



jQuery权威指南 第二版

◎作者：陶国荣

目录

第一章 jQuery 简介

第二章 jQuery 选择器

第三章 jQuery 操作DOM

第十四章 jQuery Mobile基础知识

第十五章 jQuery Mobile 综合案例开发

第十六章 jQuery 综合案例开发



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[⊖]本书主页下载。

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

[⊖] 华章网址www.hzbook.com。



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。

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。



图 1-1 jQuery 官方网站

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>
        您好！欢迎来到 jQuery 的精彩世界。
    </div>
</body>
</html>
```

```

        });
    </script>
</head>
<body>
    <div></div>
</body>
</html>

```

(3) 页面效果

页面效果如图 1-2 所示。

(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
}

```



图 1-2 第一个 jQuery 程序

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

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

标记是不可见的，当用户单击主题

标记时，改变自身的背景色，并将内容

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。

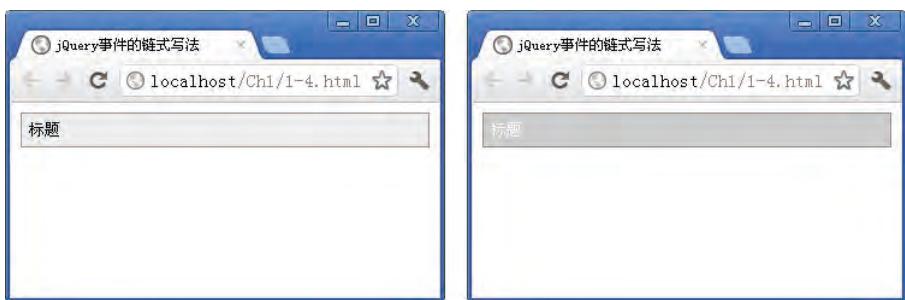


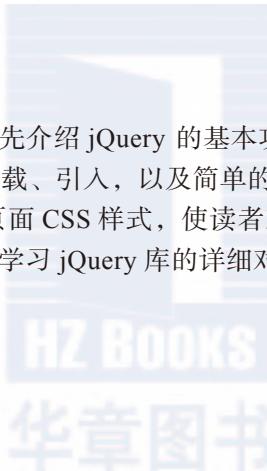
图 1-5 jQuery 控制 CSS

(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，封装了大量的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}
    </style>

```

```

    #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>

```

```

</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。

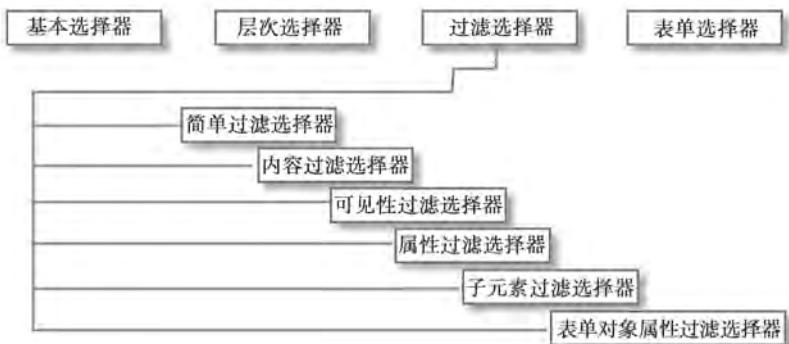


图 2-3 jQuery 选择器分类示意图

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 `` 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
<code>\$("#divOne").css("display","block");</code>	显示 ID 为 divOne 的页面元素	
<code>\$(". div span ").css("display","block");</code>	显示元素名为 span 的页面元素	
<code>\$(".clsFrame .clsOne").css("display","block");</code>	显示类别名为 clsOne 的页面元素	
<code>\$("*").css("display","block");</code>	显示页面中的所有元素	
<code>\$("#divOne,span").css("display","block");</code>	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 `` 标记，在该 `` 标记中又新增一个 `` 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">
```

