HZ BOOKS
华章科技

内容全面，系统地讲解了jQuery的方方面面
实战性强，全书包含118个示例和2个综合案例
资深专家亲自执笔，4大专业社区一致鼎力推荐

實戰

基于最新版 jQuery

陶国荣 著

jQuery 权威指南

机械工业出版社
China Machine Press

實戰

jQuery中文社区联合策划

jQuery的发展之迅速和取得的成功之巨大是其他所有开发框架都难以企及的，它已经成为Web开发者必备的核心技能之一。如果你尚未掌握jQuery或功力还不够，推荐你认真阅读这本书并付诸实践。与同类的jQuery图书相比，它有3大优势：内容非常全面，几乎包含jQuery的所有内容；基于jQuery的最新版本撰写，所有的新功能特性都一览无余；实战性极强，不仅有118个示例，而且还有2个综合案例。

——jQuery中文社区 (jquery.org.cn)

jQuery因为使用简单、功能强大、插件丰富而深受Web开发者青睐。本书不仅完整地呈现了最新版jQuery本身所有的功能，而且讲解了jQuery UI等扩展功能；更值得一提的是，它还包括最佳实践和性能优化方面的技巧，内容全面、结构合理。除此之外，本书还包括大量的示例，不仅每个知识点都配有小例子，而且还有两个综合性的案例。对于初学者而言，本书应该是学习jQuery的首选。

——jQuery中文用户组

jQuery因为易于使用和功能强大著称，是所有Web开发者应该掌握的一种利器。初学者如何才能快速而有效地掌握jQuery呢？最好的方法莫过于一边学习理论，一边动手实践这些理论，本书就是按照这种思路为读者打造的，强烈推荐！

——JavaScript开发者社区

jQuery从众多的Ajax框架中脱颖而出，已经成为Web开发领域的事实标准。本书除了理论知识丰富且全面外，它还有一个最大的特点就是注重实战，每个知识点都有一个完整的案例，包括需求分析、代码实现和结果展示3个部分，而且还包含2个综合性的案例，它的实践性之强是目前所有同类书都不具备的，恰好这一点又是初学者最需要的。如果能阅读本书并付诸实践，进入jQuery开发的佳境便指日可待的了。

——Ajax中国

客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

封面设计：陈子平

上架指导：计算机 / 程序设计 / Web开发

ISBN 978-7-111-32543-7

9 787111 325437

定价：59.00元

實戰

jQuery: The Definitive Guide

jQuery 权威指南

陶国荣 著

机械工业出版社
China Machine Press

本书由国内资深 Web 技术专家亲自执笔，4 大 Web 开发社区一致鼎力推荐，权威性毋庸置疑。

内容新颖，基于 jQuery 的最新版本撰写，所有新功能和新特性一览无余；内容全面，不仅讲解了 jQuery 技术本身的方方面面，而且还包括与 jQuery 相关的扩展知识；实战性强，不仅每个知识点都配有完整的小案例，而且还有两个综合性的案例。本书不仅能满足读者系统学习理论知识的需求，还能满足需要充分实践的需求。

全书一共分为 11 章，首先以示例的方式对 jQuery 做了全局性的介绍，以便于为读者建立 jQuery 的大局观，这对初学者尤为重要；其次详细讲解了 jQuery 的各种选择器、jQuery 操作 DOM 的方法、jQuery 中的事件与应用、jQuery 中的动画和特效、Ajax 在 jQuery 中的应用，以及各种常用的 jQuery 插件的使用方法和技巧，所有这些知识点都配有完整的示例（包括需求分析、代码实现和结果展示三部分）；再接着对 jQuery UI 和 jQuery 实用工具函数等扩展知识，以及 jQuery 的开发技巧与性能优化等方面的重要知识做了详尽的阐述；最后以两个具有代表性的综合案例结束全书，希望能帮助读者将前面所学的理论知识真正贯穿于实践中，迅速进入 jQuery 的殿堂。

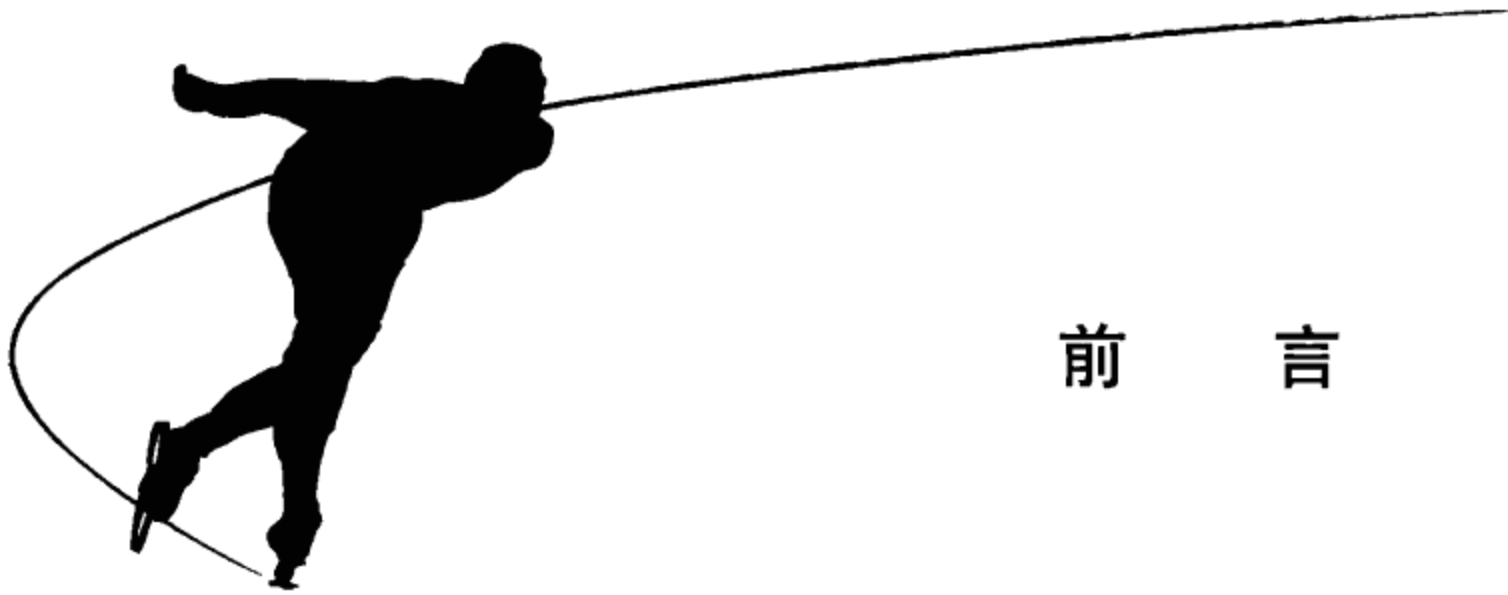
图书在版编目（CIP）数据

jQuery 权威指南 / 陶国荣著 .—北京：机械工业出版社，2011.1
ISBN 978-7-111-32543-7

Ⅰ. j… Ⅱ. 陶… Ⅲ. JAVA 语言－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2010）第 227721 号

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：陈佳媛
北京京北印刷有限公司印刷
2011 年 1 月第 1 版第 1 次印刷
186mm×240mm • 24 印张
标准书号：ISBN 978-7-111-32543-7
定价：59.00 元



前　言

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

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

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，当我们要开发一个 Web 应用时，不仅仅只是要考虑其功能是否足够完备，更重要的是要考虑如何才能提高用户的体验满意度。这是理性的回归，也是 Web 开发技术发展的必然趋势，而 jQuery 恰恰是满足这一理性需求的坚实利刃。

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。综观现在已经出版的中文类 jQuery 图书，不是简单的概念性介绍，就是缺乏真正的实践指导，而且版本相对陈旧。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，本书诞生了，相信它不会让你失望。

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，完美地展现了 jQuery 最新版本的功能和特性。
- 内容全面、丰富、翔实，不仅由浅入深地讲解了 jQuery 的所有必备基础知识，还介绍了 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。
- 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。不仅书中的每一个小知识点都配有精心选择的小案例（总共 100 多个），而且还有两个非常实用的综合性案例。所有案例的讲解都非常详细，不仅有功能需求分析和完整实现代码，而且还有最终效果的展示，更重要的是，将所有理论知识都巧妙地贯穿于其中，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，在实际开发中应该就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

由于本书的结构是层进式的，章节之间有一定的关联，因此建议读者按章节的编排顺序逐章阅读。但在阅读本书的示例时，请尽量不要照抄书中的所有示例，而是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，这样才能真正领会示例所反映出的 jQuery 技术的理论本质。

联系作者

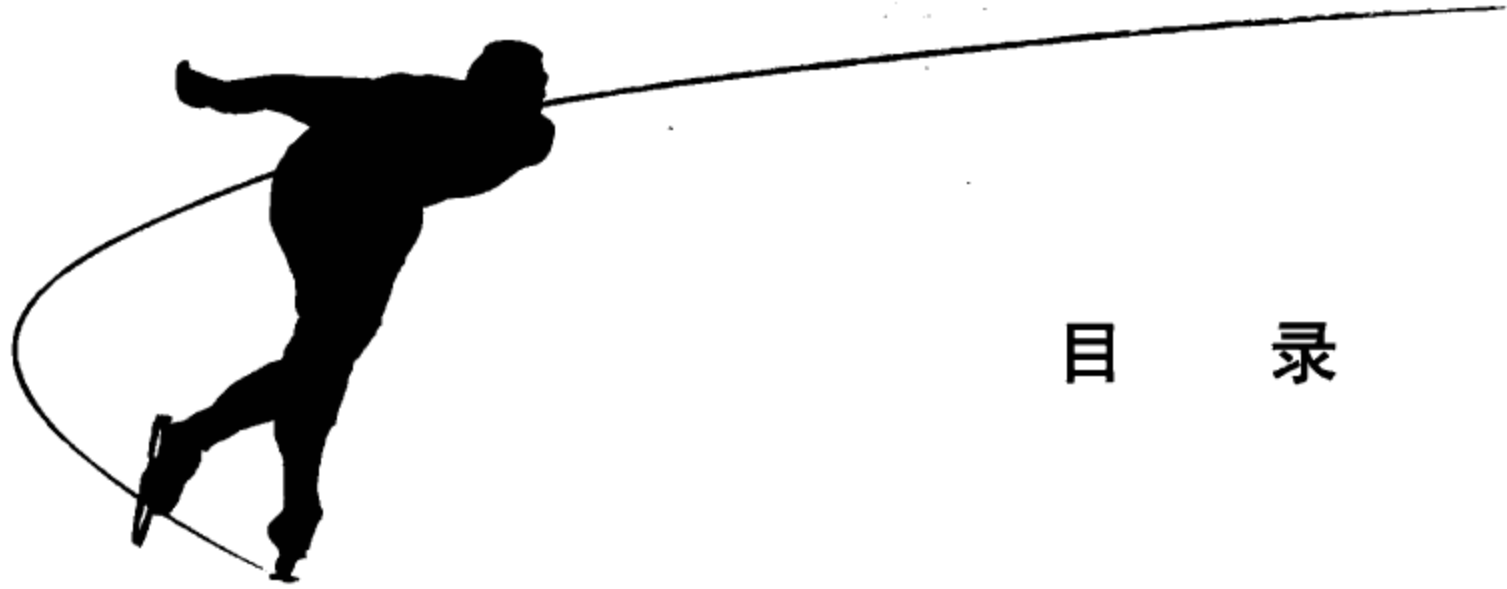
希望这部耗时数月、承载了我近 4 年 jQuery 开发心得和体会的拙著能给每一位阅读过它的读者带来技术上的提升和思路上的启发。非常希望能借本书出版的机会与国内热衷于 jQuery 技术的开发者交流，如果大家想联系我，欢迎给我发邮件：tao_guo_rong@163.com。

致谢

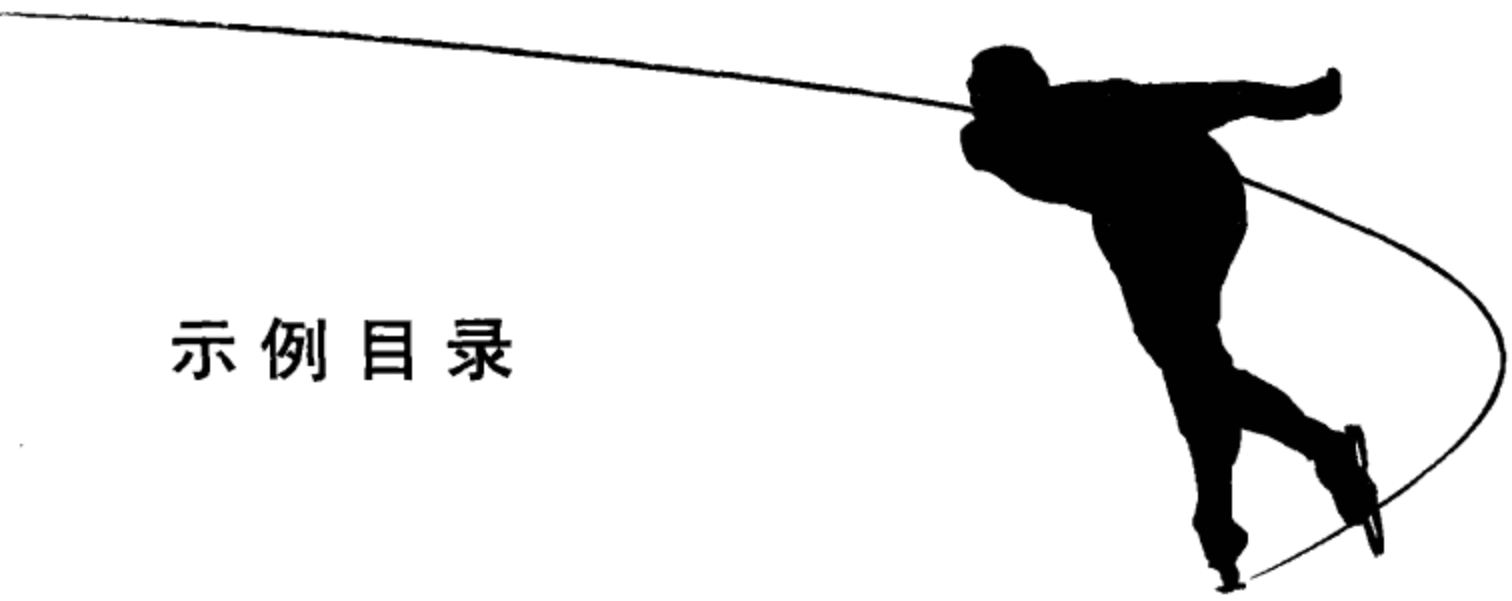
本书能顺利出版，首先要感谢机械工业出版社华章分社的编辑们，尤其是杨福川编辑。正是由于他们在我写作的整个过程中不断地给予专业的指导，才使得我整体的创作思路不断被提升和改进，使本书能保质保量地完成。同时，我还要感谢我的家人，正是他们的理解与默默支持，才使得我能全心写作、顺利完成本书的编写。

陶国荣

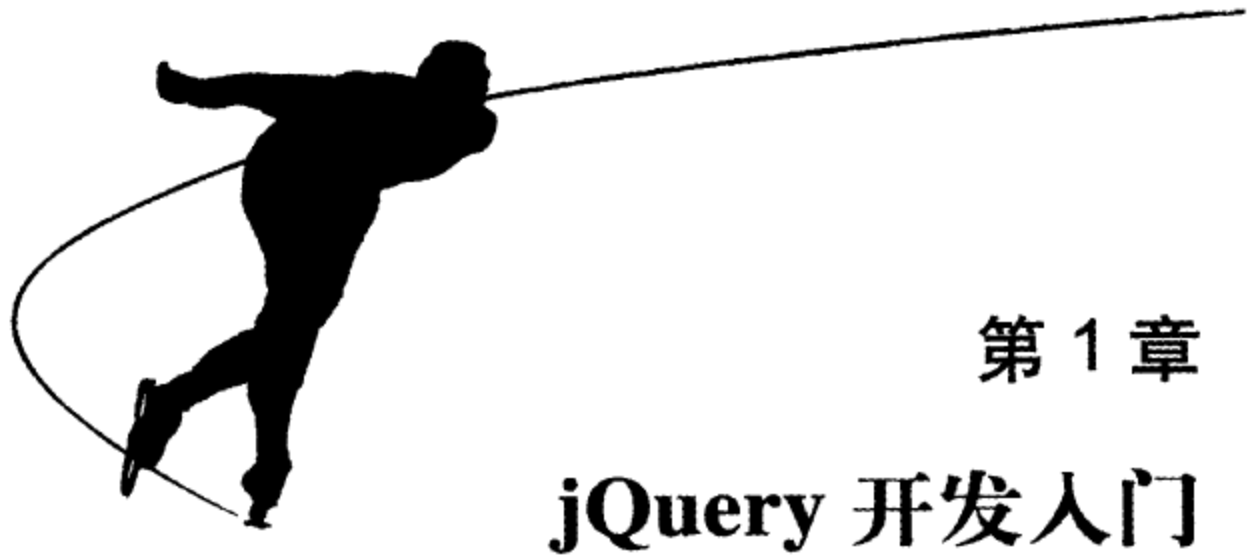
2010 年 11 月



目　　录



示例目录



第 1 章

jQuery 开发入门

本章内容

- jQuery 概述
- jQuery 的简单应用
- 本章小结

随着互联网的迅速发展，Web 页面得到了广泛应用，但人们的需求已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，帮助程序开发者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，代码高效、兼容性强等特点使得它近年来风靡全球，越来越多的开发者痴迷其中。

1.1 jQuery 概述

1.1.1 认识 jQuery

jQuery 是由美国人 John Resig 于 2006 年创建的一个开源项目，随着被人们熟知，越来越多的程序高手加入其中，完善并壮大其项目内容，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）。

1.1.2 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、移动还是复制某元素，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；其具体示例，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，可以使页面的表现层与功能开发分离，开发者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。然后，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地丰富了页的展示效果，使原来使用 JavaScript 代码遥不可及的功能通过插件的引入而轻松地实现。

5. 与 Ajax 技术的完美结合

Ajax 的异步读取服务器数据的方法，极大地方便了程序的开发，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，通过其内部对象或函数，加上几行代码就可以实现复杂的功能。

综上所述，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。

1.1.3　搭建 jQuery 开发环境

1. 下载 jQuery 文件库

在 jQuery 的官方网站（http:// jquery.com）中，下载最新版本的 jQuery 文件库，其网站页面如图 1-1 所示。

图 1-1　jQuery 的官方网站

在网站中，选择大小为 24KB 的压缩包，单击 Download（下载）按钮，便可以将最新版的 jQuery 框架下载到本地。目前（截止到 2010 年 7 月）最新版本为 V1.4.2。

2. 引入 jQuery 文件库

下载完 jQuery 框架文件后，并不需要任何安装，仅需要使用 <script> 文件导入标记，将 jQuery 框架文件 jquery-1.4.2.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-1.4.2.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，就可以开始我们的 jQuery 之旅了。

1.1.4　编写第一个简单的 jQuery 应用

首先，我们来编写一个简单的程序，参见下面的示例。

示例 1-1 编写第一个简单的 jQuery 程序

（1）功能描述

在页面加载时，弹出一个模式对话框，显示“您好，欢迎来到 jQuery 世界”字样，单击“确定”按钮后关闭该窗口。

（2）实现代码

新建一个 HTML 文件 1-1.html，加入如代码清单 1-1 所示的代码。

代码清单 1-1 第一个简单的 jQuery 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>第一个简单的 jQuery 程序 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></script>
    <script type="text/javascript">
           $(document).ready(function(){
              alert(" 您好，欢迎来到 jQuery 世界 ");
           })
    </script>
</head>
<body>
</body>
</html>
```

在上述文件的代码中，有一段如下的代码：

```
$(document).ready(function(){
                  // 程序段
})
```

该段代码类似于传统的 JavaScript 代码：

```
window.onload=function(){
                  // 程序段
}
```

虽然上述两段代码在功能上可以互换，但它们之间又有许多区别：

- 执行时间不同：$(document).ready 在页面框架下载完毕后就执行；而 window.onload 必须在页面全部加载完毕（包含图片下载）后才能执行。很明显，前者的执行效率高于后者。
- 执行数量不同：$(document).ready 可以重复写多个，并且每次执行结果不同；而 window.onload 尽管可以执行多个，但仅输出最后一个执行结果，无法完成多个结果的输出。
- $(document).ready(function(){}) 可以简写成 $(function(){})，因此与下面的代码是等价的。

```
$(document).ready(function(){
                // 程序段
})
```

等价于

```
$(function(){
                // 程序段
})
```

（3）页面效果

HTML 文件 1-1.html 最后实现的页面效果如图 1-2 所示。

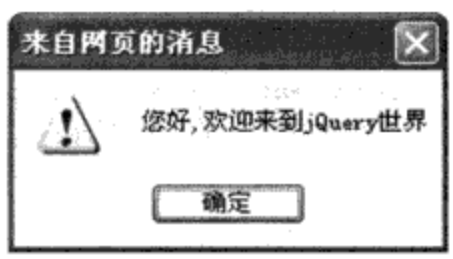


图 1-2　第一个 jQuery 程序运行后的页面效果

1.1.5　jQuery 程序的代码风格

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，都必须使用该符号，可以说它是 jQuery 程序的标志。

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

在页面中，有一个 标记，在该标记内，包含二个 标记，一个为主题，另一个为内容，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，加入如代码清单 1-2 所示的代码。

代码清单 1-2　jQuery 事件的链式写法

```
<!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>jQuery 事件的链式写法 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></script>
    <style type="text/css">
           .divFrame{width:260px;border:solid 1px #666;
                     font-size:10pt}
           .divTitle{background-color:#eee;padding:5px}
           .divContent{padding:5px;display:none}
           .divCurrColor{ background-color:Red}
    </style>
    <script type="text/javascript">
         $(function(){
          $(".divTitle").click(function(){
            $(this).addClass("divCurrColor")
                   .next(".divContent").css("display","block");
          });
         });
    </script>
</head>
<body>
    <div class="divFrame">
         <div class="divTitle"> 主题 </div>
         <div class="divContent">
             <a href="#"> 链接一 </a><br />
             <a href="#"> 链接二 </a><br />
             <a href="#"> 链接三 </a>
         </div>
    </div>
</body>
</html>
```

在上述文件的代码中，加粗的代码是链式写法。

代码功能说明：当用户单击 Class 名称为 "divTitle" 的元素时，自身增加名称为 "divCurrColor" 的样式；同时，将接下来的 Class 名称为 "divContent" 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。

（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。

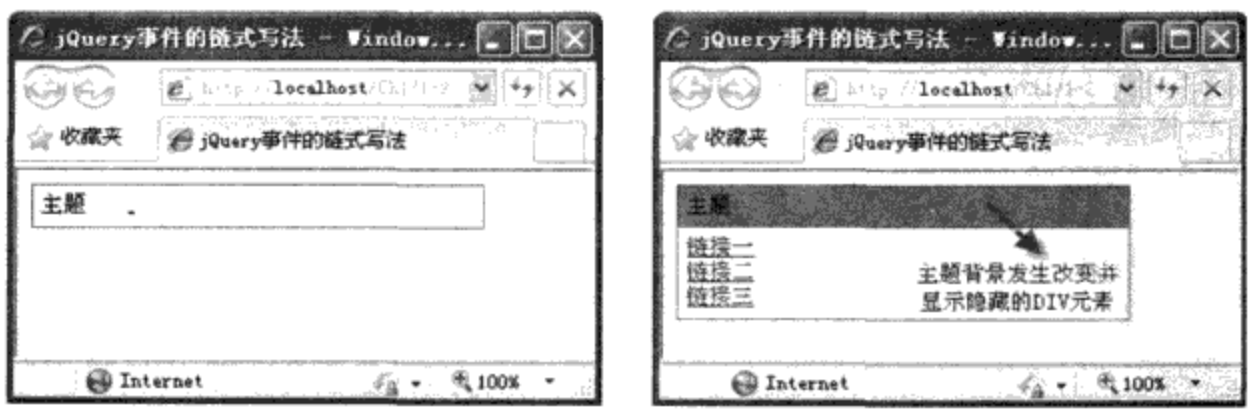


图 1-3　DIV 元素单击前后的页面对比效果

1.2　jQuery 的简单应用

1.2.1　jQuery 访问 DOM 对象

1. 什么是 DOM 对象

DOM（Document Object Model，文本对象模型）的每一个页面都是一个 DOM 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：在页面中 2 个 标记元素，其代码如下：

```
<div id="divTmp">测试文本</div>
<div id="divOut"></div>
```

通过下面的 JavaScript 代码可以访问 DOM 对象和获取或设置其内容值：

```
var tDiv=document.getElementById("divTmp"); // 获取 DOM 对象
var oDiv=document.getElementById("divOut"); // 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 "divTmp" 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，我们称之为 jQuery 对象；为了同样实现在 ID 为 "divOut" 的标记中显示 ID 为 "divTmp" 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=$("#divTmp"); // 获取 jQuery 对象
var oDiv=$("#divOut"); // 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
```

通过代码的对比，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。

1.2.2　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 标记中。

（2）实现代码

新建一个 HTML 文件 1-3.html, 加入如代码清单 1-3 所示的代码。

代码清单 1-3　使用传统的 JavaScript 方法访问 DOM 对象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>控制 DOM 对象 </title>
    <style type="text/css">
            .divFrame{width:260px;border:solid 1px #666;
                      font-size:10pt}
            .divTitle{background-color:#eee;padding:5px}
            .divContent{padding:8px;font-size:9pt}
            .divTip{width:244px;border:solid 1px #666;
                    padding:8px;font-size:9pt;
                    margin-top:5px;display:none}
            .txtCss{border:solid 1px #ccc}
            .divBtn{padding-top:5px}
            .divBtn .btnCss{border:solid 1px #535353;width:60px}
    </style>
    <script type="text/javascript">
         function btnClick(){
          //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("divTip").style.display="block";
          //设置文本元素的内容
          document.getElementById("divTip").innerHTML=oTxtValue+"<br>"
          +oRdoValue+"<br>"+oChkValue;
         }
    </script>
</head>
<body>
<div class="divFrame">
    <div class="divTitle">请输入如下信息 </div>
    <div class="divContent">
        姓名：<input id="Text1" type="text" class="txtCss"/><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" />
        <div class="divBtn"><input id="Button1" type="button"
```

```
                    value=" 提交 " class="btnCss"
                    onclick="btnClick();" />
        </div>
    </div>
</div>
<div id="divTip" class="divTip"></div>
</body>
</html>
```

