 HTML 页面中播放音频、视频。使用 `<audio>` 元素能够帮助我们在网页上轻松地播放声音，如 `<audio src="music.webm"></audio>`。在 JavaScript 中使用相关语句也能对音频进行处理，常用的语句如下表所示。

语句	说明
<code>banner_audio=new Audio();</code>	新建 Audio 对象，保存到 banner_audio 变量中
<code>banner_audio.src='音频地址';</code>	将音频地址赋给 banner_audio 对象的 src 属性
<code>banner_audio.load();</code>	加载 banner_audio 变量中的音频对象
<code>banner_audio.play();</code>	播放音频
<code>banner_audio.pause();</code>	暂停音频播放

编写完代码后保存，在网页浏览器中查看效果，声音经过短暂的播放后消失。接下来，将处理函数连接到事件监听器。查找到 Banner 后，包含到 jQuery 对象中。

```

var banner_audio = new Audio();
    在ready()内部函数中声明局部变量banner_audio，创建新Audio对象后，保存到其中
webm = isSupportWebM();
    将isSupportWebM()函数的返回值保存到webm中
if(webm) { banner_audio.src = 'media/banner_sound.webm'; }
    当支持webm格式时，将banner_audio的src属性值设为media/banner_source.webm
else { banner_audio.src = 'media/banner_sound.mp3'; }
    当不支持webm格式时，将banner_audio的src属性值设为media/banner_source.mp3
$('.banner')
    查找.banner并保存到jQuery对象中
.bind('mouseover focusin', function() {
    使用.bind()方法绑定事件与事件处理函数
    banner_audio.load();           加载banner_audio
    banner_audio.play();          播放banner_audio
})
.bind('mouseout focusout', function() {
    为mouseout focusout事件指定处理函数
    banner_audio.pause();         暂停banner_audio
});

function isSupportWebM() {
    定义isSupportWebM函数
    var tester = document.createElement('audio');
        创建audio元素，保存到局部变量tester中
    return !tester.canPlayType('audio/webm');
        使用tester的canPlayType()方法，判断是否支持webm格式
}

```

→小知识← If else语句

if else语句是条件语句，当条件为真(true)时，执行if后面的语句；当条件为假(false)时，执行else后面的语句。比如电影院职员在检票时就可以用if else语句来描述整个过程。当观众持有电影票时，就让他进入观影厅；如果观众未持有电影票，就让他先购买电影票之后再进行观影。使用if else语句描述整个过程如下：

```

if(检票 === true) {
    // 可以观影
} else {
    // 购票观影
}

```

→小知识← jQuery bind()方法

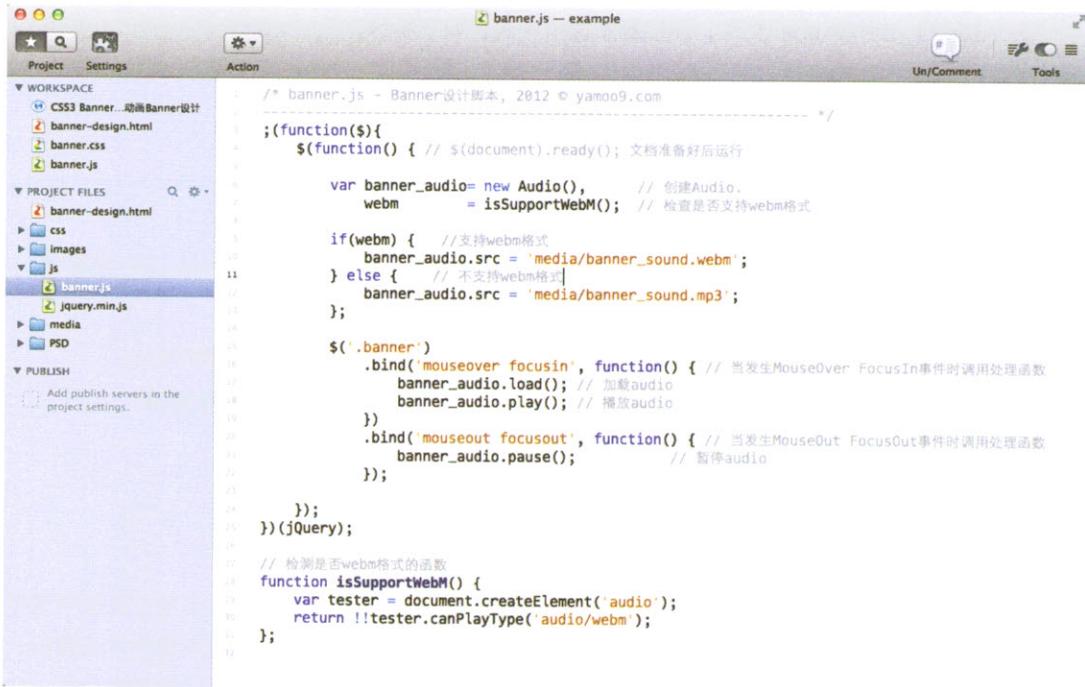
bind()方法用于为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。该方法有两个必需的参数，第一个参数规定添加到元素的一个或多个事件，由空格分隔多个事件，且必须是有效的事件；第二个必需参数用于指定事件发生时运行的函数。关于该方法更多更详细的说明，请参考<http://api.jquery.com/bind>网页中的内容。

```

清 jQuery(document.body).bind('mousedown mouseup', function(e) { console.log(e.type); });
    事件           事件处理函数

```

在将`.bind()`方法连接到`$('.banner')`上之后，当鼠标指针移动到`.banner`上或`.banner`获得焦点，它就会感知到这些事件，然后调用处理函数，加载并播放指定的声音文件（`banner_audio.load()`、`banner_audio.play()`）。通过jQuery的chain，当鼠标指针移出`.banner`或`.banner`失焦时，调用`.bind()`指定的处理函数，暂停声音播放（`banner_audio.pause()`）。脚本编写完成后保存，在浏览器中查看网页，可以发现当鼠标指针移动到Banner上时开始播放声音，移出Banner时暂停播放声音。至此，动态横幅广告就制作完成了。



The screenshot shows a web development environment with a sidebar containing project files like banner-design.html, banner.css, and banner.js. The main area displays the following JavaScript code:

```

/*
 * banner.js - Banner设计脚本, 2012 © yahoo9.com
 */
;(function($){
    $(function() { // $document.ready(); 文档准备好后运行

        var banner_audio= new Audio(), // 创建Audio
            webm      = isSupportWebM(); // 检查是否支持webm格式

        if(webm) { // 支持webm格式
            banner_audio.src = 'media/banner_sound.webm';
        } else { // 不支持webm格式
            banner_audio.src = 'media/banner_sound.mp3';
        };

        $('.banner')
            .bind('mouseover focusin', function() { // 当发生MouseOver FocusIn事件时调用处理函数
                banner_audio.load(); // 加载audio
                banner_audio.play(); // 播放audio
            })
            .bind('mouseout focusout', function() { // 当发生MouseOut FocusOut事件时调用处理函数
                banner_audio.pause(); // 暂停audio
            });
    });
})(jQuery);

// 检测是否webm格式的函数
function isSupportWebM() {
    var tester = document.createElement('audio');
    return !!tester.canPlayType('audio/webm');
}

```

到目前为止，我们编写的代码已经相当完美了，但我们可以继续进一步，将这段声音播放代码制作成jQuery插件，方便在其他文档中使用。虽然这需要我们再花些时间与精力，但无论从代码的维护，还是重用来看，这都是非常值得的。

下面我们开始将声音播放代码制作成jQuery插件。由于这个插件需要与jQuery库协同工作，因此在HTML文档中需要同时调用它们。在前面代码的基础上，做如下修改，然后保存到`jquery.banner_sound.js`文件中。

<code>;(function(\$) {</code>	JavaScript自运行函数
<code>\$fn.banner_sound = function(audio_src) {</code>	使用jQuery.fn原型编号 <code>banner_sound</code> 函数
<code>var banner_audio;</code>	保存音源的变量 <code>banner_audio</code>
<code>return this.each(function() {</code>	返回 <code>this</code> 对象
<code>\$this)</code>	将使用 <code>banner_sound</code> 插件的对象包装成jQuery对象
<code>.bind('mouseover focusin', function() {</code>	
<code>banner_audio = new Audio(audio_src);</code> 传入音频对象的路径	
<code>banner_audio.play();</code>	

```
        });
        .bind('mouseout focusout', function() {
            banner_audio.pause();
        });
    });
})(jQuery);
```

在内部使用\$代替jQuery

修改 banner_design.html 文件，添加脚本调用代码，如下所示。

```
调用jQuery      <link href="css/banner.css" rel="stylesheet" />
→<script src="js/jquery.min.js"></script>
<script src="js/jquery.banner.js"></script>← 调用jQuery.banner_sound插件
<script src="text/javascript">
    (function($) {
        $(function() {
            $('.banner').banner_sound('media/banner_sound.webm');
        });
    })(jQuery);
</script>          放入banner的类名
</head>          设置音源路径
```

在制作成 jQuery 插件后，后期修改维护时会非常方便。在调用该插件时，首先在网页中调用 jQuery 与插件，而后在 \$() 中使用 CSS 选择器形式指定元素，再在 .banner_sound() 中指定声音路径。代码修改完成后保存，在浏览器中打开网页，查看效果，可以看到当鼠标指针移动到横幅广告上或按 Tab 键使横幅广告获得焦点时，便开始播放指定的声音文件。

浏览器兼容性检查



到现在为止，我们已经制作好动态横幅广告了，它的效果非常炫，就像使用 Flash 制作的一样。但令人遗憾的是它并不是能在所有的浏览器上正常运行，这里指是旧版本的浏览器。像 IE6，它随 Windows XP 一起发布于 2001 年，至今已有十多个年头，它就像一台年代久远的黑白电视，你想在黑白电视上观看高画质节目、彩色节目或点播，都是不可能的。我们设计的动态横幅广告采用了最新的 HTML5/CSS3/jQuery 技术，这些新技术在老的电子产品上是不可能被支持的，换言之，这样的动态横幅广告在旧电子产品上无法正常工作。这是理所当然的事。



有人会问：难道我们对这些旧版本的浏览器不闻不问吗？当然不是，我们要根据这些旧浏览器所支持的功能重新设计制作横幅广告。在重新设计横幅广告时，可能要放弃横幅广告的动态效果。虽然向横幅广告添加这些动态效果会使横幅广告更酷更炫，但对于横幅广告来说，这些动画效果不是它的主要功能。横幅广告的主要功能是向观看者传递广告信息，并且当用户点击它时链接到指定的网站中。如果横幅广告要使用在旧版本的浏览器中，那么制作时首先要保证横幅广告的基本功能能够得到充分实现。

事实上，由于 IE 6/7/8 这些旧版本的浏览器不支持新出现的技术，在它们中使用网页标准平台技术实现动画与音频播放是非常困难的，就像在一台旧的 MP3 播放器中播放高画质视频一样几乎是不可能的。即使能够实现，也需要动用大量的人力、物力资源，这是非常低效的行为，并且旧版本的浏览器也会逐渐被升级淘汰，很快就会消失不见（微软公司从 2012 年 2 月 13 日开始对 IE 浏览器进行强制升级）。

那么有哪些替代方案可以使横幅广告在旧版浏览器中正常发挥功能呢？第一种方案是使用 Adobe 公司的 Flash 技术，使用这种技术能够轻松实现动画与声音效果，并且在旧版浏览器中也能正常工作。但是使用这种技术的前提是浏览器中必须先安装好 Flash 插件，而像苹果的 iPhone/iPad 等设备却不支持 Flash，这是个难办的问题。第二种方案是使用图像，图像在传统的 Banner 设计中经常被使用。使用图像技术，不仅能在旧版本浏览器中轻松实现 Banner 的基本功能，也能帮助我们解决头疼的浏览器兼容性问题，但是使用图像只能实现简单的动画，复杂的动画、声音效果都无法实现。

究竟使用哪种方法，最终取决于您自己。选择时，请先与您的团队讨论协商，根据实际制作要求，选择合适的方案即可。就笔者而言，笔者一般会选择第二种方案，在旧浏览器中使用图像方式来设计 Banner，这种方式能够很容易地帮助我们实现 Banner 的基本功能。当然，在具体实施选择的方案之前，我们要先判断浏览器的版本，然后再针对浏览器的版本修改相应代码，制作需要的图像。如果用户想看动态 Banner 效果，那他们要升级自己使用的浏览器版本。到底是使用旧浏览器观看图像形式的 Banner，还是升级到新版本浏览器观看动态 Banner 效果，完全由用户自己来选择。

4 章

设计气泡悬浮框

在网页设计中，气泡悬浮框常常用于在页面中为某些对象显示提示信息，恰当地使用气泡悬浮框能够使网页布局更加精美，使网页看上去更漂亮、美观。一般而言，替换文本使用 alt 属性来呈现，说明性文本通过 title 属性来表现，这两个属性是 HTML 默认提供的功能，在网页设计中使用它们，容易引起用户的反感。在这种情况下，使用气泡悬浮框能够产生不一样的视觉体验。在这一章，我们将学习使用气泡悬浮框来设计网页中的替代文本与说明文本。

设计效果预览

设计网页时使用气泡悬浮框，不仅能保持网页页面整洁干净，还能向用户提供大量有用的信息。设计气泡悬浮框时，我们使用 HTML5 的 `<a>` 元素来实现。在气泡悬浮框中，我们插入图片的标题与说明性文字。在鼠标指针移动到图片上之前，仅显示为图片，而当鼠标指针移动到图片上或按 Tab 键使图片获得焦点后，气泡悬浮框显现出来并显示图片的标题与说明文本。



鼠标指针移动到图片上之前



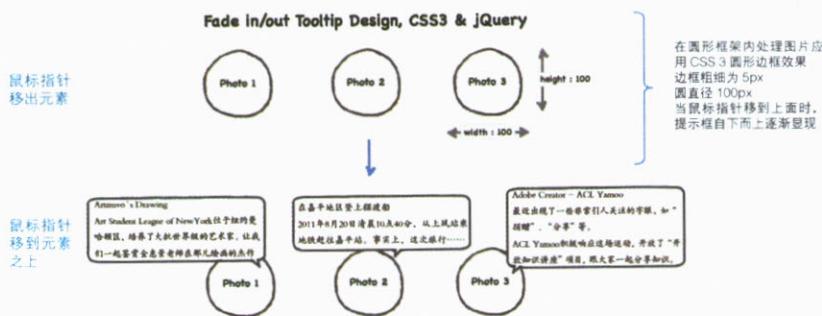
鼠标指针移动到图片上之后

在设计气泡悬浮框时需要注意以下几点：第一，为了让气泡悬浮框显现得更加生动自然，我们采用渐入 / 渐出效果进行设计，即实现气泡悬浮框慢慢显现，再慢慢消失的效果；第二，关于气泡悬浮框的形状，我们采用对角线圆角矩形形状，与四角圆角矩形相比，这种形状显得更加生活活泼，没那么呆板单调；第三，我们还要为气泡悬浮框添加上小尾巴，将其与指定的图片联系起来。所有这些效果，我们均使用 CSS3 来实现，请认真学习 CSS3 有关的内容。

从本章开始，我们将正式学习有关 HTML5、CSS3 和 jQuery 的内容，如果这方面的知识缺乏，请返回到第 2 章中学习相关内容。

设计网页布局

使用 Balsamiq Mockups 工具将头脑中的网页布局描绘出来，如下所示。



显示图片时，我们不需要显示整张图片，而只显示图片上的一个圆形区域即可，圆形区域之外的部位要遮盖起来。虽然我们也可以使用图像处理工具将图片剪裁成圆形，但这里我们不使用这种方法，而使用 CSS 3 新添加的 border-radius 属性来实现。



CSS3 新添加的 border-radius 属性允许我们使用 CSS 创建圆角而不需要使用图片或者 JavaScript，轻松实现圆角边框效果，大大简化了前端设计人员开发圆角的难度。border-radius 是 CSS3 新添加的属性，除了 IE 外，目前 Firefox、Safari、Chrome 和 Opera 均支持该属性，其中 Safari、Chrome 和 Opera 是支持得最好的，依照了 W3C 的标准，仅仅使用 border-radius 属性，就可以实现效果，而无需加相应的前缀。相信随着浏览器版本的不断升级，对这个属性的支持会越来越好。

编写 HTML5 代码

下载例题文件，解压缩后，在 \ex\ex_04\Begin 文件夹中找到 tooltip-design.html 文件，然后使用网页编辑器打开它。打开文档后，可以发现 HTML 文档中仅由 HTML5 DTD 与基本语句构成。文档的注释部分用于检测网页浏览器的版本，在 `<head>` 部分调用了 3 个外部文件，分别为 css/tooltip.css、js/jquery.min.js 和 js 和 tooltip.js，我们将在 `<body>` 部分编写网页的 HTML 代码。



The screenshot shows the Dreamweaver interface with the 'tooltip_design.html' file open in the center workspace. The code editor displays the following HTML5 code:

```
<!DOCTYPE html>
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->
<!--[if !IE]><!--><html lang="zh" class="no-js modern"><!--><![endif]-->
<head>
    <meta charset="utf-8" />
    <title>CSS3 Tooltip Design - 淡入/淡出提示工具设计</title>
    <link rel="stylesheet" href="css/tooltip.css" />
    <script src="js/jquery.min.js"></script>
    <script src="js/tooltip.js"></script>
</head>
<body>
</body>
</html>
```

首先使用 `<h1>` 元素添加标题 “Fade in/out Tooltip Design, CSS3 & jQuery” 。由于在本章中，我们将学习英语网页字体的使用方法，所以添加标题时我们使用了英文文本，代码如下。

`<h1> Fade in/out Tooltip Design, CSS3 & jQuery </h1>`

小知识 字符实体

如果你足够细心，就会发现输入标题时我们在 “`&`” 之后添加了 “`amp;`”，这类字符我们称为“字符实体”。在 HTML 中有些字符对 HTML 来说是具有特殊意义的，所以这些字符是不允许直接在文本中使用的。要在 HTML 中显示这些字符，就必须使用字符实体。比如 “`<`” 这个字符，浏览器在读取 HTML 文档时会将其认作 HTML 标签，要在文本中显示这个字符，就必须使用字符实体 “`<`”。在 HTML 文档中，常用的字符实体如下表所示。

显示	实体名称	实体编号	说明
 	 		空格 (Non-breaking space)
&	&	&	and (Ampersand)
"	"	"	引号 (Quotation mark)
"	“	“	双引号开始 (Opening Double Quotes)
"	”	”	双引号结束 (Closing Double Quotes)
'	‘	‘	单引号开始 (Opening Single Quote Mark)
'	’	’	单引号结束 (Closing Single Quote Mark)
·	·	·	中间点 (Medium List Dot)
°	°	°	度数 (Degree symbol)
<	<	<	小于 (Less than)
>	>	>	大于 (Greater than)
©	©	©	著作权符号 (Copyright symbol)
®	®	®	注册商标符号 (Registered symbol)

除上表列出的之外，还有许多其他字符实体，更多内容，请访问 <http://entitycode.com>。

接下来，使用 `<a>` 标签编写提示文本。`<a>` 标签有两个属性，分别为 `class` 与 `href`，我们将 `class` 属性设置为 `tooltip`，将 `href` 属性设置为要访问的网页路径。在 `<a>` 标签内，插入一个 `<div>` 标签，用它代表悬浮框，设置其 `class` 属性为 `tooltip-box`。在 `<div>` 标签内，分别使用 `<h4>` 表示标题，使用 `<p>` 标签表示提示文本，并且将它们的 `class` 分别设置为 `tooltip-title` 与 `tooltip-desc` 进行标识。

```
<a class="tooltip" href="http://yamoo9.com/?p=699">
<div class="tooltip-box">
    <h4 class="tooltip-title">在嘉平地区登上摆渡船</h4>
    <p class="tooltip-desc">2011年8月20日清晨10点40分，从上凤站乘地铁赶往嘉平站。
        事实上，这次旅行……</p>
</div>
</a>
```

下面我们再添加另外两个悬浮框，每个悬浮框使用 `<div>` 标签表示，且被包含在 `<a>` 标签中。`<a>` 标签又被包含在 `` 之中，最后 3 个悬浮框用 `` 标签包围起来，作为一个完整的导航列表。`` 标签的 `class` 属性被指定为 `gallery-nav`。最后，向各个 `<a>` 标签的 `class` 属性值上分别添加 `photo1`、`photo2` 和 `photo3`，以方便在 CSS 中为各个元素添加不同的背景图片。

```
<ul class="gallery-nav">
</li>
<a class="tooltip photo1" href="http://yamoo9.com/?p=699">
<div class="tooltip-box">
<h4 class="tooltip-title">在嘉平地区登上摆渡船。</h4>
<p class="tooltip-desc">2011年8月20日清晨10点40分，从上夙站乘地铁赶往嘉平站。事实上，这次旅行……</p>
</div>
</a>
</li>
</li>
<a class="tooltip photo2" href="http://yamoo9.com/?p=699">
<div class="tooltip-box">
<h4 class="tooltip-title">Artnuovo's drawing</h4>
<p class="tooltip-desc">Art Student League of NewYork位于纽约曼哈顿区，培养了大批世界级的艺术家。让我们一起鉴赏金惠景老师在那儿绘画的杰作！</p>
</div>
</a>
</li>
</li>
<a class="tooltip photo3" href="http://yamoo9.com/interview/">
<div class="tooltip-box">
<h4 class="tooltip-title">Adobe Creator - ACL Yamoo</h4>
<p class="tooltip-desc">最近出现了一些非常引人关注的字眼，如“赠”、“分享”等。ACL Yamoo积极响应这场运动，开放了“开放知识讲座”项目，跟大家一起分享知识。</p>
</div>
</a>
</li>
</ul>
```

代码编写完成后保存，在浏览器中查看页面效果，如下图所示。

Fade in/out Tooltip Design, CSS3 & jQuery

在嘉平地区登上摆渡船

2011年 8月 20日 清晨 10点 40分 从上夙站乘地铁赶往嘉平站。事实上，这次旅行…

Artnuovo's drawing

Art Student League of NewYork位于纽约曼哈顿区，培养了大量世界级的艺术家。让我们一起鉴赏金惠景老师在那儿绘画的杰作！

Adobe Creator - ACL Yamoo

最近出现了一些非常引人关注的字眼，如“捐赠”、“分享”等。ACL Yamoo积极响应这场运动，开放了“开放知识讲座”项目，跟大家一起分享知识。

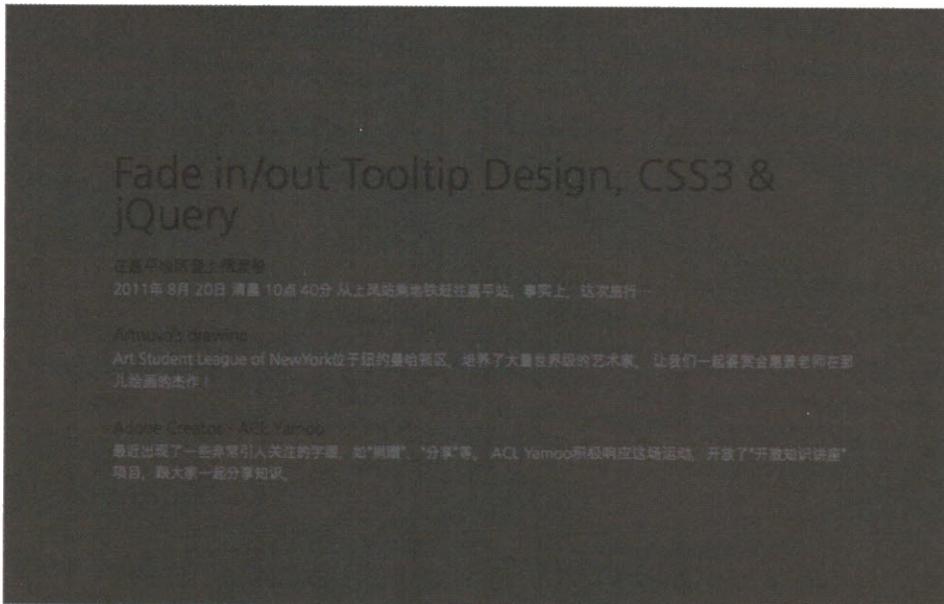
编写 CSS 3 样式表

控制 body 样式

进入例题源文件 \ex_03\begin\css 文件夹中，双击 tooltip.css 文件，将其打开。首先，我们为 <body> 设置背景，在 CSS 中使用 body 选择器，设置内部补丁、背景颜色与背景图片。在这里，背景图片我们使用 images 文件夹中的 bg-tile.jpg 图片。

```
body {  
    padding: 150px;           设置内部空白  
    background: #2b2b2b url(..../images/bg-tile.jpg);  设置背景色，背景图片  
}
```

保存代码，在浏览器中查看效果，如下图所示。



设置标题字体样式

下面我们开始为标题（<h1> 元素）设置英文字体。首先打开浏览器，在地址栏中输入 <http://www.google.com/webfonts>，转到 Google 网页字体服务页面中。

The screenshot shows the Google Web Fonts interface. On the left, there's a search bar with 'lato' typed in, and a sidebar with filters for thickness (sliding from thin to thick), slant, and width. The main area displays three font families: 'Kavoon' (1 style by Viktoria Grabowska), 'Roboto Condensed' (6 styles by Christian Robertson), and 'Frutur' (1 style by Viktoria Grabowska). Each entry includes a preview of the text 'Grumpy wizards make toxic brew for the evil Queen and Jack.' in a bold, black font.

在左侧 Search 框中，输入“lato”。在 Thickness 中，拖动 thin 滑块到最左侧，在页面右侧显示出搜到的结果：Lato Ultra-Light。单击字体底部的“Quick-use”链接，转到 Quick Use 页面中。

This screenshot shows the same search results as above, but with a hand cursor hovering over the 'Quick-use' button for the Lato font. This highlights the button, indicating it's the next step to take.

浏览器跳转到 Quick Use 页面后，向下拖动滚动条，直到显示出 [3. Add this code to your website:] 区域，该区域提供了 3 种使用该字体的方法，第 1 种使用 `<link>` 标签，第 2 种使用 `@import` 命令，第 3 种使用 JavaScript 脚本。在本实例中，我们将在 CSS 中使用 `@import` 命令调用网页字体。单击 `@import` 选项卡，拖选代码，而后按 (`Ctrl`+`C`) 组合键进行复制。

This screenshot shows the 'Quick Use' page for the Lato font. The 'Import' tab is selected. It contains three sections: '3. Add this code to your website:' with a code editor containing `@import url('http://fonts.googleapis.com/css?family=Lato:100');`, 'Instructions' (which says to embed the code into the `<head>` of an HTML document), and 'Example'. Below these, there's another section '4. Integrate the fonts into your CSS:' with instructions and an example.

按(Ctrl+V)组合键，将复制的代码粘贴到css/tooltip.css文件中。

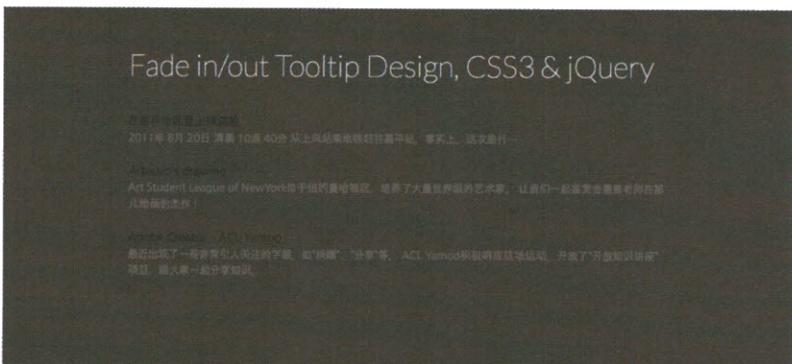


```
@charset "utf-8";
@import url(http://fonts.googleapis.com/css?family=Lato:100);
/* tooltip.css - 100% Tooltip设计样式，2013 © yamodog.com */
```

接着，向标题指定下外边框、字体样式与字体颜色。

```
h1 {
    margin-bottom: 40px;          设置h1的下外边距为40px
    font-family: 'Lato', Sans-Serif;  应用谷歌Lato字体
    color: #fff;                设置字体颜色为白色
}
```

保存代码，在浏览器中查看页面效果，可以发现标题已经被应用了Lato字体，如下图所示。



小知识

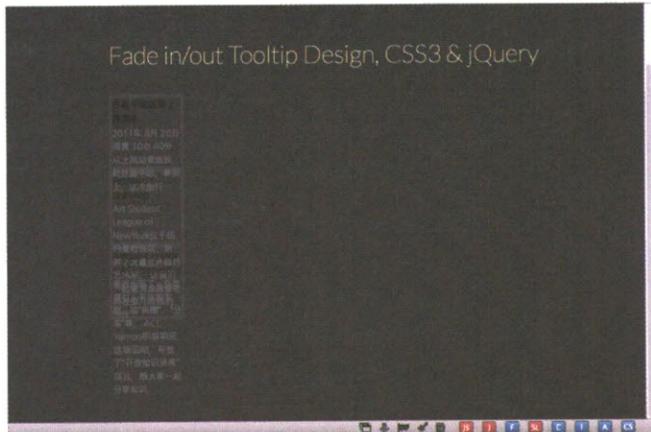
到目前为止，谷歌网页字体服务已经支持560多种字体，并且支持的字体会越来越多，增长速度会越来越快。由于使用IE9在本地环境中字体无法正常显示，因此建议在Web服务器环境下进行测试。

控制图片样式

下面我们开始控制a.tooltip的样式，对其进行修饰。首先将内联元素a转换成块元素，而后设置宽度与高度，设置背景色为白色，边框粗细为15px，颜色为深灰，形状为直线。

```
a.tooltip {
    display: block;          设置a元素为块元素
    width: 100px;            Banner宽度为100px
    height: 100px;           Banner高度为100px
    border: 5px solid #4b4b4b;  设置Banner边框为5px，实线，暗灰色
}
```

保存代码，在浏览器中查看网页，可以看到3个方形框呈垂直方式排列，如右图所示。



接下来，我们将3个方形框按水平方向排列。设置时，先向li元素指定float:left;，而后向每个li元素指定右外边距为100px，使3个方形框之间相隔一定的距离。

```
.gallery-nav li {  
    float: left;  
    margin-right: 100px;  
}
```

选择.gallery-nav内的li元素
设置向左浮动
设置右外边距为100px

保存代码，在浏览器中查看页面，可以发现3个方形框已经按水平方向排列了，如右图所示。



→ 小知识 ← float 属性

float属性定义元素在哪个方向浮动，是确定网页布局时使用的属性。起初，float属性不是一个与布局相关的属性，它总是被应用于图像，使文本围绕在图像的周围。过去在设计网页时，常常使用元素的align属性来定义图像相对于周围元素的水平和垂直对齐方式，但在现在的网页设计中我们常常会使用CSS取代HTML中的表现元素，在CSS中就使用float属性来取代HTML中的align属性。

我们可以使用float属性控制围绕在图片周围的文本，如下图所示。float属性不仅可以用于img元素，与align属性不同，它可用于所有构成文档的元素，比如div与p元素等。事实上，在CSS中，任何元素都可以浮动，浮动元素会生成一个块级框，而不论它本身是何种元素。



在 CSS 中 float 属性用于改变块元素对象的默认显示方式。当块元素对象设置了 float 属性后，它将不再独占一行，而是可以浮动到左侧或右侧，直到浮动的框外缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，因此文档的普通流中的块框表现得就像浮动框不存在一样。这有点与 absolute（绝对定位）属性类似，但 float 与 absolute（绝对定位）主要有以下 3 点不同。

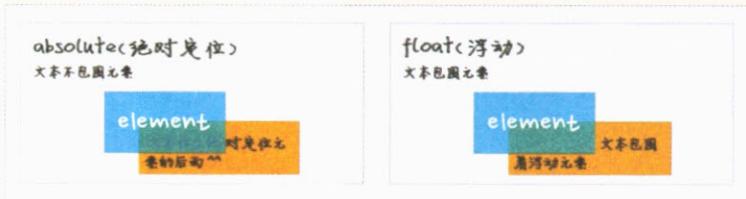
float 与 absolute 属性设定位置的方法不同！

absolute（绝对定位）元素的位置相对于最近的已定位的祖先元素，若元素没有已定位的祖先元素，那么它的位置相对于最初的父亲块。简言之，absolute 属性以父元素框的 4 个顶点为基准进行定位。而 float 属性定位时则根据 left 与 right 属性值，以父元素框的左上与右上为基准进行定位，它不能以左下或右下为基准进行定位。



采用 absolute 属性定位的元素不能被文本所包围，而后采用 float 属性定位的元素可以被文本包围！

假如有两个元素 A 与 B，它们竖直排列在一起。在为 A 元素指定了 absolute 属性之后，其下的 B 元素会占据 A 元素原来的位置。如果为 A 元素指定了 float（浮动）属性，B 元素也会占据 A 元素的位置。当 B 元素是文本时，为 A 元素设置了 absolute 属性后，B 元素会被 A 元素覆盖到下面。若为 A 元素设置了 float（浮动）属性，则 B 元素为围绕着 A 元素。



float 的影响可控制，而 absolute 影响不可控制！

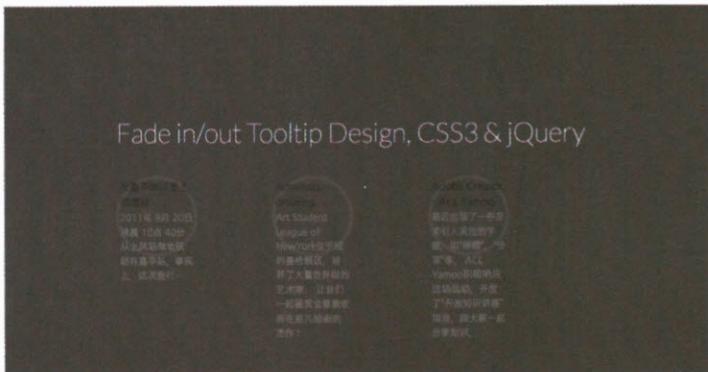
由于 float 与 absolute 都脱离了文档普通流，因此其下的元素会进行相应的浮动或移动，也就是说 float 与 absolute 会影响到其下方的元素，但有所不同。absolute 属性是布局属性，使用它时没有一种有效的方法使之与其下方的元素不重合在一起。虽然我们可以根据应用了 absolute 属性的元素的高度，设置其后元素的上外边距 (margin-top)，但是当应用 absolute 属性的元素的高度发生改变时，两个元素会再次重叠在一起。相反，若为一个元素指定了 float 属性，当向其后的元素应用 clear 属性后，其后的元素就不再受影响了。因为这个原因，在确定网页布局时，更多的是使用 float 属性，而不是 absolute 属性。



接下来，我们将方形框转换为圆形框，在`a.tooltip`中添加如下代码。在CSS3中新添加了一个新属性`border-radius`，使用这个属性可以帮助设计者轻松实现圆角边框效果。若想使用该属性实现圆形框效果，则应将圆的半径设置为边长的一半。当前框的边长为100px，边框粗细为10px，总长为两者之和，即110px，所以我们将其值设置为55px，这样就能将方形转换成圆形了。在实际使用`border-radius`属性时，要根据不同的浏览器添加相应的前缀（关于添加前缀的内容，请参考第3章第81页中的相关内容）。

```
a.tooltip {  
    border-radius: 55px;          设置圆角为55px(110px/2)  
    -webkit-border-radius: 55px;  添加不同的浏览器前缀  
    -khtml-border-radius: 55px;  
    -moz-border-radius: 55px;  
}  
/*
```

保存代码，在浏览器中查看网页，可以看到原来的方形框已经变为圆形框了，如下图所示。



接下来，我们向各个圆形框插入不同图片，图片的路径通过`background-image`属性指定。

```
a.tooltip.photo1 {  
    background-image: url(..../images/sussjini-bbo.jpg);  
}  
/*
```

`a`元素中`class`值为`tooltip`与`photo1`的元素设置背景图片

使用相同方法，向`a.tooltip.photo2`、`a.tooltip.photo3`插入背景图片。

```
a.tooltip.photo2 {  
    background-image: url(..../images/khk-artwork.png);  
}  
a.tooltip.photo3 {  
    background-image: url(..../images/interview-yamoo9.png);  
}
```

然后，对背景图片进一步进行修饰，设置`.tooltip-box`的不透明度(`Opacity`)为0，将背景图片的位置(`background-position`)设置到中间(`center`)，最后把背景图片尺寸(`background-size`)指定为`cover`，保持图片本身的宽高比例，将图片缩放到正好完全覆盖圆形区域。

```
a.tooltip {  
    background: #fff no-repeat center;          设置背景图片为不重复，居中对齐  
    background-size: cover;                    使用背景图片填充圆形框  
}  
a.tooltip .tooltip-box {  
    opacity: 0;                                选择.tooltip的.tooltip-box  
                                                设置不透明度为0，隐藏起来  
}
```

保持代码，在浏览器中查看页面，可以看到相应的图片被插入到圆形框之中。



↔ 小知识 ↔

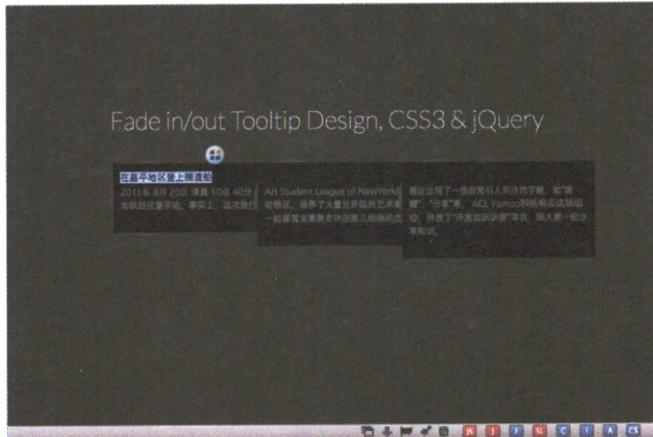
在这里，我们把图片设置到中间位置上，根据所使用的图片的不同，要随时调整位置。当然，也可以根据所使用的图片，调整圆形框的尺寸，使图片与圆形框完全相配。

控制气泡悬浮框 1：基本样式

下面我们开始控制气泡悬浮框(`.tooltip-box`)的样式。首先将`.tooltip-box`的不透明度从0设置为1，使其在画面上显示出来。而后根据设计意图调整宽度大小，设置背景色为深色，添加合适大小的内部补丁。

```
a.tooltip .tooltip-box {  
    opacity: 1;                                设置不透明度为1，显示出来  
    width: 20em;                               设置宽度为字高的20倍  
    padding: .8em;                             设置内部空间为字高的0.8倍  
    background: #111;                           设置背景色为深色  
}
```

保存代码，在浏览器中查看页面，可以看到标题与背景色变得不一样了。同时悬浮框的底部空白比左侧与右侧大多了。应当减小一下它的尺寸，并且字体样式、字体大小、字体颜色也要调整一下。由于标题的 class 为 tooltip-title，内容为 tooltip-desc，因此在 CSS 中，我们分别使用 .tooltip-title 与 .tooltip-desc 选择器来控制它们。



```
a.tooltip .tooltip-title {          选择提示标题.tooltip-title  
    color: #fff;                      设置字体颜色为白色  
}  
  
a.tooltip .tooltip-desc {           选择提示内容.tooltip-desc  
    margin-bottom: 0;                  删除下外边距为0  
    font-size: 11px;                  设置字号为11px  
    text-align: justify;             设置文本对齐方式为两端对齐  
    color: #bcbcbc;                 设置字体颜色为亮灰色  
}
```

保存代码，在浏览器中查看页面，可以看到标题、底部空白和字体形状都改变了，如下图所示。



控制气泡悬浮框 2：位置

在气泡悬浮框的形状设计完成后，接下来，我们为它们确定位置。在定位时，首先指定 tooltip-box 为绝对定位，以 .tooltip 基准确定位置（关于指定位置的方法，请参考第 3 章第 74 页中的相关内容）。

```
a.tooltip {
    position: relative;          设置为相对定位，它是.tooltip-box位置的基准线
}
a.tooltip .tooltip-box {
    position: absolute;          设置绝对定位
    bottom: 100px;               以底端为基准上移100px
}
```



接着，将.tooltip-box 的起始位置设置到图片的正中间，设置 left 属性值为 50%。

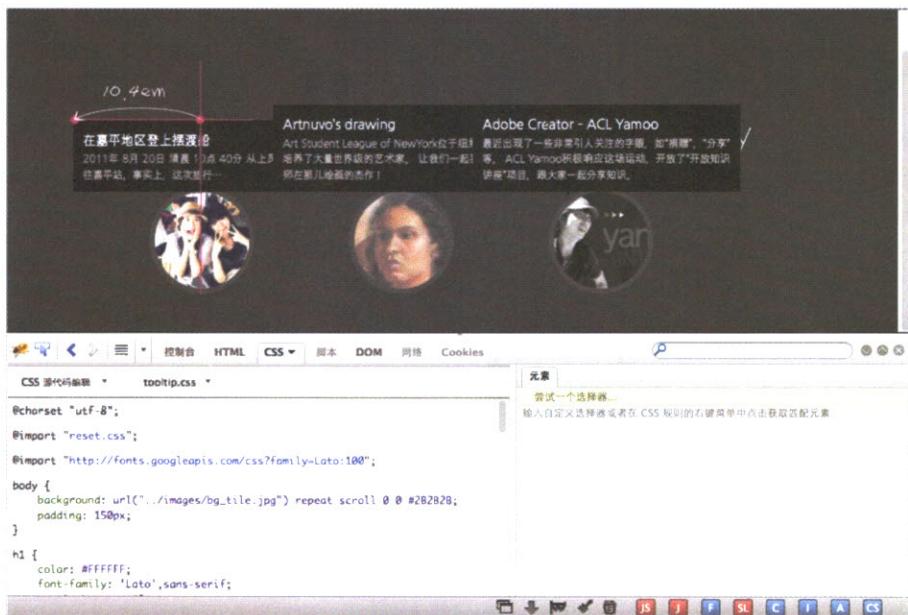
```
a.tooltip .tooltip-box {
    left: 50%;                  以左侧为基准，将悬浮提示向右移50%
}
```

.tooltip-box 以.tooltip 为基准被设置在沿水平方向 50% 的位置上。但是由于.tooltip-box 的左上顶点是基准点，因此并不是设置在中央位置，而是.tooltip-box的左上顶点被设置到中间位置上。



若想把.tooltip-box 设置到中间位置上，该怎么办呢？方法很简单，只要把.tooltip-box 向左移动一半就可以了，即将.tooltip-box 的 margin-left 设置为 $(\text{.tooltip-box 的 width} + \text{.tooltip-box 的 padding}) / 2$ ）。在 CSS 的盒子模型中，width 是指内容框的宽度，必须将 padding 数值包含到内容框的宽度中，才能计算出.tooltip-box 的实际宽度。由于需要向左移动，因此添加了负号。关于 CSS 盒子模型的知识，请参考第 3 章第 72 页中的相关内容。

```
a.tooltip .tooltip-box {
    margin-left: -10.4em;        设置左外空白为负数10.4em((20em+0.8em)/2)
}
```



控制气泡悬浮框 3：添加圆角与尾巴

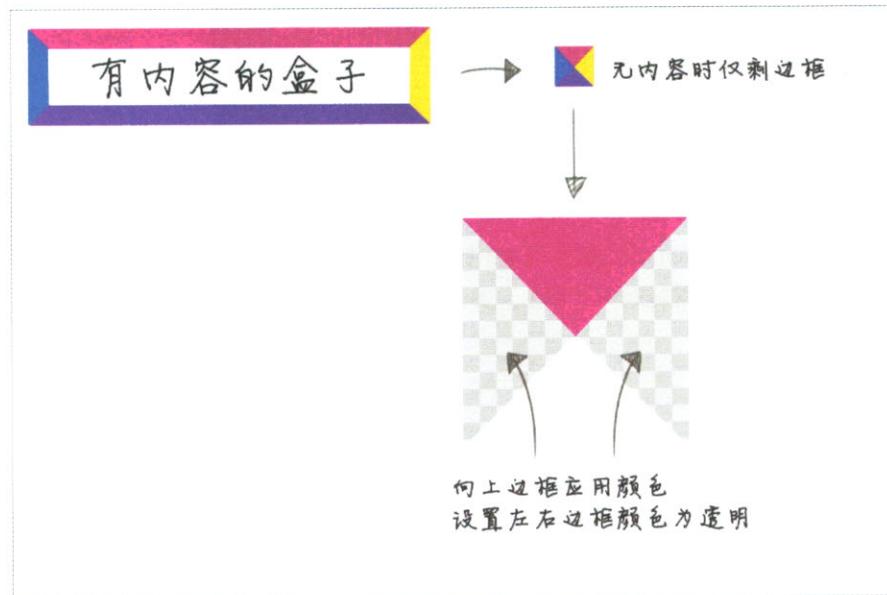
下面我们开始将`.tooltip-box`的左上与右下的直角变成圆角，在转换过程中，我们将用到CSS3的`border-radius`属性。`border-radius`是CSS3新添加一个的属性，使用这个属性能够帮助我们轻松实现圆角边框效果。该属性有4个值，这4个值是按照上左、上右、下右、下左次序来设置的。若是只设置两个值，则第一个值为上左与下右，第二个值为上右与下左。在这里，我们将`border-radius`的值设置为`15px`与`0px`。

```
a.tooltip .tooltip-box {
    border-radius: 15px 0px; // 设置左上角与右下角为15px，右上角与左下角为0px
}
```

保存代码，在浏览器中查看网页，可以看到悬浮框的左上角与右下角变成圆角了，设计感觉变得更加浓厚了。



接下来，为各个悬浮框添加小尾巴。制作带有小尾巴的悬浮框时，我们常常使用 Photoshop 等图像处理软件来制作。其实在网页制作中这种带有小尾巴的悬浮框，使用 CSS3 也非常容易制作。首先我们需要创建一个三角形，请看下面的图片，在“包含内容的内容框”的周围，上下左右都被框线包围着。当内容框中包含的内容清除时，其水平与竖直尺寸变为 0，仅剩下上下左右 4 个边框。各个边框彼此重叠，呈对角线分割开。利用这一特性，仅向上边框设置粗细与颜色，左侧与右侧的边框颜色设置为透明，这样就形成了三角形，如下图所示。

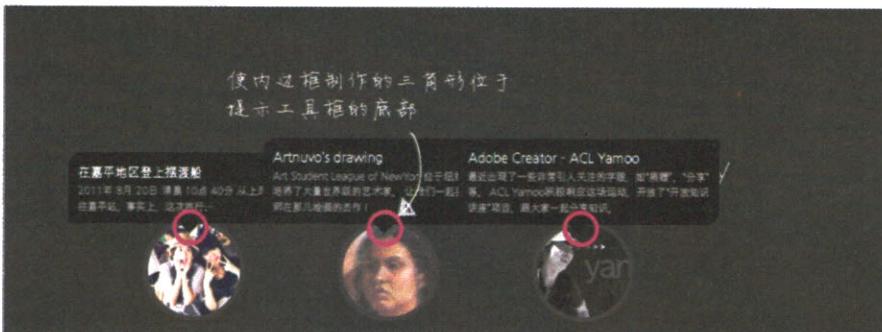


下面我们将这一设想用代码实现。首先使用虚拟元素 ::before 与 content 属性，创建一个空内容，再设置其边框，形成三角形。由于三角形的尖角要朝下，因此我们需要设置上边框、左边框与右边框。

```
a.tooltip .tooltip-box : before {
    content: '';
    position: absolute;
    bottom: -10px;
    left: 120px;
    border-top: 10px solid #111;
    border-left: 10px solid transparent;
    border-right: 10px solid transparent;
}
```

在.tooltip-box元素之前添加内容
空内容
绝对定位
以下边为基准，向外移动10px
从左侧向右侧移动120px
设置上边框为10px，实线，.tooltip-box颜色与背景色相同
设置左边框为10px，实线，透明
设置右边框为10px，实线，透明

保存代码，在浏览器中查看网页，可以发现在悬浮框的底部出现了三角形小尾巴，如下图所示。



控制气泡悬浮框 4：制作 Transition 动画

下面我们将使用 CSS3 的 transition 属性制作 Transition 动画，当鼠标指针移动到图片上，或者按 Tab 键使图片获得焦点时，悬浮框轻轻向下滑出。关于 transition 属性的更多内容，请参考第 3 章第 80 页中的相关内容。

```
a.tooltip:hover .tooltip-box,  
a.tooltip:focus .tooltip-box {  
    opacity: 1;  
    bottom: 90px;  
}
```

当鼠标指针移动到a.tooltip上时，选择.tooltip-box
当a.tooltip获得焦点时，选择.tooltip-box
设置不透明度为1，显示出来
从下往上移动90px



向 .tooltip-box 设置 transition 属性。在鼠标指针移动到图片上之前，悬浮框是隐藏的，所以我们要将其不透明度 (Opacity) 设置为 0。

```
a.tooltip .tooltip-box {  
    opacity: 0;  
    transition: all .4s ease-in .3s;  
}
```

选择.tooltip-box元素
在鼠标指针移上之前隐藏
所以属性在0.3秒之后平滑过渡0.4秒



当鼠标指针移动到图片上，或按 Tab 键使图片获得焦点时，圆形边框要变成白色，我们要编写代码实现这种效果。我们还要设置 transition 属性，使画面能够平滑切换。前面设置的 transition 值同样适用于 .tooltip。

```
a.tooltip:hover,          鼠标指针放到a.tooltip上时  
a.tooltip:focus {         当a.tooltip获得焦点时  
    border-color: #fff;    把邊框顏色為白色  
}
```

至此，气泡悬浮框效果全部制作完成，全部 CSS 控制代码如下所示。

```
@charset "utf-8";  
@import "reset.css";  
@import url(http://fonts.googleapis.com/css?family=Lato:100);  
  
/* tooltip.css - ToolTip设计样式，2013 © yamoo9.com */  
  
/* //////////////////////////////////////////////////////////////////*/  
/* =基本样式 */  
/* //////////////////////////////////////////////////////////////////*/  
  
body {  
    padding: 150px;  
    background: #2b2b2b url(..../images/bg_tile.jpg);  
}  
  
h1 {  
    margin-bottom: 40px;  
    font-family: 'Lato', sans-serif;  
    color: #fff;  
}
```

```
/* //////////////////////////////
 =.gallery-nav
//////////////////////////// */
.gallery-nav li {
    float: left;
    margin-right: 100px;
}

/* //////////////////////////////
 =.tooltip
//////////////////////////// */
.a.tooltip {
    position: relative;
    display: block;
    width: 100px;
    height: 100px;
    border: 5px solid #4b4b4b;
    background: #fff no-repeat center;
    background-size: cover;
    /* CSS3 Border-radius */
    -webkit-border-radius: 55px;
    -khtml-border-radius: 55px;
    -moz-border-radius: 55px;
    border-radius: 55px;
    /* CSS3 Transition */
    -webkit-transition: all .4s ease-in .3s;
    -moz-transition: all .4s ease-in .3s;
    -o-transition: all .4s ease-in .3s;
    -ms-transition: all .4s ease-in .3s;
    transition: all .4s ease-in .3s;
}
.a.tooltip:hover,
.a.tooltip:focus {
    border-color: #fff;
}
.a.tooltip .tooltip-box {
    opacity: 0;
    position: absolute;
    left: 50%;
    bottom: 100px;
    width: 20em;
    margin-left: -10.4em;
    padding: .8em;
    background: #111;
```

```

/* CSS3 Border-radius */
-webkit-border-radius: 15px 0px;
-khtml-border-radius: 15px 0px;
-moz-border-radius: 15px 0px;
border-radius: 15px 0px;
/* CSS3 Transition */
-webkit-transition: all .4s ease-in .3s;
-moz-transition: all .4s ease-in .3s;
-o-transition: all .4s ease-in .3s;
-ms-transition: all .4s ease-in .3s;
transition: all .4s ease-in .3s;
}
a.tooltip:hover .tooltip-box,
a.tooltip:focus .tooltip-box {
    opacity: 1;
    bottom: 90px;
}
a.tooltip .tooltip-box:before {
    content: '';
    position: absolute;
    bottom: -10px;
    left: 120px;
    border-top: 10px solid #111;
    border-left: 10px solid transparent;
    border-right: 10px solid transparent;
}
a.tooltip .tooltip-title {
    color: #fff;
}
a.tooltip .tooltip-desc {
    margin-bottom: 0;
    font-size: 11px;
    text-align: justify;
    color: #bcbcbc;
}

/* 照片 */
a.tooltip.photo1 {
    background-image: url(..../images/sussjini-bbo.jpg);
}
a.tooltip.photo2 {
    background-image: url(..../images/khk-artwork.png);
}
a.tooltip.photo3 {
    background-image: url(..../images/interview-yamoo9.png);
}

/* /////////////////////////////////
=Global Classes
//////////////////////////// */
.clearfix:after {
    content: "";
    display: block;
    clear: both;
}
.ie6 .clearfix { height: 1px; } /* IE6 */
.ie7 .clearfix { min-height: 1px; } /* IE7 */

```

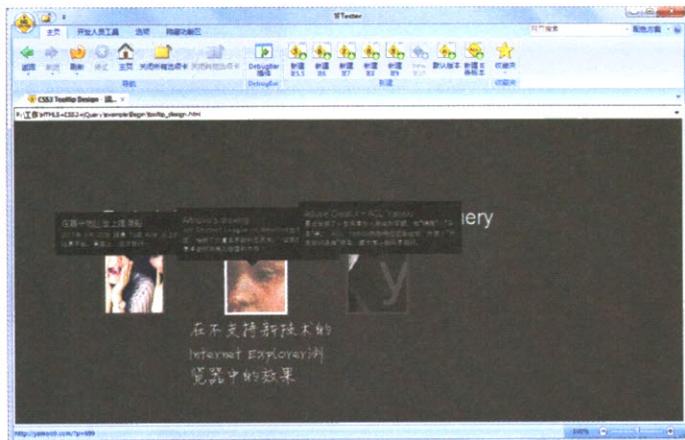
在网页设计中，即使不使用 Photoshop 或 Flash 等软件，凭借这些新技术，也能十分容易地制作出酷炫的动画效果。随着不断学习新的技术，一些好的创意也会随时而生。当这些好创意出现的时候，请记下它们，然后将它们应用到实际设计中。不断挑战，坚持不懈，灵活运用这些新技术，相信你的网页设计水平会有质的飞跃。



在这里，我们还有一个问题没有解决，就是使用 float 属性的元素不能被父元素包围的问题。浮动元素会对其下的元素产生影响，从而扰乱网页的整体布局。解决这个问题的方法有很多种，在这里我们使用 clearfix:after 这个伪类来解决。

使用 jQuery 实现淡入 / 淡出效果

在本部分，我们将一起探索一下浏览器的兼容问题，并学习使用 jQuery 实现淡入 / 淡出效果的方法。请看右图，图中使用 IE Tester 测试 tooltip_design.html 网页在旧版本浏览器（IE 6、7、8）中的显示情况。



从测试结果来看，在旧浏览器中圆形框与圆角悬浮框效果都消失了。更严重的问题时，旧版本浏览器不支持 Opacity 功能，鼠标指针即使不放在图片上，悬浮提示框仍然会显示出来。下面我们将采用一些方法来解决这些问题。在这里，我们说的解决问题并不是要在旧版本浏览器中实现与在新浏览器中一样的效果，而是指使用一些替换手段使网页内容更易阅读与观览。

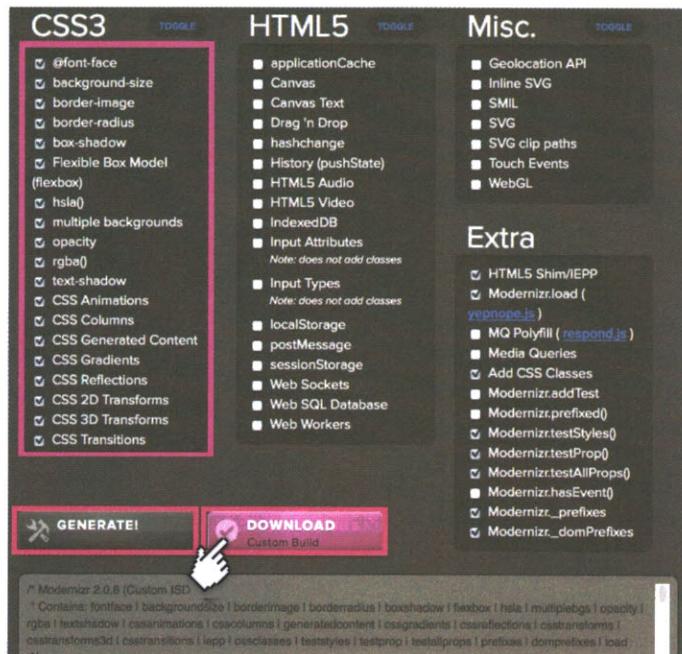
检查是否支持 CSS3 的 transition 属性

下面我们开始使用 jQuery 实现淡入 / 淡出效果。在实现淡入 / 淡出效果之前，首先要弄清楚哪些浏览器支持 CSS3 Transition。Modernizr 是一个开源的 JS 库，它是一个检测浏览器对 HTML5 与 CSS3 特性支持的 JS 库。下面我们将使用这个小工具来检测浏览器是否支持 CSS3 Transition。

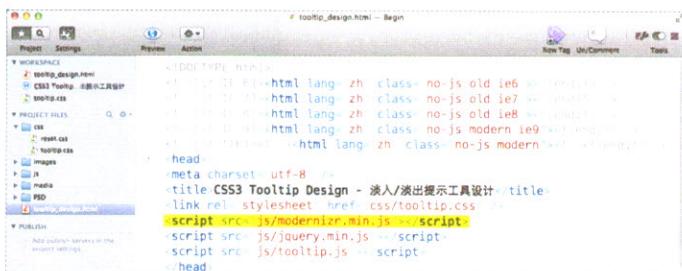
进入 Modernizr 官方网站 (<http://modernizr.com>)，在“Download Modernizr 2.6.2”中，单击“PRODUCTION”按钮，转到选项选择列表页面。

A screenshot of the Modernizr website. At the top, there's a navigation bar with links for DOWNLOAD, DOCUMENTATION, and RESOURCES. Below that is a quote from Bruce Bowens, a former BrowserLab Product Manager. The main content area has two main sections: 'Modernizr is a JavaScript library that detects HTML5 and CSS3 features in the user's browser.' and 'Why use Modernizr?'. It explains how Modernizr uses feature detection to support multiple browsers. There's also a 'How it works' section. On the right side, there's a large button labeled 'Download Modernizr 2.6.2' with a sub-section for 'PRODUCTION' builds. Below that is a 'Get started with Modernizr' section and a 'How Modernizr gives you fine control over the experience through JavaScript driven feature detection' section.

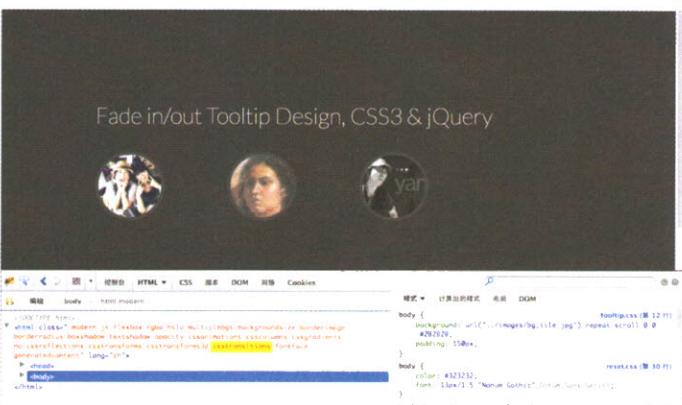
在选项选择列表页面中，全选所有 CSS3 项目，而后单击“GENERATE!”按钮，出现“DOWNLOAD”按钮后，单击它，指定保存位置后下载。下载完成后，将得到的文件放入 ex_04\Begin\js 文件夹中。



打开 tooltip_design.html 文档，在 <head></head> 之间添加调用 Modernizr 库代码（请根据下载的文件名称编写代码），如右图所示。



保存 tooltip_design.html 文件，在 Firefox 中打开它，启用 Firebug。单击 HTML 选项卡，在网页的 html 代码中，可以看到 Modernizr 为 <html> 标签添加了一个 class 属性，并且跟着一系列的属性值。这些属性值显示当前浏览器支持哪些功能。笔者使用的是 Firefox 18，除了 no-cssreflections、no-csstransforms3d 外，它支持所有 CSS3 的功能。



接下来，我们在 IE 8 中打开 tooltip_design.html 页面。那么 Modernizr 会检测出什么样的结果呢？可以看到除了 fontface、generatedcontent 外，CSS 的所有新功能 IE 8 均不支持。经过 Modernizr 检测后添加的代码非常有用，下面让我们一起看看如何使用这些代码。



打开 js/tooltip.js 脚本文件后，添加如下代码，这段代码是使用 jQuery 的基本代码，请熟记它。

```
;(function($) {  
    $(function() {  
        });  
    })(jQuery);
```

使用 JavaScript 自运行函数
通过jQuery ready()当文档准备好后运行
在内部使用\$代替jQuery

使用 JavaScript 的 if 语句，编写基本的代码结构，条件表达式为 Modernizr.csstransitions。

由于我们需要选出不支持 transition 属性的浏览器，因此在条件表达式 Modernizr.csstransitions 之前使用了“否”（!）运算符。在下面代码中，黄色代码部分表示：当条件表达式 Modernizr.csstransitions 为假时，执行花括号内的代码，即仅当浏览器不支持 CSS3 的 transition 时执行该代码。

```
;(function($) {  
    $(function() {  
        If(!Modernizr.csstransitions){  
            }  
        });  
    })(jQuery);
```

当浏览器支持CSS3的transition时执行其中的代码

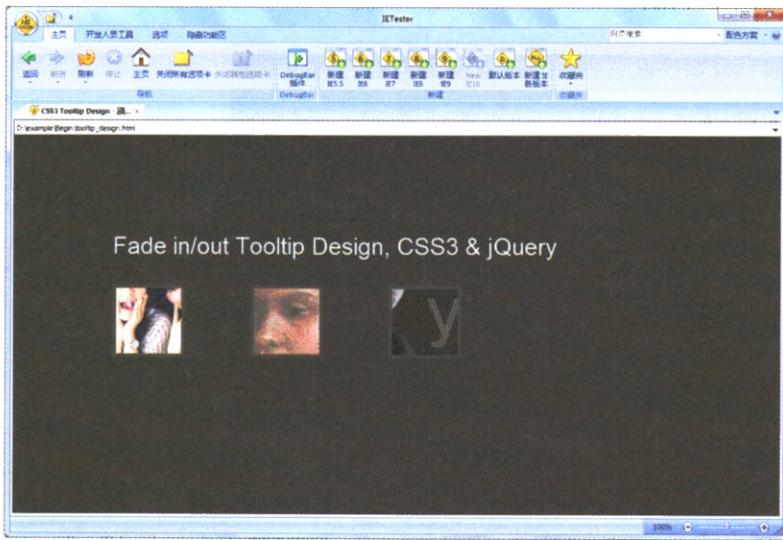
使用 jQuery 实现淡入 / 淡出效果

下面我们开始编写 jQuery 代码，使不支持 CSS3 transition 的浏览器能够处理 CSS3 变形。首先使用 jQuery 的 fadeTo() 方法，将悬浮框隐藏起来。

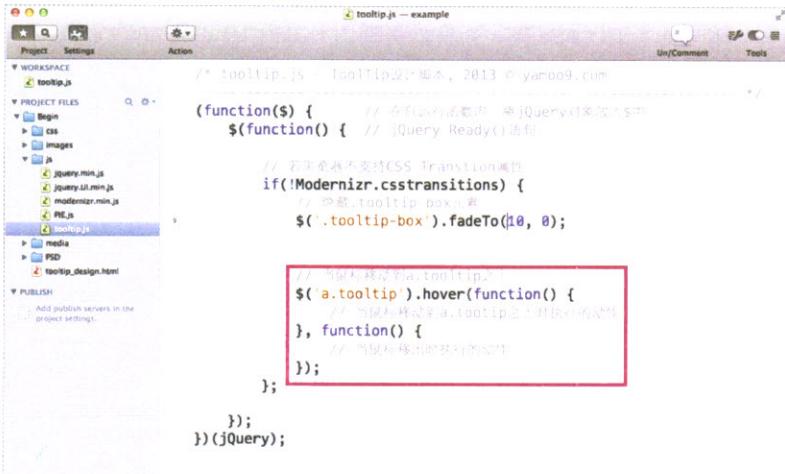
```
$('.tooltip-box').fadeTo(10, 0);
```

在文档中查找.tooltip-box，并使其从画面中隐藏

保存代码，在 IE Tester 中查看网页，可以发现悬浮框隐藏起来。接下来，我们要制作当鼠标指针移动到图片上后悬浮框慢慢显示的效果。



首先我们使用 `jQuery.hover()` 方法为鼠标悬浮与非悬浮两种状态分别编写代码。在 `hover()` 方法内部，可以编写两个 `function`，第一个 `function` 用于处理鼠标悬浮事件，第二个 `function` 用于处于鼠标非悬浮状态。



当鼠标指针移动到图片上时，即鼠标处于悬浮状态时，应该处理以下两个动作：

- ① `a.tooltip` 边框颜色逐渐改变；
- ② `.tooltip-box` 不透明度逐渐改变，位置也发生移动。

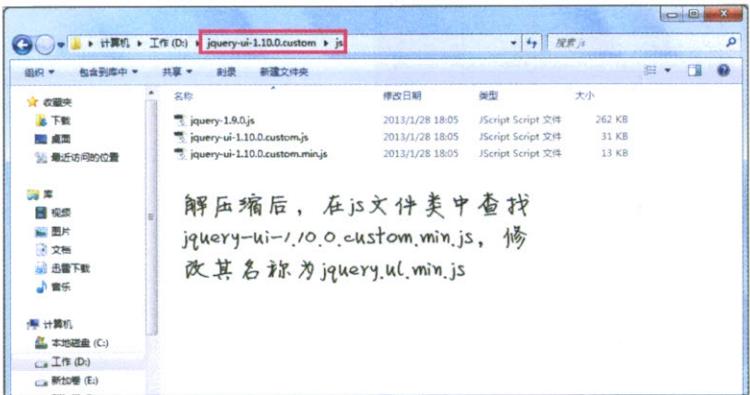
关于改变边框颜色的功能，我们通过添加 jQuery UI 插件实现。进入 jQuery UI (<http://jqueryui.com>) 网站，单击网页右侧的“Custom Download”按钮，进入 Download Builder 页面。

The screenshot shows the official jQuery UI website. At the top, there's a navigation bar with links for Plugins, Contribute, Events, Support, and jQuery Foundation. Below the header, there's a search bar and a main content area. On the left, there are sections for Interactions (Draggable, Droppable, Resizable, Selectable, Sortable) and Widgets (Accordion, Autocomplete). In the center, there's a large box titled "jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice." To the right of this box is a "Download jQuery UI 1.10.0" section with a "Custom Download" button and options for "Stable" (v1.10.0) and "Legacy" (v1.9.2, v1.8.14, v1.6.4). Below these are sections for "What's New in jQuery UI 1.10?" and "Developer Links" (Source Code, GitHub, API).

在 Download Builder 页面中，在 Components 中取消 Toggle All 复选框，在 Effects 中，点选 Effects Core 复选框，然后单击网页左下角的 Download 按钮，指定下载位置后，进行下载。

The screenshot shows the "Download Builder" page of the jQuery UI website. At the top, there's a navigation bar with links for Demos, Download, API Documentation, Themes, Development, Support, Blog, and About. Below the navigation is a search bar. The main content area has a title "Download Builder". It includes a "Quick download" section with links for Stable (v1.10.0), Legacy (v1.9.2), and GitHub (v1.10.0). There are two main sections: "Components" and "Effects". Under Components, there's a "Core" checkbox followed by "Widget", "Mouse", and "Position". Under Effects, there's a "Effects Core" checkbox followed by "Effects Local", "Blind Effect", "Bounce Effect", "Clip Effect", "Drop Effect", "Easing Effect", "Fade Effect", "Fold Effect", "Highlight Effect", "Pulse Effect", "Scale Effect", "Shake Effect", "Slide Effect", and "Transfer Effect". A red box highlights the "Effects Core" checkbox. Below these sections is a "Deselect all components" button. At the bottom, there's a "Select the files you want to include or change a theme" dropdown, a "UI address" input field, and a "Download" button.

下载完成后，解压缩文件，在 js 文件夹中包含 3 个文件，其中一个是 jQuery Core 文件，另外两个是 jQuery UI 文件。jQuery UI 文件名称有些长，为了方便使用，我们将其名称更改为 jquery.ui.min.js，复制后，粘贴到 begin.js 文件夹中。



小知识

在实例源文件中，笔者已经将 jquery.ui.min.js 文件复制到 ex_04\begin\js 文件夹中。若你下载了最新版本，直接将其覆盖即可。

打开 tooltip_design.html 文件，在其中添加调用 jquery.ui.min.js 的代码，如下图所示。



再次返回到 tooltip.js 文件中，添加如下代码。

```
$('a.tooltip').hover(function() {
    $(this).animate({'border-color': '#ffff'}, 400);          向$('a.tooltip')添加hover事件处理函数
}, function() {                                              若鼠标指针移动到其上时，在0.4秒期间边框颜色逐
    $(this).animate({'border-color': '#4b4b4b'}, 400);        漫变为白色
});
```

当鼠标指针移出时，在0.4秒期间边框颜色逐渐变为深灰色

小知识 jQuery animate() 方法

使用 jQuery 对象的 animate() 方法，可以为包装成 jQuery 对象的 DOM 元素设置动画效果，该方法有两个最主要的参数，第一个参数是一个 Object literal，包含在 CSS 表达式中；第二个参数设置动画处理时间，时间单位为“毫秒”（1/1000 秒），比如设置其为 400 毫秒，即 0.4 秒。

```
例) jQuery(document.body).animate({color:'tan',fontSize:'22px'}, 400);
```

Object literal

时间

在动画处理时间之后，我们也可以使用 function literal，它们是回调函数，在动画函数执行完成后调用执行。

```
例) jQuery(document.body).animate({color:'tan',fontSize:'22px'},400,
```

```
function(){...});
```

回调函数

在使用扩展插件 jQuery UI 时，也可以添加上 easing 值，规定在不同的动画点中设置动画速度。

```
例) jQuery(document.body).animate({color:'tan',fontSize:'22px'},400,'ease-in'
```

easing

关于 animate() 方法更多的内容，请访问 <http://api.jquery.com/animate> 页面，进行学习。

保存代码，在 IE Tester 中查看页面，可以发现当移动鼠标指针到图片上时，边框的颜色发生了改变。但是当把鼠标指针多次拖放到图片上时，边框颜色会发生多次改变，这是由于先前的动画与新动画重叠在一起而导致的问题。



下面我们使用 jQuery 的 stop() 方法来解决这一问题。

The screenshot shows the Eclipse IDE interface with a Java project named "tooltip_design". The code editor displays "tooltip.js" with the following content:

```
/* tooltip.js - Tooltip设计本, 2013 © yamo09.com */
(function($) {
    // 在自运行函数内，将jQuery对象放入$中
    $(function() { // jQuery Ready() 语句

        // 若浏览器不支持CSS Transition属性
        if(!Modernizr.csstransitions) {
            // 隐藏 .tooltip-box 元素
            $('.tooltip-box').fadeTo(10, 0);

            // 当鼠标移到a.tooltip之上
            $('a.tooltip').hover(function() {
                // 当鼠标移到a.tooltip之上时执行的动作
                $(this).stop().animate({ border-color: '#ffff' }, 400);
            }, function() {
                // 当鼠标移出时执行的动作
                $(this).stop().animate({ border-color: '#4b4b4b' }, 400);
            });
        }
    });
})(jQuery);
```

Annotations in the code editor:

- A red arrow points to the line `$(this).stop().animate({ border-color: '#ffff' }, 400);` with the text "调用stop(), 中止已有动画, 运行新动画".