```

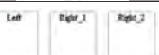
<head>
    <title> 使用 jQuery 层次选择器 </title>
    <script language="javascript" type="text/javascript"
        src="Jscript/jquery-1.8.2.min.js"></script>
    <style type="text/css">
        body{font-size:12px;text-align:center}
        div,span{float:left;border:solid 1px #ccc; margin:8px;display:none}
        .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
<code>\$("div span") .css("display","block");</code>	显示 <div> 中所有的 标记	
<code>\$("div>span") .css("display","block")</code>	显示 <div> 中子 标记	
<code>\$("#divMid + div") .css("display","block");</code>	显示 ID 为 divMid 元素后的下一个 <div>	
<code>\$("#divMid ~ div") .css("display","block");</code>	显示 ID 为 divMid 元素后的所有 <div>	
<code>\$("#divMid").siblings("div") .css("display","block");</code>	显示 ID 为 divMid 元素的所有相邻 <div>	

(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 标记并在其中放置四个 ，再创建一个 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
        })
    </script>

```

```

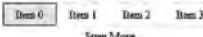
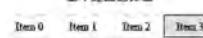
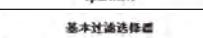
        $("#spnMove:animated").addClass("GetFocus");
    })
    function animateIt() { // 动画效果
        $("#spnMove").slideToggle("slow", animateIt);
    }
</script>
</head>
<body>
<div>
    <h1> 基本过滤选择器 </h1>
    <ul>
        <li class="DefClass">Item 0</li>
        <li class="DefClass">Item 1</li>
        <li class="NotClass">Item 2</li>
        <li class="DefClass">Item 3</li>
    </ul>
    <span id="spnMove">Span Move</span>
</div>
</body>
</html>

```

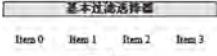
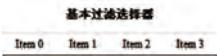
(3) 页面效果

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
<code>\$("#li:first"). addClass("GetFocus");</code>	增加第一个元素的类别	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:last"). addClass("GetFocus");</code>	增加最后一个元素的类别	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:not(.NotClass)"). addClass("GetFocus");</code>	增加去除所有与给定选择器匹配的元素类别	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:even"). addClass("GetFocus");</code>	增加所有索引值为偶数的元素类别，从 0 开始计数	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:odd"). addClass("GetFocus");</code>	增加所有索引值为奇数的元素类别，从 0 开始计数	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:eq(1)"). addClass("GetFocus");</code>	增加一个给定索引值的元素类别，从 0 开始计数	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:gt(1)"). addClass("GetFocus");</code>	增加所有大于给定索引值的元素类别，从 0 开始计数	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
<code>\$("#li:lt(4)"). addClass("GetFocus");</code>	增加所有小于给定索引值的元素类别，从 0 开始计数	 基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move

(续)

关键代码	功能描述	页面效果
<code>\$(":header"). .addClass("GetFocus");</code>	增加标题类元素类别	
<code>\$("#spanMove:animated"). .addClass("GetFocus");</code>	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滤选择器查找 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

标记，并在其中一个

中新建一个标记，其余

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滤选择器锁定 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 和

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
<code>\$(“div:visible”).addClass(“GetFocus”);</code>	增加所有可见元素类别	
<code>\$(“span:hidden”).show().addClass(“GetFocus”);</code>	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">
```

```

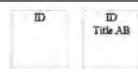
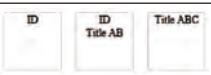
        src="Jscript/jquery-1.8.2.min.js"></script>
<style type="text/css">
    body{font-size:12px;text-align:center}
    div{float:left;border:solid 1px #ccc;margin:8px;
        width:65px;height:65px;display:none}
</style>
<script type="text/javascript">
    $(function(){ // 显示所有含有 id 属性的元素
        $("div[id]").show(3000);
    })
    $(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
<code>\$(“div[id]”). show(3000);</code>	显示所有含有 id 属性的元素	
<code>\$(“div[title=‘A’]”). show(3000);</code>	显示所有属性 title 的值是 "A" 的元素	
<code>\$(“div[title!=‘A’]”). show(3000);</code>	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
<code>\$("#div[title^='A']").show(3000);</code>	显示所有属性 title 的值以 "A" 开始的元素	
<code>\$("#div[title\$='C']").show(3000);</code>	显示所有属性 title 的值以 "C" 结束的元素	
<code>\$("#div[title*= 'B']").show(3000);</code>	显示所有属性 title 的值中含有 "B" 的元素	
<code>\$("#div[id='divAB'][title*= 'B']").show(3000);</code>	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
<code>:nth-child(eq even odd index)</code>	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
<code>:first-child</code>	获取每个父元素下的第一个子元素	元素集合
<code>:last-child</code>	获取每个父元素下的最后一个子元素	元素集合
<code>:only-child</code>	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 `` 标记，前两个标记中设置四个 ``，后一个标记中设置一个 ``。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

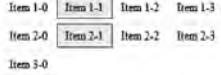
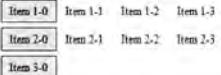
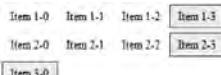
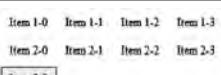
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<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
<code>\$("#li:nth-child(2)").addClass("GetFocus");</code>	增加每个父元素下的第二个子元素类别	
<code>\$("#li:first-child").addClass("GetFocus");</code>	增加每个父元素下的第一个子元素类别	
<code>\$("#li:last-child").addClass("GetFocus");</code>	增加每个父元素下的最后一个子元素类别	
<code>\$("#li:only-child").addClass("GetFocus");</code>	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>

