

# Web开发秘方

Web Development Recipes



[美] Brian P. Hogan,  
Chris Warren,  
Mike Weber,  
Chris Johnson,  
Aaron Godin 著

七印部落 译



华中科技大学出版社  
<http://www.hustp.com>

# 读者对本书的赞誉

---

## What Readers Are Saying About Web Development Recipes

《Web 开发秘方》精选 Web 开发实用技巧，方便学习 Web 设计和 Web 开发的读者快速掌握日常工作所需技巧，内容涵盖用户界面设计、测试方法、CSS、jQuery 等多个方面。全书行文言简意赅，尤其适合渴望学习新技巧的 Web 开发者阅读。

► **Peter Cooper**

Ruby Inside、HTML5 Weekly、JavaScript Weekly 网站编辑

我从未见过内容如此丰富的 Web 开发图书，这才是实实在在可以用到实际项目中的技巧。

► **Matt Margolis**

Getty Images 公司应用开发部经理

《Web 开发秘方》不仅实用，而且适用面广。凡是从事 Web 开发和 Web 设计工作的读者都能从书中找到解决实际问题的技巧和提示。

► **Ray Camden**

Adobe 公司技术培训师

## II ► 读者对本书的赞誉

这本书是我目前读过的最棒的 Web 开发工具书。进入这一行的新手如果单凭自己摸索，往往要花很长时间才能积累有效的经验。阅读本书可以在最短的时间内掌握这些技巧。即便是有经验的开发者，也能从中发现许多新技巧。

### ► **Steve Heffernan**

VideoJS 创始人

这本书堪称 Web 开发领域的设计模式，其中的解决方案几乎适用于所有的 Web 开发平台。这本书既适合新手学习，也可以作为有经验开发者的参考书。作者能把丰富的内容以简单易懂的形式展现出来，实属不易。

### ► **Derick Bailey**

Muted Solutions 公司独立软件开发者

# 目录

## Contents

致谢 .....	V
前言 .....	IX
第 1 章 养眼效果.....	1
1 号秘方 设计按钮和链接.....	2
2 号秘方 使用 CSS 设计评论 .....	6
3 号秘方 用 CSS3 变形技术创建动画.....	13
4 号秘方 用 jQuery 创建交互幻灯片.....	18
5 号秘方 设计创建行内帮助对话框.....	24
第 2 章 用户界面.....	33
6 号秘方 创建 HTML 格式的电子邮件模板 .....	34
7 号秘方 多 Tab 界面的内容切换.....	45
8 号秘方 可访问的展开和折叠.....	52
9 号秘方 使用快捷键与网页交互.....	59
10 号秘方 使用 Mustache 创建 HTML.....	67
11 号秘方 用无尽分页方式显示信息.....	73
12 号秘方 带状态的 Ajax.....	79
13 号秘方 通过 Knockout.js 使客户端交互更清爽 .....	84
14 号秘方 使用 Backbone.js 组织代码.....	93
第 3 章 数据处理.....	111
15 号秘方 嵌入一幅 Google 地图.....	112
16 号秘方 使用 Highcharts 创建图表和图形 .....	118
17 号秘方 创建简单的联系人表单.....	126
18 号秘方 利用 JSONP 访问跨网站数据 .....	134
19 号秘方 创建 Widget 嵌入其他站点 .....	138

20 号秘方	使用 JavaScript 和 CouchDB 建立带状态的网站 .....	144
<b>第 4 章</b>	<b>移动开发.....</b>	<b>153</b>
21 号秘方	面向移动设备的开发.....	154
22 号秘方	触摸响应式下拉菜单.....	159
23 号秘方	移动设备上的拖放.....	162
24 号秘方	利用 jQuery Mobile 创建用户界面 .....	169
25 号秘方	CSS Sprite 技术.....	178
<b>第 5 章</b>	<b>流程优化.....</b>	<b>183</b>
26 号秘方	使用栅格快速有效地进行设计 .....	184
27 号秘方	以 Jekyll 创建简单 Blog .....	193
28 号秘方	以 Sass 搭建模块化样式表.....	201
29 号秘方	以 CoffeeScript 清理 JavaScript.....	209
30 号秘方	以 Git 管理文件.....	216
<b>第 6 章</b>	<b>测试方法.....</b>	<b>227</b>
31 号秘方	调试 JavaScript.....	228
32 号秘方	用户点击热图分析.....	234
33 号秘方	使用 Selenium 测试浏览器.....	237
34 号秘方	Cucumber 驱动 Selenium 测试 .....	242
35 号秘方	Javascript 测试框架 Jasmine .....	255
<b>第 7 章</b>	<b>安装部署.....</b>	<b>267</b>
36 号秘方	使用 Dropbox 来托管静态网站.....	268
37 号秘方	建立虚拟机.....	272
38 号秘方	使用 Vim 修改 Web 服务器配置文件.....	277
39 号秘方	使用 SSL 和 HTTPS 来加强 Apache 安全.....	283
40 号秘方	保护你的内容.....	287
41 号秘方	URL 重写来保护链接.....	291
42 号秘方	使用 Jammit 和 Rake 自动化部署静态网站 .....	296
<b>附录</b>	<b>安装 Ruby .....</b>	<b>305</b>
<b>参考文献</b> .....		<b>309</b>
<b>索引</b> .....		<b>311</b>
<b>翻译审校名单</b> .....		<b>323</b>

# 致谢

---

## Acknowledgments

---

人常说光靠作者一个人是写不出书的，此话不假。虽然本书由五人合作完成，但是也少不了朋友们的帮助。没有大家的付出，就不会有这本书的出版和我们所获得的写作体验。

责任编辑 Susannah Pfalzer 不断与我们沟通，以保证书稿的语句完整、内容连贯、衔接自然。我们致力于展现最新的 Web 开发解决方案和工具，而 Susannah 则时刻提醒我们写清楚为什么要解决，以及如何解决问题，增强了书稿的可读性。

因为时间仓促，书中的错误和疏漏在所难免。好在我们有强大的技术审校团队：Charley Stran、Jessica Janiuk、Kevin Gisi、Matt Margolis、Eric Sorenson、Scott Andreas、Joel Andritsch、Lyle Johnson、Kim Shrier、Steve Heffernan、Noel Rappin、Sam Elliott、Derick Bailey 和 Kaitlin Johnson。感谢他们的无私付出。

特别感谢 Dave Gamache 针对 Skeleton 提出的宝贵建议，感谢 Trevor Burn 针对 CoffeeScript 提出的反馈意见，还有 Steve Sanderson 纠正了我们使用 Knockout.JS 的错误，以及 Benoit Chesneau 帮助我们解决了 Couchapp 的安装问题。

David Kelly 设计了封面。虽然我们也喜欢“熏肉 style”的那款设计，但总的来说大家对现在的封面都很满意。

衷心感谢 Pragmatic Bookshelf 出版公司的 Dave Thomas 和 Andy Hunt 提供了这次写作机会。他们不仅分享了宝贵的写作经验，更难得的是把作者放在最重要的位置。有了这样的信任和尊重，任何困难都变得微不足道了。

此外，还要感谢我们的业务合作伙伴 Erich Tesky、Austen Ott、Emma Smith、Jeff Holland 和 Nick LaMuro。感谢他们在出版过程中提供的帮助和反馈。

## Brian Hogan

虽然我已经在 Pragmatic Bookshelf 出版了两本书，但这次写作仍然充满挑战（尽管我只写了 1/5 的内容）。感谢本书的合著者们适时出现，Chris、CJ、Mike 和 Aaron 都贡献了精彩的内容，这本书值得我自豪，谢谢他们！

我还要感谢妻子 Carissa，她的付出保证了我有足够的时间写作。没有她的支持，我不可能完成这项任务。

## Chris Warren

感谢妻子 Kaitlin 的理解和支持，容忍我夜以继日，独自伏案写作。没有她，那些日子会变得更难熬。

感谢合著者的慷慨分享。大家都是老朋友了，第一次合作写书非常愉快。特别要感谢 Brian，在我学习编程的过程中，他对我的帮助最大，也是他邀请我参与写作的。

最后，感谢父母对我学习编程的鼓励和支持。写作的事我一直瞒着你们，只因我想给你们一个惊喜。

## Mike Weber

首先要感谢 Brian Hogan，他既是我进入 Web 开发领域的引路人，也是我多年来的老师。现在他又邀请我写书。没有他，这一切都不会发生。

感谢 Chris、CJ 和 Aaron，谢谢他们在写作过程中给予的支持，以及一直以来对我的帮助。

感谢我的家人，他们的督促帮助我能按时交稿。

最后感谢妻子 Kaley，为了安心写作，我许多晚上都不能陪她。

## Chris Johnson

感谢我妻子 Laura 在写作过程中给予的支持。为了这本书，她舍弃了许多本该与我共渡的美好时光，主动充当我的司机，还放弃了计划好的夏季活动。

感谢父母教会我为梦想努力，从不轻言放弃。爸爸，谢谢你，为了我写作，你推迟了自己的创业计划。

感谢 Brian、Chris、Mike 和 Aaron。你们的建议和鼓励让我受益匪浅，帮助我克服了一个个难关，衷心表示感谢。

感谢所有的审校者，谢谢你们出色的技术审校工作。

## Aaron Godin

Brian、Chris、Mike 和 CJ 激励着我，他们都是我学习的榜样。感谢大家对我的督促，尤其是在我想放弃的时候。特别要感谢 Brian，你是最好的良师益友。

感谢 Brian 对我的关心和耐心，感谢 Taylor 的鼓励和支持，你们是我克服困难的精神支柱。

最后，感谢父母对我无条件的关爱、理解和支持，并放手让我做自己想做的事。感谢你们赐予我智慧和勇气面对困难和挑战。





# 前言

---

## Preface

只会一点 HTML、CSS 和 JavaScript 已经远远不够用了，今天的 Web 开发者不仅要会编写可测试的代码、设计交互界面、集成各种服务，还要学会配置服务器（或者至少会一点后端知识）。本书精选新一代 Web 开发者需要掌握的开发技巧和解决方案，内容涵盖了从美化前端用户界面的 CSS 技巧到优化后端服务器的配置方法。

### 本书为谁而写？

#### Who's This Book For?

这本书是写给从事 Web 开发相关工作的读者看的。希望扩展业务技能的 Web 设计师和前端开发人员可以从中学学习新的开发工具的工作流程，从而提高工作效率，同时还能接触到一些实用的服务器端应用技巧。

从事服务器端开发工作的读者可以从中学会前端开发技巧，找到各种前端问题的解决方案，特别是提高测试效率和改善工作流程的技巧。

阅读本书最好具备基本的客户端编程知识，能够阅读 JavaScript 和 jQuery 代码。如果有些部分读不懂也没关系，你可以先使用书里的代码解决问题，回头再慢慢研究原理。

### 本书包含哪些内容？

#### What's in This Book?

本书收集了大量 Web 开发技巧，我们称之为秘方。每个秘方针对一个具体的问题，由问题描述和解决方案两部分组成。这些问题包括如何测试应用在不同浏览器下的表现，如何快速搭建简单的静态网站，如何从大量电子邮件中提

取联系人信息，如何配置 Apache 来重定向 URL 并提供更稳定的服务，等等。

我们尽可能解释每个秘方的工作原理，让读者知其然，也知其所以然，做到活学活用。由于篇幅有限，本书不可能讲解复杂的系统架构，每个秘方末尾的深入研究部分提供了进一步学习的相关资源。

