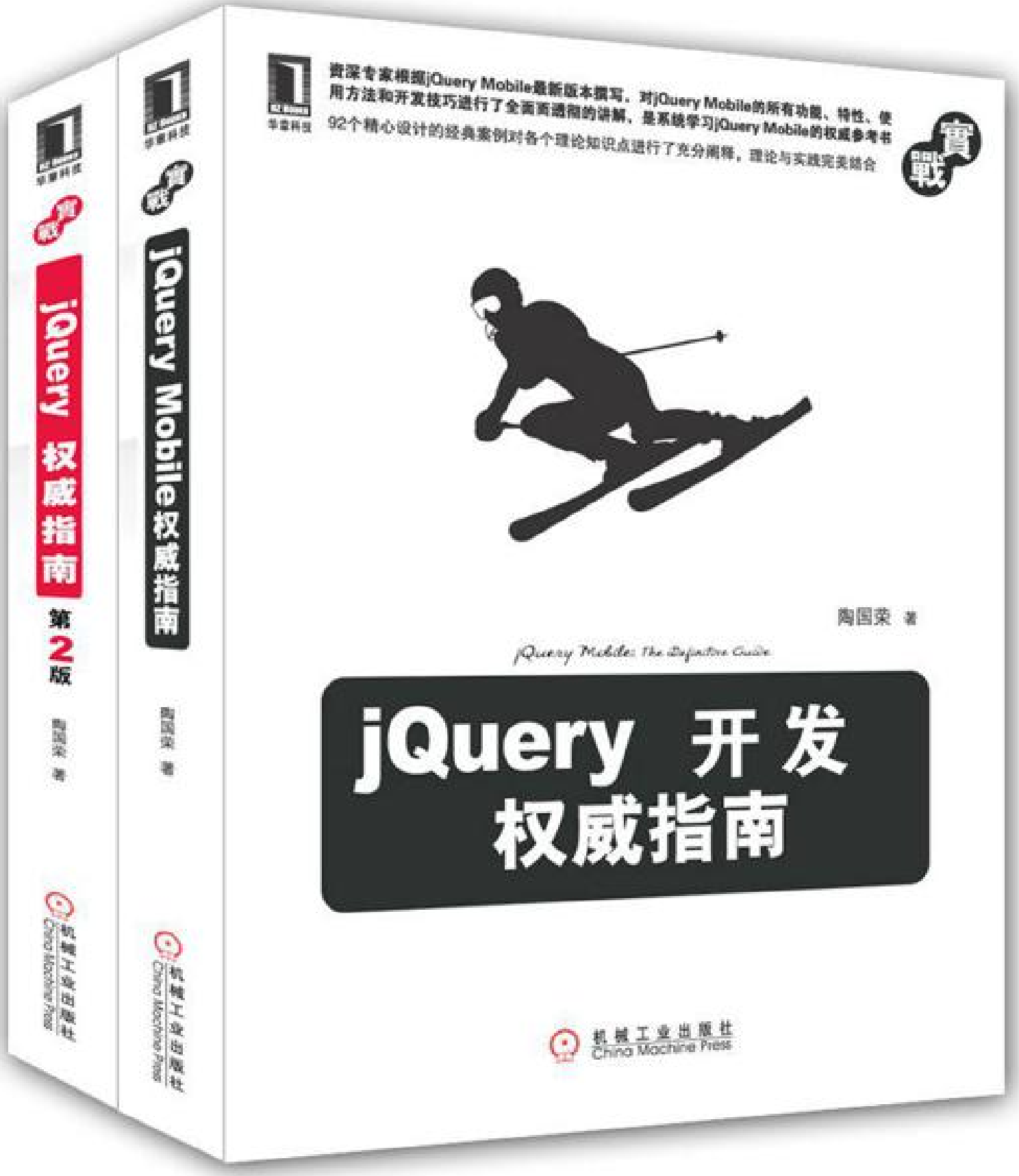

资深专家根据jQuery Mobile最新版本撰写，对jQuery Mobile的所有功能、特性、使用方法和开发技巧进行了全面而透彻的讲解，是系统学习jQuery Mobile的权威参考书
92个精心设计的经典案例对各个理论知识点进行了充分阐释，理论与实践完美结合
实战
陶国荣 著
jQuery Mobile: The Definitive Guide
jQuery 开发
权威指南
机械工业出版社
China Machine Press
jQuery Mobile权威指南
陶国荣 著
jQuery 权威指南
第2版
陶国荣 著

jQuery开发权威指南：jQuery权威指南（第2版）、jQuery Mobile权威指南

陶国荣　著

ISBN：978-7-111-39278-1

ISBN：978-7-111-43593-8

jQuery Mobile权威指南纸版由机械工业出版社于2012年出版，jQuery权威指南（第2版）纸版由机械工业出版社于2013年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.bbbvip.com

新浪微博 @研发书局

腾讯微博 @yanfabook

目录

jQuery Mobile权威指南

jQuery权威指南（第2版）

资深专家根据jQuery Mobile最新版本撰写，对jQuery Mobile的所有功能、特性、使用方法和开发技巧进行了全面而透彻的讲解，是系统学习jQuery Mobile的权威参考书

92个精心设计的经典案例对各个理论知识点进行了充分阐释，理论与实践完美结合

陶国荣 著

jQuery Mobile: The Definitive Guide

jQuery Mobile 权威指南

机械工业出版社
China Machine Press

前言

创作背景

在近几年的IT行业发展中，最热、最快的无疑是移动互联网，它是继互联网之后的又一次信息技术革命。移动互联网巨大的市场潜力创造了无限商机，人们纷纷加入其中，各种与之相应的新技术、新应用层出不穷。

移动应用离不开移动终端设备，如PC、手机、平板电脑等。据估算，到今年年底，市场中各类型智能手机的出货量将突破1.4亿台，应用的前景十分乐观。目前，移动应用的开发平台大体分为三个方向，一个是收益稳定的iOS系统，另一个是如日中天的Android系统，还有一个是蓄势待发的Windows Phone系统。丰富的开发平台增加了开发人员选择的灵活性，但同时也带了一个问题，就是应用平台的兼容性，而这种兼容性目前在各平台中是不可调和的。因此，一个应用需要开发三个不同平台的版本，这种现状将直接导致应用开发和后期维护成本的上升。针对目前的这种多平台现状，如果以Web作为统一的平台、浏览器作为应用的入口，即以WebApp的方式实现相应的应用，将大大简化开发过程和减少应用投入成本，这点也逐渐成为广大Web开发者的共识。

虽然WebApp开发应用将会成为移动开发的一个主方向，但移动设备中浏览器的运行环境远比PC端更为复杂，因此，使用原有的页面框架远不能解决各类型移动设备中浏览器的兼容性问题，必须引入其他轻量级、高性能的Web页面框架。而在这方面，jQuery Mobile在众多移动页面开发框架中脱颖而出，成为众多开发者的首选。

jQuery Mobile的宗旨是在各移动设备的浏览器平台中形成一个统一、灵活、渐进增强的系统，并使该系统在基于jQuery和jQuery UI的基础上进行无缝地工作。它的简略之处在于以纯HTML结构为中心的布局，页面放置DIV和其他标准的HTML组件元素，这种结构方式更容易集成其他的Web服务，同时，与jQuery的集成也将更加有利于移动框架的扩展。

jQuery Mobile除继承了jQuery轻量级、兼容性好的框架主要特征外，还把“write less, do more”主旨提升到更高的一个层次，使开发者在学习和开发成本较低的前提下开发一个兼容各种主流移动平台浏览器的WebApp的愿望成为现实。

作为一项新的技术，jQuery Mobile虽然建立在jQuery和jQuery UI基础之上，以HTML 5新特征为依据，但其中包括众多的页面结构、UI部件、主题和各类型的API，开发人员需要相应的书籍来引导以快速而高效地学习并掌握它，并最终能够真正运用到自己钟爱的移动项目

中。相信本书的诞生，一定会让你在WebApp开发方面有所得、有所悟。

本书特点

“学以致用”是本书的一个重要特点。全书始终体现一个“用”字，无论是理论知识的介绍，还是实例的开发，无一例外都是从实用的角度出发，每一个实例都是经过精心选择，每一个示意图都是作者精心编排并进行扼要说明，使读者通过直观的页面效果加深对应用的理解。

全书以实例为主线，由浅入深，逐步推进，通过全面、详细和完整地介绍，不但能够激发读者的阅读兴趣，还能使读者迅速地掌握jQuery Mobile的页面布局和API用法。

如何阅读本书

全书从一个普通移动页面开发者的角度，详细地介绍了关于jQuery Mobile所涉及的全部新组件与API的用法。全书共10章，整体框架分为三个部分：

第一部分（第1～5章）为jQuery Mobile基础知识。包括初识jQuery Mobile、页面与对话框、工具栏与格式化内容、页面常用组件、jQuery Mobile主题。

第二部分（第6～7章）为jQuery Mobile插件和API介绍。详细介绍了jQuery Mobile的各种插件的使用方法，以及jQuery Mobile API的应用。

第三部分（第8～10章）为jQuery Mobile实践技巧与案例开发。系统地讲解了jQuery Mobile开发的技巧与方法，并通过两个完整的案例，将本书的理论应用于实际的开发中，使读者能够做到学以致用。

本书针对的是Web开发者，无论是前端开发还是后台程序，都可以使用本书。本书的结构是层进式的，章节之间有一定的关联，因此，建议读者按章节的编排，逐章阅读。在阅读过程中，尽量不要照抄实例，重点是理解核心代码，自己动手开发相似功能的应用，并逐步完善其功能，这样才能真正掌握其代码的实质。

勘误和支持

希望这部历时半年、积累我个人心得与技术感悟的拙著能给读者带来移动开发思路上的启发与技术上的提升，使每位读者通过本书能够有所悟或有所得。同时，也非常希望能借本书出版之机与国内热衷于WebApp移动开发技术的开发者进行交流。

除封面署名作者外，参加本书编写工作的还有：刘义、李建洲、李静、裴星如、李建勤、陶红英、陈建平、孙文华、孙义、陶林英、闵慎华、孙芳、赵刚。由于作者的水平有限、编写时间仓促，书中难

免会出现一些错误或者不准确之处，恳请读者批评指正。书中的全部源代码文件可以从华章网站[1]下载。如果您有更多的宝贵意见，也欢迎发送邮件至邮箱tao_guo_rong@163.com，期待能够得到大家真挚的反馈。

致谢

本书的出版首先要感谢机械工业出版社华章分社的编辑们，尤其是杨福川与白宇编辑，正是由于他们的斧正点拨和全程指导，才使本书的创作思路不断提升、案例框架不断优化，并最终顺利完稿。另外，还要感谢我的家人，正是由于他们的理解与默默支持，使我全心写作，顺利完成本书的编写。

陶国荣

2012年6月

[1] 华章网站：www.hzbook.com——编辑注

第1章　初识jQuery Mobile

本章内容

jQuery Mobile简介

如何获取jQuery Mobile

jQuery Mobile应用开发迅速上手

本章小结

在当前的网络应用中，最热的莫过于移动互联网的产品，特别是随着HTML 5的渐进发展、3G网络的逐步成熟、其他互联网应用的日趋饱和，移动互联网将代表下一阶段互联网发展的一个方向。许多可供移动设备终端下载的应用，无须下载或升级，直接通过浏览器登录即可使用。

如果通过移动设备终端的浏览器登录网站直接使用产品或应用，那么，面临的最大问题就是各移动终端设备浏览器的兼容性，这些浏览器的种类比传统的PC端还要多，且调试更为复杂。解决这些兼容性问题、开发出一个可以跨移动平台的应用，需要引入一个优秀、高效的Web脚本框架——jQuery Mobile。

确切来说，jQuery Mobile是专门针对移动终端设备的浏览器开发的Web脚本框架，它基于强悍的jQuery和jQuery UI基础之上，统一用户系统接口，能够无缝隙运行于所有流行的移动平台之上，并且易于主题化地设计与建造，是一个轻量级的Web脚本框架。它的出现，打破了传统JavaScript对移动终端设备的脆弱支持的局面，使开发一个跨移动平台的Web应用真正成为可能。

本书所有实例使用的jQuery Mobile框架都是基于近期最新发布的1.0.1版本，而jQuery框架为1.6.4版本。接下来将通过本书各章节的介绍，逐步带领大家进入jQuery Mobile的精彩世界。

1.1　jQuery Mobile简介

jQuery是一个非常流行的Web程序开发时使用的JavaScript类库，但它只是为PC端的浏览器而设计的。在移动互联网中为了满足浏览器更好地运行Web程序的需求，在基于jQuery与jQuery UI的基础之上，推出了jQuery Mobile这套框架，其主旨就是在进行移动项目开发的过程中，为开发者提供统一的接口与特征，依靠强大的jQuery类库，节省JavaScript代码的开发时间，提高项目开发的效率。

1.1.1　功能特点

jQuery Mobile为开发移动应用程序提供十分简单的应用接口，而这些接口的配置则是由标记驱动的，开发者在HTML页中无须使用任何JavaScript代码，就可以建立大量的程序接口。使用页面元素标记驱动是jQuery Mobile仅是它众多特点之一，概括而言，它具有如下特点：

特点1　强大的AJAX驱动导航

无论页面数据的调用还是页面间的切换，都是采用AJAX进行驱动的，从而保持了动画转换页面的干净与优雅。

特点2　以jQuery与jQuery UI为框架核心

jQuery Mobile的核心框架是建立在jQuery基础之上的，并且利用了jQuery UI的代码与运用模式，使熟悉jQuery语法的开发者能通过最小的学习曲线迅速掌握。

特点3　强大的浏览器兼容性

jQuery Mobile继承了jQuery的兼容性优势，目前所开发的应用兼容于所有主要的移动终端浏览器，使用开发者集中精力做功能开发，而不需要考虑复杂的浏览兼容性问题。

特点4　框架轻量级

目前jQuery Mobile最新的稳定版本为1.0.1，压缩后的体积大小为24KB，与之相配套的CSS文件压缩后的体积大小为6KB，框架的轻量级将大大加快程序执行时的速度。

特点5　支持触摸与其他鼠标事件

jQuery Mobile提供了一些自定义的事件，用来侦测用户的移动触摸动作，如tap（单击）、tap-and-hold（单击并按住）、swipe（滑动）等事件，极大提高了代码开发的效率。

特点6　强大的主题化框架

借助于主题化的框架和ThemeRoller应用程序，jQuery Mobile可以快速地改变应用程序的外观或自定义一套属于产品自身的主题，有助于树立应用产品的品牌形象。

1.1.2 支持平台

目前jQuery Mobile 1.0.1版本支持绝大多数的台式机、智能手机、平板和电子阅读器的平台，此外，对有些不支持的智能手机与旧版本的浏览器，通过渐进增强的方法，将逐步实现能够完全支持。下面通过一个分级的支持系统，详细说明各浏览器对jQuery Mobile 1.0.1的支持状况。

A（最优）

苹果iOS 3.2～5.0-最早的iPad（4.3/5.0）、iPad 2（4.3），最早的iPhone（3.1）、iPhone 3（3.2）、iPhone 3GS（4.3）和iPhone 4（4.3/5.0）

安卓2.1～2.3-HTC（2.2）、最早的Droid（2.2）、Nook Color（2.2）、HTC Aria（2.1）、谷歌Nexus S（2.3）

安卓Honeycomb-三星Galaxy Tab 10.1和摩托罗拉XOOM

Windows Phone 7～7.5-HTC Surround（7.0）、HTC Trophy（7.5）和LG-E900（7.5）

黑莓6.0-Torch 9800和Style 9670

黑莓7-BlackBerry® Torch 9810

黑莓Playbook-PlayBook版本1.0.1/1.0.5

Palm WebOS（1. 4～2.0）-Palm Pixi（1.4）、1.4前版本（1.4）、2.0前版本（2.0）

Palm WebOS 3. 0-HP触摸板

Firebox Mobile（Beta）-安卓2.2

Opera Mobile 11. 0-安卓2.2

Meego 1. 2-Nokia 950和N9机型

Kindle 3和Fire-内置的每个WebKit浏览器

Chrome（11～15）桌面浏览器-基于OS X 10.6.7和Windows 7操作系统

Firefox（4～8）桌面浏览器-基于OS X 10.6.7和Windows 7操作系统

Internet Explorer（7～9）-Windows XP、Vista和Windows 7（有轻微的CSS错误）

Opera（10～11）桌面浏览器-基于OS X 10.6.7和Windows 7操作系统

B（良好）

黑莓5.0-Storm 2 9550和Bold 9770

Opera Mini（5. 0～6.0）-基于iOS 3.2/4.3操作系统

诺基亚Symbian^3-诺基亚N8（Symbian^3）、C7（Symbian^3）、N97（Symbian^1）机型

C（较差）

黑莓4.x-Curve 8330

Windows Mobile-HTC Leo（WInMo 5. 2）

所有版本较老的智能手机平台将都不支持

说明　浏览支持系统为三个级别，分别为A、B、C。A级表示完全基于AJAX的动画页面转换增加的体验效果，代表最优；B级表示仅是除了没有AJAX的动画页面转换增加的体验效果，其他都可以很好地支持，代表良好；C级表示能够支持实现基本的功能，没有体验效果，代表较差。

1.1.3 与jQTouch、Sencha Touch、SproutCore的比较

移动Web开发有易于上手、开发周期相对短以及可以自动更新等众多的优点，因此，除jQuery Mobile外，还有很多框架可支持开发Web应用，如jQTouch、Sencha Touch、SproutCore等。那它们与jQuery Mobile有什么区别呢？接下来我们进行详细说明。

1. jQTouch

jQTouch与jQuery Mobile十分相似，也是一个jQuery插件，同样也支持HTML页面标签驱动，实现移动设备视图切换效果。但与jQuery Mobile不同在于，它是专为WebKit内核的浏览器打造的，可以借助该浏览器的专有功能对页面进行渲染；此外，开发时所需的代码量更少。如果所开发的项目中，目标用户群都使用WebKit内核的浏览器，可以考虑此框架。

官方下载地址：http：//www.jqtouch.com/

2. Sencha Touch

Sencha Touch是一套基于ExtJS开发的插件库。它与jQTouch相同，也是只针对WebKit内核的浏览器开发移动应用，拥有众多效果不

错的页面组件和丰富的数据管理，并且全部基于最新的HTML 5与CSS 3的Web标准。与jQuery Mobile不同之处在于，它的开发语言不是基于HTML标签，而是类似于客户端的MVC风格编写JavaScript代码，相对来说，学习周期较长。

官方下载地址：http：//www.sencha.com/products/touch/

3.SproutCore

SproutCore同样也是一款开源的JavaScript框架，以少量的代码开发强大的Web应用。

开始仅用于桌面浏览器的应用开发，后来，由于功能强大，许多知名的厂商也纷纷使用它来开发移动Web应用。但与jQuery Mobile相比，SproutCore对一些主流终端浏览的支持还有许多不足之处，如屏幕尺寸略大，开发代码相对复杂些。

官方下载地址：http：//www.sproutcore.com/

1.2 如何获取jQuery Mobile

想在浏览器中正常运行一个jQuery Mobile移动应用页面，需要先获取与jQuery Mobile相关的插件文件。其获取的方法有两种，分别为下载相关插件文件和使用URL方式加载相应文件。

1.2.1 下载插件文件

要运行jQuery Mobile移动应用页面需要包含3个文件，分别为：

jQuery-1. 6.4.min.js：jQuery主框架插件，目前稳定版本为1.6.4；

jQuery. Mobile-1.0.1.min.js：jQuery Mobile框架插件，目前最新版本为1.0.1；

jQuery. Mobile-1.0.1.min.css：与jQuery Mobile框架相配套的CSS样式文件，最新版本为1.0.1。

登录jQuery Mobile官方网站（http：//jquerymobile.com），单击导航条中的“Download”链接进入文件下载页面，如图1-1所示。

图　1-1　jQuery Mobile资源下载页

在图1-1的jQuery Mobile下载页中，可以下载上述3个必需文件中的任意一个，也可以单击下载地址（http：//code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.zip），获取jQuery Mobile页面执行所需的全部文件（包含压缩前后的JavaScript与CSS样式）和实例文件。

1.2.2　使用URL方式加载插件文件

除在jQuery Mobile下载页下载对应文件外，jQuery Mobile还提供了URL方式从jQuery CDN下载插件文件。CDN的全称是Content Delivery Network，用于快速下载跨Internet常用的文件，只要在页面的＜head＞元素中加入下列代码，同样可以执行jQuery Mobile移动应用页面。加入的代码如下所示：

```
    ＜link
rel="stylesheet"href="http: //code.jquery.com/mobile/1.0.1/jquery.mobile-
    1.0.1.min.css"/＞
    ＜script src="http: //code.jquery.com/jquery-
1.6.4.min.js"＞＜/script＞
    ＜script
src="http: //code.jquery.com/mobile/1.0.1/jquery.mobile-
1.0.1.min.js"＞＜/script＞
```

通过URL加载jQuery Mobile插件的方式使版本的更新更加及时，但由于是通过jQuery CDN服务器请求的方式进行加载，在执行页面时必须时时保证网络的畅通，否则，不能实现jQuery Mobile移动页面的效果。

1.3　jQuery Mobile应用开发迅速上手

jQuery Mobile的工作原理是：提供可触摸的UI小部件和AJAX导航系统，使页面支持动画式切换效果。以页面中的元素标记为事件驱动对象，当触摸或单击时进行触发，最后在移动终端的浏览器中实现一个个应用程序的动画展示效果。

与开发桌面浏览中的Web页面相似，构建一个jQuery Mobile页面也十分容易。jQuery Mobile通过＜div＞元素组织页面结构，根据元素的“data-role”属性设置角色。每一个拥有“data-role”属性的＜div＞元素就是一个容器，它可以放置其他的页面元素。接下来通过一个简单实例详细介绍如何开发第一个jQuery Mobile页面。

实例1-1　Hello World页面的实现

1.功能说明

使用HTML 5结构编写一个jQuery Mobile页面，在页面中输出“Hello World！”字样。

2.实现代码

新建一个HTML页面1-1.htm，加入代码如代码清单1-1所示。

代码清单1-1　Hello World页面的实现

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile应用程序</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div id="page1"data-role="page">
<div data-role="header"><h1>jQuery Mobile</h1></div>
<div data-role="content"class="content">
<p>Hello World! </p>
</div>
<div data-role="footer"><h1>荣拓工作室版权所有</h1>
</div>
</div>
</body>
</html>
```

3.页面效果

为了更好地在PC端浏览jQuery Mobile页面在移动终端的执行效果，可以下载Opera公司的移动模拟器Opera Mobile Emulator，下载地址：http：//cn.opera.com/developer/tools/mobile/，目前最新的版本为12.0。本书全部的页面效果都在Opera Mobile Emulator 12.0中演示。

该页面在Opera Mobile Emulator 12.0下执行的效果如图1-2所示。

图　1-2　Hello World页面

4.源码分析

在页面代码的＜head＞头部元素中，先通过＜meta＞标记的“content”属性设置页面的宽度与模拟器的宽度一致，以保证页面可以在浏览器中完全填充；接下来，导入了三个框架性文件，需要注意它们导入时的顺序。

在代码的＜body＞主体元素中，通过多个＜div＞元素进行层次的划分。因为在jQuery Mobile中每个＜div＞元素都是一个容器，根据

指定的“data-role”属性值，确定容器对应的身份，如果属性“data-role”的值为“header”，则该＜div＞元素的为头部区域。

“data-role”属性是HTML 5的一个新特征，通过设置该属性，jQuery Mobile就可以很快地定位到指定的元素，并对内容进行相应的处理。

由于jQuery Mobile已经全面支持HTML 5结构，因此，＜body＞主体元素的代码也可以修改为以下代码：

```
    ＜section id="page1"data-role="page"＞
    ＜header data-role="header"＞＜h1＞jQuery Mobile＜/h1＞
＜/header＞
    ＜div data-role="content"class="content"＞
    ＜p＞Hello World! ＜/p＞
    ＜/div＞
    ＜footer data-role="footer"＞＜h1＞荣拓工作室版权所有＜/h1＞
＜/footer＞
    ＜/section＞
```

上述代码执行后的效果与修改前完全相同。

实例1-2　多页面的切换

通过实例1-1可以知道，在jQuery Mobile中，如果将页面元素的“data-role”属性值设置为“page”，则该元素成为一个容器，即页面的某块区域。在一个页面中，可以设置多个元素成为容器，虽然元

素的“data-role”属性值都为“page”，但它们对应的Id号不允许相同。

在jQuery Mobile中，将一个页面中的多个容器当作多个不同的页面，它们之间的界面切换是通过增加一个＜a＞元素、并将该元素的“href”属性值设为“#”加对应Id号的方式来进行。下面通过一个简单实例来说明多页面切换实现的过程。

1.功能说明

使用HTML 5结构编写一个jQuery Mobile页面，在页面中设置两个“page”区域，当单击第一个区域中的“详细页”链接时，进入第二个区域；在第二个区域中，单击“返回首页”链接时，则切换至第一个区域。

2.实现代码

新建一个HTML页面1-2.htm，加入代码如代码清单1-2所示。

代码清单1-2　多页面的切换

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞jQuery Mobile应用程序＜/title＞
＜meta name="viewport"content="width=device-width,
initial-scale=1"/＞
＜link href="Css/jquery.mobile-1.0.1.min.css"
```

```
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <section id="page1"data-role="page">
    <header data-role="header"><h1>jQuery Mobile</h1>
</header>
    <div data-role="content"class="content">
    <p>这是首页</p>
    <p><a href="#page2">详细页</a></p>
    </div>
    <footer data-role="footer"><h1>荣拓工作室版权所有</h1>
</footer>
    </section>
    <section id="page2"data-role="page">
    <header data-role="header"><h1>jQuery Mobile</h1>
</header>
    <div data-role="content"class="content">
    <p>这是详细页</p>
    <p><a href="#page1">返回首页</a></p>
    </div>
    <footer data-role="footer"><h1>荣拓工作室版权所有</h1>
</footer>
    </section>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图1-3所示。

图　1-3　多页面间的切换

4.源码分析

在jQuery Mobile中，针对一个页面中多个“page”区域间的切换称为内链接，其方式为添加一个＜a＞元素，并将该元素的“href”属性值设置为“#”加对应“Id”形式，如下所示：

```
＜a href="#page2"＞详细页＜/a＞
```

上述代码表示，单击“详细页”链接时，将切换到Id号为“page2”的区域中。除内链接外，还有一个外链接。所谓外链接指的是通过单击页面中的某个链接，跳转到另外一个页面中，而不是在同一个页面间的切换。其实现的方式与内链接相同，只是在＜a＞元素中

增加了另外一个“rel”属性，并将该属性值设置为“external”表示外链接，如下所示：

```
<a href="a1.htm"rel="external">详细页</a>
```

上述代码表示，单击“详细页”链接时，将跳转至文件名为a1的HTML页面。在页面切换过程中，无论是内链接还是外链接，jQuery Mobile都支持动画的效果，只需要在切换的<a>元素中，添加一个“data-transition”属性，并选择某种属性值即可，如下所示：

```
<a href="a1.htm"rel="external"data-transition="pop">详细页</a>
```

上述代码表示，单击“详细页”链接时，将以弹出的动画效果跳转至文件名为“a1”的HTML页面。更多<a>元素的“data-transition”属性值如表1-1所示。

表 1-1 <a> 元素的“data-transition”属性值

值	描　述	默认值
slide	从右到左滑动的动画效果	是
pop	以弹出的效果打开页面	否
slideup	向上滑动的动画效果	否
slidedown	向下滑动的动画效果	否
fade	渐变退色的动画效果	否
flip	旧页飞出，新页飞入的动画效果	否

说明　当页面进行切换时，切换前的页面将自动隐藏，链接的区域或页面自动展示在当前页面中。如果是内链接，仅显示指定Id号并

且“data-role”属性值为“page”的区域，其他范围都隐藏。

1.4　本章小结

本章先从功能特点、支持平台方面对jQuery Mobile进行详细介绍，然后介绍jQuery Mobile的获取方法与工作原理，最后通过两个简单完整的开发实例的介绍，使读者对jQuery Mobile开发移动应用程序有一个初步的了解，为下一章的学习奠定基础。

第2章　页面与对话框

本章内容

jQuery Mobile页面结构

预加载与页面缓存

页面的脚本

对话框

本章小结

在第1章中，我们通过两个简单实例的介绍，初步了解了jQuery Mobile的工作原理和执行流程，本章将继续深入剖析jQuery Mobile的页面结构。

从页面结构上来说，jQuery Mobile可以支持单个页面和一个页面中的多个“page”容器。这种结构模型的好处在于，可以使普通的链接标记不需要任何复杂配置就可以优雅地工作，并且可以很方便地使一些富媒体应用本地化。

另外，在jQuery Mobile页面中，通过AJAX功能可以很方便地自动读取外部页面，支持使用一组动画的效果进行页面间的相互切换；也可以通过调用对应的脚本函数，实现预加载、缓存、创建、跳转页面的功能；同时，支持将页面以对话框的形式展示在移动终端的浏览器中。接下来我们逐一进行详细的介绍。

2.1　jQuery Mobile页面结构

jQuery Mobile的许多功能效果需要借助于HTML 5的新增标记和属性，因此，页面必须以HTML 5的声明文档开始，在＜head＞标记中分别依次导入jQuery Mobile的样式文件、jQuery基础框架文件和jQuery Mobile插件文件，下面看一个jQuery Mobile的基本页面结构。

2.1.1　基本框架

在jQuery Mobile中，有一个基本的页面框架模型，就是在页面中通过将一个＜div＞标记的“data-role”属性设置为“page”，形成一个容器或视图；而在这个容器中最直接的子节点应该就是“data-role”属性为“header”、“content”、“footer”3个子容器，分别形成了“标题”、“内容”、“页脚”3个组成部分，用于容纳不同的页面内容。接下来通过一个简单实例进行展示。

实例2-1　jQuery Mobile基本的页面框架

1.功能说明

创建一个jQuery Mobile的基本框架页，并在页面组成部分中分别显示其对应内容名称。

2.实现代码

新建一个HTML页面2-1.htm，加入代码如代码清单2-1所示。

代码清单2-1　jQuery Mobile基本页面框架

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile基本页面框架</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>标题</h1></div>
<div data-role="content"><p>内容部分</p></div>
<div data-role="footer"><h4>页脚</h4></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-1所示。

图　2-1　jQuery Mobile基本页面展示

4.源码分析

本实例源码中，为了更好地支持HTML 5的新增加功能与属性，第一行以HTML 5的声明文档开始，即添加如下代码：

```
<! DOCTYPE html>
```

在<head>元素中添加一个名称为“viewport”的<meta>元素，并设置该元素的“content”属性，代码如下所示：

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

这行代码的功能是：设置移动设备中浏览器缩放的宽度与等级。通常情况下，移动设备的浏览器默认以“900px”的宽度显示页面，这种宽度会导致屏幕缩小，页面放大，不适合浏览。如果在页面中添加＜meta＞元素，并设置“content”的属性值为“width=device-width, initial-scale=1”，可以使页面的宽度与移动设备的屏幕宽度相同，更加适合用户浏览。

在接下来的＜body＞元素中，将第一个＜div＞元素的“data-role”属性设置为“page”，形成一个容器；然后，在容器中分别添加3个＜div＞元素，依次将“data-role”属性设置为“header”、“content”、“footer”，从而形成了一个标准的jQuery Mobile页面框架。详细实现过程如代码中加粗部分所示。

2.1.2　多容器页面结构

在一个供jQuery Mobile使用的HTML页面中，可以包含一个元素属性“data-role”值为“page”的容器，也允许包含多个，从而形成多容器页面结构。容器之间各自独立，拥有唯一的Id号属性。页面加载时，以堆栈的方式同时加载；容器访问时，以内部链接“#”加对应“Id”的方式进行设置。单击该链接时，jQuery Mobile将在页面文档寻找对应Id号的容器，以动画效果切换至该容器中，实现容器间内容的访问。

实例2-2　jQuery Mobile多容器间的切换

1.功能说明

新建一个HTML页面，并在页面中添加2个“data-role”属性为“page”的＜div＞元素，作为2个页面容器。用户在第一个容器中选择需要查看天气预报的日期，单击某天后，切换至第二个容器，显示所选日期的详细天气情况。

2.实现代码

新建一个HTML页面2-2.htm，加入代码如代码清单2-2所示。

代码清单2-2　jQuery Mobile多容器间的切换

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile多容器页面结构</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>天气预报</h1></div>
<div data-role="content">
<p><a href="#w1">今天</a>|<a href="#">明天</a></p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="w1"data-add-back-btn="true">
<div data-role="header"><h1>今天天气</h1></div>
<div data-role="content">
<p>4～-7℃<br/>晴转多云<br/>微风</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-2所示。

图　2-2　页面中多容器间的切换

4.源码分析

在本实例页面中，从第一个容器切换至第二个容器时，采用的是“#”加对应“Id”的内部链接方式。因此，在一个页面中，不论相同框架的“page”容器有多少，只要对应的Id号唯一，就可以通过内部链接的方式进行容器间的切换。在切换时，jQuery Mobile会在文档中寻找对应“Id”的容器，然后通过动画的效果切换到该页面中。

从第一个容器切换至第二个容器后，如果想要从第二个容器返回第一个容器，有下列两种方法：

在第二个容器中，增加一个＜a＞元素，通过内部链接“#”加对应“Id”的方式返回第一个容器。

在第二个容器的最外层框架＜div＞元素中，添加一个“data-add-back-btn”属性。该属性表示是否在容器的左上角增加一个“回退”按钮，默认值为“false”；如果设置为“true”，将出现一个“back”按钮，单击该按钮，回退上一级的页面显示。

说明　如果是在一个页面中，通过“#”加对应“Id”的内部链接方式，可以实现多容器间的切换；但如果不是在一个页面，此方法将失去作用。因为在切换过程中，先需要找到页面，再去锁定对应“Id”容器的内容，而并非直接根据“Id”切换至容器中。

2.1.3　外部页面链接

虽然在一个页面中，可以借助容器的框架实现多种页面的显示，但是，把全部代码写在一个页面中会延缓页面被加载的时间，使代码冗余，且不利于功能的分工与维护的安全性。

因此，在jQuery Mobile中，可以采用开发多个页面并通过外部链接的方式，实现页面相互切换的效果。下面通过一个简单的实例来介绍它是如何实现的。

实例2-3　jQuery Mobile外部页面链接切换

1.功能说明

在实例2-2的基础上，在Id号为“w1”的第二个容器中添加一个＜em＞元素。在该元素中显示“rttop.cn提供”字样。单击“rttop.cn”文本链接时，将以外部页面链接的方式加载一个名为“about.htm”的HTML页面。

2.实现代码

新建一个HTML页面2-3.htm，加入代码如代码清单2-3所示。

代码清单2-3　jQuery Mobile外部页面链接切换

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile外部页面链接</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>天气预报</h1></div>
<div data-role="content">
<p><a href="#w1">今天</a>|<a href="#">明天</a></p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="w1"data-add-back-btn="true">
<div data-role="header"><h1>今天天气</h1></div>
<div data-role="content">
<p>4～-7℃<br/>晴转多云<br/>微风</p>
<em style="foat: right; padding-right: 5px">
<a href="about.htm">rttop.cn</a>提供
</em>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

另外，新建一个用于外部链接的HTML页面about.htm，加入以下代码：

```
<! DOCTYPE html>
<html>
```

```
    <head>
    <title>关于rttop</title>
    <meta name="viewport"content="width=device-width"/>
    </head>
    <body>
    <div data-role="page"data-add-back-btn="true">
    <div data-role="header"><h1>关于rttop</h1></div>
    <div data-role="content">
    <p>
    rttop.cn是一家新型高科技企业，正在努力实现飞翔的梦想。
    </p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-3所示。

图　2-3　外部页面链接的切换效果

4.源码分析

在jQuery Mobile中，如果单击一个指向外部页面的超级链接（如about.htm），jQuery Mobile将自动分析该URL地址，自动产生一个AJAX请求。在请求过程中，会弹出一个显示进度的提示框。如果请求成功，jQuery Mobile将自动构建页面结构，注入主页面的内容；同时，初始化全部的jQuery Mobile组件，将新添加的页面内容显示在浏览器中；如果请求失败，jQuery Mobile将弹出一个错误信息提示框，数秒后该提示框自动消失，页面也不会刷新。

如果不想采用AJAX请求的方式打开一个外部页面，只需要在链接元素中将“rel”属性设置为“external”，该页面将脱离整个jQuery Mobile的主页面环境，以独自打开的页面效果在浏览器中显示。

说明　如果采用AJAX请求的方式打开一个外部页面，注入主页面的内容也是以“page”为目标，“page”以外的内容将不会被注入主页面中；另外，必须确保外部加载页面URL地址的唯一性。

2.1.4　页面后退链接

上一节介绍了将“page”容器的“data-add-back-btn”属性设置为“true”可以后退至上一页，在jQuery Mobile页面中，还可以添加一个<a>元素，将该元素的“data-rel”属性设置为“back”，同样可以实现后退至上一页的功能。因为一旦该链接元素的“data-rel”属性设置为“back”，单击该链接将被视为后退行为，并且将忽视“href”属性的URL值，直接退回至浏览器历史的上一页面。

实例2-4　jQuery Mobile页面后退链接

1.功能说明

在新建的HTML中，添加2个“page”容器，当单击第一个容器中的“测试后退链接”链接时，切换到第二个容器；单击第二个容器中的“返回首页”链接时，将以回退的方式返回到第一个容器中。

2.实现代码

新建一个HTML页面2-4.htm，加入代码如代码清单2-4所示。

代码清单2-4　jQuery Mobile页面后退链接

```
<! DOCTYPE html>
<html>
```

```
<head>
<title>jQuery Mobile页面后退链接</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>测试</h1></div>
<div data-role="content">
<p><a href="#e">测试后退链接</a></p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="e">
<div data-role="header"><h1>测试</h1></div>
<div data-role="content">
<p>
<a href="http: //www.rttop.cn"data-rel="back">
返回首页
</a>
</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-4所示。

图　2-4　页面的后退功能

4.源码分析

在本实例的第二个“page”容器中，用户在容器中单击“返回首页”时，可以后退到上一页，方法是在添加＜a＞元素时，将“data-rel”属性设置为“back”，表明任何的单击操作都被视为回退动作，并且忽视元素“href”属性值设置的URL地址，只是直接回退到上一个历史记录页面；这种页面切换的效果可以用于关闭一个打开的对话框或页面。

说明　在设置回退链接属性时，除将“data-rel”属性设置为“back”外，还要尽量将“href”属性设置为一个可以访问的正确URL

地址，以确保更多的浏览器可以单击该链接按钮。

2.2　预加载与页面缓存

通常情况下，移动终端设备的系统配置要低于PC终端，因此，在开发移动应用程序时，更要注意页面在移动终端浏览器中加载时的速度。如果速度过慢，用户的体验将会大打折扣。为了加快页面移动终端访问的速度，在jQuery Mobile中，使用预加载与页面缓存都是十分有效的方法。当一个被链接的页面设置好预加载后，jQuery Mobile将在加载完成当前页面后自动在后台进行预加载设置的目标页面；另外，使用页面缓存的方法，可以将访问过的“page”容器都缓存到当前的页面文档中，下次再访问时，将可以直接从缓存中读取，而无需再重新加载页面。

2.2.1　预加载

在开发移动应用程序时，对需要链接的页面进行预加载是十分有必要的。因为当一个链接的页面设置成预加载方式时，在当前页面加载完成之后，目标页面也被自动加载到当前文档中，用户单击就可以马上打开，大大加快了页面访问的速度。

实例2-5　jQuery Mobile页面预加载

1.功能说明

在新建的HTML页面中添加一个＜a＞元素，将该元素的“href”属性值设置为“about.htm”，并将“data-prefetch”属性值设置为“true”，表示预加载＜a＞元素的链接页面。

2.实现代码

新建一个HTML页面2-5.htm，加入代码如代码清单2-5所示。

代码清单2-5　jQuery Mobile页面预加载

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞jQuery Mobile页面预加载＜/title＞
＜meta name="viewport"content="width=device-width，
initial-scale=1"/＞
＜link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/＞
＜script src="Js/jquery-1.6.4.js"
type="text/javascript"＞＜/script＞
＜script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"＞＜/script＞
＜/head＞
＜body＞
＜div data-role="page"＞
＜div data-role="header"＞＜h1＞预加载页＜/h1＞＜/div＞
＜div data-role="content"＞
＜p＞
＜a href="about.htm"data-prefetch="true"＞点击进入＜/a＞
＜/p＞
＜/div＞
＜div data-role="footer"＞＜h4＞©2012 rttop.cn studio＜/h4
＞＜/div＞
＜/div＞
＜/body＞
＜/html＞
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-5所示。

图　2-5　使用预加载方式注入链接页面

4.源码分析

从图2-5可以很清楚地看到，＜a＞元素链接的目标页面“about.htm”中，“page”容器的内容已经通过预加载的方式注入当前文档中。

在jQuery Mobile中，想要实现页面的预加载，方法有两种，如下所示：

在需要链接页面的元素中添加“data-prefetch”属性，设置属性值为“true”或不设置属性值均可；设置完该属性值后，jQuery Mobile将在加载完成当前页面以后，自动加载该链接元素所指的目标页面，即“href”属性的值。

调用JavaScript代码中的全局性方法$.mobile.loadPage（）来预加载指定的目标HTML页面，其最终的效果与设置元素的“data-prefetch”属性一样，详细的实现过程将在2.3.3节进行介绍。

说明　无论是添加元素的“data-prefetch”属性，还是使用全局性方法$.mobile.loadPage（）在实现页面的预加载功能时，都允许同时加载多个页面。但在进行预加载的过程中需要加大页面HTTP的访问请求，这可能会延缓页面访问的速度，因此，该功能需要有选择性地使用。

2.2.2　页面缓存

在jQuery Mobile中，可以通过页面缓存的方式将访问过的历史内容写入页面文档的缓存中；当用户重新访问时，不需要重新加载，只要从缓存中读取就可以。

实例2-6　jQuery Mobile页面缓存

1.功能说明

新建一个HTML页面，在内容区域中显示“这是一个被缓存的页面”文字。将“page”容器的“data-dom-cache”属性设置为“true”，可以将该页面的内容注入文档的缓存中。

2.实现代码

新建一个HTML页面2-6.htm，加入代码如代码清单2-6所示。

代码清单2-6　jQuery Mobile页面缓存

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile页面缓存</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
```

```
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script type="text/javascript">
    $（function（）{
    $.mobile.page.prototype.options.domCache=true;
    }）
    </script>
    </head>
    <body>
    <div data-role="page"data-dom-cache="true">
    <div data-role="header"><h1>缓存页面</h1></div>
    <div data-role="content">
    <p>这是一个被缓存的页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-6所示。

图　2-6　将页面的内容注入文档缓存中

4.源码分析

本实例通过添加“page”容器的“data-dom-cache”属性，将对应容器中的全部内容写入缓存中。一般说来，如果需要将页面的内容写入文档缓存中，有以下两种方式：

在需要被缓存的元素属性中添加一个“data-dom-cache”属性，设置该属性值为“true”或不设置属性值均可。该属性的功能是将对应的元素内容写入缓存中。

通过JavaScript代码设置一个全局性的jQuery Mobile属性值为“ture”，即添加代码

“$.mobile.page.prototype.options.domCache=true”，可以将当前文档写入缓存中。

说明　使用页面缓存的功能将会使DOM内容变大，可能导致某些浏览器打开的速度变得缓慢。因此，一旦选择了开启使用缓存功能，就要管理好缓存的内容，并做到及时清理。

2.3　页面的脚本

jQuery Mobile中通过AJAX请求的方式加载页面，在编写页面脚本时，需要与PC端开发页面区分开。通常情况下，页面在初始化时会触发pagecreate事件，在该事件中可以做一些页面组件初始化的动作；如果需要通过JavaScript代码改变当前的工作页面，可以调用jQuery Mobile中提供的changePage（）方法，此外，还可以调用loadPage（）方法来加载指定的外部页面，注入当前文档中。接下来我们逐一对这些常用的页面脚本中的事件与方法进行介绍。

2.3.1　创建页面

在jQuery Mobile中，页面是被请求后注入当前的DOM结构中，因此，jQuery中提及的$（document）. ready（）事件在jQuery Mobile中不会被重复执行，只有在初始化加载页面时才会被执行一次。如果要跟踪不同页面的内容注入当前的DOM结构，可以将页面中的“page”容器绑定pagecreate事件。该事件在页面初始化时触发，绝大多数的jQuery Mobile组件都在该事件之后进行一些数据的初始化。

实例2-7　jQuery Mobile中的pagecreate事件

1.功能说明

新建一个HTML页面，添加一个Id号为“e1”的“page”容器，并将该容器与pagebeforecreate和pagecreate事件进行绑定。在页面执行时，通过绑定的事件跟踪执行的过程。

2.实现代码

新建一个HTML页面2-7.htm，加入代码如代码清单2-7所示。

代码清单2-7　jQuery Mobile中的pagecreate事件

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile中的pagecreate事件</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
$("#e1").live("pagebeforecreate", function(){
alert("正在创建页面！");
})
$("#e1").live("pagecreate", function(){
alert("页面创建完成！");
})
</script>
</head>
<body>
<div data-role="page"id="e1">
<div data-role="header"><h1>创建页面</h1></div>
<div data-role="content">
<p>页面创建完成！</p>
</div>
```

```
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-7所示。

图　2-7　页面中的pagebeforecreate与pagecreate事件

4.源码分析

在本实例中，Id号为“e1”的“page”容器绑定了pagebeforecreate和pagecreate两个事件。pagebeforecreate事件早

于pagecreate事件，即在页面被加载、jQuery Mobile组件开始初始化前触发。通常在这一事件中，可以添加一些页面加载的动画提示效果，直到pagecreate事件触发时动画效果结束。

在本实例的JavaScript代码中，既可以使用live（）方法绑定元素触发的事件，还可以使用bind（）与delegate（）方法为绑定的元素添加指定的事件。

2.3.2　跳转页面

如果使用JavaScript代码切换当前显示的页面，可以调用jQuery Mobile中的changePage（）方法。该方法可以设置跳转页面的URL地址、跳转时的动画效果和需要携带的数据，接下来通过一个简单的实例详细说明该方法的使用过程。

实例2-8　使用changePage（）方法跳转页面

1.功能说明

新建一个HTML页面，在页面中显示“页面正在跳转中……”文字，然后调用changePage（）方法，从当前页以“slideup”的动画切换效果跳转到“about.htm”页面。

2.实现代码

新建一个HTML页面2-8.htm，加入代码如代码清单2-8所示。

代码清单2-8　jQuery Mobile中的changePage（）方法

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile跳转页面</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script type="text/javascript">
    $(function(){
    $.mobile.changePage("about.htm",
    {transition: "slideup"});
    })
    </script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header"><h1>跳转页面</h1></div>
    <div data-role="content">
    <p>页面正在跳转中……</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-8所示。

图　2-8　使用changePage（）方法动态跳转页面

4. 源码分析

在本实例中，由于changePage（）方法在页面加载时被执行，因此，在浏览主页面时，便直接跳转至目标页“about.htm”；使用changePage（）方法除了可以跳转页面外，还能携带数据传递给跳转的目标页，如下面代码所示：

```
$.mobile.changePage（"login.php",
{type: "post",
data: $（"form#login"）.serialize（）
},
"pop", false, false
）
```

上述代码表示：将Id号为“login”的表单数据进行序列化后，传递给“login.php”页面进行处理。另外，“pop”表示跳转时的页面效果，第一个“false”值表示跳转时的方向，如果为“true”则表示反方向进行跳转，默认值为“false”；第二个“false”值表示完成跳转后是否更新历史浏览记录，默认值为“true”，表示更新。

说明　当指定跳转的目标页面不存在或传递的数据格式不正确时，都会在当前页面出现一个错误信息提示框，几秒钟后自动消失，不影响当前页面的内容显示。

2.3.3　加载页面

在2.2.1节中，通过添加元素的“data-prefetch”属性，实现预加载指定链接页面的功能；除此之外，如果想使用JavaScript方法动态加载任意指定的页面，可以调用jQuery Mobile提供的loadPage（）公共方法，其实现的最终效果与设置元素的属性一样，都可以使当前页在加载完成后自动将目标页面注入至DOM中。

实例2-9　使用loadPage（）方法加载页面

1.功能说明

新建一个HTML页面，调用jQuery Mobile中的loadPage（）方法加载“about.htm”页面。加载完成后，单击显示的链接字符，可以切换至“about.htm”页面。

2.实现代码

新建一个HTML页面2-9.htm，加入代码如代码清单2-9所示。

代码清单2-9　使用loadPage（）方法加载页面

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile加载页面</title>
```

```
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script type="text/javascript">
    $(function(){
    $.mobile.loadPage("about.htm");
    })
    </script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header"><h1>加载页面</h1></div>
    <div data-role="content">
    <p>页面已加载成功!<a href="about.htm">点击</a></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-9所示。

图　2-9　使用loadPage（）方法加载指定页面

4.源码分析

在本实例中，通过调用jQuery Mobile中提供的loadPage（）方法，可以将任意页面加载到当前的DOM中。执行完该方法后，pagecreate事件将会被重新触发，因为整个DOM的结构发生了变化，指定的目标页面已注入当前文档中，这时可以通过查看当前页面的源代码查找被加载的目标页面内容，如图2-9所示。

2.4　对话框

jQuery Mobile中创建对话框的方式十分方便，只需要在指向页面的链接元素中添加一个“data-rel”属性，并将该属性值设置为“dialog”。单击该链接时，打开的页面将以一个对话框的形式展示在浏览器中。单击对话框中的任意链接时，打开的对话框将自动关闭，并以“回退”的形式切换至上一页。此外，还可以在对话框中创建一个“取消”按钮，通过设置元素属性或编写JavaScript代码的方式关闭当前打开的对话框。

2.4.1　创建简单对话框

将链接元素的“data-rel”属性值设置为“true”，打开的对话框实际上是一个标准的“page”容器。因此，在打开时，也可以通过设置“data-transition”属性值，选择打开对话框时切换页面的动画效果。下面通过一个简单的实例来说明如何创建一个简单对话框。

实例2-10　jQuery Mobile中创建对话框

1.功能说明

新建一个HTML页面，在页面中添加一个＜a＞元素，并将该元素的“data-rel”属性值设置为“dialog”，表示以对话框的形式打开链

接元素指定的目标URL地址。

2.实现代码

新建一个HTML页面2-10.htm，加入代码如代码清单2-10所示。

代码清单2-10　jQuery Mobile中创建对话框

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile打开对话框</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="e1">
<div data-role="header"><h1>对话框</h1></div>
<div data-role="content">
<p>
<a href="dialog.htm"
data-rel="dialog"
data-transition="pop">打开对话框
</a>
</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

另外，创建一个对话框页面dialog.htm，加入代码如下所示：

```
<! DOCTYPE html>
<html>
<head>
<title>简单的对话框</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>主题</h1></div>
<div data-role="content">
<p>这是一个简单的对话框！</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-10所示。

图　2-10　在浏览器中打开标准对话框的效果

4.源码分析

本实例设置链接的“data-rel”属性值为“dialog”，通过该链接打开的页面将以对话框的形式展示在当前页面中。该对话框以模式的方式浮在当前页的上面，背景添加深色，四周是圆角的效果，左上角自带一个“×”关闭按钮，单击该按钮，对话框将关闭。

2.4.2　关闭对话框

在打开的对话框中，可以使用自带的“×”关闭按钮关闭打开的对话框，此外，在对话框内添加其他链接按钮，将该链接的“data-rel”属性值设置为“back”，单击该链接也可以实现关闭对话框的功能。下面通过一个简单的实例来说明如何创建一个可关闭的对话框。

实例2-11　jQuery Mobile中关闭对话框

1.功能说明

新建一个HTML页面，并添加一个＜a＞元素的链接，单击该链接时，将以对话框的形式弹出一个指定的页面，单击对话框中的“关闭”按钮，可以直接关闭打开的对话框。

2.实现代码

新建一个HTML页面2-11.htm，加入代码如代码清单2-11所示。

代码清单2-11　jQuery Mobile中关闭对话框

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞jQuery Mobile关闭对话框＜/title＞
＜meta name="viewport"content="width=device-width,
initial-scale=1"/＞
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header"><h1>对话框</h1></div>
    <div data-role="content">
    <p>
    <a href="close.htm"
    data-rel="dialog"
    data-transition="pop">关闭
    </a>
    </p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

另外，创建一个HTML页面close.htm，用于系统提示。加入代码如下所示：

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>系统提示</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>提示</h1></div>
    <div data-role="content">
    <p>真的要关闭弹出的对话框吗？</p>
    <p>
    <a href="#"
```

```
    data-role="button"
    data-rel="back"
    data-theme="a">关闭
    </a>
    </p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图2-11所示。

图　2-11　在浏览器中打开自带“关闭”按钮对话框

4.源码分析

本实例在对话框中将链接元素的“data-rel”属性设置为“back”，单击该链接将关闭当前打开的对话框。这种方法在不支持JavaScript代码的浏览器中，同样可以实现对应的功能；另外，编写JavaScript代码也可以实现关闭对话框的功能。代码如下所示：

```
$（'.ui-dialog'）.dialog（'close'）;
```

2.5 本章小结

本章首先从jQuery Mobile应用程序的基本页面结构讲起，通过一个个简单的实例开发，使读者逐步了解移动应用的基本框架和多容器页面的结构，以及如何实现链接外部页面与后退的方法。

在了解页面结构的基础上，本章进一步阐述实现页面预加载和页缓存的方法与技巧；另外，详细说明了jQuery Mobile页面中常用事件与方法的调用；最后，介绍了如何在jQuery Mobile中创建与关闭对话框。

通过本章节的学习，读者能够进一步了解与掌握jQuery Mobile基本框架与常用元素的使用技巧。

第3章　工具栏与格式化内容

本章内容

头部栏

导航栏

尾部栏

内容格式化

本章小结

jQuery Mobile中提供了一整套标准的工具栏组件，移动应用只需对元素添加相应的属性值，就可以直接调用。通常情况下，工具栏由移动应用的头部栏、导航栏、尾部栏三部分组成，分别放置在移动应用程序中的标题部分、内容部分、页尾部分，并通过添加不同样式和设定工具栏的位置，满足和实现各种移动应用的页面需求和效果。

除此之外，jQuery Mobile还提供了许多非常有用的工具组件。通过调用这些组件，开发人员无需编写任何代码，就可以很方便地对移动应用的页面内容实现折叠面板、网格布局等页面效果，极大地提高了项目开发的效率。

3.1　头部栏

头部栏是移动应用中工具栏的组成部分，用来说明该页面的主题内容。头部栏是“page”容器中的第一个元素，放置的位置十分重要。头部栏由页面标题和按钮（最多两个）组成，其中的按钮可以使用“后退”按钮，也可以添加表单元素中的按钮，并可以通过设置相关属性控制头部按钮的相对位置。接下来我们逐一进行详细介绍。

3.1.1　头部栏的基本结构

头部栏由标题文字和左右两边的按钮构成，标题文字通常使用＜h＞标记，取值范围在1～6之间，常用＜h1＞标记，无论取值是多少，在同一个移动应用项目中都要保持一致。标题文字的左右两边可以分别放置一或两个按钮，用于标题中的导航操作。下面通过一个简单实例展示移动应用中头部栏的基本结构。

实例3-1　头部栏的基本结构

1.功能说明

新建一个HTML页面，添加一个“page”容器，在容器中添加一个“data-role”属性为“header”的＜div＞元素作为头部栏，在头部栏中添加一个＜h1＞元素作为标题，标题内容设为“头部栏标题”。

2.实现代码

新建一个HTML页面3-1.htm，加入代码如代码清单3-1所示。

代码清单3-1　头部栏的基本结构

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile头部栏基本结构</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header">
<h1>头部栏标题</h1>
</div>
<div data-role="content">
<p>默认头部栏的特征</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-1所示。

图　3-1　头部栏的基本结构

4.源码分析

由于移动设备的浏览器分辨率不尽相同，如果尺寸过小，而头部栏的标题内容又很长时，jQuery Mobile会自动调整需要显示的标题内容，隐藏的内容以“……”的形式显示在头部栏中。

另外，头部栏默认的主题样式为“a”，如果要修改主题样式，只需要在头部栏标签中添加一个“data-theme”属性，设置对应的主题样式值即可。关于更多jQuery Mobile中的主题内容，本书的第5章会有详细的介绍。

3.1.2　设置后退按钮的文字

本书2.1.2节中曾介绍过，给“page”容器元素添加“data-add-back-btn”属性可以在头部栏的左侧增加一个默认名为“back”的后退按钮。此外，还可以通过修改“page”容器元素的“data-back-btn-text”属性值，设置后退按钮中显示的文字。

实例3-2　设置后退按钮的文字

1.功能说明

新建的HTML页面中，添加3个“page”容器，Id号分别为“e1”、“e2”、“e3”，分别用于显示“首页”、“下一页”、“尾页”内容。当切换到“下一页”时，头部栏的“后退”按钮文字为默认值“back”；切换到“尾页”时，头部栏的头部栏的“后退”按钮文字为“首页”。

2.实现代码

新建一个HTML页面3-2.htm，加入代码如代码清单3-2所示。

代码清单3-2　设置后退按钮的文字

```
<! DOCTYPE html>
<html>
```

```
    <head>
    <title>jQuery Mobile设置后退按钮的文字</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="e1"data-add-back-btn="true">
    <div data-role="header">
    <h1>后退按钮文字</h1>
    </div>
    <div data-role="content">
    <p><a href="#e2">下一页</a></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="e2"data-add-back-btn="true">
    <div data-role="header">
    <h1>后退按钮文字</h1>
    </div>
    <div data-role="content">
    <p><a href="#e3">尾页</a></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="e3"data-add-back-btn="true"
    data-back-btn-text="首页">
    <div data-role="header">
    <h1>后退按钮文字</h1>
    </div>
    <div data-role="content">
    <p><a href="#e1">首页</a></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-2所示。

图　3-2　设置后退按钮的文字

4.源码分析

在本实例中，首先将“page”容器元素的“data-add-back-btn”属性设置为“true”，表示切换到该容器时，头部栏显示默认的“back”按钮；然后，在“page”容器元素中添加另一个“data-back-btn-text”属性，用来显示后退按钮上的文字内容，可以根据需要进行手动修改。

此外，还可以编写JavaScript代码进行设置，在HTML页的＜head＞元素中，加入如下代码：

```
    $.mobile.page.prototype.options.backBtnText="后退按钮文字";
```

该代码是一个全局性的属性设置，因此，页面中所有添加“data-add-back-btn”属性的“page”容器，其头部栏中“后退”按钮的文字内容都为以上代码设置的值即“后退按钮文字”。如果需要修改，可以在页面中找到对应的“page”容器，添加“data-back-btn-text”属性进行单独设置。

另外，如果浏览的当前页面并没有可以后退的页面，那么，即使在页面的“page”容器中添加了“data-add-back-btn”属性，也不会出现“后退”按钮，如图3-2所示。

3.1.3　添加按钮

在头部栏中，还可以手动编写代码添加按钮标记。该标记通常设置为＜a＞元素，其他按钮类型的标记也可以放置在头部栏中。由于头部栏空间的局限性，所添加按钮都是内联类型的，即按钮宽度只允许放置图标与文字这两个部分。下面通过一个实例说明添加按钮的方法。

实例3-3　添加按钮

1.功能说明

在新建的HTML页面中，添加两个“page”容器，Id号分别为“e1”、“e2”，在两个容器的头部栏中分别添加两个按钮，左侧为“上一张”，右侧为“下一张”。单击第一个容器的“下一张”按钮时，切换到第二个容器；单击第二个容器的“上一张”按钮时，又返回到第一个容器。

2.实现代码

新建一个HTML页面3-3.htm，加入代码如代码清单3-3所示。

代码清单3-3　添加按钮

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile添加按钮</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/css3.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="e1">
<div data-role="header"data-position="inline">
<a href="#"data-icon="arrow-l">上一张</a>
<h1>图片</h1>
<a href="#e2"data-icon="arrow-r">下一张</a>
</div>
<div data-role="content">
<span class="img-spn">
<img src="Images/2011年作品.jpg"/>
</span>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="e2">
<div data-role="header"data-position="inline">
<a href="#e1"data-icon="arrow-l">上一张</a>
<h1>图片</h1>
<a href="#"data-icon="arrow-r">下一张</a>
</div>
<div data-role="content">
<span class="img-spn">
<img src="Images/2010年作品.jpg"/>
</span>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
```

```
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-3所示。

图　3-3　在头部栏中添加按钮元素

4.源码分析

在本实例中，头部栏通过添加“inline”属性进行定位。使用这种定位模式，无需编写其他JavaScript或CSS代码便可以确保头部栏在更多的移动浏览器中显示。

头部栏中的按钮链接元素是头部栏的首个元素，默认位置是在标题的左侧，默认按钮个数只有一个。当在标题左侧添加两个链接按钮时，左侧链接按钮会按排列顺序保留第一个，第二个按钮会自动放置在标题的右侧。因此，在头部栏中放置链接按钮时，鉴于内容长度的限制，尽量在标题栏的左右两侧分别放置一个链接按钮。

3.1.4　设定按钮位置

在头部栏中，如果只放置一个链接按钮，不论放置在标题的左侧还是右侧，其最终显示在标题的左侧。如果想改变位置，需要添加新的类别属性“ui-btn-left”和“ui-btn-right”，前者表示按钮居标题左侧（默认值），后者表示居右侧。

实例3-4　设定按钮位置

1.功能说明

在实例3-3的基础上，对头部栏中“上一张”、“下一张”两个按钮位置进行设定。在第一个“page”容器中，仅显示“下一张”按钮；切换到第二个“page”容器中时，只有“上一张”按钮存在。

2.实现代码

新建一个HTML页面3-4.htm，加入代码如代码清单3-4所示。

代码清单3-4　定位按钮

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile定位按钮位置</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/css3.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="e1">
<div data-role="header"data-position="inline">
<h1>图片</h1>
<a href="#e2"data-icon="arrow-r"class="ui-btn-right">
下一张
</a>
</div>
<div data-role="content">
<span class="img-spn">
<img src="Images/2009年作品.jpg"/></span>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="e2">
<div data-role="header"data-position="inline">
<a href="#e1"data-add-back-btn="false"
data-icon="arrow-l"class="ui-btn-left">上一张</a>
<h1>图片</h1>
</div>
<div data-role="content">
<span class="img-spn">
<img src="Images/2008年作品.jpg"/></span>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

---

3. 页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-4所示。

图　3-4　在头部栏中定位按钮元素

4.源码分析

本实例中，在头部栏中对需要定位的链接按钮添加“ui-btn-left”和“ui-btn-right”两个类别属性，用来设置头部栏中标题两侧的按钮位置，该类别属性在只有一个按钮并且想放置在标题右侧时非常有用。另外，通常情况下，需要将该链接按钮的“data-add-back-btn”属性值设置为“false”，以确保在“page”容器切换时不会出现“后退”按钮，影响标题左侧按钮的显示效果。

3.2　导航栏

jQuery Mobile为导航栏提供了专门的组件，使用时只需要将＜div＞标签的“data-role”属性值设置为“navbar”，便会产生一个导航栏容器。在该容器内，通过＜ul＞元素设置导航栏的各子类导航按钮，每一行最多可以放置5个按钮，超出个数的按钮自动显示在下一行；另外，导航栏中的按钮可以引用系统的图标，也可以自定义图标。

3.2.1　导航栏的基本结构

jQuery Mobile中的导航栏是一个被＜div＞元素包裹的容器，常常放置在页面的头部或尾部。在容器内，如果需要设置某个子类导航按钮为选中状态，只需在按钮的元素中添加一个“ui-btn-active”类别属性即可。

实例3-5　添加尾部导航栏

1.功能说明

在新建的HTML页面中，为页脚部分添加一个导航栏；在其中创建3个子类导航按钮，分别在按钮上显示“北京”、“上海”、“广州”字样，并将第一个按钮设置为选中状态。

2.实现代码

新建一个HTML页面3-5.htm，加入代码如代码清单3-5所示。

代码清单3-5　添加尾部导航栏

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile尾部导航栏</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏标题</h1></div>
<div data-role="content">
<p>添加尾部导航栏</p>
</div>
<div data-role="footer">
<div data-role="navbar">
<ul>
<li><a href="a.html"class="ui-btn-active">北京</a>
</li>
<li><a href="b.html">上海</a></li>
<li><a href="b.html">广州</a></li>
</ul>
</div>
</div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-5所示。

图　3-5　添加尾部导航栏

4.源码分析

本实例将一个简单的导航栏容器通过嵌套的方式放置在底部容器中，形成底部导航栏的页面效果。在导航栏的内部容器中，每个子类导航按钮的宽度都是一致的，因此，每增加一个子类按钮，都会将原先按钮的宽度按照等比例的方式进行均分。即如果原来有2个按钮，它们的宽度各为浏览器宽度的1/2；再增加1个按钮时，原先的2个按钮宽

度又变了1/3，依此类推。当导航栏窗口中子类按钮的数量超过5个时，将自动从2列多行的形式展示。

3.2.2　头部导航栏

除了将导航栏放置在底部外，还可以将它放置在头部，形成头部导航栏。在该导航栏中，也可以保留头部栏中的标题与按钮，只需要将导航栏容器以嵌套的方式放置在头部即可。接下来通过一个简单的实例介绍头部导航栏是如何创建的。

实例3-6　添加头部导航栏

1.功能说明

在新建的HTML页面中，添加两个“page”容器，Id号为“e1”和“e2”，分别在容器中，为页面头部添加一个导航栏。单击第一个导航栏中“音乐”按钮，页面将切换至第二个“page”容器中，并将导航栏中“音乐”按钮的状态设置为选中样式。

2.实现代码

新建一个HTML页面3-6.htm，加入代码如代码清单3-6所示。

代码清单3-6　添加头部导航栏

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile头部导航栏</title>
```

```
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header">
    <h1>图书频道</h1>
    <div data-role="navbar">
    <ul>
    <li><a href="#"class="ui-btn-active">图书</a></li>
    <li><a href="#e2">音乐</a></li>
    <li><a href="#">影视</a></li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>这是图书页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="e2">
    <div data-role="header">
    <h1>音乐频道</h1>
    <div data-role="navbar">
    <ul>
    <li><a href="#e1">图书</a></li>
    <li><a href="#"class="ui-btn-active">音乐</a></li>
    <li><a href="#">影视</a></li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>这是音乐页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
```

```
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-6所示。

图　3-6　添加头部导航栏

4.源码分析

本实例通过“page”容器间嵌套的方式，在头部栏中添加导航栏。在实际开发过程中，常常在头部栏中只嵌套导航栏，而不显示标题内容和左右两侧的按钮，特别是在导航栏中选项按钮添加了图标时，只显示页面头部栏中导航栏，效果也是不错的。

3.2.3　导航栏的图标

在导航栏中，各子类导航链接按钮是通过＜a＞元素来实现的，如果想要给导航栏中的子类链接按钮添加图标，只需要在对应的＜a＞元素中增加一个“data-icon”属性，并在jQuery Mobile自带的系统图标集合中选择一个图标名作为该属性的值，如“info”表示显示“ ”图标，图标的默认位置在按钮链接文字的上面，更多的图标名称对应的图标样式如表3-1所示。

表 3-1　“data-icon”属性对应的图标

名称	图标样式	名称	图标样式
arrow-l		refresh	
arrow-r		forward	
arrow-u		search	
arrow-d		back	
delete		grid	
plus		star	
minus		alert	
check		info	
gear		home	

上述列表中图标“data-icon”属性对应的图标名称，不仅用于导航栏中的子类链接按钮，也适用于各类按钮型元素增加图标。

实例3-7　为导航栏链接按钮添加图标

1.功能说明

在实例3-6的基础上，分别给导航栏的链接按钮添加图标。

2.实现代码

新建一个HTML页面3-7.htm，加入代码如代码清单3-7所示。

代码清单3-7　为导航栏链接按钮添加图标

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile导航栏的图标</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="e1">
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#"data-icon="info"
class="ui-btn-active">图书</a>
</li>
<li><a href="#e2"data-icon="alert">音乐</a></li>
<li><a href="#"data-icon="gear">影视</a></li>
</ul>
</div>
</div>
<div data-role="content">
<p>这是图书页面</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="e2">
<div data-role="header">
<div data-role="navbar">
```

```
    <ul>
    <li><a href="#e1"data-icon="info">图书</a></li>
    <li><a href="#"data-icon="alert"
    class="ui-btn-active">音乐</a>
    </li>
    <li><a href="#"data-icon="gear">影视</a></li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>这是音乐页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-7所示。

图　3-7　显示带图标的导航栏链接按钮

4.源码分析

在本实例中，首先给链接按钮元素添加“data-icon”属性，然后选择一个图标名，导航链接按钮上便添加了对应的图标。此外，还可以手动控制图标在链接按钮中的位置和自定义按钮图标。在接下来的章节中，将结合完整的实例逐一进行介绍。

3.2.4　设置导航栏图标位置

导航栏中的图标默认放置在按钮内容文字的上面，如果需要调整图标的位置，只需要在该导航栏容器元素中添加另外一个属性“data-iconpos”。该属性用于控制整个导航栏容器中图标的位置，默认值为“top”，表示图标在按钮文字的上面，此外，还可以选择“left”、“right”、“bottom”，分别表示图标在文字的左边、右边和下面。

实例3-8　设置导航栏链接按钮图标的位置

1.功能说明

在新建的HTML页面中，向头部栏添加3个导航栏，并分别将导航栏中按钮的图标位置设置为“left”、“right”、“bottom”。

2.实现代码

新建一个HTML页面3-8.htm，加入代码如代码清单3-8所示。

代码清单3-8　设置导航栏链接按钮图标的位置

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile设置导航栏图标位置</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header">
    <div data-role="navbar"data-iconpos="left">
    <ul>
    <li><a href="#"data-icon="info"
    class="ui-btn-active">图书</a>
    </li>
    <li><a href="#e2"data-icon="alert">音乐</a></li>
    <li><a href="#"data-icon="gear">影视</a></li>
    </ul>
    </div>
    <div data-role="navbar"data-iconpos="right">
    <ul>
    <li><a href="#"data-icon="info"
    class="ui-btn-active">图书</a>
    </li>
    <li><a href="#e2"data-icon="alert">音乐</a></li>
    <li><a href="#"data-icon="gear">影视</a></li>
    </ul>
    </div>
    <div data-role="navbar"data-iconpos="bottom">
    <ul>
    <li><a href="#"data-icon="info"
    class="ui-btn-active">图书</a>
    </li>
    <li><a href="#e2"data-icon="alert">音乐</a></li>
    <li><a href="#"data-icon="gear">影视</a></li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>展示导航栏中图标的不同位置</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
```

```
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-8所示。

图　3-8　设置导航栏链接按钮图标的位置

4.源码分析

在本实例中，通过在导航栏容器中增加“data-iconpos”属性，改变导航栏按钮图标的位置。但是该属性针对的是整个导航栏容器，而不是导航栏内某个导航链接按钮图标的位置。因此，“data-

iconpos”是一个全局性的属性，针对的是整个导航栏内全部的链接按钮。

3.2.5　自定义图标

开发者也可以根据自己的喜好自定义图标，实现的方法是：另外创建一个CSS样式文件，在文件中添加链接按钮的图标地址与显示位置。下面通过一个简单的实例来介绍如何给导航栏链接按钮自定义图标。

实例3-9　自定义导航栏链接按钮的图标

1.功能说明

在实例3-7的基础上，将导航栏中链接按钮的图标替换成自定义的3个图标。

2.实现代码

新建一个HTML页面3-9.htm，加入代码如代码清单3-9所示。

代码清单3-9　自定义导航栏链接按钮的图标

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile自定义图标</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
```

```
    <link href="Css/css3.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="e1">
    <div data-role="header"class="nav-3-9">
    <div data-role="navbar"class="nav-3-9">
    <ul>
    <li><a href="#"data-icon="custom"
    class="ui-btn-active books">图书</a>
    </li>
    <li><a href="#e2"data-icon="custom"
    class="music">音乐</a>
    </li>
    <li><a href="#"data-icon="custom"
    class="movie">影视</a>
    </li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>这是图书页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></</div>
    </div>
    <div data-role="page"id="e2">
    <div data-role="header"class="nav-3-9">
    <div data-role="navbar"class="nav-3-9">
    <ul>
    <li><a href="#e1"data-icon="custom"
    class="books">图书</a>
    </li>
    <li><a href="#"data-icon="custom"
    class="ui-btn-active music">音乐</a>
    </li>
    <li><a href="#"data-icon="custom"
    class="movie">影视</a>
    </li>
    </ul>
    </div>
```

```
    </div>
    <div data-role="content">
    <p>这是音乐页面</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

另外，创建一个用于实例3-9引用的样式文件css3.css，加入以下代码：

```
    /*实例3-9所需添加的样式*/
    .nav-3-9.ui-btn.ui-btn-inner
    {
    padding-top: 40px! important;
    }
    .nav-3-9.ui-btn.ui-icon
    {
    width: 30px! important;
    height: 30px! important;
    margin-left: -15px! important;
    box-shadow: none! important;
    -moz-box-shadow: none! important;
    -webkit-box-shadow: none! important;
    -webkit-border-radius: 0! important;
    border-radius: 0! important;
    }
    .books.ui-icon
    {
    background: url（icons/01.png）50%50%no-repeat;
    background-size: 18px 26px;
    }
    .music.ui-icon
    {
    background: url（icons/02.png）50%50%no-repeat;
    background-size: 15px 24px;
    }
    .movie.ui-icon
    {
```

```
background: url（icons/03.png）50%50%no-repeat;
background-size: 19px 25px;
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-9所示。

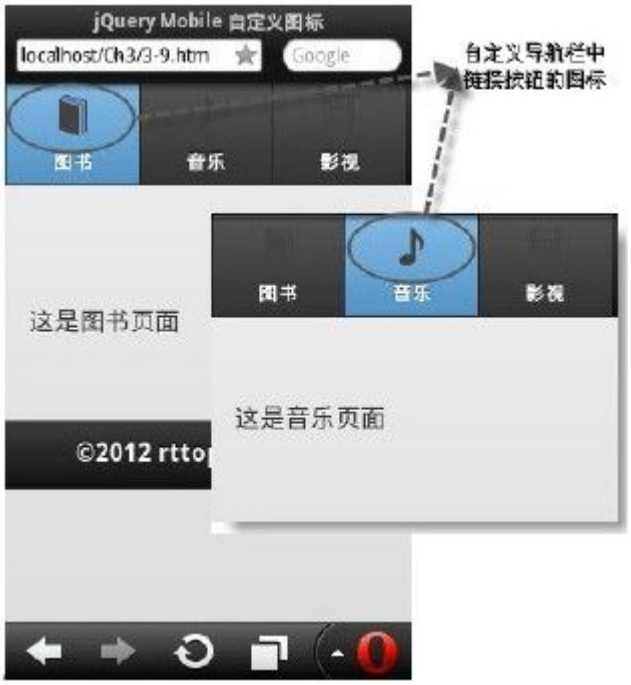


图　3-9　自定义导航栏链接按钮的图标

4.源码分析

在本实例中，自定义导航栏的链接按钮图标需要以下2个步骤：

步骤1　新建一个CSS样式文件。

在该文件中设置某个链接按钮的自定义图标地址与显示位置，部分代码如下所示：

```
……省略部分代码
.books.ui-icon
{
background: url（icons/01.png）50%50%no-repeat;
background-size: 18px 26px;
}
……省略部分代码
```

在上述代码中，新建一个“books”类别，在该类别下编写“ui-icon”类别的内容。“ui-icon”类别有2行代码，第一行通过“background”设置自定义图标的地址和显示方式，第二行通过“background-size”设置自定义图标显示的长度与宽度。

步骤2　在导航栏中引用新建的“books”类别，并将“data-icon”属性值设置为“custom”。

此步骤为了表示该项子类导航链接按钮的图标是自定义形式的，完整的代码如下：

```
   <li><a href="#e1"data-icon="custom"class="books">图书</a></li>
```

3.3　尾部栏

实例3-5介绍了在尾部栏中添加导航栏的方法，其实，尾部栏与头部栏的结构差不多，区别是设置的“data-role”属性值不同。相对头部栏来说，尾部栏的代码更加简洁。在尾部栏中可以添加按钮组和表单中的各个元素；同时，还可以对某个尾部栏进行定位处理。

3.3.1　添加按钮

向尾部栏添加按钮时，为了减少各按钮的间距，通常需要在按钮的外围添加一个“data-role”属性值为“controlgroup”的容器，形成一个按钮组显示在尾部栏中；同时，在该容器中添加一个“data-type”属性，并将该属性的值设置为“horizontal”，表示容器中的按钮按水平顺序排列。

实例3-10　在尾部栏中添加按钮

1.功能说明

新建一个HTML页面，在页面尾部栏中添加一个按钮组。在该按钮组中添加2个带图标的按钮，按钮中的文本内容分别为“关于公司”和“联系我们”。

2.实现代码

新建一个HTML页面3-10.htm，加入代码如代码清单3-10所示。

代码清单3-10　在尾部栏中添加按钮

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile添加尾部栏按钮</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏标题</h1></div>
<div data-role="content">
<p>添加尾部栏按钮</p>
</div>
<div data-role="footer">
<div data-role="controlgroup"data-type="horizontal">
<a href="#"data-role="button"
data-icon="home">关于公司
</a>
<a href="#"data-role="button"
data-icon="forward">联系我们
</a>
</div>
</div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-10所示。

图　3-10　在尾部栏中添加按钮

4.源码分析

在实例中，由于底部栏中的按钮外围被一个“data-role”属性值为“controlgroup”的容器所包裹，因此，按钮间没有任何“padding”空间。如果想要给底部栏中的按钮添加“padding”空间，则不需要使用容器包裹，另外给底部栏容器添加一个“ui-bar”类别属性，代码如下：

……省略部分代码
<div data-role="footer"class="ui-bar">
<a href="#"data-role="button"
data-icon="home">关于公司
<a href="#"data-role="button"
data-icon="forward">联系我们

……省略部分代码

3.3.2　添加表单元素

在底部栏中，可以添加按钮组，也可以向容器内增加表单中的元素，如＜select＞、＜text＞等。为了确保表单元素在底部栏的正常显示，需要在底部栏容器中增加“ui-bar”类别，使新增加的表单元素间保持一定的间距；此外，将“data-position”属性值设置为“inline”，用于统一设定各表单元素的显示位置。

实例3-11　在尾部栏中添加表单元素

1.功能说明

新建一个HTML页面，在页面尾部栏中添加一个表单元素中的下拉列表框，用于显示“友情链接”的公司信息。

2.实现代码

新建一个HTML页面3-11.htm，加入代码如代码清单3-11所示。

代码清单3-11　在尾部栏中添加表单元素

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞jQuery Mobile添加表单元素＜/title＞
＜meta name="viewport"content="width=device-width,
initial-scale=1"/＞
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏标题</h1></div>
<div data-role="content">
<p>在尾部栏添加表单元素</p>
</div>
<div data-role="footer"
class="ui-bar"data-position="inline">
<label for="selLink">友情链接</label>
<select name="selLink"id="selLink">
<option value="0">请选择</option>
<option value="1">公司1</option>
<option value="2">公司2</option>
<option value="3">公司3</option>
<option value="4">公司4</option>
</select>
</div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-11所示。

图　3-11　在尾部栏中添加表单元素

4.源码分析

在本实例中，为尾部栏添加了一个＜select＞表单元素。如图3-11所示，移动终端与PC端的浏览器在显示表单元素时，还是存在一些细微的区别。比如＜select＞元素，在PC端的浏览器中是以下拉列表框的形式展示，而在移动终端，则是以弹出框的形式展示全部的列表内容。

3.4　内容格式化

jQuery Mobile中提供了许多非常有用的工具与组件，如多列的网格布局、折叠形的面板控制等，这些组件可以帮助开发者快速实现正文区域内容的格式化。

3.4.1　网格布局

jQuery Mobile提供的CSS样式“ui-grid”可以实现内容的网格布局。该样式有4种预设的配置布局：“ui-grid-a”、“ui-grid-b”、“ui-grid-c”、“ui-grid-d”，分别对应两列、三列、四列、五列的网格布局形式，可以最大范围满足页面多列的需求。

使用网格布局时，整个宽度为100%，无任何“padding”和“margin”及背景色，因此不会影响到其他元素放入网格中的位置。

实例3-12　在内容区域添加多种类型的网格布局

1.功能说明

新建一个HTML页面，在内容区域中添加4种预设的网格布局，并且通过不同的主题颜色进行区分，以分块的形式显示在页面中。

2.实现代码

新建一个HTML页面3-12.htm，加入代码如代码清单3-12所示。

代码清单3-12　在内容区域添加多种类型的网格布局

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile网格布局</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/css3.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏标题</h1></div>
<div class="ui-grid-a">
<div class="ui-block-a">
<div class="ui-bar ui-bar-b h60">A</div>
</div>
<div class="ui-block-b">
<div class="ui-bar ui-bar-b h60">B</div>
</div>
</div>
<div class="ui-grid-b">
<div class="ui-block-a">
<div class="ui-bar ui-bar-c h60">A</div>
</div>
<div class="ui-block-b">
<div class="ui-bar ui-bar-c h60">B</div>
</div>
<div class="ui-block-c">
<div class="ui-bar ui-bar-c h60">C</div>
</div>
</div>
<div class="ui-grid-c">
```

```
    <div class="ui-block-a">
    <div class="ui-bar ui-bar-d h60">A</div>
    </div>
    <div class="ui-block-b">
    <div class="ui-bar ui-bar-d h60">B</div>
    </div>
    <div class="ui-block-c">
    <div class="ui-bar ui-bar-d h60">C</div>
    </div>
    <div class="ui-block-d">
    <div class="ui-bar ui-bar-d h60">D</div>
    </div>
    </div>
    <div class="ui-grid-d">
    <div class="ui-block-a">
    <div class="ui-bar ui-bar-e h60">A</div>
    </div>
    <div class="ui-block-b">
    <div class="ui-bar ui-bar-e h60">B</div>
    </div>
    <div class="ui-block-c">
    <div class="ui-bar ui-bar-e h60">C</div>
    </div>
    <div class="ui-block-d">
    <div class="ui-bar ui-bar-e h60">D</div>
    </div>
    <div class="ui-block-e">
    <div class="ui-bar ui-bar-e h60">E</div>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-12所示。

图　3-12　在内容区域添加多种类型的网格布局

4.源码分析

在本实例的代码中，要增加一个多列的网格区域，首先通过＜div＞元素构建一个容器，如果是两列，则给该容器添加的“class”属性值为“ui-grid-a”；三列则为“ui-grid-b”，依此类推。

然后，在已构建的容器中添加子容器，如果是两列，则给两个子容器分别添加“ui-block-a”、“ui-block-b”样式属性；如果是三列，则给三个子容器分别添加“ui-block-a”、“ui-block-b”、“ui-block-c”样式属性；其他多列依此类推。

最后，在子容器中放置需要显示的内容。在本实例中，每个子容器都分别放置了一个＜div＞元素，代码如下所示：

```
<div class="ui-bar ui-bar-b h60">A</div>
```

在上述代码中，＜div＞元素通过“class”属性添加了三个样式。第一个和第二个都是jQuery Mobile自带的样式，“ui-bar”用于控制各子容器的间距，“ui-bar-b”用于设置各子容器的主题样式。第三个样式“h60”为自定义样式，将子容器的高度设置为“60px”。

说明　如果容器选择的样式为两列即“class”值为“ui-grid-a”，而在它的子容器中添加了三个子项即“class”值为“ui-block-c”，那么该列自动被放置在下一行。

3.4.2　可折叠的区块

上节使用“ui-grid”显示多列的网格布局效果，在jQuery Mobile中，还可以对指定的区块进行折叠。实现对区块的折叠需要完成以下3步操作：

步骤1　创建一个＜div＞容器，将该容器的“data-role”属性设为“collapsible”，表示该容器是一个可折叠的区块。

步骤2　在容器中添加一个＜h3＞标题文字标记，该标记以按钮的形式展示。按钮的左侧有一个“+”号，表示该标题可以点开。

步骤3　在标题的下面放置需要折叠显示的内容，通常使用＜p＞段落元素。当用户单击标题中的“+”号时，显示＜p＞元素中的内容，标题左侧中“+”号变成“-”号；再次单击时，隐藏＜p＞元素中的内容，标题左侧中“-”号变成“+”号。

下面通过一个简单的实例来介绍在jQuery Mobile中可折叠区块实现的方式。

实例3-13　在正文中显示可折叠的区块

1.功能说明

新建一个HTML页面，在内容区域中添加一个可折叠的区块。当用户单击区块中的标题时，如果是“+”号，则显示标题下的内容；再次单击时，如果是“-”号，则隐藏标题下的内容。

2.实现代码

新建一个HTML页面3-13.htm，加入代码如代码清单3-13所示。

代码清单3-13　实现可折叠的区块

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile可折叠的区块</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏标题</h1></div>
<div data-role="collapsible"data-collapsed="false">
<h3>点击查看更多</h3>
<p>一位优秀的Web端工程师，不仅要有过硬的技术，
而且要有执着、沉稳的品质。</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-13所示。

图　3-13　在正文中显示可折叠的区块

4.源码分析

在本实例中，通过将容器的“data-role”属性设置为“collapsible”，使该容器形成一种折叠式的页面效果。容器内的标题字体可以在“h1～h6”之间选择，根据需求进行设置。另外，在该容器中通过设置“data-collapsed”属性值，可以调整容器折叠的状态。该属性默认值为“true”，表示标题下的内容是隐藏的，为收缩

状态；如果将该属性值设置为“false”，标题下的内容是显示的，为下拉状态。

3.4.3　可嵌套的折叠区块

jQuery Mobile允许对折叠的区块进行嵌套显示，即在一个折叠区域块的内容中，再添加一个折叠区块，依此类推。但建议这种嵌套最多不超过3层，否则，用户体验和页面性能都将比较差。

实例3-14　在正文中显示可嵌套的折叠区块

1.功能说明

新建一个HTML页面，在内容区域中添加3个“data-role”属性值为“collapsible”的折叠区块，分别以嵌套的方式进行组合。单击“第一层”标题时，显示“第二层”折叠区；单击“第二层”标题时，显示“第三层”折叠区。

2.实现代码

新建一个HTML页面3-14.htm，加入代码如代码清单3-14所示。

代码清单3-14　实现可嵌套的折叠区块

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile可嵌套的折叠区块</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏标题</h1></div>
    <div data-role="collapsible">
    <h3>第一层</h3>
    <p>这是第一层中的内容</p>
    <div data-role="collapsible">
    <h3>第二层</h3>
    <p>这是第二层中的内容</p>
    <div data-role="collapsible">
    <h3>第三层</h3>
    <p>这是第三层中的内容</p>
    </div>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-14所示。

图　3-14　在正文中显示可嵌套的折叠区块

4.源码分析

本实例展示了3层折叠区块相互嵌套的效果。在jQuery Mobile中，折叠容器中的内容区域可以放置任何想要折叠的HTML标记，当然，也允许再添加一个折叠块，从而形成嵌套式的折叠区块。虽然是嵌套的折叠区块，但各自的“data-collapsed”属性是独立的，即每层只控制各自的内容是收缩还是下拉。

3.4.4　折叠组标记

折叠区块可以嵌套，也可以形成折叠组，实现的方法是：在一个“data-role”属性为“collapsible-set”的＜div＞容器中添加多个折叠区块，而这些区块就是折叠组区块。它们同属于一个容器，在视觉上形成“手风琴”的效果；在同一时间，折叠组中只有一个折叠区块是打开的，当打开别的折叠区块时，其他“组成员”将自动关闭。

实例3-15　在正文中显示折叠组标记

1.功能说明

新建一个HTML页面，添加一个“data-role”属性为“collapsible-set”的折叠组容器，在该容器中增加3个折叠区块，标题分别对应“图书”、“音乐”、“影视”。初次显示时，“音乐”折叠区块为打开状态。

2.实现代码

新建一个HTML页面3-15.htm，加入代码如代码清单3-15所示。

代码清单3-15　折叠组标记

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>jQuery Mobile折叠组标记</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏标题</h1></div>
    <div data-role="collapsible-set">
    <div data-role="collapsible">
    <h3>图书</h3>
    <p><a href="#">文艺</a></p>
    <p><a href="#">少儿</a></p>
    <p><a href="#">社科</a></p>
    </div>
    <div data-role="collapsible"data-collapsed="false">
    <h3>音乐</h3>
    <p><a href="#">流行</a></p>
    <p><a href="#">民族</a></p>
    <p><a href="#">通俗</a></p>
    </div>
    <div data-role="collapsible">
    <h3>影视</h3>
    <p><a href="#">欧美</a></p>
    <p><a href="#">怀旧</a></p>
    <p><a href="#">娱乐</a></p>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></</div>
    </div>
    </body>
    </html>
```

---

3. 页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图3-15所示。

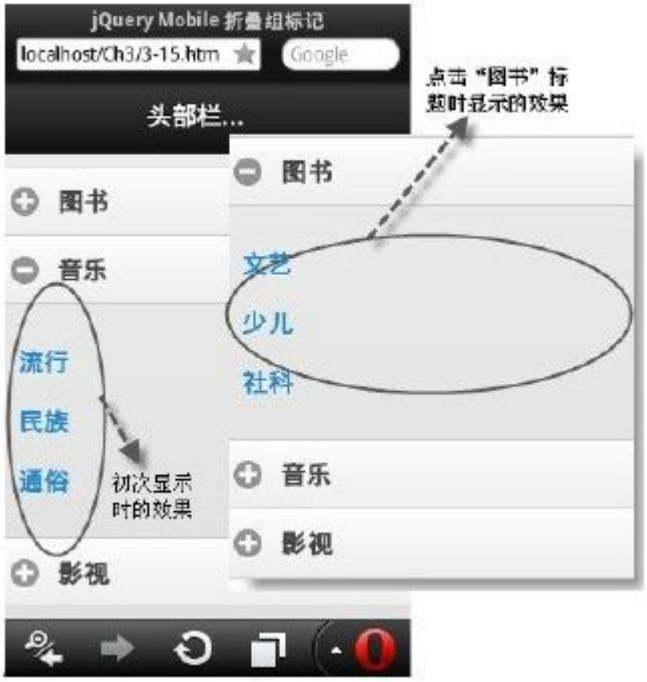


图　3-15　在正文中显示折叠组的效果

4.源码分析

折叠组中所有的折叠区块默认状态都是收缩的，如果想在默认状态下使某个折叠区块为下拉状态，只要将该折叠区块的“data-collapsed”属性值设置为“false”。如本实例中，将标题为“音乐”的折叠区块的“data-collapsed”属性值设置为“false”。需要注意的是，由于同处在一个折叠组内，这种下拉状态在同一时间只允许有一个。

3.5　本章小结

本章重点介绍了两个部分的内容，一是工具栏中的成员，包括头部栏、导航栏、尾部栏的使用方法与技巧；二是正文区域的内容格式化。通过介绍网格布局、折叠效果展示内容的方式，使读者在掌握jQuery Mobile基本页面布局的基础上，进一步了理解工具栏与内容组件在处理页面内容时带来的效果，同时为第4章常用组件的学习打下基础。

第4章　页面常用组件

本章内容

按钮

表单

列表

本章小结

除了第3章提到的工具栏组件外，jQuery Mobile还提供了许多常用的组件，例如通过＜a＞元素衍生的按钮、专门针对表单提供的各种类型的内部元素组件、以列表方式展示更多内容的列表组件等。在jQuery Mobile移动项目中，使用这些组件能够丰富开发页面的方法与工具，接下来逐一进行详细的介绍。

4.1 按钮

jQuery Mobile中的按钮由两类元素形成：一类是＜a＞元素，将该元素的“data-role”属性值设置为“button”，jQuery Mobile便会自动给该元素一些Class样式属性，形成可单击的按钮形状；另一类是在表单内无须添加“data-role”属性，jQuery Mobile会自动将＜input＞元素中“type”属性值为“submit”、“reset”、“button”、“image”形成按钮样式，在内容中放置按钮时，可以采用内联或按钮组的方式进行排版。

4.1.1 内联按钮

在jQuery Mobile中，被样式化的按钮元素默认都是块状，并且自动填充页面宽度。如果要取消默认效果，只需要在按钮的元素中添加“data-inline”属性，将该属性值设为“true”，该按钮将会根据它内容中文字和图片的宽度自动进行缩放，形成一个紧凑型的按钮。

如果想要将缩放后的按钮在同一行显示，可以在多个按钮的外层增加一个＜div＞容器，在该容器中将“data-inline”属性值设为“true”，这样就可以使容器中的按钮样式自动缩放至最小宽度，并且以浮动效果在一行中显示。

在内联按钮中，如果想使两个以上的按钮在同一行，且自动均分页面宽度，可以使用网格分栏的方式，将多个按钮放置在分栏后的同一行中。接下来通过一个简单实例说明该方法实现的过程。

实例4-1　内联按钮

1.功能说明

新建一个HTML页面，通过分栏的方式在页面中添加一个普通按钮和一个表单按钮，使两个按钮在同一行显示。

2.实现代码

新建一个HTML页面4-1.htm，加入代码如代码清单4-1所示。

代码清单4-1　内联按钮

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile内联按钮</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
```

```
    <div class="ui-grid-a">
    <div class="ui-block-a">
    <a href="#"data-role="button"
    class="ui-btn-active">确定
    </a>
    </div>
    <div class="ui-block-b">
    <input type="submit"value="取消"/>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-1所示。

图　4-1　内联按钮在页面中展示的效果

4.源码分析

在本实例中，运用分栏容器使两个按钮显示在同一行，两个按钮的宽度可以与移动终端

浏览器的宽度进行自动等比缩放，因此适应移动终端中不同分辨率的浏览器。

如果希望形成的按钮不与浏览器等比缩放，且多个按钮也要在同一行显示，可以将按钮元素的“data-inline”属性值设置为“true”，本实例的代码可以修改为：

```
    ……省略部分代码
    ＜a href="#"data-role="button"class="ui-btn-active"
    data-inline="true"＞确定＜/a＞
    ＜a href="#"data-role="button"data-inline="true"＞取消＜/a
＞
    ……省略部分代码
```

上述代码同样可以使两个按钮以内联的方式同一行显示在页面中，只是由于固定了宽度，导致不能与浏览器的宽度进行等比缩放。

4.1.2　按钮组标记

在jQuery Mobile中，多个按钮除不但能以内联的形式显示，还可以全部放入按钮组，即“controlgroup”容器中，按照垂直或水平方向展现按钮列表。默认情况下，按钮组是以垂直方向展示一组按钮列表，可以通过给按钮组容器添加“data-type”属性来修改按钮组默认的显示方式。

实例4-2　按钮组标记

1.功能说明

新建一个HTML页面，创建两个不同“data-type”属性的按钮组，一个以垂直方向的形式展示两个按钮列表，另一个以水平方向的形式展示两个按钮列表。

2.实现代码

新建一个HTML页面4-2.htm，加入代码如代码清单4-2所示。

代码清单4-2　按钮组标记

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile按钮组标记</title>
```

```
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="controlgroup">
    <a href="#"data-role="button"
    data-icon="check"class="ui-btn-active">确定</a>
    <a href="#"data-role="button"data-icon="delete">取消</a
>
    </div>
    <div data-role="controlgroup"data-type="horizontal">
    <a href="#"data-role="button"data-icon="check"
    class="ui-btn-active">确定</a>
    <a href="#"data-role="button"data-icon="delete">取消</a
>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-2所示。

图　4-2　以按钮组的不同方式显示按钮列表

4.源码分析

在本实例中，当按钮列表被按钮组标记包裹时，每个被包裹的按钮都会自动删除自身“margin”的距离和背景的阴影，并且只在第一个按钮上面的两个角和最后一个按钮下面的两个角使用圆角的样式，这样使整个按钮列表在显示效果上更加像一个组的集合。

如果按钮组以水平的方式显示按钮列表，那么默认情况下，所有按钮向左边靠拢，自动缩放到各自适合的宽度，最左边按钮的左侧与最右边按钮的右侧两个角使用圆角的样式，完整的显示效果如图4-2所示。

说明　如果在按钮组中仅放置一个按钮，那么，该按钮仍是以正常圆角的效果显示在页面中。

4.2 表单

在HTML元素中，表单占有十分重要的地位。针对表单，jQuery Mobile提供了一套完全基于HTML原始代码且适合触摸操作的框架。在该框架下，所有的表单元素先由原始的代码升级为jQuery Mobile组件，然后调用各自组件提供的方法与属性，实现在jQuery Mobile下表单元素的各项操作。例如，在jQuery Mobile表单中，一个“type”属性值为“checkbox”的元素先通过对应的“checkboxradio”插件升级为组件，完成相应数据的初始化后，就可以调用jQuery UI中组件的方法与属性，实现该表单元素的相应功能。

需要说明的是，在表单中，各元素通过原始HTML代码升级为jQuery Mobile是自动完成的；当然，也可以阻止这种升级行为，只要将该表单元素的“data-role”属性值设置为“none”即可。另外，由于在单个页面中可能会出现多个“page”容器，为了保证表单在提交数据时的唯一性，必须确保每一个表单的Id号是唯一的。

4.2.1 文本输入

在jQuery Mobile中，文本输入包括文本输入框和文本输入域，以及HTML 5中新增的输入类型。文本输入框使用标准的HTML原始元素，借助jQuery Mobile的渲染效果，使其更易于触摸型使用；在jQuery

Mobile中使用的文本输入域的高度会自动增加，无须因高度问题拖动滑动条。

另外，HTML 5中新增的输入类型（如“number”），在jQuery Mobile中会被渲染成除数字输入框外、还在输入框的最右端有两个可调节大小的“+”和“-”号按钮，方便移动终端的用户修改输入框中的数字。

实例4-3　不同类型的文本输入

1.功能说明

新建一个HTML页面，在内容区域中创建三个不同类型的输入框元素，分别对应“search”、“text”、“number”，用于显示在jQuery Mobile中不同类型的输入框元素的渲染效果。

2.实现代码

新建一个HTML页面4-3.htm，加入代码如代码清单4-3所示。

代码清单4-3　文本输入

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile文本输入</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/css4.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header">
    <h1>头部栏</h1>
    </div>
    <div data-role="content">
    搜索：<input
type="search"name="password"id="search"value=""/>
    姓名：<input type="text"name="name"id="name"value=""/>
    年龄：<input
type="number"name="number"id="number"value="0"/>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-3所示。

图　4-3　不同类型输入框的页面效果

4.源码分析

从图4-3可以看出，在jQuery Mobile中，“type”类型是“search”的搜索输入文本框的外围有圆角，最左端有一个圆形的搜索图标。当输入框中有内容字符时，它的最右侧会出现一个圆形的“×”按钮，单击该按钮时，可以清空输入框中的内容。

在“type”类型是“number”的数字输入文本中，单击最右端的上下两个调整按钮，可以动态改变数字输入框中的值的大小，使用十分方便。

4.2.2　滑块

在页面中，如果添加一个＜input＞元素并将“type”的属性值设为“range”时，便可以创建一个滑块组件。在jQuery Mobile中，滑块组件由两部分组成，一个部分是可调整大小的数字输入框，另一部分是可拖动修改输入框数字的滑动条。滑块元素可以通过添加“min”和“max”属性值来设置滑动条的取值范围，如“min”的属性值为“0”，“max″的属性值为“10”，表示该滑块只能在0～10之间进行取值。

实例4-4　拖动滑块改变元素背景色

1.功能说明

新建一个HTML页面，在页面中添加一个滑块元素和一个Id号为“spnPrev”的＜span＞元素，当拖动滑动条或修改数字输入框的值时，＜span＞元素的背景色也随之发生变化。

2.实现代码

新建一个HTML页面4-4.htm，加入代码如代码清单4-4所示。

代码清单4-4　滑块

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile滑块</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/css4.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
//JavaScript Document
function$$（id）{
return document.getElementById（id）;
}
//动态改变区块背景色
function setSpnColor（）{
var strColor="rgb（"+$（"#txtR"）.val（）+", 233, 244）";
$$（"spnPrev"）.style.backgroundColor=strColor;
}
</script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<input type="range"id="txtR"value="0"
min="0"max="255"onchange="setSpnColor（）"/>
<span id="spnPrev"></span>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

---

3. 页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-4所示。

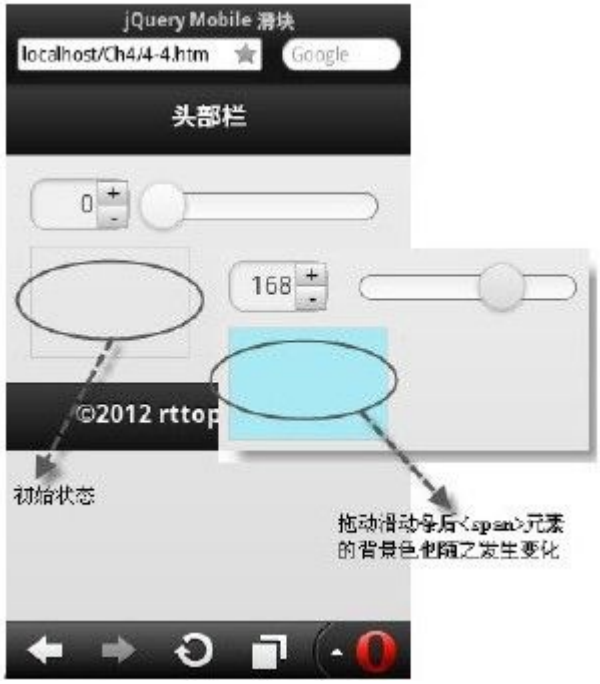


图　4-4　拖动滑块改变元素背景色的效果

4.源码分析

在本实例中，滑块拖动时改变的值是数字输入框的值，而“min”与“max”的属性值是指定滑动条的取值范围。拖动滑动条或单击数字输入框中的“+”或“-”号可以修改滑块值，此外，在键盘上单击方向键或“PageUp”、“PageDown”、“Home”、“End”键，也可以调节滑块值的大小。当然，通过JavaScript代码也可以设置滑块的值，但必须完成设置后对滑块的样式进行刷新。例如通过JavaScript代码设置滑块的值为“180”，代码如下：

```
$（function（）{
$（"input[type=range]"）.val（180）.slider（"refresh"）;
}）
```

上述代码将当前滑块的值设置为“180”，并刷新了滑动条的样式，使其滑动到“180”这个刻度上，与数字输入框的值相对应。

4.2.3　翻转切换开关

在jQuery Mobile中，将＜select＞元素的“data-role”属性值设置为“slider”，可以将该下拉列表元素下的两个＜option＞选项样式变成一个翻转切换开关。第一个＜option＞选项为“开”，取值为“true”或“1”；第二个＜option＞选项为“关”，取值为“false”或“0”。它是移动设备上常用的UI元素之一，常用于一些系统默认值的设置。

实例4-5　翻转切换开关

1.功能说明

新建一个HTML页面，在页面中添加一个＜select＞元素，并将该元素的“data-role”属性值设置为“slider”，形成一个切换开关。在元素中增加两个＜option＞选项，一个显示文本为“开”，取值为“1”，另一个显示文本为“关”，取值为“0”。切换开关时，显示当前开关所选择的值。

2.实现代码

新建一个HTML页面4-5.htm，加入代码如代码清单4-5所示。

代码清单4-5　翻转切换开关

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile翻转切换开关</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/css4.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
//显示翻转切换开关当前的值
function ChangeEvent（）{
$（"#pTip"）.html（$（"#slider"）.val（））;
}
</script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<select id="slider"data-role="slider"
onchange="ChangeEvent（）; ">
<option value="1">开</option>
<option value="0">关</option>
</select>
<p id="pTip"></p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-5所示。

图　4-5　获取翻转开关的不同选择值

4.源码分析

在本实例中，翻转开关在滑动时，将会触发一个“change”事件，在该事件中，可以获取切换后的值，即Id号为“slider”的翻转开关中被选中项的值，而不是显示的文本内容。与滑块相同，如果使用JavaScript代码设置翻转开关的值，完成设置后必须进行刷新，如下所示：

```
$（function（）{
$（"#slider"）[0].selectedIndex=1;
```

```
$（"#slider"）.slider（"refresh"）；
}）
```

上述代码将第一个选项设置为选中状态，并且刷新了一次整个翻转开关元素。

4.2.4　单选按钮

在jQuery Mobile中，单选按钮样式化后，更加容易被点击和触摸。在通常情况下，先使用“data-role”属性值为“controlgroup”的＜fieldset＞元素，包裹全部的＜input＞和＜label＞元素，这样可以以整个组的形式样式化容器中的全部标记；然后，在组成员结构中，每个＜label＞元素都通过“for”属性对应一个类型为“radio”的＜input＞元素。为了便于用户触摸，这些＜label＞元素将会被拉长。实际上，当用户触摸某个单选按钮时，单击的是该单选按钮对应的＜label＞元素。

实例4-6　用两种方式显示单选按钮

1.功能说明

新建一个HTML页面，使用＜fieldset＞容器以不同的“data-type”属性值包裹一个单选按钮组，该按钮组有3个单选按钮，分别对应“A”、“B”、“C”三个选项。单击某个单选按钮，将显示被选中按钮的值。

2.实现代码

新建一个HTML页面4-6.htm，加入代码如代码清单4-6所示。

代码清单4-6　单选按钮

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile单选按钮</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
$（function（）{
//获取单选按钮选择时的值
$（"input[type='radio']"）.bind（"change",
function（event, ui）{
$（"#pTip"）.html（this.value）;
}）
}）
</script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<feldset data-role="controlgroup"data-type="horizontal">
<input
type="radio"name="rdoA"id="rdo1"value="1"checked="checked"/>
<label for="rdo1">A</label>
<input type="radio"name="rdoA"id="rdo2"value="2"/>
<label for="rdo2">B</label>
<input type="radio"name="rdoA"id="rdo3"value="3"/>
<label for="rdo3">C</label>
</feldset>
<p id="pTip"></p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
```

```
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-6所示。

图　4-6　以不同的方式展现单选按钮的效果

4.源码分析

在本实例中，以垂直和水平两种方式展现了单选按钮的效果。由于被<fieldset>元素以组的形式包裹，无论是垂直还是水平的方向，单选按钮的四周都有圆角的样式，以一个整体组的形式显示在页

面中。单击某个单选按钮时，将触发对应的“change”事件，并在该事件中可以获取单选按钮对应的值，效果如图4-6所示。

与众多表单元素一样，如果使用JavaScript代码改变选按钮组中某个单选按钮的值，设置后必须对应整个单选按钮组进行刷新，以确保使对应的样式同步，如下代码所示：

```
$（function（）{
$（"input[type='radio']: first"）.attr（"checked", true）
.checkboxradio（"refresh"）；
}）
```

上述代码在设置完第一个单选按钮的选中属性后，对整个单选按钮组进行了一次刷新。

4.2.5　复选框

与单选按钮相类似，多个复选框选项被一个“data-role”属性值为“controlgroup”的＜fieldset＞元素所包裹。通常情况下，多个复选框选项组合成的复选框按钮组放置在标题下面，通过jQuery Mobile固有的样式自动删除各个按钮间的“margin”距离，使其看起来更像一个整体；另外，复选框按钮组默认是垂直显示，也可以将＜fieldset＞元素的“data-type”属性值修改为“horizontal”，变成水平显示。如果是水平显示，将自动隐藏各个复选框的Icon，并浮动成一排显示。这种效果类似于编辑器中“字体”、“颜色”、“下划线”等的设置效果。

实例4-7　用两种方式显示复选框

1.功能说明

新建一个HTML页面，以垂直和水平两种方式显示复选框按。当用户单击多个选项时，分别获取所选择复选框对应的值，并同时显示在页面中。

2.实现代码

新建一个HTML页面4-7.htm，加入代码如代码清单4-7所示。

代码清单4-7　复选框

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile复选框</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
$（function（）{
var strChangeVal="";
var objCheckBox=$（"input[type='checkbox']"）;
//设置复选框选择时的值
objCheckBox.bind（"change", function（event, ui）{
if（this.checked）{
strChangeVal+=this.value+", ";
}else{
strChangeVal=GetChangeValue（objCheckBox）;
}
$（"#pTip"）.html（strChangeVal）;
}）
}）
//获取全部选择按钮的值
function GetChangeValue（v）{
var strS="";
v.each（function（）{
if（this.checked）{
strS+=this.value+", ";
}
}）;
return strS;
}
</script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
```

```
    <div data-role="content">
    <feldset data-role="controlgroup"data-type="horizontal">
    <input type="checkbox"name="chkA"id="chk1"value="1"/>
    <label for="chk1">A</label>
    <input type="checkbox"name="chkA"id="chk2"value="2"/>
    <label for="chk2">B</label>
    <input type="checkbox"name="chkA"id="chk3"value="3"/>
    <label for="chk3">C</label>
    </feldset>
    <p id="pTip"></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></</div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-7所示。

图　4-7　以不同的方式展现复选框按钮的效果

4.源码分析

在本实例中，如果想获取被选中的复选框按钮值，需要遍历整个按钮组，根据各个选项的选中状态，以累加的方式记录被选中的复选框值。由于复选框也可以取消选中状态，因此，用户选中后又取消时，需要再次遍历整个按钮组，重新以累加的方式记录所有被选中的复选框值，详细的实现代码如源码中加粗部分所示。

与单选按钮一样，如果使用JavaScript代码控制复选框按钮中的选中状态，在设置完成后需要对整个按钮组的样式进行刷新一次，代码如下所示：

```
    $（function（）{
    $（"input[type='checkbox']: first"）.attr（"checked",
true）.
    checkboxradio（"refresh"）;
    }）
```

上述代码在将第一个复选框设置为选中状态后，对整个复选框按钮组进行了一次刷新，以确保整体的样式与选中的复选框保持同步。

4.2.6　选择菜单

与单选按钮和复选框不同，＜select＞元素形成的选择菜单在jQuery Mobile中样式发生了很大的变化。它分为两种类别：一种是原生菜单类型，这种类型继续保持了原来PC端浏览器的样式，单击右端的向下箭头，出现一个下拉列表，选择其中的某一项；另一种类型是自定义菜单类型，该类型专用于移动设备的浏览器显示，使用该类型时，jQuery Mobile中提供的自定义菜单样式将取代原始选择菜单的样式，使选择菜单在显示时发生变化。

将选择菜单的类型设为自定义的方法很简单，只要在＜select＞元素中，将“data-native-menu”属性值设置为“false”，该选择菜单便成为一个自定义的菜单。这种类型的菜单由按钮和菜单两部分组成，当用户单击按钮时，对应的菜单选择器将会自动打开，选其中某一项后，菜单自动关闭，被单击按钮的值将自动更新为菜单中用户所选择中的值。

实例4-8　用两种方式显示选择菜单

1.功能说明

新建一个HTML页面，在选择菜单组容器中添加两个选择菜单，一个用于选择“年”，另一个用于选择“月”。用户单击按钮并选中某

选项后，页面中将显示被选中的选项值。

2.实现代码

新建一个HTML页面4-8.htm，加入代码如代码清单4-8所示。

代码清单4-8　选择菜单

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile选择菜单</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
$（function（）{
var strYearVal="";
var strMonthVal="";
var objSelY=$（"#selY"）;
var objSelM=$（"#selM"）;
//设置复选框选择时的值
objSelY.bind（"change", function（）{
if（objSelY.val（）！="年份"）{
strYearVal=objSelY.val（）+", ";
}
$（"#pTip"）.html（strYearVal+strMonthVal）;
}）
objSelM.bind（"change", function（）{
if（objSelM.val（）！="月份"）{
strMonthVal=objSelM.val（）+", ";
}
$（"#pTip"）.html（strYearVal+strMonthVal）;
}）
}）
```

```
    </script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content">
    <feldset data-role="controlgroup"data-type="horizontal">
    <select name="selY"id="selY"data-native-menu="false">
    <option>年份</option>
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    </select>
    <select name="selM"id="selM"data-native-menu="false">
    <option>月份</option>
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    <option value="10">10</option>
    <option value="11">11</option>
    <option value="12">12</option>
    </select>
    <p id="pTip"></p>
    </feldset>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></</div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-8所示。

图　4-8　以不同的方式展现选择菜单的效果

4.源码分析

在本实例中，两个选择菜单被“data-role”属性值为“controlgroup”的＜fieldset＞元素所包裹，因此，以一个整体组的形式显示在页面中。通过设置＜fieldset＞元素的“data-type”属性值，可以调节选择菜单组展现的方式。

由于选择菜单将“data-native-menu”属性值设置为“false”，因此，它变成了一个自定义类型的选择菜单。用户单击年份按钮时，页面中将弹出一个菜单形式的对话框，用户在对话框中选择某选项后，触发选择菜单的“change”事件，该事件中将在页面中显示用户

所选择的菜单选择值，同时对话框自动关闭，并更新对应菜单按钮中所显示的内容。

说明　在编写选择菜单“change”事件的代码时，应该首先检测用户是否选择了某值，如果没有选择，应作相应的提示信息或检测，以确保获取数据的完整性。

4.2.7　多项选择菜单

与原生的页面中的选择菜单不同，jQuery Mobile中的选择菜单还可以通过设置“multiple”属性值，实现菜单的多项选择。如果将某个选择菜单的“multiple”属性值设置为“true”，单击该按钮弹出的菜单对话框中，全部菜单选项的右侧将会出现一个可勾选的复选框，用户通过单击该复选框，可以选中任意多个选项。选择完成后，单击左上角的“关闭”按钮，已弹出的对话框将关闭，对应的按钮自动更新为用户所选择的多项内容值。

实例4-9　多项选择菜单

1.功能说明

将实例4-8中选择菜单按钮的“multiple”属性值设置为“true”，从而使用“年”、“月”这两个选择菜单变成多项选择。用户选择后，选中的内容值将显示在对应按钮中。

2.实现代码

新建一个HTML页面4-9.htm，加入代码如代码清单4-9所示。

代码清单4-9　选择菜单

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile多项选择菜单</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<feldset data-role="controlgroup">
<select name="selY"id="selY"
data-native-menu="false"multiple="true">
<option>年份</option>
<option value="2011">2011</option>
<option value="2010">2012</option>
</select>
<select name="selM"id="selM"
data-native-menu="false"multiple="true">
<option>月份</option>
<option value="jan">1</option>
<option value="dec">2</option>
<option value="feb">3</option>
<option value="mar">4</option>
<option value="apr">5</option>
<option value="may">6</option>
<option value="jun">7</option>
<option value="jul">8</option>
<option value="aug">9</option>
<option value="sep">10</option>
<option value="oct">11</option>
<option value="nov">12</option>
</select>
</feldset>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
```

```
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-9所示。

图　4-9　多项选择菜单执行时的效果

4.源码分析

在本实例中，在用户选择后，多项选择菜单对应的按钮中不仅会显示所选择的内容值，而且超过2项选择时，在下拉图标的左侧还会有一个圆形的标签，在标签中显示用户所选择的选项总数，另外，在弹

出的菜单选择对话框中，选择某一个选项后，对话框不会自动关闭，必须单击左上角圆形的“关闭”按钮，才算完成一次菜单的选择。

单击“关闭”按钮后，各项选择的值将会变成一行用逗号分隔的文本，显示在对应按钮中。如果按钮长度不够，多余部分将显示成省略号。

与所有的表单元素一样，无论是选择菜单还是多项选择菜单，如果使用JavaScript代码控制选择菜单所选中的值，必须对该选择菜单刷新一次，从而使对应的样式与选择项同步，如下列代码所示：

```
$（function（）{
$（"#selM"）[0].selectedIndex=2;
$（"#selM"）.selectmenu（"refresh"）;
}）
```

上述代码将使月份的第2项为选中状态，同时刷新整个选择菜单，使选择值与样式同步。

4.3　列表

在jQuery Mobile中，如果在＜ul＞元素中，将“data-role”属性值设置为“listview”，便形成了一个无序的列表，并且将会对列表渲染对应的样式，如列表的宽度与屏幕同比缩放，在列表选项的最右侧有一个带箭头的链接图标；另外，列表还有包括许多种类，如基本列表、嵌套列表、编号列表等，同时，还可以对列表中选项的内容进行分割与格式化。

4.3.1　基本列表

一个＜ul＞元素一旦被定义为列表，jQuery Mobile将对该列表进行对应样式的渲染，列表中的选项也变得易于触摸。如果单击某选项，将会通过AJAX的方式异步请求一个对应的URL地址，并在DOM中创建一个新的页面，借助默认的切换效果进入该页面中。

实例4-10　基本列表

1.功能说明

新建一个HTML页面，在页面中添加一个“data-role”属性值为“listview”的＜ul＞元素作为列表容器；另外，在容器中再添加两个选项内容分别为“图书”和“音乐”。

2.实现代码

新建一个HTML页面4-10.htm，加入代码如代码清单4-10所示。

代码清单4-10　基本列表

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile基本列表</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview">
<li><a href="#">图书</a></li>
<li><a href="#">音乐</a></li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-10所示。

图　4-10　只有两个选项的基本列表效果

4.源码分析

在本实例中，jQuery Mobile通过自带的样式对＜ul＞元素进行了渲染，使列表中的各选项拉长，更加容易触摸。选项最右侧的圆形带箭头的链接图标示意用户该选项有链接。单击时，通过切换页面的方式，跳转到各选项＜a＞元素中“href”属性值所指的页面中。

4.3.2　嵌套列表

在jQuery Mobile中，＜ul＞、＜ol＞元素不仅可以被渲染成列表，而且该列表还可以进行嵌套。实现的方法是在父列表＜ul＞、＜ol＞元素的＜li＞标签中，添加子列表＜ul＞或＜ol＞元素，形成嵌套列表的格局。当用户单击父列表中的某个选项时，jQuery Mobile会自动生成一个包含子列表＜ul＞或＜ol＞元素全部内容的新页面，页面的主题则为父列表的标题内容。

1.功能说明

新建一个HTML页面，添加一个＜ul＞元素，并在该元素中增加两个＜li＞选项元素，主题分别为“图书”、“音乐”；同时，在两个＜li＞元素中，分别添加另外两个与之对应的＜ul＞列表元素作为子列表。单击父列表中某个选项时，将切换至对应的子列表页面中。

2.实现代码

新建一个HTML页面4-11.htm，加入代码如代码清单4-11所示。

代码清单4-11　嵌套列表

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

```
    <title>jQuery Mobile嵌套列表</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <ul data-role="listview">
    <li>
    <h3>图书</h3>
    <p>一本好书，就是一个良师益友。</p>
    <ul>
    <li><a href="#">计算机</a></li>
    <li><a href="#">社科</a></li>
    </ul>
    </li>
    <li>
    <h3>音乐</h3>
    <p>好的音乐可以陶冶人的情操。</p>
    <ul>
    <li><a href="#">流行</a></li>
    <li><a href="#">通俗</a></li>
    </ul>
    </li>
    </ul>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-11所示。

图　4-11　嵌套列表的效果

4.源码分析

在本实例中，当用户单击父列表框中某个选项内容时，将弹出一个新建的页面，页面中显示与父列表相对应的子列表内容。这个动态生成的子列表的默认主题样式为“蓝色”，以区分于父列表，表示这是一个二级列表。

当然，这种列表的嵌套可以有很多层，但从视觉效果来说，建议最好不超过三层。无论有多少层，jQuery Mobile都会自动处理页面打开与链接的效果。

4.3.3　有序列表

与＜ul＞无序列表元素相对应，使用＜ol＞元素可以创建一个有序的列表。在有序列表中，借助排列的编号顺序可以展现一种有序的列表效果。

在有序列表显示时，jQuery Mobile会优先使用CSS样式给列表添加编号。如果浏览器不支持这种CSS样式，jQuery Mobile将会调用JavaScript中的方法向列表写入编号，以确保有序列表的效果可以兼容各种浏览器。

实例4-12　有序列表

1.功能说明

新建一个HTML页面，添加一个＜ol＞元素作为有序列表的容器，在容器中通过＜li＞元素显示不同类别图书的销售排行榜。

2.实现代码

新建一个HTML页面4-12.htm，加入代码如代码清单4-12所示。

代码清单4-12　有序列表

```
＜! DOCTYPE html＞
```

```
<html>
<head>
<title>jQuery Mobile有序列表</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ol data-role="listview">
<li><a href="#">计算机</a></li>
<li><a href="#">文艺</a></li>
<li><a href="#">社科</a></li>
</ol>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-12所示。

图　4-12　有序列表的效果

4.源码分析

在本实例中，jQuery Mobile使用＜ol＞元素可以实现一个有序列表，该项功能常用于商品排行榜的显示；另外，由于jQuery Mobile已全面支持HTML 5的新特征和属性，原则上＜ol＞元素中的“start”属性是允许使用的，表示规定起始数字；但jQuery Mobile中考虑到浏览器的兼容性，对该属性暂时不支持。此外，＜ol＞元素的“type”、“compact”属性在HTML 5中不建议使用，且jQuery Mobile对这两个属性也是不支持的。

4.3.4　分割按钮列表

在jQuery Mobile的列表中，有时需要对选项内容做两个不同的操作，这时，需要对选项中的链接按钮进行分割。实现分割的方法非常简单，只需要在＜li＞元素中再增加一个＜a＞元素，便可以在页面中实现分割的效果。

分割后的两部分之间通常有一条竖直的分割线，分割线左侧为缩短长度后的选项链接按钮，右侧为后来增加的＜a＞元素。该元素的显示效果只是一个带图标的按钮，可以通过设置＜ul＞元素中“data-split-icon”属性的值，来改变该按钮中的图标。

实例4-13　分割按钮列表

1.功能说明

新建一个HTML页面，添加一个＜ul＞元素，并在＜ul＞元素中增加两个＜li＞元素。在这两个＜li＞元素中，分别采用分割按钮的方式，图文并茂地展现两本图书的资料信息。

2.实现代码

新建一个HTML页面4-13.htm，加入代码如代码清单4-13所示。

代码清单4-13　分割按钮列表

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile分割按钮列表</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview"data-split-icon="gear"
data-split-theme="d">
<li>
<a href="#">
<img src="Images/2011年作品.jpg"/>
<h3>HTML 5实战</h3>
<p>一本全面介绍HTML 5新增特征与API的原创图书。</p>
</a>
<a href="#"data-rel="dialog"
data-transition="slideup">
2011年作品
</a>
</li>
<li>
<a href="#">
<img src="Images/2010年作品.jpg"/>
<h3>jQuery权威指南</h3>
<p>通过一个个精选的实例详细完整地介绍jQuery的方方面面。</p>
</a>
<a href="#"data-rel="dialog"
data-transition="slideup">
2010年作品
</a>
</li>
</ul>
```

```
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-13所示。

图　4-13　列表中分割按钮的效果

4.源码分析

在本实例中，向<li>元素中多添加一个<a>元素后，便可以通过一条分割线将列表选项中的链接按钮分割成两部分。其中，分割线

左侧区域的长度可以随着移动终端设备分辨率的不同进行等比缩放；而右侧区域仅是一个只有图标的链接按钮，它的长度是自动适应且固定不变的。

说明　目前在jQuery Mobile中，列表中的分割只支持分成两部分，即在＜li＞元素中，只允许有两个＜a＞元素出现，如果添加两个以上的元素，会将最后一个元素作为分割线右侧部分。

4.3.5　分割列表项

在jQuery Mobile中，除了可以分割列表项中的按钮外，还可以对列表进行分割。这里所说的分割其实质是分类、归纳的意思，即在列表中，通过分割项将同类的列表项组织起来，形成相互独立的同类列表组，组的下面是一个个列表项。

实现分割列表项的方法很简单，只需要在分割的位置增加一个＜li＞元素，并将该元素的“data-role”属性值设置为“list-divider”，表示该＜li＞元素是一个分割列表项。默认情况下，普通列表项的主题色为“浅灰色”，分割列表项的主题色为“蓝色”，两者通过主题颜色的区别，形成层次上的包含效果。

实例4-14　分割列表项

1.功能说明

新建一个HTML页面，添加一个＜ul＞列表元素，并增加两个＜li＞元素作为分割列表项，一个用于“图书”分类，另外一个用于“音乐”分类，并分别显示各分割列表项下的同类列表项。

2.实现代码

新建一个HTML页面4-14.htm，加入代码如代码清单4-14所示。

代码清单4-14　分割列表项

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile分割列表项</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview">
<li data-role="list-divider">图书</li>
<li><a href="#">计算机</a></li>
<li><a href="#">社科</a></li>
<li><a href="#">文艺</a></li>
<li data-role="list-divider">音乐</li>
<li><a href="#">流行</a></li>
<li><a href="#">通俗</a></li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-14所示。

图　4-14　分割列表项的效果

4.源码分析

在本实例中，给＜ul＞元素添加一个“data-role”属性值为“list-divider”的＜li＞元素，便在列表中自动形成一个不同主题色的分割列表项。该列表项的主题颜色也可以通过修改＜ul＞元素中的“data-divider-theme”属性值进行修改。

分割列表项的作用只是将列表中的选项内容进行分类归纳，因此不要滥用；且在一个列表中不宜过多使用分割列表项，每一个分割列表项下的列表项数量不要太少。

4.3.6　图标与计数器

在jQuery Mobile的列表＜ul＞或＜ol＞元素中，如果将一个＜img＞元素作为＜li＞元素中的第一个子元素，那么，该＜img＞图片元素将自动缩放成一个边长为“80”像素的正方形，作为图片的缩略图，详细实例代码见实例4-13。

但是，如果＜img＞元素中的图片是一个图标，则需要给该元素添加一个名称为“ui-li-icon”的类别属性，才能在列表的最左侧正常显示该图标。另外，如果想在列表项中的最右侧显示一个计数器，只要添加一个＜span＞元素，并在该元素中增加一个名称为“ui-li-count”的类别属性即可。

实例4-15　列表中的图标与计数器

1.功能说明

新建一个HTML页面，添加一个＜ul＞列表元素。在列表中，增加一个显示图标的＜img＞元素和显示计数器的＜span＞元素，并分别设置对应的类别属性，显示在页面中。

2.实现代码

新建一个HTML页面4-15.htm，加入代码如代码清单4-15所示。

代码清单4-15　列表中的图标与计数器

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile缩略图与计数器</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview">
<li>
<a href="#">
<img src="Images/01.png"alt="图书"
class="ui-li-icon"/>图书
<span class="ui-li-count">3</span>
</a>
</li>
<li>
<a href="#">
<img src="Images/02.png"alt="音乐"
class="ui-li-icon"/>音乐
<span class="ui-li-count">2</span>
</a>
</li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-15所示。

图　4-15　含有图标和计数器的列表项效果

4.源码分析

在本实例中，＜img＞元素所放置的图标尺寸大小控制在“16”像素以内。如果图标尺寸过大，虽然也会进行缩放，但将会与图标右侧的标题部分不协调，从而影响到用户的体验；另外，如果计数器＜span＞元素中显示的内容过长，该元素将会固定右侧位置自动向左侧伸展，直到完全显示为止。

4.3.7　内容格式化与计数器

jQuery Mobile支持以HTML语义化的元素（如＜span＞、＜h＞、＜p＞）显示列表中所需的内容格式。通常情况下，使用＜span＞元素，并添加一个名为“ui-li-count”的类别，可以在列表项的右侧生成一个计数器。使用＜h＞元素来突显列表项中显示的内容，＜p＞元素用于减弱列表项中显示的内容，两者结合，可以使列表项中显示的内容具有层次关系。如果要增加补充信息，例如日期，可以在显示的＜p＞元素中，添加一个名为“ui-li-aside”的类别。

实例4-16　内容格式化与计数器

1.功能说明

新建一个HTML页面，创建一个＜ul＞列表，并在列表中通过使用＜h＞、＜span＞、＜p＞元素，并结合名为“ui-li-count”、“ui-li-aside”的类别，显示两本图书的相应信息。

2.实现代码

新建一个HTML页面4-16.htm，加入代码如代码清单4-16所示。

代码清单4-16　内容格式化与计数器

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile内容格式化与计数器</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview">
<li data-role="list-divider">2010年、2011年作品集
<span class="ui-li-count">2</span></li>
<li>
<a href="#"><h3>2011年作品</h3>
<p><strong>HTML 5实战</strong></p>
<p>一本全面介绍HTML 5新增特征与API的原创图书。</p>
<p class="ui-li-aside">
<strong>2011.01</strong>出版
</p>
</a>
</li>
<li>
<a href="#"><h3>2010年作品</h3>
<p><strong>jQuery权威指南</strong></p>
<p>通过一个个精选的实例详细完整地介绍jQuery的方方面面。</p>
<p class="ui-li-aside">
<strong>2010.01</strong>出版
</p>
</a>
</li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图4-16所示。

图　4-16　内容格式化后的列表项效果

4.源码分析

在本实例中，通过对列表项中的内容进行格式化，可以将大量的信息层次清晰地显示在页面中，实现的效果如上述示意图4-16所示。

如果想使用搜索方式过滤列表项中的标题内容，可以将＜ul＞元素的“data-filter”属性值设为“true”，jQuery Mobile将会在列

表的上方自动增加一个搜索框。当用户在搜索框中输入字符时，jQuery Mobile将会自动过滤掉不包含搜索字符内容的列表项。

与其他元素类似，如果通过JavaScript代码添加列表中的列表项，则需要调用列表的刷新（refresh）方法，更新对应的样式并将添加的列表项同步到原有列表中，代码如下：

```
$（'ul'）.listview（'refresh'）;
```

上述代码的功能是：对标签是＜ul＞的元素整体刷新一次。

4.4　本章小结

本章先从按钮讲起，由浅入深地介绍了表单中的各个常用组件在jQuery Mobile中的使用方法；并且结合一个个简单的示例，完整详细地介绍了jQuery Mobile重要组件——列表所涉及方方面面的知识，使读者在了解第3章的基础之上，结合本章节的学习内容，全面理解与掌握jQuery Mobile中各种重要组件的使用方法与技巧。

第5章　jQuery Mobile主题

本章内容

主题的定义及使用场景

列表与表单元素的主题

工具栏与页面内容的主题

本章小结

主题是一个Web站点或应用的皮肤，是最直接面对用户的界面，关系到用户的最终体验，其重要性不言而喻。在jQuery Mobile中，由于每一个页面中的布局和组件都被设计成一个全新的面向对象的CSS框架，整个站点或应用的视觉风格可以通过这个框架得到统一。统一后的视觉设计主题称为jQuery Mobile主题样式系统，它有以下几个特点：

文件的轻量级：使用CSS 3来处理圆角、阴影和颜色渐变的效果，而没有使用图片，大大减轻了服务器的负担。

主题的灵活度高：框架系统提供了多套可选择的主题和色调，并且每套主题之间都可以混搭，丰富视觉纹理的设计。

自定义主题便捷：除使用系统框架提供的主题外，还允许开发者自定义自己的主题框架，用于保持设计的多样性。

图标的轻量级：在整个主题框架中，使用了一套简化的图标集，它包含了绝大部分在移动设备中使用的图标，极大减轻了服务器对图标处理的负荷。

从上述jQuery Mobile主题的特点我们不难看出，jQuery Mobile中的每个应用程序或组件都提供了样式丰富、文件轻巧、处理便捷的样式主题，极大方便了开发人员的使用。

5.1　主题的定义及使用场景

显而易见，在jQuery Mobile中，组件和页面布局的主题定义是通过使用一套完整的CSS框架来实现的，在这套CSS框架中包括两个重要组成部分：

结构：用于控制元素的在屏幕中显示的位置、填充效果、内外边距等。

主题：用于控制元素的颜色、渐变、字体、圆角、阴影等视觉效果，并包含了多套的色板，每套色板中都定义了列表项、按钮、表单、工具栏、内容块、页面的全部视觉效果。

jQuery Mobile中，CSS框架中的结构和主题是分离的，因此只要定义一套结构就可以反复与一套或多套主题配合或混合使用，从而实现页面布局和组件主题多样化的效果。

5.1.1　默认主题

在jQuery Mobile中，系统自带了5套主题样式，分别用字母“a”、“b”、“c”、“d”、“e”进行引用，各主题的使用场景说明如表5-1所示。

表 5-1　jQuery Mobile 中主题使用场景

主题字母名称	使用场景说明
a	整体色为黑色，是使用级别最高的主题
b	整体色为蓝色，使用级别仅次于 a 级主题
c	整体为基准灰色，系统默认主题
d	整体色为灰白色，用于 c 级备用的主题
e	整体色为金黄色，用于强调、突出性主题

除使用上述系统自带的5种主题外，开发者还可以很方便地修改系统主题中的各类属性值，并快捷地自定义属于自己的主题，相关内容将在接下来的来章节中进行详细介绍。

在默认情况下，jQuery Mobile中头部栏与尾部栏的主题是“a”字母，因为“a”字母代表最高的视觉效果。如果需要改变某组件或容器当前的主题，只需要将它的“data-theme”属性值设置成主题对应的样式字母即可。

实例5-1　通过下拉框选择并保存主题

1.功能说明

新建一个HTML页面，并在内容区域中创建一个下拉列表框，用于选择系统自带的5种类型主题，当用户通过下拉列表框选择某一主题时，使用“cookie”的方式保存所选择的主题值，并在刷新页面时，将内容区域的主题设置成“cookie”所保存的主题值。

2.实现代码

新建一个HTML页面5-1.htm，加入代码如代码清单5-1所示。

代码清单5-1　选择并保存主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile选择并保存主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.cookie.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script type="text/javascript">
$（function（）{
var objSelTheme=$（"#selTheme"）;
objSelTheme.bind（"change", function（）{
//如果选择的值不为空
if（objSelTheme.val（）！=""）{
//使用cookie保存所选择的主题
$.cookie（"StrTheme", objSelTheme.val（）, {
path: "/", expires: 7
}）
//重新刷新一次页面，运用主题
window.location.reload（）;
}
}）
}）
//如果主题不为空，则运用主题
if（$.cookie（"StrTheme"））{
$.mobile.page.prototype.options.theme=
$.cookie（"StrTheme"）;
}
</script>
</head>
<body>
<div data-role="page">
```

```
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content">
    <select name="selTheme"id="selTheme"
    data-native-menu="false">
    <option value="">选择主题</option>
    <option value="a">主题a</option>
    <option value="b">主题b</option>
    <option value="c">主题c</option>
    <option value="d">主题d</option>
    <option value="e">主题e</option>
    </select>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-1所示。

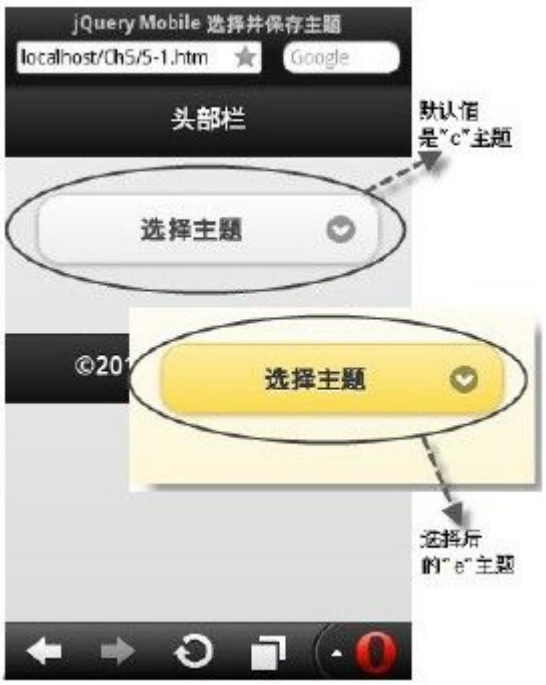


图　5-1　通过下拉框选择并保存主题

4.源码分析

在本实例中，首先引用了一个cookie插件文件jquery.cookie.js。在下拉列表框的“change”事件中，当用户选择的主题值不为空时，调用插件中的方法，将用户选择的主题值保存至名称为“StrTheme”的cookie变量中；当页面刷新或重新加载时，如果名为“StrTheme”的cookie变量不为空，那么，通过“$.mobile.page.prototype.options.theme=$.cookie（"StrTheme"）”语句，将页面内容区域的主题设置为用户所选择的主题值。

由于使用cookie方式保存页面的主题值，即使是关闭浏览器重新再打开时，用户所选择的主题依然有效，除非手动清除cookie值或对应的cookie值到期后自动失效，页面才会自动恢复到默认的主题值。

5.1.2　修改默认主题

虽然jQuery Mobile中提供了5种系统自带的主题，但大部分开发人员还是希望可以根据应用的需求，修改相应的主题结构和色调。实现的方法也很简单，只要打开定义主题的CSS文件jquery.mobile-1.0.1.min.css，找到需要修改的元素，调整对应的属性值，然后保存文件即可。

实例5-2　修改默认主题

1.功能说明

打开jQuery Mobile用于控制页面主题的CSS文件jquery.mobile-1.0.1.min.css，找到类别名称“ui-bar-a”，将它对应的“color”属性值修改为“blue”，并保存；然后，新建一个HTML页面，分别添加头部栏、内容区域和尾部栏，浏览该页面时，工具栏的字体颜色已由白色变成蓝色。

2.实现代码

新建一个HTML页面5-2.htm，加入代码如代码清单5-2所示。

代码清单5-2　修改默认主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile修改默认主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<p>导航条的字体颜色发生了变化</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在jQuery Mobile框架的jquery.mobile-1.0.1.min.css文件中，找到“ui-bar-a”类别名称，将该类别中对应的“color”属性值修改为“blue”，部分代码如下：

```
.ui-bar-a{
border: 1px solid#2A2A2A;
background: #111111;
color: blue;
font-weight: bold;
text-shadow: 0-1px 1px#000000;
background-image: -moz-linear-gradient（top，#3c3c3c，
#111111）；
……省略部分代码}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-2所示。

图　5-2　修改默认主题样式的效果

4.源码分析

在本实例中，被修改的系统主题类别名称“ui-bar-a”有着特定结构，其中，字符“-a”代表该类别属于系统主题“a”级别，定义系统主题的类别结构都是一样，仅是色调不同而异；另外，字符“-bar”表示该类别是用于控制“header”和“footer”容器显示的色调效果。

jquery. mobile-1.0.1.min.css文件的前600行以内，均是定义系统5种主题的色调和元素的一些通用属性，如按钮圆角、阴影等，如下代码所示：

```
.ui-btn-corner-tl{
-moz-border-radius-topleft: 1em;
-webkit-border-top-left-radius: 1em;
border-top-left-radius: 1em;
}
……省略部分代码
```

这些类别都是通用的，不依赖于任何指定的色调，各自独立地实现特定的效果。考虑到各种浏览器对CSS 3的兼容性不同，每种类别都要重复编写3行功能相同的代码，开发者可以根据需要，任意修改这些类别中实现效果的属性值。

5.1.3　自定义主题

上一节介绍了如何修改系统自带的主题，实现的方法十分简单。但由于是对原CSS样式文件进行的修改，每次当版本更新后，都需要对新版本的文件重新覆盖修改后的代码，操作不是很方便。为此，可以重新编写一个单独的CSS文件，专门用于定义页面与组件的主题样式。该文件与系统文件同时并存，实现用户自定义主题的功能。

实例5-3　自定义主题

1.功能说明

新建一个HTML页面，先在页面中引用自定义的一个CSS主题文件jquery.mobile-f.css，然后在“page”容器中，将“data-theme”属性值设置为“f”，表示使用自定义CSS文件中的主题；并在内容区域中添加一个“collapsible”容器，显示自定义主题“f”的页面效果。

2.实现代码

新建一个HTML页面5-3.htm，加入代码如代码清单5-3所示。

代码清单5-3　自定义主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile自定义主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/jquery.mobile-f.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"data-theme="f">
<div data-role="header"><h1>头部栏标题</h1></div>
<div data-role="collapsible"data-collapsed="false">
<h3>点击查看更多</h3>
<p>一位优秀的Web端工程师，不仅要有过硬的技术，而且要有执着、沉稳
的品质。</p>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

自定义jquery.mobile-f.css文件的代码，如下所示：

```
.ui-btn-up-f{
border: 1px solid#222;
background: #02BA19;
font-weight: bold;
color: #fff;
text-shadow: 0-1px 1px#000;
background-image: -moz-linear-gradient（top，#0E5D90，
#02A3EF）；
background-image: -webkit-gradient（linear, left top, left
bottom，
color-stop（0，#0E5D90），
```

```
    color-stop（1, #02A3EF））;
    -ms-flter: "progid: DXImageTransform.Microsoft
    .gradient（startColorStr='#0E5D90',
EndColorStr='#02A3EF'）";
    }
    .ui-btn-up-crush a.ui-link-inherit{
    color: #fff;
    }
    .ui-btn-hover-f{
    border: 1px solid#000;
    background: #444444;
    font-weight: bold;
    color: #000;
    text-shadow: 0-1px 1px#fff;
    background-image: -moz-linear-gradient（top, #FFFFFF,
#778899）;
    background-image: -webkit-gradient（linear, left top, left
bottom,
    color-stop（0, #FFFFFF）,
    color-stop（1, #778899））;
    -ms-flter: "progid: DXImageTransform.Microsoft
    .gradient（startColorStr='#FFFFFF',
EndColorStr='#778899'）";
    }
    .ui-btn-hover-f a.ui-link-inherit{
    color: #fff;
    }
    .ui-btn-down-f{
    border: 1px solid#000;
    background: #02BA19;
    font-weight: bold;
    color: #fff;
    text-shadow: 0-1px 1px#000;
    background-image: -moz-linear-gradient（top, #778899,
#FFFFFF）;
    background-image: -webkit-gradient（linear, left top, left
bottom,
    color-stop（0, #778899）,
    color-stop（1, #FFFFFF））;
    -ms-flter: "progid: DXImageTransform.Microsoft
    .gradient（startColorStr='#778899',
EndColorStr='#FFFFFF'）";
    }
    .ui-btn-down-f a.ui-link-inherit{
    color: #000;
```

```
    }
    .ui-btn-up-f,
    .ui-btn-hover-f,
    .ui-btn-down-f{
    font-family: CandelaBookItalic, Helvetica, Arial, sans-
serif;
    text-decoration: none;
    }
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-3所示。

图　5-3　自定义主题样式的效果

4.源码分析

在本实例中，为了自定义按钮的主题，需要重新定义“ui-btn-up-f”、“ui-btn-hover-f”、“ui-btn-down-f”三个类别。它们的基本结构都相同，本实例中自定义的主题只是修改颜色，因此，只要将这三个类别的“background”、“background-image”、“-ms-filter”属性值进行修改即可。在修改时，后两个属性是设置按钮背景的渐变效果，因此，需要指定渐变色的开始值与结束值。

说明　在设置按钮渐变效果时，考虑到浏览器对样式的兼容性不同，需要编写三种类别来应对不同类型的浏览器，第一个“background-image”属性是专门针对火狐浏览器来编写的，第二个“background-image”属性是专门针对WebKit内核浏览器来编写的，第三个“-ms-filter”属性显然是专门针对微软浏览器来编写的，详细实现过程如代码清单中加粗部分所示。

5.2　列表与表单元素的主题

在jQuery Mobile中，除可以修改系统主题或自定义页面主题外，还可以通过“data-theme”属性设置或变更列表与表单元素的主题，并且能在页面的元素中实现主题的混搭效果；另外，可以通过类别来设置按钮特有的激活样式的主题，接下来我们逐一进行详细的介绍。

5.2.1　列表主题

jQuery Mobile的列表默认的框架主题是“c”，默认的分隔选项主题是“b”。当然，也可以通过“data-theme”和“data-divider-theme”属性，分别修改列表框架和分隔选项的默认主题。此外，列表中还允许添加用于显示计数器效果的图标，可以通过“data-count-theme”属性来修改它在列表中显示的主题。

实例5-4　列表容器混搭主题

1.功能说明

新建一个HTML页面，添加一个“listview”列表容器，在容器中增加两个分隔选项，每个分隔选项下分别添加三个和二个子选项，并在各个子选项中添加计数器效果的图标。由于列表容器中设置了不同组件的主题，最后列表容器混搭主题的效果将显示在页面中。

2.实现代码

新建一个HTML页面5-4.htm，加入代码如代码清单5-4所示。

代码清单5-4　列表主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile列表主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<ul data-role="listview"data-theme="c"
data-divider-theme="b"data-count-theme="e">
<li data-role="list-divider">图书</li>
<li><a href="#">计算机
<span class="ui-li-count">100</span></a></li>
<li data-theme="d"><a href="#">社科
<span class="ui-li-count">101</span></a></li>
<li><a href="#">文艺
<span class="ui-li-count">102</span></a></li>
<li data-role="list-divider">音乐</li>
<li><a href="#">流行
<span class="ui-li-count">103</span></a></li>
<li><a href="#">通俗
<span class="ui-li-count">104</span></a></li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
```

```
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-4所示。

图　5-4　列表中各组件主题混搭的效果

4.源码分析

在本实例中，虽然各组件、元素的主题都可以应用到列表中，但是有些标签的主题只有在“listview”元素的属性中才能设置，如“data-divider-theme”、“data-count-theme”，因为这些元素具有整体性，不太适合单个设置。

另外，还可以利用JavaScript代码，设置或修改列表中元素的主题，如下代码所示：

```
$.mobile.listview.prototype.options.dividerTheme="a";
```

上述代码可以将列表中的分隔项的主题全部设置为字母“a”级别。

5.2.2　表单主题

jQuery Mobile提供了丰富的主题系统来运用至表单元素中，开发者可以轻松地定制属于自己的主题风格。通常情况下，表单容器采用一个主题来定义表单中所有的元素，这样做的好处在于，可以用较少量的代码统一表单的样式风格；表单中单个元素也可以通过修改“data-theme”主题属性，来自定义属于元素自身的主题。

实例5-5　改变表单主题

1.功能说明

新建一个HTML页面，在内容区域中添加一个“text”、“select”表单元素和一个“checkbox”复选按钮组，分别用于输入字符、滑动选择开关键和进行多项选择，并在页面中使用同一种主题来展示这些放置在表单中的元素。

2.实现代码

新建一个HTML页面5-5.htm，加入代码如代码清单5-5所示。

代码清单5-5　表单主题

```
<! DOCTYPE html>
<html>
```

```
    <head>
    <title>jQuery Mobile表单主题</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content"data-theme="a">
    <label for="txta">文本输出框：</label>
    <input type="text"name="txta"id="txta"value=""/>
    <label for="sela">滑动开关：</label>
    <select name="sela"id="sela"data-role="slider">
    <option value="off">关</option>
    <option value="on">开</option>
    </select>
    <fieldset data-role="controlgroup"data-
type="horizontal">
    <legend>多项复选框：</legend>
    <input type="checkbox"name="chka"
    id="chka"class="custom"/>
    <label for="chka">b</label>
    <input type="checkbox"name="chkb"
    id="chkb"class="custom"/>
    <label for="chkb"><em>i</em></label>
    <input type="checkbox"name="chkc"
    id="chkc"class="custom"/>
    <label for="chkc">u</label>
    </fieldset>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

---

3. 页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-5所示。

图　5-5　不同表单主题的页面效果

4.源码分析

本实例执行后的页面效果如图5-5左图所示，如果将“content”容器的主题修改为“e”，其执行后的页面效果如图5-5右图所示。另外，实例中的各个表单元素都继承了“content”容器中所设置的“data-theme”主题风格。虽然如此，由于每一个表单元素都是一个独立的组件，在表单中，仍然可以使用组件中的“data-theme”属性单独设置主题。一旦设置完成，将采用“就近”的原则，忽略整体容

器的主题，采用组件自身“data-theme”属性设置的主题风格，比如，将文本输入框的“data-theme”属性值设置为“c”，那么，无论“content”容器的主题是什么，文本输入框在页面中始终以主题“c”的样式显示在页面中。

5.2.3　按钮主题

对于按钮而言，jQuery Mobile拥有丰富的主题风格与之匹配。按钮是将任意一个链接的“data-role”属性值设置为“button”值后形成的，因此，当该按钮被放置在任意主题的容器中，按钮本身将自动继承容器的主题，形成与容器相匹配的样式。例如，在一个主题为“a”的容器中添加一个按钮，该按钮的主题自动分配为“a”级别。

实例5-6　显示5种按钮主题风格

1.功能说明

新建一个HTML页面，并添加两个三列的网格容器，分别将按钮元素自带的5种系统主题风格显示在页面中。

2.实现代码

新建一个HTML页面5-6.htm，加入代码如代码清单5-6所示。

代码清单5-6　显示5种按钮主题风格

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile按钮主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div class="ui-grid-b">
<div class="ui-block-a">
<a href="#"data-role="button"
data-theme="a"data-icon="arrow-l">a</a>
</div><div class="ui-block-b">
<a href="#"data-role="button"
data-theme="b"data-icon="arrow-l">b</a>
</div><div class="ui-block-c">
<a href="#"data-role="button"
data-theme="c"data-icon="arrow-l">c</a>
</div>
</div>
<div class="ui-grid-b">
<div class="ui-block-a">
<a href="#"data-role="button"
data-theme="d"data-icon="arrow-l">d</a>
</div><div class="ui-block-b">
<a href="#"data-role="button"
data-theme="e"data-icon="arrow-l">e</a>
</div>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-6所示。

图　5-6　显示5种系统自带的按钮主题

4.源码分析

在本实例中，通过按钮元素的“data-theme”属性来设置按钮本身的主题。除此之外，还可以借助按钮外围容器的主题自动匹配按钮的主题风格，如下代码所示：

```
<div class="ui-body ui-body-a">
<a href="#"data-role="button">点击我</a>
</div>
```

上述代码中，按钮本身并没有设置主题。但是通过自动匹配外围 元素的主题“a”，在页面中按钮显示的也是主题“a”；当按

钮外围＜div＞元素的主题发生变化时，被包裹的按钮主题也将随之变化。

5.2.4　激活状态主题

jQuery Mobile中有一种单独的主题，称为激活状态主题。该主题通过在元素属性中添加一个“ui-btn-active”类别属性来实现。该主题不受任何其他框架或组件主题的影响，始终将蓝色作为该主题的显示色调。

实例5-7　激活状态主题

1.功能说明

新建一个HTML页面，在内容区域中增加两个按钮，一个显示与内容区域相匹配的主题，另一个设置为激活状态的主题。

2.实现代码

新建一个HTML页面5-7.htm，加入代码如代码清单5-7所示。

代码清单5-7　激活状态主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile激活状态主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
```

```
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content"data-theme="a">
    <a href="#"data-role="button"
    data-icon="arrow-l">默认状态</a>
    <a href="#"data-role="button"
    data-icon="arrow-l"class="ui-btn-active">选中状态</a>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-7所示，其中“选中状态”按钮就处在激活状态。

图　5-7　激活状态主题的效果

4.源码分析

在本实例中，通过给按钮添加一个名为“ui-btn-active”类别属性，将该按钮的主题设置为激活状态。该主题的风格是固定的，对于按钮而言，是蓝色的背景色，白色的字体，并且不受按钮本身自带主题的约束，即使在按钮元素中增加了“data-theme”属性值，也优先显示激活状态主题。

5.3　工具栏与页面内容的主题

通常情况下，jQuery Mobile中页面的工具栏默认为主题“a”级别，整体页面与内容区域默认为主题“c”级别。这种主题的混搭，一方面为了突显工具栏在页面中位置，另一方面也使页面首尾两端与内容区域之间存在一定的色差，来区分页面中显示的重点。

5.3.1　工具栏主题

在jQuery Mobile中，工具栏所包含的头部栏与尾部栏默认的主题是“a”级别，因此，在两者中增加的按钮与将自动匹配成“a”级别的主题。当然，也可以直接修改按钮的“data-theme”属性值，单独设置按钮的主题风格，下面通过一个详细的实例来进行介绍。

实例5-8　工具栏主题混搭效果

1.功能说明

新建一个HTML页面，分别增加两个头部栏和尾部栏。在第一个头部栏中放置两个默认主题的按钮，第二个头部栏中添加一个自定义主题的按钮；同时，在两个尾部栏中分别增加两个默认主题的按钮，并将工具栏中主题混搭的效果显示在页面中。

2.实现代码

新建一个HTML页面5-8.htm，加入代码如代码清单5-8所示。

代码清单5-8　工具栏主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile工具栏主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"data-position="inline">
<a href="#"data-icon="delete"
iconpos="notext">取消</a>
<h1>头部栏A</h1>
<a href="#"data-icon="arrow-r"
data-iconpos="right">保存</a>
</div>
<div data-role="header"
data-position="inline"data-theme="b">
<h1>头部栏B</h1>
<a href="#"data-icon="plus"data-theme="c">新建</a>
</div>
<div data-role="content"data-theme="e">
<p>这是正文部分</p>
</div>
<div data-role="footer"data-theme="a">
<a href="#"data-role="button"data-icon="arrow-l">前进</a
>
<a href="#"data-role="button"data-icon="arrow-r">后退</a
>
```

```
</div>
<div data-role="footer"data-theme="b">
<a href="#"data-role="button"
data-icon="arrow-l"data-theme="c">前进</a>
<a href="#"data-role="button"
data-icon="arrow-r"data-theme="c">后退</a>
</div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-8所示。

图　5-8　工具栏主题混搭的效果

4.源码分析

从本实例中我们可以看出，工具栏本身拥有默认的主题，开发人员也可以自定义它的主题。在工具栏中添加的按钮或文本，都继承了工具栏的主题风格，这样可以使整个工具栏的主题具有完整性和统一性。当然，开发人员也可以抛开工具栏中既有的主题，通过修改“data-theme”属性自定义工具栏中各元素的主题。

5.3.2　页面主题

jQuery Mobile内含丰富的主题系统，使开发人员在定义页面主题时拥有更多的选择。在设置页面主题时，应该修改页面“page”容器的“data-theme”属性值，这样可以确保所选择的主题能够覆盖整体页面的＜div＞或容器，但头部栏与尾部栏的主题依然是默认值“a”级别，这种多色板混合的主题风格，可以使页面形成各元素的最佳对比度，创造视觉上的美感。

实例5-9　改变页面主题

1.功能说明

新建一个HTML页面，先将页面中“page”容器的主题设为“e”级别，然后在内容区域中分别添加＜h＞、＜p＞、＜a＞元素。浏览该页面，查看这些元素继承容器主题后呈现的效果。

2.实现代码

新建一个HTML页面5-9.htm，加入代码如代码清单5-9所示。

代码清单5-9　页面主题

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>jQuery Mobile页面主题</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"data-theme="e">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content">
    <h3>jQuery Mobile主题架构</h3>
    <p>它提供了页面、工具栏、内容、表单主题、列表、按钮等多方面的主题
定制功能
    <a href="#"class="ui-link">详细</a>。
    </p>
    <a href="#"data-role="button"data-inline="true">进入</a
>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-9所示。

图　5-9　不同页面主题的效果

4. 源码分析

本实例在页面容器中分别使用了“e”、“a”、“b”三种主题，通过图5-9不难看出，页面容器内的全部元素都继承了页面的主题，展现出与主题风格相匹配的色调样式；而工具栏中的头部栏与尾部栏始终是保持默认主题“a”级别，但这并不影响它与整个页面主题的色调协调，反而使整体页面形成很强的色彩对比效果，进一步突显内容区域的重要位置，而这样的效果对系统所提供的5种默认主题都是有效的。

5.3.3　内容主题

与页面主题相比而言，内容主题所影响的范围小一些。内容主题所针对的范围仅局限于页面的“content”容器中，该容器之外的元素、背景色将停止匹配。正因为如此，才会出现内容区域中的色调与尾部栏之外色调不一致的现象。

此外，在内容区域“content”容器中，还可以通过“data-content-theme”属性设置内容折叠块中显示区域的主题，而这一主题是独立的、自定义的，不受限于内容区域“content”容器的主题。

实例5-10　折叠区域中的内容主题

1.功能说明

新建一个HTML页面，在内容区域“content”容器中分别添加两个内容折叠区，并各自设置显示区域的不同主题。浏览该页面，查看这两个内容折叠区域中各自展示的主题效果。