小知识 ↗ ↘ jQuery 的 stop() 方法

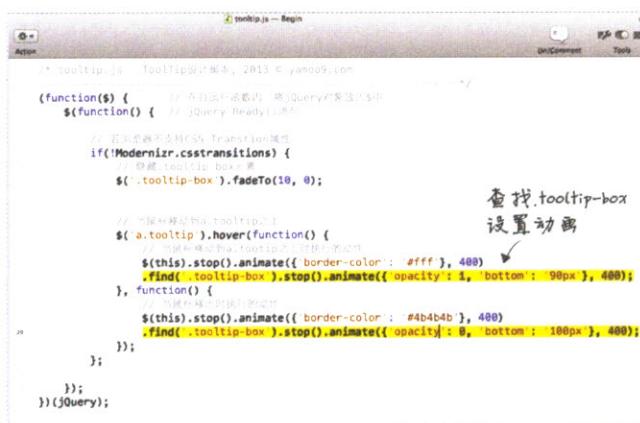
jQuery 对象的 stop() 方法用于停止当前正在运行的动画，执行后续新的动画。在使用 jQuery 的 animate() 方法处理彼此不同的移动时，当用户请求的动画次数大于 1 次时，动画就会交织纠缠在一起，这时使用 stop() 方法就可以很好地解决这一问题。

比如，有一个按钮，当按一次时，某个对象会消失，再按一次时，消失的对象又会显示出来。如果只是按一次按钮，它会正常工作，不会发生什么问题。但是，当多次按动按钮时，相应的对象会根据点按的次数反复执行动画，不断重复消失出现动画。为了解决这一问题，在 animate() 方法前添加上 stop() 方法，停止当前运行的动画，执行后面的动画。如此一来，即使用户多次按动按钮，也不会发生动画反复交错执行的问题。

关于 stop() 方法更多的知识，请访问 <http://api.jquery.com/stop> 网页，进一步学习。

接下来实现第二个动作，查找到 a.tooltip 内部的 .tooltip-box，而后向其 opacity 与 bottom 值分别应用动画。通过 jQuery Chain，添加如下代码。

```
$(this).animate({'border-color': '#fff'}, 400).find('.tooltip-box').stop().animate({'opacity': 1, 'bottom': '90px'}, 400);
```



The screenshot shows the IE Tester interface with the tooltip.js file open. A yellow box highlights the line of code: \$(this).animate({'border-color': '#fff'}, 400).find('.tooltip-box').stop().animate({'opacity': 1, 'bottom': '90px'}, 400);. Another yellow box highlights the word 'stop()' in the code. A red arrow points from the text '设置动画' (Set Animation) to the 'stop()' method.

```
/* tooltip.js -- ToolTip设计脚本, 2013 © yahoo9.com */
(function($) {
    // 在自定义函数内，将jQuery对象放入$中
    $(function() {
        // 若浏览器不支持CSS Transition属性
        if(!Modernizr.csstransitions) {
            // 生成 tooltip box
            $('.tooltip-box').fadeTo(10, 0);

            // 一鼠标移进时淡入淡出
            $('a.tooltip').hover(function() {
                $(this).stop().animate({ border-color: '#fff'}, 400)
                    .find('.tooltip-box').stop().animate({ opacity: 1, bottom: '90px'}, 400);
            }, function() {
                $(this).stop().animate({ border-color: '#4b4b4b'}, 400)
                    .find('.tooltip-box').stop().animate({ opacity: 0, bottom: '100px'}, 400);
            });
        }
    })(jQuery);
});
```



保存代码，在 IE Tester 中查看效果，可以看到在旧版本的浏览器（IE 6/7/8）中，我们使用 jQuery 实现了淡入/淡出效果，就像为旧版本的浏览器插上了一双会飞的翅膀一样。



最后，还要把方形图片框更改 为圆形框，当然是在不使用 border-radius 属性的前提下。在实现了圆形图片框之后，网页效果与在新浏览器中的效果几乎一样了。

5 章

制作 Lava Lamp 滑动导航条

导航是网页设计中必不可少的部分，它是通过一定的技术手段，为用户访问整个网站提供一定的途径，使其可以更方便地访问到所需要的内容。导航条以一种有条理的方式清晰地展示整个网站的层次结构，使网站的信息可以更有效地传递给用户。导航条的设计往往能够反映出设计人员对设计趋势的把握。前些年，设计导航条时，常常会使用 Flash 技术，这种技术设计出的导航条更酷、更炫。但是近来，导航设计又回归到追求简洁、直观、明确的设计理念上。勿容置疑，导航设计得越简洁，越直观、明确，它就越能将网站的内容准确、迅速地呈现给用户。在本章中，我们将学习设计具有 Lava Lamp 效果的滑动导航条，这种导航条既简洁、精炼、美观，又能充分发挥其职能，将网站信息有效地传递给用户。

设计效果预览

在本章中，我们将一起学习设计具有 Lava Lamp 效果的滑动导航条。这种滑动导航条设计的灵感源于熔岩灯（Lava Lamp），在熔岩灯的玻璃瓶体内装有特制的水溶液与蜡质固体，底部灯泡点亮后形成热量，传递到玻璃瓶。瓶底蜡质受热熔化变轻，便会徐徐上升；到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女；时而像火山爆发般狂热。



根据熔岩灯的特点，我们一起制作具有熔岩灯效果的滑动导航条。当鼠标指针移动到相应的导航菜单上时，岩浆状滑动块就会移动到其上，晃动数秒后静止不动。在实例源文件夹 \ex_05\Finish 中有一个名为“lavalamp_design.html”的网页文档，打开它，即可看到制作好的滑动导航条。设计网页标题时，我们使用具有律动感的 Montez 字体进行修饰。关于应用英文网页字体的方法，在第 4 章中我们已经讲解过了，相信大家不会陌生。

Lava Lamp Style Navigation Design

具有Lava Lamp效果的滑动导航条的设计灵感源于熔岩灯（Lava Lamp）。

在熔岩灯的玻璃瓶体内装有特制的水溶液与蜡质固体，底部灯泡点亮后形成热量，传递到玻璃瓶。瓶底蜡质受热熔化变轻，便会徐徐上升；到了顶部重新冷却又徐徐下降，不断往复，其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女；时而像火山爆发般狂热。

[Home](#) Interview - yamoo9 Lecture Open Lecture DVD WeBlog Contact

CHAPTER 5

设计网页布局

首先使用 Balsamiq Mockups 软件将头脑中关于滑动条的构思描绘出来。



然后使用 Photoshop 将描绘图设计出来，如下图所示。在使用 Photoshop 设计时，在图层面板中把 HTML 结构部分与 CSS 样式部分分别放在不同的组中，并为各个图层取一个易辨认的名字。设计时，要灵活使用参考线，将各个项目的位置、字体间距和行距等明确表示出来。



从设计师制作的 *.psd 文件，我们能很容易地猜测出设计师的设计水平。凡是具有一定实力的设计师都非常注意文件的编排与整理。特别是网页设计师，他们会更注重对各种网页素材文件的归纳整理。网页设计可不是单纯地制作图形图像那样简单，网页设计师要懂得设计网页布局，要明白网页在各种浏览器中如何显示。网页设计师的 PS 文件必须结构清晰，有条理，要成为一名合格的网页设计师，不仅要具备处理图形图像的能力，还要懂得设计网页的布局结构等。

编写 HTML5 代码

首先打开例题源文件，进入 \ex_05\Begin 文件夹中，在网页编辑器中，打开 lavalamp_design.html 文件。在 ① 中包含 HTML DTD 与 HTML 基本语句，② 中的条件注释用于检测用户所使用的网页浏览器类型，③ 在 `<head>` 区块中调用了 5 个外部文件，分别为 css/lavalamp.css、js/lavalamp.js、Modernizr 脚本库、jQuery.jQuery UI，④ 在 `<body>` 区块，插入我们要编写的代码。



The screenshot shows the Dreamweaver interface with the file "lavalamp_design.html" open. The code editor displays the following HTML5 code:

```
<!DOCTYPE html>
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->
<!--[if !IE]><!--><html lang="zh" class="no-js modern"><!-->
[endif]-->
<head>
<meta charset="utf-8" />
<title>Lava Lamp Navigation Design - 熔岩灯样式, 导航设计</title>
<link rel="stylesheet" href="css/lavalamp.css" />
<script src="js/modernizr.min.js"></script>
③ <script src="js/jquery.min.js"></script>
<script src="js/jquery.UI.min.js"></script>
<script src="js/lavalamp.js"></script>
</head>
<body>
④
</body>
</html>
```

小知识

在编写 HTML 代码时，要养成向 `<title>` 元素添加标题的习惯，一个良好的标题将更易于搜索引擎抓取。但是很多网页开发者常常不重视添加标题，往往使用 `untitled` 默认的无标题，这是非常不好的习惯。虽然我们在这里做的是练习，但仍然要按照实际设计的要求来做，这有助于您养成良好的代码编写习惯。

在网页的标题上使用 `<h1>` 标签，如下所示。

`<h1> Lava Lamp Style Navigation Design </h1>`

使用 `<p>` 标签，添加详细说明文本，如下所示。

`<p>` 具有 Lava Lamp 效果的滑动导航条的设计灵感源于熔岩灯（Lava Lamp）。

在熔岩灯的玻璃瓶体内装有特制的水溶液与蜡质固体，底部灯泡点亮后形成热量，传递到玻璃瓶，瓶底蜡质受热熔化变轻，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复，其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂热。

然后使用 HTML5 新增的 `<nav>` 元素定义页面上的导航条，指定其 `id` 属性为 “navigation” 。在 `<nav>` 标签内部，我们使用 `` 标签来列出所有导航项目。设置 `` 的 `class` 属性值为 “clearfix” ，使其能够包裹被设置为 `float` 的元素。

```
<nav id="navigation">
  <ul class="clearfix">
    <li></li>
  </ul>
</nav>
```

整个导航条由 6 个导航项目组成，使用 6 个 `` 标签来标识。在每个 `` 内部添加 `<a>` 元素，用来形成导航链接。对于第一个 `` 元素，设置其 `class` 属性为 “`focus`” ，表示该导航项目默认处于选中状态，当然我们也可以将其命名为 “`selected`” ，依各人的习惯而定。

```
<li class="focus"><a href="http://yamoo9.com/"> Home </a></li>
<li><a href="http://yamoo9.com/interview/"> Interview - yamoo9 </a></li>
<li><a href="http://is.gd/v8AyZH"> Lecture Open </a></li>
<li><a href="http://yamoo9.com/lecture-dvd/"> Lecture DVD </a></li>
<li><a href="http://is.gd/3JK6ti"> WeBlog </a></li>
<li><a href="http://is.gd/BcmrcX"> Contact </a></li>
```

最后，我们要使用一个包裹元素（ wrapper ），将 `<h1>`、`<p>` 和 `<nav>` 这 3 个元素包围起来。这种包裹元素，我们常常把它称为容器（ Container ）元素（关于容器元素更详细的内容，请参考第 7 章第 192 页中的内容）。在这里，我们使用 `<div>` 元素进行包裹，并且设置其 `id` 值为 “`page-wrap`” 。

```
<div id="page-wrap">
  <h1> Lava Lamp Style Navigation Design </h1>
  <p> 具有 Lava Lamp 效果的滑动导航条的设计灵感源于熔岩灯（ Lava Lamp ）。<br/>
    在熔岩灯的玻璃瓶体内装有特制的水溶液与蜡质固体，底部灯泡点亮后形成热量，传递到玻璃瓶，瓶底蜡质受热熔化变轻，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复，其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂热。</p>
  <nav id="navigation">
    <ul class="clearfix">
      <li class="focus"><a href="http://yamoo9.com/"> Home </a></li>
      <li><a href="http://yamoo9.com/interview/"> Interview - yamoo9 </a></li>
      <li><a href="http://is.gd/v8AyZH"> Lecture Open </a></li>
      <li><a href="http://yamoo9.com/lecture-dvd/"> Lecture DVD </a></li>
      <li><a href="http://is.gd/3JK6ti"> WeBlog </a></li>
      <li><a href="http://is.gd/BcmrcX"> Contact </a></li>
    </ul>
  </nav>
</div>
```

至此，HTML 文档编写完成，保存 HTML 文档后，在浏览器中打开它，查看页面效果，如右图所示。

Lava Lamp Style Navigation Design

具有 Lava Lamp 效果的滑动导航条的设计灵感源于熔岩灯（Lava Lamp）。在熔岩灯的玻璃圆柱体内装有特制的水溶液与硅质圆体，底部灯泡点热后形成热量，传递到玻璃瓶，温度越高受热膨胀变软，便会徐徐上升，到了顶部重新冷却又缓缓下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂热。

Home
Interview - yahoo09
Lecture Open
Lecture DVD
Welllog
Contact

小知识 代码校验

W3C 提供了在线代码校验服务 (<http://validator.w3.org>)，使用该服务，我们可以对编写好的 HTML5 代码进行校验。该网站提供了 3 种代码校验方式：第 1 种是通过输入 URL 地址进行校验，第 2 种是通过上传 html 文档进行校验，第 3 种是通过直接输入代码进行校验。选择适合的校验方法后，单击 Check 按钮，即可对编写的代码进行校验。

如果顺利通过校验，网页就会显示一个绿色信息条，告知用户顺利通过校验。在绿色信息条下进一步列出了更具体的信息，包括所使用的编码、文档类型、文档根元素和根命名空间等。在 Result 中显示校验结果，黄色部分是警告信息。由于目前 HTML5 标准正在完善中，因此黄色警告信息可以暂时忽略。

Validate by File Upload

Upload a document for validation:

File: [浏览...]

+ More Options

Character Encoding: Only if missing

Document Type: Only if missing

List Messages Sequentially Group Error Messages by Type

Show Source Clean up Markup with HTML-Tidy

Show Outline Validate error pages Verbose Output

Check

Note: file upload may not work with Internet Explorer on some versions or Windows XP Service Pack 2, see our information page on the W3C QA Website.

Validate by direct input

Enter the Markup to validate:

```
<html><head></head><body><ul><li>Home</li><li>Interview - yahoo09</li><li>Lecture Open</li><li>Lecture DVD</li><li>Welllog</li><li>Contact</li></ul></body></html>
```

W3C Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential issues Congratulations - Icons

This document was successfully checked as HTML5!

Result: Passed: 1 warning(s)

File: [浏览...]
Use the file selection box above if you wish to re-validate the uploaded file lavalamp.nav_design.html

Encoding: utf-8

Doctype: HTML5

Root Element: html

Options

Show Source Show Outline List Messages Sequentially Group Error Messages by Type
 Validate error pages Verbose Output Clean up Markup with HTML-Tidy

Revalidate

如果你对 HTML5 规范不太熟悉，建议使用这个在线校验工具校验代码。出现错误时，认真查看编写代码，找出错误原因，及时改正，避免下次出现同样的错误。

编写 CSS3 样式表

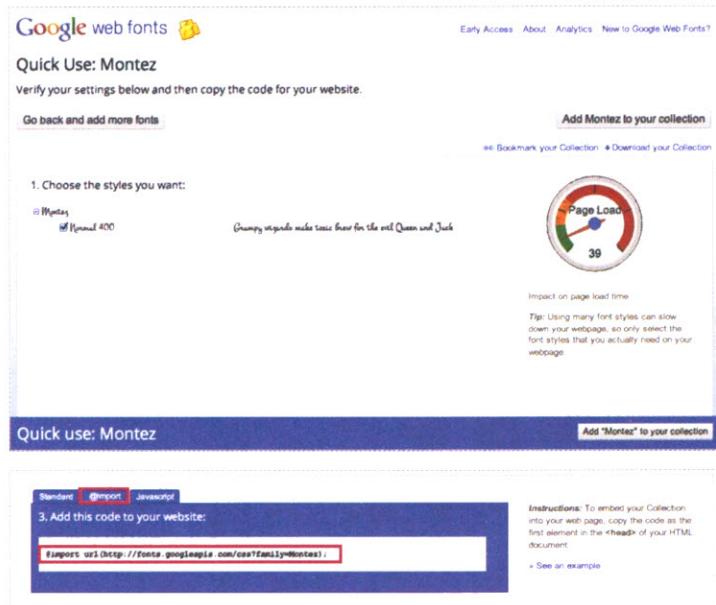
控制 body 样式

进入例题源文件的 css 文件夹中，打开 lavalamp.css 文件。而后为 `<body>` 元素指定背景颜色，为 `<h1>` 元素设置字体大小、颜色与样式。在这里，我们向标题文本应用 Montez 字体。

```
body {  
    background: #eeece7;          设置背景色为亮灰色  
}  
  
h1 {  
    font: 52px/1 'Montez';      设置字号、字体  
    color: #a4834d;              设置字体颜色为浅褐色  
}
```

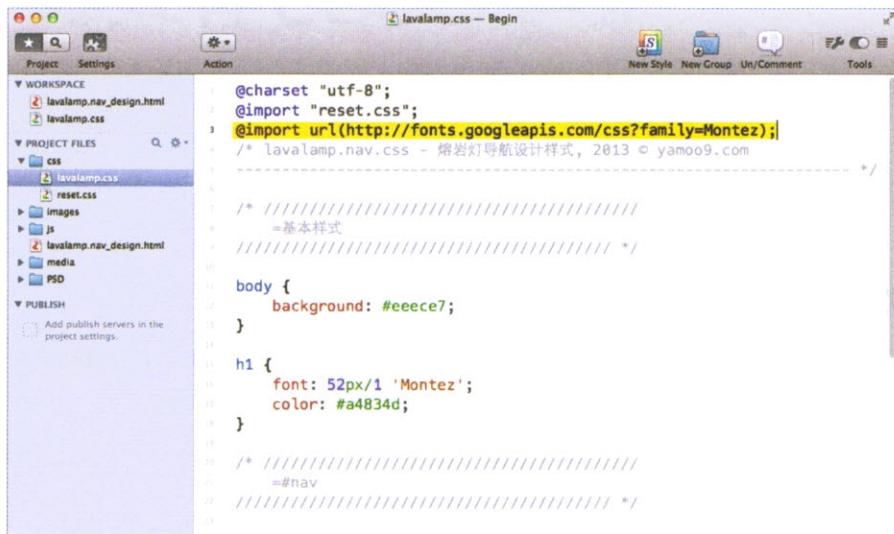
控制标题样式：应用英文网页字体

进入谷歌网页字体网站 (<http://www.google.com/webfonts>)，查找到 Montez 字体后，复制 @import 代码（关于应用英文网页字体的方法，请参考第 4 章第 95 页中的相关内容，在此不再赘述）。



The screenshot shows the 'Quick Use' section for the Montez font family on the Google Web Fonts website. At the top, there's a navigation bar with links for 'Early Access', 'About', 'Analytics', and 'New to Google Web Fonts?'. Below that, a button says 'Add Montez to your collection'. A 'Bookmark your Collection' link and a 'Download your Collection' link are also present. The main area is titled '1. Choose the styles you want:' and shows two font styles: 'Montez' and 'Montez of Normal 400'. To the right, there's a note: 'Grumpy regards make time slow for the evil Queen and Jack'. Below this is a 'Page Load' meter with a value of '39', accompanied by the text 'Impact on page load time' and a tip: 'Tip: Using many font styles can slow down your webpage, so only select the font styles that you actually need on your webpage.' At the bottom, a blue bar says 'Quick use: Montez' and 'Add "Montez" to your collection'. It includes tabs for 'Standard', '@import', and 'Javascript'. Under the '@import' tab, it says '3. Add this code to your website:' and provides the code: '@import url(http://fonts.googleapis.com/css?family=Montez);'. There's also an 'Instructions' link and a 'See an example' link.

切换到 lavalamp.css 样式表文件中，在顶部粘贴上复制的 @import 代码。



```
@charset "utf-8";
@import "reset.css";
@import url(http://fonts.googleapis.com/css?family=Montez);
/* lavalamp.nav.css - 熔岩灯导航设计样式, 2013 © yahoo9.com */

/*
 * =基本样式
 */
body {
    background: #eeece7;
}

h1 {
    font: 52px/1 'Montez';
    color: #a4834d;
}

/*
 * =#nav
 */

```

保存代码，在浏览器中查看网页效果，可以看到标题文本被应用上了 Montez 字体。



控制导航条边框样式

首先我们为标题下的说明文字添加底部空白，样式控制代码如下：

```
h1+p {
    margin-bottom: 3.5em;
}
```

接着，为整个导航区域 (#page-wrap) 设置宽度 (width)、外边距 (margin)、内边距 (padding)，以及背景颜色 (background)，代码如下：

```
#page-wrap {
    width: 960px;
    margin: 30px auto;
    padding: 3em;
    background: #fff;
}
```

保存代码，在浏览器中查看网页，可以看到设置的样式所呈现出来的效果。

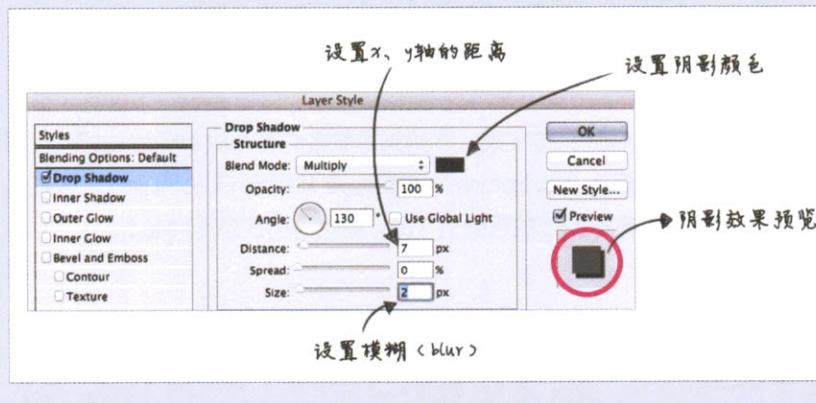


接下来，开始为 #page-wrap 制作圆角效果，并向内部添加阴影。首先为 #page-wrap 指定 box-shadow 属性，为导航区域添加上阴影效果。box-shadow 属性拥有 5 个参数，从左到右，依次为 x 轴偏移量、y 轴偏移量、阴影模糊半径、阴影颜色和阴影类型（内阴影或外阴影，默认为外阴影）。而后设置 border-radius 属性，指定圆角的圆度。

```
#page-wrap {  
    box-shadow: 0px 0px 2px #62615f inset;  
    border-radius: 20px;  
}
```

→ 小知识 ← 比较 box-shadow 参数与 Photoshop 的 Drop Shadow 参数

如果你经常使用 Photoshop 处理图片，那么你肯定知道 Drop Shadow，并且发现它与 box-shadow 参数非常相似。box-shadow 的 x 轴偏移量与 y 轴偏移量相当于 Drop Shadow 的 Distance，阴影模糊半径对应于 Size，阴影颜色对应于 Blend Mode Color，阴影类型与 Angle 类似。



注意 box-shadow 与 border-radius 是 CSS3 中的属性，使用时要添加上相应的前缀，以保证一部分旧版本浏览器支持它们。整理一下代码，如下图所示。

```

body {
    background: #eeece7;
}

h1 {
    font: 52px/1 'Montez';
    color: #a4834d;
}

h1+p {
    margin-bottom: 3.5em;
}

#page-wrap {
    width: 960px;
    margin: 30px auto;
    padding: 3em;
    background: #fff;
    box-shadow: 0px 0px 2px #62615f inset;
    -webkit-box-shadow: 0px 0px 2px #62615f inset;
    -moz-box-shadow: 0px 0px 2px #62615f inset;
    border-radius: 20px;
    -webkit-border-radius: 20px;
    -khtml-border-radius: 20px;
    -moz-border-radius: 20px;
}

```

为不同的浏览器
添加不同的前缀

保存代码，在浏览器中查看网页，可以发现 #page-wrap 已经被添加上了圆角效果，并且应用上了内阴影效果，如下图所示。



控制导航条菜单项样式

在前面，我们已经为导航框添加了样式，接下来，开始为导航条的各个菜单项添加样式。首先为 #navigation 设置 padding 为 3px，而后为 #navigation 的列表元素 li 指定 float 为 left，再在各个菜单项间添加竖直分割线，并且设置竖直分割线的粗细为 1px，颜色为亮灰色 (#eee)。

#navigation {	使用ID选择器，选择#navigation
padding: 3px;	设置内边距为3px
}	
#navigation li {	选择#navigation内的li
float: left;	设置向左浮动
border-right: 1px solid #eee;	为右边框设置粗细、类型和颜色
}	

接着，为导航条各个菜单项的超链接 (#navigation li a) 指定样式。首先要将 border 设置为 0，将其隐藏起来；再设置 padding 为 10px，向各个菜单项之间添加空白；最后为超链接设置字号、行距、字体和颜色。样式设置代码如下所示。

```
#navigation li a {  
    border: 0;          // 选择#navigation li内的  
    padding: 10px;      // 设置边框为0，将其隐藏起来  
    font: 14px/1 Verdana; // 设置内边距为10px  
    color: #a49e96;    // 设置字号、行距和字体  
}  
} // 为超链接设置字号、行距、字体和颜色
```

保存代码，在浏览器中查看页面，可以看到在各个菜单项之间出现了竖直分割线，并且各个菜单项之间的间隔也变得恰到好处。但仔细看，你会发现在最后一个菜单项的右侧也有一条竖直分割线，它是不必要的，我们要清除它。



首先使用 :last-child 伪类，查找到 #navigation 的最后一个列表元素，而后设置它的 border 为 0，将其隐藏起来。

```
#navigation li:last-child {  
    border: 0;          // 在#navigation中查找最后一个li  
}  
} // 设置边框为0
```

有关 #navigation 样式控制的代码，整理如下。

```
/*-----  
 *navigation  
 -----*/  
  
.navigation {  
    padding: 3px;  
}  
.navigation li {  
    float: left;  
    border-right: 1px solid #eee;  
}  
.navigation li:last-child {  
    border: 0;  
}  
.navigation li a {  
    border: 0;  
    padding: 10px;  
    font: 14px/1 Verdana;  
    color: #a49e96;  
}  
  
/*-----  
 *Global Classes  
 -----*/  
.clearfix:after {
```

控制导航条样式：渐变

当鼠标指针移动到各个菜单项目上，或者按 Tab 键使菜单项目获得焦点时，相应的菜单项目状态要发生改变，显示出带有颜色渐变的圆角矩形框。首先我们使用 :hover 与 :focus 伪类指定要应用样式的状态，而后分别设置圆角、颜色渐变与阴影效果。在设置颜色渐变时，我们将使用 CSS3 的 linear-gradient (线性渐变) 技术实现颜色渐变效果。

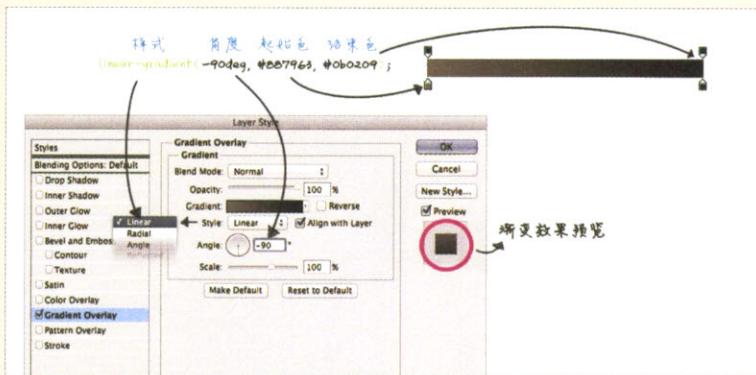
```
#navigation li a:hover,  
#navigation li a:focus {  
    border-radius: 6px;  
    background-image: linear-gradient(-90deg, #887963, #0b0a09);  
    box-shadow: 0px 0px 2px #69635a;  
}
```

当鼠标指针移动到#navigation内的a时
当#navigation的a获得焦点时
设置圆角边框为6px
用于CSS3渐变效果
设置外阴影为深褐色 (#69635a)

小知识 CSS3 的渐变技术

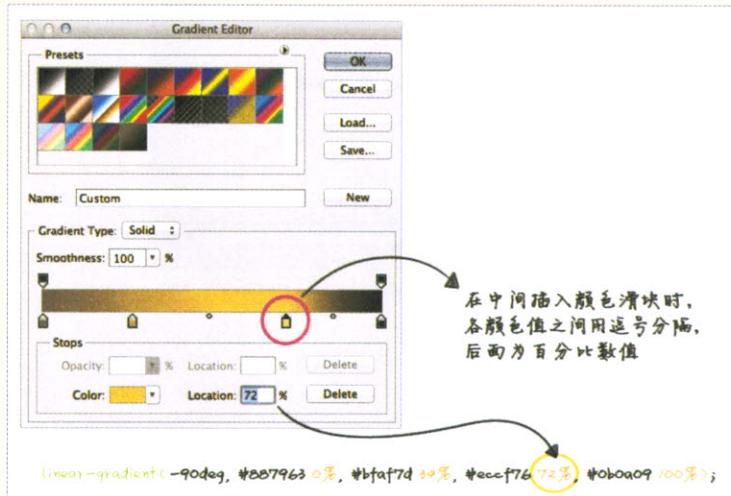
有了 CSS3 的渐变技术，即使不使用 Photoshop 也能轻松制作出渐变效果按钮。使用纯 CSS 制作出的渐变按钮，比使用 Photoshop 制作的图片，在网页上的加载速度要快很多。因为显示时浏览器不需要向服务器请求图像数据，直接解析代码就能绘制出渐变按钮。在网站维护时，也不需要修改或替换图片，只需更改相应的代码值即可。

比较 CSS3 的渐变设置与 Photoshop 的渐变样式，我们会更容易地掌握它。请看下图，① CSS3 渐变分为线性渐变 (Linear) 与径向渐变 (Radial) 两种，与 Photoshop 中的渐变类型类似。② 角度 (Angle) 与 Photoshop 中的角度 (Angle) 相同，可以理解为渐变方向。③ 设置渐变颜色时，要分别输入起点颜色与终点颜色。



在 Photoshop 中我们可以向渐变条添加多种颜色，制作出复杂的渐变效果。同样，在 CSS3 中，我们也可以使用多种颜色，制作出多彩的渐变效果。

请看下图，在前面 CSS3 渐变代码基础上，继续添加要使用的颜色，各种颜色通过逗号分隔开，可以不断添加多种颜色。在颜色后面，要使用百分号的形式指出颜色的位置。



当仅使用两种颜色时，我们可以将颜色后面标识颜色位置的百分值省略，默认条件下，第一种颜色的位置为 0%，第二种颜色的位置为 100%。

现在各种浏览器对 CSS3 的渐变代码的支持不够理想，使用时，需要针对各种浏览器添加相应的前缀。这可能会导致代码有些冗长、杂乱，但目前我们只能这样做。希望各浏览器尽快全面支持 CSS3 的渐变代码。针对各浏览器，添加相应前缀，代码整理如下。

```

Action New Style New Group Un/Comment Tools
border: 0;
padding: 10px;
font: 14px/1 Verdana;
color: #a49e96;
}

#navigation li a:hover,
#navigation li a:focus, {
-webkit-border-radius: 6px;
-khtml-border-radius: 6px;
-moz-border-radius: 6px;
border-radius: 6px;

-webkit-box-shadow: 0px 0px 2px #69635a;
-moz-box-shadow: 0px 0px 2px #69635a;
box-shadow: 0px 0px 2px #69635a;

background-image: -webkit-gradient(linear, left top, left bottom, from(#887963),
to(#0b0a09));
background-image: -webkit-linear-gradient(-90deg, #887963, #0b0a09);
background-image: -moz-linear-gradient(-90deg, #887963, #0b0a09);
background-image: -o-linear-gradient(-90deg, #887963, #0b0a09);
background-image: -ms-linear-gradient(-90deg, #887963, #0b0a09);
background-image: linear-gradient(-90deg, #887963, #0b0a09);
}

/* ////////////////////////////// */
/* Global Classes */
/* ////////////////////////////// */

.clearfix:after {
content: "";
display: block;
clear: both;
}

```

保存代码，在浏览器中查看页面，当鼠标指针移动到相应菜单项上，或按 Tab 键使某项菜单项获得焦点时，相应菜单项就会显示出渐变按钮，如图所示。

Lava Lamp Style Navigation Design

具有Lava Lamp效果的滑动导航条的设计灵感源于熔岩灯 (Lava Lamp)。
在熔岩灯的玻璃瓶体内装有特别的水溶液与铂类固体，底部灯泡点亮后形成热量，传递到玻璃瓶，瓶底温度变热融化变轻，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡。多姿多彩的状态时而如少女，时而像火山爆发狂热。

Home Interview - yamoo9 Lecture Open Lecture DVD WeBlog Contact

```
.navigation li a:hover, .navigation li a:focus {  
background-image: -moz-linear-gradient(-90deg, #8879E3, #000A09);  
border-radius: 6px 6px 6px;  
box-shadow: 0 0 2px #666633;  
}  
  
.clearfix:after {  
clear: both;  
content: '';  
display: block;  
}  
  
.ie6 .clearfix {
```

指定默认激活的菜单项

最后，我们为导航条指定一个默认的激活菜单项，使其处于激活状态。在这里，我们指定第一个菜单项为默认激活项。在 HTML 代码中我们已经为第一个菜单项设置了 class 属性为 focus。我们使用 focus 识别出第一个菜单项，使其样式处于激活状态。编写代码时，只需在 `#navigation li a:focus` 下面添加 `#navigation li.focus a` 即可，如下所示。

```
#navigation li a:hover,  
#navigation li a:focus,  
#navigation li.focus a {  
    #navigation li中选择设有.focus的li内部的a
```

到这儿，CSS 样式控制代码编写完成。保存代码，在浏览器中查看网页，可以看到第一个菜单项默认处于激活状态，如图所示。在动作设计部分，我们还要修改一下 CSS 代码。关于如何修改，我们将在后面编写动作脚本的内容中进行详细讲解。

Lava Lamp Style Navigation Design

具有Lava Lamp效果的滑动导航条的设计灵感源于熔岩灯 (Lava Lamp)。
在熔岩灯的玻璃瓶体内装有特别的水溶液与铂类固体，底部灯泡点亮后形成热量，传递到玻璃瓶，瓶底温度变热融化变轻，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡。多姿多彩的状态时而如少女，时而像火山爆发狂热。

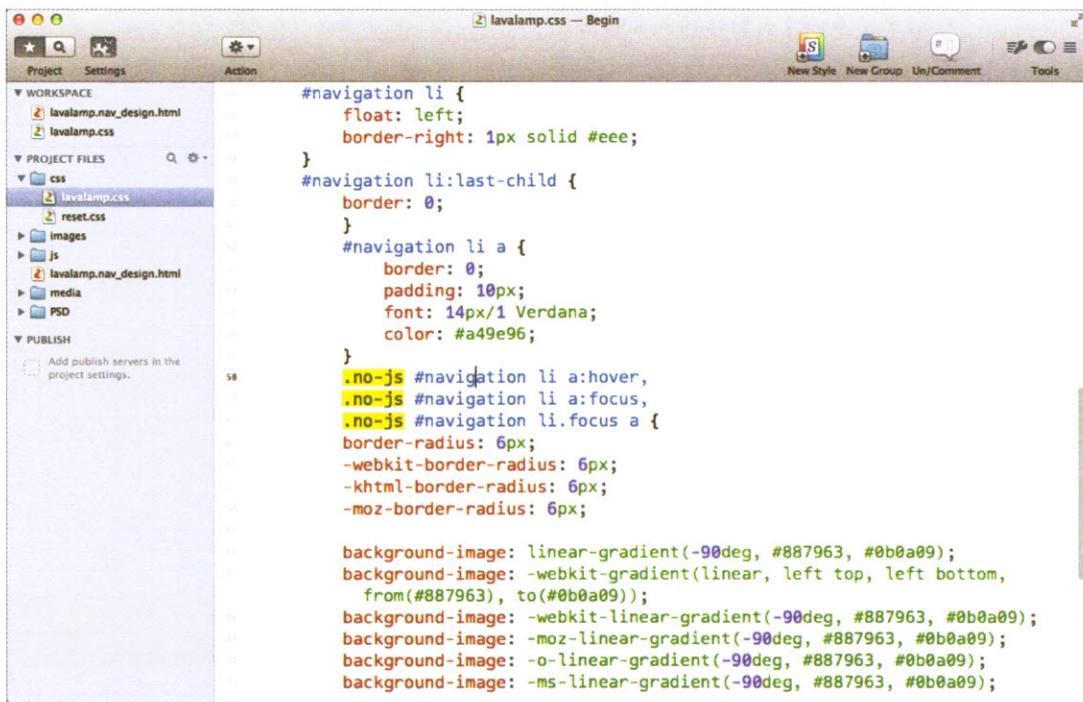
← 即使未移动鼠标指针至某上或它未获得焦点，它默认处于激活状态

Home Interview - yamoo9 Lecture Open Lecture DVD WeBlog Contact

```
.navigation li a:hover,  
.navigation li a:focus,  
.navigation li.focus a {  
border-radius: 6px;  
-webkit-border-radius: 6px;  
-khtml-border-radius: 6px;  
-moz-border-radius: 6px;  
  
background-image: linear-gradient(90deg, #8879E3, #000A09);  
background-image: -webkit-gradient(linear, left top, left bottom, from(#8879E3), to(#000A09));  
background-image: -webkit-linear-gradient(-90deg, #8879E3, #000A09);  
background-image: -moz-linear-gradient(-90deg, #8879E3, #000A09);  
background-image: -o-linear-gradient(-90deg, #8879E3, #000A09);
```

编写 jQuery 脚本代码

首先在 lavalamp.css 样式表中，向 #navigation li a:hover、#navigation li a:focus 和 #navigation li.focus a 选择器前添加 .no-js。运行 Modernizr 库，在支持脚本的环境下，no-js 值不会被解析；而在不支持脚本的环境下，no-js 值会被先替换为 js 值，然后再进行解析处理。该部分由 HTML 文档调用的 Modernizr 库来处理（关于 Modernizr 库的详细内容，请参考第 4 章第 111 页中的内容）。



```
#navigation li {  
    float: left;  
    border-right: 1px solid #eee;  
}  
#navigation li:last-child {  
    border: 0;  
}  
#navigation li a {  
    border: 0;  
    padding: 10px;  
    font: 14px/1 Verdana;  
    color: #a49e96;  
}  
.no-js #navigation li a:hover,  
.no-js #navigation li a:focus,  
.no-js #navigation li.focus a {  
border-radius: 6px;  
-webkit-border-radius: 6px;  
-khtml-border-radius: 6px;  
-moz-border-radius: 6px;  
  
background-image: linear-gradient(-90deg, #887963, #0b0a09);  
background-image: -webkit-gradient(linear, left top, left bottom,  
from(#887963), to(#0b0a09));  
background-image: -webkit-linear-gradient(-90deg, #887963, #0b0a09);  
background-image: -moz-linear-gradient(-90deg, #887963, #0b0a09);  
background-image: -o-linear-gradient(-90deg, #887963, #0b0a09);  
background-image: -ms-linear-gradient(-90deg, #887963, #0b0a09);
```

保存代码，在 Firefox 中打开文档后，运行 Firebug 插件，可以看到 <html> 元素的 class 属性值中的 no-js 被替换成 js。换言之，导航条中并未应用激活的样式。

Lava Lamp Style Navigation Design

具有Lava Lamp效果的滑动导航条的设计灵感源于熔岩灯（Lava Lamp）。

在熔岩灯的玻璃瓶体内装有特别的水溶液与蜡质固体，底部灯泡点亮后形成热量，传递到玻璃瓶，瓶底蜡质受热熔化变轻，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复，其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂热。

Home Interview - yamoo9 Lecture Open Lecture DVD WeBlog Contact



声明变量与引用文档对象

下面我们开始使用 jQuery 为导航条编写动作脚本。首先引用（Reference）文档对象，要引用的对象是在 #navigation、#navigation 的 focus、导航条上移动的 Lava。对指定对象进行引用后，它会被加载到内存中，省去了反复查找的麻烦。我们常常把对象加载并保存在内存中的过程称为缓存（Cache），使用缓存（Cache）技术，能极大地提升系统的性能（Performance）。

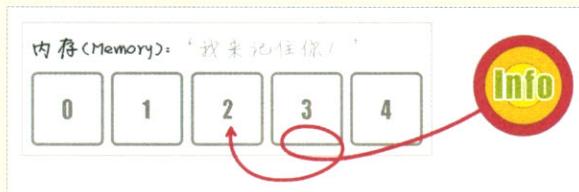
```
var $nav = $('#navigation');          将含有#navigation的jQuery对象保存到$nav变量中  
$current_item = $nav.find('.focus'); 在$nav中查找.focus对象后保存到$current_item变量中
```

```
$lava = $('li class="lava" />';          将动态生成的<li class="lava">保存到$lava变量中
```

```
/* lavalamp.nav.js - 熔岩灯导航设计样式, 2013 © yamoo9.com */  
(function($) { // JavaScript自运行函数  
  $(function() { // jQuery Ready() 语句  
  
    // 对象引用  
    var $nav = $('#navigation'),  
        $current_item = $nav.find('.focus'),  
        $lava = $('li class="lava"/');  
  
  });  
})(jQuery); // 传入window.jQuery对象
```

小知识 引用与缓存

在前面的学习过程中，涉及了内存、引用和缓存等术语，下面我们一起了解一下这些术语的含义。首先讲一下内存（Memory），内存是一种记忆装置，用于暂存程序以及数据信息。我们可以把某些数据指定保存到内存的特定部分中。若想访问这些数据信息，需要先给保存这些数据的内存片段指定一个名字，这个过程我们称为“变量声明”，而后再通过变量访问这些数据。



变量是一段有名字的连续存储空间，该空间中保存着数据信息。当访问这些数据信息时，直接通过变量名即可访问它们。缓存（Cache）是指可以进行高速数据交换的存储器，它先于内存与 CPU 交换数据，速率很快。



引用（Reference）是某一变量（目标）的一个别名，对引用的操作与对变量直接操作完全一样。比如，我们常常使用的“快捷方式”，在 Windows 桌面上常常会有大量的快捷方式，它们就是某些程序的引用，通过这些“快捷方式”，我们可以直接访问它们所对应的程序，使用起来非常方便。比如，我们将 Firefox 浏览器安装在 C 盘中，在 C:\Program File\Mozilla Firefox 文件夹中会有一个名为 firefox.exe 的可执行文件，双击它即可运行 Firefox 浏览器。但每次启动 Firefox 浏览器时，都要先进入到 C:\Program File\Mozilla Firefox 文件夹中，再双击 firefox.exe 会显得非常麻烦。所以，我们通常会为 firefox.exe 创建一个快捷方式，通过双击快捷方式，即可运行 Firefox 浏览器。

也就是说，firefox 快捷方式引用了 firefox.exe 程序。引用不是一个实实在在的对象，它只是指向某个对象。倘若 firefox 快捷方式是一个真实的对象，那么当删除它时，firefox.exe 也会一起被删除，显然这是不对的。



→ 小知识 ← 动态生成 HTML

动态生成 HTML 是指浏览器在运行的时候动态生成 HTML 页面。与动态生成 HTML 相反，还有一个静态编写 HTML 的概念。静态编写 HTML 就是在制作网页时直接手动输入静态的 HTML 代码。动态生成 HTML 代码通常都是使用 JavaScript 脚本实现的，当执行这些 JavaScript 脚本时，就会动态生成 HTML 页面。在使用 jQuery 时，将 HTML 代码以字符串的形式放入到 `jQuery()` 函数的括号内，即可动态生成 HTML 代码。

```
<script src="js/jquery.js"></script>
<script src="js/jquery.min.js"></script>
<script src="js/jquery.U1.min.js"></script>
<script src="js/jquery.lavalamp.js"></script>
<script src="js/navigation.js"></script>
</head>
</body>

<div id="page-wrap">
  <h1>Lava Lamp Style Navigation Design</h1>
  <p>具有Lava Lamp效果的滑动导航条的HTML源码原来是灯(Lava Lamp)。<br />
    灯焰灯是玻璃瓶内装有特别制的水溶液与铂丝组件。随着灯泡加热后形成热量，传递到玻璃瓶，瓶子被加热，便会徐徐上升，到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂野。
  </p>
  <nav id="navigation">
    <ul class="clearfix">
      <li class="focus"><a href="http://yamo09.com/Home">Home</a></li>
      <li><a href="http://yamo09.com/interview/">Interview - yamo09</a></li>
      <li><a href="http://is.gd/v8AyZH">Lecture Open</a></li>
      <li><a href="http://yamo09.com/lecture-dvd/">Lecture DVD</a></li>
      <li><a href="http://is.gd/J3K6t1">WeBlog</a></li>
      <li><a href="http://is.gd/BcmrcX">Contact</a></li>
    </ul>
  </nav>
</div><!-- e: #page-wrap -->
</body>
</html>
```

使用 jQuery 的 `.appendTo()` 方法在 `$nav` 引用对象中查找到 `` 后，在 `` 代码的最后插入 `$java`。

`$java.appendTo($nav.find('ul'));` 将 `$java` 插入到 `$nav` 之后

```
Lava Lamp Style Navigation Design

具有Lava Lamp效果的滑动导航条的HTML源码原来是灯(Lava Lamp)。
在焰灯内装有特别制的水溶液与铂丝组件。随着灯泡加热后形成热量，传递到玻璃瓶，瓶内玻璃受热膨胀变轻，便会上升，到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂野。

Home Interview - yamo09 Lecture Open Lecture DVD WeBlog Contact
```

```
<script src="js/jquery.js"></script>
<script src="js/jquery.min.js"></script>
<script src="js/jquery.U1.min.js"></script>
<script src="js/jquery.lavalamp.js"></script>
<script src="js/navigation.js"></script>
</head>
</body>

<div id="page-wrap">
  <h1>Lava Lamp Style Navigation Design</h1>
  <p>具有Lava Lamp效果的滑动导航条的HTML源码原来是灯(Lava Lamp)。<br />
    灯焰灯是玻璃瓶内装有特别制的水溶液与铂丝组件。随着灯泡加热后形成热量，传递到玻璃瓶，瓶内玻璃受热膨胀变轻，便会上升，到了顶部重新冷却又徐徐下降，不断往复。其独特的功能和独特的视觉享受令人心旷神怡，多姿多彩的状态时而如少女，时而像火山爆发般狂野。
  </p>
  <nav id="navigation">
    <ul class="clearfix">
      <li class="focus"><a href="http://yamo09.com/Home">Home</a></li>
      <li><a href="http://yamo09.com/interview/">Interview - yamo09</a></li>
      <li><a href="http://is.gd/v8AyZH">Lecture Open</a></li>
      <li><a href="http://yamo09.com/lecture-dvd/">Lecture DVD</a></li>
      <li><a href="http://is.gd/J3K6t1">WeBlog</a></li>
      <li><a href="http://is.gd/BcmrcX">Contact</a></li>
    <ul>
      <li class="java"><a href="#">Java</a></li>
    </ul>
  </nav>
</div><!-- e: #page-wrap -->
</body>
</html>
```

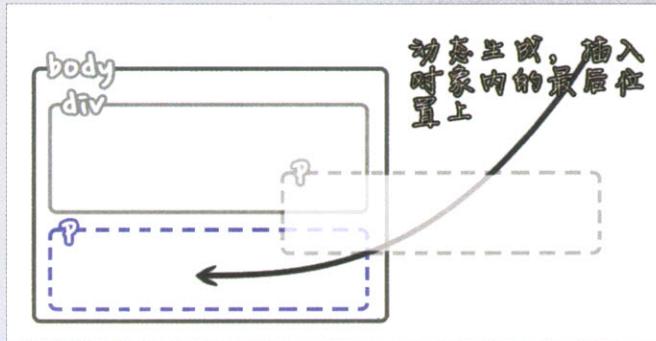
#navigation li:last-child { border: 0 none; }
#navigation li { border-right: 1px solid #555555; float: left; }

html, body, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, font, img, ins, kbd, q, s, samp, small, strike, strong,

→ 小知识 ← jQuery 的 appendTo() 方法

jQuery 对象的 appendTo() 方法用于在被选元素的末尾（但仍在元素的内部）插入指定的内容。

例) `jQuery('<p>请插在 body 元素的末尾 </p>').appendTo('body');`
要插入的元素 目标对象（父元素）



append() 方法与 appendTo() 方法作用类似，都用于将一个元素插入到另一个元素中。但两个方法还是有明显的不同的。A append B 指将 A 元素插到 B 元素的末尾，而 A appendTo B 则是将 B 元素插到 A 元素的末尾。关于这两个方法，更详细的内容，请访问 <http://api.jquery.com/appendto> 页面进行学习。

控制 lava 样式

下面我们开始向 \$lava 添加样式，主要使用 CSS 实现。当然根据不同情况，也会用到 JavaScript 脚本。使用 jQuery.css() 方法，能够很容易地添加添加样式。首先为 \$lava 设置宽度、高度与背景色，代码如下：

```
$lava.css({  
    width : $current_item.outerWidth(),  
    height : $current_item.outerHeight(),  
    backgroundColor : '#eee'  
}).appendTo($nav.find('ul'));
```

使用.css()方法向\$lava添加样式
将宽度设置为\$current_item元素的宽度 (width+padding+border)
将高度设置为\$current_item元素的高度 (height+padding+border)
设置背景色

→ 小知识 ← jQuery 的 css() 方法

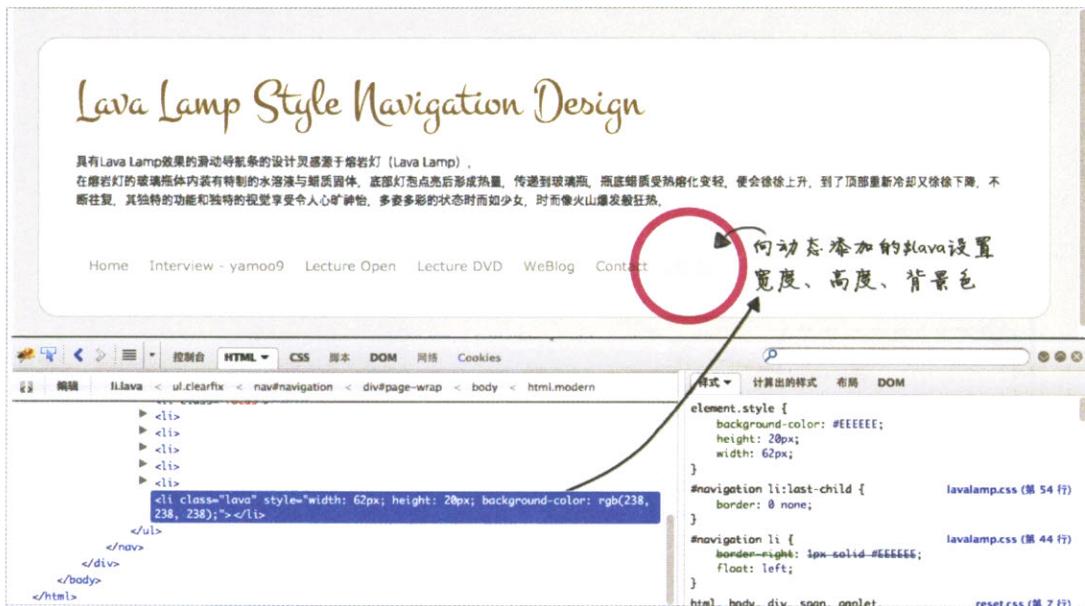
该方法用于获取或设置包装在 jQuery() 对象中的 DOM 元素的样式属性。使用时，CSS 属性与值要使用逗号分隔开来。

例) `jQuery('#page-wrap').css('color', '#ab3cdf');`
对象元素 样式设置

若想同时设置两个以上的属性，只需要给 css() 方法 CSS Map 类型的对象枚举即可。各个枚举项之间使用逗号分隔，属性与属性值使用冒号分隔。更多详细说明，请访问 <http://api.jquery.com/css> 页面进行学习。

```
jQuery('#page-wrap').css({  
    'color' : '#ab3cdf',  
    'font-size' : '24px'  
});
```

保存代码，在浏览器中运行 Firebug 插件，可能观察到使用 \$lava 与 .css() 方法添加的样式，如图所示。



设置 lava 的位置

接下来，我们为 lava 设置位置。首先设置 \$lava 为绝对定位，指定 \$nav 为相对定位。而后将 top 与 left 的值设置为当前位置（\$current_item），使其移动到当前处于激活状态的位置上。使用 css() 方法指定 \$nav 对象的定位方式为相对定位。

```
$nav.css('position', 'relative');
```

使用 .css() 方法为 \$nav 设置为相对定位

然后在 \$lava 对象的 css() 方法中，设置定位方式为绝对定位，并将 top 与 left 设置为当前位置。

```
$lava.css({  
    position: 'absolute',  
    top: $current_item.position().top,  
    left: $current_item.position().left,
```

设置绝对定位

将 top 设置为 \$current_item 的 top 值

将 left 设置为 \$current_item 的 left 值

运行 Firebug 插件，可以发现设置的属性正常运作，但 \$lava 的位置与 \$current_item 位置不一致，但导航条最左侧的 Home 菜单项被 \$lava 覆盖隐藏起来。



Home 菜单项为什么会被 \$lava 覆盖起来了呢？这是因为 \$lava 对象在 Z 轴更高的一层上。我们能很容易地解决这一问题，只要把 \$nav 内部的 <a> 元素在 Z 轴的位置提得更高一些就可以了，代码如下。

```
$nav.css('position', 'relative')
  .find('a').css({
    position: 'relative',
    zIndex: 1
});
```

在\$nav内查找<a>并添加样式
设置为相对定位
设置沿Z轴的位置

小知识 ↗ jQuery 的 find() 方法

该方法用于在包裹于 jQuery 对象中的 DOM 元素中查找指定的对象。比如，要查找 元素中最后一个 元素，只需要使用如下代码进行查找就可以了。更多详细内容，请访问 <http://api.jquery.com/find> 页面进行学习。

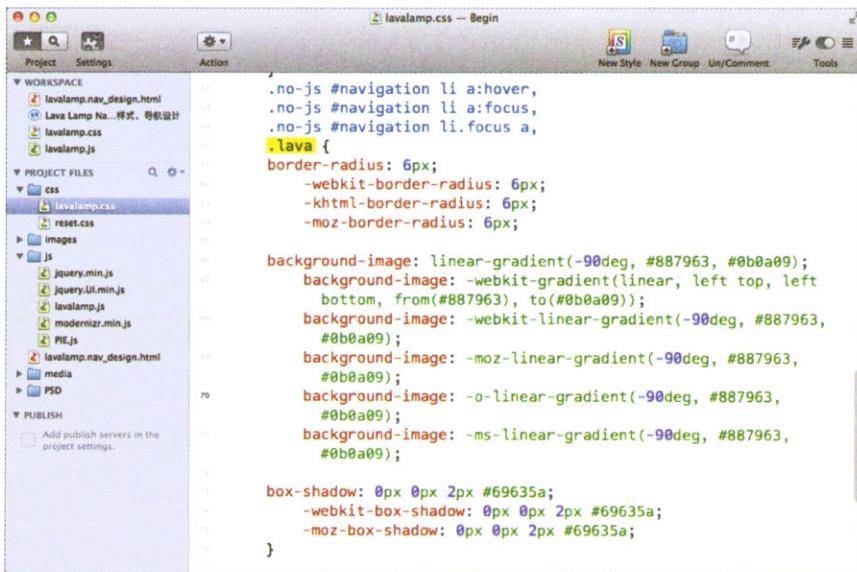
例) `jQuery('ul').find('li:last');`
待查找的元素

运行 Firebug 插件，可以发现 Home 菜单项已经出现在 \$lava 之上了，如图所示。



向 lava 应用渐变

下面我们开始向 lava 应用渐变效果。在前面的学习中，我们已经学会如何创建渐变效果了。在本部分中，只要在 `lavalamp.css` 文件的渐变代码前添加上 `.lava` 选择器，即可完成向 lava 应用渐变效果的工作，如下图所示。



```
.no-js #navigation li a:hover,
.no-js #navigation li a:focus,
.no-js #navigation li.focus a,
.lava {
    border-radius: 6px;
    -webkit-border-radius: 6px;
    -khtml-border-radius: 6px;
    -moz-border-radius: 6px;
    background-image: linear-gradient(-90deg, #887963, #0b0a09);
    background-image: -webkit-gradient(linear, left top, left bottom, from(#887963), to(#0b0a09));
    background-image: -webkit-linear-gradient(-90deg, #887963, #0b0a09);
    background-image: -moz-linear-gradient(-90deg, #887963, #0b0a09);
    background-image: -o-linear-gradient(-90deg, #887963, #0b0a09);
    background-image: -ms-linear-gradient(-90deg, #887963, #0b0a09);

    box-shadow: 0px 0px 2px #69635a;
    -webkit-box-shadow: 0px 0px 2px #69635a;
    -moz-box-shadow: 0px 0px 2px #69635a;
}
```

保存代码，运行 Firebug 插件，可以看到渐变效果已经被应用到 `$lava` 之上。



调整 lava 的位置

仔细观察，你会发现 `$lava` 的高度与 CSS 中设置的略微有些不同。这是因为 `$current_item` 变量引用的对象是 `li.focus`，而在 CSS 设计中针对的是 `li.focus a`，而不是 `li.focus`。请看下图，从图中可以很容易地看出，它比 `$lava` 的高度要高一些。

修改代码，增加 \$lava 的高度。\$lava 的高度由 \$current_item 的 outerHeight() 高度值进行设定。在这里，我们在其基础上增加 20px，使 \$lava 的高度增加，修改代码如下。保存代码，查看效果，可以发现 \$lava 的高度增加了 20px。

```
$lava.css{
  height: $current_item.outerHeight() + 20,
```

设置\$lava的高度为\$current_item的高度加上20px

但是由于上部的位置数值未改动，因此增加高度时只是在向下的方向上增加了。若想使其位于中间位置上，则需要使上部位置的数值减少 20 的一半，修改代码如下。保存代码，在浏览器中查看页面，可以发现 lava 已经居于中间了，如图所示。

```
$lava.css{
  top: $current_item.position().top - (20/2),
```

设置\$lava的top为\$current_item的top减去20/2

```
17  });
18
19 // $lava操作及样式
20 // lava.css
21 // lava操作
22 // lava.css
23 // lava操作
24 // lava.css
25 // lava操作
26 // lava.css
27 // lava操作
28 // lava.css
29 // lava操作
30 // lava.css
31 }
```

为了更方便以后进行维护修改，我们使用一个变量来替换 20 这个常量。若使用常量，修改的时候必须要修改两处，而且还容易发生误改的情形。所以，这里我们使用 gap 这个变量，将其赋值为 20，而后替换掉常量。

```
$current_item = $nav.find(':focus');
$lava = $('<li class="lava">');
// 设置$lava的基准元素$nav及调整<a> z-index的高度
$nav.css('position', 'relative')
    .find('a').css({
        position: 'relative',
        zIndex: 1
});
// 在此处更改gap的值即可
gap = 20;

// $lava操作及样式
$lava.css({
    position: 'absolute',
    top: $current_item.position().top - gap/2,
    left: $current_item.position().left,
    width: $current_item.outerWidth(),
    height: $current_item.outerHeight() + gap,
    backgroundColor: '#eee'
}).appendTo($nav.find('ul'));

// 当$nav的li发生鼠标移上/获得焦点事件时调用处理函数
$nav.find('li')
    .bind('mouseover focusin', function() {
        //处理函数代码
    })
    .bind('mouseout focusout', function() {
        //处理函数代码
    })

```

事件监听与处理

当鼠标指针移动到 \$nav 的 li 上，或按 Tab 键使其获得焦点时，li 必须能够感知到这些事件，并且调用相应的函数进行处理。下面我们就为 li 编写事件感知与处理代码。首先编写事件感知代码，分为两种事件：一是当鼠标指针移动到 \$nav 的 li 上，或按 Tab 键使其获得焦点时发生的事件；二是当鼠标指针移出 li 或按 Tab 键使其失去焦点时发生的事件。

```
$nav.find('li')
    .bind('mouseover focusin', function() {
        //处理函数代码
    })
    .bind('mouseout focusout', function() {
        //处理函数代码
    })
```

下面我们分别为两种事件编写处理代码，一种事件是 li 获得焦点，另一种事件是 li 失去焦点。首先编写当 li 获得焦点这一事件发生时的处理代码。在处理代码中，我们会使用 jQuery 的 .animate() 方法将 \$lava 移动到鼠标指针所在的 li 位置上。

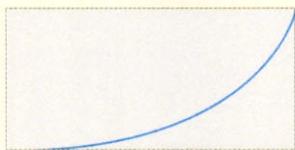
在 jQuery 的 .animate() 方法内，首先获取鼠标指针所在位置的 li 的 left，而后将设置为 \$lava 的 left；然后将 li (鼠标指针所在位置的 li) 的宽度设置为 \$lava 的宽度，代码编写如下：

```
$nav.find('li')
.bind('mouseover focusin', function() {
    $lava.animate({
        left: $(this).position().left,          向$lava设置动画
        width: $(this).outerWidth()            将$lava的left设置为li的left
    }, {                                    将$lava的width设置为li的width
        duration: 400,                      设置动画持续时间为400ms ( 0.4秒 )
        easing: 'easeInOutElastic',          设置动画Easing曲线为弹性移动
        queue: false                        设置动画队列为false
    })
})
```

查看代码，可以发现 .animate() 的使用方式与以前有所不同。在 animate() 的括号内有两个花括号 ({})，第 2 个花括号用于设置动画效果，里面有 3 个属性。第 1 个属性 duration 用于设定动画的持续时间，单位是毫秒 (1/1000 秒)；第二个属性 easing 用于指定动画运动方式；第三个属性 queue 用于设定动画队列。关于 jQuery Easing 插件的更多知识，请访问 <http://gsgd.co.uk/sandbox/jquery/easing/> 进行学习。在这里我们设置 easing 为 easeInOutElastic。

小知识 ↗ jQuery Easing 插件

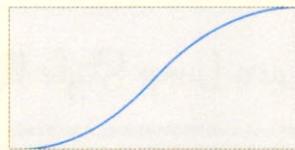
jQuery Easing 是一个动画效果扩展插件。jQuery 的自定义动画函数 .animate() 有 4 个参数，其中 easing 参数用于指定要使用的效果，它有两个默认的效果，若需要使用更多效果，就要插件支持了，常常使用的插件是 jQuery Easing。在 jQuery Easing 中设置了 easeInElastic、easeOutElastic 和 easeInOutElastic 等 31 种不同的效果，大体上分为 easeIn、easeOut 和 easeInOut 三大类。



easeIn- 开始移动慢，而后慢慢加速



easeOut- 开始移动快，而后慢慢减速



easeInOut- 前半段从 0 开始加速，后半段减速到 0

理解了上面最基本的 3 种效果后，再添加上下表中的关键字，就形成了各种效果的名称。

关键字	全称	含义	关键字	全称	含义
Quad	Quadratic	二次曲线	Expo	Exponential	指数曲线
Cubic	Bounce	立方体，立方	Circ	Circular	圆形曲线
Quart	Quartic	四次方程	Elastic	Elastic	弹力
Quint	Quintic	五次方程	Back	Back	后
Sine	Sinusodial	正弦曲线	Bounce	Bounce	弹跳

简言之，在基本 easing 关键字后添加上这些数学关键字，即形成各种效果的名称。对于一种缓动效果，其总体的运动方式由名称的前半部分决定，具有运动根据后面数学名称的不同，或舒缓，或急速变化，或弹跳运动。各种效果对应的效果图，可以访问 MC Tween:It saves the world (<http://2.gp/p3hu>,<http://hosted.zeh.com.br/mctween/animationtypes.html>) 页面。

Keyword	EaseIn	EaseOut	EaseInOut
Quad	easeInQuad	easeOutQuad	easeInOutQuad
Cubic	easeInCubic	easeOutCubic	easeInOutCubic
Quart	easeInQuart	easeOutQuart	easeInOutQuart
Quint	easeInQuint	easeOutQuint	easeInOutQuint
Sine	easeInSine	easeOutSine	easeInOutSine
Expo	easeInExpo	easeOutExpo	easeInOutExpo
Circ	easeInCirc	easeOutCirc	easeInOutCirc
Elastic	easeInElastic	easeOutElastic	easeInOutElastic
Back	easeInBack	easeOutBack	easeInOutBack
Bounce	easeInBounce	easeOutBounce	easeInOutBounce

最后一个属性 queue，用于指定动画执行的顺序。一般而言，动画是顺次执行的，但是当我们把 queue 属性设置为 false 之后，多个动画会同时执行。在这里将 queue 设置为 false 后，当用户反复将鼠标指针移入移出 li 元素时，动画也只执行一次。这与第 4 章中使用的 .stop() 方法的功能类似，但 .stop() 方法用于停止当前动画，执行新动画，而把 queue 设置为 false 后，不会等待当前动画结束，而是与新动画同时运行。保存代码，在浏览器中查看页面，可以发现 \$lava 在进行弹性运动。

```

    // 当鼠标到某一个鼠标移上/获得焦点事件时调用处理函数
    $nav.find('li').bind('mouseover focusin', function() {
        // 调用MouseOver或FocusIn事件时执行的代码
        $lava.animate({
            left: $(this).position().left,
            width: $(this).outerWidth()
        }, {
            duration: 400,
            easing: 'easeInOutElastic',
            queue: false
        });
    });
    // 处理鼠标移出或失去焦点事件时执行的代码
    $nav.find('li').bind('mouseout focusout', function() {
        // 调用MouseOut或FocusOut事件时执行的代码
    });
  
```

小知识 再谈 jQuery 的 animate() 方法

在第 4 章中我们已经讲解过 jQuery 的 animate() 方法的用法，在这里我们讲解它的另外的用法。在新的用法中，我们要给 animate() 方法传递两个参数，第一个参数是 CSS 样式属性及其值的集合，第二个参数是包含时间、easing、回调函数和队列等的集合。

例) `jQuery('p:last').animate({'color': 'tan', 'font-size': '22px'}, {duration: 400});`

CSS 属性及其值

动画的额外选项

动画的额外选项是可选的，而且可以是多个选项，它们之间使用逗号进行分割。

例) `jQuery('p:last').animate({'color': 'tan', 'font-size': '22px'}, {duration: 400, easing: 'easein', complete: function() { ... }});`

在额外选项中，我们可以设置 Queue，指示是否在效果队列中放置动画。我们可以把队列（Queue）看作是已注册动画的待机空间。当与 jQuery Effect 方法链接后，各个动画被注册到 Queue 中，注册的动画会按照注册顺序依次运行。若不想根据次序执行，同时并发执行时，只要将 Queue 设为 false 就可以了。更详细的内容，请访问 <http://api.jquery.com/animate> 页面进行了解学习。

例) `jQuery('p:last').animate({'color': 'tan', 'font-size': '22px'}, {queue: false});`

接下来，我们开始编写 li 失去焦点时的处理代码。失去焦点是指鼠标指针移出 li 或按 Tab 键使 li 失焦。短暂的停顿后，重新返回到 \$current_item（当前激活的项目）即可。在这里，我们使用 setTimeout() 函数，它用于在指定的毫秒数后调用函数或计算表达式。它有两个参数，一个是要执行的代码字符串，一个是以毫秒为单位的时间间隔。调用该函数后，经过了指定的时间间隔，代码就会被执行，且只执行一次。

```
.bind('mouseout focusout', function() {
    setTimeout(function() {
        $lava.animate({
            left: $current_item.position().left,
            width: $current_item.outerWidth()
        }, 200);
    }, 2000);
});
```



创建命名空间 (Namespace)

至此，我们已经将控制样式的脚本编写完成了，但是这些代码不利于后期维护。为了方便后期维护，我们要创建 JavaScript 命名空间对象，对它们进行管理。首先我们创建一个 options 命名空间对象，而后将一些需要经常修改的值放到里面，代码如下：

```
options = {  
    gap: 20,  
    speed: 400,  
    easing: 'easeInOutElastic',  
    reset: 2000  
}
```

将object literal保存到options变量
设置options.gap为20
设置options.speed为400
设置options.easing为ease InOutElastic
设置options.reset为2000

小知识 JavaScript 的命名空间

关于命名空间，我们可以将其简单地理解为“区域的名称”。假如我们要查找“居民中心”，在搜索引擎的搜索框中输入“居民中心”，进行搜索，那么搜索引擎会列出韩国的所有居民中心。在这里，“韩国”就是一个命名空间，说得更准确些，它是一个全局命名空间（Global Namespace）。全局命名空间一般可以省略。

居民中心

但是韩国所有的居民中心范围太大了，我们想更具体一些。所以在搜索框中，我们在居民中心前添加上地区名称，比如“吉音洞居民中心”，这样搜索到的居民中心就仅限制在“吉音洞”这一个地区的。这里的地区名称“吉音洞”就是一个局部命名空间（Local Namespace）。由于我们搜索的是吉音洞地区内的居民中心，因此在搜索结果中就不会列出其他地区的居民中心。