虽然秘方根据不同的主题分成 7 章，但是读者不必受章节限制，可以自由挑选感兴趣的章节阅读。每章的内容都是由浅入深，最难的秘方通常放在最后。

第 1 章 养眼效果利用 CSS 和其他技巧美化 Web 应用的外观细节。

第 2 章 用户界面旨在提高用户界面的质量，用到了各种 JavaScript 框架（比如 Knockout 和 Backbone 等）。读者还将学习到优化模板的技巧，以提高发送 HTML 电子邮件的效率。

第 3 章 数据处理介绍各种处理数据的方法，比如建立简单的联系人清单，使用 CouchDB 中的 CouchApp 搭建数据库驱动的 Web 应用等。

第 4 章 移动开发讲解各种移动平台上的应用技巧，比如发挥 jQuery Mobile 的作用，处理多点触控事件，思考哪些应用适合为用户提供移动服务等。

第 5 章 流程优化介绍提高开发效率的技巧，比如，使用 SASS 大幅提高处理复杂样式表的效率，使用 CoffeeScript 编写更简洁、更易于管理的代码等。

第 6 章 测试方法讲解自动测试的技巧，读者将学会如何测试自己的 JavaScript 代码。

第 7 章 安装部署介绍了搭建虚拟机环境的方法，这样读者可以更安全地测试自己的 Web 应用。此外，读者还将学习提高网站安全性和稳定性的技巧，以及重定向的方法。此外，还介绍了自动部署网站的方法，降低遗漏上传文件的机率。

## 阅读前的准备

### What You Need

阅读前的准备本书包含了许多最新的 **Web** 开发技巧。虽然新技术的发展日新月异，但这些技术的基本原理已经稳定，值得推荐给大家。读者可以在本书的网站上下载相关库函数和示例的源代码。

虽然我们想方设法降低了阅读难度，但读者还是需要掌握一些基础知识才能更好地阅读本书。

## HTML5 和 jQuery

### HTML5 and jQuery

本书示例使用的都是 **HTML5** 风格的标记，比如，我们避免使用自封闭标签，但尝试运用了 `<header>` 和 `<section>` 之类的新标签。如果读者对 **HTML5** 不熟悉，建议先阅读 **HTML5** 的相关资料。

由于我们使用的库函数大多都是基于 **jQuery** 的，所以书中许多示例用到了 **jQuery**。我们使用的版本主要是 **Google** 的内容分发网络提供的 **jQuery 1.7**。有个别示例使用的是其他版本，遇到这种情况会另做说明。

## Shell

### The Shell

本书大量使用了命令行程序。一行简单的命令往往可以起到点击若干次鼠标的作用，因此使用命令行程序（特别是批处理命令行程序）可以大大提高工作效率。解释命令的工具统称为 **shell**，在 **Windows** 操作系统中，**shell** 是命令行（**command prompt**），在 **OS X** 和 **Linux** 操作系统中，**shell** 是命令终端（**the terminal**）。

下面是一行很常见的命令：

```
$ mkdir javascripts
```

这里的 `$` 是 **shell** 的提示符，不需要读者输入。虽然不同操作系统可能使用不同的提示符，但实际用法大同小异，相信读者很快就能适应。

## Ruby

有些秘方用到了 Ruby，还有些工具（如 Rake 和 Sass）必须安装 Ruby 才能使用。附录 1 介绍了安装 Ruby 的方法。

## QEDServer

有几个秘方用到了 QEDServer。QEDServer<sup>1</sup>是一个独立的、自带数据库的 Web 服务器程序，配置起来非常简单，而且可以跨平台工作，但需要事先安装 Java 运行时环境。书中提到的“开发服务器”如无特殊说明，指的就是 QEDServer。QEDServer 可以为 Web 应用提供一个相对稳定的测试环境，并且可以方便地处理本地的 Ajax 请求。

书中使用的 QEDServer 版本以及示例代码可以在本书的网站下载。

QEDServer 很容易启动。Windows 操作系统上的启动文件是 `server.bat`，如果是 OS X 或 Linux 操作系统，请运行 `./server.sh` 文件。运行后，系统会建立一个公共文件夹作为工作空间。只要在文件夹下创建一个 `index.html` 文件，就可以使用地址 `http://localhost:8080/index.html` 在浏览器里访问该页面。

## 虚拟机 A Virtual Machine

有几章用到了利用 Apache 和 PHP 搭建的基于 Linux 的 Web 服务器。37 号秘方讲解了安装该服务器和虚拟机的方法。另外，读者还可以在本书的网站上直接下载已经配置好的虚拟机。注意，运行虚拟机之前需要先安装 VirtualBox<sup>2</sup>。

## 在线资源 Online Resources

读者可以访问本书网站<sup>3</sup>下载所有的示例代码，以及 QEDServer 和配置好的虚拟机。

---

1 本书的版本可从 <http://webdevelopmentrecipes.com/> 得到。

2 <http://www.virtualbox.org/>

3 <http://pragprog.com/titles/wbdev/>

我们衷心希望《Web 开发秘方》能帮助您更好地完成下一个 Web 开发项目。

**Brian、Chris、CJ、Mike 和 Aaron**



# 第 1 章

---

## 养眼效果

## Eye-Candy Recipes

应用稳定固然很好，适当修饰的用户界面则是锦上添花。如果它们容易实现，那就更好了。本章将使用 CSS 来设计按钮和文本，然后用 CSS 和 JavaScript 来实现一些动画效果。



# 1 号秘方 设计按钮和链接

## Styling Buttons and Links

### 问题

#### Problem

在我们与网站的交互中，按钮是很重要的元素，因此，把按钮设计得与网站的风格相匹配很有必要。例如，有时我们希望提交按钮和取消表单链接的外观看上去更协调，但其前提是不必为每个元素单独制作图片。

### 要素

#### Ingredients

符合 CSS3 标准的浏览器，如 Firefox 4, Safari 5, Google Chrome 5, Opera 10, 或 Internet Explorer 9。

### 解决方案

#### Solution

通过使用样式和一些 CSS 规则，我们能创建一个样式表，让链接和按钮拥有一致的外观，比如看上去都像是按钮。我们从创建包含一个链接和一个按钮的简单 HTML 页面开始。

cssbuttons/index.html

```
<p>
  <input type="button" value="A Button!" class="button" />
  <a href="http://pragprog.com" class="button">A Link!</a>
</p>
```

注意，两个元素都指定了 `button` 样式。我们把这个样式同时用在链接和按钮上，这样它们就具有相似的外观了。

当我们建立 `button` 样式的时候，大多数属性是同时应用于链接和按钮的，然而，还有一小部分属性是为了使两者一致而进行微调用的。

我们先给两者加上基本的 CSS 属性。

```
cssbuttons/css-buttons.css
font-weight: bold;
background-color: #A69520;
text-transform: uppercase;
font-family: verdana;
border: 1px solid #282727;
```

结果看起来像是这样：



两个元素看上去大致相似，但是离我们的目标还差得很远。字体大小和间距都不一样，很容易就能分辨出来。

```
font-size: 1.2em;
line-height: 1.25em;
padding: 6px 20px;
```



通过设置样式的字体大小、行高和间距，我们覆盖了已经设置在链接和按钮上的样式。

但还是有一些不一致的地方要处理。

```
cursor: pointer;
color: #000;
text-decoration: none;
```

默认的连接会使光标从箭头变为手形，但按钮不会。此外，链接有默认的颜色而且有下划线。

在浏览器里放大页面，你会发现尽管两者的高度非常地接近，但链接还是会稍微小一点。这种差异在移动设备上使用放大功能的时候会更加明显，所以我们还得设法调整。

```
input.button {
  line-height: 1.22em;
}
```

用 `button` 样式给按钮元素设置一个稍大一点的行高，使它与链接看起来高度一致。这里没有简单的方法找出需要的高度，只有在浏览器里放大它们并调整行高直到按钮看起来一样高。

消除了最后的差异，然后我们再优化整体外观，比如使用圆角和添加阴影效果，就像这样：

```
border-radius: 12px;  
-webkit-border-radius: 12px;  
-moz-border-radius: 12px;  
  
box-shadow: 1px 3px 5px #555;  
-moz-box-shadow: 1px 3px 5px #555;  
-webkit-box-shadow: 1px 3px 5px #555;
```



我们给半径和阴影属性各添加三行代码来确保这个效果兼容尽可能多的浏览器。对于支持 `CSS3` 的浏览器来说，每组的第一行代码就足够了，但是 `-webkit-*` 和 `-moz-*` 可以提高对低版本 `Safari` 和 `Firefox` 的兼容性。

接下来我们给按钮的背景添加渐变效果。稍后设置按钮按下效果时，也用得着它。

```
background: -webkit-gradient(linear, 0 0, 0 100%, from(#FFF089), to(#A69520));  
background: -moz-linear-gradient(#FFF089, #A69520);  
background: -o-linear-gradient(#FFF089, #A69520);  
background: linear-gradient(top center, #FFF089, #A69520);
```



同样，我们用多行代码来兼容不同的浏览器，并给按钮的背景创建了渐变效果。注意，`-o-*`是为了支持 Opera 浏览器，这在最后一组 CSS 属性中不是必须的<sup>1</sup>。

最后，我们要添加样式来处理点击事件，这样当按钮被按下时就会出现相应的变化。否则会使用户感到困惑。有很多方法来表现按钮被按下，最简单的办法是反转渐变效果。

```
.button:active, .button:focus {  
  color: #000;  
  background: -webkit-gradient(linear, 0 0, 100% 0,  
    from(#A69520), to(#FFF089));  
  background: -moz-linear-gradient(#A69520, #FFF089);  
  background: -o-linear-gradient(#A69520, #FFF089);  
  background: linear-gradient(left center, #A69520, #FFF089);  
}
```

有很多方法能够反转渐变效果，最简单的方法是交换渐变两端的颜色。通过在 `.button:active` 和 `.button:focus` 上设置这个背景色，我们可以确保不论是链接还是按钮被点击时，都会发生变化。

通过 CSS 样式来控制链接和按钮的外观，让我们能用最合适的方式设计并使用它们，像是页面间的跳转链接或者提交数据的按钮。既能保持界面的一致性，又不需要依靠 JavaScript，用链接来提交表单或是点击按钮跳转到其他页面。这样不仅避免了老版本浏览器的兼容问题，也能更容易地理解页面的工作原理。

## 深入研究

### Further Exploration

如果你不想让用户点击某个按钮，你可以把它从页面上删掉，或者给它添加一个 `disabled` 样式。这时它看起来会是什么样的呢？有了合适的不可用按钮样式之后，怎样才能真正禁用这个按钮呢？Input 按钮有一个 `disabled` 属性，如果是链接的话，你就需要用到 JavaScript 了。

## 另请参考

### Also See

- 2 号秘方 使用 CSS 设计评论
- 28 号秘方 以 Sass 搭建模块化样式表

---

<sup>1</sup> 为了帮助你正确地设置渐变，请查看 <http://www.westciv.com/tools/gradients/>。

## 2 号秘方 使用 CSS 设计评论 Styling Quotes with CSS

### 问题

#### Problem

专家的引文和用户的称赞非常重要，所以我们经常促使别人去注意它们。我们会留出一些边距、增大字体或者使用大大的弯曲的引号来突出这些引文。在网站上，我们会使用简单且可重复的方式，让引文的内容和代码能够区分开来。