2.实现代码

新建一个HTML页面5-10.htm，加入代码如代码清单5-10所示。

代码清单5-10　内容主题

```
<! DOCTYPE html>
<html>
<head>
<title>jQuery Mobile内容主题</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content"data-theme="e">
<div data-role="collapsible"data-content-theme="c">
<h3>今天天气</h3>
<p>晴，气温<code>18～4℃</code>西风<em>3-4</em>级</p
>
</div>
<div data-role="collapsible"data-content-theme="b">
<h3>明天天气</h3>
<p>晴，气温<code>17～6℃</code>西风<em>4-5</em>级</p
>
</div>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图5-10所示。

图　5-10　折叠区中内容区域不同主题的效果

4.源码分析

在本实例中，整个“page”页面容器使用的是默认主题“a”级别，而“content”内容容器使用的是主题“e”级别。因此，在尾部栏之外的区域与页面内容显示区域之间存在色调不一致的效果，见图5-10。

此外，在“collapsible”内容折叠块容器中，设置“data-theme”和“data-content-theme”属性的值，可以修改内容折叠块的主题。前者针对的是折叠块标题部分，后者针对的是折叠块的内容显示区域部分，如果两个属性都不设置，将自动继承“content”内容容器所使用或默认的主题级别。

5.4　本章小结

本章先对jQuery Mobile提供的默认主题进行介绍，逐步深入地介绍了自定义主题和修改默认主题的方法；然后，使用了大量精选的实例，详细地介绍了jQuery Mobile中，各种重要元素或组件应用主题的方法与技巧。通过本章节的学习，读者能够全面了解并掌握jQuery Mobile中主题的概念与基本用法。

第6章　jQuery Mobile插件

本章内容

图片滑动浏览插件PhotoSwipe

图片幻灯片插件Camera

滚动选择时间插件Mobiscroll

搜索插件AutoComplete

日期对话框插件DateBox

简单对话框插件SimpleDialog

快捷标签插件ActionSheet

本章小结

与jQuery一样，jQuery Mobile同样具有很强的可拓展性，基于此，许多优秀的插件可以直接融入jQuery Mobile项目中。虽然针对jQuery Mobile移动开发的插件并不多，但是这些插件有一个共同的特点：依赖于jQuery主库插件，基于jQuery Mobile插件，面向移动终端设备应用或站点的使用。另外，这些插件无一例外都继承了jQuery插

件代码轻量级、效果优雅的特性，通过在移动项目中应用这些插件所实现的各项功能，可以使整个项目的进度加快、效率提高，并带来高效的用户体验，因此，深受广大移动项目开发者的青睐。

本章就来认识其中7种插件的使用方法。

6.1　图片滑动浏览插件PhotoSwipe

PhotoSwipe是在移动设置和触摸设备上使用的照片浏览插件，基于JavaScript、CSS、HTML代码，功能是以左右滑动的效果浏览每张图片。

PhotoSwipe插件是一个标准的JavaScript代码库，因此，它很容易被集成到移动网站项目中。目前，它兼容多个流行的移动设备浏览器和JavaScript代码库，常用于图片集的单张浏览。例如，在一个相册中存有10张照片，如果使用该插件，用户可以在浏览某一张照片时，通过手指的左右滑动操作，浏览当前照片的上一张和下一张照片。另外，在浏览某一张图片时，用户还可以通过单击尾部栏中的“  ”图标链接，实现相册集中全部照片自动播放浏览的功能。

接下来通过一个完整的实例介绍该插件在移动项目中的使用方法。在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

Js/Js6. 1/klass.min.js

Js/Js6. 1/photoswipe.js

Css/Css6. 1/photoswipe.css

下载地址：http：//www.photoswipe.com/latest/examples/04-jquery-mobile.html

实例6-1　图片滑动浏览插件PhotoSwipe的应用

1.功能说明

新建一个HTML页面，分别添加两个“page”容器，第一个容器放置相册集中的各个专题，单击“图书作品集”专题后，进入第二个容器展示该专题下的全部图片；单击某张图片后，便切换到该图片全屏浏览界面。在该界面中，通过左右滑动可以查看上一张或下一张图片。

2.实现代码

新建一个HTML页面6-1.htm，加入代码如代码清单6-1所示。

代码清单6-1　图片滑动浏览插件PhotoSwipe的应用

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>photoswipe插件应用程序</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/Css6.1/photoswipe.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/Js6.1/klass.min.js"
```

```
    type="text/javascript"><</script>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"><</script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"><</script>
    <script src="Js/Js6.1/photoswipe.js"
    type="text/javascript"><</script>
    <script type="text/javascript">
    $（function（）{
    $（"#bookpic"）.live（"pageshow", function（e）{
    //实例化滑动图片对象
    var currentPage=$（e.target）,
    options={},
    photoSwipeInstance=$（"ul.gallery a",
    e.target）.photoSwipe（options,
currentPage.attr（"id"））；
    return true;
    }）.live（"pagehide", function（e）{
    var currentPage=$（e.target）,
    photoSwipeInstance=PhotoSwipe
    .getInstance（currentPage.attr（"id"））；
    //分离图片列表与单个图片效果
    if（typeof photoSwipeInstance! ="undefned"＆＆
    photoSwipeInstance! =null）{
    PhotoSwipe.detatch（photoSwipeInstance）；
    }
    return true;
    }）
    }）
    </script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>相册集</h1></div>
    <div data-role="content">
    <ul data-role="listview"data-inset="true">
    <li data-role="list-divider">请选择所属专题</li>
    <li><a href="#bookpic">图书作品集</a></li>
    <li><a href="#">个人生活集</a></li>
    </ul>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
```

```
    <div data-role="page"id="bookpic"data-add-back-
btn="true">
    <div data-role="header"><h1>图书作品</h1></div>
    <div data-role="content">
    <ul class="gallery">
    <li><a href="Images/Img6.1/pic08.jpg"rel="external">
    <img src="Images/Img6.1/pic08.jpg"
    alt="2008年图书作品"title=""/>
    </a></li>
    <li><a href="Images/Img6.1/pic09.jpg"rel="external">
    <img src="Images/Img6.1/pic09.jpg"
    alt="2009年图书作品"title=""/>
    </a></li>
    <li><a href="Images/Img6.1/pic10.jpg"rel="external">
    <img src="Images/Img6.1/pic10.jpg"
    alt="2010年图书作品"title=""/>
    </a></li>
    <li><a href="Images/Img6.1/pic11.jpg"rel="external">
    <img src="Images/Img6.1/pic11.jpg"
    alt="2011年图书作品"title=""/>
    </a></li>
    </ul>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-1所示。

图　6-1　使用PhotoSwipe插件滑动浏览图片的效果

4.源码分析

在本实例中，用户在“相册集”页面中（如图6-1左侧图所示）单击“图书作品集”专题链接，切换到“图书作品”页面，如图6-1中间图所示。在该页面中，任意单击浏览某一张图片，页面将触发绑定的“pageshow”事件。在该事件中，先将获取的当前页保存在变量“currentPage”中，然后设置插件“options”值的内容，最后调用插件的“photoSwipe”方法，实例化图片的浏览页面（如图6-1右侧图所示），使该页面具有滑动浏览图片的特征。

当用户结束单张图片的浏览返回“图书作品”页面时，将触发绑定的“pagehide”事件，在该事件中，先获取已实例化的页面对象“photoSwipeInstance”，如果该对象的值不为“null”或类型不为

“undefined”，则调用插件中“detatch”方法，清除该对象所保存的内容，释放对应空间，便于下次使用，详细实现过程见代码中加粗部分所示。

在使用PhotoSwipe插件时，需要设置一个选项值“options”，该值是一个对象，它所包括的重要属性和说明如表6-1所示。

表 6-1　PhotoSwipe 插件中“options”对象的重要属性和说明

属性	描述	默认值
allowUserZoom	是否允许用户使用放大镜效果查看图片	true
autoStartSlidesho	是否在浏览图片时自动开启幻灯片模式	false
backButtonHideEnabled	当用户单击“后退”按钮，是否隐藏界面	true
captionAndToolbarAutoHideDelay	设置自动隐藏标题与工具栏的等待时间	5000 毫秒
captionAndToolbarHide	是否隐藏标题与工具栏	false
captionAndToolbarOpacity	标题与工具栏的透明度	0.8
enableDrag	是否开启拖动的方式浏览上一个或下一个图片	true
fadeInSpeed	浏览图片元素时，淡入的速度	250
fadeOutSpeed	浏览图片元素时，淡出的速度	250
slideshowDelay	使用幻灯片模式浏览图片时各图片间隔时间	3000

6.2　图片幻灯片插件Camera

Camera插件是一个基于jQuery插件的开源项目，功能是对所指定的图片集实现轮播的效果。在轮播过程中，用户可以查看每一张图片的主题信息，手动中止轮播过程，通过单击查看每一张被播放的图片。此外，轮播的图片还支持缩略图单击预览方式，方便用户以缩略图的方式浏览多张图片。

在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

Js/Js6. 2/jquery.easing.1.3.js

Js/Js6. 2/camera.min.js

Css/Css6. 2/camera.css

下载地址：http：//www.pixedelic.com/plugins/camera/

实例6-2　图片幻灯片插件Camera的应用

1.功能说明

新建一个HTML页面，并在页面的正文区域放置3张图片，调用Camera插件中的方法，实现3张图片轮流播放的动画效果。

2.实现代码

新建一个HTML页面6-2.htm，加入代码如代码清单6-2所示。

代码清单6-2　图片幻灯片插件Camera的应用

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>camera插件应用程序</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/Css6.2/camera.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/Js6.2/jquery.easing.1.3.js"
    type="text/javascript"></script>
    <script src="Js/Js6.2/camera.min.js"
    type="text/javascript"></script>
    <script type="text/javascript">
    $（function（）{
    $（'#camera_wrap_1'）.camera（{
    time: 1000,
    thumbnails: true
    }）
    }）;
    </script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header"><h1>幻灯图片</h1></div>
    <div class="camera_wrap
camera_azure_skin"id="camera_wrap_1">
    <div data-thumb="Images/Img6.2/thumb/list_1.jpg"
    data-src="Images/Img6.2/list_1.jpg">
```

```
    <div class="camera_caption fadeFromBottom">
    第<em>1</em>幅图片的说明文字
    </div>
    </div>
    <div data-thumb="Images/Img6.2/thumb/list_2.jpg"
    data-src="Images/Img6.2/list_2.jpg">
    <div class="camera_caption fadeFromBottom">
    第<em>2</em>幅图片的说明文字
    </div>
    </div>
    <div data-thumb="Images/Img6.2/thumb/list_3.jpg"
    data-src="Images/Img6.2/list_3.jpg">
    <div class="camera_caption fadeFromBottom">
    第<em>3</em>幅图片的说明文字
    </div>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-2所示。

图　6-2　使用Camera插件以幻灯片形式轮播图片

4.源码分析

在本实例中，可以将图片以轮播的方式在移动设备的浏览器中展现，如图6-2左侧图所示；同时，还可以以缩略图的方式预览下一张图，如图6-2右侧图所示。该功能的具体实现方法是：在内容区域中先添加一个＜div＞元素作为放置轮播图片的容器，Id号和类别名分别为“camera_wrap_1”、“camera_wrap”。在该容器中，同样使用＜div＞元素添加被轮播的图片，其代码格式如下：

```
    ＜div data-thumb="预览缩略图片来源"data-src="被轮播图片的来源"＞
    ＜div class="camera_caption"＞
    轮播图片的说明性文字
    ＜/div＞
    ＜/div＞
```

说明　虽然在轮播图片的容器中使用上述格式可以添加多张图片，但在移动设备上浏览时，建议添加3～5张图片是比较适宜的。

在页面中，播放图片的容器和被轮播的全部图片元素添加完成后，还必须在页面的初始化事件中调用插件的camera（）方法，才能实现在执行该页面时图片容器中的各个图片以幻灯片形式轮播的效果。

在调用插件的camera（）方法时，括号中也可以添加一个“options”对象，通过设置该对象，可以控制轮播图片的效果与特征。“options”对象的重要属性与描述如表6-2所示。

表 6-2　Camera 插件中“options”对象的重要属性和描述

属　性	描　述	默 认 值
alignment	轮播图片对齐方式	center
barPosition	文字说明条所在的位置	bottom
height	轮播图片显示的高度比例	50%
loader	加载时显示的进度图标，可设为 pie、bar 或不设置	pie
loaderStroke	加载时显示的进度图标边框的大小值	7
pagination	是否显示分页按钮，分页按钮指的是轮播图片底的小实心圆	true
slideOn	轮播图片播放时显示的顺序，可设置为 next、prev、random	random
time	轮播图片播放时间隔的时间，单位是毫秒	7000

在使用Camera插件轮播图片时，还可以使用公用的方法中止或暂停图片的播放，方法如下：

```
$（'#camera_wrap_1'）.cameraStop（）;
```

上述方法是停止图片的幻灯片式轮流播放。

```
$('#camera_wrap_1').cameraPlay();
```

上述方法是开始轮播放容器中的图片。

$('#camera_wrap_1'). cameraPause();

上述方法是暂停图片的幻灯片式轮流播放。

```
$('#camera_wrap_1').cameraResume();
```

上述方法是暂停后重新播放容器中的图片。

在使用Camera插件时，当触发函数指定事件时还可以回调函数，函数事件如下：

```
onStartLoading: function() {}
```

上述事件是当幻灯片的图片开始加载时触发。

```
onLoaded: function() {}
```

上述事件是当幻灯片的图片加载完成时触发。

```
onStartTransition: function() {}
```

上述事件是当幻灯片的图片开始过渡切换时触发。

```
onEndTransition: function（）{}
```

上述事件是当幻灯片的图片过渡切换完成时触发。

6.3　滚动选择时间插件Mobiscroll

在页面中输入日期或时间是一件很麻烦的事，因为考虑到日期或时间的特殊性，往往需要对输入的格式与内容进行有效性验证。而在移动终端的浏览器中，这样的验证还将更为复杂。为了解决这一问题，可以引用专门针对移动项目开发的滚动选择时间插件Mobiscroll。

Mobiscroll插件默认风格是以触摸屏的方式，通过滚轮选择日期或时间的值；当然，也可以自定义选择日期或时间的风格，如Android、Sense UI和iOS。该插件专门针对移动触摸设备设计的UI效果，广泛应用于众多的移动项目中，深受开发人员喜爱。

Mobiscroll插件的使用方法也很简单，只需要经过下面两个步骤，就可以实现单击绑定的文本框时，弹出选择日期或时间的窗口。

步骤1　在页面中添加一个Id号为“date1”的文本框元素，将它的“readonly”属性设置为“true”，表示该文本框是只读类型的。

步骤2　编写JavaScript代码绑定文本框和插件。

可选择的方法有3种：

1）调用插件的默认设置绑定指定的文本框，代码如下：

```
$（"#date1"）.scroller（）；
```

2）调用插件时间型的设置绑定指定的文本框，代码如下：

```
$（"#date1"）.scroller（{preset: 'time'}）；
```

3）调用插件日期与时间型的设置绑定指定的文本框，代码如下：

```
$（'#date1'）.scroller（{preset: 'datetime'}）；
```

在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

Js/Js6. 3/mobiscroll-1.6.js

Css/Css6. 3/mobiscroll-1.6.css

下载地址：http：//code.google.com/p/mobiscroll/

实例6-3　滚动选择时间插件Mobiscroll的应用

1.功能说明

新建一个HTML页面，在正文区添加一个表单元素，在表单中增加两个“readonly”属性值为“true”的文本框。第一个用于绑定Mobiscroll插件的默认设置，第二个用于绑定Mobiscroll插件的时间

型设置。单击这两个文本框，将分别弹出不同的选择日期和时间的窗口。

2.实现代码

新建一个HTML页面6-3.htm，加入代码如代码清单6-3所示。

代码清单6-3　滚动选择时间插件Mobiscroll的应用

```
<! DOCTYPE html>
<html>
<head>
<title>mobiscroll插件应用程序</title>
<meta name="viewport"content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/Css6.3/mobiscroll-1.6.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/Js6.3/mobiscroll-1.6.js"
type="text/javascript"></script>
<script type="text/javascript">
$（function（）{
$（"#date1"）.scroller（）;
$（"#date2"）.scroller（{preset: "time"}）;
}）
</script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>选择时间</h1></div>
<div data-role="content"data-theme="e">
<form id="testform">
日期：<input type="text"name="date1"
```

```
    id="date1"readonly="true"/>
    时间：<input type="text"name="date2"
    id="date2"readonly="true"/>
    </form>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-3所示。

图　6-3　使用Mobiscroll插件滚动选择日期和时间的效果

4.源码分析

在本实例中，Id号为“date1”的文本框绑定了插件的默认设置类型，如果将该文本框绑定Mobiscroll插件的日期时间类型，需要将对

应的JavaScript代码修改成如下代码：

```
$（"#date1"）.scroller（{preset: "datetime"}）;
```

此时，如果单击Id号为“date1”的文本框，将会弹出一个选择日期和时间组合在一起的窗口。用户在该窗口中既可以选择日期，也能选取不同的时间，单击“确定”按钮后，绑定的文本框中将显示所选择的日期与时间组合的值。

需要说明的是：下载的Mobiscroll插件以英文的形式显示“年”、“月”、“日”、“时”、“分”、“秒”、“确定”、“取消”等字符，如果需要修改成中文，打开插件文件mobiscroll-1.6.js，找到“defaults”对象，进行如下的代码修改即可：

```
……省略部分代码
monthText: '月',
dayText: '日',
yearText: '年',
hourText: '时',
minuteText: '分',
secText: '秒',
ampmText: ' ',
setText: '确定',
cancelText: '取消',
……省略部分代码
```

与普通的插件相同，Mobiscroll插件在调用scroller（）方法时，括号中是一个“options”对象，该对象可以设置多个属性或事件，其中几个重要的属性如表6-3所示。

表 6-3　Mobiscroll 插件中“options”对象的重要属性和描述

属性	描述	默认值
height	滚轮中各个单元格的高度，单位是像素	40
width	滚轮中各个单元格的最小宽度，单位是像素	80
rows	滚轮中供选择的可见行数	3
showValue	显示或隐藏头部标签的文字	true
showLabel	显示或隐藏滚轮上面标签的文字	true
theme	滚轮的主题，可以设置为“android”、“sense-ui”、“ios”	""
mode	选择日期与时间的模式，可选值为“scroller”、“clickpick”	“scroller”
preset	预设类型，可选值为“date”、“time”、“datetime”	“date”

Mobiscroll插件中还有以下几个常用的方式，其功能和使用方法如下所示：

```
$（"#date1"）.scroller（'destroy'）
```

上述方法将删除使用滚轮选择日期或时间的功能。

```
$（"#date1"）.scroller（'disable'）
```

上述方法将禁用滚轮选择日期或时间的功能。

```
$（"#date1"）.scroller（'enable'）
```

上述方法将启用滚轮选择日期或时间的功能。

```
$（"#date1"）.scroller（'getValue'）
```

上述方法将获取一个数组，该数据保存了滚轮选择日期或时间的值。

除此而外，Mobiscroll插件还可以在指定的事件中，触发自定义的函数，常用的事件如下：

```
onClose: function（valueText, inst）{}
```

上述事件在关闭滚轮选择窗口时触发，被调函数接收选定文本和滚动实例作为参数值，如果返回true，表示关闭成功，否则关闭失败。

```
onSelect: function（valueText, inst）{}
```

上述事件在开始通过滚轮的方式选择日期或时间时触发，被调函数接收选定时间值和滚动实例作为参数值。

```
onCancel: function（valueText, inst）{}
```

上述事件在取消滚轮选择窗口时触发，被调函数接收选定时间值和滚动实例作为参数值。

6.4　搜索插件AutoComplete

与jQuery中的AutoComplete插件相类似，jQuery Mobile中AutoComplete插件的功能是：在搜索文本框中输入任意关键字时，将自动完成与关键字相似或相近字符集的匹配，即实现搜索关键字的联想功能。

匹配后的字符集可以是本地的数组或JSON格式的数据，也支持通过远程URL访问的方式返回一个数组或JSON格式的数据。

匹配后的字符集以列表的形式显示在搜索文本框中的底部，当单击选项中某一个字符集时，将自动按设定的跳转链接地址，进入指定的页面中。

在jQuery Mobile的移动项目中，使用AutoComplete插件的方法非常简单，操作步骤如下：

步骤1　在页面中添加一个“type”属性值为“search”类型的搜索文本框元素，用于绑定导入的AutoComplete插件；另外，增加一个“listview”容器，用于显示匹配后返回的字符集数据。

步骤2　编写JavaScript代码，绑定页面的“pageshow”事件。在该事件中，通过搜索文本框调用AutoComplete插件的

“autocomplete”方法，在该方法中，将匹配后返回的字符集数据与“listview”容器相绑定。

在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

插件文件：Js/Js6.4/jqm.autoComplete-1.3.js

下载地址：http：//www.andymatthews.net/code/autocomplete/

实例6-4　搜索插件AutoComplete的应用

1.功能说明

新建一个HTML页面，在页面中分别添加一个Id号为“txtSearch”的搜索文本框和一个Id号为“ulSearchStr”的列表容器。用户在文本框中输入搜索关键字时，调用AutoComplete搜索插件，自动在文本框的底部以列表的形式显示匹配后返回的字符集数据。

2.实现代码

新建一个HTML页面6-4.htm，加入代码如代码清单6-4所示。

代码清单6-4　搜索插件AutoComplete的应用

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>autoComplete插件应用程序</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/Js6.4/jqm.autoComplete-1.3.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="mainPage">
    <div data-role="header"><h1>搜索联想</h1></div>
    <div data-role="content">
    <input type="search"id="txtSearch"
    placeholder="请输入搜索关键字">
    <ul id="ulSearchStr"data-role="listview"
    data-inset="true"></ul>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    <script type="text/javascript">
    $("#mainPage").bind("pageshow", function(e){
    var arrUserName=["张三", "王小五", "张才子",
    "李四", "张大三", "李大四", "王五", "刘明",
    "李小四", "刘促明", "李渊", "张小三", "王小明"];
    $("#txtSearch").autocomplete({
    target: $('#ulSearchStr'),
    source: arrUserName,
    link: 'clickUrl.html?s=',
    minLength: 0
    })
    })
    </script>
    </html>
```

---

3. 页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-4所示。

图　6-4　使用AutoComplete插件实现搜索联想功能

4.源码分析

在本实例中，Id号为“txtSearch”的搜索文本框通过调用AutoComplete插件的autocomplete（）方法，对文本框中输入的字符进行自动匹配，实现搜索自动联想功能。

jQuery Mobile与jQuery中的AutoComplete插件功能类似，只是前者调用autocomplete（）方法时，括号中“options”对象所包含的属性名称不同，该插件调用的标准方式为：

```
$（"#搜索文本框Id"）.autocomplete（{
target: $（）,
source: null,
callback: null,
link: null,
minLength: 0,
transition: 'fade'
}）
```

在上述标准调用方式中，autocomplete（）方法括号中的“options”对象所包含的属性和详细描述如表6-4所示。

表 6-4　AutoComplete 插件中“options”对象的属性和描述

属性	描述	默认值
target	设置显示返回数据容器的 Id 号，默认为“listview”类型列表容器	$()
source	匹配后返回的字符集数据，该数据可以是一个本地数组或远程 URL	null
callback	设置单击列表容器中某一项返回字符集时，回调的函数	null
link	设置单击列表容器中某一项返回字符集时，对应的链接地址	null
minLength	设置搜索文本框中允许最小输入的字符长度	0
transition	设置单击链接地址时页面跳转的方式	“fade”

在设置“source”属性值时，允许使用本地的数组或JSON格式的数据，在本实例中使用的是本地数组，也可以修改为JSON格式的数据，实现代码如下：

```
……省略部分代码
var arrUserName=$.parseJSON（'[
{"value": "张三", "text": "张三"},
{"value": "王小五", "text": "王小五"}]
'）;
……省略部分代码
```

此外，“source”属性值也可以是一个远程访问的URL地址，当然该地址必须返回一个数组或JSON格式的数据，并且不允许跨域访问。

另外，当用户单击列表容器中某一项匹配的搜索数据时，将触发“callback”属性值所设置的回调函数，在该函数中，可以通过返回的“e”对象，获取搜索字符的内容，如下列代码：

```
……省略部分代码
callback: function（e）{
var$obj=$（e.currentTarget）;
$（'#txtSearch'）.val（$obj.text（））;
$（"#txtSearch"）.autocomplete（'clear'）;
},
……省略部分代码
```

在上述代码中，先通过“$obj”变量保存当前搜索字符自动匹配后的数据对象，然后，通过对象的“text（）”属性，将用户单击时对应的字符内容赋予搜索文本框，最后调用插件的“clear”方法，清空文本框中原有的字符内容。

6.5　日期对话框插件DateBox

DateBox插件和Mobiscroll插件都是专门用于jQuery Mobile移动项目的插件，前者是通过滚动齿轮的方式选择某一个年月日和时分秒日期数据，而后者简单直观地展示一个日期与时间的对话框，用户直接单击其中的某个按钮，便完成了日期选择的操作。相比之下，后者更加易操作，可扩展性也更强。

与其他用于jQuery Mobile移动项目中的日期型插件相比较，DateBox日期对话框插件在使用时有以下几个显著的特点：

允许使用多种模式输入数据，如“Android”、“Calendar”、“Slide”、“Flip Wheel”、“time”模式，模式不同其弹出的日期选择对话框的风格也不同。

可以设置4种不同的日期显示模式。

使用时数据完全本地化。

允许对输入日期数据的限制，如设置最大或最小年份、设置某一日作为黑名单中的日期、设置特定的某一天等操作。

自动解析手动输入或预先输入的日期。

使用data-role=“DateBox”绑定页面中文本框元素，通过“data-options”属性设置数据的各选项配置。

在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

Css/Css6. 5/jquery.mobile.datebox.css

Js/Js6. 5/jquery.mobile.datebox.js

下载地址：http：//dev.jtsage.com/jQM-DateBox/

实例6-5　日期对话框插件DateBox的应用

1.功能说明

新建一个HTML页面，添加一个“readonly”属性值为“true”的文本框元素，分别将该元素的“data-role”属性值设置为“datebox”，“data-options”属性值设置为“'{"mode"："calbox"}'”。浏览该页面时，将在文本框的最右侧出现一个圆形小按钮，单击该按钮时，将弹出日期选择对话框。

2.实现代码

新建一个HTML页面6-5.htm，加入代码如代码清单6-5所示。

代码清单6-5　日期对话框插件DateBox的应用

```
<! DOCTYPE html>
<html>
<head>
<title>datebox插件应用程序</title>
<meta name="viewport"content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/Css6.5/jquery.mobile.datebox.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/Js6.5/jquery.mobile.datebox.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"><h1>日期插件</h1></div>
<div data-role="content">
选择日期:
<input name="lang1"id="lang1"type="text"readonly="true"
data-role="datebox"data-options='{"mode": "calbox"}'/>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-5所示。

图　6-5　使用DateBox插件选择日期的效果

4.源码分析

在本实例中，将文本框元素的“data-role”属性值设置为“datebox”，使该文本框与DateBox插件绑定。绑定后的文本框将会在最右则有一个圆形小按钮，单击该按钮后，将会弹出一个日期选择对话框，完整效果如图6-5所示。

此外，还可以通过添加文本框的“data-options”属性设置插件对应的选项配置，如设置日期数据输入的模式为“calbox”，“data-options”属性对应值为“'{"mode":"calbox"}'”。

如果设置某一日期为黑名单，“data-options”属性对应值为：

```
    '{"fixDateArrays": true, "blackDates": ["2012-10-
2", "2012-05-04"], "mode":
    "calbox"}'
```

该行代码将“2012-10-2”和“2012-05-04”这两天列为黑名单，在日期对话框选择日期时，设置为黑名单的日期是不可以被选择的，仅用于显示。

6.6　简单对话框插件SimpleDialog

SimpleDialog插件可以取代JavaScript中dialog对话框的功能。SimpleDialog插件有3种固定的对话模式，功能说明如下：

“bool”模式：该模式为默认模式，在该模式下，单击对话框中的“确定”按钮后，将返给被调用页面一个“bool”类型的值。

“string”模式：在该模式下，单击对话框中的“确定”按钮后，将返给被调用页面一个“string”类型的值。

“blank”模式：在该模式下，会弹出一个用户自定义内容的对话框。

SimpleDialog插件的使用方法十分简单，操作步骤如下：

步骤1　在页面中添加一个＜a＞元素，用于单击该元素时，弹出SimpleDialog插件所形成的对话框。

步骤2　编写JavaScript代码，调用SimpleDialog插件的simpledialog（）方法，将＜a＞元素与插件相绑定。

接下来，通过一个完整的实例来介绍SimpleDialog插件在jQuery Mobile移动项目中使用的方法与过程。在使用该插件之前，请先将如

下资源文件放置到所在页面的＜head＞元素中。

Css/Css6. 6/jquery.mobile.simpledialog.css

Js/Js6. 6/jquery.mobile.simpledialog.js

下载地址：http：//dev.jtsage.com/jQM-SimpleDialog/

实例6-6　简单对话框插件SimpleDialog的应用

1.功能说明

新建一个HTML页面，在页面中添加一个“listview”列表容器。用户单击容器选项最右侧的“删除”按钮时，将调用SimpleDialog插件，弹出一个确定删除的对话框；单击对话框中“确定”按钮时，删除所选择的选项；单击“取消”按钮时，关闭弹出的对话框。

2.实现代码

新建一个HTML页面6-6.htm，加入代码如代码清单6-6所示。

代码清单6-6　简单对话框插件SimpleDialog的应用

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞simpledialog插件应用程序＜/title＞
＜meta name="viewport"content="width=device-width,
```

```
    initial-scale=1.0，maximum-scale=1.0，user-scalable=0；"/
>
    ＜link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/＞
    ＜link href="Css/Css6.6/jquery.mobile.simpledialog.css"
    rel="Stylesheet"type="text/css"/＞
    ＜script src="Js/jquery-1.6.4.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/Js6.6/jquery.mobile.simpledialog.js"
    type="text/javascript"＞＜/script＞
    ＜script type="text/javascript"＞
    $（function（）{
    $（"li a[data-transition='slideup']"）
    .each（function（index）{
    $（this）.bind（"click"，function（）{
    $（this）.simpledialog（{
    'mode'：'bool'，
    'prompt'：'您真的要删除所选择的记录吧？'，
    'useModal'：true，
    'buttons'：{
    '确定'：{
    click：function（）{
    var$delId="li"+index；
    $（"#"+$delId）.remove（）；
    }
    }，
    '取消'：{
    click：function（）{
    //编写单击取消按钮事件
    }，
    icon："delete"，
    theme："c"
    }
    }
    }）
    }）
    }）
    }）；
    ＜/script＞
    ＜/head＞
    ＜body＞
    ＜div data-role="page"＞
    ＜div data-role="header"＞＜h1＞对话框＜/h1＞＜/div＞
```

```
    <div data-role="content">
    <ul data-role='listview'data-split-icon="delete"data-
split-theme="c">
    <li id="li0"><a href="#">图书</a>
    <a href="#"data-transition="slideup">删除图书大类</a>
    </li>
    <li id="li1"><a href="#">影视</a>
    <a href="#"data-transition="slideup">删除影视大类</a>
    </li>
    <li id="li2"><a href="#">音乐</a>
    <a href="#"data-transition="slideup">删除音乐大类</a>
    </li>
    </ul>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-6所示。

图　6-6　应用SimpleDialog插件删除和确定的效果

4.源码分析

在本实例的JavaScript代码中，先使用each（）方法遍历列表中各选项，获取最右侧的圆形删除按钮元素；然后，将各个获取的删除按钮元素绑定SimpleDialog插件的simpledialog（）方法；最后，在该方法的“options”对象中，完成相关属性值的设置。

在删除按钮绑定SimpleDialog插件以后，当用户单击对话框中的“确定”按钮时，将触发按钮的“click”事件。在该事件中，先根据“index”传回的值，获取用户选中的是哪一行；然后，使用remove（）方法删除指定Id号的列表选项，实现在页面中移除所选择记录的效果。

在调用SimpleDialog插件中simpledialog（）方法时，可以在该方法的括号中通过对应的“options”对象，设置弹出对话框插件的相关属性。“options”对象所包含的属性和详细描述如表6-5所示。

表 6-5　SimpleDialog 插件中“options”对象的属性和描述

属性	描述	默认值
mode	弹出对话框的模式	bool
prompt	给用户显示的内容	Are you sure?
cleanOnClose	当用户单击关闭按钮时，是否彻底清除已打开的对话框	false
clickEvent	绑定按钮的事件名称	click
subTitle	是否在对话框内容的第二行显示字幕信息	false
fullHTML	当模式为“blank”时，对话框中自定义的内容	null
inputPassword	是否使用密码输入字符格式	false

除此而外，SimpleDialog插件还允许在指定的事件中回调自定义的函数，常用的事件如下：

```
onCreated: function（）{}
```

上述事件在创建一个对话框时触发。

```
onOpened: function（）{}
```

上述事件在打开一个对话框时触发。

```
onClosed: function（）{}
```

上述事件在关闭一个对话框时触发。

```
onShown: function（）{}
```

上述事件在打开对话框动画效果结束时触发。

6.7　快捷标签插件ActionSheet

ActionSheet插件无须编写任何JavaScript代码，完全通过HTML 5新增的属性控制。通过引入该插件，可以在页面中以优雅的动画效果弹出一个任意的标签，该标签中的内容可以是任何HTML代码元素。

鉴于该插件的快捷性，ActionSheet插件广泛应用于页面中的内容显示、信息通知和广告发布等，当然，也可以用于设置用户退出时弹出的询问对话框或快捷的弹出式用户登录对话框。

ActionSheet插件完全依靠元素的相关属性进行设置，使用相对简单，操作步骤通常分以下2步。

步骤1　在页面中添加不同元素，创建一个用于弹出的标签对话框，并设置Id号属性。

步骤2　在页面中，添加一个用于调用标签对话框的元素，将该元素的“data-role”属性设置为“actionsheet”，表示该元素用于弹出标签对话框；再通过将元素的“data-sheet”属性值设置为标签对话框的Id号，最终实现元素与标签对话框的绑定。

在使用该插件之前，请先将如下资源文件放置到所在页面的＜head＞元素中。

Css/Css6. 7/jquery.mobile.actionsheet.css

Js/Js6. 7/jquery.mobile.actionsheet.js

下载地址：
https：//github.com/hiroprotagonist/jquery.mobile.actionsheet/

实例6-7　快捷标签插件ActionSheet的应用

1.功能说明

新建一个HTML页面，在页面中分别创建两个快捷标签对话框，第一个在单击头部最右侧“退出”按钮时弹出显示；第二个在单击内容区域中“登录”按钮时弹出显示。

2.实现代码

新建一个HTML页面6-7.htm，加入代码如代码清单6-7所示。

代码清单6-7　快捷标签插件ActionSheet的应用

```
<! DOCTYPE html>
<html>
<head>
<title>actionsheet插件应用程序</title>
<meta name="viewport"content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/Css6.7/jquery.mobile.actionsheet.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/Js6.7/jquery.mobile.actionsheet.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header">
<h1>快捷标签</h1>
<a data-icon="gear"class="ui-btn-right"
data-role="actionsheet">退出
</a>
<div>
<p class="pTip">您真的要退出本系统吗？</p>
<div class="ui-grid-a">
<div class="ui-block-a">
<a data-role="button"
class="ui-btn-active">确定</a>
</div>
<div class="ui-block-b">
<a data-role="button"
data-rel="close">取消</a>
</div>
</div>
</div>
<div data-role="content">
<a data-icon="star"data-sheet="login"
data-role="actionsheet">登录</a>
<form id="login"action="#">
<span class="spnLogin">用户登录</span>
<input name="user"type="text"
placeholder="请输入名称"/>
<input name="pass"type="password"
placeholder="请输入密码"/>
<div class="ui-grid-a">
<div class="ui-block-a">
<a data-role="button"type="submit"
class="ui-btn-active">确定</a>
```

```
    </div>
    <div class="ui-block-b">
    <a data-role="button"type="reset">取消</a>
    </div>
    </div>
    </form>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图6-7所示。

图　6-7　使用ActionSheet插件弹出标签框

4.源码分析

在本实例中，创建了2个标签框（actionsheet）。第一个在“header”容器中，通过＜div＞标记进行包裹，并且＜div＞标记的位置必须放在调用该标签框元素的下一行，即“header”容器中。第二个在“content”容器中，也是被＜div＞元素包裹；同时，在调用标签框＜a＞元素中，还要将“data-role”属性值设置为“actionsheet”，表示该元素将调用使用＜div＞元素包裹的标签框（actionsheet）。

与“header”容器不同，在“content”容器中，调用标签框元素与标签框本身的位置必须是前后关系，还要为标签框添加一个Id号属性，调用标签框元素不仅将“data-role”属性值设置为“actionsheet”，而且还要添加“data-sheet”属性，并将该属性的值设置为标签框的Id号，才完成与标签框的绑定。

说明　使用ActionSheet插件创建标签框（actionsheet），并与指定的链接元素相绑定，其实现的过程不需要编写任何JavaScript代码，只需要把握两个元素间的放置顺序关系，并通过添加“data-role”、“data-sheet”属性，将两者进行绑定。

6.8　本章小结

本章列举了7款专门针对jQuery Mobile开发移动项目的插件，通过一个个精选的实例，详细介绍了这些插件在jQuery Mobile项目中的使用方法、开发技巧和注意事项。通过本章的学习，读者能够了解在jQuery Mobile开发移动项目中如何借助插件的优势，加快项目开发进度，提高开发效率。

第7章　jQuery Mobile API详解

本章内容

基本配置项

事件

访问地址的相关方法

本章小结

jQuery Mobile完全使用HTML 5和CSS 3特征开发移动项目的页面功能，为开发者提供了大量实用、可供扩展的API接口。通过这些接口提供的方法，开发者可以实现继承拓展jQuery Mobile的初始化事件、创建自定义的命名空间、设置当前激活页的样式、配置默认页和对话框的效果、自定义页面加载与出错的提示信息等功能；另外，借助jQuery Mobile API拓展事件，可以在页面触摸、滚动、加载、显示与隐藏的事件中，编写特定代码，实现事件触发时需要完成的功能。

7.1　基本配置项

在jQuery Mobile中，框架的基本配置项是可以被修改的。由于配置项针对的是全局功能的使用，jQuery Mobile在页面加载到增强特征时就需要使用这些配置项，而这个加载过程早于document.ready事件的触发，因此，在该事件中进行修改是无效的，而要选择更早的“mobileinit”事件，在该事件中，可以编写新的配置项来覆盖原有的基本配置项设置。

在document.mobileinit事件中自定义配置项，可以使用jQuery中的$.extend方法扩展，也可以借助$.mobile对象进行设置，下面通过两个实例分别介绍各自实现的过程。

7.1.1　自定义页面加载和出错提示信息

当用户在移动设备端浏览jQuery Mobile开发的移动项目页面时，如果是首次加载或速度较慢，会在页面的居中位置显示滚动的加载动画和“Loading”的文字信息。另外，如果访问的某个链接页面不存在，也会出现“Error Loading Page”的提示信息，而这些默认配置项都可以在document.mobileinit事件中进行自定义设置。

实例7-1　自定义页面加载和出错提示信息

1.功能说明

新建一个HTML页面，在页面中增加一个＜a＞元素，将该元素的“href”属性值设置为一个不存在的页面文件“error.htm”。用户单击该元素时，将显示自定义的出错提示信息。

2.实现代码

新建一个HTML页面7-1.htm，加入代码如代码清单7-1所示。

代码清单7-1　自定义页面加载和出错提示信息

```
    ＜! DOCTYPE html＞
    ＜html＞
    ＜head＞
    ＜title＞jQuery Mobile默认配置＜/title＞
    ＜meta name="viewport"content="width=device-width，
    initial-scale=1.0，maximum-scale=1.0，user-scalable=0；"/
＞
    ＜link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/＞
    ＜script src="Js/jquery-1.6.4.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/7-1.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"＞＜/script＞
    ＜/head＞
    ＜body＞
    ＜div data-role="page"＞
    ＜div data-role="header"＞
    ＜h1＞API＜/h1＞
    ＜/div＞
    ＜div data-role="content"＞
    ＜h3＞修改默认配置值＜/h3＞
    ＜p＞＜a href="error.htm"＞点击我＜/a＞＜/p＞
```

```
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-1.js”文件。在该文件中，使用$.mobile对象在“mobileinit”完成了对默认配置信息的修改，代码如下所示：

```
    $（document）.bind（"mobileinit", function（）{
    $.extend（$.mobile，{
    loadingMessage：'努力加载中……',
    pageLoadErrorMessage：'找不到对应页面！'
    }）；
    }）
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-1所示。

图　7-1　修改默认配置值后的出错提示

4.源码分析

本实例中，借助$.mobile对象，在“mobileinit”事件中通过下列两行代码，分别修改了页面加载时和加载出错时的提示信息，代码如下：

```
$.extend（$.mobile，{
loadingMessage：'努力加载中……',
pageLoadErrorMessage：'找不到对应页面！'
}）
```

上述代码调用了jQuery中的$.extend（）方法进行扩展，也可以使用$.mobile对象直接对各配置值进行设置，因此，上述代码等同于：

```
$.mobile.loadingMessage='努力加载中……';
$.mobile.pageLoadErrorMessage='找不到对应页面！';
```

通过在“mobileinit”事件中加入上述代码中的任意一种，都可以实现修改默认配置项“loadingMessage”和“pageLoadErrorMessage”的显示内容。

说明　由于“mobileinit”事件是在加载时立刻触发，因此，无论是在页面上直接编写JavaScript代码，还是引用js文件，都必须将它放在引用“jquery.mobile-1.0.1.js”之前，否则代码无效。

7.1.2　使用函数修改gradeA配置值

在jQuery Mobile的默认配置中，“gradeA”配置项表示检测浏览器是否属于支持类型中的“A”级别，配置值为布尔型，默认为“$.support.mediaquery”。除此之外，也可以通过代码检测当前的浏览器是否是支持类型中的“A”级别，接下来通过一个实例进行详细的说明。

实例7-2　使用函数修改gradeA配置值

1.功能说明

新建一个HTML页面，在页面中添加一个Id号为“pTip”的＜p＞元素。当执行页面的浏览器属于“A”类支持级别时，在＜p＞元素中显示“浏览器是否为"A"类级别：true”字样，否则显示“浏览器是否为"A"类级别：false”字样。

2.实现代码

新建一个HTML页面7-2.htm，加入代码如代码清单7-2所示。

代码清单7-2　使用函数修改gradeA配置值

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>jQuery Mobile默认配置</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/7-2.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script type="text/javascript">
    $(function(){
    var strTmp='浏览器是否为"A"类级别：';
    $("#pTip").html(strTmp+$.mobile.gradeA());
    })
    </script>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header">
    <h1>头部栏</h1>
    </div>
    <div data-role="content">
    <h3>修改默认配置gradeA的值</h3>
    <p id="pTip"></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-2.js”文件。在该文件中，使用函数的方式创建一个 元素，然后检测各类浏览器对该元素中CSS 3样式的支持状态，并将函数返回的值作为“gradeA”配置项的新值。7-2.js文件代码如下所示：

```
    $（document）.bind（"mobileinit", function（）{
    $.extend（$.mobile，{
    gradeA: function（）{
    //创建一个临时的div元素
    var divTmp=document.createElement（"div"）;
    //设置元素的内容
    divTmp.innerHTML='＜div style="-webkit-transform:
rotate（360deg）;
    -moz-transform: rotate（360deg）; "＞＜/div＞';
    //定义一个初始值
    var btnSupport=false;
    btnSupport=
    （divTmp.frstChild.style.webkitTransform! =undefned）
    ||
    （divTmp.frstChild.style.MozTransform! =undefned）;
    return btnSupport;
    }
    }）;
    }）;
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-2所示。

图　7-2　显示“gradeA”配置项不同值时的效果

4.源码分析

在本实例的JavaScript代码中，当触发“mobileinit”事件时，通过“$.mobile”对象重置“gradeA”配置值，该配置值是一个函数的返回值。在这个函数中，先创建一个＜div＞元素，并在该元素中设置一个翻转360度的CSS 3样式效果；然后，根据浏览器对该样式效果的支持情况，返回值“false”或“true”；最后，将该值作为整个函数的返回值，对“gradeA”的配置值进行修改。如果返回值为“false”，表示浏览器对该样式的支持并未达到“A”类级别，页面效果如图7-2左侧所示；如果返回值为“true”，表示浏览器对该样式

的支持已达到“A”类级别，页面效果如图7-2右侧所示。详细实现的过程见代码中加粗部分。

除实例7-1和实例7-2中所介绍的配置选项外，其他jQuery Mobile默认配置项名称与使用方法如表7-1所示。

表 7-1　jQuery Mobile 中默认配置项使用说明

选项名	描述	值类型	默认值
ns	该选项可以自定义自己的命名空间，格式为“data-自定义名-role”，并按“.ui-mobile [data-自定义名-role=page]”的格式在主题样式中增加对应的样式类别	string	""
autoInitializePage	如果该选项的值为 true，那么当页面加载完成后，jQuery Mobile 框将自动调用 $.mobile.initializePage() 方法，初始化页面的数据；否则，不初始化数据	boolean	true
subPageUrlKey	该选项可以自定义子页引用时 URL 格式，对比如下： 修改前为：index.html&ui-page=value 修改后为：index.html&my-page=value	string	ui-page

（续）

选项名	描述	值类型	默认值
activePageClass	该选项可以自定义当前页面或切换页的类别名称	string	ui-page-active
activeBtnClass	该选项可以自定义当前被选中的按钮类别名称	string	ui-btn-active
ajaxEnabled	如果该选项的值为 true，那么，jQuery Mobile 会自动通过 AJAX 的方式来处理单击链接或提交表单时的数据请求；否则，使用 HTTP 方式请求数据	boolean	true
defaultPage-Transition	该选项可以设置页面在切换时的转换效果，如果该值为“none”，则没有任何页面切换时的转换效果	string	slide
minScrollBack	该选项可以设置页面的最小滚动距离	string	150

7.2　事件

在移动终端设备中，有一类事件无法触发（如鼠标事件或窗口事件），但它们又客观存在。因此，在jQuery Mobile中，借助框架的API将这类型的事件扩展为专门用于移动终端设备的事件，如触摸、设备翻转、页面切换等，开发人员可以使用live（）或bind（）进行绑定。

7.2.1　触摸事件

在jQuery Mobile中，触摸事件包括5种类型，详细说明如下：

1）tap（轻击）事件，用户完成一次快速完整的轻击页面屏幕时触发。

2）taphold（轻击不放）事件，用户完成一次轻击页面屏幕且保持不放（大约1秒）时触发。

3）swipe（划动）事件，用户在1秒内水平拖曳屏幕距离大于30px或垂直拖曳屏幕距离小于20px时触发。触发该事件时需要注意下列属性：

scrollSupressionThreshold：该属性默认值为10px，水平拖曳大于该值则停止。

durationThreshold：该属性默认值为1000ms，划动时超过该值则停止。

horizontalDistanceThreshold：该属性默认值为30px，水平拖曳超出该值时才能滑动。

verticalDistanceThreshold：该属性默认值为75px，垂直拖曳小于该值时才能滑动。

4）swipeleft（向左边划动）事件，用户向左边划动屏幕时触发。

5）swiperight（向右边划动）事件，用户向右边划动屏幕时触发。

其swipeleft与swiperight事件常用于移动项目中的页面元素向左或向右的滑动查看，如相册中的图片浏览。接下来我们通过一个完整的实例，介绍使用swipeleft与swiperight事件以滑动的方式查看图片的过程。

实例7-3　使用触摸事件滑动浏览图片

1.功能说明

新建一个HTML页面，在页面中通过＜ul＞列表中的＜li＞元素添加4幅图片。当页面加载完成后，用户可以对并行显示的全部图片进行向左或向右的滑动浏览。

2.实现代码

新建一个HTML页面7-3.htm，加入代码如代码清单7-3所示。

代码清单7-3　使用触摸事件滑动浏览图片

```
    ＜! DOCTYPE html＞
    ＜html＞
    ＜head＞
    ＜title＞jQuery Mobile触摸事件＜/title＞
    ＜meta name="viewport"content="width=device-width,
    initial-scale=1.0，maximum-scale=1.0，user-scalable=0; "/
＞
    ＜link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/＞
    ＜link href="Css/7-3.css"
    rel="Stylesheet"type="text/css"/＞
    ＜script src="Js/jquery-1.6.4.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"＞＜/script＞
    ＜/head＞
    ＜body＞
    ＜div data-role="page"＞
    ＜div data-role="header"＞
    ＜h1＞头部栏＜/h1＞
    ＜/div＞
    ＜div data-role="content"＞
    ＜div class="ifrswipt"＞
    ＜div class="inner"＞
```

```
    <ul id="ifrswipt">
    <li><img src="Images/pic08.jpg"alt=""
    class="imgswipt"/></li>
    <li><img src="Images/pic09.jpg"alt=""
    class="imgswipt"/></li>
    <li><img src="Images/pic10.jpg"alt=""
    class="imgswipt"/></li>
    <li><img src="Images/pic11.jpg"alt=""
    class="imgswipt"/></li>
    </ul>
    </div>
    </div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    <script src="Js/7-3.js"type="text/javascript"></script
>
    </html>
```

在上述代码中，<head>元素引用了一个“7-3.css”文件，在该文件中编写控制装载全部图片的框架与图片样式，代码如下所示：

```
    /*滑动截图*/
    .ifrswipt
    {
    width: 223px; height: 168px;
    margin: 0 auto; position: relative;
    padding: 3px 20px 3px 20px
    }
    .ifrswipt.inner
    {
    width: 223px; height: 168px;
    overflow: visible; position: relative
    }
    .ifrswipt ul
    {
    width: 920px; list-style: none;
    overflow: hidden; position: absolute;
    top: 0px; left: 0; margin: 0; padding: 0
```

```
}
.ifrswipt li
{
width: 120px; height: 168px;
display: inline; line-height: 168px;
float: left; position: relative;
margin-right: 15px
}
.ifrswipt li.imgswipt
{
width: 120px; height: 160px;
cursor: pointer; padding: 3px;
border: solid 1px#eee
}
```

在代码清单7-3中，页面底部引用了一个“7-3.js”文件。在该文件中编写JavaScript代码调用“swipeleft”和“swiperight”事件，实现滑动浏览图片的功能。7-3.js文件如下所示：

```
//全局命名空间
var swiptimg={
$index: 0,
$width: 120,
$swipt: 0,
$legth: 3
}
var$imgul=$（"#ifrswipt"）;
$（".imgswipt"）.each（function（）{
$（this）.swipeleft（function（）{
if（swiptimg.$index＜swiptimg.$legth）{
swiptimg.$index++;
swiptimg.$swipt=-swiptimg.$index*swiptimg.$width;
$imgul.animate（{left: swiptimg.$swipt}, "slow"）;
}
}）.swiperight（function（）{
if（swiptimg.$index＞0）{
swiptimg.$index--;
swiptimg.$swipt=-swiptimg.$index*swiptimg.$width;
$imgul.animate（{left: swiptimg.$swipt}, "slow"）;
}
```

```
    })
})
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-3所示。

图　7-3　调用触摸事件滑动图片时的效果

4.源码分析

在本实例中，首先在类别名为“ifrswipt”的＜div＞容器中添加一个＜ul＞列表，并将全部滑动浏览的图片添加至列表的＜li＞元素中。

然后，在本实例对应的js文件中，先定义了一个全局性对象“swiptimg”，在该对象中设置需要使用的变量，并将获取的图片加载框架元素，保存在“$imgul”变量中。

最后，无论是将图片绑定“swipeleft”事件还是“swiperight”事件，都要调用each（）方法遍历全部图片。在遍历时，通过“$（this）”对象获取当前的图片元素，并将它与“swipeleft”和“swiperight”事件相绑定。

在“swipeleft”事件中，先判断当前图片的索引变量“swiptimg.$index”值是否小于图片总量值“swiptimg.$legth”。如果成立，索引变量自动增加1，然后将需要滑动的长度值保存到变量“swiptimg.$swipt”中。最后，通过前面保存元素的“$imgul”变量调用jQuery的animate（）方法，以动画的方式向左边移动指定的长度。

在“swiperight”事件中，由于是向右滑动，因此，先判断当前图片的索引变量“swiptimg.$index”的值是否大于0。如果成立，说明整个图片框架已向左边滑动过，索引变量自动减少1。然后，获取滑动时的长度值并保存到变量“swiptimg.$swipt”中。最后，通过前面保存元素的“$imgul”变量调用jQuery的animate（）方法，以动画的方式向右边移动指定的长度。详细实现过程见代码加粗部分。

说明　每次滑动的长度值都与当前图片的索引变量相连，因此，每次的滑动长度都会不一样；另外，图片加载完成后，根据滑动的条件，必须按照先从右侧滑动至左侧，然后再从左侧滑动至右侧的顺序进行，其中每次滑动时的长度和图片总数变量读者可以自行修改。

7.2.2　设置翻转事件

在jQuery Mobile事件中，当用户使用移动终端设备浏览页面时，如果手持设备的方向发生变化，即横向或纵向手持时，将触发“orientationchange”事件。在该事件中，通过获取回调函数中返回对象的“orientation”属性，可以判断用户手持设备的当前方向。该属性有两个值，分别为“portrait”和“landscape”，前者表示纵向垂直，后者表示横向水平。

实例7-4　使用翻转事件检测移动设备的手持方向

1.功能说明

新建一个HTML页面，并在页面中添加一个＜p＞元素。当用户变换移动设备的手持方向时，＜p＞元素中显示的文字内容和背景样式将随之发生变化。

2.实现代码

新建一个HTML页面7-4.htm，加入代码如代码清单7-4所示。

代码清单7-4　使用翻转事件检测移动设备的手持方向

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>jQuery Mobile设备翻转事件</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/7-4.js"
    type="text/javascript"></script>
    <link href="Css/7-4.css"
    rel="Stylesheet"type="text/css"/>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header">
    <h1>头部栏</h1>
    </div>
    <div data-role="content">
    <p/>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-4.js”文件。在该文件中编写触发“orientationchange”事件时控制<p>元素内容与样式的功能，代码如下所示：

```
    $（function（）{
    var$p=$（"p"）;
    $（'body'）.bind（'orientationchange', function（event）{
    var$oVal=event.orientation;
    if（$oVal=='portrait'）{
    $p.html（"垂直方向"）;
```

```
$p.attr（"class", "p-portrait"）;
}else{
$p.html（"水平方向"）;
$p.attr（"class", "p-landscape"）;
}
}）
}）
```

在代码清单7-4中，＜head＞元素引用了一个“7-4.css”文件。在该文件中编写用于控制垂直与水平方向样式类别，代码如下所示：

```
/*纵向垂直时的样式*/
.p-portrait
{
width: 75px; height: 150px;
line-height: 150px; text-align: center;
background-color: #eee; border: solid 2px#666
}
/*横向水平时的样式*/
.p-landscape
{
width: 150px; height: 75px;
line-height: 75px; text-align: center;
background-color: #ccc; border: solid 2px#666
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-4所示。

图　7-4　调用翻转事件控制页面元素时的效果

4.源码分析

在本实例中，先在页面中添加一个＜p＞元素，并设置两个样式类别“p-portrait”和“p-landscape”，分别控制移动设备的手持方向为垂直和水平时＜p＞元素的样式。

在实例对应的js文件中，页面加载时，将＜body＞元素绑定“orientationchange”事件。在该事件的回调函数中，通过传回的“orientation”属性值检测用户移动设备的手持方向。如果为“portrait”，则＜p＞元素的文字内容为“垂直方向”字样，设置样式类别名为“p-portrait”；反之，＜p＞元素的文字内容为“水平方向”字样，设置样式类别名为“p-landscape”，从而实现根据不同的

移动设备的手持方向，动态地改变＜p＞元素显示的文字内容与样式功能。

说明　在页面中，“orientationchange”事件的触发前提是必须将$.mobile.orientation-ChangeEnabled配置项的值设为“true”；如果改变该选项的值，将不会触发该事件，只会触发“resize”事件。

7.2.3　屏幕滚动事件

在jQuery Mobile中，屏幕滚动事件包含两个类型，一种为开始滚动事件（scrollstart），另一种为结束滚动事件（scrollstop）。这两种类型的事件主要区别在于触发时间不同，前者是用户开始滚动屏幕中页面时触发，而后者是用户停止滚动屏幕中页面时触发。接下来通过一个完整的实例介绍如何在移动项目的页面中绑定这两个事件。

实例7-5　使用屏幕滚动事件控制页面显示的文字与样式

1.功能说明

新建一个HTML页面，并在页面中添加一个＜p＞元素。用户开始或停止滚动屏幕时，触发已绑定的对应事件。在事件中，动态改变＜p＞元素中的文字内容和背景颜色。

2.实现代码

新建一个HTML页面7-5.htm，加入代码如代码清单7-5所示。

代码清单7-5　使用屏幕滚动事件控制页面显示的文字与样式

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞jQuery Mobile屏幕滚动事件＜/title＞
```

```
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/7-5.js"
    type="text/javascript"></script>
    <link href="Css/7-5.css"
    rel="Stylesheet"type="text/css"/>
    </head>
    <body>
    <div data-role="page">
    <div data-role="header">
    <h1>头部栏</h1>
    </div>
    <div data-role="content">
    <p/>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-5.js”文件。在该文件中编写屏幕开始或停止滚动时绑定相应事件、并在事件中动态改变<p>元素中文字内容和背景样式，7-5.js文件如下所示：

```
    $('div[data-role="page"]').live('pageinit',
function(event, ui){
    var eventsElement=$('p');
    $(window).bind('scrollstart', function(){
    eventsElement.html("开始滚动");
    eventsElement.css('background', 'green');
    })
    $(window).bind('scrollstop', function(){
```

```
eventsElement.html（"滚动停止"）；
eventsElement.css（'background', 'red'）；
}）
}）
```

在代码清单7-5中，＜head＞元素引用了一个“7-5.css”文件。在该文件中编写用于显示提示信息的＜p＞元素样式，代码如下所示：

```
/*设置提示元素的样式*/
p
{
height: 23px; line-height: 23px;
padding: 5px; text-align: center;
border: solid 2px#666
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-5所示。

图　7-5　调用屏幕滚动事件控制页面元素时的效果

4.源码分析

在本实例中，页面在触发“pageinit”事件时，将“window”屏幕对象分别与“scrollstart”和“scrollstop”事件相绑定。“window”屏幕开始滚动时触发“scrollstart”事件，在该事件中将＜p＞元素显示的文字设为“开始滚动”字样，背景色变成“green”。

当“window”屏幕停止滚动时，触发“scrollstop”事件。在该事件中，将＜p＞元素显示的文字设为“滚动停止”字样，背景色也变成“red”。

说明　iOS系统中的屏幕在滚动时将停止DOM的操作，停止滚动后再按队列执行已终止的DOM操作，因此，在这样的系统中，屏幕的滚动事件将无效。

7.2.4　页面显示或隐藏事件

在jQuery Mobile中，当不同页面间或同一个页面不同容器间相互切换时，将触发页面中的显示或隐藏事件。具体的事件类型有4个，说明如下：

1）pagebeforeshow（页面显示前事件）：当页面在显示之前、实际切换正在进行时触发，此事件回调函数传回的数据对象中有一个“prevPage”属性，该属性是一个jQuery集合对象，它可以获取正在切换远离页的全部DOM元素。

2）pagebeforehide（页面隐藏前事件）：当页面在隐藏之前、实际切换正在进行时触发，此事件回调函数传回的数据对象中有一个“nextPage”属性，该属性是一个jQuery集合对象，它可以获取正在切换目标页的全部DOM元素。

3）pageshow（页面显示完成事件）：当页面切换完成时触发，此事件回调函数传回的数据对象中有一个“prevPage”属性，该属性是一个jQuery集合对象，它可以获取切换之前上一页的全部DOM元素。

4）pagehide（页面隐藏完成事件）：当页面隐藏完成时触发，此事件回调函数传回的数据对象中有一个“nextPage”属性，该属性是一个jQuery集合对象，它可以获取切换之后当前页的全部DOM元素。

接下来我们通过一个完整的实例，详细介绍如何绑定页面显示或隐藏各类型事件。

实例7-6　绑定页面的显示与隐藏事件

1.功能说明

新建一个HTML页面，在页面中添加两个“page”容器，对应Id号分别为“page1”和“page2”。在每个容器中添加一个＜a＞元素，用来实现两容器间的切换。在切换过程中绑定页面的显示与隐藏事件，通过浏览器的控制台显示各类型事件执行的详细信息。

2.实现代码

新建一个HTML页面7-6.htm，加入代码如代码清单7-6所示。

代码清单7-6　绑定页面的显示与隐藏事件

```
    ＜! DOCTYPE html＞
    ＜html＞
    ＜head＞
    ＜title＞jQuery Mobile显示与隐藏事件＜/title＞
    ＜meta name="viewport"content="width=device-width，
    initial-scale=1.0，maximum-scale=1.0，user-scalable=0；"/
＞
    ＜link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/＞
    ＜script src="Js/jquery-1.6.4.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"＞＜/script＞
    ＜script src="Js/7-6.js"
```

```
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header">
<h1>头部栏</h1>
</div>
<div data-role="content">
<a href="#page2">下一页</a>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="page2">
<div data-role="header">
<h1>头部栏</h1>
</div>
<div data-role="content">
<a href="#page1">上一页</a>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在上述代码中，<head>元素引用了一个“7-6.js”文件。在该文件中，通过绑定页面的显示与隐藏事件，将事件的执行过程显示在浏览器控制台中，代码如下所示：

```
$（function（）{
$（'div'）.live（'pagebeforehide', function（event, ui）{
console.log（'1.'+ui.nextPage[0].id+'正在显示中……'）;
}）;
$（'div'）.live（'pagebeforeshow', function（event, ui）{
console.log（'2.'+ui.prevPage[0].id+'正在隐藏中……'）;
}）;
$（'div'）.live（'pagehide', function（event, ui）{
console.log（'3.'+ui.nextPage[0].id+'显示完成！'）;
}）;
```

```
$('div').live('pageshow', function(event, ui){
console.log('4.'+ui.prevPage[0].id+'隐藏完成!');
})
})
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-6所示。

图　7-6　控制台捕获执行信息的效果

4.源码分析

在本实例的JavaScript代码中，将＜div＞容器元素与各类型的页面显示和隐藏事件相绑定。在这些事件中，通过调用“console”的log（）方法，记录每个事件中回调函数传回的数据对象属性，这些属性均是jQuery对象。在显示事件中，该对象可以获取切换之前页面

（prevPage）的全部DOM元素；在隐藏事件中，该对象可以获取切换之后页面（nextPage）的全部DOM元素，各事件中获取的返回对象不同，不能混用。

除实例7-3～实例7-6所介绍的事件之外，其他重要的jQuery Mobile事件名称与使用方法如表7-2所示。

表 7-2　jQuery Mobile 中其他重要事件使用说明

事件名	描　述
pagebeforeload	该事件在加载请求发出前触发，在绑定的回调函数中，可以调用 preventDefault() 方法，表示由该事件来处理 load 事件
pageload	该事件当页面加载成功并创建了全部的 DOM 元素后触发，被绑定的回调函数作为一个数据对象，该对象有二个参数，其中第二个参数包含如下信息：url 表示调用地址，absurl 表示绝对地址
pageloadfailed	该事件当页面加载失败时触发，默认情况下，触发该事件后，jQuery Mobile 框架将以页面的形式显示出错信息
pagebeforechange	当页面在切换或改变之前触发该事件，在回调函数中包含两个数据对象参数，其中第一个参数 toPage 表示指定内 / 外部的页面绝对 / 相对地址，第二个参数 options 表示使用 changePage() 方法时的配置选项
pagechange	当完成 changePage() 方法请求的页面并完成 DOM 元素加载时触发该事件，在触发任何 pageshow 或 pagehide 事件之前，此事件已完成了触发
pagechangefailed	当使用 changePage() 方法请求页面失败时触发，其回调函数与 pagebeforechange 事件一样，数据对象包含相同的两个参数
pagebeforecreate	当页面在初始化数据之前触发，在触发该事件之前，jQuery Mobile 的默认部件将自动初始化数据，另外，通过绑定 pagebeforecreate 事件然后返回 false，可以禁止页面中的部件自动操作

（续）

事件名	描　述
pagecreate	当页面在初始化数据之后触发，是用户在自定义自己的部件或增强子部件中标记时，最常调用的一个事件
pageinit	当页面的数据初始化完成，还没有加载 DOM 元素时触发该事件，在 jQuery Mobile 中，AJAX 会根据导航把每个页面的内容加载到 DOM 中，因此，要在任何新页面中加载并执行脚本，就必须绑定 pageinit 事件而不是 ready 事件
pageremove	当试图从 DOM 中删除一个外部页面时触发该事件，在该事件的回调函数中可以调用事件对象的 preventDefault() 方法，防止删除的页面被访问
updatelayout	当动态显示或隐藏内容的组成部分时触发该事件，该事件以冒泡的形式通知页面中需要同时更新的其他组件

7.3　访问地址的相关方法

jQuery Mobile通过API拓展了许多绑定的实用事件，此外，还借助$.mobile对象提供了不少简单、容易上手的方法，其中有些方法在本书2.3节有过介绍，在此不再赘述。本节主要介绍使用$.mobile对象中的方法实现URL地址的转换、验证和域名比较以及纵向滚动的功能。

7.3.1　访问路径和URL地址转换方法

在使用jQuery Mobile开发移动项目过程中，有时需要将文件的访问路径进行统一转换，将一些不规范的文件访问的相对地址转换成标准的绝对地址，这项功能可以通过调用$.mobile对象中的makePathAbsolute（）来实现，该方法调用格式为：

```
$.mobile.path.makePathAbsolute（relPath, absPath）
```

其中，参数relPath为字符型，必填项，表示相对文件的路径；参数absPath为字符型，必填项，表示绝对文件的路径。

该方法的功能是：以绝对路径为标准，根据相对路径所在目录级别，将相对路径转成一个绝对路径，返回值是一个转换成功的绝对路径字符串。

与makePathAbsolute（）相类似，makeUrlAbsolute（）方法是将一些不规范的URL地址，转成统一标准的绝对URL地址，该方法调用的格式为：

```
$.mobile.path.makeUrlAbsolute（relUrl, absUrl）
```

其中，参数relUrl为字符型，必填项，表示相对URL的地址；参数absUrl为字符型，必填项，表示绝对URL的地址，该方法的功能是：以绝对URL地址为标准，根据相对URL地址所在目录级别，将相对URL的地址转成一个绝对URL地址，返回值是一个转换成功的绝对URL地址字符串。

接下来通过一个完整的实例，介绍这两个方法实现的过程。

实例7-7　makePathAbsolute（）和makeUrlAbsolute（）转换方法

1.功能说明

新建一个HTML页面，添加两个“page”容器，通过导航栏切换这两个容器。在第一个容器的文本框中输入一个文件的相对路径后，将返回一个转换后的绝对路径；在第二个容器的文本框中输入一个相对文件的URL地址后，将返回一个转换后的绝对URL地址。

2.实现代码

新建一个HTML页面7-7.htm，加入代码如代码清单7-7所示。

代码清单7-7　makePathAbsolute（）和makeUrlAbsolute（）转换方法

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>makePathAbsolute（）和makeUrlAbsolute（）转换方法
</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/7-7.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/7-7.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="page1">
    <div data-role="header">
    <div data-role="navbar">
    <ul>
    <li><a href="#page1"
    class="ui-btn-active">转换路径</a></li>
    <li><a href="#page2">转换Url</a></li>
    </ul>
    </div>
    </div>
    <div class="dchange">
    <div>绝对路径：</div>
    <div class="dtip"id="page1-a">/a/b/c/index.htm</div>
```

```
    <div>相对路径：</div><input id="page1-txt"type="text"/>
    <div>转换结果：</div><div class="dtip"id="page1-b"">
</div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="page2">
    <div data-role="header">
    <div data-role="navbar">
    <ul>
    <li><a href="#page1">转换路径</a></li>
    <li><a href="#page2"
    class="ui-btn-active">转换Url</a></li>
    </ul>
    </div>
    </div>
    <div class="dchange">
    <div>绝对Url：</div>
    <div class="dtip"id="page2-a">
    http：//rttop.cn/a/b/c/index.htm</div>
    <div>相对Url：</div><input id="page2-txt"type="text"/>
    <div>转换结果：</div><div class="dtip"id="page2-b">
</div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-7.js”文件。在该文件中，分别调用$.mobile对象的makePathAbsolute（）和makeUrlAbsolute（）方法，将相对路径或URL地址转成绝对字符串。7-7.js文件代码如下所示：

```
    $（"#page1"）.live（"pagecreate", function（）{
    var$p1="#page1-";
    $（$p1+"txt"）.bind（"change", function（）{
```

```
var strPath=$（$p1+"a"）.html（）;
var absPath=$.mobile.path.makePathAbsolute（
$（this）.val（）, strPath）;
$（$p1+"b"）.html（absPath）
}）
}）
$（"#page2"）.live（"pagecreate", function（）{
var$p2="#page2-";
$（$p2+"txt"）.bind（"change", function（）{
var strPath=$（$p2+"a"）.html（）;
var absPath=$.mobile.path.makeUrlAbsolute（
$（this）.val（）, strPath）;
$（$p2+"b"）.html（absPath）
}）
}）
```

在代码清单7-7中，＜head＞元素还引用了一个“7-7.css”文件，该文件用于控制各容器的框架结构与元素的样式，代码如下所示：

```
/*设置示例区的样式*/
.dchange
{
float: left; padding: 5px
}
.dchange.dtip
{
border: solid 1px#ccc; color: #666;
background-color: #eee; padding: 3px;
height: 23px; line-height: 23px
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-7所示。

图　7-7　将相对路径或URL转成绝对字符串

4. 页面效果

在本实例中，通过导航栏切换页面中的两个容器，为了确保在容器的切换过程中JavaScript代码依然被执行，需要将JavaScript代码分别放置在各容器的“pagecreate”事件中。

在Id号为“page1”的容器中，“pagecreate”事件首先定义一个“$p1”保存容器各元素的公共特征，然后设置文本框“change”事件。在该事件中，将获取的绝对路径值保存在变量“strPath”中，并将获取文本框值和“strPath”变量作为实参，调用makePathAbsolute（）。最后，将返回的绝对字符串保存在变量“absPath”中，并显示在页面元素中。

在Id号为“page2”的容器中获取绝对URL地址字符串的过程，与在Id号为“page1”的容器中获取绝对路径字符串的过程基本相同，只是调用的方法不一样，不再赘述。

7.3.2　URL地址验证方法

在jQuery Mobile中，$.mobile对象提供了两个URL地址验证的方法，分别为isRelativeUrl（）和isAbsoluteUrl（）。前者可以验证指定的URL地址是否为相对URL地址，该方法调用的格式如下：

```
$.mobile.path.isRelativeUrl（url）
```

其中，参数url为字符型，必填项，表示一个相对或绝对URL地址。该方法返回一个布尔值，即如果是相对URL地址，则返回true，否则返回false。

isAbsoluteUrl（）可以验证指定的URL地址是否为绝对URL地址，该方法调用的格式如下：

```
$.mobile.path.isAbsoluteUrl（url）
```

其中，参数url为字符型，必填项，表示一个相对或绝对URL地址。该方法返回一个布尔值，即如果是绝对URL地址，则返回true，否则返回false。

接下来通过一个完整的实例，介绍这两个方法实现的过程。

实例7-8　isRelativeUrl（）和isAbsoluteUrl（）验证方法

1.功能说明

新建一个HTML页面，分别添加两个“page”容器。在第一个容器的文本框中输入任意URL地址，如果是相对地址，在页面中显示“是”，否则显示“否”；在第二个容器的文本框中输入任意URL地址，如果是绝对地址，在页面中显示“是”，否则显示“否”。

2.实现代码

新建一个HTML页面7-8.htm，加入代码如代码清单7-8所示。

代码清单7-8　isRelativeUrl（）和isAbsoluteUrl（）验证方法

```
<! DOCTYPE html>
<html>
<head>
<title>isRelativeUrl（）和isAbsoluteUrl（）验证方法</title
>
<meta name="viewport"content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/7-7.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/7-8.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header">
```

```
    <div data-role="navbar">
    <ul>
    <li><a href="#page1"
    class="ui-btn-active">相对Url</a></li>
    <li><a href="#page2">绝对Url</a></li>
    </ul>
    </div>
    </div>
    <div class="dchange">
    <div>相对Url：</div>
    <input id="page1-txt"type="text"/>
    <div>验证结果：</div>
    <div class="dtip"id="page1-b""></div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="page2">
    <div data-role="header">
    <div data-role="navbar">
    <ul>
    <li><a href="#page1">相对Url</a></li>
    <li><a href="#page2"
    class="ui-btn-active">绝对Url</a></li>
    </ul>
    </div>
    </div>
    <div class="dchange">
    <div>绝对Url：</div>
    <input id="page2-txt"type="text"/>
    <div>转换结果：</div>
    <div class="dtip"id="page2-b"></div>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-8.js”文件。在该文件中，分别调用$.mobile对象的isRelativeUrl（）和

isAbsoluteUrl（）方法，用来验证输入的任意URL地址。7-8.js文件代码如下所示：

```
$（"#page1"）.live（"pagecreate", function（）{
var$p1="#page1-";
$（$p1+"txt"）.bind（"change", function（）{
var blnResult=$.mobile.path.isRelativeUrl（
$（this）.val（））?"是": "否";
$（$p1+"b"）.html（blnResult）
}）
}）;
$（"#page2"）.live（"pagecreate", function（）{
var$p2="#page2-";
$（$p2+"txt"）.bind（"change", function（）{
var blnResult=$.mobile.path.isAbsoluteUrl（
$（this）.val（））?"是": "否";
$（$p2+"b"）.html（blnResult）
}）
}）;
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-8所示。

图　7-8　对输入的任意URL地址验证的不同效果

4.页面效果

本实例分别在两个“page”容器的“pagecreate”事件中，编写触发对应文本框“change”事件的代码。在第一个“page”容器的文本框“change”事件中，将获取文本框的值作为调用isRelativeUrl（）方法的实参，通过“blnResult”变量保存方法调用时的返回值，并将该值显示在指定Id号的页面元素中。

在第二个“page”容器的文本框“change”事件中，同样将获取的文本框的值作为实参，传给isAbsoluteUrl（），将该方法的返回值赋予“blnResult”变量，并将该值显示在指定Id号的页面元素中，详细实现过程如代码中加粗部分所示。

7.3.3　域名比较方法

在jQuery Mobile中，除提供URL地址验证的方法外，还可以通过isSameDomain（）方法比较两个任意URL地址字符串内容是否为同一个域名，该方法的调用格式如下：

```
$.mobile.path.isSameDomain（url1，url2）
```

其中，参数url1为字符型，必填项，是一个相对的URL地址字符串；另一个参数url2为字符型，必填项，是一个相对或绝对的URL地址字符串。

该方法的功能是：当url1与url2的域名相同时，返回true，否则返回false。

接下来通过一个完整的实例，介绍该方法实现的过程。

实例7-9　域名比较方法isSameDomain（）的使用

1.功能说明

新建一个HTML页面，添加一个“page”容器，并在容器中增加两个文本框。当用户在两个文本框中输入不同URL地址后，将调用

isSameDomain（）方法对这两个地址进行比较，如果是相同域名则在页面中显示“是”，否则显示“否”。

2.实现代码

新建一个HTML页面7-9.htm，加入代码如代码清单7-9所示。

代码清单7-9　域名比较方法isSameDomain（）的使用

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>isSameDomain（）域名比较方法</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/7-7.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/7-9.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="page1">
    <div data-role="header">
    <h1>域名比较</h1>
    </div>
    <div class="dchange">
    <div>地址1：</div><input id="page1-txt1"type="text"/>
    <div>地址2：</div><input id="page1-txt2"type="text"/>
    <div>验证结果：</div><div class="dtip"id="page1-b"">
</div>
    </div>
```

```
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在上述代码中，<head>元素引用了一个“7-9.js”文件。在该文件中调用$.mobile对象的isSameDomain（）方法，验证输入的任意两个URL地址是否同属一个域名，代码如下所示：

```
    $（"#page1"）.live（"pagecreate", function（）{
    var$p1="#page1-";
    $（"#page1-txt1, #page1-txt2"）.live（"change",
function（）{
    var$txt1=$（$p1+"txt1"）.val（）;
    var$txt2=$（$p1+"txt2"）.val（）;
    if（$txt1! =""＆＆$txt2! =""）{
    var blnResult=$.mobile.path.isSameDomain（
    $（$p1+"b"）.html（blnResult）
    }
    }）
    }）;
```

$txt1, $txt2）?"是": "否";

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-9所示。

图　7-9　验证URL地址是否同属一个域名的不同效果

4.页面效果

在本实例中，两个文本框都绑定了一个“change”事件，在该事件中，先获取各文本框输入的值，并分别保存在变量“$txt1”和“$txt2”中；然后，判断这两个变量的值是否为空，如果不为空，将这两个值作为调用isSameDomain（）方法的实参，再根据该方法的返回值转换成相应的“是”或“否”字符，并将该字符赋值于变量“blnResult”；最后，通过指定Id号的元素将该变量值显示在页面中，详细实现过程见代码中加粗部分。

7.3.4　纵向滚动方法

在jQuery Mobile中，$.mobile对象还提供了一个纵向滚动的方法silentScroll（），即滚动至Y轴的一个指定位置。该方法在执行时不会触滚动事件，它的调用格式如下：

```
$.mobile.silentScroll（yPos）
```

该方法的功能是：在Y轴上滚动到指定的位置，其中参数yPos为整数型，默认值为0，如果参数值为10，则表示整个屏幕向上滚动到Y轴的10px处。

接下来通过一个完整的实例，介绍该方法实现的过程。

实例7-10　纵向滚动方法silentScroll（）的使用

1.功能说明

新建一个HTML页面，添加一个＜span＞元素，并将它的初始内容设置为“开始”字样。单击该元素时，它的内容变成不断增加的动态数值，并且整个屏幕也按照该值的距离不断向上滚动，直到该值为显示为“30”时终止。

2.实现代码

新建一个HTML页面7-10.htm，加入代码如代码清单7-10所示。

代码清单7-10　纵向滚动方法silentScroll（）的使用

```
<! DOCTYPE html>
<html>
<head>
<title>silentScroll（）纵向滚动方法</title>
<meta name="viewport"content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/7-7.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/7-10.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header">
<h1>纵向滚动</h1>
</div>
<div data-role="content">
<div class="dchange">
<div>
正在向上滚动距离是:
<span id="page1-a1"class="dtip">开始</span>
</div>
</div>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在上述代码中，＜head＞元素引用了一个“7-10.js”文件。在该文件中，自定义一个“AutoScroll”函数，该函数调用silentScroll（）方法滚动屏幕，并通过调用一个定时器方法，实现屏幕动态向上滚动的效果。7-10.js文件代码如下所示：

```
var$intInterval;
var$intHeight=0;
var$p1="#page1-";
$（"#page1"）.live（"pagecreate", function（）{
$（$p1+"a1"）.live（"click", function（）{
$intInterval=window.setInterval（"AutoScroll（）", 1000）;
}）
}）
//编写自动滚动函数
function AutoScroll（）{
if（$intHeight＜30）{
$.mobile.silentScroll（$intHeight）;
$（$p1+"a1"）.html（$intHeight）;
$intHeight=$intHeight+2;
}else{
window.clearInterval（$intInterval）;
}
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图7-10所示。

图　7-10　屏幕动态向上滚动时的效果

4.页面效果

在本实例的JavaScript代码中，首先自定义了一个“AutoScroll”函数。该函数的功能是：当保存滚动高度变量“$intInterval”的值小于“30px”时，以该值作为实参调用silentScroll（）方法，使屏幕向上滚动至Y轴的变量“$intInterval”值处，并在页面的＜span＞元素中显示该变量值；同时，变量“$intInterval”的值累加“2px”；当变量“$intInterval”的值大于或等于“30px”时，调用“window”对象中的clearInterval（）方法清除定时器变量“$intInterval”，从而终止设置的定时操作。在容器的“pagecreate”事件中，设置＜span＞元素的“click”事件，在该事件中调用“window”对象中

setInterval（）方法，设置每隔1秒执行一次“AutoScroll”函数的定时操作。通过该定时操作，可以实现屏幕动态向上滚动的效果。

7.4　本章小结

本章首先通过两个实例的完整开发过程，介绍了如何更改jQuery Mobile基本配置项；然后，再通过四个实用、简洁的实例，由浅入深地介绍了一些常用事件的使用方法，使读者进一步掌握高效绑定事件的技巧；最后，通过一个个精选的小实例，详细介绍了jQuery Mobile中几个常用方法的调用过程，为读者全面了解与掌握jQuery Mobile提供的API使用方法打下扎实的理论与实践基础。

第8章　jQuery Mobile开发技巧与最佳实践

本章内容

开启或禁用列表项中的箭头

使用悬浮的方式固定头部栏与尾部栏

初始化页面随机显示背景图

按钮标题文字的控制

侦听HTML 5画布元素的触摸事件

在jQuery Mobile中提交表单数据

切换按钮的开启/禁用状态

开启或禁用AJAX方式打开页面链接

使用localStorage传递链接参数

在jQuery Mobile中构建离线功能

本章小结

jQuery Mobile作为jQuery插件库的附属成员，完全继承了jQuery的“少量代码实现大量功能”的主旨，其轻量级的UI框架、相对其他语言的低学习成本，都是受到青睐的原因。但作为一个新型的移动框架，在使用它开发项目的过程中，还有许多需要注意的地方。通过本章总结的一些开发技巧与实现经验的学习，使开发人员尽量少走弯路，不断提升代码执行的效率与性能。

8.1　开启或禁用列表项中的箭头

在jQuery Mobile中，＜ul＞列表的使用十分频繁，几乎所有需要加载大量格式化数据的时候都会考虑使用该元素。为了单击列表选项时链接到某个页面，在列表的＜li＞选项元素中，常常会增加一个＜a＞元素，用于实现单击列表项进行链接的功能。一旦添加＜a＞元素后，jQuery Mobile默认会在列表项的最右侧自动增加一个圆形背景的小箭头，用来表示列表中的选项是一个超级链接。

当然，在实际的开发过程中，开发者可以通过修改数据集中的图标属性“data-icon”，实现该小箭头图标开启与禁用的功能。

接下来通过一个完整的实例，介绍这个方法实现的过程。

实例8-1　开启或禁用列表项中的箭头

1.功能说明

新建一个HTML页面，在页面中添加两个“page”容器，一个用于显示“启用”箭头图标的列表项，另一个用于显示“禁用”箭头图标的列表项，且各列表项都可以单击链接。

2.实现代码

新建一个HTML页面8-1.htm，加入代码如代码清单8-1所示。

代码清单8-1　开启或禁用列表项中的箭头

```
<! DOCTYPE html>
<html>
<head>
<title>开启或禁用列表项中的箭头</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#page1"class="ui-btn-active">启用</a>
</li>
<li><a href="#page2">禁用</a></li>
</ul>
</div>
</div>
<ul data-role="listview">
<li><a href="#">计算机</a></li>
<li><a href="#">社科</a></li>
<li><a href="#">文艺</a></li>
</ul>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="page2">
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#page1">启用</a></li>
```

```
    <li><a href="#page2"class="ui-btn-active">禁用</a>
</li>
    </ul>
    </div>
    </div>
    <ul data-role="listview">
    <li data-icon="false"><a href="#">计算机</a></li>
    <li data-icon="false"><a href="#">社科</a></li>
    <li data-icon="false"><a href="#">文艺</a></li>
    </ul>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-1所示。

图　8-1　开启或禁用列表项中的箭头时的效果

4.源码分析

在本实例中，通过设置列表项中＜a＞元素的“data-icon”属性值，可以开启或禁用列表项中最右侧箭头图标的显示状态。该属性默认值为“true”，表示显示，如果设置为“false”则为禁用。

如果在“data-role”属性值为“button”的＜a＞元素中，“data-icon”属性值为按钮中的图标名称；如“data-icon”属性值为“delete”，则显示一个“删除”按钮的小图标。也可以将该属性值设置为“true”或“false”，用来开启或禁用按钮中图标的显示状态。

此外，在jQuery Mobile中，＜a＞元素的“data-icon”属性可以控制图标的显示状态，而另外一个“data-mini”属性则可以控制按钮显示时的高度。该属性默认值为“false”，表示正常高度显示；如果设置为“true”，将显示一个高度紧凑型的按钮。

8.2　使用悬浮的方式固定头部栏与尾部栏

在移动设备的浏览器中查看页面时，默认页面滑动是从上至下，或从下至上的方式。如果加载的内容较多页面很长时，要从尾部栏返回头部栏中导航条再单击链接地址，以这种方式就会比较麻烦。

在头部栏或尾部栏的容器元素中增加“data-position”属性，将该属性值设置为“fixed”，可以将滚动屏幕时隐藏的头部栏或尾部栏，在停止滚动或单击时重新出现；再次滚动时，又自动隐藏，由此实现将头部栏或尾部栏以悬浮的形式固定在原有位置。

接下来通过一个完整的实例，介绍该方法实现的过程。

实例8-2　使用悬浮的方式固定头部栏与尾部栏

1.功能说明

新建一个HTML页面，并分别添加一个头部栏和尾部栏容器，当上下滚动滑动屏幕时，头部栏和尾部栏自动隐藏；停止滚动时，又自动显示在原有位置。

2.实现代码

新建一个HTML页面8-2.htm，加入代码如代码清单8-2所示。

代码清单8-2　使用悬浮的方式固定头部栏与尾部栏

```
<! DOCTYPE html>
<html>
<head>
<title>使用悬浮的方式固定头部与尾部栏</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page">
<div data-role="header"data-position="fxed">
<h1>荣拓科技</h1>
</div>
<ul data-role="listview">
<li><a href="#">计算机</a></li>
<li><a href="#">社科</a></li>
<li><a href="#">文艺</a></li>
<li><a href="#">生活</a></li>
<li><a href="#">历史</a></li>
<li><a href="#">经济</a></li>
</ul>
<div data-role="footer"data-position="fxed">
<h4>©2012 rttop.cn studio</h4>
</div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-2所示。

图　8-2　使用悬浮的方式固定头部栏与尾部栏

4.源码分析

在本实例中，在工具栏中添加“data-position”属性，可以使头部栏或尾部栏以悬浮的形式显示在原有位置上。用户在上下滑动屏幕时，仍然可以方便地使用它们。

此外，在工具栏中，还可以增加全屏显示属性“data-fullscreen”。如果将该属性的值设置为“true”，那么当以全屏的方式浏览图片或其他信息时，工具栏仍然以悬浮的形式显示在全屏的页面上。与“data-position”属性不同，属性“data-fullscreen”并不是在原有位置上的隐藏与显示切换，而是在屏幕中完全消失，当出现全屏幕页面时，又重新返回页面中。

8.3　初始化页面随机显示背景图

在jQuery Mobile中，页面的加载过程与在jQuery中并不一样，它可以很容易地捕捉到一些非常有用的事件，例如“pagecreate”页面初始化事件，该事件中所有请求的DOM元素已经完成了创建，正在开始加载，此时，用户可以自定义部件元素，实现一些自定义样式效果，如显示加载进度条或随机显示页面背景图片等。

接下来通过一个完整的实例，介绍该方法实现的过程。

实例8-3　初始化页面随机显示背景图

1.功能说明

新建一个HTML页面，在页面的正文区域中添加一个＜p＞元素。每次加载页面时，该元素的背景色将以随机的方式显示。

2.实现代码

新建一个HTML页面8-3.htm，加入代码如代码清单8-3所示。

代码清单8-3　初始化页面过程中随机显示背景图

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞初始页面过程中随机显示背景图＜/title＞
```

```
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <link href="Css/8-3.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <script src="Js/8-3.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="page1">
    <div data-role="header"><h1>头部栏</h1></div>
    <div data-role="content">
    <p class="p p0"id="pChange">
    随机显示背景图片
    </p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在代码清单8-3中，页面的<head>元素包含了一个“8-3.js”文件。在该文件中，页面容器“page1”绑定“pagecreate”事件，在该事件中实现<p>元素背景色随机变化的功能，8-3.js文件代码如下所示：

```
    $('#page1').live("pagecreate", function(){
    var$randombg=Math.floor(Math.random()*4); //0 to 3
    var$p=$('#pChange');
    $p.removeClass("p0").addClass('p'+$randombg);
    })
```

此外，在代码清单8-3中，页面的＜head＞元素还包含了一个“8-3.css”文件。该样式文件用于定义不同类别名的随机样式和控制＜p＞元素的显示样式，代码如下所示：

```
.p/*p元素基本样式*/
{
text-align: center;
border: solid 1px#ccc;
font-size: 14px;
line-height: 28px
}
.p0/*p元素随机样式*/
{
background: transparent url（images/bg-0.png）0 0 repeat
}
.p1
{
background: transparent url（images/bg-1.png）0 0 repeat
}
.p2
{
background: transparent url（images/bg-2.png）0 0 repeat
}
.p3
{
background: transparent url（images/bg-3.png）0 0 repeat
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-3所示。

图　8-3　随机显示背景图片

4.源码分析

本实例通过绑定页面中“page1”容器的“pagecreate”事件，使页面在初始化过程中随机获取＜p＞元素的背景图片并显示。在对应的JavaScript代码中，先将0～3之间的随机数保存在变量“$randombg”中，然后通过jQuery中的“removeClass”方法移除原有的样式类别，并调用“addClass”方法将随机数组合的样式添加至＜p＞元素中，从而实现元素背景色随机显示的功能。

此外，当页面以动态方式加载较多内容时，可以在“pagecreate”事件中定义一个进度条图片，在数据开始加载时显示该图片，加载完成后自动隐藏该图片，极大地丰富了用户体验。

8.4　按钮标题文字的控制

在jQuery Mobile中，如果列表选项或按钮中的标题文字过长时，将自动被截断，并用“……”符号表示被截断的部分。当然，该功能也可以通过重置“ui-btn-text”类别属性恢复正常显示。此外，如果在按钮中将“data-iconpos”的属性值设为“notext”，还可以创建一个没有任何标题文字的按钮，接下来，通过一个实例介绍这两个功能实现的方法。

实例8-4　按钮标题文字的控制

1.功能说明

新建一个HTML页面，在正文区域中添加两个“data-role”属性值为“button”的＜a＞元素，第一个正常显示按钮中超长的标题文字，第二个不显示按钮的标题。

2.实现代码

新建一个HTML页面8-4.htm，加入代码如代码清单8-4所示。

代码清单8-4　按钮中标题文字的控制

```
＜! DOCTYPE html＞
＜html＞
```

```
<head>
<title>按钮中标题文字的控制</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<style type="text/css">
.ui-btn-text{white-space: normal}
</style>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<a href="#"data-role="button"
data-theme="a">一个很长标题文字的按钮</a>
<a href="#"data-role="button"
data-theme="a"data-iconpos="notext">
这是一个很长标题的按钮</a>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-4所示。

图　8-4　控制按钮标题文字的效果

4.源码分析

在本实例中，将“ui-btn-text”的类别值重置为“white-space：normal”，所有使用该类别的按钮标题文字将正常显示，不再出现截断显示的状态；如果在按钮的＜a＞元素中，将添加的“data-iconpos”属性值设置为“notext”，可以创建一个无标题的按钮。

此外，如果在一个列表项＜li＞中，标题和段落重置的类别名称分别为“ui-li-heading”、“ui-li-desc”，前者用于描述列表中文本标题的样式，后者用于描述列表中段落文本的样式，使用方法如下所示：

```
.ui-li-heading{white-space: normal; color: Red}
.ui-li-desc{white-space: normal; color: Green}
```

上述代码中，按正常长度显示列表项＜li＞中标题和段落的文字内容，并且将列表项＜li＞中标题的文字重置为“红色”，段落的文字重置为“绿色”。

8.5　侦听HTML 5画布元素的触摸事件

jQuery Mobile中大量应用了HTML 5的新增特征和元素，＜canvas＞画布元素就是其中之一。由于jQuery Mobile支持绝大多数的触摸事件，因此，可以很轻松地绑定画布的“tap”触摸事件，获取用户在触摸时返回的坐标数据信息。

接下来通过一个实例详细介绍在画布指定位置中绘制触摸点的方法。

实例8-5　侦听HTML 5画布元素的触摸事件

1.功能说明

新建一个HTML页面，在内容区域添加一个＜canvas＞画布元素。触摸画布时，将在触摸处绘制一个半径为5的实体小圆心，同时在画布的最上面显示此次触摸时的坐标数据信息。

2.实现代码

新建一个HTML页面8-5.htm，加入代码如代码清单8-5所示。

代码清单8-5　侦听HTML 5画布元素的触摸事件

```
＜! DOCTYPE html＞
```

```
<html>
<head>
<title>侦听HTML5画布元素的触摸事件</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/8-5.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/8-5.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<span id="spnTip"></span>
<canvas id="cnvMain"></canvas>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在上述代码清单8-5中，<head>元素包含了一个“8-5.js”文件，在该文件中编写代码绑定画布的“tap”事件，并显示该事件返回的坐标数据信息，8-5.js文件如下所示：

```
$（function（）{
var cnv=$（"#cnvMain"）;
var cxt=cnv.get（0）.getContext（'2d'）;
var w=window.innerWidth/1.2;
var h=window.innerHeight/1.2;
var$tip=$（'#spnTip'）;
cnv.attr（"width", w）;
```

```
cnv.attr（"height", h）;
//绑定画布的tap事件
cnv.bind（'tap', function（event）{
var obj=this;
var t=obj.offsetTop;
var l=obj.offsetLeft;
while（obj=obj.offsetParent）{
t+=obj.offsetTop;
l+=obj.offsetLeft;
}
tapX=event.pageX;
tapY=event.pageY;
//开始画圆
cxt.beginPath（）;
cxt.arc（tapX-l, tapY-t, 5, 0, Math.PI*2, true）;
cxt.closePath（）;
//填充圆的颜色
cxt.fllStyle="#666";
cxt.fll（）;
$tip.html（"X: "+（tapX-l）+"Y: "+tapY）;
}）
}）
```

另外，在代码清单8-5中，＜head＞元素还包含了一个“8-5.css”文件，在该文件中定义画布的基本样式，代码如下所示：

```
canvas
{
border: dashed 1px#666;
cursor: pointer
}
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-5所示。

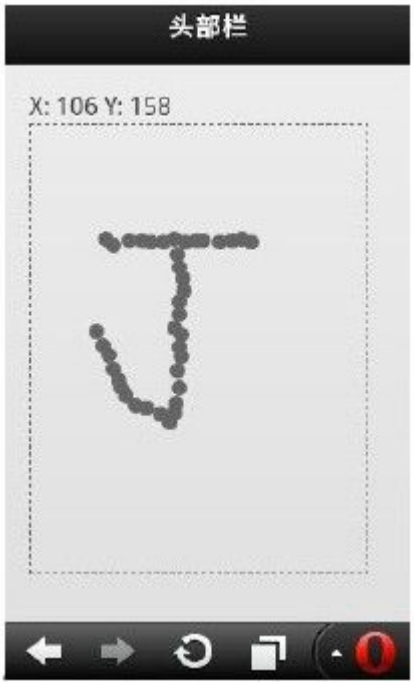


图　8-5　侦听HTML 5画布元素的触摸事件

4.源码分析

在本实例的JavaScript代码中，首先获取页面中的画布元素并保存在变量“cnv”中，并通过画布变量“cnv”取得画布的上下文环境对象保存在变量“cxt”中。接下来，根据文档显示区的宽度与高度计算出画布显示时的宽度与高度，并分别保存在变量“w”和“h”中，再通过attr（）方法将这两个值赋予画布元素。

然后，通过bind（）方法绑定画布元素的“tap”事件。在该事件中，先计算画布元素在屏幕中的坐标距离并保存至变量“t”和“l”中。“l”表示横坐标，在计算时该值时，先通过“offsetLeft”属性获取画布元素的左边距离，如果画布元素还存在父容器，则通过

“while”语句将父容器的左边距离与画布元素的左边距离相累加，并将最终值赋予变量“l”；“t”表示纵坐标，同样道理，将计算后的画布上边距离最终值赋予变量“t”。另外，通过“tapX”、“tapY”两个变量分别记录触摸画布时返回的横坐标与纵坐标的值。

最后，开始画圆。圆的横坐标为触摸事件返回的横坐标值“tapX”减去画布在屏幕中的横坐标值“l”，因为“tapX”变量值包含“l”变量值，两者相减后，就是画布中圆的真实横坐标值；同理，变量“tapY”与“t”相减后得到画布中圆的真实纵坐标值；根据获取的圆的坐标值，以5像素为半径。在画布中，调用arc（）方法绘制一个360度的圆形，通过fill（）方法为圆形填充设置的颜色，并将圆形的坐标位置显示通过＜span＞元素显示在页面中。

说明　在开始计算画布的宽与高时，使用的“1.2”是一个常用值，读者可以自行调整，它的作用是：使画布元素的宽与高能够很好地填充到屏幕中。

8.6　在jQuery Mobile中提交表单数据

提交表单数据，是项目开发过程中重要的一项操作。在jQuery Mobile中，借助jQuery中的serialize（）方法，可以将表单中的每项数据字段序列化，通过$.ajax（）方法可以将序列化后的数据以异步的形式提交给服务端。服务器处理完成后，返回的信息可以在$.ajax（）方法请求成功时，通过回调函数进行处理，从而完成一次完整的表单数据请求。

接下来通过一个用户登录实例详细介绍如何使用表单提交数据。

实例8-6　在jQuery Mobile中提交表单数据

1.功能说明

在新建的HTML页面中添加一个表单元素，在表单中增加两个用于输入用户名和密码的文本框元素，以及一个用于登录的提交按钮。用户完成用户名和密码的输入单击按钮时，如果登录成功，显示“操作提示，登录成功！”，否则显示“用户名或密码错误！”。

2.实现代码

新建一个HTML页面8-6.htm，加入代码如代码清单8-6所示。

代码清单8-6　在jQuery Mobile中提交表单数据

```
<! DOCTYPE html>
<html>
<head>
<title>在jQuery Mobile中提交表单数据</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/8-6.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<form id="form1"name="form1">
<label for="Name">用户名：</label>
<input type="text"name="Name"
id="Name"value=""/>
<label for="Pass">密码：</label>
<input type="password"name="Pass"
id="Pass"value=""/>
<div id="divTip"></div>
<input type="submit"name="btnSub"
id="btnSub"value="登录"/>
</form>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在代码清单8-6中，＜head＞元素包含了一个“8-6.js”文件。在该文件中绑定“登录”按钮的单击事件，在该事件中通过$.ajax（）方法提交表单的数据，8-6.js文件如下所示：

```
$（function（）{
$（"#btnSub"）.click（function（）{
var frmData=$（"#form1"）.serialize（）;
$.ajax（{
type: "POST",
url: "8-6.aspx",
cache: false,
data: frmData,
success: function（data）{
if（data=="True"）{
$（"#divTip"）.html（"操作提示，登录成功！"）;
}
else{
$（"#divTip"）.html（"用户名或密码错误！"）;
}
}
}）
return false;
}）
}）
```

在代码清单8-6中，数据请求的服务端页面为“8-6.aspx”。该页面的功能是：获取前台页面序列化后的数据，简单处理完成后，返回前台页面一个数据信息，关键代码如下所示（.NET版）：

```
……
string strName=Request["Name"];
string strPass=Request["Pass"];
bool blnLogin=false;
if（strName=="admin"＆＆strPass=="123456"）
{
blnLogin=true;
```

```
}
Response.Clear（）；
Response.Write（blnLogin）；
Response.End（）；
……
```

当然，请求的服务端页面也可以使用PHP来编写代码，功能相同，关键代码如下所示（PHP版）：

```
＜?php
$strName=$_POST[Name];
$strPass=$_POST[Pass];
$blnLogin=false;
if（（$strName='admin'）＆＆（$strPass='123456'））{
$blnLogin=true;
}
echo（$blnLogin）；
?＞
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-6所示。

图　8-6　在jQuery Mobile中提交表单数据的不同效果

4.源码分析

在本实例的JavaScript代码中，使用$.ajax（）方法提交表单数据前，先将整个表单的数据通过serialize（）方法进行序列化。serialize（）方法可以序列化表单值，创建URL编码文本字符串，该字符串可以在生成AJAX请求时查看。如本实例登录成功时，表单提交后的序列化字符串为：“Name=admin&Pass=123456”，该字符串将以“键/值”的形式与表单中全部数据字段相对应。

当服务端在完成数据的处理后，返回的数据信息可以在$.ajax（）方法请求成功时，通过回调函数获取数据，并根据该数据信息作相应的显示。

8.7　切换按钮的开启/禁用状态

在使用jQuery Mobile开发移动项目过程中，有时需要对表单中的按钮进行动态地控制。例如，当用户登录时，如果用户名和密码这两项的内容都为空，那么“登录”按钮将是不可用的；而如果两项内容中至少一项不为空，那么“登录”按钮将是可用的。要实现这个效果，需要在JavaScript代码中调用按钮的button（）方法。

接下来通过一个简单的实例详细介绍如何实现这一效果。

实例8-7　切换按钮的开启/禁用状态

1.功能说明

新建一个HTML页面，在正文区域中添加一个“开关”组件和一个类型为“submit”的提交按钮。用户滑动开关后，该按钮的可用性状态将随开关滑动值的变化而变化。

2.实现代码

新建一个HTML页面8-7.htm，加入代码如代码清单8-7所示。

代码清单8-7　开启或禁用按钮的可用状态

```
<! DOCTYPE html>
```

```
<html>
<head>
<title>开启或禁用按钮的可用状态</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/8-7.js"
type="text/javascript">
</script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<select name="slider"id="slider"data-role="slider">
<option value="1">开</option>
<option value="0">关</option>
</select>
<input type="submit"name="btnTmp"
id="btnTmp"value="提交"/>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></</div>
</div>
</body>
</html>
```

在代码清单8-7中，<head>元素中包含了一个“8-7.js”文件。在该文件中绑定开关组件的“change”事件，实现动态开启或禁用按钮可用状态的功能。8-7.js文件如下所示：

```
$（function（）{
$（"#slider"）.bind（"change", function（）{
if（$（this）.val（）==0）{
$（'#btnTmp'）.button（'disable'）;
```

```
}else{
$('#btnTmp').button('enable');
}
})
})
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-7所示。

图　8-7　切换按钮的开启/禁用状态

4.源码分析

在本实例的JavaScript代码中，先绑定开关组件的“change”事件。在该事件中，用户从“开”状态切换至“关”状态时，开关组件

的值为“0”，此时，将按钮的状态通过button（）方法设置为“disable”值，表示不可用；用户从“关”状态切换至“开”状态时，开关组件的值为“1”，此时，再将按钮的状态值设置为“enable”，表示可用。

根据上述实现效果可以在用户登录页面中使用。在使用过程中，触发按钮改变状态的是文本框中的值，如果都为空，则按钮的状态值为“disable”，否则为“enable”。

说明　按钮的button（）方法只是针对表单中的按钮，即通过＜input＞元素指定类型来创建，而对于＜a＞元素中通过“data-role”属性创建的按钮则无效。

8.8　开启或禁用AJAX方式打开页面链接

在jQuery Mobile中，在同一域名下的所有页面链接都会自动转成AJAX请求，使用哈希值来指向内部的链接页面，通过动画效果实现页面间的切换。但这种链接方式仅限于目标页面是单个“page”容器，如果目标页面中存在多个“page”容器，必须禁止使用AJAX请求的方式链接，才能在打开目标页面之后完成各个“page”之间的正常切换功能。

接下来通过一个简单的实例详细介绍实现的过程。

实例8-8　开启或禁用AJAX方式打开页面链接

1.功能说明

新建两个HTML页面，一个作为链接源页面，另一个作为目标链接页。在链接源页面中添加两个“page”容器，当切换至第二个容器并单击“更多”链接时，进入目标链接页。在目标链接页中也添加两个“page”容器，当切换到第二个容器并单击“返回”链接时，重返链接源页面。

2.实现代码

新建一个HTML页面8-8.htm作为链接源页面，加入代码如代码清单8-8所示。

代码清单8-8　开启或禁用AJAX方式打开页面链接

```
<! DOCTYPE html>
<html>
<head>
<title>开启或禁用AJAX方式打开页面链接</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1_1">
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#page1_1"
class="ui-btn-active">图书</a></li>
<li><a href="#page1_2">音乐</a></li>
</ul>
</div>
</div>
<div data-role="content"><p>这是图书页</p></div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="page1_2">
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#page1_1">图书</a></li>
<li><a href="#page1_2"
class="ui-btn-active">音乐</a></li>
</ul>
```

```
    </div>
    </div>
    <div data-role="content">
    <p>这是音乐页，
    <a href="target.htm"data-ajax="false">更多</a>
    </p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

在代码清单8-8中，单击“更多”链接进入目标链接页“target.htm”，为此，新建另外一个HTML页面target.htm，加入以下代码：

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>开启或禁用AJAX方式打开页面链接</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1"/>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="Stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="page2_1">
    <div data-role="header">
    <div data-role="navbar">
    <ul>
    <li><a href="#page2_1"
    class="ui-btn-active">流行</a></li>
    <li><a href="#page2_2">通俗</a></li>
    </ul>
    </div>
```

```
    </div>
    <div data-role="content">
    <p>这是音乐中的流行歌曲页</p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    <div data-role="page"id="page2_2">
    <div data-role="header">
    <div data-role="navbar">
    <ul>
    <li><a href="#page2_1">流行</a></li>
    <li><a href="#page2_2"
    class="ui-btn-active">通俗</a></li>
    </ul>
    </div>
    </div>
    <div data-role="content">
    <p>这是音乐中的通俗歌曲页，
    <a href="8-8.htm"data-ajax="false">返回</a></p>
    </div>
    <div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
    </div>
    </body>
    </html>
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-8所示。

图　8-8　开启和禁用AJAX方式打开页面链接的效果

4.源码分析

在本实例中，链接源页面与目标链接页间都放置了多个“page”容器，如果按照默认的方式使用AJAX请求页面链接，那么在打开目标页时只能显示默认的第一个容器，而打开其他容器的链接将无效，原因是使用AJAX记录链接历史的哈希值与页面内部链接指向的哈希值存在冲突。为了解决这个问题，我们在链接多容器的目标页时将链接元素的“data-ajax”属性值设置为“false”，告知浏览器将目标链接页作一次刷新，清除URL中的AJAX值，从而实现多容器目标页中各容器间的正常切换效果。

此外，要在链接中禁用AJAX请求，还可以将“rel”属性值设置为“external”或增加“target”属性。但在使用时，还是有侧重的，

“rel”和“target”属性主要用于链接的目标页是其他域名下的页面；而“data-ajax”属性主要用于链接的目标页在同一域名下，只是告知浏览器，禁止使用AJAX请求的方式链接。

8.9　使用localStorage传递链接参数

在使用jQuery Mobile开发移动项目过程中，常常需要在“page”容器或页面间传递链接参数，使用传统的URL方式传参对于一个HTML静态页面来说有诸多不便，代码实现相对复杂，兼容性不强。考虑到jQuery Mobile是完全基于HTML 5标准开发的，可以使用HTML 5中的“localStorage”对象来实现链接参数值的传递。更多关于“localStorage”对象的使用方法，可以参阅本人的另外一部图书作品《HTML 5实战》的第8章。

接下来通过一个完整的实例详细介绍如何使用“localStorage”对象来实现传参。

实例8-9　使用localStorage传递链接参数

1.功能说明

新建一个HTML页面，添加两个“page”容器。在第一个容器中单击“传值”链接时，通过“localStorage”对象设置该值，页面切换至第二个容器中，并显示“localStorage”对象保存的值。

2.实现代码

新建一个HTML页面8-9.htm，加入代码如代码清单8-9所示。

代码清单8-9　使用localStorage传递链接参数

```
<! DOCTYPE html>
<html>
<head>
<title>使用localStorage传递链接参数</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/8-9.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content"></div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
<div data-role="page"id="page2">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content"></div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在代码清单8-9中，<head>元素包含了一个“8-9.js”文件。在该文件中绑定容器的“pagecreate”事件，实现调用“localStorage”对象传值的功能，8-9.js文件如下所示：

```
var rttophtml5mobi={
```

```
    author: 'tgrong',
    version: '1.0',
    website: 'http: //www.rttop.cn'
    }
    rttophtml5mobi.install={
    setParam: function（name, value）{
    localStorage.setItem（name, value）
    },
    getParam: function（name）{
    return localStorage.getItem（name）
    }
    }
    $（"#page1"）.live（"pagecreate", function（）{
    var$content=$（this）.fnd（'div[data-role="content"]'）;
    var$strhtml='＜a href="#page2"data-id="50000"＞传值＜/a
＞';
    $content.html（$strhtml）;
    $content.delegate（'a', 'click', function（e）{
    rttophtml5mobi.install.setParam（
    'p_link_id', $（this）.data（'id'））
    }）
    }）
    $（"#page2"）.live（"pagecreate", function（）{
    var$content=$（this）.fnd（'div[data-role="content"]'）;
    var$strhtml='传回的值是: ';
    var$p_link_id=rttophtml5mobi.install
    .getParam（'p_link_id'）;
    $content.html（$strhtml+$p_link_id）;
    }）
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-9所示。

图　8-9　使用localStorage传递链接参数的效果

4.源码分析

在本实例的JavaScript代码中，首先，定义一个“rttophtml5mobi”对象，设置一些基础的内容值。在接下来的“install”对象中定义两个方法，一个为“setParam”，即调用“localStorage”对象中的setItem（）方法设置参数名称和值；另一个为“getParam”，即调用“localStorage”对象中的getItem（）方法获取设置的对应参数值。

然后，在“page1”容器的“pagecreate”事件中，先获取正文区域元素，并设置一个带链接元素的字符串内容，将该内容显示在元素

中；同时，使用delegate（）方法绑定链接元素的单击事件，在该事件中，调用自定义的setParam（）方法设置需要传递的参数值。

最后，在“page2”容器的“pagecreate”事件中，先获取正文区域元素，然后调用自定义的getParam（）方法获取传递来的参数值，并将它显示在元素中。

说明　在设置的链接字符串中，先为元素添加一个“data-id”属性，该属性可以修改为“data-”加任意字母的格式，通过调用data（）方法可以获取该属性的值。

8.10　在jQuery Mobile中构建离线功能

在HTML 5中，调用新增缓存机制，可以在线时将对应文件缓存在本地，离线时调用这些本地文件，从而实现页面或数据在离线后仍可以访问或读取的功能。更多关于离线应用的开发和使用方法，可以参阅本人的另外一部图书作品《HTML 5实战》中第9章。

而在jQuery Mobile开发移动项目时，也能借助HTML 5的离线功能实现应用的离线访问。接下来，通过一个简单离线页面的开发，详细介绍该功能的实现过程。

实例8-10　在jQuery Mobile中构建离线功能

1.功能说明

新建一个HTML页面，在正文区域中增加一个＜p＞元素和＜div＞元素，前者用于显示一段文字内容，后者用于显示网络的当前状态。该页面在网络正常时和在离线时都可以访问。如果是离线访问，那么网络状态将显示“离线”，否则显示“在线”。

2.实现代码

新建一个HTML页面8-10.htm，加入代码如代码清单8-10所示。

代码清单8-10　在jQuery Mobile中构建离线功能

```
<! DOCTYPE html>
<html manifest="cache.manifest">
<head>
<title>在jQuery Mobile中构建离线功能</title>
<meta name="viewport"content="width=device-width,
initial-scale=1"/>
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="Stylesheet"type="text/css"/>
<link href="Css/8-10.css"
rel="Stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
<script src="Js/8-10.js"
type="text/javascript">
</script>
</head>
<body>
<div data-role="page"id="page1">
<div data-role="header"><h1>头部栏</h1></div>
<div data-role="content">
<p>
rttop.cn是一家新型高科技企业，正在努力实现飞翔的梦想。
</p>
<div class="status"></div>
</div>
<div data-role="footer"><h4>©2012 rttop.cn studio</h4
></div>
</div>
</body>
</html>
```

在代码清单8-10中，<head>元素包含一个“8-10.js”文件。在该文件中绑定“page1”容器的“pagecreate”事件，在该事件中根据网络状态显示不同内容，8-10.js文件如下所示：

```
$（"#page1"）.live（"pagecreate", function（）{
if（navigator.onLine）{
$（".status"）.html（"在线"）;
}else{
$（".status"）.html（"离线"）;
}
}）
```

在代码清单8-10中，＜head＞元素包含一个“8-10.css”文件，用于控制显示网络状态内容元素的样式，代码如下所示：

```
.status
{
float: right; font-style: italic;
font-family: 黑体; font-size: 14px;
padding-right: 10px
}
```

此外，在代码清单8-10中，＜html＞元素通过“manifest”属性绑定了一个缓存列表文件cache.manifest，用于列出在线时需要缓存的文件，该文件的代码如下所示：

```
CACHE MANIFEST
#version 0.0.1
NETWORK:
*
CACHE:
Css/jquery.mobile-1.0.1.min.css
Css/8-10.css
Js/8-10.js
Js/jquery-1.6.4.js
Js/jquery.mobile-1.0.1.js
Css/images/ajax-loader.png
Css/images/icons-18-black.png
Css/images/icons-18-white.png
Css/images/icons-36-black.png
```

```
Css/images/icons-36-white.png
Css/images/icon-search-black.png
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图8-10所示。

4.源码分析

在本实例的JavaScript代码中，编写页面“page1”容器的“pagecreate”事件。在该事件中，通过调用“navigator”对象的“onLine”属性判断当前的网络状态，在页面中显示不同的内容值。

图　8-10　在jQuery Mobile中构建离线功能

页面还绑定了一个缓存列表清单文件，首次在线访问该页面时，浏览器将请求返回文件中全部的资源文件，并将新获取的资源文件更新至本地缓存中。当浏览器再次访问该页面时，如果cache.manifest文件没有发生变化，将直接调用本地的缓存响应用户的请求，从而实现浏览页面的功能。

说明　目前，主要手机端浏览器对页面离线功能的支持并不好，仅有少数浏览器支持；相信今后随着各手机浏览厂商的不断升级，应用程序的离线功能支持将会越来越好。

8.11　本章小结

在jQuery框架中，jQuery Mobile针对的是移动终端的应用开发。作为一项全新的技术，虽然对于新手来说，学习成本要远低于其他开发移动设备应用的语言，但在实际的开发过程中，还是会出现诸多很初级的问题。本章列举了初学者在开发过程中容易遇到的常见问题，并通过理论与实例相结合的方式，逐一进行解答。通过本章的学习，开发人员有望在实践中少走弯路，不断提升使用jQuery Mobile开发移动应用的效率。

第9章　开发移动终端新闻订阅管理系统

本章内容

需求分析

数据结构

系统封面开发

系统首页开发

订阅管理页开发

类别新闻页开发

新闻详情页开发

其余文件

本章小结

随着移动互联网的不断发展，人们上网的习惯也在悄然发生变化，由原来的PC端桌面浏览器逐步向移动终端设备过渡，而开发基于移动终端设备的应用系统已成为各互联网企业的共识，也是时下最热

的话题。本章使用jQuery Mobile框架，开发一个移动终端新闻订阅管理系统，实现在移动终端自由订阅各类新闻、动态浏览的各项功能。

9.1　需求分析

在本系统中，需要实现的需求包括以下几个方面：

在进入系统前，先浏览封面页，停留3秒后自动进入首页。

在首页中，显示用户自己订阅的新闻类别，单击某类别时，进入相应的类别页；当用户在首页单击“管理订阅”按钮时，进入订阅管理页。

在类别新闻页中，浏览该类别中的今日图片与列表新闻，单击图片或列表中的某选项时，进入对应的新闻详情页。

在订阅管理页中，以列表的方式，展示用户还没有订阅的新闻类别，当用户单击列表中最右侧的“添加”按钮时，即完成了订阅功能。

在新闻详情页中，显示某条新闻的对应主题、加入时间、来源和正文信息。

9.1.1　总体设计

考虑到移动终端设备中各浏览器的复杂特性和与PC端在机器性能、网络环境的诸多差异，在使用jQuery Mobile开发移动应用项目

时，必须把握下面几个主要方面：

易操作，鉴于移动设备的屏幕特征，必须使开发出来的功能容易操作。

体积小，大部分的移动设备在使用上网服务时，需要根据流量来计费，因此，如果项目在使用时加载的数据过大，将会消耗用户很高的流量。

性能好，必须保证用户在使用移动网络时，数据的交互流畅、安全，因此，不应过多请求服务器的数据，尽可能使用本地或CDN缓存技术实现数据交互。

综合上述各方面的因素，并考虑到整体的实际需求，本系统的总体设计如图9-1所示。

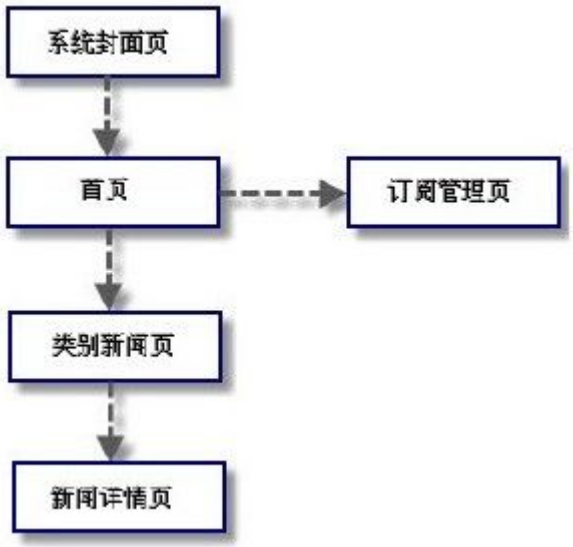


图　9-1　移动终端新闻订阅管理系统总体设计示意图

图9-1中列出了系统的功能和操作流程，本系统有5个功能，分别由系统封面页、首页、订阅管理页、类别新闻页、新闻详情页实现。在操作流程上，先通过系统封面页进入首页，在首页中可以进入类别新闻页和订阅管理页，只有在类别新闻页中，才能进入新闻详情页。

9.1.2　功能设计

本系统针对需求分析，使用jQuery Mobile开发了5项功能，说明如下：

系统封面页：通过使用JavaScript中计时器的功能，在指定的时间内，自动跳转到系统首页中。

首页：在页面中添加一个“page”容器，在绑定的容器“pagecreate”事件中，将API获取的指定数据显示在容器中的列表元素中；并添加一个按钮，单击后进入“订阅管理页”。

订阅管理页：在该页中获取用户已订阅的新闻类别，并与全部类别相比较，将未订阅的新闻类别以列表的形式显示在“page”容器中；单击列表右侧“添加”按钮时，更新用户已订阅的新闻类别信息，并刷新当前容器。

类别新闻页：该页由上下两部分组成，上面展示一幅本类别的专属图片，图片的下面以列表的形式显示本类别下的全部新闻标题；单击新闻标题后，进入新闻详情页。

新闻详情页：在该页中接收“localStorage”对象传回的新闻Id号，根据该Id号调用API获取对应新闻的详细内容，并显示在页面指定

的各元素中。

9.2　数据结构

本系统的数据结构流程是：使用数据库中的3个表保存项目中对应的数据，通过服务器端代码（如.NET、PHP、JSP等）获取数据库中的数据，并以API接口的形式将转换后的JSON格式数据返回给调用的HTML前端页面，页面接收传回的JSON数据，按格式进行组织并显示在页面中。

9.2.1　数据库设计

在名为news_mobile的数据库中新建3个表：imgnews_data、news_cate、news_data，分别用于保存新闻图片、新闻类别和详细的新闻数据，其完整的结构如下。

新闻图片表（imgnews_data）

该表用于存放含有图片的新闻，其结构如表9-1所示。

表 9-1　新闻图片表（imgnews_data）

字段名	描　述	类　型	长　度	是否为空	是否为主键
imgnews_id	记录 Id 号	int	4	否	是，自增项
imgnews_imgurl	图片 URL	nvarchar	50	是	否
news_id	新闻 Id 号	int	4	否	否，是外键

如果以SQL Server 2005为例，那么imgnews_data表在数据库中生成的脚本代码如下：

```
    CREATE TABLE[dbo].[imgnews_data]（
    [imgnews_id][int]IDENTITY（1, 1）NOT NULL,
    [imgnews_imgurl][nvarchar]（50）COLLATE Chinese_PRC_CI_AS
NULL,
    [news_id][int]NOT NULL,
    CONSTRAINT[PK_imgnews_data]PRIMARY KEY CLUSTERED
    （
    [imgnews_id]ASC
    ）WITH（PAD_INDEX=OFF, IGNORE_DUP_KEY=OFF）ON[PRIMARY]
    ）ON[PRIMARY]
```

imgnews_data表在数据库中的示例数据如图9-2所示。

	imgnews_id	imgnews_imgurl	news_id
	1	images/1.jpg	2
	2	images/2.jpg	4
	3	images/3.jpg	6
	4	images/4.jpg	8
	5	images/5.jpg	10

图　9-2　imgnews_data表中的示例数据

新闻类别表（news_cate）

该表用于存放新闻全部的类别，其结构如表9-2所示。

表 9-2　新闻类别（news_cate）

字段名	描　述	类　型	长　度	是否为空	是否为主键
news_cateid	记录 Id 号	int	4	否	是，自增项
news_iconurl	类别图标 URL	nvarchar	50	是	否
news_catename	类别名称	nvarchar	50	否	否
news_catedesc	类别描述	nvarchar	500	是	否

news_cate表在SQL Server 2005数据库中生成的脚本代码如下：

```
    CREATE TABLE[dbo].[news_cate]（
    [news_cateid][int]IDENTITY（1, 1）NOT NULL,
```

```
    [news_iconurl][nvarchar]（50）COLLATE Chinese_PRC_CI_AS
NULL,
    [news_catename][nvarchar]（50）COLLATE Chinese_PRC_CI_AS
NOT NULL,
    [news_catedesc][nvarchar]（500）COLLATE Chinese_PRC_CI_AS
NULL,
    CONSTRAINT[PK_news_cate]PRIMARY KEY CLUSTERED
    （
    [news_cateid]ASC
    ）WITH（PAD_INDEX=OFF, IGNORE_DUP_KEY=OFF）ON[PRIMARY]
    ）ON[PRIMARY]
```

news_cate表在数据库中的示例数据如图9-3所示。

	news_cateid	news_iconurl	news_catename	news_catedesc
	1	images/icon/1.png	头条	关注最新最热的新闻
	2	images/icon/2.png	科技	追踪前沿科技
	3	images/icon/3.png	经济	探讨经济热点
	4	images/icon/4.png	娱乐	浏览最时尚的娱乐资讯
	5	images/icon/5.png	教育	浏览最详近的育人方法

图　9-3　news_cate表中的示例数据

新闻数据表（news_data）

该表用于存放全部的新闻数据信息，其结构如表9-3所示。

表 9-3　新闻数据（news_data）

字段名	描　述	类　型	长　度	是否为空	是否为主键
news_id	记录 Id 号	int	4	否	是，自增项
news_title	新闻主题	nvarchar	50	否	否
news_content	新闻内容	nvarchar	500	否	否
news_source	新闻来源	nvarchar	50	否	否
news_cateid	新闻类别 Id 号	int	4	否	否，是外键
news_adddate	新闻增加时间	datetime	8	否	否

news_data表在SQL Server 2005数据库中生成的脚本代码如下：

```
    CREATE TABLE[dbo].[imgnews_data]（
    [imgnews_id][int]IDENTITY（1, 1）NOT NULL,
    [imgnews_imgurl][nvarchar]（50）COLLATE Chinese_PRC_CI_AS
NULL,
    [news_id][int]NOT NULL,
    CONSTRAINT[PK_imgnews_data]PRIMARY KEY CLUSTERED
    （
    [imgnews_id]ASC
    ）WITH（PAD_INDEX=OFF, IGNORE_DUP_KEY=OFF）ON[PRIMARY]
    ）ON[PRIMARY]
```

news_data表在数据库中的示例数据如图9-4所示。

news_id	news_title	news_content	news_source	news_cateid	news_adddate
1	jQuery Mobil...	jQuery Mobil...	荣拓新闻	1	2012-5-1 0:00:00
2	头条-2	内容-2	荣拓新闻	1	2012-5-2 0:00:00
3	科技-1	科技-1	荣拓新闻	2	2012-5-3 0:00:00
4	科技-2	科技-2	荣拓新闻	2	2012-5-4 0:00:00
5	经济-1	经济-1	荣拓新闻	3	2012-5-5 0:00:00
6	经济-2	经济-2	荣拓新闻	3	2012-5-6 0:00:00
7	娱乐-1	娱乐-1	荣拓新闻	4	2012-5-7 0:00:00
8	娱乐-2	娱乐-2	荣拓新闻	4	2012-5-8 0:00:00
9	教育-1	教育-1	荣拓新闻	5	2012-5-9 0:00:00
10	教育-2	教育-2	荣拓新闻	5	2012-5-10 0:00:00

图　9-4　news_data表中的示例数据

9.2.2　输出API设计

除通过数据库存储数据之外，本系统还提供了数据输出的API接口，用于前端HTML页面的调用；根据系统的功能描述，需要设计4个相对应的API接口，详细说明如下：

1）用于首页和新闻订阅管理页的全部新闻类别输出，URL地址为：

```
http: //localhost/ch9/NewsApi.ashx?act=index
```

其中，“NewsApi.ashx”为.NET中的一般处理程序，读者可以自行选择服务器端语言开发，“act”为操作类型参数，该接口返回的JSON数据如图9-5所示。

2）用于类别列表页中某类别中图片新闻的输出，URL地址为：

```
http: //localhost/ch9/NewsApi.ashx?act=cate_img&cateid=1
```

{"Table":[{"news_cateid":1,"news_catename":"头
条","news_iconurl":"images\/icon\/1.png","news_catedesc":"关注最新最热的新闻"},
{"news_cateid":2,"news_catename":"科
技","news_iconurl":"images\/icon\/2.png","news_catedesc":"追踪前沿科技"},
{"news_cateid":3,"news_catename":"经
济","news_iconurl":"images\/icon\/3.png","news_catedesc":"探访经济热点"},
{"news_cateid":4,"news_catename":"娱
乐","news_iconurl":"images\/icon\/4.png","news_catedesc":"浏览最时尚的娱乐资讯"},
{"news_cateid":5,"news_catename":"教
育","news_iconurl":"images\/icon\/5.png","news_catedesc":"洞察最深远的育人方法"}]}

图　9-5　显示全部新闻类别的JSON格式数据

其中，“act”为操作类型参数，如果该参数的值为“cate_img”，表示类别下的图片新闻；“cateid”为类别Id号参数，用于指定类别的Id值，通过该接口返回的JSON数据如图9-6所示。

图　9-6　显示头条类别下图片新闻的JSON格式数据

3）用于类别列表页中某类别下全部新闻的输出，URL地址为：

```
http: //localhost/ch9/NewsApi.ashx?act=cate_lst&cateid=1
```

其中，“act”为操作类型参数，如果该参数的值为“cate_lst”，表示类别下的全部新闻；“cateid”为类别Id号参数，用于指定类别的Id值，通过该接口返回的JSON数据如图9-7所示。

图　9-7　显示头条类别下全部新闻的JSON格式数据

4）用于某条新闻详细内容数据输出，URL地址为：

```
http: //localhost/ch9/NewsApi.ashx?act=detail&newsid=1
```

其中，“act”为操作类型参数，如果该参数的值为“detail”，表示获取指定Id号的新闻详细数据；“newsid”为新闻Id号参数，通过该接口返回的JSON数据如图9-8所示。

图　9-8　显示某条新闻详细内容的JSON格式数据

说明　在编写API输出JSON格式数据时，为了数据交互的安全，可以对中文字符、关键内容进行加密或转换处理；同时，如需返回大量数据，必须根据页码分段输出，以提升数据交互的效率。

9.3　系统封面开发

本系统既可以通过移动终端的浏览器直接浏览，也能以嵌入页面的方式打包成apk或其他应用程序。如果是后者，那么在进入首页之前，通常有一个系统封面页，用于声明系统版权或推广产品，3～5秒后自动跳转到系统首页。

实例9-1　新闻订阅系统封面开发

1.功能说明

新建一个HTML页面，添加一个“page”容器。在容器中添加一个＜div＞和多个＜p＞元素，用于显示系统封面文字和图标信息，并将容器中的头部栏与尾部栏设置为悬浮状。

2.实现代码

新建一个HTML页面load.htm，加入代码如代码清单9-1所示。

代码清单9-1　新闻订阅系统封面开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>封面页_荣拓移动新闻系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="stylesheet"type="text/css"/>
<link href="Css/rttopHtml5.css"
rel="stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="load_index"data-theme="c">
<div data-role="header"data-position="fixed">
<h4>荣拓新闻</h4>
</div>
<p class="border_p01"></p>
<div class="load">
<p class="t">一个力求报道与还原真实内容的移动端平台</p>
<p><img src="images/rttop_loading.gif"alt=""/></p>
<p class="l">正在加载数据……</p>
</div>
<div data-role="footer"data-position="fixed">
<h1>©2012 rttop.cn studio</h1>
</div>
</div>
<script src="Js/rttopHtml5.base.js"
type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
type="text/javascript"></script>
</body>
</html>
```

在代码清单9-1中，包含两个js文件“rttopHtml5.base.js”和“rttopHtml5.news.js”。其中，rttopHtml5.base.js文件用于设置系统的一些基础属性值和定义设置与获取localStorage对象键名、键值的方法，该文件如代码清单9-2所示。

代码清单9-2　系统封面开发所对应的rttopHtml5.base.js文件

```
var rttophtml5mobi={
author: 'tgrong',
version: '1.0',
website: 'http: //localhost'
}
rttophtml5mobi.utils={
setParam: function（name, value）{
localStorage.setItem（name, value）
},
getParam: function（name）{
return localStorage.getItem（name）
}
}
```

rttopHtml5. news.js文件通过jQuery Mobile框架实现各页面的对应功能，它包含了本系统中全部页面各功能模块实现的JavaScript代码。该文件中，用于实现系统封面的代码如代码清单9-3所示。

代码清单9-3　rttopHtml5.news.js文件中实现系统封面功能对应的代码

```
//封面页面创建事件
function changepage（）{
window.location.href="index.htm";
}
$（'#load_index'）.live（"pagecreate", function（）{
var id=setInterval（"changepage（）", 3000）;
}）
……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图9-9所示。

图　9-9　新闻订阅系统封面页的效果

4.源码分析

在实现本页面功能的JavaScript代码中，为了使页面能在设定的时间内跳转至首页，先创建一个自定义函数changepage（）。在该函数中，通过设置“window”对象的“location.href”路径值，实现当前页面的转向功能；然后，在本页面绑定的“pagecreate”事件中调用setInterval（）方法，在该方法中每隔3秒自动执行自定义函数changepage（），从而实现页面自动跳转的功能。

9.4　系统首页开发

系统封面页停留3秒之后，便进入系统首页，首页中显示用户已订阅的新闻类别列表与总数量。如果用户需要订阅其他类别新闻，可以单击首页中的“订阅管理”按钮，进入订阅管理页进行更多类别的新闻订阅。

实例9-2　新闻订阅系统首页开发

1.功能说明

新建一个HTML页面，在页面中添加一个“page”容器，在容器中添加一个带计数器的＜ul＞列表元素，用于显示用户已订阅的各类别新闻总量和列表；同时，增加一个＜a＞元素的按钮，单击该按钮进入订阅管理页。

2.实现代码

新建一个HTML页面index.htm，加入代码如代码清单9-4所示。

代码清单9-4　新闻订阅系统首页开发

```
＜! DOCTYPE html＞
＜html＞
＜head＞
＜title＞首页_荣拓移动新闻系统＜/title＞
＜meta name="viewport"content="width=device-width,
```

```
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="index_index">
    <div data-role="header"data-position="fixed">
    <h4>荣拓新闻</h4>
    </div>
    <p class="border_p01"></p>
    <ul data-role="listview"data-dividertheme="e"></ul>
    <a href="newsub.htm"data-theme="d"data-mini="true"
    data-role="button"data-icon="plus">订阅管理</a>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.news.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.news.js中，用于实现系统首页的代码如代码清单9-5所示。

代码清单9-5　rttopHtml5.news.js文件中实现系统首页功能对应的代码

```
    ……省略其他与本页面功能不相关代码
    //首页面创建事件
```

```
    $('#index_index').live("pagecreate", function() {
    var$li="";
    var$strSubStr="";
    var intSubNum=0;
    var$webSite=rttophtml5mobi.website;
    var$webUrl=$webSite+'/ch9/NewsApi.ashx?act=index';
    var$listview=$(this).find('ul[data-role="listview"]');
    var$tpl_Index_List=function($p_array, $p_items){
    if(rttophtml5mobi.utils.getParam('user_sub_str')!
=null){
    $strSubStr=rttophtml5mobi.utils.getParam('user_sub_str')
;
    var$arrSubStr=new Array();
    $arrSubStr=$strSubStr.split(", ");
    intSubNum=$arrSubStr.length-1;
    for(var i=0; i<$arrSubStr.length-1; i++){
    $.each($p_items.Table, function(index, item){
    if(item.news_cateid==$arrSubStr[i]){
    $li='<li class="lst"data-icon="false">
    <a href="newscate.htm"data-ajax="false"
    data-catename="'+item.news_catename+'"
    data-id="'+item.news_cateid+'"
    style="margin: 0px; padding: 0px 0px 0px 55px">
    <img src="'+item.news_iconurl+'"alt=""/>
    <h3>'+item.news_catename+'</h3>
    </li>';
    $p_array.push($li);
    }
    })
    }
    }else{
    $li='<li style="text-align: center">
    您还没有订阅任务类型新闻!
    </li>';
    $p_array.push($li);
    }
    }
    var$lst_Index_List=function(){
    $.getJSON($webUrl, {},
    function(response){
    var li_array=[];
    $tpl_Index_List(li_array, response);
    var strTitle='<li data-role="list-divider">
    我的订阅<span class="ui-li-count">'+
    intSubNum+'</span></li>';
```

```
$listview.html（strTitle+li_array.join（''））；
$listview.listview（'refresh'）；
$listview.delegate（'li a', 'click', function（e）{
rttophtml5mobi.utils.setParam（'cate_link_id',
$（this）.data（'id'））
rttophtml5mobi.utils.setParam（'cate_link_name',
$（this）.data（'catename'））
}）
}）
}
$lst_Index_List（）；
}）
```

……省略其他与本页面功能不相关代码

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图9-10所示。

图　9-10　新闻订阅系统首页

4.源码分析

在实现首页功能的代码清单9-5中，将全部需要执行的代码放置在页面“page”容器的“pagecreate”事件中，该事件将在页面创建完成时触发。

在“page”容器的“pagecreate”事件中，需要执行的源码分为三部分：

第一部分：变量的初始化。在该部分初始化变量值（如“$li”、“$strSubStr”等）供后续代码调用。

第二部分：定义一个名为“$tpl_Index_List”的函数型变量。在该函数中定义两个参数“$p_array”和“$p_items”。前者是一个数组型变量，以追加形式保存格式化后的数据字符串；后者为返回数据对象，该对象用来保存通过API返回的数据集。在函数体中，先通过调用自定义方法getParam（）获取键名为“user_sub_str”对应的值，该值为用户已订阅新闻的类别Id号，格式为“1，2，3，……”。如果该值不为null，则先使用“split”方法分割该值的内容，并使用for语法遍历分割后的内容。在遍历过程中，再使用$.each方法遍历返回的全部新闻类别数据集。如果用户已订阅的新闻类别Id号与返回数据集中的Id号相等，则获取该Id号新闻类别的其他信息，以字符串的形式保存在变量“$li”中，通过调用数组的“push”方法，将变量

“$li”的内容追加到参数数组“$p_array”中。如果用户已订阅新闻的类别值为null，那么将一句提示信息的赋值给变量“$li”，并追加到参数数组“$p_array”中。

第三部分：定义一个名为“$lst_Index_List”的函数型变量。在该函数中，先通过$.getJSON方法请求指定的API地址，在该方法的回调函数中接收返回JSON数据集，并保存在对象变量“response”中。另外，再定义一个名为“li_array”的数组变量，将这两个变量作为实参，调用第二部分中的“$tpl_Index_List”函数型变量，使“li_array”数组变量接收函数返回的字符串内容并通过join（）方法处理后，作为＜ul＞列表元素显示的内容；同时刷新＜ul＞列表元素，使它能即时显示已赋值的数据内容。

另外，在列表对象“$listview”中调用“delegate”方法，绑定＜a＞元素的单击事件。在该事件中调用自定义的“setParam”方法，将类别的Id号与名称保存在相应键名的“localStorage”对象中，实现单击传参的功能。详细实现方法见代码清单中加粗部分。

9.5　订阅管理页开发

在系统首页中，单击“订阅管理”按钮时，进入订阅管理页。该页面以列表的形式显示用户还没有订阅的新闻类别信息。用户单击列表选项右侧的  图标订阅该类别新闻。

实例9-3　新闻订阅管理页开发

1.功能说明

新建一个HTML页面，在页面中添加一个“page”容器。在容器中添加＜ul＞列表，在列表选项中放置两个＜a＞元素。第一个＜a＞元素内显示类别图标、名称和简单描述，单击进入某类别新闻页；第二个＜a＞元素内显示订阅图标，单击实现订阅某类别新闻的功能。

2.实现代码

新建一个HTML页面newsub.htm，加入代码如代码清单9-6所示。

代码清单9-6　新闻订阅管理页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>订阅管理页_荣拓移动新闻系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="stylesheet"type="text/css"/>
<link href="Css/rttopHtml5.css"
rel="stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="newsub_index">
<div data-role="header"data-position="fixed">
<h3>订阅管理</h3>
</div>
<p class="border_p01"></p>
<ul data-role="listview"data-dividertheme="e"></ul>
<div data-role="footer"data-position="fixed">
<h1>©2012 rttop.cn studio</h1>
</div>
</div>
<script src="Js/rttopHtml5.base.js"
type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
type="text/javascript"></script>
</body>
</html>
```

在本系统的全局JavaScript文件rttopHtml5.news.js中，用于实现订阅管理页的代码如代码清单9-7所示。

代码清单9-7　rttopHtml5.news.js文件中实现订阅管理页功能对应的代码

```
……省略其他与本页面功能不相关代码
//订阅管理页面创建事件
$（'#newsub_index'）.live（"pagecreate", function（）{
var$li="";
var$strSubStr="";
var$webSite=rttophtml5mobi.website;
```

```
    var$webUrl=$webSite+'/ch9/NewsApi.ashx?act=index';
    var$listview=$（this）.find（'ul[data-role="listview"]'）;
    var$tpl_Sub_List=function（$p_array, $p_items）{
    if（rttophtml5mobi.utils.getParam（'user_sub_str'）!
=null）{
    $strSubStr=rttophtml5mobi.utils.getParam（'user_sub_str'）
;
    $.each（$p_items.Table, function（index, item）{
    if（$strSubStr.indexOf（item.news_cateid）==-1）{
    $li='＜li class="lst"data-icon="false"＞
    ＜a href="newscate.htm"data-ajax="false"
    data-catename="'+item.news_catename+'"
    data-id="'+item.news_cateid+'"
    style="margin: 0px; padding: 0px 0px 0px 55px"＞
    ＜img src="'+item.news_iconurl+'"alt=""/＞
    ＜h3＞'+item.news_catename+'＜/h3＞＜p＞'+
    item.news_catedesc+'＜/p＞＜/a＞
    ＜a data-id="'+item.news_cateid+'"class="a1"
    href="javascript: "＞＜/a＞＜/li＞';
    $p_array.push（$li）;
    }
    }）
    }else{
    $.each（$p_items.Table, function（index, item）{
    $li='＜li class="lst"data-icon="false"＞
    ＜a href="newscate.htm"data-ajax="false"
    data-catename="'+item.news_catename+'"
    data-id="'+item.news_cateid+'"
    style="margin: 0px; padding: 0px 0px 0px 55px"＞
    ＜img src="'+item.news_iconurl+'"alt=""/＞
    ＜h3＞'+item.news_catename+'＜/h3＞＜p＞'+
    item.news_catedesc+'＜/p＞＜/a＞
    ＜a data-id="'+item.news_cateid+'"class="a1"
    href="javascript: "＞＜/a＞＜/li＞';
    $p_array.push（$li）;
    }）
    }
    }
    var$lst_Sub_List=function（）{
    $.getJSON（$webUrl, {},
    function（response）{
    var li_array=[];
    $tpl_Sub_List（li_array, response）;
    var strTitle='＜li data-role="list-divider"＞精品推荐＜/li
＞';
```

```
    $listview.html（strTitle+li_array.join（''））；
    $listview.listview（'refresh'）；
    $listview.delegate（'li a', 'click', function（e）{
    rttophtml5mobi.utils.setParam（'cate_link_id', $
（this）.data（'id'））
    rttophtml5mobi.utils.setParam（'cate_link_name',
    $（this）.data（'catename'））
    }）
    $listview.delegate（'li.a1', 'click', function（e）{
    $strSubStr+=$（this）.data（'id'）+", ";
    rttophtml5mobi.utils.setParam（'user_sub_str',
$strSubStr）；
    window.location.reload（）；
    }）
    }）
    }
    $lst_Sub_List（）；
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图9-11所示。

图　9-11　新闻订阅系统中订阅管理页的效果

4.源码分析

在实现新闻订阅功能的代码清单9-7中，结构与实现系统首页功能基本相同，也是由三部分组成。

第一部分：定义和初始化变量，供后续代码的调用。

第二部分：定义一个名为“$tpl_Sub_List”的函数型变量。在该函数中定义两个形参，一个用于保存返回格式化显示数据的数组，另一个用于存储API请求时返回的数据集。在函数体中，先通过自定义的getParam（）方法获取键名为“user_sub_str”的用户订阅新闻类别“Id”字符信息，如果该值不为null，则在遍历返回的数据集时，使

用“indexOf”方法检测字符信息值中是否存在遍历过程中的新闻类别Id号。如果检测返回的值为“-1”，表示不存在，则通过“$li”变量保存新闻类别的其他数据信息，并调用数组中的push（）方法将“$li”变量内容追加至形参数组“$p_array”中；如果用户订阅新闻类别“Id”字符信息值为空，则直接使用$.each（）方法遍历返回的新闻类别数据集，并将格式化后的字符串追加至形参数组“$p_array”中。

第三部分：定义另外一个函数型变量“$lst_Sub_List”。在该函数中，先使用$.getJSON（）方法请求API返回指定的数据集。在该方法的回调函数中，将数组变量“li_array”和对象变量“response”作为第二部分自定义的函数型变量“$tpl_Sub_List”实参。调用该函数型变量，将获取的数组使用join（）方法处理后，作为＜ul＞列表元素中显示的内容，并刷新该列表组件，使设置好的内容即时显示在页面中。

最后，使用列表元素的“delegate”方法绑定两个单击事件。

1）单击列表中某个选项时，触发的列表单击事件。在该事件中，调用自定义的setParam（）方法，将新闻类别Id号和类别名称保存至自己命名的键名中。当页面切换后，再调用自定义的getParam（）方法获取保存的对应键值，从而实现页面切换时参数的传递。

2）单击列表中最右侧添加图标时，触发图标的单击事件。在该事件中，以逗号分隔的方式保存用户所选择的新闻类别Id号，并调用自定义的setParam（）方法，将该字符串信息保存至键名为“user_sub_str”的“localStorage”对象中。

详细实现方法，见代码清单中加粗部分。

9.6　类别新闻页开发

无论是在系统首页还是订阅管理页，当用户单击新闻类别列表选项时，都将进入类别新闻页。该页面由上下两部分组成，上部分用于显示该类别下的图片新闻，下部分以列表的形式展示该类别下的全部新闻标题，单击该标题时，进入新闻详情页。

实例9-4　类别新闻页开发

1.功能说明

新建一个HTML页面，在添加的“page”容器中创建多个＜div＞元素，用于显示图片新闻的图片、标题和标题背景；另外，添加一个＜ul＞列表，用于显示该类别下的所有新闻标题信息。

2.实现代码

新建一个HTML页面newscate.htm，加入代码如代码清单9-8所示。

代码清单9-8　类别新闻页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>类别新闻页_荣拓移动新闻系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="newscate_index">
    <div data-role="header"data-position="fixed"><h4></h4
></div>
    <p class="border_p01"></p>
    <div id="news_wrap">
    <div id="news_bg"></div>
    <div id="news_info"></div>
    <div id="news_list"></div>
    </div>
    <ul data-role="listview"data-dividertheme="e"></ul>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.news.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.news.js中，用于实现类别新闻页的代码如代码清单9-9所示。

代码清单9-9　rttopHtml5.news.js文件中实现类别新闻页功能对应的代码

```
……省略其他与本页面功能不相关代码
//类别新闻页面创建事件
```

```
    $('#newscate_index').live("pagecreate", function() {
    var$li="";
    var$strId="";
    var$strName="";
    var$webUrl1="";
    var$webUrl2="";
    var$webSite=rttophtml5mobi.website;
    var$catename=$(this).find('[data-role="header"]h4');
    var$listview=$(this).find('ul[data-role="listview"]');
    var$adlist=$("#news_list");
    var$adinfo=$("#news_info");
    var$tpl_Cate_Ad=function($p_array, $p_items){
    $.each($p_items.Table, function(index, item){
    $li='<a href="newsdetail.htm"data-ajax="false"
    data-catename="'+item.news_catename+'"
    data-id="'+item.news_id+'">
    <img src="'+item.imgnews_imgurl+'"alt=""/></a>';
    $adinfo.html(item.news_title);
    $p_array.push($li);
    })
    }
    var$tpl_Cate_List=function($p_array, $p_items){
    $.each($p_items.Table, function(index, item){
    $li='<li class="lst"data-icon="false">
    <a href="newsdetail.htm"data-ajax="false"
    data-catename="'+item.news_catename+'"
    data-id="'+item.news_id+'"
    style="margin: 0px; padding: 0px">
    <h3>'+item.news_title+'</h3></a></li>';
    $p_array.push($li);
    })
    }
    var$lst_Cate_Ad=function(){
    $strId=rttophtml5mobi.utils.getParam('cate_link_id');
    $strName=rttophtml5mobi.utils.getParam('cate_link_name')
;
    $webUrl1=$webSite+'/ch9/NewsApi.ashx?act=cate_img&
cateid='+$strId;
    $.getJSON($webUrl1, {},
    function(response){
    $catename.html($strName);
    var li_array=[];
    $tpl_Cate_Ad(li_array, response);
    $adlist.html(li_array.join(''));
    $adlist.delegate('a', 'click', function(e){
```

```
    rttophtml5mobi.utils.setParam（'p_link_id', $
（this）.data（'id'））；
    rttophtml5mobi.utils.setParam（'cate_link_name',
    $（this）.data（'catename'））；
    }）
    }）
    }
    var$lst_Cate_List=function（）{
    $strId=rttophtml5mobi.utils.getParam（'cate_link_id'）；
    $strName=rttophtml5mobi.utils.getParam（'cate_link_name'）
；
    $webUrl2=$webSite+'/ch9/NewsApi.ashx?act=cate_lst&
cateid='+$strId;
    $.getJSON（$webUrl2，{}，
    function（response）{
    var li_array=[];
    $tpl_Cate_List（li_array, response）；
    $listview.html（li_array.join（''））；
    $listview.listview（'refresh'）；
    $listview.delegate（'li a', 'click', function（e）{
    rttophtml5mobi.utils.setParam（'p_link_id', $
（this）.data（'id'））
    rttophtml5mobi.utils.setParam（'cate_link_name',
    $（this）.data（'catename'））
    }）
    }）
    }
    $lst_Cate_Ad（）；
    $lst_Cate_List（）；
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图9-12所示。

图　9-12　单击“头条”类别新闻的效果

4.源码分析

在实现浏览类别新闻功能的代码清单9-9中，结构与实现与前几个页面基本相同，只是在处理细节时略有差别，同样也是由三部分组成。

第一部分：定义和初始化变量，供后续代码使用时调用。

第二部分：由于需要展示图片与普通新闻两部分数据，需要分开编写两个函数型变量“$tpl_Cate_Ad”和“$tpl_Cate_List”。前者用于遍历API返回的图片新闻数据集，并将格式化后的字符串追加至形

参数组中；后者用于遍历API返回的普通新闻数据集，将格式化后的字符串追加至形参数组中。

第三部分：定义另外两个函数型变量“$lst_Cate_Ad”和“$lst_Cate_List”，分别用于调用第二部分中自定义的“$tpl_Cate_Ad”和“$tpl_Cate_List”函数型变量。在调用前，先通过getParam（）方法接收页面传递的类别Id号和名称参数值，分别保存在变量“$strId”和“$strName”中，并根据变量“$strId”的值，组成$.getJSON（）方法请求的URL地址，实现根据类别Id号动态获取对应数据集的功能；同时，在$.getJSON（）方法的回调函数中，定义一个数组“li_array”和接收返回数据集变量“response”，将这两个变量作为调用“$tpl_Cate_Ad”和“$tpl_Cate_List”函数型变量的实参；最后，将函数调用返回的数组使用join（）方法处理后，显示在页面相应的元素中。如果是＜ul＞列表，则再次执行刷新操作后，被赋值的内容便可即时显示在＜ul＞列表中。

最后，使用“delegate”方法设置图片新闻和列表选项元素的单击事件。在该事件中，分别使用自定义的“setParam”方法，将新闻的Id号和类别名称保存在对应键名的“localStorage”对象中，在切换页面时调用。更多详细实现方法，见代码清单中加粗部分。

9.7　新闻详情页开发

在类别新闻页中，单击新闻的图片或标题，都将进入新闻详情页。该页面中展示某条新闻的标题、添加时间、来源和新闻正文信息。

实例9-5　新闻详情页开发

1.功能说明

新建一个HTML页面，在添加的“page”容器中增加一个＜div＞元素。在该元素中，添加一个＜h＞和两个＜p＞元素，分别用于显示新闻的标题、添加时间、来源和正文信息。

2.实现代码

新建一个HTML页面newsdetail.htm，加入代码如代码清单9-10所示。

代码清单9-10　新闻详情页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>新闻详情页_荣拓移动新闻系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="detail_index">
    <div data-role="header"data-position="fixed"><h4></h4
></</div>
    <p class="border_p01"></p>
    <div class="detail">
    <h4 id="news_detail_title"></h4>
    <p id="news_detail_info"class="news_detail_info"></p>
    <p id="news_detail_content"
    class="news_detail_content"></p>
    </div>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.news.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.news.js中，用于实现新闻详情页的代码如代码清单9-11所示。

代码清单9-11　rttopHtml5.news.js文件中实现新闻详情页功能对应的代码

```
    ……省略其他与本页面功能不相关代码
    //新闻详情页面创建事件
```

```
    $('#detail_index').live("pagecreate", function() {
    var$strId="";
    var$strName="";
    var$webSite=rttophtml5mobi.website;
    var$webUrl="";
    var$catename=$(this).fnd('[data-role="header"]h4');
    var$title=$("#news_detail_title");
    var$info=$("#news_detail_info");
    var$content=$("#news_detail_content");
    var$lst_Detail_List=function() {
    $strId=rttophtml5mobi.utils.getParam('p_link_id');
    $strName=rttophtml5mobi.utils.getParam('cate_link_name')
;
    $webUrl=$webSite+'/ch9/NewsApi.ashx?act=detail&
newsid='+$strId;
    $.getJSON($webUrl, {},
    function(response) {
    $catename.html($strName);
    $.each(response.Table, function(index, item) {
    $title.html(item.news_title);
    var strHTML=item.news_adddate+"
    来源："+item.news_source;
    $info.html(strHTML);
    $content.html("    "+item.news_content);
    });
    })
    }
    $lst_Detail_List();
    })
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图9-13所示。

图　9-13　进入新闻详情页的效果

4.源码分析

实现浏览新闻详情页功能的代码清单9-11相对简单，首先定义和初始化变量，供后续代码的调用。然后，定义一个函数型变量“$lst_Detail_List”，在函数中，先通过自定义的getParam（）方法获取传回的新闻Id号和类别名称；然后，根据该Id号使用$.getJSON（）方法请求相应的JSON数据；最后，在该方法的回调函数中遍历返回的数据集，并将对应的数字段值显示在页面指定的元素中，从而实现浏览新闻详情页的功能。更多详细实现方法，见代码清单中加粗部分。

9.8　其余文件

在开发本系统的功能时，除上述HTML页面和JavaScript文件外，还有两个全局性文件，一个是样式文件rttopHtml5.css，另一个是API接口调用文件NewsApi.ashx，接下来分别对该两个文件进行详细说明。

9.8.1　样式文件

本系统只有一个样式文件rttopHtml5.css，用于控制整个系统的页面样式与结构布局，详细代码如代码清单9-12所示。

代码清单9-12　rttopHtml5.css文件全部代码

```
/*系统增封面*/
.load
{
text-align: center; padding: 10px;
line-height: 1.8em
}
.load.t
{
padding: 5px 20px 5px 20px; font-family: 黑体;
color: #e63f38
}
.load.l
{
color: #666; font-size: 12px
}
.border_p01
{
```

```
width: 100%; height: 3px; background: #e41400;
margin: 0; padding: 0
}
/*列表区域*/
.lst
{
height: 100%; margin: 0px;
padding: 0px 5px 0px 5px
}
.lst a img
{
padding: 0px; max-height: 50px; max-width: 50px;
padding-top: 5px; padding-bottom: 5px; margin: 0px
}
.lst a h3
{
width: 80%
}
/*新闻详情页*/
.detail
{
text-align: center; padding: 8px
}
.detail.news_detail_info
{
font-size: 12px; color: #666;
padding-bottom: 5px; border-bottom: solid 1px#ccc
}
.detail.news_detail_content
{
text-align: left
}
/*新闻推荐图片*/
#news_wrap
{
position: relative; width: 100%; height: auto;
min-height: 160px; overflow: hidden;
}
#news_list img
{
border: 0px; width: 100%; height: auto
}
#news_bg
{
position: absolute; width: 100%; min-height: 30px;
```

```
    line-height: 30px; bottom: 0; background-color: #000;
    filter: Alpha（Opacity=40）; opacity: 0.4; z-index: 1;
    cursor: pointer;
    }
    #news_info
    {
    position: absolute; min-height: 30px; line-height: 30px;
    bottom: 0; left: 3px; font-size: 12px;
    color: #fff; z-index: 1; cursor: pointer
    }
    /*列表右侧按钮*/
    .ui-li-link-alt
    {
    position: absolute; width: 52px; height: 100%;
    border-width: 0; background: url（images/add.png）no-
repeat;
    background-position: center; border-left-width: 1px;
    top: 0; right: 0; margin: 0; padding: 0; z-index: 2;
    }
    .ui-li-link-alt.ui-btn
    {
    display: none; overfow: hidden; position: absolute;
    right: 8px; top: 50%; margin: -11px 0 0 0;
    border-bottom-width: 1px; z-index: -1;
    }
```

上述代码中，有3个样式类别需要说明。

1）为了能在列表的＜li＞选项元素中自定义最右侧的单击按钮图标，首先在＜li＞选项元素中添加两个＜a＞元素，并重置“ui-li-link-alt”类别下的“ui-btn”子类别，将“display”属性值设置为“none”，表示隐藏原有按钮元素。

2）重置“ui-li-link-alt”类别。在该类别中，以背景的方式添加一个自定义的图标，作为单击按钮的图标。

3）由于隐藏了最右侧的原有按钮，且标题的长度默认为“100%”，该值将会使标题的内容覆盖最右侧的自定按钮图标区域。为了规避这一现象，将＜h3＞标题的长度修改为“80%”可以形成两列独自显示的页面效果。

9.8.2 API接口文件

在本系统的JavaScript代码中，调用$.getJSON（）方法访问接口文件NewsApi.ashx，获取指定的JSON格式数据。该文件是.NET服务器端语言开发的一般程序处理文件，主要功能是：根据传回的参数获取数据库中的对应数据，并转成JSON格式数据集传递给调用的页面。介于篇幅，该文件的详细代码不在本书中列出，感兴趣的读者可以在本书的源码文件（……/slnMobile/Ch9/）中获取。

9.9 本章小结

本章详细介绍了使用jQuery Mobile开发一个完整的移动终端新闻订阅管理系统的过程，这一过程中，既包括需求分析、设计数据库、功能开发等框架搭建的方法，也涵盖了如何使用JSON格式传递数据集、localStorage对象传递参数的技巧。通过本章的学习，读者能够全面了解并掌握使用jQuery Mobile开发移动项目时框架搭建与开发的技巧。

第10章　开发移动终端记事本管理系统

本章内容

需求分析

新手导航页开发

系统首页开发

记事列表页开发

记事详细页开发

修改记事内容页开发

添加记事内容页开发

样式文件

本章小结

第9章通过一个完整的新闻订阅管理系统的开发，详细介绍了在jQuery Mobile中，通过调用服务器端API的方式获取并组织JSON格式数据的过程。在移动终端浏览该系统页面时，可能需要即时通过网络

与服务器端进行数据交互，而移动终端的网络环境受流量和通信信号覆盖的限制，相比PC端而言要复杂得多，因此，系统页面最终执行效率会受影响。

在当前移动终端网络环境不太流畅的情况下，针对处理少量数据交互的系统开发，我们可以借助HTML 5中新增的localStorage对象对数据进行存储与管理，这样无须即时通过网络与服务器端发生数据的交互，将极大地提高页面操作响应的速度和用户最终体验。

本章将通过一个完整的记事本管理系统的开发，由浅入深地介绍在jQuery Mobile中使用localStorage对象开发移动项目的方法与技巧。

10.1　需求分析

在记事本管理系统中，主要包括如下几个需求：

在新手导航页中，以左右滑动图片的方式显示系统截图，当用户滑至最后一幅截图时，自动进入首页。

进入首页后，以列表的形式展示各类别记事数据的总量信息，单击某类别选项进入该类别的记事列表页。

在某类别下的记事列表页中展示该类别下的全部记事主题内容，并增加根据记事主题进行搜索的功能。

如果单击类别下的某记事主题，则进入记事信息详细页，在该页面中展示记事信息的主题和正文信息；另外，添加一个删除该条记事信息的按钮。

如果在记事信息的详细页中单击“编辑”按钮，则进入记事信息编辑页，在该页中可以编辑主题和正文信息。

无论是在首页或记事列表页，单击“新增”按钮，则进入记事信息增加页，在该页中可以增加一条新的记事信息。

10.1.1　总体设计

本系统的功能是：方便、快捷地记录和管理用户的记事数据。在总体设计时，重点把握操作简洁、流程简单、系统可拓展性强的原则。综上考虑，本系统的总体设计如图10-1所示。

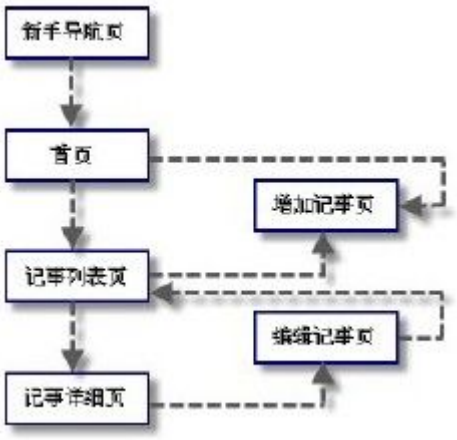


图　10-1　移动终端记事本管理系统总体设计示意图

图10-1中列出了系统的功能和操作流程。本系统有6个功能：新手导航页、分类列表页、记事列表页、记事详细页、编辑记事页和增加记事页。

在操作流程上，首次进入时，先通过新手导航页进入首页；再次进入时，则直接进入首页。在分类和记事列表页中都可以进入增加记事页，只有在记事列表页中，才能进入记事详细页；在记事详细页中，才能进入编辑记事页。另外，无论是完成了增加还是编辑记事的操作，都返回相应类别的记事列表页。

10.1.2　功能设计

针对上述的需求，本系统使用jQuery Mobile开发了6项功能页，具体说明如下。

新手导航页：在“page”容器中添加多张功能说明截图，当用户滑动图片时，触发绑定的滑动事件，驱使图片向左或向右滑动；滑到最后一幅图片时，通过“localStorage”对象保存查看状态。当用户再次进入系统时，该状态值不为空，则进入系统首页，否则进入新手导航页。

首页：遍历“localStorage”对象保存的记事本数据，在遍历过程中，以累加方式记录各类别下记事数据的总量，并通过＜ul＞列表显示类别名称和对应记事数据总量；另外，当单击＜ul＞列表中某选项时，则进入某类别下的记事列表页。

记事列表页：根据“localStorage”对象传回的记事类别名称，获取该类别名称下的记事本数据，并通过列表的方式将记事主题信息显示在页面中。同时，将列表元素的“data-filter”属性值设置为“true”，使该列表具有根据记事主题信息进行搜索的功能；当单击＜ul＞列表中某选项时，则进入该主题的记事详细页。

记事详细页：在该页面中，根据“localStorage”对象传回的记事Id号，获取对应的记事数据，并将记录的主题与内容显示在页面中。同时，在该页面中。当单击头部栏左侧“编辑”按钮时，进入记事编辑页；单击头部栏右侧“删除”按钮时，弹出询问对话框，单击“确定”后，删除该条记事数据。

编辑记事页：在该页面中，以文本框的方式显示某条记事数据的类别、主题和内容，用户可以对这三项内容进行修改；修改后，单击头部栏右侧“更新”按钮，便完成了该条记事数据的更新。

增加记事页：在分类列表页或记事列表页中，当单击头部栏右侧“增加”按钮时，便进入增加记事页。在该页面中，用户可以选择记事的类别，输入记事主题、内容后，单击该页面中的头部栏右侧“保存”按钮，便完成了一条新记事数据的增加。

10.2　新手导航页开发

jQuery Mobile开发的移动项目，既可以在移动设备的浏览器中查看，也可以将页面打包到应用程序中，如apk文件。如果是后者，那么，新手导航页将是系统在运行前需要浏览的页面，在该页面中，以滑动图片的方式，引导用户使用本系统的重点功能。

实例10-1　新手导航页开发

1.功能说明

新建一个HTML页面，在正文“content”容器中添加两个＜div＞元素和一个＜ul＞列表元素。前者用于放置系统的功能图片，后者＜ul＞列表元素放置在图片下面，用于滑动图片时以圆点的方式显示选中或未选图片的状态。当左右滑动图片时，图片下方的小圆点也将动态变换。

2.实现代码

新建一个HTML页面notenav.htm，加入代码如代码清单10-1所示。

代码清单10-1　新手导航页开发

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>新手导航_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    <style type="text/css">
    .ui-page{background: #414141}
    </style>
    </head>
    <body>
    <div data-role="page"id="notenav_index">
    <div data-role="header"><h4>新手导航</h4></div>
    <div data-role="content">
    <div id="notenav_wrap">
    <div id="notenav_list">
    <a href="javascript: ">
    <img src="Images/nav1.jpg"alt=""/></a>
    <a href="javascript: ">
    <img src="Images/nav2.jpg"alt=""/></a>
    </div>
    <ul id="notenav_icon"><li></li><li></li></ul>
    </div>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.note.js"
    type="text/javascript"></script>
    </body>
    </html>
```

代码清单10-1中包含了两个js文件“rttopHtml5.base.js”和“rttopHtml5.note.js”。

rttopHtml5. base.js文件用于设置系统的一些基础属性值，定义设置和获取localStorage对象键名、键值的方法，该文件如代码清单10-2所示。

代码清单10-2　新手导航页面开发所对应的rttopHtml5.base.js文件

```
var rttophtml5mobi={
author: 'tgrong',
version: '1.0',
website: 'http: //localhost'
}
rttophtml5mobi.utils={
setParam: function（name, value）{
localStorage.setItem（name, value）
},
getParam: function（name）{
return localStorage.getItem（name）
}
}
```

rttopHtml5. note.js文件通过jQuery Mobile框架实现各页面的对应功能，它包含了本系统中全部页面各功能模块实现的JavaScript代码。在该文件中，用于实现新手导航页的代码如代码清单10-3所示。

代码清单10-3　rttopHtml5.note.js文件中实现新手导航功能对应的代码

```
……省略其他与本页面功能不相关代码
```

```
    //新手导航页面创建事件
    $（"#notenav_index"）.live（"pagecreate", function（）{
    if（rttophtml5mobi.utils.getParam（'bln_look'）！=null）{
    $.mobile.changePage（"index.htm", "slideup"）；
    }else{
    var$count=$（"#notenav_list a"）.length;
    $（"#notenav_list a: not（: frst-child）"）.hide（）；
    $（"#notenav_icon li: frst-
child"）.addClass（'on'）.html（"1"）；
    $（"#notenav_list a img"）.each（function（index）{
    $（this）.swipeleft（function（）{
    if（index＜$count-1）{
    var i=index+1;
    var s=i+1;
    $（"#notenav_list a"）.flter（": visible"）
    .fadeOut（500）.parent（）.children（）
    .eq（i）.fadeIn（1000）；
    $（"#notenav_icon li"）.eq（i）.html（s）；
    $（"#notenav_icon li"）.eq（i）.toggleClass（"on"）；
    $（"#notenav_icon li"）.eq（i）.siblings（）
    .removeAttr（"class"）.html（""）；
    if（s==$count）{
    rttophtml5mobi.utils
    .setParam（'bln_look', 1）；
    $.mobile.changePage（"index.htm", "slideup"）；
    }
    }
    }）.swiperight（function（）{
    if（index＞0）{
    var i=index-1;
    var s=i+1;
    $（"#notenav_list a"）.flter（": visible"）
    .fadeOut（500）.parent（）.children（）
    .eq（i）.fadeIn（1000）；
    $（"#notenav_icon li"）.eq（i）.html（s）；
    $（"#notenav_icon li"）.eq（i）.toggleClass（"on"）；
    $（"#notenav_icon li"）.eq（i）.siblings（）
    .removeAttr（"class"）.html（""）；
    }
    }）
    }）
    }
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-2所示。