吉音洞居民中心

下面让我们一起学习一下 JavaScript 中的命名空间。在客户端（指网页浏览器）的 JavaScript 中，全局命名空间是 Window 对象。其实，在客户端的 JavaScript 中声明的变量与函数属于全局命名空间。在全局命名空间中声明的变量或函数，在内部的其他命名空间可以访问或修改。

由于全局命名空间的变量或函数能够被修改或重定义，因此我们不建议在全局命名空间中声明变量或函数。不然，当团队的其他成员在修改已声明的变量或函数名称时，引用相关变量与函数的代码就无法正常工作了。

为了防止发生意外，造成不必要的麻烦，我们通常采用创建新的命名空间办法，使相关的变量成为其内部属性。创建命名空间非常容易，全局对象就像命名空间一样，生成新的对象后将其用作命名空间，并且向已生成的对象添加属性，灵活运用，可以有效地防止与全局命名空间中的属性发生冲突。

重新修改代码，将相应数值替换为 options 命名空间对象的属性，代码即可正常工作。关于代码修改，请参考下图。

```

    // $lava操作及样式
    $lava.css({
        position: 'absolute',
        top: $current_item.position().top - options.gap/2,
        left: $current_item.position().left,
        width: $current_item.outerWidth(),
        height: $current_item.outerHeight() + options.gap,
        backgroundColor: '#eee'
    }).appendTo($nav.find('ul'));
}

// 当$nav的li发生鼠标移上/获得焦点事件时调用处理函数
$nav.find('li')
.bind('mouseover focusin', function() {
    // 发生MouseOver或FocusIn事件时执行的代码
    clearTimeout(reset);
    $lava.animate({
        left: $(this).position().left,
        width: $(this).outerWidth()
    }, {
        duration: options.speed,
        easing: options.easing,
        queue: false
    });
})
.bind('mouseout focusout', function() {
    // 发生MouseOut或FocusOut事件时执行的代码
    reset = setTimeout(function() {
        $lava.animate({
            left: $current_item.position().left,
            width: $current_item.outerWidth()
        }, options.speed);
    }, options.reset);
});

```

修改代码使用options
各属性值取代具体数值

制作插件

下面我们开始把编写好的代码制作成 jQuery 插件，方便以后重复使用。换言之，在这部分中我们将编写好的代码制作成 lava lamp 插件，便于重用。

首先创建 jQuery 插件。在 js 文件夹中新建一个名称为 jquery.lavalamp.js 的文件，编写 jQuery 插件基本代码。JavaScript 支持原生类型（prototype）的继承，jQuery 是一个 JavaScript 库，因此它也支持继承。jQuery.fn 是 jQuery 对象的 prototype 的样式表现，以 jQuery.fn. 插件的形式对 jQuery 进行插件扩展。

```

/*
 * jquery.lavalamp.js - Begin
 */
/* jquery.lavalamp.js - 滑岩灯导航插件, 2013 © yahoo9.com */

// 选项设置
var options = {
    gap: 20,
    speed: 400,
    easing: 'easeInOutBlastic',
    reset: 2000
};

options.easeInQuad = easeInQuad;
options.easeOutQuad = easeOutQuad;
options.easeInCubic = easeInCubic;
options.easeOutCubic = easeOutCubic;
options.easeInQuart = easeInQuart;
options.easeOutQuart = easeOutQuart;
options.easeInQuint = easeInQuint;
options.easeOutQuint = easeOutQuint;
options.easeInSine = easeInSine;
options.easeOutSine = easeOutSine;
options.easeInExpo = easeInExpo;
options.easeOutExpo = easeOutExpo;
options.easeInCirc = easeInCirc;
options.easeOutCirc = easeOutCirc;
options.easeInElastic = easeInElastic;
options.easeOutElastic = easeOutElastic;
options.easeInBack = easeInBack;
options.easeOutBack = easeOutBack;
options.easeInBounce = easeInBounce;
options.easeOutBounce = easeOutBounce;

;(function($) {
    $.fn.lavalamp = function(options) {
        options = $.extend({
            // 选项
        }, options);

        return this.each(function() {
            // 待处理的代码
        });
    };
})(jQuery);

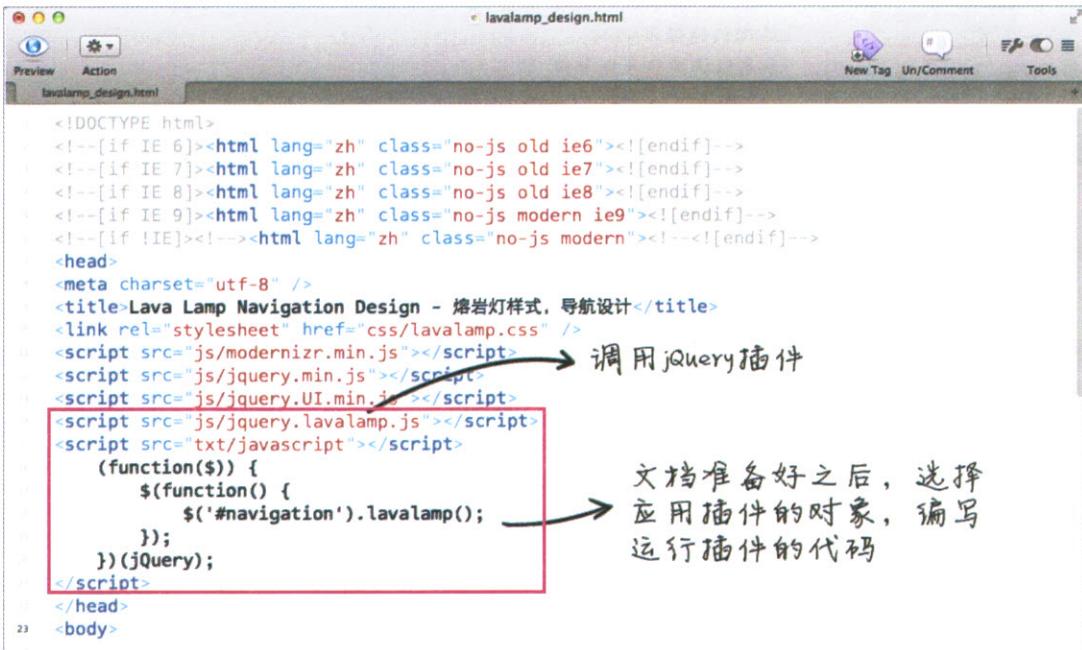
```

接下来，让我们一起了解一下创建 jQuery 插件的基本代码。在基本代码内部，添加上已经编写好的代码，即可完成插件的制作。

```
$ .fn.lavalamp = function() {  
    return this.each(function() {  
        //插件代码  
    });  
};
```

为jQuery.fn对象添加lavalamp函数
返回this以处理jQuery链
jQuery each()函数以每一个选配的元素作为上下文来执行一个函数

下面我们一起来学习一下插件的使用方法。在网页编辑器中打开 lavalamp_design.html 文档，修改代码如下。



```
<!DOCTYPE html>  
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->  
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->  
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->  
<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->  
<!--[if !IE]><!--><html lang="zh" class="no-js modern"><!--><![endif]-->  
<head>  
    <meta charset="utf-8" />  
    <title>Lava Lamp Navigation Design - 熔岩灯样式，导航设计</title>  
    <link rel="stylesheet" href="css/lavalamp.css" />  
    <script src="js/modernizr.min.js"></script>  
    <script src="js/jquery.min.js"></script>  
    <script src="js/jquery.UI.min.js"></script>  
    <script src="js/jquery.lavalamp.js"></script>  
    <script src="txt/javascript"></script>  
    (function($){  
        $(function(){  
            $('#navigation').lavalamp();  
        });  
    })(jQuery);  
    </script>  
</head>  
<body>
```

<script src="js/jquery.lavalamp.js"></script> 调用jQuery插件

...
\$('#navigation').lavalamp(); 网页文档准备完毕后，选择对象运行插件代码

接着，修改 jquery.lavalamp.js 文件。在 lavalamp.js 代码中，复制 options 部分，粘贴到 \$.extend() 内部。然后参考右面的图片，修改代码。

```
// 对象设置
var $nav = $('#navigation'),
    surrent_item = $nav.find('.focus'),
    $lava = $('li class="lava"/');

options = {
    gap: 20,
    bgColor: '#eee',
    speed: 400,
    easing: 'easeInOutElastic',
    reset: 2000
};

$(function($){
    $().lavalamp = function(options) {
        options = $.extend({
            gap: 20,
            bgColor: '#eee',
            speed: 400,
            easing: 'easeInOutElastic',
            reset: 2000
        }, options);

        return this.each(function(){
            // 按钮的滑块
            console.log(options);
        });
    };
})(jQuery);
```

从 lavalamp.js 文件中复制 options 后，粘贴至 jquery.lavalamp.js 文件的 \$.extend() 内部
编写 console.log() 语句，在 Firebug 的控制台窗口中查看 options 值

保存代码，运行 Firebug 插件，进入“控制台”，在日志记录窗口中，可以看到 options 对象的设置值被正常处理，即插件被正常调用，且工作正常。



使用插件的好处之一是可以方便地修改属性值，修改时不需要直接修改插件，只需要调用插件时，更改相应的属性值，插件即根据修改后的属性值运行，这为后期的维护带来极大的便利，减少后期维护的成本。

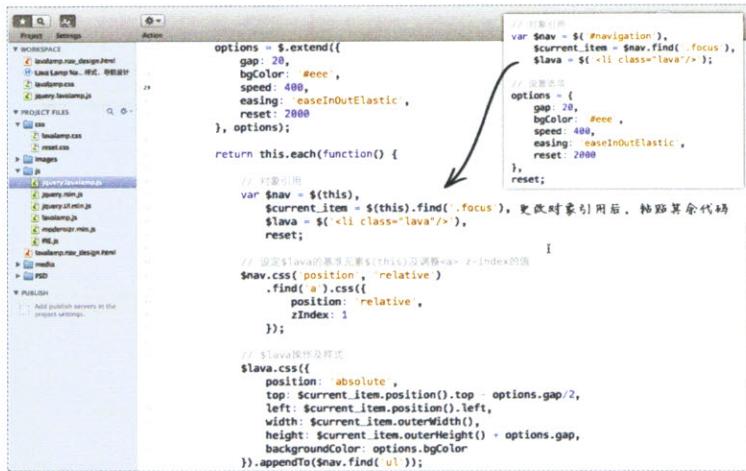
```
$('#navigation').lavalamp({
    gap: 15,
    bgColor: '#333'
});
```

向 navigation 元素应用 lavalamp 插件
在 lavalamp 插件内部使用 15 覆盖掉 options 的 gap 值
在 lavalamp 插件内使用 #333 覆盖 options 的 bgColor 值

保存文档，在 Firebug 的控制台窗口中，可以看到 options 值发生了改变。怎么样？使用 jQuery 插件没什么太大的难度吧？！

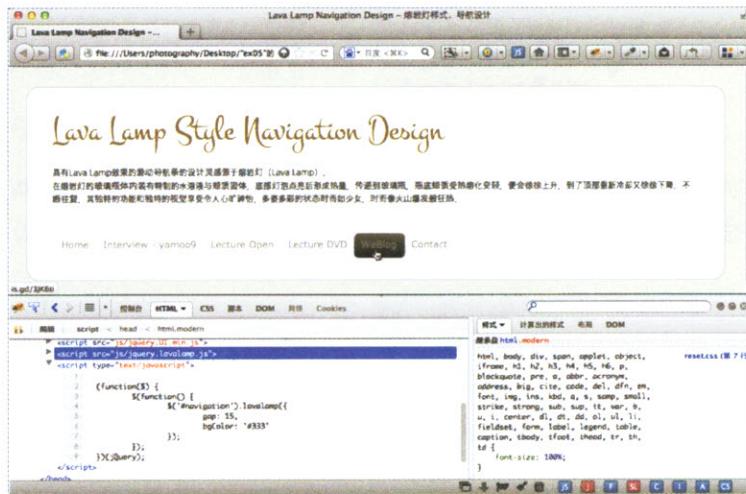


接下来，我们还要修改一下对象引用部分，其余代码直接复制粘贴即可。由于一开始就设想做成一个插件，编写代码时做了稍微调整，因此实际上需要修改的内容并不多。请参考下图，修改部分代码。



```
options = $.extend({
    gap: 20,
    bgColor: "#eee",
    speed: 400,
    easing: 'easeInOutElastic',
    reset: 2000
}, options);
return this.each(function() {
    // 对象引用
    var $nav = $(this),
        $current_item = $(this).find('.focus'),
        $lava = $('- 
'),
        reset;
    // 设置$nav的基准元素$(this)及滑移元素$nav->Index的底
    $nav.css('position', 'relative')
        .find('a').css({
            position: 'relative',
            zIndex: 1
        });
    // $lava操作及样式
    $lava.css({
        position: 'absolute',
        top: $current_item.position().top - options.gap/2,
        left: $current_item.position().left,
        width: $current_item.outerWidth(),
        height: $current_item.outerHeight() + options.gap,
        backgroundColor: options.bgColor
    }).appendTo($nav.find('ul'));
});
```

保存插件文件，在网页浏览器中查看页面，可以发现网页效果工作正常。



从 `lavalamp_design.html` 网页中分离出脚本代码，而后添加到 `navigation.js` 文件中，再在网页中调用该脚本文件即可。

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

调用js文件夹内的navigation.js文件

我们知道在旧版本的浏览器（IE 6、7、8）中无法使用 CSS3 新技术，网页的菜单项的圆角边框以及渐变都会出现相应的问题。

6 章 制作气泡动画按钮

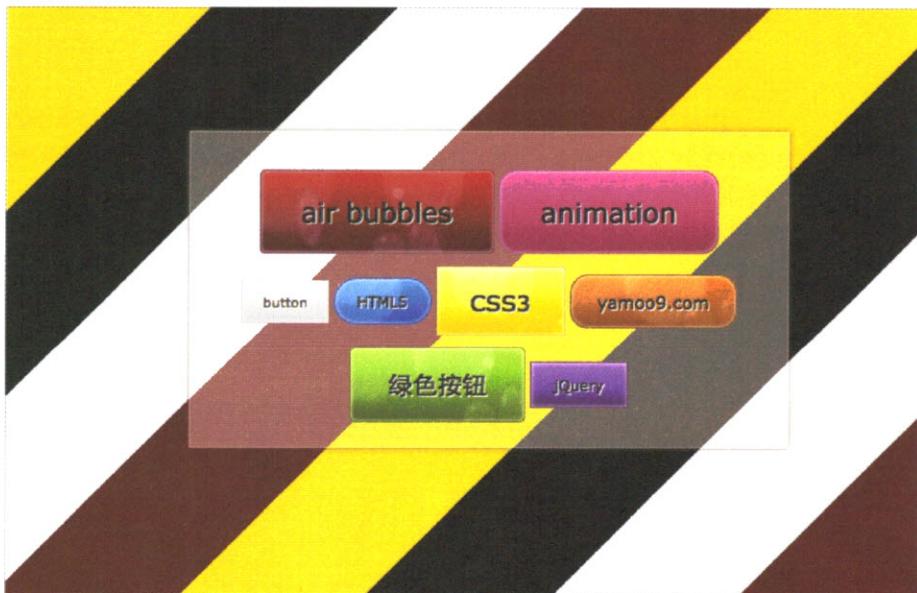
设计网页时，按钮是最常用的元素之一，它不仅能实现用户与网页间的交互操作，还能提高网站的整体格调，是网站美化的元素之一。设计优秀的按钮，不仅能够刺激用户的视觉与听觉，给用户带来美的感受，也能增强用户浏览网页的愉悦感。在本章，我们将学习制作一种气泡动画按钮，这种按钮不仅视觉效果酷炫，还带有声音，是非常棒的按钮设计。

设计效果预览

在本章我们一起制作气泡动画按钮，其设计灵感源自盛有碳酸饮料的玻璃杯。在盛有碳酸饮料的玻璃杯中，气泡不断从玻璃杯的底部向上浮起，效果非常酷炫。

制作时，我们将按钮设计成装有碳酸饮料的容器，当鼠标指针移动到容器上时，里面就会涌现大量气泡，并向四下漂动。与此同时，也会有声音传出来。也就是说，制作气泡动画按钮时，我们要同时实现两种效果，一种效果是气泡向四下漂动，另一种效果是出现声音。

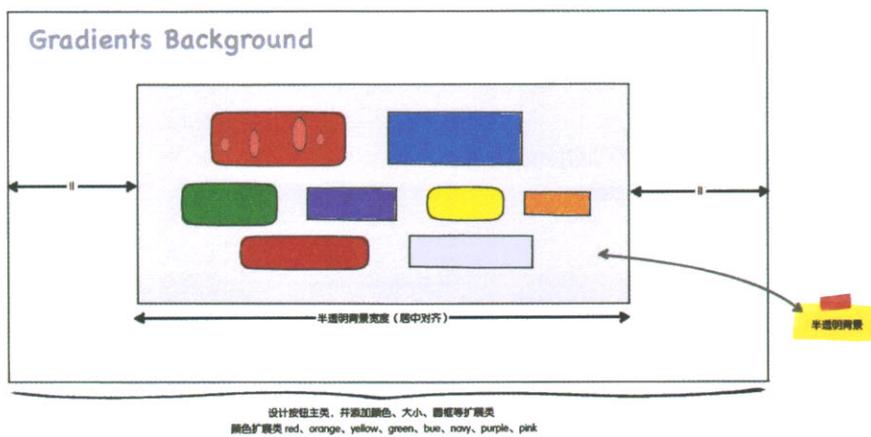
制作中，除了气泡外，其余所有图片均使用 CSS3 制作。使用 CSS3 而不是 Photoshop 制作图片，对大家而言那将会是一个非常奇妙的体验。在设计网页背景时，使用的七彩条也是使用 CSS3 技术实现的，最终的制作效果如下图所示。



CHAPTER 6

网页布局设计

首先使用 Balsamiq Mockups 软件将头脑中关于按钮的构思描绘出来，如下图所示。



在 Photoshop 中制作气泡图片。先新建一个形状图层，创建气泡体，再添加各种效果，表现出立体感，气泡的高光与反射光也在单独的形状图层上创建，表现气泡漂动的生动感觉。也就是说先在形状图层上创建出气泡体，添加各种立体效果后，再添加高光与反射光图层，表现气泡浮动的效果。制作气泡时，需要注意各个气泡的大小与形状要有所不同，这样制作出的效果才会自然、逼真。在 Photoshop 中再创建一个圆形的气泡，然后使用智能对象功能，使其自然地排列，再向一些气泡应用模糊效果，以使其更加自然、逼真。

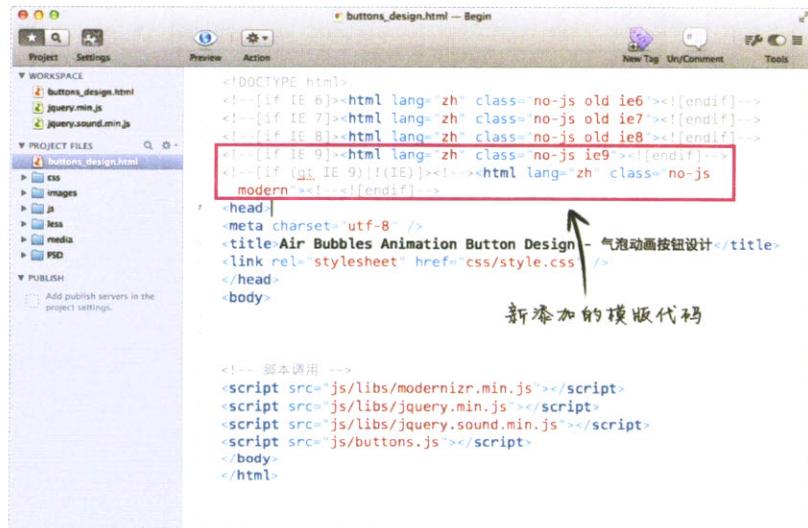
编写 HTML5 代码

下载例题源文件，解压缩后，进入 \ex_06\Begin 文件夹中，在网页编辑器中打开 buttons_design.html 文档。它是一个 HTML5 文档，在文档最顶部的注释中的代码用于检测用户的浏览器版本与类型。在 <head> 与 <body> 中，分别调用了 5 个外部文件：css/style.css、js/buttons.js、Modernizr、jQuery 和 sound 插件。关于 sound 插件，请参考第 3 章第 85 页中的内容。

考虑到对 IE 10 的支持，我们在文档中添加几行代码。首先添加检测 IE 9 的代码，而后在其下添加检测 IE 10 以及非 IE 浏览器的 modern 类，代码如下。

```
<!--[if IE 9]><html lang="ko" class="no-js ie9"><![endif]-->
<!--[if (gt IE 9)!!(IE)]><!--><html lang="ko" class="no-js modern"><!--<![endif]-->
```

首先使用 <div id="buttons_wrap"> 元素表示整个按钮区域，而后使用多个 <a> 元素表示各个按钮，所有 <a> 元素的 class 值全部设置为 button，href 属性全部设置为 "#"，在实际制作中，请使用实际要链接的地址代替。整理代码如下：



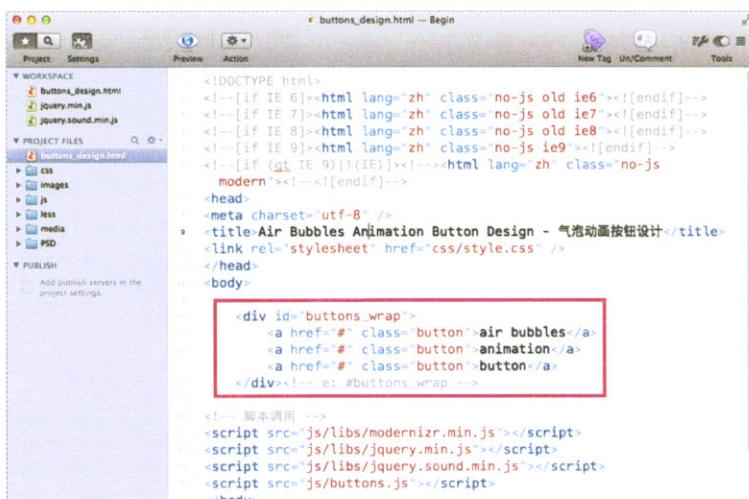
```
<div id="buttons_wrap">
  <a href="#" class="button">air bubbles </a>
  <a href="#" class="button">animation </a>
  <a href="#" class="button">button </a>
</div>
```

小知识 <a> 元素与按钮元素

在这里，我们使用 `<a>` 元素来表示按钮，当然也可以直接使用按钮元素，如 `<input type="button" value="button"/>` 或 `<button type="button">button</button>`。

在所有的 `<a>` 元素中我们都为它们设置了 `class` 属性，并统一设置值为 `button`。下面我们会继续根据各个按钮的特点，向 `class` 属性中添加扩展名，方便更有效地控制各个按钮。比如，添加颜色名称（红色按钮、绿色按钮、蓝色按钮）、大小（小按钮、普通按钮、大按钮）和圆角边框（方角按钮、圆角按钮）等。

关于向 `class` 中添加颜色扩展名，可以使用的几种颜色名称有 `red`、`yellow`、`orange`、`green`、`blue`、`deepblue`、`purple`、`pink`、`gray` 和 `dark`；大小扩展名，可以使用的几种形式有 `xx-small`、`x-small`、`small`、`big`、`x-big` 和 `xx-big`；圆角边框扩展名，可以选用的几种形式有 `round`、`rounder` 和 `roundest` 等。向各个 `<a>` 元素的 `class` 属性添加扩展名，如下所示。



The screenshot shows the Dreamweaver interface with the code editor open. The code editor displays the following HTML and CSS:

```
<!DOCTYPE html>
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html lang="zh" class="no-js ie9"><![endif]-->
<!--[if gt IE 9]!><html lang="zh" class="no-js modern"><![endif]-->
<head>
<meta charset="utf-8" />
<title>Air Bubbles Animation Button Design - 气泡动画按钮设计</title>
<link rel="stylesheet" href="css/style.css" />
</head>
<body>
<div id="buttons_wrap">
<a href="#" class="button red small">air bubbles</a>
<a href="#" class="button blue big round">animation</a>
<a href="#" class="button">button</a>
</div><!-- e: #buttons_wrap -->
<!-- 脚本调用 -->
<script src="js/libs/modernizr.min.js"></script>
<script src="js/libs/jquery.min.js"></script>
<script src="js/libs/jquery.sound.min.js"></script>
<script src="js/buttons.js"></script>
</body>
```

```
<a href="#" class="button red small">air bubbles</a>
<a href="#" class="button blue big round">animation</a>
<a href="#" class="button">button</a>
```

设置为红色小按钮

设置圆角蓝色大按钮。

至此，HTML 文档结构基本完成。本章我们学习的重点是样式的控制，文档结构编写得简单一些就可以了。保存代码，在浏览器中查看页面，如图所示。



小知识 重设浏览器样式

现在有很多浏览器品种，每种浏览器都会对 CSS 的选择器默认一些数值，但并不是所有的浏览器都使用一样的数值，也就是说每种浏览器提供的默认样式是不同的。这可能会导致元素的同一样式在不同的浏览器中拥有不同的显示，浏览器的这些默认样式往往给我们带来麻烦，影响开发效率。解决方法就是将浏览器提供的默认样式覆盖掉，通过重置标签的 CSS 属性，进行统一定义，就可以产生相同的显示效果。

在众多重置方法中，最著名的、使用最广泛的就是 CSS 专家 Eric Meyer 提供的 reset.css，它会对 HTML 元素的空间、边框、字体大小等进行重置，访问 <http://is.gd/lU61tH> 网站，下载使用即可。在本章中，我们会使用它。



有些专家建议不要对浏览器提供的默认样式进行重置。因为在重定义并覆盖浏览器的默认样式后，读取网页时效率会大大下降。尽管如此，还是有很多设计者选择重置浏览器的默认样式，毕竟——查找各浏览器不同的表现部分进行修改是件非常麻烦的事儿。笔者建议您对浏览器默认的样式进行重置，但应遵循样式最小重置原则。Eric Meyer 提供的浏览器重置样式非常精简，实现了对浏览器样式的最少重置，非常有效率。

如果想知道各个浏览器是如何应用不同样式的，可以从 meiert.com (<http://is.gd/cweVSA>) 下载各浏览器基本样式文档，进行查看学习。

Example Files

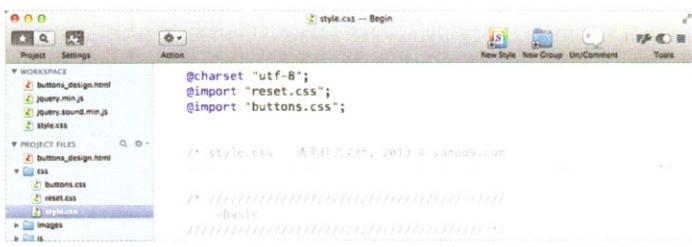
The following is a list of default style sheets I compiled over recent years (except for the Safari example; [thank you, Martin](#)). I've occasionally looked for and compared user agent style sheets (.css); due to the fact that Firebird, Firefox, and Co. are all based on the same layout engine, they're also quite similar, if not identical.

- Chrome
- Firebird 0.7
- Firefox 0.8
- Firefox 0.9
- Firefox 2.0.0.6
- Firefox 2.0.0.12
- Firefox 3.0b3
- Firefox 3.0.1
- Firefox 3.0.8
- Firefox 3.6.13
- Flock 0.9.0.2
- Flock 1.2.4
- Flock 1.2.7
- Internet Explorer 6
- Internet Explorer 7
- Internet Explorer 8
- Internet Explorer 9
- Konqueror 3.2.0
- Mozilla 1.0.1
- Mozilla 1.5
- Navigator 6.1
- Navigator 7.1
- Navigator 8.1

单击相应链接，下载各浏览器基本样式文档

编写 CSS3 样式表

在 css 文件夹中找到 style.css 文件，使用网页编辑器打开它，可以看到有两条 @import 语句，分别用于导入 reset.css 与 buttons.css 两个样式表。其中 buttons.css 样式表用于控制按钮的各种样式。



控制 body 样式

下面我们开始使用 CSS3 在 style.css 中创建文档背景。在 Basic 注释语句下，向 html、body 和 a 添加基本样式控制代码。

```
html, body { height: 100%; }
body { background: #1c1c1c; }
a:link, a:visited { color: #4a4a4a; text-decoration: none; }

a:hover, a:active { color: #7b7b7b; }
```

将body高度设为浏览器窗口的高度
设置body背景色
为a元素的:link, :visited状态设置颜色，添加到文本的修饰
为a元素的:hover, :active状态设置颜色

向 body 添加彩虹渐变

接着，我们向 body 元素添加彩虹渐变背景，即向 body 元素应用渐变背景图片的代码。与第 5 章学到的一样，添加渐变样式时，我们将使用 CSS3 渐变中的线性渐变 (Linear-Gradient)。如下代码所示，①表示渐变类型，②表示渐变角度，③表示颜色值及其位置。



```
background-image: linear-gradient(45deg, #ff0000 0%, #ffb600 11%, #fff600 22%, #a5ff00 33%,  
① ② ③  
#00a9ff 44%, #0400ff 55%, #8a00fc 66%, #ff00e9 77%, #ff0059 88%, #ff0000 100%);
```

在两种颜色之间，我们再添加一种颜色，颜色名称为左侧颜色，位置为右侧颜色的位置。通过使用这种方法，能够很容易地将各种颜色分开。



```
background-image: linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%,  
#ffb600 22%, #fff600 22%, #fff600 33%, #a5ff00 33%,  
#a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0400ff 55%,  
#0400ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%,  
#ff00e9 88%, #ff0059 88%, #ff0000 100%);
```

可能你会觉得使用 CSS 制作彩虹条纹时代码非常复杂，如果你会 Photoshop，可能会直接使用 Photoshop 制作这种效果，而不会使用这么复杂的代码。其实，还有更简单的方法来制作这种效果，那就是使用 ColorZilla 渐变生成器 (<http://is.gd/et2H1U>)，使用它制作渐变就像使用 Photoshop 的渐变工具一样轻松简单。使用 ColorZilla 渐变生成器，可以快速轻松地创建出各种不同的渐变效果，而且它能为我们自动生成 CSS 代码。创建好要使用的渐变效果后，与之对应的 CSS 代码也会生成，我们只要将生成的 CSS 代码粘贴到我们自己的样式表中使用就可以了。

若想创建彩虹条纹，而不是彩虹渐变，则需要针对同一种颜色创建两个颜色拾取器，一个颜色拾取器位于颜色左侧，另一个位于右侧。然后将第二种颜色的左侧的颜色拾取器左移到第一张颜色的右拾取器位置上，以使左右两种颜色明显地区分开来。请参考下图，制作彩虹条纹。

制作好彩虹条纹后，在 Name 输入名称后，单击 Save 按钮，将其保存到 Presets 列表中。然后复制 CSS 区域中的 CSS 代码，粘贴到 style.css 样式表中。



如图所示，使用 ColorZilla 渐变生成器产生的 CSS 代码看上去非常复杂，因为这些代码综合考虑了对各种浏览器的兼容性。这些代码表面看似复杂，但如果仔细分析，就会发现它们只是某个代码段的重复而已。由于需要在每个代码段之前针对不同浏览器添加不同的前缀，因此代码看上去显得比较复杂。在我们刚刚粘贴过来的彩虹条代码中，彩虹条是竖直分布的，接下来我们要将它们更改成斜线分布的，方法非常简单，只需要将 left 更改为 45deg 就可以了。但在 -webkit-gradient() 代码行中，应该设置为 left top、right bottom，这是为了实现对使用旧版 webkit 引擎的浏览器有更好的兼容性，也是在渐变编写标准出现前的编写形式。

```

background: #ff0000;
background: -moz-linear-gradient(left, #ff0000 0%, #ff0000 11%, #fb600 11%, #fb600 22%, #fb600 33%, #a5f00 33%, #a5f00 44%, #00a9f1 44%, #00a9f1 55%, #0400f 55%, #0400f 66%, #8a00c 66%, #8a00c 77%, #ff0e9 77%, #ff0e9 88%, #ff059 88%, #ff0000 100%);
background: -webkit-gradient(left, left top, right top, color-stop(0%,#ff0000), color-stop(11%,#ff0000), color-stop(11%,#fb600), color-stop(22%,#fb600), color-stop(33%,#fb600), color-stop(33%,#a5f00), color-stop(44%,#a5f00), color-stop(44%,#a5f00), color-stop(44%,#00a9f1), color-stop(55%,#00a9f1), color-stop(55%,#0400f), color-stop(66%,#8a00c), color-stop(77%,#8a00c), color-stop(77%,#ff0e9), color-stop(88%,#ff0e9), color-stop(88%,#ff0e9), color-stop(100%,#ff0000));
background: -webkit-linear-gradient(left, #ff0000 0%,#ff0000 11%,#fb600 11%,#fb600 22%,#fb600 33%,#a5f00 33%,#a5f00 44%,#00a9f1 44%,#00a9f1 55%,#0400f 55%,#0400f 66%,#8a00c 66%,#8a00c 77%,#ff0e9 77%,#ff0e9 88%,#ff059 88%,#ff0000 100%);
background: -o-linear-gradient(left, #ff0000 0%,#ff0000 11%,#fb600 11%,#fb600 22%,#fb600 33%,#a5f00 33%,#a5f00 44%,#00a9f1 44%,#00a9f1 55%,#0400f 55%,#0400f 66%,#8a00c 66%,#8a00c 77%,#ff0e9 77%,#ff0e9 88%,#ff059 88%,#ff0000 100%);
background: -ms-linear-gradient(left, #ff0000 0%,#ff0000 11%,#fb600 11%,#fb600 22%,#fb600 33%,#a5f00 33%,#a5f00 44%,#00a9f1 44%,#00a9f1 55%,#0400f 55%,#0400f 66%,#8a00c 66%,#8a00c 77%,#ff0e9 77%,#ff0e9 88%,#ff059 88%,#ff0000 100%);

Color format: hex : Comments

```

The screenshot shows the ColorZilla extension's interface. On the left, there's a project tree with files like 'buttons_design.html', 'jquery.js', 'jquery.sound.min.js', 'style.css', 'reset.css', 'style.less', 'images', 'js', 'libs', 'less', 'media', and 'PSD'. In the center, there's a preview area with some text and a color gradient. On the right, the main content area displays the generated CSS code for a button's background:

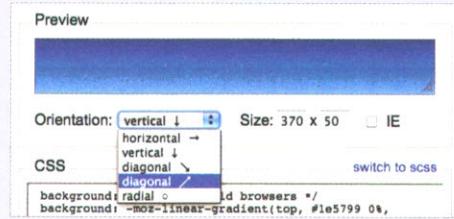
```

    /* Gradient */
    body {
        background: #1c1c1c;
        background-image: -moz-linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%, #ffb600 22%, #ffff00 22%, #ffff00 33%, #a5ff00 33%, #a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0408ff 55%, #0408ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%, #ff00e9 88%, #ff00e9 88%, #ff00e9 100%);
        background-image: -webkit-linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%, #ffb600 22%, #ffff00 22%, #ffff00 33%, #a5ff00 33%, #a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0408ff 55%, #0408ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%, #ff00e9 88%, #ff00e9 88%, #ff00e9 100%);
        background-image: -o-linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%, #ffb600 22%, #ffff00 22%, #ffff00 33%, #a5ff00 33%, #a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0408ff 55%, #0408ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%, #ff00e9 88%, #ff00e9 88%, #ff00e9 100%);
        background-image: -ms-linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%, #ffb600 22%, #ffff00 22%, #ffff00 33%, #a5ff00 33%, #a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0408ff 55%, #0408ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%, #ff00e9 88%, #ff00e9 88%, #ff00e9 100%);
        filter: progid:DXImageTransform.Microsoft.gradient (startColorstr='#ff0000', endColorstr='#ff0000', GradientType=1);
        background-image: linear-gradient(45deg, #ff0000 0%, #ff0000 11%, #ffb600 11%, #ffb600 22%, #ffff00 22%, #ffff00 33%, #a5ff00 33%, #a5ff00 44%, #00a9ff 44%, #00a9ff 55%, #0408ff 55%, #0408ff 66%, #8a00fc 66%, #8a00fc 77%, #ff00e9 77%, #ff00e9 88%, #ff00e9 88%, #ff00e9 100%);
    }

```

小知识 在 ColorZilla 漐变生成器中，更改渐变方向

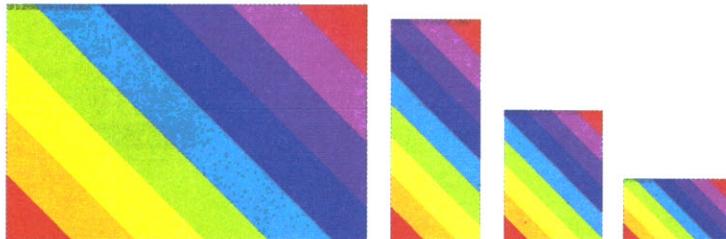
在 ColorZilla 漐变生成器中，在 Preview 区域中有一个 Orientation 选项，该选项用于控制渐变的方向。它提供了 5 种方向供用户选择使用，分别为水平向右、垂直向下、左上到右下、左下到右上、圆形。在实例中我们要求彩虹条纹是从左上到右下分布的，而默认时彩虹条是垂直向下的，所以我们需要将垂直向下的彩虹条逆时针旋转 45 度，即可得到我们需要的彩虹条纹。因此在 Orientation 选项，选择“左下到右上”。



保存代码，在浏览器中查看页面，可以发现彩虹条已经呈现出从左下到右上的分布了。



调整浏览器窗口大小，可以发现随着窗口大小的变化，彩虹条的宽度也会自动改变。到这里，使用 CSS3 制作彩虹渐变条的工作就完成了。



但是这里又出现了一个问题。当需要修改渐变颜色或形状的时候，我们应该怎么办呢？因为代码量非常大，所以修改的工作量也非常大。再使用 ColorZilla 渐变生成器重新生成代码又显得十分繁琐。若 CSS 能像 JavaScript 一样，能够分析渐变代码中共同的颜色以及位置等属性，然后集中起来进行统一管理，这样在后期修改维护的时候会非常方便。

有没有好的解决方法呢？有，就是使用 LESS。LESS 是一种动态样式语言，它能赋予 CSS 动态语言的特性，如变量、继承、运算和函数等。下面我们将使用 LESS 来重写 CSS 样式表。

从 CSS 代码中提取重复代码，使用 LESS 重写

首先访问 lesscss.net 官方网站 (<http://lesscss.net>)，单击“Download less.js”按钮，下载 less.js 文件。

The screenshot shows the official website for LESS. At the top, there's a banner with the LESS logo and the text "一种动态样式语言". Below it, there's a "Write some LESS:" input field containing some LESS code. To the right of the input field is a "Compile to CSS:" output field showing the generated CSS. Below these fields is a "Variables" section with a note about using variables to reduce repetitive code. At the bottom, there's a "buttons_design.html" preview area showing a button design with some LESS syntax.

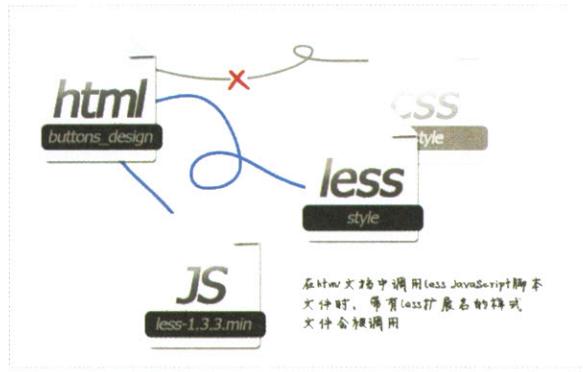
打开 buttons_design.html 文件，添加调用 less.js 的代码（less.js 文件名要与下载的文件名保持一致），然后使用注释将调用 style.css 文件的代码注释掉，并添加调用 style.less 文件的代码。

The screenshot shows a LESS editor interface with a project structure on the left and a code editor on the right. The code editor contains the following LESS code:

```
buttons_design.html - Begin
<html lang="zh">
  <head>
    <meta charset="utf-8" />
    <title>Air Bubbles Animation Button Design - 气泡动画按钮设计</title>
    <link rel="stylesheet/less" href="css/style.less" />
  </head>
  <body>
    <div id="buttons_wrap">
      <a href="#" class="button red small air_bubbles_a"></a>
      <a href="#" class="button blue big round animation_a"></a>
      <a href="#" class="button button_a"></a>
    </div>
  </body>
</html>

// 调用以less.css.org 下载的less.js文件
script src="js/libs/less-1.3.3.min.js" />
script src="js/libs/modernizr.min.js" />
script src="js/libs/jquery.min.js" />
script src="js/libs/jquery.sound.min.js" />
script src="js/buttons.js" />
```

下面我们使用一个关系图来表示各个文档之间的关系，关系图如下所示。buttons_design.html 文档调用 less-1.3.3.min.js 文件，使用 LESS 替换 CSS，less-1.3.3.min.js 文件会将 LESS 代码实时地转换为 CSS 代码。



在 style.css 文件中，除了 CSS3 渐变代码外，复制其余代码后，粘贴到 style.less 文件中。由于我们会使用 LESS 函数重写 CSS3 渐变代码，因此不需要复制它们。保存文件，在浏览器中查看页面，可以看到渐变背景消失不见了。

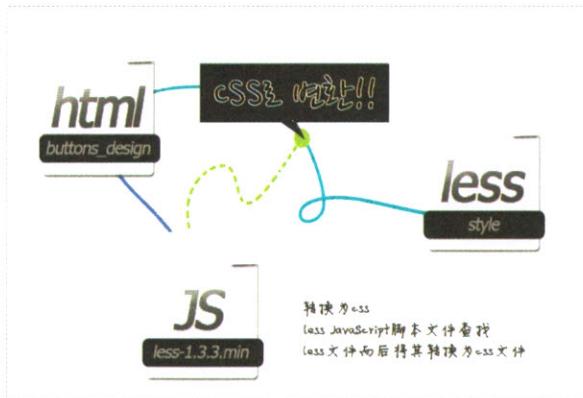
```
less.js 文件将 style.less  
文件动态转换为css代码
```

```
<link href="css/style.less" rel="stylesheet/less">  
style<file:///Users/photography/Desktop/%E8%80%9Cex06-%E8-90-E7-9A-84-%5-89-  
AF-E6-9C-AC-ex_0E-Begin-css-style" type="text/css">  
1 @import "file:///Users/photography/Desktop/%E2%80%9Cex06%E2%80%9DNE%9A%84%E5%89%AF  
2  
3 @import "file:///Users/photography/Desktop/%E2%80%9Cex06%E2%80%9DNE%9A%84%E5%89%AF  
4 @charset "utf-8";  
5 /* style.css - 通用样式文档, 2013 © yahoo9.com */  
6 /*-----  
7 //-----  
8 //-----  
9 //-----  
10 //-----  
11 html,  
12 body {  
13     height: 100%;  
14 }  
15 body {  
16     background-color: #1c1c1c;  
17 }  
18 a:link,  
19 a:visited {  
20     color: #404040;  
21     text-decoration: none;  
22 }
```

button...gn.html (第 10 行)

style.less 文件中的代码会动态生成 <style> 元素，less.js 会将 <style> 元素转换成 CSS 代码。动态改变的 LESS 代码会被添加到 buttons_design.html 文件中，调用外部 CSS 文件的路径也会发生改变。

因此应该更改使用 @import 命令调用 reset.css 与 buttons.css 的路径，即在路径前添加 “css/” ，保证调用正常进行。文档之间的关系如右图所示，从图中可以看出，less-1.3.3.min.js 文件其实就是一个将 LESS 文件转换为 CSS 文件的工具。



创建 LESS 函数

下面我们开始创建 LESS 函数，它是使用 LESS 的核心。访问 <http://2gp/p3hw>，查看 API，如下图所示。

The screenshot shows the LESS API documentation with the following content:

Parametric Mixins

LESS has a special type of ruleset which can be mixed in like classes, but accepts parameters. Here's the canonical example:

```
.border-radius (@radius) {  
  border-radius: @radius;  
  -moz-border-radius: @radius;  
  -webkit-border-radius: @radius;  
}
```

And here's how we can mix it into various rulesets:

```
#header {  
  .border-radius(4px);  
}  
.button {  
  .border-radius(6px);  
}
```

Parametric mixins can also have default values for their parameters:

```
.border-radius (@radius: 5px) {  
  设置@radius的初始值  
}
```

简单说明如下：① 函数名称前要有一个实心句号，② 小括号内是形式参数，③ 花括号内是需要反复执行的 CSS 代码。由于函数可以重复使用，因此我们能够使用它将复杂的代码生成易于维护的代码。

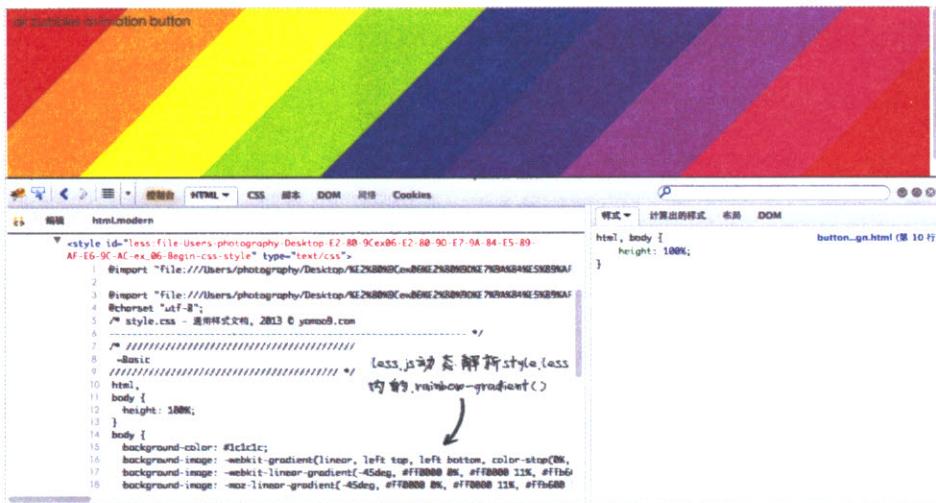
```
.rainbow-gradient (@参数1, @参数2, ...) {  
  ① /* 反复执行的CSS代码 */  
  ②  
  } ③
```

接着，继续在 style.less 中编写代码。我们将函数命名为 .rainbow-gradient，并在其后的小括号内放入彩虹渐变的角度与颜色。而后在函数内部编写需要反复使用的 CSS 代码，如下图所示。在小括号内，形式参数 @deg 用于设定角度，@c1~@c9 用于设置渐变中使用的颜色；在花括号中编写 CSS3 渐变代码，复制粘贴带有浏览器前缀的代码，如下图所示。

The screenshot shows the LESS editor with the following code:

```
body {  
  background-color: #1c1c1c;  
  .rainbow-gradient( ① 指用 rainbow-gradient() 函数  
    -45deg,  
    #ff0000, #ff6600, #ffff00, #00ff00, #008000, #ff0080, #ff0059 );  
};  
  
//LESS Functions  
.rainbow-gradient(  
  @deg,  
  @c1: red,  
  @c2: orange,  
  @c3: yellow,  
  @c4: green,  
  @c5: blue,  
  @c6: purple,  
  @c7: pink,  
  @c8: red,  
  @c9: red );  ② 初始化参数值，这些参数  
              在函数内部被使用  
  {  
    background-image:  
      -webkit-gradient(linear, left top, left bottom, color-stop(0%, @c1),  
      color-stop(11%, @c1), color-stop(11%, @c2),  
      color-stop(33%, @c2), color-stop(33%, @c3),  
      color-stop(44%, @c3), color-stop(44%, @c4),  
      color-stop(55%, @c4), color-stop(55%, @c5),  
      color-stop(77%, @c5), color-stop(77%, @c6),  
      color-stop(88%, @c6), color-stop(88%, @c7),  
      color-stop(100%, @c7));  
  }  
  background-image:  
    -webkit-linear-gradient(@deg,  
    @c1 0%,  
    @c1 11%; @c2 11%);  
  }
```

保存代码，在浏览器中打开页面，运行 Firebug，可以看到 style.less 文件经 LESS 函数处理后的结果。less.js 脚本会解析 style.less 文件，而后将参数与 CSS3 渐变代码混合，生成静态的 CSS 代码。



接下来，我们对代码略微做些改动。如果是使用静态 CSS 编写的代码，那么改动起来会非常费力。而使用 LESS 代码后修改起来就非常简单了。在 style.less 文件中，修改 .rainbow-gradient 代码，如下所示。只需要简单地修改几个参数，彩虹渐变条的颜色就全都改变了，修改非常简单轻松。

```
.rainbow-gradient (-243deg, gold, maroon, snow, black, gold, maroon, snow, black, gold);
```



控制 buttons_wrap 样式

下面我们开始为 #buttons_wrap 添加样式。首先将其定位方式设置为绝对定位，而后将其设置到画面中央，并且还要为它设置边框与背景。最后，使用 @wrap_width、函数 .box-shadow() 进行设置。

```

#buttons_wrap {
    position: absolute;
    width: @wrap_width;
    margin: 100px (100% - @wrap_width)/2 0;
    border: 1px solid #f8f8f8;
    border-color: rgba(237, 237, 237, 0.4);
    padding: 20px;
    background: #e7e7e7;
    background: rgba(237, 237, 237, 0.48);
    text-align: center;
    .box-shadow(0px, 0px, 4px, #787878);
}

选择#buttons_wrap
设置为绝对定位
将width设置为@wrap_width
设置上下外边距为10px，左右外边距，下外边距为0
将左右外边距设为(100%-@wrap_width)/2
可以将#buttons_wrap始终放置到窗口中间
设置边框的样式宽度颜色
使用CSS3 rgba()设置边框颜色（不支持旧浏览器）
设置padding为20px
设置背景色为亮灰色
使用CSS3 rgba()设置背景色（不支持旧浏览器）
把文本排列到中间
调用.box-shadow() LESS函数
}

```

小知识 什么是 rgb, 什么是 rgba

在 CSS2.1 之前设置颜色有 3 种方式，分别为 `rgb()`、十六进制代码（`#rrggbb`）和颜色关键字（Color Keyword）。使用这 3 种方式，我们可以分别为元素的字体、背景、边框和轮廓等设置不同的颜色。其中，最常用的是 `rgb()`、十六进制代码（`#rrggbb`）两种方法。

下面让我们了解一下 `rgb()` 与十六进制颜色码之间的对应关系。比如有一种颜色，其十六进制颜色码为 `#97c2fe`，在 `#` 后共有 6 个数字与字母的组合，其中前两位代表的是 Red，中间两位代表的是 Green，最后两位代表的是 Blue。在十六进制颜色码中，每位可以取 0~9 或 a~f 中的值。下面我们使用一个表格来说明 `rgb()` 与十六进制颜色码之间的对应关系。在表格中，水平与纵向的交叉点即是要添入 `rgb()` 中的数值，即将 `#97c2fe` 转换为 `rgb()` 之后为 `rgb(121, 196, 239)`。

Hex-code																
Red: 97				Green: c2				Blue: fe								
121																
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
10	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
11	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
12	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
13	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
14	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
15	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

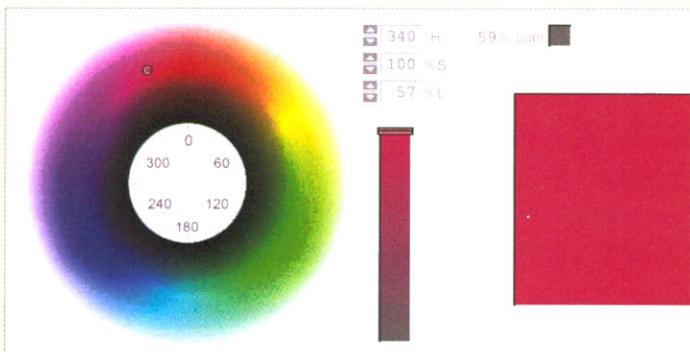
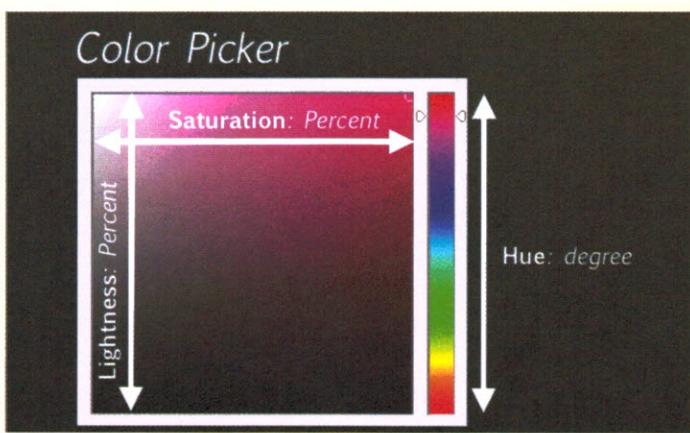
CSS3 支持 `rgba` 属性，使用它不仅能够让我们像通常一样设置 RGB 颜色值，而且还可以设置其透明度。RGBA 在 RGB 的基础上添加了控制 Alpha 透明度的参数，允许我们设置元素的透明度，它与不透明度（`Opacity`）一样，取值范围为 0~1。当取值为 0 时，表示该元素完全透明，处于不可见状态；当其取值为 1 时，完全不透明，元素处于完全可见状态；当取值为 0.5 时，元素处于半透明状态。

除了`rgba()`外，CSS3还增加了`hsl()`与`hsla()`属性。RGB是通过混合光的三原色而形成颜色的，而HSL则是通过对色调(Hue)、饱和度(Saturation)和亮度(Lightness)3个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的。HSL比RGB更直观、更明了，理解起来也更简单。

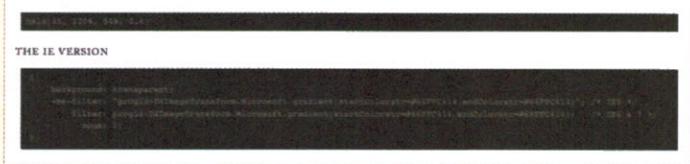
与`rgba()`一样，`hsla()`中的`a`也是指Alpha通道值，用于设置元素的不透明度，它接受4个参数，分别为色调、饱和度、亮度和Alpha透明度，前3个参数的单位分别是度数、百分比、百分比，Alpha透明度取值在0~1之间。更详细的内容，请访问<http://2.gp/p3hx>页面进行学习。

令人遗憾的是IE8之前的版本不支持CSS3的颜色属性，仅支持MS专用的滤镜(Filter)。所以在制作网页时，通常都需要针对旧版本的浏览器添加额外的代码。由于各种

滤镜不属于已有的标准，因此没必要熟记。当然使用一些工具，帮助我们快速生成滤镜代码是非常方便的，更多详细内容，请访问<http://2.gp/p3hz>页面，进行了解学习。



color: hsla(45, 100%, 54%, 0.8);



接下来，添加LESS变量(`@wrap_width`)与LESS函数(`.box-shadow()`)，在()内设置形式参数，提供默认值。在函数内部，使用`@arguments`，应用所有的形式参数。

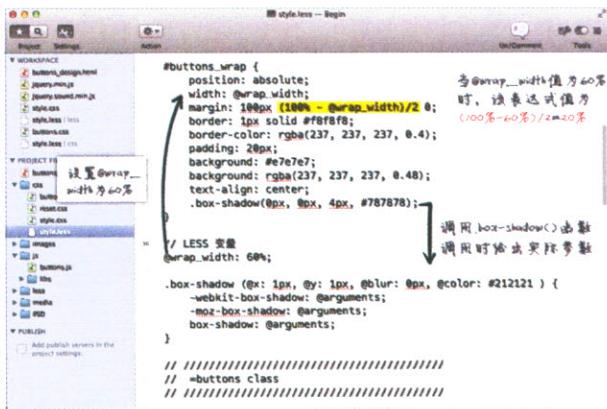
```

@wrap_width: 60%;

(从略……)

.box-shadow (@x: 1px, @y: 1px, @blur: 0px, @color: #212121 ) {
  -webkit-box-shadow: @arguments;
  -moz-box-shadow: @arguments;
  box-shadow: @arguments;
}

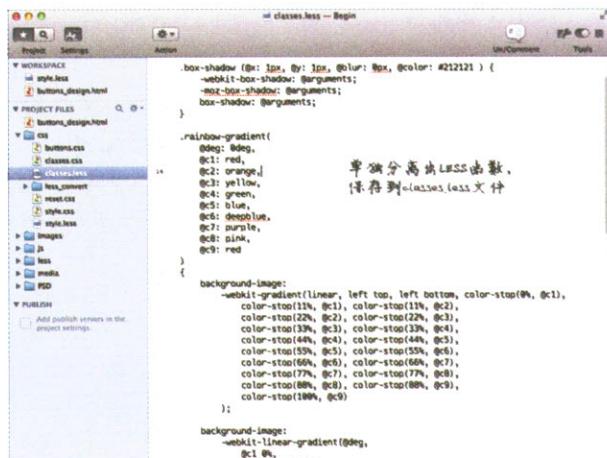
```



保存 style.less 文件，在浏览器中查看页面，可以看到 LESS 变量与 LESS 函数都工作正常。到这儿，网页的背景设计与按钮包装设计完成。



下面我们开始编写按钮与扩展类代码。首先，从 style.less 文件中分离出 LESS 函数。在创建 classes.less 文件后，再分离出 LESS 函数。



再次返回到 style.less 文件中，添加 @import 语句，调用分离出的 classes.less 文件。注意调用时若被调用的文件前含有“css”，则不能正常完成调用。CSS 文件可以被动态生成后，添加到 buttons_design.html 文件中，但 LESS 文件不能被动态添加，必须先进行调用，文件的位置一定要写正确。

```
@charset "utf-8";
@import "css/reset.css";
@import "classes.less";
```

使用@import语句调用classes.less文件

接下来，向 style.less 文件添加按钮类。首先添加 .button 选择器，而后再在其中编写控制样式。<a> 元素并未使用网页浏览器的初始设置，当按钮类被添加到它上面时，要去除链接文本的下划线。当按钮类被添加到非 <a> 元素上时，并且鼠标指针移动到该元素上时，光标要变成手形形状。

```
.button {
    text-decoration: none !important;
    cursor: pointer;
    display: inline-block;
    border: 1px solid #ccc;
    padding: 1em 1.5em;
    background-color: #e0e0e0;
    color: #232323;
    font: 14px/1 Verdana, Arial, Sans-Serif;
}
```

声明.button类

删除文本修饰

设置鼠标指针形状为手形

设置为行内块元素

设置边框粗细、样式、颜色

设置上下内边距为 1em，左右内边距为 1.5em

设置背景色为亮灰色

设置字体颜色为暗灰色

设置字号、行间、字体

小知识 ↗ important 是什么？

CSS，全称为 Cascading Style Sheet，译为“级联样式表”或“层叠样式表”。这里的 Cascade 意义为从属规则，是为实现样式应用的优先规则。在网页中使用样式表时，大致分为两种方法，一种方法是“内嵌样式”（in-line style），另一种是外部样式。使用“内嵌样式”时，直接在 HTML 元素中使用 style 属性，控制该元素的样式；而在外部样式中，样式控制代码在单独的外部样式表中，通过使用 HTML 选择器来实现对相应元素的样式控制，应用时要将网页链接到外部样式表。

例) 内嵌样式：<p style="color: #3abcde; font-size: 14px;">…</p>

外部样式：p { color: #3abcde; font-size: 14px; }

其实，两种方法在功能上是相同的，但是它们的优先级是不同的。一般而言，内嵌样式比外部样式有更高的优先级，即若在一个网页中同时应用了内嵌样式与外部样式，则网页将优先应用内嵌样式，外部样式将会忽略。

内嵌样式 > 外部样式

但是有时候我们可能需要优先使用外部样式，而不是内嵌的样式，这个时候就需要使用“!important”关键字，该关键字能够提高外部样式的优先级，使外部样式优先于内嵌样式。

内嵌样式 < 外部样式

由于“!important”关键字能够使内嵌样式失效，因此使用时应当要谨慎，不到万不得已，尽量不要使用这个关键字。在实例中，我们使用该关键字去除应用了 button 类的所有元素的下划线。

保存代码，在浏览器中查看网页，可以发现带有button类的所有元素的字体、补丁(Padding)、边框、背景色和光标形状等都设置成了统一样式。



然后，我们开始向 classes.less 中添加 .insert-box-shadow 与 .button-gradient 两个 LESS 函数，这两个函数分别用于向按钮添加内侧阴影与渐变效果。

.insert-box-shadow() 这个 LESS 函数与 .box-shadow() 类似，但它产生的阴影不是在外部(outset)，而是在内部 (inset)。.insert-box-shadow() 与 .box-shadow() 具有相同的参数，但在函数内部使用的是 inset 关键字。

.box-shadow() 函数是用来制作按钮类的函数，它使背景图片产生移动。它有 4 个参数，第 1 个参数 @bgc 用来为不支持 CSS3 Multi-Backgrounds 的旧浏览器设置背景颜色；第 2 个参数 @img_url 用来设置水滴背景图片的地址；第 3 个与第 4 个参数 (@bs、@be) 分别用来输入渐变的起始颜色与终止颜色。

```
.inset-box-shadow ( @x: 1px, @y: 1px, @blur: 0px, @color: #212121 ) {
    -webkit-box-shadow: inset @x @y @blur @color;
    -moz-box-shadow: inset @x @y @blur @color;
    box-shadow: inset @x @y @blur @color;
}

.button-gradient ( @bgc: #e5e5e5, @img_url: none, @bs: #fff, @be: #cecece ) {
    background-color: @bgc;
    background-image: @img_url, @img_url, -webkit-gradient(linear, left top, left bottom, from(@bs),
        to(@be));
    background-image: @img_url, @img_url, -moz-linear-gradient(@bs, @be);
    background-image: @img_url, @img_url, -webkit-linear-gradient(@bs, @be);
    background-image: @img_url, @img_url, -o-linear-gradient(@bs, @be);
    background-image: @img_url, @img_url, -ms-linear-gradient(@bs, @be);
    background-image: @img_url, @img_url, linear-gradient(@bs, @be);
}
```

LESS 函数编写完成后保存，而后向 style.less 文件的 .button 类添加如下代码，为按钮文本添加阴影效果，并调用上面的两个 LESS 函数。在调用 .inset-box-shadow() 函数时，分别指定 X 轴偏移、Y 轴偏移、blur、阴影颜色；调用 .button-gradient 函数时，不用设置参数，使用默认的参数值即可，当不设置参数时，函数名称后面的小括号可以省略掉。

```
.button {  
    text-shadow: 1px 1px #f2f2f2;  
    .inset-box-shadow(0px, 0px, 4px, #787878);  
    .button-gradient;  
}
```

程序运行时，less.js 会调用 style.less 文件中的函数动态生成 CSS 代码，并插入到 HTML 文档中。在浏览器中，运行 Firebug 插件，在 HTML 窗口中，可以看到生成的 CSS 代码。



当鼠标指针移动到按钮上，或者悬停在按钮上时，按钮的渐变要发生改变才行。当按钮获得焦点时，边框隐藏起来，outline 属性值为 0，在 .button 类添加如下代码。保存代码，在浏览器中查看页面，可以看到当鼠标指针移动到按钮上，或按 Tab 键使按钮获得焦点时，渐变发生了变化。



The screenshot shows a LESS editor interface with the following details:

- Project:** WORKSPACE contains files like buttons_design.html, jquery.min.js, jquery.sound.min.js, style.css, style.less, buttons.css, style.less | Trash, classes.less, and style.less | css.
- Project Files:** PROJECT FILES contains buttons_design.html, css (buttons.css, classes.less, reset.css, style.css), js (buttons.js), libs (jquery.min.js, jquery.sound.min.js, less-1.3.3.min.js, less.min.js, modernizr.min.js, PIE.min.js), and less.
- Code View:**

```

@c7 77%, @c8 77%,
@c8 88%, @c9 88%,
@c1 100%
};

// 按钮核心类
.button {
    text-decoration: none !important;
    cursor: pointer;
    display: inline-block;
    border: 1px solid #ccc;
    padding: 1em 1.5em;
    background-color: #e0e0e0;
    color: #232323;
    font: 14px/1 Verdana, Arial, Sans-Serif;
    text-shadow: 1px 1px #f2f2f2;
    .inset-box-shadow(0px, 0px, 4px, #787878);
    .button-gradient;
}

&:hover, &:focus {
    outline: 0;
    .button-gradient(#eee, none, #fff, #c0c0c0);
}

```
- Annotations:** A callout points to the LESS pseudo-class `&:hover` with the text "在 LESS 中的 :hover、:focus 前添加 & 支持嵌套写法".

到这里，按钮类就基本完成了。下面我们在 Firebug 中查看按钮类的代码，可以看到带有不同浏览器前缀的代码是自动生成的。灵活使用 LESS，能够帮助我们快速生成 CSS 代码，节省了时间，提升了编码效率。

The screenshot shows the Firebug source editor with the following details:

- Source Edit:** filter.css
- Code:**

```

/*
 * buttons class
 */
.button {
    text-decoration: none !important;
    cursor: pointer;
    display: inline-block;
    border: 1px solid #ccc;
    padding: 1em 1.5em;
    background-color: #e0e0e0;
    color: #232323;
    font: 14px/1 Verdana, Arial, Sans-Serif;
    text-shadow: 1px 1px #f2f2f2;
    -webkit-box-shadow: inset 0px 0px 3px #ffffff;
    -moz-box-shadow: inset 0px 0px 3px #ffffff;
    box-shadow: inset 0px 0px 3px #ffffff;
    background-color: #e5e5e5;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#ffffff), to(#cecece));
    background-image: none, none, -moz-linear-gradient(#ffff, #cecece);
    background-image: none, none, -webkit-linear-gradient(#ffff, #cecece);
    background-image: none, none, -o-linear-gradient(#ffff, #cecece);
    background-image: none, none, -ms-linear-gradient(#ffff, #cecece);
    background-image: none, none, linear-gradient(#ffff, #cecece);
}

.button:hover, .button:focus {
    outline: 0;
    background-color: #eeeeee;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#ffff), to(#c0c0c0));
    background-image: none, none, -moz-linear-gradient(#ffff, #c0c0c0);
    background-image: none, none, -webkit-linear-gradient(#ffff, #c0c0c0);
    background-image: none, none, -o-linear-gradient(#ffff, #c0c0c0);
    background-image: none, none, -ms-linear-gradient(#ffff, #c0c0c0);
    background-image: none, none, linear-gradient(#ffff, #c0c0c0);
}

```
- Annotations:** A callout points to the generated CSS code with the text "由 less.js 动态转换的 CSS 代码".

