Web开发典藏大系

jQuery网页开发实例精解

13.5小时多媒体教学视频

黄格力　等编著

jQuery中文社区鼎力推荐

立足实战，以短小精悍的代码解析jQuery开发技巧
通过实际应用，体验jQuery在Web开发中的无穷魅力
通过93个实例剖析、10个插件使用详解，提升实战技能

清华大学出版社

jQuery
网页开发实例精解

《jQuery网页开发实例精解》一书用通俗易懂的实例、简单生动的文字，引领jQuery初学者进入网页交互开发的殿堂，是一本适合广大jQuery初学者入门的经典好书，值得阅读！

——jQuery中文社区（http://jquery.org.cn）

本书涵盖的内容及视频时间

- ◎ 了解jQuery（31分钟视频）
- ◎ jQuery原理与运行机制（87分钟视频）
- ◎ DIV层的控制（85分钟视频）
- ◎ 设计列表（41分钟视频）
- ◎ 网站导航（50分钟视频）
- ◎ 设计表格（66分钟视频）
- ◎ 设计表单（116分钟视频）
- ◎ 设计图片（55分钟视频）
- ◎ 设计对话框（53分钟视频）
- ◎ 设计滑动条（64分钟视频）
- ◎ 页面编辑器插件（22分钟视频）
- ◎ 多媒体（35分钟视频）
- ◎ 动画设计（38分钟视频）
- ◎ 拖放插件（25分钟视频）
- ◎ 插件开发（39分钟视频）

下载提示

本书提供配套的教学PPT，需要的读者请在清华大学出版社网站（http://www.tup.com.cn）上搜索到本书页面后下载。

Web开发典藏大系

（部分图书封面效果展示）

本系列图书涵盖Web开发的各种热门技术，陆续推出，敬请关注，好书不容错过！

清华大学出版社数字出版网站
WQBook 书文局泉
www.wqbook.com

◎ 本书配套教学视频
◎ 本书源文件

上架：计算机/Web开发

ISBN 978-7-302-28100-9

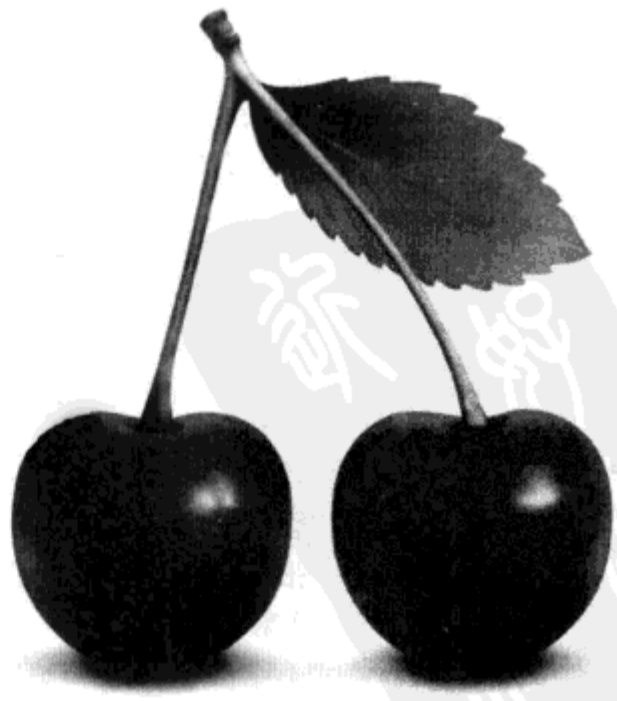
附DVD光盘1张

Web开发典藏大系

jQuery 网页开发实例精解

13.5小时多媒体教学视频

黄格力　等编著



清华大学出版社

北　京

内 容 简 介

本书是一本介绍 jQuery 网页开发基础应用的书籍，是一本简单易懂，把复杂问题简单化的书。本书立足于 jQuery 入门基础技术，对 jQuery 的产生、jQuery 基础、如何将 jQuery 应用于页面开发、如何产生页面元素特效等都进行了细致入微的讲解。这必将使每一个阅读本书的读者少走弯路，快速上手，建立用 jQuery 进行开发网页的信心。

本书共 15 章，分 2 篇。第 1 篇 jQuery 基础理论，主要介绍 JavaScript 脚本、HTML DOM、DIV 的控制等知识，这些内容都是学习后续章节所必须掌握的基础知识。第 2 篇 jQuery 实战开发与应用，主要介绍如何利用 jQuery 控制页面元素；如何使页面元素产生特殊效果；不同功能插件的使用方法及注意事项；如何编写 jQuery 插件，以帮助读者理解、使用和开发插件。

本书配带 1 张光盘，收录了本书涉及的源代码和书中重点内容的教学视频。

本书适合没有基础的 jQuery 入门新手阅读；对于有一定基础的读者，可通过本书进一步理解 jQuery 的各个重要知识点和概念；对于大、中专院校的学生和培训班的学员，本书不失为一本好教材。

图书在版编目（CIP）数据

jQuery 网页开发实例精解 / 黄格力等编著. —北京：清华大学出版社，2012.7
（Web 开发典藏大系）
ISBN 978-7-302-28100-9

Ⅰ. ①j…　Ⅱ. ①黄…　Ⅲ. ①Java 语言－程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 030481 号

责任编辑：夏兆彦
封面设计：欧振旭
责任校对：徐俊伟
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：22.75　　**字　　数**：571 千字
（附光盘 1 张）
版　　次：2012 年 7 月第 1 版　　**印　　次**：2012 年 7 月第1 次印刷
印　　数：1～5000
定　　价：49.80 元

产品编号：045286-01

前　言

“工欲善其事，必先利其器”。作为一名从事 Web 开发多年的工作者，我对每一种新技术的出现与应用都充满了渴望与期待，渴望它能解决现存疑难，进一步提高程序开发的效率；期待它能超越旧俗，引领技术未来的发展方向。近年来，Web 开发领域的新技术和新工具层出不穷，它们的出现极大地推动了 Web 开发技术的发展。其中，jQuery 的诞生在 Web 技术发展进程中具有划时代的意义。

1．jQuery脱颖而出

jQuery 发布于 2006 年，因为它易于使用、功能强大、展现优雅、兼容性极佳而迅速赢得了 Web 开发者的钟爱，不断地吸引着全球开发者社区的技术爱好者、精英和专家们加入其阵营。这也使得它在众多的 JavaScript 框架中脱颖而出，几近成为 Web 开发领域的事实标准。恰好是在 2006 年，jQuery 也深深地吸引了我，直至今日，我还醉心于它。

随着 Web 开发技术的发展及用户对应用体验要求的日益提高，程序员开发一个 Web 应用时，不仅要考虑其功能是否足够完备，更重要的是要考虑如何做才能提高用户的体验满意度。这是理性的回归，也是 Web 开发技术发展的必然趋势。而 jQuery 恰恰是满足这一理性需求的坚实利刃。

2．jQuery容易上手

5 年前我认识了 jQuery，从此一发不可收拾。jQuery 给我的第一印象非常好，它非常容易上手，大部分思想都是从 HTML 和 CSS 的结构中借鉴而来的，所以编程经验不多的设计师也能够快速学会如何使用，我经常因为 jQuery 做出的各种动态效果而兴奋不已，常常惊叹于 jQuery 如此小巧，却有如此强大的功能。

jQuery 是众多脚本框架库中的一员，它以轻量型、编写简单、易上手等优点吸引着前台设计开发人员。随着 jQuery 功能的不断改进，它在轻量型前台脚本框架中的应用将会越来越多。

3．本书的诞生

Web 2.0 的出现使用户对于网页的效果有了全新的体验。随之而来的众多前台脚本框架脱颖而出，它们各有优势，并且在功能上还在不断更新。

虽然 jQuery 使用简单，但它毕竟是一门新的技术，与传统的 JavaScript 在性能与语法上存在诸多差异，需要相应的书籍来引导开发者迅速而有效地掌握它，并能真正付诸实践。但目前已经出版的中文类 jQuery 图书，多是简单的概念性介绍，缺乏真正的实践指导，而且版本相对陈旧。为了让所有希望掌握 jQuery 技术的开发者能迅速步入 jQuery 殿堂，本

书诞生了，相信它不会让读者失望。

本书以讲解 jQuery 的开发基础与应用为主导，配以各种实际应用，详解设计思路和设计步骤，向读者展示了 jQuery 如何应用于网页前台的开发。书中还介绍了大量的插件使用知识，以帮助读者更灵活、快速地利用 jQuery 进行开发。

本书特色

1. 内容的讲解不枯燥

本书不以枯燥的理论来解释技术知识点，而是理论与实例相辅相成，融入笔者在实际项目开发中的经验，介绍最常见的应用和技巧，容易理解并掌握。

2. 本书覆盖面广

本书基本包含了 jQuery 技术所有常用知识点，包含了对最新的 jQuery 插件应用的介绍，即使是初学者通过阅读本书，也可以开发出 Web 2.0 风格的网站。

3. 循序渐进，由浅入深

阅读本书不需要对 jQuery 有所了解，甚至不需要对 JavaScript 有很深的了解。为了方便读者学习，本书首先介绍 Web 开发的基础知识，如 JavaScript 的基础语法、DOM、BOM 等，以期读者可以在不参考其他资料的情况下顺利过渡到 jQuery 的学习和使用。

4. 真实案例，随学随用

本书注重实践，通过若干比较完整的案例来讲解 jQuery 在具体项目中的使用，如从最简单的 DIV 控制到滑动条效果，都是从最简单的示例开始讲解，理论联系实际代码。读者通过这些实例会对 jQuery 的理论知识有更加深刻的理解，同时稍作修改即可用于实际项目开发，实现各种精彩的效果。

5. 语言通俗易懂

本书在讲解各个案例、知识点时，语言通俗易懂，非常适合入门级读者及广大爱好者学习。

6. 图示丰富，容易理解

本书采用多插图的形式来展现 jQuery 的效果。通过前后的对比图，读者能很快掌握知识点。

本书内容及体系结构

第1篇　jQuery基础理论（第1～2章）

本篇介绍 jQuery 开发的基础知识，包括 jQuery 概述、搭建 jQuery 运行环境、JavaScript

的浏览器对象模型 BOM 操作、JavaScript 的 HTML 文档 DOM 操作、Ajax 原理、jQuery 工作原理、jQuery 运行机制等。这些内容都是学习后续章节所必须掌握的基础知识。

第2篇　jQuery实战开发与应用（第3～15章）

本篇主要介绍 jQuery 的实际开发和应用，包括 DIV 层的控制、设计列表、网站导航、设计表格、设计表单、设计图片、设计对话框、设计滑动条、页面编辑器插件、多媒体、动画设计、拖动插件、插件开发等内容。这些内容覆盖了利用 jQuery 控制页面元素、制作特效及插件的使用与开发。

本书读者对象

- 没有基础的 jQuery 初学者；
- Web 开发人员；
- jQuery 开发爱好者；
- 刚入职的初、中级程序员；
- 大、中专院校师生；
- 相关培训学校的学员。

本书作者

本书由黄格力主笔编写。其他参与编写的人员有陈世琼、陈欣、陈智敏、董加强、陈杰、陈冠军、项宇峰、张帆、陈刚、程彩红、毛红娟、聂庆亮、王志娟、武文娟、颜盟盟、姚志娟、尹继平、张昆、张薛。

编著者

目　录

第 1 篇　jQuery 基础理论

第 1 章　了解 jQuery（教学视频：31 分钟） ····· 2
1.1　认识 jQuery ····· 2
1.1.1　jQuery 的起源 ····· 2
1.1.2　什么是 jQuery ····· 4
1.2　jQuery 能做什么 ····· 5
1.2.1　jQuery 能实现什么 ····· 5
1.2.2　jQuery 与其他脚本库的区别 ····· 7
1.3　搭建 jQuery 运行环境 ····· 8
1.3.1　jQuery 库的选择 ····· 8
1.3.2　jQuery 库的引入 ····· 9
1.3.3　jQuery 的第一个示例 ····· 11
1.3.4　如何学习 jQuery ····· 12
1.4　小结 ····· 12

第 2 章　jQuery 原理与运行机制（教学视频：87 分钟） ····· 13
2.1　JavaScript 的浏览器对象模型 BOM 操作 ····· 13
2.1.1　Window 对象——窗口对象 ····· 14
2.1.2　Navigator 对象——浏览器对象 ····· 24
2.2　JavaScript 的 HTML 文档 DOM 操作 ····· 29
2.2.1　什么是 DOM ····· 29
2.2.2　DOM 节点 ····· 29
2.2.3　访问文档节点 ····· 30
2.2.4　节点操作 ····· 33
2.3　Ajax 原理 ····· 40
2.3.1　Ajax 组成 ····· 40
2.3.2　Ajax 与基本 Web 应用工作比较 ····· 40
2.3.3　Ajax 核心对象 XMLHTTPRequest ····· 42
2.3.4　Ajax 工作示例 ····· 42
2.4　jQuery 工作原理 ····· 44

2.5 jQuery 运行机制……45
2.5.1 jQuery 的元素选择……45
2.5.2 jQuery 事件……47
2.6 小结……50

第 2 篇 jQuery 实战开发与应用

第 3 章 DIV 层的控制（教学视频：85 分钟）……52
3.1 DIV 的鼠标选取……52
3.1.1 利用鼠标悬停实现 DIV 的选取……52
3.1.2 利用鼠标单击实现 DIV 的选取……54
3.2 DIV 层的尺寸……55
3.2.1 jQuery 动态读取 DIV 层的尺寸……55
3.2.2 jQuery 动态修改 DIV 层的尺寸……56
3.3 层的显示与隐藏……59
3.3.1 利用 jQuery 的显示与隐藏函数实现……59
3.3.2 利用 jQuery 的滑动效果实现……62
3.3.3 利用 jQuery 的淡入淡出效果实现……64
3.4 DIV 内的内容控制……66
3.4.1 内容清空……66
3.4.2 内容替换……67
3.4.3 内容复制……68
3.4.4 内容添加……69
3.4.5 内容包装……73
3.5 层的定位……74
3.6 小结……74
3.7 习题……75

第 4 章 设计列表（教学视频：41 分钟）……76
4.1 控制列表宽度……76
4.1.1 参差不齐的列表……76
4.1.2 截取文字内容实现控制列表宽度……77
4.1.3 修改层的宽度控制列表宽度……78
4.2 控制列表项符号图片……81
4.2.1 样式死板的列表项符号……82
4.2.2 利用 jQuery 与 CSS 控制列表项符号图片……82
4.3 列表项的滚动……83
4.4 图片列表……86
4.4.1 大小不一的图片不规则排列……86

4.4.2　利用 jQuery 控制图片列表 …… 86
4.5　列表的显示与收缩 …… 88
4.5.1　占用页面空间的静态列表 …… 88
4.5.2　利用 jQuery 动态控制列表内容展开与收缩 …… 89
4.6　列表项动态排序 …… 91
4.6.1　构建一个无序列表 …… 91
4.6.2　利用 jQuery 对无序列表排序 …… 92
4.7　小结 …… 94
4.8　习题 …… 94

第 5 章　网站导航（教学视频：50 分钟） …… 95
5.1　菜单设计 …… 95
5.1.1　普通下拉菜单 …… 95
5.1.2　下拉级联菜单 …… 98
5.1.3　横向伸缩菜单 …… 100
5.2　第三方菜单插件 …… 101
5.2.1　jQuery 级联菜单插件 …… 101
5.2.2　SuperFish 菜单插件 …… 103
5.2.3　折叠菜单插件 …… 106
5.2.4　滚动动态列表菜单插件 …… 107
5.2.5　滑动效果菜单 …… 108
5.2.6　仿 Mac 的停靠菜单插件 …… 109
5.3　TreeView 设计 …… 111
5.3.1　普通 TreeView …… 111
5.3.2　加入淡入淡出效果的 TreeView …… 112
5.4　第三方 TreeView 插件 …… 114
5.5　小结 …… 117
5.6　习题 …… 118

第 6 章　设计表格（教学视频：66 分钟） …… 119
6.1　表格基本设计 …… 119
6.1.1　表格边框样式的变换 …… 119
6.1.2　表格单元格的合并 …… 120
6.1.3　表格行列的添加与删除 …… 123
6.1.4　jQuery 控制表格行的上下移动 …… 125
6.2　表格内容动态排序 …… 128
6.3　设置分页 …… 130
6.4　表格行条纹效果 …… 131
6.5　表格的折叠和展开 …… 132
6.6　表格动态内容筛选 …… 134
6.7　可编辑表格 …… 135

6.8 表格插件 …… 137
6.8.1 jExpand 表格插件 …… 137
6.8.2 Table Pagination 表格分页插件 …… 138
6.8.3 Spreadsheet Web 电子表格插件 …… 139
6.9 小结 …… 140
6.10 习题 …… 141

第 7 章 设计表单（教学视频：116 分钟） …… 142
7.1 表单基本操作 …… 142
7.1.1 表单清空 …… 142
7.1.2 重置表单 …… 144
7.1.3 表单元素的赋值与取值 …… 145
7.2 表单验证框架 …… 146
7.2.1 基本验证功能 …… 147
7.2.2 API 使用方法 …… 149
7.2.3 自定义验证方法 …… 155
7.2.4 radio、checkbox、select 的验证 …… 157
7.3 表单特效 …… 159
7.3.1 文本输入框特效 …… 159
7.3.2 单选按钮、复选框特效 …… 163
7.3.3 按钮特效 …… 163
7.4 表单插件 …… 165
7.4.1 Validation 插件 …… 165
7.4.2 JQF1 插件 …… 173
7.5 小结 …… 175
7.6 习题 …… 175

第 8 章 设计图片（教学视频：55 分钟） …… 176
8.1 图片切换 …… 176
8.1.1 利用淡入效果实现图片切换 …… 176
8.1.2 利用自定义动画切换图片 …… 177
8.2 图片滚动 …… 179
8.3 图片动态弹出 …… 181
8.4 动态图文结合 …… 183
8.5 图片剪切 …… 184
8.6 图片预览 …… 186
8.7 图片局部平移 …… 187
8.8 图片插件 …… 189
8.8.1 MobilyNotes 插件 …… 189
8.8.2 Fancybox 插件 …… 190
8.8.3 desSlideshow 插件 …… 195

8.9　小结 …… 197
8.10　习题 …… 197

第 9 章　设计对话框（教学视频：53 分钟） …… 198
9.1　设计基本对话框 …… 198
9.2　模式对话框 …… 199
9.3　输入对话框 …… 201
9.4　提示对话框 …… 202
9.5　确认对话框 …… 204
9.6　对话框插件 …… 206
9.6.1　jqModal 对话框插件 …… 206
9.6.2　jQuery.UI.Dialog 对话框插件 …… 212
9.7　小结 …… 222
9.8　习题 …… 222

第 10 章　设计滑动条（教学视频：64 分钟） …… 223
10.1　自定义滑动条 …… 223
10.2　滑动条插件 …… 225
10.2.1　jQuery.UI.Slider 插件 …… 225
10.2.2　jScrollPane 插件 …… 236
10.3　小结 …… 247
10.4　习题 …… 247

第 11 章　页面编辑器插件（教学视频：22 分钟） …… 248
11.1　markItUp 插件 …… 248
11.1.1　安装插件 …… 248
11.1.2　键盘的使用 …… 249
11.1.3　标记语言定制 …… 249
11.2　jwysiwyg 插件 …… 253
11.2.1　jwysiwyg 插件基本介绍 …… 253
11.2.2　基本应用 …… 255
11.2.3　全功能编辑器 …… 256
11.2.4　可调整大小的编辑器 …… 260
11.2.5　编辑器内容的左右移动 …… 261
11.2.6　根据内容自动调整大小的编辑器 …… 261
11.2.7　自定义功能控制按钮 …… 263
11.3　HtmlBox 插件 …… 265
11.3.1　如何安装 …… 266
11.3.2　属性特点 …… 266
11.3.3　内置函数 …… 267
11.3.4　编辑器可使用的工具 …… 267

11.3.5 插件应用举例 …… 268
11.4 Lightweight RTE 插件 …… 271
11.5 小结 …… 273
11.6 习题 …… 273

第 12 章 多媒体（教学视频：35 分钟） …… 274
12.1 jQuery.Flash 插件 …… 274
12.1.1 基本 Flash 文件嵌入 …… 274
12.1.2 Flash 替换文本内容 …… 275
12.1.3 MP3 播放示例 …… 276
12.1.4 使用内联样式设定插件播放 …… 277
12.2 jPlayer 插件 …… 278
12.2.1 jPlayer 插件基本介绍 …… 278
12.2.2 jPlayer 插件基本使用方式：播放音频文件 …… 280
12.2.3 jPlayer 插件基本使用方式：播放视频文件 …… 282
12.2.4 自定义播放器操作 …… 283
12.3 jlEmbed 插件 …… 291
12.3.1 插件基本使用 …… 292
12.3.2 使用 Windows Media Player 播放 …… 292
12.3.3 使用 QuickTime 播放 …… 293
12.3.4 使用 SilverLight 播放 …… 293
12.3.5 使用 Flash Player 播放 …… 293
12.4 小结 …… 294
12.5 习题 …… 294

第 13 章 动画设计（教学视频：38 分钟） …… 295
13.1 jQuery 动画基础 …… 295
13.1.1 jQuery 动画函数 …… 295
13.1.2 jQuery 动画简单示例 …… 298
13.2 jQuery UI 中实现的动画效果 …… 299
13.2.1 jQuery UI 动画特效使用 …… 299
13.2.2 百叶窗效果 …… 301
13.2.3 跳跃效果 …… 302
13.2.4 缩减效果 …… 305
13.2.5 移动效果 …… 306
13.2.6 分裂效果 …… 307
13.2.7 折叠效果 …… 309
13.2.8 高亮淡入淡出效果 …… 311
13.2.9 脉冲闪烁效果 …… 313
13.2.10 摆动效果 …… 314
13.3 小结 …… 316

13.4　习题 316

第 14 章　拖动插件（教学视频：25 分钟） 317

14.1　jQuery UI 拖动插件 317

14.1.1　jQuery UI 拖动插件基本介绍 317

14.1.2　jQuery UI 拖动插件示例 318

14.2　jQuery UI 投放插件 328

14.2.1　jQuery UI 投放插件基本介绍 328

14.2.2　jQuery UI 投放插件示例 328

14.3　小结 333

14.4　习题 333

第 15 章　插件开发（教学视频：39 分钟） 334

15.1　jQuery 插件开发基础 334

15.1.1　jQuery 插件介绍 334

15.1.2　jQuery 插件开发基础知识 335

15.1.3　创建一个简单 jQuery 插件 335

15.2　插件示例 343

15.3　插件开发规范 349

15.4　小结 350

15.5　习题 350

第 1 篇　jQuery 基础理论

- 第 1 章　了解 jQuery
- 第 2 章　jQuery 原理与运行机制

第 1 章　了解 jQuery

本章是全书的开篇，主要讲解 jQuery 的基本入门知识及 jQuery 的相关特点。随着互联网的迅速发展，在前台页面的用户体验需求越来越高。虽然 JavaScript 这种动态语言极大的灵活性导致了项目中每个人的代码风格截然不同，但是其在功能和浏览器的兼容性上并不能实现高标准、严要求。正是在这种情形下出现了 jQuery，它能够帮助我们实现各种酷炫的页面效果而不用担心浏览器的兼容性。

1.1　认识 jQuery

本节主要介绍 jQuery 的起源和 jQuery 到底是什么。

1.1.1　jQuery 的起源

jQuery 的起源要从 JavaScript 说起。JavaScript 是网景公司在它自己的 LiveScript 基础上产生的。JavaScript 的出现是前台脚本语言发展的一个里程碑。它是一种基于对象的事件驱动的解释性语言。它的实时性、跨平台、开发使用简单并且相对安全的特点决定了它在 Web 前台设计中的重要地位。

但是，随着浏览器种类的推陈出新，JavaScript 的兼容性受到了挑战。对前台设计效果的要求越来越高，于是，JavaScript 语言本身的设计能力变得捉襟见肘。2006 年，美国人 John Resig 创建了 JavaScript 的另一个框架，它就是 jQuery。

与 JavaScript 相比，jQuery 更简洁，浏览器的兼容性更强，语法更灵活，对 Xpath 的支持更强大，一个$符就可以遍历文档中的各级元素。下面用一个示例来具体比较一下。

具体需求是这样的：在页面上有一个无序列表，我们需要将所有列表项中的文本内容提取出来并显示。效果如图 1.1 所示。

首先，我们来看一下 JavaScript 的实现。

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 </head>
6 <body>
7   <ul>
8       <li>List Item 1</li>
9       <li>List Item 2</li>
10       <li>List Item 3</li>
11       <li>List Item 4</li>
```

```
12        <li>List Item 5</li>
13    </ul>
14    <script type="text/javascript">
15      var listitems=document.getElementsByTagName("li");
                                                    //获取所有列表项组成的数组
16      var str="";                                   //定义保存文本内容的变量
17      for(i=0;i<listitems.length;i++)               //循环数组元素
18          str+=listitems[i].firstChild.nodeValue;//提取列表项的文本内容
19      alert(str);                                   //显示所有文本内容
20  </script>
21</body>
22</html>
```

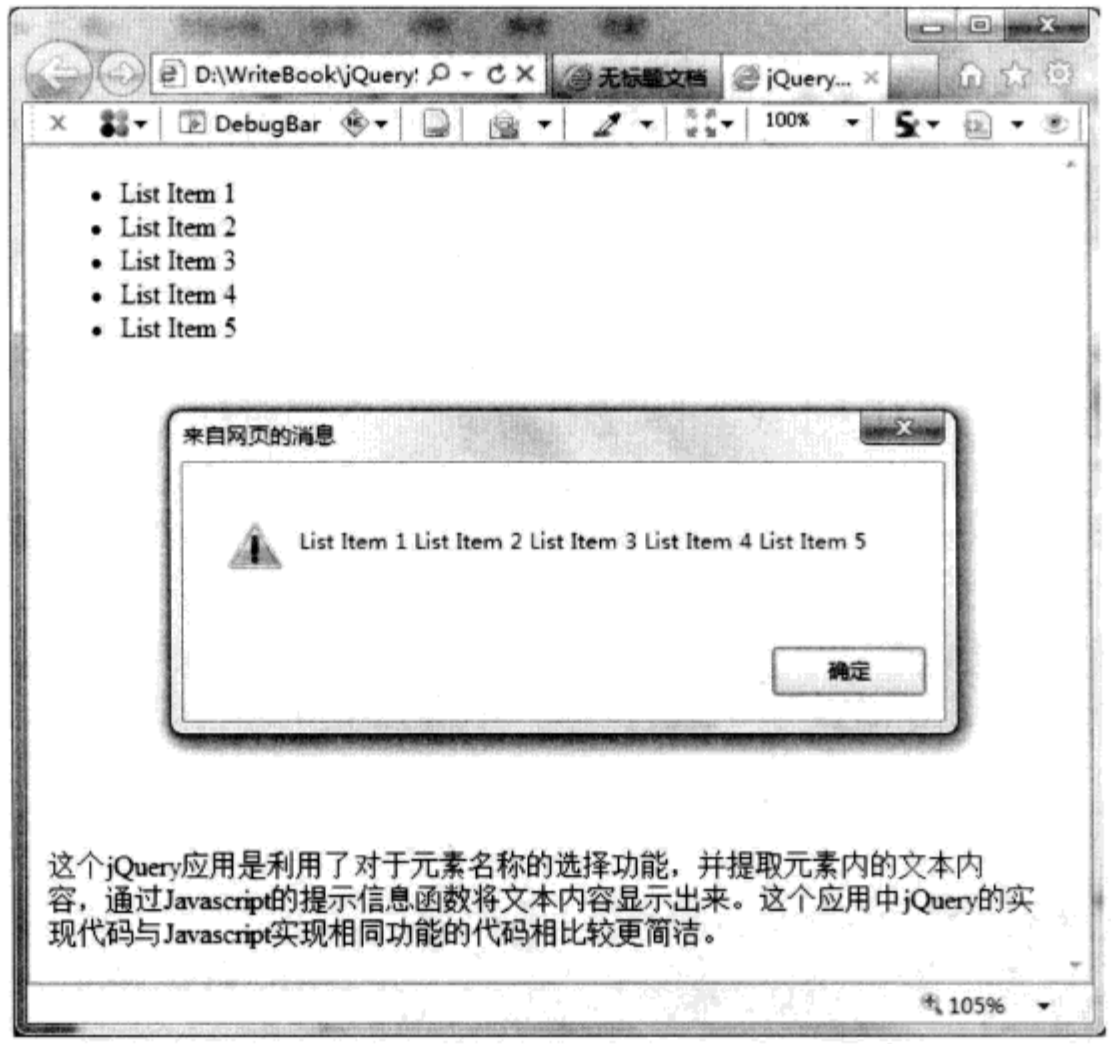


图 1.1　JavaScript 与 jQuery 的代码比较

通过上面的代码可以了解 JavaScript 实现这个功能的基本步骤：获取所有元素对象，通过遍历这些元素对象得到文本内容，然后显示。下面看一下 jQuery 的实现。

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <script type="text/javascript" src="jslib/jquery-1.6.js"></script>
6 <script type="text/javascript">
7   $(function(){alert($("li").text());});     //获取所有列表项的文本并输出
8 </script>
9 </head>
10<body>
11  <ul>
12      <li>List Item 1</li>
13       <li>List Item 2</li>
14       <li>List Item 3</li>
```

```
15        <li>List Item 4</li>
16        <li>List Item 5</li>
17    </ul>
18</body>
19</html>
```

两段代码相比较，可以发现使用 jQuery 处理简洁很多，代码量明显减少。jQuery 实现中的第 7 行代码就是 jQuery 实现功能部分。

1.1.2　什么是 jQuery

jQuery 是什么？在 jQuery 官方网站上是这样解释的：jQuery 是一个快速简洁的 JavaScript 库，它可以简化 HTML 文档的元素遍历、事件处理、动画及 Ajax 交互，快速开发 Web 应用。它的设计是为了改变 JavaScript 程序的编写风格。jQuery 特点如下。

（1）轻量型，jQuery 是一个轻量型框架，程序短小，配置简单。

（2）DOM 选择，可以轻松获取任意 DOM 元素或 DOM 元素封装后的 jQuery 对象。

（3）CSS 处理，可以轻松设置、删除、读取 CSS 属性。

（4）链式函数调用，可以将多个函数链接起来被一个 jQuery 对象一次性调用。

（5）事件注册，可以对一个或多个对象注册事件，让画面和事件分离。

（6）对象克隆，可以克隆任意对象及其组件。

（7）Ajax 支持，跨浏览器，支持 Internet Explorer 6.0+、Opera 9.0+、Firefox 2+、Safari 2.0+、Google Chrome 1.0+。

如图 1.2 所示是 jQuery 的官方网站截图。可以在这个网站上获取各种版本的 jQuery 库文件及官方插件，学习 jQuery，并且还可以提交在使用 jQuery 的过程中所发现的 Bug。

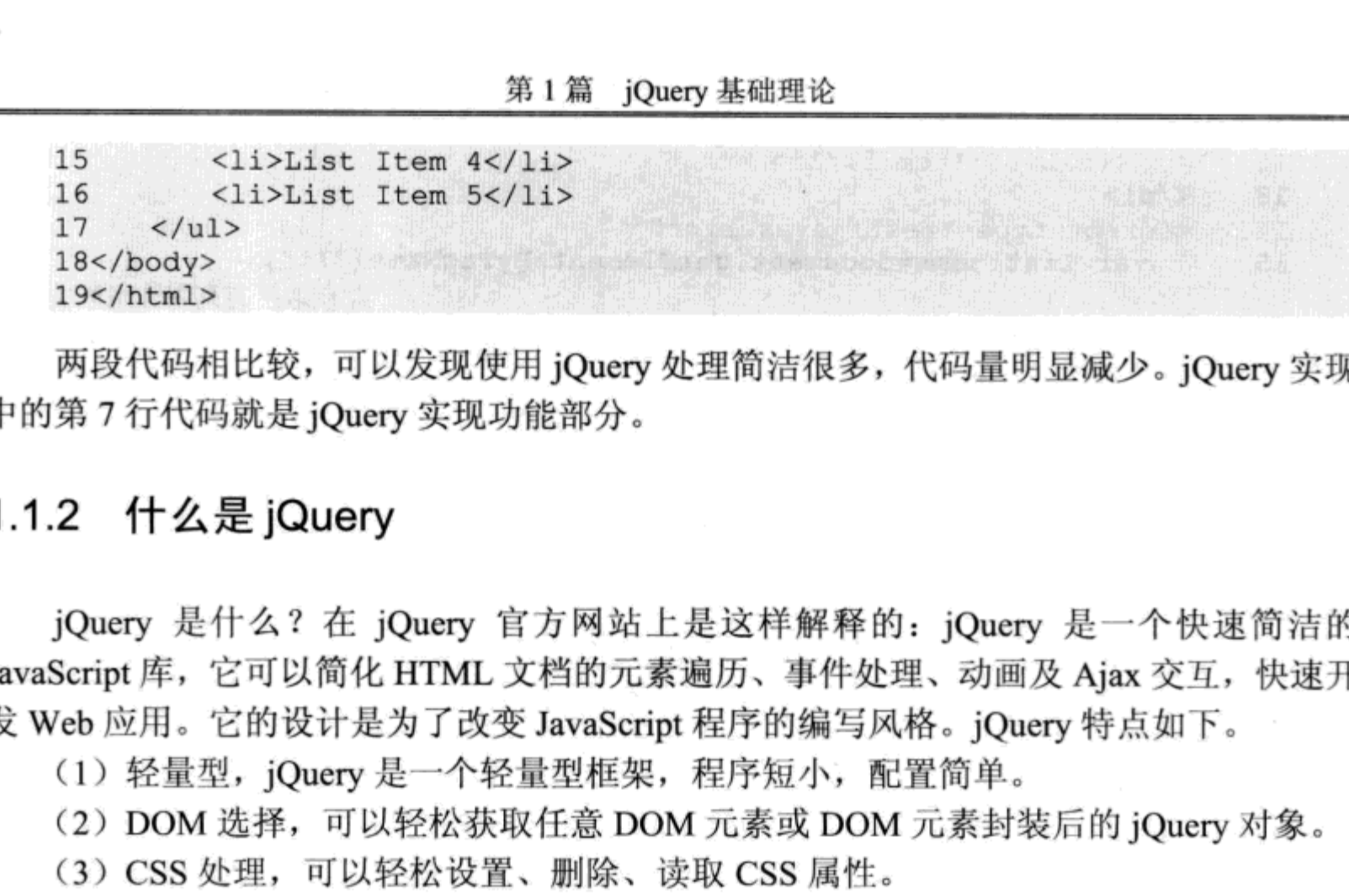


图 1.2　jQuery 官方网站的截图

1.2　jQuery 能做什么

上一节认识了什么是 jQuery，以及它的特点和来历。本节来看一下 jQuery 可以做什么。

1.2.1　jQuery 能实现什么

jQuery 库为 Web 脚本编程提供了通用的抽象层，使得它几乎适用于任何脚本编程的情形。由于它易于扩展而且不断有新插件面世来增强其功能，因此，一本书根本无法涵盖它所有可能的用途和功能。抛开这些不谈，仅就其核心特性而言，jQuery 能够满足下列需求。

（1）取得页面中的元素。通过上一节的示例可以发现，通过一条 jQuery 语句就可以获取页面中相同标记名的所有元素。

（2）修改页面的外观。在 jQuery 的众多功能函数中，有专门修改 CSS 样式设定的函数，通过这些函数可以动态修改页面外观。例如，可以在页面中通过下拉列表框选择动态修改一个 DIV 的背景色。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<script type="text/javascript" src="jslib/jquery-1.6.js"></script>
<script type="text/javascript">
  $(function(){
     $("#choice").click(function(){                    //下拉列表框的单击事件
        $("#div1").css({background:$("#color").val()});
                                                       //更改 Div 层的背景色的值
     });
  });
</script>
</head>
<body>
<center>
  <div id="div1" style="width:200px;height:200px;border:black 1px
  dotted"></div>
    <select id="color">
      <option value="white">默认</option>
       <option value="black">黑色</option>
       <option value="gray">灰色</option>
       <option value="orange">橙色</option>
    </select><br />
    <input type="button" id="choice" value="更改背景色" />
</center>
</body>
</html>
```

示例实现效果如图 1.3 和图 1.4 所示。

图 1.3　修改 DIV 背景色一

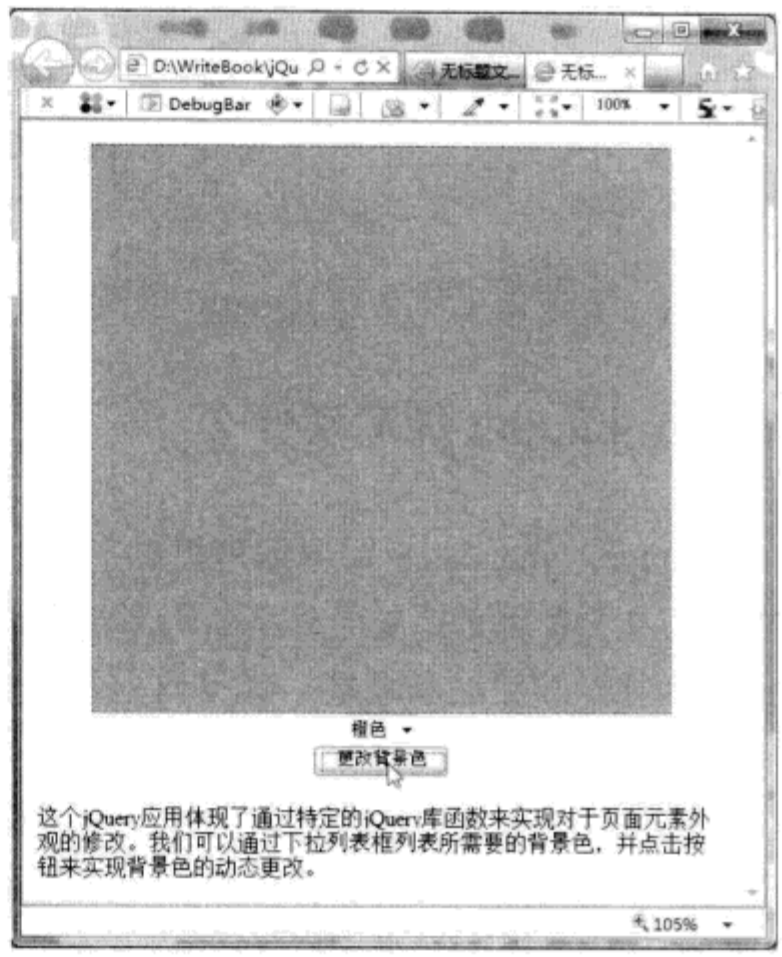


图 1.4　修改 DIV 背景色二

（3）改变页面的内容。jQuery 能够影响的范围并不局限于简单的外观变化。使用少量的代码，jQuery 就能改变文档的内容。它还可以改变文本，插入或翻转图像，对列表重新排序，甚至对 HTML 文档的整个结构都能重写和扩充——所有这些只需一个简单易用的 API。下面看一个通过 DIV 单击修改文本内容的例子。

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <script type="text/javascript" src="jslib/jquery-1.6.js"></script>
6 <script type="text/javascript">
7   $(function(){
        //创建 DIV 的鼠标点击触发事件
8       $("#div1").toggle(function(){$(this).text("jQuery 能够影响的范围并不局限于简单的外观变化，使用少量的代码，jQuery 就能改变文档的内容。可以改变文本、插入或翻转图像、对列表重新排序，甚至，对 HTML 文档的整个结构都能重写和扩充—所有这些只需一个简单易用的 API。");},
9       function(){$(this).text("在 jQuery 的众多功能函数中，有专门修改 CSS 样式设定的函数，通过这些函数我们可以动态修改页面外观。");});
10  });
11</script>
12</head>
13<body>
14  <center>
15      <div id="div1" style="width:400px;height:400px;border:black 1px dotted; font-family:Verdana, Geneva, sans-serif; font-size:36px; color:#2A0055">在 jQuery 的众多功能函数中，有专门修改 CSS 样式设定的函数，通过这些函数我们可以动态修改页面外观。</div>
16    </center>
17    <p>这个 jQuery 应用体现了通过特定的 jQuery 库函数来实现对于页面元素文本的修改。我们可以通过点击文本显示区域来实现页面元素文本的动态更改。</p>
```

```
18</body>
19</html>
```

示例的实现效果如图 1.5 和图 1.6 所示。

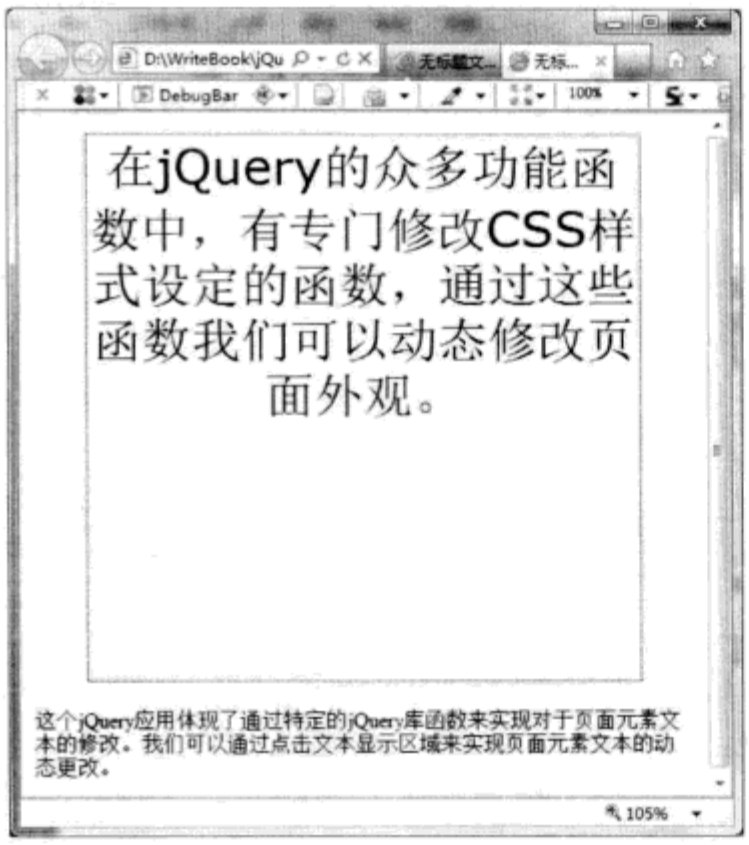


图 1.5　动态修改 DIV 文本内容一

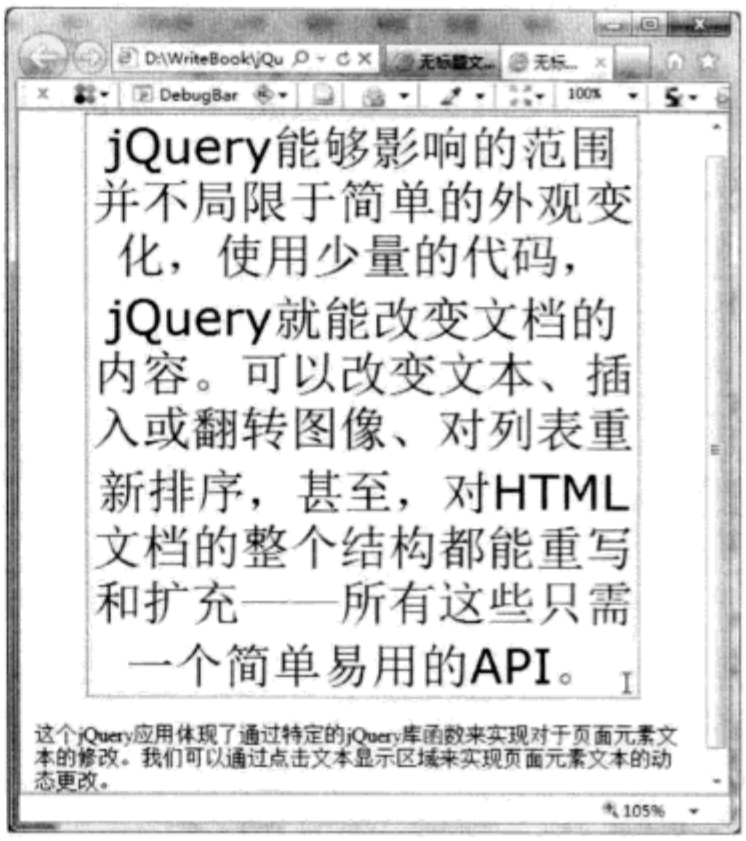


图 1.6　动态修改 DIV 内容二

（4）响应用户的页面操作。即使是最强大和最精心设计的行为，如果无法控制它何时发生，也毫无用处。jQuery 提供了截取形形色色的页面事件（如用户单击一个链接）的适当方式，而不需要使用事件处理程序搞乱 HTML 代码。此外，它的事件处理 API 也消除了经常困扰 Web 开发人员的浏览器不一致性。从上面的两个示例可以发现，在真正的 HTML 代码中，不需要在元素中加入任何事件说明，所有事件的注册操作全部集中在 jQuery 代码中，只需要一个元素的 ID 属性就万事大吉了。

（5）为页面添加动态效果。为了实现某种交互式行为，设计者必须向用户提供视觉上的反馈。jQuery 中内置的一批淡入、擦除之类的效果及制作新效果的工具包，为此提供了便利。这个特点在第 3 章中进行了专门的讲解。

（6）无须刷新页面即可从服务器获取信息。这种编程模式就是众所周知的 Ajax（Asynchronous JavaScript and XML，异步 JavaScript 和 XML），它能辅助 Web 开发人员创建出反应灵敏、功能丰富的网站。jQuery 通过消除这一过程中的浏览器特定的复杂性，使开发人员得以专注于服务器端的功能设计。关于 Ajax 请参考第 2 章的相关内容。

（7）简化常见的 JavaScript 任务。除了这些完全针对文档的特性之外，jQuery 也提供了对基本的 JavaScript 结构（如迭代和数组操作等）的增强。

1.2.2　jQuery 与其他脚本库的区别

jQuery 并不是唯一的 JavaScript 库，除 jQuery 之外，还有很多优秀的 JavaScript 库，如 Propetype、Dojo、Ext、YUI、MooTools 等。每款 JavaScript 库都有其自身的优点和缺点，要根据不同的使用场景进行选择。如表 1.1 所示是几款流行的脚本库比较。

表 1.1　脚本类库比较

类　　库	jQuery & jQueryUI	Propetype & script.aculo.us	Dojo	ExtJS	YUI	MooTools
文件大小（KB）	54	46～278	26	84～502	31	65
许可认证	MIT/GPL	MIT	BSD&AFL	Commercial & GPL	BSD	MIT
XMLHTTPREQUES 获取数据	是	是	是	是	是	是
JSON 数据获取	是	是	是	是	是	是
支持拖放	是	是	是	是	是	是
简单视觉效果	是	是	是	是	是	是
动画效果	是	是	是	是	是	是
事件处理	是	是	是	是	是	是
页面浏览历史	是	附加插件	是	History Manager	是	附加插件
输入验证	附加插件	是	是	是	是	附加插件
数据网格	附加插件	附加插件	是	是	是	附加插件
文本编辑器	附加插件	附加插件	是	是	是	附加插件
自动完成	是	是	是	是	是	是
HTML 自动生成	是	是	是	是	是	是
主题/皮肤选择	是	是	是	是	是	是
易用性	是	否	是	否	是	是
离线存储	否	否	是	Google Gears/Adobe Air	是	否
IE 版本	6+	6+	6+	6+	6+	6+
Firefox 版本	2+	1.5+	1.5+	1.5+	2+	1.5+
Safari	2+	2+	3+	3+	3+	2+
Opera	9+	9.25+	9+	9+	9+	9+

在表 1.1 中属于轻量型脚本库的是 jQuery 和 MooTools 这两个。在网站开发中，我们应该选择这种轻量型的脚本库。在这两个轻量型脚本库中，jQuery 以上手简单、文档全面、易用、运行稳定、高效等因素而受到绝大多数开发人员的青睐。

1.3　搭建 jQuery 运行环境

本节介绍如何搭建 jQuery 运行环境。因为 jQuery 是个轻量型框架，所以它的运行环境搭建很简单。

1.3.1　jQuery 库的选择

2006 年 8 月，jQuery 第一个版本 1.0 版正式面世，在这个版本里加入了 CSS 选择器、

事件处理、Ajax 接口。随着，jQuery 功能的不断更新，先后出现了 1.1 版、1.1.3 版、1.2 版、1.2.6 版、1.3 版、1.3.2 版、1.4 版、1.5 版、1.6 版等，时至今日，最新版 1.6.2 版已面世。在版本不断更新的过程中，jQuery 的功能和性能在不断增强。2007 年，jQuery UI 发布，这是一个包含大量预定义好的部件（widget）及一组用于构建高级元素（如可拖动的界面元素）的工具。

jQuery 的官方网站是 www.jQuery.com，下载地址为：http://docs.jquery.com/Downloading_jQuery。在 jQuery 官网上可以找到各种版本的 jQuery 库下载，每种版本几乎都有如下三种形式。

（1）Uncompressed——未压缩的脚本库文件。

（2）Minified——压缩后的类库文件，在网站正式上线运行时，我们应该使用这种形式的库文件。

（3）Visual Studio——这种版本是专门为 VS 工具提供的库文件，其中带有完整文档注释，可以为 VS 工具提供智能感知支持。

本书中绝大部分示例都使用了 jQuery 1.6 版本的库文件，图 1.7 和图 1.8 所示是 jQuery 官网上的下载页面。

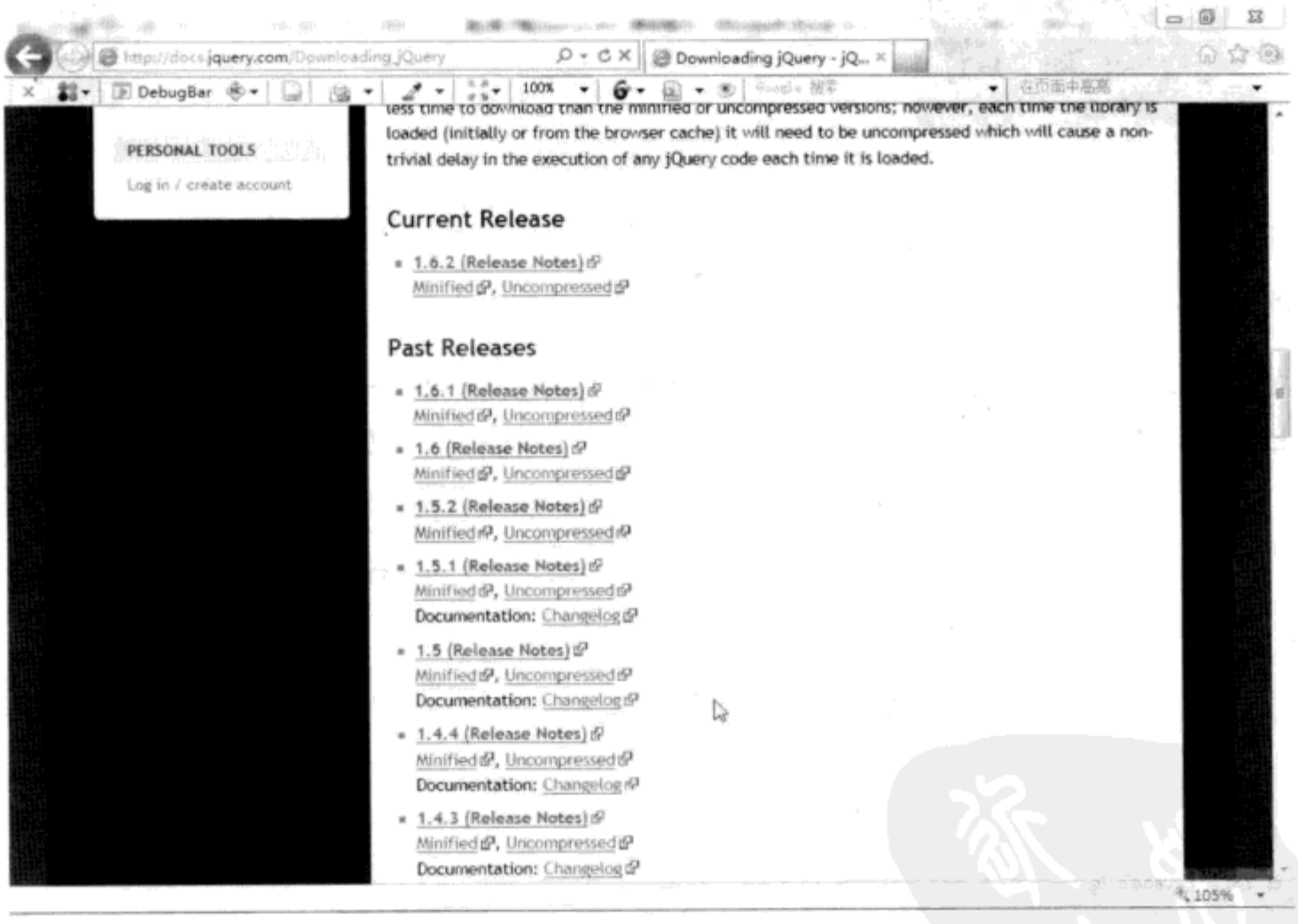


图 1.7　jQuery 各版本下载页面一

1.3.2　jQuery 库的引入

在 1.3.1 节说过 jQuery 的每个版本都有三种文件形式。在我们开发的过程中可以使用未压缩版，在真正发布网站时需要使用压缩库文件。引入 jQuery 库需要使用 HTML 的脚

本标记<script>，并通过在这个标记中设定库文件的位置及文件名实现 jQuery 的引入。例如：

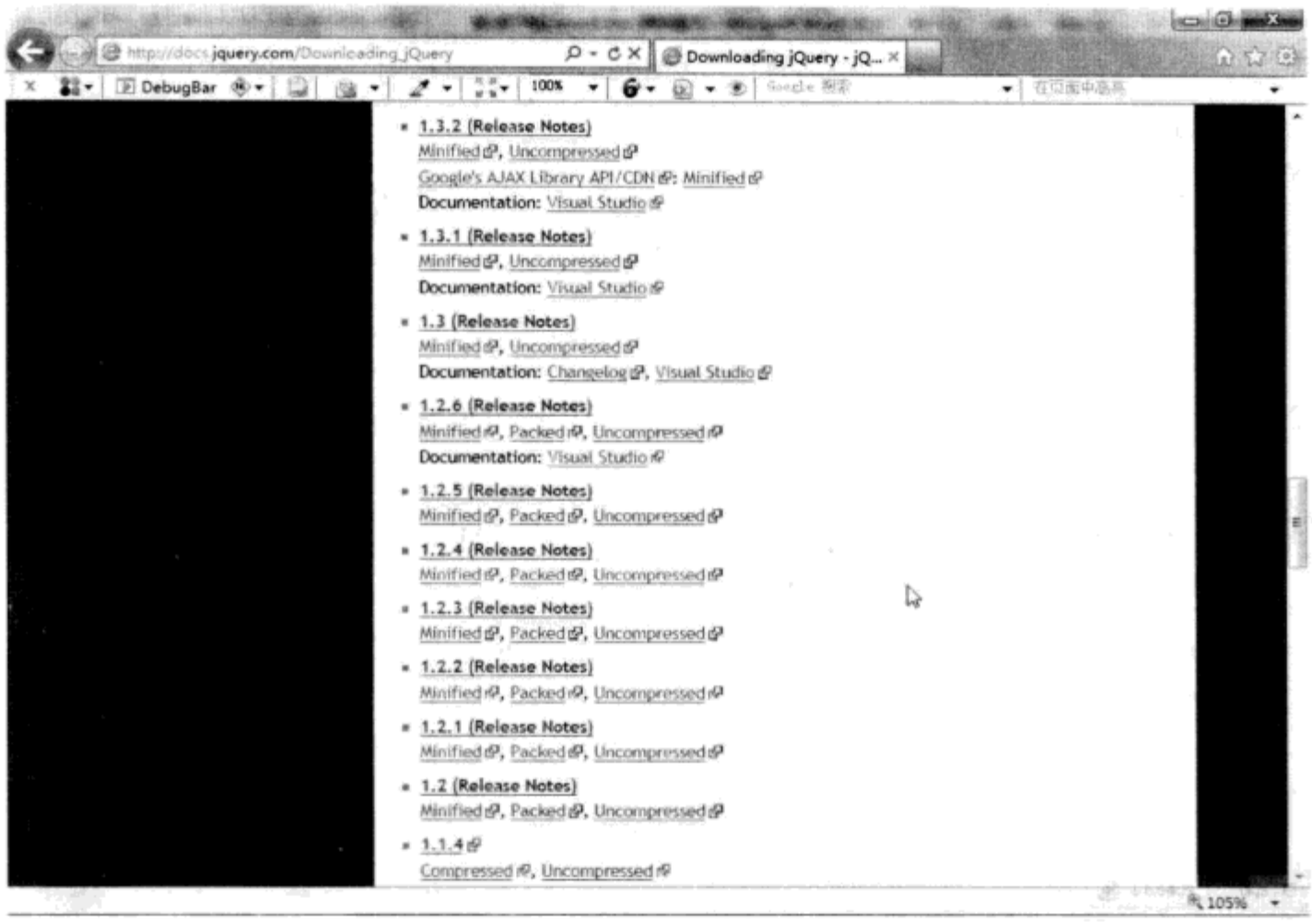


图 1.8　jQuery 各版本下载页面二

```
<script type="text/javascript" src="jslib/jquery-1.6.js"></script>
```

对于 jQuery 的库文件的存放位置，应该保存在独立的一个文件夹内，而不要同其他 HTML 文件、CSS 样式文件和 JS 脚本文件混合在一起存放，如图 1.9 所示。

图 1.9　jQuery 库文件存放位置

1.3.3　jQuery 的第一个示例

下面来见识一下 jQuery 的特殊页面效果，如图 1.10～图 1.12 所示。

图 1.10　页面初始加载完成

图 1.11　隐藏层放大过程

图 1.12　jQuery 显示隐藏元素

在这个示例中需要实现的需求是这样的：页面上有一个按钮和一个隐藏的层元素，当我们单击按钮时，这个隐藏的层逐渐放大显示出来。首先我们来看一下源代码：

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <script type="text/javascript" src="jslib/jquery-1.6.js"></script><!--
  引入 jQuery 库文件-->
6 <script type="text/javascript">
7   $(function(){
8       $("#btn").click(function(){     //按钮的单击事件
9           $("#div1").show(2000);      //DIV 层显示
10      });
11  });
12</script>
13</head>
14<body>
15<center>
16  <input id="btn" type="button" value="第一个 jQuery 效果" />
17   <div id="div1" style="display:none;width:400px;height:400px;border:
    solid 1px #000080;background-color:#AFA; font-family:'MS Serif', 'New
    York', serif; font-size:xx-large; color:#2A0000">Hello World!</div>
18</center>
19</body>
20</html>
```

上述代码第 5 行就是 jQuery 脚本库的引入，这里使用了 1.6 版本的脚本库。第 7 行使用了 jQuery 的最重要的一个事件——文档加载完成事件，这个事件用 ready()函数表示，我们这里使用了省略语法。第 8 行属于事件注册，将按钮的单击事件注册到按钮上。第 9 行使用了 jQuery 特效中的显示功能方法 show()，将隐藏的元素显示出来，这里给定了显示过程的时间为 2000 毫秒。上述代码中的$符号是 jQuery 的选择器符号，jQuery 的强大选择功

能都是通过它完成的。

通过上面这个示例我们初步见到了 jQuery 的功能。后面的章节将会帮助读者逐步了解 jQuery 的使用。

1.3.4　如何学习 jQuery

在学习 jQuery 之前，需要掌握一些相关的入门知识：HTML、CSS、JavaScript 基本编程、JavaScript DOM 编程、XML 基础、Ajax 原理等。

除了要掌握基础知识外，还要掌握 jQuery 的核心功能选择器的使用方法，以及 jQuery 工具函数的使用。在 HTML 5 中国官网上有 jQuery 的相关工具函数介绍，如图 1.13 所示。

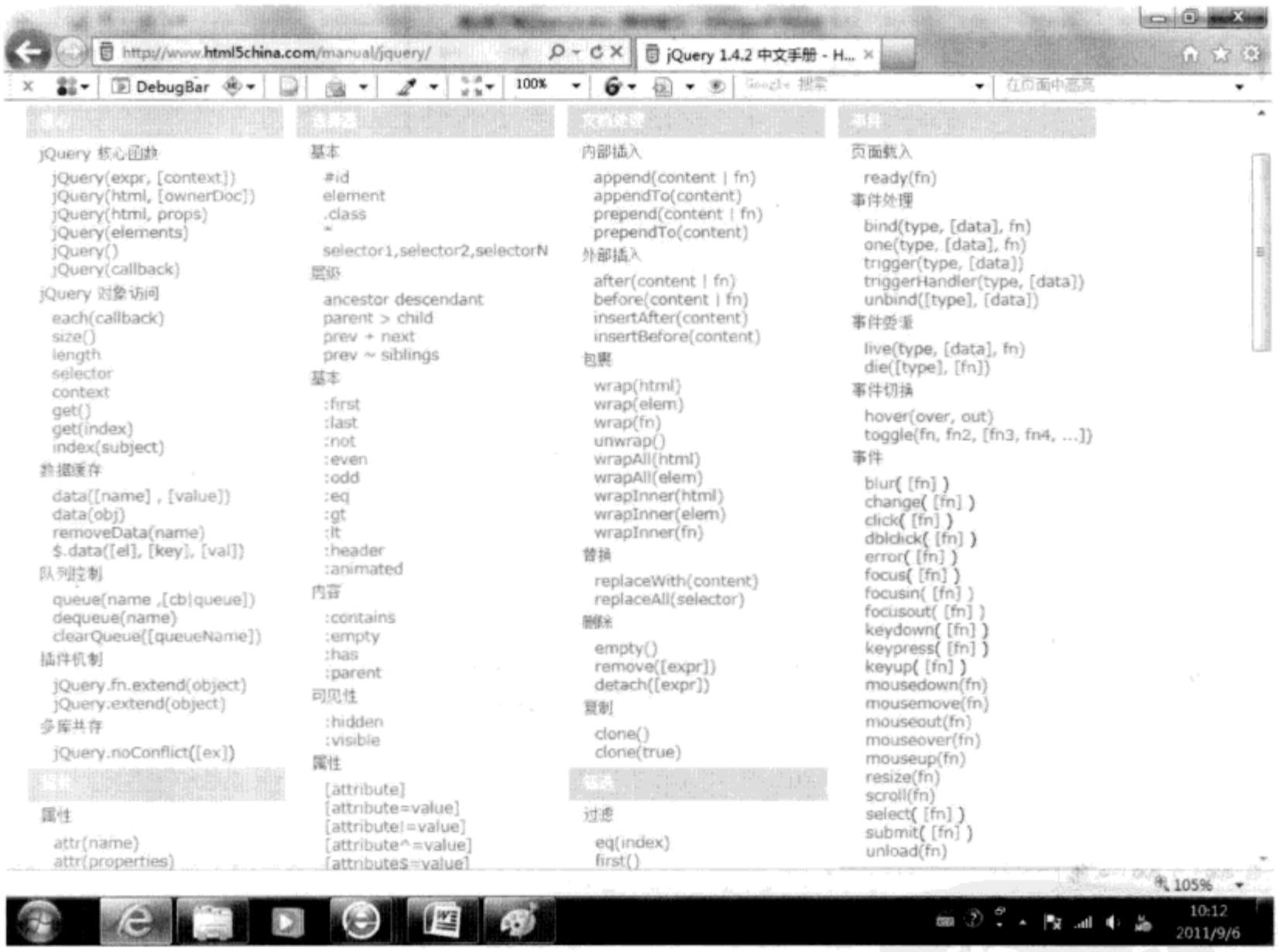


图 1.13　jQuery 相关工具函数介绍

1.4　小　　结

本章主要帮助读者初步认识什么是 jQuery、jQuery 具有什么特点，以及如何在页面中使用 jQuery。本章的重点在于 jQuery 库的选择与引入。下一章将就 jQuery 的一些基础知识及工作原理进行讲解。

第 2 章　jQuery 原理与运行机制

应该说 jQuery 是一种综合应用，因为它所涉及的知识内容相对比较繁杂。要想学好 jQuery，需要理解它的应用原理及运行机制。学习 jQuery 的原理与运行机制有助于更好地理解 jQuery 的核心选择器功能并更合理地使用 jQuery 选择器，有助于利用 jQuery 创建出高效的动画效果。下面针对 jQuery 的基本知识点进行讲解，包括 JavaScript BOM、JavaScript DOM、Ajax 等内容。

2.1　JavaScript 的浏览器对象模型 BOM 操作

BOM 是浏览器对象模型的简称。JavaScript 将整个浏览器窗口按照实现的功能拆分成若干个对象，这样 JavaScript 语言就可以以对象的形式来操作浏览器。

在一个完整的 BOM 中，主要包括 Window、Navigator、Screen、History、Location、Document 等对象。其中，Document 和 Location 对象既属于 BOM，也属于 DOM。Window 对象是整个 BOM 的顶层对象。各个对象所处位置、关系如图 2.1 和图 2.2 所示。

图 2.1　BOM 对象对应位置

下面依次介绍 JavaScript 如何操作这些对象。

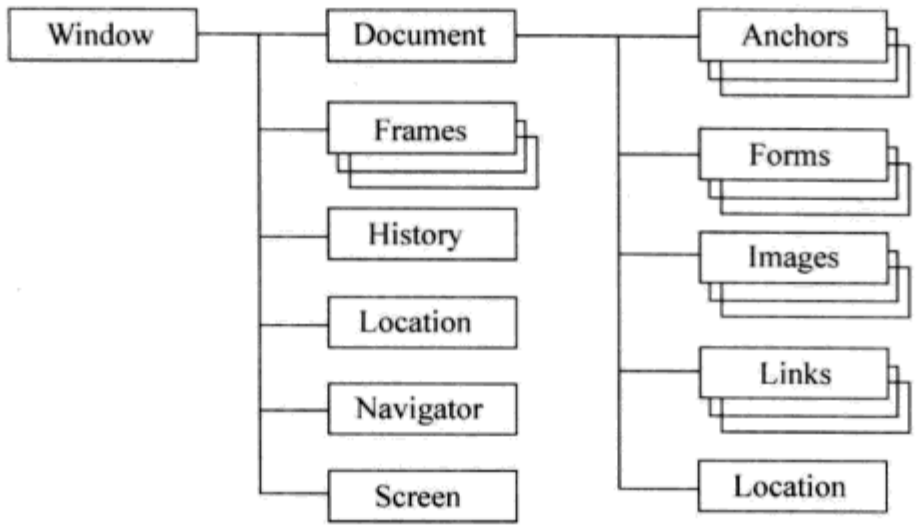


图 2.2　BOM 对象关系

2.1.1　Window 对象——窗口对象

Window 对象包含浏览器打开的窗口，也包括框架窗口。这个对象的属性和方法分别如表 2.1 和表 2.2 所示。

表 2.1　Window对象的属性说明

属　性	描　述
closed	返回窗口是否已被关闭
defaultStatus	设置或返回窗口状态栏中的默认文本
document	对 Document 对象的只读引用
history	对 History 对象的只读引用
innerheight	返回窗口的文档显示区的高度
innerwidth	返回窗口的文档显示区的宽度
length	设置或返回窗口中的框架数量
location	用于窗口或框架的 Location 对象
name	设置或返回窗口的名称
Navigator	对 Navigator 对象的只读引用
opener	返回对创建此窗口的窗口的引用
outerheight	返回窗口的外部高度
outerwidth	返回窗口的外部宽度
pageXOffset	设置或返回当前页面相对于窗口显示区左上角的 *X* 位置
pageYOffset	设置或返回当前页面相对于窗口显示区左上角的 *Y* 位置
parent	返回父窗口
Screen	对 Screen 对象的只读引用
self	返回对当前窗口的引用。等价于 window 属性
status	设置窗口状态栏的文本
top	返回最顶层的先辈窗口
window	window 属性等价于 self 属性，它包含了对窗口自身的引用
screenLeft screenTop screenX screenY	只读整数。声明了窗口的左上角在屏幕上的 *X* 坐标和 *Y* 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY

表 2.2　Window对象的方法说明

方　　法	描　　述
alert()	显示带有一段消息和一个确认按钮的警告框
blur()	把键盘焦点从顶层窗口移开
clearInterval()	取消由 setInterval()设置的 timeout
clearTimeout()	取消由 setTimeout()方法设置的 timeout
close()	关闭浏览器窗口
confirm()	显示带有一段消息及“确认”按钮和“取消”按钮的对话框
createPopup()	创建一个 pop-up 窗口
focus()	把键盘焦点给予一个窗口
moveBy()	可相对窗口的当前坐标把它移动到指定的像素
moveTo()	把窗口的左上角移动到一个指定的坐标
open()	打开一个新的浏览器窗口或查找一个已命名的窗口
print()	打印当前窗口的内容
prompt()	显示可提示用户输入的对话框
resizeBy()	按照指定的像素调整窗口的大小
resizeTo()	把窗口的大小调整到指定的宽度和高度
scrollBy()	按照指定的像素值来滚动内容
scrollTo()	把内容滚动到指定的坐标
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式
setTimeout()	在指定的毫秒数后调用函数或计算表达式

下面通过示例来说明如何使用 Window 对象。

1. 打开窗口

打开窗口操作主要使用 Window 对象的 open()方法，并在该方法中传入需要打开的文档位置。open()方法语法形式如下：

```
window.open(URL,name,features,replace)
```

注：URL 参数表示一个可选的地址字符串；name 参数表示窗口名称；features 表示窗口特征；replace 表示是否替换浏览器历史中的当前条目。

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>打开窗口</title>
5 <script type="text/javascript">
6   function open_win()
7   {
8       window.open("http://www.google.com.hk");//在当前窗口打开 Google 主页
9   }
10</script>
11</head>
12<body>
```

```
13  <form>
14      <input type=button value="打开窗口" onclick="open_win()">
15  </form>
16</body>
17</html>
```

上述代码第 8 行是按钮单击事件函数打开窗口的实现，在这里我们打开了 Google 网站，并且是在一个新窗口打开的，效果如图 2.3 所示。

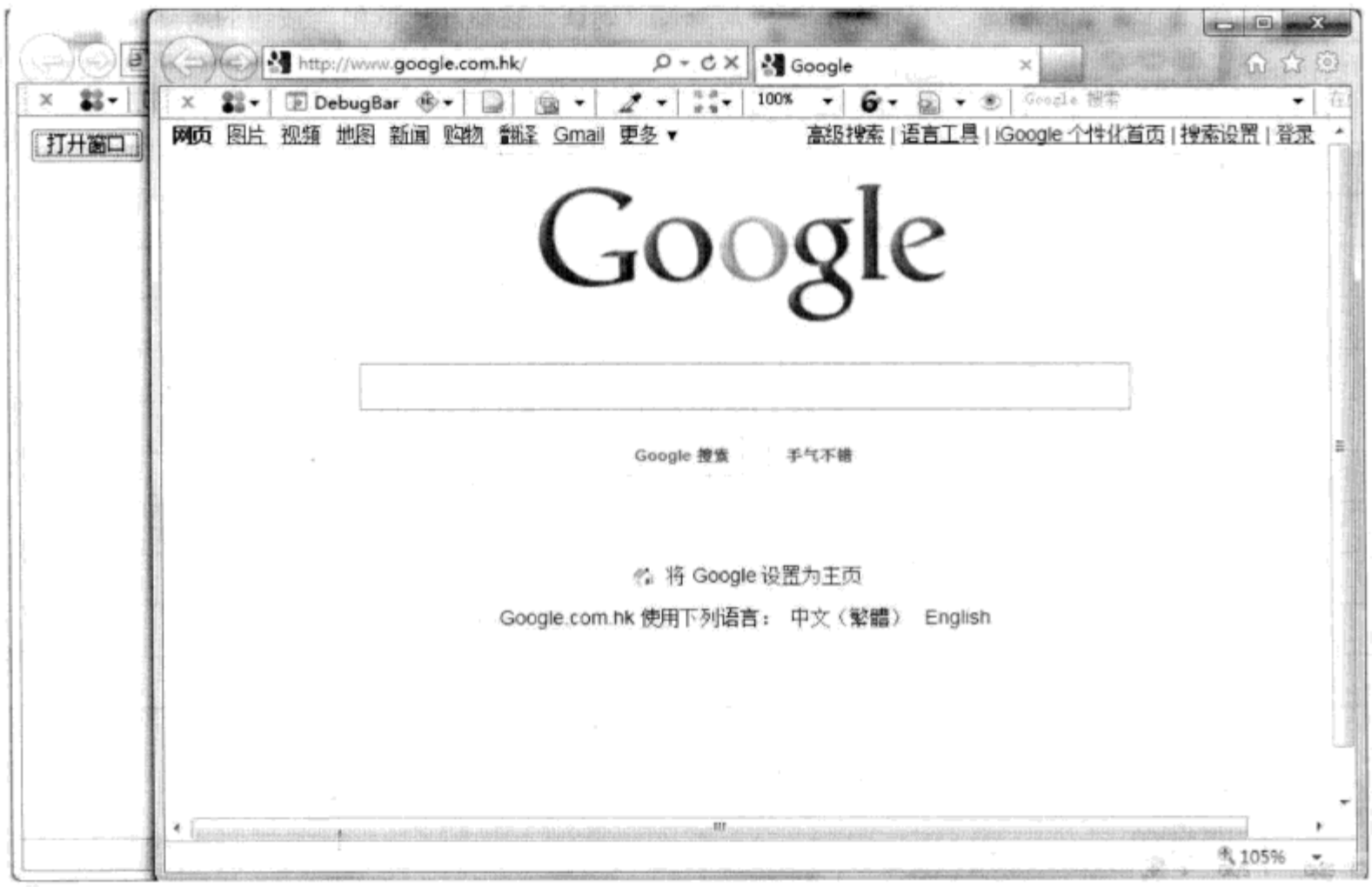


图 2.3　在新窗口打开页面

刚才看到了简单地打开一个窗口的实现。如果想控制打开窗口的尺寸大小，就需要对这个打开方法进行加工。

```
1 <script type="text/javascript">
2   function open_win()
3   {
4       window.open("http://www.google.com.hk","_blank","toolbar=yes,
      location=yes, status=no, menubar=yes, scrollbars=yes, resizable=no,
       width=400, height=400")                    //在新窗口打开 Google 主页
5   }
6 </script>
```

上面的代码在 open()函数中加入了 3 个参数：第一个参数是打开的文档的地址；第二个参数是打开位置，这里使用了新窗口的设置；第三个参数对窗口的大小及窗口上出现的内容进行设定，包括工具条显示、地址栏显示、状态栏隐藏、菜单栏显示、滚动条显示、不可调整大小。效果如图 2.4 所示。

图 2.4　设定打开窗口样式

2. 设置窗口的状态栏文本

这个功能是通过设置 Window 对象的 status 属性值实现的。status 语法形式如下：

```
window.status=sometext
```

注：status 属性是一个可读可写的字符串，声明了要在窗口状态栏中显示的一条消息。

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>设置状态栏</title>
5 </head>
6 <body>
7   <script type="text/javascript">
8       window.status="新的状态栏文本!!"          //更改窗口状态栏文本
9   </script>
10    <p>设置状态栏中的文本。</p>
11</body>
12</html>
```

效果如图 2.5 所示。

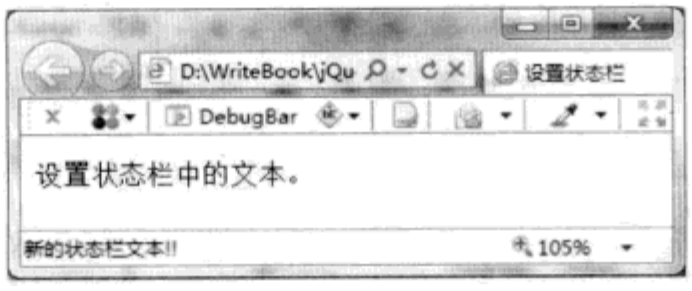


图 2.5　设置窗口状态栏文本

3. 改变窗口大小

Window 对象中提供了改变当前窗口大小的函数 resizeBy()和 resizeTo()。resizeBy()语法形式如下：

```
resizeBy(width,height)
```

注：width 和 height 参数分别表示窗口增加的宽度和高度，正负数均可。

resizeTo()语法形式如下：

```
resizeTo(width,height)
```

注：width 和 height 参数分别表示窗口想要达到的宽度和高度。

下面的代码完成了窗口宽和高都缩小 100 个像素的功能。

```
1 <html>
2 <head>
3 <script type="text/javascript">
4   function resizeWindow()
5   {
6       window.resizeBy(-100,-100)      //调整窗口大小
7   }
8 </script>
9</head>
10<body>
11    <form>
12       <input type="button" onclick="resizeWindow()" value="改变窗口大小">
13    </form>
14    <p><b>说明：</b>JavaScript 语言可以通过 Window 对象的 resizeBy()函数动态修
      改窗口的大小。</p>
15</body>
16</html>
```

效果如图 2.6 和图 2.7 所示。

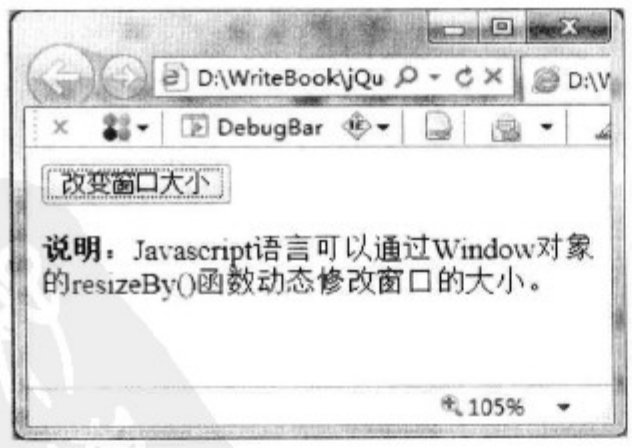


图 2.6　窗口初始大小

图 2.7　改变宽度和高度后的窗口大小

刚才的示例是通过差值改变窗口大小。也可以直接给定要达到的宽度和高度值。

```
1 <script type="text/javascript">
2   function resizeWindow()
3   {
4       window.resizeTo(300,300)    //调整窗口大小
5   }
6 </script>
```

这里直接指定了将窗口大小调整为 300 像素高、300 像素宽，效果如图 2.8 所示。

图 2.8　变化到指定宽度和高度

4. 滚动页面文本内容

Window 对象支持将页面内容进行滚动的操作。scrollBy()语法形式如下：

```
scrollBy(xnum,ynum)
```

注：xnum 参数表示文档向右滚动的像素数，ynum 参数表示文档向下滚动的像素数。

scrollTo()语法形式如下：

```
scrollTo(xpos,ypos)
```

注：xpos 参数表示要在窗口文档显示区左上角显示的文档的 *X* 坐标，ypos 参数表示要在窗口文档显示区左上角显示的文档的 *Y* 坐标。

下面的代码将页面内容动态向上滚动了 100 个像素距离。

```
1 <html>
2 <head>
3 <script type="text/javascript">
4   function scrollWindow()
5   {
6       window.scrollBy(100,100);      //滚动页面内容
7   }
8 </script>
9 </head>
10<body>
11<input type="button" onclick="scrollWindow()" value="滚动" />
12<p>JavaScript</p>
13<br /><br /><br /><br /><br />
14<br /><br /><br />
15<p>JavaScript</p>
16<br /><br /><br /><br /><br />
17<br /><br /><br />
18<p>JavaScript</p>
19</body>
20</html>
```

效果如图 2.9 和图 2.10 所示。

图 2.9　页面内容初始位置

图 2.10　页面内容滚动后的效果

也可以将上述代码改成移动到指定位置。

```
1 <script type="text/javascript">
2   function scrollWindow()
3   {
4       window.scrollTo(100,200);        //滚动页面内容
5   }
6 </script>
```

上述代码使用了 Window 对象的移动到某一位置的方法，效果与刚才的示例类似。

5. setInterval()和setTimeout()之间的区别

Window 对象中的 setInterval()和 setTimeout()这两个函数都和计时器相关。它们之间的区别是前者执行多次，后者只执行一次。setTimeout()语法形式如下：

```
setTimeout(code,millisec)
```

注：code 参数表示要调用的函数后要执行的 JavaScript 代码串，millsec 参数表示在执行代码前需等待的毫秒数。

setInterval()语法形式如下：

```
setInterval(code,millisec[,"lang"])
```

注：code 参数表示要调用的函数后要执行的 JavaScript 代码串，millsec 参数表示周期性执行或调用 code 之间的时间间隔，以毫秒计。

首先来看 setTimeout()方法。下面的代码表示当单击按钮后延迟两秒钟弹出一个警告对话框。

```
1 <html>
2 <head>
3 <script type="text/javascript">
4   function timedMsg()
5   {
6       var t=setTimeout("alert('2 秒!')",2000) //间隔两秒钟后弹出提示对话框
7   }
8 </script>
```

```
9 </head>
10<body>
11<input type="button" value="弹出时间警告!" onClick="timedMsg()">
12</body>
13</html>
```

效果如图 2.11 所示。

图 2.11　定时调用代码效果

可以将 setTimeout 改成 setInterval 多次调用对话框显示。但是，这里有一个问题，setInterval 是无限次使用，如果需要停止，则需要调用 clearInterval()这个函数清空计时器。

6. 坐标值的获取

Window 对象代表窗口对象，这个对象中也涉及坐标获取与设置的操作。但是，BOM 中的一些坐标属性并不是所有浏览器、所有版本都能够接受的。所以，当浏览器类型或版本不同时，需要进行判定。

下面的代码主要根据不同浏览器对不同对象及属性的支持标准进行判断，获取了页面内容滚动的偏移量。例如，网景浏览器支持 Window 对象的 pageXOffset 和 pageYOffset 属性，而 IE 浏览器则支持 document 中 documentElement 的 scrollTop 和 scrollLeft 属性。

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <script type="text/javascript">
6   function getScrollXY() {
7     var scrOfX = 0, scrOfY = 0;
8     if( typeof( window.pageYOffset ) == 'number' ) {
9       //网景浏览器适配
10      scrOfY = window.pageYOffset;
11      scrOfX = window.pageXOffset;
12    } else if( document.body && ( document.body.scrollLeft || document.
    body.scrollTop ) ) {
13      //DOM 适配
14      scrOfY = document.body.scrollTop;
15      scrOfX = document.body.scrollLeft;
16    } else if( document.documentElement && ( document.documentElement.
    scrollLeft || document.documentElement.scrollTop ) ) {
17      //IE6 标准适配
18      scrOfY = document.documentElement.scrollTop;
19      scrOfX = document.documentElement.scrollLeft;
20    }
21    alert( 'Horizontal scrolling = ' + scrOfX + '\nVertical scrolling ='
    + scrOfY );;
22  }
23</script>
24</head>
25<body>
26  <br /><br /><br /><br />
```

```
27    <br /><br /><br /><br /><br />
28    <br /><br /><br /><br />
29    <br /><br /><br /><br />
30    <br /><br /><br /><br /><br />
31    <br /><br /><br /><br />
32    <br /><br /><br /><br />
33    <br /><br /><br /><br /><br />
34    <br /><br /><br /><br />
35    <a href="javascript:getScrollXY()">     获取偏移量</a>
36</body>
37</html>
```

效果如图 2.12 所示。

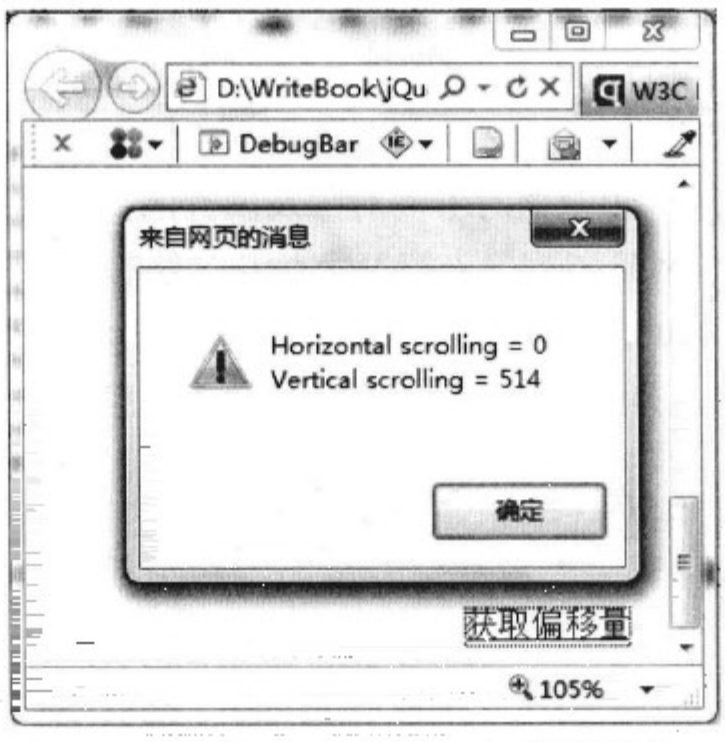


图 2.12　测试页面内容滚动偏移量

当我们获取窗口大小的时候，同样碰到了上面提到的情况。所以，还是要进行浏览器类型及版本号的判断。下面的代码检测在不同浏览器类型和版本下获得窗口大小的情况。

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <script type="text/javascript">
6   function alertSize() {
7     var myWidth = 0, myHeight = 0;
8     if( typeof( window.innerWidth ) == 'number' ) {
9       //非 IE 浏览器
10      myWidth = window.innerWidth;
11      myHeight = window.innerHeight;
12    } else if( document.documentElement && ( document.documentElement.
      clientWidth || document.documentElement.clientHeight ) ) {
13      //IE 6 以上的标准浏览器
14      myWidth = document.documentElement.clientWidth;
15      myHeight = document.documentElement.clientHeight;
```

```
16    } else if( document.body && ( document.body.clientWidth || document.
      body.clientHeight ) ) {
17      //IE 4 浏览器
18      myWidth = document.body.clientWidth;
19      myHeight = document.body.clientHeight;
20    }
21    alert( 'Width = ' + myWidth+'\n Height = ' + myHeight  );
22  }
23</script>
24</head>
25<body>
26  <a href="javascript:alertSize()">获取窗口大小</a>
27</body>
28</html>
```

效果如图 2.13 所示。

图 2.13　获取窗口大小

不过，现在最新版本的浏览器基本上都已经支持了 Window 对象的这些属性。我们来看下面这个示例。

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 </head>
6 <body>
7   <a href="javascript:alert('Inner Height: '+window.innerHeight+'\n'+
    'Inner Width: '+window.innerWidth)">窗口尺寸</a>
8    <a href="javascript:alert('Outer Height: '+window.outerHeight+'\n'+
    'Outer Width: '+window.outerWidth)">窗口外部尺寸</a>
9   <a href="javascript:alert('screenX: '+window.screenX+'\nscreenY:
    '+window.screenY)">窗口在屏幕上的位置</a>
10   <br /><br /><br /><br />
11   <br /><br /><br /><br />
```

```
12    <br /><br /><br /><br />
13    <br /><br /><br /><br />
14    <br /><br /><br /><br />
15    <br /><br /><br /><br />
16    <a href="javascript:alert('PageXOff: '+window.pageXOffset+'\nPageYOff:
      '+window.pageYOffset)">页面滚动偏移量</a>
17</body>
18</html>
```

上述代码使用了 Window 对象有关坐标及窗口尺寸的相关属性。测试环境为 IE 9，效果如图 2.14～图 2.17 所示。

图 2.14　窗口内部大小

图 2.15　窗口外部大小

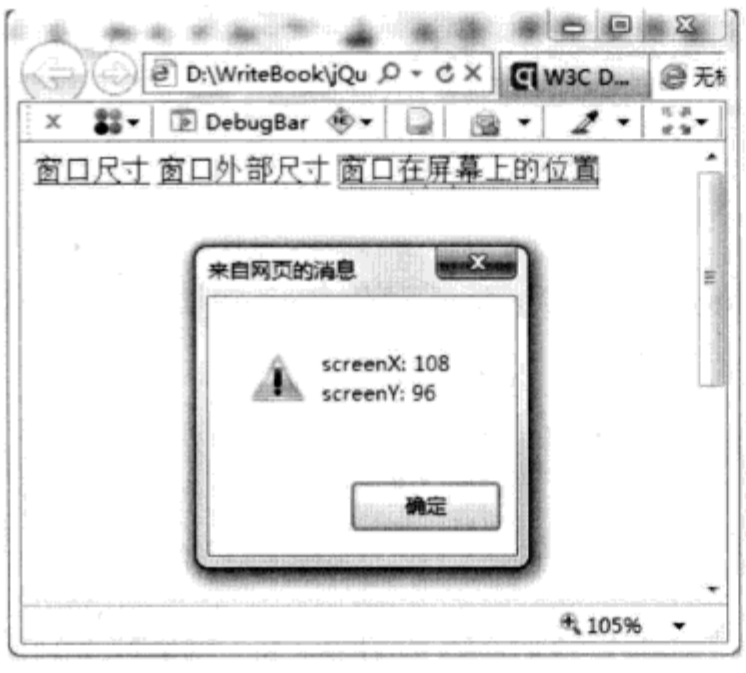


图 2.16　窗口在屏幕上的位置

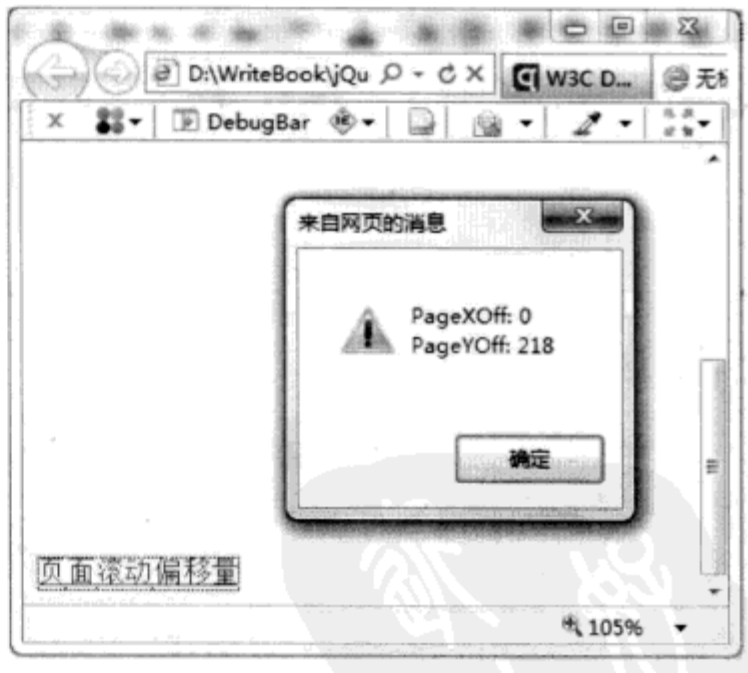


图 2.17　页面内容滚动偏移量

2.1.2　Navigator 对象——浏览器对象

Navigator 对象代表了浏览器相关信息集合。在这个对象中可以获取有关当前被使用的浏览器的信息内容。有关这个对象的属性及方法分别如表 2.3 和表 2.4 所示。

表 2.3　Navigator对象的属性说明

属　性	描　述
appCodeName	返回浏览器的代码名
appMinorVersion	返回浏览器的次级版本
appName	返回浏览器的名称
appVersion	返回浏览器的平台和版本信息
browserLanguage	返回当前浏览器的语言
cookieEnabled	返回指明浏览器中是否启用 cookie 的布尔值
cpuClass	返回浏览器系统的 CPU 等级
onLine	返回指明系统是否处于脱机模式的布尔值
platform	返回运行浏览器的操作系统平台
systemLanguage	返回 OS 使用的默认语言
userAgent	返回由客户机发送给服务器的 user-agent 头部的值
userLanguage	返回 OS 的自然语言设置

表 2.4　Navigator对象的方法说明

方　法	描　述
javaEnabled()	规定浏览器是否启用 Java
taintEnabled()	规定浏览器是否启用数据污点（data tainting）

下面利用这个对象的属性来获取浏览器的所有信息。

```
<html>
<body>
<script type="text/javascript">
  document.write("CodeName=" + navigator.appCodeName);
  document.write("<br />");
  document.write("MinorVersion=" + navigator.appMinorVersion);
  document.write("<br />");
  document.write("Name=" + navigator.appName);
  document.write("<br />");
  document.write("Version=" + navigator.appVersion);
  document.write("<br />");
  document.write("CookieEnabled=" + navigator.cookieEnabled);
  document.write("<br />");
  document.write("CPUClass=" + navigator.cpuClass);
  document.write("<br />");
  document.write("OnLine=" + navigator.onLine);
  document.write("<br />");
  document.write("Platform=" + navigator.platform);
  document.write("<br />");
  document.write("UA=" + navigator.userAgent);
  document.write("<br />");
  document.write("BrowserLanguage=" + navigator.browserLanguage);
  document.write("<br />");
  document.write("SystemLanguage=" + navigator.systemLanguage);
  document.write("<br />");
  document.write("UserLanguage=" + navigator.userLanguage);
</script>
</body>
</html>
```

效果如图 2.18 所示。

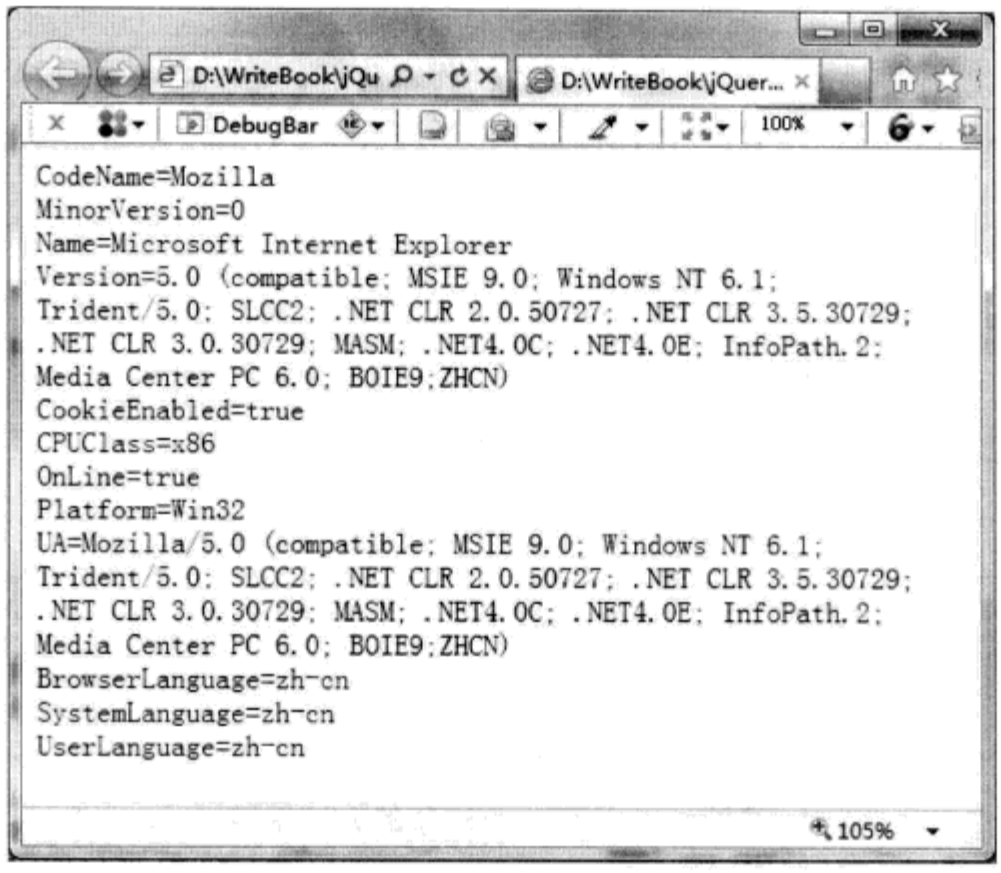


图 2.18　获取浏览器信息

1. Screen对象

这个对象包含了用户当前使用的显示器的相关信息，其属性如表 2.5 所示。

表 2.5　Screen对象的属性说明

属　性	描　述
availHeight	返回显示屏幕的高度（除 Windows 任务栏之外）
availWidth	返回显示屏幕的宽度（除 Windows 任务栏之外）
bufferDepth	设置或返回调色板的比特深度
colorDepth	返回目标设备或缓冲器上的调色板的比特深度
deviceXDPI	返回显示屏幕的每英寸水平点数
deviceYDPI	返回显示屏幕的每英寸垂直点数
fontSmoothingEnabled	返回用户是否在显示控制面板中启用了字体平滑
height	返回显示屏幕的高度
logicalXDPI	返回显示屏幕每英寸的水平方向的常规点数
logicalYDPI	返回显示屏幕每英寸的垂直方向的常规点数
pixelDepth	返回显示屏幕的颜色分辨率（比特每像素）
updateInterval	设置或返回屏幕的刷新率
width	返回显示器屏幕的宽度

下面利用 Screen 对象的属性来获取当前屏幕的相关信息。

```
1 <html>
2 <body>
3 <script type="text/javascript">
```

```
4   document.write("Screen resolution: ")
5   document.write(screen.width + "*" + screen.height)
6   document.write("<br />")
7   document.write("Available view area: ")
8   document.write(screen.availWidth + "*" + screen.availHeight)
9   document.write("<br />")
10  document.write("Color depth: ")
11  document.write(screen.colorDepth)
12  document.write("<br />")
13  document.write("Buffer depth: ")
14  document.write(screen.bufferDepth)
15  document.write("<br />")
16  document.write("DeviceXDPI: ")
17  document.write(screen.deviceXDPI)
18  document.write("<br />")
19  document.write("DeviceYDPI: ")
20  document.write(screen.deviceYDPI)
21  document.write("<br />")
22  document.write("LogicalXDPI: ")
23  document.write(screen.logicalXDPI)
24  document.write("<br />")
25  document.write("LogicalYDPI: ")
26  document.write(screen.logicalYDPI)
27  document.write("<br />")
28  document.write("FontSmoothingEnabled: ")
29  document.write(screen.fontSmoothingEnabled)
30  document.write("<br />")
31  document.write("PixelDepth: ")
32  document.write(screen.pixelDepth)
33  document.write("<br />")
34  document.write("UpdateInterval: ")
35  document.write(screen.updateInterval)
36  document.write("<br />")
37</script>
38</body>
39</html>
```

效果如图 2.19 所示。

图 2.19　屏幕对象信息

2．History对象

这个对象代表了用户访问 URL 的历史信息记录，它的属性和方法分别如表 2.6 和表 2.7 所示。

表 2.6　History对象的属性说明

属　性	描　述
length	返回浏览器历史列表中的 URL 数量

表 2.7　History对象的方法说明

方　法	描　述
back()	加载 history 列表中的前一个 URL
forward()	加载 history 列表中的下一个 URL
go()	加载 history 列表中的某个具体页面

这个对象相对比较简单，因此这里不再给出具体示例。有一点要说明的是在这个对象中 go()方法可以替代 back()和 forward()方法，即使用 go(-1)相当于调用 back()方法，使用 go(1)相当于调用 forward()方法。

3．Location对象

这个对象代表了当前访问的 URL 的所有信息，它的属性和方法分别如表 2.8 和表 2.9 所示。

表 2.8　Location对象的属性说明

属　性	描　述
hash	设置或返回从井号（#）开始的 URL（锚）
host	设置或返回主机名和当前 URL 的端口号
hostname	设置或返回当前 URL 的主机名
href	设置或返回完整的 URL
pathname	设置或返回当前 URL 的路径部分
port	设置或返回当前 URL 的端口号
protocol	设置或返回当前 URL 的协议
search	设置或返回从问号（?）开始的 URL（查询部分）

表 2.9　Location对象的方法说明

方　法	描　述
assign()	加载新的文档
reload()	重新加载当前文档
replace()	用新的文档替换当前文档

这个对象相对比较简单，因此不再给出具体示例。有兴趣的读者可以自己编写代码测试这个对象。

2.2　JavaScript 的 HTML 文档 DOM 操作

DOM 是文档对象模型的简称，JavaScript 可以通过文档对象的形式对 HTML 文档中的所有对象进行访问操作。jQuery 中的选择器实际上和 JavaScript 通过 DOM 来选取元素对象有很大关系。

2.2.1　什么是 DOM

DOM（Document Object Model，文档对象模型）是 JavaScript 中的 Document 方法。通过 JavaScript，可以重构整个 HTML 文档，还可以添加、移除、改变或重排页面上的项目。要改变页面的某个东西，JavaScript 就需要获得对 HTML 文档中所有元素进行访问的入口。这个入口，连同对 HTML 元素进行添加、移动、改变或移除的方法和属性，都是通过 DOM 来获得的。1998 年，W3C 发布了第一级 DOM 规范。这个规范允许访问和操作 HTML 页面中的每一个单独的元素。DOM 可被 JavaScript 用来读取、改变 HTML、XHTML 及 XML 文档。

2.2.2　DOM 节点

HTML 文档中的每个元素都是一个节点。DOM 中规定如下。

（1）整个文档是一个文档节点。

（2）每个 HTML 标签是一个元素节点。

（3）包含在 HTML 元素中的文本是文本节点。

（4）每一个 HTML 属性是一个属性节点。

（5）注释属于注释节点。

DOM 节点树如图 2.20 所示。

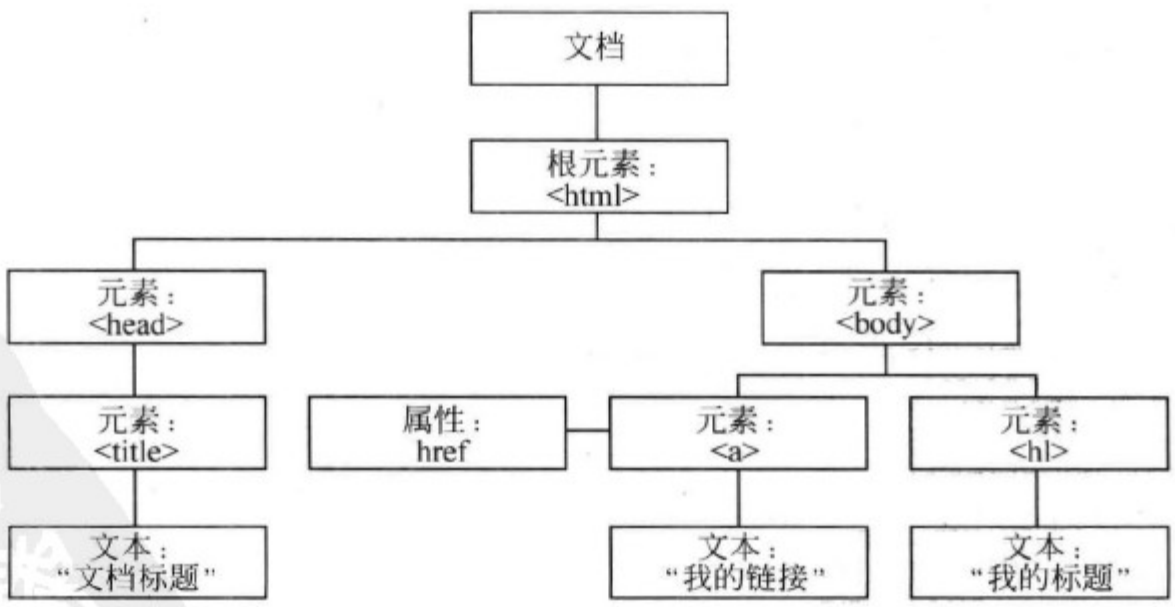


图 2.20　DOM 节点树

可以用一个简单的 HTML 文档来说明节点的特点。看下面这个静态 HTML 文档：

```
<html>
 <head>
   <title>DOM Tutorial</title>
 </head>
 <body>
   <h1>DOM Lesson one</h1>
   <p>Hello world!</p>
 </body>
</html>
```

上面所有的节点彼此间都存在关系。

（1）除文档节点之外的每个节点都有父节点。例如，<head>和<body>的父节点是<html>节点，文本节点“Hello world！”的父节点是<p>节点。大部分元素节点都有子节点。例如，<head>节点有一个子节点：<title>节点。<title>节点也有一个子节点：文本节点“DOM Tutorial”。

（2）当节点分享同一个父节点时，它们就是同辈（同级节点）。例如，<h1>和<p>是同辈，因为它们的父节点均是<body>节点。

（3）节点也可以拥有后代，后代指某个节点的所有子节点，或者这些子节点的子节点，依此类推。例如，所有的文本节点都是<html>节点的后代，而第一个文本节点是<head>节点的后代。

（4）节点也可以拥有先辈。先辈是某个节点的父节点，或者父节点的父节点，依此类推。比如，所有的文本节点都可以把<html>节点作为先辈节点。

2.2.3　访问文档节点

访问文档节点可以动态获取或者设置各节点的显示样式、内容等信息，便于前台设计人员更好地控制页面元素。访问文档节点可以通过一些方法来实现。

1．通过ID访问页面元素

通过 ID 访问页面元素的语法形式如下：

```
document.getElementById(id)
```

参数：id，必选项，为字符串（String）。返回值：对象。返回相同 id 对象中的第一个，如果无符合条件的对象，则返回 null。

可以通过访问节点来得到节点中的 HTML 内容，例如：

```
<html>
<head>
<script type="text/javascript">
  function getValue()
  {
      var x=document.getElementById("myHeader")  //通过 ID 访问页面元素
      alert(x.innerHTML)
```

```
8    }
9 </script>
10</head>
11<body>
12<h1 id="myHeader" onclick="getValue()">这是标题</h1>
13<p>单击标题，会提示出它的值。</p>
14</body>
15</html>
```

效果如图 2.21 所示。

图 2.21　利用 ID 访问页面元素

2. 通过name访问页面元素

通过 name 访问页面元素的语法形式如下：

```
document.getElementsByName(name)
```

参数：name，必选项，为字符串（String）。返回值：数组对象。如果无符合条件的对象，则返回空数组。

可以通过访问节点来得到节点中的 HTML 内容，例如：

```
1 <html>
2 <head>
3 <script type="text/javascript">
4   function getElements()
5   {
6    var x=document.getElementsByName("myInput"); //通过 name 属性访问页面元素
7    alert(x.length);
8   }
9 </script>
10</head>
11<body>
12<input name="myInput" type="text" size="20" /><br />
13<input name="myInput" type="text" size="20" /><br />
```

```
14<input name="myInput" type="text" size="20" /><br />
15<br />
16<input type="button" onclick="getElements()" value="名为 'myInput' 的元
  素有多少个？" />
17</body>
18</html>
```

效果如图 2.22 所示。

图 2.22　利用 name 访问页面元素

3. 通过TagName访问页面元素

通过 TagName 访问页面元素的语法形式如下：

```
document.getElementsByTagName(tagname)
```

参数：tagname，必选项，为字符串（String）。返回值：数组对象。如果无符合条件的对象，则返回空数组。

也可以批量访问标记名相同的所有节点，例如：

```
1 <html>
2 <head>
3 <script type="text/javascript">
4   function getElements()
5   {
6      var x=document.getElementsByTagName("input");//通过标记名访问页面元素
7      alert(x.length);
8   }
9 </script>
10</head>
11<body>
12<input name="myInput" type="text" size="20" /><br />
13<input name="myInput" type="text" size="20" /><br />
14<input name="myInput" type="text" size="20" /><br />
```

```
15<br />
16<input type="button" onclick="getElements()" value="标记名为 'input' 的元
  素有多少个？" />
17</body>
18</html>
```

效果如图 2.23 所示。

图 2.23　利用 TagName 访问页面元素

2.2.4　节点操作

上一节介绍了如何通过 JavaScript 来访问已有节点。下面来看一下 JavaScript 对 HTML 文档节点的其他操作。

1．节点信息

前面介绍了与 HTML 相关的获取操作。在 HTML 文档中实际可得的节点是多样的。每个节点都有相应的属性来描述并区分不同类型的节点。JavaScript 通过三种属性来识别不同的节点：nodeName，节点名称；nodeValue，节点值；nodeType，节点类型。

（1）nodeName 节点名称

- 元素节点的 nodeName 是标签名称。
- 属性节点的 nodeName 是属性名称。
- 文本节点的 nodeName 永远是#text。
- 文档节点的 nodeName 永远是#document。

（2）nodeValue 节点值

- 对于文本节点，nodeValue 属性包含文本。
- 对于属性节点，nodeValue 属性包含属性值。
- nodeValue 属性对于文档节点和元素节点是不可用的。

（3）nodeType 节点类型

节点类型如表 2.10 所示。

表 2.10　节点类型说明

元 素 类 型	节 点 类 型	元 素 类 型	节 点 类 型
元素	1	注释	8
属性	2	文档	9
文本	3		

我们可以获取节点本身的一些说明信息，如节点类型、节点名称、节点值等，例如：

```
1 <html>
2 <head>
3  <title>JavaScript!</title>
4 </head>
5 <body>
6   <div id="div1">这是一个层!</div>
7     <script type="text/javascript">
8       var x=document.getElementById("div1");
9       alert(x.nodeName);        //节点名
10      alert(x.nodeValue);       //节点值
11      alert(x.nodeType);        //节点类型
12    </script>
13 </body>
14</html>
```

效果如图 2.24～图 2.26 所示。

图 2.24　元素的节点名称

图 2.25　元素的节点值

请读者考虑图 2.25 中为何显示为空。

2. 节点的遍历操作

前面讨论了如何获取指定节点及节点的相关信息。JavaScript 还提供了根据已知节点遍历相关节点的操作属性。parentNode，父节点；firstChild，第一个子节点；lastChild，最后一个子节点。下面用两个示例来说明它们的使用方法。

图 2.26　元素的节点类型

例一：下面的代码使用 firstChild 属性，获取第一个子节点的节点值、节点类型和节点名称。

```
<html>
  <head>
        <title>JavaScript!</title>
  </head>
  <body>
    <p id="intro">My first paragraph...</p>
    <ul>
       <li>List item 1</li>
       <li>List item 2</li>
        <li>List item 3</li>
        <li>List item 4</li>
        <li>List item 5</li>
     </ul>
     <script type="text/javascript">
       //将 UL 中的列表项创建为节点列表
       var allListItems = document.getElementsByTagName('li');
       //现在我们可以使用 for 循环遍历列表项
       for (var i = 0, length = allListItems.length; i < length; i++) {
           //提取其文本节点并 alert 它的内容
           alert( allListItems[i].firstChild.nodeValue );
           alert(allListItems[i].firstChild.nodeType);
           alert(allListItems[i].firstChild.nodeName);
       }
     </script>
</body>
</html>
```

效果如图 2.27 所示。

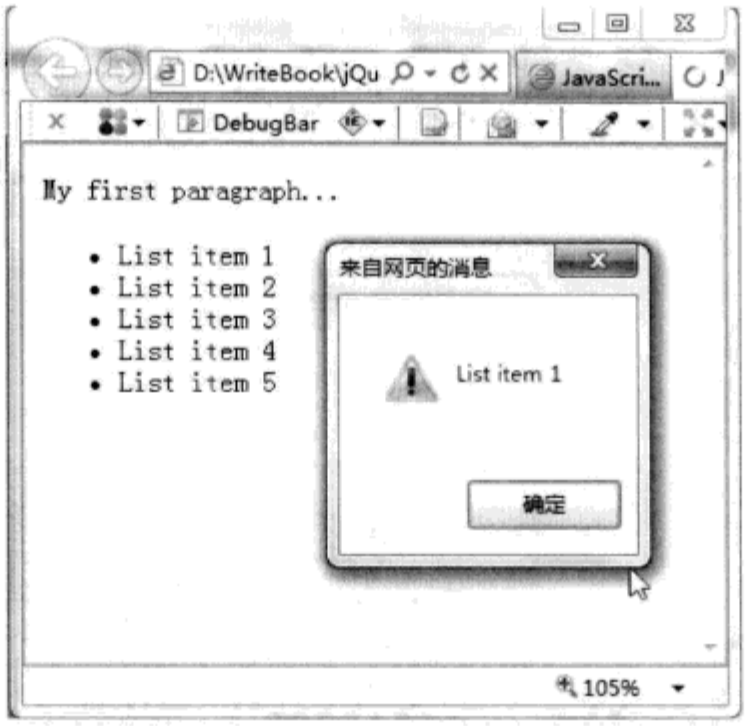


图 2.27　利用 firstChild 获取文本节点

例二：下面的代码分别使用了 parentNode、firstChild 和 lastChild 等属性。

```
1 <HTML>
2 <HEAD>
3  <TITLE>JavaScript</TITLE>
4 <script type="text/javascript">
5   function getTest()
6   {
7       var x = document.getElementById("test");
8       x.style.border = "1px dashed";
9   }
10  function getParent()
11  {
        //对父节点的操作
12      var x = document.getElementById("test");x.parentNode.style.border
        = "1px dashed";
13  }
14  function getFirst()
15  {
        //对第一个子节点的操作
16      var x = document.getElementById("test");alert(x.firstChild.
        nodeValue);
17  }
18  function getLast()
19  {
        //对最后一个子节点的操作
20      var x = document.getElementById("test2");alert(x.lastChild.
        nodeValue);
21  }
22</script>
23 </HEAD>
24 <BODY>
25  <div>
26  <p id="test">我是第一个 p 的文字</p>
```

```
27  <p id="test2">我是第二个 p 的文字，span 的文字</p></div>
28  <input type="button" onclick="getTest()" value="getTest()">
29  <input type="button" onclick="getParent()" value="getParent()">
30  <input type="button" onclick="getFirst()" value="getFirst()">
31  <input type="button" onclick="getLast()" value="getLast()">
32 </BODY>
33</HTML>
```

效果如图 2.28 和图 2.29 所示。

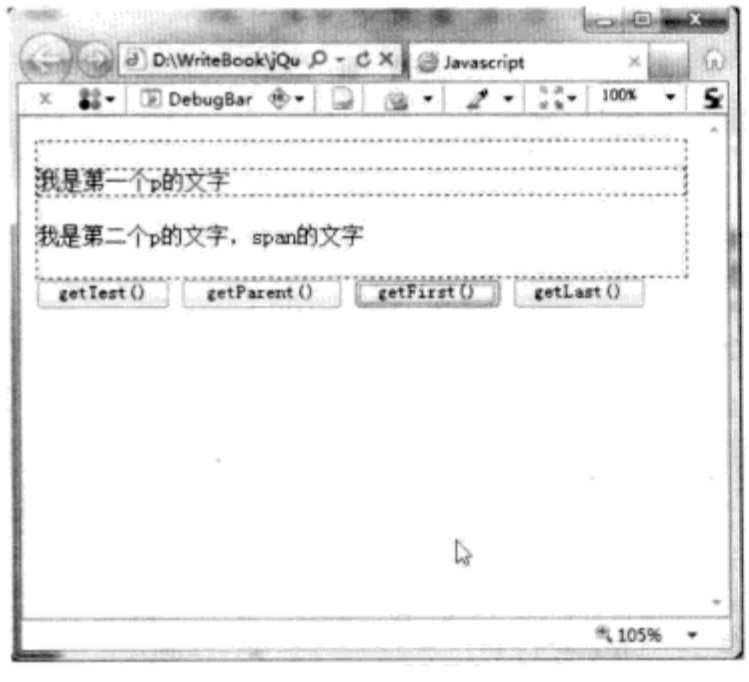


图 2.28　parentNode 获取父节点

图 2.29　lastChild 获取最后一个子节点

3. 节点的其他操作

这里附加了一些节点的其他比较有用的操作，这些操作也是 DOM 所特有的。innerHTML 属性设置或返回表格行的开始和结束标签之间的 HTML。举例如下：

```
1 <HTML>
2 <HEAD>
3  <TITLE>JavaScript</TITLE>
4 <script type="text/javascript">
5   function testW()
6   {
7       var str = "<p align='center'>面目全非！完全不一样喽！</p>";
    document.getElementById("test").innerHTML = str;
8   }
9 </script>
10 </HEAD>
11 <BODY>
12  <div id="test">
13  <p>我是测试段落。</p>
14  </div>
15  <input type="button" value="testW" onclick="testW()">
16 </BODY>
17</HTML>
```

效果如图 2.30 所示。

图 2.30　innerHTML 测试

实际上在 JavaScript 中还有另一个和 innerHTML 类似的属性 innerText。但是，由于浏览器的兼容性，innerText 属性只能在 IE 下使用。为了能设计出通用的 JavaScript 脚本程序，希望读者尽量使用 innerHTML。

- childNodes 属性，表示当前节点的直接子节点的集合。
- removeChild()方法，移除子节点。
- appendChild()方法，添加子节点。

下面用两个示例分别说明这两个方法和一个属性。

例一：使用 childNodes 属性及 removeChild()方法举例。

```
1 <HTML>
2 <HEAD>
3  <TITLE> New Document </TITLE>
4  <script type="text/javascript">
5  function remove()
6  {
7     var test = document.getElementById("test");
8     var children = test.childNodes;          //获取子节点集合
9     for(i=0;i<children.length;i++)
10    {
11       test.removeChild(children[i]);        //移除指定子节点
12    }
13 }
14 </script>
15 </HEAD>
16 <BODY>
17  <div id="test">
18  <p>我是将要被删除的节点</p>
19  <hr size=5 width=200 color=red>
20  </div>
21  <input type="button" value="Delete" onclick="remove()">
22 </BODY>
23</HTML>
```

上述代码使用了 childNodes 属性及 removeChild()方法，效果如图 2.31 和图 2.32 所示。

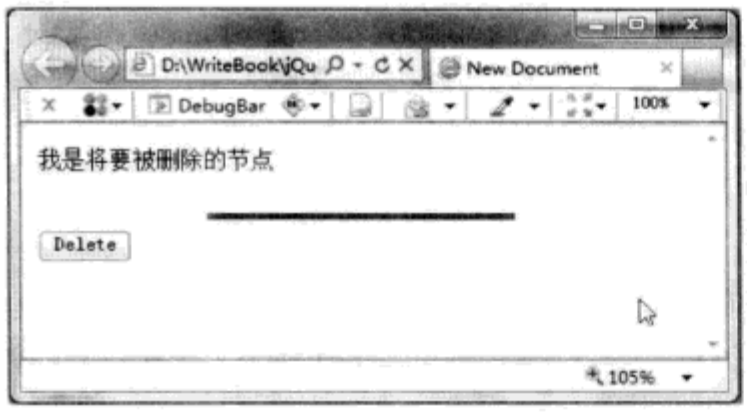


图 2.31　删除子节点前

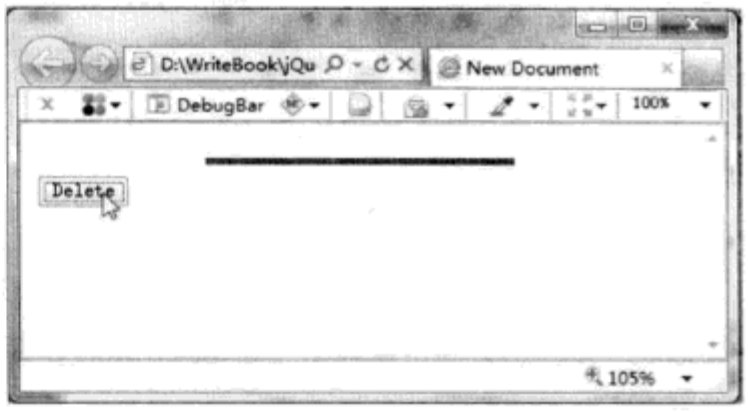


图 2.32　删除子节点后

例二：使用文档对象来创建一个段落标记、文本节点方法、文本节点等。

```
1 <HTML>
2 <HEAD>
3  <TITLE> New Document </TITLE>
4   <script type="text/javascript">
5       function test()
6       {
7           var test = document.getElementById("test");
8           var para = document.createElement("P");          //创建 HTML 元素
9           var text = document.createTextNode("要添加的文本");
                                                             //创建文本元素
10          para.appendChild(text);                  //向 HTML 元素中添加文本元素
11          test.appendChild(para);                  //将 HTML 元素添加到页面中
12      }
13  </script>
14 </HEAD>
15 <BODY>
16  <div id="test" style="border:1px solid"></div>
17  <input type="button" value="insert" onclick="test()">
18 </BODY>
19</HTML>
```

上述代码中使用了文档对象中的创建元素方法，第 8 行创建了一个段落标记。还使用了创建文本节点方法，第 9 行创建了文本节点。第 10 行利用添加节点方法将文本节点加入到段落中。第 11 行将段落添加到层中。效果如图 2.33 所示。

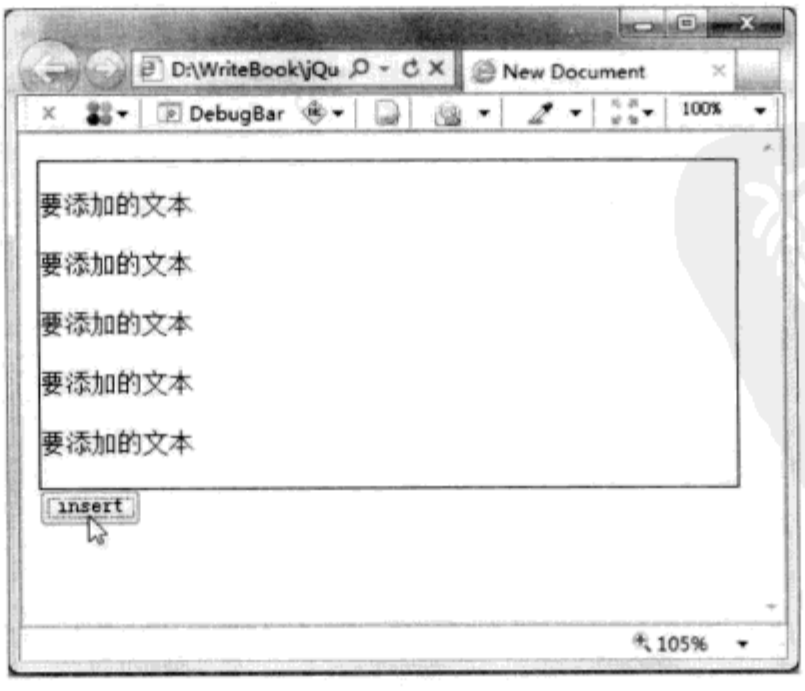


图 2.33　添加节点操作

2.3　Ajax 原理

jQuery 可以在客户端浏览器实现无刷新的情况下页面内容发生改变。但 jQuery 本身操作的还是浏览器上的数据。如果 jQuery 可以和 Ajax 搭配使用则会发挥更大功效。本书中很多插件都支持 Ajax 数据。下面简单介绍一下 Ajax 的工作原理及使用方法。

2.3.1　Ajax 组成

Ajax（Asynchronous JavaScript and XML）包括 HTML/XHTML、CSS、DOM、XML、XSLT、XMLHTTP、JavaScript。它使用 XHTML 和 CSS 标准化呈现，使用 DOM 实现动态显示和交互，使用 XML 和 XSLT 进行数据交换与处理，使用 XMLHTTPRequest 进行异步数据读取，使用 JavaScript 绑定和处理所有数据。

2.3.2　Ajax 与基本 Web 应用工作比较

以往我们浏览网页的原理是由 Client 向 Server 提交页面申请，再由 Server 将申请通过 HTTP 传回给 Client 生成浏览页面，如图 2.34 所示。