### 工具

#### Ingredients

支持 HTML5 和 CSS3 的浏览器

### 解决方案

#### Solution

通常我们使用 CSS 来区分介绍和内容，设计引文也不例外。新的浏览器支持一些更高级的属性，不用在页面上添加额外的标记，我们就能突出我们的引文内容。

在 2 号秘方中，我们关注的是设计引文的样式，但所讨论的技巧也可以应用于其他情况。例如，把我们要写的 CSS 与 7 号秘方（第 45 页）中的代码结合在一起，我们就可以进一步自定义不同例子中的样式，通过修改颜色区分不同的数据集。我们还可以应用于 25 号秘方（第 178 页），给我们的引用或例子添加背景图片。

我们要给某产品页面添加一些客户的评论。它们往往只有几句话，但是每个产品页面有多个引用，而且我们希望它们能从产品描述中突显出来。首先介绍实现这个需求会用到的 HTML 和 CSS 技术。

我们将从建立 HTML 结构开始，为 CSS 搭建一个基础框架。使用 `<blockquote>` 和 `<cite>` 标签分别包住评论和来源。

```
cssquotes/quote.html
```

```
<html>
  <head>
    <link rel="stylesheet" href="basic.css">
  </head>
  <body>
    <blockquote>
      <p>
        Determine that the thing can and shall be done,
        and then we shall find the way.
      </p>
    </blockquote>
    <cite>Abraham Lincoln</cite>
  </body>
</html>
```

该引用有很好的语义标记，下面设计它们的样式。先用一个简单的方法给引用添加一个边框线，增大字号，同时突出作者的名字并使之右对齐，如图 1 所示。

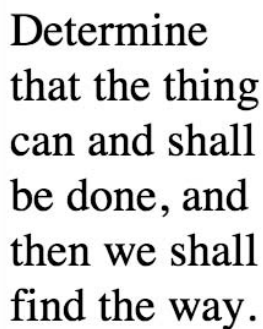
```
cssquotes/basic.css
```

```
blockquote {
  width: 225px;
  padding: 5px;
  border: 1px solid black;
}

blockquote p {
  font-size: 2.4em;
  margin: 5px;
}

blockquote + cite {
  font-size: 1.2em;
  color: #AAA;
  text-align: right;
  display: block;
  width: 225px;
  padding: 0 50px;
}
```

在该基础样式中，我们给主要元素 `<blockquote>` 和 `<cite>` 设置了一样的宽度。我们在 `<cite>` 标签上使用了相邻节点选择器，确保只更改了紧随 `<blockquote>` 标签出现的 `<cite>` 标签样式，而不去改动其他的 `<cite>` 标签。除此之外，我们改变了作者名字的颜色，并调整了文字间距，最终得到了简单美观的引文。



Determine  
that the thing  
can and shall  
be done, and  
then we shall  
find the way.

*Abraham Lincoln*

图1 基础引文样式

我们已经建立了引文的基础样式，接下来我们会做得更花哨一些。相比于使用边框，这次我们在引文上添加一个大大的引号“ ”来吸引眼球并让它在内容中突显出来，如图2所示。

```
cssquotes/quotation-marks.css
blockquote {
  width: 225px;
  padding: 5px;
}

blockquote p {
  font-size: 2.4em;
  margin: 5px;
  z-index: 10;
  position: relative;
}

blockquote + cite {
  font-size: 1.2em;
  color: #AAA;
  text-align: right;
  display: block;
  width: 225px;
  padding: 0 50px;
}

blockquote:before {
  content: open-quote;
  position: absolute;
  z-index: 1;
  top: -30px;
```

Determine  
that the thing  
can and shall  
be done, and  
then we shall  
find the way.

-- Abraham Lincoln

---

图 2 添加 CSS 后的引文样式

```
left: 10px;
font-size: 12em;
color: #FAA;
font-family: serif;
}

blockquote:after {
  content: close-quote;
  position: absolute;
  z-index: 1;
  bottom: 80px;
  left: 225px;
  font-size: 12em;
  color: #FAA;
  font-family: serif;
}
blockquote + cite:before {
  content: "-- ";
}
```

上面的样式在文字内容后面插入了引号，在作者的名字前面添加了破折号，并且去除了黑色的边框。

为了实现这个效果，我们用到了 `:before` 和 `:after` 选择器，在页面上遇到指定的标签时就能插入相应内容。使用 `content` 属性可以指定 `content` 的内容，比如开合的引号或者字符串。



除了适当的引文，我们还添加了一些属性，大多数看名字都能明白，比如 color, font family 和 font size。特别要注意的是 z-index 属性，以及 blockquote p 标签上的 position:relative; 属性。使用 position 属性和 z-index 属性，可以把引号放在引文的下面，不需要额外的空间，而且文字覆盖在引号上看起来很酷。同样，我们可以把 blockquote:after 放在底部，这样无论引文有多长，引号总是会显示在最后。

最后一个样式，我们可以不遗余力地将引文做得像说话泡泡，借助 CSS3 的超酷属性打造圆角和渐变的背景色，使引文看起来如图 3 所示。

```
cssquotes/speech-bubble.css
blockquote {
  width: 225px;
  padding: 15px 30px;
  margin: 0;
  position: relative;
  background: #faa;
  background: -webkit-gradient(linear, 0 0, 20% 100%,
    from(#C40606), to(#faa));
  background: -moz-linear-gradient(#C40606, #faa);
  background: -o-linear-gradient(#C40606, #faa);
  background: linear-gradient(#C40606, #faa);
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  border-radius: 20px;
}

blockquote p {
  font-size: 1.8em;
  margin: 5px;
  z-index: 10;
  position: relative;
}

blockquote + cite {
  font-size: 1.1em;
  display: block;
  margin: 1em 0 0 4em;
}

blockquote:after {
  content: "";
  position: absolute;
  z-index: 1;
  bottom: -50px;
  left: 40px;
  border-width: 0 15px 50px 0px;
  border-style: solid;
```

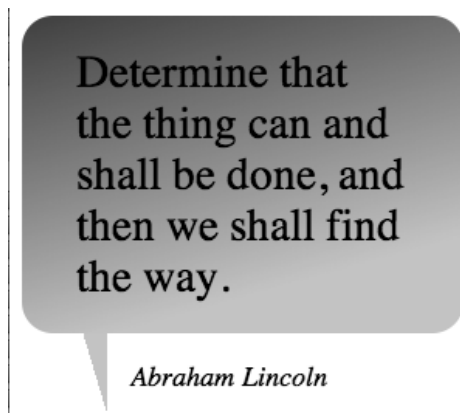


图 3 CSS3 式样说话泡泡

```
border-color: transparent #faa;
display: block;
width: 0;
}
```

使用 CSS3，我们不需要图片就可以把引文放在说话泡泡里。我们给 `blockquote` 设置了背景颜色，即使在不支持 CSS3 效果的浏览器里也能显示。接下来，我们用 `linear-gradient` 属性设置了渐变的背景，再用 `border-radius` 属性给元素添加了圆角。

不同浏览器对于 `linear-gradient` 和 `border-radius` 的语法不同，我们需要使用多行代码来得到相同或相似的效果。`-moz` 和 `-webkit` 前缀分别表示对应于 Firefox 和基于 WebKit 的浏览器（如 Safari 和 Chrome）。最后我们添加 CSS3 的标准属性，覆盖所有的基础部分。

`blockquote p` 和 `blockquote + cite` 的样式做了一些微调，少数属性的大小也稍作调整，但总体上还是一致的。字体颜色、大小、间距较易调整，使之与网站样式更加匹配。

最后我们设计 `blockquote:after` 元素的样式，给说话泡泡的底部加一个三角形。元素内容是空字符串，因为这里并不需要实际内容；我们只需要它的边框就可以了。通过给上下左右的边框设置不同的粗细，就做好了一个三角形。任何 CSS 属性的每一边都可以按照上、右、下、左（顺时针）的顺序设置不同的值。我们用这个方法来说设置边框的粗细和颜色，包括上下的透明边框线

和左右边框线的颜色。

## 深入研究

### Further Exploration

你还能想出引文的什么样式呢？在最后一个例子中，我们设计了一个说话泡泡。把 `blockquote:after` 样式的边框从右换成左，可以使它垂直翻转，但是如果要把作者的姓名和三角形移到泡泡的上方，我们又该怎么做呢？

IE 的渐变过滤器也可以做出与我们之前最后一个引文样式相同的效果，但是方法有些差别。IE 的渐变是直接应用于对象的，而不是像其他浏览器那样在背景图片上渐变。根据微软的文档，你能够使这个效果也支持老版本的 IE 吗？<sup>2</sup>

## 另请参考

### Also See

- 1 号秘方 设计按钮和链接
- 25 号秘方 CSS Sprites 技术
- 7 号秘方 多 Tab 界面的内容切换
- 28 号秘方 以 Sass 搭建模块化样式表

---

<sup>2</sup> <http://msdn.microsoft.com/en-us/library/ms532997.aspx>

## 3 号秘方 用 CSS3 变形技术创建动画

### Creating Animations with CSS3 Transformations

#### 问题

##### Problem

对于许多 Web 开发者来说, Flash 是制作网站动画的首选工具, 但是在不支持 Flash 的设备(如 iPad、iPhone)上就无法实现。如果动画对客户非常重要, 那我们就必须找到一个不使用 Flash 的解决方案。

#### 工具

##### Ingredients

- CSS3
- jQuery

#### 解决方案

##### Solution

随着 CSS3 变换和变形技术的出现, 我们可以不必再使用 Flash 这样的插件, 而做出原生的动画。这种动画只能在较新的移动设备浏览器和最新版本的 Firefox、Chrome、Safari 和 Opera 上显示, 不过即使用户看不到动画, 还是可以看到 logo 的。想要让动画在其他浏览器也可见, 我们还是得依靠 Flash。

客户当前的网站有一个 Flash 做的 logo, 用户打开页面时, 可以看到 logo 上闪过的发光效果。他拿来了一个新版的 iPad, 失望地发现动画无法显示, 但他更担心 logo 也不会正常显示。丢失的效果不会破坏网站的整体感觉, 但 logo 也看不到就会影响网站的品牌了。我们要使 logo 在所有的浏览器上都可见, 然后再为支持 CSS3 变形技术的浏览器加上动画效果。

让我们从包含 logo 的 header 标签开始。我们会给<img>标签添加一个样式, 稍后在样式表中会使用到。

csssheen/index.html

```
<header>
  <div class="sheen"></div>
  
</header>
```

为了实现这个效果，我们要创建一个半透明、倾斜且边缘雾化的 HTML 块，当页面文档对象模型（DOM）加载完成时，它将从屏幕上滑过。首先从定义 header 部分的基础样式开始，我们需要一个通栏的蓝色 banner 置于页面顶部。因此需要指定头部的宽度，然后在左上角放置我们的 logo。

```
csssheen/style.css
body {
  background: #CCC;
  margin: 0;
}
header {
  background: #436999;
  margin: 0 auto;
  width: 800px;
  height: 150px;
  display: block;
  position: relative;
}
header img.logo {
  float: left;
  padding: 10px;
  height: 130px;
}
```