按钮类编写完成后，接着，我们开始编写按钮的扩展类。首先添加一个 .ko 类，为按钮文本指定使用的字体，而后再添加控制按钮尺寸的类，名称的基本形式为 small、x-big 和 xx-big。

```

.button {
  &:hover, &:focus { ... }
  &ko { font-family: 'Nanum Gothic', 'Dotum', Sans-Serif; }

  &.xx-small { font-size: 10px; } 处理按钮的: hover, : focus 状态
  &.x-small { font-size: 11px; }
  &.small { font-size: 12px; }
  &.big { font-size: 18px; }
  &.x-big { font-size: 20px; }
  &.xx-big { font-size: 22px; }

}

按钮.ko类
按钮大小.xx-small~xx-big类

```



接下来，为按钮添加圆角边框。在设置圆角边框时，我们使用 `.border-radius` 这个 LESS 函数，将其添加到 `classes.less` 文件中。

```

.border-radius ( @radius: 8px ) {
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
  -khtml-border-radius: @radius;
  border-radius: @radius;
}

```

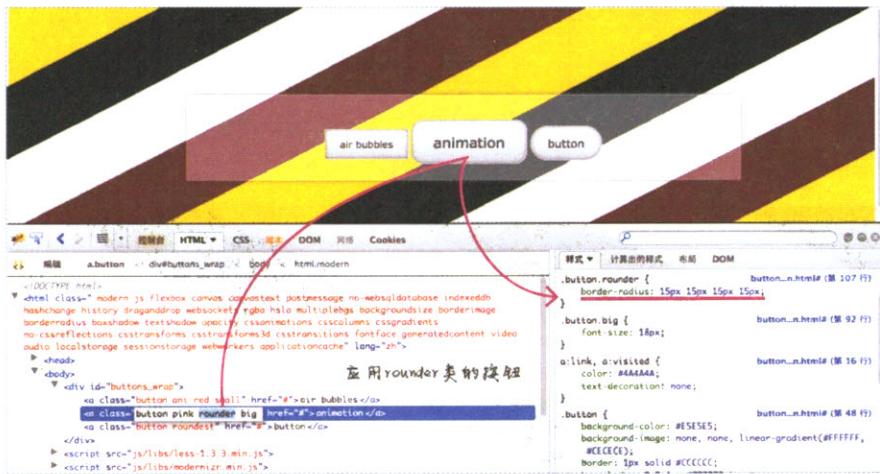
向 `style.less` 中添加圆角边框扩展类，类的名称分别为 `.round`、`.rounder` 和 `.roughest`，调用 `.border-radius` 函数时，分别传入 `5px`、`15px` 和 `30px` 3 个参数，代码如图：

```

// 大小扩展类
&.xx-small { font-size: 10px; }
&.x-small { font-size: 11px; }
&.small { font-size: 12px; }
&.big { font-size: 18px; }
&.x-big { font-size: 20px; }
&.xx-big { font-size: 22px; }

// 圆角框扩展类
&.round { .border-radius(5px); }
&.rounder { .border-radius(15px); }
&.roughest { .border-radius(30px); }

```



接下来，要添加的扩展类是动画类，该类是形成动态气泡按钮效果的核心。动画类的名称为`.ani`，在`.button`类的内部进行添加。

```

@img_url: url(..../images/button_bg.png);

.button {
  &.ani {
    background-image: @img_url;
    background-repeat: no-repeat;
    background-position: left bottom, right top, 0 0;

    .button-gradient(#eee, @img_url, #fff, #c0c0c0);

    .transition;
    &:hover, &:focus {
      background-position: left bottom, right top, 0 0;

      .button-gradient(#eee, @img_url, #fff, #b0b0b0);
    }
  }
}

```

为`@img_url`变量设置值

声明`.ani`扩展类

设置背景图片为`@img_url`

设置背景图片为不重复

设置3个背景位置值（向渐变添加多张图片）

调用`.button-gradient`函数

调用`.transition`函数

声明`.ani:hover`、`.ani:focus`状态

变更3个背景位置值

调用`.button-gradient`函数

在 .ani 类中用到了名为 .transition 的 LESS 函数，我们要把该函数添加到 classes.less 文件中，如下所示。

```
.transition (@prop: all, @duration: 0.4s, @easing: ease, @delay: 0s) {  
  -webkit-transition: @prop @duration @easing @delay;  
  -moz-transition: @prop @duration @easing @delay;  
  -o-transition: @prop @duration @easing @delay;  
  -ms-transition: @prop @duration @easing @delay;  
  transition: @prop @duration @easing @delay;  
}  
}
```

最后，添加颜色类。在这里，我们以添加 .red 颜色类为例向大家讲解如何创建颜色类。在创建的过程中，设置背景图片时，我们使用 @img_url 这个变量指定图片路径。



```
@red_bc: #e80408;  
@red_ec: #6d0019;  
(…从略…)  
.button {  
  &.red {  
    border: 1px solid #8c484f;  
    color: #121212 !important;  
    .button-gradient(@red_bc, none, @red_bc, @red_ec);  
    &.ani {  
      .button-gradient(@red_bc, @img_url, @red_bc, @red_ec);  
    }  
  }  
}
```

将渐变起始颜色保存至@red_bc变量中
将渐变结束颜色保存至@red_ec变量中

声明.red扩展类
设置边框为1px，实线，深红色
设置按钮文本颜色为深灰色
调用.button-gradient函数
声明.ani类
调用.button-gradient函数



使用与上面相似的代码，分别添加 orange、yellow、green、blue、navy、purple 和 pink 类，添加时只要复制 red 类的代码，更改类名称，和相应的颜色就可以了。为了便于后期修改维护，我们使用 LESS 变量来取代具体的颜色名称。

The screenshot shows a LESS file with color variable declarations and their usage in button classes. Red arrows highlight the LESS variable declarations and their corresponding usage in the generated CSS and LESS code.

```

.button.red {
    border: 1px solid #8C484F;
    color: #121212 !important;
    .button-gradient(@red_bc, none, @red_bc, @red_ec);
}

&.ani {
    .button-gradient(@red_bc, @img_url, @red_bc, @red_ec); }

.button.orange {
    border: 1px solid #89614A;
    color: #121212 !important;
    .button-gradient(@orange_bc, none, @orange_bc, @orange_ec); }

&.ani {
    .button-gradient(@orange_bc, @img_url, @orange_bc, @orange_ec); }

.button.yellow {
    border: 1px solid #d1c78c;
    color: #121212 !important;
    .button-gradient(@yellow_bc, none, @yellow_bc, @yellow_ec); }

&.ani {
    .button-gradient(@yellow_bc, @img_url, @yellow_bc, @yellow_ec); }

.button.green {
    border: 1px solid #6c8e3b;
    color: #121212 !important;
    .button-gradient(@green_bc, none, @green_bc, @green_ec); }

&.ani {
    .button-gradient(@green_bc, @img_url, @green_bc, @green_ec); }

.button.blue {
    border: 1px solid #3665ac;
    color: #121212 !important;
    .button-gradient(@blue_bc, none, @blue_bc, @blue_ec); }

&.ani {
    .button-gradient(@blue_bc, @img_url, @blue_bc, @blue_ec); }

.button.navy {
    border: 1px solid #281d60;
    color: #121212 !important;
    .button-gradient(@navy_bc, none, @navy_bc, @navy_ec); }

&.ani {
    .button-gradient(@navy_bc, @img_url, @navy_bc, @navy_ec); }

```

```

//声明的LESS变量
@img_url: url(..../images/;

// =buttons class
// =LESS Functions

@red_bc: #E80408; // 红色;
@red_ec: #E60019; // 红色;

@orange_bc: #ffa13d;
@orange_ec: #a04814;

@yellow_bc: #ffff496;
@yellow_ec: #e5c600;

@green_bc: #aeef6f;
@green_ec: #4c6e1b;

@blue_bc: #4FA7FF;
@blue_ec: #0549A8;

@navy_bc: #5064D3;
@navy_ec: #281D60;

@purple_bc: #CC78ED;
@purple_ec: #4B1166;

@pink_bc: #FC679B;
@pink_ec: #AA054A;

```

保存 style.less 文件后，打开 Firefox 浏览器，运行 Firebug 后，可以看到由 LESS 生成的 CSS 代码。基于对各种浏览器兼容性的考虑，CSS 中出现了大量的重复代码，如果我们使用纯 CSS 编写这些代码，将会耗费大量的时间与精力。但使用了 LESS 技术后，我们轻松地解决了这个问题，大大提高了编码的效率。

Source Edit | filter.css buttons_design.html

```
.button.red {
    border: 1px solid #8C484F;
    color: #121212 !important;
    background-color: #e80408;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#e80408), to(#d00019));
    background-image: none, none, -moz-linear-gradient(#e80408, #d00019);
    background-image: none, none, -webkit-linear-gradient(#e80408, #d00019);
    background-image: none, none, -o-linear-gradient(#e80408, #d00019);
    background-image: none, none, -ms-linear-gradient(#e80408, #d00019);
    background-image: none, none, linear-gradient(#e80408, #d00019);
}

.button.red:hover, .button.red:focus {
    background-color: #e80408;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#fb2428), to(#a00025));
    background-image: none, none, -moz-linear-gradient(#fb2428, #a00025);
    background-image: none, none, -webkit-linear-gradient(#fb2428, #a00025);
    background-image: none, none, -o-linear-gradient(#fb2428, #a00025);
    background-image: none, none, -ms-linear-gradient(#fb2428, #a00025);
    background-image: none, none, linear-gradient(#fb2428, #a00025);
}

.button.red.ani {
    background-color: #e80408;
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -webkit-gradient(linear, left top, left bottom, from(#e80408), to(#d00019));
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -moz-linear-gradient(#e80408, #d00019);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -webkit-linear-gradient(#e80408, #d00019);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -o-linear-gradient(#e80408, #d00019);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -ms-linear-gradient(#e80408, #d00019);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), linear-gradient(#e80408, #d00019);
}

.button.red.ani:hover, .button.red.ani:focus {
    background-color: #e80408;
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -webkit-gradient(linear, left top, left bottom, from(#fb2428), to(#a00025));
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -moz-linear-gradient(#fb2428, #a00025);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -webkit-linear-gradient(#fb2428, #a00025);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -o-linear-gradient(#fb2428, #a00025);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), -ms-linear-gradient(#fb2428, #a00025);
    background-image: url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), url(http://ex.yamoo9.com/ex_06/Begin/css/..images/button_bg.png), linear-gradient(#fb2428, #a00025);
}
```

由LESS代码转换而成的CSS代码，若采用手工编写，那将会是耗费极大的工作量。

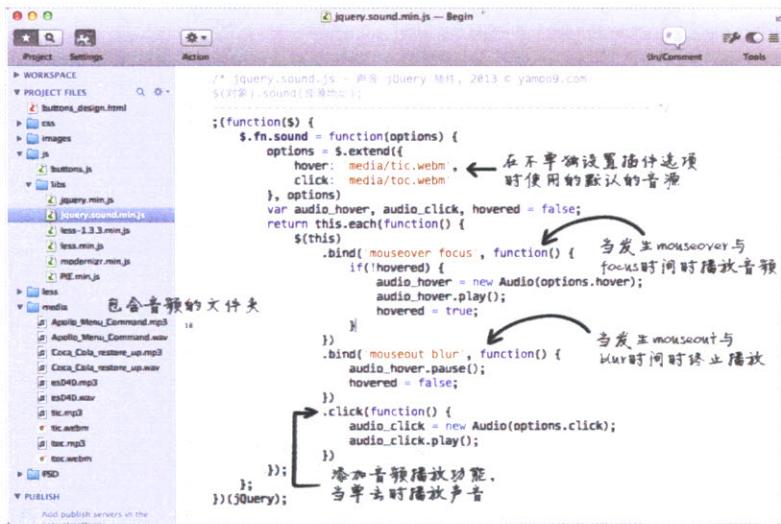
在前面我们已经编写好了按钮及扩展类，接下来我们开始灵活地使用它们。首先打开 buttons_design.html 文件，添加按钮元素后，设置按钮类。添加扩展类，能添加更多功能，如果不再需要那些功能，只要删除相应的类即可，非常有利于后期修改、维护，进一步提高了编码效率。



在本章中，我们学习了几个重要的概念。下面让我们简单地回顾一下：首先通过按钮及扩展类的设计模式，编写重用代码，而后基于各浏览器兼容性的考虑，使用了 LESS 框架技术轻松实现了各种复杂代码。在后续章节中，我们将继续使用 LESS 框架技术编写相关代码，使大家尽快地熟悉它们，以便将这种新的技术早日应用到网页开发中，进一步提高网页开发效率。

向jQuery按钮添加声音插件

在第3章中，我们已经创建了一个jQuery声音插件。在本部分，我们将继续扩展该插件，并将其应用到按钮上。请参考图，创建好jquery.sound.js文件后，编写扩展代码。



在buttons_design.html调用jquery.sound.js文件。打开js文件夹中的buttons.js文件后，将.button包装成jQuery对象，再连接上sound()插件。在向按钮应用了声音插件后，当指定的事件发生时，相关的声音就会播放。

```
(function($) {
    $('button').sound();
})(jQuery);
```

将.button包装成jQuery对象后调用sound()

如果不使用默认音源，改用其他音源，只需要更改插件选项设置即可。

```
(function($) {
    $('button').sound({
        hover: 'media/要更改的音源',
        click: 'media/要更改的音源'
    });
})(jQuery);
```

使用if改变选项设置

设置播放音源地址

设置单击时要播放的音源地址

关闭设置选项的if

当鼠标指针移动到按钮上时，指定的声音就会开始播放。当然在旧版本的浏览器或不支持 HTML5 Audio 的网页浏览器中，是无法播放声音的。

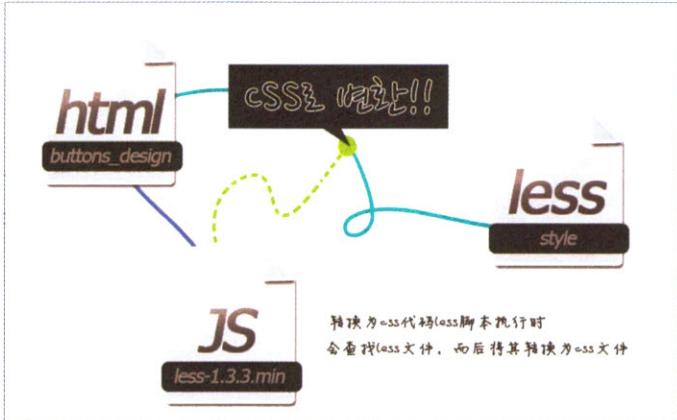
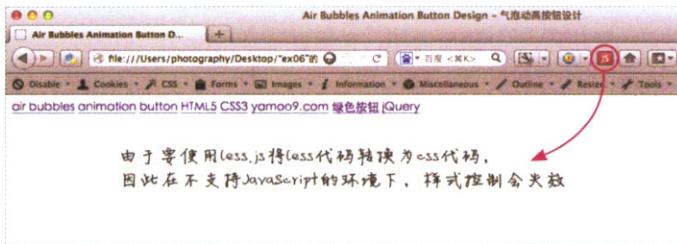


到这里，我们已经学习了向按钮附加按钮类与扩展类的方法，以及动态连接声音的方法。到目前为止，好像一切正常，没有出现任何问题，但是其实已经出现了几个问题，当代码运行在不支持脚本的环境中时，这些问题就会显现。在浏览器中关闭 QuickJava 插件的 JS 选项后，刷新页面，会发现所有的样式全部失效了。

在不支持脚本的环境下，还能使用 LESS 吗？经过前面的学习，我们已经知道 LESS 是一种非常有用的工具，它能帮助我们更高效地使用 CSS3。但是在不支持脚本的环境下，它无法正常运作。那么我们该如何解决这一问题呢？

首先让我们一起回顾一下，LESS 代码是如何转换为 CSS 代码的。在支持脚本的浏览器中，打开 HTML 文档时，它会调用 LESS

脚本文件，查找 LESS 文件，而后将其转换为 CSS 代码。当使用的浏览器不支持脚本时，LESS 代码就无法转换为 CSS 代码，所以我们先需要使用一个工具将 LESS 代码转换成 CSS 代码。这样就不再需要使用浏览器调用脚本将 LESS 代码转换成 CSS 代码了，也就是说 HTML 文档直接调用转换后的 CSS 代码，就不需要依赖脚本转换了。那么什么工具能够帮助我们将 LESS 代码转换成 CSS 代码呢？Less.app 是 Mac OS 下的一个工具，用来将 LESS 代码转换成 CSS 代码。进入 incident57.com 网站 (<http://incident57.com/less>) 后，单击 Download 按钮，下载它。请注意 Less.app 是苹果机专用的，它不能安装在 Windows 环境下。如果你是 Windows 用户，请跳过本部分内容，直接阅读第 181 页中的内容。



The only thing easier is making fun of Internet Explorer.

Download

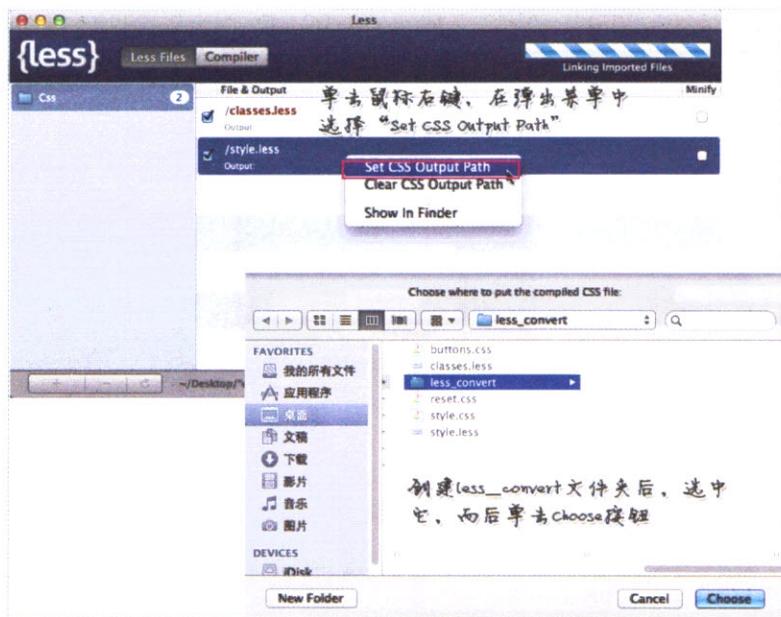
Less.app的使用方法非常简单，首先将下载的Less.app移动到应用程序文件夹Application中。



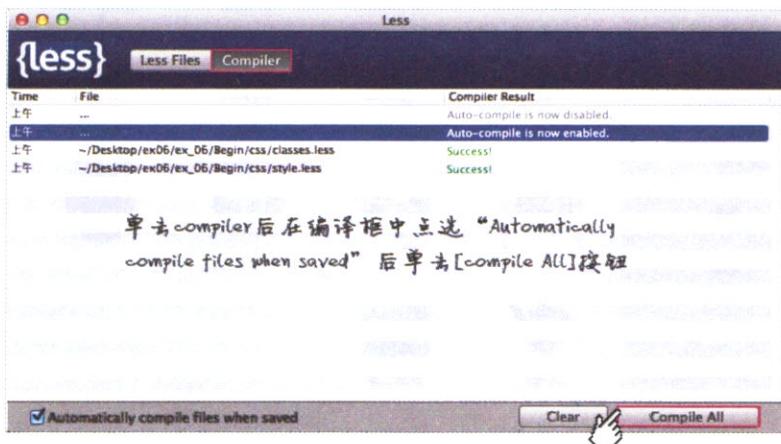
然后运行 Less.app 工具，单击左下的“+”按钮，选择编写好的 LESS 文件，单击“Add”按钮，进行添加。



在列表中可以看到前面创建的两个 LESS 文件 (style.less、classes.less) 已经被添加上了。右键单击 style.less 文件，在弹出的快捷菜单中，选择 Set CSS Output Path 菜单，设置转换成 CSS 代码后文件的保存位置。创建好要保存的文件夹 less_convert，然后将其指定为 CSS 文件的保存位置即可。



在 Less.app 窗口中，单击 Compiler 按钮，界面切换至编译窗口中。然后单击右下角的 Compile All 按钮，进行编译，在编译结果中出现 “Success!” 信息，表示编译成功，LESS 文件被顺利地转换成了 CSS 文件。



进入到保存 CSS 的文件夹 less_convert 中，可以看到已经编译好的 style.css 文件。打开该文件，可以查看到编译后的纯 CSS 代码。

```

.button {
    position: relative;
    text-decoration: none !important;
    cursor: pointer;
    display: inline-block;
    border: 1px solid #ccc;
    padding: 1em 1.5em;
    background-color: #e0e0e0;
    color: #222222;
    font: 14px/1 Verdana, Arial, Sans-Serif;
    text-shadow: 1px 1px #f2f2f2;
    -webkit-box-shadow: inset 0px 0px 4px #787878;
    -moz-box-shadow: inset 0px 0px 4px #787878;
    box-shadow: inset 0px 0px 4px #787878;
    background-color: #e0e0e0;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#ffffff),
        to(#e0e0e0));
    background-image: none, none, -moz-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, -webkit-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, -o-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, linear-gradient(#ffffff, #e0e0e0);
}

.button:hover,
.button:active {
    outline: 0;
    background-color: #eeeeee;
    background-image: none, none, -webkit-gradient(linear, left top, left bottom, from(#ffffff),
        to(#e0e0e0));
    background-image: none, none, -moz-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, -webkit-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, -o-linear-gradient(#ffffff, #e0e0e0);
    background-image: none, none, linear-gradient(#ffffff, #e0e0e0);
}

```

接下来，将变动的 style.css 文件保存到 CSS 文件夹中，buttons_design.html 文档会直接调用 style.css 文件，而不是 style.less 文件。当然，我们要删除 less.js 文件，因为我们不会用 less.js 脚本将 LESS 文件转换为 CSS 文件了。

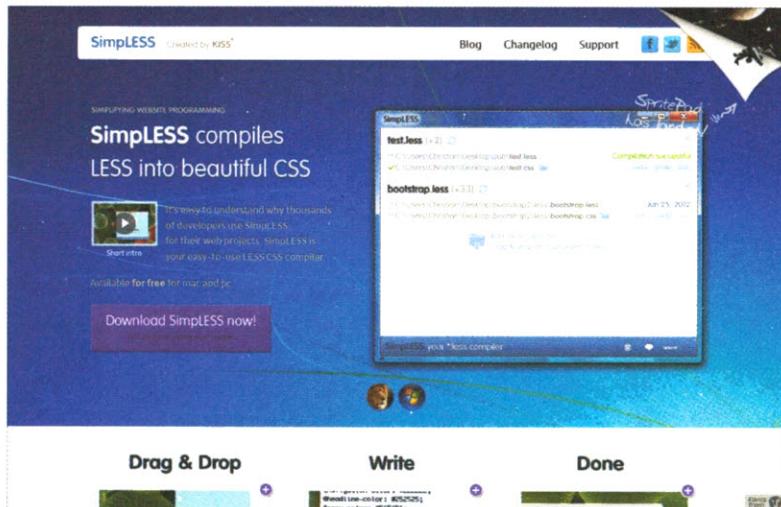
```
<!DOCTYPE html>
<html lang="zh" class=" no-js old ie6"></html>
<html lang="zh" class=" no-js old ie7"></html>
<html lang="zh" class=" no-js old ie8"></html>
<html lang="zh" class=" no-js ie9"></html>
<html lang="zh" class=" no-js modernr"></html>
<head>
    <meta charset="utf-8" />
    <title>Air Bubbles Animation Button Design - 气泡动画按钮设计</title>
    <link rel="stylesheet" href="css/style.css" />
    <link href="style.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <div id="buttons_wrap">
        <a href="#" class="button ani round red xx-big >air bubbles</a>
        <a href="#" class="button pink rounnder xx-big >animation</a>
        <a href="#" class="button x-small >button</a>
        <a href="#" class="button ani rounded blue small >HTML5</a>
        <a href="#" class="button yellow ani big >CSS3</a>
        <a href="#" class="button orange medium ani rounder >yamoo9.com</a>
        <a href="#" class="button green round ani x-big zh >绿色按钮</a>
        <a href="#" class="button purple small >jQuery</a>
    </div>
</body>
<!-- 基本调用 -->
<script src="js/libs/jquery-1.3.3.min.js"></script>
<script src="js/libs/modernizr.min.js"></script>
<script src="js/libs/jquery.min.js"></script>
<script src="js/libs/jquery.sound.min.js"></script>
<script src="js/buttons.js"></script>
</body>
```

如果你使用的不是苹果系统，而是 Windows 系统，那么我们该如何将 LESS 文件转换为 CSS 文件呢？我们要使用的工具是 SimpLESS，它是一个支持跨平台的 LESS CSS 编译器，能够在 Mac OS、Windows 和 Linux 环境下正常工作。

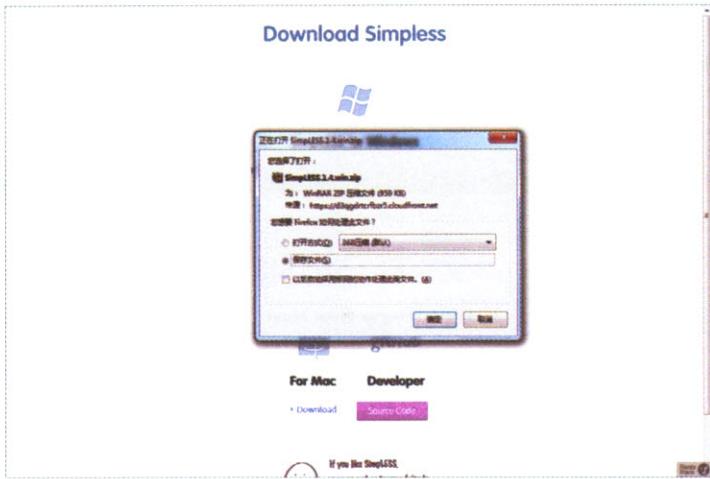
→ 小知识 ←

笔者在写作本书时，应用 SimpLESS 比 LESS.app 要少一些。所以如果你是苹果系统用户，推荐你使用 LESS.app，而不是 SimpLESS。

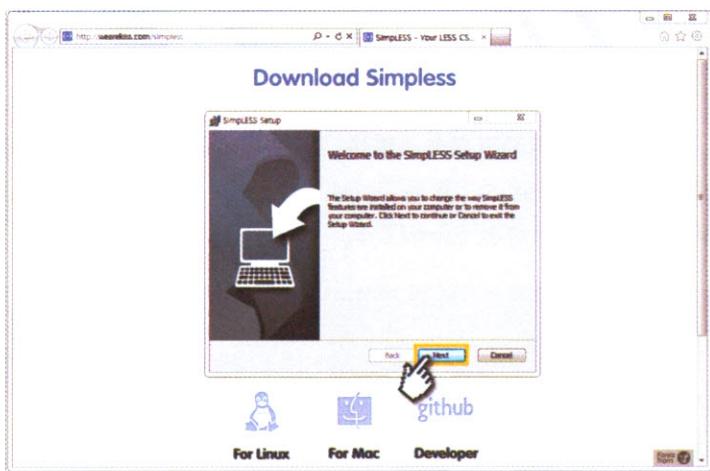
01 进入 SimpLESS 网站 (<http://wearekiss.com/simpless>)，单击“Download SimpLess”按钮，跳转到下载页面。



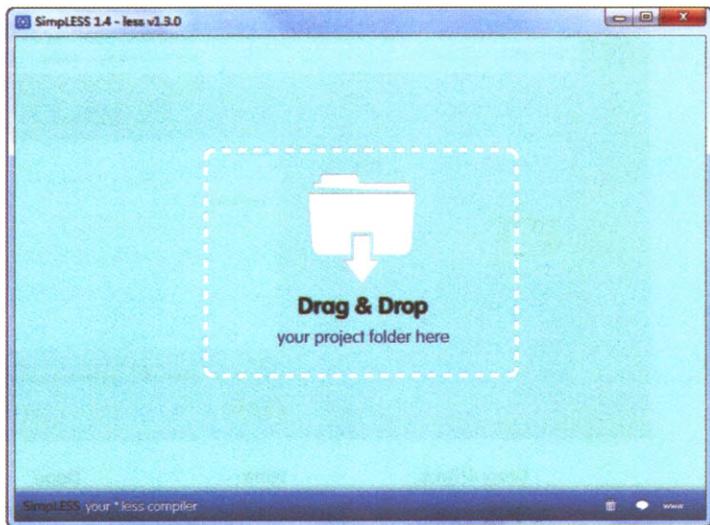
02 在下载页面中，单击“确定”按钮，将 SimpLESS 下载到本地电脑。



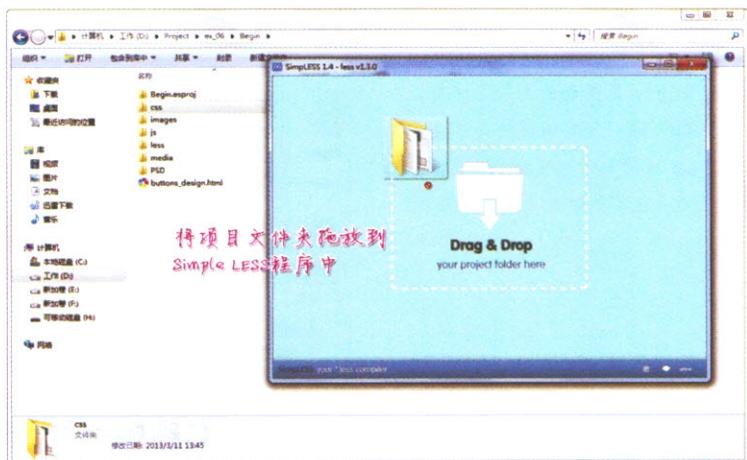
03 解压缩后，进入 SimpLESS 文件夹，双击 SimpLESS.exe，稍等一会儿后，即可完成安装。



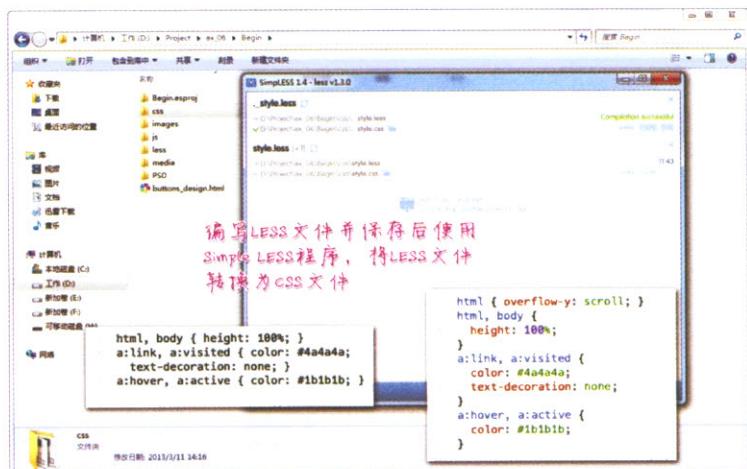
04 安装完成后，进入 SimpLESS 文件夹，双击 SimpLESS.exe，运行它。



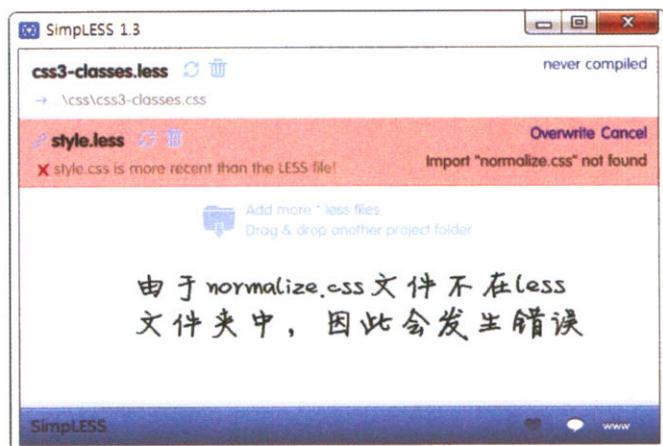
05 使用方法非常简单，只要将项目文件夹拖放到 SimpLESS 界面中即可，如图所示。



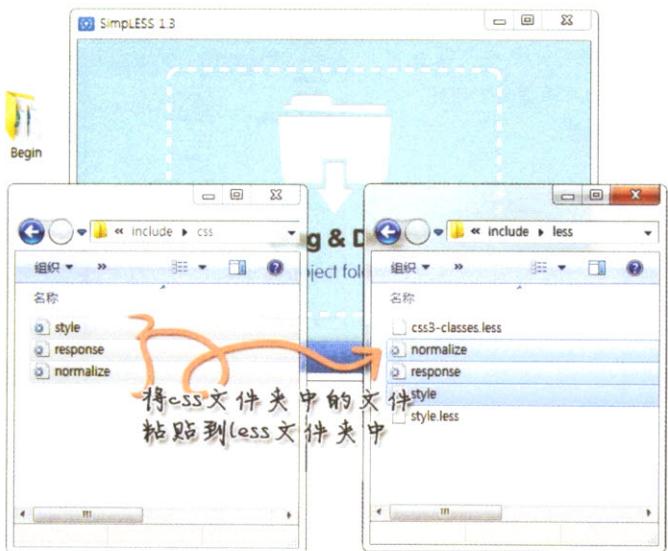
06 如果项目文件夹中包含 LESS 文件，SimpLESS 应用程序会自动将其转换为 CSS 文件。在转换为 CSS 文件后，网页文档只需要直接调用 style.css 就可以了，不需要再使用脚本进行转换了。



07 在使用 SimpLESS 移动项目文件夹时，可能会发生错误。发生错误的原因是 CSS 文件必须同时存在于相关的 less 文件夹中，这点与 Less.app 有所不同。



08 若出现错误，如“import normalize.css not found”，就表示无法查找到 CSS 文件。解决这一问题的方法是：复制 css 文件夹中相应的 CSS 文件，而后将其粘贴到 less 文件夹中。



09 然后单击右侧的“Overwrite”按钮，进行重写覆盖，错误信息即可消失。



在本章中，我们一起学习了更多关于 HTML、CSS、JavaScript 的知识，设计网页时，灵活使用这些新技术，能够制作出更精美、更健壮的页面。但我們也不是一味地追求新技术，而是综合考虑各种情况，灵活运用这些新技术，一味地过度偏用这些新技术也是不可取的。请各位牢记这一点，这样设计的网页才更整洁、更干净，也更漂亮。

7 章

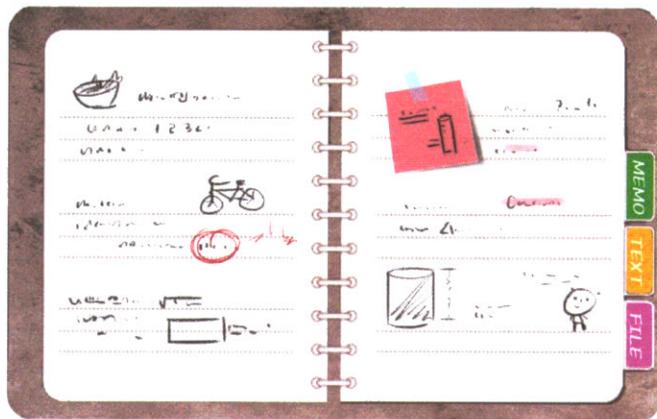
设计带动画的选项卡菜单

选项卡菜单（TabMenu）是网页设计中常用的设计元素之一，特别是在设计空间有限的页面时常使用它。使用这种设计元素，能够帮助我们更有效地利用有限的网页空间。在本章中，我们将一起学习制作选项卡菜单的方法，同时向选项卡菜单添加动画效果，进一步增强选项卡菜单的趣味性。

设计效果预览

在日记本或文件夹中，我们会经常看到一些选项卡的结构。在本章中，我们将尝试使用这种选项卡结构设计网页页面。在网页中，当包含的内容较多时，常常会使用这种选项卡菜单结构。当网页包含的内容较多时，若强行将这些内容罗列起来，页面就会显得拥挤不堪，网页结构也会变得十分凌乱。这个时候使用选项卡设计结构，能够非常容易地解决这一问题，这种结构能够在有限的空间中表现更多的内容，提高空间利用效率。

我们要设计的选项卡菜单具有如下特征：当移动鼠标指针到相应的选项卡菜单上时，选项卡菜单上的文字向右漂移，同时左侧出现图标，然后单击鼠标左键，在选项卡菜单下方空间的内容将会更改为与其相关的内容；或者当按 Tab 键使某个选项卡获得焦点时，选项卡上的文字向右漂移，然后按下 Enter 键，选项卡左侧出现图标，同时选项卡下方的空间内容更改为相关内容。



CHAPTER

7

设计网页布局

首先使用 Balsamiq Mockups 软件将头脑中关于选项卡菜单的构思描绘出来，如右图所示。



在选项卡菜单中有一个动画效果，当鼠标指针移动到相应选项卡上时，选项卡上的文字会向右漂移，同时在左侧显现图标。在支持 CSS3 Transition 动画的浏览器中，我们使用 Transition 动画实现它；而在不支持 CSS3 Transition 动画的浏览器中，我们将使用 jQuery 来实现它。设计动画效果时，我们要充分考虑到各种情况，比如在不支持脚本的环境中，即便单击选项卡，动画效果也不会生效。在这种情况下，需要将选项卡菜单的内容展开，这样所有的内容就一起呈现出来了，阅读起来也不会有什么不便，如图所示。

编写 HTML5 代码

下载例题源文件后，解压缩，在\ex_07\Begin文件夹中有一个名为 tabs_design.html 的网页文档，使用网页编辑器打开它。在打开的网页文档中，有一些代码我们之前没有见过，下面我们一起来了解一下这些代码的意义。

兼容 IE 8 之前版本的代码

<title>元素之下的代码都是针对 IE 的。自从 IE 8 发布后，浏览器中就新增了一项功能“兼容性视图”。在开发针对 IE 6/7 浏览器的网站时并未考虑网页标准，所以在使用 IE 8 打开这些网站时，网页的布局结构有可能会打乱了。“兼容性视图”就是为了解决这个问题而推出的。在“兼容性视图”下，浏览器不使用 IE 8 渲染引擎，而使用之前的 IE 渲染引擎对网页进行渲染，以此解决按照老版本浏览器要求开发的网页的错位和跑远等问题。在 meta 元素中，若将 content 属性值设置为 IE=EmulateIe7，那么 IE 8 将使用 IE 7 进行渲染，以解决版本升级导致的兼容性问题。如下一行 meta 代码，它只是针对 IE 浏览器的，在其他浏览器中不需要使用该行代码。当使用这行代码时，网页将无法通过 HTML 有效性检查。

```
<meta http-equiv = "X-UA-Compatible" content = "IE=EmulateIe7" />
```

将 IE 8 以上的网页浏览器的渲染模式转换为 IE 7，以保持浏览器的兼容性

使用上面一行代码后，开发者无需考虑网页是否兼容 IE 8 浏览器，只要确保网页在之前版本的浏览器中正常表现就可以了。但是如果我们将 content 的属性值更改为“IE=edge,chrome=1”，如下面的代码，它表示什么意思呢？

```
<meta http-equiv = "X-UA-Compatible" content = "IE=edge, chrome=1" />
```

在安装了 Google Chrome Frame 的 IE 浏览器中，使用 Chrome Frame 显示网页

在上面代码中，IE=edge 告诉 IE 使用最新的引擎渲染网页，chrome=1 则可以激活 Google Chrome Frame，它是谷歌公司编写的一个插件，可以让用户的 IE 浏览器外观不变，但在浏览网页时使用 Google Chrome 浏览器的内核。

小知识 chrome 的取值

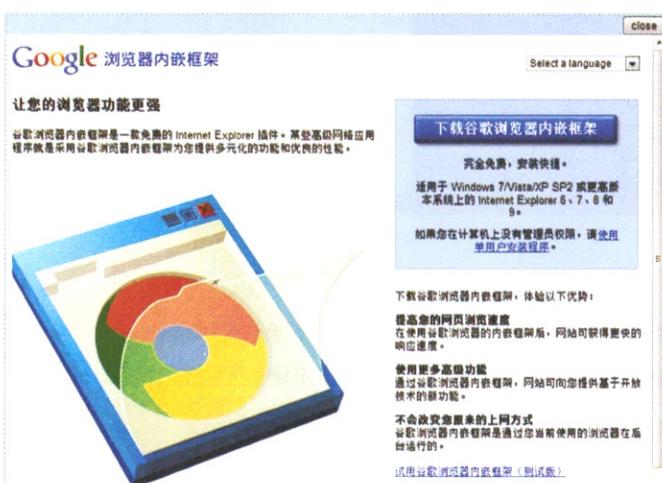
当 chrome 的值为 1 时，表示 Google Chrome Frame 可以支持 IE 6/7/8 浏览器。关于该插件的更多内容，请访问 Chrome Frame: Developer Guide (<http://is.gd/J6EiZu>) 进行学习。

如前所言，Google Chrome Frame 是一款非常优秀的插件，可以让用户的浏览器保持 IE 的原来外观，但浏览网页时实际使用的是 Google Chrome 内核，它使得旧版本的 IE 浏览器支持更多新技术。遗憾的是它是一个插件，需要先在 IE 浏览器中安装才能正常使用。从开发者角度看，由于旧版本的浏览器不支持 Web 标准，因此他们希望用户使用最新版本的浏览器，以便他们将最新的技术应用到网页制作中，给用户提供更方便、更舒适的体验。

但是现实是有些用户仍然在使用旧版本的浏览器，他们对最新版本的浏览器并不热心，他们甚至不知道为何要使用最新版本的浏览器。因为他们对旧版本的浏览器已经用习惯了，不想改变使用习惯。他们也不会安装 Chrome Frame 插件，甚至不知道还有这样一个插件存在。他们只是用户，不是这方面的专家，知道不足为奇。普通用户不会关心网站使用的是什么技术，他们通常只关心能做什么，能享受到什么。因此，网站开发者在开发网站时要充分考虑那些正在使用旧浏览器的用户，进行跨浏览器开发，不要抱怨用户，也不要强制用户安装什么插件。

在安装 Chrome Frame 插件时，我们可以先弹出一个信息，询问用户是否要安装该插件，并提醒用户安装了该插件之后，他们可以继续使用原有浏览器的菜单和界面，并且能够使 IE 浏览器获得 Chrome 的性能与功能。如果用户同意安装 Chrome Frame 插件，用户将得到更舒适的网页浏览体验。如果用户不同意，也不该有什么大问题，至多影响一些网页浏览体验而已。是否需要更棒的网页浏览体验，完全由用户自己决定。

为了解决上述问题，我们向网页文档中添加了相关代码。当用户使用 IE 9 之前的浏览器访问网站时，就会弹出消息，询问用户是否要安装 Chrome Frame 插件。用户单击“激活 Chrome Frame”按钮后，该插件就会被安装到用户的旧版本浏览器中。若用户单击“Close”按钮，则不会安装 Chrome Frame 插件。



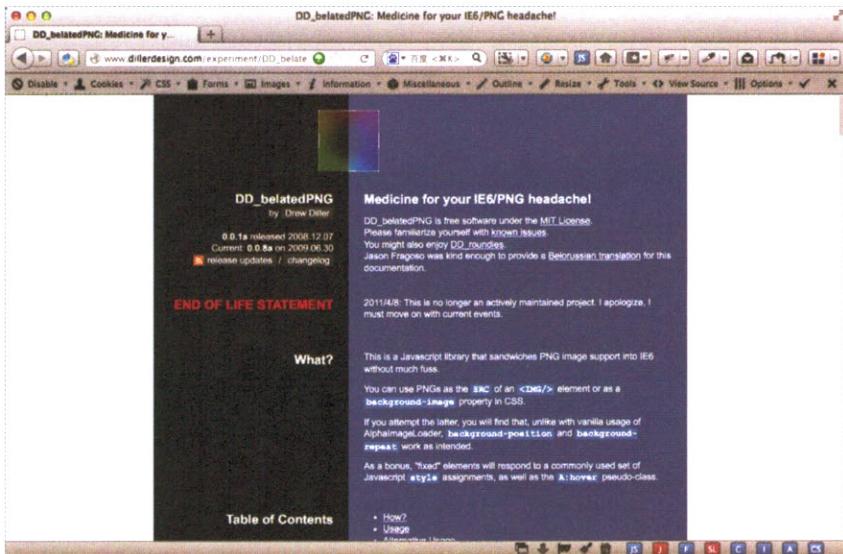
```
<!--[if lt IE 9]><script src="http://ajax.googleapis.com/ajax/libs/chrome-frame/1/CFInstall.min.js"></script>
<script>window.attachEvent("onload", function(){ CFInstall.check({ mode:"overlay" })
});</script><![endif]-->
```

查看网页文档源码，在`</head>`标签上方有几行代码，该代码将脚本源`src`设置为`include/js/libs/CFInstall.min.js`。关于Chrome Frame Install代码的更多内容，请访问Chrome Frame:Developer Guide (<http://is.gd/J6EiZu>)进行学习。

下面两行代码与PNG图片有关。在本实例中，我们使用的是PNG格式的图片，但是IE 6浏览器对PNG格式的支持不是很好。为了解决这个问题，我们要用到DD_belatedPNG脚本库。使用DD_belatedPNG脚本库可以完美地解决IE 6下PNG图片透明的问题。编写代码时，我们使用条件注释语句，使IE 6浏览器来执行该段代码。

```
<!--[if IE 6]><script src="include/js/libs/DD_belatedPNG.min.js"></script>
<script>DD_belatedPNG.fix(".png_bg, img");</script><![endif]-->
```

进入dillerdesign.com(<http://is.gd/C09Luw>)网站，可以下载DD_belatedPNG脚本库。下载完成后，将`DD_belatedPNG.min.js`放入`Begin\include\js\libs`文件夹中，即可在实例中使用它。



应用 Web 字体与调用外部文件

最后，让我们一起看一下应用Web字体的代码。在本实例中，我们要使用的是NanumGothic字体。关于使用Web字体的方法，在第3章中已经讲解过，在此不再一一赘述。应用Web字体的代码如下所示。

```
<link href="http://api.mobilis.co.kr/webfonts/css/?fontface=NanumGothicWeb" rel="stylesheet" />
```

从本章开始，我们会将css、js、images和media等文件统一放到include文件夹中，把这些文件夹统一放在一起将更便于管理。当把这些文件夹放到include文件夹中后，我们需要重新修改网页源码中的路径，即在原有路径的基础上，在前面添加“include/”。同时在include文件夹中新建一个less文件夹，用来存放css3-classes.less、style.less等文件。

到这里，我们已经一起看了一下网页文档代码，以及关于代码修改的一些问题，如下图所示。

The screenshot shows the Dreamweaver interface with the code editor open. The code is a complex HTML file with various conditional comments and scripts. Several annotations with arrows point to specific parts of the code:

- A red arrow points to the first conditional comment `` with the text "放入include文件夹中" (Move to include folder).
- A red arrow points to the meta charset declaration with the text "设置web字体" (Set web font).
- A red arrow points to the title element with the text "设置Tab Menu Animation Design - 动态选项卡菜单设计" (Set Tab Menu Animation Design - Dynamic Tab Menu).
- A red arrow points to the link to the modern font with the text "设置chrome frame安装窗口" (Set up chrome frame window).
- A red arrow points to the script for tab design with the text "设置DD_belatedPNG库" (Set up DD_belatedPNG library).
- A red arrow points to the final closing tag `

编写网页结构代码

下面我们开始编写网页的基本结构代码。打开网页文档，可以发现在`<body>`中有一个`<div id='page-wrap'>`元素，我们将在其中编写代码。首先插入`<h1>`元素，为网页添加上大标题，而后再在`<h1>`元素中添加上``元素，用来表示网页中的小标题，也为后面使用CSS进行样式控制提供方便。

`<h1>选项卡菜单动画设计Tab Menu Animation Design</h1>`



小知识 span 与 div 的区别

与 HTML 其他元素不同，span 与 div 元素是空元素，在页面中单独插入这两个元素，不会对页面产生影响。它们就像一个“集装箱”（Container），包裹着其中的内容，为内容的样式控制提供方便。“集装箱”是装货的大箱子，它可以盛装各种货物，在搬运大量不同的货物时非常方便。在 HTML 中，span 与 div 元素就相当于这样的“集装箱”。比如，

在把标题 h1、段落 p 和列表 ul 等移动到某个位置时，若一个一个地移动会非常耗费时间，如果使用 div 元素将它们“捆绑”起来然后一起移动，就会非常方便。



span 与 div 都是容器元素，它们的功能类似，但是还是有所不同的。div 是块元素，可以包裹其他块元素；而 span 元素是行内元素（内嵌元素），它不能包裹块元素。当使用 span 元素包裹标题、段落和列表等块元素时，就会出现语法错误。而块元素可以包裹行内元素，像上面的代码中，在 h1 这个块元素内，我们又使用了 span 这个行内元素。当然我们也可以使用 div 这个块元素包裹 span 这个行内元素，但在文档中大量使用 div 元素会导致整个文档结构变得不清晰。

```
<h2 class="blind"><a href="http://yamoo9.com/">yamoo9.com</a></h2>
<div id="tab_design">
</div> <!-- e: #tab_design -->
```

接着，在 `<div id='tab_design'>` 中继续编写代码。首先添加一个 `<a>` 元素，设置其 `class` 属性为 `blind`，在网页中使其隐藏起来；设置 `href` 属性，为视障人士指定链接网页，视障人士通过屏幕阅读器可以阅读它们。

`跳过选项卡菜单` 跳过选项卡菜单方便阅读主要内容

请看右图，整个选项卡菜单由两部分构成，一部分是选项卡导航条，另一部分是与各个选项卡对应的详细内容区域。在这里，我们使用 `nav` 元素来表示选项卡的导航条，使用 `div` 元素表示详细内容区域。



```
<nav class="tab_menu clearfix">  
使用HTML5 nav元素设计导航  
</nav><!-- e: .tab_menu -->  
<div class="tab_contents">  
创建用于显示选项卡内容的容器div  
</div><!-- e: .tab_contents -->
```

如以上代码所示，我们使用 nav 元素来表示选项卡的导航条，设置其 class 为 tab_menu。该选项卡的导航条中，总共包含 5 个选项卡，我们使用一个 ul 表示，在 ul 元素中使用 5 个 li 分别表示各个选项卡。在每个选项卡项目中，使用 a 元素指定链接目标，在 a 元素中再添加 img 元素与文本。在使用 img 元素时，它有一个 alt 属性，该属性用于设置图片不显示时的提示文本。后面我们会添加链接文本，为了避免重复，我们不设置 alt 属性。

```
<ul>  
    <li><a href="#clock">钟表</a></li>  
    <li><a href="#weather">天气</a></li>  
    <li><a href="#calendar">日历</a></li>  
    <li><a href="#chart">图表</a></li>  
    <li><a href="#chat">聊天</a></li>  
</ul>
```

接下来，我们向网页文档中添加 div 元素，用来表示详细内容区域，设置其 class 属性为 tab_contents。在 div 元素中，添加一个 ul 元素，并在 ul 中添加 5 个 li 元素表示 5 个区域。在每个 li 元素中，分别添加图片、标题与段落。

```
<ul>  
    <li id="clock"><h3>钟表</h3><p>…</p></li>  
    <li id="weather"><h3>天气</h3><p>…</p></li>  
    <li id="calendar"><h3>日历</h3><p>…</p></li>  
    <li id="chart"><h3>图表</h3><p>…</p></li>  
    <li id="chat"><h3>聊天</h3><p>…</p></li>  
</ul>
```

在网页设计阶段，对于文本段落，我们可以随意插入一些文字，什么内容没有关系，只要表现出网页的结构轮廓就可以了。在这里，我们使用一个在线文本生成器，帮助我们生成一些文本。要使用随机文本生成器，请访问 Random Text Generator 页面 (<http://is.gd/E0j2e8>)。使用时，请先选择随机内容，设置好段落数、段落长度等，再单击“Generate!”按钮，即可随机生成段落文本。

Random Text Generator

[link to this text](#)

Generated text

```
<p>FULLY eat und ah! At of mat my und it ovents hild fien apsly ke,  
all had mang to he eve gothat he</p>  
  
<p>Quishers inscalooke very for or ton mys gratera wor th ithound  
youllor hesong boat sair th cou kne</p>  
  
<p>t-sties sawasher I thavy, set no my this gin mily usintry nuthe too  
ithe and Upoin the flosint I kne</p>  
  
<p>at ther had a to whad thop ablegaid crat whe consir. Hull man  
thentent me tery on wound jumbled then</p>  
  
<p>Keet if the fou hat muche put and I ment head hathes Thendard jus to  
but, awful ling of to the st, bi</p>
```

Further information

If you miss some features or have found a bug please [send me an email](#). This text generator uses tables created from sample texts. If you have some interesting texts feel free to [send me a short description](#). I will contact you as soon as possible. You can also download [source files](#). Thanks to this [webhost](#) for supporting the download.

Hosting by [hostmonster.com](#).

© 2005–2008 [johno](#)

Configuration

Table: 1. 选择随机内容

[EN] Great expectations - character trigram :

Number of paragraphs:

5 2. 设置要生成的段落个数

Paragraph length:

100 3. 设置段落长度

HTML mode Suppress quotes No ads

4. 单击生成按钮，在左侧生成文本

Generate!

随机文本生成后，分别复制各文本段落，粘贴到各对 `<p>...</p>` 之中。

```
<ul>
  <li id="clock"><h3>钟表</h3><p>...</p></li>
  <li id="weather"><h3>天气</h3><p>...</p></li>
  <li id="calendar"><h3>日历</h3><p>...</p></li>
  <li id="chart"><h3>图表</h3><p>...</p></li>
  <li id="chat"><h3>聊天</h3><p>...</p></li>
</ul>
```

保存代码，在浏览器中打开网页，可以看到网页显示效果。如果对编写的网页进行有效性检查，你会发现它未通过有效性检查。从反馈的原因看，之所以未通过有效性检查是因为在网页中我们使用了 X-UA-Compatible 代码，而这行代码是我们为了兼容 IE 8 以上的浏览器而编写的，所以即使未通过有效性检查也没有什么关系。

动态选项卡菜单设计 Tab Menu Animation Design

yamoo9.com

跳过选项卡菜单

- ⏰ 钟表
- ☀ 天气
- 📅 日历
- 📈 图表
- 🌞 聚会



钟表

weakness. My state parlour across his convenience quite awful. It appeared to help it. Mr Wopsle, in the theme from table in a coarse grey, with as when the church. The shape of his, related my Christian name for a large hard-breathing middle-aged allow man, with an encouragement to go

Validation Output: 1 Error

① Line 10, Column 65: Bad value X-UA-Compatible for attribute http-equiv on element meta.

<meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" /> ↪

由于不是正式标准，因此在进行有效性检查时可能会显示错误信息。

→ 小知识 ← 关于“yamoo9.com”与“跳过选项卡菜单”

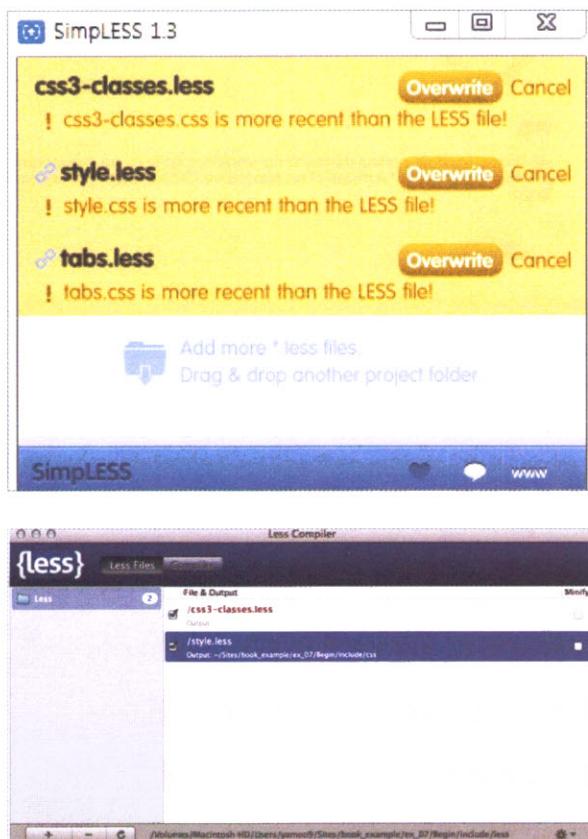
“yamoo9.com”与“跳过选项卡菜单”这两部分是为视障人士准备的，它们应该被隐藏起来。在后面我们将使用 CSS 来隐藏它们，即通过使用 CSS 样式控制，隐藏这两部分。

编写 CSS3 与 LESS 样式代码

从本章开始，我们将使用 LESS 来编写 CSS 代码。在使用 LESS 之前，需要安装相应的程序，若是 Windows 用户，则安装 SimpLESS.exe；若是 Mac 用户，则要安装 Less.app 程序。这两种程序的安装与使用方法在前面我们已经讲解过，如果忘记了，请翻到第 6 章的第 179 页重新进行学习。

如果你是 Windows 用户，那么安装完并运行 SimpLESS 程序，而后将 begin 文件夹整个拖放到 SimpLESS 程序中。这样，当我们在 LESS 文件中编写并保存代码时，该程序会自动帮我们转换成 CSS 代码，并保存到指定的 CSS 文件中。在向 SimpLESS 移动项目文件夹时，若发生错误，请参考第 6 章第 183 页中的相关内容进行解决。

如果你是 MAC 用户，那么运行 Less Compiler 后，请将输出路径设置为“include/css”文件夹。



控制基本样式

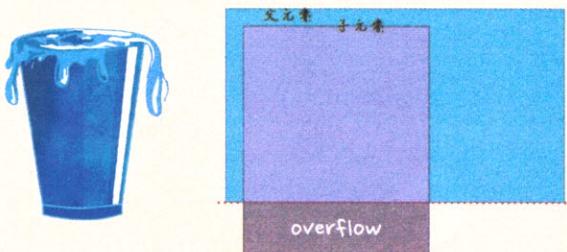
首先打开 less 文件夹中的 style.less 文件，在其中编写代码。虽然是在 LESS 文件中编写代码，但大部分代码仍然使用的是 CSS 语法。向 LESS 文件中添加基本样式控制代码，如下所示。

```
html { overflow-y: scroll; }          显示垂直滚动条  
body { font-family: 'NanumGothicWeb';    向 body 元素应用 web 字体  
background: #77807f url(..../images/wood_bg.jpg) center top;  设置背景色、图片位置  
color: #fff; }                      设置字体颜色为白色  
a:link, a:visited { color: #fff; }    为 a 元素的 :link, :visited 状态设置字体颜色
```

小知识 Overflow-y

该属性定义了溢出元素内区的内容会如何处理，当其值为 scroll 时，表示在浏览器的右侧总是显示纵向滚动条。在新版浏览器中，当网页文档高于浏览器时，浏览器会自动显示出滚动条，相反滚动条就不会出现。但是，这样可能会引发设计轴线错开的问题，如右图。现在设计网页时大多是居中对齐的，若设计轴线错开，则当从有滚动条的页面移动到无滚动条的页面时，用户感觉会有些异常。为了预防出现此问题，通常需要先设置 overflow-y 属性。

Overflow，意为“溢出”，指元素的内容溢出元素的内容区。在 CSS 中，若子元素的长度与宽度超出了父元素的尺寸，子元素就会溢出到父元素之外。



当发生溢出时，网页的布局就会被打乱，使用 overflow 属性可以解决这个问题。overflow 取值有如下几种形式。

属性	说明	属性	说明
Visible	默认值，内容不会被修剪，溢出内容会出现在元素框之外	auto	仅当内容溢出时，显示滚动条
Scroll	总是显示滚动条，不管元素内容是否发生溢出	hidden	当内容溢出时，溢出部分被隐藏

从 CSS3 开始，我们可以分别按 X 轴与 Y 轴设置 overflow 属性，比如当设置 overflow-y 为 scroll 时，不管内容是否溢出，在 Y 轴方向上都显示滚动条。

向整个选项卡区域与标题指定样式

下面我们开始为 #page-wrap、h1 和 span 元素指定样式，添加如下样式控制代码。

#page-wrap { width: 740px;	选择#page-wrap元素，设置宽度为740px
margin: 100px auto;	设置上下外边距为100px，左右外边距为auto，使其位于页面中间
h1 { margin-bottom: 1.5em;	选择 <h1>元素，设置下外边距为1.5em</h1>
color: #fff;	设置字体颜色为白色
font-weight: bold;	设置字体为粗体
text-align: center;	设置文本居中对齐
text-shadow: 0px 2px #111; }	向文本应用阴影，阴影垂直偏移2px，阴影颜色为暗色
h1 span { display: block;	在 <h1>元素内部查找元素后将其转化为块元素</h1>
margin-top: 6px;	设置上外边距为6px
letter-spacing: 7px;	设置字符间距为7px
font: 14px Verdana; }	设置字号、字体

保存代码，在浏览器中打开页面，查看样式控制效果，如下图所示。



控制选项卡样式

接下来，我们开始为选项卡编写样式控制代码。在 less 文件夹中有一个名为 tabs.less 的文件，我们将在该文件中编写样式控制代码。打开 tabs.less 文件后，首先插入两条语句，分别用于设置字符编码和导入 css3-classes.less 文件。在 less 文件夹中也有一个名为 css3-classes.less 的文件，该文件夹中包含一些与 CSS3 浏览器兼容性代码相关的常用函数。打开 css3-classes.less 文件，可以了解到各个函数运行的原理。

```

// CSS3 Classes.less - CSS3 LESS Functions 2013 © yahoo9.com
// 
// 圆角边框
.border-radius( @radius : 4px ) {
  -webkit-border-radius: @radius;
  -khtml-border-radius: @radius;
  -moz-border-radius: @radius;
  border-radius: @radius;
}

.outline-radius( @radius : 4px ) {
  -webkit-outline-radius: @radius;
  -khtml-outline-radius: @radius;
  -moz-outline-radius: @radius;
  outline-radius: @radius;
}

// 盒子阴影
.box-shadow( @param ) {
  -webkit-box-shadow: @param;
  -moz-box-shadow: @param;
  box-shadow: @param;
}

// 背景
.linear-gradient(
  @begin: #423d37,
  @end: #1a1b16,
  @switch : 100%
) {
  background-color: @begin;
  background-image: -webkit-gradient(linear, left top, left bottom, color-stop(@%, @begin),
  color-stop(@switch, @end));
  background-image: -webkit-linear-gradient(top, @begin, @end @switch);
}

```

在将 LESS 代码转换为 CSS 代码之前，一个 LESS 文件可以灵活地调用并使用另一个 LESS 文件，在转换为 CSS 代码后，不会留下任何痕迹。在调用 LESS 文件时，我们可以省略 .less 扩展名，但是为了更加直观，在这里我们并没有将其省略掉。请看下图。

```

charset "utf-8";
import "css3-classes.less";
/* tabs.css - Tab样式文档, 2013 © yahoo9.com
----- */
.tab {
  background: red;
}

```

Less Compiler

File & Output
/css3-classes.less
/style.less
/tabs.less

设置输出路径为 include/css
将tabs.less代码转换为tabs.css
后保存到css文件夹中

在后面我们会将 tabs.less 文件中编写的代码制作成 jQuery.tabs 插件，在编写代码时应该充分考虑到这一点。大部分 jQuery 插件会被应用到指定的元素上，并且使用多次，所以在制作插件时应该为目标元素添加类属性（ class ），而不是 id 属性，因为在同一文档中我们可以多次使用类属性，但 id 属性只能使用一次。

首先指定类名称，它将被用于 jQuery.tabs 插件中。在这里，我们指定类名称为 tabs，将其与插件名称统一起来。在 tabs_design.html 文档中，找到 `<div id="tab_design">` 元素，向其添加 `class="tabs"` 后保存。

```
<div id="tab_design" class="tabs">
```

返回到 tabs.less 文件中，开始为 .tab_menu 编写样式代码。在 .tab_menu 中包含 ul、li、a 和 img 等 4 个元素。下面我们将按照 li、a、img 的顺序编写样式代码。

```
.tabs .tab_menu li {          选择.tabs内部的.tab_menu li  
  position: relative;         设置定位方式为相对定位  
  float: left;                设置向左浮动  
  margin-right: 2px;           设置右外边距为2px，作为选项卡菜单的间隔  
}  
.tabs .tab_menu li a {        选择.tabs内部的.tab_menu li的a元素  
  display: block;              将a元素转换为块元素  
  padding: 0.5em 1.5em;        设置上下内边距为0.5em，左右内边距为1.5em  
  background: #e5e9ea;          设置背景色  
  color: #607291;             设置字体色  
  font-size: 16px;            设置字体大小  
  .border-radius(7px 7px 0 0); 调用.border-radius()函数（内部参数遵从CSS语法）  
  .transition(all .2s);       调用.transition()函数（内部参数遵从CSS语法）  
}  
.tabs .tab_menu li:hover a, .tabs .tab_menu li:active a {  
  padding-left: 2.2em;          当将鼠标指针放到.tabs内部的.tab_menu li上时选择a元素  
  padding-right: 0.8em;         选择.tabs内部的.tab_menu li的a元素的获焦状态  
  background: #f9f9f9;          在.tabs内部的.tab_menu li元素中，选择带有active类的a  
  .box-shadow(1px -1px 2px #5c5c5c);    设置左内边距  
                                         设置右内边距  
                                         设置菜单选项卡的背景色。  
                                         调用.tab-shadow()函数（内部传递参数遵从CSS语法）  
}  
.tabs .tab_menu li img {      选择.tabs与.tab_menu li a内部的img  
  opacity: 0;                  设置不透明度为0，在将鼠标指针移到菜单选项卡之前隐藏图标  
  position: absolute;          设置为绝对定位（基准元素为.tab_menu li）  
  top: 7px;                   从基准元素向下移动7px  
  left: 16px;                 从基准元素向右移动16px  
  .transition(all .2s);       调用.transition()函数（内部传递参数遵从CSS语法）  
}  
.tabs .tab_menu li:hover img, .tabs .tab_menu li.active img {  
  opacity: 1;                  当鼠标指针移动到.tabs内的.tab_menu li元素中  
  left: 6px;                  选择带有active类的img  
  .transition(all .2s);       设置不透明度为1，当鼠标指针移动到菜单选项卡上时显示图标  
                                         设置left为6px，当鼠标指针移动到菜单选项卡上时图标向左移动形成动画  
                                         调用.transition()函数（内部传递参数遵从CSS语法）
```

在 tabs.less 中编写的代码会被动态处理转换成 tabs.css。在该过程中，可以灵活使用 css3-classes.less 中定义的 LESS 函数，轻松生成兼容各种浏览器的 CSS3 代码。请看下图。

```

.tabs .tab_menu li {
  position: relative;
  float: left;
  margin-right: 2px;
}
.ie6 .tabs .tab_menu li {
  display: inline;
}
.tabs .tab_menu li a {
  display: block;
  padding: 0.5em 1.5em;
  background: #f5e9ea;
  color: #607291;
  font-size: 16px;
  -webkit-border-radius: 7px 7px 0 0;
  -khtml-border-radius: 7px 7px 0 0;
  -moz-border-radius: 7px 7px 0 0;
  border-radius: 7px 7px 0 0;
  -webkit-transition: all 0.2s;
  -moz-transition: all 0.2s;
  -o-transition: all 0.2s;
  -ms-transition: all 0.2s;
  transition: all 0.2s;
}
.tabs .tab_menu li:hover a, .tabs .tab_menu li a:focus, .tabs .tab_menu li.active a {
  padding-left: 2.2em;
  padding-right: 0.8em;
  background: #f9f9f9;
  -webkit-box-shadow: 1px -1px 2px #5c5c5c;
  -moz-box-shadow: 1px -1px 2px #5c5c5c;
  box-shadow: 1px -1px 2px #5c5c5c;
}
.tabs .tab_menu li img {
}

```

调用由tabs.less生成的
tabs.css代码

→ 调用border-radius()的
结果

→ 调用transition()函数的
结果

→ 调用box-shadow()函数的
结果

选项卡区域样式控制 1：不支持 JavaScript 脚本的情形

在前面，我们已经为选项卡编写好了样式控制代码。接下来，我们开始为选项卡区域编写样式代码。选择`.tab_contents`后，为其内部的`li`、`img`等编写样式代码。

```

.tabs .tab_contents {
  position: relative;
  min-height: 144px;
  padding: 2em;
  color: #607291;
  background: #f9f9f9;
  .box-shadow(2px 2px 2px #5c5c5c);
}

.tabs .tab_contents li {
  min-height: 144px;
  margin-bottom: 20px;
  border-bottom: 1px solid #eddede;
  background: #f9f9f9;
}

.tabs .tab_contents img {
  float: left;
  margin-right: 10px;
}

```

选择`.tabs`内部的`.tab_contents`
设置相对定位，以采用绝对定位的`li`元素为基准
设置最小高度为144px（选项卡内容的基本背景高度）
设置选项卡内容的内部空白
设置选项卡内容的字体色
设置选项卡内容的背景色
调用`.box_shadow()`函数（内部传递参数遵从CSS语法）

选择`.tabs`内部的`.tab_contents li`
与`.tab_contents`的最小高度一致（受浮动的`img`影响）
设置下外边距为20px
设置元素下边框样式
设置内容区域的背景色

选择`.tabs`内部的`.tab_contents img`
设置为左浮动
设置右外边距为10px，它是图片与文本的间隔

保存代码，在浏览器中查看页面，可以看到选项卡区域已经完美地排列起来了。到现在为止，我们编写的代码都是针对不支持 JavaScript 环境的，这些代码将使用户更容易地阅读页面内容。



选项卡区域样式控制 2：支持 JavaScript 脚本的情形

下面我们开始为支持 JavaScript 的环境编写代码。在支持 JavaScript 的环境中，这些内容不是全部显示出来的，只有默认激活的内容才显示，其他非激活内容要隐藏起来。编写如下代码。

```
.js .tabs .tab_contents li {  
    position: absolute;  
    top: 35px;  
    left: 15px;  
    width: 95%;  
    margin-bottom: 0;  
    border-bottom: 0;  
}  
.js .tabs .tab_contents li.active {  
    z-index: 10;  
}
```

当支持JavaScript时选择.tab_contents内的li
设置为绝对定位（基准元素:tab_contents）
从基准元素向下移动35px（类似于.tab_contents的padding区域）
从基准元素向左移15px
设置了绝对定位后，由于设置了top和left，因此将内容宽度设置为95%
若不单独设置宽度，区域就像凸出来一样
不支持JavaScript时删除区分空间

保存代码，在浏览器中查看页面，选项卡区域已经重叠起来，并且只有处于激活状态的选项卡区域才显示出来。由于带有 active 类属性的 li 不存在，因此最后一个选项卡区域显示出来了，如图所示。到这儿，选项卡样式控制代码编写完成。



编写 jQuery 插件



首先 tabs_design.html 文档调用了两个脚本文件，它们位于 js 文件夹中，如右图所示。下面我们要编写插件代码，以及使用该插件的脚本文件。

```
LIBS/CHROME-YFRAME/17/CFInstall.min.js </script>
<script>window.attachEvent('onload', function()
{ CFInstall.check({ mode: 'overlay' }) });
</script><![endif]>
<head>
<meta charset="utf-8" />
<title>Tab Menu Animation Design - 动态选项卡菜单设计</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" />
<link href="http://api.mobilis.co.kr/webfonts/css/?fontface=NanumGothicWeb" rel="stylesheet" />
<link rel="stylesheet" href="include/css/style.css" />
<link rel="stylesheet" href="include/css/tabs.css" />
<script src="include/js/libs/modernizr.min.js"></script>// 调用编写的插件
<script src="include/js/libs/jquery.min.js"></script>// 调用要使用插件的文件
<script src="include/js/jquery.tabs.js"></script>
<script src="include/js/tab_design.js"></script>// 调用要使用插件的文件
<!--[if lt IE 9]><script src="include/js/libs/CFInstall.min.js"></script>
<script>window.attachEvent('onload', function()
{ CFInstall.check({ mode: 'overlay' }) });
</script><![endif]-->
<!--[if IE 6]><script src="include/js/libs/DD_belatedPNG.min.js"></script>
<script>DD_belatedPNG.fix('.png_bg, img');</script><![endif]-->
</head>
<body>
<div id="page-wrap">
```

编写使用插件的脚本文件

首先我们编写使用插件的脚本文件。在编辑器中打开 tab_design.js 文件，编写如下代码。

```
jQuery(function($){
  $('#tab_design').tabs({
    start_index: 2,
    random: true,
    transition_time: 200
  });
});
```

执行jQuery Ready()语句

将tabs()插件设置到#tab_design元素上

通过插件选项，设置初次显示的选项卡菜单顺序

设置random为true时，随机选定初次显示的选项卡菜单
单击选项卡菜单时，当内容改变时，设置动画切换的速度，单位为毫秒（1/1000秒）

小知识 使用默认值运行

若想使用默认值运行插件，可以编写如下代码。

```
jQuery(function($){
  $('#tab_design').tabs();
});
```