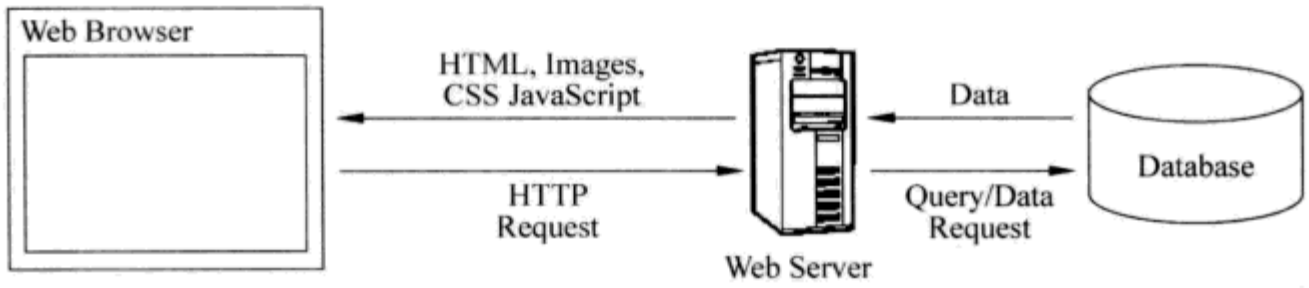


图 2.34　基本 Web 应用工作过程

使用 Ajax 后的工作原理如图 2.35 所示。

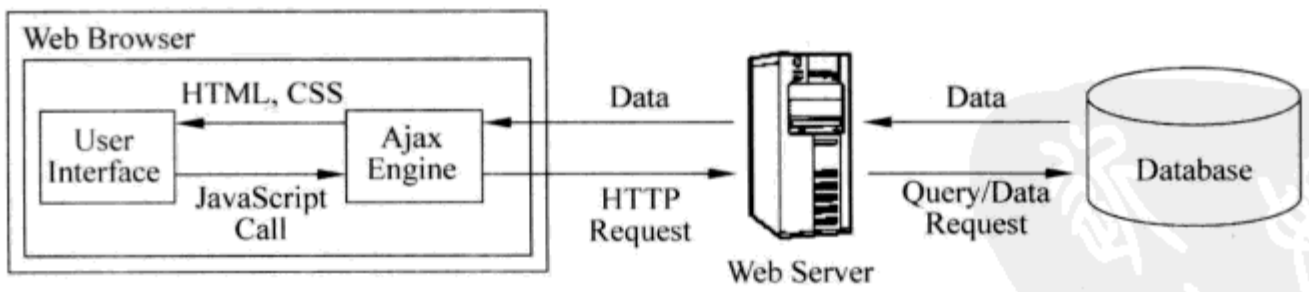


图 2.35　Ajax 工作原理

也可以直接从交互时序上来观察使用了 Ajax 的效果，如图 2.36 和图 2.37 所示。通过以上几幅图的对比可以总结 Ajax 的工作原理和优点。

1．工作原理

Ajax 的工作原理是相当于在用户和服务器之间加了一个中间层，使用户操作与服务器

响应异步化。并不是所有的用户请求都提交给服务器，像一些数据验证和数据处理等都交给 Ajax 引擎自己来做，只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求。当请求被响应时，服务器返回的数据是 XML 格式。由 Ajax 引擎进行解析，然后交给浏览器显示处理。

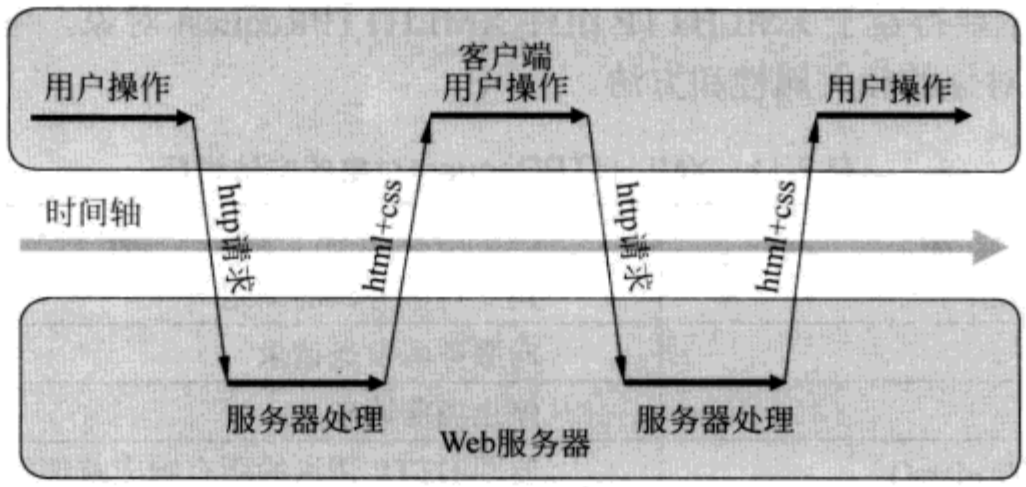


图 2.36　基本 Web 应用时序

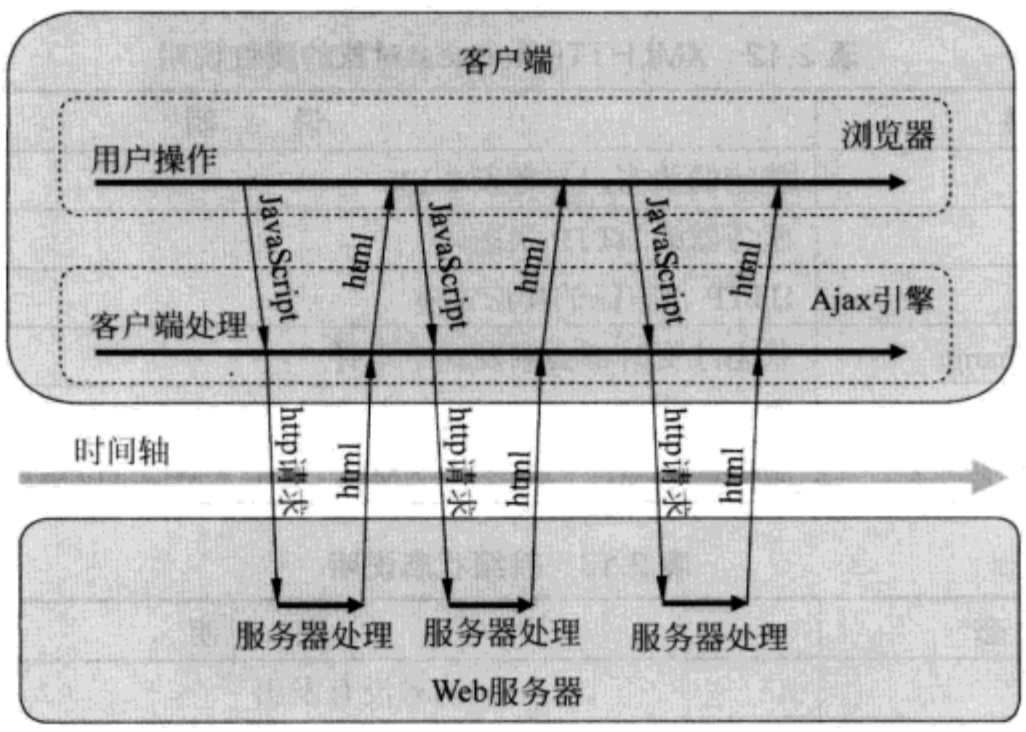


图 2.37　Ajax 工作时序

Ajax 核心只有 JavaScript、XMLHTTPRequest 和 DOM，在旧的交互方式中，由用户触发一个 HTTP 请求到服务器，服务器对其进行处理后再返回一个新的 HTML 页到客户端。每当服务器处理客户端提交的请求时，客户都只能空闲等待，并且哪怕只是一次很小的交互、只需从服务器端得到很简单的一个数据，都要返回一个完整的 HTML 页。而用户每次都要浪费时间和带宽去重新读取整个页面。使用 Ajax 后用户会感觉几乎所有的操作都会很快响应，没有页面重载（白屏）的等待。

2．Ajax优点

通过适当的 Ajax 应用可以达到更好的用户体验，把以前的一些服务器负担的工作转嫁到客户端，利用客户端闲置的处理能力来处理，减轻服务器和带宽的负担，从而达到节约 ISP 的空间及带宽租用成本的目的。

2.3.3　Ajax 核心对象 XMLHTTPRequest

Ajax 最大的特点是无须刷新页面便可向服务器传输或读写数据（又称无刷新更新页面），这一特点主要得益于 XMLHTTP 组件 XMLHTTPRequest 对象。可以参考表 2.11 和表 2.12 了解这个对象的相关属性和方法。

表 2.11　XMLHTTPRequest对象的方法说明

方　　法	说　　明
open("method", "url")	建立对服务器的调用
send(content)	向服务器发送请求
abort()	停止当前请求
getAllResponseHeaders()	返回 HTTP 请求的所有响应首部的键/值对
getResponseHeader("header")	返回指定响应首部的串值
setRequestHeader("header", "value")	设置指定首部的值，在设置前必须先调用 open()

表 2.12　XMLHTTPRequest对象的属性说明

属　　性	说　　明
readyState	就绪的状态，参考表 2.13
status	服务器的 HTTP 状态码
statusText	HTTP 状态码的相应文本
onreadystatechange	状态改变时都会触发这个事件
responseText	服务器的响应，表示为一个字符串
responseXML	服务器的响应，表示为 XML，这个 XML 可以解析为一个 DOM 对象

表 2.13　就绪状态说明

就 绪 状 态	说　　明
0	请求没有发出
1	请求已经建立但还没有发出
2	请求已经发出，正在处理之中
3	请求已经处理
4	响应已完成

2.3.4　Ajax 工作示例

下面用一个示例来演示 Ajax 的工作过程及 XMLHTTPRequest 的使用方法。因为 Ajax 的工作需要后台服务器支撑，所以我们用到的这个示例要在有服务器的环境下使用。读者可以在具有 IIS 的计算机上测试这个示例。

首先，我们来看它的 JavaScript 源代码。HTML 代码请参考光盘内容。

```
1 <script type="text/javascript">
2   var xmlHttp=null;
3   try
```

```
4   {
5       xmlHttp=new XMLHttpRequest();
        //检查浏览器是否使用本地 JavaScript 对象技术支持 XMLHTTPRequest
6   }
7   catch (e)
8   {
9       try
10      {
11          xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                                                    //使用新版本的 XMLHTTPRequest
12      }
13      catch (e)
14      {
15          xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                                                    //早期版本的 XMLHTTPRequest
16      }
17  }
18
19  function getpage(serverPage,objId)
20  {
21      var obj = document.getElementById(objId);
22      xmlHttp.open("GET", serverPage);          //打开与请求页面的连接
23      xmlHttp.onreadystatechange = function()//XML 模块的准备状态改变事件
24      {
25          if (xmlHttp.readyState == 4 && xmlHttp.status == 200)
                                                    //判断是否是正确返回结果
26          {
27              obj.innerHTML = xmlHttp.responseText;
28          }
29      }
30      xmlHttp.send(null);                       //发送请求,在 Get 方式下
31  }
32</script>
```

上述代码第 2 行定义了获取 XMLHTTPRequest 对象的变量。第 3～17 行创建 XMLHTTPRequest 对象，其中用到了 JavaScript 的异常处理技术 try…catch。如果能够使用本地 JavaScript 对象技术支持 XMLHTTPRequest，则创建，否则判断新老版本的 XMLHTTPRequest 哪一个可以使用并创建。第 21 行表示获取需要动态显示 Ajax 请求返回内容的区域。第 22 行指定用 GET 方式向 URL 发送一个异步请求，获取指定的页面。

第 23 行是一个事件订阅方式，这个事件当 XMLHTTPRequest 的就绪状态发生改变时激发。第 25 行判断就绪状态是否响应结束及 HTTP 的返回状态是否良好。第 27 行将服务器返回的请求页面内容文本显示在页面的区域内。第 30 行把请求发送给指定的目标资源，当发送请求的方式为 POST 时，send 方法可以指定一个参数：串/DOM 对象，它作为请求体的一部分发送到目标 URL。当发送方式为 GET 时，参数为 null。效果如图 2.38 所示，当我们单击链接，显示另一个页面时，浏览器是不会刷新的。

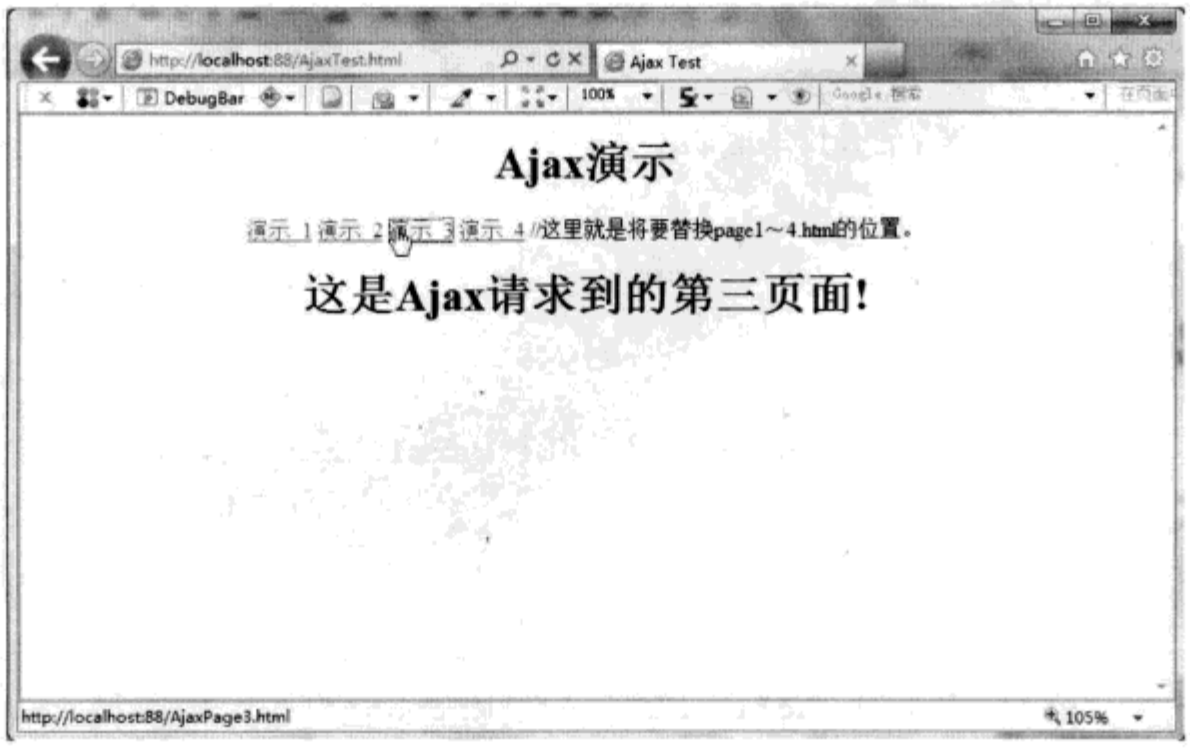


图 2.38　Ajax 测试效果

2.4　jQuery 工作原理

jQuery 在实际应用中基本是依靠它的选择器筛选匹配的页面元素对象，并调用它提供的功能函数来完成我们所需要的工作。它的编写和我们前面所看到的 JavaScript 的编写很不一样。所以，官方在介绍 jQuery 的时候引用了这样一段话：jQuery 是为了改变 JavaScript 的编码方式而设计的。下面我们用图 2.39 来说明 jQuery 的工作原理。

图 2.39　jQuery 工作原理

通过图 2.39 可以认识到一个 jQuery 应用程序是如何编写及执行的。jQuery 本身具有下面的特点。

（1）在 jQuery 应用中对于元素的选择是关键，所有的 jQuery 的功能函数都是绑定到一定元素上才执行的。从 jQuery 的源代码中可以看到，var$=jQuery。因此，当我们进行

$(selector)操作时，其实就是执行 jQuery(selector)，创建的是一个 jQuery 对象。正确的写法应该是：var jq = new $(selector);。而 jQuery 使用了一个小技巧在外部避免了 new，在 jQuery 方法内部实现：if (window == this) return new jQuery(selector)。

（2）函数的使用是依靠 jQuery 对象来执行的，而创建出来的 jQuery 对象形式可能不同，有时代表单一元素，有时代表一组元素。所以可以通过 each()函数对一组元素的对象进行遍历操作。

（3）jQuery 具有可扩展性，不管是从框架扩展还是从对象扩展都可以实现。

2.5　jQuery 运行机制

jQuery 运行机制主要包含两个方面：元素选择和事件机制。

2.5.1　jQuery 的元素选择

jQuery 的强大之处就在于它本身支持多种选择器样式。在编写 jQuery 应用的时候可以根据要求使用不同的选择器来选择元素。下面分别介绍一下它所支持的选择器。

1．基本元素选择器

在使用这种选择器时，可以使用元素标记名、元素类名、元素 ID 来填写选择器。例如：

```
$("p")                          //选取 <p> 元素
$("p.intro")                    //选取所有 class="intro" 的 <p> 元素
$("p#demo")                     //选取 id="demo" 的第一个 <p> 元素
```

分层选择器：使用这种选择器时，需要传入多个值，并用空格或大于号分隔。例如：

```
$("div input");                 //div 下所有 input
$("div > input);                //父元素下的子元素
```

2．基本条件选择器

使用这种选择器时，需要在元素的选择符后加上基本条件运算符，这些条件运算符都是 jQuery 内置的运算符。例如：

```
$("p:first")                    //选择第一个段落
$("p:last")                     //选择最后一个段落
$("tr:even")                    //选择偶数表格行
$("tr:odd")                     //选择奇数表格行
$("input:not(:checked)")        //选择所有未被选中的元素
$("tr:eq(1)")                   //选择索引值为 1 的表格行
$("tr:gt(0)")                   //选择索引值大于 0 的表格行
$("tr:lt(2)")                   //选择索引值小于 2 的表格行
$(":header")                    //选择所有标题元素
$(":animated")                  //选择所有正在执行动画的元素
```

3. 内容条件选择器

使用这种选择器时，需要在元素的后面加上内容筛选运算符。例如：

```
$("div:contains('John')")          //选择包含'John'文本的层元素
$("td:empty")                      //选择不包含文本或者子元素的表格单元
$("div:has(p)")                    //选择包含段落元素的层元素
$("td:parent")                     //选择包含子元素或者文本的表格单元
```

4. 可见性条件选择器

使用这种选择器时，需要在元素后面加上可见性条件。例如：

```
$("tr:hidden")                     //选择所有隐藏的表格行
$("tr:visible")                    //选择所有可见的表格行
```

5. 属性选择器

使用这种选择器时，需要利用元素属性并使用一定条件来进行选择。例如：

```
$("div[id]")                       //选择具有 id 属性的层
$("input[name='newsletter']")      //选择具有属性 name 并且属性值为'newsletter'的
                                     表单输入元素
$("input[name!='newsletter']")     //选择具有属性 name 并且属性值不为'newsletter'
                                     的表单输入元素
$("input[name^='news']")           //选择具有属性 name 并且属性值以'news'为起始内容
                                     的表单输入元素
$("input[name$='letter']")         //选择具有属性 name 并且属性值以'letter'为结束
                                     内容的表单输入元素
$("input[name*='man']")            //选择具有属性 name 并且属性值包含'man'内容的表
                                     单输入元素
$("input[id][name$='man']")        //选择具有属性 id 和 name 并且 name 的值以'man'为
                                     结束内容的表单输入元素
```

6. 子元素选择器

使用这种选择器时，需要加入子元素的选择条件。例如：

```
$("ul li:nth-child(2)")            //选择第 2 个列表项
$("ul li: nth-child(even)")        //选择偶数索引列表项
$("ul li: nth-child(odd)")         //选择奇数索引列表项
$("ul li: nth-child(3n)")          //选择索引值为 3 的倍数的列表项
$("ul li:first-child")             //选择第一个列表项
$("ul li: last-child")             //选择最后一个列表项
$("ul li:only-child")              //选择列表出现且仅出现一个的列表项
```

7. 表单元素选择器

使用这种选择器时，需要加入代表不同表单元素类型的标识符。例如：

```
$(":input")                //选择所有 input, textarea, select 和 button 元素
$(":text")                 //选择当行文本框
$(":password")             //选择密码框
```

```
$(":radio")              //选择单选按钮
$(":checkbox")           //选择复选框
$(":submit")             //选择提交按钮
$(":image")              //选择所有图像域
$(":reset")              //选择重置按钮
$(":button")             //选择普通按钮
$(":file")               //选择文件域
$(":hidden")             //选择隐藏域
```

8. 表单属性选择器

使用这种选择器时，需要利用对表单属性的筛选操作符。例如：

```
$("input:enabled")              //选择所有可用元素
$("input:disabled")             //选择所有不可用元素
$("input:checked")              //选择所有被选中的单选按钮、复选框
$("select option:selected")     //选择所有被选中的 option
```

2.5.2　jQuery 事件

在 JavaScript 的事件模型中，一般是这样处理事件的：事件处理程序是通过把函数实例的引用指派到 DOM 元素的属性而声明的。定义这些属性用来处理特定类型的事件。例如，指派函数到 onclick 属性用来处理单击事件。指派函数到 onmouseover 属性用来处理 mouseover 事件，而元素支持这些事件类型。这种事件模型称为 DOM 0 模型。但是，这种模型有一定缺陷，即浏览器的差异性问题。

jQuery 弥补了浏览器中的差异性，提供了一种统一的事件模型，相对来说简单了许多。要添加一个事件处理程序，使用 bind(eventType,data,listener)方法。其中 eventType 是事件的名称，data 会被作为 event 对象的 data 属性附加到 event 对象，传给事件响应函数 listener()。data 可以省略，如果省略，第二个参数就是 listener。这个函数还是比较简单的。

jQuery 还提供了几种变形。对于常用的事件，可以采用 eventTypeName(listener)的函数来绑定，例如 $('#somediv').click(somefunction），这样就绑定了 click 事件的响应函数为 somefunction。one(listener)方法绑定一个事件处理函数，一旦此函数被执行过一次，就删除它。用 bind 命令绑定的事件处理程序被调用时，event 对象总是作为处理程序的第一个参数被传入的，这样也弥补了不同浏览器对于 event 的处理。下面把 jQuery 的事件处理功能大致介绍一下，具体使用会在后续章节中进行讲解。

jQuery 事件处理函数分类如下。

（1）页面载入 ready()

当 DOM 载入就绪可以查询及操纵时绑定一个要执行的函数。这是事件模块中最重要的函数，因为它可以极大地提高 Web 应用程序的响应速度。简单地说，这个方法纯粹是向 window.load 事件注册事件的替代方法。使用这个方法，可以在 DOM 载入就绪能够读取并操纵时立即调用你所绑定的函数，而 99.99%的 JavaScript 函数都需要在那一刻执行。

（2）事件处理 bind()

为每个匹配元素的特定事件绑定事件处理函数。bind()方法是用于往文档上附加行为的

主要方式。所有 JavaScript 事件对象，如 focus、mouseover 和 resize，都可以作为 type 参数传递进来。

（3）事件处理 one()

为每一个匹配元素的特定事件（像 click）绑定一个一次性的事件处理函数。在每个对象上，这个事件处理函数只会被执行一次。其他规则与 bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止（浏览器）默认的行为。如果既想取消默认的行为，又想阻止事件冒泡，这个事件处理函数必须返回 false。

（4）事件处理 trigger()

在每一个匹配的元素上触发某类事件。这个函数也会导致浏览器同名的默认行为的执行。例如，如果用 trigger()触发一个'submit'，则同样会导致浏览器提交表单。如果要阻止这种默认行为，应返回 false。

（5）事件处理 triggerHandler()

这个特别的方法将会触发指定的事件类型上所有绑定的处理函数，但不会执行浏览器默认动作，也不会产生事件冒泡。

（6）事件委派 live()

jQuery 给所有匹配的元素附加一个事件处理函数，即使这个元素是以后再添加进来的也有效。

（7）事件委派 die()

解除用 live 注册的自定义事件。

（8）事件切换 hover()

一个模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）的方法。这是一个自定义的方法，它为频繁使用的任务提供了一种“保持在其中”的状态。

（9）事件切换 toggle()

每次单击后依次调用函数。

（10）事件 blur()

触发每一个匹配元素的失去焦点事件。

（11）事件 change()

触发每一个匹配元素的状态改变事件

（12）事件 click()

触发每一个匹配元素的单击事件。

（13）事件 dbclick()

触发每一个匹配元素的双击事件。

（14）事件 error()

触发每一个匹配元素的 error 事件。

（15）事件 focus()

触发每一个匹配元素的获得焦点事件。

（16）事件 focusin()

当一个元素或者其内部任何一个元素获得焦点的时候会触发这个事件。它与 focus 事

件的区别在于，它可以在父元素上检测子元素获取焦点的情况。

（17）事件 focusout()

当一个元素或者其内部任何一个元素失去焦点的时候会触发这个事件。它与 blur 事件的区别在于，它可以在父元素上检测子元素失去焦点的情况。

（18）事件 keydown()

这个函数会调用执行绑定到 keydown 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。keydown 事件会在键盘按下时触发。

（19）事件 keypress()

这个函数会调用执行绑定到 keypress 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。keypress 事件会在键盘按下时触发。

（20）事件 keyup()

这个函数会调用执行绑定到 keyup 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。keyup 事件在按键释放时触发。

（21）事件 mousedown()

mousedown 事件在鼠标在元素上单击后触发。

（22）事件 mousemove()

mousemove 事件在鼠标在元素上移动时触发。事件处理函数会被传递一个变量——事件对象，其.clientX 和.clientY 属性代表鼠标的坐标。

（23）事件 mouseout()

mouseout 事件在鼠标从元素上离开后触发。

（24）事件 mouseover()

mouseover 事件在鼠标移入对象时触发。

（25）事件 mouseup()

mouseup 事件在鼠标单击对象释放时触发。

（26）事件 resize()

当文档窗口改变大小时触发。

（27）事件 scroll()

当滚动条发生变化时触发。

（28）事件 select()

这个函数会调用执行绑定到 select 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。

（29）事件 submit()

这个函数会调用执行绑定到 submit 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。

（30）事件 unload()

在每一个匹配元素的 unload 事件中绑定一个处理函数。

2.6　小　　结

本章介绍了与 jQuery 相关的基础知识。其中，JavaScript 及 DOM 部分是本章的重点，是后续章节的基础，同时这两部分也是本章难点。下一章将讲解如何利用 jQuery 操作基本元素 DIV。

第2篇 jQuery 实战开发与应用

- 第 3 章　DIV 层的控制
- 第 4 章　设计列表
- 第 5 章　网站导航
- 第 6 章　设计表格
- 第 7 章　设计表单
- 第 8 章　设计图片
- 第 9 章　设计对话框
- 第 10 章　设计滑动条
- 第 11 章　页面编辑器插件
- 第 12 章　多媒体
- 第 13 章　动画设计
- 第 14 章　拖动插件
- 第 15 章　插件开发

第 3 章　DIV 层的控制

在 HTML+CSS 的设计模式中，DIV 这个标记一直起着很大的作用。jQuery 在对页面元素进行操作时，很多情况下都有 DIV 直接或者间接参与。本章将讲解 jQuery 如何操作 DIV，为后续章节的学习打下基础。

3.1　DIV 的鼠标选取

jQuery 对于 DIV 的鼠标选取可以通过鼠标悬停和鼠标单击来实现。在 jQuery 的众多应用中能够准确选择 DIV 是关键。

3.1.1　利用鼠标悬停实现 DIV 的选取

一般在动态菜单或图片切换等应用中需要使用这种效果。其中，需要使用 jQuery 的函数 ready()、mouseover()。使用 ready()函数在文档加载完成后注册 DIV 的鼠标悬停事件，在鼠标悬停事件中做出响应。

1. jQuery函数ready()——文档加载完成

该函数表示当 DOM 载入就绪、可以查询及操纵时绑定一个要执行的函数。其语法形式如下：

```
ready(fn)
```

注：这是事件模块中最重要的一个函数，因为它可以极大地提高 Web 应用程序的响应速度。这个方法纯粹是对向 Window.load 事件注册事件的替代方法。通过使用这个方法，可以在 DOM 载入就绪、能够读取并操纵时立即调用所绑定的函数。

2. jQuery函数mouseover()——鼠标悬停事件

该函数在每一个匹配元素的鼠标悬停事件中绑定一个处理函数。其语法形式如下：

```
mouseover(fn)
```

注：fn 表示需要绑定的函数。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#div1").mouseover(function(){          //鼠标悬停事件
          alert("层被选择");
      });
  });
</script>
```

图 3.1　鼠标选择 DIV 层

效果如图 3.1 所示。上面的示例使用 jQuery 的 mouseover()函数实现鼠标对层的悬停选取。在 jQuery 中还有一个函数也可以作为鼠标悬停事件处理函数使用，即 hover()函数。它和 mouseover()的区别是：hover()不仅可以模拟鼠标的悬停，也会对鼠标的离开做出响应，并修正了鼠标离开 mouseout()的一些错误。

3．jQuery函数hover()——鼠标悬停/离开切换事件

该方法模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）。其语法形式如下：

```
hover(over,out)
```

注：当鼠标移动到一个匹配的元素上面时，会触发指定的第一个函数。当鼠标移出这个元素时，会触发指定的第二个函数。而且，会伴随着对鼠标是否仍然处在特定元素中的检测，如果是，则会继续保持“悬停”状态，而不触发移出事件（修正了使用 mouseout 事件的一个常见错误）。over，鼠标悬停在元素上的函数，out，鼠标离开元素的函数。

我们用 hover()替换 mouseover()：

```
<script type="text/javascript">
  $(function(){
      $("#div1").hover(function(){
      //鼠标悬停/离开切换事件
          alert("层被选择");
      },
      function(){
          alert("层被反选");
      }
    );
});
</script>
```

图 3.2　鼠标离开 DIV 层

因为 hover()可以对两个事件都做出响应，所以在 hover()中使用了两个 function 参数，第一个是鼠标悬停事件，第二个是修正了的鼠标离开事件。当鼠标悬停在 DIV 上时效果如图 3.1 所示，当鼠标离开 DIV 时效果如图 3.2 所示。

3.1.2　利用鼠标单击实现 DIV 的选取

有时我们不希望鼠标悬停就选取一个DIV，而是希望当鼠标对DIV进行单击时选择它。如果是这样，需要使用鼠标的单击事件来操作对 DIV 的选取。其中，需要使用 jQuery 函数 ready()、click()。使用 ready()函数在文档加载完成后注册 DIV 的鼠标单击事件，在鼠标单击事件中做出响应。

1．jQuery函数click()——鼠标单击事件

该函数触发每一个匹配元素的单击事件。其语法形式如下：

```
click([fn])
```

注：这个函数会调用执行绑定到 click 事件的所有函数。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#div1").click(function(){
        //鼠标单击事件
4           alert("单击层被选择");
5       });
6   });
7 </script>
```

图 3.3　鼠标单击选择 DIV 层

效果如图 3.3 所示。在 jQuery 中还有一个事件响应函数 toggle()。它对鼠标单击的不同次数进行响应，而不仅仅是单击事件。当发生多次单击的时候，每次单击事件都可以用这个函数做处理操作，它和上面所介绍的 hover()函数同属于事件切换函数。这里我们想要实现当鼠标第一次单击 DIV 的时候选取，第二次单击 DIV 的时候撤销选取。

2．jQuery函数toggle()——单击切换

该函数表示每次单击后依次调用函数。其语法形式如下：

```
toggle(fn1,fn2,[fn3,fn4,…])
```

注：如果单击了一个匹配的元素，则触发指定的第一个函数，当再次单击同一元素时，则触发指定的第二个函数，如果有更多函数，则再次触发，直到最后一个。随后的每次单击都重复对这几个函数的轮番调用。

```
1 <script type="text/javascript">
2   $(function(){
```

```
3        $("#div1").toggle(function(){
         //鼠标单击事件切换
4                alert("单击层被选择");
5            },
6            function(){
7                alert("再次单击层被反选");
8            }
9        );
10   });
11</script>
```

在 toggle()中添加了两个 function()函数，第一个表示鼠标第一次单击层（如果后面还有多次单击，则它表示奇数次单击）。第二个表示鼠标第二次单击层（它表示偶数次单击）。第一次单击效果如图 3.3 所示，第二次单击效果如图 3.4 所示。

图 3.4　再次单击取消 DIV 的选择

3.2　DIV 层的尺寸

jQuery 可以对 DIV 层的尺寸进行动态读取和动态修改。本节介绍如何通过 jQuery 控制 DIV 的尺寸。

3.2.1　jQuery 动态读取 DIV 层的尺寸

jQuery 在动态读取 DIV 的高和宽的时候需要两个固有函数 height()、width()。这两个函数分别获取匹配元素对象的高和宽。可以在 DIV 的单击事件中使用这两个函数来取值。

1．jQuery函数height()——元素高度

该函数获取或者设置元素的高度，单位为像素。其语法形式如下：

```
height([val])
```

注：该函数也可以获取 window 和 document 的高。

2．jQuery函数width()——元素宽度

该函数获取或者设置元素的宽度，单位为像素。其语法形式如下：

```
width([val])
```

注：该函数也可以获取 window 和 document 的宽。

HTML 代码和 CSS 样式参考光盘内容。这里直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#div1").click(function(){
          var width=$(this).width();          //获取元素宽度
          var height=$(this).height();        //获取元素高度
          alert("层的宽: "+width+"px"+"层的高: "+height+"px");
      });
  });
</script>
```

图 3.5　单击获取层的尺寸

上述代码中的$(this)就代表$("#div1")，效果如图 3.5 所示。

上面的代码是直接通过尺寸函数来获取层的大小，也可以通过 css()函数来获取大小。但是，有一点需要注意：尺寸函数返回的值是整型数字，但 css()函数返回的是带有 px 像素单位的字符串。

3．jQuery函数css()——样式设定

该函数获取或设定匹配元素的 CSS 样式。其语法格式如下：

```
css(name)
css(properties)
css(name,value|fn)
```

注：properties 表示要设置为样式属性的名/值对；fn 函数返回要设置的属性值，接受两个参数，index 为元素在对象集合中的索引位置，value 为原先的属性值。

我们可以将上面的代码稍做修改：

```
<script type="text/javascript">
  $(function(){
      $("#div1").click(function(){
          var width=$(this).css("width");     //通过 CSS 函数获取元素宽度
          var height=$(this).css("height");   //通过 CSS 函数获取元素高度
          alert("层的宽: "+width+"px"+"层的高: "+height+"px");
      });
  });
</script>
```

3.2.2　jQuery 动态修改 DIV 层的尺寸

jQuery 动态修改 DIV 层的尺寸同样可以使用前面提到的尺寸函数，只是这里需要将参数值带给这两个函数。在下面的示例中用两个文本框接收用户动态输入 DIV 的宽和高，并当用户单击“修改尺寸”按钮时，修改 DIV 的尺寸。

1. JavaScript功能实现

HTML 代码和 CSS 样式参考光盘内容。这里直接看一下 JavaScript 功能实现：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           var width=$("#divwidth").val();
5           var height=$("#divheight").val();
6           $("#div1").width(width).height(height);
            //通过高度和宽度函数设定元素高和宽
7       });
8   });
9 </script>
```

效果如图 3.6 和图 3.7 所示。

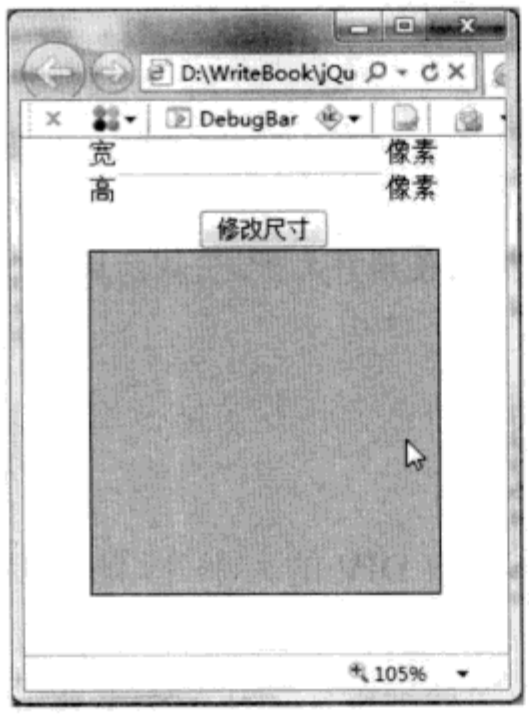


图 3.6　DIV 初始尺寸大小

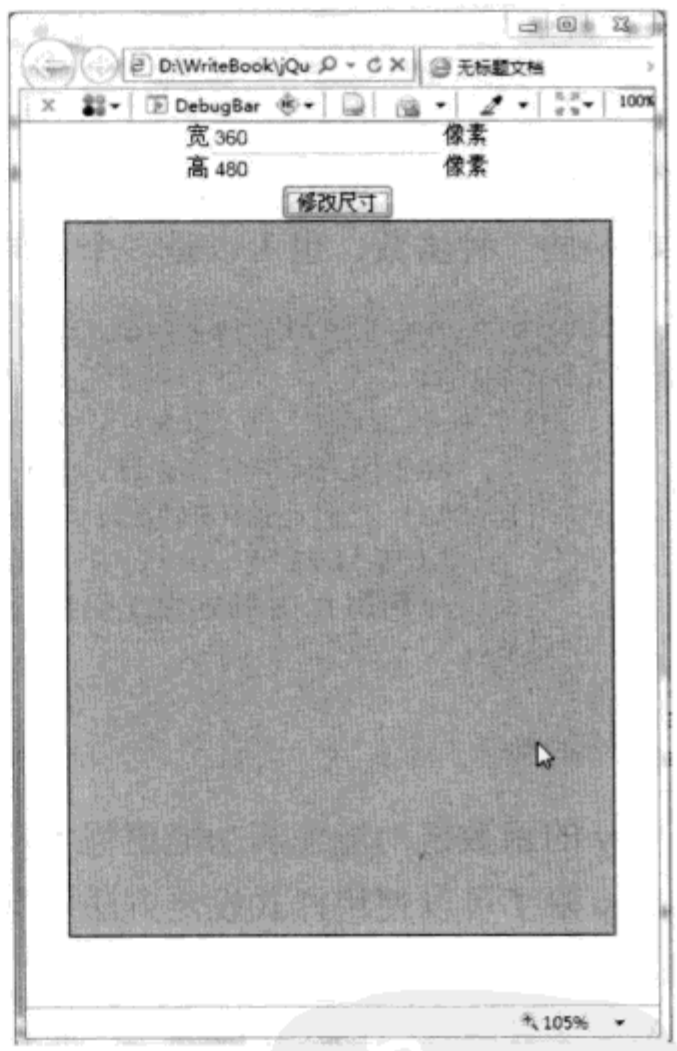


图 3.7　动态修改 DIV 的尺寸

2. 通过css()函数修改DIV尺寸

与上面获取尺寸操作一样，也可以通过 css()函数来修改 DIV 的尺寸。修改后代码如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           var width=$("#divwidth").val();
5           var height=$("#divheight").val();
6           $("#div1").css({"width":width+"px","height":height+"px"});
```

```
            //通过 CSS 样式设定函数设定元素的高和宽
7       });
8   });
9 </script>
```

这里使用了 css()函数属性对象的形式设定了 DIV 层的大小。jQuery 的一个主要特点是函数链，所谓函数链，就是可以将多个元素操作函数以链条的形式接在元素对象后面，从而完成对元素对象的顺序多重操作。例如，可以使用刚才的示例以函数链的形式编写：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           var width=$("#divwidth").val();
5           var height=$("#divheight").val();
6           $("#div1").css("width",width+"px").css("height",height+"px");
            //两次调用 CSS 样式设定函数链接使用方式设定元素的高度和宽度
7       });
8   });
9 </script>
```

上述代码用两个 css()函数链接在 DIV 元素对象后面实现了对尺寸大小的修改。这种函数链可以是同一种函数，也可以是多个不同的函数。例如：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           var width=$("#divwidth").val();
5           var height=$("#divheight").val();
6           $("#div1").css("width",width+"px").height(height);
            //利用 CSS 样式设定函数与高度设定函数链接使用方式设定元素的高度和宽度
7       });
8   });
9 </script>
```

jQuery 的函数链功能使其功能更强大，代码更简洁。

jQuery 除了可以使用样式设定方法及尺寸方法动态修改 DIV 的大小外，还可以使用自定义动画来实现 DIV 的大小变化。在自定义动画中可以将 DIV 要修改的大小尺寸传递给函数，由函数以动画的形式完成变化，而且可以设定一次动画操作完成所有变化，也可以分多次完成。

3. jQuery函数animate()——自定义动画

该函数用于创建自定义动画。其语法形式如下：

```
animate(params,[duration],[easing],[callback])
animate(params,options)
```

注：这个函数的关键在于指定动画形式及结果样式属性对象。这个对象中每个属性都表示一个可以变化的样式属性（如"height"、"top"或"opacity"）。params，一组包含动

画属性和终值的样式属性及其值的集合；duration，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；easing，要使用的擦除效果的名称（需要插件支持），jQuery 默认提供"linear"和"swing"；callback，在动画完成时执行的函数；options，一组包含动画选项的值的集合。

下面这段代码可以完成一次动画将 DIV 的大小变化完成：

```
<script type="text/javascript">
  $(function(){
      $("#update").click(function(){
          var width=$("#divwidth").val();
          var height=$("#divheight").val();

  $("#div1").animate({width:width+"px",height:height+"px"},4000);
  //利用自定义动画函数动态显示元素宽度和高度的更改
      });
  });
</script>
```

具体完成效果和前面的相同。也可分两次完成动画：

```
<script type="text/javascript">
  $(function(){
      $("#update").click(function(){
          var width=$("#divwidth").val();
          var height=$("#divheight").val();
          $("#div1").animate({width:width+"px"},4000).animate({height:
          height+"px"},4000);
      });
  });
</script>
```

上述代码的效果是 DIV 先横向变化，到达设定值后再纵向变化。

3.3　层的显示与隐藏

本节介绍如何利用 jQuery 控制 DIV 的显示与隐藏。实际上 DIV 的显示与隐藏可以通过多种途径实现，我们从最简单的开始介绍。

3.3.1　利用 jQuery 的显示与隐藏函数实现

1．显示和隐藏函数

在 jQuery 的众多函数中，专门提供了操作元素显示和隐藏的函数 show()、hide()。

（1）jQuery 函数 show()——显示元素

该函数显示隐藏的元素。其语法形式如下：

```
show()
show(speed,[callback])
```

注：如果选择的元素是可见的，这个方法将不会改变任何东西。无论这个元素是通过 hide()方法隐藏的还是在 CSS 里设置了 display:none;，这个方法都将有效。speed，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次。

（2）jQuery 函数 hide()——隐藏元素

该函数隐藏显示的元素。其语法形式如下：

```
hide()
hide(speed,[callback])
```

注：如果选择的元素是隐藏的，这个方法将不会改变任何东西。speed，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次。

HTML 代码和 CSS 样式参考光盘内容。这里看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#update").toggle(function(){
              $("#div1").hide();      //隐藏元素
          },
          function(){
              $("#div1").show();      //显示元素
          }
      );
  });
</script>
```

上述代码简单实现了 DIV 的显示与隐藏的切换。页面初始加载及按钮的偶数次单击时效果如图 3.8 所示，奇数次单击“显示/隐藏”按钮后效果如图 3.9 所示。

图 3.8　DIV 的显示

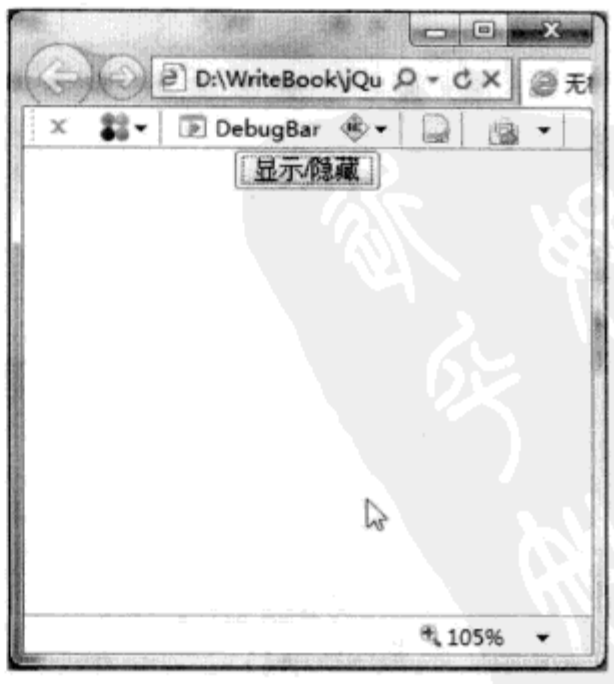


图 3.9　DIV 的隐藏

刚才的示例使用了显示与隐藏函数的基本形式。实际上这两个函数还有变形形式，即添加了显示与隐藏速度的设定及附加了回调函数。我们可以将上述代码修改如下：

```
<script type="text/javascript">
  $(function(){
      $("#update").toggle(function(){
            $("#div1").hide(4000,function(){alert("You can't see me!");});
            //设定隐藏元素过程的时间跨度，并利用消息提示框显示附加信息
        },
        function(){
            $("#div1").show(4000,function(){alert("You can see me!");});
            //设定显示元素过程的时间跨度，并利用消息提示框显示附加信息
        }
      );
  });
</script>
```

上述代码在显示与隐藏函数中加入了时间跨度 4000 毫秒，并在层的显示及隐藏后出现提示对话框。隐藏效果是 DIV 向中间逐渐缩小，显示效果是 DIV 由中间逐步放大到预定大小，效果如图 3.10 和图 3.11 所示。

图 3.10　DIV 隐藏

图 3.11　DIV 显示

2. 转换函数

前面讲解了转换函数 toggle()，这个函数也可对层的显示状态进行转换。其语法形式如下：

```
toggle()
toggle(switch)
toggle(speed,[callback])
```

注：如果 switch 设为 true，则调用 show()方法来显示匹配的元素，如果 switch 设为 false，则调用 hide()来隐藏元素。speed，三种预定速度之一的字符串（"slow"、"normal"

或"fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次。

下面先来看最基础的应用。将上面的代码稍做修改：

```
<script type="text/javascript">
  $(function(){
      $("#update").click(function(){
          $("#div1").toggle();                    //切换元素的显示状态
      });
  });
</script>
```

toggle()函数在这里就是转换 DIV 的显示状态的。如果 DIV 是显示的则隐藏，反之则显示，效果如图 3.8 和图 3.9 所示。这个函数还有变形形式。例如，给定需要转换的状态，这个通过给函数设定参数来实现。将上面的代码进行再次修改：

```
<script type="text/javascript">
  $(function(){
      $("#update").toggle(function(){
          $("#div1").toggle(false);   //指定将元素的显示状态切换成隐藏
      },
      function(){
          $("#div1").toggle(true);    //指定将元素的显示状态切换成显示
      }
      );
  });
</script>
```

当参数为真时，相当于调用 show()函数，为假时调用 hide()。这个函数也可以指定状态切换速度及附加的回调函数，代码如下：

```
<script type="text/javascript">
  $(function(){
      $("#update").click(function(){
          $("#div1").toggle(4000,function(){alert("Change Visible
          Status");});    //指定元素显示状态切换过程的时间跨度，并显示附加信息
      });
  });
</script>
```

效果如图 3.12 和图 3.13 所示。

3.3.2　利用 jQuery 的滑动效果实现

在 jQuery 的众多特效函数中，可以利用滑动函数来操作 DIV 的显示与隐藏。

1. jQuery函数slideUp()——滑动函数

该函数向上减小高度动态隐藏所有匹配的元素。其语法形式如下：

```
slideUp(speed,[callback])
```

图 3.12　DIV 的隐藏

图 3.13　DIV 的显示

注：speed，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次。

2．jQuery函数slideDown()——滑动函数

该函数向下增大高度动态显示所有匹配的元素。其语法形式如下：

```
slideDown(speed,[callback])
```

注：参数同上面的介绍。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#update").toggle(function(){
          $("#div1").slideUp(4000,function(){alert("Slide Up DIV");});
          //利用向上滑动函数设定元素向上滑动及时间跨度，并显示附加信息
        },
        function(){
          $("#div1").slideDown(4000,function(){alert("Slide  Down
          DIV");});//利用向下滑动函数设定元素向下滑动及时间跨度，并显示附加信息
        }
      );
  });
</script>
```

上述代码中的 slideUp()和 slideDown()这两个函数实现 jQuery 特效函数的向上滑动与向下滑动。在这里给出了效果的持续时间和附加回调函数，效果如图 3.14 和图 3.15 所示。

和前面提到的切换函数类似，对于滑动操作，也有相应的滑动切换函数 slideToggle()。它在滑上和滑下两种状态下切换。

图 3.14　DIV 滑动隐藏

图 3.15　DIV 滑动显示

3. jQuery函数slideToggle()——切换高度变化

该函数通过高度变化来切换所有匹配元素的可见性。其语法形式如下：

```
slideToggle(speed,[callback])
```

注：参数同上面的介绍。

可以将上述代码修改如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           $("#div1").slideToggle(4000,function(){alert("Change Visible
            Status");});    //利用滑动方向切换函数切换元素滑动动作及时间跨度，并显
                              示附加信息
5       });
6   });
7 </script>
```

上述代码就是切换滑动动作函数的用法，效果和前面的 toggle()函数相同。

3.3.3　利用 jQuery 的淡入淡出效果实现

前面提到的滑动效果是通过改变 DIV 的高来实现显示与隐藏操作。还可以通过改变 DIV 的透明度来实现显示与隐藏。这需要使用到 jQuery 的淡入淡出函数 fadeIn()、fadeout()。

1. jQuery函数fadeIn()——淡入效果

该函数通过不透明度变化实现元素的淡入。其语法形式如下：

```
fadeIn(speed,[callback])
```

注：speed，三种预定速度之一的字符串（"slow"，"normal"or "fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次。

2. jQuery函数fadeOut()——淡出效果

该函数通过不透明度变化实现元素的淡出。其语法形式如下：

```
fadeOut(speed,[callback])
```

注：speed，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如：1000）；callback，在动画完成时执行的函数，每个元素执行一次。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").toggle(function(){
4               $("#div1").fadeOut(4000,function(){alert("Fade Out DIV");});
                //设定元素淡出效果，并加入淡出时间跨度及显示附加信息
5           },
6           function(){
7               $("#div1").fadeIn(4000,function(){alert("Fade In DIV");});
                //设定元素淡入效果，并加入淡入时间跨度及显示附加信息
8           }
9       );
10  });
11</script>
```

上述代码实现的效果如图 3.16 和图 3.17 所示。

图 3.16　DIV 的淡出隐藏

图 3.17　DIV 的淡入显示

淡入淡出效果也可用一个转换单独实现，它的实现是依靠 DIV 的不透明属性的变化来完成的。

3. jQuery函数fadeTo()——淡入淡出切换

该函数把所有匹配元素的不透明度以渐进方式调整到指定的不透明度。其语法形式如下：

```
fadeTo(speed,opacity,[callback])
```

注：speed，三种预定速度之一的字符串（"slow"、"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；callback，在动画完成时执行的函数，每个元素执行一次；opacity，要调整到的不透明度。

可以将上面的代码修改如下：

```
<script type="text/javascript">
  $(function(){
      $("#update").toggle(function(){
              $("#div1").fadeTo(4000,0,function(){alert("Fade Out DIV");});
              //利用元素淡入淡出动作切换函数，设定元素淡出效果，并加入淡出时间跨度
                及显示附加信息
          },
          function(){
              $("#div1").fadeTo(4000,1,function(){alert("Fade In DIV");});
              //利用元素淡入淡出动作切换函数，设定元素淡入效果，并加入淡入时间跨度
                及显示附加信息
          }
      );
  });
</script>
```

上述代码中使用了 fadeTo()函数，这是一个调整元素透明度的函数，透明度参数为这个函数的第二个参数位置，透明度取值范围在 0～1 之间，0 表示彻底透明，1 表示不透明，效果如图 3.16 和图 3.17 所示。

3.4　DIV 内的内容控制

对于 DIV 的内容控制介绍如下几种操作：内容清空、内容替换、内容复制、内容添加、内容包装等。这些操作都涉及 jQuery 的文档处理方法。

3.4.1　内容清空

首先来介绍 DIV 中的内容清空。在 jQuery 的文档操作中具有清空文档子内容的方法 empty()。这里可以利用它来完成内容清空。

该函数删除匹配的元素集合中所有的子节点。其语法形式如下：

```
empty()
```

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#update").click(function(){
          $("#div1").empty();     //清空元素内容
      });
  });
</script>
```

当单击页面上的按钮时，jQuery 就会清空 DIV 下所有的子内容，效果如图 3.18 和图 3.19 所示。

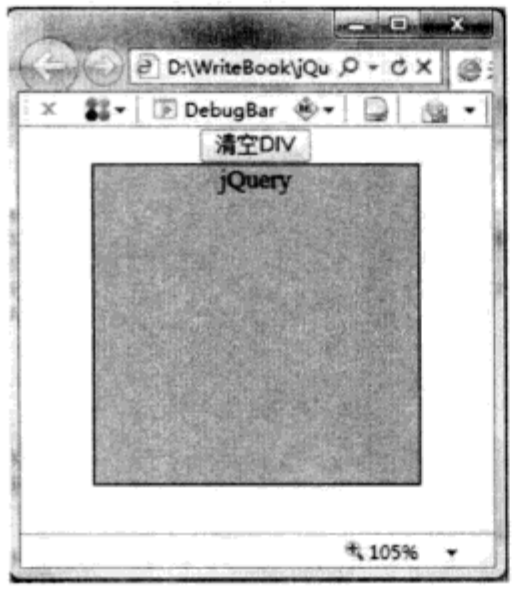


图 3.18　DIV 原有内容

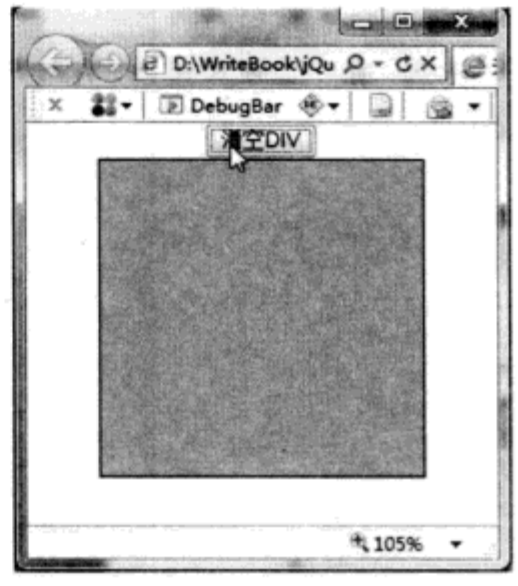


图 3.19　清空 DIV 内容

3.4.2　内容替换

内容替换操作使用了 jQuery 工具函数中的 replaceWith()函数。这个函数用参数携带的内容来替换匹配元素的内容。

1. jQuery函数replaceWith()——替换元素内容

该函数将所有匹配的元素替换成指定的 HTML 或 DOM 元素。其语法形式如下：

```
replaceWith(content)
```

注：content，用于替换匹配元素的内容。如果这里传递一个函数进来，则函数返回值必须是 HTML 字符串。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           $("#div1").children().replaceWith("<b>Paragraph. </b>");
            //将元素内容替换为指定内容
5       });
6   });
7 </script>
```

当单击按钮时，DIV 原有内容被替换成粗体的 Paragraph，效果如图 3.20 所示。

和 replaceWith()实现效果类似，但操作不同的是 relaceAll()函数。relaceWith()是由被替换的对象调用，替换内容为参数。而 relaceAll()则是由替换内容调用，参数是被替换的匹配元素。

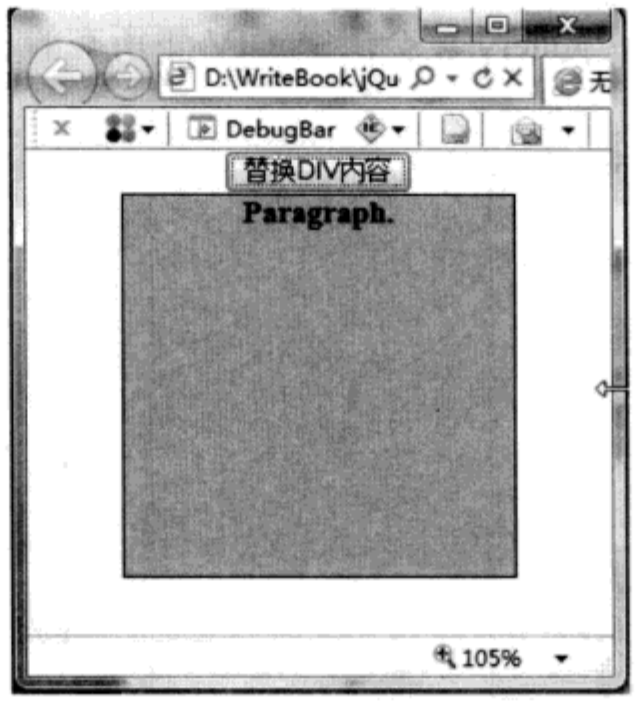


图 3.20　DIV 内容替换

2．jQuery函数replaceAll()——元素替换

该函数用匹配的元素替换掉所有 selector 匹配到的元素。其语法形式如下：

```
replaceAll(selector)
```

可以将上面的代码修改如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#update").click(function(){
4           $("<b>Paragraph. </b>").replaceAll($("#div1").children());
            //利用指定内容去替换匹配元素中的内容
5       });
6   });
7 </script>
```

效果和图 3.20 一致。

3.4.3　内容复制

有时需要将 DIV 中的内容复制出来进行其他操作。在后面的章节中会多次用到这个功能。jQuery 进行内容复制使用 clone()函数，它将匹配的所有元素复制出来并返回它们。

该函数克隆匹配的 DOM 元素（及其事件）并且选中这些克隆的副本。其语法形式如下：

```
clone()
clone(true)
```

注：设置为 true 以便复制元素的所有事件处理。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
1<script type="text/javascript">
2   $(function(){
```

```
3        $("#update").click(function(){
4            $("#clonetxt").text($("#div1").children().clone().text());
             //复制 div1 中的子元素，并将其文本填入 clonetxt
5        });
6    });
7</script>
```

上面的代码实现了单击按钮后将 DIV 中的复制出来并将其文本内容显示到文本区域内，效果如图 3.21 所示。

下面将前面的替换函数和这里介绍的复制函数组合起来使用。这个示例所要达到的目的是首先复制出 DIV 原有内容，然后在复制出来的对象上进行文本替换，再用复制出来的对象替换原有 DIV 中的内容。实现功能的 JavaScript 代码如下：

```
1 <script type="text/javascript">
2   $(function(){
3        $("#update").click(function(){
4            var txt=$("#clonetxt").text();
5            $("#div1").children().replaceWith($("#div1").children().
             clone().text(txt)); //复制 div1 中的子元素，并用其文本替换掉 clonetxt
                                 中的原有内容
6        });
7   });
8 </script>
```

具体效果如图 3.22 所示。

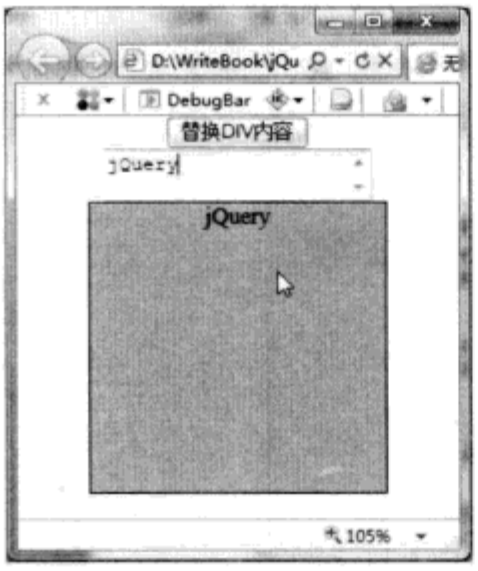


图 3.21　复制 DIV 内容

图 3.22　用复制内容替换原有内容

3.4.4　内容添加

jQuery 本身对于内容的添加分成内部添加和外部添加两种，每种方式又都有前后两种添加位置。

1．向内部添加函数

首先来看内部添加如何操作。DIV 的内部添加需要用到 jQuery 的 append()、prepend() 函数。这两个函数分别负责在 DIV 内容的后端和前端添加内容。

（1）jQuery 函数 append()——添加元素内容

该函数向每个匹配的元素内部追加内容。其语法形式如下：

```
append(content|fn)
```

（2）jQuery 函数 prepend()——添加元素内容

该函数向每个匹配的元素内部前置内容。其语法形式如下：

```
prepend(content|fn)
```

下面用一个示例来演示如何向 DIV 中添加内容。HTML 代码和 CSS 样式参考光盘内容。这里直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#append").click(function(){
          $("#div1").append($("#txt").val());
                                        //将文本框中的内容追加到 div1 尾部
      });
      $("#prepend").click(function(){
          $("#div1").prepend($("#txt").val());
                                        //将文本框中的内容添加到 div1 头部
      });
  });
</script>
```

上述代码的执行效果如图 3.23 所示。

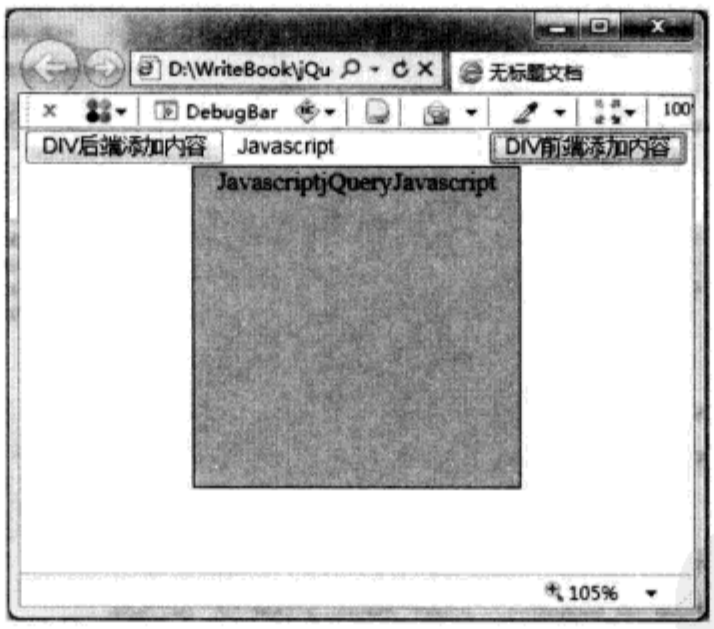


图 3.23　向 DIV 内添加内容一

和 append()、prepend()这两个函数有相同效果的还有 appendTo()、prependTo()。只不过这两个函数的调用者是被添加的元素，而参数是 DIV。

（3）jQuery 函数 appendTo()——添加元素内容

该函数把所有匹配的元素追加到另一个指定的元素集合中。其语法形式如下：

```
appendTo(content)
```

（4）jQuery 函数 prependTo()——添加元素内容

该函数把所有匹配的元素前置到另一个、指定的元素集合中。其语法形式如下：

```
prependTo(content)
```

可以修改上面的代码如下：

```
<script type="text/javascript">
  $(function(){
      $("#append").click(function(){
          $("<p>"+$("#txt").val()+"</p>").appendTo($("#div1"));
          //将指定内容追加到 div1 尾部
      });
      $("#prepend").click(function(){
          $("<p>"+$("#txt").val()+"</p>").prependTo($("#div1"));
          //将指定内容内容添加到 div1 头部
      });
  });
</script>
```

上述代码的效果如图 3.24 所示。

2. 向外部添加函数

前面介绍了如何使用内部添加函数向 DIV 中添加内容。下面来看一下如何使用外部添加函数。外部添加函数包括 after()、before()。这两个函数表示对当前匹配的元素做添加操作。

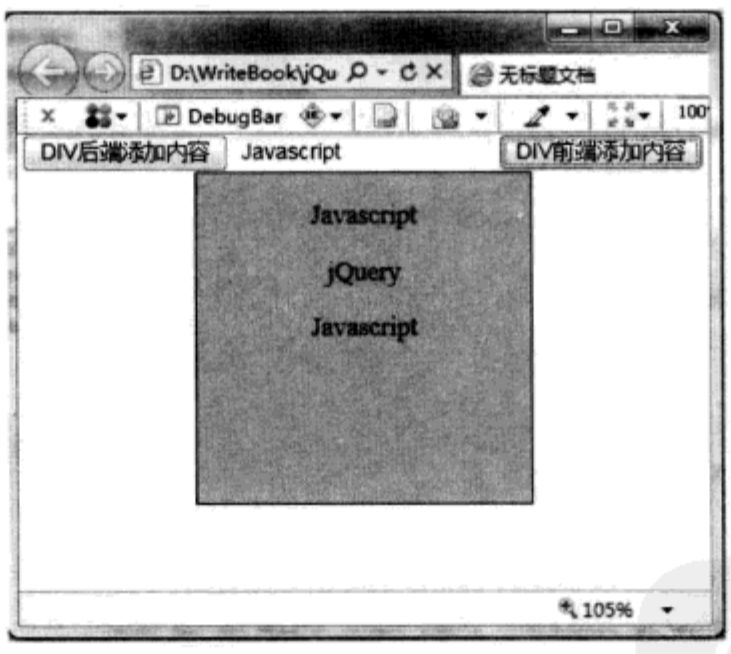


图 3.24　向 DIV 内添加内容二

（1）jQuery 函数 after()——外部添加元素内容

该函数在每个匹配的元素之后插入内容。其语法形式如下：

```
after(content|fn)
```

注：fn 函数必须返回一个 HTML 字符串。

（2）jQuery 函数 before()——外部添加元素内容

该函数在每个匹配的元素之前插入内容。其语法形式如下：

```
before(content|fn)
```

注：fn 函数必须返回一个 HTML 字符串。

在使用这两个函数的时候需要先获取 DIV 内的子元素才可进行操作。可以将上面的代码修改如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#append").click(function(){
4           $("#div1").children().after($("#txt").val());
            //在 div1 的子元素后插入文本框内容
5       });
6       $("#prepend").click(function(){
7           $("#div1").children().before($("#txt").val());
            //在 div1 的子元素前插入文本框内容
8       });
9   });
10</script>
```

上述代码获取 DIV 内的子元素，并在子元素外添加内容来实现在 DIV 中添加内容，效果如图 3.25 所示。

对于外部插入函数，也有类似于内部插入函数的反调用形式，即插入内容调用方法，如 insertAfter()、insertBefore()。这两个函数就是待插入的内容调用函数，插入位置作为参数传递给函数。

（3）jQuery 函数 insertAfter()——外部插入元素内容

该函数把所有匹配的元素插入到另一个指定的元素集合的后面。其语法形式如下：

```
insertAfter(content)
```

（4）jQuery 函数 insertBefore()——外部插入元素内容

该函数把所有匹配的元素插入到另一个指定的元素集合的前面。其语法形式如下：

```
insertBefore(content)
```

可以将上面的代码修改如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#append").click(function(){
4           $("<p>"+$("#txt").val()+"</p>").insertAfter($("#div1").
            children());          //在 div1 的子元素后插入文本框内容
5       });
6       $("#prepend").click(function(){
7           $("<p>"+$("#txt").val()+"</p>").insertBefore($("#div1").
            children());          //在 div1 的子元素前插入文本框内容
8       });
9   });
10</script>
```

效果如图 3.25 和图 3.26 所示。

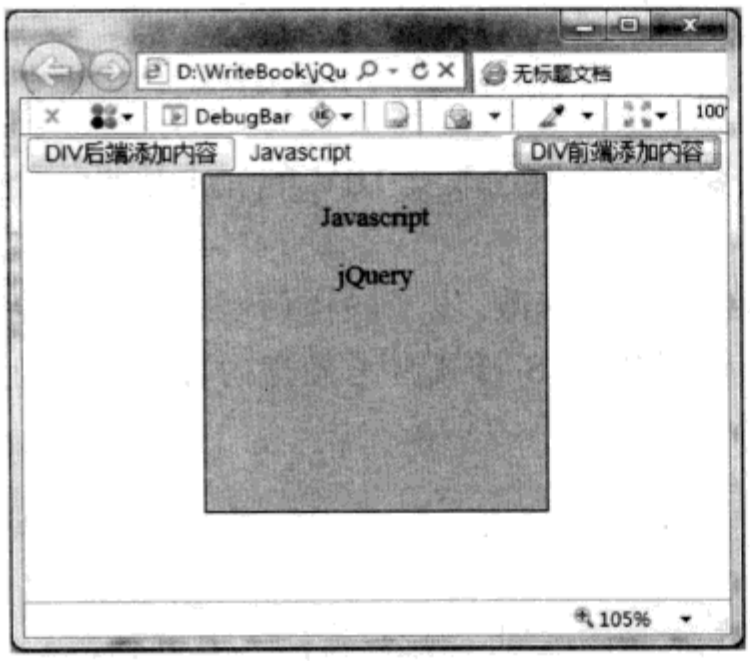


图 3.25　DIV 利用外部插入添加内容一

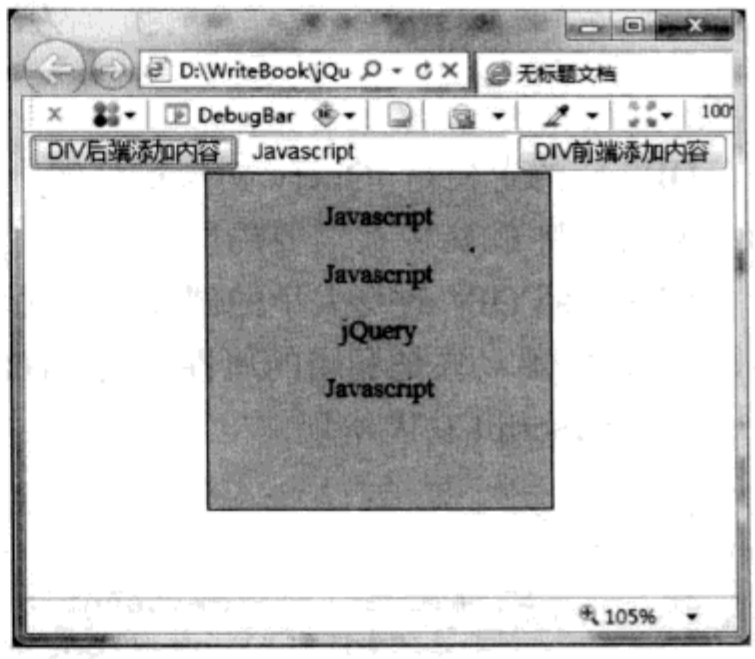


图 3.26　DIV 利用外部插入添加内容二

3.4.5　内容包装

所谓内容包装，实际上是在 DIV 中的内容外层再包裹上一层标记。内容包装需要使用 jQuery 的工具函数 wrap()。例如，我们想利用动态包装操作将 DIV 中原有的文本转换成文字超链接。

该函数把所有匹配的元素用其他元素的结构化标记包裹起来。其语法形式如下：

```
wrap(html)
wrap(elem)
wrap(fn)
```

注：html，HTML 标记代码字符串，用于动态生成元素并包裹目标元素；elem，用于包装目标元素的 DOM 元素；fn，生成包裹结构的一个函数。

HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#wrap").click(function(){
4           $("#div1").wrapInner("<a href=
    'http://jQuery.com'></a>");
            //在 div1 内包装了一个超链接元素
5       });
6   });
7 </script>
```

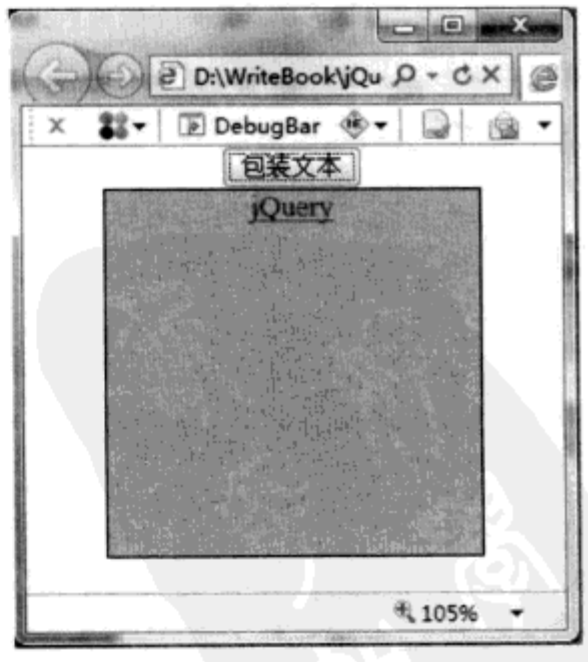


图 3.27　DIV 内元素包装

效果如图 3.27 所示。

3.5　层 的 定 位

DIV 层的定位是 jQuery 对层操作的一个主要应用。通过改变 DIV 的左上角坐标能够随心所欲地将 DIV 放在需要的位置。

前面介绍 DIV 改变大小的部分可以通过 css()函数来修改。本节同样可以通过这个函数修改 DIV 位置，只是利用的属性不同。HTML 代码和 CSS 样式参考光盘内容。下面直接看一下 JavaScript 功能实现：

```
<script type="text/javascript">
  $(function(){
      $("#move").click(function(){
          $("#div1").css({position:"absolute",top:$("#Y_index").val()+
          "px",left:$("#x_index").val()+"px"});
          //通过 CSS 样式设定函数定位 div1 在浏览器中的位置
      });
  });
</script>
```

上述代码通过在 css()函数中设定 DIV 的左上角的坐标位置进行定位，效果如图 3.28 所示。

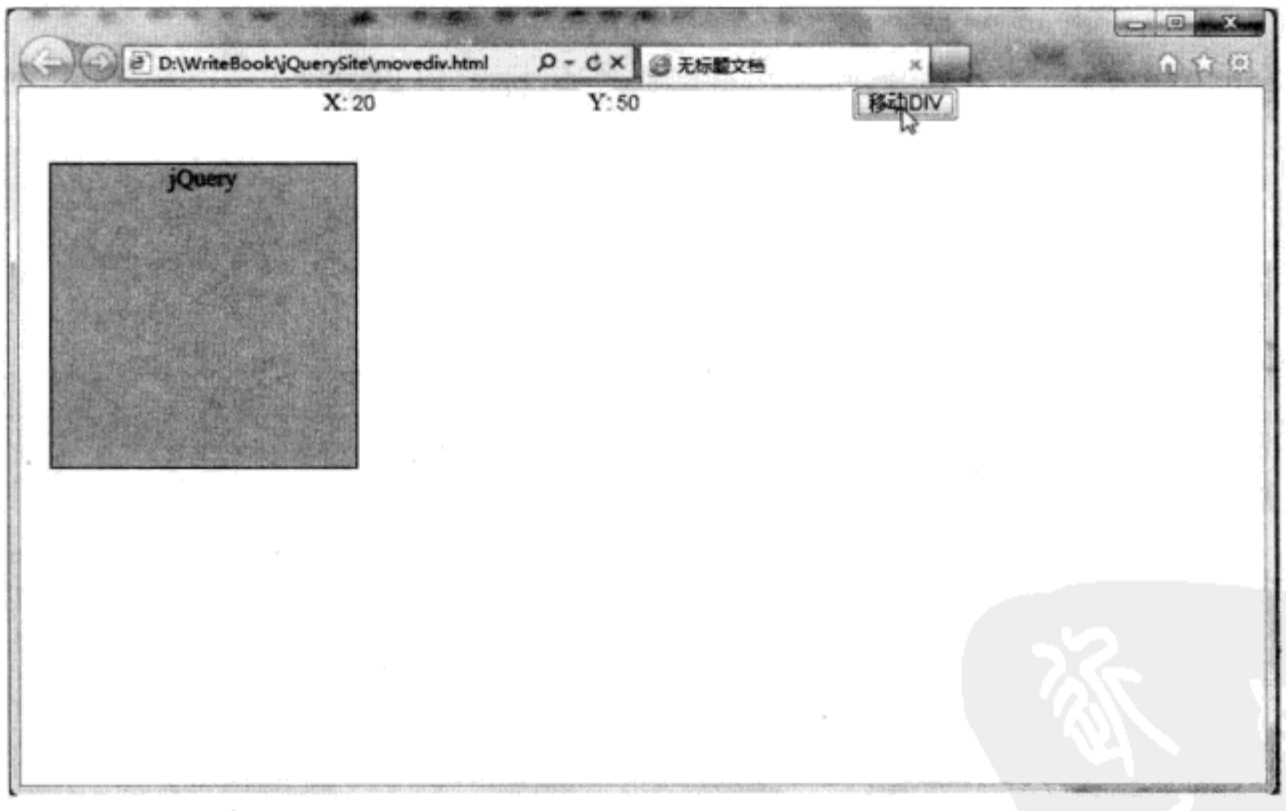


图 3.28　改变 DIV 的位置

3.6　小　　结

本章主要介绍了 jQuery 对 DIV 层的常用操作。重点内容是 DIV 层的显示与隐藏、DIV

层改变大小、DIV 层定位、DIV 层内容的包装。DIV 层内容的包装是本章的难点部分。下一章将讲解列表的设计。

3.7　习　　题

【习题一】利用本章所讲解的内容实现每间隔 2 秒修改一个层的显示状态。

【习题二】利用本章所讲解的内容实现每间隔 10 秒检查用户在文本框中输入的数字和层中显示的数字是否相同，如果不同，则用文本框中的数字替换层中显示的数字。

【习题三】利用本章所讲解的内容实现当用户点击按钮时将页面上普通文本文字“新浪”包装成带超链接功能的链接文字。

第4章　设计列表

列表是网页陈述信息的一种方式，是由一定格式的文字和图片元素构成的。在静态页面中可以设计列表的内容、列表的嵌套等。但是，对于列表的其他一些特性控制起来相对困难。本章将讲解如何使用 jQuery 更有效地控制列表。

4.1　控制列表宽度

在实际应用中，创建列表有多种不同的形式。在这里列举两种创建形式：一是直接使用 HTML 标记中的<OL>或者搭配使用<UL>与<LI>创建列表；二是使用<DIV>的嵌套来创建。下面将讲解以这两种形式创建的列表如何控制列表宽度。

4.1.1　参差不齐的列表

在创建列表的时候，由于不同列表项的文字内容不同，必然造成列表项宽度不同。例如下面这个静态页面：

```
1 <div>
2      <ul>
3          <li><span class="list"> jQuery 是一个 JavaScript 库。</span></li>
4          <li><span class="list"> jQuery 极大地简化了 JavaScript 编程。
           </span></li>
5          <li><span class="list"> jQuery 很容易学习。</span></li>
6          <li><span class="list"> jQuery 拥有供 AJAX 开发的丰富函数（方法）库。
           </span></li>
7       </ul>
8 </div>
```

结果如图 4.1 所示。

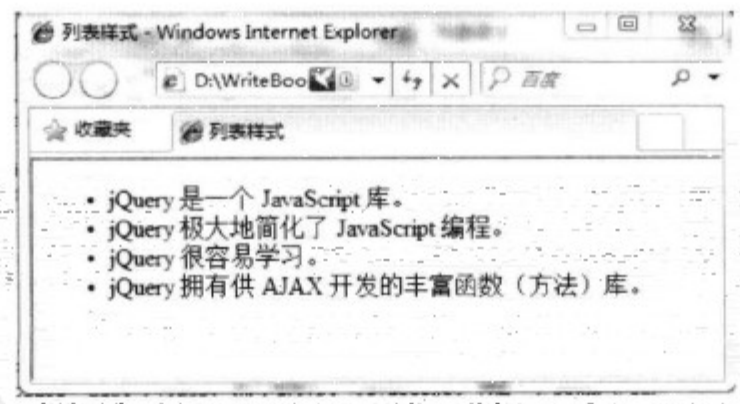


图 4.1　列表中宽度不同的列表项

这样的列表在页面中占用大量的空间，而且列表项长度参差不齐，影响页面效果。在页面布局中其他部分必须要适应这种列表宽度来安排，不能做到灵活布局。对于这样的问题，可以用 jQuery 来解决。jQuery 控制列表的宽度往往有两种解决方法，下面依次进行讲解。

4.1.2　截取文字内容实现控制列表宽度

截取文字内容实现控制列表宽度的原理为：可以对所有列表项截取相同长度。这样截取出来的列表项宽度就一致了。当然这种解决方式是基于字符等宽的基础上，如汉字字符。但是，如果是英文字符，这种解决方法就不尽如人意了。这种解决方法使用到了 jQuery 的 ready()、each()和 text()函数以及 JavaScript 的 substr()函数。

1．jQuery的ready()函数——文档加载完成事件

该函数在文档就绪后，添加特殊效果，或者加入动态事件。其语法如下：

```
语法一　$(document).ready(function)
语法二　$().ready(function)
语法三　$(function)
```

注：以上三种语法形式都表示调用 ready()函数。function 参数是一个必选参数，表示函数定义。在其中定义了当文档加载后要运行的功能。

2．jQuery的each()函数——遍历jQuery对象

该函数对 jQuery 对象进行遍历，为每个匹配元素执行函数。其语法如下：

```
语法一　each(function)
语法二　each(object,function)
```

注：语法一形式是以每一个匹配的元素作为上下文来执行一个函数。语法二形式为通用遍历方法，可用于遍历对象和数组。

3．jQuery的text()函数——所有匹配元素的内容

该函数可以获取并设置所有匹配包含的文本内容所组合起来的文本。其语法如下：

```
text([val|function])
```

其中，参数 val 表示文本内容，function 表示产生文本的函数。

4．功能实现

控制列表宽度的步骤如下。

（1）需要利用 jQuery 的 ready()函数来实现在页面整体加载完成后执行特效效果。

（2）在 function()内部使用 each()函数遍历列表项。

（3）在 each()函数中获取当前被遍历的<li>中嵌套的元素对象的文本内容。

（4）如果文本长度超出范围，则通过 substr()函数截取并加上删节号。

首先，把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码，利用 jQuery 的选择器找出每个列表项，获取文字长度，并按照要求来进行截取。完整代码如下：

```
1<script type="text/JavaScript">
2       $(document).ready(function() {       //使用 ready()函数，实现页面加载后
                                               即出现特效
3           $(".list").each(function() {     //使用 each()遍历每一个<span>
4               var inText = $(this).text(); //获取<span>中的文字内容
5               if (inText.length > 10) {    //判断文字内容长度
                 //使用 text()函数设置<span>中的文本，并用 substr()截取
6                   $(this).text(inText.substr(0,10)+"...");
7               }
8           });
9       });
10</script>
```

图 4.2 是加入 jQuery 代码调整列表项长度后的效果。将其与图 4.1 进行比较可以发现，列表项长度统一，不会占用大量的页面空间，更方便页面布局。而且，这种修改列表项长度的方式能灵活控制长度值，灵活适应页面布局。

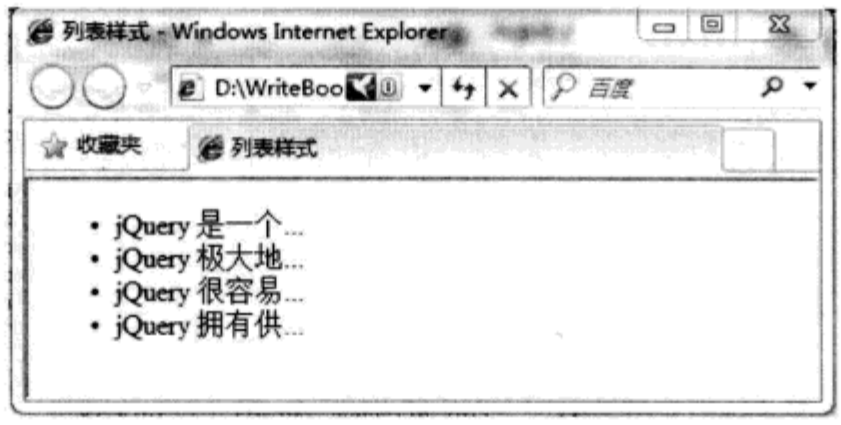


图 4.2　截取文字内容实现列表中宽度相同的列表项

就像前面提到的，在这里如果文字内容是英文字符，由于英文字符不是等宽字符，因此得出的列表项并不是相同长度。

4.1.3　修改层的宽度控制列表宽度

修改层的宽度控制列表宽度的原理为：可以把所有列表项所在的层设置为长度相同。这样截取出来的列表项宽度就一致了。这种解决方式与字符宽度无关，所以字符宽度不同不会影响到列表项的宽度。

先创建列表，列表的创建需使用 DIV，步骤如下。

（1）在页面头部添加下面的样式定义代码：

```
1<style type="text/CSS">
2       .list
```

```
3       {
4           font-size:14px;
5           width:150px;
6        }
7        .subone
8        {
9           float:left;
10           overflow:hidden;
11       }
12       .subtwo
13       {
14           float:left;
15           color:Blue;
16       }
17</style>
```

上面的样式定义代码中.list 是一个完整列表项所在的层的样式定义，.subone 和.subtwo 分别是列表项中第一部分和第二部分所在层的样式定义。

（2）下面的代码利用 DIV 标记创建列表：

```
1<div>
2       <div class="list">
3           <div class="subone"> jQuery 是一个 JavaScript 库。</div>
4           <div class="subtwo"> 简介</div>
5       </div>
6       <div class="list">
7           <div class="subone"> jQuery 极大地简化了 JavaScript 编程。</div>
8           <div class="subtwo"> 简介</div>
9       </div>
10       <div class="list">
11           <div class="subone"> jQuery 很容易学习。</div>
12           <div class="subtwo"> 简介</div>
13       </div>
14       <div class="list">
15           <div class="subone"> jQuery 拥有供 AJAX 开发的丰富函数（方法）库。
             </div>
16           <div class="subtwo"> 简介</div>
17       </div>
18 </div>
```

如图 4.3 所示是列表的实际效果。

对于这种形式的列表，需要使用 jQuery 来修改层的宽度。要实现控制列表项长度的效果，需要用到的函数有：jQuery 的 ready()、each()、width()和 css()函数。

1．jQuery的width()函数——元素宽度

该函数取得或者设置匹配元素当前计算的宽度值（px）。其语法形式如下：

```
语法一   width()
语法二   width(val)
```

图 4.3　使用 DIV 创建的参差不齐的列表

注：第一种语法形式表示取得第一个匹配元素当前计算的宽度值。第二种语法形式表示设置每个匹配元素当前计算的宽度值，val 为宽度值的参数。

2. jQuery的css()函数——元素的层叠样式

该函数取得或者设置匹配元素的层叠样式。其语法形式如下：

```
语法一    css(name)              //访问第一个匹配元素的样式属性
语法二    css(properties)        //把一个"名/值对"对象设置为所有匹配元素的样式属性
语法三    css(name,value|fn)     //在所有匹配元素中，设置一个样式的值，数字自动转换为像素值
```

注：第一种语法形式中 name 参数必须是有效的属性名。第二种语法形式中参数以 CSS 样式设定的形式出现，并用花括号包围。第三种语法形式参数分成两种表现形式，一种是(name，value)，name 为有效属性名，value 为属性值；另一种是(name,fn)，name 同样是有效属性名，fn 为返回属性值的函数。

3. 功能实现

控制列表项的宽度步骤如下。

（1）需要利用 jQuery 的 ready()函数实现在页面整体加载完成后执行特效效果。

（2）在 function()内部使用 each()函数遍历列表项。

（3）判断每一个列表项的整体宽度是否大于预定义宽度。

（4）如果大于，计算出列表项第一部分需要的长度并设置。

（5）适当调整每个列表项的高度。

首先，把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码，利用 jQuery 的选择器找出每个列表项，获取需要的长度，并按照要求来进行设置。完整代码如下：

```
<script type="text/JavaScript">
      $(function () {                                    //隐式调用 ready()函数
         var linelen = 120;                              //预定义列表项的宽度
         $.each($(".list"), function (i) {               //使用 each()遍历每一个 div
          //判断列表项实际长度是否超出范围
            if (($(".subone:eq(" + i + ")").width() + $(".subtwo:eq(" +
            i + ")").width()) > linelen) {
                                                         //计算出列表项第一部分需要的宽度
               var length = linelen - $(".subtwo:eq(" + i + ")").width();

               $(".subone:eq(" + i + ")").CSS("width", length + "px");
                                                         //设置列表项第一部分的宽度
                 $(".subone:eq("+i+")").CSS("height",65+"px");
                                                         //适当调整列表项的高度
            }
         });
      });
</script>
```

这里假定列表项第二部分，也就是“简介”的宽度是相同的。由于层的宽度改变后原有内容要变成多行，因此在代码的最后一部分可以加上修改标题所在 DIV 的高度，使标题信息可以完全显示出来，而不会被其他内容覆盖掉。图 4.4 为应用了 jQuery 调整列表项宽度后的效果。

图 4.4　调整层的宽度实现列表中的列表项宽度相同

以上两种控制列表项宽度的方法实现原理不同，个人认为第二种适用性更强，读者可根据现实需求灵活掌握。

4.2　控制列表项符号图片

本节将讨论列表项符号的灵活运用问题，主要是将原来死板的列表项符号用各种新颖的图片来代替。以图片代替原来的列表项符号，使列表看起来更美观生动。

4.2.1　样式死板的列表项符号

在创建列表的时候，<UL>中的 type 属性可以指定列表项符号样式。但是，单纯依靠该属性指定列表项样式效果过于单调死板。例如，下面这个静态页面。

```
1<ul id="ulstyle">
2       <li> jQuery 是一个 JavaScript 库。</li>
3       <li> jQuery 极大地简化了 JavaScript 编程。</li>
4       <li> jQuery 很容易学习。</li>
5       <li> jQuery 拥有供 AJAX 开发的丰富函数（方法）库。</li>
6</ul>
```

结果如图 4.5 所示。

在 CSS 中也有特定属性来设定列表项符号的图片（list-style-image）。但是，这个属性不被大多数浏览器所支持。所以，需要用另一种方法来设定符号图片。

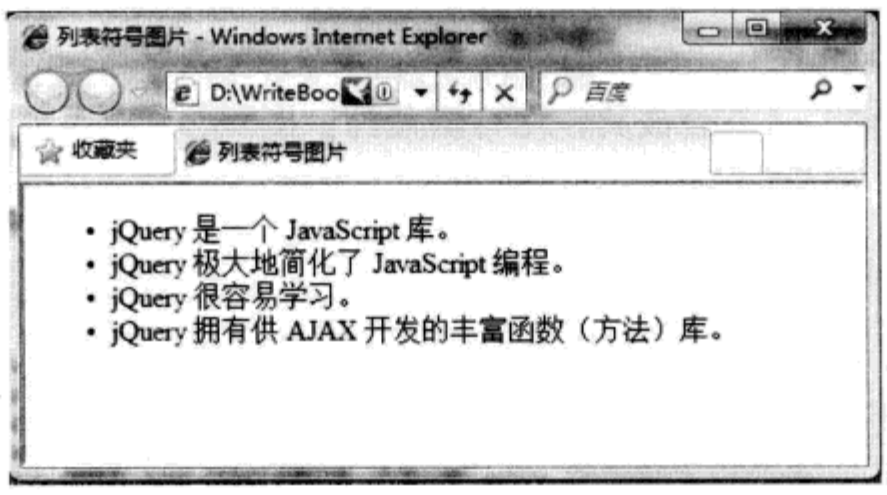


图 4.5　样式死板的列表项符号

4.2.2　利用 jQuery 与 CSS 控制列表项符号图片

利用 jQuery 与 CSS 控制列表项符号图片的原理是：利用 jQuery 为每个列表项动态加载 CSS 样式背景图片，并隐藏原有的列表项标号。其中，用到了 jQuery 的 addClass()函数。

1．jQuery的addClass()函数——添加样式类选择

该函数为每个匹配的元素添加指定的类名。其语法形式如下：

```
addClass(class|fn)
```

注：参数中 class 表示一个或多个要添加的类名，类名间可用空格分隔。fn 的使用形式为 function(index, class)，此函数必须返回一个或多个空格分隔的 class 名。fn 接收两个参数，index 参数为对象在这个集合中的索引值；class 参数为这个对象原先的 class 属性值。

2. 功能实现

控制列表项的图片符号步骤如下。

（1）添加含有指定背景图片的 CSS 样式类。

（2）利用 jQuery 的 ready()函数来实现在页面整体加载完成后执行特效效果。

（3）在 function()内部使用 each()函数遍历列表项。

（4）为每一个列表项添加 CSS 样式类。

首先，加入 CSS 样式类定义：

```
1<style type="text/CSS">
2    #ulstyle {list-style:none;}              //设定列表样式，取消原有的列表项符号
3    .list {background:url(img/listico.jpg) no-repeat;padding-left:20px}
                                              //设定背景图片样式
4</style>
```

然后，把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

最后，添加 JavaScript 代码，利用 jQuery 的选择器找出每个列表项，并将 CSS 类添加到列表项上。完整代码如下：

```
1<script language="JavaScript">
2    $(document).ready(function(){
3        $("#ulstyle li").each(function(){          //使用 each()遍历每一个 li
4            $(this).addClass("list");              //为 li 添加样式类
5        });
6    });
7</script>
```

图 4.6 是最后显示的效果。但是，这里要注意的是一定不能让背景图片重复显示。因为如果背景图片重复摆放，会造成文字部分模糊不清，失去了期望的效果。

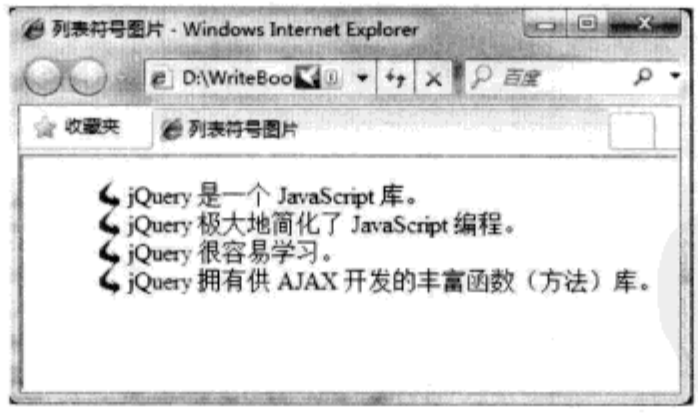


图 4.6　控制列表项图片符号

4.3　列表项的滚动

现在，众多网站中采用一种可以垂直滚动的新闻列表，每一个列表项都作为一个新闻

标题。让静态的列表滚动起来可以加强页面的动感效果，列表项的滚动一般适用于实时更新的网站内容标题。本节就来探讨如何让静态列表滚动起来，出现滚动新闻的效果。静态页面如图 4.5 所示。

使用 jQuery 产生滚动列表的原理是：利用隐藏与添加这两个动作。即首先需要取得列表的滚动区间，然后获取滚动内容的第一列表项并隐藏第一列表项，将隐藏的第一列表项添加到整个列表结尾。

其中，使用到了 jQuery 的 hover()、find()、height()、animate()、appendTo()、trigger()函数，以及：first 属性、mouseleave 事件和 JavaScript 的 clearInterval()、setInterval()函数，如表 4.1 所示。

（1）jQuery 的 animate()函数——自定义动画

该函数负责创建自定义动画。其语法形式如下：

```
语法形式一　animate(params,option)
语法形式二　animate(params,[duration],[easing],[fn])
```

注：语法形式一中，第一个参数含义同语法形式一，第二个参数表示一组包含动画选项的值的集合。语法形式二后面三个参数可省，依次表达的意义为：一组包含作为动画属性和终值的样式属性及其值的集合；三种预定速度之一的字符串（"slow"、"normal"or "fast"）或表示动画时长的毫秒数值（如 1000）；要使用的擦除效果的名称（需要插件支持），jQuery 默认提供"linear"和"swing"；在动画完成时执行的函数。

表 4.1　列表项的滚动所用到的函数

名　称	功　能	语　法	说　明
jQuery 函数			
hover()	自定义方法，模仿鼠标的悬停与移出事件	hover(over,out)	参数中 over 表示鼠标的悬停事件处理函数，out 表示鼠标的移出事件处理函数
find()	搜索所有与指定表达式匹配的元素	find(expr)	参数 expr 表示用于查找的表达式，这个函数是用于找出正在处理的元素的后代元素的适宜方法
height()	取得或设置正在处理的元素的高度	语法形式一：height() 语法形式二：height(val)	无参函数表示取得元素高度，有参函数表示设置元素高度，默认单位为像素
appendTo()	把所有匹配的元素加入到另一个指定的元素集合中	appendTo(content)	参数 content 表示指定的元素集合
trigger()	在每一个匹配的元素上触发指定的事件	trigger(type,[data])	该函数即可以触发固有事件也可以触发自定义事件。参数 type 表示事件对象或者触发的事件类型，data 表示传递给事件处理函数的附加参数，以数组形式存在
mouseleave	代替原有的 JavaScript 的 onmouseout 事件，但是修正了其中的一些错误	mouseleave(fn)	此事件不会像 onmouseout 事件一样，在不同的子元素间移动并不会触发它。参数 fn 表示事件处理函数
JavaScript 函数			
clearInterval()	取消周期方法调用	clearInterval(val)	参数 val 是函数 setInterval 返回的一个 timer ID

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

（2）jQuery 的属性：first——获取第一个子元素

该属性获取当前匹配元素的第一个子元素。其语法形式如下：

```
Selector:first
```

（3）JavaScript 函数 setInterval()——设定一定时间间隔调用函数

该函数将不停地按照指定的周期调用函数或者表达式，直到窗口关闭或者调用了clearInterval()函数。其语法形式如下：

```
setInterval(fn,millsecond)
```

注：参数 fn 表示被调用的函数或者表达式，参数 millsecond 表示毫秒值。

1．功能实现

实现列表项滚动的步骤如下。

（1）取得整个列表的滚动区间。

（2）使用 jQuery 的 hover()函数分别响应鼠标的悬停与离开事件。

（3）在鼠标的悬停时间中获取滚动内容的第一列表项并隐藏第一列表项，将隐藏的第一列表项添加到整个列表结尾。

（4）设定滚动间隔，以及滚动过程中动画持续时间。

首先，加入 CSS 样式定义：

```
<style type="text/CSS">
   ul.scrollline{height:90px;}
</style>
```

然后，把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

最后，添加完整代码如下：

```
<script language="JavaScript">
   $(function(){
         var area=$('ul.scrollline');    //取得滚动区域
         var timespan=1000;              //定义滚动时间间隔
         var timeID;                     //需要清除的动画
         area.hover(function(){          //自定义鼠标悬停与移出事件处理
             clearInterval(timeID);
                                 //当鼠标在滚动区域中时，停止滚动，移出事件处理
             },function(){   //鼠标悬停事件处理
                 timeID=setInterval(function(){
                                 //设置滚动时间间隔及滚动动作
                 var moveline=area.find('li:first');
                                 //最先需要获取列表当前的第一行，这个位置很重要
                 var lineheight=moveline.height();  //取得每次滚动高度
                                 //通过取负 margin 值，隐藏第一行
                 moveline.animate({marginTop:-lineheight+'px'},500,
```

```
                    function(){
                //隐藏后，将该行的 margin 值置零，并添加到列表尾部，实现循环滚动
13                  moveline.CSS('marginTop',0).appendTo(area);
14              })
15          },timespan)          //滚动间隔时间取决于 timespan
16          }).trigger('mouseleave');
                                 //函数载入时，模拟执行 mouseleave，即自动滚动
17  });
18</script>
```

因为这种特效属于动态形式，所以这里不给出效果图片，请读者自行测试。

4.4　图 片 列 表

图片是网页中传达信息的基本元素，也是设计网页时最常用到的信息表现形式。但是，在页面布局中图片的摆放与排列却是让人很头疼的问题。本节主要介绍如何使用 jQuery 控制图片以列表形式规则摆放。

4.4.1　大小不一的图片不规则排列

如果单纯依靠<li>和<img>标记来摆放图片，实现图片大小统一并规则摆放是件很麻烦的事情。例如，下面这个静态页面：

```
1 <ul>
2  <li><img src="img/img1.jpg"/></li>
3  <li><img src="img/img2.gif" /></li>
4  <li><img src="img/img3.jpg"/></li>
5  <li><img src="img/img4.gif" /></li>
6  <li><img src="img/img5.jpg" /></li>
7  <li><img src="img/img6.jpg"/></li>
8  <li><img src="img/img7.gif" /></li>
9  <li><img src="img/img8.gif"/></li>
10</ul>
```

4.4.2　利用 jQuery 控制图片列表

利用 jQuery 控制图片列表的原理是：为每张图片设定一个相同大小的显示区域，并把这些显示区域通过列表的形式规则摆放。其中使用到了 jQuery 的 ready()、each()函数和 JavaScript 对象属性 style。

1. JavaScript对象属性：style——CSS属性

该属性获取或者设置匹配的标签的 CSS 属性。其语法形式如下：

```
object.style
```

注：可以通过 style 引用 CSS 各种子属性。

2. 功能实现

实现图片列表的步骤如下。

（1）设定图片在列表中排列的格式。

（2）通过 jQuery 选择器及函数获取每一个图片元素对象。

（3）为每一个图片元素对象设定一定尺寸。

首先，加入 CSS 样式定义：

```
1<style type="text/CSS">
2  .imglist{float:left;width:400px;list-style:none} //每行排列四个图片列表项
3  .imglist li{float:left;width:90px;margin:3px;text-align:center}
                                          //设定每个列表项的宽度、边距，内容居中
4  .imglist li .area{height:60px;width:80px;display:block;}
                                          //设定图片所在区域的大小并显示
5</style>
```

然后，把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.js" type="text/JavaScript"></script>
```

并对原来的列表的代码部分进行修改：

```
1 <ul class="imglist">
2  <li><span class="area"><img src="img/img1.jpg"/></span></li>
3  <li><span class="area"><img src="img/img2.gif" /></span></li>
4  <li><span class="area"><img src="img/img3.jpg"/></span></li>
5  <li><span class="area"><img src="img/img4.gif" /></span></li>
6  <li><span class="area"><img src="img/img5.jpg" /></span></li>
7  <li><span class="area"><img src="img/img6.jpg"/></span></li>
8  <li><span class="area"><img src="img/img7.gif" /></span></li>
9  <li><span class="area"><img src="img/img8.gif"/></span></li>
10</ul>
```

最后，添加完整代码如下：

```
1 <script language="JavaScript">
2 $(document).ready(function(){
3  $(".imglist").find("img").each(
4  function(){
5   var imgwidth=80;                          //图片显示标准宽度
6   var imgheight=60;                         //图片显示标准高度
7   var percent=0;//缩放比例
8   var image=new Image();                    //创建临时图片对象
9   image.src=this.src;                       //保存当前遍历到的图片对象

10    if(image.width>0 && image.height>0){ //获取加载完成的图片实际大小
    //计算图片需要缩放的比例
11  percent = (imgwidth/image.width < imgheight/image.height)?imgwidth/
    image.width:imgheight/image.height;
12   if(percent <= 1){                        //如果图片太大，则按照一定比例缩小图片
13     this.style.width = image.width*percent;
```

```
14      this.style.height =image.height*percent;
15    }
16     else {                                          //否则，保持图片原大小
17           this.style.width = image.width;
18           this.style.height =image.height;
19        }
20    }
21  }
22  );
23 });
24 </script>
```

图 4.7 是最后显示的效果。如前面的操作步骤所述，把列表项的排列形式设成了左对齐，并设定了列表的整体宽度为 400px。取得每个列表项中的图片并根据图片的大小缩小到合适的范围。其中，标记就是图片显示的容器。

图 4.7　控制图片列表

4.5　列表的显示与收缩

本节主要介绍列表的显示与收缩。在很多网站中都会出现类似的网页特效，即页面加载完成后，某一部分只显示大的标题。当把鼠标移动到标题上时，会在标题下动态出现列表信息项。例如，网站信息、产品分类说明等都可以使用这种效果。

4.5.1　占用页面空间的静态列表

当需要对页面中的某些信息进行分项描述时，列表是最有效的手段。但是，在静态页面中过分使用列表会占用大量空间，而且不易布局，影响页面美观性。静态页面代码及 CSS 样式设定如下。

CSS 样式设定：

```
1 <style type="text/CSS">
2  ul{list-style:none;}
3  .menu{position:relative;width:300px;padding:0px;}
4  .menu ul{display:block;background:#fefefe;border:1px solid #ddd;width:
```

```
    300px;padding:5px;margin:0px;}
5  .menu ul li{padding:5px 0;border-bottom:1px dotted #ddd;}
6 </style>
```

HTML 代码：

```
1  jQuery特点<br/>
2            <ul class="content">
3               <li>jQuery 是一个 JavaScript 库。</li>
4                <li>jQuery 极大地简化了 JavaScript 编程。</li>
5                <li>jQuery 很容易学习。</li>
6                <li>jQuery 拥有供 AJAX 开发的丰富函数（方法）库。</li>
7            </ul>
```

这种通过 HTML 标记产生的静态页面如图 4.8 所示。

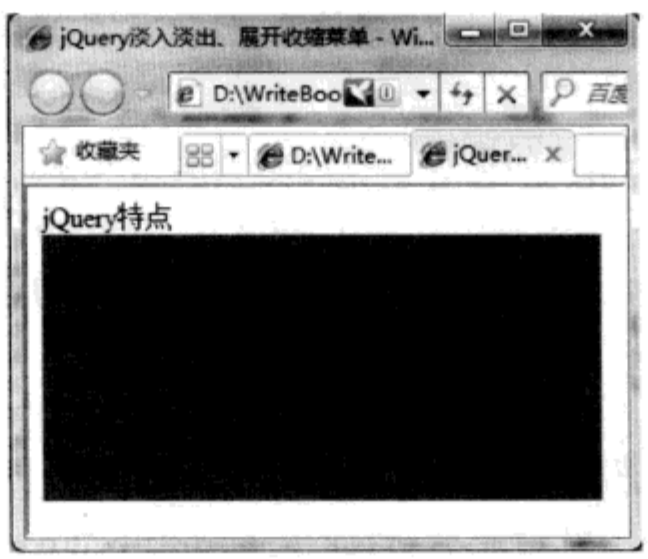


图 4.8　静态显示列表

4.5.2　利用 jQuery 动态控制列表内容展开与收缩

利用 jQuery 动态控制列表内容展开与收缩的原理是：通过 jQuery 的滑动效果函数将原本隐藏的列表内容滑动显示或者将原本显示的列表内容滑动隐藏。其中使用到的 jQuery 的函数有：ready()、hover()、toggle()、slideDown()、slideUp()函数。

1．jQuery函数介绍

（1）jQuery 的 toggle()函数——事件切换

该函数每次单击后依次调用函数。其语法形式如下：

```
toggle(fn1,fn2,[fn3,fun4,…])
```

注：如果单击了一个匹配的元素，则触发指定的第一个函数。当再次单击同一元素时，则触发指定的第二个函数。如果有更多函数，则再次触发，直到最后一个。随后的每次单击都重复对这几个函数的轮番调用。可以使用 unbind("click")来删除。fn1，第一次单击要执行的函数；fn2，第二次单击要执行的函数；fn3、fn4 等都是可选参数，表示更多次单击要调用的函数。

（2）jQuery 的 slideDown()函数——向下滑动

该函数通过高度变化（向下增大）来动态地显示所有匹配的元素，在显示完成后可选地触发一个回调函数。其语法形式如下：

```
slideDown(speed,[callback])
```

注：这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式显示出来。speed，三种预定速度之一的字符串（"slow"，"normal"or "fast"）或表示动画时长的毫秒数值（如 1000）；function 是在动画完成时执行的函数。

（3）jQuery 的 slideUp()函数——向上滑动

该函数通过高度变化（向上减小）来动态地隐藏所有匹配的元素，在隐藏完成后可选地触发一个回调函数。其语法形式如下：

```
slideUp(speed,[callback])
```

注：这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式隐藏起来。speed，三种预定速度之一的字符串（"slow"，"normal"或"fast"）或表示动画时长的毫秒数值（如 1000）；function 是在动画完成时执行的函数。

2．功能实现

实现列表内容展开与收缩的步骤如下。

（1）选定需要动态触发隐藏与显示列表内容的事件，hover()事件或者 toggle()事件。

（2）在事件中编写隐藏与显示的具体动作，并设置动作持续时间。

首先，把 jQuery 库引入进来：

```
<script language="JavaScript" src="jslib/jquery-1.6.js"></script>
```

然后，添加完整代码：

```
1 <script language="JavaScript" type="text/JavaScript">
2  $(document).ready(function(){
3     $(".menu").hover(                              //鼠标悬停与离开事件
4         function(){
5             $(".content").slideDown(800);          // 展开
6         },function(){
7             $(".content").slideUp(1000)            // 收缩
8         });
9  })
10 </script>
```

上面这段代码设定了鼠标覆盖事件，也可以使用鼠标单击事件。

```
1 <script language="JavaScript" type="text/JavaScript">
2  $(document).ready(function(){
3     $(".menu").toggle(                             //鼠标单击事件
4         function(){
5             $(".content").slideDown(800);          // 展开
6         },function(){
```

```
7            $(".content").slideUp(1000)          // 收缩
8        });
9  })
10 </script>
```

上述两段代码只是在事件的使用上有所不同。在这里还要注意一点，当加入 jQuery 代码后，在 CSS 样式设定中需要把原来的列表显示状态改成隐藏状态，即：

```
.menu ul{display:block;background:#fefefe;
border:1px solid #ddd;width: 300px;padding:
5px;margin:0px;}
```

当页面加载完毕时，效果如图 4.9 所示。

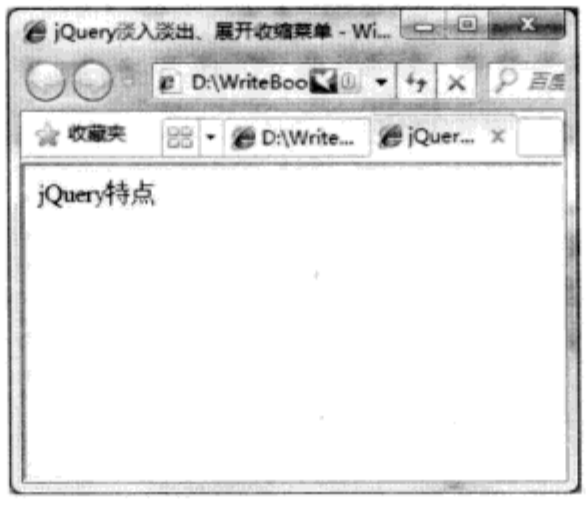


图 4.9　页面加载完毕隐藏列表

当单击或者将鼠标停滞在“jQuery 特点”这几个字上时，隐藏的列表会向下滑动并展开显示，如图 4.8 所示，整个列表内容都会显示出来。再次单击或者将鼠标移开后，页面又恢复到如图 4.9 所示的效果。

4.6　列表项动态排序

本节主要介绍如何通过 jQuery 在客户端实现对无序列表项进行排序。这种做法虽然应用不是很广泛，但是从减轻服务器负载的角度来看还是有其优点的。

4.6.1　构建一个无序列表

首先创建一个无序列表，代码如下。

```
1 <ul class="orderobj">
2    <li>Tom</li>
3    <li>Snoopy</li>
4    <li>Jerry</li>
5    <li>Micky</li>
6 </ul>
```

效果如图 4.10 所示。

图 4.10　无序的列表项

4.6.2　利用 jQuery 对无序列表排序

利用 jQuery 对无序列表排序的原理是：获取到无序列表中的所有列表项，并转成数组形式，使用 JavaScript 函数对其进行排序后再次输出。其中使用到的 jQuery 函数有 ready()、get()、text()、each()、append()和 JavaScript 函数 sort()。

1．jQuery函数介绍

（1）jQuery 函数 get()——获取匹配元素集合

该函数取得所有匹配元素的一种向后兼容的方式（不同于 jQuery 对象，实际上是元素数组）。其语法形式如下：

```
object.get()
```

注：如果你想要直接操作 DOM 对象而不是 jQuery 对象，这个函数非常有用。

（2）jQuery 函数 text()——获取和设置元素内容

该函数获取和设置匹配元素的文本内容。其语法形式如下：

```
object.text([val|fn])
```

注：val 和 fn 参数可选。val 是设置元素的文本内容值；fn(index,text)函数返回一个字符串，接受两个参数，index 为元素在集合中的索引位置，text 为原先的 text 值。

（3）jQuery 函数 append()——向元素追加内容

该函数向每个匹配的元素内部追加内容。其语法形式如下：

```
object.append(content|fn)
```

注：这个操作与对指定的元素执行 appendChild 方法并将它们添加到文档中的情况类似。content 参数表示追加的内容；fn(index,html)返回一个 HTML 字符串，用于追加到每一个匹配元素内部，接受两个参数，index 参数为对象在这个集合中的索引值，html 参数为这个对象原先的 html 值。

2. JavaScript函数介绍

JavaScript 函数 sort()——元素排序，用于对数组元素进行排序。其语法形式如下：

```
arrayObject.sort([sortby])
```

注：sortby 可选，规定排列顺序，必须是函数。返回值为排序后的数组本身。如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序。说得更精确点，是按照字符编码的顺序进行排序。要实现这一点，首先应把数组的元素都转换成字符串（如有必要），以便进行比较。

如果想按照其他标准进行排序，就需要提供比较函数，该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，其返回值如下：若 a 小于 b，在排序后的数组中 a 应该出现在 b 之前，则返回一个小于 0 的值。若 a 等于 b，则返回 0。若 a 大于 b，则返回一个大于 0 的值。

3. 功能实现

实现无序列表项排序功能的步骤如下。

（1）获取所有的列表项，并将其装入数组。

（2）对数组对象进行排序。

（3）将排好序的数组重新填充到无序列表中。

首先，把 jQuery 库引入进来：

```
<script language="JavaScript" src="jslib/jquery-1.6.js"></script>
```

然后，添加完整代码：

```
<script language="JavaScript" type="text/JavaScript">
$(document).ready(function(){
  var items = $(".orderobj li").get();    //获取所有待排序 li 装入数组 items
  items.sort(function(a,b)                //调用 JavaScript 内置函数 sort
  {
      var elementone = $(a).text();
      var elementtwo = $(b).text();
      if(elementone < elementtwo) return -1;
      if(elementone > elementtwo) return 1;
      return 0;
  });

  var ul = $(".orderobj");
  $.each(items,function(i,li)
              //通过遍历每一个数组元素，填充无序列表
  {
      ul.append(li);
  });
});
</script>
```

以上代码通过数组排序并重新填充无序列表使得列表项有序。具体效果如图 4.11 所示。

图 4.11　无序列表项排序

4.7　小　　结

本章主要介绍了如何通过 jQuery 操作列表对象。其中包括列表宽度的控制、列表项符号的定制、列表项的滚动、图片列表、列表项的收缩与展开、无序列表列表项的排序等功能。本章的重点是列表内容的动态变化及不同列表项内容的实现，难点部分在于理解如何控制列表项的滚动。下一章将讲解如何利用 jQuery 实现网站导航。

4.8　习　　题

【习题一】利用本章所学内容实现单击页面按钮更改列表中列表项宽度递增 20 像素。

【习题二】利用本章所学内容实现展开列表项时更改列表文字颜色。

【习题三】利用本章所学内容实现对相同内容的列表项用其他颜色标示出来。

第5章 网站导航

网站导航是所有网站所必备的元素之一。它可以使网站的用户清楚自己所浏览的页面位置，并能快速找到自己感兴趣的页面。目前，网站导航的表现形式基本为菜单样式或TreeView样式。本章将针对这两种样式来讲解jQuery如何控制网站导航。

5.1 菜单设计

菜单形式的网站导航在页面中适合放在页眉部分。菜单设计主要应用到HTML中的 标记和
的配合使用，再加上jQuery实现动态效果。本节用几个示例来展示菜单形式的网站导航功能。

5.1.1 普通下拉菜单

首先介绍效果最简单的下拉菜单形式的网站导航。下拉菜单是最常见的菜单模式。该模式主要用于表达整个网站的站点地图。

这种下拉菜单的实现原理是：获取鼠标指针是否通过顶层菜单。如果动作发生，则将下层菜单定位并显示出来，其间利用下拉动画实现效果。如果鼠标指针从当前菜单位置离开，则利用动画将子菜单收起，效果如图5.1和图5.2所示。

图5.1 普通下拉菜单加载

图5.2 鼠标进入菜单区域触发子菜单下拉

其中，应用到的jQuery函数有ready()、mouseenter()、stop()、hide()、parent()、next()、css()、offset()、height()、slideDown()、slideUp()、mouseleave()，如表5.1所示。

表 5.1　普通下拉菜单所用的函数

名　称	功　能	语　法	说　明
jQuery函数			
mouseenter ()	该函数响应鼠标进入到元素时产生的事件	$(selector).mouseenter()	与 mouseover 事件不同，只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。如果鼠标指针穿过任何子元素，同样会触发 mouseover 事件
stop ()	该函数停止所有在指定元素上正在运行的动画	stop([clearQueue],[goto-End])	clearQueue 参数，布尔型参数，可选，清空动画队列，如果设置为真，则立即清空队列。动画立即停止。gotoEnd 参数，布尔型参数，可选，让当前正在执行的动画立即完成，并重新设定 show 和 hide 的原始样式。如果队列中有等待执行的动画（并且 clearQueue 不为真），则马上执行
hide ()	该函数隐藏所有匹配的元素	语法形式一：hide() 语法形式二：hide (speed,[fn])	第一种语法形式简单地隐藏匹配元素。第二种语法形式以动画隐藏匹配元素，并在显示完成后可选地触发回调函数。speed 参数规定了动画时长，fn 参数为回调函数，并且每个匹配元素都执行一次
parent ()	该函数取得一个包含着所有匹配元素的唯一父元素的元素集合	parent([expr])	expr 参数表示筛选表达式
next ()	该函数取得一个包含匹配的元素集合中每一个元素紧邻的后面同辈元素的元素集合	next([expr])	expr 参数表示筛选表达式。这个函数只返回后面那个紧邻的同辈元素，而不是后面所有的同辈元素
offset	该函数获取匹配元素在当前视口的相对偏移	mouseleave(fn)	无参数调用时，返回的对象包含两个整型属性：top 和 left。此方法只对可见元素有效。携带参数时重新设置元素位置，此元素位置是相对于 document 对象。如果元素的 position 属性是 static，会被改成 relative 实现重新定位。coordinates 参数是包含 top 和 left 坐标属性的对象

1．功能实现

动态下拉菜单的实现步骤如下。

（1）需要利用 jQuery 的 ready()函数来实现在页面整体加载完成后执行特效效果。

（2）在 function()内部实现主菜单的鼠标进入事件，所有的下拉效果都在这个事件中实现。

（3）停止播放所有特效动画并隐藏下级菜单。

（4）获取下级菜单对象，重新设定下级菜单位置。

（5）停掉下级菜单其他动画并使其下拉。

（6）添加下级菜单的鼠标移出事件，让下级菜单向上收起。

为了使页面效果美观，在 HTML 文件中加入 CSS 样式设定。需要设定一些元素的样式，如 ul、ul li、ul li a 等。具体 CSS 代码请参考光盘内容。

2. 代码的实现

在页面中加入无序列表作为菜单主体：

```
<div id="menudiv">
  <ul>
<li><a href="#" class="mainmenu">Menu One</a></li>
 <li class="submenu">
<a href="#">Link 1</a>
<a href="#">Link 2</a>
<a href="#">Link 3</a>
<a href="#">Link 4</a>
<a href="#">Link 5</a>
</li>
 </ul>
  <ul>
<li><a href="#" class="mainmenu">Menu Two</a></li>
 <li class="submenu">
<a href="#">Link 1</a>
<a href="#">Link 2</a>
<a href="#">Link 3</a>
<a href="#">Link 4</a>
<a href="#">Link 5</a>
</li>
 </ul>
 </div>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码。完整代码如下：

```
<script language="JavaScript">
$(function(){
  $('.mainmenu').mouseenter(function(){          //实现主菜单的鼠标进入事件
    $('.submenu').stop(false, true).hide();
                                                 //停止播放所有特效动画并隐藏下级菜单
    var submenu = $(this).parent().next();       //获取下级菜单对象
    submenu.css({                                //设定子菜单样式，重新定位
                position:'absolute',
                top: $(this).offset().top + $(this).height() + 'px',
                left: $(this).offset().left + 'px',
                zIndex:1000
            });
    submenu.stop().slideDown(300);               //停掉下级菜单其他动画并使其下拉
```

```
13      submenu.mouseleave(function(){
                                    //添加下级菜单的鼠标移出事件，让下级菜单向上收起
14                                    $(this).slideUp(300);
15                          });
16    });
17  });
18 </script>
```

最后效果如图 5.1 和图 5.2 所示。

5.1.2　下拉级联菜单

继续以前面实现的下拉菜单为例，在原有下拉菜单的基础上添加第二级子菜单，第二级子菜单横向显示。实现原理与上一节介绍的动态下拉菜单类似：获取到顶层菜单后，模拟顶层菜单的鼠标悬停与离开事件，并设置下层菜单的可见状态，效果如图 5.3 和图 5.4 所示。

图 5.3　多级下拉菜单加载

图 5.4　多级下拉菜单展开

其中，应用到的 jQuery 函数有 ready()、css()、hover()、find()、show()和 jQuery 属性:first。

1. 功能实现

多级下拉菜单的实现步骤如下。

（1）隐藏所有子菜单。

（2）设定主菜单项及各级子菜单项的模仿悬停事件。

（3）在模仿悬停事件中设定鼠标悬停则显示下级菜单，鼠标离开则收回菜单。

为了使页面效果美观，在 HTML 文件中加入 CSS 样式设定。需要设定一些元素的样式，如 ul、ul li、ul li a 等。具体 CSS 代码请参考光盘内容。

2. 代码实现

在页面中加入无序列表作为菜单主体：

```
<ul id="mainmenu"><!--最上层菜单-->
  <li><a href="#">1 HTML</a></li>
  <li><a href="#">2 CSS</a></li>
  <li><a href="#">3 JavaScript </a>
    <ul><!--一级子菜单-->
        <li><a href="#">3.1 jQuery</a>
          <ul><!--二级子菜单-->
              <li><a href="#">3.1.1 Download</a></li>
              <li><a href="#">3.1.2 Tutorial</a></li>
          </ul>
        </li>
        <li><a href="#">3.2 Mootools</a></li>
        <li><a href="#">3.3 Prototype</a></li>
    </ul>
  </li>
</ul>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码，完整代码如下：

```
<script type="text/JavaScript">
 $(document).ready(function(){
  $(" # mainmenu ul ").css({display: "none"});        //隐藏各级菜单
   $(" # mainmenu li").hover(function(){ //为第一级菜单加入模仿鼠标悬停事件
     $(this).find('ul:first').css({visibility: "visible"}).show(400);
     //鼠标悬停则显示下级菜单
     },function(){
     $(this).find('ul:first').css({visibility: "hidden"});
     //鼠标离开则收起下级菜单
     });
 });
</script>
```

上述代码中不容易理解的是第 3 行代码，这一行代码表示为所有菜单项设定鼠标悬停

事件，不只是最上层菜单，请读者注意这一点。具体实现效果如图 5.3 和图 5.4 所示。

5.1.3　横向伸缩菜单

前面所见到的菜单都是横向下拉菜单，本节认识一下纵向排列、横向伸缩的菜单。横向伸缩菜单动画感更强，而且对于页面布局有它自己的优点。

这种菜单的实现原理是：设定每个上层菜单项的鼠标悬停与离开事件，并在对应的鼠标悬停事件中加宽这一行的宽度，将本来我们无法看到的菜单内容显示出来。当鼠标离开时重新将这一行宽度设为初始值，给用户一种菜单收缩的感觉。