有了合适的基础布局，就可以添加动画的装饰元素了。首先是创建边缘雾化的 HTML 元素，这些额外的效果几乎没什么实际用途，我们希望使用尽可能少的 HTML 标记，而通过之前定义好的带有闪光样式的<div>来实现。

```
csssheen/style.css
header .sheen {
  height: 200px;
  width: 15px;
  background: rgba(255, 255, 255, 0.5);
  float: left;
  -moz-transform: rotate(20deg);
  -webkit-transform: rotate(20deg);
  -o-transform: rotate(20deg);
  position: absolute;
  left: -100px;
  top: -25px;
  -moz-box-shadow: 0 0 20px #FFF;
  -webkit-box-shadow: 0 0 20px #FFF;
  box-shadow: 0 0 20px #FFF;
}
```



图 4 光泽可以看见但是还没有样式

如图 4 所示，我们添加了一根细白的透明线条，其长度比我们的头部稍微高。这是一个好的开始，接下来我们要重新放置这个线条，使它模糊一些，略微倾斜，并从头部的左边开始移动。

下面的操作需要一点小技巧。因为浏览器对于变形和变换的支持都不相同，我们需要添加特定的前缀来确保每个浏览器都能理解这个样式的改变。所以我们至少要给每个样式声明相同的参数，添加各种前缀，以确保每个浏览器都能应用这个样式。我们还要添加一个没有前缀的样式定义，在支持 CSS3 的时候正常展示。你可以看到，我们没有声明 `-o-box-shadow` 样式，新版的 Opera 不再识别这个样式，Firefox 4 以上的版本也不再使用 `-moz-box-shadow` 样式，但会识别并把它转换为 `box-shadow`。但是，我们还是保留了 `-moz-box-shadow style` 样式以支持 Firefox 3。第 14 页的代码上，为了功能性我们舍弃了一些整洁度。

样式到位以后，我们差不多要准备给光泽元素加上动画了。我们先加上变换声明，用来控制动画，接下来，就要依靠特定浏览器的前缀。

```
csssheen/style.css
header .sheen {
  -moz-transition:    all 2s ease-in-out;
  -webkit-transition: all 2s ease-in-out;
  -o-transition:     all 2s ease-in-out;
  transition:        all 2s ease-in-out;
}
```

变换定义有三个参数。第一个参数告诉浏览器要跟踪哪个 CSS 属性。在例子中，我们只需要跟踪 `left` 属性，因为闪光的动画是从头部滑过，也可以设置为 `all` 控制变换的所有属性改变。

第二个参数定义了动画花费的时间，其单位是秒。这个值可以是小数，比如 0.5 秒，如果希望一个长时间变换慢慢发生的话也可以设置很多秒。最后一个参数是所使用功能的名字，我们仅仅只用了一个默认功能，你也可以定义你自己的。**Ceaser**<sup>3</sup>是个很好用的自定义功能的工具。

接下来，我们还要添加一个样式声明来定义闪光动画结束的位置。因此，闪光应该在头部的右侧结束，我们可以给 `hover event` 加上如下代码

```
header: hover .sheen {  
  left: 900px;  
}
```

像上面这样写的话，当用户鼠标从头部移开时，闪光又会回到它开始的地方。我们希望这个动画是一次性的，所以不得不使用 **Javascript** 来改变页面的状态。我们给样式表添加一个特殊的样式 `loaded`，可以把闪光的位置始终固定在 `logo` 的尾部，例如：

```
csssheen/style.css  
header.loaded .sheen {  
  left: 900px;  
}
```

然后用 **jQuery** 给 `header` 添加这个样式，触发变换。

```
$(function() { $('header').addClass('loaded') })
```

如图 5 所示，做了这么多，只是把一个模糊的小棒子在屏幕上移动，但我们已经做好了闪光的样式，接下来需要在整体外观上添加单独的样式并稍作调整。我们会添加 `overflow: hidden;`，把悬在头部以外的闪光元素隐藏起来。

```
csssheen/style.css  
header {  
  overflow: hidden;  
}
```

---

<sup>3</sup> <http://matthewlein.com/ceaser/>



图 5 光泽的样式已经有了，但是在头部以外还是能被看到

添加好合适的样式以后，我们只需改变一个元素的 CSS 样式就能触发整个动画。我们不用再依靠 JavaScript 或者 Flash 就能在网站上添加一个流畅的动画。

这种方法还有一个额外的好处，就是节省用户的流量。尽管这对大多数用户来说没有影响，但有时用户会使用 iPad 或者其他移动设备访问我们的网站，这时就意味着更少的下载量和更快的访问速度。在设计、开发网站时要始终把网站优化这个思想放在心上。

对于不支持新样式的浏览器，我们的网站只显示 logo 图片。通过样式与内容分离，我们获得了很好的向下兼容以及更好的用户可用性，因为<img>标签包含了替代文字。

如果要让所有的浏览器都能运行这个动画效果，我们可以把上述方法作为 Flash 解决方案的后备，将<img>标签嵌套放置在 Flash 动画使用的<object>标签里面。

## 深入研究

### Further Exploration

我们只介绍了一点适合我们的变形和变换效果，还有其他合适的选择（如缩放和偏移）。我们还可以做些更细致的控制，例如每个变换的时间长短，甚至是我们究竟要做哪些变换。有些浏览器还可以让你自己定义变换，对于动画的内部可控性是令人兴奋且满意的。

## 另请参考

### Also See

- 1 号秘方 设计按钮和链接
- 2 号秘方 使用 CSS 设计评论
- 28 号秘方 以 Sass 搭建模块化样式表



## 4 号秘方 用 jQuery 创建交互幻灯片

### Creating Interactive Slideshows with jQuery

#### 问题

#### Problem

几年前，如果你想在网站上放置一个活动的幻灯片，则需要制作一个Flash动画。简单的工具使这一过程变得容易，但是对幻灯片里的照片进行维护则意味着重做Flash动画。此外，很多移动设备并不支持Flash播放，用户也就看不到幻灯片了。我们需要一个替代方案，能够在多平台上运行，并且易于维护。

#### 工具

#### Ingredients

- jQuery
- The jQuery Cycle plug-in<sup>4</sup>

#### 解决方案

#### Solution

我们可以使用jQuery和jQuery Cycle插件创建既简单又优雅的图片幻灯展示。这个开源工具只要在支持JavaScript的浏览器上就能给用户展示一个优美的幻灯片。

有许多基于JavaScript的图片循环插件，但是jQuery Cycle与众不同的地方就在于它的易用性。它有很多内置的变换效果，并且为操作图片提供了控制选项。它非常易于维护，还有一个很活跃的开发社区。所以jQuery Cycle是我们制作幻灯片的最好选择。

我们当前的主页有点死板、令人乏味，所以我们的boss要我们做一个幻灯片来展示公司形象。我们会拿一些照片来做示范，建立一个简单的原型以理解jQuery Cycle插件是如何工作的。

我们将从创建一个包含幻灯片的简单主页模版index.html开始，该模板通常包含以下这些代码：

---

<sup>4</sup> <https://github.com/malsup/cycle>

```
image_cycling/index.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>AwesomeCo</title>
  </head>
  <body>
    <h1>AwesomeCo</h1>
  </body>
</html>
```

接下来，我们要建立一个 `images` 文件夹，把 `boss` 给我们用来展示的图片放进去，你可以在本书源码目录的 `image_cycling` 文件夹下找到它们。

然后，我们在 `<head>` 部分紧接着 `<title>` 元素之后，引入 `jQuery` 和 `jQuery Cycle` 插件。我们还需要添加一个指向 `rotate.js` 的链接，里面包含了所有设置图片旋转用到的 `JavaScript`。

```
image_cycling/index.html
<script type="text/javascript"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js">
</script>
<script type="text/javascript"
  src="http://cloud.github.com/downloads/malsup/cycle/jquery.cycle.all.2.74.js">
</script>
<script type="text/javascript" src="rotate.js"></script>
```

添加一个 `ID` 为 `slideshow` 的 `<div>`，在 `<div>` 里加入图片，像这样：

```
image_cycling/index.html
<div id="slideshow">
  
  
  
  
  
  
</div>
```

在浏览器里打开我们的页面，你会看到图 6 所示的样子。这也告诉了我们当用户不启用 `JavaScript` 的时候，页面看起来会是什么样。可以看到，所有内容对于用户都是可见的，他们不会错过任何东西。

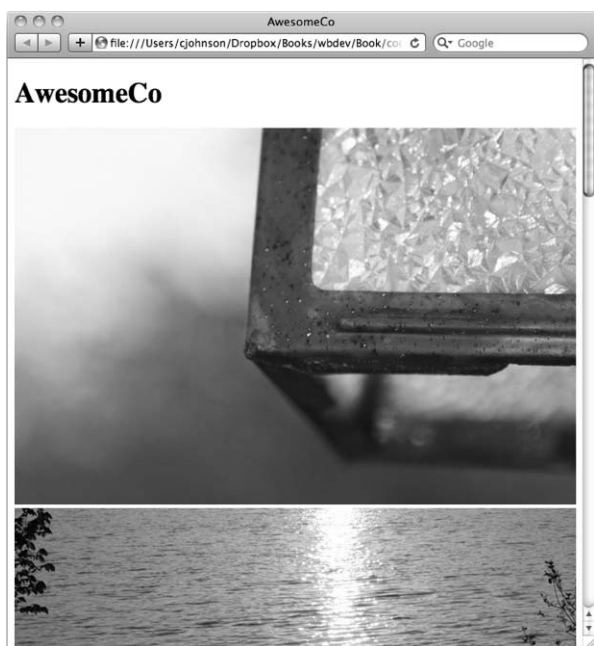


图6 图片并没有循环起来

我们还没有添加功能函数来触发 jQuery Cycle 插件，所以只能看到一些有序排列的图片。现在让我们添加 JavaScript 来初始化插件，开始展示幻灯片。先创建文件 `rotate.js` 并加上下列代码，用来设置 jQuery Cycle 插件：

```
image_cycling/rotate.js
$(function() {
  $('#slideshow').cycle({fx: 'fade'});
});
```

jQuery Cycle 插件有许多不同的选项。我们可以让图片在变换时淡入淡出，同时缩放、擦除甚至是摇动<sup>5</sup>。你可以在 jQuery Cycle 的官网上找到全部选项的列表。这里我们继续使用 `fade` 功能，它非常简单又很优美。我们在调用 `cycle()` 时，在方法里添加一小段代码定义它。

`fx: 'fade'`

现在所有东西都准备就绪，我们再来看一下页面。这次我们只看到一张图片，几秒钟之后，图片开始循环播放。

---

<sup>5</sup> <http://jquery.malsup.com/cycle/options.html>

## 添加播放和暂停按钮

现在我们有了一个可以正常运行的幻灯片，并给 boss 看了下效果，她说：“这很棒，不过我希望能有一个暂停按钮，这样当客户看到喜欢的图片时就可以把幻灯片暂停播放。”幸运的是，jQuery Cycle 插件自带了这样的功能。