在上面的代码中，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("divTip").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。修改后的 JavaScript 代码如下所示：

```
...
<script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.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
    $("#divTip").css("display","block").html(oTxtValue+"<br>"+oRdoValue+"<br>"+
    oChkValue);
    })
})
</script>
...
```

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，则返回“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 代码要简洁得多，并且还兼容各种浏览器。

（3）页面效果

最终实现的页面效果如图 1-4 所示。

1.2.3　jQuery 控制页面 CSS

在 jQuery 框架中，通过自带的 JavaScript 编程，提供全部的 CSS3 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指定的某元素中，而不用考虑浏览器的兼容性。

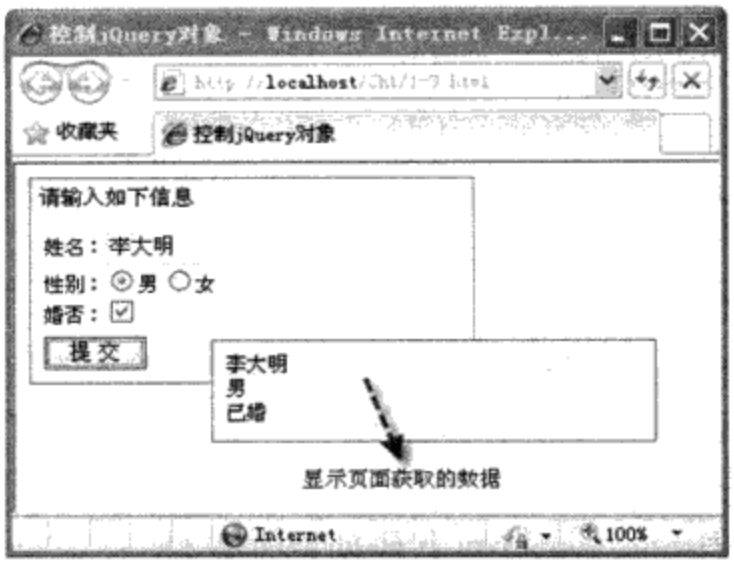


图 1-4　控制 jQuery 对象

下面通过一个简单的示例来介绍如何使用 jQuery 中的 toggleClass(className) 方法来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 元素标记，当用户单击该元素时，变换其文本内容和背景色，再次单击时，恢复其初始的内容和背景色。

（2）实现代码

新建一个 HTML 文件 1-4.html，加入如代码清单 1-4 所示的代码。

代码清单 1-4　jQuery 控制 CSS 样式

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>jQuery 控制 CSS 样式 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></script>
    <style type="text/css">
           .divDefalut{width:260px;font-size:10pt;padding:5px}
           .divClick{width:260px;border:solid 1px #666;
           font-size:10pt;background-color:#eee;padding:5px}
    </style>
    <script type="text/javascript">
         $(function(){
              $(".divDefalut").click(function(){
              $(this).toggleClass("divClick").html(" 点击后的样式 ");
              });
         });
```

```
    </script>
</head>
<body>
    <div class="divDefalut">点击前的样式</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。

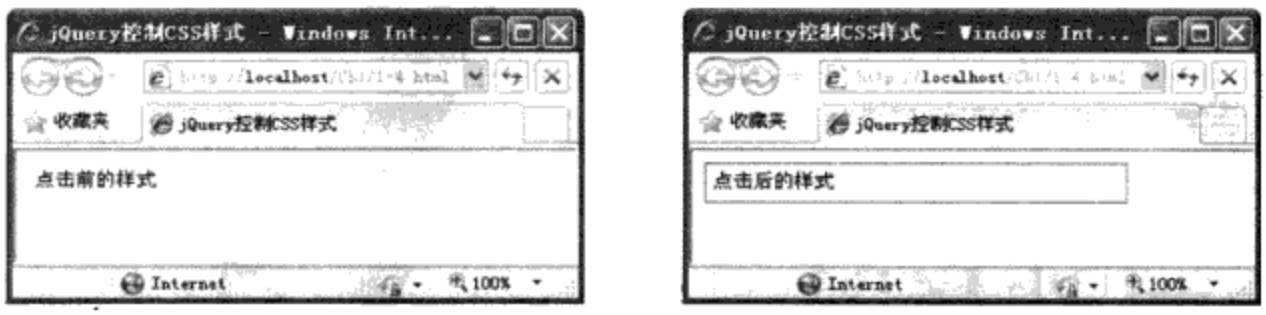
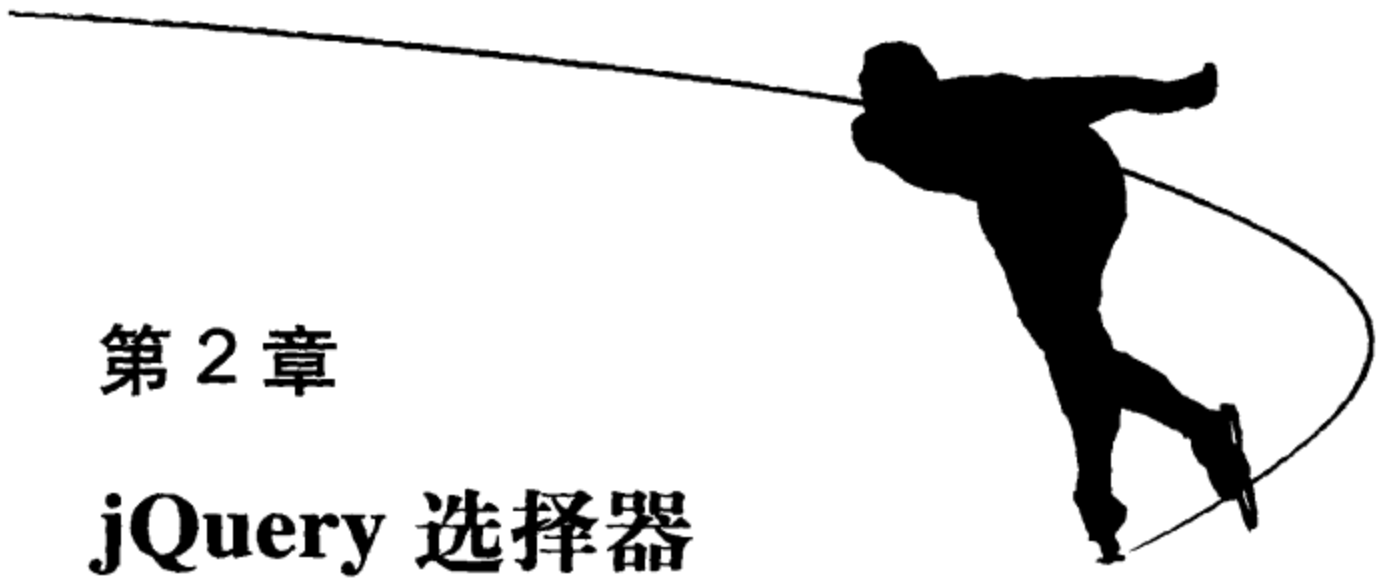


图 1-5　jQuery 控制 CSS

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。

1.3　本章小结

本章通过循续渐进的方式，先从 jQuery 的基础概念入手，介绍 jQuery 库的下载，引入简单应用方法；后部分侧重于 jQuery 控制 DOM 对象和页面 CSS 样式的介绍，通过一些简单的小示例，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

- jQuery 选择器概述
- jQuery 选择器详解
- 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。

2.1　jQuery 选择器概述

2.1.1　什么是选择器

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。

2.1.2　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。

1. 代码更简单

由于在 jQuery 库中，封装了大量可以通过选择器直接调用的方法或函数，使编写代码更加简单轻松，简单几行代码就可以实现较为复杂的功能。

下面通过一个实现表格隔行变色功能的示例，分别使用传统的 JavaScript 语言与 jQuery 语言加以说明。

示例 2-1　使用 JavaScript 实现隔行变色

（1）功能描述

在页面中，通过一个 <table> 标记展示数据列表信息，在元素标记中，奇数行与偶数行的背景色不同，从而实现隔行变色的页面展示效果。

（2）实现代码

使用传统的 JavaScript 实现该页面功能。新建一个 HTML 文件 2-1.html，加入如代码清单 2-1 所示的代码。

代码清单 2-1　使用 JavaScript 实现隔行变色

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

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 JavaScript 实现隔行变色 </title>
    <style type="text/css">
        body{font-size:12px;text-align:center}
        #tbStu{width:260px;border:solid 1px #666;
               background-color:#eee}
        #tbStu tr{line-height:23px}
        #tbStu tr th{background-color:#ccc;color:#fff}
        #tbStu .trOdd{background-color:#fff}
    </style>
    <script type="text/javascript">
        window.onload=function(){
        var oTb=document.getElementById("tbStu");
           for(var i=0;i<oTb.rows.length-1;i++){
              if(i%2){
              oTb.rows[i].className="trOdd";
              }
           }
        }
    </script>
</head>
<body>
<table id="tbStu" cellpadding="0" cellspacing="0">
    <tbody>
        <tr>
            <th>学号 </th><th>姓名 </th><th>性别 </th><th>总分 </th>
        </tr>
            <!-- 奇数偶 -->
        <tr>
            <td>1001</td><td>张小明 </td><td>男 </td><td>320</td>
        </tr>
            <!-- 偶数偶 -->
        <tr>
            <td>1002</td><td>李明琪 </td><td>女 </td><td>350</td>
        </tr>
        <!--...-->
    </tbody>
</table>
</body>
</html>
```

在代码清单 2-1 中，首先通过 ID 号获取表格元素，然后遍历表格的各行，根据行号的奇偶性，动态设置该行的背景色，从而实现隔行变色的页面效果。

页面中的 JavaScript 代码虽可以实现最终效果，但循环页面的元素会影响打开速度，并且代码较为复杂，如果使用 jQuery 选择器实现上述页面效果，则需要在页面中加入一些代码，详见下面的示例。

示例 2-2　使用 jQuery 选择器实现隔行变色

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 jQuery 选择器实现隔行变色

```
...
<head>
    <title>使用 jQuery 选择器实现隔行变色 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.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
```

从代码清单 2-2 中可以看出，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。

（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

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，无需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 2-3　使用 JavaScript 输出文字信息

（1）功能描述

在页面 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 2-3.html，加入如代码清单 2-3 所示的代码。

代码清单 2-3　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>
```

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，则在浏览器中将出现如图 2-2 所示的出错提示信息。

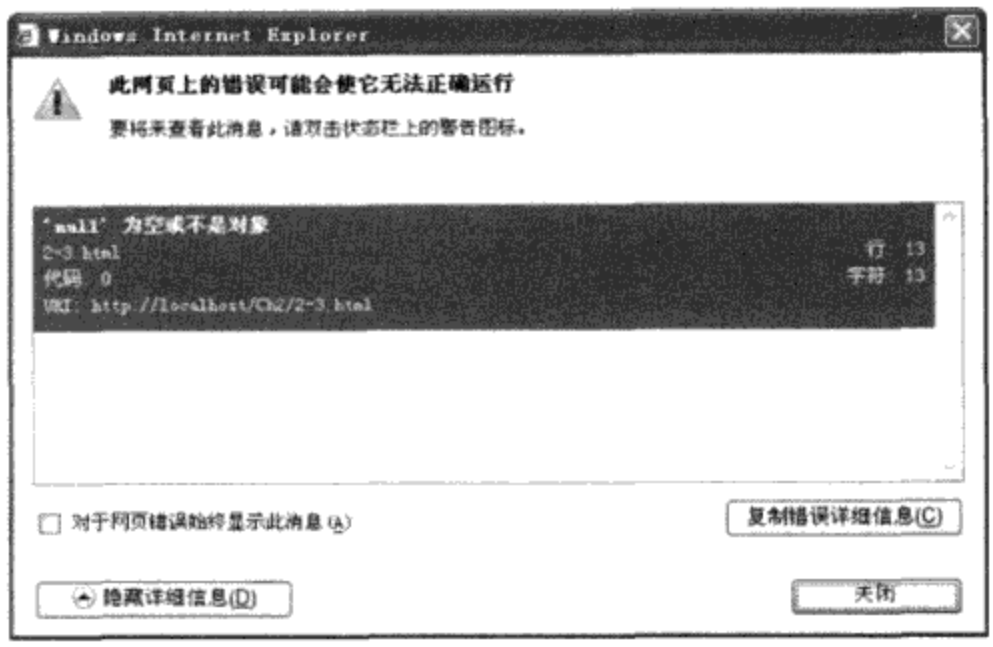


图 2-2　页面对象不存在的出错提示信息

如果将例 2-3 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例所示。

示例 2-4　使用 jQuery 输出文字信息

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 输出文字信息

```
...
<head>
    <title>jQuery 代码检测页面元素 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></script>
    ... 省略样式代码
    <script type="text/javascript">
          $(function(){
            $("#divT").html(" 这是一个检测页面 ");
         })
    </script>
</head>
... 省略页面主体代码
```

2.2　jQuery 选择器详解

根据所获取页面中元素的不同，可以将 jQuery 选择器分为：基本选择器、层次选择器、过滤选择器、表单选择器四大类。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 6 种。其分类结构如图 2-3 所示。

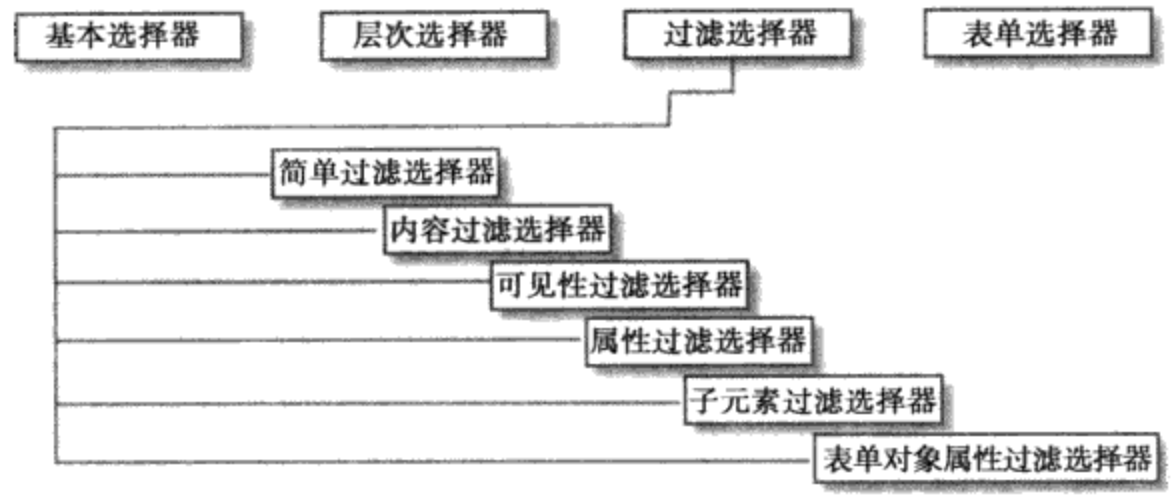


图 2-3　jQuery 选择器分类示意图

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	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 2-5 来介绍各种基本选择器在页面中使用的方法。

示例 2-5　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 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 2-5.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-1.4.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           .clsFrame{width:300px;height:100px}
           .clsFrame div,span{display:none;float:left;
                    width:65px;height:65px;border:solid 1px #ccc;
                    margin:8px}
           .clsOne{background-color:#eee}
    </style>
    <script type="text/javascript">
            $(function(){ //ID 匹配元素
              $("#divOne").css("display","block");
            })
            $(function(){ // 元素名匹配元素
              $("div span").css("display","block");
            })
            $(function(){ // 类匹配元素
              $(".clsFrame .clsOne").css("display","block");
            })
            $(function(){ // 匹配所有元素
              $("*").css("display","block");
            })
            $(function(){ // 合并匹配元素
              $("#divOne,span").css("display","block");
            })
    </script>
</head>
<body>
<div class="clsFrame">
    <div id="divOne">ID</div>
    <div class="clsOne">CLASS</div>
    <span>SPAN</span>
</div>
</body>
</html>
```

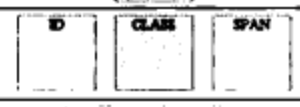
（3）页面效果

为了能更清楚地看到每个基本选择器执行后的结果，下面通过表格的方式展示页面效果，如表 2-2 所示。

表 2-2　页面执行效果

关键代码	功能描述	页面效果
$("#divOne") .css("display","block");	显示 ID 为 divOne 的页面元素	

（续）

关键代码	功能描述	页面效果
$(" div span ") .css("display","block");	显示元素名为 span 的页面元素	
$(".clsFrame .clsOne") .css("display","block");	显示类别名为 clsOne 的页面元素	
$("*") .css("display","block");	显示页面中的所有元素	
$("#divOne,span") .css("display","block");	显示 ID 为 divOne 和元素名为 span 的页面元素	

2.2.2 层次选择器

层次选择器通过 DOM 元素间的层次关系获取元素，其主要的层次关系包括后代、父子、相邻、兄弟关系，通过其中某类关系可以方便快捷地定位元素，其详细说明如表 2-3 所示。

表 2-3 层次选择器语法

选择器	功能	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

说明 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 2-6 来演示各种层次选择器在页面中选择 DOM 元素的方法。

示例 2-6 使用 jQuery 层次选择器选择元素

（1）功能描述

在页面中，设置 4 块 标记，其中在第二块 中，添加 1 个 标记，在该 标记中又新增 1 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 2-6.html，加入如代码清单 2-6 所示的代码。

代码清单 2-6 使用 jQuery 层次选择器选择元素

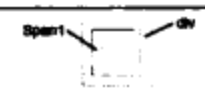
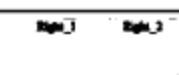
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<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-1.4.2.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

代码清单 2-6 执行后的效果如表 2-4 所示。

表 2-4 页面执行效果

关键代码	功能描述	页面效果
$("div span") .css("display","block");	显示 中所有的 标记	
$("div>span") .css("display","block")	显示 中子 标记	
$("#divMid + div") .css("display","block");	显示 ID 为 divMid 元素后的下一个	
$("#divMid ~ div") .css("display","block");	显示 ID 为 divMid 元素后的所有	
$("#divMid").siblings("div") .css("display","block");	显示 ID 为 divMid 元素的所有相邻	

说明 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	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 2-7 来介绍如何通过过滤选择器定位 DOM 元素的方法。

示例 2-7 使用 jQuery 基本过滤选择器选择元素

（1）功能描述

在页面中，设置一个 <h1> 标记用于显示主题，创建 <ul> 标记并在其中放置四个 ，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 2-7.html，加入如代码清单 2-7 所示的代码。

代码清单 2-7　使用 jQuery 基本过滤选择器选择元素

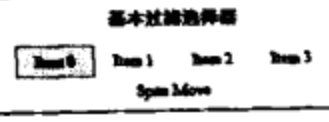
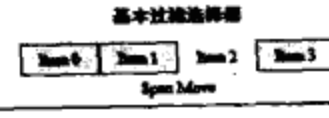
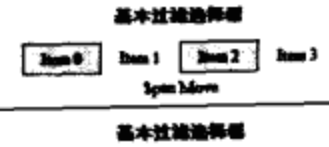
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 基本过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></script>
    <style type="text/css">
            body{font-size:12px;text-align:center}
            div{width:241px;height:83px;border:solid 1px #eee}
            h1{font-size:13px}
            ul{list-style-type:none;padding:0px}
            .DefClass,.NotClass{height:23px;width:60px;
                      line-height:23px;float:left;
                      border-top:solid 1px #eee;border-bottom:solid 1px #eee}
            .GetFocus{width:58px;border:solid 1px #666;
                      background-color:#eee}

            #spnMove{width:238px;height:23px;line-height:23px;}
    </style>
    <script type="text/javascript">
            $(function(){ // 增加第一个元素的类别
      $("li:first").addClass("GetFocus");
    })
    $(function(){ // 增加最后一个元素的类别
      $("li:last").addClass("GetFocus");
    })
    $(function(){ // 增加去除所有与给定选择器匹配的元素类别
      $("li:not(.NotClass)").addClass("GetFocus");
    })
    $(function(){ // 增加所有索引值为偶数的元素类别，从 0 开始计数
      $("li:even").addClass("GetFocus");
    })
    $(function(){ // 增加所有索引值为奇数的元素类别，从 0 开始计数
      $("li:odd").addClass("GetFocus");
    })
    $(function(){ // 增加一个给定索引值的元素类别，从 0 开始计数
      $("li:eq(1)").addClass("GetFocus");
    })
    $(function(){ // 增加所有大于给定索引值的元素类别，从 0 开始计数
      $("li:gt(1)").addClass("GetFocus");
    })
    $(function(){ // 增加所有小于给定索引值的元素类别，从 0 开始计数
      $("li:lt(4)").addClass("GetFocus");
    })
```

```
            $(function(){ // 增加标题类元素类别
              $("div h1").css("width","238");
              $(":header").addClass("GetFocus");
            })
            $(function(){
              animateIt(); // 增加动画效果元素类别
              $("#spnMove:animated").addClass("GetFocus");
            })
            function animateIt() { // 动画效果
              $("#spnMove").slideToggle("slow", animateIt);
            }
    </script>
</head>
<body>
    <div>
        <h1>基本过滤选择器</h1>
        <ul>
            <li class="DefClass">Item 0</li>
            <li class="DefClass">Item 1</li>
            <li class="NotClass">Item 2</li>
            <li class="DefClass">Item 3</li>
        </ul>
        <span id="spnMove">Span Move</span>
    </div>
</body>
</html>
```

（3）页面效果

代码清单 2-7 执行后的效果如表 2-6 所示。

表 2-6　页面执行效果

关键代码	功能描述	页面效果
$("li:first") .addClass("GetFocus");	增加第一个元素的类别	
$("li:last") .addClass("GetFocus");	增加最后一个元素的类别	
$("li:not(.NotClass)") .addClass("GetFocus");	增加去除所有与给定选择器匹配的元素类别	
$("li:even") .addClass("GetFocus");	增加所有索引值为偶数的元素类别，从 0 开始计数	
$("li:odd") .addClass("GetFocus");	增加所有索引值为奇数的元素类别，从 0 开始计数	
$("li:eq(1)") .addClass("GetFocus");	增加一个给定索引值的元素类别，从 0 开始计数	

（续）

关键代码	功能描述	页面效果
$("li:gt(1)") .addClass("GetFocus");	增加所有大于给定索引值的元素类别，从 0 开始计数	
$("li:lt(4)") .addClass("GetFocus");	增加所有小于给定索引值的元素类别，从 0 开始计数	
$(":header") .addClass("GetFocus");	增加标题类元素类别	
$("#spnMove:animated") .addClass("GetFocus");	增加动画效果元素类别	

说明　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	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 2-8 来展示在页面中如何通过内容过虑选择器查找 DOM 元素的方法。

示例 2-8　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，并在其中的 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 2-8.html，加入如代码清单 2-8 所示的代码。

代码清单 2-8　使用 jQuery 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-1.4.2.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

代码清单 2-8 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	

说明　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	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 2-9 介绍如何使用可见性过虑选择器锁定 DOM 元素的方法。

示例 2-9　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 2-9.html，加入如代码清单 2-9 所示的代码。

代码清单 2-9　使用 jQuery 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-1.4.2.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

代码清单 2-9 执行后的效果如表 2-10 所示。

表 2-10 页面执行效果

关键代码	功能描述	页面效果
$("div:visible") .addClass("GetFocus");	增加所有可见元素类别	
$("span:hidden") .show().addClass("GetFocus");	增加所有不可见元素类别	

说明 :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	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 2-10 介绍使用属性过滤选择器获取 DOM 元素的方法。

示例 2-10 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 标记，并设置不同的 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 2-10.html，加入如代码清单 2-10 所示的代码。

代码清单 2-10　使用 jQuery 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-1.4.2.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

代码清单 2-10 执行后的效果如表 2-12 所示。

表 2-12 页面执行效果

关键代码	功能描述	页面效果
$("div[id]") .show(3000);	显示所有含有 id 属性的元素	ID ID Title AB
$("div[title='A']") .show(3000);	显示所有属性 title 的值是 "A" 的元素	Title A
$("div[title!='A']") .show(3000);	显示所有属性 title 的值不是 "A" 的元素	ID ID Title AB Title ABC
$("div[title^='A']") .show(3000);	显示所有属性 title 的值以 "A" 开始的元素	Title A ID Title AB Title ABC
$("div[title$='C']") .show(3000);	显示所有属性 title 的值以 "C" 结束的元素	Title ABC
$("div[title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 的元素	ID Title AB Title ABC
$("div[id='divAB'][title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 且属性 id 的值是 "divAB" 的元素	ID Title AB

说明 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	返回值
:nth-child(eq|even|odd|index)	获取每个父元素下的特定位置元素，索引号从 1 开始	元素集合
:first-child	获取每个父元素下的第一个子元素	元素集合
:last-child	获取每个父元素下的最后一个子元素	元素集合
:only-child	获取每个父元素下的仅有一个子元素	元素集合

下面通过示例 2-11 来演示使用子元素过滤选择器获取元素的过程。

示例 2-11 使用 jQuery 子元素过滤选择器选择元素

（1）功能描述

在页面中，创建三个 <ul> 标记，前两个标记中设置四个 ，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 2-11.html，加入如代码清单 2-11 所示的代码。