其中，使用到的 jQuery 函数有 ready()、mouseover()、mouseout()、stop()、animate()。

1．功能实现

纵向排列菜单横向伸缩效果实现步骤如下。

（1）设定菜单项的鼠标悬停事件，在事件中停止动画队列里的其他动画，并将鼠标所在行动画伸展。

（2）设定菜单项的鼠标离开事件，在事件中停止动画队列里的其他动画，并将鼠标所在行动画收缩。

为了使页面效果美观，在 HTML 文件中加入 CSS 样式设定。需要设定一些元素的样式，如 ul、ul li、ul li a 等。具体 CSS 代码请参考光盘内容。

2．代码的实现

在页面中加入无序列表作为菜单主体：

```
1 <div id="menudiv">
2 <ul>
3   <li><a href="#"><img src="img/home.png" alt="" width="50" height="50"
    border="0"/></a>
 Home</li>
4   <li><a href="#"><img src="img/portfolio.png" alt="" width="50"
    height="50" border="0"/></a>
 Portfolio</li>
5   <li><a href="#"><img src="img/about.png" alt="" width="50" height="50"
    border="0"/></a>
About</li>
6   <li><a   href="#"><img   src="img/contact.png"   alt=""   width="50"
    height="50" border="0"/></a>
 Contact</li>
7 </ul>
8 </div>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码。完整代码如下：

```
<script type="text/JavaScript">
$(document).ready(function(){
   $("li").mouseover(function(){        //设置鼠标悬停事件
      $(this).stop().animate({
      width: "152px", }, 500 );
   });
   $("li").mouseout(function(){         //设置鼠标离开事件
      $(this).stop().animate({
      width: "52px",}, 500 );
    });
});
</script>
```

最后效果如图 5.5 和图 5.6 所示。

图 5.5　纵向菜单加载

图 5.6　纵向排列菜单横向伸缩

5.2　第三方菜单插件

本节将为大家介绍 6 个菜单插件。当然 jQuery 的菜单并不只有目前所要介绍的 6 个，这里只是介绍几种比较有代表性的菜单插件。本节所介绍的这些插件都是开源插件，所以读者在使用时可以从我们所给网址直接下载并参照配套光盘上的用例。

5.2.1　jQuery 级联菜单插件

jQuery 级联菜单插件与前面介绍的多级菜单很类似。它的实现原理也是通过设定模仿悬停事件，并设定淡入淡出效果来实现级联菜单的效果。在这个菜单插件中主要利用了 jQuery 的 find()、parent()、children()函数来定位元素。hover()事件是整个插件的核心事件，

它实现了动态效果。fadeIn()、fadeout()函数实现了动画过程中淡入淡出的效果。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用该插件及实现插件效果。在页面中加入无序列表作为菜单主体，具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1   <link rel="stylesheet" type="text/css" href="CSS/jquerycssmenu.css" />
2   <script type="text/JavaScript" src="../../jslib/jquery-1.6.js">
    </script>
3   <script type="text/JavaScript" src="jQueryCode/jquerycssmenu.js">
    </script>
```

其中，第一行代码设定了菜单的样式，第二行代码加入了 jQuery 库文件，第三行代码调用插件文件。具体实现效果如图 5.7 和图 5.8 所示。本插件英文介绍网址为 http://www.dynamicdrive.com/style/csslibrary/item/jquery_multi_level_css_menu_horizontal_blue/。

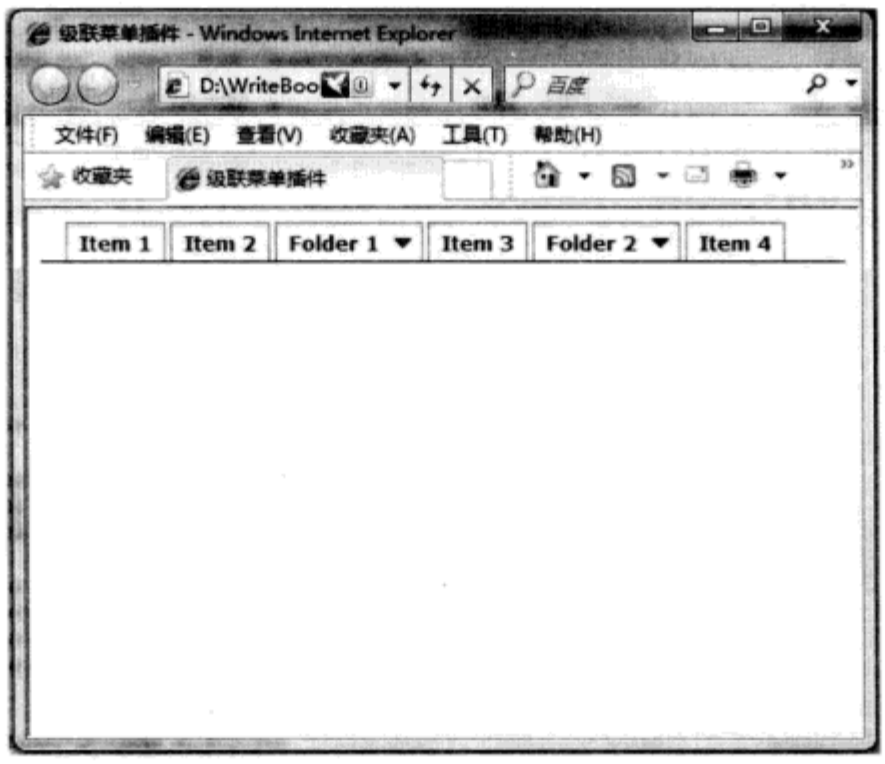


图 5.7　级联菜单插件应用一

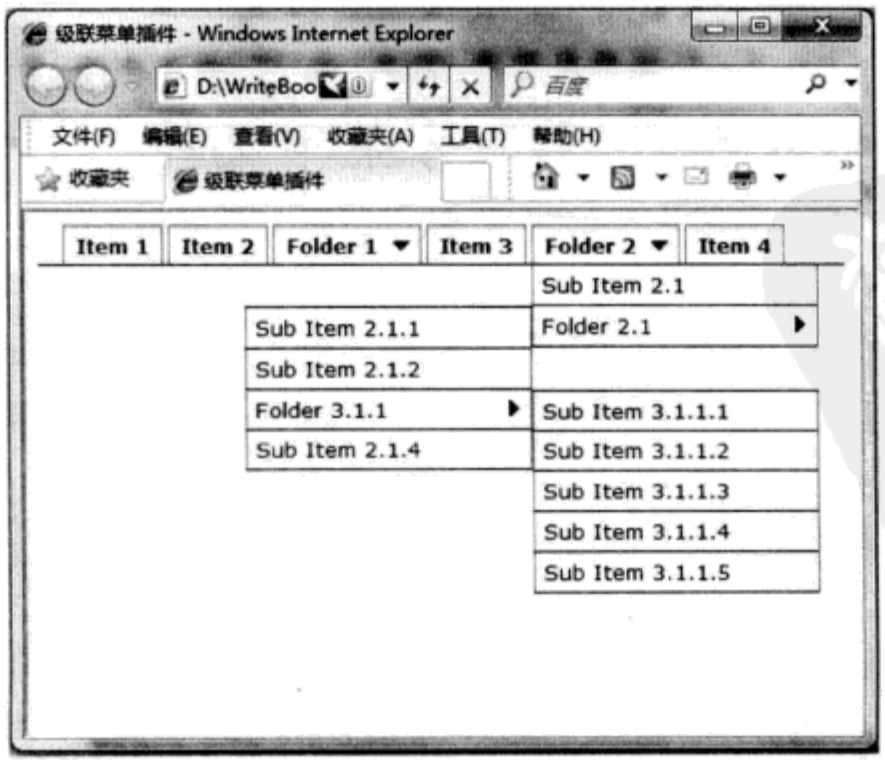


图 5.8　级联菜单插件应用二

5.2.2　SuperFish 菜单插件

本插件的作者是 Joel Birch，英文介绍网址为 http://users.tpg.com.au/j_birch/plugins/superfish/#examples。SuperFish 菜单插件支持水平/垂直方向。这种菜单的实现原理也是通过设定模仿悬停事件。hover()事件是整个插件的核心事件，它实现了动态效果。removeClass()、addClass()函数通过修改菜单项样式实现了菜单的显示与隐藏的效果。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用该插件及实现插件效果。在页面中加入无序列表作为菜单主体，具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1   <link rel="stylesheet" type="text/css" href="css/superfish.css" media=
    "screen">
2   <script type="text/JavaScript" src="../../jslib/jquery-1.6.js">
    </script>
3   <script type="text/JavaScript" src="jQueryCode/hoverIntent.js">
    </script>
4   <script type="text/JavaScript" src="jQueryCode/superfish.js">
    </script>
5   <script type="text/JavaScript">
6       jQuery(function(){
7           jQuery('ul.sf-menu').superfish();
8       });
9   </script>
```

在上述代码中，第 1 行加入了 CSS 样式定义。第 2 行引入 jQuery 库文件。第 3、4 行加入了 SuperFish 菜单插件的文件。第 6、7 行初始化插件，将要处理的菜单部分“ul.sf-menu”交给插件。实现效果如图 5.9 和图 5.10 所示。

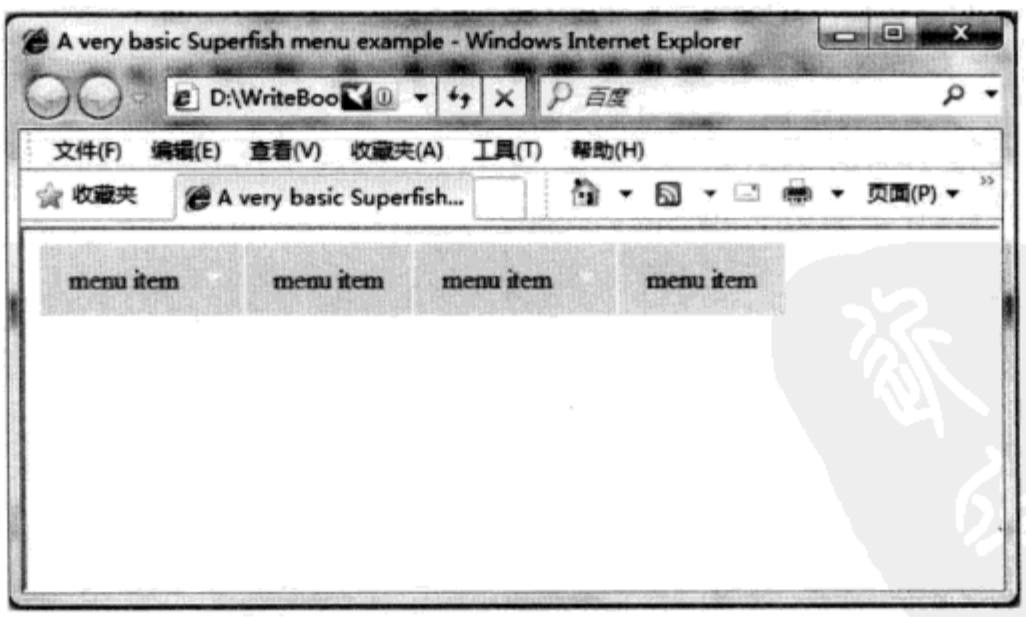


图 5.9　SuperFish 菜单插件应用一

这个插件可以有多种形态的菜单样式，如垂直排列形式菜单和导航条形式菜单。如果需要垂直排列形式菜单，只需要将前面的代码修改如下：

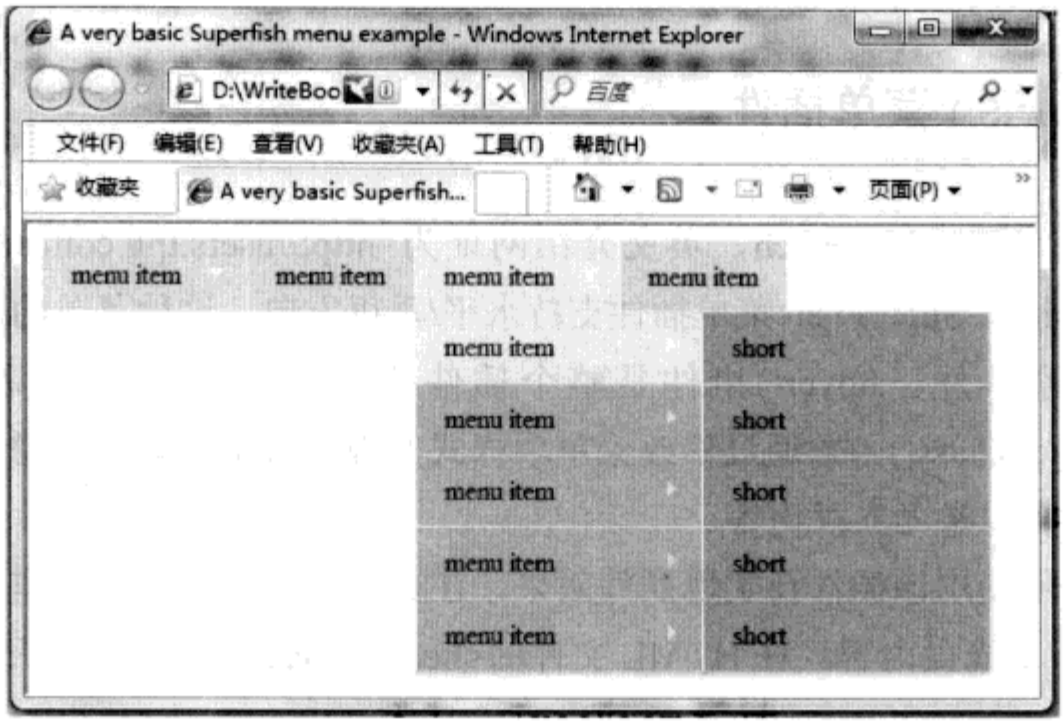


图 5.10　SuperFish 菜单插件应用二

```
1   <link rel="stylesheet" type="text/css" href="css/superfish.css"
    media="screen">
2    <link rel="stylesheet" type="text/css" href="CSS/superfish-vertical.
    css" media="screen">
3   <script type="text/JavaScript" src="../../jslib/jquery-1.6.js">
    </script>
4   <script type="text/JavaScript" src="jQueryCode/hoverIntent.js">
    </script>
5   <script type="text/JavaScript" src="jQueryCode/superfish.js">
    </script>
6   <script type="text/JavaScript">
7       $(document).ready(function(){
8       $("ul.sf-menu").superfish({
9           animation: {height:'show'},      //使用滑动效果，不使用淡出效果
10           delay:     1200                  //1.2 秒延迟鼠标离开
11          });
12      });
13  </script>
```

垂直样式如图 5.11 和图 5.12 所示。

图 5.11　SuperFish 菜单插件应用三

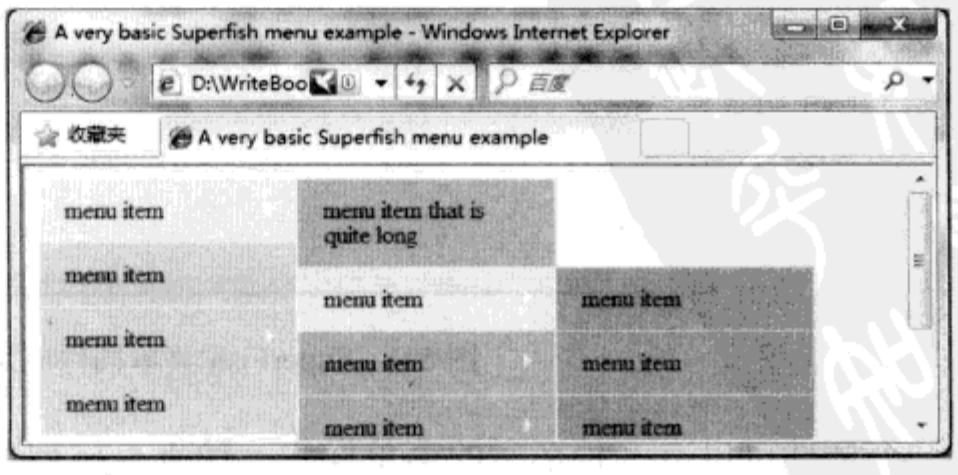


图 5.12　SuperFish 菜单插件应用四

如果需要导航条样式菜单，只需要将前面的代码修改如下：

```
<link rel="stylesheet" type="text/css" href="css/superfish.css" media="screen">
 <link rel="stylesheet" type="text/css" href="CSS/superfish-navbar.css" media="screen">
<script type="text/JavaScript" src="../../jslib/jquery-1.6.js"></script>
<script type="text/JavaScript" src="jQueryCode/hoverIntent.js"></script>
<script type="text/JavaScript" src="jQueryCode/superfish.js"></script>
<script type="text/JavaScript">
$(document).ready(function(){
    $("ul.sf-menu").superfish({
        pathClass:  'current'
        });
    });
</script>
```

导航栏样式如图 5.13 和图 5.14 所示。

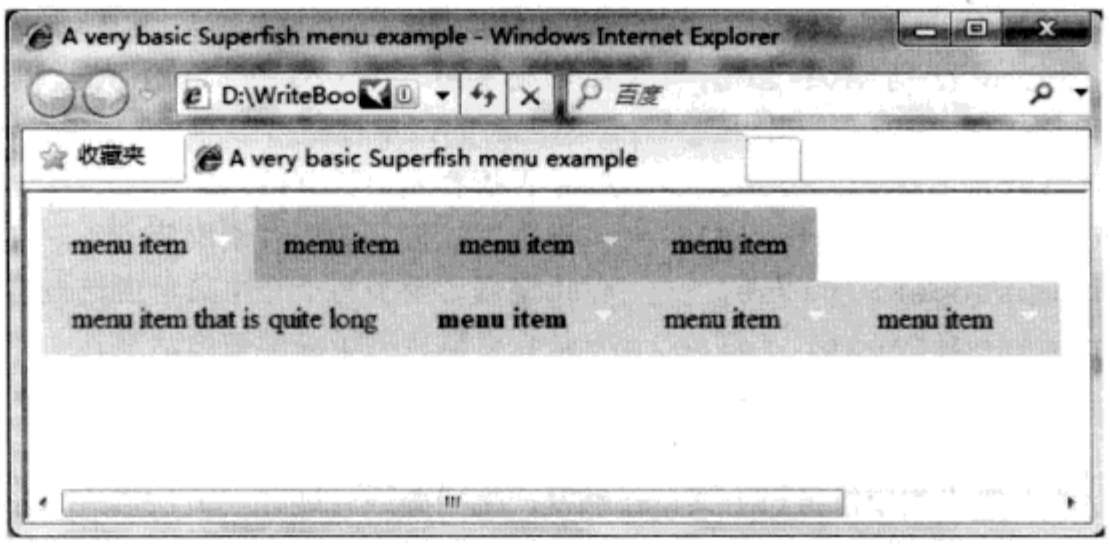


图 5.13　SuperFish 菜单插件应用五

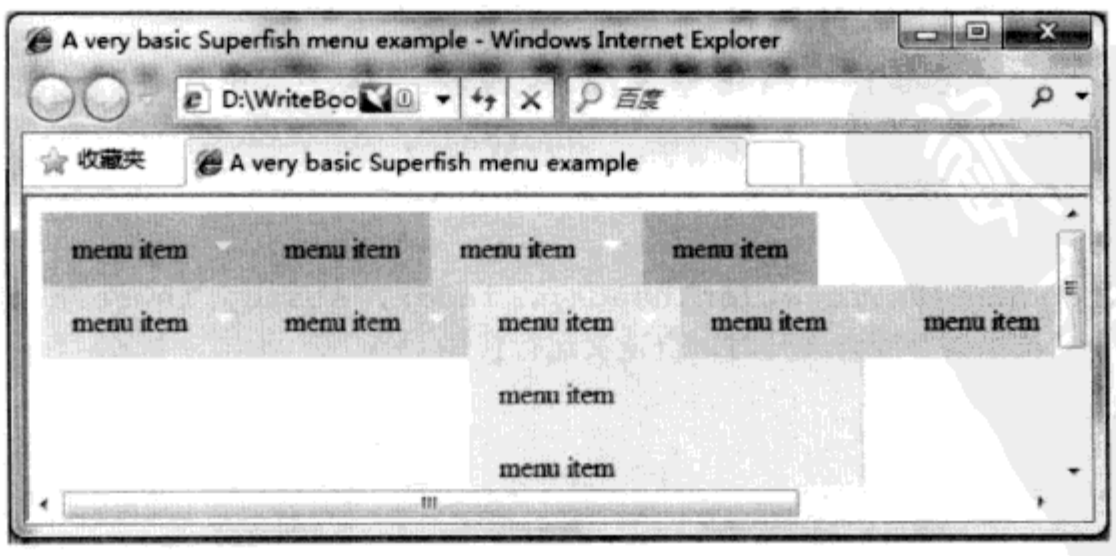


图 5.14　SuperFish 菜单插件应用六

5.2.3　折叠菜单插件

折叠菜单插件与前面介绍的两种菜单插件不同。前面介绍的菜单插件是通过模仿鼠标悬停事件产生菜单动态效果，而折叠菜单则是通过鼠标的单击触发事件产生菜单的折叠与展开效果。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用折叠菜单插件及实现插件效果。在页面中加入无序列表作为菜单主体，具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1   <script type="text/JavaScript" src="../../jslib/jquery-1.6.js">
    </script>
2   <script type="text/JavaScript" src="jQueryCode/ddaccordion.js">
    </script>
3   <script type="text/JavaScript">
4   ddaccordion.init({
5       headerclass: "expandable",          //主菜单样式类名 expandable
6       contentclass: "categoryitems",      //子菜单样式类名 categoryitems
7       revealtype: "click",
        //转换内容样式当鼠标单击或者鼠标悬停事件发生? 合法值: "click", "clickgo"
          或者 "mouseover"
8       mouseoverdelay: 200,    //如果 revealtype="mouseover", 设置鼠标的悬停
                                  延迟时间, 单位毫秒
9       collapseprev: true,     //是否折叠先前打开的内容? true/false
10      defaultexpanded: [0],   //设置默认打开的内容索引值
11      onemustopen: false,     //指定是否至少有一个主菜单应该被展开
12      animatedefault: false,  //是否有默认的子菜单被激活展开
13      persiststate: true,     //在浏览器会话过程中是否持续展开状态
14      toggleclass: ["", "openheader"],
        //两个样式类被指定当主菜单被触发产生折叠或者展开, 语法形式["class1",
          "class2"]
15      togglehtml: ["prefix", "", ""],
        //当主菜单折叠或者展开时, 在主菜单上附加的 HTML 代码, 语法形式"position",
          "html1", "html2"]
16      animatespeed: "fast",   //激活速度: 以毫秒为单位, 或者指定"fast",
                                  "normal", or "slow"
17      oninit:function(headers, expandedindices){      //自定义初始化代码
18      },
19      onopenclose:function(header, index, state, isuseractivated){
        //当菜单折叠或者展开时, 自定义运行代码
20      }
21  })
22  </script>
```

在上述代码中，第 4～20 行是菜单插件初始化代码。效果如图 5.15 和图 5.16 所示。本插件英文介绍网址为 http://www.dynamicdrive.com/dynamicindex17/ddaccordionmenu.htm。

图 5.15　折叠菜单插件应用一

图 5.16　折叠菜单插件应用二

5.2.4　滚动动态列表菜单插件

滚动动态列表菜单插件基于 JSON 数据。JSON 数据是与后台进行数据交互的一种轻量型数据交换格式。本书不对 JSON 做过多介绍，感兴趣的读者可以查看相关资料。这个菜单插件的原理是将 JSON 数据转换成网页中的列表，并利用第 4 章中介绍的滚动列表的原理实现列表项的滚动，并在菜单上部的数字指示区域加入了鼠标的单击事件。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用该插件及实现插件效果。在页面中加入无序列表作为菜单主体，具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1   <script src="../../jslib/jquery-1.6.js"></script>
2   <script src="jQueryCode/jaws.js"></script>
3   <script>
4       $(function() {
5           new Jaws({container: '#jawsHolder', //为菜单插件指定网页容器元素
6            title: 'latest news',              //出现在菜单顶部的标题内容
7            jsonFile: 'json.json',     //提供给菜单插件的 JSON 数据文件的路径
8            itemsPerFold: 4,                   //每屏所能显示的菜单项数量
9            doNumbers: true});                 //如果为真，则列表项被项目编号
10      });
11  </script>
```

在上述代码中，第 5～9 行是菜单插件初始化代码，效果如图 5.17 所示。本插件英文介绍网址为 http://www.mitya.co.uk/scripts/Jaws-news-ticker-and-mini-accordion-139?iframe=true&width=100%&height=100%。

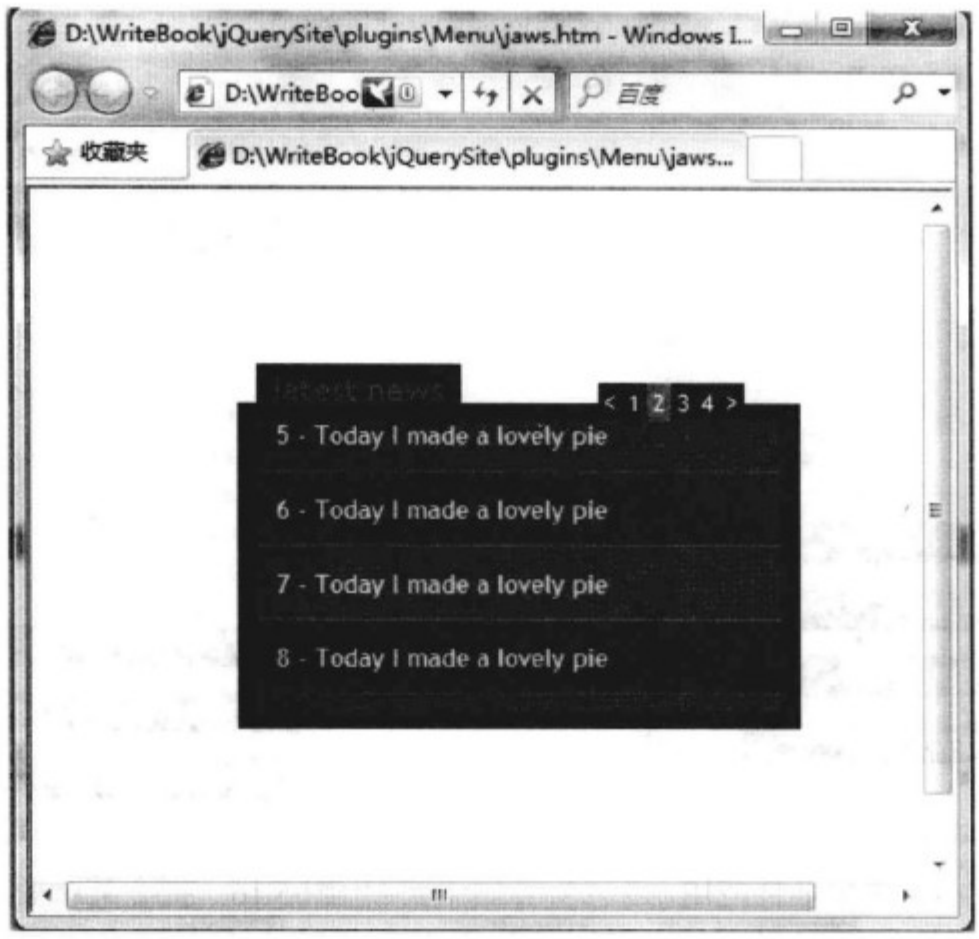


图 5.17　滚动动态列表菜单插件应用

5.2.5　滑动效果菜单

滑动效果菜单插件所要实现的功能是创建一个类似于 Flash 动画效果的动态菜单。实现原理是：利用在每一个菜单项上绑定鼠标的悬停与离开事件，并自定义在这两个事件中触发的动画效果，在动画效果中修改菜单的 CSS 样式。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用该插件及实现插件效果。具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1   <link rel="stylesheet" href="css/main.css" type="text/css" />
2   <script type="text/JavaScript" src="../../jslib/jquery-1.6.js" >
    </script>
3   <script type="text/JavaScript" src="jQueryCode/jquery-bp.js" >
    </script>
4   <script type="text/JavaScript" src="jQueryCode/navigation.js" >
    </script>
```

在上述代码中，第 3 行是一个修正了 jQuery 对于自定义动画关于背景位置样式使用的 Bug 的插件。第 4 行是我们使用的导航菜单插件。读者会发现在代码中并没有 JavaScript 代码，这是因为这个菜单插件已经把元素选择工作都放在了插件代码中，这样便于使用。但是，这样做也有不足之处，不够灵活，当我们想重新定制自己的菜单内容时，需要修改插件源文件。这个菜单插件的效果如图 5.18 和图 5.19 所示。本插件英文介绍网址 http://net.tutsplus.com/tutorials/JavaScript-ajax/create-a-cool-animated-navigation-with-css-and-jquery/。

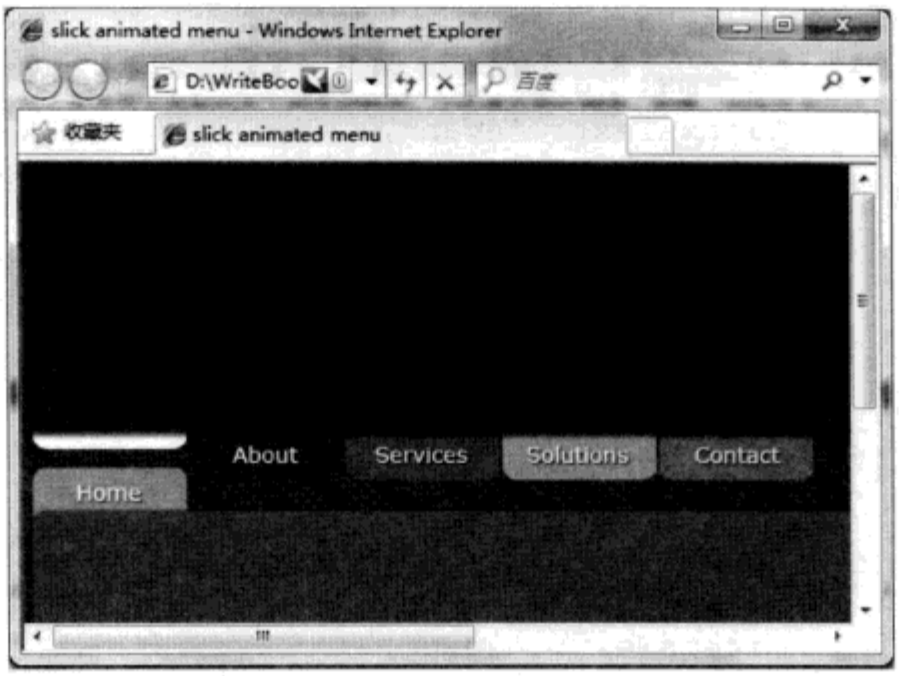


图 5.18　滑动效果菜单插件应用一

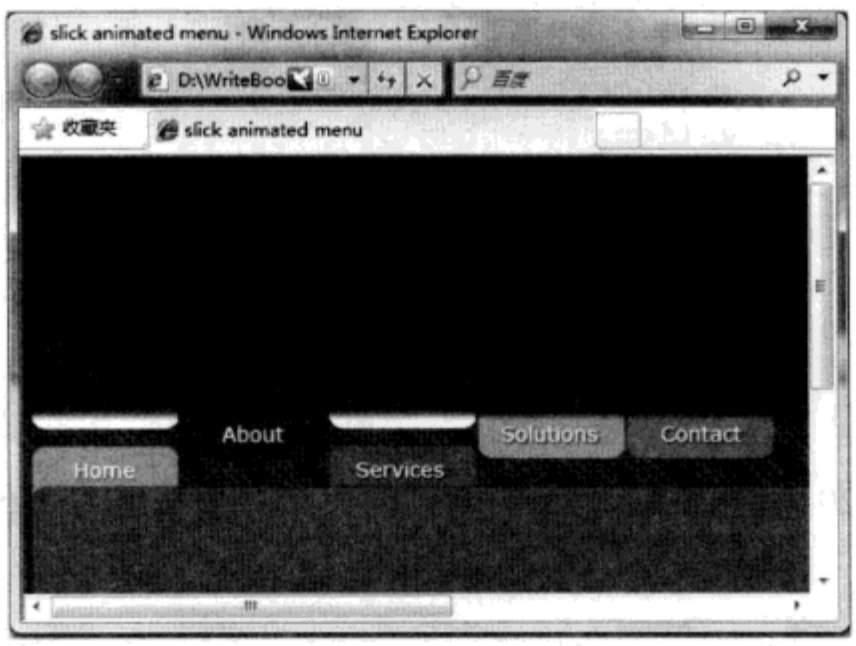


图 5.19　滑动效果菜单插件应用二

5.2.6　仿 Mac 的停靠菜单插件

仿 Mac 的停靠菜单插件实现的动态效果是模仿 Mac 系统效果。它的实现原理也是应用了菜单项的鼠标悬停与离开事件并修改菜单项的 CSS 样式设定。关于插件的具体代码可以参考光盘内容。

下面介绍如何使用该插件及实现插件效果，具体 HTML 代码参考光盘内容。在 HTML 文件的<head></head>中加入如下代码：

```
1 <script type="text/JavaScript" src="../../jslib/jquery-1.6.js">
  </script>
2 <script type="text/JavaScript" src="jQueryCode/interface.js"></script>
3 <link href="CSS/style.css" rel="stylesheet" type="text/css" />
4 <script type="text/JavaScript">
5   $(document).ready(
6       function()
7       {
8           $('#dock').Fisheye(                    //Fisheye 组件初始化
9               {
```

```
10                  maxWidth: 50,                       //菜单项最大宽度
11                  items: 'a',                         //菜单项元素标记
12                  itemsText: 'span',                  //菜单项文本元素标记
13                  container: '.dock-container',       //菜单容器
14                  itemWidth: 40,                      //菜单项宽度
15                  proximity: 90,                      //接近宽度
16                  halign : 'center'                   //水平对齐方式
17              }
18          )
19          $('#dock2').Fisheye(
20              {
21                  maxWidth: 60,
22                  items: 'a',
23                  itemsText: 'span',
24                  container: '.dock-container2',
25                  itemWidth: 40,
26                  proximity: 80,
27                  alignment : 'left',
28                  valign: 'bottom',
29                  halign : 'center'
30              }
31          )
32      }
33  );
34 </script>
```

具体效果如图 5.20 和图 5.21 所示。本插件英文介绍网址为 http://ndesign-studio.com/blog/ css-dock-menu。

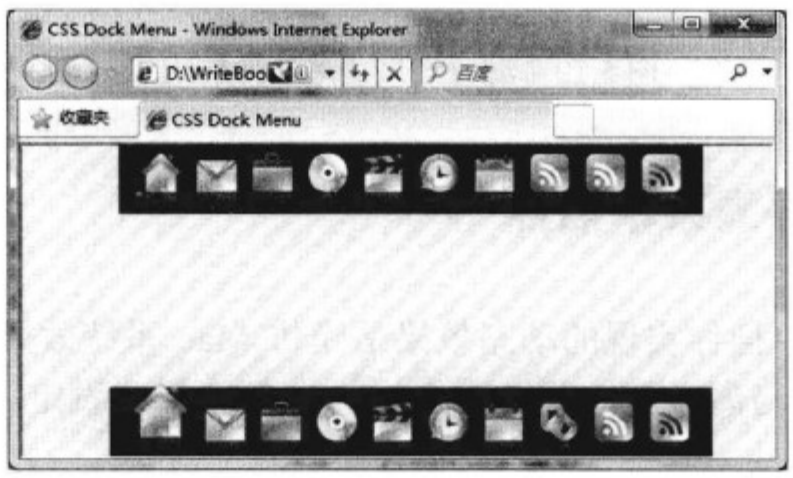


图 5.20　仿 Mac 效果菜单插件应用一

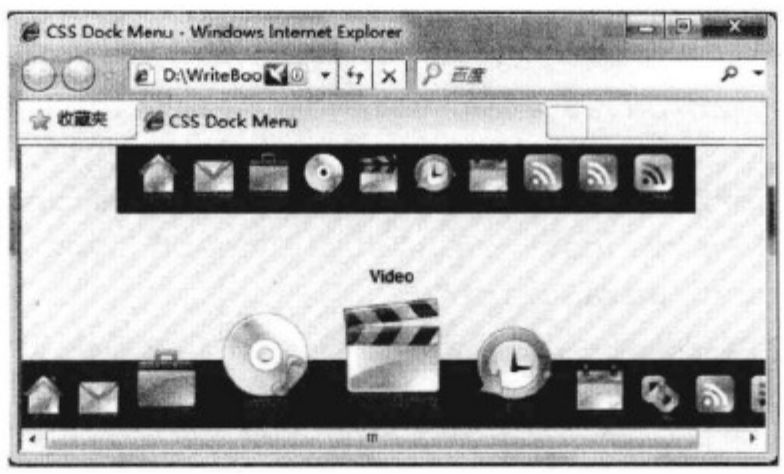


图 5.21　仿 Mac 效果菜单插件应用二

5.3　TreeView 设计

网站导航效果除了可以使用菜单效果实现外，还可以使用树形视图来实现。以下使用 TreeView 来代替树形视图。TreeView 一般被布置在页面的侧边栏位置，并可折叠收缩。TreeView 设计主要应用 HTML 中的 标记和
的配合使用，再加上 jQuery 实现动态效果。本节用几个示例来展示 TreeView 形式的网站导航功能。

5.3.1　普通 TreeView

在此介绍最简单的一种 TreeView 效果实现，它的原理是大部分 TreeView 实现的基础。它的工作原理是：通过设定上层节点中特定元素的鼠标单击触发事件，对下层子节点进行显示样式更改，效果如图 5.22 和图 5.23 所示。

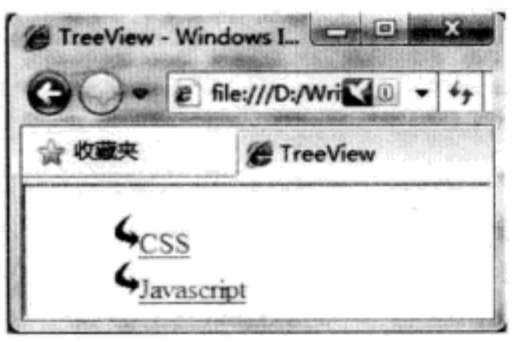


图 5.22　普通 TreeView 折叠效果

图 5.23　普通 TreeView 展开效果

其中，应用到了 jQuery 函数 ready()、children()、toggle()、parent()、next()、hide()、show()。

1．功能实现

普通 TreeView 实现步骤如下。

（1）获取需要响应鼠标单击事件的上层节点元素对象。

（2）对鼠标单击事件进行加工，修改子节点样式。

为了使页面效果美观，在 HTML 文件中加入 CSS 样式设定。需要设定一些元素的样式，如 ul、ul li、ul li a 等。

2．代码的实现

具体 CSS 代码请参考光盘内容。在页面中加入无序列表作为菜单主体：

```
1   <ul>
2       <li class="pe_u_thumb_list"><img src="img/listicon.jpg" /><a
        href="#">CSS</a></li>
3           <ul>
```

```
                <li><a href="#">级联样式单</a></li>
                <li><a href="#">统一设定标记元素样式</a></li>
            </ul>
        <li class="pe_u_thumb_list"><img src="img/listicon.jpg" /><a href=
        "#">JavaScript</a></li>
            <ul>
                <li><a href="#">网页前端脚本语言</a></li>
                 <li><a href="#">对事件响应实现网页动态效果</a></li>
             </ul>
    </ul>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码。完整代码如下：

```
<script language="JavaScript">
  $(document).ready(function(){
          $(".pe_u_thumb_list").children("img").toggle(function(){

          //对父节点中的图片元素设定触发事件
          $(this).parent().next("ul").hide();
                                          //鼠标单击图片奇数次时隐藏下层节点
             },function(){
          $(this).parent().next("ul").show();
                                          //鼠标单击图片偶数次时展开下层节点
          });
  });
</script>
```

最后效果如图 5.22 和图 5.23 所示。

5.3.2　加入淡入淡出效果的 TreeView

在此介绍类似于上面普通 TreeView 中加入淡入淡出效果的实现。它的工作原理是：通过设定父节点中特定元素的鼠标单击触发事件，对子节点进行显示样式更改，效果如图 5.24 和图 5.25 所示。

其中，应用到了 jQuery 函数 ready()、click()、nextAll()、toggle()。

图 5.24　淡入淡出效果 TreeView 加载

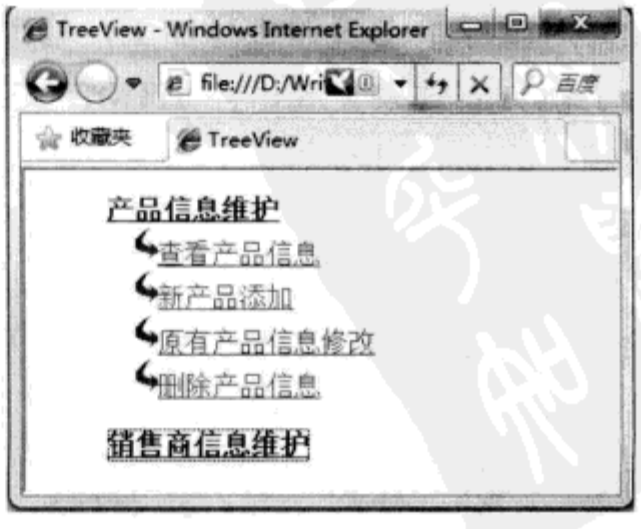


图 5.25　淡入淡出效果 TreeView 展开

1. jQuery的nextAll()函数——查找后续元素

该函数查找当前匹配元素的所有同辈元素。其语法如下：

```
nextAll([expr])
```

注：expr 参数可选，字符串过滤表达式，过滤符合条件的后续同辈元素。

2. 功能实现

淡入淡出 TreeView 的实现步骤如下。

（1）取得需要显示的节点对象，并在其上定义单击事件。

（2）在单击事件中取得下层节点即后续标记元素对象，并在其上加入切换显示状态动画。

为了使页面效果美观，在 HTML 文件中加入 CSS 样式设定。需要设定一些元素的样式，如 ul、ul li、ul li a 等。

3. 代码的实现

具体 CSS 代码请参考光盘内容。在页面中加入无序列表作为菜单主体：

```
<ul>
   <a href="#"><strong>产品信息维护</strong></a>
   <li><img src="img/listicon.jpg" /><a href="#">查看产品信息</a></li>
   <li><img src="img/listicon.jpg" /><a href="#">新产品添加</a></li>
   <li><img src="img/listicon.jpg" /><a href="#">原有产品信息修改</a></li>
   <li><img src="img/listicon.jpg" /><a href="#">删除产品信息</a></li>
</ul>
<ul>
   <a href="#"><strong>销售商信息维护</strong></a>
    <li><img src="img/listicon.jpg" /><a href="#">查看销售商信息</a></li>
    <li><img src="img/listicon.jpg" /><a href="#">添加销售商信息</a></li>
    <li><img src="img/listicon.jpg" /><a href="#">修改销售商信息</a></li>
    <li><img src="img/listicon.jpg" /><a href="#">删除销售商信息</a></li>
 </ul>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码。完整代码如下：

```
<script language="JavaScript">
 $(document).ready(function() {
    var main = $("ul > a");                    //取得代表 TreeView 父节点的元素对象
   main.click(function() {                     //为父节点加入单击事件
    var sub = $(this);
    var item = sub.nextAll("li");              //取得代表 TreeView 所有子节点的元素对象
    item.toggle("slow");                       //修改子节点的显示状态
     });
 });
</script>
```

最后效果如图 5.24 和图 5.25 所示。

上面讲解的两个示例只是 TreeView 雏形，并没有加入节点间的连接线、多形态节点等功能。在下一节要介绍的第三方 TreeView 插件中会见到这些功能。

5.4　第三方 TreeView 插件

本节主要介绍 jQuery 中的 TreeView 插件。以 jQuery.TreeView 为代表的插件可以展现多种形态、多种功能。在这里主要讲解静态数据生成 TreeView 的过程。本插件英文介绍网址为 http://bassistance.de/jquery-plugins/jquery-plugin-treeview/。

1．插件特点

（1）支持静态的树，即一次性将全部数据加载到客户端。

（2）支持异步树，即一次只加载一级或若干级节点，子节点可以异步加载数据。

（3）支持 Checkbox 树（静态/异步），用于选择（如选择组织机构、选择数据字典项）等。

（4）支持节点级联。

（5）能够承载大数据量。

2．参数说明

（1）animated：动画效果。

（2）collapsed：初始状态，为 true 时折叠，为 false 时展开。

（3）unique：同一层只允许同时打开一个，true/false。

（4）persist：目录树状态保存。location 为不保存，cookie 将目录树状态保存在 cookie 文件中。

（5）cookieId：保存目录树状态的 cookie 名称。

（6）control：控制目录树展开与折叠的元素表示。

（7）toggle：目录树中节点触发事件回调函数。

（8）add：添加节点事件。

（9）remove：删除节点事件。

下面来看一下目录树效果，如图 5.26～图 5.29 所示。

上面的几幅图例都是 jQuery 目录树官方网站给出的示例，可以参考它的源代码来了解这个插件的使用方法。其中 HTML 和 CSS 及完整的 JavaScript 代码可以参考光盘内容。

图 5.26 所示的示例使用了目录树最基本的使用方法，没有加入任何参数形成的效果。JavaScript 代码如下：

```
1   // first example
2   $("#browser").treeview();
```

代码中利用 jQuery 选择器选中需要加载的目录树标识#browser，并调用插件初始化函数 treeview()。因为没有给出任何使用参数，所以此目录树中各层节点的展开与折叠是不统

一的，并且刷新页面后，目录树又恢复到页面最初加载完成后的状态。值得注意的是，此目录树的 HTML 中 class="closed"部分，这个类的指定表示该节点下的所有子节点都折叠起来，和插件中对于样式的定义枚举值相符。

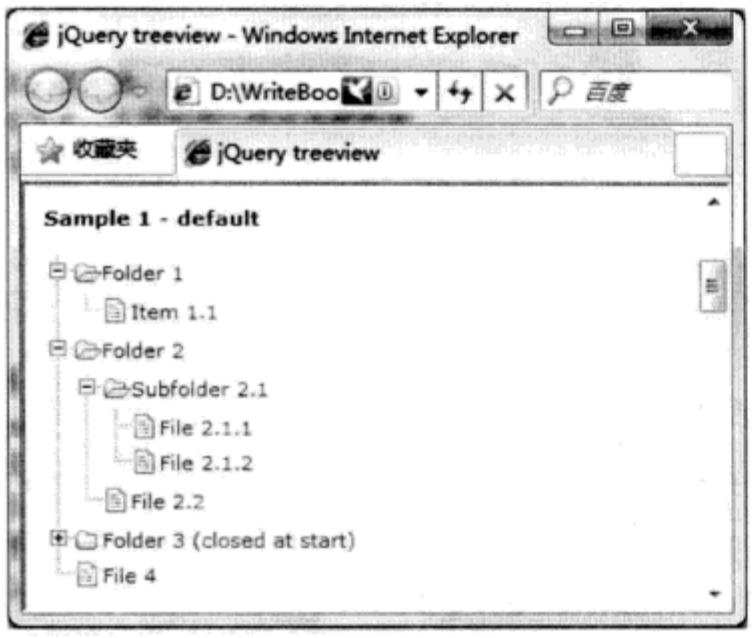


图 5.26　目录树默认配置效果

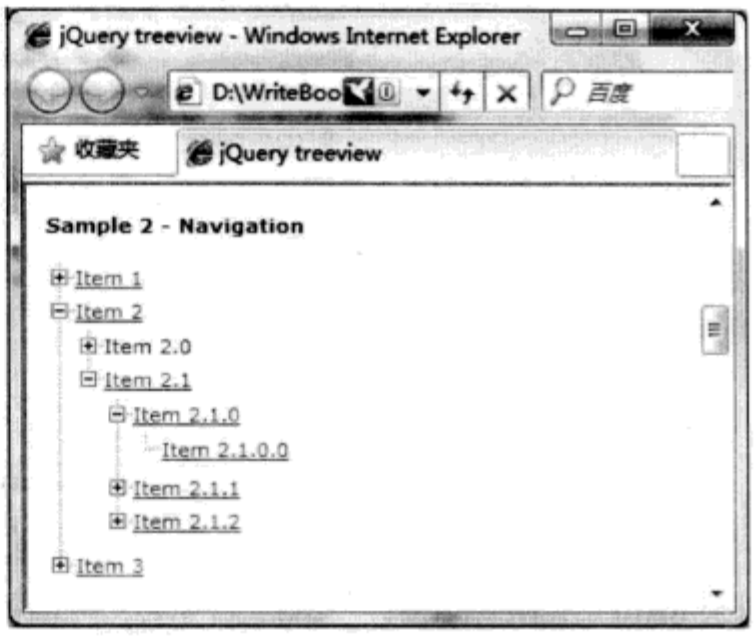


图 5.27　目录树同级展开唯一节点效果

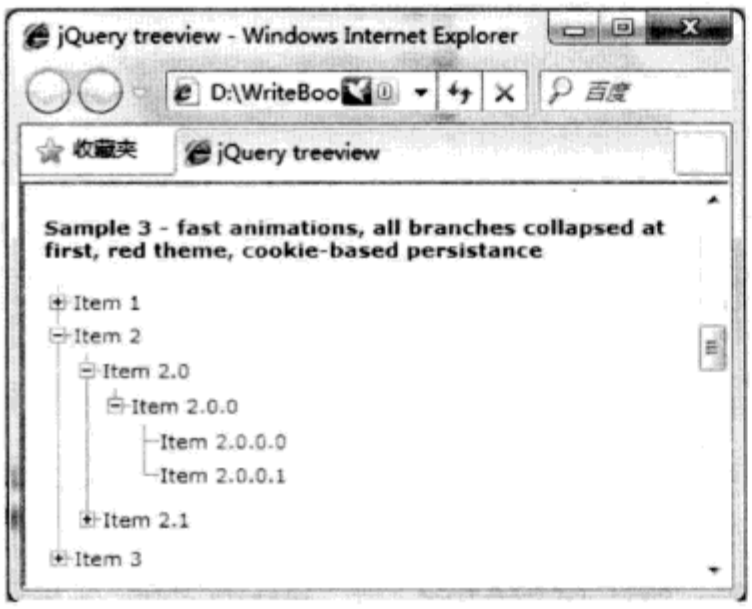


图 5.28　快速动画利用 cookie 存储状态目录树效果

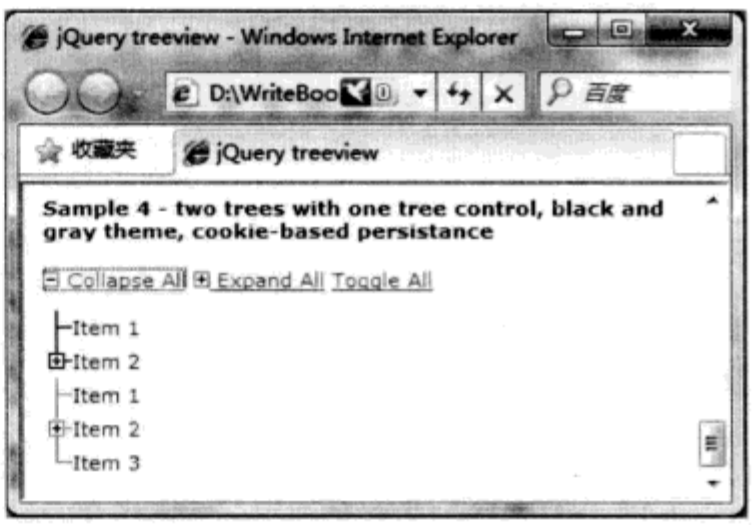


图 5.29　利用控制元素控制不同目录树展开折叠效果

图 5.27 所示示例使用了目录树的三个参数，存储状态，折叠，是否唯一节点展开。JavaScript 代码如下：

```
1   // second example
2   $("#navigation").treeview({
3       persist: "location",
4       collapsed: true,
5       unique: true
6   });
```

代码中使用了上述三个参数，故页面初次加载完成后所有节点都是折叠的。如果想展开节点，会发现在同一级中的兄弟节点中有且只有一个可以处于展开状态。并且，刷新页面后，此目录树又恢复到初始状态。

图 5.28 所示示例与图 5.27 相比使用了 cookie 存储目录树状态，并加入了快速动画及

节点触发事件效果。JavaScript 代码如下：

```
// third example
$("#red").treeview({
    animated: "fast",
    collapsed: true,
    unique: true,
    persist: "cookie",
    toggle: function() {
        alert("Node was toggled");
    }
});
```

代码中使用了目录树的 5 个参数。第 1 个参数即快速动画效果；第 2 个和第 3 个参数与上一个例子相同；第 4 个参数使用了 cookie 存储目录树状态，所以当我们刷新页面后，可以看到目录树没有发生改变；第 5 个参数指定了节点触发事件，这里只是简单地调用了 JavaScript 的 alert()函数呈现一个消息对话框。

图 5.29 所示示例除了指定目录树的存储配置外还利用了控制器功能。JavaScript 代码如下：

```
// fourth example
$("#black, #gray").treeview({
    control: "#treecontrol",
    persist: "cookie",
    cookieId: "treeview-black"
});
```

代码中使用了 control 这个参数，并指定了能同时控制两个不同目录树展开与折叠的元素标识#treecontrol，也可以叫它控制器。可以单击控制器来同时控制两个目录树完全展开与完全折叠。当我们刷新页面时，目录树状态不发生改变。

下面介绍目录树添加与删除节点的示例。这个示例需要用到 add 和 remove 这两个事件参数来指定如何添加和删除节点，效果如图 5.30 所示。

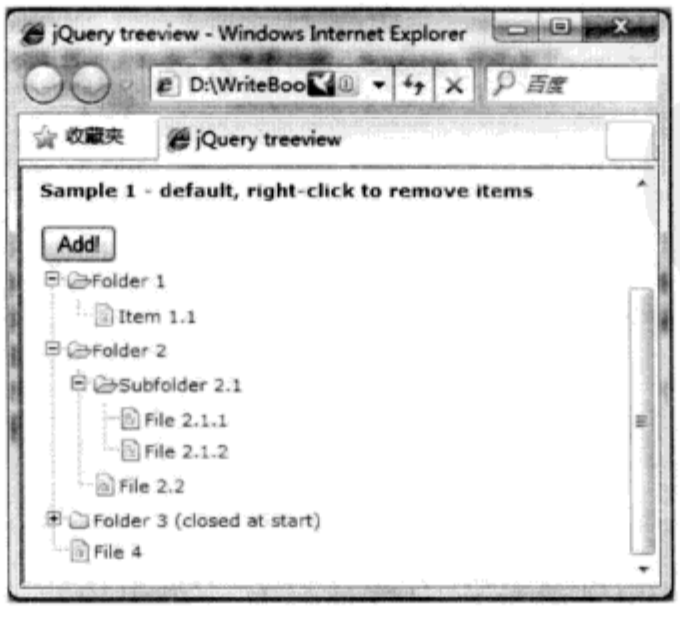


图 5.30　添加和删除目录树节点

实现添加和删除目录树节点的 JavaScript 功能代码如下：

```
<script type="text/JavaScript">
    $(function() {
        $("#browser").treeview();
        $("#add").click(function() {
            var branches = $("<li><span class='folder'>New Sublist
            </span><ul>" +
                "<li><span class='file'>Item1</span></li>" +
                "<li><span class='file'>Item2</span></li></ul>
                </li>").appendTo("#browser");
            // branches 在整个目录树的最后添加一个有两个子节点的节点
            $("#browser").treeview({
                add: branches
            }); //调用目录树的 add 事件参数改变目录树状态
            branches = $("<li class='closed'><span class='folder'>New
            Sublist</span><ul><li><span    class='file'>Item1</span>
            </li><li><span class='file'>Item2</span></li></ul>
            </li>").prependTo("#folder21");
            //在#folder21 位置之前添加一个有两个子节点的节点
            $("#browser").treeview({
                add: branches
            }); //调用目录树的 add 事件参数改变目录树状态
        });
        $("#browser").bind("contextmenu", function(event) {
                                            //绑定右键菜单事件
            if ($(event.target).is("li") || $(event.target).parents
            ("li").length) {
            //判断是否是可删除节点
                $("#browser").treeview({
                    remove: $(event.target).parents("li").filter
                    (":first")          //删除节点
                });
                return false;
            }
        });
    })
</script>
```

上述代码利用了目录树中的 add 事件参数添加了两个节点，remove 事件参数响应右键单击删除节点。

5.5　小　　结

本章主要介绍了如何利用 jQuery 控制网站导航特效。其中，重点是菜单导航和目录树

的基本实现原理和菜单插件，以及目录树插件的使用方法，难点是如何自己构建菜单和目录树。下一章将介绍 jQuery 对表格的控制效果。

5.6　习　　题

【习题一】利用本章所学内容实现一个多级纵向下拉菜单。
【习题二】利用本章所介绍的菜单插件设计一个滑动效果菜单。
【习题三】利用本章所学内容实现一个可多级展开 TreeView。

第6章 设计表格

表格是网页中进行内容布局的一个主要工具。传统的 HTML 表格属性和 CSS 样式设定虽然可以起到美化表格的效果，但是动态变化效果不理想。本章将利用 jQuery 对表格的动态效果进行加工。

6.1 表格基本设计

表格基本设计包括表格边框样式的变换、表格单元格的合并、表格行和列的动态添加与删除、行的上下移动等。下面将就这些内容进行详细的讲解。

6.1.1 表格边框样式的变换

利用 jQuery 变换表格边框样式主要是对表格边框的颜色、样式、像素大小进行变换，这种变换主要是实现表格的高亮显示，突出表格在页面上的显示形式。实现原理主要是利用 jQuery 的元素样式设定函数动态修改边框。其中将要用到 jQuery 函数 ready()、mouseover()、mouseout()、css()。

1. 功能实现

表格边框样式的变换步骤如下。

（1）设定表格的鼠标悬停和离开事件。

（2）在两个事件中分别设定表格边框的不同样式。

2. 实例操作步骤

首先，创建一个包含一行一列表格的静态页面。

```
<table>
    <tr>
    <td>
        jQuery 对表格边框控制
       </td>
    </tr>
 </table>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
<script type="text/ecmascript">
    $(document).ready(function() {
        $("table").mouseover(function(){          //表格区域的鼠标悬停事件
            $(this).css("border", "dotted black 3px");      });
            //更改表格边框样式（黑色,3 个像素,点状）
        $("table").mouseout(function(){           //表格区域的鼠标离开事件
            $(this).css("border", "inset red 5px");     });
            //更改表格边框样式（红色,5 个像素,内嵌）
    });
</script>
```

效果如图 6.1 和图 6.2 所示。

图 6.1　鼠标停在表格上边框样式

图 6.2　鼠标离开表格边框样式

6.1.2　表格单元格的合并

当相邻行或列中出现相同内容时，需要合并单元格。合并单元格后，表格内容更清晰，便于用户了解信息。

表格的众多属性中有 rowspan、colspan 这两个属性。这两个属性的主要作用就是纵向行合并和横向列合并。但是，这两个属性只能以静态形式合并单元格。如果需要根据动态产生数据，动态合并单元格，就需要利用 jQuery 来操作以上两个属性。这里将讨论如何合并文本内容相同的单元格。这种合并适合表格数据动态分类时使用，避免出现不必要的表格嵌套。

表格单元格合并的实现原理是：通过遍历每行中限定个数的单元格，比较其中内容，如果内容相同则动态进行合并，如果找不到相同内容的单元格则放弃合并。其中，使用到 jQuery 函数 ready()、each()、text()、hide()、attr()和 eq、nth-child 属性。

1．jQuery的hide()函数——隐藏元素

该函数隐藏显示的元素。其语法形式如下：

```
hide()
hide(speed,[callback])
```

注：第一种形式是无动画效果地隐藏元素，第二种形式加入了动画效果。如果选择的元素是隐藏的，则不会改变任何内容。

2. jQuery的attr()函数——设置元素属性

该函数为匹配的元素设置或取得属性值。其语法形式如下：

```
语法形式一：attr(name)
语法形式二：attr(properties)
语法形式三：attr(key,value)
语法形式四：attr(key,fn)
```

注：第一种语法形式获取 name 参数指定的属性值，如果元素没有相应属性，则返回 undefine。第二种语法形式将一个“名/值”形式的对象设置为所有匹配元素的属性。第三种语法形式为所有匹配的元素设置一个属性值。第四种语法形式为所有匹配的元素设置一个计算值。

3. jQuery的nth-child属性——获取子元素

该属性匹配所有父元素下第 N 个子元素或奇偶子元素。其语法形式如下：

```
:nth-child(expr)
```

注：它将为每个父元素匹配子元素，并且索引号从 1 开始。

4. 功能实现

jQuery 对表格进行行合并的实现步骤如下。

（1）获得所有行指定索引的列集合，并遍历这个集合。

（2）取得上一行的文本与下面的行进行比对，如果相同，则合并。

首先，创建一个 4 行 4 列表格。

```
1   <table border="1" id="colstb">
2       <tr>
3       <td>jQuery</td>
4           <td>jQuery</td>
5           <td>JavaScript</td>
6           <td>XML</td>
7       </tr>
8       <tr>
9       <td>jQuery</td>
10          <td>jQuery</td>
11          <td>JavaScript</td>
12          <td>XML</td>
13       </tr>
14       <tr>
15          <td>jQuery1</td>
16          <td>jQuery</td>
17          <td>JavaScript</td>
18          <td>XML</td>
19       </tr>
```

```
20        <tr>
21            <td>jQuery</td>
22            <td>jQuery</td>
23            <td>JavaScript</td>
24            <td>XML</td>
25        </tr>
26    </table>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
1   function trowspan(tbl,col){
2       var tdfst = "";
3       var compare="";
4       var tdcur = "";
5       var nums = 1;
6       tdarr = $(tbl + " tr td:nth-child(" + col + ")");
                                          //获取所有行指定列的集合
7       tdarr.each(function(i){           //遍历集合
8       if(i==0){                         //获取首行对象
9           tdfst = $(this);
10          compare=tdfst.text();
11       }else{
12          tdcur = $(this);
13          if(compare==tdcur.text()){ //比较文本内容，如果相同，则进行合并
14              nums++;
15              tdcur.hide();             //隐藏相同内容的单元格
16              tdfst.attr("rowSpan",nums);     //设定合并后的行跨度属性
17          }else{                        //如果不同，则记录新行的内容
18              tdfst = $(this);
19              compare=tdfst.text();
20              nums=1;
21          }
22       }
23      });
24  }
```

效果如图 6.3 所示。

图 6.3　表格动态行合并

jQuery 对表格进行列合并的实现步骤如下。

（1）获取指定行，在指定行上遍历每个单元格文本内容。

（2）如果前后单元格文本内容相同，则合并。

创建表格代码与上面介绍的行合并相同。把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
function tcolspan(tbl,row){
    var tdfst = "";
    var compare="";
    var tdcur = "";
    var nums = 1;
    $(tbl + " tr:eq(" + row + ")").children().each(function(i){
                                          //遍历选定行的所有单元格
        if(i==0){                         //获取首列对象
            tdfst = $(this);
            compare =tdfst.text();
         }else{
            tdcur = $(this);
            if(compare==tdcur.text()){    //比较文本内容，如果相同，则进行合并
                nums++;
                tdcur.hide();                    //隐藏相同内容的单元格
                tdfst.attr("colSpan",nums); //设定合并后的列跨度属性
             }else{                              //如果不同，则记录新列的内容
                tdfst = $(this);
                compare=tdfst.text()
                 nums = 1;
            }
         }
        });

}
```

效果如图 6.4 所示。

图 6.4　表格动态列合并

6.1.3　表格行列的添加与删除

下面介绍如何通过 jQuery 向原有表格动态添加行和列。它的实现原理很简单：将原有

表格看成单元格或者行的集合，对集合进行增、删操作就实现了行和列的添加和删除。其中，要用到 jQuery 函数 append()、appendTo()、clone()、remove()。

1. jQuery函数clone()——复制元素

该函数克隆匹配的 DOM 元素并且选中克隆的副本。其语法形式如下：

```
语法形式一：clone()
语法形式二：clone(true)
```

注：第二种语法形式表示不仅复制元素本身，元素上的事件也一同被复制。在把 DOM 元素中的副本添加到其他位置时适宜使用此函数。

2. 实现步骤

（1）调用 jQuery 的 remove()函数可以直接删除行或者列。

（2）调用 jQuery 的 clone()函数复制要添加的行，再使用 appendTo()函数向表的尾部加入新行。

（3）调用 jQuery 的 append()函数添加列。

首先，创建一个 2 行 2 列的简单表格，并加入添加或者删除行列的处理按钮。

```
1 <table border="1">
2 <tr>
3    <td>JQuery</td>
4    <td>JQuery</td>
5 </tr>
6 <tr>
7    <td>JQuery</td>
8    <td>JQuery</td>
9 </tr>
10 </table>
11 <input type="button" name="button" id="button" value="加列" onclick=
   "AddCol();" />
12 <input type="button" name="button" id="button" value="加行" onclick=
   "AddRow();" />
13 <input type="button" name="button" id="button" value="减列" onclick=
   "RemoveCol();" />
14 <input type="button" name="button" id="button" value="减行" onclick=
   "RemoveRow();" />
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
1   <script type="text/JavaScript">
2   function RemoveRow()
3   {
4       $("tr:last").remove();                              //移除行
5   }
6   function RemoveCol()
```

```
7   {
8       $("tr td:last-child").remove();                    //移除列
9   }
10  function AddRow()
11  {
12      $("tr:first").clone().appendTo("table:first"); //复制并添加行
13  }
14  function AddCol()
15  {
16      $("tr").append("<td>JQuery</td>");                 //添加列
17  }
18  </script>
```

效果如图 6.5～图 6.7 所示。

图 6.5　表格行和列的添加与删除初始表格

图 6.6　表格动态添加行和列

图 6.7　表格动态删除一列

6.1.4　jQuery 控制表格行的上下移动

下面介绍如何通过 jQuery 动态调整行的上下位置，实现客户端手动排序表格行。这种操作适合于用户有手动排序要求的情况。它的实现原理是：比较选定行与插入参考行在表格中上下位置，并重新插入合适的位置。其中，将要用到 jQuery 函数 ready()、click()、each()、addClass()、parent()、children()、text()、position()、insertAfter()、insertBefore()、removeClass()。

1．jQuery函数insertAfter()——向后插入元素

该函数把所有匹配的元素插入到另一个匹配的元素后面。其语法形式如下：

```
InsertAfter(content)
```

注：参数 content 用于匹配元素的 jQuery 表达式，表示插入的参考位置。

2．jQuery函数insertBefore()——向前插入元素

该函数把所有匹配的元素插入到另一个匹配的元素前面。其语法形式如下：

```
InsertBefore(content)
```

注：参数 content 用于匹配元素的 jQuery 表达式，表示插入的参考位置。

3. jQuery函数removeClass()——删除元素类

该函数从所有匹配的元素中删除全部或者指定的类。其语法形式如下：

```
语法形式一：removeClass([class])
语法形式二：removeClass(function(index,class))
```

注：function()函数必须返回一个或多个由空格分隔的 class 名。接受两个参数，index 参数为对象在这个集合中的索引值，class 参数为这个对象原先的 class 属性值。class 参数表示一个或多个要删除的 CSS 类名。

4. 功能实现

jQuery 动态移动行的实现步骤如下。

（1）设定表格中每个单元格的单击事件，在事件中首先定位单元格所在的行。

（2）判定单击次数，标志出需要变换位置的行及参考行位置。

（3）如果两行不是同一行，则选定行；如果在参考行的上面，则插入参考行下面。反之，插入参考行上面。

首先，创建一个 5 行 3 列的表格。

```
<table style="width:60%;">
    <tr>
        <td>1</td>
        <td>jQuery</td>
        <td>JavaScript</td>
    </tr>
    <tr>
        <td>2</td>
        <td>.Net</td>
        <td>C#</td>
    </tr>
    <tr>
        <td>3</td>
        <td>J2EE</td>
        <td>Java</td>
    </tr>
    <tr>
        <td>4</td>
        <td>PHP</td>
        <td> </td>
    </tr>
    <tr>
        <td>5</td>
        <td>Perl</td>
        <td> </td>
    </tr>
</table>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
$(document).ready(function(){
        var destRow=undefined,curtRow,chgRow=undefined, clickcount = 0;
        $('table td').each(function(){                    //遍历每一个单元格
            var currentTd = $(this);
            currentTd.click(function(){                   //为单元格设定单击事件
                clickcount = clickcount + 1;              //单击次数加 1
                var currentRow = currentTd.parent('tr');
                if(clickcount % 2 === 1){ //如果单击奇数次，则更改选定行样式
                    chgRow = currentRow;
                    chgRow.addClass('bgColor');
                }
                else{
                    destRow = currentRow;
                }
                if(chgRow !== undefined && destRow !== undefined){

                    if(chgRow.position().top < destRow.position().
                    top){
                    //判定选定行与参考行的位置上下关系
                            chgRow.insertAfter(destRow);
                                        //在目的行下插入行
                    }
                    else if (chgRow.position().top > destRow.
                    position().top){
                            chgRow.insertBefore(destRow);
                                        //在目的行上插入行
                    }
                destRow = undefined;    //重新初始化变量
                i = 0;
                chgRow.removeClass('bgColor');
                chgRow = undefined;
                }
            });
        });
    });
```

效果如图 6.8～图 6.10 所示。

图 6.8　表格行动态移动初始样式

图 6.9　选定要插入的行

图 6.10　动态插入结束后结果

6.2　表格内容动态排序

本节介绍表格内容动态排序。这个应用充分利用了客户端的闲置处理能力，减轻服务器端处理负载。它的实现原理是：将原有表格转换成二维数组，并通过列的选定及升序或降序的选择，比较选定列中每行单元格的内容，对原数组本身重新排序，再把数组转成表格。其中，将要用到 jQuery 函数 ready()、each()、clich()、siblings()、find()、eq()、get()、children()、text()、append()和 JavaScript 函数 sort()。

1．jQuery函数siblings()——同级元素集合

该函数取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的集合。其语法形式如下：

```
siblings([expr])
```

注：expr 参数表示筛选同辈元素的表达式。

2．jQuery函数eq()——筛选元素

该函数获取第 *N* 个元素。其语法形式如下：

```
eq(index)
```

注：这个元素的位置是从 0 算起的。

3．功能实现

表格动态排序的实现步骤如下。

（1）设定每个表头单元的单击事件。

（2）在单击事件中取出行数组，对数组进行排序。

（3）将排好序的数组重新填入表格。

首先，创建一个 5 行 3 列的表格。

```
1   <table border="1">
2       <thead>
3           <tr>
4               <th> 编号 </th><th> 语言 </th><th>
                 课时 </th>
5           </tr>
6       </thead>
7       <tbody>
8           <tr>
9               <td>c001</td><td>C#</td><td>80</td>
10           </tr>
```

```
        <tr>
            <td>c002</td><td>Java</td><td>70</td>
        </tr>
        <tr>
            <td>c003</td><td>PHP</td><td>60</td>
        </tr>
        <tr>
            <td>c004</td><td>Perl</td><td>50</td>
        </tr>
    </tbody>
</table>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
<script type="text/JavaScript">
$().ready(function(){
        var sort_direction=1;                    //排序标志，1 为升序，-1 为降序
        $('th').each(function(i){
            $(this).click(function(){            //表头单元单击事件
                if(sort_direction==1)
                {
                    sort_direction = -1;
                }
                else
                {
                   sort_direction = 1;
                }
                var trarr = $('table').find('tbody > tr').get();
                                                 //获取行数组
                trarr.sort(function(a, b){       //数组排序
                    var col1 = $(a).children('td').eq(i).
                    text().toUpperCase();
                    var col2 = $(b).children('td').eq(i).
                    text().toUpperCase();
                    return (col1 < col2) ? -sort_direction :
                    (col1 > col2) ? sort_direction : 0;
                  //返回-1 表示 a>b，返回 1 表示 a<b，否则为 0
                });
                $.each(trarr,function(i, row){
                  //将排好序的数组重新填回表格
                    $('tbody').append(row);
                });

            });
        });
});
</script>
```

效果如图 6.11 和图 6.12 所示。

图 6.11　表格动态排序示例——课时降序排序

图 6.12　单击“课时”升序排序

6.3　设 置 分 页

当要在表格中显示的数据量比较多时，通常情况下一屏不能完全显示。但是，如果要用户频繁地拖拉滚动条上下翻看又会影响使用效果。此时可以使用 jQuery 对表格应用分页效果。它的实现原理是：默认加载表格所有的行，然后根据当前页的起始行数和结束行数显示当前页的行，隐藏其他行。其中，将用到 jQuery 函数 ready()、bind()、find()、hide()、slice()、show()、trigger()、appendTo()、append()、insertAfter()和 JavaScript 函数 Math.ceil()。

1．jQuery函数slice()——元素子集

该函数选取一个匹配的元素子集。其语法形式如下：

```
slice(start,[end])
```

注：start 参数表示选取子集的位置，元素从 0 开始编号。end 为可选参数，表示结束选取的位置，如果不指定，则到末尾。

2．功能实现

（1）设定起始页位置和每页大小。
（2）绑定自定义的分页事件，在事件中隐藏当前页不需要显示的行。
（3）为表格添加页链接。
（4）绑定链接的单击事件，在事件中触发表格分页事件。
首先，创建一个和上一节示例相同的表格（代码见上一节），把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
1   $(document).ready(function()
```

```
{
    var $table = $('table');
    var currentPage = 0;                                    //当前页索引
    var pageSize = 2;                                       //每页行数(不包括表头)
    $table.bind('paging', function()                        //绑定分页事件
    {
        //隐藏所有的行，取出当前页的行显示
        $table.find('tbody tr').hide()
            .slice(currentPage * pageSize, (currentPage + 1) * page-
            Size).show();
    });
    var sumRows = $table.find('tbody tr').length;  //记录总行数
    var sumPages = Math.ceil(sumRows/pageSize);    //总页数
    var $pager = $('<div class="page"></div>');    //分页 div
    for( var pageindex = 0; pageindex < sumPages; pageindex++ )
    {
        //为分页标签加上链接
        $('<a href="#" ><span>'+ (pageindex+1) +'</span></a>')
            .bind("click", { "newPage": pageindex }, function(event)
            {
                currentPage = event.data["newPage"];
                $table.trigger("paging");
            })
          .appendTo($pager);
        $pager.append(" ");
    }
    $pager.insertAfter($table);                             //分页 div 插入 table 末尾
    $table.trigger("paging");                               //触发分页事件
});
</script>
```

效果如图 6.13 所示。

图 6.13　表格动态分页

6.4　表格行条纹效果

表格的行条纹效果是表格最常用也是最容易掌握的技术。它的实现原理很简单，区分出表格的奇数行与偶数行，为不同的行设置不同的背景色。其中，要用到 jQuery 函数 addClass()和属性:nth-child()。

实现表格行条纹效果先要取出表格中所有的偶数行，为其加上不同的背景色。操作步骤如下。

首先，创建一个表格，此表格的具体代码参考光盘内容。把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
1  <script type="text/JavaScript">
2     $(document).ready(function() {
3        $("table tr:nth-child(even)").addClass("striped");
                                                    //为偶数行设定样式类
4        $("tr").mouseover(function() {
5           $(this).addClass("over");
6        }).mouseout(function() {
7           $(this).removeClass("over");
8        })
9     })
10  </script>
```

在上面的代码中加入了鼠标的悬停与离开事件，主要是为了让表格看起来更生动，效果如图 6.14 所示。

图 6.14　表格行条纹效果

6.5　表格的折叠和展开

表格的折叠与展开是表格动态显示内容的一种常用方式。它适用于表格中某些行具有大数据量，而页面加载后又不希望马上显示的情况。表格的折叠与展开的实现原理是：利用对表格行的显示样式的设定来实现表格的折叠与展开，显示即展开，隐藏即折叠。其中，将要用到 jQuery 函数 ready()、toggle()、css()。

表格的折叠与展开的实现步骤如下。

（1）设定标题行的触发事件函数。

（2）在触发事件函数中根据鼠标单击次数更改描述行的显示样式。

首先，创建一个 4 行 1 列的表格：

```
<table border="1">
<tr class="titletr" id="one"><td>jQuery</td></tr>
<tr id="descone" style="display:none"><td>
 jQuery 是一个 JavaScript 库，它有助于简化 JavaScript? 以及 Asynchronous
 JavaScript + XML (Ajax) 编程。
</td>
</tr>
<tr class="titletr" id="two"><td>JavaScript</td></tr>
<tr id="desctwo" style="display:none"><td>
JavaScript 是一种基于对象和事件驱动的客户端脚本语言。 JavaScript 最初的设计是为了
 检验 HTML 表单输入的正确性
</td>
</tr>
</table>
```

把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
 <script type="text/JavaScript">
$(document).ready(function(){
  $('.titletr').toggle(function(){                    //标题行鼠标单击事件
      $('#desc'+this.id).css("display","block");      //显示描述行
      },function(){
       $('#desc'+this.id).css("display","none");      //隐藏描述行
  });
  });
</script>
```

效果如图 6.15 和图 6.16 所示。

图 6.15　表格行折叠

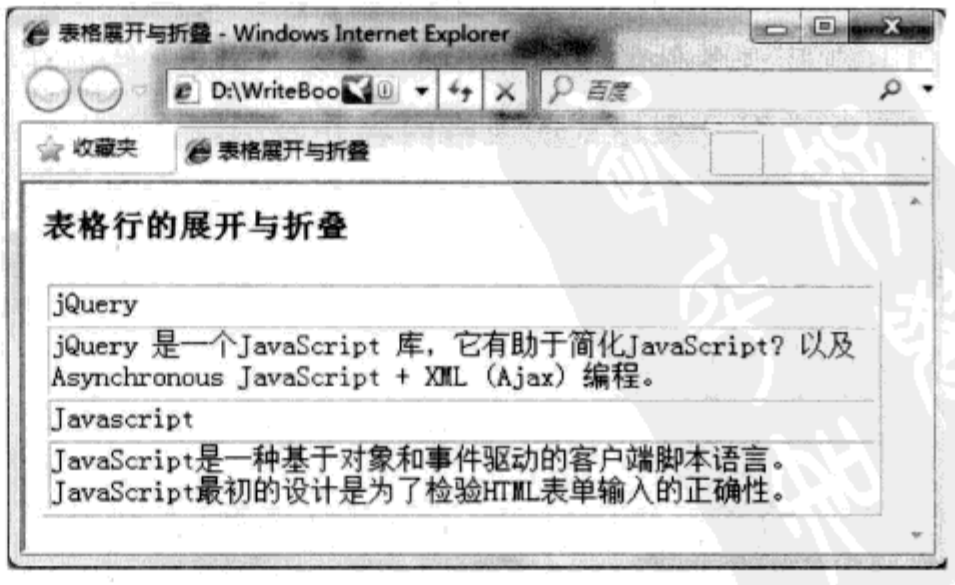


图 6.16　表格行的展开

6.6　表格动态内容筛选

在客户端，有时候用户需要能够分类筛选查看表格中的内容。在传统的 Web 应用中，这种功能需要服务器的支持才能实现。现在，有了 jQuery 就可以直接在客户端完成动态表格内容的筛选，而不必再依靠服务器。它的实现原理是：将表格中所有内容全部隐藏，然后通过用户的筛选条件过滤出需要的内容重新显示。其中，将要用到 jQuery 函数 ready()、keyup()、hide()、filter()、val()、show()。

1．jQuery函数keyup()——键盘释放

该函数触发每一个匹配元素的键盘释放事件。其语法形式如下：

```
语法形式一：keyup(fn)
语法形式二：keyup()
```

注：第一种语法形式为每一个匹配的元素绑定一个键盘释放事件。第二种语法形式触发每一个匹配元素的键盘释放事件。

2．jQuery函数filter()——筛选元素

该函数筛选出匹配元素的集合。其语法形式如下：

```
语法形式一：filter(expr)
语法形式二：filter(fn)
```

注：第一种语法形式中，expr 参数是筛选表达式。第二种语法形式中，fn 函数参数返回每一个元素是否匹配的结果，并删除不匹配的元素。

3．功能实现

表格动态内容筛选的实现步骤如下。

（1）在筛选条件的文本框中加入键盘释放事件，在键盘事件中隐藏所有表格内容。

（2）筛选符合表达式的表格内容并显示。

首先，创建一个表格，具体代码参考光盘内容。把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
1 <script type="text/JavaScript">
2  $(function(){
3       $("#condition").keyup(function(){          //键盘释放事件
4         $("table tbody tr")
5                   .hide()                        //隐藏所有内容
6                   .filter(":contains('"+( $(this).val() )+"')")
```

```
                                                    //筛选符合表达式的元素
7                       .show();                    //显示筛选后的内容
8       }).keyup();                                 //触发键盘释放事件
9   })
10 </script>
```

效果如图 6.17 和图 6.18 所示。

图 6.17　表格动态筛选加载

图 6.18　表格动态筛选效果

6.7　可编辑表格

单纯的显示内容的表格看起来相对死板。有时候用户需要对表格的内容进行动态编辑，这个时候就需要利用 jQuery 来修改单元格的状态，适应用户编辑操作。它的实现原理是：利用鼠标单击单元格，使其从普通文本显示状态转成输入状态，即用文本框代替原来的文本。当用户输入完成或者取消输入后，单元格再恢复普通文本状态。其中，将要用到 jQuery 函数 ready()、click()、children()、html()、css()、width()、val()、appendTo()、trigger()、keyup()。

1. jQuery函数html()——操作元素的HTML内容

该函数获取或者设置匹配元素的 HTML 内容。其语法形式如下：

```
语法形式一：html()
语法形式二：html(val)
语法形式三：html(function(index,html))
```

注：第一种语法形式从匹配的元素中取出 HTML 内容。第二种语法形式将参数 val 中的内容填入匹配元素，作为它的 HTML 内容。第三种语法形式将 function()函数返回的 HTML 内容填入匹配的元素中，index 为元素在集合中索引位置，html 参数为原来的 HTML 内容。

2. jQuery函数val()——元素的值

该函数获取或者设置匹配元素的当前值。其语法形式如下：

```
val()
val(value)
val(function(index,value))
val(array)
```

注：第一种语法形式获取匹配元素的当前值。第二种语法形式为匹配元素设置参数 value 所代表的值。第三种语法形式为匹配元素设置函数 function 返回的值，index 为元素在集合中的位置，value 为原来的值。第四种语法形式主要为选择性输入元素赋值，如 check、select、radio 等。

3. 功能实现

可编辑表格的实现步骤如下。

（1）设定单元格的单击事件，判定被单击单元格是否已经是可编辑状态。

（2）取出单元格原有内容，向单元格中加入文本框，并把原有内容显示在文本框中。

（3）当用户编辑完成或者取消编辑后，将文本框值取出，删除文本框，并将值在单元格中显示。

首先，创建一个表格，具体 HTML 代码参考光盘内容。把 jQuery 库引入进来：

```
<script src="jslib/jquery-1.6.min.js" type="text/JavaScript"></script>
```

然后，添加 JavaScript 代码：

```
<script type="text/JavaScript">
    $(document).ready(function(){
    $("tbody td").click(function(){                    //单元格单击事件
        var altertd = $(this);
        if(altertd.children("input").length>0){        //判断单元格是否是编辑状态
         return false;
    }
        var htmlcontent = altertd.html();              //取出单元格原有内容
        altertd.html("");                              //清空单元格内容
        var textbox = $("<input type='text' />").css("border-width",
        "1") .css("background-color","gray").width(altertd.width()).
        val(htmlcontent).appendTo(altertd);
         //将文本框加入单元格并显示原有内容
        textbox.trigger("focus").trigger("select");
        textbox.click(function(){
            return false;
        });
        textbox.keyup(function(event){
            var keycode = event.which;
```

```
17              if(keycode == 13){    //用户单击回车键表示编辑完成，单元格刷新内容
18                  var inputtext = $(this).val();
19                  altertd.html(inputtext);
20              }
21              if(keycode == 27){ //用户单击 Esc 键表示放弃编辑，单元格恢复原始内容
22                  altertd.html(htmlcontent);
23              }
24          });
25      });
26  });
27  </script>
```

效果如图 6.19～图 6.21 所示。

图 6.19　可编辑表格初始状态

图 6.20　可编辑表格编辑状态

图 6.21　可编辑表格编辑完成

6.8　表 格 插 件

本节将介绍三个表格插件，它们都具有丰富的表格操作功能，方便我们使用。有了这些插件，可以节省为了达到表格效果而自己编写 jQuery 代码所花费的大量时间。

6.8.1　jExpand 表格插件

jExpand 表格插件利用 CSS 和 jQuery 扩展了表格的收缩与展开功能。它的实现原理和前面讲到的收缩与展开表格是相同的。它的英文网址为 http://www.jankoatwarpspeed.com/post/2009/07/20/Expand-table-rows-with-jQuery-jExpand-plugin.aspx。

该插件的效果如图 6.22 所示。

下面介绍如何使用这个插件。可以说这个插件是最简单的一个插件。具体代码参考光盘内容。这里只说明如何调用。调用代码如下：

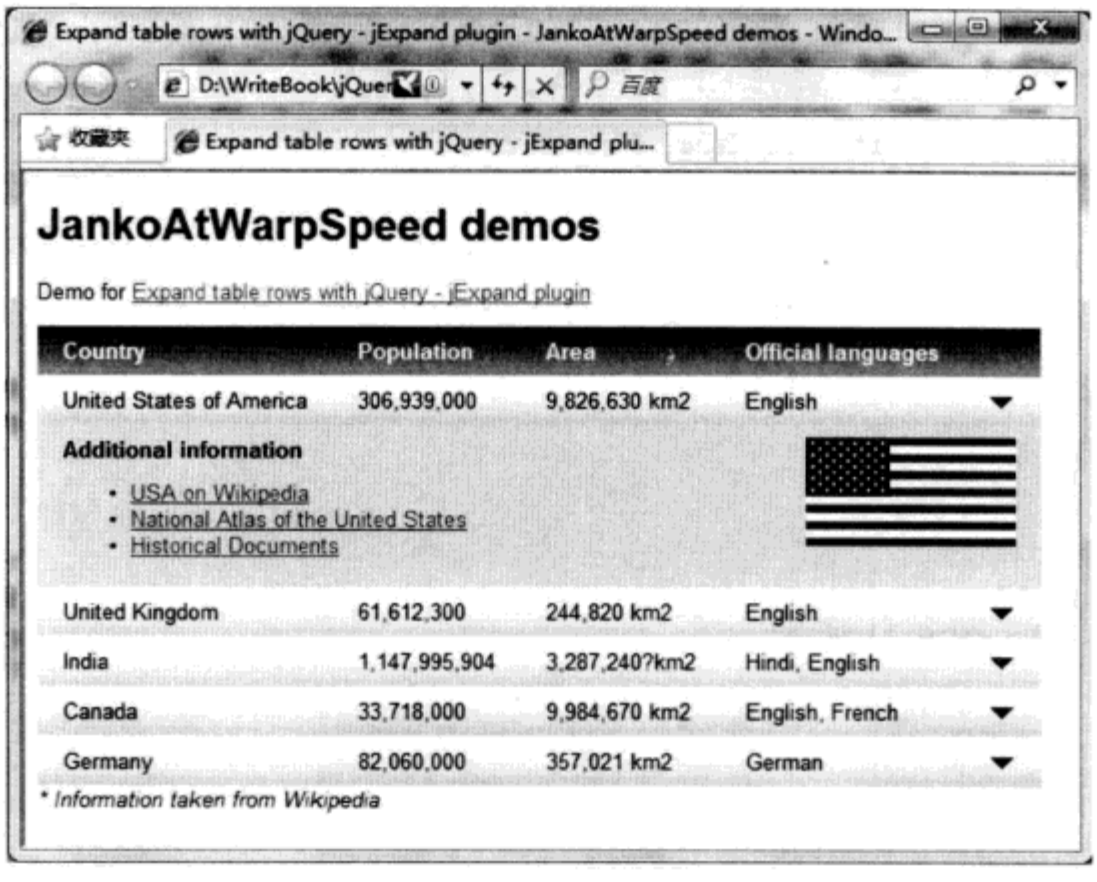


图 6.22　jExpand 表格插件应用效果

```
<script type="text/JavaScript">
    $(document).ready(function(){
        $("#report").jExpand(); //#report，使用插件的表格 ID
    });
</script>
```

这个插件不需要任何初始化参数，只需要指明在哪个表格上使用这个插件就可以了。

6.8.2　Table Pagination 表格分页插件

Table Pagination 表格分页插件是一个带工具栏的可以做分页操作的表格插件。它的实现原理与前面介绍的表格分页原理相同，也是隐藏当前页不需要显示的行。它的实现效果如图 6.23 所示。

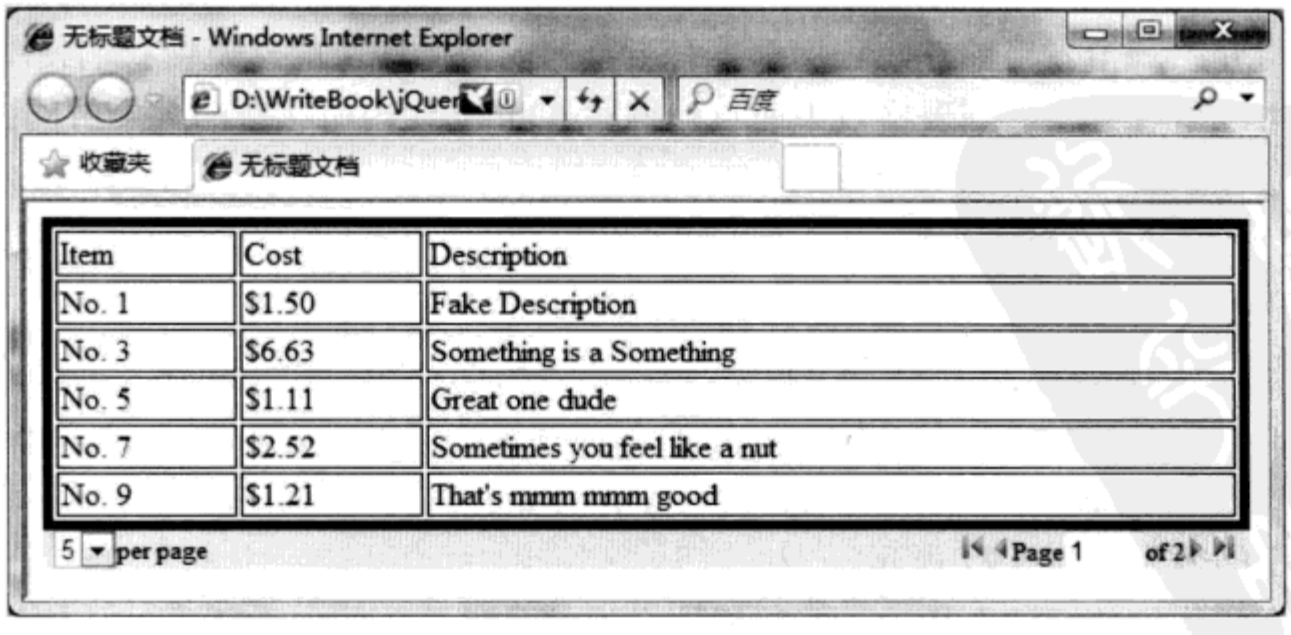


图 6.23　带工具栏的表格分页插件

接下来介绍该插件的使用方法。具体代码请参考光盘内容。这里主要说明插件如何初始化。

```
<script type="text/JavaScript">
  $(function(){
      $('tbody tr:odd', $('#menuTable')).hide();
          var options = {                                    //插件初始化参数对象
            currPage : 2,                                    //当前页
            ignoreRows : $('tbody tr:odd', $('#menuTable')),
                                                             //忽视显示的行
            optionsForRows : [2,3,5],                        //每页可以显示行数的集合
            rowsPerPage : 5,                                 //每页的行数
            firstArrow : (new Image()).src="./img/firstBlue.gif",
                                                             //第一页图标
             prevArrow : (new Image()).src="./img/prevBlue.gif",
                                                             //前一页图标
             lastArrow : (new Image()).src="./img/lastBlue.gif",
                                                             //最后一页图标
             nextArrow : (new Image()).src="./img/nextBlue.gif"
                                                             //下一页图标
          }
          $('#menuTable').tablePagination(options);
                                                    //利用参数对象初始化插件
  });
</script>
```

利用上面的初始化代码可以得到如图 6.24 所示的结果。

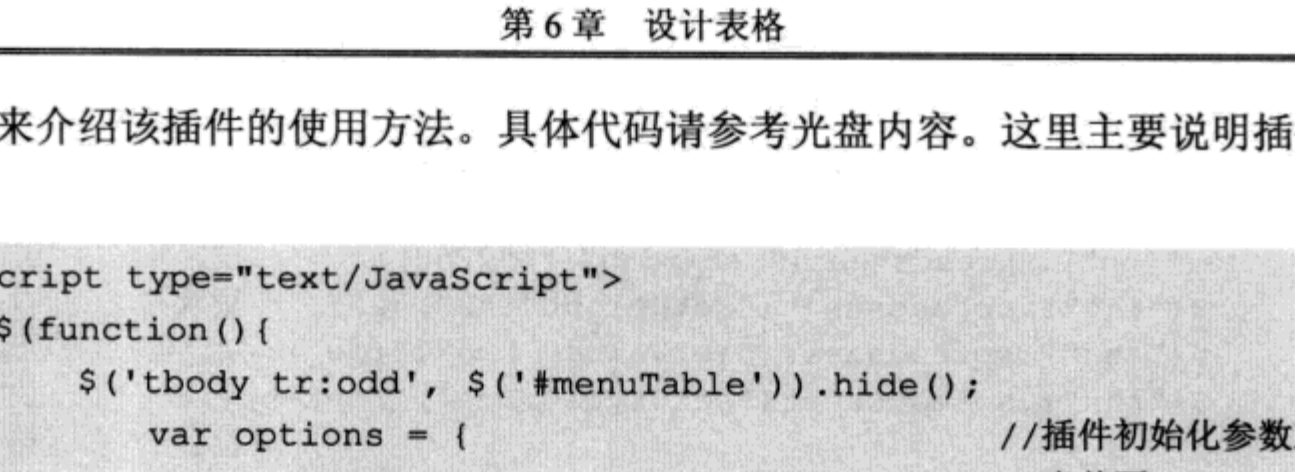


图 6.24　参数设定的带工具栏的表格分页插件

6.8.3　Spreadsheet Web 电子表格插件

Spreadsheet Web 电子表格插件实现了在浏览器中模仿 Excel 形式的电子表格，它的效果如图 6.25 所示。

该插件的调用方式代码如下：

```
<script type="text/JavaScript">
    $(function()
```

```
3        {
4        $("#hn").spreadsheet({fullscreen:true});
                                          //设置插件全屏显示，即填充整个浏览器窗口
5       $("#hn").spreadsheet("setWidth","b",200);     //设置列名为“b”的列的列宽
6        $("#hn").spreadsheet("setHeight","2",30);  //设置第 2 行的行高
7        $("#hn").spreadsheet("setValue","c5","foo");//设置 c5 单元格内容
8        $("#hn").spreadsheet("setStyle","c5","font-style","italic");
                                                   //设置 c5 单元格字体
9        $("#hn").spreadsheet("addName","b2:c3","newRange");
                                                   //为表格区域加上名称
10        });
11    </script>
```

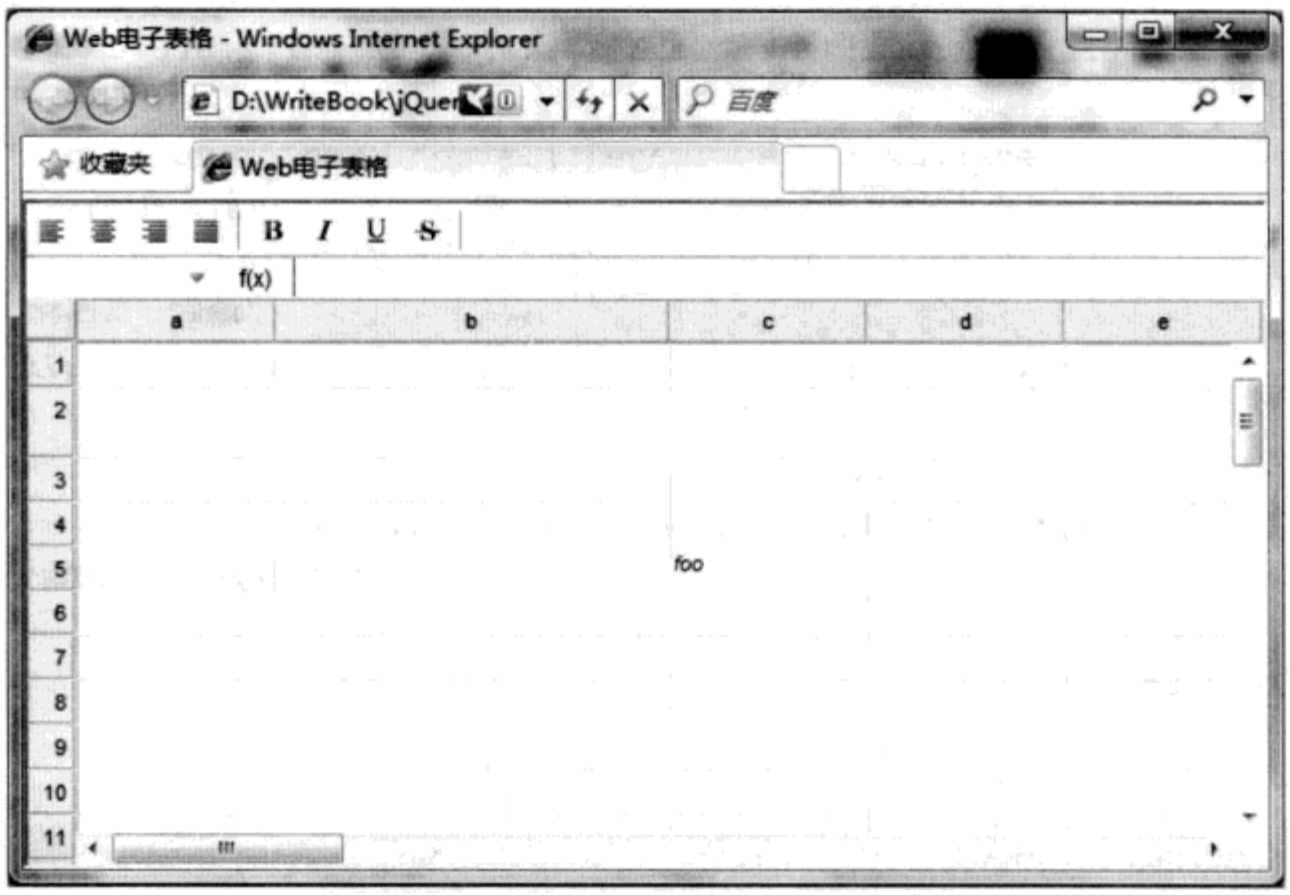


图 6.25　Web 电子表格

上面介绍的三个表格插件相对都比较简单。在实际使用插件的过程中还有一些功能更强大、使用更复杂的表格插件。例如，DataTables、Flexigrid 等插件，这些插件的强大功能需要服务器提供数据源，因为本书主要讲解前台设计部分，所以，对于复杂表格插件不做讲解，读者可以自行研究。

6.9　小　　结

本章主要介绍了如何利用 jQuery 控制表格特效。其中，主要介绍表格分页、合并单元格、表格的折叠与展开，动态内容筛选、表格的可编辑性的基本实现原理和简单表格插件的使用方法。重点是表格分页、合并单元格、动态内容筛选，表格的可编辑性。难点是如何由自己构建综合效果的表格，即如何将本章所介绍的内容作用在同一张表格上。下一章将介绍 jQuery 对表单的控制效果。

6.10　习　　题

【习题 1】利用本章所学内容实现可编辑、可排序并带有分页功能的表格。

【习题 2】利用本章所学内容实现根据用户输入的查询条件，高亮显示符合查询内容的行。

【习题 3】利用本章所学内容实现折叠具有相同内容的行，并可展开。

第7章 设计表单

本章将讨论 jQuery 对表单元素的控制。表单是交互式网页的主要工具，用户通过表单登录信息，并提交给服务器，然后由服务器返回结果。但是，传统的表单过于呆板，功能单一。可以使用 jQuery 增强表单功能，美化表单元素。

7.1 表单基本操作

表单最简单的操作就是提交功能。本节利用 jQuery 来扩展表单的一些常用简单操作，如清空、重置、表单元素的赋值与取值等操作。

7.1.1 表单清空

在某些网站上会看到表单的底部除了正常的“提交”按钮和“重置”按钮以外，还有一个“清空”按钮。这个清空按钮不像“提交”和“重置”按钮一样已经完善了功能。它实际上是一个普通按钮，清空功能需要我们自己来完成。下面就来介绍“清空”按钮所要实现的功能。

它的原理是：获取表单中所有元素，除了按钮和隐藏元素，利用 jQuery 的赋值函数为每一个元素赋一个空内容。其中，需要用到 jQuery 函数 not()、val()、removeAttr()和 jQuery 选择器:input、:button、:submit、:reset、:hidden。

1. jQuery函数not()——删除元素

该函数删除与指定表达式匹配的元素。其语法形式如下：

```
not(expr)
```

注：expr 参数可以为一个表达式、一个元素或者一组元素。

2. jQuery选择器

jQuery 中代表表单元素的选择器如表 7.1 所示。

表 7.1　jQuery表单元素选择器

选择器	语法	说明
:input	:input	匹配所有 input、textarea、select 和 button 元素
:button	:button	匹配所有按钮
:submit	:submit	匹配所有提交按钮
:reset	:reset	匹配所有重置按钮
:hidden	:hidden	匹配所有隐藏元素

3. 功能实现

表单清空的实现步骤如下。

（1）取得所有表单元素。

（2）删除按钮和隐藏元素。

（3）为表单元素设置空内容，去除选择性元素的选中属性值。

首先，需要创建一个表单，具体 HTML 代码参考光盘内容。把 jQuery 库引入进来：

```
<script src="jslib/jquery.js" type="text/javascript"></script>
```

然后，添加 JavaScript 代码：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#empty").click(function(){
4  $(':input','#form1').not(':button, :submit, :reset, :hidden').val ('').
   removeAttr("checked").removeAttr("selected");  //清空表单中用户输入内容
5       });
6   });
7 </script>
```

上述代码中，第 4 行代码实现清空表单功能。val('')表示清除所有文本框的值，包括文本区域。调用 removeAttr()函数是清除 radio、checkbox 和 select 的选择效果。当向表单填入内容后，效果如图 7.1 所示。

图 7.1　表单填写内容

当单击“清空”按钮后，表单效果如图 7.2 所示。

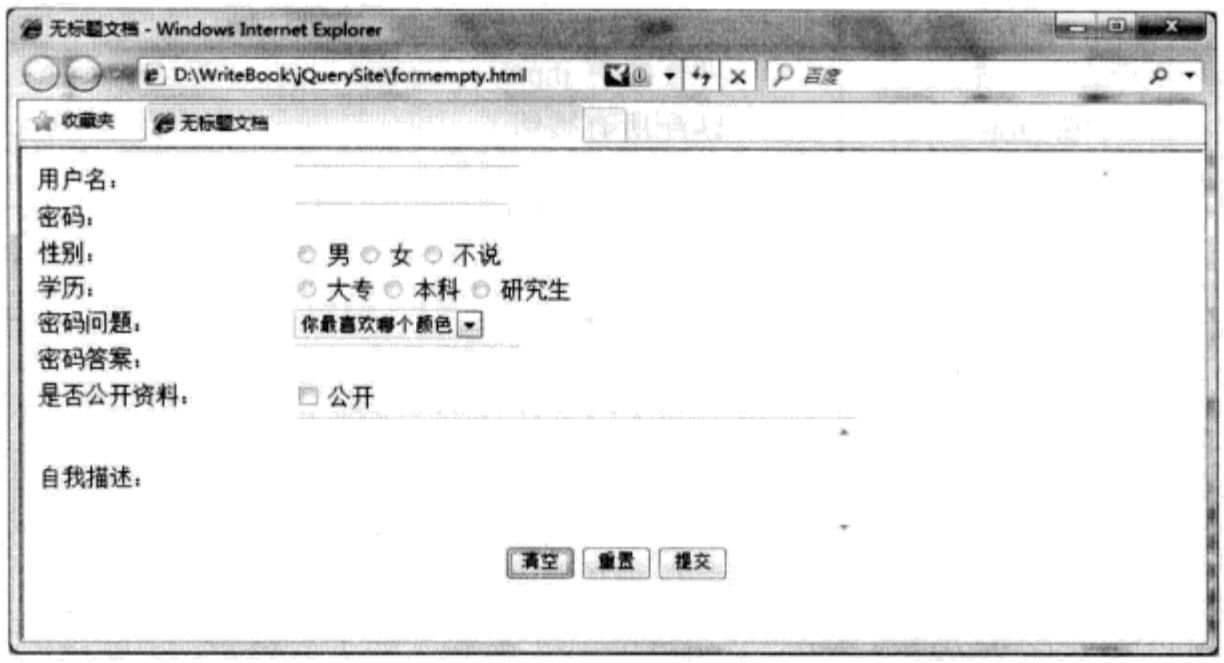


图 7.2　清空表单

7.1.2　重置表单

虽然在表单元素中有实现重置表单元素功能的“重置”按钮，但是，为了能够更灵活地操作表单重置动作，还是可以用 jQuery 来实现表单重置操作。在实际实现中，jQuery 本身的函数是不能够操作重置表单的。所以，还是需要由 DOM 中的 reset()方法来实现。表单的初始状态如图 7.3 所示。

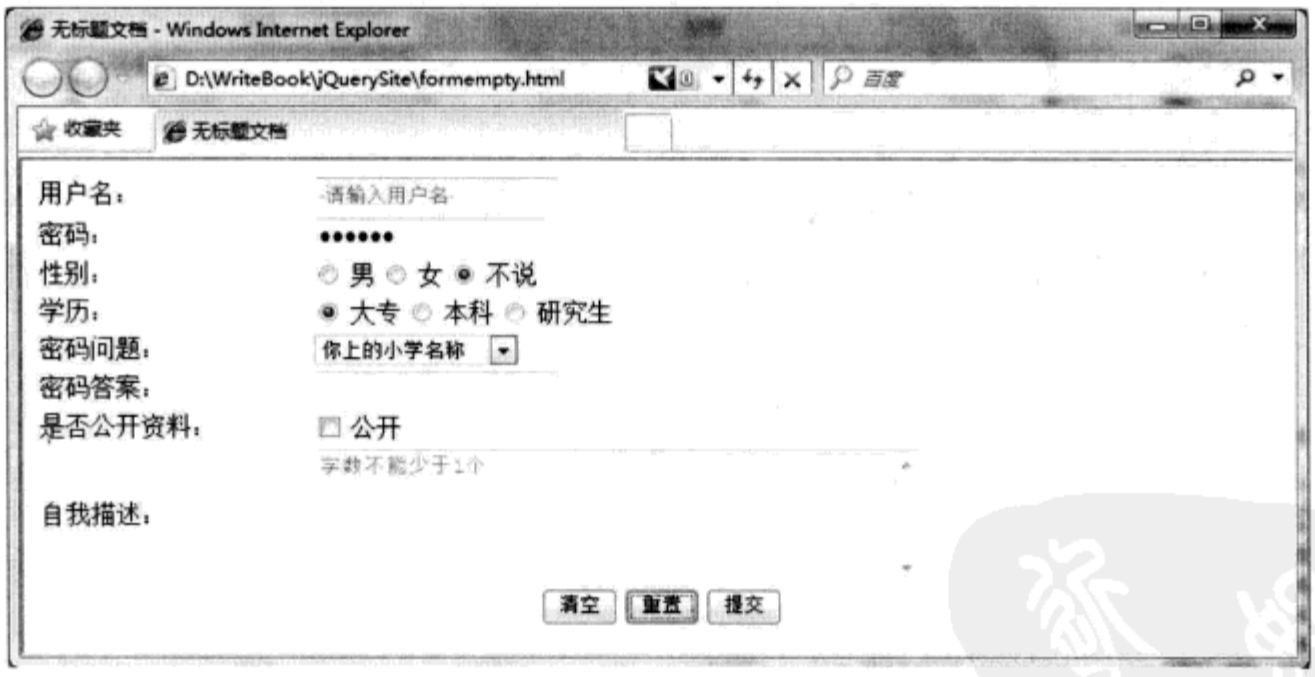


图 7.3　表单的初始状态

在 HTML 文件头部加入如下代码：

```
1   $("#init").click(function(){
2         $("#form1")[0].reset();          //重置表单内容
3   });
```

其中，“#init”是添加实现重置功能的按钮 ID，$("#form1")[0]表示我们的表单对象，通过这个对象调用 reset()方法重置表单。

7.1.3　表单元素的赋值与取值

表单元素的赋值与取值在 JavaScript 的 DOM 中就可以实现。但是，使用 jQuery 会更便捷。可以通过 jQuery 在客户端对表单元素的值进行加工，而且不用页面刷新，实现用户友好的界面。这个功能主要是通过 jQuery 的 val()函数和表单元素的选择器来实现的。

1．表单元素取值实现

在前面创建的表单基础上，添加一个文本区域来显示获取到的表单元素中用户输入的内容和一个触发获取表单内容的按钮。具体 HTML 代码参考光盘内容。取值功能实现代码如下：

```
$("#getresult").click(function(){
        var content="";
        content+="username:"+$("#username").val()+";"; //获取用户名内容
        content+="password:"+$("#pwd").val()+";";       //获取密码内容
        content+="sex:"+$("input[@name=sex][@checked]").val()+";";
                                                  //获取性别选择内容
        content+="grade:"+$("input[@name=grade][@checked]").val()+";";
                                                  //获取学历选择内容
        if($("#question").val()==0)               //获取密码问题选择及答案内容
        content+="你上的小学名称:"+$("#answer").val()+";";
        else if($("#question").val()==1)
        content+="你母亲的生日:"+$("#answer").val()+";";
        else if($("#question").val()==2)
        content+="你最喜欢哪个颜色:"+$("#answer").val()+";";
        if($("#public").attr("checked"))
        content+="公开资料"+";";                        //获取是否公开资料选择内容
        content+="自我描述："+$("#desc").attr("value");
                                                  //获取自我描述文本域内容
        $("#result").attr("value",content);//显示
    });
```

实现效果如图 7.4 所示。

图 7.4　获取表单元素内容值

上面的代码的第 7～12 行显得逻辑稍微有些复杂。可以对其进行简化修改，代码如下：

```
content+=$("select[@name=question] option[@selected]").text()+":"+$("#an
swer").val();
```

我们不再判断下拉列表框的取值并根据取值取下拉列表框中的文本，而是直接获取选中文本。这样操作降低了程序代码的复杂性。

2. 表单元素赋值实现

还是以上面的表单为例，在页面中加入一个自动填写表单的按钮。利用 jQuery 为表单元素赋值。赋值功能实现代码如下：

```
1   $("#write").click(function(){
2       $("#username").attr('value','admin');              //填写用户名内容
3       $("#pwd").attr('value','admin');                   //填写密码
4       $("input[@name=sex]").get(1).checked=true;         //设置性别为女
5       $("input[@name=grade]").get(1).checked=true;       //设置学历为本科
6       $("#question").attr('value',0);                  //设置密码提示问题为第一个
7       $("#answer").attr('value','aaaa');               //填写密码提示答案
8       $("#public").attr('checked',true);               //设置资料可以公开
9       $("#desc").attr('value','我的自我描述');          //填写自我描述
10  });
```

当单击“填写表单”按钮后，执行上述代码，效果如图 7.5 所示。

图 7.5　自动填写表单

7.2　表单验证框架

jQuery 表单验证框架 Validate 是 jQuery 的一个著名插件。它在客户端发挥了很大的表

单元素验证功能。下面分 4 个步骤介绍这个验证框架：基本验证功能、API 使用方法、自定义验证方法、其他常用表单元素的验证方式。

7.2.1　基本验证功能

现在的验证功能插件基本上都具有两个功能：默认校验规则和默认提示信息。

1. 默认校验规则

Validate 具有自己的校验规则，如表 7.2 所示。

表 7.2　Validate校验规则

校 验 规 则	取　　值	功　　能
required	true/false	表示验证必填字段
remote	服务器端处理程序	表示通过 Ajax 调用处理程序验证输入值
email	true/false	表示验证电子邮件标准格式
url	true/false	表示验证网址格式
date	true/false	表示验证日期格式
dateISO	true/false	表示验证（ISO）日期格式
number	true/false	表示验证数字格式
digits	true/false	表示验证整数
creditcard	true/false	表示验证信用卡格式
equalTo	另一个网页元素 ID 值	表示验证两个元素内容是否相同
accept		表示验证拥有合法后缀名的字符串
maxlength	整型	表示最大长度
minlength	整型	表示最小长度，它和上面的最大长度在判断中文时，单个汉字算一个字符
rangelength	两个整型值的组合	表示输入长度范围
range	数字	表示输入值范围
max	数字	表示输入值的最大值
min	数字	表示输入值的最小值

2. 默认提示信息

这个验证框架为每种验证都提供了相应的验证消息，如表 7.3 所示。

表 7.3　默认验证消息

验 证 规 则	默 认 消 息
required	This field is required.
remote	Please fix this field.
email	Please enter a valid email address.
url	Please enter a valid URL.
date	Please enter a valid date.
dateISO	Please enter a valid date (ISO).
number	Please enter a valid number.

续表

验 证 规 则	默 认 消 息
digits	Please enter only digits
creditcard	Please enter a valid credit card number.
equalTo	Please enter the same value again.
accept	Please enter a value with a valid extension.
maxlength	$.validator.format("Please enter no more than {0} characters.")
minlength	$.validator.format("Please enter at least {0} characters.")
rangelength	$.validator.format("Please enter a value between {0} and {1} characters long.")
range	$.validator.format("Please enter a value between {0} and {1}.")
max	$.validator.format("Please enter a value less than or equal to {0}.")
min	$.validator.format("Please enter a value greater than or equal to {0}.")