我们使用 JavaScript 在页面上添加这些按钮，因为只有当幻灯片是可活动的时候才需要它们。这样，就不会向没有 JavaScript 支持的用户显示无用的控制按钮了。为了实现这个功能，我们要创建两个功能函数 `setupButtons()` 和 `toggleControls()`。第一个功能函数将在页面上添加按钮并且给它们绑定 `click()` 事件。点击事件会通知幻灯片是要暂停还是重新开始播放。我们还要用 `click()` 事件来调用 `toggleControls()` 功能切换按钮，只显示出相关的那一个。

image\_cycling/rotate.js

```
var setupButtons = function(){
    var slideShow = $('#slideshow');

    var pause = $('<span id="pause">Pause</span>');
    pause.click(function() {
        slideShow.cycle('pause');
        toggleControls();
    }).insertAfter(slideShow);

    var resume = $('<span id="resume">Resume</span>');
    resume.click(function() {
        slideShow.cycle('resume');
        toggleControls();
    }).insertAfter(slideShow);

    resume.toggle();
};

var toggleControls = function(){
    $('#pause').toggle();
    $('#resume').toggle();
};
```

你能注意到我们在给 jQuery 选择器设置变量。这样我们能够以更加简便的方式来操作 DOM。几乎所有的 jQuery 方法都会返回 jQuery 对象，这也给我们带来了便利，正因为如此，我们才能把 `insertAfter()` 函数和 `click()` 绑定事件链接到一起。

为了触发 `setupButtons()` 函数，我们需要在 jQuery 的准备事件中，在 `cycle()` 调用的下面添加对 `setupButtons()` 的调用。

```
image_cycling/rotate.js
$(function() {
  $('#slideshow').cycle({fx: 'fade'});
  setupButtons();
});
```

让我们在浏览器里再看一下页面。我们能看到如图 7 所示的暂停按钮。当幻灯片开始播放以后，我们可以点击暂停按钮，变换停止的同时，暂停按钮也被恢复按钮替代了。点击恢复按钮，图片便继续播放。

## 深入研究

### Further Exploration

这个幻灯片很容易实现，而且所有的设置在插件的官网上都有提到。我们可以把它扩展一下<sup>6</sup>，使它包含更多的功能。

为了增强视觉体验，jQuery Cycle 插件有多种变换效果的设置，比如随机播放、摇动或者揭开效果。我们只需要在调用 `cycle()` 时改变 `fx` 选项的值，就可以让幻灯片使用其中任意一种效果。除了图片，也可以循环其他元素，包括复杂的 HTML 代码块。这只是 jQuery Cycle 插件可发掘潜力中的一部分，赶紧去探索并尝试一下吧。

## 另请参考

### Also See

- 3 号秘方 用 CSS3 变形技术创建动画
- 35 号秘方 Javascript 测试框架 Jasmine

---

<sup>6</sup> <http://jquery.malsup.com/cycle/options.html>

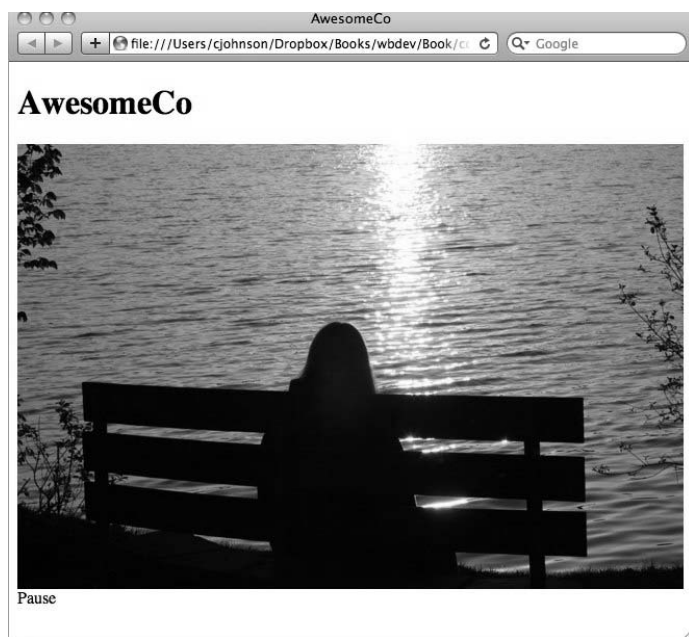


图 7 带控制的滚动图片

## 5 号秘方 设计创建行内帮助对话框

### Creating and Styling Inline Help Dialogs

#### 问题

##### Problem

我们有一个有很多链接的页面，这些链接指向网站其他地方的补充内容。点击这些链接时会跳转到这些页面，当我们读到段落中间的时候，弹出一个新窗口会打断阅读流。所以时尚的行内显示内容会非常棒，只有当完全必要的时候才保留页面内容。

#### 工具

##### Ingredients

- jQuery
- jQuery UI<sup>7</sup>
- jQuery Theme<sup>8</sup>

#### 解决方案

##### Solution

我们希望信息能成为页面流的一部分，即使在旧版本的浏览器中也能运行，我们将用 JavaScript 来替代 HTML 链接指向附件内容，并且在页面上行内显示。这样做的话，对于不支持 JavaScript 的浏览器也能保证内容的可读性，同时对于启用了 JavaScript 的新版浏览器的用户会有更佳样式和更平滑的体验。加载这些内容的时候，我们可通过使用一些 jQuery 动画效果、添加一个规整的对话框以使其看起来更棒，如图 8 所示。

在开始介绍 JavaScript 之前，先来创建一个加载 jQuery 和 jQuery UI 的简单页面，接着用 jQuery theme 插件来添加第一个指向行内内容的链接。

```
inlinehelp/index.html
<html>
  <head>
    <link rel="stylesheet" href="jquery_theme.css"
          type="text/css" media="all" />
```

---

<sup>7</sup> <http://jqueryui.com>

<sup>8</sup> <http://jqueryui.com/themeroller/>

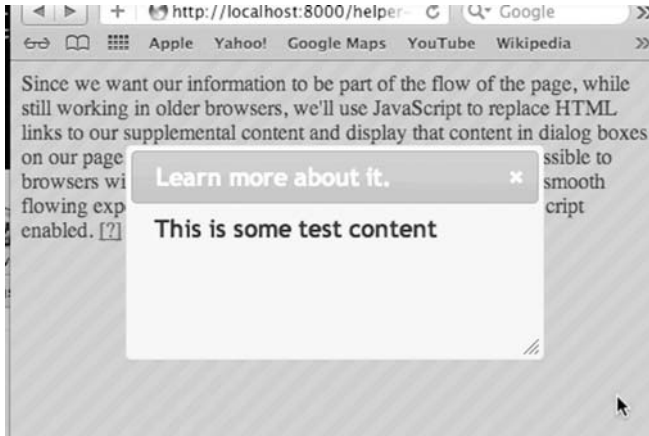


图 8 覆盖在内容上的模式对话框

```
<script type="text/javascript"
src='http://ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js'>
</script>
<script type="text/javascript"
src='http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.14/jquery-ui.min.js'>
</script>
<script type="text/javascript" src='inlinehelp.js'></script>
</head>
<body>
  <p>
    This is some text.
    <a href="test-content.html"
      id="help_link_1"
      class="help_link"
      data-style="dialog"
      data-modal='true'>
      Learn more about it.
    </a>
  </p>
</body>
</html>
```

我们希望在网站各处频繁地添加这个功能，所以它实施起来应该尽可能容易些。当所有代码都完成以后，我们的帮助链接看起来就像是这样：



```

<a href="a.html" id="help_a" class="help_link" data-style="dialog">
  More on A
</a>
<a href="b.html" id="help_b" class="help_link" data-style="dialog"
  data-modal="true">
  More on B
</a>
<a href="c.html" id="help_c" class="help_link" data-style="clip">
  More on C
</a>
<a href="d.html" class="help_link" data-style="fold">
  More on D
</a>

```

注意，我们使用了 `data-` 属性来声明样式和模式设置。这是 **HTML5** 规范的一部分，允许在 **HTML** 元素上自定义数据属性。这使得我们在给元素设置信息的同时还可以维持标记的有效性。

下列代码是我们写的脚本的一个简单示例，只设置了几个参数，然后调用了 `displayHelpers()` 函数。全部设置好以后，我们要做的就是给需要行内显示内容的链接添加一个样式以及可选的动画样式，并且指定是否用模式对话框。

```

inlinehelp/inlinehelp.js
$(function() {
  var options = {
    helperClass: "help_dialog"
  }

  displayHelpers(options);
});

```

使用 **jQuery** 的 `ready()` 函数，我们可以确保在开始操作 **DOM** 之前页面已经完全加载了。这样页面上所有的内容都能呈现出来，代码开始运行时，我们不会错过任何东西。这里设置了几个参数，尽管它们不是必需的，但是能让我们的链接和对话框看起来更棒。然后我们把参数传进 `displayHelpers()` 函数，开始更新页面。

```

inlinehelp/inlinehelp.js
function displayHelpers(options) {
  if (options != null) {
    setIconTo(options['icon']);
    setHelperClassTo(options['helper_class']);
  }
}

```

```

else {
    setIconTo();
    setHelperClassTo();
}

$("a.help_link").each(function(index,element) {
    if ($(element).attr("id") == "") { $(element).attr("id", randomString()); }
    appendHelpTo(element);
});
$("a.help_link").click(function() { displayHelpFor(this); return false; });
}

```

我们从设置图标或文字开始，表明那里有些东西需要用户查看一下。我们希望这些链接和它们周围的内容相关联，这样一来就可以移除实际的文本。我们还要给对话框添加一个样式来设置它们的外观。

```

inlinehelp/inlinehelp.js
function setIconTo(helpIcon) {
    var isImage = /jpg|jpeg|png|gif$/
    if (helpIcon == undefined)
        { icon = "[?]"; }
    else if (isImage.test(helpIcon))
        { icon = "<img src='"+helpIcon+"'>"; }
    else
        { icon = helpIcon; }
}

```

setIconTo() 函数首先查看 help\_icon 选项有没有被传递进来。如果没有，就使用默认值[?]。如果有参数传递进来，我们通过查看字符串结尾是否是通常的图片后缀名来判断它是不是一个图片路径。如果是的话，我们要把它插入到<img>元素的路径里，否则就显示传进来的文本。这样即使我们传进来一个完整的<img>元素也不会有问题，它还是可以正常显示。

接下来我们要给对话框加一个样式，这样它就能按照我们自己写的 CSS 或者使用 jQuery UI 主题指定的外观显示出来。

```

inlinehelp/inlinehelp.js
function setHelperClassTo(className) {
    if (className == undefined)
        { helperClass = "help_dialog"; }
    else
        { helperClass = className; }
}

```

setHelperClassTo()函数会查看是否给对话框设置了可用的样式参数。如果有的话,就使用这个参数值,如果没有,则使用默认的 help\_dialog 样式。

我们还要确认每个链接都有唯一的 ID,因为我们需要使用这个 ID 来关联它们各自的<div>对话框。如果没有的话,我们就要先给链接添加一个。

```
inlinehelp/inlinehelp.js
$( "a.help_link" ).each( function( index, element ) {
    if ( $(element).attr("id") == "" ) { $(element).attr("id", randomString()); }
    appendHelpTo(element);
});
```