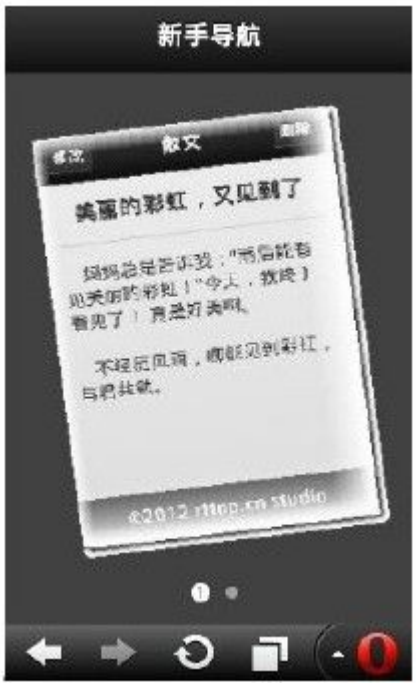


图　10-2　显示新手导航页时的效果

4.源码分析

在实现新手导航页功能的代码清单10-3中，首先检测键名为“bln_look”的localStorage对象值是否为null。如果不为null，表示不是首次进入本系统，则调用“$.mobile”对象提供的changePage（）方法，直接跳转至首页；如果为空，表示是首次进入本系统，那么首先获取＜div＞元素中图片链接元素的总数量，并保存

到变量“$count”中，然后使用hide（）方法隐藏除第一个图片链接元素之外的其他元素，并使用addClass（）方法增加＜ul＞列表元素中图片被选中时对应选项的样式和初始值。

接下来，遍历＜div＞元素中的全部图片，并在遍历过程中，通过“$（this）”方式获取每个图片元素，绑定该元素的“swipeleft”和“swiperight”事件；在图片元素的“swipeleft”事件中，先判断当前元素的索引号“index”值是否小于图片总量，如果成立，那么当前索引号自动增加“1”，使图片的索引号指向下一张，并通过fadeout（）和fadeIn（）方法实现当前图片的隐藏和下一张图片的显示；同时，在显示下一张图片时，调用toggleClass（）和html（）方法切换该图片选中时的样式和数字内容；与此同时，使用removeAttr（）和html（）方法移除其他未选中图片的原有样式和数字内容。

图片元素的“swiperight”事件中执行的代码与“swipeleft”事件基本相同，区别在于，在图片元素的“swiperight”事件中，先判断当前元素的索引号“index”值是否大于0，如果成立，那么当前索引号自动减少“1”，使图片的索引号指向上一张，其余代码与“swipeleft”事件相同，在此不再赘述，更多详细实现方法，见代码清单中加粗部分。

说明　在移动设备浏览器中，向左滑动图片表示浏览下一张图片，触发“swipeleft”事件；向右滑动图片表示浏览上一张图片，触发“swiperight”事件。

10.3　系统首页开发

当在新手导航页中浏览完全部的导航图片或非首次进入记事本系统时，都将进入系统首页面，在该页面中，通过＜ul＞列表显示记事数据的全部类别名称，并将各类别记事数据的总量，显示在列表中对应类别的右侧。

实例10-2　记事本管理系统首页开发

1.功能说明

新建一个HTML页面，在页面“page”容器中添加一个＜ul＞列表元素，在列表中显示记事数据的分类名称与类别总量，单击该列表选项进入记事列表页。

2.实现代码

新建一个HTML页面index.htm，加入代码如代码清单10-4所示。

代码清单10-4　记事本管理系统首页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>首页_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
```

```
<link href="Css/jquery.mobile-1.0.1.min.css"
rel="stylesheet"type="text/css"/>
<link href="Css/rttopHtml5.css"
rel="stylesheet"type="text/css"/>
<script src="Js/jquery-1.6.4.js"
type="text/javascript"></script>
<script src="Js/jquery.mobile-1.0.1.js"
type="text/javascript"></script>
</head>
<body>
<div data-role="page"id="index_index">
<div data-role="header"data-position="fixed"
data-position="inline">
<h4>荣拓记事</h4>
<a href="addnote.htm"class="ui-btn-right">新增</a>
</div>
<div data-role="content">
<ul data-role="listview"></ul>
</div>
<div data-role="footer"data-position="fixed">
<h1>©2012 rttop.cn studio</h1>
</div>
</div>
<script src="Js/rttopHtml5.base.js"
type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
type="text/javascript"></script>
</body>
</html>
```

在本系统的全局JavaScript文件rttopHtml5.note.js中，用于实现系统首页的代码如代码清单10-5所示。

代码清单10-5　rttopHtml5.note.js文件中系统首页功能对应的代码

```
……省略其他与本页面功能不相关代码
//首页页面创建事件
$（"#index_index"）.live（"pagecreate", function（）{
```

```
    var$listview=$（this）.fnd（'ul[data-role="listview"]'）;
    var$strKey="";
    var$m=0, $n=0;
    var$strHTML="";
    for（var intI=0; intI＜localStorage.length; intI++）{
    $strKey=localStorage.key（intI）;
    if（$strKey.substring（0, 4）=="note"）{
    var
getData=JSON.parse（rttophtml5mobi.utils.getParam（$strKey））
;
    if（getData.type=="a"）{
    $m++;
    }
    if（getData.type=="b"）{
    $n++;
    }
    }
    }
    var$sum=parseInt（$m）+parseInt（$n）;
    $strHTML+='＜li data-role="list-divider"＞全部记事本内容
    ＜span class="ui-li-count"＞'+$sum+'＜/span＞＜/li＞';
    $strHTML+='＜li＞＜a href="list.htm"data-ajax="false"
    data-id="a"data-name="散文"＞散文
    ＜span class="ui-li-count"＞'+$m+'＜/span＞＜/li＞';
    $strHTML+='＜li＞＜a href="list.htm"data-ajax="false"
    data-id="b"data-name="随笔"＞随笔
    ＜span class="ui-li-count"＞'+$n+'＜/span＞＜/li＞';
    $listview.html（$strHTML）;
    $listview.delegate（'li a', 'click', function（e）{
    rttophtml5mobi.utils.setParam（'link_type', $
（this）.data（'id'））
    rttophtml5mobi.utils.setParam（'type_name', $
（this）.data（'name'））
    }）
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-3所示。

图　10-3　记事本管理系统首页

4.源码分析

在实现系统首页功能的代码清单10-5中，首先定义一些数值和元素变量，供后续代码的使用。然后，由于全部的记事数据都保存在“localStorage”对象中，要遍历全部的“localStorage”对象，根据键值中前4个字符为“note”的标准，筛选对象中保存的记事数据，并通过JSON.parse（）方法，将该数据字符内容转换成JSON格式对象，再根据该对象的类型值，将不同类型的记事数量进行累加，分别保存在变量“$m”和“$m”中。

最后，组织显示在页面列表＜ul＞元素的内容，并保存在变量“$strHTML”中，调用列表＜ul＞元素的html（）方法，将内容赋值

于页面列表＜ul＞元素中；同时，使用delegate（）方法设置列表选项触发单击事件时需要执行的代码。更多详细实现方法，见代码清单中加粗部分。

说明　本系统的数据全部保存在用户本地的“localStorage”对象中，读取数据的速度很快，当将字符串内容赋值给列表＜ul＞元素时，已完成样式加载，无需再调用refresh（）方法。

10.4　记事列表页开发

用户在首页单击列表中某类别选项时，将类别名称写入“localStorage”对象的对应键值中，当从首页切换至记事列表页时，再将这个已保存的类别键值与整个“localStorage”对象保存的数据进行匹配，获取该类别键值对应的记事数据，并通过＜ul＞列表将数据内容显示在页面中。

实例10-3　记事列表页开发

1.功能说明

新建一个HTML页面，并在页面“page”容器中添加一个＜ul＞列表元素，用于显示某类别下的记事数据；同时，将＜ul＞列表元素的“data-filter”的属性值设置为“true”，使该列表可以根据记事主题进行搜索。

2.实现代码

新建一个HTML页面list.htm，加入代码如代码清单10-6所示。

代码清单10-6　记事列表页开发

```
＜! DOCTYPE html＞
＜html＞
```

```
    <head>
    <title>记事列表页_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="list_index">
    <div data-role="header"data-position="fixed"
    data-position="inline">
    <a href="index.htm">返回</a>
    <h4>记事列表</h4>
    <a href="addnote.htm">新增</a>
    </div>
    <div data-role="content">
    <ul data-role="listview"data-flter="true"></ul>
    </div>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.note.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.note.js中，用于实现记事列表页的代码如代码清单10-7所示。

代码清单10-7　rttopHtml5.note.js文件中记事列表页功能对应的代码

```
    ……省略其他与本页面功能不相关代码
    //记事列表页面创建事件
    $（"#list_index"）.live（"pagecreate", function（）{
    var$listview=$（this）.fnd（'ul[data-role="listview"]'）;
    var$strKey="", $strHTML="", $intSum=0;
    var$strType=rttophtml5mobi.utils.getParam（'link_type'）;
    var$strName=rttophtml5mobi.utils.getParam（'type_name'）;
    for（var intI=0; intI＜localStorage.length; intI++）{
    $strKey=localStorage.key（intI）;
    if（$strKey.substring（0, 4）=="note"）{
    var
getData=JSON.parse（rttophtml5mobi.utils.getParam（$strKey））
;
    if（getData.type==$strType）{
    $strHTML+='＜li data-icon="false"
    data-ajax="false"＞
    ＜a href="notedetail.htm"data-id="'
    +getData.nid+'"＞'+getData.title
    +'＜/a＞＜/li＞';
    $intSum++;
    }
    }
    }
    var strTitle='＜li data-role="list-divider"＞'
    +$strName+'＜span class="ui-li-count"＞'
    +$intSum+'＜/span＞＜/li＞';
    $listview.html（strTitle+$strHTML）;
    $listview.delegate（'li a', 'click', function（e）{
    rttophtml5mobi.utils.setParam（'list_link_id', $
（this）.data（'id'））
    }）
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-4所示。

图　10-4　记事列表默认页和搜索时的效果

4.源码分析

在实现记事列表页功能的代码清单10-7中，首先定义一些字符和元素对象变量，并通过自定义的函数方法getParam（）获取传递的类别字符和名称，分别保存在变量“$strType”和“$strName”中；然后遍历整个“localStorage”对象筛选记事数据。在遍历过程中，将记事的字符数据转换成JSON对象，再根据对象的类别与保存的类别变量相比较，如果符合，则将该条记事的Id号和主题信息追加到字符串

变量“$strHTML”中，并通过变量“$intSum”累加该类别下的记事数据总量。

最后，将获取的数字变量“$intSum”放入列表＜ul＞元素的分割项中，并将保存分割项内容的字符变量“strTitle”和保存列表项内容的字符变量“$strHTML”组合，通过元素的html（）方法将组合后的内容赋值给列表元素；同时，使用delegate（）方法设置列表选项被点击时执行的代码。更多详细实现方法，见代码清单中加粗部分。

10.5　记事详细页开发

当用户在记事列表页中单击某记事主题选项时，将该记事主题的Id号通过“key/value”的方式保存在“localStorage”对象中。

当进入记事详细页时，先调出保存的键值作为传回的记事数据Id号，并将该Id号作为键名获取对应的键值，然后将获取的键值字符串数据转成JSON对象，再将该对象的记事主题和内容显示在页面指定的元素中。

实例10-4　记事详细页开发

1.功能说明

新建一个HTML页面，在“page”容器的正文区域中添加一个＜h3＞和两个＜p＞元素，分别用于显示记事信息的主题和内容；单击头部栏左侧的“修改”按钮进入记事编辑页；单击头部栏右侧的“删除”按钮，可以删除当前的记事数据。

2.实现代码

新建一个HTML页面notedetail.htm，加入代码如代码清单10-8所示。

代码清单10-8　记事详细页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>记事详细页_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="notedetail_index">
    <div data-role="header"data-position="fixed"
    data-position="inline">
    <a href="editnote.htm"data-ajax="false">修改</a>
    <h4></h4>
    <a href="javascript: "id="alink_delete">删除</a></div>
    <div data-role="content">
    <h3 id="title"></h3>
    <p class="notep"></p>
    <p id="content"></p>
    </div>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.note.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.note.js中，用于实现记事详细页的代码如代码清单10-9所示。

代码清单10-9　rttopHtml5.note.js文件中记事详细页功能对应的代码

```
    ……省略其他与本页面功能不相关代码
    //记事详细页面创建事件
    $（"#notedetail_index"）.live（"pagecreate", function（）{
    var$type=$（this）.fnd（'div[data-role="header"]h4'）;
    var$strId=rttophtml5mobi.utils.getParam（'list_link_id'）
;
    var$titile=$（"#title"）;
    var$content=$（"#content"）;
    var
listData=JSON.parse（rttophtml5mobi.utils.getParam（$strId））
;
    var strType=listData.type=="a"?"散文": "随笔";
    $type.html（strType）;
    $titile.html（listData.title）;
    $content.html（listData.content）;
    $（this）.delegate（'#alink_delete', 'click', function（e）
{
    var yn=confrm（"您真的要删除吗？"）;
    if（yn）{
    localStorage.removeItem（$strId）;
    window.location.href="list.htm";
    }
    }）
    }）
    ……省略其他与本页面功能不相关代码
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-5所示。

图　10-5　记事详细页和删除数据时的效果

4.源码分析

实现记事详细页功能的代码清单10-9相对简单，首先定义一些元素对象变量，并通过自定义的函数方法getParam（）获取传递的某记事Id号，并保存在变量“$strId”中；然后将该变量作为键名，获取对应的键值字符串，并将键值字符串调用JSON.parse（）方法转换成JSON对象，在该对象中依次获取记事的主题和内容，显示在页面中的指定元素中。

另外，通过delegate（）方法添加单击“删除”按钮触发“单击”事件时执行的代码。在该事件中，先通过变量“yn”保存confirm（）函数返回的true或false值，如果为真，那么根据记事数据的键名值使用removeItem（）方法，删除指定键名的全部对应键

值，实现删除记事数据的功能；同时，页面返回记事列表页。更多详细实现方法，见代码清单中加粗部分。

10.6　修改记事内容页开发

当在记事详细页中单击头部栏左侧的“修改”按钮时，进入修改记事内容页，在该页面中，可以修改某条记事数据的类别、主题和内容信息，修改完成后返回记事详细页。

实例10-5　修改记事内容页开发

1.功能说明

新建一个HTML页面，在“page”容器的正文区域中，通过水平式的单选按钮组显示记事数据的所属类别，一个文本框和一个文本区域框显示记事数据的主题和内容，用户可以重新选择所属类别、编辑主题及内容数据。单击“更新”按钮，则完成数据的修改操作，并返回记事类别页。

2.实现代码

新建一个HTML页面editnote.htm，加入代码如代码清单10-10所示。

代码清单10-10　修改记事内容页开发

```
<! DOCTYPE html>
<html>
```

```
    <head>
    <title>修改记事_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="editnote_index">
    <div data-role="header"data-position="fixed"
    data-position="inline">
    <a href="notedetail.htm"data-ajax="false">返回</a>
    <h4>编辑记事</h4>
    <a href="javascript: ">更新</a>
    </div>
    <div data-role="content">
    <label for="rdo-type">类型：</label>
    <feldset data-role="controlgroup"id="rdo-type"
    data-mini="true"data-type="horizontal"
    style="padding: 5px 0px 0px 0px; margin: 0px">
    <input type="radio"name="rdo-type"
    id="rdo-type-0"value="a"/>
    <label for="rdo-type-0"id="lbl-type-0">散文</label>
    <input type="radio"name="rdo-type"
    id="rdo-type-1"value="b"/>
    <label for="rdo-type-1"id="lbl-type-1">随笔</label>
    <input type="hidden"id="hidtype"value="a"/>
    </feldset>
    <label for="txt-title">标题：</label>
    <input type="text"name="txt-title"
    id="txt-title"value=""/>
    <label for="txta-content">正文：</label>
    <textarea name="txta-content"id="txta-content">
</textarea>
    </div>
    <div data-role="footer"data-position="fixed">
    <h1>©2012 rttop.cn studio</h1>
    </div>
```

```
    </div>
    <script src="Js/rttopHtml5.base.js"
    type="text/javascript"></script>
    <script src="Js/rttopHtml5.note.js"
    type="text/javascript"></script>
    </body>
    </html>
```

在本系统的全局JavaScript文件rttopHtml5.note.js中，用于实现修改记事内容页的代码如代码清单10-11所示。

代码清单10-11　rttopHtml5.note.js文件中修改记事内容页功能对应的代码

```
    ……省略其他与本页面功能不相关代码
    //修改记事页面创建事件
    $("#editnote_index").live("pageshow", function() {
    var$strId=rttophtml5mobi.utils.getParam('list_link_id')
;
    var$header=$(this).fnd('div[data-role="header"]');
    var$rdotype=$("input[type='radio']");
    var$hidtype=$("#hidtype");
    var$txttitle=$("#txt-title");
    var$txtacontent=$("#txta-content");
    var
editData=JSON.parse(rttophtml5mobi.utils.getParam($strId))
;
    $hidtype.val(editData.type);
    $txttitle.val(editData.title);
    $txtacontent.val(editData.content);
    if(editData.type=="a"){
    $("#lbl-type-0").removeClass("ui-radio-off")
    .addClass("ui-radio-on ui-btn-active");
    }else{
    $("#lbl-type-1").removeClass("ui-radio-off")
    .addClass("ui-radio-on ui-btn-active");
    }
    $rdotype.bind("change", function() {
    $hidtype.val(this.value);
```

```
});
$header.delegate('a', 'click', function(e){
if($txttitle.val().length>0&&
$txtacontent.val().length>0){
var strnid=$strId;
var notedata=new Object;
notedata.nid=strnid;
notedata.type=$hidtype.val();
notedata.title=$txttitle.val();
notedata.content=$txtacontent.val();
var jsonotedata=JSON.stringify(notedata);
rttophtml5mobi.utils.setParam(strnid, jsonotedata);
window.location.href="list.htm";
}
})
})
```

……省略其他与本页面功能不相关代码

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-6所示。

图　10-6　修改记事数据时的效果

4.源码分析

在实现修改记事内容页功能的代码清单10-11中，首先调用自定义的getParam（）方法获取当前修改的记事数据Id号并保存在变量“$strId”中；然后，将该变量值作为“localStorage”对象的键名，通过该键名获取对应的键值字符串，并将该字符串转换成JSON格式对象。在对象中，通过属性的方式获取记事数据的类别、主题和正文信息，依次显示在页面指定的元素中。

当通过水平式的单选按钮组显示记事类型数据时，先将对象的类型值保存在Id号为“hidtype”的隐藏类元素中，再根据该值的内容，使用removeClass（）和addClass（）方法修改按钮组中单个按钮的样式，使整个按钮组的选中项与记事数据的类型一致；同时，设置单选按钮组的“change”事件，在该事件中，当修改原有类型时，Id号为“hidtype”的隐藏类元素的值也随之发生变化，以确保记事类型修改后的值可以实时保存。

最后，设置头部栏中右侧“更新”按钮的单击事件。在该事件中，先检测主题文本框和内容区域框的字符长度是否大于0，来检测主题和内容是否为空。当两者都不为空时，实例化一个新的“Object”对象，并将记事数据的各信息作为该对象的属性值，保存在该对象中；然后，通过调用JSON.stringify（）方法将对象转换成JSON格式的文本字符串，使用自定义的setParam（）方法，将数据写入

“localStorage”对象对应键名的键值中，最终实现记事数据更新的功能。更多详细实现方法，见代码清单中加粗部分。

10.7　添加记事内容页开发

在系统首页和记事列表页中，单击头部栏右侧的“增加”按钮后，都将进入添加记事内容页，在该页面中，用户可以通过单选按钮组选择记事类型，在文本框中输入记事主题，在区域框中输入记事内容，单击该页面头部栏右侧的“保存”按钮后，便新增了一条记事数据。

实例10-6　添加记事内容页开发

1.功能说明

新建一个HTML页面，在“page”容器的正文区域中，水平式的单选按钮组用于选择记事类型，一个文本框和一个文本区域框分别用于输入记事主题和内容，当用户选择记事数据类型和输入记事数据主题和内容并单击“保存”按钮后，则完成数据的添加操作，将返回记事详细页。

2.实现代码

新建一个HTML页面addnote.htm，加入代码如代码清单10-12所示。

代码清单10-12　添加记事内容页开发

```
    <! DOCTYPE html>
    <html>
    <head>
    <title>增加记事页_荣拓移动记事本系统</title>
    <meta name="viewport"content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0; "/
>
    <link href="Css/jquery.mobile-1.0.1.min.css"
    rel="stylesheet"type="text/css"/>
    <link href="Css/rttopHtml5.css"
    rel="stylesheet"type="text/css"/>
    <script src="Js/jquery-1.6.4.js"
    type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.0.1.js"
    type="text/javascript"></script>
    </head>
    <body>
    <div data-role="page"id="addnote_index">
    <div data-role="header"data-position="fixed"
    data-position="inline">
    <a href="javascript: "data-rel="back">返回</a>
    <h4>增加记事</h4>
    <a href="javascript: ">更新</a>
    </div>
    <div data-role="content">
    <label for="rdo-type">类型：</label>
    <feldset data-role="controlgroup"id="rdo-type"
    data-mini="true"data-type="horizontal"
    style="padding: 5px 0px 0px 0px; margin: 0px">
    <input type="radio"name="rdo-type"
    id="rdo-type-0"value="a"checked="checked"/>
    <label for="rdo-type-0">散文</label>
    <input type="radio"name="rdo-type"
    id="rdo-type-1"value="b"/>
    <label for="rdo-type-1">随笔</label>
    <input type="hidden"id="hidtype"value="a"/>
    </feldset>
    <label for="txt-title">标题：</label>
    <input type="text"name="txt-title"
    id="txt-title"value=""/>
    <label for="txta-content">正文：</label>
    <textarea name="txta-content"id="txta-content">
</textarea>
    </div>
```

```
<div data-role="footer"data-position="fixed">
<h1>©2012 rttop.cn studio</h1>
</div>
</div>
<script src="Js/rttopHtml5.base.js"
type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
type="text/javascript"></script>
</body>
</html>
```

在本系统的全局JavaScript文件rttopHtml5.note.js中，用于实现添加记事内容页的代码如代码清单10-13所示。

代码清单10-13　rttopHtml5.note.js文件中添加记事内容页功能对应的代码

```
//增加记事页面创建事件
$("#addnote_index").live("pagecreate", function() {
var$header=$(this).fnd('div[data-role="header"]');
var$rdotype=$("input[type='radio']");
var$hidtype=$("#hidtype");
var$txttitle=$("#txt-title");
var$txtacontent=$("#txta-content");
$rdotype.bind("change", function() {
$hidtype.val(this.value);
});
$header.delegate('a', 'click', function(e) {
if($txttitle.val().length>0&&
$txtacontent.val().length>0) {
var strnid="note_"+RetRndNum(3);
var notedata=new Object;
notedata.nid=strnid;
notedata.type=$hidtype.val();
notedata.title=$txttitle.val();
notedata.content=$txtacontent.val();
var jsonotedata=JSON.stringify(notedata);
rttophtml5mobi.utils.setParam(strnid, jsonotedata);
window.location.href="list.htm";
```

```
}
});
function RetRndNum(n){
var strRnd="";
for(var intI=0; intI<n; intI++){
strRnd+=Math.foor(Math.random() *10);
}
return strRnd;
}
})
```

3.页面效果

该页面在Opera Mobile Emulator 12.0下执行的效果如图10-7所示。

4.源码分析

实现增加记事内容页功能的代码清单10-13相对简单，首先通过定义一些变量保存页面中的各元素对象，并设置单选按钮组的“change”事件。在该事件中，当单选按钮的选项中发生变化时，保存选项值的隐藏型元素值也将随之变化。

然后，使用delegate（）添加头部元素右侧“更新”按钮的单击事件。在该事件中，先检测主题文本框和内容文本域的内容是否为空，如果不为空，那么调用一个自定义的按长度生成随机数的函数，生成一个一个3位数的随机数字，并与“note_”字符一起组成记事数据的Id号保存在变量“strnid”中。

图　10-7　添加记事数据时的效果

最后，实例化一个新的“Object”对象，将记事数据的Id号、类型、标题、正文内容都作为该对象的属性值赋值于对象，再使用JSON.stringify（）方法将对象转换成JSON格式的文本字符串，通过自定义的setParam（）方法，保存在以记事数据的Id号为键名的对应键值中，实现添加记事数据的功能。更多详细实现方法，见代码清单中加粗部分。

10.8　样式文件

在开发本系统功能时，除上述HTML页面和JavaScript文件外，还有一个全局性的样式文件rttopHtml5.css，用于控制整个系统的页面样式与结构布局，该文件的详细代码如代码清单10-14所示。

代码清单10-14　rttopHtml5.css文件全部代码

```
#notenav_wrap
{
position: relative; width: 100%;
height: auto; min-height: 322px;
overflow: hidden;
}
#notenav_wrap ul
{
position: absolute; list-style-type: none;
z-index: 2; margin: 0; bottom: 0px;
padding: 0; left: 45%;
}
#notenav_wrap ul li
{
background: url（images/icons_off.png）
center no-repeat; width: 12px;
height: 12px; float: left;
margin-right: 8px;
}
#notenav_wrap ul li.on
{
background: url（images/icons_on.png）
center no-repeat; width: 12px;
height: 12px; line-height: 12px;
float: left; margin-right: 8px; font-size: 10px;
text-align: center; color: #666; font-family: Arial
}
#notenav_list a
{
```

```
position: absolute; width: 100%;
}
#notenav_list a img
{
border: 0px; width: 100%;
height: auto; height: 298px;
}
#title
{
margin: 0px; text-align: center
}
.notep
{
border-bottom: solid 1px#ccc
}
.ui-btn-corner-all
{
border-radius: .2em;
}
.ui-header.ui-btn-inner
{
font-size: 12.5px; padding: .35em 6px.3em;
}
.ui-btn-inner
{
padding: .3em 20px; display: block;
text-overfow: ellipsis; overfow: hidden;
white-space: nowrap;
position: relative; zoom: 1;
}
```

在上述代码中，有两个样式类别需要说明：

1）为了使头部栏两侧按钮的两端圆角弧度更小，在样式文件中重置了“ui-btn-corner-all”类别，修改了“border-radius”属性值。

2）为了修改头部栏两侧按钮的高度和单选按钮组的内边距离，在样式文件中重置了“ui-btn-inner”类别，修改了相应的“padding”属性值。

样式文件中的其余代码，一部分用于滑动图片时控制图片和列表元素的样式；另一部分用于显示记事数据详细页时的样式控制。

10.9　本章小结

本章通过一个完整的移动终端记事本管理系统的开发，详细介绍了在jQuery Mobile框架中，如何使用“localStorage”实现数据的增加、删除、修改和查询。“localStorage”对象是HTML 5新增加的一个对象，用于在客户端保存用户的数据信息，它以“key/value”的方式进行数据的存取，并且该对象目前被绝大多数新版移动设备的浏览器所支持，因此，使用“localStorage”对象开发项目越来越多。相信通过本章实例的学习，为读者在移动项目中如何使用“localStorage”对象打下扎实的理论和实践基础。

超级畅销书全新升级，修订篇幅超过40%，第1版被公认为jQuery领域标杆之作，销量和口碑俱佳

资深Web开发专家根据jQuery最新版本撰写，内容更加全面、深入、实用，近200个案例，实战性更强

實戰

第2版

陶国荣 著

jQuery: The Definitive Guide, Second Edition

jQuery 权威指南

机械工业出版社
China Machine Press

前言

为什么要写这本书

“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，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技术的发展进程中具有划时代的意义。

jQuery发布于2006年，因为其易于使用、功能强大、展现优雅、兼容性极佳等特点而迅速赢得了Web开发者的青睐。在此期间，jQuery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，这使得它在众多的Ajax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深深地吸引了我，令我深入其中，不能自拔。

随着Web开发技术的发展，以及用户对应用体验的要求日益提高，致使我们开发一个Web应用时，不仅仅考虑其功能是否足够完备，更重要的是考虑如何才能提高用户体验。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利刃。

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们迅速而有效地掌握它，并能真正付诸实践。为了让所有还没有完全掌握jQuery技术的开发者能迅速步入jQuery的殿堂，于是本书诞生了，相信它不会让你失望。

第2版与第1版的区别

本书第1版自2011年上半年面市以来，一直受到广大读者的关爱，在此笔者深表感谢。

从2011年初至2012年底，jQuery框架历经了多次版本升级，目前稳定版本为1.8.2，在这些升级的版本中，1.5和1.7这两个版本十分重要，前者首次引入了Ajax重写和延迟对象的概念，后者新增了多项事件并提升了在IE浏览器中对HTML 5的支持性能，而在2012年底发布的1.8版本，则是最为稳定和完善的版本，本书中所有案例均以1.8.2版本为框架，旨在使读者在本书中感受jQuery最新版本的强大功能。

从本书第1版上市以来，收到了读者发来的大量邮件，提出了许多宝贵的意见和期盼，基于读者的这些想法，我们在本书的第2版作了重大改动，主要表现在如下几个方面。

1）新增第7章“jQuery中调用JSON与XML数据”。

JSON和XML是前端与服务端最常用的数据交互格式。在新增章节中，从基础概念讲起，以实例为主线，逐步深入地介绍在jQuery中应用JSON和XML数据格式的方法与技巧。

2）新增第11章“jQuery常用开发技巧”。

在jQuery框架中，常用的开发技巧与解决方案一直都是初学者最需要了解的。在新增章节中，列举了目前使用jQuery开发Web页面时，最常遇到的一些问题和优化方案。通过对本章的学习，读者能够最大化地优化代码，完善结构提供方向。

3）新增第13章“jQuery在HTML 5中的应用”。

HTML 5是前端开发人员的新方向和标准。在新增章节中，精选了4个完整案例，以HTML 5新增元素＜canvas＞、＜video＞为主线，以项目开发的形式，介绍在HTML 5标准中，开发基于jQuery框架的Web应用。关注HTML 5中使用jQuery框架开发游戏和视频的读者，本章的案例不容错过。

4）新增第14章“jQuery Mobile基础知识”。

移动开发是当前最为热门的话题之一，而jQuery Mobile是jQuery专门针对移动开发推出的一个全新的移动开发框架。在新增章节中，将从框架的基础组件讲起，由浅入深地介绍jQuery Mobile框架中常用API的使用方法和技巧。学习本章节的内容，可以为从事Web App开发打下扎实的理论基础。

5）新增第15章“jQuery Mobile综合案例开发”。

秉着“学以致用”的目的，为了使读者更好地了解并掌握使用jQuery Mobile框架开发Web App的详细过程，加深对jQuery Mobile框架中各组件和API的使用，在新增的本章节中，将结合第14章中的基础应用知识，精选两个典型的案例，以项目开发的方式，介绍使用jQuery Mobile框架开发Web App的完整过程。

另外，在新版的第8章、第9章、第12章中，新增加了若干小节，其中，结合了当前最为流行的应用需求，以实践开发为主，技术逻辑讲解为辅，完整、翔实地为读者展示每一个技术点，相信这些新章节将为读者朋友在技术上带来全新的开发思路和全面的应用体验。

本书特点

与国内目前已经出版的同类书相比较，本书具有以下几个独有的特点：

❑基于jQuery的最新版本撰写，完美地展现jQuery最新版本的功能和特性。

❑内容全面、丰富、翔实，由浅入深地对jQuery的所有必备基础知识进行了详尽介绍，还对jQuery UI等扩展知识以及jQuery开发中的技巧与性能优化方面的高级知识进行了讲解。

❑本书极其注重实战，因为动手实践才是掌握一门新技术的最有效途径。书中的每一个小知识点都配有经过精心选择的示例（总共近200个），还有多个非常实用的综合性案例。所有案例的讲解非常详细，包括功能需求分析和完整实现代码，还有最终效果展示，更重要的是将所有理论知识都巧妙地贯穿其中，非常易于读者理解。如果读者能在阅读本书的过程中逐一亲手实现这些案例，应该可以在实际开发中具备相当的动手能力了。

本书面向的读者

本书适合所有希望迅速掌握jQuery并将之付诸实践的Web开发者阅读。

如何阅读本书

本书结构是层进式的，章节之间有一定的关联，建议读者按章节的编排逐章阅读。在阅读本书的案例时，建议不要照抄书中的所有案例，重在理解代码的实现思路，自己动手开发相似功能的应用，并逐步完善其功能，这样才能真正领会案例所反映出的jQuery技术的理论本质。

联系作者

希望这部耗时数月、承载了我近六年jQuery开发心得和体会的拙著，能给每一位阅读过它的读者带来技术上的提升和思路上的启发。我非常希望能借这本书出版的机会与国内热衷于jQuery技术的开发者交流，如果大家想联系我，欢迎发邮件（tao_guo_rong@163.com）。此外，本书中的示例代码可以从华章网站[1]本书主页下载。

[1] 华章网址www.hzbook.com。

致谢

本书能顺利出版，首先要感谢机械工业出版社华章分社的编辑们，尤其是杨福川和白宇编辑，他们在我写作的整个过程中不断地给予专业的指导，才使得整体的创作思路不断被提升和改进，最后使本书能保质保量地完成。同时，我还要感谢我的家人，正是他们的理解与默默支持才使得我能全心写作，顺利完成本书的写作。

陶国荣

第1章　jQuery简介

本章内容

认识jQuery

搭建jQuery开发环境

jQuery程序的代码风格

jQuery简单应用

本章小结

随着互联网的迅速发展，Web页面的广泛应用，人们的需求不仅限于页面的功能，而更多地注重页面展示形式和用户体验度。JavaScript语言可以很好地满足程序开发者的需求，开发出用户体验度很高的页面，因而越来越受到广大程序员的关注；jQuery是JavaScript库中的优秀一员，近年来，随着它的代码高效、兼容性强而风靡全球，越来越多的开发者痴迷其中。

1.1　认识jQuery

jQuery是由美国人John Resig于2006年创建的一个开源项目，随着被人们的熟知，越来越多的程序高手加入其中，完善和壮大其项目内容；如今已发展成为集JavaScript、CSS、DOM、Ajax于一体的强大框架体系，它的主旨是：以更少的代码，实现更多的功能（Write less，do more）。

1.1.1　jQuery基本功能

1.访问和操作DOM元素

使用jQuery库，可以很方便地获取和修改页面中的某元素，无论是删除、移动、复制某元素，jQuery都提供了一整套方便、快捷的方法，既减少了代码的编写，又大大提高了用户对页面的体验度。具体示例我们将在后面的章节中陆续展示。

2.控制页面样式

通过引入jQuery，程序开发人员可以很便捷地控制页面的CSS文件。浏览器对页面文件的兼容性一直以来都是页面开发者最为头痛的事，而使用jQuery操作页面的样式，却可以很好地兼容各种浏览器。

3.对页面事件的处理

引入jQuery库后，使页面的表现层与功能开发分离，开发者可以更多地专注于程序的逻辑与功效；页面设计者侧重于页面的优化与用户体验，通过事件绑定机制，可以很轻松地实现二者的结合。

4.大量插件在页面中的运用

在引入jQuery库后，还可以使用大量的插件来完善页面的功能和效果，如表单插件、UI插件，这些插件的使用，极大丰富了页的展示效果，原来使用JavaScript代码遥不可及的功能，通过插件的引入都可以轻松实现。

5.与Ajax技术的完美结合

Ajax的异步读取服务器数据的方法，极大方便了程序的开发，加深了用户的页面体验度；而引入jQuery库后，不仅完善了原有的功能，而且减少了代码的书写，利用其内部对象或函数，加上几行代码就可以实现复杂的功能。

1.1.2　jQuery 1.8新增功能与特征

本书的全部案例以jQuery 1.8.2为框架，该版本具有以下几个重要的新增功能与特征。

1.根据浏览器类型自动为CSS属性添加对应的前缀名称

在jQuery 1.8及以上版本中，使用jQuery设置一些尚未正式纳入W3C标准的样式属性时，将会根据浏览器的类型，自动在属性前添加对应的前缀名称，如设置"marquee-direction"属性时，如果在Chrome浏览器中执行时，则会自动变为"-webkit-marquee-direction"。

2.重构了动画方法

在jQuery 1.8及以上版本中，通过改进后的$.Animation函数，用户可以更加容易地添加或修改动画。在改进功能的同时，还修复了许多动画的Bug，使动画效果既具有综合性，又具有代码的扩展性。

3.优化了选择器引擎

在jQuery 1.8及以上版本中，不仅重写了选择器引擎，而且还对原有的引擎功能进行了性能优化，修复了一些边缘问题和Bug，其中包括对多个选择符“～>+”功能的改进；同时，还清理了代码，使jQuery 1.8及以上版本比jQuery 1.7.2的体积少几百字节。

4.强化XSS防护功能

XSS为Cross Site Scripting缩写，意为跨站点脚本代码攻击，为避免与CSS缩写重复，故缩写为XSS。在jQuery 1.8及以上版本中，通过新增加的"$.parseHTML"方法，可以将方法中的字符串解析为DOM元素块，又可以控制字符串中脚本的执行，防范脚本代码的攻击。

5.自定义专属版本

我们在开发过程中，往往使用的jQuery功能只有少量部分，还有大部分的功能被闲置，而在jQuery 1.8及以上版本中，用户可以通过基于grunt的构建系统，移除这些被闲置的模块，重新自定义一个专属版本，目前可移除的模块包括ajax、css、dimensions、effects和offset。

此外，jQuery在页面中的功能还有很多，我们将在接下来的章节中一一介绍。

1.2　搭建jQuery开发环境

由于jQuery是一个完整的JavaScript文件库，因此，搭建jQuery开发环境十分简单，无须安装任何文件，只需要先在jQuery官方网站下载最新的文件库，然后将该文件库引入页面中的<head>元素间即可。

1.2.1　下载jQuery文件库

在jQuery的官方网站（http://jquery.com）下载最新版本的jQuery文件库，其网站页面如图1-1所示。

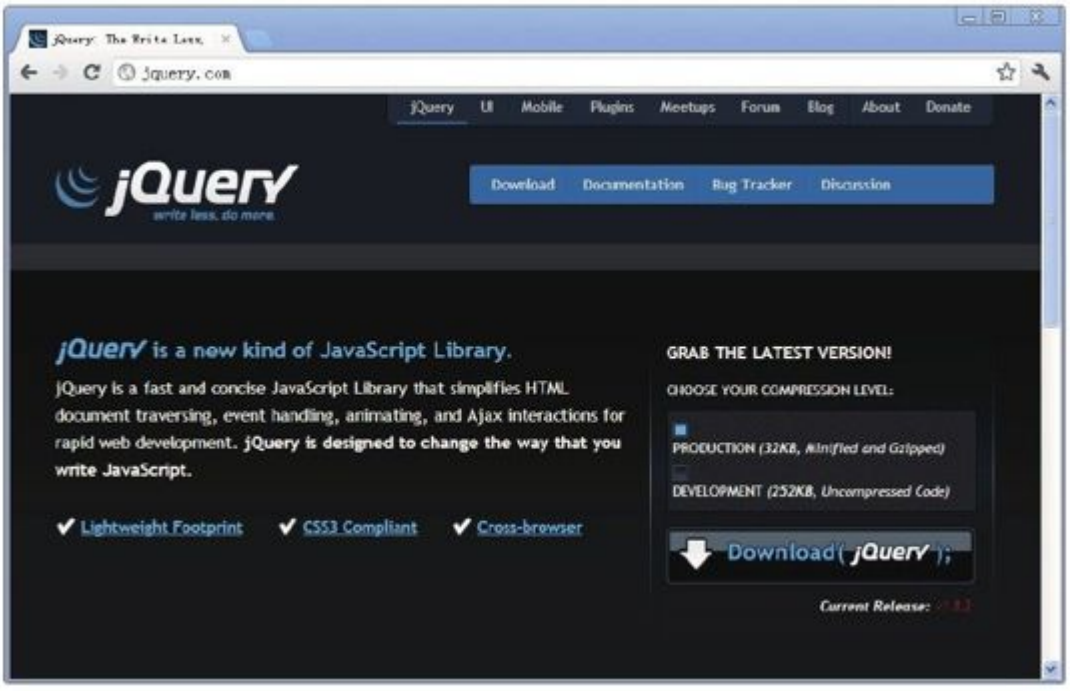


图　1-1　jQuery官方网站

在网站中，选择大小为32KB的压缩包，单击下载按钮，便可以将最新版的jQuery框架下载到本地；目前最新版本为v1.8.2。

1.2.2　引入jQuery文件库

下载完jQuery框架文件后，并不需要任何的安装，仅需要使用＜script＞文件导入标记，将jQuery框架文件jquery-1.8.2.min.js导入页面中即可。假设该文件下载后保存在项目文件夹Jscript中，那么，在页面的＜head＞＜/head＞中加入如下代码：

```
<script language="javascript" type="text/javascript" src="Jscript/jquery-1.8.2.min.js"></script>
```

在页面的头部分加入上述代码后，便完成了jQuery框架的引入，现在可以开始我们的jQuery之旅了。

1.2.3　编写第一个简单的jQuery程序

首先，我们来编写一个简单的程序，见示例1-1。

示例1-1　编写第一个简单的jQuery程序

（1）功能描述

当页面加载时，以居中的方式在页面中显示“您好！欢迎来到jQuery的精彩世界。”字样。

（2）实现代码

新建一个HTML文件1-1.html,加入如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <title>第一个简单的 jQuery 程序 </title>
    <style type="text/css">
       div{padding:8px 0px;font-size:12px;text-align:center;border:solid 1px #888;}
    </style>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $("div").html(" 您好！ 欢迎来到 jQuery 的精彩世界。");
```

```
        });
    </script>
</head>
<body>
    <div></div>
</body>
</html>
```

（3）页面效果

页面效果如图1-2所示。

图　1-2　第一个jQuery程序

（4）代码分析

在上述文件的代码中，有一段如下的代码：

```
$(document).ready(function(){
                    // 程序段
                })
```

该段代码类似于传统的JavaScript代码：

```
window.onload=function(){
        // 程序段
}
```

虽然上述两段代码在功能上可以互换，但它们之间又有许多区别：

❑执行时间不同：$(document).ready在页面框架下载完毕后就执行；而window.onload必须在页面全部加载完毕（包含图片下载）后才能执行。很明显前者的执行效率高于后者。

❑执行数量不同：$(document).ready可以重复写多个，并且每次执行结果不同；而window.onload尽管可以执行多个，但仅输出最后一个执行结果，无法完成多个结果的输出。

$(document).ready(function(){})可以简写成$(function(){})，因此下面的代码是等价的。

```
$(document).ready(function(){
                    // 程序段
                })
// 等价于
$(function(){
                    // 程序段
                })
```

1.3　jQuery程序的代码风格

1.3.1　“$”美元符的使用

在jQuery程序中，使用最多的莫过于“$”美元符了，无论是页面元素的选择、功能函数的前缀都须使用该符号，可以说它是jQuery程序的标志。例如下列代码：

```
$("#tip").html("hello world").show(1000);
```

上述代码表示1000毫秒后，在ID号为"tip"的元素中显示"hello world"字样。

1.3.2　事件操作链接式书写

在编写页面某元素事件时，jQuery程序可以使用链接式的方式编写该元素的所有事件。为了更好地说明该书写方法的使用，我们通过一个示例加以阐述。

示例1-2　jQuery事件的链式写法

（1）功能描述

在示例1-1基础之上，增加两个＜div＞元素，一个为框架，另一个为标题。示例1-1显示的文字为内容，框架元素包含标题和内容元素。当页面首次加载时，被包含的内容＜div＞标记是不可见的，当用户单击主题＜div＞标记时，改变自身的背景色，并将内容＜div＞标记显示出来。

（2）实现代码

新建一个HTML文件1-2.html,加入如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 事件的链式写法 </title>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
       .iframe{ border:solid 1px #888;font-size:13px;}
       .title{ padding:6px;background-color:#EEE;}
       .content{ padding:8px 0px;font-size:12px; text-align:center;display:none;}
       .curcol{ background-color:#CCC}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".content").html(" 您好! 欢迎来到 jQuery 的精彩世界。 ");
            $(".title").click(function() {
                $(this).addClass("curcol")
                .next(".content").css("display", "block");
            });
        });
    </script>
</head>
<body>
   <div class="iframe">
       <div class="title"> 标题 </div>
       <div class="content"></div>
   </div>
</body>
</html>
```

在上述文件的代码中，加粗代码就是链式写法。

（3）页面效果

执行HTML文件1-2.html，实现的页面效果如图1-3所示。

图　1-3　DIV元素单击前后的页面对比效果

（4）代码分析

当用户单击Class名称为"title"的元素时，自身增加名称为"curcol"的样式；同时，将接下来的Class名称为"content"元素显示出来。可以看出两个功能的实现通过“.”符号链接在一起。

1.4　jQuery简单应用

在介绍使用jQuery开发简单应用之前，首先需要了解如何使用jQuery访问DOM元素，如何将DOM对象转化成jQuery对象，然后通过控制转成的jQuery对象，实现各类应用的功能。

1.4.1　jQuery访问DOM对象

1.什么是DOM对象

每一个页面都是一个DOM（Document Object Model，文本对象模型）对象，通过传统的JavaScript方法访问页面中的元素，就是访问DOM对象。

例如，页面中有两个＜div＞标记元素如下：

```
<div id="Tmp">测试文本</div>
<div id="Out"></div>
```

通过下面的JavaScript代码可以访问DOM对象，以及获取或设置其内容值：

```
var tDiv=document.getElementById("Tmp"); // 获取 DOM 对象
var oDiv=document.getElementById("Out"); // 获取 DOM 对象
var cDiv=tDiv.innerHTML;                 // 获取 DOM 对象中的内容
oDiv.innerHTML=cDiv;                     // 设置 DOM 对象中的内容
```

如果执行上面的JavaScript代码，将在ID为"divOut"的标记中显示ID为"Tmp"的标记内容。

2.什么是jQuery对象

在jQuery库中，通过本身自带的方法获取页面元素的对象，称为jQuery对象；为了同样实现在ID为"Out"的标记中显示ID为"Tmp"的标记内容，采用jQuery访问页面元素的方法，其实现的代码如下：

```
var tDiv=$("#Tmp");    // 获取 jQuery 对象
var oDiv=$("#Out");    // 获取 jQuery 对象
var cDiv=tDiv.html(); // 获取 jQuery 对象中的内容
oDiv.html(cDiv);       // 设置 jQuery 对象中的内容
```

通过代码对比可以看出，jQuery对象访问方法比DOM对象访问方法更简单、高效，它们都实现同样的功能。

1.4.2　jQuery控制DOM对象

在介绍使用jQuery控制DOM对象前，先通过一个简单的示例，说明如何用传统的JavaScript方法访问DOM对象。

示例1-3　控制DOM对象

（1）功能描述

在页面中，用户输入姓名、性别和婚姻状况，单击“提交”按钮后，将获取到的数据信息显示在页面＜div＞标记中。

（2）实现代码

新建一个HTML文件1-3.html,加入如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <title>控制 DOM 对象 </title>
    <style type="text/css">
           .iframe{ border:solid 1px #888;font-size:13px;}
           .title{ padding:6px;background-color:#EEE;}
           .content{ padding:8px;font-size:12px;}
           .tip{ background-color:#EEE;display:none; font-size:12px;padding:8px;}
           .txt{ border:solid 1px #888;}
           .btn{ border:solid 1px #888;width:60px;}
           .w260{ width:260px;}
    </style>
    <script type="text/javascript">
         function btn_Click(){
          // 获取文本框的值
          var oTxtValue=document.getElementById("Text1").value;
          // 获取单选框按钮值
          var oRdoValue=(Radio1.checked)?"男":"女";
          // 获取复选框按钮值
          var oChkValue=(Checkbox1.checked)?"已婚":"未婚";
          // 显示提示文本元素
          document.getElementById("Tip").style.display="block";
          // 设置文本元素的内容
          document.getElementById("Tip").innerHTML=
          oTxtValue+"<br>"+oRdoValue+"<br>"+oChkValue;
          }
    </script>
</head>
<body>
<div class="iframe">


     <div class="title">请输入如下信息 </div>
     <div class="content">
          姓名: <input id="Text1" type="text" class="txt"/><br />
          性别: <input id="Radio1" name="rdoSex" type="radio" value="男" />男
               <input id="Radio2" name="rdoSex" type="radio" value="女" />女 <br />
          婚否: <input id="Checkbox1" type="checkbox" /><br /><br />
               <input id="btnSubmit" type="button" value="提 交" class="btn"
          onclick="btn_Click();" /><br /><br />
     </div>
     <div id="Tip" class="tip"></div>
 </div>
 </body>
 </html>
```

以上代码使用传统的JavaScript方法获取用户输入的信息，并保存到变量中，最后通过document.getElementById("Tip").innerHTML方法显示所有的数据信息。

下面将示例1-3中的JavaScript代码进行修改，引入jQuery库，通过jQuery中的方法获取元素的值，并将获取的数据显示出来。其修改后的JavaScript代码如下所示：

```
<script language="javascript" type="text/javascript"
        src="Jscript/jquery-1.8.2.min.js"></script>
<script type="text/javascript">
$(function(){
   $("#btnSubmit").click(function(){
    //获取文本框的值
    var oTxtValue=$("#Text1").val();
    //获取单选框按钮值
    var oRdoValue=$("#Radio1").is (":checked")?"男":"女";
    //获取复选框按钮值
    var oChkValue=$("#Checkbox1").is (":checked")?"已婚":"未婚";
    //显示提示文本元素和内容
     $("#Tip").css("display","block").html(oTxtValue+"<br>"+oRdoValue+"<br>"+oChkValue);
     })
  })
   </script>
```

（3）页面效果

HTML文件1-3.html，最终实现的页面效果如图1-4所示。

图　1-4　控制jQuery对象

（4）代码分析

在修改后的JavaScript代码中，$("#Text1").val()在jQuery库中表示获取ID号为"Text1"的值；$("#Radio1").is(":checked")表示ID号为"Radio1"的单选按钮是否被选中，如果选中返回“男”，否则返回“女”。

可以看出，通过jQuery库中的方法访问或控制页面中的元素比使用传统的JavaScript代码更简洁，并且还兼容各种浏览器。

1.4.3　jQuery控制页面CSS

jQuery框架中通过自带JavaScript编程，提供全部CSS 3下的选择器，开发者可以首先定义自己的样式文件，然后通过jQuery中的addClass()方法，将该样式轻松地添加到页面中指定的某元素中，而不用考虑浏览器的兼容性。

下面通过一个简单的示例介绍如何使用jQuery中的toggleClass(className)方法实现页面样式的动态切换功能。

示例1-4　jQuery控制CSS样式

（1）功能描述

在页面中，增加一个＜div＞元素标记，用户单击该元素时，变换其字体和背景色，再次单击时，恢复其初始的字体和背景色。

（2）实现代码

新建一个HTML文件1-4.html,加入如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 事件的链式写法 </title>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
       .iframe{ border:solid 1px #888;font-size:13px;}
       .defcol{ padding:6px;background-color:#EEE;}
       .curcol{ padding:6px;background-color:#CCC;color:#FFF}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".defcol").click(function() {
                $(this).toggleClass("curcol");
            });
        });
    </script>
</head>
<body>
   <div class="iframe">
       <div class="defcol"> 标题 </div>
   </div>
</body>
</html>
```

（3）页面效果

HTML文件1-4.html执行后，最终实现的页面效果如图1-5所示。

图　1-5　jQuery控制CSS

（4）代码分析

在jQuery库中，通过简单的几行代码，就可以实现传统JavaScript大量代码完成的功能，节省开发者的时间，提高工作效率。

1.5　本章小结

本章通过循序渐进的方式，首先介绍jQuery的基本功能，同时列举了1.8版本的新增功能与特征；然后介绍jQuery库的下载、引入，以及简单的应用方法；通过一些简单的小示例，介绍了jQuery控制DOM对象和页面CSS样式，使读者对jQuery在页面中的功能应用有一个大致的了解，为下一章节进一步学习jQuery库的详细对象和方法奠定基础。

第2章　jQuery选择器

本章内容

选择器的优势

jQuery选择器的类型

综合案例分析——导航条在项目中的应用

本章小结

通过对第1章的介绍，相信大家对jQuery在前台页面中的应用有了一个初步的了解。在页面中为某个元素添加属性或事件时，必须先准确地找到该元素——在jQuery库中，可以通过选择器实现这一重要的核心功能。本章将详细介绍在jQuery中如何通过选择器快速定位元素的方法和技巧。

jQuery选择器继承了CSS与Path语言的部分语法，允许通过标签名、属性名或内容对DOM元素进行快速、准确的选择，而不必担心浏览器的兼容性，通过jQuery选择器对页面元素的精准定位，才能完成元素属性和行为的处理。

2.1　选择器的优势

与传统的JavaScript获取页面元素和编写事务相比，jQuery选择器具有明显的优势，具体表现在以下两个方面：

❑代码更简单。

❑完善的检测机制。

下面将详细介绍这两个方面。

2.1.1　代码更简单

由于在jQuery库中，封装了大量可以通过选择器直接调用的方法或函数，使编写代码更加简单轻松，简单几行代码就可以实现较为复杂的功能。

下面通过一个实现表格隔行变色功能的示例，分别使用传统的JavaScript语言与jQuery语言加以说明。

示例2-1　分别使用JavaScript和jQuery实现隔行变色

（1）功能描述

在页面中，通过一个＜table＞标记展示数据列表信息，在元素标记中，奇数行与偶数行的背景色不同，从而实现隔行变色的页面展示效果。

（2）实现代码

使用传统的JavaScript实现该页面功能。新建一个HTML文件2-1.html，加入如代码清单2-1所示的代码。

代码清单 2-1　使用 JavaScript 实现隔行变色

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 JavaScript 实现隔行变色 </title>
    <style type="text/css">
        body{font-size:12px;text-align:center}
        #tbStu{width:260px;border:solid 1px #666;background-color:#eee}
        #tbStu tr{line-height:23px}
```

```
        #tbStu tr th{background-color:#ccc;color:#fff}
        #tbStu .trOdd{background-color:#fff}
    </style>
    <script type="text/javascript">
        window.onload=function(){
        var oTb=document.getElementById("tbStu");
           for(var i=0;i<oTb.rows.length-1;i++){
              if(i%2){
                 oTb.rows[i].className="trOdd";
              }
           }
        }
    </script>
</head>
<body>
<table id="tbStu" cellpadding="0" cellspacing="0">
    <tbody>
        <tr>
            <th>学号</th><th>姓名</th><th>性别</th><th>总分</th>
        </tr>
            <!--奇数偶-->
        <tr>
            <td>1001</td><td>张小明</td><td>男</td><td>320</td>
        </tr>
            <!--偶数偶-->
        <tr>
            <td>1002</td><td>李明琪</td><td>女</td><td>350</td>
        </tr>
        <!--...-->
    </tbody>
</table>
</body>
</html>
```

在代码清单2-1中，首先通过ID号获取表格元素，然后遍历表格的各行，根据行号的奇偶性，动态设置该行的背景色，从而实现隔行变色的页面效果。

页面中的JavaScript代码虽可以实现最终效果，但循环页面的元素会影响打开速度，并且代码较为复杂，如果使用jQuery选择器实现上述页面效果，则需要在页面中加入一些代码：

```
...
<head>
    <title>使用 jQuery 选择器实现隔行变色 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    ... 省略样式代码
    <script type="text/javascript">
          $(function(){
            $("#tbStu tr:nth-child(even)").addClass("trOdd");
          })
    </script>
</head>
... 省略页面主体代码
```

（3）页面效果

虽然示例2-1和示例2-2中的代码不同，但都实现了页面数据隔行变色的功能，其最终实现的页面效果完全相同，如图2-1所示。

图　2-1　页面数据隔行变色效果

可以看出，使用jQuery选择器可以很快捷地定位页面中的某个元素，并设置该元素的相应属性，具有代码简单、执行效果高的优点。

2.1.2　完善的检测机制

在传统的JavaScript代码中，给页面中某元素设置事务时必须先找到该元素，然后赋予相应的属性或事件；如果该元素在页面中不存在或被前面代码所删除，那么浏览器将提示运行出错信息，影响后续代码的执行。因此，在JavaScript代码中，为了避免显示这样的出错信息，先要检测该元素是否存在，然后再运行其属性或事件代码，从而导致代码冗余，影响执行效率。

在jQuery选择器定位页面元素时，无须考虑所定位的元素在页面中是否存在，即使该元素不存在该元素，浏览器也不提示出错信息，极大地方便了代码的执行效率。

下面通过一个简单的示例分别使用JavaScript语言与jQuery语言来说明该检测机制在页面中的实现效果。

示例2-2　分别使用JavaScript和jQuery输出文字信息

（1）功能描述

在页面＜div＞标记中输出一行“这是一个检测页面”的字符。

（2）实现代码

使用传统的JavaScript实现该页面功能。

新建一个HTML文件2-2.html，加入如代码清单2-2所示的代码。

代码清单 2-2　使用 JavaScript 输出文字信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">


 <head>
     <title>JavaScript 代码检测页面元素 </title>
     <style type="text/css">
         body{font-size:12px;text-align:center}
     </style>
     <script type="text/javascript">
         window.onload=function(){
            if(document.getElementById("divT")){
                   var oDiv=document.getElementById("divT");
                   oDiv.innerHTML=" 这是一个检测页面 ";
                  }
         }
     </script>
 </head>
 <body>
 </body>
 </html>
```

（3）页面效果

在JavaScript代码中，有一行代码如下：

```
if(document.getElementById("divT")){...}
```

该行代码用于检测所定位的页面元素是否存在，如果存在，则执行下面的代码，否则不执行；假设不编写该行代码检测元素的存在，则浏览器中将出现如图2-2所示的出错提示信息。

图　2-2　页面对象不存在的出错提示信息

如果将例2-2中的JavaScript代码改写成jQuery选择器方式获取页面元素，那么不需要检测元素是否存在，且页面正常执行，其修改后的代码如下所示：

```
<head>
    <title>jQuery 代码检测页面元素 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    ... 省略样式代码
    <script type="text/javascript">
          $(function(){
            $("#divT").html(" 这是一个检测页面 ");
         })
```

```
    </script>
</head>
... 省略页面主体代码
```

2.2　jQuery选择器的类型

根据所获取页面中元素的不同，可以将jQuery选择器分为四大类：基本选择器、层次选择器、过滤选择器、表单选择器。其中，在过滤选择器中又可分为：简单过滤选择器、内容过滤选择器、可见性过滤选择器、属性过滤选择器、子元素过滤选择器、表单对象属性过滤选择器。其分类结构如图2-3所示。

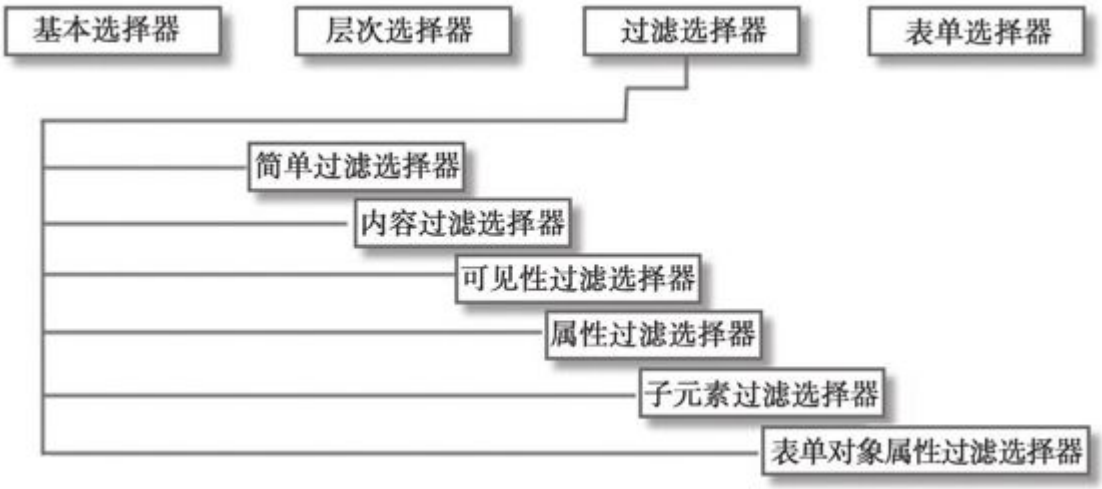


图　2-3　jQuery选择器分类示意图

2.2.1　基本选择器

基本选择器是jQuery中使用最频繁的选择器，它由元素ID、Class、元素名、多个选择符组成，通过基本选择器可以实现大多数页面元素的查找，其详细说明如表2-1所示。

表 2-1　基本选择器语法

选择器	功能描述	返回值
#id	根据给定的 ID 匹配一个元素	单个元素
element	根据给定的元素名匹配所有元素	元素集合
.class	根据给定的类匹配元素	元素集合
*	匹配所有元素	元素集合
selector1,selectorN	将每一个选择器匹配到的元素合并后一起返回	元素集合

下面通过示例2-3来介绍各种基本选择器在页面中使用的方法。

示例2-3　使用jQuery基本选择器选择元素

（1）功能描述

一个页面包含两个＜div＞标记，其中一个用于设置ID属性，另一个用于设置Class属性；我们再增加一个＜span＞标记，全部元素初始值均为隐藏，然后通过jQuery基本选择器显示相应的页面标记。

（2）实现代码

新建一个HTML文件2-3.html，加入如代码清单2-3所示的代码。

代码清单 2-3　使用 jQuery 基本选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 基本选择器</title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           .clsFrame{width:300px;height:100px}
           .clsFrame div,span{display:none;float:left;
                    width:65px;height:65px;border:solid 1px #ccc;
                    margin:8px}
           .clsOne{background-color:#eee}
    </style>
    <script type="text/javascript">
           $(function(){ //ID 匹配元素
             $("#divOne").css("display","block");
           })
           $(function(){ // 元素名匹配元素
             $("div span").css("display","block");
           })
           $(function(){ // 类匹配元素
             $(".clsFrame .clsOne").css("display","block");
           })
           $(function(){ // 匹配所有元素
             $("*").css("display","block");
           })
           $(function(){ // 合并匹配元素
             $("#divOne,span").css("display","block");
           })
    </script>
</head>
<body>
<div class="clsFrame">
    <div id="divOne">ID</div>
    <div class="clsOne">CLASS</div>
    <span>SPAN</span>
</div>
</body>
</html>
```

（3）页面效果

为了能更清楚地看到每个基本选择器执行后的结果，下面通过表格的方式展示页面效果，如表2-2所示。

表 2-2　页面执行效果

关键代码	功能描述	页面效果
$("#divOne") .css("display","block");	显示 ID 为 divOne 的页面元素	
$(" div span ") .css("display","block");	显示元素名为 span 的页面元素	
$(".clsFrame .clsOne") .css("display","block");	显示类别名为 clsOne 的页面元素	
$("*") .css("display","block");	显示页面中的所有元素	
$("#divOne,span") .css("display","block");	显示 ID 为 divOne 和元素名为 span 的页面元素	

2.2.2 层次选择器

层次选择器通过DOM元素间的层次关系获取元素，其主要的层次关系包括后代、父子、相邻、兄弟关系，通过其中某类关系可以方便快捷地定位元素，其详细说明如表2-3所示。

表 2-3 层次选择器语法

选择器	功能描述	返回值
ancestor descendant	根据祖先元素匹配所有的后代元素	元素集合
parent > child	根据父元素匹配所有的子元素	元素集合
prev + next	匹配所有紧接在 prev 元素后的相邻元素	元素集合
prev ~ siblings	匹配 prev 元素之后的所有兄弟元素	元素集合

ancestor descendant与parent＞child所选择的元素集合是不同的，前者的层次关系是祖先与后代，而后者是父子关系；另外，prev+next可以使用.next()代替，而prev～siblings可以使用nextAll()代替。

下面通过示例2-4来演示各种层次选择器在页面中选择DOM元素的方法。

示例2-4 使用jQuery层次选择器选择元素

（1）功能描述

在页面中设置四个＜div＞标记，在第二个＜div＞中添加一个＜span＞标记，在该＜span＞标记中又新增一个＜span＞标记，全部元素初始值均为隐藏，然后通过jQuery层次选择器显示相应的页面标记。

（2）实现代码

新建一个HTML文件2-4.html，加入如代码清单2-4所示的代码。

代码清单 2-4　使用 jQuery 层次选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>使用 jQuery 层次选择器</title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           div,span{float:left;border:solid 1px #ccc; margin:8px;display:none}
           .clsFraA{width:65px;height:65px}
           .clsFraP{width:45px;height:45px;background-color:#eee}
           .clsFraC{width:25px;height:25px;background-color:#ddd}
    </style>
    <script type="text/javascript">
            $(function(){ //匹配后代元素
              $("#divMid").css("display","block");
              $("div span").css("display","block");
            })
            $(function(){ //匹配子元素
              $("#divMid").css("display","block");
              $("div>span").css("display","block");
            })
            $(function(){ //匹配后面元素
              $("#divMid + div").css("display","block");
              $("#divMid").next().css("display","block");
            })*/
            $(function(){ //匹配所有后面元素
              $("#divMid ~ div").css("display","block");
              $("#divMid").nextAll().css("display","block");
            })
            $(function(){ //匹配所有相邻元素
              $("#divMid").siblings("div").css("display","block");
            })
    </script>
</head>
<body>
    <div class="clsFraA">Left</div>
    <div class="clsFraA" id="divMid">
         <span class="clsFraP" id="Span1">
              <span class="clsFraC" id="Span2"></span>
         </span>
    </div>
    <div class="clsFraA">Right_1</div>
    <div class="clsFraA">Right_2</div>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-4所示。

表 2-4　页面执行效果

关键代码	功能描述	页面效果
$("div span") .css("display","block");	显示 中所有的 标记	
$("div>span") .css("display","block")	显示 中子 标记	
$("#divMid + div") .css("display","block");	显示 ID 为 divMid 元素后的下一个	
$("#divMid ~ div") .css("display","block");	显示 ID 为 divMid 元素后的所有	
$("#divMid").siblings("div") .css("display","block");	显示 ID 为 divMid 元素的所有相邻	

（4）代码分析

siblings()方法与选择器prev～siblings区别在于，前者获取全部的相邻元素，不分前后，而后者仅获取标记后面全部相邻元素，不能获取前面部分。

2.2.3　简单过滤选择器

过滤选择器根据某类过滤规则进行元素的匹配，书写时都以冒号（:）开头；简单过滤选择器是过滤选择器中使用最广泛的一种，其详细说明如表2-5所示。

表 2-5　简单过滤选择器语法

选择器	功能描述	返回值
first() 或 :first	获取第一个元素	单个元素
last() 或 :last	获取最后一个元素	单个元素
:not(selector)	获取除给定选择器外的所有元素	元素集合
:even	获取所有索引值为偶数的元素，索引号从 0 开始	元素集合
:odd	获取所有索引值为奇数的元素，索引号从 0 开始	元素集合
:eq(index)	获取指定索引值的元素，索引号从 0 开始	单个元素
:gt(index)	获取所有大于给定索引值的元素，索引号从 0 开始	元素集合
:lt(index)	获取所有小于给定索引值的元素，索引号从 0 开始	元素集合
:header	获取所有标题类型的元素，如 h1、h2……	元素集合
:animated	获取正在执行动画效果的元素	元素集合

下面通过示例2-5来介绍如何通过过滤选择器定位DOM元素的方法。

示例2-5　使用jQuery基本过滤选择器选择元素

（1）功能描述

在页面中设置一个＜h1＞标记用于显示主题，创建＜ul＞标记并在其中放置四个＜li＞，再创建一个＜span＞标记，用于执行动画效

果。通过简单过滤选择器获取元素，将选中的元素改变其类名称，从而突出其被选中的状态。

（2）实现代码

新建一个HTML文件2-5.html，加入如代码清单2-5所示的代码。

代码清单 2-5　使用 jQuery 基本过滤选择器选择元素

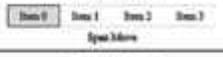
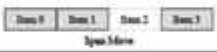
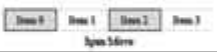
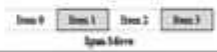
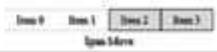
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 基本过滤选择器 </title>
    <script language="javascript" type="text/javascript"
           src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           div{width:241px;height:83px;border:solid 1px #eee}
           h1{font-size:13px}
           ul{list-style-type:none;padding:0px}
           .DefClass,.NotClass{height:23px;width:60px;
                    line-height:23px;float:left;
                    border-top:solid 1px #eee;border-bottom:solid 1px #eee}
           .GetFocus{width:58px;border:solid 1px #666;
                    background-color:#eee}
           #spnMove{width:238px;height:23px;line-height:23px;}
    </style>
    <script type="text/javascript">
           $(function(){ // 增加第一个元素的类别
             $("li:first").addClass("GetFocus");
           })
           $(function(){ // 增加最后一个元素的类别
             $("li:last").addClass("GetFocus");
           })
           $(function(){ // 增加去除所有与给定选择器匹配的元素类别
             $("li:not(.NotClass)").addClass("GetFocus");
           })
           $(function(){ // 增加所有索引值为偶数的元素类别，从 0 开始计数
             $("li:even").addClass("GetFocus");
           })
           $(function(){ // 增加所有索引值为奇数的元素类别，从 0 开始计数
             $("li:odd").addClass("GetFocus");
           })
           $(function(){ // 增加一个给定索引值的元素类别，从 0 开始计数
             $("li:eq(1)").addClass("GetFocus");
           })
           $(function(){ // 增加所有大于给定索引值的元素类别，从 0 开始计数
             $("li:gt(1)").addClass("GetFocus");
           })
           $(function(){ // 增加所有小于给定索引值的元素类别，从 0 开始计数
             $("li:lt(4)").addClass("GetFocus");
           })
           $(function(){ // 增加标题类元素类别
             $("div h1").css("width","238");
             $(":header").addClass("GetFocus");
           })
           $(function(){
             animateIt(); // 增加动画效果元素类别
```

```
            $("#spnMove:animated").addClass("GetFocus");
        })
        function animateIt() { //动画效果
            $("#spnMove").slideToggle("slow", animateIt);
        }
    </script>
</head>
<body>
    <div>
        <h1>基本过滤选择器</h1>
        <ul>
            <li class="DefClass">Item 0</li>
            <li class="DefClass">Item 1</li>
            <li class="NotClass">Item 2</li>
            <li class="DefClass">Item 3</li>
        </ul>
        <span id="spnMove">Span Move</span>
    </div>
</body>
</html>
```

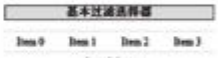
（3）页面效果

执行后的效果如表2-6所示。

表 2-6　页面执行效果

关键代码	功能描述	页面效果
$("li:first") .addClass("GetFocus");	增加第一个元素的类别	
$("li:last") .addClass("GetFocus");	增加最后一个元素的类别	
$("li:not(.NotClass)") .addClass("GetFocus");	增加去除所有与给定选择器匹配的元素类别	
$("li:even") .addClass("GetFocus");	增加所有索引值为偶数的元素类别，从 0 开始计数	
$("li:odd") .addClass("GetFocus");	增加所有索引值为奇数的元素类别，从 0 开始计数	
$("li:eq(1)") .addClass("GetFocus");	增加一个给定索引值的元素类别，从 0 开始计数	
$("li:gt(1)") .addClass("GetFocus");	增加所有大于给定索引值的元素类别，从 0 开始计数	
$("li:lt(4)") .addClass("GetFocus");	增加所有小于给定索引值的元素类别，从 0 开始计数	

（续）

关键代码	功能描述	页面效果
$(":header") .addClass("GetFocus");	增加标题类元素类别	
$("#spnMove:animated") .addClass("GetFocus");	增加动画效果元素类别	

（4）代码分析

选择器animated在捕捉动画效果元素时，先自定义一个动画效果函数animateIt()，然后执行该函数，选择器才能获取动画效果元素，并增加其类别。

2.2.4　内容过滤选择器

内容过滤选择器根据元素中的文字内容或所包含的子元素特征获取元素，其文字内容可以模糊或绝对匹配进行元素定位，其详细说明如表2-7所示。

表 2-7　内容过滤选择器语法

选择器	功能描述	返回值
:contains(text)	获取包含给定文本的元素	元素集合
:empty	获取所有不包含子元素或者文本的空元素	元素集合
:has(selector)	获取含有选择器所匹配的元素	元素集合
:parent	获取含有子元素或者文本的元素	元素集合

下面通过示例2-6来展示在页面中如何通过内容过虑选择器查找DOM元素的方法。

示例2-6　使用jQuery内容过滤选择器选择元素

（1）功能描述

在页面中，根据需要创建四个＜div＞标记，并在其中一个＜div＞中新建一个＜span＞标记，其余＜div＞标记中输入内容（或为空），通过内容过滤选择器获取指定的元素，并显示在页面中。

（2）实现代码

新建一个HTML文件2-6.html，加入如代码清单2-6所示的代码。

代码清单 2-6　使用 jQuery 内容过滤选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 内容过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">


          body{font-size:12px;text-align:center}
          div{float:left;border:solid 1px #ccc;margin:8px;
              width:65px;height:65px;display:none}
          span{float:left;border:solid 1px #ccc;margin:8px;
               width:45px;height:45px;background-color:#eee}
    </style>
    <script type="text/javascript">
           $(function(){ //显示包含给定文本的元素
             $("div:contains('A')").css("display","block");
           })
           $(function(){ //显示所有不包含子元素或者文本的空元素
             $("div:empty").css("display","block");
           })
           $(function(){ //显示含有选择器所匹配的元素
             $("div:has(span)").css("display","block");
           })
           $(function(){ //显示含有子元素或者文本的元素
             $("div:parent").css("display","block");
           })
    </script>
</head>
<body>
    <div>ABCD</div>
    <div><span></span></div>
    <div>EFaH</div>
    <div></div>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-8所示。

表 2-8　页面执行效果

关键代码	功能描述	页面效果
$("div:contains('A')") .css("display","block");	显示包含给定文本“A”的元素	
$("div:empty") .css("display","block");	显示所有不包含子元素或者文本的空元素	
$("div:has(span)") .css("display","block");	显示含有 标记的元素	
$("div:parent") .css("display","block");	显示含有子元素或者文本的元素	

（4）代码分析

在:contains(text)内容过滤选择器中，如果是查找字母，则有大小写的区别。

2.2.5　可见性过滤选择器

可见性过滤选择器根据元素是否可见的特征获取元素，其详细的说明如表2-9所示。

表 2-9　可见性过滤选择器语法

选择器	功能描述	返回值
:hidden	获取所有不可见元素，或者 type 为 hidden 的元素	元素集合
:visible	获取所有的可见元素	元素集合

下面通过示例2-7介绍如何使用可见性过虑选择器锁定DOM元素的方法。

示例2-7　使用jQuery可见性过滤选择器选择元素

（1）功能描述

在页面中，创建一个＜span＞和＜div＞标记，分别设置标记的display属性为"none"和"block"；然后根据可见性过滤选择器显示页面元素。

（2）实现代码

新建一个HTML文件2-7.html，加入如代码清单2-7所示的代码。

代码清单 2-7　使用 jQuery 可见性过滤选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 可见性过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
          body{font-size:12px;text-align:center}
          div,span{float:left;border:solid 1px #ccc;
                   margin:8px;width:65px;height:65px}
          .GetFocus{border:solid 1px #666;background-color:#eee}
    </style>
    <script type="text/javascript">
          $(function(){ //增加所有可见元素类别
            $("span:hidden").show();
            $("div:visible").addClass("GetFocus");
          })*/
          $(function(){ //增加所有不可见元素类别
            $("span:hidden").show().addClass("GetFocus");
          })
    </script>
</head>
<body>
    <span style="display:none">Hidden</span>
    <div>Visible</div>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-10所示。

表 2-10　页面执行效果

关键代码	功能描述	页面效果
$("div:visible") .addClass("GetFocus");	增加所有可见元素类别	
$("span:hidden") .show().addClass("GetFocus");	增加所有不可见元素类别	

（4）代码分析

":hidden"选择器所选择的不仅包括样式为display:none所有元素，而且还包括属性type="hidden"和样式为visibility:hidden的所

有元素。

2.2.6　属性过滤选择器

属性过滤选择器根据元素的某个属性获取元素，如ID号或匹配属性值的内容，并以“[”号开始、以“]”号结束。其详细的说明如表2-11所示。

表 2-11　属性过滤选择器语法

选择器	功能描述	返回值
[attribute]	获取包含给定属性的元素	元素集合
[attribute=value]	获取等于给定的属性是某个特定值的元素	元素集合
[attribute!=value]	获取不等于给定的属性是某个特定值的元素	元素集合
[attribute^=value]	获取给定的属性是以某些值开始的元素	元素集合
[attribute$=value]	获取给定的属性是以某些值结尾的元素	元素集合
[attribute*=value]	获取给定的属性是以包含某些值的元素	元素集合
[selector1][selector2][selectorN]	获取满足多个条件的复合属性的元素	元素集合

下面通过示例2-8介绍使用属性过滤选择器获取DOM元素的方法。

示例2-8　使用jQuery属性过滤选择器选择元素

（1）功能描述

在页面中增加四个＜div＞标记，分别设置不同的ID和Title属性值，然后通过属性过滤选择器获取所指定的元素集合，并显示在页面中。

（2）实现代码

新建一个HTML文件2-8.html，加入如代码清单2-8所示的代码。

代码清单 2-8　使用 jQuery 属性过滤选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 属性过滤选择器 </title>
    <script language="javascript" type="text/javascript"


            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
          body{font-size:12px;text-align:center}
          div{float:left;border:solid 1px #ccc;margin:8px;
             width:65px;height:65px;display:none}
    </style>
    <script type="text/javascript">
          $(function(){ //显示所有含有 id 属性的元素
            $("div[id]").show(3000);
          })
          $(function(){ //显示所有属性 title 的值是 "A" 的元素
            $("div[title='A']").show(3000);
          })
          $(function(){ //显示所有属性 title 的值不是 "A" 的元素
            $("div[title!='A']").show(3000);
          })
          $(function(){ //显示所有属性 title 的值以 "A" 开始的元素
            $("div[title^='A']").show(3000);
          })
          $(function(){ //显示所有属性 title 的值以 "C" 结束的元素
            $("div[title$='C']").show(3000);
          })
          $(function(){ //显示所有属性 title 的值中含有 "B" 的元素
            $("div[title*='B']").show(3000);
          })
          $(function(){ //显示所有属性 title 的值中含有 "B"
                        //且属性 id 的值是 "divAB" 的元素
            $("div[id='divAB'][title*='B']").show(3000);
          })
    </script>
</head>
<body>
    <div id="divID">ID</div>
    <div title="A">Title A</div>
    <div id="divAB" title="AB">ID <br />Title AB</div>
    <div title="ABC">Title ABC</div>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-12所示。

表 2-12　页面执行效果

关键代码	功能描述	页面效果
$("div[id]") .show(3000);	显示所有含有 id 属性的元素	
$("div[title='A']") .show(3000);	显示所有属性 title 的值是 "A" 的元素	
$("div[title!='A']") .show(3000);	显示所有属性 title 的值不是 "A" 的元素	

（续）

关键代码	功能描述	页面效果
$("div[title^='A']") .show(3000);	显示所有属性 title 的值以 "A" 开始的元素	
$("div[titleS='C']") .show(3000);	显示所有属性 title 的值以 "C" 结束的元素	
$("div[title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 的元素	
$("div[id='divAB'][title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 且属性 id 的值是 "divAB" 的元素	

（4）代码分析

show()是jQuery库中的一个显示元素函数，括号中的参数表示显示时间，单位是毫秒，show(3000)表示用3 000毫秒显示。

2.2.7　子元素过滤选择器

在页面开发过程中，常常遇到突出指定某行的需求。虽然使用基本过滤选择器":eq(index)"可实现单个表格的显示，但不能满足大量数据和多个表格的选择需求。为了实现这样的功能，jQuery中可以通过子元素过滤选择器轻松获取所有父元素中指定的某个元素。其详细的说明如表2-13所示。

表 2-13　子元素过滤选择器语法

选择器	功能描述	返回值
:nth-child(eq|even|odd|index)	获取每个父元素下的特定位置元素，索引号从 1 开始	元素集合
:first-child	获取每个父元素下的第一个子元素	元素集合
:last-child	获取每个父元素下的最后一个子元素	元素集合
:only-child	获取每个父元素下的仅有一个子元素	元素集合

下面通过示例2-9来演示使用子元素过滤选择器获取元素的过程。

示例2-9　使用jQuery子元素过滤选择器选择元素

（1）功能描述

在页面中创建三个＜ul＞标记，前两个标记中设置四个＜li＞，后一个标记中设置一个＜li＞。通过子元素过滤选择器获取指定的页面元素，并改变其选择后的状态，显示在页面中。

（2）实现代码

新建一个HTML文件2-9.html，加入如代码清单2-9所示的代码。

代码清单 2-9　使用 jQuery 子元素过滤选择器选择元素

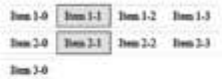
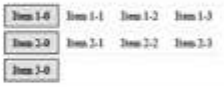
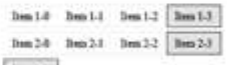
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 子元素过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           ul{width:241px;list-style-type:none;padding:0px}
           ul li{height:23px;width:60px;line-height:23px;
                 float:left;border-top:solid 1px #eee;
           border-bottom:solid 1px #eee;margin-bottom:5px}
           .GetFocus{width:58px;border:solid 1px #666;
                 background-color:#eee}
    </style>
    <script type="text/javascript">
           $(function(){ //增加每个父元素下的第二个子元素类别
             $("li:nth-child(2)").addClass("GetFocus");
           })
           /*$(function(){ //增加每个父元素下的第一个子元素类别
             $("li:first-child").addClass("GetFocus");
           })
           $(function(){ //增加每个父元素下的最后一个子元素类别
             $("li:last-child").addClass("GetFocus");
           })
           $(function(){ //增加每个父元素下只有一个子元素类别
             $("li:only-child").addClass("GetFocus");
           })*/
    </script>
</head>
<body>
     <ul>
        <li>Item 1-0</li>
        <li>Item 1-1</li>
        <li>Item 1-2</li>
        <li>Item 1-3</li>
     </ul>
     <ul>
        <li>Item 2-0</li>
        <li>Item 2-1</li>
        <li>Item 2-2</li>
        <li>Item 2-3</li>
     </ul>
     <ul>
        <li>Item 3-0</li>
     </ul>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-14所示。

表 2-14　页面执行效果

关键代码	功能描述	页面效果
$("li:nth-child(2)") .addClass("GetFocus");	增加每个父元素下的第二个子元素类别	
$("li:first-child") .addClass("GetFocus");	增加每个父元素下的第一个子元素类别	
$("li:last-child") .addClass("GetFocus");	增加每个父元素下的最后一个子元素类别	
$("li:only-child") .addClass("GetFocus");	增加每个父元素下的仅有一个子元素类别	

2.2.8　表单对象属性过滤选择器

表单对象属性过滤选择器通过表单中的某对象属性特征获取该类元素，如enabled、disabled、checked、selected属性。其详细的说明如表2-15所示。

表 2-15　表单对象属性过滤选择器语法

选择器	功能描述	返回值
:enabled	获取表单中所有属性为可用的元素	元素集合
:disabled	获取表单中所有属性为不可用的元素	元素集合
:checked	获取表单中所有被选中的元素	元素集合
:selected	获取表单中所有被选中 option 的元素	元素集合

下面通过示例2-10来介绍通过表单对象属性过滤选择器获取表单对象的方法。

示例2-10　使用jQuery表单对象属性过滤选择器获取表单对象

（1）功能描述

在一个表单中创建两个文本框对象，一个属性设置为enabled，另一个属性设置为disabled；再放置两个复选框对象，一个设置成被选中状态，另一个设置成未选中状态；同时新建一个列表框对象，并选中其中两项，通过表单对象属性过滤选择器获取某指定元素，并处理该元素。

（2）实现代码

新建一个HTML文件2-10.html，加入如代码清单2-10所示的代码。

代码清单 2-10　使用 jQuery 表单对象属性过滤选择器选择元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>使用 jQuery 表单对象属性过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
          body{font-size:12px;text-align:center}
          div{display:none}
          select{height:65px}
          .clsIpt{width:100px;padding:3px}
          .GetFocus{border:solid 1px #666;background-color:#eee}
    </style>
    <script type="text/javascript">
          $(function(){ // 增加表单中所有属性为可用的元素类别
            $("#divA").show(3000);
            $("#form1 input:enabled").addClass("GetFocus");
          })
          $(function(){ // 增加表单中所有属性为不可用的元素类别
            $("#divA").show(3000);
            $("#form1 input:disabled").addClass("GetFocus");
          })
          $(function(){ // 增加表单中所有被选中的元素类别
            $("#divB").show(3000);
            $("#form1 input:checked").addClass("GetFocus");
          })
          $(function(){ // 显示表单中所有被选中 option 的元素内容
            $("#divC").show(3000);
            $("#Span2").html("选中项是: "+
            $("select option:selected").text());
          })
    </script>
</head>
<body>
     <form id="form1" style="width:241px">
          <div id="divA">
              <input type="text" value="可用文本框" class="clsIpt" />
              <input type="text" disabled="disabled"
                     value="不可用文本框" class="clsIpt" />
          </div>
          <div id="divB">
              <input type="checkbox" checked="checked" value="1" /> 选中
              <input type="checkbox" value="0" /> 未选中
          </div>
          <div id="divC">
          <select multiple="multiple">
              <option value="0">Item 0</option>
              <option value="1" selected="selected">
                     Item 1
              </option>
              <option value="2">Item 2</option>
              <option value="3" selected="selected">
                     Item 3
              </option>
          </select>
          <span id="Span2"></span>
          </div>


     </form>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-16所示。

表 2-16　页面执行效果

关键代码	功能描述	页面效果
$("#form1 input:enabled") .addClass("GetFocus");	增加表单中所有属性为可用的元素类别	
$("#form1 input:disabled") .addClass("GetFocus");	增加表单中所有属性为不可用的元素类别	
$("#form1 input:checked") .addClass("GetFocus");	增加表单中所有被选中的元素类别	
$("#Span2").html(" 选中项是："+ $("select option:selected").text());	显示表单中所有被选中 option 的元素内容	

2.2.9　表单选择器

无论是提交还是传递数据，表单在页面中的作用是显而易见的。通过表单进行数据的提交或处理，在前端页面开发中占据重要地位。因此，为了使用户能更加方便地、高效地使用表单，在jQuery选择器中引入表单选择器，该选择器专为表单量身打造，通过它可以在页面中快速定位某表单对象。其详细的说明如表2-17所示。

表 2-17　表单选择器语法

选择器	功能描述	返回值
:input	获取所有 input、textarea、select	元素集合
:text	获取所有单行文本框	元素集合
:password	获取所有密码框	元素集合
:radio	获取所有单选按钮	元素集合
:checkbox	获取所有复选框	元素集合
:submit	获取所有提交按钮	元素集合
:image	获取所有图像域	元素集合
:reset	获取所有重置按钮	元素集合
:button	获取所有按钮	元素集合
:file	获取所有文件域	元素集合

下面通过示例2-13来介绍使用表单选择器直接获取表单对象的方法。

示例2-11　使用jQuery表单过滤选择器获取元素

（1）功能描述

在一个页面表单中，创建11种常用的表单对象，根据表单选择器，先显示所有表单对象的总量，然后显示各种不同类型的表单对象。

（2）实现代码

新建一个HTML文件2-11.html，加入如代码清单2-11所示的代码。

代码清单 2-11　使用 jQuery 表单过滤选择器获取元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 表单过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           form{width:24lpx}
           textarea,select,button,input,span{display:none}
           .btn {border:#666 1px solid;padding:2px;width:60px;
                 filter: progid:DXImageTransform.Microsoft.
                 Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
           .txt{border:#666 1px solid;padding:3px}
           .img{padding:2px;border:solid 1px #ccc}
           .div{border:solid 1px #ccc;
               background-color:#eee;padding:5px}
    </style>
    <script type="text/javascript">
           $(function(){ //显示 Input 类型元素的总数量
             $("#form1 div").html(" 表单共找出 Input 类型元素 :"+
             $("#form1 :input").length);
             $("#form1 div").addClass("div");
           })
           $(function(){ //显示所有文本框对象
             $("#form1 :text").show(3000);
           })
           $(function(){ //显示所有密码框对象
             $("#form1 :password").show(3000);
           })
           $(function(){ //显示所有单选按钮对象
             $("#form1 :radio").show(3000);
             $("#form1 #Span1").show(3000);
           })
           $(function(){  //显示所有复选框对象
             $("#form1 :checkbox").show(3000);
             $("#form1 #Span2").show(3000);
           })
           $(function(){ //显示所有提交按钮对象
             $("#form1 :submit").show(3000);
           })
           $(function(){ //显示所有图片域对象
             $("#form1 :image").show(3000);
           })
           $(function(){ //显示所有重置按钮对象
             $("#form1 :reset").show(3000);
           })
           $(function(){ //显示所有按钮对象
```

```
            $("#form1 :button").show(3000);
        })
        $(function(){ //显示所有文件域对象
            $("#form1 :file").show(3000);
        })
    </script>
</head>
<body>
    <form id="form1">
        <textarea class="txt"> TextArea</textarea>
        <select><option value="0"> Item 0</option></select>
        <input type="text" value="Text" class="txt"/>
        <input type="password" value="PassWord" class="txt"/>
        <input type="radio" /><span id="Span1"> Radio</span>
        <input type="checkbox" />
        <span id="Span2"> CheckBox</span>
        <input type="submit" value="Submit" class="btn"/>
        <input type="image" title="Image"
            src="Images/logo.gif" class="img"/>
        <input type="reset" value="Reset" class="btn"/>
        <input type="button" value="Button" class="btn"/>
        <input type="file" title="File" class="txt"/>
        <div id="divShow"></div>
    </form>
</body>
</html>
```

（3）页面效果

执行后的效果如表2-18所示。

表 2-18　页面执行效果

关键代码	功能描述	页面效果
$("#form1 div").html(" 表单共找出 Input 类型元素 :"+ $("#form1 :input").length);	显示 Input 类型元素的总数量	表单共找出 Input 类型元素:11
$("#form1 :text").show(3000);	显示所有文本框对象	Text
$("#form1 :password").show(3000);	显示所有密码框对象	••••••••
$("#form1 :radio").show(3000);	显示所有单选按钮对象	Radio
$("#form1 :checkbox").show(3000);	显示所有复选框对象	CheckBox
$("#form1 :submit").show(3000);	显示所有提交按钮对象	Submit
$("#form1 :image").show(3000);	显示所有图片域对象	jQuery

（续）

关键代码	功能描述	页面效果
$("#form1 :reset").show(3000);	显示所有重置按钮对象	
$("#form1 :button").show(3000);	显示所有按钮对象	
$("#form1 :file").show(3000);	显示所有文件域对象	

2.3　综合案例分析——导航条在项目中的应用

2.3.1　需求分析

本案例的需求主要有以下两点：

1）在页面中创建一个导航条，单击标题时，可以伸缩导航条的内容，标题中的提示图片也随之改变。

2）单击“简化”链接时，隐藏指定的内容，并将“简化”字样改为“更多”；单击“更多”链接时，返回初始状态，并改变指定显示元素的背景色。

2.3.2　界面效果

案例实现的界面效果如图2-4、图2-5所示。

图　2-4　单击导航条标题前后的界面

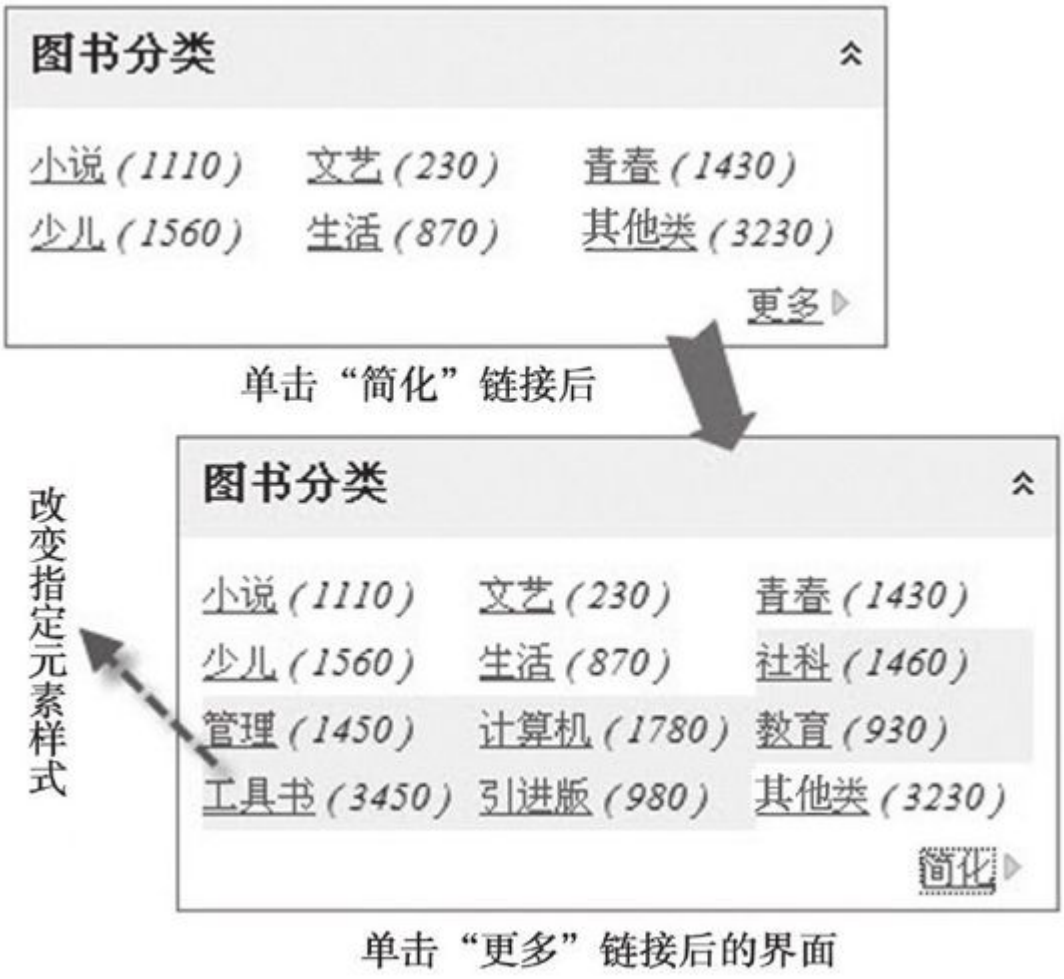

图书分类
小说 (1110)
文艺 (230)
青春 (1430)
少儿 (1560)
生活 (870)
其他类 (3230)
更多
单击“简化”链接后
图书分类
小说 (1110)
文艺 (230)
青春 (1430)
少儿 (1560)
生活 (870)
社科 (1460)
管理 (1450)
计算机 (1780)
教育 (930)
工具书 (3450)
引进版 (980)
其他类 (3230)
简化
改变指定元素样式
单击“更多”链接后的界面

图　2-5　“简化”和“更多”界面

2.3.3　功能实现

在项目中，新建一个HTML文件htmNav.html，加入如代码清单2-12所示的代码。

代码清单 2-12　导航条在项目中的应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>导航条在项目中的应用</title>
    <script language="javascript" type="text/javascript"
          src="Jscript/jquery-1.8.2.min.js"></script>
    <style>
        body{font-size:13px}
        #divFrame{border:solid 1px #666;
        width:301px;overflow:hidden}
        #divFrame .clsHead{background-color:#eee;padding:8px;
                 height:18px;cursor:hand}
        #divFrame .clsHead h3{padding:0px;margin:0px;float:left}
        #divFrame .clsHead span{float:right;margin-top:3px}
        #divFrame .clsContent{padding:8px}
        #divFrame .clsContent ul {list-style-type:none;
                 margin:0px;padding:0px}
        #divFrame .clsContent ul li{ float:left;
                 width:95px;height:23px;line-height:23px}
        #divFrame .clsBot{float:right;padding-top:5px;
                 padding-bottom:5px}
        .GetFocus{background-color:#eee}
    </style>
    <script type="text/javascript">
          $(function(){ //页面加载事件
            $(".clsHead").click(function(){          //图片单击事件
              if($(".clsContent").is(":visible")){ //如果内容可见
                 $(".clsHead span img")
                 .attr("src","Images/a1.gif"); //改变图片
                 //隐藏内容
                 $(".clsContent").css("display","none");
              }else{
                 $(".clsHead span img")
                 .attr("src","Images/a2.gif"); //改变图片
                 //显示内容
                 $(".clsContent").css("display","block");
              }
            });

            $(".clsBot > a").click(function(){ //热点链接单击事件
              //如果内容为"简化"字样
              if($(".clsBot > a").text()=="简化"){
                 //隐藏 index 号大于 4 且不是最后一项的元素
                 $("ul li:gt(4):not(:last)").hide();
                 //将字符内容更改为"更多"
                 $(".clsBot > a").text("更多");
              }else{
                 $("ul li:gt(4):not(:last)")
```

```
                    .show()
                    .addClass("GetFocus"); //显示所选元素且增加样式
                    //将字符内容更改为"简化"
                    $(".clsBot > a").text("简化");
                }
            });
        });
    </script>
</head>
<body>
    <div id="divFrame">
        <div class="clsHead">
            <h3>图书分类</h3>
            <span><img src="Images/a2.gif" alt=""/></span>
        </div>
        <div class="clsContent">
         <ul>
            <li><a href="#">小说</a><i>(1110)</i></li>
            <li><a href="#">文艺</a><i>(230)</i></li>
            <li><a href="#">青春</a><i>(1430)</i></li>
            <li><a href="#">少儿</a><i>(1560)</i></li>
            <li><a href="#">生活</a><i>(870)</i></li>
            <li><a href="#">社科</a><i>(1460)</i></li>
            <li><a href="#">管理</a><i>(1450)</i></li>
            <li><a href="#">计算机</a><i>(1780)</i></li>
            <li><a href="#">教育</a><i>(930)</i></li>
            <li><a href="#">工具书</a><i>(3450)</i></li>
            <li><a href="#">引进版</a><i>(980)</i></li>
            <li><a href="#">其他类</a><i>(3230)</i></li>
         </ul>
        <div class="clsBot"><a href="#">简化</a>
            <img src="Images/ a5.gif" alt=""/>
        </div>
        </div>
    </div>
</body>
</html>
```

2.3.4 代码分析

在页面代码中，首先通过如下代码获取类名称为"clsContent"的元素集合，并实现其内容的显示或隐藏：

```
$(".clsContent").css("display","none");
```

同时，通过下面代码变换图片：

```
$(".clsHead span img").attr("src","Images/a1.gif");
```

其中，".clsHead span img"表示获取类型clsHead中＜span＞下的＜img＞标记，即图片元素；attr(key,value)是jQuery中一个设置元素属性的函数，其功能是为所匹配的元素设置属性值，key是属性名称，value是属性值或内容。因此，此行代码的功能是，获取图片元素并改变其图片来源。

为了能够实现单击标题后内容可以伸缩的功能，首先通过如下代码，检测当前内容的隐藏或显示状态：

```
if($(".clsContent").is(":visible"))
```

其中"$(".clsContent")"用于获取内容元素，"is"是判断，":visible"表示是否可见，此行代码用于判断内容元素是否可

见，它返回一个布尔值，如果为true，则执行if后面的语句块，否则执行else后面的语句块。

在超级链接单击事件中，通过下面的代码检测单击的是“简化”还是“更多”字样。

```
if($(".clsBot > a").text()==" 简化 ")
```

其中"$(".clsBot＞a")"用于获取超级链接元素，text()是jQuery中一个获取元素内容的函数。此行代码的意思为，判断超级链接元素的内容是否为“简化”字样，然后根据检测结果，执行不同的语句块。

在代码中，通过如下的代码实现指定内容的隐藏：

```
$("ul li:gt(4):not(:last)").hide();
```

其中"ul li"用于获取元素，":gt(4)"和":not(:last)"分别为两个并列的过滤选择条件，前者表示Index号大于4，后者表示不是最后一个元素，二者组合成一个过滤条件，即选Index号大于4并且不是最后一个的元素集合；hide()是jQuery中一个隐藏元素的函数。此行代码的意思为，将通过过滤选择器获取的元素隐藏。

2.4　本章小结

选择器是jQuery的核心。本章通过将实例与理论相结合，从选择器的优势和类别入手，详细介绍了jQuery中的选择器语法和使用技巧，最后通过一个简单通用导航条的功能开发，进一步巩固前面章节所学的知识，并为第3章的深入学习创造条件。

第3章　jQuery操作DOM

本章内容

DOM树状模型

元素属性操作

获取和设置元素

元素样式操作

页面元素操作

综合案例分析——数据删除和图片预览在项目中的应用

本章小结

DOM为文档提供了一种结构化表示方法，通过该方法可以改变文档的内容和展示形式。在实际运用中，DOM更像是桥梁，通过它可以实现跨平台、跨语言的标准访问。在本章中，我们将详细介绍如何使用jQuery操作或控制DOM中的各种元素或对象。

3.1　DOM树状模型

与DOM（Document Object Model，文档对象模型）密不可分的是JavaScript脚本技术，DOM在客户端的应用也是基于该技术，通过该技术我们可以很方便地访问、检索、操作文档中的任何一个元素。因此，学好JavaScript脚本技术，是掌握DOM对象的一个重要条件。

Document即文档，当我们创建一个页面并加载到Web浏览器时，DOM模型根据该页面的内容创建一个文档文件。

Object即对象，是指具有独立特性的一组数据集合，例如，我们把新建的页面文档称之为文档对象，与对象相关联的特征称之为对象属性，访问对象的函数称之为对象方法。

Model即模型，在页面文档中，通过树状模型展示页面的元素和内容，其展示的方式则是通过节点（node）来实现的。下面通过示例3-1加以说明。

示例3-1　创建一个DOM页面文档

（1）功能描述

完成一个DOM页面文档的创建。

（2）实现代码

新建一个HTML页面3-1.html，加入如代码清单3-1所示的代码。

代码清单 3-1　创建一个 DOM 页面文档

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>DOM 树状文档 </title>
    <style>
      body{font-size:13px}
      table,div,p,ul{width:280px;border:solid 1px #666;
            margin:10px 0px 10px 0px;
            padding:0px;background-color:#eee}
    </style>
</head>
<body>
     <table>
       <tr><td>Td1</td></tr>
       <tr><td>Td2</td></tr>
     </table>
     <div>Div</div>
     <p>P</p>
     <div><span>Span</span></div>


     <ul>
       <li>Li1</li>
       <li>Li2</li>
     </ul>
</body>
</html>
```

（3）页面效果

页面执行后的效果如图3-1所示。

图　3-1　DOM树状页面效果

根据上述页面文档创建的DOM树状结构如图3-2所示。

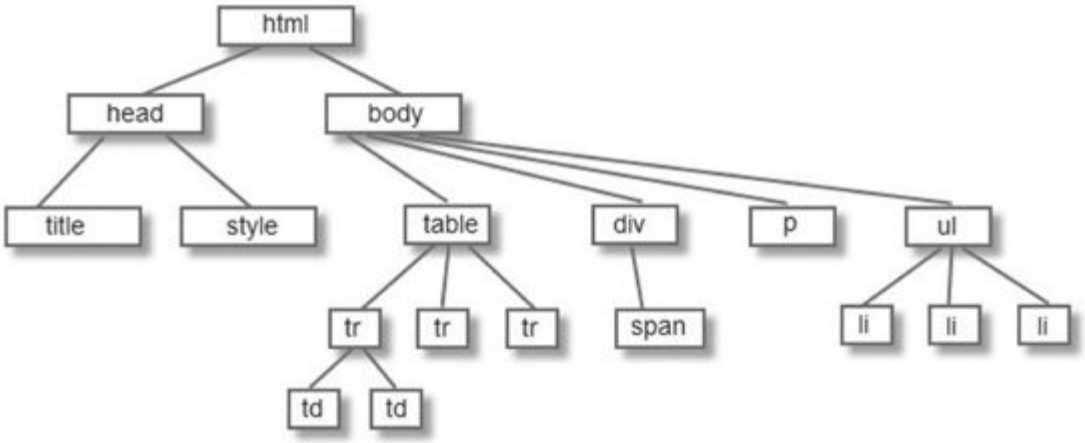


图　3-2　DOM树状模型

在访问页面时，需要与页面中的元素进行交互式的操作。在操作中，元素的访问是最频繁、最常用的，主要包括对元素属性、内容、值、CSS的操作。下面通过实例详细介绍这些操作的使用方法和技巧。

3.2　元素属性操作

在jQuery中，可以对元素的属性执行获取、设置、删除的操作，通过attr()方法可以对元素属性执行获取和设置操作，而removeAttr()方法则可以轻松删除某一指定的属性。

3.2.1　获取元素的属性

获取元素属性的语法格式如下：

```
attr(name)
```

其中，参数name表示属性的名称。示例3-2将介绍如何通过调用attr()方法，以元素属性名称为参数的方式，来获取元素的属性的过程。

示例3-2　使用attr(name)方法获取元素的属性

（1）功能描述

在一个页面中，创建一个＜img＞标记，通过jQuery中的attr()方法获取标记的src和title属性值，并显示在页面中。

（2）实现代码

新建一个HTML文件3-2.html,加入如代码清单3-2所示的代码。

代码清单 3-2　获取元素的属性

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>获取元素的属性</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           div{float:left;padding-left:10px}
           img{border:solid 1px #ccc;padding:3px;float:left}
    </style>
    <script type="text/javascript">
        $(function() {
            var strAlt = $("img").attr("src");                  // 属性值一
            strAlt += "<br/><br/>" + $("img").attr("title");    // 属性值二
            $("#divAlt").html(strAlt);                          // 显示在页面中
        })
    </script>
</head>
<body>
     <img alt="" title="这是一幅风景画"
          src="Images/img01.jpg" />
     <div id="divAlt"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-3所示。


获取元素的属性 - Windows Internet Explorer
http://localhost/Ch3/3-2.html
收藏夹
获取元素的属性
Images/img01.jpg
这是一幅风景画
属性src的值
属性title的值
这是一幅风景画
Internet
100%

图　3-3　获取元素的属性

3.2.2　设置元素的属性

在页面中，attr()方法不仅可以获取元素的属性值，还可以设置元素的属性，其设置属性语法格式如下所示：

```
attr(key, value)
```

其中，参数key表示属性的名称，value表示属性的值。如果要设置多个属性，也可以通过attr()方法实现，其语法格式如下所示：

```
attr({key0:value0, key1:value1})
```

下面通过示例3-3来说明如何通过attr(name,value)的方式设置元素的属性。

示例3-3　使用attr(name,value)方法设置元素的属性

（1）功能描述

在页面中创建一个＜img＞标记，在页面加载时，通过jQuery中的attr()方法设置该标记的img和title属性值，并显示在页面中。

（2）实现代码

新建一个HTML文件3-3.html，加入如代码清单3-3所示的代码。

代码清单 3-3　设置元素的属性

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>设置元素的属性</title>


   <script type="text/javascript"
           src="Jscript/jquery-1.8.2.min.js">
   </script>
   <style type="text/css">
          body{font-size:12px}
          .clsSpn{float:left;padding-top:10px;padding-left:10px}
          .clsImg{border:solid 1px #ccc;padding:3px;float:left}
   </style>
   <script type="text/javascript">
       $(function() {
           $("img").attr("src", "Images/img01.jpg"); //设置 src 属性
           $("img").attr("title", "这是一幅风景画");   //设置 title 属性
           $("img").attr({ src: "Images/img02.jpg",
                           title: "这是一幅风景画" }); //同时设置两个属性
           $("img").addClass("clsImg");               //增加样式
           $("span").html("加载完毕");                //显示加载状态
       })
   </script>
</head>
<body>
    <img alt="" src="Images/img03.gif" style="float:left" />
    <span class="clsSpn">正在加载图片...</span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-4所示。

图　3-4　设置元素的属性

attr()方法还可以绑定一个function()函数，通过该函数返回的值作为元素的属性值，其语法格式如下所示：

```
attr(key, function(index))
```

其中，参数index为当前元素的索引号，整个函数返回一个字符串作为元素的属性值。下面我们通过示例3-4详细介绍。

示例3-4　使用function()函数随机展示图片

（1）功能描述

在页面中，通过function()随机函数返回一个随机数字，依据该数字组成一个字符串作为标记＜img＞属性src的值。由于是随机数字，每次获取的src属性值都不一样，从而实现在页面中随机展示图片的效果。

（2）实现代码

新建一个HTML文件3-4.html，加入如代码清单3-4所示的代码。

代码清单 3-4　随机展示图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>设置元素的属性（二）</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           .clsSpn{float:left;padding-top:10px;padding-left:10px}
           .clsImg{border:solid 1px #ccc;padding:3px;float:left}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").attr("src", function() {
               return "Images/img0" +
               Math.floor(Math.random() * 2 + 1) + ".jpg" }); // 设置 src 属性
            $("img").attr("title", "这是一幅风景画 ");          // 设置 title 属性
            $("img").addClass("clsImg");                       // 增加样式
        })
    </script>
</head>
<body>
     <img alt="" src="Images/img03.gif" style="float:left" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-5所示。

图　3-5　随机展示图片

3.2.3　删除元素的属性

jQuery中通过attr()方法设置元素的属性后，使用removeAttr()方法可以将元素的属性删除，其使用的语法格式为：

```
removeAttr(name)
```

其中，参数name为元素属性的名称。

例如，可以通过如下代码删除标记＜img＞中的src属性：

```
$("img").removeAttr("src");
```

3.3　获取和设置元素

3.3.1　获取和设置元素内容

在jQuery中，操作元素内容的方法包括html()和text()。前者与JavaScript中的innerHTML属性类似，即获取或设置元素的HMTL内容；后者类似于JavaScript中的innerText属性，即获取或设置元素的文本内容。二者的格式与功能的区别如表3-1所示。

表 3-1　html() 和 text() 方法的区别

语法格式	参数说明	功能描述
html()	无参数	用于获取元素的 HTML 内容
html(val)	val 参数为元素的 HTML 内容	用于设置元素的 HTML 内容
text()	无参数	用于获取元素的文本内容
text(val)	val 参数为元素的文本内容	用于设置元素的文本内容

html()方法仅支持XHTML的文档，不能用于XML文档，而text()既支持HTML文档，也支持XML文档。

示例3-5讲解了如何通过调用html()方法，以带参数或不带参数的方式，来设置和获取元素的内容。

示例3-5　使用html()和text()方法设置和获取元素的内容

（1）功能描述

在页面中，用html()和text()方法获取div标记中的内容，将内容分别作为html(val)和text(val)的参数，分别设置元素的内容，并将结果显示在页面中。

（2）实现代码

新建一个HTML文件3-5.html，加入如代码清单3-5所示的代码。

代码清单 3-5　设置和获取元素的内容

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>获取或设置元素的内容</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:15px;text-align:center}
           div{border:solid 1px #666; padding:5px;width:220px;margin:5px}
    </style>
    <script type="text/javascript">
        $(function() {
              var strHTML = $("#divShow").html();// 获取 HTML 内容


              var strText = $("#divShow").text();// 获取文本内容
              $("#divHTML").html(strHTML);        // 设置 HTML 内容
              $("#divText").text(strText);        // 设置文本内容
        })
    </script>
</head>
<body>
    <div id="divShow"><b><i>Write Less Do More</i></b></div>
    <div id="divHTML"></div>
    <div id="divText"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-6所示。

图　3-6　获取和设置元素的内容

3.3.2　获取和设置元素值

在jQuery中，要获取元素的值通过val()方法实现，其语法格式如下所示：

```
val(val)
```

其中，如果不带参数val，是获取某元素的值；反之，则是将参数val的值赋给某元素，即设置元素的值。该方法常用于表单中获取或设置对象的值。

另外，通过val()方法还可以获取select标记中的多个选项值，其语法格式如下所示：

```
val().join(",")
```

示例3-6演示了如何通过调用val()方法设置和获取元素的值。

示例3-6　使用val()方法设置和获取元素的值

（1）功能描述

在页面中，创建一个可以多选的select标记，设置该标记的change事件，当按Ctrl键选择多项时，通过p标记将获取的选项值显示在页面中；另外，创建一个文本框元素，设置该文本框的change和focus事件，当文本框获得焦点时，清空文本框中的内容；当在文本框输入字符时，通过另外一个p标记将所获取文本的值即时显示在页面中。

（2）实现代码

新建一个HTML文件3-6.html,加入如代码清单3-6所示的代码。

代码清单 3-6　设置和获取元素的值

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>获取或设置元素的值</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:12px;text-align:center}
          div{padding:3px;margin:3px;width:120px;float:left}
          .txt{border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("select").change(function() {     // 设置列表框 change 事件
                // 获取列表框所选中的全部选项的值
                var strSel = $("select").val().join(",");
                $("#p1").html(strSel);          // 显示列表框所选中的全部选项的值
            })
            $("input").change(function() {      // 设置文本框 focus 事件
                var strTxt = $("input").val(); // 获取文本框的值
                $("#p2").html(strTxt);          // 显示文本框所输入的值
            })
            $("input").focus(function() {       // 设置文本框 focus 事件
                $("input").val("");             // 清空文本框的值
            })
        })
    </script>
</head>
<body>
     <div>
         <select multiple="multiple"
                 style="height:96px;width:85px">
                 <option value="1">Item 1</option>
                 <option value="2">Item 2</option>
                 <option value="3">Item 3</option>
                 <option value="4">Item 4</option>
                 <option value="5">Item 5</option>
                 <option value="6">Item 6</option>


         </select>
         <p id="p1"></p>
     </div>
     <div>
         <input type="text" class="txt"/>
         <p id="p2"></p>
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-7所示。

图　3-7　获取和设置元素的值

（4）代码分析

在val（val）方法中，如果有参数，其参数还可以是数组的形式，即val(array)，其作用是设置元素被选中。因此$(":radio").val(["radio2","radio3"])代码的意思为：Value值为radio2和radio3的单选框被选中。

3.4　元素样式操作

在页面中，元素样式的操作包含：直接设置样式、增加CSS类别、类别切换、删除类别几部分。下面通过示例介绍其使用的语法和方法。

3.4.1　直接设置元素样式值

在jQuery中，可以通过css()方法为某个指定的元素设置样式值，其语法格式如下所示：

```
css(name, value)
```

其中name为样式名称，value为样式的值。

示例3-7演示了如何调用css(name,value)方法直接设置元素的值。

示例3-7　使用CSS()方法直接设置元素样式值

（1）功能说明

在页面中，创建一个p标记，单击该元素时，其中的文字加粗、斜体及增加背景色。

（2）实现代码

新建一个HTML文件3-7.html,加入如代码清单3-7所示的代码。

代码清单 3-7　直接设置元素样式值

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>直接设置元素样式值</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:15px}
           p{padding:5px;width:220px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").click(function() {
                $(this).css("font-weight", "bold");      // 字体加粗
                $(this).css("font-style", "italic");     // 斜体
                $(this).css("background-color", "#eee");// 增加背景色
            })
        })
    </script>
</head>
<body>
     <p>Write Less Do More</p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-8所示。

图　3-8　直接设置元素样式值

3.4.2　增加元素CSS类别

通过addClass()方法增加元素类别的名称，其语法格式如下：

```
addClass(class)
```

其中，参数class为类别的名称，也可以增加多个类别的名称，只需要用空格将其隔开即可，其语法格式为：

```
addClass(class0 class1 ...)
```

示例3-8演示了如何通过调用addClass(class0)为页面中的元素增加CSS类别。

示例3-8　使用addClass()方法增加元素CSS类别

（1）功能描述

在页面中，设置两个样式类cls1和cls2，当单击页面中p元素时，增加这两个样式类别，并将改变后的效果显示在页面中。

（2）实现代码

新建一个HTML文件3-8.html,加入如代码清单3-8所示的代码。

代码清单 3-8　增加元素 CSS 类别

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>增加CSS类别</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:15px}
           p{padding:5px;width:220px}
           .cls1{font-weight:bold;font-style:italic}
           .cls2{ border:solid 1px #666;background-color:#eee}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").click(function() {
                  $(this).addClass("cls1 cls2");// 同时新增两个样式类别
            })
        })
    </script>
</head>
<body>
     <p>Write Less Do More</p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-9所示。

图　3-9　增加CSS类别

（4）代码分析

使用addClass()方法仅是追加样式类别，即它还保存原有的类别。例如，原有标记为＜p class="cls0"/＞，执行代码$("p").addClass("cls1 cls2")后，其最后元素类别为＜p class="cls0 cls1 cls2"/＞，仍然保留原有类别cls0，仅是新增了类别cls1和cls2。

3.4.3　切换元素CSS类别

通过toggleClass()方法切换不同的元素类别，其语法格式如下：

```
toggleClass(class)
```

其中，参数class为类别名称，其功能是当元素中含有名称为class的CSS类别时，删除该类别，否则增加一个该名称的CSS类别。

示例3-9演示了通过调用toggleClass()方法切换元素CSS的类别。

示例3-9　使用toggleClass()方法切换元素CSS类别

（1）功能描述

在页面中创建一个图片标记，通过toggleClass()方法实现对该标记中图片的切换显示。

（2）实现代码

新建一个HTML文件3-9.html,加入如代码清单3-9所示的代码。

代码清单 3-9　切换元素 CSS 类别

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>类别切换</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           .clsImg{border:solid 1px #666;padding:3px}
    </style>


   <script type="text/javascript">
       $(function() {
           $("img").click(function() {
                $(this).toggleClass("clsImg"); //切换样式类别
           })
       })
   </script>
</head>
<body>
     <img src="Images/logo.gif" alt="" title="jQuery" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-10所示。

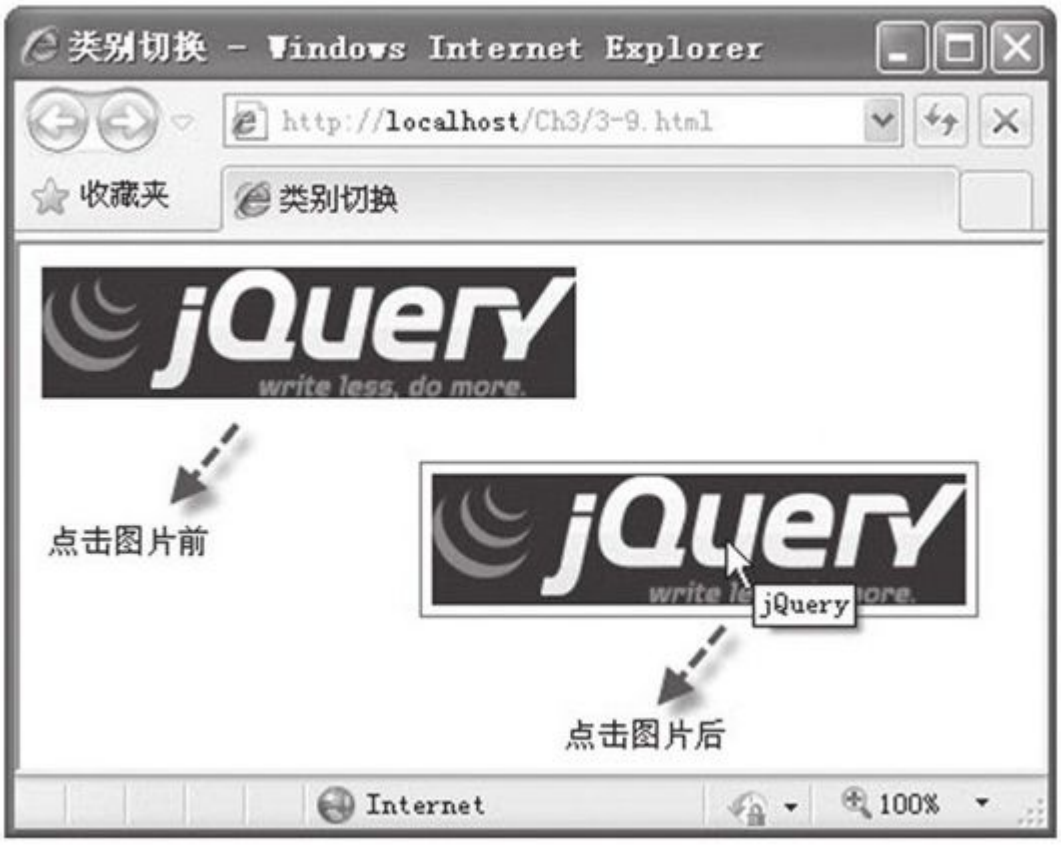


图　3-10　类别切换

（4）代码分析

toggleClass()方法可以实现类别间的切换，如在示例3-9中，单击图片后，图片样式发生变化，再次单击时，又返回单击前的样式，而css()或addClass()方法仅是增加新的元素样式，并不能实现类别的切换功能。

3.4.4　删除元素CSS类别

与增加类别的addClass()方法相对应，removeClass()方法则用于删除类别，其语法格式如下：

```
removeClass([class])
```

其中，参数class为类别名称，该名称是可选项。当选该名称时，删除名称是class的类别，有多个类别时用空格隔开。如果不选名称，则删除元素中的所有类别。

如果要删除p标记是cls0的类别，可以使用如下的代码：

```
$("p").removeClass("cls0");
```

如果要删除p标记是cls0和cls1的类别，可以使用如下的代码：

```
$("p").removeClass("cls0 cls1");
```

如果要删除p标记的全部类别，可以使用如下的代码：

```
$("p").removeClass();
```

3.5　页面元素操作

在本章开始时，我们讲过整个页面是一个DOM模型，页面中的各元素通过模型的节点相互关联形成树状，因此，如果要在页面中增加某个元素，只需要找到元素的上级节点，通过函数$(html)完成元素的创建后，增加到该节点中。

3.5.1　创建节点元素

函数$()用于动态创建页面元素，其语法格式如下：

```
$(html)
```

其中，参数html表示用于动态创建DOM元素的HTML标记字符串，即如果要在页面中动态创建一个div标记，并设置其内容和属性，可以加入如下代码：

```
var $div = $("<div title='jQuery理念'>Write Less Do More</div>");
           $("body").append($div);
```

执行上述代码后，将在页面文档body中创建一个div标记，其内容为"Write Less Do More"，属性title的值为“jQuery理念”。

示例3-10介绍了如何通过$()函数将页面元素动态增加到指定节点中。

示例3-10　使用$()函数动态创建节点元素

（1）功能描述

在页面中，分别设置左右两个部分，左边部分设置需要创建的元素对应的参数，如元素名、内容等，当用户设置完成后，单击“创建”按钮，则在页面的右边即时显示创建的元素。

（2）实现代码

新建一个HTML文件3-10.html,加入如代码清单3-10所示的代码。

代码清单 3-10　动态创建节点元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动态创建节点元素</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          ul{padding:0px;list-style:none}
          ul li{line-height:2.0em}
          .divL{float:left;width:200px;
                background-color:#eee;border:solid 1px #666;
                margin:5px;padding:0px 8px 0px 8px}
          .divR{float:left;width:200px;display:none;
                border:solid 1px #ccc;
                margin:5px;padding:0px 8px 8px 8px}
          .txt{border:#666 1px solid;padding:3px;width:120px}
          .btn {border:#666 1px solid;padding:2px;width:60px;
                filter: progid:DXImageTransform.Microsoft.
                Gradient(GradientType=0,
                StartColorStr=#ffffff,EndColorStr=#ECE9D8);}
```

```
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() {
                /* 获取参数 */
                var $str1 = $("#select1").val();// 获取元素名
                var $str2 = $("#text1").val();  // 获取内容
                var $str3 = $("#select2").val();// 获取属性名
                var $str4 = $("#text2").val();  // 获取属性值
                var $div = $("<" + $str1 + " " + $str3 + "='"
                           + $str4 + "'>" + $str2 + "</"
                           + $str1 + ">");       // 创建 DOM 对象
                $(".divR").show().append($div); // 插入节点中
            })
        })
    </script>
</head>
<body>
     <div class="divL"><p ></p>
         <ul>
            <li>元素名：
                <select id="select1">
                  <option value='p'>p</option>
                  <option value='div'>div</option>
                </select>
            </li>
            <li>内容为：
                <input type="text" id="text1" class="txt" />
            </li>
            <li>属性名：
                <select id="select2">
                  <option value='title'>title</option>
                </select>
            </li>
            <li>属性值：
                <input type="text" id="text2" class="txt"/>
            </li>
            <li style="text-align:center;padding-top:5px">
                <input id="Button1" type="button"
                       value=" 创建 " class="btn" />
            </li>
         </ul>
     </div>
     <div class="divR"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-11所示。

图　3-11　动态创建节点元素

（4）代码分析

函数$(html)只完成DOM元素创建，加入到页面还需要通过元素节点的插入或追加操作；同时，在创建DOM元素时，要注意字符标记是否完全闭合，否则达不到预期效果。

3.5.2 内部插入节点

在页面中动态创建元素需要执行节点的插入或追加操作。而在jQuery中，有多种方法可以实现该功能，append()方法仅是其中之一。按照插入元素的顺序来分，可以分为内部和外部两种方法。下面将分别对这些方法进行详细介绍。

内部插入节点的方法如表3-2所示。

表 3-2 内部插入节点的方法

语法格式	参数说明	功能描述
append(content)	content 表示追加到目标中的内容	向所选择的元素内部插入内容
append(function(index, html))	通过 function 函数返回追加到目标中的内容	向所选择的元素内部插入 function 函数所返回的内容
appendTo(content)	content 表示被追加的内容	把所选择的元素追加到另一个指定的元素集合中
prepend(content)	content 表示插入目标元素内部前面的内容	向每个所选择的元素内部前置内容
prepend(function(index, html))	通过 function 函数返回插入目标元素内部前面的内容	向所选择的元素内部前置 function 函数所返回的内容
prependTo(content)	content 表示用于选择元素的 jQuery 表达式	将所选择的元素前置到另一个指定的元素集合中

下面结合示例介绍其中几个重要的节点插入方法。

1.append(function(index,html))

该方法是jQuery 1.4中新增的，其功能是将一个function函数作为append方法的参数，该函数的功能必须返回一个字符串，作为

append方法插入的内容，其中index参数为对象在这个集合中的索引值，html参数为该对象原有的html值。

示例3-11演示了通过调用append()方法，以function返回的字符串作为参数，插入节点的过程。

示例3-11　使用append()方法插入节点

（1）功能说明

在页面中，通过一个funtion函数返回一段字符串，并将该字符串插入指定的div标记元素内容中。

（2）实现代码

新建一个HTML文件3-11.html,加入如代码清单3-11所示的代码。

代码清单 3-11　动态插入节点

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("div").append(retHtml);// 插入内容
            function retHtml() {
                var str = " <b>Write Less Do More</b> ";
                return str;
            }
        })
    </script>
</head>
<body>
     <div>jQuery</div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-12所示。

图　3-12　插入节点

2. appendTo(content)

该方法用于将一个元素插入另一个指定的元素内容中。例如果要将span标记插入div标记中，则执行下列代码：

```
$("span").appentTo($("div"));
```

即把appentTo方法前部分的内容插入其后部分的内容中。

示例3-12演示了调用appentTo()方法，将一个元素标记插入另一个标记中的过程。

示例3-12　使用appentTo()方法插入节点

（1）功能描述

在页面中创建一个img和两个span标记，通过appentTo()方法将img标记插入span标记中。

（2）实现代码

新建一个HTML文件3-12.html,加入如代码清单3-12所示的代码。

代码清单 3-12　将一个元素的内容动态插入另一个元素中

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          img{border:solid 1px #ccc;padding:3px;margin:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").appendTo($("span")); //插入内容
        })
    </script>
</head>
<body>
     <img title="2008年新书封面" src="Images/img03.jpg" />
     <span><img title="2010年新书封面"
                src="Images/img04.jpg" /></span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-13所示。

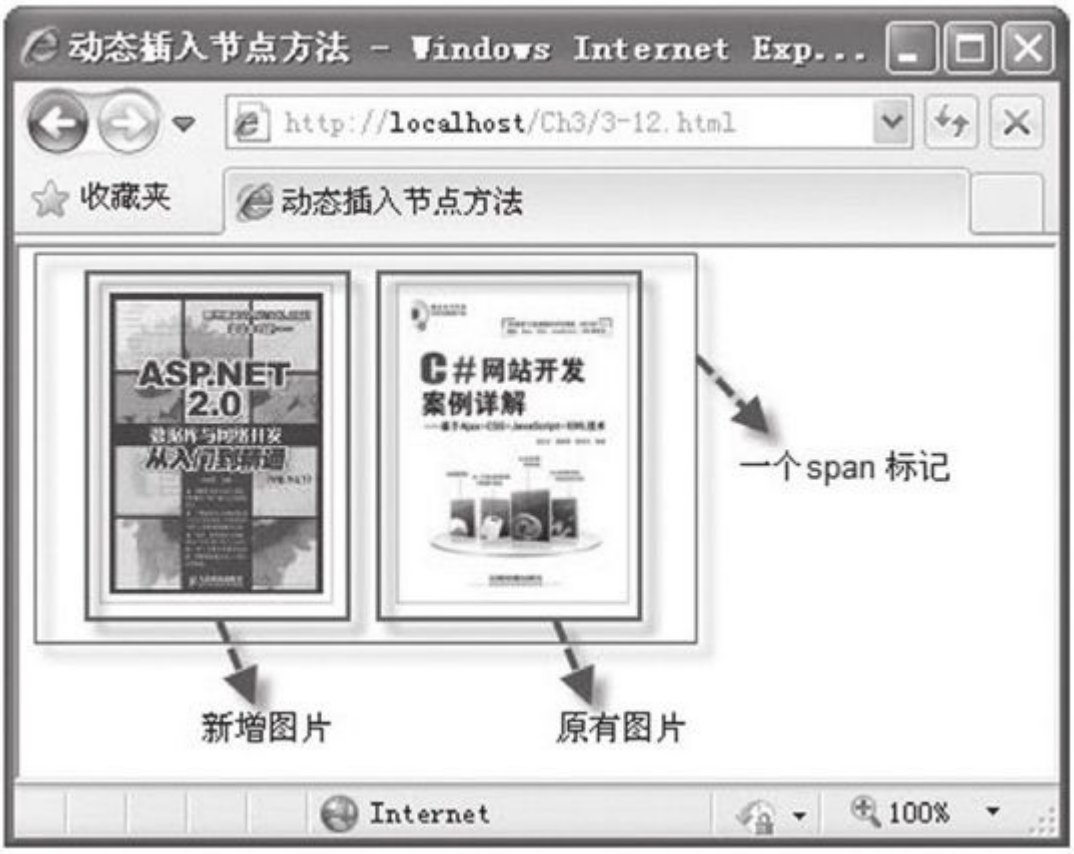


图　3-13　动态插入节点

3.5.3　外部插入节点

外部插入节点的方法如表3-3所示。

表 3-3　外部插入节点的方法

语法格式	参数说明	功能描述
after(content)	content 表示插入目标元素外部后面的内容	向所选择的元素外部后面插入内容
after(function)	通过 function 函数返回插入目标外部后面的内容	向所选择的元素外部后面插入 function 函数所返回的内容
before(content)	content 表示插入目标元素外部前面的内容	向所选择的元素外部前面插入内容
before(function)	通过 function 函数返回插入目标外部前面的内容	向所选择的元素外部前面插入 function 函数所返回的内容
insertAfter(content)	content 表示插入目标元素外部后面的内容	将所选择的元素插入另一个指定的元素外部后面
insertBefore(content)	content 表示插入目标元素外部前面的内容	将所选择的元素插入另一个指定的元素外部前面

after(function)方法也是jQuery 1.4中新增的方法，其function()参数将返回插入元素外部后面部分的内容。示例3-13演示了调用after()方法，以function的返回值为参数，在指定元素的外部插入一个节点的过程。

示例3-13　使用after()方法外部插入节点

（1）功能说明

在页面中创建一个span标记，然后通过function函数返回另外一个span标记，并将该标插入页面中的span标记后。

（2）实现代码

新建一个HTML文件3-13.html,加入如代码清单3-13所示的代码。

代码清单 3-13　动态插入节点

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("span").after(retHtml); //插入内容
            function retHtml() {
                var str = "<span><b>Write Less Do More</b><span>";
                return str;
            }
        })


    </script>
</head>
<body>
     <span>jQuery</span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-14所示。

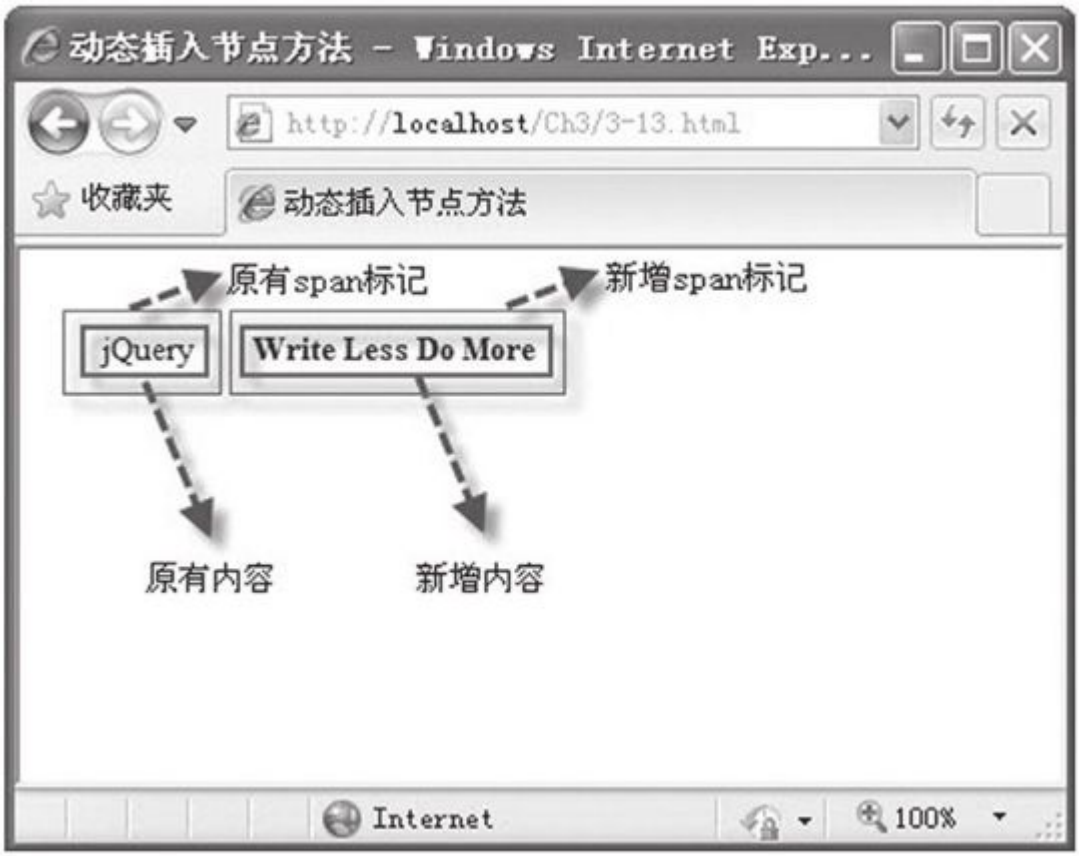

动态插入节点方法 - Windows Internet Exp...
http://localhost/Ch3/3-13.html
收藏夹
动态插入节点方法
原有span标记
新增span标记
jQuery
Write Less Do More
原有内容
新增内容
Internet
100%

图　3-14　外部插入节点

3.5.4　复制元素节点

在页面中，有时需要将某个元素节点复制到另外一个节点，如购物网站中购物车的设计。在传统的javaScript中，需要编写较为复杂的代码，而在jQuery中，可以通过方法clone()轻松实现，该方法的语法格式为：

```
clone()
```

其功能为复制匹配的DOM元素并且选中复制成功的元素。该方法仅是复制元素本身，被复制后的新元素不具有任何元素行为。如果需要在复制时将该元素的全部行为也进行复制，可以通过方法clone(true)实现，其格式为：

```
clone(true)
```

其中的参数设置为true，就可以复制元素的所有事件处理。示例3-14很好地展示了调用clone()方法，将一个元素标记复制成另一外标记的使用方法。

示例3-14　使用clone()方法复制元素节点

（1）功能描述

在页面中，创建一个img标记，用于显示一幅图片；单击该图片时，在其右侧通过clone()方法复制一幅图片。

（2）实现代码

新建一个HTML文件3-14.html,加入如代码清单3-14所示的代码。

代码清单 3-14　复制指定元素节点

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>复制元素节点</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          img{border:solid 1px #ccc;padding:3px;margin:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").click(function() {
                $(this).clone(true).appendTo("span");
            })
        })
    </script>
</head>
<body>
    <span><img title="封面" src="Images/img04.jpg" /></span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-15所示。

图　3-15　复制元素节点

（4）代码分析

本示例中使用的是clone(true)方法，单击被复制的新图片时，由于它具有原图片的事务处理，因此，将在该图片的右侧出现一幅通过其复制的新图片；如果使用clone()方法，那么只有单击原图片才可以复制新的图片元素，新复制的图片元素不具有任何功能。

3.5.5　替换元素节点

在jQuery中，如果要替换元素中的节点，可以使用replaceWith()和replaceAll()这两种方法，其语法格式分别如下：

```
replaceWith(content)
```

该方法的功能是将所有选择的元素替换成指定的HTML或DOM元素，其中参数content为被所选择元素替换的内容。

```
replaceAll(selector)
```

该方法的功能是将所有选择的元素替换成指定selector的元素，其中参数selector为需要被替换的元素。

示例3-15介绍这两种方法在使用上的区别。

示例3-15　使用replaceWith()和replaceAll()方法替换元素节点

（1）功能描述

在页面中创建两个span标记，ID号分别为span1和span2，然后通过jQuery中的两种替换元素的方法，分别替换元素span1和span2。

（2）实现代码

新建一个HTML文件3-15.html,加入如代码清单3-15所示的代码。

代码清单 3-15　替换页面元素内容

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>替换元素节点</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           span{font-weight:bold}
           p{background-color:#eee;padding:5px;width:200px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Span1").replaceWith("<span title='replaceWith'>陶国荣</span>");
            $("<span title='replaceAll'>
            tao_guo_rong@163.com</span>").replaceAll("#Span2");
        })
    </script>
</head>
<body>
    <p>姓名: <span id="Span1"></span></p>
    <p>邮箱: <span id="Span2"></span></p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-16所示。

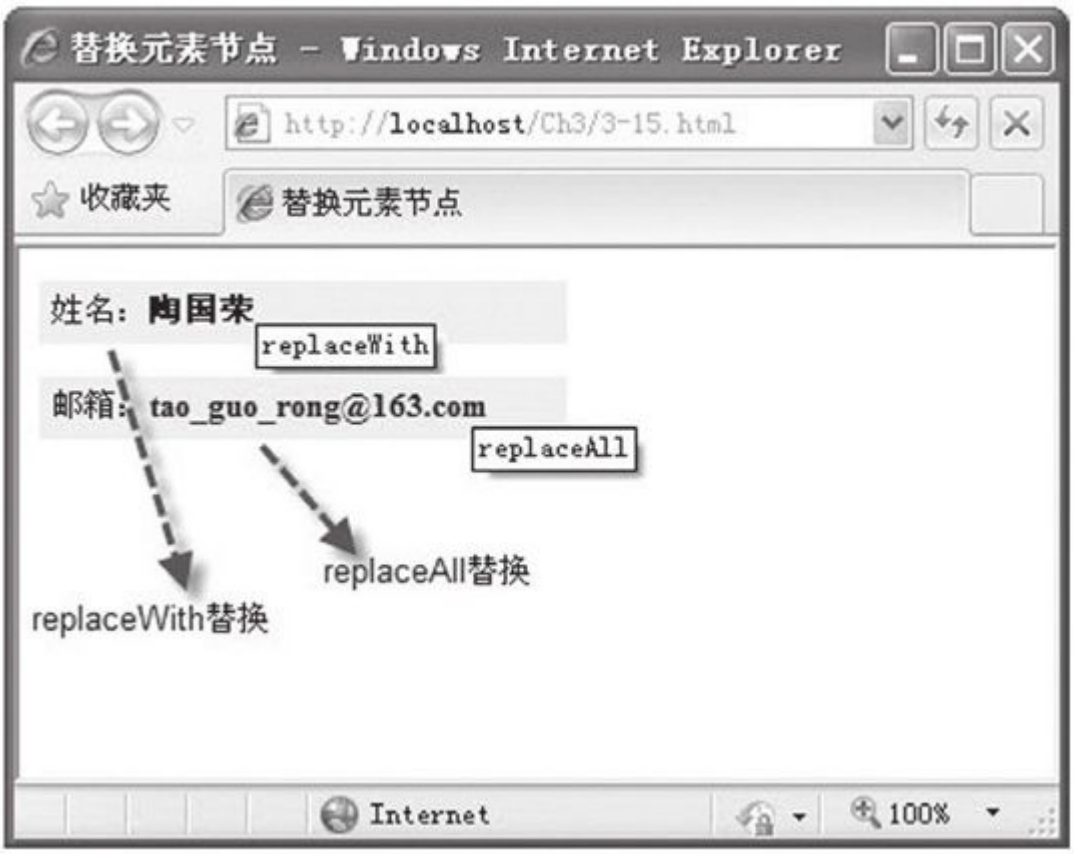


图　3-16　替换元素节点

（4）代码分析

replaceWith()与replaceAll()方法都可以实现元素节点的替换，二者最大的区别在于替换字符的顺序，前者是用括号中的字符替换所选择的元素，后者是用字符串替换括号中所选择的元素。一旦完成替换，被替换元素中的全部事件都将消失。

3.5.6 包裹元素节点

在jQuery中，不仅可以通过方法替换元素节点，还可以根据需求包裹某个指定的节点，对节点的包裹也是DOM对象操作中很重要的一项，其与包裹节点相关的全部方法如表3-4所示。

表 3-4 包裹元素节点

语法格式	参数说明	功能描述
wrap(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素用其他字符串代码包裹起来
wrap(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素用其他 DOM 元素包装起来
wrap(fn)	fn 参数为包裹结构的一个函数	把所有选择的元素用 function 函数返回的代码包裹起来
unwrap()	无参数	移除所选元素的父元素或包裹标记
wrapAll(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素用单个元素包裹起来
wrapAll(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素用单个 DOM 元素包裹起来
wrapInner(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素的子内容（包括文本节点）用字符串代码包裹起来
wrapInner(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素的子内容（包括文本节点）用 DOM 元素包裹起来
wrapInner(fn)	fn 参数为包裹结构的一个函数	把所有选择的元素的子内容（包括文本节点）用 function 函数返回的代码包裹起来

在上述表格中，wrap(html)与wrapInner(html)方法比较常用，前者包裹外部元素，后者包裹元素内部的文本字符。下面通过示例3-16介绍这两种常用方法在页面中包裹目标元素的使用效果。

示例3-16 使用wrap()和wrapInner()方法包裹元素节点

（1）功能描述

在页面中放置两个段落p标记，并在该标记内分别设置两个span标记，通过wrap()与wrapInner()两种方法，改变标记中的外部元素与内部文本的字体显示方式。

（2）实现代码

新建一个HTML文件3-16.html,加入如代码清单3-16所示的代码。

代码清单 3-16　包裹外部元素和内部文本的方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>包裹元素节点</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           p{background-color:#eee;padding:5px;width:200px}
    </style>
    <script type="text/javascript">
        $(function() {
           $("p").wrap("<b></b>");          // 所有段落标记字体加粗
           $("span").wrapInner("<i></i>");// 所有段落中的 span 标记斜体
        })
    </script>
</head>
<body>
    <p>最喜爱的体育运动：<span>羽毛球</span></p>
    <p>最爱看哪类型图书：<span>网络、技术</span></p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-17所示。

图　3-17　包裹元素节点的页面效果和源码

3.5.7　遍历元素

在DOM元素操作中，有时需要对同一标记的全部元素进行统一操作。在传统的JavaScript中，先获取元素的总长度，然后以for循环语句递减总长度，访问其中的某个元素，代码相对复杂；而在jQuery中，可以直接使用each()方法实现元素的遍历。其语法格式如下：

```
each(callback)
```

其中，参数callback是一个function函数，该函数还可以接受一个形参index，此形参为遍历元素的序号（从0开始）；如果需要访问元素中的属性，可以借助形参index，配合this关键字来实现元素属性的设置或获取。示例3-17演示了调用each()方法遍历全部元素获取每个元素属性的过程。

示例3-17　使用each()方法遍历元素获取属性

（1）功能描述

在页面中设置几幅图片，通过each()方法遍历全部的图片，并设置每幅图片的title属性。

（2）实现代码

新建一个HTML文件3-17.html,加入如代码清单3-17所示的代码。

代码清单 3-17　遍历元素获取属性

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>遍历元素</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{border:solid 1px #ccc;
           padding:3px;margin:5px;width:143px;height:101px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").each(function(index) {
                // 根据形参 index 设置元素的 title 属性
                this.title = "第" + index + "幅风景图片，alt 内容是" + this.alt;
            })
        })
    </script>
</head>
<body>
   <p>
   <img src="Images/img05.jpg" alt="第 0 幅风景画" />

    <img src="Images/img06.jpg" alt="第 1 幅风景画" />
    <img src="Images/img07.jpg" alt="第 2 幅风景画" /></p>
 </body>
 </html>
```

（3）页面效果

代码执行后的效果如图3-18所示。


遍历元素 - Windows Internet Explorer
http://localhost/Ch3/3-17.html
收藏夹
遍历元素
第1幅风景图片，alt内容是第1幅风景画
完成
Internet
100%

图　3-18　遍历元素

3.5.8　删除页面元素

在DOM操作页面时，删除多余或指定的页面元素是非常必要的，jQuery提供了两种删除元素的方法，即remove()和empty()。严格来说，empty()方法并非真正意义上的删除，使用该方法，仅仅可以清空全部的节点或节点所包括的所有后代元素，并非删除节点和元素。

remove()方法的语法格式如下：

```
remove([expr])
```

其中参数expr为可选项，如果接受参数，则该参数为筛选元素的jQuery表达式，通过该表述式获取指定的元素，并进行删除。

empty()方法的语法格式如下：

```
empty()
```

其功能为清空所选择的页面元素或所有的后代元素。

示例3-18说明了调用remove()方法删除某个页面元素的过程。

示例3-18　使用remove()方法删除页面元素

（1）功能说明

在页面中，通过ul标记展示列表式的数据信息，并设置一个按钮。单击该按钮时，将删除指定的标记元素。

（2）实现代码

新建一个HTML文件3-18.html，加入如代码清单3-18所示的代码。

代码清单 3-18　删除页面中指定的元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>删除元素</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           ul{width:200px}
           ul li{ list-style:none;padding:0px;height:23px}
           span{padding-left:20px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
                 filter: progid:DXImageTransform.Microsoft.
                 Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $("ul li:first").css("font-weight", "bold");//设置首行
            $("#Button1").click(function() {
                $("ul li").remove("li[title=t]");// 删除指定属性的元素
                $("ul li:eq(1)").remove();       // 删除节点中第 2 个元素
            })
        })
    </script>
</head>
<body>
     <ul>
         <li>学号</li>
         <li title="t">1001</li>
         <li>1002</li>
         <li>1003</li>
         <li style="text-align:center;padding-top:5px">
                <input id="Button1" type="button" value="删除" class="btn" />
         </li>
     </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图3-19所示。

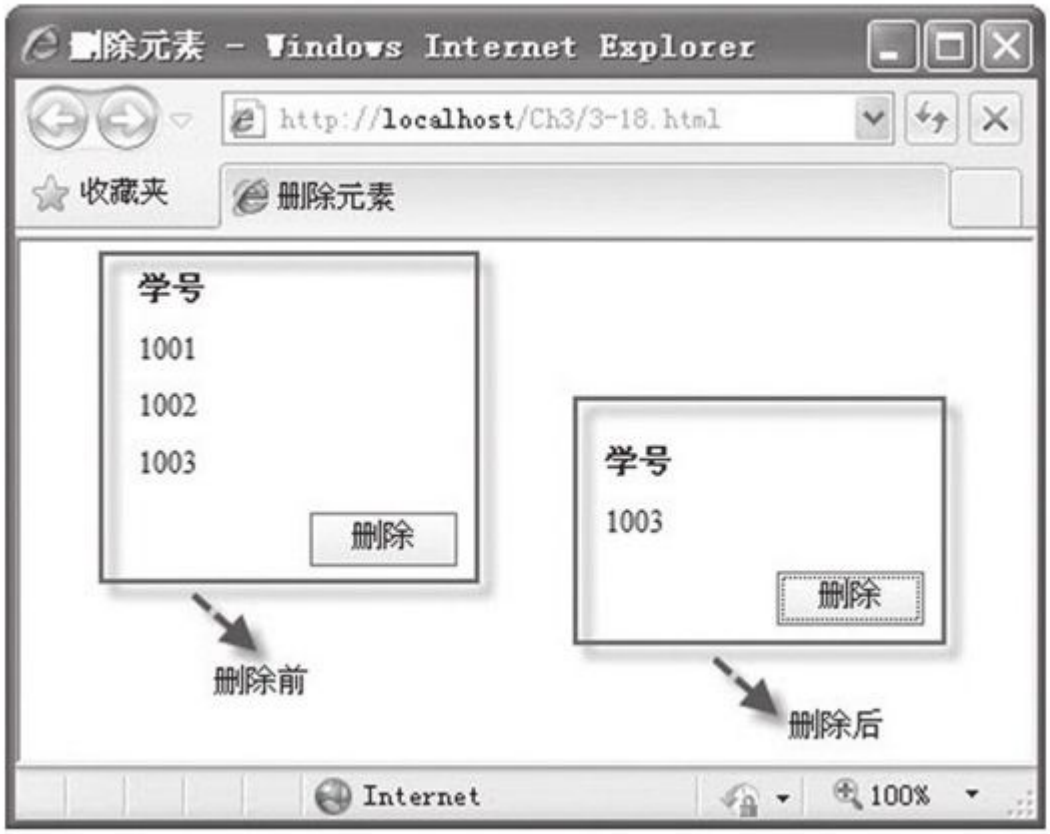


图　3-19　删除元素

（4）代码分析

在本示例中，删除按钮执行二次删除元素操作，首先是删除title等于t的元素；其次是删除li节点中的第2项元素。当学号为“1001”删除后，第2项元素就是“1002”，故仅留下学号“1003”。

3.6　综合案例分析——数据删除和图片预览在项目中的应用

3.6.1　需求分析

经分析，该案例的需求如下：

1）在页面中创建一个表格，用于展示多项数据信息，各行间采用隔行变色的方法展示每一行的数据。

2）如果选中表格中某行的复选项，并单击表格下面的“删除”按钮，那么将删除其选中的行；选中“全选”复选框后，再次单击“删除”按钮时，将删除表格的全部行数据。

3）如果将鼠标移到表格中某行的小图片上，将在该图片的右下角出现一幅与之相对应的大图片，用以实现图片预览的效果。

3.6.2　界面效果

页面初始化状态，使用隔行变色展示数据，其效果如图3-20所示。

图　3-20　隔行变色展示数据

当选择某行中的“选项”复选框后，单击“删除”按钮时，将删除其选中的行数据，其效果如图3-21所示。

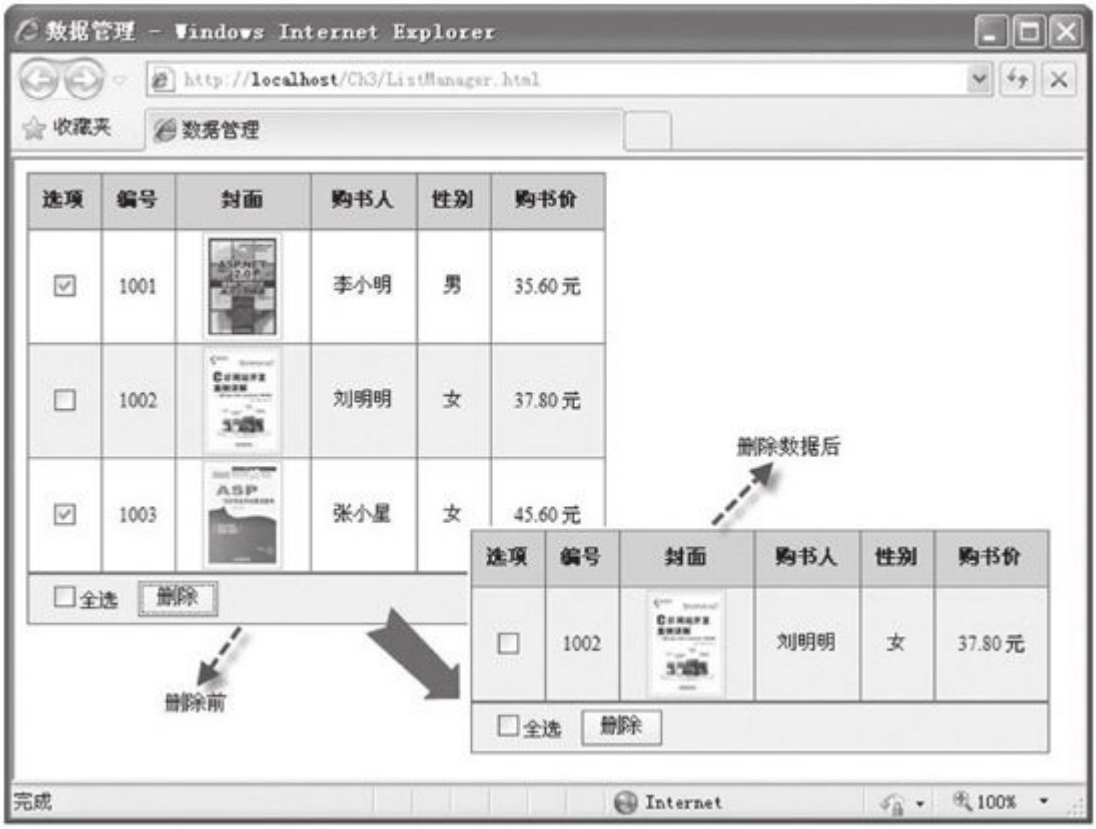


图　3-21　删除数据的前后对比

当鼠标移到“封面”小图片上时，将在该图片的右下角出现的一幅与之相对应的大图片，实现图片预览的效果，其效果如图3-22所示。

图　3-22　图片预览的效果

3.6.3　功能实现

在该项目中，新建一个HTML文件ListManager.html,加入如代码清单3-19所示的代码。

代码清单 3-19　数据删除和图片预览在项目中的应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/  TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>数据管理</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           table{width:360px;border-collapse:collapse}
           table tr th,td{border:solid 1px #666;text-align:center}
           table tr td img{border:solid 1px #ccc;
                 padding:3px;width:42px;height:60px;cursor:hand}
           table tr td span{float:left;padding-left:12px;}
           table tr th{background-color:#ccc;height:32px}
           .clsImg{position:absolute;border:solid 1px #ccc;
                 padding:3px;width:85px;height:120px;
                 background-color:#eee;display:none}
           .btn {border:#666 1px solid;padding:2px;width:50px;
                 filter: progid:DXImageTransform.Microsoft
                 .Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript" >
        $(function() {
            $("table tr:nth-child(odd)")
            .css("background-color", "#eee"); //隔行变色

            /** 全选复选框单击事件 **/
            $("#chkAll").click(function() {
                if (this.checked) {// 如果自己被选中
                    $("table tr td input[type=checkbox]")
                    .attr("checked", true);
                }
                else {// 如果自己没有被选中
                    $("table tr td input[type=checkbox]")
                    .attr("checked", false);
                }
            })

            /** 删除按钮单击事件 **/
            $("#btnDel").click(function() {
             var intL = $("table tr td
             input:checked
             :not('#chkAll')").length; // 获取除全选复选框外的所有选中项
                if (intL != 0) {// 如果有选中项
                    $("table tr td input[type=checkbox]:not('#chkAll')")
                    .each(function(index) {// 遍历除全选复选框外的行
                        if (this.checked) {// 如果选中
                            $("table tr[id=" + this.value + "]").remove();
                                        // 获取选中的值，并删除该值所在的行
                        }
                    })
                }
            })
            /** 小图片鼠标移动事件 **/
```

```
            var x = 5; var y = 15;// 初始化提示图片位置
            $("table tr td img").mousemove(function(e) {
                $("#imgTip")
                .attr("src", this.src)// 设置提示图片 scr 属性
                .css({ "top": (e.pageY + y) + "px",
                       "left": (e.pageX + x) + "px" })// 设置提示图片的位置
                .show(3000);// 显示图片
            })

            /** 小图片鼠标移出事件 **/
            $("table tr td img").mouseout(function() {
                $("#imgTip").hide();// 隐藏图片
            })

        })
    </script>
</head>
<body>
    <table>
        <tr>
            <th> 选项 </th>
            <th> 编号 </th>
            <th> 封面 </th>
            <th> 购书人 </th>
            <th> 性别 </th>
            <th> 购书价 </th>
        </tr>
        <tr id="0">
            <td><input id="Checkbox1" type="checkbox"
                       value="0"/></td>
            <td>1001</td>
            <td><img src="Images/img03.jpg" alt="" /></td>
            <td> 李小明 </td>
            <td> 男 </td>
            <td>35.60 元 </td>
        </tr>
        <tr id="1">
            <td><input id="Checkbox2" type="checkbox"
                       value="1"/></td>
            <td>1002</td>
            <td><img src="Images/img04.jpg" alt="" /></td>
            <td> 刘明明 </td>
            <td> 女 </td>
            <td>37.80 元 </td>
        </tr>
        <tr id="2">
            <td><input id="Checkbox3" type="checkbox"
                       value="2"/></td>
            <td>1003</td>
            <td><img src="Images/img08.jpg" alt="" /></td>
            <td> 张小星 </td>
            <td> 女 </td>
            <td>45.60 元 </td>
        </tr>
    </table>
```

```
    <table>
        <tr>
           <td style="text-align:left;height:28px">
               <span><input id="chkAll"
                           type="checkbox" /> 全选 </span>
               <span><input id="btnDel" type="button" value=" 删除 "
                           class="btn" /></span>
           </td>
        </tr>
     </table>
     <img id="imgTip" class="clsImg" src="Images/img03.gif"/>
</body>
</html>
```

3.6.4 代码分析

在“全选”复选框单击事件中，有如下的代码：

```
if (this.checked) {// 如果自己被选中
      $("table tr td input[type=checkbox]")
      .attr("checked", true);
}
else {// 如果自己没有被选中
      $("table tr td input[type=checkbox]")
      .attr("checked", false);
}
```

由于复选框是开关型按钮，所以首先通过if语句检测当前自身的状态，如果当前是被选中的状态，那么通过下面代码获取表格中的所有复选框，并将它们的属性设置为选中状态。

```
$("table tr td input[type=checkbox]").attr("checked", true);
```

反之，将它们的属性设置为未选中状态，代码如下：

```
$("table tr td input[type=checkbox]").attr("checked", false);
```

在“删除”按钮单击事件中，有如下代码：

```
var intL = $("table tr td
input:checked
:not('#chkAll')").length; // 获取除全选复选框外的所有选中项
if (intL != 0) {// 如果有选中项
    $("table tr td
        input[type=checkbox]
        :not('#chkAll')")
        .each(function(index) {// 遍历除全选复选框外的行
            if (this.checked) {// 如果选中
                $("table tr[id=" + this.value + "]").remove();
                    // 获取选中的值，并删除该值所在的行
            }
```

由于是删除操作，所以首先获取是否有可删除的项目，即通过下面代码获取表格中处于选中状态总行数。

```
var intL = $("table tr td
           input:checked :not('#chkAll')").length;
```

上行代码中:not('#chkAll')表示除掉表格底部的“全选”复选框。如果intL变量不为0，即有可删除的行，那么遍历除表格底部“全选”复选框外，表格中所有的复选框，代码如下：

```
$("table tr td input[type=checkbox]:not('#chkAll')")
.each(function(index) {
    // 删除代码
})
```