表格中{0}和{1}分别表示设定验证规则时预先给出的比较参数。不幸的是，这些验证消息全是英文的。万幸的是，我们可以自己修改这些验证消息，把它们变成中文的。下面对前一节的表单进行修改，增加验证功能。HTML 代码请参考光盘内容。

首先，需要创建一个 JavaScript 文件，用来显示中文的验证消息。代码如下：

```
1   jQuery.extend(jQuery.validator.messages, {
2         required: "必选字段",
3  remote: "请修正该字段",
4  email: "请输入正确格式的电子邮件",
5  url: "请输入合法的网址",
6  date: "请输入合法的日期",
7  dateISO: "请输入合法的日期 (ISO).",
8  number: "请输入合法的数字",
9  digits: "只能输入整数",
10  creditcard: "请输入合法的信用卡号",
11  equalTo: "请再次输入相同的值",
12  accept: "请输入拥有合法后缀名的字符串",
13  maxlength: jQuery.validator.format("请输入一个长度最多是 {0} 的字符串"),
14  minlength: jQuery.validator.format("请输入一个长度最少是 {0} 的字符串"),
15  rangelength: jQuery.validator.format("请输入一个长度介于 {0} 和 {1} 之间
    的字符串"),
16  range: jQuery.validator.format("请输入一个介于 {0} 和 {1} 之间的值"),
17  max: jQuery.validator.format("请输入一个最大为 {0} 的值"),
18  min: jQuery.validator.format("请输入一个最小为 {0} 的值")
19  });
```

将上面的代码保存为一个 JavaScript 文件后和其他 jQuery 文件关联到我们的 HTML 中：

```
1   <script type="text/javascript" src="../../jslib/jquery.js"></script>
2   <script type="text/javascript" src="JS/jquery.validate.js"></script>
                                                    //验证框架文件
3   <script type="text/javascript" src="JS/jquery.metadata.js"></script>
                                                    .//元数据插件文件
4   <script type="text/javascript" src="JS/MyValidateMessage.js"></script>
                                                    //自定义的验证消息文件
```

最后，只需要加上简单的功能实现代码：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#form1").validate();//调用验证框架
4   });
5 </script>
```

效果如图 7.6 和图 7.7 所示。

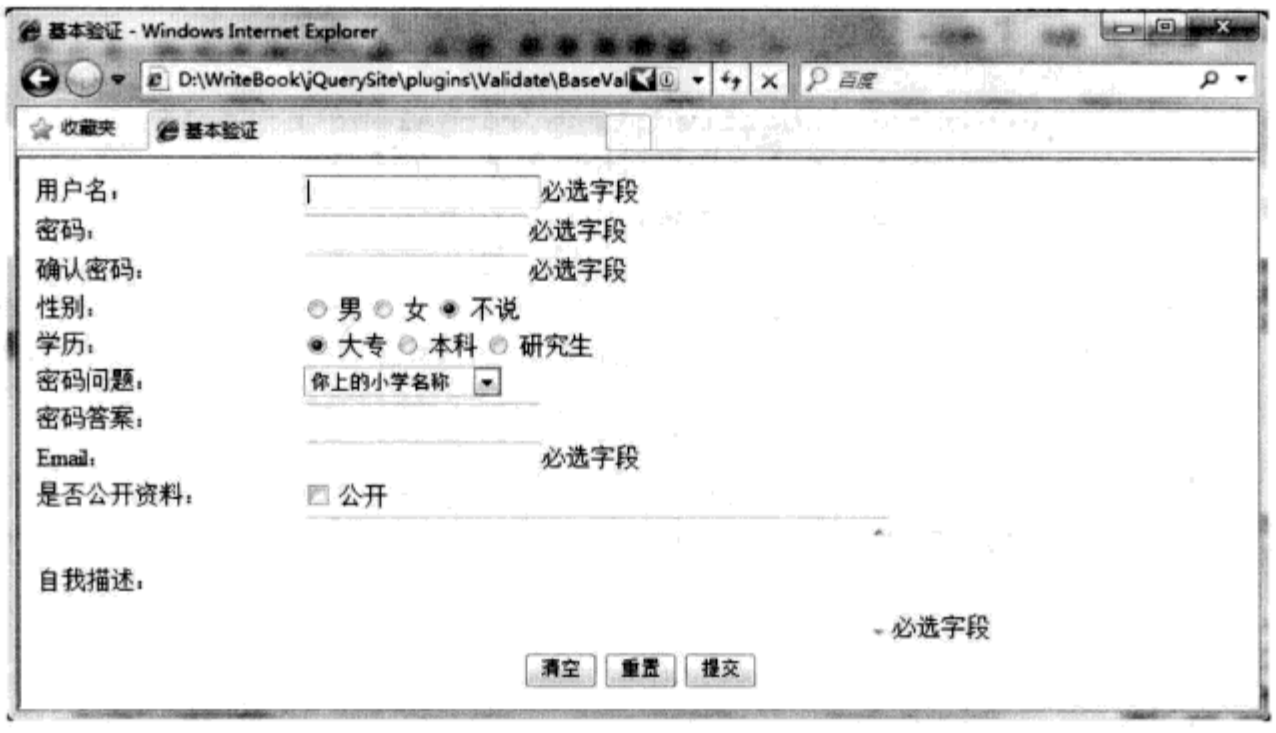


图 7.6　必填字段的验证

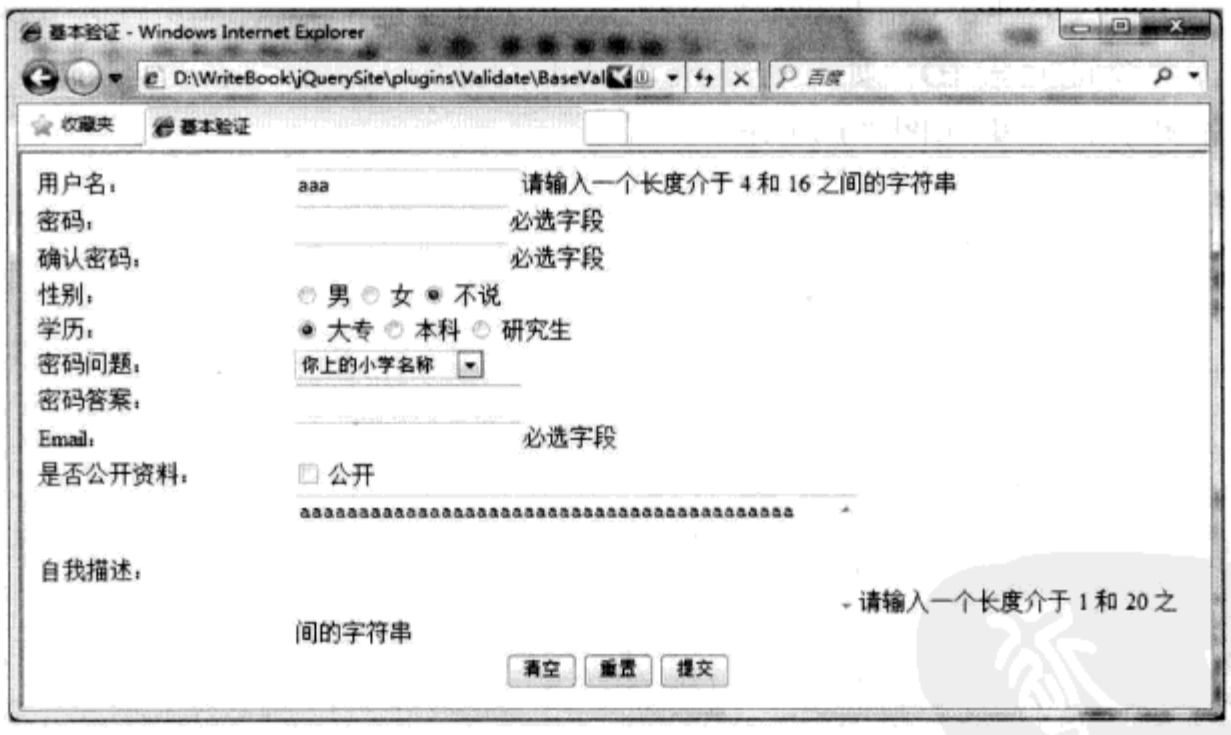


图 7.7　输入长度范围的验证

7.2.2　API 使用方法

本节主要介绍在使用验证框架时如何将校验规则组合写在代码中，以及几种常见方法及注意问题。

1．在代码中编写校验规则

在前面的那个基本验证的示例中，把验证规则写在了 HTML 标记中。下面介绍如何把校验规则写在 JavaScript 代码中。代码如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#form1").validate({
4           rules:{
5               username:{required:true,rangelength:[4,16]},
            //用户名字段为必填项，并限制长度为 4～16 个字符
6               pwd:{required:true,minlength:6},
            //密码字段为必填项，并限制长度最小为 6 个字符
7               confirmpwd:{required:true,minlength:6,equalTo:'#pwd'},
            //确认密码字段为必填项，并限制长度最小为 6 个字符，且要与密码字段内容相同
8               email:{email:true,required:true},
            //电子邮件字段必填并需符合电子邮件格式
9               desc:{required:true,rangelength:[1,20]}
            //自我描述字段必填并限制长度为 1～20 个字符
10          }
11      });
12  });
13 </script>
```

上述代码中只给出了规则指定部分，并没有给出验证消息文本。验证消息文本还沿用上面那个示例中的文件 MyValidateMessage.js。在规则设定部分（5～9 行）分成两个部分：第一部分是需要验证元素 ID，如 username、pwd、confirmpwd、email、desc。后面花括号中是规则指定部分，效果如图 7.8 所示。

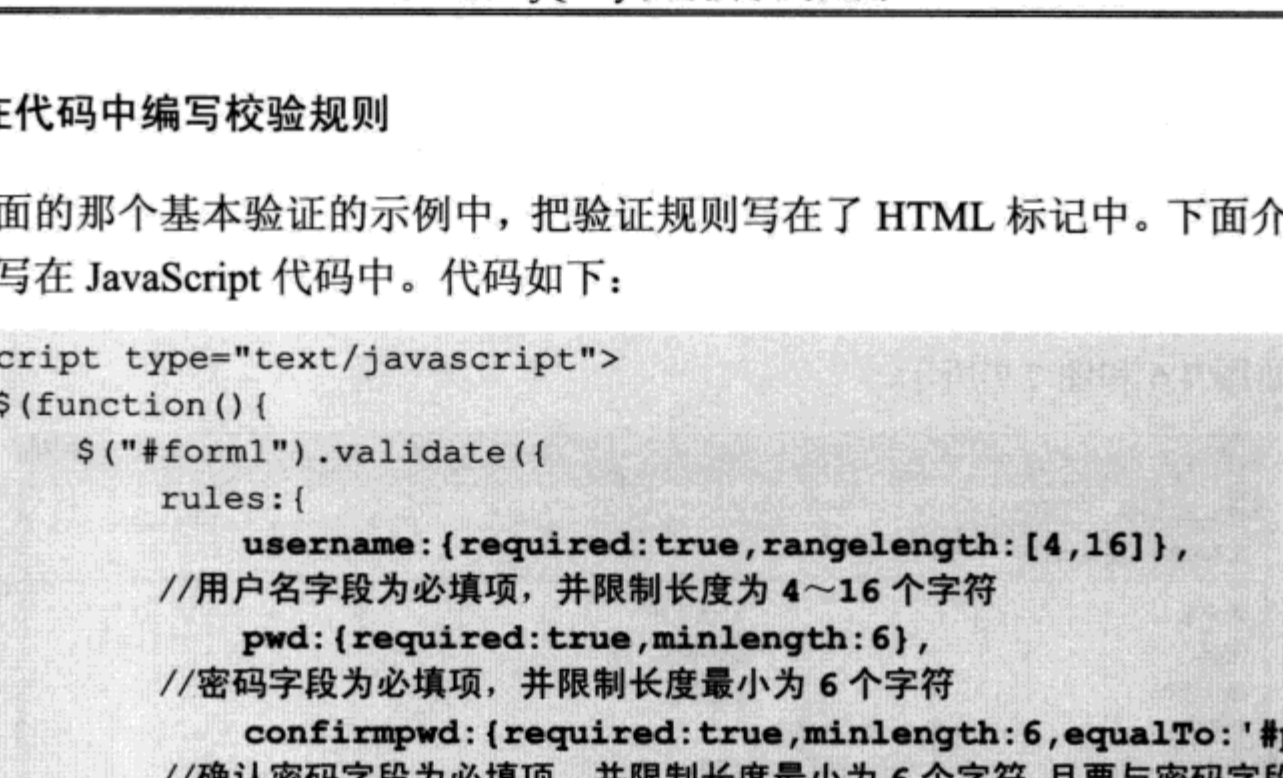


图 7.8　Validate 验证效果

如果想针对当前页面指定验证消息信息，可以在规则定义的下面添加对验证消息的指定。例如：

```
<script type="text/javascript">
  $(function(){
      $("#form1").validate({
          rules:{
              username:{required:true,rangelength:[4,16]},
              pwd:{required:true,minlength:6},
              confirmpwd:{required:true,minlength:6,equalTo:'#pwd'},
              email:{email:true,required:true},
              desc:{required:true,rangelength:[1,20]}
          },
          //下面的代码设定错误消息
          messages: {
              username:{required:"请输入用户名",rangelength:jQuery.validator.format("用户名长度介于 {0} 和{1}之间的字符串")},
              email: {
          required: "请输入 Email 地址",
          email: "请输入正确的 email 地址"
        },
        pwd: {
          required: "请输入密码",
          minlength: jQuery.format("密码不能小于{0}个字符")
        },
        confirmpwd: {
          required: "请输入确认密码",
          minlength: "确认密码不能小于 5 个字符",
          equalTo: "两次输入密码不一致"
          },
        desc:{required:"请输入自我描述",rangelength:jQuery.validator.format("自我描述长度介于 {0} 和 {1} 之间的字符串")}
          }
      });
  });
</script>
```

上述代码中 messages 部分就是我们为当前页面指定的验证消息。

2. 验证框架的提交操作

验证框架有自己的提交处理操作，可以在这个操作里添加自己定制的内容。例如如下代码：

```
  submitHandler:function(form){
      alert("submitted");
       form.submit();      //表单提交
      }
```

要实现上面这段代码的功能，需要在 HTML 中的“提交”按钮上不能出现值为“submit”的 ID 和 Name，并且触发提交必须是利用“提交”按钮而不应该是普通按钮。另外，在上述代码中不要写成$(form).submit()。实现效果如图 7.9 所示。

图 7.9　添加表单提交附加功能

3. 验证框架添加表单的调试效果

前台设计页面在使用验证框架的时候，需要进行验证框架调试。但是，验证框架的效果需要提交表单操作，这样可能会携带一些非法数据提交给服务器，对服务器处理程序造成不良影响。验证框架为了解决这种缺陷，特别提供了调试模式。在验证框架的调试模式中，可以检验验证框架的验证效果而不会提交表单。验证框架设定调试模式有两种方法：单一表单调试，多个表单调试。

单一表单调试设定代码如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#form1").validate({
4           debug:true  //表单调试功能设定
5       });
6   });
7 </script>
```

第 4 行代码设定 ID 为“form1”的表单，具有调试模式。效果和图 7.6 相同，但是，会发现页面无刷新，也就是没有真正的提交操作。如果页面中有多个表单，可以用另一种设定形式，代码如下：

```
1 <script type="text/javascript">
2   $(function(){
3       $.validator.setDefaults({
4           debug: true
5       });
6       $("#form1").validate({
7       });
8   });
9 </script>
```

第 3～5 行主要是设定多个表单的调试状态。

4. 表单元素忽视验证设定

当不是所有的表单元素都需要验证的时候，可以使用表单元素的忽视验证设定。把前面的对于用户名的验证设定改成如下内容：

```
required:true,rangelength:[4,16],ignore:".ignore"
```

再次对页面进行验证操作时会发现用户名这个部分不再有验证信息，效果如图 7.10 所示。

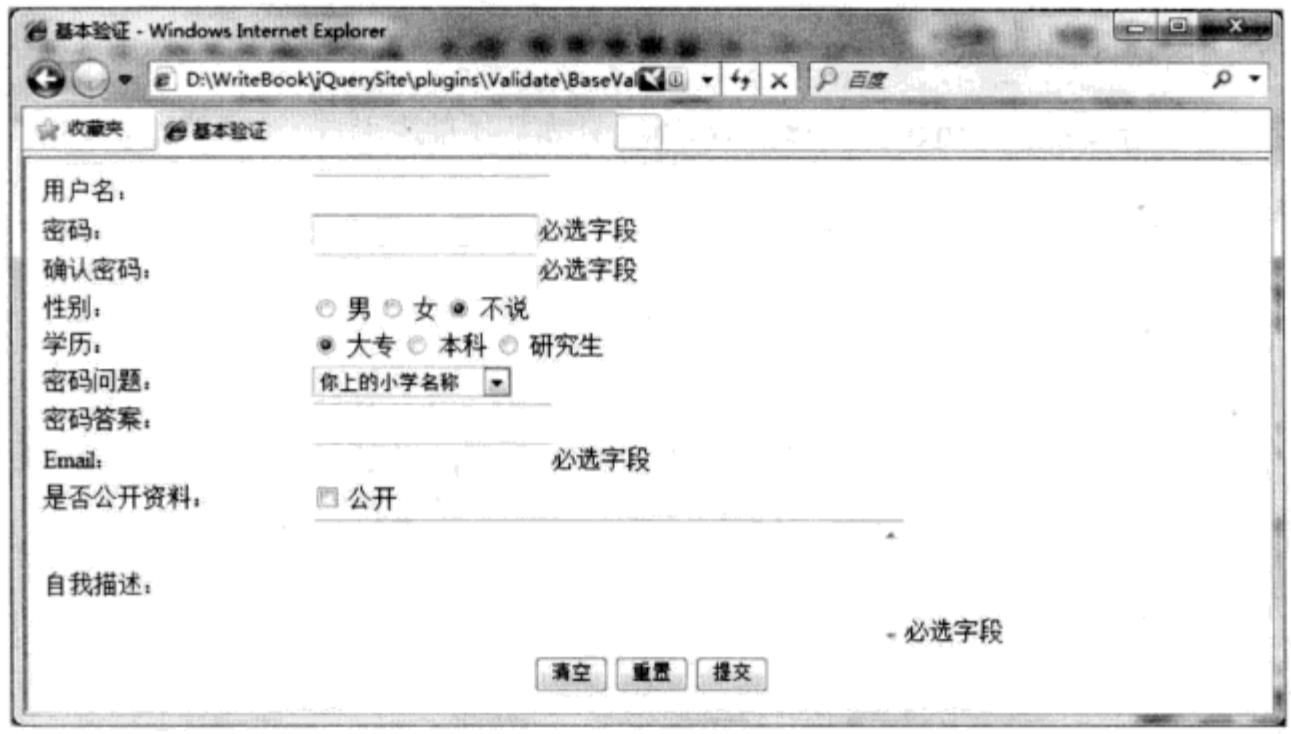


图 7.10　忽视表单元素验证

5. 验证信息的显示位置

前面的示例中验证的错误信息都是紧跟在表单旁边的。如果想要更改验证信息的显示位置，可以通过元素的验证信息位置属性设定显示位置。

首先，来看 errorPlacement 这个属性。它可以统一设定所有验证信息的显示位置。例如，在表单的下面加入一个 DIV，并设定 ID 为 summary。现在，需要将所有验证信息都显示在这个层中。JavaScript 代码如下：

```
<script type="text/javascript">
  $(function(){
      $.validator.setDefaults({
              debug: true
          });
      $("#form1").validate({
      errorPlacement: function(error, element) {
              //element.after(error);
              error.appendTo($("#summary"));      //设定验证信息的显示位置
      },
      rules:{
              username:{required:true,rangelength:[4,16]},
              pwd:{required:true,minlength:6},
```

```
14              confirmpwd:{required:true,minlength:6,equalTo:'#pwd'},
15              email:{email:true,required:true},
16              desc:{required:true,rangelength:[1,20]}
17          },
18          messages: {
19              username:{required:"请输入用户名<br/>",rangelength:jQuery.
                validator.format("用户名长度介于 {0} 和 {1} 之间的字符串"<br/>)},
20              email: {
21          required: "请输入 Email 地址<br/>",
22          email: "请输入正确的 email 地址<br/>"
23        },
24        pwd: {
25          required: "请输入密码<br/>",
26          minlength: jQuery.format("密码不能少于{0}个字符<br/>")
27        },
28        confirmpwd: {
29          required: "请输入确认密码<br/>",
30          minlength: "确认密码不能少于 5 个字符<br/>",
31          equalTo: "两次输入密码不一致<br/>"
32        },
33        desc:{required:"请输入自我描述<br/>",rangelength:jQuery.validator.
          format("自我描述长度介于 {0} 和 {1} 之间的字符串<br/>")}
34        }
35      });
36  });
37 </script>
```

上面代码的第 8 行是验证框架的默认信息显示位置。第 9 行是修改后的显示位置设定。为了美观，在每一个显示消息的最后都加上了回车换行标记，效果如图 7.11 所示。

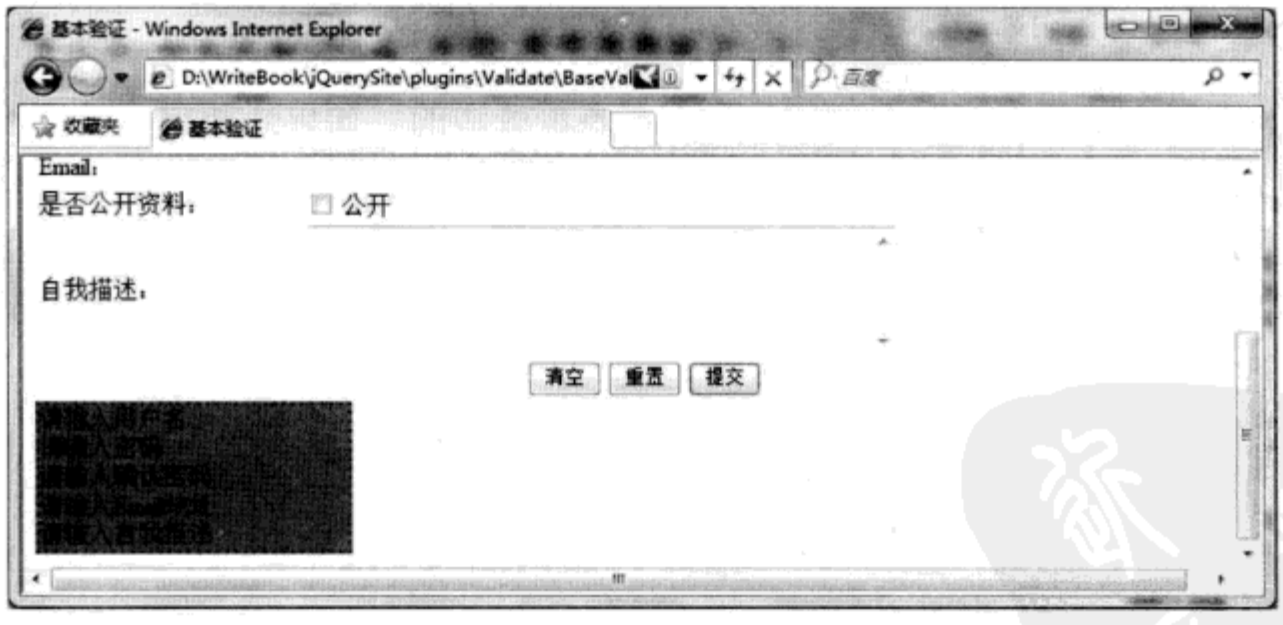


图 7.11　验证消息显示位置

除了上面介绍的 errorPlacement 属性可以设定验证消息的显示位置外，还有其他几个属性，功能如下。

（1）errorClass：指定错误提示的 CSS 类名，可以自定义错误提示的样式，默认为“error”。

（2）errorElement：用什么标签标记错误，默认为 label，可以改成 em。

（3）errorContainer：显示或者隐藏验证信息，可以自动实现有错误信息出现时把容器属性变为显示，无错误时隐藏。

（4）errorLabelContainer：把错误信息统一放在一个容器里面。

（5）wrapper：用什么标签把上边的 errorElement 包起来。

（6）success：要验证的元素通过验证后的动作，如果跟一个字符串，会当作一个 CSS 类，也可跟一个函数。

我们用下面两行代码替换上面代码的第 7～9 行。

```
errorLabelContainer: $("#summary"),
wrapper: "li",
```

得到的效果如图 7.12 所示，这里加入了 CSS 样式设定，具体 CSS 代码参考光盘内容。

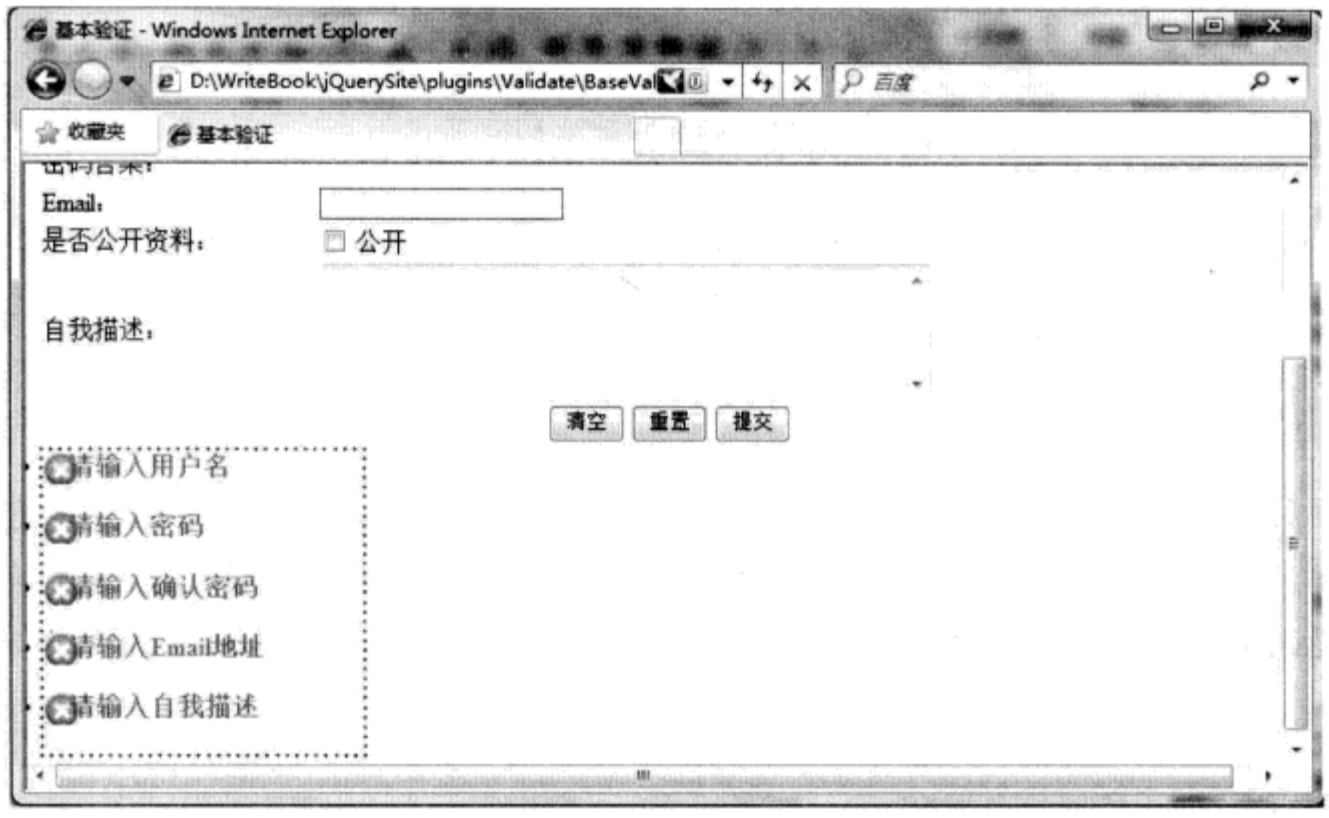


图 7.12　设定特殊样式的验证信息显示

如果只是错误信息显示页面，还不是很友好，所以这里再添加一个验证合格的样式设定。将下面这段代码加在验证信息指定位置的后面。

```
success: function(label) {
    label.html(" ").addClass("checked").text("OK!");
},
```

效果如图 7.13 所示。

7.2.3　自定义验证方法

Validate 框架的验证方法虽然很丰富，但是用户需要填写的内容信息种类更多。为了能够满足各种各样的验证，框架进行了用户扩展，允许我们自定义验证方法。自定义验证方法需要用到 addMethod 方法。我们用下面这段代码来说明如何使用这个方法注册自定义验证方法。

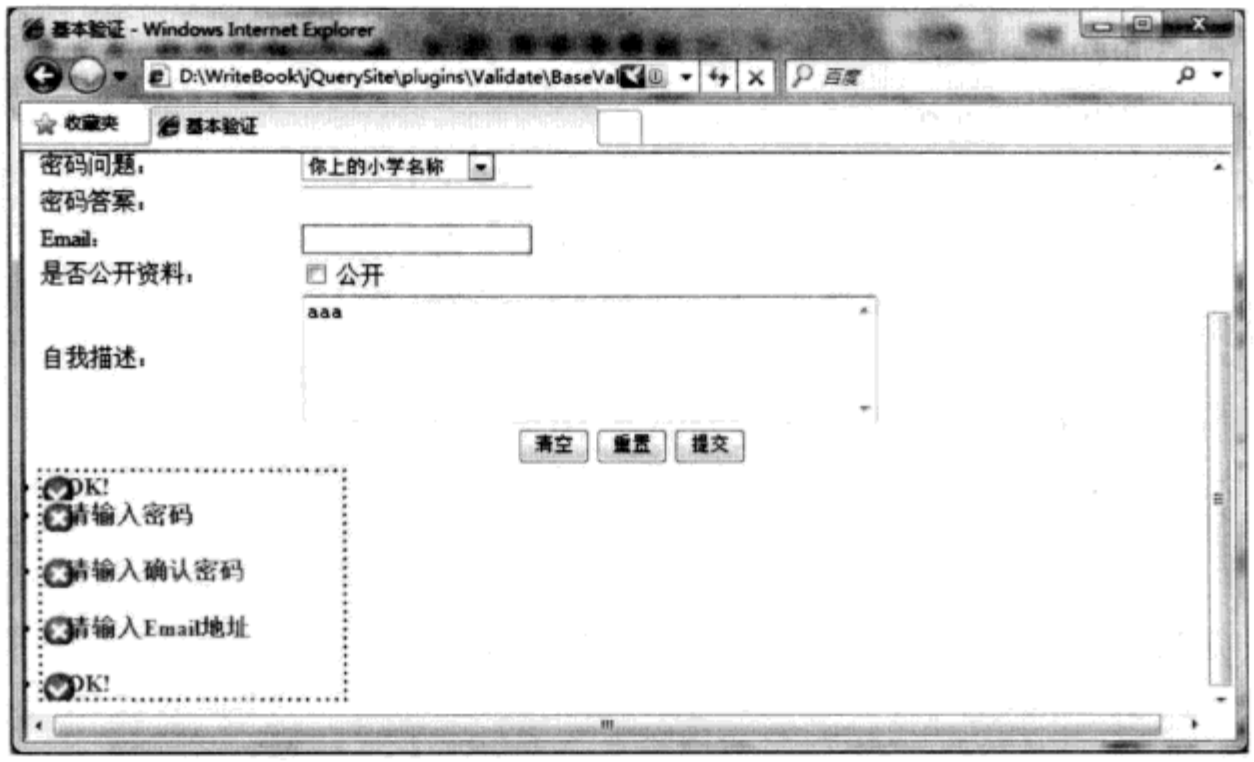


图 7.13　添加验证成功信息

```
1   jQuery.validator.addMethod("isZipCode", function(value, element) {
                                                          //自定义验证邮编
2       var tel = /^[0-9]{6}$/;
3       return this.optional(element) || (tel.test(value));
4   }, "请正确填写您的邮政编码");
```

这个自定义验证方法是比较简单的邮政编码验证。第 1 行调用 addMethod 方法，第 1 个参数是方法名称，第 2 个参数是验证操作，第 3 个参数是验证信息。这段代码可以独立成一个 JavaScript 文件，也可以写在 ready()函数的前面来使用。在 rules 中加上下面的代码：

```
zip:{required:true,isZipCode:true},
```

效果如图 7.14 所示。

图 7.14　自定义邮政编码验证

7.2.4　radio、checkbox、select 的验证

1．radio的验证

对 radio 的验证，只适用 required 的验证功能，表示必须选中其中之一。因为在 rules 中判定对哪个元素施加验证是通过元素的 name 属性实现的，而 radio 的 name 属性恰恰表示一组 radio 元素。所以，可以看成对一组同组的 radio 元素施加验证。例如，对性别选择施加验证，在 rules 中添加如下代码：

```
sex:{required:true}
```

在 messages 中添加如下代码：

```
sex:{required:"请选择性别"}
```

效果如图 7.15 所示。

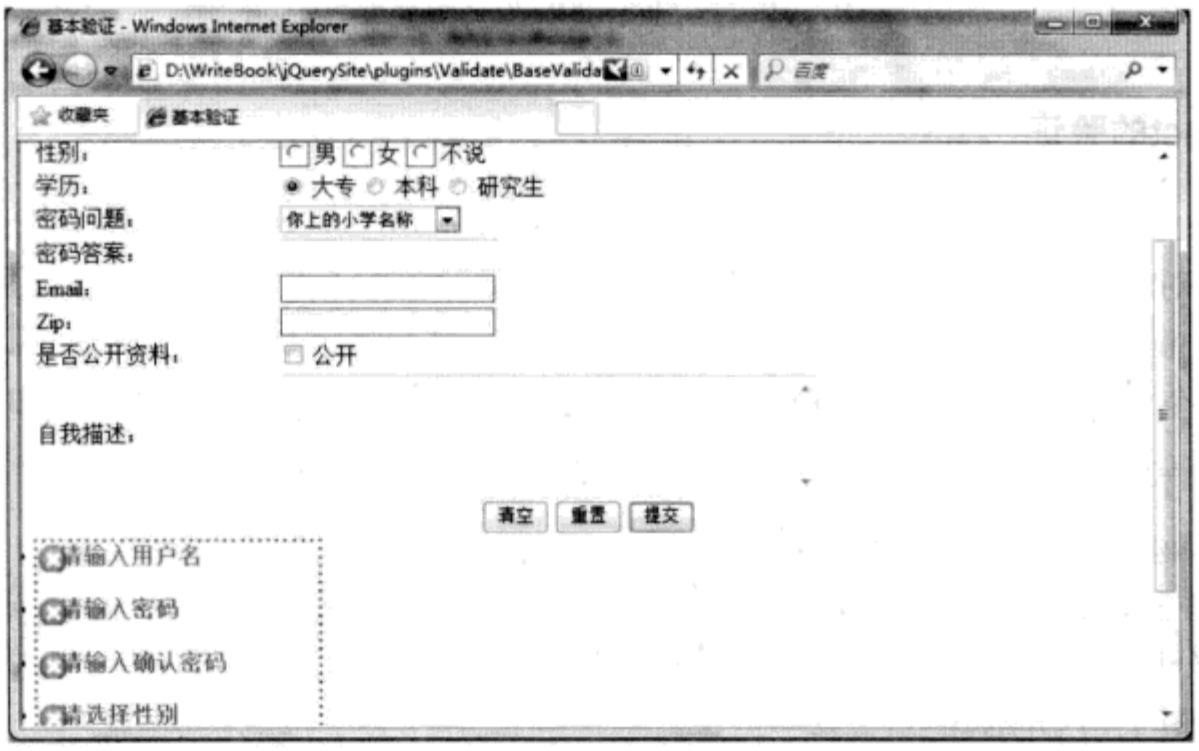


图 7.15　对 radio 元素施加验证

2．checkbox的验证

对 checkbox 的验证可以使用 required、maxlength、minlength、rangelength 这 4 个验证规则。其中，required 表示必选验证，maxlength 表示同组最多选择几个，minlength 表示同组最少选择几个，rangelength 表示同组中可选个数的范围。

在原有表单中加入一个爱好选择，所有的 checkbox 的 name 属性为 fun，表示同一组。HTML 代码参考光盘内容。然后，对爱好里面的 checkbox 施加验证。这里使用 required 和 rangelength 两个验证规则。在 rules 中添加如下代码：

```
fun:{required:true,rangelength:[2,3]}
```

在 messages 中添加验证消息设定：

```
fun:{required:"请选择爱好",rangelength:jQuery.validator.format("至少选择{0}
个爱好,至多选择{1}个爱好<br/>")}
```

效果如图 7.16 所示。

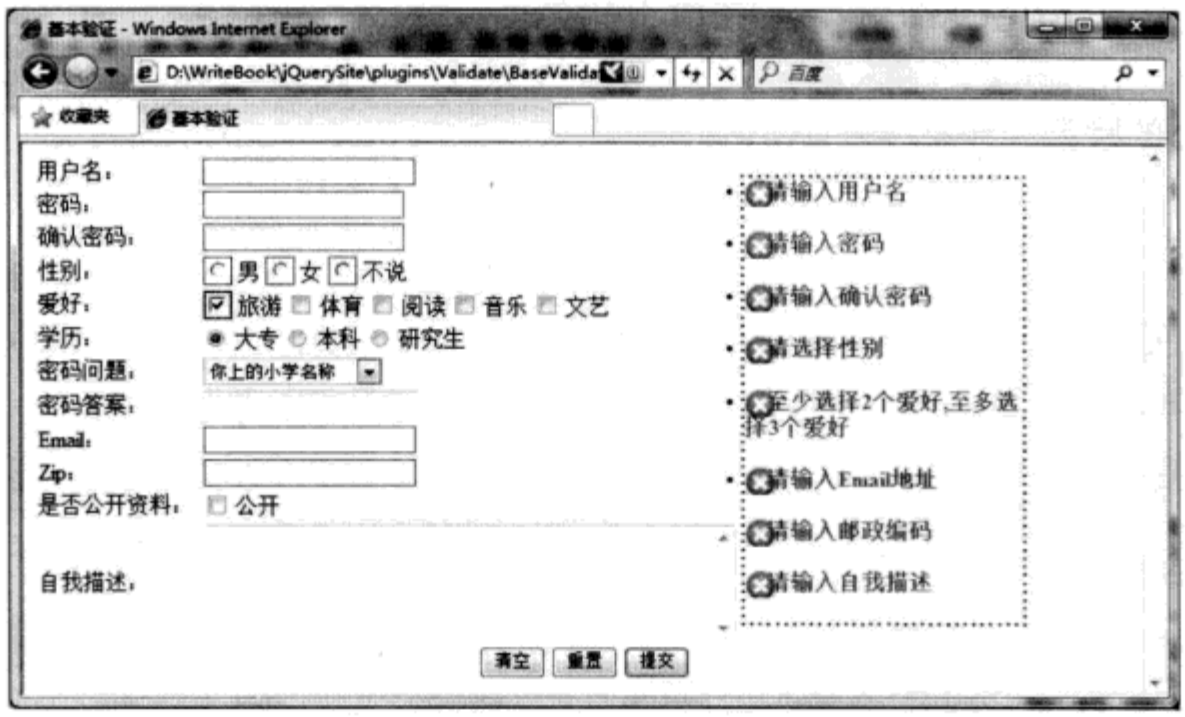


图 7.16　对 checkbox 元素施加验证

3．select的验证

对 select 元素的验证和 checkbox 类似。因为 select 的选择有两种形式，当 select 为单选形式时，对其只能使用 required 验证。当 select 为多选形式时，可以使用 required、maxlength、minlength、rangelength 这 4 个验证规则。在这个表单中加上一个可以多选的 select 元素，然后在这个元素上施加 required 和 rangelength 验证。在 rules 中添加如下代码：

```
city:{required:true,rangelength:[2,3]}
```

在 messages 中添加验证消息设定：

```
city:{required:"请选择城市",rangelength:jQuery.validator.format("至少选择{0}个城市,至多选择{1}个城市<br/>")}
```

效果如图 7.17 所示。

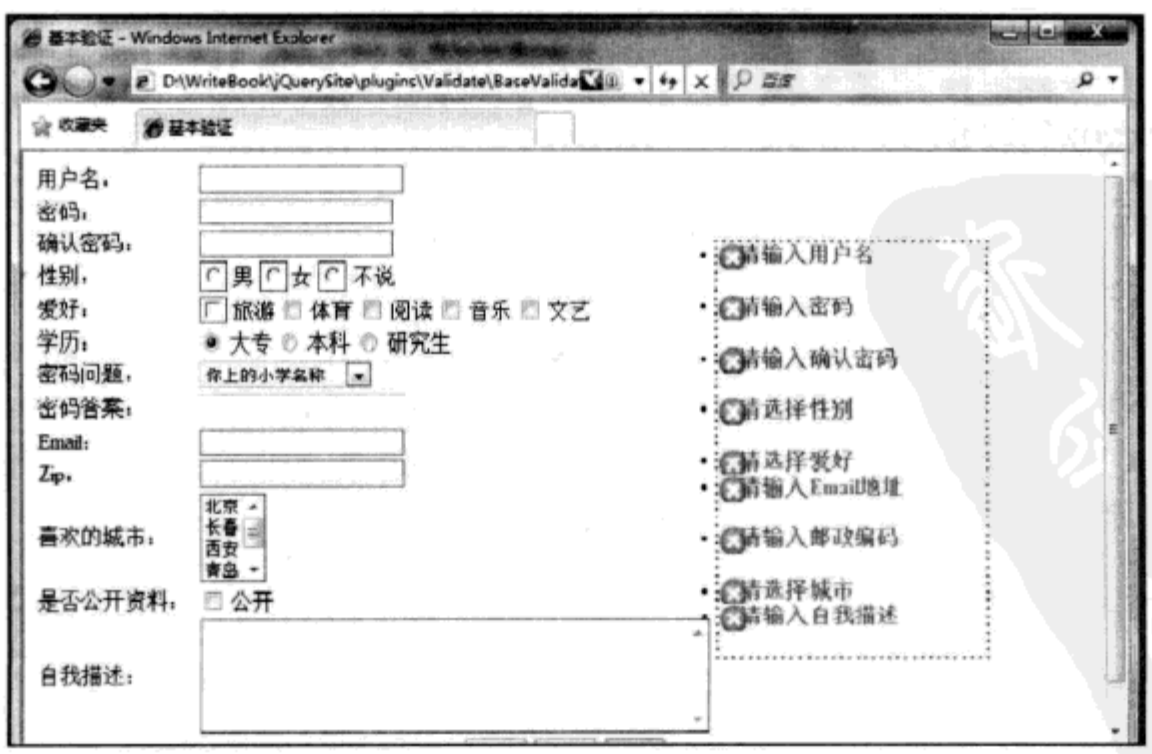


图 7.17　对 select 元素施加验证

7.3　表 单 特 效

常用的表单元素包括文本输入框，单选按钮、复选框、按钮等。jQuery UI 提供的特效插件很多，可以直接使用。本节主要通过几个示例说明 jQuery 对这些表单元素使用特效的基本原理。

7.3.1　文本输入框特效

这里针对文本输入框讲解两个特效：获取焦点后文本框内部样式改变，利用文本框模拟 select 效果。对于 GOOGLE 自动提示的特效是需要后台服务器支持的，本书主要讲解前台设计部分，所以，不对 GOOGLE 自动提示的特效进行讲解。

1．获取焦点后文本框内部样式改变

这种特效和前面讲解的特效的实现原理类似，通过获取文本框的焦点获取状态，修改文本框的样式设定，效果如图 7.18 和图 7.19 所示。

图 7.18　文本框失去焦点样式

图 7.19　文本框得到焦点样式

其中将用到 jQuery 函数 ready()、focus()、removeClass()、addClass()、blur()、select()。部分函数说明如表 7.4 所示。

表 7.4　函 数 说 明

函数名	语 法 形 式	使 用 说 明
focus()	语法形式一：focus() 语法形式二：focus(fn)	第一种语法形式触发匹配元素的得到焦点事件；第二种语法形式在匹配元素的得到焦点事件中绑定一个处理函数
blur()	语法形式一：blur() 语法形式二：blur(fn)	第一种语法形式触发匹配元素的失去焦点事件；第二种语法形式在匹配元素的失去焦点事件中绑定一个处理函数
select()	语法形式一：select() 语法形式二：select(fn)	第一种语法形式触发匹配元素的选中事件；第二种语法形式在匹配元素的选中事件中绑定一个处理函数

2. 功能实现

文本框内部样式改变实现步骤如下。

（1）设定文本框的得到焦点事件与失去焦点事件。

（2）在两个事件中修改文本框样式。

（3）在两个事件中修改文本框的值。

具体 HTML 代码可以参考光盘内容。JavaScript 代码如下。

```
<script type="text/javascript">
    $(document).ready(function() {
        $('input[type="text"]').addClass("blurField");
                                                    //设定文本框加载后状态
        $('input[type="text"]').focus(function() {      //文本框得到焦点
            $(this).removeClass("blurField").addClass("focusField");
                                                    //修改样式
            if (this.value == this.defaultValue){       //设定内容值
            this.value = '';
            }
            if(this.value != this.defaultValue){
            this.select();  //选定文本框
            }
        });
        $('input[type="text"]').blur(function() {       //文本框失去焦点
            $(this).removeClass("focusField").addClass("blurField");
                                                    //修改样式
            if ($.trim(this.value) == ''){              //设定内容值
            this.value = (this.defaultValue ? this.defaultValue : '');
            }
        });
    });
</script>
```

在上面的代码中 defaultValue 代表文本框的默认值，也就是定义文本框时赋给 value 属性的值。

3. 文本框模拟select效果

下面讲解如何利用文本框模拟 select 效果。其原理是利用层的显示与隐藏，将按钮层加载到文本框的右侧来控制下拉内容显示，将内容层加载到文本框的下面实现下拉效果。

其中，将用到 jQuery 函数 ready()、css()、position()、width()、height()、after()、hide()、attr()、append()、mouseover()、mouseout()、mousedown()、mouseup()、show()、get()、val()、html()，效果如图 7.20 和图 7.21 所示。

图 7.20　下拉列表框加载

图 7.21　下拉列表框展开

4．功能实现

文本框模拟下拉列表框的实现步骤如下。

（1）在文本框右边添加图片层，模拟按钮。

（2）在文本框下方添加内容层，模拟下拉内容，并隐藏。

（3）为图片层添加鼠标的悬停、离开、按下、抬起事件，在事件修改内容层的显示与隐藏，更改图片样式。

（4）为内容层添加鼠标悬停事件，更改项目层的样式。

（5）为整个页面添加鼠标抬起事件，更改文本框内容。

具体实现功能的 JavaScript 代码如下：

```
<script type="text/javascript" language="JavaScript">
  $(document).ready(function(){
    //定义一个图片层，并配置样式（位置，定位点坐标，大小，背景图片），追加到文本框后面
      $DIV = $('<div></div>').css('position', 'absolute').css('left', $('#content').position().left + $('#content').width() - 10 + 'px').css('top', $('#content').position().top + 2 + 'px').css('background', 'transparent url(img/choice.gif) no-repeat top left').css('height', '16px').css('width', '15px');
      $('#content').after($DIV);
    //定义一个内容层，并配置样式（位置，定位点坐标，宽度），先将其隐藏
      $SELECT = $('<div></div>').css('position', 'absolute').css('border', '1px solid #000000').css('left', $('#content').position().left + 'px').css('top', $('#content').position().top + $('#content').height() + 5 + 'px').css('width', $('#content').width()-12 + 'px');
      $('#content').after($SELECT);
      $SELECT.hide();
    //定义 5 个项目层，并配置样式（宽度），添加 name、value 属性，加入内容层
```

```
8           for (var i = 0; i <= 5; i++) {
9               $OPTION = $('<div class="item">option' + i + '</div>').attr
                ('name', 'option').attr('value', 'value' + i).css('width',
                $SELECT.width());
10              $SELECT.append($OPTION);
11          };
            //内容层的鼠标移入移出样式
12          $SELECT.mouseover(function(event){
13              if ($(event.target).attr('name') == 'option') {
                    //移入时背景色变深，字色变灰
14                  $(event.target).css('background-color', '#000077').
                    css('color', 'gray');
15              }
16          });
            //项目层鼠标移出事件
17          $(".item").each(function(){
18              $(this).mouseout(function(){
19                  $(this).css('background-color', 'white').css('color',
                    '#000000');
20              });
21          });
            //鼠标进入修改背景图位置
22          $DIV.mouseover(function(){
23              $DIV.css('background-position', ' 0% -16px');
24          });
            //鼠标移出修改背景图位置
25          $DIV.mouseout(function(){
26              $DIV.css('background-position', ' 0% -0px');
27          });
            //鼠标按下修改背景图位置
28          $DIV.mousedown(function(){
29              $DIV.css('background-position', ' 0% -32px');
30          });
            //鼠标释放修改背景图位置
31          $DIV.mouseup(function(){
32              $DIV.css('background-position', ' 0% -16px');
33              $SELECT.show();
34          });
            //通过单击位置，判断弹出的显示
35          $(document).mouseup(function(event){
             //如果是图片层或内容层，则依然显示内容层
36              if (event.target == $SELECT.get(0) || event.target == $DIV.
                get(0)) {
37                  $SELECT.show();
38              }
39              else {
40                  //如果是项目层，则改变文本框的值
41                  if ($(event.target).attr('name') == 'option') {
42                      $('#content').val($(event.target).html());
43                  }
                    //如果是其他位置，则将内容层隐藏
44                  if ($SELECT.css('display') == 'block') {
```

```
                    $SELECT.hide();
                }
            }
        });
    });
</script>
```

7.3.2　单选按钮、复选框特效

单选按钮和复选框特效原理也是基于鼠标的单击事件来实现效果，并在单击事件中修改样式，在单击前后替换不同的图片。其中，将要用到 jQuery 函数 ready()、addClass()、removeClass()、children()、attr()、siblings()。其功能实现如下。

（1）设定单选按钮、复选框的单击事件。

（2）在单击事件中更改 CSS 样式，更换图片。

实现功能的 JavaScript 代码如下：

```
<script type="text/javascript">
    $( function () {
        //单选按钮
        $(".fr").click(
            function () {
                $(this).addClass("checked").removeClass("unchecked").
                siblings(".checked").removeClass("checked").addClass
                ("unchecked");                    //更改单选按钮被单击时样式
            }
        );
        //复选框
        $(".fck").toggle(
            function () {
                $(this).addClass("right");
                $(this).children("input").attr("right", "checked");
            },
            function () {
                $(this).removeClass("right");
                $(this).children("input").removeAttr("checked");
            }
        );
    });
</script>
```

上述代码通过更改不同的 CSS 样式类，实现了按钮在不同状态下的图片效果。具体 CSS 代码和 HTML 代码参考光盘内容。效果如图 7.22 所示。

7.3.3　按钮特效

本节将做一个图片在按钮上出现并能够跳跃，实现动感的效果。它的实现原理是通过在 jQuery 中动态修改图片的位置，并利用 jQuery 的动画效果在修改后的位置上产生跳跃

的动感。其中，将要用到 jQuery 函数 ready()、css()、position()、hover()、animate()。其功能实现如下。

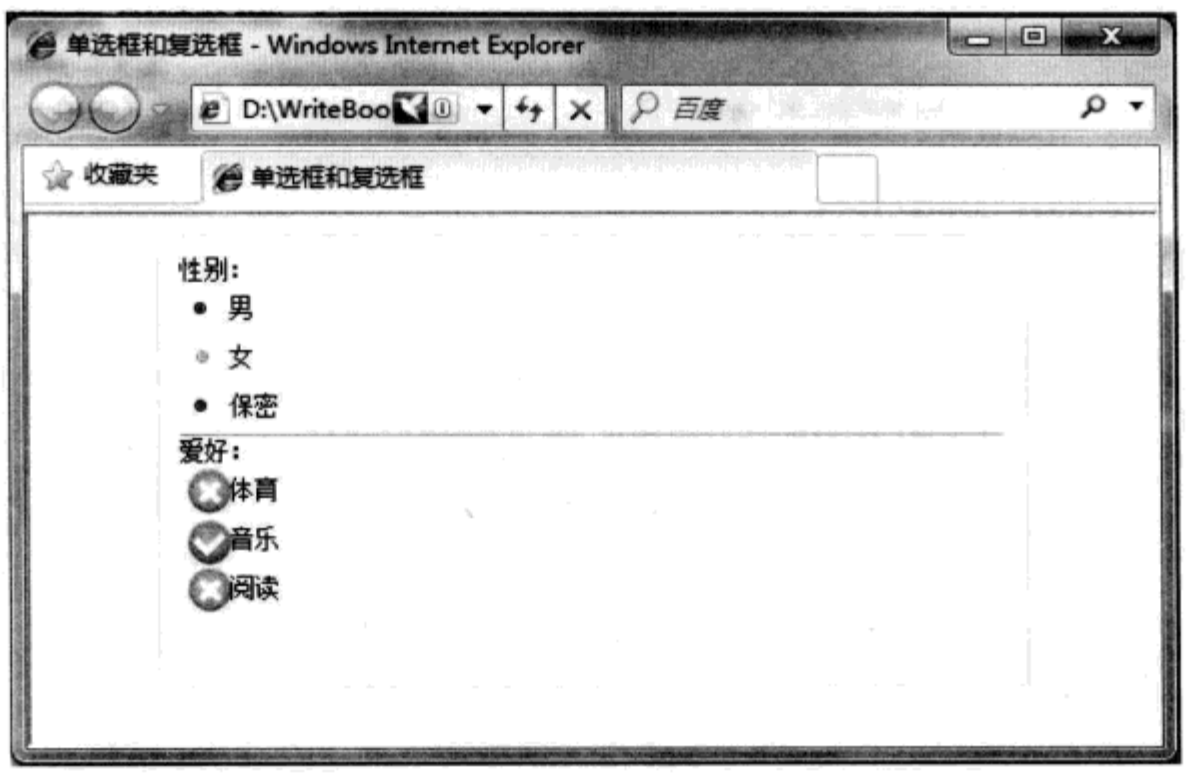


图 7.22　单选按钮、复选框特效

（1）通过 css()函数修改图片位置，将图片覆盖在按钮的左边。

（2）设定按钮的鼠标悬停与离开事件。

（3）在事件中利用 animate()实现图片的跳跃动画效果。

实现功能的 JavaScript 代码如下：

```
<script type="text/javascript">
    $(document).ready(function(){
    $("img").css('position','absolute').css('top',$(".button").posit-
     ion().top-4+"px").css('left',$(".button").position().left-12+
     "px").css('border',"none");          //设定图片相对于按钮的位置
        var topvalue=$(".button").position().top;
                                          //取得按钮跳跃的水平相对位置
        $(".button").hover(function(){  //鼠标悬停与离开时图片动画跳跃 4 次
                $("img")
                .animate({top:topvalue-11+"px"}, 200).animate({top:
                topvalue-4+"px"}, 200)
                .animate({top:topvalue-9+"px"}, 200).animate({top:
                topvalue-4+"px"}, 200)
                .animate({top:topvalue-7+"px"}, 100).animate({top:
                topvalue-4+"px"}, 100)
                .animate({top:topvalue-6+"px"}, 100).animate({top:
                topvalue-4+"px"}, 100);
        });
    });
</script>
```

静态效果如图 7.23 所示，动态的跳跃效果请读者自己测试。

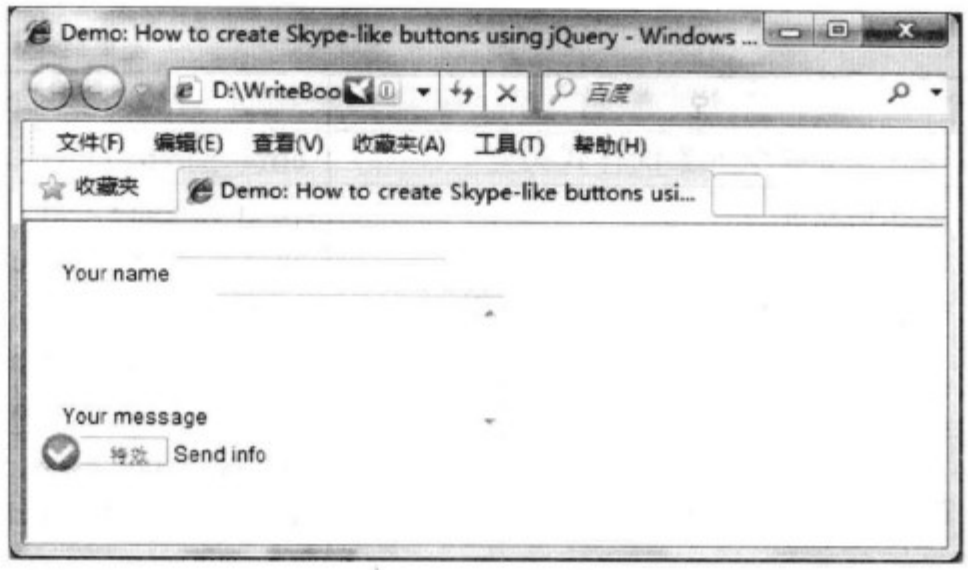


图 7.23　带跳跃图片的按钮静态效果

7.4　表 单 插 件

7.2 节介绍了 Validate 校验框架。其实，校验框架也是 jQuery 的插件之一。只不过它的侧重点不在美化效果上而是在数据的检查上。本节将介绍两个表单插件，主要实现表单的美化效果。当然，它们也有一定的校验功能。

7.4.1　Validation 插件

这里要介绍的 Validation 插件是一个验证插件。它的英文介绍网址为 https://github.com/ablomen/Validation。这个插件的特点如下。

（1）易用。

（2）易扩展。

（3）自定义验证规则和输出信息。

（4）使用样式指定用户行为。

（5）使用 jQuery 1.3.2 和 1.4 框架。

（6）可以加入 Ajax 功能，对表单内外的元素都可以验证。

这个插件的核心方法是 validate()。目前，它具有的验证属性如表 7.5 所示。

表 7.5　Validation验证属性

属 性 名 称	说　　明	默　认　值
required_class	这个验证插件的必填元素验证的样式类	validate_required
error_attribute	错误信息属性	title
error_output	每个字段的默认输出类型	{text:"class", email:"class", select:"class", textarea:"class", other:"append"}
error_class	不合法元素的样式类	validate_invalid
error_append_char	当不合法元素出现时，在其后出现的标识符	*
error_append_class	修饰不合法标识符的样式类型	error_append_class

续表

属 性 名 称	说　　明	默　认　值
error_parent_class	不合法元素父元素的样式类型	error_parent_class
types	待验证元素验证后返回类型列表	{text:validate_text, email:validate_email, checkbox:validate_checkbox, radio:validate_radio, select:validate_select, textarea:validate_text}
output	输出功能实现模块	{class:output_class, inline:output_inline, append:output_append, alert:output_alert, parent:output_parent}

下面用几个示例来说明如何使用这个插件。

1. 基本验证

下面的示例使用了这个验证框架的所有默认值，简单调用默认框架的核心函数进行验证。

首先，创建一个表单，HTML 代码如下：

```
<form id="form1" method="post" action="">
<table>
  <tr>
      <th>Normal input</th>
      <td><input name="foo" type="text" class="validate_required"/></td>
  </tr>
  <tr>
      <th>Normal select</th>
      <td>
          <select name="foo" class="validate_required">
              <option value="">Please select something</option>
              <option>Something</option>
          </select>
      </td>
</tr>
<tr>
      <th>Normal checkbox</th>
      <td><input name="foo" type="checkbox" class="validate_required"/>
      please check me</td>
</tr>
<tr>
      <th>Normal radiobox</th>
      <td>
          <input name="foo" type="radio" value="foo" class="validate_
          required"/> check me
          <input name="foo" type="radio" value="bar" class="validate_
          required"/> or me
      </td>
</tr>
<tr>
```

```
28      <th>Normal textarea</th>
29      <td><textarea name="foo" class="validate_required"></textarea></td>
30  </tr>
31  <tr>
32      <td></td>
33      <td><input type="submit" value="Submit"/></td>
34  </tr>
35</table>
36</form>
```

在上面的代码中我们看到需要验证的元素都加上验证样式类。其中，第 5、10、18、23、24、29 行都使用了 validate_required 这个样式类。再来看一下它的 CSS 样式设定内容：

```
1 <style type="text/css">
2   th, td{
3       padding:        5px;
4   }
5   input, textarea, select{
6       border:          1px solid #000;
7   }
8   .validate_required{
9       border-left:    3px solid red;
10  }
11  .validate_invalid{
12      border-left:    3px solid red;
13      border-color:   red;
14      color:          red;
15  }
16  .validate_invalid_append{
17      color:          red;
18  }
19  .validate_invalid_parent{
20      background:     red;
21  }
22 </style>
```

其中，第 2～7 行是表单元素的基本样式设定；第 8～10 行是表示必填验证元素的样式设定；第 11～15 行是验证操作后不合法元素的样式设定；第 16～18 行是添加的非法元素标识符的样式设定；第 19～21 行是不合法元素父元素的样式设定。

对上述表单进行验证前需要先引入 jQuery 库和插件文件：

```
<script type="text/javascript" src="../../jslib/jquery-1.4.min.js"></script>
<script type="text/javascript" src="JS/validation.js"></script>
```

验证的功能代码：

```
1 <script type="text/javascript">
2   $(function(){
3       $("#form1").validate();
4   });
5 </script>
```

代码相对简单，容易掌握，不需要我们进行任何属性设定。效果如图 7.24 所示。

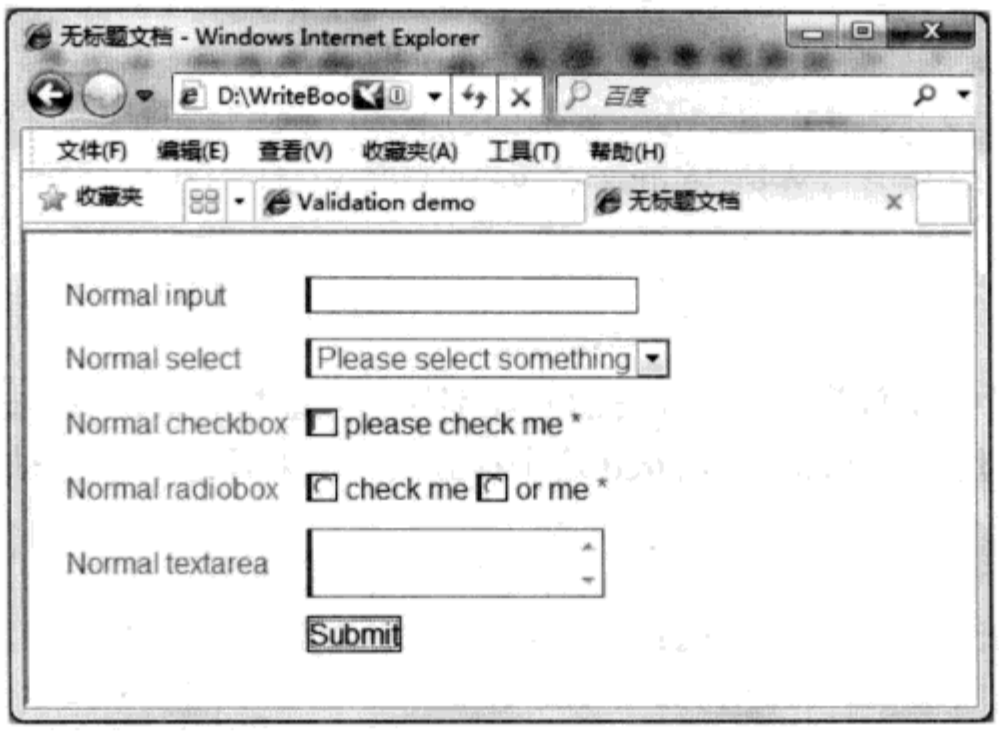


图 7.24　Validation 基本验证效果

2. 改变验证信息输出

前面已看到了基本验证效果。下面看一下如何改变验证消息的输出形式。可以对非法元素更改不同的样式设定，同时可以按提示信息的形式输出验证消息。

样式设定还是沿用上面的代码，但是需要新建一个表单：

```
1 <form id="form2" method="post" action=" ">
2 <table>
3   <tr>
4       <th>Class error</th>
5       <td><input name="foo" type="text" class="validate_required validate_
        error_class"/></td>
6   </tr>
7   <tr>
8       <th>Inline error</th>
9       <td><input name="foo" type="text" class="validate_required validate_
        error_inline" title="Please enter some text"/></td>
10  </tr>
11  <tr>
12      <th>Append error</th>
13      <td><input name="foo" type="text" class="validate_required validate_
        error_append"/></td>
14  </tr>
15  <tr>
16      <th>Parent error</th>
17      <td><input name="foo" type="text" class="validate_required validate_
        error_parent" title="Please enter some text"/></td>
18  </tr>
19  <tr>
20      <th>Alert error</th>
21      <td><input name="foo" type="text" class="validate_required validate_
```

```
        error_alert" title="Please enter some text"/></td>
22  </tr>
23  <tr>
24      <td></td>
25      <td><input type="submit" value="Submit"/></td>
26  </tr>
27</table>
28</form>
```

上面的代码中第 5、9、13、17、21 行都对验证元素指定了验证样式和不同于上面例子中的错误输出样式。调用验证功能的 JavaScript 代码同上例一样，效果如图 7.25 所示。

图 7.25　改变验证消息输出效果

3. 通过JavaScript改变验证消息输出

刚才的示例是通过在 HTML 中添加不同的 CSS 类改变验证消息输出。下面将这个功能加在 JavaScript 代码中。使用 HTML 新建一个表单，如下所示：

```
1 <form id="form3" method="post" action="./demo.html">
2 <table>
3   <tr>
4       <th>Text</th>
5       <td><input name="foo" type="text" class="validate_required"/></td>
6   </tr>
7   <tr>
8       <th>Select</th>
9       <td>
10          <select name="foo" class="validate_required">
11              <option value="">Please select something</option>
12              <option>Something</option>
13          </select>
14      </td>
15  </tr>
16  <tr>
```

```
17      <th>Checkbox</th>
18      <td><input name="foo" type="checkbox" class="validate_required"
        title="please check me"/> please check me</td>
19  </tr>
20  <tr>
21      <th>Textarea</th>
22      <td><textarea name="foo" class="validate_required" title="please
        enter text in the textarea"></textarea></td>
23  </tr>
24  <tr>
25      <td></td>
26      <td><input type="submit" value="Submit"/></td>
27  </tr>
28</table>
29</form>
```

在对这个表单的验证消息输出进行设定时，JavaScript 代码如下：

```
1   $("#form3").validate({
2       error_output:   {
3       text:           "inline",
4       select:         "append",
5       other:          "parent",
6       textarea:       "alert"
7       }
8   });
```

这段代码中，第 2 行表示要对验证消息输出属性进行设定。第 3 行表示对单行文本框的验证消息输出使用内联样式，也就是 validate_required 类所设定的样式。第 4 行表示对下拉列表框的验证消息中的标识符进行样式设定。第 5 行表示对非单行文本框、非下拉列表框、非文本区域的其他元素的验证消息输出的样式设定。第 6 行表示对文本框的验证消息设定，弹出 JavaScript 信息提示框，效果如图 7.26 所示。

图 7.26　通过 JavaScript 代码改变验证消息输出

4. 添加自定义验证功能

Validation 这个框架还为我们提供了自定义验证功能。使用这个功能可以指定验证消息的输出形式、验证规则如何实现、验证功能返回值类型等。

首先，创建一个简单的表单：

```
1 <form id="form4" method="post" action="./demo.html">
2 <table>
3   <tr>
4       <th>Please enter one's and zero's only</th>
5       <td><input name="foo" type="text" class="validate_required validate_
        type_binary"/></td>
6   </tr>
7   <tr>
8       <td></td>
9       <td><input type="submit" value="Submit"/></td>
10  </tr>
11 </table>
12 </form>
```

在第 5 行中，除了指定这个元素是必填字段外还指定了二进制验证，即布尔值验证。在 JavaScript 代码中需要为这种验证类型加上消息输出样式，加上验证功能的实现。验证功能实现后返回结果应为布尔类型表示验证是否通过。具体代码如下：

```
1  $("#form4").validate({
2      // Define the type of validation to use
3      error_output:  {
4          binary:          "append"
5      },
6      // Define the validation function
7      types:           {
8          binary:     function (element, object) {
9              var valid   =   true,
10                 pattern     =   /[^0|1]/;
11
12             if (pattern.exec(element.el.val()) || element.el.val().
               length < 1) {
13                 valid       =   false;
14             }
15
16             return valid;
17         }
18     }
19 });
```

第 3～5 行设定验证消息输出形式为标识符标志验证结果。第 7～18 行是自定义验证功能，验证返回类型为布尔，验证文本框的输入内容使用了正则表达式进行匹配，只能输入 0 或者 1，如果输入错误则返回假，效果如图 7.27 所示。

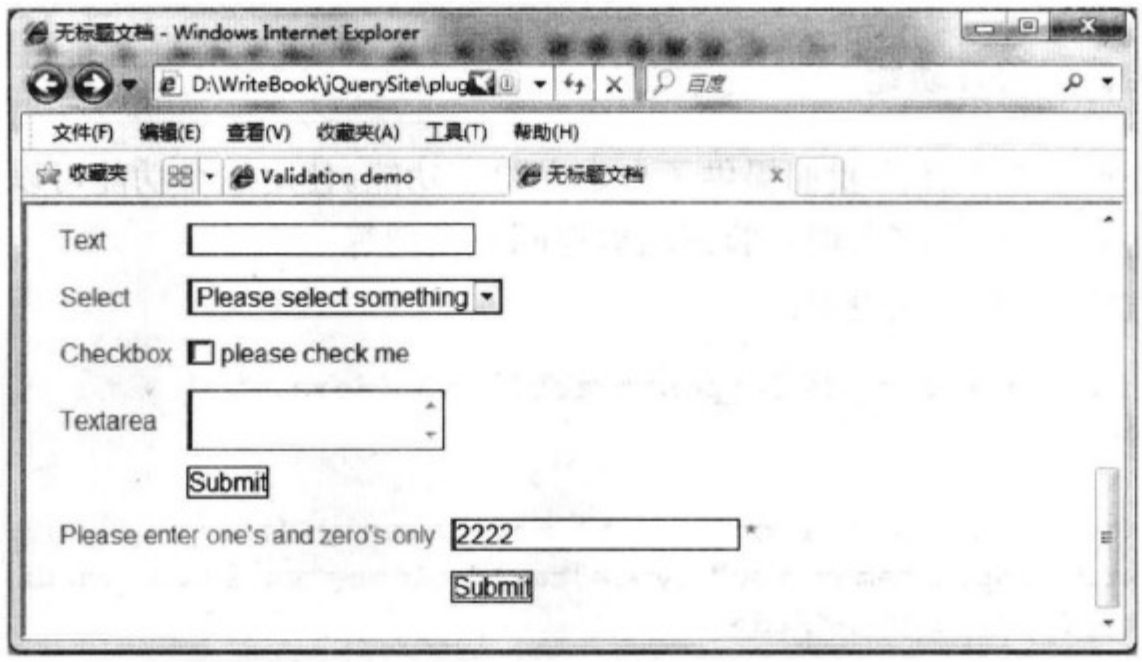


图 7.27　自定义验证功能

5. 自定义输出功能

在前面的例子中，虽然可以修改验证消息的不同表现形式。但是，终究还是在使用 Validation 的自身输出功能。其实，可以对验证消息输出进行自己加工。有特色的验证消息输出更能被用户接受。

首先，创建一个简单的表单：

```
<form id="form5" method="post" action="./demo.html">
<table>
  <tr>
      <th>Enter text please</th>
      <td><input name="foo" type="text" class="validate_required"/></td>
  </tr>
  <tr>
      <td></td>
      <td><input type="submit" value="Submit"/></td>
  </tr>
</table>
</form>
```

自定义的输出功能加在 JavaScript 代码部分：

```
$("#form5").validate({
    // Define the type of error output to use
    error_output:  {
        text:           "fadeInOut"
    },
    // Define the output function
    output:         {
        fadeInOut:  function (element, object) {

            var i,
            fadeIn  =   function(i){
                element.el.fadeIn(100, function(){
```

```
13                      i++;
14                      if(i < 20){
15                          fadeOut(i);
16                      }
17                  });
18              },
19              fadeOut =   function(i){
20                  element.el.fadeOut(100, function(){
21                      i++;
22                      if(i < 20){
23                          fadeIn(i);
24                      }
25                  });
26              };
27              fadeOut(0);
28
29          }
30      }
31  });
```

第 3～5 行定义了自定义输出的函数名及对应哪一个表单元素（文本框）使用此输出。第 7～30 行定义了输出功能的实现，这里使用了 jQuery 的淡入淡出效果，使文本框闪烁来表示验证错误。它的效果因为是动态形式，所以这里我们无法给出效果图，请读者使用光盘中的代码自行测试效果。

6. 对独立于表单的元素进行验证

前面所有的验证都是在表单范围内对元素进行验证的。而且，读者会发现只有我们输入正确的元素内容后，表单才会发生提交动作，否则页面只显示验证消息，表单并没有提交。也可以对不存在于表单内部的元素进行验证。

首先，创建一个单行文本框：

```
1 <input type="text" class="validate_required validateMePlease"/>
2 <a href="#" class="clickMeToValidate">click here to validate</a>
```

在 JavaScript 代码部分，利用超链接的单击事件触发对文本框的验证：

```
1   $(".clickMeToValidate").click(function(){
2       if ($(".validateMePlease").validate() ){
3           alert("Yay I'm valid");
4       }
5       return(false);
6   });
```

具体效果如图 7.28 所示。

7.4.2　JQF1 插件

JQF1 插件为表单的元素提供了新颖的外观样式。它的英文网址为 http://plugins.jquery.com/project/JQF1。首先，在 HTML 的头部分把相关的 JavaScript 文件及 CSS 样式文件引用进来：

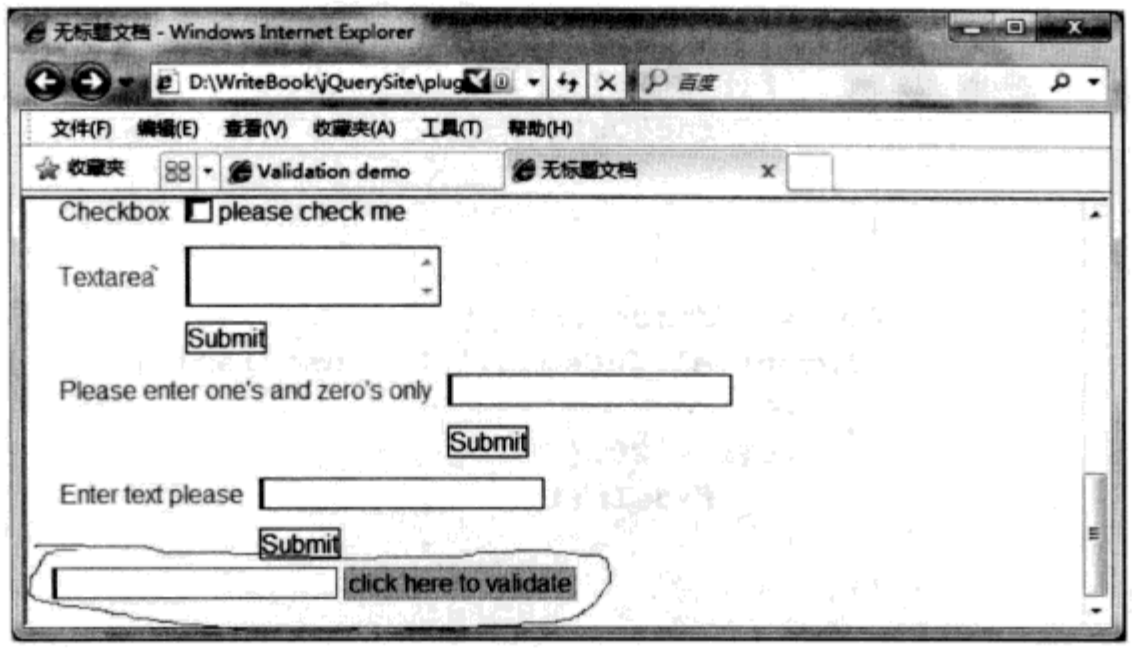


图 7.28　表单外元素验证

```
<script type="text/javascript" src="JS/jquery.js"></script>
<script type="text/javascript" src="JS/jq.jqf1.js"></script>
<script type="text/javascript" src="JS/jqf1.english.js"></script>
<link href="css/jqf1.css" rel="stylesheet" type="text/css" />
```

然后，添加 JavaScript 调用插件的代码：

```
1 <script type="text/javascript">
2   $(document).ready(function(){
3         $('.ul').jqf1();
4         $('#divRadioExample').jqf1();
5   });
6 </script>
```

可以看到代码很简单，只需要调用插件的核心函数就可以了，所有的样式操作都由插件来完成。效果如图 7.29 和图 7.30 所示。

图 7.29　JQF1 插件效果一

图 7.30　JQF1 插件效果二

7.5　小　　结

本章对表单的特效进行了讲解。主要内容包括表单元素的基本特效的实现，以及表单元素插件的使用。重点部分是表单验证框架的使用，同时这一部分也是本章的难点。还对验证框架进行了较深入的讨论。下一章将讲解 jQuery 对图片特效的实现。

7.6　习　　题

【习题 1】利用本章所学内容，应用 Validate 框架实现一个表单的验证功能。

【习题 2】利用本章所介绍的验证插件，制作一个带验证特效的表单。

第8章 设计图片

不管是静态网页还是动态网页，图片都是传达信息的主要载体。很少会有用户能耐下心来阅读长篇大论的说明文章。如果用图片配合文字就很容易被用户接受。如果图片有各种各样的动态效果，就更能吸引用户了。本章主要讲解如何通过 jQuery 来实现图片特效。

8.1 图片切换

jQuery 对图片的应用最常见的就是图片切换效果的实现。图片切换能够节省网页空间，给用户呈现动态效果。网页上一般在表现图片新闻时使用此技术。图片切换的原理是利用 jQuery 的淡入效果或者自定义动画用待显示的图片覆盖现在正在显示的图片。

8.1.1 利用淡入效果实现图片切换

图片切换效果中淡入切换是最简单的一种切换效果。使用到 jQuery 函数 ready()、mouseover()、addClass()、siblings()、removeClass()、fadeIn()、attr()。

图片淡入切换效果实现步骤如下。

（1）为图片中的编号添加鼠标悬停事件。

（2）在事件中将鼠标所停的标号的地方设定为活动状态，其他标号都为非活动状态。

（3）在图片区域将标号所对应的图片应用淡入效果，替换原有图片。

首先，利用 HTML 创建两个列表，一个是图片列表，一个是图片标号列表。CSS 样式设定代码参考光盘内容。

```
1 <div id="content">
2  <ul class="imgarea">
3     <li ><a href="#"><img src="img/bj_0.jpg" width="573" height="257"
      /></a></li>
4     <li ><a href="#"><img src="img/bj_0.jpg" width="573" height="257"
      /></a></li>
5     <li ><a href="#"><img src="img/bj_0.jpg" width="573" height="257"
      /></a></li>
6     <li ><a href="#"><img src="img/bj_0.jpg" width="573" height="257"
      /></a></li>
7  </ul>
8  <ul id="imgID">
9    <li id="bj_0" class="active">按钮 1</li>
10    <li id="bj_1" >按钮 2</li>
```

```
11    <li id="bj_2">按钮 3</li>
12    <li id="bj_3">按钮 4</li>
13  </ul>
14 </div>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/jquery-1.4.min.js"></script>
```

最后，加入 JavaScript 功能代码：

```
1 <script type="text/javascript">
2   $(function(){
3       $('#imgID li').mouseover(function(){          //标号列表项鼠标悬停事件
4           $("#"+this.id).addClass("active").siblings().removeClass
            ("active");                                //将对应的标号设置为激活
5           $('.imgarea li img').fadeIn(4000).attr("src","img/"+(this.id)+
            ".jpg");                                   //根据标号淡入图片
6           });
7   });
8 </script>
```

效果如图 8.1 所示。

图 8.1　淡入切换图片

8.1.2　利用自定义动画切换图片

图片切换效果中自定义动画切换是比较简单的一种切换效果。图片切换的原理是利用 jQuery 的自定义动画函数，在仅可显示一幅图片的区域内，调整整个图片列表底端的位置，每次调整一幅图片的高度，这样就可以在该区域内不断更换图片了。

其中，使用到 jQuery 函数 ready()、mouseover()、index()、hover()、eq()、trigger()、height()、animate()、removeClass()、addClass()，以及 JavaScript 函数 clearInterval()、setInterval()。

其实现的步骤如下。

（1）设定图片标号的鼠标悬停事件。

（2）在事件中利用自定义动画函数调整显示图片，并修改对应标号样式。

（3）为图片显示区域设定鼠标悬停事件。

（4）当鼠标停在该区域时，清除图片切换动画定时器。

（5）当鼠标离开该区域时，重启图片切换动画，每隔 2 秒换一张图片。

首先，利用 HTML 创建两个列表，一个是图片列表，一个是标号列表。CSS 样式设定代码参考光盘内容。

```
<div class="content">
 <div class="main" >
   <ul class="imgarea" >
     <li><a href="#"><img src="img/bj_0.jpg" /></a></li>
     <li><a href="#"><img src="img/bj_1.jpg"/></a></li>
     <li><a href="#"><img src="img/bj_2.jpg"/></a></li>
     <li><a href="#"><img src="img/bj_3.jpg"/></a></li>
   </ul>
   <ul class="imgID" >
      <li>1</li>
      <li>2</li>
      <li>3</li>
    <li>4</li>
    </ul>
 </div>
</div>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/jquery-1.4.min.js"></script>
```

最后，加入 JavaScript 功能代码：

```
<script type="text/javascript">
  $(function(){
       var index = 0;
       var timer;
       $(".imgID li").mouseover(function(){          //图片标号的鼠标悬停事件
          index  =  $(".imgID li").index(this);  //获取图片标号的索引值
          animateImg(index);                          //显示与索引值匹配的图片
       }).eq(0).mouseover();
       $('.main').hover(function(){                  //图片显示区域的鼠标悬停事件
            clearInterval(timer);                    //清除定时器
        },function(){
            timer = setInterval(function(){//设定定时器，循环显示每张图片
               animateImg(index)
               index++;
               if(index==$(".imgID > li").length){index=0;}
             } , 2000);
       }).trigger("mouseleave");
```

```
18  })
19  function animateImg(index){
20          var divh = $(".content .main").height();
21          $(".imgarea").stop(true,false).animate({top: -divh*index},1000);
            //利用动画效果调整图片列表行高
22          $(".imgID li").removeClass("active")
23              .eq(index).addClass("active"); //更改图片标号样式
24  }
25</script>
```

效果如图 8.2 所示。

图 8.2　利用自定义动画切换图片

8.2　图 片 滚 动

图片滚动效果大体分为垂直滚动和水平滚动两种。8.1.2 节介绍的图片切换其实就是一种垂直循环滚动效果。本节将介绍水平滚动效果。

它的实现原理是利用 jQuery 的 scrollLeft()函数不断修改水平滚动条的偏移量，使图片相对原有位置发生偏移，产生滚动效果。jQuery 还有一个函数 scrollTop()，它的作用是垂直滚动。其中，使用到 jQuery 函数 ready()、html()、scrollLeft()、width()、hover()和 JavaScript 函数 clearInterval()、setInterval()。

1．jQuery的函数scrollLeft()——水平滚动

该函数返回或设置匹配元素的滚动条的水平位置。其语法形式如下：

```
语法形式一：scrollLeft()
语法形式二：scrollLeft(position)
```

注：第一种语法形式返回元素在滚动条的当前位置。第二种语法形式通过参数 position 设定元素的滚动条位置。

2. 功能实现

（1）构建滚动区域，也就是定义滚动条的距离。

（2）设定滚动区域的鼠标悬停与离开事件。

（3）利用 jQuery 的 scrollLeft()函数实现滚动效果。

首先，通过 HTML 将滚动区域和图片创建出来：

```
<div id="scrollarea" style="overflow: hidden; width: 500px;">
    <table border="0" align="center">
        <tr>
            <td id="area1" valign="top" bgcolor="ffffff">
                <table border="0" cellspacing="0" cellpadding="0">
                    <tr align="center">
                        <td>
                            <a href="#" target="_blank">
                                <img src="img/bj_0.jpg" width="335" alt="
                                第一张图" height="260"></a>
                        </td>
                        <td>
                            <a href="#" target="_blank">
                                <img src="img/bj_1.jpg" width="335" alt="
                                第二张图" height="260"></a>
                        </td>
                        <td>
                            <a href="#" target="_blank">
                                <img src="img/bj_2.jpg" width="335" alt="
                                第三张图" height="260"></a>
                        </td>
                        <td>
                            <a href="#" target="_blank">
                                <img src="img/bj_3.jpg" width="335" alt="
                                第四张图" height="260"></a>
                        </td>
                    </tr>
                </table>
            </td>
            <td id="area2" valign="top">
            </td>
        </tr>
    </table>
</div>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
<script type="text/javascript">
var timer;
$("#area2").html($("#area1").html());                    //构建滚动区域
function imgMarquee(){
    if($("#scrollarea").scrollLeft()>=$("#area1").width())
                                                         //判定滚动条长度是否超长
        $("#scrollarea").scrollLeft(0);                  //判定滚动条长度是否超长
    else{
        $("#scrollarea").scrollLeft($("#scrollarea").scrollLeft()+5);
                                                         //变化滚动条偏移位置
    }
 }
$("#scrollarea").hover(function(){                       //滚动区域的鼠标悬停事件
        clearInterval(timer);                            //停止滚动
    },
    function(){
        timer=setInterval(imgMarquee,10);                //连续滚动
});
 </script>
```

效果如图 8.3 所示，这里只是静态效果，具体滚动效果请读者运行光盘中的代码查看。

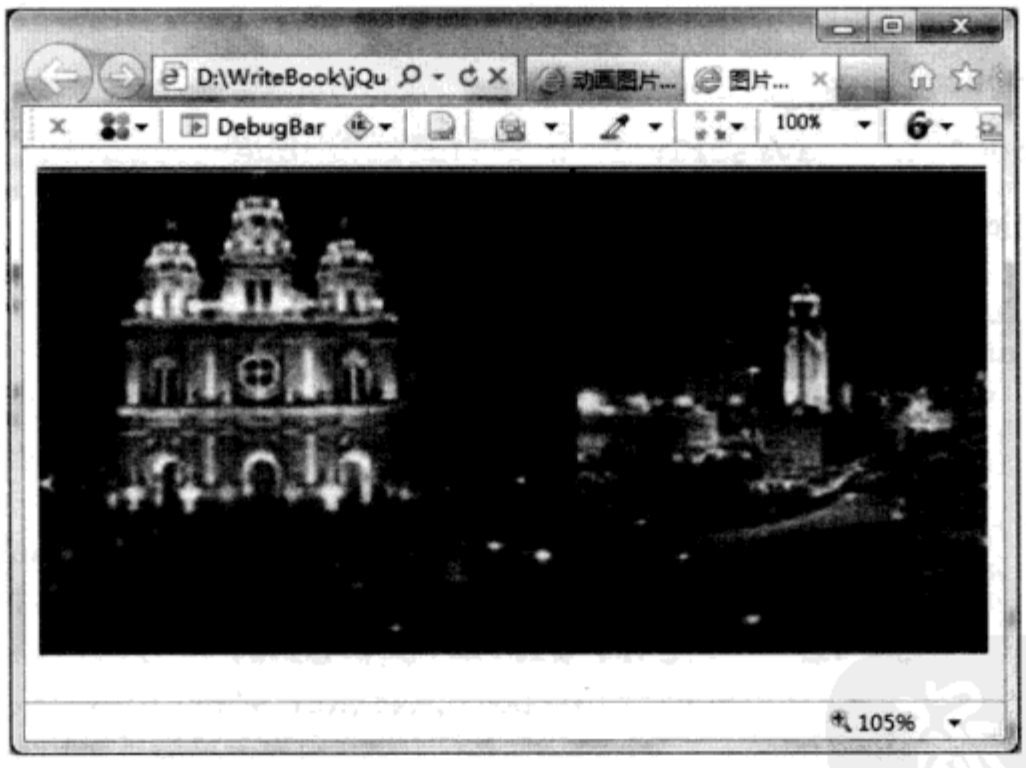


图 8.3　水平滚动图片

8.3　图片动态弹出

图片的弹出在很多商业网站上都有应用，它的主要作用是动态展示产品外观。因为产品图片大小不一，全部图片放在同一个页面中显示出来是一件很痛苦的事情，所以，可以

先将图片隐藏起来，等用户需要显示的时候再动态弹出给用户。

它的实现原理很简单，利用层的隐藏与显示，将待显示的图片放在层上。如果层被隐藏，则图片不可见。如果某个事件触发显示了层，则图片就有了弹出效果。其中，使用到 jQuery 函数 ready()、click()、css()、height()、width()、hide()、show()。

1. 功能实现

（1）设定页面中触发图片的弹出事件，这里使用了一个超链接的单击事件。

（2）在事件中设定图片所在层需要显示的位置，大小根据图片自动调整。

（3）在图片上还有一个小的关闭样式的图片，设定它的单击事件。

（4）在事件中隐藏图片。

首先，通过 HTML 将滚动区域和图片创建出来。样式设定的 CSS 代码请读者参考光盘内容。

```
<a   id="popup" href="#1">弹出图片</a>
<div id="imgarea">
    <div class="biaoti"><a href="#1"><img id="close" src="img/wrong.
    png" border="0" alt="close"/></a></div>
    <img src="img/bj_0.jpg" alt="aaa"/>
</div>
<div id="mask">
</div>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
<script>
    $(function(){
        $("#popup").click(function(){   //触发图片弹出事件
            $("#imgarea").css("top",($(window).height()-$("#imgarea").
            height())/2+"px")
            .css("left",($(window).width()-$("#imgarea").width())/2+
            "px").show();                    //计算图片所在层在页面中的位置
            $("#mask").css("width",$(window).width()+"px")
            .css("height",$(window).height()+"px").css("opacity","0.5").
            show();                          //显示屏蔽页面其他内容的层
        });
        $("#close").click(function(){   //触发隐藏图片事件
            $("#imgarea").hide();        //隐藏图片
            $("#mask").hide();           //隐藏屏蔽层
        });
    });
</script>
```

效果如图 8.4 所示。

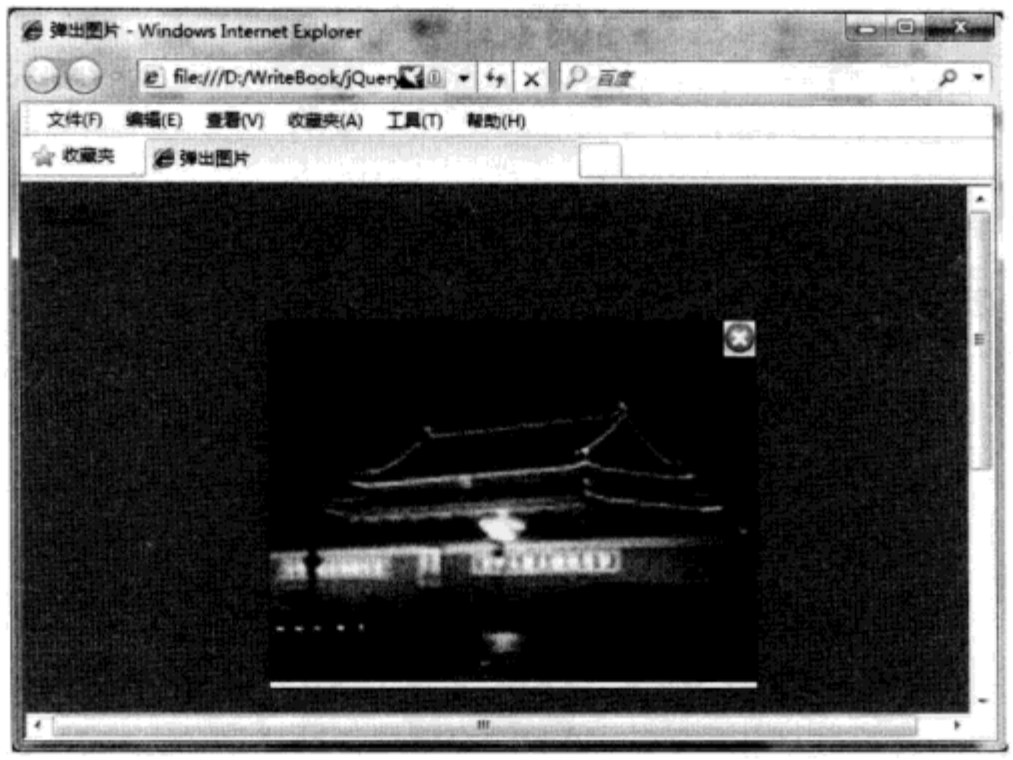


图 8.4　图片弹出效果

8.4　动态图文结合

动态图文结合的典型应用就是图片上的文字提示功能，也就是当鼠标悬停在图片上时动态出现的文字提示效果。这种效果省去了在页面上出现大段文字描述，是一种友好的用户体验。它的实现原理是利用图片的鼠标悬停与离开事件，当鼠标悬停在图片上时，跟随鼠标位置动态显示提示文字，当鼠标离开图片时，提示文字所在的层隐藏。

其中，使用到了 jQuery 函数 ready()、mouseover()、mouseout()、append()、fadeIn()、fadeout()、css()。图片动态文字提示实现步骤如下。

（1）通过 JavaScript 向页面中加入一个包含提示文字的隐藏层。

（2）设定图片的鼠标悬停和离开事件。

（3）在鼠标的悬停事件中显示层并根据鼠标位置定位。

（4）在鼠标离开事件中隐藏层。

首先，编写 HTML 代码。样式设定的 CSS 代码请读者参考光盘内容。

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
<script type="text/javascript">
$(document).ready(function(){
    $("body").append("<div id='tips'>小破孩</div>");    //添加信息提示层
    $("img").mouseout(function(){
        $("#tips").fadeOut("fast");    //鼠标离开图片，淡出层
    });
    $("img").mousemove(function(e){    //鼠标进入图片，淡入层并重新定位
        $("#tips")
```

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

```
9              .css("top",(e.pageY - 5) + "px")
10             .css("left",(e.pageX + 5) + "px").fadeIn("slow");
11         });
12 });
13 </script>
```

效果如图 8.5 所示。

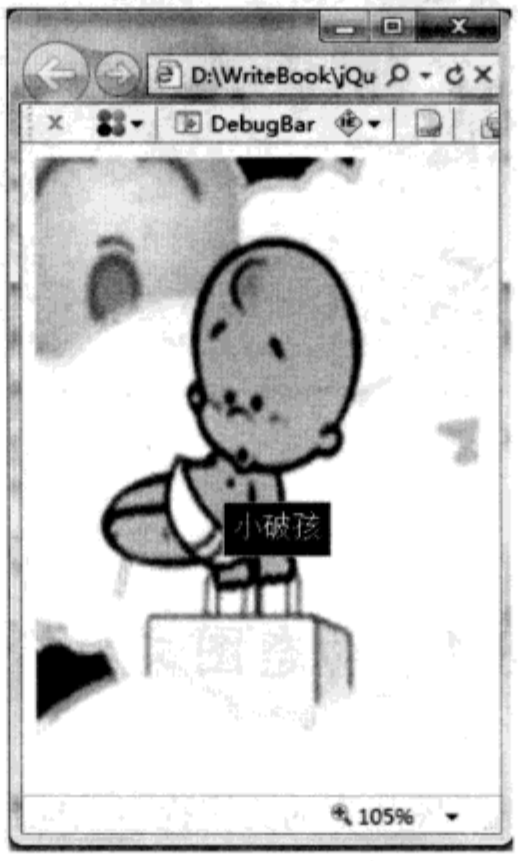


图 8.5　图片动态添加文字提示

8.5　图 片 剪 切

图片剪切主要是为了解决若干个图片大小不一的情况下如何调整统一大小。图片剪切不是缩小图片大小，而是将图片的部分截取下来。它的工作原理是：将图片与显示区域大小进行对比，按照显示区域大小截取相应大小比例的图片部分进行显示。其中，要使用到 jQuery 函数 ready()、parent()、height()、width()、css()。

图片剪切功能实现步骤如下。

（1）获取图片的大小及显示区域大小。

（2）比较图片与显示区域的大小，判断是否全部显示图片。

（3）根据比较结果在显示区域显示图片的部分区域。

首先，编写 HTML 代码。样式设定的 CSS 代码请读者参考光盘内容。

```
1 <div class="part1" style="margin-top:20px;">
2   <ul>
3       <li><a href="#"><img src="img/img1.jpg" width="300" height="372"
        class="t-img"></a></li>
4       <li><a href="#"><img src="img/img3.jpg" width="497" height="306"
        class="t-img"></a></li>
```

```
5       <li><a href="#"><img src="img/img5.jpg" width="452" height="531"
        class="t-img"></a></li>
6       <li><a href="#"><img src="img/img6.jpg" width="428" height="319"
        class="t-img"></a></li>
7   </ul>
8 </div>
9 <div class="part2" style="margin-top:20px;">
10  <ul>
11      <li><a href="#"><img src="img/img1.jpg" width="300" height="372"
        class="t-img"></a></li>
12      <li><a href="#"><img src="img/img3.jpg" width="497" height="306"
        class="t-img"></a></li>
13      <li><a href="#"><img src="img/img5.jpg" width="452" height="531"
        class="t-img"></a></li>
14      <li><a href="#"><img src="img/img6.jpg" width="428" height="319"
        class="t-img"></a></li>
15  </ul>
16 </div>
17 <div class="part3" style="margin-top:20px;">
18  <ul>
19      <li><a href="#"><img src="img/img1.jpg" width="300" height="372"
        class="t-img"></a></li>
20      <li><a href="#"><img src="img/img3.jpg" width="497" height="306"
        class="t-img"></a></li>
21      <li><a href="#"><img src="img/img5.jpg" width="452" height="531"
        class="t-img"></a></li>
22      <li><a href="#"><img src="img/img6.jpg" width="428" height="319"
        class="t-img"></a></li>
23  </ul>
24 </div>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
1 <script type="text/javascript">
2   $(function(){
3       $(".cutimg").each(function(){
4           if($(this).height()>$(this).parent().height() && $(this).width
            ()>$(this).parent().width())
                //判断是否全部显示图片
5           {
                //在显示区域显示图片的部分区域
6               $(this).css("top",$(this).height()*($(this).parent().height
                ()/$(this).height())+"px");
7               $(this.css("left",$(this).width()*($(this).parent().width
                ()/$(this).width())+"px");
8           }
9       });
10  });
11 </script>
```

效果如图 8.6 所示。

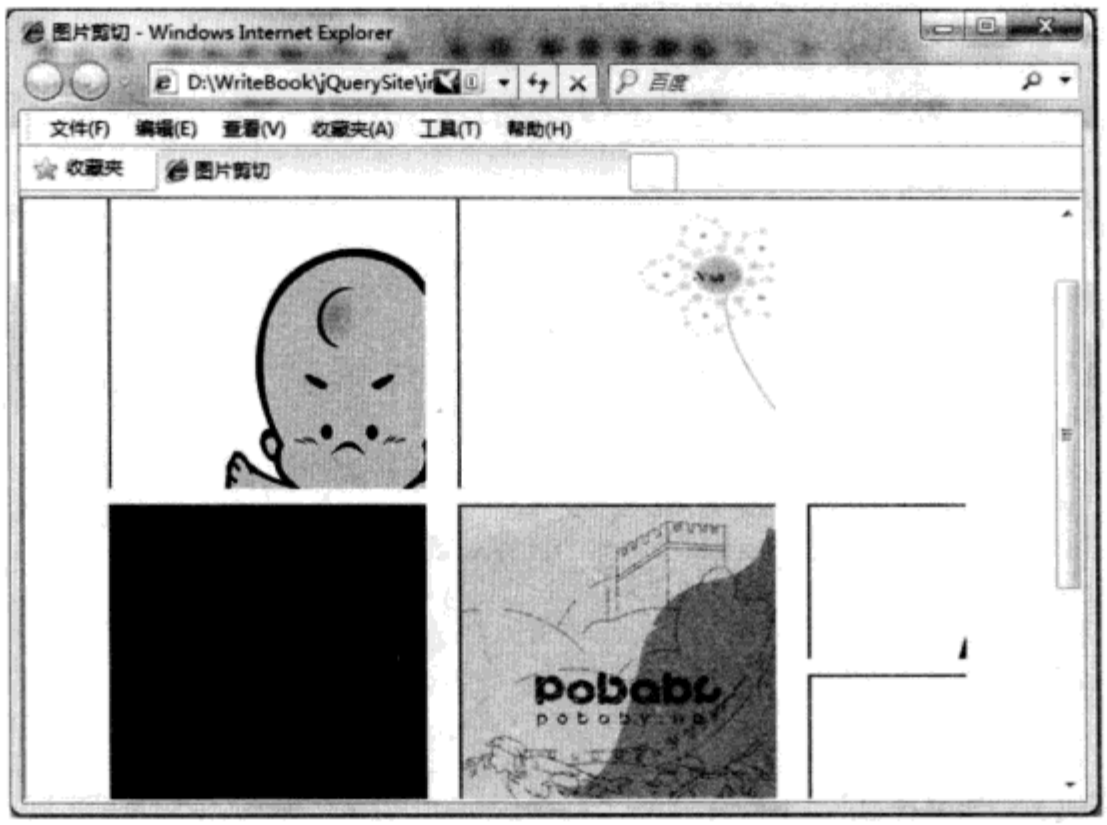


图 8.6　图片的裁减

8.6　图 片 预 览

在类似于淘宝的一些网站上会出现这么一种效果，页面加载完成后显示小图片，但当将鼠标放在小图片上时会出现相应的大图片，鼠标从小图片上移开后大图也随之消失。这种效果称为图片预览。它的实现原理就是利用鼠标的悬停与移开事件，在鼠标悬停的时候将隐藏的大图片显示出来，并跟随鼠标在小图上移动，鼠标离开小图后，大图继续隐藏。其中，使用到了 jQuery 函数 ready()、hover()、appendTo()、fadeIn()、remove()、mouseover()。

图片预览实现步骤如下。

（1）在小图的鼠标悬停事件中，添加大图元素，并设定大图的位置距鼠标当前位置有一定偏移，大图淡入。

（2）在小图的鼠标离开事件中，移除大图。

首先，利用 HTML 创建小图，并添加 CSS 样式设定，具体代码请参考光盘内容。

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
1    <script type="text/javascript">
2        $(function(){
3            ImgPreview();
4        });
5
6        var ImgPreview = function() {
7            $("#smallimg").hover(function(e) {      //鼠标悬停与离开事件
8                jQuery("<img class='preview' src='" + this.src + "' />")
```

```
            .appendTo("body");                          //添加大图元素
9           $(".preview")                               //设定大图位置并淡入
10              .css("top", (e.pageY - 5) + "px")
11          .css("left", (e.pageX + 5) + "px")
12              .fadeIn("fast");
13      }, function() {
14          $(".preview").remove();                     //移除大图
15      });
16      $("#smallimg").mousemove(function(e) {//补充鼠标悬停事件
17          $(".preview")
18              .css("top", (e.pageY - 5) + "px")
19          .css("left", (e.pageX + 5) + "px")
20      });
21  };
22  </script>
```

效果如图 8.7 所示。

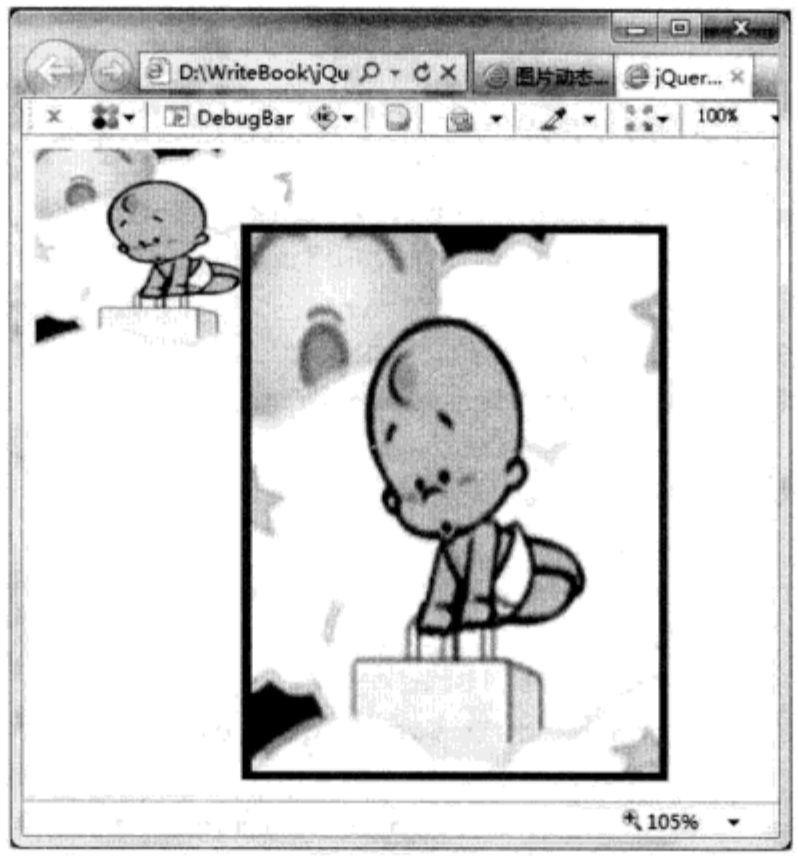


图 8.7　图片预览效果

8.7　图片局部平移

图片局部平移也是经常出现在一些商业性网站如京东商城中的图片效果。这种效果主要是在原始图片较大、直接在页面显示会影响页面布局与美观、但用户又无法看清小图片的情况下使用。它的实现原理是：在小图的鼠标悬停事件中获取鼠标的位置，并按照一定比例裁减原大图的部分内容，在固定的层上进行显示，当鼠标离开时层取消。其中，使用到了 jQuery 函数 ready()、hover()、get()、attr()、after()、css()、width()、height()、show()、mouseover()、hide()、unbind()、remove()。

图片局部平移实现步骤如下。

（1）设定小图的鼠标悬停与离开事件。

（2）在鼠标悬停事件中向页面添加显示图片局部内容的层，在层上加载大图。

（3）在鼠标悬停事件中添加页面的鼠标移动事件，在这个事件中判断鼠标在小图的位置并改变层对于滚动条的偏移量，以实现显示大图不同位置的效果。

（4）在鼠标离开事件中撤销层的显示，并撤销页面的鼠标移动事件。

首先，利用 HTML 创建小图，并添加 CSS 样式设定，具体代码请参考光盘内容。

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/ jquery-1.6.js "></script>
```

最后，加入 JavaScript 功能代码：

```
<script type="text/javascript">
    $(function(){
        $("img.smallimg").hover(function(){          //小图的鼠标悬停事件
            var imageLeft = $(this).get(0).offsetLeft;
            var imageTop =  $(this).get(0).offsetTop;
            var imageWidth = $(this).get(0).offsetWidth;
            var imageHeight = $(this).get(0).offsetHeight;
            $(this).after("<div class='rawimage'><img class='rawimg'
            src='"+$(this).attr("alt")+"'/></div>"); //添加显示局部图的层
            leftpos = imageLeft + imageWidth +10;
            $("div.rawimage").css({ top: imageTop,left: leftpos });
            $("div.rawimage").show();                //设定层位置并显示
            $("body").mousemove(function(e){         //页面的鼠标移动事件
                var bigwidth = $(".rawimg").get(0).offsetWidth;
                var bigheight = $(".rawimg").get(0).offsetHeight;
                var scalex = Math.round(bigwidth/imageWidth) ;
                var scaley = Math.round(bigheight/imageHeight);
                scrolly = e.pageX - imageTop - ($("div.rawimage").
                height()*1/scaley)/2 ;
                $("div.rawimage").get(0).scrollTop = scrolly * scaley ;
                                                 //纵向滚动大图到合适位置
                scrollx = e.pageY - imageLeft - ($("div.rawimage").
                width()*1/scalex)/2 ;
                $("div.rawimage").get(0).scrollLeft = (scrollx) * scalex ;
                                                 //横向滚动大图到合适位置
                                                 //移动层显示图片的不同部分
            });
        },
        function(){     //小图的鼠标离开事件，隐藏层，解除绑定页面鼠标移动事件
            $("div.rawimage").hide();
            $("body").unbind("mousemove");
            $("div.rawimage").remove();
        });
    });
</script>
```

在上面的代码中既使用了层的显示与隐藏，也使用了层相对于滚动条的移动。显示大图局部内容的层大小不宜过大，以原图的 1/4 大小左右比较适合，效果如图 8.8 所示。

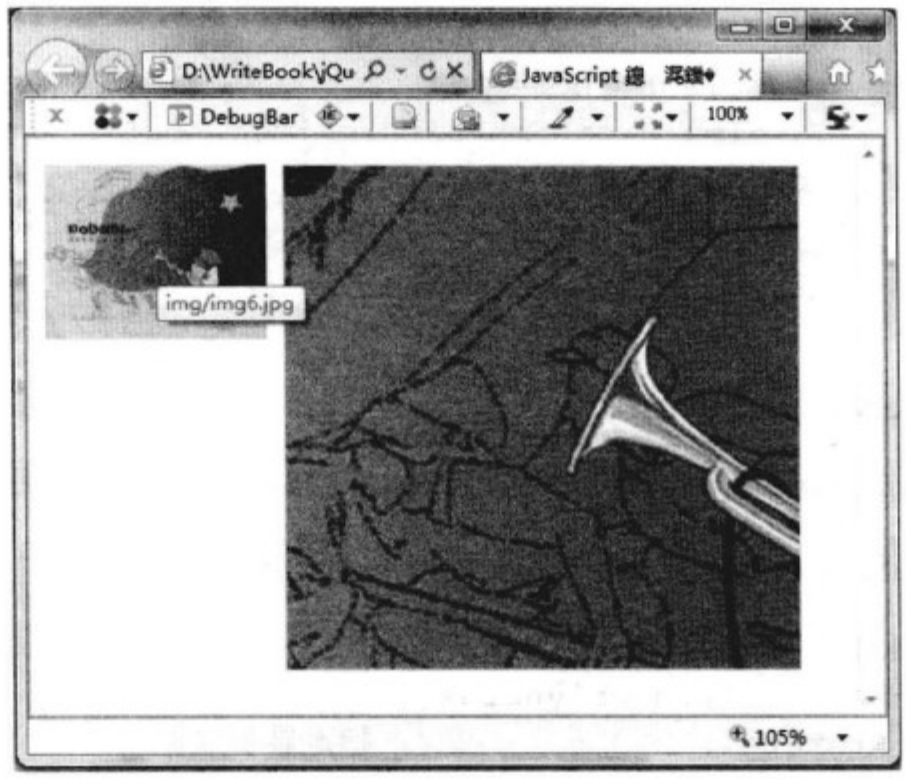


图 8.8　图片局部平移效果

8.8　图 片 插 件

前面讲解了部分通过 jQuery 实现的图片效果。因为图片是网页中的基本元素之一，所以关于它的插件也比较多。下面介绍三种关于图片的插件供读者参考。

8.8.1　MobilyNotes 插件

MobilyNotes 是一个轻量级的 jQuery 插件（只有 2KB），可以以堆叠的形式显示其图片集或者 HTML 内容集。这个插件的特点如下。

（1）循环显示内容。

（2）自动播放。

（3）自动产生按钮。

首先，使用 HTML 来创建页面。CSS 代码和插件代码请参考光盘内容。

```
<div id="content">
    <div class="wrap">
        <div class="notes_img">
            <div class="note">
                <img src="img/nature/img1.jpg" alt="" />
            </div>
            <div class="note">
                <img src="img/nature/img2.jpg" alt="" />
            </div>
            <div class="note">
                <img src="img/nature/img3.jpg" alt="" />
            </div>
            <div class="note">
                <img src="img/nature/img4.jpg" alt="" />
```

```
            </div>
            <div class="note">
                <img src="img/nature/img5.jpg" alt="" />
            </div>
        </div>
    </div>
</div>
```

然后，引入 jQuery 库文件：

```
<script src="js/jquery.js" type="text/javascript"></script>
```

最后，通过 JavaScript 功能代码讲解如何使用这个插件：

```
$(function(){
    $('.notes_img').mobilynotes({
    init: 'rotate',                 // 指定显示方式
    positionMultiplier: 5,          // 显示图片个数
    title: null,                    // 设置标题
    showList: true,                 // 创建无序列表
    autoplay: true,                 // 自动播放
    interval: 4000                  // 自动播放间隔时间
    });
});
```

上述代码中第 2 行是插件的调用函数；第 3 行设定了初始显示方式，rotate 表示显示时图片会有旋转，plain 表示图片正常显示。第 4 行表示循环显示图片的个数。第 5 行表示从元素中获取标题。第 6 行表示创建无序列表。第 7 行表示是否自动播放。第 8 行表示图片轮换的间隔时间，以毫秒为单位。经过上面的设置后，效果如图 8.9 所示。

图 8.9　MobilyNotes 循环播放图片

8.8.2　Fancybox 插件

Fancybox 是一个浮动展示放大图片的插件。它可以设定浮动层出现图片、HTML 内容、

多媒体，也可以在同一个浮动层浏览多张图片，还可以设定浮动层出现不同动画效果。

这个插件的特点如下。

（1）可以显示多种元素，如图片、HTML 内容、多媒体、Ajax 请求内容。

（2）自由定制 CSS 样式。

（3）相关项目分组和添加导航。

（4）可以添加鼠标滚轮相应功能。

（5）支持多种转换。

（6）具有完美的浮动层阴影。

插件相关参数如表 8.1 所示。

表 8.1　Fancybox参数

属性关键字	默　认　值	描　　述
padding	10	嵌入内容与 Fancybox 之间的空间距离
margin	20	Fancybox 与其他内容之间的空间距离
opacity	false	在弹性转换方式下内容的透明度是否改变
modal	false	当其为真的时候，'overlayShow' 被设置为真，'hideOnOverlayClick'，'hideOnContentClick'，'enableEscapeButton'，'showCloseButton'被设置为假
cyclic	false	图片是否循环显示，可以步进和后退
scrolling	auto	覆盖层是否具有滚动条，可以为“auto”、“yes”和“no”
width	560	设置“iframe”，“swf”的宽度。如果禁止自动调整大小，则可以设置内容的宽度
height	340	设置“iframe”，“swf”的高度。如果禁止自动调整大小，则可以设置内容的高度
autoscale	true	调整 Fancybox 的大小，使之正好适合可见区域
autoDimensions	true	自动为内容调整大小
centerOnScroll	false	Fancybox 居中
ajax		Ajax 选项
swf	{wmode: 'transparent'}	输出 swf 内容选项
hideOnOverlayClick	false	是否点击内容关闭 Fancybox
overlayShow	true	是否显示覆盖层
overlayOpacity	0.3	覆盖层的透明度，从 0 到 1
overlayColor	'#666'	覆盖层颜色
titleShow	true	标题
titlePosition	'outside'	标题位置，可以设置为'outside', 'inside' 或者'over'
titleFormat	null	标题格式，可以设置 HTML 内容、图片、导航功能
transitionIn, transitionOut	'fade'	转换类型，可以设置为'elastic', 'fade' or 'none'
speedIn, speedOut	300	淡入淡出或者弹入弹出的速度
changeSpeed	300	改变图片时重新设置大小的速度
changeFade	'fast'	改变图片时的速度
easingIn, easingOut	'swing'	弹出动画时的设置
showCloseButton	true	显示关闭按钮
showNavArrows	true	显示导航箭头
enableEscapeButton	true	Esc 按钮退出 Fancybox

续表

属性关键字	默　认　值	描　　述
type		指定内容类型，可以为'image', 'ajax', 'iframe', 'swf' 或者 'inline'
href		指定内容资源
title		指定标题
content		指定内容
orig		设置对象弹出位置与大小
index		设定开始显示图片的索引号

插件相关函数如表 8.2 所示。

表 8.2　Fancybox的函数

方　　法	描　　述
$.fancybox.showActivity	显示加载动画
$.fancybox.hideActivity	隐藏加载动画
$.fancybox.next	显示下一个画廊图片
$.fancybox.prev	显示前一个画廊图片
$.fancybox.pos	显示画廊图片的索引值
$.fancybox.cancel	取消加载内容
$.fancybox.close	隐藏 Fancybox
$.fancybox.resize	自动调整 Fancybox 大小
$.fancybox.center	Fancybox 居中

下面就利用官方给出的示例来看一下这个插件的基本使用情况。

首先，建立静态页面，具体的 HTML 代码请参考光盘内容。在 HTML 文件的头部分加入如下内容：

```
1   <script type="text/javascript" src="http://ajax.googleapis.com/ajax/
    libs/jquery/1.4/jquery.min.js"></script>
2   <script>
3       !window.jQuery && document.write('<script src="jquery-1.4.3.min.
        js"><\/script>');
4   </script>
5   <script type="text/javascript" src="./fancybox/jquery.mousewheel-3.0.
    4.pack.js"></script>                          //添加鼠标滚动相应文件
6   <script type="text/javascript" src="./fancybox/jquery.fancybox-1.3.
    4.pack.js"></script>                          //添加插件文件
7   <link rel="stylesheet" type="text/css" href="./fancybox/jquery.fancy
    box-1.3.4.css" media="screen" />              //添加插件文件所需 CSS 文件
8   <link rel="stylesheet" href="style.css" />
```

第 1～4 行是添加 jQuery 库文件代码，第 5 行是添加 jQuery 对鼠标滚轮事件支持的库文件，第 6 行是 Fancybox 的插件文件，第 7 行是 Fancybox 的 CSS 样式文件，第 8 行是当前页面的 CSS 样式文件。

下面，分析调用 Fancybox 的 JavaScript 功能代码：

```
1   <script type="text/javascript">
2       $(document).ready(function() {
```

```
3           /*
4           *   使用用例 1
5           */
6           $("a#example1").fancybox();//默认初始化插件
7           $("a#example2").fancybox({ //不使用遮盖层，使用弹出弹入转换效果
8               'overlayShow'   : false,
9               'transitionIn'  : 'elastic',
10              'transitionOut' : 'elastic'
11          });
12          $("a#example3").fancybox({ //不使用转换特效
13              'transitionIn'  : 'none',
14              'transitionOut' : 'none'
15          });
16          $("a#example4").fancybox({ //不透明，无遮盖层，弹入效果，无弹出效果
17              'opacity'       : true,
18              'overlayShow'   : false,
19              'transitionIn'  : 'elastic',
20              'transitionOut' : 'none'
21          });
22          $("a#example5").fancybox();//默认初始化插件
23          $("a#example6").fancybox({   //设定图片标题位置,遮盖层黑色,不透明度 0.9
24              'titlePosition'     : 'outside',
25              'overlayColor'      : '#000',
26              'overlayOpacity'    : 0.9
27          });
28          $("a#example7").fancybox({ //设定标题位置
29              'titlePosition' : 'inside'
30          });
31          $("a#example8").fancybox({ //设定标题位置
32              'titlePosition' : 'over'
33          })
34          $("a[rel=example_group]").fancybox({
                                //不使用弹性特效，设定标题位置，格式化标题显示内容
35              'transitionIn'      : 'none',
36              'transitionOut'     : 'none',
37              'titlePosition'     : 'over',
38              'titleFormat'       : function(title, currentArray, curren
                tIndex, currentOpts) {
39                  return '<span id="fancybox-title-over">Image ' + (cur
                    rentIndex + 1) + ' / ' + currentArray.length + (title.
                    length ? '   ' + title : '') + '</span>';
40              }
41          });
42          /*
43          *   使用用例 2
44          */
45          $("#various1").fancybox({   //设定标题位置，不使用弹性特效
46              'titlePosition'     : 'inside',
47              'transitionIn'      : 'none',
48              'transitionOut'     : 'none'
49          });
50          $("#various2").fancybox(); //默认初始化插件
```

```
51          $("#various3").fancybox({
                                    //设定插件区域大小，不自动调整，不使用弹性特效
52              'width'             : '75%',
53              'height'            : '75%',
54              'autoScale'         : false,
55              'transitionIn'      : 'none',
56              'transitionOut'     : 'none',
57              'type'              : 'iframe'
58          });
59          $("#various4").fancybox({
                                    //无内边距空间，不自动调整大小，不使用弹性特效
60              'padding'           : 0,
61              'autoScale'         : false,
62              'transitionIn'      : 'none',
63              'transitionOut'     : 'none'
64          });
65      });
66  </script>
```

第 6 行是最简单的一种调用 Fancybox 的代码，Fancybox 的所有属性值均未指定，使用默认值，静态效果如图 8.10 所示。第 7～11 行指定了图片弹出弹入方式，并设定图片弹出后不使用遮盖层，静态效果如图 8.11 所示。第 12～15 行指定了图片转换出时无特效，它的静态效果如图 8.10 所示。第 16～21 行指定了图片转换后改变透明效果，不显示遮盖层，只有弹出效果而无弹入效果，静态效果如图 8.11 所示。第 22 行设定和第 6 行相同，都使用默认属性值。

第 23～27 行指定了标题显示位置在图片外，遮盖层颜色和透明度，静态效果如图 8.12 所示。第 28～30 行指定了图片的标题位置在图片区域内，静态效果如图 8.13 所示。第 31～33 行指定了标题浮动于图片上，静态效果如图 8.14 所示。第 34～41 行指定了多张图片以画廊形式出现，并设定每张图片的标题位置浮动在图片之上，并设定了标题样式的 HTML 格式，并支持通过鼠标滚轮更换图片，静态效果如图 8.15 所示。

第 45～49 行指定内联页面中的部分文本、标题内置、无转换动画，静态效果如图 8.16 所示。第 50 行指定加载外部文件。第 51～58 行指定加载一个 iframe 元素，并指定了高和宽，不自动调整大小，无转换动画，静态效果如图 8.17 所示。第 59～64 行指定加载的 swf 文件无间隔，不自动调整，无转换动画效果，静态效果如图 8.18 所示。

图 8.10　使用默认属性值的图片展示

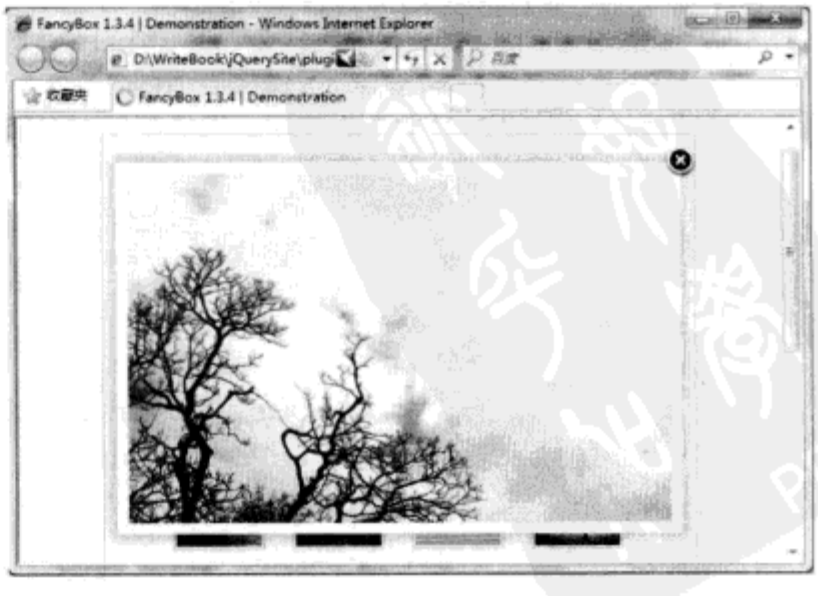


图 8.11　无遮盖层的图片展示

图 8.12　改变遮盖层样式的图片展示

图 8.13　标题内置图片展示

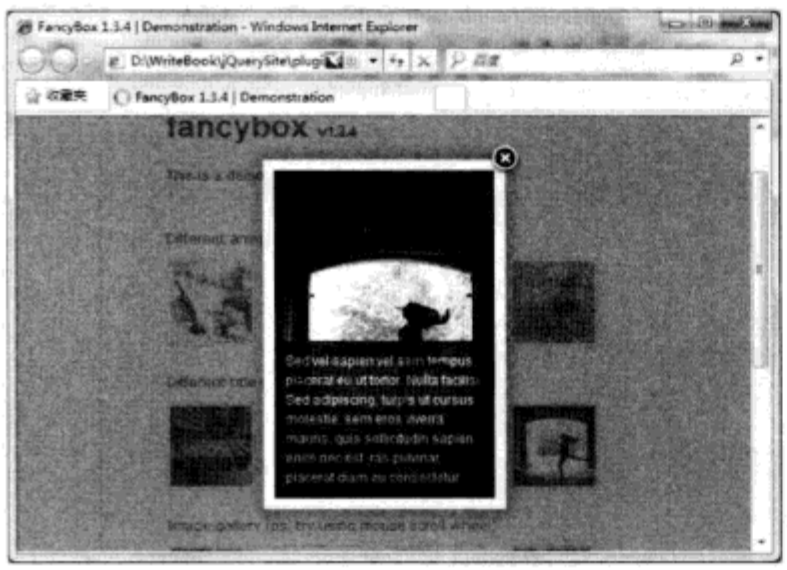


图 8.14　浮动标题的图片展示

图 8.15　画廊效果

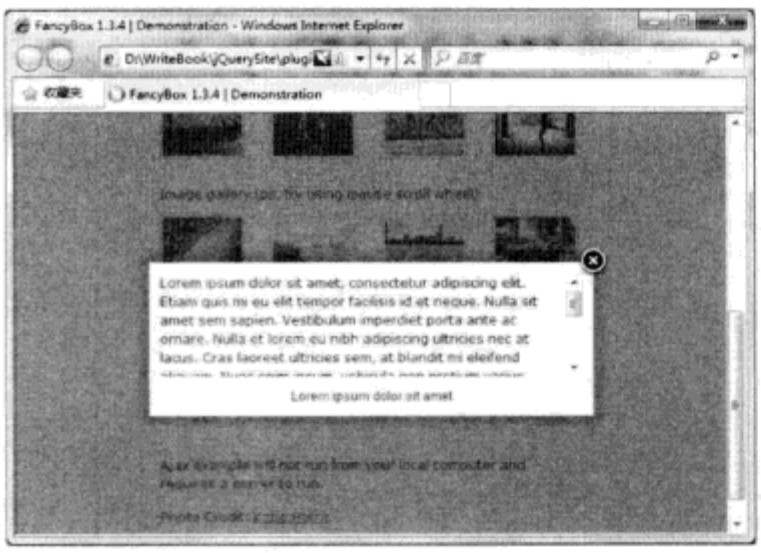


图 8.16　内联文本

图 8.17　iframe 的展示

8.8.3　desSlideshow 插件

这个插件是一个图片轮播展示插件，它支持自动轮播和手动选择。下面通过一个示例来讲解如何使用这个插件。

首先，建立静态页面。HTML 代码和 CSS 样式代码参考光盘内容。在页面的头部添加如下代码：

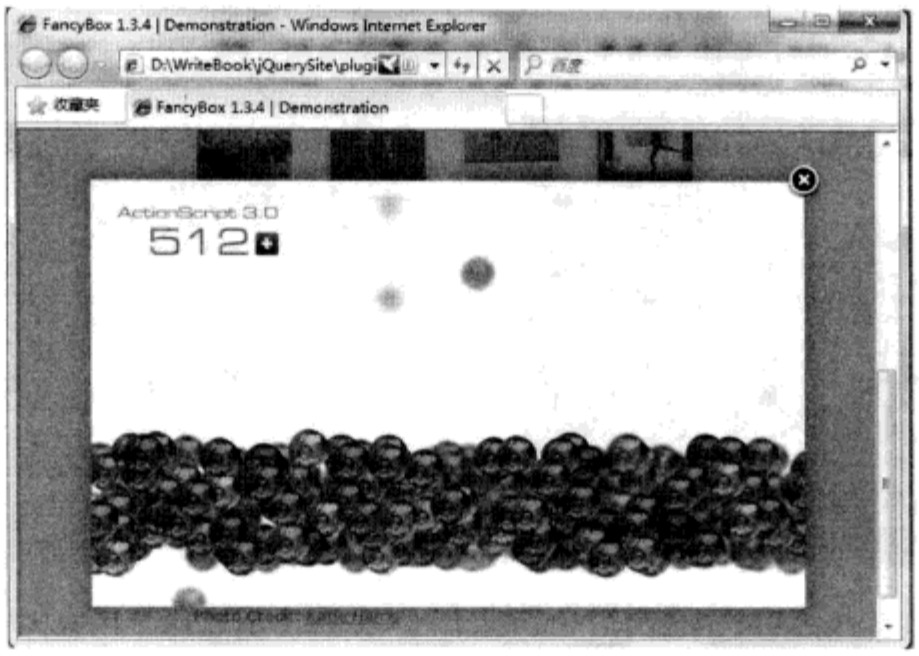


图 8.18　多媒体的展示

```
<link href="css/style.css" rel="stylesheet" type="text/css" />
<script src="../../../jslib/jquery-1.6.js"></script>
<script src="js/desSlideshow.js"></script>     //添加图片插件文件
```

具体使用这个插件的功能代码如下：

```
<script language="javascript" type="text/javascript">
    $(function() {
        $("#desSlideshow1").desSlideshow({
            autoplay: 'enable',              //启用自动播放
            slideshow_width: '800',          //滑动窗口宽
            slideshow_height: '249',         //滑动窗口高
            thumbnail_width: '200',          //提示栏宽度
            time_Interval: '4000',           //图片切换时间间隔
            directory: 'images/'             //默认图片路径位置
        });
      $("#desSlideshow2").desSlideshow({
            autoplay: 'disable',             //禁用自动播放
            slideshow_width: '600',          //滑动窗口宽
          slideshow_height: '249',           //滑动窗口高
          thumbnail_width: '120',            //提示栏宽
          time_Interval: '4000',             //图片切换时间间隔
          directory: 'images/'               //默认图片路径位置
      });
  });
</script>
```

上述代码中第 3 行和第 11 行是调用插件核心函数，在调用过程中使用到了一些相关属性。autoplay 表示是否自动轮播图片动画，slideshow_width 表示插件占用的宽度，slideshow_height 表示插件占用的高度，thumbnail_width 表示图片右侧的提示栏宽度，time_Interval 表示自动轮播时的动画间隔，directory 表示右侧提示栏中出现的图片的路径，静态效果如图 8.19 和图 8.20 所示。