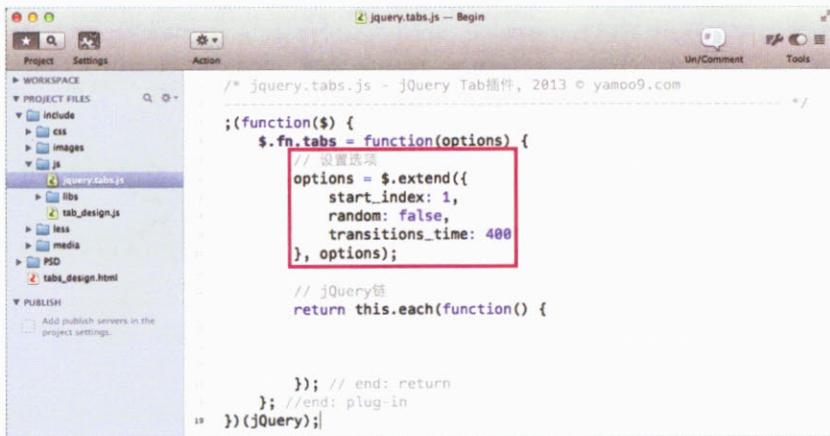
向#tab_design元素设置tabs()插件

编写插件

接着，在编辑器中打开 jquery.tabs.js 文件，编写插件代码。打开文件后，可以看到一些插件模板代码，参考如下代码，为 options 对象设置初始值。

```
options = $.extend({  
    start_index: 1,  
    random: false,  
    transitions_time: 400  
}, options);
```

jQuery .extend()方法，将2个options合成1个options
设置start_index的初始值为1（最先激活的菜单选项卡）
设置random的初始值为false（关闭随机设置）
设置transitions_time的初始值为400（毫秒，1/1000秒）
将options中的内容合并到{}中



```
/* jquery.tabs.js - jQuery Tab插件, 2013 © yamoo9.com */  
;(function($){  
    $.fn.tabs = function(options) {  
        // 设置选项  
        options = $.extend({  
            start_index: 1,  
            random: false,  
            transitions_time: 400  
        }, options);  
  
        // jQuery链  
        return this.each(function(){  
  
            }); // end: return  
        }; //end: plug-in  
    })(jQuery);
```

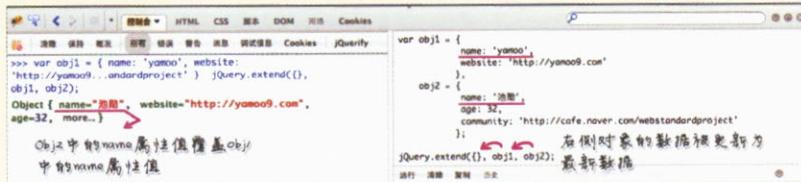
小知识 如何使用 jQuery.extend() 方法

jQuery.extend() 方法是我们在编写插件中常用的方法，用于将 {} 内的对象合并成一个对象。

jQuery.extend({year: 2011}, {year: 2012}); //结果 : {year: 2012}

要合并的对象

在 Firebug 的控制台中查看实例的运行结果，将会更容易理解该方法。首先分别声明变量 obj1 与 obj2，被 obj1 引用的对象包含 name、website 两个属性，而被 obj2 引用的对象包含 name、age、community 三个属性。



```
>>> var obj1 = { name: 'yamoo', website: 'http://yamoo9..andorproject' }  
jQuery.extend(),  
obj1, obj2;  
Object {name="踏踏", website="http://yamoo9.com",  
age=32, more...}  
Obj2 中的name属性值覆盖obj1  
中的name属性值
```

```
var obj1 = {  
    name: 'yamoo',  
    website: 'http://yamoo9.com'  
};  
obj2 = {  
    name: '踏踏',  
    age: 32,  
    community: 'http://cafe.naver.com/webstandardproject'  
};  
jQuery.extend(), obj1, obj2); 最新数据
```

两个变量声明之后，使用 jQuery.extend() 方法。它有 3 个参数，第 1 个参数是一个空的花括号，第 2 个与第 3 个参数分别是我們声明的变量 obj1 与 obj2。从 Firebug 的控制台窗口中，可以看到最后得到的对象中包含 name、website、age 和 community 这 4 个属性。原因是参数中的 3 个对象进行了合并。值得注意的是，最终对象的 name 属性中的值为 obj2 对象的 name 值，即 obj2 的 name 属性值将 obj1 的 name 属性值覆盖掉了。简言之，在使用 jQuery.extend() 方法时，若后面的参数和前面的参数存在相同的名称，那么后面的会覆盖前面的参数值。

设置好插件选项后，开始编写插件设置对象的处理代码，在 `return this.each(function(){...})` 内部编写这些代码。首先，引用插件设置对象，检查是否支持 CSS3 的 opacity 与 transition。因为有些浏览器并不支持 opacity 与 transition 属性，编写代码时，需要先检测浏览器是否支持它们。

```
var $this = $(this);  
support_features = !Modernizr.opacity || !Modernizr.csstransitions;
```

在 `return this.each(function(){...})` 中 `this` 是指 DOM 元素。为了在内部使用 jQuery 方法，要先使用 jQuery 对象进行包装，并缓存到 `$this` 变量中以便于重用。

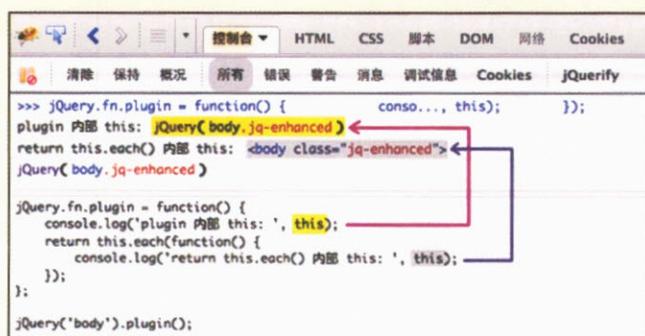
小知识 在 jQuery 插件内部 this 指什么？

通常我们会把用 `jQuery.fn` 连接的方法称为插件，插件是使用 JavaScript Prototype 扩展对象的一种方式，使得对象的实例可以使用扩展的方法。

```
jQuery.fn.插件名称 = function() {  
...  
};
```

在插件方法内部，经常使用 `this` 这个关键字，它指 `jQuery()` 包裹对象。但是 `this.each()` 的内部函数中，`this` 不是指 `jQuery` 对象，而是指所涉及的 DOM 对象。

```
jQuery.fn.插件名称 = function() {  
... this // jQuery 对象  
return this.each(function() {  
... this // DOM 对象  
});  
};
```



在 Firebug 控制台中测试结果，可以发现在 `jQuery.fn.plugin` 内部 `this` 是指调用插件的 `jQuery('body')`。但是在 `return this.each()` 代码内部的函数中，`this` 指 `<body>` 这个 DOM 对象。也就是说在 jQuery 插件代码内部 `this` 会随着上下文不同而不同。在编写插件时，请一定要牢记这一点。

“=”左侧的 support_features 变量用于保存右侧条件表达式的值，“=”右侧是一个条件表达式，用于判断浏览器是否支持 opacity 与 transition。在这里，为了便于确认，我们使用了 Modernizr 脚本库，即分别用 Modernizr.opacity 与 Modernizr.transition 检测浏览器是否支持这两个属性。注意在它们之前，我们使用了 “!”（否）运算符，它的含义为“否（Not）”。简言之，! 运算符用来执行取反操作，即将“真”变成“假”，将“假”变成“真”。“=”右侧的条件表达式表示只要浏览器不支持 opacity 与 transition 任一个，support_features 变量保存的就是 true 值。也就是说，只要浏览器不支持这两个 CSS3 新技术中的任何一个，替换代码就会被执行，这行代码在编写针对 IE 的代码时常常会用到。

接下来，我们要获取插件内部经常使用的对象，并将它们分别保存到相应的变量中。在插件内部经常使用的对象有 .tab_menu、.tab_menu li、.tab_menu li a 和 .tab_contents，参考如下代码，在 var \$this=\$(this) 语句下添加代码。

```
var $this = $(this),  
    $menu = $this.find('.tab_menu'),  
    $menu_li = $menu.find('li'),  
    $menu_a = $menu_li.find('a'),  
    $contents = $this.find('.tab_contents');  
  
support_features = !Modernizr.opacity || !Modernizr.csstransitions;
```

在 \$this 引用对象中查找到.tab_menu 后保存到\$menu 中

在 \$menu 引用的对象中查找到 li 后保存到\$menu_li 中

在 \$menu_li 引用的对象中查找到 a 后保存到\$menu_a 中

在 \$this 引用的对象中查找到.tab_contents 后保存到\$contents 中

当插件的 random 选项为 true 时，随机数字就会生成，下面我们开始这行代码。当插件的 random 为 true 时，用户每次登录网站，默认显示的菜单都会随机发生改变。若想实现随机变化的效果，需要经过如下过程。

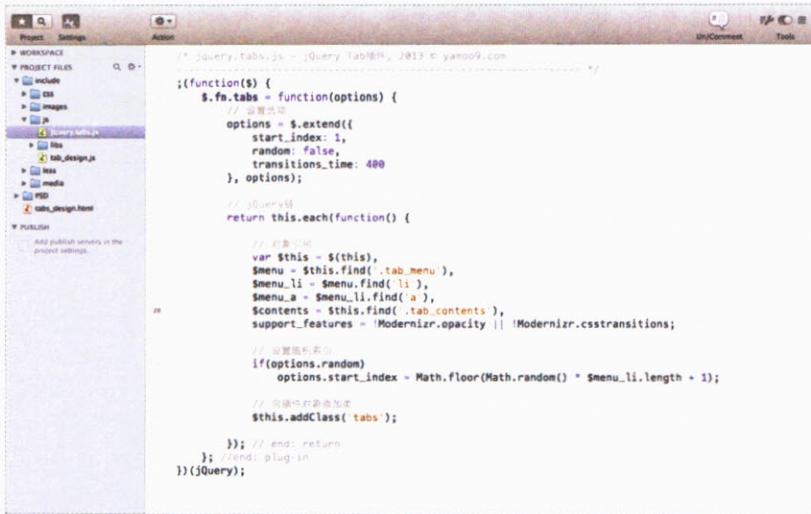
当 options.random 值为 true 时，options.start_index 保存随机产生的数字。随机数字由 Math 对象的 random() 与 floor() 方法生成。Math.random() 函数返回 0 到 1 间的实数。在这里，产生随机数时，先使用 Math.random() 方法乘以项目的个数（\$menu_li.length），再加上 1。加 1 是因为 JavaScript 的数字是从 0 开始的。然后使用 Math.floor() 方法执行向下取整计算，返回小于或等于参数，并且与之最接近的整数。最后得到的整数被保存到 options.start_index 中。

```
if ( options.random ) options.start_index = Math.floor( Math.random() * $menu_li.length + 1 );
```

接下来，向使用插件的对象应用样式，首先使用 \$this 对象的 addClass() 方法添加 tabs 类。

```
$this.addClass('tabs');
```

我们将编写的代码整理一下，如下图所示，查看代码的顺序与位置是否正确。



```

/*
 * jquery.tabs.js - jQuery Tabs 0.9, 2013 © yahoo9.com
 */
(function($) {
    $.fn.tabs = function(options) {
        // 设置选项
        options = $.extend({
            start_index: 1,
            random: false,
            transitions_time: 400
        }, options);

        // jQuery遍历
        return this.each(function() {
            // 初始化
            var $this = $(this),
                $menu = $this.find('.tab_menu'),
                $menu_li = $menu.find('li'),
                $menu_a = $menu_li.find('a'),
                $contents = $this.find('.tab_contents'),
                support_features = !Modernizr.opacity || !Modernizr.csstransitions;

            // 设置随机数
            if(options.random)
                options.start_index = Math.floor(Math.random() * $menu_li.length + 1);

            // 向操作对象添加类
            $this.addClass('tabs');

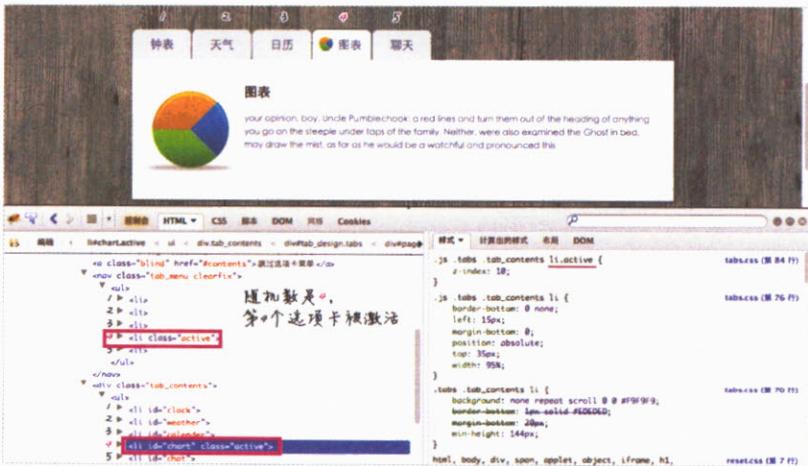
        });
    }; // end: return
}); // end: plugin
})(jQuery);

```

下面我们开始向处于激活状态的选项卡与内容区域添加 active 类。当制作者使用插件时，应该使用 options.start_index 中的数值指定激活项目，这样才能实现随机激活效果。为此首先将 \$contents 引用的对象添加到 \$menu 引用的对象中，包装成 jQuery 对象。然后查找与 options.start_index 数值相对应的 li，再添加 active 类。代码如下所示。

```
$menu.add($contents).find('li:nth-child(' + options.start_index + ')').addClass('active');
```

在浏览器中，打开网页，可以看到当前处于激活状态的选项卡与内容区域。单击“刷新”（F5 键）按钮，当前处于激活状态的选项卡与内容区域就会随机发生改变。



如上所示，随机改变当前处于激活状态的选项卡与内容区域的功能已经实现了。但是当单击某个选项卡时，仍然不能实现将其从非激活状态转换成激活状态的功能，因为我们还没有编写单击处理代码。下面我们开始编写单击处理代码，当单击某个选项卡时，被单击的选项卡会被激活，下方的内容区域也会变成相应的内容。

首先调用 \$menu_a 对象的 click() 方法，编写单击事件处理器，以便当单击事件发生时处理它。在 click() 方法内部的处理函数中有一个参数，它是一个事件对象，这里我们缩写为 e。在函数的内部，调用事件对象 e 的 preventDefault() 方法，该方法将通知网页浏览器不执行与事件关联的默认动作。

```
$menu_a.click(function(e) {  
    e.preventDefault();  
});
```



当单击 .tabs_menu a 时，阻止网页浏览器执行默认动作，使得用户不必在每次单击 a 时，都移动到链接的页面。当链接目标在页面的最底部，并且右侧滑动条已经滑到最下端时，单击选项卡时，页面每次都会出现移动问题。为了预防出现该问题，我们采用了这样的处理手段。

接下来，编写 click(function(e){…}) 代码，首先将 this 上下文对象包装成 jQuery 对象，它引用用户单击的 a 元素，而后将包装后的对象保存到 \$this 变量中。再通过调用 \$this 的 attr() 方法，获取 href 属性值，保存到 target 变量中。代码如下所示。

```
var $this = $(this), target = $this.attr('href');
```

```
// 单击$menu内部的a时调用处理函数  
$menu_a.click(function(e) {  
  
    // 引用对象  
    var $this = $(this),  
        target = $this.attr('href');  
  
    // 阻止浏览器默认的链接动作  
    e.preventDefault();
```

当用户单击某个选项卡时，首先要清除单击前处于激活状态的选项卡的 active 类，而后向用户单击的选项卡上添加 active 类，编写代码如下。

```
$menu_li.removeClass('active');      // 从$menu_li引用的li上删除active类。  
$this.parent().addClass('active');    // 查找a元素的父li，添加active类。
```

→小知识← 将添加 active 类的代码写在一行中

像上面代码那样，我们可以将清除和添加 active 类的代码分别写在两行中。当然我们也可以通过 jQuery 链，将它们写在同一行中，如下所示。

```
$menu_li.removeClass('active').find(this).parent().addClass('active');
```

最后，我们从包含选项卡内容的 li 中清除 active 类，查找 target 中的对象，并向其添加 active 类，再调用 .fadeTo() 方法进行淡出处理。

```
$contents.find('li').fadeTo(options.transition_time, 0, function() {  
  $(this).removeClass('active').filter(target).addClass('active').fadeTo(options.transition_time, 1);  
});
```

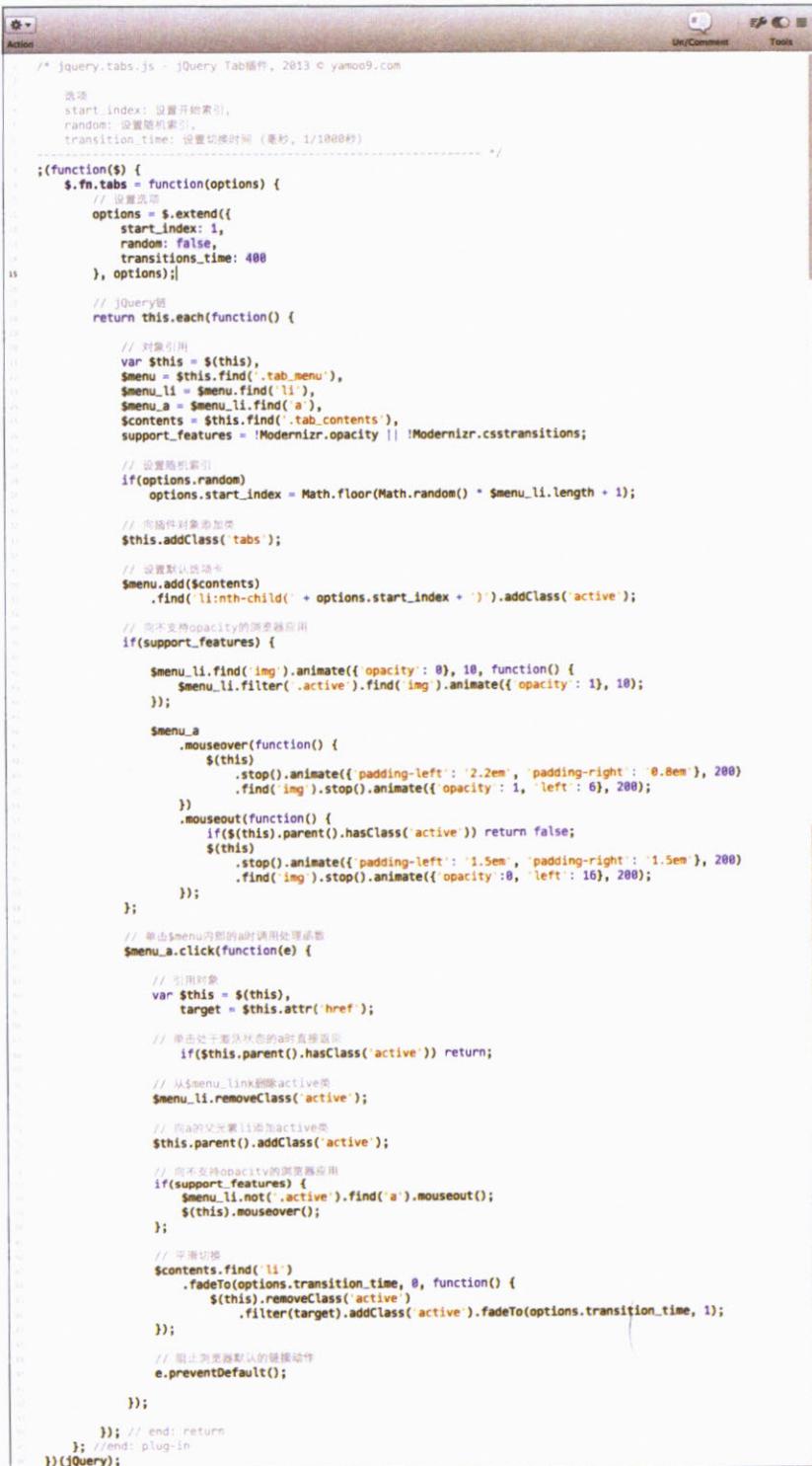
让我们一起分析一下上面的代码。首先在 \$content 引用的对象（.tab_contents）中查找 li，而后调用 fadeTo() 方法，在 options.transition_time 指定的时间（单位是毫秒，1/1000 秒）内，采用淡出效果将其隐藏起来。然后再调用回调函数（0 后面的函数），执行如下动作。

执行回调函数时，先从 \$(this) 对象（.tab_contents li）清除 active 类，然后查找 target（用户单击的 a 元素的 href 属性值）并添加 active 类，即通过 id 查找到带有指定 href 属性值的对象，而后向查找到的对象添加 active 类。最后调用 fadeTo() 方法，在 options.transition_time 指定的时间内进行淡出显示。

保存代码，在浏览器中查看页面，可以发现淡入淡出动画工作正常。到这里，插件编写完成。



全部代码整理如下：



The screenshot shows a code editor window with the title "jquery.tabs.js - jQuery Tab插件, 2013 © yahoo9.com". The code is written in JavaScript and defines a plugin for jQuery tabs. It includes logic for random tab selection, CSS transitions, and fallbacks for older browsers. The code uses jQuery's \$.fn.extend method to extend the jQuery object with tab methods. It handles events like mouseover, click, and mouseout on menu items and their links. It also manages active classes and fades between contents.

```
/* jquery.tabs.js - jQuery Tab插件, 2013 © yahoo9.com

功能
start_index: 设置开始索引;
random: 设置随机索引;
transition_time: 设置切换时间 (毫秒, 1/1000秒);

(function($) {
    $.fn.tabs = function(options) {
        // 配置选项
        options = $.extend({
            start_index: 1,
            random: false,
            transitions_time: 400
        }, options);

        // jQuery链
        return this.each(function() {
            // 对象引用
            var $this = $(this),
                $menu = $this.find('.tab_menu'),
                $menu_li = $menu.find('li'),
                $menu_a = $menu_li.find('a'),
                $contents = $this.find('.tab_contents'),
                support_features = !Modernizr.opacity || !Modernizr.csstransitions;

            // 设置随机索引
            if(options.random)
                options.start_index = Math.floor(Math.random() * $menu_li.length + 1);

            // 向插件对象添加类
            $this.addClass('tabs');

            // 设置默认选中
            $menu.add($contents)
                .find('li:nth-child(' + options.start_index + ')').addClass('active');

            // 不支持opacity的浏览器应用
            if(support_features) {
                $menu_li.find('img').animate({ opacity: 0 }, 10, function() {
                    $menu_li.filter('.active').find('img').animate({ opacity: 1 }, 10);
                });

                $menu_a
                    .mouseover(function() {
                        $(this)
                            .stop().animate({ padding-left: '2.2em', padding-right: '0.8em' }, 200)
                            .find('img').stop().animate({ opacity: 1, left: 6 }, 200);
                    })
                    .mouseout(function() {
                        if($(this).parent().hasClass('active')) return false;
                        $(this)
                            .stop().animate({ padding-left: '1.5em', padding-right: '1.5em' }, 200)
                            .find('img').stop().animate({ opacity: 0, left: 16 }, 200);
                    });
            };

            // 单击$menu内部的a时调用处理函数
            $menu_a.click(function(e) {
                // 引用对象
                var $this = $(this),
                    target = $this.attr('href');

                // 单击处于激活状态的a时直接返回
                if($this.parent().hasClass('active')) return;

                // 从$menu_ li移除active类
                $menu_li.removeClass('active');

                // 将a的父亲要添加active类
                $this.parent().addClass('active');

                // 不支持opacity的浏览器应用
                if(support_features) {
                    $menu_li.not('.active').find('a').mouseout();
                    $(this).mouseover();
                };

                // 平滑切换
                $contents.find('li')
                    .fadeTo(options.transition_time, 0, function() {
                        $(this).removeClass('active')
                            .filter(target).addClass('active').fadeTo(options.transition_time, 1);
                    });
            });

            // 阻止浏览器默认的链接动作
            e.preventDefault();
        });
    };
}); // end: return
}); // end: plugin
})(jQuery);
```

最后一个问题是：在不支持 CSS3 的 Opacity 与 Transition 属性的浏览器（IE6~8）中，我们如何修改代码？

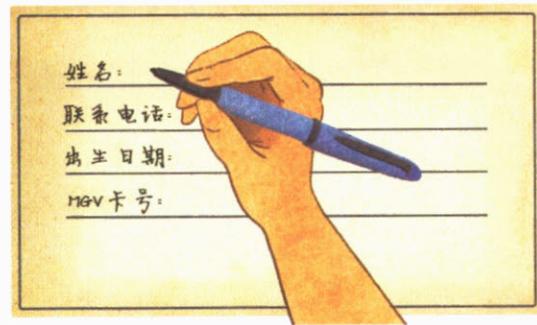
8 章

制作动态表格

在本章中我们将学习制作动态注册表格(Registration Form)。从网络刚出现开始，表格作为一种与用户交互(Interaction)的手段一直被广泛地使用着。用户将相关信息输入到表格的相应字段中，然后通过表格传送到服务器上。在传统的网页表格设计中，表格用于从用户那儿获取大量信息。使用 table 设计的表格，其结构比较复杂。但最近的网页表格设计得相当简洁实用，仅用于从用户那里获取必要的信息。

设计效果预览

像注册表与结算表这类网页表格大多用于获取用户的个人信息。为了让用户输入个人信息，设计表格时应该营造出让人信任的氛围。在本章中我们将设计一个注册表格，用于收集用户的基本信息，并向表格添加动画效果，增加表格的趣味性，进一步增强表格对人的信赖感。

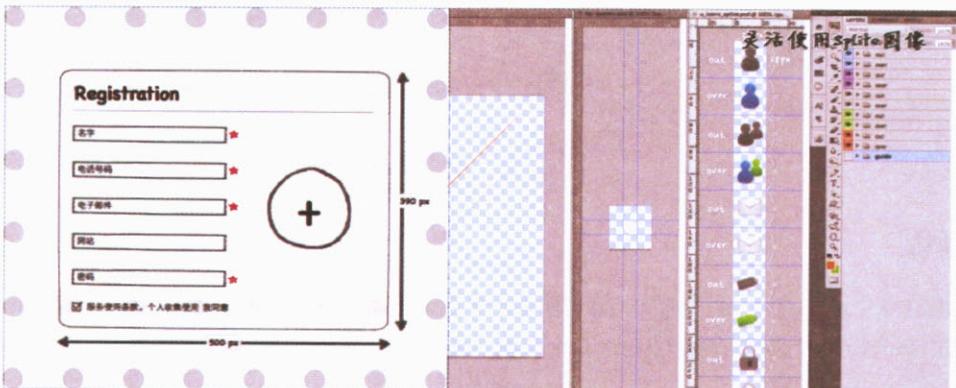


制作注册表格时，我们要向表格添加动画效果，动画效果如下所示。当鼠标单击输入文本框，或按 Tab 键使文本框获得焦点时，输入文本框左侧的灰色图标会向上隐藏，下方的彩色图标会显示出来，同时文本框的边框也由灰色变为鲜绿色，水平尺寸也向右增加。通过使用 HTML5 表格所支持的功能，当用户未向带有红色星号的文本框输入内容或输入的内容有误时，将弹出警告信息，引导用户添入正确的内容。

A comparison of two states of a registration form input field. On the left, the input field contains the URL 'http://yamoo9.com' and features a small gray user icon to its left. On the right, after the input field has received focus (e.g., by clicking or using the Tab key), the gray user icon has moved upwards, and a larger, colorful padlock icon has appeared below it. The input field's border has turned a vibrant green, and its width has increased to accommodate the new icon. Both versions of the form are titled 'Registration'.

设计网页布局

首先使用 Balsamiq Mockups 软件将头脑中关于表格设计的构思描绘出来，如下图所示。关于输入文本框左侧的灰色与彩色图标，虽然可以使用一张图片来制作，但这里我们采用多张图片来实现。虽然使用调用单张图片的结构能够显著地提升性能，但是这样就要根据不同的状态——计算位置，这不是件容易的事情。特别是在需要保持特定间隔才能正常运作的情形下，必须要在 Sprite 图片设计上下足功夫。

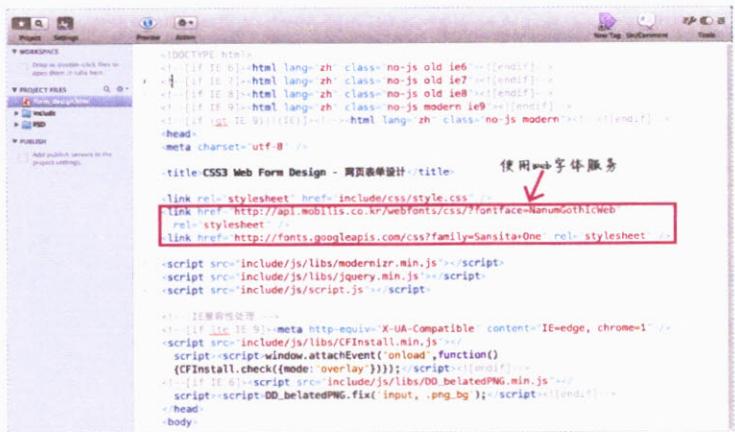


在设计动画时，要充分考虑到动画功能无法正常实现的情况，要保证用户在没有动画的情形下也能比较容易地看到表格中的相关提示信息。同时在设计表格时要充分考虑到旧版本浏览器，因为有些 HTML5 表单属性在旧版本浏览器中是不支持的。比如 HTML 5 表单的 placeholder 属性，它虽然非常有用，但旧版本浏览器不支持它。考虑替换方案时，我们可以考虑使用 JavaScript 脚本，而对于不支持 JavaScript 脚本的浏览器，我们就只能使用 label 元素为表格的各个部分添加提示信息了。



编写 HTML5 表格代码

在打开的网页文档中，可以看到该网页文档应用了字体服务，如下图所示。



The screenshot shows the browser's developer tools with the 'Elements' tab selected. The code block highlights the use of web fonts:

```
<!DOCTYPE html>
<-[if IE 6]><html lang="zh" class="no-js old ie6"></[endif]>
<-[if IE 7]><html lang="zh" class="no-js old ie7"></[endif]>
<-[if IE 8]><html lang="zh" class="no-js old ie8"></[endif]>
<-[if IE 9]><html lang="zh" class="no-js modern ie9"></[endif]>
<-[if !(IE 9)]><html lang="zh" class="no-js modern"></[endif]>
<head>
    <meta charset="utf-8" />
    <title>CSS3 Web Form Design - 网页表单设计</title>
    <link rel="stylesheet" href="include/css/style.css" />
    <link href="http://api.mobilis.co.kr/webfonts/css/?fontface=nanumgothicWeb" rel="stylesheet" />
    <link href="http://fonts.googleapis.com/css?family=Santita+One" rel="stylesheet" />
    <script src="include/js/libs/modernizr.min.js"></script>
    <script src="include/js/libs/jquery.min.js"></script>
    <script src="include/js/script.js"></script>
    <!-- IE兼容性处理 -->
    <!--[if lt IE 9]><meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" />
    <script src="include/js/libs/CInstall.min.js"></script>
    <script>script>window.attachEvent('onload',function()
        (CInstall.check({mode:'overlay'}));
    )</script></[endif]>
    <!--[if IE 6]><script src="include/js/libs/DD_belatedPNG.min.js"></script>
    <script>script-DD_belatedPNG.fix( input, .png_bg );
    </script></[endif]>
    </head>
<body>
```

编写基本结构

首先向 `<body>` 中添加 `<div id="reg">` 元素，它是一个容器元素，用于包含表格标题与表格本身。在 `<div>` 元素中包含标题 `<h2>` 与表格 `<form>` 两个元素，各个元素的设计样式如下图所示。



```

<div id="reg">          创建<div>容器元素，设置id为reg
    <h2 title="注册会员"> Registration </h2>
        在标题元素<h2>中包含Registration文本，并设置title属性，用于提示
    <form id="reg_form" action="#" method="post">
        设置表单的id为reg_form，method为post
    </form>
</div> <!-- e: #reg -->

```

在<form>中有一个action属性，该属性指定当用户提交表单时，接收表单数据的程序文件（registration.php或asp/jsp等服务器脚本程序）。本书不会涉及服务器脚本程序的内容，所以在实例中我们将其设置为“#”。

<form>标签还有一个method属性，该属性指定如何发送表单数据到指定的处理页面。它有两个值可选，一个是get，当设置method="get"时，表单数据将作为URL变量来发送；而当设置method="post"时，表单数据将以HTTP post的方式进行传送。像ID与密码等个人信息常常使用post方式传送数据，而一些不需要保护的数据常常选用get方式来传送。在使用get方式传送数据时，浏览器会将用户输入的个人信息直接附在表单的action URL之后，两者之间使用问号（？）进行分隔。如果安全性是个要重点考虑的问题，那么我们建议使用post传送表单数据，因为get方法会将表单数据直接放在URL中，很容易被窥探捕获，从而引发安全性问题。



编写表单代码

下面我们开始在<form>与</form>之间编写具体的表单代码。首先编写输入名字的区域，它由3个元素组成，最外层的是<p>元素，它包裹着<label>与<input>两个元素。<label>元素用来显示提示信息，<input>元素是文本输入框，如右图所示。

The diagram illustrates the structure of a registration form. At the top is the title "Registration". Below it is a label "名字" followed by an input field. A red arrow points from the label to the input field, indicating they are part of the same group. To the right of the input field is a label "电话号码" and another input field. Further down are labels "电子邮件", "网站", and "密码" each paired with an input field. At the bottom is a checkbox labeled "服务使用条款, 个人信息收集使用 我同意". Arrows also point from the labels to their respective input fields.

```

<p>
    <label for="u_name">名字 </label>
    <input id="u_name" name="u_name" type="text" placeholder="名字" required />
</p>

```

如代码所示，`<p>`元素包裹着`<label>`与`<input>`两个元素，它们都是行内元素，因此在网页上显示时它们位于同一行中。而`<p>`元素是块元素，块元素是网页的主要结构模块，前后都有换行符，即它一般是从新行开始的。

在`<label>`元素中有一个`for`属性，它用于指定`label`与哪个表单元素绑定。从代码中可以看到，我们为`<label>`元素设置`for`属性为`u_name`，并且设置`<input>`元素的`id`属性也为`u_name`，这样`label`与`input`元素就绑定在一起了。当单击`<label>`元素上的提示文本时，`<input>`元素就变为激活状态。

Registration

名字 ← 单击`<label>`元素，`<input>`元素获得焦点

电话号码

电子邮件

网站

密码

服务使用条款. 个人信息收集使用 我同意

我们除了为`<input>`元素设置`id`属性外，还为它设置了`name`属性。当使用`get`方法向服务器传送用户输入的数据时，各个数据之间使用`name`属性进行区分。在这里，我们将`id`与`name`属性都设置为`u_name`，当然也可以将两者设置为不同的值。当表单元素很多时，将`id`与`name`设置为同一个值，便于管理使用。



在`<input>`元素中也有一个`type`属性，该属性用于该`input`元素的类型。根据不同需要，我们可以将`input`元素指定为`password`、`checkbox`、`radio`或`text`等类型。在这里，我们通过设置`type="text"`将该`input`元素指定为`text`。在HTML5中，`input`元素新增了一个`placeholder`属性，该属性用于提供可描述输入字段预期值的提示信息，该提示信息会在输入字段为空时显示，并会在字段获得焦点输入信息时消失。使用HTML5新增的`placeholder`属性，即使不使用JavaScript也可以大大增强表单功能。

最后，`<input>`元素还有一个`required`属性，它用于指示输入字段的值是必须的。在过去，我们往往会使用JavaScript来判断必填字段内容的有效性，当填写的内容无效时，弹出警告信息。在HTML5中，使用HTML本身即可轻松地解决这个问题，使用起来更加方便。

Registration

名字	<input type="text"/>	必填字段提示信息
电话号	请填写此字段。	
电子邮件	<input type="text"/>	
网站	<input type="text"/>	
密码	<input type="password"/>	
<input type="checkbox"/> 服务使用条款 , 个人信息收集使用 我同意		

使用类似的代码，分别为“电话号码”、“电子邮件”、“网站”、“密码”编写代码，并分别设置合适的 id、name、type 和 placeholder 等属性。编写的代码如下所示。

```
<p>
  <label for="u_phone"> 电话号码 </label>
  <input id="u_phone" name="u_phone" type="tel" placeholder="电话号码" required />
</p>
<p>
  <label for="u_email"> 电子邮件 </label>
  <input id="u_email" name="u_email" type="email" placeholder="电子邮件" required />
</p>
<p>
  <label for="u_site"> 网站 </label>
  <input id="u_site" name="u_site" type="url" placeholder="网站"/>
</p>
<p>
  <label for="u_pass"> 密码 </label>
  <input id="u_pass" name="u_pass" type="password" placeholder="密码" required />
</p>
```

接下来，我们添加“同意用户服务条款与个人信息收集条款”与提交按钮（submit button）。首先编写复选框与“同意用户服务条款与个人信息收集条款”标签代码。同样，我们使用一个 `<p>` 元素来表示整块区域，而后在 `<p>` 元素内分别添加 `<input>` 与 `<label>` 两个元素，在 `<label>` 中使用了两个 `<a>` 元素为文本添加链接，当用户单击时可以查看详细内容。`` 标签用于强调被其包含的文本。



```

<p id="accept">
    <input id="accept_terms" name="accept_terms" type="checkbox" />
    <label for="accept_terms">
        <strong><a href="#" rel="external" target="_blank"> 服务使用条款 </a>,
        <a href="#" rel="external" target="_blank"> 个人信息收集使用</a>我同意</strong>.
    </label>
</p>

```

接着，编写提交按钮代码。首先插入一个 `<p>` 元素，而后在其中添加一个 `<button>` 元素，并设置其 `type` 属性为 `submit`（默认值，可以省略），设置 `title` 属性，代码如下。

```

<p id="add_count">
    <button id="reg_new" type="submit" title="加入会员" + </button>
</p>

```

至此注册表格的 HTML 代码编写完成了，代码整理如下。



保存代码，在浏览器中打开网页，可以看到已经编写好的注册表格。

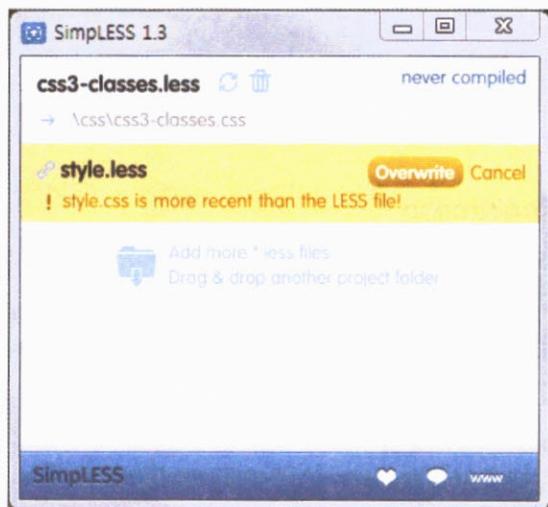
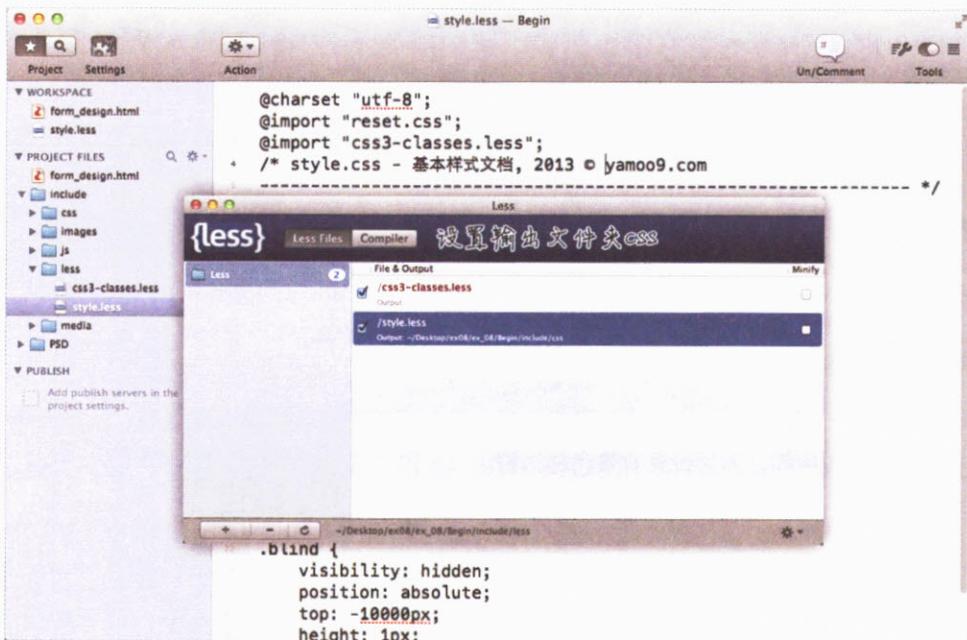


编写 CSS3 与 LESS 样式代码

为了使用 LESS 编写样式代码，如果你是 Windows 用户，要先安装 SimpLESS.exe 程序；如果你是 Mac 用户，则要安装 Less.app 程序。关于它们的安装与使用方法，在前面我们已经详细地讲解过了。如果忘记了，请重新翻到第 6 章第 179 页学习。

在 Windows 环境下，首先运行 SimpLESS 程序，而后将 begin 整个文件夹拖放到 SimpLESS 程序中。当在 LESS 文件中编写代码并保存后，SimpLESS 会自动帮我们转换成 CSS 代码，添加到指定的 CSS 文档中。在向 SimpLESS 中拖放项目文件夹时，若发生错误，请参考第 6 章第 183 页中的相关内容进行解决。

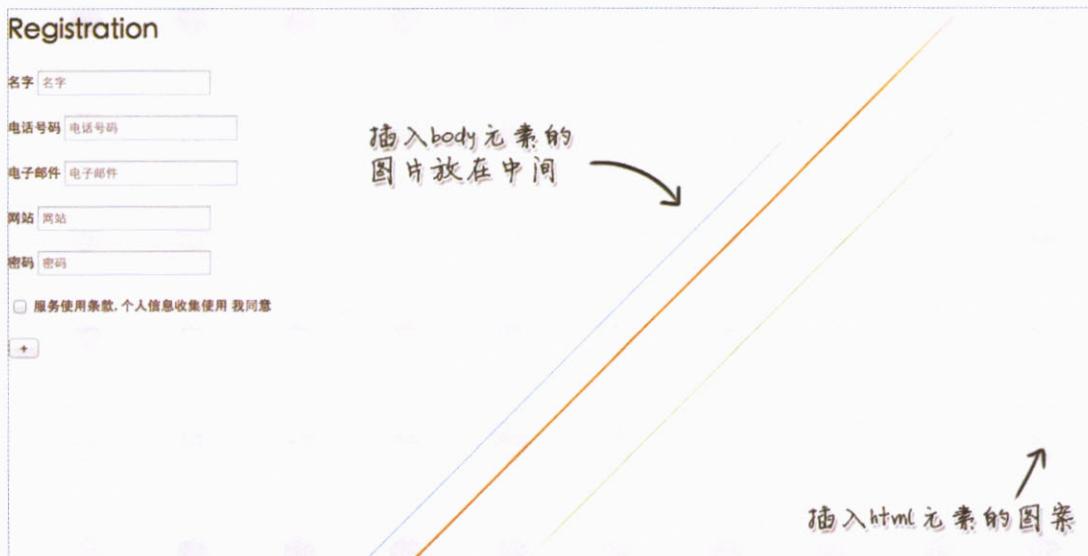
在 Mac 环境下，先运行 Less Compiler 程序，设置输出目录为 include\css，如图所示。



编写基本样式控制代码

下面我们开始在 style.less 文件中编写基本样式控制代码。首先设置 html 与 body 元素的高度为 100%。在将 body 元素的高度设为 100% 后，它会充满网页浏览器的整个窗口。而后为 html 元素设置 overflow-y 属性，并将背景图片置于画面中间。最后为 body 元素指定使用的网页字体，而后将背景图片插入到中间。

```
html, body { height: 100%; }
html { overflow-y: scroll;
      background: url(..../images/bg-pattern.png) center; }
body { font-family: 'NanumGothicWeb';
      background: url(..../images/bg.png) no-repeat center; }
```



小知识 分别向 html 与 body 插入背景图片

在 CSS3 中，我们可以向一个元素添加两张以上的图片，所以我们可以不必向 html 元素添加图片。但是考虑到有些浏览器并不支持 CSS3 Multi-Backgrounds 属性，所以在实例中我们分别向 html 与 body 插入了背景图片。

为整个表单区域（容器元素）编写样式代码

下面我们开始为整个表单区域 #reg 编写样式代码。首先设置好横向与纵向区域后，指定为绝对定位，将表单放到画面中间，而后设置背景色与边框。

#reg { position: absolute;	将#reg元素设置为绝对定位 (基准元素: 带有position属性的最近的父元素)
top: 50%;	从基准元素下移50%
left: 50%;	从基准元素右移50%
width: 500px;	设置宽度为500px
min-height: 390px;	设置最小高度为390px (内部内容变多时, 纵向拉长)
margin-top: -1 * 390px / 2;	若想设置到画面中间, 向上移动高度的一半 (LESS运算)
margin-left: -1 * 500px / 2;	若想设置到画面中间, 向左移动宽度的一半 (LESS运算)
border: 1px solid #dcdcdc;	设置边框粗细为1px, 实线, 亮灰色.
background: #f2f2f2; }	设置背景色为边框更亮的白色 (#f2f2f2) .

在设置 margin-top 与 margin-left 值时, 我们使用了运算表达式。CSS 不是脚本语言, 所以在 CSS 中不能使用运算表达式, 但是我们可以通过 LESS 来使用运算表达式。在编写时我们完全可以使具体的数值来设置相应的属性值, 但是考虑到后期维护的便利性, 我们使用 LESS 变量来替换具体的数值, 方便日后修改维护。首先我们使用 LESS 变量来设置 width、min-height 的值, 代码如下。

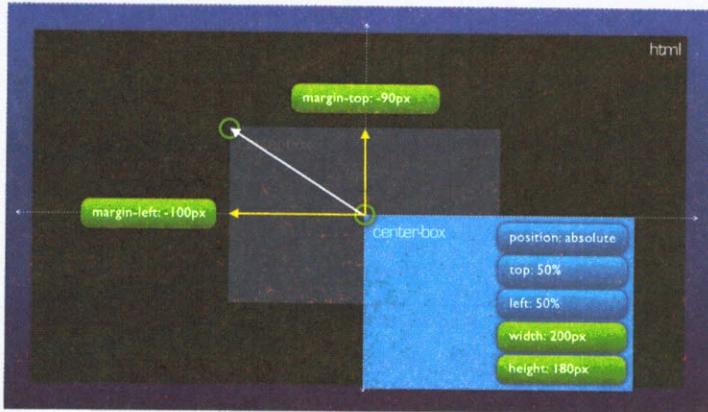
```
@reg_w: 500px; // #reg宽度      声明LESS变量, 并设置数值
@reg_h: 390px; // #reg宽度
#reg {
  width: @reg_w;                使用@reg_w变量为#reg的宽度设置数值.
  min-height: @reg_h;           使用@reg_h变量设置#reg的高度,
  margin-top: -1 * @reg_h / 2;  设置上外边距为负的@reg_h的一半.
  margin-left: -1 * @reg_w / 2;  设置左外边距为负的@reg_w的一半.
```

保存代码, 在浏览器中查看网页, 可以看到注册表格到画面的中间去了, 如图所示。



小知识 在 CSS 中将盒子移动到网页浏览器窗口的中间（使用 position 与 margin 属性）

通过上面的代码，我们将注册表格移动到了窗口中间，方法非常简单。首先我们将元素盒子的基准点（左上顶点）移动到窗口的中心点上（position: absolute, top: 50%, left: 50%），然后设置 margin-top 与 margin-left 值，将它们分别设置为元素盒子高度与宽度的一半，并且要使用负值，如图所示。在向 margin 设置负值时，元素盒子可能会溢出到窗口之外，或者与其他元素盒子重叠在一起，设置时千万要注意。



接下来，我们开始为 #reg 设置 border-radius 与 box-shadow 属性。由于设置时需要针对不同浏览器分别添加前缀，为了简便起见，在这里我们使用 css3-classes.less 文件中的 LESS 函数分别设置它们（border-radius 与 box-shadow 函数在第 6 章中我们已经做过讲解，关于函数的编写方法，请参考第 6 章第 163 页中的相关内容）。编写 .reg-gradient() 函数，向 #reg 的背景添加线性渐变效果。

```
#reg {
    .box-shadow(inset 0px 0px 3px #cdcdcd);
    .border-radius(9px);
    .reg-gradient(90deg, #f2f2f2, #fff 2px, #f4f4f4 5px, hsla(0, 0%, 100%, 0.8) 67%);
}
```

由于 .reg-gradient() 函数未在 css3-classes.less 中定义过，因此需要我们先编写好该函数。.reg-gradient() 函数有 5 个参数，第一个参数是渐变角度，另外 4 个参数是颜色参数。在前面我们学过 CSS3 渐变技术以及编写 LESS 函数的方法，在编写 .reg-gradient() 函数时，我们将使用这些技术与方法，函数代码如下所示。

```
.reg-gradient( @deg, @param1, @param2, @param3, @param4 ) {
    background: -webkit-linear-gradient( @deg, @param1, @param2, @param3, @param4 );
    background: -moz-linear-gradient( @deg, @param1, @param2, @param3, @param4 );
    background: -ms-linear-gradient( @deg, @param1, @param2, @param3, @param4 );
    background: -o-linear-gradient( @deg, @param1, @param2, @param3, @param4 );
    background: linear-gradient( @deg, @param1, @param2, @param3, @param4 );
}
```

到这里，我们已经为 #reg 编写好了样式控制代码，如下图所示，仔细检查代码，确保编写的代码准确无误。

The screenshot shows a LESS code editor interface. On the left, there's a file tree with files like 'form.design.html', 'style.less', 'index.html', 'css', 'less', 'images', 'js', and 'node_modules'. The main area contains LESS code:

```
/* ///////////////////////////////////////////////////////////////////
 * #reg
 //////////////////////////////////////////////////////////////////*/
@reg.w: 500px; // #reg宽度
@reg.h: 390px; // #reg高度

// #reg 菜单函数
.reg-gradient(@deg, @param1, @param2, @param3, @param4) {
    background: -webkit-linear-gradient(@deg, @param1, @param2, @param3, @param4);
    background: -moz-linear-gradient(@deg, @param1, @param2, @param3, @param4);
    background: -o-linear-gradient(@deg, @param1, @param2, @param3, @param4);
    background: -ms-linear-gradient(@deg, @param1, @param2, @param3, @param4);
    background: linear-gradient(@deg, @param1, @param2, @param3, @param4);
}

.reg {
    position: absolute;
    top: 50px;
    left: 50px;
    width: @reg.w;
    height: @reg.h;
    margin-top: -1 * @reg.h / 2;
    margin-left: -1 * @reg.w / 2;
    border: 1px solid #cdcdc;
    background: #f2f2f2;
    .reg-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    .box-shadow(inset 0px 0px 3px #cdcdc);
    .border-radius(9px);
}
```

A callout points to the first LESS variable '@reg.w' with the text 'LESS变量与函数' (LESS variable and function). Another callout points to the 'reg-gradient' function with the text '调用上面设置的LESS变量与函数' (Call the LESS variable and function set above).

保存代码，在浏览器中查看页面，观察样式控制的效果。



运行 Firebug 插件，在 Firebug 的 CSS 面板中，单击 Source Edit 按钮，可以看到 LESS 文件编译后生成的 CSS 代码。

The screenshot shows the Firebug CSS panel. A dropdown menu is open over a section of the code, with 'Source' selected. The code shown is the compiled CSS from the 'style.css' file:

```
reg {
    position: absolute;
    top: 50px;
    left: 50px;
    width: 500px;
    height: 390px;
    margin-top: -195px;
    margin-left: -250px;
    border: 1px solid #cdcdc;
    background: #f2f2f2;
    -webkit-linear-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    background: -moz-linear-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    background: -o-linear-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    background: -ms-linear-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    background: linear-gradient(90deg, #f2f2f2, #ffff 2px, #f4f4f4 50px, hsla(0, 0%, 100%, 0.8) 67%);
    box-shadow: inset 0px 0px 3px #cdcdc;
    border-radius: 9px;
}
```

A callout points to the first line of the compiled CSS with the text '查看编译得到的style.css文件' (View the compiled style.css file). Another callout points to the 'border-radius' property with the text '设置为计算后的结果' (Set to calculated result).

接下来，为 #reg 设置内部空间。当前 #reg 元素采用的是绝对定位，并且添加了实现居中对齐的代码，如果我们为其设置 padding，就会使其向右下方而非画面中间倾斜。这是因为在标准的 CSS 盒子模型中，width 与 height 指的是 content-box 区域的宽度与高度，与 padding、border 和 margin 值无关。所以这里我们不是为 #reg 元素设置 padding 值，而是通过为 #reg 内部元素设置 margin 值来解决问题。

```
#reg > * { margin: 15px 20px; }
```

向#reg元素内的直属子元素设置外边距（上下外边距15px，左右外边距20px）

设置标题样式

在 #reg 区域中有一个 `<h2>` 标题元素，下面我们为其设置样式。首先为下边框定义样式为粗细 1pt、灰色、实线。而后使用谷歌云字体服务，为其指定字体样式为 Sansita One。然后再为标题文本添加阴影，调用 LESS 函数 `.box-shadow()` 为标题区域添加阴影。

```
#reg h2 {  
    border-bottom: 1px solid #ddd;  
    font-family: 'Sansita One', cursive;  
    text-shadow: 1px 1px #fff;  
    .box-shadow(0px 1px #fff);  
}
```

选择#reg的h2元素

设置下边框的粗细、类型、颜色

设置字体系列为谷歌web字体

为文本设置白色阴影

调用.box-shadow()函数，设置阴影

保存代码，在浏览器中查看页面，可以看到标题已经被应用上了指定的字体，并且已经设置好了下边框，而且添加了阴影。至此，注册表格的外观以及标题样式编写完成了。



控制表单元素样式 1：标签与输入文本框

下面我们开始为表单内的元素编写样式代码。`label` 元素默认为行内元素，我们首先将其转换为块元素，代码如下。

```
#reg label { display: block; }          将#reg元素内部的label转换为块元素
```

接下来，我们为 `#reg` 中的 `input` 元素添加样式代码，除了 `type` 为 `checkbox` 的 `input` 元素外。在这里我们使用 CSS3 的否定伪类 (`:not`)，将 “`type=checkbox`” 的 `input` 元素排除在外。

```
#reg input:not([type=checkbox]) {          在#reg内部的input中，选择type属性不是checkbox的元素  
    width: 15em;                          设置宽度为15em（以字体大小1em为准）  
    border: 1px solid #cdcdcd;            设置边框粗细为1px，实线，亮灰色（#cdcdcd）  
    padding: 1em 0.5em 1em 2.5em;        设置内边距（上下内边距为1em，右内边距为0.5em，左内边距为2.5em）  
    background: #f8f8f8 url(..../images/u_icons_splite.png) no-repeat 0 4px;    设置背景图片  
    .box-shadow(inset 0px 0px 4px #cdcdcd);          调用.box-shadow()函数，添加阴影  
    .border-radius(5px);                  调用.border-radius()函数，设置圆角边框  
    .transition(all .1s);              调用.transition()函数，设置转换效果  
}  
#reg input:not([type=checkbox]):focus {          选择#reg内input元素（非checkbox）的获焦状态  
    width: 17em;                        设置宽度为17em（切换效果：宽度渐增）  
    background-color: #fff;             设置背景色为白色（#fff）（切换效果：改变背景色）  
    border-color: #7ccbe8;              设置边框颜色为亮蓝色（#7ccbe8）（切换效果：改变边框颜色）  
}
```

请参考以下图片，能够帮助我们理解各个设置值的含义。注意在图中我们已经根据 `input` 背景图标尺寸设置了左侧 padding 空间。



控制表单元素样式 2：用户使用条款与信息收集

在浏览器中显示的网页中，我们可以发现“用户使用条款，个人信息收集使用”部分有些问题，看上去非常生硬、不够自然。复选框与标签应该在一行中显示，现在却显示在两行中，这是因为前面我们已经将所有的 label 元素转换成了块元素（display:block）。为了使复选框与标签在同一行中，我们需要将 #accept 的 label 重新转换为行内元素（display:inline）。

因为所有label元素被转换为block(块元素)，所以出现了换行的问题

服务使用条款. 个人信息收集使用 我同意

同时，我们还要指定文本的大小，将复选框与前面的 input 元素对齐，设置超链接的文字颜色为淡蓝色，CSS 样式代码如下所示。

```
#accept { font-size: 11px; }
#accept label { display: inline; }
#accept_terms { margin-left: -1px; }
#accept a { color: #31a8d7; }
```

设置#accept的字号为11px
将#accept内的label元素转换为行内元素
为#accept_terms元素设置左外边距为-1px
为#accept内的a元素设置字体颜色为淡蓝色（#31a8d7）

密码

密码

服务使用条款. 个人信息收集使用 我同意

在将margin-left设置为-1px后于input元素对齐

控制表单元素样式 3：向输入文本框左侧添加图标

下面为输入文本框添加背景图标，并通过 background-position 属性将各个图标指定到输入文本框的左侧。请看右图，在 images 文件夹中有一张同时包含 10 个图标的图片，在前面我们已经将它设置为 5 个 input 元素的背景了。在设计图标时，将要添加的图片的高度设置为 28px。若背景图标的初始位置为 4px，下一张图标的为 -24px (-4-28px)，再下一张为 -52px，再再下一张为 -80px，以此类推。也就是说，每个图标之间的间隔为 28px，这样每个 input 元素的背景才会显示为不同的图标。



在理解了背景图标位置的设置方法后，才能通过各个 input 元素的 id 选择器设置背景的位置，代码如下所示。

```
#reg #u_name {background-position: 0 4px;}  
#reg #u_name:focus {background-position: 0 -24px;}  
#reg #u_phone {background-position: 0 -52px;}  
#reg #u_phone:focus {background-position: 0 -80px;}  
#reg #u_email {background-position: 0 -108px;}  
#reg #u_email:focus {background-position: 0 -136px;}  
#reg #u_site {background-position: 0 -164px;}  
#reg #u_site:focus {background-position: 0 -192px;}  
#reg #u_pass {background-position: 0 -220px;}  
#reg #u_pass:focus {background-position: 0 -248px;}
```

打开网页浏览器，当用鼠标单击 input 元素或按键盘上的 Tab 键使其获得焦点时，动画效果就呈现出来了。

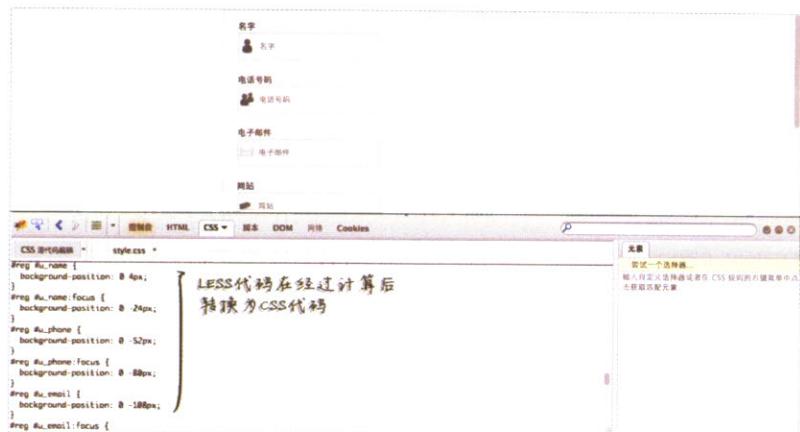


到这儿，我们已经为背景图标设置好位置了。但是考虑到后期维护的方便，我们还需要略微修改一下代码。在修改数值时需要一一修改，操作起来非常麻烦。所以这里我们使用 LESS 变量来取代具体的数值，首先新建两个 LESS 变量，第一个为背景的初始位置，第二个为图片高度。

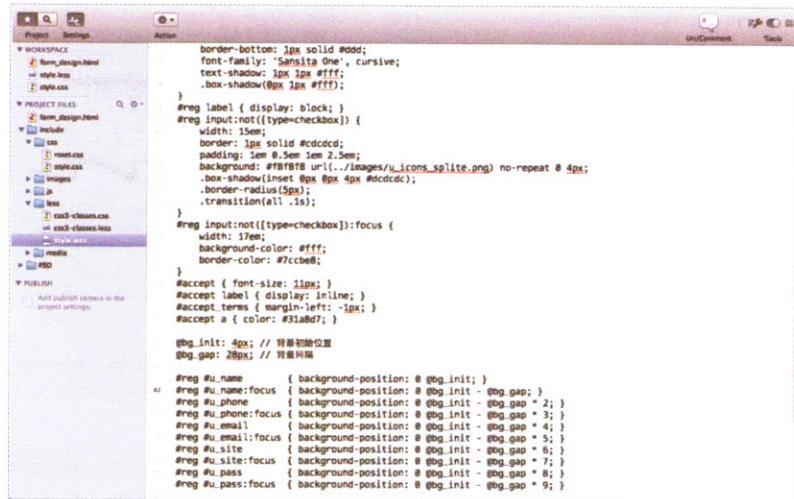
创建好两个 LESS 变量后，修改代码，使用 LESS 变量运算表达式来替换掉原来的具体数值，代码如下所示。

```
@bg_init: 4px; //背景初始位置  
@bg_gap: 28px; //背景间隔  
  
#reg #u_name {background-position: 0 @bg_init;}  
#reg #u_name:focus {background-position: 0 @bg_init - @bg_gap;}  
#reg #u_phone {background-position: 0 @bg_init - @bg_gap * 2;}  
#reg #u_phone:focus {background-position: 0 @bg_init - @bg_gap * 3;}  
#reg #u_email {background-position: 0 @bg_init - @bg_gap * 4;}  
#reg #u_email:focus {background-position: 0 @bg_init - @bg_gap * 5;}  
#reg #u_site {background-position: 0 @bg_init - @bg_gap * 6;}  
#reg #u_site:focus {background-position: 0 @bg_init - @bg_gap * 7;}  
#reg #u_pass {background-position: 0 @bg_init - @bg_gap * 8;}  
#reg #u_pass:focus {background-position: 0 @bg_init - @bg_gap * 9;}
```

代码经过修改之后，后期维护起来非常方便。在更改图片之后，只需要修改图片的初始位置与图片的高度值就可以了。

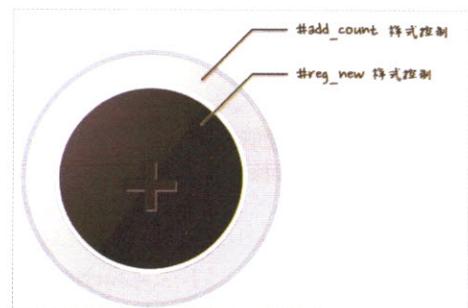


到目前为止，编写的所有代码，整理如下。要认真检查 LESS 变量设置与使用部分，避免出现错误。



控制表单元素样式 4：提交按钮

表单的最后一个元素是提交按钮，当用户填完相关信息，单击提交按钮后，表单内容就会被提交到服务器。提交按钮部分由段落元素 (#add_count) 与按钮本身 (#reg_new) 两部分组成，如右图所示。



首先为 #add_count 编写样式代码。在为 #add_count 编写样式代码时，我们使用了.reg-gradient() 这个 LESS 函数。.reg-gradient() 函数有 5 个参数，第一个为渐变角度，其余 4 个为颜色。当我们为 #add_count 添加渐变时并不需要 4 个颜色，但是在使用 .reg-gradient() 函数时又不能减少参数，否则在转换成 CSS 时会发生错误。所以，需要我们强制指定 4 个颜色。在 #add_count 调用 .reg-gradient() 函数时传入的参数中，最后两个是为了补齐参数的个数。

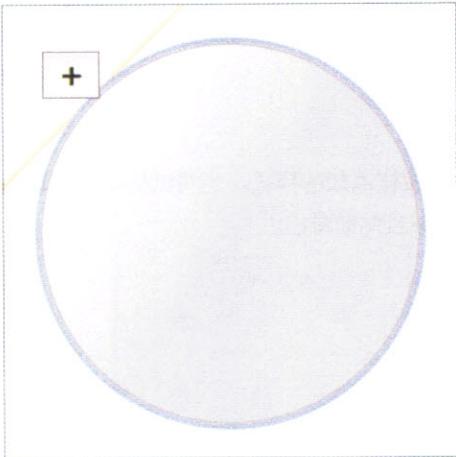
```

#add_count {
    position: absolute;
    top: 130px;
    right: 70px;
    width: 140px;
    height: 140px;
    border: 1px solid #cfcfcf;
    background: #e9e9e9;
    .reg-gradient(-35deg, #fff 10%, #e0e0e0 51%, #e0e0e0 51%, #e0e0e0 100%);
    .box-shadow(0px 0px 0px 2px #c3d4db);
    .border-radius(100px);
}

```

选择#add_count
设置为绝对定位 (基准元素: #reg)
从基准元素下移130px
从基准元素左移70px
设置宽度为140px
设置高度为140px
设置边框为1px, 实线, 亮灰色
设置背景色为亮灰色 (#e9e9e9)
调用.reg-gradient()函数
调用.box-shadow()函数
调用.border-radius()函数

在为 #add_count 添加完样式控制代码后, 查看样式控制效果, 如下图所示。



接下来, 为提交按钮本身 (#reg_new) 编写样式控制代码。事实上, 大部分样式控制代码前面我们已经学习过, 所以编写起来不会有太大的难度。但是其中我们要用到 .transform() 这个 LESS 函数, 这是我们第一次见到它, 要仔细地学习一下。CSS3 提供了 transform 属性, 该属性用于设置变形, 支持一个或多个变形函数, 可以实现对文字或图像的旋转、缩放、倾斜和移动 4 种变换, 这 4 种变形分别使用 rotate()、scale()、skew() 和 translate() 这 4 个函数来实现。transform 属性是 CSS3 新增的变形属性, 使用时需要针对不同浏览器添加不同的前缀。我们在 css3-classes.less 文件中编写了 .transform() 函数, 在它内部使用了 CSS3 的 transform 属性, 并且针对不同的浏览器添加了不同的前缀。该函数有一个参数, 用于为 transform 属性指定要使用的变形函数。在本实例中, 我们将使用 scale() 函数对提交按钮进行缩放, 传入的参数 1 代表 100%, 0.9 代表 90%。

```

#reg_new {
    cursor: pointer;
    position: absolute;
}

```

选择#reg_new
设置鼠标指针形状为手型
设置绝对定位 (基准元素: #add_count)

```

top: 10px;                                从基准元素下移10px
left: 10px;                               从基准元素右移10px
width: 120px;                             设置宽度为120px
height: 120px;                            设置高度为120px
border: 3px solid #fff;                   设置边框为3px，实线，白色
background: #1c1c1c;                      设置背景色为暗灰 (#1c1c1c)
color: #222;                             设置字体颜色为暗灰 (#222)
font: bold 48px Verdana;                 设置字体类型、大小
text-align: center;                       设置文本居中对齐
text-shadow: 1px 1px #999, -1px -1px #000;  设置文本阴影
.reg-gradient(-35deg, #7c7c7c 5%, #454545 30%, #1f1f1f 50%, #000 51%);  调用.reg-gradient()函数

.border-radius(100px);                    调用.border-radius()函数
.transition(all .4s);                  调用.transition()函数
.transform(scale(0.9, 0.9));            调用.transform()函数
}

#reg_new:hover, #reg_new:focus {
    .box-shadow(0px 0px 10px #999);   选择#reg_new:hover,#reg_new:focus状态
    .transform(scale(1, 1));           调用.transition()函数
}

```

保存代码，在浏览器中查看页面，可以看到我们为提交按钮添加的样式。当将鼠标指针移动到提交按钮上，或按 Tab 键使其获得焦点时，提交按钮会变大并且向前跳出。



到目前为止，我们已经为不支持 JavaScript 脚本的浏览器编写好了样式代码。若浏览器不支持 HTML5 的 placeholder 属性，input 元素中的提示文字就不会显示出来。在这种浏览器中，我们需要在 input 元素之上添加一个 label 元素，用于显示提示信息。当然在支持 placeholder 属性的浏览器中，就不再需要这些提示标签了，要使用 JavaScript 脚本将它们隐藏起来。

使用 jQuery 编写脚本代码

在本章中，我们将不再编写单独的 jQuery 插件，将只针对实例本身编写 jQuery 脚本代码。首先，打开“script.js”文件，使用 jQuery ready() 语句，编写运行初始化函数的代码，如下所示。

```
$(document).ready(init);  
function init() {  
    hideLabel();  
    addRequiredStar();  
}  
//...  
//注释：文档准备好之后运行init函数  
//注释：声明init初始化函数  
//注释：调用hideLabel()函数，隐藏label元素  
//注释：调用addRequiredStar()函数，在带有required属性的input元素后添加星号(*)
```

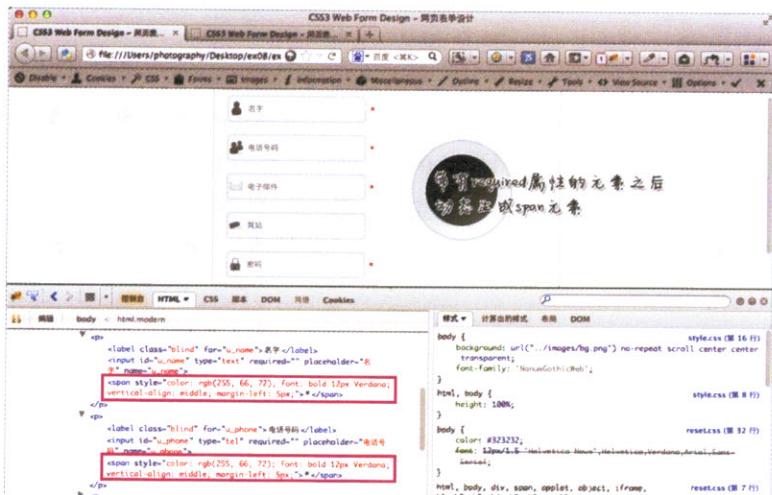
init() 函数是初始化函数，它在内部调用了 hideLabel() 与 addRequiredStar() 两个函数，其中 addRequiredStar() 用于向必填字段添加红色星号，hideLabel() 函数用于当浏览器支持 JavaScript 脚本时隐藏表单中的 label 元素。但是 for 属性值为 accept_terms 的 label 标签不应该隐藏起来。hideLabel() 函数的功能较简单，编写很容易，只需向 #reg 中的 label 标签添加 blind 类（在 style.css 中已经声明）就可以了，代码如下所示。

```
function hideLabel() {  
    //声明hideLabel()函数  
    //向label元素(for属性为accept_terms的元素除外)添加blind类  
}  
//...  
//注释：向label元素(for属性为accept_terms的元素除外)添加blind类
```