```

```
</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
<code>\$("#form1 input:enabled").addClass("GetFocus");</code>	增加表单中所有属性为可用的元素类别	
<code>\$("#form1 input:disabled").addClass("GetFocus");</code>	增加表单中所有属性为不可用的元素类别	
<code>\$("#form1 input:checked").addClass("GetFocus");</code>	增加表单中所有被选中的元素类别	
<code>\$("#Span2").html("选中项是：" + \$("select option:selected").text());</code>	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。

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。

表 2-17 表单选择器语法

选择器	功能描述	返回值
<code>:input</code>	获取所有 input、textarea、select	元素集合
<code>:text</code>	获取所有单行文本框	元素集合
<code>:password</code>	获取所有密码框	元素集合
<code>:radio</code>	获取所有单选按钮	元素集合
<code>:checkbox</code>	获取所有复选框	元素集合
<code>:submit</code>	获取所有提交按钮	元素集合
<code>:image</code>	获取所有图像域	元素集合
<code>:reset</code>	获取所有重置按钮	元素集合
<code>:button</code>	获取所有按钮	元素集合
<code>:file</code>	获取所有文件域	元素集合

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:241px}
        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);
        })
    </script>

```

```

        $("#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
<code>\$("#form1 div").html(" 表单共找出 Input 类型元素 :" + \$("#form1 :input").length);</code>	显示 Input 类型元素的总数量	表单共找出 Input 类型元素:11
<code>\$("#form1 :text").show(3000);</code>	显示所有文本框对象	<input type="text"/> Text
<code>\$("#form1 :password").show(3000);</code>	显示所有密码框对象	<input type="password"/> *****
<code>\$("#form1 :radio").show(3000);</code>	显示所有单选按钮对象	<input checked="" type="radio"/> Radio
<code>\$("#form1 :checkbox").show(3000);</code>	显示所有复选框对象	<input type="checkbox"/> CheckBox
<code>\$("#form1 :submit").show(3000);</code>	显示所有提交按钮对象	<input type="submit"/> Submit
<code>\$("#form1 :image").show(3000);</code>	显示所有图片域对象	

(续)

关键代码	功能描述	页面效果
<code>\$("#form1 :reset").show(3000);</code>	显示所有重置按钮对象	
<code>\$("#form1 :button").show(3000);</code>	显示所有按钮对象	
<code>\$("#form1 :file").show(3000);</code>	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：