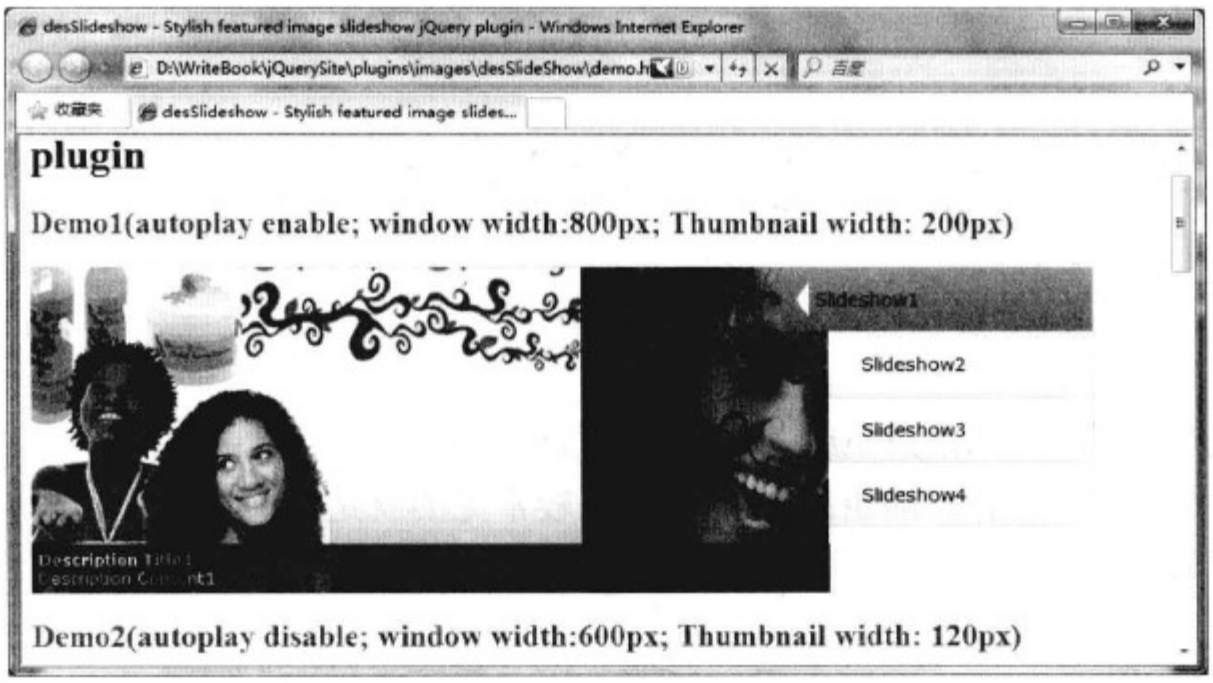


图 8.19　自动播放图片切换

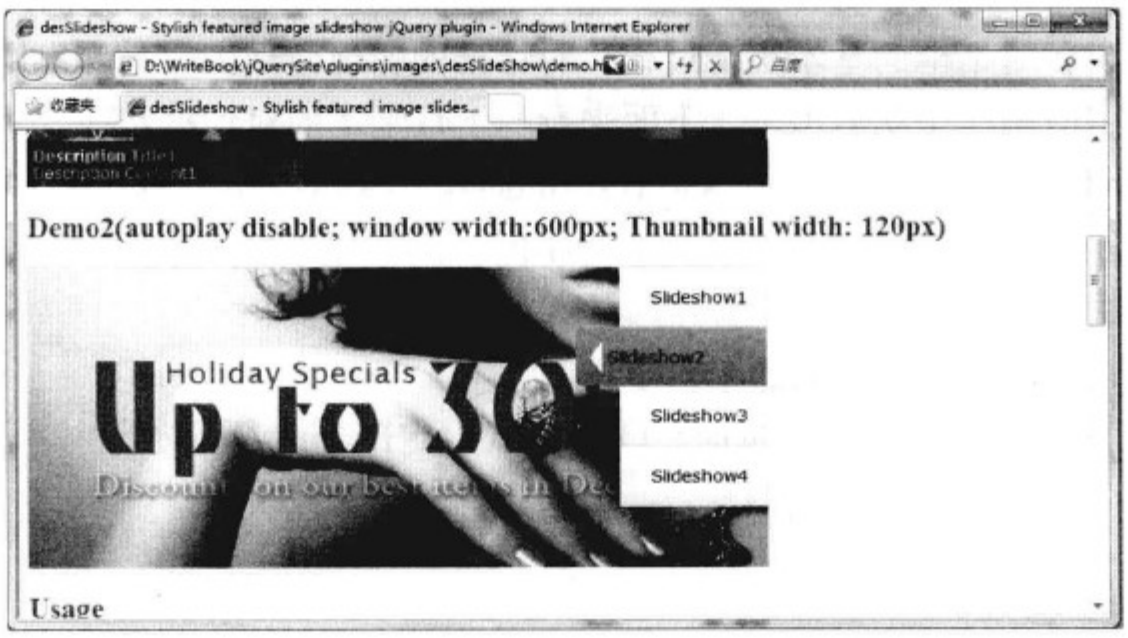


图 8.20　手动切换图片

8.9　小　　结

本章对图片的特效进行了讲解。主要内容包括图片切换、图片滚动、图片动态弹出、动态图文结合、图片剪切、图片预览效果、图片局部平移、图片插件。重点部分是图片切换、图片动态弹出、图片预览与平移，同时这一部分也是本章的难点。下一章将讲解 jQuery 对对话框的设计。

8.10　习　　题

【习题 1】练习掌握本章所介绍的 Fancybox 插件使用方法。

【习题 2】练习掌握本章所介绍的 desSlideshow 插件使用方法。

第 9 章　设计对话框

Ajax 的出现给 B/S 结构的应用程序注入了新的活力，因为它能在无刷新页面的情况下得到新的数据内容。富客户端也就是 RIA 中比较常见的就是页面对话框。本章主要讲解通过 jQuery 实现对话框的设计原理及主要对话框插件的使用。

9.1　设计基本对话框

本节主要讲解对话框的基本实现过程。对话框的实现原理主要是弹出层操作，当需要触发对话框时将隐藏的层显示出来。下面来看一下最基本的对话框实现。其中，要使用到 jQuery 函数 ready()、click()、css()、show()、hide()。其功能实现如下。

（1）为页面中触发弹出对话框的元素添加单击事件，在事件中触发对话框所在层显示，并定位对话框的具体显示位置。

（2）为对话框添加触发隐藏的单击事件。

首先，利用 HTML 创建静态页面。CSS 样式设定部分请参考光盘内容。

```
<div id="btn">弹出对话框</div>
 <div id="showMessage">
     <div id="titlearea">
         <div id="close"></div>
     <div id="titles">jQuery 对话框</div>
     </div>
     <div id="content">利用层的弹出产生对话框，这是一个最简单的对话框。</div>
 </div>
```

然后，加入 jQuery 库文件：

```
<script type="text/JavaScript" src="jslib/jquery.js"></script>
```

最后，加入 JavaScript 代码：

```
<script type="text/JavaScript">
    $(document).ready(function(){
        $("#btn").click(function(){     //触发显示对话框事件
            //定位并显示对话框
            $("#showMessage").css("left",($(window).width()-$
            ("#showMessage").width())/2+"px")
            .css("top",($(window).height()-$("#showMessage").
            height())/2+"px").show();});
        $("#close").click(function(){   //隐藏对话框事件
```

```
7                $("#showMessage").hide();
8            });
9        });
10   </script>
```

上述代码中第 3 行是触发对话框显示的事件，第 4、5 行是定位对话框显示在整个页面的中间位置，第 6、7 行是对话框隐藏功能。效果如图 9.1 所示。

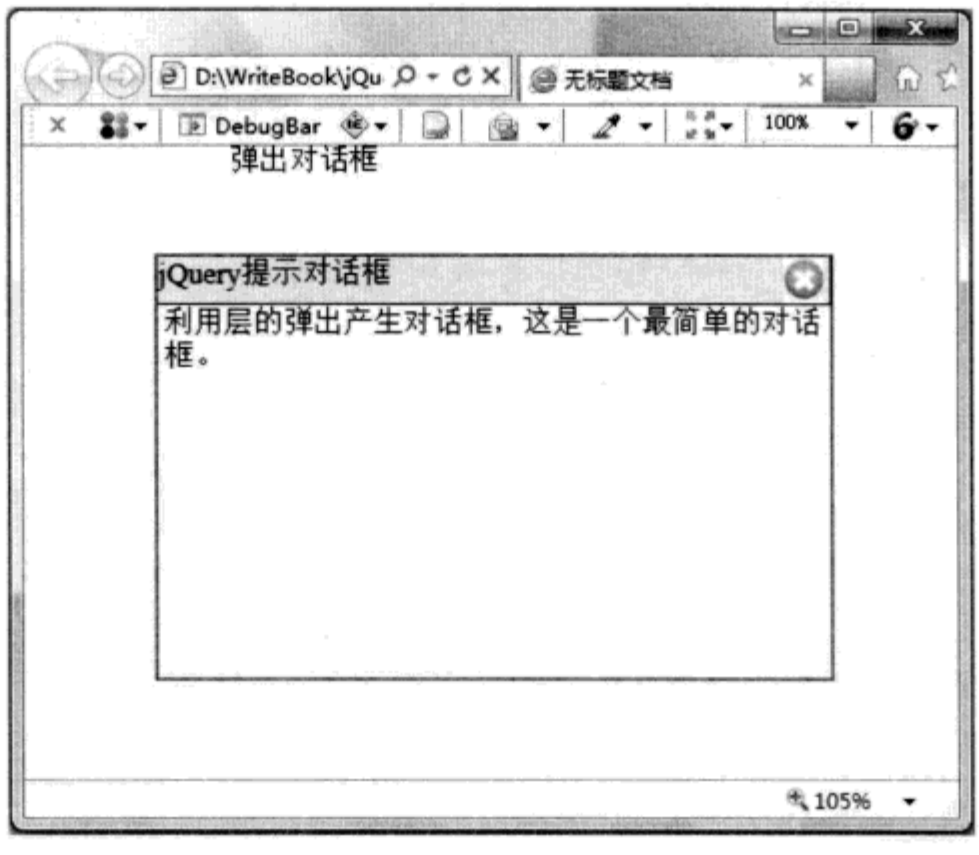


图 9.1　基本对话框

9.2　模式对话框

通过上一节所讲解的示例我们初步认识了利用 jQuery 创建一个对话框的基本步骤。但是，上一节所讲解的示例是有缺陷的，也就是当对话框出现的时候，触发对话框产生的层还是可以被单击，这样容易给用户产生错觉，认为能够产生多个对话框。对于这个问题，可以对该示例进行加工，创建一个模式对话框。

模式对话框的实现原理是：在触发对话框的同时创建一个所谓的遮盖层。遮盖层覆盖在页面的所有内容之上（除了对话框），但是遮盖层需要设定一定的透明度。其中，要使用到 jQuery 函数 ready()、click()、css()、show()、hide()。

其功能实现如下。

（1）为页面中触发弹出对话框的元素添加单击事件，在事件中触发对话框所在层显示，并定位对话框的具体显示位置。

（2）在触发对话框事件中显示遮盖层。

（3）为对话框添加触发隐藏的单击事件，同时隐藏遮盖层。

首先，在上一节介绍的示例的基础上添加遮盖层：

```
<div id="background"></div>
```

然后，修改 JavaScript 功能代码：

```
1   <script type="text/JavaScript">
2       $(document).ready(function(){
3           $("#btn").click(function(){          //触发显示对话框及遮盖层
                //遮盖层显示
4               $("#background").css("width",$(window).width()+"px")
5                   .css("height",$(window).height()+"px").show();
                //对话框显示并定位
6               $("#showMessage").css("left",($(window).width()-$
                ("#showMessage").width())/2+"px")
7               .css("top",($(window).height()-$("#showMessage").
                height())/2+"px").show();});
8               $("#close").click(function(){        //隐藏对话框和遮盖层
9               $("#background").css("display","none");
10              $("#showMessage").hide();
11          });
12      });
13  </script>
```

上述代码中第 4、5 行显示遮盖层并设定遮盖层的高和宽，第 9 行隐藏遮盖层，效果如图 9.2 所示。

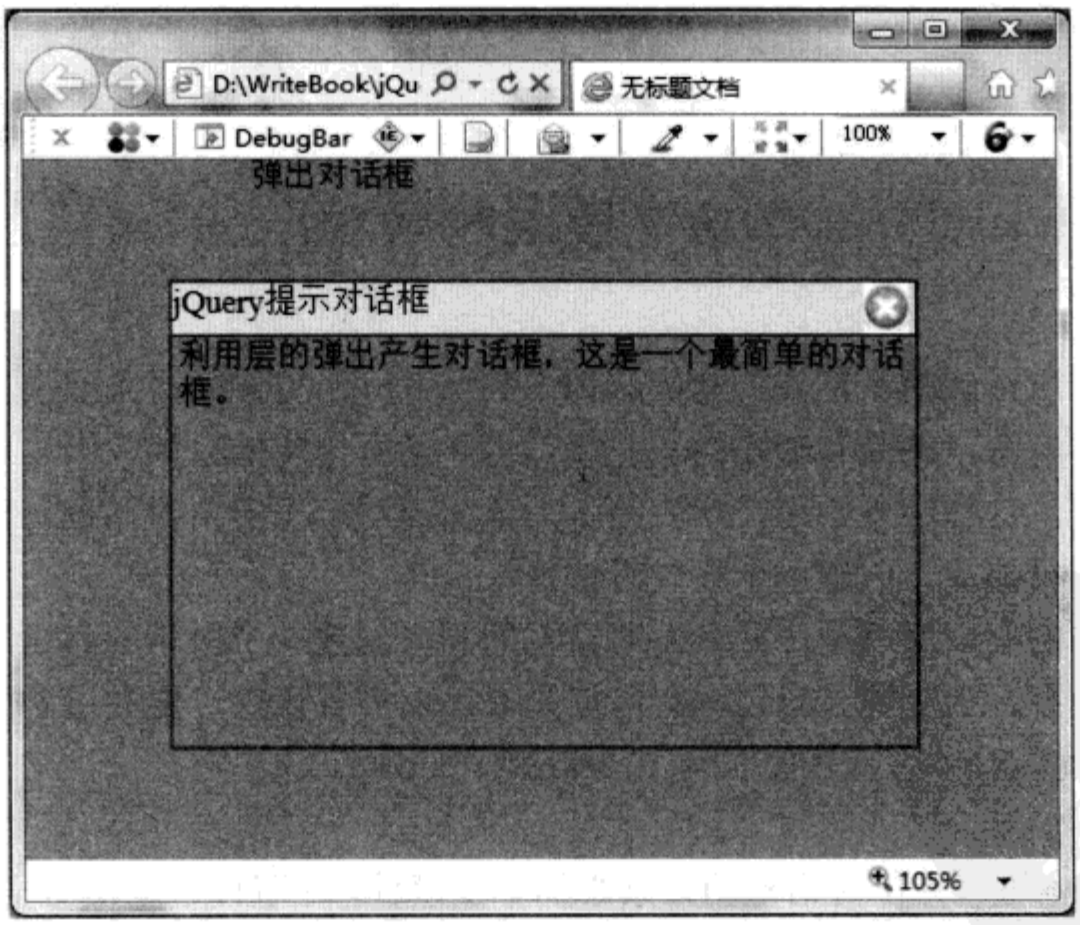


图 9.2　模式对话框

这个对话框显示时，页面上触发对话框的层是不可单击的。单击对话框右上角关闭图标，对话框关闭，页面上触发对话框的层又可以单击。

9.3　输入对话框

前面在介绍 JavaScript 的时候，曾介绍过它的三个基本对话框 alert()、confirm()、prompt()。它们分别表示提示对话框、确认对话框、输入对话框。从本节开始介绍如何通过 jQuery 来模拟创建这三个对话框。本节创建输入对话框，其实现原理是：利用弹出层，在层上添加用户输入功能，并在层隐藏后将用户输入内容显示在页面上。其中，使用到了 jQuery 函数 ready()、click()、css()、show()、hide()、text()、val()。

其功能实现如下。

（1）为页面中触发弹出对话框的元素添加单击事件，在事件中触发对话框所在层显示，并定位对话框的具体显示位置。

（2）在触发对话框事件中显示遮盖层。

（3）为对话框添加触发隐藏的单击事件，同时隐藏遮盖层。

（4）为对话框的内容提交按钮添加隐藏对话框和遮盖层功能，并在页面中显示用户输入的内容。

首先，对上一节的对话框中的样式进行修改：

```
<div id="btn">弹出对话框</div>
<p id="result">对话框用户输入内容：</p>
<div id="showMessage">
    <div id="titlearea">
        <div id="close"></div>
    <div id="titles">jQuerys 输入对话框</div>
    </div>
    <div id="content"><input id="userdata" /><input type="button"
    id="tijiao" value="确定" /></div>
</div>
<div id="background"></div>
```

然后，修改 JavaScript 功能代码：

```
<script type="text/JavaScript">
    $(document).ready(function(){
        $("#btn").click(function(){     //触发输入对话框
        //显示遮盖层
            $("#background").css("width",$(window).width()+"px")
                .css("height",$(window).height()+"px").show();
        //显示输入对话框
            $("#showMessage").css("left",($(window).width()-$
            ("#showMessage").width())/2+"px")
            .css("top",($(window).height()-$("#showMessage").
            height())/2+"px").show();});
        //隐藏遮盖层和对话框
        $("#close").click(function(){
        $("#background").css("display","none");
        $("#showMessage").hide();
```

```
11        });
          //隐藏遮盖层和对话框，在页面上显示输入内容
12        $("#tijiao").click(function(){
13            $("#background").css("display","none");
14            $("#showMessage").hide();
15            $("#result").text($("#result").text()+$("#userdata").val());
16        });
17    });
18    </script>
```

上述代码第 12～16 行是对话框中内容提交按钮的单击事件，在这个事件中隐藏遮盖层，隐藏对话框，并将用户在输入对话框中输入的内容在页面中显示出来，效果如图 9.3 和图 9.4 所示。

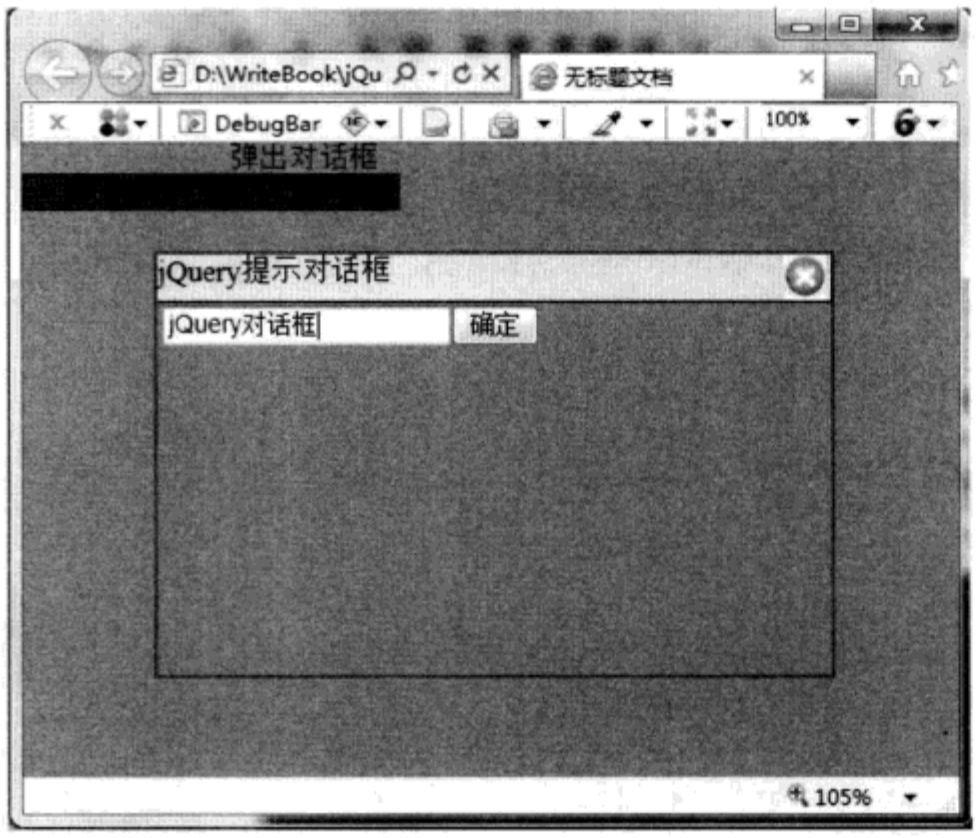


图 9.3　jQuery 输入对话框

图 9.4　jQuery 输入对话框返回内容

9.4　提示对话框

本节将介绍提示对话框。提示对话框主要显示提示信息。它的基本工作原理和 9.2 节所介绍的模式对话框类似，当用户输入相应内容并触发对话框弹出时，自动将用户的内容携带进对话框。其中，要使用到 jQuery 函数 ready()、click()、css()、show()、hide()、attr()。

其功能实现如下。

（1）为页面中触发弹出对话框的元素添加单击事件，在事件中触发对话框所在层显示，并定位对话框的具体显示位置，将用户输入内容填入对话框。

（2）在触发对话框事件中显示遮盖层。

（3）为对话框添加触发隐藏的单击事件，同时隐藏遮盖层。

首先，对上一节的对话框中的样式进行修改：

```
<div id="btn">弹出对话框<br />
<input type="checkbox" id="public" value="0" name="sss" name="public"/>公开资料内容
</div>
<!--<p id="result">对话框用户输入内容：</p>-->
<div id="showMessage">
    <div id="titlearea">
        <div id="close"></div>
    <div id="titles">jQuery 提示对话框</div>
    </div>
    <div id="content"><span id="message">您选择了</span><input type="button" id="tijiao" value="确定" /></div>
 </div>
<div id="background"></div>
```

然后，修改 JavaScript 功能代码：

```
<script type="text/JavaScript">
    $(document).ready(function(){
        var msg=undefined;
        $("#btn").click(function(){
                //弹出提示对话框和遮盖层，在提示对话框中显示用户提交内容
            if($("#public").attr("checked"))
                msg="公开资料内容";
            else
                msg="不公开资料内容";
            $("#background").css("width",$(window).width()+"px")
                .css("height",$(window).height()+"px").show();
            $("#showMessage").css("left",($(window).width()-$("#showMessage").width())/2+"px")
            .css("top",($(window).height()-$("#showMessage").height())/2+"px").show();
            $("#message").text($("#message").text()+msg);
        });
        $("#close").click(function(){       //隐藏对话框和遮盖层
            $("#background").css("display","none");
            $("#showMessage").hide();
        });
        $("#tijiao").click(function(){      //隐藏对话框和遮盖层
        $("#background").css("display","none");
        $("#showMessage").hide();
        msg="";
        $("#message").text("您选择了");
        //$("#result").text($("#result").text()+$("#userdata").val());
    });
});
</script>
```

上述代码第 5～8 行提取了用户在页面上的选择内容，并记录下来。第 13 行将用户的选择内容填入对话框，提示用户操作。第 22、23 行，当对话框隐藏的时候将其内容进行重

新设置，以便下次使用，效果如图 9.5 和图 9.6 所示。

图 9.5　用户输入选择内容

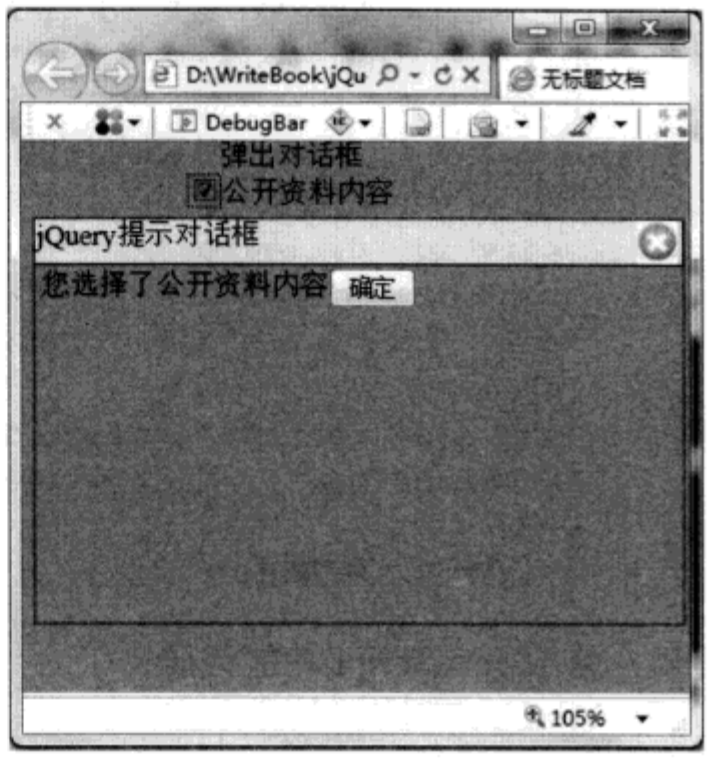


图 9.6　提示对话框

9.5　确认对话框

前面讨论了输入对话框和信息提示对话框。下面来看确认对话框如何实现。确认对话框和前两节介绍的对话框的不同在于，它是一种选择性操作对话框。确认对话框为用户提供两种选择结果，并且用户的选择会直接影响页面上的操作。

它的实现原理是：从页面提取用户操作内容在对话框显示时，提示用户是否确认自己的操作。当用户对对话框做出确认或者否定操作时，又返回来直接修改页面内容。其中，使用到了 jQuery 函数 ready()、click()、css()、show()、hide()、attr()。

其功能实现如下。

（1）为页面中触发弹出对话框的元素添加单击事件，在事件中触发对话框所在层显示，并定位对话框的具体显示位置，将用户输入内容填入对话框。

（2）在触发对话框事件中显示遮盖层。

（3）为对话框的确认按钮添加触发隐藏的单击事件，同时隐藏遮盖层。

（4）为对话框的取消按钮添加触发隐藏的单击事件，同时隐藏遮盖层。如果用户勾选了页面内容，则取消勾选操作。

首先，对上一节的对话框中的样式进行修改：

```
1    <div id="btn">弹出对话框<br />
2    <input type="checkbox" id="public" value="0" name="sss" name=
     "public"/>公开资料内容
3    </div>
4    <!--<p id="result">对话框用户输入内容：</p>-->
5    <div id="showMessage">
6        <div id="titlearea">
```

```
7          <div id="close"></div>
8        <div id="titles">jQuery 提示对话框</div>
9        </div>
10       <div id="content"><span id="message">您选择了</span><input type=
         "button" id="tijiao" value="确定" />
11        <input type="button" id="cancel" value="取消" />
12        </div>
13    </div>
14  <div id="background"></div>
```

然后修改 JavaScript 功能代码：

```
1   <script type="text/JavaScript">
2       $(document).ready(function(){
3           var msg=undefined;
4           /*$("#btn").click(function(){
5               $("#background").css("width",$(window).width()+"px")
6                   .css("height",$(window).height()+"px").show();
7               $("#showMessage").css("left",($(window).width()-$
                ("#showMessage").width())/2+"px")
8               .css("top",($(window).height()-$("#showMessage").
                height())/2+"px").show();}); */
9           $("#btn").click(function(){
                        //弹出提示对话框和遮盖层，在提示对话框中显示用户提交内容
10              if($("#public").attr("checked"))
11                  msg="公开资料内容";
12              else
13                  msg="不公开资料内容";
14              $("#background").css("width",$(window).width()+"px")
15                  .css("height",$(window).height()+"px").show();
16              $("#showMessage").css("left",($(window).width()-$
                ("#showMessage").width())/2+"px")
17              .css("top",($(window).height()-$("#showMessage").
                height())/2+"px").show();
18              $("#message").text($("#message").text()+msg);
19          });
20          $("#close").click(function(){       //隐藏对话框和遮盖层
21              $("#background").css("display","none");
22              $("#showMessage").hide();
23          });
24          $("#tijiao").click(function(){
                                        //隐藏对话框和遮盖层，并确定用户选择
25              $("#background").css("display","none");
26              $("#showMessage").hide();
27              msg="";
28              $("#message").text("您选择了");
29              //$("#result").text($("#result").text()+$("#userdata").
                val());
30          });
31          $("#cancel").click(function(){//隐藏对话框和遮盖层，并取消用户选择
32              $("#background").css("display","none");
33              $("#showMessage").hide();
```

```
34                msg="";
35                $("#message").text("您选择了");
36                if($("#public").attr("checked"))
37                    $("#public").attr('checked',false);
38            });
39        });
40    </script>
```

上述代码的基本内容和 9.4 节相同，第 31～38 行是确认对话框的取消按钮功能，第 36、37 行判断用户是否勾选了相应内容，如果勾选了，则取消勾选。效果如图 9.7 所示。

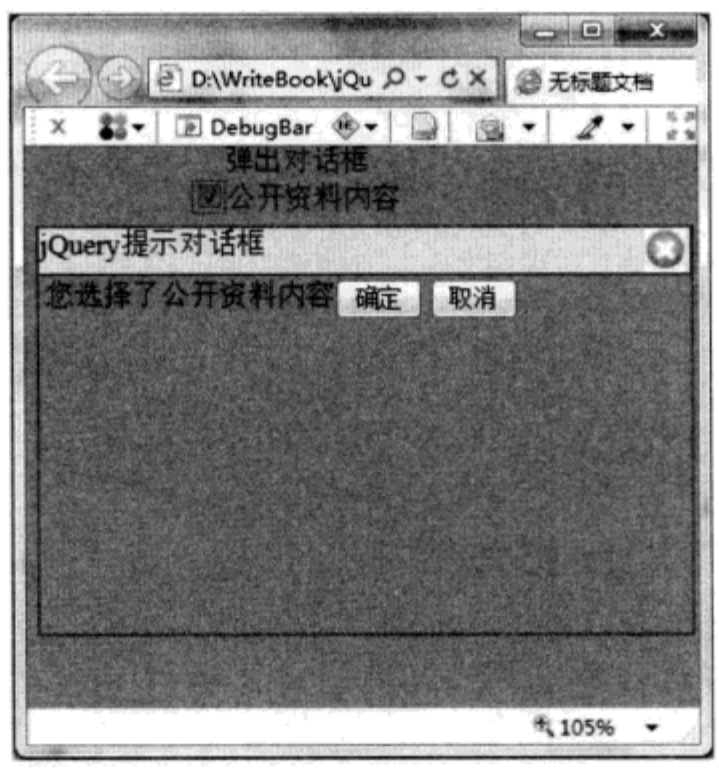


图 9.7　确认对话框

从图 9.7 中可以看到，当勾选复选框后弹出确认对话框。但是，如果尝试单击“取消”按钮，对话框会消失，同时复选框又处于未选中状态。如果单击“确定”按钮，则只是对话框消失，并不影响复选框。

9.6　对话框插件

对话框插件是一种在网页页面上模拟跳出模式或者非模式对话框的插件。它可以帮助我们完成更多的用户交互操作，并节省开发自定义对话框的时间。本节介绍两个对话框插件。它们的基本功能和前面几节介绍的自定义对话框类似。

9.6.1　jqModal 对话框插件

jqModal 对话框插件帮助我们在浏览器中显示告示、对话框、模式窗口。它富有弹性，小巧，就如同“瑞士军刀”一样。它的英文介绍网址为 http://dev.iceburg.net/jquery/jqModal/。

jqModal 特点如下。

（1）设计友好，不用复杂的操作。

（2）翻译友好，不用硬编码英文字符串。
（3）开发友好，利用回调函数实现多功能。
（4）支持远程加载内容。
（5）多模块支持。

jqModal 基本使用步骤如下。

（1）添加对话框容器。
（2）初始化对话框。
（3）触发对话框。

jqModal 中使用的函数如表 9.1 所示。

表 9.1　jqModal函数说明

函　数	例　子	说　明
jqm	$('#dialog').jqm(); $('.dialogs').jqm({ajax:'@href',modal:true});	jqModal 对话框的初始化函数，接收一组参数对象
jqmShow	$('.dialogs').jqmShow();	显示对话框
jqmHide	$('.dialogs').jqmHide();	隐藏对话框
jqmAddTrigger	$('.dialogs').jqmAddTrigger($('#openDialogs a'));	单击触发对话框的页面元素，可以为字符串形式的 jQuery 选择器，也可以为 jQuery 对象，或者一个 DOM 元素
jqmAddClose	$('.dialogs').jqmAddClose($('#hideDialogs a'));	单击关闭对话框的页面元素，可以为字符串形式的 jQuery 选择器，也可以为 jQuery 对象，或者一个 DOM 元素

jqModal 中使用到的参数说明如表 9.2 所示。

表 9.2　jqModal的参数说明

参　数	数 据 类 型	默　认　值	说　明
overlay	整型	50	覆盖层的透明度，按照百分比取值
overlayClass	字符串	'jqmOverlay'	覆盖层的 CSS 样式类
closeClass	字符串或者为假	'jqmClose'	单击关闭对话框元素的 CSS 样式类
trigger	字符串、对象、或者为假	'.jqModal'	单击触发对话框的页面元素，可以为字符串形式的 jQuery 选择器，或者一个 DOM 元素，或者不使用触发元素
ajax	字符串或者假	false	指定需要加载远程内容的地址
ajaxText	字符串	''	Ajax 文本内容
target	字符串或者假	false	Ajax 内容加载目标元素，可以为字符串形式的 jQuery 选择器，或者一个 DOM 元素，或者 Ajax 将重写整个对话框内容
modal	布尔型	false	模式对话框参数
toTop	布尔型	false	将对话框置于所有其他元素的上层

下面通过几个示例来了解这个插件的使用方法，这里不涉及 Ajax 内容。

1. 默认对话框形式

这种对话框的使用完全套用了所有属性为默认值的情况。首先，需要建立静态页面：

```
1   <a href="#" class="jqModal">view</a>
2   ...
3   <div class="jqmWindow" id="dialog">
4   <a href="#" class="jqmClose">Close</a>
5   <hr>
6   <em>READ ME</em> -->
7   This is a "vanilla plain" jqModal window. Behavior and appeareance extend
    far beyond this.
8   The demonstrations on this page will show off a few possibilites.
    I recommend walking
9   through each one to get an understanding of jqModal <em>before</em> using it.
10   <br /><br />
11   You can view the sourcecode of examples by clicking the JavaScript,
    CSS, and HTML tabs.
12   Be sure to checkout the <a href="README">documentation</a> too!
13   <br /><br />
14   <em>NOTE</em>; You can close windows by clicking the tinted background
    known as the "overlay".
15   Clicking the overlay will have no effect if the "modal" parameter is
    passed, or if the
16   overlay is disabled.
17   </div>
```

上面的 HTML 代码第 1、2 行是触发对话框的页面元素；第 3～17 行是对话框的内容部分。其中“jqmWindow”为对话框的容器层，第 4 行是对话框的关闭元素。然后在<head>中加入 CSS 样式设定：

```
1   <style type="text/css">
2    .jqmWindow {
3       display: none;
4       position: fixed;
5       top: 17%;
6       left: 50%;
7       margin-left: -300px;
8       width: 600px;
9       background-color: #EEE;
10       color: #333;
11       border: 1px solid black;
12       padding: 12px;
13    }
14    .jqmOverlay { background-color: #000; }
15    * html .jqmWindow {
16         position: absolute;
17                top: expression((document.documentElement.scrollTop ||
                  document.body.scrollTop) + Math.round(17 * (document.
                  documentElement.offse-tHeight || document.body.
                  clientHeight) / 100) + 'px');
```

```
18    }
19    </style>
```

第 2～13 行，第 15～18 行是对话框样式的设定，第 14 行是覆盖层的样式。最后引入 jQuery 库文件及 jqModal 插件文件，并加入 JavaScript 代码：

```
1    <script type="text/JavaScript" src="../../jslib/jquery.js"></script>
2    <script type="text/JavaScript" src="JS/jqModal.js"></script>
3    <script type="text/JavaScript">
4        $().ready(function() {
5          $('#dialog').jqm();
6        });
7    </script>
```

代码相对简单，没有任何参数设定，效果如图 9.8 所示。

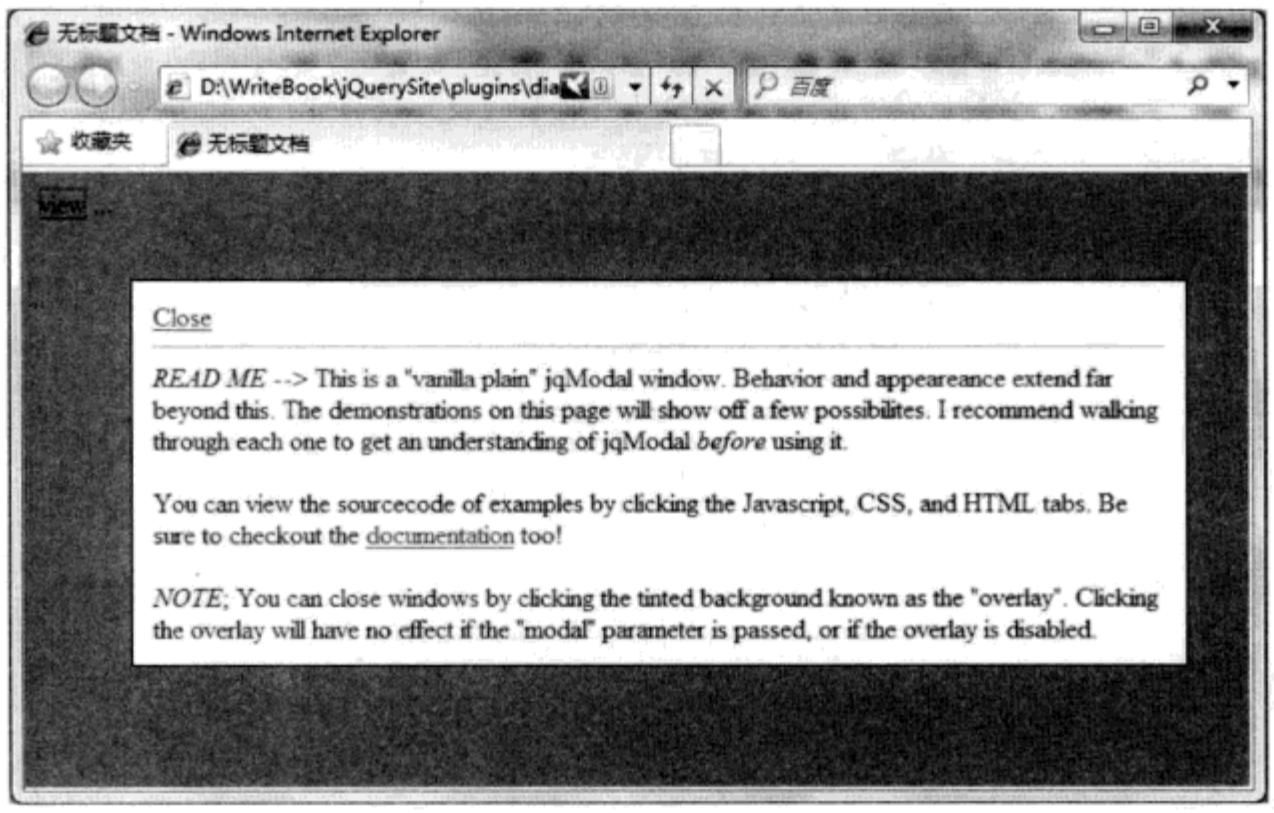


图 9.8　jqModal 默认对话框

2. 可拖动的带特殊背景的模式对话框

这种对话框使用了 jqModal 的一部分属性，并加入了对话框的拖动功能。

首先，建立静态页面。静态页面及 CSS 样式与上面的例子基本相同，只是加入了若干样式设定，具体内容请参考光盘内容。下面看一下具体 JavaScript 功能代码。

```
1    <script type="text/JavaScript">
2        $().ready(function() {
3          $('#ex3a').jqm({
4            trigger: '#ex3aTrigger',
5            overlay: 30, /* 0-100 (int) : 0 is off/transparent, 100 is opaque */
6            overlayClass: 'whiteOverlay'})
7            .jqDrag('.jqDrag'); /* make dialog draggable, assign handle
             to title */
8          $('input.jqmdX')
9          .hover(
```

```
10          function(){ $(this).addClass('jqmdXFocus'); },
11          function(){ $(this).removeClass('jqmdXFocus'); })
12        .focus(
13          function(){ this.hideFocus=true; $(this).addClass
            ('jqmdXFocus'); })
14        .blur(
15          function(){ $(this).removeClass('jqmdXFocus'); });
16      });
17    </script>
```

上述代码中第 3～7 行是 jqModal 插件的基本参数配置，第 4 行表示设定触发元素，第 5 行设定覆盖层的透明度，第 6 行是设定覆盖层的样式类，第 7 行设定对话框拖动时的样式；第 8～15 行是对话框关闭按钮的各种事件的样式设定，包括鼠标悬停、得到焦点、失去焦点，效果如图 9.9 和图 9.10 所示。

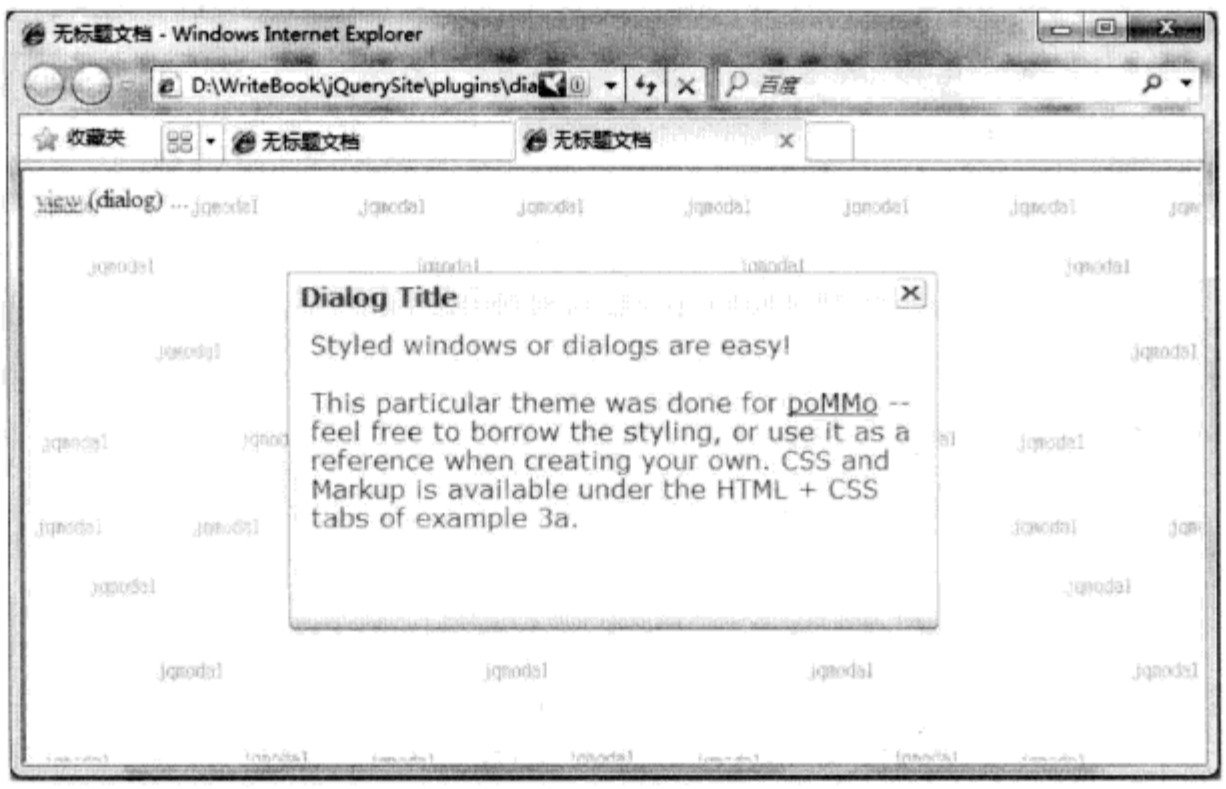


图 9.9　可拖动对话框

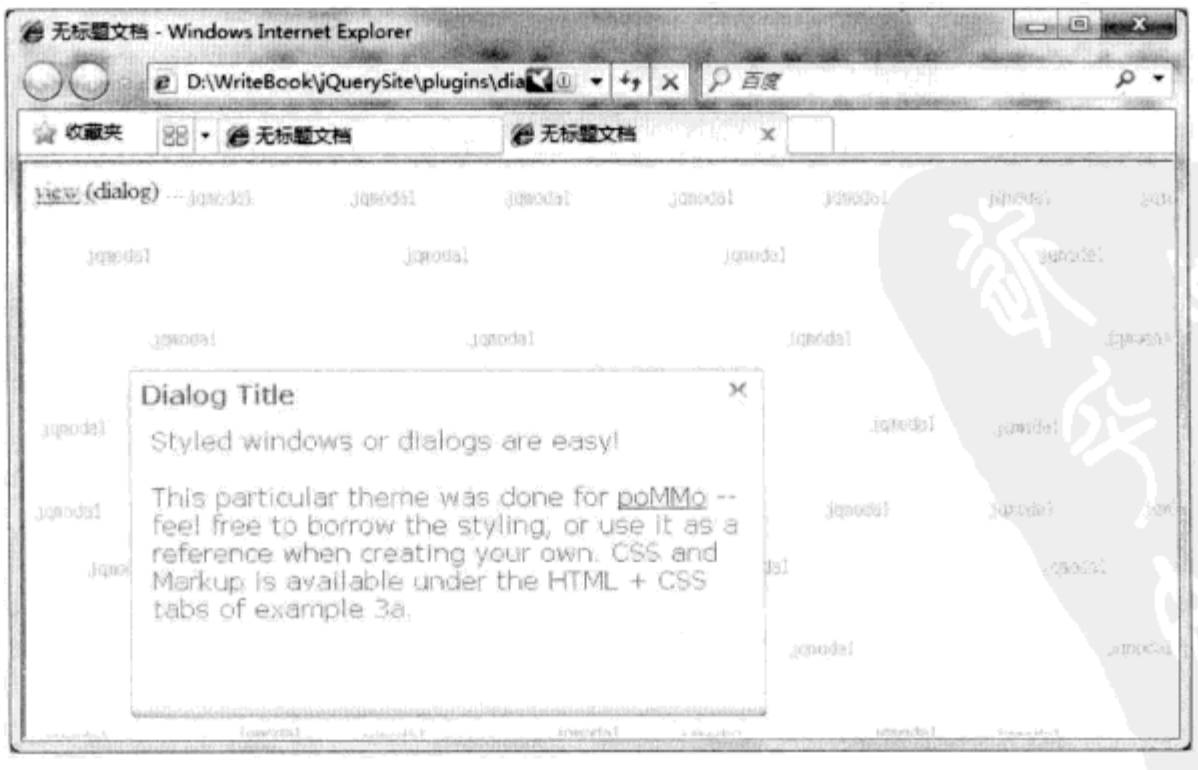


图 9.10　对话框拖动效果

3. jqModal的显示与隐藏

这个例子使用了插件的显示与隐藏方法控制弹出对话框层，同时使用了添加触发对话框的元素和关闭对话框的元素的方法。静态页面和 CSS 代码请参考光盘内容。下面主要讨论一下 JavaScript 的功能代码。

```
1   $().ready(function() {
2     var show = $('#ex5show');
3     var hide = $('#ex5hide');
4     $('div.square')
5       .jqm({overlay: 0, trigger: false})
6       .jqDrag()                    //可拖动操作
7       .jqmAddTrigger(show)
8       .jqmAddClose(hide)
9       $('#ex5multi').click(function() {
10      $('div.square:even').jqmShow();
11      $('div.square:odd').jqmHide();
12      return false;
13    });
14  });
```

上述代码中第 4、5 行进行插件初始化属性设定，覆盖层透明度为 0，不使用默认触发功能。第 7 行添加触发对话框的页面元素。第 8 行添加触发关闭对话框的页面元素。第 9～13 行设定了弹出层的显示效果，奇数层隐藏，偶数层显示。效果如图 9.11 和图 9.12 所示。

这个插件的功能较多，这里挑选了上面三种基本使用方式，其他功能可以参考光盘内容或者去插件的官网查看范例。

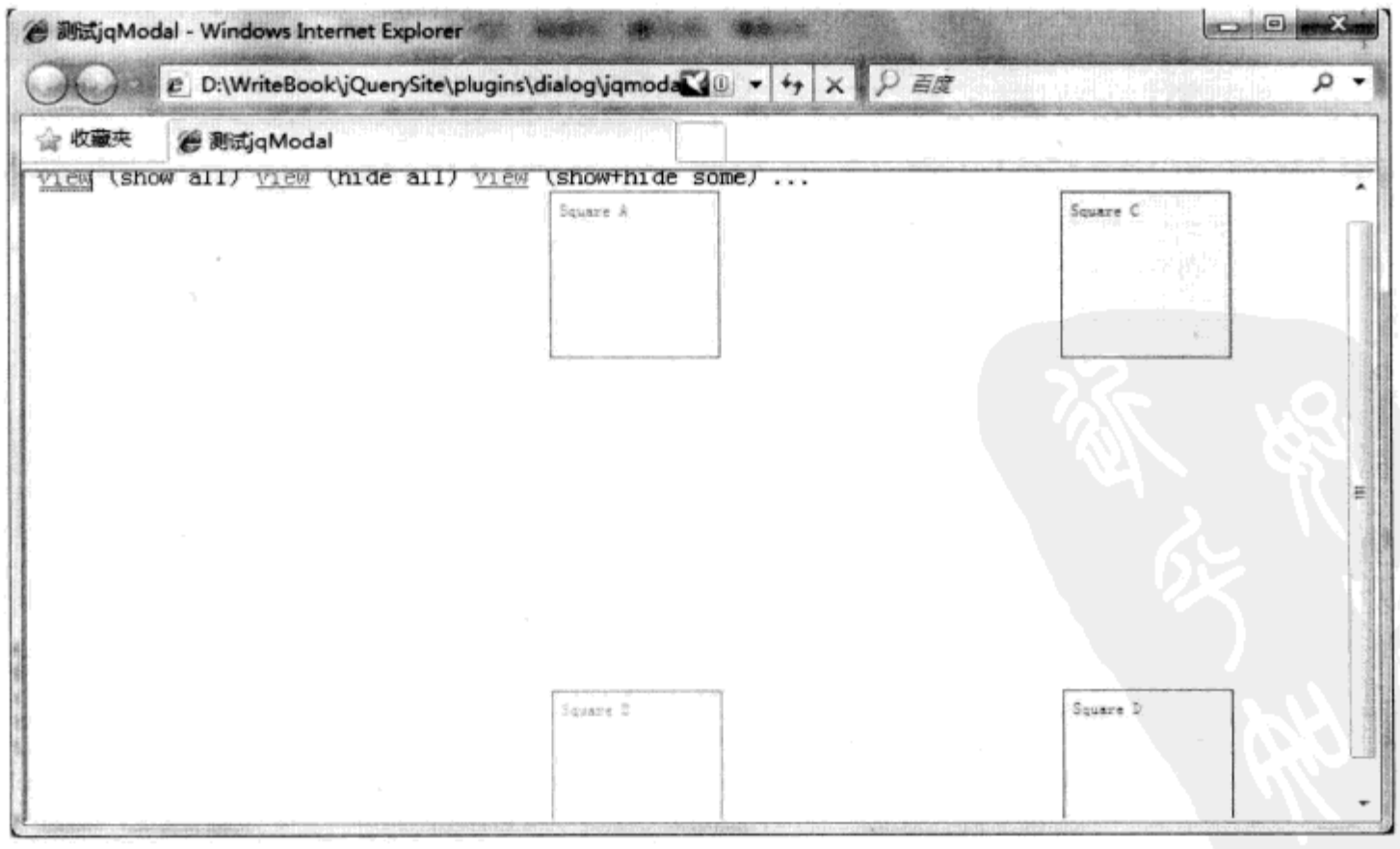


图 9.11　触发显示所有弹出层

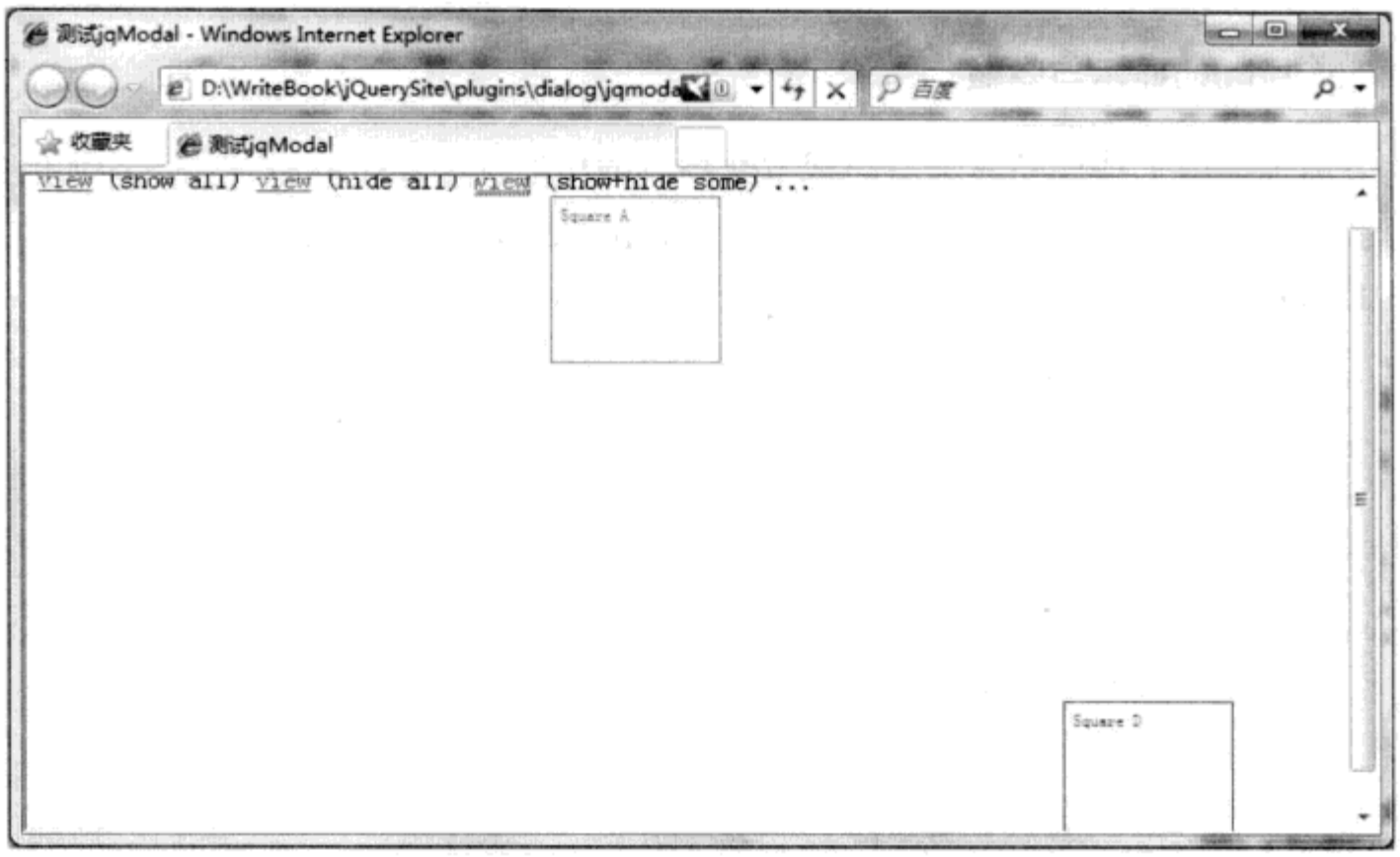


图 9.12　显示部分弹出层

9.6.2　jQuery.UI.Dialog 对话框插件

jQuery.UI.Dialog 对话框插件是 jQuery 官方网站提供的一整套 UI 插件中的一部分。它的主要功能是利用 jQuery 生成动态浮动对话框。这个插件需要以下 jQuery 文件的支持：jquery.ui.core.js、jquery.ui.widget.js、jquery.ui.position.js、jquery.ui.mouse.js（可选，当需要通过鼠标拖动对话框或者改变对话框的大小时使用）、jquery.ui.draggable.js（可选）、jquery.ui.resizable.js。

这个插件有一些相关属性，可以设置对话框的样式和行为，如表 9.3 所示。

表 9.3　Dialog属性说明

属　性	类　型	默认值	说　明
disable	布尔	false	对话框的启动选项，可以在对话框初始化时加载
autoOpen	布尔	true	当对话框被调用的时候自动打开。如果打开失败，则对话框总是处于隐藏状态。dialog("open")表示调用对话框
buttons	对象	{}	指定显示按钮选择。这个属性的键为按钮的文本，值为按钮的单击事件回调函数
buttons	数组	[]	指定显示按钮的选择。数组中的每个元素都必须针对按钮定义
closeOnEscape	布尔	true	设定对话框是否在得到焦点时可以通过 Esc 键关闭
closeText	字符串	close	指定关闭按钮的文本内容
dialogClass	字符串	''	指定初始化对话框的对话框类
draggable	布尔	true	设定对话框是否可以拖动
height	数字	'auto'	对话框的高度，单位为像素
hide	字符串，对象	null	当对话框关闭后的效果
maxHeight	数字	false	对话框可调整的最大高度值，单位为像素
maxWidth	数字	false	对话框可调整的最大宽度值，单位为像素

续表

属　性	类　型	默认值	说　明
minHeight	数字	150	对话框可调整的最小高度值，单位为像素
minWidth	数字	150	对话框可调整的最小宽度值，单位为像素
modal	布尔	false	模式对话框选项
position	字符串，数组	'center'	指定对话框显示的位置：1，单一字符串表示位置'center', 'left', 'right', 'top', 'bottom'；2，两个数字组合数组，单位为像素；3，两个字符串组合的数组
resizable	布尔	true	对话框是否可调整大小选项
show	字符串，对象	null	当对话框打开时的效果
stack	布尔	true	指定当对话框获得焦点时是否可以停靠在多个对话框的最上层
title	字符串	''	对话框的标题属性
width	数字	300	对话框的宽度，单位为像素
zIndex	整型	1000	对话框的初始层位置

与这个对话框相关的还有事件选项，如表 9.4 所示。

表 9.4　Dialog事件说明

事　件	类　型	说　明
create	dialogcreate	对话框创建时触发
beforeClose	dialogbeforeclose	对话框尝试关闭时触发，如果关闭失败，将阻止对话框关闭
open	dialogopen	对话框被打开后触发
focus	dialogfocus	对话框得到焦点时触发
dragStart	dialogdragstart	对话框开始被拖动时触发
drag	dialogdrag	对话框被拖动时触发
dragStop	dialogdragstop	对话框停止拖动时触发
resizeStart	dialogresizestart	对话框开始改变大小时触发
resize	dialogresize	对话框改变大小时触发
resizeStop	dialogresizestop	对话框停止改变大小时触发
close	dialogclose	对话框关闭后触发

有一些行为可以控制该对话框的相关操作，如表 9.5 所示。

表 9.5　Dialog行为说明

行　为	表　示	说　明
destroy	.dialog("destroy")	移除所有对话框的功能
disable	.dialog("disable")	屏蔽对话框
enable	.dialog("disable")	使能对话框
option	.dialog("option" , optionName , [value])	获取或设置对话框参数选项
option	.dialog("option" , options)	设置多个对话框选项
widget	.dialog("widget")	返回对话框元素
close	.dialog("close")	关闭对话框
isOpen	.dialog("isOpen")	如果对话框已经关闭则返回真
moveToTop	.dialog("moveToTop")	移动对话框到页面最上层
open	.dialog("open")	打开对话框

下面通过示例来介绍这个插件的使用方法。

1. 默认对话框

这个示例使用 Dialog 的默认参数形式，只调用了 Dialog 插件的初始化函数。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   <script type="text/JavaScript">
2       $(function() {
3           $( "#dialog" ).dialog();
4       });
5   </script>
```

第 3 行中的.dialog()就是 Dialog 插件的初始化函数。它使用了 Dialog 的默认样式，并在页面加载后自动显示对话框，效果如图 9.13 所示。

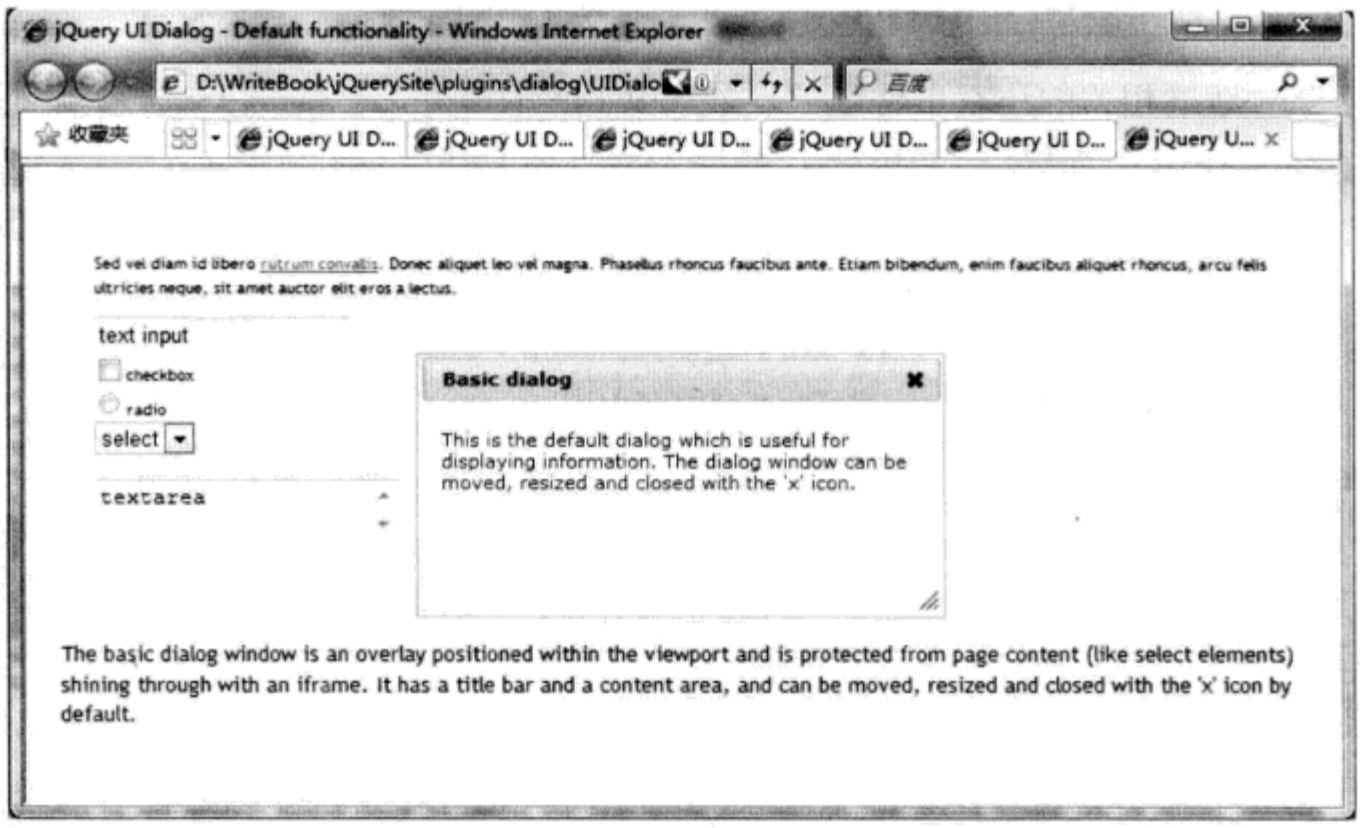


图 9.13　Dialog 插件默认样式

2. 动画效果对话框

这个示例使用了关于 Dialog 打开与关闭的属性设置和相关方法调用，实现 Dialog 动画效果打开和关闭。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   <script>
2   //increase the default animation speed to exaggerate the effect
3       $.fx.speeds._default = 1000;
4       $(function() {
5           $( "#dialog" ).dialog({
6               autoOpen: false,    //不自动打开对话框
7               show: "blind",      //显示样式设定
8               hide: "explode"     //隐藏样式设定
```

```
9           });
10          $( "#opener" ).click(function() {
11              $( "#dialog" ).dialog( "open" );
12              return false;
13          });
14      });
15  </script>
```

上述代码中第 6 行设定对话框不自动打开，第 7 行指定对话框打开时的动画效果为 blind，这个动画效果在 jQuery.effects.blind.js 中定义。第 8 行指定对话框关闭时的动画效果为 explode，这个动画效果在 jQuery.effects. explode.js 中定义。第 10～13 行设定了打开按钮的单击事件，第 11 行使用了 Dialog 的打开行为，具体效果如图 9.14 和图 9.15 所示。当单击 Open Dialog 按钮时，对话框动画渐入打开，当关闭对话框时，对话框分解成 6 个小部分逐渐消失。

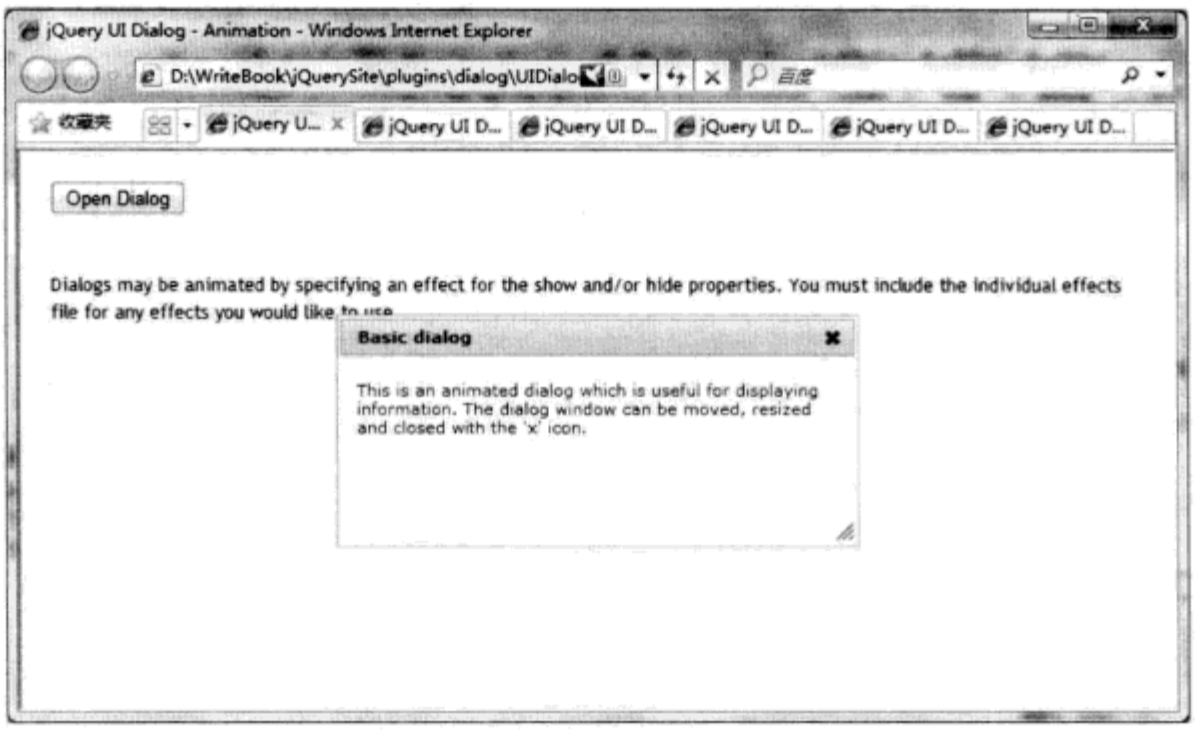


图 9.14　对话框打开

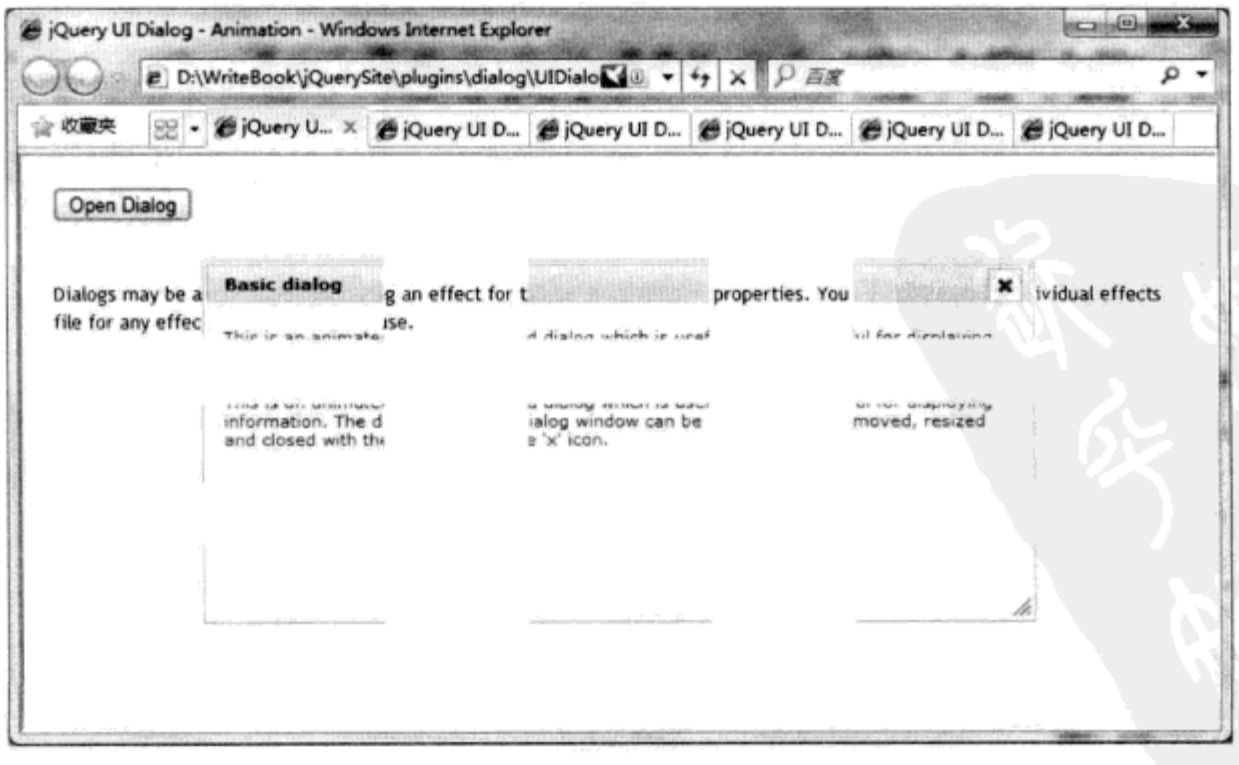


图 9.15　对话框关闭动态过程

3. 模式对话框

前面两个示例所看到的对话框都是非模式对话框，也就是当对话框出现时，对话框后面页面上的内容还是可以操作的。下面介绍如何通过 Dialog 创建模式对话框。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   <script>
2       $(function() {
3           //移除所有对话框功能
4           $( "#dialog:ui-dialog" ).dialog( "destroy" );
5                   $( "#dialog-modal" ).dialog({
6               height: 140,    //设定对话框的高
7               modal: true     //创建模式对话框
8           });
9       });
10  </script>
```

上述代码第 7 行使用了 Dialog 插件的模式对话框属性“modal”，并将其设为真。当页面加载完成后，自动打开一个模式对话框，效果如图 9.16 所示。

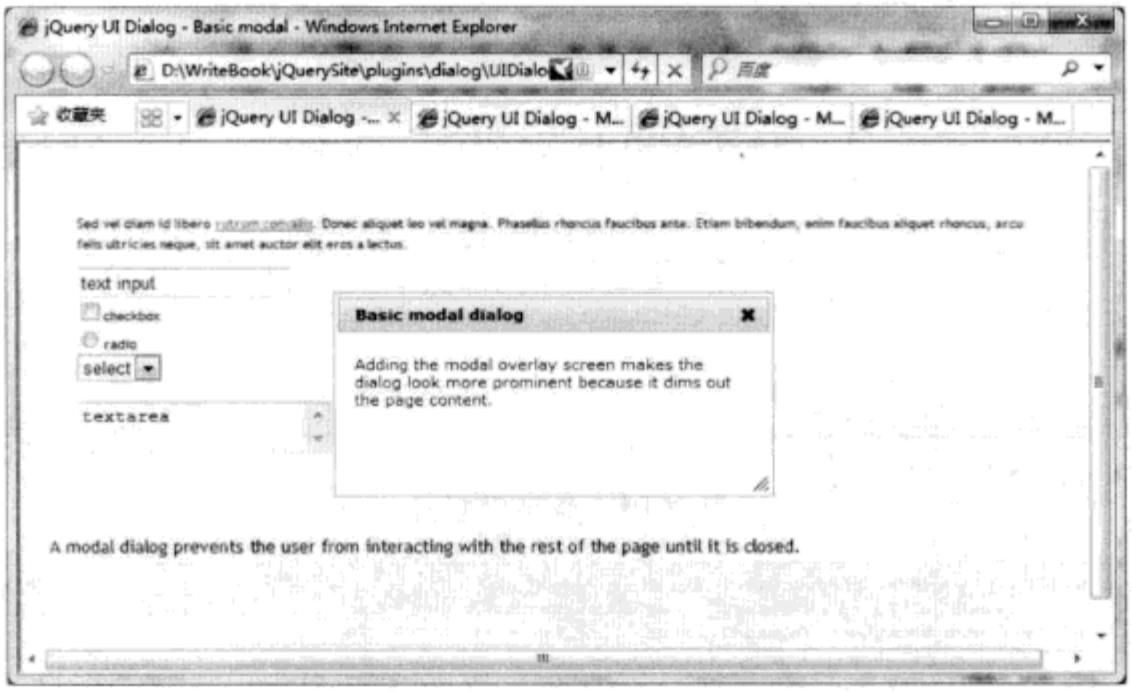


图 9.16　模式对话框

4. 模式确认对话框

这个对话框是将原有对话框功能取消掉，利用 Dialog 属性重新创建对话框功能，以实现确认对话框。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       //a workaround for a flaw in the demo system (http://dev.jqueryui.
        com/ticket/4375), ignore!
4       $( "#dialog:ui-dialog" ).dialog( "destroy" );
5       $( "#dialog-confirm" ).dialog({
```

```
6           resizable: false,   //不可调整对话框大小
7           height:140,         //对话框高度
8           modal: true,        //模式对话框
9           buttons: {
10              "Delete all items": function() {
11                  $( this ).dialog( "close" );    //设定对话框功能按钮
12              },
13              Cancel: function() {
14                  $( this ).dialog( "close" );    //对话框“取消”按钮
15              }
16          }
17      });
18  });
19  </script>
```

上述代码第 4 行销毁原有对话框的所有默认属性。第 6 行设定对话框禁止调整大小。第 7 行设定对话框高度为 140 像素。第 8 行设定模式对话框。第 9～16 行添加按钮功能。第 10 行指定文本为“Delete all items”的按钮的关闭功能。第 11 行指定了对话框的关闭行为。第 13 行同样指定了“取消”按钮的对话框关闭行为。效果如图 9.17 所示。

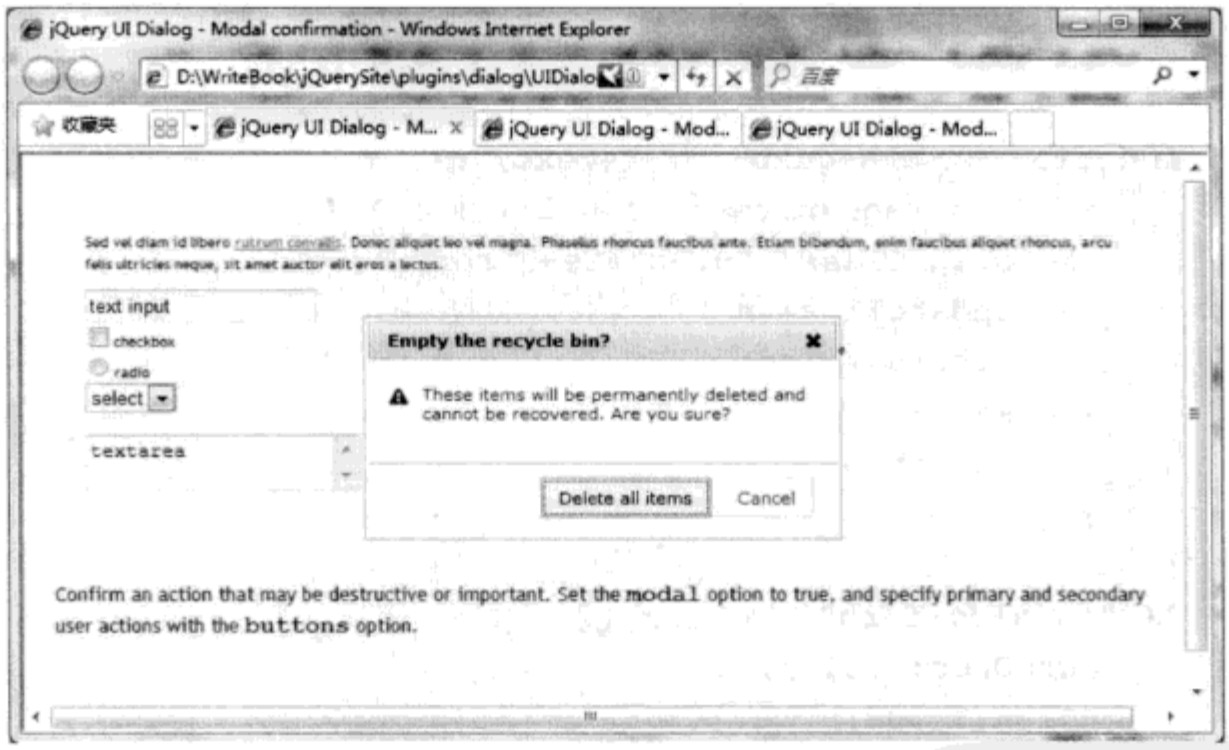


图 9.17　确认模式对话框

5. 模式表单对话框

这个对话框是将原有对话框功能取消掉，利用 Dialog 属性重新创建对话框功能，以实现用户输入表单对话框。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   $(function() {
2       //移除所有对话框功能
3       $( "#dialog:ui-dialog" ).dialog( "destroy" );
4
```

```
5       var name = $( "#name" ),
6           email = $( "#email" ),
7           password = $( "#password" ),
8           allFields = $( [] ).add( name ).add( email ).add( password ),
9           tips = $( ".validateTips" );
10      function updateTips( t ) {
11          tips
12              .text( t )
13              .addClass( "ui-state-highlight" );
14          setTimeout(function() {
15              tips.removeClass( "ui-state-highlight", 1500 );
16          }, 500 );
17      }
18      function checkLength( o, n, min, max ) {
19          if ( o.val().length > max || o.val().length < min ) {
20              o.addClass( "ui-state-error" );
21              updateTips( "Length of " + n + " must be between " +
22                  min + " and " + max + "." );
23              return false;
24          } else {
25              return true;
26          }
27      }
28      function checkRegexp( o, regexp, n ) {
29          if ( !( regexp.test( o.val() ) ) ) {
30              o.addClass( "ui-state-error" );
31              updateTips( n );
32              return false;
33          } else {
34              return true;
35          }
36      }
37      $( "#dialog-form" ).dialog({
38          autoOpen: false,
39          height: 300,
40          width: 350,
41          modal: true,
42          buttons: {
43              "Create an account": function() { //实现表单信息添加功能按钮
44                  var bValid = true;
45                  allFields.removeClass( "ui-state-error" );
46                  bValid = bValid && checkLength( name, "username", 3, 16 );
47                  bValid = bValid && checkLength( email, "email", 6, 80 );
48                  bValid = bValid && checkLength( password, "password", 5, 16 );
49                  bValid = bValid && checkRegexp( name, /^[a-z]([0-9a-z_])
                    +$/i, "Username may consist of a-z, 0-9, underscores,
                    begin with a letter." );
                    //From jquery.validate.js (by joern), contributed by Scott
```

```
                Gonzalez: http://projects.scottsplayground.com/email_
                address_validation/
                bValid = bValid && checkRegexp( email, /^((([a-z]
                |\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF
                \uF900-\uFDCF\uFDF0-\uFFEF])+(\.([a-z]|\d|[!#\$%&'\
                *\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
                \uFFEF])+)*)|((\x22)((((\x20|\x09)*(\x0d\x0a))?(\x20|
                (([\x01-\x08\x0b\x0c\x0e-\x1f\x7f]|\x21|\x09)+)?[\x23-
                \x5b]|[\x5d-\x7e]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\
                uFFEF])|(\\([\x01-\x09\x0b\x0c\x0d-\x7f]|[\u00A0-\uD7FF\
                uF900-\uFDCF\uFDF0-\uFFEF])))*(((\x20|\x09)*(\x0d\x0a))?
                (\x20|\x09)+)?(\x22)))@((([a-z]|\d|[\u00A0-\uD7FF\
                uF900-\uFDCF\uFDF0-\uFFEF])|(([a-z]|\d|[\u00A0-\uD7FF\
                uF900-\uFDCF\uFDF0-\uFFEF])([a-z]|\d|-|\.|_|~|[\u00A0-
                \uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-z]|\d|[\u00A0-
                \uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-z]|[\u00A0-
                \uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(([a-z]|[\u00A0-
                \uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-z]|\d|-|\.|_|~|
                [\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-z]|
                [\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.?$/i,
                "eg. ui@jquery.com" );
50              bValid = bValid && checkRegexp( password, /^([0-9a-zA-
                Z])+$/, "Password field only allow : a-z 0-9" );
51              if ( bValid ) {
52                  $( "#users tbody" ).append( "<tr>" +
53                      "<td>" + name.val() + "</td>" +
54                      "<td>" + email.val() + "</td>" +
55                      "<td>" + password.val() + "</td>" +
56                  "</tr>" );
57                  $( this ).dialog( "close" );
58              }
59          },
60          Cancel: function() {
61              $( this ).dialog( "close" );
62          }
63      },
64      close: function() {
65          allFields.val( "" ).removeClass( "ui-state-error" );
66      }
67  });
68  $( "#create-user" )
69      .button()
70      .click(function() {
71          $( "#dialog-form" ).dialog( "open" );
72      });
73  });
74  </script>
```

上述代码第 3 行取消对话框原有功能。第 5～9 行获取对话框表单元素对象及验证提示信息显示对象。第 10～17 行根据验证的错误更新验证信息显示内容。第 18～27 行检查每个表单元素的用户输入长度。第 28～36 行进行正则表达式验证。第 37 行初始化插件。第 38 行设定对话框不自动打开。第 39、40 行设定对话框宽 350 像素、高 300 像素。第 41 行设定对话框为模式对话框。第 42～66 行设定对话框相关按钮的功能。

其中第 43～59 行为文本为“Create an account”的按钮设定表单元素内容验证功能。如果验证通过，则在页面上的表格中添加一条用户信息，否则在验证消息部分显示错误消息。第 60～62 行设定“取消”按钮的关闭对话框功能。第 64～66 行设定“关闭”按钮的附加功能，将表单元素内容设置为空，并取消验证错误样式。

第 68～72 行添加调用打开对话框行为的按钮功能，效果如图 9.18～图 9.20 所示。

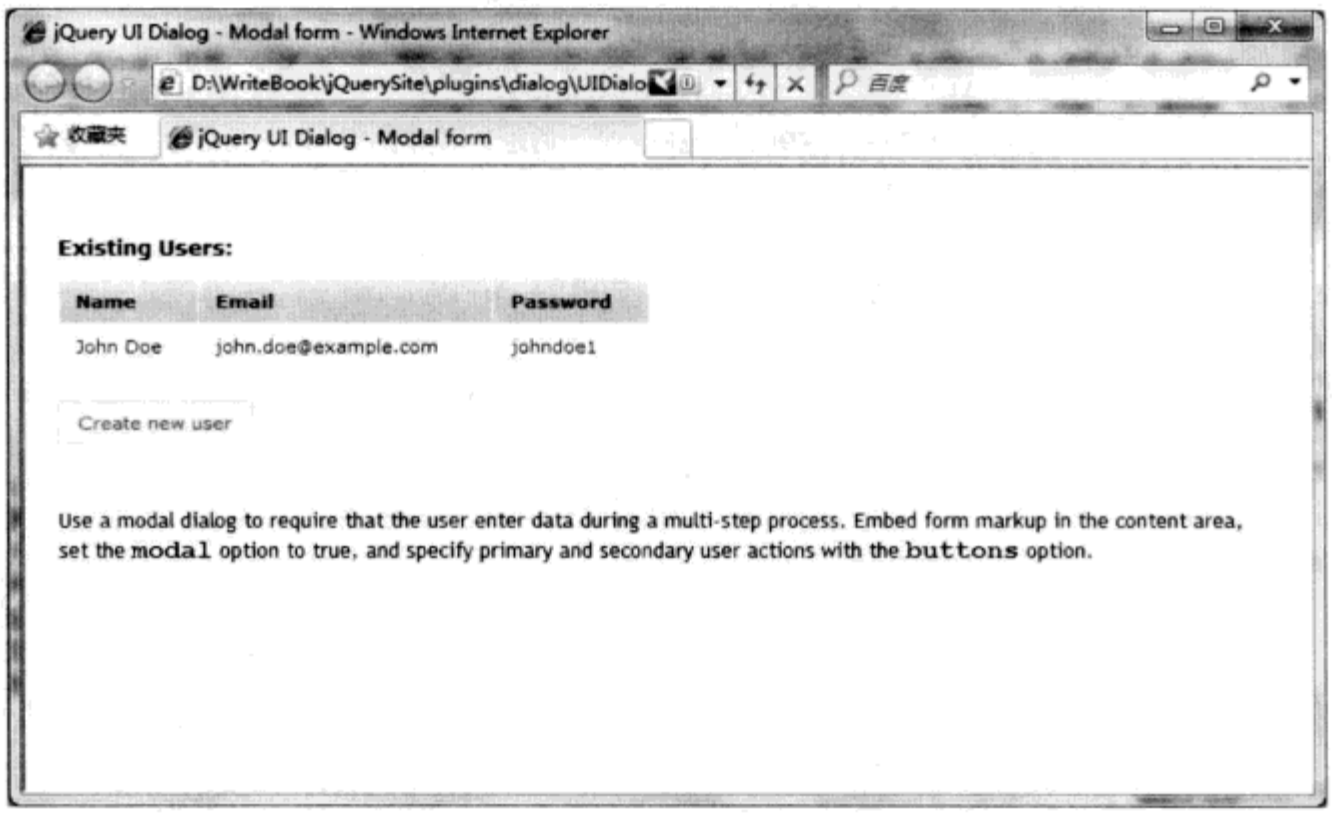


图 9.18　页面加载后效果

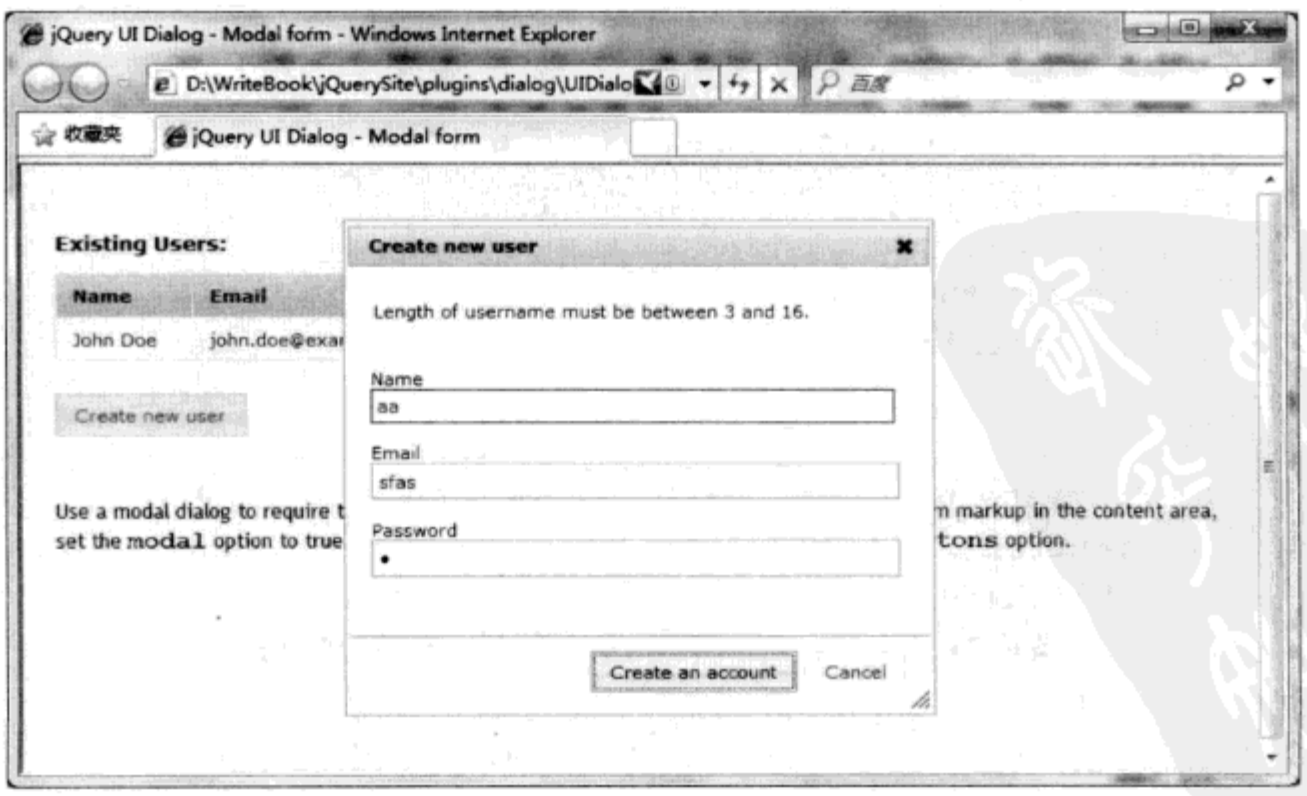


图 9.19　对话框弹出后验证出现错误效果

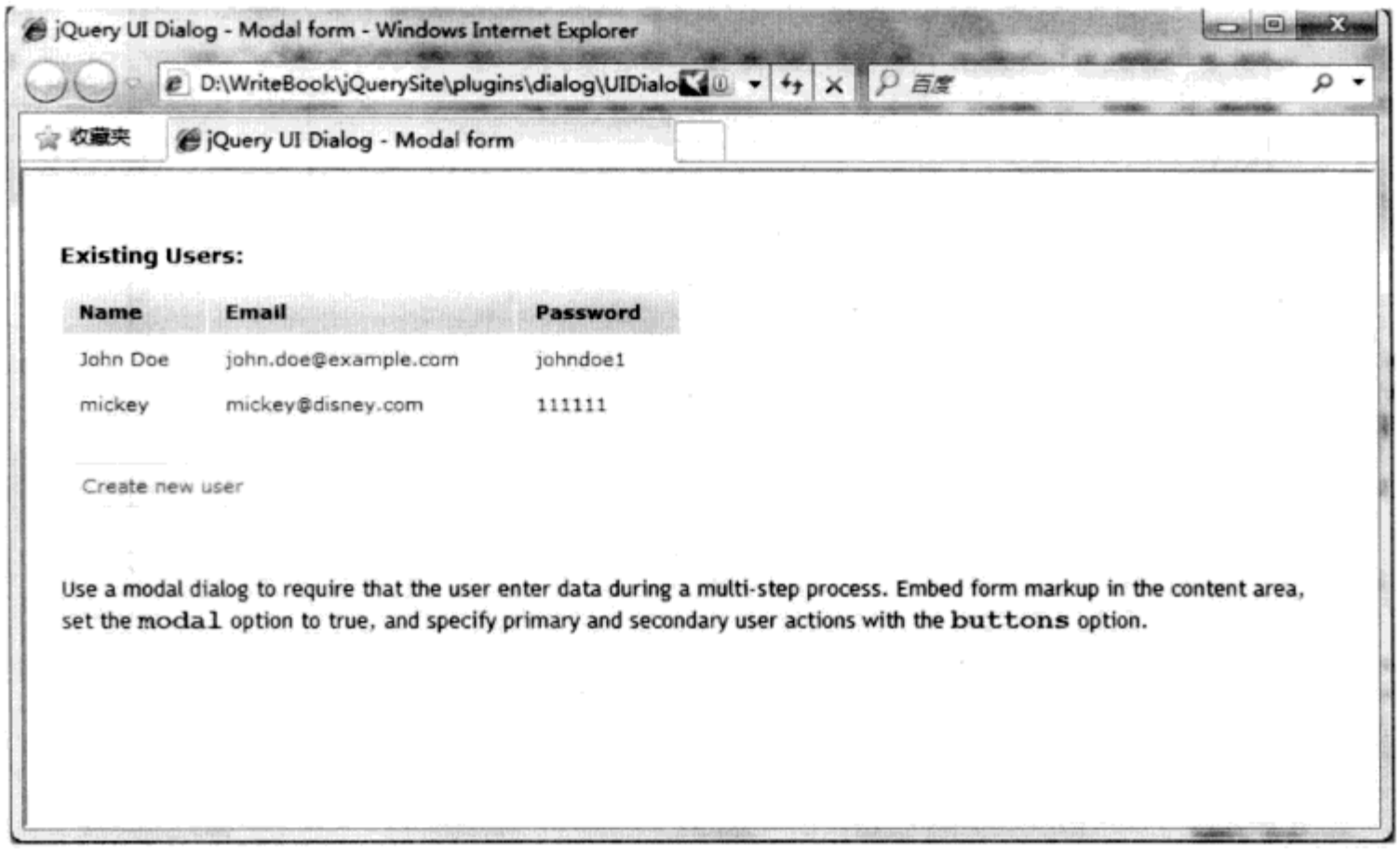


图 9.20　在对话框中成功填入信息后的页面效果

6. 模式消息对话框

这个对话框取消原有对话框功能，利用 Dialog 属性重新创建对话框功能，并在对话框中显示一定的消息内容。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       //a workaround for a flaw in the demo system (http://dev.
        jqueryui.com/ticket/4375), ignore!
4       $( "#dialog:ui-dialog" ).dialog( "destroy" );
5       $( "#dialog-message" ).dialog({
6           modal: true,
7           buttons: {
8               Ok: function() {
9                   $( this ).dialog( "close" );
10              }
11          }
12      });
13  });
14  </script>
```

上述代码第 4 行的功能与前面的示例相同。第 6 行设定模式对话框形式。第 8、9 行为 OK 按钮添加对话框的关闭功能。效果如图 9.21 所示。

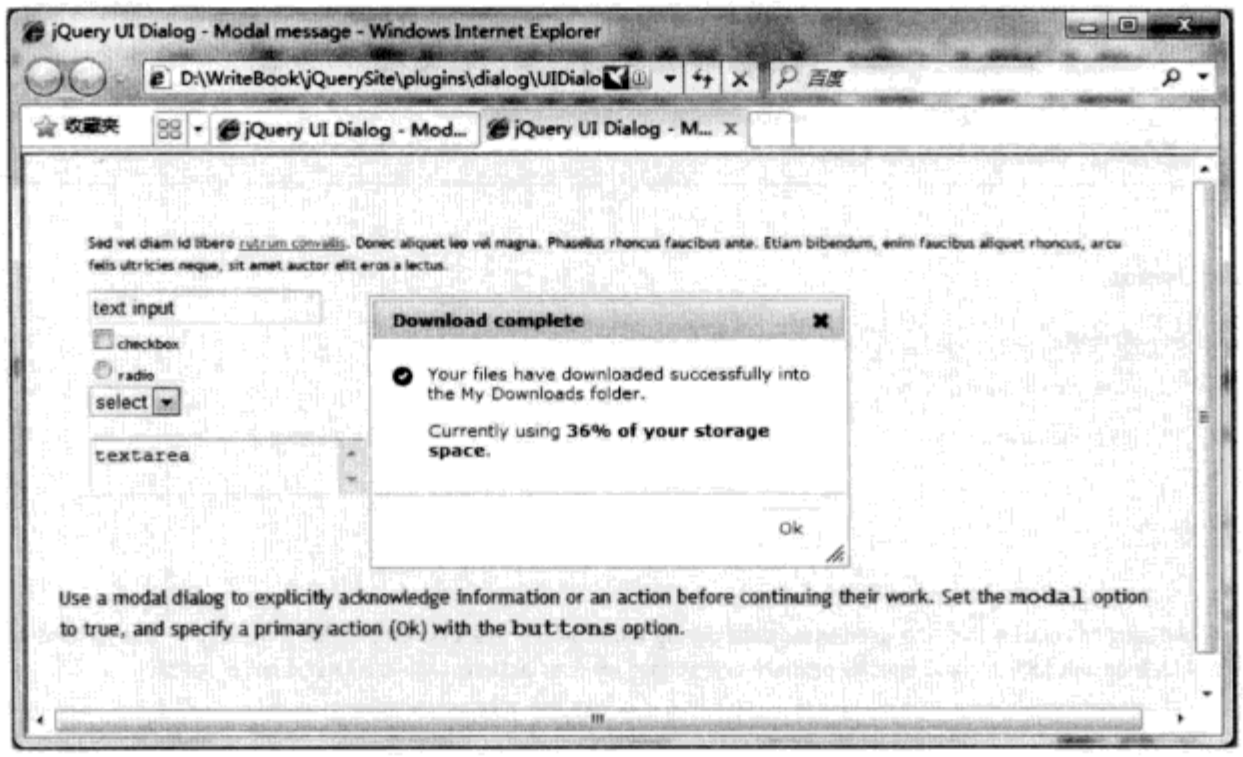


图 9.21　模式消息对话框

9.7　小　　结

本章对 jQuery 创建网页对话框进行了讲解。主要内容包括基本对话框实现原理，模式对话框及常用对话框的创建及操作，并介绍了两个对话框插件。重点部分是对话框的实现原理及 Dialog 插件的使用，同时这一部分也是本章的难点。下一章将讲解 jQuery 对滑动条的实现。

9.8　习　　题

【习题 1】利用本章所学内容自定义一个带表单录入功能的模式对话框。

【习题 2】练习 jQuery.UI.Dialog 对话框插件的使用方法。

【习题 3】练习 jqModal 对话框插件的使用方法。

第 10 章　设计滑动条

在很多网页中，都会看到页面中的部分区域可以使用滑动条操作显示内容，但是这里的滑动条与浏览器的滑动条不同。这种滑动条其实也是页面元素的一部分，可以通过程序来控制其出现和如何操作。本章主要讲解如何使用 jQuery 创建及操作滑动条。

10.1　自定义滑动条

本节主要讲解如何使用 jQuery 创建区域滑动条。它的实现原理是：在页面的内容显示区域中，创建长短适当的滑动条层，并在滑动条层上添加鼠标移动和鼠标键释放事件，以产生内容和移动条上下移动的效果。其中，使用到了 jQuery 函数 ready()、height()、show()、css()、find()、mousemove()、mousedown()。

其功能实现如下。

（1）判断内容所在层的高度是否大于显示区域的预定高度。如果大于，则加入滑动条，否则不添加滑动条。

（2）加载滑动条效果。

（3）设定滑动条区间的鼠标按下事件，在事件中提取鼠标位置。

（4）设定滑动块鼠标移动事件，判断鼠标移动位置，并按照鼠标移动的合理位置移动显示内容。

首先，利用 HTML 创建静态页面。相关的 CSS 代码请参考光盘内容。

```
<div id="subwindow">
  <div id="scrollarea">
    <div id="scrollblock">
      <img id="scrollimg" src=""/><br />
      </div>
  </div>
  <div id="content">
      <img src="img/img8.gif" />
  </div>
</div>
```

然后，引入 jQuery 库文件：

```
<script type="text/JavaScript" src="jslib/jquery.js"></script>
```

最后，加入 JavaScript 功能代码：

```
<script type="text/JavaScript">
  $(function(){
      if($("#content").height()>$("#subwindow").height())
      {
          $("#scrollarea").show("slow"); //显示滚动区域
          //调整滚动块的高度，单位为像素，值为整数
  $("#scrollblock").find("img").height(parseInt($("#subwindow").
  height()*$("#subwindow").height()/ $("#content").height()));
          $("#scrollblock").show("slow");//显示滚动块
          //定义鼠标位置变量及内容坐标变量
          var mouseX=0;
          var mouseY=0;
          var contentX=0;
          var contentY=0;
          $("#scrollarea").mousedown(function(e){
                                          //鼠标在滚动区间内的左键按下事件
              var devent=e||event;        //获取事件参数
              mouseX=devent.clientX;      //获取鼠标坐标
              mouseY=devent.clientY;
              contentX=parseInt($("#scrollblock").css("left"));
                                          //获取内容位置
              contentY=parseInt($("#scrollblock").css("top"));

          });
          $("#scrollblock").mousemove(function(e){    //鼠标移动滚动块事件
              var devent=e||event;                    //获取事件参数
              var distance=devent.clientY-mouseY+contentY;
                                          //计算鼠标纵向移动影响内容移动位移
              if(distance<0)
                  distance=0;
              else if(distance>=$("#scrollarea").height()-$
              ("#scrollblock").height())
                  distance=$("#scrollarea").height()-$("#scrollblock").
                  height();
              $("#scrollblock").css("top",distance+"px");
                                          //修改滑动块纵向位移
  $("#content").css("top",-(distance*(parseInt($("#content").height()
  -$("#subwindow").height()))/(parseInt($("#scrollarea").height())-
  parseInt($("#subwindow").height()*$("#subwindow").height()/ $
  ("#content").height()))+"px");
          });                             //更改内容移动位移

      }
  });
</script>
```

上述代码第 5～7 行对滑动条的显示进行设置。第 12～18 行设定滑动条区域的鼠标按下事件。在这个事件中获取了鼠标位置和滑动块的位置。第 19～28 行设定了滑动块的鼠标移动事件，判断鼠标的偏移位移是否在合理区间内。如果滑动块可以移动，则移动滑动块，并移动对应位移的内容。效果如图 10.1 所示。

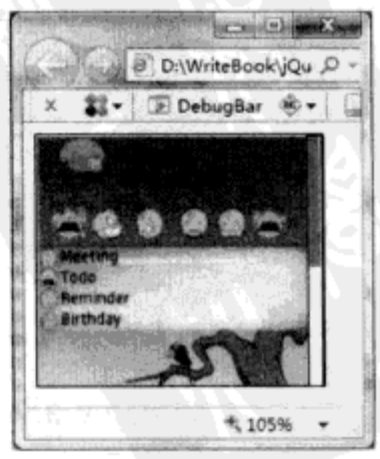


图 10.1　自定义滑动块

10.2　滑动条插件

本节介绍几种滑动条插件。这些插件为页面创建出了不同样式、不同功能的滑动条效果。

10.2.1　jQuery.UI.Slider 插件

jQuery.UI.Slider 插件是 jQuery 官方网站提供的一整套 UI 插件中的一部分。它的主要功能是利用 jQuery 生成特定功能的滑动条。这个插件需要以下 jQuery 文件的支持：jquery.ui.core.js、jquery.ui.widget.js、jquery.ui.mouse.js。

这个插件有一些相关属性，可以设置滑动条的样式和行为，如表 10.1 所示。

表 10.1　滑动条的属性

属　性	类　型	默 认 值	说　明
disable	布尔	false	屏蔽或者启用滑动条功能
animate	布尔，字符串，数字	false	当用户单击滑动块外部时，它的处理效果是否平滑；或者使用不同字符串表示动画速度；也可以用数字表示动画具体执行毫秒数
max	数字	100	滑动条所能表示的最大值
min	数字	0	滑动条所能表示的最小值
orientation	字符串	'horizontal'	滑动条的滑动方向，是从左向右，还是从下向上
range	布尔，字符串	false	范围设定，如果设置为真，则检测是否有两个滑动块，一块从小向大滑动，另一块从大向小移动
step	数字	1	滑动块移动的步长或者单位间隔
value	数字	0	设置滑动块的值，如果有多个滑动块，则对第一个进行设置
values	数组	null	设置多个滑动块的值

与这个滑动条相关的还有事件选项，如表 10.2 所示。

表 10.2　滑动条的事件

事　件	类　型	说　明
create	slidecreate	滑动条被创建时触发
start	slidestart	当用户开始滑动滑动块时触发
slide	slide	当鼠标移动滑动块时触发
change	slidechange	滑动操作停止时触发，或者通过编程改变当前值时触发
stop	slidestop	当用户停止滑动操作时触发

滑动条有一些行为可以控制滑动条的相关操作，如表 10.3 所示。

表 10.3　滑动条的行为

行　为	表　示	说　明
destroy	.slider("destroy")	完全删除滑动条的功能
disable	.slider("disable")	屏蔽滑动条
enable	.slider("enable")	启用滑动条
option	.slider("option" , optionName , [value])	获取或设置滑动条属性
widget	.slider("widget")	返回滑动条元素
value	.slider("value" , [value])	获取或设置单滑动块的滑动条的值
option	.slider("option" , options)	一次设定多个滑动条属性
values	.slider("values" , index , [value])	获取或设置多个滑动块的滑动条的所有值

下面通过示例来介绍一下这个插件的使用方法。

1. 默认样式滑动条

这个示例使用滑动条的默认参数形式，只调用了滑动条插件的初始化函数。这种情况下滑动块可以随意移动，并且移动的单位长度也不固定。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       $( "#slider" ).slider();    //利用滑动条插件基本功能
4   });
5   </script>
```

第 3 行中的 slider()就是滑动条插件的初始化函数。它使用了滑动条的默认样式，并在页面加载后自动显示滑动条，效果如图 10.2 所示。

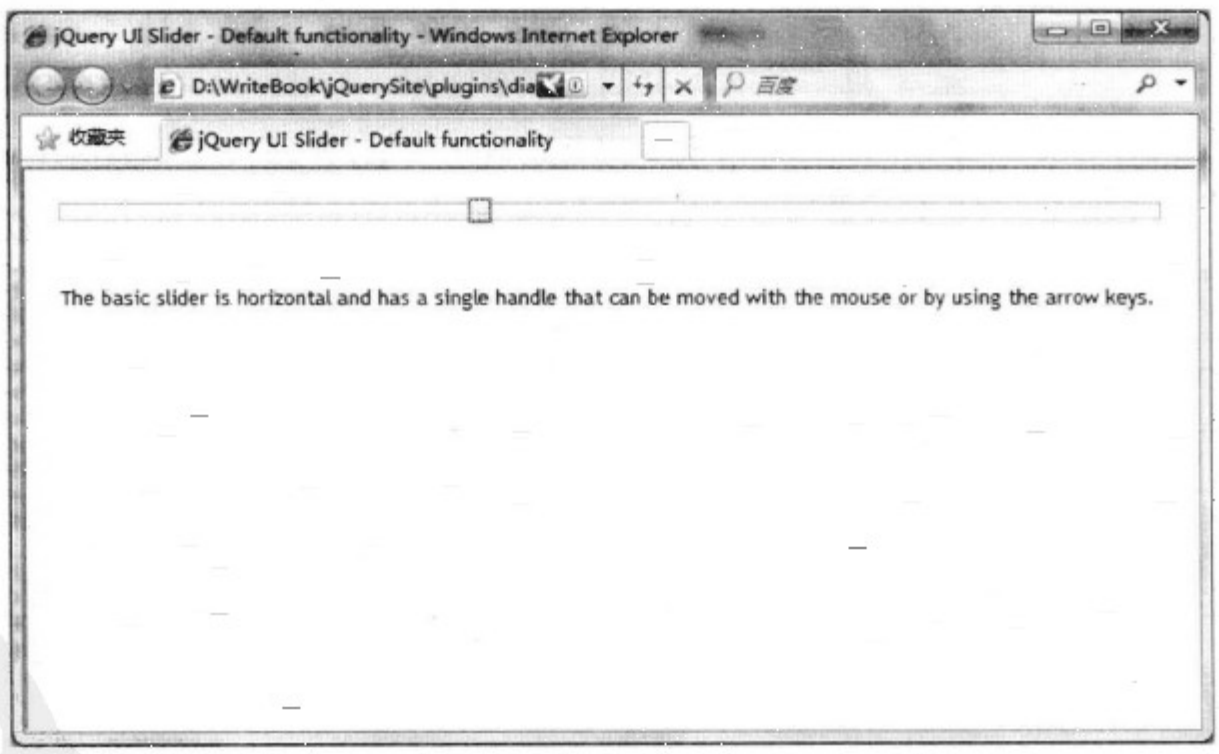


图 10.2　默认样式滑动条

2. 控制步进滑动条

这个示例使用前面介绍的滑动条的步进属性（即滑动条移动的单位长度），并使用滑

动条插件的初值及最大最小值属性和滑动事件。与前面的示例不同的是滑动块具有了最大和最小移动范围，并且单位移动长度已知，便于我们通过判断滑动块的位置而得到它所代表的值。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       $( "#slider" ).slider({
4           value:100,                          //滑动条初始位置
5           min: 0,                             //滑动条最小表示位置
6           max: 500,                           //滑动条最大表示位置
7           step: 50,                           //滑动条单位移动位移
8           slide: function( event, ui ) {      //滑动条滑动事件处理
9               $( "#amount" ).val( "$" + ui.value );
10          }
11      });
12      $( "#amount" ).val( "$" + $( "#slider" ).slider( "value" ) );
13  });
14  </script>
```

上述代码中第 4 行设定滑动块的初始位置。第 5 行设定滑动条最小值。第 6 行设定最大值。第 7 行设定步进为 50。第 9、10 行设定当滑动块移动时更改显示值。效果如图 10.3 所示。

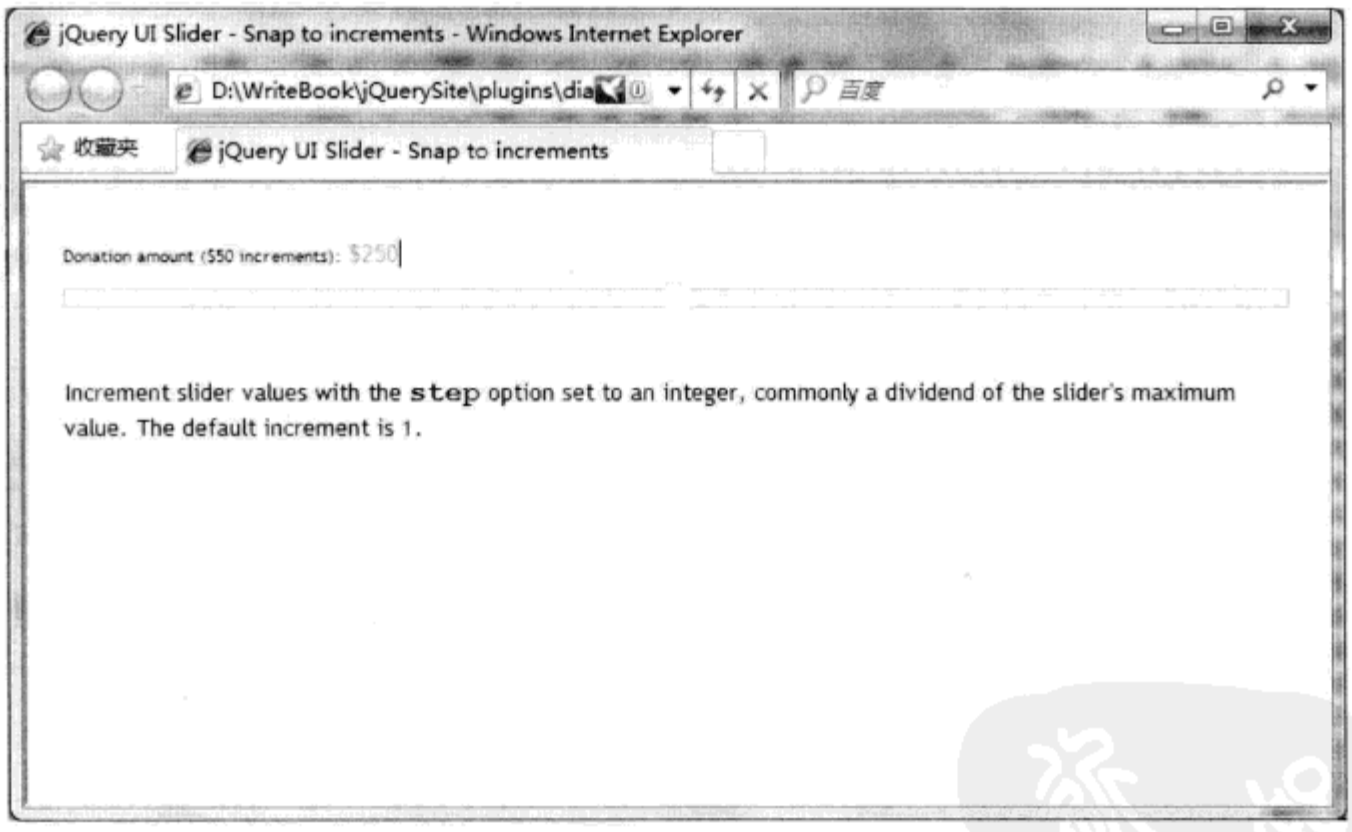


图 10.3　设定步进滑动条

3. 设定表示范围的滑动条

这个示例使用滑动条的范围参数，并使用滑动条插件的初值及最大最小值属性和滑动事件。表示范围的滑动框由分别表示范围上限和范围下限的两个滑动块组成，并且每个滑动块都有自己的移动区间，用户可以更形象地通过移动滑动块来获取范围空间。JavaScript 功能代码如下：

```
<script>
$(function() {
    $( "#slider-range" ).slider({
        range: true,                              //设定表示范围功能
        min: 0,                                   //范围最小值可为 0
        max: 500,                                 //范围最大值为 500
        values: [ 75, 300 ],                      //表示范围区间的初始值
        slide: function( event, ui ) { //滑动条滑动事件
            $( "#amount" ).val( "$" + ui.values[ 0 ] + " - $" + ui.
            values[ 1 ] );
        }
    });
    $( "#amount" ).val( "$" + $( "#slider-range" ).slider( "values", 0 ) +
        " - $" + $( "#slider-range" ).slider( "values", 1 ) );
});
</script>
```

上述代码中第 4 行设定了滑动条功能为表示范围，即具有两个滑动块，分别表示最大值和最小值。第 7 行设定了范围初始值，即两个滑动块的初始位置。效果如图 10.4 所示。

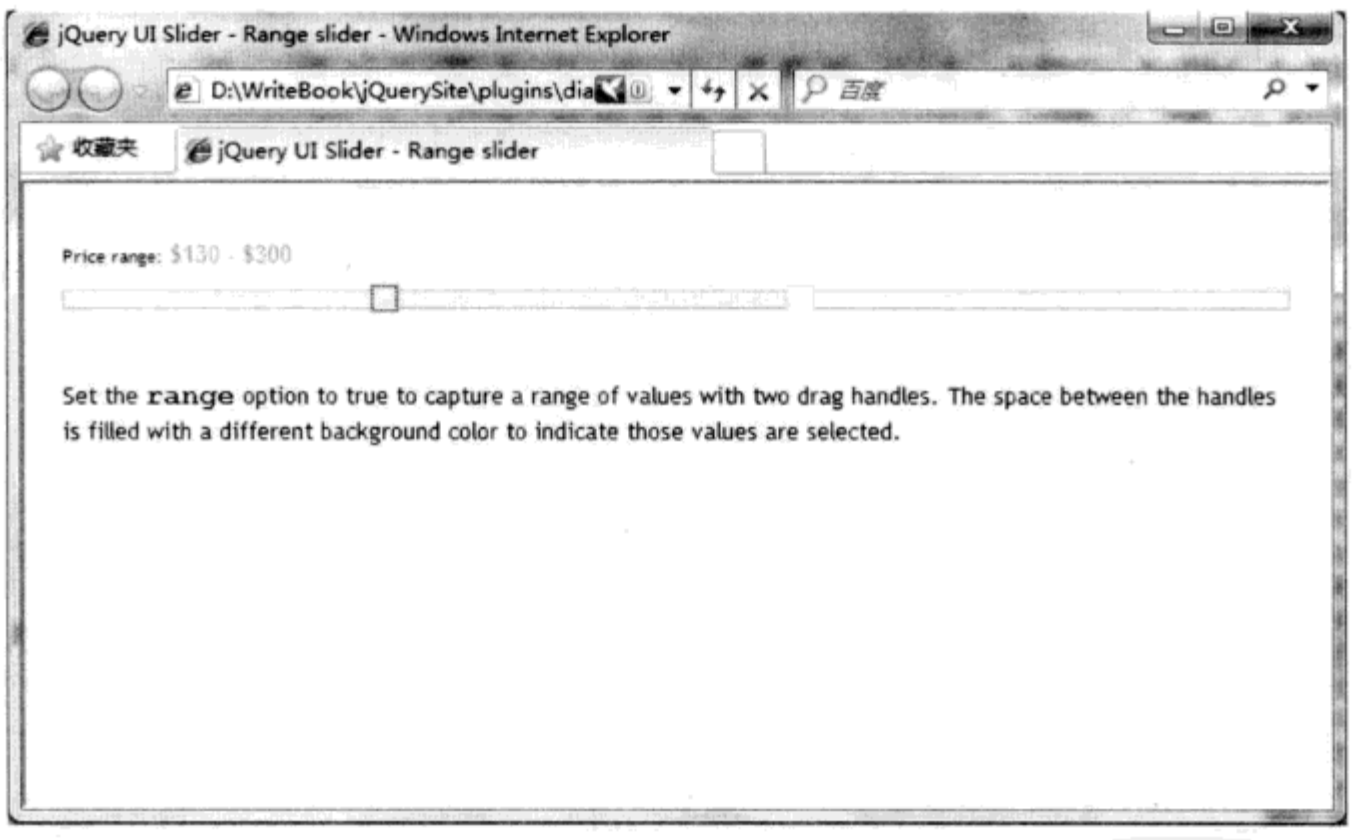


图 10.4　设定范围滑动条

4．固定了最小值范围的滑动条

这个示例使用滑动条的范围参数，并使用滑动条插件的初值及最大最小值属性和滑动事件。固定最小值滑动块的位置即固定了范围下限，通过代表范围上限的滑动块的移动获取区间值。JavaScript 功能代码如下：

```
<script>
$(function() {
    $( "#slider-range-min" ).slider({
```

```
4           range: "min",                          //设定最小值范围
5           value: 37,                             //最小范围值为 37
6           min: 1,                                //最小值为 1
7           max: 700,                              //最大值为 700
8           slide: function( event, ui ) { //滑动条滑动事件
9               $( "#amount" ).val( "$" + ui.value );
10          }
11      });
12      $( "#amount" ).val( "$" + $( "#slider-range-min" ).slider
        ( "value" ) );
13  });
14  </script>
```

上述代码第 4 行设定滑动条功能为表示范围，但固定了范围最小值不动，用户只可修改范围最大值。效果如图 10.5 所示。

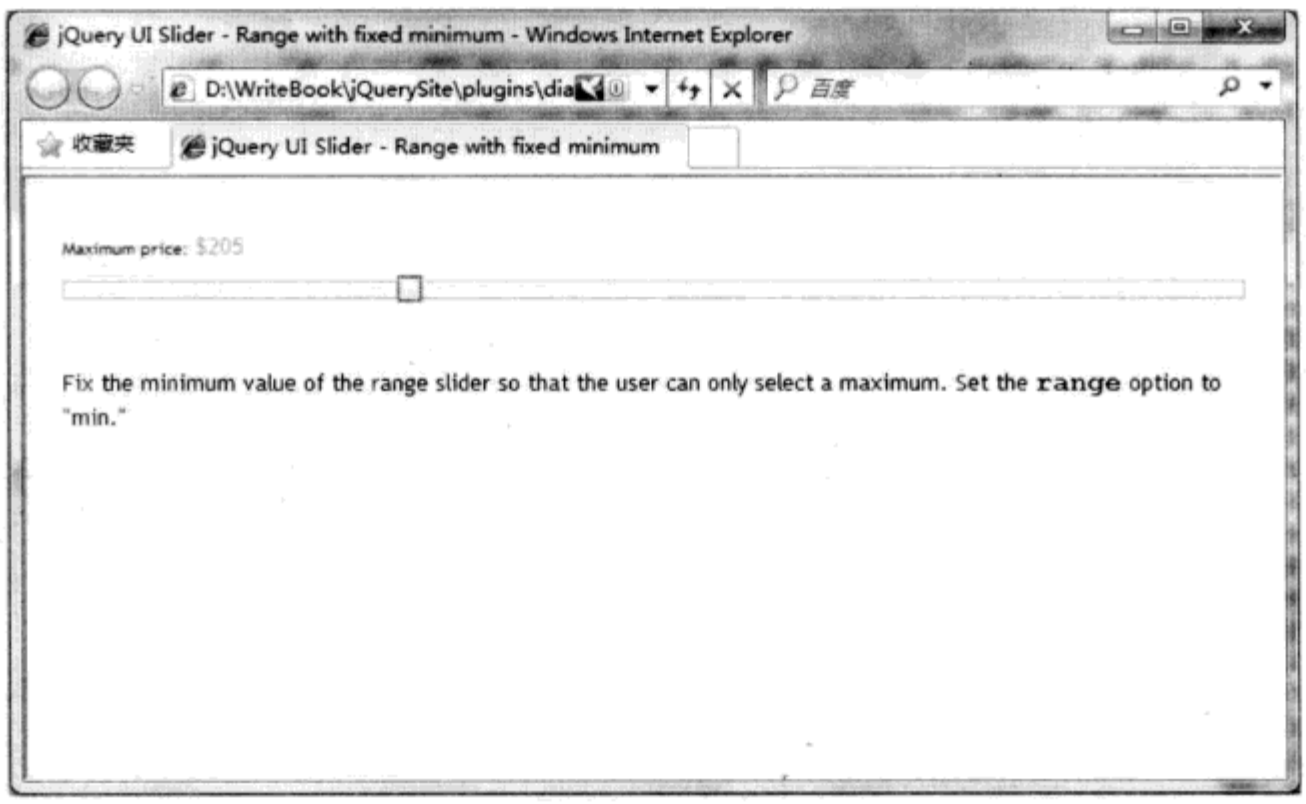


图 10.5　固定了最小值的范围滑动条

5. 固定最大值的滑动条

这个示例使用滑动条的范围参数，并使用滑动条插件的初值及最大最小值属性和滑动事件。与上例相反，此处是下限可调整。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       $( "#slider-range-max" ).slider({
4           range: "max",                          //设定最大范围表示功能
5           min: 1,                                //最小表示值为 1
6           max: 10,                               //最大表示值为 10
7           value: 2,                              //初始值为 2
8           slide: function( event, ui ) { //滑动条滑动事件
9               $( "#amount" ).val( ui.value );
10          }
11      });
```