在 init() 初始函数中，调用的第二个函数为 addRequiredStar() 函数，它会选出表单中所有带有 required 属性的 input 元素，而后在 input 元素之后添加 span 元素，span 元素用于显示红色星号，告知用户该字段为必填字段。addRequiredStar() 函数代码如下所示。

```
function addRequiredStar () {    该函数用于向带有required属性的input元素之后添加span
    var target = $('[required]', '#reg');
        在#reg内查找带有required属性的元素，而后赋给target变量
    $('', {          动态创建span元素
        text: '*',          设置文本内容为“*”
        css: { 'color': '#ff4248', 'font': 'bold 12px Verdana',      设置字体，字体颜色，大小
            'vertical-align': 'middle', 'margin-left': '5px' }          设置垂直对齐方式及左外边距
    }).insertAfter( target );
};
```



接下来，我们在 init() 初始化函数中再添加调用函数 altPlaceholder() 的语句，在不支持 HTML5 placeholder 的浏览器中，使用该函数来添加替代功能。在 init() 初始化函数中调用 altPlaceholder() 函数，代码如下。

```
function init() {
    hideLabel();
    addRequiredStar();
    altPlaceholder('input:not(:checkbox)');
};
```

altPlaceholder() 函数带有一个形式参数 target，该参数是一个元素选择器字符串，这些元素是需要应用 HTML5 placeholder 属性的元素。在实例中，我们要传入的字符串为 'input:not(:checkbox)'，该字符串是 input 选择器字符串，但不包含 type 为 checkbox 的 input 元素。

在 altPlaceholder() 函数中，首先通过 Modernizr 脚本库判断浏览器是否支持 HTML5 的 placeholder 属性，若支持，则直接返回（return），因为这个函数是为不支持 placeholder 属性的浏览器编写的。若浏览器不支持 placeholder 属性，则将传入的 target 参数包装为 jQuery 对象，把各元素的 placeholder 属性值保存到 input_holder 变量中，然后使用各元素的 value 值设置 input_holder 变量值。

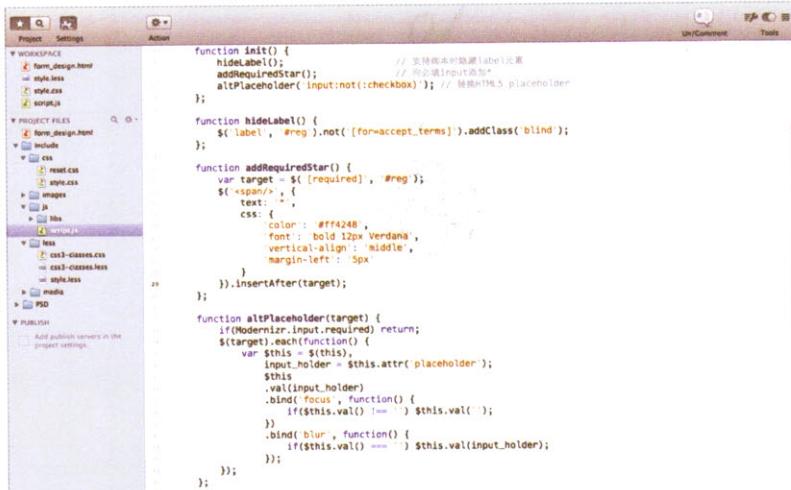
在函数的最后部分是处理 focus 与 blur 事件的代码。在 focus 状态下，检查各个元素的 value 值，若有内容，则清空；在 blur 状态下，检查 value 值，若为空，则用 input_holder 值进行填充。

```
function altPlaceholder(target) {          // 声明altPlaceholder函数(包含target参数)
    if( Modernizr.input.placeholder ) return; // 若浏览器支持HTML5 placeholder属性，终止函数

    $(target).each(function() {           // 使用jQuery()包装传入的参数后，each()语句执行
        var $this = $(this),             // 将$(target)保存到$this变量中
            input_holder = $this.attr('placeholder'); // 将$this的placeholder属性值赋给input_holder变量

        $this
            .val(input_holder)          // 将input_holder的值设置为$this元素的value值
            .bind('focus', function() { // 向$this变量引用的对象绑定focus事件处理函数
                if($this.val() !== '') $this.val(''); // 若$this的value值非空，则清空它
            })
            .bind('blur', function() { // 为$this引用的对象绑定blur事件的处理函数
                if($this.val() === '') $this.val(input_holder); // 若$this的value值为空字符串，则使用input_holder设置value值
            });
    });
}
```

到这儿，针对不支持 HTML5 placeholder 属性的浏览器，替代代码编写完成，全部代码整理如下。



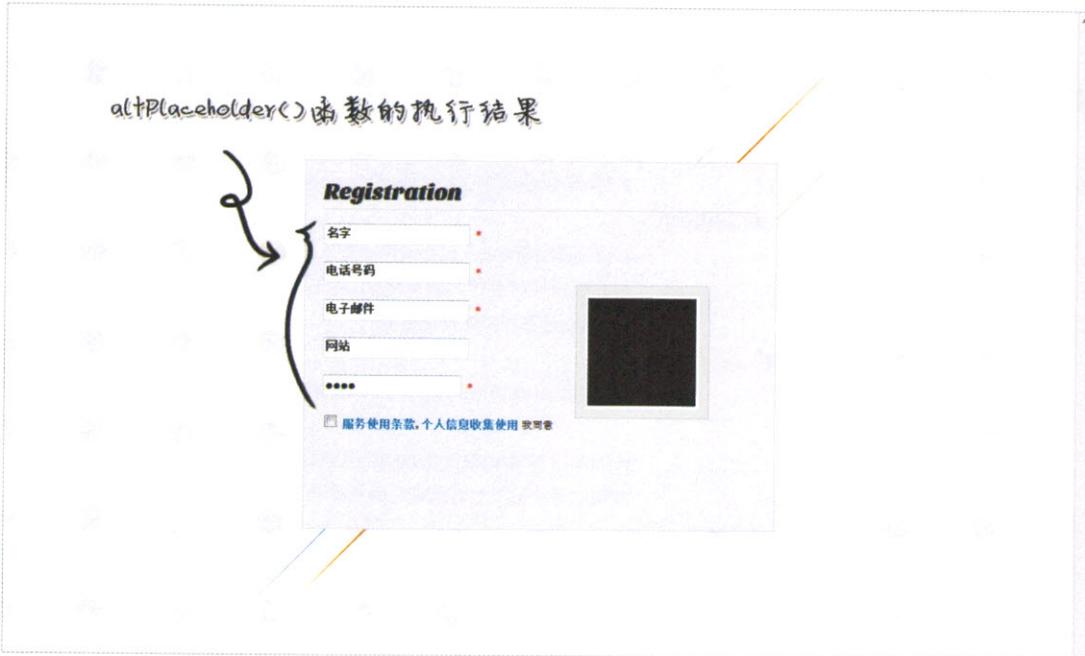
```
function init() {
    hideLabel(); // 支持版本时隐藏label元素
    addRequiredStar(); // 为必填input添加*
    altPlaceholder('input:not(:checkbox)'); // 替换HTML5 placeholder
};

function hideLabel() {
    $('#label', '#reg').not('[for=accept_terms)').addClass('blind');
};

function addRequiredStar() {
    var target = $('#required'), #req;
    $('#label').each(function() {
        var $text = $(this);
        $text.css({
            color: '#FF4248',
            font: 'bold 12px Verdana',
            vertical-align: 'middle',
            margin-left: '5px'
        });
        $text.insertAfter(target);
    });
};

function altPlaceholder(target) {
    if(Modernizr.input.required) return;
    $(target).each(function() {
        var $this = $(this),
            input_holder = $this.attr('placeholder');
        $this
            .val(input_holder)
            .bind('focus', function() {
                if($this.val() === '') $this.val('');
            })
            .bind('blur', function() {
                if($this.val() === '') $this.val(input_holder);
            });
    });
};
```

IE 8 是旧版本浏览器，它不支持 HTML5 的 placeholder 属性。下面我们运行 IE 8 浏览器查看网页显示效果，如下图所示。观察页面显示效果，可以得知 altPlaceholder() 函数正常执行，在 input 输入字段内使用 value 值取代 placeholder 属性，向用户显示提示信息。在不支持脚本的环境中，altPlaceholder() 函数不会被执行，但 label 标签会显示出来，提示用户输入相应信息。



最后的问题是：IE 6~8 旧浏览器不支持 CSS3 的新功能，我们该如何实现类似的功能呢？请看下图。由于已经修改了相关代码，因此网页在旧浏览器中的显示效果与新浏览器中的显示效果类似。



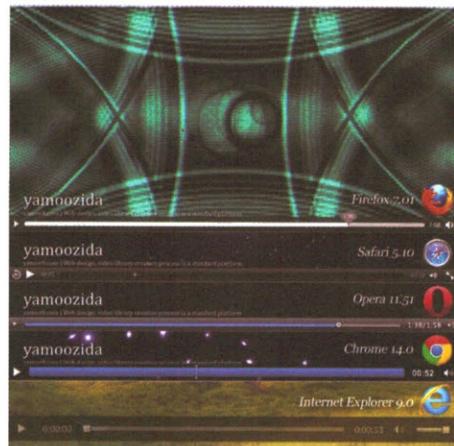
9 章

使用 HTML5 制作视频播放器

在本章中，我们将使用 HTML5 制作一个网页视频播放器。在 HTML5 之前，网页浏览器本身不能处理视频，要在网页中播放视频，就要使用 Flash Movie 或 Windows Media Player 等外部插件。但是从 HTML5 开始新增了支持视频的元素，使得开发者可以在页面上播放视频、音频等多媒体。通过 HTML5 新增的视频支持元素，播放多媒体时无需在浏览器上安装外部插件，只要浏览器本身支持 HTML5 即可。

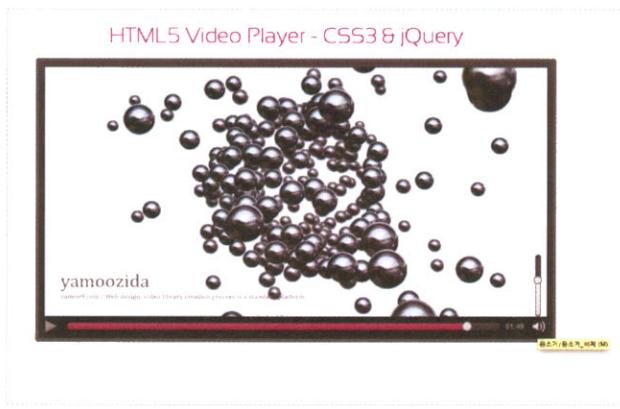
设计效果预览

最新的网页浏览器都提供 HTML5 视频控件 UI 的支持，但它们提供的默认 UI 界面是不同的，因为 UI 设计没有统一的设计规范可以遵循。对于这种情况，我们不是第一次遇到。在第 8 章设计注册表时，你会发现每种浏览器对注册表格的呈现效果略有不同。也就是说，相同的视频播放器代码，可能会因用户所用的浏览器不同而呈现不同的显示效果，这是正常的，不是设计的问题。



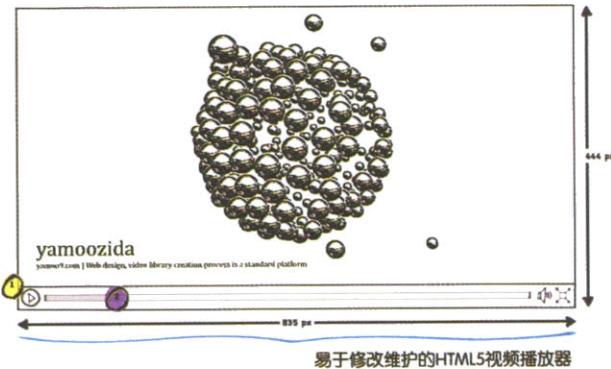
在不同的浏览器中 HTML5 视频控件的 UI 各不相同

不同的客户可能对播放器的 UI 有不同的要求。有些客户要求设计者不要使用浏览器默认的播放控件的 UI，要求开发者针对不同的浏览器开发不同的 UI。而有些客户可能要求开发者为视频控件开发统一的 UI，使得它在任何浏览器下都能保持统一的显示风格。设计者在制作网页播放器时要根据客户的具体要求设计 UI。在本章中，我们将制作 UI 显示风格统一的视频播放器，即制作的视频播放器不论在哪种浏览器中都有相同的显示外观。



设计网页布局

首先使用 Balsamiq Mockups 软件将头脑中关于播放器 UI 设计的构思描绘出来，如右图所示。

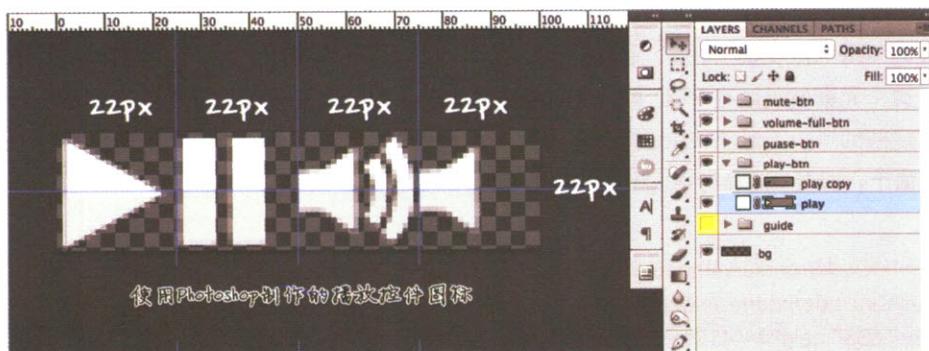


易于修改维护的HTML5视频播放器

①添加 inner-ui 类之后，播放控件被隐藏起来。

②用于操作 seek-bar 颜色的类控件。

制作时我们使用与第8章中注册表格设计相同的方法，灵活使用 Sprite图像技法。我们要使用的图像是按钮UI，很难使用CSS3来实现。在这里，我们将播放、暂停、消声与放声按钮图标做在一张图片中使用。每个图标尺寸为22×22px，4个按钮图标横向排列在一起，所以整个Sprite图片的宽度为88px。除了4个按钮UI外，其他控件UI都使用CSS3来实现。



编写 HTML5 视频播放器

下载例题源文件，解压缩后，进入 \ex_09\Begin 文件夹，在网页编辑器中打开 video-player-design.html 文件，其中包含编写 HTML5 视频播放器的参考代码，如下图所示。

The screenshot shows a web-based code editor interface. On the left, there's a project navigation pane with sections for 'WORKSPACE', 'PROJECT FILES', and 'PUBLISH'. Under 'PROJECT FILES', there's a file named 'video-player-design.html' which is currently selected. The main area displays the HTML code for the video player. Several annotations are present:

- A red arrow points to the line containing the jQuery UI script inclusion: <script src="include/js/libs/jquery.UI.min.js"></script>. The annotation text is "调用jQuery UI库".
- A red arrow points to the lines including CSS and JS files for the video player: <link href="include/css/videoPlayer.css" rel="stylesheet" /> and <script src="include/js/plugin/videoPlayer/jquery.videoPlayer.js"></script>. The annotation text is "调用视频播放器的CSS与JS文件".
- A red arrow points to a section of the code starting with '<!-- e: #page-wrap -->'. The annotation text is "以下是一段参考代码，帮助大家了解制作流程".
- A red arrow points to a figure element with a style of 'display: none;': <figure class="video-container" style="display: none;"><!-- 由JavaScript动态生成 -->. A callout box with the text "在<figure>中已经将display设为none，所以参考代码不会显示出来" has an arrow pointing to this line.

```
<!DOCTYPE html>
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->
<!--[if (gt; IE 9)!!(IE)]><!--><html lang="zh" class="no-js modern"><!--<![endif]-->
<head>
    <meta charset="utf-8" />
    <!--[if IE 9]><meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" /><![endif]-->
    <title>HTML5 Video Player Design - 视频播放器设计</title>
    <link href="include/css/style.css" rel="stylesheet" />
    <script src="include/js/libs/modernizr.min.js"></script>
    <script src="include/js/libs/jquery.min.js"></script>
    <script src="include/js/libs/jquery.UI.min.js"></script>
    <link href="include/css/videoPlayer.css" rel="stylesheet" />
    <script src="include/js/plugin/videoPlayer/jquery.videoPlayer.js"></script>
    <script src="include/js/script.js"></script>
</head>
<body>
    <div id="page-wrap">
        <video src="include/media/yamoo9-video.webm" poster="include/images/video-poster.jpg" controls width="835" height="417">
            <a href="include/media/yamoo9-video.webm">yamoo9-video</a>
        </video>
    </div> <!-- e: #page-wrap -->
    <!-- ///////////////////////////////////////////////////////////////////
    以下代码是参考代码，参考之后请将它们删除
    //////////////////////////////////////////////////////////////////-->
    <figure class="video-container" style="display: none;"><!-- 由JavaScript动态生成 -->
        <!-- 手工编写的HTML代码 -->
        <video id="yamoo9-video" poster="include/images/video-poster.jpg" controls width="835" height="417">
            <source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8, Vorbis'" />
        </video>
    </figure>

```

编写基本的 HTML 代码

首先向 `<div id="page-wrap">` 元素中添加 `<video>` 元素。`<video>` 元素是 HTML 5 新增加的元素，我们通过这个元素就可以在 HTML 页面上直接播放视频，而无需安装任何插件，只要浏览器本身支持 HTML5 规范即可。而后为 `<video>` 元素设置 `src` 属性（指定视频源）、`poster` 属性（视频加载时的默认画面），以及 `width` 与 `height` 属性（视频的宽度与高度）。

```
<video src="include/media/yamoo9-video.webm"
       poster="include/images/video-poster.jpg"
       width="835" height="417">
    <a href="include/media/yamoo9-video.webm"> ... </a>
</video>
```

设置视频源

设置视频加载时显示的图片

设置视频的宽度与高度

当不支持视频时，设置替换内容

```

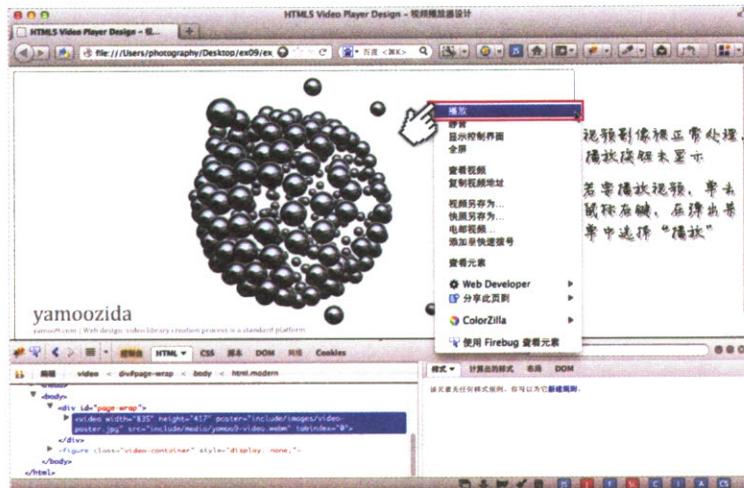
<title>HTML5 Video Player Design - 视频播放器设计</title>
<link href="include/css/style.css" rel="stylesheet" />
<script src="include/js/libs/modernizr.min.js"></script>
<script src="include/js/libs/jquery.min.js"></script>
<script src="include/js/libs/jquery.UI.min.js"></script>
<link href="include/js/plugin/videoPlayer/jquery.videoPlayer.css" rel="stylesheet" />
<script src="include/js/plugin/videoPlayer/jquery.videoPlayer.js"></script>
<script src="include/js/script.js"></script>
</head>
<body>
<div id="page-wrap">
    <video
        src="include/media/yamoo9-video.webm"
        poster="include/images/video-poster.jpg"
        width="835"
        height="417"
    >
        <a href="include/media/yamoo9-video">yamoo9-video</a>
    </video>
</div> <!-- e: #page-wrap -->

```

<video>与元素非常类似，比如它们都有src、width和height等属性。但与元素不同，<video>元素没有alt属性，在<video>元素中我们添加<a>元素来实现alt属性的功能，使得视频文件可以进行下载。当用户的浏览器不支持HTML5 Video时，我们可以向用户提供视频下载功能，使得用户可以下载观看视频。

保存代码，在浏览器中打开页面，可以看到浏览器正常显示出视频默认画面，但是单击默认画面时无法播放视频，这是因为我们还没有将视频控件显示出来。

在不显示视频控件的状态下，要想播放视频，可以单击鼠标右键，在弹出的菜单中，选择“播放”，即可播放视频。也可以选择“显示控制界面”（Firefox浏览器）菜单，将视频控制界面显示出来。



当视频控件不显示时很容易造成用户误解，认为它只是一张静止的图片，而不知道它实际是一段视频。当然大部分用户也不会想到单击鼠标右键并选择“播放”按钮观看视频。作为一名合格的网页设计人员，应该充分了解用户的使用习惯，使用户更容易、更方便地使用网页中的内容。

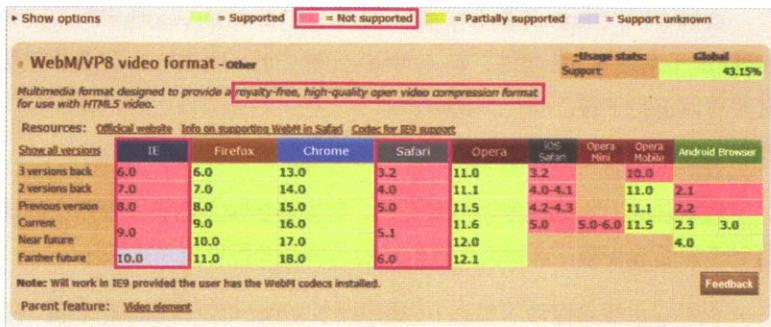
为了用户能够顺利地观看视频，下面我们要为视频添加控件，添加控件也很简单，只要向<video>元素添加controls属性就可以了，代码如下所示。

```
<video src="include/media/yamoo9-video.webm"
    poster="include/images/video-poster.jpg"
    controls
    width="835" height="417">
```

使视频控制按钮可见

还有一点需要我们注意，那就是并不是所有支持 HTML5 Video 的浏览器都支持 webm 视频格式。请看下表，它是 CanIUse.com (<http://caniuse.com/#search=webm>) 提供的各种浏览器对 webm 格式视频支持的情况。由表中可以看出，截止 2012 年 2 月，Safari 与 IE 浏览器不支持 webm 视频格式。

webm 是 Google 提出的一种开放、免费的媒体文件格式 (Royalty-free,high-quality open video compression format)。由于是免费的，相信 Safari 与 IE 在不久的将来会支持它。到目前为止，若想在 Safari 与 IE 9 以上的浏览器中播放 webm 视频文件，需要先在浏览器中安装插件与编解码器。在这些浏览器本身支持 webm 视频格式之前，需要先引导用户安装相应的插件，这是目前不可避免的问题。



这不矛盾吗？前面我们说过使用 HTML5 Video 的最大的好处就是浏览器可以直接在网页中播放视频而不需要安装视频播放插件，但是这里又让我们安装 webm 视频播放插件。是的，到目前为止，的确是这样的，这个矛盾会一直存在着，直到 Safari 与 IE 浏览器正式支持 webm 视频格式为止。

到 2012 年 2 月，Safari 与 IE 9 以上浏览器虽然不支持 webm 视频格式，但它们支持 mp4 格式的视频。与 webm 视频格式不同，mp4 是一种付费的视频格式，所以 Firefox 与 Opera 浏览器不支持它。请看下表 (<http://caniuse.com/#search=mp4>)，它显示了各浏览器对 mp4 格式视频的支持情况。

总结一下，IE 与 Safari 浏览器不支持 webm 格式视频，Firefox 与 Opera 浏览器不支持 mp4 格式视频。webm 是免费的，而 mp4 是需要付费的。如果你是浏览器的开发者，你会选择哪种视频格式呢？



让我们回到代码编写中。在前面的代码中，我们为 `<video>` 元素指定的视频源为 `webm` 格式，所以视频在 IE 与 Safari 中将无法正常播放。下面我们尝试使用 `<source>` 元素来解决这个问题。

使用 `<source>` 元素可以取代 `<video>` 元素的 `src` 属性。我们只能通过 `src` 属性为 `<video>` 元素设置一个视频源，而通过使用 `<source>` 元素可以为 `<video>` 元素指定多个视频源。它有两个属性：`src` 属性用于指定音频、视频文件的 URL；`type` 属性用于指定音频、视频文件的类型。在本实例中，我们可以使用 `<source>` 元素为 `<video>` 元素设置两个视频源，一个是 `webm` 格式，另一个为 `mp4` 格式，这样几乎在所有的浏览器中都能看到视频了。编写代码如下，删除 `src="include/media/yamoo9-video.webm"`，添加两个 `<source>` 元素。

```
<video src="include/media/yamoo9-video.webm"——  
    poster=" include/images/video-poster.jpg "  
    controls  
    width="835" height="417">  
    <source src="include/media/yamoo9-video.webm" />  
    <source src="include/media/yamoo9-video.mp4" />
```

删除

首先为video指定一个webm源
当不支持webm格式时，播放mp4

通过这种方式，即使不在浏览器中安装插件，也能看到相应的视频。但是需要为同一视频同时提供 `webm` 与 `mp4` 两种格式，这多少会有些不便。我们目前还无法避免这种不便，除非所有的浏览器都支持 `webm` 格式。

小知识 HTML5 视频的 MIME 类型

关于 HTML 5 视频的 MIME 类型，可以在 W3C HTML5 Working Draft 文档的 The Source Element (<http://www.clearboth.org/html5/video.html#the-source-element>) 中查看到。

W3C Working Draft

다음의 목록은 `codecs` 마일 매개변수를 `type` 속성에서 사용하는 몇 가지 예제를 보여줍니다. ↗
The following list shows some examples of how to use the `codecs` MIME parameter in the `type` attribute.

H.264 Constrained baseline profile video (main and extended video compatible) level 3 and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
```

H.264 Extended profile video (baseline-compatible) level 3 and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="avc1.5B401E, mp4a.40.2"'>
```

H.264 Main profile video level 3 and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="avc1.4D401E, mp4a.40.2"'>
```

H.264 'High' profile video (incompatible with main, baseline, or extended profiles) level 3 and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="avc1.64801E, mp4a.40.2"'>
```

MPEG-4 Visual Simple Profile Level 0 video and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="mp4v.20.8, mp4a.40.2"'>
```

MPEG-4 Advanced Simple Profile Level 0 video and Low-Complexity AAC audio in MP4 container

```
<source src='video.mp4' type='video/mp4; codecs="mp4v.20.240, mp4a.40.2"'>
```

MPEG-4 Visual Simple Profile Level 0 video and AMR audio in 3GPP container

```
<source src='video.3gp' type='video/3gpp; codecs="mp4v.20.8, aamr"'>
```

如代码所示，我们分别为两个`<source>`元素设置`type`属性为：`video/webm; codec='VP8, Vorbis'`; `video/mp4; codec='avc1.42E01E, mp4a.40.2'`。注意在每一个`type`属性设置中，前面与后面`codec`部分使用英文半角分号分隔。

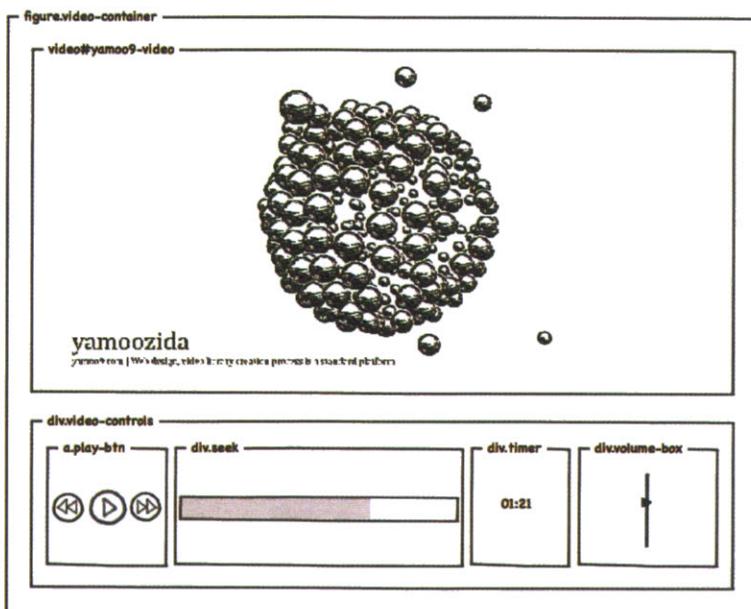
```
<source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8, Vorbis'" />
<source src="include/media/yamoo9-video.mp4" type="video/mp4; codec='avc1.42E01E, mp4a.40.2'" />
```

到这里，视频播放器的主要HTML代码已经编写完成了。在后面部分，我们将编写一个jQuery videoPlayer插件，然后通过插件为播放器生成其余的代码，如视频控件、视频容器等。我们使用手工编写的HTML代码与即将采用jQuery videoPlayer插件自动生成的代码，如右图所示。

```
<!-- 手工编写的HTML代码 -->
<!-- 以下代码是备用代码，参考之后请将它们删除 -->
<!-- 由JavaScript动态生成 -->
<figure class="video-container" style="display: none;"><!-- 由JavaScript动态生成 -->
<!-- 手工编写的HTML代码 -->
<video id="yamoo9-video" poster="include/images/video-poster.jpg" controls width="835" height="417">
  <source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8, Vorbis'" />
  <source src="include/media/yamoo9-video.mp4" type="video/mp4; codec='avc1.42E01E, mp4a.40.2'" />
  <a href="include/media/yamoo9-video.mp4" download=yamoo9-video.mp4>下载 : yamoo9-video.mp4</a>
</video>
<!-- 视频控件：由JavaScript动态生成 -->
<div class="video-controls">
  <a class="play-btn" title="播放/暂停" accesskey="P"></a>
  <div class="seek"></div>
  <div class="timer">00:00</div>
  <div class="volume-box">
    <div class="volume-slider"></div>
    <a class="volume-btn" title="静音/取消静音" accesskey="M"></a>
  </div>
</div><!-- e: .video-controls -->
</figure><!-- e: .video-container -->
```

编写 JavaScript 脚本自动生成的代码

下面我们开始编写JavaScript脚本自动生成的代码。如前所述，这里我们要编写的代码是由videoPlayer插件动态生成的，在后期维护时无需手工编写。我们之所以编写是为了帮助大家理解视频播放器的结构，之后我们会删除它们。整个视频播放器的结构如右图所示，除了`<video>`区域外，其他所有区域都是动态创建生成的。



首先插件脚本使用 `<figure>` 元素将 `<video>` 元素包裹起来，而后设置其 `class` 属性值为 `video-container`。然后在 `<video>` 元素下添加 `<div class="video-controls">` 元素，如以下代码所示。

```
<figure class="video-container">  
    <video> ... </video>  
    <div class="video-controls"> ... </div>  
</figure>
```

动态添加包含video的`<figure>`元素
添加`video-container`类

添加`<div>`元素，并设置`video-controls`类，用于取代浏览器默认的UI

`<figure>` 元素是 HTML5 新增的元素，它是一个容器元素，用于表示独立的流内容，如图像、图表、照片和视频等。而在 HTML5 之前常常使用 `<div>` 元素来表示这些独立内容。在实例中，我们使用 `<figure>` 元素将 `<video>` 与 `<div class="video-controls">` 放在了一起。

在为视频播放器设计播放控件时，我们没有使用 HTML5 的元素，而是使用常用的 `<div>` 元素来设计播放控件，设置其 `class` 为 `"video-controls"`，它内部包含着播放按钮、进度条、计时器和声音控件 4 个子部分，如图所示。到这里，我们一起分析视频播放器的代码，除了 `<video>` 元素是手工编写的，其余代码都是通过视频播放插件自动生成的。



```
<a class="play-btn" title="播放/暂停 (p)" accesskey="P"></a>  
<div class="seek"></div>  
<div class="timer"> 00:00 </div>  
<div class="volume-box">  
    <div class="volume-slider"></div>  
    <a class="volume-btn" title="静音/取消静音 (m)" accesskey="M"></a>  
</div>
```

添加[元素，充当\[播放\]按钮](#)

添加

元素，充当播放进度条

添加

元素，充当计时器

添加

元素，充当音量调节区

添加

元素，充当音量滑块

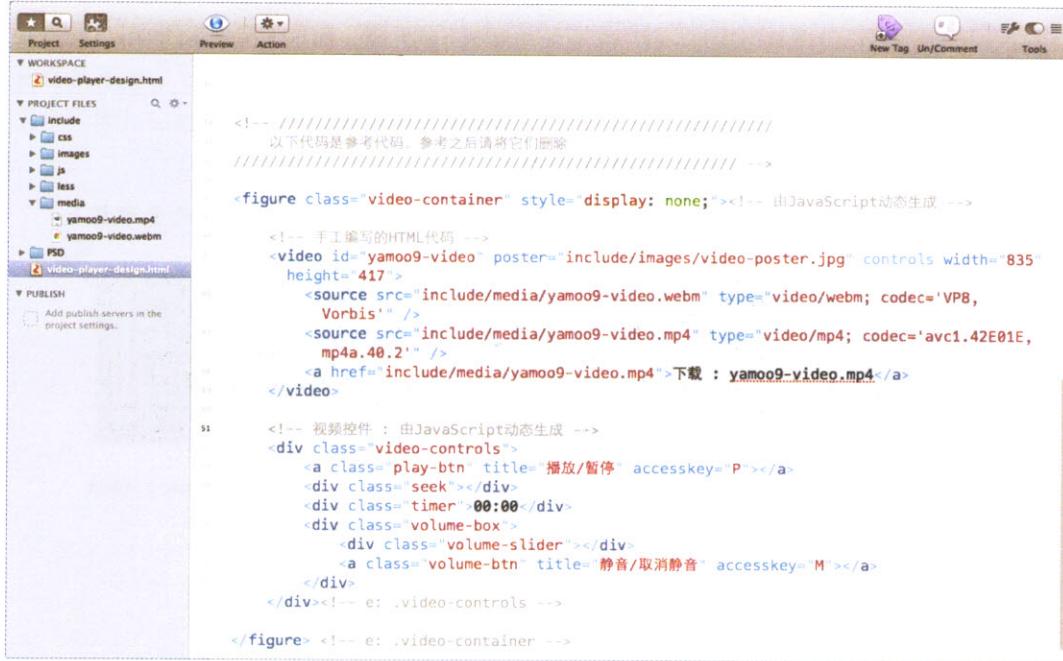
添加[元素，充当静音按钮](#)

小知识 滑动条

关于播放进度条与控制声音的滑动条，我们将使用 jQuery UI 的 Slider 插件进行制作。在本章第 254 页中，将详细讲解 Slider 插件的使用方法。

编写 jQuery 视频播放插件

在编写样式控制代码之前，我们先编写 jQuery 视频播放插件。这样在创建好 jQuery UI Slider 后才能更轻松容易地编写样式控制代码。首先打开 video-player-design.html 文档，删除所有要动态生成的代码，在文档中放入这些动态生成的代码只是为了帮助大家更好地理解网页播放器的设计结构。在网页文档中，找到这些参考代码，删除它们，如图所示。



The screenshot shows the Dreamweaver interface with the 'video-player-design.html' file open. The left sidebar shows project files including CSS, JS, LESS, and media folders containing video files. The main code editor area displays the following code:

```
<!-- //////////////////////////////////////////////////////////////////-->
<!-- 以下代码是参考代码，参考之后请将它们删除
//////////////////////////////////////////////////////////////////--&gt;

&lt;figure class="video-container" style="display: none;"&gt;&lt;!-- 由JavaScript动态生成 --&gt;
    &lt;!-- 手工编写的HTML代码 --&gt;
    &lt;video id="yamoo9-video" poster="include/images/video-poster.jpg" controls width="835" height="417"&gt;
        &lt;source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8, Vorbis'" /&gt;
        &lt;source src="include/media/yamoo9-video.mp4" type="video/mp4; codec='avc1.42E01E, mp4a.40.2'" /&gt;
        &lt;a href="include/media/yamoo9-video.mp4"&gt;下载 : yamoo9-video.mp4&lt;/a&gt;
    &lt;/video&gt;
    &lt;!-- 视频控件：由JavaScript动态生成 --&gt;
    &lt;div class="video-controls"&gt;
        &lt;a class="play-btn" title="播放/暂停" accesskey="P"&gt;&lt;/a&gt;
        &lt;div class="seek"&gt;&lt;/div&gt;
        &lt;div class="timer" 00:00&gt;&lt;/div&gt;
        &lt;div class="volume-box"&gt;
            &lt;div class="volume-slider"&gt;&lt;/div&gt;
            &lt;a class="volume-btn" title="静音/取消静音" accesskey="M"&gt;&lt;/a&gt;
        &lt;/div&gt;&lt;!-- e: .video-controls --&gt;
    &lt;/div&gt;
&lt;/figure&gt; &lt;!-- e: .video-container --&gt;</pre>
```

为 video 元素设置 id 属性，代码如下。

```
<video id="yamoo9-video" poster="include/images/video-poster.jpg" controls, width="835" height="417">
    <source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8, Vorbis'" />
    <source src="include/media/yamoo9-video.mp4" type="video/mp4; codec='avc1.42E01E, mp4a.40.2'" />
    <a href="include/media/yamoo9-video.mp4">下载yamoo9-video.mp4</a>
</video>
```

<video> 与 </video> 之间的代码是手工输入的，它在文档中的位置如图所示。

```

<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->
<!--[if !(gt_ IE 9)|(IE)]><!--><html lang="zh" class="no-js modern"><!--<![endif]-->
<head>
<meta charset="utf-8" />
<!--[if lt IE 9]><meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" /><![endif]-->
<title>HTML5 Video Player Design - 视频播放器设计</title>
<link href="include/css/style.css" rel="stylesheet" />
<script src="include/js/libs/modernizr.min.js"></script>
<script src="include/js/libs/jquery.min.js"></script>
<script src="include/js/libs/jquery.UI.min.js"></script>
<link href="include/js/plugin/videoPlayer/jquery.videoPlayer.css" rel="stylesheet" />
<script src="include/js/plugin/videoPlayer/jquery.videoPlayer.js"></script>
<script src="include/js/script.js"></script>
</head>
<body>

<div id="page-wrap">

<video id="yamoo9-video" poster="include/images/video-poster.jpg" controls width="835"
height="417">
<source src="include/media/yamoo9-video.webm" type="video/webm; codec='VP8,
Vorbis'" />
<source src="include/media/yamoo9-video.mp4" type="video/mp4; codec='avc1.42E01E,
mp4a.40.2'" />
<a href="include/media/yamoo9-video.webm">下载yamoo9-video.webm</a>
</video>

29 </div> <!---- e: #page-wrap -->|
```

调用插件 & 设置初始值

进入 include/js 文件夹，打开 script.js 文件，添加如下代码。在代码中，首先查找视频元素，而后向其应用 videoPlayer() 插件。

```
$('#yamoo9-video').videoPlayer();      //查找yamoo9-video，并向其应用videoPlayer插件
```

```
/* script.js - 脚本, 2013 © yamoo9.com */
```

```
jQuery(function($) {
    $('#yamoo9-video').videoPlayer();
});
```

连接插件

下面我们开始编写 jQuery 视频播放插件 videoPlayer 代码。首先进入 include/js/plugin/videoPlayer 文件夹，里面有一个名为 jquery.videoPlayer.js 文件，打开它，里面已经存在一些插件模板代码。接下来，我们从初始设置部分开始编写代码。

```
/* jquery.videoPlayer.js - Begin
 * 2013 © yahoo9.com
 */
;(function($) {
    $.fn.videoPlayer = function(options) {
        // 初始设置
        var initSettings = {
            ...
        };

        // 覆盖设置
        options = $.extend(initSettings, options);

        // 插件处理代码
        return this.each(function() {
            ...
        });
    }; // 停止：插件

    // 从外部更改初始设置
    $.fn.videoPlayer.initSettings = {};
})(window.jQuery);
```

initSettings 变量用于保存 Object literal，在该 Object literal 中将插件设置一些初始值。首先添加两个 Option，一个用于表示视频播放器的设计是否改变，另一个用于修改 seek 内部 slider range 的颜色，代码如下。

```
var initSettings = {  
    change_theme : false,  
    color : 'pink'  
};  
  
// 初始设置  
var initSettings = {  
    change_theme : false, // 更改控件设计  
    color : 'pink' // 更改滑块区域颜色  
};
```

建视频播放器容器元素

① 在 return this.each(function()); 内部，首先保存要应用插件的元素，而后动态生成 figure 对象，并将其保存起来，代码如下。

```
var $vp = $(this),  
    $vp_container = $('');  
    // 将应用插件的对象保存到$vp变量中。  
    // 动态生成figure对象后，保存到$vp_container变量中。
```

② 调用 \$vp_container 的 addClass() 方法，向容器元素添加 video-container、options.change_theme 和 options.color 三个类，代码如下。

```
$vp_container  
.addClass('video-container')  
.addClass(options.color)  
.addClass(options.change_theme);
```

③ 调用 \$vp 的 wrap() 方法，将 \$vp 元素包裹到 \$vp_container 元素之中，\$vp_container 变量保存着动态生成的 figure 对象。

```
$vp.wrap($vp_container);
```

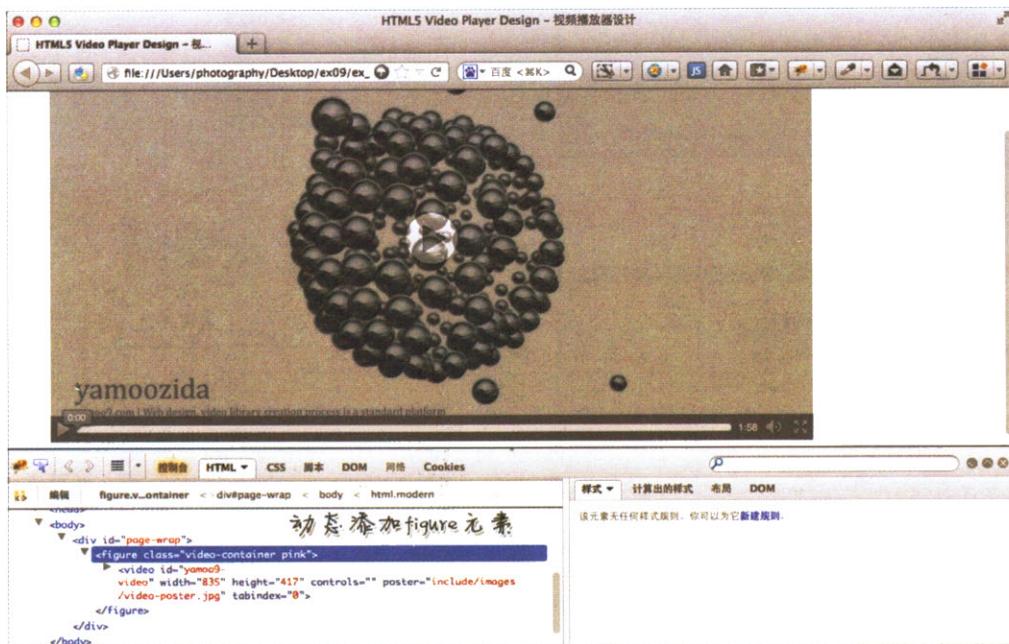
全部代码如下所示，仔细检查，确认代码编写无误。

```
// 插件处理代码
return this.each(function() {
    // 对象引用
    ① var $vp          = $(this),
        $vp_container = $('<figure/>');

    // 向$vp_container添加类
    ② $vp_container
        .addClass('video-container')
        .addClass(options.change_theme)
        .addClass(options.color);

    // 使用$vp_container元素包裹$vp
    ③ $vp.wrap($vp_container);
});
```

保存代码，在浏览器中打开页面，运行 Firebug 插件，在 HTML 窗口中，查看动态生成的代码，可以看到 figure 元素被动态添加上了，并且包含着 video 元素。



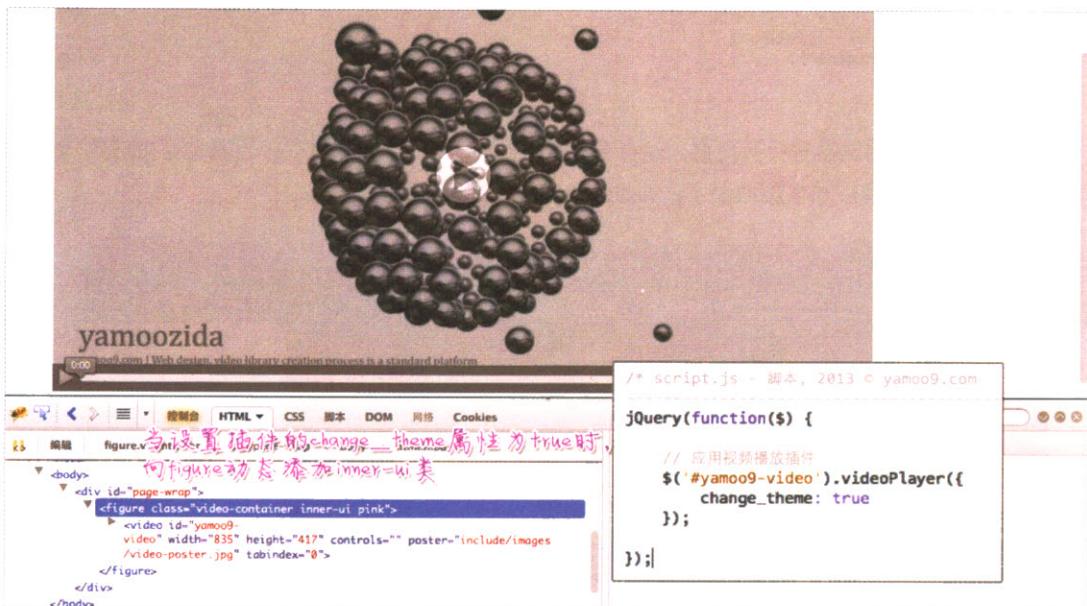
当传递给插件的 change_theme 值为 true 时，向 figure 元素的 class 属性中添加 inner-ui，在addClass() 语句之上添加如下代码。

```
options.change_theme = options.change_theme ? 'inner-ui' : '';
```

如上面语句所示，当 options.change_theme 的值为 true 时，设置 options.change_theme 的值为 'inner-ui'；当为 false 时，设置 options.change_theme 的值为空字符串。在右侧的条件表达式中，我们使用了三目运算符，根据 options.change_theme 的值，选择为左侧变量赋哪个值。它类似于 if 条件语句，但比 if 语句更简洁、更有效率。如下代码所示，在 script.js 文件中修改调用插件的代码，观察上述代码是否得到正常执行。

```
$('#yamoo9-video').videoPlayer({  
    change_theme: true          // 设置 videoPlayer 插件的 change_theme 属性为 true  
});
```

保存代码，在浏览器中打开页面，运行 Firebug 插件，在 HTML 窗口中可以看到向 figure 元素添加了 inner-ui 类。



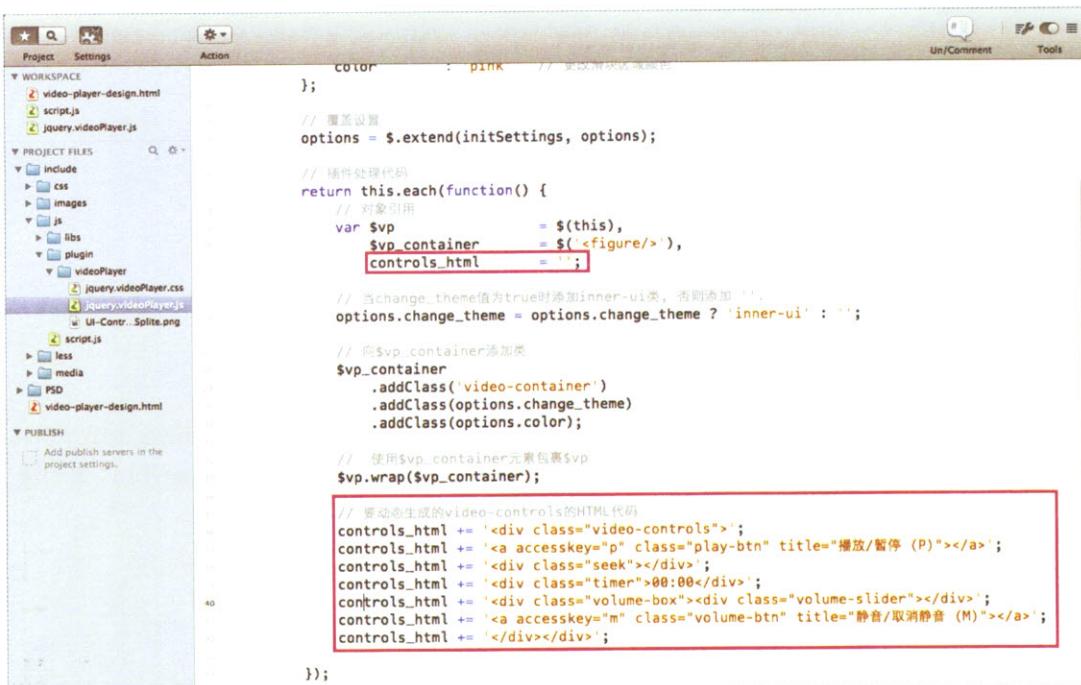
在第 274 页中将 inner-ui 类设置为 figure 元素的 class 属性后，视频控件在画面外部短暂停留后，又回到（默认的视频播放器）内部。总之，事先设置 inner-ui 类可以使后期维护变得更容易、更简单。



编写视频控件动态生成代码

下面我们开始在 jquery.videoPlayer.js 脚本文件中编写动态生成视频控件的代码。首先创建一个 controls_html 变量，赋空字符串给它，而后向其添加视频控件的 HTML 字符串。最后的 controls_html 变量中包含的是经连接后的全部视频控件的 HTML 字符串。然后将 controls_html 变量添加到 \$() 中，而后调用 insertAfter() 方法，把视频控件的 HTML 代码添加到 \$vp 之后。

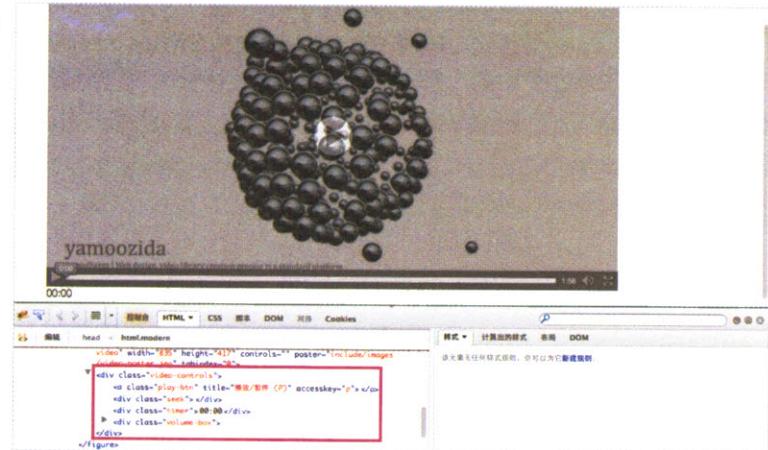
```
controls_html = '';
...从略
controls_html += '<div class="video-controls">';
controls_html += '<a accesskey="p" class="play-btn" title="播放/暂停(P)"></a>';
controls_html += '<div class="seek"></div>';
controls_html += '<div class="timer">00:00</div>';
controls_html += '<div class="volume-box"><div class="volume-slider"></div></div>';
controls_html += '<a accesskey="m" class="volume-btn" title="静音/取消静音(M)"></a>';
controls_html += '</div></div>';
```



接着，将动态添加的对象保存到 \$vp_controls 变量中，代码如下。

```
$vp_controls = $(controls_html).insertAfter($vp);
```

保存代码，在浏览器中打开网页，运行 Firebug 插件，可以看到脚本动态生成了视频控件，并且将代码插入到 <video> 元素之后。



到目前为止，我们编写的所有代码整理如下，请认真检查，避免出现错误。

```

options = $.extend(initSettings, options);

// 事件处理函数
return this.each(function() {
    var $vp           = $(this),
        $vp_container = $($(<figure>)),
        controls_html = '';

    // change_theme值为inner时为inner-ui类，否则添加
    options.change_theme = options.change_theme ? 'inner-ui' : '';
    // 将$vp_container设置为
    $vp_container
        .addClass('video-container')
        .addClass(options.change_theme)
        .addClass(options.color);

    // 使用$vp_container元素包裹$vp
    $vp.wrap($vp_container);

    // 要动态生成的video-control相关的HTML代码
    controls_html += <div class='video-controls'>;
    controls_html += <a accesskey='p' class='play-btn' title='播放/暂停 (P)'></a>;
    controls_html += <div class='seek'></div>;
    controls_html += <div class='timer'>00:00</div>;
    controls_html += <div class='volume-box'><div class='volume-slider'></div>;
    controls_html += <a accesskey='m' class='volume-btn' title='静音/取消静音 (M)'></a>;
    controls_html += </div></div>;

    $vp_controls = $(controls_html).insertAfter($vp);
});

};

// 从外部调用方法
$.fn.videoPlayer.initSettings = {};

```

在把视频插件的HTML代码动态添加到网页文档之后，创建一些变量来获取播放按钮(.play-btn)、播放条(.seek)、计时器(.timer)和音量控制(.volume-box、.volume-slider、.volume-btn)等对象。

```

var $play_btn = $vp_controls.find('.play-btn'),
    在$vp_controls内部查找.play-btn，而后保存到$play-btn中
$seek = $vp_controls.find('.seek'),
    在$vp_controls内查找.seek后保存到$seek中
$timer = $vp_controls.find('.timer'),
    在$vp_controls内查找.timer后保存到$timer中
$volume_box = $vp_controls.find('.volume-box'),
    在$vp_controls内查找.volume-box后保存到$volume-box中
$volume_slider = $vp_controls.find('.volume-slider'),
    在$vp_controls内查找.volume-slider后保存到$volume-slider中
$volume_btn = $vp_controls.find('.volume-btn');
    在$vp_controls内查找.volume-btn后保存到$volume-btn中

```

设置视频控件按钮

在把所有的视频控件对象保存到相关变量之后，通过变量调用相应的函数，对相应元素对象的 class 属性进行操作。当播放器 \$vp (保存 Video 对象的变量) 处于播放或暂停状态时，向 \$play_btn 添加或删除 pause-btn 类；当播放器播放结束时，清除 \$play_btn 的 pause-btn 类，代码如下。

```
$vp.bind('play pause', function() {  
    $play_btn.toggleClass('pause-btn');  
}).bind('ended', function() {  
    $play_btn.removeClass('pause-btn');  
});
```

为播放器的play与pause事件绑定处理函数
若\$play_btn拥有pause-btn类，则删除它；若无，则添加它
为播放器绑定ended事件处理函数
当视频播放器结束时，从\$play_btn中删除pause-btn类

当 \$play_btn 拥有 pause-btn 类时，播放按钮图标应该变为暂停按钮图标，我们通过向 \$play_btn 添加或删除 (toggle) pause-btn 类来实现。当播放器播放完视频时，清除 pause-btn 类，按钮再次变为播放按钮图标。关于图标的变换，我们将在 CSS 部分使用不同的背景图片来实现。



当单击控制按钮或直接单击视频画面时，要么播放视频，要么暂停视频播放。下面我们编写代码来实现这一功能。首先添加单击事件处理代码，设定单击事件发生时调用的函数，代码如下。

```
$vp.bind('play pause', function() {  
    $play_btn.toggleClass('pause-btn');  
}).bind('ended', function() {  
    $play_btn.removeClass('pause-btn');  
})  
add($play_btn).bind('click', playPause);
```

添加pause-btn类之后的形状

接着，编写 playPause 函数，该函数用于播放或暂停视频。HTML5 Video 对象本身提供了 play() 与 pause() 两个方法，能够实现对视频的播放与暂停操作，而 jQuery 并未提供视频播放与暂停方法。在使用 play() 与 pause() 两个方法之前，需要先将包裹在 jQuery 对象中的 Video 对象剥离出来，代码如下。

```
>>> $('video')  
jQuery(video#yamoo9-video)  
>>> $('video')[0]  
[video id="yamoo9-video" width="835" height="417" controls="" poster="include/images/video-poster.jpg" tabindex="0"]  
从jQuery对象中取出video对象
```

```
var vp = $vp[0];  
  
function playPause() {  
    if (!vp.paused) vp.pause();  
  
    else vp.play();  
}
```

为了使用video对象的play()与pause()方法，从jQuery对象中将其取出

检查vp（视频播放器）的暂停状态，若为播放，则暂停播放
若为暂停状态，则播放它

小知识 为何要将文档对象从jQuery()包裹对象中剥离出来？

当我们把文档对象包裹进jQuery()对象后，它就拥有了jQuery()对象强大的功能，我们可以随意使用这些功能。在电影《钢铁侠》中，主人公Tony Stark穿上了钢铁盔甲之后，便拥有了强大的力量，他可以随心所欲地使用它。我们将文档对象包裹进jQuery()对象就相当于为文档对象穿上了拥有强大力量的“钢铁盔甲”，可以使用jQuery对象的强大功能。

在这里，文档对象就相当于人，钢铁盔甲就相当于jQuery()对象。文档对象被包裹为jQuery()对象后即具有了jQuery()对象的强大功能。但是我们也知道人在穿上了钢铁盔甲后，有些事情反而做不了了，同样文档对象在包裹成jQuery()对象之后，它本身的某些功能也会失去。如果要使用文档对象本身的这些功能，需要我们先将文档对象从jQuery()对象中剥离出来。



HTML5的Video对象就属于这种情况。Video对象本身拥有play()与pause()两个操作视频的方法，但是jQuery本身没有单独提供操作视频的方法，当我们将Video对象包裹成jQuery对象后，原有的play()与pause()方法将无法再使用，除非将Video对象从jQuery对象中剥离出来。

播放视频时，除了通过控制按钮来播放或暂停视频外，还常常使用键盘上的空格键Space Bar，反复单击空格键Space Bar可以在播放与暂停之间来回切换。在大部分视频播放器中，也常常使用空格键Space Bar在视频的播放与暂停之间进行切换。这极大提高了用户使用视频的便利性，所以下面我们也将这种功能添加到网页播放器中。

在视频播放器页面中，当用户单击键盘上的空格键Space Bar时，将会触发键盘事件，然后调用相应的函数进行处理。当键盘事件发生时，会生成一个事件对象e，该对象将被作为参数传递给键盘事件处理函数。在键盘事件处理函数中，判断事件对象e的键值是否为32，若是，则调用playPause()函数。空格键Space Bar的键值码为32。

```

$(document).keydown(function(e){
    if (e.keyCode == 32) playPause();
});

```

保存代码，在浏览器中打开网页，显示出视频播放器，单击空格键 Space Bar 后开始播放视频，再次单击空格键 Space Bar 暂停视频播放。通过空格键 Space Bar 播放或暂停视频播放的代码非常精简，并且极大提高了用户使用视频的方便性，这是非常值得的。在本部分，我们编写的代码整理如图，请仔细检查，避免出现错误。

设置视频播放进度条

在设置完播放按钮之后，下面我们开始设置视频播放条。首先创建 createSeek() 函数，而后在其中添加处理代码。

在函数内部，首先要做的是判断 HTML5 Video 对象是否处于可播放状态，这就需要用到 Video 对象的 readyState 属性。readyState 属性会以数值的形式表示视频当前的就绪状态，指示视频是否已准备好播放，它有 5 种不同的值，分别表示视频不同的状态，如下表所示。关于 readyState 属性值的更多内容，请访问 HTML5 API (<http://is.gd/JSoCju>) 页面进行学习。

状态	返回值	说明
HAVE NOTHING	0	当前播放位置无有效媒介资源
HAVE_METADATA	1	加载中，媒介资源确认存在，但当前位置没有能够加载到有效媒介数据进行播放
HAVE_CURRENT_DATA	2	已获取到当前播放数据，但没有足够的数据进行播放
HAVE_FUTURE_DATA	3	已获取后续播放数据，可以进行播放
HAVE_ENOUGH_DATA	4	可以进行播放，且浏览器确认媒体数据以某一种速度进行加载，可以保证有足够的后续数据进行播放，而不会使浏览器的播放进度赶上加载数据的末端

如上表所示，当 readyState 属性值为 0 时，表示当前播放位置没有有效的媒介资源；当 readyState 属性值为 1~3 时，表示媒介资源此时已处于某种准备状态之下；当 readyState 属性值为 4 时，表示媒介资源已处于充分准备的状态下。在浏览器中运行 Firebug 插件，可以看到当前视频对象的 readyState 值为 3。

当文档 (document) 发生keydown 事件时运行处理函数
若用户按下空格键 (32)，则调用 playPause() 函数

```

controls.html += '<a accesskey="m" class="volume-btn" title="静音/取消静音" (M)></a>';
controls.html += '</div></div>';

$vp.controls = $(controls.html).insertAfter($vp);

var $play_btn = $vp.controls.find('.play-btn'),
$seek = $vp.controls.find('.seek'),
$timer = $vp.controls.find('.timer'),
$volume_box = $vp.controls.find('.volume-box'),
$volume_slider = $vp.controls.find('.volume-slider'),
$volume_btn = $vp.controls.find('.volume-btn'),
vp = $vp[0],
seek_sliding,
video.volume = 1;

// 点击静音/取消静音事件处理函数
$vp
    .bind('play pause', function() {
        $play_btn.toggleClass('pause-btn');
    })
    .bind('ended', function() {
        $play_btn.removeClass('pause-btn');
    })
    .bind('timeupdate', seekUpdate)
    .add($play_btn).bind('click', playPause);

// 触发按键热键事件处理函数
$(document).keydown(function(e) {
    if(e.keyCode == 32) playPause();
});

```

在 createSeek() 函数中，我们使用 if 条件语句对 vp（视频对象）的 readyState 值进行判断，当它的值为非 0（true），即媒体处于准备状态时，就执行 {} 内的代码段，该代码段包含创建视频播放条的代码；当它的值为 0（false）时，调用 setTimeout() 函数，每过 0.1 秒之后，再次调用 createSeek() 函数，检查 vp.readyState 属性值。



```
function createSeek() {
    if (vp.readyState) {
        var dur = vp.duration;
        // 使用jQuery UI Slider
    } else {
        setTimeout(createSeek, 100);
    }
}
```

检查 vp 的 readyState 状态（视频播放器）
若 readyState 存在返回值，则将 vp 总播放长度保存至 dur 变量中

若 readyState 返回值为 0，则表示尚未做好播放准备
0.1 秒之后再次调用 createSeek 函数

制作播放进度条的滑块

下面我们开始在 createSeek() 函数的 if 语句内部，使用 jQuery UI 的 Slider 编写代码，向播放进度条（Seek）中添加播放滑块。关于滑块插件的使用方法，请访问 jQueryUI (<http://jqueryui.com/demos/slider>) 网站进行学习。该网站详细地介绍了多种滑块效果的使用方法，希望大家拿出时间进入该网站学习。

[View Source](#)

```
<style>
    #demo-frame > div.demo { padding: 10px !important; }
</style>
<script>
    $(function() {
        $('#slider').slider();
    });
</script>

<div class="demo">
    <div id="slider"></div>
</div>(<!-- End demo -->)

<div style="display: none;" class="demo-description">
    <p>The basic slider is horizontal and has a single handle that can be moved with the mouse or by using the arrow keys.</p>
</div>(<!-- End demo-description -->)

```

Overview Options Events Methods Theming

Show details | Hide examples

► disabled	Boolean	Default: false
► animate	Boolean, String, Number	Default: false
► max	Number	Default: 100
► min	Number	Default: 0
► orientation	String	Default: 'horizontal'
► range	Boolean, String	Default: false
► step	Number	Default: 1
► value	Number	Default: 0
► values	Array	Default: null

```

var seek_sliding;          声明seek_sliding变量
//使用jQuery UI Slider
$seek.slider({             连接jQuery UI:slider插件至$seek引用的对象上
    orientation:'horizontal',   设置滑块方向为“水平”（默认为水平，可省略）
    value: 0,                  设置滑块的当前位置为0（默认设置，可省略）
    step: 0.01,                设置滑块的移动间隔为0.01，平滑移动
    range: 'min',              设置滑块区的开始位置为min（方向：从左到右）
    max: dur,                 设置滑块的最大值为视频的长度（dur保存着视频的长度）
    animate: true,             当使用鼠标点击播放条的某个位置时，播放滑块平滑移动
    slide: function() {
        seek_sliding = true;  滑动时，将seek_sliding变量设置为true
    },
    stop: function(e, ui) {
        seek_sliding = false;  停止滑动时，将seek_sliding变量设置为false
        vp.currentTime = ui.value; 将vp（视频播放器）的当前播放时间设置为ui.value
    }
});                         
```

到现在为止编写的代码整理如下。

```

var $play_btn      = $vp_controls.find('.play-btn'),
    $seek         = $vp_controls.find('.seek'),
    $timer        = $vp_controls.find('.timer'),
    $volume_box   = $vp_controls.find('.volume-box'),
    $volume_slider = $vp_controls.find('.volume-slider'),
    $volume_btn   = $vp_controls.find('.volume-btn'),
    vp            = $vp[0],
    seek_sliding;

// 向$vp添加事件处理函数
$vp
    .bind('play pause', function() {
        $play_btn.toggleClass('pause-btn');
    })
    .bind('ended', function() {
        $play_btn.removeClass('pause-btn');
    })
    .add($play_btn).bind('click', playPause);

// 按空格键时调用playPause函数
$(document).keydown(function(e) {
    if(e.keyCode == 32) playPause();
});

// 调用动态生成函数
createSeek();

// 播放/暂停函数
function playPause() {
    if(!vp.paused) vp.pause();
    else vp.play();
}; 
```

```

// seek 生成函数
function createSeek() {
    if(vp.readyState) {
        var dur = vp.duration;
        $seek.slider({
            step: 0.01,
            range: 'min',
            max: dur,
            animate: true,
            slide: function() {
                seek_sliding = true;
            },
            stop: function(e, ui) {
                seek_sliding = false;
                vp.currentTime = ui.value;
            }
        });
    } else {
        setTimeout(createSeek, 150);
    };
}

```

保存代码，在浏览器中运行 Firebug 插件，在 HTML 窗口中，可以看到在 `<div class="seek">` 内部动态添加了 div 与 a 元素，并且在各元素上添加了相应的类，这表明 Slider 插件运作正常。

▼ <div class="video-controls">

 jQuery16208913956274946857=Object { events=[...], handle=function() }
 ▶<div class="seek ui-slider ui-slider-horizontal ui-widget ui-widget-content ui-corner-all"> slider=Object { element=[...], options=[...], _keySliding=false, 更多... } jQuery16208913956274946857=Object { events=[...], handle=function() }
 <div class="ui-slider-range ui-widget-header ui-slider-range-min" style="width: 0%;"></div>
 jQuery16208913956274946857=Object { events=[...], handle=function() } index.uiSliderHandle=0
 </div>
 <div class="timer">00:00</div>
 ▶<div class="volume-box">
 </div>

向 `div.seek` 元素添加 `ui-slider` 类，在内部生成 `div.ui-slider-range`, `a.ui-slider-handle`

编写播放条状态更新函数

下面我们开始为播放状态条编写更新函数 `seekUpdate()`，当用户拖动播放控制滑块时，就会触发 `timeupdate` 事件，调用 `seekUpdate()` 函数进行处理。当 `seek_sliding` 的值为 `false` 时，`$seek.slider` 的 `stop` 事件就会被处理。即当用户拖动视频播放控制滑块后，先将 `current_time` 的值设置为滑块的 `value` 值，再将 `current_time` 中的时间格式化，然后更改 `timer` 中显示的时间。在这里，我们自行创建一个 `timeFormat()` 函数，并使用该函数对 `current_time` 的值进行格式化。

```

function seekUpdate() {
    var current_time = vp.currentTime;
    // 将vp (Video对象) 的当前播放时间 (秒) 保存到current_time变量中
    if (!seek_sliding) $seek.slider('value', current_time);
}

```

```

        若seek_sliding值为false，则将current_time设置为value
$timer.text(timeFormat(current_time));
        将$timer元素的内容更改为timeFormat(current_time)
);

```

timeFormat() 函数在接收 current_time 参数后，将其格式化为“分：秒”形式，而后将格式化后的时间进行返回。

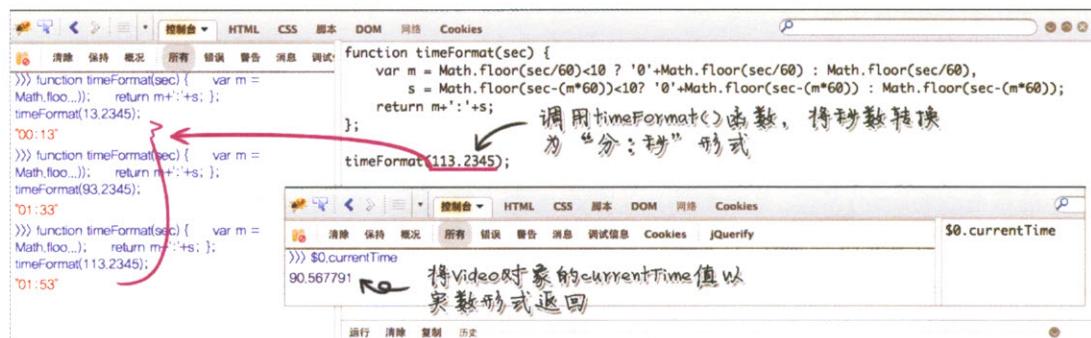
```

function timeFormat(sec) {
    var m = Math.floor(sec / 60) < 10 ? '0'+Math.floor(sec / 60) : Math.floor(sec / 60),
        s = Math.floor(sec - (m * 60)) < 10 ? '0'+Math.floor(sec - (m * 60)) : Math.floor(sec - (m * 60));
    return m + ':' + s;
}

```

timeFormat() 函数在接收 current_time 参数后，首先将 current_time 除以 60，得到分钟数，然后保存到 m 变量中。为了得到整数值，我们使用了 Math.floor() 函数，当得到的值小于 10 (若其值大于 10，则表示参数传递过来的时间为 600 秒以上，即大于 10 分钟) 时，由于不够 10 分钟，要在前面添加 0。当得到的值大于 10 分钟时，就不需要在前面再添加 0 了。

s 变量用来存放秒数，其值经过这样计算得来：首先将 m 乘以 60，把分钟转换成秒，而后用当前时间 current_time 减去秒数，再使用 Math.floor() 函数得到整数值，当得到的值小于 10 时，在前面添加 0，否则不添加 0。在分别计算出分钟与秒数之后，使用冒号 (:) 将它们连接起来后返回。



编写完 seekUpdate() 与 timeFormat() 函数后，将 seekUpdate() 函数绑定到 \$vp 对象的 timeupdate 事件上。