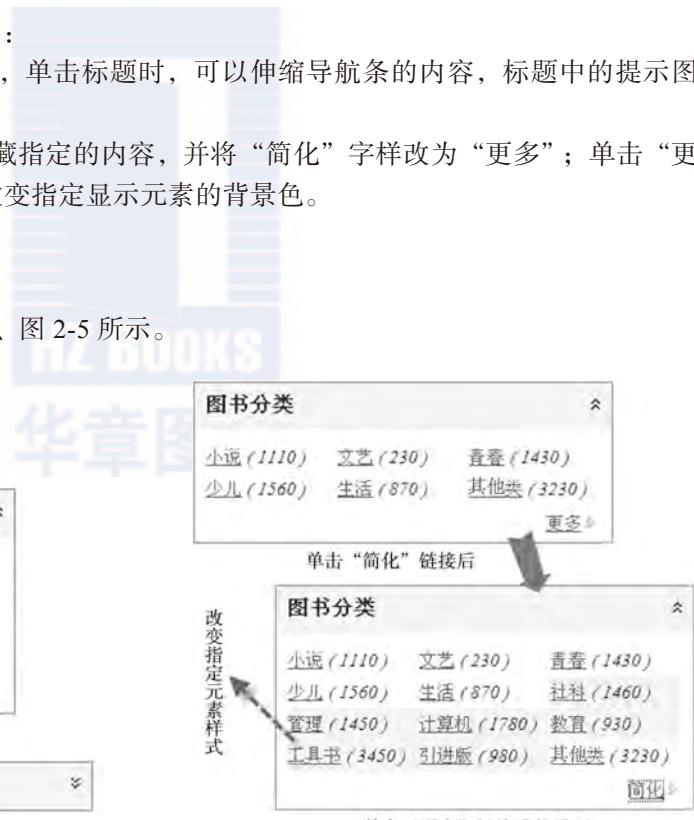
- 1) 在页面中创建一个导航条，单击标题时，可以伸缩导航条的内容，标题中的提示图片也随之改变。
- 2) 单击“简化”链接时，隐藏指定的内容，并将“简化”字样改为“更多”；单击“更多”链接时，返回初始状态，并改变指定显示元素的背景色。

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。



单击标题后的界面



图书分类

小说 (1110) 文艺 (230) 查看 (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)
[更多 >](#)

单击“简化”链接后

图书分类

小说 (1110) 文艺 (230) 查看 (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)
[商业 >](#)

单击“更多”链接后的界面

改变指定元素样式

图 2-4 单击导航条标题前后的界面

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
            $(".clsHead").click(function(){
                if($(".clsContent").is(":visible")){
                    $(".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");
                }
            });

            $(".clsBot > a").click(function(){
                // 如果内容为 "简化" 字样
                if($(".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)")
                }
            });
        });
    </script>

```

```

        .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></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>
            
        </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中下的标记，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。

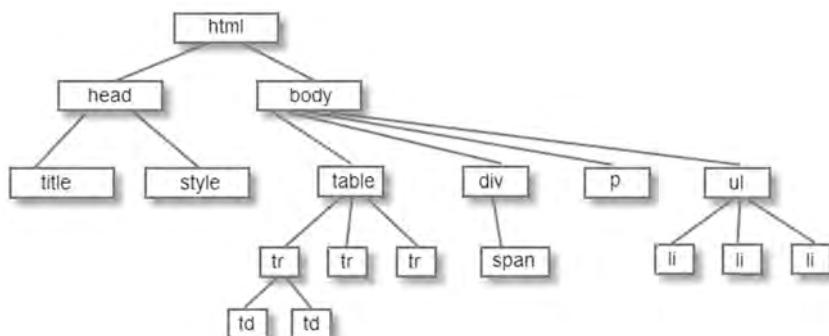


图 3-2 DOM 树状模型

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 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>
    
    <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。



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 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>
    
    <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" });
            $("img").attr("title", "这是一幅风景画");
            $("img").addClass("clsImg");
        })
    </script>
</head>
<body>
    
</body>
</html>
```

(3) 页面效果

代码执行后的效果如图3-5所示。

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中的src属性：

```
$(“img”).removeAttr(“src”);
```