代码清单 2-11　使用 jQuery 子元素过滤选择器选择元素

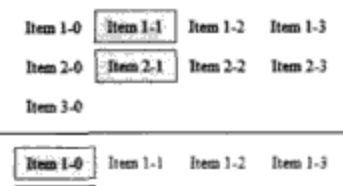
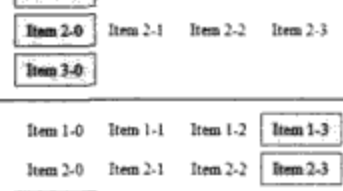
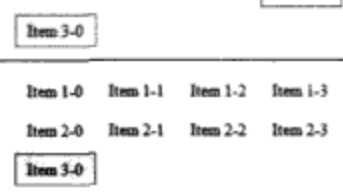
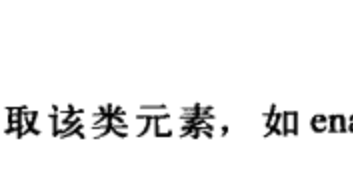
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 子元素过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></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 2 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

代码清单 2-11 执行后的效果如表 2-14 所示。

表 2-14　页面执行效果

关键代码	功能描述	页面效果
$("li:nth-child(2)") .addClass("GetFocus");	增加每个父元素下的第两个子元素类别	
$("li:first-child") .addClass("GetFocus");	增加每个父元素下的第一个子元素类别	
$("li:last-child") .addClass("GetFocus");	增加每个父元素下的最后一个子元素类别	
$("li:only-child") .addClass("GetFocus");	增加每个父元素下的仅有一个子元素类别	

2.2.8　表单对象属性过滤选择器

表单对象属性过滤选择器通过表单中的某对象属性特征获取该类元素，如 enabled、disabled、checked、selected 属性。其详细的说明如表 2-15 所示。

表 2-15　表单对象属性过滤选择器语法

选择器	功能	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 2-12 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 2-12　通过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 2-12.html，加入如代码清单 2-12 所示的代码。

代码清单 2-12　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-1.4.2.min.js"></script>
    <style type="text/css">
           body{font-size:12px;text-align:center}
           div{display:none}
           select{height:65px}
           .clsIpt{width:100px;padding:3px}
           .GetFocus{border:solid 1px #666;background-color:#eee}
    </style>
    <script type="text/javascript">
            $(function(){ // 增加表单中所有属性为可用的元素类别
              $("#divA").show(3000);
              $("#form1 input:enabled").addClass("GetFocus");
            })
            $(function(){ // 增加表单中所有属性为不可用的元素类别
              $("#divA").show(3000);
              $("#form1 input:disabled").addClass("GetFocus");
            })
            $(function(){ // 增加表单中所有被选中的元素类别
              $("#divB").show(3000);
              $("#form1 input:checked").addClass("GetFocus");
            })
            $(function(){ // 显示表单中所有被选中 option 的元素内容
              $("#divC").show(3000);
              $("#Span2").html("选中项是："+
              $("select option:selected").text());
            })
    </script>
</head>
<body>
      <form id="form1" style="width:241px">
            <div id="divA">
                <input type="text" value="可用文本框"
                       class="clsIpt" />
                <input type="text" disabled="disabled"
                       value="不可用文本框" class="clsIpt" />
            </div>
```

```
            <div id="divB">
                <input type="checkbox" checked="checked"
                       value="1" /> 选中
                <input type="checkbox" value="0" /> 未选中
            </div>
            <div id="divC">
            <select multiple="multiple">
                <option value="0">Item 0</option>
                <option value="1" selected="selected">
                        Item 1
                </option>
                <option value="2">Item 2</option>
                <option value="3" selected="selected">
                        Item 3
                </option>
            </select>
            <span id="Span2"></span>
            </div>
        </form>
</body>
</html>
```

（3）页面效果

代码清单 2-12 执行后的效果如表 2-16 所示。

表 2-16 页面执行效果

关键代码	功能描述	页面效果
$("#form1 input:enabled") .addClass("GetFocus");	增加表单中所有属性为可用的元素类别	可用文本框
$("#form1 input:disabled") .addClass("GetFocus");	增加表单中所有属性为不可用的元素类别	可用文本框
$("#form1 input:checked") .addClass("GetFocus");	增加表单中所有被选中的元素类别	选中 未选中
$("#Span2").html(" 选中项是：" + $("select option:selected").text());	显示表单中所有被选中 option 的元素内容	Item 0 Item 1 Item 2 Item 3 选中项是：Item 1Item 3

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

示例 2-13　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，先显示出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 2-13.html，加入如代码清单 2-13 所示的代码。

代码清单 2-13　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-1.4.2.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);
})
$(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

代码清单 2-13 执行后的效果如表 2-18 所示。

表 2-18　页面执行效果

关键代码	功能描述	页面效果
$("#form1 div").html(" 表单共找出 Input 类型元素 :"+ $("#form1 :input").length);	显示 Input 类型元素的总数量	表单共找出 Input 类型元素:11
$("#form1 :text").show(3000);	显示所有文本框对象	Text
$("#form1 :password").show(3000);	显示所有密码框对象	••••••••
$("#form1 :radio").show(3000);	显示所有单选按钮对象	Radio
$("#form1 :checkbox").show(3000);	显示所有复选框对象	CheckBox
$("#form1 :submit").show(3000);	显示所有提交按钮对象	Submit
$("#form1 :image").show(3000);	显示所有图片域对象	jQuery
$("#form1 :reset").show(3000);	显示所有重置按钮对象	Reset
$("#form1 :button").show(3000);	显示所有按钮对象	Button
$("#form1 :file").show(3000);	显示所有文件域对象	浏览...

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，同时，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”字样改变成“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 效果界面

案例实现的界面效果如图 2-4、图 2-5 所示。

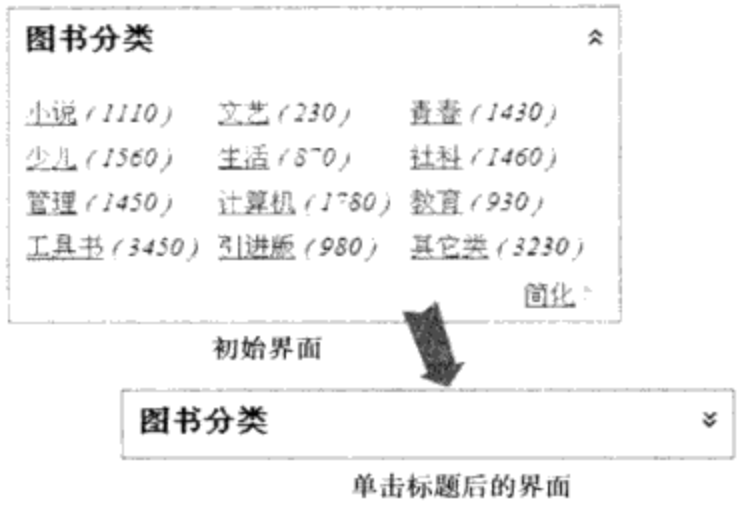


图 2-4 单击导航条标题前后的界面

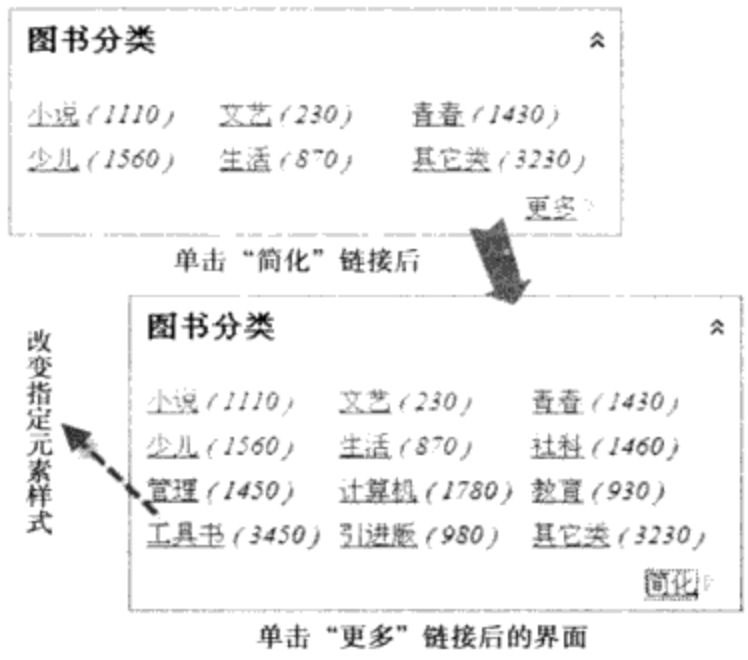


图 2-5 单击超级链接“简化”和“更多”后的界面

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 2-14 所示的代码。

代码清单 2-14 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-1.4.2.min.js"></script>
<style>
     body{font-size:13px}
     #divFrame{border:solid 1px #666;
     width:301px;overflow:hidden}
     #divFrame .clsHead{background-color:#eee;padding:8px;
           height:18px;cursor:hand}
     #divFrame .clsHead h3{padding:0px;margin:0px;float:left}
     #divFrame .clsHead span{float:right;margin-top:3px}
     #divFrame .clsContent{padding:8px}
     #divFrame .clsContent ul {list-style-type:none;
           margin:0px;padding:0px}
     #divFrame .clsContent ul li{ float:left;
           width:95px;height:23px;line-height:23px}
     #divFrame .clsBot{float:right;padding-top:5px;
           padding-bottom:5px}
     .GetFocus{background-color:#eee}
</style>
<script type="text/javascript">
        $(function(){ // 页面加载事件
          $(".clsHead").click(function(){ // 图片单击事件
            if($(".clsContent").is(":visible")){ // 如果内容可见
               $(".clsHead span img")
               .attr("src","Images/a1.gif"); // 改变图片
               // 隐藏内容
               $(".clsContent").css("display","none");
            }else{
               $(".clsHead span img")
               .attr("src","Images/a2.gif"); // 改变图片
               // 显示内容
               $(".clsContent").css("display","block");
            }
          });

          $(".clsBot > a").click(function(){ // 热点链接单击事件
            // 如果内容为 " 简化 " 字样
            if($(".clsBot > a").text()==" 简化 "){
               // 隐藏 index 号大于 4 且不是最后一项的元素
               $("ul li:gt(4):not(:last)").hide();
               // 将字符内容更改为 " 更多 "
               $(".clsBot > a").text(" 更多 ");
            }else{
               $("ul li:gt(4):not(:last)")
               .show()
               .addClass("GetFocus"); // 显示所选元素且增加样式
               // 将字符内容更改为 " 简化 "
               $(".clsBot > a").text(" 简化 ");
            }
          });
        });
```

```
    </script>
</head>
<body>
     <div id="divFrame">
          <div class="clsHead">
               <h3>图书分类</h3>
               <span><img src="Images/a2.gif" alt=""/></span>
          </div>
          <div class="clsContent">
           <ul>
              <li><a href="#">小说</a><i> ( 1110 ) </i></li>
              <li><a href="#">文艺</a><i> ( 230 ) </i></li>
              <li><a href="#">青春</a><i> ( 1430 ) </i></li>
              <li><a href="#">少儿</a><i> ( 1560 ) </i></li>
              <li><a href="#">生活</a><i> ( 870 ) </i></li>
              <li><a href="#">社科</a><i> ( 1460 ) </i></li>
              <li><a href="#">管理</a><i> ( 1450 ) </i></li>
              <li><a href="#">计算机</a><i> ( 1780 ) </i></li>
              <li><a href="#">教育</a><i> ( 930 ) </i></li>
              <li><a href="#">工具书</a><i> ( 3450 ) </i></li>
              <li><a href="#">引进版</a><i> ( 980 ) </i></li>
              <li><a href="#">其他类</a><i> ( 3230 ) </i></li>
           </ul>
          <div class="clsBot"><a href="#">简化</a>
               <img src="Images/ a5.gif" alt=""/>
          </div>
          </div>
     </div>
</body>
</html>
```

2.3.4 代码分析

在页面代码中，首先通过如下代码：

```
$(".clsContent").css("display","none");
```

获取类名称为“clsContent”的元素集合，并实现其内容的显示或隐藏。与此同时，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 <img> 标记，即图片元素；attr(key，value) 是 jQuery 中一个设置元素属性的函数，其功能是为所匹配的元素设置属性值，key 是属性名称，value 是属性值或内容。因此，此行代码的功能是，获取图片元素并改变其图片来源。

为了能够实现单击标题后内容可以伸缩的功能，首先通过如下代码，检测当前内容的隐

藏或显示状态：

```
if($(".clsContent").is(":visible"))
```

其中“$(".clsContent")”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")”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”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 基础
- 访问元素
- 创建节点元素
- 插入节点
- 复制节点
- 替换节点
- 包裹节点
- 遍历元素
- 删除元素
- 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（Document Object Model，文档对象模型）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，我们将详细介绍使用 jQuery 如何操作或控制 DOM 中的各种元素或对象。

3.1　DOM 基础

与 DOM 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。

单词 Document 即文档，当我们创建一个页面并加载到 Web 浏览器时，DOM 模型则根据该页面的内容创建了一个文档文件；单词 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；单词“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

新建一个 HTML 页面 3-1.html，加入如代码清单 3-1 所示的代码，便完成了一个 DOM 页面文档的创建。

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>
```

页面执行后的效果如图 3-1 所示。

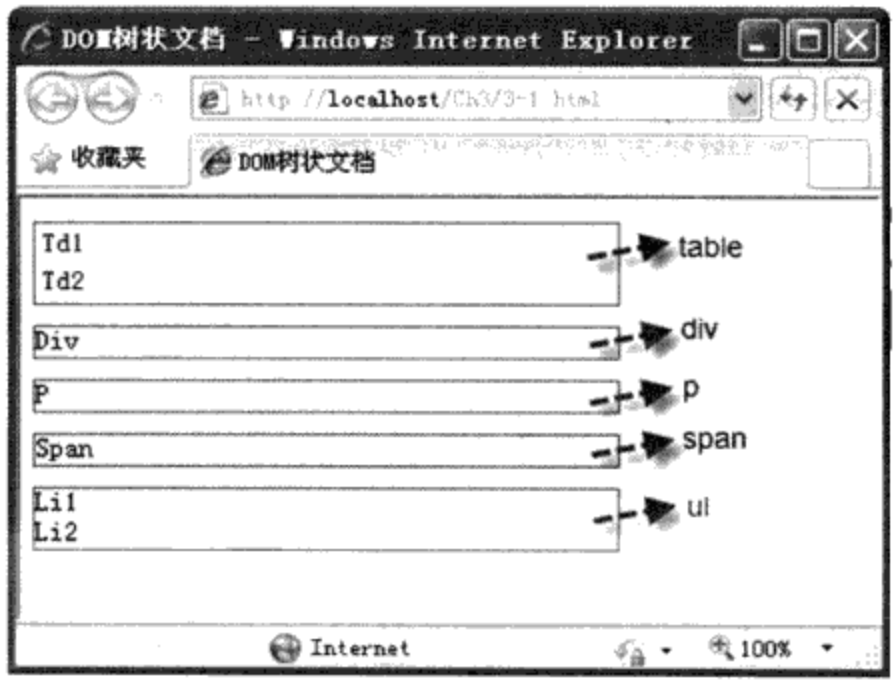


图 3-1　DOM 树状页面效果

根据上述页面文档创建出的 DOM 树结构如图 3-2 所示。

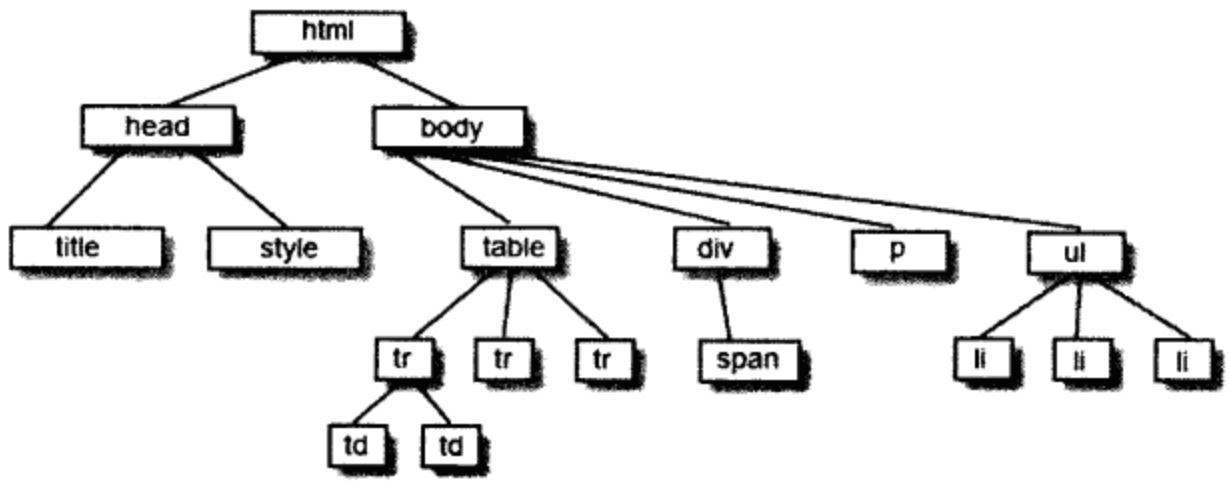


图 3-2　DOM 树状模型

3.2　访问元素

在访问页面时，需要与页面中的元素进行交互式的操作。在操作中，元素的访问是最频

繁、最常用的，主要包括对元素属性、内容、值、CSS 的操作。下面通过实例详细介绍这些操作的使用方法和技巧。

3.2.1　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。

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　通过 attr(name) 方法获取元素的属性

（1）功能描述

在一个页面中，创建一个 <img> 标记，通过 jQuery 中的 attr() 方法获取标记的 src 和 title 属性值，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 3-2.html, 加入如代码清单 3-2 所示的代码。

代码清单 3-2　获取元素的属性

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>获取元素的属性</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:12px}
          div{float:left;padding-left:10px}
          img{border:solid 1px #ccc;padding:3px;float:left}
    </style>
    <script type="text/javascript">
        $(function() {
            var strAlt = $("img").attr("src"); //属性值一
            strAlt += "<br/><br/>" +
            $("img").attr("title"); //属性值二
            $("#divAlt").html(strAlt);// 显示在页面中
        })
```

```
    </script>
</head>
<body>
     <img alt="" title="这是一幅风景画 "
          src="Images/img01.jpg" />
     <div id="divAlt"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-3 所示。

图 3-3　获取元素的属性

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})
```

下面将通过示例 3-3 来看看如何通过 attr(name,value) 的方式设置元素的属性。

示例 3-3　设置元素的属性（一）

（1）功能描述

在页面中，创建一个 <img> 标记，通过 jQuery 中的 attr() 方法，在页面加载时，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-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           .clsSpn{float:left;padding-top:10px;padding-left:10px}
           .clsImg{border:solid 1px #ccc;padding:3px;float:left}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").attr("src", "Images/img01.jpg");// 设置 src 属性
            $("img").attr("title", "这是一幅风景画");// 设置 title 属性
            $("img").attr({ src: "Images/img02.jpg",
                   title: "这是一幅风景画" });// 同时设置两个属性
            $("img").addClass("clsImg");// 增加样式
                $("span").html("加载完毕");// 显示加载状态
          })
      </script>
</head>
<body>
     <img alt="" src="Images/img03.gif" style="float:left" />
     <span class="clsSpn">正在加载图片 ...</span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-4 所示。

另外，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。

下面我们将通过示例 3-4 介绍在方法 attr() 中，通过 function 返回一个随机数作为该方法的参数，来设置元素的属性。

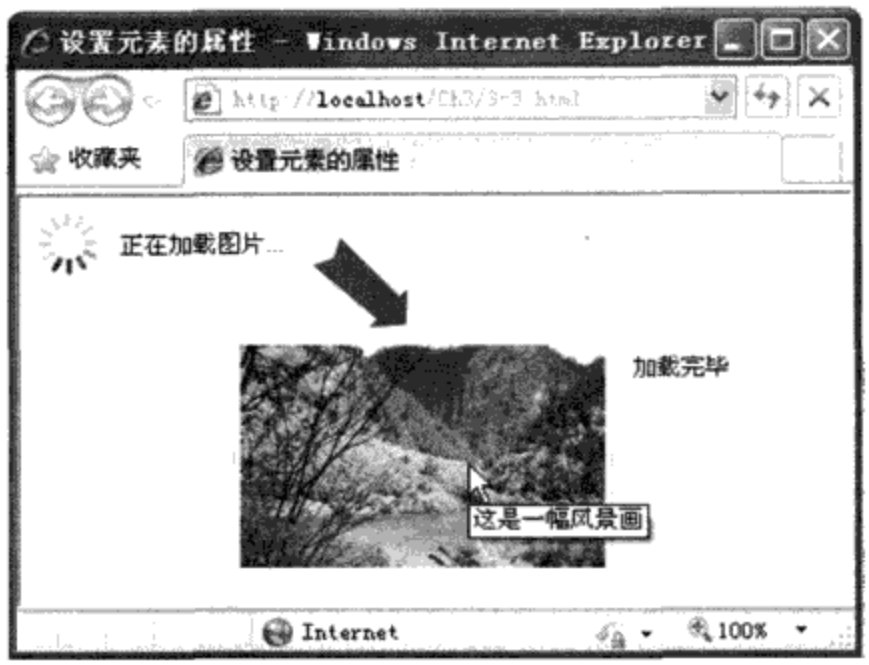


图 3-4　设置元素的属性

示例 3-4　设置元素的属性（二）

（1）功能描述

在页面中，通过 function() 随机函数返回一个随机数字，依据该数字组成一个字符串作为标记 <img> 属性 src 的值，由于是随机数字，因此每次获取的 src 属性值都不一样，从而实现在页面中随机展示图片的效果。

（2）实现代码

新建一个 HTML 文件 3-4.html，加入如代码清单 3-4 所示的代码。

代码清单 3-4　设置元素的属性（二）

```
<!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-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           .clsSpn{float:left;padding-top:10px;padding-left:10px}
           .clsImg{border:solid 1px #ccc;padding:3px;float:left}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").attr("src", function() {
            return "Images/img0" +
            Math.floor(Math.random() * 2 + 1) +
```

```
            ".jpg" }); // 设置 src 属性
            $("img").attr("title", "这是一幅风景画"); // 设置 title 属性
            $("img").addClass("clsImg"); // 增加样式
        })
    </script>
</head>
<body>
     <img alt="" src="Images/img03.gif" style="float:left" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-5 所示。

图 3-5　设置元素的属性（二）

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，可以通过如下的代码删除标记 <img> 中的 src 属性：

```
$("img").removeAttr("src");
```

3.2.2　元素内容操作

在 jQuery 中，操作元素内容的方法包括 html() 和 text()。前者与 JavaScript 中的 innerHTML 属性类似，即获取或设置元素的 HMTL 内容；后者类似于 JavaScript 中的 innerText 属性，即获取或设置元素的文本内容。二者的格式与功能的区别如表 3-1 所示。

表 3-1 html() 和 text() 方法的区别

语法格式	参数说明	功能描述
html()	无参数	用于获取元素的 HTML 内容
html(val)	val 参数为元素的 HTML 内容	用于设置元素的 HTML 内容
text()	无参数	用于获取元素的文本内容
text(val)	val 参数为元素的文本内容	用于设置元素的文本内容

说明 html() 方法仅支持 XHTML 的文档，不能用于 XML 文档，而 text() 则既支持 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，来设置或获取元素的内容。

示例 3-5 设置或获取元素的内容

（1）功能描述

在页面中，用 html() 和 text() 方法获取 div 标记中的内容，并将内容分别作为 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 设置或获取元素的内容

```
<!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-1.4.2.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　获取或设置元素的内容

3.2.3　获取或设置元素值

在 jQuery 中，如果要获取元素的值，是通过 val() 方法实现的，其语法格式如下所示：

```
val(val)
```

其中，如果不带参数 val，则是获取某元素的值；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　设置或获取元素的值

（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 标记将所获取的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　设置或获取元素的值

```
<!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-1.4.2.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。

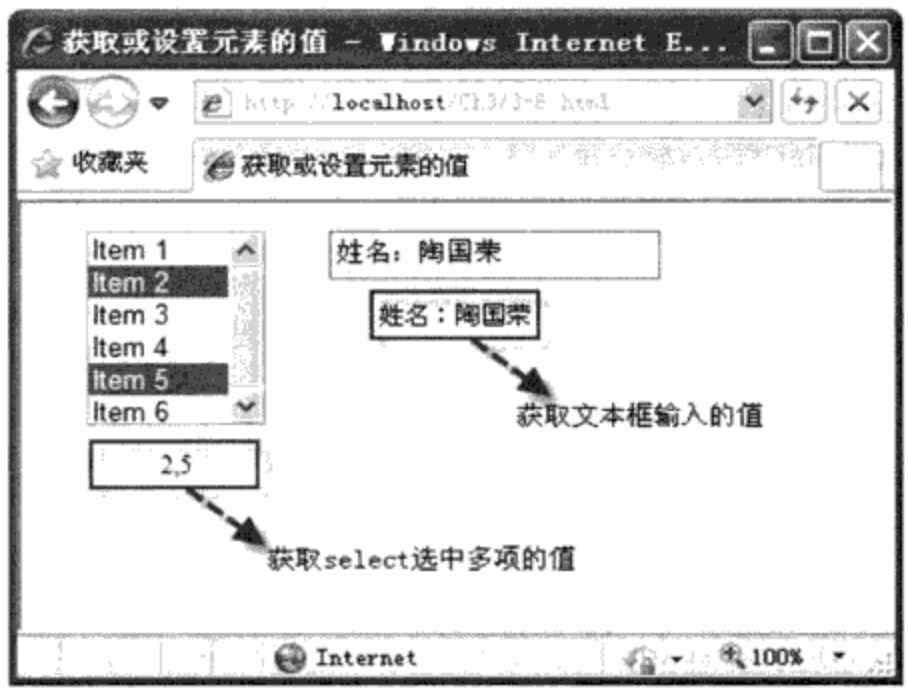


图 3-7　获取或设置元素的值

注意　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"]) 此句代码的意思为：ID 号为 radio2 和 radio3 的单选框被选中。

3.2.4　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。

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　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>直接设置元素样式值</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.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。

2. 增加 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 ...)
```