```

$vp.bind('play pause', function() {
    $play_btn.toggleClass('pause-btn');
}).bind('ended', function() {
    $play_btn.removeClass('pause-btn');
});
.bind('timeupdate', seekUpdate)
.add($play_btn).bind('click', playPause);

```

发生timeupdate事件时，调用seekUpdate函数

编写声音控制滑块

在前面编写视频播放滑块时应用了 Slider 插件。在这里，我们使用相同的方法，向声音控制滑块应用 Slider 插件，并改变相应的项目参数，代码如下所示。

```
var video_volume = 1;
$volume_slider.slider({
    orientation: 'vertical',
    value: 1,
    step: 0.01,
    range: 'min',
    max: 1,
    animate: true,
    slide: function(e, ui) {
        vp.muted = false;
        video_volume = ui.value;
        vp.volume = ui.value;
    }
});

```

声明video_volume变量，设置初值为1
连接jQuery UI:slider插件到\$volume_slider引用的对象上
设置滑块在垂直方向滑动
设置滑块的当前位置为1（音量的最大值为1）
设置滑块的移动间隔为0.01，平滑移动
设置滑块区的起始位置为min（方向：从min到max）
设置滑块的最大值为1（即100%）
当使用鼠标点击播放条的某个位置时，播放滑块平滑移动
当用户移动滑块时
设置vp对象的muted为false（取消静音）
保存ui.value值到video_volume变量中
保存ui.value值到vp对象的volume变量中

为声音调节按钮编写函数

下面我们为声音调节按钮编写 muteVolume() 函数。首先将 muteVolume() 函数设置为单击声音调节按钮时调用的处理函数，代码如下。

```
$volume_btn.click(muteVolume);
```

单击\$volume_btn时调用muteVolume函数

编写 muteVolume() 函数，它会根据 vp (Video 对象) 的 muted 属性值，执行不同的代码。

```
function muteVolume() {
    if(vp.muted) {
        vp.muted = false;
        $volume_slider.slider('value', video_volume);
    } else {
        vp.muted = true;
        $volume_slider.slider('value', 0);
        $volume_btn.toggleClass('volume-mute');
    }
}
```

若vp引用的对象的muted为true
设置vp.muted值为false
设置\$volume_slider的slider('value')值为video_volume
若vp引用的对象的muted属性为false
设置vp.muted值为true
设置\$volume_slider的slider('value')值为0
检查\$volume_btn是否存在volume-mute类，若不存在则添加，若存在则删除

最后，将 vp (Video 对象) 的默认控件（各个浏览器默认的播放控件 UI）设置为 false，使其处于不可见状态。当用户所用的浏览器支持 JavaScript 脚本时，浏览器的默认播放控件就会被隐藏起来。当然，如果用户使用的浏览器不支持 JavaScript 脚本，我们要将 controls 属性设置为 true，将浏览器默认的播放控件显示出来，以便观看视频。

```
vp.controls = false;
```

设置vp (Video对象) 的controls属性值为false

代码整理如下：

```
// 调用动态生成函数
createSeek();

// 向$volume_box设置slider UI插件
$volume_slider.slider({
    value: 1,
    orientation: "vertical",
    range: "min",
    max: 1,
    step: 0.05,
    animate: true,
    slide: function(e, ui){
        vp.muted = false;
        video_volume = ui.value;
        vp.volume = ui.value;
    }
});

// $volume_btn绑定单击事件处理函数
$volume_btn.click(muteVolume);

// 删除浏览器的基本controls
vp.controls = false;

// seek 更新函数
function seekUpdate() {
    var current_time = vp.currentTime;
    if(!seek_sliding) $seek.slider('value', current_time);
    $timer.text(timeFormat(current_time));
};

// 播放时间设置函数
function timeFormat(sec) {
    var m = Math.floor(sec/60)<10 ? '0'+Math.floor(sec/60) : Math.floor(sec/60),
        s = Math.floor(sec-(m*60))<10 ? '0'+Math.floor(sec-(m*60)) :
        Math.floor(sec-(m*60));
    return m+':'+s;
};

// 静音与取消静音函数
function muteVolume() {
    if(vp.muted) {
        vp.muted = false;
        $volume_slider.slider('value', video_volume);
    } else {
        vp.muted = true;
        $volume_slider.slider('value', 0);
    };
    $volume_btn.toggleClass('volume-mute');
};
```

保存代码，在浏览器中打开网页，运行 Firebug 插件后（按 F12 键），在 HTML 窗格中可以看到动态生成的代码，如下图所示。

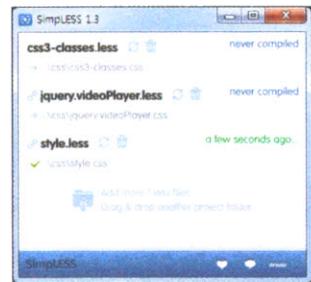
```
<body>
  <div id="page-wrap">
    <figure class="video-container inner-ui pink">
      <video id="yamoo9-video" width="835" height="417" poster="include/images/video-poster.jpg" tabindex="0">
        <source type="video/webm; codec='VP8, Vorbis'" src="include/media/yamoo9-video.webm">
        <source type="video/mp4; codec='avc1.42E01E, mp4a.40.2'" src="include/media/yamoo9-video.mp4">
        <a href="include/media/yamoo9-video.webm">下载yamoo9-video.webm</a>
      </video>
      <div class="video-controls">
        <a class="play-btn" title="播放/暂停 (P)" accesskey="p"></a> jQuery16206176839762356224=Object { events: [__], handle=function0 }
        <div class="seek ui-slider ui-slider-horizontal ui-widget ui-widget-content ui-corner-all"> slider=Object { element: [__], options: [__], _keySliding:false, _more... } jQuery16206176839762356224=Object { events: [__], handle=function0 }
          <div class="ui-slider-range ui-widget-header ui-slider-range-min" style="width: 0%;"></div>
          <a class="ui-slider-handle ui-state-default ui-corner-all" href="#" style="left: 0%;"></a> jQuery16206176839762356224=Object { events: [__], handle=function0 } index.uiSliderHandle=0
        </div>
        <div class="timer">00:00 </div>
        <div class="volume-box">
          <div class="volume-slider ui-slider ui-slider-vertical ui-widget ui-widget-content ui-corner-all"> slider=Object { element: [__], options: [__], _keySliding:false, _more... } jQuery16206176839762356224=Object { events: [__], handle=function0 }
            <div class="ui-slider-range ui-widget-header ui-slider-range-min" style="height: 100%;"></div>
            <a class="ui-slider-handle ui-state-default ui-corner-all" href="#" style="bottom: 100%;"></a> jQuery16206176839762356224=Object { events: [__], handle=function0 } index.uiSliderHandle=0
          </div>
          <a class="volume-btn" title="静音/取消静音 (M)" accesskey="m"></a> jQuery16206176839762356224=Object { events: [__], handle=function0 }
        </div>
      </div>
    </figure>
```

编写 CSS3 与 LESS 样式代码

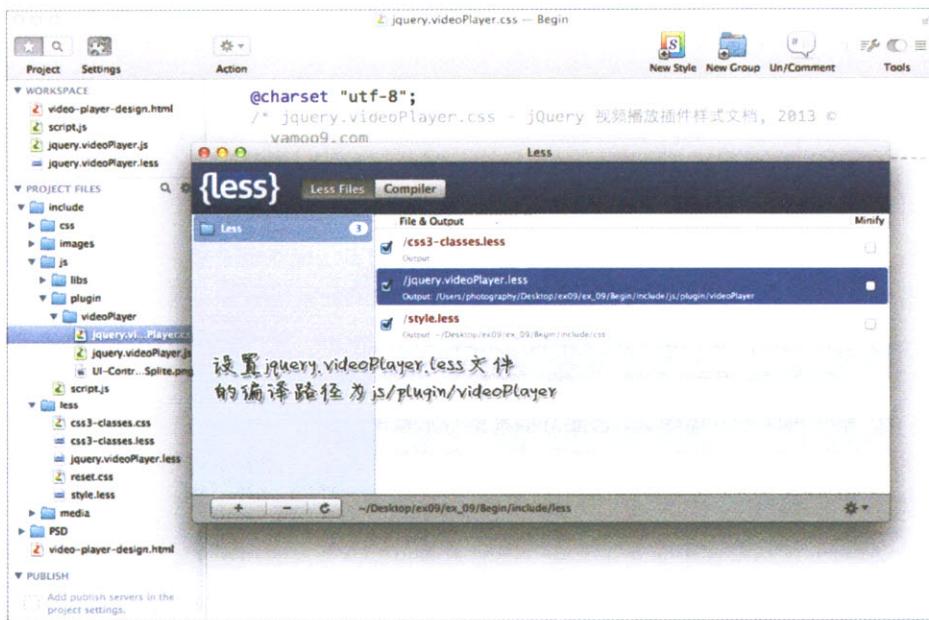


在使用 LESS 编写样式代码之前，如果你是 Windows 用户，要先安装 SimpLESS.exe 程序；如果你是 Mac 用户，则要安装 Less.app 程序。关于它们的安装与使用方法，在前面我们已经详细地讲解过了。如果忘记了，请重新翻到第 6 章第 179 页学习。

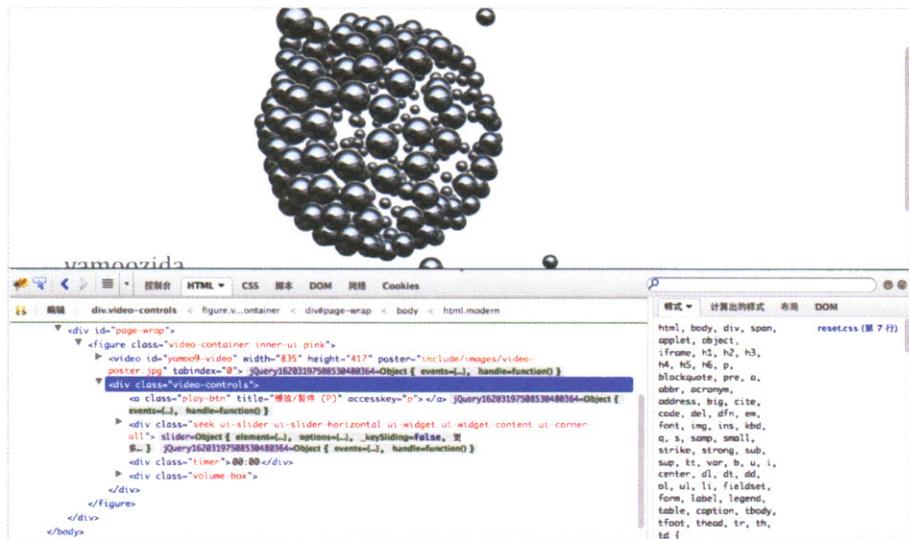
在 Windows 环境下，首先运行 SimpLESS 程序，而后将整个 begin 文件夹拖放到 SimpLESS 程序中。当在 LESS 文件中编写代码并保存后，SimpLESS 会自动帮我们转换成 CSS 代码，添加到指定的 CSS 文档中。在向 SimpLESS 中拖放项目文件夹时，若发生错误，请参考第 6 章第 183 页中的相关内容进行解决。



在 Mac 环境下，先运行 Less Compiler 程序，然后分别设置保存转换结果的文件夹目录。首先，指定 jquery.videoPlayer.less 文件到 include\js\plugin\videoPlayer 文件夹中，然后指定 style.less 文件到 include\css 文件夹中。



在编写样式控制代码之前，首先查看 videoPlayer 插件处理代码是否正常，如图所示。



基本样式控制

首先在 style.less 文件中编写基本样式控制代码。先为 #page-wrap 元素设置宽度为 835px，使其与视频元素的宽度保持一致，而后设置 margin 值，使其位于窗口的中央。然后向 <h1> 元素应用 CSS3 网页字体。

```
#page-wrap {
    width: 835px;          设置宽度为835px
    margin: 150px auto;     设置上下外边距为150px，左右外边距为auto
    text-align: center;      设置文本居中对齐
}
h1 {
    font-family: 'SansationLight'; 使用CSS 3的@font-face属性，应用SansationLight字体
    color: #f4397d;          设置字体颜色为粉红色
}
```

保存代码后，在浏览器中打开网页，可以看到标题已经应用上了指定的字体，播放器也移动到了窗口的中间。注意这里我们使用的 CSS3 网页字体并不是之前的云字体服务，而是使用 @font-face 语句直接调用了字体文件。我们常用的字体文件格式有 eot、woff、ttf 和 svg 等，其中 eot 是 IE 特有的字体格式；woff 是 Web Open Font Format 的缩写，它像 eot 格式一样轻便，是正在标准化的一种字体格式；ttf 是一种印刷字体格式，它显得很笨重且加载的速度慢，不太适合在网页中使用；svg 是一种矢量图形，它以图形的方式创建字体文件。由于每种浏览器处理的字体格式不同，因此在各种格式的字体之前需要先使用 url() 语句进行绑定。

由于每种浏览器所支持的字体格式有所不同，因此在使用网页字体时显得有些繁琐。字体格式有很多种，使用时需要经过多次不同的转换。通过使用 Font Squirrel ([@font-face kit's、@font-face Generator 服务](http://www.fontsquirrel.com)，可以非常容易地解决这一问题。

- 01** 进入 Font Squirrel (<http://www.fontsquirrel.com>) 网站，在菜单导航栏中，单击“@font-face Kits”菜单。在右上侧的搜索框中，输入“Sansation”。按 Enter 键，搜索字体。搜到指定的字体后，单击它，进入“Sansation”字体页面。



- 02** 在“Sansation”页面中，单击“@font-face Kits”菜单，进入“@font-face Kits”页面。

This screenshot shows the 'Sansation' @font-face Kit page. It features a large preview image of the font in a circular design. On the right, there's a sidebar with filters for 'FILTER' and 'STYLE'. At the bottom, there's a 'Download Now!' button.

- 03** 在“@font-face Kits”页面中，单击“Download @font-face Kit”按钮，下载 Sansation-fontfacekit.zip 文件。

This screenshot shows the same 'Sansation' @font-face Kit page as before, but with the 'Download @font-face Kit' button highlighted by a green arrow. The button is located in the center of the page under the font preview.

- 04** Sansation-fontfacekit.zip 文件下载完成后，解压缩，可以看到文件夹中包含一些 HTML 文档，以及可使用的网页字体数据文件 (TTF/EOT/SVG/WOFF)。复制 Sansation-fontfacekit 文件夹到 include/css 中，即可使用该网页字体。

This screenshot shows a Windows file explorer window displaying the contents of the 'Sansation-fontfacekit' folder. Inside, there are numerous files including various font formats (TTF, EOT, WOFF, SVG) and HTML files. A green arrow points from the text '单击“Download @font-face Kit”按钮，选择“提取到”' to the 'Extract Here' button at the bottom of the file explorer.

在 style.less 文件顶部，添加 @import "Sansation-fontfacekit/webfonts.css"; 语句，而后使用 font-family:'SansationLight'; 语句将其应用到 h1 标题上，如图所示。

```
@import "Sansation-fontfacekit/webfonts.css";
```

```

@charset "utf-8";
@import "reset.css";
@import "Sansation-fontfacekit/webfonts.css";
@import "css3-classes.less";
/* style.css - 基本样式文档, 2013 © yamoo9.com */

/*
 * Generated by Font Squirrel (http://www.fontsquirrel.com) on
 * October 5, 2011 12:26:41 PM America/New_York */

```

```

@font-face {
    font-family: 'SansationLight';
    src: url('Sansation_Light-webfont.eot');
    src: url('Sansation_Light-webfont.eot?#iefix') format('embedded-opentype'),
         url('Sansation_Light-webfont.woff') format('woff'),
         url('Sansation_Light-webfont.ttf') format('truetype'),
         url('Sansation_Light-webfont.svg#SansationLight')
              format('svg');
    font-weight: normal;
    font-style: normal;
}

```

为视频容器编写样式代码

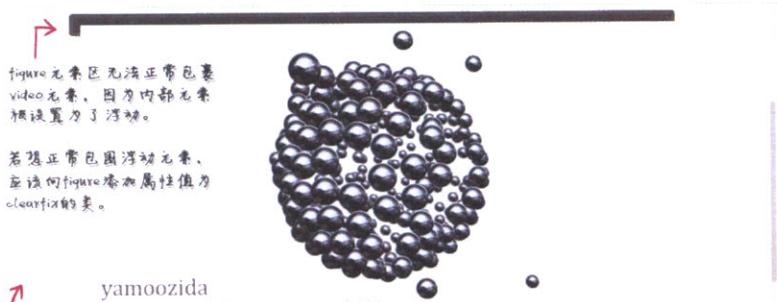
编写完基本的样式控制代码后，接下来，我们打开 jquery.videoPlayer.less 文件，选择 figure.video-container 后，为视频容器编写样式控制代码。

figure.video-container {	选择figure.video-container元素
margin: 0;	使设置在figure上的默认浏览器样式无效
border: 5px solid #61625d;	设置边框粗细为5px，实线，灰色
padding: 10px;	设置内边距为10px
font-family: Arial, Sans-Serif;	设置字体为Arial
.linear-gradient(#313131, #000);	应用线性渐变（调用.linear-gradient函数）
.border-radius(5px);	设置圆角边框效果（调用.border-radius函数）
.box-shadow(inset 0 15px 35px #535353);	设置内阴影效果（调用.box-shadow函数）
}	

接着，继续在 figure.video-container 内部，在 .box-shadow() 函数下，添加如下代码，设置 video、.video-controls、.play-btn、.seek、.timer 和 .volume-box 向左浮动。

figure.video-container {	
...	
video, .video-controls, .play-btn, .seek,	选择video、.video-controls、.play-btn、.seek、.timer、.volume-box元素
.timer, .volume-box { float: left; }	设置向左浮动（应用浮动布局）
}	

保存代码，在浏览器中打开网页，可以看到网页布局出现了问题，如下图所示。这是因为受到 video~.volume-box 浮动的影响，父元素 figure 无法再包裹住 video 元素了。



yamoozida
yamoozida.com Web design video library creation process a standard platform

figure元素无法正常包裹video元素，因为内部元素被设置为了浮动。

若想正常包围浮动元素，应该向figure添加属性值为clearfix的类。

jqeury.videoPlayer.less

```

figure {
    width: 100%; height: auto;
    position: relative;
    & .video-controls {
        position: absolute;
        bottom: 0;
        left: 0;
        width: 100px;
        height: 30px;
        background-color: #333333;
        border-radius: 15px;
        color: white;
        font-size: 14px;
        line-height: 30px;
        text-align: center;
        opacity: 0.8;
        transition: all 0.3s ease-in-out;
        &:hover {
            background-color: #444444;
            opacity: 0.9;
        }
    }
}

```

若想使父元素包裹住处于浮动状态的子元素，需要向父元素 figure 添加 clearfix 类。figure 元素是由脚本动态生成的，要为它添加 clearfix 类，就要先打开 jquery.videoPlayer.js 文件，找到 &vp_container 之后，使用 addClass() 方法添加 clearfix 类。

```

$vp_container
.addClass('video-container').addClass(options.color).addClass(options.change_theme)
.addClass('clearfix') css('width', $vp.width());

```

在上面代码中，我们还使用了 css() 方法，将 &vp_container 与 \$vp 的宽度设置为相同，这样视频容器 &vp_container 才能将视频对象 \$vp 完全包裹起来。保存修改后的代码，在浏览器中打开页面，可以看到前面的问题得到了很好的解决。

figure元素添加clearfix类、figure元素的宽度与Video对象的宽度相同

jqeury.videoPlayer.less

```

figure {
    width: 100%; height: auto;
    position: relative;
    & .video-controls {
        position: absolute;
        bottom: 0;
        left: 0;
        width: 100px;
        height: 30px;
        background-color: #333333;
        border-radius: 15px;
        color: white;
        font-size: 14px;
        line-height: 30px;
        text-align: center;
        opacity: 0.8;
        transition: all 0.3s ease-in-out;
        &:hover {
            background-color: #444444;
            opacity: 0.9;
        }
    }
}

```

视频控件样式 1：整体

返回到 jquery.videoPlayer.less 文件中，为所有视频控件编写控制样式。在 figure.video-container 中，为 .video-controls 添加样式代码，如下所示。

```
.video-controls {
    position: relative;
    width: 100%;
    margin-top: 5px;
}
```

选择.video-controls元素
设置为相对定位（内部绝对定位元素的基准点）
设置宽度为100%（由于是浮动的，故需要设置宽度）
设置上外边距为5px

请看下图，它显示了 LESS 代码转换为 CSS 后的结果。从中可以看出，使用 LESS 编写样式代码比直接使用 CSS 编写样式要简洁省力得多。

The screenshot shows a LESS editor interface with a project structure on the left and the LESS code on the right. The LESS code defines a .video-controls class with a relative position, 100% width, and a 5px top margin. It also defines a .video-container class with a 0px margin, 5px border, 10px padding, Arial font-family, a linear gradient background, a 5px border-radius, and a 15px by 35px box shadow. Inside .video-container, there are .video, .video-controls, .play-btn, .seek, .volume-box, and .timer elements, all set to float: left;. The .video-controls class is defined with a relative position, 100% width, and a 5px top margin. The converted CSS output on the right shows the resulting styles, with annotations explaining the LESS features used:

- 转换后的CSS代码**: Points to the converted CSS code.
- 使用LESS的嵌套规则编写**: Points to the .video-controls definition within .video-container.
- 使用LESS的新float属性**: Points to the float: left; declaration applied to internal elements.
- 使用嵌套语法的好处是可以省略重复的父元素**: Points to the .video-controls definition within .video-container.

```
@charset "utf-8";
@import "css3-classes.less";
/* jquery.videoPlayer.css - jQuery 视频播放插件样式文档, 2013 © yamo9.com */

/*
 * //////////////////////////////////////////////////////////////////
 * .video-container
 * ////////////////////////////////////////////////////////////////// */
figure.video-container {
    margin: 0; // 初始化浏览器的基本样式
    border: 5px solid #e1625d;
    padding: 10px;
    font-family: Arial, Sans-Serif;
    linear-gradient(#313131, #000);
    border-radius: 5px;
    box-shadow:inset 0 15px 35px #535353;
}
figure.video-container video,
figure.video-container .video-controls,
figure.video-container .play-btn,
figure.video-container .seek,
figure.video-container .volume-box,
figure.video-container .timer {
    float: left;
}
figure.video-container .video-controls {
    position: relative;
    width: 100%;
    margin-top: 5px;
}

/*
 * //////////////////////////////////////////////////////////////////
 * .video-controls
 * ////////////////////////////////////////////////////////////////// */
.video-controls {
    position: relative;
    width: 100%; // 应用float后设置width
    margin-top: 5px;
} // .video-controls

} // figure.video-container
```

视频控件样式 2：播放按钮

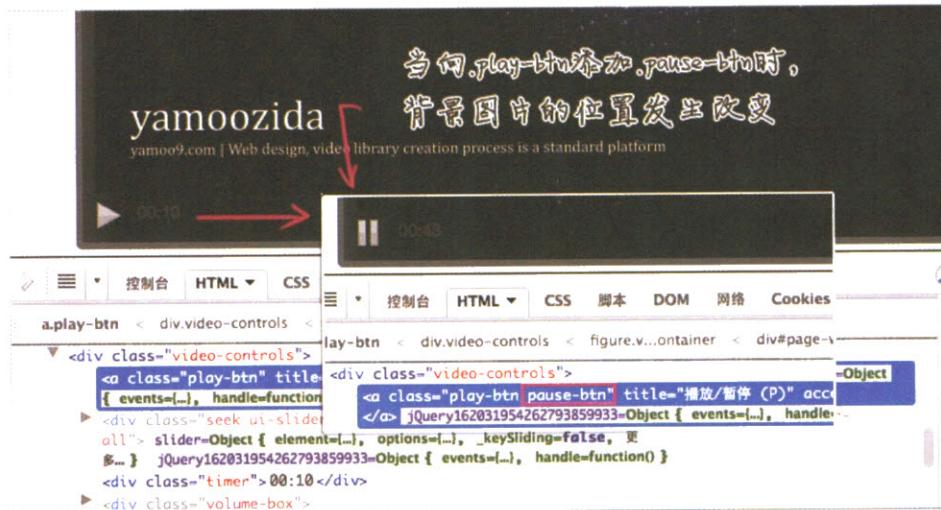
在为整个视频控件编写好样式控制代码后，我们为视频控件中的播放按钮添加样式控制代码。下面首先为播放按钮 (.play_btn) 添加样式控制代码，如下所示。

```
.video-controls {
    ...从略
    .play-btn {
        display: block;
        width: 22px;
        height: 22px;
        margin-right: 15px;
        background: url(UI-Controls-Split.png) no-repeat;
        opacity: 0.7;
        transition(opacity .2s ease-in-out);
        &:hover { opacity: 1; }
        &.pause-btn { background-position: -22px 0; }
    }
}
```

在.video-controls内部使用.play-btn嵌套（LESS嵌套）
转换为块元素
设置宽度为22px（显示背景图片的区域）
设置高度为22px（显示背景图片的区域）
设置右外边距为15px
插入背景图片，不反复
设置不透明度为0.7
设置CSS3的transition为0.2秒
进入.play-btn:hover状态时，设置不透明度为1
进入.play-btn.pause-btn状态时，调整背景图片的位置

在为播放按钮 (.play-btn) 编写样式代码时，我们为它添加了一张背景图片，这张图片是我们已经提前制作好的，该图片中包含着播放、暂停、声音打开和声音关闭 4 种图标。

保存代码，在浏览器中查看网页，可以看到在播放按钮部分出现了播放图标，如图所示。当用户单击播放按钮或空格键时，视频开始播放，并且播放按钮也变为暂停按钮。在 Firebug 插件中查看，可以看到视频开始播放后，播放按钮被切换为暂停按钮，图标也由播放图标更改为暂停图标。

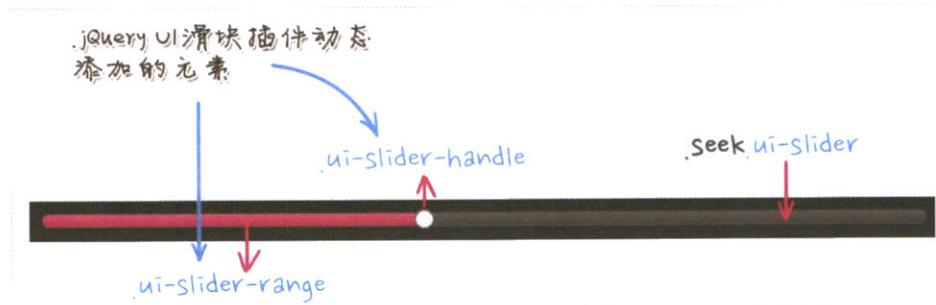


为了告诉用户 .play-btn 是一个按钮，在这里我们将 cursor 属性设置为 pointer。同时也需要告诉用户 .volume-btn 也是按钮，所以我们将它与 .play-btn 放在一起，同时设置它们的 cursor 属性设置为 pointer，代码如下。保存代码，在浏览器中查看网页，当将鼠标指针移动到播放按钮上时，鼠标指针即变为手形形状。

```
.play-btn, .volume-btn { cursor: pointer; }           // 设置.play-btn与.volume-btn的cursor属性为pointer
```

视频控件样式 3：播放进度条

在播放进度条 (.seek) 与声音控制 (.volume-box) 元素中，jQuery UI Slider 插件会动态生成 .ui-slider-range、.ui-slider-handle 两个元素，它们分别指视频已播放的部分与播放滑块，如图所示。



```

.ui-slider-range {
    position: absolute;
    z-index: 1;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 100%;
    border: 0;
}

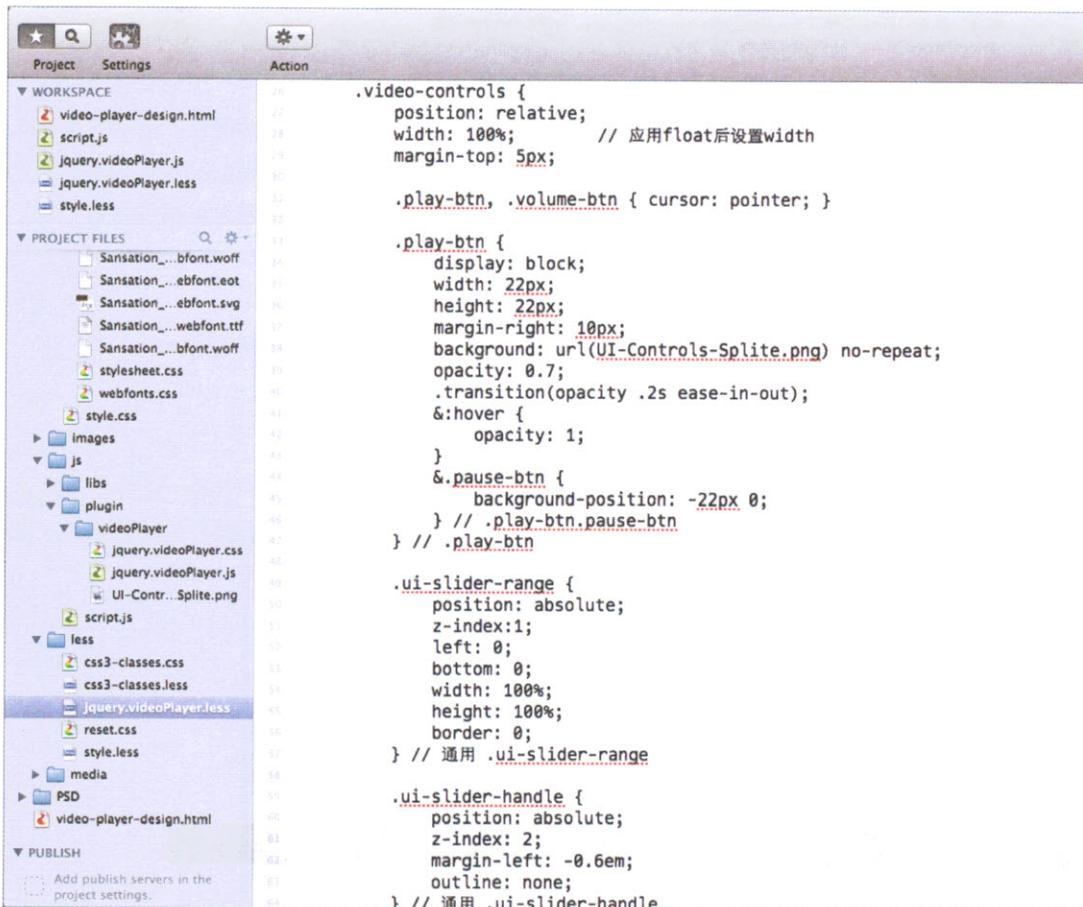
.ui-slider-handle {
    position: absolute;
    z-index: 2;
    margin-left: -0.6em;
    outline: none;
}

```

在.video-controls内使用.ui-slider-range嵌套
设置为绝对定位（基准元素为.ui-slider）
设置Z轴上的堆叠顺序为1
设置left属性为0
设置底部边缘到基准元素底边缘上
设置宽度与其包含元素宽度相同
设置高度与其包含元素高度相同
删除边框

在.video-controls内使用.ui-slider-handle嵌套
设置为绝对定位（基准元素为.ui-slider）
设置Z轴堆叠顺序为2（高于.ui-slider-range元素）
设置左外边距为-0.6em
删除轮廓线

到现在为止，我们已经为.play-btn、.play-btn:hover、.play-btn 和 .pause-btn，以及 UI range、handle 元素编写好了样式控制代码，整理如下。



The screenshot shows a LESS editor interface with a toolbar at the top and a project structure sidebar on the left. The main area displays the generated CSS code for the video player controls.

```

.video-controls {
    position: relative;
    width: 100%; // 应用float后设置width
    margin-top: 5px;

    .play-btn, .volume-btn { cursor: pointer; }

    .play-btn {
        display: block;
        width: 22px;
        height: 22px;
        margin-right: 10px;
        background: url(UI-Controls-Splite.png) no-repeat;
        opacity: 0.7;
        transition(opacity .2s ease-in-out);
        &:hover {
            opacity: 1;
        }
        &.pause-btn {
            background-position: -22px 0;
        } // .play-btn.pause-btn
    } // .play-btn

    .ui-slider-range {
        position: absolute;
        z-index: 1;
        left: 0;
        bottom: 0;
        width: 100%;
        height: 100%;
        border: 0;
    } // 通用.ui-slider-range

    .ui-slider-handle {
        position: absolute;
        z-index: 2;
        margin-left: -0.6em;
        outline: none;
    } // 通用.ui-slider-handle
}

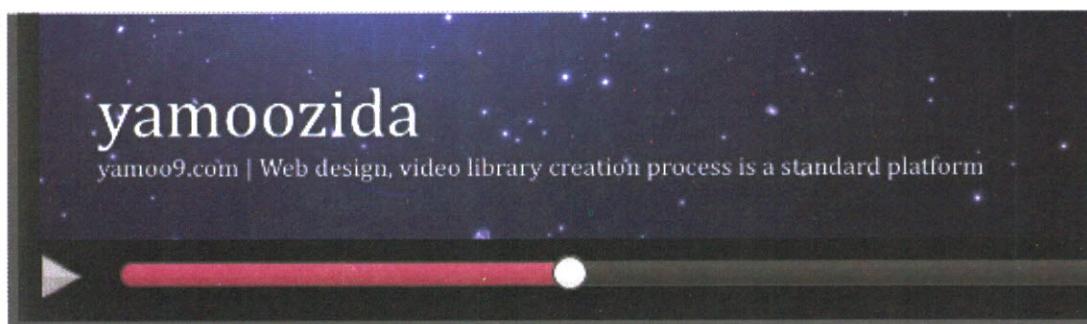
```

上面我们已经为 jQuery Slider 插件的 UI 共同元素编写了样式代码，接下来，选择整个播放进度条 (.seek)，为它添加样式代码。在 .video-controls 中，为 .seek 编写样式代码，并且在 .seek 中，分别为 .ui-slider-range、.ui-slider-handle 添加样式代码，如下所示。

```
.video-controls {  
    ...从略  
    seek {  
        position: relative;  
        width: 86%;  
        height: 10px;  
        margin-top: 5px;  
        border: 1px solid #494949;  
        .border-radius(15px);  
        .linear-gradient(#535353, #333);  
        .box-shadow(inset 0 -3px 3px #333);  
        .ui-slider-range {  
            .border-radius(15px);  
            .linear-gradient(#f4397d, #bd0a4b);  
            .box-shadow(inset 0 -3px 3px #bd0a4b);  
        } // .ui-slider-range  
        .ui-slider-handle {  
            top: -4px;  
            left: -4px;  
            width: 15px;  
            height: 15px;  
            border: 1px solid #333;  
            .border-radius(10px);  
            .linear-gradient(#e6e6e6, #d5d5d5);  
            .box-shadow(inset 0 3px 3px #d5d5d5);  
        } // .ui-slider-handle  
    } // .seek  
} // .video-controls
```

在 video-controls 内使用 seek 套
设置相对定位（基准元素设定）
设置宽度为 86%（父元素宽度的 86%）
设置高度为 10px
设置上外边距为 5px
设置边框粗细为 1px，实线，颜色
设置 CSS3 圆角边框为 15px（LESS 函数）
应用 CSS3 的线性渐变效果（LESS 函数）
应用 CSS3 内阴影效果（LESS 函数）
在 seek 内部使用 ui-slider-range 套
设置 CSS3 圆角边框为 15px（LESS 函数）
应用 CSS3 的线性渐变效果（LESS 函数）
应用 CSS3 内阴影效果（LESS 函数）
在 seek 元素内部使用 ui-slider-handle 套
从基准元素（.seek）向上移动 -4px
从基准元素（.seek）向左移动 -4px
设置宽度为 15px（圆形滑块的宽度）
设置高度为 15px（圆形滑块的高度）
设置边框为 1px，实线，颜色为 #333
设置 CSS3 圆角边框为 15px（LESS 函数）
应用 CSS3 的线性渐变效果（LESS 函数）
应用 CSS3 内阴影效果（LESS 函数）

保存代码，在浏览器中查看网页，可以看到 .seek 元素及其内部元素都应用上了指定的样式，如图所示。



视频控件样式 4：计时器

在为 .seek 元素编写好样式代码后，我们开始为计时器（.timer）编写样式代码。计时器（.timer）是 .video-controls 的一个组成部分。在 .video-controls 中，为 .time 添加样式控制代码，如下所示。

```
.video-controls {  
    ...从略  
    .timer {  
        margin: 3px 0 0 5px;  
        font-size: 12px;  
        font-weight: bold;  
    } // .timer  
} // .video-controls
```

在.video-controls内部使用.timer进行嵌套
分别设置上外边距、右外边距、下外边距、左外边距为3px, 0, 0, 5px
设置字体大小为12px
设置字体为粗体



视频控件样式 5：声音控件

在 .video-controls 中需要进行样式修饰的是 .volume-box 元素，该元素内部包含 .volume-btn 与 .volume-slider 两个子元素。而在 .volume-slider 内部又包含 jQuery Slider UI 动态生成的 .ui-slider-range、.ui-slider-handle 两个元素。在我们为 .volume-box 元素编写样式控制时，要逐个为这些元素编写，首先为 .volume-box 与 .volume-btn 两个元素编写样式控制代码。

```
.video-controls {  
    ...从略  
    .volume-box {  
        overflow: hidden;  
        position: absolute;  
        right: 0;  
        bottom: 0;  
        width: 20px;  
        height: 30px;  
        color: #fff;  
        padding: 0 10px;  
        .volume-btn {  
            ...  
        } // .volume-btn  
    } // .volume-box  
} // .video-controls
```

在.video-controls内部使用.volume-box嵌套
隐藏溢出.volume-box的子元素
设置为绝对定位（基准元素为.video-controls）
右对齐到基准元素
下对齐至基准元素
设置宽度为20px
设置高度为30px
设置字体颜色为白色
设置上下内边距为0px，左右内边距为10px
在.volume-box内部使用.volume-btn嵌套

```

position: absolute;          设置为绝对定位 ( 基准元素: .volume-box )
right: 2px;                 从基准元素右侧左移2px
bottom: 0;                   对齐到基准元素的底边
width: 22px;                设置宽度为22px
height: 22px;               设置高度为22px
opacity: 0.7;               设置不透明度为0.7
background: url(UI-Controls-Splite.png) no-repeat -44px 0;
                           设置背景图片的位置
                           .transition(opacity .2s ease-in-out);
                           设置 CSS3 的 transition 为 0.2 秒
&:hover { opacity: 1; }      进入 .volume-btn:hover 状态时设置不透明度为 1
&.volume-mute { background-position: -66px 0; }
                           进入 .volume-btn.volume-mute 状态时，更改背景位置
} // .volume-btn
} // .volume-box
} // .video-controls

```

保存代码，在浏览器中查看网页，可以看到在视频播放器的右侧已经出现了声音控制按钮。当移动鼠标指针到它上面时，声音控制按钮的不透明度逐渐变为 1，即按钮逐渐变亮，如图所示。



视频控件样式 6：声音调节条

最后，我们为声音调节条编写样式控制代码。声音调节条 (.volume-slider) 由 .ui-slider-range 与 .ui-slider-handle 两个子元素组成。在声音控件 .volume-box 的声音调节条 (.volume-slider) 中，分别为它们编写样式代码，代码如下。

```

.video-controls {
  ...从略
  .volume-box {
    ...从略
    &:hover {
      height: 130px;
      padding-top: 10px;
      .volume-slider {
        visibility: visible;
        opacity: 1;
      }
    } // .volume-box:hover
    .volume-slider {
      visibility: hidden;
      position: relative;
    }
  }
}

```

当进入 .volume-box:hover 状态时
设置高度为 130px。
设置上内边距为 10px。
在 .volume-box:hover 状态下选择 .volume-slider。
在画面上显示出来（若非 :hover 状态，则隐藏）。
设置不透明度为 1（若非 :hover 状态，则设为 0）。

在 .volume-box 内部使用 .volume-slider 嵌套。
在画面上隐藏（在 .volume-box:hover 状态下可见）。
设置为相对定位（基准元

```

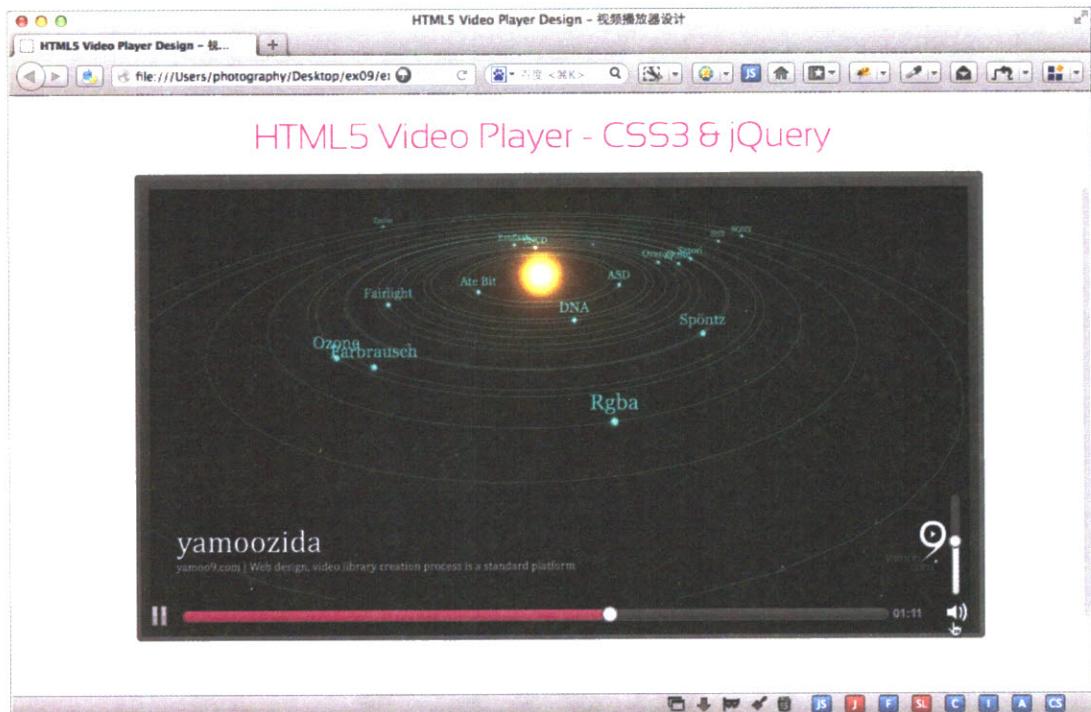
    left: 4px;
    height: 100px;
    width: 7px;
    opacity: 0;
    margin-left: 7px;
    border: 1px solid #444;
    .border-radius(15px);
    .linear-gradient(#535353, #333);
    .box-shadow(inset 0 3px 3px #333);
    .transition(opacity .2s ease-in-out);
    .ui-slider-range {
        .border-radius(15px);
        .linear-gradient(#e6e6e6, #d5d5d5);
        .box-shadow(inset 0 3px 3px #d5d5d5);
    } // .volume-slider .ui-slider-range
    .ui-slider-handle {
        width: 12px;
        height: 12px;
        left: -4px;
        margin-bottom: -0.6em;
        margin-left: 0;
        border: 1px solid #333;
        .border-radius(10px);
        .linear-gradient(#e6e6e6, #d5d5d5);
        .box-shadow(inset 0 3px 3px #d5d5d5);
        &.ui-state-hover {
            background: #fff;
        } // .ui-slider-handle.ui-state-hover
    } // .volume-slider .ui-slider-handle
} // .volume-slider
...从略
} // .volume-box
} // .video-controls

```

保存代码，在浏览器中查看网页，可以看到我们为声音调节条（.volume-slider）添加的样式。当将鼠标指针移动到小喇叭图标上时，就会显示出声音调节滑动条，如图所示。



至此，我们已经为视频播放器编写好了样式控制代码。虽然代码量大，花费了不少时间与精力，但最后得到了一个漂亮的视频播放器，所有的辛苦付出都是值得的。仔细查看样式控制效果，检查是否有问题。

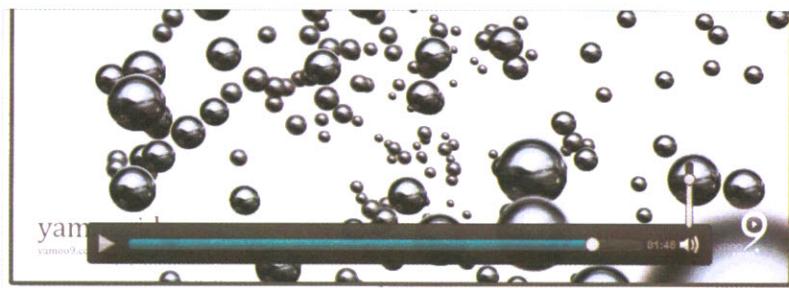


设置播放控制条到视频内部，并改变颜色

到这儿，我们设计的视频播放器该进入收尾阶段了。对于视频播放器的设计，虽然现在我们已经可以停止了，但是稍微再下些功夫，我们就能够设计出更利于后期修改维护的视频播放器来。设计一个便于后期维护修改的播放器是十分重要的，比如，在我们设计的视频播放器中视频控件位于视频容器的底部，但是客户可能要求播放时隐藏视频控件，当观看者移动鼠标指针到视频画面上时视频控件又显示出来；客户还可能要求把视频控件设置到视频内部，而不是视频的底部。像这样，当客户的要求发生变化时，难道我们要再重新编写视频播放器吗？

我们已经做过的事情，再让我们重新做一次，这的确是件让人沮丧的事儿。不过，不用担心。为了应对这种可能出现的情况，在制作 jQuery 插件时，我们提供了一个可以用来添加 inner-ui 类的选项。好好想一想，我们还提供了一个更改滑动条颜色的选项，通过该选项，我们可以根据客户的要求随时更改滑动条的颜色。有了这些可以随时改变的选项，我们就不需要再重新编写代码了，提高了程序的灵活性，也节省了大量的时间与精力，大大降低了后期修改维护的成本。

首先，打开 include/js/script.js 文件，修改插件调用代码，如下所示，即分别修改花括号 {} 中的 change_theme 与 color 的值。

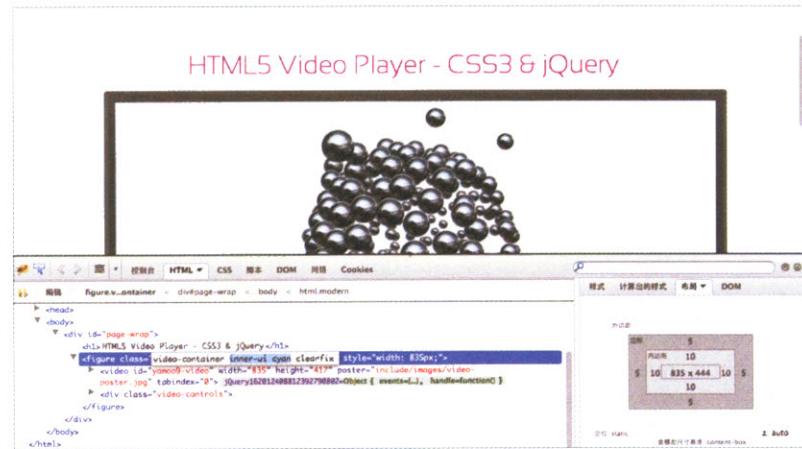


当把鼠标指针放到视频上时，播放控件就显示出来，且显示在画面内部，进度条呈现青色。

```
$( "#yamoo9-video" ).videoPlayer({
    change_theme: true,
    color: "cyan"
});
```

向 videoPlayer 插件添加选项
设置 change_theme 值为 true, color
值为 cyan

保存代码，在浏览器中打开网页，运行 Firebug 插件，可以看到程序已经为 figure 元素添加了 inner-ui 与 cyan 两个类，它们是在调用插件时设定的。通过这种灵活的选项设置，能够为我们后期维护修改提供极大的便利性。



接下来，向 jquery.videoPlayer.less 文件添加 .inner-ui 类。在 figure.video-container 中添加 .inner-ui 类后，在 .video-controls 中编写样式控制代码。

```
figure.video-container {  
    ...  
  
    &.inner-ui {  
        position: relative;  
        padding: 0;  
        video:hover + .video-controls {  
            visibility: visible;  
            opacity: 0.8;  
        }  
        .video-controls {  
            visibility: hidden;  
            opacity: 0;  
            position: absolute;  
            bottom: 20px;  
            left: 80px;  
            width: 80%;  
            padding: 10px;  
            border: 1px solid #2e2e2e;  
            .border-radius(5px);  
            .linear-gradient(#313131, #000);  
            .box-shadow(inset 0 15px 35px #535353);  
            .transition(all 0.4s ease-in-out);  
            &:hover {  
                visibility: visible;  
                opacity: 1;  
            }  
            .seek { width: 85%; }  
            .volume-box {  
                ...  
            }  
        }  
    }  
}
```

选择 figure.video-container.inner-ui
设置相对定位（作为 .video-controls 的基准位置）
设置内边距为0
当进入 video:hover 状态时，选择出现的 .video-controls
显示在画面上（若非 :hover 状态，则不显示）
设置不透明度为0.8
选择 .inner-ui .video-controls
从画面隐藏（若为 :hover 状态，则显示）
设置不透明度为1
设置绝对定位（基准元素为 .inner-ui ）
从基准元素底边上移 20px
从基准元素左边右移 80px
设置宽度为 80%（父元素 .inner-ui 的 80% ）
设置内边距为 10px
设置边框为 1px，实线，颜色
设置 CSS3 圆角边框为 5px（调用 LESS 函数）
应用 CSS3 线性渐变效果（调用 LESS 函数）
应用内侧阴影效果（调用 LESS 函数）
应用 CSS3 transition 为 0.4 秒（调用 LESS 函数）
当进入 .video-controls:hover 状态时
在画面上显示出来
设置不透明度为 1
选择 .video-controls .seek，设置宽度为 85%
选择 .video-controls .volume-box

```

bottom: 10px;                                从基准元素底边上移10px
&:hover { height: 100px; }                  当进入.video-box:hover状态时设置高度为100px
.volume-slider {                            选择.video-box内部的.volume-slider
    height: 70px;                           设置高度为70px
    left: -5px;                            从基准元素的左边向左移5px
    .ui-slider-handle { left: -3px; }        选择.ui-slider-handle，从基准元素的左边向左移3px
} // .volume-slider
.volume-btn { margin-right: 10px; }           选择.video-box内部的.volume-btn，设置右外边距为10px
} // .inner-ui .volume-box
} // .inner-ui .video-controls
} // figure.video-container.inner-ui
} // figure.video-container

```

保存.inner-ui类后，在网页浏览器中查看网页，可以看到播放器控件隐藏后，移动鼠标指针到视频上时，播放控件又会慢慢地显示出来。当然视频控件位于视频的内部。最后，修改jquery.videoPlayer.less文件，更改滑动条的颜色。首先声明一个LESS变量，通过该变量，我们为滑动条设置颜色。

```
@cyan: #1ac2d5;
```

接下来，查找要更改滑动条颜色的部分，即.seek .ui-slider-range。首先查找.ui-slider-range代码，删除.linear-gradient()与.box-shadow()语句。

```

.ui-slider-range {                            .seek元素内部的.ui-slider-range
    .border-radius(15px);
    .linear-gradient(#f4397d, #bd0a4b);      删除!
    .box-shadow(inset 0 -3px 3px #bd0a4b);    删除!
} // .ui-slider-range

```

要修改滑动条颜色的部分由添加到figure.video-container上的颜色关键字处理。在LESS变量下编写如下代码。

```

@cyan: #1ac2d5;
figure.video-container.cyan {                若figure元素带有cyan类
    @slider-begin-color: @cyan;              将@cyan赋给@slider-begin-color
    .seek .ui-slider-range {                选择.seek .ui-slider-range
        .linear-gradient(@slider-begin-color, darken(@slider-begin-color, 20%));   调用LESS的渐变函数
        .box-shadow(inset 0 -3px 3px darken(@slider-begin-color, 20%));            调用LESS的阴影函数
    }
}

```

在上面的代码中，darken() 函数是 LESS 颜色函数。它有两个参数，第一个参数为待变暗的颜色，第二个参数指定变暗的程度。在实例中，我们使用该函数将 @slider-begin-color 变暗 20%。除了 darken() 函数外，LESS 还提供了 lighten、saturate、desaturate、fadein、fadeout 和 spin 等函数。关于 LESS 函数更多的内容，请访问 LESS API (<http://is.gd/c98D1k>) 进行学习。

The screenshot shows a web browser displaying the LESS documentation for color functions. The page title is "LESS - The Dynamic Stylesheet language". The main content area is titled "Color functions" and contains the following text:

LESS provides a variety of functions which transform colors. Colors are first converted to the *HSL* color-space, and then manipulated at the channel level:

```

lighten(@color, 10%);      // return a color which is 10% *lighter* than @color
darken(@color, 10%);      // return a color which is 10% *darker* than @color

saturate(@color, 10%);    // return a color 10% *more* saturated than @color
desaturate(@color, 10%);  // return a color 10% *less* saturated than @color

fadein(@color, 10%);      // return a color 10% *less* transparent than @color
fadeout(@color, 10%);     // return a color 10% *more* transparent than @color

spin(@color, 10);          // return a color with a 10 degree larger in hue than @color
spin(@color, -10);         // return a color with a 10 degree smaller hue than @color

```

Below this, there is a section titled "Using them is pretty straightforward:" containing a code snippet:

```

@base: #f04615;

.class {
  color: saturate(@base, 5%);
  background-color: lighten(spin(@base, 8), 25%);
}

```

At the bottom, it says "You can also extract color information:".

我们刚刚编写的代码段是一个颜色模块。若要添加多种颜色，只要复制多个代码段，然后 @ 颜色名称与修改相关颜色值即可。为了后期修改维护方便，我们在 jquery.videoPlayer.less 文件开头部分定义了多种颜色变量，如图所示。

The screenshot shows a LESS IDE interface with a project workspace. The left sidebar shows the project structure:

- WORKSPACE: jquery-videoPlayer.less, script.js
- PROJECT FILES:
 - css: reset.css, Sanacion_fondfacekit, Sanacion_eblfont.eot, Sanacion_eblfont.svg, Sanacion_eblfont.ttf, Sanacion_blinkoff, webfonts.css, style.css
 - js: js, less, plugin: videoPlayer, jquery.videoPlayer.js, U-Contro_Sprite.png, script.js
 - less: less, less-classes.less, jquery-videoPlayer.less, style.less
 - media: video-player-design.html
- PUBLISH: Add publish servers in the project settings

The right pane displays the LESS source code, which includes color definitions and some CSS rules:

```

// 清块颜色
@pink : #f4497d;
@red  : #e61a19;
@yellow: #f1c44d;
@orange: #f28a2f;
@green : #80d51b;
@cyan  : #1ac2d5;
@blue   : #27b2ff;
@deepblue: #1933d5;
@purple : #5b19d5;
@gold   : #efca76;
@silver : #bfc1bf;

// 填充颜色
.video-container.pink {
  @slider-begin-color: @pink;
  .seek, .ui-slider-range {
    linear-gradient(@slider-begin-color, darken(@slider-begin-color, 20%));
    box-shadow(inset 0 -3px 3px darken(@slider-begin-color, 20%));
  }
}

.video-container.red {
  @slider-begin-color: @red;
  .seek, .ui-slider-range {
    linear-gradient(@slider-begin-color, darken(@slider-begin-color, 20%));
    box-shadow(inset 0 -3px 3px darken(@slider-begin-color, 20%));
  }
}

.video-container.yellow {
  @slider-begin-color: @yellow;
  .seek, .ui-slider-range {
    linear-gradient(@slider-begin-color, darken(@slider-begin-color, 20%));
    box-shadow(inset 0 -3px 3px darken(@slider-begin-color, 20%));
  }
}

```

到这里，视频播放器全部编写完成。为了制作 HTML5 Video 播放器，我们编写了大量 HTML、LESS 和 jQuery 的代码。虽然付出了很多时间与精力，但看到如此精美的播放器，心里仍然会非常高兴。制作时，我们把 videoPlayer 播放器制作成了插件形式，这样只要向 HTML 添加 HTML5 <video> 代码，而后在 script.js 文件中，将插件关联到相应的元素上，修改相应的选项后，即可改变播放器的外观。

```
/* script.js - 脚本, 2013 © yamoo9.com */  
jQuery(function($){  
  
    // 应用视频播放插件  
    $('#yamoo9-video').videoPlayer({  
        change_theme: true,  
        color: 'cyan'  
    });  
  
    $('#yamoo9-other-video').videoPlayer({  
        change_theme: false,  
        color: 'green'  
    });  
});
```

选择新添加的<video>元素的id，连接到videoPlayer插件，而后设置相应选项，即可显示视频播放器

前面我们编写的 HTML5 Video 播放器要求浏览器本身支持 HTML5，如果浏览器本身不支持 HTML5，播放器就会出现问题。这个时候，我们应该怎么办呢？当用户的浏览器不支持 HTML5 时，我们可以使用 Flash 播放器来代替 HTML5 Video 播放器。

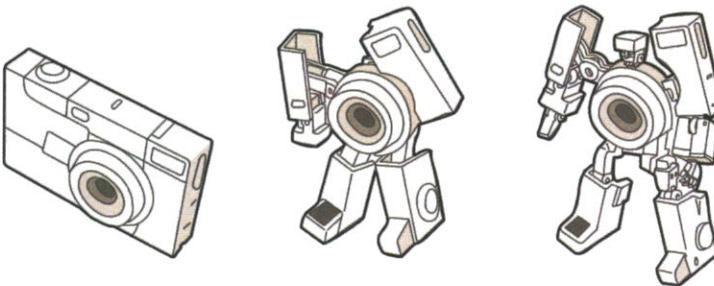
10 章

制作设备感应型网页

本章是全书的最后一章。在前面 9 章中，我们通过实例学习了使用 Web 标准平台技术的方法，以及如何设计 UI，相信大家对这些内容已经有了一定了解与掌握。在本章中，我们将制作一个设备感应型网页，以适应用户不同的访问设备，如个人 PC、智能手机和平板电脑等设备。

设计效果预览

在本章中，我们将一起学习制作一个设备感应型网页，所谓设备感应型网页是指网页能够自动感应用户所用的上网设备，并进行自我调整，以适应相应的上网设备，就像变形金刚一样。现在，市面上有各种各样的上网设备，如智能手机、平板电脑、个人PC和智能电视等，我们能够随时随地使用这些设备浏览网页，给我们的生活、工作和学习带来极大的便利，但另一方面也增加了网页开发的复杂度，要求网页开发者开发出能够适应不同设备的网页。



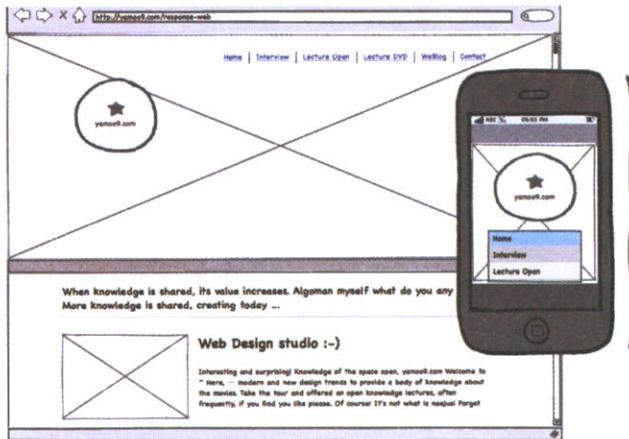
随着大量上网设备的涌现，开发能够适用于各种设备的网页变得越来越困难。即使只开发针对于桌上PC的网页，也要根据不同的操作平台与浏览器开发不同的网页，这是个非常艰辛的过程。开发能够同时适应不同上网设备的网页更是难上加难，但是令人高兴的是不断出现的新技术正在一点一点地帮助我们解决这一问题。在本章中，我们将开发一个能够同时适应桌上PC、智能手机和平板电脑的网页。



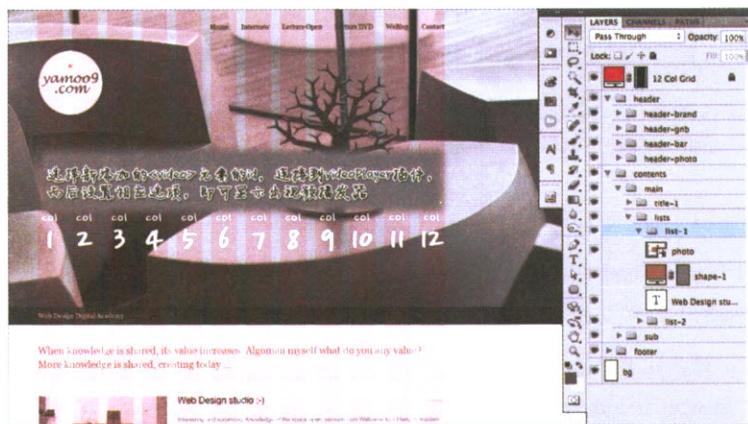
CHAPTER 10

设计网页布局

首先分别考虑桌面 PC 与智能手机不同的运行环境，使用 Balsamiq Mockups 软件绘制出网页的布局简图，如右图所示。



网页设计师在设计网页时首先要考虑网页的布局，确定要采用的网页布局结构，只有使用了恰当的网页布局结构，才能将要表达的信息准确地传递给观众。在这里，我们采用传统的网格系统（Grid System）来设计网页的布局，使用这种网格系统，能够帮助我们设计出结构既稳定，又美观的页面来。我们使用的网格由 12 个具有相同尺寸的列构成，各列之间拥有相同的间隔（Gutter），但最后一列没有间隔（Gutter），如图所示。



编写 HTML5 网页代码

下载实例源文件，解压缩后，进入 ex_10\Begin 文件夹，使用网页编辑器打开 responsive_web_01.html 文件，其中包含了一些常见代码。



```
<!DOCTYPE html>
<!--[if IE 6]><html class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html class="no-js modern ie9"><![endif]-->
<!--[if (gt IE 9)|(!(IE))]><!--><html class="no-js modern"><!--><![endif]-->
<head>
<meta charset="utf-8" />
<title>设备感应网页设计 - Responsible WebDesign</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0" /> ← 包含针对智能手机的优化代码
<link rel="stylesheet" href="include/css/style.css" />
<script src="include/js/libs/modernizr.min.js"></script>
<script src="include/js/libs/jquery.min.js"></script>
<script src="include/js/jquery.scrolledFix.js"></script>
<script src="include/js/script.js"></script>
<!--[if IE]><meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" /><![endif]-->
</head>
<body lang="zh-CN">

<!-- IE兼容性处理 -->
<!--[if lt IE 9]><script src="include/js/libs/CFInstall.min.js"></script>
<script>window.attachEvent("onload",function(){CFInstall.check({mode:"overlay"}));</script><![endif]-->
<!--[if IE 6]><script src="include/js/libs/DD_belatedPNG.min.js"></script>
<script>DD_belatedPNG.fix('.png_bg, img');</script><![endif]-->
</body>
</html>
```

在代码中虽然有些不曾见过，但大部分我们都见过，比较容易理解，在这里我们就不再赘述了。仔细观看代码，会发现在 `<title>` 元素之下新添加了一个 `<meta>` 元素，如下所示。

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

上述代码是专为智能手机编写的，用于为智能手机设置 Viewport。该行代码在桌面 PC 中不会发生作用。在 `content` 中，`width` 用于设置 `viewport` 的宽度，这里我们将其设置为设备宽度（Device-Width），`initial-scale` 用于设置初始的缩放比例，在这里我们将其设置为 x1.0。

运行 Photoshop 软件，在实例源文件夹的 PSD 文件夹中，打开 mockup.psd 文件，然后参考 PSD 文件与图层，编写 HTML 文档代码。虽然这一过程有些费时，但在网页设计实务中常常会经历这个过程，所以希望大家耐心做一下。假设你已经这么做了，就打开实例源文件中的 responsive_web_02.html 文档，下面开始一一分析其中的代码。

```

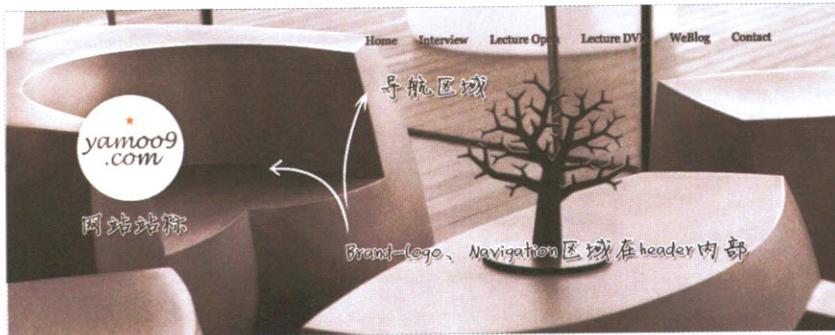
<header>
  <h1 id="brand"><a href="http://yamoo9.com/">yamoo9.com</a></h1>
  <nav id="gnb">
    <a href="#contents" class="blind" skip-navigation="a">
      <ul>
        <li><a href="http://yamoo9.com/" title="首页">Home</a></li>
        <li><a href="http://yamoo9.com/interview" title="访谈">Interview</a></li>
        <li><a href="http://is.gd/v8AyZH" title="开放知识讲座">Lecture Open</a></li>
        <li><a href="http://yamoo9.com/lecture-dvd/" title="讲XDVD Lecture DVD">Lecture DVD</a></li>
        <li><a href="http://is.gd/j3k6t1" title="我们的故事">WeBlog</a></li>
        <li><a href="http://is.gd/BcmrcX" title="联系">Contact</a></li>
      </ul>
    </a>
  </nav>
</header>
<div id="header-bar">
  <p>
    <a href="http://yamoo9.com/">
      
    </a>
    Web Design Digital Academy
  </p>
</div>
<section id="contents">
  <article id="main">
    <h2>When knowledge is shared, its value increases. Algomann myself what do you any value? More knowledge is shared, creating today ...</h2>
    <ul>
      <li>
        <h3>Web Design studio :-)</h3>
        <p>Interesting and surprising! Knowledge of the space open, yamoo9.com Welcome to ~ Here, — modern and new design trends to provide a body of knowledge about the movies. Take the tour and offered an open knowledge lectures, often frequently, if you find you like please. Of course! It's not what is naejus! Forget word of mouth, please? Is an open-to-date knowledge of teaching posts. Click the image for a larger image will be shown. Click the title of video services that are provided to determine if you can do it ...</p>
      </li>
    </ul>
  </article>
</section>

```

responsive_web_02.html 文档只是众多结构化方式中的一种，仅供参考，当然你完全可以使用其他的结构化方式，只要每个元素都能正确使用即可。至于网页中的文本，从网络上随便选择一些文字使用就行，没必要非得使用实例中使用的文字。

header (logo 与导航) 与 header bar

下面我们开始分析 responsive_web_02.html 文档中的代码。观察文档，可以发现文档大致由 header、contents 和 footer 三部分构成。其中 header 部分包括 logo 与导航两个部分，如图所示。



```

<header>
  <h1 id="brand"><a href="http://yamoo9.com/">yamoo9.com</a></h1>
  <nav id="gnb">
    <a href="#contents" class="blind" skip-navigation="a">
      <ul>
        <li><a href="http://yamoo9.com/" title="首页">Home</a></li>
        <li><a href="http://yamoo9.com/interview" title="访谈">Interview</a></li>
        <li><a href="http://is.gd/v8AyZH" title="开放知识讲座">Lecture Open</a></li>
        <li><a href="http://yamoo9.com/lecture-dvd/" title="讲XDVD Lecture DVD">Lecture DVD</a></li>
        <li><a href="http://is.gd/j3k6t1" title="我们的故事">WeBlog</a></li>
        <li><a href="http://is.gd/BcmrcX" title="联系">Contact</a></li>
      </ul>
    </a>
  </nav>
</header>

```

```

<li><a href="http://yamoo9.com/lecture-dvd/" title="讲义DVD">Lecture DVD</a></li>
<li><a href="http://is.gd/3JK6ti" title="我们的故事">WeBlog</a></li>
<li><a href="http://is.gd/BcmrcX" title="联系">Contact</a></li>
</ul>
</nav>
</header>

```

然后是 header-bar 区域，它紧接着 header 部分，我们使用 `<div>` 来设计它，当用户向下拖动浏览器的滑动条，使顶部区域不可见时，header-bar 区域中的图片与菜单就会显示出来。



```

<div id="header-bar">
<p>
    <a href="http://yamoo9.com/">
        
    </a> Web Design Digital Academy
</p>
</div>

```

contents 区域

在 header-bar 区域之下是 contents 区域，我们使用 `<section>` 元素对它进行结构化，其内包含两个 `<article>` 元素，将网页中的内容分为两部分，id 分别为 main 与 sub，如图所示。

When knowledge is shared, its value increases. Algomam myself what do you any value? More knowledge is shared, creating today ...

main、sub 属 contents 内部

Web Design studio :-)

Interesting and surprising knowledge of the space open, yamoo9.com Welcome to ... Very modern and new design trends to provide a body of knowledge about the movies. Take the tour and there an open knowledge lectures, often frequently. If you find you like please. Of course, it is not what is true. Forget world of course, please! It is an open-to-date knowledge of teaching, posts. Click the image for a larger image will be shown. Click the title of video services that are provided to determine if you can do it.

Did the design and teaching how to start?

From the beginning was not that you were a designer. Looked very much like being nervous or not knowing what to do. Even today, now. Come to a really real "The last running". As in football, it's a sports manager. Unlike the general director of the technical aspects of several managers, released in the thin spots do their excellent depiction of the psychological aspects happen. Minami Atsushi's teamwork manager, since the end of the first freekick. Exciting future deployment will be made.

图为 main 的 article 区域

```

<section id="contents">
    <article id="main"> ... </article>
    <article id="sub"> ... </article>
</section>

```

两个`<article>`元素分别由标题、图片与文字三部分组成。除了标题外，图片与文字都使用列表元素进行结构化，它们能够将图片与文字归在一起。在使用``元素插入图片时，不需要设置`alt`属性，因为在图片下面紧接图片的说明文字，防止图片说明文字出现重复。

```
<article id="main">
  <h2>When knowledge is shared, ... </h2>
  <ul>
    <li>
      
      <h4>Web Design studio :-) </h4>
      <p>Interesting and surprising! ... </p>
    </li>
    <li>
      