图3-5 随机展示图片

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 HTML 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
<code>html()</code>	无参数	用于获取元素的 HTML 内容
<code>html(val)</code>	<code>val</code> 参数为元素的 HTML 内容	用于设置元素的 HTML 内容
<code>text()</code>	无参数	用于获取元素的文本内容
<code>text(val)</code>	<code>val</code> 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>
</head>
```

```

<script type="text/javascript">
$(function() {
    $("img").click(function() {
        $(this).toggleClass("clsImg"); // 切换样式类别
    })
})
</script>
</head>
<body>
    
</body>
</html>

```

(3) 页面效果

代码执行后的效果如图 3-10 所示。

(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();
```

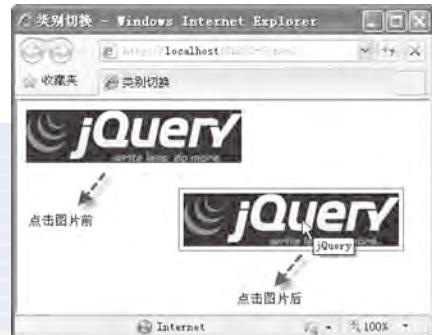


图 3-10 类别切换

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。

(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。



图 3-11 动态创建节点元素

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
<code>append(content)</code>	<code>content</code> 表示追加到目标中的内容	向所选择的元素内部插入内容
<code>append(function(index, html))</code>	通过 <code>function</code> 函数返回追加到目标中的内容	向所选择的元素内部插入 <code>function</code> 函数所返回的内容
<code>appendTo(content)</code>	<code>content</code> 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
<code>prepend(content)</code>	<code>content</code> 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
<code>prepend(function(index, html))</code>	通过 <code>function</code> 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 <code>function</code> 函数所返回的内容
<code>prependTo(content)</code>	<code>content</code> 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 function 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。

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。例如如果要将 span 标记插入 div 标记中，则执行下列代码：

```
$(“span”).appendTo( $("div") );
```

即把 appendTo 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 appendTo() 方法，将一个元素标记插入另一个标记中的过程。



图 3-12 插入节点

示例 3-12 使用 appendTo() 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 appendTo() 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>
    
    <span></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>

```

```

        </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。



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></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。

(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串替换所选择的元素，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
<code>wrap(html)</code>	<code>html</code> 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
<code>wrap(elem)</code>	<code>elem</code> 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
<code>wrap(fn)</code>	<code>fn</code> 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
<code>unwrap()</code>	无参数	移除所选元素的父元素或包裹标记
<code>wrapAll(html)</code>	<code>html</code> 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
<code>wrapAll(elem)</code>	<code>elem</code> 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
<code>wrapInner(html)</code>	<code>html</code> 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
<code>wrapInner(elem)</code>	<code>elem</code> 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
<code>wrapInner(fn)</code>	<code>fn</code> 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。



图3-16 替换元素节点

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>
        
    </p>
</body>

```

```


</p>
</body>
</html>

```

(3) 页面效果

代码执行后的效果如图 3-18 所示。



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，通过该表达式获取指定的元素，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。

(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”。

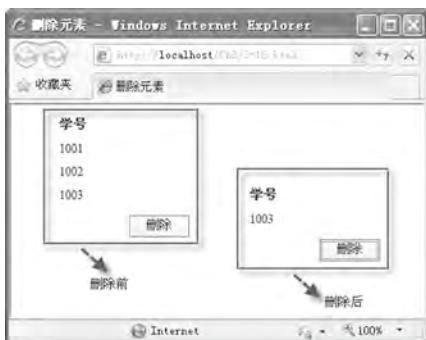


图 3-19 删除元素

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。

选项	编号	封面	购书人	性别	购书价
<input type="checkbox"/>	1001		李小明	男	35.60 元
<input type="checkbox"/>	1002		刘明明	女	37.80 元
<input type="checkbox"/>	1003		张小丽	女	45.60 元

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:pointer}
        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 **/
    </script>