在遍历过程中，再次检测复选框是否被选中，如果选中，获取复选框的值，即this.value，然后通过该值与行的id值相匹配，如果符合，则删除该行，代码如下：

```
if (this.checked) {// 如果选中
    $("table tr[id=" + this.value+ "]").remove(); // 获取选中的值，并删除该值所在的行
}
```

注意：在实际的项目开发中，如果要删除某行数据，先获取该行数据选中后的ID号，即this.value值，然后将该值通过Ajax技术传给后台页面，执行数据库中的删除操作，即真正实现记录的删除功能，本实例仅是实现页面中行的删除。

在小图片鼠标移动事件中，首先设置提示图片scr属性，然后设置该图片与小图片的相对位置。由于该图片的display属性为none，因此还需要使用show()方法显示该图片。其代码如下：

```
$("#imgTip")
.attr("src", this.src)// 设置提示图片 scr 属性
.css({ "top": (e.pageY + y) + "px",
       "left": (e.pageX + x) + "px" })// 设置提示图片的位置
.show(3000);// 显示图片
```

在鼠标移出事件中隐藏该提示图片，代码如下：

```
$("#imgTip").hide();// 隐藏图片
```

3.7　本章小结

在页面中，控制元素是DOM对象主要的行为，本章介绍jQuery中访问或设置页面元素的常用方法，使读者全面了解如何通过jQuery中的方法完全操控页面，编写功能更丰富、更高效的页面代码，并为下一章节的学习奠定坚实的语法基础。

第4章　jQuery中的事件与应用

本章内容

事件机制

页面载入事件

绑定事件

切换事件

移除事件

其他事件

jQuery中的事件应用

综合案例分析——删除数据时的提示效果在项目中的应用

本章小结

当用户浏览页面时，浏览器会对页面代码进行解释或编译——这个过程实质上是通过事件来驱动的，即页面在加载时，执行一个Load

事件，在这个事件中实现浏览器编译页面代码的过程。事件无论在页面元素本身还是在元素与人机交互中，都占有十分重要的地位。

4.1　事件机制

众所周知，页面在加载时，会触发Load事件。当用户单击某个按钮时，触发该按钮的Click事件，通过种种事件实现各项功能或执行某项操作。事件在元素对象与功能代码中起着重要的桥梁作用。那么，事件被触发后是如何执行代码的呢？

4.1.1　事件中的冒泡现象

严格来说，事件在触发后分为两个阶段，一个是捕获（Capture），另一个则是冒泡（Bubbling）；但有些遗憾的是，大多数浏览器并不是都支持捕获阶段，jQuery也不支持。因此在事件触发后，往往执行冒泡过程。所谓的冒泡其实质就是事件执行中的顺序。下面通过示例4-1来说明事件执行过程中的冒泡现象。

示例4-1　事件中的冒泡现象

（1）功能描述

在页面中创建一个＜div＞标记，并在该标记中创建一个按钮。当用户单击页面或者＜div＞标记或者按钮时，都在页面中显示一句问候语“您好，欢迎来到jQuery世界!”。

（2）实现代码

新建一个HTML文件4-1.html,加入如代码清单4-1所示的代码。

代码清单 4-1　事件中的冒泡现象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>事件中的冒泡现象</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .clsShow{border:#ccc 1px solid;background-color:#eee;
               margin-top:15px;padding:5px;width:220px;
               line-height:1.8em;display:none}
           .btn {border:#666 1px solid;padding:2px;width:50px;
               filter: progid:DXImageTransform.Microsoft
               .Gradient(GradientType=0,StartColorStr=#ffffff,
               EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {


            var intI = 0;// 记录执行次数
            $("body,div,#btnShow").click(function() {// 单击事件
                intI++;// 次数累加
                $(".clsShow")
                .show()// 显示
                .html("您好，欢迎来到jQuery世界！")// 设置内容
                .append("<div>执行次数 <b>" + intI + "</b> </div>");// 追加文本
            })
        })
    </script>
</head>
<body>
    <div>
        <input id="btnShow" type="button" value="点击"
               class="btn" />
    </div>
    <div class="clsShow"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-1所示。

图　4-1　事件中的冒泡现象

（4）代码分析

为什么在示例4-1中，仅是单击按钮，但却执行了3次同样的程序呢？这是因为事件在执行过程中存在冒泡现象，即虽然单击的是按钮，但按钮的外围＜div＞元素的事件也被触发，同时＜div＞元素的外围＜body＞元素的事件也随之被触发，其整个事件波及的过程就像水泡一样向外冒，故称为冒泡过程。在示例4-1中，按钮被单击后，事件的执行过程如图4-2所示。

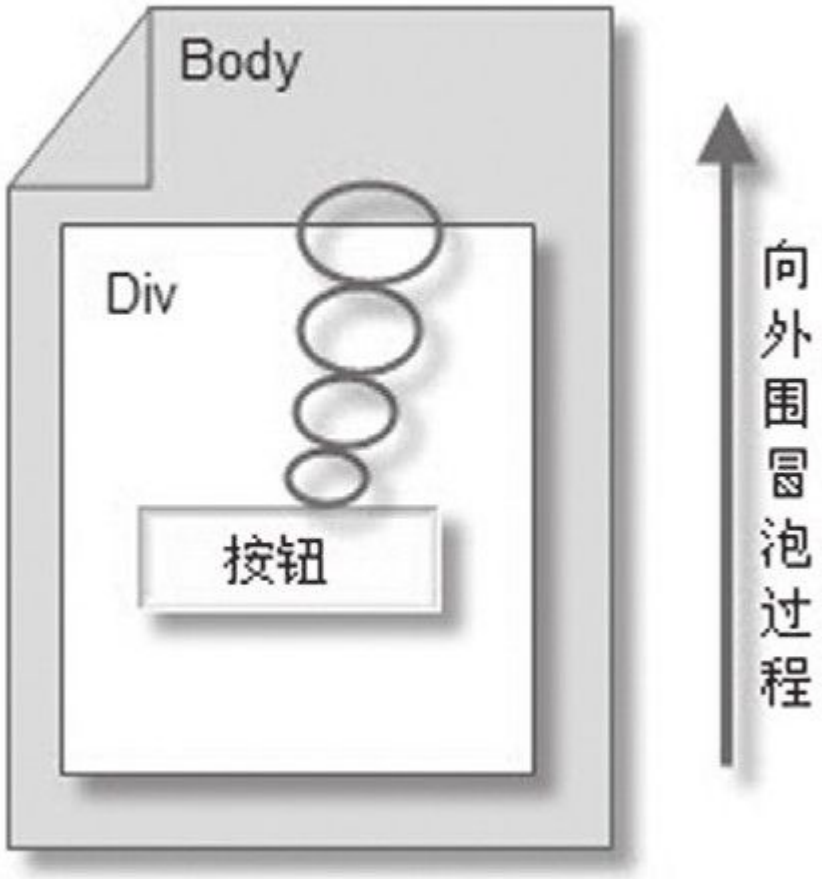


图　4-2　冒泡过程

4.1.2　如何阻止冒泡的发生

通过图4-2我们很清楚地看出，单击按钮时，事件在冒泡过程中先后执行了元素＜div＞和元素＜body＞中的事件，因此尽管执行的是按钮中的单击事件，但实际又触发了其他两个事件，所以共有3次事件的执行过程。

而在实际需要中，我们并不希望事件的冒泡现象发生，即单击按钮就执行单一的单击事件，并不触发其他外围的事件。在jQuery中，可以通过stopPropagation()方法可以阻止冒泡过程的发生。我们将示例4-1中的jQuery代码进行部分修改，如下所示：

```
$(function() {
    var intI = 0; //记录执行次数
    $("body,div,#btnShow").click(function(event) {//单击事件
          intI++; //次数累加
          $(".clsShow")
          .show()//显示
          .html("您好，欢迎来到jQuery世界！")//设置内容
          .append("<div>执行次数 <b>" + intI + "</b> </div>"); //追加文本
          event.stopPropagation();//阻止冒泡过程
    })
})
```

修改完成后，执行示例4-1，单击按钮后，其页面显示的执行次数为1，即仅触发了按钮的单击事件，而没有涉及其他外围元素事件。

说明　在代码中，除了使用stopPropagation()方法阻止事件的冒泡过程外，还可以通过语句return false实现停止事件的冒泡过程。

4.2　页面载入事件

在第1章节中，我们简单地介绍了jQuery中的页面载入事件ready()方法。该方法类似于传统JavaScript中的OnLoad()方法，只不过在事件执行时间上有区别：OnLoad()方法的执行必须是页面中的全部元素完全加载到浏览器后才触发，在这种情况下，如果页面中的图片过多或图片过大，那么有可能要等OnLoad()方法执行完毕，用户才能进行其他的操作；如果使用jQuery中的ready()方法加载页面，则只要页面的DOM模型加载完毕，就会触发ready()方法。因此，两者在事件的执行效果上ready()方法明显优于JavaScript中的OnLoad()方法。

4.2.1　ready()方法的工作原理

我们解剖一下jQuery中ready()方法的工作原理。

在jQuery脚本加载到页面时，会设置一个isReady的标记，用于监听页面加载的进度。遇到执行ready()方法时，通过查看isReady值是否被设置，如果未被设置，那么就说明页面并未加载完成，在此情况下，将未完成的部分用一个数组缓存起来，当全部加载完成后，再将未完成的部分通过缓存一一执行。

4.2.2 ready()方法的几种写法

以下几种代码写法其执行的效果是一样的。

写法一：

```
$(document).ready(function(){
    //代码部分
})
```

写法二：

```
$(function(){
    //代码部分
})
```

写法三：

```
jQuery(document).ready(function(){
    //代码部分
})
```

写法四：

```
jQuery(function(){
    // 代码部分
})
```

其中写法二简洁明了，使用较为广泛。

4.3 绑定事件

我们在示例4-1中，使用如下的代码绑定按钮的单击事件：

```
$(function(){
    $("#btnShow").click(function(){
        // 执行代码
    })
})
```

除了上述绑定事件的写法外，在jQuery中，还可以使用bind()方法进行事件的绑定。下面对该方法进行详细的介绍。

4.3.1 使用bind()方法绑定事件

bind()功能是为每个选择元素的事件绑定处理函数，其语法格式如下：

```
bind(type, [data], fn)
```

其中参数type为一个或多个类型的字符串，如"click"或"change"，也可以自定义类型；可以被参数type调用的类型包括blur、focus、load、resize、scroll、unload、click、dblclick、mousedown、mouseup、mousemove、mouseover、

mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup、error。

参数data是作为event.data属性值传递给事件对象的额外数据对象。

参数fn是绑定到每个选择元素的事件中的处理函数。

下面通过两个例子来讲解绑定方法。

示例4-2　使用bind()方法绑定事件

（1）功能描述

在页面中，设置一个按钮，通过bind()方法给按钮绑定一个click事件，在该事件中，将自身的是否可用属性设置成false，即单击按钮后就不可使用。

（2）实现代码

新建一个HTML文件4-2.html,加入如清单4-2所示的代码。

代码清单 4-2　使用 bind() 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>bind 方法绑定事件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           .btn {border:#666 1px solid;padding:2px;width:50px;
                 filter: progid:DXImageTransform.Microsoft
                 .Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#btnBind").bind("click", function() {
                $(this).attr("disabled", "disabled");//按钮不可用
            })
        })
    </script>
</head>
<body>
     <input id="btnBind" type="button" value="Button" class="btn" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-3所示。

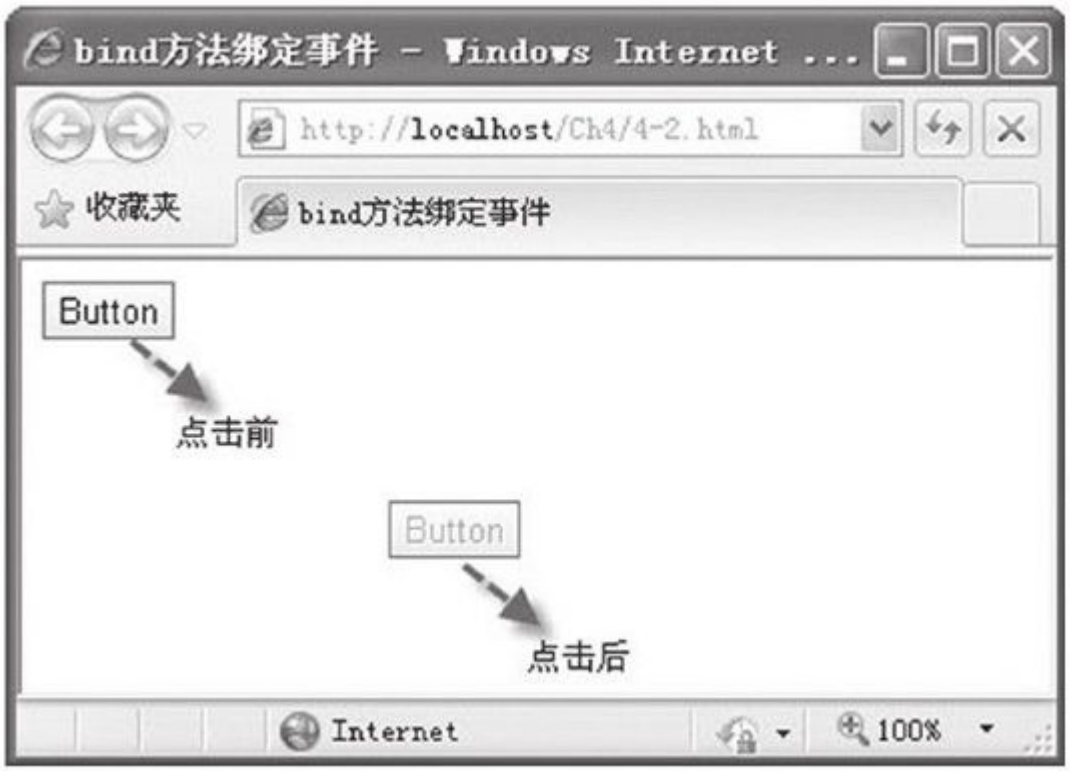


图　4-3　bind方法绑定事件

（4）代码分析

如果要在一个元素中绑定多个事件，可以将事件用空格隔开，如示例4-2中，除单击事件外，如果需要加一个mouseout事件，可以将代码修改为如下所示：

```
... 省略头部代码
$(function() {
    $("#btnBind").bind("click mouseout", function() {
        $(this).attr("disabled", "disabled"); // 按钮不可用
    })
})
... 省略主体代码
```

4.3.2　通过映射方式绑定事件

在jQuery绑定事件时，还可以通过传入一个映射，对所选对象绑定多个事件处理函数，见下面的示例。

示例4-3　使用映射方式绑定不同的事件

（1）功能描述

在页面中，设置一个文本框，通过映射的方式，给文本框绑定两个事件，一个是focus事件，另一个是change事件。这两个事件执行时，均为显示事件的名称。

（2）实现代码

新建一个HTML文件4-3.html,加入如清单4-3所示的代码。

代码清单 4-3　映射方式绑定不同的事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>映射方式绑定不同的事件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .clsTip{border:#ccc 1px solid;
           background-color:#eee;margin-top:15px;
           padding:5px;width:185px;display:none}
           .txt{border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".txt").bind({ focus: function() {
                $("#divTip")
                .show()// 显示
                .html(" 执行的是 focus 事件 ");// 设置文本
            },
                change: function() {
                    $("#divTip")


                    .show()// 显示
                    .html(" 执行的是 change 事件 ");// 设置文本
                }
            })
        })
    </script>
</head>
<body>
     <div>姓名: <input type="text" class="txt" /></div>
     <div id="divTip" class="clsTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-4所示。

图　4-4　映射方式绑定不同的事件

（4）代码分析

在bind()方法中，第二个参数data为可选项，表示作为event.data属性值传递到事件对象的额外数据对象，实际上，该参数很少使用，如果使用，那么可以通过该参数将一些附加的信息传递给事件处理函数fn中，在示例4-3中，将代码中的bind方法修改成带data参数，其修改后的代码如下所示：

```
... 省略头部代码
$(function() {
    var message = "执行的是focus事件";
    $(".txt").bind("focus", { msg: message }, function(event) {
        $("#divTip")
        .show()// 显示
        .html(event.data.msg); // 设置文本
      });
    message = "执行的是change事件";
    $('.txt').bind('change', { msg: message }, function(event) {
        $("#divTip")
        .show()// 显示
        .html(event.data.msg); // 设置文本
      });
})
... 省略主体代码
```

代码执行后，页面效果与图4-4一模一样。

4.4　切换事件

在jQuery中，有两个方法用于事件的切换，一个是方法hover()，另一个是方法toggle()。所谓切换事件，即有两个以上的事件绑定于一个元素，在元素的行为动作间进行切换。如一个超级链接标记＜a＞，若想实现当鼠标悬停时触发一个事件，鼠标移出时又触发另一个事件，就可以调用jQuery中的hover()方法轻松实现。

4.4.1　hover()方法

调用jQuery中的hover()方法可以使元素在鼠标悬停与鼠标移出的事件中进行切换，该方法在实现运用中，也可以通过jQuery中的事件mouseenter与mouseleave进行替换。下列代码是等价的。

```
$("a").hover(function() {
    //执行代码一
  }, function() {
    //执行代码二
})
```

等价于：

```
$("a").mouseenter(function() {
    // 执行代码一
 })
$("a").mouseleave(function() {
    // 执行代码二
 })
```

hover()功能是当鼠标移动到所选的元素上面时，执行指定的第一个函数；当鼠标移出这个元素时，执行指定的第二个函数，其语法格式如下：

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

其中，参数over为鼠标移到元素时触发的函数，参数out为鼠标移出元素时触发的函数。下面举例说明。

示例4-4　使用hover()方法切换事件

（1）功能描述

在页面中，创建一个＜div＞标记，在该标记中，设置另外两个＜div＞元素，一个用于显示标题，另一个则用于显示内容。调用jQuery中的hover()方法。将鼠标移到标题时，自动显示内容；鼠标移出标题时，关闭显示的内容。

（2）实现代码

新建一个HTML文件4-4.html,加入如清单4-4所示的代码。

代码清单 4-4　hover 方法切换事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">


<head>
    <title>切换事件 hover</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .clsFrame{border:solid 1px #666;width:220px}
           .clsFrame .clsTitle{background-color:#eee;
           padding:5px;font-weight:bold}
           .clsFrame .clsContent{padding:5px;display:none}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".clsTitle").hover(function() {
                $(".clsContent").show();
            }, function() {
                $(".clsContent").hide();
            })
        })
    </script>
</head>
<body>
     <div class="clsFrame">
          <div class="clsTitle">jQuery 简介 </div>
          <div class="clsContent">  jQuery 是由美国人 John Resig 于 2006
          年创建的一个开源项目，它的主旨是：以更少的代码，实现更多的功能（Write less，do
          more）。</div>
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-5所示。

图　4-5　hover方法切换事件

4.4.2 toggle()方法

在toggle()方法中，可以依次调用N个指定的函数，直到最后一个函数，然后重复对这些函数轮番调用。

toggle()方法的功能是每次单击后依次调用函数，请注意“依次”这两个字，说明该方法在调用函数时并非随机或指定调用，而是通过函数设置的前后顺序进行调用，其调用的语法格式如下：

```
toggle(fn, fn2, [fn3, fn4, ...])
```

其中参数fn，fn2，...，fnN为单击时被依次调用的函数。

示例4-5 使用toggle()方法切换事件

（1）功能描述

在页面中，设置一个＜img＞标记，当用户第一次单击该图片时，变换另外一幅图片；第二次单击图片时，变换第三幅图片；第三次单击时，返回第一次单击前的图片；依次轮番显示。

（2）实现代码

新建一个HTML文件4-5.html,加入如清单4-5所示的代码。

代码清单 4-5　toggle() 方法切换事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>切换事件 toggle</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{border:solid 1px #ccc;padding:3px}
   </style>
    <script type="text/javascript">
        $(function() {
            $("img").toggle(function() {
                $("img").attr("src", "Images/img05.jpg");
                $("img").attr("title", this.src);
            }, function() {
                $("img").attr("src", "Images/img06.jpg");
                $("img").attr("title", this.src);
            }, function() {
                $("img").attr("src", "Images/img07.jpg");
                $("img").attr("title", this.src);
            })
        })
    </script>
</head>
<body>
     <img />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-6所示。

图　4-6　toggle方法切换事件

4.5　移除事件

在DOM对象的实践操作中，既然存在用于绑定事件的bind方法，也相应存在用于移除绑定事件的方法。在jQuery中，可以通过unbind()方法移除绑定的所有事件或指定某一个事件。

4.5.1　unbind()方法移除元素绑定事件

unbind()的功能是移除元素绑定的事件，其调用的语法格式如下：

```
unbind([type], [fn])
```

其中，参数type为移除的事件类型，fn为需要移除的事件处理函数。如果该方法没有参数，移除所有绑定的事件；如果带有参数type，移除该参数所指定的事件类型；如果带有参数fn，则只移除绑定时指定的函数fn。下面举例说明。

示例4-6　使用unbind()方法移除元素绑定事件

（1）功能描述

在页面中设置三个按钮，前两个按钮分别执行各自的事件，第三个按钮通过unbind方法移除所绑定的全部事件。即单击第三个按钮后，前两个按钮的事件将不会执行。

（2）实现代码

新建一个HTML文件4-6.html,加入如清单4-6所示的代码。

代码清单 4-6　unbind() 方法移除事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>移除事件 unbind</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
           div{line-height:1.8em}
    </style>
    <script type="text/javascript">
        $(function() {
            function oClick() { //自定义事件
                $("#divTip").append("<div>按钮二的单击事件</div>");
            }
             $("input:eq(0)").bind("click", function() { //绑定单击事件
                $("#divTip").append("<div>按钮一的单击事件</div>");
            });
             $("input:eq(1)").bind("click", oClick); //绑定自定义事件
             $("input:eq(2)").bind("click", function() { //移除全部单击事件$
            ("input").unbind();
            });
        })
    </script>
</head>
<body>
     <div>
         <input id="Button1" type="button" value="按钮一" class="btn" />
         <input id="Button2" type="button" value="按钮二" class="btn"/>
         <input id="Button3" type="button" value="删除事件" class="btn"/>
     </div>
     <div id="divTip" style="padding-top:10px"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-7所示。

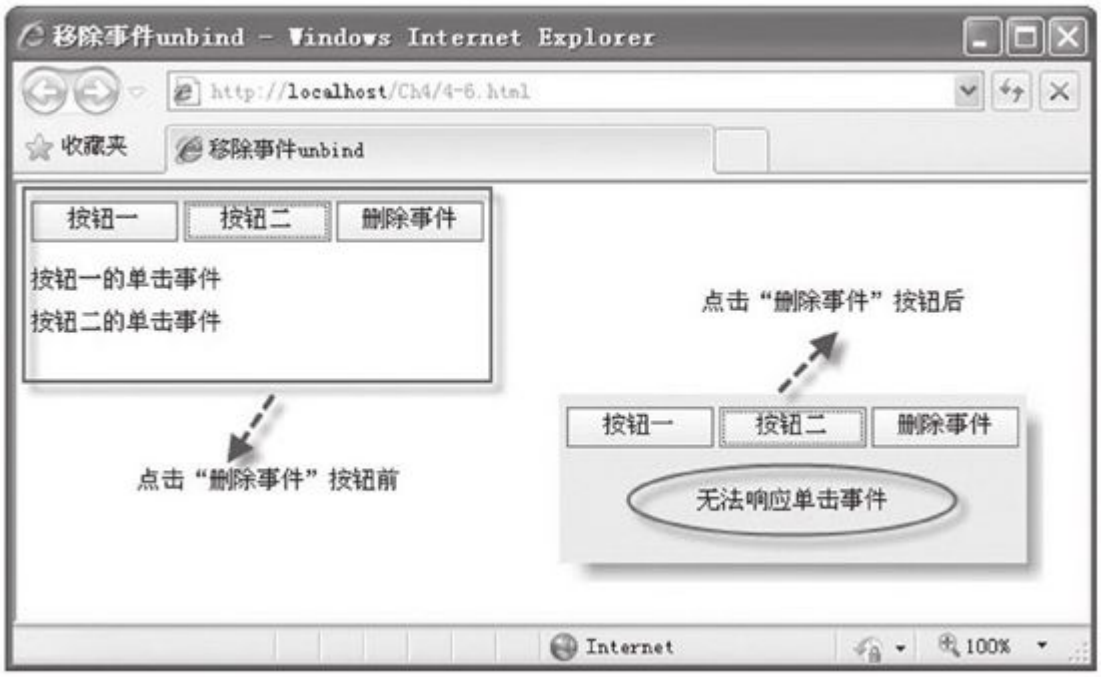


图　4-7　unbind方法移除事件

4.5.2 unbind()方法移除自定义事件

通过语法介绍我们知道，unbind()方法不仅可以移除某类型的全部事件，还可移除某个指定的自定义事件，如在代码清单4-6中，如果将下列代码：

```
$("input:eq(2)").bind("click", function() { // 移除全部单击事件
    $("input").unbind();
 });
```

修改成下列代码：

```
$("input:eq(2)").bind("click", function() { // 移除全部单击事件
    $("input").unbind("click",oClick);
 });
```

那么，当单击“删除事件”按钮后，单击“按钮一”将有对应事件的响应，而单击“按钮二”将无对应事件的响应，因为在“删除事件”的按钮中，移除的仅仅是“按钮二”的自定义事件，因此，只有该按钮没有事件的响应。

4.6　其他事件

除上述介绍的几种事件方法外，在jQuery中还有很多的事件处理方法，下面介绍其中较为实用的两种处理事件的方法one()和trigger()。

4.6.1　one()方法

one()方法功能是为所选的元素绑定一个仅触发一次的处理函数，其调用的语法格式为：

```
one(type, [data], fn)
```

其中，参数type为事件类型，即需要触发什么类型的事件；参数data为可选参数，表示作为event.data属性值传递给事件对象的额外数据对象；fn为绑定事件时所要触发的函数。下面举例说明。

示例4-7　使用one()方法绑定触发一次的事件

（1）功能描述

在页面中设置一个按钮，初始值为“点击查看联系方式”。单击该按钮时，通过jQuery中的one()方法将联系方式显示在按钮上，再次

单击则不响应任何事件。

（2）实现代码

新建一个HTML文件4-7.html,加入如代码清单4-7所示的代码。

代码清单 4-7　one()方法绑定触发一次的事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>其他事件one</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>


    <style type="text/css">
           .btn {border:#666 1px solid;padding:2px;width:160px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            function btn_Click() { // 自定义事件
                this.value = "010-12345678"
            }
            $("input").bind("click", btn_Click); // 绑定自定义事件
        })
    </script>
</head>
<body>
    <input id="Button1" type="button" value="点击查看联系方式" class="btn" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-8所示。

图　4-8　one方法绑定事件

4.6.2　trigger()方法

在前端页面开发中，有时希望页面在DOM加载完毕后，自动执行一些人性化的操作，如：文本框中的内容处于全部被选中状态，某个按钮处于焦点中。利用传统的JavaScript语言需要编写复杂的代码才可实现上述功能；而在jQuery中，仅需要调用一个trigger()方法就可以轻松实现。

trigger()方法的功能是在所选择的元素上触发指定类型的事件。其调用的语法格式为：

```
trigger(type, [data])
```

其中，参数type为触发事件的类型，参数data为可选项，表示在触发事件时传递给函数的附加参数。下面举例说明。

示例4-8　使用trigger()方法触发指定类型事件

（1）功能描述

在页面中创建一个文本框，并给文本框设置一个默认值。该页面加载时，自动获取文本框中的值显示在页面中，同时，文本框处于选

中状态。

（2）实现代码

新建一个HTML文件4-8.html,加入如清单4-8所示的代码。

代码清单 4-8　trigger() 方法触发指定类型事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>其他事件trigger</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
            body{font-size:13px}
           .txt{border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            var oTxt = $("input"); //获取文本框
            oTxt.trigger("select"); //自动选中文本框
            oTxt.bind("btn_Click", function() {//编写文本框自定义事件
                var txt = $(this).val();        //获取自身内容
                $("#divTip").html(txt);         //显示在页面中
            })
            oTxt.trigger("btn_Click");          //自动触发自定义事件
        })
    </script>
</head>
<body>
    姓名: <input id="Text1" type="text" class="txt"  value="梅国荣" />
         <div id="divTip" style="padding-top:5px"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-9所示。

图　4-9　trigger方法触发指定类型事件

（4）代码分析

trigger()方法可以实现触发性事件，即不必用户做任何动作，自动执行该方法中的事件。在这种情形下，其最终效果可能会有异样发生。如果不希望页面自动执行，可使用triggerHandler()方法，该方法与trigger()方法基本相同，只是该方法不会自动执行其包含的事件。

4.7　jQuery中的事件应用

我们在前面的章节中介绍jQuery选择器时，有专门用于表单的选择器，可见表单在jQuery中占有十分重要的地位。表单在HTML中处理数据的优势也是不言而喻的，因此，在jQuery事件中，表单的应用也显得尤为重要。

4.7.1　文本框中的事件应用

文本框是表单中使用最为普遍的元素之一，其前端用户页面的体验度十分重要。下面通过一个简单示例，介绍使用jQuery中的事件改变文本框的样式，以提高用户体验。

示例4-9　使用jQuery事件改变文本框的样式

（1）功能描述

在页面中，创建一个用于输入邮箱地址的文本框，结合使用jQuery与CSS，实现当用文本框获取时样式发生变化，同时提示用户输入邮箱的方法。

当用户在文本框中输入邮箱后，丢失焦点时，将检测其内容是否为空，如果不为空或邮箱格式不符，样式将再次发生变化，同时提示

出错信息；如果输入正确，样式将返回初始状态，并显示一个打勾的图片。

（2）实现代码

新建一个HTML文件4-9.html,加入如清单4-9所示的代码。

代码清单 4-9　文本框中的事件应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> 文本框中的事件应用 </title>
```

```
<script type="text/javascript"
        src="Jscript/jquery-1.8.2.min.js">
</script>
<style type="text/css">
        body{font-size:13px}
        /* 元素初始状态样式 */
        .divInit{width:390px;height:55px;line-height:55px;
        padding-left:20px}
        .txtInit{border:#666 1px solid;padding:3px;
        background-image:url('Images/bg_email_input.gif')}
        .spnInit{width:179px;height:40px;line-height:40px;
        float:right;margin-top:8px;padding-left:10px;
        background-repeat:no-repeat}

        /* 元素丢失焦点样式 */
        .divBlur{background-color:#FEEEC2}
        .txtBlur{border:#666 1px solid;padding:3px;
        background-image:url('Images/bg_email_input2.gif')}
        .spnBlur{background-image:
        url('Images/bg_email_wrong.gif')}

        .divFocu{background-color:#EDFFD5}/* div 获取焦点样式 */
        .spnSucc{background-image:
        url('Images/pic_Email_ok.gif');
        margin-top:20px}/* 验证成功时 span 样式 */
</style>
<script type="text/javascript">
    $(function() {
        $("#txtEmail").trigger("focus");// 默认时文本框获取焦点
        $("#txtEmail").focus(function() {// 文本框获取焦点事件
            $(this).removeClass("txtBlur")
            .addClass("txtInit");
            $("#email").removeClass("divBlur")
            .addClass("divFocu");
            $("#spnTip").removeClass("spnBlur")
            .removeClass("spnSucc")
            .html(" 请输入您常用邮箱地址! ");
        })

        $("#txtEmail").blur(function() {// 文本框丢失焦点事件
            var vtxt = $("#txtEmail").val();
            if (vtxt.length == 0) {
                $(this).removeClass("txtInit")
                .addClass("txtBlur");
                $("#email").removeClass("divFocu")
                .addClass("divBlur");
                $("#spnTip").addClass("spnBlur")
                .html(" 邮箱地址不能为空! ");
            }
            else {
```

```
                    if (!chkEmail(vtxt)) {//检测邮箱格式是否正确
                        $(this).removeClass("txtInit")
                        .addClass("txtBlur");
                        $("#email").removeClass("divFocu")
                        .addClass("divBlur");
                        $("#spnTip").addClass("spnBlur")
                        .html("邮箱格式不正确! ");
                    }
                    else {//如果正确
                        $(this).removeClass("txtBlur")
                        .addClass("txtInit");
                        $("#email").removeClass("divFocu");
                        $("#spnTip").removeClass("spnBlur")
                        .addClass("spnSucc").html("");
                    }
                }
            })
            /*
             *验证邮箱格式是否正确
             *参数 strEmail，需要验证的邮箱
            */
            function chkEmail(strEmail) {
                if (!/^\w+([-+.]\w+)*@\w +([-.]\w+)*\.\w+([-.]\w+)*$/
                    .test(strEmail)) {
                    return false;
                }
                else {
                    return true;
                }
            }
        })
    </script>
</head>
<body>
    <form id="form1" action="#">
        <div id="email" class="divInit">邮箱:
            <span id="spnTip" class="spnInit"></span>
            <input id="txtEmail" type="text" class="txtInit" />
        </div>
    </form>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-10所示。

图　4-10　文本框用于输入邮箱时的不同状态

（4）代码分析

在代码中，为了提升用户页面体验度，我们针对文本框编写了两个事件，一个是focus事件，另一个是blur事件。在focus事件中，我们加入了如下的代码。

```
$("#txtEmail").focus(function() {// 文本框获取焦点事件
    $(this).removeClass("txtBlur")
    .addClass("txtInit");
    $("#email").removeClass("divBlur")
    .addClass("divFocu");
    $("#spnTip").removeClass("spnBlur")
    .removeClass("spnSucc")
    .html(" 请输入您常用邮箱地址！ ");
  })
```

在文本框中的focus获取焦点事件代码中，涉及三个元素的样式发生变化。

第一个元素是文本框自身，用this表示。在获取焦点时，有可能丢失焦点，因此，首先删除丢弃焦点的样式，然后再加入获取焦点的样式，即使用下列代码：

```
$(this).removeClass("txtBlur")
.addClass("txtInit");
```

第二个元素是外层Div区域，同理也是先删除丢弃焦点的样式，然后再加入获取焦点的样式，其代码如下：

```
$("#email").removeClass("divBlur")
.addClass("divFocu");
```

第三个元素是用于文本框的提示信息span元素，同理也是删除全部加载过的样式，使其恢复到初始样式，并显示相关的提示信息，其代码如下：

```
$("#spnTip").removeClass("spnBlur")
.removeClass("spnSucc")
.html("请输入您常用邮箱地址！");
```

在文本框的blur丢失焦点事件中，其代码与foucs事件有相似之处，都是先清理原先加载过的页面样式，然后增加本身事件中的样式。不同之处在于，在文本框的blur事件中，还要进行文本框内容是

否为空和邮箱格式是否符合的检测操作。如果不为空，再进行检测，其代码如下：

```
if (vtxt.length == 0) {
    // 文本内容为空时执行的代码
    ...
   }
else { // 如果不为空时
    if (!chkEmail(vtxt)) {// 检测邮箱格式是否正确
    // 邮箱格式不正确时执行的代码
    ...
    }
    else {// 如果正确
          // 邮箱格式正确时执行的代码
          ...
     }
}
```

在jQuery中，addClass()方法的功能是增加某种CSS样式，为了更好地体现设置的样式，在增加新的CSS样式前，应先通过removeClass()方法，删除已加载过的页面样式，以达到预期的页面效果。

4.7.2　下拉列表框中的事件应用

下拉列表框是最为常用的表单对象，该对象可以通过较小的页面空间展示大量的数据；同时，多个列表框通过联动效果，展示数据的应用也相当广泛。下面通过一个示例，介绍如何在jQuery中，实现三个下拉列表框联动展示数据的功能。

示例4-10　三个下拉列表框联动展示数据

（1）功能描述

为实现根据厂商、名牌、型号查询车型的功能，在页面中，设置三个下拉列表框，分别用于保存厂商、名牌、型号的数据。当用户在选择厂商时，名牌和型号下拉列表框随其数据变化而变化；当用户选择名牌时，型号下拉列表框随其所选数据变化而改变，从而实现了三个下拉列表框联动展示数据的功能。单击“查询”按钮时，显示用户所选择的全部选项。

（2）实现代码

新建一个HTML文件4-10.html,加入如代码清单4-10所示的代码。

代码清单 4-10　下拉列表框中的事件应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>列表框中事件应用</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
```

```
  </script>
  <style type="text/css">
        body{font-size:13px}
        .clsInit{width:435px;height:35px;line-height:35px;
        padding-left:10px}
        .clsTip{padding-top:5px;background-color:#eee;
        display:none}
        .btn {border:#666 1px solid;padding:2px;width:65px;
        float:right;margin-top:6px;margin-right:6px;
        filter: progid:DXImageTransform.Microsoft.
        Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8);}
  </style>
<script type="text/javascript">
    $(function() {
        function objInit(obj) {// 下拉列表框初始化
            return $(obj).html("<option> 请选择 </option>");
        }
        var arrData = { // 定义一个数组保存相关数据
            厂商 1: { 品牌 1_1: " 型号 1_1_1, 型号 1_1_2",
                      品牌 1_2: " 型号 1_2_1, 型号 1_2_2" },
            厂商 2: { 品牌 2_1: " 型号 2_1_1, 型号 2_1_2",
                      品牌 2_2: " 型号 2_2_1, 型号 2_2_2" },
            厂商 3: { 品牌 3_1: " 型号 3_1_1, 型号 3_1_2",
                      品牌 3_2: " 型号 3_2_1, 型号 3_2_2" }
        };
        $.each(arrData, function(pF) { // 遍历数据增加厂商项
            $("#selF").append("<option>" + pF + "</option>");
        });
        $("#selF").change(function() { // 厂商列表框选项改变事件
            objInit("#selT");
            objInit("#selC");
            $.each(arrData, function(pF, pS) {
                // 如果厂商选中项与数据匹配
                if ($("#selF option:selected").text() == pF) {
                    // 遍历数据增加品牌项
                    $.each(pS, function(pT, pC) {
                        $("#selT").append("<option>" + pT +"</option>");
                    });
                    // 品牌列表框选项改变事件
                    $("#selT").change(function() {
                        objInit("#selC");
                        $.each(pS, function(pT, pC) {
                            // 如果品牌选中项与数据匹配
                            if ($("#selT option:selected")
                                .text() == pT) {
                                // 遍历数据增加型号项
                                $.each(pC.split(","), function() {
                                    $("#selC").append("<option>"
                                    + this + "</option>");
                                });
                            }
```

```
                    });
                });
            }
        });
    });
    $("#Button1").click(function() { //注册按钮单击事件
        var strValue = "您选择的厂商:";
        strValue += $("#selF option:selected").text();
        strValue += " 您选择的品牌:";
        strValue += $("#selT option:selected").text();
        strValue += " 您选择的型号:";
        strValue += $("#selC option:selected").text();
        $("#divTip")
        .show()
        .addClass("clsTip")
        .html(strValue);//显示提示信息并增加样式
    });
})
</script>
</head>
<body>
    <div class="clsInit">
      厂商:<select id="selF"><option>请选择</option></select>
      品牌:<select id="selT"><option>请选择</option></select>
      型号:<select id="selC"><option>请选择</option></select>
      <input id="Button1" type="button" value="查询" class="btn" />
    </div>
    <div class="clsInit" id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行代码后的效果如图4-11所示。

图　4-11　三个列表框的联动应用

（4）代码分析

在示例4-10中，通过数组保存各下拉列表框所需要的数据，代码如下：

```
var arrData = { //定义一个数组保存相关数据
      厂商1: { 品牌1_1: "型号1_1_1,型号1_1_2",
               品牌1_2: "型号1_2_1,型号1_2_2" },
      厂商2: { 品牌2_1: "型号2_1_1,型号2_1_2",
               品牌2_2: "型号2_2_1,型号2_2_2" },
      厂商3: { 品牌3_1: "型号3_1_1,型号3_1_2",
               品牌3_2: "型号3_2_1,型号3_2_2" }
};
```

该数组的结构包含三次分隔部分：首先是通过冒号(：)完成第一次分隔，产生A与B两部分；然后，在第一次分隔后，对B部分的数据通过逗号(，)完成第二次分隔，产生独立的B0部分；最后，在第二次分隔后，在B0部分的数据通过冒号(：)完成第三次分隔，其示意图如图4-12所示。

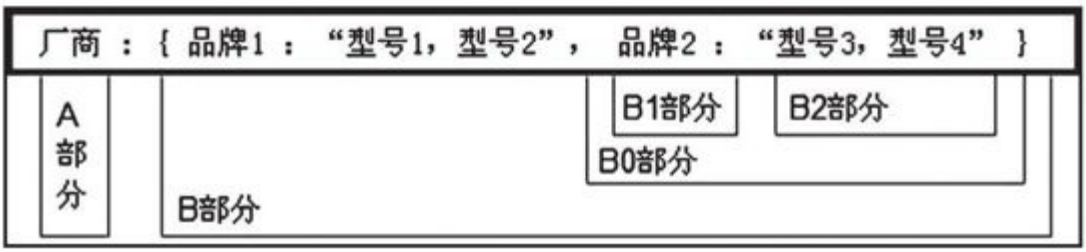


图　4-12　数组结构

在掌握数组结构后，接下来的任务就是通过jQuery中的each()方法遍历数组，获取需要的数组元素。首先通过第一轮的遍历数组，获取“厂商”的信息，并加入下拉列表框中，其代码如下所示：

```
$.each(arrData, function(pF) { //遍历数据增加厂商项
      $("#selF").append("<option>" + pF + "</option>");
 });
```

其中参数pF实际上就是图4-12中的A部分。

接下来进行第二轮的遍历数据，获取与“厂商”信息相匹配的“品牌”数组元素，并增加到第二个下拉列表框中，其代码如下：

```
$.each(arrData, function(pF, pS) {
        //如果厂商选中项与数据匹配
        if ($("#selF option:selected").text() == pF) {
                //遍历数据增加品牌项
                $.each(pS, function(pT, pC) {
                        $("#selT").append("<option>" + pT +"</option>");
                 });
                 ... 省略部分代码
          }
 });
```

代码中参数pS为图4-12中的B0部分，代码"$("#selF option:selected").text()==pF"表示“厂商”选项所选中的值与数组中某个“厂商”元素匹配，才遍历B0部分，其中参数pT为示意图中的B1部分，pC为B2部分，在B0部分遍历过程中，将与“厂商”相匹配的“品牌”数组元素增加到第二个下拉列表框中。

同理，“品牌”下拉列表框在change事件中，依然需要遍历整个数组，才能获取其相对应的型号数组元素数据，其实现的代码如下：

```
... 省略部分代码
$.each(arrData, function(pF, pS) {
        ... 省略部分代码
        $.each(pS, function(pT, pC) {
                //如果品牌选中项与数据匹配
                if ($("#selT option:selected")
                    .text() == pT) {
                    //遍历数据增加型号项
                    $.each(pC.split(","), function() {
                          $("#selC").append("<option>" + this + "</option>");
                    });
                 }
          });
   ... 省略部分代码
});
... 省略部分代码
```

由于参数pC所获取的数据中，需要通过“，”逗号分隔后才可使用，因此在第三次遍历元素时，使用了pC.split(",")方法获取分隔后

的数据，在增加时，this就是遍历时的各个分隔后的元素，通过append()方法，将该值增加到“型号”下拉列表框中。

最后，在按钮的单击事件中，通过jQuery中的val()方法获取各下拉列表框中的所选值，并进行累加，通过一个＜div＞元素展示在页面中，其代码如下：

```
...省略部分代码
var strValue = "您选择的厂商:";
strValue += $("#selF option:selected").text();
strValue += " 您选择的品牌:";
strValue += $("#selT option:selected").text();
strValue += " 您选择的型号:";
strValue += $("#selC option:selected").text();
$("#divTip")
.show()
.addClass("clsTip")
.html(strValue);//显示提示信息并增加样式
...省略部分代码
```

在显示时，由于默认是隐藏的，因此先通过show()方法，再增加一个样式，最后使用html()方法将全部累加的文本内容显示在＜div＞元素中。

4.7.3　列表中的导航菜单应用

在页面开发中，经常使用列表标记＜ul＞。在设计展示数据或导航菜单的页面中，列表的使用相当广泛。下面通过一个简单的示例，介绍在jQuery中，如何使用＜ul＞标记实现导航菜单的功能。

示例4-11　列表中的导航菜单应用

（1）功能描述

在页面表单中，分别展示某类产品的全部子类项，当用户将鼠标移动某项子类时，所选子类样式发生变化，并在该子类的右边以浮动的形式展示该类的全部产品；当用户将鼠标移出某项子类时，所选子类样式恢复到初始值，同时，隐藏已显示的全部子类产品。

（2）实现代码

新建一个HTML文件4-11.html,加入如代码清单4-11所示的代码。

代码清单 4-11　列表中的导航菜单应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>列表中的导航菜单应用</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          ul,li{list-style-type:none;padding:0px;margin:0px}
          .menu{width:190px;border:solid 1px #E5D1A1;
          background-color:#FFFDD2}
          .optn{width:190px;line-height:28px;
          border-top:dashed 1px #ccc}
          .content{padding-top:10px;clear:left}
          a{text-decoration:none;color:#666;padding:10px}
          .optnFocus{background-color:#fff;font-weight:bold}
          div{padding:10px}
          div img{float:left;padding-right:6px}
          span{padding-top:3px;font-size:14px;
          font-weight:bold;float:left}
          .tip{width:190px;border:solid 2px #ffa200;
          position:absolute;padding:10px;
          background-color:#fff;display:none}
          .tip li{line-height:23px;}
          #sort{position:absolute;display:none}
    </style>
  <script type="text/javascript">
      $(function() {
          var curY; //获取所选项的 Top 值
          var curH; //获取所选项的 Height 值
          var curW; //获取所选项的 Width 值
          var srtY; //设置提示箭头的 Top 值
          var srtX; //设置提示箭头的 Left 值
          var objL; //获取当前对象
          /*
          *设置当前位置数值
          *参数 obj 为当前对象名称
          */
          function setInitValue(obj) {
              curY = obj.offset().top
```

```
            curH = obj.height();
            curW = obj.width();
            srtY = curY + (curH / 2) + "px";          //设置提示箭头的 Top 值
            srtX = curW - 5 + "px";                   //设置提示箭头的 Left 值
        }
        //设置当前所选项的鼠标滑过事件
        $(".optn").mouseover(function() {
            objL = $(this);                           //获取当前对象
            setInitValue(objL);                       //设置当前位置
            var allY = curY - curH + "px";            //设置提示框的 Top 值
            objL.addClass("optnFocus");               //增加获取焦点时的样式
            objL.next("ul").show()
            .css({ "top": allY, "left": curW })       //显示并设置提示框的坐标
            $("#sort").show()
            .css({ "top": srtY, "left": srtX });      //显示并设置提示箭头的坐标
        })
         //设置当前所选项的鼠标移出事件
        .mouseout(function() {
            $(this).removeClass("optnFocus");         //删除获取焦点时的样式
            $(this).next("ul").hide();                //隐藏提示框
            $("#sort").hide();                        //隐藏提示箭头
        })
        $(".tip").mousemove(function() {
            $(this).show();                           //显示提示框
            objL = $(this).prev("li");                //获取当前的上级 li 对象
            setInitValue(objL);                       //设置当前位置
            objL.addClass("optnFocus");               //增加上级 li 对象获取焦点时的样式
            //显示并设置提示箭头的坐标
            $("#sort").show().css({ "top": srtY, "left": srtX });
        })
        .mouseout(function() {
            $(this).hide();                           //隐藏提示框
            $(this).prev("li").removeClass("optnFocus"); //删除获取焦点时的样式
            $("#sort").hide();                        //隐藏提示箭头
        })
    })
    </script>
</head>
<body>
    <ul>
        <li class="menu">
            <div>
                <img alt="" src="Images/icon.gif" />
                <span>电脑数码类产品</span>
            </div>
            <ul class="content">
                <li class="optn"><a href="#">笔记本</a></li>
                <ul class="tip">
                    <li><a href="#">笔记本 1</a></li>
                    <li><a href="#">笔记本 2</a></li>
                    <li><a href="#">笔记本 3</a></li>
                    <li><a href="#">笔记本 4</a></li>
                    <li><a href="#">笔记本 5</a></li>
                </ul>
                <li class="optn"><a href="#">移动硬盘</a></li>
```

```
                <ul class="tip">
                    <li><a href="#">移动硬盘1</a></li>
                    <li><a href="#">移动硬盘2</a></li>
                    <li><a href="#">移动硬盘3</a></li>
                    <li><a href="#">移动硬盘4</a></li>
                    <li><a href="#">移动硬盘5</a></li>
                </ul>
                <li class="optn"><a href="#">电脑软件</a></li>
                <ul class="tip">
                    <li><a href="#">电脑软件1</a></li>
                    <li><a href="#">电脑软件2</a></li>
                    <li><a href="#">电脑软件3</a></li>
                    <li><a href="#">电脑软件4</a></li>
                    <li><a href="#">电脑软件5</a></li>
                </ul>
                <li class="optn"><a href="#">数码产品</a></li>
                <ul class="tip">
                    <li><a href="#">数码产品1</a></li>
                    <li><a href="#">数码产品2</a></li>
                    <li><a href="#">数码产品3</a></li>
                    <li><a href="#">数码产品4</a></li>
                    <li><a href="#">数码产品5</a></li>
                </ul>
            </ul>
            <img id="sort" src="Images/sort.gif" alt=""/>
        </li>
    </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-13所示。

图　4-13　列表中导航菜单的应用

（4）代码分析

通过列表元素＜ul＞实现菜单导航条的功能，在很多大型的购物网站中随处可见。这样的导航菜单的功能之一是，当鼠标移到某个子菜单选项上，浮动显示子菜单相对应的全部产品，并且提示箭头指向该子菜单选项，同时，子菜单选项的样式发生变化。

为实现该功能，首先通过下列代码获取所有子菜单，并设置mousemove事件，其代码如下：

```
$(".optn").mouseover(function() {
...// 执行代码
 })
```

在子菜单选项的mouseover事件中，先通过自定义的函数setInitValue获取提示箭头的位置，即srtY与srtX变量值。然后根据当前所选的子菜单项的Top和Width值，设置浮动提示框的Top和Left值，即代码中的allY与curW变量的值，其代码如下：

```
... 省略部分代码
objL = $(this);                    // 获取当前对象
setInitValue(objL);                // 设置当前位置
var allY = curY - curH + "px"; // 设置提示框的 Top 值
... 省略部分代码
```

接下来，获取子菜单项对应的浮动提示框，并根据设置的位置显示提示框和提示箭头，同时，自身增加获取焦点时的样式，其实现的代码如下：

```
... 省略部分代码
objL.addClass("optnFocus");           // 增加获取焦点时的样式
objL.next("ul").show()
.css({ "top": allY, "left": curW })   // 显示并设置提示框的坐标
$("#sort").show()
.css({ "top": srtY, "left": srtX }); // 显示并设置提示箭头的坐标
... 省略部分代码
```

当子菜单中的鼠标移出时，触发mouseout事件，在该事件中，先移除所增加的焦点样式，然后隐藏对应的提示框和提示箭头。其实现

的代码如下：

```
... 省略部分代码
.mouseout(function() {                    // 设置当前所选项的鼠标移出事件
   $(this).removeClass("optnFocus"); // 删除获取焦点时的样式
   $(this).next("ul").hide();          // 隐藏提示框
   $("#sort").hide();                    // 隐藏提示箭头
 })
... 省略部分代码
```

由于子菜单在鼠标移出mouseout事件，隐藏了对应显示的提示框和提示箭头，因此，当用户将鼠标移出子菜单时，提示框和提示箭头都不见了，这不是我们想要的功能。

为了避免这样的效果，需要设置提示框的鼠标mousemove和mouseout事件，提示框注册mousemove事件的代码如下：

```
... 省略部分代码
$(".tip").mousemove(function() {
    ...// 执行代码

 })
... 省略部分代码
```

在提示框的mousemove事件中，为了改变所选子菜单的样式，先获取子菜单，其代码如下：

```
objL = $(this).prev("li"); // 获取当前的上级 li 对象
```

然后，通过自定义函数setInitValue，获取提示箭头的位置，并显示在页面中，同时，再增加获取焦点时的样式。其代码如下：

```
... 省略部分代码
setInitValue(objL); // 设置当前位置
objL.addClass("optnFocus");// 增加上级 li 对象获取焦点时的样式
$("#sort").show().css({ "top": srtY, "left": srtX }); // 显示并设置提示箭头的坐标
... 省略部分代码
```

在提示框的mouseout事件中，移除所增加的焦点样式，并隐藏提示框和提示箭头，其实现的代码如下：

```
... 省略部分代码
.mouseout(function() {
   $(this).hide(); // 隐藏提示框
   $(this).prev("li").removeClass("optnFocus"); // 删除获取焦点时的样式
   $("#sort").hide(); // 隐藏提示箭头
 })
... 省略部分代码
```

4.7.4　网页选项卡的应用

在页面中，除使用＜ul＞列表标记实现滑动效果的菜单导航条外，还用于网页选项卡的设计。选项卡的功能十分简单，通过单击标题实现内容隐藏或显示的切换，由于它可以在有限的空间中展示大量的数据，因此，被广泛使用在各综合性的门户网站中。下面通过一个简单的示例，介绍网页选项卡快速实现的方法。

示例4-12　网页选项卡的应用

（1）功能描述

在页面中设置三个不同名称的选项卡，单击某个选项卡时，下面相对应的区域显示其内容信息，同时选项卡的背景色与内容信息的背景色浑然一体，并且字体加粗，表示处于选中状态。

（2）实现代码

新建一个HTML文件4-12.html，加入如代码清单4-12所示的代码。

代码清单 4-12　网页选项卡的应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>网页选项卡应用</title>
    <script type="text/javascript"
```

```
        src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
        body{font-size:13px}
        ul,li {margin:0;padding:0;list-style:none}
        #menu li {text-align:center;float:left;padding:5px;
        margin-right:2px;width:50px;cursor:pointer}
        #menu li.tabFocus {width:50px; font-weight:bold;
        background-color:#f3f2e7;border:solid 1px #666;
        border-bottom:0;z-index:100;
        position:relative}
        #content {width:260px;height:80px;padding:10px;
        background-color:#f3f2e7;clear:left;
        border:solid 1px #666;
        position:relative;top:-1px}
        #content li{display:none}
        #content li.conFocus{display:block}
    </style>
    <script type="text/javascript">
        $(function() {
            // 带参数遍历各个选项卡
            $("#menu li").each(function(index) {
                $(this).click(function() { // 注册每个选卡的单击事件
                  // 移除已选中的样式
                  $("#menu li.tabFocus").removeClass("tabFocus");
                  $(this).addClass("tabFocus"); // 增加当前选中项的样式
                  // 显示选项卡对应的内容并隐藏未被选中的内容
                  $("#content li:eq(" + index + ")").show().siblings().hide();
                });
            });
        })
    </script>
</head>
<body>
    <ul id="menu">
        <li class="tabFocus">家居</li>
        <li>电器</li>
        <li>二手</li>
    </ul>
    <ul id="content">
        <li class="conFocus">我是家居的内容</li>
        <li>欢迎您来到电器城</li>
        <li>二手市场，产品丰富多彩</li>
    </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图4-14所示。

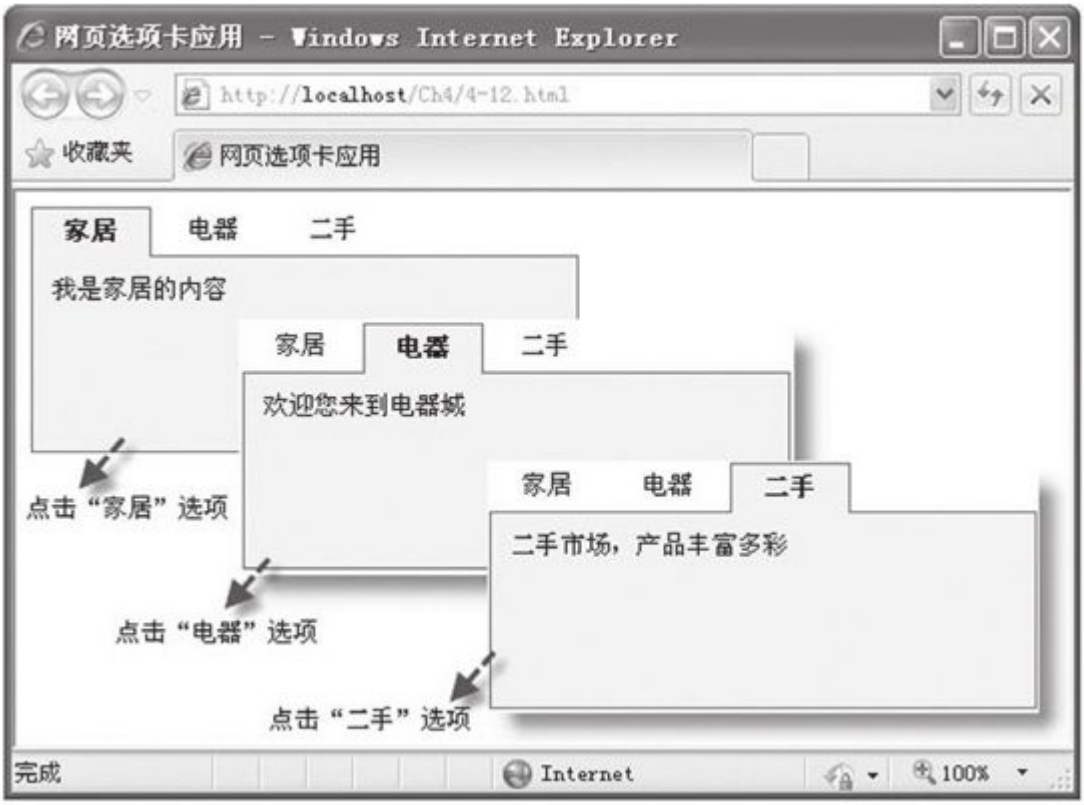


图　4-14　网页选项卡应用

（4）代码分析

在源码中，为了找到每个选项卡，先带参数遍历整个选项卡，其实现的代码如下：

```
$("#menu li").each(function(index) { // 遍历整个选项卡
    // 执行代码
    ...
});
```

在选项卡的事件遍历中，注册每个选项卡的单击事件，代码如下：

```
$(this).click(function() { // 注册每个选卡的单击事件
    // 执行代码
      ...
});
```

在选项卡的单击事件中，先移除以前被选中的选项卡焦点样式，然后增加当前选项卡的焦点样式；在显示对应内容时，先通过$("#content li:eq("+index+")")语句获取当前选项卡所对应的内容，并显示该内容，同时，通过siblings()方法隐藏其对应的兄弟内容。其实现的代码如下：

```
...省略部分代码
$("#menu li.tabFocus").removeClass("tabFocus"); // 移除已选中的样式
$(this).addClass("tabFocus"); // 增加当前选中项的样式
$("#content li:eq(" + index + ")").show().siblings().hide();
// 显示选项卡对应的内容并隐藏未被选中的内容
...省略部分代码
```

4.8　综合案例分析——删除数据时的提示效果在项目中的应用

4.8.1　需求分析

经分析，该案例的需求如下：

1）当用户单击“删除”按钮时，整个页面背景类似于关机效果，“删除”提示框突出显示，用户可以选“关闭”按钮，或单击“确定”或“取消”的操作。

2）删除提示框一直居中显示，不论页面大小发生如何变化，这个提示框始终居中显示。

3）如果对某条记录打勾，当用户单击提示框中的“确定”按钮时，将在页面中删除该条记录，同时关闭提示框，页面背景恢复正常。

4.8.2　界面效果

该案例的界面效果如下所示。

1）用户单击“删除”按钮前，其实现的界面如图4-15所示。

图　4-15　记录删除前

2）当用户单击“删除”按钮后，出现删除提示对话框，其实现的界面如图4-16所示。

图　4-16　删除记录的提示对话框

3）当页面大小发生变化时，提示对话框一直居中显示，其实现的界面如图4-17所示。

图　4-17　页面变化后的提示对话框

4.8.3　功能实现

在该项目中，新建一个HTML文件delete.html，加入如代码清单4-13所示的代码。

代码清单 4-13　删除记录时的提示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>删除记录时的提示效果</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
     <style type="text/css">
          body{font-size:13px}
          .divShow{line-height:32px;height:32px;
          background-color:#eee;width:280px;padding-left:10px}
          .divShow span{padding-left:50px}
          .dialog{width:360px;border:solid 5px #666;
          position:absolute;display:none;z-index:101}
          .dialog .title{background-color:#fbaf15;padding:10px;
          color:#fff;font-weight:bold}
          .dialog .title img{float:right}
          .dialog .content{background-color:#fff;padding:25px;height:60px}
          .dialog .content img{float:left}
          .dialog .content span{float:left;padding-top:10px;
          padding-left:10px}
          .dialog .bottom{text-align:right;
          padding:10px 10px 10px 0px;background-color:#eee}
          .mask {width:100%;height:100%; background-color:#000;
          position:absolute;top:0px;left:0px;
```

```
        filter:alpha(opacity=30);display:none;z-index:100)
        .btn {border:#666 1px solid;padding:2px;width:65px;
        filter:progid:DXImageTransform.Microsoft
        .Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8);}
</style>
 <script type="text/javascript">
     $(function() {
         $("#Button1").click(function() { //注册删除按钮点击事件
             $(".mask").show();   //显示背景色
             showDialog();        //设置提示对话框的 Top 与 Left
             $(".dialog").show(); //显示提示对话框
         })
         /*
         * 根据当前页面与滚动条位置，设置提示对话框的 Top 与 Left
         */
         function showDialog() {
             var objW = $(window);     //当前窗口
             var objC = $(".dialog"); //对话框
             var brsW = objW.width();
             var brsH = objW.height();
             var sclL = objW.scrollLeft();
             var sclT = objW.scrollTop();
             var curW = objC.width();
             var curH = objC.height();
             var left = sclL + (brsW - curW) / 2;    //计算对话框居中时的左边距
             var top = sclT + (brsH - curH) / 2;     //计算对话框居中时的上边距
             objC.css({ "left": left, "top": top });//设置对话框在页面中的位置
         }

         $(window).resize(function() {//页面窗口大小改变事件
             if (!$(".dialog").is(":visible")) {
                 return;
             }
             showDialog(); //设置提示对话框的 Top 与 Left
         });

         $(".title img").click(function() { //注册关闭图片点击事件
             $(".dialog").hide();
             $(".mask").hide();
         })

         $("#Button3").click(function() {//注册取消按钮点击事件
             $(".dialog").hide();
             $(".mask").hide();
         })

         $("#Button2").click(function() {//注册确定按钮点击事件
             $(".dialog").hide();
             $(".mask").hide();
```

```
                if ($("input:checked").length != 0) {// 如果选择了删除行
                    $(".divShow").remove(); // 删除某行数据
                }
            })
        })
    </script>
</head>
<body>
    <div class="divShow">
        <input id="Checkbox1" type="checkbox" />
        <a href="#">这是一条可删除的记录</a>
        <span>
            <input id="Button1" type="button" value="删除" class="btn" />
        </span>
    </div>
    <div class="mask"></div>
    <div class="dialog">
        <div class="title">
            <img src="Images/close.gif" alt="点击可以关闭" />删除时提示
        </div>
        <div class="content">
            <img src="Images/delete.jpg" alt="" />
                <span>您真的要删除该条记录吗? </span>
        </div>
        <div class="bottom">
            <input id="Button2" type="button"
                value="确定" class="btn"/>  
            <input id="Button3" type="button"
                value="取消" class="btn"/>
        </div>
    </div>
</body>
</html>
```

4.8.4 代码分析

本案例核心功能是当用户在删除某条记录时，弹出一个提示对话框，等待用户单击“确定”按钮后，才能直接删除。为了实现该功能，首先注册“删除”按钮的单击事件，其代码如下：

```
$("#Button1").click(function() { // 注册删除按钮点击事件
    // 执行代码
    ...
})
```

在“删除”按钮单击事件中，先显示设置好的背景元素，再通过自定义的函数showDialog，设置好对话框的显示位置，最后显示提示对话框，其代码如下：

```
... 省略部分代码
$(".mask").show();     // 显示背景色
showDialog();          // 设置提示对话框的 Top 与 Left
$(".dialog").show(); // 显示提示对话框
... 省略部分代码
```

在自定义的函数showDialog中，根据窗口的长与宽和滚动条的Top与Left值及对话框自身的长与宽，计算出使提示对话框始终居中的坐标Top和Left变量值，并根据该变量值，设置对话框的Top与Left属性值，其代码如下：

```
...省略部分代码
function showDialog() {
   var objW = $(window);     // 当前窗口
   var objC = $(".dialog"); // 对话框
   var brsW = objW.width();
   var brsH = objW.height();
   var sclL = objW.scrollLeft();
   var sclT = objW.scrollTop();
   var curW = objC.width();
   var curH = objC.height();
   var left = sclL + (brsW - curW) / 2;    //计算对话框居中时的左边距
   var top = sclT + (brsH - curH) / 2;     //计算对话框居中时的上边距
   objC.css({ "left": left, "top": top });//设置对话框在页面中的位置
 }
...省略部分代码
```

为了更好地理解上述自定义的函数showDialog中代码的功能，我们通过图4-18说明如何才能使提示对话框居中。

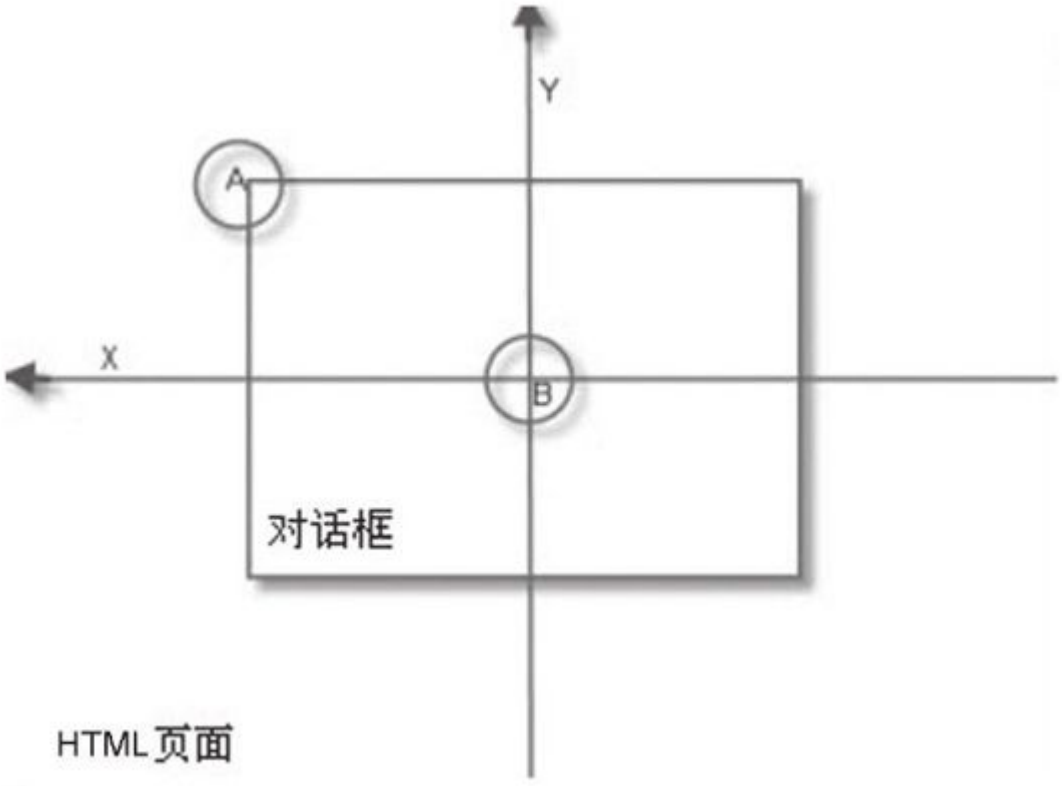


图 4-18 对话框在页面中居中示意图

我们清楚地看到，如果对话框的原点与页面的原点完全重合，即都在B点，对话框则居中，那么，对话框的坐标，实际上就是A点的位置，它的Left值为页面的Width值减掉对话框的Width值后除以2，再加上滚动条的Left值，即如下代码：

```
var left = sclL + (brsW - curW) / 2;
```

Top值为页面的Height值减掉对话框的Height值后除以2，再加上滚动条的Top值，即如下代码：

```
var top = sclT + (brsH - curH) / 2;
```

获取对话框的变量值后，再在窗口的resize事件中，重新根据上述方法进行提示对话框坐标的计算，以保证当页面大小发生变化时，对话框始终居中显示，resize事件执行的代码如下：

```
...省略部分代码
$(window).resize(function() {// 页面窗口大小改变事件
    if (!$(".dialog").is(":visible")) {
            return;
     }
    showDialog(); // 设置提示对话框的 Top 与 Left
 });
...省略部分代码
```

代码if(!$(".dialog").is(":visible"))表示，如果没有出现对话框，则不执行该事件中的代码，以避免不必要的资源占用。

当用户单击“确定”按钮时，首先检测是否有选中的可删除记录，然后再进行删除的操作，其代码如下：

```
...省略部分代码
$("#Button2").click(function() {// 注册确定按钮点击事件
    $(".dialog").hide();
    $(".mask").hide();
    if ($("input:checked").length != 0) {// 如果选择了删除行
            $(".divShow").remove(); // 删除某行数据
        }
 })
...省略部分代码
```

当用户单击“关闭”图片按钮或单击“取消”按钮时，均执行同样的代码事件，即先隐藏居中显示的对话框，同时，隐藏背景，使页面恢复到初始状态，其实现的代码如下：

```
...省略部分代码
$(".dialog").hide();
$(".mask").hide();
...省略部分代码
```

4.9　本章小结

在DOM操作中，离不开事件的触发。本章以循序渐进的方式，详细地介绍了jQuery中的各种常用的事件类型，通过简单的事件应用示例，阐述各类元素注册事件的方法。本章最后还介绍如何使用jQuery中的注册事件，实现目前页面体验中最为常见的各种效果。

第5章　jQuery的动画与特效

本章内容

显示与隐藏

滑动

淡入淡出

自定义动画

动画效果综述

综合案例分析——动画效果浏览相册中的图片

本章小结

如何能最大化地优化页面的用户体验度，是每个前端页面开发人员在设计页面时需要考虑的一个重要问题。无可置疑，jQuery中众多的动画与特效方法为提高页面的用户体验度带来了极大的方便，通过少量的几行代码，就可以实现元素的飞动、淡入淡出等动画效果，还可以自定义各种动画效果。

5.1　显示与隐藏

在页面中，元素的显示与隐藏是使用最频繁的操作，在传统的JavaScript中，一般通过改变元素显示的方式实现，下列代码将ID号为"p1"的元素显示出来：

```
document.getElementById("p1").style.display = "block";
```

如果想隐藏该元素，可以使用下列代码：

```
document.getElementById("p1").style.display = "none";
```

而在jQuery中，元素的显示与隐藏的方法比传统的JavaScript要多，并且实现的效果也很优雅。下面我们逐步介绍在jQuery中实现元素的显示与隐藏的方法。

5.1.1　show()与hide()方法

我们在前面的章节中不止一次地使用过show()与hide()方法，前者是显示页面中的元素，等同于下列jQuery代码：

```
$("#p1").css("display":"block");
```

后者是隐藏页面中的元素，与show()方法正好相反，等同于下列jQuery代码：

```
$("#p1").css("display":"none");
```

下面通过一个示例介绍这两个jQuery方法的使用过程。

示例5-1　使用show()与hide()方法显示和隐藏文本

（1）功能描述

在显示大量文本内容时，为了能显示更多的段落内容，有时仅显示一部分的提要，隐藏另一部分的内容。当用户需要查看这些隐藏的内容时，只要单击页面中的“显示”链接就可以；查看完后，再单击“隐藏”链接便将该部分内容再次隐藏起来。

（2）实现代码

新建一个HTML文件5-1.html,加入如代码清单5-1所示的代码。

代码清单 5-1　show() 与 hide() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
    <title>show() 与 hide() 方法 </title>
    <script type="text/javascript"
          src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .artFram{border:solid 1px #ccc;background-color:#eee;
                width:260px;padding:8px;word-break:break-all}
          .artList{line-height:1.8em}
    </style>
    <script type="text/javascript">
        $(function() {
            var $link = $(".artList a");        //获取单击文本
            var $hide = $(".artList :eq(2)");   //获取隐藏的段落
            $link.click(function() {            //文本单击事件
                if ($(this).html() == "显示") {//如果未显示
                    $(this).html("隐藏");       //改变文本内容
                    $hide.show();               //显示隐藏的文本
                } else {
                    $(this).html("显示");
                    $hide.hide();
                }
            })
        })
    </script>
</head>
<body>
    <div class="artFram">
        <div class="artList">
            <span>     </span>
            <span>一个优秀的程序开发员，除需掌握专业的开发语言，还要具有 </span>
            <span style="display:none">执着、沉稳、细致的专业素质 </span>
             <a href="javascript:void(0)">显示 </a>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-1所示。

图　5-1　show()与hide()方法

（4）代码分析

在显示隐藏部分的内容时，为了获取当前显示状态，先通过语句if($(this).html()=="显示")进行检测，然后根据检测结果执行不同的操作。

5.1.2　动画效果的show()与hide()方法

jQuery中的show()与hide()方法，还可以完成有动画效果的显示与隐藏，只需在方法的括号中加入相应的参数即可，其调用的语法格式如下。

动画效果的显示功能如下所示：

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

动画效果的隐藏功能如下所示：

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

方法中的参数speed表示执行动画时的速度，该速度有3个默认字符值"slow"、"normal"、"fast"，其对应的速度分别是“0.6秒”、“0.4秒”、“0.2秒”；如果不使用默认的字符值，也可以直接写入数字，如“3000”表示该动画执行的速度为3 000毫秒。可选型参数[callback]为在动画完成时执行的回调函数，该函数每个元素执行一次。下面通过示例来介绍这两个方法的使用。

示例5-2　使用show()与hide()方法动画显示和隐藏图片

（1）功能描述

在页面中单击“显示”链接，通过show()方法以动画的方式显示一幅图片，同时，在方法中执行一个回调函数，用于改变图片的边框样式；单击已显示的图片时，通过hide()方法以动画的方式隐藏该图片。

（2）实现代码

新建一个HTML文件5-2.html，加入如代码清单5-2所示的代码。

代码清单 5-2　动画效果的 show() 与 hide() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画效果的 show() 与 hide() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{display:none;cursor:pointer}
    </style>
    <script type="text/javascript">
        $(function() {

            $("a").click(function() {// 显示链接点击事件
                $("img").show(3000, function() {// 显示完成时执行的函数
                    $(this).css("border", "solid 1px #ccc");
                })
            })
            $("img").click(function() {// 显示图片的点击事件
                $(this).hide(3000);// 动画效果隐藏
            })
        })
    </script>
</head>
<body>
    <a href="javascript:void(0)">显示 </a>
    <img src="Images/img04.jpg" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-2所示。

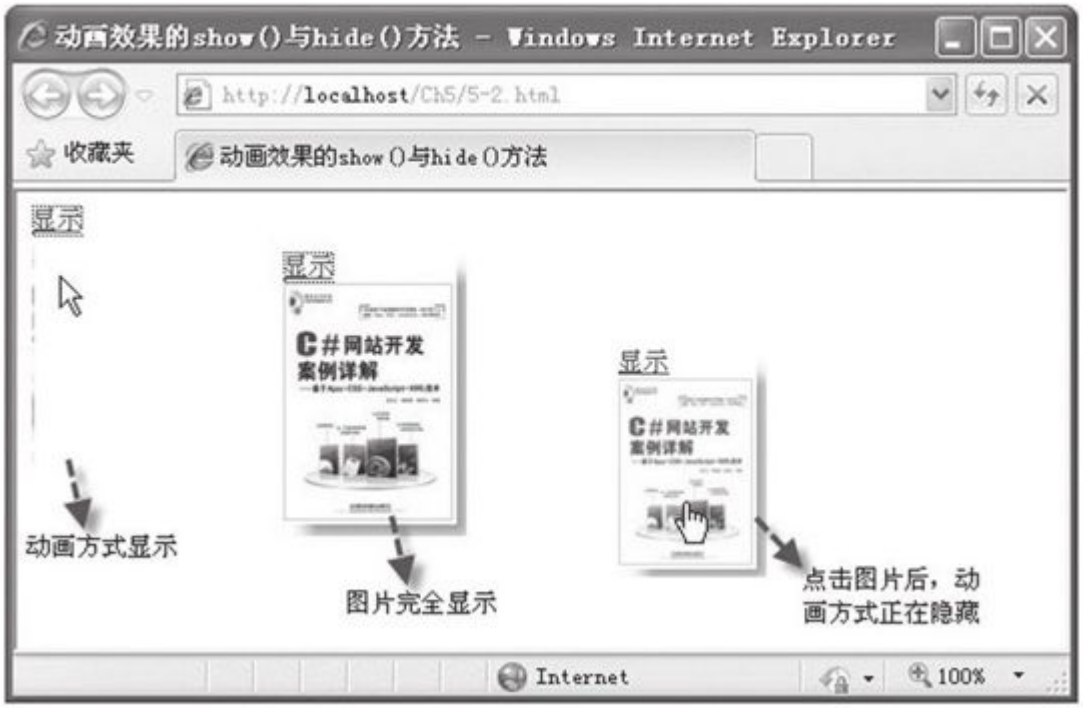


图　5-2　动画效果的show()与hide()方法

5.1.3 toggle()方法

在使用show()或hide()方法显示或隐藏页面元素时，为了正确执行切换显示的动作，通常需要检测当前元素的显示状态，然后根据该状态再执行元素是否显示或隐藏。这样一来，代码显得有些冗长，在jQuery中，解决这个问题可以使用toggle()方法。该方法的功能就是切换元素可见状态，即如果是显示状态，则变成隐藏状态；如果是隐藏状态，则变成显示状态。该方法有三种调用的形式，代码如下所示。

形式一，无参数调用格式：

```
toggle()
```

形式二，逻辑参数调用格式：

```
toggle(switch)
```

参数switch为一个布尔值，即true或false。当该值为true时，显示元素；否则，隐藏元素。

形式三，动画效果调用格式：

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

其中参数speed和可选参数[callback]与方法show(speed, [callback])中的参数所表示的意义是一样的，在此不作赘述。

下面通过示例来介绍toggle的使用方法。

示例5-3　使用toggle()方法切换元素可见状态

（1）功能描述

为了更直观地区分开toggle()方法中的三种形式，在页面中设置了三个按钮分别对应三种调用的形式。通过单击按钮，触发toggle()方法，实现图片的切换显示效果。

（2）实现代码

新建一个HTML文件5-3.html，加入如代码清单5-3所示的代码。

代码清单 5-3　toggle() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>toggle() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:180px}
           .divFrame .divMenu{float:left}
           .divFrame .divContent{float:right}
           .divFrame .divContent img{display:none}
           .btn {border:#666 1px solid;padding:2px;
           width:80px;margin-bottom:5px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
           $("input:eq(0)").click(function() { //无参数方法
                $("img").toggle();
           })
           $("input:eq(1)").click(function() { //根据参数 switch 显示
                var intI = 0;
                var blnA = intI > 1; //获取逻辑值
                $("img").toggle(blnA);
           })
           $("input:eq(2)").click(function() { //动画方式显示
               $("img").toggle(3000, function() {
                   $(this) //以动画方式显示，并执行回调函数


                       .css({ "border": "solid 1px #ccc", "padding": "2px" });
               });
           })
        })
    </script>
</head>
<body>
    <div class="divFrame">
         <div class="divMenu">
              <input id="Button1" type="button" value=" 无参数 " class="btn" /><br />
              <input id="Button2"  type="button" value=" 逻辑显示 " class="btn" /><br />
              <input id="Button3"  type="button" value=" 动画显示 " class="btn" />
         </div>
         <div class="divContent">
              <img src="Images/img03.jpg" alt="" />
         </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-3所示。

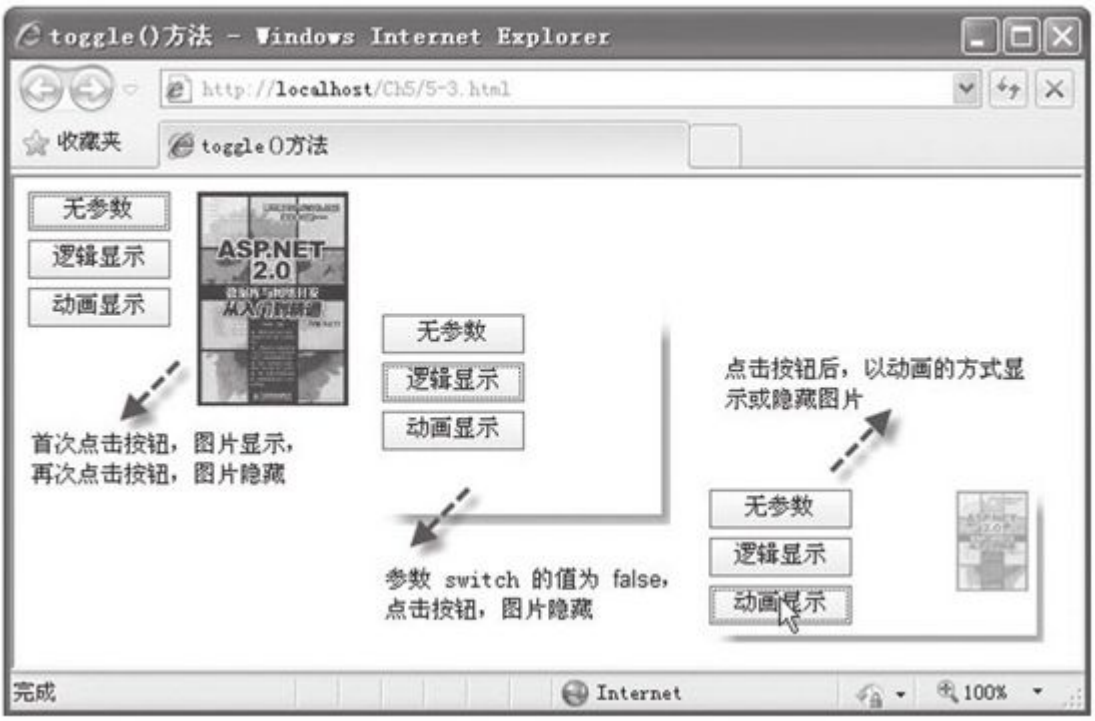


图　5-3　toggle()方法切换元素可见状态

（4）代码分析

无论是show()和hide()还是toggle()方法，当以动画效果切换页面元素可见状态时，其元素的width、height、padding和margin属性都将以动画的效果展示。

5.2 滑动

在jQuery中，还有一种滑动的动画效果改变元素的高度，即“拉窗帘”的效果。

5.2.1 slideDown()与slideUp()方法

要实现元素的滑动效果，需要调用jQuery中的两个方法，一个是slideDown()，另一个是slideUp()，其调用的语法格式介绍如下。

slideDown()方法的格式如下：

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

其功能是以动画的效果将所选择元素的高度向下增大，使其呈现一种“滑动”的效果，而元素的其他属性并不发生变化；参数speed为动画显示的速度，可选项[callback]为动画显示完成后，执行的回调函数。

slideUp()方法的格式如下：

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

其功能是以动画的效果将所选择元素的高度向上减小，同样也是仅改变高度属性，其包含的参数作用与slideDown()方法一样。

下面通过示例来介绍这两种方法。

示例5-4　使用slideDown()与slideUp()方法实现滑动效果

（1）功能描述

在页面中，单击“标题”栏时，通过slideUp()方法，以动画的效果将“内容”栏中的图片向上滑动，直到完全看不见，并改变“标题”栏中的内容；再次单击“标题”栏时，通过slideDown()方法，将“内容”栏中的图片向下滑动，直到全部显示，“标题”栏中的内容也同时发生相应改变。

（2）实现代码

新建一个HTML文件5-4.html,加入如代码清单5-4所示的代码。

代码清单 5-4　slideDown() 与 slideUp() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>slideDown() 与 slideUp() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .divFrame{width:86px;border:solid 1px #666}
          .divFrame .divTitle{padding:5px;background-color:#eee}
          .divFrame .divContent{padding:8px;}
          .divFrame .divContent img{border:solid 1px #ccc;padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            var blnShow = false;          // 初始化一个布尔变量值
            var $Title = $(".divTitle"); // 定义变量获取标题部分
            var $Tip = $("#divTip");      // 定义变量获取提示元素
            $Title.click(function() {     // 点击标题部分事件


                if (!blnShow) {
                    // 图片高度向上减小，执行完成后，回调一个函数
                    $("img").slideUp(3000, function() {
                        $Tip.html(" 关闭成功! ");
                    })
                    $(this).html(" 显示图片 ");   // 改变标题内容
                    blnShow = true;              // 改变布尔变量值
                }
                else {
                    $Tip.html("");               // 清空提示内容
                    $("img").slideDown(3000); // 图片高度向下增大
                    $(this).html(" 隐藏图片 ");
                    blnShow = false;
                }
            })
        })
    </script>
</head>
<body>
      <div class="divFrame">
           <div class="divTitle"> 隐藏图片 </div>
           <div class="divContent">
                <img src="Images/img02.jpg" alt=""/>
                <div id="divTip"></div>
           </div>
      </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-4所示。

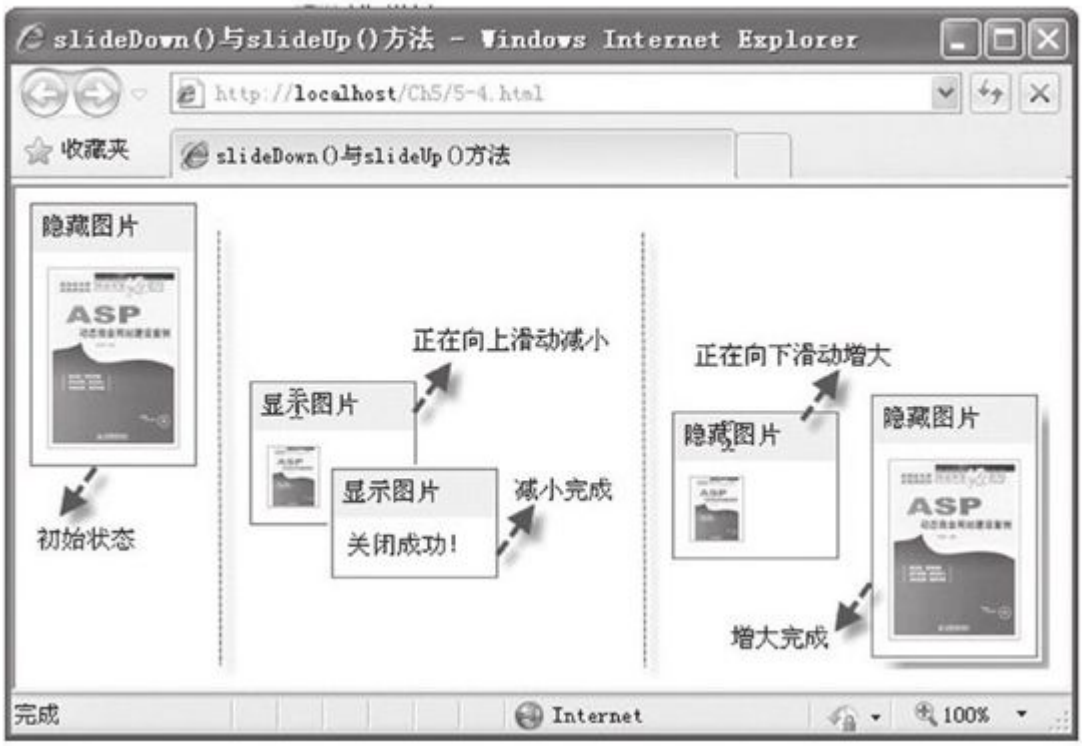


图　5-4　slideDown()与slideUp()方法

（4）代码分析

无论是slideDown()还是slideUp()方法，它们的动画效果仅是减小或增加元素的高度，同时，如果元素有margin或padding值，这些属性也会以动画效果一起发生改变。

5.2.2　slideToggle()方法

在示例5-4中，为了判断当前元素的显示状态，先定义了一个变量值blnShow，根据这个变量值，决定是执行slideDown()还是slideUp()方法，即元素是向上减小还是向下增大。在jQuery中，通过slideToggle()方法，无需定义变量，该方法可以根据当前元素的显示状态，自动进行切换，其调用的语法格式如下所示：

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

该方法的功能是以动画的效果切换所选择元素的高度，即：如果高，则减小；如果低，则增大。同时，在每次执行动画完成后，可执行一个用于回调的函数。其包含的参数功能与方法slideDown()或slideUp()一样，在此不赘述。下面通过示例来介绍这个方法。

示例5-5　使用slideToggle()方法的动画效果自动切换图片高度

（1）功能描述

在页面中，使用一个＜div＞标记包含一个＜img＞图片，当单击＜div＞元素时，通过slideToggle()方法，以动画的效果自动切换图片高度状态。

（2）实现代码

新建一个HTML文件5-5.html，加入如代码清单5-5所示的代码。

代码清单 5-5　slideToggle() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>slideToggle() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #666;background-color:#eee;
           padding:5px;width:149px}
           .divFrame img{border:solid 1px #eee;padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".divFrame").click(function() { //div 元素点击事件
                // 图片高度状态自动切换，并执行一个回调函数
                $("img").slideToggle(3000, function() {
                    $("img").css("border", "solid 1px #ccc");
                })
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <img src="Images/img01.jpg" alt="" title=" 风景图 " />


    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-5所示。

图　5-5　slideToggle()方法

5.3　淡入淡出

在前两节中，我们介绍了通过动画效果切换元素显示状态和改变元素高度，除上述效果外，在jQuery中，还可以实现通过元素渐渐变换背景色的动画效果来显示或隐藏元素，即淡入淡出效果。

5.3.1　fadeIn()与fadeOut()方法

show()和hide()方法与fadeIn()和fadeOut()方法相比较，相同之处是都切换元素的显示状态，不同之处在于，前者的动画效果使用元素的width与height属性都发生了变化，而后者仅是改变元素的透明度，并不修改其他属性。fadeIn()和fadeOut()方法调用格式介绍如下。

fadeIn()方法的格式如下：

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

该方法的功能是通过改变所选元素透明度，实现淡入的动画效果，并在完成时，可执行一个回调的函数。参数speed为动画效果的速度，可选项参数[callback]为动画完成时执行的函数。

fadeOut()方法的格式如下：

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

该方法的功能是通过改变所选元素透明度，实现淡出的动画效果，其包含参数的功能与fadeIn()方法一样，在此不再赘述。下面通过示例来讲解这两个方法。

示例5-6　使用fadeIn()和fadeOut()方法实现淡入淡出效果

（1）功能描述

在页面中设置两个按钮，一个用于淡入图片，另一个用于淡出图片；无论是淡入还是淡出图片，效果完成后，都执行一个回调函数，显示当前操作的结果。

（2）实现代码

新建一个HTML文件5-6.html,加入如代码清单5-6所示的代码。

代码清单 5-6　fadeIn() 和 fadeOut() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> fadeIn() 和 fadeOut() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          .divFrame{border:solid 1px #666;
          width:188px;text-align:center;}
          .divFrame .divTitle{background-color:#eee;
          padding:5px 0px 5px 0px}
          .divFrame .divContent{padding:5px 0px 5px 0px}
          .divFrame .divContent img{border:solid 1px #eee;
          padding:2px}
          .divFrame .divTip{position:absolute;
          padding:90px 0px 0px 60px;font-size:13px;
          font-weight:bold}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $img = $("img");      // 获取图片元素对象
            $tip = $(".divTip"); // 获取提示元素对象
            $("input:eq(0)").click(function() { // 第一个按钮单击事件
                $tip.html(""); // 清空提示内容
                // 在 3000 毫秒中淡入图片，并执行一个回调函数
                $img.fadeIn(3000, function() {
                    $tip.html(" 淡入成功！ ");
                })
            })
            $("input:eq(1)").click(function() { // 第二个按钮单击事件
                $tip.html(""); // 清空提示内容
                // 在 3000 毫秒中淡出图片，并执行一个回调函数
                $img.fadeOut(3000, function() {
                    $tip.html(" 淡出成功！ ");
                })
            })
        })


    </script>
</head>
<body>
     <div class="divFrame">
          <div class="divTitle">
              <input id="Button1" type="button"
                     value="fadeIn" class="btn" />
              <input id="Button2" type="button"
                     value="fadeOut" class="btn" />
          </div>
          <div class="divContent">
              <div class="divTip"></div>
              <img src="Images/img05.jpg" alt="" title=" 风景图片 " />
          </div>
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-6所示。

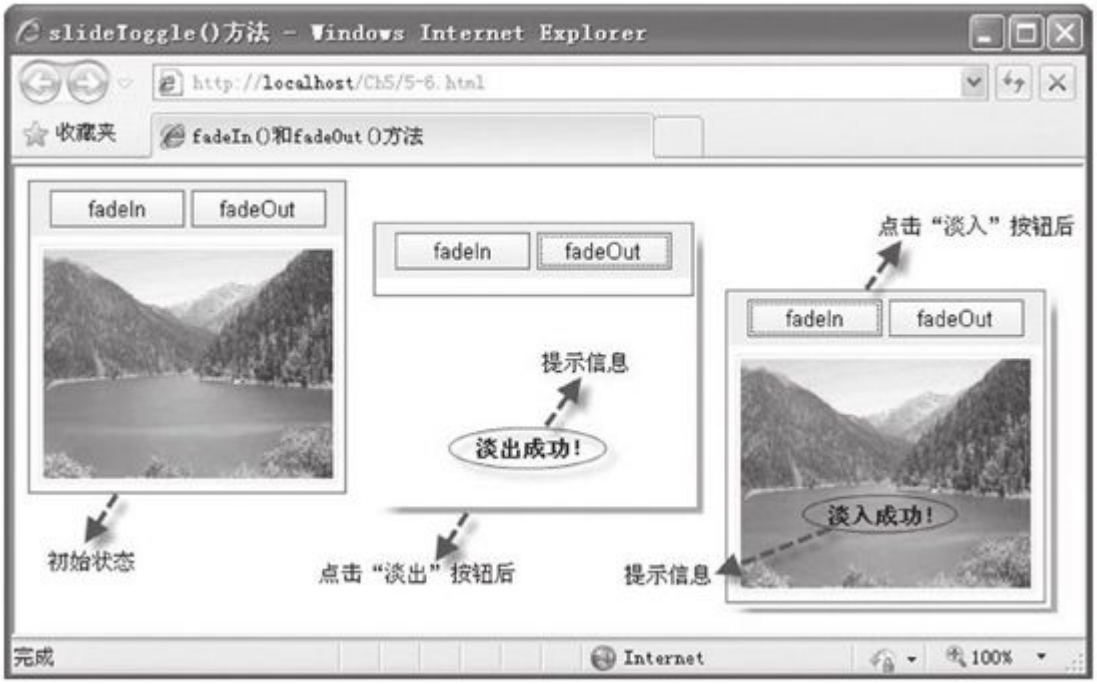


图　5-6　fadeIn()和fadeOut()方法

5.3.2 fadeTo()方法

在jQuery中，fadeIn()和fadeOut()方法通过动画效果，改变元素的透明度切换元素显示状态，其透明度从0.0到1.0淡出或从1.0到0.0淡入，从而实现淡入淡出的动画效果；如果要将透明度指定到某一个值，则需要调用fadeTo()方法。其调用的语法格式为：

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

该方法的功能是将所选择元素的不透明度以动画的效果调整到指定的不透明度值，动画完成时，可以执行一个回调函数，参数speed为动画效果的速度，参数opacity为指定的不透明值，取值范围是0.0～1.0，可选项参数[callback]为动画完成时执行的函数。下面通过示例来讲解。

示例5-7 使用fadeTo()方法改变图片透明度

（1）功能描述

为了更好地展示各种透明度值的效果，在页面中创建一个下拉列表框，用于保存各种透明度值，当选择下拉列表框中某选项时，页面中图片的透明度便指定为该选项值。

（2）实现代码

新建一个HTML文件5-7.html,加入如代码清单5-7所示的代码。

代码清单 5-7　fadeTo() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>fadeTo() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #666;
           width:197px;text-align:center;}
           .divFrame .divTitle{background-color:#eee;
           padding:5px 0px 5px 0px}
           .divFrame .divContent{padding:5px 0px 5px 0px}
           .divFrame .divContent img{border:solid 1px #eee;
           padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            var $img = $("img");                //获取图片元素对象
            var $sel = $("select");             //获取下拉列表框对象
            $sel.change(function() {            //下拉列表框选项改变事件
                var fltValue = $sel.val();      //获取选中的值
                $img.fadeTo(3000, fltValue);  //改变图片的透明度
            })
        })
    </script>
</head>
<body>
     <div class="divFrame">
           <div class="divTitle">
               <select id="Select1">
                   <option value="0.2">0.2</option>
                   <option value="0.4">0.4</option>
                   <option value="0.6">0.6</option>
                   <option value="0.8">0.8</option>
                   <option value="1.0"
                           selected="selected">1.0</option>
               </select>
           </div>
           <div class="divContent">
               <img src="Images/img06.jpg" alt="" title="风景图片" />
           </div>


      </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-7所示。

图　5-7　fadeTo()方法改变透明度

5.4　自定义动画

上面几节介绍的动画效果都是元素局部属性发生变化，如高度、宽度、可见性等。在jQuery中，也允许用户自定义动画效果，通过使用animate()方法，可以制作出效果更优雅、动作更复杂的页面动画效果。

5.4.1　简单的动画

animate()方法给开发者自定义各种复杂、高级的动画提供了极大的方便和空间，其调用的语法格式为：

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

其中，参数params表示用于制作动画效果的属性样式和值的集合。可选项[duration]表示三种默认的速度字符"slow"、"normal"、"fast"或自定义的数字。可选项[easing]为动画插件使用，用于控制动画的表现效果，通常有"linear"和"swing"字符值。可选项[callback]为动画完成后，执行的回调函数。下面举例说明。

示例5-8　简单的动画

（1）功能描述

在页面中，单击某块＜div＞元素，其自身的高度与宽度以动画的效果增大。动画完成后，元素的边框加粗，并且边框颜色及＜div＞元素内容发生变化。

（2）实现代码

新建一个HTML文件5-8.html，加入如代码清单5-8所示的代码。

代码清单 5-8　简单的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>简单的动画</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #ccc;background-color:#eee;
           width:60px;height:60px;
           font-size:13px;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".divFrame").click(function() { //div 元素点击事件
                $(this).animate({ // 宽与高变化的动画效果
                    width: "20%",
                    height: "70px"
                },
                 3000, function() { // 动画完成后执行的回调函数
                     $(this).css({ "border": "solid 3px #666" })
                     .html(" 变大了! ");
                 })
            })
        })
    </script>
</head>
<body>
     <div class="divFrame">
          点击变大
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-8所示。

图　5-8　简单的动画

（4）代码分析

在动画方法animate()中，第一个参数params在表示动画属性时，需要采用“骆驼”写法，即如果是"font-size"，必须写成"fontSize"才有效，否则报错。

5.4.2　移动位置的动画

通过animate()方法，不仅可以用动画效果增加元素的长与宽，还能以动画效果移动页面中的元素，即改变其相对位置。例如将方法中的参数params，加入如下代码：

```
$("p").animate({
        left: "20px",
        top: "70px"
        },
   3000)
```

这段代码执行后，页面中的＜p＞元素，在3000毫秒内，其对应的位置以动画的效果向右移动20像素、向下移动70像素。除上述方式移动元素位置外，还可以根据元素当前的位置进行累加或累减的移动。下面通过一个简单的示例说明其使用的方法。

示例5-9　移动位置的动画

（1）功能描述

在页面中，创建两个按钮，单击第一个“左移”按钮后，将页面中的＜div＞元素在当前的位置上，以动画的效果向左移动52像素；单

击第二个“右移”按钮后，页面中的＜div＞元素在当前的位置上，以动画的效果向右移动52像素。

（2）实现代码

新建一个HTML文件5-9.html,加入如代码清单5-9所示的代码。

代码清单 5-9　移动位置的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>移动位置的动画</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{border:solid 1px #666;
           width:168px;text-align:center;}
           .divFrame .divTitle{background-color:#eee;
           padding:5px 0px 5px 0px}
           .divFrame .divContent{width:108px;height:52px;
           padding:5px 0px 5px 0px;margin:0px 30px 0px 30px;
           overflow:hidden}
           .divFrame .divContent .divList{width:162px;
           position:relative}
           .divFrame .divContent .divList span{
           border:solid 1px #ccc;background-color:#eee;width:50px;
           height:50px;float:left;margin-right:2px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8)}
    </style>
```

```
    <script type="text/javascript">
        $(function() {
            $("input:eq(0)").click(function() { //左移按钮点击事件
                //在3000毫秒内，以动画的形式向左移动52像素
                $(".divList").animate({ left: "-=52px" }, 3000);
            })
            $("input:eq(1)").click(function() { //右移按钮点击事件
                //在3000毫秒内，以动画的形式向右移动52像素
                $(".divList").animate({ left: "+=52px" }, 3000);
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <input id="Button1" type="button"
                   value="左移" class="btn" /> 
            <input id="Button2" type="button"
                   value="右移" class="btn" />
        </div>
        <div class="divContent">
            <div class="divList">
                <span>1</span>
                <span>2</span>
                <span>3</span>
            </div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-9所示。

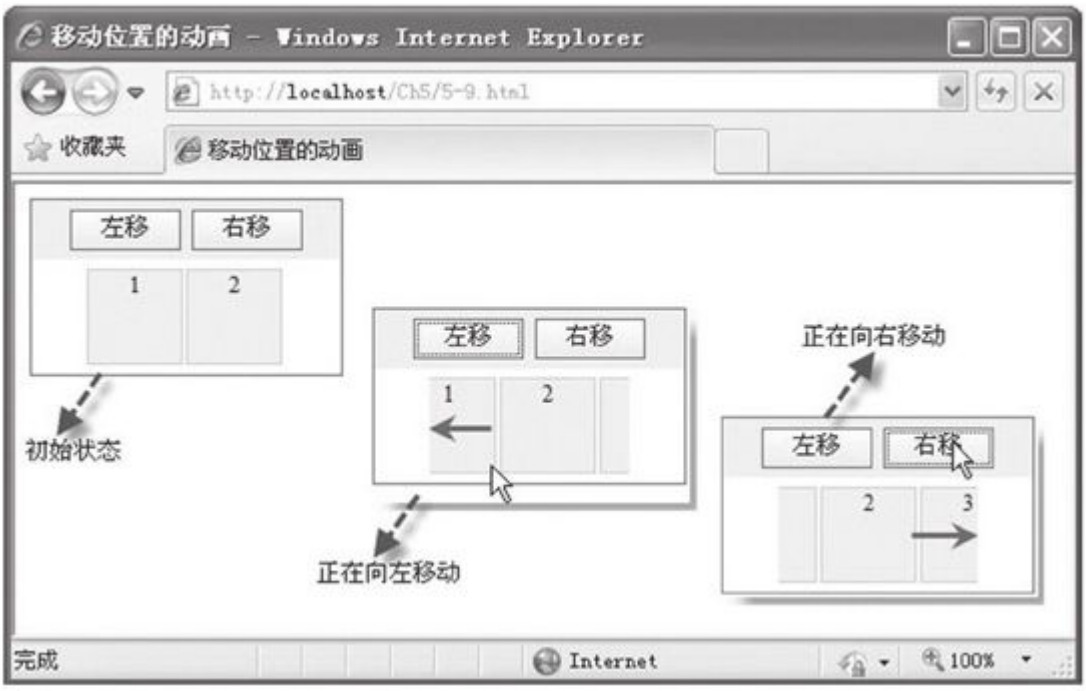


图　5-9　移动位置的动画

（4）代码分析

要使页面中的元素以动画效果移动，必须首先将该元素的"position"属性设置成为"relative"或"absolute"，否则，无法移动该元素的位置。

5.4.3　队列中的动画

所谓“队列”动画，是指在元素中执行一个以上的多个动画效果，即有多个animate()方法在元素中执行，因此，根据这些animate()方法执行的先后顺序，形成了动画“队列”，产生“队列”后，动画的效果便按“队列”的顺序进行展示。下面举例说明。

示例5-10　队列中的动画

（1）功能描述

在页面中，单击某个固定宽、高的＜div＞标记，首先，以动画的效果，将宽、高在原有基础上加大1倍；然后，再以动画的效果，将宽、高在原有基础上减小1倍。

（2）实现代码

新建一个HTML文件5-10.html,加入如代码清单5-10所示的代码。

代码清单 5-10　队列中的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>队列中的动画</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           div{border:solid 1px #666;background-color:#eee;
               width:50px;height:50px;font-size:13px;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("div").click(function() { //div 块单击事件
                $(this)
                .animate({ height: 100 }, "slow") // 第 1 列
                .animate({ width: 100 }, "slow")  // 第 2 列
                .animate({ height: 50 }, "slow")  // 第 3 列
                .animate({ width: 50 }, "slow");  // 第 4 列
            })
        })
    </script>
</head>
<body>
     <div>队列</div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-10所示。

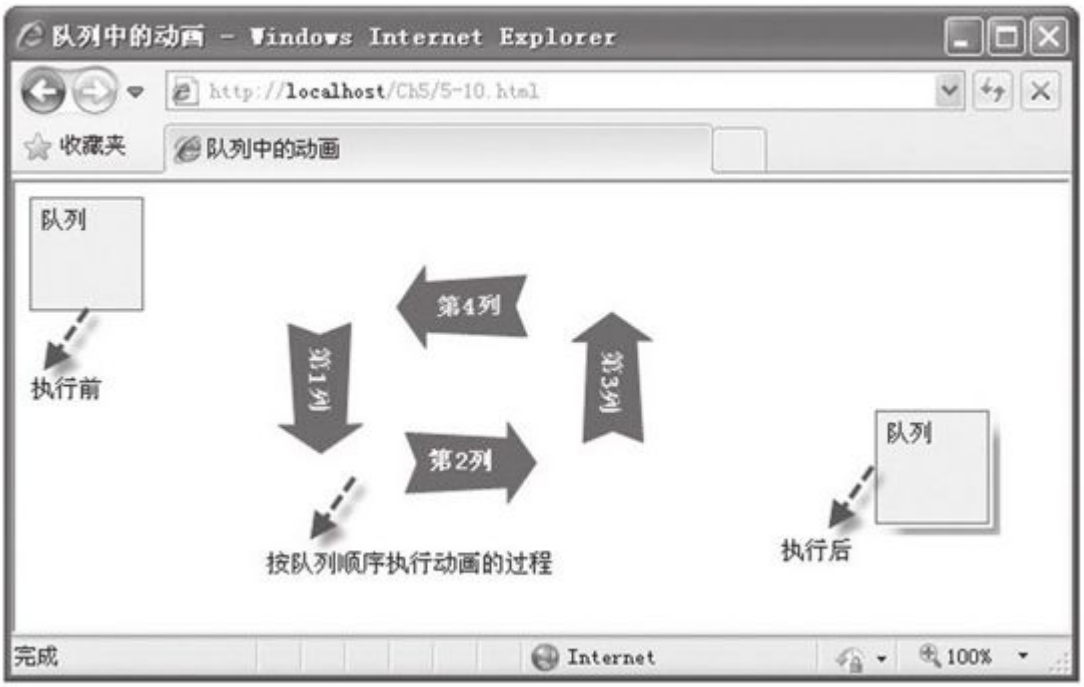


图　5-10　队列中的动画

（4）代码分析

理解动画队列执行的过程后，可以清楚分析各队列中各动画的效果，并可以在指定的某队列中插入其他方法，如队列延时方法delay()。该方法的运用将在下节中介绍。

5.4.4　动画停止和延时

在jQuery中，通过animate()可以实现元素的动画显示，但在显示的过程中，必然要考虑各种客观因素和限制性条件的存在，因此，在执行动画时，可通过stop()方法停止或delay()方法延时某个动画的执行。stop()与delay()方法的语法调用格式介绍如下。

stop()方法的格式如下：

```
stop([clearQueue], [gotoEnd])
```

该方法的功能是停止所选元素中正在执行的动画，其中可选参数[clearQueue]是一个布尔值，表示是否停止正在执行的动画，另外一个可选参数[gotoEnd]也是一个布尔值，表示是否立即完成正在执行的动画。

delay()方法的格式如下：

```
delay(duration, [queueName])
```

该方法的功能是设置一个延时值来推迟后续队列中动画的执行，其中参数duration为延时的时间值，单位是毫秒。可选参数

[queueName]表示队列名词，即动画队列。

下面举例说明。

示例5-11　动画停止和延时

（1）功能描述

在页面中设置三个超级链接，分别为“开始”、“停止”、“延时”。单击“开始”后，页面中的图片以“拉窗帘”的方式动画切换显示状态；单击“停止”后，立刻停止了正在执行中的动画效果；单击“延时”后，动画切换显示效果在延时2000毫秒后，再执行。

（2）实现代码

新建一个HTML文件5-11.html，加入如代码如清单5-11所示的代码。

代码清单 5-11　动画停止和延时

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画停止和延时</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .divFrame{border:solid 1px #666;
          width:233px;text-align:center;}
          .divFrame .divTitle{background-color:#eee;
          padding:5px 0px 5px 0px}
          .divFrame .divContent{padding:5px 0px 5px 0px}
          .divFrame .divContent img{border:solid 1px #eee;
          padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("a:eq(0)").click(function() { //点击"开始"链接事件
                $("img").slideToggle(3000); //"拉窗帘"方式切换图片
            })
            $("a:eq(1)").click(function() { //点击"停止"链接事件
                $("img").stop(); //停止正在执行的动画
            })
            $("a:eq(2)").click(function() { //点击"延时"链接事件
                $("img") //延时切换图片
                .delay(2000)
                .slideToggle(3000);
            })
        })
    </script>
</head>
<body>
     <div class="divFrame">
          <div class="divTitle">
              <a href="javascript:void(0)">开始</a>
                |  
              <a href="javascript:void(0)">停止</a>
                |  
              <a href="javascript:void(0)">延时</a>
          </div>
          <div class="divContent">
              <img src="Images/img07.gif" alt="" title="Log图片" />
          </div>


      </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图5-11所示。

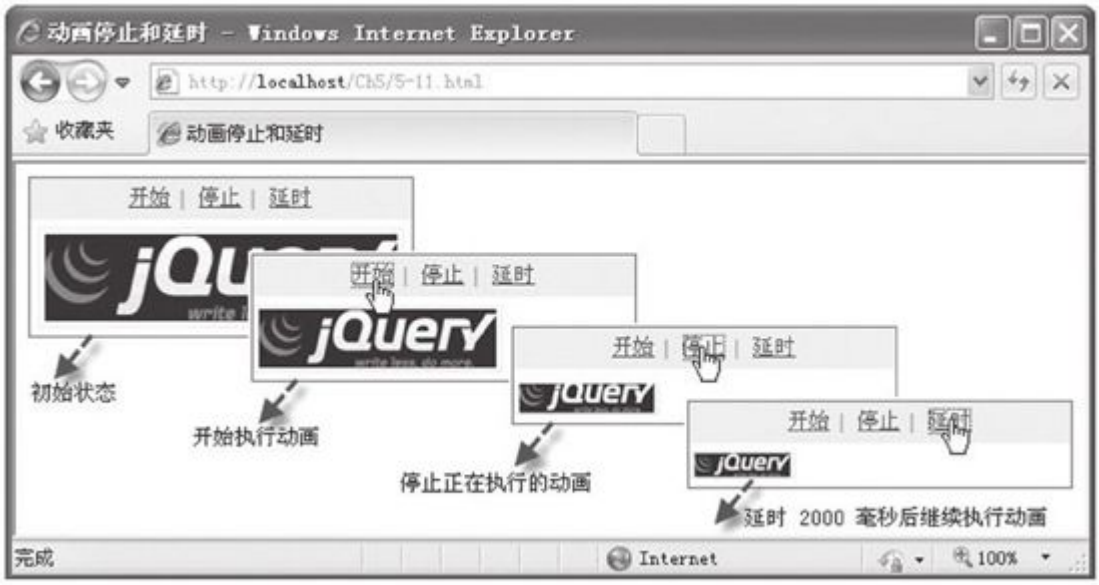


图　5-11　动画停止和延时

5.5　动画效果综述

在jQuery中，虽然有很多动画方法，实现各种各样的动画效果，但综合起来，绝大部分的动画方法仅仅是改变元素的属性或样式，如高度、宽度、透明度和CSS样式属性，详细说明如下。

5.5.1　各种动画方法说明

❑show()与hide()方法，元素以动画效果实现显示与隐藏，可以同时改变元素的多个属性，如宽度、高度、透明度。

❑fadeIn()与fadeOut()方法，元素以动画的效果淡入与淡出，仅改变元素的透明度。

❑slideUp()与slideDown()方法，元素以“卷窗帘”的动画效果显示与隐藏，仅改变元素的高度，其他属性不发生变化。

❑fadeTo()方法，元素按指定的透明度进行渐进式调整，从而达到一种动画效果。该方法改变的是元素的透明度，宽度与高度都不发生变化。

❑toggle()方法，可以代替show()与hide()两个方法，因此，其改变的元素属性也与show()与hide()方法一样。

❑slideToggle()方法，可以代替slideUp()与slideDown()两个方法，改变的元素也与slideUp()和slideDown()方法一样。

❑animate()方法，自定义元素的动画效果，可以实现上述6种方法中全部属性改变的功能，同时，还可以用动画的效果，改变其他的元素属性，该方法是上述6种方法的核心。

5.5.2　使用animate()方法代替其他动画效果

animate()方法不仅可以使元素实现各种各样的动画效果，还可以代替其他的动画方法，详细代码如下所示：

1）animate()方法代替hide()方法，代码如下：

```
$(" 页面元素 ").animate({ height: "hide", width: "hide", opacity: "hide" }, 600);
```

等价于下列代码：

```
$(" 页面元素 ").hide(600);
```

2）animate()方法代替fadeOut()方法，代码如下：

```
$(" 页面元素 ").animate({ opacity: "hide" }, 600);
```

等价于下列代码：

```
$(" 页面元素 ").fadeOut(600);
```

3）animate()方法代替slideUp()方法，代码如下：

```
$(" 页面元素 ").animate({ height: "hide" }, 600);
```

等价于下列代码：

```
$("页面元素").slideUp(600);
```

4）animate()方法代替fadeTo()方法，代码如下：

```
$("页面元素").animate({ opacity: "0.8" }, 600);
```

等价于下列代码：

```
$("页面元素").fadeTo(600, "0.8");
```

5.6　综合案例分析——动画效果浏览相册中的图片

5.6.1　需求分析

经分析，该案例的需求如下：

1）将尺寸不同的相片统一成宽度与高度相同的缩略图片，展示在页面中。

2）当单击“点击放大”链接时，以动画的效果放大至其原始图片，同时，隐藏“点击放大”链接，显示该图的基本信息。

3）当点击原始图片中的“关闭”按钮时，以动画的效果将图片缩小成单击前的缩略图，即返回到图片初始状态。

4）在浏览放大后的原始图片时，又单击其他缩略图片，那么，处于放大状态的原始图片自动以动画的效果进行关闭，使得整个相册始终只有一个图片处于放大状态。

5.6.2　界面效果

该案例的界面效果如下所示。

1）将不同尺寸的相片形成统一的缩略图片显示在页面中，即相册初始化，如图5-12所示。

图　5-12　相册初始化

2）当单击某一个图片的“点击放大”链接后，该图片以动画的效果进行放大，并展示该图片的基本信息，如图5-13所示。

图　5-13　图片放大后的效果

3）在图5-13中，如果再单击其他图片的“点击放大”链接时，将自动以动画的效果缩小当前已放大的图片，并同时放大单击过的图片，其切换状态如图5-14所示。

图　5-14　自动以动画效果缩小已放大的图片

4）单击放大后图片中的“关闭”链接时，将以动画效果缩小该图片，效果如图5-15所示。

图　5-15　单击“关闭”链接后，以动画效果缩小图片

5.6.3　功能实现

在该项目中，新建一个HTML文件listImg.html,加入如代码清单5-12所示的代码。

代码清单 5-12　相册中的动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画方式浏览图片</title>
    <link type="text/css" href="Css/css_Animate.css"
          rel="Stylesheet" />
    <script type="text/javascript"
            src="Jscript/js_Animate.js"></script>
</head>
<body>
    <div class="p_Lst">
         <img  alt=""src="images/list_1.jpg" />
         <div class="p_Alt">
              <h3>风景一</h3>
         </div>
    </div>
    <div class="p_Lst">
         <img  alt=""src="images/list_2.jpg"/>
         <div class="p_Alt">
                <h3>风景二</h3>
         </div>
     </div>
     <div class="p_Lst">
          <img  alt=""src="images/list_3.jpg"/>
          <div class="p_Alt">
               <h3>风景三</h3>
          </div>
    </div>
    <div class="p_Lst">
           <img  alt=""src="images/list_4.jpg"/>


          <div class="p_Alt">
               <h3>风景四</h3>
          </div>
    </div>
</body>
</html>
```

为了代码的调用方便和后续维护的快捷，在HTML中，单独调用了两个辅助文件，一个是用于控制页面样式的CSS文件

css_Animate.css，另外一个是用于实现页面功能的JS文件js_Animate.js。这两个文件的代码参见代码清单5-13和代码清单5-14。

代码清单 5-13　CSS 文件 css_Animate.css

```
body {font-size:13px}
/*图片外框样式*/
.p_Lst {
    position: relative;
    float: left;
    width: 90px;
    height: 98px;
    padding: 8px;
    border: 1px solid #666;
    margin: 10px 8px 20px 8px;
}
/*图片最近外框样式*/
.p_Img {
    width: 90px;
    height: 90px;
    margin-bottom: 5px;
    overflow: hidden;
}
/*图片信息样式*/
.p_Alt {
    display:none;
}
/*缩略图片中"点击放大"链接样式*/
.p_Big {
    display: block;
    width: 90px;
    height: 23px;
    background: url(../Images/imgLarge.jpg);
    cursor: pointer;
}
/*原始放大图片中"关闭"按钮样式*/
.p_Cls {
     position: absolute;
    right: 10px;
    bottom: 10px;
    display: block;
    width: 20px;
    height: 21px;
    background: url(../Images/imgClose.jpg);
    text-indent: -9999px;
}
```

代码清单 5-14 JS 文件 js_Animate.js

```
/// <reference path="jquery-1.8.2.min.js"/>

$(function() {
    var curIndex = -1; // 初始化当前打开图片值
    var intImgL = "-120px";
    var intImgT = "-120px";
    // 零参数 index 遍历图片外框 Div
    $(".p_Lst").each(function(index) {
        var $this = $(this);                    // 获取每个外框 Div
        var $img = $this.find("img");           // 查找其中的图片元素
        var $info = $this.find(".p_Alt");       // 查询其中的图片信息元素
        var arrPic = {};                        // 定义一个空数组保存初始的长与宽
        arrPic.imgw = $img.width();
        arrPic.imgh = $img.height();
        arrPic.orgw = $this.width();
        arrPic.orgh = $this.height();
        $img.css({ // 设置初始时的图片外边距位置
            marginLeft: intImgL,
            marginTop: intImgT
        });
        // 将图片、点击放大链接、关闭按钮放入外框 Div 中
        var $drag = $("<div class='p_Img'>")
                    .append($img).prependTo($this);
        var $open = $("<a href='javascript:void(0)' class='p_Big'
                    title=' 点击放大 '></a>").appendTo($this);
        var $clos = $("<a href='javascript:void(0)' class='p_Cls'
                    title=' 点击关闭 '></a>").appendTo($info);
        // 保存放入元素后的外框 Div 的长与宽
        arrPic.dragw = $drag.width();
        arrPic.dragh = $drag.height();

        // 放大按钮单击事件
        $open.click(function() {
            $this.animate({ // 外框动画
                width: arrPic.imgw,
                height: (arrPic.imgh + 85), //85 是图片信息的高度
                borderWidth: "5"
            }, 3000);
            $open.fadeOut(); // 点击放大链接淡出
            $(".p_Alt", $this).fadeIn(); // 图片提示信息淡入
            $drag.animate({ // 加入图片后的 Div 框动画
                width: arrPic.imgw,
                height: arrPic.imgh
            }, 3000);
            $img.animate({ // 以动画的形式自动调整位置
                marginTop: "0px",
                marginLeft: "0px"
            }, 3000);
            // 获取当前被放大或原始图的图片各组成部分
            var $f_this = $(".p_Lst:eq(" + curIndex + ")");
            var $f_open = $(".p_Big:eq(" + curIndex + ")");
            var $f_drag = $(".p_Img:eq(" + curIndex + ")");
            var $f_larg = $(".p_Alt:eq(" + curIndex + ")");
```

```
            var $f_imgs = $("img:eq(" + curIndex + ")");
            if (curIndex != -1) { // 如果当前有已放大的图片
                // 自动以动画的形式关闭该图片
                cls_Click($f_this, $f_open, $f_drag, $f_imgs,$f_larg);
            }
            // 重新获取当前放大图片的索引号
            curIndex = index;
        });

        //关闭按钮单击事件
        $clos.click(function() {
            // 以动画的形式缩小当前所点击的图片
            cls_Click($this, $open, $drag, $img, 1);
            // 初始化索引号
            curIndex = -1;
        });

        /*
        *@param 参数 pF 表示图片最外层 Div
        *@param 参数 pO 表示图片点击前的放大按钮
        *@param 参数 pW 表示紧邻图片层 Div
        *@param 参数 pI 表示紧连中的图片元素
        *@param 参数 blnS 表示图片中的说明内容
        */
        function cls_Click(pF, pO, pW, pI, blnS) {
            var $strInit;
            pF.animate({
                width: arrPic.orgw,
                height: arrPic.orgh,
                borderWidth: "1"
            }, 3000);
            pO.fadeIn();
            if (blnS) {
                $strInit = $(".p_Alt", pF);
            } else {
                $strInit = blnS;
            }
            $strInit.fadeOut();
            pW.animate({
                width: arrPic.dragw,
                height: arrPic.dragh
            }, 3000);
            pI.animate({
                marginTop: intImgT,
                marginLeft: intImgL
            }, 3000);
        }
    });
})
```

5.6.4　代码分析

在js_Animate.js文件中，为了实现各相册中的图片单击放大查看的动画效果，先带参数index遍历整个图片最外框的Div元素，其代码如下：

```
$(".p_Lst").each(function(index) {
// 执行代码部分
...
})
```

在遍历过程中，先获取设置好的外框与图片的长与宽，并将改变后的图片和相关元素先放入类别为"p_Img"的Div中，然后全部放入最外框的Div元素中，从而完成图片初始化的页面效果。其代码如下：

```
... 省略部分代码
var $this = $(this);                    //获取每个外框 Div
var $img = $this.find("img");           //查找其中的图片元素
var $info = $this.find(".p_Alt");       //查询其中的图片信息元素
var arrPic = {};                        //定义一个空数组保存初始的长与宽
arrPic.imgw = $img.width();
arrPic.imgh = $img.height();
arrPic.orgw = $this.width();
arrPic.orgh = $this.height();
$img.css({ //设置初始时的图片外边距位置
    marginLeft: intImgL,
    marginTop: intImgT
});
//将图片、点击放大链接、关闭按钮放入外框 Div 中
var $drag = $("<div class='p_Img'>")
            .append($img).prependTo($this);
var $open = $("<a href='javascript:void(0)' class='p_Big'
            title='点击放大'></a>").appendTo($this);
var $clos = $("<a href='javascript:void(0)' class='p_Cls'
            title='点击关闭'></a>").appendTo($info);
//保存放入元素后的外框 Div 的长与宽
arrPic.dragw = $drag.width();
arrPic.dragh = $drag.height();
... 省略部分代码
```

当单击图片初始状态中的“点击放大”链接时，首先，以3000毫秒的速度放大图片最外框Div元素，接着以同样的速度放大最近一层包含图片的Div，与此同时，以同样的速度显示或隐藏相应的页面内容，其代码如下：

```
...省略部分代码
$this.animate({ //外框动画
    width: arrPic.imgw,
    height: (arrPic.imgh + 85), //85是图片信息的高度
    borderWidth: "5"
}, 3000);
$open.fadeOut(); //点击放大链接淡出
$(".p_Alt", $this).fadeIn(); //图片提示信息淡入
$drag.animate({ //加入图片后的Div框动画
    width: arrPic.imgw,
    height: arrPic.imgh
}, 3000);
$img.animate({ //以动画的形式自动调整位置
```

```
    marginTop: "0px",
    marginLeft: "0px"
}, 3000);
...省略部分代码
```

为了在单击“点击放大”链接时，缩小正在放大的图片，使仅有一个图片处于放大显示状态中，首先，需要获取当前处于放大显示图片的索引号，然后，根据该索引号获取图片的各个组成部分，再将这些部分以动画效果进行缩小。

根据上述原理，定义一个公用变量curIndex，用于记录当前放大显示的图片索引号，并赋初始值为-1，当首次单击“点击放大”链接时，由于curIndex等于-1，因此不执行缩小的操作。这时，再通过下面代码记录当前放大的图片索引号：

```
curIndex = index;
```

当再次单击另一个“点击放大”链接时，将根据该索引号，获取图片的各个组成部分，并执行缩小的动画效果操作，其代码如下：

```
... 省略部分代码
// 获取当前被放大成原始图的图片各组成部分
var $f_this = $(".p_Lst:eq(" + curIndex + ")");
var $f_open = $(".p_Big:eq(" + curIndex + ")");
var $f_drag = $(".p_Img:eq(" + curIndex + ")");
var $f_larg = $(".p_Alt:eq(" + curIndex + ")");
var $f_imgs = $("img:eq(" + curIndex + ")");
if (curIndex != -1) { // 如果当前有已放大的图片，自动以动画的形式关闭该图片
     cls_Click($f_this, $f_open, $f_drag, $f_imgs, $f_larg);
}
... 省略部分代码
```

当在原始放大图片中，单击“关闭”按钮时，先执行缩小操作，再将变量curIndex赋值为-1，表示没有放大的原始图片，其代码如下：

```
... 省略部分代码
// 以动画的形式缩小当前所点击的图片
cls_Click($this, $open, $drag, $img, 1);
// 初始化索引号
curIndex = -1;
... 省略部分代码
```

由于图片在放大与缩小时，都执行某项同样的操作，因此，将该项操作自定义为一个函数cls_Click()。该函数的功能是根据图片的各

个组成部分，以同样的速度，采用动画的效果，缩小所选的图片，其代码如下所示：

```
... 省略部分代码
/*
*@param 参数 pF 表示图片最外层 Div
*@param 参数 pO 表示图片点击前的放大按钮
*@param 参数 pW 表示紧邻图片层 Div
*@param 参数 pI 表示紧选中的图片元素
*@param 参数 blnS 表示图片中的说明内容
*/


function cls_Click(pF, pO, pW, pI, blnS) {
    var $strInit;
    pF.animate({
        width: arrPic.orgw,
        height: arrPic.orgh,
        borderWidth: "1"
    }, 3000);
    pO.fadeIn();
    if (blnS) {
        $strInit = $(".p_Alt", pF);
    } else {
        $strInit = blnS;
    }
    $strInit.fadeOut();
    pW.animate({
        width: arrPic.dragw,
        height: arrPic.dragh
    }, 3000);
    pI.animate({
        marginTop: intImgT,
        marginLeft: intImgL
    }, 3000);
}
... 省略部分代码
```

5.7　本章小结

动画效果无疑是众多jQuery爱好者最爱之一。本章通过介绍jQuery中常用动画效果的基本语法、简单实例，使读者了解和掌握jQuery动画实现的基本方法，为开发出更优雅、高效的页面打下扎实的语言基础。

第6章　Ajax在jQuery中的应用

本章内容

加载异步数据

请求服务器数据

$.ajax()方法

Ajax中的全局事件

综合案例分析——用Ajax实现新闻点评即时更新

本章小结

Ajax是Asynchronous JavaScript and XML的缩写，其核心是通过XMLHttpRequest对象，以一种异步的方式，向服务器发送数据请求，并通过该对象接收请求返回的数据，从而完成人机交互的数据操作。这种利用Ajax技术进行的数据交互并不局限于Web动态页面，在普通的静态HTML页面中同样可以实现，因此，在这样的背景下，Ajax技术在页面开发中得以广泛使用。在jQuery中，同样有大量的函数与方法为Ajax技术提供支持。

6.1　加载异步数据

在页面开发过程中，为了加快整体页面打开的速度，对于某局部的数据采用异步读取（Ajax技术）的方法获取，这一方法的应用极大地优化了用户的体验，优化了页面的执行。

6.1.1　传统的JavaScript方法

抛开jQuery，使用传统的JavaScript方法，基于XMLHttpRequest对象，也可以将数据加载到页面中。下面通过一个简单的示例来说明这一方法实现的过程。

示例6-1　使用传统JavaScript方法实现Ajax功能

（1）功能描述

创建两个新页面a.html和b.html，前者用于加载页，在该页中，单击“加载”按钮后，在不刷新该页面的情况下，将b.html页中的内容显示在a.html页面的＜div＞标记中；后者用于被加载页，在该页中，通过＜div＞标记包含一些文字内容。

（2）实现代码

新建一个HTML文件a.html和b.html，加入如代码清单6-1a和代码清单6-1b所示的代码。

代码清单 6-1a　加载页面显示内容

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>传统的 JavaScript 方法实现 Ajax 功能 </title>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:260px;border:solid 1px #666}
           .divFrame .divTitle{padding:5px;background-color:#eee}
           .divFrame .divContent{padding:8px}
           .divFrame .divContent .clsShow{font-size:14px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        var objXmlHttp = null; // 声明一个空的 XMLHTTP 变量
        function CreateXMLHTTP() {
            // 根据浏览器的不同，返回该变量的实体对象
```

```
            if (window.ActiveXObject) {
               objXmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            else {
               if (window.XMLHttpRequest) {
                    objXmlHttp = new XMLHttpRequest();
               }
               else {
                    alert(" 初始化 XMLHTTP 错误! ");
               }
            }
        }
        function GetSendData() {
            document.getElementById("divTip").innerHTML =
            "<img alt='' title=' 正在加载中 ...'
            src='Images/Loading.gif' />"; // 初始化内容
            // 设置发送地址变量并赋初始值
            var strSendUrl = "b.html?date="+Date();
            CreateXMLHTTP(); // 实例化 XMLHttpRequest 对象
            //open 方法初始化 XMLHttpRequest
            objXmlHttp.open("GET", strSendUrl, true);
            objXmlHttp.onreadystatechange =
                function() { // 回调事件函数
                if (objXmlHttp.readyState == 4) { // 如果请求完成加载
                    if (objXmlHttp.status == 200) { // 如果响应已成功
                        // 显示获取的数据
                        document.getElementById("divTip")
                        .innerHTML = objXmlHttp.responseText;
                    }
                }
            }
            objXmlHttp.send(null); //send 发送设置的请求
        }
    </script>
</head>
<body>
  <div class="divFrame">
        <div class="divTitle">
            <input id="Button1" type="button"
            onclick="GetSendData()" class="btn" value=" 获取数据 " />
        </div>
        <div class="divContent">
            <div id="divTip"></div>
        </div>
    </div>
</body>
</html>
```

代码清单 6-1b　被加载页面用于设置内容

```
<div class="clsShow">
姓名: 陶国荣 <br />
性别: 男 <br />
邮箱: tao_guo_rong@163.com
</div>
```

（3）页面效果

代码执行后的效果如图6-1所示。

图　6-1　传统的JavaScript方法实现Ajax功能

（4）代码分析

在a.html页面的JavaScript代码中，为了能即时获取被加载页面b.html变化了的数据，在设置发送地址URL时，后面跟有一参数date，即"b.html?date="+Date()，功能就是清空缓存中已加载的变量数据信息，重新获取新的即时数据。

6.1.2　jQuery中的load()方法

在传统的JavaScript中，使用XMLHttpRequest对象异步加载数据；而在jQuery中，使用load()方法可以轻松实现获取异步数据的功能。其调用的语法格式为：

```
load(url, [data], [callback])
```

其中，参数url为被加载的页面地址，可选项[data]参数表示发送到服务器的数据，其格式为key/value。另一个可选项[callback]参数表示加载成功后，返回至加载页的回调函数。下面举例说明。

示例6-2　使用load()方法实现异步获取数据

（1）功能描述

使用load()方法实现示例6-1的功能。

（2）实现代码

新建一个HTML文件6-2.html,加入如代码清单6-2所示的代码。

代码清单 6-2　load() 方法实现异步获取数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>load() 方法实现 Ajax 功能 </title>
    <script type="text/javascript"

            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮点击事件
                $("#divTip").load("b.html"); //load() 方法加载数据
            })
        })
    </script>
</head>
<body>
   <div class="divFrame">
        ... 省略主体部分代码
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图6-2所示。

图　6-2　load()方法实现异步获取数据

（4）代码分析

在load()方法中，参数url还可以用于过滤页面中的某类别的元素，如代码"$("#divTip").load("b.html.divContent")"，则表示获取页面b.html中类别名为"ivContent"的全部元素。

6.1.3　jQuery中的全局函数getJSON()

虽然使用load()方法可以很快地加载数据到页面中，但有时需要对获取的数据进行处理，如果将用load()方法获取的内容进行遍历，也可以进行数据的处理，但这样获取的内容必须先插入页面中，然后才能进行，因此，执行的效率不高。

为了解决这个问题，我们采用将要获取的数据另存为一种轻量极的数据交互格式，即JSON文件格式，由于这种格式很方便计算机的读取，因而颇受开发者的青睐。在jQuery中，专门有一个全局函数getJSON()，用于调用JSON格式的数据，其调用的语法格式为：

```
$.getJSON(url, [data], [callback])
```

参数url表示请求的地址，可选项[data]参数表示发送到服务器的数据，其格式为key/value。另外一个可选项[callback]参数表示加载成功时执行的回调函数。下面举例说明。

示例6-3　使用全局函数getJSON()实现异步获取数据

（1）功能描述

创建一个JSON格式的文件UserInfo.json，用于保存人员资料信息，另外，新建一个页面，当单击页面中的“获取数据”按钮时，将通过全局函数getJSON()获取文件UserInfo.json中的全部数据，并展示在页面中。

（2）实现代码

新建一个HTML文件6-2.html,加入如代码清单6-3所示的代码。

代码清单 6-3　全局函数 getJSON() 实现异步获取数据

```
<!DOCTYPE html PUBLIC "~//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>getJSON 函数获取数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数处理获取的数据
                $.getJSON("UserInfo.json", function(data) {
                    $("#divTip").empty(); // 先清空标记中的内容
                    var strHTML = "";     // 初始化保存内容变量
                    // 遍历获取的数据
                    $.each(data, function(InfoIndex, Info) {
                        strHTML += "姓名：" + Info["name"] + "<br>";
                        strHTML += "性别：" + Info["sex"] + "<br>";
                        strHTML += "邮箱：" + Info["email"] + "<hr>";
                    })
                    $("#divTip").html(strHTML); // 显示处理后的数据
                })
            })
        })
    </script>
</head>
<body>
   <div class="divFrame">
        ... 省略主体部分代码


    </div>
</body>
</html>
```

另外,创建一个文本文件，另存为JSON格式,在UserInfo.json文件中加入如下代码：

```
[
  {
    "name": "陶国荣",
    "sex": "男",
    "email": "tao_guo_rong@163.com"
  },
  {
    "name": "李建洲",
    "sex": "女",
    "email": "xiaoli@163.com"
  }
]
```

（3）页面效果

代码执行后的效果如图6-3所示。

图　6-3　全局函数getJSON()实现异步获取数据

（4）代码分析

示例6-3中，在单击按钮“获取数据”时，使用全局函数$.each()遍历所获取的数据data，在遍历数据前，先清空ID号为"divTip"元素中的内容，以确保重新构建HTML内容，然后通过当前项[‘元素名称’]的方式获取每一项的数据，最后将叠加后的数据显示在ID号为"divTip"的元素中。

6.1.4　jQuery中的全局函数getScript()

在jQuery中，除通过全局函数getJSON获取JSON格式的文件内容外，还可以通过另外一个全局函数getScript()获取JS文件的内容。其实，在页面中获取JS文件的内容有很多方法，如下列代码：

```
<script type="text/javascript" src="Jscript/xx.js"></script>
```

动态设置的代码如下：

```
$("<script type='text/javascript' src='Jscript/xx.js' />")
.appendTo("head");
```

但这样的调用方法并不是最理想的。在jQuery中，通过全局函数getScript()加载JS文件后，不仅可以像加载页面片段一样轻松地注入脚本，而且所注入的脚本会自动执行，大大提高了页面的执行效率。函数getScript()的调用格式如下所示：

```
$.getScript(url, [callback])
```

参数url为等待加载的JS文件地址，可选项[callback]参数表示加载成功时执行的回调函数。

下面通过示例来解释这种方法。

示例6-4　使用全局函数getScript()实现异步获取数据

（1）功能描述

创建一个UserInfo.js文件，在该文件中以数组的方式保存人员资料信息，然后通过$.each()方法遍历各元素，将每次遍历元素的内容以叠加的方式保存至变量strHTML，并将该变量的值作为ID号为"divTip"元素的内容显示在页面中。另外，新建一个页面，在该页中单击“获取数据”按钮打开UserInfo.js文件。

（2）实现代码

新建一个HTML文件6-4.html,加入如代码清单6-4所示的代码。

代码清单 6-4　全局函数 getScript() 实现异步获取数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>getScript 函数获取数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                $.getScript("UserInfo.js");// 打开已获取返回数据的文件
            })
        })
    </script>
</head>
<body>
   ...省略主体部分代码
</body>
</html>
```

另外，新建一个JS文件UserInfo.js，该文件的代码内容如下所示：

```
var data = [
  {
      "name": "陶国荣",
      "sex": "男",
      "email": "tao_guo_rong@163.com"
  },
  {
      "name": "李建洲",
      "sex": "女",
      "email": "xiaoli@163.com"
  }
];

var strHTML = ""; //初始化保存内容变量
$.each(data, function() { //遍历获取的数据
    strHTML += "姓名: " + this["name"] + "<br>";
    strHTML += "性别: " + this["sex"] + "<br>";
    strHTML += "邮箱: " + this["email"] + "<hr>";
})
$("#divTip").html(strHTML); //显示处理后的数据
```

（3）页面效果

代码执行后的效果如图6-4所示。

图　6-4　全局函数getScript()实现异步获取数据

（4）代码分析

与所有全局函数一样，getScript()也有一个回调函数，该回调函数在文件加载成功后执行，如代码“$.getScript('UserInfo.js',function(){alert("加载成功！");});”表示在文件加载成功后，将弹出一个内容是“加载成功！”的窗口。

6.1.5　jQuery中异步加载XML文档

在前几节中，我们通过jQuery中的各种全局函数，实现了不同格式数据的异步加载，如HTML、JSON、JS格式数据。在日常的页面开发中，有时也会遇到使用XML文档保存数据的情况，对于这种格式的数据，jQuery中使用全局函数$.get()进行访问，其调用的语法格式为：

```
$.get(url, [data], [callback], [type])
```

其中，参数url表示等待加载的数据地址，可选项[data]参数表示发送到服务器的数据，其格式为key/value，可选项[callback]参数表示加载成功时执行的回调函数，可选项[type]参数表示返回数据的格式，如HTML、XML、JS、JSON、TEXT等。下面举例说明。

示例6-5　使用全局函数get()实现异步获取XML文档数据

（1）功能描述

将人员资料信息保存到一个XML格式的文档UserInfo.xml中，另外新建一个页面，在页面中单击“获取数据”按钮，通过全局函数$.get()打开XML格式的文档UserInfo.xml，并获取文档中的全部数据，显示在页面中。

（2）实现代码

新建一个HTML文件6-5.html,加入如代码清单6-5所示的代码。

代码清单 6-5　全局函数 get() 实现异步获取 XML 文档数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>get 实现异步获取 XML 文档数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数处理获取的数据
                $.get("UserInfo.xml", function(data) {
                    $("#divTip").empty();     // 先清空标记中的内容
                    var strHTML = "";         // 初始化保存内容变量
                    $(data).find("User")
                        .each(function() {    // 遍历获取的数据
                        var $strUser = $(this);
                        strHTML += "姓名：" +
                        $strUser.find("name").text() + "<br>";
                        strHTML += "性别：" +
                        $strUser.find("sex").text() + "<br>";
                        strHTML += "邮箱：" +
                        $strUser.find("email").text() + "<hr>";
                    })
                    $("#divTip").html(strHTML); // 显示处理后的数据
                })
            })
        })
    </script>

</head>
<body>
   ... 省略主体部分代码
</body>
</html>
```

另外，新建一个XML文件UserInfo.xml，该文件的代码内容如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<Info>
  <User id="1">
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>

  <User id="2">
    <name> 李建洲 </name>
    <sex> 女 </sex>
    <email>xiaoli@163.com</email>
  </User>
</Info>
```

（3）页面效果

代码执行后的效果如图6-5所示。

图　6-5　全局函数get()实现异步获取XML文档数据

（4）代码分析

在示例的JS代码中，先通过each()方法遍历文档中的User节点，然后在遍历的过程中使用find方法，查询该节点中的name、sex、email选项，并通过text()方法获取各选项的值，最后，将各值以叠加的形式保存到变量strHTML，并显示在页面中。

6.2　请求服务器数据

前面6.1节介绍的是如何在HTML页面中加载异步数据的方法，即如何从服务器上取得静态的数据。但页面的应用远不仅局限于此，Ajax技术最终体现在与服务器的动态数据实现人机交互中，即客户端传递带有参数的请求，服务器接收后，分析所传递来的请求，并做出相应的响应，发送对应数据至客户端，客户端接收请求返回的数据，从而实现了数据的双向传递。下面就介绍交互式函数的应用。

6.2.1　$.get()请求数据

在6.1.5小节中，通过调用全局函数$.get()实现了XML文档的加载。除加载数据外，$.get()函数还可以实现数据的请求。下面通过一个示例介绍$.get()函数带参请求服务器中的数据。

示例6-6　使用全局函数get()向服务器请求数据

（1）功能描述

创建一个服务器页面UserInfo.aspx，该页面的功能是获取客户端发送的请求，并分析传来的参数值，返回对应的结果。另外，在客户端HMTL页面的文本框中，输入传递的参数值，单击“请求数据”按钮后，显示服务器返回的数据结果。

（2）实现代码

新建一个HTML文件6-6.html,加入如代码清单6-6所示的代码。

代码清单 6-6　全局函数 get() 向服务器请求数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.get 发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                // 打开文件，并通过回调函数返回服务器响应后的数据
                $.get("UserInfo.aspx",
                { name: encodeURI($("#txtName").val()) },
                function(data) {
                    $("#divTip")
                    .empty()      // 先清空标记中的内容
                    .html(data); // 显示服务器返回的数据
                })
            })
        })
    </script>
</head>
<body>
    ... 省略主体部分代码
</body>
</html>
```

另外，新建一个服务器端文件UserInfo.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility
                     .UrlDecode(Request["name"]);// 解码姓名字符
    string strHTML = "<div class='clsShow'>";     // 初始化保存内容变量
    if (strName == "陶国荣")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        strHTML += "邮箱：tao_guo_rong@163.com<hr>";
    }
    else if (strName == "李建洲")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        strHTML += "邮箱：xiaoli@163.com<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

代码执行后的效果如图6-6所示。

图　6-6　全局函数get()向服务器请求数据

（4）代码分析

客户端向服务器传递参数时，使用的格式是{key0:value0,key1:value1,...}，"key0"为参数名称，"value0"为参数的值。如果参数的值是中文格式，必须通过使用encodeURI()进行转码，当然，在客户端接收时，使用decodeURI()进行解码即可。

6.2.2　$.post()请求数据

$.post()也是带参数向服务器发出数据请求。全局函数$.post()与$.get()在本质上没有太大的区别，所不同的是，GET方式不适合传递数据量较大的数据，同时，其请求的历史信息会保存在浏览器的缓存中，有一定的安全风险。而以POST方式向服务器发送数据请求，则不存在这两个方面的不足。

$.post()函数调用的语法格式如下所示：

```
$.post(url, [data], [callback], [type])
```

其中，参数与$.get()函数参数代表的意义完全相同，在此不赘述。下面举例说明。

示例6-7　使用全局函数post()向服务器请求数据

（1）功能描述

在示例6-6的基础上，在客户端页面中增加一个下拉列表框，用于“性别”的选择，单击“请求”按钮后，传递两个参数name与sex的值，向服务器页面User_Info.aspx发出数据请求，服务器页面接收参数值后，响应请求，将相应数据发送到客户端。

（2）实现代码

新建一个HTML文件6-7.html,加入如代码清单6-7所示的代码。

代码清单 6-7　全局函数 post() 向服务器请求数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.post 发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                //打开文件，并通过回调函数返回服务器响应后的数据
                $.post("User_Info.aspx",
                { name: encodeURI($("#txtName").val()),
                  sex: encodeURI($("#selSex").val())
                },
                function(data) {
                    $("#divTip")
                    .empty()      //先清空标记中的内容
                    .html(data); //显示服务器返回的数据
                })
            })
        })
    </script>
</head>
<body>
   ... 省略主体部分代码
</body>
</html>
```

另外，新建一个服务器端文件User_Info.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility
                     .UrlDecode(Request["name"]);// 解码姓名字符
    string strSex = System.Web.HttpUtility
                    .UrlDecode(Request["sex"]);  // 解码性别字符
    string strHTML = "<div class='clsShow'>";    // 初始化保存内容变量
    if (strName == "陶国荣" && strSex=="男")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        strHTML += "邮箱：tao_guo_rong@163.com<hr>";
    }
    else if (strName == "李建洲" && strSex == "女")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        strHTML += "邮箱：xiaoli@163.com<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

执行后的效果如图6-7所示。

图　6-7　全局函数post向服务器请求数据

6.2.3　serialize()序列化表单

在使用全局函数$.get()和$.post()向服务器传递参数时，其中的参数是通过名称属性逐个搜索输入字段的方式进行传输的，如果表单的输入字段过多，这种方式就比较麻烦，而且不利于拓展。为了解决这个问题，jQuery引入serialize()方法，该方法可以简化参数传值的方式，其调用的语法格式如下：

```
serialize()
```

该方法的功能是将所选择的DOM元素转换成能随Ajax传递的字符串，即序列化所选择的DOM元素。下面举例说明。

示例6-8　使用serialize()序列化表单

（1）功能描述

在示例6-7的基础上，在客户端页面中，再增加一个“邮箱”复选框，用于控制页面是否显示邮箱地址，当单击“请求”按钮时，通过调用表单的serialize()方法，获取全部的输入字段值，并向服务器页面User-Info.aspx发出数据请求，服务器页面接收参数值后，响应请求，将相应数据发送到客户端。

（2）实现代码

新建一个HTML文件6-8.html,加入如代码清单6-8所示的代码。

代码清单 6-8　serialize() 序列化表单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>serialize() 序列化表单 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数返回服务器响应后的数据
                $.post("User-Info.aspx",
                $("#frmUserInfo").serialize(),// 序列化表单数据
                function(data) {
                    $("#divTip")
                    .empty()      // 先清空标记中的内容
                    .html(data); // 显示服务器返回的数据
                })
            })
        })
    </script>
</head>
<body>
  <form id="frmUserInfo">
     ... 省略主体部分代码
  </form>
</body>
</html>
```

另外，新建一个服务器端文件User-Info.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%
```

```
    string strName = System.Web.HttpUtility
                     .UrlDecode(Request["txtName"]); //解码姓名字符
    string strSex = System.Web.HttpUtility
                    .UrlDecode(Request["selSex"]);   //解码性别字符
    string strEmail = Request["chkEmail"];           //是否显示邮件字符
    string strHTML = "<div class='clsShow'>";        //初始化保存内容变量
    if (strName == "陶国荣" && strSex=="男")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        if (strEmail=="1")
        {
            strHTML += "邮箱：tao_guo_rong@163.com";
        }
        strHTML += "<hr>";
    }
    else if (strName == "李建洲" && strSex == "女")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        if (strEmail == "1")
        {
            strHTML += "邮箱：xiaoli@163.com<hr>";
        }
        strHTML += "<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

代码执行后的效果如图6-8所示。

图　6-8　serialize()序列化表单

（4）代码分析

虽然serialize()方法可以很完美地模拟浏览器提交表单的操作，同时自动解决了中文编码的问题，但它自身有很多不足，如表单中有多项被选中时，该方法只能传递一项的值，因此，在选择传递参数时，须慎重考虑。

6.3 $.ajax()方法

除了可以使用全局性函数load()、get()、post()实现页面的异步调用和与服务器交互数据外，在jQuery中，还有一个功能更为强悍的最底层的方法$.ajax()，该方法不仅可以方便地实现上述三个全局函数完成的功能，更多地关注实现过程中的细节。下面我们详细介绍该方法。

6.3.1 $.ajax()中的参数及使用方法

在jQuery中，$.ajax()是最底层的方法，也是功能最强的方法，前几节中的$.get()、$.post()、$.getScript()、$.getJSON()都是在此方法的基础之上建立的。其调用的语法格式为：

```
$.ajax([options])
```

其中，可选项参数[options]为$.ajax()方法中的请求设置，其格式为key/value，既包含发送请求的参数，也含有服务器响应后回调的数据，其全部名称如表6-1所示。

表 6-1　$.ajax() 方法中的参数列表

参数名	类型	功能描述
url	String	发送请求的地址（默认为当前页面）
type	String	数据请求方式（post 或 get），默认为 get
data	String 或 Object	发送到服务器的数据。如果不是字符串则自动转成字符串格式，如果是 get 请求方式，那么，该字符串将附在 url 的后面
dataType	String	服务器返回的数据类型。如果没有指定，jQuery 将自动根据 HTTP 包 MIME 信息自动判断，服务器返回的数据根据自动判断的结果进行解析，传递给回调函数，其可用类型为： html：返回纯文本的 HTML 信息，包含的 Script 标记会在插入页面时被执行 script：返回纯文本 JavaScript 代码 text：返回纯文本字符串 xml：返回可被 jQuery 处理的 XML 文档 json：返回 JSON 格式的数据
beforeSend	Function	该函数用于发送请求前修改 XMLHttpRequest 对象，其中的参数就是 XMLHttpRequest 对象，由于该函数本身是 jQuery 事件，因此，如果函数返回 false，则表示取消本次事件
complete	Function	请求完成后调用的回调函数，该函数无论数据发送成功或失败都会调用，其中有两个参数，一个是 XMLHttpRequest 对象，另外一个是 strStatus，用于描述成功请求类型的字符串
success	Function	请求成功后调用的回调函数，该函数有两个参数，一个是根据参数 dataType 处理后服务器返回的数据，另外一个是 strStatus，用于描述状态的字符串
error	Function	请求失败后调用的回调函数，该函数有三个参数，第一个是 XMLHttpRequest 对象，第二个是出错信息 strError，第三个是捕捉到的错误对象 strObject
timeout	Number	请求超时的时间（毫秒），该设置将覆盖 $.ajaxSetup() 方法中的同样设置
global	Boolean	是否响应全局事件，默认是 true，表示响应，如果设置成 false，表示不响应，那么，全局事件 $.ajaxStart 等将不响应
async	Boolean	是否为异步请求，默认是 true，表示异步，如果设置成 false，表示同步请求
cache	Boolean	是否进行页面缓存，true 表示进行缓存，false 表示不进行缓存

6.3.2　$.ajax()在数据交互中的应用

下面通过一个简单示例介绍jQuery中$.ajax()方法在数据交互过程中的应用。

示例6-9　使用$.ajax()方法发送请求

（1）功能描述

创建一个用于登录的HTML页login.html，在页面中设置用于输入“用户名”和“密码”的文本框及“登录”和“取消”按钮，该页面不做任何数据处理。

另外，创建一个服务器端页面login.aspx，用于处理静态页发送来的登录请求；同时，再创建一个HTML页6-9.html。在页面6-9.html中，先使用$.ajax()方法请求调用login.html页面，然后通过页面中“登录”按钮的单击事件，将获取的用户名称和密码发送到服务器，服务器响应请求，向客户端发送处理后的数据，客户端根据回调的数据显示不同的页面效果。

（2）实现代码

新建一个HTML文件6-9.html,加入如代码清单6-9所示的代码。

代码清单 6-9　$.ajax() 方法发送请求

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.ajax() 方法发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $.ajax({ //请求登录页
                url: "login.html", //登录静态页
                dataType: "html",
                success: function(HTML) { //返回页面内容
                    //将页面内容置入表单
                    $("#frmUserLogin").html(HTML);
                    //" 登录 " 按钮单击事件
                    $("#btnLogin").click(function() {
                        //获取用户名称
                        var strTxtName = encodeURI($("#txtName").val());
                        //获取输入密码
                        var strTxtPass = encodeURI($("#txtPass").val());
                        //开始发送数据
                        $.ajax({ //请求登录处理页
                            url: "login.aspx", //登录处理页
                            dataType: "html",
                            //传送请求数据
                            data: { txtName: strTxtName,
                                    txtPass: strTxtPass },


                            //登录成功后返回的数据
                            success: function(strValue) {
                                //根据返回值进行状态显示
                                if (strValue == "True") {
                                    $(".clsShow")
                                    .html(" 操作提示，登录成功！ ");
                                }
                                else {
                                    $("#divError")
                                    .show().html(" 用户名或密码错误！ ");
                                }
                            }
                        })
                    })
                }
            })
        })
    </script>
</head>
<body>
  <form id="frmUserLogin"></form>
</body>
</html>
```

另外，再新建一个HTML页面Login.html，该文件的代码内容如下所示：

```
<div class="divFrame">
     <div class="divTitle">
          <span>用户登录</span>
     </div>
     <div class="divContent">
          <div class="clsShow">
               <div id="divError" class="clsError"></div>
               <div>名称: <input id="txtName" type="text"
                                 class="txt" /></div>
               <div>密码: <input id="txtPass" type="password"
                                 class="txt" /></div>
               <div>
                   <input id="btnLogin" type="button"
                          value="登录" class="btn" />  
                   <input id="btnReset"  type="reset"
                          value="取消" class="btn" />
               </div>
           </div>
      </div>
</div>
```

新建一个服务器端文件Login.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility

                      .UrlDecode(Request["txtName"]); //解码姓名字符
     string strPass = System.Web.HttpUtility
                     .UrlDecode(Request["txtPass"]); //解码密码字符
     bool blnLogin = false;
     if (strName == "admin" && strPass == "123456")
     {
         blnLogin = true;
     }
     Response.Write(blnLogin);
 %>
```

（3）页面效果

代码执行后的效果如图6-9所示。

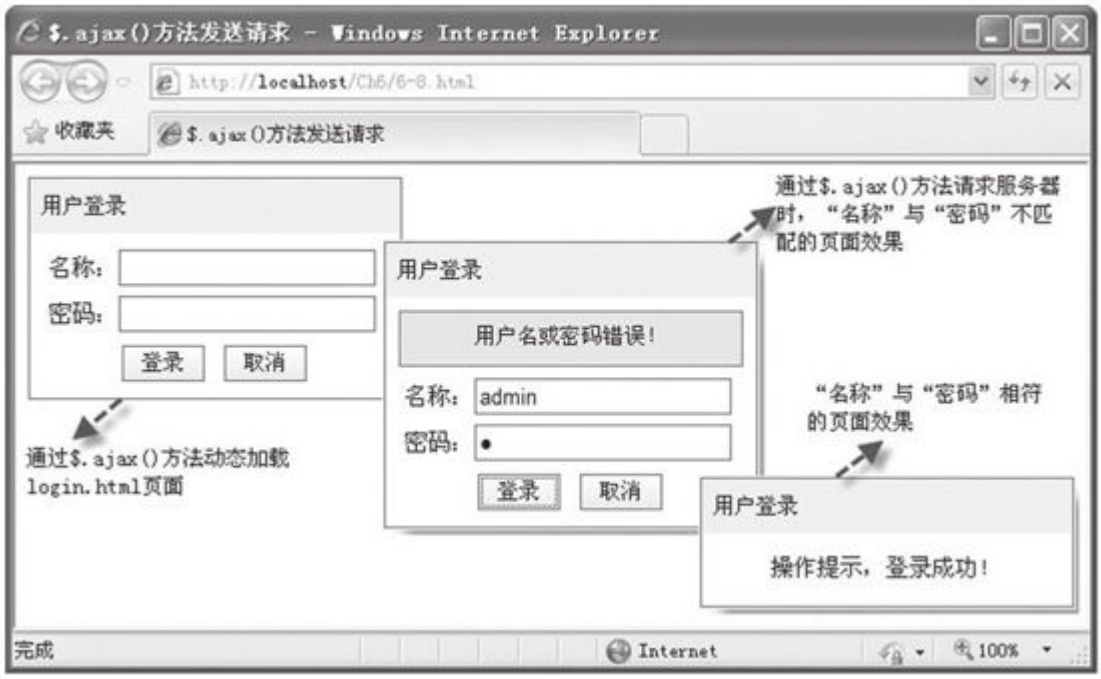


图　6-9　$.ajax()方法发送请求

（4）代码分析

由于$.ajax()方法是jQuery中最底层的方法，因此，凡使用全局函数$.getScript()、$.get()、$.post()、$.getJSON()调用的示例，都可以使用$.ajax()方法进行代替，限于篇幅，这里不赘述，读者可自行测试。

6.3.3　$.ajaxSetup()设置全局Ajax

在使用$.ajax()方法时，有时需要调用多个$.ajax()方法，如果每个方法都设置其中的请求细节，将是一件十分麻烦的事。为了简化这种工作，在jQuery中，可以使用$.ajaxSetup()方法设置全局性的Ajax默认选项，一次设置，全局有效，这样大大简化了$.ajax()方法中细节的编写，该方法的调用格式为：

```
$.ajaxSetup([options])
```

其中，可选项参数[options]是一个对象，通过该对象可以设置$.ajax()方法中的参数。下面举例说明。

示例6-10　使用$.ajaxSetup()方法全局设置Ajax

（1）功能描述

在页面中，设置三个按钮，分别通过$.ajax()方法请求一个XML文档中的某部分数据，并将回调的数据展示在页面中。在请求前，使用$.ajaxSetup()方法进行一些参数项的全局性设置。

（2）实现代码

新建一个HTML文件6-10.html,加入如代码清单6-10所示的代码。

代码清单 6-10　$.ajaxSetup() 方法全局设置 Ajax

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.ajaxSetup() 方法全局设置 Ajax</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $.ajaxSetup({ // 设置全局性的 Ajax 选项
                type: "GET",
                url: "UserInfo.xml",
                dataType: "xml"
            })
            $("#Button1").click(function() {          //" 姓名 " 按钮的单击事件
                $.ajax({
                    success: function(data) {         // 传回请求响应的数据
                     ShowData(data, " 姓名 ", "name"); // 显示 " 姓名 " 部分
                    }
                })
            })
            $("#Button2").click(function() {          //" 性别 " 按钮的单击事件
                $.ajax({
                    success: function(data) {         // 传回请求响应的数据
                     ShowData(data, " 性别 ", "sex");  // 显示 " 性别 " 部分
                    }
                })
            })
            $("#Button3").click(function() {          //" 邮箱 " 按钮的单击事件
                $.ajax({
                    success: function(data) {         // 传回请求响应的数据
                     ShowData(data, " 邮箱 ", "email"); // 显示 " 邮箱 " 部分
                    }
                })
            })
            /*
            * 根据名称与值，获取请求响应数据中的某部分
            *@Param d 为请求响应后的数据
```

```
            *@Param n 为数据中文说明字符
            *@Param d 为数据在响应数据中的元素名称
            */
            function ShowData(d, n, v) {
                $("#divTip").empty();                    //先清空标记中的内容
                var strHTML = "";                        //初始化保存内容变量
                $(d).find("User").each(function() { //遍历获取的数据
                    var $strUser = $(this);
                    strHTML += n + ": " + $strUser.find(v).text() + "<hr>";
                })
                $("#divTip").html(strHTML);              //显示处理后的数据
            }
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <span><input id="Button1" type="button"
                         value="姓名" class="btn" /></span>
            <span><input id="Button2" type="button"
                         value="性别" class="btn" /></span>
            <span><input id="Button3" type="button"
                         value="邮箱" class="btn" /></span>
        </div>
        <div class="divContent">
            <div id="divTip" class="clsShow"></div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图6-10所示。

图　6-10　$.ajaxSetup()方法全局设置Ajax

（4）代码分析

在JS代码中，由于使用了$.ajaxSetup()方法设置部分全局性的Ajax参数选项，使后续代码中的异步数据请求非常简单，避免了重复编写相同的代码。由于每次请求都要分析响应后的数据，因此通过一个自定义的函数ShowData，在每次调用时，根据不同的数据元素名称，返回对应的数据。

6.4　Ajax中的全局事件

在jQuery中，除了全局性的函数外，还有6个全局性的Ajax事件。由于在使用$.ajax()方法时，其中的选项参数global的值默认情况下为true，这也就意味着，所有在执行的Ajax的数据请求，都绑定了全局事件。

6.4.1　Ajax全局事件的参数及功能

jQuery中的6个全局性Ajax事件的详细说明见表6-2。

表 6-2　Ajax 中的全局事件

事件名称	参数	功能描述
ajaxComplete(callback)	callback	Ajax 请求完成时执行函数
ajaxError(callback)	callback	Ajax 请求发生错误时执行函数，其中捕捉到的错误可以作为最后一个参数进行传递
ajaxSend(callback)	callback	Ajax 请求发送前执行函数
ajaxStart(callback)	callback	Ajax 请求开始时执行函数
ajaxStop(callback)	callback	Ajax 请求结束时执行函数
ajaxSuccess(callback)	callback	Ajax 请求成功时执行函数

说明　在jQuery中，所有的全局事件都是以XMLHttpRequest对象和设置作为参数传递给回调函数，在处理回调函数时，只要分析其传回的参数值即可。

6.4.2　ajaxStart与ajaxStop全局事件

在jQuery中，使用Ajax获取异步数据时，ajaxStart与ajaxStop这两个全局事件的使用频率非常高。前者是当请求开始执行时触发，往往用于编写一些准备性的工作，如提示“正在获取数据……”字样；后者是当请求结束时触发，在这一事件中，常常与前者配合，说明请求的最后进展状态，如将显示中的“正在获取数据……”字样修改为“已成功获取数据!”，然后渐渐消失。

在事件绑定时，Ajax中的全局事件可以绑定在页面的任何一个元素中，如下列代码将全局事件ajaxStart绑定在一个＜div＞标记中：