图 3-8　直接设置元素样式值

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　增加 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-1.4.2.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

注意　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. 类别切换

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 演示了如何通过调用 toggleClass() 方法单击元素时，切换元素 CSS 的类别。

示例 3-9　类别切换

（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 CSS 的类别

```
<!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-1.4.2.js">
    </script>
    <style type="text/css">
           .clsImg{border:solid 1px #666;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").click(function() {
                $(this).toggleClass("clsImg"); // 切换样式类别
            })
        })
    </script>
</head>
<body>
     <img src="Images/logo.gif" alt="" title="jQuery" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-10 所示。

4. 删除类别

与增加 CSS 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，则删除名称是 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");
```

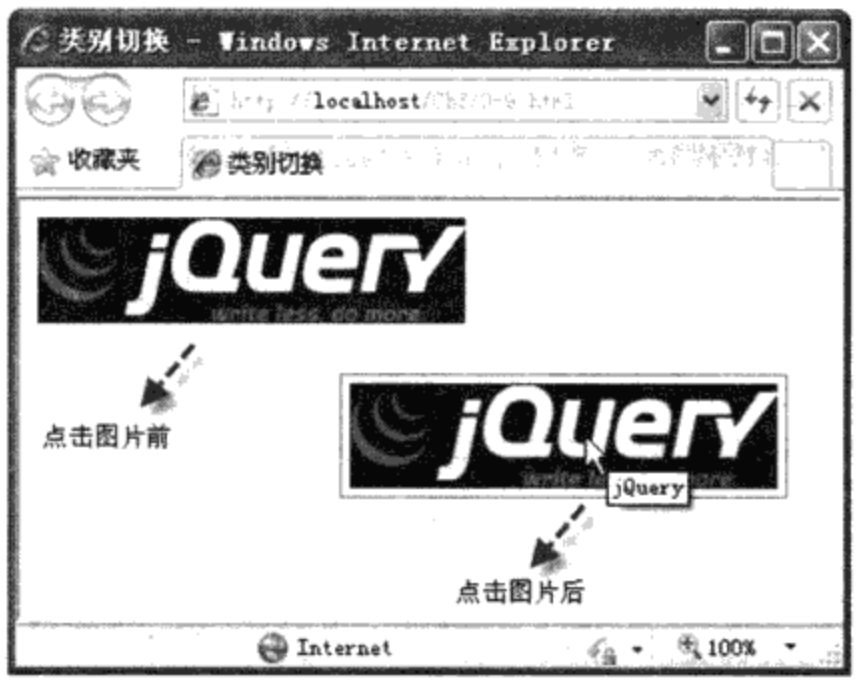


图 3-10　类别切换

注意　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。

如果要删除 p 标记是 cls0 和 cls1 的类别，则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，则可以使用如下的代码：

```
$("p").removeClass();
```

3.3　创建节点元素

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。

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　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-1.4.2.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。

3.4 插入节点

从第 3.3 节中我们知道，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插入方法。下面将分别对这些方法进行详细介绍。

3.4.1 内部插入节点方法

内部插入节点的方法如表 3-2 所示。

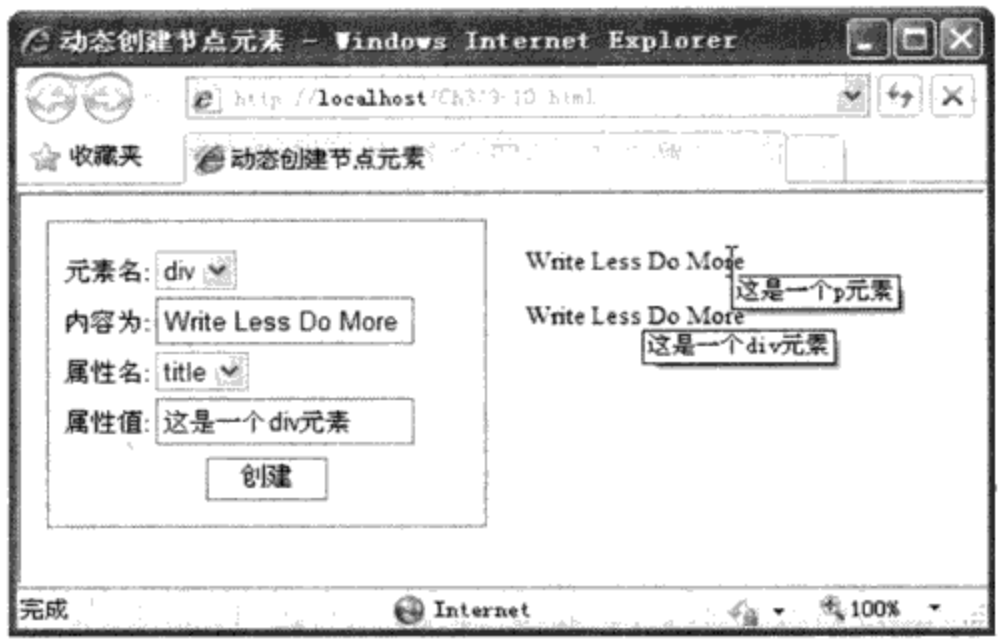


图 3-11　动态创建节点元素

注意　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。

表 3-2　内部插入节点

语法格式	参数说明	功能描述
append(content)	content 表示追加到目标中的内容	向所选择的元素内部插入内容
append(function(index, html))	通过 function 函数返回追加到目标中的内容	向所选择的元素内部插入 function 函数所返回的内容
appendTo(content)	content 表示被追加的内容	把所选择的元素追加到另一个指定的元素集合中
prepend(content)	content 表示插入目标元素内部前面的内容	向每个所选择的元素内部前置内容
prepend(function(index, html))	通过 function 函数返回插入目标元素内部前面的内容	向所选择的元素内部前置 function 函数所返回的内容
prependTo(content)	content 表示用于选择元素的 jQuery 表达式	将所选择的元素前置到另一个指定的元素集合中

下面结合示例介绍其中几个最新的节点插入方法。

1. append(function(index, html))

该方法是版本 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 演示了如何通过调用 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 插入节点（一）

（1）功能说明

在页面中，通过一个 funtion 函数返回一段字符串，并将该字符串插入指定的 div 标记元素内容中。

（2）实现代码

新建一个 HTML 文件 3-11.html, 加入如代码清单 3-11 所示的代码。

代码清单 3-11 插入元素节点

```
<!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>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.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，即如果要将 span 标记插入 div 标记中，则执行下列代码：

```
$("span").appentTo($("div"));
```

即把 appentTo 方法前部分的内容插入其后部分的内容中。

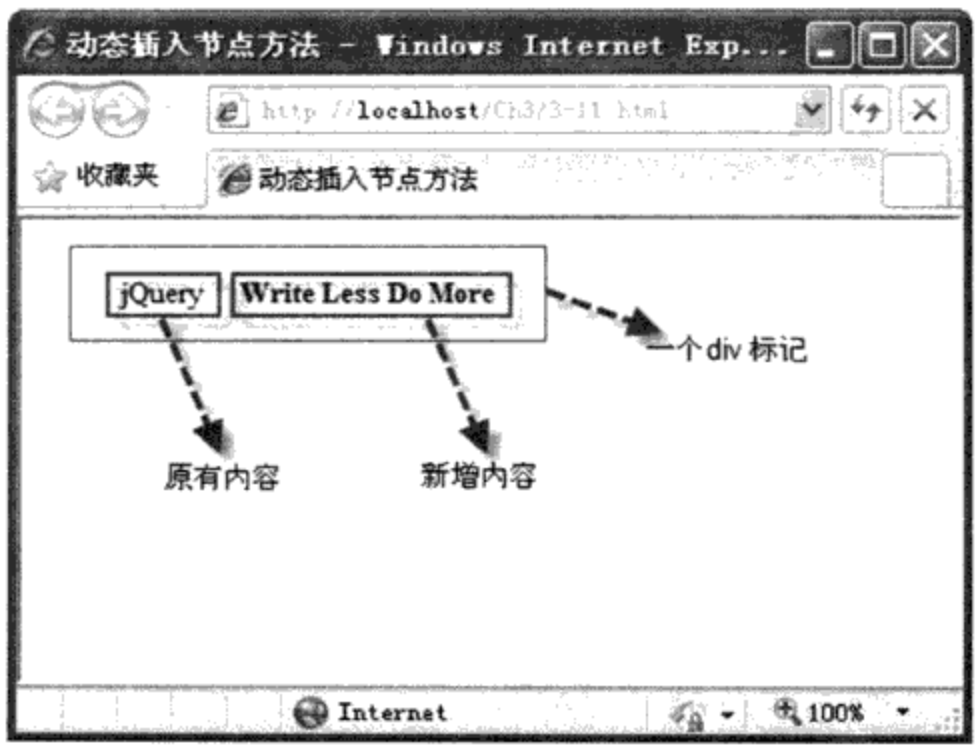


图 3-12　插入节点（一）

示例 3-12 演示了调用 appentTo() 方法，将一个元素标记插入另一个标记中的过程。

示例 3-12　插入节点（二）

（1）功能描述

在一个页面中，创建一个 img 和两个 span 标记，通过 appentTo() 方法将 img 标记插入 span 标记中。

（2）实现代码

新建一个 HTML 文件 3-12.html, 加入如代码清单 3-12 所示的代码。

代码清单 3-12　将一个元素的内容动态插入另一个元素中

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{border:solid 1px #ccc;padding:3px;margin:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").appendTo($("span")); // 插入内容
        })
```

```
    </script>
</head>
<body>
     <img title="2008 年新书封面 " src="Images/img03.jpg" />
     <span><img title="2010 年新书封面 "
                src="Images/img04.jpg" /></span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-13 所示。

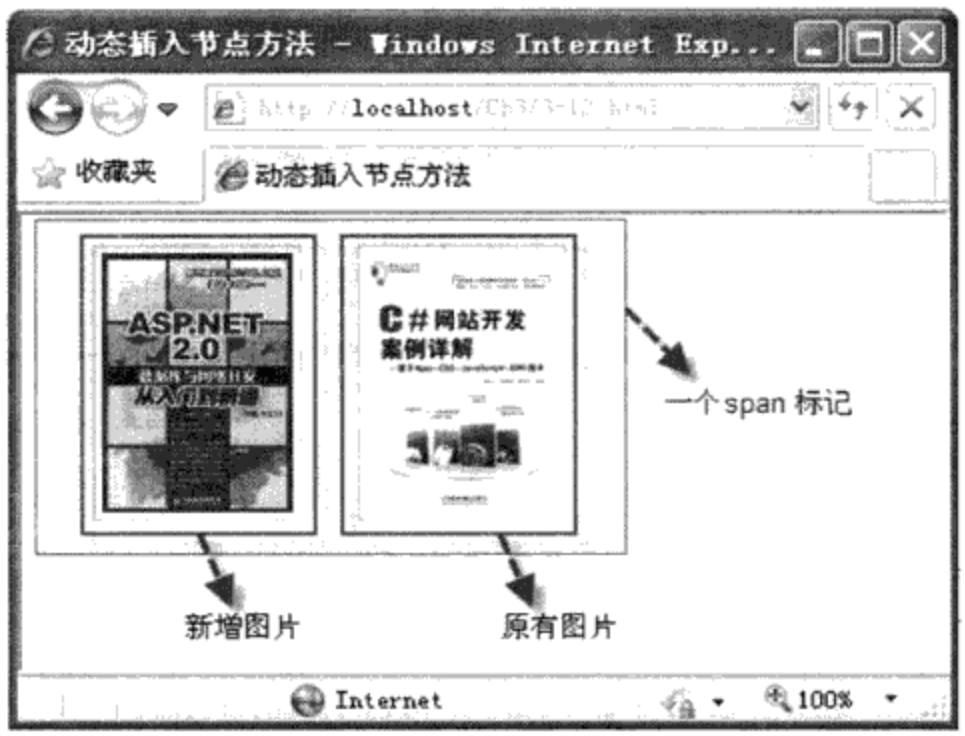


图 3-13　插入节点（二）

3.4.2　外部插入节点方法

外部插入节点的方法如表 3-3 所示。

表 3-3　外部插入节点

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

同样，下面通过示例介绍其中几个节点插入方法的使用技巧。

after(function)

该方法也是 jQuery1.4 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　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 function 函数返回值至元素节点

```
<!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>动态插入节点方法</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("span").after(retHtml); //插入内容
            function retHtml() {
                var str = "<span><b>Write Less Do More</b><span>";
                return str;
            }
        })
    </script>
</head>
<body>
     <span>jQuery</span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-14 所示。

图 3-14　外部插入节点

3.5　复制节点

在页面中，有时需要将某个元素节点复制到另外一个节点后，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　复制元素节点

（1）功能描述

在页面中，创建一个 img 标记，显示一幅图片；当单击该图片时，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-1.4.2.js">
    </script>
    <style type="text/css">
           img{border:solid 1px #ccc;padding:3px;margin:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").click(function() {
                $(this).clone(true).appendTo("span");
            })
        })
    </script>
</head>
<body>
    <span><img title="封面" src="Images/img04.jpg" /></span>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-15 所示。

图 3-15　复制元素节点

注意 由于本示例中使用的是 clone(true) 方法，因此，当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。

3.6 替换节点

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 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-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           span{font-weight:bold}
           p{background-color:#eee;padding:5px;width:200px}
    </style>
```

```
    <script type="text/javascript">
        $(function() {
            $("#Span1").replaceWith("<span
            title='replaceWith'>陶国荣 </span>");
            $("<span title='replaceAll'>
            tao_guo_rong@163.com</span>").replaceAll("#Span2");
        })
    </script>
</head>
<body>
    <p>姓名：<span id="Span1"></span></p>
    <p>邮箱：<span id="Span2"></span></p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-16 所示。

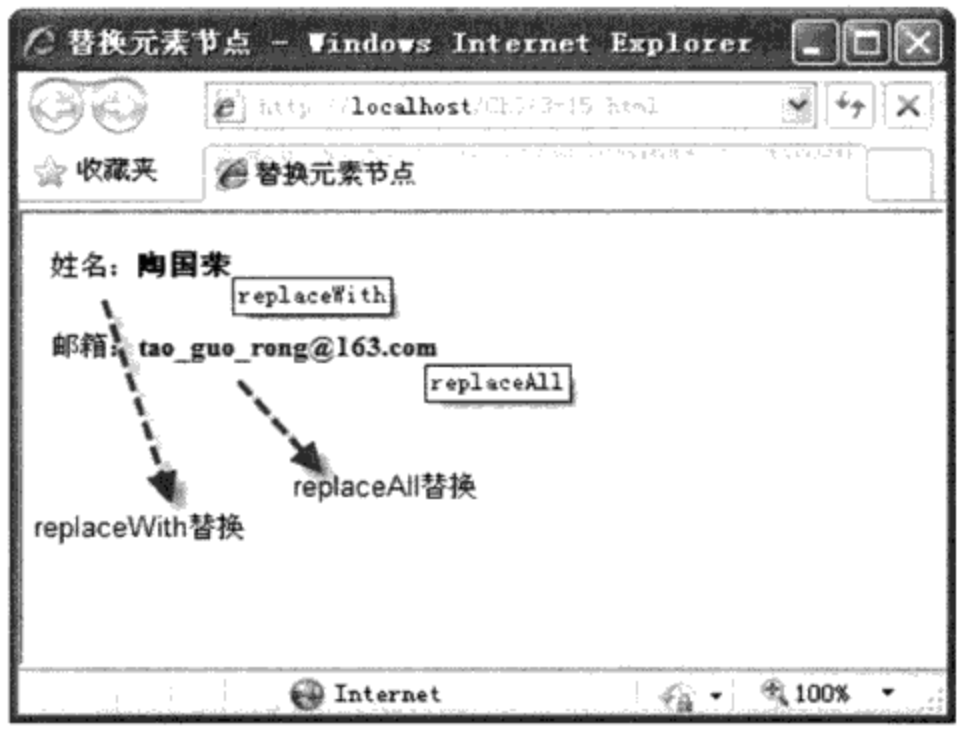


图 3-16　替换元素节点

注意　replaceWith() 与 replaceAll() 方法都可以实现元素节点的替换，二者最大的区别在于替换字符的顺序，前者是用括号中的字符替换所选择的元素，后者是用字符串替换括号中所选择的元素。同时，一旦完成替换，被替换元素中的全部事件都将消失。

3.7　包裹节点

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 包裹节点

语法格式	参数说明	功能描述
wrap(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素用其他字符串代码包裹起来
wrap(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素用其他 DOM 元素包装起来
wrap(fn)	fn 参数为包裹结构的一个函数	把所有选择的元素用 function 函数返回的代码包裹起来
unwrap()	无参数	移除所选元素的父元素或包裹标记
wrapAll(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素用单个元素包裹起来
wrapAll(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素用单个 DOM 元素包裹起来
wrapInner(html)	html 参数为字符串代码，用于生成元素并包裹所选元素	把所有选择的元素的子内容（包括文本节点）用字符串代码包裹起来
wrapInner(elem)	elem 参数用于包装所选元素的 DOM 元素	把所有选择的元素的子内容（包括文本节点）用 DOM 元素包裹起来
wrapInner(fn)	fn 参数为包裹结构的一个函数	把所有选择的元素的子内容（包括文本节点）用 function 函数返回的代码包裹起来

在上述表格中，wrap(html) 与 wrapInner(html) 方法较为常用，前者包裹外部元素，后者包裹元素内部的文本字符。下面通过示例 3-16 介绍这两种常用方法在页面中包裹目标元素的使用效果。

示例 3-16 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-1.4.2.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。

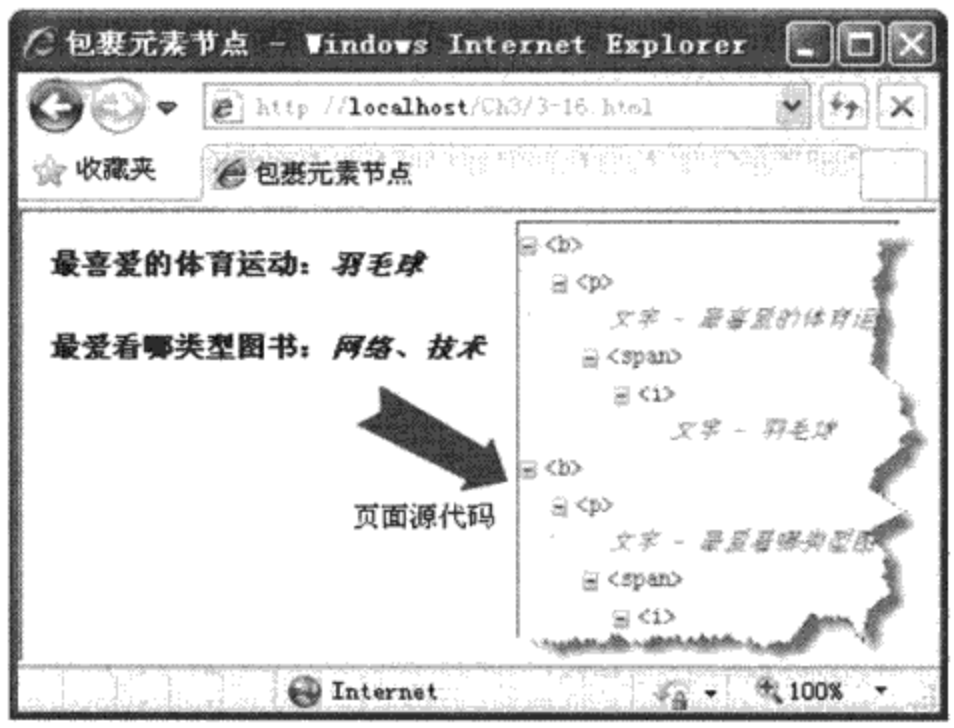


图 3-17　包裹元素节点的页面效果和源码

3.8　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　遍历元素

（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-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2-vsdoc.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{border:solid 1px #ccc;
           padding:3px;margin:5px;width:143px;height:101px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("img").each(function(index) {
                // 根据形参 index 设置元素的 title 属性
                this.title = "第 " + index +
                "幅风景图片，alt 内容是 " + this.alt;
            })
        })
    </script>
</head>
<body>
   <p>
   <img src="Images/img05.jpg" alt=" 第 0 幅风景画 " />
   <img src="Images/img06.jpg" alt=" 第 1 幅风景画 " />
   <img src="Images/img07.jpg" alt=" 第 2 幅风景画 " /></p>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-18 所示。

图 3-18　遍历元素

3.9　删除元素

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 提供了两种可以删除元素的方法，即 remove() 和 empty()。严格来说，empty() 方法并非真正意义上的删除，使用该方法，仅仅可以清空全部的节点或节点所包括的所有后代元素，并非删除节点和元素。

remove() 方法的语法格式如下：

```
remove([expr])
```

其中参数 expr 为可选项，如果接受参数，则该参数为筛选元素的 jQuery 表达式，通过该表述式获取指定的元素，并进行删除。

empty () 方法的语法格式如下：

```
empty()
```

其功能为清空所选择的页面元素或所有的后代元素。

示例 3-18 说明了调用 remove() 方法，删除某个页面元素的的过程。

示例 3-18　删除元素

（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-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           ul{width:200px}
           ul li{ list-style:none;padding:0px;height:23px}
           span{padding-left:20px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
                 filter: progid:DXImageTransform.Microsoft.
                 Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $("ul li:first").css("font-weight", "bold");// 设置首行
            $("#Button1").click(function() {
                $("ul li").remove("li[title=t]");// 删除指定属性的元素
                $("ul li:eq(1)").remove();// 删除节点中第 2 个元素
            })
        })
    </script>
</head>
<body>
     <ul>
         <li>学号 </li>
         <li title="t">1001</li>
         <li>1002</li>
         <li>1003</li>
         <li style="text-align:center;padding-top:5px">
                <input id="Button1" type="button"
                       value=" 删除 " class="btn" />
         </li>
     </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 3-19 所示。

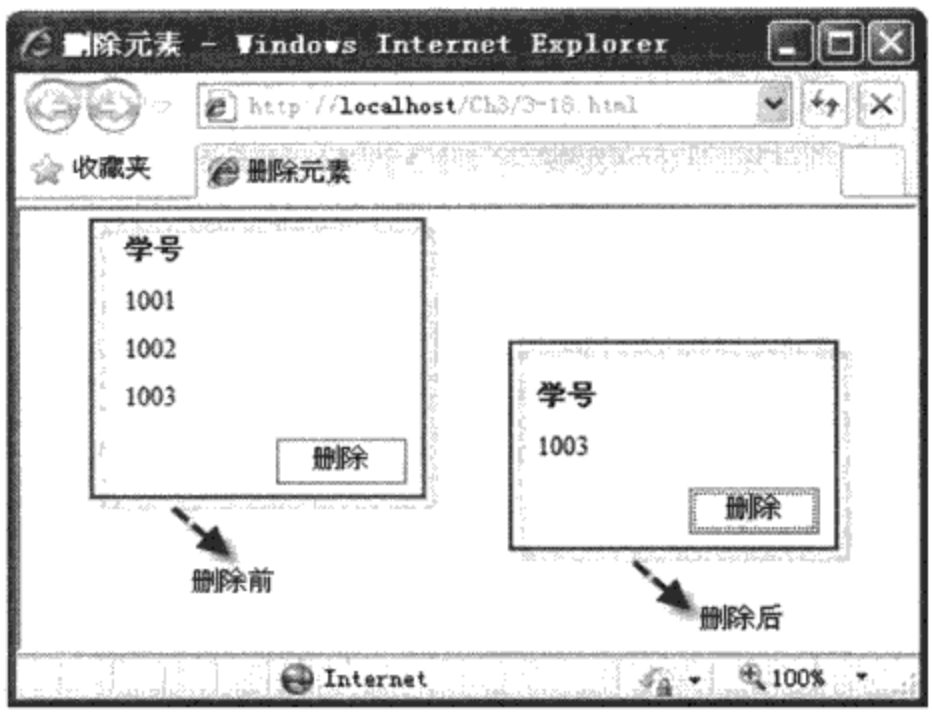


图 3-19　删除元素

注意　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”被删除后，第 2 项元素就是“1002”，故仅留下学号为“1003”。