为了确保所有的链接都有 ID,我们给页面上的每个链接都加上 help\_link 样式,并检查它们是否设置了 ID 属性。如果没有,就生成一个随机字符串来作为 ID。

```
inlinehelp/inlinehelp.js
function randomString() {
    var chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    var stringLength = 8;
    var randomstring = '';
    for (var i=0; i<stringLength; i++) {
        var rnum = Math.floor(Math.random() * chars.length);
        randomstring += chars.substring(rnum,rnum+1);
    }
    return randomstring;
}
```

randomString()是一个生成包含字母和数字的 8 位随机字符串的简单函数。这足以给页面上任何没有 ID 属性的链接附上一个 ID 值。

确认链接有了 ID 之后,我们调用 appendHelpTo()函数给链接加上帮助图标,并且将存有链接指向页面内容的对话框元素准备好。

```
inlinehelp/inlinehelp.js
Line 1 function appendHelpTo(element) {
-   if ( $(element).attr("title") != undefined ) {
-       title = $(element).attr("title");
-   } else {
5       title = $(element).html(); 5
-   }
-   var helperDiv = document.createElement('div');
-   helperDiv.setAttribute("id",
-       $(element).attr("id") + "_" + $(element).attr("data-style"));
10  helperDiv.setAttribute("class", 10
-       $(element).attr("data-style") + " " + helperClass);
```

```

- helperDiv.setAttribute("style", "display:none;");
- helperDiv.setAttribute("title", title);
-
15 $(element).after(helperDiv); 15
- $(element).html(icon);
-
}

```

appendHelpTo() 函数被调用之后，会在链接被点击时插入一个包含内容的 <div>。我们把链接的 ID 和刚开始传入的样式参数组合在一起后赋给它作为 ID 值。我们给它设置了几个样式：参数传入的样式及以指定使用的动画样式。最后，把 <div> 设置为 display:none，使它在被点击以后才在页面上显示出来。

appendHelpTo() 的第三行用我们的图标替换了原始链接，加上了行内的 [?] 或是参数中设置的其他值。

inlinehelp/inlinehelp.js

```

$( "a.help_link" ).click( function() { displayHelpFor( this ); return false; } );

```

现在我们加上调用最后一行的 displayHelpers() 函数，遍历所有样式名为 help\_link 的元素，用 displayHelpFor() 来覆盖默认的响应函数并且返回 false 值，使正常的点击事件不会被执行。

inlinehelp/inlinehelp.js

```

function displayHelpFor( element ) {
    url = $(element).attr("href");
    helpTextElement = "#" + $(element).attr("id") + "_" +
        $(element).attr("data-style");
    if ( $(helpTextElement).html() == "" ) {
        $.get( url, { },
            function( data ) {
                $(helpTextElement).html( data );
                if ( $(element).attr("data-style") == "dialog" ) {
                    activateDialogFor( element, $(element).attr("data-modal") );
                }
                toggleDisplayOf( helpTextElement );
            } );
    }
    else { toggleDisplayOf( helpTextElement ); }
}

```

displayHelpFor() 函数首先从最近点击的链接里取得 URL 地址，这样就知道要显示哪个页面。接下来给之前插入页面的 <div> 元素建立 ID。这个 <div> 就是我们放置链接页面内容的地方。但是在加载内容之前，我们需要先确认该内容还没有被加载过。

如果<div>是空的，那就说明还没有加载过。如果已经加载过一次，就没有必要重复加载了，所以我们需要调用 `toggleDisplayOf()` 函数。通过避免重复加载，既可缩短用户的等待时间，又可降低我们的带宽开销。

如果页面内容没被加载过，就用 jQuery 的 `get()` 函数通过 Ajax 获取 URL 的内容并填充到<div>里去。完成之后，我们还要查看一下行内文字的请求样式。如果我们使用了对话框样式，就需要调用 `activateDialogFor()` 函数在 DOM 中准备好对话窗口并且为它添加好样式。

```
inlinehelp/inlinehelp.js
function activateDialogFor(element,modal) {
    var dialogOptions = { autoOpen: false };
    if (modal == "true") {
        dialogOptions = {
            modal: true,
            draggable: false,
            autoOpen: false
        };
    }
    $("#"+$(element).attr("id")+"_dialog").dialog(dialogOptions);
}
```

这样就在页面中注册了这个对话框元素，能够访问它了。激活以后，通过设置 `autoOpen` 参数值为 `false` 来确保对话框是关闭的。之所以这样做，是因为要与其他对话框保持一致，需要打开对话框时使用 `toggleDisplayOf()` 函数来操作。

```
inlinehelp/inlinehelp.js
function toggleDisplayOf(element) {
    switch(displayMethodOf(element)) {
        case "dialog":
            if ($(element).dialog('isOpen')) {
                $(element).dialog('close');
            }
            else {
                $(element).dialog('open');
            }
            break;
        case "undefined":
            $(element).toggle("slide");
            break;
        default:
            $(element).toggle(displayMethod);
    }
}
```

```
inlinehelp/inlinehelp.js
function displayMethodOf(element) {
    helperClassRegex = new RegExp(" "+helperClass);
    if ($(element).hasClass("dialog"))
        { displayMethod = "dialog"; }
    else
        {displayMethod=$(element).attr("class").replace(helperClassRegex,"");}
    return displayMethod;
}
```

在 `toggleDisplayOf()` 函数里，才最终显示新的内容。首先我们用 `displayMethodOf()` 函数解决怎样显示的问题。可以使用 jQuery UI Effects 库或者 Dialog 库里的任何动画方法，所以首先检查链接本身有没有带上对话框的样式。如果有，就返回它的值，否则就取得链接的样式再移除我们指定的样式，这样就只保留了显示内容要用到的动画样式。

再回到 `toggleDisplayOf()` 上来，我们用 `display` 方法来决定如何显示或隐藏内容。如果是对话框，我们就用 jQuery 的辅助方法 `isOpen` 来判断它是否已经打开，然后根据需要打开或关闭对话框。如果不能确定动画样式，就使用默认方式和效果来显示元素。最后，如果真的有 `display_method` 函数，就使用该函数来控制内容的隐藏和显示。

这些代码全部完成之后，就能很容易地添加新的行内元素了，而且还能很好地维持在所有浏览器上的兼容性。另外，我们的代码是松耦合的，可以很好地处理一些新的动画效果，不用为了解决新版本 jQuery 的兼容性而去做什么改动。

## 深入研究

### Further Exploration

在初始化代码的时候，预先声明一些参数是个非常好的习惯，尤其是帮助链接的样式以及默认的动画样式。现在这些属性都是写死的，所以我们需要确保即使没做任何设置，它们也会有一个默认值，就像我们给放置内容的 `<div>` 和帮助图标/文字设置样式一样。

现在，我们舍弃了原始链接的文字内容，并将其替换成图标。除了完全抛弃这些文字之外，我们还可以将它们作为链接的 `title`，能够让用户的鼠标在移到链接上时，大致明白该页面的内容，同时也能使我们的页面保持更好的连贯性。

## 另请参考

### Also See

- 29 号秘方 以 CoffeeScript 清理 JavaScript
- 35 号秘方 JavaScript 测试框架 Jasmine

## 第 2 章

---

## 用户界面

# User Interface Recipes

无论是传递静态的内容，还是表现交互式应用，都需要可用的界面。本章探讨一种信息展示的新方法，以此建立更易维护和响应的客户端界面。



## 6 号秘方 创建 HTML 格式的电子邮件模板

### Creating an HTML Email Template

#### 问题

#### Problem

创建 HTML 格式的电子邮件感觉似乎有点回到 CSS 之前，当时人们使用表格进行布局、使用<font>标签进行各种样式的控制。很多如今常见的技术，HTML 格式的电子邮件都无法识别和处理。在多种浏览器上测试网页相比在 Outlook、Hotmail、Gmail 或者 Thunderbird 等邮件工具上做大量的测试还是相对容易得多，更别提移动设备上各种各样的邮件应用程序了。

但我们的工作绝不是抱怨事情有多困难，而是找到解决方法。这其实是一项颇具工作量的任务，不仅需要生成可读的 HTML 格式的电子邮件，还需要确保邮件不被识别为垃圾邮件。

#### 工具

#### Ingredients

- 使用 Litmus.com 的免费试用账号用于测试邮件

#### 解决方案

#### Solution

由于电子邮件客户端的限制，设计 HTML 格式的电子邮件意味着需要舍弃很多当前的网页开发技术，除此之外，还需要避免邮件被识别为垃圾邮件，并能够比较方便地在多个设备上进行测试。在多个平台上需要同时具有可用性、可读性和高效性，最好的实现便是采用基于表格布局的 HTML，虽然古老但是可靠。

## HTML 电子邮件基础

### HTML Email Basics

从概念上而言，HTML 电子邮件并不困难。毕竟，创建一个简单的 HTML 页面并不费力，但与网页一样，我们不能保证用户看到的就是我们所创建的，因为每个邮件客户端在展示邮件信息上都会有些许差异。

对于大部分人而言,使用各种基于网页的客户端,比如 Gmail、Hotmail 和 Yahoo,它们往往去掉或者忽略在标记中所定义的样式表。实际上,Google 邮箱会删除定义在<style>标签中的样式,以避免邮件与其界面风格不符。我们也不能依赖外部的样式表,这是因为大部分邮件客户端不会在不提示用户的情况下自动获取远程文件,因此,无法真正地在 HTML 邮件中使用 CSS 来布局。

由于 Google 和 Yahoo 邮箱都会删除或者重命名邮件中的<body>标签,所以最好将邮件封装在另一个可以取代<body>的标签中。

有些客户端无法解析 CSS 缩略申明,因此必须列出每一项定义,比如:

```
#header{padding: 20px;}
```

可能被一些旧的客户端所忽略,需要替代为完整写法:

```
#header{
  padding-top: 20px;
  padding-right: 20px;
  padding-bottom: 20px;
  padding-left: 20px;
}
```

Outlook 2007 和 Lotus Notes 等桌面客户端无法处理背景图片,而 Lotus Notes 更是无法显示 PNG 图片。这乍一看似乎没什么大不了的,但要知道,数百万的企业用户是将它们作为首选客户端的。

这些不是我们会遇到的所有问题,但却是最普遍的。针对各种不同客户端,电子邮件标准项目<sup>1</sup>拥有一份全面的问题清单。

## 如同 1999 年般聚会

总结之前的分析,最有效的 HTML 邮件设计中需要具备以下最基本的 HTML 特征:

- 使用简单的 HTML 标签,配以最少的 CSS 样式;
- 使用 HTML 表格进行布局,而不是过多的现代技术;
- 不使用复杂的排版;
- 使用极简单的 CSS 样式。

总之,需要设计一个简单的邮件模板,而不使用近十年来的网络开发技术。

---

<sup>1</sup> <http://www.email-standards.org/>

考虑到这一点，模板中需要使用表格进行布局，而应用开发者将真实的内容添入模板内。需要找出解决方案的是如何编写模板，使其在所有流行的邮件客户端上可读。

发货单上一般需要有几项：包括页眉、页脚、发货地址和账单地址部分、顾客所购物品清单、物品价格、物品数量、金额小计、总金额，以及备注。

由于一些基于网页的邮件的客户端会去除或者重命名<body>元素，需要使用自己的顶层元素来作为邮件的容器。为了尽可能保证鲁棒性，创建一个外层表格作为容器，并在其中放置其他表格来实现页眉、页脚和内容的区分。图 9 所示是发货单模板，并给出了一个使用此模板的简单例子。

使用 HTML 4.0 来编写邮件的模板：

```
htmlmail/template.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  <title>Invoice</title>
</head>
<body>
  <center>
    <table id="inv_container"
      width="95%" border="0" cellpadding="0" cellspacing="0">
      <tr>
        <td align="center" valign="top">
          </td>
        </tr>
      </table>
    </center>
  </body>
</html>
```