```
$("divTip").ajaxStart(function(){

  $(this).html(" 正在处理中 ...");
})
```

通过编写上述代码，使页面中的任何Ajax请求在开始执行时都会触发，即显示“正在处理中……”的字样。下面举例说明。

示例6-11　jQuery中的全局事件

（1）功能描述

创建一个HTML页面，设置一个文本框和一个按钮，单击“请求”按钮后，获取文本框数据，向服务器发送请求，服务器响应请求后，返回对应的数据，在数据交互过程中，请求开始时，触发绑定的ajaxStart全局事件，显示“正在发送数据请求……”的字样；请求结束后，触发绑定的ajaxStop全局事件，将“正在发送数据请求……”改成“已成功获取数据”，并隐藏起来。

（2）实现代码

新建一个HTML文件6-11.html,加入如代码清单6-11所示的代码。

代码清单 6-11　jQuery 中的全局事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Ajax 中的全局事件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            // 元素绑定全局 ajaxStart 事件
            $("#divMsg").ajaxStart(function() {
                $(this).show(); // 显示元素
            })
            // 元素绑定全局 ajaxStop 事件
            $("#divMsg").ajaxStop(function() {
                $(this).html(" 已成功获取数据。").hide();
            })
            // 按钮的单击事件
            $("#Button1").click(function() {
                $.ajax({ // 带参数请求服务器
                    type: "GET",
                    url: "GetData.aspx",
                    // 获取加码后的数据并作为参数传给服务器
                    data: { txtData:
                            encodeURI($("#txtData").val()) },
                    dataType: "html",
                    success: function(data) { // 显示解码后的返回数据
```

```
                    $("#divTip").html(decodeURI(data));
                }
            })
        })
    })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <span>数据: <input id="txtData" type="text"
                              class="txt" /></span>
            <span><input id="Button1" type="button"
                              value="发送" class="btn" /></span>
        </div>
        <div class="divContent">
            <div id="divMsg" class="clsTip">正在发送数据请求 ...</div>
            <div id="divTip" class="clsShow"></div>
        </div>
    </div>
</body>
</html>
```

另外，新建一个服务器端文件GetData.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"ResponseEncoding="gb2312" %>
<%
    string strName = Request["txtData"]; //返回传回的参数
    Response.Write(strName);             //返回传回的参数
%>
```

（3）页面效果

代码执行后的效果如图6-11所示。

图　6-11　jQuery中的全局事件

（4）代码分析

由于是全局性的事件，因此只要有Ajax请求发生，就会触发所设置的全局事件。该事件被绑定在某个元素上，通过一些很人性化的显示或隐藏进行切换，极大地丰富了提交数据时的用户体验，同时建立了页面加载反馈系统。

6.5　综合案例分析——使用Ajax实现新闻点评即时更新

6.5.1　需求分析

经分析，该案例的需求如下：

1）使用XML文档保存新闻点评数据，在初始化页面和发表点评内容时，使用无刷新的方式调用该数据。

2）使用无刷新的方式，获取页面中提交的数据，通过服务器文件，写入保存新闻点评数据的XML文档中。

3）无论是获取数据，还是请求服务器响应，在页面中都有人性化的提示信息侦察进程的状态，即操作时显示，操作完成后则隐藏。

6.5.2　界面效果

该案例的界面效果如下。

1）初次加载点评数据时，在获取数据前出现提示进程状态的信息，如图6-12所示。

图　6-12　初始化新闻点评数据

2）输入点评内容与用户名称后，单击“发表”按钮，同样出现提示状态信息，并根据请求的进度切换状态信息显示的内容和效果，其页面效果如图6-13所示。

图　6-13　展示获取的点评数据

3）当服务器响应请求后，将传来的内容与用户名参数解码后写入XML中，同时，返回请求成功的信息，客户端根据这一信息，重新加载点评数据，将新增加的数据显示在页面中，其页面效果如图6-14所示。

图　6-14　展示最新写入的数据

6.5.3 功能实现

新建一个HTML文件CommentData.html，加入如代码清单6-12所示的代码。

代码清单 6-12 实现新闻点评即时更新

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>新闻点评功能</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link type="text/css" href="Css/css_Ajax.css" rel="Stylesheet" />
    <script type="text/javascript" src="Jscript/js_Ajax.js"></script>
</head>
<body>
     <div class="divFrame">
         <div class="divTitle">
              <span>最新点评</span>
         </div>
         <div class="divContent">
         </div>
         <div class="divSubmit">
             <div class="SubTitle">发表评论
                  <span id="divMsg" class="clsTip">
                        正在发送数据请求...
                  </span>
             </div>
             <div class="SubContent">
                 <textarea id="txtContent" rows="6"


                           class="txt"></textarea>
                 <div class="SubBtn">
                     <span style="float:left">用户名:
                           <input id="txtName" type="text"
                                  class="txt" />
                     </span>
                     <span style="float:right">
                           <input id="Button1" type="button"
                                  value="发表" class="btn" />
                     </span>
                 </div>
             </div>
         </div>
    </div>
</body>
</html>
```

另外，为了便于模块化调用，针对该点评页面新建一个控制样式的css_Ajax.css文件，其代码如代码清单6-13所示。

代码清单 6-13　css_Ajax.css 文件

```
body{font-size:13px}
a
{
    text-decoration:none
}
/*外框样式*/
.divFrame
{
    width:572px;
    border:solid 1px #666;
    background-color:#fafcff
}
/*外框中主题样式*/
.divFrame .divTitle
{
    padding:5px;
    background-color:#eee
}
.divFrame .divTitle span
{
    padding:2px;
    padding-top:5px;
    font-family:黑体;
    font-size:14px
}
/*外框中内容样式*/
.divFrame .divContent
{
    padding:8px
}
...省略样式部分代码
.divFrame .divContent .clsShow .ShowBottom
{
```

```
    text-align:right;
    font-size:12px;
    color:#555
}
/* 外框中提交点评内容样式 */
.divFrame .divSubmit
{
    padding:20px
}
... 省略样式部分代码
.divFrame .divSubmit .SubContent .SubBtn
{
    padding-top:10px;
    padding-bottom:12px;
    font-size:12px;
    color:#555;
    font-weight:bold
}
/* 侦察请求状态样式 */
.clsTip
{
    position:absolute;
     width:160px;
     text-align:center;
     font-size:13px;
     border:solid 1px #cc3300;
     padding:2px;
     margin-bottom:5px;
     background-color:#ffe0a3
 }
 ... 省略样式部分代码
```

用于实现页面功能的JS文件js_Ajax.js，其代码如代码清单6-14所示。

代码清单 6-14　js_Ajax.js 文件

```
/// <reference path="jquery-1.8.2.min.js"/>
$(function() {
    // 元素绑定全局 ajaxStart 事件
    $("#divMsg").ajaxStart(function() {
        $(this).show(); // 显示元素
    })
    // 元素绑定全局 ajaxStop 事件
    $("#divMsg").ajaxStop(function() {
        $(this).html(" 数据处理已完成。").hide();
    })
    // 初始化点评数据
    LoadData();
    $("#Button1").click(function() { // 点击 " 发表 " 按钮事件
        // 获取加码后的用户名
        var strName = encodeURI($("#txtName").val());
        // 获取加码后的发表内容
        var strContent = encodeURI($("#txtContent").val());
        $.ajax(
            {
```

```
                type: "GET",
                url: "AddData.aspx", //请求增加数据动态页
                dataType: "html",
                data: { name: strName, content: strContent },
                success: function(msg) {
                    alert(msg);
                    LoadData();
                    $("#txtName").val("");
                    $("#txtContent").val("");
                }
            })
        })
        /*
        *动态加载XML格式的点评数据
        */
        function LoadData() {
            $.ajax(
                {
                    type: "GET",
                    url: "CommentData.xml", //请求XML格式数据
                    dataType: "xml",
                    cache: false,
                    success: function(data) {
                        $(".divContent").empty(); //先清空标记中的内容
                        var strHTML = "";         //初始化保存内容变量
                        //如果没有找到数据
                        if ($(data).find("Data").length == 0) {
                            strHTML = "<div style='text-align:center'>
                                       目前还没有找到点评数据! </div>";
                        }
                        //遍历获取的数据
                        $(data).find("Data").each(function() {
                            var $strUser = $(this);
                            strHTML += "<div class='clsShow'>";
                            strHTML += "<div class='ShowTitle'>荣拓网友
                              <a href=''>" +
                            $strUser.find("name").text() + "</a></div>";
                            strHTML += "<div class='ShowContent'>" +
                            $strUser.find("content").text() + "</div>";
                            strHTML += "<div class='ShowBottom'>发送时间
                              " +
                            $strUser.find("date").text() + "</div>"
                            strHTML += "</div>"
                        })
                        $(".divContent").html(strHTML); //显示处理后的数据
                    }
                })
        }
    })
```

以下代码用于响应客户端请求，并处理数据的服务器端页面AddData.aspx，其实现的代码如代码清单6-15所示。

代码清单 6-15　服务器端文件 AddData.aspx

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.IO" %>
<%
    string strName = System.Web.HttpUtility
                    .UrlDecode(Request["name"]);        //解码点评用户名称
    string strContent = System.Web.HttpUtility
                     .UrlDecode(Request["content"]); //解码点评提交内容
    string strFileName = "CommentData.xml";
    //定义 XML 文档变量
    XmlDocument xmlDoc = new XmlDocument();
    //打开指定的 XML 文档
    xmlDoc.Load(Server.MapPath(strFileName));
    //查找根节点元素
    XmlNode xmlN = xmlDoc.SelectSingleNode("Comment");
    //加入一个节点元素
    XmlElement xmlE = xmlDoc.CreateElement("Data");

    //创建一个子节点
    XmlElement xmlEn = xmlDoc.CreateElement("name");
    //设置节点文本
    xmlEn.InnerText = strName;
    //添加到节点中
    xmlE.AppendChild(xmlEn);
    //创建一个子节点
    XmlElement xmlEc = xmlDoc.CreateElement("content");
    //设置节点文本
    xmlEc.InnerText = strContent;
    //添加到节点中
    xmlE.AppendChild(xmlEc);
    //创建一个子节点
    XmlElement xmlEd = xmlDoc.CreateElement("date");
    //获取时间的时分秒
    string strSendTime = DateTime.Now.Hour + ":" +
                         DateTime.Now.Minute + ":" + DateTime.Now.Second;
    xmlEd.InnerText =strSendTime;
    //添加到节点中
    xmlE.AppendChild(xmlEd);

    //将节点加入根节点中
    xmlN.AppendChild(xmlE);
    //保存创建好的 XML 文档
    xmlDoc.Save(Server.MapPath(strFileName));
    Response.Write("您的点评已成功发表！ ");
%>
```

6.5.4 代码分析

为了即时侦察出客户端各种Ajax请求的进展状态，使用全局事件ajaxStart与ajaxStop绑定客户端页面中ID号为"divMsg"的元素，在请求触发和停止时，显示不同的内容和状态，其代码如下所示：

```
//元素绑定全局 ajaxStart 事件
$("#divMsg").ajaxStart(function() {
    $(this).show(); //显示元素
})
//元素绑定全局 ajaxStop 事件
$("#divMsg").ajaxStop(function() {
    $(this).html(" 数据处理已完成。").hide();
})
```

由于页面在初始化与提交点评内容后，都要加载XML格式的点评数据，为了避免重复，我们自定义一个JS函数LoadData，其功能就是使用$.ajax()方法，请求加载XML格式的点评数据，其代码如下：

```
/*
* 动态加载 XML 格式的点评数据
*/
function LoadData() {
    $.ajax(
        {
            type: "GET",
            url: "CommentData.xml", // 请求 XML 格式数据
            dataType: "xml",
            cache: false,
            success: function(data) {
                // 请求数据成功后执行的代码
                ...
            }
        })
}
```

在自定义读取XML点评数据的函数中，当请求成功而返回数据时，执行一个回调函数，在该函数中，先清空显示加载数据的页面元素，然后，遍历返回的数据，在遍历过程中，通过find()方法定位各指定名称的元素内容，并以叠加的方式保存在字符变量strHTML中，最后，通过元素的html()方法将数据显示在页面中，其实现的代码如下：

```
... 省略部分代码
$(".divContent").empty(); // 先清空标记中的内容
var strHTML = "";         // 初始化保存内容变量
if ($(data).find("Data").length == 0) {// 如果没有找到数据
     strHTML = "<div style='text-align:center'>
               目前还没有找到点评数据! </div>";
    }
$(data).find("Data").each(function() { // 遍历获取的数据
  var $strUser = $(this);
  strHTML += "<div class='clsShow'>";
  strHTML += "<div class='ShowTitle'>荣拓网友
               <a href=''>" + $strUser.find("name").text()+ "</a></div>";
  strHTML += "<div class='ShowContent'>" +
             $strUser.find("content").text() + "</div>";
  strHTML += "<div class='ShowBottom'>发送时间   " +
             $strUser.find("date").text() + "</div>"
```

```
    strHTML += "</div>"
  })
    $(".divContent").html(strHTML); // 显示处理后的数据
  ... 省略部分代码
```

当用户单击“提交”按钮时，先以加码的方式获取页面中“内容”和“用户名”的值，并将该值作为请求服务器的参数，执行一个$.ajax()方法，向服务器端页面AddData.aspx发出请求，其实现的代码如下所示：

```
... 省略部分代码
// 获取加码后的用户名
var strName = encodeURI($("#txtName").val());
// 获取加码后的发表内容
var strContent = encodeURI($("#txtContent").val());
$.ajax(
  {
    type: "GET",
    url: "AddData.aspx", // 请求增加数据动态页
    dataType: "html",
    data: { name: strName, content: strContent },
    success: function(msg) {
       // 请求数据成功后执行的代码
       ...
    }
})
... 省略部分代码
```

在单击“提交”按钮，传递参数请求服务器并响应成功后，执行一个请求成功的回调函数，在该函数中，先弹出一个窗口，显示服务器传回的值，然后，重新执行自定义的函数LoadData，即再次加载点评数据。最后，清空文本框“内容”和“用户名”中的值，其实现的代码如下所示：

```
... 省略部分代码
alert(msg);
LoadData();
$("#txtName").val("");
$("#txtContent").val("");
... 省略部分代码
```

6.6　本章小结

本章以理论与实例相结合的方式，介绍jQuery中支持Ajax的各种方法，阐述了客户端与服务器端通过Ajax互相访问的方式，并详细介绍了核心方法$.ajax()的运用，最后，系统介绍了Ajax中的全局函数与事件，并结合一个新闻点评的应用示例，帮助巩固前面所学的全部知识。

第7章　jQuery中调用JSON与XML数据

本章内容

jQuery调用JSON数据

jQuery调用XML数据

综合案例分析——调用JSON实现下拉列表框三级联动

综合案例分析——调用XML实现无刷新即时聊天

本章小结

JSON（JavaScript Object Notation）表示一种轻量级的数据交互格式，这种格式采用一种完全独立于编程语言的语法文本形式去交换数据。由于该格式易于阅读与编译，使其快速成为一种数据交换语言格式的首选。

XML（Extensible Markup Language）表示一种可扩展性语言标记，可以用来标记数据，定义类型；由于XML可以提供了一套独立且统一的方法与各类型应用程序进行数据交互，因此，它常常用于Web数据的存储与传输。

作为一种数据交互语言，JSON与XML有很多相似的地方，例如体积轻量型、编码易于阅读；同时，它们之间也存在差异，例如JSON编码的扩展性不及XML丰富，但在存储复杂的JavaScript复合对象数据时，JSON的优势是显而易见的。

在本章中，我们先从最基础的JSON与XML数据格式讲起，然后逐步介绍如何使用jQuery解析、遍历、操作这两种格式的数据，最后结合两个常用的案例，进一步阐述它们在实例中的运用方法。

7.1　jQuery调用JSON数据

JSON数据交互格式源于JavaScript语法，使用jQuery可以十分方便地读取该类型的数据。在jQuery中，有专门用于调用JSON格式的全局函数getJSON()，其使用方法在第6章中已详细介绍过，在此不赘述。

JSON是一种数据交换格式，它通过key/value或数组的形式保存数据，通过使用jQuery框架，我们可以很轻松地读取、遍历、修改其中的任意内容。接下来我们先从JSON的基础结构开始，逐步介绍其在jQuery框架中的运用过程。

7.1.1　JSON数据的基础知识

从功能上来说，JSON的特点是将JavaScript中的对象转换成一种轻量型、易解析的字符串，这种字符串不仅能在JavaScript中传递，也可以通过异步的方式在Web应用程序之间进行数据交互。JSON的结构包含两种，一种是name/value形式，另一种为数组格式，后者用于处理复合的JavaScript对象。JSON的对象转换过程，实质上就是将JavaScript对象转成上述两种结构的过程，接下来介绍这两种介绍的组成方式。

1.name/value形式表示

该形式是JSON格式中最为简单的一个表示方式，如下面代码：

```
{"name": "taoguorong"}
```

上述形式的代码功能等价于如下代码，表示"name"对应的值为"taoguorong"：

```
name=taoguorong
```

如果有多个对应关系，则这种表示方式的优势则显现出来，如下代码：

```
{
"name": "taoguorong",

 "email": "tao_guo_rong@163.com",
 "sex": "male"
 }
```

上述形式与简单的字符串相比，更加易于阅读。同时，通过大括号使括号中的每个值存在一一对应的关系，而且形成一个统一整体，各个元素是相互关联的。

2.数组形式表示

JSON也可以通过数组的形式来表示一组数据，实现的方式也十分简单，只需要将多个带花括号的记录通过括号组合成一个name名称对应的值，这种方式不仅简便易于理解，而且大大减少了数据的复杂性，如下代码：

```
{"1132":
[
{"id","102","name": "张小虎","chinese": "80","english": "75"},
{"id","103","name": "李成雄","chinese": "90","english": "85"}
...
]
}
```

上述代码中“1132”表示某班级的“编号”，是一个变量名称，该变量对应的值是通过中括号组合多个元素的数组，每一个元素对应多条记录，如"id"、"name"等。此外，可以使用多个变量名，对应更多的数组元素。如下代码：

```
{"1132":
[
{"id","102","name": "张小虎","chinese": "80","english": "75"},
{"id","103","name": "李成雄","chinese": "90","english": "85"}
]
"1133":
[
{"id","112","name": "刘大成","maths": "82","music": "76"},
{"id","113","name": "罗明汉","maths": "95","music": "87"}
]
"1134":
[
{"id","122","name": "钟伟","art": "80","sports": "74"},
{"id","123","name": "秦明","art": "90","sports": "82"}
]
...
}
```

上述代码中“1132”、“1133”、“1134”分别表示班级的“编号”，属于变量名，各自对应数组元素。在数组元素对应的记录中，由于JSON在定义数据结构时完全是动态的，因此，同一个结构的变量名对应的记录结构可以不同。如上述示例中，变量名“1132”中存在"chinese"记录名称，而其他变量则没有。

7.1.2　jQuery读取JSON数据

在初步了解JSON数据格式之后，接下来学习在jQuery框架下，如何读取一个JSON数据。由于JSON格式是JavaScript的原生态格式，因此，使用jQuery访问JSON数据是十分方便的，只需要将一个JSON格式的数据赋值给一个变量，就可以通过点号“.”，以“变量名.记录名称”的形式读取名称对应的值。下面通过一个简单的示例加以说明。

示例7-1　使用jQuery读取JSON数据

（1）功能描述

在页面中，当用户单击“获取数据”按钮时，将读取一个JSON格式数据，并将读取的内容显示在页面中。

（2）实现代码

新建一个HTML文件7-1.html,加入的代码如代码清单7-1所示。

代码清单　7-1 使用 jQuery 读取 JSON 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 读取JSON 数据</title>
    <script src="JScript/jquery-1.8.2.min.js"
          type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          .iframe{width:260px;border:solid 1px #666}
          .iframe .title{padding:5px;background-color:#eee}
          .iframe .content{padding:8px;font-size:12px}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
          EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        var objInfo = {
            'name': '陶国荣',
            'sex': '男',
            'email': 'tao_guo_rong@163.com'
        };
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                var strHTML = "";              //初始化保存内容变量
                strHTML += "姓名: " + objInfo.name + "<br>";
                strHTML += "性别: " + objInfo.sex + "<br>";
                strHTML += "邮箱: " + objInfo.email + "<hr>";
                $("#Tip").html(strHTML); //显示处理后的数据
            }); ;
        });
    </script>
</head>
<body>
   <div class="iframe">
         <div class="title">
              <input id="Button1" type="button"
                     class="btn" value="获取数据" />
         </div>
         <div class="content">


              <div id="Tip"></div>
         </div>
    </div>
</body>
</html>
```

（3）页面效果

执行后的效果如图7-1所示。

图　7-1　jQuery读取JSON格式数据

（4）代码分析

在示例7-1中，先将一个JSON格式的数据赋值给变量"objInfo"，当用户单击“获取数据”按钮时，以“变量名.名称”的方式读取JSON中名称对应的值，并将该值的内容以累加的方式进行保存在变量"strHTML"中。最后，将变量的内容显示在ID号为"Tip"的元素中，实现读取并显示JSON格式数据的功能。

7.1.3　jQuery遍历JSON数据

在JSON格式数据中，通常使用"name/value"形式来展示一些简单的数据，而对于复杂的JavaScript对象，则往往需要借助数组的形式，而JSON中的数组是针对"value"值而言的，如果以数组的形式表示"value"值，那么将以左中括号“[”开始，以右中括号“]”结束，括号中间可以一条或多条JSON数据，各数据之间用逗号隔开；另外，"value"值的类型包括字符型、数值型、逻辑型、对象或数组，并支持各种类型值的嵌套形式。

由于JSON数据中的"value"值是以数组的形式展示的，因此，在取值时，只需要遍历整个JSON数据，以“变量名.名称”的形式逐层读取即可。接下来通过一个简单的示例来演示在jQuery框架中遍历一个JSON数据的过程。

示例7-2　使用jQuery遍历JSON数据

（1）功能描述

在页面的导航栏中，显示全部优秀学生所在的班级，单击班级链接后，在内容区中显示所选班级对应的学生信息。

（2）实现代码

新建一个HTML文件7-2.html,加入如代码清单7-2所示的代码。

代码清单 7-2　使用 jQuery 遍历 JSON 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 遍历 JSON 数据 </title>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          .iframe{width:260px;border:solid 1px #666}
          .iframe .title{padding:5px;background-color:#eee}
          .iframe .content{padding:8px;font-size:12px}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
          EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        var objData = {
            member: [{
                grade: "一年级",
                students: {
                    name: ["刘小芳", "李大明"]
                }
            }, {
                grade: "二年级",
                students: {
                    name: ["陈优", "王小海", "朱红"]
                }
            }, {
                grade: "三年级",
                students: {
                    name: ["张妍", "鼻霞"]
                }
            }]
        };
        function add_Grade() {
            var objmember = objData.member;
            var strHTML_0 = ""; // 初始化保存内容变量
            $.each(objmember, function(index) {
                strHTML_0 += '<a href="javascript:"
                onclick="lnk_Click(' + index + ')">'
                + objmember[index].grade + '</a>  ';
            });
            $(".title").html("年级优秀学生: " + strHTML_0);
        };
        function lnk_Click(i) {
            var objstudent =  objData.member[i].students.name;
            var strHTML_1 = ""; // 初始化保存内容变量
            $.each(objstudent, function(index) {
                strHTML_1 += ' ' + objstudent[index] +' ';
            });
            $("#Tip").html(strHTML_1);
        };
```

```
            $(function() {
                add_Grade();
                lnk_Click(0)
            });
    </script>
</head>
<body>
    <div class="iframe">
        <div class="title"></div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

执行后的效果如图7-2所示。

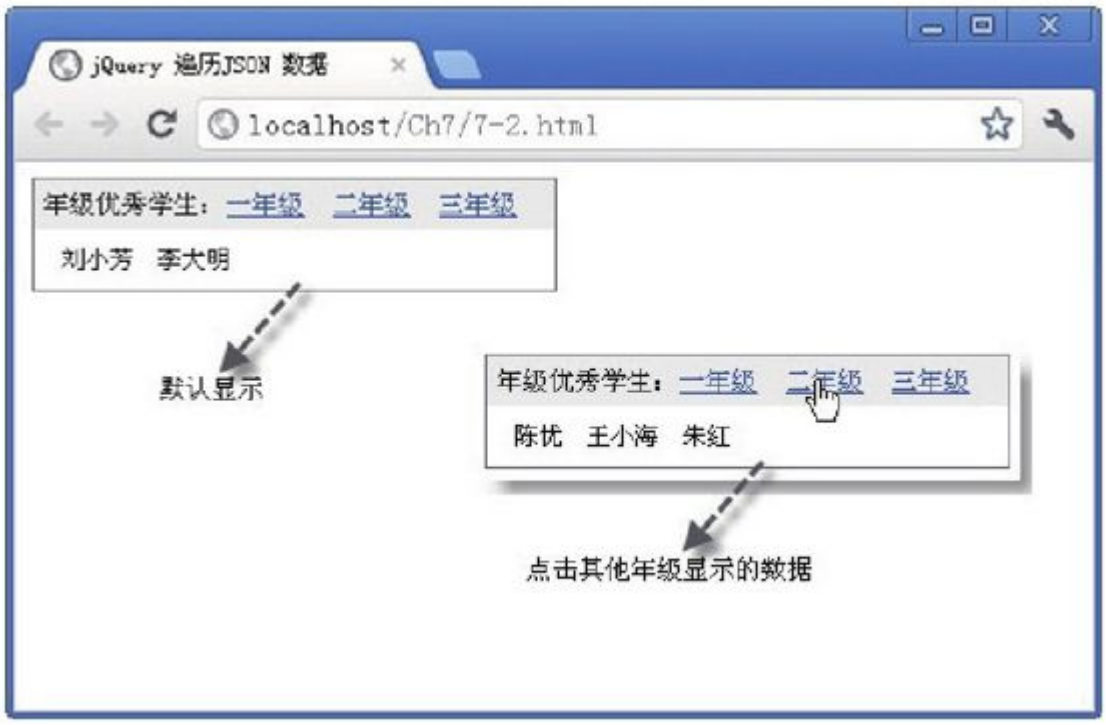


图 7-2 jQuery遍历JSON数据

（4）代码分析

在本示例的JSON格式数据中，由于"member"名称对应的"value"值是一个数组，为了获取包含在数组名年级名称对应的值，编写一个自定义函数add_Grade()。在该函数中，先遍历"objData.member"数据，在遍历过程中通过"objData.member[index].grade"的形式获取各对应年级的名称，其中"index"为数组的索引号，该值从0开始。如"objData.member[0].grade"表示获取第1个年级"grade"名称对应的值，即“一年级”。最后，将获取的全部年级值以累加的方式保存在变量"strHTML_0"中，通过指定的元素显示在页面中。

用户单击“年级”时，为了实现显示对应学生数据的功能，编写另外一个自定义函数lnk_Click(i)。调用该函数时，需要传递一个年级对应的索引号值，在函数中根据该传回的索引号值，以"objData.member[i].students.name"形式获取该索引号下，以及年级对应的学生"name"名称数据。最后遍历该数据，以累加的方式保存在变量"strHTML_1"中，并通过指定的元素将变量内容显示在页面中，详细见代码中加粗部分。

7.1.4　jQuery操作JSON数据

在示例7-2中，使用jQuery中的全局函数$.each遍历JSON对象，获取"name"对应的"value"值。另外，当获取JSON对象后，还可以对原有的JSON数据中的"value"值进行修改，实现的方法就是找到该对象中需要修改的"name"名称，并对该名称重新进行赋值。下面通过一个简单的示例加以说明。

示例7-3　使用jQuery操作JSON数据

（1）功能描述

在页面中，当用户单击“原始数据”按钮时，将读取一个JSON格式的数据并将内容显示在页面中，同时按钮中的文字显示为“变化数据”，此时再单击按钮时，将读取一个变化后的JSON格式数据并将内容显示在页面中。

（2）实现代码

新建一个HTML文件7-3.html,加入的代码如代码清单7-3所示。

代码清单 7-3　使用 jQuery 操作 JSON 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 操作 JSON 数据 </title>
    <script src="JScript/jquery-1.8.2.min.js"
          type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          .iframe{width:260px;border:solid 1px #666}
          .iframe .title{padding:5px;background-color:#eee}
          .iframe .content{padding:8px;font-size:12px}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
          EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        var strInfo = "{'name': '陶国荣','sex': '男',
                        'email': 'tao_guo_rong@163.com',
                        'date': 13493408373591}";
        $(function() {                          //定义按钮文字变量
            var strV0 = "原始数据";
            var strV1 = "变化数据";
            $("#Button1").click(function() {//按钮单击事件
                var strHTML = "";               //初始化保存内容变量
                var objInfo = eval('(' + strInfo + ')');//将字符串转成 JSON 对象
                //根据按钮文字改变 JSON 对象中的值
                if ($(this).val() == strV1) {
                    objInfo.date = new Date().getTime();
                }
```

```
                strHTML += "姓名: " + objInfo.name + "<br>";
                strHTML += "性别: " + objInfo.sex + "<br>";
                strHTML += "邮箱: " + objInfo.email + "<br>";
                strHTML += "时间: " + objInfo.date + "<hr>";
                //切换按钮显示的文字
                if ($(this).val() == strV0) {
                    $(this).val(strV1);
                } else {
                    $(this).val(strV0);
                }
                //显示处理后的数据
                $("#Tip").html(strHTML);
            });
        });
    </script>
</head>
<body>
    <div class="iframe">
        <div class="title">
            <input id="Button1" type="button"
                   class="btn" value="原始数据" />
        </div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

执行后的效果如图7-3所示。

图　7-3　jQuery操作JSON数据

（4）代码分析

在本示例中，根据单击按钮显示的文本内容，在页面中显示原始或修改后的JSON数据。

如果按钮文本内容为“原始数据”，则单击该按钮时，先将字符串通过eval()方法转换成JSON格式的对象，再以“对象.属性”的方式读取JSON格式数据中的各个"value"值，并以叠加的形式保存在变量中，最后指定的元素将变量中的内容显示在页面中。

当按钮文本内容为“变化数据”时，如果单击该按钮，则在字符串转换成JSON格式对象之后，采用“对象.属性”的方式对"date"名称

进行再次赋值，并以同样的方式读取修改后的"value"值，最后将全部的内容显示在页面指定的元素中。

7.2　jQuery调用XML数据

作为一种理想的数据交互语言，XML与JSON有太多的相似之处，在编码的可读性和解码难度上，XML都优于JSON，只是在处理复合的JavaScript对象时不及JSON。

在表现形式上，XML采用的是标准的标签，因此，如果需要完整地解析一个XML格式的数据，有两种方式：一种方式是通过DOM文档模型进行解析，另一种方式是遍历各个标签的节点（如childNodes等）。本节将逐步介绍在jQuery中如何解析、遍历、操作XML数据的详细过程。

7.2.1　使用传统JavaScript调用XML的方法

在介绍使用jQuery调用XML数据之前，我们先来看看在传统的JavaScript代码中，是如何借助XMLHttpRequest对象，打开并读取一个XML格式数据的。

示例7-4　使用传统JavaScript调用XML的方法

（1）功能描述

在页面中，用户单击“获取数据”按钮后，将利用传统的JavaScript代码，打开一个指定的XML文件并读取该文件中的内容，最后显示在页面中。

（2）实现代码

新建一个HTML文件7-4.html，加入的代码如代码清单7-4所示。

代码清单 7-4　使用传统 JavaScript 调用 XML 的方法

```
<!DOCTYPE html>
<html>
<head>
    <title>使用传统 JavaScript 调用 XML 的方法 </title>
    <style type="text/css">
           body{font-size:13px}
           .iframe{width:260px;border:solid 1px #666}
           .iframe .title{padding:5px;background-color:#eee}
           .iframe .content{padding:8px;font-size:12px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        // 加载 XML 文件
        function loadXML(xmlFile) {
```

```
            var xmlDoc, xmlhttp;
            // 兼容 IE 7+、Firefox、Chrome、Opera、Safari
            if (window.XMLHttpRequest) {
             xmlhttp = new XMLHttpRequest();
            }
            else {// 兼容 IE 5、IE 6
             xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.open("GET", xmlFile, false);
            xmlhttp.send();
            xmlDoc = xmlhttp.responseXML;
            return xmlDoc;
        }
        function btn_Click() { //按钮单击事件
            var strHTML = "";  // 初始化保存内容变量
            var NewXmlDoc = loadXML("Xml/7-4.xml");
            var NewTmpTag = NewXmlDoc.getElementsByTagName("User");
            strHTML += "姓名：" + NewTmpTag[0]
                       .getElementsByTagName("name")[0]
                       .firstChild.nodeValue + "<br>";
            strHTML += "性别：" + NewTmpTag[0]
                       .getElementsByTagName("sex")[0]
                       .firstChild.nodeValue + "<br>";
            strHTML += "邮箱：" + NewTmpTag[0]
                       .getElementsByTagName("email")[0]
                       .firstChild.nodeValue + "<hr>";
            // 显示处理后的数据
            document.getElementById("Tip").innerHTML = strHTML;
        }
    </script>
</head>
<body>
   <div class="iframe">
        <div class="title">
            <input type="button" onclick="btn_Click()"
                   class="btn" value="获取数据" />
        </div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

本示例的源码中调用的XML文件"7-4.xml"内容如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<Info>
  <User>
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>
</Info>
```

（3）页面效果

执行后的效果如图7-4所示。

图　7-4　使用传统JavaScript调用XML的方法

（4）代码分析

在本示例中，为了获取7-4.xml文件中的内容，首先自定义一个loadXML(f)函数，其中形参"f"为调用XML文件的路径。在该函数中，先根据不同浏览器取得XMLHttpRequest对象，并将该对象并保存至xmlhttp变量中，然后将传来的实参"f"值作为参数调用该对象的open()方法，最后调用该对象的send()方法获取XML文件读取成功后的数据对象，并作为函数的返回值传递给调用函数。

在用户单击按钮时，为了将获取后的XML数据显示在页面中，自定义一个btn_Click()函数。在该函数中，先调用loadXML(f)函数，并将该函数返回的数据对象保存至变量NewXmlDoc中；然后，调用getElementsByTagName()方法获取数据对象中的各个标签，使用"firstChild.nodeValue"形式获取各标签下对应的值；最后，将获取的各标签值以叠加的形式保存至变量，通过指定的元素将变量的内容显示在页面中。

7.2.2　jQuery解析XML数据

与传统的JavaScript调用XML数据相比，使用jQuery框架解析XML数据将方便很多，通常情况下，在jQuery框架中，先使用$.ajax()方法请求并打开指定的XML文件，然后，在回调函数中获取返回的XML文件数据对象，使用find()、children()、text()方法获取各标签下的元素并读取元素的值。下面通过一个简单的示例来介绍jQuery解析XML文件中的数据。

示例7-5　jQuery解析XML数据

（1）功能描述

在页面中，用户单击“获取数据”按钮时，将调用jQuery框架中的$.ajax()方法打开指定的XML文件，然后通过遍历的方式将读取的数据显示在页面中。

（2）实现代码

新建一个HTML文件7-5.html,加入的代码如代码清单7-5所示。

代码清单 7-5　jQuery 解析 XML 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 解析XML 数据</title>
    <script src="JScript/jquery-1.8.2.min.js"
          type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          .iframe{width:260px;border:solid 1px #666}
          .iframe .title{padding:5px;background-color:#eee}
          .iframe .content{padding:8px;font-size:12px}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
          EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                var strHTML = ""; //初始化保存内容变量
                $.ajax({
                    url: 'Xml/7-5.xml',
                    dataType: 'xml',
                    success: function(data) {
                        var $strUser = $(data).find("User");
                        strHTML += "编号：" + $strUser
                                .attr("id")+ "<br>";
                        strHTML += "姓名：" + $strUser
                                .children("name").text() + "<br>";
                        strHTML += "性别：" + $strUser
                                .children("sex").text() + "<br>";
                        strHTML += "邮箱：" + $strUser
                                .children("email").text() + "<hr>";
                        //显示处理后的数据
                        $("#Tip").html(strHTML);
                    }
                });
            });
        });
    </script>
</head>
<body>
   <div class="iframe">
        <div class="title">
            <input id="Button1" type="button"
                  class="btn" value="获取数据" />
        </div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

本示例的源码中调用的XML文件"7-5.xml"的内容如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<Info>
  <User id="10001">
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>
</Info>
```

（3）页面效果

执行后的效果如图7-5所示。

图　7-5　jQuery解析XML数据

（4）代码分析

在本示例中，当用户单击“获取数据”按钮时，先调用$.ajax()方法打开指定路径的XML文件。在该方法的回调函数中，获取传回的XML格式数据对象data；然后调用data数据对象的find()方法，获取XML文件中的根目录标签"User"，并保存在"$strUser"变量中；最后，通过调用attr()、children()、text()方法获取标签的属性和内容值，并将这些值以叠加的形式保存在变量中，再通过指定的元素将变量的内容显示在页面中。

7.2.3　jQuery读取XML数据

在jQuery中，除示例7-5中解析XML文件中的数据之外，还可以对获取的XML数据进行分组或遍历，实现的方法也十分简单，下面通过一个示例来进行介绍。

示例7-6　jQuery读取XML数据

（1）功能描述

在页面中，用户单击“获取数据”按钮时，将打开指定的XML文件获取相应的数据对象，并以年班分组的方式将各年级的同学信息显示在页面中。

（2）实现代码

新建一个HTML文件7-6.html,加入的代码如代码清单7-6所示。

代码清单 7-6　jQuery 读取 XML 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 读取XML 数据</title>
    <script src="JScript/jquery-1.8.2.min.js"
          type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          h3{ padding:0px; margin:8px 0px}
          .iframe{width:260px;border:solid 1px #666}
          .iframe .title{padding:5px;background-color:#eee}
          .iframe .content{padding:8px;font-size:12px}
          .btn {border:#666 1px solid;padding:2px;width:80px;
          filter: progid:DXImageTransform.Microsoft
          .Gradient(GradientType=0,StartColorStr=#ffffff,
          EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        var arrGrade = new Array(980886, 980666);
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                var strHTML = "";           // 初始化保存内容变量
                $.ajax({
                   url: 'Xml/7-6.xml?',
                    dataType: 'xml',
                    success: function(data) {
                      $.each(arrGrade, function(i) {
                          var $strUser = $(data).find("User[grade=" + arrGrade[i] + "]");
                          strHTML += "<h3>年级: " + arrGrade[i] + "</h3>";
                          $strUser.each(function() {
                             strHTML += "姓名: " + $(this)
                                    .children("name").text() + "<br>";
                             strHTML += "性别: " + $(this)
                                    .children("sex").text() + "<br>";
                             strHTML += "邮箱: " + $(this)
                                    .children("email").text() + "<hr>";
                          });
                      });
                    // 显示处理后的数据
                    $("#Tip").html(strHTML);
                    }
                });
            });
        });
    </script>
</head>
<body>
   <div class="iframe">
        <div class="title">
            <input id="Button1" type="button"
                  class="btn" value="获取数据" />
        </div>
        <div class="content">


            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

在本示例的源码中调用的XML文件"7-6.xml"所包含的内容如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<Info>
  <User grade="980886">
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>
  <User grade="980886">
    <name> 李建洲 </name>
    <sex> 女 </sex>
    <email>xiaoli@163.com</email>
  </User>
  <User grade="980666">
    <name> 张天虎 </name>
    <sex> 男 </sex>
    <email>tianhu@163.com</email>
  </User>
  <User grade="980666">
    <name> 陈小燕 </name>
    <sex> 女 </sex>
    <email>xiaoyan@163.com</email>
  </User>
</Info>
```

（3）页面效果

执行后的效果如图7-6所示。

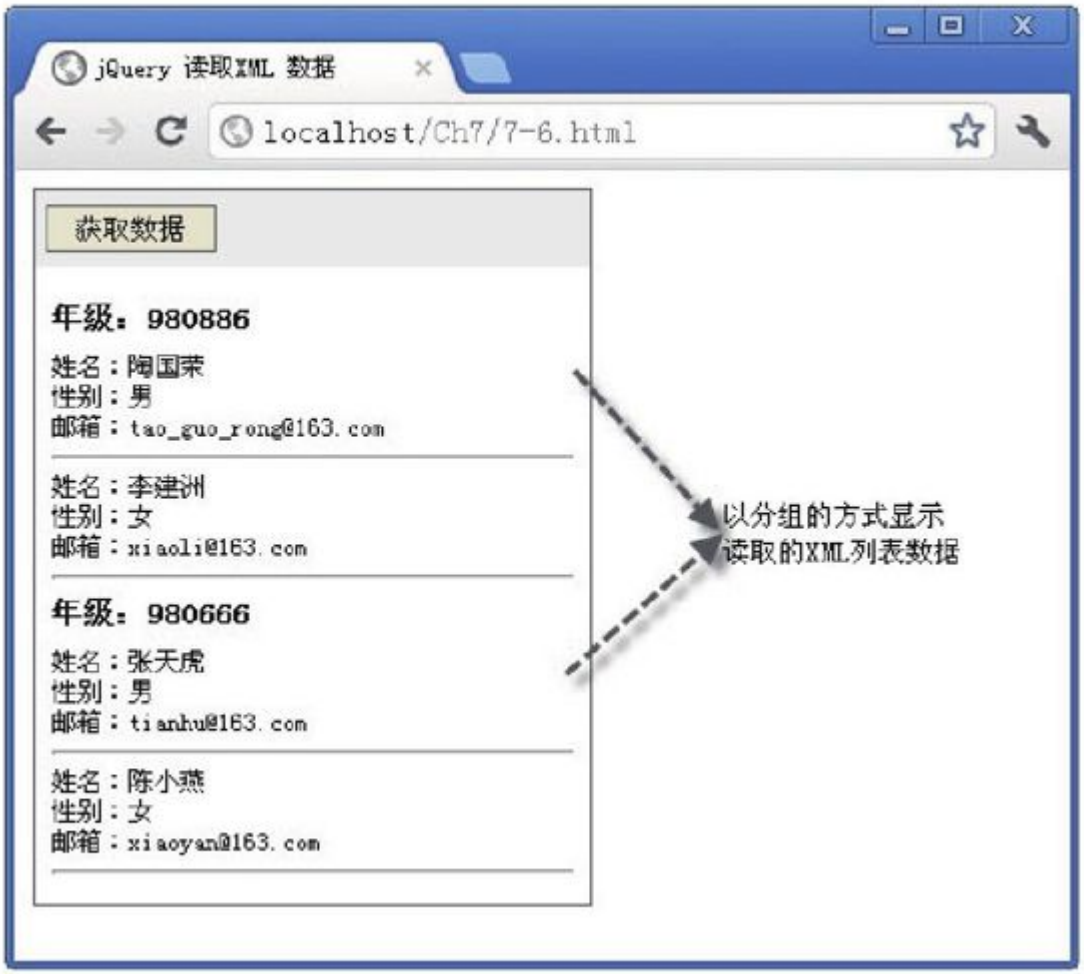


图　7-6　jQuery读取XML数据

（4）代码分析

在本示例中，为了实现分组显示列表数据的功能，分别两次调用了jQuery框架中的$.each函数。第一次用于遍历年级数组，在遍历过程中，根据获取的各个年级数组编号，通过find()方法寻找与编号相对应的学生数据信息，并对该获取的信息进行第二次遍历；在这次遍历过程中，分别通过单个遍历元素调用children()、text()方法，获

取各标签结点中的内容值，并将这些值以叠加的形式保存在变量中，再通过指定的元素将变量的内容显示在页面。在第二次遍历过程中，$(this)表示遍历时的当前元素自身，详细见代码中加粗部分。

7.2.4　jQuery操作XML数据

XML数据以xml为扩展名的文件形式保存，如果修改文件中的数据，必须与服务端配合。使用jQuery向服务端发送请求，由服务端编写代码，修改XML文件的内容。接下来以服务端为C#语言为例，介绍jQuery操作XML数据的过程。

示例7-7　jQuery操作XML数据

（1）功能描述

在示例7-6的基础之上，向XML数据中增加一个学生ID，并在页面中增加一个“删除”链接。用户单击该链接时，获取传回的学生ID号，向服务端发送删除请求；服务端接收请求后，删除对应学生"ID"记录，修改XML数据，并将删除成功的标志返回前端页面；页面接收标志后，重新分组获取对应年级的学生数据，实现删除功能。

（2）实现代码

新建一个HTML文件7-7.html,加入代码如代码清单7-7所示。

代码清单 7-7　jQuery 操作 XML 数据

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery 操作XML数据</title>
    <script src="JScript/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <style type="text/css">
           body{font-size:13px}
           h3{ padding:0px; margin:8px 0px}
           .iframe{width:260px;border:solid 1px #666}
           .iframe .title{padding:5px;background-color:#eee}
           .iframe .content{padding:8px;font-size:12px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff, \
           EndColorStr=#ECE9D8);}
           #Tip a{ float:right}
    </style>
    <script type="text/javascript">
        var arrGrade = new Array(980886, 980666);
        // 自定义根据 ID 号删除记录函数
```

```
        function lnk_Delete(i) {
            var btnYN = confirm("您真的要删除吗? ");
            if (btnYN) {
                $.post("7-7.ashx?Id=" + i, function(data) {
                    if (data) {// 重新执行单击获取数据按钮事件
                        $("#Button1").click();
                    } else {
                        alert(data);
                    }
                });
            }
        }
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                var strHTML = ""; // 初始化保存内容变量
                $.ajax({
                   url: 'Xml/7-7.xml?',
                   dataType: 'xml',
                   success: function(data) {
                     $.each(arrGrade, function(i) {
                       var $strUser = $(data).find("User[grade=" + arrGrade[i] + "]");
                       strHTML += "<h3>年级: " + arrGrade[i] + "</h3>";
                       $strUser.each(function() {
                           var $strId = $(this).children("id").text();
                           strHTML += "编号: " + $strId + "<a href='javascript:'
                           onclick='lnk_Delete(" + $strId + ")'>删除</a><br>";
                           strHTML += "姓名: " + $(this).children("name").text() + "<br>";
                           strHTML += "性别: " + $(this).children("sex").text() + "<br>";
                           strHTML += "邮箱: " + $(this).children("email").text() + "<hr>";
                           });
                       });
                       // 显示处理后的数据
                       $("#Tip").html(strHTML);
                   }
                });
            });
        });
    </script>
</head>
<body>
   <div class="iframe">
        <div class="title">
            <input id="Button1" type="button"
                   class="btn" value="获取数据" />
        </div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

在本示例的源码中调用的XML文件"7-7.xml"所包含的内容如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<Info>
  <User grade="980886">
    <id>1001</id>
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>
  <User grade="980886">
    <id>1002</id>
    <name> 李建洲 </name>
    <sex> 女 </sex>
    <email>xiaoli@163.com</email>
  </User>
  <User grade="980666">
    <id>1003</id>
    <name> 张天虎 </name>
    <sex> 男 </sex>
    <email>tianhu@163.com</email>
  </User>
  <User grade="980666">
    <id>1004</id>
    <name> 陈小燕 </name>
    <sex> 女 </sex>
    <email>xiaoyan@163.com</email>
  </User>
</Info>
```

此外，本示例中新建了一个名为"7-7.ashx"的服务端程序，该程序的功能是根据获取的记录ID号删除对应的XML数据结点，实现的代码如下所示：

```
<%@ WebHandler Language="C#" Class="_7_7" %>
using System;
using System.Web;
using System.Xml;
public class _7_7 : IHttpHandler {
    public void ProcessRequest(HttpContext context) {
        context.Response.ContentType = "text/plain";
        string strId = context.Request.QueryString["Id"].ToString();
        int intStatus = 0;
        XmlDocument xmlDoc = new XmlDocument();
        try {
            xmlDoc.Load(context.Server.MapPath("Xml/7-7.xml"));// 查找
            XmlNodeList xmlNodeList = xmlDoc.SelectNodes("Info/User[id='" + strId + "']");
            XmlNode xmlNode = xmlNodeList.Item(0);
            xmlNode.ParentNode.RemoveChild(xmlNode);
            xmlDoc.Save(context.Server.MapPath("Xml/7-7.xml"));
            intStatus = 1;
        }
        catch (Exception) {
            throw;
        }
        context.Response.Write(intStatus);
    }
    public bool IsReusable {
        get {

            return false;
        }
    }
}
```

（3）页面效果

执行后的效果如图7-7所示。

图　7-7　jQuery操作XML数据

（4）代码分析

在本实例中，为了实现修改XML数据的功能，在代码中增加了一个自定义的函数lnk_Delete(i)。在该函数中，i为表示学生编号的形参，调用该函数时，通过全局函数$.post()将传来的学生编号发送到服务端，服务端接收该编号，在XML文件中查询该编号，并将查询到的节点移除；最后，再次在服务端中保存该XML文件，同时向页面发送删

除成功的标志，页面接收该标志后，以"$("#Button1").click()"形式重新加载一次XML数据，显示删除成功后的页面效果。

7.3　综合案例分析——调用JSON实现下拉列表框三级联动

7.3.1　需求分析

本案例的需求包括如下两个方面：

1）在页面中，以并列的方式放置三个下拉列表框。用户选择第一个下拉列表框时，另外两个下拉列表框的值将随之变化；用户选择第二个下拉列表框时，第三个下拉列表框的值也将随之发生变化。

2）在三个下拉列表框中，无论选择哪一个下拉列表框，都将各个下拉列表框当前选择的内容显示在页面中。

7.3.2 界面效果

案例实现的页面效果如下图7-8所示。

图 7-8 使用jQuery调用JSON实现下拉列表框三级联动

7.3.3 功能实现

在项目中，新建一个HTML文件SeleArea.html，加入的代码如代码清单7-8所示。

代码清单 7-8 使用 jQuery 调用 JSON 实现列表框三级联动

```
<!DOCTYPE html>
<html>
<head>
    <title>使用 JSON 和 jQuery 实现列表框三级联动 </title>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
            body{font-size:13px}
            h3{ padding:0px; margin:8px 0px}
            .iframe{width:360px;border:solid 1px #666}
            .iframe .title{padding:5px;background-color:#eee}
            .iframe .content{padding:8px;font-size:12px}
    </style>
    <script type="text/javascript">
        // 将 JSON 格式的数据保存在变量中
        var AreaData = {
            "省份": { val: "", items: { "城市": { val: "", items: { "区县": ""}}} },
            "北京市": { val: "01", items:
                { "所辖区": { val: "0101", items:
                        { "东城区": "010101", "西城区": "010102" }
                            },    "所辖县": { val: "0102", items:
                                { "密云县": "010201", "延庆县": "010202"}}}
                    },
             "天津市": { val: "02", items:
```

```
        { "所辖区": { val: "0201", items:
                { "和平区": "020101", "河东区": "020102" }
                  },      "所辖县": { val: "0202", items:
                    { "宁河县": "020201", "静海县": "020202"}}}
            },
    "上海市": { val: "03", items:
        { "所辖区": { val: "0301", items:
                { "黄浦区": "030101", "卢湾区": "030102" }
                 },       "所辖县": { val: "0302", items:
                    { "奉贤县": "030201"}}}
            }
};
$(function() {
    //使用变量保存各页面元素
    var $s1 = $("#sele1");
    var $s2 = $("#sele2");
    var $s3 = $("#sele3");
    var $t1 = $("#Tip");
    //定义显示选中值变量
    var $t2 = "";
    //设置各个下拉列表框的默认值
    var v1 = "01";
    var v2 = "0101";
    var v3 = "010101";
    //设置第一个下拉列表框的数据
    $.each(AreaData, function(k, v) {
        appendOptTo($s1, k, v.val, v1);
    });
    //自定义添加下拉列表框数据函数
    function appendOptTo($o, k, v, d) {
        var $opt = $("<option>").text(k).val(v);
        if (v == d) { $opt.attr("selected", "true") }
        $opt.appendTo($o);
    };
    //编写第一个下拉列表框内容改变事件并执行该事件
    $s1.change(function() {
        $s2.html("");
        $s3.html("");
        if (this.selectedIndex == -1) return;
        var s1_curr_val = this.options[this.selectedIndex].value;
        var s1_curr_txt = this.options[this.selectedIndex].text;
        $.each(AreaData, function(k, v) {
            if (s1_curr_val == v.val) {
                if (v.items) {
                    $.each(v.items, function(k, v) {
                        appendOptTo($s2, k, v.val, v2);
                    });
                }
            }
        });
        $t2 = s1_curr_txt;
        $t1.html($t2);
        $s2.change();
    }).change();
    //编写第二个下拉列表框内容改变事件并执行该事件
```

```
            $s2.change(function() {
                $s3.html("");
                var s1_curr_val = $s1[0].options[$s1[0].selectedIndex].value;
                if (this.selectedIndex == -1) return;
                var s2_curr_val = this.options[this.selectedIndex].value;
                $.each(AreaData, function(k, v) {
                    if (s1_curr_val == v.val) {
                        if (v.items) {
                            $.each(v.items, function(k, v) {
                                if (s2_curr_val == v.val) {
                                    $.each(v.items, function(k, v) {
                                        appendOptTo($s3, k, v, v3);
                                    });
                                }
                            });
                        }
                    }
                });
                showChangeItem();
            }).change();
            // 编写第三个下拉列表框内容改变事件
            $s3.change(function() {
                showChangeItem();
            });
            // 自定义得列表框选择内容显示在页面中的函数
            function showChangeItem() {
                var s1_curr_txt = $s1[0].options[$s1[0].selectedIndex].text;
                var s2_curr_txt = $s2[0].options[$s2[0].selectedIndex].text;
                var s3_curr_txt = $s3[0].options[$s3[0].selectedIndex].text;
                $t2 = " 您所选择的是: " + s1_curr_txt + " > "
                    + s2_curr_txt + " > " + s3_curr_txt;
                $t1.html($t2);
            }
        });
    </script>
</head>
<body>
    <div class="iframe">
        <div class="title">请选择省市:
            <select id="sele1" name="sele1"></select>
            <select id="sele2" name="sele2"></select>
            <select id="sele3" name="sele3"></select>
        </div>
        <div class="content">
            <div id="Tip"></div>
        </div>
    </div>
</body>
</html>
```

7.3.4 代码分析

在本实例中，第一步将所有JSON格式的数据保存在变量AreaData中，用于后续下拉列表框数据的调用。第二步，自定义了一个appendOptTo函数，将获取的数据绑定＜option＞元素的value和text属性值，并通过元素的appendTo方法插入到各个拉列表框元素内。第三步，通过下列代码，调用自定义的appendOptTo函数，将JSON格式中的“省份”名称插入到ID号为"sele1"的下拉列表框中。

```
$.each(AreaData, function(k, v) {
    appendOptTo($s1, k, v.val, v1);
});
```

上面三个步骤，只是完成了ID号为"sele1"的下拉列表框数据的绑定。接下来需要根据ID号为"sele1"的下拉列表框所选择值，获取对应JSON格式中“城市”和“区县”的数据，并分别绑定ID号为"sele2"和"sele3"的下拉列表元素。具体步骤如下。

1）在ID号为"sele1"的下拉列表元素的"change"事件中，先获取下拉列表框所选中的值，再遍历整个JSON格式数据。在遍历过程中，将该选中的值与遍历时“省份”名称对应的value值匹配，如果一致，则获取与该名称对应的"items"数据，并将该数据中“城市”名称对应的value值插入到ID号为"sele2"的下拉列表框中。

2）在ID号为"sele2"的下拉列表元素的"change"事件中，以相同的方法筛选JSON格式数据中与所选“省份”和“城市”名称对应的value值一致的"items"数据，并将该数据中“区县”名称对应的value值，通过遍历和调用自定义函数的方式，插入到ID号为"sele3"的下拉列表框中。

3）为了显示各个下拉列表框当前所选中的文本内容，自定义一个showChangeItem函数。该函数的主要功能将各个下拉列表框中所选中的文本内容显示在页面元素中，在ID号为"sele2"和"sele3"下拉列表的"change"事件中，分别调用了该函数，以实现动态显示选中内容的功能，详细过程见代码中加粗部分所示。

7.4　综合案例分析——调用XML实现无刷新即时聊天

7.4.1　需求分析

本案例有以下两方面的需求：

1）成功登录后获取用户的ID号，根据选择的ID号，实现用户之间即时且无刷新的聊天功能；并且在不同用户使用同一页面时，区分显示的聊天内容。

2）所有对话数据以XML格式保存在服务端，在页面显示内容或进行对话时，将调用服务端提供的接口，由接口负责返回或处理XML数据。

7.4.2 界面效果

案例实现的页面效果如图7-9所示。

图 7-9 使用jQuery调用XML实现无刷新即时聊天

7.4.3　功能实现

在项目中，新建一个HTML文件Chat.html，加入如代码清单7-9所示的代码。

代码清单 7-9　使用 jQuery 调用 XML 实现无刷新即时聊天

```
<!DOCTYPE html>
<html>
<head>
    <title>使用 jQuery 调用 XML 实现无刷新即时聊天 </title>
    <script src="JScript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
            body{font-size:13px}
            h3{ padding:0px; margin:8px 0px}
            .iframe{width:260px;border:solid 1px #666}
            .iframe .title{padding:5px;background-color:#eee}
            .iframe .title .spnId{font-weight:bold;font-family:Arial}
            .iframe .content{padding:8px;font-size:12px}
            .iframe .content .lst{height:180px;border:solid 1px #ccc;padding:3px;
            line-height:1.5em; overflow-y:scroll;}
            .iframe .content .nav{color:#006EFE}
            .iframe .content .nav .time,
            .iframe .content .msg{margin-left:5px}
            .iframe .content .txt{height:50px;width:238px;border:solid 1px #ccc}
            .btn {border:#666 1px solid;padding:2px;width:80px;
            filter: progid:DXImageTransform.Microsoft
            .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
    </style>
```

```
    <script type="text/javascript">
        $(function() {
            //设置聊天时本人和对方 ID
            var $FromId = 1243242;
            var $SendId = 1234560;
            $("#spnSendId").html($SendId);
            //定时获取与对方的聊天数据
            setTimeout(showList($FromId, $SendId), 3000);
            $("#btnSend").click(function() {
                $.ajax({
                    url: 'Chat.ashx?s=' + $SendId + '&f=' + $FromId + '&c=' + txtA.value,
                    success: function(data) {
                        if (data == 1) {  //发送成功
                            showList($FromId, $SendId);
                            txtA.value = "";
                        }
                    }
                });
            });
            //获取即时更新的聊天数据
            function showList(f, s) {
                $.ajax({
                    url: 'Xml/Chat.xml',
                    dataType: 'xml',
                    success: function(data) {
                        var $strUser = $(data).find("message");
                        var strHTML = ""; //初始化保存内容变量
                        $strUser.each(function() {
                            //过滤用户数据
                            if (($(this).attr("fId") == f &&
                                 $(this).attr("sId") == s) ||
                                ($(this).attr("fId") == s &&
                                 $(this).attr("sId") == f)) {
                                var strNav = $(this).attr("fId");
                                if (strNav == f) {
                                strNav = "我说"; }
                                strHTML += '<div class=\"nav\">
                                   <span>(' + strNav + ')<\/span>
                                   <span class=\"time\">' +
                                   $(this).children("datetime").text() + '<\/span><\/div>';
                                strHTML += '<div class=\"msg\">'+
                                   $(this).children("content").text() + '<\/div>';
                            }
                        });
                        //显示处理后的数据
                        $("#chatList").html(strHTML);
                    }
                });
            }
        });
    </script>
</head>
<body>
   <div class="iframe">
        <div class="title">荣拓即时聊天与
```

```
            (<span id="spnSendId" class="spnId"></span>)聊天中
            </div>
            <div class="content">
                <div id="chatList" class="lst"></div>
            </div>
            <div class="content">
                <textarea id="txtA" class="txt"></textarea>
                <input id="btnSend" type="button"
                       value="发送" class="btn" />
            </div>
        </div>
    </body>
    </html>
```

此外，本案例中新建了一个名为"Chat.ashx"的服务端程序，该程序的功能是根据页面传回的聊天信息，生成对应的XML格式数据并保存在服务端，实现的代码如下所示：

```
<%@ WebHandler Language="C#" Class="Chat" %>
using System;
using System.Web;
using System.Xml;
public class Chat : IHttpHandler {
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "text/plain";
        //定义变量接收页面传回的参数值
        string strContent = context.Request.QueryString["c"].ToString();
        string strFromId = context.Request.QueryString["f"].ToString();
        string strSendId = context.Request.QueryString["s"].ToString();
        //定义并初始化回调变量值
        int intSuccess = 0;
        //格式化当前的时间
        string strDate = DateTime.Now.ToString("HH:mm:ss");
        //定义一个XML文档变量
        XmlDocument xmlDoc = new XmlDocument();
        try
        {
            //打开指定的XML文件
            xmlDoc.Load(context.Server.MapPath("Xml/Chat.xml"));
            //查找到根节点元素
            XmlNode root = xmlDoc.SelectSingleNode("chat");
            //创建一个根节点下的子节点元素
            XmlElement xmlE = xmlDoc.CreateElement("message");
            //设置该元素的两个相关属性
            xmlE.SetAttribute("fId", strFromId);
            xmlE.SetAttribute("sId", strSendId);
            //创建一个子节点下的元素
            XmlElement xmlEd = xmlDoc.CreateElement("datetime");
            //设置节点文本
            xmlEd.InnerText = strDate;
            //追加至上一级节点中
            xmlE.AppendChild(xmlEd);
            //创建一个子节点下的元素
            XmlElement xmlEc = xmlDoc.CreateElement("content");
            //设置节点文本
            xmlEc.InnerText = strContent;


            //追加至上一级节点中
            xmlE.AppendChild(xmlEc);
            //将整个节点追加至根节点中
            root.AppendChild(xmlE);
            //保存创建好的XML文件
            xmlDoc.Save(context.Server.MapPath("Xml/Chat.xml"));
            //设置操作返回的值
            intSuccess = 1;
        }
        catch (Exception ex)
        {
            throw ex;
        }
        context.Response.Write(intSuccess);
    }
    public bool IsReusable {
        get {
            return false;
        }
    }
}
```

7.4.4　代码分析

在本示例中，为了实现即时聊天的功能，有两次与服务端进行交互的操作，第一次是隔时调用聊天数据，另外一次是当用户单击发送按钮时。

第一次的隔时调用聊天数据，通过setTimeout()方法，每隔3秒调用showList自定义函数实现的。在该函数中，通过调用jQuery框架中的$.ajax()方法，以XML方式向服务端请求一个数据，将对返回的数据进行格式设置和内容过滤，用于区分是对方还是自己应该显示的数据，最后将获取的过滤后的数据显示在页面的相应元素中。

用户在聊天文本框中输入内容后，单击“发送”按钮时，触发了与服务端的第二次数据交互，此次数据交互，同样是通过调用jQuery框架中的$.ajax()方法将聊天时的内容发送至服务端，服务端在接收传回的内容后，向XML文件中追加相应的记录。追加成功后，向页面返回一个为1的数值，页面根据返回的该值，再次调用自定义的showList()函数，用于重新获取并在页面中即时显示聊天数据，详细过程见代码中加粗部分。

说明　在进行聊天前，用户必须进行登录。成功登录后，在用户列表中选择一个聊天对象，用于获取用户本人和聊天对象的ID号，由

于篇幅有限，在此省略此功能的实现，有兴趣的读者可以继续完善此功能。

7.5　本章小结

本章由浅入深地介绍了在jQuery中访问JSON和XML两种格式数据的过程，包括对这两种格式数据的读取、解析、操作；最后，通过两个案例开发的完整分析，进一步夯实读者之前所掌握基础知识。

第8章　jQuery中的插件

本章内容

如何调用jQuery插件

jQuery常用插件

自定义jQuery插件

综合案例分析——使用uploadify插件实现文件上传功能

本章小结

虽然使用jQuery库可以满足绝大部分的应用需求，但是随着各种应用的层出不穷，在考虑主体程序库大小与通用性的前提下，如何丰富jQuery库中的功能，满足人们对一些特定应用的需求，无疑jQuery中的插件是解决这一问题的最好答案。

目前，有超过近百种的各类插件应用在全球的各种项目中，插件的使用充分展示了jQuery中又一核心功能——强大的易展性（Extension），下面就开始我们的插件之旅吧！

8.1　如何调用jQuery插件

插件是以jQuery的核心代码为基础，编写出符合一定规范的应用程序，并将程序进行打包，调用时，仅需要包含该打包后的JS文件即可。如需要使用表单插件，按下列步骤就可以实现插件的调用。

1）在页面中导入包含表单插件的JS文件，并确定它位于主jQuery库后，其代码如下：

```
<head>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
       <script type="text/javascript"
            src="Jscript/jquery.form.js">
    </script>
</head>
```

2）在JS文件或页面JS代码中，编写如下代码完成插件的调用：

```
$(function() {
     $("form").ajaxSubmit();
})
```

代码中.ajaxSubmit()就是调用插件中的方法，其作用就是获取表单中的字段值，向服务器发送请求。

最新插件可以从jQuery官方网站（http://plugins.jquery.com）获取，其页面如图8-1所示。

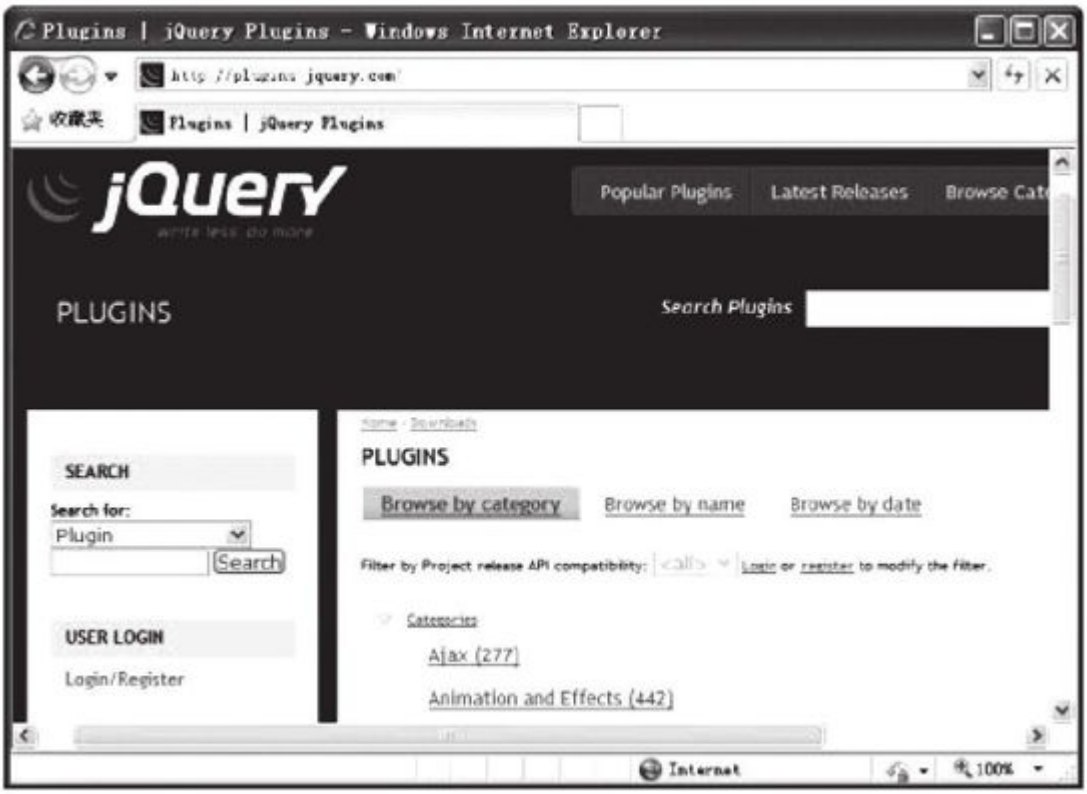


图　8-1　jQuery插件下载官网页面

8.2 jQuery常用插件

与浏览器插件不同，jQuery插件凭借其易加载、体积小、功能独立的特点深受广大Web开发人员的喜爱，接下来我们将以示例的方式，详细介绍目前jQuery框架中最为常用的各种功能性插件。

8.2.1 验证插件validate

validate是一个十分优秀的表单验证插件之一，它广泛地使用在全球各个的项目中，并得到广大程序开发人员的认可，该插件具有以下几个功能：

❑自带验证规则：其中包含必填、数字、URL等众多验证规则。

❑验证信息提示：可以使用默认的提示信息，也可以自定义提示信息，覆盖默认内容。

❑多种事件触发：不仅在表单提交时触发验证，在"keyup"或"blur"事件中也能触发。

❑允许自定义验证规则：除使用自带的验证规则外，开发者还可以很方便地自定义验证规则。

下面举例说明这个插件的使用方法。

示例8-1　validate插件的使用

（1）插件文件

Js-8-1/jquery.validate.js

Js-8-1/jquery.validate.messages_cn.js

（2）下载地址

http://plugins.jquery.com/project/validate

（3）功能描述

在页面中创建一个表单标记，ID号为"frmV"。在表单中设置两个文本框，一个用于输入“姓名”，另一个用于输入“邮箱”。单击“提交”按钮提交表单数据时，通过validate插件，根据设置的验证规则检测表单中的各个字段元素。

（4）实现代码

新建一个HTML文件8-1.html,加入如代码清单8-1所示的代码。

代码清单 8-1　使用 validate 插件验证表单提交时的基本信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>validate 验证插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-1/jquery.validate.js">
    </script>
```

```
    <script type="text/javascript"
            src="Js-8-1/jquery.validate.messages_cn.js">
    </script>
    <style type="text/css">
            ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#frmV").validate(
              {
                  /* 自定义验证规则 */
                  rules: {
                      username: { required: true, minlength: 6 },
                      email: { required: true, email: true }
                  },
                  /* 错误提示位置 */
                  errorPlacement: function(error, element) {
                      error.appendTo(element.siblings("span"));
                  }
              }
            );
        })
    </script>
</head>
<body>
     <form id="frmV" method="get" action="#">
     <div class="divFrame">
         <div class="divTitle">
              请输入下列资料
         </div>
         <div class="divContent">
              <div>
              用户名: <br />
              <input id="username" name="username"
                     type="text" class="txt" />
              <font color="red">*</font><br />
              <span></span>
              </div>
              <div>
              邮箱: <br />
              <input id="email" name="email"
                     type="text" class="txt" />
              <font color="red">*</font><br />
              <span></span>
              </div>
         </div>
         <div class="divBtn">
              <input id="sbtUser" type="submit"
                     value=" 提交 " class="btn" />
         </div>
     </div>
     </form>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-2所示。

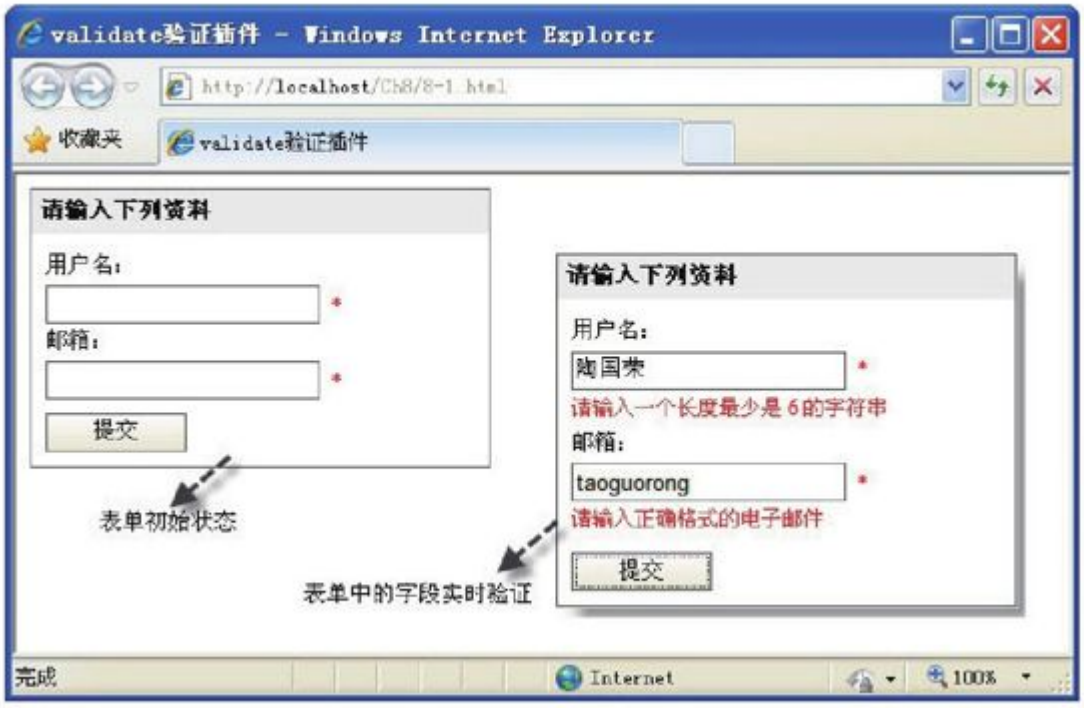


图　8-2　validate验证插件

（6）代码分析

由于validate验证插件默认的提示信息是英文版的，为了汉化验证提示信息，页面部分需要引用一个中文验证信息库，其代码如下：

```
<script type="text/javascript"
        src="Js-8-1/jquery.validate.messages_cn.js">
</script>
```

导入该中文验证信息库后，就可以将显示的验证提示信息汉化。

验证插件validate导入完成后，如果要使表单在提交数据时触发验证，只需加入如下代码：

```
$(function() {
    $("#frmV").validate(
    {
    ...// 表单验证时执行的代码
      }
    );
 })
```

在本示例中，由于进行的是简单的“用户名”与“邮箱”格式的验证，因此，只要在validate()方法中增加一个规则（rules）属性，用表单字段的"name"属性与规则进行关联，声明默认的验证规则，其实现的代码如下所示：

```
/* 自定义验证规则 */
rules: {
    username: { required: true, minlength: 6 },
    email: { required: true, email: true }
}
```

为了使验证后的提示信息显示在页面指定的位置中，在validate()方法中新增加一个提示信息位置属性。该属性执行一个回调函数，传递错误提示信息与执行验证字段两个参数，把提示信息的内容通过appendTo()方法增加到触发验证字段的“兄弟”元素＜span＞中，其实现的代码如下所示：

```
/* 错误提示位置 */
errorPlacement: function(error, element) {
  error.appendTo(element.siblings("span"));
}
```

8.2.2　表单插件form

form插件是专门为页面的表单而设计的，引入该插件后，通过调用ajaxForm()或ajaxSubmit()两个方法，可以很容易地实现Ajax方式提交数据，并通过方法中的options对象设置参数，获取服务器返回的数据。同时，该插件还包含如下一些重要方法。

❑formSerialize()：用于格式化表单中有用的数据，并将其自动整理成适合Ajax异步请求的URL地址格式。

❑clearForm()：清除表单中所有输入值的内容。

❑restForm()：重置表单中所有的字段内容，即将所有表单中的字段内容都恢复到页面加载时的默认值。

下面举例说明这个插件的使用方面。

示例8-2　form插件的使用

（1）插件文件

Js-8-2/jquery.form.js

（2）下载地址

http://plugins.jquery.com/project/form

（3）功能描述

在页面中创建一个ID号为"frmUserInfo"的表单，在表单中新建一个文本框，用于输入“用户名”。新建一个口令文本框，用于输入“密码”，单击“提交”按钮后，将向服务器文件Login.aspx发送请求，提交表单中的各元素值，服务器响应请求，将返回的内容显示在ID号为"divData"页面元素中。

（4）实现代码

新建一个HTML文件8-2.html,加入如代码清单8-2所示的代码。

代码清单 8-2　使用 form 插件实现表单数据的提交

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>form 表单插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
```

```
            src="Js-8-2/jquery.form.js">
    </script>
    <style type="text/css">
        ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
        // 定义一个表单提交时的对象
        var options = {
             // 默认为 form 中的 action，设置后便覆盖默认值
             url:"Login.aspx",
             // 将服务器返回的数据显示在 ID 号为 divData 元素中
             target: "#divData"
            }
            // 以 Ajax 的方式提交表单
            $("#frmUserInfo").ajaxForm(options);
        })
    </script>
</head>
<body>
     <form id="frmUserInfo" method="get" action="#">
     <div class="divFrame">
         <div class="divTitle">
              用户登录
         </div>
         <div class="divContent">
              <div>
              用户名：<br />
              <input id="username" name="username"
                     type="text" class="txt" />
              </div>
              <div>
              密码：<br />
              <input id="userpass" name="userpass"
                     type="password" class="txt" />
              </div>
         </div>
         <div class="divBtn">
              <input id="sbtUser" type="submit"
                     value="提交" class="btn" />
         </div>
         <div id="divData"></div>
     </div>
     </form>
</body>
</html>
```

服务器端文件Login.aspx的代码如下所示：

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%
    string strName = Request["username"]; //姓名字符
    string strPass = Request["userpass"]; //密码字符
    string strRetValue = "用户名：" + strName + "<br>密码：" + strPass;
    Response.Write(strRetValue);
%>
```

（5）页面效果

代码执行后的效果如图8-3所示。

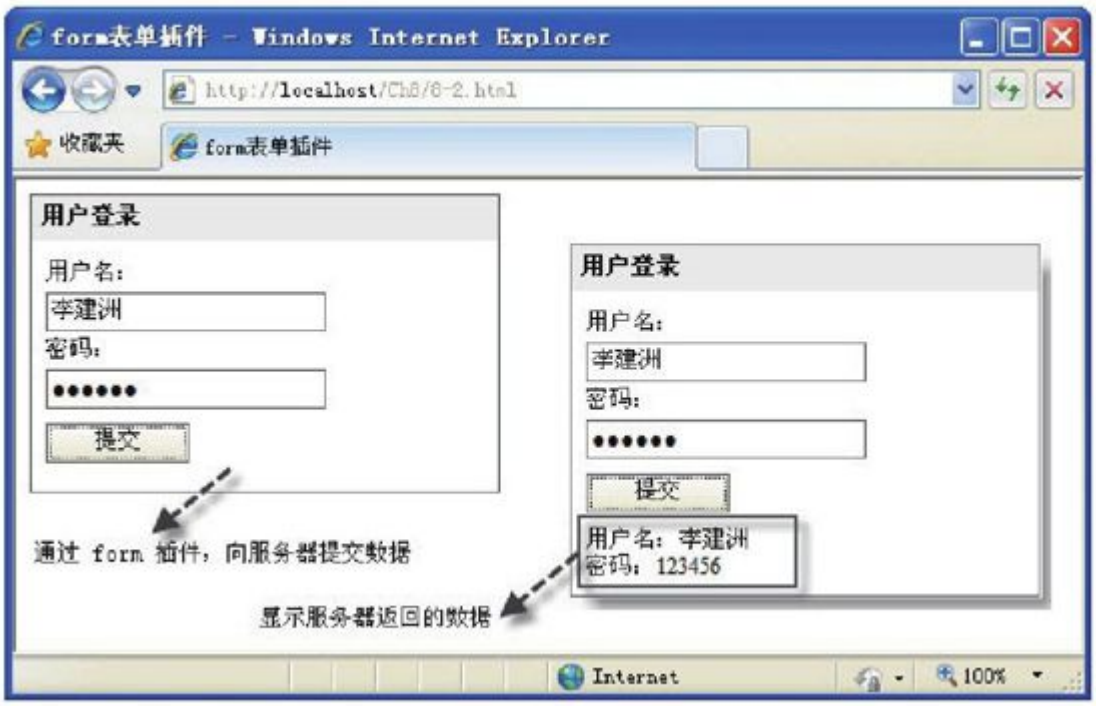


图　8-3　form表单插件

（6）代码分析

在导入form表单插件后，提交数据变得十分轻松，仅仅一行代码：

```
$("#frmUserInfo").ajaxForm();
```

这行代码等于如下代码：

```
$("#frmUserInfo").submit(function(){
    $("#frmUserInfo").ajaxSubmit();
    return false;
})
```

表单插件form的ajaxForm()与ajaxSubmit()方法中，可以没有参数，也可以传递1个参数；该参数既可以是一个回调型函数，也可以是一个options对象，通过该对象可以实现更多的页面互动功能。该对象常用的属性如下所示：

```
var options={
    url:url,                    //form 提交数据的地址（url）
    type:type,                  //form 提交的方式（method）
    target:target,              // 显示服务器返回数据的元素 ID 号
    beforeSubmit:function(),    // 提交前执行的回调函数
    success:function(),         // 提交成功后执行的回调函数
    dataType:null,              // 服务器返回数据类型
    clearForm:true,             // 提交成功后，清空表单中的字段值
    restForm:true,              // 提交成功后，重置表单中的字段值
    timeout:6000                // 设置请求时间，超过该时间后，自动退出请求（单位：毫秒）
}
```

8.2.3　Cookie插件cookie

在jQuery中，引入cookie插件后，可以很方便地定义某个Cookie名称，并设置Cookie值。通过设置好的Cookie，可以很便利地保存用户的页面浏览记录。在用户选择保存的情况下，还可以保存用户的登录信息。下面举例说明。

示例8-3　cookie插件的使用

（1）插件文件

Js-8-3/jquery.cookie.js

（2）下载地址

http://plugins.jquery.com/project/cookie

（3）功能描述

在示例8-2的页面中，再增加一个是否保存用户名的复选框，当选中复选框时，并单击“提交”按钮，完成Cookie的设置。重新打开浏览器时，由于使用了Cookie保存用户名，在用户名文本框中显示上次登录的用户名；如果输入用户名后不选中复选框，单击“提交”按钮将销毁已存在的Cookie对象。

（4）实现代码

新建一个HTML文件8-3.html,加入如代码清单8-3所示的代码。

代码清单 8-3　使用 cookie 插件保存表单中的用户名

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>cookie 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-3/jquery.cookie.js">
    </script>
    <stylе type="text/css">
           ... 省略样式部分代码
    </stylе>
    <script type="text/javascript">
        $(function() {
            if ($.cookie("UserName")) { // 如果有值，则显示在文本框中
                $("#UserName").val($.cookie("UserName"));
            }
            $("#sbtUser").submit(function() { // 表单提交事件
               // 如果选中了保存 " 用户名 " 项
               if ($("#chkSave").attr("checked")) { // 设置 Cookie 值
                    $.cookie("UserName", $("#UserName").val(), {
                        path: "/", expires: 7
                    })
                }
                else { // 销毁 Cookie 值
                    $.cookie("UserName", null, {
                        path: "/"
                    })
                }
```

```
                return false; //表单不提交
            })
        })
    </script>
</head>
<body>
    <form id="frmUserInfo" method="get" action="#">
    <div class="divFrame">
        <div class="divTitle">
            用户登录
        </div>
        <div class="divContent">
            <div>
            用户名：<br />
            <input id="UserName" name="UserName"
                type="text" class="txt" />
            </div>
            <div>
            密码：<br />
            <input id="UserPass" name="UserPass"
                type="password" class="txt" />
            </div>
            <div><input id="chkSave" type="checkbox" />
                是否保存用户名
            </div>
        </div>
        <div class="divBtn">
            <input id="sbtUser" type="submit"
                value="提交" class="btn" />
        </div>
        <div id="divData"></div>
    </div>
    </form>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-4所示。

图　8-4　cookie插件

（6）代码分析

在导入cookie插件后，可以通过一个全局性的方法管理客户端的Cookie对象，格式如下：

```
$.cookie(名称, 值, [option])
```

如果是写入或设置Cookie值，其调用的格式如下：

```
$.cookie(cookieName,cookieValue)
```

其中，参数"cookieName"表示要设置的Cookie名称，"cookieValue"表示相对应的值。

如果是读取Cookie值，其调用的格式如下：

```
$.cookie(cookieName)
```

其中，参数"cookieName"表示要读取的Cookie名称。

如果是销毁Cookie值，调用的格式如下：

```
$.cookie(cookieName, NULL)
```

其中，参数"cookieName"表示要销毁的Cookie名称。在销毁的Cookie时，其可选项参数中的路径（path）和域名（domain）必须与设置时一样，否则不能销毁。

另外，在$.cookie方法中，可选项参数[option]以对象的形式展示，用于补充说明设置的cookie对象，其常用的属性如下：

```
$.cookie(cookieName,cookieValue, {
   expires: //有限日期，可以是一个整数或一个日期（单位：天）
   path:    //Cookie值被保存的路径，默认值与创建页路径一致
   domin:   //Cookie域名属性，默认值与创建页域名一样
   secure:  //一个布尔值，表示传输Cookie值时，是否需要一个安全协议
})
```

8.2.4　搜索插件AutoComplete

AutoComplete为自动填充、展示之意。jQuery中引入该插件后，用户在使用文本框搜索信息时，使用插件中的autocomplete方法绑定文本框，当在文本框中输入某个字符时，通过该方法中的指定的数据URL，返回相匹配的数据，自动显示在文本框下，提醒用户进行选择。下面举例说明。

示例8-4　AutoComplete插件的使用

（1）插件文件

Js-8-4/jquery.autocomplete.js

Css-8-4/jquery.autocomplete.css

Images-8-4/indicator.gif

（2）下载地址

http://jquery.bassistance.de/autocomplete/jquery.autocomplete.zip

（3）功能描述

在页面中创建一个文本框，用于输入查询信息。当在文本框中输入字符时，通过AutoComplete插件绑定设置好的数组信息，并与文本框中的字符匹配，将匹配后的结果序列化后展示在文本框的底部，为提供用户选择。

（4）实现代码

新建一个HTML文件8-4.html,加入如代码清单8-4所示的代码。

代码清单 8-4　AutoComplete 插件提示用户搜索信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>autocomplete 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-4/jquery.autocomplete.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-8-4/jquery.autocomplete.css" />
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            var arrUserName = ["张三", "王小五", "张才子",
                "李四", "张大三", "李大四", "王五", "刘明",
                "李小四", "刘促明", "李逼", "张小三", "王小明"];
            $("#txtSearch").autocomplete(arrUserName, {
                minChars: 0 // 双击空白文本框时显示全部提示数据
            });
        })
    </script>
</head>
<body>
    <div class="divFrame">
         <div class="divTitle">
              搜索用户
         </div>
         <div class="divContent">
              <span style="padding:0 5px 0 10px">
                        <a href="#">新闻 </a></span>
              <span style="padding:0 5px 0 5px">
                        <b>用户 </b></span>
              <div>
                  <input type="text" id="txtSearch" class="txt" />
                  <input type="button" id="btnSearch"
                         value="查一下" class="btn" />
              </div>
         </div>
     </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-5所示。

图　8-5　AutoComplete插件提示搜索数据

（6）代码分析

在页面的JS代码中，由于导入了AutoComplete插件，代码十分简单。首先自定义一个数组，用于与文本框中的数据相匹配，其代码如下：

```
var arrUserName = ["张三", "王小五", "张才子",
    "李四", "张大三", "李大四", "王五", "刘明",
    "李小四", "刘促明", "李渊", "张小三", "王小明"];
```

在实际的开发中，这部分数据可以与某个页面的URL绑定获取，同样也可以实现提示效果。在设置完数据源后，调用插件的autocomplete方法，完成数据与文本框间的绑定。其代码如下：

```
$("#txtSearch").autocomplete(arrUserName, {
    minChars: 0 // 双击空白文本框时显示全部提示数据
});
```

AutoComplete插件的重要方法是autocomplete，其调用的语法格式如下：

```
$(文本框元素).autocomplete(urlOrData, [option])
```

其中，参数urlOrData表示绑定数据的路径或名称，可选项参数[option]为一个对象，其所设配置的属性名称如下所示：

```
$(文本框元素).autocomplete(urlOrData,
  {
   minChars:      // 自动完成前填入的最小字符，如果为0，双击空白文本时显示全部的提示数据信息
   max:           // 显示的提示数据总条数
   autoFill:      // 布尔值，表示是否选中时自动填充至文本框
   mustMatch:     // 布尔值，表示是否匹配数据，设为True时，文本框中内容必须
                  // 是匹配数据，如果不是，则清空文本框
   matchContains:// 布尔值，表示是否包含匹配数据，设置为True时，文本
                  // 框中内容只要被匹配数据包含则显示提示数据，否则不显示
   scrollHeight: // 设置匹配数据滚动显示的高度
   formatItem: function(data, i, total) {
         return "<I>"+data[0]+"</I>";
  }, // 格式化匹配数据显示的格式，如使用<I>标记
  formatMatch: function(data, i, total) {
         return data[0];

    },  // 未格式化的源匹配数据
    formatResult: function(data) {
            return data[0];
    }  // 返回最终匹配的结果
  })
```

在示例8-4中，我们对JS部分的代码再次进行修改，增加两个功能：一是改变匹配数据显示的格式，另一个功能是单击“查一下”按钮后，显示所选中的匹配结果。

```
... 省略部分代码
$("#txtSearch").autocomplete(arrUserName, {
    minChars: 0, // 双击空白文本框时显示全部提示数据
    formatItem: function(data, i, total) {
       return "<I>" + data[0] + "</I>"; // 改变匹配数据显示的格式
    },
    formatMatch: function(data, i, total) {
       return data[0];
    },
    formatResult: function(data) {
       return data[0];
    }
  }).result(SearchCallback);// 选中匹配数据中的某项数据时，调用插件的 result 方法
 // 自定义返回匹配结果函数
 function SearchCallback(event, data, formatted) {
   $("#divData").html(" 您的选择是: " + (!data ? " 空 " : formatted));
 }
// 单击 " 查一下 " 按钮后，触发插件的 search() 方法
$("#btnSearch").click(function() {
    $("#txtSearch").search();
     });
... 省略部分代码
```

通过修改JS代码后，执行的效果如图8-6所示。

图　8-6　调用AutoComplete插件中的方法

说明　AutoComplete插件中有两个重要的方法，一个是result(handler)方法，用于响应查看匹配的结果，其中参数handler为自定义的函数；另一个是search()方法，一般与“查询”按钮关联，调用绑定的result(handler)方法。

8.2.5　图片灯箱插件NotesForLightBox

NotesForLightBox是一个基于jQuery基础开发的图片放大浏览插件，它支持绝大部分浏览器，广泛应用在图片查看的项目中。该插件有以下几个强大的功能：

❑圆角的方式展示展示选中的图片。

❑按钮式查看“上一张”或“下一张”图片。

❑加载图片时带有进度条，显示加载进度。

❑可以采用自动播放的方式浏览图片。

❑多个样式属性可以随意设置。

下面通过示例来演示其功能。

示例8-5　NotesForLightBox插件的使用

（1）插件文件

Js-8-5/jquery.notesforlightbox.js

Css-8-5/jquery.notesforlightbox.css

Images-8-5/全部图片

Pic-8-5/全部图片

（2）下载地址

http://www.notesfor.net/file.axd?file=NFLightBox.zip

（3）功能描述

在页面中，以列表的方式展示某个相册中的全部图片，当用户单击其中某张图片时，通过引入的NotesForLightBox插件采用“灯箱”方式显示所选中的图片。

（4）实现代码

新建一个HTML文件8-5.html,加入如代码清单8-5所示的代码。

代码清单 8-5　NotesForLightBox 插件浏览相册中的图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>notesforlightbox 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-5/jquery.notesforlightbox.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-8-5/jquery.notesforlightbox.css" />
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
       $(function() {
```

```
            $('.divPics a').lightBox({
                overlayBgColor: "#666",    // 浏览图片时的背景色
                overlayOpacity: 0.5,       // 背景色的透明度
                containerResizeSpeed: 600 // 图片切换时的速度
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            我的相册
        </div>
        <div class="divContent">
            <div class="divPics">
                <ul>
                    <li><a href="Pic-8-5/img01.jpg"
                            title="第1篇风景图片">
                        <img src="Pic-8-5/img01.jpg" alt="" />
                    </a></li>
                    <li><a href="Pic-8-5/img02.jpg"
                            title="第2篇风景图片">
                        <img src="Pic-8-5/img02.jpg" alt="" />
                    </a></li>
                    <li><a href="Pic-8-5/img03.jpg"
                            title="第3篇风景图片">
                        <img src="Pic-8-5/img03.jpg" alt="" />
                    </a></li>
                    <li><a href="Pic-8-5/img04.jpg"
                            title="第4篇风景图片">
                        <img src="Pic-8-5/img04.jpg" alt="" />
                    </a></li>
                    <li><a href="Pic-8-5/img05.jpg"
                            title="第5篇风景图片">
                        <img src="Pic-8-5/img05.jpg" alt="" />
                    </a></li>
                    <li><a href="Pic-8-5/img06.jpg"
                            title="第6篇风景图片">
                        <img src="Pic-8-5/img06.jpg" alt="" />
                    </a></li>
                </ul>
            </div>
        </div>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-7所示。

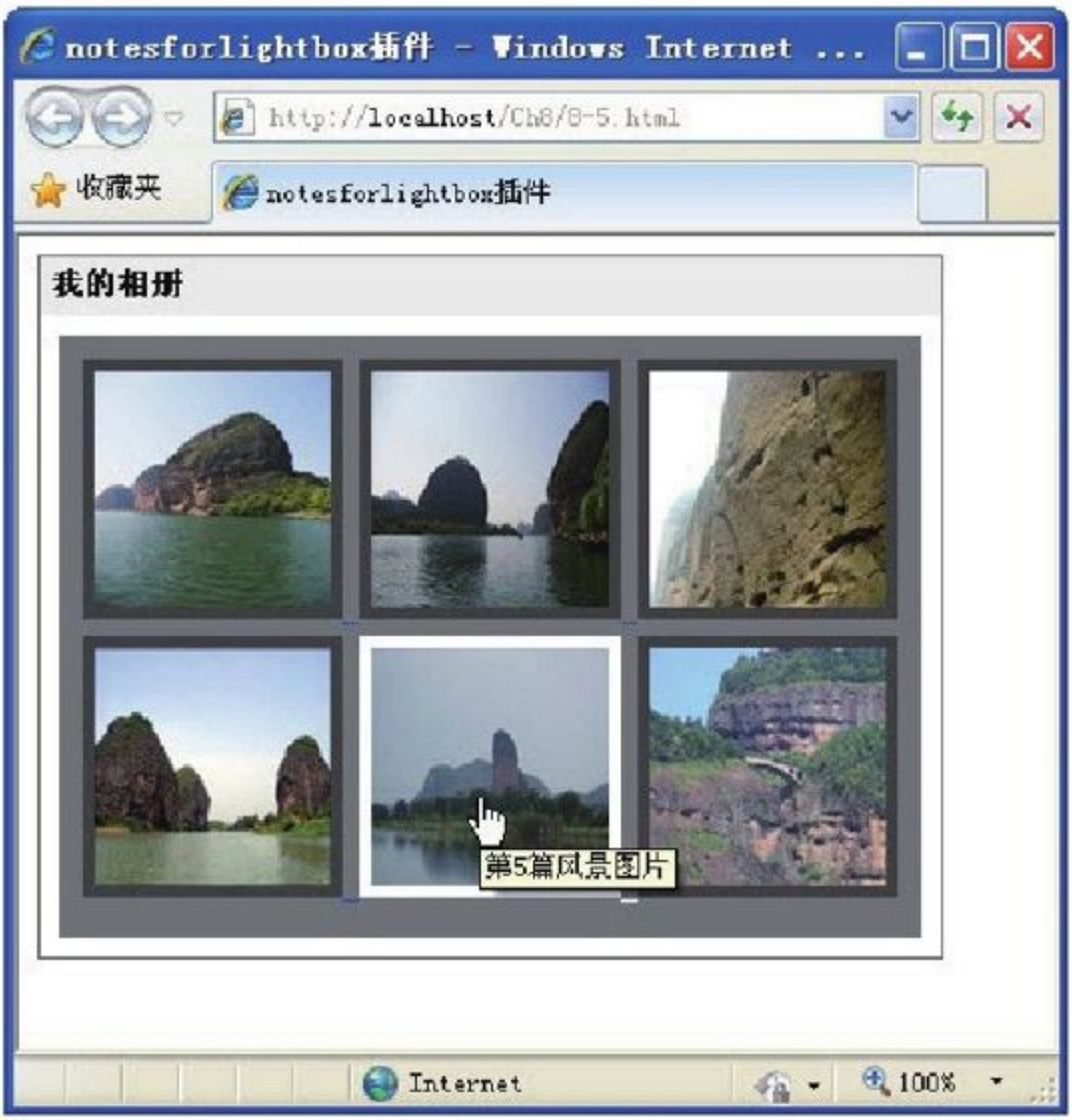


图　8-7　在相册中选中某张照片

在相册中，用户单击某张照片后，通过NotesForLightBox插件以“灯箱”的方式浏览该图片，其效果如图8-8所示。

图　8-8　浏览选中的图片

（6）代码分析

为了更好地展现NotesForLightBox插件浏览图片的效果，除了在页面中引入JS插件库文件外，还应包含一个该插件所固有的CSS文件jquery.notesforlightbox.css，其代码如下：

```
<link rel="stylesheet" type="text/css"
      href="Css-8-5/jquery.notesforlightbox.css" />
```

当然，用户可以对该CSS文件进行修改，以实现更好的页面效果。同时，该插件的一些功能性图片，如“上一张”、“下一张”图片默认是保存在文件夹Image中，如果其保存地址发生了变化，应打开JS插件文件jquery.notesforlightbox.js找到下列代码，进行设置：

```
...
// Configuration related to images
imageLoading: 'Images-8-5/loading.gif',
imageBtnPrev: 'Images-8-5/prev.png',
imageBtnNext: 'Images-8-5/next.png',
imageBtnClose: 'Images-8-5/close.png',
imageBlank: 'Images-8-5/lightbox-blank.gif',
imageBtnBottomPrev: 'Images-8-5/btm_prev.gif',
imageBtnBottomNext: 'Images-8-5/btm_next.gif',
imageBtnPlay: 'Images-8-5/start.png',
imageBtnStop: 'Images-8-5/pause.png',
...
```

全部设置完成以后，接下来的工作就是将页面中的图片与插件相关联，首先找到页面中的图片，然后通过插件中的lightBox()方法，绑定页面中找到的图片，其实现的代码如下：

```
$('.divPics a').lightBox();
```

lightBox()方法中有一个可选项参数[option]，用于设置在浏览图片时的各种页面属性，其常用的各种属性如下所示：

```
$('.divPics a').lightBox({
      overlayBgColor:      //设置一个字符串，表示图片浏览时页面的背景色
      overlayOpacity:      //设置一个数字，表示背景色的透明度，范围（0.0-0.9）
      fixedNavigation:     //设置一个布尔值，表示是否隐藏图片中的导航按钮
                           //默认为true，表示只有鼠标移动时才出现，否则隐藏
      containerBorderSize: //设置一个数字，表示图片的内容边框宽度

      containerResizeSpeed: //设置一个数字，表示图片间相互切换的速度，默认为500（单位:毫秒）
});
```

8.2.6　右键菜单插件ContextMenu

ContextMenu是一款轻量型、功能完善的插件，利用该插件可以在页面的任何位置，设置一个触发右键事件的元素。选中该元素并单击鼠标右键时，通过插件中的contextMenu方法弹出一个设计精美快捷菜单。该插件具有以下几个显著特点：

❑可以在同一个页面中设置多个不同样式的菜单。

❑一个菜单可以绑定页面中的多个元素。

❑可随意设置菜单样式。

❑轻松访问与绑定菜单中的各选项。

下面通过示例来演示其用法。

示例8-6　ContextMenu插件的使用

（1）插件文件

Js-8-6/jquery.contextmenu.js

Css-8-6/jquery.contextmenu.css

Images-8-6/全部图片

（2）下载地址

http://www.trendskitchens.co.nz/jquery/contextmenu/jquery.contextmenu.r2.js

（3）功能描述

在页面中创建一个文本编辑框，选中该文本框，单击鼠标右键，弹出一个设置好的快捷菜单，单击菜单选项时显示所选择的选项内容。

（4）实现代码

新建一个HTML文件8-6.html,加入如代码清单8-6所示的代码。

代码清单 8-6　ContextMenu 插件弹出快捷菜单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>ContextMenu 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-6/jquery.contextmenu.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-8-6/jquery.contextmenu.css" />
    <style type="text/css">
          ... 省略样式部分代码
```

```
    </style>
    <script type="text/javascript">
        $(function() {
            $('#txtContent').contextMenu('sysMenu',
              { bindings:
                {
                    'Li1': function(Item) {
                        alert("在ID号为:" + Item.id + "编辑框中，您点击了“新建”项");
                    },
                    'Li2': function(Item) {
                        alert("在ID号为:" + Item.id + "编辑框中，您点击了“打开”项");
                    }
                    ...//设置其他选项事件
                }
              });
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">点击右键</div>
        <div class="divContent">
            <textarea id="txtContent" cols="30" rows="5"></textarea>
        </div>
    </div>
    <div class="contextMenu" id="sysMenu">
      <ul>
        <li id="Li1"><img src="Images-8-6/new.png" /> 新建</li>
        <li id="Li2"><img src="Images-8-6/folder.png" /> 打开</li>
        <li id="Li3"><img src="Images-8-6/disk.png" /> 保存</li>
        <hr />
        <li id="Li4"><img src="Images-8-6/cross.png" /> 退出</li>
      </ul>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-9所示。

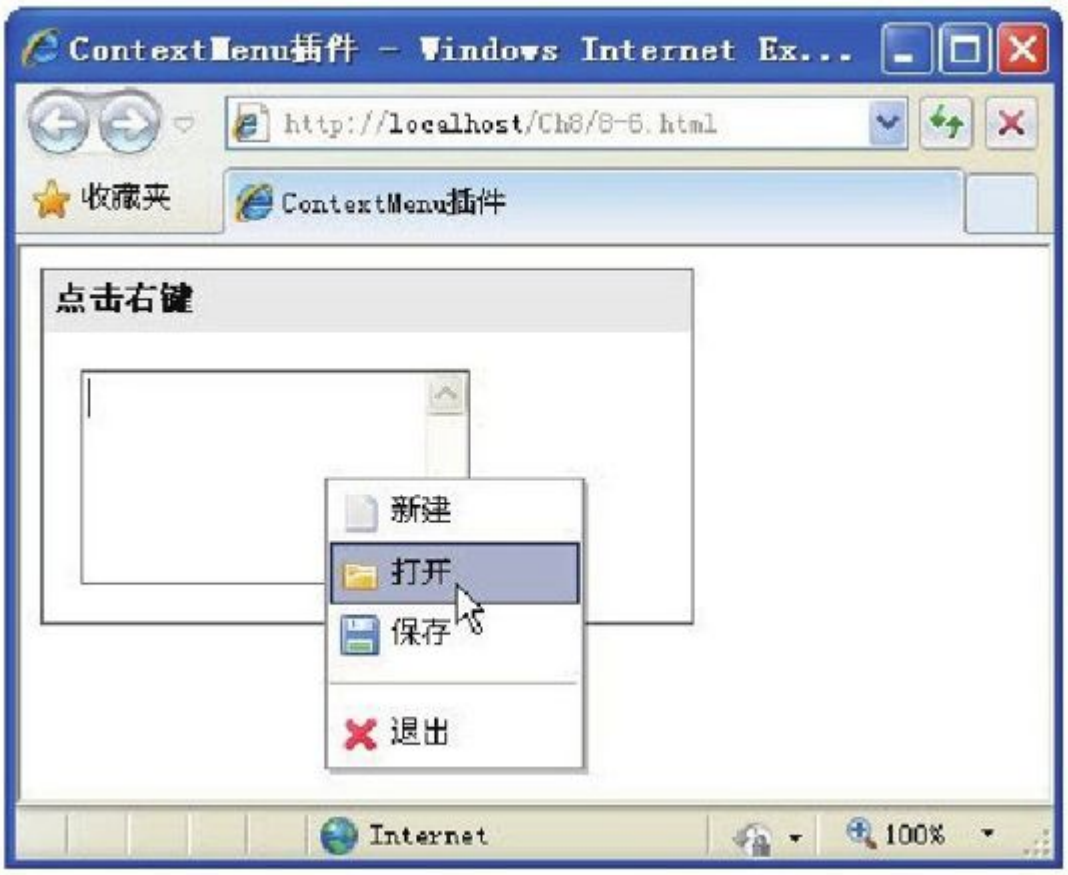


图　8-9　ContextMenu插件弹出快捷菜单

（6）代码分析

为了更好地实现页面效果，首先下载插件的JS文件，并重新命名为jquery.contextmenu.js，然后在页面的＜title＞标记中导入，其执行的代码如下：

```
<script type="text/javascript"
        src="Js-8-6/jquery.contextmenu.js">
</script>
```

在HTML页面中，调用插件的方法如下：

```
$(element).contextMenu(Mid [,Option])
```

其中，字符参数Mid表示与菜单绑定的元素ID号。可选项对象Option可以设置元素与菜单绑定时的属性或事件，其包含的属性和事件代码如下：

```
$(element).contextMenu(Mid ,
      {
        bindings{// 绑定菜单时，根据 ID 号获取或设置各选项属性和事件
           " 选项 ID 号 ", function(T){// 参数 T 为被绑定的元素
                ... // 获取或设置该选项的属性
              }
            }
          }
         )
```

在上述示例中，单击菜单中“打开”选项时将弹出一个对话框，显示绑定菜单元素的ID号和设置的选项内容，其代码如下：

```
bindings:
 {
   'Li1': function(Item) {
      alert(" 在 ID 号为: " + Item.id + " 编辑框中，您点击了“新建”项 ");
    },
   'Li2': function(Item) {
      alert(" 在 ID 号为: " + Item.id + " 编辑框中，您点击了“打开”项 ");
    }
   ... // 设置其他选项事件
 }
```

弹出的对话框如图8-10所示。

图　8-10　单击菜单中的“打开”选项时弹出的对话框

在可选项option参数中，除了设置绑定菜单的选项外，还可以设置菜单的默认样式，其样式可以设置如下三部分：

```
menuStyle: { // 菜单外框样式
      ...
}
itemStyle: { // 菜单选项样式
      ...
}
itemHoverStyle: { // 选项选中样式
}
```

在本示例中，为了改变弹出快捷菜单的样式加入如下的代码：

```
$(function() {
    $('#txtContent').contextMenu('sysMenu',
       { bindings:{
            ...
         },
```

```
        menuStyle:{
              border: '2px solid #999'
           },
        itemStyle:{
              fontFamily: 'verdana',
              backgroundColor: '#666',
              color: 'white',
              border: 'none',
              padding: '1px'
           },
        itemHoverStyle: {
              color: '#666',
              backgroundColor: '#f7f7f7',
              border: 'none'
           }
      });
})
```

弹出菜单样式改变后的页面效果如图8-11所示。

图　8-11　样式改变后的弹出菜单

为了使用读者对ContextMenu插件能有更进一步的了解，下面列出该插件在使用时所调用的全部属性和事件，其完整的调用格式如下所示：

```
$(element).contextMenu(Mid
    {
        bindings{// 绑定菜单时，根据 ID 号获取或设置各选项属性和事件
            "选项 ID 号", function(T){// 参数 T 为被绑定的元素
                    ...// 获取或设置该选项的属性
                }
        }
    },
menuStyle: { // 菜单外框样式
     ...
},
itemStyle: { // 菜单选项样式
     ...
},
itemHoverStyle: { // 选项选中样式
},    ...
Shadow:    // 设置一个布尔值，默认为 true，即显示背景阴影
eventPosY: // 指定菜单弹出时，在页面中的 Y 值
eventPosX: // 指定菜单弹出时，在页面中的 X 值
onContextMenu: function(e) {// 菜单内容显示事件
      // 其中通过回调函数中的参数 e，可以获取被绑定菜单的元素，即 e.target
      // 如果该函数返回 false，将不会弹出快捷菜单
},
onShowMenu: function(e, menu){// 菜单显示事件
     // 其中通过回调函数中的参数 e，可以获取被绑定菜单的元素，即 e.target
     // 参数 menu 是显示的菜单，通过该参数可以访问其中的某一个选项，如移除第 1 个选项，代码如下：
            $(' 第 1 个选项 ID 号 ', menu).remove();
          return menu;
}
}
```

8.2.7　图片放大镜插件jQZoom

jQZoom是一款基于jQuery库的图片放大插件，在页面中实现放大的方法是先准备两张一大一小的相同图片，在页面打开时展示小图片，当鼠标在小图片的任意位置移动时，调用插件中的jqzoom()方法绑定另外一张相同的大图片，在指定位置显示与小图片所选区域相同的大图片区域，从而实现逼真的放大效果。该插件非常适合在产品展示时使用，下面举例说明。

示例8-7　jQZoom插件的使用

（1）插件文件

Js-8-7/jquery.jqzoom.js

Css-8-7/jquery.jqzoom.css

Images-8-7/全部图片

（2）下载地址

http://www.mind-projects.it/projects/jqzoom/archives/jqzoom_ev1.0.1.zip

（3）功能描述

在页面中放置一张图片，当鼠标在图片中移动时，出现一块选择区域，在图片的右边显示放大后的所选区域。

（4）实现代码

新建一个HTML文件8-7.html,加入如代码清单8-7所示的代码。

代码清单 8-7　jQZoom 插件实现图片放大效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>JQZoom 放大镜插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-8-7/jquery.jqzoom.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-8-7/jquery.jqzoom.css" />
    <style type="text/css">
            ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#jqzoom").jqzoom( //绑定图片放大插件 jQZoom
                {
                    zoomWidth: 230,
                    zoomHeight: 230
                }
            );
        });
    </script>
</head>


<body>
    <div class="divFrame">
        <div class="divTitle">
            图片放大镜
        </div>
        <div class="divContent">
            <a href="Images-8-7/bag.jpg" id="jqzoom" title=" 我的背包 ">
                <img src="Images-8-7/bagsmall.jpg">
            </a>
        </div>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-12所示。

图　8-12　jQZoom插件实现图片的放大效果

（6）代码分析

在页面中，为了实现图片放大的效果，除引用插件的JS文件外，还必须导入与之匹配CSS文件，因此在页面的头文件中加入如下代码：

```
<script type="text/javascript" src="Js-8-7/jquery.jqzoom.js">
</script>
<link rel="stylesheet" type="text/css"
      href="Css-8-7/jquery.jqzoom.css" />
```

引用完JS和CSS文件后，接下来的工作就是将图片与插件绑定，为实现这一目的，首先在页面中增加一个＜img＞标记用于显示小图片，并将图片用一个＜a＞标记包裹起来；同时，将＜a＞标记的href属性设置为大图的URL，并设置title属性，其实现的代码如下：

```
<a href="Images-8-7/bag.jpg" id="jqzoom" title=" 我的背包 ">
          <img src="Images-8-7/bagsmall.jpg">
</a>
```

然后，在JS文件中通过插件的jqzoom()方法绑定对应的放大元素，其实现的代码如下：

```
$("#jqzoom").jqzoom( // 绑定图片放大插件 jQZoom
    {
      zoomWidth: 230, // 小图片中所选区域的宽
      zoomHeight: 230 // 小图片中所选区域的高
      }
);
```

在插件调用的jqzoom([Option])方法中，可选项参数Option是一个对象，除上述的zoomWidth与zoomHeight属性外，还有如下的常用属性：

```
$(element).jqzoom( // 绑定图片放大插件 jQZoom
   {
    zoomType:     // 放大镜类型，默认为 "standard"，如果设为 "reverse"，在小图
                  // 片中移动鼠标时，所选区域将高亮显示，非选中区域为淡灰色
        xOffset: // 放大后的图片与小图片间的 x( 横坐标 ) 距离
        yOffset: // 放大后的图片与小图片间的 y( 纵坐标 ) 距离
        position: // 放大后的图片相对原图片的位置，默认为 "right"，可设置为 "left" 、"top" 、"bottom"
    lens:         // 一个布尔值，表示是否显示小图片中的选择区域，默认值为 "true"
                  // 如果设为 "false"，则鼠标在小图片中移动时，不会出现选择区域
    title:        // 一个布尔值，是否显示放大图片的主题，默认值为 "true"
                  // 如果设为 "false"，则放大后的图片上面不会出现主题字样
    imageOpacity:// 当 zoomType 的值为 "reverse" 时，用来设置非选中区域透
                  // 明度的值，取值范围 (0.0-1.0)
   }
);
```

8.2.8　图片切换插件Nivo Slider

Nivo Slider是一款基于jQuery的多图片切换插件，它的体积非常小（打包后仅有十几KB），但功能强大，拥有多种图片切换时的动画效果，支持键盘导航和连接影像功能。此外，使用时代码简洁，图片中的标题支持HTML标记并兼容各类浏览器，是一款十分理想的图片切换插件，广泛应用于页面的推荐和广告图片切换。下面举例说明。

示例8-8　Nivo Slider插件的使用

（1）插件文件

Js-8-8/jquery.nivo.slider.js

Css-8-8/nivo-slider.css

Css-8-8/theme-default.css

Pic-8-8/全部图片

（2）下载地址

http://nivo.dev7studios.com

（3）功能描述

在页面中添加5张图片，通过图片切换插件Nivo Slider实现手动切换或自动播放的效果，同时，在切换图片的下方，单击缩略的小图片同样可以实现图片的浏览或切换效果。

（4）实现代码

新建一个HTML文件8-8.html,加入如代码清单8-8所示的代码。

代码清单 8-8　图片切换插件 Nivo Slider

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>图片切换插件Nivo Slider</title>
        <link href="Css-8-8/nivo-slider.css"
              rel="stylesheet" type="text/css" />
        <link href="Css-8-8/theme-default.css"
              rel="stylesheet" type="text/css" />
        <script type="text/javascript"
                src="Jscript/jquery-1.8.2.min.js">
        </script>
        <script type="text/javascript"
                src="Js-8-8/jquery.nivo.slider.js">
        </script>
        <script type="text/javascript">
            $(function() {
                 $('#slider').nivoSlider({
                 effect: 'boxRandom',           // 图片随机切换效果
                 beforeChange: function() { }, // 图片切换前回调函数
                 afterLoad: function() { },     // 图片加载后回调函数
                 controlNavThumbs: true,        // 使用缩略图控制导航
                 manualAdvance: true            // 需要手动切换
                 });
            });
        </script>
    </head>
    <body>
        <div class="theme-default">
            <div id="slider" class="nivoSlider">
                <a href="javascript:">
                   <img src="Pic-8-8/img01.jpg" alt="01"
                data-thumb="Pic-8-8/img01.jpg" title=" 第 1 幅图片 "/>
                 </a>
                 <a href="javascript:">
                   <img src="Pic-8-8/img02.jpg" alt="02"
                data-thumb="Pic-8-8/img02.jpg" title=" 第 2 幅图片 "/>
                 </a>
                 <a href="javascript:">
                   <img src="Pic-8-8/img03.jpg" alt="03"
                data-thumb="Pic-8-8/img03.jpg" title=" 第 3 幅图片 "/>
                 </a>
                 <a href="javascript:">
                   <img src="Pic-8-8/img04.jpg" alt="04"
                data-thumb="Pic-8-8/img04.jpg" title=" 第 4 幅图片 "/>
                 </a>
                 <a href="javascript:">
                   <img src="Pic-8-8/img05.jpg" alt="05"
                data-thumb="Pic-8-8/img05.jpg" title=" 第 5 幅图片 "/>
                 </a>
            </div>
        </div>
    </body>
    </html>
```

（5）页面效果

代码执行后的效果如图8-13所示。

图　8-13　图片切换插件Nivo Slider

（6）代码分析

在页面中，为了通过插件实现多张图片切换的效果，除在＜head＞元素中引用jQuery框架外，还需要导入插件的JS文件和与之对应的CSS文件。此外，还要导入一个控制图片导航键和缩略导航图片的样式文件"theme-default.css"，即在＜head＞元素中加入如下代码：

```
<link href="Css-8-8/nivo-slider.css"
        rel="stylesheet" type="text/css" />
<link href="Css-8-8/theme-default.css"
        rel="stylesheet" type="text/css" />
<script type="text/javascript"
        src="Jscript/jquery-1.8.2.min.js">
</script>
<script type="text/javascript"
        src="Js-8-8/jquery.nivo.slider.js">
</script>
```

接下来，在ID号为"slider"、类别名为"nivoSlider"的页面＜div＞元素中，创建＜img＞元素插入需要切换的图片。如果需要增加图片的超级链接，只要在＜img＞元素外围再创建一个＜a＞元素即可。此外，切换图片的标题内容，可以通过设置＜img＞元素的title属性值来实现，该属性值支持HTML格式，＜div＞元素中插入切换图片的代码如下：

```
<img src="Pic-8-8/img01.jpg" alt="01"
      data-thumb="Pic-8-8/img01.jpg" title="第1幅图片"/>
```

上述代码中，＜img＞元素的title属性表示切换图片的主题内容，data-thumb属性表示切换图片底部导航图的路径，通常与src属性值相同。完成导入与图片切换插件相关的JS框架和创建相应页面元素操作之后，就是编写JavaScript代码，调用JS插件中的方法，其调用格式如下：

```
$(图片集合外围元素).nivoSilder(options);
```

通过上述调用格式与页面中的切换图片元素相绑定，其实现的代码如下：

```
$('#slider').nivoSlider({
    effect: 'boxRandom',              //图片随机切换效果
    beforeChange: function() { },     //可编写图片切换之前回调函数
    afterLoad: function() { },        //可编写图片加载之后回调函数
    controlNavThumbs: true,           //使用缩略图控制导航
    manualAdvance: true               //需要手动切换
});
```

在上述代码中，ID号为"slider"的元素调用图片切换插件中的nivoSlider()方法。在调用时，设置了图片切换时的一些基本效果。如"effect"表示图片切换时的效果属性，该值为"boxRandom"表示随机效果；"controlNavThumbs"表示图片切换时是否要显示缩略图导航条，"true"表示需要显示，"false"表示不需要，默认值为"false"，即通过对应小圆圈作为图片切换的导航条，效果如图8-13左侧所示。

此外，在调用图片切换插件的nivoSlider()方法时，括号中的参数options是一个对象集合，它包含了多个属性和事件，如表8-1所示。

表 8-1　选项 options 中的常用参数

参数名称	功能描述
effect	图片切换效果，目前支持 11 种效果，如“sliceDownRight”、“sliceDownLeft”、“sliceUpRight”、“sliceUpLeft”、“sliceUpDown”、“sliceUpDownLeft”、“fold”、“fade”、“slideInRight”、“slideInLeft”、“boxRandom”默认值为 random
animSpeed	图片切换时的速度，默认单位是毫秒，默认值为 500
pauseTime	图片切换时停留的时间，默认单位是毫秒，默认值为 3000
startSlide	图片开始切换的位置，表示从第几张图片开始切换，默认值为 0
directionNav	是否在切换图片中使用左右按钮导航，默认值为 true
directionNavHide	当鼠标移动至切换图片中时，是否显示左右按钮导航，默认值为 true
pauseOnHover	当鼠标移动至切换图片中时，是否停止切换效果，默认值为 true
controlNav	在切换图片的底部，是否显示图片导航条，默认值为 true
controlNavThumbs	在切换图片的底部，是否显示带缩略图片的导航条，默认值为 false
controlNavThumbsFromRel	是否使用切换图片的 rel 属性关联缩略图，默认值为 false
controlNavThumbsSearch	导航条中缩略图片的类型，默认值为 “.jpg”
controlNavThumbsReplace	导航条中缩略图片的扩展名格式，默认值为 “_thumb.jpg”
keyboardNav	是否支持键盘中的方向键实现图片切换功能，默认值为 true
manualAdvance	是否手动切换图片，默认值为 false
captionOpacity	设置图片标题内容的背景透明度，默认值为 0.8
beforeChange	回调函数，当发生切换之前触发
afterChange	回调函数，当发生切换之后触发
slideshowEnd	回调函数，完全所有图片切换之后触发
lastSlide	回调函数，切换最后一张图片时触发
afterLoad	回调函数，当整个切换图片加载完成后触发

8.2.9　动画表格排序插件TableSort

TableSort插件是专门针对表格＜table＞中的各列＜td＞以动画效果进行排序，排序形式包括正则匹配、按字母升降、ASCII或数值，无论选择哪种排序形式，在表格中，都以动画的形式展示排序切换的过程，效果非常好。

示例8-9　TableSort插件的使用

（1）插件文件

Js-8-9/jquery.tableSort.js

（2）下载地址

http://www.mitya.co.uk/scripts/Animated-table-sort-REGEXP-friendly-111

（3）功能描述

在页面中创建一个＜table＞表格，第一行为列标题，分别对应“学号”、“姓名”、“性别”、“总分”等5个列标题名。用户分别单击这5个列标题名时，对应该列的数据将以动画的效果进行排序，在排序时，如果已经是降序，单击后将按照升序排列；如果已经是升

序，单击后将按照降序排列，同时，在表格的上方显示当前排序的列标题名称和排序方式。

（4）实现代码

新建一个HTML文件8-9.html,加入如代码清单8-9所示的代码。

代码清单 8-9　动画表格排序插件 TableSort

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>动画表格排序插件TableSort </title>
        <script src="Jscript/jquery-1.8.2.min.js"
                type="text/javascript"></script>
        <script src="Js-8-9/jquery.tableSort.js"
                type="text/javascript"></script>
        <link href="Css-8-9/8-9.css" rel="stylesheet"
                type="text/css" />
        <script type="text/javascript">
            $(function() {
                var $tb = $("#tbStudent");
                var $tip = $("#tip");
                var $bln = true;
                var $str = "降";
                //遍历table标题中的a元素
                $(".table tr th a").each(function(i) {
                    $(this).bind("click", function() {
                        $bln = $bln ? false : true;
                        $str = $bln ? "降" : "升";
                        $tip.show().html("当前按 <b>" +
                        $(this).html() + $str + "序</b> 排列");
                        $tb.sortTable({
                            onCol: i + 1,
                            keepRelationships: true,
```

```
                    sortDesc: $bln
                });
            });
        });
    });
    </script>
</head>
<body>
<div id="tip"></div>
<table id="tbStudent" class="table">
    <tr>
        <th><a href="javascript:">编号 </a></th>
        <th><a href="javascript:">姓名 </a></th>
        <th><a href="javascript:">性别 </a></th>
        <th><a href="javascript:">总分 </a></th>
    </tr>
    <tr>
        <td>1031</td>
        <td> 李渊 </td>
        <td> 男 </td>
        <td>654</td>
    </tr>
    <tr>
        <td>1021</td>
        <td> 张扬 </td>
        <td> 男 </td>
        <td>624</td>
    </tr>
    <tr>
        <td>1011</td>
        <td> 吴敏 </td>
        <td> 女 </td>
        <td>632</td>
    </tr>
    <tr>
        <td>1026</td>
        <td> 李小明 </td>
        <td> 男 </td>
        <td>610</td>
    </tr>
    <tr>
        <td>1016</td>
        <td> 周谨 </td>
        <td> 女 </td>
        <td>690</td>
    </tr>
    <tr>
        <td>1041</td>
        <td> 谢小敏 </td>
        <td> 女 </td>
        <td>632</td>
    </tr>
    <tr>
        <td>1072</td>
        <td> 刘明明 </td>
```

```
            <td> 男 </td>
            <td>633</td>
        </tr>
        <tr>
            <td>1063</td>
            <td> 蒋忠公 </td>
            <td> 男 </td>
            <td>675</td>
        </tr>
    </table>
    </body>
    </html>
```

（5）页面效果

代码执行后的效果如图8-14所示。

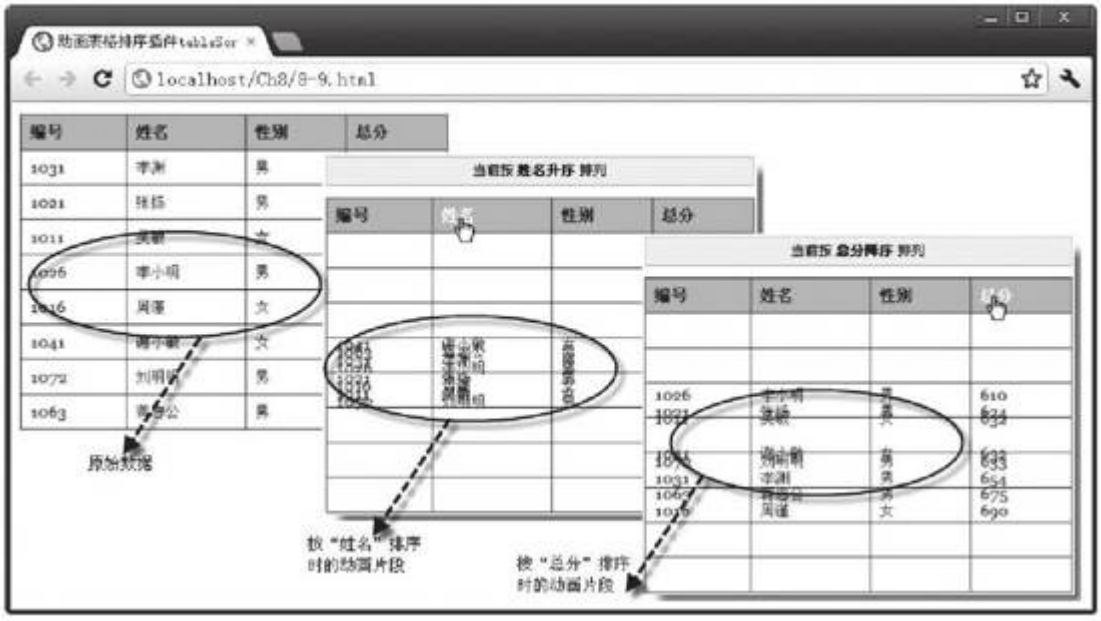


图　8-14　动画表格排序插件TableSort

（6）代码分析

在本示例中，先在＜head＞元素中导入插件对应的JS文件，并在页面中添加＜table＞表格元素，接下来编写JavaScript代码，实现用

户单击列标题内容时以动画效果实现排序的功能，实现的代码如下所示：

```
var $tb = $("#tbStudent");
var $tip = $("#tip");
var $bln = true;
var $str = "降";
//遍历table标题中的a元素
$(".table tr th a").each(function(i) {
     $(this).bind("click", function() {
          $bln = $bln ? false : true;
          $str = $bln ? "降" : "升";
          $tip.show().html("当前按 <b>" +
          $(this).html() + $str + "序</b> 排列");
          $tb.sortTable({
               onCol: i + 1,

                keepRelationships: true,
                sortDesc: $bln
            });
       });
});
```

在上述代码中，前4行定义变量保存页面中的元素和初始化显示值及排序方式变量。接下来，通过遍历＜table＞表格元素标题中所有＜a＞元素的方式，设置单击列标题的click事件。在该事件中，先改变变量$bln的值，如果该值为true，则改为false，否则取当前值的反值。然后，根据该值获取$str的内容，并将该内容显示在页面指定ID号的元素中。

最后，通过页面中的＜table＞表格元素调用插件的sortTable()方法，实现表格中对应列数据的排序功能。在该方法中，"onCol"属性

表示排序列的索引号，该值初始值为0。"keepRelationships"属性表示在选中列排序时，除该列之外的其余列的顺序是否要也要随之改变；"sortDesc"属性表示排序的方式，如果为true，表示按降序排列，否则按升序排列。

在插件的sortTable([options])方法中，可选项参数options除在示例中使用到的各属性之外，还有如下常用属性：

```
$(表格元素).sortTable({
"regexp":        //用于排序的正则表达式，其中用于排序部分由"regexpIndex"属性决定，默认值为null
"regexpIndex": //决定用于排序正则表达式的索引号，默认值为0
"child":         //用于排序列的子元素内容，即按子元素中<td>内容排序，而不是父元素中的<td>内容
"sortType":     //排序时的类型，如"numeric"表示按照数字类型排序，默认按"ascii"类型排序
});
```

8.2.10　进度条插件ProgressBar

ProgressBar是一款基于jQuery开发的进度条插件，它的优点是：使用代码简洁，可轻松定制它的外观；当页面请求耗时较长，需要用户等待时使用，如文件上传和大图片、多数据加载时，可添加该插件，以增加用户的体验。

ProgressBar插件在images文件夹中对应6张图片，其中1张是背景色图片，另外5张为自定义的5种进度条颜色图片。这5张图片都由长度相等的两个部分组成，前部分用于显示所完成的进度值，后部分用于显示余下的进度值，通过不同的颜色进行区分。

示例8-10　ProgressBar插件的使用

（1）插件文件

Js-8-10/jquery.progressbar.js

（2）下载地址

http://t.wits.sg/jquery-progress-bar/

（3）功能描述

在页面中，增加一个ID号为"pb"的＜p＞元素，编写JavaScript代码将该元素与进度条插件相绑定，分别显示使用不同参数调用时的页面效果。

（4）实现代码

新建一个HTML文件8-10.html,加入如代码清单8-10所示的代码。

代码清单 8-10　进度条插件 ProgressBar

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>进度条插件 ProgressBar</title>
        <script src="Jscript/jquery-1.8.2.min.js"
                type="text/javascript"></script>
        <script src="Js-8-10/jquery.progressbar.js"
                type="text/javascript"></script>
        <style type="text/css">
               body{font-size:13px}
               h3,p{padding:0px;margin:0px}
        </style>
        <script type="text/javascript">
            $(function() {
                $("#pb").progressBar(); //绑定初始化
                /*$("#pb").progressBar(68, {
                    barImage: 'images/progressbg_orange.gif',
                    showText: false
                }); //改变进度条显示的图片颜色 */
                /*$("#pb").progressBar(68, {
                    max: '100',
                    textFormat: 'fraction',
                    callback: function(data) {
                        if (data.running_value == data.value) {
                            alert("上传成功!");
                        }
                    }
                });// 调用回调函数 */
                /*$("#pb").progressBar(68, {
                    barImage: 'images/progressbg_yellow.gif',
                    width: 120,
                    height: 12
                });// 自定义进度条的宽与高 */
            });
        </script>
    </head>
    <body>
       <div>
            <h3>进度条: </h3>
            <p id="pb">100</p>
       </div>
    </body>
    </html>
```

（5）页面效果

代码执行后的效果如图8-15所示。

图　8-15　进度条插件ProgressBar

（6）代码分析

在本示例中，页面中的元素与进度条插件绑定的代码格式如下：

```
$(页面元素).progressBar(persent, config)
```

其中，$(页面元素)表示需要被绑定的页面元素，progressBar()为进度条插件初始化并绑定元素的方法，该方法中的参数persent表示当前进度的百分比值，参数config是一个对象，通过该对象可以设置进度条插件在初始化时的各项属性值，如进度条的速度、背景色等。

页面中的元素在调用progressBar()初始化进度条时，参数persent、config都是可选项，代码如下：

```
$(页面元素).progressBar()
```

上述代码仅是表示初始化一个进度表条页面元素，并没有指定该进度条的值，另外，进度条的当前值也可以从元素中获取。

此外，也可以指定进度条的值，初始化页面元素，代码如下：

```
$(页面元素).progressBar(persent)
```

上述代码表示按指定的百分比值，初始化一个进度条页面元素，persent表示进度条插件的当前百分比进度值。此外，也可以调用两个参数初始化页面进度条元素，代码如下：

```
$(页面元素).progressBar(persent, config)
```

上述代码表示，按指定的persent百分比值和config对象设置的属性值，初始化一个进度条页面元素。config是一个对象，它包含设置进度条的多项属性和事件，其属性和事件的详细功能说明如表8-2所示。

表 8-2　config 对象中的常用参数

参数名称	功能描述
steps	进度条向前增进的步长，默认值为 2 像素
stepDuration	进度条向前滑进时的速度，默认值为 15，值越大，滑进的速度越慢
showText	是否显示进度条右侧的百分比文字，默认值为 true，表示显示
boxImage	设置滑动条边框图片背景 URL，默认值为“images/progressbar.gif”
barImage	设置进度条主体图片背景 URL，插件自带 5 种主体图片背景，默认值为“images/progressbg_green.gif”，其余 4 种 URL 地址分别为“images/progressbg_black.gif”、“images/progressbg_orange.gif”、“images/progressbg_red.gif”、“images/progressbg_yellow.gif”

（续）

参数名称	功能描述
width	设置进度条的宽度，该值必须与边框和主体背景图片相一致，默认值为 120 像素
height	设置进度条的宽度，该值必须与边框和主体背景图片相一致，默认值为 12 像素
max	设置进度条的最大值，即最大范围，进度条的当前值必须小于或等于该值
textFormat	设置进度条右侧百分比文字显示的格式，该格式有两种：一种是百分比，另一种为分数形式，默认为百分比形式
callback	回调函数，当进度条滑动的值等于设置的百分比值时触发该回调函数，即当进度条滑动效果结束时，调用该函数

说明　当使用页面元素调用进度条的progressBar()方法时，不管该方法中有无参数，只能对一个元素使用一次，不能重复使用，否则，进度条的显示将会有异常。

8.2.11　页面加载遮盖插件LoadMask

LoadMask是一款能够屏蔽DOM元素并显示加载提示的jQuery插件，它的主要功能是可以屏蔽指定元素中的内容区域。如该元素是一个提交数据的表单，那么只要执行该插件的mask()方法，在表单区域中将显示页面遮盖层和加载提示信息。此时，用户不能操作该表单中的任何元素，而表单的提交动作并不受影响。由于该插件具有这一特性，常用于表单提交、异步加载等耗时较长的操作中。此外，该插件体积很小，使用方法也十分方便，接下来我们通过一个简单的示例来说明该插件的使用过程。

示例8-11　LoadMask插件的使用

（1）插件文件

Js-8-11/jquery.loadmask.js

（2）下载地址

http://code.google.com/p/jquery-loadmask/

（3）功能描述

在页面中添加三个按钮，分别为“开始”、“结束”、“延时”。单击“开始”按钮时，执行页面加载遮盖提示；单击“结束”按钮时，隐藏页面加载遮盖提示；单击“延时”按钮时，按照设置的时间，延时执行页面加载遮盖提示。

（4）实现代码

新建一个HTML文件8-11.html,加入如代码清单8-11所示的代码。

代码清单 8-11　页面加载遮盖插件 LoadMask

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>页面加载遮盖插件 LoadMask</title>
        <link href="Css-8-11/jquery.loadmask.css"
```

```
        rel="stylesheet" type="text/css" />
    <script src="Jscript/jquery-1.8.2.min.js"
        type="text/javascript"></script>
    <script src="Js-8-11/jquery.loadmask.js"
        type="text/javascript"></script>
    <style type="text/css">
        body{font-size:13px}
        #ifra{border:solid 1px #555;
            width:260px;height:160px}
        #title{text-align:center;}
        #content{padding:10px;width:240px;
        height:114px;background-color:#fff;}
        .txt{border:#666 1px solid;
        padding:2px;width:150px;margin-right:3px}
        .btn {border:#666 1px solid;padding:2px;width:80px;
        filter: progid:DXImageTransform.Microsoft
        .Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            var $cnt = $("#content");
            $("#btnStart").click(function() {
                // 执行页面加载遮盖提示
                $cnt.mask(" 正在加载中 ...");
            });
            $("#btnEnd").click(function() {
                // 关闭页面加载遮盖提示
                if ($cnt.isMasked()) {
                    $cnt.unmask();
                }
            });
            $("#btnDelay").click(function() {
                // 延时执行页面加载遮盖提示
                $cnt.mask(" 正在加载中 ...", 1000);
            });
        });
    </script>
</head>
<body>
    <div id="ifra">
        <div id="title">
            <input type="button" id="btnStart"
                value=" 开始 " class="btn" />
            <input type="button" id="btnEnd"
                value=" 结束 " class="btn"/>
            <input type="button" id="btnDelay"
                value=" 延时 " class="btn"/>
        </div>
        <div id="content">
          用户名: <br />
              <input id="username" name="username"
                    type="text" class="txt" />
                    <font color="red">*</font><br />
          密码: <br />
```

```
                <input id="password" name="password"
                       type="password" class="txt" />
                       <font color="red">*</font><br />
                <input id="sbtUser" type="submit"
                       value="提交" class="btn" />
            </div>
        </div>
    </body>
    </html>
```

（5）页面效果

代码执行后的效果如图8-16所示。

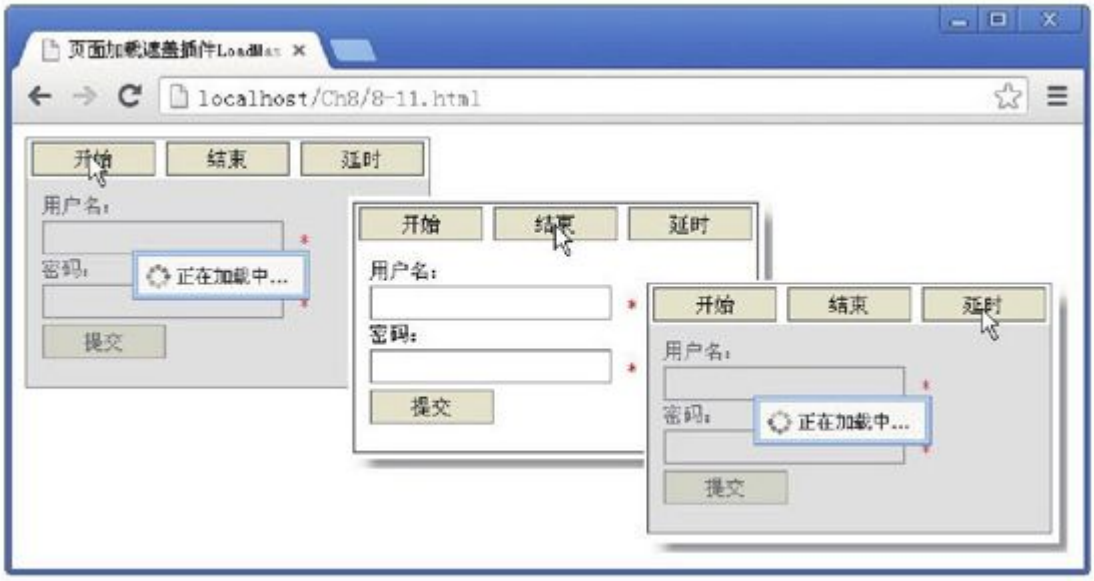


图　8-16　页面加载遮盖插件LoadMask

（6）代码分析

在本示例中，为了保证加载遮盖插件能够将提示内容正常显示在页面中，除在＜head＞元素中导入插件的JavaScript格式文件外，还需要引入相应的CSS样式文件，代码如下：

```
<link href="Css-8-11/jquery.loadmask.css"
          rel="stylesheet" type="text/css" />
<script src="Js-8-11/jquery.loadmask.js"
          type="text/javascript"></script>
```

接下来，为了显示页面加载遮盖层，需要调用插件对应的方法，LoadMask插件拥有3个自定义方法，分别为mask()、unmask()、isMasked()。

第一个方法调用的格式为：

```
$(页面元素).mask(label, [delay])
```

该方法的功能是在页面元素的区域中显示加载遮盖层，其中参数label表示提示内容，可选项参数delay表示延时加载时间，增加该参数时，将在设置的延时到达后，在元素的区域内显示加载遮盖层。

第二个方法调用的格式为：

```
$(页面元素).unmask()
```

该方法的功能是隐藏已显示的加载遮盖层，该方法无参数。

第三个方法调用的格式为：

```
$(页面元素).isMasked()
```

该方法的功能是返回页面元素区域中是否存在加载遮盖层，如果存在，返回true，否则，返回false。开发人员可以根据这一特征，在隐藏加载遮盖层之前，使用该方法进行判断，如果存在，则进行隐藏，完整代码见源码中加粗部分。

8.2.12　数据分页插件Pagination

Pagination是一款不但加载数据而且还能进行分页的jQuery插件，实现的过程是：先一次性将需要显示的数据载入到页面中，然后根据当前页面的索引号，获取指定某页需要显示的数据，并将这部分数据追加到内容显示容器中，从而实现分页加载数据的效果。

根据这一个过程，Pagination插件在加载和显示数据时，是一次性、及时的，而不是异步请求，因此，它在执行分页显示数据时，速度很快，用户体验很好。但这种一次性加载数据到页面的操作并不适合对大量数据的请求，因此在页面中，如果是小数据量的分页显示需求，使用该插件还是非常理想的选择。

接下来我们通过一个简单的示例来说明该插件的使用过程。

示例8-12　Pagination插件的使用

（1）插件文件

Js-8-12/jquery.pagination.js

（2）下载地址

http://d-scribe.de/webtools/jquery-pagination/demo/demo_options.htm

（3）功能描述

在页面中，通过数据分页插件，以列表的形式显示全部的订单信息，包括“订单号”、“姓名”、“性别”、“总金额”、“详细”等内容。用户单击“详细”列中“查看”链接时，再次通过数据分页插件显示对应订单的全部图片信息。

（4）实现代码

新建一个HTML文件8-12.html,加入如代码清单8-12所示的代码。

代码清单 8-12　数据分页插件 Pagination

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>数据分页插件 Pagination</title>
        <link href="Css-8-12/jquery.pagination.css"
              rel="stylesheet" type="text/css" />
        <script src="Jscript/jquery-1.8.2.min.js"
                type="text/javascript"></script>
        <script src="Js-8-12/jquery.pagination.js"
                type="text/javascript"></script>
        <style type="text/css">
```

```
            ... 省略样式部分代码
        </style>
        <script type="text/javascript">
            function psCallback(pi, jq) {
                //获取指定页码中的内容
                var newcxt = $('#Result ul:eq(' + pi + ')').clone();
                //向内容区添加指定页码的内容
                $('#Content').empty().append(newcxt);
                return false;
            }
            function isCallback(pi, jq) {
                //隐藏图片区域中的图片
                $('#Picture img:visible').hide();
                //显示图片区域中指定页码的图片
                $('#Picture img:eq(' + pi + ')').show();
            }
            function initPagination() {
                //获取全部记录数量
                var SumNum = $('#Result ul').length;
                //记录显示区绑定数据分页插件
                $(".divpage").pagination(SumNum, {
                    num_edge_entries: 1,
                    callback: psCallback,
                    items_per_page: 1
                });
                //获取全部图片数量
                SumNum = $('#Picture img').length;
                //图片显示区绑定另一个数据分页插件
                $(".divimg").pagination(SumNum, {
                    callback: isCallback,
                    items_per_page: 1
                });
            }
            function showPic() {
                $("#ImgLst").toggle();
            }
            $(function() {
                //调用数据分页插件
                initPagination();
            });
        </script>
    </head>
    <body>
    <div id="Iframe">
           <div class="divpage"></div>
           <br style="clear:both" />
           <ul>
              <li class="li">
                 <span>订单号 </span>
                 <span>姓名 </span>
                 <span>性别 </span>
                 <span>总金额 </span>
                 <span>详细 </span>
              </li>
           </ul>
```

```
<div id="Content">
   <span class="spntip">正在加载数据 ...</span>
</div>
<div class="divpage"></div>
<div id="Result">
      <ul>
         <li>
            <span>10101</span>
            <span>张小虎 </span>
            <span>男 </span>
            <span>465.98</span>
            <span>
                <a href="javascript:showPic()">
                   查看
                </a>
            </span>
         </li>
         <li>
            <span>10102</span>
            <span>刘海 </span>
            <span>女 </span>
            <span>600.12</span>
            <span>
                <a href="javascript:showPic()">
                   查看
                </a>
            </span>
         </li>
         <li>
            <span>10103</span>
            <span>李逗海 </span>
            <span>男 </span>
            <span>830.12</span>
            <span>
                <a href="javascript:showPic()">
                   查看
                </a>
            </span>
         </li>
      </ul>
      <ul>
         <li>
            <span>10201</span>
            <span>张三百 </span>
            <span>男 </span>
            <span>565.98</span>
            <span>
                <a href="javascript:showPic()">
                   查看
                </a>
            </span>
         </li>
         <li>
            <span>10202</span>
            <span>接小海 </span>
```

```
        <span>男</span>
        <span>900.12</span>
        <span>
            <a href="javascript:showPic()">
                查看
            </a>
        </span>
    </li>
    <li>
        <span>10203</span>
        <span>李明明</span>
        <span>女</span>
        <span>630.12</span>
        <span>
            <a href="javascript:showPic()">
                查看
            </a>
        </span>
    </li>
</ul>
<ul>
    <li>
        <span>10301</span>
        <span>陈虎</span>
        <span>男</span>
        <span>765.98</span>
        <span>
            <a href="javascript:showPic()">
                查看
            </a>
        </span>
    </li>
    <li>
        <span>10302</span>
        <span>周明海</span>
        <span>女</span>
        <span>802.12</span>
        <span>
            <a href="javascript:showPic()">
                查看
            </a>
        </span>
    </li>
    <li>
        <span>10303</span>
        <span>胡小明</span>
        <span>男</span>
        <span>630.12</span>
        <span>
            <a href="javascript:showPic()">
                查看
            </a>
        </span>
    </li>
</ul>
```

```
        </div>
        <div id="ImgLst">
           <div id="Picture" class="pic">
               <img src="Images-8-12/pic08.jpg"/>
               <img src="Images-8-12/pic09.jpg"/>
               <img src="Images-8-12/pic10.jpg"/>
               <img src="Images-8-12/pic11.jpg"/>
           </div>
           <div class="divimg"></div>
         </div>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-17所示。

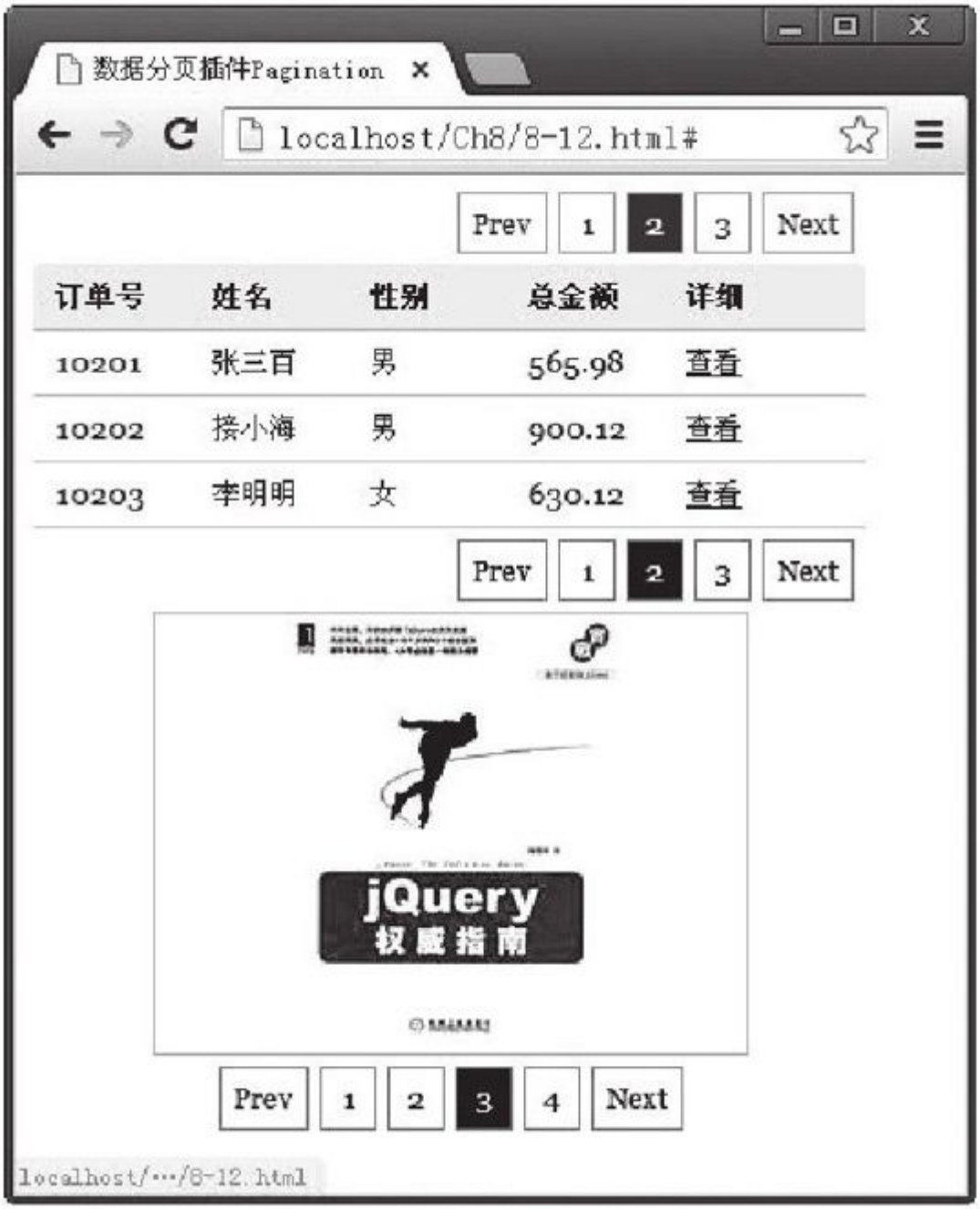


图　8-17　数据分页插件Pagination

（6）代码分析

在本示例中，在使用数据分页插件Pagination之前，首先在页面的＜head＞元素中导入对应的CSS样式和插件文件，代码如下：

```
<link href="Css-8-12/jquery.pagination.css"
      rel="stylesheet" type="text/css" />
<script src="Jscript/jquery-1.8.2.min.js"
      type="text/javascript"></script>
<script src="Js-8-12/jquery.pagination.js"
      type="text/javascript"></script>
```

在导入插件对应文件时，需要注意与jQuery框架的先后顺序，因为数据分页插件Pagination是基于jQuery开发的，应该首先导入jQuery框架，然后导入插件对应的JS文件，否则使用将会有异常。

在完成插件导入后，接下来的工作就是编写对应的JavaScript代码，将页面中的内容元素与数据分页插件相绑定，实现数据分页显示的效果，核心代码如下所示：

```
function psCallback(pi, jq) {
      //获取指定页码中的内容
      var newcxt = $('#Result ul:eq(' + pi + ')').clone();
      //向内容区添加指定页码的内容
      $('#Content').empty().append(newcxt);
      return false;
  }
  function isCallback(pi, jq) {
      //隐藏图片区域中的图片
      $('#Picture img:visible').hide();
      //显示图片区域中指定页码的图片
      $('#Picture img:eq(' + pi + ')').show();
  }
  function initPagination() {
      //获取全部记录数量
      var SumNum = $('#Result ul').length;
```

```
    //记录显示区绑定数据分页插件
    $(".divpage").pagination(SumNum, {
        num_edge_entries: 1,
        callback: psCallback,
        items_per_page: 1
    });
    //获取全部图片数量
    SumNum = $('#Picture img').length;
    //图片显示区绑定另一个数据分页插件
    $(".divimg").pagination(SumNum, {
        callback: isCallback,
        items_per_page: 1
    });
    }
.. 省略部分代码
```

在上述代码中，首先自定两个用户插件回调的函数psCallback和isCallback，分别用于记录和图片数据在绑定数据分页插件时的回调处理；在这两个回调函数中都有两个参数：

❑pi：int类型，表示新的页码索引号。

❑jq：一个jQuery对象，表示加载分页链接数据的容器。

在编写完回调函数后，接下来自定义一个initPagination()函数，将页面中的元素数据分页插件相绑定，绑定的格式如下：

```
$(内容显示容器). pagination(num_display_entries,[options]
```

在上述代码中，参数num_display_entries为必选项，表示需要显示的总条目数量，页面一次性加载后的记录总量；参数options是一个可选项对象，该对象可以设置数据插件绑定元素时的相关属性，如显示主体记录条数、上一页或下一页对应的文本内容，该对象的详细功能属性和事件说明如表8-3所示。

表 8-3　options 对象中的常用参数

参数名称	功能描述
items_per_page	每页显示的条目数量，默认值为 10
num_display_entries	连续分页页码主体部分显示的数量，默认值为 10
current_page	当前选中的页面，默认值为 0，表示第 1 页
num_edge_entries	分页页码首尾部分显示的条目数量，默认值为 0
link_to	分页时链接，默认值为“#”
prev_text	“上一页”分页按钮显示的文字内容，默认值为“Prev”
next_text	“下一页”分页按钮显示的文字内容，默认值为“Next”
ellipse_text	省略的分页页码的表示文字，默认值为“…”
prev_show_always	是否显示“上一页”分页按钮，默认值为 true，表示显示
next_show_always	是否显示“下一页”分页按钮，默认值为 true，表示显示
renderer	分页时的模块名称，默认值为“defaultRenderer”
show_if_single_page	如果只有一页，是否显示分页导航，默认值为 false，表示不显示
load_first_page	是否首先显示第 1 页，默认值为 false，表示不显示
callback	数据插件绑定时，执行的回调函数，默认值为无执行效果

8.2.13　消息通知条插件Activebar2

Activebar2插件可以跨浏览器平台，其依附于HTML 5和CSS 3语法，基于jQuery框架而开发。该插件的功能是：在紧靠浏览器的最顶部，以消息栏提醒的方式通知用户自定义的内容。当网站有重要、紧急的事宜需要通知网友时，使用该插件是一个不错的选择。

Activebar2插件是基于jQuery框架编写的纯JavaScript代码。该插件仅有10KB，使用十分方便，只需调用插件自带的activebar()方法，并在方法中设置显示的内容即可。接下来通过一个简单的示例来说明该插件的使用过程。

示例8-13　Activebar2插件的使用

（1）插件文件

Js-8-13/activebar2.js

（2）下载地址

http://westhoffswelt.de/projects/activebar2.html

（3）功能描述

在页面中调用消息通知条插件Activebar2，页面加载完成时，在浏览器的最顶部显示自定义的网站“温馨提示”信息。

（4）实现代码

新建一个HTML文件8-13.html,加入如代码清单8-13所示的代码。

代码清单 8-13　消息通知条插件 Activebar2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>消息通知条插件 Activebar2</title>
        <script src="Jscript/jquery-1.8.2.min.js"
                type="text/javascript"></script>
        <script src="Js-8-13/activebar2.js"
                type="text/javascript"></script>
        <style type="text/css">
              body{font:13px/1.6 Georgia, serif;color:#333}
        </style>
        <script type="text/javascript">
            $(function() {
                // 设置消息栏通知内容
            var $strTip = '温馨提示:明天凌晨2:00~3:00服务器升级，届时所有页面将无法浏览，敬请谅解！';
                // 在 div 元素中显示通知内容
                $('<div></div>').html($strTip).activebar({
                    'font': 'serif',              // 字体名
                    'url': 'http://www.rttop.cn' // 单击内容的 URL
                });
            });
        </script>
    </head>
    <body>

        <div>
            <h3>Activebar2 插件的应用 </h3>
            <p> 使用方法 </p>
        </div>
    </body>
    </html>
```

（5）页面效果

代码执行后的效果如图8-18所示。

图　8-18　消息通知条插件Activebar2在不同浏览器下显示的页面效果

（6）代码分析

在本示例的页面中，嵌入消息通知条的方式非常简单，首先在＜head＞元素中导入消息通知条插件，然后编写下列代码，就可以在浏览器的最顶部显示消息通知条：

```
$(function() {
        // 设置消息条通知内容
        var $strTip = '温馨提示:明天凌晨2:00~3:00服务器升级，届时所有页面将无法浏览，敬请谅解！';
            // 在div元素中显示通知内容
            $('<div></div>').html($strTip).activebar({
                'font': 'serif',              // 字体名
                'url': 'http://www.rttop.cn' // 单击内容的URL
            });
        });
```

在上述代码中，定义$strTip变量保存消息栏的内容，接下来使用$('＜div＞＜/div＞')代码向当前页插入了一个新的＜div＞元素，并将变量$strTip的值设置为该＜div＞元素的内容，同时用activebar()

方法，将该＜div＞元素与消息通知条插件相绑定。在绑定的过程中设置对应的属性，如"font"属性，表示通知条内容的字体名称等，当用户单击通知条最右侧的“关闭”按钮时，将隐藏该＜div＞元素，实现关闭消息通知条的效果。

在调用activebar()方法时，还提供了定制消息通知条的一系列选项options，通过该选项对象，可以设置如前面所提及的通知条内容的"font"属性，除此之外，还有一些其他相关属性的设置，如表8-4所示。

表 8-4　options 对象中的常用参数

参数名称	功能描述
background	消息通知条所使用的背景色
border	消息通知条所使用的 1 像素的边框背景色
highlight	当鼠标移到消息通知条时，所显示的背景色
font	消息通知条内容使用的字体名称
fontColor	消息通知条内容使用的字体颜色
fontSize	消息通知条内容使用的字体大小
icon	消息通知条内容左侧的信息图标 URL
button	消息通知条内容右侧的删除图标 URL
url	单击通知条内容时，所跳转的页面 URL

8.2.14　滚动条插件NiceScroll

众所周知，在页面中实现滚动的效果不是太容易，尤其要兼容其他的浏览器，代码相对要复杂些，特别是兼容移动设备的浏览器。NiceScroll是一款完全基于jQuery框架的滚动条插件，它不仅有着类似iOS系统设备的滚动条外观，而且还支持任意的＜div＞、＜iframe＞、＜body＞元素的滚动效果，该滚动效果也完全兼容各类桌面和移动设备的主流浏览器。此外该插件还拥有iOS系统设备中的触摸事件，比较适用于iOS系统设备中的页面展示。

接下来我们通过一个简单的示例来说明该插件在桌面浏览器中的使用过程。

示例8-14　NiceScroll插件的使用

（1）插件文件

Js-8-14/jquery.nicescroll.js

（2）下载地址

http://areaaperta.com/nicescroll

（3）功能描述

在页面中添加三个用于元素滚动的区域。第一个区域用于图片的滚动查看，第二个区域用于一个＜div＞元素的滚动查看，第三个区域用于一个＜iframe＞元素的滚动查看。当页面加载完成后，分别查看这三个区域各元素的滚动效果。

（4）实现代码

新建一个HTML文件8-14.html,加入如代码清单8-14所示的代码。

代码清单 8-14　滚动条插件 NiceScroll

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <title>jQuery 滚动插件 NiceScroll</title>
         <script src="Jscript/jquery-1.8.2.min.js"
                 type="text/javascript"></script>
```

```
    <script src="Js-8-14/jquery.nicescroll.js"
            type="text/javascript"></script>
    <style type="text/css">
           body{font:13px/1.6 Georgia, serif;color:#333}
           .bs{padding:5px;height:120px;margin:10px 0px;
               border:1px solid #555; overflow:auto;width:380px}
            .bs .scrimg{ width:762px}
            .bs .scrimg img{ width:150px; height:200px}
            .p{height:500px}
</style>
    <script type="text/javascript">
        $(function() {
            // 图片区域显示滚动效果
            $("#Box_Scr_1").niceScroll("#Box_Scr_1 div", {
                touchbehavior: true,
                cursorcolor: "#333",
                cursoropacitymax: 0.6,
                autohidemode: false,
                boxzoom: true
            });
            //div 区域显示滚动效果
            $("#Box_Scr_2").niceScroll("#content", {
                cursorcolor: "#666",
                autohidemode: false,
                boxzoom: true
            });
            //iframe 区域显示滚动效果
            $("#Box_Scr_3").niceScroll("iframe", {
                cursorcolor: "#ccc",
                autohidemode: false,
                boxzoom: true
            });
        });
    </script>
</head>
<body>
<div id="Box_Scr_1" class="bs">
   <div class="scrimg">
       <img src="Images-8-14/pic10.jpg"
            onmousedown="return false"/>
       <img src="Images-8-14/pic10.jpg"
            onmousedown="return false"/>
       <img src="Images-8-14/pic10.jpg"
            onmousedown="return false"/>
       <img src="Images-8-14/pic10.jpg"
            onmousedown="return false"/>
       <img src="Images-8-14/pic10.jpg"
            onmousedown="return false"/>
   </div>
</div>
<div id="Box_Scr_2" class="bs">
   <div id="content">
```