3.10　综合案例分析——数据删除和图片预览在项目中的应用

3.10.1　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。

3.10.2　效果界面

页面初始化状态，使用隔行变色展示数据，其效果如图 3-20 所示。

图 3-20　隔行变色展示数据

当选择某行中的“选项”复选框后，那么单击“删除”按钮时，将删除其选中的行数据，其效果如图 3-21 所示。

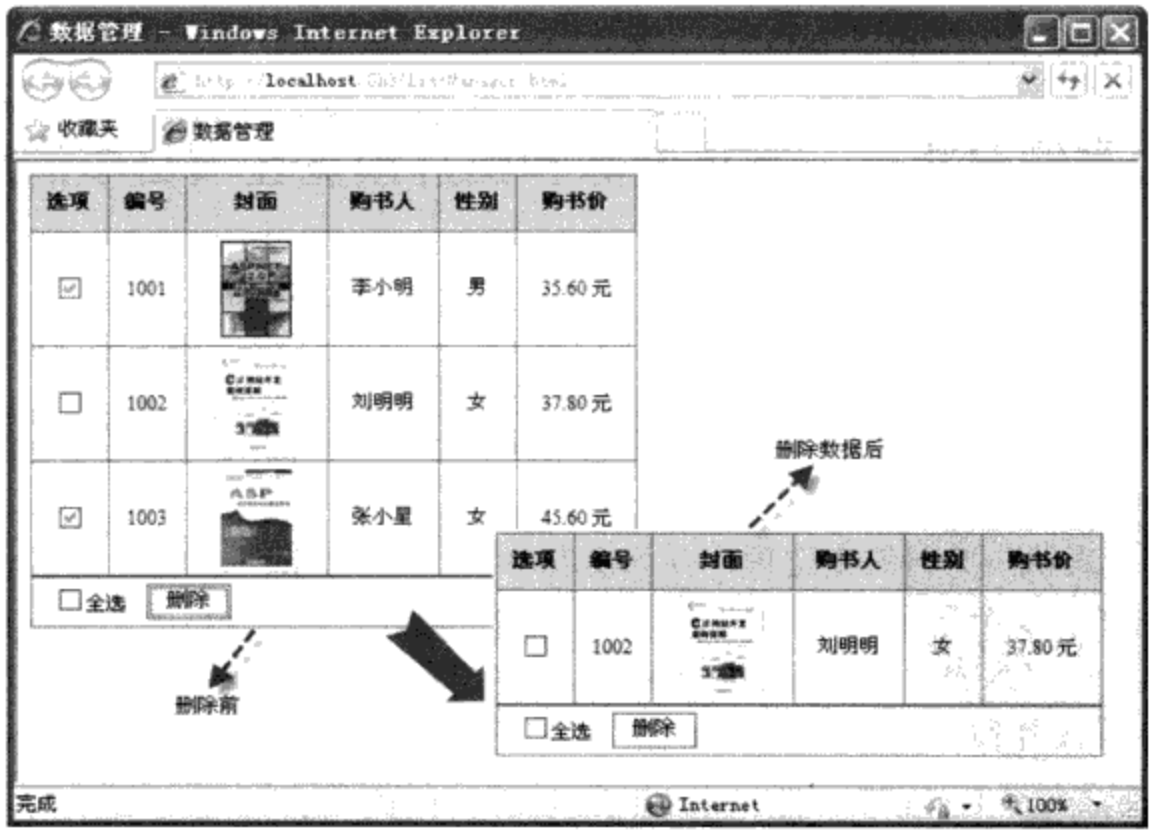


图 3-21　删除数据的前后对比

当将鼠标移到“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

3.10.3　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-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:12px}
           table{width:360px;border-collapse:collapse}
           table tr th,td{border:solid 1px #666;text-align:center}
           table tr td img{border:solid 1px #ccc;
                 padding:3px;width:42px;height:60px;cursor:hand}
           table tr td span{float:left;padding-left:12px;}
           table tr th{background-color:#ccc;height:32px}
           .clsImg{position:absolute;border:solid 1px #ccc;
```

```
            padding:3px;width:85px;height:120px;
            background-color:#eee;display:none}
        .btn {border:#666 1px solid;padding:2px;width:50px;
            filter: progid:DXImageTransform.Microsoft
            .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
</style>
<script type="text/javascript" >
    $(function() {
    $("table tr:nth-child(odd)")
    .css("background-color", "#eee"); // 隔行变色

        /** 全选复选框单击事件 **/
        $("#chkAll").click(function() {
            if (this.checked) {// 如果自己被选中
                $("table tr td input[type=checkbox]")
                .attr("checked", true);
            }
            else {// 如果自己没有被选中
                $("table tr td input[type=checkbox]")
                .attr("checked", false);
            }
        })

        /** 删除按钮单击事件 **/
        $("#btnDel").click(function() {
         var intL = $("table tr td
         input:checked
         :not('#chkAll')").length; // 获取除全选复选框外的所有选中项
            if (intL != 0) {// 如果有选中项
                $("table tr td
                input[type=checkbox]
                :not('#chkAll')")
                .each(function(index) {// 遍历除全选复选框外的行
                    if (this.checked) {// 如果选中
                        $("table tr[id=" + this.value
                + "]").remove(); // 获取选中的值，并删除该值所在的行
                    }
                })
            }
})

/** 小图片鼠标移动事件 **/
var x = 5; var y = 15;// 初始化提示图片位置
$("table tr td img").mousemove(function(e) {
    $("#imgTip")
    .attr("src", this.src)// 设置提示图片 scr 属性
    .css({ "top": (e.pageY + y) + "px",
    "left": (e.pageX + x) + "px" })// 设置提示图片的位置
    .show(3000);// 显示图片
```

```
            })

            /** 小图片鼠标移出事件 **/
            $("table tr td img").mouseout(function() {
                $("#imgTip").hide();// 隐藏图片
            })

        })
    </script>
</head>
<body>
    <table>
        <tr>
            <th>选项 </th>
            <th>编号 </th>
            <th>封面 </th>
            <th>购书人 </th>
            <th>性别 </th>
            <th>购书价 </th>
        </tr>
        <tr id="0">
            <td><input id="Checkbox1" type="checkbox"
                       value="0"/></td>
            <td>1001</td>
            <td><img src="Images/img03.jpg" alt="" /></td>
            <td> 李小明 </td>
            <td> 男 </td>
            <td>35.60 元 </td>
        </tr>
        <tr id="1">
            <td><input id="Checkbox2" type="checkbox"
                       value="1"/></td>
            <td>1002</td>
            <td><img src="Images/img04.jpg" alt="" /></td>
            <td> 刘明明 </td>
            <td> 女 </td>
            <td>37.80 元 </td>
        </tr>
        <tr id="2">
            <td><input id="Checkbox3" type="checkbox"
                       value="2"/></td>
            <td>1003</td>
            <td><img src="Images/img08.jpg" alt="" /></td>
            <td> 张小星 </td>
            <td> 女 </td>
            <td>45.60 元 </td>
        </tr>
    </table>
    <table>
        <tr>
```

```
            <td style="text-align:left;height:28px">
                <span><input id="chkAll"
                            type="checkbox" /> 全选 </span>
                <span><input id="btnDel" type="button" value=" 删除 "
                            class="btn" /></span>
            </td>
          </tr>
        </table>
        <img id="imgTip" class="clsImg" src="Images/img03.gif"/>
</body>
</html>
```

3.10.4 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
 }
```

值得注意的是：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，则隐藏该提示图片，代码如下：

```
$("#imgTip").hide();// 隐藏图片
```

3.11　本章小结

在页面中，控制元素是 DOM 对象主要的行为，本章通过介绍 jQuery 中大量访问或设置页面元素的方法，其主要目的是使读者了解如何通过 jQuery 中的方法完全操控页面，编写出功能更方便、更高效的页面代码，并为下一章节的学习奠定坚实的语法基础。

第 4 章

jQuery 中的事件与应用

本章内容

- 事件机制
- 页面载入事件
- 绑定事件
- 切换事件
- 移除事件
- 其他事件
- 表单应用
- 列表应用
- 网页选项卡的应用
- 综合案例分析——删除数据时的提示效果在项目中的应用
- 本章小结

当用户浏览页面时，浏览器会对页面代码进行解释或编译——这个过程实质上是通过事件来驱动的，即页面在加载时，执行一个 Load 事件，在这个事件中实现浏览器编译页面代码的过程。事件无论在页面元素本身还是在元素与人机交互中，都占有十分重要的地位。

4.1　事件机制

众所周知，页面在加载时，会触发 Load 事件。当用户单击某个按钮时，触发该按钮的 Click 事件，通过种种事件实现各项功能或执行某项操作。事件在元素对象与功能代码中起着重要的桥梁作用。那么，事件被触发后是如何执行代码的呢？

严格来说，事件在触发后被分为两个阶段，一个是捕获（Capture），另一个则是冒泡（Bubbling）；但有些遗憾的是，大多数浏览器并不是都支持捕获阶段，jQuery 也不支持。因此在事件触发后，往往执行冒泡过程。所谓的冒泡其实质就是事件执行中的顺序。下面通过示例 4-1 来说明事件执行过程中的冒泡现象。

示例 4-1　事件中的冒泡现象

（1）功能描述

在页面中创建一个 标记，并在该标记中创建一个按钮。当用户单击页面或者 标记或者按钮时，都在页面中显示一句问候语“您好，欢迎来到 jQuery 世界！”。

（2）实现代码

新建一个 HTML 文件 4-1.html, 加入如代码清单 4-1 所示的代码。

代码清单 4-1　事件中的冒泡现象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>事件中的冒泡现象</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .clsShow{border:#ccc 1px solid;background-color:#eee;
               margin-top:15px;padding:5px;width:220px;
               line-height:1.8em;display:none}
           .btn {border:#666 1px solid;padding:2px;width:50px;
               filter: progid:DXImageTransform.Microsoft
               .Gradient(GradientType=0,StartColorStr=#ffffff,
               EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
```

```
        $(function() {
            var intI = 0;// 记录执行次数
            $("body,div,#btnShow").click(function() {// 单击事件
                intI++;// 次数累加
                $(".clsShow")
                .show()// 显示
                .html("您好，欢迎来到 jQuery 世界！")// 设置内容
                .append("<div> 执行次数 <b>" + intI
                + "</b> </div>");// 追加文本
            })
        })
    </script>
</head>
<body>
    <div>
        <input id="btnShow" type="button" value=" 点击 "
               class="btn" />
    </div>
    <div class="clsShow"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-1 所示。

图 4-1　事件中的冒泡现象

为什么在示例 4-1 中，仅是单击按钮，但却执行了 3 次同样的程序呢？这是因为事件在执行过程中存在冒泡现象，即虽然单击的是按钮，但按钮的外围 元素的事件也被触发，同时 元素的外围 <body> 元素的事件也随之被触发，其整个事件波及的过程就像水泡一样向外冒，故称为冒泡过程。在示例 4-1 中，按钮被单击后，事件的执行过程如图 4-2 所示。

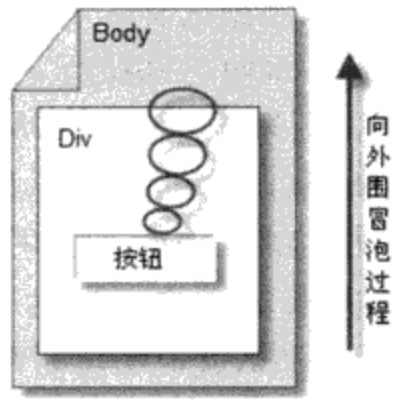


图 4-2　冒泡过程

通过示意图 4-2，我们很清楚地看出，当单击了按钮时，事件在冒泡过程中先后又执行了元素 和元素 <body> 中的事件，因此尽管执行的是按钮中的单击事件，但实际又触发了其他两个事件，所以共有 3 次事件的执行过程。

而在实际需要中，我们并不希望事件的冒泡现象发生，即单击按钮就执行单一的单击事件，并不触发其他外围的事件。在 jQuery 中，可以通过 stopPropagation() 方法来实现，你会发现该方法可以阻止冒泡过程的发生。我们将示例 4-1 中的 jQuery 代码进行部分修改，如下所示：

```
$(function() {
    var intI = 0; // 记录执行次数
    $("body,div,#btnShow").click(function(event) {// 单击事件
          intI++; // 次数累加
          $(".clsShow")
          .show()// 显示
          .html(" 您好，欢迎来到 jQuery 世界 !")// 设置内容
          .append("<div> 执行次数 <b>" + intI + "</b> </div>"); // 追加文本
          event.stopPropagation();// 阻止冒泡过程
    })
})
```

修改完成后，执行示例 4-1，单击按钮后，其页面显示的执行次数为 1 ，即仅是触发了按钮的单击事件，而没有波及其他外围元素事件。

说明　在代码中，除了使用 stopPropagation() 方法阻止事件的冒泡过程外，还可以通过语句 return false 实现停止事件的冒泡过程。

4.2　页面载入事件

4.2.1　ready() 方法的工作原理

在前面的第 1 章节中，我们简单地介绍了 jQuery 中的页面载入事件 ready() 方法。该方法类似于传统 JavaScript 中的 OnLoad() 方法，只不过在事件执行时间上有区别：OnLoad() 方法

的执行必须是页面中的全部元素完全加载到浏览器后才触发，在这种情况下，如果页面中的图片过多或图片过大，那么有可能要等 OnLoad() 方法执行完毕，用户才能进行其他的操作；如果使用 jQuery 中的 ready() 方法加载页面，则只要页面的 DOM 模型加载完毕，就会触发 ready() 方法。因此，两者在事件的执行效果上 ready() 方法明显优于 JavaScript 中的 OnLoad() 方法。

我们解剖一下 jQuery 中 ready() 方法的工作原理：在 jQuery 脚本加载到页面时，会设置一个 isReady 的标记，用于监听页面加载的进度，当然遇到执行 ready() 方法时，通过查看 isReady 值是否被设置，如果未被设置，那么就说明页面并未加载完成，在此情况下，将未完成的部分用一个数组缓存起来，当全部加载完成后，再将未完成的部分通过缓存一一执行。

4.2.2 ready() 方法的几种相同写法

以下几种代码写法其执行的效果是一样的。

写法一：

```
$(document).ready(function(){
    // 代码部分
})
```

写法二：

```
$(function(){
    // 代码部分
})
```

写法三：

```
jQuery(document).ready(function(){
    // 代码部分
})
```

写法四：

```
jQuery(function(){
    // 代码部分
})
```

其中写法二简洁明了，使用较为广泛。

4.3 绑定事件

我们在示例 4-1 中，使用如下的代码绑定按钮的单击事件：

```
$(function(){
    $("#btnShow").click(function(){
        // 执行代码
    })
})
```

除了上述绑定事件的写法外，在 jQuery 中，还可以使用 bind() 方法进行事件的绑定。下面对该方法进行详细的介绍。

bind() 功能是为每个选择元素的事件绑定处理函数，其语法格式如下：

```
bind(type, [data], fn)
```

其中参数 type 为一个或多个类型的字符串，如“click" 或“change”，也可以自定义类型；可以被参数 type 调用的类型包括 blur、focus、load、resize、scroll、unload、click、dblclick、mousedown、mouseup、mousemove、mouseover、mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup、error。

参数 data 是作为 event.data 属性值传递给事件对象的额外数据对象。

参数 fn 是绑定到每个选择元素的事件中的处理函数。

下面通过两个例子来讲解绑定方法。

示例 4-2　用 bind 方法绑定事件

（1）功能描述

在页面中，设置一个按钮，通过 bind 方法给按钮绑定一个 click 事件，在该事件中，将自身的是否可用属性设置成 false，即单击按钮后就不可使用。

（2）实现代码

新建一个 HTML 文件 4-2.html, 加入如清单 4-2 所示的代码。

代码清单 4-2　bind 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>bind 方法绑定事件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           .btn {border:#666 1px solid;padding:2px;width:50px;
                 filter: progid:DXImageTransform.Microsoft
                 .Gradient(GradientType=0,StartColorStr=#ffffff,
                 EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#btnBind").bind("click", function() {
                $(this).attr("disabled", "disabled");// 按钮不可用
            })
        })
    </script>
```

```
</head>
<body>
     <input id="btnBind" type="button" value="Button"
           class="btn" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-3 所示。

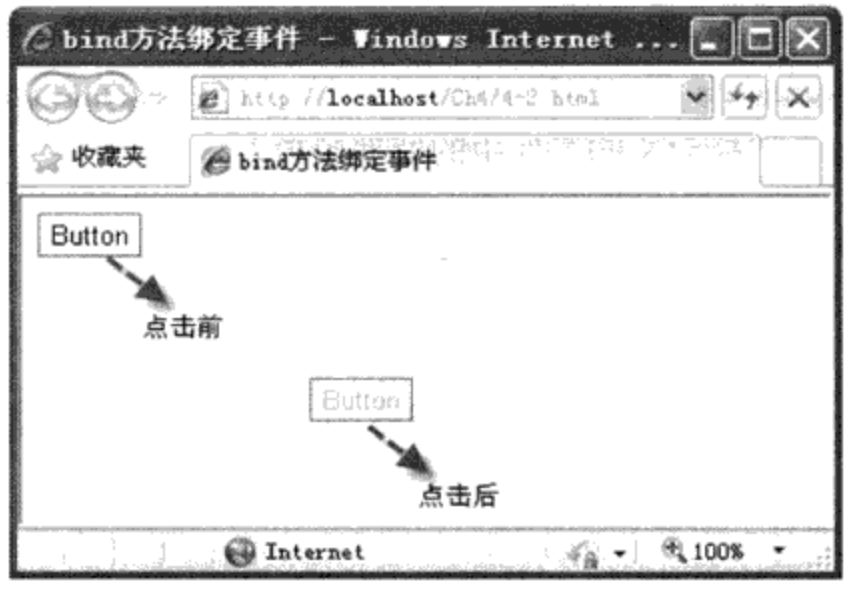


图 4-3　bind 方法绑定事件

如果要在一个元素中绑定多个事件，可以将事件用空格隔开，如示例 4-2 中，除单击事件外，如果需要加一个 mouseout 事件，则可以将代码修改为如下所示：

```
... 省略头部代码
$(function() {
    $("#btnBind").bind("click mouseout", function() {
       $(this).attr("disabled", "disabled"); // 按钮不可用
    })
})
... 省略主体代码
```

在 jQuery 绑定事件时，还可以通过传入一个映射，对所选对象绑定多个事件处理函数，见下面的示例。

示例 4-3　用映射方式绑定不同的事件

（1）功能描述

在页面中，设置一个文本框，通过映射的方式，给文本框绑定两个事件，一个是 focus 事件，另一个是 change 事件。这两个事件执行时，均为显示事件的名称。

（2）实现代码

新建一个 HTML 文件 4-3.html, 加入如清单 4-3 所示的代码。

代码清单 4-3　映射方式绑定不同的事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>映射方式绑定不同的事件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .clsTip{border:#ccc 1px solid;
           background-color:#eee;margin-top:15px;
           padding:5px;width:185px;display:none}
           .txt{border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".txt").bind({ focus: function() {
                $("#divTip")
                .show()// 显示
                .html("执行的是focus事件");// 设置文本
            },
                change: function() {
                    $("#divTip")
                    .show()// 显示
                    .html("执行的是change事件");// 设置文本
                    }
                })
            })
        </script>
    </head>
    <body>
         <div>姓名：<input type="text" class="txt" /></div>
         <div id="divTip" class="clsTip"></div>
    </body>
    </html>
```

（3）页面效果

代码执行后的效果如图 4-4 所示。

在方法 bind 中，第二个参数 data 为可选项，表示作为 event.data 属性值传递到事件对象的额外数据对象，实际上，该参数很少使用，如果使用，那么可以通过该参数将一些附加的信息传递给事件处理函数 fn 中，在示例 4-3 中，将代码中的 bind 方法修改成带 data 参数，其修改后的代码如下所示：

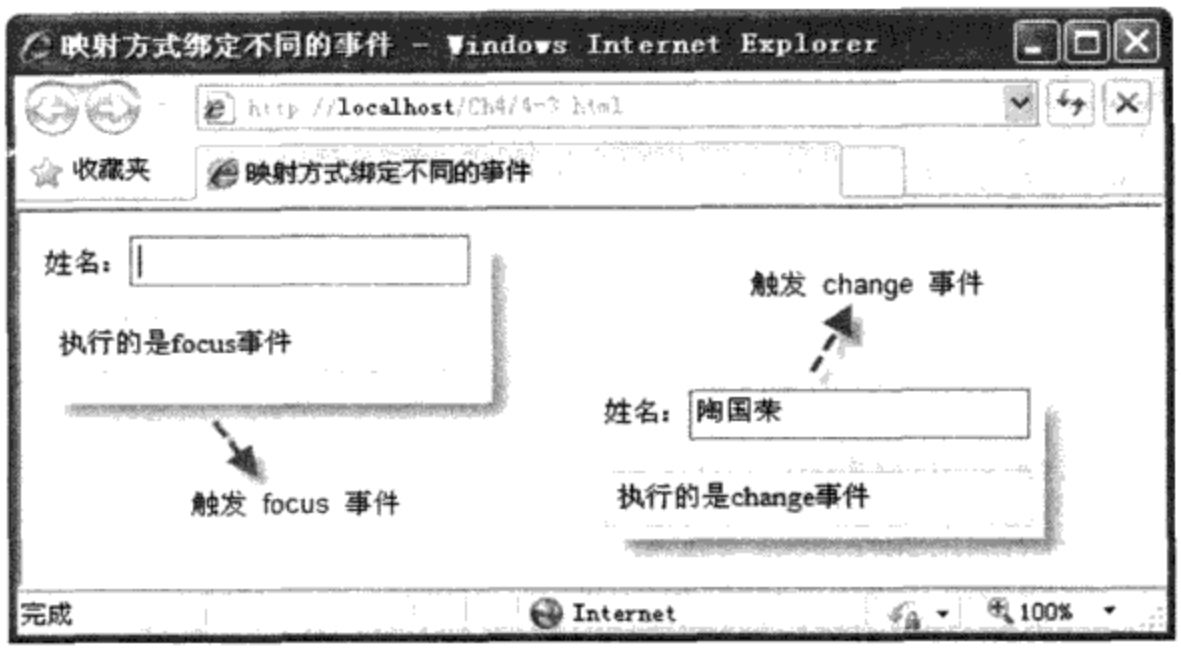


图 4-4　映射方式绑定不同的事件

```
... 省略头部代码
$(function() {
    var message = "执行的是 focus 事件";
    $(".txt").bind("focus", { msg: message }, function(event) {
        $("#divTip")
        .show()// 显示
        .html(event.data.msg); // 设置文本
      });
    message = "执行的是 change 事件";
    $('.txt').bind('change', { msg: message }, function(event) {
        $("#divTip")
        .show()// 显示
        .html(event.data.msg); // 设置文本
      });
})
... 省略主体代码
```

代码执行后，页面效果与图 4-4 一模一样。

4.4　切换事件

在 jQuery 中，有两个方法用于事件的切换，一个是方法 hover()，另一个是方法 toggle()。所谓切换事件，即有两个以上的事件绑定于一个元素，在元素的行为动作间进行切换。如一个超级链接标记 <a>，若想实现当鼠标悬停时触发一个事件，鼠标移出时又触发另一个事件，就可以调用 jQuery 中的 hover() 方法轻松实现。

4.4.1　hover() 方法

调用 jQuery 中的 hover() 方法可以使元素在鼠标悬停与鼠标移出的事件中进行切换，该方

法在实现运用中，也可以通过 jQuery 中的事件 mouseenter 与 mouseleave 进行替换。下列代码是等价的。

代码一：

```
$("a").hover(function() {
    // 执行代码一
  }, function() {
    // 执行代码二
})
```

代码二：

```
$("a").mouseenter(function() {
    // 执行代码一
 })
$("a").mouseleave(function() {
    // 执行代码二
 })
```

hover() 功能是当鼠标移动到所选的元素上面时，执行指定的第一个函数；当鼠标移出这个元素时，执行指定的第二个函数，其语法格式如下：

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

参数 over 为鼠标移到元素时触发的函数，参数 out 为鼠标移出元素时触发的函数。下面举例说明。

示例 4-4　用 hover 方法绑定事件

（1）功能描述

在页面中，创建一个 标记，在该标记中，设置另外两个 元素，一个用于显示标题，另一个则用于显示内容，调用 jQuery 中的 hover() 方法。将鼠标移到标题时，自动显示内容；鼠标移出标题时，关闭显示的内容。

（2）实现代码

新建一个 HTML 文件 4-4.html, 加入如清单 4-4 所示的代码。

代码清单 4-4　hover 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>切换事件 hover</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
```

```
            body{font-size:13px}
            .clsFrame{border:solid 1px #666;width:220px}
            .clsFrame .clsTitle{background-color:#eee;
            padding:5px;font-weight:bold}
            .clsFrame .clsContent{padding:5px;display:none}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".clsTitle").hover(function() {
                $(".clsContent").show();
            }, function() {
                $(".clsContent").hide();
            })
        })
    </script>
</head>
<body>
     <div class="clsFrame">
          <div class="clsTitle">jQuery 简介 </div>
          <div class="clsContent">  jQuery 是由美国人 John Resig 于 2006 年创
          建的一个开源项目，它的主旨是：以更少的代码，实现更多的功能（Write less, do more）。
          </div>
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-5 所示。

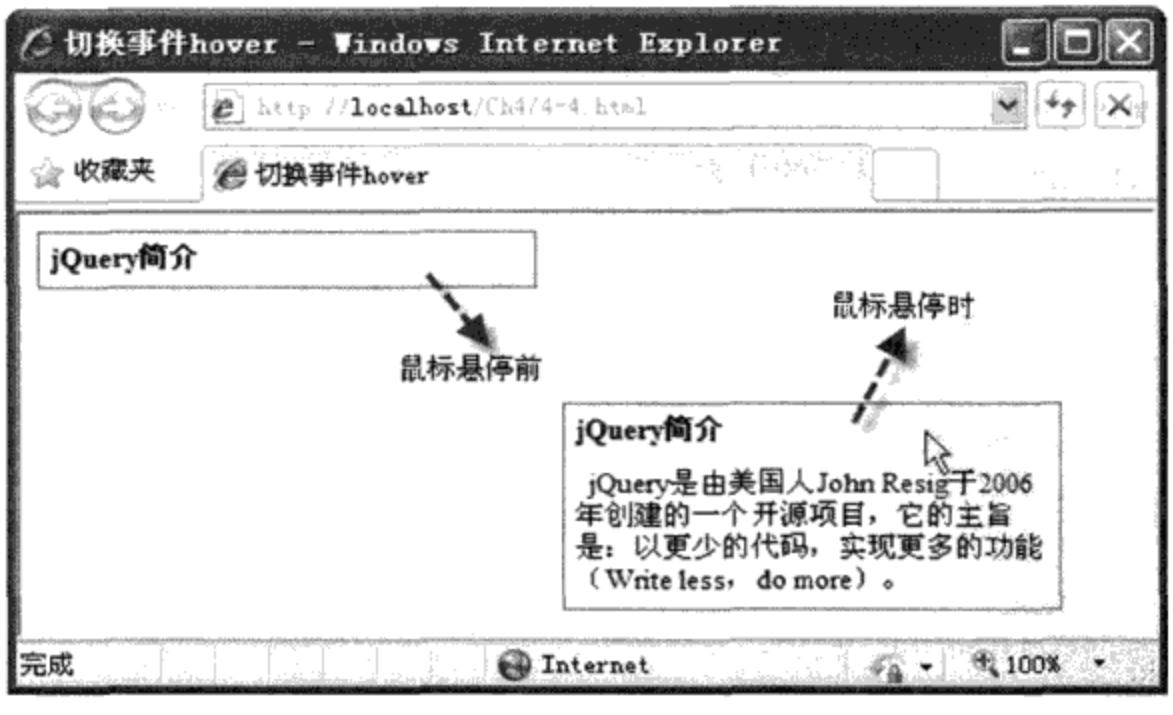


图 4-5　hover 方法绑定事件

4.4.2　toggle() 方法

在 toggle() 方法中，可以依次调用 N 个指定的函数，直到最后一个函数，然后重复对这些函数轮番调用。

toggle() 方法的功能是每次单击后依次调用函数，请注意到“依次”这两个字，说明该方法在调用函数时并非随机或指定调用，而是通过函数设置的前后顺序进行调用，其调用的语法格式如下：