```

```

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></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></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></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>

</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言基础。



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，新添加许多酷炫的部件，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 rtttop.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。

(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。

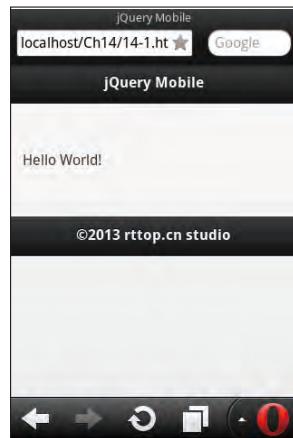


图 14-1 第一个 jQuery Mobile 页面

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">
            
        </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">
            
        </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

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

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

段落元素，当用户点击标题中的“+”号时，显示

元素中的内容，标题左侧中“+”号变成“-”号再次点击时，隐藏

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>
            </div>
        </div>
    </div>
</body>

```

```

<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

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。

(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



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

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各异的渲染效果。

(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。

(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 `` 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 `` 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，``、`` 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 ``、`` 元素的 `` 标签中，添加子列表 `` 或 `` 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 `` 或 `` 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 `` 元素，并在该元素中增加两个 `` 选项元素，主题分别为“图书”、“音乐”；然后，在两个 `` 元素中，分别添加另外两个与之对应的 `` 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" />
```



图 14-6 表单元素

```

<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[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。

(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。



图 14-8 默认配置项设置

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。

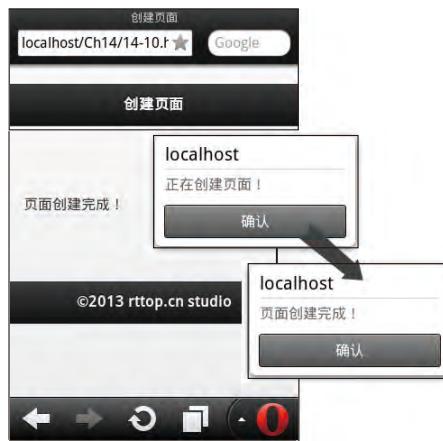


图 14-10 jQuery Mobile 中事件的触发

(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能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: 控制元素在屏幕上显示的位置、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。

(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。



图 14-11 jQuery Mobile 中选择主题



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。

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。

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 15-1 新闻订阅系统封面页的效果



图 15-2 新闻订阅系统首页



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。

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。



图 15-4 “头条”类别新闻



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></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 `` 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 `` 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>

```

```

        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

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

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 `<h4>` 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-1.8.2.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="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">
                                ' + item.news_title +
                            '</a>
                        </li>';
                        $listview.append($li);
                    }
                });
            }
        }
    };
    $tpl_Index_List($p_array, $p_items);
});
```

```

        data-catename=" + item.news_catename + ""
        data-id=" + item.news_cateid + "
        style="margin:0px;padding:0px 0px 0px 55px">
        
        <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="newscale.htm" data-ajax="false"
                        data-catename=" + item.news_catename + "
                        data-id=" + item.news_cateid + "
                        style="margin:0px;padding:0px 0px 0px 55px">
                        
                        <h3>' + item.news_catename + '</h3>
                    ';
                    $p_array.push($li);
                }
            })
        }
    }
    $tpl_Sub_List($p_array, $p_items);
})

```

```

        <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">
        
        <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 + '"'
            '</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 + " &ampnbsp来源：" + item.news_source;
                $info.html(strHTML);
                $content.html(" &ampnbsp&ampnbsp&ampnbsp" +
                    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 选项元素中自定义最右侧的单击按钮图标，首先在 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，作为列表元素显示的内容；同时，刷新列表元素，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”的函数型变量。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 `` 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 `` 列表，则再执行刷新操作后，被赋值的内容便可即时显示在 `` 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。
- 2) 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；根据该 ID 号使用 \$.getJSON() 方法请求相应的 JSON 数据。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。

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，主要包括以下几个需求：

- 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。

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 `` 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 15-6 新手导航页

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 `` 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。

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 15-9 记事详细页和删除数据时的效果



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

元素和一个

列表元素。前者用于放置系统的功能图片，后者

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:">
                    </a>
                <a href="javascript:">
                    </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”容器中添加一个列表元素，用于显示某类别下的记事数据。

另外，将列表元素的“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

和两个 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>