```
12      $( "#amount" ).val( $( "#slider-range-max" ).slider( "value" ) );
13  });
14  </script>
```

上述代码第 4 行设定滑动条功能为表示范围，但固定了范围最大值不动，用户只可修改范围最小值。效果如图 10.6 所示。

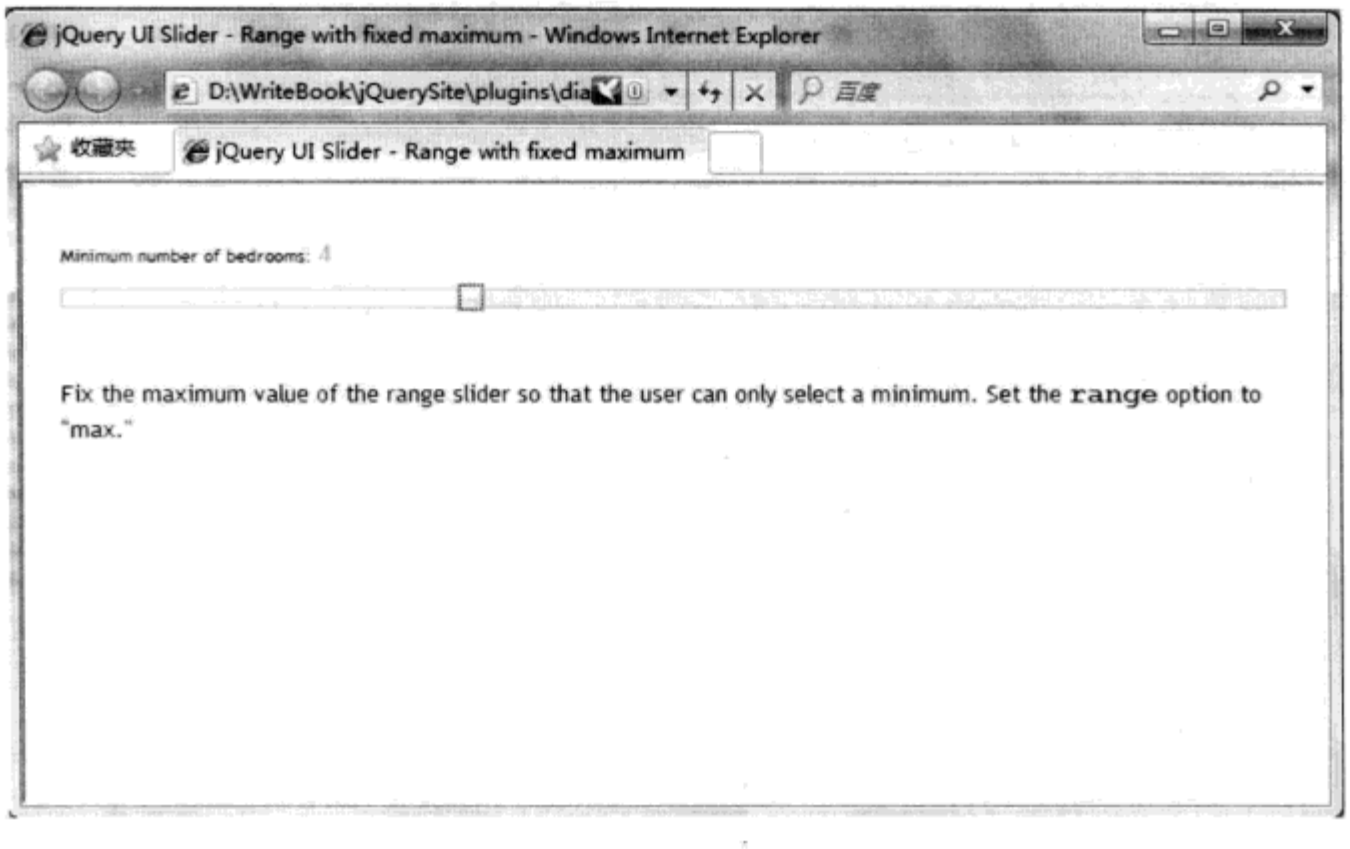


图 10.6　固定了最大值的范围滑动条

6. 通过外部元素改变滑动条

这个示例使用滑动条的范围参数，并使用滑动条插件的初值及最大最小值属性和滑动事件。通过用户在其他部分输入范围值，并将此值在滑动块上显示出来，使用户更形象地了解自己选择的范围值。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       var select = $( "#minbeds" );
4       var slider = $( "<div id='slider'></div>" ).insertAfter
        ( select ).slider({
5           min: 1,                                    //最小表示值为 1
6           max: 6,                                    //最大表示值为 6
7           range: "min",                              //设定最小范围表示功能
8           value: select[ 0 ].selectedIndex + 1,
9           slide: function( event, ui ) {             //滑动条滑动事件
10              select[ 0 ].selectedIndex = ui.value - 1;
11          }
12      });
13      $( "#minbeds" ).change(function() {
14          slider.slider( "value", this.selectedIndex + 1 );
15      });
16  });
17  </script>
```

上述代码第 4 行通过 jQuery 动态创建了滑动条所在层，并调用滑动条初始化函数。第 8 行对表示范围上限的滑动块设定初始值，但是这个值是通过下拉列表框获取的。这个滑动条也是范围滑动条，固定了最小值不变化，并可通过下拉列表框修改最大值。效果如图 10.7 所示。

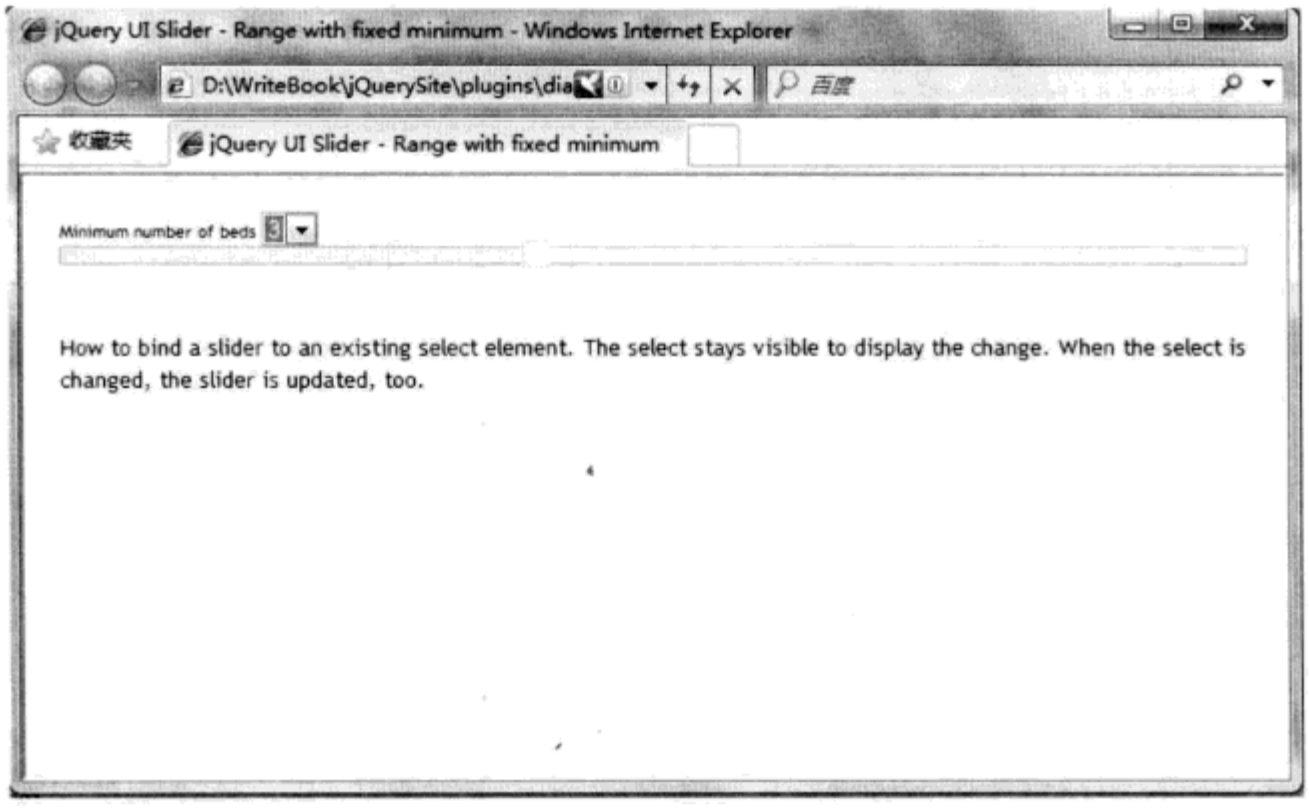


图 10.7　外部元素控制最大值变化滑动条

7．垂直滑动条

这个示例使用滑动条的滑动方向和范围参数，并使用滑动条插件的初值及最大最小值属性和滑动事件。这里只是简单地将前面使用的滑动条纵向摆放，产生了垂直滑动条的效果。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
3       $( "#slider-vertical" ).slider({
4           orientation: "vertical",          //垂直滑动
5           range: "min",                     //最小表示范围
6           min: 0,                           //最小表示值为 0
7           max: 100,                         //最大表示值为 100
8           value: 60,                        //初始值为 60
9           slide: function( event, ui ) {    //滑动条滑动事件
10              $( "#amount" ).val( ui.value );
11          }
12      });
13      $( "#amount" ).val( $( "#slider-vertical" ).slider( "value" ) );
14  });
15  </script>
```

上述代码第 4 行通过滑动条的滑动方向参数，设定了滑动条垂直滑动，并且最小值滑

动块固定在整个滑动条的下方。效果如图 10.8 所示。

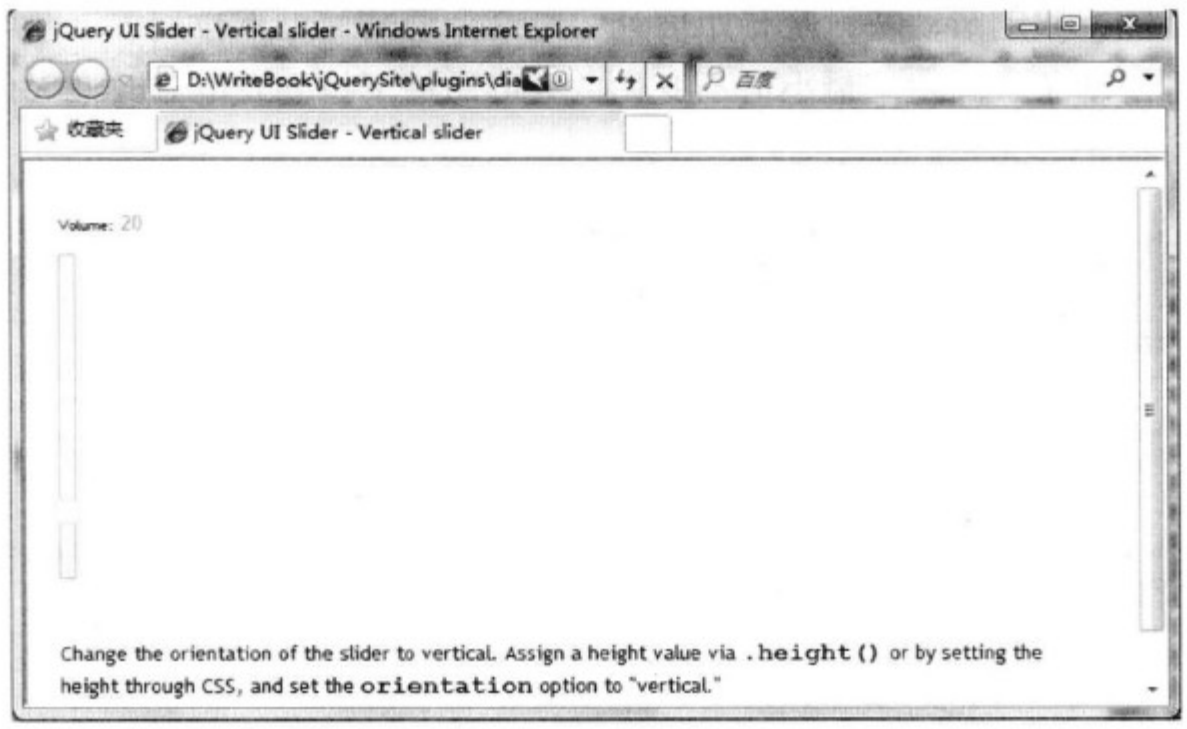


图 10.8　垂直滑动条

8. 配色功能滑动条

这个示例实现通过表示三原色的三个滑动条的取值变化，产生不同颜色搭配。该示例使用滑动方向、范围、最大值、初始值属性及滑动块移动和值改变事件。通过三个滑动条中滑动块的移动代表三原色的不同的值，并对这些值进行运算，搭配出混合后的颜色。JavaScript 功能代码如下：

```
function hexFromRGB(r, g, b) {
    var hex = [                                    //三原色十六进制值字符串
        r.toString( 16 ),
        g.toString( 16 ),
        b.toString( 16 )
    ];
    $.each( hex, function( nr, val ) { //每种颜色值用两位十六进制数表示
        if ( val.length === 1 ) {
            hex[ nr ] = "0" + val;
        }
    });
    return hex.join( "" ).toUpperCase();//拼接成一个六位的十六进制字符串
}
function refreshSwatch() {
    var red = $( "#red" ).slider( "value" ),
        green = $( "#green" ).slider( "value" ),
        blue = $( "#blue" ).slider( "value" ),
        hex = hexFromRGB( red, green, blue );
    $( "#swatch" ).css( "background-color", "#" + hex );
}
$(function() {
```

```
22      $( "#red, #green, #blue" ).slider({
23          orientation: "horizontal",      //水平滑动
24          range: "min",                   //最小表示范围功能
25          max: 255,                       //最大表示值
26          value: 127,                     //初始值
27          slide: refreshSwatch,           //滑动事件
28          change: refreshSwatch           //滑动块改变位置事件
29      });
30      $( "#red" ).slider( "value", 255 );
31      $( "#green" ).slider( "value", 140 );
32      $( "#blue" ).slider( "value", 60 );
33  });
34  </script>
```

上述代码第 1～13 行表示将三个不同滑动条的十进制整型值转换成一个六位的十六进制字符串。第 14～20 行表示提取三个滑动条的整型值并进行转换后将颜色显示出来。第 22～29 行是三个滑动条的初始化设定，并指定了两个事件所关联的函数。第 30～31 行分别指定三个滑动条的不同起始值。效果如图 10.9 所示。

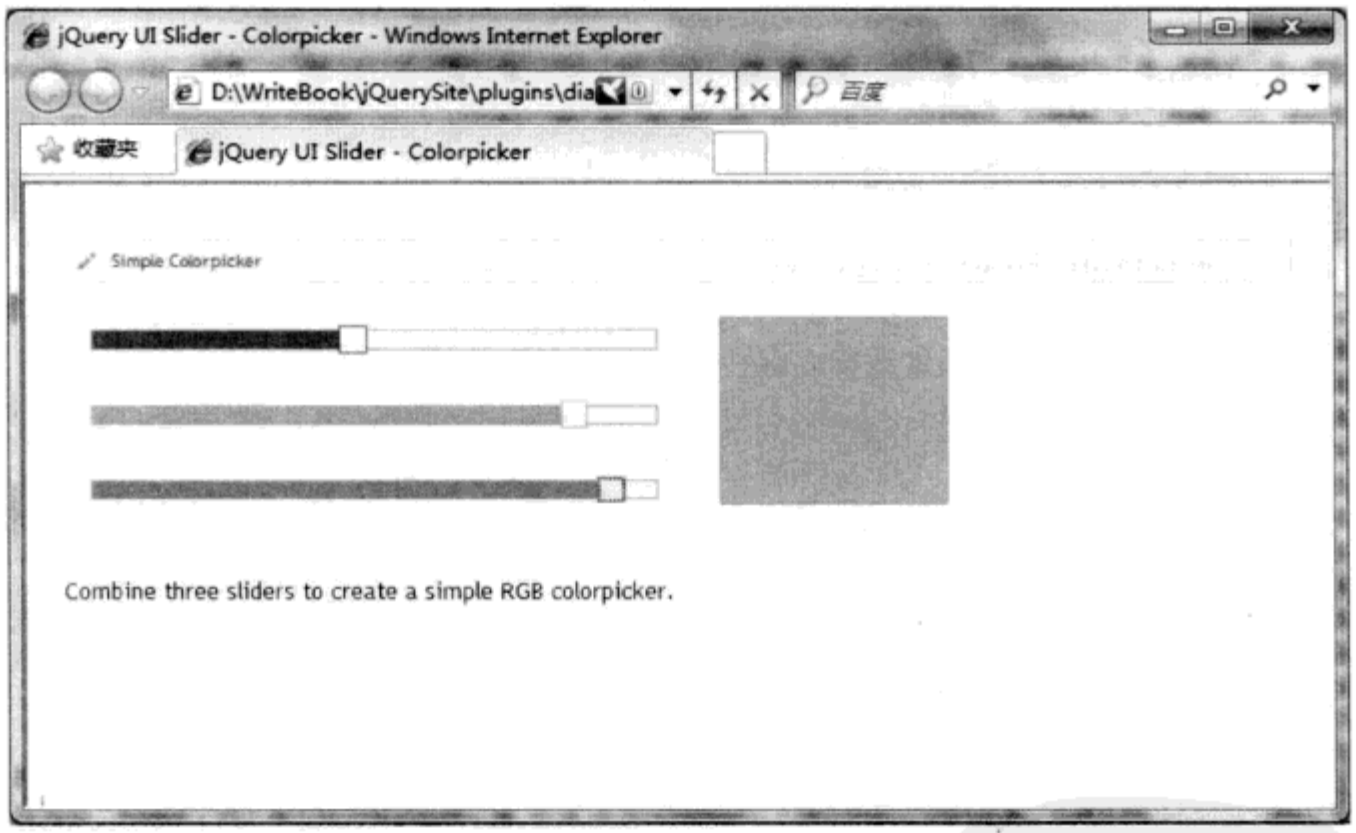


图 10.9　滑动条实现颜色搭配

9. 内容滚动条

这个示例使用滑动条与内容层搭配产生滚动条效果，主要是对滑动条的初始值和滑动事件进行操作，并且在这个示例中滑动条可根据窗口大小自动调整宽度。与本章最开始讲解的例子类似，通过将滑动条的位移转换成内容的位移，就产生了内容滚动效果。JavaScript 功能代码如下：

```
1   <script>
2   $(function() {
```

```
3       //滚动区域
4       var scrollPane = $( ".scroll-pane" ),
5           scrollContent = $( ".scroll-content" );
6
7       //build slider
8       var scrollbar = $( ".scroll-bar" ).slider({
9           slide: function( event, ui ) {
10              if ( scrollContent.width() > scrollPane.width() ) {
                                        //判断显示内容宽度与显示区域宽度
11                  scrollContent.css( "margin-left", Math.round(
12                      ui.value / 100 * ( scrollPane.width() - scroll-
                        Content.width() )
13                  ) + "px" );     //设定内容显示位置
14              } else {
15                  scrollContent.css( "margin-left", 0 );
16              }
17          }
18      });
19
20      //
21      var handleHelper = scrollbar.find( ".ui-slider-handle" )
22      .mousedown(function() {
23          scrollbar.width( handleHelper.width() );
24      })
25      .mouseup(function() {
26          scrollbar.width( "100%" );
27      })  //设定滑动条的鼠标左键按下与释放事件
28      .append( "<span class='ui-icon ui-icon-grip-dotted-vertical'>
        </span>" )                          //添加图标
29      .wrap( "<div class='ui-handle-helper-parent'></div>" ).parent();
30
31      //change overflow to hidden now that slider handles the scrolling
32      scrollPane.css( "overflow", "hidden" );
33
34      //size scrollbar and handle proportionally to scroll distance
35      function sizeScrollbar() {
36          var remainder = scrollContent.width() - scrollPane.width();
            //内容宽度与显示区域宽度的差
37          var proportion = remainder / scrollContent.width();
            //未显示宽度占整个内容宽度的比例
38          var handleSize = scrollPane.width() - ( proportion * scrollPane.
            width() );
            //滑动区间的宽度
39          scrollbar.find( ".ui-slider-handle" ).css({
40              width: handleSize,
41              "margin-left": -handleSize / 2
42          });
43          handleHelper.width( "" ).width( scrollbar.width() - handleSize );
```

```
44      }
45
46      //reset slider value based on scroll content position
47      function resetValue() {
48          var remainder = scrollPane.width() - scrollContent.width();
49          var leftVal = scrollContent.css( "margin-left" ) === "auto" ? 0 :
50              parseInt( scrollContent.css( "margin-left" ) );
            //显示内容的左偏移量
51          var percentage = Math.round( leftVal / remainder * 100 );
            //左偏移量占整个未显示宽度的百分比
52          scrollbar.slider( "value", percentage );
53      }
54
55      //滑动块到最大位置,当窗口变大时,重新设置显示内容
56      function reflowContent() {
57              var showing = scrollContent.width() + parseInt(scroll-
                Content.css( "margin-left" ), 10 );
                //当前显示内容的偏移位置
58              var gap = scrollPane.width() - showing;
                //显示位置与显示区域的差
59              if ( gap > 0 ) {    //是否超出显示范围，如果大于 0，则可显示
60                  scrollContent.css( "margin-left", parseInt( scroll-
                    Content.css( "margin-left" ), 10 ) + gap );
61              }
62      }
63
64      //change handle position on window resize
65      $( window ).resize(function() {
66          resetValue();
67          sizeScrollbar();
68          reflowContent();
69      });
70      //init scrollbar size
71      setTimeout( sizeScrollbar, 10 );//safari wants a timeout
72  });
73  </script>
```

上述代码第 4、5 行获取显示内容的层对象。第 8～18 行建立滑动条，设定滑动事件。当内容长度大于显示区域宽度时，按照滑动块所在位置算出内容宽度百分比，进行内容的相对位移。第 21～24 行设定鼠标按下和抬起时滑动条宽度。第 28、29 行添加图标到滑动块，第 32 行设定浮动层隐藏。

第 35～44 行根据显示内容宽度与显示区域的宽度比来设定滑动块和滑动条的宽度。第 47～53 行根据显示内容的当前位置和宽度比来设定新的滑动块的位置。第 56～62 行设定显示内容的当前显示位置。第 65～69 行根据窗口大小的改变修改滑动块的位置。效果如图 10.10 所示。

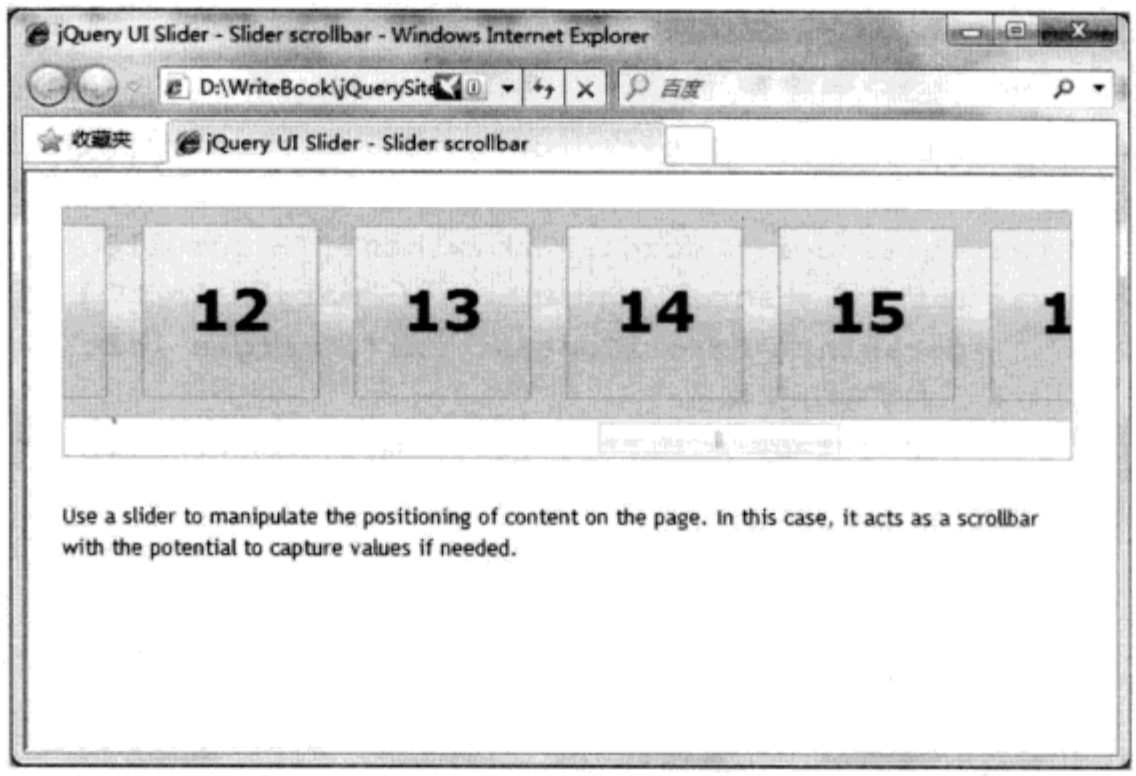


图 10.10　滑动条改变显示内容

10.2.2　jScrollPane 插件

jScrollPane 插件是一个跨浏览器的 jQuery 插件，它的英文网址为 http://jscrollpane.kelvinluck.com/。它可以将浏览器默认的滚动条转为 HTML 结构的，并可通过 CSS 灵活改变外观的滚动条形式。jScrollPane 灵活易用。

这个插件有一些相关属性，可以设置滚动条的样式和行为，如表 10.4 所示。

表 10.4　jScrollPane插件的属性说明

属　　性	类型	默认值	说　　明
showArrows	布尔	false	是否显示两端的箭头按钮
maintainPosition	布尔	true	当重新初始化时，滚动条是否保持原有位置
stickToBottom	布尔	false	在 maintainPosition 属性为真的时候，当整个滚动条已经到达底端的时候，保持这个状态
stickToRight	布尔	false	在 maintainPosition 属性为真的时候，当整个滚动条已经到达右端的时候，保持这个状态
autoReinitialise	布尔	false	在第一次初始化后，是否再次自动初始化。当内容发生改变时，不建议使用
autoReinitialiseDelay	整型	500	自动初始化间隔的毫秒数
verticalDragMinHeight	整型	0	垂直拖动的最小高度
verticalDragMaxHeight	整型	99999	垂直拖动的最大高度
horizontalDragMinWidth	整型	0	水平拖动的最小宽度
horizontalDragMaxWidth	整型	99999	水平拖动的最大宽度
contentWidth	整型		滚动条内容宽度
animateScroll	布尔	false	当调用滚动方法时使用动画效果
animateDuration	整型	300	动画滚动的持续时间，单位为毫秒
animateEase	字符串	'linear'	动画类型

续表

属　性	类型	默认值	说　明
hijackInternalLinks	布尔	false	是否在插件作用范围内使用书签链接
verticalGutter	整型	4	设置显示内容到垂直滚动条边框空格的个数
horizontalGutter	整型	4	设置显示内容到水平滚动条边框空格的个数
mouseWheelSpeed	整型	10	鼠标滚轮步进的位移量
arrowButtonSpeed	整型	10	单击箭头按钮移动的单位位移量
arrowRepeatFreq	整型	100	箭头按钮点击频率，单位为毫秒
arrowScrollOnHover	布尔	false	当鼠标悬停在箭头按钮上时，是否触发滚动
verticalArrowPositions	字符串	split	垂直滚动条箭头按钮出现的位置，可选值为 split|before|after|os
horizontalArrowPositions	字符串	splite	水平滚动条箭头按钮出现的位置，可选值为 split|before|after|os
enableKeyboardNavigation	布尔	true	是否可以使用键盘方向键操作滚动条
hideFocus	布尔	false	是否隐藏焦点，建议设为隐藏
clickOnTrack	布尔	true	是否允许单击改变滚动条滚动位置
trackClickSpeed	整型	30	鼠标单击一次的步进值
trackClickRepeatFreq	整型	100	鼠标单击间隔时间，单位为毫秒

滚动条有一些函数可以控制滚动条的相关操作，如表 10.5 所示。

表 10.5　jScrollPane插件的函数说明

函 数 形 式	说　明
reinitialise(s)	再次初始化滚动条，如果有新的属性设置则重新配置，如果没有则沿用上一次初始化的属性设置
scrollToElement(ele, stickToTop, animate)	将一个 jQuery 对象、DOM 节点或者 jQuery 选择器字符串所指向的元素移动到指定显示位置，如果 stickToTop 为真，则移动到可见位置的最上端，否则在可见位置下端，也可指定动画效果
scrollTo (destX, destY, animate)	设定滚动条滚动到内容的指定纵向横向位置，可以添加动画效果
scrollToX (destX, animate)	设定滚动条横向滚动到的位置，可以添加动画效果
scrollToY (destY, animate)	设定滚动条纵向滚动到的位置，可以添加动画效果
scrollToPercentX (destPercentX, animate)	设定滚动条横向滚动到占最大值指定百分比的位置，可以添加动画效果
scrollToPercentY (destPercentY, animate)	设定滚动条纵向滚动到占最大值指定百分比的位置，可以添加动画效果
scrollBy(deltaX, deltaY, animate)	设定横向纵向滚动的相对偏移量，单位为像素，可以添加动画效果
scrollByX(deltaX, animate)	设定横向滚动的相对偏移量，单位为像素，可以添加动画效果
scrollByY(deltaY, animate)	设定纵向滚动的相对偏移量，单位为像素，可以添加动画效果
positionDragX (x, animate)	水平拖动的位置，可以添加动画效果

续表

函 数 形 式	说　　明
positionDragY (y, animate)	垂直拖动的位置，可以添加动画效果
animate (ele, prop, value, stepCallback)	当滚动条动画到一个新位置时调用这个函数，ele 为使用动画的元素，prop 为使用动画的属性，value 为动画中的属性值，stepCallback 为更新属性值需要执行的方法
getContentPositionX()	返回滚动条的当前横向 *X* 轴位置
getContentPositionY()	返回滚动条的当前纵向 *Y* 轴位置
getContentWidth()	返回滚动条宽度
getContentHeight()	返回滚动条高度
getIsScrollableH()	返回判断是否有横向滚动条
getPercentScrolledX()	返回滚动条相对显示区域横向位置
getPercentScrolledY()	返回滚动条相对显示区域纵向位置
getIsScrollableV()	返回判断是否有纵向滚动条
getContentPane()	获得一个滚动条的引用
scrollToBottom(animate)	滚动条滚动到下端
hijackInternalLinks()	书签链接操作
destroy()	销毁滚动条

下面通过示例来介绍一下这个插件的使用方法。

1. 默认样式滚动条

这个示例使用滚动条的默认参数形式，只调用了滚动条插件的初始化函数，并适当加入了一些简单属性设置。其中，HTML 代码和 CSS 代码请参考光盘内容。JavaScript 功能代码如下：

```
<script type="text/JavaScript">
    $(function()
    {
        //设置默认属性选择值，显示上下移动箭头
        $.extend($.fn.jScrollPane.defaults, {showArrows:true});
        //上一行代码也可进行如下设定
        //$.fn.jScrollPane.defaults.showArrows = true;
        //插件初始化
        $('#pane1').jScrollPane();
        $('#pane2').jScrollPane();
        //将第 3 个插件的上下移动箭头取消，并设定滚动条的宽和间隔距离
        $('#pane3').jScrollPane({scrollbarWidth:20, scrollbar-
        Margin:10, showArrows:false});
        $('#pane4').jScrollPane();

        //向第 4 个插件中加入内容事件
        //当向其中加入内容时，会要求重新初始化这个插件
        $('#add-content').bind(
            'click',
            function()
```

```
20                  {
21                      $('#pane4').append($('<p></p>').html($('#intro').
                        html())).jScrollPane();
22                  }
23              );
24              //从第 4 个插件中删除内容事件
25              //这里也会要求重新初始化插件
26              $('#remove-content').bind(
27                  'click',
28                  function()
29                  {
30                      $('#pane4').empty().append($('<p></p>').html
                        ($('#intro').html())).jScrollPane();
31                  }
32              );
33          });
34
35      </script>
```

上述代码第 5 行设定了插件的显示箭头按钮的属性。第 9、10、12、13 行初始化了 4 个滚动条。第 12 行配置了滚动条的宽为 20 像素，间隔为 10 像素，不显示箭头按钮。第 17～23 行绑定了向第 4 个滚动条所在区域添加内容的事件。第 26～32 行绑定了从第 4 个滚动条所在区域删除内容的事件。效果如图 10.11 和图 10.12 所示。

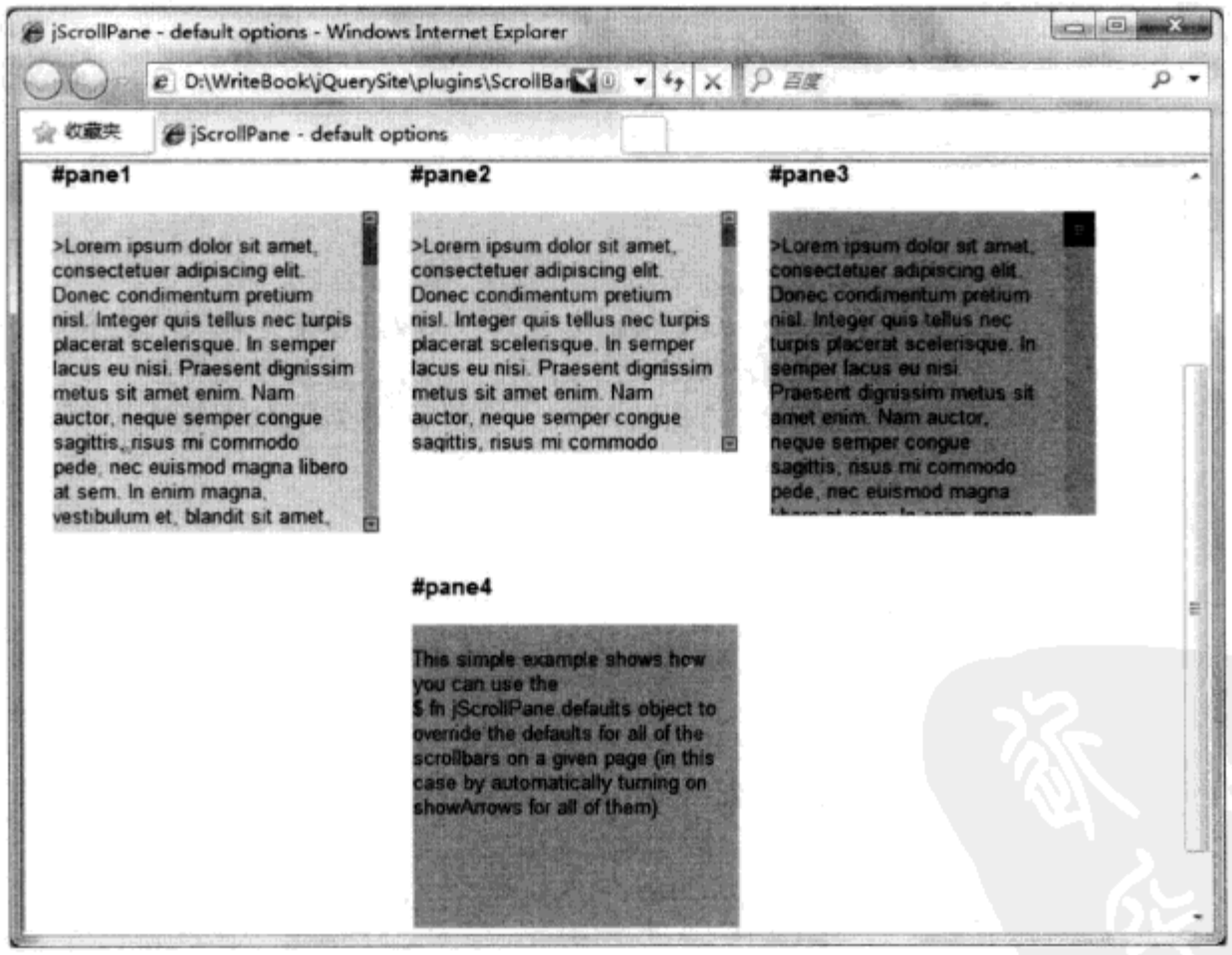


图 10.11　通过默认选项创建滚动条

2. 自定义浏览器滚动条

这个示例利用 jScrollPane 创建的滚动条来代替浏览器默认的滚动条。在某些情况下浏

览器自身的滚动条的样式满足不了需要，此时可以使用这个插件创建属于我们自己的浏览器滚动条。JavaScript 功能代码如下：

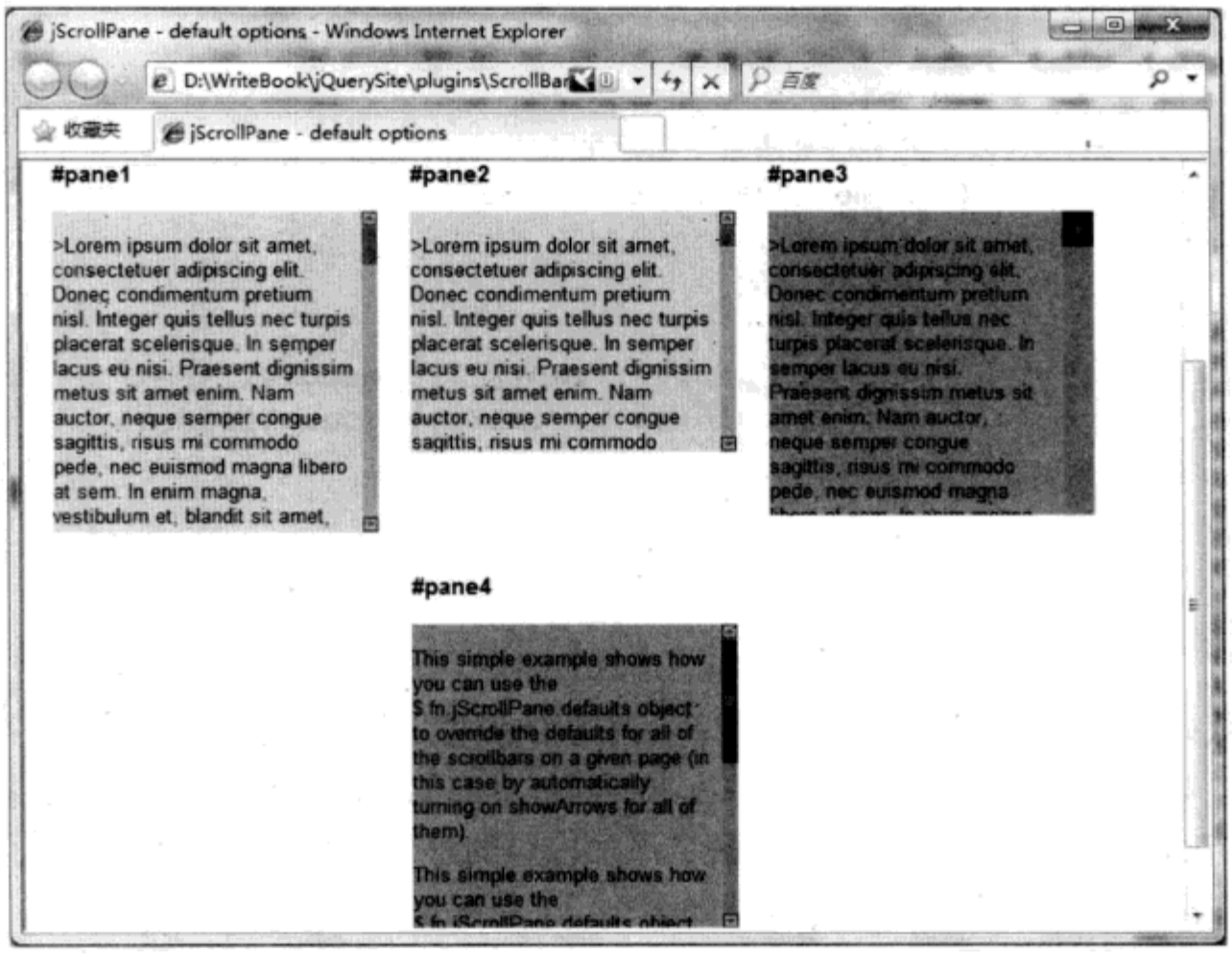


图 10.12　向滚动条所在区域动态添加内容后效果

```
<script type="text/JavaScript">
    $(function()
    {
        $.extend($.fn.jScrollPane.defaults, {
            showArrows:true,          //显示表示滑动方向的箭头
            scrollbarWidth:15,        //设定滑动条宽度
            arrowSize:16              //设置箭头图标大小
            }
        );
        var isResizing;
        //获取整个容器高度
        var setContainerHeight = function()
        {
            //IE 触发浏览器大小改变事件
            //避免浏览器冲突
            if (!isResizing) {
                isResizing = true;
                $w = $(window);
                $c = $('#container');
                var p = (parseInt($c.css('paddingLeft')) || 0) +
                (parseInt($c.css('paddingRight')) || 0);
                $('body>.jScrollPaneContainer').css({'height':
                $w.height() + 'px', 'width': $w.width() + 'px'});
                $c.css({'height': ($w.height()-p) + 'px', 'width':
                ($w.width() - p) + 'px', 'overflow':'auto'});
```

```
23                      $c.jScrollPane();
24                      isResizing = false;
25                  }
26              }
27              $(window).bind('resize', setContainerHeight);
28              setContainerHeight();
29
30              //这里需要再次调用获取容器高度函数
31              setContainerHeight();
32          });
33      </script>
```

上述代码第 4～9 行对滚动条设置宽度为 15 像素，滚动条使用箭头按钮，按钮大小为 16 像素。第 12～26 行定义了对浏览器添加自定义滚动条函数。其中第 17 行设定浏览器窗口可调整尺寸标志。第 18 行获取浏览器窗口对象。第 19 行获取最底层容器层对象。第 20 行获取文字内容与边框之间的距离。第 21 行设置滚动条的尺寸。第 22 行设置容器层的尺寸。第 23 行初始化滚动条。第 27 行绑定窗口的改变大小事件。最后调用添加自定义滚动条函数，文档中说明这个函数要调用两次。效果如图 10.13 所示。

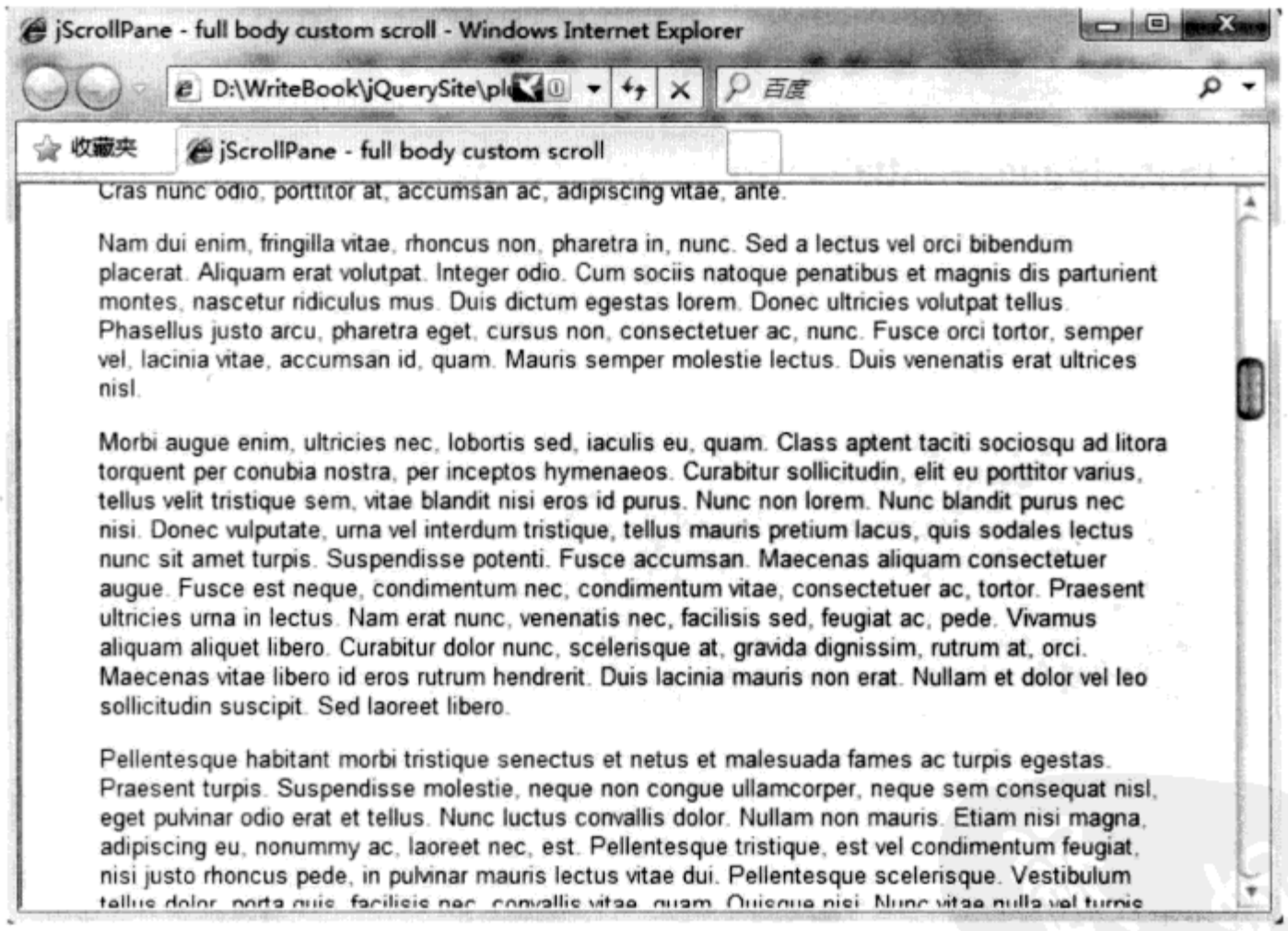


图 10.13　自定义浏览器窗口滚动条

3．重新初始化属性的应用

这个示例使用了插件的重新初始化属性，要实现当图片加载完成后再次初始化操作。当图片或者内容大小较大时，加载时间会有延迟，因此当图片全部加载完成或者内容全部加载后，需要根据实际内容的大小重新初始化滚动条的大小及滚动范围。JavaScript 功能代码如下：

```
1       <script type="text/JavaScript">
2           $(function()
3           {
4               //一旦图片被完全加载，则重新初始化插件
5               $('#pane1')
6                   .jScrollPane(
7                       {
8                           showArrows:true,    //显示表示滑动方向箭头
9                           scrollbarWidth: 20,//滑动条宽度
10                          reinitialiseOnImageLoad: true
                                                //图片加载后重新初始化插件
11                      }
12                  );
13          });
14      </script>
```

上述代码第 10 行就是再次初始化选项，在这里设置为真则表示允许滚动条再次初始化。效果如图 10.14 所示。

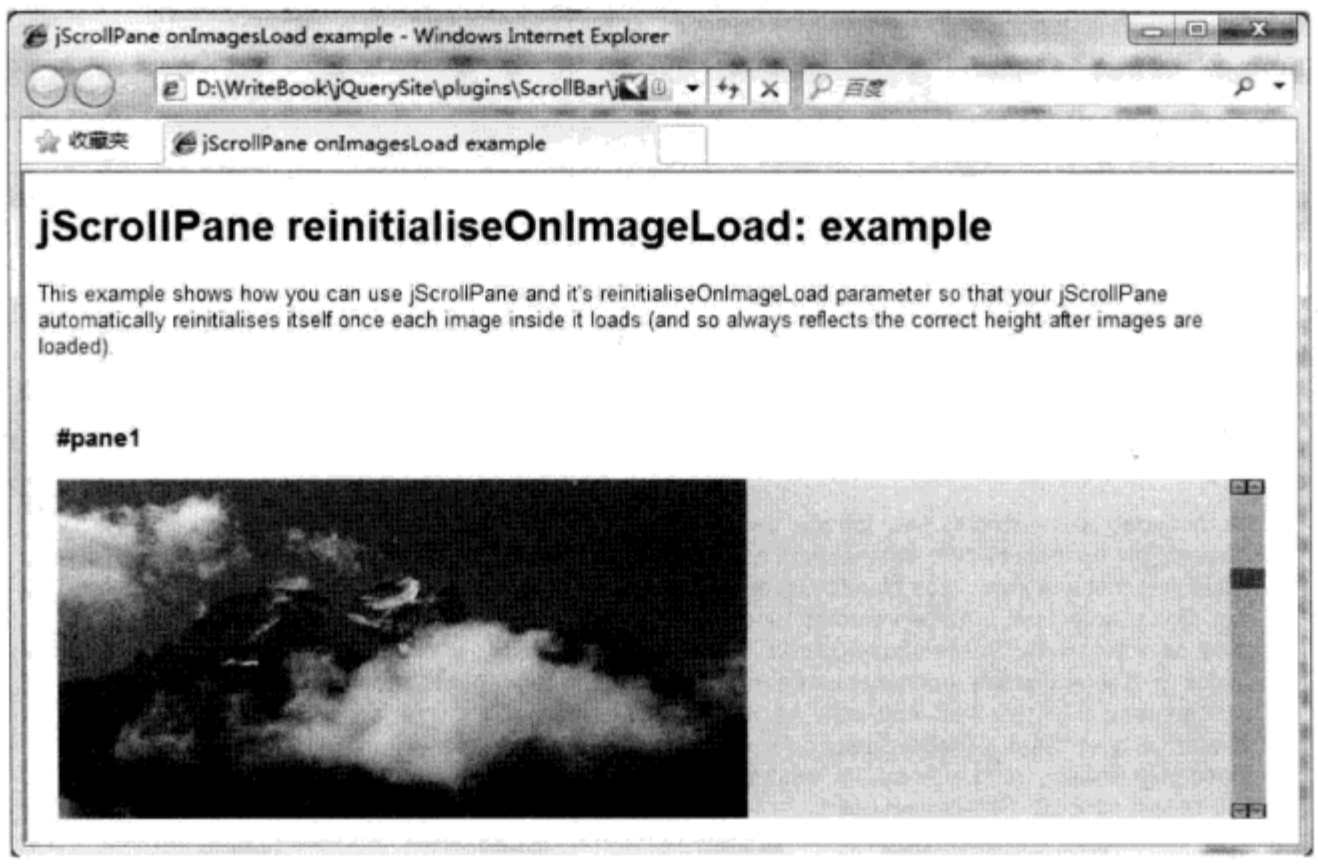


图 10.14　再次加载滚动条

4. 滚动事件举例

这个示例使用了滚动条的滚动事件，并在这个事件中添加了记录浏览器日志的操作。JavaScript 功能代码如下：

```
1       <script type="text/JavaScript">
2           $(function()
3           {
4               //简单初始化插件
5               $('#pane1').jScrollPane();
6               $('.scroll-pane').bind(     //绑定滑动事件
```

```
7                    'scroll',
8                    function(event)
9                    {
10                       console.log(event.target);
11                   }
12               )
13           });
14       </script>
```

上述代码第 6～12 行绑定了滚动条的滚动事件。第 7 行指定绑定事件。第 10 行表示在滚动事件发生时记录浏览器日志。效果如图 10.15（此图效果是在 IE 8 中实现的）所示。

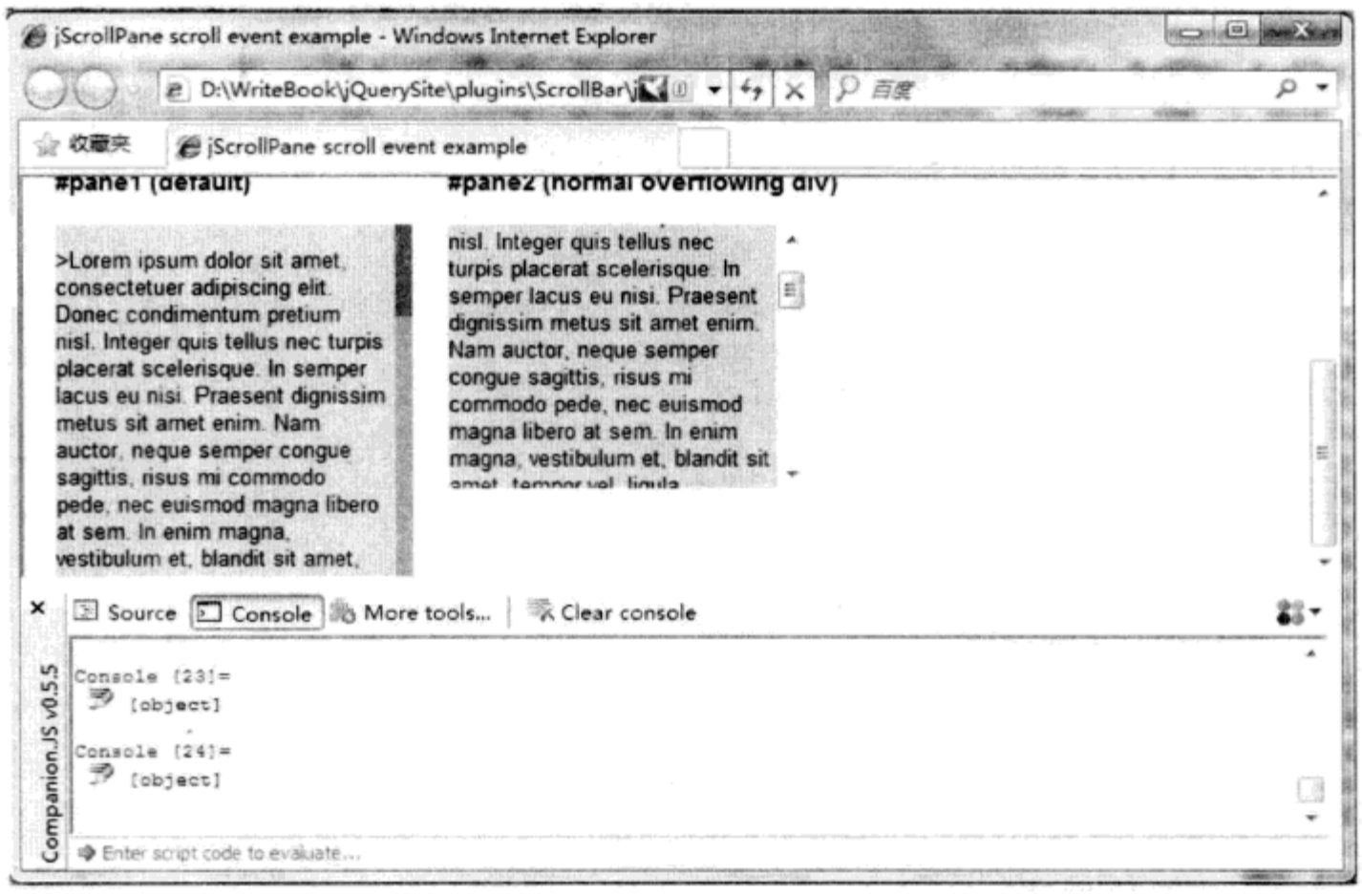


图 10.15　利用滚动条滚动事件记录浏览器日志

5. 滚动条编程设定滚动位置

这个示例使用了滚动条更改滚动位置的相关操作函数，通过编程动态修改滚动条位置。也可以通过外界元素的变化影响滚动条的变化，更改滚动块的位置。JavaScript 功能代码如下：

```
1        <script type="text/JavaScript">
2            $(function()
3            {
4                //分别初始化三个插件
5                $('#pane1').jScrollPane();
6                $('#pane2').jScrollPane({showArrows:true});
7                $('#pane3').jScrollPane({scrollbarWidth:20, scrollbar-
                 Margin:10, animateTo:true});
8               //初始化页面其他链接的单击事件，修改滚动块的位置
9               $('.scroll-to-demo').bind(
```

```
        'click',
        function()
        {
            $('#pane1')[0].scrollTo(200);
            return false;
        }
    );
    $('.scroll-by-demo').bind(
        'click',
        function()
        {
            $('#pane2')[0].scrollBy(parseInt($(this).
            attr('rel')));
            return false;
        }
    );
    var $targets = $('#pane3-scroll-targets');
    var $pane3 = $('#pane3');
    var pane3top = parseInt($pane3.offset().top);
    $('#pane3 p').each(
        function(index)
        {
            $targets.append(
                $('<li></li>').append(
                    $('<a></a>')
                        .attr({'href':'JavaScript:;', 'rel':
                        $(this).offset().top})
                        .text('Scroll to paragraph ' + (index+1))
                        .bind(
                            'click',
                            function()
                            {
                                $pane3[0].scrollTo(parseInt
                                ($(this).attr('rel')) - pane3top);
                            }
                        )
                    )
                );
        }
    );
    //对第 4 个插件设定滚动块的编程改变位置事件
    var $pane4 = $('#pane4');
    $pane4.jScrollPane({animateTo:true});
    $('a.scroll-to-element-demo').bind(
        'click',
        function()
        {
            var targetElementSelectorString = $(this).
            attr('rel');
            $pane4[0].scrollTo(targetElementSelectorString);
                                //设定滑动到指定位置
            return false;
```

```
57                    }
58                );
59            });
60        </script>
```

上述代码第 5、6、7 行和第 49 行分别初始化了 4 个滚动条，其中第 3 个滚动条设定了滚动条宽度为 20 像素，边框间隔为 10 像素，使用动画。第 4 个滚动条也使用了动画。第 9～16 行设定第 1 个滚动条动态滚动事件，将滚动块定位在 200 位置。第 17～24 行设定了第 2 个滚动条动态滚动事件，利用链接元素的单击事件改变滚动块位置，每次都以链接的 rel 属性值为移动量。

第 25～46 行定义了对第 3 个滚动条的滚动触发事件，并以每一段落的起始行位置为移动位移量。第 48～58 行定义了第 4 个滚动条的滚动触发事件，移动的位置是通过 jQuery 选择器而不是通过坐标位置定位的。效果如图 10.16 和图 10.17 所示。

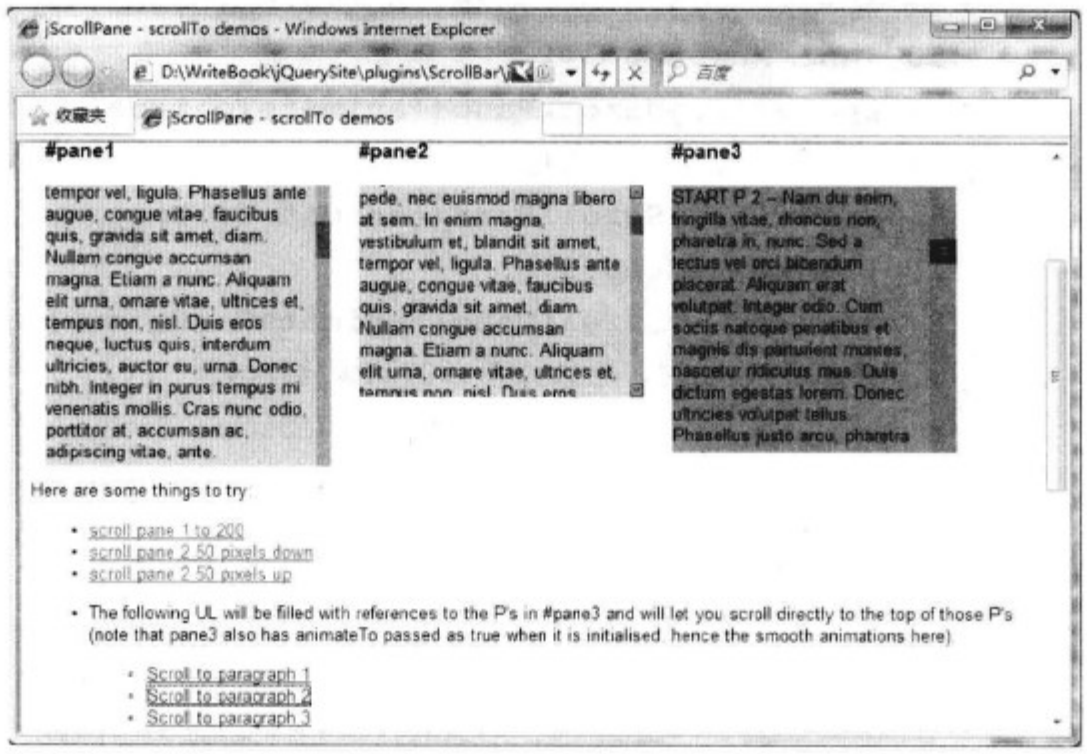


图 10.16　编程调用滚动方法效果一

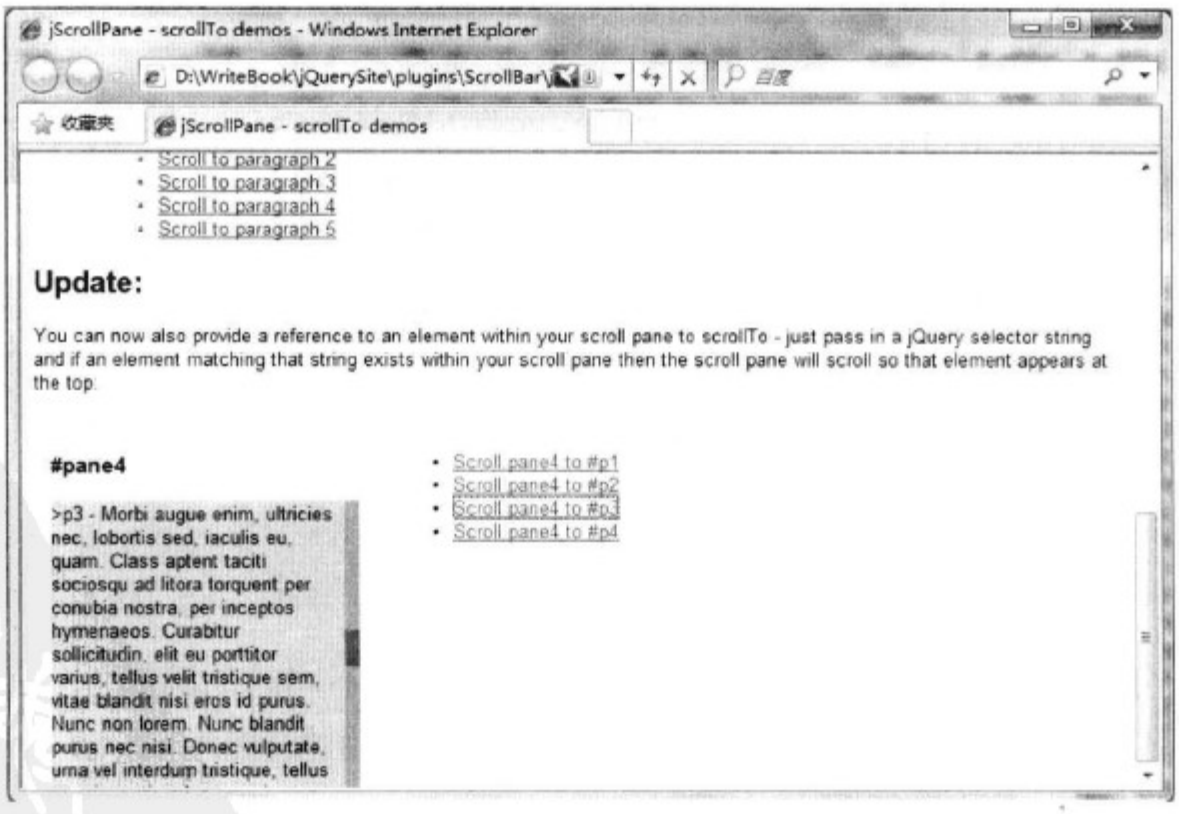


图 10.17　编程调用滚动方法效果二

6. 加入动画效果的滚动

这个示例通过设定不同的超链接的单击事件，使对应的滚动条以不同的动画形式滚动定位到不同位置。滚动条动画效果的移动也是一种全新的用户体验。JavaScript 功能代码如下：

```
1       <script type="text/JavaScript">
2           $(function()
3           {
                //初始化各个插件
                //设定动画间隔时间，以及动画移动时的单位移动距离
4               $('#pane1').jScrollPane({animateTo:true, animateInterval:
                50, animateStep:5});
5               $('#pane2').jScrollPane({animateTo:true, animateInterval:
                50, animateStep:3});
6               $('#pane3').jScrollPane({animateTo:true, animateInterval:
                150, animateStep:5});
7               $('#pane4').jScrollPane({animateTo:true, animateInterval:
                150, animateStep:3});
8               $('#pane5').jScrollPane({animateTo:false});
                //超链接单击事件，触发滚动条的动画移动效果
9               $('a.scroll-to-element-demo').bind(
10                  'click',
11                  function()
12                  {
13                      $this = $(this);
14                      var destinationSelector = $(this).attr('rel');//
15                      $('.scroll-pane', $this.parent().parent().
                        parent()).each(
16                          function()
17                          {
18                              this.scrollTo(destinationSelector);
                                //设定滑动到指定位置
19                          }
20                      );
21                      return false;
22                  }
23              );
24
25          });
26      </script>
```

上述代码第 4～8 行初始化了 5 个滚动条，它们具有不同的动画属性。第 9～23 行对每个滚动条都使用了 4 个超链接，利用超链接的单击事件触发滚动条滚动，并定位到每个段落的起始位置。静态效果如图 10.18 所示。

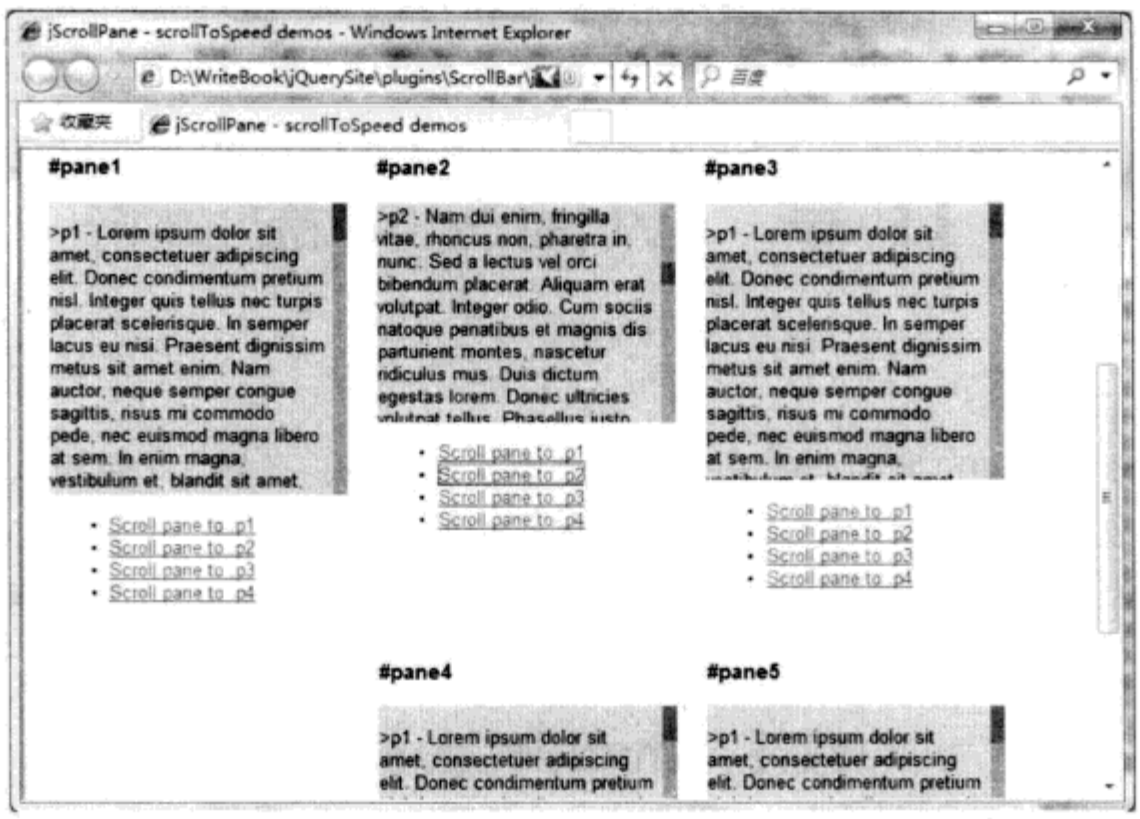


图 10.18　设置不同动画效果的滚动条滚动静态效果

10.3　小　　结

本章对如何用 jQuery 创建滑动条进行了讲解。主要内容包括基本滑动条实现原理、滑动条的事件过程处理，并介绍了两个滑动条插件。重点部分是滑动条的实现原理及滑动条插件的使用方法。滑动条的实现原理是本章的难点。下一章将讲解 jQuery 页面编辑器插件。

10.4　习　　题

【习题 1】利用本章所讲解内容自定义实现横向滑动条。

【习题 2】练习 jScrollPane 插件的使用方法。

第 11 章　页面编辑器插件

页面编辑器是在浏览器中创建一个具有文本编辑功能、类似文本编辑软件的区域。这种页面效果可以应用于网站论坛、网站后台管理、B/S 结构的 OA 平台等 CMS。它方便用户在浏览器中编辑文章，并在编辑完成后直接发表。本章将介绍 6 个页面编辑器插件。

11.1　markItUp 插件

markItUp 插件是一个轻量级的、具有客户定制并富有弹性的编辑引擎、面向开发人员的 jQuery 插件。它可应用于 CMS（内容管理系统）、博客、网站论坛等。这个插件的英文网址为 http://markitup.jaysalvat.com/home/，可以在该网址下载插件文件、文档及相关示例。

该插件的特点如下。

（1）可快捷地整合进新的项目中。

（2）支持键盘快捷键。

（3）具有工具栏和下拉菜单。

（4）灵活的客户定制。

（5）使用的随意性。

（6）Ajax 动态预览。

（7）可定制主题。

下面就来介绍这个插件的使用方法。

11.1.1　安装插件

在安装过程中，在引入插件文件之前需要加载 jQuery 库文件，并在引入插件文件之后继续引用插件的配置文件，如下所示。

（1）引入 JS 文件：

```
<!-- jQuery -->
<script type="text/JavaScript" src="../../../jslib/jquery-1.6.js"></script>
<!-- markItUp! -->
<script  type="text/JavaScript"  src="markitup/jquery.markitup.js"></script>
                                        //引入插件文件
```

（2）引入配置插件的 JSON 文件：

```
<!-- markItUp! toolbar settings -->
<script type="text/JavaScript" src="markitup/sets/default/set.js"></script>
                                                    //引入插件要使用的 JSON 文件
```

（3）引入 CSS 样式文件：

```
<!-- markItUp! skin -->
<link rel="stylesheet" type="text/css" href="markitup/skins/markitup/
style.css" />
<!--  markItUp! toolbar skin -->
<link rel="stylesheet" type="text/css" href="markitup/sets/default/style.
css" />
```

（4）加入插件调用 JavaScript 功能代码。在调用插件初始化方法时，可以使用元素 id、元素的 class、元素标记名：

```
<script type="text/JavaScript">
<!--
    $('#markItUp').markItUp(mySettings);    //初始化插件
-->
</script>
```

11.1.2　键盘的使用

键盘的使用主要有以下几种。

（1）Ctrl+其他键，标记插入。

（2）Shift+其他键，新标记的插入。

（3）Ctrl+Shift+其他键，在选中的所有行添加标记。

（4）Alt+其他键，使用可选标记插入。

其中，“其他键”为除去 Ctrl、Shift、Alt 等键的按键，主要是标记使用的键盘快捷操作。例如，当需要在编辑器内添加一对粗体字标记时，可以使用 Ctrl+B 组合键来操作。

11.1.3　标记语言定制

标记语言定制主要设定插件可以处理的是何种标记语言、这种标记语言的语法如何，以及用户在编辑器中插入该标记语言时的行为特征等。标记语言定制包含若干属性，如表 11.1 所示。

表 11.1　标记语言定制的属性说明

属　性	类　型	说　明
nameSpace	字符串	为封装的 DIV 指定一个类名，避免 CSS 样式冲突
resizeHandle	布尔	是否允许调整编辑器大小
previewAutoRefresh	布尔	当编辑内容改变时，是否刷新预览窗口
previewParserPath	字符串	自定义的预览语法分析
previewParserVar	字符串	发送给语法分析的内容名称
previewTemplatePath	字符串	预览模板位置

续表

属　性	类　型	说　明
previewParser	函数	预览前分析内容的语法函数
previewPosition	字符串	预览窗口的位置
onEnter		定义当回车键按下时的操作
onCtrlEnter		定义当 Ctrl 和回车键同时按下时的操作
onShiftEnter		定义当 Shift 和回车键同时按下时的操作
onTab		定义当 Tab 键按下时的操作
beforeInsert	函数	定义在插入标记前执行的函数
afterInsert	函数	定义在插入标记后执行的函数
markupSet		标记集合操作为编辑器设置属性

markupSet 中的属性说明如表 11.2 所示。

表 11.2　markupSet中的属性说明

属　性	类　型	说　明
name	字符串	按钮名称
classname	字符串	按钮应用的类名
key	字符串	按钮对应的快捷键
openWith	字符串或函数	在选择位置前标记添加
closeWith	字符串或函数	在选择位置后标记添加
replaceWith	字符串或函数	在鼠标或选择位置替换文本
openBlockWith	字符串或函数	在完整的块前加入文本
closeBlockWith	字符串或函数	在完整的块后加入文本
mutiline	布尔	标记是否插入每一行或者一个完整的块
placeHolder	字符串或函数	占位符文本被插入
beforeInsert	函数	标记插入前被调用的函数
afterInsert	函数	标记插入后被调用的函数
beforeMutiInsert	函数	多行标记插入前调用的函数
afterMutiInsert	函数	多行标记插入后调用的函数
dropMenu		打开下拉菜单的按钮

下面用一个示例来介绍这个插件的使用方法。这个示例主要是想介绍初始化插件时如何通过配置文件配置插件样式及功能。在这个示例中使用了插件的开始标记、结束标记、填充文本内容等行为属性，以及如何通过插件内置删除函数在页面中删除插件效果。下面看一下 JavaScript 的功能代码：

```
1   <script type="text/JavaScript">
    <!--
2   $(document).ready(function()    {
    //添加插件及应用配置信息
    //$('textarea').markItUp( { Settings }, { OptionalExtraSettings } );
3       $('#markItUp').markItUp(mySettings);
```

```
    //可以在任何位置添加任意内容
    //$.markItUp( { Settings } );
4       $('.add').click(function() {
5           $.markItUp( {   openWith:'<opening tag>',
6                       closeWith:'<closing tag>',
7                       placeHolder:"New content"
8                   }
9               );
10      return false;
11      });

    //在任意时刻添加或者删除插件效果
    //$(textarea).markItUpRemove();
12      $('.toggle').click(function() {
13          if ($("#markItUp.markItUpEditor").length === 1) {
14              $("#markItUp").markItUpRemove();
15              $("span", this).text("get markItUp! back");
16          } else {
17              $('#markItUp').markItUp(mySettings);
18              $("span", this).text("remove markItUp!");
19          }
20          return false;
21      });
22  });
    -->
23  </script>
```

上述代码第 3 行初始化了这个页面编辑器，其中利用了 set.js 文件中的设置内容。第 4～11 行是向编辑器中添加带有标记的文本。第 5 行进行编辑器的参数设定，设定文本内容的起始标记。第 6 行设定文本内容的结束标记。第 7 行为文本占位符；第 12～21 行，设定对编辑器的添加与删除操作。第 13 行判定页面中是否已经存在编辑器。第 14、15 行表示如果存在编辑器则删除，并修改单击元素的文本。第 17、18 行表示如果页面中不存在编辑器则创建，同时修改单击元素的文本。

为了对上面讲解的编辑器的相关属性有所了解，下面来看一下如何对编辑器进行常用配置，以 set.js 文件为例。

```
1 var mySettings = {
2   onShiftEnter:   {keepDefault:false, replaceWith:'<br />\n'},
3   onCtrlEnter:    {keepDefault:false, openWith:'\n<p>', closeWith:
    '</p>'},
4   onTab:          {keepDefault:false, replaceWith:'    '},
5   markupSet:  [
6       {name:'Bold', key:'B', openWith:'(!(<strong>|!|<b>)!)', closeWith:
        '(!(</strong>|!|</b>)!)' },
7       {name:'Italic', key:'I', openWith:'(!(<em>|!|<i>)!)', closeWith:
        '(!(</em>|!|</i>)!)'  },
8       {name:'Stroke through', key:'S', openWith:'<del>', closeWith:
        '</del>' },
9       {separator:'---------------' },
```

```
10      {name:'Bulleted List', openWith:'    <li>', closeWith:'</li>',multi-
        line:true, openBlockWith:'<ul>\n', closeBlockWith:'\n</ul>'},
11      {name:'Numeric List', openWith:'    <li>', closeWith:'</li>', multi-
        line:true, openBlockWith:'<ol>\n', closeBlockWith:'\n</ol>'},
12      {separator:'---------------' },
13      {name:'Picture', key:'P', replaceWith:'<img src="[![Source:!:http:
        //]!]" alt="[![Alternative text]!]" />' },
14      {name:'Link', key:'L', openWith:'<a href="[![Link:!:http:
        //]!]"(!( title="[![Title]!]")!)>', closeWith:'</a>', placeHolder:
        'Your text to link...' },
15      {separator:'---------------' },
16      {name:'Clean', className:'clean', replaceWith:function(markitup)
        { return markitup.selection.replace(/<(.*?)>/g, "") } },
17      {name:'Preview', className:'preview',  call:'preview'}
18  ]
19  }
```

上面这段 JavaScript 代码是编辑器的属性设定内容。第 2 行对 Shift 和回车键同时按下取消默认操作，在编辑器中使用'
\n'来代替。第 3 行对 Ctrl 和回车键同时按下取消默认操作，在编辑器中使用段落标记来代替。第 4 行对 Tab 键的按下取消默认操作，使用 4 个空格代替。第 6 行定义添加粗体字标记按钮。第 7 行定义添加斜体字标记按钮。第 8 行定义添加删除线标记按钮。

第 9 行定义分隔线样式。第 10 行定义添加无序列表样式标记按钮。第 11 行定义添加有序列表样式标记按钮。第 12 行定义分隔线样式。第 13 行定义添加图片元素标记按钮。第 14 行定义能够添加超链接元素标记按钮。第 15 行定义分隔线样式。第 16 行定义清空前后标记按钮功能。第 17 行定义展现预览视图。效果如图 11.1 和图 11.2 所示。

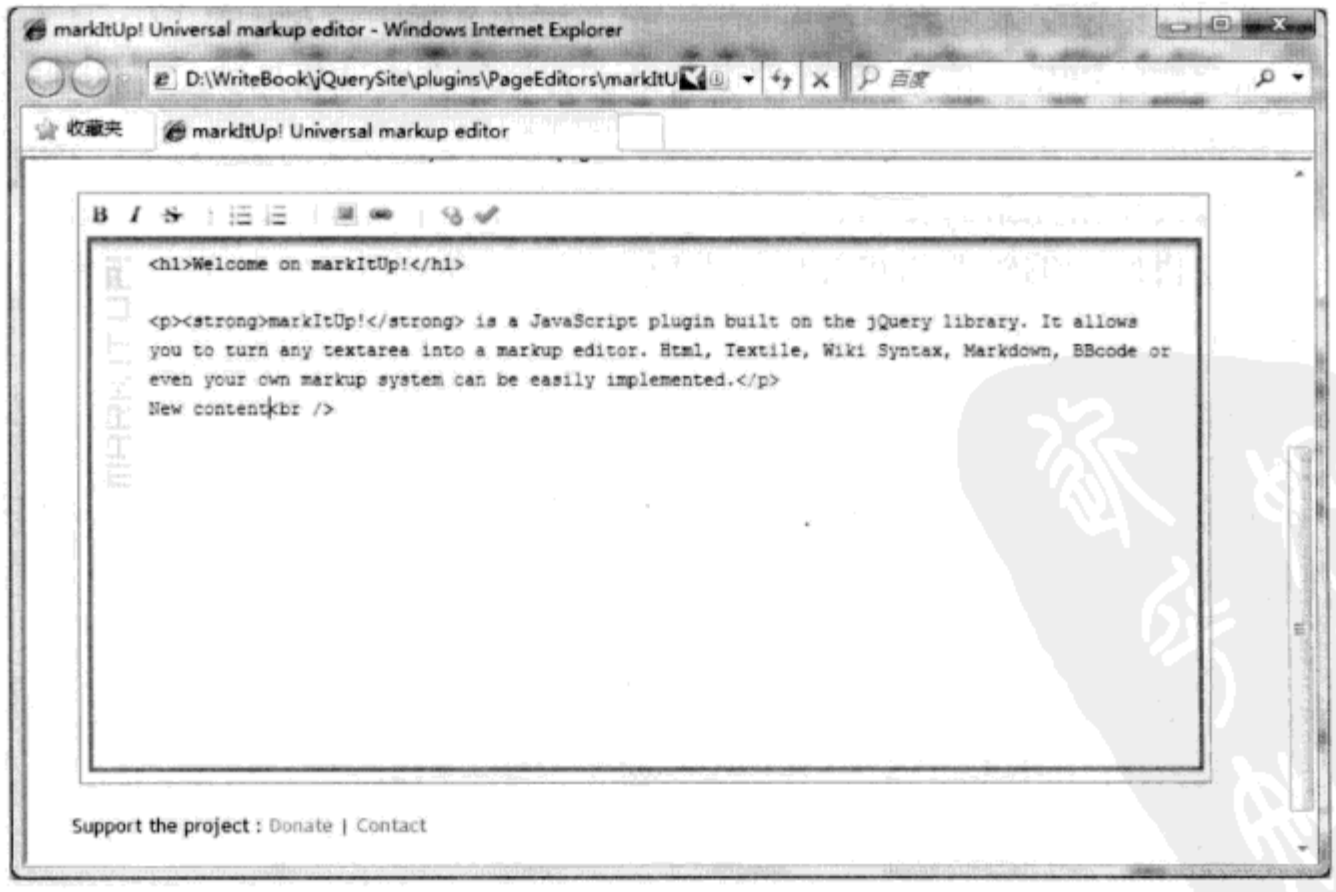


图 11.1　markItUp 编辑器样式

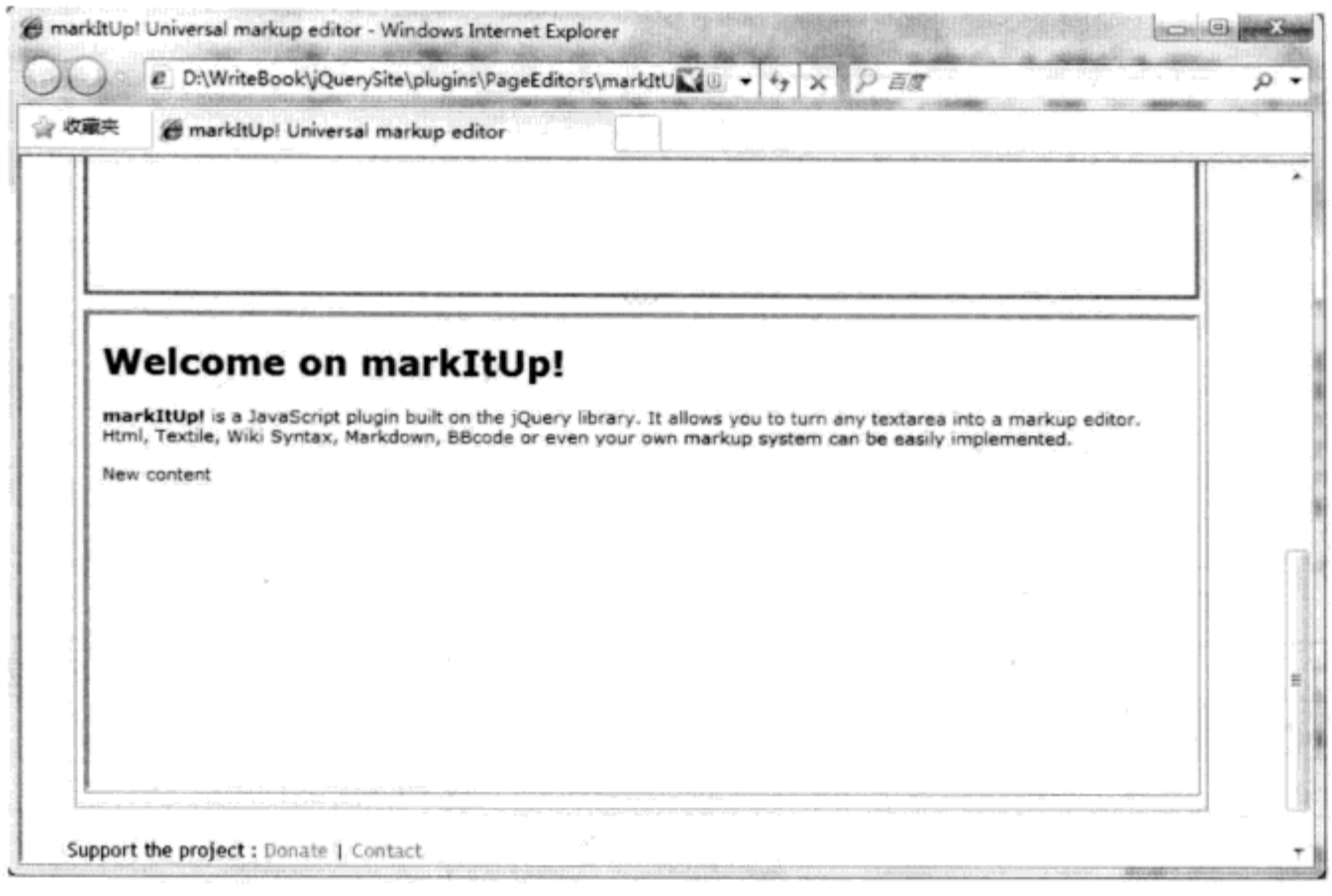


图 11.2　markItUp 预览样式

11.2　jwysiwyg 插件

jwysiwyg 插件是一个所见即所得的 jQuery 页面编辑器。使用这个编辑器需要引用 jQuery 1.3.2 以后版本，还需要 jQuery.UI.Dialog 和 jQuery.UI.Resizable 库文件。这个插件的英文网址为 http://github.com/akzhan/jwysiwyg。

11.2.1　jwysiwyg 插件基本介绍

这个插件主要通过配置属性与函数来控制它的工作，具体属性说明如表 11.3 所示。

表 11.3　jwysiwyg插件的属性说明

属　性	类　型	默 认 值	说　明
html	字符串		在编辑器内的 HTML 源代码
debug	布尔		是否可以调试
css	字符串		编辑器使用的 CSS 样式文件的路径
autoGrow	布尔		框架尺寸自动增长
autoSave	布尔		是否自动保存
brIE	布尔		如果为真，在 IE 中插入 用于创建新行
formHeight	整型		对话框的高
formWidth	整型		对话框的宽
iFrameClass	字符串		使用的 iFrame 的样式类

续表

属　性	类　型	默 认 值	说　明
initialContent	字符串	Initial Content	编辑器的初始内容
maxHeight	整型		自动增长的最大高度
maxLength	整型		编辑器可接收的最多字符个数，但不包括 HTML 标记
messages	jQuery 对象，形式为键/值对		自定义的消息内容
plugins			插件使用基本设置
autoload	布尔或者对象		是否自动加载默认插件对象，或者指定其他加载对象
i18n	布尔或者对象		是否适用国际化，也可指定语言对象
rmMSWordMarkup	布尔		是否移除 MS Word 标记
toolbarHtml	字符串		HTML 源代码
resizeOptions	布尔		是否允许编辑器调整大小
removeHeadings	布尔		是否移除标题
rmUnusedControls	布尔		是否移除未在控制属性集合中未启用的功能（参见表 11.5）
rmUnwantedBr	布尔		是否添加 标记
tableFiller	字符串		添加表格时单元格中的默认文本
events	事件对象		为插件指定事件，键为事件名，值为回调函数
controls	对象		功能按钮集合

插件具有内置函数，为我们操作插件提供了方便，如表 11.4 所示。

表 11.4　jwysiwyg插件的内置函数说明

函　数	说　明
addControl(name,settings)	添加控制按钮
clear	清除内容
createLink(szURL)	创建超链接
destroy	销毁编辑器对象
document	编辑器中的文档
getContent	获取编辑器内容
insertImage(szURL,attribute)	插入图片
insertTable(colCount,rowCount,filler)	添加表格
removeFormat	移除格式
save	存储编辑器中文本的改变
selectAll	选中编辑器中所有内容
setContent	设置编辑器中文本内容

编辑器中包含一些控制器按钮，为用户提供文本编辑操作，效果图 11.3 是常用按钮样式。按钮说明如表 11.5 所示。

图 11.3　常用功能按钮

表 11.5　jwysiwyg插件的按钮说明

按　钮	说　明	按　钮	说　明
bold	粗体字	insertHorizontalRule	插入水平线
italic	斜体字	createLink	创建超链接
strikeThrougn	删除线	insertImage	插入图片
underline	下划线	h1	添加 h1 标题
justifyLeft	左对齐	h2	添加 h2 标题
justifyCenter	居中对齐	h3	添加 h3 标题
justifyRight	右对齐	paragraph	添加段落或者 h1～h6 标题
justifyFull	全对齐	cut	剪切
indent	文本缩进	copy	复制
outdent	文本突出	paste	粘贴
subscript	文本下标	increaseFontSize	字号变大
supscript	文本上标	decreaseFontSize	字号变小
undo	撤销操作	html	文本区域的 HTML 源代码
redo	重做	removeFormat	移除所有格式
insertOrderedList	插入有序列表	insertTable	插入表格
insertUnorderedList	插入无序列表		

下面通过示例，学习如何使用该插件。本书所有的效果图都是基于 IE8 的，这个插件如果在 IE 环境下使用，需要打开 jquery.wysiwyg.css 文件，将 jquery.wysiwyg.gif 修改为 jquery.wysiwyg.no-alpha.gif，才能看到正常效果。

11.2.2　基本应用

这个示例使用了不带任何参数设置的初始化参数。在这里使用默认编辑器的形态，加载了常用按钮，使用基本功能。在 HTML 文件头部分需要引入下列文件。

```
<link rel="stylesheet" type="text/css" href="../lib/blueprint/screen.css"
media="screen, projection" />
<link rel="stylesheet" type="text/css" href="../lib/blueprint/print.css"
media="print" />
<!--[if lt IE 8]><link rel="stylesheet" href="../lib/blueprint/ie.css"
type="text/css" media="screen, projection" /><![endif]-->
<link rel="stylesheet" href="../../jquery.wysiwyg.css" type="text/css"/>
                                        //插件的 CSS 文件
<script type="text/JavaScript" src="../lib/jquery.js"></script>
<script  type="text/JavaScript"  src="../../jquery.wysiwyg.js"></script>
                                        //插件文件
<script type="text/JavaScript" src="../../controls/wysiwyg.image.js"></script>
```

```
                                                    //插件图片操作文件
<script type="text/JavaScript" src="../../controls/wysiwyg.link.js"></script>
                                                    //插件链接操作文件
<script type="text/JavaScript" src="../../controls/wysiwyg.table.js"></script>
                                                    //插件表格文件
```

JavaScript 功能代码相对简单：

```
1   <script type="text/JavaScript">
2   (function($) {
3       $(document).ready(function() {
4           $('#wysiwyg').wysiwyg();    //插件初始化
5       });
6   })(jQuery);
7   </script>
```

第 4 行调用插件初始化函数。效果如图 11.4 所示。

图 11.4　jwysiwyg 插件的基本应用效果

11.2.3　全功能编辑器

这个示例在插件初始化过程中将插件所支持的功能全部加载。这里使用了前面介绍的关于插件中添加标记按钮的属性，并为一些标记属性（如标题按钮）指定了依据不同浏览器实现功能的样式，以及插件的单击实现效果。JavaScript 功能代码如下：

```
<script type="text/JavaScript">
(function($) {
  $(document).ready(function() {
      $('#wysiwyg').wysiwyg({                    //配置插件所有功能
        controls: {
          bold            : { visible : true },
          italic          : { visible : true },
          underline       : { visible : true },
          strikeThrough   : { visible : true },

          justifyLeft     : { visible : true },
          justifyCenter   : { visible : true },
          justifyRight    : { visible : true },
          justifyFull     : { visible : true },
          indent  : { visible : true },
          outdent : { visible : true },
          subscript   : { visible : true },
          superscript : { visible : true },

          undo : { visible : true },
          redo : { visible : true },

          insertOrderedList    : { visible : true },
          insertUnorderedList  : { visible : true },
          insertHorizontalRule : { visible : true },
          h4: {
              visible: true,
              className: 'h4',
              command: ($.browser.msie || $.browser.safari) ?
                  'formatBlock' : 'heading',
              arguments: ($.browser.msie || $.browser.safari) ? '<h4>' : 'h4',
              tags: ['h4'],
              tooltip: 'Header 4'
          },
          h5: {
              visible: true,
              className: 'h5',
              command: ($.browser.msie || $.browser.safari) ?
              'formatBlock' : 'heading',
              arguments: ($.browser.msie || $.browser.safari) ? '<h5>' : 'h5',
              tags: ['h5'],
              tooltip: 'Header 5'
          },
          h6: {
              visible: true,
```

```
44              className: 'h6',
45              command: ($.browser.msie || $.browser.safari) ?
                'formatBlock' : 'heading',
46              arguments: ($.browser.msie || $.browser.safari) ? '<h6>' : 'h6',
47              tags: ['h6'],
48              tooltip: 'Header 6'
49          },
50
51          cut   : { visible : true },
52          copy  : { visible : true },
53          paste : { visible : true },
54          html  : { visible: true },
55          increaseFontSize : { visible : true },
56          decreaseFontSize : { visible : true },
57          exam_html: {
58              exec: function() {
59                  this.insertHtml('<abbr title="exam">Jam</abbr>');
60                  return true;
61              },
62              visible: true
63          }
64      },
65          events: {
66            click: function(event) {    //插件单击事件
67                if ($("#click-inform:checked").length > 0) {
68                    event.preventDefault();
69                    alert("You have clicked jWysiwyg content!");
70                }
71            }
72          }
73      });
74      $('#wysiwyg').wysiwyg("insertHtml", "Sample code");
                                                    //设定插件内文本内容
75  });
76  })(jQuery);
77  </script>
```

上述代码第 5 行开始设定编辑器的控制按钮功能。第 6～25 行设定粗体、斜体、下划线、删除线、左对齐、右对齐、居中对齐、全对齐、内缩进、外突出、上标、下标、取消、重做、有序列表插入、无序列表插入、水平线等按钮全部显示。

第 26～33 行设定<h4>标题按钮相关属性，按钮可见，样式类名为 h4，如果当前是 IE 浏览器则命令名称为格式化块，如果当前浏览器是 Safari 浏览器则为标题，如果当前浏览器是 IE 浏览器则参数内容为<h4>，如果当前浏览器是 Safari 浏览器则为 h4，标记['h4']，工具提示'Header 4'。第 34～41 行设定<h5>标题按钮相关属性。

第 42～49 行设定<h6>标题按钮相关属性，相关内容和<h4>的设定类似。第 51～56 行设定剪切、复制、粘贴、HTML 源代码、字体放大、字体缩小按钮全部显示。第 57～63 行设定 exam_html 按钮的功能，向文本部分添加 HTML 代码<abbr title="exam">Jam</abbr>，按钮可见。第 65～72 行添加编辑器文本部分单击事件，如果选中复选框则单击文本的时候跳出信息提示对话框。

效果如图 11.5 和图 11.6 所示。

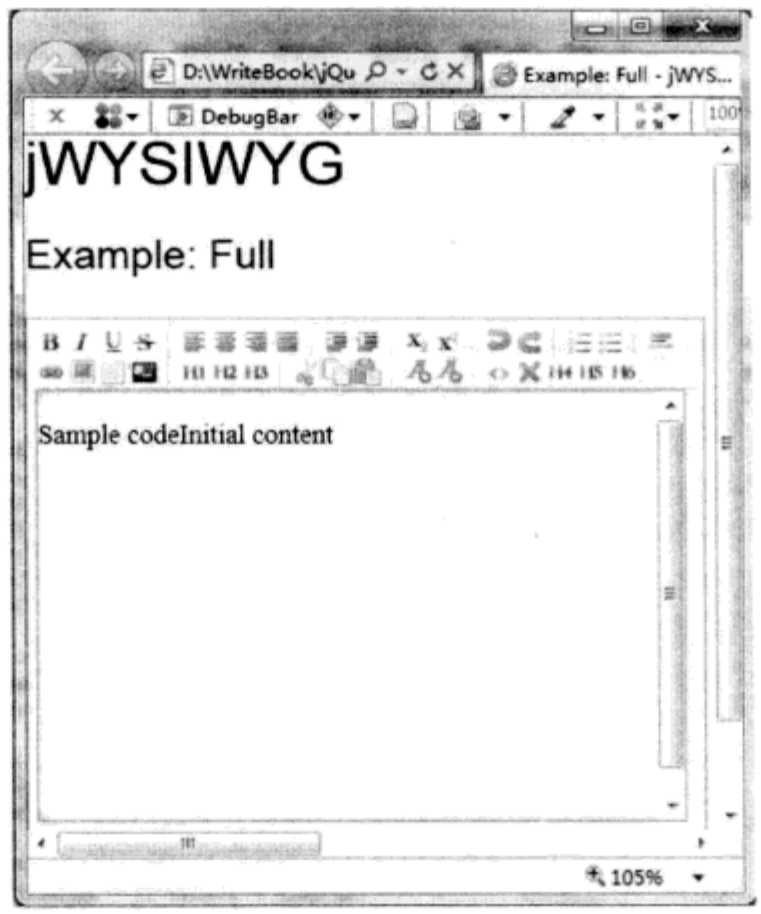


图 11.5　全功能编辑器

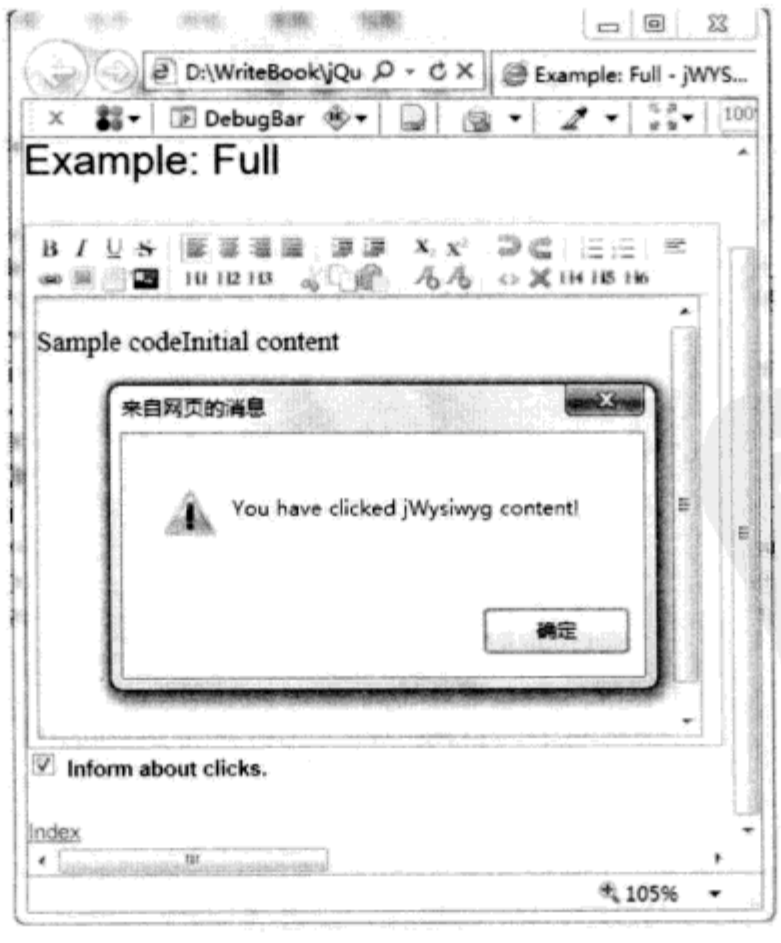


图 11.6　单击文本跳出提示对话框

11.2.4　可调整大小的编辑器

这个示例利用了 jQuery UI 库的调整元素大小的库文件，并对 jwysiwyg 插件的相关属性进行了调整，设定了插件的可调整大小属性，但是要在 HTML 文件头部多引入 jQuery UI 库文件：

```
<script type="text/JavaScript" src="../lib/ui/jquery.ui.core.js"></script>
<script type="text/JavaScript" src="../lib/ui/jquery.ui.widget.js"></script>
<script type="text/JavaScript" src="../lib/ui/jquery.ui.mouse.js"></script>
<script type="text/JavaScript" src="../lib/ui/jquery.ui.resizable.js"></script>
```

JavaScript 代码如下：

```
1  <script type="text/JavaScript">
2    (function ($) {
3        $(document).ready(function () {
4            $('#wysiwyg').wysiwyg({
5                resizeOptions: {},      //调整插件大小选项
6                controls: {
7                    html: { visible: true }
8                }
9            });
10       });
11   })(jQuery);
12   </script>
```

上述代码第 5 行设定编辑器可调整大小，在编辑器的右下角能看到调整大小的箭头。效果如图 11.7 和图 11.8 所示。

图 11.7　编辑器原始尺寸

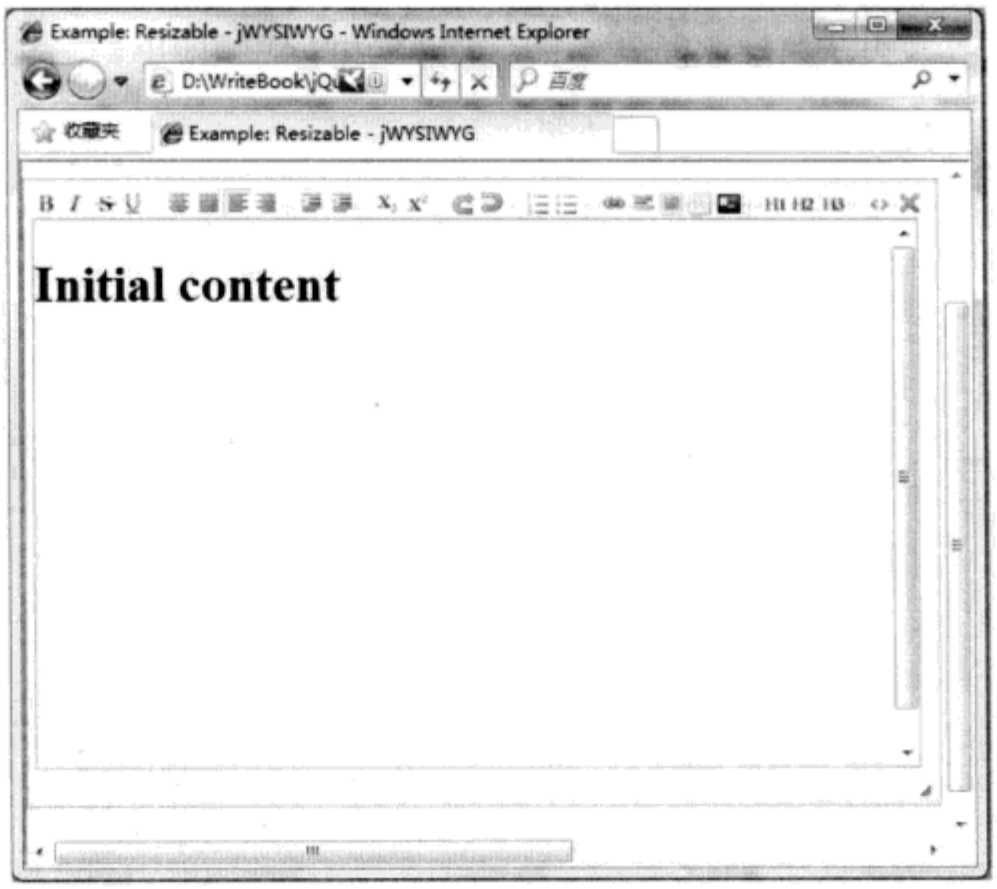


图 11.8　调整大小后效果

11.2.5　编辑器内容的左右移动

这个示例在编辑器的控制按钮集合中加入了左右移动按钮，可以使编辑器内容从左边框移至右边框，也可反方向移动。实际上这里使用了插件中内容的左右对齐属性。JavaScript 代码如下：

```
1  <script type="text/JavaScript">
2    (function ($) {
3        $(document).ready(function () {
4            $('#wysiwyg').wysiwyg({
5                controls: {
6                    rtl: { visible: true },     //左对齐功能可见
7                    ltr: { visible: true }      //右对齐功能可见
8                }
9            });
10       });
11   })(jQuery);
12 </script>
```

上述代码第 6、7 两行设定了 Right to Left 和 Left to Right 按钮为显示状态。效果如图 11.9 和图 11.10 所示。

11.2.6　根据内容自动调整大小的编辑器

这个示例使用了编辑器的自动增长属性，使编辑器可以根据内容自动增长。这里使用

了前面介绍的自动增长属性及最大高度属性。JavaScript 代码如下：

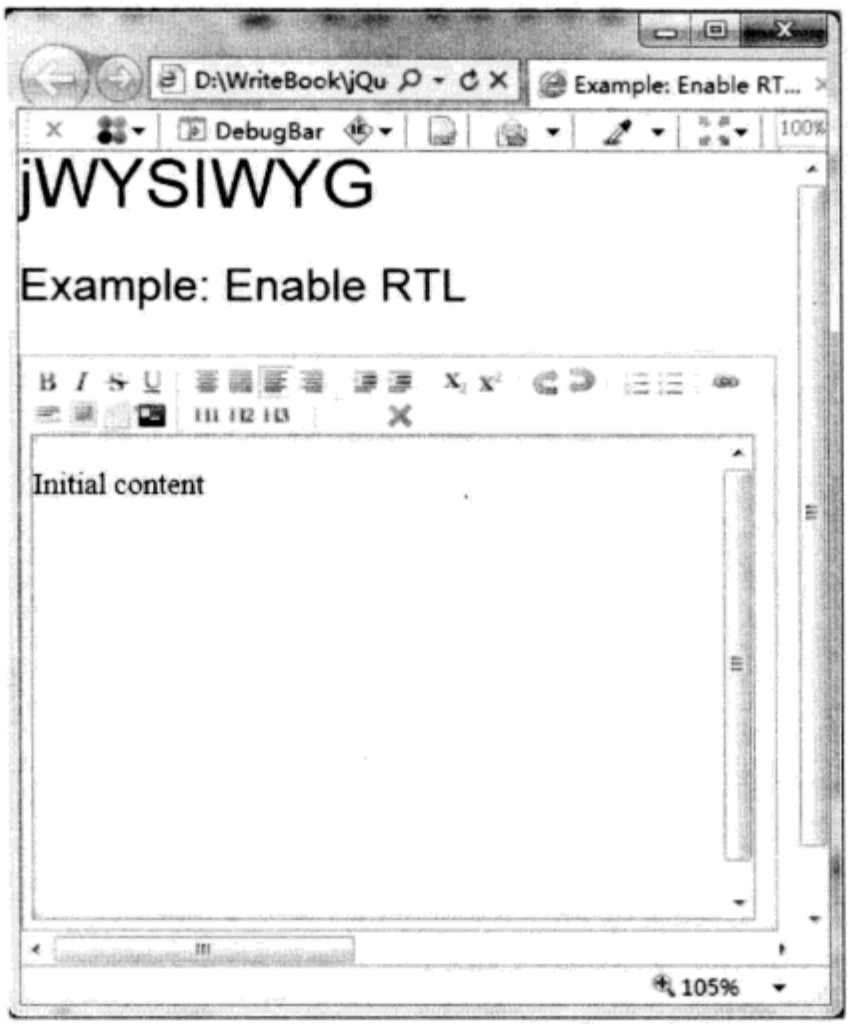


图 11.9　编辑器内容左右移动按钮效果一

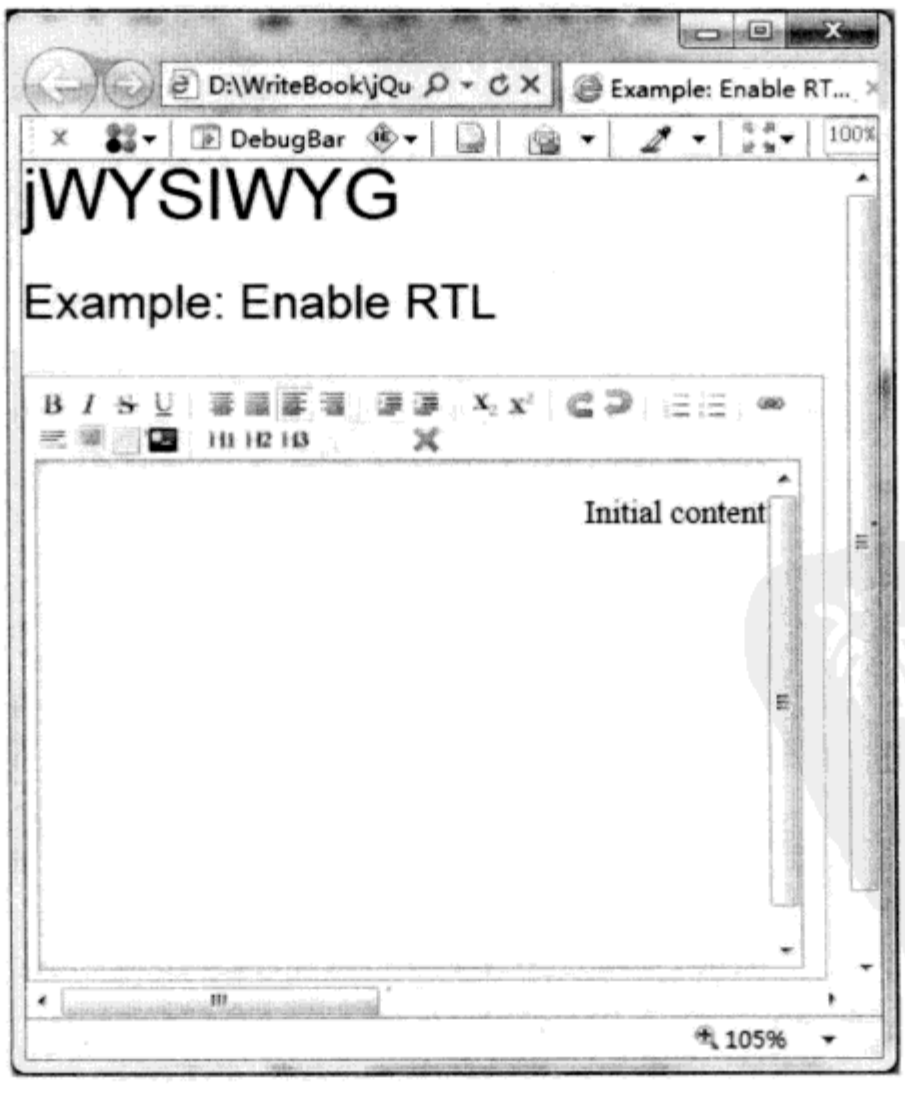


图 11.10　编辑器内容左右移动按钮效果二

```
<script type="text/JavaScript">
  (function ($) {
      $(document).ready(function () {
          $('#wysiwyg').wysiwyg({
              autoGrow: true, //插件可自动增长
              maxHeight: 600  //插件最大高度为 600 像素
          });
      });
  })(jQuery);
</script>
```

上述代码第 5 行设定编辑器可自动增长，第 6 行限制了自动增长的最大高度为 600。效果如图 11.11 所示。

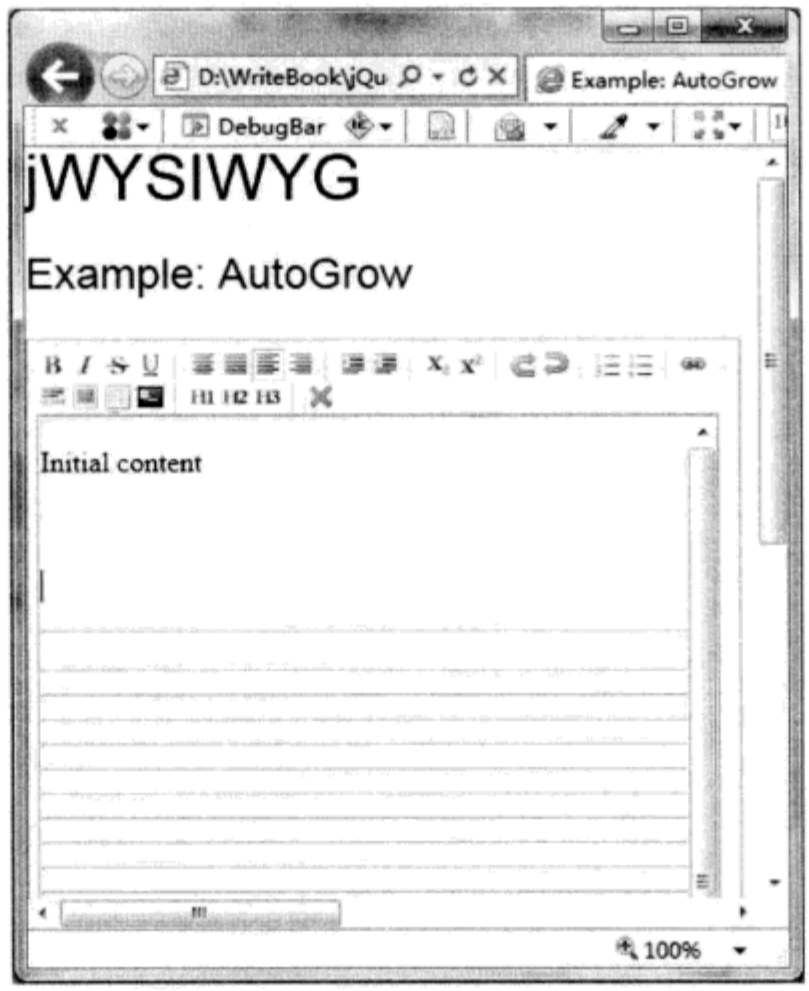


图 11.11　自动增长大小的编辑器

11.2.7　自定义功能控制按钮

这个示例使用了可以自定义功能的控制按钮，这个按钮的功能是在编辑器内容中添加引号。这里添加了一个自定义的内缩进标记按钮，并使用了 blockquote 标记包装我们选择的文本。JavaScript 代码如下：

```
<script type="text/JavaScript">
  (function($) {
      $(document).ready(function() {
          $("textarea").wysiwyg({
              rmUnusedControls: true,                    //隐藏不使用的按钮
              controls: {
```

```
7                bold: { visible : true },              //粗体功能
8                html: { visible : true },              //HTML 源代码查看功能
9                insertOrderedList: { visible : true }, //插入有序列表功能
10               removeFormat: { visible : true }       //移除原有格式功能
11           }
12       });
13       $("input[name=add]").click(function () {
14           $("textarea").wysiwyg("addControl", "quote", {
15               groupIndex: 2,
16               icon: './../tests/images/quote02.gif',
17               tooltip: 'Quote',
18               tags: ['blockquote'],
19               exec: function () {
20                   var range   = this.getInternalRange(),
21                       common  = range.commonAncestorContainer,
22                       blockquote = this.dom.getElement("blockquote");
                             //如果选中的是文本内容，则使用标记对其进行包装
23                   if (common.nodeType === 3) {
24                       common = common.parentNode;
25                   }
26                   if (blockquote && $(blockquote).hasClass("quote")) {
27                       $(common).unwrap();
28               }
29                   else {
30                       if ("body" !== common.nodeName.toLowerCase()) {
31                           $(common).wrap("<blockquote class='quote' />");
32                       }
33                   }
34               },
35               callback: function (event, Wysiwyg) {
36                   alert("callback triggered!");
37               }
38           });
39       });
40       $("input[name=makeBold]").click(function () {
41           $("textarea").wysiwyg("triggerControl", "bold");
42       });
43   });
44 })(jQuery);
45 </script>
```

上述代码第 5 行设定不使用的按钮不显示。第 7～10 行设定使用粗体、查看 HTML 源代码、插入有序列表、移除格式等功能按钮。第 13 行设定当点击添加新控制的时候会在编辑器的工具栏上添加一个实现<blockquote>标记的按钮，选中内容并单击这个按钮，会产生右缩进效果。效果如图 11.12 和图 11.13 所示。

图 11.12　添加右缩进功能按钮

图 11.13　使用右缩进功能按钮

11.3　HtmlBox 插件

HtmlBox 插件是一个跨浏览器、交互式、开源的、所见即所得的 jQuery 编辑器。这个插件安装简单直观，配置方便，适用于 CMS、论坛、用户留言、联系信息等应用。它的英文网址为 http://remiya.com/htmlbox/。

HtmlBox 插件的特点如下。

（1）易整合。

（2）多浏览器支持。

（3）占用空间小。

（4）XHTML 输出。

（5）支持 Ajax。

（6）更换外观方便。

（7）自定义图标。

下面介绍这个插件的基本使用方法。

11.3.1　如何安装

插件的安装实际就是插件文件的引入。在引入这个插件之前同样需要先引入 jQuery 库文件。这个插件使用的 jQuery 库文件版本较低。

（1）引入 JS 文件：

```
<script language="JavaScript" src="jquery-1.3.2.min.js" type="text/
JavaScript"></script>
<script language="JavaScript" src="htmlbox.colors.js" type="text/
JavaScript"></script>                    //插件颜色操作文件
<script language="JavaScript" src="htmlbox.styles.js" type="text/
JavaScript"></script>                    //插件样式操作文件
<script language="JavaScript" src="htmlbox.syntax.js" type="text/
JavaScript"></script>                    //插件语法检查文件
<script language="JavaScript" src="xhtml.js" type="text/JavaScript">
</script>                                //XHtml 标准文件
<script language="JavaScript" src="htmlbox.min.js" type="text/
JavaScript"></script>                    //插件核心文件
```

（2）创建一个文本区域：

```
<textarea id='ha'></textarea>
```

（3）设置运行插件的属性特点。属性的使用方法参考下面介绍的示例。

11.3.2　属性特点

在运行插件时需要对插件进行初始化，即配置运行属性，如表 11.6 所示。

表 11.6　HtmlBox属性说明

属　性	类　型	默 认 值	说　明
idir	字符串	./image/	需要使用图片的路径位置，可以是绝对路径，也可以是相对 URL 地址
about	布尔	true	是否使用 about 说明对话框

续表

属　性	类　型	默 认 值	说　明
icons	字符串	default	设置需要使用的图标
skin	字符串	default	设置皮肤样式
toolbars	数组	["bold","italic","underline"]	设置工具栏内容
output	字符串	xhtml	设置是哪种输出内容标准
css	字符串	body{margin:3px;font-family:vendana;font-size:11;}p{margin:0px;}	设置使用的 CSS 样式
toolbar_height	整型	16	工具栏的高
tool_height	整型	16	工具的高
tool_width	整型	16	工具的宽
tool_image_height	整型	16	工具栏上图片的高
tool_image_width	整型	16	工具栏上图片的宽
success	函数	function(data){alert(data);}	一个成功的 Ajax 调用返回后调用的函数
error	函数	function(a,b,c){return this;}	当 Ajax 调用出错时调用的函数

11.3.3　内置函数

HtmlBox 通过内置函数实现特定功能，如表 11.7 所示。

表 11.7　HtmlBox内置函数

函　数	说　明
get_html()	返回编辑器中内容的 HTML 源代码，如果 xhtml.js 被引入，则返回 XHTML 源代码
get_text()	返回在 HTML 标记中的文本内容
set_text()	设置编辑器中的文本
get(url,data)	使用 get 方式利用 Ajax 向服务器提交编辑器中的数据，如果 data 为空，则使用 get_text()获取数据
post(url,data)	使用 post 方式利用 Ajax 向服务器提交编辑器中的数据，如果 data 为空，则使用 get_text()获取数据
success(fn)	如果上两个方法成功返回，则使用函数处理，默认警告浏览返回
error(fn)	如果 Ajax 返回失败，则使用函数处理
cmd(cmd,arg)	执行一个指定的命令
change(fn)	文本改变时使用对应函数处理
remove()	移除编辑器实例
wap_tags(start,end)	在选中的文本外包裹标记

11.3.4　编辑器可使用的工具

HtmlBox 可支持的对文本进行加工的工具如表 11.8 所示。

表 11.8　HtmlBox工具说明

工　具	说　明	工　具	说　明
cut	从编辑器中剪切选中文本到剪切板中	justify	选中文本分散对齐
copy	从编辑器中复制选中文本到剪切板中	ol	选中行添加有序列表
paste	粘贴剪切板中的内容到编辑器的光标处	ul	选中行添加无序列表
bold	将选中的文本改成粗体	indent	选中文本右缩进
italic	将选中的文本改成斜体	outdent	选中文本突出
underline	在选中的文本下添加下划线	hyperlink	在选中文本上创建超链接
strike	在选中的文本上添加删除线	image	插入图片
sup	将选中的文本改成上标形式	code	显示 HTML 代码
sub	将选中的文本改成下标形式	fontsize	改变选中文本的字号大小
left	左对齐选中文本	fontfamily	改变选中文本的字体
right	右对齐选中文本	fontcolor	改变选中文本的字体颜色

11.3.5　插件应用举例

（1）全功能编辑器。这个示例使用了编辑器的所有工具功能。这里使用了插件的工具条属性，并在该属性中设定了常用工具按钮，同时设定了插件的外观颜色为蓝色。JavaScript 代码如下：

```
<script language="JavaScript" type="text/JavaScript">
$("#ha").css("height","100%").css("width","100%").htmlbox({
  toolbars:[
     [
     //剪切、复制、粘贴功能
     "separator","cut","copy","paste",
     //撤销、重做功能
     "separator","undo","redo",
     //粗体字、斜体字、下划线、删除线、上标、下标功能
     "separator","bold","italic","underline","strike","sup","sub",
     //左对齐、右对齐、居中对齐、两端对齐功能
     "separator","justify","left","center","right",
     //有序列表、无序列表、缩进、突出功能
     "separator","ol","ul","indent","outdent",
     //超链接、移除超链接、图片功能
     "separator","link","unlink","image"

     ],
     [//显示代码
     "separator","code",
      //格式化、字号大小、字体集、字体颜色、高亮功能
      "separator","formats","fontsize","fontfamily",
     "separator","fontcolor","highlight",
     ],
     [
     //移除格式、去除标记、分隔线、段落功能
     "separator","removeformat","striptags","hr","paragraph",
```

```
28        //引号、样式、语法功能
29        "separator","quote","styles","syntax"
30        ]
31   ],
32   skin:"blue"      //皮肤颜色为蓝色
33 });
34 </script>
```

上述代码第 2 行设定编辑器的大小，并调用 HtmlBox 初始化函数。第 3 行使用 toolbars 添加工具集。第 4～31 行将前面介绍的工具添加到工具栏上。其中有些工具在表 11.8 中没有介绍，separator 表示不同工具之间的分隔线，unlink 表示取消超链接，removeformat 表示删除原有格式，striptags 表示去除标记，hr 表示添加分隔线，paragraph 表示添加段落，quote 表示突出选中内容，style 表示 CSS 样式设定，syntax 表示编程语言语法。第 32 行表示插件使用蓝色为主色调。效果如图 11.14 所示。

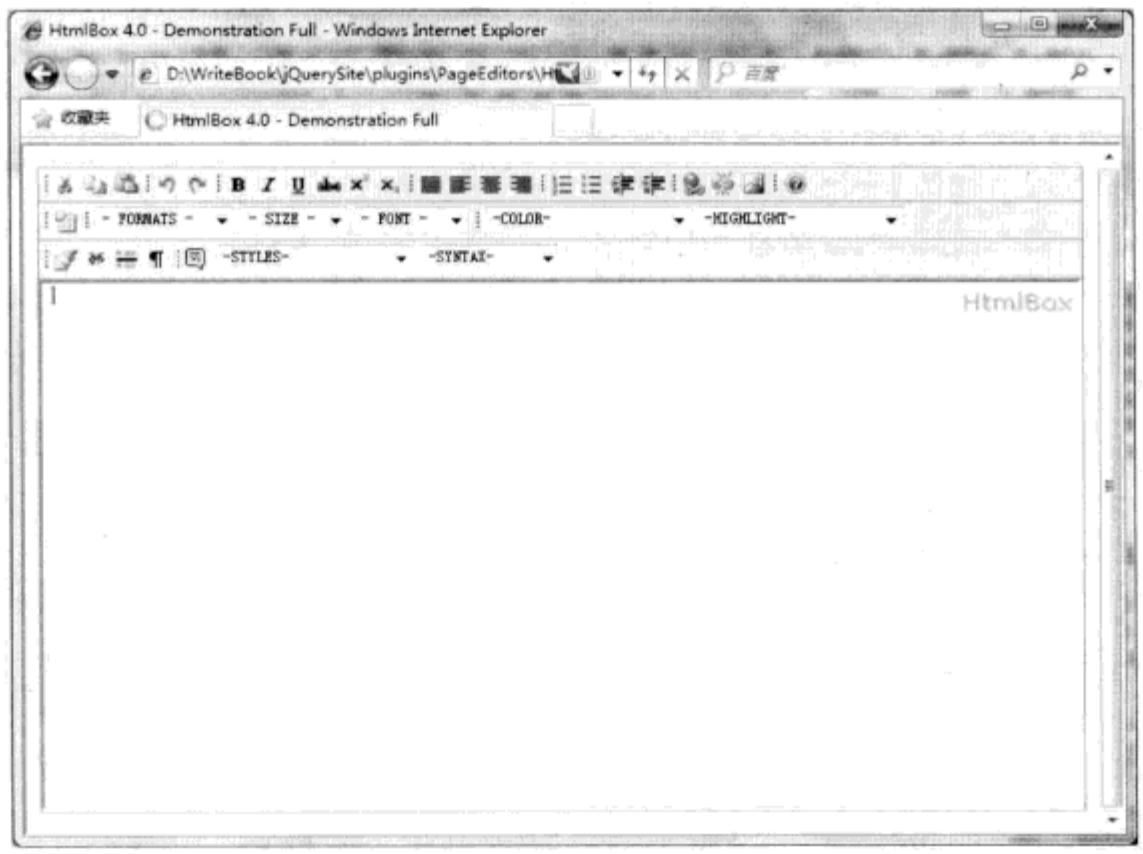


图 11.14　全功能 HtmlBox 编辑器

（2）添加新功能编辑器。这个示例添加了新的图标和图标对应的功能。这个插件同样可以添加自定义的工具按钮并设定它们的功能。新功能的定义在插件初始化的工具栏设定中完成。JavaScript 代码如下：