```
toggle(fn, fn2, [fn3, fn4, ...])
```

其中参数 fn，fn2，...，fnN 为单击时被依次调用的函数。

示例 4-5　用 toggle 方法绑定事件

（1）*功能描述*

在页面中，设置一个 <img> 标记，当用户第一次单击该图片时，变换另外一幅图片；第二次单击图片时，变换第三幅图片；第三次单击时，返回第一次单击前的图片；依次轮番显示。

（2）*实现代码*

新建一个 HTML 文件 4-5.html, 加入如清单 4-5 所示的代码。

代码清单 4-5　toggle 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>切换事件 toggle</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{border:solid 1px #ccc;padding:3px}
   </style>
    <script type="text/javascript">
        $(function() {
            $("img").toggle(function() {
                $("img").attr("src", "Images/img05.jpg");
                $("img").attr("title", this.src);
            }, function() {
                $("img").attr("src", "Images/img06.jpg");
                $("img").attr("title", this.src);
            }, function() {
                $("img").attr("src", "Images/img07.jpg");
                $("img").attr("title", this.src);
```

```
            })
        })
    </script>
</head>
<body>
    <img />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-6 所示。

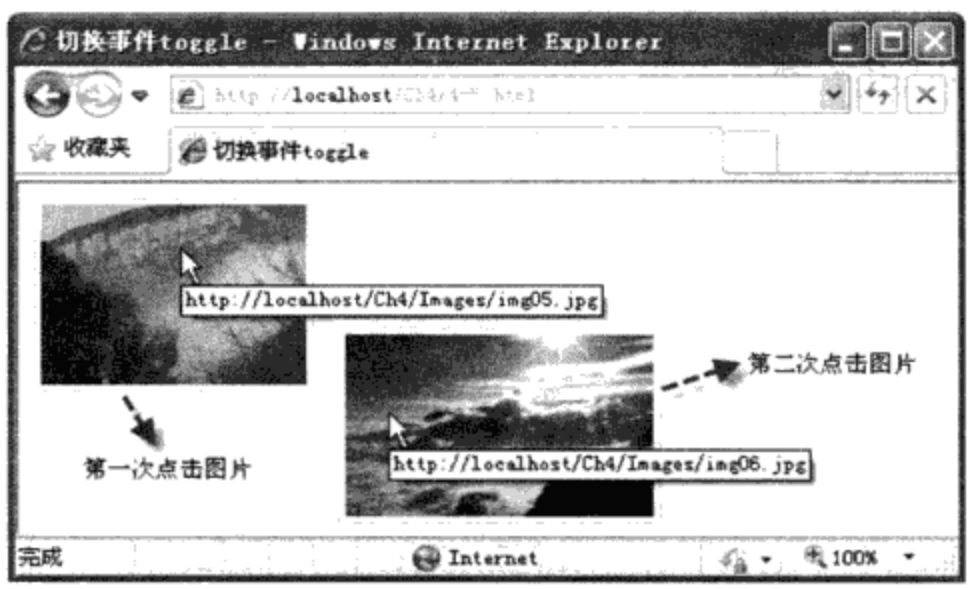


图 4-6　toggle 方法绑定事件

4.5　移除事件

在 DOM 对象的实践操作中，既然存在用于绑定事件的 bind 方法，也相应存在用于移除绑定事件的方法，在 jQuery 中，可以通过 unbind() 方法移除绑定的所有事件或指定某一个事件。

unbind() 的功能是移除元素绑定的事件，其调用的语法格式如下：

```
unbind([type], [fn])
```

其中，参数 type 为移除的事件类型，fn 为需要移除的事件处理函数；如果该方法没有参数，则移除所有绑定的事件；如果带有参数 type，则移除该参数所指定的事件类型；如果带有参数 fn，则只移除绑定时指定的函数 fn。下面举例说明。

示例 4-6　用 unbind 方法移除事件

（1）功能描述

在页面中，设置三个按钮，前两个按钮分别执行各自的事件，第三个按钮通过 unbind 方

法移除所绑定的全部事件。即点击第三个按钮后，前两个按钮的事件将不会执行。

（2）实现代码

新建一个 HTML 文件 4-6.html, 加入如清单 4-6 所示的代码。

代码清单 4-6　unbind 方法移除事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>移除事件 unbind</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
           div{line-height:1.8em}
    </style>
    <script type="text/javascript">
        $(function() {
            function oClick() { // 自定义事件
             $("#divTip").append("<div>按钮二的单击事件 </div>");
            }
           $("input:eq(0)").bind("click", function() { // 绑定单击事件
           $("#divTip").append("<div>按钮一的单击事件 </div>");
            });
           $("input:eq(1)").bind("click", oClick); // 绑定自定义事件
           $("input:eq(2)").bind("click",
                  function() { // 移除全部单击事件
                  $("input").unbind();
            });
         })
    </script>
</head>
<body>
     <div>
         <input id="Button1" type="button"
                value=" 按钮一 " class="btn" />
         <input id="Button2" type="button"
                value=" 按钮二 " class="btn"/>
         <input id="Button3" type="button"
                value=" 删除事件 " class="btn"/>
     </div>
     <div id="divTip" style="padding-top:10px"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-7 所示。

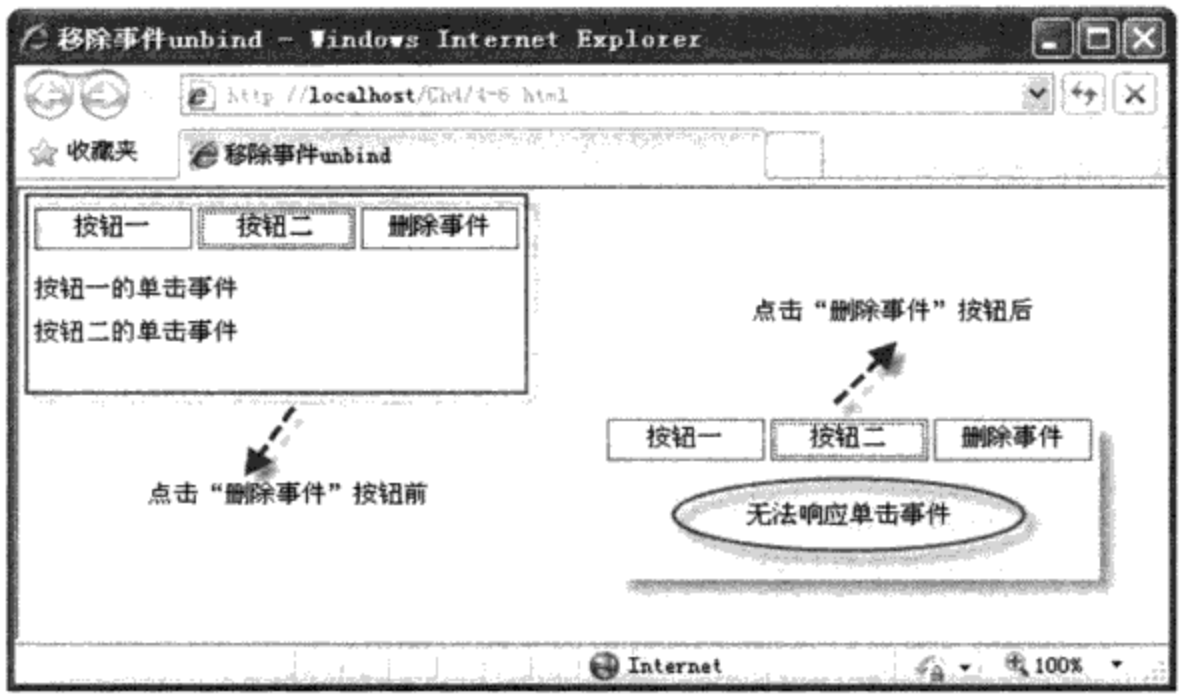


图 4-7　unbind 方法移除事件

通过语法介绍我们知道，unbind() 方法不仅可以删除某类型的全部事件，还可删除某个指定的自定义事件，如在代码清单 4-6 中，如果将下列代码：

```
...省略前部分
$("input:eq(2)").bind("click", function() { //移除全部单击事件
    $("input").unbind();
 });
...省略后部分
```

修改成下列代码：

```
...省略前部分
$("input:eq(2)").bind("click", function() { //移除全部单击事件
    $("input").unbind("click",oClick);
 });
...省略后部分
```

那么，当单击“删除事件”按钮后，单击“按钮一”将有对应事件的响应，而单击“按钮二”将无对应事件的响应，因为在“删除事件”的按钮中，移除的仅仅是“按钮二”的自定义事件，因此，只有该按钮没有事件的响应。

4.6　其他事件

除上述介绍的几种事件方法外，在 jQuery 中还有很多的事件处理方法，下面介绍其中较为实用的两种处理事件的方法 one() 和 trigger()。

4.6.1　方法 one()

one() 方法功能是为所选的元素绑定一个仅触发一次的处理函数，其调用的语法格式为：

```
one(type, [data], fn)
```

其中参数 type 为事件类型，即需要触发什么类型的事件；参数 data 为可选参数，表示作为 event.data 属性值传递给事件对象的额外数据对象；fn 为绑定事件时所要触发的函数。下面举例说明。

示例 4-7　用 one 方法绑定事件

（1）*功能描述*

在页面中，设置一个按钮，初始值为“点击查看联系方式”，当单击该按钮时，通过 jQuery 中的 one() 方法，将联系方式显示在按钮上，再次单击则不响应任何事件。

（2）*实现代码*

新建一个 HTML 文件 4-7.html, 加入如代码清单 4-7 所示的代码。

代码清单 4-7　one 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>其他事件 one</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
            .btn {border:#666 1px solid;padding:2px;width:160px;
            filter: progid:DXImageTransform.Microsoft
            .Gradient(GradientType=0,StartColorStr=#ffffff,
             EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            function btn_Click() { // 自定义事件
                this.value = "010-12345678"
            }
            $("input").bind("click", btn_Click); // 绑定自定义事件
        })
    </script>
</head>
<body>
    <input id="Button1" type="button"
           value="点击查看联系方式" class="btn" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-8 所示。

图 4-8　one 方法绑定事件

4.6.2　方法 trigger ()

在前端页面开发中，有时希望页面在 DOM 加载完毕后，自动执行一些很人性化的操作，如：文本框中的内容处于全部被选中状态，某个按钮处于焦点中。利用传统的 JavaScript 语言需要编写复杂的代码才可实现上述功能；而在 jQuery 中，仅需要调用一个 trigger() 方法就可以轻松实现。

trigger() 方法的功能是在所选择的元素上触发指定类型的事件。其调用的语法格式为：

```
trigger(type, [data])
```

其中参数 type 为触发事件的类型，参数 data 为可选项，表示在触发事件时，传递给函数的附加参数。下面举例说明。

示例 4-8　用 trigger 方法绑定事件

（1）功能描述

在页面中，创建一个文本框，并给文本框设置一个默认值，当该页面加载时，自动获取文本框中的值，显示在页面中，同时，文本框处于选中状态。

（2）实现代码

新建一个 HTML 文件 4-8.html, 加入如清单 4-8 所示的代码。

代码清单 4-8　trigger 方法绑定事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
    <title>其他事件 trigger</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
            body{font-size:13px}
           .txt{border:#666 1px solid;padding:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            var oTxt = $("input"); // 获取文本框
            oTxt.trigger("select"); // 自动选中文本框
            oTxt.bind("btn_Click", function() {// 编写文本框自定义事件
                var txt = $(this).val(); // 获取自身内容
                $("#divTip").html(txt); // 显示在页面中
            })
            oTxt.trigger("btn_Click"); // 自动触发自定义事件
        })
    </script>
</head>
<body>
    姓名：<input id="Text1" type="text" class="txt"
              value="陶国荣" />
         <div id="divTip" style="padding-top:5px"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-9 所示。

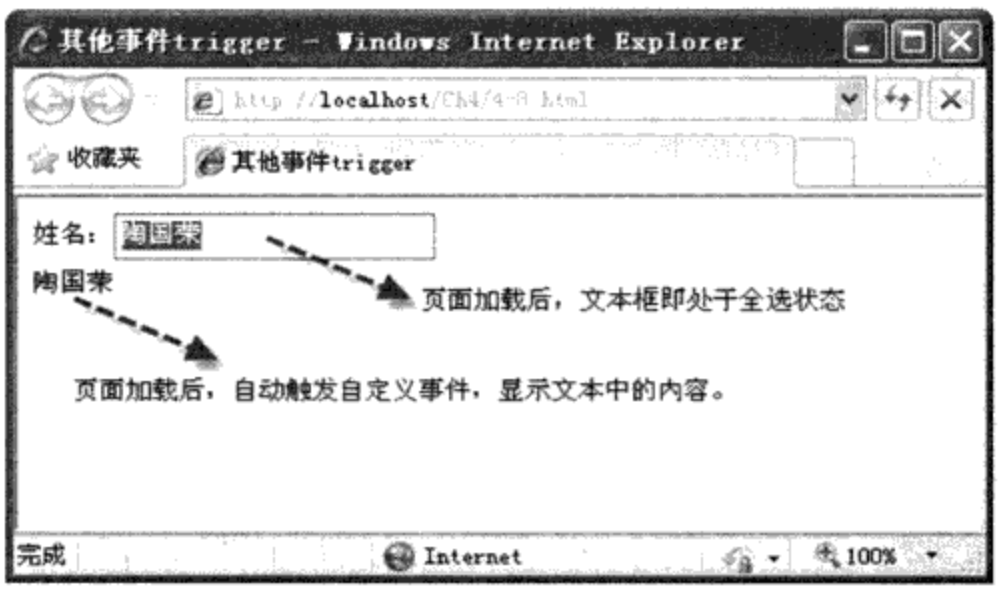


图 4-9　trigger 方法绑定事件

说明　trigger() 方法可以实现触发性事件，即不必用户做任何动作，自动执行该方法中的事件。在这种情形下，其最终效果可能会有异样发生。如果不希望页面自动执行，可使用 triggerHandler() 方法，使用方法与 trigger() 方法基本相同，只是该方法不会自动执行其包含的事件。

4.7 表单应用

我们在前面的章节中介绍 jQuery 选择器时，有专门用于表单的选择器，可见表单在 jQuery 中占有十分重要的地位，同时，表单在 HTML 中处理数据的优势也是不言而喻的，因此，在 jQuery 事件中，表单的应用也变得尤为重要。

4.7.1 文本框中的事件应用

文本框是表单中使用最为普遍的元素之一，因此，其前端用户页面的体验度显得十分重要。下面通过一个简单示例，介绍使用 jQuery 中的事件改变文本框的样式，以提高用户体验。

示例 4-9 文本框中的事件应用

（1）功能描述

在页面中，创建一个用于输入邮箱地址的文本框，使用 jQuery 与 CSS 结合，当用文本框获取时，样式发生变化，同时提示用户输入邮箱的方法。

当用户在文本框中输入邮箱后，丢失焦点时，将检测其内容是否为空，如果不为空或邮箱格式不符，样式将再次发生变化，同时提示出错信息；如果输入正确，样式将返回初始状态，并显示一个打勾的图片。

（2）实现代码

新建一个 HTML 文件 4-9.html, 加入如清单 4-9 所示的代码。

代码清单 4-9 文本框中的事件应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>文本框中的事件应用</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
            body{font-size:13px}
            /* 元素初始状态样式 */
            .divInit{width:390px;height:55px;line-height:55px;
            padding-left:20px}
            .txtInit{border:#666 1px solid;padding:3px;
            background-image:url('Images/bg_email_input.gif')}
            .spnInit{width:179px;height:40px;line-height:40px;
            float:right;margin-top:8px;padding-left:10px;
            background-repeat:no-repeat}

            /* 元素丢失焦点样式 */
```

```
        .divBlur{background-color:#FEEEC2}
        .txtBlur{border:#666 1px solid;padding:3px;
        background-image:url('Images/bg_email_input2.gif')}
        .spnBlur{background-image:
        url('Images/bg_email_wrong.gif')}

        .divFocu{background-color:#EDFFD5}/* div 获取焦点样式 */
        .spnSucc{background-image:
        url('Images/pic_Email_ok.gif');
        margin-top:20px}/* 验证成功时 span 样式 */
</style>
<script type="text/javascript">
    $(function() {
        $("#txtEmail").trigger("focus");// 默认时文本框获取焦点
        $("#txtEmail").focus(function() {// 文本框获取焦点事件
            $(this).removeClass("txtBlur")
            .addClass("txtInit");
            $("#email").removeClass("divBlur")
            .addClass("divFocu");
            $("#spnTip").removeClass("spnBlur")
            .removeClass("spnSucc")
            .html(" 请输入您常用邮箱地址！ ");
        })

        $("#txtEmail").blur(function() {// 文本框丢失焦点事件
            var vtxt = $("#txtEmail").val();
            if (vtxt.length == 0) {
                $(this).removeClass("txtInit")
                .addClass("txtBlur");
                $("#email").removeClass("divFocu")
                .addClass("divBlur");
                $("#spnTip").addClass("spnBlur")
                .html(" 邮箱地址不能为空！ ");
            }
            else {
                if (!chkEmail(vtxt)) {// 检测邮箱格式是否正确
                    $(this).removeClass("txtInit")
                    .addClass("txtBlur");
                    $("#email").removeClass("divFocu")
                    .addClass("divBlur");
                    $("#spnTip").addClass("spnBlur")
                    .html(" 邮箱格式不正确！ ");
                }
                else {// 如果正确
                    $(this).removeClass("txtBlur")
                    .addClass("txtInit");
                    $("#email").removeClass("divFocu");
                    $("#spnTip").removeClass("spnBlur")
                    .addClass("spnSucc").html("");
                }
```

```
                }
            })
            /*
             * 验证邮箱格式是否正确
             * 参数 strEmail，需要验证的邮箱
            */
            function chkEmail(strEmail) {
                if (!/^\w+([-+.]\w+)*@\w
                    +([-.]\w+)*\.\w+([-.]\w+)*$/
                    .test(strEmail)) {
                    return false;
                }
                else {
                    return true;
                }
            }
        })
    </script>
</head>
<body>
    <form id="form1" action="#">
        <div id="email" class="divInit">邮箱：
            <span id="spnTip" class="spnInit"></span>
            <input id="txtEmail" type="text" class="txtInit" />
        </div>
    </form>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-10 所示。

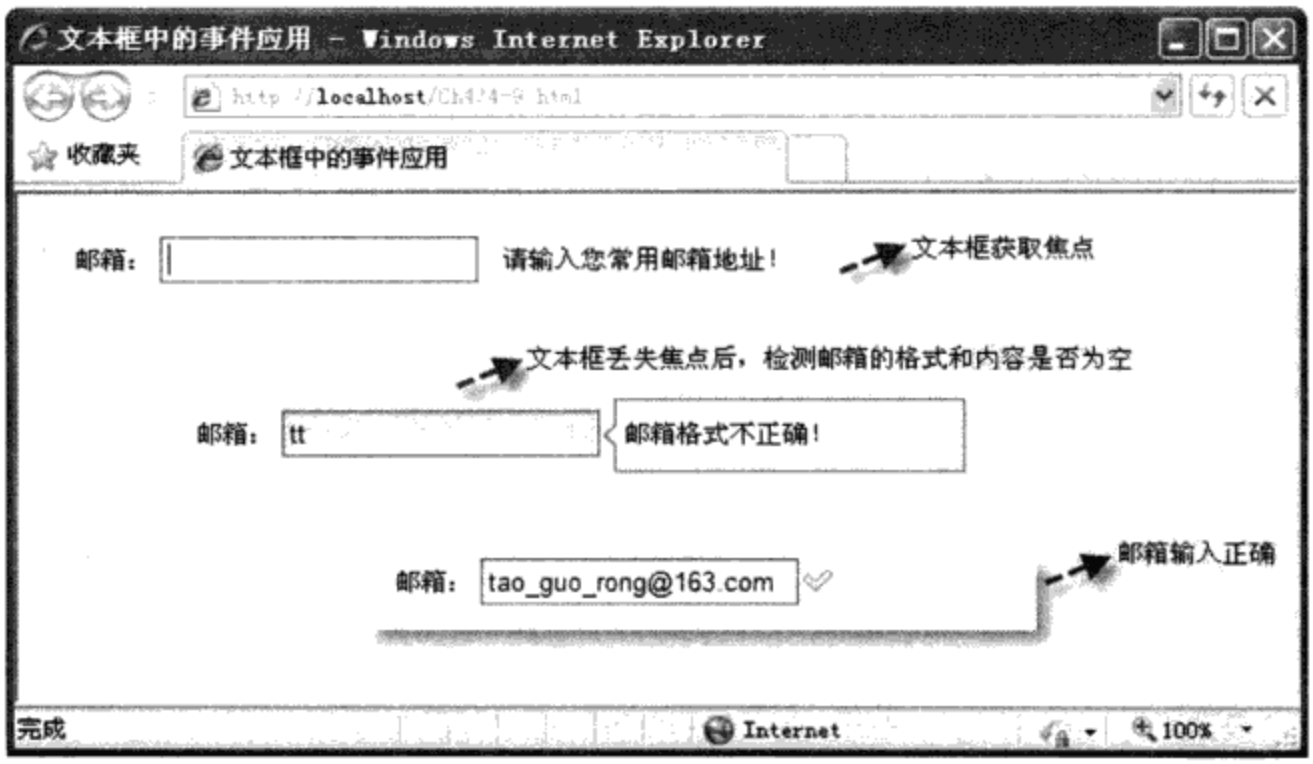


图 4-10　文本框用于输入邮箱时的不同状态

（4）代码分析

在代码中，为了提升用户页面体验度，我们针对文本框编写了两个事件，一个是 focus 事件，另一个是 blur 事件。在 focus 事件中，我们加入了如下的代码：

```
$("#txtEmail").focus(function() {// 文本框获取焦点事件
    $(this).removeClass("txtBlur")
    .addClass("txtInit");
    $("#email").removeClass("divBlur")
    .addClass("divFocu");
    $("#spnTip").removeClass("spnBlur")
    .removeClass("spnSucc")
    .html(" 请输入您常用邮箱地址！ ");
  })
```

在文本框中的 focus 获取焦点事件代码中，涉及三个元素的样式发生变化。一个是文本框自身，用 this 表示，在获取焦点时，有可能来源于丢失焦点，因此，首先删除丢弃焦点的样式，然后再加入获取焦点的样式，即使用下列代码：

```
$(this).removeClass("txtBlur")
.addClass("txtInit");
```

第二个元素是外层 Div 区域，同理也是先删除丢弃焦点的样式，然后再加入获取焦点的样式，其代码如下：

```
$("#email").removeClass("divBlur")
.addClass("divFocu");
```

第三个元素是用于文本框的提示信息 span 元素，同理也是删除全部加载过的样式，使其恢复到初始样式，并显示相关的提示信息，其代码如下：

```
$("#spnTip").removeClass("spnBlur")
.removeClass("spnSucc")
.html(" 请输入您常用邮箱地址！ ");
```

在文本框的 blur 丢失焦点事件中，其代码与 foucs 事件有相似之处，都是先清理原先加载过的页面样式，然后增加本身事件中的样式。不同之处在于，在文本框的 blur 事件中，还要进行文本框内容是否为空和邮箱格式是否符合的检测操作。如果不为空，再进行检测，其代码如下：

```
if (vtxt.length == 0) {
    // 文本内容为空时执行的代码
    ...
   }
else { // 如果不为空时
    if (!chkEmail(vtxt)) {// 检测邮箱格式是否正确
    // 邮箱格式不正确时执行的代码
    ...
    }
```