为了确保发货单在邮件客户端的中间显示，需要采用古老且被弃用的<center>标签，因为这是唯一跨客户端的方式。不过也别担心，至少我们不会使用<blink>标签。

header				
invoice			date	
from		to		
	line items			
subtotals				
notes				
footer				

图 9 发货单模板

下一步，创建页眉。用一个表格填入公司名称，用另一个两列的表格填入订单号和日期。

```
htmlmail/template.html
<table border="0" cellpadding="0" cellspacing="0" width="100%">
  <tr>
    <td align="center" bgcolor="#5d8eb6" valign="top">
      <h1><font color="white">AwesomeCo</font></h1>
    </td>
  </tr>
</table>

<table border="0" cellpadding="0" cellspacing="0" width="98%">
  <tr>
    <td align="left" width="70%"><h2>Invoice for Order #533102 </h2></td>
    <td align="right" width="30%"><h3>December 30th, 2011</h3></td>
  </tr>
</table>
```

由于一些基于网页的客户端去除了CSS，我们不得不使用HTML属性来指定背景和文本的颜色。将第一个表格的宽度设为100%，而将第二个设为98%，由于整个外层表格在页面上是居中的，需要在左右边缘留下一定空间使表格内文本不接触外层表格的边缘。

下一步，添加包含“发货地址”和“收货地址”的表格。

htmlmail/template.html

```
<table id="inv_addresses" border="0"
  cellpadding="2" cellspacing="0" width="98%">
  <tr>
    <td align="left" valign="top" width="50%">
      <h3>From</h3>
      AwesomeCo Inc. <br>
      123 Fake Street <br>
      Chicago, IL 55555
    </td>
    <td align="left" valign="top" width="50%">
      <h3>To</h3>
      GNB <br>
      456 Industry Way <br>
      New York, NY 55555
    </td>
  </tr>
</table>
```

下一步，添加用于发货单本身的表格。

htmlmail/template.html

```
<table border="0" cellpadding="2" cellspacing="0" width="98%">
  <caption>Order Summary</caption>
  <tr>
    <th bgcolor="#cccccc" align="left" valign="top">SKU</th>
    <th bgcolor="#cccccc" align="left" valign="top">Item</th>
    <th bgcolor="#cccccc" align="left" valign="top">Price</th>
    <th bgcolor="#cccccc" align="left" valign="top" width="10%">QTY</th>
    <th bgcolor="#cccccc" align="left" valign="top" width="10%">Total</th>
  </tr>
  <tr>
    <td valign="top">10042</td>
    <td valign="top">15-inch MacBook Pro</td>
    <td align="right" valign="top">$1799.00</td>
    <td align="center" valign="top">1</td>
    <td align="right" valign="top">$1799.00</td>
  </tr>
  <tr>
    <td valign="top">20005</td>
    <td valign="top">Mini-Display Port to VGA Adapter</td>
    <td align="right" valign="top">$19.99</td>
```

```

        <td align="center" valign="top">1</td>
        <td align="right" valign="top">$19.99</td>
    </tr>
</table>

```

实际上，这是一个数据表格，所以需要确保它使用了正确的属性，比如表头和标题。

对于统计项，需要使用独立的表格来展示，这是由于某些邮件客户端在跨列行的显示上存在问题。

htmlemail/template.html

```

<hr>
<table border="0" cellpadding="2" cellspacing="0" width="98%">
    <tr>
        <td align="right" valign="top">Subtotal: </td>
        <td align="right" valign="top" width="10%">$1818.99</td>
    </tr>

    <tr>
        <td align="right" valign="top">Total Due: </td>
        <td align="right" valign="top"><b>$1818.99</b> </td>
    </tr>
</table>

```

放置另一个表格用于显示发货单的备注。

htmlemail/template.html

```

<table border="0" cellpadding="0" cellspacing="0" width="98%">
    <tr><td align="left">
        <h2>Notes</h2>
        <p>Thank you for your business!</p>
    </td></tr>
</table>

```

最后，添加页脚，如同页眉一样，表格只有一个单元格，并设置宽度为100%。

htmlemail/template.html

```

<table id="inv_footer" border="0"
    cellpadding="0" cellspacing="0" width="100%">
    <tr>
        <td align="center" valign="top">
            <h4>Copyright &copy; 2012 AwesomeCo</h4>
            <h4>
                You are receiving this email because you purchased
                products from us.
            </h4>
        </td>
    </tr>
</table>

```

页脚是向用户解释为何会收到这一邮件的好途径。对发货单而言也是如此，但是对于新闻邮件而言，还需要提供给用户一些链接来管理他们自己所订制的邮件。

至此，我们已经创建了一个简单但是可读的 HTML 发货单。但是对于那些无法处理 HTML 邮件的客户端呢？

## 支持不支持 HTML 的情况

并不是每一个 HTML 邮件客户端都支持 HTML 邮件，同时，那些支持的也都表现不一致。为了让人们在不支持 HTML 邮件的设备上阅读，最常见的解决方法就是在邮件的顶部提供一个链接，指向邮件在服务器上的拷贝，点击链接后，用户能够在其浏览器上读到整个内容。

我们只需在用户账号信息处放置指向发货单拷贝的链接，比如邮件的右上角，整个表格之上，就很容易被发现。更进一步，有些客户端软件提供预览功能，这使得这部分用户能够在不打开邮件的情况下直接访问发货单。

```
htmlmail/template.html
```

```
<p>
  Unable to view this invoice?
  <a href="#">View it in your browser instead</a>.
</p>
```

MailChimp 和 Campaign Monitor 等第三方系统还让用户在他们的服务器上如同访问静态页面一样访问 HTML 邮件。

我们也可以构建多部分邮件，同时发送纯文本和 HTML 两种版本。在邮件中插入两部分主体，同时在邮件中使用特殊的头信息来告诉客户端，邮件中包含纯文本和 HTML 两种版本。按照这一方案，除了 HTML 版本外，还需要开发并维护一份纯文本版本的发货单，或者只是放置一个链接，指向发货单的网页版本。

发送多部分邮件已经超出了此秘方的范围，大部分基于网页的邮件客户端有发送多部分邮件的选项。维基百科中关于 MIME<sup>2</sup>的词条对此有一个很好的概述。

---

<sup>2</sup> <http://en.wikipedia.org/wiki/MIME>

// Joe 问:

☹ 我们能否使用语义化标记来替代表格?

许多在意标准的开发者避免使用表格, 而更乐意使用语义化标记, 并依赖 CSS 来管理布局。他们并不关心邮件客户端是否将 CSS 去除, 因为邮件本身是可读和可访问的。

不幸的是, 万一你的老板坚持邮件在客户端上也需要有一致的表现。而邮件客户端这种东西, 基于标准的网页开发技术是无法消除它的。这就是为什么在此秘方中使用表格实现的原因。

## 使用 CSS 设计样式

使用表格来布局是由于很多基于页面的邮件客户端去除了 CSS 样式, 而无法依赖 CSS 的浮动或者绝对定位。那些基于网页的客户端将所有东西去除, 并不是因为其开发者是心胸狭隘的标准厌恶者, 而是因为如果允许 CSS, 就有可能使邮件和基于页面的应用之间产生样式冲突。

然而, 仍然有两个原因支持使用 CSS。首先, 对于那些使用支持 CSS 邮件客户端的用户看起来更棒; 其次, 可以重用发货单模板在静态页面上, 如同“支持不支持 HTML 的情况”中谈到的那样。

很多邮件客户端去除了文档中的<head>部分, 只将<style>标签中的样式信息放置于容器表格之上。

去掉表头周围的间隔, 可以去掉浪费的空间。除了页脚外, 给每个部分应用背景颜色、表格边框, 同时增大内部表格之间的空间, 这样就不会显得拥挤。

htmlemail/template.html

```
<style>
table#inv_addresses h3,
table#inv_footer h4{
  margin: 0; }
table{
  margin-bottom: 20px;
}
```



```
table#inv_footer{  
    margin-bottom: 0;  
}  
  
body{  
    background-color: #eeeeee;  
}  
  
table#inv_container{  
    background-color: #ffffff;  
    border: 1px solid #000000;  
}  
</style>
```

正确使用了这些样式，发货单看起来就如图 10 所示的那样。不过我们还没结束，至少需要测试一下。

## 测试邮件

在查看客户端表现之前，可以先看一下邮件在阅读者那边的情况。发送给周围的同事，或者创建 Gmail、Yahoo 和 Hotmail 的账号来展示效果，不过手动测试非常费时。

Litmus<sup>3</sup>提供了一系列工具来帮助人们测试网页和邮件，这些工具不仅支持大量的邮件客户端和浏览器，还包括移动设备。但是这项服务并不免费，我们可以使用试用账号来确保我们的发货单如期工作。

登录 Litmus 账号，创建一个测试，选择目标客户端，然后发送邮件给 Litmus 提供的地址或者通过网络接口上传 HTML 文件。使用 HTML 上传的方法不提供纯文本的反馈，因此一些测试只能以 HTML 的方式反馈，这对测试而言已经足够了。

Litmus 会给出一份详细报告，展示我们所发送的邮件在各邮件客户端上所呈现的样子，如图 11 所示。

由图可知，所写的代码使得各种主要平台上所看到的发货单邮件基本一致，且对大部分用户而言颇具可读性。

---

<sup>3</sup> <http://litmus.com/>

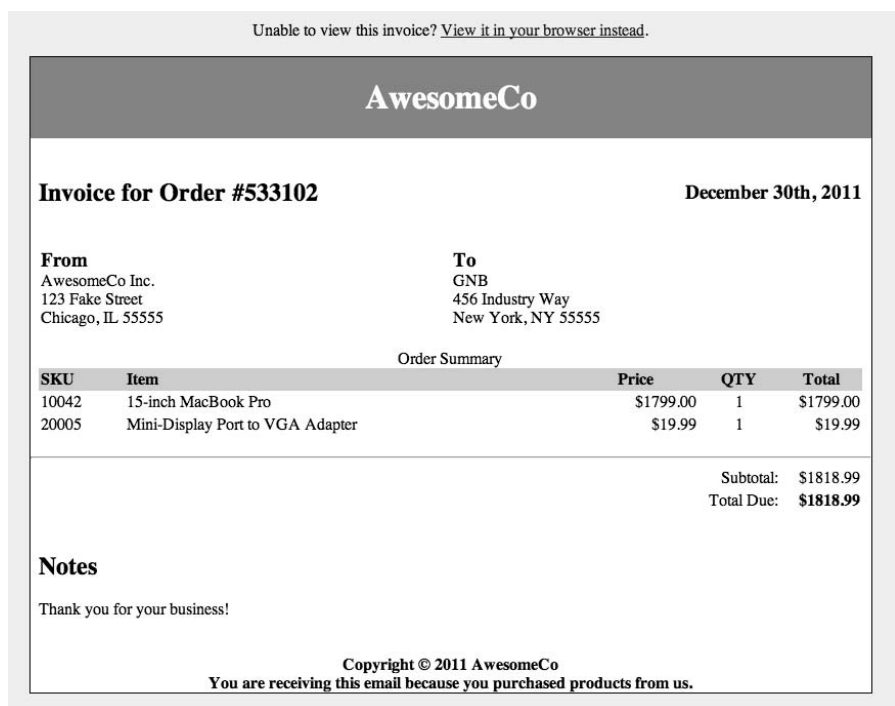


图 10 完整的发货单

## 图像和邮件

在此秘方中并没有谈及图像，主要因为两个原因。首先，提供图像就意味着需要在服务器上放置图像并在邮件中包含此图像的链接；其次，很多公司利用图像来追踪邮件是否被打开，所以大部分邮件客户端是关闭加载图像的。