```

```

</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 _ 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="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>
    </div>
</div>

```

```

</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).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 jsonnotedata = JSON.stringify(notedata);
        rttophtml5mobi.utils.setParam(strnid, jsonnotedata);
        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 $title = $("#title");
    var $content = $("#content");
    var listData = JSON.parse(rttophtml5mobi.utils.getParam($strId));
    var strType = listData.type == "a" ? "散文" : "随笔";
    $type.html(strType);
    $title.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 $xtxacontent = $("#xtxa-content");
    var editData = JSON.parse(rttophtml5mobi.utils.getParam($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 jsonnotedata = JSON.stringify(notedata);
        rttophtml5mobi.utils.setParam(strnid, jsonnotedata);
        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 rtttopHtml5.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 rtttopHtml5.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 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元素的内容，并保存在变量“\$strHTML”中，调用列表元素的 html() 方法，将内容赋值于页面列表元素中；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 rtttopHtml5.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 rtttopHtml5.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 rtttopHtml5.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。

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。

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。

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 16-2 选择切割区后的页面效果

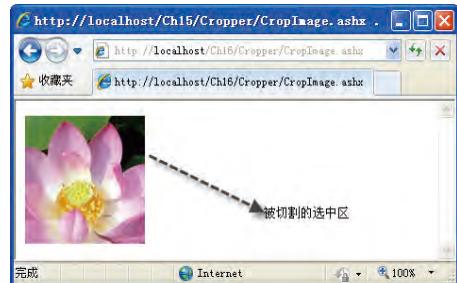


图 16-3 最终切割的图片

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() {
            $('#cropbox').Cropper({
                aspectRatio: 1, // 调用切割方法
                onChange: showPreview, // 设置所选区域的长宽比例
                onSelect: showPreview // 设置长宽变化后触发的事件
            })
        });
        // 自定义提交切割数据时的检测函数
        function checkCoords() {
            if (parseInt($('#w').val())) return true;
        }
    </script>

```

```

        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>
                
                <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">
                </div><div class="clsAlt">100*100 像素</div>
                <div class="img80">
                </div><div class="clsAlt">80*80 像素</div>
                <div class="img60">
    </div><div class="clsAlt">60*60 像素</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({
    aspectRatio: 1,           // 调用切割方法
    onChange: showPreview,   // 设置所选区域的长宽比例
    onSelect: showPreview    // 设置长宽变化后触发的事件
})
});
```

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，但不少企业和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。

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。

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 16-4 用户登录聊天室



图 16-5 首次登录聊天主窗口页面



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。

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>用户名: <input id="txtName" type="text" class="txt" /></div>
            <div>密码: <input id="txtPass" type="password" class="txt"/></div>
            <div class="btnCenter">
```

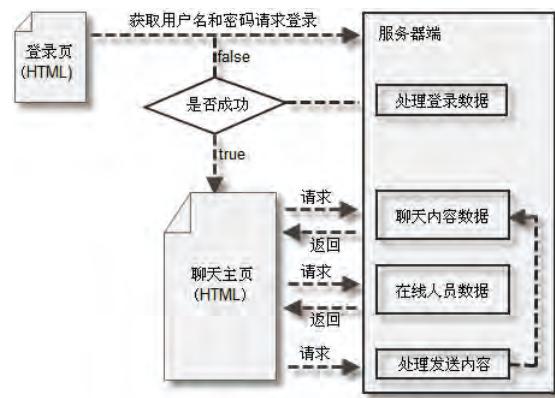


图 16-7 在线聊天室操作流程图

```

<input id="Button1" type="button" value="登录"
       class="btn" />&nbsp;&nbsp;
    <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
}
/* 偷察请求状态样式 */
.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 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 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
}
/* 偷察请求状态样式 */
.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.ToString("yyyy-MM-dd HH:mm:ss") +
        " 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。