```
            <h3>这是一个 div 区域</h3>
            <p class="p">内容区域</p>
        </div>
    </div>
    <div id="Box_Scr_3" class="bs">
        <iframe src="8-14-1.html" height="100%"
                width="100%" frameborder="0"
                onload="$(this).height($(this).contents().height());">
        </iframe>
    </div>
    </body>
    </html>
```

在本示例中，＜iframe＞元素的src属性中导入了一个名为“8-14-1”的HTML页面文件，该文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>iframe 页</title>
    <style type="text/css">
        body{font:13px/1.6 Georgia, serif;color:#333}
        p{height:500px}
    </style>
</head>
<body>
   <div>
       <h3>这是一个 iframe 页</h3>
       <p>内容区域</p>
   </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图8-19所示。

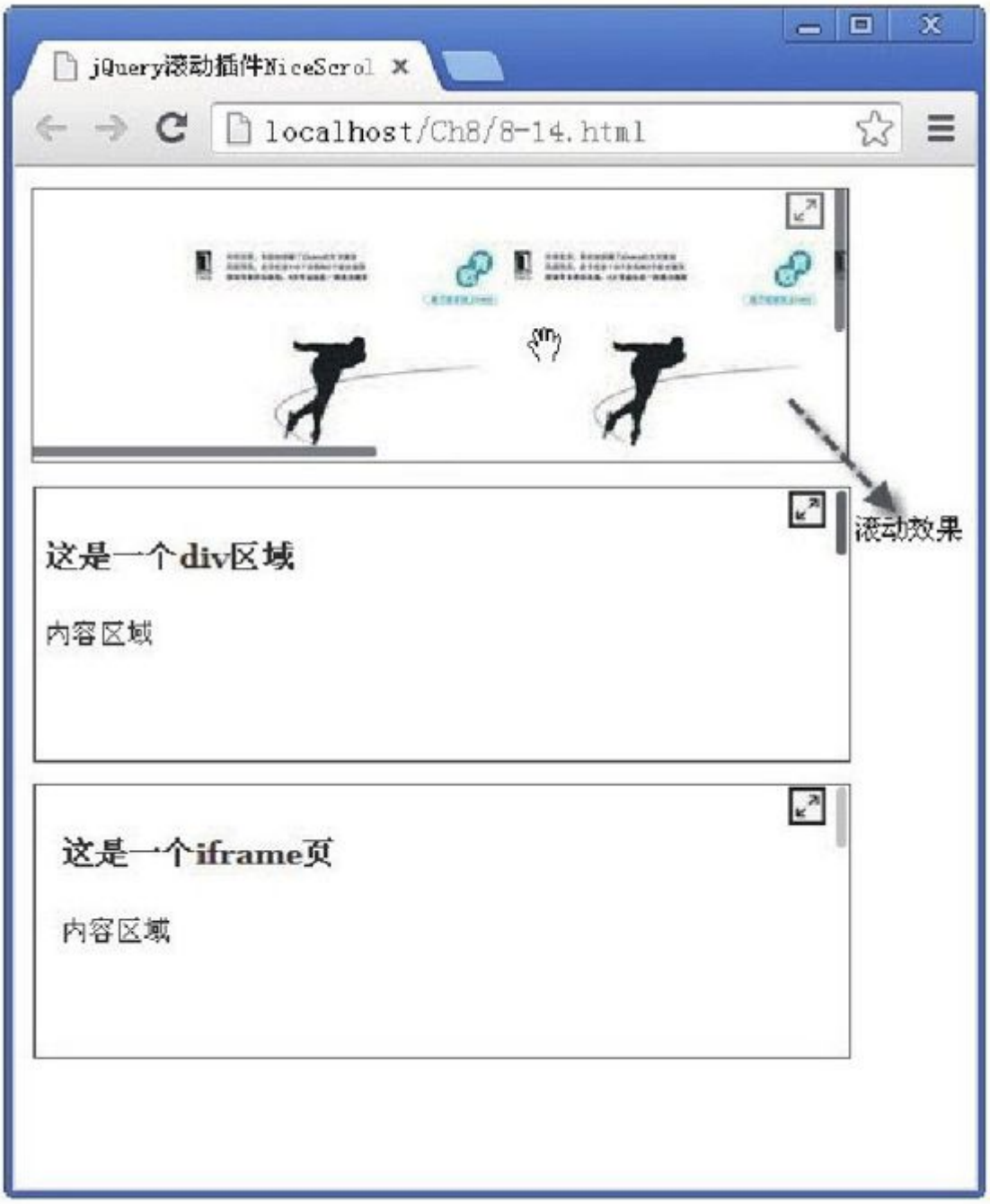


图　8-19　滚动条插件NiceScroll实现各类页面元素的滚动效果

（6）代码分析

在本示例中，第一个区域用于图片的滚动查看。在该区域中，图片区的宽度已超出外围区域的宽度，因此，通过下列代码将图片区与滚动条插件绑定：

```
//图片区域显示滚动效果
$("#Box_Scr_1").niceScroll("#Box_Scr_1 div", {
    touchbehavior: true,
    cursorcolor: "#333",
    cursoropacitymax: 0.6,
    autohidemode: false,
    boxzoom: true
    });
```

上述代码中，$("#Box_Scr_1")表示获取图片的外围区域，niceScroll()方法中的实参"#Box_Scr_1 div"，表示被插件绑定的滚动区域元素标识，本示例中用的ID属性，也可以使用元素的其他属性来标识被插件绑定的滚动区域。

第二个参数为一个选项对象，在该对象中，可以设置元素在被滚动插件绑定时的各类属性，如"boxzoom"属性表示是否显示放大的图标，如果该值为true，将在滚动条最上方显示一个  图标，单击该图标时，将直接显示滚动条区域中的内容；在该内容区域中，可以单击右上角的  图标返回内容区域的上级容器页面。

另外两个区域分别使用滚动条与＜div＞和＜iframe＞元素相绑定，使用这两个元素在高度超过外围高度值时，可以通过拖动滚动条进行查看，其绑定的方法与第一外区域基本相同，只是在设置滚动条的颜色时，各有差异。另外，如果需要将＜body＞元素与滚动条插件相绑定时，只需要加入如下代码：

```
$("body").niceScroll();
```

与其他jQuery插件相类似，滚动条插件NiceScroll在调用niceScroll()时，也自带一个可选项对象options，通过该对象可以设置绑定滚动时的相关属性。该对象的一些常用属性如表8-5所示。

表 8-5　options 对象中的常用参数

参数名称	功能描述
touchbehavior	是否是触摸式滚动效果
cursorcolor	滚动条的颜色值
cursoropacitymax	滚动条的透明度值
cursorwidth	滚动条的宽度值
background	滚动条的背景色
autohidemode	滚动条是否是自动隐藏，默认值为 true，表示是自动隐藏
boxzoom	是否显示放大图标按钮，默认值为 false，表示不显示

8.3　自定义jQuery插件

虽然有大量优秀的插件可供用户免费下载使用，但开发人员更希望自己能编写根据自身项目特点使用的插件，提供给团队或他人使用。在jQuery中编写插件不是一件很难的事，只要编写的代码符合插件的各项功能要求，就可以实现自定义各项目功能的插件。在介绍如何编写自定义的插件前，先来了解插件的基础知识。

8.3.1　自定义插件的种类

从广义上来说，插件分类3类，即封装方法插件、封闭函数插件、选择器插件，但最后一种很少有人去开发使用，自定义的插件种类多数属于前面两种。

1.封装方法插件

封装方法插件在本质上来说，是一个对象级别的插件，该类插件首先通过jQuery选择器获取对象并为对象添加方法，然后将方法进行打包封装成一个插件。这种类型的插件编写简单，极易调用，方便地使用了jQuery中功能强大的选择器，因此成为开发插件的首选。

2.封闭函数插件

封闭函数插件是一个类级别的插件，该类插件最大的特点，就可以直接给jQuery添加静态方法，并且可以将函数置于jQuery命名空间中，如最为常见的就是$.ajax()、$.trim()全局性函数，都是以内部插件的形式植入jQuery内核中。

8.3.2　插件开发要点

在了解完插件的种类后，接下来我们介绍插件所具有的特点，这些特点也是我们在开发插件时必须注意的事项。

❑插件的文件命名必须严格遵循jQuery.[插件名].js的规则，以便于与其他JS文件的区分，如新插件文件jquery.newplugin.js。

❑如果是对象级别插件，所有的方法都应依附于jquery.fn主体对象；如果是类级别插件，所有的方法都应依附于jQuery对象。

❑无论是对象级别还是类级别插件，结尾都必须以分号结束，否则在文件被压缩时，会出现错误的提示信息。

❑在插件内部的代码中，如果要访问每个元素，可以使用this.each方法来遍历全部元素。

❑需要说明的是在插件的内部，this所代表的是通过jQuery选择器所获取的对象，而非传统意义上的对象的引用。

❑由于jQuery代码在调用方法时，可以采用链写的方法同时调用多个方法，为了保证这功能的实现，插件本身必须返回一个jQuery对象。

❑“$”美元符虽然可以与"jQuery"字符相代替，但在编写插件的代码中，尽量不要使用“$”符号，避免与别的代码的冲突。

❑在编写对象级别的插件时，使用jQuery.fn.extend()方法进行功能扩展，而针对类级别的插件，则使用jQuery.extend()方法进行扩展。

8.3.3　对象级别插件的开发

示例8-15　对象级别插件的开发

（1）功能描述

在列表＜ul＞元素中，鼠标在表项＜li＞元素移动时，可以自定义其获取焦点（focus）时的背景颜色，即设置表项＜li＞元素选中时的背景色。

（2）搭建框架

新建一个JS文件，命名为jquery.lifocuscolor.js，并在文件中使用$.fn.extend()方法完成框架的搭建，其实现的代码如下：

```
/*-------------------------------------------------------------------/
功能：设置列表中表项获取鼠标焦点时的背景色
参数：li_col【可选】鼠标所在表项行的背景色
返回：原调用对象
示例：$("ul").focusColor("red");
/-------------------------------------------------------------------*/
;(function($){
   $.fn.extend({
         "yourPluginName": function(pram_value){
           // 各种默认属性或参数的设置
               this.each(function(){
               // 插件实现的代码
               })
           }
       })
})(jQuery);
```

（3）代码编写

根据功能描述，在搭建的框架中，首先设置插件的默认属性值，由于允许用户设置自己的颜色值，因此创建一个颜色参数，并对该值进行初始化设置；同时，根据参数是否为空，赋予该参数不同的颜色值；另外设置一个变量，保存丢失焦点时的颜色值，其实现的代码如下所示：

```
;(function($) {
    $.fn.extend({
      "focusColor": function(li_col) {
         var def_col = "#ccc"; // 默认获取焦点的色值
         var lst_col = "#fff"; // 默认丢失焦点的色值
         // 如果设置的颜色不为空，使用设置的颜色，否则为默认色
         li_col = (li_col == undefined) ? def_col : li_col;
         // 遍历表项 <li> 中的全部元素
         $(this).find("li").each(function() {
         ...
        });
      }
    });
})(jQuery);
```

在遍历表项＜li＞中，需针对象编写两个事件。

一个是鼠标移入事件mouseover()，在该事件中，将传回的变量def_col值设置为对象的背景色，其代码如下：

```
$(this).find("li").each(function() {
    $(this).mouseover(function() {                    // 获取鼠标焦点事件
        $(this).css("background-color", li_col); // 使用设置的颜色
    })
})
```

另一个是鼠标移出事件mouseout()，在该事件中，将背景色还原成鼠标移入前被变量lst_col保存的颜色值，其代码如下：

```
$(this).find("li").each(function() {
    $(this).mouseout(function() { // 鼠标焦点移出事件

        $(this).css("background-color", lst_col); // 恢复原来的颜色
        })
})
```

这两个事件可以进行合并，最终的代码如下：

```
$(this).find("li").each(function() {                    // 遍历表项 <li> 中的全部元素
    $(this).mouseover(function() {                       // 获取鼠标焦点事件
        $(this).css("background-color", li_col); // 使用设置的颜色
        }).mouseout(function() {                        // 鼠标焦点移出事件
        $(this).css("background-color", "#fff"); // 恢复原来的颜色
        })
 })
```

在代码最后结果时，必须返回一个jQuery对象，以方便其调用对象的链写操作。因此，最后加上一行返回jQuery对象，代码如下：

```
return $(this); // 返回 jQuery 对象，保持链式操作
```

经过上述的分析，JS文件jquery.lifocuscolor.js最终完整的代码如下所示：

```
;(function($) {
    $.fn.extend({
    "focusColor": function(li_col) {
      var def_col = "#ccc"; //默认获取焦点的色值
      var lst_col = "#fff"; //默认丢失焦点的色值

      //如果设置的颜色不为空，使用设置的颜色，否则为默认色
      li_col = (li_col == undefined) ? def_col : li_col;
      //遍历表项<li>中的全部元素
      $(this).find("li").each(function() {
         $(this).mouseover(function() {              //获取鼠标焦点事件
           $(this).css("background-color", li_col); //使用设置的颜色
                 })
         .mouseout(function() {                          //鼠标焦点移出事件
           $(this).css("background-color", "#fff"); //恢复原来的颜色
                 })
           });
           return $(this); //返回jQuery对象，保持链式操作
       }
    });
})(jQuery);
```

（4）引用插件

自己编写的插件保存为JS文件后，就可以像其他插件一样，被需要使用的文件所调用。其使用的方法也是一样，先引入JS文件，然后在JS代码中调用该插件中的各种方法。为了验证文件jquery.lifocuscolor.js中插件功能，新建一个HTML文件pl_LiGetFocus.html，引入该插件文件并调用该插件的方法，其加入的代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>编写一个对象级别的插件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
```

```
    </script>
    <script type="text/javascript"
            src="Plugin/jquery.lifocuscolor.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#ul").focusColor();        // 调用自定义的插件
            //$("#ul").focusColor("red");// 调用自定义的插件
        })
    </script>
</head>
<body>
    <div class="divFrame">
         <div class="divTitle">
              对象级别的插件
         </div>
         <div class="divContent">
              <ul id="ul">
                  <li><span> 张三 </span><span> 男 </span></li>
                  <li><span> 李四 </span><span> 女 </span></li>
                  <li><span> 王五 </span><span> 男 </span></li>
              </ul>
         </div>
     </div>
</body>
</html>
```

（5）页面效果

执行HTML文件pl_LiGetFocus.html后，其实现的页面效果如图8-20所示。

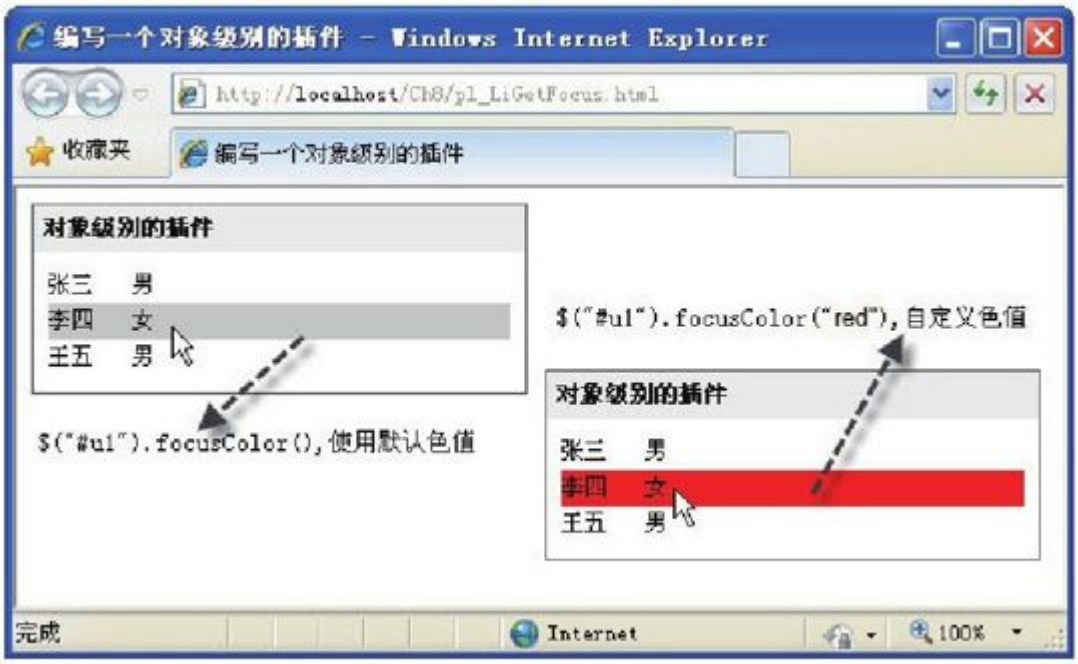


图　8-20　引用自定义的对象级别插件

8.3.4　类级别插件的开发

示例8-16　类级别插件的开发

（1）功能描述

新增两个类级别的全局函数，分别用于计算两数之和与两数之差，并将结果返回调用的页面中。

（2）搭建框架

新建一个JS文件，命名为jquery.twoaddresult.js，在该文件中使用$.extend()方法完成框架的搭建，其实现的代码如下：

```
/*--------------------------------------------------------------------/
功能：计算二个数字相加或相减的结果
参数：数字 p1,p2
返回：两数相加后的结果
示例：$.AddNum(1,2);
/--------------------------------------------------------------------*/
; (function($){
    $.extend({
       "yourPluginName": function(pram_value){
         //插件实现的代码
           }
    });
})(jQuery);
```

（3）代码编写

根据功能描述，编写一个用于计算两数之和的全局函数，该函数先对用户传来的两个参数进行检测，判断其是否为undefined类型，以

获取其最终用于计算的值，然后通过return语句返回其最终的计算结果，其实现的代码如下：

```
"addNum": function(p1, p2) {
   //如果传入的数字不为空，使用传入的数字，否则为0
   p1 = (p1 == undefined) ? 0 : p1;
   p2 = (p2 == undefined) ? 0 : p2;
   var intResult = parseInt(p1) + parseInt(p2);
   return intResult;
}
```

使用同样的方式，增加一个用户计算两数之差的全局函数，由于最后返回值是一个两数相减的结果，因此，在进行基本检测后，判断第一个参数是否大于第二个参数，如果大于，则返回前者减去后者的结果，其实现的代码如下：

```
"subNum": function(p1, p2) {
   //如果传入的数字不为空，使用传入的数字，否则为0
   var intResult = 0;
   p1 = (p1 == undefined) ? 0 : p1;
   p2 = (p2 == undefined) ? 0 : p2;
   if (p1 > p2) {//如果传入的参数前者大于后者
       intResult = parseInt(p1) - parseInt(p2);
   }
   return intResult;
}
```

使用jQuery.extend()方法直接对jQuery对象进行拓展，以扩充其全局函数，其最终实现的完整代码如下：

```
; (function($) {
    $.extend({
```

```
        "addNum": function(p1, p2) {
            //如果传入的数字不为空，使用传入的数字，否则为0
            p1 = (p1 == undefined) ? 0 : p1;
            p2 = (p2 == undefined) ? 0 : p2;
            var intResult = parseInt(p1) + parseInt(p2);
            return intResult;
        },
        "subNum": function(p1, p2) {
            //如果传入的数字不为空，使用传入的数字，否则为0
            var intResult = 0;
            p1 = (p1 == undefined) ? 0 : p1;
            p2 = (p2 == undefined) ? 0 : p2;
            if (p1 > p2) { //如果传入的参数前者大于后者
                intResult = parseInt(p1) - parseInt(p2);
            }
            return intResult;
        }
    });
})(jQuery);
```

（4）引用插件

与引用对象级别插件一样，类级别的插件也是先在＜title＞标记中导入插件的JS文件，然后编写JS代码，调用插件中的公用方法或函数。

为检测插件的功能，新建一个HTML文件pl_TwoCalculate.html。在该页面中，单击上下两个“等于”按钮后，分别调用插件中的全局函数addNum与subNum，计算文本框中的两数之和与两数之差，其实现的页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>编写一个类级别的插件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Plugin/jquery.twoaddresult.js">
    </script>
    <style type="text/css">
          ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() {
                $("#Text3").val(
                $.addNum($("#Text1").val(),$("#Text2").val()));
            }); //调用自定义的插件计算两数之和
            $("#Button2").click(function() {
                $("#Text6").val(
                $.subNum($("#Text4").val(),$("#Text5").val()));
            }) //调用自定义的插件计算两数之差
        })
    </script>


</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            类级别的插件
        </div>
        <div class="divContent">两数相加:
            <input id="Text1" type="text" class="txt" />
            +
            <input id="Text2" type="text" class="txt" />
            =
            <input id="Text3" type="text" class="txt" />
            <input id="Button1" type="button"
                   value="等于" class="btn" />
            <hr />两数相减:
            <input id="Text4" type="text" class="txt" />
            - 
            <input id="Text5" type="text" class="txt" />
            =
            <input id="Text6" type="text" class="txt" />
            <input id="Button2" type="button"
                   value="等于" class="btn"/>
        </div>
     </div>
    </body>
</html>
```

（5）页面效果

执行HTML文件pl_TwoCalculate.html后，实现的页面效果如图8-21所示。

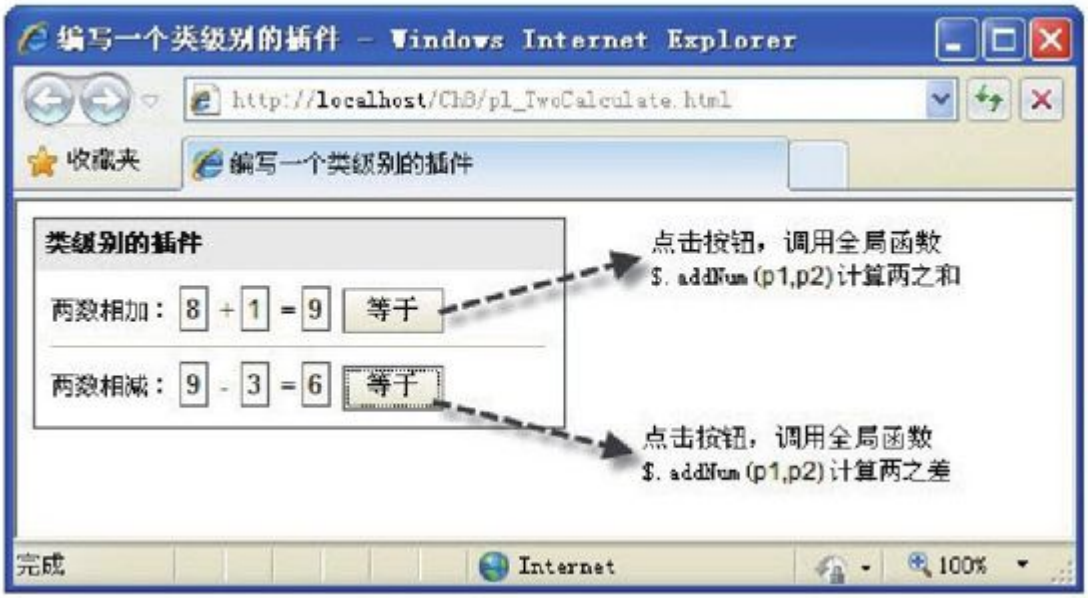


图　8-21　引用自定义的类级别插件

说明　无论是编写对象级别插件，还是编写类级别插件，都要严格遵守插件开发的要素，先搭框架，后进行开发，不同的插件使用不同的拓展方法。

8.4　综合案例分析——使用uploadify插件实现文件上传功能

8.4.1　需求分析

该案例的需求如下：

1）可以选择上传文件类型、限制上传文件大小，实现单个或多个文件同时上传功能，在上传过程中显示文件上传进度比例信息。

2）文件成功上传后，如果是图片文件，则可以进行图片预览功能。

3）无论是单个或多个文件成功上传后，可删除其队列中的某一个文件。

8.4.2　界面效果

选择单个或多个文件上传时，显示带百分比的进度条，提示文件上传进度信息，实现的效果如图8-22所示。

图　8-22　单个或多个文件上传时的进度信息提示

文件上传成功后，通过回调函数可以返回上传文件的URL。这个URL地址如果是图片，可以实现图片预览的功能，其实现的页面效果如图8-23所示。在文件上传成功后预览图片时，选择某个文件，单击“删除”链接，可以删除指定的某个上传后的文件，其实现的页面效果如图8-24所示。


uploadify文件上传插件 - Windows Inter...
http://localhost/Ch8/UploadPic.html
收藏夹
uploadify文件上传插件
上传图片
ASP
ASPNET 2.0
C#网站开发
删除
删除
删除
上传图片
Internet
100%

图　8-23　图片文件上传


uploadify文件上传插件 - Windows Inter...
http://localhost/Ch8/UploadPic.html
收藏夹
uploadify文件上传插件
上传图片
删除
删除
上传图片
点击“删除”链接后，删除第 2 张上传后的图片
Internet
100%

图　8-24　删除指定图片

8.4.3 插件介绍

在本案例中，使用了基于jQuery库开发的上传文件插件uploadify。该插件是众多插件中较为优秀的一款，支持多文件自动上传，带上传进度条和百分比例，并支持多种函数触发事件，如函数onComplete（完成后）、onError（出错时）等。该插件广泛应用于各项目中，深受开发者的喜爱。

在本案例中，使用了uploadify插件的一些属性和函数，该插件的核心方法是uploadify()，其调用的格式如下：

```
$(上传文件元素).uploadify(options)
```

其中参数options是一个对象，它所包含的常用属性见表8-6。

表 8-6　options 对象包含的常用属性

参数名称	功能描述
uploader	上传控件的主体文件，一个 Flash 控件，默认值为“uploadify.swf”
script	相对路径的服务器端文件名，它将处理上传的文件，绝对路径前缀或“/”或“http”的路径。默认值为“uploadify.php”
folder	上传文件保存在服务器端的路径
queueID	上传文件的队列 ID 号，与客户端页的 ID 号相同
queueSizeLimit	设置在一次队列中的文件总数，单击“浏览”按钮后，可以同时选择多少个文件
multi	是否允许同时上传多文件，可设定 true 或 false。默认值为 false
auto	选定文件后是否自动上传，可设定 true 或 false。默认值为 false
fileDesc	出现在上传对话框中，对文件类型描述，必须与 fileExt 结合使用才有效果
fileExt	上传文件所支持的格式，启用本项时需同时声明 fileDesc。如：“*.jpg,*.bmp,*gif”
sizeLimit	设置上传文件的最大值，单位为字节
simUploadLimit	在多个文件上传时，设置同时上传文件数目。默认值为 1
buttonText	默认选择上传文件按钮的文字。默认值为 BROWER
buttonImg	默认选择上传文件按钮的图片地址
onInit	该函数用于检测上传文件时的初始状态，使用方法如下： onInit: function() { $("#id").html(" 正在初始化 ..."); }
onComplete	该函数用于文件上传成功后触发，可传回 5 个参数，其详细使用见本案例
onSelect	该函数用于选择文件后触发，可传回 3 个参数 event、queueID、fileObj，各参数代表的意义与 onComplete 函数一样
onError	该函数用于文件上传出错后触发，可传回 4 个参数 event、queueID、fileObj、errorObj。其中前三个参数与 onComplete 函数一样，errorObj 参数有两个属性：一个属性是 type，表示错误类型，有三种类型，即“HTTP”、“IO”、“Security”；另一个属性是 info，表示错误信息的描述

（1）插件文件

Js/jquery.uploadify.v2.1.0.js

Js/swfobject.js

Images/全部文件

Css/uploadify.css

（2）下载地址

http://www.uploadify.com/download/

8.4.4　功能实现

上传文件插件uploadify下载完成后，其使用方法也十分简单，下面介绍其使用的步骤：

1)在调用文件中引用两个JS文件：jquery.uploadify.v2.1.0.js用于文件的上传，swfobject.js用于设置选择上传文件的按钮。其实现的代码如下：

```
<script type="text/javascript" src="Js/swfobject.js"></script>
<script type="text/javascript"
          src="Js/jquery.uploadify.v2.1.0.js">
</script>
```

2)在引用插件的页面中编写JS代码，设置uploadify插件上传文件时必须使用的属性，如保存上传文件的地址、处理文件上传的服务器端文件名等。其实现的代码如下所示：

```
$(function() {
    $("#uploadify").uploadify({
        'uploader': 'Images/uploadify.swf',
        'script': 'Deal/UploadFile.ashx',
        'cancelImg': 'Images/cancel.png',
        'folder': 'Uploads/',
        'queueID': 'fileQueue',
        'buttonImg': 'Images/uploadify.jpg',
        'auto': true,
        'multi': true,
    })
})
```

在上述JS代码中，为了不使用插件默认的选择上传文件的按钮，我们调用了一个"buttonImg"属性，并通过该属性设置自己定义的按钮地址，为确保文件的正确上传，其他属性所指定的文件路径或ID号必须准确无误。

3)当插件的JS文件被引用并且相关的基本属性设置完成之后，在引用的页面文件中加入如下的HTML代码，用于实现上传文件的选择和上传队列进度信息的展示：

```
<div id="fileQueue"></div>
<input type="file" name="uploadify" id="uploadify" />
```

其中，＜div＞标记的ID号与JS代码中"queueID"属性的值相同的，该属性表示上传文件对列的ID号，即ID号为"fileQueue"的＜div＞标记用于存放上传文件的队列，并通过下列代码使页面中ID号为"uploadify"的文件上传元素与插件中的uploadify()方法相绑定：

```
$("#uploadify").uploadify({...})
```

4)编写处理文件上传的服务器端代码。

要实现文件的真正上传，还需要编写一个简单的服务器端文件，接受传来的文件队列，逐个保存在服务器。在本案例中，通过一个程序处理文件UploadFile.ashx实现上述的功能。其实现的代码如下：

```
<%@ WebHandler Language="C#" Class="UploadFile" %>
using System;
using System.IO;
using System.Web;
public class UploadFile : IHttpHandler {
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "text/plain";
        context.Response.Charset = "utf-8";
        HttpPostedFile file = context.Request.Files["Filedata"];
        // 获取传回的保存文件地址
        string uploadPath = HttpContext.Current
        .Server.MapPath(@context.Request["folder"]) + "\\";
        // 如果有文件
        if (file != null)
        {
            if (!Directory.Exists(uploadPath))        // 目录不存在
            {
                Directory.CreateDirectory(uploadPath);// 新创建目录
            }
            file.SaveAs(uploadPath + file.FileName);  // 逐个保存文件
            // 返回一个值，确保上传成功后，在前台的文件队列自动消失
            context.Response.Write("1");
        }
        else
        {
            context.Response.Write("0");
        }
    }
    ...
}
```

8.4.5 代码分析

经过上述四个步骤，就可以在一个HTML页面中通过引用uploadify插件中的方法，实现一个或多个文件的上传，完成了需求中的第一项功能。

在通过uploadify插件实现文件上传过程中有一个onComplete()函数，该函数在每个文件上传成功后便触发，并且在触发时传回5个参数，分别为event、queueId、fileObj、response、data，其各自代码的意义如下：

❑event：表示触发事件的对象。

❑queueId：上传文件中队列的标识，用于区分每一个上传文件，它由六位随机数组成。

❑fileObj：选择上传的文件对象。通过该对象，可以访问上传文件的相关属性，如name（名称）、size（大小）、creationDate（创建时间）、modificationDate（修改时间）、type（文件类型）、filePath（保存路径）。

❑response：服务器端文件上传处理程序返回的值，即文件UploadFile.ashx返回值。在本案例中，文件上传成功后，返回值为

1。

❑data：表示文件上传数据流。该对象有两个重要属性，一个是fileCount，表示还没有上传完的文件总量；另一个是speed表示文件的平均上传速度。

根据对onComplete()函数中各个返回参数的描述，可以通过第三个参数fileObj中的filePath属性值，返回上传文件后的URL。如果是图片上传，将返回的图片URL设置为客户端中＜img＞标记的src属性值，从而实现图片上传后预览的功能。其实现的JS代码如下：

```
function SetFileContent(objFile) { //根据上传对象返回预览的图片
    var sLi = "";
    sLi += "<li>";
    sLi += "<img src='" + objFile.filePath + "' width='100' height='100'>";
    sLi += "<input type='hidden' value='" + objFile.filePath + "'>";
    sLi += "<br />";
    sLi += "<a href='javascript:void(0)'> 删除 </a>";
    sLi += "</li>";
    return sLi;
}
```

其中，自定义的函数SetFileContent中的参数objFile就是onComplete()函数中的第三个参数fileObj，将上述函数SetFileContent放入onComplete()事件中，即实现了图片文件上传成功后，可以进行预览的功能，从而完成了第二项需求。其实现的代码如下：

```
$(function() {
   $("#uploadify").uploadify({
   ...
   'onComplete': function(event, queueID, fileObj, response, data) {
            $('ul').append(SetFileContent(fileObj));
     }
   })
})
```

为了在页面中展示预览的图片，需要加入一个列表＜ul＞标记，其页面代码如下：

```
<ul></ul>
<div id="fileQueue"></div>
<input type="file" name="uploadify" id="uploadify" />
```

图片上传成功后进行预览时，为了可以删除不太理想的图片文件，需要确定被删除表项＜li＞的ID号，使用jQuery中的remove()方法移除指定的表项。最后，在增加新表项内容时，重新分配表项的ID号，从而保证ID号的不重复。具体操作过程如下。

1)通过each方法遍历全部的表项＜li＞并设置对应的ID号，代码如下：

```
//设置各表项ID号
$("ul li").each(function(l_i) {
      $(this).attr("id", "li_" + l_i);
})
```

2)通过each方法遍历全部表项中“删除”链接，设置对应的"rel"属性，用于传递删除表项的ID号，其代码如下：

```
//设置各链接的rel属性值
$("ul li a").each(function(a_i) {
      $(this).attr("rel", a_i);
})
```

3)根据表项中“删除”链接中的"rel"属性，设置删除某个表项的单击事件，其实现的代码如下：

```
//设置各链接的单击事件
$("ul li a").click(function() {
      //通过自身的rel值，获取表项的ID号
      var li_id = "#li_" + this.rel;
      //根据ID号，删除某个表项
      $(li_id).remove();
 })
```

由于上述操作都在同一函数中触发，并且可以进行链写，因此将其封装到另外一个自定义的函数SetUploadFile中，其最后完整的代码如下：

```
function SetUploadFile() {
    //设置各表项ID号
    $("ul li").each(function(l_i) {
        $(this).attr("id", "li_" + l_i);
    })

    //设置各链接的rel属性值
    $("ul li a").each(function(a_i) {
        $(this).attr("rel", a_i);
    }).click(function() {//设置各链接的单击事件
        //通过自身的rel值，获取表项的ID号
        var li_id = "#li_" + this.rel;
        //根据ID号，删除某个表项
        $(li_id).remove();
    })
}
```

最后，将上述自定义函数SetUploadFile放入onComplete()函数中，就可以在预览上传图片时进行删除的操作，从而完成了第三项需求。其实现的代码如下：

```
$(function() {
   $("#uploadify").uploadify({
        ...
        'onComplete': function(event, queueID, fileObj, response, data) {
                $('ul').append(SetFileContent(fileObj));
                SetUploadFile();
        }
   })
})
```

为了更好地实现页面与代码的分离，本案例中创建一个HTML文件UploadPic.html，用于用户选择上传的文件，其完整的代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>uploadify文件上传插件</title>
    <script type="text/javascript"
            src="Js/fileUpload.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css/uploadify.css" />

     <style type="text/css">

     </style>
</head>
<body>
   <div class="divFrame">
        <div class="divTitle">
              上传图片
        </div>
        <div class="divContent">
             <ul></ul>
             <div id="fileQueue"
                       style="clear:both;padding-top:5px"></div>
             <div style="padding-top:5px">
                       <input type="file" name="uploadify"
                              id="uploadify" />
                  </div>
        </div>
   </div>
</body>
</html>
```

另外，新建一个JS文件fileUpload.js，用于实现页面中的文件上传和预览，以及删除上传后的文件，其完整代码如下：

```
/// <reference path="../Jscript/jquery-1.8.2.min.js"/>
/// <reference path="../Js/swfobject.js"/>
/// <reference path="../Js/jquery.uploadify.v2.1.0.js"/>
$(function() {
    $("#uploadify").uploadify({
        'uploader': 'Images/uploadify.swf',
        'script': 'Deal/UploadFile.ashx',
        'cancelImg': 'Images/cancel.png',
        'folder': 'Uploads/',
        'queueID': 'fileQueue',
        'buttonImg': 'Images/uploadify.jpg',
        'auto': true,
        'multi': true,
        'fileExt': '*.jpg;*.jpeg;*.gif;*.png',
        'onComplete': function(event, queueID, fileObj, response, data) {
            $('ul').append(SetFileContent(fileObj));
            SetUploadFile();
        }
    })
})
function SetFileContent(objFile) { //根据上传对象返回预览的图片
    var sLi = "";
    sLi += "<li>";
    sLi += "<img src='" + objFile.filePath + "' width='100' height='100'>";
    sLi += "<input type='hidden' value='" + objFile.filePath + "'>";
    sLi += "<br />";
    sLi += "<a href='javascript:void(0)'>删除</a>";
    sLi += "</li>";
    return sLi;
}


 function SetUploadFile() {
     //设置各表项ID号
     $("ul li").each(function(l_i) {
         $(this).attr("id", "li_" + l_i);
     })
     //设置各链接的rel属性值
     $("ul li a").each(function(a_i) {
         $(this).attr("rel", a_i);
     }).click(function() {//设置各链接的单击事件
         //通过自身的rel值，获取表项的ID号
         var li_id = "#li_" + this.rel;
         //根据ID号，删除某个表项
         $(li_id).remove();
     })
 }
```

8.5 本章小结

本章首先介绍如何调用jQuery中的插件，然后通过示例与插件文件相结合的方式，详细介绍了目前比较常用插件的使用方法，以及如何自定义编写对象和类级别插件，最后通过一个完整的案例介绍上传文件插件uploadify的详细使用方法。插件在jQuery中占据重要位置，因此下一章节我们将继续介绍另外一种全新的jQuery UI插件。

第9章　jQuery UI插件

本章内容

认识jQuery UI

jQuery UI交互性插件

jQuery UI微型插件

jQuery UI 1.9新增功能

综合案例分析——使用jQuery UI插件以拖动方式管理相册

本章小结

jQuery UI是一个以jQuery为基础的用户体验代码库，它的本质源于一个名为interface的jQuery插件，后来对该插件内部的API进行重构，并升级了版本，重新取名为jQuery UI。由于jQuery库注重于后台，没有很好的前台界面，而jQuery UI则补充了其不足之处，两者相互交织。随着jQuery技术的发展壮大，也同时确定了jQuery UI作为jQuery官方插件的地位。

9.1 认识jQuery UI

jQuery UI注重于用户界面的体验，根据其体验角度的不同，主要分为以下三个部分：

❑交互（Interactions）：在该部分中，展示一些与鼠标操作相关的插件内容，如拖动（Draggable）、放置（Droppable）、缩放（Resizable）、复选（Selectable）、排序（Sortable）等。

❑微件（Widgets）：该部分包括一些可视化的细小控件，通过这些小控件，可以极大优化用户在页面中的体验度，如折叠面板（Accordion）、日历（Datepicker）、对话框（Dialog）、进度条（Progressbar）、滑动模块（Slider）、标签页（Tabs）等。

❑效果或动画（Effects）：该部分包括一些动画效果插件，使我们的动画不再拘泥于animate()方法，可以通过该部分的插件，实现一些复杂的动画效果。在该部分中，改进后的动画方法如show()、hide()、toggle()等。

目前，jQuery UI的最新版本可以在地址http://jqueryui.com/download的页面中进行下载，其下载界面如图9-1所示。

图　9-1　jQuery UI官方下载页面

如该页面右侧所示，在版本（Version）处选中单选按钮“1.9.2”，然后单击"Download"按钮，就可以下载最新的jQuery UI库jquery-ui-1.9.2.custom.rar压缩包文件。该压缩包中，包括最新的jQuery UI库文件和demos、themes、docs等文件包，用户可以选择性解压缩。

本章所介绍的示例基于jQuery UI版本1.9.2和jQuery版本1.8.2。

9.2　jQuery UI交互性插件

9.2.1　拖曳插件draggable

draggable插件能使请求的对象拖动，通过这个插件可以使用DOM元素跟随鼠标进行移动，通过设置方法中的option选项，实现各种各样的拖动需求，其调用的语法格式为：

```
draggable(options)
```

其中选项options接受各种各样的参数值，用于控制拖动时的页面效果。其常用的参数值如表9-1所示。

表 9-1　选项 options 可接受的常用参数

参数名称	功能描述
helper	表示被拖曳的对象，默认值为 original，即拖动自身；如果设置为 clone，那么，以复制的形式进行拖动
handle	表示触发拖曳的对象，常用一个 DOM 元素
dragPrevention	设置不触发拖曳的元素
start	当拖曳启动时触发的回调函数 function(e,ui)，其中参数 e 表示 event 事件，e.target 表示被拖曳的对象；参数 ui 表示与拖曳相关的对象
stop	停止拖曳是触发的回调函数，其中的参数说明与 start 中一样
drag	在拖动过程中触发的回调函数，其中的参数说明与 start 中一样
zIndex	设置被拖曳时，helper 对象的 z-index 值
axis	设置拖曳时的坐标，可以设为 "x" 或 "y" 值
containment	设置拖曳时的区域，可以设为 "document"、"parent" 和其他指定的元素和对象
grid	设置拖曳时的步长，如 grid:[50,60]，表示 x 坐标每次移动 50px，y 坐标每次移动 60px
opacity	设置对象在拖曳过程中的透明度，范围是（0.0~1.0）
revert	设置一个布尔值，如果为 true，表示对象被拖曳结束后，又会自动返回原地；如果为 false，则不会返回原地，默认值为 false
scroll	设置一个布尔值，如果为 true，则表示对象在拖曳时，容器自动滚动，默认为 true 值
disable	临时性禁用拖动功能
enable	重新开启对象的拖动功能
destroy	彻底移除对象上的拖动功能

下面通过一个简单的示例介绍DOM元素通过draggable插件实现拖曳的过程。

示例9-1　使用draggable插件实现对象的拖曳操作

（1）插件文件

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-Pub/jquery.ui.mouse.js

Js-9-1/jquery.ui.draggable.js

（2）功能描述

在页面中，创建三个可供拖曳的＜div＞块，分别实现透明度为0.35的任意拖曳和在x轴中任意拖曳及在父窗口中任意拖曳的操作后，返回原来的初始位置。

（3）实现代码

新建一个HTML文件9-1.html，加入如代码清单9-1所示的代码。

代码清单 9-1　使用 draggable 插件实现对象的拖曳操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 draggable 插件实现对象的拖曳操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-9-1/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-9-1/jquery.ui.mouse.js">
    </script>
    <script type="text/javaascript"
            src="Js-9-1/jquery.ui.draggable.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px}
           .divFrame{border:dashed 1px #ccc;background-color:#eee;
                    height:69px;width:200px}
           .divDrag{border:solid 1px #ccc;background-color:#eee;
                    cursor:move;width:40px;
                    padding:20px;text-align:center}
    </style>
    <script type="text/javascript">
        $(function() {
            // 以透明度 0.35 随意拖动
            $("#divDragD").draggable({ opacity: 0.35 });
            // 以透明度 0.35 随意在 X 轴方向拖动
            $("#divDragX").draggable({ axis: "x", opacity: 0.35 });
            // 以透明度 0.35 随意在父窗口中拖动，拖动后返回原地
            $("#divDragC").draggable(
            { containment: "parent",
                opacity: 0.35,
                revert: true
            });
        })
    </script>
</head>


 <body>
      <div id="divDragD" class="divDrag"> 随意 </div>
      <div class="divFrame">
          <div id="divDragX" class="divDrag">X 轴 </div>
      </div>
      <div class="divFrame">
          <div id="divDragC" class="divDrag"> 父窗口 </div>
      </div>
 </body>
 </html>
```

（4）页面效果

代码执行后的效果如图9-2所示。

图　9-2　使用draggable插件实现对象的拖曳操作

（5）代码分析

在示例9-2中，通过draggable插件，开发人员可以很方便地设置页面中的某个元素进行拖动的操作，如果要禁止拖动动作，只要加入如下代码：

```
$("div").draggable("diable");
```

禁止后，如果想开启元素的拖动操作，需要加入如下代码：

```
$("div").draggable("enable");
```

9.2.2　放置插件droppable

在jQuery UI中，除使用draggable插件进行拖曳对象外，还可以通过droppable插件“存放”拖曳的对象，即类似网上商城中购物车的效果，其调用的语法格式为：

```
droppable(options)
```

其中选options可接收的常用参数如表9-2所示。

表 9-2　选项 options 可接受的常用参数

参数名称	功能描述
accept	可以为字符串或函数：如果是字符串，表示通过字符串获取的元素允许接收；如果是函数，表示只有执行函数后，返回 true 时，才能允许接收
activeClass	被接收的对象在拖曳时，接收容器的 CSS 样式
hoverClass	被接收的对象在进入接收容器时，容器的 CSS 样式
active	被接收的对象在拖曳时，调用的函数 function(e,ui)。其中参数 e 表示 event 事件，e.target 表示被拖曳的对象，参数 ui 表示与拖曳相关的对象
deactive	被接收的对象停止拖曳时，调用的函数，其中的参数说明与 active 中一样
over	被接收的对象拖曳到接收容器上方时，调用的函数，其中的参数说明与 active 中一样
out	被接收的对象拖出接收容器时，调用的函数，其中的参数说明与 active 中一样
drop	被接收的对象拖曳后，完全进入接收容器时，调用的函数，其中的参数说明与 active 中一样

下面通过一个简单的示例介绍DOM元素通过droppable插件实现放置的过程。

示例9-2　使用droppable插件实现对象的放置操作

（1）插件文件

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-Pub/jquery.ui.mouse.js

Js-9-1/jquery.ui.draggable.js

Js-9-2/jquery.ui.droppable.js

（2）功能描述

在页面中，创建一个产品区与购物车区，通过使用droppable插件将产品区中的产品拖动并放入购物车区中，同时，改变购物车的背景色。

（3）实现代码

新建一个HTML文件9-2.html，加入如代码清单9-2所示的代码。

代码清单 9-2　使用 droppable 插件实现对象的置放操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 draggable 插件实现对象的放置操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.mouse.js">
    </script>
    <script type="text/javascript"
            src="Js-9-1/jquery.ui.draggable.js">
    </script>
    <script type="text/javascript"
            src="Js-9-2/jquery.ui.droppable.js">
    </script>
    <style type="text/css">
         body{font-size:13px}
         #divFrame{width:400px}
         .divLeft{float:left;border:solid 1px #666;width:100px}
         .divRight{float:right;border:solid 1px #666;width:200px}
         .divTitle{background-color:#ccc;padding:5px}
         .divDrag{padding:5px;cursor:move}
         .divGet{background-color:#eee}
         #divCart div{padding:5px;height:85px;font-size:11px}
    </style>
```

```
    <script type="text/javascript">
        $(function() {
            // 以透明度 0.35 随意拖动
            $(".divDrag").draggable({ opacity: 0.35 });
            $("#divCart").droppable({
                drop: function() {
                    $(this)
                .addClass("divGet") // 改变购物车的 CSS
                .append($("<div></div>"))
                .find("#divTip").remove(); // 删除原有内容
                }
            })
        })
    </script>
</head>
<body>
    <div id="divFrame">
        <div class="divLeft">
            <div class="divTitle">产品</div>
            <div class="divDrag">
                <img alt="" src="Images-9-2/img03.jpg" />
            </div>
        </div>
        <div class="divRight">
            <div class="divTitle">购物车</div>
            <div id="divCart">
                <div id="divTip">还没有产品</div>
            </div>
        </div>
    </div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图9-3所示。

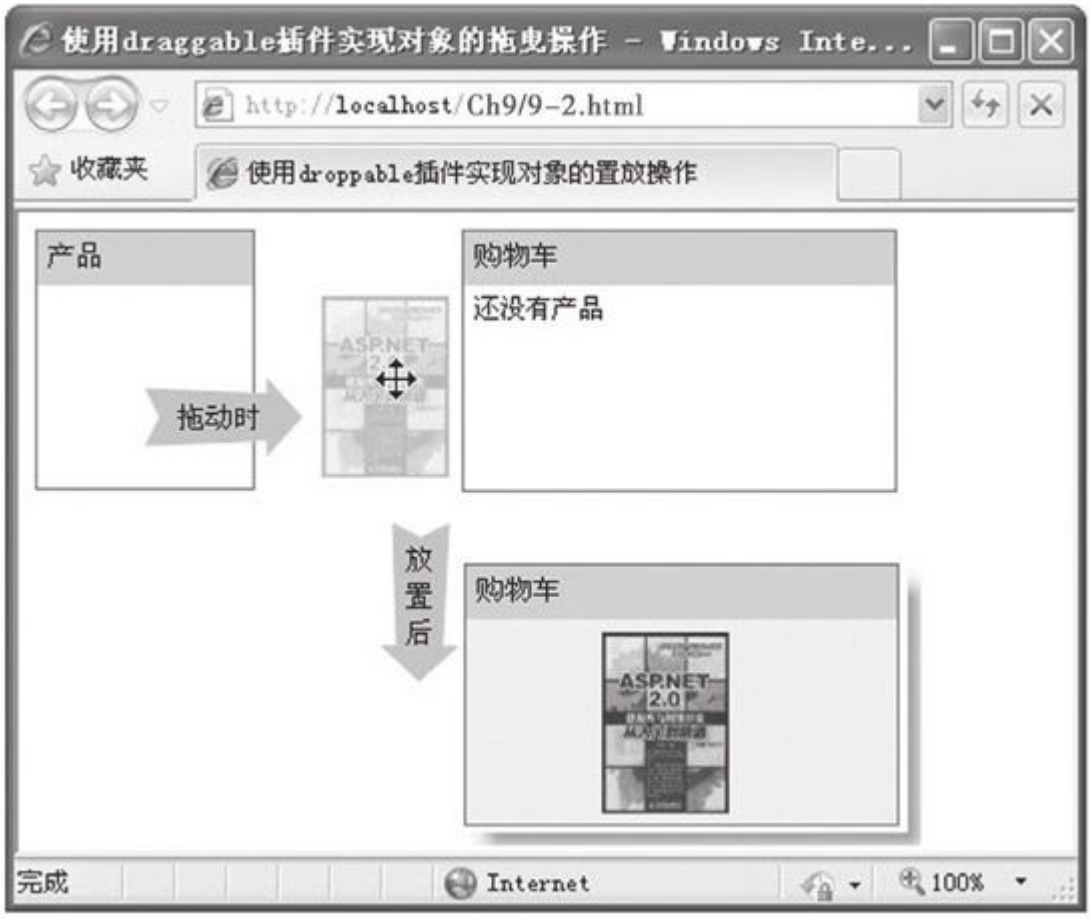


图　9-3　使用droppable插件实现对象的放置操作

（5）代码分析

在示例9-2中，产品从产品区拖至购物车时，通过find("#divTip").remove()代码将购物车中原有提示信息消失，通过append($("＜div＞＜/div＞"))代码将产品放入购物车中。另外，通过addClass("divGet")代码改变购物车中的背景色。

9.2.3　排序插件sortable

在jQuery UI中，除拖曳、放置指定元素外，还可以通过sortable插件将有序列的标记，按照用户自己的想法任意拖曳其所在位置，形成一个新的序列，从而实现拖曳排序的功能。该插件的调用语法格式如下：

```
sortable(options)
```

其中，选项options所调用的参数与插件draggable中的options参数有很多相似之处。需要说明的是参数item，该参数用于申明在页面中哪些元素以拖曳的方式进行排序。如下列代码：

```
$(element).sortable("option", "items", 'li' );
```

以上代码说明，在页面中，有＜option＞、＜items＞、＜li＞标记的元素可以通过拖曳的方式进行排序，如果不声明，则默认全部的表项型元素都可以。

下面通过一个简单的示例演示在列表中，通过sortable插件实现表项拖曳排序的过程。

示例9-3　使用sortable插件实现列表中表项的拖曳排序操作

（1）插件文件

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-Pub/jquery.ui.mouse.js

Js-9-3/jquery.ui.sortable.js

（2）功能描述

在页面中，创建1个列表＜ul＞，并在列表中增加5个表项＜li＞标记，通过jQuery UI中的排序插件sortable，实现每个表项拖曳排序的功能。

（3）实现代码

新建一个HTML文件9-3.html，加入如代码清单9-3所示的代码。

代码清单 9-3　使用 sortable 插件实现列表中表项的拖曳排序操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 sortable 插件实现列表中表项的拖曳排序操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
```

```
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.mouse.js">
    </script>
    <script type="text/javascript"
            src="Js-9-3/jquery.ui.sortable.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           ul{padding:0px;margin:0px;
           list-style-type:none;width:260px}
           ul li{margin: 0 3px 3px 3px;border:solid 1px #ccc;
                 background-color:#eee;padding:0.4em;cursor:move;
                 font-size: 1.4em;height: 19px;}
    </style>
    <script type="text/javascript">
        $(function() {
            $("ul").sortable(
            { delay: 2, // 为防与点击事件冲突，延时 2 秒
            opacity: 0.35 // 以透明度 0.35 随意拖动
            })
            });
    </script>
</head>
<body>
       <ul>
          <li>Item 1</li>
          <li>Item 2</li>
          <li>Item 3</li>
          <li>Item 4</li>
          <li>Item 5</li>
       </ul>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图9-4所示。

图　9-4　使用sortable插件实现列表中表项的拖曳排序操作

（5）代码分析

在示例9-3的JS代码中，仅是通过sortable插件在页面中展示拖曳排序的操作，而实际的数据并没有进行保存，如果要保存页面中拖曳后的排列顺序，可以在sortable插件的选项options中设置一个stop参数，该参数执行一个回调函数，通过$.post方法与后台代码进行关联，相关的代码如下：

```
stop: function() {
        $.post("后台文件",
        $("ul").sortable("serialize"),
```

```
            function(data) {
                alert(data);
            });
    }
```

在执行上述代码前，先给列表＜ul＞中的各个表项＜li＞设置相应的ID号，那么，通过代码$("ul").sortable("serialize")就可以获取各ID号排列的顺序，将这些顺序数据发送给后台，后台文件根据获取的顺序进行保存，从而真正实现表项的拖曳排序。

9.3 jQuery UI微型插件

9.3.1 折叠面板插件accordion

jQuery UI插件accordion可以实现页面中指定区域的折叠效果，这种效果俗语“手风琴”，即通过单击某块面板中的标题栏，就会展开相应的内容，单击其他面板标题栏时，已展开的内容会自动关闭，通过这种方式，实现多个面板数据在一个页面中有序展示。其调用的语法格式为：

```
accordion(options)
```

其中选项options常用的参数如表9-3所示。

表 9-3 选项 options 中的常用参数

参数名称	功能描述
animated	设置折叠时的效果，默认为“slide”；也可以自定义动画。如果设置为 false，表示不要设置折叠时的动画效果
active	设置默认展开的主题选择，默认值为“1”
autoHeight	内容高度是否设置为自动增高，默认值为“true”
event	设置展开选项的事件，默认值为“click”，也可以设置双击、鼠标滑过事件
fillSpace	设置内容是否充满父元素的高度，默认值为“false”，如果设置为“true”，autoHeight 参数设置的值无效
icon	设置小图标，其设置的格式为{"header":"主题默认图标类别名","headerSelected":"主题选中时图标类别名"}

下面通过一个简单的示例演示在页面中创建多个相同的区域，实现按主题折叠的效果。

示例9-4　使用accordion插件实现页面中多区域的折叠操作

（1）插件文件

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-9-4/jquery.ui.accordion.js

（2）功能描述

在页面中，通过accordion插件展示几个相同区域的折叠效果。

（3）实现代码

新建一个HTML文件9-4.html，加入如代码清单9-4所示的代码。

代码清单 9-4　使用 accordion 插件实现页面中多区域的折叠操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 accordion 插件实现区域块的折叠操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-9-4/jquery.ui.accordion.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:260px}
           .divFrame h3{padding:5px;font-size:12px;
           border:solid 1px #ccc;background-color:#eee}
           .divFrame .divOpt{padding:5px;border:solid 1px #ccc}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#accordion").accordion();
        })
    </script>
</head>
<body>
  <div class="divFrame">
   <div id="accordion">
   <h3><a href="#">Section 1</a></h3>
   <div class="divOpt">
           <p>


             Item1
           </p>
   </div>
   <h3><a href="#">Section 2</a></h3>
   <div class="divOpt">
           <p>
             Item2
           </p>
   </div>
   <h3><a href="#">Section 3</a></h3>
   <div class="divOpt">
           <p>
             Item3
           </p>
   </div>
   <h3><a href="#">Section 4</a></h3>
   <div class="divOpt">
           <p>
             Item4
           </p>
   </div>
</div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图9-5所示。

图　9-5　使用accordion插件页面中多区域的折叠操作

（5）代码分析

如果accordion插件在调用后，还需要获取或设置选项options中的值，可以通过下列方法进行。如获取初始化后的active值，代码如下所示：

```
var active = $("#accordion").accordion("option", "active" );
```

设置的代码如下：

```
var active = $("#accordion").accordion("option", "active",2);
```

其他参数在accordion插件初始化后的获取与设置与上述方法基本一样。

9.3.2　日历插件datepicker

在页面开发中，经常遇到需要用户输入日期的操作。通常的做法是，提供一个文本框（text），让用户输入，然后，编写代码验证输入的数据，检测其是否是日期型。这样比较麻烦，同时，用户输入日期的操作也不是很方便，影响用户体验。如果使用jQuery UI中的datepicker插件，这些问题都可以迎刃而解。该插件调用的语法格式如下：

```
$(".selector").datepicker(options);
```

其中".selector"表示DOM元素，一般指文本框，由于该插件的作用是提供日期选择，因此，常与一个文本框绑定，将选择后的日期显示在该文本框中。

选项options是一个对象，与前面章节中插件的options一样，通过改变其参数对应的值，从而实现插件功能的变化，在datepicker插件中，选择options常用参数如表9-4所示。

表 9-4　选项 options 中的常用参数

参数名称	功能描述
changeMonth	设置一个布尔值，如果为 true，则可以在标题处出现一个下拉选择框，可以选择月份，默认值为 false
changeYear	设置一个布尔值，如果为 true，则可以在标题处出现一个下拉选择框，可以选择年份，默认值为 false
showButtonPanel	设置一个布尔值，如果为 true，则在日期的下面显示一个面板，其中有两个按钮：一个为“今天”，另一个按钮为“关闭”，默认值为 false，表示不显示
closeText	设置关闭按钮上的文字信息，这项设置的前提是，showButtonPanel 的值必须为 true，否则显示不了效果
dateFormat	设置显示在文本框（text）中的日期格式，可设置为 { dateFormat: 'yy-mm-dd' }，表示日期的格式为年 - 月 - 日，如 2012-10-1
defaultDate	设置一个默认日期值，如 {defaultDate: +7}，表示弹出日期选择窗口后，默认的日期是在当前日期加上 7 天
showAnim	设置显示弹出或隐藏日期选择窗口的方式，可以设置的方式有“show”、“slideDown”、“fadeIn”，或者为“”，表示没有弹出日期选择窗口的方式
showWeek	设置一个布尔值，如果为 true，则可以显示每天对应的星期，默认值为 false
yearRange	设置年份的范围，如 {yearRange: '2000:2010'}，表示年份下拉列表框的最小值为 2000，最大值为 2010，默认值为 c-10:c+10，当前年份的前后 10 年

下面通过一个示例，展示datepicker插件在页面中的基本用法。

示例9-5　使用datepicker插件实现日期选择的基本操作

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-9-5/jquery.ui.datepicker.js

Css-Pub/demos.css

（2）功能描述

在页面中单击文本框时，通过datepicker插件弹出一个日期选择窗口，该窗口可以使用下拉列表框的方式选择年份与月份，并显示与日期相对应的星期，并且可以单击面板中的“关闭”按钮，关闭弹出的窗口。

（3）实现代码

新建一个HTML文件9-5.html，加入如代码清单9-5所示的代码。

代码清单 9-5　使用 datepicker 插件实现日期选择的基本操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 datepicker 插件实现选择日期的操作一</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link rel="stylesheet" type="text/css"
          href="Css-Pub/ui.all.css" />
    <script type="text/javascript"
          src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
          src="Js-9-5/jquery.ui.datepicker.js">
   </script>
    <link rel="stylesheet" type="text/css"
          href="Css-Pub/demos.css" />
    <style type="text/css">
          .txt{border:#666 1px solid;
          padding:2px;width:100px;margin-right:3px}
    </style>
    <script type="text/javascript">
       $(function() {
           $("#txtDate").datepicker(
          { changeMonth: true,      //显示下拉列表月份
            changeYear: true,       //显示下拉列表年份
            showWeek: true,         //显示日期对应的星期
            showButtonPanel: true, //显示"关闭"按钮面板
            closeText: 'Close'      //设置关闭按钮的文本
          });
       })
   </script>
</head>
<body>
<div class="demo-description">
    <div>
        输入日期: <input name="txtDate" id="txtDate" class="txt" />
    </div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图9-6所示。

图　9-6　使用datepicker插件实现选择日期的操作

示例9-5展示了datepicker插件基本的用法，但该插件的功能远不止这些，例如，在页面中，有时需要分时间段查询，这时，要求用户输入开始与结束时间，并且结束时间必须大于或等于开始时间。如果使用传统的JavaScript的方法编写，代码相对复杂；而使用datepicker插件则可以轻松选择两个符合条件的时间。下面通过一个示例介绍该功能实现的过程。

示例9-6　使用datepicker插件实现分段时间的选择

（1）功能描述

在页面中设置两个文本框（text），通过datepicker插件分别获取开始时间与结束时间。在选择开始时间后，再选择结束时间时，结束时间在大于或等于开始时间中选择。

（2）实现代码

新建一个HTML文件9-6.html，加入如代码清单9-6所示的代码。

代码清单 9-6　使用 datepicker 插件实现分段时间的选择

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>使用 datepicker 插件实现选择日期的操作二 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link rel="stylesheet" type="text/css"
           href="Css-Pub/ui.all.css" />
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.core.js">
   </script>
   <script type="text/javascript"
           src="Js-9-5/jquery.ui.datepicker.js">
   </script>
   <script type="text/javascript"
           src="Js-9-6/jquery.ui.datepicker-zh-CN.js">
```

```
    </script>
     <link rel="stylesheet" type="text/css"
            href="Css-Pub/demos.css" />
     <style type="text/css">
            .txt{border:#666 1px solid;padding:2px;
            width:100px;margin-right:3px}
     </style>
    <script type="text/javascript">
        $(function() {
            $("#txtStart").datepicker( //绑定开始日期
             { changeMonth: true,       //显示下拉列表月份
               changeYear: true,        //显示下拉列表年份
               showWeek: true,          //显示日期对应的星期
               firstDay: "1",
               onSelect: function(dateText, inst) {
                //设置结束日期的最小日期
                $('#txtEnd').datepicker('option',
                'minDate', new Date(dateText.replace('-', ',')))
               }
            })

            $("#txtEnd").datepicker( //绑定结束日期
             { changeMonth: true,       //显示下拉列表月份
               changeYear: true,      //显示下拉列表年份
               showWeek: true,        //显示日期对应的星期
               firstDay: "1",
               onSelect: function(dateText, inst) {
                //设置开始日期的最大日期
                $('#txtStart').datepicker('option',
                'maxDate', new Date(dateText.replace('-', ',')))
               }
            })
        })
    </script>
</head>
<body>
<div class="demo-description">
     <div>
        开始日期: <input name="txtStart" id="txtStart" class="txt" />
        结束日期: <input name="txtEnd" id="txtEnd" class="txt" />
     </div>
</div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图9-7所示。

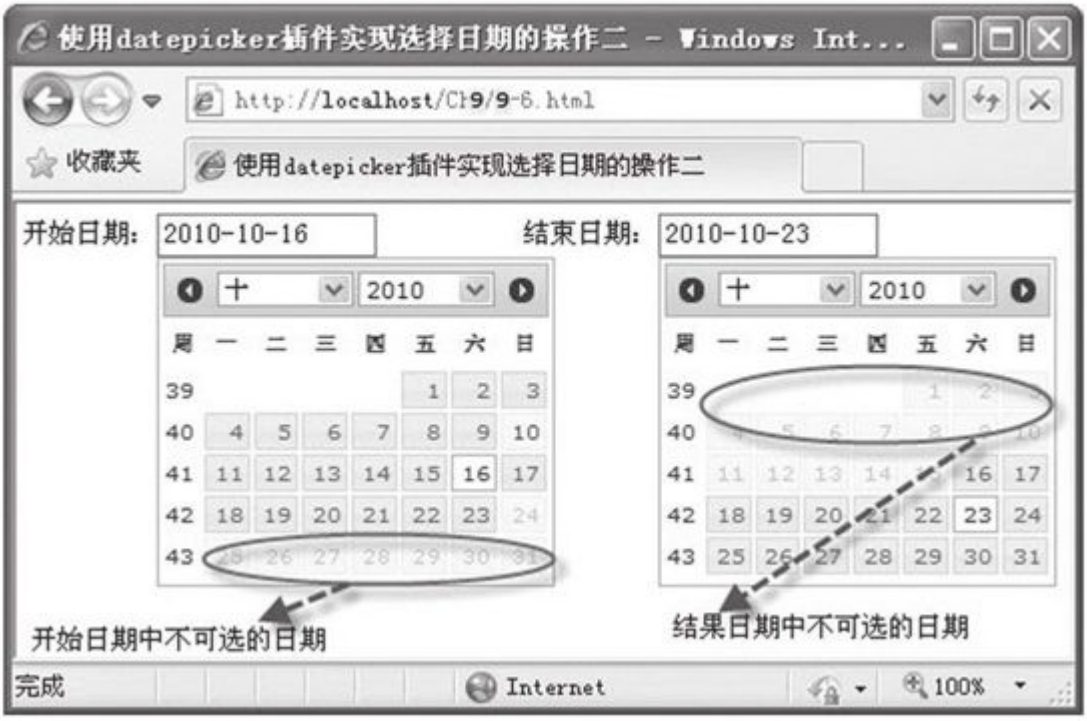


图　9-7　使用datepicker插件实现分段时间的选择

（4）代码分析

从图9-7中我们可以看出，通过对datepicker插件中onSelect事件的设置，使选择开始时间时，改变了结束日期的最小日期的选择范围。同理，在选择结束日期时，改变了开始日期的最大日期的选择范围，从而有效地规避了时间段中不规范数据的出现。

另外，为了更好地展示datepicker插件弹出日期选择窗口的原始效果，在页面中需引用与该插件配套的两个官方提供的样式文件，代码如下：

```
<link rel="stylesheet" type="text/css" href="Css-Pub/ui.all.css" />
<link rel="stylesheet" type="text/css" href="Css-Pub/demos.css" />
```

同时，为了使弹出的日期选择窗口支持中文，需要在页面再引入一个JS文件，该文件将插件中的英文部分翻译成中文，其实现的代码如下：

```
<script type="text/javascript"
            src="Js-9-6/jquery.ui.datepicker-zh-CN.js">
```

导入JS文件jquery.ui.datepicker-zh-CN.js后，在再次弹出的日期选择窗口中，所显示的英文字符全部转换成了中文，其示意图如图9-7所示。

9.3.3　选项卡插件tabs

tabs在页面中的使用非常广泛，尤其是各大门户网站的首页，因为以选项卡的形式可以实现使用少量的空间展示更多内容的效果，同时，其快速切换的效果，也增加了用户的体验。在jQuery UI中，通过在页面中导入tabs插件，并调用插件中的tabs()方法直接针对列表生成对应菜单，轻松实现这种选项卡的功能。其调用的语法格式如下：

```
tabs(options)
```

其中选项options的常用参数如表9-5所示。

表 9-5　选项 options 中的常用参数

参数名称	功能描述
collapsible	是否可折叠选项卡的内容，设置一个布尔值，如果为 true，允许用户可折叠选卡的内容，即首次单击展开，再单击关闭，默认值为 false
disabled	设置不可用选项卡，如 {disabled: [1, 2] }，表示选项卡中，第 1 项和第 2 项不可用
event	设置触发切换选项卡的事件，默认值为 “click”，也可以设置为 “mousemove”
fx	设置切换选项卡时的一些动画效果
selected	设置被选中选项卡的 Index，如 {selected: 2}，表示第 2 项选项卡被选中

下面通过一个示例展示tabs插件在页面中的基本用法。

示例9-7　使用tabs插件展示选项卡的基本功能

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-9-7/jquery.ui.tabs.js

Css-Pub/demos.css

（2）功能描述

在页面中，创建1个列表＜ul＞标记，并在＜ul＞中创建3个＜li＞标记，通过tabs插件中的tabs方法使这3个＜li＞标记形成选项卡。其中，前两个选项卡对应指定页面中的＜div＞内容，后一个选项通过Ajax方式动态加载一个HTML页面内容，同时，在变换选项卡时展示相应的切换效果。

（3）实现代码

新建一个HTML文件9-7.html，加入如代码清单9-7所示的代码。

代码清单 9-7　使用 tabs 插件展示选项卡的基本功能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>使用 tabs 插件展示选项卡的基本功能 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link rel="stylesheet" type="text/css"
           href="Css-Pub/ui.all.css" />
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.core.js">
   </script>
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.widget.js">
   </script>
   <script type="text/javascript"
           src="Js-9-7/jquery.ui.tabs.js">
   </script>
    <link rel="stylesheet" type="text/css"
           href="Css-Pub/demos.css" />
   <script type="text/javascript">
       $(function() {
           $("#tabs").tabs({
               // 设置各选项卡在切换时的动画效果

               fx: { opacity: "toggle", height: "toggle" },
               // 通过移动鼠标事件切换选项卡
               event: "mousemove"
           })
       })
   </script>
</head>
<body>
<div class="demo-description">
     <div id="tabs" style="width:360px">
      <ul>
           <li><a href="#tabs-1">One</a></li>
           <li><a href="#tabs-2">Two</a></li>
           <!-- 第三个选项卡的内容为 Ajax 方法获取的静态页 -->
           <li><a href="Ajax-9-7/Content3.html">Three</a></li>
   </ul>
   <div id="tabs-1">
           <p>One 选项卡中的内容 </p>
   </div>
   <div id="tabs-2">
           <p>Two 选项卡中的内容 </p>
   </div>
</div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图9-8所示。

图　9-8　使用tabs插件展示选项卡的基本功能

在示例9-7中，第三个选项卡的内容通过Ajax方式调用一个HTML页面content3.html获取，该页面的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <p>Three 选项卡中的内容</p>
</body>
</html>
```

（5）代码分析

在上述示例9-7中，除直接通过超级链接的方式加载选项卡的内容外，还可以调用tabs插件中的url()方法，改变Ajax方式加载的页面内容。如下列代码：

```
$("#tabs").tabs("url", 2, "9-1.html");
```

执行上述代码后，第三个选项卡的内容在使用Ajax方式加载时，已经发生了变化，不再指向页面content3.html，而是指向HTML页面9-1.html。

另外，可以通过下列代码，增加或删除一个选项卡，代码如下：

```
//在选项卡后面，动态增加一个标签为"four"的选项卡，内容指向"9-1.html"
$("#tabs").tabs('add', "9-1.html", "four");
//删除选卡中的第三个选项卡，其中"2"为索引号，索引号从0开始
$("#tabs").tabs('remove', 2);
```

9.3.4　对话框插件dialog

在页面开发过程中，经常要与用户进行交互，例如在提交表单时，如果文本框（text）中内容为空，需要提醒用户输入内容，一般的做法是使用传统的JavaScript语言中的alert()函数弹出一个信息窗口；另外，在删除某项记录时，也需要告知用户确定，可用JavaScript语言中的confirm()函数，虽然这两个函数都可以实现相应的功能，但没有动画效果，功能单一，用户体验效果差。在jQuery UI中，通过dialog插件，不仅完成可以实现传统JavaScript语言中alert()函数与confirm()函数的功能，而且界面优雅，功能丰富，操作简便。该插件导入页面后，其调用语法格式代码如下：

```
$(".selector").dialog(options)
```

其中".selector"表示DOM元素，一般指定一个＜div＞标记，用于显示弹出对话框的内容和设置的按钮，选项options是一个对象，它常用的参数如表9-6所示。

表 9-6　选项 options 中的常用参数

参数名称	功能描述
autoOpen	设置一个布尔值，如果为 false，不显示对话框，默认值为 true
bgiframe	设置一个布尔值，如果为 true，表示如果在 IE 6 下，弹出的对话框可以遮盖住页面中类似于 <select> 标记的下拉列表框，默认值为 false
buttons	设置对话框中的按钮，如 { "buttons", { "Ok": function() { $(this).dialog("close"); }}} 表示设置一个文本内容为 "Ok" 的按钮，单击该按钮将关闭对话框

（续）

参数名称	功能描述
closeOnEscape	设置一个布尔值，如果为 false，表示不使用 ESC 快捷键的方式关闭对话框，默认值为 true
draggable	设置一个布尔值，表示是否可以拖动对话框，默认值为 true
hide	设置对话框关闭时的动画效果，可以设置为“slide”等各种动画效果，默认值为 null
modal	设置对话框是否以模式的方式显示，模式指的是页面背景变灰，不允许操作，焦点锁定对话框的效果，默认值为 false
position	设置对话框弹出时在页面中的位置，可以设置为“top”、“center”、“bottom”、“left”、“right”，默认值为 center
show	设置对话框显示时的动画效果，相关说明与 hide 参数一样
title	设置对话框中主题部分的文字，如“系统提示”，默认值为空

下面通过一个示例，展示dialog插件在页面中弹出提示信息对话框与确定信息对话框的过程。

示例9-8　使用dialog插件弹出提示和确定信息对话框

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-Pub/jquery.ui.mouse.js

Js-9-1/jquery.ui.draggable.js

Js-Pub/jquery.ui.position.js

Js-9-8/jquery.ui.dialog.js

Css-Pub/demos.css

（2）功能描述

在页面中，设置一个文本框和两个按钮，同时，以＜span＞标记显示一条记录；单击第一个“提交”按钮时，将检测文本框的内容是否为空，如果为空，则通过dialog插件弹出一个对话框，提示文本框内容不能为空；单击第二个“删除”按钮时，先通过dialog插件弹出一个确定对话框，当用户单击“确定”按钮时，将删除页面中＜span＞标记的内容。

（3）实现代码

新建一个HTML文件9-8.html，加入如代码清单9-8所示的代码。

代码清单 9-8　使用 dialog 插件弹出提示和确定信息对话框

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>使用 dialog 插件弹出提示和确定信息对话框 </title>
   <script type="text/javascript"
           src="Jscript/jquery-1.8.2.min.js">
```

```
</script>
<link rel="stylesheet" type="text/css"
        href="Css-Pub/ui.all.css" />
<script type="text/javascript"
        src="Js-Pub/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.widget.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.mouse.js">
</script>
<script type="text/javascript"
        src="Js-9-1/jquery.ui.draggable.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.position.js">
</script>
<script type="text/javascript"
        src="Js-9-8/jquery.ui.dialog.js">
</script>
<link rel="stylesheet" type="text/css"
       href="Css-Pub/demos.css" />
<style type="text/css">
        div{line-height:1.8em}
        .txt{border:#666 1px solid;padding:2px;width:190px;margin-right:3px}
        button,.btn {border:#666 1px solid;padding:2px;
        width:60px;filter: progid:DXImageTransform.Microsoft
        .Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D9);}
</style>
<script type="text/javascript">
    $(function() {
        //检测按钮事件
        $("#btnSubmit").bind("click", function() {
            if ($("#txtName").val() == "") { //如果文本框为空
                sys_Alert("姓名不能为空! 请输入姓名");
            }
        });
        //询问按钮事件
        $("#btnDelete").bind("click", function() {
            if ($("#spnName").html() !=null) { //如果对象不为空
                sys_Confirm("您真的要删除该条记录吗? ");
                return false;
            }
        });
    });
    function sys_Alert(content) { //弹出提示信息窗口
        $("#dialog-modal").dialog({
            height: 140,
            modal: true,
            title: '系统提示',
            hide: 'slide',
            buttons: {
```

```
                    Cancel: function() {
                        $(this).dialog("close");
                    }
                },
                open: function(event, ui) {
                    $(this).html("");
                    $(this).append("<p>" + content + "</p>");
                }
            });
        }
        function sys_Confirm(content) { //弹出询问信息窗口
            $("#dialog-modal").dialog({
                height: 140,
                modal: true,
                title: '系统提示',
                hide: 'slide',
                buttons: {
                    '确定': function() {
                        $("#spnName").remove();
                        $(this).dialog("close");
                    },
                    '取消': function() {
                        $(this).dialog("close");
                    }
                },
                open: function(event, ui) {
                    $(this).html("");
                    $(this).append("<p>" + content + "</p>");
                }
            });
        }
    </script>
</head>
<body>
    <div class="demo-description">
    <div style="background-color:#eee;padding:5px;width:260px">
        <input id="txtName" type="text" class="txt" />
        <input id="btnSubmit" type="button" value="提交" class="btn" />
    </div>
    <div style="padding:5px;width:260px">
        <span id="spnName">张三</span>
        <input id="btnDelete" type="button" value="删除" class="btn" />
    </div>
    <div id='dialog-modal'></div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后，单击“提交”按钮时，如果文本框中的内容为空，将弹出一个带模式的提示对话框，其实现的页面效果如图9-9所示。

图　9-9　显示带模式的提示对话框

在示例9-8执行的页面中，如果单击“删除”按钮，将出现一个带模式的询问对话框，当单击对话框中的“确定”按钮后，才能将页面中的＜span＞标记内容删除，页面效果如图9-10所示。

图　9-10　显示带模式的询问对话框

（5）代码分析

dialog插件实现的对话框通常与一个＜div＞标记绑定，只要执行插件中的dialog()方法就可以弹出对话框，但在实际应用中，需要根据特定的条件，弹出不同内容与类型的对话框（如提示、询问），这时，需要将插件实现的弹出对话框的代码进行封装，即自定义一个函数，如下列代码：

```
function sys_Alert(content) { //弹出提示信息窗口
    $("#dialog-modal").dialog({
        height: 140,
        modal: true,
        title: '系统提示',

        hide: 'slide',
        buttons: {
            Cancel: function() {
                $(this).dialog("close");
            }
        },
        open: function(event, ui) {
            $(this).html("");
            $(this).append("<p>" + content + "</p>");
        }
    });
}
```

页面在交互过程中需要弹出对话框时，直接调用该函数即可。例如文本框内容为空时弹出提示信息，代码如下所示：

```
...省略部分代码
if ($("#txtName").val() == "") { //如果文本框为空
      sys_Alert("姓名不能为空！请输入姓名");
}
...省略部分代码
```

删除时的询问对话框也是先自定义一个函数，该函数执行时，通过dialog插件弹出一个询问对话框。页面在交互时，如果要弹出询问对话框，直接调用该函数即可。

9.4　jQuery UI 1.9新增功能

在2012年10月中旬，jQuery UI的1.9.0版本正式诞生。该版本在原来1.8.26的基础之上进行了大量的功能改进，其中包含多个API性能的改造，同时修复了近500个Bug，使整个UI插件在功能和性能上都有了极大的提升。此外，在2012年10月下旬和11月下旬又更新了两个子版本，截止到本书完稿时，jQuery UI的最新版本为1.9.2版。

在jQuery UI 1.9版本中，性能的提升仅是一方面，另一方面是在Widgets（微件）部分新增了3个功能强大的新部件，分别为：

❑菜单工具（Menu Widget）：该部件用于创建嵌套式菜单，包括内联式和弹出式菜单。使用方法十分简单，对于需要创建多级复杂的内联式菜单，该插件是一个不错的选择。

❑微调按钮（Spinner Widget）：该部件由一个文本框和两个用于向上和向下的按钮组成，通过点击两个按钮，可以调节文本框中的数值。同时，该部件还支持方向键修改文本框的值，除数值外，还支持各地区货币和日期的输入。

❑工具提示（Tooltip Widget）：该部分在元素聚焦或鼠标悬停时，显示元素的更多内容信息。该信息既可通过元素

的"title"或"href"属性静态获取，也可以采用Ajax方式异步远程取得，同时，在显示时支持HTML元素标记。

接下来，我们详细介绍jQuery UI 1.9版本中新增部件的使用方法。

9.4.1 菜单工具插件menu

menu插件可以通过＜u＞元素创建多级内联或弹出式菜单，支持通过键盘上的方向键控制菜单的滑动显示，允许为菜单中的各个选项添加图标，并且能通过调用格式中的options对象设置某选项功能失效。

该插件在使用时需要使用一些样式功能，因此，该插件必须与对应的CSS样式文件配套使用。其调用的语法格式为：

```
$( ".selector" ).menu(options);
```

其中，参数options是一个对象，它可以接受的常用参数值如表9-7所示。

表 9-7　选项 options 中的常用参数

参数名称	功能描述
disabled	定义菜单项是否不可用。如果该值为 true，表示不可用，则整体菜单变灰色，父菜单项可以点击，但与之对应的子菜单不显示，默认值为 false
icons	定义父菜单指向子菜单的图标。该值为 jQuery UI CSS 文件中的样式类别名称，默认值为 "{ submenu: "ui-icon-carat-1-e" }"，"submenu" 表示子菜单项目
menus	定义父菜单，子菜单的框架元素。该属性的默认值是 "ul"，也可以设置其他的元素，如 div，代码为 "{ menus: "div" }"
position	设置子菜单相对于父菜单的位置，该属性的默认值是 "{ my: "left top", at: "right top" }"
select	菜单选中事件。当一个菜单选项被选中时触发该事件，通过事件回调函数中的 ui 参数，可以获取当前被点击的选项内容，如 "ui.item.html()"
create	菜单创建事件。当菜单创建完成时触发该事件，当执行该事件的回调函数时，表明一个菜单已创建成功

接下来，通过一个简单的示例来演示menu插件在页面中的基本用法。

示例9-9　使用menu插件演示菜单的基本功能

（1）插件文件

Css/jquery-ui-1.9.2.custom.min.css

Jscript/jquery-ui.js

（2）功能描述

在本书的第4.7节介绍列表应用时，我们通过编写大量的JavaScript代码，自定义了一个列表中的导航菜单。而在本示例中，通过引入menu插件，再编写少量代码，同样实现列表中导航菜单的功能与页面效果。

（3）实现代码

新建一个HTML文件9-9.html，加入如代码清单9-9所示的代码。

代码清单 9-9　使用 menu 插件演示菜单的基本功能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>使用嵌套菜单展示多项列表 </title>
    <link href="Css/jquery-ui-1.9.2.custom.min.css"
          rel="stylesheet" type="text/css" />
    <style type="text/css">
        body{font-size:13px;}
        .ui-menu{width:200px;}
    </style>
    <script type="text/javascript"
         src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
         src="Jscript/jquery-ui.js">
    </script>
    <script>
        $(function() {
            $("#menu").menu();
        });
    </script>
</head>
<body>
    <ul id="menu">
        <li style="padding: 10px">
            <img alt="" src="Images/icon.gif" />
            <span>电脑数码类产品 </span></li>
        <li><a href="#">
            <span class="ui-icon ui-icon-note"></span>
            笔记本
            </a>
            <ul>
                <li><a href="#">笔记本 1</a></li>
                <li><a href="#">笔记本 2</a></li>
                <li><a href="#">笔记本 3</a></li>
                <li><a href="#">笔记本 4</a></li>
                <li><a href="#">笔记本 5</a></li>
            </ul>
        </li>
        <li><a href="#">
                <span class="ui-icon ui-icon-disk"></span>
                移动硬盘
                 </a>
            <ul>
                <li>
                <a href="#">
                     <span class="ui-icon ui-icon-disk"></span>
                     移动硬盘 1
                </a>
                </li>
                <li><a href="#">移动硬盘 2</a></li>
                <li><a href="#">移动硬盘 3</a></li>
                <li><a href="#">移动硬盘 4</a></li>
                <li><a href="#">移动硬盘 5</a></li>
            </ul>
        </li>
        <li><a href="#">电脑软件 </a>
```

```
            <ul>
                <li><a href="#">电脑软件1</a></li>
                <li><a href="#">电脑软件2</a></li>
                <li><a href="#">电脑软件3</a></li>
                <li><a href="#">电脑软件4</a></li>
                <li><a href="#">电脑软件5</a></li>
            </ul>
        </li>
        <li><a href="#">数码产品</a>
            <ul>
                <li><a href="#">数码产品1</a></li>
                <li><a href="#">数码产品2</a></li>
                <li><a href="#">数码产品3</a></li>
                <li><a href="#">数码产品4</a></li>
                <li><a href="#">数码产品5</a></li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

（4）页面效果

实现的页面效果如图9-11所示。

图　9-11　使用嵌套菜单展示多项列表

（5）代码分析

在本示例中，插件在使用时，有时需要使用样式文件中的某个类别名称，如设置父或子菜单选项的图标时，必须指定一个类别的名称，因此，在页面的＜head＞元素中，先导入与插件相对应的CSS文件，代码如下：

```
<link href="Css/jquery-ui-1.9.2.custom.min.css"
      rel="stylesheet" type="text/css" />
```

接下来，由于jQuery UI插件是建立在jQuery框架基础之上，因此，在页面的＜head＞元素中先导入jQuery基础框架文件，再加载jQuery UI插件文件，代码如下：

```
<script type="text/javascript"
          src="Jscript/jquery-1.8.2.min.js">
</script>
<script type="text/javascript"
         src="Jscript/jquery-ui.js">
</script>
```

最后，编写JavaScript代码，调用插件的menu()方法将插件与页面的元素相绑定，实现代码如下：

```
$("#menu").menu();
```

在上述代码中，"#menu"是一个ID号为"menu"的最外层＜ul＞框架元素，菜单UI插件不仅可以将嵌套的＜ul＞元素转成内联式菜单，只要具备父子关系的元素，都可以绑定菜单UI插件，转成各种弹出或内联式菜单，默认为＜ul＞元素，如果不是该类型的元素，则除调用菜单插件的menu()方法外，还需要在options对象中进行申明，代码如下：

```
$("#menu").menu({ menus: "div" });
```

此外，在菜单选项中，添加图标也十分方便，只需要在选项文本内容前增加一个＜span＞元素，在该元素中通过"class"属性选择不同的图标。

9.4.2 微调按钮插件spinner

在jQuery UI 1.9版本中，spinner是另一个新增加的部件型插件，该插件是一个增强型的文本输入框，不仅可以在文本框中直接输入数值，还可以通过点击输入框右侧的上下两个按钮修改输入框中的数值，此外，还支持通过键盘中的上下方向键改变输入框中的数值。

另外，该插件与其他本地化工具相结合后，还支持多语言环境下的货币和日期的输入，功能强大，且使用的方法十分简单，该插件的调用格式如下：

```
$( ".selector" ).spinner(options);
```

与绝大部分的jQuery UI插件相同，微调按钮插件的调用方法中，options是一个可选项对象，所有与该插件相关的属性、方法、事件都可以通过该对象进行设置，该对象常用的属性值如表9-8所示。

表 9-8 选项 options 中的常用参数

参数名称	功能描述
disabled	设置插件是否不可用。如果属性值为 true，表示不可用，整体插件变成灰色，不能进行任务的页面操作，默认值为 false
max	设置插件中输入框允许输入的最大值。如果该值存在，则所有输入的数值都不能大于该数值，如果大于，则自动变为最大值，默认值为 null
min	设置插件中输入框允许输入的最小值。如果该值存在，则所有输入的数值都不能小于该数值，如果小于，则自动变为最小值，默认值为 null

（续）

参数名称	功能描述
step	设置当点击插件微调按钮或上下方向键时，增加或减少的步长。默认值为 1，也可以修改默认属性值，如 '{ step: 2 }'，即将步长设置为 2
value	设置或获取插件当前的值，是插件的一个方法。'元素名 .spinner("value")' 表示获取插件值；'元素名 .spinner("value", 50)' 表示设置插件的值为 50
change	插件的值变化事件。当插件输入框的值已改变并且焦点不在输入框时触发该事件，点击微调按钮或方向键时不触发该事件，因为焦点还在输入框中
spin	插件的值改变事件。点击微调按钮或是方向键修改插件值时，都将触发该事件，在事件回调函数中，根据 ui.value 获取当前的插件值

接下来通过一个简单的示例来演示spinner插件在页面中的基本用法。

示例9-10　使用spinner按钮改变元素背景色

（1）插件文件

Css/jquery-ui-1.9.2.custom.min.css

Jscript/jquery-ui.js

（2）功能描述

在新建的页面中创建三个＜input＞元素，分别用于设置颜色中的“红色”（r）、“绿色”（g）、“蓝色”（b）；另外，新建一个＜p＞元素，用于展示微调按钮插件改变值时的颜色区。用户任意修改某个绑定颜色的＜input＞元素时，对应的颜色区背景色会随之发生变化，同时，颜色区下面显示与背景色对应的色彩值（RGB）。

（3）实现代码

新建一个HTML文件9-10.html，加入如代码清单9-10所示的代码。

代码清单 9-10　使用 spinner 按钮改变元素背景色

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用微调按钮改变元素背景色</title>
    <link href="Css/jquery-ui-1.9.2.custom.min.css"
          rel="stylesheet" type="text/css" />
    <style type="text/css">
        body{font-size:12px;}
        fieldset{padding:10px;width:280px;float:left;}
        #spnColor{width:150px;float:left;}
        #spnPrev{width:100px;height:70px;
                 border:solid 1px #ccc;float:right;}
        #pColor{font-weight:bold;clear:both;
                text-align:center;padding-top:5px}
    </style>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery-ui.js">
```

```
    </script>
    <script type="text/javascript">
        $(function() {
            //定义变量
            var intR = 0, intG = 0, intB = 0, strColor;
            $("input").each(function(index) {
                //初始化插件
                $(this).spinner({ max: 255, min: 0 });
                //设置微调按钮递增/递减事件
                $(this).spinner({
                    spin: function(event, ui) {
                        setSpnColor(index, ui.value);
                    },
                    //设置微调按钮值改变事件
                    change: function(event, ui) {
                        var intTmp = $(this).spinner("value");
                    if (intTmp < 0)
                        $(this).spinner("value", 0);
                    if (intTmp > 255)
                        $(this).spinner("value", 255);
                        setSpnColor(index,
                        $(this).spinner("value"));
                    }
                });
            });
            // 自定义函数改变元素背景色
            function setSpnColor(i, v) {
                switch (i) {
                    case 0:
                        intR = v;
                        break;
                    case 1:
                        intG = v;
                        break;
                    case 2:
                        intB = v;
                        break;
                }
                strColor = "rgb(" + intR + "," + intG + "," + intB + ")";
                $("#pColor").html(strColor);
                $("#spnPrev").css("background-color", strColor);
            }
        });
    </script>
</head>
<body>
    <form id="frmTmp">
    <fieldset>
        <legend>选择颜色值：</legend><span id="spnColor">
            <input /><input /><input />
        </span><span id="spnPrev"></span>
        <p id="pColor">
            rgb(0,0,0)</p>
    </fieldset>
    </form>
</body>
</html>
```

（4）页面效果

实现的页面效果如图9-12所示。

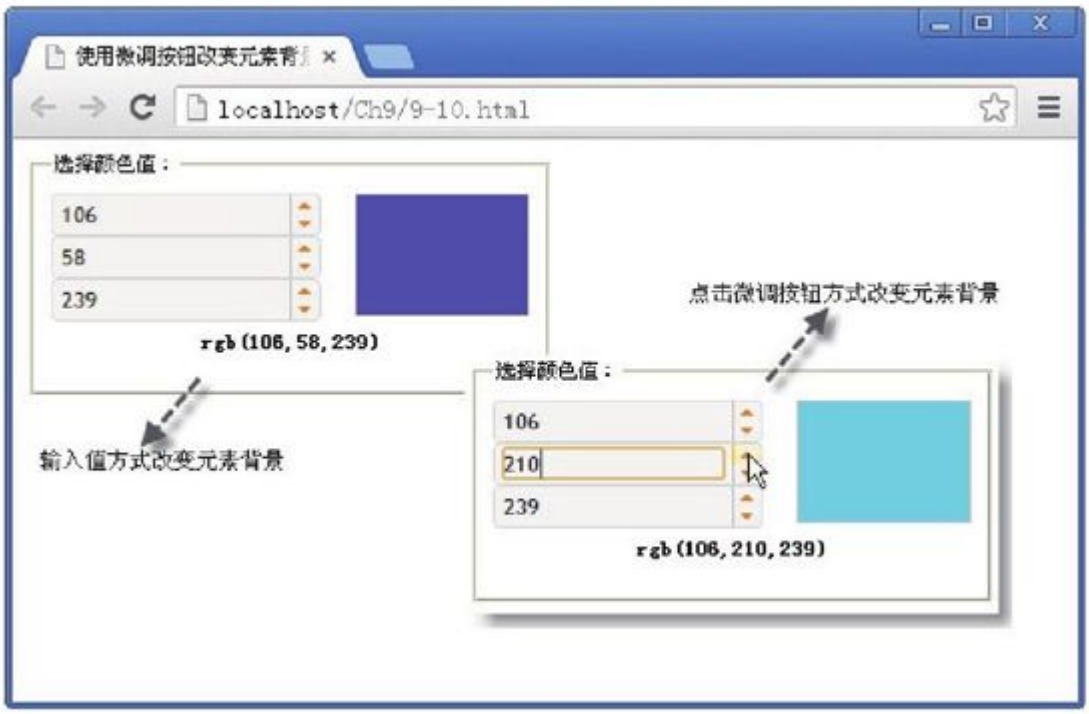


图　9-12　使用微调按钮改变元素背景色

（5）代码分析

在本示例中，先使用each()方法遍历全部的＜input＞元素。在遍历过程中，使用执行函数的index参数来标志当前元素的索引号，并假设索引为0值元素。改变红色值的微调按钮，索引号为1和2分别对应改变绿色和蓝色值的微调按钮。

在＜input＞元素遍历过程中，首先设置各颜色微调按钮的取值范围，代码如下：

```
$(this).spinner({ max: 255, min: 0 });
```

上述代码中，$(this)表示遍历过程中的当前元素，"{max:255,min:0}"设置微调按钮插件的最大值为255，最小值为0。在接下来的代码中，为了获取用户操作各个微调按钮插件时的值，需要设置各个微调按钮元素的spin和change事件。当用户点击上下微调按钮时触发spin事件，通过ui.value方式获取修改后的元素值，绑定代码如下：

```
spin: function(event, ui) {
      setSpnColor(index, ui.value);
}
```

当用户在输入框中手动录入数值，并离开输入框时触发change事件。该事件中，通过$(this).spinner("value")方式获取修改后的元素值，绑定代码如下：

```
change: function(event, ui) {
     var intTmp = $(this).spinner("value");
     if (intTmp < 0) $(this).spinner("value", 0);
     if (intTmp > 255) $(this).spinner("value", 255);
     setSpnColor(index, $(this).spinner("value"));
}
```

在上述代码中，虽然在开始遍历各＜input＞元素时，设置了当前微调按钮元素的取值范围，但对手动录入值时无效，因此，还要再进行一次取值范围的检测，如果不在范围内，则通过$(this).spinner("value",0)方式来修改当前元素的值。

最后，调用自定义函数setSpnColor动态改变元素的背景色。在该自定义函数中，定义了两个形参，分别为i和v，前者表示遍历元素时的索引号，用于指定控制RGB三个颜色的元素，后者表示该元素的当前值，代码如下：

```
switch (i) {
     case 0:
          intR = v;
          break;
     case 1:
          intG = v;
          break;
     case 2:
          intB = v;
          break;
     }
 strColor = "rgb(" + intR + "," + intG + "," + intB + ")";
 $("#pColor").html(strColor);
 $("#spnPrev").css("background-color", strColor);
```

在上述代码的函数体内，先根据i的值将元素的当前值分别保存在不同的变量中，然后形成一个RGB格式的字符串，并将该字符串内容保存在"strColor"变量中，最后通过html()和css()方法将该字符串分别显示和改变指定元素的背景色。

9.4.3　工具提示插件tooltip

在jQuery UI 1.9版本中，除上述章节中提及的两个新增的插件外，还有一个很重要的新增加插件——tooltip（工具提示），该插件也属于部件类型，它的核心功能是定制、主题化提示外观。与元素自带的"title"属性相比，该插件有以下几个特点：

❑显示除title（主题）外的其他内容，如脚本变量、Ajax获取的远程内容等。

❑自定义提示内容显示的位置，如“居中”、“偏左”、“偏右”等。

❑自定义提示内容的外观，如警告或错误提示的样式等。

该插件功能强大，且使用起来十分方便，元素与该插件绑定的格式代码如下：

```
$( ".selector" ).tooltip(options)
```

在上述格式调用代码中，如果不设置工具提示内容的来源，默认值为元素的title属性值。另外，options同样也是一个对象，与该插件相关的所有配置项都可以通过该对象进行设置，其常用的属性值如表9-9所示。

表 9-9　选项 options 中的常用参数

参数名称	功能描述
content	设置工具提示的显示内容，省略该属性时，显示元素的 title 属性值。默认值为函数的返回值或 title 属性
hide	设置工具提示隐藏时的效果，默认值为 null，该值可以为布尔值、数字、字符串和对象。如为数字，则在该值的时间内隐藏；如为字符串，则按该字符串的效果进行隐藏；如是对象，则按其效果进行隐藏

（续）

参数名称	功能描述
show	设置工具提示显示的效果，与 hide 属性值相同。该属性可以接收多种类型的值，并根据各类型进行不同效果的显示
tooltipClass	设置或获取工具提示插件的当前样式类别名称，默认值为 null。如果定义了一个名为“custom”的样式，设置代码为“元素名.tooltip({ tooltipClass: "custom" })”
open	工具提示插件的一个方法。可以通过该方法以编程的方式打开一个工具提示栏。该方法常用于动态控制工具提示插件隐藏或显示时使用
create	工具提示插件创建事件。当一个工具提示插件在创建时触发，通过事件的回调函数可以获取工具提示插件的内容和各种状态值

接下来通过一个简单的示例来演示tooltip插件在页面中的基本用法。

示例9-11　使用tooltip插件演示菜单的基本功能

（1）插件文件

Css/jquery-ui-1.9.2.custom.min.css

Jscript/jquery-ui.js

（2）功能描述

在新建的页面中，以列表的方式展示“作品集”中的“书名”和“出版时间”，当用户将鼠标移至“书名”选项时，将与工具提示插

件相绑定，以悬浮的形式展示该选项作品的图片封面，鼠标离开后将自动隐藏该工具提示框。

（3）实现代码

新建一个HTML文件9-11.html，加入如代码清单9-11所示的代码。

代码清单 9-11　使用 tooltip 插件显示列表项的图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用工具提示插件显示列表项的图片</title>
    <link href="Css/jquery-ui-1.9.2.custom.min.css"
          rel="stylesheet" type="text/css" />
    <style type="text/css">
        #frame{width:290px;font-size:13px;border:solid 1px #ccc}
        #frame .title{padding:5px;background-color:#eee;font-size:14px}
        #frame .content{padding:10px 10px}
        #frame  .content ul{list-style-type:none;line-height:1.8em;padding:0;margin:0px}
        #frame .content .spanT{width:210px;float:left;font-weight:bold}
        #frame .content .spanL{width:210px;float:left}
    </style>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery-ui.js">
    </script>
```

```
    <script>
        // 定义数据保存图片路径
        var arrImg = new Array("Images/img04.jpg", "Images/img05.jpg", "Images/img06.jpg");
        $(function() {
            // 遍历所有列表选项
            $(".spanL").each(function(index) {
                // 增加 title 属性
                $(this).attr("title", "<img src='" + arrImg[index] + "'></img>");
                $(this).tooltip(
                            { position: { my: "left+100"} }
                    );
            });
        });
    </script>
</head>
<body>
    <div id="frame">
        <div class="title">
            本人部分作品集 </div>
        <div class="content">
            <ul>
                <li><span class="spanT"> 书名 </span>
                    <span><b> 出版时间 </b></span>
                </li>
                <li>
                <span class="spanL">
                    <a href="javascript:">
                    jQuery 权威指南
                    </a>
                </span>
                <span>2011.5</span>
                </li>
                <li>
                <span class="spanL">
                    <a href="javascript:">
                    HTML 5 实战
                    </a>
                </span>
                <span>2012.1</span>
                </li>
                <li>
                <span class="spanL">
                    <a href="javascript:">
                    jQuery Mobile 权威指南
                    </a>
                </span>
                <span>2012.8</span>
                </li>
            </ul>
        </div>
    </div>
</body>
</html>
```

（4）页面效果

实现的页面效果如图9-13所示。

图　9-13　使用工具提示插件显示列表项的图片

（5）代码分析

在本示例中，先定义一个用于保存图书封面图片路径的数组变量"arrImg"，代码如下：

```
//定义数据保存图片路径
var arrImg = new Array("Images/img04.jpg",
                       "Images/img05.jpg", "Images/img06.jpg");
```

然后，遍历列表项中类别名为"spanL"的选项，在遍历过程中，先将当前选项元素的"title"属性值设置为与遍历索引号相同的数组内容，再通过"tooltip"方法将当前的选项元素与工具提示插件相绑定。在绑定时，设置工具提示框的位置是加100居左侧，代码如下：

```
//遍历所有列表选项
$(".spanL").each(function(index) {
     //增加title属性
     $(this).attr("title", "<img src='" + arrImg[index] + "'></img>");
     $(this).tooltip({ position: { my: "left+100"} });
});
```

注意　在实际项目中，想通过点击内容显示与内容相关的图片，需要在页面中显示数据时，将图片URL与元素title属性直接相绑定，再调用插件的tooltip方法即可。如果在遍历过程中再异步获取或匹配，将影响图片显示速度和用户UI体验。

9.5　综合案例分析——使用jQuery UI插件以拖动方式管理相册

9.5.1　需求分析

经分析，该案例的需求如下：

1）可以将列表中的某张图片，通过拖动或单击“删除”链接的方式移至回收站。

2）在回收站中，可以将列表中的某张图片，通过拖动或单击“还原”链接的方式返回相册。

3）在拖动或还原图片时，需要有动画效果，以表示正在进行的动作。

9.5.2　界面效果

该案例的界面效果如下所示。

1）在图片列表中，单击某张图片，出现移动鼠标样式后就可以拖动该图片至回收站中。其实现的界面如图9-14所示。

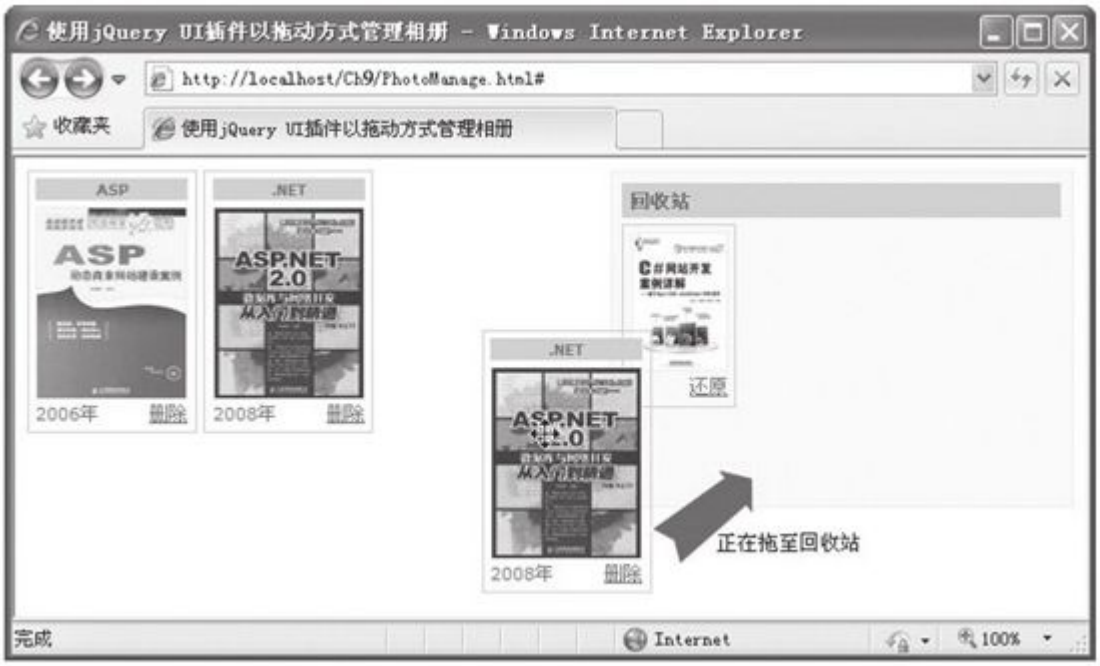


图　9-14　将列表中的图片拖至回收站

2）在回收站中，如果单击“还原”链接或拖动某图片，可以将回收站中的图片还原至列表中，其实现的界面如图9-15所示。

图　9-15　将回收站中的图片还原至列表

3）无论是“删除”还是“还原”图片，在拖动至板块后，以动画的效果进行排列。如将图片拖至回收站中，其排列的动画效果如图9-16所示。

图　9-16　将删除的图片以动画的效果排列至回收站

9.5.3　功能实现

在本案例中，运用了拖动（draggable）和放置（droppable）两款jQuery UI插件，实现图片从列表拖至回收站中，然后从回收站还原到列表中的功能。

新建一个页面文件PhotoManage.html，加入如代码清单9-12所示的代码。

代码清单 9-12　使用 jQuery UI 插件以拖动的方式管理图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>使用 jQuery UI 插件以拖动的方式管理相册 </title>
<script type="text/javascript"
        src="Jscript/jquery-1.8.2.min.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.widget.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.mouse.js">
</script>
<script type="text/javascript"
        src="Js-9-1/jquery.ui.draggable.js">
</script>
<script type="text/javascript"
        src="Js-9-2/jquery.ui.droppable.js">
</script>
<link rel="stylesheet" type="text/css"
```

```
        href="Css/PhotoManage.css" />
<script type="text/javascript">
    $(function() {
      // 使用变量缓存 DOM 对象
      var $photo = $("#photo");
      var $trash = $("#trash");
      // 可以拖动包含图片的表项标记
      $("li", $photo).draggable({
          revert: "invalid", // 在拖动过程中，停止时将返回原来位置
          helper: "clone",   // 以复制的方式拖动
          cursor: "move"
      });
      // 将相册中的图片拖动到回收站
      $trash.droppable({
          accept: "#photo li",
          activeClass: "highlight",
          drop: function(event, ui) {
              deleteImage(ui.draggable);
          }
      });
      // 将回收站中的图片还原至相册
      $photo.droppable({
          accept: "#trash li",
          activeClass: "active",
          drop: function(event, ui) {
              recycleImage(ui.draggable);
          }
      });
      // 自定义图片从相册中删除到回收站的函数
      var recyclelink = "<a href='#' title='从回收站还原'
                         class='phrefresh'>还原</a>";
      function deleteImage($item) {
          $item.fadeOut(function() {
              var $list = $("<ul class='photo reset'/>")
                          .appendTo($trash);
              $item.find("a.phtrash").remove();
              $item.append(recyclelink)
              .appendTo($list).fadeIn(function() {
                  $item
                  .animate({ width: "61px" })
                  .find("img")
                  .animate({ height: "86px" });
              });
          });
      }
      // 自定义图片从回收站还原至相册时的函数
      var trashlink = "<a href='#' title='放入回收站'
                       class='phtrash'>删除</a>";
      function recycleImage($item) {
          $item.fadeOut(function() {
              $item
              .find("a.phrefresh")
              .remove()
              .end()
              .css("width", "85px")
```

```
                .append(trashlink)
                .find("img")
                .css("height", "120px")
                .end()
                .appendTo($photo)
                .fadeIn();
            });
        }
        // 根据图片所在位置绑定删除或还原事件
        $("ul.photo li").click(function(event) {
            var $item = $(this),
                $target = $(event.target);
            if ($target.is("a.phtrash")) {
                deleteImage($item);
            } else if ($target.is("a.phrefresh")) {
                recycleImage($item);
            }
            return false;
        });
    });
</script>
</head>
<body>
<div class="phframe">
  <ul id="photo" class="photo">
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">ASP</h5>
          <img src="Images/img01.jpg" alt="2006年图书作品"
               width="95" height="120" />
          <span>2006年</span>
          <a href="#" title="放入回收站" class="phtrash">删除</a>
   </li>
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">.NET</h5>
          <img src="Images/img02.jpg" alt="2009年图书作品"
               width="95" height="120" />
          <span>2009年</span>
          <a href="#" title="放入回收站" class="phtrash">删除</a>
   </li>
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">C#</h5>
          <img src="Images/img03.jpg" alt="2010年图书作品"
               width="95" height="120" />
          <span>2010年</span>
          <a href="#" title="放入回收站" class="phtrash">删除</a>
   </li>
  </ul>
  <div id="trash" class="photoframecontent">
      <h4 class="photoframeheader">回收站</h4>
  </div>
</div>
</body>
</html>
```

为了更好地展示图片在各个板块中的拖动页面效果，本案例中创建了一个CSS样式文件PhotoManage.css，加入如代码清单9-13所示的代码。

代码清单 9-13　页面 PhotoManage.html 中导入的 CSS 文件

```
body{font-size:11px;font-family:Verdana,Arial,sans-serif;font-size:11px}
/* 图片样式设置 */
#photo {float:left;width:55%; min-height:12em;padding:0;margin:0px;
        list-style-type:none}
.photo li { float: left; width: 96px; padding: 0.4em;
            margin: 0 0.4em 0.4em 0; text-align:center}
.photo li h5 { margin: 0 0 0.4em; cursor: move}
.photo li span { float:left}
.photo li a { float: right; }
.photo li img { width: 99%;cursor:move;border:solid 1px #eee;
                margin-bottom:3px}
/* 图片外框样式设置 */
.phframe {width:660px}
.photoframecontent {border: 1px solid #ccc;color:#666}
.photoframecontent a {color:#666}
.photoframeheader {border: 1px solid #ccc;background:#ccc;
                   color:#666;font-weight:bold}
.photoframeheader a {color: #666}
.photoframetr {-moz-border-radius-topright:4px;
               -webkit-border-top-right-radius:4px}
/* 特殊样式设置 */
.active {background:#eee;}
.highlight {border:1px solid #fcefa1;background: #fbf9ee }
.reset {margin:0;padding:0; border:0;outline:0;line-height:1.3;
        text-decoration:none;font-size:100%; list-style:none}
.phtrash, .phrefresh{color:#666}
/* 回收站样式设置 */
#trash {float:right;width:42%;min-height:19em;padding:1%}
#trash h4 {line-height: 16px; margin: 0 0 0.4em; padding:4px 0px 0px 4px}
#trash .photo h5, #trash span { display:none;}
```

9.5.4 代码分析

本案例的核心功能是图片以拖动或单击链接的方式，从一个板块转移至另外一个板块，因此，首先在页面中，导入与拖动和放置相关的jQuery UI插件，两款功能插件的导入代码如下所示：

```
... 省略部分代码
<script type="text/javascript"
        src="Js-9-1/jquery.ui.draggable.js">
</script>
<script type="text/javascript"
        src="Js-9-2/jquery.ui.droppable.js">
</script>
... 省略部分代码
```

在页面头文件部分，除引入上面两款功能插件和jQuery库外，其他导入的JS文件中，jquery.ui.core.js为jQuery UI的核心库，另外两个JS文件为辅助功能插件文件。

为了实现相册与回收站板块间图片的相互拖动和放置，先通过变量缓存这两个jQuery对象，为后续的操作提供方便和节省空间，代码如下：

```
//使用变量缓存 DOM 对象
var $photo = $("#photo");
var $trash = $("#trash");
```

因为图片以表项＜li＞的形式展示，因此，在前者定义"$photo"对象中，查找表项＜li＞标记，并设置该对象可以进行拖动，其实现的代码如下所示：

```
//可以拖动包含图片的表项标记
$("li", $photo).draggable({
    revert: "invalid", // 在拖动过程中，停止时将返回原来位置
    helper: "clone",    // 以复制的方式拖动
    cursor: "move"
});
```

接下来要完成的代码是，拖放后的图片放置。放置图片有两种需求，一是从图片列表中拖入回收站，另外一种是从回收站还原至图片列表。因此，根据两种情况分别编写代码。

实现第一种情况的代码如下所示：

```
//将相册中的图片拖动到回收站
$trash.droppable({
    accept: "#photo li",
    activeClass: "highlight",
    drop: function(event, ui) {
          deleteImage(ui.draggable);
    }
});
```

上述代码设置了$trash元素接收对象为"#photo li"，即图片列表中的表项，拖动时的样式类别名为"active"。当图片拖动后放入回收站时，触发一个回调函数deleteImage()，在该函数中，获取的表

项"$item"在淡出时，先在回收站中增加一个列表＜ul＞标记，并移除原先列表中的“删除”链接"a.phtrash"，然后，将“还原”链接字符串"recyclelink"加入表项中，部分代码如下：

```
$item.fadeOut(function() {
   var $list = $("<ul class='photo reset'/>").appendTo($trash);
   $item.find("a.phtrash").remove();
   $item.append(recyclelink);
   ...
});
```

同时，在回收站中，以淡入的方式展示图片时，先将获取的表项"$item"，采用动画的方式缩小长度，然后，获取该表项中的＜img＞元素，并以动画的方式缩小＜img＞元素的高度，通过这样连贯的动作，使图片在放入回收站时动画效果更流畅。其代码如下：

```
$item.appendTo($list).fadeIn(function() {
   $item.animate({ width: "61px" })

    .find("img").animate({ height: "96px" });
});
```

第二种情况是，将图片从回收站还原至图片列表中。为此，图片列表绑定放置方法droppable()，接收从回收站中拖放的图片，其实现的代码如下：

```
//将回收站中的图片还原至相册
$photo.droppable({
    accept: "#trash li",
    activeClass: "active",
    drop: function(event, ui) {
          recycleImage(ui.draggable);
    }
});
```

在图片列表绑定droppable()方法中，声明接收对象为"#trash li"，拖动时的样式类别名称为"active"，当图片拖动至图片列表中时，触发一个回调函数recycleImage()，该函数中先获取可以放置的对象"$item"，在该对象淡出回收站的过程中，移出“还原”链接字符"a.phrefresh"，新增“删除”链接字符串"trashlink"，其实现代码如下所示：

```
$item.fadeOut(function() {
    $item
    .find("a.phrefresh")
    .remove()
    .end()
    .css("width", "85px")
    .append(trashlink)
     ...
});
```

此外，在回调函数recycleImage()中，当图片从回收站中淡出时，获取放置对象"$item"中的＜img＞元素，并还原其高度，插入图片列表中，并以淡入的方式显示插入的图片，其实现的代码如下所示：

```
$item.fadeOut(function() {
    $item
     ...
    .find("img")
    .css("height", "120px")
    .end()
    .appendTo($photo)
    .fadeIn();
});
```

图片除拖动方式实现“删除”与“还原”功能外，还可以单击列表中图片下方的“删除”与“还原”字符链接，单击链接时，直接调用相应的自定义函数即可。因此，还需要设置图片列表＜li＞中字符链接的单击事件，其实现的代码如下：

```
//根据图片所在位置绑定删除或还原事件
$("ul.photo li").click(function(event) {
   var $item = $(this),
   $target = $(event.target);
   if ($target.is("a.phtrash")) {
```

```
            deleteImage($item);
        } else if ($target.is("a.phrefresh")) {
            recycleImage($item);
        }
    return false;
});
```

代码中，"$target"为事件触发的对象，如果该对象为"a.phtrash"，表示单击了“删除”链接，则执行自定义函数deleteImage($item)，否则，执行自定义函数recycleImage($item)。最后，加入return false语句，避免单击链接时页面跳转。

另外，为了展示图片列表中的图片与回收站中的图片不同，本案例通过一个CSS样式隐藏回收站中图片的＜h5＞与＜span＞标记，其代码如下所示：

```
#trash .photo h5, #trash span {display:none;}
```

9.6　本章小结

jQuery UI插件在jQuery库中占有重要地位，本章先是通过一个个完整的示例，对jQuery UI常用插件的功能和运用进行了详细的介绍，最后，通过一个综合的案例，介绍拖动（draggable）与放置（droppable）插件在图片管理中的进阶应用，旨在使读者能够通过这些示例，掌握jQuery UI常用插件的基本用法，并能够运用到日常的页面开发中，减少代码的编写总量，提高项目开发速度。

第10章　jQuery实用工具函数

本章内容

工具函数的分类

浏览器的检测

数组和对象的操作

字符串操作

测试操作

URL操作

其他工具函数

工具函数的扩展

综合案例分析——使用jQuery扩展工具函数实现对字符串指定类型的检测

本章小结

在前面的章节中，我们曾经使用过$.each()、$.extend()函数，在jQuery中，它们是非常实用的全局性工具函数中的成员。除此之外，还有很多功能强大的工具函数为我们在处理对象和操作数组方面提供便利，下面我们详细介绍这些工具函数的定义和调用方法。

10.1　工具函数的分类

在jQuery中，工具函数是指直接依附于jQuery对象，针对jQuery对象本身定义的方法，即全局性的函数，我们统称为工具函数（或Utilities函数），它们有一个明显的特征，一般情况下，采用如下的格式进行调用：

```
$.函数名()或jQuery.函数()
```

工具函数对应的官方网址是http://api.jquery.com/category/utilities/。

根据工具函数处理对象的不同，可以将其分为几大类别：浏览器的检测、数组和对象的操作、字符串操作、测试操作、URL操作。下面分别详细介绍这几类工具函数。

10.2　浏览器的检测

在浏览器检测中，又可分为浏览器类型与特征的检测，前者获取浏览器的名称或版本信息，后者检测浏览是否支持标准的W3C盒子模型。

10.2.1　浏览器名称或版本信息

虽然jQuery有很好的浏览器兼容性，但有时程序开发人员需要获取浏览器的相关信息，提供给用户或程序。在jQuery中，可以通过访问$.browser对象的属性获取。$.browser对象即jQuery.browser对象，用于处理与浏览器相关的事务，该对象的属性如表10-1所示。

表 10-1　$.browser 对象的属性

属性名称	功能描述
webkit	如果是 WebKit 相关的浏览器，为 true，否则为 false。这是 1.4 及以上版本新增的属性
mozilla	如果是 Mozilla 相关的浏览器，为 true，否则为 false
safari	如果是 Safari 浏览器，为 true，否则为 false
opera	如果是 Opera 浏览器，为 true，否则为 false
msie	如果是 IE 浏览器，为 true，否则为 false
version	获取浏览器的版本号

下面通过一个示例介绍在页面中查看浏览器的相关信息。

示例10-1　使用browser对象获取浏览器信息

（1）功能描述

根据浏览器的类型，在页面中显示浏览器的名称与版本号。

（2）实现代码

新建一个HTML文件10-1.html,加入如代码清单10-1所示的代码。

代码清单 10-1　使用 browser 对象获取浏览器信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 browser 对象获取浏览器信息 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "您的浏览器名称是: ";
            if ($.browser.msie) { //IE 浏览器
                strTmp += "IE";
            }
            if ($.browser.mozilla) { // Firefox 相关浏览器
            strTmp += "Mozilla FireFox";
            }
            strTmp += " 版本号是: " + $.browser.version; //获取版本号
            $("#divTip").html(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-1所示。

图　10-1　在IE 8与Firefox下显示浏览器的信息

（4）代码分析

在本示例的JavaScript代码中，先通过$.browser对象获取当前浏览器的名称，并保存在变量"strTmp"中；然后根据$.browser对象的version属性获取浏览器版本号信息，并将该值以链加的形式一起保存在变量"strTmp"中；最后通过ID号为"divTip"的元素将变量内容显示在页面中。

说明　在Mozilla Firefox浏览器中，$.browser.version获取的是其内核版本号，虽然版本是3.6.10，但其内核依然是1.9.2.10。

10.2.2　盒子模型

盒子模型是CSS中的专术名词，用以描述页面设置中的各种属性，如内容（content）、填充（padding）、边框（border）、边界（margin）。由于这些属性拼合在一起，与日常生活中的“盒子”很相像，因此称为盒子模型。

盒子模型分为两类：一类是W3C盒子模型；另一类是IE盒子模型。两者最根本的区别在于属性height与width这两个值是否包含padding与border。

W3C盒子模型不包含padding与border，仅指内容（content）的height和width。W3C盒子模型如图10-2所示。IE盒子模型的height和width的长度包含padding与border，如图10-3所示。

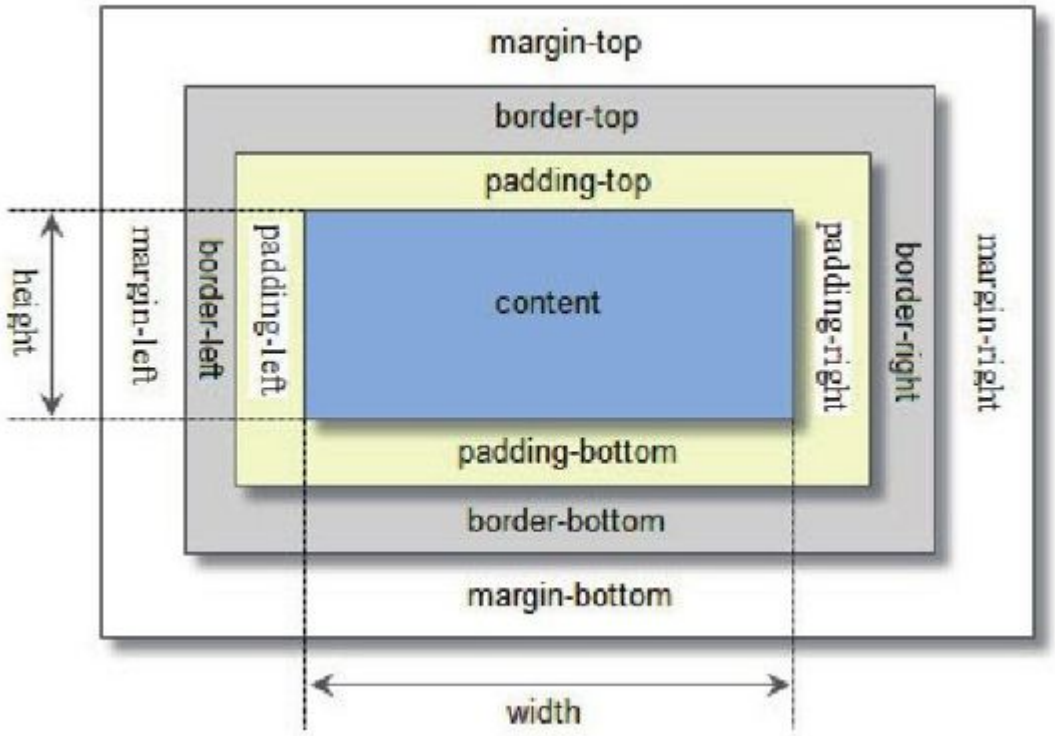


图　10-2　W3C盒子模型

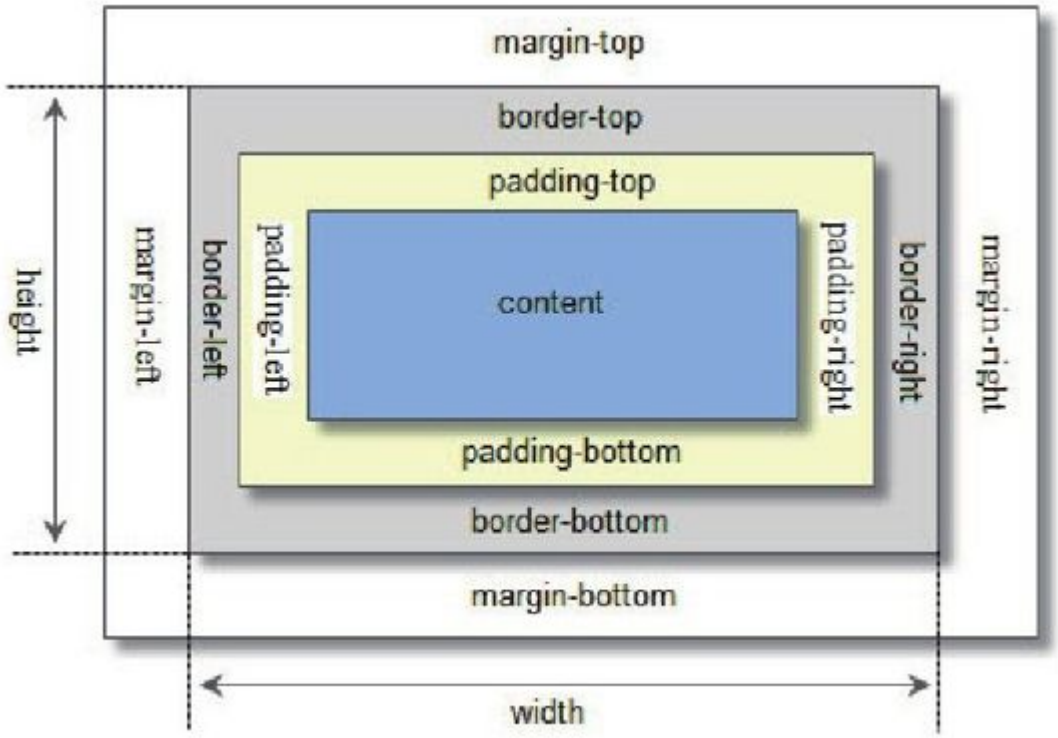


图　10-3　IE盒子模型

在jQuery中，可以通过jQuery.support.boxModel对象返回的属性值，确定页面是否是标准的W3C盒子模型，其调用的方法有两种，如下所示：

```
jQuery.support.boxModel
$.support.boxModel
```

该方法返回一个布尔值，如果是true表示是W3C盒子模型，否则不是W3C盒子模型。下面通过一个示例介绍检测页面是否是W3C盒子模型。

示例10-2　使用boxModel对象检测是否是W3C盒子模型

（1）功能描述

根据页面的特征，显示是否是标准的W3C盒子模型。

（2）实现代码

新建一个HTML文件10-2.html,加入的代码如代码清单10-2所示。

代码清单 10-2　使用 boxModel 对象检测是否是 W3C 盒子模型

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 boxModel 对象检测是否是 W3C 盒子模型 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "您打开的页面是: ";
            if ($.support.boxModel) { // 是 W3C 盒子模型
                strTmp += "W3C 盒子模型 ";
            }
            else { // 是 IE 盒子模型
                strTmp += "IE 盒子模型 ";
            }
            $("#divTip").html(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-4所示。在代码清单10-2中，如果删除下列代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

图　10-4　W3C盒子模型

重新执行的效果如图10-5所示。

图　10-5　IE盒子模型

（4）代码分析

在本示例的JavaScript代码中，通过$.support.boxModel对象判断当前页面是否是盒子模型，如果是，返回true，否则返回false；根据该返回值，将不同的文字内容保存在变量"strTmp"中，最后通过ID号为"divTip"的元素将变量内容显示在页面中。

说明　编写页面代码时，尽量先声明DOCTYPE类型，使用标准的W3C盒子模型，避免多个浏览间的相互不兼容。

10.3　数组和对象的操作

在Web页开发过程中，经常使用数组保存一些特定的数据，在jQuery中，使用框架自带的一些工具函数，可以很方便地对数组进行操作，如遍历、筛选、修改数组。在使用遍历工具函数时，不仅可以遍历数据，还可以遍历对象，在接下来的章节中，将对这些工具函数的使用方法进行逐一介绍。

10.3.1　遍历数组

使用$.each()工具函数可以实现页面中元素的遍历，此外还可以完成指定数组的遍历，其调用的语法格式如下：

```
$.each(obj,fn(para1,para2))
```

其中，参数obj表示要遍历的数组或对象。fn为每个遍历元素执行的回调函数，该函数包含两个参数：para1表示数组的序号或对象的属性；para2表示数组的元素和对象的属性。

下面通过一个示例介绍$.each()工具函数遍历数组的方法。

示例10-3　使用$.each()工具函数遍历数组

（1）功能描述

定义一个数组，用于保存学生的相关资料，通过$.each()工具函数获取数组中各元素的名称与内容，显示在页面中。

（2）实现代码

新建一个HTML文件10-3.html,加入的代码如代码清单10-3所示。

代码清单 10-3　使用 $.each() 工具函数遍历数组

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.each() 工具函数遍历数组 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">

       body{font-size:13px}
       ul{padding:0px;margin:0px;list-style-type:none;
          width:260px;border-top:dashed 1px #ccc;
          border-left:dashed 1px #ccc;border-right:dashed 1px #ccc}
          ul li{border-bottom:dashed 1px #ccc;padding:8px}
          .title{background-color:#eee; font-weight:bold}
    </style>
    <script type="text/javascript">
        $(function() {
            var arrStu = { "张三": "60", "李四": "70", "王二": "80" }
            var strContent = "<li class='title'>姓名: 分数 </li>";
            $.each(arrStu, function(Name, Value) {
                strContent += "<li>" + Name + Value + "</li>";
            })
            $("ul").append(strContent);
        })
    </script>
</head>
<body>
    <ul></ul>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-6所示。

图　10-6　遍历数组

（4）代码分析

在本示例的JavaScript代码中，先定义一个数组变量"arrStu"，用于保存列表项的数据；再定义一个字符变量"strContent"，该变量的初始化内容为列表项的标题数据，然后使用工具函数$.each()对"arrStu"数组进行遍历，获取数组中的各项内容，并以链加的方式保存在变量"strContent"中；最后通过append方法将全部的内容追加至元素＜ul＞中，并最终显示在页面中。

10.3.2　遍历对象

$.each()函数除了遍历数组外，还可以遍历对象，获取对象的属性和值。常用于对一些未知对象的遍历，其使用方法与遍历数组基本一致，下面通过一个示例说明其遍历对象的使用方法。

示例10-4　使用$.each()工具函数遍历ajaxSettings对象

（1）功能描述

通过$.each()工具函数遍历ajaxSettings对象，获取该对象全部的属性和值，并显示在页面中。

（2）实现代码

新建一个HTML文件10-4.html,加入的代码如代码清单10-4所示。

代码清单 10-4　使用 $.each() 工具函数遍历 ajaxSettings 对象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.each() 工具函数遍历 ajaxSettings 对象</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
       body{font-size:13px}
       ul{padding:0px;margin:0px;list-style-type:none;
          width:350px;border-top:dashed 1px #ccc;
          border-left:dashed 1px #ccc;border-right:dashed 1px #ccc}
          ul li{border-bottom:dashed 1px #ccc;padding:8px}
          .title{background-color:#eee; font-weight:bold}
    </style>
    <script type="text/javascript">
        $(function() {
            var strContent = "<li class='title'>属性: 值</li>";
            $.each($.ajaxSettings, function(Property, Value) {
            strContent += "<li>" + Property + ": " + Value + "</li>";
            })
            $("ul").append(strContent);
        })
    </script>
</head>
<body>
    <ul></ul>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-7所示。

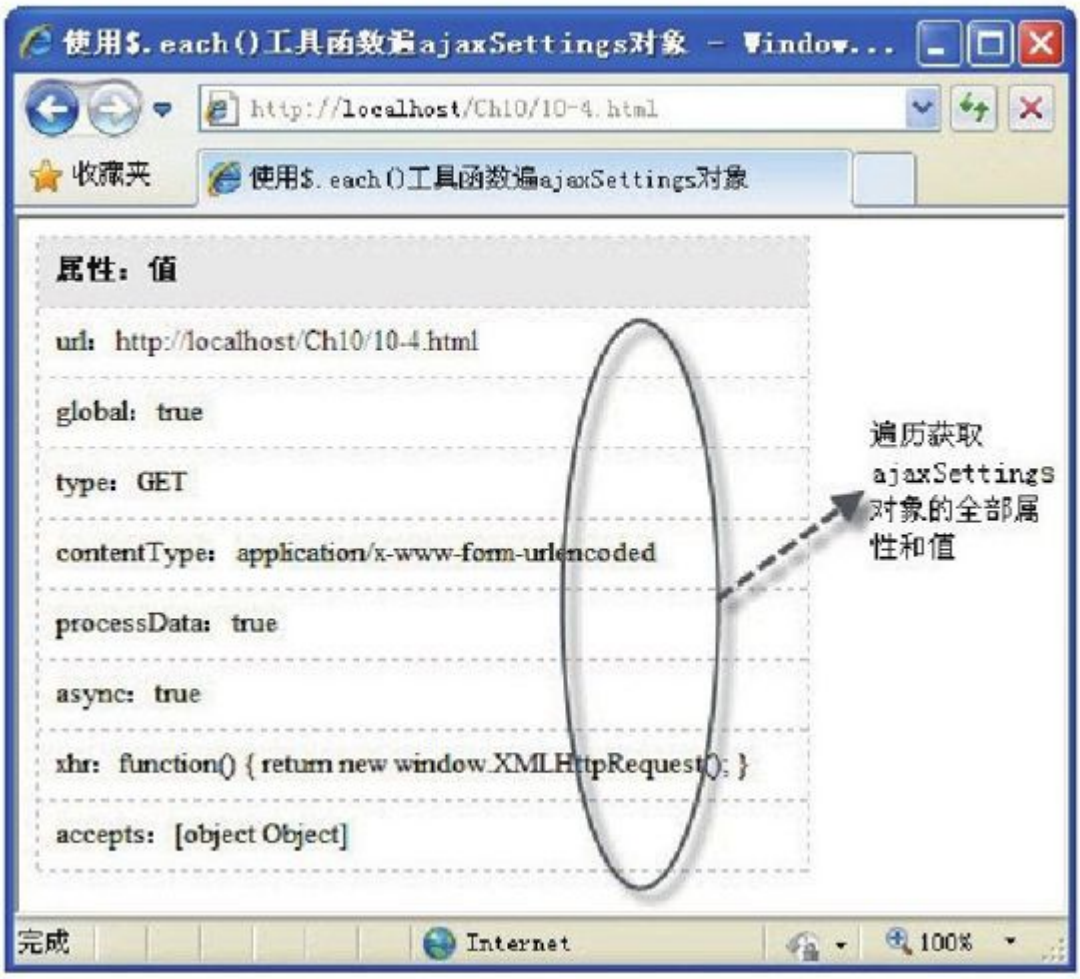


图　10-7　使用$.each()工具函数遍历ajaxSettings对象

（4）代码分析

在本示例的JavaScript代码中，虽然ajaxSettings是一个对象，但它的内部结构属性则是以数组的形式进行存取的，因此，通过$.each()函数可以获取该对象的属性名和对应的值。在遍历过程中，先将获取的属性值以链加的方式保存在变量"strContent"中，然后将变量内容通过"append"方法追加至页面的元素中。

10.3.3　数据筛选

在操作数组时，有时需要根据各种条件筛选元素，传统的JavaScript代码将遍历整个数组，在遍历中设置筛选规则，然后获取符合规则的元素。而在jQuery中，可以使用工具函数jQuery.grep()，很方便地筛选数组中的任何元素，该函数调用的语法格式如下：

```
$.grep(array, function(elementOfArray, indexInArray),[invert])
```

参数array为要筛选的原数组，回调函数fn中可以设置两个参数，其中elementOfArray为数组中的元素，indexInArray为元素在数组中的序列号。另外，可选项[invert]为布尔值，表示是否根据fn的规则取反向结果，默认值为false，表示不取反，如果为true表示取反，即返回与回调函数fn规则相反的数据。

下面通过一个示例介绍工具函数jQuery.grep筛选数组中的元素。

示例10-5　使用$.grep()工具函数筛选数组中的元素

（1）功能描述

通过$.grep()工具函数查询数组中大于5且序号小于8的元素，并显示在页面中。

（2）实现代码

新建一个HTML文件10-5.html,加入的代码如代码清单10-5所示。

代码清单 10-5　使用 $.grep() 工具函数筛选数组中的元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.grep() 工具函数筛选数组中的元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "筛选前数据: ";
            var arrNum = [2, 8, 3, 7, 4, 9, 3, 10, 9, 7, 21];
            var arrGet = $.grep(arrNum, function(ele, index) {
                return ele > 5 && index < 8 // 元素值大于 5 且序号小于 8
            })
            strTmp += arrNum.join();
            strTmp += "<br/><br>筛选后数据: "
            strTmp += arrGet.join();
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-8所示。

图　10-8　筛选数组中的元素

（4）代码分析

在本示例的JavaScript代码中，由于数组的序号是从0开始的，因此第8位元素10尽管大于5，符合第一个条件，但它不符合序列号小于8的条件，因此筛选结果中没有出现该元素。

10.3.4　数据变更

虽然可以通过$.grep()函数筛选数组中的元素，但如果要按指定条件修改数组中的所选元素，还需调用另外一个工具函数$.map()，其调用的语法格式如下：

```
$.map(array, callback(elementOfArray, indexInArray)
```

其中，参数array为要变更的原数组。回调函数fn中可以设置两个参数，其中elementOfArray为数组中的元素，indexInArray为元素在数组中的序列号。

下面在示例10-5的基础上调用$.map()函数实现数组中元素的变更。

示例10-6　使用$.map()工具函数变更数组中的元素

（1）功能描述

通过$.map()工具函数将数组中大于5且序号小于8的元素都增加3，并显示在页面中。

（2）实现代码

新建一个HTML文件10-6.html,加入如代码清单10-6所示的代码：

代码清单 10-6　使用 $.map() 工具函数变更数组中的元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.map() 工具函数变更数组中的元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "变更前数据: ";
            var arrNum = [2, 8, 3, 7, 4, 9, 3, 10, 9, 7, 21];
            var arrGet = $.map(arrNum, function(ele, index) {
                if (ele > 5 && index < 8) { //元素值大于 5 且序号小于 8
                    return ele + 1; //元素增加 1
                }

            })
            strTmp += arrNum.join();
            strTmp += "<br/><br>变更后数据: "
            strTmp += arrGet.join();
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-9所示。

图　10-9　变更数组中的元素

（4）代码分析

在本示例的JavaScript代码中，使用$.map()工具函数前，先通过"if(ele＞5＆＆index＜8)"语句筛选元素，然后使用"ele+1"语句实现每个符合条件元素的更新。

10.3.5　数据搜索

在jQuery中，如果要在数组中搜索某个元素，可以使用工具函数$.inArray()。该方法相当于用JavaScript中的indexOf()函数搜索字符串中的某个字符。在工具函数$.inArray()中，如果找到了指定的某个元素，则返回该元素在数组中的索引号，否则返回-1值。其调用的格式如下所示：

```
$.inArray(value, array)
```

其中，参数value表示要搜索的对象，array表示搜索对象的数组。

下面通过一个示例介绍如何通过$.inArray()函数在指定的数组中搜索某个字符的方法。

示例10-7　使用$.inArray()工具函数搜索数组中指定元素的位置

（1）功能描述

通过$.inArray()函数在数组arrStr中搜索内容为“2”的第一个匹配元素的位置，并将结果显示在页面中。

（2）实现代码

新建一个HTML文件10-7.html,加入如代码清单10-7所示的代码。

代码清单 10-7　使用 $.inArray() 工具函数搜索数组中指定元素的位置

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.inArray() 工具函数搜索数组中指定元素的位置 </title>

    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "待搜索数据: ";
            var arrNum = [4, 21, 2, 12, 5];
            var arrPos = $.inArray(2, arrNum);
            strTmp += arrNum.join();
            strTmp += "<br/><br>搜索后结果: "
            strTmp += arrPos;
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-10所示。

图　10-10　使用$.inArray()工具函数搜索数组中指定元素的位置

（4）代码分析

在本示例的JavaScript代码中，由于$.inArray函数在搜索过程中，搜索对象匹配的是单个元素，而非元素内容的相似，因此，尽管第1号元素“21”中含有“2”，但它不是单个元素，仅是包含搜索对象，因此不匹配。

10.4　字符串操作

在jQuery中，如果要除掉字符中左右两边的空格符，可以使用工具函数$.trim()。该函数常用于字符串操作，也是jQuery核心库中唯一针对字符串操作的工具函数，其调用的语法格式如下所示：

```
$.trim(str)
```

其中，参数str为需要删除左右两边空格符的字符串。

下面通过一个示例介绍$.trim()函数除掉空格的方法。

示例10-8　使用$.trim()工具函数删除字符串左右两边的空格符

（1）功能描述

通过$.trim()函数删除一个两边均有空格符的字符串，并将其执行前后的字符长度都显示在页面中。

（2）实现代码

新建一个HTML文件10-8.html,加入如代码清单10-8所示的代码。

代码清单 10-8　使用 $.trim() 函数删除字符串左右两边的空格符

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.trim() 工具函数除掉字符串的空格符 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "内容: ";
            var strOld = "  jQuery,write less do more    ";
            var strNew = $.trim(strOld);
            strTmp += strOld;
            strTmp += "<br/><br>除掉空格符前的长度: "
            strTmp += strOld.length;
            strTmp += "<br/><br>除掉空格符后的长度: "
            strTmp += strNew.length;
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-11所示。

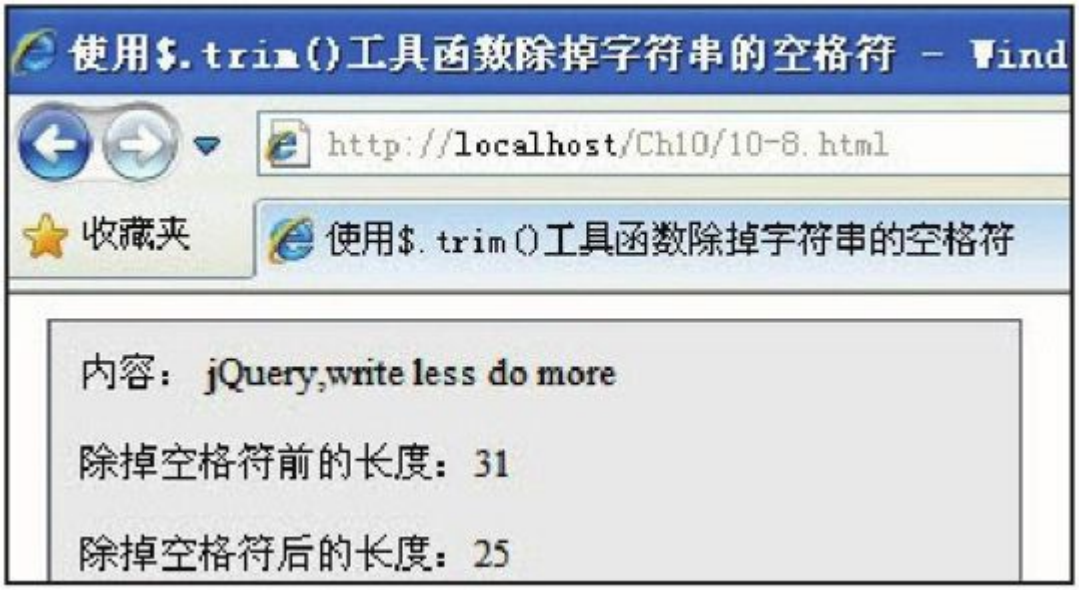


图　10-11　使用$.trim()工具函数删除字符串左右两边的空格符

（4）代码分析

在本示例的JavaScript代码中，变量"strOld"保存的内容中左右都留有空格，而使用$.trim()函数可以将字符左右的空格删除，因此，删除空格后的字符变量"strNew"长度比删除空格前的字符变量"strNew"长度要小。该函数常用于用户输入文本内容后，向服务端提交数据时使用，主要用于更好地格式化用户输入的数据。

10.5 测试操作

在编写代码时，有时需要先获取对象或元素的各种状态，如是否为空、是否是对象等，然后根据这些状态值决定程序的下一步操作。因此，相关状态检测的工具函数，在代码编写中也十分重要。jQuery 1.4.2及以上版本中，有5个涉及状态检测的工具函数，其详细信息如表10-2所示。

表 10-2 jQuery 中测试工具函数

函数名称	功能描述	支持版本
$.isArray(obj)	返回一个布尔值，检测参数 obj 是否是一个数组对象，如果为 true 表示是，否则表示不是	1.3 以上
$.isFunction(obj)	返回一个布尔值，检测参数 obj 是否是一个函数，如果为 true 表示是，否则表示不是	1.2 以上
$.isEmptyObject(obj)	返回一个布尔值，检测参数 obj 是否是一个空对象，如果为 true 表示是，否则表示不是	1.4 以上
$.isPlainObject(obj)	返回一个布尔值，检测参数 obj 是否是一个纯粹对象，如果为 true 表示是，否则表示不是	1.4 以上
$.contains(container, contained)	返回一个布尔值，检测一个 DOM 节点是否包含另一个 DOM 节点，如果为 true 表示是，否则表示不是	1.4 以上

下面通过三个示例，介绍在1.4以上版本中新增检测工具函数的使用方法。

10.5.1 检测对象是否为空

$.isEmptyObject()函数可以检测对象本身和原型继承属性这两个特征是否为空，对象的原型继承属性是指对象是否使用hasOwnProperty，拥有自己的属性名称。

$.isEmptyObject()函数的参数是一个普通的JavaScript对象，该函数返回true时，表示是一个空对象，否则为非空对象。下面通过一个示例介绍该函数的使用方法。

示例10-9　使用$.isEmptyObject()函数检测对象是否为空

（1）功能描述

通过$.isEmptyObject()函数检测某个指定的对象是否为空，并将结果显示在页面中。

（2）实现代码

新建一个HTML文件10-9.html,加入如代码清单10-9所示的代码。

代码清单 10-9　使用 $.isEmptyObject() 函数检测对象是否为空

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.isEmptyObject() 函数检测对象是否为空</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var obj0 = {};
            var obj1 = { "name": "taoguorong" };
            var strTmp = "obj0 是否为空: " + $.isEmptyObject(obj0);
            strTmp += "<br><br>obj1 是否为空: " + $.isEmptyObject(obj1);
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-12所示。

图　10-12　检测对象是否为空

（4）代码分析

因为对象obj0的内容为{}，obj1的内容为{"name":"taoguorong"}，所以$.isEmptyObject()函数检测obj0时返回true，检测obj1时返回false。

10.5.2　检测对象是否为原始对象

在jQuery 1.4以上版本中，除可以检测对象是否为空外，还可以通过$.isPlainObject()函数检测对象是否是一个纯粹、原始的对象，即对象是否通过{}或new Object()关键字创建。下面通过一个示例介绍该函数的使用方法。

示例10-10　使用$.isPlainObject()函数检测对象是否为原始对象

（1）功能描述

通过$.isPlainObject()函数检测某个指定的对象是否为原始对象，并将结果显示在页面中。

（2）实现代码

新建一个HTML文件10-10.html,加入如代码清单10-10所示的代码。

代码清单 10-10　使用 $.isPlainObject() 函数检测对象是否为原始对象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.isPlainObject() 函数检测对象是否为原始对象</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var obj0 = {};
            var obj1 = new Object();
            var obj2 = "null";
            var strTmp = "obj0 是否为原始对象: " + $.isPlainObject(obj0);
            strTmp += "<br><br>obj1 是否为原始对象: " + $.isPlainObject(obj1);
            strTmp += "<br><br>obj2 是否为原始对象: " + $.isPlainObject(obj2);
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图10-13所示。

图　10-13　检测对象是否为原始对象

（4）代码分析

在使用$.isPlainObject()函数检测对象是否为原始对象时，由于obj1定义时为new Object()，因此该对象为原始对象；如果obj1定义时带参数，如new Object("name")，这时当传递参数生成对象时，构造器不一定指向Object，因此$.isPlainObject()函数检测时，将返回false。

10.5.3　检测两个节点的包含关系

$.contains()函数的作用是检测在一个DOM节点中是否包含另外一个DOM节点，其调用的语法格式如下：

```
$.contains(container, contained)
```

其中，参数container为Object，是一个DOM元素，作为容器可以包容其他DOM元素；参数contained也是一个DOM是一个节点，可能被其他元素所包含。整个函数返回一个布尔值，如果container对象包含contained对象，则结果为true，否则结果为false。

下面通过一个示例介绍$.contains()函数检测节点的方法。

示例10-11　使用$.contains()函数检测两个节点的包含关系

（1）功能描述

通过$.contains()函数检测指定的两个节点之间是否存在包含的关系，并将结果显示在页面中。

（2）实现代码

新建一个HTML文件10-11.html,加入如代码清单10-11所示的代码。

代码清单 10-11　使用 $.contains() 函数检测两个节点是否包含

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.contains() 函数检测两个节点是否包含 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var node0 = document.documentElement;
            var node1 = document.body;
            var strTmp = "对象 node0 是否包含对象 node1: "
            strTmp += $.contains(node0, node1);//检测两者的包含关系
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-14所示。

图　10-14　检测两个节点是否包含

（4）代码分析

在DOM中，documentElement是文档的根节点，而body是根节点下的一个子节点，变量node0代表documentElement，node1代表body。因此，如果执行$.contains(node0,node1)，返回true，反之返回false。

10.6　URL操作

在6.2.3节中，我们曾介绍过使用serialize()方法来序列化表单向服务器提交的数据，即URL操作，而serialize()方法的核心则是工具函数$.param()。通过该函数可以使数组或jQuery对象按照name/value的格式进行序列化，普通对象按照key/value的格式进行序列化。工具函数$.param()调用的语法格式为：

```
$.param(obj, [traditional])
```

其中，参数obj表示需要进行序列化的对象，该对象可以是数组、jQuery元素、普通对象。可选项参数[traditional]表示是否使用普通的方式浅层序列化，该函数返回一个序列化后的字符串。

下面通过一个完整的示例，介绍如何调用工具函数$.param()进行数组元素序列化的方法。

示例10-12　使用$.param()函数对数组进行序列化

（1）功能描述

定义两个不同内容的数组，第一个数组arrInfo用于保存基本信息（如ID号、名称）；第二个数组arrScore用于保存分数和汇总信息。

使用$.param()函数分别对这两个数组进行序列化，使其各自成为可以执行传值的URL，并将该URL分别显示在页面中。

（2）实现代码

新建一个HTML文件10-12.html，加入如代码清单10-12所示的代码。

代码清单 10-12　使用 $.param() 函数对数组进行序列化

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.param() 进行数组元素序列化 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}

    </style>
    <script type="text/javascript">
        $(function() {
            //基本信息数组
            var arrInfo = { id: 101, name: "tao", sex: 0 };
            //分数和汇总信息数组
            var arrScore = { Score: { chinese: 90, maths: 100, english: 98 },
                SunNum: { Score: 288, Num: 3 }
            };
            //序列化各数组
            var arrNewInfo = $.param(arrInfo);
            var arrNewScore = $.param(arrScore);
            var arrDecScore = decodeURIComponent($.param(arrScore));
            //显示序列化后的数组
            var strTmp = "<b>arrInfo 数组序列化后 </b>: ";
            strTmp += arrNewInfo;
            strTmp += "<br><br><b>arrScore 数组序列化后 </b>: ";
            strTmp += arrNewScore;
            strTmp += "<br><br><b>arrScore 序列化解码后 </b>: ";
            strTmp += arrDecScore;
            //显示在页面中
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-15所示。

图　10-15　数组元素序列化

（4）代码分析

为了更加清晰地展示URL在操作过程中每个参数对应值的状况，我们采用ecodeURIComponent()方法对序列化后的数组arrScore的URL进

行解码，通过解码后展示在页面中的字符串，可以很清楚地知道各参数传递时对应的值。

10.7　其他工具函数

在jQuery中，除了上述介绍的各种工具函数外，还有很多实用的函数，尤其在jQuery 1.4.2以上版本中，新增了很多简洁、高效的工具函数，其中新增函数$.proxy()在处理不同作用域对象事件时相当实用。

10.7.1　$.proxy()函数调用语法

$.proxy()函数返回一个新的函数，并且这个函数始终保持特定的作用域。所谓作用域，就是执行该函数对象的范围。当一个事件函数被元素绑定时，其作用域原则上应指向该元素，但有些事件函数的作用域，在被元素绑定时并不指向元素本身，而是指向另外一个对象。这时，为了使用元素的绑定事件能正常执行，必须调用$.proxy()函数进行处理，经过处理后的事件函数不仅可以被绑定的元素执行，而且还可以传递原先函数取消事件的绑定，该函数调用的语法格式为：

```
$.proxy(function,scope)
```

其中，参数function为要改变作用域的事件函数，参数scope为被事件函数设置作用域的对象，即事件函数的作用域将设置到该对象中。

该函数还有另外一种调用的语法格式，代码如下：

```
$.proxy(scope, name)
```

其中，参数scope为被事件函数设定的作用域对象；参数name为将要设置作用域的函数名，并且该参数必须是scope作用域对象的一个属性。

10.7.2　改变事件函数的作用域

下面通过一个详细的示例介绍$.proxy()函数在页面中的应用。

示例10-13　使用$.proxy()函数改变事件函数的作用域

（1）功能描述

本示例强制性设置事件函数的作用域，使this指向对象函数objMyInfo，而不是被绑定的元素对象#Button1，但又可以执行objMyInfo对象中的ShowEvent事件，将设置的个人信息展示在页面中。

（2）实现代码

新建一个HTML文件10-13.html,加入如代码清单10-13所示的代码。

代码清单 10-13　使用 $.proxy() 函数改变事件函数的作用域

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用函数$.proxy()改变事件函数的作用域</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
           input{margin:5px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
           filter: progid:DXImageTransform.Microsoft
               .Gradient(GradientType=0,StartColorStr=#ffffff,
               EndColorStr=#ECE9D8);}
    </style>


    <script type="text/javascript">
        $(function() {
            var objMyInfo = {
                name: "陶国荣",//设置对象name属性
                sex: "男", //设置对象sex属性
                ShowEvent: function() { //设置执行的事件
                    $("#divShow").html("姓名: " + this.name + "<br><br>性别: " + this.sex);
                }
            }
            $("#Button1").bind("click", //通过proxy函数绑定设置的事件
            $.proxy(objMyInfo.ShowEvent, objMyInfo));
        })
    </script>
</head>
<body>
   <input id="Button1" type="button" value="显示" class="btn" />
   <div id="divShow"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-16所示。

图　10-16　改变事件函数的作用域

（4）代码分析

在上述码中，如果不使用$.proxy()函数改变事件函数的作用域，而是直接将对象的事件函数与执行元素进行绑定，即将下面代码：

```
$("#Button1").bind("click", //通过proxy函数绑定设置的事件
  $.proxy(objMyInfo.ShowEvent, objMyInfo));
```

修改成如下代码：

```
$("#Button1").bind("click", //直接绑定对象函数的事件
  objMyInfo.ShowEvent);
```

其执行后的页面效果如图10-17所示。

图　10-17　直接绑定对象函数的事件

从图中可以很明显地看出，在单击按钮时，由于传递的this作用域与事件函数中this的作用域不同，无法访问事件函数中的name与sex属性，所以提示为"underfined"错误信息。

以下两段代码只是调用的方式不同，实现的功能是相同的。即：

```
$("#Button1").bind("click", //通过proxy函数绑定设置的事件
$.proxy(objMyInfo.ShowEvent, objMyInfo));
```

等价于如下代码：

```
$("#Button1").bind("click", //通过proxy函数绑定设置的事件
$.proxy(objMyInfo, "ShowEvent"));
```

10.8　工具函数的扩展

工具函数的扩展，其实质就是自己编写类级别的插件用于扩展jQuery对象本身。为实现这一目的，我们引入另外一个工具函数$.extend()。

通过$.extend()函数可以很方便地定义自己的工具函数。

10.8.1　使用$.extend()扩展工具函数

下面通过一个示例介绍使用$.extend()函数扩展工具函数的过程。

示例10-14　使用$.extend()函数扩展工具函数

（1）功能描述

编写两个自定义的工具函数，一个用于返回两个数中最大值，另一个用于返回两个数中最小值，并在页面中进行调用，将返回的结果显示在页面中。

（2）实现代码

新建一个HTML文件10-14.html,加入如代码清单10-14所示的代码。

代码清单 10-14　使用 $.extend() 函数扩展工具函数

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用函数 $.extend() 扩展工具函数 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        /*-----------------------------------------------------------
        功能：返回两个数中最大值
        参数：数字 p1,p2
        返回：最大值的一个数
        示例：$.MaxNum(1,2);
        -----------------------------------------------------------*/
        ; (function($) {
            $.extend({
                "MaxNum": function(p1, p2) {
```

```
                return (p1 > p2) ? p1 : p2;
            }
        });
    })(jQuery);

    /*-------------------------------------------------------------*/
    功能：返回两个数中最小值
    参数：数字 p1,p2
    返回：最小值的一个数
    示例：$.MinNum(1,2);
    /*-------------------------------------------------------------*/
    ; (function($) {
        $.extend({
            "MinNum": function(p1, p2) {
                return (p1 > p2) ? p2 : p1;
            }
        });
    })(jQuery);

    $(function() {
        var strTmp = "5 与 6 中最大的数是：";
        strTmp += $.MaxNum(5, 6);
        strTmp += "<br><br>7 与 8 中最小的数是：";
        strTmp += $.MinNum(7, 8);
        $("#divTip").append(strTmp);
    })
</script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的效果如图10-18所示。

图　10-18　使用函数$.extend()扩展工具函数

10.8.2　使用$.extend()扩展Object对象

工具函数$.extend()除可以扩展jQuery自身函数外，还有另外一个很强悍的功能，就是扩展已有的Object对象，其调用的语法格式为：

```
$.extend(target, object1,... [objectN])
```

其中，参数target表示合并后的对象；object为被合并对象，即可以将一个或多个对象合并成一个对象，最后返回该对象，如执行下列代码：

```
var objName = { name: "张三", sex: "男" };
var objInfo = { name: "李四", age: 30 };
var objLast = $.extend(objName, objInfo);
```

其最后返回的结果是：