```
        else {// 如果正确
              // 邮箱格式正确时执行的代码
              ...
         }
}
```

在 jQuery 中，addClass() 方法的功能是增加某种 CSS 样式，为了更好地体现设置的样式，在增加新的 CSS 样式前，应先通过 removeClass() 方法，删除已加载过的页面样式，以达到预期的页面效果。

4.7.2 下拉列表框中的事件应用

下拉列表框是最为常用的表单对象，该对象可以通过较小的页面空间，展示大量的数据；同时，多个列表框通过联动效果，展示数据的应用也相当广泛。下面通过一个示例，介绍如何在 jQuery 中，实现三个下拉列表框联动展示数据的功能。

示例 4-10 下拉列表框中的事件应用

（1）功能描述

为实现根据厂商、名牌、型号查询车型的功能，在页面中，设置三个下拉列表框，分别用于保存厂商、名牌、型号的数据。当用户在选择厂商时，名牌和型号下拉列表框随其数据变化而变化；当用户选择名牌时，型号下拉列表框随其所选数据变化而改变，从而实现了三个下拉列表框联动展示数据的功能。单击“查询”按钮时，显示用户所选择的全部选项。

（2）实现代码

新建一个 HTML 文件 4-10.html, 加入如代码清单 4-10 所示的代码。

代码清单 4-10 下拉列表框中的事件应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>列表框中事件应用</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .clsInit{width:435px;height:35px;line-height:35px;
          padding-left:10px}
          .clsTip{padding-top:5px;background-color:#eee;
          display:none}
          .btn {border:#666 1px solid;padding:2px;width:65px;
          float:right;margin-top:6px;margin-right:6px;
          filter: progid:DXImageTransform.Microsoft.
```

```
        Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8);}
  </style>
<script type="text/javascript">
    $(function() {
        function objInit(obj) {// 下拉列表框初始化
            return $(obj).html("<option> 请选择 </option>");
        }
        var arrData = { // 定义一个数组保存相关数据
            厂商 1: { 品牌 1_1: " 型号 1_1_1, 型号 1_1_2",
                      品牌 1_2: " 型号 1_2_1, 型号 1_2_2" },
            厂商 2: { 品牌 2_1: " 型号 2_1_1, 型号 2_1_2",
                      品牌 2_2: " 型号 2_2_1, 型号 2_2_2" },
            厂商 3: { 品牌 3_1: " 型号 3_1_1, 型号 3_1_2",
                      品牌 3_2: " 型号 3_2_1, 型号 3_2_2" }
        };
        $.each(arrData, function(pF) { // 遍历数据增加厂商项
            $("#selF").append("<option>" + pF + "</option>");
        });
        $("#selF").change(function() { // 厂商列表框选项改变事件
            objInit("#selT");
            objInit("#selC");
            $.each(arrData, function(pF, pS) {
                // 如果厂商选中项与数据匹配
                if ($("#selF option:selected").text() == pF) {
                    // 遍历数据增加品牌项
                    $.each(pS, function(pT, pC) {
                        $("#selT").append("<option>"
                        + pT +"</option>");
                    });
                    // 品牌列表框选项改变事件
                    $("#selT").change(function() {
                        objInit("#selC");
                        $.each(pS, function(pT, pC) {
                            // 如果品牌选中项与数据匹配
                            if ($("#selT option:selected")
                                .text() == pT) {
                                // 遍历数据增加型号项
                                $.each(pC.split(","), function() {
                                    $("#selC").append("<option>"
                                    + this + "</option>");
                                });
                            }
                        });
                    });
                }
            });
    });

    $("#Button1").click(function() { // 注册按钮单击事件
```

```
                var strValue = " 您选择的厂商 :";
                strValue += $("#selF option:selected").text();
                strValue += "  您选择的品牌 :";
                strValue += $("#selT option:selected").text();
                strValue += "  您选择的型号 :";
                strValue += $("#selC option:selected").text();
                $("#divTip")
                .show()
                .addClass("clsTip")
                .html(strValue);// 显示提示信息并增加样式
            });
        })
        </script>
</head>
<body>
     <div class="clsInit">
       厂商 : <select id="selF"><option> 请选择 </option></select>
       品牌 : <select id="selT"><option> 请选择 </option></select>
       型号 : <select id="selC"><option> 请选择 </option></select>
       <input id="Button1" type="button" value=" 查询 " class="btn" />
    </div>
    <div class="clsInit" id="divTip"></div>
</body>
</html>
```

（3）页面效果

执行代码后的效果如图 4-11 所示。

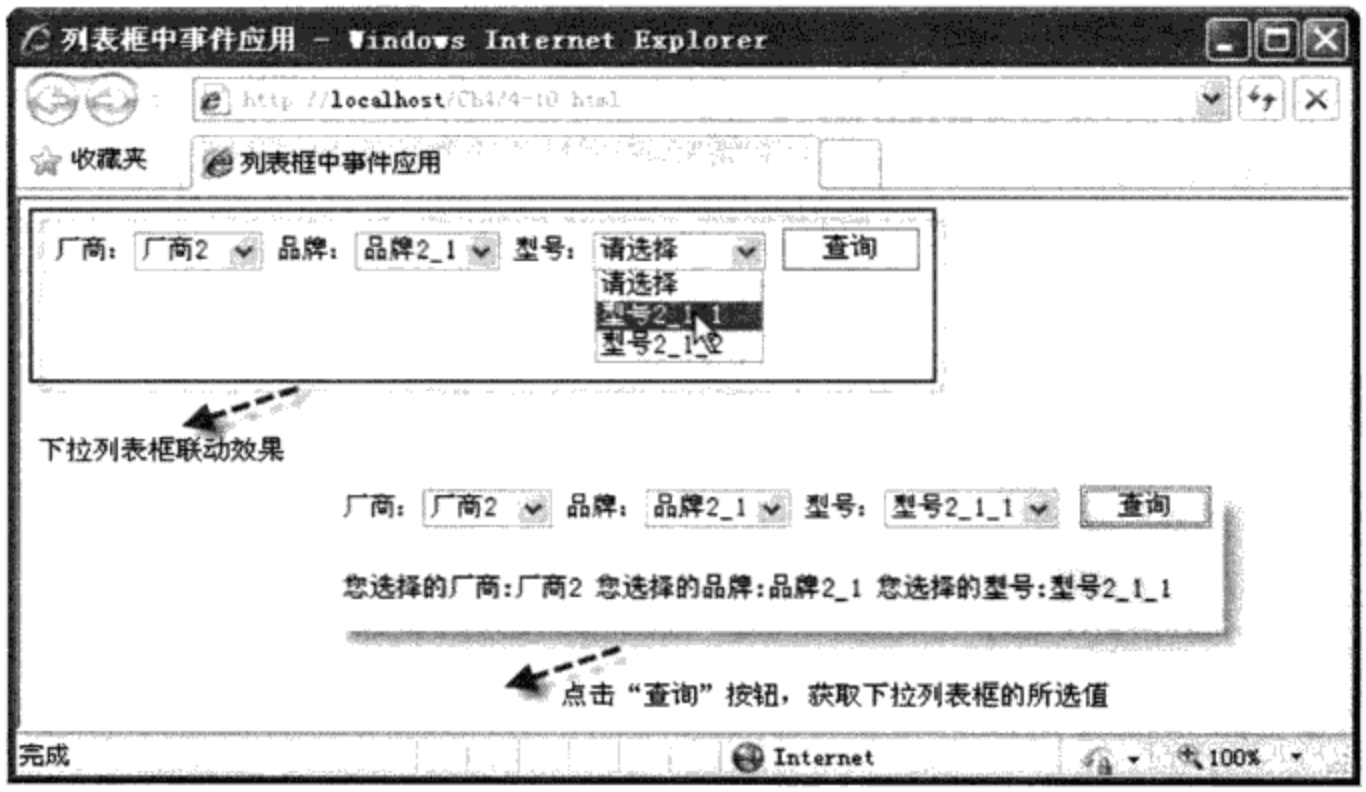


图 4-11　多个列表框的联动应用

（4）代码分析

在示例 4-10 中，通过数组保存各下拉列表框所需要的数据，代码如下：

```
var arrData = { //定义一个数组保存相关数据
      厂商 1: { 品牌 1_1: "型号 1_1_1, 型号 1_1_2",
                品牌 1_2: "型号 1_2_1, 型号 1_2_2" },
      厂商 2: { 品牌 2_1: "型号 2_1_1, 型号 2_1_2",
                品牌 2_2: "型号 2_2_1, 型号 2_2_2" },
      厂商 3: { 品牌 3_1: "型号 3_1_1, 型号 3_1_2",
                品牌 3_2: "型号 3_2_1, 型号 3_2_2" }
};
```

该数组的结构包含三次分隔部分：首先是通过“：”冒号完成第一次分隔，产生 A 与 B 两部分；然后，在第一次分隔后，对 B 部分的数据通过“,”逗号完成第二次分隔，产生独立的 B0 部分；最后，在第二次分隔后，在 B0 部分的数据通过“：”冒号完成第三次分隔，其示意图如图 4-12 所示。

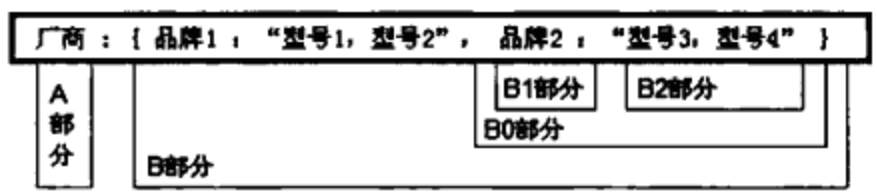


图 4-12　数组结构

在掌握数组结构后，接下来的任务就是通过 jQuery 中的 each() 方法遍历数组，获取需要的数组元素。首先通过第一轮的遍历数组，获取“厂商”的信息，并加入下拉列表框中，其代码如下所示：

```
$.each(arrData, function(pF) { //遍历数据增加厂商项
      $("#selF").append("<option>" + pF + "</option>");
 });
```

其中参数 pF 实际上就是示意图 4-12 中的 A 部分。

接下来进行第二轮的遍历数据，获取与“厂商”信息相匹配的“品牌”数组元素，并增加到第二个下拉列表框中，其代码如下：

```
...省略部分代码
$.each(arrData, function(pF, pS) {
        //如果厂商选中项与数据匹配
        if ($("#selF option:selected").text() == pF) {
              //遍历数据增加品牌项
              $.each(pS, function(pT, pC) {
                    $("#selT").append("<option>"
                    + pT +"</option>");
               });
               ...省略部分代码
          }
```

```
});
... 省略部分代码
```

代码中参数 pS 为示意图 4-12 中的 B0 部分，代码“$("#selF option:selected").text()== pF”表示“厂商”选项所选中的值与数组中某个“厂商”元素匹配，才遍历 B0 部分，其中参数 pT 为示意图中的 B1 部分，pC 为 B2 部分，在 B0 部分遍历过程中，将与“厂商”相匹配的“品牌”数组元素增加到第二个下拉列表框中。

同理，“品牌”下拉列表框在 change 事件中，依然需要遍历整个数组，才能获取其相对应的型号数组元素数据，其实现的代码如下：

```
... 省略部分代码
$.each(arrData, function(pF, pS) {
        ... 省略部分代码
        $.each(pS, function(pT, pC) {
                //如果品牌选中项与数据匹配
                if ($("#selT option:selected")
                    .text() == pT) {
                    //遍历数据增加型号项
                    $.each(pC.split(","), function() {
                            $("#selC").append("<option>"
                            + this + "</option>");
                    });
                }
        });
    ... 省略部分代码
});
... 省略部分代码
```

由于参数 pC 所获取的数据中，需要通过“,”逗号分隔后才可使用，因此在第三次遍历元素时，使用了“pC.split(",")”的方法获取分隔后的数据，在增加时，this 就是遍历时的各个分隔后的元素，通过 append() 方法，将该值增加到“型号”下拉列表框中。

最后，在按钮的单击事件中，通过 jQuery 中的 val() 方法获取各下拉列表框中的所选值，并进行累加，通过一个 元素展示在页面中，其代码如下：

```
... 省略部分代码
var strValue = "您选择的厂商：";
strValue += $("#selF option:selected").text();
strValue += " 您选择的品牌：";
strValue += $("#selT option:selected").text();
strValue += " 您选择的型号：";
strValue += $("#selC option:selected").text();
$("#divTip")
.show()
.addClass("clsTip")
.html(strValue);// 显示提示信息并增加样式
... 省略部分代码
```

在显示时，由于默认是隐藏的，因此先通过 show() 方法，再增加一个样式，最后使用 html() 方法将全部累加的文本内容显示在 元素中。

4.8　列表应用

在页面开发中，经常使用 ，即列表标记。在设计展示数据或导航菜单的页面中，列表的使用相当广泛。下面通过一个简单的示例，介绍在 jQuery 中，如何使用列表 <ul> 标记实现导航菜单的功能。

示例 4-11　列表中的导航菜单应用

（1）功能描述

在页面表单中，分别展示某类产品的全部子类项，当用户将鼠标移动某项子类时，所选子类样式发生变化，并在该子类的右边以浮动的形式展示该类的全部产品；当用户将鼠标移出某项子类时，所选子类样式恢复到初始值，同时，隐藏已显示的全部子类产品。

（2）实现代码

新建一个 HTML 文件 4-11.html, 加入如代码清单 4-11 所示的代码。

代码清单 4-11　列表中的导航菜单应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>列表中的导航菜单应用</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           ul,li{list-style-type:none;padding:0px;margin:0px}
           .menu{width:190px;border:solid 1px #E5D1A1;
           background-color:#FFFDD2}
           .optn{width:190px;line-height:28px;
           border-top:dashed 1px #ccc}
           .content{padding-top:10px;clear:left}
           a{text-decoration:none;color:#666;padding:10px}
           .optnFocus{background-color:#fff;font-weight:bold}
           div{padding:10px}
           div img{float:left;padding-right:6px}
           span{padding-top:3px;font-size:14px;
           font-weight:bold;float:left}
           .tip{width:190px;border:solid 2px #ffa200;
           position:absolute;padding:10px;
           background-color:#fff;display:none}
```

```
            .tip li{line-height:23px;}
            #sort{position:absolute;display:none}
    </style>
<script type="text/javascript">
    $(function() {
        var curY; // 获取所选项的 Top 值
        var curH; // 获取所选项的 Height 值
        var curW; // 获取所选项的 Width 值
        var srtY; // 设置提示箭头的 Top 值
        var srtX; // 设置提示箭头的 Left 值
        var objL; // 获取当前对象
        /*
        * 设置当前位置数值
        * 参数 obj 为当前对象名称
        */
        function setInitValue(obj) {
            curY = obj.offset().top
            curH = obj.height();
            curW = obj.width();
            srtY = curY + (curH / 2) + "px"; // 设置提示箭头的 Top 值
            srtX = curW - 5 + "px"; // 设置提示箭头的 Left 值
        }
        // 设置当前所选项的鼠标滑过事件
        $(".optn").mouseover(function() {
            objL = $(this); // 获取当前对象
            setInitValue(objL); // 设置当前位置
            var allY = curY - curH + "px"; // 设置提示框的 Top 值
            objL.addClass("optnFocus"); // 增加获取焦点时的样式
            objL.next("ul").show()
            // 显示并设置提示框的坐标
            .css({ "top": allY, "left": curW })
            $("#sort").show()
            // 显示并设置提示箭头的坐标
            .css({ "top": srtY, "left": srtX });
        })
        .mouseout(function() {// 设置当前所选项的鼠标移出事件
            // 删除获取焦点时的样式
            $(this).removeClass("optnFocus");
            $(this).next("ul").hide(); // 隐藏提示框
            $("#sort").hide(); // 隐藏提示箭头
        })
        $(".tip").mousemove(function() {
            $(this).show(); // 显示提示框
            objL = $(this).prev("li"); // 获取当前的上级 li 对象
            setInitValue(objL); // 设置当前位置
            // 增加上级 li 对象获取焦点时的样式
            objL.addClass("optnFocus");
            // 显示并设置提示箭头的坐标
            $("#sort").show().css({ "top": srtY, "left": srtX });
        })
```

```
            .mouseout(function() {
                $(this).hide(); // 隐藏提示框
                // 删除获取焦点时的样式
                $(this).prev("li").removeClass("optnFocus");
                $("#sort").hide(); // 隐藏提示箭头
            })
        })
        </script>
</head>
<body>
    <ul>
        <li class="menu">
            <div>
                <img alt="" src="Images/icon.gif" />
                <span>电脑数码类产品</span>
            </div>
            <ul class="content">
                <li class="optn"><a href="#">笔记本</a></li>
                <ul class="tip">
                    <li><a href="#">笔记本 1</a></li>
                    <li><a href="#">笔记本 2</a></li>
                    <li><a href="#">笔记本 3</a></li>
                    <li><a href="#">笔记本 4</a></li>
                    <li><a href="#">笔记本 5</a></li>
                </ul>
                <li class="optn"><a href="#">移动硬盘</a></li>
                <ul class="tip">
                    <li><a href="#">移动硬盘 1</a></li>
                    <li><a href="#">移动硬盘 2</a></li>
                    <li><a href="#">移动硬盘 3</a></li>
                    <li><a href="#">移动硬盘 4</a></li>
                    <li><a href="#">移动硬盘 5</a></li>
                </ul>
                <li class="optn"><a href="#">电脑软件</a></li>
                <ul class="tip">
                    <li><a href="#">电脑软件 1</a></li>
                    <li><a href="#">电脑软件 2</a></li>
                    <li><a href="#">电脑软件 3</a></li>
                    <li><a href="#">电脑软件 4</a></li>
                    <li><a href="#">电脑软件 5</a></li>
                </ul>
                <li class="optn"><a href="#">数码产品</a></li>
                <ul class="tip">
                    <li><a href="#">数码产品 1</a></li>
                    <li><a href="#">数码产品 2</a></li>
                    <li><a href="#">数码产品 3</a></li>
                    <li><a href="#">数码产品 4</a></li>
                    <li><a href="#">数码产品 5</a></li>
                </ul>
            </ul>
```

```
            <img id="sort" src="Images/sort.gif" alt=""/>
        </li>
    </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-13 所示。

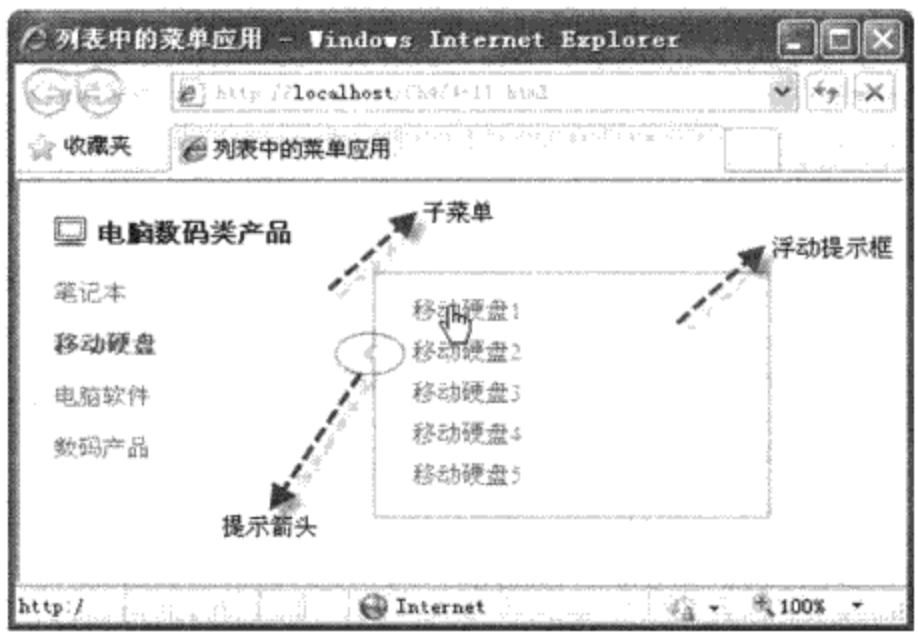


图 4-13　列表中导航菜单的应用

（4）代码分析

通过列表元素 <ul> 实现菜单导航条的功能，在很多大型的购物网站中随处可见。这样的导航菜单的功能之一是，当鼠标移到某个子菜单选项上，浮动显示子菜单相对应的全部产品，并且提示箭头指向该子菜单选项，同时，子菜单选项的样式发生变化。

为实现该功能，首先通过下列代码获取所有子菜单，并设置 mousemove 事件，其代码如下：

```
$(".optn").mouseover(function() {
// 执行代码
...
 })
```

在子菜单选项的 mouseover 事件中，先通过自定义的函数 setInitValue 获取提示箭头的位置，即 srtY 与 srtX 变量值。然后根据当前所选的子菜单项的 Top 和 Width 值，设置浮动提示框的 Top 和 Left 值，即代码中的 allY 与 curW 变量的值，其代码如下：

```
... 省略部分代码
objL = $(this); // 获取当前对象
setInitValue(objL); // 设置当前位置
var allY = curY - curH + "px"; // 设置提示框的 Top 值
... 省略部分代码
```

接下来，获取子菜单项对应的浮动提示框，并根据设置的位置显示提示框和提示箭头，同时，自身增加获取焦点时的样式，其实现的代码如下：

```
...省略部分代码
objL.addClass("optnFocus"); // 增加获取焦点时的样式
objL.next("ul").show()
// 显示并设置提示框的坐标
.css({ "top": allY, "left": curW })
$("#sort").show()
// 显示并设置提示箭头的坐标
.css({ "top": srtY, "left": srtX });
...省略部分代码
```

当子菜单中的鼠标移出时，触发 mouseout 事件，在该事件中，先移除所增加的焦点样式，然后隐藏对应的提示框和提示箭头。其实现的代码如下：

```
...省略部分代码
.mouseout(function() {// 设置当前所选项的鼠标移出事件
    $(this).removeClass("optnFocus"); // 删除获取焦点时的样式
    $(this).next("ul").hide(); // 隐藏提示框
    $("#sort").hide(); // 隐藏提示箭头
 })
...省略部分代码
```

由于子菜单在鼠标移出 mouseout 事件，隐藏了对应显示的提示框和提示箭头，因此，当用户将鼠标移出子菜单时，提示框和提示箭头都不见了，这不是我们想要的功能。

为了避免这样的效果，需要设置提示框的鼠标 mousemove 和 mouseout 事件，提示框注册 mousemove 事件的代码如下：

```
...省略部分代码
$(".tip").mousemove(function() {
    // 执行代码
    ...
 })
...省略部分代码
```

在提示框的 mousemove 事件中，为了改变所选子菜单的样式，先获取子菜单，其代码如下：

```
objL = $(this).prev("li"); // 获取当前的上级 li 对象
```

然后，通过自定义函数 setInitValue，获取提示箭头的位置，并显示在页面中，同时，再增加获取焦点时的样式。其代码如下：

```
...省略部分代码
setInitValue(objL); // 设置当前位置
// 增加上级 li 对象获取焦点时的样式
objL.addClass("optnFocus");
// 显示并设置提示箭头的坐标
```

```
$("#sort").show().css({ "top": srtY, "left": srtX });
... 省略部分代码
```

在提示框的 mouseout 事件中，移除所增加的焦点样式，并隐藏提示框和提示箭头，其实现的代码如下：

```
... 省略部分代码
.mouseout(function() {
   $(this).hide(); // 隐藏提示框
   // 删除获取焦点时的样式
   $(this).prev("li").removeClass("optnFocus");
   $("#sort").hide(); // 隐藏提示箭头
 })
... 省略部分代码
```

4.9 网页选项卡的应用

在页面中，除使用列表 <ul> 标记实现滑动效果的菜单导航条外，还用于网页选项卡的设计。选项卡的功能十分简单，通过单击标题实现内容隐藏或显示的切换，由于它可以在有限的空间中展示大量的数据，因此，被广泛使用在各综合性的门户网站中。下面通过一个简单的示例，介绍网页选项卡快速实现的方法。

示例 4-12 网页选项卡的应用

（1）功能描述

在页面中，设置三个不同名称的选项卡，当单击某个选项卡时，下面相对应的区域显示其内容信息，同时选项卡的背景色与内容信息的背景色浑然一体，并且字体加粗，表示处于选中状态。

（2）实现代码

新建一个 HTML 文件 4-12.html，加入如代码清单 4-12 所示的代码 .