```
1 <script language="JavaScript" type="text/JavaScript">
2  var hb_icon_set_default = $("#htmlbox_icon_set_default").css("height",
  "100").css("width","600").htmlbox({
3    toolbars:[
4   ["cut","copy","paste","separator_dots","bold","italic","underline",
    "strike","separator","sub","sup","separator_dots","undo","redo",
    "separator_dots","left","center","right","justify","separator_dots",
    "ol","ul","indent","outdent","separator_dots","link","unlink","image"],
5        ["code","removeformat","striptags","separator_dots","quote",
        "paragraph","hr","separator_dots",
6            {icon:"new.gif",tooltip:"New",command:function(){hb_icon_
```

```
                set_default.set_text("<p></p>");}},    //新增内容功能
7               {icon:"open.gif",tooltip:"Open",command:function(){alert
                ('Open')}},                                    //打开操作功能
8               {icon:"save.gif",tooltip:"Save",command:function(){alert
                ('Save')}}                                     //保存操作功能
9           ]
10  ],
11  icons:"default",
12  skin:"default"
13 });
14 </script>
```

上述代码的第 2 行和上面的例子类似，向工具栏中添加工具按钮。第 6 行是新添加一段。第 7 行添加模拟打开功能。第 8 行添加模拟保存功能。效果如图 11.15～图 11.17 所示。

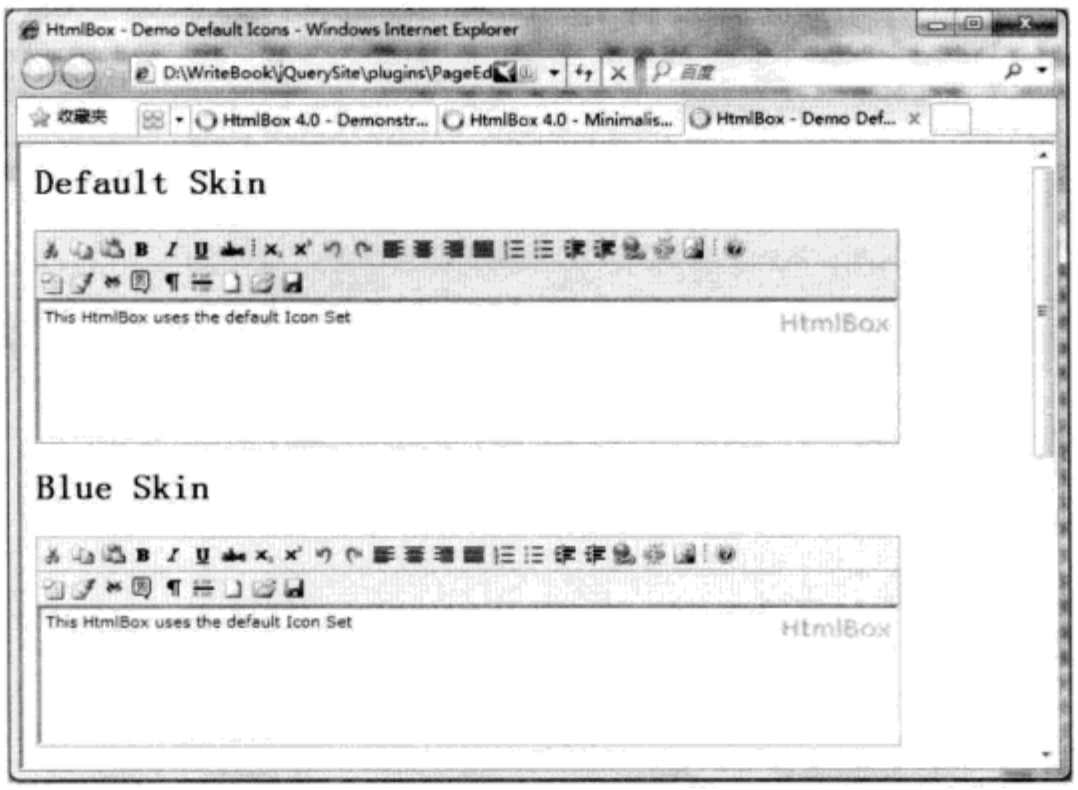


图 11.15　添加新功能的编辑器

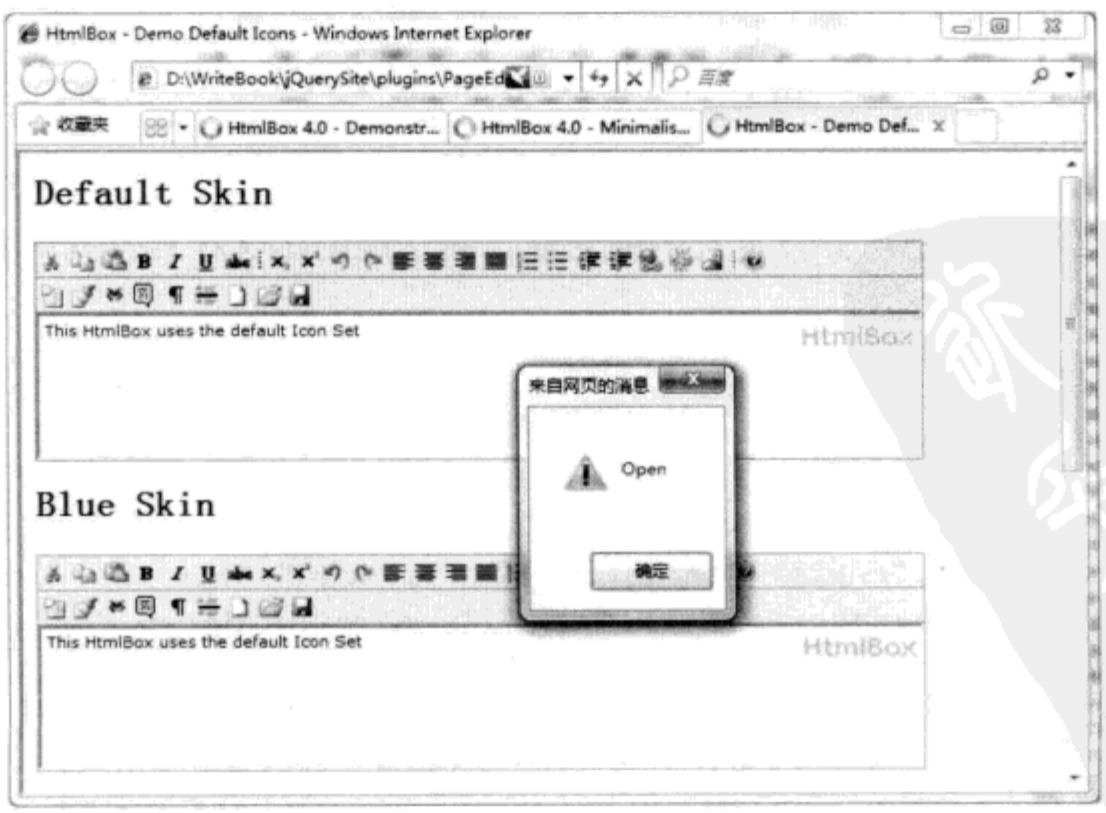


图 11.16　模拟打开功能

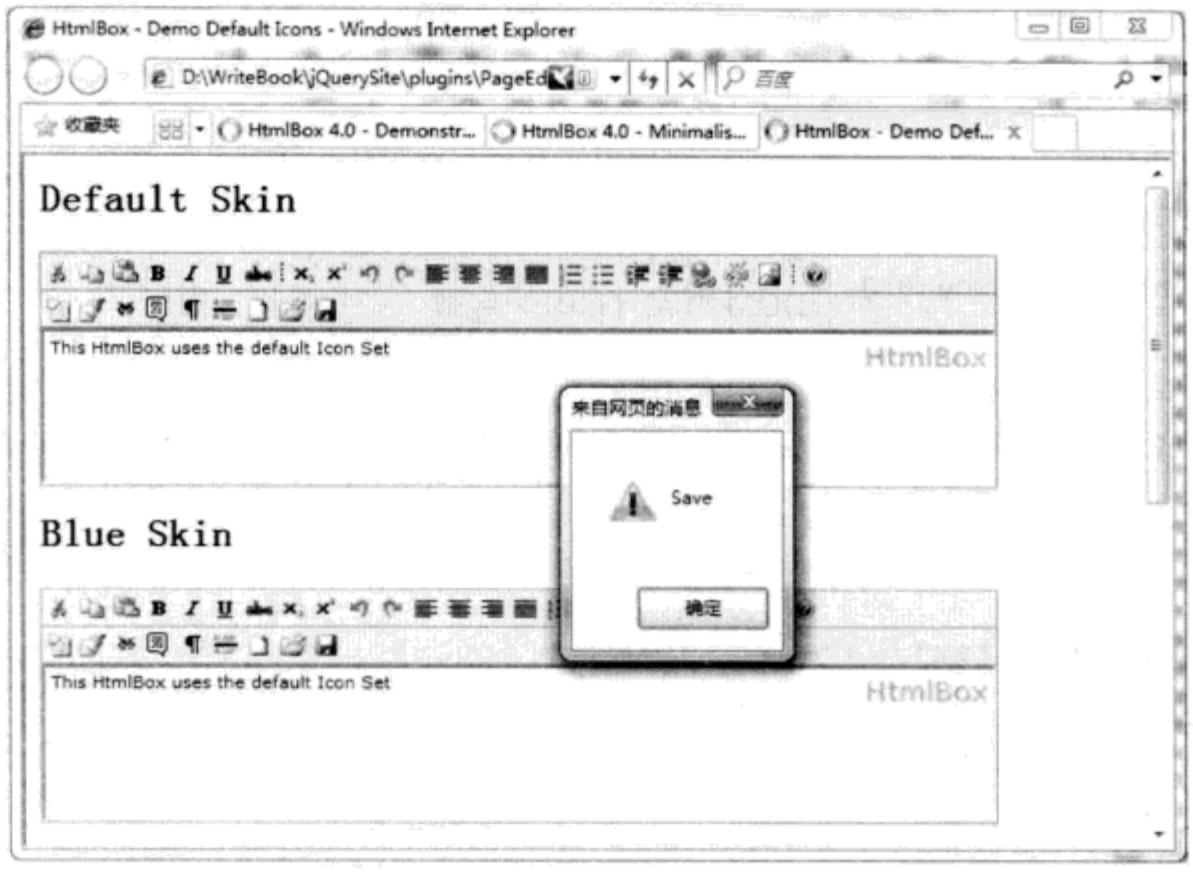


图 11.17　模拟保存功能

11.4　Lightweight RTE 插件

Lightweight RTE 插件也是 jQuery 的富文本编辑器。它也是一个轻量型的插件，易于扩展，为不同的设计模式和资源模式设定不同的工具栏。它的英文介绍网址为 http://code.google.com/p/lwrte/。下面首先介绍一下这个插件的相关函数及说明，如表 11.9 所示。

表 11.9　Lightweight RTE函数说明

函　　数	说　　明
get_content()	返回当前编辑器内容
set_content(content)	设置内容到编辑器
get_select_text()	获取选中文本，不包含标记内容，只工作在设计模式
get_select_html()	获取选中文本，包含标记内容，只工作在设计模式
selection_replace_with(html)	用参数替换选中内容，只工作在设计模式
editor_cmd(command,args)	在设计模式下执行内建命令，携带参数，工作在设计模式
enable_design_mode()	启用设计模式
disable_design_mode(submit)	关闭设计模式，如果参数为 false，则创建新的文本域，从浮动框架复制内容，删除浮动框架，如果参数为 true，则创建新的隐藏域，并提交浮动框架的文本内容
create_toolbar(controls)	创建工具栏，通过控制工具集合
create_panel(title,width)	创建面板，可以指定标题，宽度参数
get_toolbar()	获得当前活动的工具栏
activate_toolbar(editor,tb)	移除编辑器原有工具栏，插入参数 tb 所代表的工具栏
toolbar_click(obj,control)	工具栏上控制按钮的单击事件，执行按钮所包含的执行代码

续表

函　　数	说　　明
get_selection_range()	返回当前选中内容的范围对象，工作在设计模式
get_selected_element()	返回当前鼠标所在 HTML 标记元素，工作在设计模式
set_selected_controls(node, controls)	标记工具栏中的控制按钮只对设计内容中的被选中的 HTML 的 DOM 节点（node）起效，工作在设计模式

下面看一下这个插件的示例，这里使用了插件初始化函数，并设定了插件的基本样式，加载了插件的基本控制按钮和 HTML 标记处理按钮。JavaScript 代码如下：

```
1 <script type="text/JavaScript" src="jquery.js"></script>
2 <script type="text/JavaScript" src="jquery.rte.js"></script>
                                                    //插件核心功能文件
3 <script type="text/JavaScript" src="jquery.rte.tb.js"></script>
                                                    //插件附加功能文件
4 <script type="text/JavaScript" src="jquery.ocupload-1.1.4.js"></script>
5 <script type="text/JavaScript">
6   $(document).ready(function() {
7       var arr = $('.rte1').rte({
8           css: ['default.css'],                   //默认样式设定
9           controls_rte: rte_toolbar,              //使用工具条
10          controls_html: html_toolbar             //使用 HTML 工具条
11      });
12      $('.rte2').rte({
13          css: ['default.css'],
14          width: 450,
15          height: 200,
16          controls_rte: rte_toolbar,
17          controls_html: html_toolbar
18      }, arr);
19  });
20  </script>
```

上述代码第 2、3 行是插件的库文件。第 7、12 行是插件的初始代码。第 8、13 行是插件的样式设定默认样式。第 14、15 行设定了第二个编辑器的初始大小。第 9、16 行设定了编辑器的工具栏。第 10、17 行设定了编辑器设计模式下的工具栏。效果如图 11.18 所示。

图 11.18　Lightweight RTE 插件效果

11.5　小　　结

文本编辑器是一个良好的用户体验形式。本章主要介绍了文本编辑器插件的使用方法。重点内容是插件的功能与外观定制，同时这部分也是本章的难点。下一章将介绍 jQuery 多媒体插件的使用。

11.6　习　　题

【习题 1】练习 markItUp 插件的使用方法。
【习题 2】练习 jwysiwyg 插件的使用方法。
【习题 3】练习 Lightweight RTE 插件的使用方法。

第12章 多 媒 体

在现在的网页中除了文字与静态图片外，还有各种媒体播放形式，如 MP3、Flash、Windows Media Player、Real Player、Silverlight 等。jQuery 也有相应的插件播放这些多媒体文件。多媒体插件除了播放各种媒体文件功能外，还加入了 jQuery 所特有的一些动画特效，更吸引用户。本章将介绍 3 个 jQuery 多媒体插件。

12.1 jQuery.Flash 插件

jQuery.Flash 是基于 jQuery 插件的 JavaScript 代码。它的英文网址是 http://jquery.lukelutman.com/plugins/flash/index.html。该插件主要是为了解决 Flash 在 Web 页面的嵌入式解决方案，与 AC_RunActiveContent、SWFOjbect 功能类似，但是该插件有更多的优点。

（1）完全 W3C 标准。

（2）稳定的交互性。

（3）使用简单，易于扩展。

（4）轻量型，功能强大。

下面通过 4 个示例来介绍这个插件的使用方法。

12.1.1 基本 Flash 文件嵌入

这个示例使用了插件的 src、width、height 属性，分别表示 Flash 文件位置及播放窗口的宽和高。其中，HTML 代码和 CSS 样式设定请参考光盘内容。具体 JavaScript 功能代码如下：

```
1 <script type="text/javascript" src="../../jslib/jquery-1.6.js"></script>
2 <script type="text/javascript" src="JS/jquery.flash.js"></script>
                                                          //添加插件文件
3 <script type="text/javascript">
4   $(document).ready(function(){
5       $('#example').flash({                             //插件初始化
6           src: 'MediaFile/example.swf',                 //指定播放的多媒体文件
7           width: 360,                                   //插件的显示宽度
8           height: 215                                   //插件的显示高度
9       },
10      {
11          version: 8                                    //播放文件的版本
12      });
```

```
13  });
14 </script>
```

上述代码第 5 行是插件的初始化函数，第 11 行表示 Flash 版本。效果如图 12.1 所示。

图 12.1　基本 Flash 文件嵌入

12.1.2　Flash 替换文本内容

这个示例中使用 HTML 页面上的文本作为参数，传递 Flash 文件，并在原文本位置播放 Flash。其中，使用到了插件的样式设定参数 src、flashvars 属性、Flash 运行参数。HTML 代码和 CSS 样式设定请参考光盘内容。具体 JavaScript 功能代码如下：

```
1 <script type="text/javascript">
2   $(document).ready(function(){
3       $('h3').flash(
4           {
5               src: 'MediaFile/itc_century.swf',
6               flashvars: {
7                   css: [
8                       '* { color: #666666; }',
9                       'a { color: #0055CC; text-decoration: none; }',
10                      'a:hover { text-decoration: underline; }'
11                  ].join(' ')
12              }
13          },
14          { version: 7 },
15          function(htmlOptions) {                    //添加文本显示功能
16              htmlOptions.flashvars.txt = this.innerHTML;
17              this.innerHTML = '<span>'+this.innerHTML+'</span>';
18              var $alt = $(this.firstChild);
19              htmlOptions.height = $alt.height();
20              htmlOptions.width = $alt.width();
21              $alt.addClass('alt');
```

```
22                $(this)
23                    .addClass('flash-replaced')
24                    .prepend($.fn.flash.transform(htmlOptions));
25            }
26        );
27    });
28 </script>
```

上述代码第 6 行设定插件的 Flash 参数。第 7～11 行设定文本显示样式。第 14 行设定 Flash 版本。第 15～25 行添加 HTML 文本传入并设定显示样式。其中第 16 行将 h3 内嵌套的内容作为 Flash 参数的文本内容传入插件的 HTML 选项参数。第 17 行向 h3 内嵌套标记。第 18 行获取标记对象，第 19、20 行向 Flash 传入文本所占的高和宽。第 21 行向标记添加 CSS 样式。第 23 行向 h3 添加 CSS 样式。第 24 行向 Flash 传送参数对象。效果如图 12.2 所示。

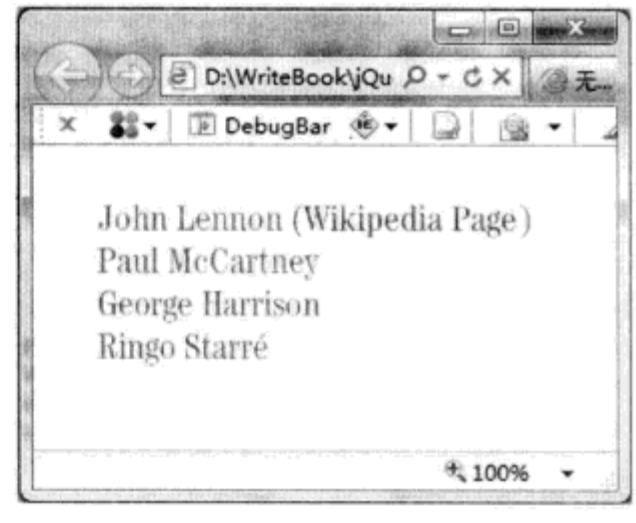


图 12.2　Flash 替换文本内容

12.1.3　MP3 播放示例

这个插件除了可以播放 Flash 文件外，还可以播放 MP3 音频文件。在本示例指定了向插件传入的参数是 MP3 音频文件类型，并指定了文件位置。HTML 代码和 CSS 样式设定请参考光盘内容。具体 JavaScript 功能代码如下：

```
1 <script type="text/javascript" src="../../jslib/jquery.js"></script>
2 <script type="text/javascript" src="JS/jquery.flash.js"></script>
3 <script type="text/javascript">
4   $(document).ready(function(){
5       $('a[@href$="mp3"]').flash(
6           { src: 'MediaFile/singlemp3player.swf', height: 20, width: 100 },
7           { version: 7 },
8           function(htmlOptions) {                    //添加超链接功能
9               $this = $(this);
10              htmlOptions.flashvars.file = $this.attr('href');
11              $this.before($.fn.flash.transform(htmlOptions));
12          }
13      );
14  });
15 </script>
```

上述代码第 5 行指定超链接标记中的资源类型为 MP3 音频文件，并调用插件初始化函数。第 6 行指定资源位置，以及播放的高和宽。第 7 行指定 Flash 版本。第 10 行指定向 Flash 传入参数的文件类型参数为超链接的地址属性值。第 11 行将 HTML 参数属性传递给插件。效果如图 12.3 所示。

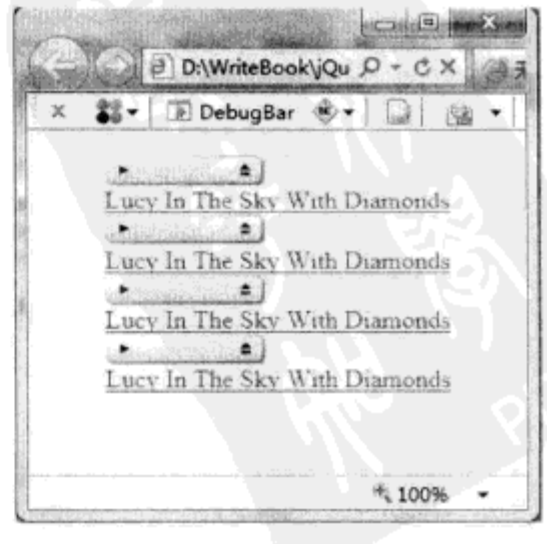


图 12.3　MP3 播放示例效果

12.1.4　使用内联样式设定插件播放

这个示例在 HTML 标记的 rel 属性内设置插件播放参数，并在插件初始化过程中设定对应参数。HTML 代码和 CSS 样式设定请参考光盘内容。具体 JavaScript 功能代码如下：

```
1 <script type="text/javascript">
2   $(document).ready(function(){
3       $('li').flash(null, { version: 8 }, function(htmlOptions) {
    //通过自定义方法代替静态参数设定播放
4           var $this = $(this);
5           var params = $this.attr('rel').split(':');
6           htmlOptions.src = params[0];                    //播放文件位置
7           htmlOptions.width = params[1];                  //显示宽度
8           htmlOptions.height = params[2];                 //显示高度
9           this.innerHTML = '<div class="alt">'+this.innerHTML+'</div>';
                                                            //内联文本
10          $this.addClass('flash-replaced').prepend($.fn.flash.transform
            (htmlOptions));                                 //添加样式和播放参数
11      });
12  });
13 </script>
```

上述代码第 3 行指定在列表项上加载插件并初始化，对于插件样式并没有设置，直接指定 Flash 版本。第 4 行获取列表项对象。第 5 行将列表项中 rel 属性中的值分隔。第 6 行设定播放 Flash 文件位置。第 7、8 行设定插件的宽和高。第 10 行向插件传递参数。效果如图 12.4 和图 12.5 所示。

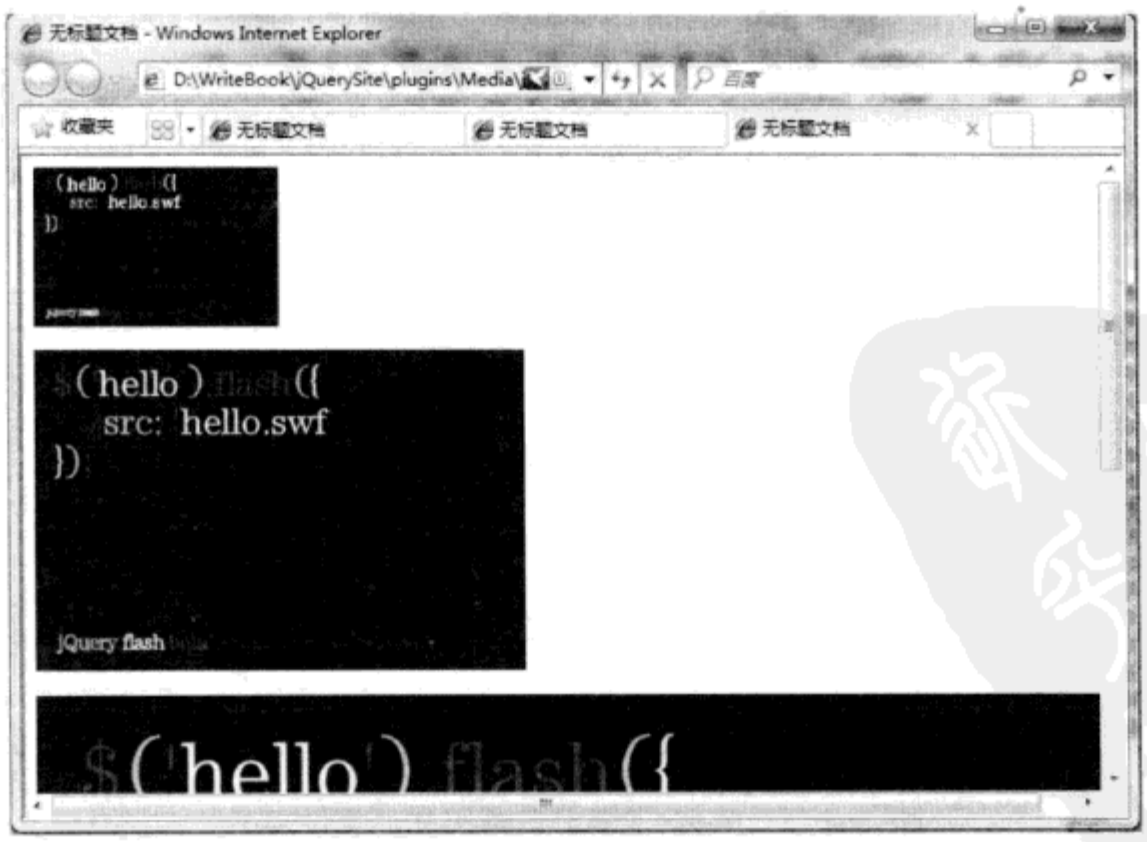


图 12.4　内联样式播放一

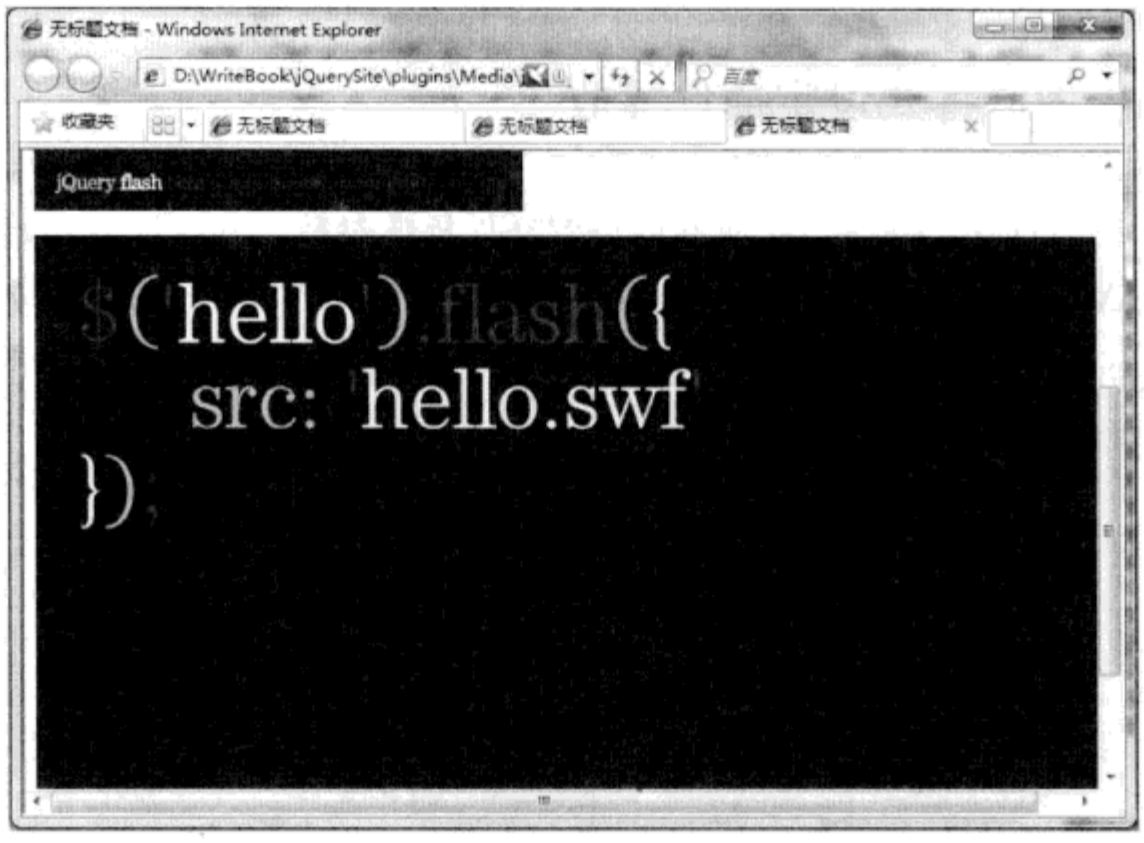


图 12.5　内联样式播放二

12.2　jPlayer 插件

jPlayer 插件和前面讲的 jQuery.Flash 插件类似，也是一个媒体播放插件。因为这个插件的最新版是基于 HTML 5 标准的，因此我们将 IE 8 升级为 IE 9 来查看插件效果。这个插件的英文网址为 http://www.jplayer.org/。下面看一下它的特点。

（1）易于使用和发布。

（2）方便通过 HTML 和 CSS 定制插件样式。

（3）轻量型插件。

（4）免费，开源。

（5）多种解码，跨平台。

12.2.1　jPlayer 插件基本介绍

在前面介绍其他插件时，很多插件在初始化时都有相应属性选项设定。同样，这个插件也有相应的属性选项定义，如表 12.1 所示。

表 12.1　jPlayer插件属性说明

属　性	类型	默 认 值	说　明
ready	函数		指定加载的资源文件，以备插件播放
swfPath	字符串	js	指定插件需要播放的 Flash 文件的提取路径，可以根据需要更改为其他路径
solution	字符串	"html, flash"	定义解决方式的主次顺序，可以按照要求颠倒顺序
supplied	字符串	"mp3"	提供给插件的文件格式

续表

属　性	类型	默 认 值	说　明
preload	字符串	"metadata"	预加载，可以为"none", "metadata" 和 "auto"
volume	数字	0.8	定义初始化播放音量，从 0 至 1
muted	布尔	false	定义初始化静音状态
backgroundColor	字符串	"#000000"	背景色
cssSelectorAncestor	字符串	"#jp_interface_1"	定义所有 CSS 样式选择器的上层选择器，通常使用 ID 选择器
cssSelector	对象	{cssSelectors}	插件使用的 CSS 选择器对象，有默认的一组 CSS 选择器
idPrefix	字符串	"jp"	在插件内使用的 HTML 元素 ID 的前缀
errorAlerts	布尔	false	是否使用错误报告
warningAlerts	布尔	false	是否使用警告报告
eventType	函数	undefined	负责绑定处理函数的事件类型，参考表 12.2

表 12.2　jPlayer插件事件说明

事　件	类　型	说　明
event	对象	标准插件事件属性
event.jPlayer	对象	插件的信息对象
event.jPlayer.error	对象	插件的错误事件对象
event.jPlayer.error.type	字符串	插件的错误事件对象类型
event.jPlayer.error.context	字符串	发生错误的原因
event.jPlayer.error.message	字符串	描述错误的消息
event.jPlayer.error.hint	字符串	修正错误的建议
event.jPlayer.flash	对象	关于 Flash 解决方式的信息
event.jPlayer.html	对象	关于 HTML 解决方式的信息
event.jPlayer.status	对象	插件状态
event.jPlayer.status.src	字符串	在插件中使用的 URL 地址
event.jPlayer.status.file	对象	设置媒体内容时，代表媒体对象文件
event.jPlayer.status.seekPercent	数值	可查找的百分比
event.jPlayer.status.currentPercentRelative	数值	当前查找时间的百分比
event.jPlayer.status.currentPercentAbsolute	数值	当前持续时间百分比
event.jPlayer.status.currentTime	数值	以秒为单位表示当前时间
event.jPlayer.status.duration	数值	多媒体文件的总播放时间
event.jPlayer.status.volume	数值	插件音量
event.jPlayer.status.muted	布尔	插件是否静音
event.jPlayer.status.srcSet	布尔	多媒体资源是否被设置
event.jPlayer.status.paused	布尔	是否暂停
event.jPlayer.status.waitForPlay	布尔	是否等待播放
event.jPlayer.status.waitForLoad	布尔	是否等待加载
event.jPlayer.status.video	布尔	是否是视频
event.jPlayer.status.width	字符串	插件样式的宽
event.jPlayer.status.height	字符串	插件样式的高
event.jPlayer.version	对象	插件版本

续表

事　　件	类　　型	说　　明
event.jPlayer.version.script	字符串	插件使用的 JavaScript 脚本语言版本
event.jPlayer.version.flash	字符串	插件使用的 Flash 版本
event.jPlayer.version.needFlash	字符串	和 JavaScript 兼容的 Flash 版本
event.jPlayer.warning	对象	插件的警告信息对象
event.jPlayer.warning.type	字符串	插件的警告类型
event.jPlayer.warning.context	字符串	警告原因
event.jPlayer.warning.message	字符串	描述警告的消息
event.jPlayer.warning.hint	字符串	避免警告的建议

插件内置了相关方法供我们调用来操作播放过程，如表 12.3 所示。

表 12.3　jPlayer插件方法说明

方 法 形 式	说　　明
$(id).jPlayer("setMedia", Object: media)	定义播放的多媒体文件，这个方法需要放在其他方法之前，包含的多媒体类型有：mp3、m4a、m4v、oga、ogv、webma、webmv、wav、poster
$(id).jPlayer("clearMedia")	清除多媒体，停止播放
$(id).jPlayer("load")	在播放前预加载多媒体文件
$(id).jPlayer("play", [Number: time])	播放多媒体，参数 time 表示从指定位置播放，时间单位为秒
$(id).jPlayer("pause", [Number: time])	暂停多媒体播放，参数 time 表示在指定位置暂停，时间单位为秒
$(id).jPlayer("pauseOthers")	暂停其他的多媒体播放
$(id).jPlayer("stop")	停止多媒体播放
$(id).jPlayer("playHead", Number: percentOfSeekable)	移动播放位置到进度条的指定位置
$(id).jPlayer("volume", Number: ratio)	媒体播放音量
$(id).jPlayer("mute")	静音操作
$(id).jPlayer("unmute")	取消静音操作
$(id).jPlayer("option", [String: key, [Variable: value]])	设置插件配置信息
$(id).jPlayer("destroy")	移除插件

下面通过示例来介绍这个插件的使用方法。

12.2.2　jPlayer 插件基本使用方式：播放音频文件

这个示例使用了插件的基本事件和方法，包括一些基本属性的使用。首先，需要创建 HTML 静态页面。

```
1        <div class="jp-audio">
2            <div class="jp-type-single">
```

```
3            <div id="jp_interface_1" class="jp-interface">
4                <ul class="jp-controls">
5                    <li><a href="#" class="jp-play" tabindex="1">play
                     </a></li>
6                    <li><a href="#" class="jp-pause" tabindex="1">pause
                     </a></li>
7                    <li><a href="#" class="jp-stop" tabindex="1">stop
                     </a></li>
8                    <li><a href="#" class="jp-mute" tabindex="1">mute
                     </a></li>
9                    <li><a href="#" class="jp-unmute" tabindex="1">unmute
                     </a></li>
10               </ul>
11               <div class="jp-progress">
12                   <div class="jp-seek-bar">
13                       <div class="jp-play-bar"></div>
14                   </div>
15               </div>
16               <div class="jp-volume-bar">
17                   <div class="jp-volume-bar-value"></div>
18               </div>
19               <div class="jp-current-time"></div>
20               <div class="jp-duration"></div>
21           </div>
22           <div id="jp_playlist_1" class="jp-playlist">
23               <ul>
24                   <li>Bubble</li>
25               </ul>
26           </div>
27       </div>
28   </div>
```

上述代码中使用了插件的内置 CSS 样式选择器，“jp-type-single”表示单一控件样式，“jp-interface”表示插件界面样式，“jp-controls”表示插件的控制面板样式，“jp-play”表示插件播放按钮样式，“jp-pause”表示插件暂停按钮样式，“jp-stop”表示插件停止按钮样式，“jp-mute”表示静音按钮样式，“jp-unmute”表示取消静音按钮样式，“jp-progress”表示插件播放进度底层 DIV 样式。

“jp-seek-bar”表示播放定位栏样式，“jp-play-bar”表示播放栏样式，“jp-volume-bar”表示音量显示样式，“jp-volume-bar-value”表示当前音量值显示样式，“jp-current-time”表示当前播放时间显示样式，“jp-duration”表示完整播放时间显示样式，“jp-playlist”表示播放曲目列表显示样式。

接下来看一下实现播放功能的 JavaScript 代码：

```
1 <link href="skin/jplayer.blue.monday.css" rel="stylesheet" type=
  "text/css" />
2 <script src="MediaFiles/html5.js"></script>
3 <script type="text/javascript" src="../../../jslib/jquery-1.6.min.js">
  </script>
4 <script type="text/javascript" src="js/jquery.jplayer.min.js"></script>
5 <script type="text/javascript">
```

```
//<![CDATA[
6   $(document).ready(function(){
7       $("#jquery_jplayer_1").jPlayer({
8           ready: function () {
9               $(this).jPlayer("setMedia", {          //设置多媒体播放参数
10                  m4a: "MediaFiles/Miaow-07-Bubble.m4a", //m4a 文件
11                  oga: "MediaFiles/Miaow-07-Bubble.ogg"  //oga 文件
12              }).jPlayer("play");                        //调用播放动作
13          },
14          ended: function (event) {                      //结束事件
15              $(this).jPlayer("play");
16          },
17          swfPath: "js",
18          supplied: "m4a, oga"
19      });
20  });
    //]]>
21 </script>
```

上述代码第 7 行调用插件初始化函数。第 8 行使用了插件的准备事件。第 9～12 行设定了向插件中添加多媒体文件，并调用插件的播放方法。第 14、15 行使用了插件的结束事件，同样调用了插件的播放方法。第 17 行指定插件需要使用的 Flash 文件路径。第 18 行指定了插件使用的多媒体文件类型。效果如图 12.6 所示。

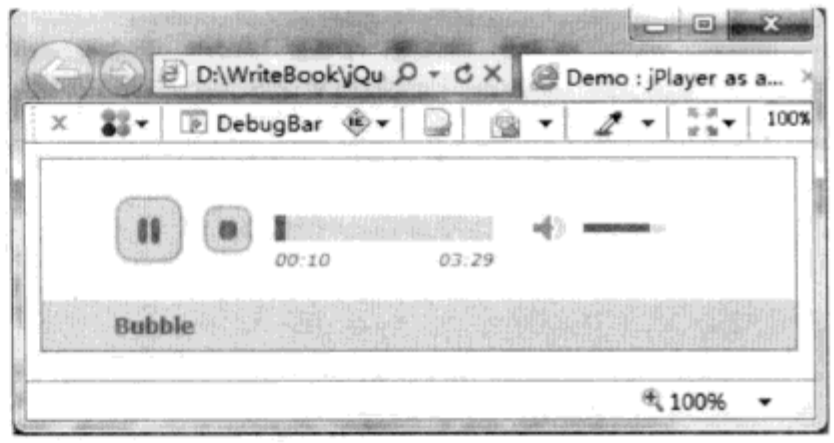


图 12.6　插件的基本使用方式（音频）示例效果

12.2.3　jPlayer 插件基本使用方式：播放视频文件

这个示例使用了插件的基本事件和方法，包括一些基本属性的使用。这个示例的静态页面与上例子类似，在此不过多讲解。不同于上面的例子的是，这里是一个视频文件的播放。下面看一下 JavaScript 功能代码：

```
1 <script type="text/javascript">
  //<![CDATA[
2   $(document).ready(function(){
3       $("#jquery_jplayer_1").jPlayer({
4           ready: function () {
5               $(this).jPlayer("setMedia", {
6                   m4v: "MediaFiles/Big_Buck_Bunny_Trailer_480x270_
                    h264aac.m4v",
7                   ogv: "MediaFiles/Big_Buck_Bunny_Trailer_480x270.ogv",
8                   poster: "MediaFiles/Big_Buck_Bunny_Trailer_480x270.
                    png"
9               });
10          },
11          ended: function (event) {
12              $(this).jPlayer("play");
```

```
13          },
14          swfPath: "js",
15          supplied: "m4v, ogv"
16      });
17  });
    //]]>
18 </script>
```

上述代码和前一个例子基本相同，只是在加入多媒体文件时指定了不同文件类型。效果如图 12.7 所示。

图 12.7　插件的基本使用方法（视频）示例效果

12.2.4　自定义播放器操作

在这个示例中，在页面上添加了两个插件，一个播放视频文件，另一个播放音频文件。每个播放器都加载了多个文件，在播放器的事件操作代码中进行了重新定义。首先，建立 HTML 静态页面：

```
1 <div class="jp-video jp-video-270p">
2           <div class="jp-type-playlist">
3               <div id="jquery_jplayer_1" class="jp-jplayer"></div>
4               <div id="jp_interface_1" class="jp-interface">
5                   <div class="jp-video-play"></div>
6                   <ul class="jp-controls">
7                       <li><a href="#" class="jp-play" tabindex="1">play
                        </a></li>
8                       <li><a href="#" class="jp-pause" tabindex="1">pause
                        </a></li>
9                       <li><a href="#" class="jp-stop" tabindex="1">stop
```

```
                    </a></li>
10                  <li><a href="#" class="jp-mute" tabindex="1">mute
                    </a></li>
11                  <li><a href="#" class="jp-unmute" tabindex="1">unmute
                    </a></li>
12                  <li><a href="#" class="jp-previous" tabindex="1">
                    previous</a></li>
13                  <li><a href="#" class="jp-next" tabindex="1">next
                    </a></li>
14              </ul>
15              <div class="jp-progress">
16                  <div class="jp-seek-bar">
17                      <div class="jp-play-bar"></div>
18                  </div>
19              </div>
20              <div class="jp-volume-bar">
21                  <div class="jp-volume-bar-value"></div>
22              </div>
23              <div class="jp-current-time"></div>
24              <div class="jp-duration"></div>
25          </div>
26          <div id="jp_playlist_1" class="jp-playlist">
27              <ul>
28                  <!-- The method Playlist.displayPlaylist() uses
                    this unordered list -->
29                  <li></li>
30              </ul>
31          </div>
32      </div>
33  </div>
34  <div id="jquery_jplayer_2" class="jp-jplayer"></div>
35  <div class="jp-audio">
36      <div class="jp-type-playlist">
37          <div id="jp_interface_2" class="jp-interface">
38              <ul class="jp-controls">
39                  <li><a href="#" class="jp-play" tabindex="1">play
                    </a></li>
40                  <li><a href="#" class="jp-pause" tabindex="1">pause
                    </a></li>
41                  <li><a href="#" class="jp-stop" tabindex="1">stop
                    </a></li>
42                  <li><a href="#" class="jp-mute" tabindex="1">mute
                    </a></li>
43                  <li><a href="#" class="jp-unmute" tabindex="1">unmute
                    </a></li>
44                  <li><a href="#" class="jp-previous" tabindex="1">
                    previous</a></li>
45                  <li><a href="#" class="jp-next" tabindex="1">next
                    </a></li>
46              </ul>
47              <div class="jp-progress">
48                  <div class="jp-seek-bar">
```

```
                        <div class="jp-play-bar"></div>
                    </div>
                </div>
                <div class="jp-volume-bar">
                    <div class="jp-volume-bar-value"></div>
                </div>
                <div class="jp-current-time"></div>
                <div class="jp-duration"></div>
            </div>
            <div id="jp_playlist_2" class="jp-playlist">
                <ul>
                    <!-- The method Playlist.displayPlaylist() uses
                    this unordered list -->
                    <li></li>
                </ul>
            </div>
        </div>
    </div>
```

上述代码较第一个示例多了两个样式名称，“jp-previous”表示前一个播放文件，“jp-next”表示后一个播放文件。另外，“jp-video-play”表示视频播放插件样式，“jp-audio”表示音频播放插件样式。

前面我们看到的插件示例均是单一文件播放，但是绝大部分情况下播放的多媒体文件是以列表形式出现的，即多个播放文件等待播放。这种功能的设计思路是通过对原有播放插件对象的继续封装，加入新的属性和方法，并加入一定显示样式，使插件从外观到功能上都具有播放多个媒体文件的功能。

实现步骤如下。

（1）创建一个能够容纳播放文件的播放列表对象，我们用这个对象封装了插件对象。

（2）在初始化插件时，指定配置参数。

（3）对播放列表中文件名部分的样式进行设定。

（4）对插件中播放前一文件和播放后一文件等相关功能的按钮编写实现代码。

（5）向播放列表中加入多媒体文件，并完善插件的相关事件功能。

```
<script type="text/javascript">
//<![CDATA[
  $(document).ready(function(){
  var Playlist = function(instance, playlist, options) {
                                              //创建播放列表对象原型
      var self = this;                        //播放列表对象
      this.instance = instance;               //字符串，指定的播放列表 HTML 元素
      this.playlist = playlist;               //数组，播放列表
      this.options = options;//初始化插件时的配置信息
      this.current = 0;       //当前播放的媒体文件索引，从第一个文件开始播放
      this.cssId = {          //插件样式，CSS 的 ID 选择值
          jPlayer: "jquery_jplayer_",
          interface: "jp_interface_",
          playlist: "jp_playlist_"
      };
      this.cssSelector = {};                          //CSS 的选择器
```

```
15      $.each(this.cssId, function(entity, id) {  //创建 ID 选择器字符串
16          self.cssSelector[entity] = "#" + id + self.instance;
17      });
18      if(!this.options.cssSelectorAncestor) {     //创建上级元素的样式选择器
19          this.options.cssSelectorAncestor = this.cssSelector.interface;
20      }
21      $(this.cssSelector.jPlayer).jPlayer(this.options); //初始化播放插件
22      $(this.cssSelector.interface + " .jp-previous").click(function() {
                                                          //播放前一个媒体文件
23          self.playlistPrev();
24          $(this).blur();
25          return false;
26      });
27      $(this.cssSelector.interface + " .jp-next").click(function() {
                                                          //播放后一个媒体文件
28          self.playlistNext();
29          $(this).blur();
30          return false;
31      });
32  };
33  Playlist.prototype = {
34      displayPlaylist: function() {                     //定义播放列表显示功能
35          var self = this;
36          $(this.cssSelector.playlist + " ul").empty();  //清空原有列表
37          for (i=0; i < this.playlist.length; i++) {
                                        //根据列表文件个数循环创建列表的 HTML 字符串
38              var listItem = (i === this.playlist.length-1) ? "<li class=
                'jp-playlist-last'>":"<li>";
39              listItem += "<a href='#' id='" + this.cssId.playlist + this.
                instance + "_item_" + i +"' tabindex='1'>"+ this.playlist[i].
                name+"</a>";
40              //创建列表项的链接
41              if(this.playlist[i].free) {     //是否是自由媒体文件
42                  var first = true;
43                  listItem += "<div class='jp-free-media'>(";
44                  $.each(this.playlist[i], function(property,value) {
                                                  //将不同类型的名称加入到列表中
45                      if($.jPlayer.prototype.format[property]) {
                                                  //检查媒体文件类型
46                          if(first) {
47                              first = false;
48                          } else {
49                              listItem += " | ";
50                          }
51                          listItem += "<a id='" + self.cssId.playlist +
                            self.instance + "_item_" + i + "_" + property
                            + "' href='" + value + "' tabindex='1'>" + property
                            + "</a>";
52                      }
53                  });
54                  listItem += ")</span>";
55              }
56              listItem += "</li>";
57              // 显示媒体文件类型
58              $(this.cssSelector.playlist + " ul").append(listItem);
                //向列表项中加入媒体文件对象
```

```
            //播放列表中媒体文件名链接的单击操作
59          $(this.cssSelector.playlist + "_item_" + i).data("index",
            i).click(function() {
60              var index = $(this).data("index");
61              if(self.current !== index) {    //单击的是否是当前播放文件
62                  self.playlistChange(index);//如果不是，则修改播放内容
63              } else {
64                  $(self.cssSelector.jPlayer).jPlayer("play");
                                                    //直接播放
65              }
66              $(this).blur();
67              return false;
68          });
69          // 强迫使用右键点击播放自由媒体文件
70          if(this.playlist[i].free) {
71              $.each(this.playlist[i], function(property,value) {
72                  if($.jPlayer.prototype.format[property]) {
                                                //检查属性是否是媒体文件类型
73                      $(self.cssSelector.playlist + "_item_" + i + "_"
                        + property).data("index", i).click(function() {
74                          var index = $(this).data("index");
75                          $(self.cssSelector.playlist + "_item_" +
                            index).click();
76                          $(this).blur();
77                          return false;
78                      });
79                  }
80              });
81          }
82      }
83  },
84  playlistInit: function(autoplay) { //播放列表初始化中检测是否是自动播放
85      if(autoplay) {
86          this.playlistChange(this.current);
                                            //如果是自动播放，则从当前文件播放
87      } else {
88          this.playlistConfig(this.current);
                                            //否则配置当前文件为播放文件，但不播放
89      }
90  },
    //下面部分是定义播放列表对象的播放相关功能，包括配置方法、改变播放文件方法、
      播放前一文件，播放后一文件
91  playlistConfig: function(index) {
                                    //配置播放文件，并修改列表中文件名的显示样式
92      $(this.cssSelector.playlist + "_item_" + this.current).
        removeClass("jp-playlist-current").parent().removeClass("jp-
        playlist-current");
93      $(this.cssSelector.playlist + "_item_" + index).addClass("jp-
        playlist-current").parent().addClass("jp-playlist-current");
94      this.current = index;
95      $(this.cssSelector.jPlayer).jPlayer("setMedia", this.playlist
        [this.current]);
96  },
97  playlistChange: function(index) {           //改变播放的媒体文件
98      this.playlistConfig(index);
99      $(this.cssSelector.jPlayer).jPlayer("play");
```

```
100     },
101     playlistNext: function() {                    //播放后一个媒体文件
102         var index = (this.current + 1 < this.playlist.length) ? this.
            current + 1 : 0;
103         this.playlistChange(index);
104     },
105     playlistPrev: function() {                    //播放前一个媒体文件
106         var index = (this.current - 1 >= 0) ? this.current - 1 : this.
            playlist.length - 1;
107         this.playlistChange(index);
108     }
109 };
    //对播放列表添加媒体文件，并对准备播放、结束播放、播放事件功能进行加工，这里添加的
      是视频文件
110 var videoPlaylist = new Playlist("1", [
111     {
112         name:"Big Buck Bunny Trailer",
113         free:true,
114         m4v:"MediaFiles/Big_Buck_Bunny_Trailer_480x270_h264aac.m4v",
115         ogv:"MediaFiles/Big_Buck_Bunny_Trailer_480x270.ogv",
116         poster:"MediaFiles/Big_Buck_Bunny_Trailer_480x270.png"
117     },
118     {
119         name:"Finding Nemo Teaser",
120         m4v: "MediaFiles/Finding_Nemo_Teaser_640x352_h264aac.m4v",
121         ogv: "MediaFiles/Finding_Nemo_Teaser_640x352.ogv",
122         poster: "MediaFiles/Finding_Nemo_Teaser_640x352.png"
123     },
124     {
125         name:"Incredibles Teaser",
126         m4v: "MediaFiles/Incredibles_Teaser_640x272_h264aac.m4v",
127         ogv: "MediaFiles/Incredibles_Teaser_640x272.ogv",
128         poster: "MediaFiles/Incredibles_Teaser_640x272.png"
129     }
130 ], {
131     ready: function() { //准备播放事件
132         videoPlaylist.displayPlaylist();
133         videoPlaylist.playlistInit(false); // 参数为自动播放的布尔值设置
134     },
135     ended: function() {                           //结束播放事件，转为播放下一个媒体文件
136         videoPlaylist.playlistNext();
137     },
138     play: function() {                            //播放事件
139         $(this).jPlayer("pauseOthers");           //暂停其他文件播放
140     },
141     swfPath: "js",
142     supplied: "ogv, m4v"
143 });
    //对播放列表添加媒体文件，并对准备播放、结束播放、播放事件功能进行加工，这里添加的
      是音频文件
144 var audioPlaylist = new Playlist("2", [
145     {
146         name:"Tempered Song",
147         mp3:"MediaFiles/Miaow-01-Tempered-song.mp3",
148         oga:"MediaFiles/Miaow-01-Tempered-song.ogg"
149     },
```

```
150     {
151         name:"Hidden",
152         mp3:"MediaFiles/Miaow-02-Hidden.mp3",
153         oga:"MediaFiles/Miaow-02-Hidden.ogg"
154     },
155     {
156         name:"Lentement",
157         free:true,
158         mp3:"MediaFiles/Miaow-03-Lentement.mp3",
159         oga:"MediaFiles/Miaow-03-Lentement.ogg"
160     },
161     {
162         name:"Lismore",
163         free:true,
164         mp3:"MediaFiles/Miaow-04-Lismore.mp3",
165         oga:"MediaFiles/Miaow-04-Lismore.ogg"
166     },
167     {
168         name:"The Separation",
169         mp3:"MediaFiles/Miaow-05-The-separation.mp3",
170         oga:"MediaFiles/Miaow-05-The-separation.ogg"
171     },
172     {
173         name:"Beside Me",
174         mp3:"MediaFiles/Miaow-06-Beside-me.mp3",
175         oga:"MediaFiles/Miaow-06-Beside-me.ogg"
176     },
177     {
178         name:"Bubble",
179         free:true,
180         mp3:"MediaFiles/Miaow-07-Bubble.mp3",
181         oga:"MediaFiles/Miaow-07-Bubble.ogg"
182     },
183     {
184         name:"Stirring of a Fool",
185         mp3:"MediaFiles/Miaow-08-Stirring-of-a-fool.mp3",
186         oga:"MediaFiles/Miaow-08-Stirring-of-a-fool.ogg"
187     },
188     {
189         name:"Partir",
190         mp3:"MediaFiles/Miaow-09-Partir.mp3",
191         oga:"MediaFiles/Miaow-09-Partir.ogg"
192     },
193     {
194         name:"Thin Ice",
195         free:true,
196         mp3:"MediaFiles/Miaow-10-Thin-ice.mp3",
197         oga:"MediaFiles/Miaow-10-Thin-ice.ogg"
198     }
199 ], {
200     ready: function() {
201         audioPlaylist.displayPlaylist();
202         audioPlaylist.playlistInit(false);// Parameter is a boolean for
            autoplay.
203     },
204     ended: function() {
205         audioPlaylist.playlistNext();
```

```
206     },
207     play: function() {
208         $(this).jPlayer("pauseOthers");
209     },
210     swfPath: "js",
211     supplied: "oga, mp3"
212 });
213 });
//]]>
214 </script>
```

上述代码第 3～32 行定义 Playlist 对象形式，其中 instance 表示插件列表对象实例，playlist 表示插件数组，options 表示插件初始化参数。第 8 行设定初始播放顺序为第一个媒体文件；第 9～13 行设定代表插件的 HTML 元素的 ID 匹配内容。第 15～17 行设定每个插件的 CSS 选择器。第 18～20 行获取上层样式设定的 ID。第 21 行利用 options 参数初始化播放器。第 22～26 行设定前一个媒体文件按钮的单击处理事件。第 27～32 行设定后一个媒体文件按钮的单击处理事件。

第 33～109 行设定 Playlist 对象属性及操作。其中第 34～83 行创建播放列表。第 38、39 行创建播放列表上单击播放文件名的超链接。第 41～54 行判断播放文件是否显示多种播放格式（mp3|oga）。第 59～68 行设定每个播放超链接的单击事件，如果单击的不是当前播放文件，则使用播放改变函数处理，否则重新播放当前文件。第 70～81 行判断播放超链接是否具有多种播放格式，强迫通过右键单击访问。

第 84～90 行为播放列表初始化函数，判断是否允许自动播放，如果允许则自动播放当前文件，否则配置播放文件位置。第 91～96 行配置播放列表播放顺序。第 97～100 行改变播放列表顺序。第 101～104 行播放下一个媒体文件。第 105～109 行播放前一个媒体文件。第 110～143 行初始化第一个视频播放插件。第 144～211 行初始化第二个音频播放插件，这两部分代码与前面的例子类似。效果如图 12.8 和图 12.9 所示。

图 12.8　自定义播放器操作视频播放

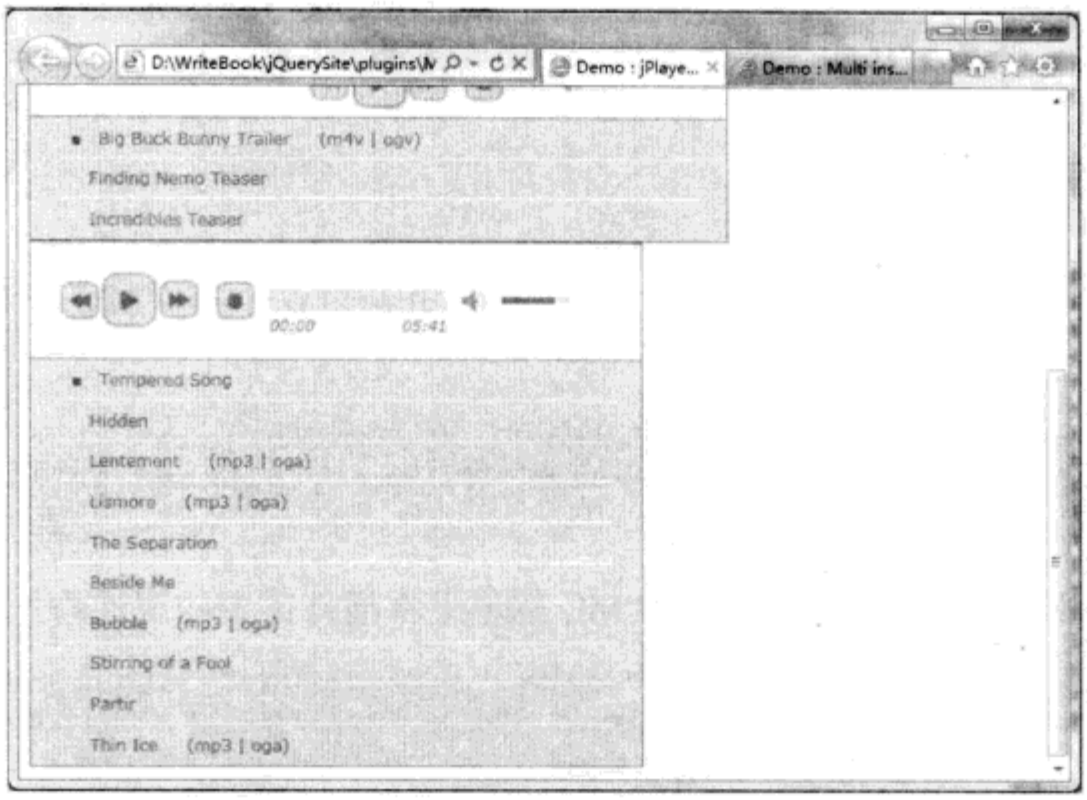


图 12.9　自定义播放器操作音频播放

12.3　jlEmbed 插件

jlEmbed 插件几乎可以嵌入任何一种媒体类型到网页上。它内建了音乐列表支持。这个插件和前面所介绍的插件不同之处在于，它的所有属性设定都写在元素中，而不是通过初始化函数来完成设定。它的英文网址为 http://jasonlau.biz/home/jlembed-for-jquery。

下面看一下这个插件的属性说明，如表 12.4 所示。

表 12.4　jlEmbed插件属性说明

属　　性	类型	默认值	说　　明
autostart	布尔	true	页面加载后自动播放，但是在 adobeflash, musicplayer, silverlight 模式下无效
caption	字符串	null	在嵌入的播放器下加入文本或者 HTML
controls	布尔	false	只在 youtube 模式下使用，在播放器下添加通用控制按钮
debug	布尔	false	只在 youtube 模式下使用，显示嵌入对象的父内容
error_alert	布尔	false	当插件运行发生错误时抛出 JavaScript 的警告对话框
format	字符串		HTML 输出格式
height	数值	varis	插件的高
id	字符串		嵌入对象的 ID
loop	布尔	false	自动循环播放
mode	字符串	null	设置插件模式
params	字符串		自定义参数
screw	布尔	false	当播放器在页面可见时加载
shuffle	布尔	false	重新排列播放列表
src	字符串		媒体资源位置
stop_onerror	布尔	false	只在 youtube 模式下使用，当发生错误时停止
volume	数值	100	只在 youtube 模式下使用，设置默认音量
width	数值	varies	嵌入播放器的高

下面通过示例来了解这个插件如何加载各种媒体文件。

12.3.1　插件基本使用

这个示例使用插件加载一个 Flash 文件。前面介绍过这个插件属于自动初始化，所以我们只需要引入相关的 jQuery 库文件：

```
<script type="text/javascript" src="../../jslib/jquery-1.6.js"></script>
<script type="text/javascript" src="JS/jquery.jlembed.js"></script>
                                                    //插件核心功能文件
```

然后建立静态页面，并在页面的 HTML 标记上将插件需要的属性添加进去就可以了：

```
<div id="mydiv" class="jlembed" data-id="myplayer" data-mode="adobeflash"
data-params='{"wmode":"transparent", "quality":"high", "allowscriptaccess":
"always", "allowfullscreen":"true"}' data-format="swfobject" data-src=
'MediaFile/example.swf' data-width="294" data-height="196" data-caption=
"This is funny." data-screw="true"></div>
```

上述代码实现了 Flash 文件的加载，这里需要说明的是如果希望在元素上内嵌插件，则该元素的类属性需要设定为“jlembed”，后面的示例也是如此。表 12.4 所提到的所有属性如果在标记中需要指定，则要在属性前加上 data 前缀。这个示例使用 params 设定了 Flash 文件加载效果，窗口模式透明（"wmode":"transparent"），画面质量高（"quality":"high"），允许脚本访问（"allowscriptaccess": "always"），允许全屏（"allowfullscreen":"true"）。效果如图 12.10 所示。

12.3.2　使用 Windows Media Player 播放

这个示例使用了 Windows Media Player 播放文件，代码如下：

```
<div id="mydiv" class="jlembed" data-id="myplayer" data-mode="windowsmedia"
data-params='{"console":"cons","controls":"all"}' data-format="objectembed"
data-src="MediaFile/mixdown.mp3"    data-width="300"    data-height="40"
data-autostart="false" data-loop="true" data-caption="This is my song."
data-screw="true"></div>
```

上述代码指定了播放 MP3 文件的格式，并使用 Windows Media Player 软件来播放。如果你的计算机中也安装了 Real Player，则将 data-mode= "windowsmedia" 修改成 data-mode="realplayer"就可以使用 Real Player 来播放了。效果如图 12.11 所示。

图 12.10　jlEmbed 基本使用示例效果

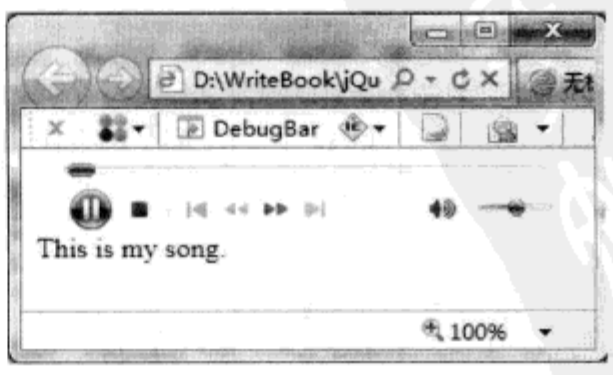


图 12.11　使用 Windows Media Player 播放 MP3

12.3.3　使用 QuickTime 播放

这个示例使用 QuickTime 播放多媒体文件。代码如下：

```
<div id="mydiv" class="jlembed" data-id="myplayer" data-mode="quicktime"
data-params='{"controller":"true"}' data-format="objectembed" data-src=
'{"Mixdown1":"MediaFile/mixdown.mp3","Mixdown2":"MediaFile/mixdown.mp3"
}' data-width="300" data-height="45" data-autostart="false" data-loop=
"true" data-shuffle="false" data-caption="This is my song." data-screw=
"true"></div>
```

上述代码指定了使用 QuickTime 播放 MP3 多媒体资源文件，效果如图 12.12 所示。

12.3.4　使用 SilverLight 播放

这个示例使用 SilverLight 播放多媒体文件。代码如下：

```
<div id="mydiv" class="jlembed" data-id="myplayer" data-mode="silverlight"
data-params='{"onLoad":"__slLoad0", "onError":"__slError0", "windowless":
"false", "maxFramerate":"30"}' data-format="objectembed" data-src='MediaFile/
WeatherWidget.xap' data-width="300" data-height="250" data-caption="This
is a test caption." data-screw="true"></div>
```

上述代码对 SilverLight 的使用做了参数设定，加载事件（"onLoad":"__slLoad0"），发生错误事件（"onError":"__slError0"），不显示无窗口插件（"windowless":"false"），设置每秒可呈现的最大帧数（"maxFramerate":"30"）。效果如图 12.13 所示。

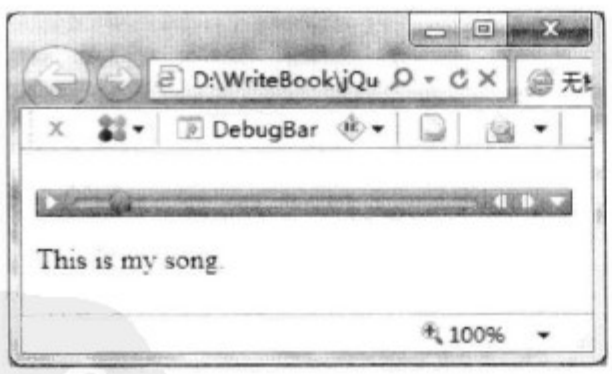


图 12.12　使用 QuickTime 播放 MP3

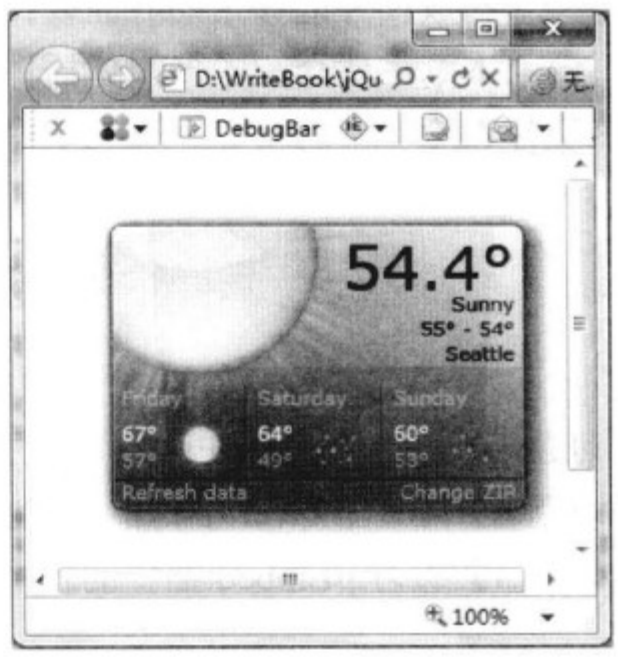


图 12.13　利用 SilverLight 播放多媒体文件

12.3.5　使用 Flash Player 播放

这个示例调用了 Adobe 的 Flash Player 播放器播放 Flash 文件，代码如下：

```
<div id="mydiv" class="jlembed" data-id="myplayer" data-mode="adobeflash"
data-params='{"wmode":"transparent", "quality":"high", "allowscriptaccess":
"always", "allowfullscreen":"true"}' data-format="swfobject" data-src=
'MediaFile/example.swf' data-width="294" data-height="196" data-caption=
"This is funny." data-screw="true"></div>
```

上述代码与 12.3.1 节介绍的示例基本相同，效果如图 12.14 所示。

图 12.14　利用 Adobe Flash Player 播放 Flash 文件

12.4　小　　结

多媒体播放插件能够帮助界面设计人员更灵活地表现页面内容，也能让用户看到更生动的页面。本章主要介绍了多媒体插件的使用方法。重点内容是插件的功能与外观定制，同时这部分也是本章的难点。下一章将介绍 jQuery 动画设计。

12.5　习　　题

【习题 1】练习 jQuery.Flash 插件的使用方法。
【习题 2】练习 jPlayer 插件的使用方法。
【习题 3】练习 jlEmbed 插件的使用方法。

第13章　动 画 设 计

动画可以更直观、更生动地表现出设计者的意图。动画可以通过专业的动画软件如Flash等来制作。现在网页上动画所占的比例越来越大。jQuery除了可以实现前面章节所完成的效果外，还可以在一定程度上实现动画效果。本章讲解如何利用jQuery在网页上制作动画。

13.1　jQuery动画基础

本节将介绍利用jQuery实现动画效果的基础知识。在前面的章节中介绍了jQuery制作动画的一些相关函数，本节总结一下这些函数，并通过示例来说明应用。

13.1.1　jQuery动画函数

jQuery的动画函数总共分成4类。

（1）基本动画函数：既有透明度渐变，又有滑动效果，是最常用的动画效果函数。

（2）滑动动画函数：仅适用滑动渐变动画效果。

（3）淡入淡出动画函数：仅适用透明度渐变动画效果。

（4）自定义动画函数：作为上述三种动画函数的补充和扩展。

下面将这几种动画函数总结一下。首先是基本动画函数，如表13.1所示。

表13.1　jQuery基本动画函数总结

函　　数	使 用 方 式	说　　明
show()	$("p").show()	显示隐藏的匹配元素。 这个函数就是 'show(speed, [callback])'的无动画版本。如果选择的元素是可见的，这个方法将不会改变任何东西。无论这个元素是通过hide()方法隐藏的还是在CSS里设置了display:none;，这个方法都将有效
show(speed,[callback])	$("p").show("slow"); $("p").show("fast",function(){ 　$(this).text("Animation Done!"); }); });	以优雅的动画显示所有匹配的元素，并在显示完成后可选地触发一个回调函数。可以根据指定的速度动态地改变每个匹配元素的高度、宽度和不透明度。在jQuery 1.3中，padding和margin也会有动画，效果更流畅
hide()	$("p").hide()	隐藏显示的元素。这个函数就是'hide(speed, [callback])' 的无动画版本。如果选择的元素是隐藏的，这个方法将不会改变任何东西

续表

函　数	使 用 方 式	说　明

hide(speed,[callback])	$("p").hide("slow");
	$("p").hide("fast",function(){
	　alert("Animation Done.");
	});
		以优雅的动画隐藏所有匹配的元素，并在显示完成后可选地触发一个回调函数。可以根据指定的速度动态地改变每个匹配元素的高度、宽度和不透明度。在 jQuery 1.3 中，padding 和 margin 也会有动画，效果更流畅

toggle()	$("p").toggle()	切换元素的可见状态。如果元素是可见的，切换为隐藏的；如果元素是隐藏的，切换为可见的

toggle(switch)	var flip = 0;
	$("button").click(function () {
	　$("p").toggle(flip++ % 2 == 0);
	});
		根据 switch 参数切换元素的可见状态（true 为可见，false 为隐藏）。如果 switch 设为 true，则调用 show()方法来显示匹配的元素，如果 switch 设为 false，则调用 hide()来隐藏元素

toggle(speed,[callback])	$("p").toggle("slow");
	$("p").toggle("fast",function(){
	　alert("Animation Done.");
	});
		以优雅的动画切换所有匹配的元素，并在显示完成后可选地触发一个回调函数。可以根据指定的速度动态地改变每个匹配元素的高度、宽度和不透明度。在 jQuery 1.3 中，padding 和 margin 也会有动画，效果更流畅

滑动动画函数如表 13.2 所示。

表 13.2　jQuery滑动动画函数总结

函　数	使 用 方 式	说　明

slideDown(speed,[Callback])	$("p").slideDown("slow");
	$("p").slideDown("fast",function(){
	　alert("Animation Done.");
	});
		通过高度变化（向下增大）来动态地显示所有匹配的元素，在显示完成后可选地触发一个回调函数。这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式显示出来。在 jQuery 1.3 中，上下的 padding 和 margin 也会有动画，效果更流畅

slideUp(speed,[Callback])	$("p").slideUp("slow");
	$("p").slideUp("fast",function(){
	　alert("Animation Done.");
	});
		通过高度变化（向上减小）来动态地隐藏所有匹配的元素，在隐藏完成后可选地触发一个回调函数

slideToogle(speed,[Callback])	$("p").slideToggle("slow");
	$("p").slideToggle("fast",function(){
	　alert("Animation Done.");
	});
		通过高度变化来切换所有匹配元素的可见性，并在切换完成后可选地触发一个回调函数

淡入淡出动画函数如表 13.3 所示。

表 13.3　jQuery淡入淡出动画函数总结

函　数	使 用 方 式	说　明

fadeIn(speed,[Callback])	$("p").fadeIn("slow");
	$("p").fadeIn("fast",function(){
	　alert("Animation Done.");
	});
		通过不透明度的变化来实现所有匹配元素的淡入效果，并在动画完成后可选地触发一个回调函数。这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化

续表

函　　数	使 用 方 式	说　　明

fadeOut(speed,[Callback])	$("p").fadeOut("slow");
	$("p").fadeOut("fast",function(){
	　alert("Animation Done.");
	});
		通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数

fadeTo(speed,[Callback])	$("p").fadeTo("slow", 0.66);
	$("p").fadeTo("fast", 0.25, function()
	{
	　alert("Animation Done.");
	});
		把所有匹配元素的不透明度以渐进方式调整到指定的不透明度，并在动画完成后可选地触发一个回调函数

自定义动画函数如表 13.4 所示。

表 13.4　jQuery自定义动画函数总结

函　　数	使 用 方 式	总　　结
animate(param, [dur],[e],[fn])	$("p").animate({ 　height: 'toggle', opacity: 　'toggle' }, "slow"); $("p").animate({ 　opacity: 'show' }, "slow", "easein");	指定动画形式及结果样式属性对象。每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。param，一组包含作为动画属性和终值的样式属性以及其值的集合；duration，三种预定速度之一的字符串（“slow”、“normal”or“fast”）或表示动画时长的毫秒数值（如1000）；easing，要使用的擦除效果的名称（需要插件支持），jQuery 默认提供“linear”和“swing”；callback，在动画完成时执行的函数
animate (param,options)	$("p").animate({ 　left: 50, opacity: 'show' }, { duration: 500 }); $("p").animate({ 　opacity: 'show' }, { duration: "slow", easing: "easein" });	指定动画形式及结果样式属性对象。每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。param，一组包含作为动画属性和终值的样式属性以及其值的集合；options，一组包含动画选项的值的集合；duration，三种预定速度之一的字符串（“slow”、“normal”or“fast”）或表示动画时长的毫秒数值（如 1000）；easing，要使用的擦除效果的名称（需要插件支持），jQuery 默认提供“linear”和“swing”；complete，在动画完成时执行的函数；step，动画步进时执行的函数；queue，（默认值：true）设定为 false 将使此动画不进入动画队列
stop([clearQueue], [gotoEnd])	$(".block").stop();	停止所有在指定元素上正在运行的动画。如果队列中有等待执行的动画（并且 clearQueue 没有设为 true），它们将被马上执行。clearQueue，如果设置成 true，则清空队列，可以立即结束动画；gotoEnd，让当前正在执行的动画立即完成，并且重设 show 和 hide 的原始样式，调用回调函数等
delay(duration, [queueName])	$('#foo').slideUp(300). delay(800).fadeIn(400);	设置一个延时来推迟执行队列中之后的项目。用于将队列中的函数延时执行。它既可以推迟动画队列的执行，也可以用于自定义队列。duration，延时时间，单位为毫秒；queueName，队列名词，默认是 Fx，动画队列

除了上述这些函数外，jQuery 中与动画相关的还有一个设置选项 jQuery.fx.off，它关闭页面上所有的动画。把这个属性设置为 true 可以立即关闭所有动画（所有效果会立即执行完毕）。有些情况下可能需要这样，例如，

- 你在配置比较低的电脑上使用 jQuery。
- 你的一些用户由于动画效果而遇到了可访问性问题。

当把这个属性设成 false 之后，可以重新开启所有动画。

13.1.2　jQuery 动画简单示例

下面用一个自定义动画的简单示例来看一下 jQuery 制作动画的过程。在这个示例中我们所要达成的效果是：在页面上垂直摆放几个超链接，当将鼠标悬停在其中一个超链接上时，这个超链接动态向右缩进，当鼠标移开超链接时它恢复到原来的位置。实现原理就是利用了前面总结的自定义动画函数 animate()，并在这个函数中修改超链接的 padding-left 属性，而且给定了动画持续时间，效果如图 13.1 所示。

图 13.1　jQuery 简单动画效果一

为了实现这个效果，我们在 HTML 静态页面部分创建列表嵌套超链接：

```
<ul><li><a href="#" class="link">首页</a></li>
  <li><a href="#" class="link">菜单一</a></li>
   <li><a href="#" class="link">菜单二</a></li>
   <li><a href="#" class="link">菜单三</a></li>
   <li><a href="#" class="link">菜单四</a></li>
   <li><a href="#" class="link">菜单五</a></li>
   <li><a href="#" class="link">菜单六</a></li>
 </ul>
```

然后，引入 jQuery 库文件：

```
<script type="text/javascript" src="jslib/jquery-1.6.js"></script>
```

最后，加上 JavaScript 功能代码：

```
<script type="text/javascript">
  $(document).ready(function() {
      $('a.link').hover(function() {
          $(this).animate({ paddingLeft: '20px' },400);
                                          //调用自定义动画实现列表项右缩进
      }, function() {
          $(this).animate({ paddingLeft: 0 }, 400);
                                          //调用自定义动画实现列表项位置还原
      });
  });
</script>
```

上述代码第 3 行使用了模仿鼠标悬停与离开事件函数。第 4 行使用了自定义动画函数，

并设定超链接左缩进 20 像素的动画效果，整个动画持续时间为 400 毫秒，这个效果在鼠标悬停在超链接上时发生。第 6 行同样使用了自定义动画函数，将超链接位置恢复为原来的位置，动画时间持续为 400 毫秒，这个效果在鼠标离开超链接时发生。如果将鼠标依次滑过每个超链接，则动画效果会更清晰。效果如图 13.2 所示。

图 13.2　jQuery 简单动画效果二

13.2　jQuery UI 中实现的动画效果

在 jQuery UI 中提供了多种动画效果供我们参考。除了前面介绍的基本动画效果、滑动效果、淡入淡出效果外，还有百叶窗效果、跳跃效果、缩减效果、移动效果、分裂效果、折叠效果、高亮淡入淡出效果、脉冲闪烁效果、摆动效果等。本节逐一介绍它们的实现。

13.2.1　jQuery UI 动画特效使用

首先，参照 jQuery UI 中给出的示例来看一下如何使用动画特效。这个示例在 jQuery UI 的 demos 文件下的 show 子文件夹中。它使用了 jQuery UI 提供的各种特效显示一个消息层。

该示例的 HTML 源代码如下：

```
1 <div class="demo">
2 <div class="toggler">
3   <div id="effect" class="ui-widget-content ui-corner-all">
4       <h3 class="ui-widget-header ui-corner-all">Show</h3>
5       <p>
6           Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
            Sed pede. Nulla lorem metus, adipiscing ut, luctus sed, hendrerit
            vitae, mi.
7       </p>
8   </div>
9 </div>
10 <select name="effects" id="effectTypes">
11  <option value="blind">Blind</option>
12  <option value="bounce">Bounce</option>
13  <option value="clip">Clip</option>
14  <option value="drop">Drop</option>
```

```
15  <option value="explode">Explode</option>
16  <option value="fold">Fold</option>
17  <option value="highlight">Highlight</option>
18  <option value="puff">Puff</option>
19  <option value="pulsate">Pulsate</option>
20  <option value="scale">Scale</option>
21  <option value="shake">Shake</option>
22  <option value="size">Size</option>
23  <option value="slide">Slide</option>
24 </select>
25 <a href="#" id="button" class="ui-state-default ui-corner-all">Run
    Effect</a>
26 </div><!-- End demo -->
27 <div class="demo-description">
28 <p>Click the button above to preview the effect.</p>
29 </div><!-- End demo-description -->
```

上述代码第 2～9 行是需要触发让其显示的信息层；第 10～24 层是选择各种效果的下拉列表框部分。

要实现这些动画效果，需要引入相应的 jQuery 库文件：

```
1   <script src="../../jquery-1.6.2.js"></script>
2   <script src="../../ui/jquery.effects.core.js"></script>
3   <script src="../../ui/jquery.effects.blind.js"></script>
4   <script src="../../ui/jquery.effects.bounce.js"></script>
5   <script src="../../ui/jquery.effects.clip.js"></script>
6   <script src="../../ui/jquery.effects.drop.js"></script>
7   <script src="../../ui/jquery.effects.explode.js"></script>
8   <script src="../../ui/jquery.effects.fold.js"></script>
9   <script src="../../ui/jquery.effects.highlight.js"></script>
10  <script src="../../ui/jquery.effects.pulsate.js"></script>
11  <script src="../../ui/jquery.effects.scale.js"></script>
12  <script src="../../ui/jquery.effects.shake.js"></script>
13  <script src="../../ui/jquery.effects.slide.js"></script>
```

上述各个 JS 文件就是 jQuery UI 提供动画特效的插件文件，每个插件文件名中都有 effects 这个单词，表示效果。

在 JavaScript 代码中按照下拉列表框中选择的效果运行插件：

```
1   <script>
2   $(function() {
3       //执行当前选择动画特效
4       function runEffect() {
5           //获取特效类型
6           var selectedEffect = $( "#effectTypes" ).val();
7           //很多特效不需要配置参数
8           var options = {};
9           //特效类型的相关参数设定
10          if ( selectedEffect === "scale" ) {
11              options = { percent: 100 };
12          } else if ( selectedEffect === "size" ) {
```

```
13              options = { to: { width: 280, height: 185 } };
14          }
15          //运行特效
16          $( "#effect" ).show( selectedEffect, options, 500, callback );
17      };
18      //回调函数使用了一个隐藏的消息层
19      function callback() {
20          setTimeout(function() {
21              $( "#effect:visible" ).removeAttr( "style" ).fadeOut();
22          }, 1000 );
23      };
24      //设置特效效果
25      $( "#button" ).click(function() {
26          runEffect();
27          return false;
28      });
29      $( "#effect" ).hide();
30  });
31  </script>
```

上述代码第 6 行获取下拉列表框的选择特效名称。第 8 行定义特效执行选项。第 10～14 行判断是使用原规模还是调整尺寸特效，如果使用原规模，则将缩放比例设成百分之百，如果是调整尺寸，则设定消息层宽和高。第 16 行利用 jQuery 的 show()函数调用特效，第 1 个参数是特效名称，第 2 个参数是特效选项，第 3 个参数是特效持续时间，第 4 个参数表示回调函数。第 19～23 行表示当特效把消息层显示出来后间隔一秒钟隐藏消息层。页面的初始加载效果如图 13.3 所示。

图 13.3　jQuery UI 特效初始页面

13.2.2　百叶窗效果

百叶窗效果使用了 jQuery UI 的 jquery.effects.core.js 和 jquery.effects.blind.js 文件。第一个文件是动画特效的核心文件，其中提供了一些实现特效的基本和常用操作函数。第二个文件是实现百叶窗效果的插件文件。它的设计思想是根据百叶窗打开方向设定动画的操作高度或者宽度，然后利用自定义动画实现动画效果。在这里使用了 jQuery 的 animate()自定义动画函数来完成动画效果。css()样式设定函数修改消息部分的样式，主要是宽和高两个参数。hide()为隐藏函数，将消息部分隐藏起来；height()为获取元素高度函数。show()为显示元素函数。width()为获取元素宽度函数。主要设计思路是根据动画显示方向（纵向或横向），通过自定义动画的特效将消息部分的高或者宽调整为原始大小。下面看一下百叶窗效果插件的功能代码：

```
1 (function( $, undefined ) {
2 $.effects.blind = function(o) {
3   return this.queue(function() {
4       //获取特效作用对象
```

```
5       var el = $(this), props = ['position','top','bottom','left',
        'right'];
6       //设置运行参数
7       var mode = $.effects.setMode(el, o.options.mode || 'hide');
        //设置对象的显示模式
8       var direction = o.options.direction || 'vertical';
        //设置动画的作用方向
9       //调整对象状态
10      $.effects.save(el, props); el.show();  //存储对象，并将对象显示出来
11      var wrapper = $.effects.createWrapper(el).css({overflow:'hidden'});
                                                  //为对象创建一个外部封装
12      var ref = (direction == 'vertical') ? 'height' : 'width';
                                                  //设定动画变化参考方向
        //设定包含封装元素的变化距离
13      var distance = (direction == 'vertical') ? wrapper.height() :
        wrapper.width();
14      if(mode == 'show') wrapper.css(ref, 0);//设定封装元素的初始状态
15      //使用自定义动画
16      var animation = {};
17      animation[ref] = mode == 'show' ? distance : 0;    //设定动画参数
18      //对封装元素使用自定义动画
19      wrapper.animate(animation, o.duration, o.options.easing, function() {
20          if(mode == 'hide') el.hide();          //隐藏元素
21          $.effects.restore(el, props); $.effects.removeWrapper(el);
                                        //恢复元素的初始状态，并移除封装元素
22          if(o.callback) o.callback.apply(el[0], arguments);
23          el.dequeue();
24      });
25  });
26 };
27 })(jQuery);
```

对于插件的编写要在第 15 章才会介绍，在这里只讨论它的实现过程。上述代码第 3 行返回当前调用插件的一个动画队列，并在这个队列中定义了回调函数实现百叶窗效果。第 5 行创建元素，接收调用插件的对象，并定义位置选项。第 7 行设置选项中的显示模式为隐藏。第 8 行设置选项中的动画方向为垂直。第 10 行保存元素对象的位置，并显示元素对象。第 11 行调用了 jquery.effects.core.js 文件中创建的包装元素函数<DIV>，并设定当元素内容溢出时隐藏。

第 12 行根据动画方向选择修改宽或者高属性。第 13 行根据动画方向选择使用包装元素的宽或者高的属性值。第 14 行设置如果元素是显示的则隐藏。第 16 行定义动画参数数组。第 17 行根据显示状态设置设定动画参数数组的值。第 19 行调用包装元素的自定义动画函数。第 20 行判断显示模式，如果隐藏则调用元素对象的隐藏方法。第 21 恢复先前存储的元素属性。第 22 行调用回调函数。第 23 行将动画从队列删除。为了能够看清效果，我们可以将动画时间延长。效果如图 13.4 和图 13.5 所示。

13.2.3　跳跃效果

跳跃效果需要应用到动画特效的核心文件和跳跃效果的插件文件 jquery.effects.bounce.js。

它的设计思想是通过设定动画跳跃的行为参数、跳跃次数、跳跃高度、跳跃用时、循环调用自定义动画实现跳跃过程。所谓跳跃，实际上是通过变换设定元素的顶端坐标值来完成的。这里主要使用了 jQuery 的 animate()自定义动画函数、css()样式设定函数、show()显示元素函数。主要设计思路是通过不断改变消息部分的顶端坐标位置来实现跳跃动画效果。插件源代码如下：

图 13.4　百叶窗效果一

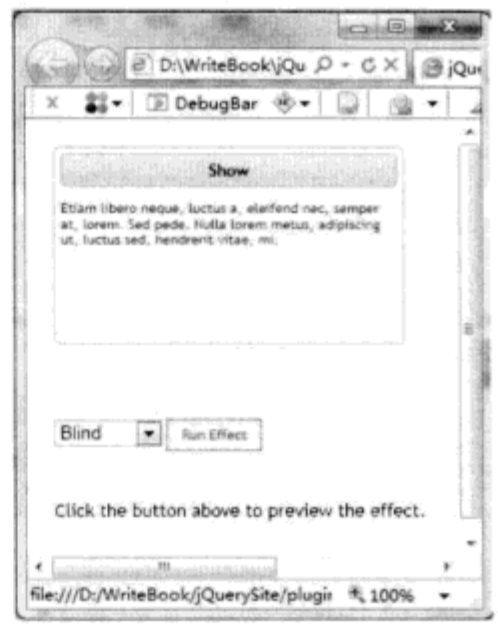


图 13.5　百叶窗效果二

```
1 (function( $, undefined ) {
2 $.effects.bounce = function(o) {
3   return this.queue(function() {
4       //获取特效作用对象
5       var el = $(this), props = ['position','top','bottom','left',
        'right'];
6       //设置动画运行参数
7       var mode = $.effects.setMode(el, o.options.mode || 'effect');
                                                    //设置对象的显示模式
8       var direction = o.options.direction || 'up';   //设置动画方向
9       var distance = o.options.distance || 20;       //默认动画运行距离
10      var times = o.options.times || 5;              //设置动画持续次数
11      var speed = o.duration || 250;                 //设置每次跳动速度
12      if (/show|hide/.test(mode)) props.push('opacity'); //设置透明参数
13      //调整对象
14      $.effects.save(el, props); el.show();          //存储对象，并显示对象
15      $.effects.createWrapper(el);                   //创建对象的封装元素
16      var ref = (direction == 'up' || direction == 'down') ? 'top' : 'left';
                                                    //设定动画运行参考坐标
17      var motion = (direction == 'up' || direction == 'left') ? 'pos' : 'neg';
                                                    //设定动画动作参数
18      var distance = o.options.distance || (ref == 'top' ? el.outerHeight
        ({margin:true}) / 3 : el.outerWidth({margin:true}) / 3);
                                                //设定动画初次实际运行距离
19      if (mode == 'show') el.css('opacity', 0).css(ref, motion == 'pos' ?
        -distance : distance);                  //设定样式
20      if (mode == 'hide') distance = distance / (times * 2);
                                                //如果对象隐藏则动画距离缩减
21      if (mode != 'hide') times--;            //动画执行次数递减
```

```
22      // 使用自定义动画
23      if (mode == 'show') {                          //第一次显示跳跃动画
24          var animation = {opacity: 1};              //设置透明度
25          animation[ref] = (motion == 'pos' ? '+=' : '-=') + distance;
                                                       //设定初次向上跳跃
26          el.animate(animation, speed / 2, o.options.easing);//执行动画
27          distance = distance / 2;                   //跳动距离减半
28          times--;                                   //跳动次数自减
29      };
30      for (var i = 0; i < times; i++) {              //循环跳动 3 次
31          var animation1 = {}, animation2 = {};
32          animation1[ref] = (motion == 'pos' ? '-=' : '+=') + distance;
                                                       //向下跳动
33          animation2[ref] = (motion == 'pos' ? '+=' : '-=') + distance;
                                                       //向上跳动
34          el.animate(animation1, speed / 2, o.options.easing).animate
            (animation2, speed / 2, o.options.easing);
35          distance = (mode == 'hide') ? distance * 2 : distance / 2;
                                                       //跳动距离减半
36      };
37      if (mode == 'hide') {
38          var animation = {opacity: 0};
39          animation[ref] = (motion == 'pos' ? '-=' : '+=')  + distance;
40          el.animate(animation, speed / 2, o.options.easing, function(){
41              el.hide(); // Hide
42              $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                       //移除外部封装
43              if(o.callback) o.callback.apply(this, arguments);
                                                       //回调方法
44          });
45      } else {                                       //最后一次跳动
46          var animation1 = {}, animation2 = {};
47          animation1[ref] = (motion == 'pos' ? '-=' : '+=') + distance;
48          animation2[ref] = (motion == 'pos' ? '+=' : '-=') + distance;
49          el.animate(animation1, speed / 2, o.options.easing).animate
            (animation2, speed / 2, o.options.easing, function(){
50              $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                       //恢复对象
51              if(o.callback) o.callback.apply(this, arguments);
52          });
53      };
54      el.queue('fx', function() { el.dequeue(); });
55      el.dequeue();
56  });
57 };
58 })(jQuery);
```

上述代码第 7 行设置显示模式。第 8～11 行设定跳跃行为参数、跳动方向、跳动距离、跳动次数、跳动间隔。第 12 行避免 IE 对于透明度的问题设置。第 13、14 行保存元素对象属性并显示。第 15 行创建包装标记。第 16 行根据跳跃方向选择坐标类型。第 17 行根据跳跃方向选择操作参数。第 18 行获取跳跃距离。第 19 行设置如果元素为显示状态则转换。

第 20 行设置如果元素为隐藏状态则距离递减。

第 21 行设置如果模式不是隐藏则跳跃次数减 1。第 23～29 行利用自定义动画行执行跳跃动画。第 30～36 行利用自定义动画上下跳跃循环。第 37～45 行利用自定义动画最后一次跳跃，并恢复元素属性，移除包装元素，调用回调函数。第 46～53 行利用自定义动画继续跳跃，并恢复元素属性，移除包装元素，调用回调函数。第 55、56 行从队列删除动画。效果如图 13.6 所示。

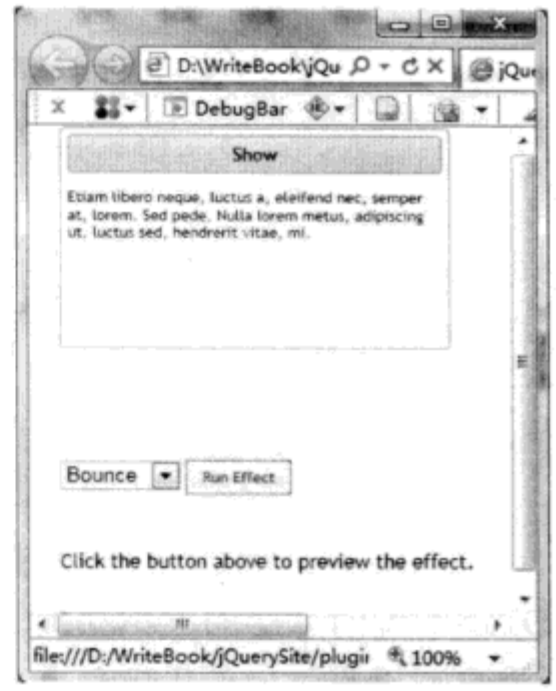


图 13.6　跳跃动画效果

13.2.4　缩减效果

缩减效果需要应用到动画特效的核心文件和缩减效果的插件文件 jquery.effects.clip.js。它的设计思想是设定元素起始出现位置，即元素高度的一半，然后元素的上下两部分依次向两边展开。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.clip = function(o) {
3   return this.queue(function() {
4       //获取动画作用对象
5     var el = $(this), props = ['position','top','bottom','left','right',
      'height','width'];
6     //设置对象运行参数
7     var mode = $.effects.setMode(el, o.options.mode || 'hide');
                                                    //设置显示方式
8     var direction = o.options.direction || 'vertical'; //默认动画方向
9     // 调整对象状态
10    $.effects.save(el, props); el.show();          //存储对象并显示
11    var wrapper = $.effects.createWrapper(el).css({overflow:'hidden'});
                                                    //创建封装元素
12    var animate = el[0].tagName == 'IMG' ? wrapper : el;
                                                  //获取动画作用对象
13    var ref = {                                   //动画运行参数
14        size: (direction == 'vertical') ? 'height' : 'width',
15        position: (direction == 'vertical') ? 'top' : 'left'
16    };
17    var distance = (direction == 'vertical') ? animate.height() : animate.
      width();                                      //获取动画运行距离
18    if(mode == 'show') { animate.css(ref.size, 0); animate.css(ref.
      position,distance / 2); }                     //初始动画对象
19    //使用自定义动画对象
20    var animation = {};
21    animation[ref.size] = mode == 'show' ? distance : 0;
                                                  //设定完整显示对象高度
22    animation[ref.position] = mode == 'show' ? 0 : distance / 2;
                                                  //设定开始动画位置，为高度一半
23    //使用自定义动画
```

```
24      animate.animate(animation, { queue: false, duration: o.duration,
        easing: o.options.easing, complete: function() {
25          if(mode == 'hide') el.hide();          //隐藏对象
26          $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                   //恢复对象，并移除封装
27          if(o.callback) o.callback.apply(el[0], arguments);
28          el.dequeue();
29      }});
30  });
31 };
32 })(jQuery);
```

上述代码同百叶窗效果类似，也是使用包装元素<DIV>实现消息层的大小动画。但是，这个插件的显示是从消息层垂直方向的中间位置向上下扩展，效果如图 13.7 所示。

图 13.7　缩减动画效果

13.2.5　移动效果

移动效果需要应用到动画特效的核心文件和移动效果的插件文件 jquery.effects.drop.js。它的设计思想类似于跳跃效果，只是在移动过程中通过一次自定义动画将元素起始左端位置加上一定的像素值，使元素向右移动。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.drop = function(o) {
3   return this.queue(function() {
4       //获取动画作用对象
5       var el = $(this), props = ['position','top','bottom','left','right',
        'opacity'];
6       //设定动画参数
7       var mode = $.effects.setMode(el, o.options.mode || 'hide');
                                                   //设定对象显示模式
8       var direction = o.options.direction || 'left'; //默认动画移动方向
9       //调整对象
10      $.effects.save(el, props); el.show();      //存储原对象并显示
11      $.effects.createWrapper(el);               //创建对象的封装
12      var ref = (direction == 'up' || direction == 'down') ? 'top' : 'left';
                                                   //创建动画运行参考坐标方向
13      var motion = (direction == 'up' || direction == 'left') ? 'pos' : 'neg';
                                                   //创建动画动作参数
14      var distance = o.options.distance || (ref == 'top' ? el.outerHeight
        ({margin:true}) / 2 : el.outerWidth({margin:true}) / 2);
                                                   //设定动画实际运动距离
        //转换对象初始状态
15      if (mode == 'show') el.css('opacity', 0).css(ref, motion == 'pos' ?
        -distance : distance);
16      // 使用自定义动画
```

```
17      var animation = {opacity: mode == 'show' ? 1 : 0};
                                                    //设定动画中需要改变的透明度
18      animation[ref] = (mode == 'show' ? (motion == 'pos' ? '+=' : '-=') :
        (motion == 'pos' ? '-=' : '+=')) + distance;
                                                    //设定动画运动最终目标参数值
19      //运行动画
20      el.animate(animation, { queue: false, duration: o.duration, easing:
        o.options.easing, complete: function()
21  {
22          if(mode == 'hide') el.hide();          //隐藏对象
23          $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                    //恢复对象，移除封装
24          if(o.callback) o.callback.apply(this, arguments);
25          el.dequeue();
26      }});
27  });
28 };
29 })(jQuery);
```

上述代码第 8 行设定动画移动方向向左。第 13 行设定如果动画移动方向向左，设定操作参数。第 14 行设定动画移动距离。第 15 行转换移动位移为负值，实际向右移动。第 17 行设定透明度。第 18 行设定自定义动画参数。第 20～26 行执行自定义动画并恢复元素属性，调用回调函数。效果如图 13.8 所示。

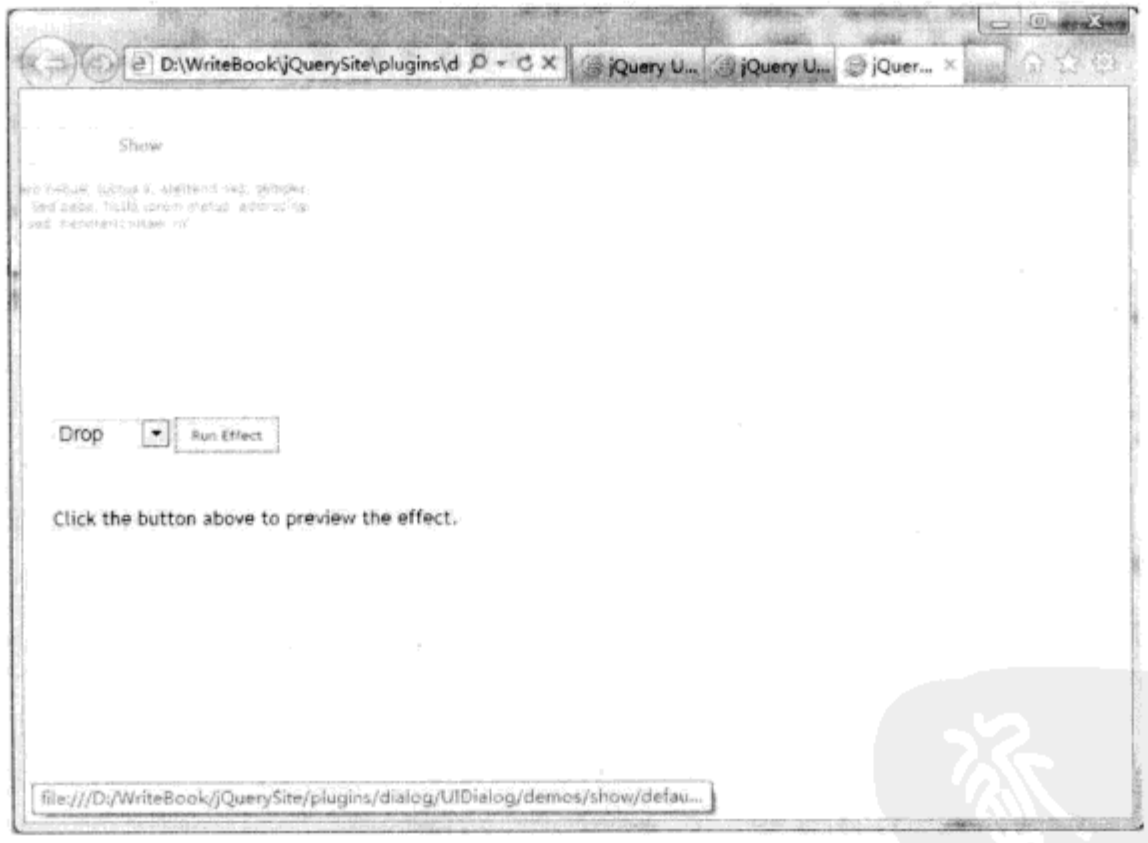


图 13.8　移动动画效果

13.2.6　分裂效果

分裂效果需要应用到动画特效的核心文件和分裂效果的插件文件 jquery.effects.explode.js。它的设计思想是复制多个元素，元素个数由动画排列的行数和列数决定，将复制出的多个元素通过自定义动画变换坐标位置由显示区域远端向中间移动完成效果。插件源代码如下：

```
(function( $, undefined ) {
$.effects.explode = function(o) {
  return this.queue(function() {
  var rows = o.options.pieces ? Math.round(Math.sqrt(o.options.pieces)) : 3;
                                                    //获取动画中行的个数
  var cells = o.options.pieces ? Math.round(Math.sqrt(o.options.pieces)) : 3;
                                                    //获取动画中列的个数
  //设定动画显示参数
  o.options.mode = o.options.mode == 'toggle' ? ($(this).is(':visible') ?
  'hide' : 'show') : o.options.mode;
  var el = $(this).show().css('visibility', 'hidden');
                                                    //获取动画作用对象并显示
  var offset = el.offset();                         //取得对象相对于浏览器的坐标位置
  //设定左上角坐标位置
  offset.top -= parseInt(el.css("marginTop"),10) || 0;
  offset.left -= parseInt(el.css("marginLeft"),10) || 0;
  var width = el.outerWidth(true);                  //元素宽度
  var height = el.outerHeight(true);                //元素高度
  for(var i=0;i<rows;i++) {                         //按照行数循环
      for(var j=0;j<cells;j++) {                    //按照列数循环
          el
              .clone()                              //克隆对象
              .appendTo('body')                     //添加到页面中
              .wrap('<div></div>')                  //为每一个复制出来的元素用层封装
              .css({
                  position: 'absolute',
                  visibility: 'visible',
                  left: -j*(width/cells),
                  top: -i*(height/rows)
              })//设定每个复制元素的位置并根据行和列位置设定高和宽
              .parent()                             //获取封装层
              .addClass('ui-effects-explode')       //在层上加入样式设定
              .css({                                //设定封装层的位置及尺寸
                  position: 'absolute',
                  overflow: 'hidden',
                  width: width/cells,
                  height: height/rows,
                  left: offset.left + j*(width/cells) + (o.options.mode
                  == 'show' ? (j-Math.floor(cells/2))*(width/cells) : 0),
                  top: offset.top + i*(height/rows) + (o.options.mode ==
                  'show' ? (i-Math.floor(rows/2))*(height/rows) : 0),
                  opacity: o.options.mode == 'show' ? 0 : 1
              }).animate({          //利用自定义动画变化封装层的位置
                  left: offset.left + j*(width/cells) + (o.options.mode
                  == 'show' ? 0 : (j-Math.floor(cells/2))*(width/cells)),
                  top: offset.top + i*(height/rows) + (o.options.mode ==
                  'show' ? 0 : (i-Math.floor(rows/2))*(height/rows)),
                  opacity: o.options.mode == 'show' ? 1 : 0
              }, o.duration || 500);
      }
  }
  //当动画完成设定定时函数
```

```
44  setTimeout(function() {
    //设定对象是否显示
45      o.options.mode == 'show' ? el.css({ visibility: 'visible' }) : el.
        css({ visibility: 'visible' }).hide();
46              if(o.callback) o.callback.apply(el[0]);
47              el.dequeue();
48              $('div.ui-effects-explode').remove();  //移除封装层
49  }, o.duration || 500);
50  });
51};
52})(jQuery);
```

上述代码第 4、5 行设定了分裂显示时的行和列的数目。第 6 行设定显示模式。第 7 行获取隐藏元素对象。第 8 行获取元素相对于浏览器窗口的坐标位置。第 10、11 行将元素的左上角坐标值分别减去边界空白宽度。第 12、13 行获取元素的外部宽和外部高，包括边框和边界空白。第 14～42 行根据行和列的个数循环复制多个消息部分，并设定每个复制出来的消息层的起始位置，然后使用自定义动画函数，将每个消息层显示在指定位置，拼凑成一个完整的消息层。第 44～49 行设定一个超时函数，实现消息层的显示状态转换，并调用回调函数，删除队列，移除 CSS 样式设定。效果如图 13.9 所示。

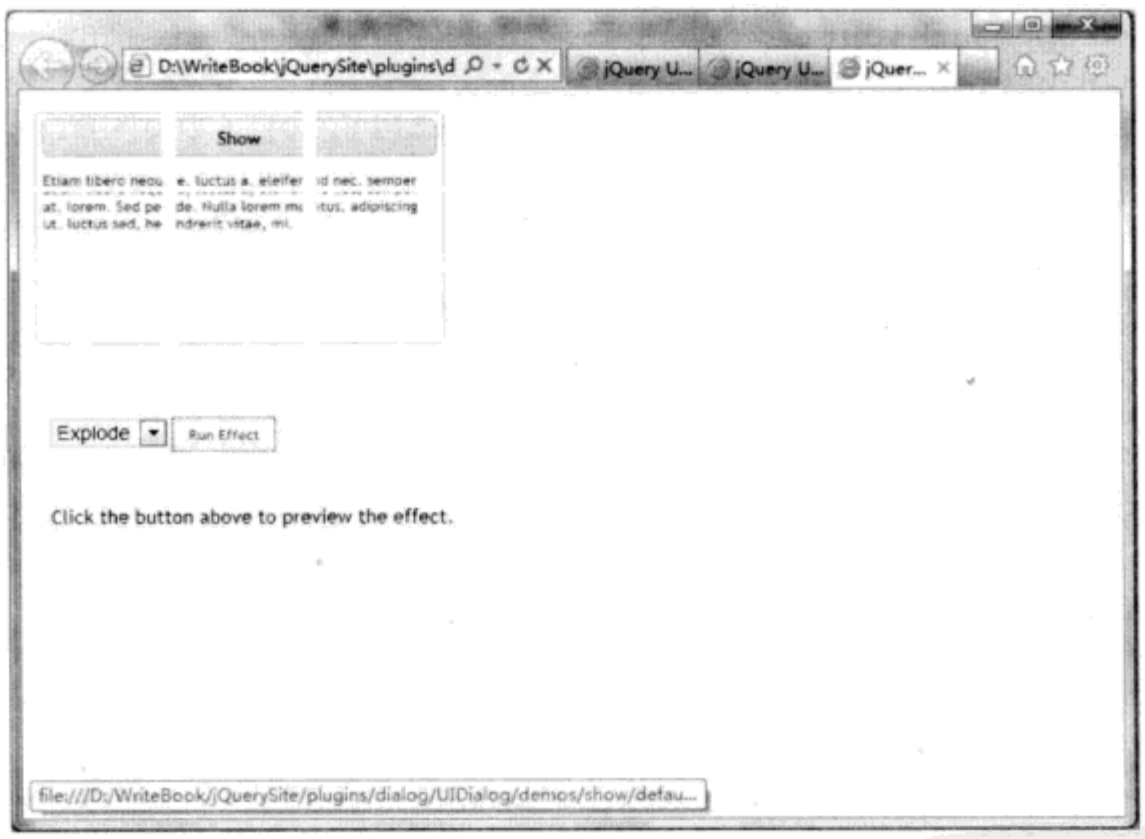


图 13.9　移动动画效果

13.2.7　折叠效果

折叠效果需要应用到动画特效的核心文件和折叠效果的插件文件 jquery.effects.fold.js。它的设计思想是按照一定的折叠方向和折叠比例，将元素用自定义动画多次修改坐标位置实现。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.fold = function(o) {
```

```
3   return this.queue(function() {
4       // 获取动画作用的对象
5       var el = $(this), props = ['position','top','bottom','left',
        'right'];
6       // 设定动画参数
7       var mode = $.effects.setMode(el, o.options.mode || 'hide');
                                                    //设定对象显示模式
8       var size = o.options.size || 15;            //设定默认折叠大小
9       var horizFirst = !(!o.options.horizFirst); //是否是起始水平折叠
10      var duration = o.duration ? o.duration / 2 : $.fx.speeds._default / 2;
                                                    //动画持续时间
11      // 调整对象
12      $.effects.save(el, props); el.show();       //存储对象并显示
13      var wrapper = $.effects.createWrapper(el).css({overflow:'hidden'});
                                                    //创建封装元素
14      var widthFirst = ((mode == 'show') != horizFirst);
                                                    //设定是否是首次修改宽度
15      var ref = widthFirst ? ['width', 'height'] : ['height', 'width'];
                                                    //设定动画操作相关参数
                                          //根据封装元素的宽或者高设定动画距离
16      var distance = widthFirst ? [wrapper.width(), wrapper.height()] :
        [wrapper.height(), wrapper.width()];
17      var percent = /([0-9]+)%/.exec(size);       //计算百分比
18      if(percent) size = parseInt(percent[1],10) / 100 * distance[mode ==
        'hide' ? 0 : 1];                            //设定折叠距离
19      if(mode == 'show') wrapper.css(horizFirst ? {height: 0, width: size} :
        {height: size, width: 0});                  //初始状态
20      // 使用动画
21      var animation1 = {}, animation2 = {};
22      animation1[ref[0]] = mode == 'show' ? distance[0] : size;
                                                    //横向展开参数
23      animation2[ref[1]] = mode == 'show' ? distance[1] : 0;
                                                    //纵向展开参数
24      // 调用自定义动画
25      wrapper.animate(animation1, duration, o.options.easing)
26      .animate(animation2, duration, o.options.easing, function() {
27          if(mode == 'hide') el.hide();           //隐藏元素
28          $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                    //恢复对象并移除封装
29          if(o.callback) o.callback.apply(el[0], arguments);
30          el.dequeue();
31      });
32  });
33};
34})(jQuery);
```

上述代码第 8 行设定默认折叠效果大小。第 9 行确认首次折叠是水平方向折叠。第 10 行设定动画持续时间。第 14、15 行确认两次折叠的先后参数。第 16 行确认两次折叠的先后参数值。第 17 行确认折叠比例百分比。第 18 行确认折叠效果大小。第 19 行转换包装后的元素样式。第 25～31 行调用自定义动画实现折叠打开效果，并删除包装元素，调用回调函数，从队列中删除。效果如图 13.10 和图 13.11 所示。

图 13.10　折叠动画效果横向打开

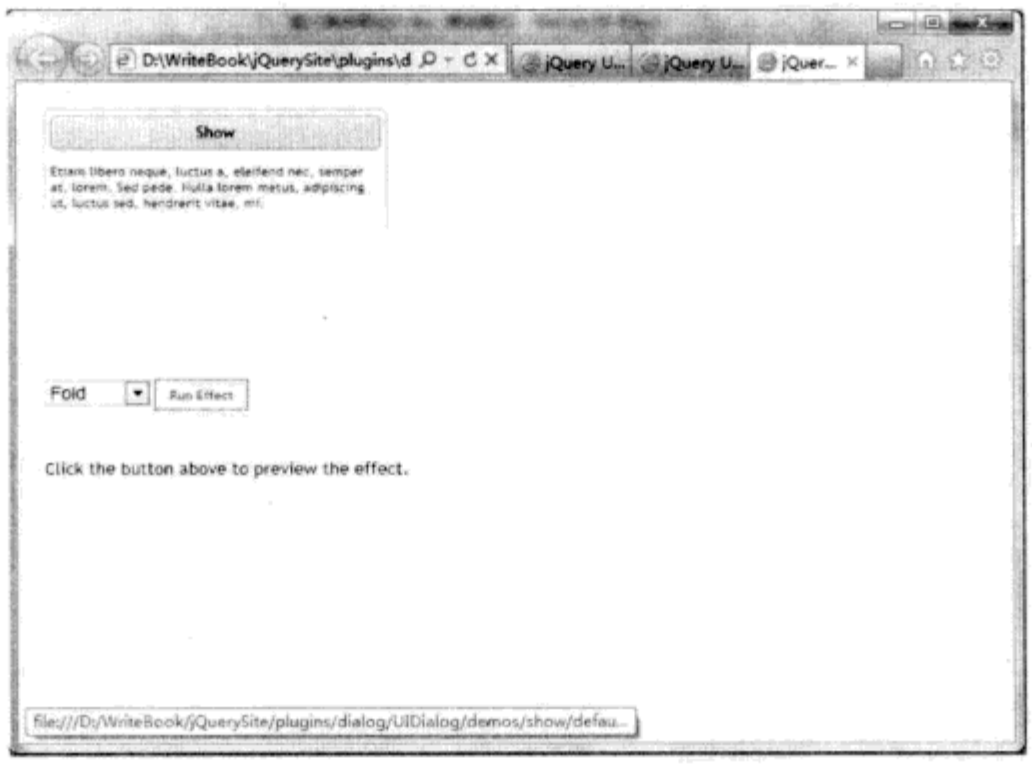


图 13.11　折叠动画效果纵向打开

13.2.8　高亮淡入淡出效果

高亮淡入淡出效果需要应用到动画特效的核心文件和高亮效果的插件文件 jquery.effects.highlight.js。它的设计思想比较简单，就是频繁更换背景色及透明度来实现效果。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.highlight = function(o) {
3   return this.queue(function() {
4       var elem = $(this),                                    //获取动画作用对象
5           props = ['backgroundImage', 'backgroundColor', 'opacity'],
6           mode = $.effects.setMode(elem, o.options.mode || 'show'),
                                                               //设置对象显示模式
7           animation = {
```

```
8                backgroundColor: elem.css('backgroundColor')
9            };                                              //动画背景色参数
10       if (mode == 'hide') {                               //动画透明度参数
11           animation.opacity = 0;
12       }
13       $.effects.save(elem, props);                        //存储对象
14       elem
15           .show()                                         //显示对象
16           .css({
17               backgroundImage: 'none',
18               backgroundColor: o.options.color || '#ffff99'
19           })                                              //去除背景图片，设定背景色
20           .animate(animation, {                           //自定义动画效果
21               queue: false,
22               duration: o.duration,
23               easing: o.options.easing,
24               complete: function() {
25                   (mode == 'hide' && elem.hide());
26                   $.effects.restore(elem, props);
27                   (mode == 'show' && !$.support.opacity && this.style.
                     removeAttribute('filter'));
28                   (o.callback && o.callback.apply(this, arguments));
29                   elem.dequeue();
30               }
31           });
32   });
33};
34})(jQuery);
```

上述代码第 7～9 行设定动画需要的背景色参数。第 18 行设定背景色。第 20～31 行调用自定义动画，更换背景色，调用回调函数，删除队列。效果如图 13.12 所示。

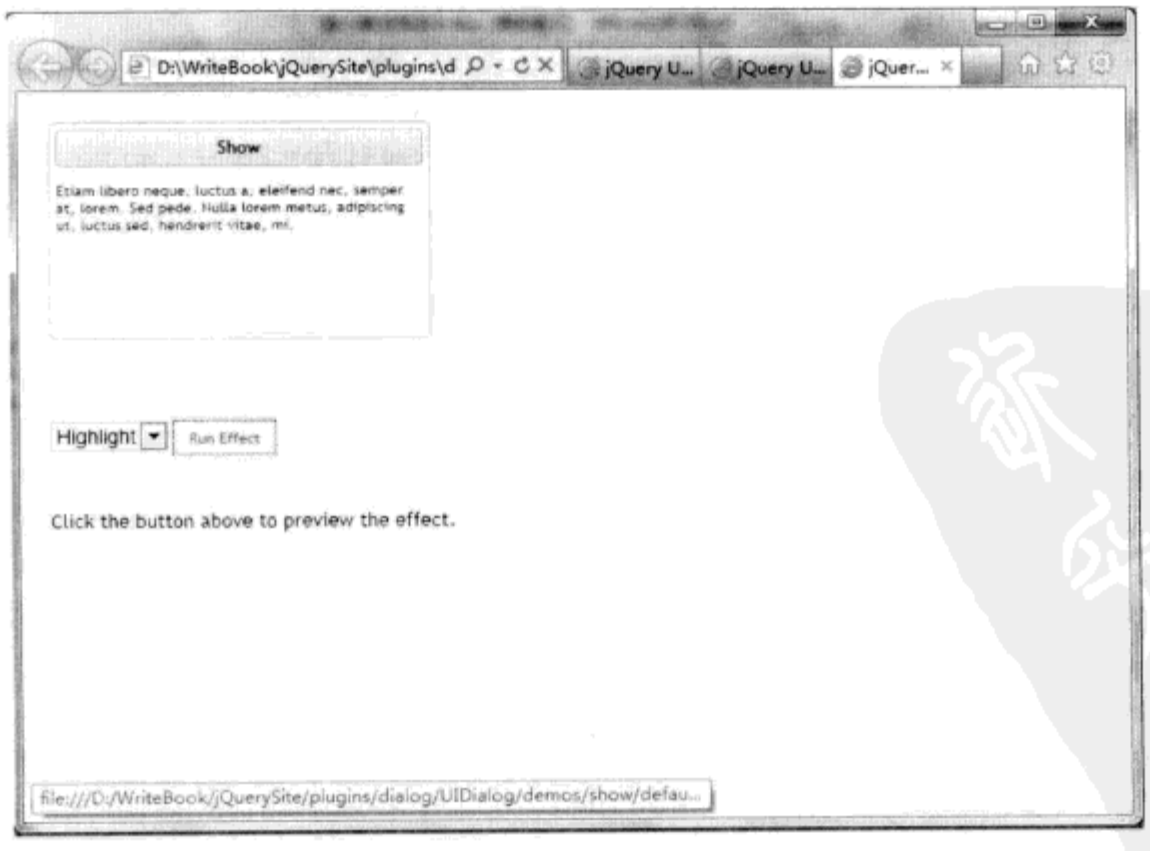


图 13.12　背景高亮动画效果

13.2.9　脉冲闪烁效果

脉冲闪烁效果需要应用到动画特效的核心文件和闪烁效果的插件文件jquery.effects.pulsate.js。它的设计思想是利用自定义动画函数频繁变化元素透明度来完成效果。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.pulsate = function(o) {
3   return this.queue(function() {
4       var elem = $(this), //获取动画作用对象
5           mode = $.effects.setMode(elem, o.options.mode || 'show');
                                                        //设置对象显示模式
6           times = ((o.options.times || 5) * 2) - 1;  //设定动画次数
7           duration = o.duration ? o.duration / 2 : $.fx.speeds._default / 2,
                                                        //设定动画持续时间
8           isVisible = elem.is(':visible'),            //判断对象的可见状态
9           animateTo = 0;                              //显示状态切换标志
10      if (!isVisible) {
11          elem.css('opacity', 0).show();              //显示对象
12          animateTo = 1;                              //显示状态
13      }
14      if ((mode == 'hide' && isVisible) || (mode == 'show' && !isVisible)) {
15          times--;                                    //动画次数自减
16      }
17      for (var i = 0; i < times; i++) {
18          elem.animate({ opacity: animateTo }, duration, o.options.easing);
                                                        //自定义动画调整对象透明度
19          animateTo = (animateTo + 1) % 2;    //更改显示状态标志
20      }
        //最后操作动画
21      elem.animate({ opacity: animateTo }, duration, o.options.easing,
        function() {
22          if (animateTo == 0) {
23              elem.hide();
24          }
25          (o.callback && o.callback.apply(this, arguments));
26      });
27      elem
28          .queue('fx', function() { elem.dequeue(); })
29          .dequeue();
30  });
31};
32})(jQuery);
```

上述代码第 6 行设定闪烁次数。第 7 行设定闪烁间隔。第 10～13 行判断元素是否可见，如果不可见则透明度为零，调用显示，设定透明度变量值为 1。第 14～16 行修改闪烁次数。第 17～20 行根据循环次数调用自定义动画函数交叉修改元素的可见性。第 21～26 行调用自定义动画函数设置透明度，并调用回调函数。效果如图 13.13 所示。