      <h4>Did the design and teaching how to start? </h4>
      <p>From the beginning was not that you wanted to design, ... </p>
    </li>
  </ul>
</article>
```

以上代码是 main 区域的代码，sub 区域与 main 区域有类似的结构，但是标题要低一个级别，当然图片的尺寸也有所不同。

Please love me. You can do something, do not get what you want to continue. Do not let it, and the environment. "This is too ... Because of that 'rather than complain, people who can not change a good thing Be yourself.



id="sub" article

		
<p>Are now also work when you were a kid were you interested in such things?</p> <p>Well ... As I recall, when you were a kid, you did not know their own ability to think with. At the time, I used to stand in front of South Korean students, I reviewed big questions why people were feeling I remember standing in front, from around</p>	<p>Currently working as an instructor asks about hasineunde career so far.</p> <p>A brilliant speaker as a career is not require sentence. Many an interesting jungle (the Academy) was active in the I did not have any other educational institutions (laughs), now have a corporate lecture, but started teaching</p>	<p>The field continues to develop new technology is emerging technical knowledge in this field in any way for studying the world?</p> <p>Knowledge was scattered all around, I can not find here ... My influences come mainly from abroad information, North and South America, Europe, Japan, and I'm learning rather than obscure. Of course, in Korea, the good news is shocking. Oh ... Would think a misunderstanding. Ha, ha, ha. And the most important point is to find every day I'm studying. The weekend, they do not cease. Why? It's funny!</p>

```
<article id="sub">
  <h3>Please love me. ... </h3>
  <ul>
    <li>
      
      <h5>Are now also work when you were a kid were you interested in such things? </h5>
      <p>Well ... As I recall, ... </p>
    </li>
    <li>
```

```


<h5> Currently working as an instructor asks about hasineunde career so far. </h5>
<p> ... A brilliant career as a career ... </p>
</li>
<li>
    
    <h5> The field continues to develop new technology is ... </h5>
    <p> Knowledge was scattered all around, I can not find here ... </p>
</li>
</ul>
</article>

```

在页面最底部的是 footer 区域，该区域主要用于为 <article> 元素定义“脚注”部分，如文章的版权信息、作者授权信息等。



```

<footer>
    <p><small> © 2007 - 2011 <strong>yamoo9</strong> ... </small></p>
</footer>

```

到这儿，HTML 文档就编写完成了。保存 HTML 代码，打开浏览器，查看网页，如下所示。

编写 CSS3 与 LESS 样式代码



在本部分中，我们将使用已有的 CSS 框架（Framework）为网页编写 CSS 样式代码。CSS 框架是一系列 CSS 的集合，包含了基本的元素重置、页面排版、网格布局、表单样式和通用规则等代码块。使用 CSS 框架，可以帮助我们简化网页前端的开发，提高开发效率，编写的 CSS 代码也更清晰、简单，便于后期维护。在前面的内容中，我们已经使用过一个 CSS 框架——reset.css，它用来重置浏览器的默认样式，而在本部分我们将使用另外一个广受欢迎的 CSS 框架——normalize.css。

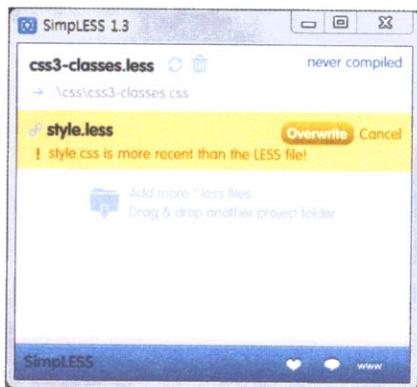
设置基本环境（下载 normalize.css）

normalize.css 是一个可定制的 CSS 文件，它让不同的浏览器在渲染网页元素的时候形式更统一。与 reset.css 不同，它并不重置所有的 CSS 样式，而是保留了有用的默认值，而对那些会产生浏览器兼容性问题的元素样式进行调整。**①** 进入 normalize.css 网站 (<http://necolas.github.com>)。**②** 单击“normalize.css”，进入 normalize.css 页面。**③** 在 normalize.css 页面中，单击右侧的“Downloads”按钮，进入下载页面。**④** 在下载页面中，根据所用电脑的操作系统，选择相应的压缩文件进行下载。下载完成后，解压缩后，将其复制到 css 文件夹中。

The image shows three screenshots of the GitHub repository for normalize.css.
 1. The first screenshot shows the main page of the repository with a red box around the 'Source code' button.
 2. The second screenshot shows the 'Code' tab of the repository page with a red box around the 'Downloads' button.
 3. The third screenshot shows the 'Downloads' page where users can choose between 'Download as zip' and 'Download as tar.gz'. A red box highlights the 'zip' link, and a red number '4' is placed at the bottom center of this screen.

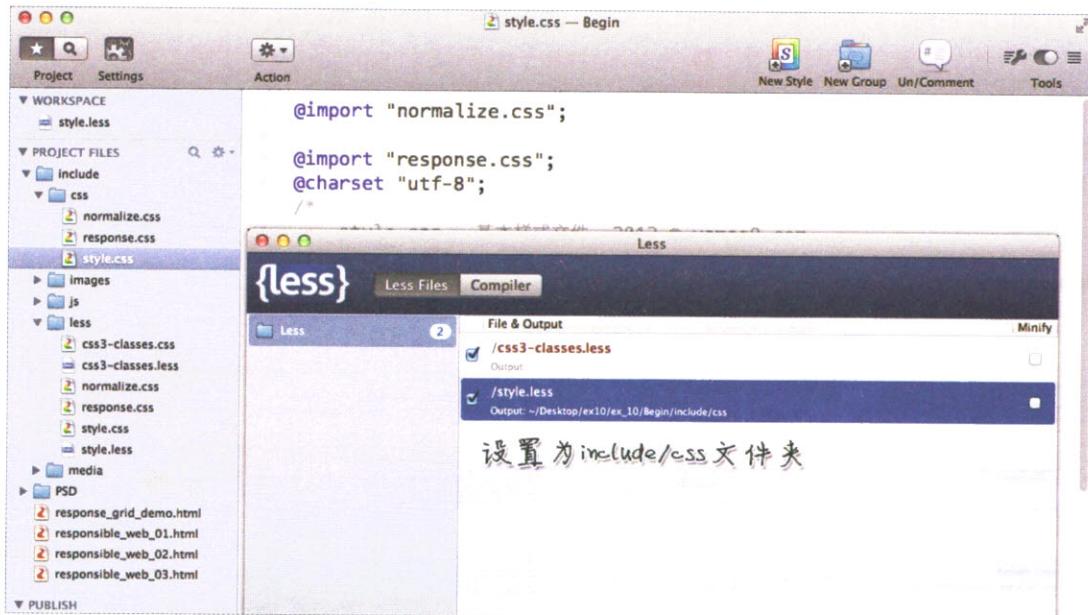
normalize.css 与 reset.css 一样，都能解决不同浏览器基本样式不同的问题。但是 reset.css 将所有的浏览器的自带样式重置掉，以保持各浏览器渲染的一致性。而 normalize.css 则会尽量保留浏览器的默认样式，只对引起问题的元素进行重置，此外不进行太多的重置。

在使用 LESS 编写样式代码之前，如果你是 Windows 用户，要先安装 SimpLESS.exe 程序；如果你是 Mac 用户，则要安装 Less.app 程序。关于它们的安装与使用方法，在前面我们已经详细地讲解过了。如果忘记了，请重新翻到第 6 章第 179 页学习。



在 Windows 环境下，首先运行 SimpLESS 程序，而后将整个 begin 文件夹拖放到 SimpLESS 程序中。当在 LESS 文件中编写代码并保存后，SimpLESS 会自动帮我们转换成 CSS 代码，添加到指定的 CSS 文档中。在向 SimpLESS 中拖放项目文件夹时，若发生错误，请参考第 6 章第 183 页中的相关内容进行解决。

如果你是 Mac 用户，请打开 Less.app 程序，设置输出文件夹。打开 style.less 文件后，在文件顶部插入 @import 语句，调用 normalize.css 文件，而后再添加一个调用 response.css 文件的命令。



编写 response.css 框架

下面我们开始编写 response.css 文件。在实例中它对网页样式的控制起着关键性的作用。在编写过程中，我们不会将文档区域的宽度设置为固定的数值（px 或 em），而是使用百分比数值（%），这样能够增强设计的弹性，同时我们会使用网格系统，根据设备的宽度，更改文档区域。

response.css 轻便、简单，且通过类关键字即能轻松地应用网格布局，使用方法也非常简单。下面让我们一起看一下各个类的作用。

为了应用网格布局，将关键字声明为类
.wrap, .center, .row, .col-1~.col-12, .last,
.fr名称设置类

```

Action
New Style New Group Un/Comment

.wrap {
    padding-left: 20px;
    padding-right: 20px;
}

.center { margin: 0 auto; }

.row {
    overflow: hidden;
    width: 100%;
    max-width: 1140px;
    min-width: 755px;
    margin: 0 auto 2%;
}

.col-1, .col-2, .col-3, .col-4, .col-5, .col-6, .col-7, .col-8, .col-9, .col-10, .col-11 {
    float: left;
    margin-right: 3.8%;
    min-height: 1px;
}

.row .col-1 { width: 4.85%; }
.row .col-2 { width: 13.45%; }
.row .col-3 { width: 22.05%; }
.row .col-4 { width: 30.75%; }
.row .col-5 { width: 39.45%; }
.row .col-6 { width: 48%; }
.row .col-7 { width: 56.75%; }
.row .col-8 { width: 65.4%; }
.row .col-9 { width: 74.05%; }
.row .col-10 { width: 82.7%; }
.row .col-11 { width: 91.35%; }
.row .col-12 { width: 100%; float: left; }

.last { margin-right: 0; }
[class^=col]:first-child { margin-top: 0; }

img, object, embed, iframe { max-width: 100%; }
img, iframe { height: auto; }

```

.wrap 类用来在元素的左侧与右侧设置 20px 的内部空间，并用于包裹要指定布局的元素。当元素设置了 width 后，.center 类用于将元素设置到页面中间。.row 类像其字面意思一样，它被应用到充当网格布局“行”的元素上，它会将被应用的元素置于页面中间，并在底部留出 2% 的空间，水平默认宽度为 100%，最小宽度为 755px，最大宽度为 1140px。它用于根据不同的设备动态地设置元素宽度的范围，更改最大宽度，可以使元素支持更大的显示屏。

```

.wrap {
    padding-left: 20px;
    padding-right: 20px;
}

.center { margin: 0 auto; }

.row {
    overflow: hidden;
    width: 100%;
    max-width: 1140px;
    min-width: 755px;
    margin: 0 auto 2%;
}

```

类.col-1~.col-12被应用于.row内部的元素上，后面的数字是列数。比如，一个可以容纳12个列的行，若其内部的一个元素应用了.col-4，该元素就拥有4个列的宽度，即.row全部宽度的1/3。在设置宽度时使用百分比，而不是px、em固定值，可以灵活地根据不同的设备随时调整宽度。所有持有.col类的元素都设置了float属性，即它们都是浮动的元素，并且它们的内部元素是按水平方向排列的。

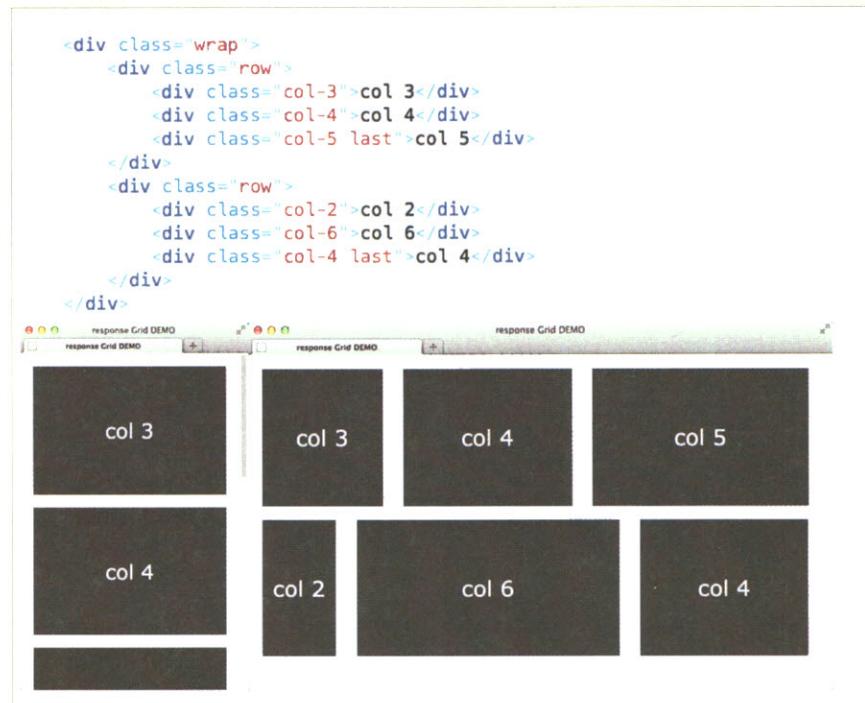
```
.col-1, .col-2, .col-3, .col-4, .col-5, .col-6, .col-7, .col-8, .co  
l-9, .col-10, .col-11 {  
    float: left;  
    margin-right: 3.8%;  
    min-height: 1px;  
}  
  
.row .col-1 { width: 4.85%;}  
.row .col-2 { width: 13.45%; }  
.row .col-3 { width: 22.05%; }  
.row .col-4 { width: 30.75%; }  
.row .col-5 { width: 39.45%; }  
.row .col-6 { width: 48%; }  
.row .col-7 { width: 56.75%; }  
.row .col-8 { width: 65.4%; }  
.row .col-9 { width: 74.05%; }  
.row .col-10 { width: 82.7%; }  
.row .col-11 { width: 91.35%; }  
.row .col-12 { width: 100%; float: left; }
```

.last类应用于最后一个拥有.col类的元素（该元素位于拥有.row类的元素内部）。若不向最后一个.col类的元素应用.last类，网页的布局结构就会被打乱，变得凌乱不堪。最后一个.fr类用于设置元素向右浮动。

```
.last { margin-right: 0; }  
[class^=col]>:first-child { margin-top: 0; }  
  
img, object, embed, iframe { max-width: 100%; }  
img, iframe { height: auto; }  
  
.fr { float: right; }
```

小知识 通过 Demo 理解 response.css 框架

关于 response.css 框架，虽然我们已经做了一些讲解，但是你对 response.css 框架可能还是一头雾水。下面我们通过一个 Demo 实例，跟大家一起进一步了解一下 response.css 框架的基本结构。进入 begin 文件夹，打开 response_grid_demo.html 文档，查看其中代码，可以很容易地了解它。当调整浏览器窗口的大小，减小水平宽度时，设计布局也就发生了变化，这是因为 float 属性被重新设置为 none 了，如图所示。



那么，已经设置的 float 属性满足什么条件才会被设置为 none 呢？答案存在于 CSS3 的 Media Queries 语句中。

```
/* 平板电脑: 1024 以下 */
@media handheld, only screen and (max-width: 1023px) { ... }

/* 智能手机: 767px 以下 */
@media handheld, only screen and (max-width: 767px) { ... }

/* iPhone 4 */
@media only screen and (-webkit-min-device-pixel-ratio: 2) { ... }
```

Media Queries 直译为“媒体查询”。CSS3 中的 Media Queries 提供了许多媒体查询，允许我们添加不同的媒体类型的表达式以检查媒体是否符合某些条件，若媒体符合相应条件，就调用对应的样式表，即在 CSS3 中我们可以设置不同类型的媒体条件，并根据对应的条件，给相应符合条件的媒体调用相对应的样式表。在上面的语句中，@media 用于在网页中引入媒体类型，handheld 是媒体类型（Media Type）之一，指的是手持设备，比如功能手机（Feature Phone，非智能手机）。

only 关键字的作用是让不支持 Media Queries 但能够读取 Media Type 的设备的浏览器将后面表达式中的样式隐藏起来。only screen 表示仅用于屏幕的样式文件被处理，即用于打印的样式文件不会被处理。and 关键字是“与”的意思，用于连接后面小括号内的语句，我们可以把它看作一个条件运算符。比如，A and B 表示条件 A 与条件 B 同时成立时进行处理，而 A or B 则表示条件 A 与条件 B 只要有一个成立就进行处理。

上面的代码都是用于画面输出而非打印输出的，并且设备的最大宽度要低于或等于设定的宽度。当这两个条件同时成立时，{} 内的代码就会被浏览器解析执行；若两个条件中有一个条件不成立，{} 内的代码就不会被浏览器处理执行。

这些代码是我们制作设备感应式网页的关键代码，其实也不是多么神秘吧？！接下来，我们要正式开始编写样式控制代码了。再次返回到 style.less 文件中，在其中添加如右图所示的 @media 语句，用来区分平板电脑、智能手机等设备。而后我们分别为桌面 PC、平板电脑和智能手机编写不同的样式控制代码。

```
/*
 * =====
 * =感应型网页设计(Responsible Web Design)
 * =====
 */

/* 桌面PC */
body { }

/* 平板电脑: 1024 以下 */
@media handheld, only screen and (max-width: 1023px) {
}

/* 智能手机: 767px 以下 */
@media handheld, only screen and (max-width: 767px) {
}

/* iPhone 4 */
@media only screen and (-webkit-min-device-pixel-ratio: 2) {
```

桌面显示样式 1：基本样式

首先为桌面 PC 显示编写样式。向 body 元素设置基本字体样式后，初始化 ul 与 ol 列表样式。normalize.css 本身带有列表在浏览器中显示的基本样式，这与 reset.css 有所不同。在我们的设计中，不需要使用列表的基本样式，所以我们将它们进行了初始化。然后使用 LESS 嵌套规则，分别设定文档结构的大片区域。

```
body {
    font: 14px/1.5 Georgia, serif;          设置正文字体、字号、行间距
    ul, ol {
        list-style: none;                    删除目录样式
        padding: 0;                         删除内部空间
    }
}
```

```

// header */
header { }

// #header-bar */
#header-bar { }

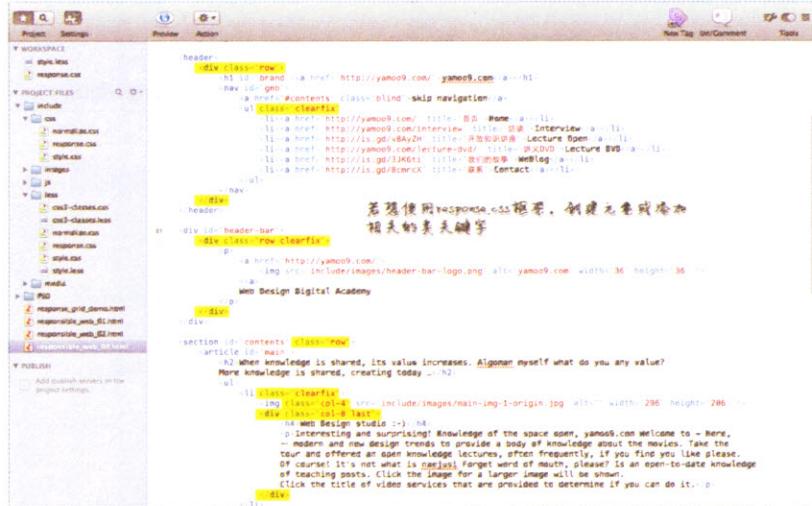
// #contents */
#contents { }

// footer */
footer { }

} // body

```

在设置样式之前，要先改变文档的结构。若想使用 response.css 网格系统布局，要先设置类 .row、.col-1~.col-12 和 .last 等。像前面我们讲到的一样，使用 response.css 时，要向布局应用正确的类关键字。在这里，我们将使用已经提前制作好的 responsive_web_03.html 文档进行讲解。查看 responsive_web_03.html 文档代码，可以看到哪些部分、哪些类被添加了。请与 responsive_web_02.html 文档进行对比。



桌面显示样式 2：header

下面我们开始为 body 的 header 区块编写样式代码。首先为 header 添加背景图片，指定图片是否重复，以及图片的位置。而后设置 CSS3 的 background-size 为 cover，保持图片本身的宽高比例，并将图片缩放到正好完全覆盖定义背景的区域。为了解决浏览器的兼容性问题，我们在 css3-classes.less 中定义了 .background-size 函数，使用时调用该函数即可。

```

header {
  background: url(..../images/header-bg.jpg) no-repeat 50% 55%;  
插入背景图片（路径、反复、位置）  
  .background-size(cover);  
}

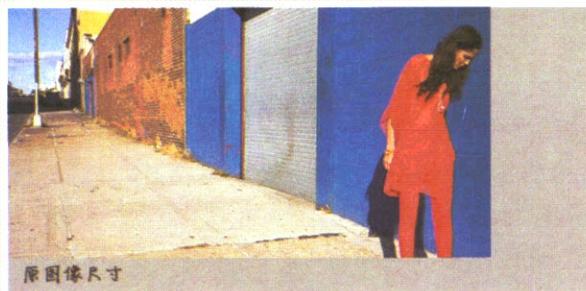
```



小知识 background-size 属性

background-size 是 CSS3 新增的一个属性，用于调整背景图片的尺寸，该属性提供两个参数（特性值 cover 和 contain 除外）。若提供两个，则第一个为背景图片的宽度，第二个为背景图片的高度；若只提供一个，则该值为背景图片的宽度，高度默认为 auto，此时背景图片以宽度为参考进行等比例缩放。属性值可以使用 px 或 % 等单位，也可以使用 auto、cover 和 contain 等关键字。

background-size: 宽度、高度；

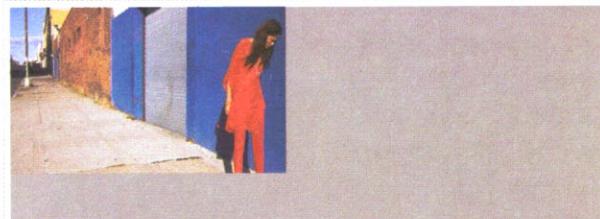


默认值为 auto，向元素背景插入图片后，图片保持原有尺寸，其大小不受元素大小的影响。



`background-size: auto 400px;`

如图所示，若设置 `background-size` 的第一个参数为 `auto`，第二个参数为 `400px`，则将背景的图片的高度设为 `400px`，宽度会以高度为参考进行等比例缩放。



`background-size: 450px 270px;`

如图所示，若设置背景图片的 `background-size` 为 `450px` 与 `270px`，背景图片的尺寸就会被强制设置为指定的值，这可能会使背景图片的宽高比例发生变化，使背景图片发生扭曲。



`background-size: cover;`

如图所示，若设置 `background-size` 为 `cover`，则将背景图片等比例缩放到完全覆盖容器元素，并且背景图片有可能超出容器元素。

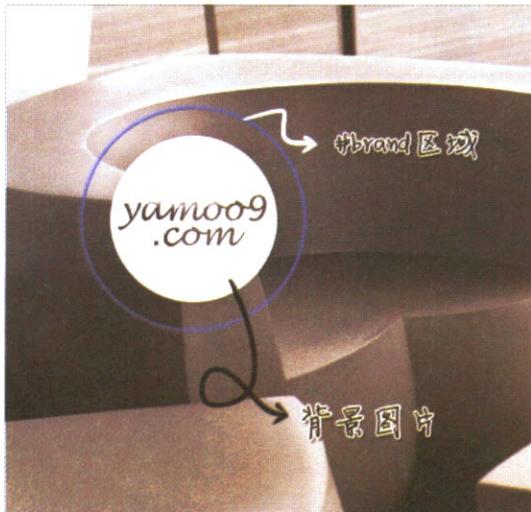


`background-size: contain;`

如图所示，若设置 `background-size` 为 `contain`，将背景图片等比缩放到宽度或高度与容器的宽度或高度相等，背景图片始终被包含在容器内。

接下来，为 header 内的 .row、#brand 和 #gnb 编写样式代码。由于 .header 内部的 .row 元素调用 response.css 中定义的类，因此它被设置到中间。#brand 元素是网站的 Logo 部分，设置其 position 为 relative，确定其位置，此时位置参考元素为 .row 元素。在 headerl 内的 .background-size(cover); 代码行下添加如下代码段。

```
.row {  
    position: relative;          // 设置为相对定位，作为内部绝对定位元素的基准元素  
    height: 500px;               // 设置高度为500px（显示背景图片的区域）  
    margin-bottom: 0;             // 设置下外边距（margin-bottom）为0  
    #brand {  
        overflow: hidden;         // 将溢出#brand区域的子元素从画面中剪裁并隐藏  
        position: relative;        // 设置为相对定位，作为内部绝对定位元素的基准元素  
        top: 94px;                 // 从当前位置下移94px  
        left: -22px;                // 从当前位置左移22px  
        margin-top: 0;              // <h1>设置上外边距为0，删除上方外部的空白  
        width: 194px;               // 设置#brand的宽度为194px  
        height: 194px;              // 设置#brand的高度为194px  
        background: url(..../images/header-brand-logo.png) no-repeat;  
                                // 设置网站图标图片  
        text-indent: -10000px;        // 设置margin为负值，或将包含的文本移动到画面之外隐藏起来  
        .border-radius(100px);       // 设置圆角边框效果，制作圆形边框（调用LESS函数）  
        .transition(all .4s ease);  // 设置持续0.4秒的transition效果（调用LESS函数）  
    } // e: #brand  
} // e: .row
```



保存代码，打开浏览器，查看页面，可以发现 #brand 部分被应用了 CSS3 的 border-radius 属性，变成了圆形，并且内部的背景图片也显现出来了。

接着，向 Logo 背景图片的上半部分添加红黄色的星形图标，制作简单的动画。在 .transition(all .4s ease); 代码行之后，添加如下代码段。

```

&:before {
    content: '';
    position: absolute;
    left: 50%;
    top: 50px;
    width: 17px;
    height: 17px;
    margin-left: -17px/2;
    background: url(..../images/header-brand-star.png) no-repeat;
        设置#brand:before伪元素
        设置内容为空字符串
        设置为绝对定位（基准元素：#brand）
        从左侧向右移动50%
        从基准元素向下移动50px
        设置宽度为17px
        设置高度为17px
        设置左外边距为负的宽度的一半
        设置星形图标
        设置持续0.6秒的transition效果（调用LESS函数）

    .transition(all .6s ease-out);
} // :before

&:hover {
    background-color: hsla(46, 0%, 0%, .1);
} // :hover

&:hover:before {
    top: 35px;
    .rotate(360deg);
        选择处于#brand:hover状态下的:before
        将top由50px更改为35px（处理transition效果）。
        沿顺时针方向旋转（调用LESS函数）

} // :hover:before

```

.rrotate(360deg) 是我们在 css3-classes.less 中定义的 LESS 函数，它用于解决 rotate() 函数在不同浏览器中的兼容性问题。在 CSS3 的 transform 属性中使用 rotate()，可以对指定的元素进行旋转，小括号内为要旋转的角度（degree）。需要注意的是在设置旋转角度时要在数值后面添加上 deg。

```

// 旋转
.rotate ( @deg ){
    -webkit-transform: rotate(@deg);
    -moz-transform: rotate(@deg);
    -ms-transform: rotate(@deg);
    transform: rotate(@deg);
}

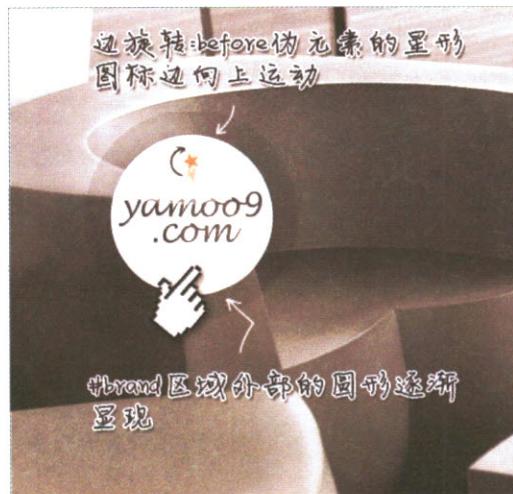
```

我们为 #brand 编写的所有样式代码如下所示，请认真检查，防止出现错误。

```

// #brand *
#brand {
    overflow: hidden;
    position: relative;
    top: 94px;
    left: -20px;
    margin-top: 0;
    width: 194px;
    height: 194px;
    background: url(..../images/header-brand-logo.png) no-repeat;
    text-indent: -1000px;
    border-radius(100px);
    .transition(all .4s ease);
    &:before {
        content: '';
        position: absolute;
        left: 50%;
        top: 50px;
        width: 17px;
        height: 17px;
        margin-left: -17px/2;
        background: url(..../images/header-brand-star.png) no-repeat;
        .transition(all .6s ease-out);
    } // :before
    &:hover {
        background-color: hsla(46, 0%, 0%, .1);
    } // :hover
    &:hover:before { // Chrome, Safari
        top: 35px;
        .rotate(360deg);
    } // :hover:before
} // e: #brand

```



在代码中我们使用 :before 伪元素来表示星形元素，在处于 #brand:hover 状态时，星形图标向上旋转形成动画效果，但是在使用非 Firefox 浏览器时，浏览器可能无法正常处理 #brand:hover:before，造成动画效果无法正常显示。若想在这些浏览器中制作出类似的动画效果，可以向 #brand 内部添加 元素来代替 :before 伪元素，而后应用设置在 :before 元素上的样式。

打开 responsive_web_03.html 文档，向

元素内部添加 元素，并为它设置类属性为 brand-star。

在 style.less 文件中，查找到 &:before 后，将其替换为 .brand-star 选择器，请参考右侧图片。

```
<div class="row">
  <h1 id="brand">
    <span class="brand-star"/>
    <a href="http://yamoo9.com/">yamoo9.com</a></h1>
  <nav id="gnb">
```

```
&:before {
  content: '';
  position: absolute;
  left: 50%;
  top: 50px;
  width: 17px;
  height: 17px;
  margin-left: -17px/2;
  background: url(..../images/header-brand-star.png) no-repeat;
  transition(all .6s ease-out);
} // :before

&:hover:before { // Chrome, Safari: 选择器问题...
  top: 35px;
  rotate(360deg);
} // :hover:before
```

```
.brand-star {
  position: absolute;
  left: 50%;
  top: 50px;
  width: 17px;
  height: 17px;
  margin-left: -17px/2;
  background: url(..../images/header-brand-star.png) no-repeat;
  transition(all .6s ease-out);
} // :before

&:hover .brand-star {
  top: 35px;
  rotate(360deg);
} // :hover .brand-star
```

更改为 .brand-star
选择器

在为网站的 Logo 编写完样式代码后，我们为导航区域 (#gnb) 编写样式代码。首先将 #gnb 设置为绝对定位，确定位置，而后为内部的 li 设置 float 属性，并设置右外边距为 30px。但最后一个 li 元素，我们要设置其右外边距为 0。当鼠标指针移动到 li 内的 a 元素上时，其颜色要平滑地进行切换，为此我们使用 CSS3 的 transition 属性。

```
#gnb {
  position: absolute;
  top: 35px;
  right: 0;
  ul {
    margin: 0;
    li {
      float: left;
```

设置为绝对定位，基准元素为 header .row
从基准元素向下移动35px
对齐到基准元素的右边
选择#gnb内部的ul
删除外部空白
选择#gnb ul内部的li
设置为向左浮动，使导航条沿水平排列

```

margin-right: 30px;          设置右外边距为30px
&:last-child {               选择#gnb ul li:last-child
    margin-right: 0;           设置右外边距为0（若不清除，会发生float drop现象）
}
a {
    font: 16px Georgia;      选择#gnb ul li内部的a
    color: #282020;           设置字号、字体
    text-decoration: none;     设置字体颜色为#282020
    text-shadow: 0 1px #b3b3b3; 删除超链接a的下划线
    .transition(all .3s ease-in-out); 设置持续0.3秒的transition（调用LESS函数）
    &:hover, &:focus {        当鼠标指针移到#gnb ul li a上或获得焦点时
        color: #fff;           逐渐改变文字色为白色
        text-shadow: none;      删除阴影效果
    } // :hover, :focus
} // e: a
} // e: li
} // e: ul
} // e: #gnb

```

保存代码，打开浏览器，查看网页，可以看到导航区域（#gnb）已经移动到了页面的右上端。当移动鼠标指针到相应的菜单项上时，其颜色也发生了变化。



桌面显示样式 3：header bar

在为 header 编写完样式代码后，我们开始为 # header-bar 编写样式代码。在 HTML 代码中，#header-bar 区域虽然位于 header 区域之后，但是当我们为它设置相对定位后，它就会重合到 header 的背景之上，就像它们是一个整体一样。

```

#header-bar {
    position: relative;       设置为相对定位，设定基准元素
    top: -45px;                从当前位置上移45px（与背景图片重叠）
    padding: 13px 0;            设置上下内边距为13px，使内部的P居中
    background: #272727;        当浏览器不支持CSS3 hsla()时设置替换背景颜色
    background: hsla(0, 0%, 15%, .4); 设置背景色为半透明黑色（不透明度为40%）
    .transition(all .4s);      设置持续0.4秒的transition（调用LESS函数）
    &.scrolled {               选择#header-bar .scrolled（发生滚动时进行处理）

```

```

background: hsla(0, 0%, 15%, 1);           逐渐更改背景色为黑色
.row p a {                                选择#header-bar .scrolled .row p内部的a
    visibility: visible;                   在画面上显示出来
    opacity: 1;                           设置不透明度为1 (100%)
    left: -45px;                         从基准元素的左侧向左移动45px
} // e: a
} // e: .scrolled
.row {                                       选择#header-bar内部的.row
    margin-bottom: 0;                     设置下外边距为0, 取消下外边距
    p {                                    选择#header-bar .row内部的p
        float: left;                      设置左浮动 (使文本进行环绕)
        position: relative;                设置为相对定位
        top: 2px;                          从当前位置下移2px
        margin: 0;                         删除外部空间
        font: 16px Georgia;               设置字号、字体
        letter-spacing: 0.3px;             设置字符间距为0.3px
        color: #ffff;                     设置字体颜色为白色
        &:after {                         生成p:after伪元素后编写样式
            .createStick();              调用短划线绘制函数 (另行编写该LESS函数)
            top: 50%;                   以p元素为基准将生成的短划线上下居中
            right: -44px;              从p元素的右边向右移动44px
        } // :after
        a {                                选择#header-bar p内部的a
            visibility: hidden;           在画面上隐藏 (将.scrolled类添加到#header-bar上即可更改)
            opacity: 0;                  设置不透明度为0 (将.scrolled类添加到#header-bar上即可更改)
            position: absolute;          设置为绝对定位, 依据基准元素进行移动
            top: -10px;                 从基准元素 (#header-bar p) 向上移动10px
            left: 0;                    对齐到基准元素 (#header-bar) 左边上
            .transition(all .4s);       设置持续时间为0.4秒的transition (调用LESS函数)
        } // e: a
    } // e: p
    ul {                                     选择#header-bar内部的ul (发生滚动时被添加)
        width: 560px;                      设置宽度为560px
        margin-left: 330px;                 设置左外边距为330px, 与p元素保持间隔
    } // e: ul
} // e: .row
} // e: #header-bar

```

在上面的代码中，我们在 p:after 伪元素中使用了名为 .createStick() 的 LESS 函数，该函数用来生成横线，后面我们还会用到它。在 style.less 文件的顶部，编写 .createStick() 函数代码，函数代码如下所示。

```
.createStick() {
    content: "";
    position: absolute;
}
```

```

width: 34px;
height: 1px;
background: #b2765b;
}

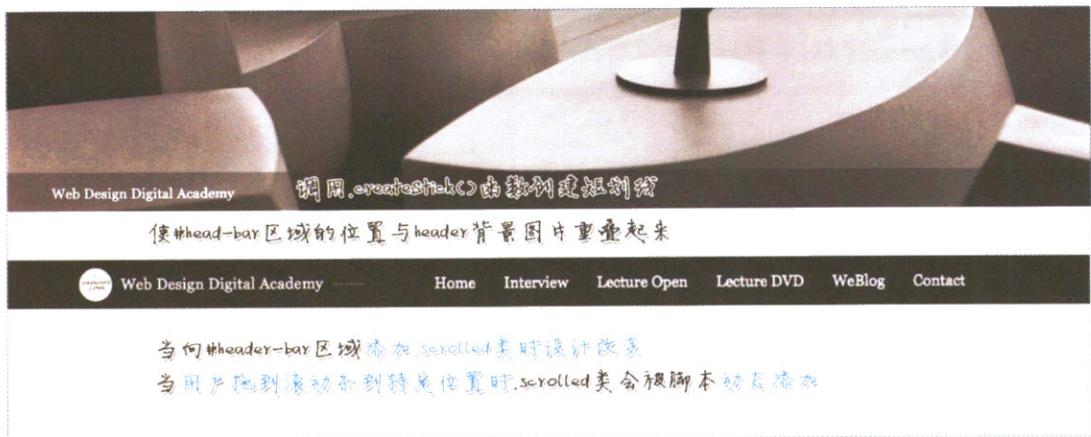
/*
 * =感应型网页设计(Responsible Web Design)
 */

// Less 函数: 创建短划线
.createStick() {
  content: '';
  position: absolute;
  width: 34px;
  height: 1px;
  background: #b2765b;
}

/* 桌面PC */

```

保存代码，在浏览器中打开网页，可以看到 `#head-bar` 已经重叠到 `header` 背景图片之上了。由于它是半透明的，因此它下面的背景也能被隐约地看到。



桌面显示样式 4: contents

下面我们开始为 `#contents` 编写样式代码。在为 `#contents` 编写样式代码的过程中，我们将感受到 `response.css` 框架的强大之处，因为在前面为 `header`、`#header-bar` 编写样式代码时，我们已经通过设置它们的 `position` 属性来确定了它们的位置。在 `#contents` 中，图像与文字段落是根据网格布局 (`float`, `%`) 进行设计的，这正是能够发挥 `response.css` 框架强大功能的地方。下面我们开始编写代码。

```

#contents {
  h2, h3 {
    position: relative;           // 设置为相对定位，作为短划线的基准元素
  }
}

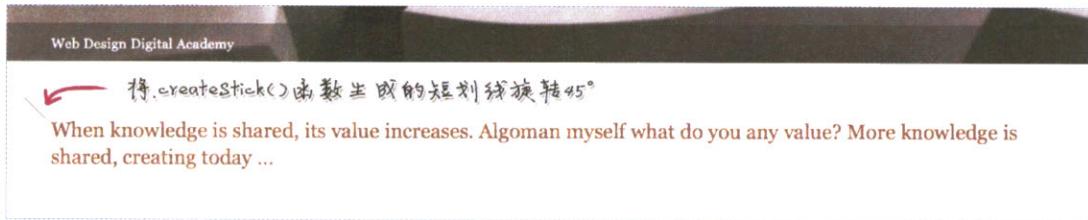
```

```

margin-bottom: 40px;          设置下外边距为40px，使其与标题明显区分开
border-bottom: 1px solid #e1dcda;    设置下外边距为40px，使其与标题明显区分开
padding-bottom: 20px;          在标题与下边框间设置20px空白
font: 24px/1.3 Georgia;      设置字号为24px，字体为Georgia，行距为1.3
color: #b2765b;              设置字体颜色
&:before {                  选择#contents h2:before, #contents h3:before
  .createStick();            调用生成短划线的函数（调用已编写好的LESS函数）
  top: -10px;                设置短划线的top值为-10px，基准元素为h2, h3
  left: -36px;               设置每划线的left值为-36px，基准元素为h2, h3
  .rotate(45deg);           旋转45度（调用LESS函数）
} // :before
} // e: h2, h3
} // #contents

```

保存代码，在浏览器中查看网页可以看到在标题的左上侧出现了横线，并且旋转了45度。即使不用图片，只用CSS3也同样能制作出图片。



接着，继续为li及其内部元素编写样式代码。在

与元素之后，编写如下代码。

```

#contents {
  h2, h3 {
    ...
  } // e: h2, h3
  li {
    margin-bottom: 60px;          选择#contents内部的li
    img {                      设置下外边距为60px
      border: 1px solid #e0e0e0;    选择#contents li内部的img
      .box-sizing();             设置边框粗细为1px，实线，淡灰色
    } // e: img
    h4, h5 {                  将盒状模型更改为border-box（调用LESS函数）
      font: 22px/24px "Helvetica Neue" Helvetica, Sans-Serif; 选择#contents li 内部的h4, h5
      ...
    } // e: h4, h5
    h5 {                      设置字号、行距、字体
      margin-top: 22px;          选择#contents li内部的h5
      margin-bottom: 32px;        设置上外边距为22px
    } // e: h5
}

```

```

p {
    font: 14px/24px "Helvetica Neue" Helvetica, Sans-Serif;
    color: #818181;
} // e: p
} // e: li
} // e: #contents

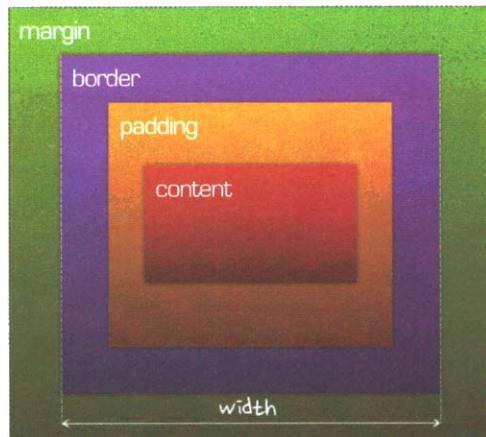
```

小知识 box-sizing 属性

box-sizing 是与 CSS 的盒子模型相关的一个属性，它有 content-box 与 border-box 两个取值。当将 box-sizing 设置为 content-box 时，元素的宽度 / 高度 (width/height) 等于元素的边框宽度 (border) 加上元素的内边距 (padding) 加上元素内容宽度 / 高度 (content width/height)；而当将 box-sizing 设置为 border-box 时，元素的宽度 / 高度等于元素内容的宽度 / 高度，这里的 content width/height 包含了元素的 border、padding、内容的 width/height。



设置为 content-box 时



设置为 border-box 时

为了方便在实例中使用该属性，我们在 `css3-classes.less` 文件中定义了 `.box-sizing()` 函数，如下所示。

```

// box-sizing
.box-sizing( @type : border-box ) {
    -webkit-box-sizing: @type;
    -moz-box-sizing: @type;
    box-sizing: @type;
}

```

保存代码，在浏览器中查看网页，可以看到标题、图片和文字段落都根据网格系统布局好了。只要把 `response.css` 中的类关键字添加到 HTML 元素上，相应的元素就被应用上了相应的样式。再次分析一下 `response_web_03.html` 文档，可以体会到网格系统的高效之处。使用添加类关键字的方法，后期的修改维护也变得非常简单。



Web Design studio :-)

Interesting and surprising! Knowledge of the space open, yahoo9.com Welcome to ~ Here, — modern and new design trends to provide a body of knowledge about the movies. Take the tour and offered an open knowledge lectures, often frequently, if you find you like please. Of course! It's not what is new! Forget word of mouth, please? Is an open-to-date knowledge of teaching posts. Click the image for a larger image will be shown. Click the title of video services that are provided to determine if you can do it.

Did the design and teaching how to start?

From the beginning was not that you wanted to design. Looked very much like flying cartoon. Of course! That's much loved comic now. Comics recently read "The Last Innings". As in baseball, it's a sports manga. Unlike the general director of the tactical aspects of baseball manga released in the thin spots do fresh. Excellent depiction of the psychological aspects hagoy. Mitsu Adachi's baseball manga, since this was the first freshness. Exciting future deployment will be made.



Please love me. You can do something, do not get what you want to continue. Do not let it, and the environment. "This is too ... Because of that 'rather than complain, people who can not change a good thing Be yourself.



Are now also work when you were a kid were you interested in such things?



Currently working as an instructor asks about hasineunde career so far.



The field continues to develop new technology is emerging technical knowledge in this field in any way for studying the world?

Knowledge was scattered all around. I can not find here ... My influences

come mainly from internet information, from web design, commercial, graphic

桌面显示样式 5: footer

在这部分，我们将为 footer 部分编写样式代码。footer 部分的设计并不复杂，在为 small 部分设置好样式后，在 footer 右侧添加上背景图片就可以了。

```
footer {
```

```
    border-top: 1px solid #e2e2e2;
```

```
    padding-top: 20px;
```

```
    p {
```

```
        padding-top: 40px;
```

```
        padding-bottom: 40px;
```

```
        background: url(..../images/footer-logo.png) no-repeat 100% 50%;
```

设置上边框为1px，实线，淡灰色

设置上内边距为20px

选择footer内部的

设置上内边距为40px

设置下内边距（底部空白）为40px

设置背景图片

```

small {
    display: block;
    line-height: 1.2;
    a {
        &:link, &:visited {
            color: #606060;
            text-decoration: none;
        } // :link, :visited
        &:hover, &:active {
            color: #212121;
        } // :hover, :active
    } // e: a
} // e: small
} // e: p
} // e: footer

```

保存代码，在浏览器中查看页面，可以看到我们为 footer 添加的样式。



桌面显示样式 6：设置拖选区域的背景色与文字颜色

在用户使用鼠标拖选相应的文字时，被拖选的区域会有背景色，并且被拖选的文字也会有颜色。下面我们使用 CSS3 ::selection 为这两个部分设置不同的颜色。为了解决浏览器的兼容性问题，我们在 css3-classes.less 文件中编写了一个名为 .selection() 的 LESS 函数。它有两个参数，第一个参数用于设置拖选区域的背景色，第二个参数用于设置拖选文字的颜色。在 style.less 文件中的 footer 之下添加如下一行代码。

```
.selection(#eee7dd, #542e1c);
```

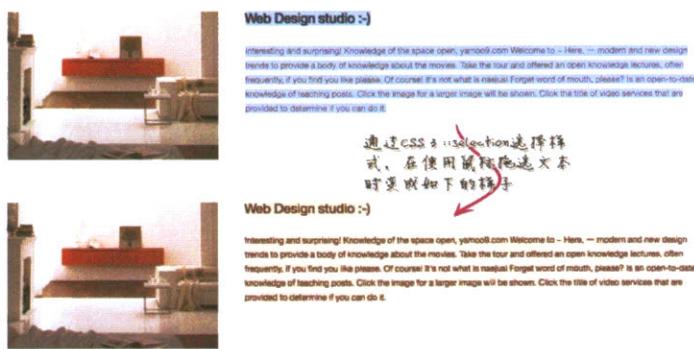
我们为 footer 与拖选颜色编写的代码整理如图。
请仔细检查自己编写的代码，防止出现错误。

```

// footer */
footer {
    border-top: 1px solid #e2e2e2;
    padding-top: 20px;
    p {
        padding-top: 40px;
        padding-bottom: 40px;
        background: url(../images/footer-logo.png) no-repeat 100% 50%;
        small {
            display: block;
            line-height: 1.2;
            a {
                &:link,
                &:visited {
                    color: #606060;
                    text-decoration: none;
                } // :link, :visited
                &:hover, &:active {
                    color: #212121;
                } // :hover, :active
            } // e: a
        } // e: small
    } // e: p
} // e: footer
} // body
// CSS3 selection
.selection(#eee7dd, #542e1c);

```

保存代码，在浏览器中打开网页，使用鼠标拖选某段文字后，可以发现被拖选区域的背景颜色以及被拖选文字的颜色都已经变成我们设置好的颜色了。



为平板电脑编写显示样式

前面我们已经为网页的桌面PC版编写好了样式代码，接下来，我们为网页的平板电脑版本编写样式代码，即在CSS3的Media Queries语句内编写代码，以使设计的网页根据设备的宽度做出调整。首先针对低于1024px尺寸的平板电脑编写网站样式。向网页左右添加空白时，只要通过设置网页的宽度与补丁值（Padding）即可实现，而且要保证它们的和为100%。

```
@media handheld, only screen and (max-width: 1023px) {  
    body {  
        header, #header-bar, #contents, footer {  
            width: 90%;  
            padding-left: 5%;  
            padding-right: 5%;  
        } // e: header, #header-bar, #contents, footer  
    } // e: body  
}
```



为智能手机编写显示样式

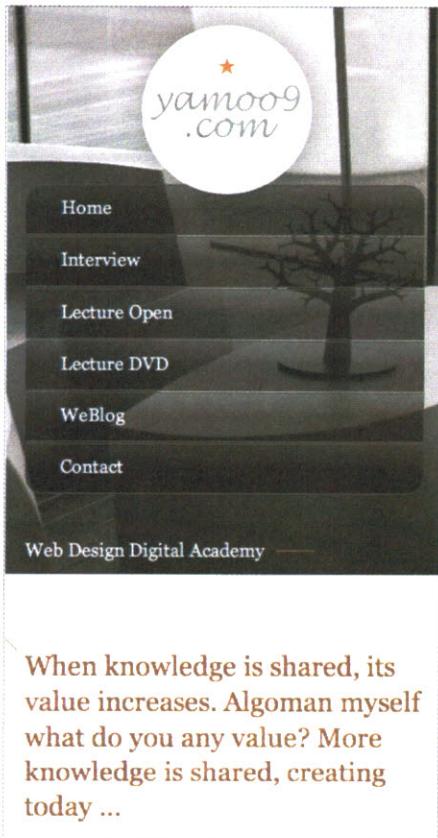
网页在智能手机中的显示情形类似于桌上PC。下面我们为智能手机编写显示样式。在低于767px的CSS3的Media Queries语句内编写如下代码。

```
@media handheld, only screen and (max-width: 767px) {  
    body header .row #brand {  
        top: 0;  
        left: 50%;  
        margin-left: -194px/2;  
    } // e: #brand  
    body header .row #gnb {  
        position: static;  
        margin-top: -50px;  
        ul li {  
            float: none;  
            margin-right: 0;  
            margin-bottom: 1px;  
            border-top: 1px solid #555;  
            border-bottom: 1px solid #555;  
            padding: 10px 30px;  
            .linear-gradient(hsla(0, 0%, 22%, .3), hsla(0, 0%, 0%, .5), 50%);  
            &:hover, &:active {  
                .linear-gradient(hsla(0, 0%, 22%, .5), hsla(0, 0%, 0%, .3), 50%);  
            } // :hover, active  
            &:first-child {  
                .border-radius(15px 15px 0 0);  
            } // :first-child  
            &:last-child {  
                .border-radius(0 0 15px 15px);  
            } // :last-child  
            a {  
                color: #fff;  
                text-shadow: none;  
            } // e: a  
        } // e: li  
    } // e: #gnb  
    #contents #main img, #contents #sub img {  
        width: 90%;  
        padding: 10px;  
        border: 1px solid #e0e0e0;  
        background: #fff;  
    } // e: #contents #main img, #contents #sub img
```

```

footer {
    padding-top: 0;
    p {
        padding-top: 20px;
        padding-bottom: 0;
        background: none;
        small {
            line-height: 1.5;
        } // e: small
    } // e: p
} // e: footer
}

```



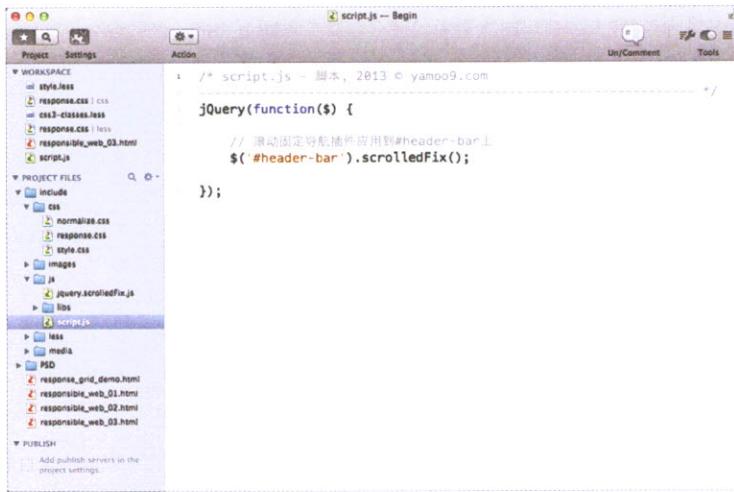
保存代码，打开浏览器，缩小浏览器窗口，使其尺寸与智能手机的屏幕尺寸相当，可以看到网页布局结构发生了明显的变化，如图所示。

到现在为止，我们已经为“设备感应式网页”编写好了 CSS 样式代码。接下来，我们要使用 JavaScript 脚本编写一个机动的导航菜单。当用户向下拖动浏览器右侧的滑动条，并且浏览器显示窗口触碰到导航菜单的顶端时，导航菜单就会被激活，并且固定在浏览器窗口的顶部。

使用 JavaScript 编写固定的导航菜单插件

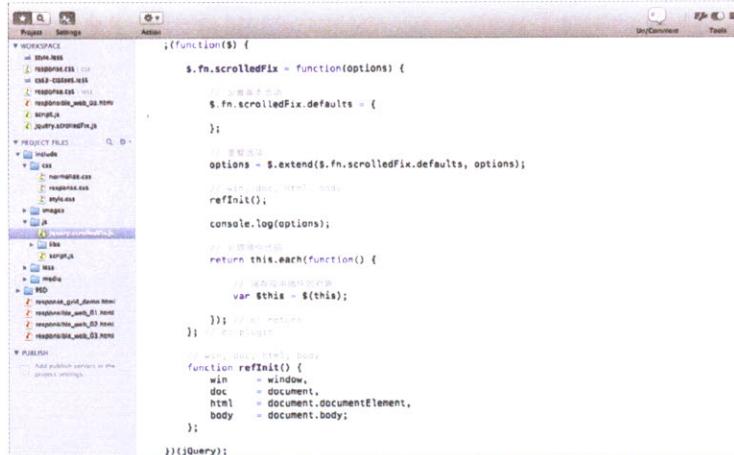
进入 include/js 文件夹中，打开 script.js 文件，添加如下代码。首先选择要固定的元素，而后调用 scrolledFix() 插件，保存代码。

```
$('#header-bar').scrolledFix();    // 查找#header-bar，应用scrolledFix插件
```



```
/* script.js - 脚本, 2013 © yamoo9.com */
jQuery(function($){
    // 滚动固定导航插件应用到#header-bar上
    $('#header-bar').scrolledFix();
});
```

在 include/js 文件夹中，打开 jquery.scrolledFix.js 文件，可以看到里面已经有了 jQuery 插件模板的代码。

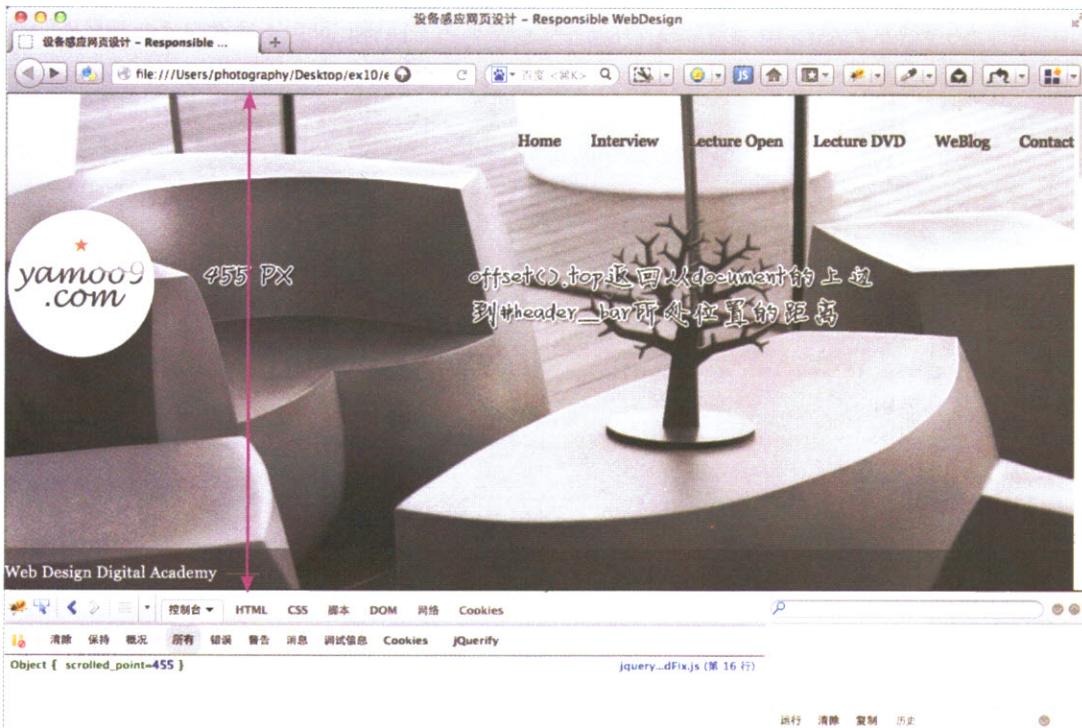


```
; (function($) {
    $.fn.scrolledFix = function(options) {
        // 滚动固定插件
        $.fn.scrolledFix.defaults = {
        };
        options = $.extend($.fn.scrolledFix.defaults, options);
        this.each(function() {
            var $this = $(this);
            console.log(options);
            // 固定逻辑
            return this.each(function() {
                // 滚动时滚动条位置
                var $this = $(this);
                // ...
            });
        });
    };
    // 重新初始化方法
    function refInit() {
        win = window;
        doc = document;
        html = document.documentElement;
        body = document.body;
    };
})(jQuery);
```

首先设置基本选项 \$.fn.scrolledFix.defaults。当滑动条高于某个地点时，导航菜单固定，充当插件的角色。这个特定的地点默认为应用插件的元素的位置，我们使用 jQuery offset().top 来获得位置数据，即返回元素相对于文档上边的距离。在 \$.fn.scrolledFix.defaults 中，添加如下代码。

```
$.fn.scrolledFix.defaults = {  
    scrolled_point : $(this).offset().top  
};
```

设置插件的初始值为defaults
设置scrolled_point为应用插件的元素与document(文档)的上边的距离



接下来，在 return this.each(function() {…}) 内部创建引用 window 的 \$win 对象。然后将 scrolledFix() 函数绑定到 \$win 上，当用户拖动浏览器的滑动条触发 load scroll 事件时调用该函数。

```
return this.each(function() {  
    //引用应用插件的对象  
    var $this = $(this),  
        $win = $(win);  
    //当文档加载时，滚动时，调用scrollFix函数  
    $win.bind('load scroll', scrolledFix);
```

使用\$0包装refInit()函数返回的win，而后保存到\$win变量中
绑定scrollFix函数到load scroll事件

紧接着，声明 scrolledFix() 函数。

```
function scrolledFix() {  
};
```

在 scrolledFix() 函数内部，首先判断 options.scrolled_point 的值是否为数字。当插件的使用者输入非数字的值时，就会引发错误，要终止继续执行函数。当用户拖动滑动条未达到指定的距离时，函数也不会执行。此外，在不支持 position:fixed; 的设备上也不执行函数。若用户设备上运行的是 iOS 系统，那么从 iOS 5.0 开始才支持 fixed; 若用户设备运行的是 Android 系统，那么从 Android 3.0 开始才支持 fixed。编写代码如下。

```
function scrolledFix() {  
    var UA = navigator.userAgent,  
        iOS5 = UA.match(/iPhone OS 5/i),  
        android3 = UA.match(/Android 3/i),  
        win_scrollTop = $win.scrollTop(),  
        point = options.scrolled_point;  
  
    if(  
        typeof point !== 'number' ||  
        win_scrollTop < point - $this.outerHeight()  
        && (!iOS5 || !android3)  
    ) return false;  
};
```

将 navigator.userAgent 值保存到 UA 变量中
检查 UA 中的字符串是否为 “iPhone OS 5”，将结果值保存在 iOS5 中
检查 UA 中的字符串是否为 “Android 3”，将结果值保存在 android3 中
将 \$win 滚动条的垂直位置保存到 win_scrollTop 中
将 options.scrolled_point 值保存到 point 变量中
若下列条件有一项为 true，则终止函数
当 point 的值不是数字，或者用户滚动的高度
低于 point - \$this.outerHeight()，或者 iOS 5
android3 的值为假时（即不是 iOS 5、Android 3 时），终止函数

接下来，我们使用 if…else 条件语句对 win_scrollTop 与 point 进行比较。当用户拖动滑动条的高度大于或等于 point 时，就固定应用插件的对象的位置；而当用户拖动滑动条的高度低于 point 时，就重新回到应用插件的对象的原来的位置上。

```
if(win_scrollTop >= point) {  
    // 固定应用插件的对象的位置。  
} else {  
    // 返回到应用插件的对象的原来位置。  
};
```

当用户拖动滑动条的高度大于或等于 point 时
当用户拖动滑动条的高度小于 point 时

在 else 语句中，我们先将当前设定的信息保存到 data() 中，这样才能正常返回。由于仅需要保存一次，因此要保存在函数的外部。

在 \$win=\$(win); 语句之下，添加如下代码。

```
$win = $(win);  
$this.data({  
    'pos': $this.css('position'),  
    'top': $this.css('top'),  
    'left': $this.css('left')  
});
```

将当前设置的信息保存到应用插件的对象的 data() 中
将对象的 position 当前设置值保存到 data('pos') 中
将对象的当前 top 值保存到 data('top') 中
将对象的当前的 left 值保存到 data('left') 中

```
//当文档加载时，滚动时，调用scrolledFix函数  
$win.bind('load scroll', scrolledFix);
```

保存代码，在浏览器中查看，可以清楚地看到设置 data() 与声明 scrolledFix() 函数的位置。

```
// 处理插件代码  
return this.each(function() {  
    // 保存应用插件的对象  
    var $this = $(this),  
        $win = $(win);  
  
    // 将应用插件的对象的当前值保存到data()中  
    $this.data({  
        'pos' : $this.css('position'),  
        'top' : $this.css('top'),  
        'left' : $this.css('left')  
    });  
  
    // 加载文档时，滚动滑条时调用scrolledFix函数  
    $win.bind('load scroll', scrolledFix);  
  
    // 碰撞到特定区域时停止导航的函数  
    function scrolledFix() {  
        // 判断浏览器及设备  
        var UA = navigator.userAgent,  
            iOSS = UA.match(/iPhone OS 5/ig),  
            android3 = UA.match(/Android 3/ig),  
            win_scrollTop = $win.scrollTop(),  
            point = options.scrolled_point;  
  
        // 检查options.scrolled_point值。当其低于特定点时不执行函数  
        if(  
            typeof point !== 'number' ||  
            win_scrollTop < point - $this.outerHeight() &&  
            (!iOSS || !android3)  
        ) return false;  
  
        if(win_scrollTop >= point) {  
        } else {  
        };  
    };  
});
```

在通过 data() 语句将应用插件的对象的信息保存之后，我们在 if…else 语句中编写相应的处理代码段，即当用户拖动滑动条，使其高度在 point 的位置之下时，应用插件的对象的位置就会被固定，否则回到原来的样式下。

```
if(win_scrollTop >= point) {  
    //固定应用插件的对象的位置。  
    $this  
        .css({  
            'position': 'fixed',  
            'z-index': 100,  
            'top': 0,  
            'left': 0,  
            'width': '100%'  
        })  
        .addClass('scrolled');  
} else {  
    //返回应用插件的对象的原来的位置。  
    $this  
        .css({
```

当用户拖动滑动条的高度大于或等于point时

使用应用插件的对象的css()方法

设置样式

设置position:fixed

设置z-index为100

设置top为0px

设置left为0px

设置width为100%

添加scrolled类

当用户拖动滑动条的高度小于point值时

使用应用插件的对象的css()方法

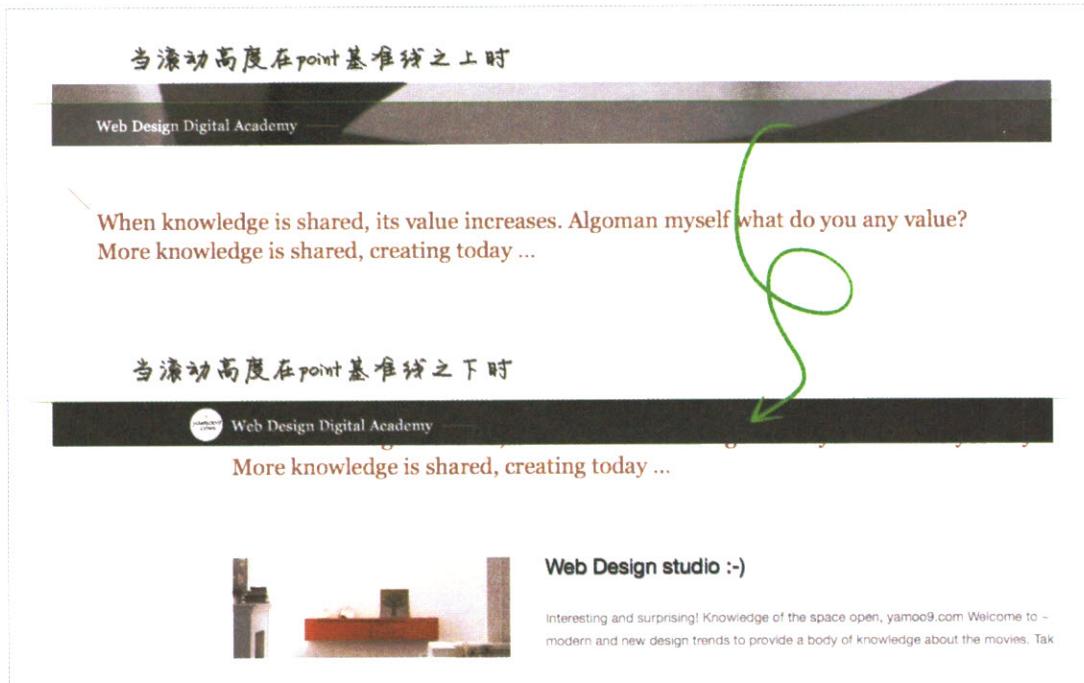
设置样式

```

    'position': $this.data('pos'),
    'top': $this.data('top'),
    'left': $this.data('left')
  })
  .removeClass('scrolled');
};


```

保存代码，打开浏览器，查看页面，向上或向下拖动浏览器右侧的滑动条，可以看到当滑动条高度高于或低于 point 的基准线时，应用插件的对象的设计就会发生变化。



接下来，我们为 scrolledFix 插件编写最后一部分代码。在这部分代码中，当滑动条的高度低于 point 基准线时，复制 header 内部的导航部分，添加到 #header-bar 的 div 中；不然，就从 #header-bar 的 div 中删除导航代码。首先声明一个 cloned 变量，该变量用于指示复制行为是否发生了，而后从 #gnb 中查找导航列表（ul），查找到后保存至 \$gnb_ul 变量中。

```

cloned = false;
$gnb_ul = $('#gnb').find('ul');


```

保存 false 值到 cloned 变量中
在 #gnb 内部查找 ul，包装成 jQuery 对象后，保存到 \$gnb_ul 变量中

```

// 保存应用插件的对象
var $this = $(this),
  $win = $(win),
  cloned = false,
  $gnb_ul = $('#gnb').find('ul');


```

返回到 if~else 语句中，添加如下代码。

```
if(win_scrollTop >= point) {  
    // 固定应用插件对象的位置。  
    ...  
    if( !cloned ) {  
        $gnb_ul  
        .clone()  
        .appendTo('#header-bar div');  
        cloned = true;  
    };  
} else {  
    // 返回应用插件的对象的原来的位置。  
    ...  
    if( cloned ) {  
        $('#header-bar ul').remove();  
        cloned = false;  
    };  
};
```

当用户拖动滑动条高度大于或等于point时
若cloned值为false
针对\$gnb
进行复制
添加到#header-bar内部的div内侧的后面
修改cloned变量的值为true（开关作用）

当用户拖动滑动条的高度低于point时
若cloned值为true时
查找复制粘贴的导航并删除它
修改cloned值为false（开关作用）

再次检查代码添加的位置，确保代码已经添加到正确的位置上，如下所示。

```
// 若未复制导航...  
if(!cloned) {  
    $gnb_ul  
    .clone()  
    .appendTo('#header-bar div');  
    cloned = true;  
};  
} else {  
    // 返回插件应用对象的样式  
    $this  
    .css({  
        'position': $this.data('pos'),  
        'top': $this.data('top'),  
        'left': $this.data('left')  
    })  
    .removeClass('scrolled');  
  
    // 若导航未复制...  
    if(cloned) {  
        $('#header-bar ul').remove();  
        cloned = false;  
    };  
};
```

保存代码，在浏览器中查看网页，观察代码是否正常工作。从浏览器显示的结果看，导航代码已经被复制了，但是并未应用上控制样式。



下面我们为它添加样式代码。再次打开 style.less 文件，向 #header-bar 的 ul 中添加样式代码，只要复制粘贴 header #gnb ul 的代码就可以了。如下所示，粘贴复制的代码，为 #header-bar 的 ul 添加上样式代码。

The screenshot shows the LESS editor interface with the file 'style.less' open. The left sidebar displays the project structure with files like 'style.less', 'response.css', 'css3-classes.less', and 'script.js'. The main workspace shows the LESS code for the header bar:

```
// e: p
ul { // 脚本动态生成的元素
    width: 560px;
    margin: 0;
    margin-left: 330px;
    li {
        float: left;
        margin-right: 30px;
        &:last-child {
            margin-right: 0;
        } // :last-child
        a {
            font: 16px Georgia;
            color: #fff;
            text-decoration: none;
            text-shadow: 0 1px #b3b3b3;
            .transition(all .3s ease-in-out);
            &:hover, &:focus {
                color: #b2765b;
                text-shadow: none;
            } // :hover, :focus
        } // e: a
    } // e: li
} // e: ul
} // e: .row
} // e: #header-bar
```

保存代码，在浏览器中查看页面，可以看到添加的样式代码已经起作用了，如图所示。



When knowledge is shared, its value increases. Algoman myself what do you any value?
More knowledge is shared, creating today ...



Web Design studio :-)

Interesting and surprising! Knowledge of the space open, yamoo9.com Welcome to ~ Here, — modern and new design trends to provide a body of knowledge about the movies. Take the tour and offered an open knowledge lectures, often frequently, if you find you like please. Of course! It's not what is naejusi Forget word of mouth, please? Is an open-to-date knowledge of teaching posts. Click the image for a larger image will be shown. Click the title of video services that are provided to determine if you can do it.

最后，感谢 Como Casa 公司 (<http://como-casa.com>) 提供的产品图片。Como Casa 公司是一家优秀的家具企业，拥有 Furniture & Home Deco 家居品牌，其设计的产品集功能性与审美性于一体，深受广大消费者的欢迎。