```
objLast ={ name: "李四", sex: "男", age: 30 };
```

从返回结果不难看出，在$.extend()函数中，如果是合并对象，且存在相同参数的名称，后面对象中的参数值将覆盖前面对象中的参数值。

说明　有关使用$.extend()开发插件的更多详细介绍参考第8章。

10.9　综合案例分析——使用jQuery扩展工具函数实现对字符串指定类型的检测

在jQuery中，仅有一个对字符串操作的全局函数$.trim()，该函数可以删除指定字符串的前后空格，在10.4节中已作了详细的说明，而在实际的案例开发过程中，常常需要验证输入字符类型的有效性，如输入“邮箱”时，需要验证用户输入的内容是否符合邮箱的格式等。本节的案例使用jQuery扩展工具开发一个全局函数chkStrByType()，该函数可以根据指定的字符验证类型，检测待验证的字符串是否符合该类型的格式。

10.9.1　需求分析

1）在文本框中输入任意一个字符，选择下拉列表框中的某个字符类型，单击“检测”按钮后，将检测文本框中的字符是否符合选择类型的格式。

2）将检测后的结果显示在页面中。

10.9.2　界面效果

在页面的文本框中输入“陶国荣”三个字，选择“汉字”类型后单击“检测”按钮，将在页面中显示检测后的结果，其实现的界面如图10-19所示。

图　10-19　验证通过时的页面效果

如果在文本框中输入的字符不变，而将选择类型改为“邮政编码”，单击“检测”按钮后，将显示字符与所选类型不符的提示结果，其实现的界面如图10-20所示。

图　10-20　验证失败时的页面效果

10.9.3　功能实现

为了实现字符串指定类型的验证并显示验证结果的功能，新建一个HTML页面chkStr.html，加入代码清单10-15所示的代码。

代码清单 10-15　使用 jQuery 扩展工具函数实现对字符串指定类型的检测

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 扩展工具函数实现对字符串指定类型的检测 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:11px}
          fieldset{width:410px}
          fieldset div{padding:8px}
          fieldset div select{font-size:9pt;padding:1px}
          #divTip{margin-top:10px;padding:10px; border:solid 1px #666;
          background-color:#eee;width:210px;display:none}
          .txt{border:#666 1px solid;padding:2px;width:120px;margin-right:3px}
          .btn {border:#666 1px solid;padding:2px;width:60px;
          filter: progid:DXImageTransform.Microsoft
              .Gradient(GradientType=0,StartColorStr=#ffffff,EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        /*-------------------------------------------------------------------
        功能：返回检测字符串指定类型的结果
        参数：checkType 为检测字符串的类型；strS 为待检测的字符串
        返回：一个 bool 值，true 表示是指定的类型，false 表示不是指定的字符类型
        示例：$.chkStrByType("陶国荣", "zh_cn");
        /-------------------------------------------------------------------*/
         ; (function($) {
             $.extend({
                 chkStrByType: function(strS, chkType) {
                     var result;
                     switch (chkType) {
                         case 'odd'://奇数型
                             var chkStr = arrRegExp['number'];
                             var reg = RegExp(chkStr, 'g');
                             var result = reg.test(strS);
                             if (true == result) {
                                 var num = parseInt(strS) % 2;
                                 if (1 == num) {
                                     result = true;
                                 } else {
```

```
                        result = false;
                    }
                } else {
                    result = false;
                }
                break;
            case 'even':// 偶数型
                var chkStr = arrRegExp['number'];
                var reg = RegExp(chkStr, 'g');
                var result = reg.test(strS);
                if (true == result) {
                    var num = parseInt(strS) % 2;
                    if (num == 0) {
                        result = true;
                    } else {
                        result = false;
                    }
                } else {
                    result = false;
                }
                break;
            default:// 其他类型按正则表达式检测
                var chkStr = arrRegExp[chkType];
                var reg = RegExp(chkStr, 'g');
                var result = reg.test(strS);
                break;
        }
        return result;
    }
});
/*  正则验证字符串表达式  */
var arrRegExp = {};
arrRegExp['email'] = '\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*';
arrRegExp['telephone'] = '(\\(\\d{3,4}\\)|\\d{3,4}-|\\s)?\\d{7,8}';
arrRegExp['mobile'] = '(86)*0*1[3,5]\\d{9}';
arrRegExp['postcode'] = '^\\d{6}$';
arrRegExp['number'] = '^-?[0-9]\\d*$';
arrRegExp['zh_cn'] = '[\\u4e00-\\u9fa5]';
arrRegExp['url'] = '[a-zA-z]+://[^\\s]*';
})(jQuery);

$(function() {
    $("#btnChkStr").click(function() {
        //获取待检测的字符串与指定的类型
        var $ChkStr = $("#txtChkStr").val();
        var $ChkType = $("#selStrType").val();
        //保存检测后的结果值
        var blnResult = $.chkStrByType($ChkStr, $ChkType);
        //返回检测后的结果
        var strTmpShow = "";
        var strTmpType = blnResult ? " 是 " : " 不是 ";
        strTmpShow = $ChkStr + strTmpType;
        strTmpShow = strTmpShow + $("select :selected").text();
        strTmpShow = strTmpShow + " 类型 ";
        //将返回后的结果显示在页面中
```

```
                $("#divTip").show().html("").append(strTmpShow);
            });
        });
    </script>
</head>
<body>
    <fieldset><legend>指定类型检测字符串</legend>
        <div>
         <span>检测内容:</span>
             <input id="txtChkStr" type="text" class="txt" />
         <span>选择类型:</span>
         <select id="selStrType">
          <option value="email">邮箱</option>
          <option value="telephone">电话号码</option>
          <option value="mobile">手机号码</option>
          <option value="postcode">邮政编码</option>
          <option value="number">整数</option>
          <option value="zh_cn">汉字</option>
          <option value="url">网址</option>
          <option value="odd">奇数</option>
          <option value="even">偶数</option>
         </select>
         <input id="btnChkStr" type="button" value="检测" class="btn" />
         <div id="divTip"></div>
        </div>
    </fieldset>
</body>
</html>
```

10.9.4　代码分析

在本案例中，首先将执行的代码使用$.extend()方法进行封装，这样可以将代码独立成为一个功能性的插件，用户只需导入该插件，通过使用“$.方法名()”就可以进行调用，其封装框架代码如下所示：

```
; (function($) {
    $.extend({
      chkStrByType: function(strS, chkType) {
        //...执行代码
      }
    })
})(jQuery);
```

在执行代码中，根据所传回的字符串类型格式分别验证传回的字符串。验证通过时返回true值，否则返回false值，其部分代码如下所示：

```
//...省略部分代码
var result;
switch (chkType) {
   case 'odd':// 奇数型
      //...验证字符是否为奇数类型
     break;
   case 'even':// 偶数型
```

```
            //...验证字符是否为偶数类型
            break;
        default:// 其他类型按正则表达式检测
            //...结合正则表达式验证相应类型
            break;
    }
    return result;
//...省略部分代码
```

在本案例中，为了验证字符串的指定类型格式，先定义了一个数组，保存各种类型验证的正则表达式，其实现的代码如下所示：

```
//...省略部分代码
/*  正则验证字符串表达式  */
var arrRegExp = {};
arrRegExp['email'] = '\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*';
arrRegExp['telephone'] = '(\\(\\d{3,4}\\)|\\d{3,4}-|\\s)?\\d{7,8}';
arrRegExp['mobile'] = '(86)*0*1[3,5]\\d{9}';
arrRegExp['postcode'] = '^\\d{6}$';
arrRegExp['number'] = '^-?[0-9]\\d*$';
arrRegExp['zh_cn'] = '[\\u4e00-\\u9fa5]';
arrRegExp['url'] = '[a-zA-z]+://[^\\s]*';
//...省略部分代码
```

如果需要验证其他的字符串格式，只要将验证的正则表达式加入到该数组中即可，通过数组保存字符类型正则表达式后，就可以在执行代码中调用该数组中的某个元素值，从而获取某类型的正则表达式。

例如在验证是否是奇数时，先调用数组元素arrRegExp['number']，获取验证是否为整数的正则表达式，然后通过RegExp对象与正则表达式进行匹配，并将匹配的结果通过RegExp对象的test()方法进行检测，返回检测结果true或false值。如果返回值为

true，则将字符串通过parseInt()方法转成整数后与2求余数，如果有余数，则为奇数，否则为偶数。其实现的部分代码如下所示：

```
//...省略部分代码
var chkStr = arrRegExp['number'];
var reg = RegExp(chkStr, 'g');
var result = reg.test(strS);
if (true == result) {
    var num = parseInt(strS) % 2;
    if (1 == num) {
        result = true;
     } else {
        result = false;
     }
} else {
  result = false;
}
//...省略部分代码
```

如果只要根据传回的类型调出数组中类型的正则表达式，就可以检测字符串是否属于该类型，则执行下列代码：

```
//...省略部分代码
var chkStr = arrRegExp[chkType];
var reg = RegExp(chkStr, 'g');
var result = reg.test(strS);
//...省略部分代码
```

其中，参数chkType为调用chkStrByType函数时，输入的参数类型。

说明　在插件的执行代码中，RegExp是JavaScript中的一个重要对象，该对象可以对字符串与正则表达式进行模式匹配，格式为new RegExp(pattern,attributes)，其中参数pattern为正则表达式，参数attributes有三个值，"g"、"i"、"m"，分别表示全局、区分大小写、多行匹配。

10.10　本章小结

在jQuery中，众多实用的工具函数为程序开发人员在程序开发时带来方便，这也是jQuery被众多爱好者热衷于的原因之一。本章首先介绍工具函数的概念与分类，通过一些简洁明了的应用示例，使读者能够快速上手使用这些常用的工具函数，并能够做到举一反三，使用扩展工具函数开发出自己的工具函数，提高代码开发效率。只有理解基础概念，加之不断练习，才能真正掌握其实质。

第11章　jQuery常用开发技巧

本章内容

快速控制页面元素

使用工具函数$.support检测浏览器的信息

调用jQuery中的方法

巧用jQuery中的事件

jQuery集合处理功能

常用自定义方法与插件

本章小结

初次学习使用jQuery框架开发Web页面的读者，在开发过程中，会对框架的灵活运用和代码技巧上存在一些不足；本章介绍日常使用jQuery框架处理和开发页面的过程中，一些常用的开发技巧与解决方案。通过这些方法的介绍，帮助和引导读者在开发过程中积累开发经验和技巧，优化代码质量和性能，从而不断提升整体项目的开发水平。

11.1　快速控制页面元素

在日常开发过程，使页面中某一元素，如＜div＞居中是最常见的需求，此外，有时还需要根据鼠标单击时的坐标位置，决定是否显示或隐藏一个页面元素，这些都是属于控制页面元素的范畴，如果是各项目都会经常用到，且与业务无关联需求，那么，可以使用编写插件的方式进行封装，如＜div＞居中弹出框的效果。下面我们通过一个个示例来详细介绍各类控件页面元素需求的解决方案。

11.1.1　居中显示元素

要使元素在屏幕中居中，先要将该元素的"position"定位属性值设置为"absolute"，表示绝对定位；然后通过设置"top"、"left"属性值，使元素居中在屏幕中。由于这样的页面效果常用于各个项目中，因此，可以将该功能编写成一个jQuery插件，便于以后的调用，接下来介绍实现元素在屏幕居中的完整过程。

示例11-1　居中显示弹出框

（1）功能描述

在新建的页面中，使用＜div＞元素创建一个弹出框，该弹出框由主题、内容、关闭按钮三部分组成。当页面加载时，弹出框元素调用

jQuery插件中的center()方法，以渐进的方式显示在屏幕中央，单击“关闭”按钮时，弹出框以渐隐的方式隐藏。

（2）实现代码

新建一个HTML文件11-1.html，加入如代码清单11-1所示的代码。

代码清单 11-1　居中显示弹出框

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使元素在屏幕中居中</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Plugs/11-1.js"
            type="text/javascript"></script>
    <style type="text/css">
        body{font-size: 13px;}
        .frame{width:276px; border: solid 1px #ccc;display:none;}
        .frame .title{font-weight: bold; width: 260px;padding: 8px; background-color: #eee}
        .frame .content{padding: 15px 8px;}
```

```
            .fl{float: left}
            .fr{float: right}
            .cls{clear: both}
        </style>
        <script type="text/javascript">
            $(function() {
                //弹出框元素调用插件中的center()方法
                $(".frame").center().show(1000);
                //设置单击关闭按钮时触发的动作
                $(".title").find("img").bind("click", function()
                {
                    //以渐隐的方式隐藏弹出框
                    $(".frame").hide(1000);
                });
            });
        </script>
    </head>
    <body>
        <div class="frame">
            <div class="title fl">
                <div class="fl">
                    主题
                </div>
                <div class="fr">
                    <img src="Images/close.png" />
                </div>
            </div>
            <div class="content cls">
                内容
            </div>
        </div>
    </body>
    </html>
```

在代码清单11-1中，通过页面中的＜head＞元素导入了一个名称为“11-1”的JS文件。在该JS文件中，自定义了一个名为"center"的插件，在插件的代码中，分别对弹出框元素的"position"、"top"、"left"属性值进行了居中效果的设置，其实现的代码如下所示：

```
jQuery.fn.center = function() {
    this.css("position", "absolute");
    this.css("top", ($(window).height() - this.height()) / 2+
            $(window).scrollTop() + "px");
    this.css("left", ($(window).width() - this.width()) / 2 +
            $(window).scrollLeft() + "px");
    return this;
}
```

（3）页面效果

代码执行后的页面效果如图11-1所示。

图　11-1　居中显示弹出框

（4）代码分析

在插件的JavaScript代码中，为了使"this"元素居中，首先，将"position"属性值设置为"absolute"，表示元素是一个绝对定位元素。其次，设置"this"元素的"top"属性，代码如下：

```
$(window).height() - this.height()) / 2+ $(window).scrollTop()
```

上述代码表示将整个屏幕的高度减去"this"元素自身高度的一半并加上屏幕纵向滚动条拉动的距离，该数值就是"this"元素居中时的"top"属性值；同理，获取"this"元素居中时的"left"属性值，并通过调用jQuery框中的CSS方法进行设置，最后，使用"return"语句返回已完成居中效果设置的"this"元素。

在页面中，可以像其他jQuery框架中的方法一样，实现对"center"插件的调用，代码如下所示：

```
$(".frame").center().show(1000);
```

上述代码表示弹出框元素调用"center"插件并以1000毫秒的速度效果居中显示。

说明　在页面的JavaScript代码中，虽然实现了弹出框的居中显示，但当屏幕大小发生变化时，弹出框并不能随之居中，因此，还需要在浏览器的"resize"事件中再次调用插件，即添加"$(window).resize(function(){$(".frame").center();})"代码。

11.1.2　捕获鼠标位置

当用户在页面中，单击当前图片的左半部分时，可以查看上一张图片，单击右半部分时，可以查看下一张图片，从而实现多张图片在页面中的简单浏览。而要想实现该功能，通常需要捕获单击图片时鼠标的位置，根据该位置来判断是在图片的左半部分还是右半部分发生的单击，从而决定是显示上一张还是下一张图片。

接下来通过一个完整实例介绍该功能实现的过程。

示例11-2　捕获鼠标位置

（1）功能描述

在新建的页面中，添加一个＜img＞元素用于加载图片，当用户单击图片的左半部分时，将在＜img＞元素中显示上一张图片，单击图片的右半部分时，显示下一张图片。同时，在显示图片的上方，实时显示当前鼠标的位置，即鼠标相对图片的x、y轴值。

（2）实现代码

新建一个HTML文件11-2.html，加入如代码清单11-2所示的代码。

代码清单 11-2　捕获鼠标位置

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>捕获鼠标位置</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
        body{font-size: 13px;}
        .frame{text-align:center}
        .frame .content{margin:8px}
        .frame .img{padding:3px;border:solid 1px #ccc}
        .frame .content .tip{margin:5px}
    </style>
    <script type="text/javascript">
        //定义装载图片的数组变量
        var arrImg = new Array("Images/2010.jpg","Images/2012.jpg");
        //定义并初始化中间变量
        var intCur = 0,intR = 0;
        $(function() {//在图片中移动鼠标事件
            $(".img").mousemove(function(e) {
                var positionX = e.originalEvent.x - $(this).offset().left ||
                                e.originalEvent.layerX - $(this).offset().left || 0;
                var positionY = e.originalEvent.y - $(this).offset().top ||
                                e.originalEvent.layerY - $(this).offset().top || 0;
                var intX = parseInt(positionX);
                var intW = parseInt($(this).width() / 2);
                intR = (intX > intW) ? 1 : 0;
                $(".tip").html("X:" + positionX + "," + "Y:" + positionY);
            }).click(function() { //图片点击事件
                if (intR) {
                    if (intCur <= arrImg.length) {
                        intCur = intCur + 1;
                    }
                } else {
                    if (intCur > 0) {
                        intCur = intCur - 1;
                    }
                }
                //加载所选择的图片
                $(".img").attr("src", arrImg[intCur]);
            });
        });


    </script>
</head>
<body>
    <div class="frame">
        <div class="content">
            <div class="tip">X:0,Y:0</div>
            <a href="javascript:">
                <img src="Images/2010.jpg" class="img" />
            </a>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-2所示。

图　11-2　捕获鼠标位置

（4）代码分析

在本示例中，首先通过一个名称为"arrImg"的数组变量，保存所有的浏览图片地址。然后，编写浏览图片时，图片触发的"mousemove"、"click"事件，在图片的"mousemove"事件中，将获取鼠标的x、y轴的位置分别保存在变量"positionX"、"positionY"中。

在获取位置时，要注意浏览器的兼容性，在IE系列浏览器下，使用"e.originalEvent.x"代码，而在Firefox系列浏览下，只能使

用"e.originalEvent.layerX"代码获取鼠标在x轴上的相对距离。当分别保存鼠标移动时x、y轴的位置后，将x轴的距离与图片二分之一些的宽度进行比较，如果大于，说明是在图片的右半部分，否则，是在图片的左半部分，并将该标志保存在变量"intR"中。

在图片的"click"事件中，根据标志变量"intR"的值，设置图片索引号变量"intCur"的值。如果鼠标单击的是右半部分即显示下一张图片，则检测变量"intCur"是否小于数组的长度，如果成立，则将该值加1；如果鼠标单击的是左半部分即显示上一张，则检测变量"intCur"是否大于0，如果成立，则将该值减1。

最后，将变量"intCur"的值作为数组"arrImg"的索引号调用attr()方法，设置页面中＜img＞元素的"src"属性值，最终实现根据鼠标位置不同加载不同图片的效果。

11.2　使用工具函数$.support检测浏览器的信息

本书前面的章节中介绍过使用$.browser工具函数获取浏览器信息的方法，随着jQuery框架的不断升级，新方法和工具函数不断出现。在最新版本中，可以使用jQuery.support或$.support工具函数获取浏览器的信息，该函数为jQuery框架中的内部函数，检测各浏览器的某些功能是否可用。通过该函数的检测，可以更加清晰地了解各个浏览器之间的区别，为处理好代码在各个浏览器中的兼容性提供依据。接下来通过一个示例介绍该功能实现的过程。

示例11-3　使用工具函数$.support检测浏览器的信息

（1）功能描述

在新创建的页面中，使用$.support工具函数检测当前浏览器对各项特性支持状态，如果支持则为true，否则为false，被检测的各项以列表的方式展示在页面中。

（2）实现代码

新建一个HTML文件11-3.html，加入如代码清单11-3所示的代码。

代码清单 11-3　使用 $.support 检测浏览器的信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用$.support检测浏览器的信息</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var list = $.support;
            var strTmp = "";
            var i = 0;
            for (key in list) {// 遍历各项检测特征
                i++;
                strTmp += i + '.' + key + '=' + list[key] + '</br>';
            }
            $("#divTip").html(strTmp);// 将检测完的结果显示在页面中
        });
    </script>
</head>
<body>
<div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-3所示。

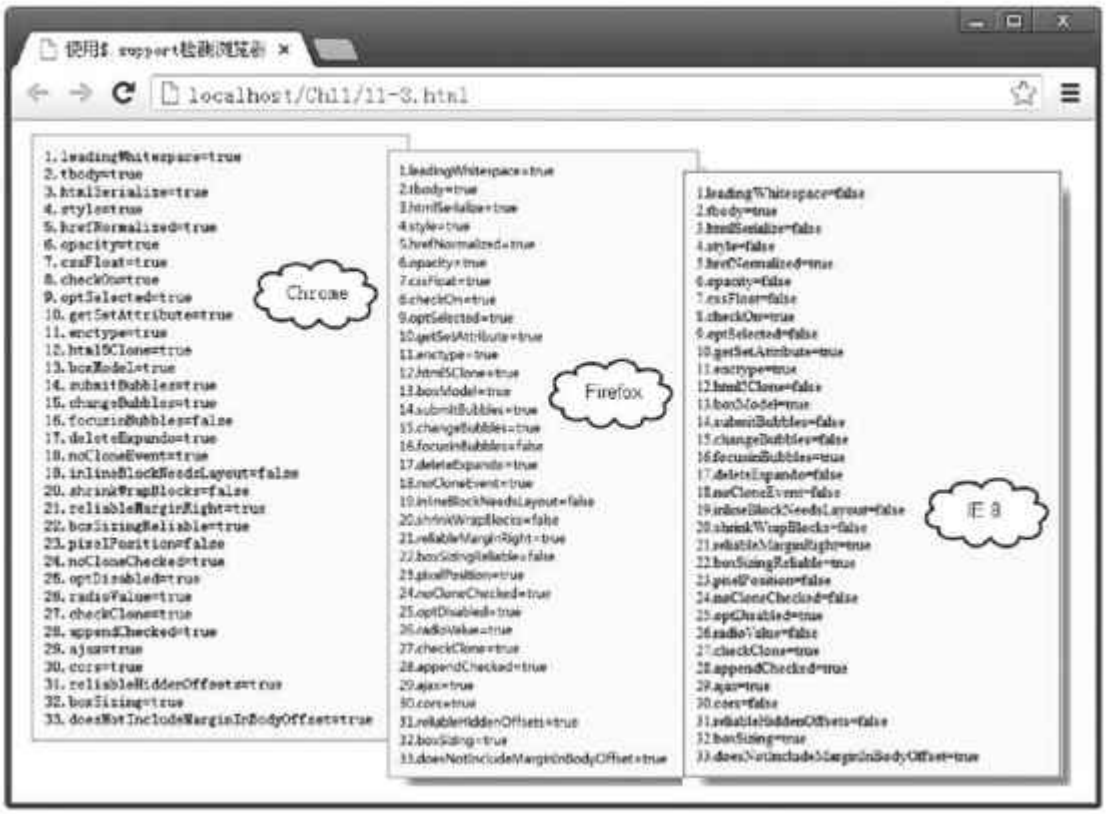


图　11-3　使用工具函数$.support检测浏览器的信息

（4）代码分析

图11-3展示了Chrome、Firefox、IE三大主流浏览器的检测结果，从示意图我们不难看出，使用$.support工具函数一共检测了33项内容，其中Chrome和Firefox浏览器支持的特征比IE 8要多一些。

如第一项"leadingWhitespace"表示在使用"innerHTML"进行赋值时，是否保留前面的空白符。Chrome和Firefox浏览器中为true，表示保留，IE 8为false，表示不保留。另外，第四项"style"表示是否可

以通过"style"属性获取DOM Element的样式。Chrome和Firefox浏览器中为true，而IE 8为false。更多主要检测项说明如表11-1所示。

表 11-1 $.support 主要检查项支持说明

检查项名称	描述	Chrome	Firefox	IE 8
leadingWhitespace	使用“innerHTML”进行赋值时，是否保留前面的空白符	true	true	false
tbody	是否自动为 table 元素插入 tbody 标签	true	true	true
htmlSerialize	link 标签是否被正确地序列化	true	true	true
style	是否可以通过“style”属性获取 DOM Element 的样式	true	true	false
hrefNormalized	href 属性是否被正确地序列化	true	true	true
opacity	检测 CSS 样式中的透明属性设置是否被支持	true	true	false
cssFloat	检测 CSS 样式中的 float 属性设置是否被支持	true	true	false

（续）

检查项名称	描述	Chrome	Firefox	IE 8
checkOn	检测 checkbox 元素的 value 属性值是否为 on	true	true	true
optSelected	检测 select 元素中的第一项是否默认被选中	true	true	false
getSetAttribute	检测是否可以通过 getAttribute 和 SetAttribute 方式获取和设置元素的属性值	true	true	true
boxModel	检测页在渲染时是否符合 W3C Box Model 模式	true	true	true
deleteExpando	检测是否允许删除附加在 DOM Element 上的数据	true	true	false
noCloneEvent	检测复制 DOM Element 时，是否与 event 一起进行复制	true	true	false
inlineBlockNeedsLayout	将原来 display 为 block 的 DOM Element 设置为 display:inline 时，形式是否一致	false	false	false
shrinkWrapBlocks	检测内部的 DOM Element 样式是否会影响外部的 DOM Element 样式	false	false	false
reliableMarginRight	检测使用 margin-right 的计算是否正确	true	true	true
noCloneChecked	检测在复制 checkbox 元素时，是否与其选中的状态一起复制	true	true	false
optDisabled	属性值为 disable 的 select 元素，其内部的 option 选项属性值是否也为 disable	true	true	true
radioValue	检测 input 元素的类型被设置为 radio 后，是否还保持原有的值	true	true	false
checkClone	检测在 fragment 中，checkbox 的选中状态是否可以被复制	true	true	true
appendChecked	检测在将 checkbox 元素添加到 DOM 中时，是否还保持原有的选中状态	true	true	true
ajax	检测是否支持 Ajax 请求	true	true	true
cors	检测是否支持跨域的 Ajax 请求	true	true	false
reliableHiddenOffsets	检测在元素 hidden 状态下，offsetWidth 和 offsetHeight 是否正确	true	true	false

11.3　调用jQuery中的方法

jQuery框中提供了大量功能强大的方法，开发人员在编写代码时，需要有选择性地使用，针对不同的需求，调用效率最高的方法，此外，在调用这些方法的过程中，也需要注意代码编写时的一些技巧，接下来我们通过三个示例进行说明。

11.3.1　使用预加载方法预览图片

在日常的Web页面开发中，使用图片是最常用的一种方式，但有些体积较大的图片，称之为“富”图片，如果在浏览器加载时不作任何处理，其速度不仅会很慢，而且会影响用户的UI体验。那么，对于这样的图片在加载时，需要考虑使用预加载方法。预加载是指图片在显示之前，浏览器已经完成了图片的下载和缓存，因此，图片经过预加载后，再进行显示，其速度和UI体验都会得到很好的提升。

接下来通过一个完整示例介绍该功能实现的过程。

示例11-4　页面打开时使用预加载图片

（1）功能描述

在新创建的页面中，使用图片预加载方法制作一个多张图片浏览工具——通过页面中的＜img＞元素首先显示页面初始化时的图片，当用户点击该元素左右两边的导航图标时，将在＜img＞元素中使用预加载方式显示当前图片的上一张和下一张。

（2）实现代码

新建一个HTML文件11-4.html，加入如代码清单11-4所示的代码。

代码清单 11-4　页面打开时使用预加载图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>页面打开时使用预加载图片</title>
    <style type="text/css">
        .box{font-size:13px;margin:0 auto;text-align:center;height:200px;width:216px}
        .imgbg{float:left;overflow:hidden;height:200px;
        width:150px;background:#fff url(images/loading.gif) center center no-repeat}
        .imgpl, .imgnl{height:200px;width:22px}
        .imgpl{float:left;padding-right:11px}
        .imgnl{float:right;padding-left:11px}
        .imgpl:hover,.imgpl:visited:hover{background:url(images/left.jpg) left 50%
        no-repeat;
        filter: alpha(opacity=60);-moz-opacity: 0.6;-khtml-opacity:
        0.6;opacity: 0.6}
        .imgnl:hover,.imgnl:visited:hover{background:url(images/right.jpg)
        right 50% no-repeat;
        filter: alpha(opacity=60);-moz-opacity: 0.6;-khtml-opacity:
           0.6;opacity: 0.6}
        .tip{clear: both;height: 23px;line-height: 23px}
    </style>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Plugs/11-4.js"
            type="text/javascript"></script>
    <script type="text/javascript">
        var arrImg = new Array("Images/pics/2008.jpg",
            "Images/pics/2009.jpg",
            "Images/pics/2010.jpg",
            "Images/pics/2011.jpg",
            "Images/pics/2012.jpg");
        $(function() {
            var $imgShow = $("#imgShow");
            var i = 0;
```

```
            //图片的最大数 (总数减1)
            var max = arrImg.length - 1;
            /*下一张*/
            $(".imgnl").bind("click", function() {
                i++;
                if (i > max) {
                    i = 0;
                }
                loadPic(i);
                return false;
            });
            /*上一张*/
            $(".imgpl").bind("click", function() {
                i--;
                if (i < 0) {
                    i = max;
                }
                loadPic(i);
                return false;
            });
            //进入页面初始化
            loadPic(0);
            function loadPic(i) {
                //应用图片预加载技术
                $imgShow.beforeload({ "src": arrImg[i] });
                $(".tip").html("第 " + parseInt(i + 1) + " 张");
            }
        });
    </script>
</head>
<body>
    <div class="box">
        <div class="imgpl">
        </div>
        <div class="imgbg">
            <img id="imgShow" />
        </div>
        <div class="imgnl">
        </div>
        <p class="tip">
        </p>
    </div>
</body>
</html>
```

在代码清单11-4中，当页面加载时，导入了一个名为11-4的JS文件，在该文件中，以自定义插件的方式扩展了一个beforeload()方法，实现图片的预加载功能。该文件实现的代码如下所示：

```
jQuery.fn.beforeload= function(options) {
    options = $.extend({
        src: ""
    }, options);
    var self = this;
    self.hide();

    var img = new Image();
    $(img).load(function() {
        self.attr("src", options.src);
        self.fadeIn("slow");
    }).attr("src", options.src);
    return _self;
}
```

（3）页面效果

代码执行后的页面效果如图11-4所示。

图　11-4　页面打开时使用预加载图片

（4）代码分析

在本示例中，首先定义一个名为arrImg的数组变量，用于保存图片的来源，然后，分别编写点击查看“上一张”和“下一张”图标的点击事件。在编写过程中需要注意，如果查看上一张图片，在当前图片索引号变量"i"减1后，需要判断"i"的值是否小于0，如果成立，则将"i"设置为图片的总数量，即从第一张返回最后一张，实现循环查看的功能。如果查看下一张图片，同样需要判断"i"的值，而此次是检测

它的值是否大于图片的总量，如果成立，则将"i"设置为0，实现循环查看的功能。

在“上一张”和“下一张”图标的点击事件中，每次都调用了一个自定义函数loadPic，该函数的功能是通过图片预加载的方式打开指定索引号的图片，并在页面中展示当前图片的索引号信息。

在该函数中，调用了jQuery自定义插件中的beforeload()方法来实现图片预加载的功能。在beforeload()方法中，在隐藏图片元素的前提下，首先调用load()方法实现图片的预加载，然后调用attr()设置图片的"src"属性值，这样的先后顺序使图片能通过预加载方式显示在页面中。

11.3.2　通过html()方法判断元素是否为空

在jQuery中，html()方法既可以设置元素中需要显示的内容，又可以获取元素中已显示的内容。例如，在页面中添加一个ID号为"tip"的＜div＞元素，执行如下JavaScript代码操作：

```
$(function() {
      var $tip = $("#tip"); //①
      $tip.html("hello!");  //②
});
```

在上述代码中的第①行，将jQuery对象先保存在一个名为"$tip"的变量中，用于后续的使用和缓存对象。在第②行中，通过html()方法设置了＜div＞元素显示的内容为"hello!"。执行上述代码后，在页面的＜div＞元素内显示"hello!"字样。

html()方法除设置元素内容外，还可以获取指定元素中的HTML内容，在上面代码的基础之上，添加如下一行JavaScript代码：

```
alert($tip.html());//③
```

上述代码通过html()方法获取＜div＞元素中所显示的HTML内容，并以alert()弹出框的方式显示。执行上述代码后，将弹出一个对话

框，显示ID号为"tip"的＜div＞元素中的内容。除了设置和获取元素中的内容外，html()方法还可以检测一个元素是否为空，在上述代码的基础之上，再次添加如下代码：

```
if ($tip.html()) {//④
    alert($tip.html());//⑤
}
```

上述代码中的第④行是一个条件语句，即通过html()方法检测元素是否为空。如果html()方法用在条件语句中，它有两层意义，一是表示指定的＜div＞元素是否存在，另一层表示该元素中是否存在内容。当这两个条件都具备时，该语句返回的值为true，如果指定的页面元素不存在或元素中没有任何内容，该语句都将返回false值。根据本示例的上下文代码，本次执行将返回true，将执行第⑤行代码，弹出一个显示＜div＞元素内容的对话框。

除使用html()方法检测元素是否为空外，通过检测元素的length值是否大于0，也可以判断该元素是否为空，在上述代码的基础之上，添加如下代码：

```
if ($tip.length > 0) {//⑥
    alert($tip.html());//⑦
}
```

上述代码的第⑥行，通过判断元素的length属性值是否大于0，检测元素是否为空。如果不为空，则执行第⑦行代码。根据本示例的上下文代码，第⑥行的条件语句将返回一个true值，并执行第⑦行代码，同样也弹出一个显示＜div＞元素内容的对话框。

虽然使用元素的html()方法和length属性都可以检测该元素是否存在，但两者在使用时存在一个很大的差别，使用html()方法不仅可以检测元素是否存在，还可以查看元素中是否包含内容，而length属性仅是判断元素是否在页面中存在，而不检测其内容。因此，当将上述示例的第②行代码注释后，第④行代码将返回false，而第⑥行代码将仍然返回true。

11.3.3　使用replace()和replaceWith()方法替换内容

在jQuery中，replace()方法的功能是在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子字符串，调用格式为：

```
strObject.replace(regexp/substr,newstr)
```

上述代码中，strObject表示一个字符串对象，regexp/substr表示与正则表达式匹配的字符串或子符串，newstr表示替换后的新字符串内容。replace()方法所替换的对象是字符串，而使用replaceWith()方法则可以替换元素，replaceWith()方法的功能是用指定的HTML内容或元素替换被选元素，调用格式为：

```
$(selector).replaceWith(content)
```

上述代码中，$(selector)表示需要被替换的元素，content表示被替换的内容或元素。replaceWith方法所替换的对象是元素或元素中的内容。接下来通过一个示例来介绍结合replace()和replaceWith()这两个方法实现文本内容中以高亮方式显示被查询内容的效果。

示例11-5　使用replace()和replaceWith()方法替换内容

（1）功能描述

在新创建的页面中，显示一个段落文本内容，当用户在文本框中输入需要在段落中查找的内容并单击“查询”按钮后，所找到的内容将以高亮的形式显示在段落中，而当重新查找时，原来显示的高亮效果将被自动清除。

（2）实现代码

新建一个HTML文件11-5.html，加入如代码清单11-5所示的代码。

代码清单 11-5　使用 replace 和 replaceWith 方法替换内容

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 replace 和 replaceWith() 方法替换内容 </title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
            body{font-size:13px}
            p{border:1px solid #666;width:280px;padding:8px;line-height:1.7em;
            font-size:13px}
            .spn{padding-left:20px}
            focus{ background-color:#ccc;color:#fff}
            .txt{border:#666 1px solid;padding:3px}
            .btn{border:#666 1px solid;padding:2px;width:60px;
            filter: progid:DXImageTransform.Microsoft
            .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
   </style>
  <script type="text/javascript">
        $(function() {
            var search = {
                searchstr: function() {
                    search.clearstr;
                    var searchText = $('#txtSearch').val();
                    var regExp = new RegExp(searchText, 'g');
                    if (searchText != "") {
                        $('p').each(function()
                        {
                            var newHTML = $(this).html().replace(regExp,
                            '<span class="focus">' + searchText + '</span>');
                            $(this).html(newHTML);
                        });
                    }
                },
```

```
                clearstr: function() {
                    $('p').each(function()
                    {
                        $(this).find('.focus').each(function()
                        {
                        $(this).replaceWith($(this).html());
                        });
                    });
                }
            }
            $('#btnSearch').bind("click", search.searchstr);
        });
    </script>
</head>
<body>
    <div>
        <input id="txtSearch" type="text" class="txt"  />
        <input id="btnSearch" type="button"
               class="btn" value=" 查询 " />
    </div>
    <p>
        <span class="spn"></span>jQuery 发布于 2006 年，在后续的版本升级中，广大的开发者被其简洁的代码，强悍的功能、优雅的展现、强大的兼容所折服，不断有人加入其阵营，而恰在那年，也深深吸引了我的眼光，从此深入其中。<br />
        <span class="spn"></span>为什么会有如此多的人钟爱 jQuery，与其超大的功能是分不开的，目前的 Web 项目开发，不仅仅是基于功能上的考虑，而更注重于用户使用体验与页面静态优化，这是一个理性的回归，也是技术发展的必然趋势，而 jQuery 恰恰是实现这一趋势的坚实利刃，并且可以在最大程度上满足各类语言开发 Web 页面的使用。
    </p>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-5所示。

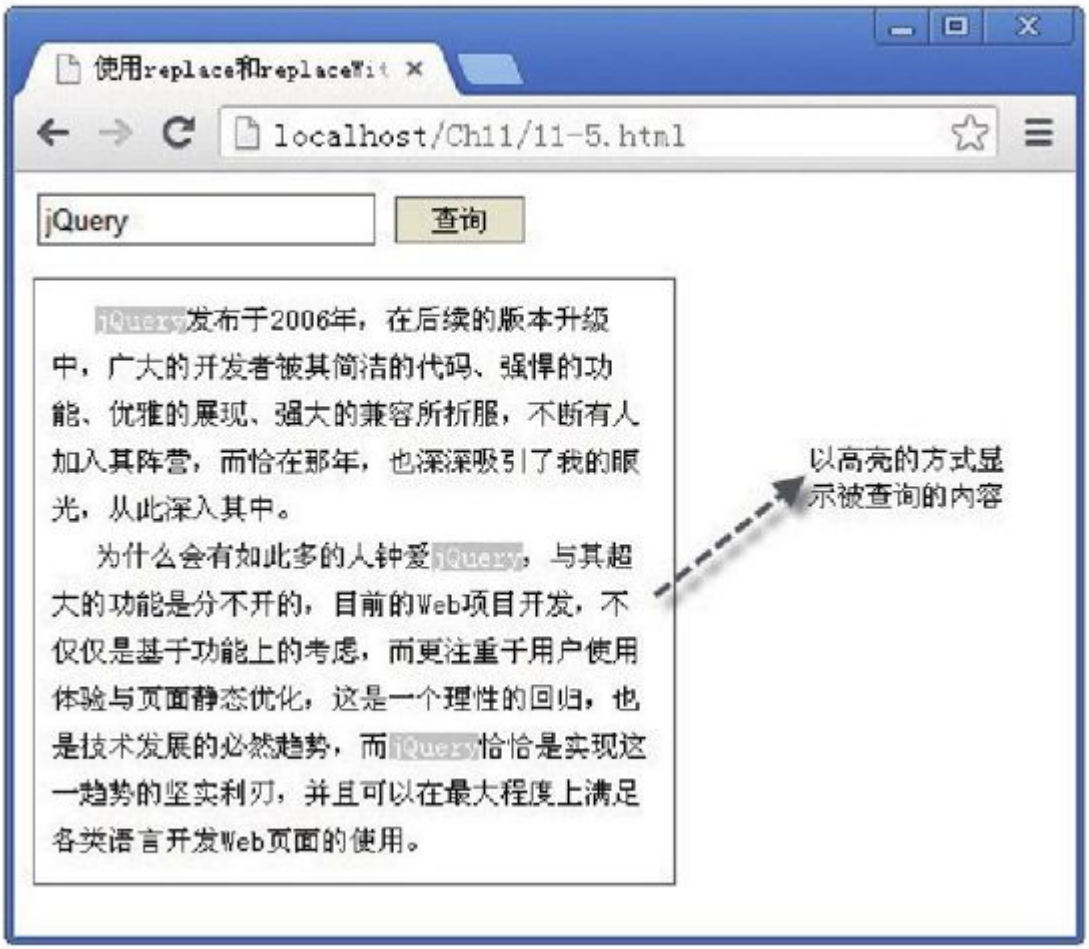


图　11-5　使用replace和replaceWith方法替换内容

（4）代码分析

在本示例的JavaScript代码中，当用户单击“查询”按钮时，将调用自定义的函数searchstr()。在该函数中，为了清空上一次高亮显示的内容，需要执行自定义的高亮清除函数clearstr。在这个clearstr()函数体中，先遍历＜p＞元素查找高亮类型的元素，并再次

遍历这些找到的高亮类型元素，使用replaceWith方法将原有高亮类型元素替换成无类型元素，从而实现清除高亮效果的功能。

在执行完高亮清除函数clearstr后，接下来将获取的查询内容创建成正则表达式，在遍历段落内容时，使用replace方法将查询内容替换成带有高亮样式的字符内容，并更新该操作，从而实现以高亮的效果显示段落中查询内容的功能。

11.4　巧用jQuery中的事件

在jQuery框架中可以很方便地绑定页面元素的各类事件，通过这些事件的触发，实现一个个相应的功能，事件驱动代码的执行，因此，在进行元素事件绑定时，需要注意一些常用的技巧，接下来通过两个简单的示例来进行说明。

11.4.1　开启或禁止页面右键菜单

在进行Web页开发过程中，有时需要禁止页面的右键菜单，而实现这一功能的方法在jQuery中非常简单，只需要在页面的"contextmenu"事件中返回false即可。除此之外，由于在该事件中，还可以传递一个"e"对象，根据这一对象，可以检测用户在电脑中的按键类型，如功能键等。

接下来通过一个简单示例的开发详细介绍这一功能实现的过程。

示例11-6　开启或禁止页面右键菜单

（1）功能描述

在新创建的页面中，通过一个复选框开启页面右键菜单的设置功能，即只有当复选框选中时，才能开启页面右键菜单功能，否则为禁

止状态；在开启状态下，通过两个单选框，可以选择单击右键和Ctrl键加单击右键组合方式来开启页面右键菜单。

（2）实现代码

新建一个HTML文件11-6.html，加入如代码清单11-6所示的代码。

代码清单 11-6　开启或禁止页面右键菜单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>开启或禁止页面右键菜单</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
            body{font-size:13px}
            fieldset{width:260px}
            legend{color:#555;font-weight:bold}
            .div1{padding:5px}
            .div2{padding:5px 15px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(document).bind("contextmenu", function(e) {
                if ($("#chkBlnRightKey").attr("checked") == true) {
                    if ($("#Radio1").attr("checked") == true)
                        return true
                    else {
                        if (!e.ctrlKey)
                            return false;
```

```
                    else
                        return true
                }
            }
            else
                return false;
        });
        $("#chkBlnRightKey").bind("change", function() {
            if ($(this).attr("checked") == true) {
                $("#Radio1,#Radio2").attr("disabled", false);
                $("#Radio1").attr("checked", true);
            }
            else {
              $("#Radio1,#Radio2").attr({ "disabled": true,"checked": false });
            }
        });
    });
    </script>
</head>
<body>
    <fieldset>
        <legend>设置</legend>
        <div class="div1">
            <input id="chkBlnRightKey" type="checkbox" />是否开启网页右键菜单
            <div class="div2">
                <input id="Radio1" type="radio"
                       name="rdoOpen" disabled="disabled"/>按鼠标右键开启 <br />
                <input id="Radio2" type="radio"
                       name="rdoOpen" disabled="disabled"/> 按Ctrl+鼠标右键开启
            </div>
        </div>
    </fieldset>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-6所示。

图　11-6　开启或禁止页面右键菜单

（4）代码分析

在本示例的JavaScript代码中，复选框绑定了一个"change"事件。在该事件中，当复选框选中时，两个单选按钮则变为可用，同时，第一个单选按钮为选中状态；当复选框未选中时，两个单选按钮变为不可用，且都为未选中状态。

在页面绑定的"contextmenu"事件中，首先检测复选框是否被选中，如果没有选中，则直接返回false，如果选中，则进一步查看两个单选按钮的选中状态。如果是第一个单选按钮选中，表示正常开启页面的右键菜单，则直接返回true即可；如果是第二个单选按钮选中，则需要根据"e.ctrlKey"获取检测用户是否按下了Ctrl键，如果按下了，则直接返回true，否则返回false。通过上面操作，实现了管理页面右键菜单的功能。

11.4.2　限制文本输入框中字符的数量

在使用文本输入框接收用户输入的内容时，通常都需要限制文本框接收的字符数量，例如微博输入框，限制输入字符数量在140以内，超出时将会出现错误提示信息，并且发送按钮不可用。为了能限制文本输入框中输入字符的数量，需要在输入框的"onkeypress"、"onkeyup"、"onchange"事件中检测输入内容的长度，并对超出部分进行截取，从而实现无法输入超出字符内容的效果。接下来我们开发一个限制文本输入框中字符数量的jQuery插件。

示例11-7　限制文本输入框中字符数量

（1）功能描述

在新创建的页面中，添加一个用于留言的＜textarea＞元素，当用户输入的字符数量等于指定的字符长度140以后，则在键盘中无法再输入新增的内容。

（2）实现代码

新建一个HTML文件11-7.html，加入如代码清单11-7所示的代码。

代码清单 11-7　限制文本输入框中字符的数量

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>限制文本输入框中字符的数量</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Plugs/11-7.js"
            type="text/javascript"></script>
    <style  type="text/css">
            body{font-size:13px}
            .frame{width:260px}
            .title{padding:5px 0px}
            .txt{height:108px;width:252px;border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $('#txtaMessage').maxLength(140, "divMaxNum"); ;
        });


    </script>
</head>
<body>
    <div class="frame">
        <div class="title">我要留言</div>
        <textarea id="txtaMessage" class="txt"></textarea>
        <div align="right" id="divMaxNum">140</div>
    </div>
</body>
</html>
```

在代码清单11-7中，当页面加载时，导入了一个名为11-7的JS文件，在该文件中，以自定义插件的方式扩展了一个maxLength()方法，实现限制文本输入框中字符数量的功能，该文件实现的代码如下所示：

```
jQuery.fn.maxLength = function(max,ele) {
    this.each(function() {
        var type = this.tagName.toLowerCase();
        var inputType = this.type ? this.type.toLowerCase() : null;
        if (type == "input" && inputType == "text"
            || inputType == "password") {
            this.maxLength = max;
        }
        else if (type == "textarea") {
            this.onkeypress = function(e) {
                var ev = e || event;
                var keyCode = ev.keyCode;
                return !(this.value.length >= max
                && (keyCode == 32 || keyCode == 13)
                && !ev.ctrlKey && !ev.altKey);
            };
            this.onkeyup = function() {
                if (this.value.length > max) {
                    this.value = this.value.substring(0, max);
                }
                $("#"+ele).html(max - this.value.length);
            };
            this.onchange = this.onkeyup;
        }
    });
};
```

（3）页面效果

代码执行后的页面效果如图11-7所示。

图　11-7　限制文本输入框中字符的数量

（4）代码分析

在本示例的JavaScript代码中，当页面加载时，将调用maxLength()方法来限制文本输入框的字符数量。在自定义的插件代码中，为了实现文本输入框中字符数量的限制，首先，需要检测文本输入框元素的类型，如果是"text"和"password"类型的输入框，只需将输入框元素的"max"属性值设置为限制的字符总量值。

如果输入框的类型为"textarea"，需要先绑定输入框的"onkeypress"事件。在该事件中，当用户输入的字符总量超出指定的长度后，按回车或空格键均返回false，即无法再从光标处输入新的字符内容。然后，绑定输入框的"onkeyup"和"onchange"事件，在这两个事件中，如果输入框的字符总量超出指定的长度后，则通过substring方法截取指定长度内的字符，从而实现限制文本输入框字符总量的功能。

11.5　jQuery集合处理功能

在jQuery中，对于jQuery返回的集合元素，如果需要访问其中的某一项，无须再次进行遍历，通过下面格式便可以快速访问：

```
[按索号排列的各种效果][索引号]
```

根据上述代码的格式编写如下代码：

```
this.style.color = ['red','green','blue'][i]
```

上述代码表示元素集合的第1、2、3位元素的字体颜色分别为"red"、"green"、"blue"，接下来通过一个简单的示例来演示这一功能的实现过程。

示例11-8　jQuery集合处理功能

（1）功能描述

在新创建的页面中，使用＜table＞元素创建一个集合列表，并使用jQuery提供的集合处理功能实现前三行的字体大小不一样，隔行变色的页面效果。

（2）实现代码

新建一个HTML文件11-8.html，加入如代码清单11-8所示的代码。

代码清单 11-8　jQuery 集合处理功能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>jQuery 集合处理功能 </title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           #tbStu{width:260px;border:solid 1px #666;background-color:#eee}
           #tbStu tr{line-height:23px}
           #tbStu tr th{background-color:#ccc;color:#fff}
    </style>
    <script type="text/javascript">
        $(function() {
            $("tr:not(:first)").each(function(i) {
                this.style.fontSize = ['13px', '14px', '15px'][i]
                this.style.backgroundColor = ['#eee', '#fff'][i % 2]
            });
        });
    </script>
</head>
<body>
<table id="tbStu" cellpadding="0" cellspacing="0">
  <tbody>
    <tr>
      <th> 学号 </th><th> 姓名 </th><th> 性别 </th><th> 总分 </th>
    </tr>
    <tr>
      <td>1001</td><td> 张小明 </td><td> 男 </td><td>320</td>
    </tr>
    <tr>
      <td>1002</td><td> 李明琪 </td><td> 女 </td><td>350</td>
    </tr>
    <tr>
      <td>1003</td><td> 刘博古 </td><td> 男 </td><td>450</td>
    </tr>
    <tr>
      <td>1004</td><td> 陈小早 </td><td> 女 </td><td>530</td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-8所示。

图　11-8　jQuery集合处理功能

（4）代码分析

在本示例的JavaScript代码中，使用"tr:not(:first)"方式获取除第一行标题外的全部元素集合并进行遍历。在遍历过程中，采用jQuery自带的集合处理功能，设置表格中的前三项元素的字体大小分别为"13px"、"14px"、"15px"，同时，采用"i%2"方式判断集合中所在行的奇偶性，并在前面的方括号中设置对应的背景色，从而实现隔行变色的功能。

11.6　常用自定义方法与插件

虽然在jQuery框架中可以很方便地调用大量的方法和事件，但在实际项目开发过程中，往往许多现有的方法或插件并不能很完整地展示开发的需求和功能，还需要再整合，即使用自定义方式加以改造。这样的改造既可以很完美地实现了功能，又能形成插件文件，方便后续项目的调用，接下来介绍三个常用的自定义方法与插件。

11.6.1　自定义选择器

选择器是jQuery框架中的利器，也是该框架的一个明显特征，获取各类型元素的选择器在jQuery框架中比比皆是，即便如此，一些更详细的选择器功能还需要重新自定义，如获取指定范围内的元素，下面我们通过一个简单的示例来介绍如何自定义一个获取指定范围元素的选择器。

示例11-9　自定义选择器获取指定范围元素

（1）功能描述

在新创建的页面中，使用自定义的选择器获取指定范围内元素，并设置这些元素的字体颜色和背景色。

（2）实现代码

新建一个HTML文件11-9.html，加入如代码清单11-9所示的代码。

代码清单 11-9　自定义选择器

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>自定义选择器</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Plugs/11-9.js"
            type="text/javascript"></script>


    <style type="text/css">
        div
        {
            width: 260px;
            padding-left: 10px;
            font-size: 13px;
            height: 23px;
            line-height: 23px;
            background-color: #eee;
        }
    </style>
    <script type="text/javascript">
        $(function() {
            for (var i = 0; i < 10; i++) {
                $("<div>" + i + "</div>").appendTo("body");
            }
            $("div:between(3-6)").css({ "background": "#555", "color": "#fff" });
        })
        </script>
</head>
<body>
</body>
</html>
```

在代码清单11-9中，当页面加载时，导入了一个名为11-9的JS文件，在该文件中，以自定义插件的方式，扩展了一个名为"between()"的方法，返回指定范围中的元素，该文件实现的代码如下所示：

```
;
(function($) {
    $.extend($.expr[":"], {
        between: function(e, i, bt) {
            var arrSingle = bt[3].split("-");
            return arrSingle[0] <= i && i <= arrSingle[1];
        }
    })
})(jQuery);
```

（3）页面效果

代码执行后的页面效果如图11-9所示。

图　11-9　自定义选择器

（4）代码分析

在本示例的JavaScript代码中，首先通过for语句向页面主体元素＜body＞添加10个＜div＞元素标签；然后，调用自定义选择器插件中的between方法，获取3～6范围内的＜div＞元素；最后，将获取的这些元素通过CSS方法，修改它们的字体和背景颜色。

在自定义的选择器插件代码中，首先，使用伪类选择器"$.expr[":"]"获取页面中的＜div＞元素集合，然后，在新定义的选择器名称"between"中，将这些元素作为形参"e"的实际参数进行遍历。在遍历过程中，形参"i"的实际参数变成遍历时各个元素的索引号，而形参"bt"则为一个过滤条件数组，数组的第3项值为取值范围，即“3～6”，经过分割后，分别获取取值范围的最小值3和最大值6，最后，将元素的索引号与最小值和最大值进行比较，返回在取值范围内的元素。

11.6.2　自定义样式

除了自定义选择器外，在页面开发的过程中，还可以自定义页面的整体风格样式。在通常情况下，为了更好地体现页面层与样式层的分离，页面的样式被写在一个CSS样式文件中，当页面加载时，在＜head＞元素中使用＜link＞标签导入该CSS文件，因此，使用JavaScript代码控制＜link＞标签中导入的文件名称，就可以实现自定义选择页面样式的功能。接下来，我们通过一个简单的示例来加以说明。

示例11-10　自定义主题样式

（1）功能描述

在新创建的页面中，通过选择下拉列表框中不同选择项，动态地改变页面元素的样式。

（2）实现代码

新建一个HTML文件11-10.html，加入如代码清单11-10所示的代码。

代码清单 11-10　自定义主题样式

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>自定义主题样式</title>
    <link rel="stylesheet" href="Css/css01.css" type="text/css"/>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script type="text/javascript">
        $(function() {
            $("#selChangeTheme").change(function() {
                $('link[rel=stylesheet]').attr('href', $(this).val());
            });
        });
    </script>
</head>
<body>
    <div>
        选择主题:
        <select id="selChangeTheme">
            <option value="Css/css01.css">默认主题</option>
            <option value="Css/css02.css">加粗和背景色</option>
        </select>
        <div class="test">自定义主题色</div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图11-10所示。

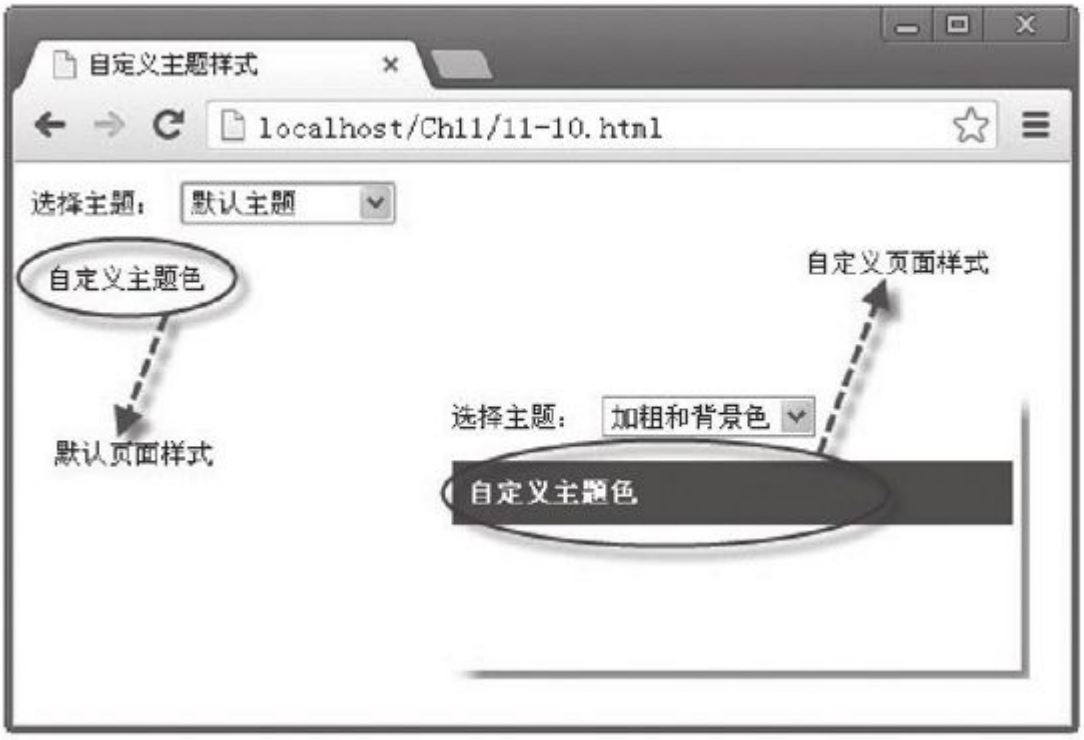


图　11-10　自定义页面样式

（4）代码分析

在本示例中，首先创建了两个CSS样式文件，分别命名为css01和css02，其中css01为默认页面样式，css02为自定义样式。然后，在页面中添加一个下拉列表框元素，并将两个样式文件的路径与列表框中选项的"value"属性值进行绑定。最后，在编写列表框"change"事件时，将获取的选项值，通过attr()方法设置为＜link＞元素的"href"属性值，从而实现页面样式的动态切换效果。

11.6.3　自定义插件

在jQuery中，能否正确使用扩展函数去自定义一些功能性插件，是衡量是否掌握jQuery框架开发的一项重要指标，多使用是掌握的一条重要途径，尤其在一些大型项目中，将一些反复、常用的功能扩展成自定义的插件，不仅有利用代码的后续维护，而且可以使各自功能独立成为模块，减少代码的冗余。

本示例通过自定义插件的方式开发一个以动画的形式向上滑动屏幕的功能。

示例11-11　自定义插件以动画的形式向上滑动屏幕

（1）功能描述

在新创建的页面中，当屏幕的纵向滑动条由顶部向下滑动后或者滑动条不停靠在顶端时，屏幕的底部将出现一个向上滑动的图标，单击该图标，可以将滑动条向上移到至最顶部，同时，底部向上滑动的自动图标隐藏。

（2）实现代码

新建一个HTML文件11-11.html，加入如代码清单11-11所示的代码。

代码清单 11-11　自定义插件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>自定义插件</title>
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Plugs/11-11.js"
            type="text/javascript"></script>
    <style type="text/css">
           body{font-size:13px}
           .artFrame{height:780px}
           .artContent{border:solid 1px #ccc;background-color:#eee;
                    width:260px;padding:8px;word-break:break-all}
           .artList{line-height:1.8em}
    </style>
    <script type="text/javascript">
        $(function() {
            $.fn.scrolltotop({
                controlPos: [6, 6],  // 调整向上图标位置
                scrollduration: 3000 // 设置滑动时速度
            });
        });
    </script>
</head>
<body>
<div class="artFrame">
    <div class="artContent">
          <div class="artList">
              <span>     </span>
              <span>一个优秀的程序开发员，除需掌握专业的开发语言，还要具有</span>
              <span>执着、沉稳、细致的专业素质。</span>
          </div>
     </div>
     </div>
</body>
</html>
```

在代码清单11-11中，当页面加载时，导入了一个名为11-11的JS文件。在该文件中，以自定义插件的方式，扩展了一个名为"scrolltotop()"的方法，实现屏幕纵向滑动条快速返回顶部的功能，该文件实现的代码如下所示：

```
(function($) {
    $.fn.scrolltotop = function(options) {
        var settings = {
            startline: 1,
            scrollto: 0,
            scrollduration: 1000,
            fadeduration: [600, 200],
            isvisible: false,
            shouldvisible: false,
            controlHTML: '<div><img src="Images/totop.png" alt=""/></div>',
            controlPos: [5, 5],


                controlTitle: '点击向上滑'
            };
            var $settings = $.extend({}, settings, options || {});
            var $control = $($settings.controlHTML);
            var $body = $('html,body');
            var scroll = {
                scrollup: function() {
                    var dest = isNaN($settings.scrollto) ? $settings.scrollto :
                    parseInt($settings.scrollto)
                    $body.animate({ scrollTop: dest }, $settings.scrollduration);
                },
                togglecontrol: function() {
                    var scrolltop = jQuery(window).scrollTop()
                    $settings.shouldvisible = (scrolltop >= $settings.startline) ? true : false
                    if ($settings.shouldvisible && !$settings.isvisible) {
                        $control.stop().animate({ opacity: 1 }, $settings.fadeduration[0])
                        $settings.isvisible = true
                    }
                    else if ($settings.shouldvisible == false && $settings.isvisible) {
                        $control.stop().animate({ opacity: 0 }, $settings.fadeduration[1])
                        $settings.isvisible = false
                    }
                },
                scrollinit: function() {
                    $control.css({position: 'fixed', bottom: $settings.controlPos[0],
                    right: $settings.controlPos[1], opacity: 0, cursor: 'pointer' })
                    .attr({ title: $settings.controlTitle })
                    .click(function() { scroll.scrollup(); return false })
                    .appendTo('body')
                }
            };
            $(window).bind("load", function() {
                scroll.scrollinit();
            });
            $(window).bind('scroll resize', function(e) {
                scroll.togglecontrol();
            })
        }
    })(jQuery);
```

（3）页面效果

代码执行后的页面效果如图11-11所示。

图　11-11　自定义插件以动画的形式向上滑动屏幕

（4）代码分析

在本示例的JavaScript中，当页面在加载时，将根据设置的初始值调用自定义插件中的scrolltotop()方法。其中"controlPos"属性用于设置向上图标在页面中的位置，即对应x、y轴的值。"scrollduration"属性用于设置纵向滑动条在向上滑动时的速度，该值越大速度越慢，越小速度越快。

在自定义插件中，除了在页面中使用的两个属性值外，还包括其他属性，都被包含在"settings"对象中，读者可以自行修改或设置。

在插件的"scroll"对象中，包含了3个自定义的函数，分别为"scrollinit"、"scrollup"、"togglecontrol"。其中"scrollinit"函数的功能是初始化向上滑动图标元素，并在元素的"click"事件中绑定"scrollup"函数；"scrollup"函数的功能是将页

面以动画的形式滑动至最顶部，从而实现纵向滑动条自动向上滑动的效果；"togglecontrol"函数的功能是在屏幕滑动或改变大小时，根据当前滑动条所在的纵坐标位置值，决定是否显示或隐藏向上滑动图标。

11.7　本章小结

jQuery初学者在使用jQuery时，由于经验不足，很难避免所编写的代码冗余、考虑不周全的情况。本章精选12个开发技巧与解决方案实例，能够帮助jQuery初学者少走弯路，积累开发经验，不断优化代码，提高开发效率。

第12章　jQuery性能优化

本章内容

jQuery性能优化常用策略

优化选择器执行的速度

使用方法优化性能

优化DOM元素的操作

jQuery库与其他库冲突的解决方案

本章小结

jQuery是继Prototype后的又一个优秀的JavaScript框架，这个简洁、快速、灵活的框架，可以帮我们在页面中实现文档操作、事件处理、Web交互等基本事务。本章将介绍jQuery在日常开发中的一些应用技巧和注意事项，使开发者在编写代码时，少走弯路并提高代码执行的效率。

12.1　jQuery性能优化常用策略

越来越多的Web开发人员在使用jQuery框架强大功能的同时，往往忽视了框架本身的性能优化。jQuery框架虽然为开发人员提供了丰富的页面元素处理功能，但不同的编码调用方式虽然最终实现了目的，而在性能上却存在巨大差别。

本节详细介绍在使用jQuery开发页面的过程中，常用的提升性能的编码方式和处理策略。

12.1.1　优先使用ID与标记选择器

在jQuery中，最快访问DOM元素的方式是通过元素ID号，其次是通过元素的标记。前者源于JavaScript中的document.getElementById()方法，而后者源于document.getElementsByTagName()方法。例如有一个页面，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>测试</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .MyClass{padding:5px;margin:5px}
    </style>
</head>
<body>
    <div id="divTip" class="MyClass" title="tmp">测试文字</div>
</body>
</html>
```

有下列三种方法可以访问页面中的＜div＞元素，代码如下：

```
var eleName0 = $("#divTip");
var eleName1 = $("div");
var eleName2 = $(".MyClass");
```

在使用代码访问元素时，如果有ID号，建议使用ID号直接访问元素，这样的速度是最快的；如果没有ID号，可使用元素标记（tag）访问，其次就是类别（class）进行访问。

在jQuery中，访问页面元素时，尽量避免出现下列的访问语法。

1）虽然ID号访问页面中的元素最快，但避免重复修饰，即使用ID号修饰ID号，其错误代码如下所示：

```
var eleName0 = $("#divTip #divShow");
```

2）避免使用tag或class来修改ID号，这样代码先执行遍历，后进行匹配，其错误代码如下所示：

```
var eleName0 = $(".MyClass #divShow");
```

或：

```
var eleName0 = $("div #divShow");
```

3）如果是通过元素的属性访问，尽量使用tag来修饰进行访问，这样可以加快访问速度，正确代码如下：

```
var eleName3 = $("div[title='tmp']");
```

12.1.2 使用jQuery对象缓存

所谓对象缓存，就是在使用jQuery对象时尽量使用变量先保存对象名，然后通过变量进行相应的方法操作。如下列代码是欠妥的：

```
$("#divTip").bind("click", function() { alert("hello!"); })
$("#divTip").css("width", "200px");
$("#divTip").css("background-color", "red");
```

优化的代码是：

```
var objTmp = $("#divTip;") //先使用变量进行保存
objTmp.bind("click", function() { alert("hello!"); })
objTmp.css("width", "200px");
objTmp.css("background-color", "red");
```

如果想使用定义的变量，在其他函数中也能被使用，那么可以将该变量定义为全局性的变量，其实现的代码如下：

```
window.objPub = { //在全局范围中，定义一个windows对象
        objTmp: $("#divTip")
 }
```

通过全局的变量可以在各个自定义的函数中访问该变量；通过变量实现DOM元素的获取，其实现的代码如下：

```
function TestFun() {
   objPub.objTmp.bind("click", function() { alert("hello!"); })
   objPub.objTmp.css("width", "200px");
   objPub.objTmp.css("background-color", "red");
 }
```

最终实现的功能与定义局部变量一样，但它却可以被不同的自定义函数所调用，也可以当成普通的jQuery对象使用。

在使用变量缓存jQuery对象时，我们在编写代码时有两个地方需要引起注意。

1）无论是局部还是全局性的变量，为了避免与其他变量名相冲突，请尽量使用“$”符号进行命名，代码如下所示：

```
window.$objPub = { //在全局范围中，定义一个windows对象
        $objTmp: $("#divTip")
 }
```

其调用全局变量时，修改后的代码如下：

```
function TestFun() {
   $objPub.$objTmp.bind("click", function() { alert("hello!"); })
   $objPub.$objTmp.css("width", "200px");
   $objPub.$objTmp.css("background-color", "red");
}
```

2）如果在同一个DOM对象中有多个对象的操作，尽量采用链接式的写法优化调用的代码，因此，上述调用全局变量的代码最后可修改成下列代码：

```
function TestFun() {
   $objPub.$objTmp.bind("click", function() { alert("hello!"); })
.css({ "width": "200px", "background-color": "red" });
 }
```

12.1.3　正确使用选择器

选择器是jQuery框架的一项重要的核心功能，正是通过借助丰富、强大的选择器功能，才使快速开发Web页面成为可能，但并非所有的选择器的性能都出色，为了能够代码开发过程中正确选择使用选择器，我们必须了解各类选择器在执行过程中的特点。

在jQuery框架中，从性能的角度可以将选择器大致分为以下三类。

1)）ID和元素标签选择器，如下面代码所示：

```
$("#Id").html("");
$("input").attr("class","red");
    ...
```

上述代码的第一行是通过ID号直接访问页面中的元素，第二行是通过元素标签进行元素的访问这类型的选择器，在执行这类选择器的过程中，jQuery内部将自动调用浏览器的原生方法，且各浏览器都支持这些方法，因此，该类选择器在执行时，速度最快。

2）元素类型选择器，如下面代码所示：

```
$(".red").html("");
...
```

上述代码表示通过类别名称访问元素，jQuery框架在执行该类选择器的过程中，由于各类浏览器对该类选择器的原生方法不尽相同。如在IE系列浏览器中，就无法调用类别的原生方法，速度较慢，而Firefox、Safari、Chrome、Opera浏览器各都可以直接调用getElementByClassName()原生方法，因此速度要快很多。

3）伪类型和属性选择器，如下面代码所示：

```
$(":input").length;
$("div[title='A']").html("");
...
```

上述代码中第一行表示获取所有＜input＞类型的元素总量，属于伪类型选择器，第二行代码表示获取"title"属性值为"A"的元素，属于属性选择器，jQuery框架在执行这两类选择器的过程中，由于浏览器没有对应的原生方法，因此速度最慢。

通过上述对jQuery框架中各类型选择器的性能分析，读者可以根据开发需求自己选择正确的选择器来访问页面中的元素，建议尽量减少使用伪类型及属性选择器去定位页面元素，少量在遍历时使用选择器，从而提升页面执行速度，不断优化代码的性能。

12.1.4　使用最新版本的jQuery

由于jQuery框架在Web项目中的使用十分广泛，其版本的更新速度也相当频繁，一个新版本的发布，与旧版本相比而言不仅增加了许多新功能，而且在性能上也是大大得到了提升。如下列代码所示：

```
$(".red",content).html("");
$(".red").val("");
...
```

上述第一行代码表示在某变量中查找类别名元素，第二行代码表示通过类别选择器获取元素，而这两行普通的代码，在不同jQuery版本中执行的次数是不同的，据测算新版本在1秒内执行的次数是旧版本的10几倍，可以明显看出新版本框架在性能上的优势，因此，建议在实际项目开发过程中，尽量下载使用最新的版本，以期达到执行速度的最优。

12.1.5　避免过度使用jQuery对象

在jQuery中，用户每次使用选择器获取页面中的元素时，都会自动生成一个jQuery对象，该对象包括众多的属性和方法，而通过对象自身去调用这些方法时，资源消耗相对要大很多；为了减少这样的消耗，可以通过对象调用方法对应的函数，如下列代码：

```
//定义保存值的变量
var   strname="";
//定义一个 jQuery 对象
var   $name=$("#name");
//通过调用对象方法获取它的值
strname=$name.text();
//通过调用 jQuery 函数获取它的值
strname=$.text($name);
```

在上述代码中，第二次通过jQuery函数获取对象值所消耗的资源要比第一次使用对象方法获取值要低得多，因为在使用jQuery函数时，不再调用庞大的jQuery对象，执行的速度要快很多。

12.1.6　更多地使用链接式写法

使用链接式（链式）写法是jQuery有别于其他JavaScript框架的特征。使用链接式写法，使执行的每一步结果都进行了自动缓存，相对于不使用链式写法的语句在执行的速度要快不少，如下代码所示：

```
// 第一种链式写法
$("div").show().addClass("red").html(" 链式写法 !");
// 第二种非链式写法
$("div").show();


$("div").addClass("red ");
$("div").html(" 非链式写法 !");
```

从上述代码执行的结果来看，使用链式写法比使用非链式写法在执行速度上要快约20%，因此我们在编写代码时，尽量将代码按链接式进行编写，提升代码的执行速度。

12.1.7　正确处理元素间父子关系

在jQuery中，可以通过多种组合的选择器，从父元素中选取子元素。概括起来，有以下几种组合方式，代码如下所示：

```
// 定义父与子元素对象
var $p = $("#parent");
var $c = $(".child");
// 第 1 种
$p.find($c);
// 第 2 种
$($c, $p);
// 第 3 种
$p.children($c);
// 第 4 种
$($p > $c);
// 第 5 种
$("#parent .child");
// 第 6 种
$(".child", $("#parent"));
```

在上述从父元素取子元素的代码中，执行速度最快的是第1种。第1种方法使用find()方法时，自动调用浏览器固有的原生方法（如getElementById等），因此在这6种方法中执行速度最快；速度最慢的

是第4种和第5种方法，因为在使用这两种方法时，jQuery内部处理顺序是从右到左，这两条语句都是先获取子元素，然后再与多个父元素相匹配，这样的操作会使执行速度大打折扣，远远低于最快的第1种方式。其他几种方法，由于在执行过程中，语句都会转换，因此在性能上损耗不少，其执行的速度都低于第1种，而高于第4种和第5种方式。

12.1.8　正确使用循环语句

在jQuery中，可以通过选择器获取的元素，尽量不要使用循环取得。因为循环操作特别耗时，如果必须使用循环操作，尽量使用原生的for或while语句，而不是each函数。从执行性能上来说，前者远远胜过后者。可以通过下面的代码测试说明，在页面中加入如下的JavaScript代码：

```
// 第一部分
var array = new Array();
for (var i = 0; i < 10000; i++) {
      array[i] = 0;
}
// 第二部分
console.time('for');
for (var i = 0; i < array.length; i++) {
      array[i] = i;
}
console.timeEnd('for');
// 第三部分
console.time('each');
$.each(array, function(i) {
      array[i] = i;
});
console.timeEnd('each');
```

上述代码分为三个部分：

❑第一部分定义一个数组，并使用for语句分别给10000个元素赋值。

❑第二部分使用for语句再次给数组赋值，并通过console.time()和console.timeEnd()方法记录此次赋值所消耗的时间。

❑第三部分使用each语句给数组赋值，也用相同的方法记录此次赋值所消耗的时间。

最后，打开Firefox浏览器下的firebug调试工具，两次赋值所消耗的时间如图12-1所示。

图　12-1　对循环语句的使用次数

从图中可以很清楚地看出，使用for语句所消耗的时间只有"1ms"，即1毫秒，大大低于使用each语句所消耗的"10ms"，其执行

速度前者比后者快了近10倍。因此，在使用循环语句时，尽量使用for类型的原生语句。

12.2　优化选择器执行的速度

由于选择器的使用在jQuery中占据十分重要的位置，优化选择器的执行速度，其实质是优化代码的执行速度，从而加快页面的浏览速度。在jQuery中，为了提升选择器获取DOM元素的速度，笔者建议从下列几方面入手，结合自己的开发实际，提升代码质量。

12.2.1　处理选择器中不规范元素标志

在12.1.1节中，我们介绍了如何优化选择器访问DOM元素的速度。但有时在自动生成的页面中，有很多元素的ID号并不是很规范的标记，常常含有特殊的符号或空格，像这样的DOM元素，如何通过选择器获取，下面将介绍其解决的方法。

在页面中，根据W3C的规定，元素的属性中不能包含类似于“#”、“（”、“[”等不规范的特殊字符，但在自动生成的有些页面的属性中，不可避免地出现了上述特殊符号。针对这样的元素，如果我们采用常规的方法，通过选择器获取元素必然会报错，如下列HTML页面中的元素代码：

```
<div id="div#2#" class=1"MyCls" title="tmp1"></div>
<div id="div[3]" class="MyCls" title="tmp2"></div>
<div id="div(4)" class="MyCls" title="tmp3"></div>
```

采用下列代码方式获取元素将不能获取：

```
$("#div#2#").html("Tmp2");
$("#div[3]").html("Tmp3");
$("#div(4)").html("Tmp4");
```

为了能正确获取这些属性中含有特殊字符的DOM元素，必须在特殊字符前添加转义符“\\”才能获取，添加转义符后的代码如下：

```
$("#div\\#2\\#").html("Tmp2");
$("#div\\[3\\]").html("Tmp3");
$("#div\\(4\\)").html("Tmp4");
```

通过在特殊字符前添加转义符“\\”的方法，可以正确获取这些元素。

在元素的属性中除了含有特殊符号外，有时还含有空格符，空格符在元素的属性中虽然不起眼，但如果我们在编写通过选择器获取元素的代码时，其选择器中含有空格符与不含有空格符将会出现两种完全不同的结果，下面通过一个示例介绍。

示例12-1　选择器中含有空格符与不含空格符的区别

（1）功能描述

分别用选择器获取含有空格符或不含有空格符的DOM对象，并用两个变量保存，然后分别计算两个对象的总个数（即length值），显示在页面中。

（2）实现代码

新建一个HTML文件12-1.html,加入如代码清单12-1所示的代码。

代码清单 12-1　选择器中含有空格符与不含空格符的区别

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>选择器中含有空格符与不含空格符的区别</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .frame{margin:5px;padding:10px;border:solid 1px #666;
          background-color:#eee;width:300px}
          .MyCls{padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            var $objTmp0 = $(".MyCls :hidden");//选择器中含有空格符
            var $objTmp1 = $(".MyCls:hidden"); //选择器中没有空格符
            var strTmp = "'.MyCls :hidden' 方式获取的元素个数是: ";
            strTmp += $objTmp0.length;          //获取含有空格符的总个数
            strTmp += "<br><br>'.MyCls:hidden' 方式获取的元素个数是: ";
            strTmp += $objTmp1.length;          //获取不含空格符的总个数
            $("#divTip").append(strTmp);        //显示在页面中
        })
    </script>
</head>


<body>
    <div class="frame">
        <div id="div0" class="MyCls">
             <div id="div1" class="MyCls" style="display:none"></div>
        </div>
        <div id="div2" class="MyCls" style="display:none"></div>
        <div id="div3" class="MyCls" style="display:none"></div>
        <div id="divTip"></div>
    </div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-2所示。

图　12-2　选择器中含有空格符与不含空格符的区别

（4）代码分析

从图12-2中可以明显看出，在选择器获取元素时，其中的一个空格符虽然并不起眼，但最后获取的结果却截然不同。

在示例12-2中，变量$objTmp0保存的是类别为MyCls中的隐藏元素，因此它的个数为1；而变量$objTmp1保存的是隐藏元素中类别为MyCls的元素，所以它的个数为3。由此可见，通过选择器获取元素时，其中的代码编写应格外注意。

12.2.2　使用子查询优化选择器性能

在jQuery中，如果要查找一个元素，而这个元素被众多别的元素所包含或嵌套在其中，这时，如果直接使用find()方法进行查找，其操作性能比较低。由于jQuery允许在一个集合中附加其他的选择操作，这样我们可以先查找最外层的元素保存起来，又查找更近一层的元素进行保存，最后，在最接近的外层中使用find()方法查找需要的元素，通过这种方式可以在本地变量中保存上一级对象，减少选择器的性能开销。

下面通过一个简单示例来说明该方法实现的过程。

示例12-2　使用子查询优化选择器性能

（1）功能描述

在页面中创建一个＜div＞标记，并在其中创建一个列表＜ul＞，列表中包含两行表项＜li＞，在其中的一个表项中设置＜span＞标记，通过子查询获取＜span＞元素与另外一个表项中的内容，并显示在页面中。

（2）实现代码

新建一个HTML文件12-2.html，加入如代码清单12-2所示的代码。

代码清单 12-2　使用子查询优化选择器性能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用子查询优化选择器性能</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px}
          .ulFrame{padding:0px;margin:0px;list-style-type:none}
          .li0{font-size:11px}
          .li1{font-size:12px}
    </style>
    <script type="text/javascript">
       $(function() {
           var $divF = $("#divFrame");        //保存最外层对象
           var $ulF = $divF.find(".ulFrame"); //在外层对象中查找
           var $li0 = $ulF.find(".li0");
           var $spn = $li0.find("span");      //在最近一层中查找
           var $li1 = $ulF.find(".li1");
           var strTmp = "最终数据: "           //初始化显示内容
           strTmp += "<br>" + $spn.html();    //获取内容
           strTmp += "<br>" + $li1.html();    //获取内容
           $("#divTip").append(strTmp);       //显示在页面中
       })
    </script>
</head>
<body>
    <div id="divFrame">
         <ul class="ulFrame">
             <li class="li0"><span>测试元素一</span></li>
             <li class="li1">测试元素二</li>
         </ul>
    </div>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-3所示。

图　12-3　使用子查询优化选择器性能

（4）代码分析

如果仅仅是为了获取列表中第一个表项中的＜span＞标记内容和第二个表项中的内容，初学者通常会编写下列代码：

```
$spn = $("div ul li span");
$li1 = $("div ul .li1");
```

上述代码同样也可以获取预期的结果，但代码没有缓存，性能较低，对选择器的开销很大，并且不利于查询同级数据，每次都是一个新开销，因此，在嵌套元素中尽量使用子查询访问元素。

12.2.3　给选择器一个上下文

在jQuery中，DOM元素的查找通过$(element)方法实现，除此之外，还可以通过$(expression,[context])方法在指定的范围内查找某个DOM元素。这个方法的优势在于缩小定位元素的范围，比一般的元素定位更快捷，效果更好。其调用的语法格式如下：

```
$(expression, [context])
```

其中，参数expression为需要查找的字符串，可选项[context]为等待查找的DOM元素集、文档或jQuery对象。

下面通过一个示例说明$(expression,[context])方法与一般元素查找方法$(elements)的使用。

示例12-3　在指定的查找范围内获取DOM元素

（1）功能描述

定义两个全局变量，其中$objTmp0通过$(expression,[context])方法获取DOM元素div0，另外一个变量$objTmp0通过$(element)方法获取元素div1。然后自定义一个名为TestFun的函数，该函数的作用是通过定义的全局变量设置DOM元素的内容，并显示在页面中。

（2）实现代码

新建一个HTML文件12-3.html,加入如代码清单12-3所示的代码。

代码清单 12-3　在指定的查找范围内获取 DOM 元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>在指定的查找范围内获取 DOM 元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .MyCls0{border:solid 1px #666;margin:5px;width:200px}
           .MyCls1{border:solid 1px #ccc;margin:5px;width:200px}
           .MyCls{background-color:#eee;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            window.$objPub = { //在全局范围中，定义一个 windows 对象
                $objTmp0: $("#div0", ".MyCls0"),
                $objTmp1: $("#div1")
            }
            TestFun();
        })

        function TestFun() { //自定义显示 div 内容的函数
            $objPub.$objTmp0.html("Tmp0");
            $objPub.$objTmp1.html("Tmp1");
        }
    </script>
</head>
<body>
    <div class="MyCls0">
         <div id="div0" class="MyCls" title="tmp0"></div>
    </div>
    <div class="MyCls1">
         <div id="div1" class="MyCls" title="tmp1"></div>
    </div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-4所示。

图　12-4　获取DOM元素

（4）代码分析

从图中可以看出，虽然两种不同的方法都获取了DOM元素，并设置了其显示的内容，但从执行效果来说，代码$objTmp0:$("#div0",".MyCls0")的执行速度明显优越于代码$objTmp1:$("#div1")，因此尽量使用$(expression,[context])方法访问DOM元素。

12.3　使用方法优化性能

在jQuery框架中，可以借助许多有效的方法来优化代码执行时的性能，如target()方法优化事件中的冒泡现象、data()缓存获取的数据。在代码过程中，通过这些方法的使用，可以极大的提升代码在执行时的效率，接下来逐一进行详细介绍。

12.3.1　使用target()方法优化事件中的冒泡现象

从前面的章节中我们知道，页面元素在互相嵌套时，如果都执行同一个事件，可能会触发事件的冒泡现象。在这现象中会出现一些意想不到的效果，为了规避这种现象的发生，在jQuery中，可以使用stopPropagation()方法来停止事件中的冒泡现象。

在元素绑定事件的过程中，还有一个target()方法，通过该方法可以获取触发事件的元素。如果是多个元素解发同一个事件，可以借助target()方法获取这些元素的父级元素，并将事件绑定父级元素，通过冒泡现象扩展到其子级元素中，这在某种程度上，比将事件绑定每个子级元素，执行效果更加优化。我们通过示例介绍使用事件绑定中的target()方法，优化多元素绑定同一事件时的冒泡现象。

示例12-4　使用事件中的target()方法优化冒泡现象

（1）功能描述

在一个表单form1中包含一个＜fieldset＞标记，在该标记中设置三个文本框（text），为了使用每个文本框在获取焦点（focus）时改变原有的样式，通过事件中的target()方法获取文本框的父级元素，并为该元素绑定一个事件，从而实现了文本框改变样式的需求。

（2）实现代码

新建一个HTML文件12-4.html,加入如代码清单12-4所示的代码。

代码清单 12-4　使用事件中的 target 方法优化冒泡现象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>事件中的 target 方法优化冒泡现象</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{line-height:1.8em}
          .txt{border:#666 1px solid;
          padding:2px;width:80px;margin-right:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#frame").bind("click", function(e) {
                $objChild = $(e.target);// 捕捉触发事件的元素
                $objChild.addClass("txt");// 增加子元素的样式
            })
        });
    </script>
</head>
<body>
    <form id="form1">
        <fieldset id="frame" style="width:200px">
          <legend>输入信息</legend>
          <div>姓名: <input id="Text1" type="text"/></div>
          <div>性别: <input id="Text2" type="text" /></div>
          <div>年龄: <input id="Text3" type="text" /></div>
        </fieldset>
    </form>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-5所示。

图　12-5　事件中的target方法优化冒泡现象

（4）代码分析

在示例12-4的JS代码中，有如下一段代码：

```
$("#frame").bind("click", function(e) {

    $objChild = $(e.target);  // 捕捉触发事件的元素
    $objChild.addClass("txt");// 增加子元素的样式
})
```

其实质是通过e.target获取触发事件的元素，并将事件绑定到父级元素中，通过冒泡现象传递给各个子元素，使其各个子元素也被同样的事件所绑定。因此，上述JS代码与下列代码是等价的，都实现相同的功能。

```
$("#frame input").bind("click", function(e) {
     $(this).addClass("txt");// 增加元素的样式
})
```

但后面的JS代码将遍历全部的input元素，逐个进行事件的绑定。如果有大量的input元素，执行效率明显低于示例12-4的JS代码，但考虑到冒泡现象的出现，如果对页面样式没有严格要求的话，采用这种方法是一种不错的优化速度的选择。

12.3.2　使用data()方法存取普通数据

我们知道，缓存数据无论是在前端页面开发，还是在后台服务器脚本编写，都有十分重要的作用。同样，在jQuery中也可以通过data()方法将数据缓存，虽然使用局部或全局的变量可以保存数据，但变量并不能进行数据的缓存，而且并不依附于某元素自身；如果使用data()方法，可以针对元素定义数据在元素中存取数据，从而避免了数据被循环引用的风险。

根据功能的不同，data()方法有下列几种使用格式：

1）根据元素中的名称、定义或返回存储的数据，其调用格式为：

```
data([name])
```

其中，可选项参数[name]为字符型，表示存储数据的名称。

2）根据元素中的名称，在元素上存储或设置数据，其调用的格式为：

```
data(name, value)
```

其中，参数name表示存储数据的名称，value表示将要被存储的任何数据。

3）除了定义和存储数据外，还可以移除元素中存放的数据，其调用格式为：

```
removeData(name)
```

其中，参数name表示将要被移除的元素上的数据名称。

下面通过一个简单的示例，介绍如何使用data()方法在元素上缓存或移除数据。

示例12-5　使用data()方法在元素上存取普通数据

（1）功能描述

在页面中创建一个＜p＞元素，在该元素上设置或保存数据并移除数据，同时，将各种状态的缓存数据值显示在页面中。

（2）实现代码

新建一个HTML文件12-5.html,加入如代码清单12-5所示的代码。

代码清单 12-5　使用 data() 方法在元素上存取普通数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 data() 方法在元素上存取普通数据</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").data("tmpData"); // 初始时
            // 显示初始化数据
            $("#divTip").append($("p").data("tmpData") == null ?
            " 初始时: null" : $("p").data("tmpData"));
            $("p").data("tmpData", " 周国荣 ") // 设置
            // 显示设置后数据
            $("#divTip").append("<br>赋值后: " + $("p").data("tmpData"));
            $("p").removeData("tmpData") // 移除
            // 显示移除后数据
            $("#divTip").append($("p").data("tmpData") == null ?
            "<br> 移除时: null" : $("p").data("tmpData"));
        })
    </script>
</head>
<body>
    <p><b>数据状态</b></p>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-6所示。

图　12-6　在元素上存取普通数据

（4）代码分析

在本示例的JavaScript代码中，首先通过data()方法初始化一个名为"tmpData"的数据存储变量，由于没有赋值，因此页面显示初始化时，该变量保存的值为"null"；然后，调用data()方法将一个内容为“陶国荣”字符串赋予存储变量，页面显示赋值后，存储变量保存的值为“陶国荣”；最后，通过removeData()方法移除该存储变量，因此，页面显示移除后，存储变量保存的值再次为"null"。

12.3.3　使用data()方法存取JSON数据

除使用data()方法存储一般数据外，还可以存储以“名称/值”格式展示的JSON数据，下面通过一个示例，介绍如何通过data()方法在元素中保存JSON格式的数据。

示例12-6　使用data()方法在元素上存取JSON格式的数据

（1）功能描述

在示例12-5的基础上，将＜p＞元素中使用data()方法存储的数据，更改成JSON格式。同样，将各种状态的缓存数据值显示在页面中。

（2）实现代码

新建一个HTML文件12-6.html,加入如代码清单12-6所示的代码。

代码清单 12-6　使用 data() 方法在元素上存取 JSON 格式的数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 data() 方法在元素上存取 JSON 格式的数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").data("tmpData"); // 初始时
            // 显示初始化数据
            $("#divTip").append($("p").data("tmpData") == null ?
            " 初始时: null" : $("p").data("tmpData"));
            $("p").data("tmpData",
           { name: " 李建州 ", sex: " 女 ", score: "80" }); // 设置
            // 显示设置后数据
            $("#divTip").append("<br> 赋值后: " + $("p").data("tmpData").name + "/" +
            $("p").data("tmpData").sex + "/" + $("p").data("tmpData").score);
            $("p").removeData("tmpData") // 移除
            // 显示移除后数据
            $("#divTip").append($("p").data("tmpData") == null ?
            "<br> 移除时: null" : $("p").data("tmpData"));
        })
    </script>
</head>
<body>
    <p><b> 数据状态 </b></p>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-7所示。

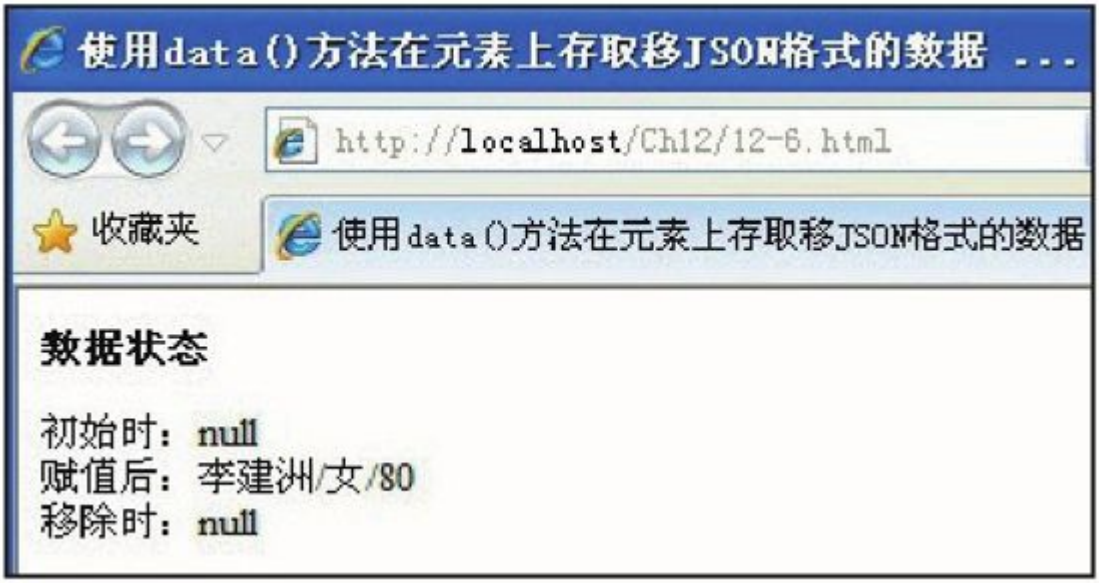


图　12-7　存取JSON格式的数据

（4）代码分析

从示例12-5与示例12-6中我们不难看出，无论是普通数据还是JSON格式的数据，都可以很方便地通过缓存使用data()方法进行存储，其数据的读取或移除也十分方便。

需要说明的是，由于缓存存储的数据是全局性的，可以在各个自定义的函数中进行调用，如编写一个自定义的函数test，调用示例12-6中的缓存数据，其代码如下：

```
function test() {
   var strName = $("p").data("tmpData").name;
   alert(strName);
}
```

执行上述代码后，将获取的数据显示在弹出的对话框中，效果如图12-8所示。

图　12-8　显示获取的数据

说明　虽然使用data()方法可以很方便地存取全局性的数据，但存储的数据随着操作的变化会越来越大，如果不进行及时清理，将会影响原有程序的执行。这一点须引起程序开发人的注意。除此之外，建议在存储针对元素的数据时使用data()方法进行保存，可以优化执行代码。

12.4　优化DOM元素的操作

DOM元素操作的原理是：先在内存中创建DOM结构，然后更新现有的DOM结构，如果直接对DOM进行操作，其性能也是很低的。

12.4.1　减少对DOM元素直接操作

为了减少这种对DOM元素直接操作的次数，有必要在操作前完善大部分的DOM操作，最后通过一次的直接操作，更新其DOM结构。

下面通过一个简单的示例介绍该操作实现的过程。

示例12-7　减少对DOM元素直接操作

（1）功能描述

在页面中创建一个列表＜ul＞标记，通过代码在列表中动态添加6个含有内容的表项，并将最终结果显示在页面中。

（2）实现代码

新建一个HTML文件12-7.html,加入如代码清单12-7所示的代码。

代码清单 12-7　减少对 DOM 元素直接操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>减少对 DOM 元素直接操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           #ulFrame{padding:0px;margin:0px;
           list-style-type:none;width:200px}
           ul li{border-bottom:dashed 1px #ccc;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() { // 定义数组
            var arrList = ["list0", "list1", "list2", "list3", "list4", "list5"];
            var strList = "";// 初始化字符
            $.each(arrList, function(index) { // 遍历后累加数组元素
                strList += "<li>" + arrList[index] + "</li>";
            }) // 一次性完成 DOM 元素的增加
            $("#ulFrame").append(strList);
        })
    </script>
</head>
<body>
    <ul id="ulFrame"></ul>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-9所示。

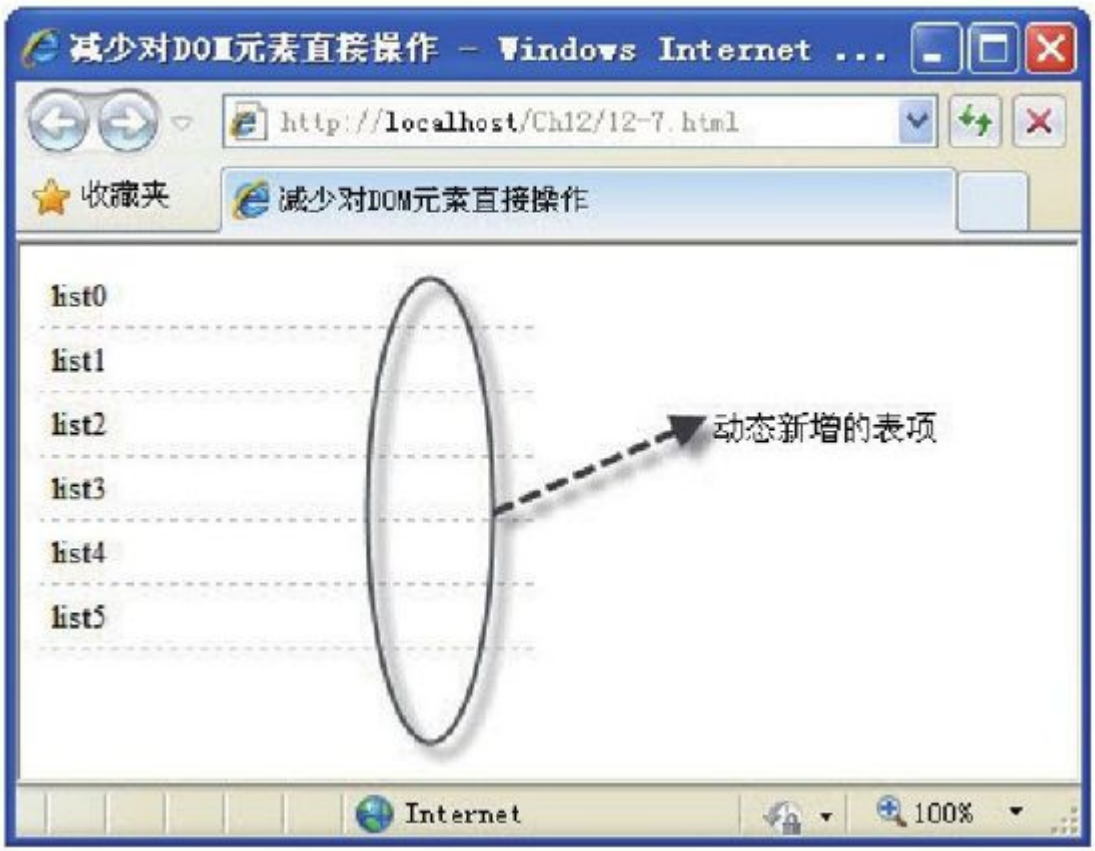


图　12-9　减少对DOM元素直接操作

（4）代码分析

在通过遍历的方式动态增加多个DOM元素时，许多初学者往往采用下列的代码：

```
...省略部分代码
$.each(arrList, function(index) {
    //遍历后累加数组元素
    $("#ulFrame").append("<li>"
    + arrList[index] + "</li>");
})
...省略部分代码
```

以上方法同样可以实现动态增加DOM元素的效果，但每遍历一次，都直接对DOM元素进行操作，效率很低，因此，建议尽量减少直接对DOM元素进行操作，以优化操作性能。

12.4.2　正确区分DOM对象与jQuery对象

由于在jQuery中可以使用JavaScript语法，初学者对DOM与jQuery对象容易混淆，一旦对象理解出错，与对象相关的操作也不会正确。在下面的内容中加以说明，同时阐述两者间的转换关系。

通过传统的JavaScript方法获取的DOM元素对象，就是DOM对象，如下列代码：

```
var objDom = document.getElementById("txtTmp");
var DomValue = objDom.value;
```

其中，变量objDom保存的就是表单中的某个指定文本框对象，即DOM对象；变量DomValue获取了ID号为"txtTmp"的DOM对象的值。

所谓jQuery对象，指的是通过jQuery语法包装原始的DOM对象后生成的新对象。jQuery对象在jQuery库中是特有的，只要是jQuery对象，就可以使用jQuery库中的所有方法与事件。如下列代码：

```
var $obj = $("#txtTmp");
var DomValue = $obj.val();
```

其中，变量$obj保存的就是表单中指定的文本框对象，由于被$()方法所包装，所以该对象变成了jQuery对象；DomValue变量保存通过

jQuery中的val()方法获取的jQuery对象的值。

需要注意的是，不能使用DOM对象调用jQuery对象的方法，下列代码是错误的：

```
var objDom = document.getElementById("txtTmp");
var DomValue = objDom.val();
```

同理，也不能使用jQuery对象调用DOM对象的方法，下列代码也是错误的：

```
var $obj = $("#txtTmp");
var DomValue = $obj.value;
```

如果需要使DOM对象与jQuery对象之间的方法互相调用，必须先实现DOM对象与jQuery对象之间的类型转换。

在jQuery中，可以很方便地完成与DOM对象的转换，只需调用jQuery中提供的[index]与get(index)方法。另外，DOM对象只要通过jQuery方法$()进行包装，就可以转换成jQuery对象。下面通过一个简单的示例演示DOM对象与jQuery对象相互转换的过程。

示例12-8　DOM对象与jQuery对象的类型转换

（1）功能描述

在页面中创建两个＜div＞标记，分别用于保存对象转换后设置的内容，ID号为div0的＜div＞标记保存DOM对象转成jQuery对象后设置的内容；ID号为div1的＜div＞标记保存jQuery对象转成DOM对象后设置的内容，并将设置的内容均显示在页面中。

（2）实现代码

新建一个HTML文件12-8.html，加入如代码清单12-8所示的代码。

代码清单 12-8　DOM 对象与 jQuery 对象的类型转换

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> DOM 对象与 jQuery 对象的类型转换 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            //***** DOM 对象转成 jQuery 对象 *****//
            var objDom0 = document.getElementById("div0");//DOM 对象

            var $obj0 = $(objDom0); // 转成 jQuery 对象
            // 调用 jQuery 中的方法设置其中的内容
            $obj0.html("DOM 对象转成 jQuery 对象后设置的内容 ");
            //***** jQuery 对象转成 DOM 对象 *****//
            var $obj1 = $("#div1"); //jQuery 对象
            var objDom1 = $obj1[0]; // 转成 DOM 对象
            // 调用 JavaScript 中的对象方法设置内容
            objDom1.innerHTML = "jQuery 对象转成 DOM 对象后设置的内容 ";
        })
    </script>
</head>
<body>
    <div id="div0"></div>
    <div id="div1"></div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-10所示。

图　12-10　类型的转换

（4）代码分析

从示例12-8中可以看出，DOM对象与jQuery对象之间的类型转换十分方便，代码也很简单，在该示例的JS代码中，除使用[index]方法将jQuery对象转换成DOM对象外，还可以使用get(index)方法，因此，下面两段代码是等价的：

```
var $obj1 = $("#div1");//jQuery 对象
var objDom1 = $obj1[0];// 转成 DOM 对象
```

等价于：

```
var $obj1 = $("#div1");     //jQuery 对象
var objDom1 = $obj1.get(0);// 转成 DOM 对象
```

通过上述示例的演示，让我们分清了DOM对象与jQuery对象之间的区别，为进一步学习和掌握对象间的方法澄清了概念。

12.5　jQuery库与其他库冲突的解决方案

在通常情况下，由于jQuery库良好的封装性，无论是全局变量（global），还是公用函数，都无一例外地限定在其固有的默认空间中。因此，在一般情况下jQuery库可以与其他的JS库（如Prototype等）并存，不会发生冲突现象。

虽然其他库与jQuery库不会发生冲突，但由于“$”是jQuery自身的快捷符，而其他JS库中也含有“$”符，如果多库共存，就存在哪个库使用“$”符的问题。为了解决这个问题，在jQuery中可以通过函数jQuery.noConflict()，将变量“$”的使用权过渡给需要使用的其他JS库，其调用的语法格式为：

```
jQuery.noConflict()
```

这个函数的作用是变更“$”变量的使用权，以确定jQuery库不与其他库相冲突，使用权变更后，就只能使用jQuery变量访问jQuery对象。

虽然通过函数jQuery.noConflict()可以很好地解决多库共库时，变量符“$”的使用权问题，但在实际的应有中，又分为jQuery在其他

库前导入与其他库后导入两种情况，下面分别进行介绍其解决的方法。

12.5.1　jQuery在其他库前导入

如果jQuery在其他库前就已经导入了，可以直接使用jQuery符号处理相应的jQuery事务，无须调用jQuery.noConflict()函数，下面通过一个示例来演示其实现的效果。

示例12-9　jQuery先于其他库导入“$”的使用权

（1）功能描述

在页面中，jQuery库先于Prototype导入，然后在页面中创建两个按钮。单击第一个按钮时，执行jQuery库中编写的代码；单击第二个按钮时，执行Prototype库中编写的代码。两个不同库编写的代码功能，都是将获取的元素内容显示在页面中。

（2）实现代码

新建一个HTML文件12-9.html,加入如代码清单12-9所示的代码。

代码清单 12-9　jQuery 库先于其他库导入“$”的使用权

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>解决jQuery库先于其他库导入时，变量“$”的使用权</title>
    <!-- 先导入jQuery库 -->
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <!-- 后导入prototype库 -->
    <script type="text/javascript"
            src="Jscript/prototype-1.6.0.3.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        // 直接使用jQuery，无须使用jQuery.noConflict()函数
        jQuery(function() {
            jQuery("#Button1").click(function() {
                // 获取对象
                var $objTmp = jQuery("#txtName");
                // 显示内容
                jQuery("#divTmp").html("J_" + $objTmp.val());
            })
        })
        // 自定义一个测试函数
        function prototype_test() {
            // 使用prototype语法获取元素内容
            $("divTmp").innerHTML = "P_" + $F("txtName");
        }
</script>
</head>
<body>
   <input type="text" id="txtName" value=" 测试内容 "
          style="display:none">
   <div>
       <input id="Button1" type="button" value="jQuery"
              class="btn" />  
       <input id="Button2" type="button" value="prototype"
              onclick="prototype_test();" class="btn" />
   </div>
   <div id="divTmp"></div>
</body>
</html>
```

（3）页面效果

执行后的页面效果如图12-11所示。

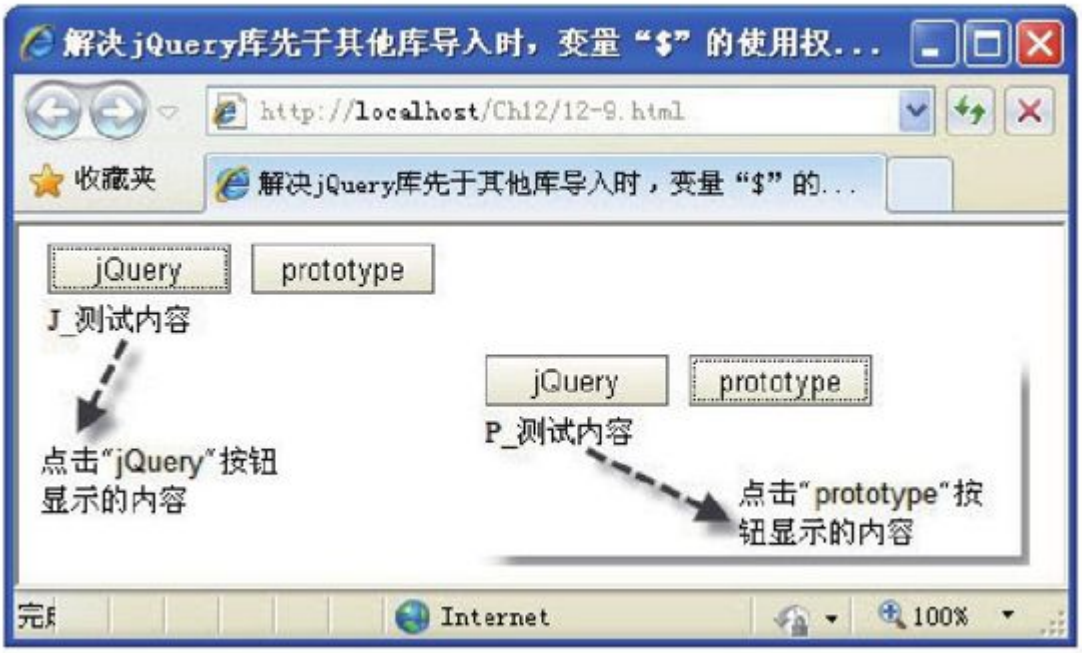


图　12-11　jQuery库先于其他库导入时变量“$”的使用权

（4）代码分析

从图中我们可以看出，两个按钮的单击事件都执行成功，说明在jQuery库先于其他库导入时，无须使用jQuery.noConflict()函数就可以将变量“$”的使用权转让给其他库；jQuery库处理事务时，不再使用“$”符，全部修改成"jQuery"符号即可以执行。

12.5.2　jQuery在其他库后导入

如果jQuery库在其他库后面导入，仍然采用示例12-9中JS代码的写法，将不能正常执行，需要引入jQuery.noConflict()函数，声明转让变量“$”的使用权；这时，在jQuery代码中，如果还想使用变量“$”符号，有两种方法可以实现：第一个方法是自定义一个含其他符号的快捷方式；第二个方法是在jQuery函数的内部继续使用变量“$”。下面通过一个示例来进行演示。

示例12-10　jQuery后于其他库导入“$”的使用权

（1）功能描述

基本功能与示例12-9相同，不同之处在于，jQuery库在其他库后导入。

（2）实现代码

新建一个HTML文件12-10.html,加入如代码清单12-10所示的代码。

代码清单 12-10　jQuery 库后于其他库导入时变量“$”的使用权

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>解决 jQuery 库后于其他库导入时，变量“$”的使用权</title>
    <!-- 先导入 prototype 库 -->
    <script type="text/javascript"
            src="Jscript/prototype-1.6.0.3.js">
```

```
    </script>
    <!-- 后导入 jQuery 库 -->
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        // 使用 jQuery.noConflict() 函数
        //jQuery.noConflict();
        //jQuery(function() {
            //jQuery("#Button1").click(function() {
                // 获取对象
                //var $objTmp = jQuery("#txtName");
                // 显示内容
                //jQuery("#divTmp").html("J_" + $objTmp.val());
            //})
        //})

        //**** 等价于方法一自定义快捷方式 ***//
        // 使用 jQuery.noConflict() 函数
        //var j=jQuery.noConflict();
        //j(function() {
            //j("#Button1").click(function() {
                // 获取对象
                //var $objTmp = j("#txtName");
                // 显示内容
                //j("#divTmp").html("J_" + $objTmp.val());
            //})
        //})

        //**** 等价于方法二在 jQuery 函数内容使用 $ 符 ***//
        // 使用 jQuery.noConflict() 函数
        jQuery.noConflict();
        jQuery(function($) {
            $("#Button1").click(function() {
                // 获取对象
                var $objTmp = $("#txtName");
                // 显示内容
                $("#divTmp").html("J_" + $objTmp.val());
            })
        })

        // 自定义一个测试函数
        function prototype_test() {
            // 使用 prototype 语法获取元素内容
            $("divTmp").innerHTML = "P_" + $F("txtName");
        }
</script>
```

```
</head>
<body>
   <input type="text" id="txtName" value="测试内容" style="display:none">
   <div>
       <input id="Button1" type="button" value="jQuery" class="btn" />  
       <input id="Button2" type="button" value="prototype"
                   onclick="prototype_test();" class="btn" />
   </div>
   <div id="divTmp"></div>
</body>
</html>
```

示例12-10最终实现的页面效果与示例12-9的效果一模一样，通过上述两种方法的介绍，相信在页面中与jQuery库同时并存的问题可以迎刃而解。

12.6　本章小结

作为一种新生的语言框架——jQuery对于程序开发者，尤其是初学者来说，在使用过程中不可避免地会出现语法或操作上的不足。本章首先介绍优化选择器查询元素的功能，列出在编写jQuery代码中各种欠妥或可优化的现象，通过理论与示例结合的方式，进行纠正与优化；同时，对一些技术技巧也进行了总结。读者可以结合自己开发过程的实际情况，针对本章节中提到的问题，合理进行规避，以实现代码质量的最优化。

第13章　jQuery在HTML 5中的应用

本章内容

使用jQuery与HTML 5开发自定义视频播放器

使用jQuery与HTML 5实现图片任意旋转效果

使用jQuery与HTML 5开发拼图游戏

使用jQuery与HTML 5开发星球大战游戏

本章小结

随着互联网的不断发展，新的技术与标准层出不穷，而HTML 5则是其中最为突出的一项，虽然它的最终统一标准尚未制定，但却受到越来越多开发者的关注。因为它不仅仅是一次简单的版本升级，而是代表未来Web应用的发展方向。

目前，各主要浏览器厂商发布的新版本浏览器都支持HTML 5的新特征或属性，而jQuery也在积极应用HTML 5的标准，结合jQuery自身的众多特效和HTML 5新增的元素，通过简单的代码，就可以开发出一个功能强悍的Web应用。

在本章中，我们将通过4个精致、完整的Web应用案例，以项目开发的方式，分别介绍使用jQuery与HTML 5中新增的video、canvas元素开发自定义视频播放器、自定义图片旋转、拼图游戏、星球大战游戏的详细过程。

13.1 使用jQuery与HTML 5开发自定义视频播放器

在HTML 5中，为了减少浏览器对外部插件的依赖，实现客户端应用的本地化，HTML 5新增了许多元素，video元素就是其中之一。该元素用于在浏览器中播放视频格式的文件，使用方法与普通元素一样简单，大大简化了HTML 4之前依靠插件播放视频的步骤。与此同时，还可以通过HTML 5中的API，控制video元素在视频播放过程中的属性，如音量、快进、截屏等，从而真正实现自定义一个视频播放器的效果。

接下来以项目开发的方式对该功能的实现进行逐一介绍。

13.1.1 需求分析

在本项目中，需要实现的需求功能包括以下几个部分：

1）视频文件可以实现播放与暂停功能，在播放状态时，可切换至暂停状态；在暂停状态时，也能切换至播放状态。

2）在视频文件播放过程中，可以实现播放的快进和慢放功能。

3）在视频文件播放过程中，可以调节音量的大小，通过拖拉滑动条的形式实现音量的控制。

4）在视频文件播放过程中，除可以调节音量之外，还可以实现静音的功能，当静音状态时，调节音量的滑动条不可用，开启声音后才能启用。

5）在视频文件播放过程中，提供视频回放的功能，通过该功能可以将视频回放至初始状态。

6）在视频文件播放过程中，提供截屏的功能，通过该功能可以将当前播放的视频画面以图片的形式展示出来。

7）全部的用户操作，都需要有操作提示，根据这些提示，用户可以确定视频播放器当前的状态。

13.1.2　界面效果

当首次打开页面时，视频文件已被加载完成，等待用户单击“播放”按钮，开始进行播放，在播放过程中，“播放”按钮变成“暂停”按钮，同时，单击“快进”、“快退”、“回放”按钮时，分别可以实现相对应的功能，实现的界面效果如图13-1和图13-2所示。

图　13-1　页面初始化状态

图　13-2　视频播放时状态

在视频播放过程中，默认状态下的音量为非静音状态，且音量滑动条在最大值位置，可以通过音量滑动条调节音量，也可以选择静音状态，如果是静音状态，则滑动条不可用，实现的界面效果如图13-3和图13-4所示。

图　13-3　调节音量时状态

图　13-4　静音时状态

当用户在视频播放过程中单击最右侧的“截图”按钮时，可以截取当前正在播放的视频画面并以图片的形式显示在页面中，实现的界面效果如图13-5所示。

图　13-5　视频播放时截图状态

13.1.3　功能实现

针对开发的需求，新建一个HTML文件，命名为Main.html。在该页面文件中，添加用于视频播放的＜video＞元素、用于显示截图的＜canvas＞元素及控制音量的"type"类型值为"range"的＜input＞元素。同时，导入用于控制页面样式的customvide.css文件和实现页面功能的customvide.js文件，完整的页面代码如代码清单13-1所示。

代码清单 13-1　使用 jQuery 与 HTML 5 开发自定义视频播放器

```
<!DOCTYPE html>
<html>
<head>
    <title>使用 jQuery 与 HTML 5 开发视频播放器 </title>
    <link href="Css/customvide.css"
     rel="stylesheet" type="text/css" />
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
</head>
<body>
    <div id="indexvideo">
        <div>
            <video id="video" src="Vide/6-test_1.mov">
                  您的浏览器不支持 video 元素
            </video>
            <div id="tip"></div>
            <canvas id="canvas"></canvas>
        </div>
        <div id="indexnav">
            <ul>
                <li><input id="btnSlow" type="button"
                           class="btn-slow" title=" 慢放 " /></li>
                <li><input id="btnPlay" type="button"
                           class="btn-play" title=" 播放 "/></li>
                <li><input id="btnQuick" type="button"
                           class="btn-quick" title=" 快进 "/></li>
                <li><input id="btnReturn" type="button"
                           class="btn-return" title=" 回放 "/></li>
                <li><input id="btnMax" type="button"
```

```
                    class="btn-volume-max focus"
                    title="音量"/></li>
                <li><input id="rngVoice" type="range"
                    class="btn-volume-rng" min="0"
                    value="1" max="1" step="0.1" /></li>
                <li><input id="btnMin" type="button"
                    class="btn-volume-min"
                    title="静音"/></li>
                <li><input id="btnPic" type="button"
                    class="btn-pic" title="截图"/></li>
            </ul>
        </div>
    </div>
    <script src="Jscript/customvide.js"
            type="text/javascript"></script>
</body>
</html>
```

导入页面的CSS文件customvide.css用于控制页面元素显示时的布局，同时设置各功能元素，如播放、暂停等元素的背景图片，完整的代码如下所示：

```
body
{
    text-align:center;font-size:13px
}
ul
{
    list-style-type:none;padding:0px;margin:0px
}
ul li
{
    float:left
}
video
{
    width:494px;padding:3px 3px 0px 3px
}
canvas
{
    width:245px;
    height:185px;
    position:absolute;
    top:10px;
    left:10px;
}
#tip
{
    position:absolute;
    top:25px;
    padding-left:20px;
    color:#fff
}
#indexvideo
{
    margin:0 auto;
```

```
    width:500px;
    background-color:#eaeaea;
    border:solid 1px #666
}
#indexnav .focus
{
    border:solid 1px #666
}
#indexnav
{
    height:35px;line-height:35px
}
#indexnav input
{
    border:none;
    cursor:pointer;
    width:16px;
    height:16px
}
#indexnav .btn-slow
{
    width:50px;
    background: url(icons/playback_slow.png) no-repeat center center;
}
#indexnav .btn-play
{
    width:50px;
    background: url(icons/playback_play.png) no-repeat center center;
}
#indexnav .btn-pause
{
    width:50px;
    background: url(icons/playback_pause1.png) no-repeat center center;
}
#indexnav .btn-quick
{
    width:50px;
    background: url(icons/playback_quick.png) no-repeat center center;
}
#indexnav .btn-return
{
    width:50px;
    background: url(icons/playback_return.png) no-repeat center center;
}
#indexnav .btn-volume-max
{
    margin-left:50px;width:22px;height:22px;
    background: url(icons/sound_max.png) no-repeat center center;
}
#indexnav .btn-volume-rng
{
    margin-top:9px;width:100px
}
#indexnav .btn-volume-min
{
```

```
    margin-right:50px;
    width:22px;
    height:22px;
    background: url(icons/sound_min.png) no-repeat center center;
}
#indexnav .btn-pic
{
    width:22px;
    background: url(icons/playback_pic.png) no-repeat center center;
}
```

在Main.html文件中，除导入CSS文件外，还添加了一个用于实现各功能的rotateimage.js文件。在该文件中，先通过遍历获取各个功能按钮，然后编写它们在被点击时需要实现的各项功能，完整代码如下所示：

```
var intSpeed = 1; //播放速率
var intInterval, intTipInterval;
var $video = document.getElementById("video");
var int_plybckrate = $video.playbackRate; //播放速率
var $canvas = document.getElementById("canvas");
var $ctx = $canvas.getContext('2d');
$(function() {
    $("#indexnav input[type='button']").each(function(i) {
        $(this).bind("click", function() {
            switch (i) {
                case 0:
                    if (!$video.paused) {
                        int_plybckrate -= 0.1
                        if (int_plybckrate < 0)
                            int_plybckrate = 0;
                        intSpeed = Math.round(int_plybckrate * 100) / 100;
                    }
                    $("#tip").html("慢放速率: " + intSpeed * 100 + "%");
                    break;
                case 1:
                    if ($video.paused) {
                        $("#tip").html("正在播放");
                        //播放视频
                        $video.play();
                        $video.playbackRate = intSpeed;
                        $(this).attr("class", "btn-pause");
                    } else {
                        $("#tip").html("已经暂停");
                        //暂停播放
                        $video.pause();
                        $(this).attr("class", "btn-play");
                    }
                    break;
                case 2:
                    if (!$video.paused) {
                        int_plybckrate += 0.1
                        intSpeed = Math.round(
                        int_plybckrate * 100) / 100;
                    }
                    $("#tip").html("快进速率: " + intSpeed * 100 + "%");
```

```
                    break;
                case 3:
                    $("#tip").html("点击回放");
                    if (!$video.paused) {
                        intInterval = setInterval(function() {
                            if ($video.currentTime == 0) {
                                clearInterval(intInterval);
                            }
                            else
                                $video.currentTime -= 1;
                        }, 200);
                    }
                    break;
                case 4:
                    $("#tip").html("取消静音");
                    $video.muted = false;
                    $("#btnMin").removeClass("focus");
                    $("#rngVoice").attr("disabled", "");
                    $(this).addClass("focus");
                    break;
                case 5:
                    $("#tip").html("静音生效");
                    $video.muted = true;
                    $("#btnMax").removeClass("focus");
                    $("#rngVoice").attr("disabled", "true");
                    $(this).addClass("focus");
                    break;
                case 6:
                    $("#tip").html("正在截图");
                    if (!$video.paused) {
                        $canvas.width = $video.videoWidth;
                        $canvas.height = $video.videoHeight;
                        $ctx.drawImage($video, 0, 0,
                        $canvas.width, $canvas.height);
                    }
                    break;
            }
        });
    });
    $("#rngVoice").bind("change", function() {
        $video.volume = $(this).val();
        $("#tip").html("音量: " + $(this).val() * 100 + "%");
    });
    $("#video").bind("timeupdate", function() {
        $video.playbackRate = intSpeed
    }).bind("ended", function() {
        $("#tip").html("播放结束");
        $("#btnPlay").attr("class", "btn-play");
        intSpeed = 1;
        int_plybckrate = intSpeed;
        $video.currentTime = 0;
    });
    intTipInterval = setInterval(function() {
        $("#tip").html("");
    }, 5000);
});
```

13.1.4　代码分析

在本项目的JavaScript文件rotateimage.js中，分别编写在点击各功能按钮时触发的事件，为了获取各个按钮，首先，使用如下代码进行页面元素的遍历：

```
$("#indexnav input[type='button']").each(function(i) {
            ...
})
```

在遍历过程中，$(this)为按钮对象，参数i表示按钮的索引号，在遍历按钮的同时绑定它的单击事件，通过索引号i区分用户点击了哪个按钮，如i为0时，表示点击的是“慢放”按钮，以此类推，代码格式如下所示：

```
$(this).bind("click", function() {
            switch (i)
{
              ...
            }
});
```

根据索引i确定被点击的按钮之后，开始编写相应功能的代码，当按钮索引号为0、2时，分别对应视频文件的“慢放”和“快进”功能，为了实现这两个功能，需要动态修改视频元素的"playbackRate"速率属性值，点击一次“慢放”按钮时，该值相应

减少0.1；点击一次“快进”按钮时，该值相应增加0.1，代码片断如下。

“慢放”实现代码：

```
if (!$video.paused) {
      int_plybckrate -= 0.1
      if (int_plybckrate < 0)
          int_plybckrate = 0;
          intSpeed = Math.round(int_plybckrate * 100) / 100;
}
$("#tip").html("慢放速率: " + intSpeed * 100 + "%");
```

在上述代码中，"!$video.paused"表示视频文件正在播放时，变量"int_plybckrate"保存了视频元素的"playbackRate"属性值，该值减少0.1，在减少的过程中当该值小于0时，变量"int_plybckrate"值为0，避免出现负数值，同时转换成百分比的小数形式，通过ID号为"tip"的页面元素，将操作的信息显示在页面中。

“快进”实现代码：

```
if (!$video.paused) {
      int_plybckrate += 0.1
      intSpeed = Math.round(int_plybckrate * 100) / 100;
}
$("#tip").html("快进速率: " + intSpeed * 100 + "%");
```

无论是在“慢放”或“快进”功能中，只是修改了视频元素的"playbackRate"属性值，而要在正在播放的视频过程中反映出修改速率后的效果，需要绑定视频文件的"timeupdate"事件。在该事件

中，重新将修改后的变量值赋予视频元素的"playbackRate"属性由于"timeupdate"事件表示播放过程事件，即当播放的进度发生变化时，就会触发该事件，赋值完成后，将立刻显示“慢放”或“快进”的播放效果，"timeupdate"事件代码如下：

```
$("#video").bind("timeupdate", function() {
        $video.playbackRate = intSpeed
})
```

当按钮索引号为1时，表示播放视频文件，为了实现“播放”与“暂停”功能的相互切换，在进行播放时，先判断当前视频文件的状态。如果是已经播放，则转成暂停状态，否则转成播放状态，同时修改对应的元素样式。实现代码如下：

```
if ($video.paused) {
    $("#tip").html(" 正在播放 ");
    // 播放视频
    $video.play();
    $video.playbackRate = intSpeed;
    $(this).attr("class", "btn-pause");
} else {
    $("#tip").html(" 已经暂停 ");
    // 暂停播放
    $video.pause();
    $(this).attr("class", "btn-play");
}
```

当按钮索引号为3时，表示回放视频文件，其实现的方法是通过"setInterval"方法在指定的时间内，减少当前视频元素的"currentTime"属性值，直至0为止，从而实现视频回放效果，代码如下：

```
if (!$video.paused) {
     intInterval = setInterval(function() {
     if ($video.currentTime == 0) {
          clearInterval(intInterval);
     }
     else
          $video.currentTime -= 1;
     }, 200);
 }
```

当按钮索引号为4和5时，表示开启或禁用音量功能，实现的方法是通过修改视频元素的"muted"属性值，该值为"true"表示开启，该值为"false"表示禁用，其实现的代码见源文件，在此不重述。另外，为了实现当用户拖动滑动条控制音量的功能，需要绑定滑动条的"change"事件，在该事件中，将滑动条的值重新赋予视频元素的"volume"属性，从而实现动态控制音量的功能，其实现代码如下：

```
$("#rngVoice").bind("change", function() {
    $video.volume = $(this).val();
    $("#tip").html("音量: " + $(this).val() * 100 + "%");
});
```

当按钮索引号为6时，表示截取视频文件正在播放时的画面，实现的功能是，首先获取视频文件的宽度与高度，然后，通过调用canvas元素的drawImage()方法在画布中绘制指定宽度与高度的视频播放文件画面，其实现的代码如下所示：

```
if (!$video.paused) {
      $canvas.width = $video.videoWidth;
      $canvas.height = $video.videoHeight;
      $ctx.drawImage($video, 0, 0, $canvas.width, $canvas.height);
}
```

13.2　使用jQuery与HTML 5实现图片任意旋转效果

在HTML 4和之前的版本中，需要编写大量复杂的JavaScript代码才能实现对图片的旋转功能。而在HTML 5中，在页面中创建新增的＜canvas＞元素，通过导入jQuery框架调用该元素加载图片的方法，就可以很简单地实现图片的旋转效果。

接下来以项目开发的方式对该功能的实现进行逐一介绍。

13.2.1　需求分析

在本项目中，需要实现的功能包括以下两个部分：

1）可以快捷实现对目标图片的左右90度的自由旋转。

2）实现对自定义任意角度的图片旋转。

13.2.2　界面效果

当用户在页面中点击“向右旋转90”或“向左旋转90”链接时，当前图片可以实现按指定方向与角度的旋转效果，其实现的界面效果如图13-6、图13-7、图13-8所示。

图　13-6　旋转图片初始化效果

图　13-7　图片向右旋转90度效果

图　13-8　图片向左旋转90度效果

当用户在页面中单击“自定义旋转”链接时，将显示一个用于输入角度的文本框，当用户在文本框中输入完角度值后，页面中的图片将按该输入的角度值进行旋转，实现的效果如图13-9所示。

图　13-9　图片按自定义角度旋转效果

13.2.3　功能实现

根据开发需求，新建一个HTML文件RotateImg.html，在该文件中，添加一个用于显示旋转效果的图片＜img＞元素，并添加一个＜ul＞元素，通过该元素中的＜li＞元素展现各项旋转链接。同时，添加一个文本输入框元素，用于输入自定义的旋转角度。此外，再导入控制页面样式的CSS文件rotateimage.css和实现图片旋转功能的JS文件rotateimg.js，完整的页面代码如代码清单13-2所示。

代码清单 13-2　使用 jQuery 与 HTML 5 实现图片任意旋转效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 与 HTML 5 实现图片任意旋转效果</title>
    <link href="Css/rotateimage.css"
          rel="stylesheet" type="text/css" />
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Jscript/rotateimgplugin.js"
            type="text/javascript"></script>
</head>
<body>
    <div id="indexrotateimg">
        <div id="contentrotateimg">
            <img src="Images/2012.jpg" alt="" id="bookimg"/>
        </div>
        <ul>
            <li>向右旋转 90</li>
            <li>向左旋转 90</li>
            <li>自定义旋转</li>
            <li id="liangle">
              <input type="text" id="txtangle" class="txt"/>
            </li>
        </ul>

    </div>
    <script src="Jscript/rotateimg.js"
            type="text/javascript"></script>
</body>
</html>
```

在上述的HTML文件中，先导入了一个用于控制页面布局和样式的CSS文件rotateimage.css，该文件的完整代码如下所示：

```
body
{
    text-align:center;
    font-size:13px
}
.txt
{
    border:#666 1px solid;width:70px
}
ul
{
    list-style-type:none;
    padding:0px;
    margin:0px;
    float: right;
    width:80px;
}
ul li
{
  text-align:left;
  padding:5px;
  border-bottom:solid 1px #ccc;
  background-color:#eee;
  height:18px;
  line-height:18px;
  cursor:pointer
}
#indexrotateimg
{
    margin:0 auto;
    width:382px;
}
#contentrotateimg
{
    background-color:#eaeaea;
    border:solid 1px #666;
    width:280px;
    float:left
}
#liangle
{
    padding-bottom:10px;
    display:none;
}
```

在本项目的HTML页面中，除导入CSS文件之外，还分别导入了两个JavaScript文件rotateimgplugin.js和rotateimg.js。前者是一个jQuery插件，用于实现图片任意角度的旋转功能，而在rotateimg.js文件中调用插件中的方法，实现页面中各项图片的旋转功能。rotateimg.js文件的完整代码如下所示：

```
$(function() {
    $("#indexrotateimg ul li").each(function(i) {
        $(this).bind("click", function() {
            switch (i) {
                case 0:
                    $("#bookimg").rotate(90);
                    break;
                case 1:
                    $("#bookimg").rotate(-90);
                    break;
                case 2:
                    $("#liangle").toggle();
                    break;
            }
        });
    });
    $("#txtangle").bind("change", function() {
        $("#bookimg").rotate($(this).val());
    });
});
```

在rotateimg.js文件中调用的元素方法rotate()来源于jQuery插件文件rotateimgplugin.js。在该文件中，通过接收用户传入的旋转角度值，随后在页面中动态创建一个canvas元素，并将页面中的图片按接收的旋转角度值，加载至canvas元素中，该文件的完整代码如下所示：

```
jQuery.fn.rotate = function(angle, whence) {
    var $a = this.get(0);
    if (!whence) {
        $a.angle = (($a.angle == undefined ? 0 : $a.angle) + angle) % 360;
    } else {
        $a.angle = angle;
    }
    if ($a.angle >= 0) {
        var rotation = Math.PI * $a.angle / 180;
    } else {
        var rotation = Math.PI * (360 + $a.angle) / 180;
    }
    var $cos = Math.round(Math.cos(rotation) * 1000) / 1000;
    var $sin = Math.round(Math.sin(rotation) * 1000) / 1000;
    var $cnv = document.createElement('canvas');
    if (!$a.oImage) {
        $cnv.oImage = new Image();
        $cnv.oImage.src = $a.src;
    } else {
        $cnv.oImage = $a.oImage;
    }
    $cnv.width = Math.abs($cos * $cnv.oImage.width) +
                 Math.abs($sin * $cnv.oImage.height);


        $cnv.height = Math.abs($cos * $cnv.oImage.height) +
                      Math.abs($sin * $cnv.oImage.width);
        var $cxt = $cnv.getContext('2d');
        $cxt.save();
        if (rotation <= Math.PI / 2) {
            $cxt.translate($sin * $cnv.oImage.height, 0);
        } else if (rotation <= Math.PI) {
            $cxt.translate($cnv.width, -$cos * $cnv.oImage.height);
        } else if (rotation <= 1.5 * Math.PI) {
            $cxt.translate(-$cos * $cnv.oImage.width, $cnv.height);
        } else {
            $cxt.translate(0, -$sin * $cnv.oImage.width);
        }
        $cxt.rotate(rotation);
        $cxt.drawImage($cnv.oImage, 0, 0, $cnv.oImage.width,$cnv.oImage.height);
        $cxt.restore();
        $cnv.id = $a.id;
        $cnv.angle = $a.angle;
        $a.parentNode.replaceChild($cnv, $a);
    }
```

13.2.4　代码分析

在本项目的rotateimg.js文件中，为了获取用户在页面中对图片的旋转角度，首先对＜ul＞中的＜li＞元素进行遍历。在遍历时，根据元素的索引号i值判断用户所选择的旋转角度，其实现的代码片断如下所示：

```
$("#indexrotateimg ul li").each(function(i) {
        $(this).bind("click", function() {
            switch (i) {
                case 0:
                    $("#bookimg").rotate(90);
                    break;
                case 1:
                    $("#bookimg").rotate(-90);
                    break;
                case 2:
                    $("#liangle").toggle();
                    break;
            }
        });
    });
```

在上述代码中，当索引号i值为2时，表示用户选择了自定义角度，此时，通过调用toggle()方法，切换时显示输入角度的文本框。当用户在文本框中输入角度值时，触发文本框的"change"事件，在该事件中编写如下代码：

```
$("#txtangle").bind("change", function() {
        $("#bookimg").rotate($(this).val());
});
```

上述代码表示，在文本框的"change"事件中，将文本框的值作为调用旋转插件中的rotate()方法的实参，实现自定义旋转角度的功能。

在介绍本项目中的jQuery旋转插件文件rotateimgplugin.js之前，先了解HTML 5中＜canvas＞元素的两个方法，translate()和rotate()，接下来将分别进行介绍。

translate()方法调用格式如下：

```
cxt.translate(x,y)
```

该方法的功能是实现对已绘制图形的移动操作。其中cxt为上下文环境对象，参数x表示将图形原点横坐标移动的距离，大于0时向右移动，小于0时向左移动；参数y表示将图形的原点纵坐标移动的距离，大于0时向下移动，小于0时向上移动。

rotate()方法调用格式如下：

```
cxt.rotate(angle)
```

该方法的功能是实现对已绘制图形的旋转操作。其中cxt为上下文环境对象，参数angle表示图形旋转的角度，大于0时顺时针旋转，小于0时逆时针旋转。

在jQuery旋转插件文件rotateimgplugin.js中，首先对传入的旋转角度进行检测和转换，并通过该角度值获取对应的sin和cos角度的值，将这两个值分别保到在变量$cos和$sin中。然后，创建一个＜canvas＞元素，并将页面中的图片绘制在该元素中，接下来根据传入角度值的不同，调用translate()方法对加载的图片进行移动，并通过rotate()方法对图片进行旋转。最后，将移动、旋转后的图片通过drawImage()方法绘制在＜canvas＞元素中，并调用restore()重新保存，同时显示在页面中，从而实现图片按钮角度旋转的效果，完整代码如rotateimgplugin.js文件所示。

13.3　使用jQuery与HTML 5开发拼图游戏

在HTML 5中，新增加的＜canvas＞标签是一个十分重要的页面元素，它不仅可以实现图片的旋转功能，还可以按指定的数量，对图片进行分割式绘制，形成拼图前效果。当用户点击某个分割后的图片小块时，再改变其在画布元素＜canvas＞中所在的位置，实现移动效果。当所有的位置都与分割前相符合时，即完成拼图功能。

接下来以项目开发的方式对该功能的实现进行逐一介绍。

13.3.1　需求分析

在本项目中，需要实现的功能包括以下三个部分：

1）整个拼图游戏有3个关卡，分别为“容易”、“较难”、“最难”，用户可以自由选择，各关卡分别显示不同的图片效果。

2）点击拼图中的某一个小图块后，当再次点击一次目标小图块时，前后两次被点击的小图块将实现位置互换功能。

3）当用户将全部分割后的小图块组合成分割前的图片效果时，拼图游戏结束，所有分割后的小图块不能再点击，并显示拼图成功信息。

13.3.2　界面效果

当拼图游戏首次加载时，默认选择“容易”关卡，背景图片为第一张，“容易”关卡将1张图片分割成9张小图块，“较难”关卡将1张图片分割成16张小图块，“最难”关卡将1张图片分割成25张小图块，用户可以自由选择，其实现的界面效果如图13-10、图13-11、图13-12所示。

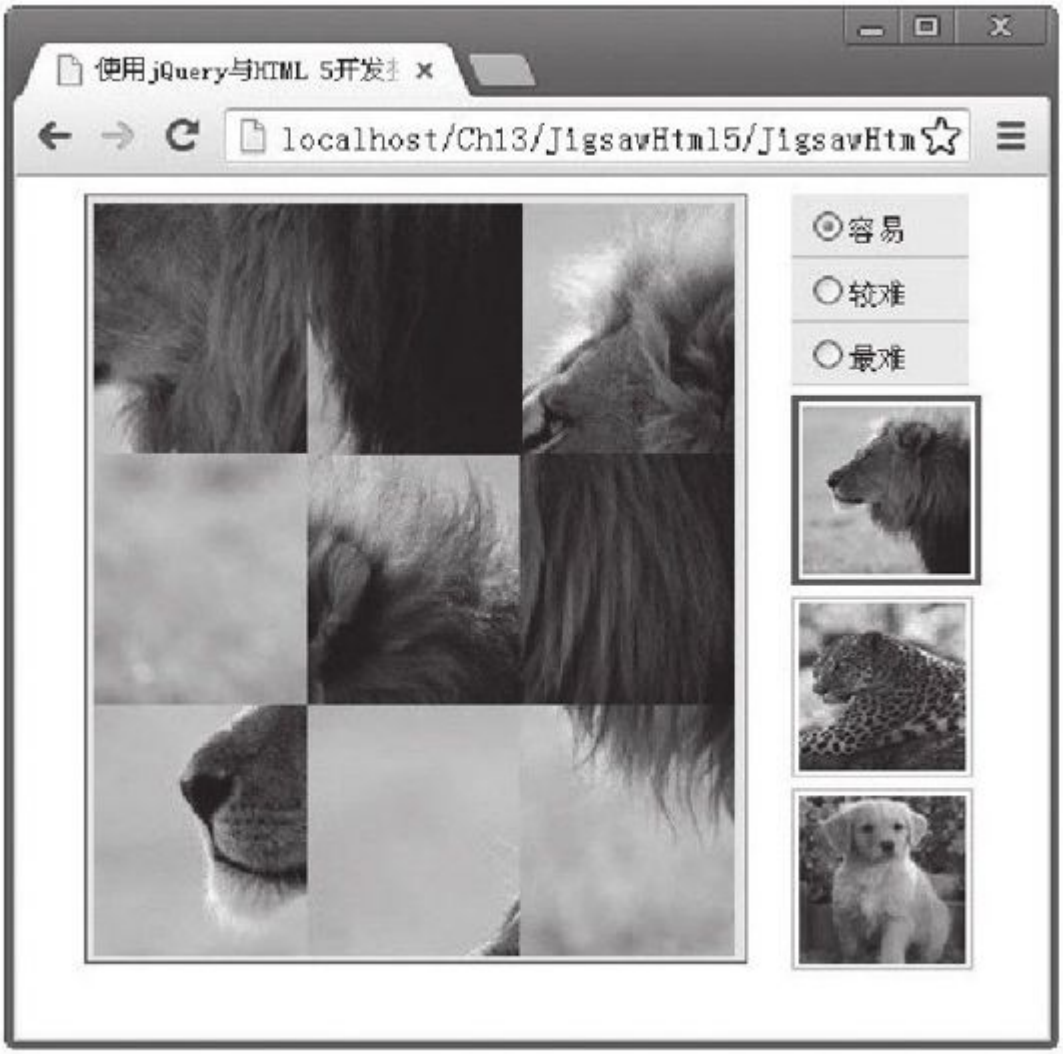


图　13-10　“容易”关卡页面显示的效果

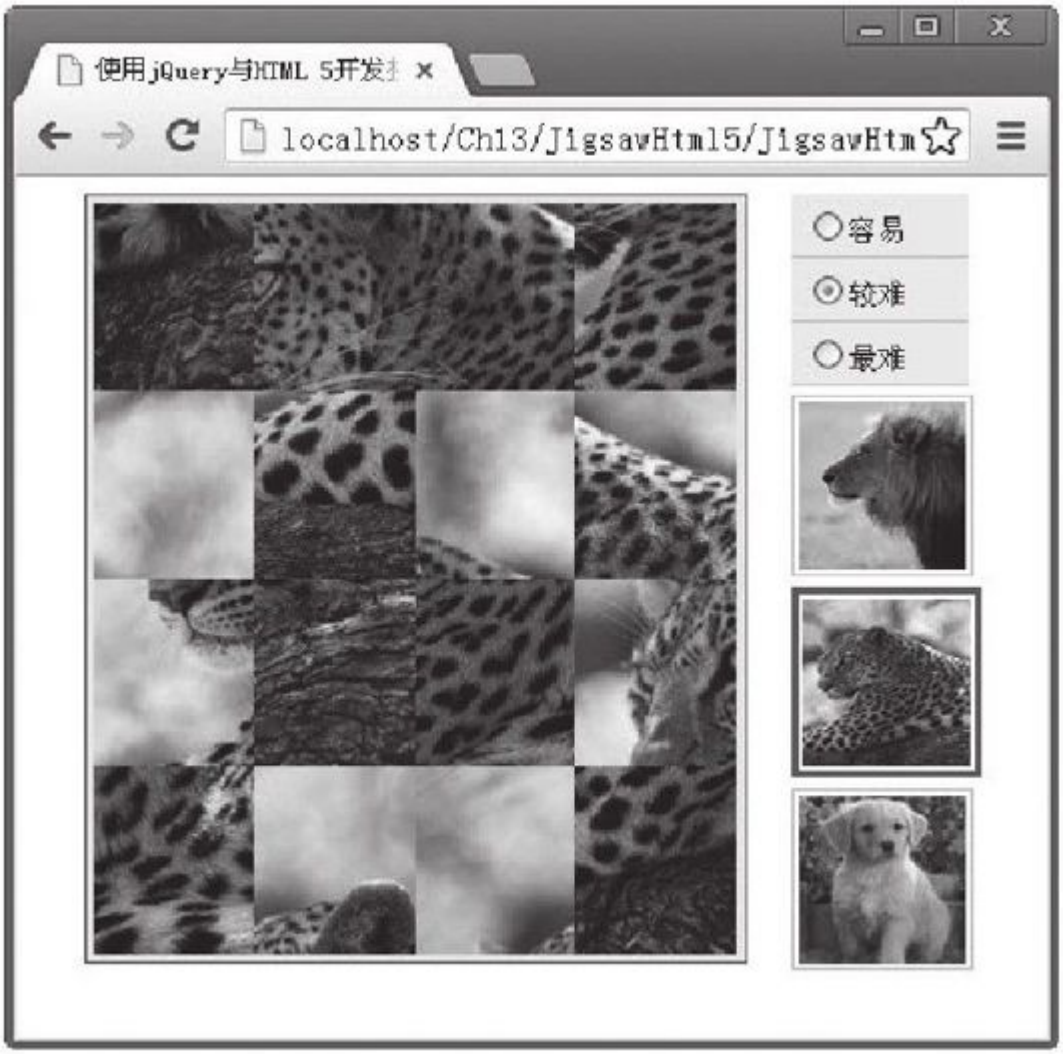


图　13-11　“较难”关卡页面显示的效果

图　13-12　“最难”关卡页面显示的效果

用户点击某一个小图块后，再点击另一小图块时，两个小图块的位置将互换，其实现的页面效果如图13-13所示。

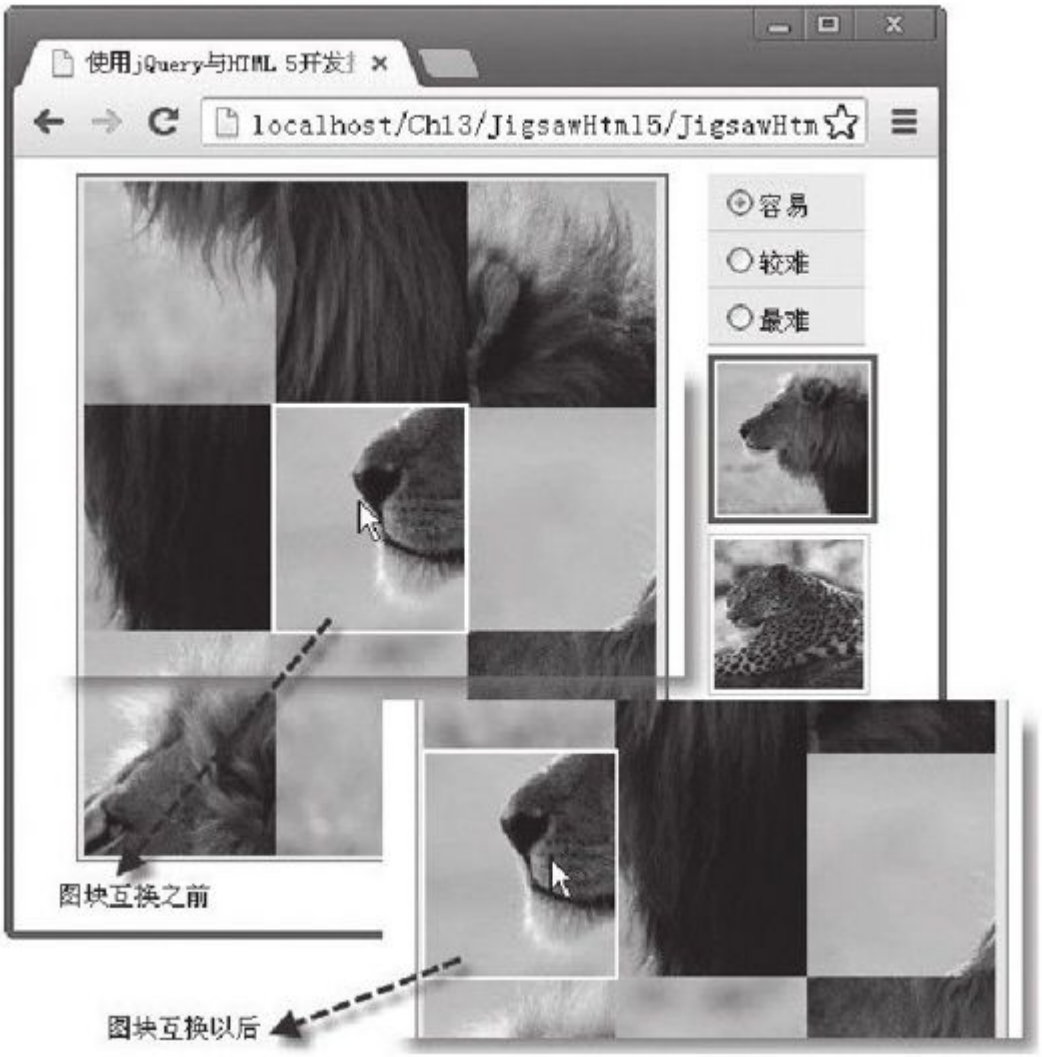


图　13-13　小图块互换前后对比效果

分割后小图块通过点击互换位置的方法，最后拼合成一张与分割前一样的完整图片，此时将显示拼图成功的提示信息，并且所有小图块都不能再点击，实现的界面效果如图13-14所示。

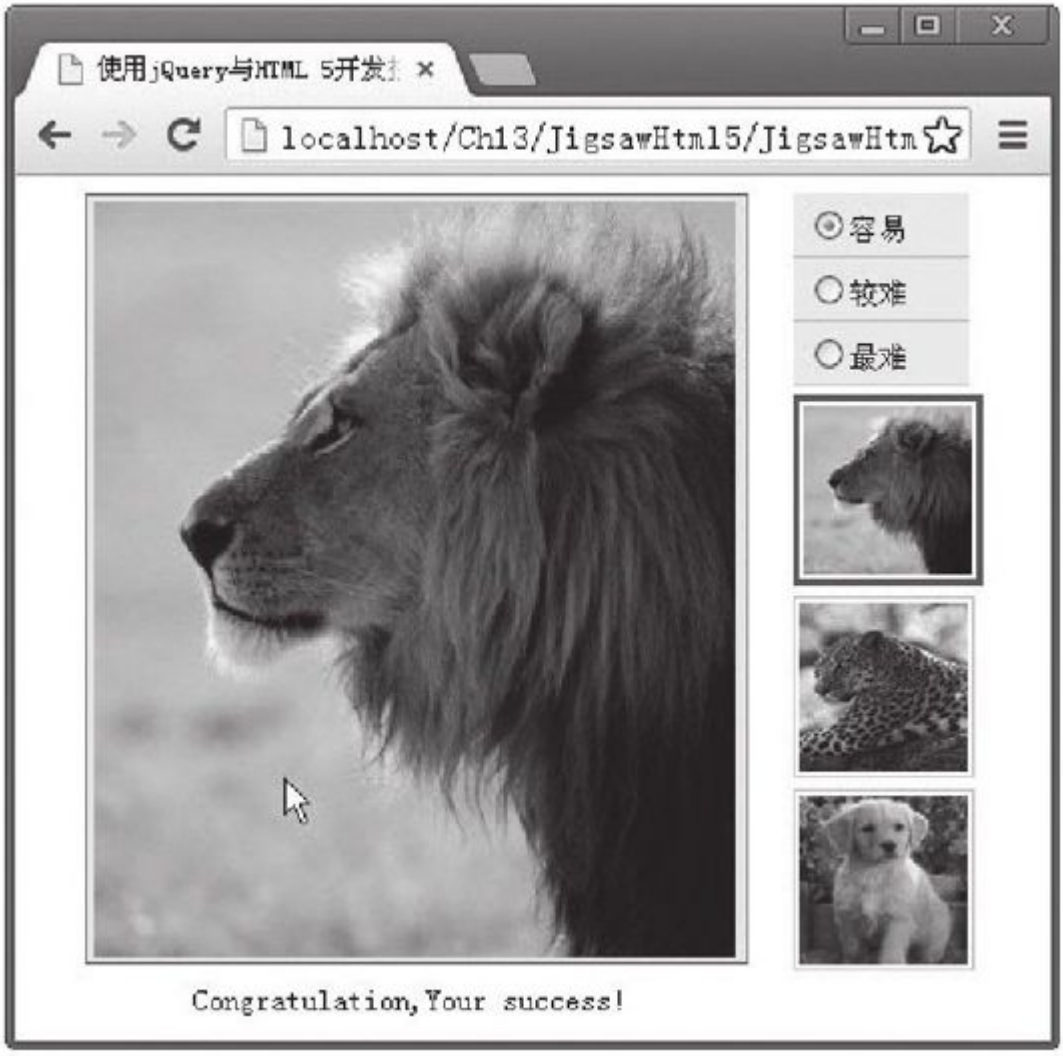


图　13-14　拼图完成后的效果

13.3.3　功能实现

根据拼图开发需求，新创建一个HTML文件JigsawHtml5.html，在该文件中，通过"type"属性值为"radio"的＜input＞元素显示游戏所设置的关卡，创建＜ul＞元素显示拼图所使用的图片。另外，在页面中导入控制页面样式的CSS文件jigsaw.css和实现拼图功能的JS文件jigsaw.js，该页面的完整代码如代码清单13-3所示。

代码清单 13-3　使用 jQuery 与 HTML 5 开发拼图游戏

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 与 HTML 5 开发拼图游戏</title>
    <link href="Css/jigsaw.css" rel="stylesheet"
          type="text/css" />
    <script src="Jscript/jquery-1.8.2.min.js"
            type="text/javascript"></script>
    <script src="Jscript/jigsawplugin.js"
            type="text/javascript"></script>
</head>
<body>
    <div id="indexjigsaw">
        <ul>
            <li><input id="Radio1" type="radio"
                       name="type" checked/>容易</li>
            <li><input id="Radio2" type="radio"
                       name="type"/>较难</li>
            <li><input id="Radio3" type="radio"
                       name="type"/>最难</li>
```

```
        </ul>
        <ul class="ulimg" >
            <li class="focus"><img src="Images/pic01.jpg" class="img"/></li>
            <li class="defalut"><img src="Images/pic02.jpg" class="img"/></li>
            <li class="defalut"><img src="Images/pic03.jpg" class="img"/></li>
        </ul>
        <div id="gamePanelContainer">
            <div id="gamePanel" />
        </div>
        <div id="tip"></div>
    </div>
    <script src="Jscript/jigsaw.js"
            type="text/javascript"></script>
</body>
</html>
```

在HTML页面的＜head＞元素中，包含了一个控制页面样式的CSS文件jigsaw.css，在该文件中，定义了页面中各元素的布局和样式属性，该文件的完整代码如下所示：

```
body
{
   text-align:center;font-size:13px
}
#tip
{
    position:absolute;top:360px;
    margin-left:40px;padding:5px;
}
.focus
{
   border:solid 3px #666;
}
.defalut
{
   border:solid 1px #ccc;
}
ul
{
   display:block;
   list-style-type:none;
   padding-left:320px;
   margin:0px;
   width:80px;
   position:absolute;
   top:8px
}
ul li
{
  text-align:left;
   padding:5px;
   border-bottom:solid 1px #ccc;
   background-color:#eee;
   height:18px;
   line-height:18px;
```

```
    cursor:pointer
}
.ulimg
{
    margin-top:86px;
}
.ulimg li
{
    padding:0px;
    width:80px;
    height:80px;
    margin-top:5px;
    background-color:#fff;
}
.img
{
    width:76px;
    height:76px;
    padding:2px;
}
#indexjigsaw
{
    margin:0 auto;
    width:406px;
}
#gamePanelContainer
{
    background-color:#eaeaea;
    border:solid 1px #666;
    padding:3px;
    width:292px;
    height:340px;
}
```

在HTML页面中，除导入CSS样式文件之外，还导入了两个JS文件，分别为jigsawplugin.js和jigsaw.js。前者为实现拼图的插件文件，后者为调用该插件的脚本代码文件。在后者的脚本代码文件中，当用户单击单选按钮和选择背景图片时，能调用拼图插件中的jigsaw方法加载分割之后小图块的功能，该文件实现的完整代码如下所示：

```
$(function() {
    var level = 3;
    var curimg = 0;
    //定义装载图片的数组变量
    var arrImg = new Array("Images/pic01.jpg",
                 "Images/pic02.jpg", "Images/pic03.jpg");
    $("#indexjigsaw ul li input").each(function(i) {
        $(this).bind("click", function() {
            switch (i) {
                case 0:
                    level = 3;
                    break;
                case 1:
                    level = 4


                    break;
                case 2:
                    level = 5
                    break;
            }
            loadjisaw(level, arrImg[curimg]);
        })
    });
    $("#indexjigsaw .ulimg li").each(function(i) {
        $(this).bind("click", function() {
            clearCss();
            $(this).removeClass("defalut");
            $(this).addClass("focus");
            curimg = i;
            switch (i) {
                case 0:
                    loadjisaw(level, arrImg[0]);
                    break;
                case 1:
                    loadjisaw(level, arrImg[1]);
                    break;
                case 2:
                    loadjisaw(level, arrImg[2]);
                    break;
            }
        })
    });
    clearCss = function() {
        $("#indexjigsaw .ulimg li").each(function() {
            $(this).removeClass("focus");
            $(this).addClass("defalut");
        })
    }
    loadjisaw = function(level, url) {
        $("#tip").html("");
        $('#gamePanel').empty();
        $.jigsaw('gamePanel', url, 290,level, 16, function() { });
    }
    //初始化
    loadjisaw(level, arrImg[0]);
});
```

在上述的脚本文件jigsaw.js中，调用了拼图插件文件jigsawplugin.js中的jigsaw（）方法，利用该方法可以获取分割后的小图片块，通过图片块的位置互换实现拼图的功能，该插件文件的完整代码如下所示：

```
$.extend({
    jigsaw: function(id, imgUri, limitWidth,
                     breakCount, swapCount, completed) {
        if (!breakCount || breakCount <= 0) breakCount = 3;
        if (!swapCount || swapCount <= 0)
            swapCount = breakCount * breakCount;
        if (!limitWidth || limitWidth <= 0)
            limitWidth = window.innerWidth;
        var swapBricks = function(b1, b2) {
```

```
    var x = b1.css('left');
    var y = b1.css('top');
    b1.css('left', b2.css('left'));
    b1.css('top', b2.css('top'));
    b2.css('left', x);
    b2.css('top', y);
    var first = null;
    for (var i = 0; i < bricks.length; i++) {
        if (bricks[i][0] === b1[0] ||
            bricks[i][0] == b2[0]) {
            if (first == null) {
                first = i;
            } else {
                var tmpBrick = bricks[first];
                bricks[first] = bricks[i];
                bricks[i] = tmpBrick;
                break;
            }
        }
    }
};
var moveCursorTo = function(brick) {
    var cursor1 = $('#cursor1');
    cursor1.css('left',parseInt(brick.css('left')) - 1);
    cursor1.css('top', parseInt(brick.css('top')) - 1);
    cursor1.css('display', 'block');
}
var hideCursor = function() {
    var cursor1 = $('#cursor1');
    cursor1.css('display', 'none');
};
var verify = function() {
    var success = true;
    for (var i = 0; i < bricks.length; i++) {
        if (bricks[i].attr('id') != i) {
            success = false;
            break;
        }
    }
    if (success) {
        for (var i = 0; i < bricks.length; i++)
            { bricks[i].unbind('click'); }
        if (completed && completed != undefined)
           { completed.apply(this, arguments); }
        $("#tip").html("Congratulation,Your success!");
    }
};
var mainPanel = $('#' + id);
mainPanel.css('position', 'relative');
var mainImage = new Image();
var bricks = [];
var selectedBrick = null;
mainImage.src = imgUri;
$(mainImage).load(function(e) {
    mainPanel.css('width', this.width + 'px');
```

```
mainPanel.css('height', this.height + 'px');
var aw = this.width / breakCount;
var ah = this.height / breakCount;
var bw = limitWidth / breakCount;
var limitHeight = this.height * limitWidth / this.width;
var bh = limitHeight / breakCount;
for (var row = 0; row < breakCount; row++) {
    for (var col = 0; col < breakCount; col++) {
        var gameCanvas = $('<canvas />');
        gameCanvas.attr('width', bw + 1 + 'px');
        gameCanvas.attr('height', bh + 1 + 'px');
        gameCanvas.css('position', 'absolute');
        gameCanvas.css('left', col * bw + 'px');
        gameCanvas.css('top', row * bh + 'px');
        gameCanvas.attr('id', row * breakCount + col);
        gameCanvas.click(function() {
            var current = $(this)
            if (selectedBrick == null) {
                selectedBrick = current;
                moveCursorTo(current);
            } else if (selectedBrick[0] == current[0]) {
                selectedBrick = null;
                hideCursor();
            } else {
                swapBricks(selectedBrick, current);
                selectedBrick = null;
                hideCursor();
                verify();
            }
        });
        var gameContext = gameCanvas[0].getContext('2d');
        gameContext.drawImage(
                 this, col * aw, row * ah, aw, ah,
                 0, 0, bw + 1, bh + 1);
        mainPanel.append(gameCanvas);
        bricks.push(gameCanvas);
    }
}
for (var i = 0; i < swapCount; i++) {
    var i1 = parseInt(Math.random() * (bricks.length));
    var i2 = parseInt(Math.random() * (bricks.length));
    while (i1 == i2) {
        i2 = parseInt(Math.random() * bricks.length);
    }
    swapBricks(bricks[i1], bricks[i2]);
}
var cursor1 = $('<div id="cursor1" />');
cursor1.css('width', bw);
cursor1.css('height', bh);
cursor1.css('display', 'none');
cursor1.css('position', 'absolute');
cursor1.css('border-top', 'solid 2px #fff');
cursor1.css('border-left', 'solid 2px #fff');
cursor1.css('border-right', 'solid 2px #fff');
cursor1.css('border-bottom', 'solid 2px #fff');
```

```
            cursor1.click(function() {
                hideCursor();
                selectedBrick = null;
            });
            mainPanel.append(cursor1);
        });
    }
});
```

13.3.4　代码分析

在本项目的jigsaw.js文件中，首先初始化两个变量level和curimg，前者表示拼图游戏的关卡值，后者表示拼图游戏的当前背景图片的索引号值。因为背景图片保存在一个名为arrImg的数组中，通过索引号可以直接获取背景图片。此外，还自定义了两个函数clearCss和loadjisaw，前者为初始背景图片的样式，后者为组织相应参数调用拼图插件中的jigsaw()方法。

然后，遍历用于控制游戏关卡的单选框按钮，获取用户选择的关卡值，保存在变量level中，根据该变量值调用loadjisaw()方法，请求加载修改关卡后的拼图数据，代码片断如下：

```
$(this).bind("click", function() {
          switch (i) {
           case 0:
                level = 3;
                break;
           case 1:
                level = 4
                break;
           case 2:
                level = 5
                break;
      }
      loadjisaw(level, arrImg[curimg]);
})
```

最后，遍历用于显示拼图游戏的背景图片元素。当用户点击某一背景图片时，改变当前图片的选中样式，并根据所选择的背景图片索

引号和关卡变量值，请求加载改变背景图片后的拼图数据，代码片断如下所示：

```
$(this).bind("click", function() {
     clearCss();
     $(this).removeClass("defalut");
     $(this).addClass("focus");
     curimg = i;
     switch (i) {
          case 0:
               loadjisaw(level, arrImg[0]);
               break;
          case 1:
               loadjisaw(level, arrImg[1]);
               break;
          case 2:

               loadjisaw(level, arrImg[2]);
               break;
           }
})
```

通过上述代码，无论用户上修改关卡还是更换拼图的背景图片，都可以即时调用拼图插件中的jigsaw()方法请求对应的拼图数据。拼图插件是一个封装的jQuery插件，在该插件中，通过获取传入的关卡值和背景图片参数，再根据关卡值遍历，在遍历过程中，获取背景图片中的指定位置和面积的小块区域，保存在图片数组中。然后，添加＜canvas＞元素，调用该元素的drawImage()，将保存的小块区域图片绘制到画布元素中，从而实现拼图数据的展示效果。最后，再次遍历图片数组，如果检测到全部图片块都拼装完成，则解除小图片的点击移动事件，显示游戏成功信息，终止该游戏，完整代码见插件文件jigsawplugin.js。

13.4　使用jQuery与HTML 5开发星球大战游戏

除使用HTML 5中的canvas元素开发拼图型的智力游戏外，还可以结合jQuery框架，利用canvas元素多项功能与属性特征，开发动态页面效果的游戏。接下来要介绍星球大战游戏，在该游戏中，当圆形的球体碰到小星星时，小星星将根据球体碰撞的部位不同，所消失的数量也不尽相同。