图 13.13　脉冲闪烁动画效果

13.2.10　摆动效果

摆动效果需要应用到动画特效的核心文件和摆动效果的插件文件 jquery.effects.shake.js。它的设计思想和跳跃效果基本相同，只是摆动方向为左右方向，并且在摆动过程中的第一次摆动、最后一次摆动及中间的摆动距离考虑的设定比跳跃效果复杂。插件源代码如下：

```
1 (function( $, undefined ) {
2 $.effects.shake = function(o) {
3   return this.queue(function() {
4       //获取动画作用对象
5       var el = $(this), props = ['position','top','bottom','left',
        'right'];
6       //设定动画运行参数
7       var mode = $.effects.setMode(el, o.options.mode || 'effect');
                                                    //设定效果模式
8       var direction = o.options.direction || 'left'; //设定默认移动方向
9       var distance = o.options.distance || 20;       //设定默认移动距离
10      var times = o.options.times || 3;              //设定动画次数
11      var speed = o.duration || o.options.duration || 140;
                                                    //设定默认动画速度
12      //调整对象
13      $.effects.save(el, props); el.show();          // 存储并显示对象
14      $.effects.createWrapper(el);                   // 创建封装对象
15      var ref = (direction == 'up' || direction == 'down') ? 'top' : 'left';
                                                    //创建动画作用方向参数
16      var motion = (direction == 'up' || direction == 'left') ? 'pos' : 'neg';
                                                    //设定运动方式
17      //使用自定动画
18      var animation = {}, animation1 = {}, animation2 = {};
```

```
19      animation[ref] = (motion == 'pos' ? '-=' : '+=')  + distance;
                                                    //设定最后一次左向摆动距离值
20      animation1[ref] = (motion == 'pos' ? '+=' : '-=')  + distance * 2;
                                                    //设定最后一次右向摆动距离值
21      animation2[ref] = (motion == 'pos' ? '-=' : '+=')  + distance * 2;
                                                    //设定最后一次左向摆动距离值
22      //首次调用自定义动画参数
23      el.animate(animation, speed, o.options.easing);
24      for (var i = 1; i < times; i++) {  //循环调用自定义动画，实现抖动效果
25          el.animate(animation1, speed, o.options.easing).animate
            (animation2, speed, o.options.easing);
26      };
27      el.animate(animation1, speed, o.options.easing).
28      animate(animation, speed / 2, o.options.easing, function(){
                                                    //最后一次抖动效果
29          $.effects.restore(el, props); $.effects.removeWrapper(el);
                                                    //恢复对象并移除封装
30          if(o.callback) o.callback.apply(this, arguments); // Callback
31      });
32      el.queue('fx', function() { el.dequeue(); });
33      el.dequeue();
34  });
35};
36})(jQuery);
```

上述代码第 8 行设定默认摆动方向。第 9 行设定摆动距离。第 10 行设定摆动次数。第 11 行设定动画持续时间。第 19～21 行设定三个动画摆动距离。第 23 行设定第一次摆动向右。第 24～26 行设定根据摆动次数多次摆动，但摆动距离比第一次多一倍。第 27～31 行设定最后一次摆动，并调用回调函数。效果如图 13.14 和图 13.15 所示。

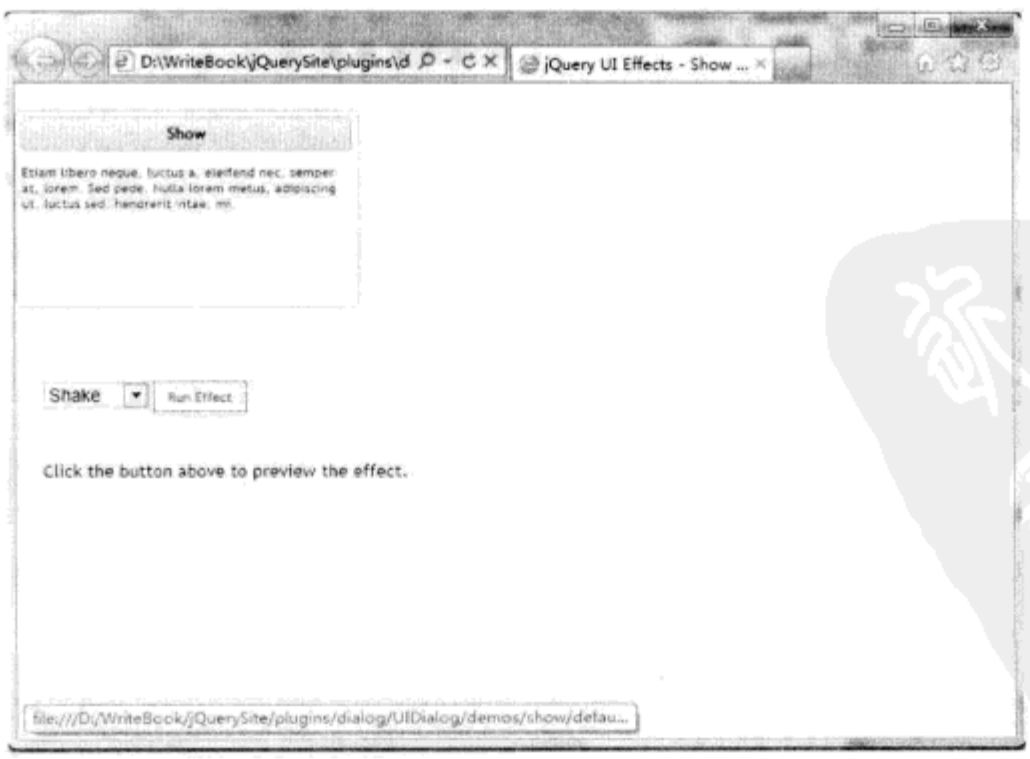


图 13.14　摆动动画效果一

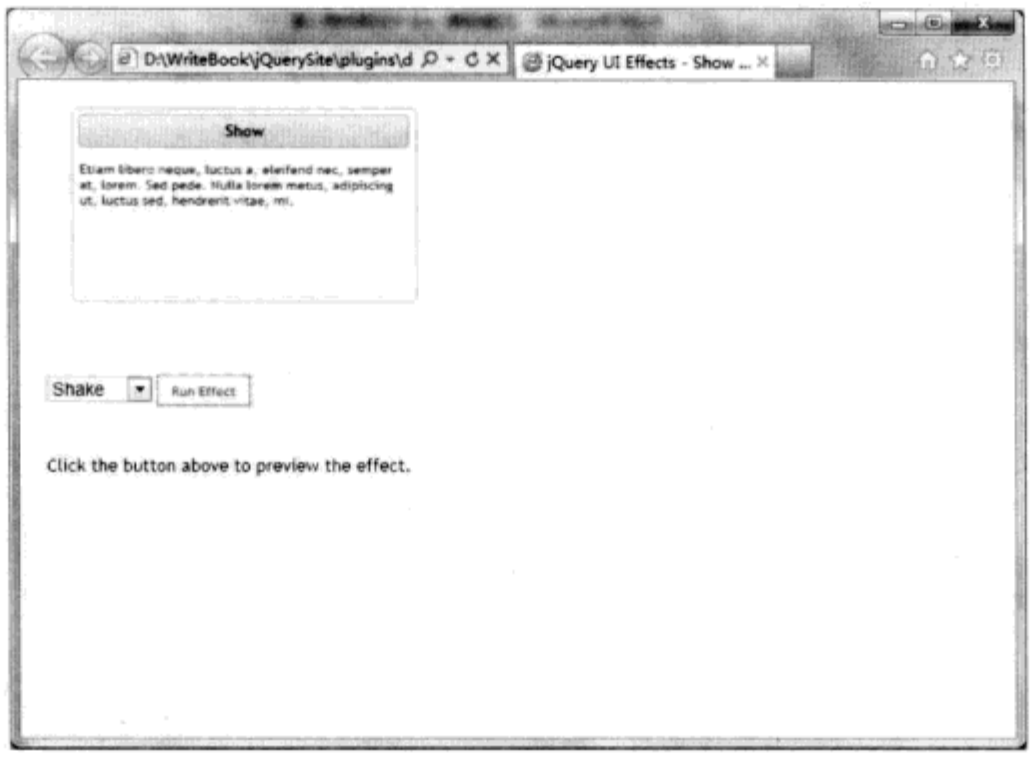


图 13.15　摆动动画效果二

13.3　小　　结

jQuery 动画效果能够给用户更愉悦的体验，也能让用户看到更生动的页面。本章主要介绍了 jQuery 动画函数，如何利用动画函数产生动画效果，以及 jQuery UI 提供的动画效果。重点内容是利用动画函数产生动画效果，jQuery UI 提供的动画效果。利用动画函数产生动画效果是本章的难点部分。下一章将介绍 jQuery 拖放组件。

13.4　习　　题

【习题 1】利用本章所学内容自定义图片的横向抖动效果。

【习题 2】利用图片练习 jQuery UI 的动画效果。

第 14 章　拖 动 插 件

现在很多网页都具有页面元素，如图片、文字段落、表格等，可以通过鼠标拖动变换位置，如博客。这种拖动效果可以通过 jQuery 的拖动插件来实现。这种拖动功能方便用户重新安排页面布局，便于用户指定符合自己风格的页面格式。本章主要讲解 jQuery 中的拖动插件。

14.1　jQuery UI 拖动插件

本节介绍 jQuery UI 工具集中的拖动插件 draggable。这个插件可以通过鼠标拖动页面元素至你想要的位置。它需要 UI Core、UI Widget、UI Mouse 库文件的支持。

14.1.1　jQuery UI 拖动插件基本介绍

jQuery UI 拖动插件的基本属性如表 14.1 所示。

表 14.1　拖动插件的属性说明

属　性	类　型	默 认 值	说　明
disable	布尔	false	拖动插件是否可用
addClasses	布尔	true	是否添加新的拖动元素样式
appendTo	元素或者选择器	'parent'	添加一个元素到插件容器中
axis	字符串	false	拖动方向轴，水平和垂直两个方向
cancel	选择器	':input,option'	取消一个指定的元素拖动
connectToSortable	选择器	false	允许将一个元素拖动到可排序表格中
containment	选择器，元素，字符串，数组	false	拖动行为的页面范围，可在页面的有边界区间内拖动
cursor	字符串	'auto'	拖动时鼠标样式
cursorAt	对象	false	设定拖动元素与鼠标的相对位移
delay	整型	0	拖动动作的延迟时间，从鼠标按下开始计算
distance	整型	1	鼠标按下并移动有效距离后才开始拖动，单位为像素
grid	数组	false	捕捉拖动元素到网格
handle	元素，选择器	false	单击元素限制拖动开始
helper	字符串，函数	'original'	允许一个操作元素进行拖动显示
iFrameFix	布尔，选择器	false	在拖动过程中阻止 iFrame 捕捉鼠标移动
opacity	浮点	false	当被拖动时是否透明

续表

属　性	类　型	默 认 值	说　明
refreshPosition	布尔	false	在鼠标移动过程中即时刷新所有拖动元素的位置
revert	布尔，字符串	false	当拖动结束后元素是否返回起始位置
revertDuration	整型	500	返回起始位置的动画持续时间，单位为毫秒
scope	字符串	'default'	可拖动元素分组范围
scroll	布尔	true	当拖动时容器是否自动滚动
scrollSentivity	整型	20	可见部分的滚动距离敏感值
scrollSpeed	整型	20	窗口滚动速度，鼠标在滚动距离敏感值内
snap	布尔，选择器	false	捕捉移动边界的元素
snapMode	字符串	'both'	确定哪个边界被捕捉
snapTolerance	整型	20	捕捉到边界元素的距离
stack	选择器	false	控制层上下关系的集合
zIndex	整型	false	当元素被拖动时层的上下关系

它还具有一些拖动事件和方法，如表 14.2 和表 14.3 所示。

表 14.2　拖动插件的事件说明

事　件	类　型	说　明
create	dragcreate	拖动插件创建事件
start	dragstart	拖动行为开始事件
drag	drag	在整个拖动过程中的事件
stop	dragstop	拖动停止事件

表 14.3　拖动插件的方法说明

方　法	使 用 方 式	说　明
destroy	.draggable("destroy")	完全移除插件功能
disable	.draggable("disable")	禁用插件
enable	.draggable("enable")	启用插件
option	.draggable("option" , optionName , [value])	获取或设置插件属性
option	.draggable("option" , options)	一次设定多个插件属性
widget	.draggable("widget")	返回插件样式元素

14.1.2　jQuery UI 拖动插件示例

1．插件的基本使用

这个示例使用了插件的属性默认值形式，具有基本的拖动功能，在插件初始化时没有为其添加任何附加特性。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable();
4   });
5   </script>
```

上述代码第 3 行使用插件初始化函数创建插件，具体效果如图 14.1 和图 14.2 所示。

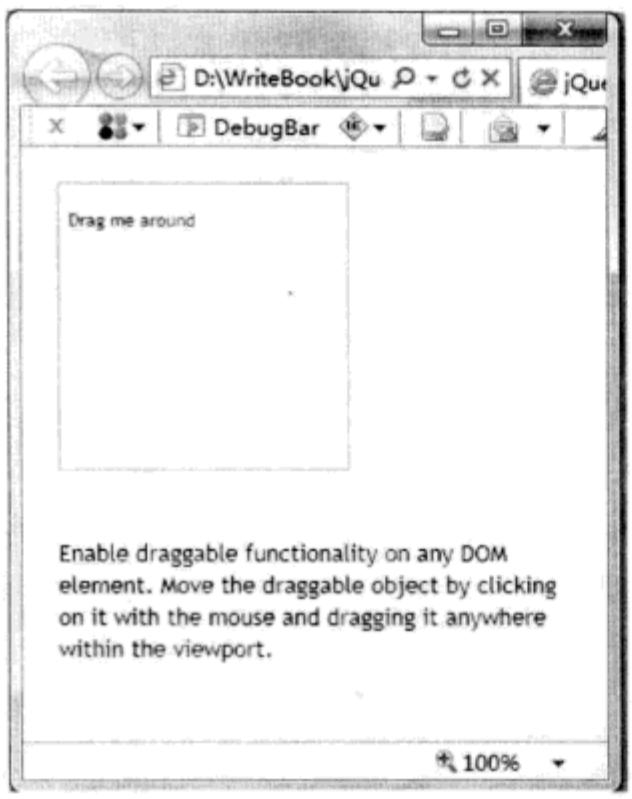


图 14.1　基本拖动应用

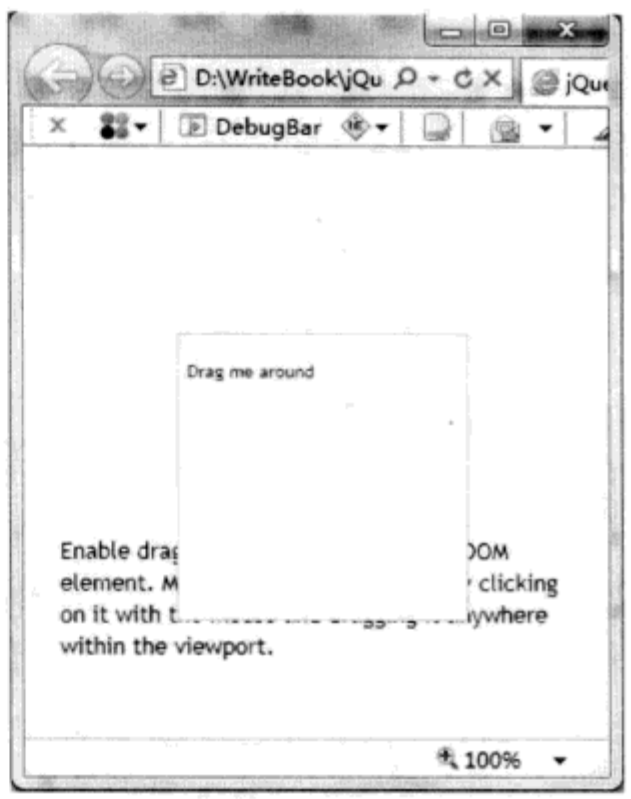


图 14.2　基本拖动应用拖动后效果

2. 拖动事件的处理

在这个示例中响应了插件的开始拖动、拖动过程、拖动结束三个事件，并为事件的发生次数计数显示在拖动元素上。使用了插件的 start、drag、stop 来表示插件开始拖动、拖动中、拖动结束三个事件。并利用自定义函数 updateCounterStatus（$event_counter, new_count）记录不同事件发生的次数。

```
1   <script>
2   $(function() {
3       var $start_counter = $( "#event-start" ),
4           $drag_counter = $( "#event-drag" ),
5           $stop_counter = $( "#event-stop" ),
6           counts = [ 0, 0, 0 ];
7       $( "#draggable" ).draggable({
8           start: function() {          //拖动行为开始事件
9               counts[ 0 ]++;
10              updateCounterStatus( $start_counter, counts[ 0 ] );
11          },
12          drag: function() {           //拖动事件
13              counts[ 1 ]++;
14              updateCounterStatus( $drag_counter, counts[ 1 ] );
15          },
16          stop: function() {           //拖动停止事件
17              counts[ 2 ]++;
18              updateCounterStatus( $stop_counter, counts[ 2 ] );
19          }
20      });
21      function updateCounterStatus( $event_counter, new_count ) {
22          //第一次更新事件计数操作
```

```
23          if ( !$event_counter.hasClass( "ui-state-hover" ) ) {
24              $event_counter.addClass( "ui-state-hover" )
25                  .siblings().removeClass( "ui-state-hover" );
26          }
27          //后续事件计数操作
28          $( "span.count", $event_counter ).text( new_count );
29      }
30  });
31  </script>
```

上述代码第 3～5 行分别获取起始拖动计数、拖动计数、结束拖动计数的显示对象。第 6 行建立一个数组来记录各个事件发生的次数；第 21～29 行根据传入的不同计数显示对象和计数值更新显示。第 8～11 行响应开始拖动事件，数组元素自增并调用更新显示函数。第 12～15 行响应拖动事件，数组元素自增并调用更新显示函数。第 16～20 行响应结束拖动事件，数组元素自增并调用更新显示函数。效果如图 14.3 和图 14.4 所示。

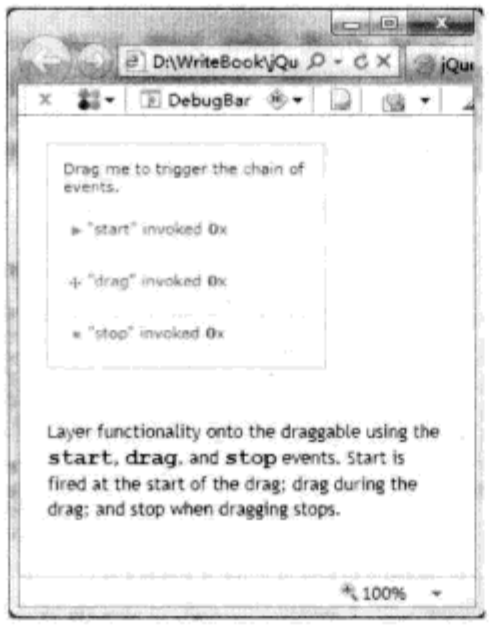


图 14.3　拖动事件处理

图 14.4　拖动动作发生后效果

3. 容器内拖动

这个示例使用了插件的拖动方向参数及拖动容器参数，规定拖动元素只可在某一方向上拖动，或者只可在某一容器范围内拖动。这里使用到了前面介绍的 axis 坐标轴属性及 containment 容器属性，限制了拖动方向及拖动范围。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ axis: "y" });     //纵向拖动限制
4       $( "#draggable2" ).draggable({ axis: "x" });    //横向拖动限制
5       //限定拖动范围
6       $( "#draggable3" ).draggable({ containment: "#containment-wrapper",
        scroll: false });
7       $( "#draggable4" ).draggable({ containment: "#demo-frame" });
8       $( "#draggable5" ).draggable({ containment: "parent" });
9   });
10  </script>
```

上述代码第 3 行规定拖动元素只可在纵向拖动。第 4 行规定拖动元素只可在横向拖动。第 6 行规定拖动元素只可在指定容器内拖动，并且不可使用滚动条。第 7 行规定拖动元素只可在指定容器内拖动，但页面上并无此容器，使用滚动条。第 8 行规定拖动元素只可在父元素中拖动。效果如图 14.5 和图 14.6 所示。

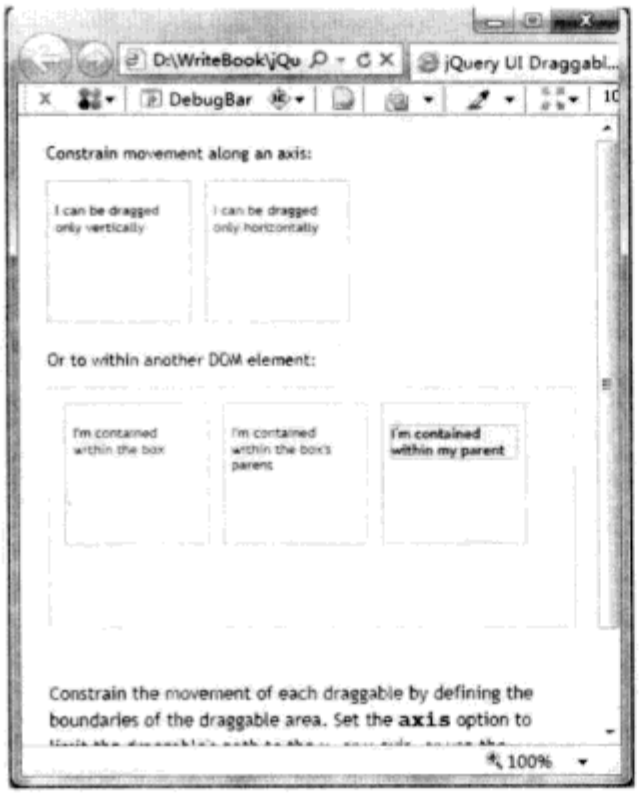


图 14.5　容器内拖动

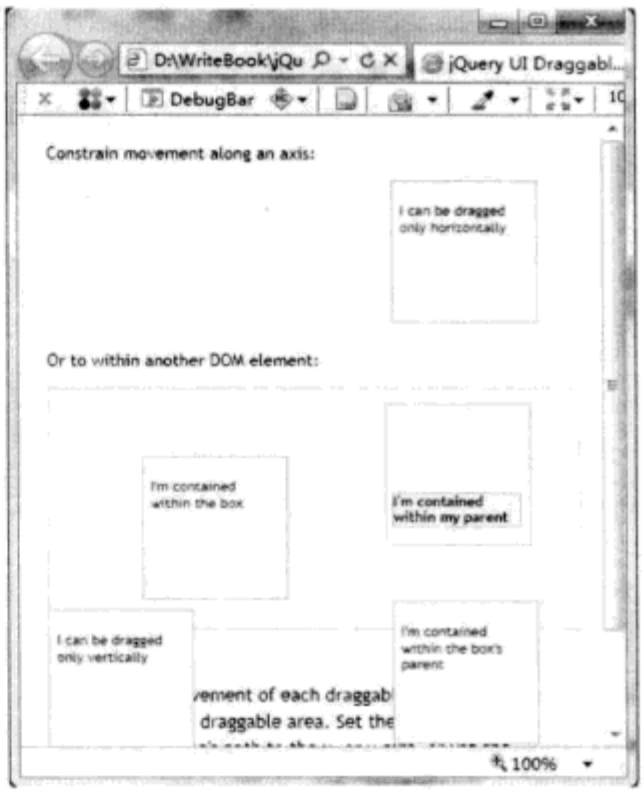


图 14.6　容器内拖动后效果

4. 延迟拖动

这个示例使用了插件在距离和时间上的延迟拖动效果，避免出现用户频繁单击出现的误操作效果。这里使用了前面介绍的 distance 延迟移动距离和 delay 延迟毫秒数属性。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ distance: 20 });
                                                    //设定拖动时鼠标反应距离
4       $( "#draggable2" ).draggable({ delay: 1000 }); //设定拖动延迟时间
5       $( ".ui-draggable" ).disableSelection();
6   });
7   </script>
```

上述代码第 3 行设定当鼠标按下并移动了 20 像素后再发生拖动操作。第 4 行设定当鼠标按下并移动 1 秒钟后再发生拖动操作。第 5 行禁止样式选定。效果如图 14.7 和图 14.8 所示。

5. 捕获至元素或者网格

这个示例使用了拖动插件的捕获属性。这里使用了前面介绍的 snap 捕获属性和 grid 网格属性。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ snap: true });   //拖动元素捕获设定
4       $( "#draggable2" ).draggable({ snap: ".ui-widget-header" });
```

```
                                                    //设定捕获拖动元素对象
5       $( "#draggable3" ).draggable({ snap: ".ui-widget-header", snapMode:
        "outer" });                                 //设定捕获拖动元素对象及捕获方式
6       $( "#draggable4" ).draggable({ grid: [ 20,20 ] });
7       $( "#draggable5" ).draggable({ grid: [ 80, 80 ] });
8   });
9   </script>
```

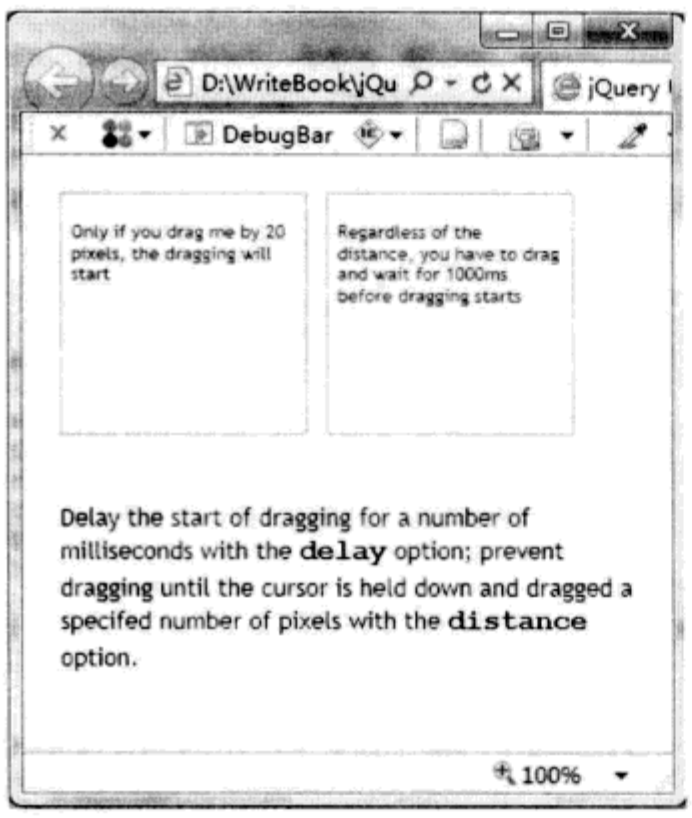


图 14.7　延迟拖动

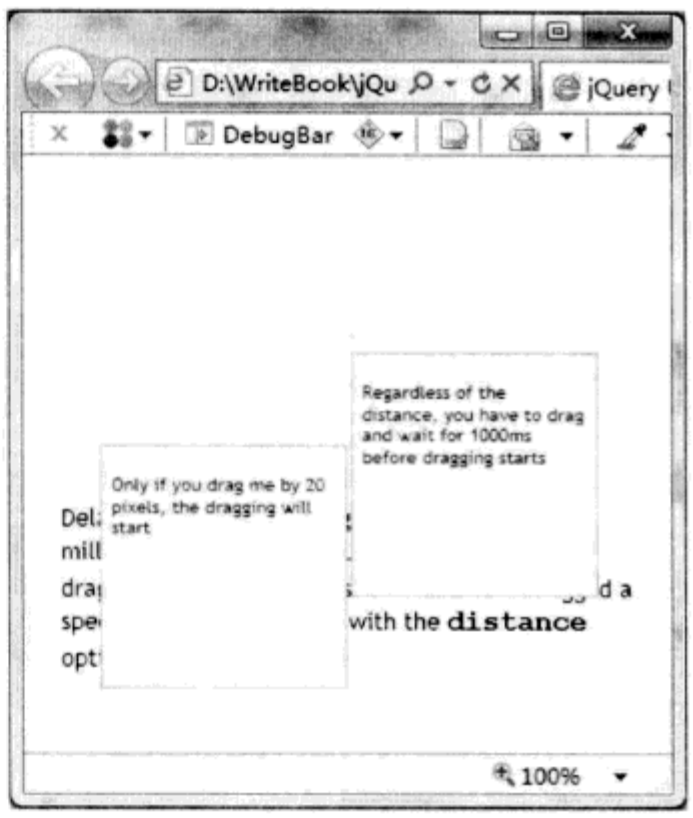


图 14.8　延迟拖动后结果

上述代码第 3 行简单设置了拖动元素可被捕获。第 4 行指定了被拖动元素被哪个元素捕获。第 5 行也是指定了捕获元素及捕获方式。第 6、7 行指定了拖动元素被捕获的网格位置。效果如图 14.9 和图 14.10 所示。

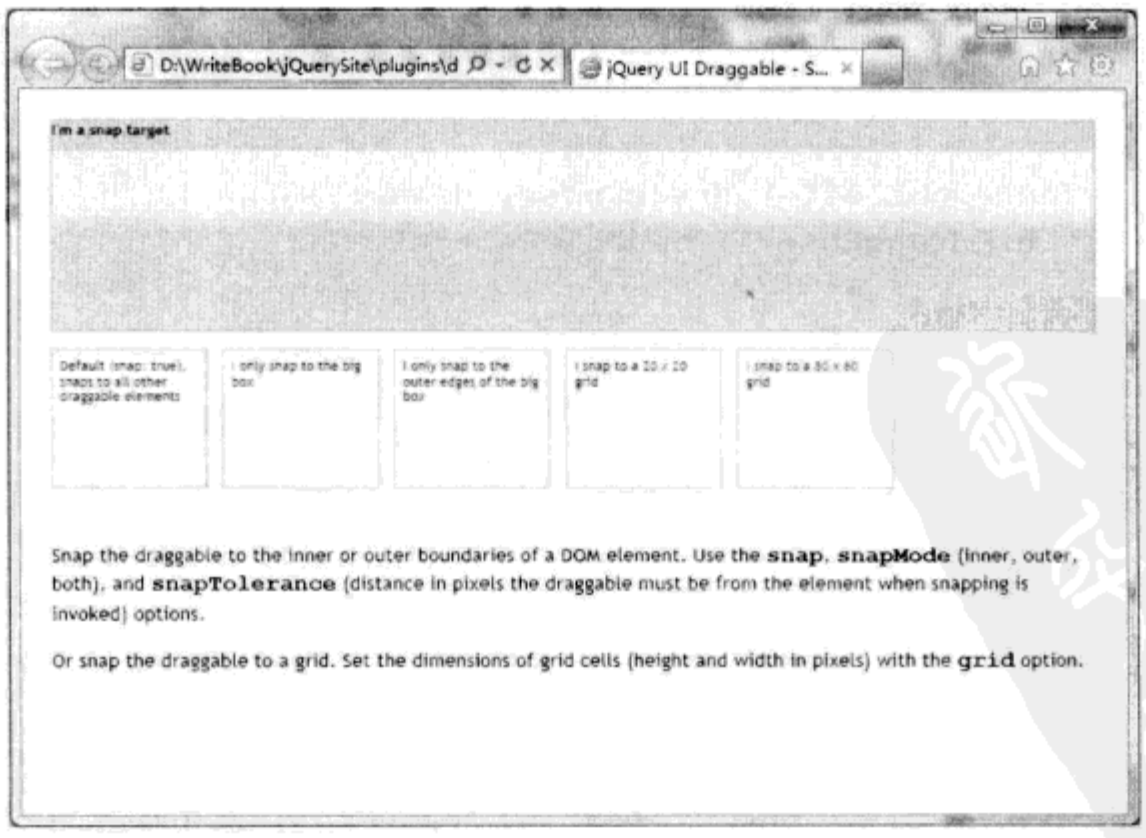


图 14.9　捕获至元素或网格

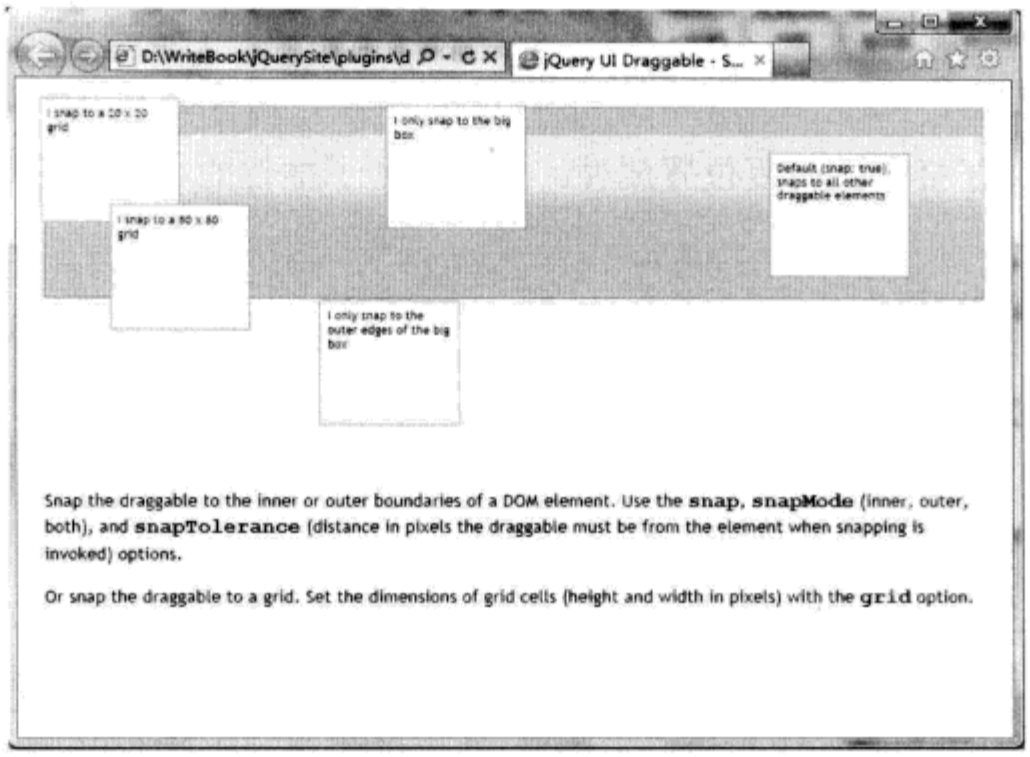


图 14.10　被元素或网格捕获效果

6. 拖动过程中窗口自动滚动

这个示例使用了与滚动条相关的三个属性：scroll，是否添加滚动条，scrollSensitiyity，滚动条敏感值，scrollSpeed，滚动条的滚动速度。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ scroll: true });
                                                    //拖动中父容器可自动产生滚动条
4       $( "#draggable2" ).draggable({ scroll: true, scrollSensitivity:
        100 });                                     //滚动条的敏感值
5       $( "#draggable3" ).draggable({ scroll: true, scrollSpeed: 100 });
                                                    //滚动条的滚动速度
6   });
7   </script>
```

上述代码第 3 行设定了窗口可随拖动操作自动滚动。第 4 行设定了窗口自动滚动的敏感距离。第 5 行设定了自动滚动速度。效果如图 14.11 和图 14.12 所示。

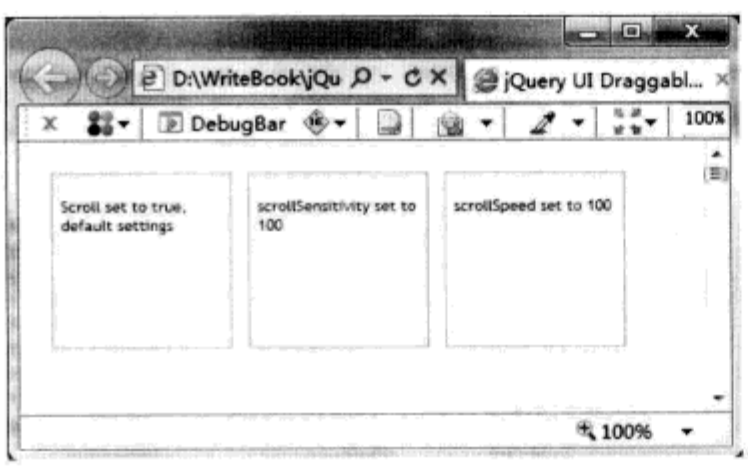


图 14.11　自动滚动窗口

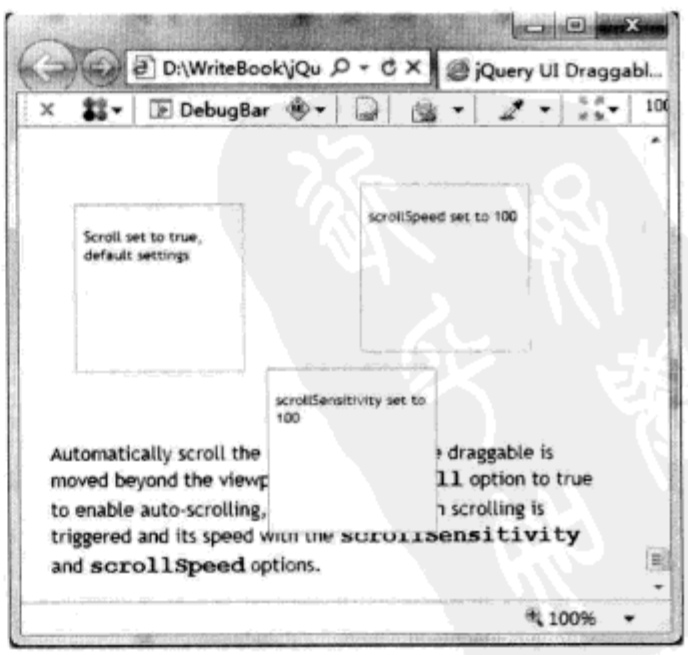


图 14.12　拖动引发窗口自动滚动

7. 恢复原始位置

这个示例使用了插件的恢复位置属性。这里使用了插件的 revert 恢复原始位置属性。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ revert: true }); //拖动后返回原始位置
4       $( "#draggable2" ).draggable({ revert: true, helper: "clone" });
                                                    //拖动后产生克隆元素
5   });
6   </script>
```

上述代码第 3 行简单设置了拖动元素的恢复原始位置属性。第 4 行设定了拖动元素在被拖动时，插件工具自动在原位置创建的克隆体，并恢复原始位置。这个例子的效果是，拖动元素还可拖动，但当我们将鼠标键抬起时，元素又恢复到了初始位置。效果如图 14.13～图 14.15 所示。

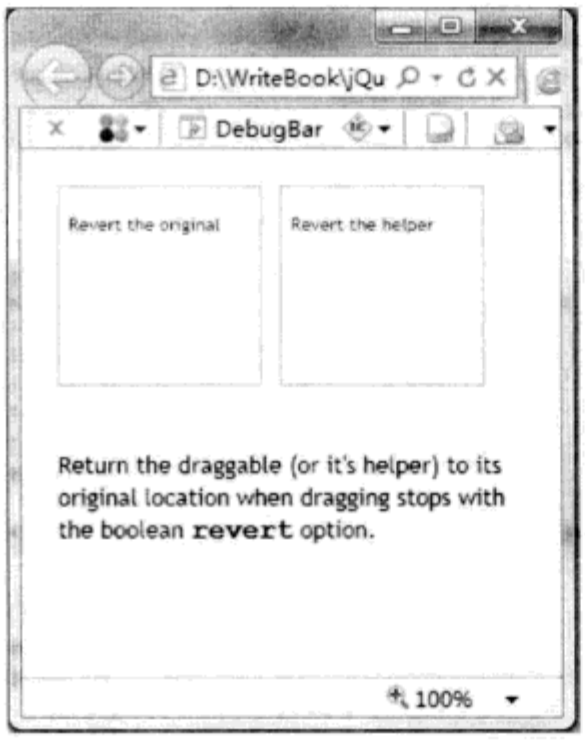


图 14.13　恢复原始位置

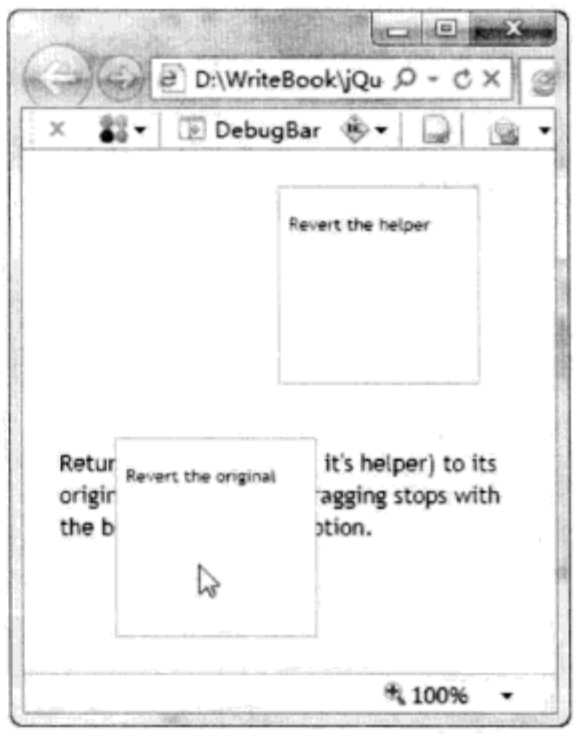


图 14.14　恢复原始位置拖动过程一

8. 可视化反馈

这个示例使用了插件的工具设定，使用插件工具在拖动时实现了不同效果，另外还使用了拖动元素的堆栈。这里使用了插件的 opacity 透明度属性，以及当拖动过程中设置鼠标样式的属性 cursor 和鼠标相对于拖动元素的位移 cursorAt 属性。使用 stack 堆栈属性设定了在同一个栈中的不同拖动元素当被拖动产生覆盖效果时，最后一次拖动的元素在其他元素的上层，并覆盖了其他被拖动的元素。

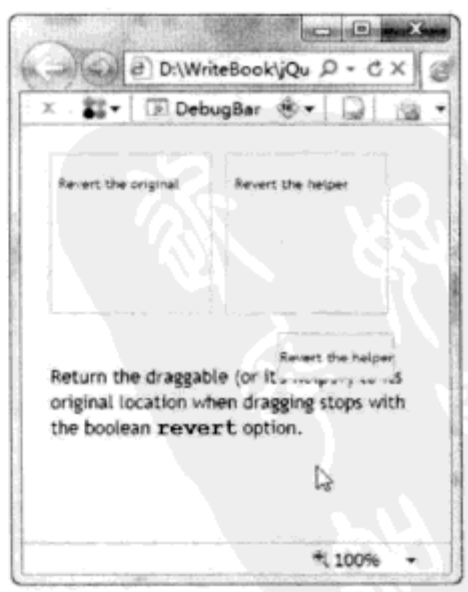


图 14.15　恢复原始位置拖动过程二

```
1   <script>
2   $(function() {
```

```
3       $( "#draggable" ).draggable({ helper: "original" });
4       $( "#draggable2" ).draggable({ opacity: 0.7, helper: "clone" });
5       $( "#draggable3" ).draggable({
6           cursor: "move",                          //鼠标样式
7           cursorAt: { top: -12, left: -20 },       //鼠标位置
8           helper: function( event ) {              //对拖动元素产生新内容
9               return $( "<div class='ui-widget-header'>I'm a custom helper
                </div>" );
10          }
11      });
12      $( "#set div" ).draggable({ stack: "#set div" });
13  });
14  </script>
```

上述代码第 3 行使用插件原始工具功能。第 4 行设定当拖动元素时，元素透明度改变，并使用克隆工具克隆元素在原有位置。第 6 行设定拖动时光标样式。第 7 行设定拖动时光标位置。第 8、9 行重写了工具的实现功能，用一个新添加的层表示拖动效果。第 12 行使用拖动元素堆栈，效果如图 14.16～图 14.18 所示。

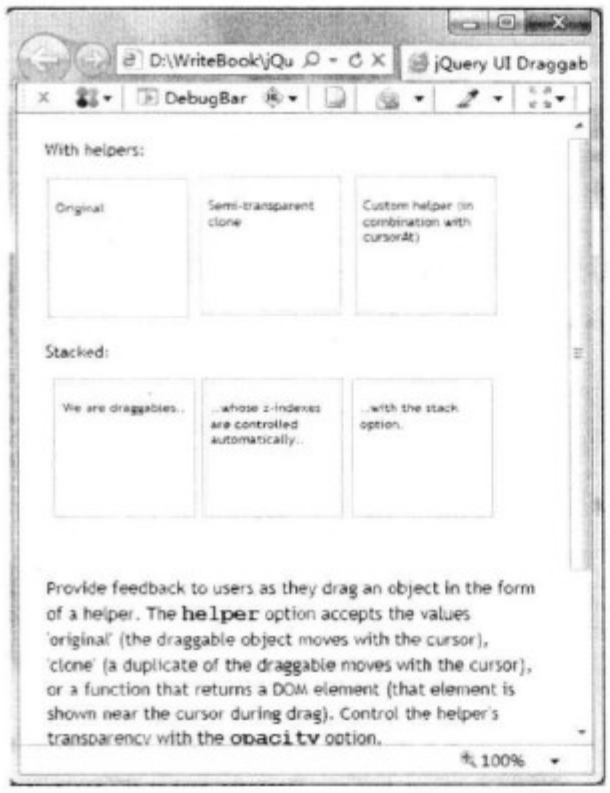


图 14.16　可视化反馈

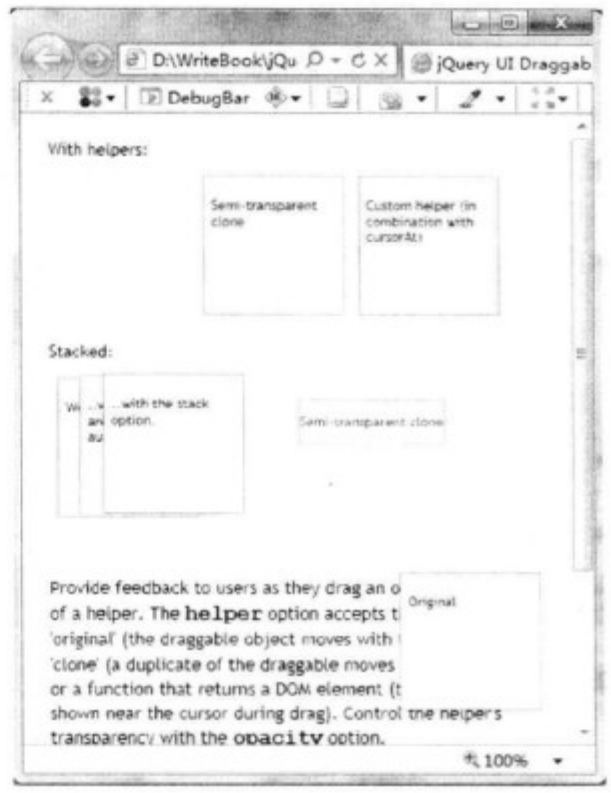


图 14.17　可视化反馈透明度 0.7 效果

9. 光标样式

这个示例使用了插件对光标设置的相关属性。这里使用了和上一个例子相同的关于光标样式设定的两个属性 cursor 和 cursorAt。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable({ cursorAt: { cursor: "move", top: 56,
        left: 56 } });
4       $( "#draggable2" ).draggable({ cursorAt: { cursor: "crosshair", top:
        -5, left: -5 } });
5       $( "#draggable3" ).draggable({ cursorAt: { bottom: 0 } });
```

```
6    });
7    </script>
```

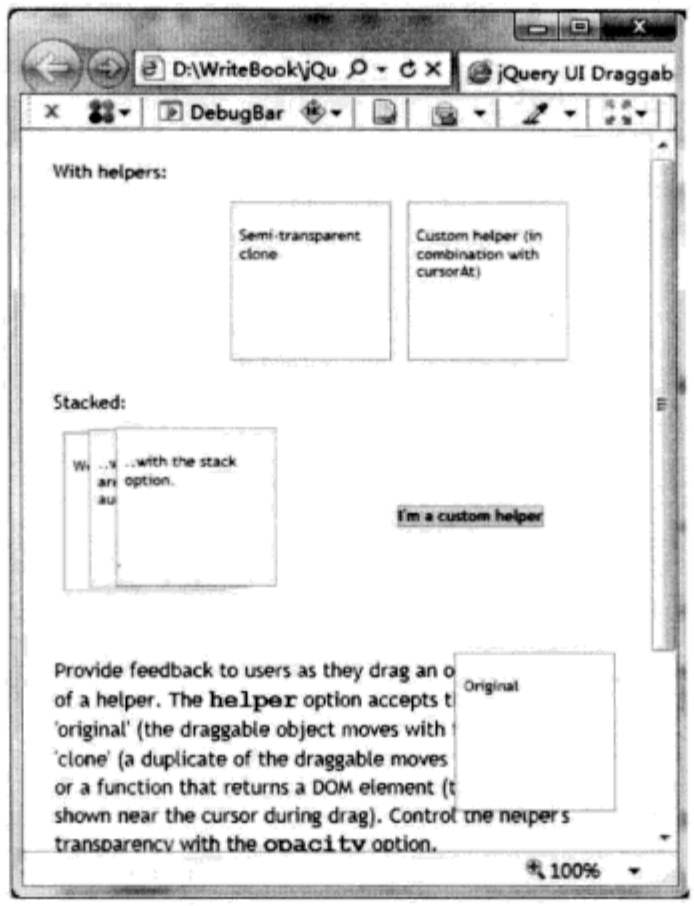


图 14.18　可视化反馈重写工具实现

上述代码第 3 行设定了光标位置，光标样式为移动样式。第 4 行设定了光标位置，光标样式为十字样式。第 5 行设定了光标位置在拖动元素底端。效果如图 14.19～图 14.21 所示。

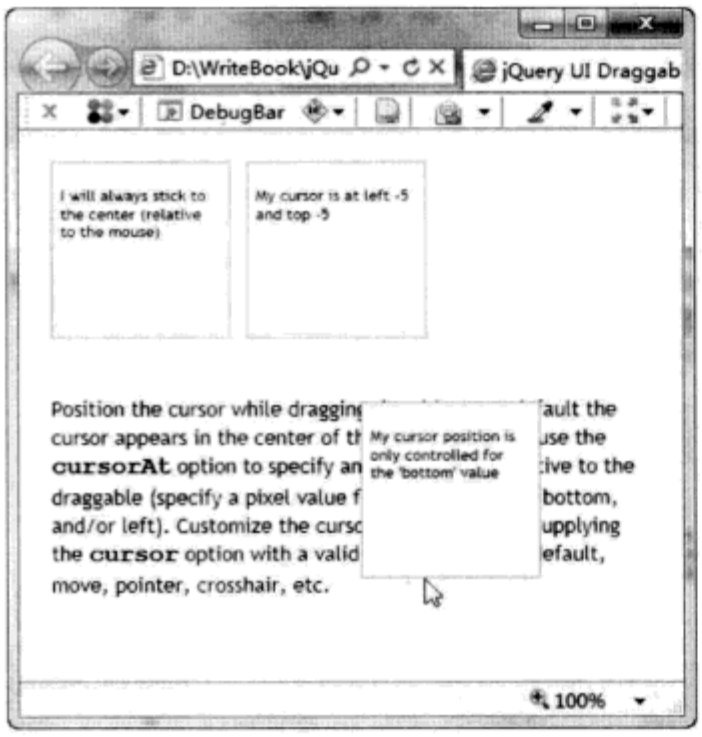


图 14.19　光标样式一

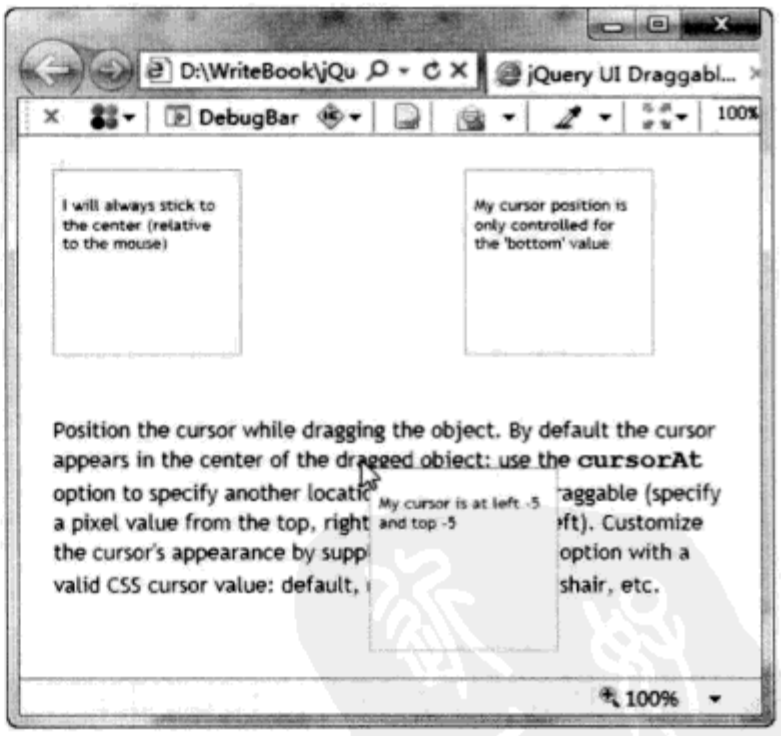


图 14.20　光标样式二

10. 拖入有序表格

这个示例使用了排序列表插件，将拖动元素拖入排序列表中。这里使用了可排序列表插件 sortable，并设定了表中元素定位属性 revert。在拖动插件中使用了 connectToSortable 属性与排序表格关联。

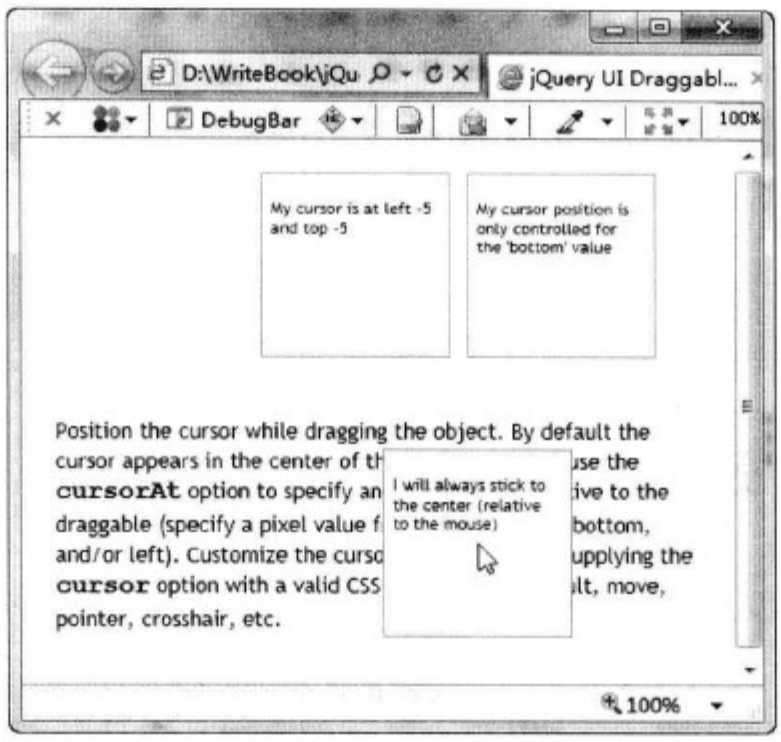


图 14.21　光标样式三

```
1   <script>
2   $(function() {
3       $( "#sortable" ).sortable({
4           revert: true
5       });
6       $( "#draggable" ).draggable({
7           connectToSortable: "#sortable",    //将拖动元素与表格关联
8           helper: "clone",
9           revert: "invalid"
10      });
11      $( "ul, li" ).disableSelection();
12  });
13  </script>
```

上述代码第 4 行设定排序表格行顺序发生变化时元素定位在新的位置。第 7 行将拖动插件与排序表格关联起来。第 8 行设定当拖动元素时克隆一个新的元素。第 9 行不允许元素恢复原始位置。效果如图 14.22 和图 14.23 所示。

图 14.22　拖入有序表格一

图 14.23　拖入有序表格二

14.2　jQuery UI 投放插件

下面介绍 jQuery UI 提供的另一种投放插件 dropable。这种插件可以与上一节讲的拖动插件搭配使用，不同的是这种插件可以接受拖动插件元素。

14.2.1　jQuery UI 投放插件基本介绍

下面来看一下 jQuery UI 投放插件的基本属性，如表 14.4 所示。

表 14.4　投放插件属性说明

属　性	类　型	默 认 值	说　明
disable	布尔	false	是否使用投放插件
accept	选择器，函数	'*'	所有的匹配选择器的拖动对象都会被接受
activeClass	字符串	false	投放元素的激活样式
addClasses	布尔	true	是否添加 ui-droppable 样式
hoverClass	字符串	false	当拖动元素悬停在投放元素上时的样式类
greedy	布尔	false	是否阻止嵌套的投放元素事件传递到上层元素
scope	字符串	'default'	投放元素分组范围
tolerance	字符串	'intersect'	指定当拖动元素覆盖在投放元素上时，投放元素的样式

该插件还具有一些投放事件和方法，如表 14.5 和表 14.6 所示。

表 14.5　投放插件事件说明

事　件	类　型	说　明
create	dropcreate	投放控件创建事件
activate	dropactivate	投放元素激活事件
deactivate	dropdeactivate	投放元素停止移动事件
over	dropover	拖动元素悬停在投放元素上事件
out	dropout	拖动元素离开投放元素事件
drop	drop	拖动元素被拖动到投放元素上事件

表 14.6　投放插件方法说明

方　法	使 用 方 式	说　明
destroy()	.droppable("destroy")	移除插件所有功能
disable()	.droppable("disable")	禁用插件
enable()	.droppable("enable")	启用插件
option()	.droppable("option" , optionName , [value])	设定插件选项
widget()	.droppable("widget")	获取插件组件

14.2.2　jQuery UI 投放插件示例

1．默认功能

这个示例使用了投放的默认属性功能。这里还使用了插件的投放事件 drop，并在投放

事件中修改投放插件的样式及文字内容。

```
1   <script>
2   $(function() {
3       $( "#draggable" ).draggable();
4       $( "#droppable" ).droppable({
5           drop: function( event, ui ) {        //投放事件
6               $( this )
7                   .addClass( "ui-state-highlight" )
8                   .find( "p" )
9                       .html( "Dropped!" );
10          }
11      });
12  });
13  </script>
```

上述代码第 3 行初始化一个拖动插件。第 4 行初始化一个投放插件。第 5～10 行响应投放插件的投放事件，在事件中修改样式及文本。效果如图 14.24 和图 14.25 所示。

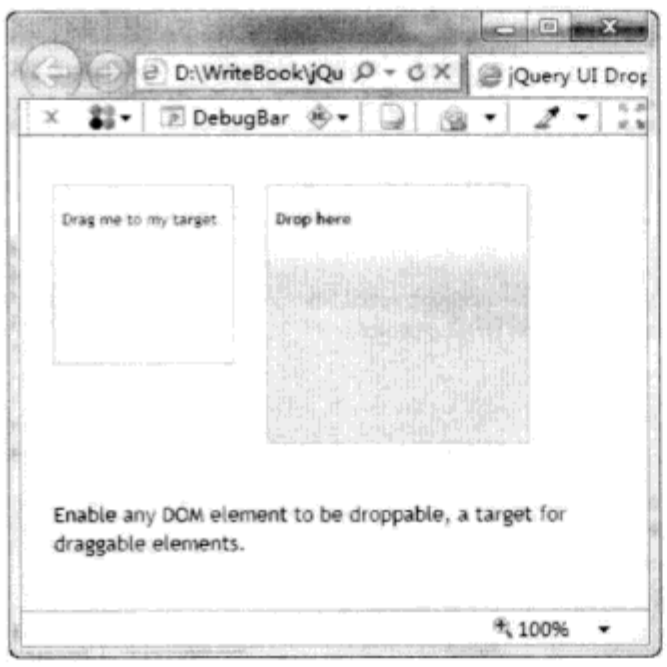


图 14.24　基本投放插件使用

图 14.25　投放插件投放效果

2. 接受拖动元素测试

这个示例中使用了两个拖动元素、一个投放元素。在投放元素中只接受其中一个拖动元素。这里使用了投放元素的 accept 属性指定可接受的拖动元素对象，并使用了 activeClass 属性设定当投放元素被激活时的样式。使用 hoverClass 属性设定当有元素覆盖到投放元素上时的样式，同时也使用了投放事件 drop 修改投放元素样式及文本。

```
1   <script>
2   $(function() {
3       $( "#draggable, #draggable-nonvalid" ).draggable();
4       $( "#droppable" ).droppable({
5           accept: "#draggable",                    //接受元素指定
6           activeClass: "ui-state-hover",           //被激活时样式
7           hoverClass: "ui-state-active",           //有其他元素悬停其上时的样式
8           drop: function( event, ui ) {
```

```
9                $( this )
10                   .addClass( "ui-state-highlight" )
11                   .find( "p" )
12                       .html( "Dropped!" );
13           }
14       });
15   });
16   </script>
```

上述代码第 3 行创建了两个拖放元素。第 5 行设定了投放元素可以接受的拖动元素。第 6 行设定投放元素激活样式。第 7 行设定当可接受的拖动元素悬停在投放元素上时的样式。第 8～13 行响应投放插件的投放事件，在事件中修改样式及文本。效果如图 14.26～图 14.28 所示。

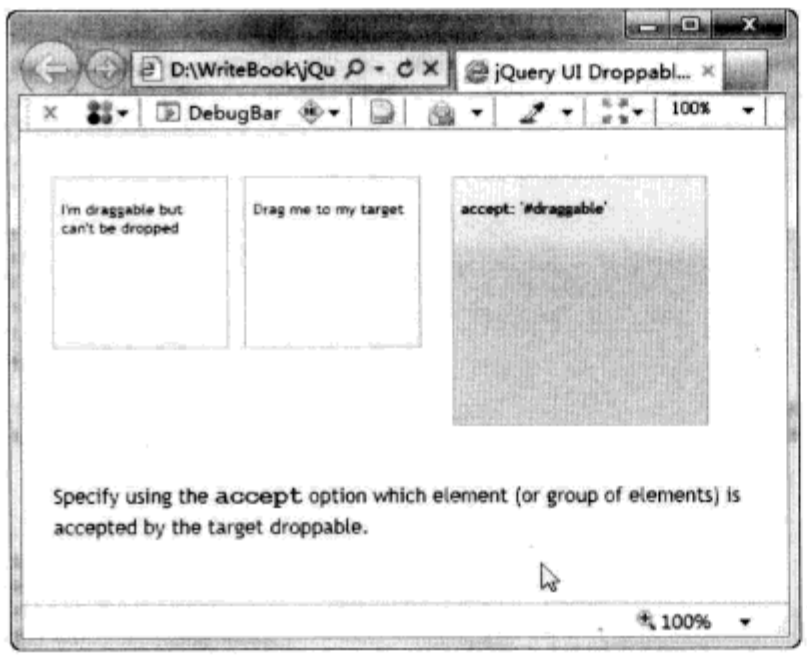


图 14.26　选择拖动元素

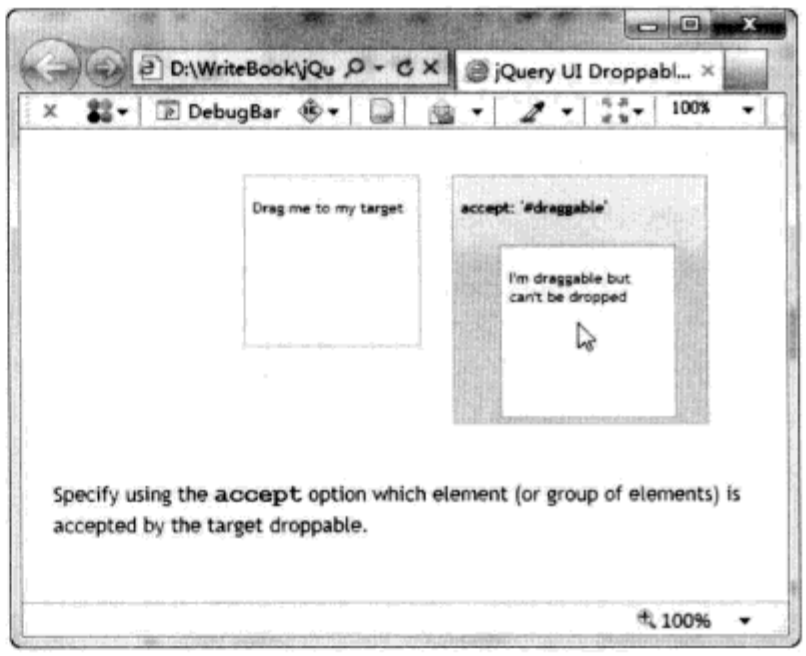


图 14.27　不被接受的拖动元素

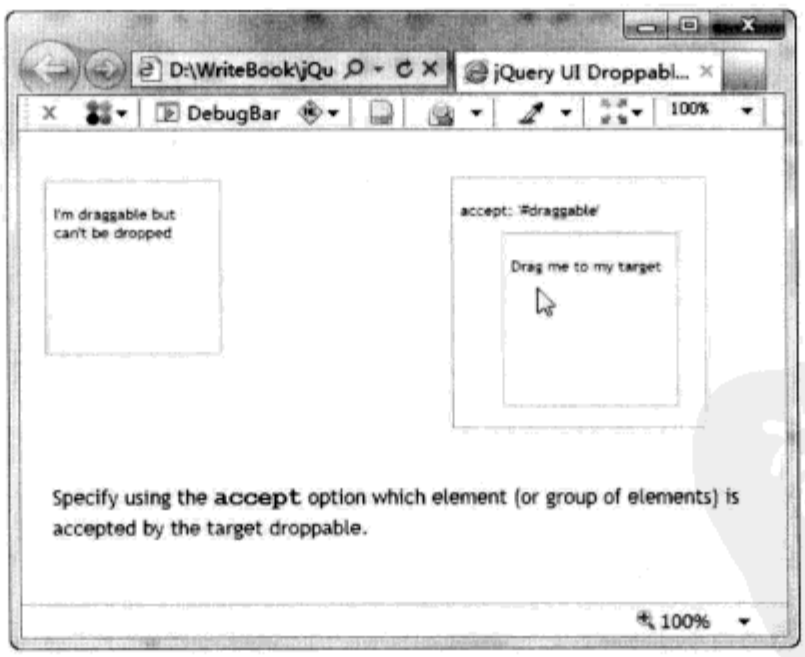


图 14.28　被接受的拖动元素

3. 限定事件传递

这个示例使用了两组投放元素、一个拖动元素，一组不限定事件传递，另一组限定事

件传递。当在 drop 事件中设定返回值为 false 时，上层（小）的投放元素的投放事件会传递到下层（大）的投放元素中，如果不设定返回值为 false，则下层（大）投放元素不会产生投放事件。

```
<script>
$(function() {
    $( "#draggable" ).draggable();
    $( "#droppable, #droppable-inner" ).droppable({
        activeClass: "ui-state-hover",
        hoverClass: "ui-state-active",
        drop: function( event, ui ) {
            $( this )
                .addClass( "ui-state-highlight" )
                .find( "> p" )
                    .html( "Dropped!" );
            return false;
        }
    });
    $( "#droppable2, #droppable2-inner" ).droppable({
        greedy: true,
        activeClass: "ui-state-hover",
        hoverClass: "ui-state-active",
        drop: function( event, ui ) {
            $( this )
                .addClass( "ui-state-highlight" )
                .find( "> p" )
                    .html( "Dropped!" );
        }
    });
});
</script>
```

上述代码第 16 行组织了事件传递效果。其他代码和前面的例子类似。效果如图 14.29～图 14.31 所示。

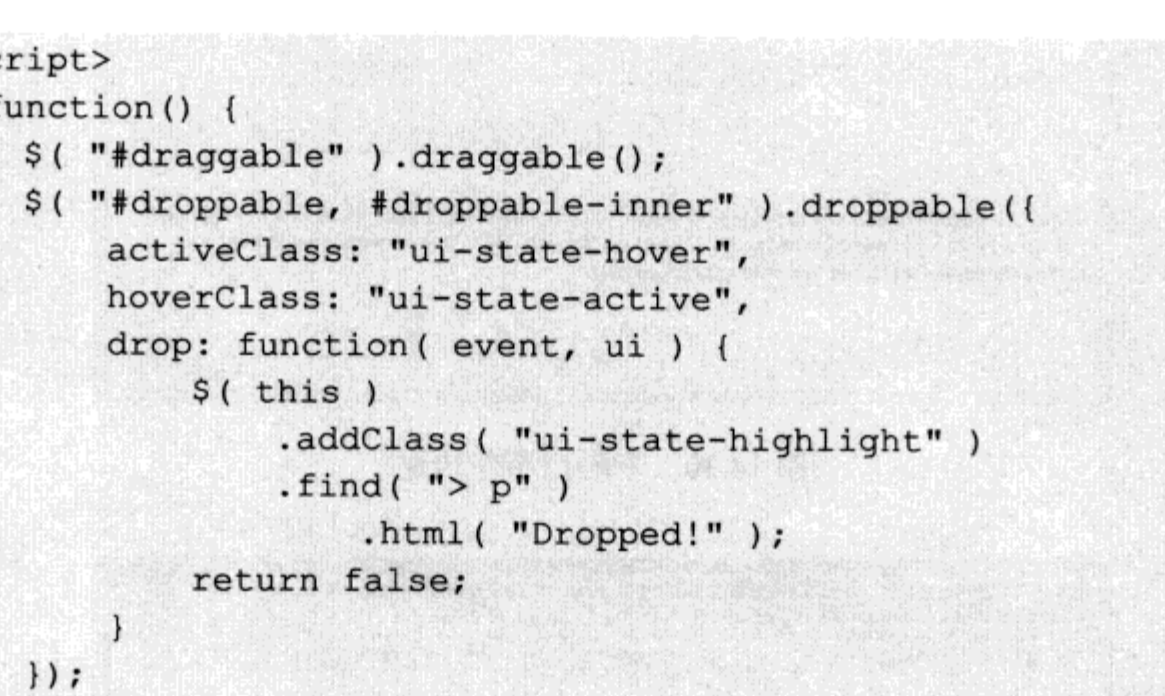


图 14.29　限定事件传递

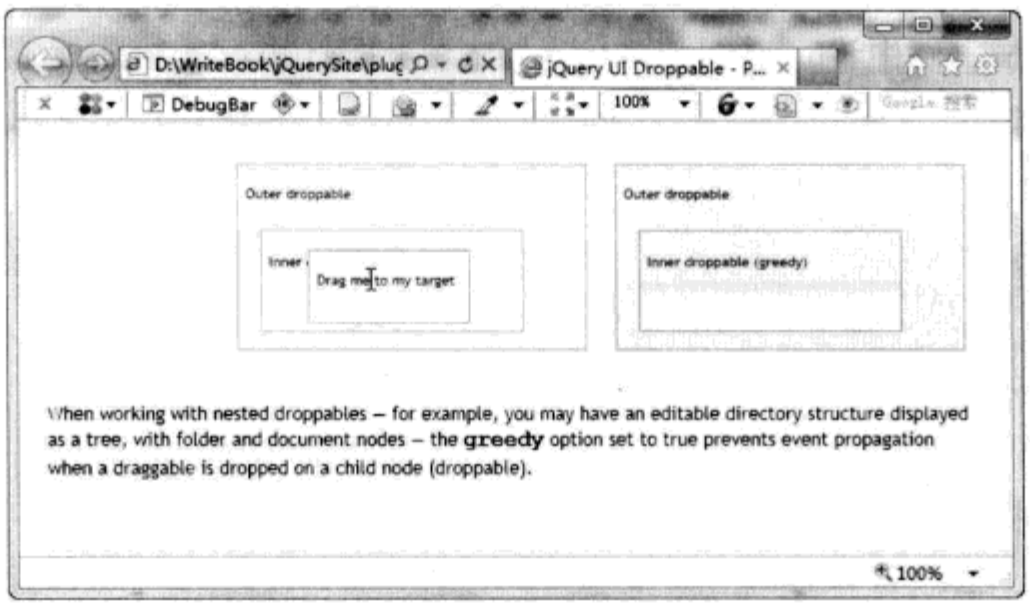


图 14.30　未阻止事件传递

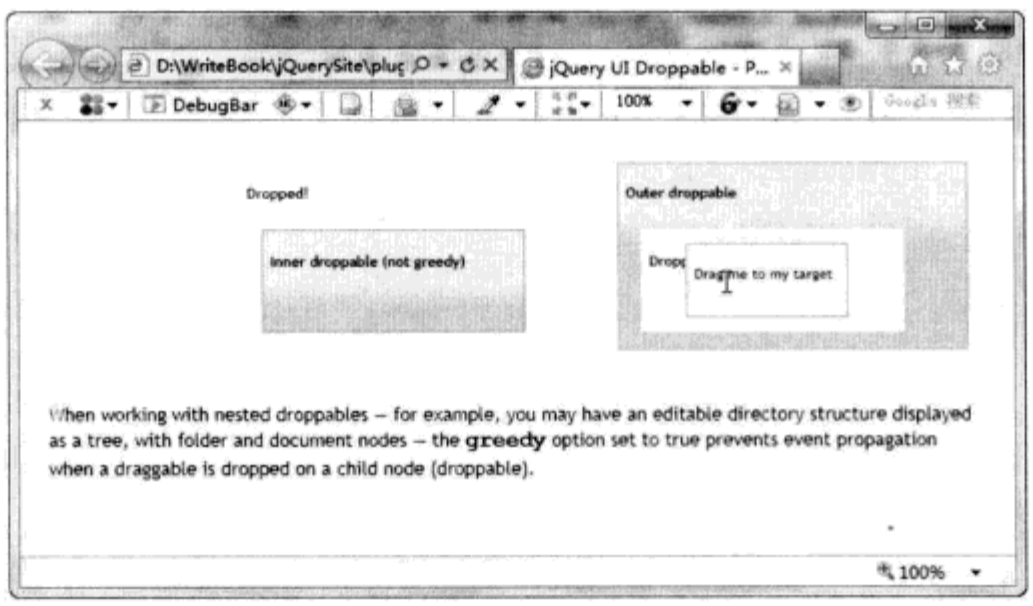


图 14.31　阻止了事件传递

4. 购物车效果

这个示例使用了折叠插件、拖动插件、投放插件和排序表格插件，实现拖放操作购物车效果。

```
<script>
$(function() {
    $( "#catalog" ).accordion();
    $( "#catalog li" ).draggable({
        appendTo: "body",        //拖动元素添加到哪一个元素中
        helper: "clone"          //设定克隆行为
    });
    $( "#cart ol" ).droppable({
        activeClass: "ui-state-default",
        hoverClass: "ui-state-hover",
        accept: ":not(.ui-sortable-helper)",   //接受元素时样式
        drop: function( event, ui ) {
            $( this ).find( ".placeholder" ).remove();
            $( "<li></li>" ).text( ui.draggable.text() ).appendTo( this );
        }
```

```
16      }).sortable({                                    //可排序表格
17          items: "li:not(.placeholder)",               //表格
18          sort: function() {
19              $( this ).removeClass( "ui-state-default" );
20          }
21      });
22  });
23  </script>
```

上述代码第 3 行对 ID 为 catalog 的层创建折叠元素。第 4～7 行创建了多个拖动元素，这些元素添加到 body 中，每个元素在拖动时都被克隆。第 8～15 行创建投放元素。第 9 行设定投放元素激活样式。第 10 行设定当可接受的拖动元素悬停在投放元素上时的样式。第 11 行设定投放元素可接受的拖动元素为去除了样式为.ui-sortable-helper 的元素。第 13 行删除投放元素中的提示内容。

第 14 行向投放元素中加入一个列表项，列表项的文本是鼠标所拖动的元素的文本。第 16～23 行在投放元素的基础上初始化一个排序列表。第 17 行指定排序元素除了投放元素中提示信息以外的所有列表项。第 18 行设定排序功能函数。第 19 行删除样式类 ui-state-default。效果如图 14.32 所示。

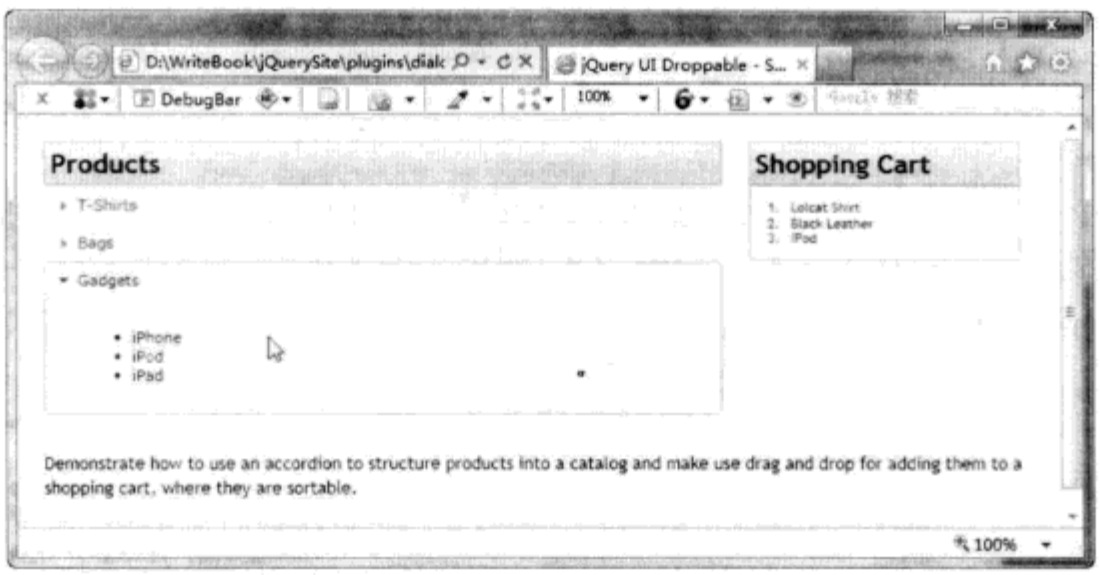


图 14.32　购物车效果

14.3　小　　结

网页元素拖放效果是现代网页中常用到的一种效果。本章主要介绍了 jQuery 拖动插件的基础知识，以及如何利用拖动插件。重点内容是如何利用拖动插件，难点是拖动插件的配合使用。下一章将介绍 jQuery 自定义插件。

14.4　习　　题

【习题 1】练习 jQuery UI 中的拖动插件使用方法。

【习题 2】练习 jQuery UI 中的投放插件使用方法。

第15章 插件开发

前面的章节除了讲解如何用 jQuery 实现特效，还讲解了很多 jQuery 插件。jQuery 插件是代码重用的一个很好的体现。那么如何开发 jQuery 插件？开发插件过程中需要用到哪些知识？需要注意什么？这正是本章要讲解的主要内容。

15.1 jQuery 插件开发基础

本节介绍在 jQuery 开发过程中需要用到哪些基础知识，插件与函数之间的区别是什么。

15.1.1 jQuery 插件介绍

jQuery 插件总体来说是对 jQuery 框架的功能补充，有着相对独立的功能，并且自成体系。它有着独立的文件和方法，通过接口进行调用。这里所说的接口就是前面在介绍插件时使用的初始化函数及其他功能函数。更形象地说，jQuery 插件是提供各种各样功能扩展的 jQuery 模块，类似于计算机的各种功能扩展卡。

1. jQuery插件分类

这里所说的分类不是指功能分类，而是在开发 jQuery 插件的过程中，开发形式上的分类。

- 封装对象方法的插件：这种插件也叫对象级别插件，它主要为对象添加方法，并将这些方法封装到对象中。
- 封装全局函数的插件：这种插件也叫类级别插件，类级别插件的开发最直接的理解就是给 jQuery 类添加类方法，可以理解为添加静态方法。典型的示例就是 $.AJAX()这个函数，将函数定义于 jQuery 的命名空间中。
- 扩展选择器插件：这种插件是在原有 jQuery 选择器的基础上对特定对象扩展使用特殊选择器。例如，MoreSelectors For Jquery 就是一个典型的 jQuery 选择器扩展。

2. jQuery插件与jQuery固有函数的区别

jQuery 固有函数是 jQuery 框架内置的函数，不需要添加额外文件引用。而 jQuery 插件则是在 jQuery 框架基础上建立起来的功能模块。在插件内同样需要使用 jQuery 内置函数。插件是需要 jQuery 固有工具函数支持的。

15.1.2　jQuery 插件开发基础知识

开发 jQuery 插件需要掌握如下知识点。

（1）HTML/XHTML，这个不用多说，我们的 jQuery 的所有实现效果最后都要通过标记来实现。

（2）CSS，在前面使用插件的过程中我们发现，很多插件除了有自己的功能代码文件外，还有配套的 CSS 文件，这些文件主要在插件动态改变元素样式时使用。

（3）JavaScript，jQuery 框架本身就是由 JavaScript 编写的，我们在使用 jQuery 时也是通过 JavaScript 语言来操作的。

jQuery 框架中需要注意的内容如下。

（1）选择器，$是 jQuery 的简写形式。所以，jQuery()和$()的意思是一样的。所有用$()的地方，$都可以用 jQuery 代替。

（2）插件类型声明，用 jQuery.extend 增加的函数，或者说扩展的函数，可以理解成添加类方法，用类名调用；用 jQuery.fn.extend 增加的函数，或者说扩展的函数，可以理解成添加对象方法，即添加成员函数，用对象名调用。

（3）对象原型，jQuery.fn=jQuery.prototype，所以，jQuery.fn 是 jQuery.prototype 的别名。

（4）内置工具函数。

15.1.3　创建一个简单 jQuery 插件

下面通过一个示例来初步认识一下 jQuery 插件开发过程。这个示例是封装对象方法的插件示例。

首先，声明插件，定义一个新的函数原型到 jQuery.fn 对象中，函数原型的名字就是插件的名字：

```
jQuery.fn.myPlugin = function() {
  //添加插件成员
};
```

在上面的代码中定义了一个名为 myPlugin 的插件。上面的写法还是有些问题，因为如果像上面那样写，可能会导致与其他 jQuery 库冲突。为了解决这个问题，可以将上面的写法修改为：

```
(function( $ ){
  $.fn.myPlugin = function() {
    // 添加插件成员
  };
})( jQuery );
```

上面这种写法叫封闭或者闭包，在封闭的函数内部可以使用 jQuery 特有的$符，而不会出现任何问题。还有一点要说明的就是在插件内部代码中出现的 this 关键字，它代表调

用当前编写的插件的 jQuery 对象，因此我们不用再次在插件内包装 this 关键字。

1. this关键字的使用

下面通过示例代码来使用 this 关键字。

```
(function( $ ){
 $.fn.myPlugin = function() {
  //在代码中不需要写$(this)，因为 this 本身就代表 DOM 元素对象
   this.fadeIn('normal', function(){
     // 功能代码
   });
 };
})( jQuery );
```

调用插件代码可以这样写：

```
$('#element').myPlugin();
```

下面补充代码：

```
(function( $ ){
 $.fn.maxHeight = function() {
    var max = 0;
   this.each(function() {
     max = Math.max( max, $(this).height() );
   });
   return max;
 };
})( jQuery );
```

这个插件完成的功能很简单，实现对所有匹配元素的高度取最大值。可以使用一个 HTML 文件来进行插件测试。首先，在页面建立 4 个<DIV>，分别设置不同内容：

```
<div id="div1">jQuery<br />jQuery</div>
<div id="div2">jQuery<br />jQuery<br />jQuery<br />jQuery</div>
<div id="div3">jQuery<br />jQuery<br />jQuery<br />jQuery<br />jQuery<br />
jQuery</div>
<div id="div4">jQuery<br />jQuery<br />jQuery<br />jQuery<br />jQuery<br />
jQuery<br />jQuery<br />jQuery</div>
```

在 CSS 样式设定中分别对它们使用不同的背景颜色，代码参考光盘内容。调用插件形式如下：

```
<script src="jslib/jquery-1.6.js" type="text/javascript"></script>
<script src="js/MyPlugin.js" type="text/javascript"></script>
<script type="text/javascript">
  $(function(){
      alert("各层最大高度为: "+$("div").maxHeight()+"px");
  });
</script>
```

效果如图 15.1 所示。

2. 插件中this对象的返回

刚才编写的插件很简单，而且返回的内容也只是一个整型值。但是，对于 jQuery 来说它的优势之一是函数链，也就是一个对象可以一次性通过调用一系列的函数完成某个功能。这需要 jQuery 函数能如实返回调用它的对象。因此，为了保证 jQuery 的链接特性，我们要保证编写的插件能够返回 this 对象。下面在原有插件的基础上进行修改，完成返回 this 对象功能。

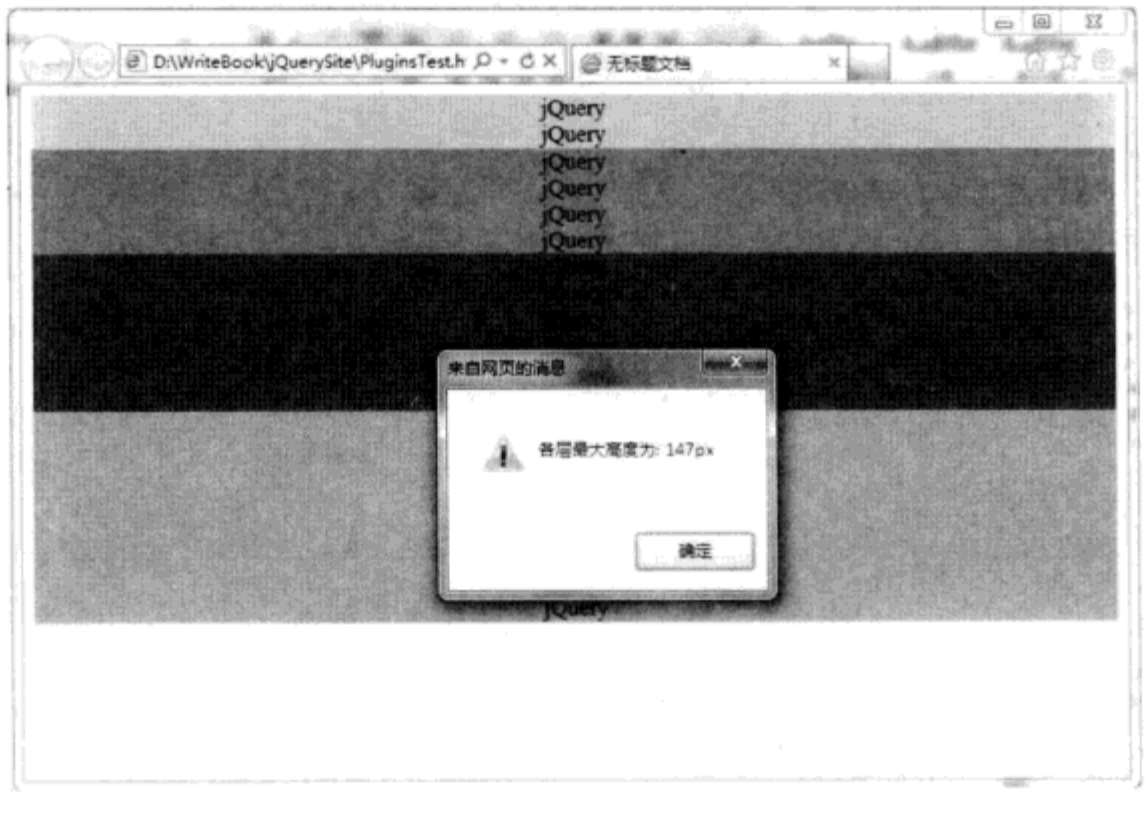


图 15.1　自定义插件示例一

```
(function( $ ){
 $.fn.lockDimensions = function( type ) {
   return this.each(function() {
     var $this = $(this);
     if ( !type || type == 'width' ) {
       alert($this.width());
     }
     if ( !type || type == 'height' ) {
       alert($this.height() );
      }
   });
 };
})( jQuery );
```

可以在页面上测试这个插件的作用，以及它是否可以完成可链接操作。利用页面元素调用插件函数名（插件函数名即我们在插件中定义的$.fn.lockDimensions），并根据插件定义定义形式传入参数。

```
<script type="text/javascript">
  $(function(){
      $("div").lockDimensions('height').css('color', 'red');
  });
</script>
```

HTML 代码和 CSS 样式与上面的例子相同，只是调用插件的方式有所不同。在插件代码的第 3 行就是我们所强调的返回对象本身，也就是将 this 返回。只有这样，在测试代码的第 3 行才可以在调用插件函数结束后继续调用 css()函数设定文字颜色。效果如图 15.2 和图 15.3 所示。

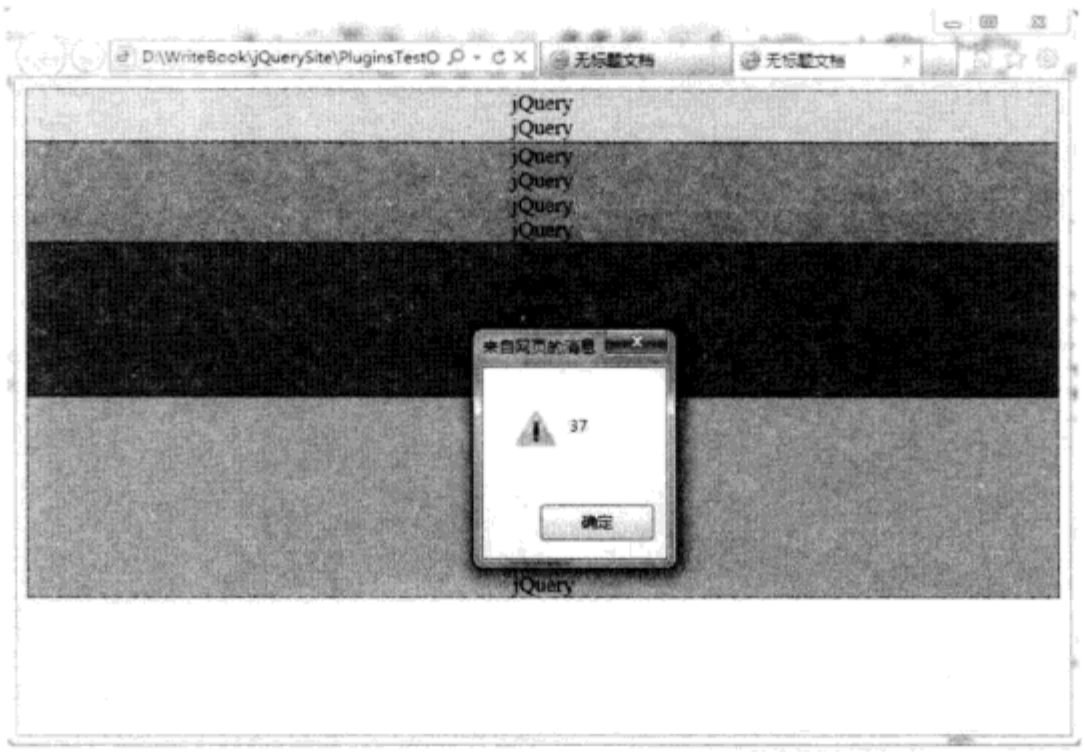


图 15.2　页面加载完成输出各层的高

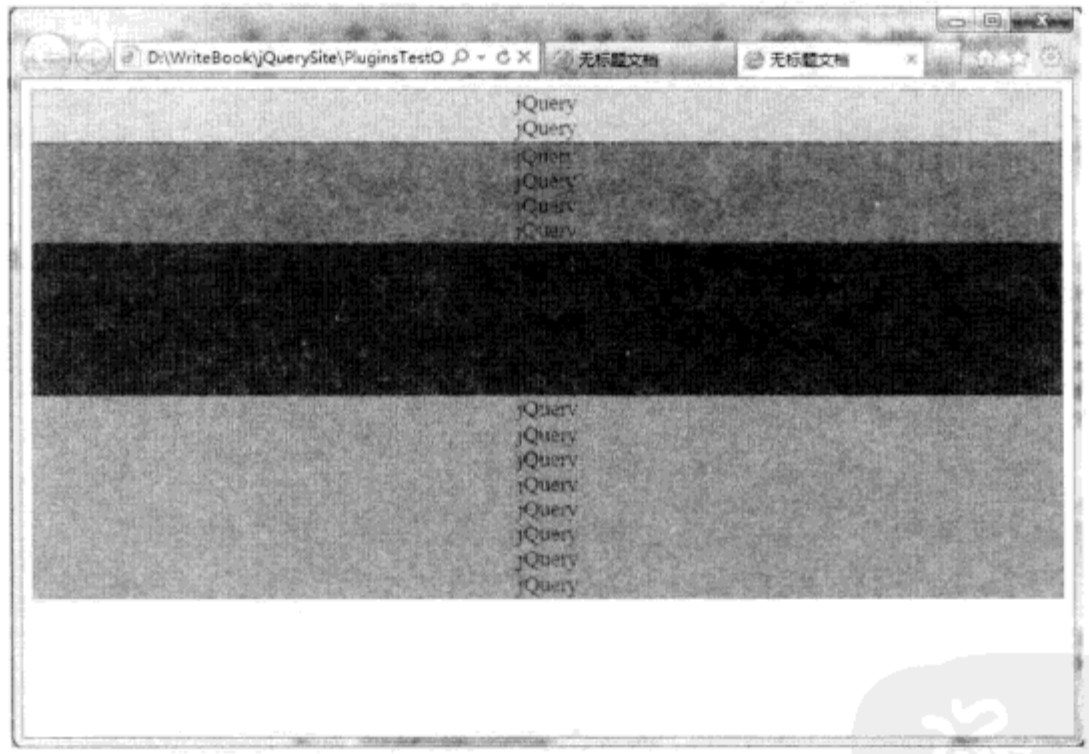


图 15.3　插件功能完成后修改字体颜色

3. 以选项对象代替零散参数

自定义插件还要设定默认值与选项。对于较复杂的插件，按照惯例不是采用外部直接传递零散参数形式，而是使用一种可迭代的选项对象来进行功能设定。可以看下面这个插件示例：

```
1 (function( $ ){
2  $.fn.tooltip = function( options ) {
```

```
3    var settings = {
4      'location'         : 'top',
5      'background-color' : 'blue'
6    };
7    return this.each(function() {
8      if ( options ) {
9        $.extend( settings, options );
10       alert("当前传入方向："+settings["location"]+"；当前传入背景色："+settings
         ["background-color"]);
11     }
12   else
13       alert("默认方向："+settings["location"]+"；默认背景色："+settings
         ["background-color"]);
14   });
15 };
16})( jQuery );
```

上述代码第 3～6 行是插件的默认选项设置。第 8 行判断调用插件时是否传入了选项对象。第 9 行将外部传进来的选项对象与默认选项合并。

在调用这个插件的时候，可以给定参数，例如：

```
<script type="text/javascript">
    $(function(){
        $('div').tooltip({
          'location'         : 'bottom',
          'background-color' : 'red'
        });
    });
</script>
```

效果如图 15.4 和图 15.5 所示。

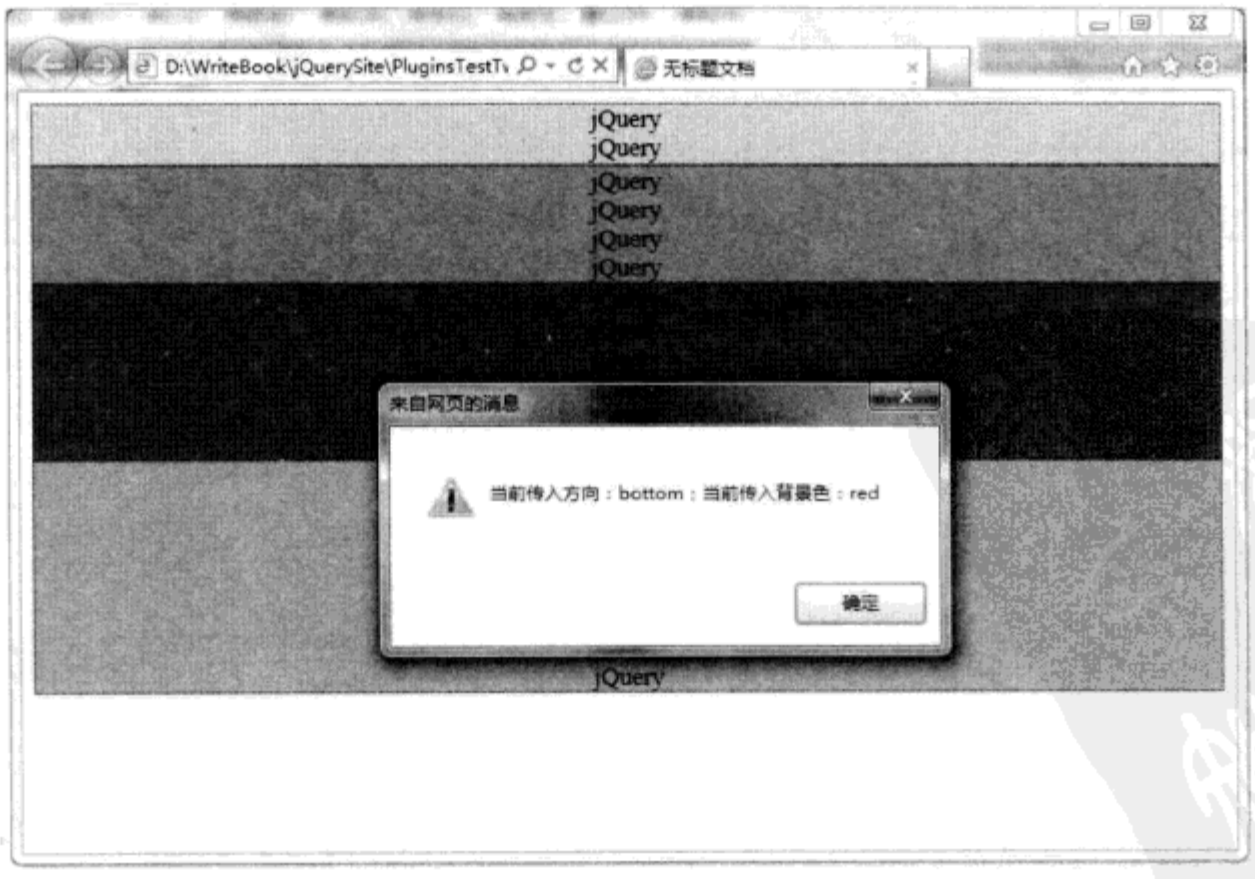


图 15.4　携带选项参数调用插件

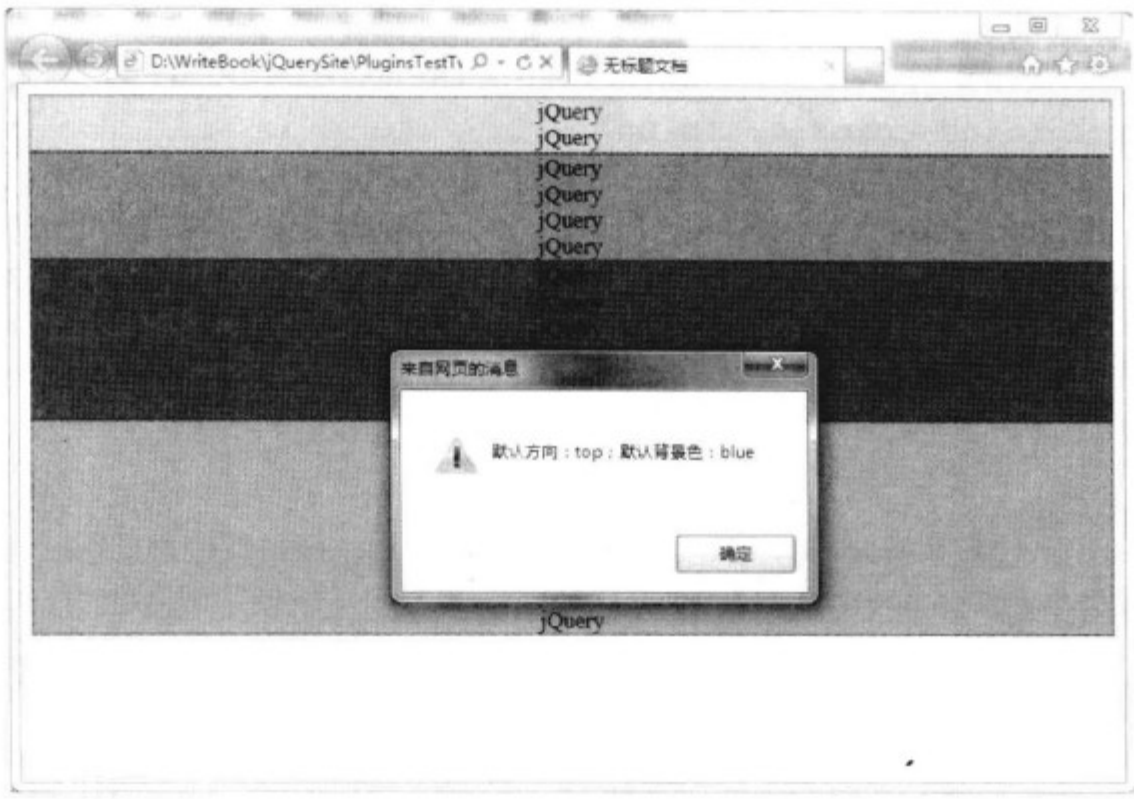


图 15.5　默认选项参数调用插件

4．向插件中添加方法

前面在介绍插件时曾提到插件有一些方法可用。那么这些方法是如何定义在插件中的呢？看下面这个例子：

```
1 (function( $ ){
2  var methods = {
    //声明插件方法
3    init : function( options ) {alert("Initial Method"); },
4    show : function( ) { alert("Show Method");   },
5    hide : function( ) { alert("Hide Method"); },
6    update : function( content ) { alert("Update Method "+"Content:
     "+content); }
7  };
8  $.fn.tooltip = function( method ) {
    //定义方法功能
9    if ( methods[method] ) {
10     return methods[ method ].apply( this, Array.prototype.slice.call
       ( arguments, 1 ));
11   } else if ( typeof method === 'object' || ! method ) {
12     return methods.init.apply( this, arguments );
13   } else {
14     $.error( 'Method ' +  method + ' does not exist on jQuery.tooltip' );
15   }
16 };
17 })( jQuery );
```

上述代码第 3～6 行定义了插件的初始化、显示、隐藏、更新等方法，并加入了简单实现。第 9 行判断调用插件时传递的方法名是否有效。第 10 行调用与传进来的方法名对应的方法执行。第 11 行设定如果方法名为空则为初始化方法。第 13 行表示如果方法名不为空但是不在插件定义的方法范围内，则利用 jQuery 的异常处理抛出异常。

下面对方法的使用进行测试：

```
<script type="text/javascript">
    $(function(){
        $('div').tooltip();
        $('div').tooltip({
          foo : 'bar'
        });
        $('div').tooltip('hide');
        $('div').tooltip('update', 'This is the new tooltip content!');
        $('div').tooltip('show');
    });
</script>
```

效果如图 15.6 和图 15.7 所示。

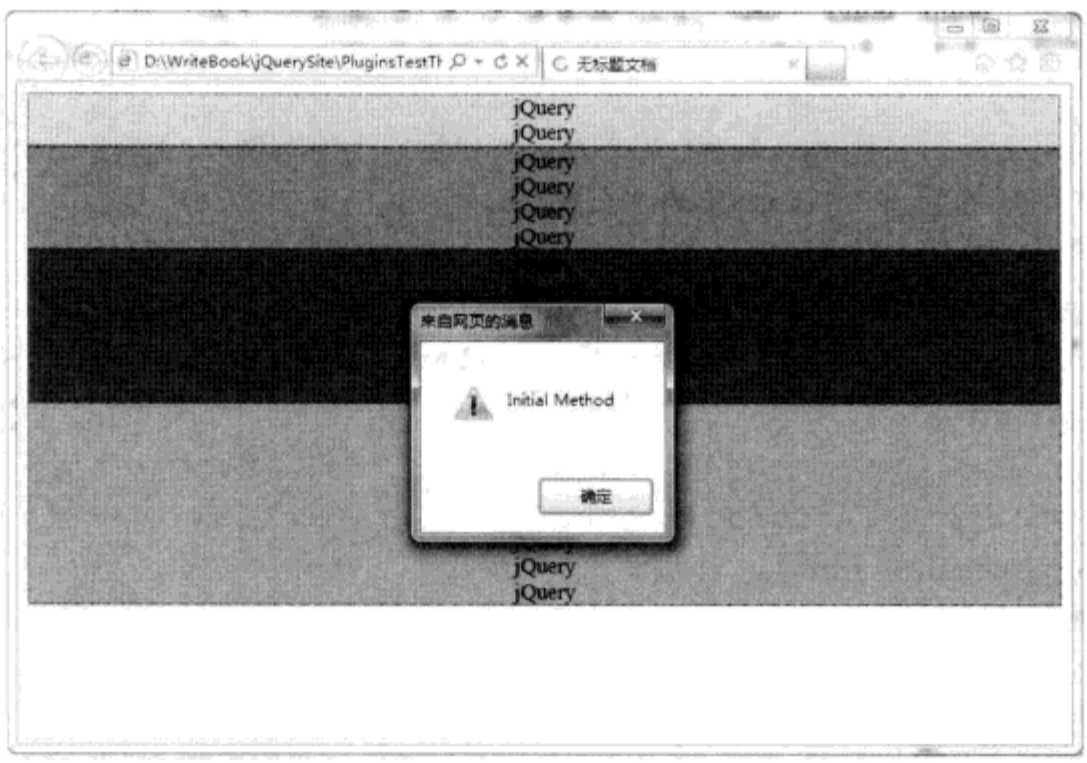


图 15.6　初始化方法调用

图 15.7　更新方法调用

5. 向插件中加入事件

除了可以在插件中声明多个方法之外，还可以对插件绑定事件，使插件能够响应事件并做相应处理。

```
1 (function( $ ){
2  var methods = {
3    init : function( options ) {
4      return this.each(function(){
5        $(window).bind('resize.tooltip', methods.reposition);
6      });
7    },
8    destroy : function( ) {                    //声明事件并添加实现
9      return this.each(function(){
10       $(window).unbind('.tooltip');
11     })
12   },
13   reposition : function( ) {alert("窗口当前宽度: "+$(window).width()+";
     窗口当前高度: "+$(window).height());},
14   show : function( ) {alert("Show Method");},
15   hide : function( ) { alert("Hide Method");},
16   update : function( content ) {alert("Update Method "+"Content:
     "+content); }
17 };
18 $.fn.tooltip = function( method ) {
19   if ( methods[method] ) {
20     return methods[method].apply( this, Array.prototype.slice.call
     ( arguments, 1 ));
21   } else if ( typeof method === 'object' || ! method ) {
22     return methods.init.apply( this, arguments );
23   } else {
24     $.error( 'Method ' +  method + ' does not exist on jQuery.tooltip' );
25   }
26 };
27})( jQuery );
```

上述代码第 5 行在插件初始化时绑定调整大小事件到浏览器窗口，并且指定事件处理程序为函数 reposition()。第 10 行在插件的销毁函数中取消窗口绑定事件。

下面可以测试插件事件：

```
1 <script type="text/javascript">
2   $(function(){
3       $('#div1').tooltip();
4       $('#div1').tooltip({
5         foo : 'bar'
6       });
7       $('#div1').tooltip('hide');
8       $('#div1').tooltip('update', 'This is the new tooltip content!');
9       $('#div1').tooltip('show');
```

```
10      $('#div1').click(function(){$('div').tooltip('destroy')});
11  });
12</script>
```

效果如图 15.8 所示。

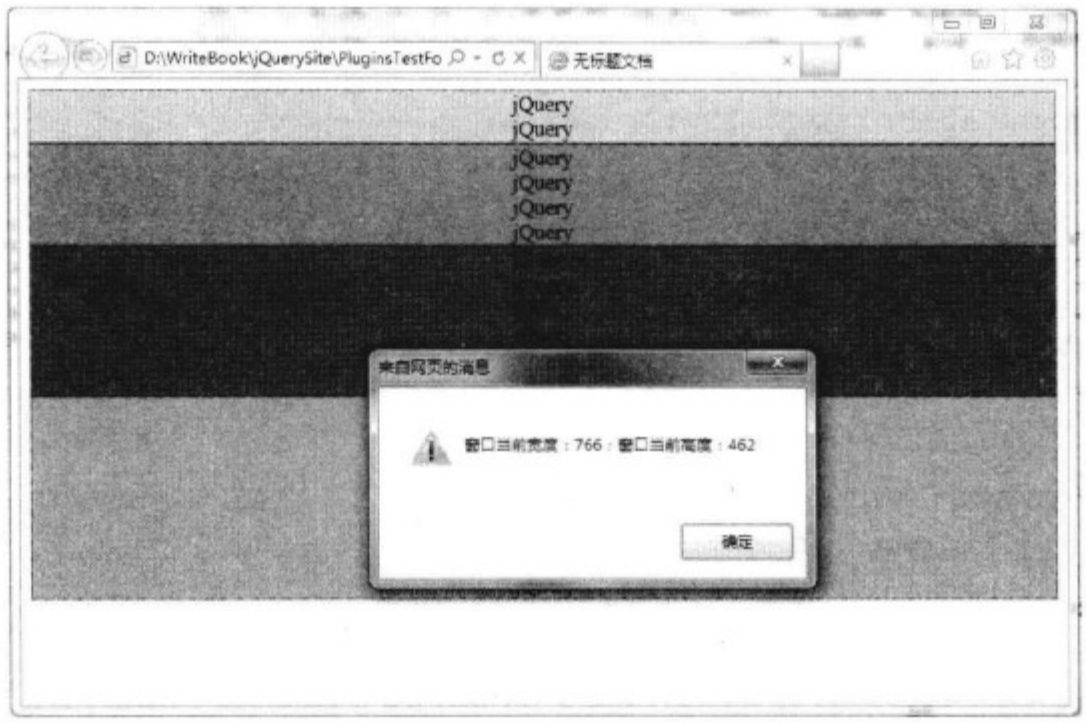


图 15.8　插件事件测试

15.2　插 件 示 例

本节将通过写一个比较具有应用性的插件来完整地体验一下插件开发过程。这个插件所要实现的功能是当鼠标悬停到超链接上时，可以出现背景颜色和文字颜色都不同的提示信息。这里暂且管它叫彩色工具提示插件。

1. 插件原理及显示风格设定

这个插件的原理是利用 HTML 标记中添加的 title 属性，将其转换至页面上，并形成一系列的工具提示。它里面共有 6 种颜色可选，当然，也可以根据个人需要添加更多种颜色选择。这个插件也是一种比较新颖的用户体验。

它的转换机制如下。当在源文件中出现这样一行 HTML 代码时：

```
<a href="index.html" class="blue" title="转到首页">首页</a>
```

使用 jQuery 插件转换后页面应该是下面的样子：

```
<a class="blue colorTipContainer" href="index">首页
<span class="colorTip" style="margin-left: -60px;">回到首页</span>
<span class="pointyTipShadow"></span>
<span class="pointyTip"></span>
</span>
</a>
```

注：上述代码中的 blue 样式类名是为了覆盖默认的提示信息的背景色。

为了能够让工具提示在正确的位置显示，并且显示样式正常，需要定制 CSS 样式文件，这个文件是和插件搭配使用的：

```
.colorTipContainer{
  position:relative;
  text-decoration:none !important;
}
.colorTip{
  /* 彩色工具提示样式类 */
  display:none;
  position:absolute;
  left:50%;
  top:-30px;
  padding:6px;
  background-color:white;
  font-family:Arial,Helvetica,sans-serif;
  font-size:11px;
  font-style:normal;
  line-height:1;
  text-decoration:none;
  text-align:center;
  text-shadow:0 0 1px white;
  white-space:nowrap;
  -moz-border-radius:4px;
  -webkit-border-radius:4px;
  border-radius:4px;
}
.pointyTip,.pointyTipShadow{
  /* 彩色提示信息下端的三角形样式 */
  border:6px solid transparent;
  bottom:-12px;
  height:0;
  left:50%;
  margin-left:-6px;
  position:absolute;
  width:0;
}
.pointyTipShadow{
  /* 边框阴影样式 */
  border-width:7px;
  bottom:-14px;
  margin-left:-7px;
}
```

上述代码中.colorTipContainer 样式类是彩色工具提示的容器元素样式类。为了创建不同颜色的工具提示信息，还需要在 CSS 文件中加入这个插件的相关主题。目前，我们给出了 6 个主题，读者可以根据自己的需要，按照主题格式创建自己的主题风格。

```
/* 6 种主题样式 */

.white .pointyTip{ border-top-color:white;}
```

```
.white .pointyTipShadow{ border-top-color:#ddd;}
.white .colorTip{
    background-color:white;
    border:1px solid #DDDDDD;
    color:#555555;
}

.yellow .pointyTip{ border-top-color:#f9f2ba;}
.yellow .pointyTipShadow{ border-top-color:#e9d315;}
.yellow .colorTip{
    background-color:#f9f2ba;
    border:1px solid #e9d315;
    color:#5b5316;
}

.blue .pointyTip{ border-top-color:#d9f1fb;}
.blue .pointyTipShadow{ border-top-color:#7fcdee;}
.blue .colorTip{
    background-color:#d9f1fb;
    border:1px solid #7fcdee;
    color:#1b475a;
}

.green .pointyTip{ border-top-color:#f2fdf1;}
.green .pointyTipShadow{ border-top-color:#b6e184;}
.green .colorTip{
    background-color:#f2fdf1;
    border:1px solid #b6e184;
    color:#558221;
}

.red .pointyTip{ border-top-color:#bb3b1d;}
.red .pointyTipShadow{ border-top-color:#8f2a0f;}
.red .colorTip{
    background-color:#bb3b1d;
    border:1px solid #8f2a0f;
    color:#fcfcfc;
    text-shadow:none;
}

.black .pointyTip{ border-top-color:#333;}
.black .pointyTipShadow{ border-top-color:#111;}
.black .colorTip{
    background-color:#333;
    border:1px solid #111;
    color:#fcfcfc;
    text-shadow:none;
}
```

至此，显示样式的准备工作基本完成了。下面就可以着手编写插件的 JavaScript 文件了。它的实现步骤如下。

（1）获取调用插件时的主题类名参数。

（2）获取元素的 title 属性，如果元素没有这个属性，则直接停止插件工作。

（3）创建鼠标在元素的悬停与离开事件发生时，插件对定时器及彩色工具提示显示与隐藏的设定。

（4）在元素上创建元素来表示彩色工具提示。

（5）为工具提示选定颜色，如果颜色在主题中被设定，则使用，否则使用默认颜色。

（6）注册产生彩色工具提示的触发事件。

2. 基本插件功能实现

首先，我们来看一下插件的基本功能代码：

```
1 (function($){
2   $.fn.colorTip = function(settings){
3       var defaultSettings = {
4           color       : 'yellow',
5           timeout     : 500
6       }
7       var supportedColors = ['red','green','blue','white','yellow',
        'black'];
8       /*将默认设置合并到设置参数中 */
9       settings = $.extend(defaultSettings,settings);
10      /*
11      *   遍历所有的元素
12      *   将插件添加至方法链中
13      */
14      return this.each(function(){
15          var elem = $(this);
16          //如果当前遍历的元素 title 属性为空则返回
17          if(!elem.attr('title')) return true;
18          //创建事件安排对象及提示信息对象
19          //这两个对象的定义在插件后续部分
20          var scheduleEvent = new eventScheduler();
21          var tip = new Tip(elem.attr('title'));
22          // 添加工具提示到当前遍历的元素上
23          // 并应用插件指定样式
24          elem.append(tip.generate()).addClass('colorTipContainer');
25          // 判断是否提供了插件支持的颜色
26          // 的类名
27          var hasClass = false;
28          for(var i=0;i<supportedColors.length;i++)
29          {
30              if(elem.hasClass(supportedColors[i])){
31                  hasClass = true;
32                  break;
33              }
34          }
35          //如果已经设置了插件支持颜色则使用
36          if(!hasClass){
37              elem.addClass(settings.color);
```

```
38            }
39            //鼠标悬停到元素上则显示提示
40            //鼠标离开则提示延迟隐藏
41            elem.hover(function(){
42                tip.show();
43                //当鼠标移动到元素上
44                //清除先前定义的延迟隐藏设置
45                scheduleEvent.clear();
46            },function(){
47                // 鼠标从元素上离开
48                // 则设定延迟隐藏提示
49                scheduleEvent.set(function(){
50                    tip.hide();
51                },settings.timeout);
52            });
53            // 删除元素上的 title 属性
54            elem.removeAttr('title');
55        });
56    }
57 })(jQuery);
```

上述代码是插件的整体代码，其中第 3～6 行是插件的默认设置。第 7 行设定了插件会支持的颜色选择数组。第 28～34 行判断调用插件所提供的颜色插件是否支持。第 41～52 行是提示显示与隐藏的触发事件，这里使用了元素的鼠标悬停与鼠标离开事件。下面给出插件中要使用的两个内置对象的定义：

```
/*
/   事件设定类定义
*/
function eventScheduler(){}
eventScheduler.prototype = {
    set : function (func,timeout){
        // set 方法设置了两个参数
        //一个是需要执行的函数，另一个是等待执行的时间周期
        this.timer = setTimeout(func,timeout);
    },
    clear: function(){
        // 清除 set 方法中定义的时间周期
        clearTimeout(this.timer);
    }
}
```

这段代码是我们的事件处理加工对象，主要负责提示隐藏周期设置和当提示显示时，删除隐藏周期。

```
1   /*
2   /   提示对象定义
3   */
4   function Tip(txt){
5       this.content = txt;
6       this.shown = false;
```

```
7   }
8   Tip.prototype = {
9       generate: function(){
10          // generate 方法产生提示对象
11          // 存储在 tip 变量中
12          //并交给插件进行处理
13          return this.tip || (this.tip = $('<span class="colorTip">
            '+this.content+
14                                          '<span class="pointyTipShadow">
                                            </span><span class="pointyTip">
                                            </span></span>'));
15      },
16      show: function(){
17          if(this.shown) return;
18          // 设定插件显示样式并以淡入效果出现
19          this.tip.css('margin-left',-this.tip.outerWidth()/2).fadeIn
            ('fast');
20          this.shown = true;
21      },
22      hide: function(){
23          this.tip.fadeOut();
24          this.shown = false;
25      }
26  }
```

上述代码是提示对象的定义，第 9～15 行利用 HTML 标记组成提示的显示样式，并以 jQuery 对象形式保存交给插件处理。第 16～21 行是提示对象的显示功能。第 22～25 行是提示的隐藏功能。

具体效果如图 15.9 和图 15.10 所示。

图 15.9　自定义插件效果一

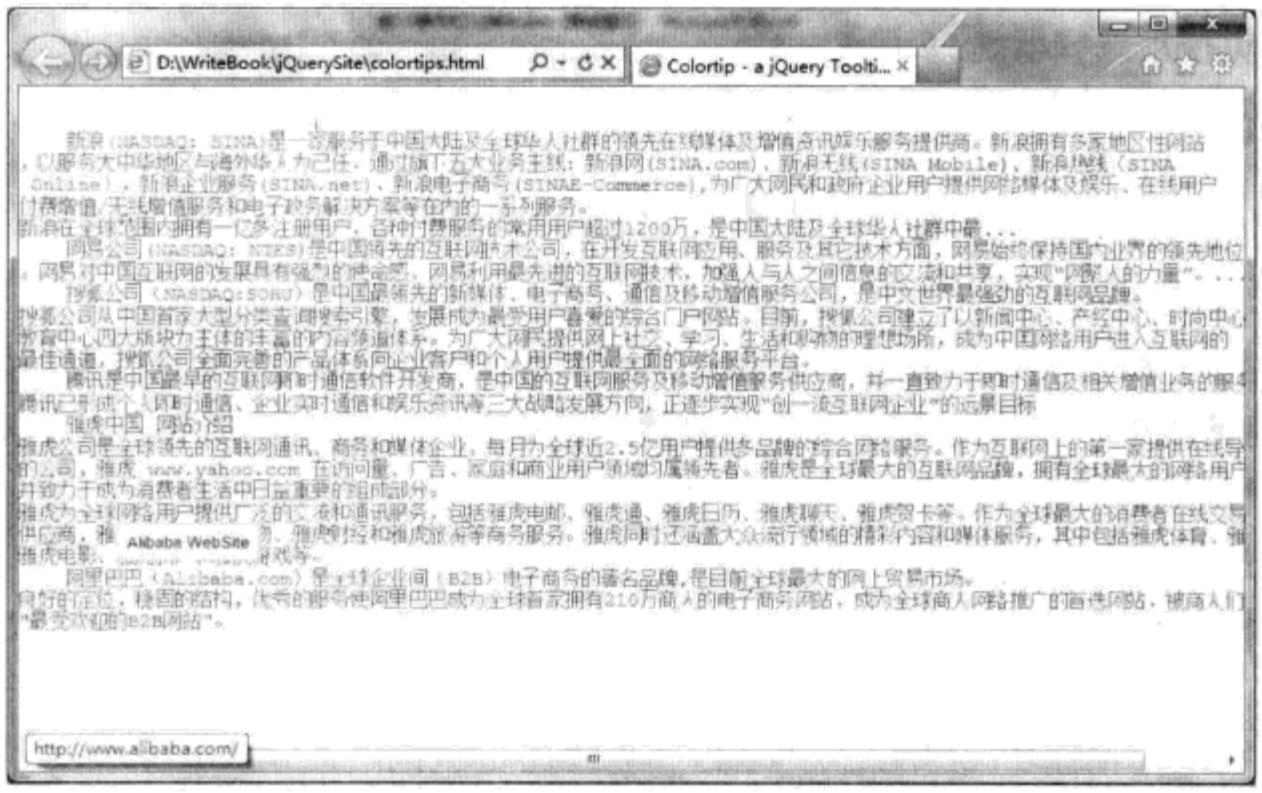


图 15.10　自定义插件效果二

15.3　插件开发规范

前面介绍了 jQuery 插件的相关方法，本节总结一下在开发 jQuery 插件的过程中应该遵守的一些规范。

（1）使用闭包。这个规范在前面介绍插件基本知识的时候已经介绍过，它的优点如下。

- 避免全局依赖。
- 避免第三方破坏。
- 兼容 jQuery 操作符。

（2）关于 this 关键字的使用。

- 不要重复包装 this 关键字。
- 除非你需要返回一个固有的值，否则必须返回 this 引用，以保持 jQuery 方法的链接性。

（3）通过选项对象传递参数给插件，而不要使用零散的参数形式。

（4）不要在插件中出现杂乱的插件命名空间。

（5）必须为插件的方法及事件命名。

（6）利用 jQuery 提供的扩展功能，建议使用$.fn.extend 而不是$.extend。

$.extend 用于扩展自身方法，如$.ajax, $.getJSON 等，$.fn.extend 则用于扩展 jQuery 类，包括方法和对 jQuery 对象的操作。为了保持 jQuery 的完整性，建议使用$.fn.extend 进行插件开发而尽量少使用$.extend。

（7）选择器的使用规范如下。

- 尽量使用 ID 选择器。
- 在样式选择中尽量搭配上标记名。

- 避免使用选择器迭代。

15.4　小　　结

本章主要介绍了 jQuery 插件的基础知识，以及如何开发插件。重点内容是如何开发具有一定功能的插件，以及在开发过程中要遵守的插件开发规范。插件开发是本章的难点部分。

15.5　习　　题

【习题】利用本章所讲内容，自定义一个控制元素显示与隐藏的插件，并可更换元素背景色，要求符合插件编写规范。

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

[General Information]
书名=Web开发典藏大系 jQuery网页开发实例精解
作者=黄格力等编著
页码=350
ISBN=350
SS号=13047516
dxNumber=0000083215 80
出版时间=2012.07
出版社=该引擎未能查询到
定价：37.40
试读地址=http://book.duxiu.com/bookDetail.jsp?dxNumber=000008321580&d=D31313D9D62B10CE7B384EA4BD2C87F0&fenlei=181704030 2&sw=jQuery%CD%F8%D2%B3%BF%AA%B7%A2%CA%B5%C0%FD%BE%AB%BD%E2
全文地址=http://cx300.5read.com/image/ss2jpg.dll?did=n34&pid=2C653E736AC3F5A58AAF13C44E053C89BAE9AE869117BE74 12F58CCACE52D5FB4051360BB52FAD63E88FA7E3AD22875E70D9BBE34A1AF5C249D7EBD3329A0CACB9BB28B97E86DB64135766 3BE60FBCF3A7AE8D76EA9D9173EF46306EE5013F34CA967EDC1617B4A5532 83E37242CCABCC211&jid=/