如果决定在邮件中使用图像，需要确保遵循以下一些简单的规则：

- 确保图像在服务器上可用，并且不要改变图像的链接，因为你不知道用户会在什么时候打开邮件；
- 由于图像往往默认是被禁用的，因此应确保在图像上声明了有描述性的 alt 属性；



图 11 测试结果

- 放置图像在规则的<img>标签内，很多邮件客户端不支持表单元格的背景图像，甚至更少支持 CSS 背景图像；
- 由于图像经常被拦截，因此使用图像作为邮件的主体是一个非常糟糕的主意。也许这样做能使外观更漂亮，但会导致可访问性的问题。

在邮件中正确使用图像是非常有效的方式，没必要害怕使用，只是需要留意会遇到的问题。

## 深入研究

### Further Exploration

简单的邮件模板为接收者展示了一个可读的发货单，但它还不具备满足营销或通信的需求。对此，需要更多样式、更多图像，以及更多针对不同邮件客户端的“特殊处理”。

MailChimp<sup>4</sup>了解发送邮件的一些事，毕竟这是其生意。如果你想学习更多关于邮件模板的事情，可以研究 MailChimp 已经开源的邮件模板<sup>5</sup>。他们已经在各主要客户端上测试过了，而且这些注释良好的代码能使我们学到很多 Hack 技巧，能适配各种主要邮件客户端。

4 <http://www.mailchimp.com>  
5 <https://github.com/mailchimp/Email-Blueprints>

## 7 号秘方 多 Tab 界面的内容切换

# Swapping Between Content with Tabbed Interfaces

### 问题

#### Problem

有时需要在一起显示多个相似的信息，比如某个短语的各种语言版本或者代码实例的多种编程语言版本。这些信息可以一个接一个地显示出来，但会占用很多空间，特别是在内容很长的时候。需要有一种方式能让用户很方便地在各信息之间切换和比较，却不占用过多的屏幕空间。

### 工具

#### Ingredients

JQuery

### 解决方案

#### Solution

使用 CSS 和 JavaScript 将内容以 Tab 式的交互放入页面，每一部分内容根据其类别会生成一张标签页，同一时间只有一张 Tab 被显示。需要确保的是设计非常灵活，以使任意数量的 Tab 都能得到满足。最终效果如图 12 所示。

为了尝试接触更广泛的受众，往往采用多种语言来显示产品要求。首先构建一个简单的页面用于证明概念，然后再决定进一步前进的方向。

### 构建 HTML

让我们从构建 HTML 开始，创建展示给用户的元素。为了证明概念，使用配套的两种文本，英语及其拉丁语翻译。

```
swapping/index.html
<!DOCTYPE html>
<html>

  <head>
    <title>Swapping Examples</title>
```

```

<script type="text/javascript"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js">
</script>
<link rel="stylesheet" href="swapping.css" type="text/css" media="all" />
<script type="text/javascript" src="swapping.js"></script>
</head>
<body>
  <div class="examples">
    <div class="english example">
      Nor again is there anyone who loves or pursues or desires
      to obtain pain of itself, because it is pain, but occasionally
      circumstances occur in which toil and pain can procure him some
      great pleasure.
    </div>

    <div class="latin example">
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
      do eiusmod tempor incididunt ut labore et dolore magna aliqua.
      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.
    </div>
  </div>
</body>
</html>

```

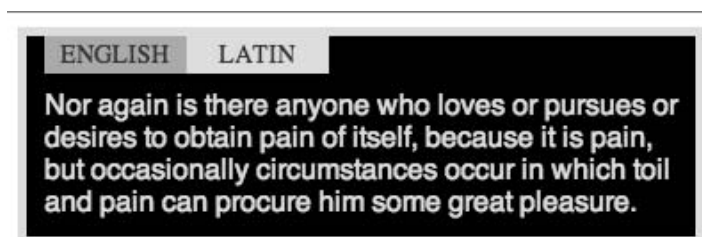


图 12 Tab 形式界面

我们已经建立了基本的结构，class 为 “examples” 的 <div> 标签包含了需要显示的所有文本，其中每一个 <div> 标签包含了供用户切换的实际内容。

现在，将一些 JavaScript 放到一起来创建 Tab 交互使得用户可以在两种文本间切换。我们将使用 jQuery 库来获得一些辅助方法和捷径。

## 创建 Tab 交互

首先，创建一个名为 styleExamples() 的函数，用于管理 JavaScript 不同部分的调用。

// Joe 问:

我们能否使用 jQuery 提供的 Tab 控件来完成这件事情?

当然可以, 但是 jQuery 提供的 Tab 控件包含了很多我们不会使用的功能, 比如事件挂载。创建自己的 Tab 可以保持实现的轻量级并让我们更了解这一切是如何运作的。

swapping/swapping.js

```
function styleExamples(){
  $("div.examples").each(function(){
    createTabs(this);
    activateTabs(this);
    displayTab($(this).children("ul.tabs").children().first());
  });
}
```

通过定位 class 为 “examples” 的 <div> 标签, 来获得我们的容器, 并将每一个子容器传入函数 createTabs(), 这一函数将创建供用户进行内容切换的 Tab 交互。现在, 我们只关注函数 createTabs(), 剩余的函数之后再来讨论。

swapping/swapping.js

```
function createTabs(container){
  $(container).prepend("<ul class='tabs'></ul>");
  $(container).children("div.example").each(function(){
    var exampleTitle = $(this).attr('class').replace('example', '');
    $(container).children("ul.tabs").append(
      "<li class='tab '+exampleTitle+'>"+exampleTitle+"</li>"
    );
  });
}
```

首先, 创建一个无序列表 (<ul>) 来放置 Tab, 并伪装其为文本实例的容器。

然后, 遍历容器中每一个实例, 这些实例拥有两个 class, 各自的标题和 “examples”。我们只需要标题, 因此通过.attr('class') 获得 class, 并替换 example 为空, 留下每个将显示在 Tab 中的实例的标题, 然后在之前创建的无序列表的 <li> 标签中放置标题。

如果现在在浏览器中打开此页面，什么都不会发生，因为 `styleExamples()` 函数并没有被调用，所以 JavaScript 没有被执行。让我们来看看下一步。

## Tab 之间的切换

我们的内容被转化为 Tab 交互，但还无法让用户在不同的 Tab 之间进行切换。为了修复这一问题，在页面加载的时候需要调用 `styleExamples()` 函数，这一函数会将各包含实例的 `<div>` 标签转变成 Tab 交互：

```
swapping/swapping.js
$(function(){
    styleExamples();
});
```

如果现在在浏览器中加载页面，将看到一个无序列表，含有“english”和“latin”两项内容。到目前为止，进展还不错，但还需要写一个函数来显示不同实例里的内容。首先隐藏所有实例，然后显示想要看到的那一个。

```
swapping/swapping.js
function displayTab(element){
    tabTitle = $(element)
        .attr('class')
        .replace('tab', '')
        .replace('selected', '').trim();

    container = $(element).parent().parent();
    container.children("div.example").hide();
    container.children("ul.tabs").children("li").removeClass("selected");

    container.children("div."+tabTitle).slideDown('fast');
    $(element).addClass("selected");
}
```

将变量 `tabTitle` 的 class 从“tab”、“selected”、“english”缩减为只剩“english”，并用于寻找正确的 `<div>` 进行显示。现在，我们需要从视野中把一切移除。

```
swapping/swapping.js
container = $(element).parent().parent();
container.children("div.example").hide();
container.children("ul.tabs").children("li").removeClass("selected");
```

我们隐藏了容器内所有 class 为“example”的 `<div>`，还移除了所有 `<li>` 标签 class 中的“selected”，即使它们根本没有。

之所以这样处理是为了相对简单，之后一次只需要处理一个元素，而不是处理每一个，同时也提高了代码的可读性。现在，我们已经准备好展示一个满足需求的例子了。

```
swapping/swapping.js
container.children("div."+tabTitle).slideDown('fast');
$(element).addClass("selected");
```

现在看一下例子，通过函数传入的 `class` 名称来匹配寻找用于显示的 `<div>`，显示方式使用了 jQuery 提供的 `slideDown()` 函数。我们还可以尝试使用 jQuery UI 提供的 `.show()`、`.fadeIn()` 或者其他提供动画效果的函数。最后，设置当前 `<li>` 的 `class` 为 “selected”，使得 CSS 明确哪个 Tab 需要显示。

现在，已经有了函数 `displayTab()`，但尚未使用。为了在实例间切换，点击实例标题需要触发 `displayTab()` 的调用。

```
swapping/swapping.js
function activateTabs(element){
    $(element).children("ul.tabs").children("li").click(function(){
        displayTab(this);
    });
}
```

简言之，容器的任务就是定位从函数 `createTabs()` 创建的 `<li>` 标签，设置它们被点击时调用函数 `displayTab()`。

## 设置 Tab 样式

最后，回到函数 `styleExamples()`，我们将看到页面加载时会如何调用一系列函数，构建 Tab 的样式。

```
swapping/swapping.js
function styleExamples(){
    $("div.examples").each(function(){
        createTabs(this);
        activateTabs(this);
        displayTab($(this).children("ul.tabs").children().first());
    });
}
```

最后调用函数 `displayTab()` 的时候设置第一个 Tab 为默认 Tab，在页面加载完成后隐藏其他 Tab 仅显示此项。

至此，已经完成了所有功能，不妨应用一点 CSS 来让最后的界面看上去更美观。



```
swapping/swapping.css
li.tab {
    color: #333;
    cursor: pointer;
    float: left;
    list-style: none outside none;
    margin: 0; padding: 0;
    text-align: center;
    text-transform: uppercase;
    width: 80px;
    font-size: 120%;
    line-height: 1.5;
    background-color: #DDD;
}

li.tab.selected {
    background-color: #AAA;
}

ul.tabs {
    font-size: 12px;
    line-height: 1;
    list-style: none outside none;
    margin: 0;
    padding: 0;
    position: absolute;
    right: 20; top: 0;
}

div.example {
    font-family: "Helvetica", "san-serif";
    font-size: 16px;
}

div.examples {
    border: 5px solid #DDD;
    font-size: 14px;
    margin-bottom: 20px;
    padding: 10px;
    padding-top: 30px;
    position: relative;
    background-color: #000;
    color: #DDD;
}
```

大功告成，我们现在有了通用代码，可以用来构建一个真正的网站，能够很容易地切换不同语言的产品介绍。

该解决方案节省了相当多空间，因此经常被使用在空间有限的网站上。一般而言，这项技术被用来以 Tab 方式展示产品信息、评论和各种相关信息，但在 JavaScript 不可用的时候，仍需使用线性布局使得所有信息可见。

## 深入研究

### Further Exploration

如果需要始终页面上加载一个特定的 Tab 呢？例如，在页面上展示 Ruby、Python 和 Java 的代码实例，而网站的一个用户想要看 Python 的例子，如果他每次访问新页面的时候不需要重新点击 Python Tab 将是非常好的用户体验。这一问题留给读者进行研究。

## 另请参考

### Also See

- 8 号秘方 可访问的展开和折叠