接下来以项目开发的方式对该功能的实现进行逐一介绍。

13.4.1　需求分析

在本项目中，需要实现的功能包括以下三个部分：

1）当页面首次加载时，球体在一个指定范围的区域内开始跳动，当撞到左右两边和底部区域时，球体会自动回弹。

2）球体在向上回弹的过程中，如果撞到顶部的小星星，将会根据撞击时的方位与力量不同，自动消失对应数量的被撞小星星。

3）当顶部所有的小星星都被撞击消失后，球体自动停止跳动。

13.4.2　界面效果

页面浏览时，圆形球体在指定的区域中不断跳动，碰撞到边线后，又自动回弹，其实现的界面效果如图13-15所示。

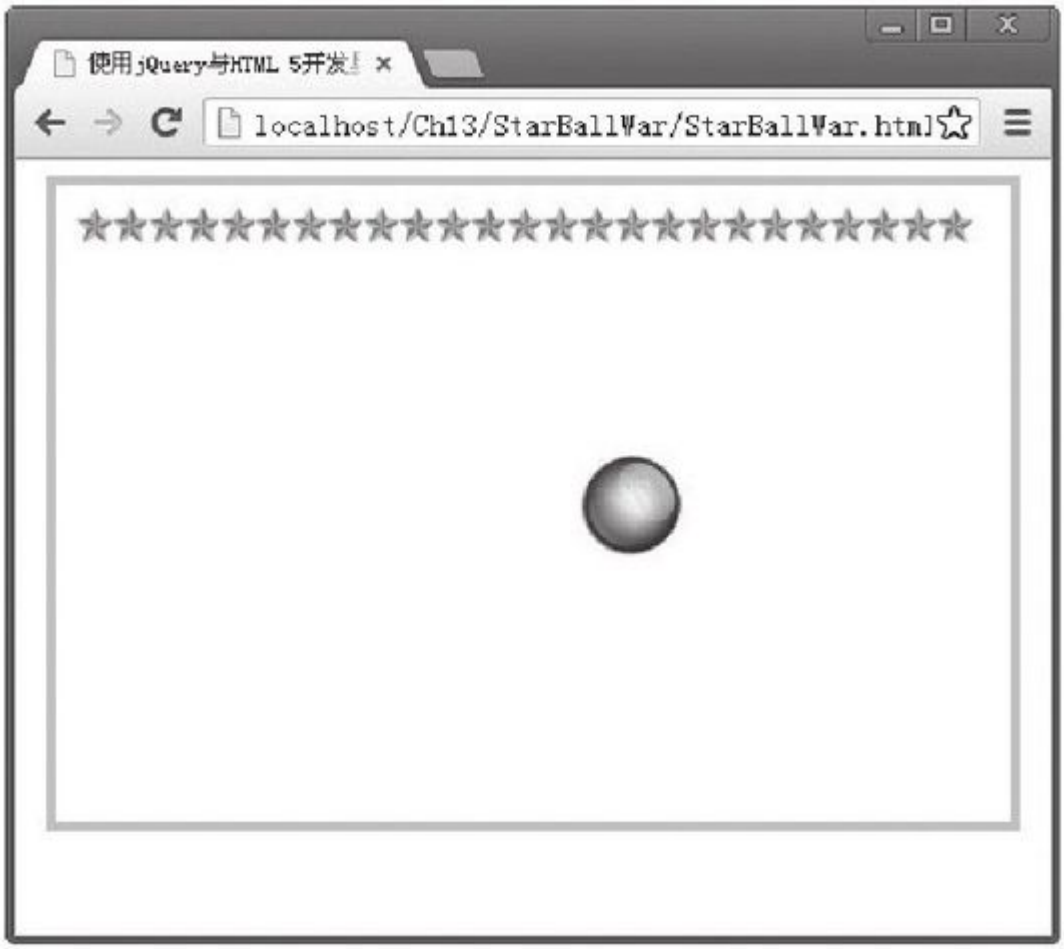


图　13-15　圆球碰撞后向上回弹

当圆球撞到顶部小星星时，被撞的小星星将会自动消失，其实现的界面效果如图13-16所示。

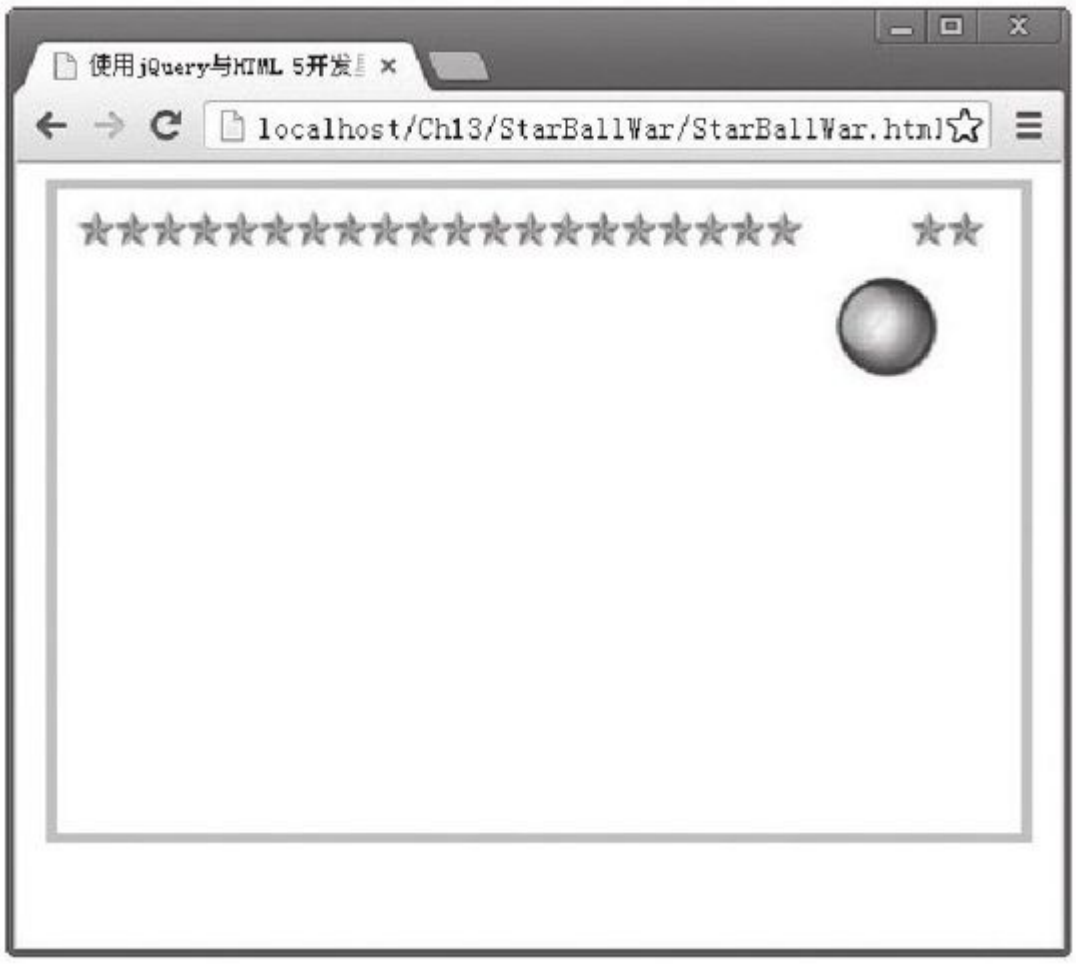


图　13-16　小星星被圆球碰撞后自动消失

当全部的小星星被碰撞消失后，圆球自动停止跳动，其实现的界面效果如图13-17所示。

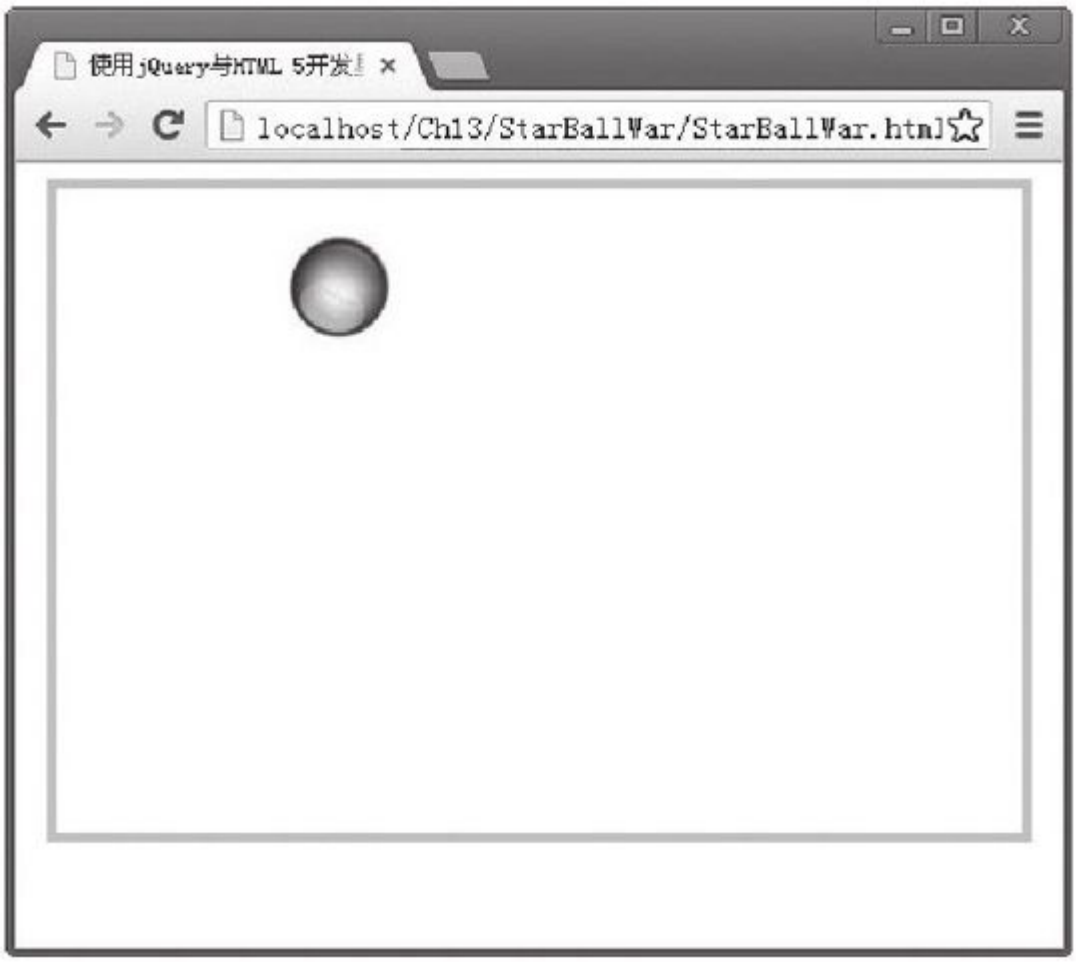


图　13-17　小星星被撞完后球体自动停止跳动

13.4.3　功能实现

根据项目的开发需求，新建一个HTML文件StarBallWar.html，在该文件中，添加一个＜canvas＞元素，通过该元素实现圆球与小星星撞击的功能。此外，在页面中还导入了控制页面样式的CSS文件starballwar.css和实现功能需求的JS文件starballwar.js，该页面完整的代码如代码清单13-4所示。

代码清单 13-4　使用 jQuery 与 HTML 5 开发星球大战游戏

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 与 HTML 5 开发星球大战游戏</title>
    <link href="Css/starballwar.css"
          rel="stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
            type="text/javascript"></script>


</head>
<body>
    <div id="container">
        <canvas id="cnvMain" width="480" height="320">
            您的浏览器不支持 html5!
        </canvas>
    </div>
    <script src="Js/starballwar.js"
            type="text/javascript"></script>
</body>
</html>
```

在导入的CSS文件starballwar.css中，设置了整体页面的布局和内容展示的大小，该样式文件的完整代码如下所示：

```
body
{
    text-align:center;
    font-size:13px
}
#container
{
    margin:0 auto;
    border:solid 5px #ccc;
    cursor:none;
    width:480px;
    height:320px
}
```

在HTML页面中，除导入样式文件之外，还导入了一个脚本文件starballwar.js，在该文件中以面向对象的编程方式。在setInterval()方法中计时反复调用＜canvas＞元素的drawImage()方法，从而实现圆形球体不断动态回弹的页面效果，该脚本文件的完整代码如下所示：

```
//全局变量
var imgAct = new Image();      //圆形球体
var ctx;                       //2D画布
var screenWidth;               //画布宽度
var screenHeight;              //画布高度
var speed = 2;                 //不变速度常量
var horizontalSpeed = speed;   //水平速度，随着碰撞会发生改变
var verticalSpeed = -speed;    //垂直速度，负数向上，随碰撞发生改变
var imgActAngle = 2;           //圆形球体旋转的角度
var star0 = new Image();       //第一个小星星
var gameloopId;                //记住循环的变量
var checkPrize;
//定义一个公用游戏对象
function objGame() {
    this.x = 0;
    this.y = 0;
    this.image = null;
}
//定义圆形球体继承游戏对象objGame
function objAct() { };
```

```
objAct.prototype = new objGame();
objAct.prototype.angle = 0;
var objAct = new objAct();
//定义小星星数组 objStars 和对象 objStar，继承游戏对象 objGame
var arrStar = new Array();
function objStar() { };
objStar.prototype = new objGame(); //继承游戏对象 objGame
objStar.prototype.row = 0;          //小星星行位置
objStar.prototype.col = 0;          //小星星列位置
objStar.prototype.hit = false;      //小星星是否被撞到
var ballgame = {
    HasActHitEdge: function() {
        //圆形球体碰到右边边界
        if (objAct.x > screenWidth - objAct.image.width) {
            if (horizontalSpeed > 0)//假如向右移动
                horizontalSpeed = -horizontalSpeed; //改变水平速度方向
        }
        //圆形球体碰到左边边界
        if (objAct.x < -10) {
            if (horizontalSpeed < 0)//假如向左移动
                horizontalSpeed = -horizontalSpeed; //改变水平速度方向
        }
        //圆形球体碰到下面边界
        if (objAct.y == screenHeight - objAct.image.height) {
            verticalSpeed = -speed; //改变垂直速度，即向上移动
        }
        //圆形球体碰到上边边界
        if (objAct.y < 0) {
            verticalSpeed = -verticalSpeed;
        }
    },
    chkHitEach: function(obj1, obj2, overlap) {
        //overlap 是重叠的区域值
        A1 = obj1.x + overlap;
        B1 = obj1.x + obj1.image.width - overlap;
        C1 = obj1.y + overlap;
        D1 = obj1.y + obj1.image.height - overlap;
        A2 = obj2.x + overlap;
        B2 = obj2.x + obj2.image.width - overlap;
        C2 = obj2.y + overlap;
        D2 = obj2.y + obj2.image.height - overlap;
        //假如它们在 x- 轴重叠
        if (A1 > A2 && A1 < B2 || B1 > A2 && B1 < B2) {
            //判断 y- 轴重叠
            if (C1 > C2 && C1 < D2 || D1 > C2 && D1 < D2) {
                //碰撞
                return true;
            }
        }
        return false;
    },
    InitStars: function() {
```

```
        var count = 0;
        var x = 0;
        for (var y = 0; y < 25; y++) {
            star = new objStar();
            star.image = star0;
            star.row = x;
            star.col = y;
            star.x = 18 * star.col + 10; //x轴位置
            star.y = 18 * star.row + 10; //y轴位置
            //装入小星星数组，用来描绘
            arrStar[count] = star;
            count++;
        }
    },
    DrawStars: function() {
        for (var x = 0; x < arrStar.length; x++) {
            var curStar = arrStar[x];
            //假如没有被撞击，则描绘
            if (!curStar.hit) {
                ctx.drawImage(curStar.image, arrStar[x].x,arrStar[x].y);
            }
        }
        if (ballgame.AllPrizesHit()) {
            clearInterval(gameloopId);
        }
    },
    HasActHitStar: function() {
        //取出所有小星星
        for (var x = 0; x < arrStar.length; x++) {
            var star = arrStar[x];
            //假如没有碰撞过
            if (!star.hit) {
                //判断碰撞
                if (ballgame.chkHitEach(star, objAct, 0)) {
                    star.hit = true;
                    //圆形球体反弹下沉
                    verticalSpeed = speed;
                }
            }
        }
    },
    AllPrizesHit: function() {
        for (var c = 0; c < arrStar.length; c++) {
            checkPrize = arrStar[c];
            if (checkPrize.hit == false)
                return false;
        }
        return true;
    },
    LoadImages: function() {
        imgAct.src = "images/act.png"; // 圆形球体
        star0.src = "images/star.png"; // 第一个小星星
        objAct.image = imgAct;
    }
};
```

```
//初始化
$(function() {
    ballgame.LoadImages();
    //获取2d画布
    ctx = document.getElementById('cnvMain').getContext('2d');
    //画布宽度
    screenWidth = parseInt($("#cnvMain").attr("width"));
    screenHeight = parseInt($("#cnvMain").attr("height"));
    //初始化圆形球体
    objAct.x = parseInt(screenWidth / 2);
    objAct.y = parseInt(screenHeight / 2);
    //初始化小星星
    ballgame.InitStars();
    gameloopId = setInterval(function() {
        ctx.clearRect(0, 0, screenWidth, screenHeight);
        ctx.save();
        //绘制小星星
        ballgame.DrawStars();
        //改变圆形球体X和Y位置
        objAct.x += horizontalSpeed;
        objAct.y += verticalSpeed;
        //改变翻滚角度
        objAct.angle += imgActAngle;
        //以当前圆形球体的中心位置为基准
        ctx.translate(objAct.x + (objAct.image.width / 2),
                       objAct.y + (objAct.image.height / 2));
        //根据当前圆形球体的角度轮换
        ctx.rotate(objAct.angle * Math.PI / 90);
        //描绘圆形球体
        ctx.drawImage(objAct.image,
                      -(objAct.image.width / 2),
                      -(objAct.image.height / 2));
        ctx.restore();
        //检测是否碰到边界
        ballgame.HasActHitEdge();
        //检测圆形球体碰撞小星星
        ballgame.HasActHitStar();
    }, 10);
});
```

13.4.4 代码分析

在本项目的JavaScript代码中，首先通过setInterval()方法反复执行圆形球体的位置变化和重新绘制，从而实现球体的动态回弹效果，完成该功能的代码片断如下：

```
ameloopId = setInterval(function() {
        ctx.clearRect(0, 0, screenWidth, screenHeight);
        ctx.save();
        ...
        // 改变圆形球体 X 和 Y 位置
        objAct.x += horizontalSpeed;
        objAct.y += verticalSpeed;
        // 改变翻滚角度
        objAct.angle += imgActAngle;


    // 以当前圆形球体的中心位置为基准
    ctx.translate(objAct.x + (objAct.image.width / 2),
                  objAct.y + (objAct.image.height / 2));
    // 根据当前圆形球体的角度轮换
    ctx.rotate(objAct.angle * Math.PI / 90);
    // 描绘圆形球体
    ctx.drawImage(objAct.image,
                  -(objAct.image.width / 2),
                  -(objAct.image.height / 2));
    ctx.restore();
    ...
}, 10);
```

在这段反复执行的代码中，有两个重要的变量horizontalSpeed和verticalSpeed，前者为水平速度，后者为垂直速度，这两个值决定了圆形球体回弹时的方向。

在圆形球体不断重新绘制的过程中，它本身的位置将会与＜canvas＞元素的边界发生交叉，因此，需要判断球体与＜canvas＞元素四周的哪个还边界发生了交叉，从而做出相应的回弹方向处理，这一功能由自定义函数HasActHitEdge来实现，该函数的代码片断如下：

```
//圆形球体碰到右边边界
if (objAct.x > screenWidth - objAct.image.width) {
    if (horizontalSpeed > 0)                    //假如向右移动
        horizontalSpeed = -horizontalSpeed; //改变水平速度方向
}
//圆形球体碰到左边边界
if (objAct.x < -10) {
     if (horizontalSpeed < 0)                   //假如向左移动
         horizontalSpeed = -horizontalSpeed; //改变水平速度方向
}
//圆形球体碰到下面边界
if (objAct.y == screenHeight - objAct.image.height) {
     verticalSpeed = -speed;                    //改变垂直速度。即向上移动
}
//圆形球体碰到上边边界
if (objAct.y < 0) {
     verticalSpeed = -verticalSpeed;
 }
```

除了需要处理圆形球体碰撞＜canvas＞元素四周边缘外，还需要判断圆形球体与小星星是否发生了碰撞，而这个判断需要根据两个物体相撞时是否有重叠区域来确定，如果有重叠区域，认为发生了相撞，否则没有发生相撞。

在本项目中，鉴定两个物体是否发生了相撞的功能，由自定义函数chkHitEach来实现，该函数的代码片断如下：

```
//overlap是重叠的区域值
A1 = obj1.x + overlap;
B1 = obj1.x + obj1.image.width - overlap;
C1 = obj1.y + overlap;
D1 = obj1.y + obj1.image.height - overlap;
A2 = obj2.x + overlap;

B2 = obj2.x + obj2.image.width - overlap;
C2 = obj2.y + overlap;
D2 = obj2.y + obj2.image.height - overlap;
//假如它们在x-轴重叠
if (A1 > A2 && A1 < B2 || B1 > A2 && B1 < B2) {
     //判断y-轴重叠
     if (C1 > C2 && C1 < D2 || D1 > C2 && D1 < D2) {
          //碰撞
          return true;
     }
}
return false;
```

从上述代码中可以看出，判断两个物体相撞依据是，当一个物体由下而上去撞击另一个物体时，那么这两个物体在y轴方向中如果有重叠部分，则认为为发生了相撞，返回true值。否则没有相撞，返回false值。

根据这个方法，当圆形球体碰撞了小星星时，如果两物体在y轴上有重叠区域，则发生了相撞，如果相撞，那么球体向下移动，小星星的hit属性为true，一旦小星星的属性为true，则在下一次重新绘制中不会再出现，该功能分别有两个自定义的函数HasActHitStar和DrawStars来实现，代码详细见上述starballwar.js文件，在此不再赘述。

当所有的小星星hit属性都为true时，则认为游戏结束，则使用clearInterval()方法清除setInterval()方法执行时的变量gameloopId，终止游戏，部分代码如下：

```
AllPrizesHit: function() {
          for (var c = 0; c < arrStar.length; c++) {
           checkPrize = arrStar[c];
           if (checkPrize.hit == false)
               return false;
          }
          return true;
}
```

上述代码用于检测小星星的hit属性值，如果全部为true返回true，表示游戏结束，否则返回false，根据这一结果，执行下列代码终止游戏：

```
if (ballgame.AllPrizesHit()) {
     clearInterval(gameloopId);
}
```

13.5　本章小结

本章精选4个jQuery与HTML 5结合的热门案例，以项目开发的方式，由浅入深地展示完整开发流程。相信通过本章的学习，读者将初步掌握应用jQuery框架开发HTML 5应用的基本方法和步骤。

第14章　jQuery Mobile基础知识

本章内容

初识jQuery Mobile

jQuery Mobile基本组件

jQuery Mobile API接口应用

本章小结

jQuery Mobile是专门针对移动终端设备浏览器开发出的一套轻量级Web脚本框架。该框架基于jQuery和jQuery UI，统一用户系统接口，能够无缝隙运行在所有流行的移动平台之上，易于主题化的设计与建造。它的出现，打破了传统JavaScript对移动终端设备的脆弱支持，使开发一个跨移动平台的Web应用真正成为可能。

本章使用的示例全部基于jQuery Mobile框架1.3.0版本，在jQuery框架1.8.4版本下进行开发。通过本章的详实介绍，逐步带领大家进入jQuery Mobile的精彩世界。

14.1　初识jQuery Mobile

在jQuery与jQuery UI的基础之上，推出的jQuery Mobile框架，其主旨就是为开发者在进行移动项目开发的过程中，提供统一的接口与特征，依附于强大的jQuery类库，节省大量JavaScript代码的开发时间，提高项目开发的效率。

14.1.1　jQuery Mobile框架简介

jQuery Mobile框架历经几个版本的迭代，功能逐步成熟与稳定，其最新的1.3.0版本在原有功能的基础之上，新增了许多强大的功能，主要体现在响应式的Web设计优化，新添加许多酷绚的部件，包括面板、双范围滑块和两个不同响应式的表格模块。同时，针对Ajax导航系统做了功能上的重构，使用其执行的效果在不同浏览器中更加接近一致。

14.1.2 jQuery Mobile工作原理

jQuery Mobile的工作原理是：通过提供可触摸的UI小部件和Ajax导航系统，使页面支持动画式切换效果，以页面中的元素标记为事件驱动对象，当触摸或点击时进行触发。最后，在移动终端的浏览器中实现一个个应用程序的动画展示效果。

与开发桌面浏览中的Web页面相似，构建一个jQuery Mobile页面也是十分容易，接下来，我们详细介绍如何开发第一个jQuery Mobile页面。

14.1.3　开发第一个jQuery Mobile页面

jQuery Mobile通过＜div＞元素组织页面结构，根据元素的"data-role"属性设置角色，每一个拥有"data-role"属性的＜div＞元素就是一个容器，它可以放置其他的页面元素，接下来通过一个简单示例来进行阐述。

示例14-1　开发第一个jQuery Mobile页面

（1）功能说明

创建一个jQuery Mobile的基本框架页面，并在页面中输出"Hello World!"字样。

（2）实现代码

新建一个HTML页面14-1.html，加入代码如代码清单14-1所示。

代码清单 14-1　开发第一个 jQuery Mobile 页面

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Mobile 应用程序 </title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
</head>
<body>
 <div data-role="page">
 <div data-role="header"><h1>jQuery Mobile</h1></div>
 <div data-role="content"><p>Hello World!</p></div>
 <div data-role="footer">
        <h4>?2013 rttop.cn studio</h4>
 </div>
 </div>
</body>
</html>
```

（3）页面效果

为了更好地在PC端浏览jQuery Mobile页面的最终效果，可以下载Opera公司的移动模拟器Opera Mobile Emulator，本章全部的页面效果都在该模拟器中演示。

该页面在Opera Mobile Emulator模拟器中执行的效果如图14-1所示。

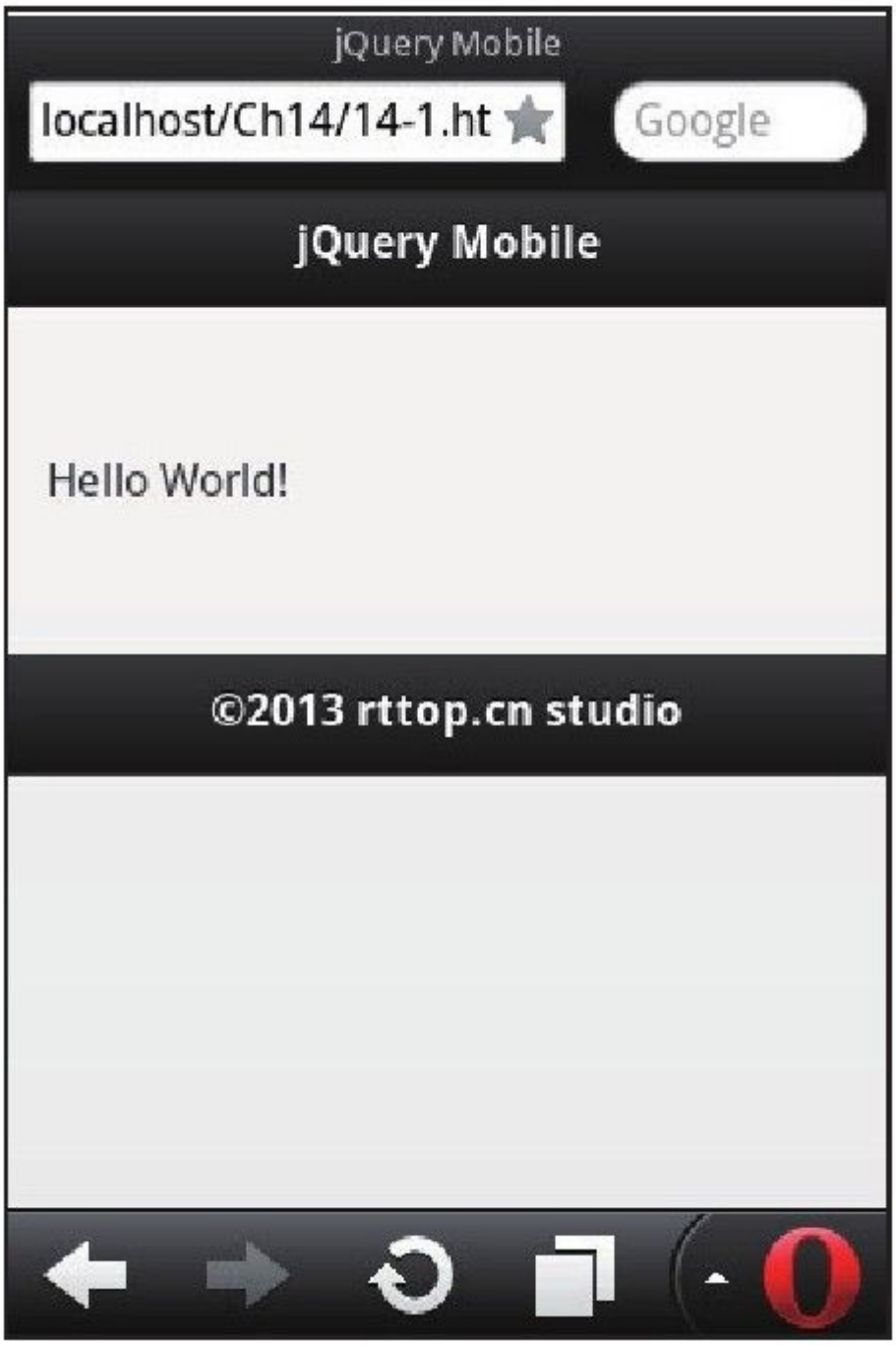


图　14-1　第一个jQuery Mobile页面

（4）代码分析

在本示例代码中，为了更好地支持HTML 5的新增功能与属性，第一行以HTML 5的声明文档开始，即添加如下代码：

```
<!DOCTYPE html>
```

在＜head＞元素中，添加了一个名称为"viewport"的＜meta＞，并设置了该元素的"content"属性，代码如下所示：

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

这行代码的功能是：设置移动设备中浏览器缩放的宽度与等级。通常情况下，移动设备的浏览器会默认一个约"900px"的宽度来显示页面，这种宽度会导致屏幕缩小，页面放大，不适合浏览；通过在页面中添加＜meta＞元素，并设置"content"的属性值为"width=device-width,initial-scale=1"，可以使页面的宽度与移动设备的屏幕宽度相同，更加适合用户浏览。

在接下来的＜body＞元素中，将第一个＜div＞元素的"data-role"属性设置为"page"，形成一个容器，然后在容器中分别添加3个＜div＞元素，并依次将"data-role"属性设置为"header"、"content"、"footer"，从而形成了一个标准的jQuery Mobile页面的框架，并在"data-role"属性值为"content"的正文区域显示"Hello World!"字样。

14.2 jQuery Mobile基本组件

在jQuery Mobile中，使用了HTML 5的新特征——自定义元素属性（dataset）。该属性是HTML 5新增加的特征，其格式要求属性名前必须带有"data-"字符，字符后面允许用户自定义属性名称，如下代码：

```
<div id="title" data-title="jQuery Mobile"
                data-time="2013-03-15" >
      jQuery 权威指南（第二版）
</div>
```

上述代码在定义＜div＞元素时，使用HTML 5中的自定义元素属性方法新增了"title"、"time"两个属性，定义完成后，可以通过如下JavaScript代码获取属性值。

```
var title = document.getElementById("title");
if (title.dataset) {
    alert(title.dataset.title);
  }
else {
     alert("error!");
}
```

上述代码在获取dataset方法设置属性时，考虑到并非所有的浏览器都支持dataset方法设置的属性。因此，先通过title.dataset方式进行检测，如果支持，则通过使用title.dataset.title方式获取"title"属性的值；如果在支持dataset方法定义属性的浏览器中执行上述代码时，则在弹出的对话框中显示"jQuery Mobile"字符。

通过自定义元素属性（dataset）的设置赋予HTML元素不同的功能，从而在jQuery Mobile中形成不同的组件类型，如对话框、工具栏、按钮等组件，接下来我们逐一进行介绍。

14.2.1　对话框元素

在jQuery Mobile中，创建对话框的方式十分方便，只需要在指向页面的链接元素中添加一个"data-rel"属性，并将该属性值设置为"dialog"，点击该链接时，打开的页面将以一个对话框的形式展示在浏览器中，当点击对话框中的任意链接时，打开的对话框将自动关闭，并以“回退”的形式切换至上一页中。

下面通过一个示例来说明如何创建一个简单对话框。

示例14-2　以对话框的形式打开目标URL地址

（1）功能说明

新建一个HTML页面，在页面中添加一个＜a＞元素，并将该元素的"data-rel"属性设置为"dialog"，表示以对话框的形式打开链接元素指定的目标URL地址。

（2）实现代码

新建一个HTML页面14-2.html，加入代码如代码清单14-2所示。

代码清单 14-2　以对话框的形式打开目标 URL 地址

```
<!DOCTYPE html>
<html>
<head>
    <title>对话框元素</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
</head>
<body>
  <div data-role="page">
  <div data-role="header"><h1>对话框</h1></div>
  <div data-role="content">
    <p>
    <a href="dialog.html"
       data-rel="dialog"
       data-transition="pop">点击打开对话框
    </a>
  </p>
</div>
<div data-role="footer">
      <h4>©2013 rttop.cn studio</h4>
  </div>
</div>
</body>
</html>
```

另外，创建一个用于对话框的HTML页面dialog.html，加入如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <title>简单的对话框</title>
    <meta name="viewport" content="width=device-width" />
</head>
<body>
  <div data-role="page">
  <div data-role="header"><h1>对话框主题</h1></div>
  <div data-role="content">
          <p>对话框正文</p>
  </div>
  <div data-role="footer">
          <h4>?2013 rttop.cn studio</h4>
  </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-2所示。

图　14-2　以对话框的形式打开目标URL地址

（4）代码分析

在本示例中，设置链接的"data-rel"属性为"dialog"值后，通过该链接打开的页面将以对话框的形式展示在当前页面中。该对话框以

模式的方式浮在当前页的上面，背景添加深色，四周以圆角的效果显示，并在左上角自带一个“×”的关闭按钮，单击该按钮后，对话框将自动关闭。

14.2.2　工具栏元素

通常情况下，工具栏由移动应用的头部栏、工具条、尾部栏三部分组成，分别被放置在移动应用程序中的标题部分、内容部分、页尾部分，并通过添加不同样式和定位工具栏的位置，满足和实现各种移动应用的页面需求和效果。

工具栏元素中的头部栏由标题文字和左右两边的按钮构成。标题文字通常使用＜h＞标记，取值范围在1～6之间，常用＜h1＞标记，无论取值是多少，在同一个移动应用项目中，都要保持一致。标题文字的左右两边可以分别放置一或二个按钮，用于标题中的导航操作。

下面通过一个简单的示例来说明如何创建一个工具栏元素中的头部栏。

示例14-3　创建一个工具栏元素的头部栏

（1）功能说明

在新建的HTML页面中，分别添加两个ID号为"e1"、"e2"的"page"容器，并在两个容器的头部栏中分别添加两个按钮，左侧为“上一张”，右侧为“下一张”。单击第一个容器的

“下一张”按钮时，切换到第二个容器；单击第二个容器的“上一张”按钮时，又返回到第一个容器中。

（2）实现代码

新建一个HTML页面14-3.html，加入代码如代码清单14-3所示。

代码清单 14-3　创建一个工具栏元素的头部栏

```
<!DOCTYPE html>
<html>
<head>
    <title>工具栏元素</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
    <style type="text/css">
        img{border:solid 1px #666;padding:5px}
    </style>
</head>
<body>
   <div data-role="page" id="e1">
       <div data-role="header" data-position="inline">
           <a href="#" data-icon="arrow-l">上一张</a>
           <h1>图片</h1>
           <a href="#e2" data-icon="arrow-r">下一张</a>
       </div>
       <div data-role="content" style="text-align:center">
           <img src="Images/2011年作品.jpg" />
       </div>
       <div data-role="footer">
           <h4>?2013 rttop.cn studio</h4>
       </div>
   </div>
   <div data-role="page" id="e2">
       <div data-role="header" data-position="inline">
           <a href="#e1" data-icon="arrow-l">上一张</a>
           <h1>图片</h1>
           <a href="#" data-icon="arrow-r">下一张</a>
       </div>
       <div data-role="content" style="text-align:center">
           <img src="Images/2010年作品.jpg" />
       </div>
       <div data-role="footer">
            <h4>?2013 rttop.cn studio</h4>
       </div>
    </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-3所示。

图　14-3　创建一个工具栏元素的头部栏

（4）代码分析

在本示例中，头部栏通过添加"inline"属性进行定位，使用这种定位的模式，可以确保头部栏在更多的移动浏览器中显示，而无须再编写其他的JavaScript或CSS代码。

头部栏中的按钮链接元素，是头部栏的首个元素，默认位置是在标题的左侧，默认按钮个数只有一个，当在标题左侧添加两个链接按钮时，左侧链接按钮会自动按排列顺序保留第一个，第二个按钮会自动放置在标题的右侧。因此，在头部栏中放置链接按钮时，由于内容长度的限制，尽量在标题栏的左右两侧分别放置一个链接按钮。

14.2.3　内容布局

在jQuery Mobile中，提供了许多非常有用的工具与组件，如多列的网格布局、折叠形的面板控制，通过这些组件，可以帮助开发者快速实现正文区域内容的格式化。

通过jQuery Mobile提供的CSS样式"ui-grid"可以实现内容的网格布局，该样式有4个预设的配置布局，"ui-grid-a"、"ui-grid-b"、"ui-grid-c"、"ui-grid-d"，分别对应两列、三列、四列、五列的网格布局形式，可以最大范围满足页面多列的需求。

在jQuery Mobile中，除使用样式"ui-grid"显示多列的网格效果之外，还可以对指定的区块进行折叠，要实现对区块的折叠，需要进行以下3步的操作：

1）创建一个＜div＞容器，并将该容器的"data-role"属性设为"collapsible"表示该容器是一个可折叠的区块。

2）在容器中，添加一个＜h3＞标题文字标记,该标记以按钮的形式展示，并在按钮的左侧有一个“+”号，表示该标题可以点开。

3）在标题的下面放置需要折叠显示的内容，通常使用＜p＞段落元素，当用户点击标题中的“+”号时，显示＜p＞元素中的内容，标

题左侧中“+”号变成“-”号再次点击时，隐藏＜p＞元素中的内容，标题左侧中“-”号变成“+”号。

除了在正文中将＜div＞容器实现折叠效果显示内容之外，jQuery Mobile还允许对折叠的区块进行嵌套显示。即在一个折叠区域块的内容中，再添加一个折叠区块，依此类推。

折叠区块除了可以嵌套外，还可以形成折叠组，实现的方法是：在一个"data-role"属性为"collapsible-set"的＜div＞容器中，添加多个折叠区块，而这些区块就是折叠组区块。因为它们同属于一个容器，在视觉上形成“手风琴”的效果。在同一时间，折叠组中只有一个折叠区块是被打开的，当打开别的折叠区块时，其他“组成员”自动关闭。

下面通过一个简单的示例来说明如何创建一个内容布局中可嵌套的折叠组。

示例14-4　创建一个内容布局中可嵌套的折叠组

（1）功能说明

新建一个HTML页面，添加一个"data-role"属性为"collapsible-set"的折叠组容器。在该容器中增加3个折叠区块，标题分别对应“图

书”、“音乐”、“影视”。初次显示时，“音乐”折叠区块为打开状态。

（2）实现代码

新建一个HTML页面14-4.html，加入代码如代码清单14-4所示。

代码清单 14-4　创建一个内容布局中可嵌套的折叠组

```
<!DOCTYPE html>
<html>
<head>
    <title>内容布局</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
</head>
<body>
   <div data-role="page">
    <div data-role="header"><h1>头部栏标题</h1></div>
    <div data-role="collapsible-set">
      <div data-role="collapsible">
        <h3>图书</h3>
        <p><a href="#">文艺</a></p>
        <p><a href="#">少儿</a></p>
        <p><a href="#">社科</a></p>
      </div>
    <div data-role="collapsible" data-collapsed="false">
       <h3>音乐</h3>
       <p><a href="#">流行</a></p>
        <p><a href="#">民族</a></p>
        <p><a href="#">通俗</a></p>
     </div>
    <div data-role="collapsible">
       <h3>影视</h3>
       <p><a href="#">欧美</a></p>


        <p><a href="#">怀旧</a></p>
        <p><a href="#">娱乐</a></p>
     </div>
     </div>
     <div data-role="footer">
            <h4>?2013 rttop.cn studio</h4>
     </div>
     </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-4所示。

图　14-4　创建一个内容布局中可嵌套的折叠组

（4）代码分析

在本示例中，折叠组中所有的折叠区块默认状态都是收缩的，如果需要在默认状态下使某个折叠区块为下拉状态，只要将该折叠区块的"data-collapsed"属性值设置为"false"。如在本示例中，就将标题为“音乐”的折叠区块的"data-collapsed"属性值设置为"false"，需

要注意，由于同处在一个折叠组内，这种下拉状态在同一时间只允许有一个。

14.2.4　按钮

在jQuery Mobile中，按钮由两类元素形成。一类是＜a＞元素，通过将该元素的"data-role"属性值设置为"button"，jQuery Mobile便会自动给该元素一些Class样式属性，形成可点击的按钮形状；另一类是在表单内，jQuery Mobile会自动将＜input＞元素中"type"属性值为"submit"、"reset"、"button"、"image"形成按钮的样式，而无须添加"data-role"属性。另外，在内容中放置按钮时，可以采用内嵌或按钮组的方式进行排版。

在jQuery Mobile中，被样式化的按钮元素默认都是块状，能自动填充页面宽度，但也可以取消该默认效果，只需要在按钮的元素中添加"data-inline"属性，并将该属性值设为"true"。那么，该按钮将会根据它内容中文字和图片的宽度自动进行缩放，形成一个宽度紧凑型的按钮。

如果想要对缩放后的按钮进行同一行显示，可以在多个按钮的外层中增加一个＜div＞容器，并在该容器中将"data-inline"属性值设为"true"。这样就可以使容器中的按钮自动通过样式缩放至最小宽度，并且有浮动效果，可以在一行中显示。

在内联的按钮中，如果想使两个以上的按钮既在同一行，又能通过样式自动均分页面宽度，可以使用网格分栏的方式，将多个按钮放置在一个分栏后的同一行中。

接下来通过一个简单示例来说明按钮的实现过程。

示例14-5　通过分栏的方式在页面中添加两个按钮

（1）功能说明

新建一个HTML页面，用分栏的方式在页面中添加一个普通按钮和一个表单按钮，使两个按钮在同一行显示。

（2）实现代码

新建一个HTML页面14-5.html，加入代码如代码清单14-5所示。

代码清单 14-5　通过分栏的方式在页面中添加两个按钮

```
<!DOCTYPE html>
<html>
<head>
    <title>按钮</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
</head>
<body>
  <div data-role="page">
  <div data-role="header"><h1>头部栏</h1></div>
  <div class="ui-grid-a">
  <div class="ui-block-a">
        <a href="#" data-role="button"
        class="ui-btn-active">确定
        </a>
  </div>
  <div class="ui-block-b">
    <input type="submit" value="取消" /></div>
</div>
<div data-role="footer">
   <h4>?2013 rttop.cn studio</h4>
</div>
</div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-5所示。

图　14-5　通过分栏的方式在页面中添加一个普通按钮和一个表单按钮

（4）代码分析

在本示例中，运用分栏容器使两个按钮显示在同一行。由于这两个按钮的宽度可以与移动终端浏览器的宽度进行自动等比缩放，因此，这样的两个按钮显示在同一行，可以适应移动终端中各种不同分辨率的浏览器。

如果希望形成的按钮不与浏览器等比缩放，且多个按钮也要在同一行显示，可以将按钮元素的"data-inline"属性值设置为"true"，如果是本示例的代码则修改为：

```
... 省略部分代码
<a href="#" data-role="button" class="ui-btn-active"
            data-inline="true"> 确定 </a>
<a href="#" data-role="button" data-inline="true"> 取消 </a>
... 省略部分代码
```

上述代码同样可以使两个按钮以内联的方式显示在页面中同一行，只是固定了宽度，不能与浏览器的宽度进行等比缩放。

14.2.5　表单元素

在HTML元素中，表单占有十分重要的地位，针对表单，jQuery Mobile提供了一套完全基于HTML原始代码，又适合触摸操作的框架。在该框架下，所有的表单元素先由原始的代码升级为jQuery Mobile组件，然后调用各自组件提供的方法与属性，实现在jQuery Mobile下表单元素的各项操作。例如，在jQuery Mobile表单中，一个"type"属性值为"checkbox"的元素，先通过对应的"checkboxradio"插件升级为组件，完成相应数据的初始化后，就可以调用jQuery UI中组件的方法与属性，实现该表单元素的相应功能。

注意　需要说明的是，在表单中，各元素通过原始HTML代码升级为jQuery Mobile是自动完成的。当然，也可以阻止这种升级行为，只要将该表单元素的"data-role"属性值设置为"none"即可。另外，由于在单个页面中，可能会出现多个"page"容器，为了保证表单在提交数据时的唯一性，必须确保每一个表单的ID号是唯一的。

在jQuery Mobile表单中，文本输入包括文本输入框和文本输入域及HTML 5中新增的输入类型。文本输入框使用标准的HTML原始元素，借助jQuery Mobile的渲染效果，使用其更易于触摸型使用；在jQuery Mobile中使用的文本输入域的高度会自动增加，无须因高度问题拖动滑动条。

另外，HTML 5中新增的输入类型"number"，在jQuery Mobile中会被渲染成除数字输入框外，还在输入框的最右端有两个可调节大小的“+”和“-”按钮，方便移动终端的用户修改输入框中的数字使用。

接下来通过一个简单示例来说明表单中文本输入框元素实现的过程。

示例14-6　表单中文本输入框元素的实现

（1）功能说明

新建一个HTML页面，并在内容区域中创建三个不同的输入框元素，分别对应"search"、"text"、"number"类型，用于显示在jQuery Mobile中不同类型的输入框元素异样的渲染效果。

（2）实现代码

新建一个HTML页面14-6.html，加入代码如代码清单14-6所示。

代码清单 14-6　表单中文本输入框元素的实现

```
<!DOCTYPE html>
<html>
<head>
    <title>表单元素</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
</head>
<body>
  <div data-role="page">
    <div data-role="header">
      <h1>头部栏</h1>
    </div>
    <div data-role="content">
      搜索: <input type="search"
                   name="password" id="search" value="" />
      姓名: <input type="text"
                   name="name" id="name" value="" />
      书号: <input type="number"
                   name="number" id="number" value="0"/>
    </div>
    <div data-role="footer">
        <h4>?2013 rttop.cn studio</h4>
    </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-6所示。

图　14-6　表单元素

（4）代码分析

从本示例的示意图可以看出，在jQuery Mobile中，"type"类型是"search"的搜索输入文本框的外围有圆角，最左端有一个圆型的搜索图标，当输入框中有内容字符时，它的最右侧会出现一个圆形的“×”按钮，单击该按钮可以清空输入框中的内容。

在"type"类型是"number"的数字输入文本中，单击最右端的上下两个调整按钮，可以动态改变数字输入框中的值的大小，使用十分方便。

14.2.6　列表视图

在jQuery Mobile中，如果在＜ul＞元素中，将"data-role"属性值设置为"listview"，便形成了一个无序的列表，并且将会对列表渲染对应的样式，如列表的宽度与屏幕同比缩放，在列表选项的最右侧，有一个带右箭头的链接图标。

当一个＜ul＞元素被定义为列表后，jQuery Mobile将对该列表进行对应样式的渲染，列表中的选项也变得易于触摸，如果单击某选项，将会通过Ajax的方式异步请求一个对应的URL地址，并在DOM中，创建一个新的页面，借助默认切换的效果，进入该页面中。

在jQuery Mobile中，＜ul＞、＜ol＞元素不仅可以被渲染成列表，而且该列表还可以进行嵌套，实现的方法是在父列表＜ul＞、＜ol＞元素的＜li＞标签中，添加子列表＜ul＞或＜ol＞元素，形成嵌套列表的格局；当用户点击父列表中的某个选项时，jQuery Mobile会自动生成一个包含子列表＜ul＞或＜ol＞元素全部内容的新页面，但页面的主题则为父列表的标题内容。

接下来通过一个简单示例来说明嵌套列表的实现过程。

示例14-7　嵌套列表的实现

（1）功能说明

新建一个HTML页面，添加一个＜ul＞元素，并在该元素中增加两个＜li＞选项元素，主题分别为“图书”、“音乐”；然后，在两个＜li＞元素中，分别添加另外两个与之对应的＜ul＞列表元素作为子列表，点击父列表中某个选项时，将自动切换至对应的子列表页面中。

（2）实现代码

新建一个HTML页面14-7.html，加入代码如代码清单14-7所示。

代码清单 14-7　列表视图

```
<!DOCTYPE html>
<html>
<head>
    <title>列表视图</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
```

```
    <script src="Js/jquery-1.8.2.min.js"
          type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
          type="text/javascript"></script>
</head>
<body>
  <div data-role="page">
    <div data-role="header"><h1>头部栏</h1></div>
    <ul data-role="listview">
      <li>
        <h3>图书</h3>
        <p>一本好书，就是一个良师益友。</p>
        <ul>
            <li><a href="#">计算机</a></li>
            <li><a href="#">社科</a></li>
        </ul>
      </li>
      <li>
      <h3>音乐</h3>
       <p>好的音乐可以陶冶人的情操。</p>
       <ul>
            <li><a href="#">流行</a></li>
            <li><a href="#">通俗</a></li>
       </ul>
      </li>
    </ul>
    <div data-role="footer">
      <h4>©2013 rttop.cn studio</h4>
    </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-7所示。

图　14-7　列表视图

（4）代码分析

在本示例中，当用户点击父列表框中某个选项内容时，将弹出一个新建的页面，页面中显示与父列表相对应的子列表内容。这个动态生成的子列表的默认主题样式为“蓝色”，以区分父列表，表示这是一个二级列表。

当然，这种列表的嵌套可以有很多层，但从视觉效果来说，建议不超过三层。但无论有多少层，jQuery Mobile都会自动处理页面打开与链接的效果。

14.3 jQuery Mobile API接口应用

jQuery Mobile可以完全使用HTML 5和CSS 3特征开发移动项目的页面功能，此外，还为开发者提供了大量实用可供扩展的API接口，通过这些接口提供的方法实现各项复杂的页面功能。

14.3.1 默认配置设置

在jQuery Mobile中，框架的基本配置项是可以修改的。由于配置项针对的是全局功能的使用，jQuery Mobile会在页面加载到增强特征时就需要使用这些配置项，而这个加载过程早于document.ready事件的触发，因此，在该事件中进行修改是无效的，而是选择更早的"mobileinit"事件，在该事件中，可以编写新的配置项来覆盖原有的基本配置项设置。

当用户在移动端浏览jQuery Mobile开发的移动项目中的页面时，如果是首次加载或速度较慢时，会在页面的居中位置显示滚动的加载动画和"Loading"的文字信息。另外，如果访问的某个链接页面不存在时，也会出现"Error Loading Page"的提示信息，而这些默认配置项都可以在document.mobileinit事件中进行自定义设置。

接下来通过一个简单示例来说明默认配置设置的实现过程。

示例14-8　默认配置设置的实现

（1）功能说明

新建一个HTML页面，在页面中增加一个＜a＞元素，将该元素的"href"属性值设置为一个不存在的页面文件"error.htm"，当用户点击该元素时，将显示自定义的出错提示信息。

（2）实现代码

新建一个HTML页面14-8.html，加入代码如代码清单14-8所示。

代码清单 14-8　默认配置设置

```
<!DOCTYPE html>
<html>
<head>
    <title>默认配置设置</title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
```

```
            type="text/javascript"></script>
    <script type="text/javascript">
        $(document).bind("mobileinit", function() {
            $.extend($.mobile, {
                loadingMessage: '努力加载中 ...',
                pageLoadErrorMessage: '找不到对应页面! '
            });
        })
    </script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
            type="text/javascript"></script>
</head>
<body>
  <div data-role="page">
    <div data-role="header">
      <h1>API</h1>
    </div>
    <div data-role="content">
  <h3>修改默认配置值</h3>
  <p><a href="error.html">点击我</a></p>
  </div>
  <div data-role="footer">
        <h4>?2013 rttop.cn studio</h4>
  </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-8所示。

图　14-8　默认配置项设置

（4）代码分析

在本示例中，借助$.mobile对象，在"mobileinit"事件中，通过下列两行代码分别修改了页面加载时和加载出错时的提示信息，代码如下：

```
$.extend($.mobile, {
        loadingMessage: '努力加载中 ...',
        pageLoadErrorMessage: '找不到对应页面! '
})
```

上述代码调用了jQuery中的$.extend()方法进行扩展，也可以使用$.mobile对象直接对各配置值进行设置，因此上述代码等价于：

```
$.mobile.loadingMessage = '努力加载中 ...';
$.mobile.pageLoadErrorMessage = '找不到对应页面! ';
```

在"mobileinit"事件中加入上述代码中的任意一种，都可以实现修改默认配置项"loadingMessage"和"pageLoadErrorMessage"的显示内容。

说明"mobileinit"事件是加载时立刻触发，因此无论是在页面上直接编写JavaScript代码，还是引用JS格式的文件，都必须将它放在引用"jquery.mobile-1.0.1.js"之前，否则代码无效。

14.3.2　方法

在jQuery Mobile中，通过API修改默认配置属性之外，还借助$.mobile对象提供了不少使用简单、容易上手的方法。

接下来通过一个简单示例来说明方法调用的实现过程。

示例14-9　jQuery Mobile中方法的调用

（1）功能说明

新建一个HTML页面，在页面中显示“页面正在跳转中……”的字样，然后通过调用changePage()方法，以"slideup"动画切换效果从当前页跳转到"about.htm"页面。

（2）实现代码

新建一个HTML页面14-9.html，加入代码如代码清单14-9所示。

代码清单 14-9　jQuery Mobile 方法的调用

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Mobile 方法的调用 </title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
    <script type="text/javascript">
        $(function() {
            $.mobile.changePage("about.html",
            { transition: "slideup" });
        })
    </script>
</head>
<body>
  <div data-role="page">
  <div data-role="header"><h1> 跳转页面 </h1></div>
  <div data-role="content">
          <p> 页面正在跳转中 ...</p>
  </div>
  <div data-role="footer">
          <h4>©2013 rttop.cn studio</h4>
  </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-9所示。

图　14-9　jQuery Mobile中方法的调用

（4）代码分析

在本示例中，由于changePage()方法在页面加载时被执行，因此在浏览主页面时，便直接通过调用changePage()方法跳转至目标页"about.html"；使用changePage()方法除可以跳转页面外，还能携带数据传递给跳转的目标页，如下面代码：

```
$.mobile.changePage("login.php",
  { type: "post",
    data: $("form#login").serialize()
  },
   "pop", false, false
)
```

上述代码表示：将ID号为"login"的表单数据进行序列化后，传递给"login.php"页面进行处理。另外，"pop"表示跳转时的页面效果，第一个"false"值表示跳转时的方向，如果为"true"则表示反方向进行跳转，默认值为"false"；第二个"false"值表示完成跳转后，是否更新历史浏览记录，默认值为"true"表示更新。

说明　当指定跳转的目标页面不存在或传递的数据格式不正确时，都会在当前页面出现一个错误信息提示框，几秒钟后自动消失，不影响当前页面的内容显示。

14.3.3　事件

在移动终端设备中有一类事件（如鼠标事件或窗口事件)无法触发，但它们又是客观存在的。在jQuery Mobile中，借助框架的API将这类型的事件扩展为专门用于移动终端设备的事件，开发人员可以使用live()或bind()方法进行绑定。

在jQuery Mobile中，页面是被请求后注入当前的DOM结构中，因此，在jQuery中提及的$(document).ready()事件在jQuery Mobile中不会被重复执行，只有在初始化加载页面时才会被执行一次。而如果想要跟踪不同页面的内容注入当前的DOM结构时，可以通过将页面中的"page"容器绑定"pagecreate"事件，该事件在页面初始化时触发，绝大多数的jQuery Mobile组件都在该事件之后进行一些数据的初始化。

接下来通过一个简单示例来说明事件触发的过程。

示例14-10　jQuery Mobile中事件的触发

（1）功能说明

新建一个HTML页面，在页面中添加一个ID号为"e1"的"page"容器，并将该容器与"pagebeforecreate"和"pagecreate"事件进行绑

定。在页面执行时，通过绑定的事件跟踪执行的过程。

（2）实现代码

新建一个HTML页面14-10.html，加入代码如代码清单14-10所示。

代码清单 14-10　jQuery Mobile 事件的触发

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Mobile 事件的触发 </title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
    <script type="text/javascript">
        $("#e1").live("pagebeforecreate", function() {
            alert(" 正在创建页面! ");
        });
        $("#e1").live("pagecreate", function() {
            alert(" 页面创建完成! ");
        });
    </script>
</head>
<body>
  <div data-role="page" id="e1">
  <div data-role="header"><h1> 创建页面 </h1></div>
  <div data-role="content">
          <p> 页面创建完成! </p>
  </div>
  <div data-role="footer">
          <h4>©2013 rttop.cn studio</h4>
  </div>
  </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-10所示。

图　14-10　jQuery Mobile中事件的触发

（4）代码分析

在本示例中，ID为"e1"的"page"容器绑定了"pagebeforecreate"和"pagecreate"两个事件。"pagebeforecreate"事件早于"pagecreate"事件，即在页面被加载、jQuery Mobile组件开始初始化前触发。通常在这一事件中，可以

添加一些页面加载的动画提示效果，直到"pagecreate"事件触发时，效果结束。

在本示例的JavaScript代码中，可以使用live()方法绑定元素的触发的事件，另外，使用bind()与delegate()方法同样可以为绑定的元素添加指定的事件。

在jQuery Mobile中，其他重要的事件名称与使用方法如表14-1所示。

表 14-1　jQuery Mobile 中重要事件使用说明

事件名称	触发条件	功能描述
pagebeforeload	在加载请求发出前触发	在绑定的回调函数中，可以调用 preventDefault() 方法，表示由该事件来处理 load 事件
pageload	页面加载成功并创建了全部的 DOM 元素后触发	被绑定的回调函数作为一个数据对象，该对象有两个参数，其中第二个参数包含如下信息：url 表示调用地址，absurl 表示绝对地址
pageloadfailed	当页面加载失败时触发	默认情况下，触发该事件后，jQuery Mobile 框架将以页面的形式显示出错信息
pagebeforechange	页面在切换或改变之前触发	在回调函数中包含两个数据对象参数，其中第一个参数 toPage 表示指定内 / 外部的页面绝对 / 相对地址，第二个参数 options 表示使用 changePage() 方法时的配置选项
pagechange	完成 changePage() 方法请求的页面并完成 DOM 元素加载时触发	在触发任何 pageshow 或 pagehide 事件之前，此事件已完成了触发
pagechangefailed	使用 changePage() 方法请求页面失败时触发	回调函数与 pagebeforechange 事件一样，数据对象包含相同的两个参数

（续）

事件名称	触发条件	功能描述
pagebeforecreate	页面在初始化数据之前触发	在触发该事件之前，jQuery Mobile 的默认部件将自动初始化数据，另外，通过绑定 pagebeforecreate 事件，然后返回 false，可以禁止页面中的部件自动操作
pagecreate	页面在初始化数据之后触发	该事件是用户在自定义自己的部件，或增强子部件中标记时最常调用的一个事件
pageinit	页面的数据初始化完成，还没有加载 DOM 元素时触发	在 jQuery Mobile 中，Ajax 会根据导航把每个页面的内容加载到 DOM 中，因此，要在任何新页面中加载并执行脚本，就必须绑定 pageinit 事件，而非 ready 事件
pageremove	试图从 DOM 中删除一个外部页面时触发	该事件的回调函数中可以调用事件对象的 preventDefault() 方法，防止删除的页面被访问
updatelayout	动态显示或隐藏内容的组成部分时触发	该事件以冒泡的形式通知页面中可以需要同时更新的其他组件

14.3.4　页面主题

在jQuery Mobile中，由于每一个页面中的布局和组件都被设计成一个全新的面向对象的CSS框架，使整个站点或应用的视觉风格可以通过这个框架得到统一，被统一后的视觉设计主题我们称之为jQuery Mobile主题样式系统。

组件和页面布局的主题定义是通过使用一套完整的CSS框架来实现的，在这套CSS框架中包括两个重要组成部分：

❑结构：控制元素的在屏幕中显示的位置、填充效果、内外边距等。

❑主题：控制元素的颜色、渐变、字体、圆角、阴影等视觉效果，并包含了多套的色板，每套色板中都定义了列表项、按钮、表单、工具栏、内容块、页面的全部视觉效果。

jQuery Mobile中，CSS框架中的结构和主题是分离的，这样就只需要定义一套结构就可以反复与一套或多套主题配合或混合使用，从而实现页面布局和组件主题多样化的效果。

在jQuery Mobile中，系统自带了5套主题样式，分别用字母"a"，"b"，"c"，"d"，"e"来进行引用，其各主题的使用场景说明如

表14-2所示。

表 14-2　jQuery Mobile 中主题使用场景

主题字母名称	使用场景说明
a	整体色为黑色，是使用级别最高的主题
b	整体色为蓝色，使用级别仅次于 a 级主题
c	整体为基准灰色，系统默认主题
d	整体色为灰白色，用于 c 级备用的主题
e	整体色为金黄色，用于强调、突出性主题

在默认情况下，jQuery Mobile中给头部栏与底部栏的主题是"a"字母，因为"a"字母代表最高的视觉效果，如果需要改变某组件或容器当前的主题，只需要将它的"data-theme"属性值设置成主题对应的样式字母即可。

接下来通过一个简单示例来说明如何选择主题。

示例14-11　jQuery Mobile中选择主题

（1）功能说明

新建一个HTML页面，并在内容区域中创建一个下拉列表框，用于选择系统自带的5种类型主题。当用户通过下拉列表框选择某一主题时，使用"cookie"的方式保存所选择的主题值，并在刷新页面时将内容区域的主题设置成"cookie"所保存的主题值。

（2）实现代码

新建一个HTML页面14-11.html，加入代码如代码清单14-11所示。

代码清单 14-11　jQuery Mobile 中选择主题

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Mobile 选择主题 </title>
    <meta name="viewport" content="width=device-width" />
    <link  href="Css/jquery.mobile-1.3.0.min.css"
           rel="Stylesheet" type="text/css" />
    <script src="Js/jquery-1.8.2.min.js"
           type="text/javascript"></script>
    <script src="Js/jquery.cookie.js"
           type="text/javascript"></script>
    <script src="Js/jquery.mobile-1.3.0.min.js"
           type="text/javascript"></script>
    <script type="text/javascript">
        $(function() {
            var objSelTheme = $("#selTheme");
            objSelTheme.bind("change", function() {
                // 如果选择的值不为空
                if (objSelTheme.val() != "") {
                    // 使用 cookie 保存所选择的主题
                    $.cookie("StrTheme", objSelTheme.val(), {
                        path: "/", expires: 7
                    })
                    // 重新刷新一次页面，运用主题
                    window.location.reload();
                }
            })
        })
        // 如果主题不为空，则运用主题
        if ($.cookie("StrTheme")) {
            $.mobile.page.prototype.options.theme = $.cookie("StrTheme");
        }
    </script>
</head>
<body>
  <div data-role="page">


    <div data-role="header"><h1> 头部栏 </h1></div>
    <div data-role="content">
      <select name="selTheme" id="selTheme" data-native-menu="false">
      <option value=""> 选择主题 </option>
      <option value="a"> 主题 a</option>
      <option value="b"> 主题 b</option>
      <option value="c"> 主题 c</option>
      <option value="d"> 主题 d</option>
      <option value="e"> 主题 e</option>
     </select>
     </div>
     <div data-role="footer">
            <h4>©2013 rttop.cn studio</h4>
    </div>
    </div>
</body>
</html>
```

（3）页面效果

该页面在Opera Mobile Emulator下执行的效果如图14-11所示。

图　14-11　jQuery Mobile中选择主题

（4）代码分析

在本示例中，首先引用了一个cookie插件文件jquery.cookie.js。然后，在下拉列表框的"change"事件中，当用户选择的主题值不为空时，调用插件中的方法，将用户选择的主题值保存至名称为"StrTheme"的cookie变量中。最后，当页面刷新或重新加

载时，如果名为"StrTheme"的cookie变量不为空时，通过"$.mobile.page.prototype.options.theme=$.cookie("StrTheme")"语句，将页面内容区域的主题设置为用户所选择的主题值。

由于使用cookie方式保存页面的主题值，因此，即使关闭浏览器，重新打开时，用户所选择的主题依然有效，除非手动清除cookie值或对应的cookie值到期后自动失效，页面才会自动恢复到默认的主题值。

14.4　本章小结

jQuery Mobile是面向移动终端开发Web App的利器，被越来越多的开发者所喜爱。本章先从jQuery Mobile框架讲起，然后再由浅入深地介绍它的基本组件、API接口，通过一个个精选的完整示例的介绍，使读者逐步了解并掌握这一框架开发页面应用的基础知识，并为使用jQuery Mobile开发Web App项目打下坚实的理论基础。

第15章　jQuery Mobile综合案例开发

本章内容

新闻订阅管理系统

记事本管理

本章小结

随着移动互联网的不断发展，人们上网的习惯也在悄然发生变化，由原先的PC端桌面浏览器逐步向移动终端设备过渡，而开发基于移动终端设备的应用系统已成为各互联网企业的共识，也是时下最热的话题。本章使用jQuery Mobile框架，开发两个移动终端管理系统，一个是新闻订阅管理系统，另一个为记事本管理系统。通过对这两个系统开发过程的详实介绍，带领大家进入一个全新的jQuery Mobile移动开发世界，体验jQuery Mobile高效、强悍的性能和完美、优雅的UI效果。

15.1　新闻订阅管理系统

本节将使用jQuery Mobile框架，开发一个移动终端的新闻订阅管理系统，实现在移动终端自由订阅各类新闻，以及动态浏览的各项功能。

15.1.1　需求分析

在本系统中，实现的需求包括以下几个方面：

1）在进入系统前先浏览封面页，停留3秒后自动进入首页。

2）在首页中显示用户自己订阅的新闻类别，单击某类别进入相应的类别页；用户在首页单击“管理订阅”按钮时，进入订阅管理页。

3）在类别新闻页浏览该类别中的今日图片与列表新闻，单击图片或列表中的某选项时，进入对应的新闻详细页。

4）在订阅管理页中，以列表的方式展示用户自己没有订阅的新闻类别，用户单击列表最右侧的“添加”按钮时，完成订阅功能。

5）在新闻的详细页中显示某条新闻的对应主题、加入时间、来源、正文信息。

15.1.2　界面效果

在本案例中，进入首页之前有一个系统封面页，如图15-1所示，用于声明系统版权或推广产品，3秒后自动跳转到系统首页。

图　15-1　新闻订阅系统封面页的效果

在系统封面页停留3秒之后，便进入系统首页，如图15-2所示。在首页中显示用户已订阅的新闻类别列表与总数量，如果用户需要订阅其他类别新闻，可以单击首页中的“订阅管理”按钮，进入订阅管理页进行更多类别的新闻订阅。

图　15-2　新闻订阅系统首页

在系统首页中，单击“订阅管理”按钮时，进入订阅管理页。该页面中，以列表的方式显示用户还没有订阅的新闻类别信息，页面效果如图15-3所示。用户单击列表右侧的图标，完成订阅该类别新闻的操作。

图　15-3　订阅管理页

无论是在系统首页还是订阅管理页，用户单击新闻类别列表选项时，都将进入类别新闻页。该页面由上下两部分组成，上部分用于显

示该类别的图片新闻，下部分以列表的形式展示该类别的全部新闻标题，单击相应的标题进入新闻的详细，其实现的页面效果如图15-4所示。


头条
阳春三月，踏青赏花好时节
jQuery Mobile 1.3.0 正式版发布
阳春三月，踏青赏花好时节
webapp开发日渐升温
©2013 rttop.cn studio

图　15-4　“头条”类别新闻

在类别新闻页中，单击新闻的图片或标题都将进入新闻详情页。该页面中展示某条新闻的标题、添加时间、来源和新闻正文信息，其实现的页面效果如图15-5所示。


新闻详细页_荣拓移动新闻系统
localhost/Ch15/News/r
Google
头条
jQuery Mobile 1.3.0 正式版发布
2013-5-1 0:00:00 来源：荣拓新闻
jQuery Mobile 发布了正式版，它的目标是在一个统一的 UI 中交付超级 JavaScript 功能，用于跨最流行的智能手机和平板电脑设备。与 jQuery 一样，JQM 是一个在 Internet 上直接托管、免费可用的开源代码。
©2013 rttop.cn studio

图　15-5　新闻详情页

15.1.3　功能实现

针对开发需求，为了实现对应功能，需要创建多个HTML页面，接下来我们逐一进行介绍。

1.新闻订阅系统封面

新建一个名为load的HTML页面，实现系统封面页功能。在该页面中添加一个"page"容器，在容器中添加一个＜div＞和多个＜p＞元素，用于显示系统封面文字和图标信息，并将容器中的标题栏与底部栏设置为悬浮状，其页面代码如代码清单15-1所示。

代码清单 15-1　新闻订阅系统封面 load.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>封面页 _ 荣拓移动新闻系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="load_index" data-theme="c">
    <div data-role="header" data-position="fixed">
      <h4>荣拓新闻</h4>
    </div>
    <p class="border_p01"></p>
    <div class="load">
       <p class="t">一个有新闻故事的移动端平台</p>
       <p><img src="images/rttop_loading.gif" alt="" /></p>
       <p class="l">正在加载数据...</p>
    </div>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
      type="text/javascript" ></script>
</body>
</html>
```

2. 系统首页

新建一个名为index的HTML页面，用于实现系统首页的功能。在页面中添加一个"page"容器，在容器中添加一个带计数器的＜ul＞列表元素，用于显示用户已订阅的各类别新闻总量和列表；同时，增加一个＜a＞元素的按钮，用于单击进入订阅管理页，其页面代码如代码清单15-2所示。

代码清单 15-2　系统首页 index.htm 文件完整代码

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

<title>首页 _ 荣拓移动新闻系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="index_index">
    <div data-role="header" data-position="fixed">
      <h4>荣拓新闻</h4>
    </div>
    <p class="border_p01"></p>
    <ul data-role="listview" data-dividertheme="e"></ul>
    <a href="newsub.htm" data-theme="d" data-mini="true"
       data-role="button" data-icon="plus">订阅管理</a>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
      type="text/javascript" ></script>
</body>
</html>
```

3. 新闻订阅管理页

新建一个名为newsub的HTML页面，用于实现新闻订阅管理的功能。在该页面中添加一个"page"容器，在容器中添加＜ul＞列表，在列表选项中放置两个＜a＞元素。第一个＜a＞元素内显示类别图标、名称和简单描述，单击时进入某类别新闻页；第二个＜a＞元素内显示订阅图标，单击时实现订阅某类别新闻的功能，其页面代码如代码清单15-3所示。

代码清单 15-3　新闻订阅管理页 newsub.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>订阅管理页 _ 荣拓移动新闻系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"


      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="newsub_index">
    <div data-role="header" data-position="fixed">
      <h3>订阅管理</h3>
    </div>
    <p class="border_p01"></p>
    <ul data-role="listview" data-dividertheme="e"></ul>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
      type="text/javascript" ></script>
</body>
</html>
```

4. 类别新闻页

新建一个名为newscate的HTML页面，用于实现类别新闻页的功能，在该页面添加的"page"容器中创建多个＜div＞元素，用于显示图片新闻的图片、标题和标题背景。另外，再添加一个＜ul＞列表，用于显示该类别下的所有新闻标题信息，其页面代码如代码清单15-4所示。

代码清单 15-4　类别新闻页 newscate.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>类别新闻页 _ 荣拓移动新闻系统 </title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="newscate_index">
    <div data-role="header" data-position="fixed"><h4></h4></div>
    <p class="border_p01"></p>
    <div id="news_wrap">
       <div id="news_bg"></div>
       <div id="news_info"></div>
       <div id="news_list"></div>
    </div>
    <ul data-role="listview" data-dividertheme="e"></ul>

    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
      type="text/javascript" ></script>
</body>
</html>
```

5. 新闻详情页

新建一个名为newsdetail的HTML页面，用于实现新闻详情页的功能，在该页面添加的"page"容器中增加一个＜div＞元素。在该元素中，添加一个＜h＞和两个＜p＞元素，分别用于显示新闻的标题、添加时间、来源和正文信息，其页面代码如代码清单15-5所示。

代码清单 15-5　新闻详情页 newsdetail.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>新闻详细页_荣拓移动新闻系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.9.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="detail_index">
    <div data-role="header" data-position="fixed"><h4></h4></div>
    <p class="border_p01"></p>
    <div class="detail">
      <h4 id="news_detail_title"></h4>
      <p id="news_detail_info" class="news_detail_info"></p>
      <p id="news_detail_content"
         class="news_detail_content"></p>
    </div>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.news.js"
      type="text/javascript" ></script>
</body>
</html>
```

6. JavaScript文件

上述全部HTML文件代码清单中，都包含两个JavaScript文件，即rttopHtml5.base.js和rttopHtml5.news.js。

rttopHtml5.base.js文件用于设置系统的一些基础属性值，定义设置与获取localStorage对象键名、键值的方法，该文件如代码清单15-6所示。

代码清单 15-6　rttopHtml5.base.js 文件完整代码

```
var rttophtml5mobi = {
    author: 'tgrong',
    version: '1.0',
    website: 'http://localhost'
}
rttophtml5mobi.utils = {
    setParam: function(name, value) {
        localStorage.setItem(name, value)
    },
    getParam: function(name) {
        return localStorage.getItem(name)
    }
}
```

rttopHtml5.news.js文件通过jQuery Mobile框架实现各页面的对应功能，它包含了本系统中全部页面各功能模块实现的JavaScript代码。该文件如代码清单15-7所示。

代码清单 15-7　rttopHtml5.news.js 文件完整代码

```
//封面页面创建事件
function changepage() {
    window.location.href = "index.htm";
}
$('#load_index').live("pagecreate", function() {
    var id = setInterval("changepage()", 3000);
})
//首页页面创建事件
$('#index_index').live("pagecreate", function() {
    var $li = "";
    var $strSubStr = "";
    var intSubNum = 0;
    var $webSite = rttophtml5mobi.website;
    var $webUrl = $webSite + '/ch15/News/NewsApi.ashx?act=index';
    var $listview = $(this).find('ul[data-role="listview"]');
    var $tpl_Index_List = function($p_array, $p_items) {
      if (rttophtml5mobi.utils.getParam('user_sub_str') != null) {
            $strSubStr = rttophtml5mobi.utils.getParam('user_sub_str');
            var $arrSubStr = new Array();
            $arrSubStr = $strSubStr.split(",");
            intSubNum = $arrSubStr.length - 1;
            for (var i = 0; i < $arrSubStr.length - 1; i++) {
                $.each($p_items.Table, function(index, item) {
                    if (item.news_cateid == $arrSubStr[i]) {
                        $li = '<li class="lst" data-icon="false">
                        <a href="newscate.htm" data-ajax="false"
```

```
                        data-catename="' + item.news_catename + '"
                        data-id="' + item.news_cateid + '"
                        style="margin:0px;padding:0px 0px 0px 55px">
                        <img src="' + item.news_iconurl + '" alt="" />
                        <h3>' + item.news_catename + '</h3></li>';
                        $p_array.push($li);
                    }
                })
            }
        } else {
            $li = '<li style="text-align:center">
                  您还没有订阅任务类型新闻! </li>';
            $p_array.push($li);
        }
    }
    var $lst_Index_List = function() {
        $.getJSON($webUrl, {},
        function(response) {
            var li_array = [];
            $tpl_Index_List(li_array, response);
            var strTitle = '<li data-role="list-divider">
            我的订阅<span class="ui-li-count">' + intSubNum + '</span></li>';
            $listview.html(strTitle + li_array.join(''));
            $listview.listview('refresh');
            $listview.delegate('li a', 'click', function(e) {
                rttophtml5mobi.utils.setParam('cate_link_id',
                $(this).data('id'))
                rttophtml5mobi.utils.setParam('cate_link_name',
                $(this).data('catename'))
            })
        })
    }
    $lst_Index_List();
})
//订阅管理页面创建事件
$('#newsub_index').live("pagecreate", function() {
    var $li = "";
    var $strSubStr = "";
    var $webSite = rttophtml5mobi.website;
    var $webUrl = $webSite + '/ch15/News/NewsApi.ashx?act=index';
    var $listview = $(this).find('ul[data-role="listview"]');
    var $tpl_Sub_List = function($p_array, $p_items) {
    if (rttophtml5mobi.utils.getParam('user_sub_str') != null) {
        $strSubStr = rttophtml5mobi.utils.getParam('user_sub_str');
            $.each($p_items.Table, function(index, item) {
                if ($strSubStr.indexOf(item.news_cateid) == -1) {
                    $li = '<li class="lst" data-icon="false">
                    <a href="newscate.htm" data-ajax="false"
                    data-catename="' + item.news_catename + '"
                    data-id="' + item.news_cateid + '"
                    style="margin:0px;padding:0px 0px 0px 55px">
                    <img src="' + item.news_iconurl + '" alt="" />
                    <h3>' + item.news_catename + '</h3>
```

```
                    <p>' + item.news_catedesc + '</p></a>
                    <a data-id="' + item.news_cateid + '"
                    class="a1" href="javascript:"></a></li>';
                    $p_array.push($li);
                }
            })
        } else {
            $.each($p_items.Table, function(index, item) {
                $li = '<li class="lst" data-icon="false">
                <a href="newscate.htm" data-ajax="false"
                data-catename="' + item.news_catename + '"
                data-id="' + item.news_cateid + '"
                style="margin:0px;padding:0px 0px 0px 55px">
                <img src="' + item.news_iconurl + '" alt="" />
                <h3>' + item.news_catename + '</h3>
                <p>' + item.news_catedesc + '</p></a>
                <a data-id="' + item.news_cateid + '"
                class="a1" href="javascript:"></a></li>';
                $p_array.push($li);
            })
        }
    }
    var $lst_Sub_List = function() {
        $.getJSON($webUrl, {},
        function(response) {
            var li_array = [];
            $tpl_Sub_List(li_array, response);
            var strTitle = '<li data-role="list-divider">
             精品推荐</li>';
            $listview.html(strTitle + li_array.join(''));
            $listview.listview('refresh');
            $listview.delegate('li a', 'click', function(e) {
                rttophtml5mobi.utils.setParam('cate_link_id',
                $(this).data('id'))
                rttophtml5mobi.utils.setParam('cate_link_name',
                $(this).data('catename'))
            })
            $listview.delegate('li .a1', 'click', function(e) {
                $strSubStr += $(this).data('id') + ",";
                rttophtml5mobi.utils.setParam('user_sub_str', $strSubStr);
                window.location.reload();
            })
        })
    }
    $lst_Sub_List();
})
//类别新闻页面创建事件
$('#newscate_index').live("pagecreate", function() {
    var $li = "";
    var $strId = "";
    var $strName = "";
    var $webUrl1 = "";
    var $webUrl2 = "";
    var $webSite = rttophtml5mobi.website;
    var $catename = $(this).find('[data-role="header"] h4');
```

```
var $listview = $(this).find('ul[data-role="listview"]');
var $adlist = $("#news_list");
var $adinfo = $("#news_info");
var $tpl_Cate_Ad = function($p_array, $p_items) {
    $.each($p_items.Table, function(index, item) {
        $li = '<a href="newsdetail.htm" data-ajax="false"
        data-catename="' + item.news_catename + '"
        data-id="' + item.news_id + '">
        <img src="' + item.imgnews_imgurl + '" alt=""/></a>';
        $adinfo.html(item.news_title);
        $p_array.push($li);
    })
}
var $tpl_Cate_List = function($p_array, $p_items) {
    $.each($p_items.Table, function(index, item) {
        $li = '<li class="lst" data-icon="false">
        <a href="newsdetail.htm" data-ajax="false"
        data-catename="' + item.news_catename + '"
        data-id="' + item.news_id + '"
        style="margin:0px;padding:0px">
        <h3>' + item.news_title + '</h3></a></li>';
        $p_array.push($li);
    })
}
var $lst_Cate_Ad = function() {
    $strId = rttophtml5mobi.utils.getParam('cate_link_id');
    $strName = rttophtml5mobi.utils
               .getParam('cate_link_name');
    $webUrl1 = $webSite + '/ch15/News/NewsApi.ashx
               ?act=cate_img&cateid=' + $strId;
    $.getJSON($webUrl1, {},
    function(response) {
        $catename.html($strName);
        var li_array = [];
        $tpl_Cate_Ad(li_array, response);
        $adlist.html(li_array.join(''));
        $adlist.delegate('a', 'click', function(e) {
            rttophtml5mobi.utils.setParam('p_link_id',
            $(this).data('id'));
            rttophtml5mobi.utils.setParam('cate_link_name',
            $(this).data('catename'));
        })
    })
}
var $lst_Cate_List = function() {
    $strId = rttophtml5mobi.utils.getParam('cate_link_id');
    $strName = rttophtml5mobi.utils
               .getParam('cate_link_name');
    $webUrl2 = $webSite + '/ch15/News/NewsApi.ashx
               ?act=cate_lst&cateid=' + $strId;
    $.getJSON($webUrl2, {},
    function(response) {
        var li_array = [];
        $tpl_Cate_List(li_array, response);
        $listview.html(li_array.join(''));
```

```
                $listview.listview('refresh');
                $listview.delegate('li a', 'click', function(e) {
                    rttophtml5mobi.utils.setParam('p_link_id',
                    $(this).data('id'))
                    rttophtml5mobi.utils.setParam('cate_link_name',
                    $(this).data('catename'))
                })
            })
        }
        $lst_Cate_Ad();
        $lst_Cate_List();
    })
    // 新闻详细页面创建事件
    $('#detail_index').live("pagecreate", function() {
        var $strId = "";
        var $strName = "";
        var $webSite = rttophtml5mobi.website;
        var $webUrl = "";
        var $catename = $(this).find('[data-role="header"] h4');
        var $title = $("#news_detail_title");
        var $info = $("#news_detail_info");
        var $content = $("#news_detail_content");
        var $lst_Detail_List = function() {
            $strId = rttophtml5mobi.utils.getParam('p_link_id');
            $strName = rttophtml5mobi.utils.getParam('cate_link_name');
            $webUrl = $webSite + '/ch15/News/NewsApi.ashx
                    ?act=detail&newsid=' + $strId;
            $.getJSON($webUrl, {},
            function(response) {
                $catename.html($strName);
                $.each(response.Table, function(index, item) {
                    $title.html(item.news_title);
                    var strHTML = item.news_addDate + "  来源：" + item.news_source;
                    $info.html(strHTML);
                    $content.html("    " +
                                item.news_content);
                });
            })
        }
        $lst_Detail_List();
    })
```

7. CSS样式文件

本系统只有一个样式文件rttopHtml5.css，用于控制整个系统的页面样式与结构布局，详细代码如代码清单15-8所示。

代码清单 15-8　rttopHtml5.css 文件完整代码

```
.load
{
  text-align:center;padding:10px; line-height:1.8em
}
```

```
.load .t
{
  padding:5px 20px 5px 20px; font-family:黑体;
  color:#e63f38
}
.load .l
{
  color:#666;font-size:12px
}
.border_p01
{
  width:100%; height:3px; background:#e41400;
  margin:0; padding:0
}
/*列表区域*/
.lst
{
  height:100%;margin:0px;
  padding:0px 5px 0px 5px
}
.lst a img
{
  padding:0px;max-height:50px;max-width:50px;
  padding-top:5px;padding-bottom:5px;margin:0px
}
.lst a h3
{
  width:80%
}
/*新闻详细页*/
.detail
{
  text-align:center;padding:8px
}
.detail .news_detail_info
{
  font-size:12px; color:#666;
  padding-bottom:5px; border-bottom:solid 1px #ccc
}
.detail .news_detail_content
{
  text-align:left
}
/*新闻推荐图片*/
#news_wrap
{
  position:relative;width:100%;height:auto;
  min-height:160px;overflow:hidden;
}
#news_list img
{
  border:0px;width:100%;height:auto
}
#news_bg
{
```

```
  position:absolute;width:100%;min-height:30px;
  line-height:30px;bottom:0;background-color:#000;
  filter:Alpha(Opacity=40);opacity:0.4;z-index:1;
  cursor:pointer;
}
#news_info
{
  position:absolute;min-height:30px;line-height:30px;
  bottom:0; left:3px;font-size:12px;
  color:#fff;z-index:1;cursor:pointer
}
/* 列表右侧按钮 */
.ui-li-link-alt
{
  position: absolute; width: 52px; height: 100%;
  border-width: 0;background:url(images/add.png) no-repeat;
  background-position:center; border-left-width: 1px;
  top: 0; right: 0; margin: 0; padding: 0; z-index: 2;
}
.ui-li-link-alt .ui-btn
{
  display:none; overflow: hidden; position: absolute;
  right: 8px; top: 50%; margin: -11px 0 0 0;
  border-bottom-width: 1px; z-index: -1;
}
```

在上述样式文件代码清单中，有3个样式类别需要说明。

❑为了能在列表的＜li＞选项元素中自定义最右侧的单击按钮图标，首先在＜li＞选项元素中添加两个＜a＞元素，并重置"ui-li-link-alt"类别下的"ui-btn"子类别，将"display"属性值设置为"none"，表示隐藏原有按钮元素。

❑然后，再重置"ui-li-link-alt"类别。在该类别中，以背景的方式添加一个自定义的图标，作为单击按钮的图标。

❑最后，由于隐藏了最右侧的原有按钮，所以标题的长度默认为“100%”，该值将会使标题的内容覆盖最右侧的自定按钮图标区域，

因此，将＜h3＞标题的长度修改为“80%”，可以规避这一现象，形成两列独自显示的页面效果。

8.API接口文件

在本系统的JavaScript代码中，调用$.getJSON()方法访问接口文件NewsApi.ashx，获取指定的JSON格式数据。该文件是.NET服务端语言开发的一般程序处理文件，主要功能是：根据传回的参数获取数据库中的对应数据，并转成JSON格式数据集传递给调用的页面。

介于篇幅，该文件的详细代码不在本书中列出，感兴趣的读者可以在本书的源码文件（.../Ch15/News）中获取。

15.1.4　代码分析

接下来按功能模块进行代码说明。

1.新闻订阅系统封面

在rttopHtml5.news.js文件的封面页面创建事件中，为了使页面能在设定的时间内跳转至首页，先创建一个自定义的函数changepage()。在该函数中，通过设置"window"对象的"location.href"路径值，实现当前页面的转向功能；然后，在本页面绑定的"pagecreate"事件中调用setInterval()方法，在该方法中隔3秒将自动执行自定义函数changepage()，从而实现自动页面跳转的功能。

2.系统首页

在rttopHtml5.news.js文件的首页面创建事件中，代码执行时分为三个部分：

1）变量的初始化。在该部分中，先初始化变量值（如"$li"、"$strSubStr"等）供后续代码调用。

2）定义一个名为"$tpl_Index_List"的函数型变量。在该函数中，定义了两个参数"$p_array"和"$p_items"。前者是一个数组型变

量，以追加形式保存格式化后的数据字符串；后者为返回数据对象，该对象保存通过API返回的数据集。

在函数体中，先通过调用自定义方法getParam()获取键名为"user_sub_str"对应的值，该值为用户已订阅新闻的类别ID号，格式为“1，2，3，…”；如果该值不为null，则先使用"split"方法分割该值的内容，并使用for语法遍历分割后的内容。

在遍历过程中，再使用$.each方法遍历返回的全部新闻类别数据集。如果用户已订阅的新闻类别ID号与返回数据集中的ID号相等，则获取该ID号新闻类别的其他信息，以字符串的形式保存在变量"$li"中，并调用数组的push()方法将变量"$li"的内容追加到参数数组"$p_array"中。如果用户已订阅新闻的类别值为null，将一句提示信息的值赋给变量"$li"，并追加到参数数组"$p_array"中。

3）定义一个名为"$lst_Index_List"的函数型变量。在该函数中，先通过$.getJSON方法请求指定的API地址，在该方法的回调用函数中接收返回JSON数据集，并保存在对象变量"response"中。另外，定义一个名为"li_array"的数组变量，将这两个变量作为实参，调用第二部分中的"$tpl_Index_List"函数型变量，使"li_array"数组变量接收函数返回的字符串内容，并通过join()方法处理后，作为＜ul＞列表元素显示的内容；同时，刷新＜ul＞列表元素，使它能即时显示已赋值的数据内容。

最后，为了使用户在单击列表中某选项时实现传递参数的功能，在列表对象"$listview"中调用"delegate"方法绑定＜a＞元素的单击事件。在该事件中，通过调用自定义的"setParam"方法，将类别的ID号与名称保存在相应键名的"localStorage"对象中，实现单击传参的功能。

3.新闻订阅管理页

在rttopHtml5.news.js文件的订阅管理页面创建事件中，代码执行时分为三个部分：

1）定义和初始化变量，供后续代码的调用。

2）定义一个名为"$tpl_Sub_List"函数型变量。在该函数中，同样定义两个形参，一个用于保存返回规格化显示数据的数组，另一个用于存储API请求时返回的数据集。

在函数体中，先通过自定义的getParam()方法获取键名为"user_sub_str"的，用户订阅新闻类别"Id"字符信息，如果该值不为null，则在遍历返回的数据集时，使用"indexOf"方法检测字符信息值中是否存在遍历过程中的新闻类别ID号。如果检测返回的值为“-1”，表示不存在，通过"$li"变量保存新闻类别的其他数据信息，并调用数组中的push()方法将"$li"变量内容追加至形参数组"$p_array"中；如果用户订阅新闻类别"Id"字符信息值为空，则直

接使用$.each()方法遍历返回的新闻类别数据集，并将格式化后的字符串追加至形参数组"$p_array"中。

3）定义另外一个函数型变量"$lst_Sub_List"。在函数中，先使用$.getJSON()方法请求API返回指定的数据集。在该方法的回调函数中，将数组变量"li_array"和对象变量"response"作为第二部分自定义的函数型变量"$tpl_Sub_List"实参调用该函数型变量，将获取的数组使用join()方法处理后，作为＜ul＞列表元素中显示的内容，并刷新该列表组件，使设置好的内容即时显示在页面中。

最后，使用列表元素的"delegate"方法绑定两个单击事件：

❑单击列表中某个选项时，触发的列表单击事件。在该事件中，调用自定义的setParam()方法，将新闻类别ID号和类别名称保存至自己命名的键名中。页面切换后，再调用自定义的getParam()方法获取保存的对应键值，从而实现页面切换时参数的传递。

❑用户单击列表中最右侧添加图标时，触发图标的单击事件。在该事件中，用逗号分割的方式保存用户所选择的新闻类别ID号，并调用自定义的setParam()方法，将该字符串信息保存至键名为"user_sub_str"的"localStorage"对象中。

4.类别新闻页

在rttopHtml5.news.js文件的类别新闻页面创建事件中，代码执行时分为三个部分：

1）定义和初始化变量，供后续代码使用时的调用。

2）需要展示图片与普通新闻两部分数据，因此分开编写两个函数型变量"$tpl_Cate_Ad"和"$tpl_Cate_List"。前者用于遍历API返回的图片新闻数据集，并将格式化后的字符串追加至形参数组中；后者用于遍历API返回的普通新闻数据集，将格式化后的字符串追加至形参数组中。

3）定义两个函数型变量"$lst_Cate_Ad"和"$lst_Cate_List"，分别用于调用第二部分中自定义的"$tpl_Cate_Ad"和"$tpl_Cate_List"函数型变量。在调用前，首先通过getParam()方法接收页面传递的类别ID号和名称参数值，分别保存在变量"$strId"和"$strName"中，并根据变量"$strId"的值，组成$.getJSON()方法请求的URL地址，实现根据类别ID号动态取对应数据集的功能。然后，在$.getJSON()方法的回调函数中，定义一个数组"li_array"和接收返回数据集变量"response"，将这两个变量作为调用"$tpl_Cate_Ad"和"$tpl_Cate_List"函数型变量的实参。最后，将函数调用返回的数组使用join()方法处理后显示在页面相应的元素中。如果是＜ul＞列表，则再执行刷新操作后，被赋值的内容便可即时显示在＜ul＞列表中。

最后，使用"delegate"方法设置图片新闻和列表选项元素的单击事件。在该事件中，使用自定义的"setParam"方法，将新闻的ID号和类别名称分别保存在对应键名的"localStorage"对象中，在切换页面时调用。

5.新闻详情页

在rttopHtml5.news.js文件的新闻详情页面创建事件中：

1）定义和初始化变量，供后续代码的调用。

15.2　记事本管理

上一节通过一个完整的新闻订阅管理系统的开发，详细介绍了在jQuery Mobile中，通过调用服务端API的方式获取并组织JSON格式数据的过程。在移动终端浏览该系统页面时，可能需要即时通过网络与服务端进行数据交互，而移动终端的网络环境受流量和通信信号覆盖的限制，相比PC端而言要复杂得多，因此，系统页面最终执行效率会受影响。

在当前移动终端网络环境不太流畅的情况下，如果针对处理少量数据交互的系统开发，可以借助HTML 5中新增的localStorage对象对数据进行存储与管理。这样无须即时通过网络与服务端发生数据的交互，将极大地加速页面操作响应的速度和用户最终体验。

本节将通过一个完整的记事本管理系统的开发，由浅入深地介绍在jQuery Mobile中使用localStorage对象开发移动项目的方法与技巧。

15.2.1　需求分析

在记事本管理系统中，主要包括如下几个需求：

1）在新手导航页中，以左右滑动图片的方式显示系统截图，用户滑至最后一幅截图时，自动进入首页。

2）进入首页后，以列表的方式展示各类别记事数据的总量信息，单击某类别选项进入该类别的记事列表页。

3）在某类别下的记事列表页中，展示该类别下的全部记事主题内容，并增加根据记事主题进行搜索的功能。

4）单击类别下的某记事主题，进入记事信息详细页，在该页面中展示记事信息的主题和正文信息；另外，添加一个删除该条记事信息的按钮。

5）如果在记事信息的详细页中单击“编辑”按钮，进入记事信息编辑页，在该页中可以编辑主题和正文信息。

6）无论是在首页或类别列表页，单击“新增”按钮时，进入记事信息增加页，在该页中可以增加一条新的记事信息。

15.2.2　界面效果

jQuery Mobile开发的移动项目，既可以在移动设备的浏览器中查看，也可以将页面打包到应用程序中（如APK文件），如果是后者，新手导航页将是系统在运行前需要浏览的页面。在该页面中，以滑动图片的方式引导用户使用本系统的重点功能，效果如图15-6所示。

图　15-6　新手导航页

在新手导航页中浏览全部的导航图片或非首次进入记事本系统时，都将进入系统首页面。在该页面中，通过＜ul＞列表显示记事数

据的全部类别名称，并将各类别记事数据的总量显示在列表中对应类别的右侧，其实现的页面效果如图15-7所示。

图　15-7　记事本管理系统首页

用户在首页单击列表中某类别选项时，将类别名称写入"localStorage"对象的对应键值中，从首页切换至记事类别页时，再根据这个已保存的类别键值与整个"localStorage"对象保存的数据进行匹配，获取该类别键值对应的记事数据，并通过＜ul＞列表将数据内容显示在页面中，其实现的页面效果如图15-8所示。

图　15-8　记事列表默认页和搜索时的效果

用户在记事列表页中，单击某记事主题选项时，将该记事主题的ID号通过"key/value"的方式保存在"localStorage"对象中。

进入详细内容页时，先调出保存的键值作为传递来的记事数据ID号，并将该ID号作为键名获取对应的键值，然后将获取的键值字符串数据转成JSON对象，再将对象的记事主题和内容显示在页面指定的元素中，其实现的页面效果如图15-9所示。

图　15-9　记事详细页和删除数据时的效果

在记事详细内容页中，单击头部栏左侧的“修改”按钮时，进入修改记事内容页。在该页面中，可以修改某条记事数据的类别、主题、内容信息，修改完成后返回记事类别页，其实现的页面效果如图15-10所示。

图　15-10　修改记事数据时的效果

15.2.3　功能实现

针对本案例的开发需求创建多个HTML页面，下面逐一进行介绍。

1.新手导航页

新建一个名为notenav的HTML页面，实现新手导航页的功能。在页面正文"content"容器中添加两个＜div＞元素和一个＜ul＞列表元素。前者用于放置系统的功能图片，后者＜ul＞列表元素放置在图片下面，用于滑动图片时以圆点的方式显示选中或未选图片的状态。左右滑动图片时，图片下方的小圆点也将动态进行变换，如代码清单15-9所示。

代码清单 15-9　记事本管理新手导航页 notenav.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>新手导航 _ 荣拓移动记事本系统 </title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
```

```
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
<style type="text/css">
.ui-page{ background:#414141}
</style>
</head>
<body>
  <div data-role="page" id="notenav_index">
    <div data-role="header"><h4>新手导航</h4></div>
      <div data-role="content">
      <div id="notenav_wrap">
       <div id="notenav_list">
          <a href="javascript:">
             <img src="Images/nav1.jpg" alt="" /></a>
          <a href="javascript:">
             <img src="Images/nav2.jpg" alt="" /></a>
       </div>
       <ul id="notenav_icon"><li></li><li></li></ul>
       </div>
      </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
      type="text/javascript" ></script>
</body>
</html>
```

2.系统首页

新建一个名为index的HTML页面，用于实现系统首页的功能。在页面"page"容器中添加一个＜ul＞列表元素，在列表中显示记事数据的分类名称与类别总量，单击该列表选项时进入记事列表页，如代码清单15-10所示。

代码清单 15-10　记事本管理系统首页 index.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>首页 _荣拓移动记事本系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="index_index">
    <div data-role="header" data-position="fixed"


        data-position="inline">
      <h4>荣拓记事</h4>
      <a href="addnote.htm" class="ui-btn-right">新增</a>
    </div>
    <div data-role="content">
      <ul data-role="listview"></ul>
    </div>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
      type="text/javascript" ></script>
</body>
</html>
```

3. 记事本类别与搜索页

新建一个名为list的HTML页面，用于实现记事本类别与搜索页功能。在页面"page"容器中添加一个＜ul＞列表元素，用于显示某类别下的记事数据。

另外，将＜ul＞列表元素的"data-filter"的属性值设置为"true"，使该列表可以根据记事主题进行搜索，如代码清单15-11所示。

代码清单 15-11　记事本管理记事本类别与搜索页 notenav.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>类别列表页 _ 荣拓移动记事本系统 </title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="list_index">
    <div data-role="header" data-position="fixed"
         data-position="inline">
      <a href="index.htm">返回 </a>
      <h4>记事列表 </h4>
      <a href="addnote.htm">新增 </a>
    </div>
    <div data-role="content">
      <ul data-role="listview" data-filter="true"></ul>
    </div>
    <div data-role="footer" data-position="fixed" >


      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
      type="text/javascript" ></script>
</body>
</html>
```

4. 详细内容页

新建一个名为notedetail的HTML页面，用于实现详细内容页功能。在页面"page"容器的正文区域中，添加一个＜h3＞和两个＜p＞元素，分别用于显示记事信息的主题和内容。单击头部栏左侧的“修改”按钮，进入记事编辑页；单击头部栏右侧的“删除”按钮，可以删除当前的记事数据。如代码清单15-12所示。

代码清单 15-12　记事本详细内容页 notedetail.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>记事详细页_荣拓移动记事本系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="notedetail_index">
    <div data-role="header" data-position="fixed"
         data-position="inline">
      <a href="editnote.htm" data-ajax="false">修改</a>
      <h4></h4>
      <a href="javascript:" id="alink_delete">删除</a></div>
    <div data-role="content">
      <h3 id="title"></h3>
      <p class="notep"></p>
      <p id="content"></p>
    </div>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>
    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
      type="text/javascript" ></script>
</body>
</html>
```

5. 修改记事内容页

新建一个名为editnote的HTML页面，用于实现修改记事内容页功能。在页面"page"容器的正文区域中，通过水平式的单选按钮组显示记事数据的所属类别。一个文本框和一个文本区域框显示记事数据的主题和内容。用户可以重新选择所属类别和编辑主题及内容数据，单击“更新”按钮后，则完成数据的修改操作，并返回记事类别页，如代码清单15-13所示。

代码清单 15-13　记事本管理系统修改记事内容页 editnote.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>修改记事 _ 荣拓移动记事本系统</title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="editnote_index">
    <div data-role="header" data-position="fixed"
        data-position="inline">
      <a href="notedetail.htm" data-ajax="false">返回</a>
      <h4>编辑记事</h4>
      <a href="javascript:">更新</a>
    </div>
    <div data-role="content">
      <label for="rdo-type">类型:</label>
      <fieldset data-role="controlgroup"
                id="rdo-type" data-mini="true"
                data-type="horizontal"
                style="padding:5px 0px 0px 0px; margin:0px">
        <input type="radio" name="rdo-type"
                id="rdo-type-0" value="a" />
        <label for="rdo-type-0" id="lbl-type-0">散文</label>
        <input type="radio" name="rdo-type"
                id="rdo-type-1" value="b"/>
        <label for="rdo-type-1" id="lbl-type-1">随笔</label>
        <input type="hidden" id="hidtype"  value="a"/>
      </fieldset>
      <label for="txt-title">标题:</label>
      <input type="text" name="txt-title"
               id="txt-title" value=""  />
      <label for="txta-content">正文:</label>
      <textarea name="txta-content" id="txta-content"></textarea>
    </div>
    <div data-role="footer" data-position="fixed" >
      <h1>©2013 rttop.cn studio</h1>

    </div>
  </div>
<script src="Js/rttopHtml5.base.js"
      type="text/javascript"></script>
<script src="Js/rttopHtml5.note.js"
      type="text/javascript" ></script>
</body>
</html>
```

6. 添加记事内容页

新建一个名为addnote的HTML页面，用于实现添加记事内容的功能。在页面"page"容器的正文区域中，水平式的单选按钮组用于选择记事类型。一个文本框和一个文本区域框分别用于输入记事主题和内容。用户选择记事数据类型和输入记事数据主题和内容并单击“保存”按钮后，则完成数据的添加操作，将返回记事类别页，如代码清单15-14所示。

代码清单 15-14　记事本管理系统添加记事内容页 addnote.htm 文件完整代码

```
<!DOCTYPE html>
<html>
<head>
<title>增加记事页 _ 采拓移动记事本系统 </title>
<meta name="viewport" content="width=device-width,
      initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" />
<link href="Css/jquery.mobile-1.3.0.min.css"
      rel="stylesheet" type="text/css"/>
<link href="Css/rttopHtml5.css"
      rel="stylesheet" type="text/css"/>
<script src="Js/jquery-1.8.2.min.js"
      type="text/javascript" ></script>
<script src="Js/jquery.mobile-1.3.0.min.js"
      type="text/javascript" ></script>
</head>
<body>
  <div data-role="page" id="addnote_index">
    <div data-role="header" data-position="fixed"
         data-position="inline">
      <a href="javascript:" data-rel="back">返回 </a>
      <h4>增加记事 </h4>
      <a href="javascript:">保存 </a>
    </div>
    <div data-role="content">
      <label for="rdo-type">类型 :</label>
      <fieldset data-role="controlgroup" id="rdo-type"
              data-mini="true" data-type="horizontal"
              style="padding:5px 0px 0px 0px; margin:0px">
        <input type="radio" name="rdo-type"
               id="rdo-type-0" value="a"
                checked="checked" />
        <label for="rdo-type-0">散文 </label>
        <input type="radio" name="rdo-type"
               id="rdo-type-1" value="b"  />
        <label for="rdo-type-1">随笔 </label>
        <input type="hidden" id="hidtype"  value="a"/>
```

```
        </fieldset>
        <label for="txt-title">标题:</label>
        <input type="text" name="txt-title"
               id="txt-title" value="" />
        <label for="txta-content">正文:</label>
        <textarea name="txta-content" id="txta-content"></textarea>
      </div>
      <div data-role="footer" data-position="fixed" >
        <h1>©2013 rttop.cn studio</h1>
      </div>
    </div>
  <script src="Js/rttopHtml5.base.js"
        type="text/javascript"></script>
  <script src="Js/rttopHtml5.note.js"
        type="text/javascript" ></script>
  </body>
  </html>
```

7. JavaScript文件

上述全部HTML文件代码清单中，都包含两个JavaScript文件，rttopHtml5.base.js和rttopHtml5.note.js。

rttopHtml5.base.js文件用于设置系统的一些基础属性值和定义设置与获取localStorage对象键名、键值的方法，该文件如代码清单15-15所示。

代码清单 15-15　rttopHtml5.base.js 文件完整代码

```
var rttophtml5mobi = {
    author: 'tgrong',
    version: '1.0',
    website: 'http://localhost'
}
rttophtml5mobi.utils = {
    setParam: function(name, value) {
        localStorage.setItem(name, value)
    },
    getParam: function(name) {
        return localStorage.getItem(name)
    }
}
```

rttopHtml5.note.js文件通过jQuery Mobile框架实现各页面的对应功能，它包含了本系统中全部页面各功能模块实现的JavaScript代码。该文件如代码清单15-16所示。

代码清单 15-16　rttopHtml5.note.js 文件完整代码

```
// 新手导航页面创建事件
$("#notenav_index").live("pagecreate", function() {
    if (rttophtml5mobi.utils.getParam('bln_look') != null) {
        $.mobile.changePage("index.htm", "slideup");
    } else {
        var $count = $("#notenav_list a").length;
        $("#notenav_list a:not(:first-child)").hide();
        $("#notenav_icon li:first-child").addClass('on').html("1");
```

```
        $("#notenav_list a img").each(function(index) {
            $(this).swipeleft(function() {
                if (index < $count - 1) {
                    var i = index + 1;
                    var s = i + 1;
                    $("#notenav_list a").filter(":visible")
                                        .fadeOut(500).parent()
                                        .children().eq(i)
                                        .fadeIn(1000);
                    $("#notenav_icon li").eq(i).html(s);
                    $("#notenav_icon li").eq(i).toggleClass("on");
                    $("#notenav_icon li").eq(i).siblings()
                                        .removeAttr("class")
                                        .html("");
                    if (s == $count) {
                        rttophtml5mobi.utils.setParam('bln_look', 1);
                        $.mobile.changePage("index.htm","slideup");
                    }
                }
            }).swiperight(function() {
                if (index > 0) {
                    var i = index - 1;
                    var s = i + 1;
                    $("#notenav_list a").filter(":visible")
                                        .fadeOut(500).parent()
                                        .children().eq(i)
                                        .fadeIn(1000);
                    $("#notenav_icon li").eq(i).html(s);
                    $("#notenav_icon li").eq(i).toggleClass("on");
                    $("#notenav_icon li").eq(i).siblings()
                                        .removeAttr("class")
                                        .html("");
                }
            })
        })
    }
})
//增加记事页面创建事件
$("#addnote_index").live("pagecreate", function() {
    var $header = $(this).find('div[data-role="header"]');
    var $rdotype = $("input[type='radio']");
    var $hidtype = $("#hidtype");
    var $txttitle = $("#txt-title");
    var $txtacontent = $("#txta-content");
    $rdotype.bind("change", function() {
        $hidtype.val(this.value);
    });
    $header.delegate('a', 'click', function(e) {
        if ($txttitle.val().length > 0 && $txtacontent.val().length > 0) {
            var strnid = "note_" + RetRndNum(3);
            var notedata = new Object;
            notedata.nid = strnid;
            notedata.type = $hidtype.val();
            notedata.title = $txttitle.val();
            notedata.content = $txtacontent.val();
```

```
            var jsonotedata = JSON.stringify(notedata);
            rttophtml5mobi.utils.setParam(strnid, jsonotedata);
            window.location.href = "list.htm";
        }
    });
    function RetRndNum(n) {
        var strRnd = "";
        for (var intI = 0; intI < n; intI++) {
            strRnd += Math.floor(Math.random() * 10);
        }
        return strRnd;
    }
})
//首页页面创建事件
$("#index_index").live("pagecreate", function() {
    var $listview = $(this).find('ul[data-role="listview"]');
    var $strKey = "";
    var $m = 0, $n = 0;
    var $strHTML = "";
    for (var intI = 0; intI < localStorage.length; intI++) {
        $strKey = localStorage.key(intI);
        if ($strKey.substring(0, 4) == "note") {
            var getData = JSON.parse(rttophtml5mobi.utils.getParam($strKey));
            if (getData.type == "a") {
                $m++;
            }
            if (getData.type == "b") {
                $n++;
            }
        }
    }
    var $sum = parseInt($m) + parseInt($n);
    $strHTML += '<li data-role="list-divider">全部记事本内容
    <span class="ui-li-count">' + $sum + '</span></li>';
    $strHTML += '<li><a href="list.htm" data-ajax="false"
                data-id="a" data-name="散文">散文
                <span class="ui-li-count">' + $m + '</span></li>';
    $strHTML += '<li><a href="list.htm" data-ajax="false"
                data-id="b" data-name="随笔">随笔
                <span class="ui-li-count">' + $n + '</span></li>';
    $listview.html($strHTML);
    $listview.delegate('li a', 'click', function(e) {
        rttophtml5mobi.utils.setParam('link_type', $(this).data('id'))
        rttophtml5mobi.utils.setParam('type_name', $(this).data('name'))
    })
})
//记事列表页面创建事件
$("#list_index").live("pagecreate", function() {
    var $listview = $(this).find('ul[data-role="listview"]');
    var $strKey = "", $strHTML = "", $intSum = 0;
    var $strType = rttophtml5mobi.utils.getParam('link_type');
    var $strName = rttophtml5mobi.utils.getParam('type_name');
```

```
    for (var intI = 0; intI < localStorage.length; intI++) {
        $strKey = localStorage.key(intI);
        if ($strKey.substring(0, 4) == "note") {
            var getData = JSON.parse(rttophtml5mobi.utils.getParam($strKey));
            if (getData.type == $strType) {
                $strHTML += '<li data-icon="false"
                         data-ajax="false"><a href="notedetail.htm"
                         data-id="' + getData.nid + '">' +
                         getData.title + '</a></li>';
                $intSum++;
            }
        }
    }
    var strTitle = '<li data-role="list-divider">' + $strName +
                   '<span class="ui-li-count">' + $intSum + '</span></li>';
    $listview.html(strTitle + $strHTML);
    $listview.delegate('li a', 'click', function(e) {
        rttophtml5mobi.utils.setParam('list_link_id',$(this).data('id'))
    })
})
//记事详细页面创建事件
$("#notedetail_index").live("pagecreate", function() {
    var $type = $(this).find('div[data-role="header"] h4');
    var $strId = rttophtml5mobi.utils.getParam('list_link_id');
    var $titile = $("#title");
    var $content = $("#content");
    var listData = JSON.parse(rttophtml5mobi.utils.getParam($strId));
    var strType = listData.type == "a" ? "散文" : "随笔";
    $type.html(strType);
    $titile.html(listData.title);
    $content.html(listData.content);
    $(this).delegate('#alink_delete', 'click', function(e) {
        var yn = confirm("您真的要删除吗? ");
        if (yn) {
            localStorage.removeItem($strId);
            window.location.href = "list.htm";
        }
    })
})
//修改记事页面创建事件
$("#editnote_index").live("pageshow", function() {
    var $strId = rttophtml5mobi.utils.getParam('list_link_id');
    var $header = $(this).find('div[data-role="header"]');
    var $rdotype = $("input[type='radio']");
    var $hidtype = $("#hidtype");
    var $txttitle = $("#txt-title");
    var $txtacontent = $("#txta-content");
    var editData = JSON.parse(rttophtml5mobi.utils.getParam($strId));
    $hidtype.val(editData.type);
    $txttitle.val(editData.title);
```

```
        $txtacontent.val(editData.content);
        if (editData.type == "a") {
            $("#lbl-type-0").removeClass("ui-radio-off")
                            .addClass("ui-radio-on ui-btn-active");
        } else {
            $("#lbl-type-1").removeClass("ui-radio-off")
                            addClass("ui-radio-on ui-btn-active");
        }
        $rdotype.bind("change", function() {
            $hidtype.val(this.value);
        });
        $header.delegate('a', 'click', function(e) {
            if ($txttitle.val().length > 0 && $txtacontent.val().length > 0) {
                var strnid = $strId;
                var notedata = new Object;
                notedata.nid = strnid;
                notedata.type = $hidtype.val();
                notedata.title = $txttitle.val();
                notedata.content = $txtacontent.val();
                var jsonotedata = JSON.stringify(notedata);
                rttophtml5mobi.utils.setParam(strnid, jsonotedata);
                window.location.href = "list.htm";
            }
        })
    })
```

8.CSS样式文件

本系统只有一个样式文件rttopHtml5.css，用于控制整个系统的页面样式与结构布局，详细代码如代码清单15-17所示。

代码清单 15-17　rttopHtml5.css 文件完整代码

```
#notenav_wrap
 {
   position:relative;width:100%;
   height:auto;min-height:358px;
   overflow:hidden;
 }
 #notenav_wrap ul
 {
   position:absolute;list-style-type:none;
   z-index:2;margin:0;bottom:0px;
   padding:0;left:45%;
 }
 #notenav_wrap ul li
 {
   background:url(images/icons_off.png)
   center no-repeat;width:12px;
   height:12px;float:left;
   margin-right:8px;
 }
 #notenav_wrap ul li.on
 {
   background:url(images/icons_on.png)
   center no-repeat;width:12px;


    height:12px;line-height:12px;
    float:left;margin-right:8px;font-size:10px;
    text-align:center;color:#666; font-family:Arial
 }
 #notenav_list a
 {
   position:absolute;
   margin-left:15px;
 }
 #notenav_list a img
 {
   border:0px;
 }
 #title
 {
   margin:0px;text-align:center
 }
 .notep
 {
   border-bottom:solid 1px #ccc
 }
 .ui-btn-corner-all
 {
   border-radius: .2em;
 }
 .ui-header .ui-btn-inner
 {
   font-size: 12.5px; padding: .35em 6px .3em;
 }
 .ui-btn-inner
 {
   padding: .3em 20px; display: block;
   text-overflow: ellipsis; overflow: hidden;
   white-space: nowrap;
   position: relative; zoom: 1;
 }
```

15.2.4　代码分析

在本案例的rttopHtml5.note.js文件代码中，接下来按功能模块进行代码说明。

1.新手导航页

在rttopHtml5.note.js文件的新手导航页面创建事件中，首先检测键名为"bln_look"的localStorage对象值是否为null。如果不为null，表示不是首次进入本系统，则调用"$.mobile"对象提供的changePage()方法，直接跳转至首页；如果为空，表示是首次进入本系统，那么首先获取＜div＞元素中图片链接元素的总数量，并保存到变量"$count"中。

然后，使用hide()方法隐藏除第一个图片链接元素之外的其他元素，并使用addClass()方法增加＜ul＞列表元素中图片被选中时对应选项的样式和初始值。

最后，遍历＜div＞元素中的全部图片，并在遍历过程中，通过"$(this)"方式获取每个图片元素，绑定该元素的"swipeleft"和"swiperight"事件。在图片元素的"swipeleft"事件中，先判断当前元素的索引号"index"值是否小于图片总量，如果成立，当前索引号自动增加“1”，使图片的索引号指定向下一张，并通

过fadeout()和fadeIn()方法实现当前图片的隐藏和下一张图片的显示。在显示下一张图片时，调用toggleClass()和html()方法切换该图片选中时的样式和数字内容；同时，使用removeAttr()和html()方法移除其他未选中图片的原有样式和数字内容。

注意　图片元素的"swiperight"事件中执行的代码与"swipeleft"事件基本相同，区别在于，在图片元素的"swiperight"事件中，先判断当前元素的索引号"index"值是否大于0，如果成立，当前索引号自动减少“1”，使图片的索引号指定向上一张。

需要说明的是，在移动设备浏览器中，向左滑动图片表示浏览下一张图片触发"swipeleft"事件，向右滑动图片表示浏览上一张图片触发"swiperight"事件。

2.系统首页

在rttopHtml5.note.js文件的系统首页创建事件中，首先定义一些数值和元素变量，供后续代码的使用。由于全部的记事数据都保存在"localStorage"对象中，因此遍历全部的"localStorage"对象，根据键值中前4个字符为"note"的标准，筛选对象中保存的记事数据，并通过JSON.parse()方法，将该数据字符内容转换成JSON格式对象，再

根据该对象的类型值，将不同类型的记事数量进入累加，分别保存在变量"$m"和"$n"中。

最后，组织显示在页面列表＜ul＞元素的内容，并保存在变量"$strHTML"中，调用列表＜ul＞元素的html()方法，将内容赋值于页面列表＜ul＞元素中；同时，使用delegate()方法设置列表选项触发单击事件时需要执行的代码。

注意　由于本系统的数据全部保存在用户本地的"localStorage"对象中，因此，读取数据的速度很快，将字符串内容赋值给列表＜ul＞元素时，已完成样式加载，无须再调用refresh()方法。

3.记事本类别与搜索页

在rttopHtml5.note.js文件的记事列表页面创建事件中，首先定义一些字符和元素对象变量，并通过自定义的函数方法getParam()获取传递的类别字符和名称，分别保存在变量"$strType"和"$strName"中。

然后，遍历整个"localStorage"对象筛选记事数据。在遍历过程中，将记事的字符数据转换成JSON对象，再根据对象的类别与保存的类别变量相比较，如果符合，则将该条记事的ID号和主题信息追加到

字符串变量"$strHTML"中，并通过变量"$intSum"累加该类别下的记事数据总量。

最后，将获取的数字变量"$intSum"放入列表＜ul＞元素的分割项中，并将保存分割项内容的字符变量"strTitle"和保存列表项内容字符变量"$strHTML"进入组合，通过元素的html()方法将组合后的内容赋值给列表元素；同时，使用delegate()方法设置列表选项被单击时执行的代码。

4.详细内容页

在rttopHtml5.note.js文件的记事详细页面创建事件中，首先，定义一些元素对象变量，并通过自定义的函数方法getParam()获取传递的某记事ID号，并保存在变量"$strId"中。

然后，将该变量作为键名，获取对应的键值字符串，并将键值字符串调用JSON.parse()方法转换成JSON对象，在该对象中依次获取记事的主题和内容，显示在页面中的指定元素中。

最后，通过delegate()方法添加单击“删除”按钮后触发“单击”事件时执行的代码。在该事件中，先通过变量"yn"保存confirm()函数返回的true或false值，如果为真，根据记事数据的键名值使用removeItem()方法，删除指定键名的全部对应键值，实现删除记事数据的功能，同时页面返回类别列表页。

5.修改记事内容

在rttopHtml5.note.js文件的修改记事页面创建事件中，首先调用自定义的方法getParam()获取当前修改的记事数据ID号并保存在变量"$strId"中。

然后，将该变量值作为"localStorage"对象的键名，通过该键名获取对应的键值字符串，并将该字符串转换成JSON格式对象。在对象中，通过属性的方式获取记事数据的类别、主题和正文信息，依次显示在页面指定的元素中。

通过水平式的单选按钮组显示记事类型数据时，先将对象的类型值保存在ID号为"hidtype"的隐藏类元素中，再根据该值的内容，使用removeClass()和addClass()方法修改按钮组中单个按钮的样式，使整个按钮组的选中项与记事数据的类型相一致；同时，设置单选按钮组的"change"事件，在该事件中，用于修改原有类型时ID号为"hidtype"的隐藏类元素的值也随之发生变化，以确保记事类型修改后的值可以实时保存。

接下来设置头部栏中右侧“更新”按钮的单击事件。在该事件中，先检测主题文本框和内容区域框的字符长度是否大于0，用于检测主题和内容是否为空。当两者都不为空时，实例化一个新

的"Object"对象，并将记事数据的各信息作为该对象的属性值，保存在该对象中。

最后，调用JSON.stringify()方法将对象转换成JSON格式的文本字符串，使用自定义的setParam()方法将数据写入"localStorage"对象对应键名的键值中，最终实现记事数据更新的功能。

6.添加记事内容页

在rttopHtml5.note.js文件的添加记事页面创建事件中，首先定义一些变量保存页面中的各元素对象，并设置单选按钮组的"change"事件。在该事件中，单选按钮的选项中发生变化时，保存选项值的隐藏型元素值也将随之变化。

然后，使用delegate()添加头部元素右侧“保存”按钮的单击事件。在该事件中，先检测主题文本框和内容文本域的内容是否为空，如果不为空，调用自定义的一个按长度生成随机数的函数，生成一个3位数的随机数字，并与"note_"字符一起组成记事数据的ID号，保存在变量"strnid"中。

15.3　本章小结

本章通过对两个jQuery Mobile案例开发过程的详细介绍，进一步巩固了前阶段所学的理论知识，为读者全面掌握jQuery Mobile框架开发WebApp应用技术提供案例参考，不断强化读者自己动手开发WebApp的能力。

第16章　jQuery综合案例开发

本章内容

切割图片

在线聊天室

本章小结

在实际的项目开发中，经常需要用户对上传后的图片进行修剪，即切割图片。如果使用传统的JavaScript语言，需要编写大量的代码，而通过导入jQuery插件，编写几行代码，就可以轻松实现。本章编写一个简单、高效的聊天室，用于日常的交流与商谈，方便日常工作。通过此案例介绍使用jQuery插件实现图片的快速切割，以及开发一个基于jQuery库、简洁、易操作的聊天室的详细过程。

16.1　切割图片

电脑中的图片，可以通过各种软件工具（如Photoshop）实现图片的修剪，但对于不太熟悉应用软件的用户来说，实现这样的操作还是比较困难。为了解决这个需求，通过导入一款基于jQuery库的图片切割插件jquery.imagecropper.js，用户就可以在页面中将上传的图片进行任意切割。

16.1.1　需求分析

用户对图片的切割有如下需求：

1）对图片的任意部位可以进行随意的切割。

2）对图片切割时，按不同比例对所选择的图片切割区域进行预览。

3）单击“确定”按钮后，将所选区域从原图片中切割下来，另存为一张图片。

4）如果没有选择切割区域，而单击“确定”按钮进行切割时，将提示错误信息。

16.1.2　界面效果

为实现上述用户需求，在HTML页面ImgCropper.html中导入一款基本jQuery库的图片切割插件jquery.imagecropper.js，并加载一张用于切割的图片，该页面的初始状态如图16-1所示。

图　16-1　图片切割页的初始状态

用户在原图片中移动鼠标时，将出现一个边框与四个角都有小正方框的可拖动的块区。用户根据需求，拖动每一个小正方框，以实现

改变所选区域大小的功能。同时，右边的三个小图片将根据不同的尺寸预览所切割区的图片效果，其实现的页面如图16-2所示。

图　16-2　选择切割区后的页面效果

用户经过不断调整切割区尺寸，最终确定将要切割的图片后，单击图片下方的“确定”按钮，一张按指定尺寸的切割图片就显示在页面中，其实现的页面效果如图16-3所示。

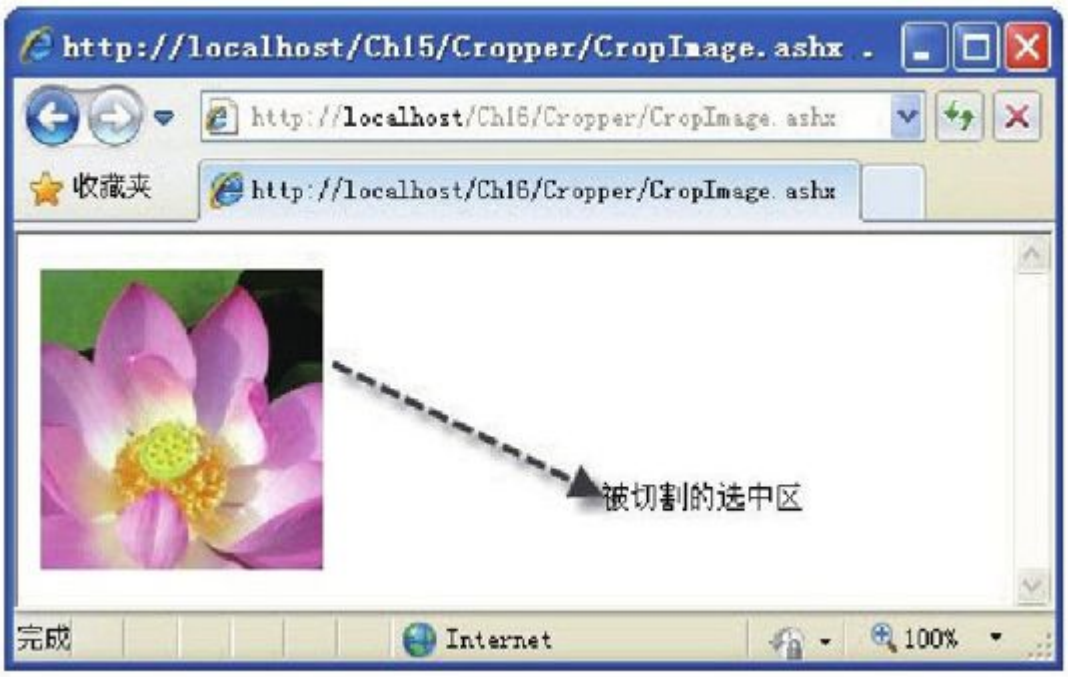


图　16-3　最终切割的图片

16.1.3　功能实现

为了实现对图片切割功能，首先需要导入一个jQuery插件jquery.imagecropper.js，然后调用该插件中的一个Cropper方法设置相关的属性值。单击“确定”按钮后，获取被选择区域的坐标与长宽，并提交给数据处理程序CropImage.ashx，由该页面程序根据获取的相关数据，生成一张被切割的图片，其实现的完整过程如下。

新建一个HTML页面ImgCropper.html，加入的代码如代码清单16-1所示。

代码清单 16-1　页面中实现任意大小的图片切割功能

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>应用案例(1)_图片切割</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery.imagecropper.js">
    </script>
    <script type="text/javascript">
        $(function() {
            $('#cropbox').Cropper({              //调用切割方法
                aspectRatio: 1,                  //设置所选区域的长宽比例
                onChange: showPreview,           //设置长宽变化后触发的事件
                onSelect: showPreview            //设置选择区域后触发的事件
            })
        });
        //自定义提交切割数据时的检测函数
        function checkCoords() {
            if (parseInt($('#w').val())) return true;
```

```
            alert('请选择一个区域后，点击确定按钮！');
            return false;
        }
        // 根据预览图片的大小调用图片预览函数
        function showPreview(coords) {
            if (parseInt(coords.w) > 0) {
                imgPrev(coords, "#img100", 100);
                imgPrev(coords, "#img80", 80);
                imgPrev(coords, "#img60", 60);
            }
            // 获取所选择区域的坐标与宽高
            $('#x').val(coords.x);
            $('#y').val(coords.y);
            $('#w').val(coords.w);
            $('#h').val(coords.h);
        }
        // 自定义所选择区域的图片预览函数
        function imgPrev(coords,obj, w) {
            var rx = w / coords.w;
            var ry = w / coords.h;
            $(obj).css({ // 预览图片
                width: Math.round(rx * 504) + 'px',
                height: Math.round(ry * 378) + 'px',
                marginLeft: '-' + Math.round(rx * coords.x) + 'px',
                marginTop: '-' + Math.round(ry * coords.y) + 'px'
            });
        }
    </script>
    <link rel="stylesheet" type="text/css"
          href="Css/ImgCropper.css"/>
</head>
<body>
    <div id="outer">
    <div class="jcExample">
     <div class="article">
     <h2>图片切割示例</h2>
     <img src="Images/imgDemo.jpg" id="cropbox" />
     <form action="CropImage.ashx" method="post"
           onsubmit="return checkCoords();">
           <input type="hidden" id="x" name="x" />
           <input type="hidden" id="y" name="y" />
           <input type="hidden" id="w" name="w" />
           <input type="hidden" id="h" name="h" />
           <input type="hidden" id="p" name="p"
           value="Images/imgDemo.jpg" />
           <input type="hidden" id="l" name="l" value="0.9"/>
           <input type="submit" value=" 确定 " class="btn" />
           </form>
       </div>
       <div class="clsPre">
            <div class="img100"><img src="Images/imgDemo.jpg"
                                     class="imgPre" id="img100"/>
                 </div><div class="clsAlt">100*100 像素</div>
            <div class="img80"><img src="Images/imgDemo.jpg"
                                    class="imgPre" id="img80"/>
                 </div><div class="clsAlt">80*80 像素</div>
            <div class="img60"><img src="Images/imgDemo.jpg"


                                  class="imgPre" id="img60"/>
             </div><div class="clsAlt">60*60 像素</div>
       </div>
    </div>
    </div>
</body>
</html>
```

注意　为了更好地展示页面中图片切割的效果，尽量使用插件自带的CSS样式文件，如果有其他的补充样式，可以在该文件中进行修改。

在HTML页面ImgCropper.html中，包含一个CSS文件ImgCropper.css，该样式文件中包含插件自带和新增的样式，其完整代码如代码清单16-2所示。

代码清单 16-2　页面 ImgCropper.html 所包含的 CSS 样式

```
body
{
   font-size:11px;
   margin: 0;
   padding: 0;
}
.jcropper-holder
{
   border: 1px #666 solid;
}
#outer
{
   text-align: center;
}
.jcExample
{
   text-align: left;
   background: #eee;
   width: 680px;
   font-size: 80%;
   margin: 3.5em auto 2em auto;
   margin: 3.5em 10% 2em 10%;
   border: 1px black solid;
   padding: 1em 2em 2em;
}
.jcExample .article
{
   width: 504px;
   float:left
}
.jcExample .clsPre
{
   float:right;
   width:100px
}
.jcExample .clsPre .img100
{
```

```
   width:100px;
   height:100px;
   overflow:hidden;
   margin-top:46px;
   margin-bottom:5px
 }
 .jcExample .clsPre .img80
{
   width:80px;
   height:80px;
   overflow:hidden;
   margin-top:5px;
   margin-bottom:5px
 }
 .jcExample .clsPre .img60
{
   width:60px;
   height:60px;
   overflow:hidden;
   margin-top:5px;
   margin-bottom:5px
 }
 .jcExample .clsPre .imgPre
{
   padding:3px;
 }
 .jcExample .clsPre .clsAlt
{
   font-size:9px;
   font-family:Arial
 }
form
{
   margin: 1.5em 0;
}
form label
{
   margin-right: 1em;
   font-weight: bold;
   color: #990000;
}

.jcrop-holder
{
   text-align: left;
}
.jcrop-vline, .jcrop-hline
{
   font-size: 0;
   position: absolute;
   background: white url('Jcrop.gif') top left repeat;
}
.jcrop-vline
{
   height: 100%;
```

```
    width: 1px !important;
}
.jcrop-hline
{
    width: 100%;
    height: 1px !important;
}
.jcrop-handle
{
    font-size: 1px;
    width: 7px !important;
    height: 7px !important;
    border: 1px #eee solid;
    background-color: #333;
    width: 9px;
    height: 9px;
}
.jcrop-tracker
{
    width: 100%; height: 100%;
}
.custom .jcrop-vline,
.custom .jcrop-hline
{
    background: yellow;
}
.custom .jcrop-handle
{
    border-color: black;
    background-color: #C7BB00;
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
}
.btn
{
    border:#666 1px solid;
    padding:2px;width:80px;
    filter: progid:DXImageTransform.Microsoft
       .Gradient(GradientType=0,StartColorStr=#ffffff, EndColorStr=#ECE9D8);
}
```

在前端HTML页面中，通过jQuery插件选择图片中的某块被切割的区域，选定后，必须将确定的图片数据信息提交给服务器页面，由该页面根据传回的数据信息进行图片的真正切割。处理数据的服务器脚本很多，本示例由.NET中的CropImage.ashx文件处理图片切割，该文件的完整代码如代码清单16-3所示。

代码清单 16-3　处理图片切割的文件 CropImage.ashx

```
<%@ WebHandler Language="c#" Class="CropImage" Debug="true" %>
using System;
using System.Web;
using System.Drawing;
using System.IO;


public class CropImage : IHttpHandler
{
 public void ProcessRequest(HttpContext context)
 {
   // 获取与所选择图片区域相关的属性
   string imgPath = Convert.ToString(context.Request["p"]); // 路径
   float imgPacity = Convert.ToSingle(context.Request["l"]); // 透明度
   int top = Convert.ToInt32(context.Request["y"]);          //y 坐标
   int left = Convert.ToInt32(context.Request["x"]);         //x 坐标
   int width = Convert.ToInt32(context.Request["w"]);        // 长度
   int height = Convert.ToInt32(context.Request["h"]);       // 高度
   context.Response.ContentType = "image/jpeg";              // 指定页面输出格式
   imgCrop(HttpContext.Current.Server.MapPath(imgPath), imgPacity,
            top, left, width, height)
            .WriteTo(context.Response.OutputStream);
 }
 /// <summary>
 /// 根据图片的坐标与长宽切割相应图片
 /// </summary>
 /// <param name="imgPath"></param>
 /// <param name="zoomLevel"></param>
 /// <param name="top"></param>
 /// <param name="left"></param>
 /// <param name="width"></param>
 /// <param name="height"></param>
 /// <returns></returns>
 public MemoryStream imgCrop(string imgPath, float imgPacity,
                     int top, int left, int width, int height)
 {
     Image img = Image.FromFile(imgPath);
     Bitmap bitmap = new Bitmap(width, height);
     Graphics g = Graphics.FromImage(bitmap);
     g.DrawImage(img, new Rectangle(0, 0, width, height),
     new Rectangle((int)(left / imgPacity),
         (int)(top / imgPacity), (int)(width / imgPacity),
         (int)(height / imgPacity)), GraphicsUnit.Pixel);
     MemoryStream ms = new MemoryStream();
     bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
     img.Dispose();
     g.Dispose();
     bitmap.Dispose();
     return ms;
 }
 public bool IsReusable
 {
     get
     {
         return false;
     }
 }
}
```

16.1.4　代码分析

在本案例中，为了实现图片的切割功能，首先导入jQuery插件jquery.imagecropper.js，其实现的代码如下：

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

然后，锁定被切割的图片，通过Cropper方法将图片与插件进行绑定，就可以实现任意切割图片的功能，其实现的代码如下所示：

```
$('#cropbox').Cropper({     // 调用切割方法
    aspectRatio: 1,         // 设置所选区域的长宽比例
    onChange: showPreview,  // 设置长宽变化后触发的事件
    onSelect: showPreview   // 设置选择区域后触发的事件
    })
});
```

在这里，需要对imagecropper插件进行简单的介绍。该插件是一款基于jQuery的图片切割插件,可以实现图片在线切割，达到和图像软件处理同样的效果。操作界面十分人性化，只要简单拖曳鼠标就可以轻松实现，与后台脚本的链接十分方便，该插件的调用格式如下所示：

```
$(图片元素).Cropper(options);
```

其中，选项options是一个对象(object)，通过该对象接收的参数，获取或设置相关的图片切割时的属性值。对象options可接收的常用参数如表16-1所示。

表 16-1　选项 options 中的常用参数

参数名称	功能描述
aspectRatio	选中区域的长宽比例，格式为宽 / 高，如果为 1，表示选中区域为正方形
minSize	设置最小的宽与高的值
maxSize	设置最大的宽与高的值
setSelect	页面初始化时，设置所选中的区域，如 [0,0,100,100] 分别表示 x 轴、y 轴、宽与高
bgColor	设置所选中区域的背景色
bgOpacity	设置所选中区域的透明度
allowResize	是否允许改变选中区域的大小，可以设置 true 或 false
allowMove	是否允许拖动选中区域，可以设置 true 或 false

在本案例中，除了可以实现在线切割图片外，还可以对所切割的图片进行预览，为了实现这一个功能，需要自定义一个图片预览函数imgPrev()。在该函数中，根据所选区域和预览图片的宽与高，计算出相对的x轴与y轴的值，根据这两个值，设置各类预览图片的width和height值及marginTop和marginLeft值，其实现的代码如下所示：

```
// 自定义所选择区域的图片预览函数
function imgPrev(coords,obj, w) {
  var rx = w / coords.w;
  var ry = w / coords.h;
  $(obj).css({ //预览图片
       width: Math.round(rx * 504) + 'px',
       height: Math.round(ry * 378) + 'px',
       marginLeft: '-' + Math.round(rx * coords.x) + 'px',
       marginTop: '-' + Math.round(ry * coords.y) + 'px'
    });
}
```

函数imgPrev()中，参数coords表示选中区，参数obj表示预览图片对象，w表示预览图片的长与宽。在本案例中，由于设置的是正方形选中区，因此相应的预览图片的长与宽为一个值，即参数w，数值“504”与“378”分别为整张被切割图片的宽度与高度。

接下来，在自定义的函数showPreview()中，先判断选中区的宽度是否大于零，如果为true，分别用三种不同大小的图片数据调用图片预览函数imgPrev()，其实现的代码如下：

```
//根据预览图片的大小调用图片预览函数
function showPreview(coords) {
  if (parseInt(coords.w) > 0) {
        imgPrev(coords, "#img100", 100);
        imgPrev(coords, "#img80", 80);
        imgPrev(coords, "#img60", 60);
  }
  ...省略部分代码
}
```

同时，函数showPreview()与插件的onChange和onSelect事件绑定，从而实现改变所选区域时，各种不同大小的预览图片也一起发生变化的效果。

另外，函数showPreview()中，通过页面中的隐藏类型元素，获取所选区域的长、高、x轴、y轴的值，在页面提交时，传递给服务器的脚本页面，其实现的代码如下所示：

```
// 获取所选择区域的坐标与宽高
$('#x').val(coords.x);
$('#y').val(coords.y);
$('#w').val(coords.w);
$('#h').val(coords.h);
```

最后，在切割图片时，为了防止提交无效的选中区域，在页面提交时，先利用自定义函数checkCoords()检测长度是否大于零，否则将出现提示信息，其函数的代码如下：

```
// 自定义提交切割数据时的检测函数
function checkCoords() {
  if (parseInt($('#w').val())) return true;
      alert('请选择一个区域后，点击确定按钮！');
      return false;
}
```

16.2　在线聊天室

虽然当前各类的交流工具十分方便快捷，但不少企业或公司还是希望开发一个简单、高效、便于公司内部交流的聊天室。下面详细介绍一个基于jQuery库的小型聊天室开发全过程。

16.2.1　需求分析

本案例的聊天室满足下列需求：

1）用户需要登录才能进入聊天室交流。

2）以无刷新的方式动态显示交流的内容。

3）以无刷新的方式动态显示在线人员基本信息。

4）登录的用户可以提交文字与表情图标，并即时显示在内容区。

16.2.2　界面效果

为了显示当前在线人员的信息，进入聊天室时必须先进行登录，保存登录用户的基本信息，登录页面实现的界面如图16-4所示。

图　16-4　用户登录聊天室

用户在登录聊天室时，调用jQuery中Ajax的全局函数$.ajax()，将获取的用户名与密码数据向数据器发送请求，并使用全局的

ajaxStart()与ajaxStop()事件绑定提示信息元素，使用户在登录时显示“正在发送登录请求”的字样，优化用户页面体验。

用户在登录时，向服务器发送登录数据请求后，服务器端的程序将接收所请求的数据，检测密码是否为“123456”，如果为假，则弹出“用户名或密码错误！”的对话框，否则进入聊天室的主窗口页面。主窗口的界面如图16-5所示。

图　16-5　首次登录聊天主窗口页面

首次登录聊天主页面时，左边的聊天内容区数据为空，右边显示登录时输入的“用户名”。在底部，可以在文本框中输入聊天内容，单击“发送”按钮后，将通过jQuery中Ajax的全局方法$.ajax()获取聊天内容，并向服务器提交请求。同时，服务器响应数据请求，写入完成后，将数据返回至主窗口页面，显示在内容区中。

聊天室主要的功能是实现多人在线聊天，另外，在发送内容时，还可以发送表情图标，以丰富发送内容。多人在线并发送表情图标显示在内容区中时的界面如图16-6所示。

图　16-6　显示多人在线并发送表情图标

16.2.3　功能实现

本案例的功能操作流程为：用户在登录页面Login.html中输入用户名和密码，通过jQuery中Ajax的$.ajax()方法向服务器发送请求。服务器将接收数据与服务器中的数据进行匹配，如果符合，返回true，否则返回false。客户端接收到true后，进入聊天室页面，否则弹出“用户名或密码错误！”的窗口，提示用户登录失败。

用户登录成功时，进入聊天室页面ChatMain.html。在该页面的加载事件中，定时执行两个自定义function函数GetMessageList()与GetOnLineList()，分别用于随时获取最新的聊天内容与在线人员数据信息。

用户在聊天室页面ChatMain.html底部的文本框中输入内容，单击“发送”按钮后，将调用另一个自定义的function函数SendContent()。该函数携带内容值向服务器发出请求，服务器将接收的内容数据写入聊天内容列表。同时，向客户端返回true值，客户端接受该返回值后，调用自定义的GetMessageList()函数，即时显示新写入的全部聊天内容。本案例的操作流程如图16-7所示。

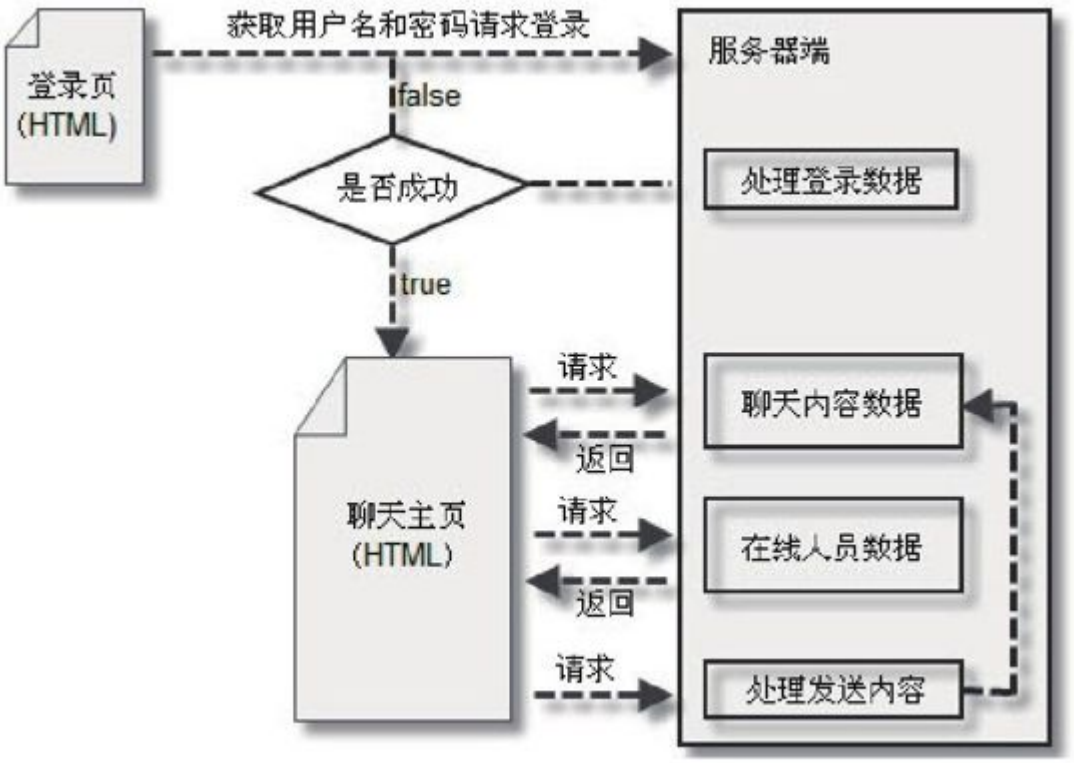


图　16-7　在线聊天室操作流程图

在案例中，用户要进入聊天室，首先进行登录验证。为实现该功能，新建一个HTML页面Login.html，加入如代码清单16-4所示的代码。

代码清单 16-4　用户登录页面 Login.html 的代码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>应用案例 (2)_用户登录_聊天室系统</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css/cssLogin.css" />
    <script type="text/javascript"
            src="Js/jsLogin.js">
    </script>
</head>
<body>
<div id="divLogin">
  <fieldset>
     <h3>用户登录</h3>
     <div class="content">
          <div>用户: <input id="txtName" type="text" class="txt" /></div>
          <div>密码: <input id="txtPass" type="password" class="txt"/></div>
          <div class="btnCenter">

             <input id="Button1" type="button" value="登录"
                    class="btn" />  
             <input id="Reset1" type="reset" value="取消" class="btn" />
          </div>
          <span id="divMsg" class="clsTip">正在发送登录请求…</span>
     </div>
  </fieldset>
</div>
</body>
</html>
```

在登录页面Login.html中，为了便于以后代码的维护与重复使用，新建一个独立的JS文件jsLogin.js，在该文件中编写处理用户登录时的代码，该文件的完整代码如代码清单16-5所示。

代码清单 16-5　jsLogin.js 文件的完整代码

```
$(function() {
    //元素绑定全局ajaxStart事件
    $("#divMsg").ajaxStart(function() {
        $(this).show(); //显示元素
    })
    //元素绑定全局ajaxStop事件
    $("#divMsg").ajaxStop(function() {
        $(this).html("请求处理已完成。").hide();
    })
    $("#Button1").click(function() { //按钮点击事件
        var $name = $("#txtName");    //用户名
        var $pass = $("#txtPass");    //密码
        if ($name.val() != "" && $pass.val() != "") {
            UserLogin($name.val(), $pass.val());
        }
        else {
            if ($name.val() == "") {
                alert("用户名不能为空!");
                $name.focus();
                return false;
            } else {
                alert("密码不能为空!");
                $pass.focus();
                return false;
            }
        }
    })
});

function UserLogin(name, pass) {
    $.ajax({
        type: "POST",
        url: "DealData.aspx",
        data: "action=Login&d=" + new Date() + "&name=" + name + "&pass=" + pass,
        success: function(data) {
            if (data == "True") {


                window.location = "ChatMain.html";
            }
            else {
                alert("用户名或密码错误!");
                return false;
            }
        }
    });
}
```

登录页面Login.html中包含一个CSS文件cssLogin.css，用于控制登录页面的整体样式。该文件的代码如代码清单16-6所示。

代码清单 16-6　cssLogin.css 文件的完整代码

```
body
{
   font-size:11px
}
#divLogin
{
   margin:15% 0 0 15%
}
#divLogin fieldset
{
   width:300px;
   padding:0px;
   margin:0px
}
#divLogin fieldset h3
{
   margin:0px;
   padding:5px;
   background-color:#eee
}
 #divLogin fieldset .content
 {
   padding:20px;
   line-height:2.6em
 }
#divLogin fieldset .content .btnCenter
{
   text-align:center
}
/*侦察请求状态样式*/
.clsTip
{
    position:absolute;
    width:160px;
    text-align:center;
    font-size:13px;
    border:solid 1px #cc3300;
    margin-top:5px;


    padding:2px;
    margin-bottom:5px;
    background-color:#ffe0a3;
    display:none;
 }
.txt
{
   border:#666 1px solid;
    padding:2px;width:180px;
    margin-right:3px
 }
.btn
{
   border:#666 1px solid;
   padding:2px;
   width:60px;
     filter: progid:DXImageTransform.Microsoft
     .Gradient(GradientType=0,StartColorStr=#ffffff,
     EndColorStr=#ECE9D8)
}
```

用户成功登录后，进入聊天室主页面。为实现在线多用户聊天的功能，新建一个HTML页面ChatMain.html，加入如代码清单16-7所示的代码。

代码清单 16-7　聊天室主页面 ChatMain.html 完整代码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <title>应用案例 (2)_聊天室主窗口_聊天室系统</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.8.2.min.js">
    </script>
    <link rel="stylesheet" type="text/css"
          href="Css/cssMain.css" />
    <script type="text/javascript"
            src="Js/jsMain.js">
    </script>
</head>
<body>
<div id="divMain">
     <div class="divtop">
         <div class="divL">
           <h3>荣拓聊天室</h3>
           <div class="divShow" id="divContent"></div>
         </div>
         <div class="divR">
           <h3>当前在线人员</h3>
           <div class="divShow" id="divOnLine"></div>
         </div>
     </div>
     <div class="divBot">
         <table cellpadding="0" cellspacing="0">
             <tr><td colspan="2" id="divFace" class="pb"></td></tr>
             <tr><td>


           <textarea id="txtContent" cols="64" rows="3"
                     class="txt"></textarea></td><td class="pl">
           <input id="Button1" type="button" value="发送" class="btn" />
       </td></tr><tr><td colspan="2" class="pt">
                          发送内容不能为空</td></tr></table>
    </div>
    <span id="divMsg" class="clsTip">正在发送数据…</span>
</div>
</body>
</html>
```

在聊天室主页面ChatMain.html中，包含了一个JS文件jsMain.js，用于处理在主页面中的各种JS代码操作，该文件的完整代码见下列代码清单16-8所示。

代码清单 16-8　jsMain.js 文件的完整代码

```
/// <reference path="../Jscript/jquery-1.8.2.min.js"/>
$(function() {
    //元素绑定全局 ajaxStart 事件
    $("#divMsg").ajaxStart(function() {
        $(this).show(); //显示元素
    })
    //元素绑定全局 ajaxStop 事件
    $("#divMsg").ajaxStop(function() {
        $(this).html("已完成").hide();
    })
    InitFace();
    GetMessageList();
    GetOnLineList();
    $("#Button1").click(function() { //按钮点击事件
        var $content = $("#txtContent"); //发送内容
        if ($content.val() != "") {
            SendContent($content.val());
        }
        else {
            alert("发送不能为空!");
            $content.focus();
            return false;
        }
    });
    $("table tr td img").click(function() { //表情图标单击事件
        var strContent = $("#txtContent").val() + "<:" + this.id + ":>";
        $("#txtContent").val(strContent);

    })
});
//*****************************************
// 自定义发送聊天内容函数
// 参数 content 为聊天内容
//*****************************************
function SendContent(content) {
    $.ajax({
        type: "POST",
```

```
            url: "DealData.aspx",
            data: "action=SendContent&d=" + new Date() + "&content="+ content,
            success: function(data) {
                if (data == "True") {
                    GetMessageList();
                    $("#txtContent").val("");
                }
                else {
                    alert("发送失败!");
                    return false;
                }
            }
        });
    }
    //*****************************************
    //自定义返回聊天内容函数
    //参数 data 为返回的聊天内容数据
    //*****************************************
    function GetMessageList() {
        $.ajax({
            type: "POST",
            url: "DealData.aspx",
            data: "action=ChatList&d=" + new Date(),
            success: function(data) {
                $("#divContent").html(data);
            }
        });
        AutoUpdContent(); //执行定时获取函数
    }
    //*****************************************
    //自定义返回在线人员函数
    //参数 data 为返回的在线人员数据
    //*****************************************
    function GetOnLineList() {
        $.ajax({
            type: "POST",
            url: "DealData.aspx",
            data: "action=OnLineList&d=" + new Date(),
            success: function(data) {
                $("#divOnLine").html(data);
            }
        });
    }
    //*****************************************
    //自定义设置表情图标函数
    //无参数
    //*****************************************
    function InitFace() {
        var strHTML = "";
        for (var i = 1; i <= 10; i++) {
            strHTML += "<img src='Face/" + i + ".gif ' id='" + i + "'/>";
        }
        $("#divFace").html(strHTML);
    }


//*****************************************
//自定义定时执行返回聊天内容与在线人员函数
//无参数
//*****************************************
function AutoUpdContent() {
    setTimeout(GetMessageList, 5000);
    setTimeout(GetOnLineList, 6000);
}
```

聊天室主页面ChatMain.html中，包含一个用于控制页面布局的CSS文件cssMain.css，该文件用于设置页面的框架结构与元素样式，其完整的代码如代码清单16-9所示。

代码清单 16-9　聊天室主页面 cssMain.css 文件的完整代码

```
body
{
    font-size:11px
}
h3
{
    margin:0px
}
.divShow
{
    border:solid 1px #ccc;
    height:300px;
    padding:5px;
    font-size:12px;
    overflow-y:scroll
}
#divMain
{
    border:solid 5px #ccc
}
#divMain .divtop
{
    padding:10px
}
#divMain .divtop .divL
{
    float:left;
    width:78%
}
#divMain .divtop .divR
{
    float:right;
    width:20%
}
#divMain .divBot
{
    clear:both;
    padding:10px
}
#divMain .divBot .pb
{
```

```
    padding-bottom:3px
}
#divMain .divBot .pl
{
    padding-left:12px
}
#divMain .divBot .pt
{
    padding-top:3px;
    color:#555
}
/* 侦察请求状态样式 */
.clsTip
{
    position:absolute;
     width:160px;
     text-align:center;
     font-size:13px;
     border:solid 1px #cc3300;
     margin-top:5px;
     padding:2px;
     margin-bottom:5px;
     background-color:#ffe0a3;
     display:none;
 }
.txt
{
    border:#666 1px solid;
     padding:2px;
     margin-right:3px
 }
.btn
{
    border:#666 1px solid;
    padding:2px;
    width:135px;
    height:54px;
    font-size:16px;
   filter: progid:DXImageTransform.Microsoft
  .Gradient(GradientType=0,StartColorStr=#ffffff,
   EndColorStr=#ECE9D8)
}
```

为了响应客户端登录和提交聊天数据的请求，需要创建一个处理客户端数据的服务器端脚本文件。在本案例中，使用.NET中的Web页面文件DealData.aspx处理客户端的用户数据请求，该文件的完整代码如代码清单16-10所示。

代码清单 16-10　处理客户端数据请求的 Web 页面文件 DealData.aspx 的完整代码

```
<%@ Import Namespace="System.Collections.Generic" %>
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
    string strAction = System.Web.HttpUtility.UrlDecode(
                       Request["action"]);
    string strName = System.Web.HttpUtility.UrlDecode(
                       Request["name"]);
    string strPass = System.Web.HttpUtility.UrlDecode(
                       Request["pass"]);
    string strContent = System.Web.HttpUtility.UrlDecode(
                       Request["content"]);
    switch (strAction)
    {
        case "Login":
            Response.Clear();
            Response.Write(UserLogin(strName,strPass));
            Response.End();
            break;
        case "ChatList":
            Response.Clear();
            Response.Write(AllChatList());
            Response.End();
            break;
        case "OnLineList":
            Response.Clear();
            Response.Write(GetOnlineUserList());
            Response.End();
            break;
        case "SendContent":
            Response.Clear();
            Response.Write(AddSendContent(strContent));
            Response.End();
            break;
    }
}
/// <summary>
/// 用户登录
/// </summary>
/// <param name="strName"></param>
/// <returns></returns>
private bool UserLogin(string strName, string strPass)
{
    bool blnR = false;
    if (strPass == "123456")// 初始化密码
    {
        List<string> OnLineUserList = (List<string>)Application["OBJUSERLIST"];
        if (OnLineUserList == null)// 对象为 NULL
        {
            OnLineUserList = new List<string>();// 实例化对象
            OnLineUserList.Add(strName);        // 新增对象元素
        }
        else
        {
            OnLineUserList.Add(strName);        // 新增对象元素
        }
        Application["OBJUSERLIST"] = OnLineUserList;
        Session["LOGINUSER"] = strName;
        blnR = true;
```

```
    }
    return blnR;
}
/// <summary>
/// 新增发送内容
/// </summary>
/// <param name="strContent"></param>
/// <returns></returns>
private bool AddSendContent(string strContent)
{
    string name = Session["LOGINUSER"].ToString();
    string strSendConent = name + " 于 " +
            DateTime.Now.ToLongTimeString().ToString()+
            " 说： " + strContent;
    List<string> strSendConentList =
                (List<string>)Application["OBJMESSAGELIST"];
    if (strSendConentList == null)
    {
        strSendConentList = new List<string>();
    }
    strSendConentList.Add(strSendConent);
    Application["OBJMESSAGELIST"] = strSendConentList;
    return true;
}
/// <summary>
/// 获取全部聊天记录
/// </summary>
/// <returns></returns>
private string AllChatList()
{
    string strSendConent = string.Empty;
    List<string> strSendConentList =
                (List<string>)Application["OBJMESSAGELIST"];
    if (strSendConentList == null)
    {
        strSendConent = " 目前还没有找到聊天记录 ";
    }
    else
    {
        foreach (string str in strSendConentList)
        {
            strSendConent += str + "</br>";
        }
    }
   strSendConent= strSendConent.Replace("<:", "<img src='Face/");
   strSendConent=strSendConent.Replace(":>", ".gif '/>");
   return strSendConent;
}
/// <summary>
/// 获取在线人员列表
/// </summary>
/// <returns></returns>
private string GetOnlineUserList()
{
    string strOnLineUserListHtml = string.Empty;
```

```
        List<string> strOnLineUserList =
                     (List<string>)Application["OBJUSERLIST"];
        if (strOnLineUserList == null)
        {
           strOnLineUserList = new List<string>();
        }
        foreach (string str in strOnLineUserList)
        {
            strOnLineUserListHtml += str + "</br>";
        }
        return strOnLineUserListHtml;
    }
</script>
```

16.2.4　代码分析

在本案例中，如果涉及HTML页面与服务器进行数据交互的操作时，全部使用jQuery中Ajax的$.ajax()方法进行数据的提交与接收，同时，将两个Ajax的全局事件ajaxStart()与ajaxStop()绑定页面中元素。触发$.ajax()方法操作时，将显示ajaxStart()与ajaxStop()事件绑定的页面元素，优化用户的交互体验。在登录页面中代码如下：

```
... 省略部分代码
// 元素绑定全局 ajaxStart 事件
$("#divMsg").ajaxStart(function() {
    $(this).show(); // 显示元素
})
 // 元素绑定全局 ajaxStop 事件
$("#divMsg").ajaxStop(function() {
    $(this).html(" 请求处理已完成。").hide();
})
... 省略部分代码
```

通过jQuery中Ajax的$.ajax()方法，将获取的用户名与密码提交给服务器，其部分代码如下：

```
function UserLogin(name, pass) {
    $.ajax({
        type: "POST",
        url: "DealData.aspx",
        data: "action=Login&d=" + new Date() + "&name=" + name + "&pass=" + pass,
        success: function(data) {
            if (data == "True") {
                //... 省略部分代码
            }
            else {
                //... 省略部分代码
            }
        }
    });
}
```

在自定义函数UserLogin()中，参数name表示用户名，pass表示登录密码。在传值过程中，参数data表示传递给服务器页面DealData.aspx的内容。其中字符串中通过加入"d=" +new Date()+"字符，可以每次返回不同的数据信息，否则，所返回的数据将被缓存，返回不到最新的数据。

在聊天室主页面中，页面初次被加载时，首先执行三个自定义的函数，分别用于显示图标、返回聊天内容和在线人员，其部分代码如下：

```
$(function() {
    ... 省略部分代码
    InitFace();
    GetMessageList();
    GetOnLineList();
    ... 省略部分代码
});
```

在GetMessageList()函数中调用了定时执行函数AutoUpdContent()，因此聊天室主页面完成加载后，将按定时的时间向服务器发送数据请求，并返回至HTML页面，以保证聊天室主页的聊天内容实时更新。

在发送聊天内容时可以加入图标表情，为了实现这个功能，需要找到每个图标，根据其设置的图标文件名，组合成发送内容的一部分，其部分代码如下：

```
$("table tr td img").click(function() { //表情图标单击事件
    var strContent = $("#txtContent").val() + "<:" + this.id + ":
    $("#txtContent").val(strContent);
})
```

在上述图标单击事件中，字符"table tr td img"表示定位全部的表情图标，以下代码表示在已写入的聊天内容后面增加“<:”与“:>”：

```
$("#txtContent").val() + "<:" + this.id + ":>"
```

并将设置好的每个图标的ID号包含其中，即""<:"+this.id+":>""。

这种格式便于客户端的数据传输及后台替换，当然，也可以设置其他的特殊字符进行代码，只要便于操作即可。在Web服务器页面中，

将使用对应的方法替换这些特殊的字符，本案例中使用的部分替换代码如下：

```
/// <summary>
/// 获取全部聊天记录
/// </summary>
/// <returns></returns>
private string AllChatList()
{
   //...省略部分代码
   strSendConent= strSendConent.Replace("<:", "<img src='Face/");
   strSendConent=strSendConent.Replace(":>", ".gif '/>");
   return strSendConent;
}
```

将客户端传回的特殊图标字符通过上述服务器端代码替换后，就可以写入数据库或返回到前台客户页面中，从而实现在发送内容时添加表情的功能。

说明　在本案例中，用户的聊天内容和在线人员数据，都是通过页面的Application对象进行保存的，单个用户名采用Session对象进行保存，而完全没有使用数据库。在实际的开发过程中，为了便于用户查询每次的聊天内容，还需要创建数据库保存实时的聊天数据。

16.3　本章小结

在本章中，首先通过演示图片切割的全过程，使读者在掌握jQuery基本知识的基础上，学会如何在HTML页面中，通过jQuery切割插件实现任意大小图片切割方法；另外，介绍使用jQuery库与静态HTML页组合开发聊天室的过程，使读者初步了解如何在实际的开发项目中，巧妙使用jQuery库的方法。