代码清单 4-12 网页选项卡的应用

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> 网页选项卡应用 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           ul,li {margin:0;padding:0;list-style:none}
           #menu li {text-align:center;float:left;padding:5px;
           margin-right:2px;width:50px;cursor:pointer}
```

```
            #menu li.tabFocus {width:50px; font-weight:bold;
            background-color:#f3f2e7;border:solid 1px #666;
            border-bottom:0;z-index:100;
            position:relative}
            #content {width:260px;height:80px;padding:10px;
            background-color:#f3f2e7;clear:left;
            border:solid 1px #666;
            position:relative;top:-1px}
            #content li{display:none}
            #content li.conFocus{display:block}
    </style>
    <script type="text/javascript">
        $(function() {
              // 带参数遍历各个选项卡
            $("#menu li").each(function(index) {
                $(this).click(function() { // 注册每个选卡的单击事件
                  // 移除已选中的样式
                  $("#menu li.tabFocus").removeClass("tabFocus");
                  $(this).addClass("tabFocus"); // 增加当前选中项的样式
                  // 显示选项卡对应的内容并隐藏未被选中的内容
                  $("#content li:eq(" + index + ")").show()
                  .siblings().hide();
                });
            });
        })
    </script>
</head>
<body>
     <ul id="menu">
            <li class="tabFocus">家居 </li>
            <li>电器 </li>
            <li>二手 </li>
    </ul>
    <ul id="content">
            <li class="conFocus">我是家居的内容 </li>
            <li>欢迎您来到电器城 </li>
            <li>二手市场，产品丰富多彩 </li>
     </ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 4-14 所示。

（4）代码分析

在源码中，为了找到每个选项卡，先带参数遍历整个选项卡，其实现的代码如下：

```
$("#menu li").each(function(index) { // 遍历整个选项卡
    // 执行代码
    ...
 });
```

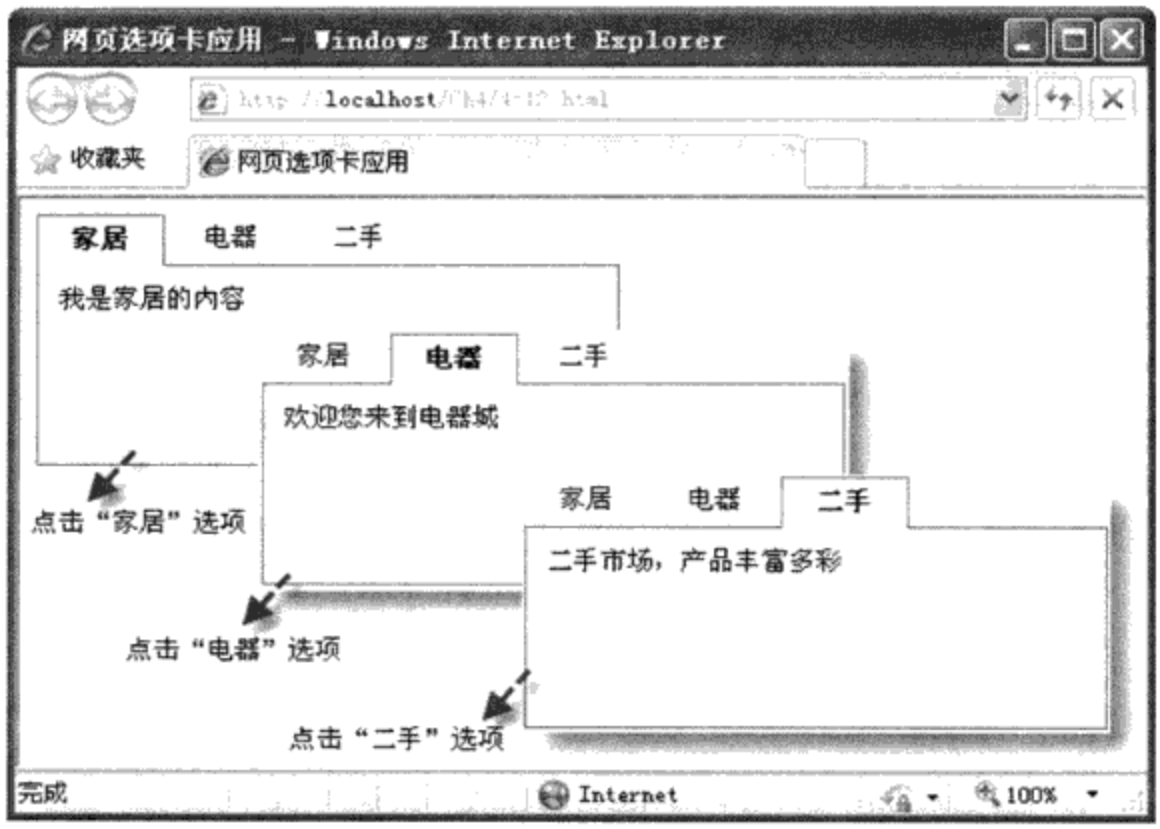


图 4-14　网页选项卡应用

在选项卡的事件遍历中，注册每个选项卡的单击事件，代码如下：

```
$(this).click(function() { // 注册每个选卡的单击事件
    // 执行代码
    ...
});
```

在选项卡的单击事件中，先移除以前被选中的选项卡焦点样式，然后增加当前选项卡的焦点样式；在显示对应内容时，先通过 $("#content li:eq("+index+")") 语句获取当前选项卡所对应的内容，并显示该内容，同时，通过 siblings() 方法隐藏其对应的兄弟内容。其实现的代码如下：

```
...省略部分代码
// 移除已选中的样式
$("#menu li.tabFocus").removeClass("tabFocus");
$(this).addClass("tabFocus"); // 增加当前选中项的样式
// 显示选项卡对应的内容并隐藏未被选中的内容
$("#content li:eq(" + index + ")").show()
.siblings().hide();
...省略部分代码
```

4.10　综合案例分析——删除数据时的提示效果在项目中的应用

4.10.1　需求分析

经分析，该案例的需求如下：

1）当用户单击“删除”按钮时，整个页面背景类似于关机效果，“删除”提示框突出显

示，用户可以选“关闭”按钮，或单击“确定”或“取消”的操作。

2）删除提示框一直居中显示，不论页面大小发生如何变化，这个提示框始终居中显示。

3）如果对某条记录打勾，当用户单击提示框中的“确定”按钮时，将在页面中删除该条记录，同时关闭提示框，页面背景恢复正常。

4.10.2　效果界面

该案例的效果界面如下所示：

1）用户单击“删除”按钮前，其实现的界面如图 4-15 所示。

图 4-15　记录删除前

2）当用户单击“删除”按钮后，出现删除提示对话框，其实现的界面如图 4-16 所示。

图 4-16　删除记录的提示对话框

3）当页面大小发生变化时，提示对话框一直居中显示，其实现的界面如图 4-17 所示。

图 4-17　页面变化后的提示对话框

4.10.3　功能实现

在该项目中，新建一个 HTML 文件 delete.html，加入如代码清单 4-13 所示的代码。

代码清单 4-13　删除记录时的提示效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>删除记录时的提示效果</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
     <style type="text/css">
           body{font-size:13px}
           .divShow{line-height:32px;height:32px;
           background-color:#eee;width:280px;padding-left:10px}
           .divShow span{padding-left:50px}
           .dialog{width:360px;border:solid 5px #666;
           position:absolute;display:none;z-index:101}
           .dialog .title{background-color:#fbaf15;padding:10px;
```

```
        color:#fff;font-weight:bold}
        .dialog .title img{float:right}
        .dialog .content{background-color:#fff;padding:25px;
        height:60px}
        .dialog .content img{float:left}
        .dialog .content span{float:left;padding-top:10px;
        padding-left:10px}
        .dialog .bottom{text-align:right;
        padding:10px 10px 10px 0px;background-color:#eee}
        .mask {width:100%;height:100%; background-color:#000;
        position:absolute;top:0px;left:0px;
        filter:alpha(opacity=30);display:none;z-index:100}
        .btn {border:#666 1px solid;padding:2px;width:65px;
        filter:progid:DXImageTransform.Microsoft
        .Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8);}
</style>
 <script type="text/javascript">
     $(function() {
         $("#Button1").click(function() { // 注册删除按钮点击事件
             $(".mask").show(); // 显示背景色
             showDialog(); // 设置提示对话框的 Top 与 Left
             $(".dialog").show(); // 显示提示对话框
         })
         /*
         * 根据当前页面与滚动条位置，设置提示对话框的 Top 与 Left
         */
function showDialog() {
    var objW = $(window); // 当前窗口
    var objC = $(".dialog"); // 对话框
    var brsW = objW.width();
    var brsH = objW.height();
    var sclL = objW.scrollLeft();
    var sclT = objW.scrollTop();
    var curW = objC.width();
    var curH = objC.height();
    // 计算对话框居中时的左边距
    var left = sclL + (brsW - curW) / 2;
    // 计算对话框居中时的上边距
    var top = sclT + (brsH - curH) / 2;
    // 设置对话框在页面中的位置
    objC.css({ "left": left, "top": top });
}

$(window).resize(function() {// 页面窗口大小改变事件
    if (!$(".dialog").is(":visible")) {
        return;
    }
    showDialog(); // 设置提示对话框的 Top 与 Left
});
```

```
            $(".title img").click(function() { //注册关闭图片点击事件
                $(".dialog").hide();
                $(".mask").hide();
            })

            $("#Button3").click(function() {//注册取消按钮点击事件
                $(".dialog").hide();
                $(".mask").hide();
            })

            $("#Button2").click(function() {//注册确定按钮点击事件
              $(".dialog").hide();
              $(".mask").hide();
              if ($("input:checked").length != 0) {//如果选择了删除行
                    $(".divShow").remove(); //删除某行数据
                }
            })
        })
    </script>
</head>
<body>
     <div class="divShow">
         <input id="Checkbox1" type="checkbox" />
         <a href="#">这是一条可删除的记录</a>
         <span>
               <input id="Button1" type="button"
                      value="删除" class="btn" />
         </span>
     </div>
     <div class="mask"></div>
     <div class="dialog">
          <div class="title">
               <img src="Images/close.gif"
                    alt="点击可以关闭" />删除时提示
          </div>
          <div class="content">
               <img src="Images/delete.jpg" alt="" />
                    <span>您真的要删除该条记录吗?</span>
          </div>
          <div class="bottom">
              <input id="Button2" type="button"
                    value="确定" class="btn"/>  
              <input id="Button3" type="button"
                    value="取消" class="btn"/>
          </div>
     </div>
</body>
</html>
```

4.10.4　代码分析

本案例核心功能是当用户在删除某条记录时，弹出一个提示对话框，等待用户单击“确定”按钮后，才能直接删除。为了实现该功能，首先注册“删除”按钮的单击事件，其代码如下：

```
$("#Button1").click(function() { // 注册删除按钮点击事件
    // 执行代码
    ...
})
```

在“删除”按钮单击事件中，先显示设置好的背景元素，再通过自定义的函数 showDialog，设置好对话框的显示位置，最后显示提示对话框，其代码如下：

```
...省略部分代码
$(".mask").show(); // 显示背景色
showDialog(); // 设置提示对话框的 Top 与 Left
$(".dialog").show(); // 显示提示对话框
...省略部分代码
```

在自定义的函数 showDialog 中，根据窗口的长与宽和滚动条的 Top 与 Left 值及对话框自身的长与宽，计算出使提示对话框始终居中的坐标 Top 和 Left 变量值，并根据该变量值，设置对话框的 Top 与 Left 属性值，其代码如下：

```
...省略部分代码
function showDialog() {
   var objW = $(window); // 当前窗口
   var objC = $(".dialog"); // 对话框
   var brsW = objW.width();
   var brsH = objW.height();
   var sclL = objW.scrollLeft();
   var sclT = objW.scrollTop();
   var curW = objC.width();
   var curH = objC.height();
   // 计算对话框居中时的左边距
   var left = sclL + (brsW - curW) / 2;
   // 计算对话框居中时的上边距
   var top = sclT + (brsH - curH) / 2;
   // 设置对话框在页面中的位置
   objC.css({ "left": left, "top": top });
 }
...省略部分代码
```

为了更好地理解上述自定义的函数 showDialog 中代码的功能，我们通过图 4-18 说明如何才能使提示对话框居中。

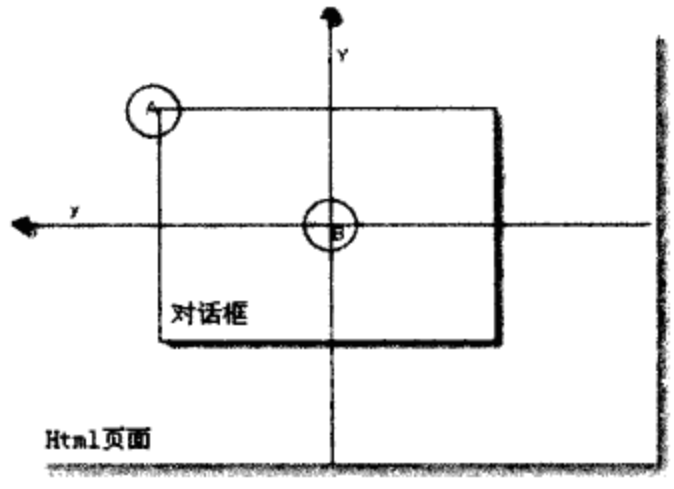


图 4-18　对话框在页面中居中示意图

我们清楚地看到，如果对话框的原点与页面的原点完全重合，即都在 B 点，对话框则居中，那么，对话框的坐标，实际上就是 A 点的位置，它的 Left 值为页面的 Width 值减掉对话框的 Width 值后除以 2，再加上滚动条的 Left 值，即如下代码：

```
var left = sclL + (brsW - curW) / 2;
```

Top 值为页面的 Height 值减掉对话框的 Height 值后除以 2，再加上滚动条的 Top 值，即如下代码：

```
var top = sclT + (brsH - curH) / 2;
```

获取对话框的变量值后，再在窗口的 resize 事件中，重新根据上述方法进行提示对话框坐标的计算，以保证当页面大小发生变化时，对话框始终居中显示，resize 事件执行的代码如下：

```
...省略部分代码
$(window).resize(function() {// 页面窗口大小改变事件
    if (!$(".dialog").is(":visible")) {
            return;
     }
    showDialog(); // 设置提示对话框的 Top 与 Left
 });
...省略部分代码
```

代码 if (!$(".dialog").is(":visible")) 表示，如果没有出现对话框，则不执行该事件中的代码，以避免不必要的资源占用。

当用户单击“确定”按钮时，首先检测是否有选中的可删除记录，然后再进行删除的操作，其代码如下：

```
...省略部分代码
$("#Button2").click(function() {// 注册确定按钮点击事件
    $(".dialog").hide();
    $(".mask").hide();
    if ($("input:checked").length != 0) {// 如果选择了删除行
```

```
                $(".divShow").remove(); // 删除某行数据
            }
        })
    ... 省略部分代码
```

当用户单击“关闭”图片按钮或单击“取消”按钮时，均执行同样的代码事件，即先隐藏居中显示的对话框，同时，隐藏背景，使页面恢复到初始状态，其实现的代码如下：

```
... 省略部分代码
$(".dialog").hide();
$(".mask").hide();
... 省略部分代码
```

4.11　本章小结

在 DOM 操作中，离不开事件的触发。本章以循续渐进的方式，详细地介绍了 jQuery 中的各种常用类型的事件，通过简单的事件应用示例，阐述各类元素注册事件的方法。本章最后还介绍如何使用 jQuery 中的注册事件，实现目前页面体验中最为常见的各种效果。

第 5 章

jQuery 的动画与特效

本章内容

- 显示与隐藏
- 滑动
- 淡入淡出
- 自定义动画
- 动画效果综述
- 综合案例分析——动画效果浏览相册中的图片
- 本章小结

如何能最大化地优化页面的用户体验度，是每个前端页面开发人员在设计页面时需要考虑的一个重要问题。无可置疑，jQuery 中众多的动画与特效方法为提高页面的用户体验度带来了极大的方便，通过少量的几行代码，就可以实现元素的飞动、淡入淡出等动画效果，还可以自定义各种动画效果。

5.1　显示与隐藏

在页面中，元素的显示与隐藏是使用最频繁的操作，在传统的 JavaScript 中，一般通过改变元素显示的方式实现，下列代码将 ID 号为“p1”的元素显示出来：

```
document.getElementById("p1").style.display = "block";
```

如果想隐藏该元素，则可以通过下列代码：

```
document.getElementById("p1").style.display = "none";
```

而在 jQuery 中，元素的显示与隐藏的方法比传统的 JavaScript 要多，并且实现的效果也很优雅。下面我们逐步介绍在 jQuery 中如何实现元素的显示与隐藏的方法。

5.1.1　show() 与 hide() 方法

我们在前面的章节中不止一次地使用过 show() 与 hide() 方法，前者是显示页面中的元素，等同于下列 jQuery 代码：

```
$("#p1").css("display":"block");
```

后者是隐藏页面中的元素，与前者 show() 的方法正好相反，等同于下列 jQuery 代码：

```
$("#p1").css("display":"none");
```

下面通过一个示例介绍这两个 jQuery 方法的使用过程。

示例 5-1　show() 与 hide() 方法简介

（1）功能描述

在显示大量文本内容时，为了能显示更多的段落内容，有时仅显示一部分的提要，隐藏另一部分的内容，当用户需要查看这些隐藏的内容时，只要单击页面中的“显示”链接就可以，查看完后，再单击“隐藏”链接便将该部分内容再次隐藏起来。

（2）实现代码

新建一个 HTML 文件 5-1.html, 加入如代码清单 5-1 所示的代码。

代码清单 5-1　show() 与 hide() 方法

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

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>show() 与 hide() 方法 </title>
    <script type="text/javascript"
           src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .artFram{border:solid 1px #ccc;background-color:#eee;
                    width:260px;padding:8px;word-break:break-all}
           .artList{line-height:1.8em}
    </style>
    <script type="text/javascript">
        $(function() {
            var $link = $(".artList a");// 获取单击文本
            var $hide = $(".artList :eq(2)");// 获取隐藏的段落
            $link.click(function() {// 文本单击事件
                if ($(this).html() == " 显示 ") {// 如果未显示
                    $(this).html(" 隐藏 ");// 改变文本内容
                    $hide.show();// 显示隐藏的文本
                } else {
                    $(this).html(" 显示 ");
                    $hide.hide();
                }
            })
        })
    </script>
</head>
<body>
     <div class="artFram">
          <div class="artList">
               <span>     </span>
               <span> 一个优秀的程序开发员，除需掌握专业的开发语言，
                    还要具有 </span>
               <span style="display:none"> 执着、沉稳、细致的
                    专业素质 </span>
                <a href="javascript:void(0)"> 显示 </a>
          </div>
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-1 所示。

5.1.2 动画效果的 show() 与 hide() 方法

jQuery 中的 show() 与 hide() 方法，不仅可以实现“无动画”版的显示与隐藏，还可以完

成有动画效果的显示与隐藏，只需在方法的括号中加入相应的参数即可，其调用的语法格式分别为以下两种。

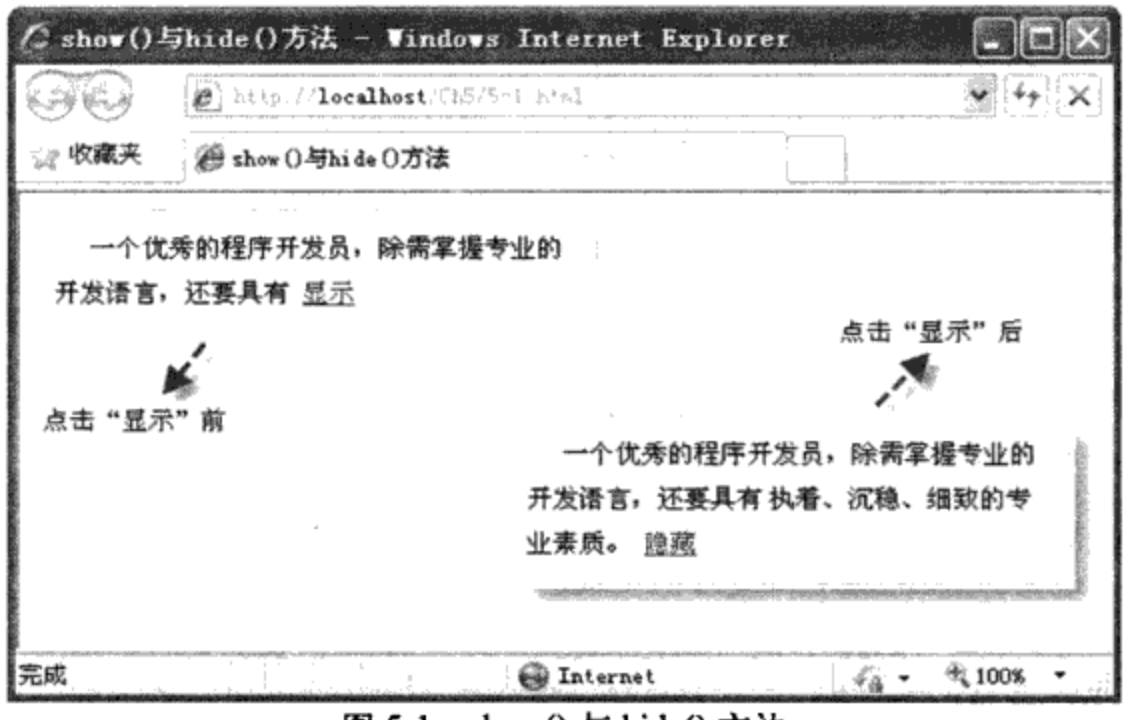


图 5-1　show() 与 hide() 方法

说明　在显示隐藏部分的内容时，为了获取当前显示状态，先通过语句 `if($(this).html()==" 显示 ")` 进行检测，然后根据检测结果，再执行不同的操作。

动画效果的显示功能如下所示：

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

动画效果的隐藏功能如下所示：

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

方法中的参数 speed 表示执行动画时的速度，该速度有 3 个默认字符值“slow”、“normal”、“fast”，其对应的速度分别是“0.6 秒”、“0.4 秒”、“0.2 秒”；如果不使用默认的字符值，也可以直接写入数字，如“3000”，表示该动画执行的速度为 3 000 毫秒。可选型参数 [callback] 为在动画完成时执行的回调函数，该函数每个元素执行一次。下面通过示例来介绍这两个方法的使用。

示例 5-2　动画效果的 show() 与 hide() 方法

（1）功能描述

在页面中单击“显示”链接，通过 show() 方法，以动画的方式显示一幅图片，同时，在方法中执行一个回调函数，用于改变图片的边框样式；当单击已显示的图片时，通过 hide() 方法，以动画的方式隐藏该图片。

（2）实现代码

新建一个 HTML 文件 5-2.html，加入如代码清单 5-2 所示的代码。

代码清单 5-2　动画效果的 show() 与 hide() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画效果的 show() 与 hide() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           img{display:none;cursor:pointer}
    </style>
    <script type="text/javascript">
        $(function() {
            $("a").click(function() {// 显示链接点击事件
                $("img").show(3000, function() {// 显示完成时执行的函数
                    $(this).css("border", "solid 1px #ccc");
                })
            })
            $("img").click(function() {// 显示图片的点击事件
                $(this).hide(3000);// 动画效果隐藏
            })
        })
    </script>
</head>
<body>
    <a href="javascript:void(0)"> 显示 </a>
    <img src="Images/img04.jpg" />
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-2 所示。

5.1.3　toggle() 方法

在使用 show() 或 hide() 方法显示或隐藏页面元素时，为了正确执行切换显示的动作，通常需要检测当前元素的显示状态，然后根据该状态再执行元素是否显示或隐藏。这样一来，代码显得有些冗长，在 jQuery 中，为了解决这个问题，可以使用 toggle() 方法，该方法的功能就是切换元素可见状态，即如果是显示状态，则变成隐藏状态；如果是隐藏状态，则变成显示状态。该方法有三种调用的形式，代码如下所示。

形式一，无参数调用格式：

```
toggle()
```

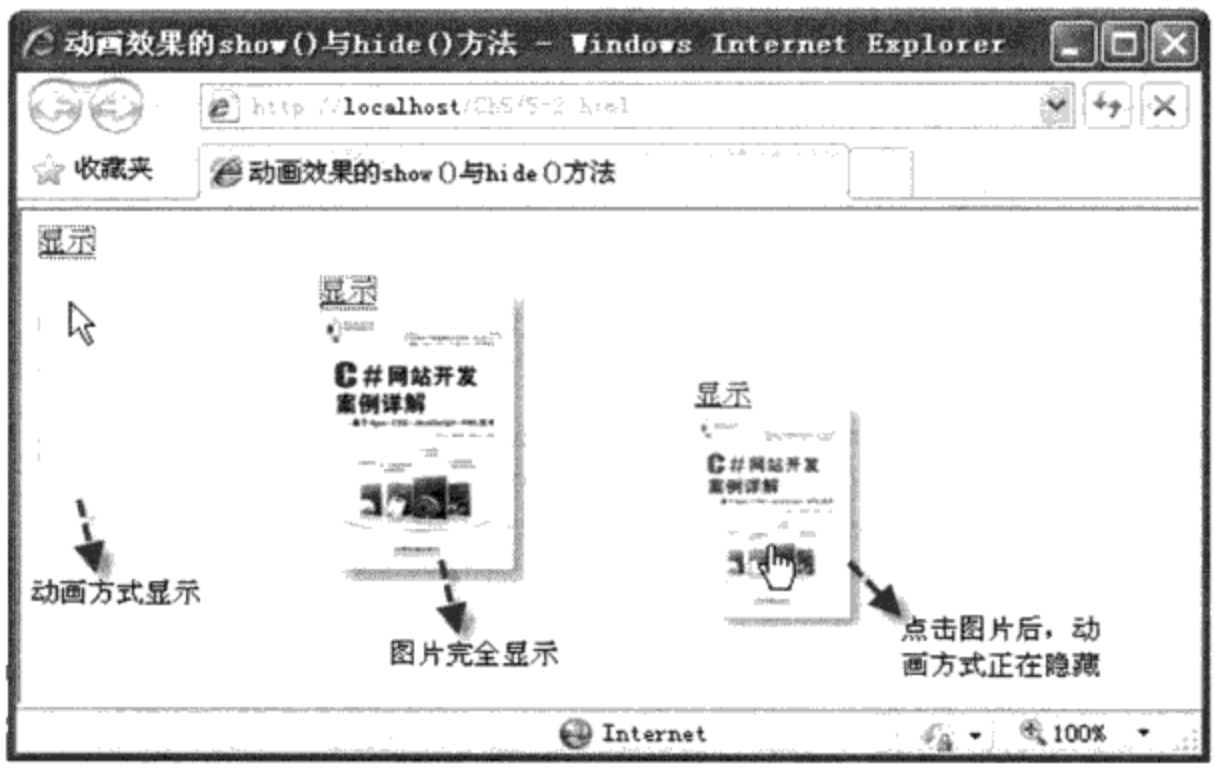


图 5-2　动画效果的 show() 与 hide() 方法

形式二，逻辑参数调用格式：

```
toggle(switch)
```

参数 switch 为一个布尔值，即 true 或 false。当该值为 true 时，显示元素；否则，隐藏元素。

形式三，动画效果调用格式：

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

其中参数 speed 和可选参数 [callback] 与方法 show(speed, [callback]) 中的参数所表示的意义是一样的，在此不作赘述。

下面通过示例来介绍 toggle 的使用方法。

示例 5-3　toggle() 方法的使用

（1）*功能描述*

为了更直观地区分开 toggle() 方法中的三种形式，在页面中设置了三个按钮分别对应三种调用的形式。通过单击按钮，触发 toggle() 方法，实现图片的切换显示效果。

（2）*实现代码*

新建一个 HTML 文件 5-3.html, 加入如代码清单 5-3 所示的代码。

代码清单 5-3　toggle() 方法

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

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>toggle() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:180px}
           .divFrame .divMenu{float:left}
           .divFrame .divContent{float:right}
           .divFrame .divContent img{display:none}
           .btn {border:#666 1px solid;padding:2px;
           width:80px;margin-bottom:5px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
            EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
           $("input:eq(0)").click(function() { // 无参数方法
                $("img").toggle();
           })
           $("input:eq(1)").click(function() { // 根据参数 switch 显示
                var intI = 0;
                var blnA = intI > 1; // 获取逻辑值
                $("img").toggle(blnA);
           })
           $("input:eq(2)").click(function() { // 动画方式显示
               $("img").toggle(3000, function() {
                   $(this) // 以动画方式显示，并执行回调函数
                   .css({ "border": "solid 1px #ccc",
                          "padding": "2px" });
               });
           })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divMenu">
            <input id="Button1" type="button"
            value=" 无参数 " class="btn" /><br />
            <input id="Button2" type="button"
            value=" 逻辑显示 " class="btn" /><br />
            <input id="Button3" type="button"
            value=" 动画显示 " class="btn" />
        </div>
        <div class="divContent">
            <img src="Images/img03.jpg" alt="" />
```

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

（3）页面效果

代码执行后的效果如图 5-3 所示。

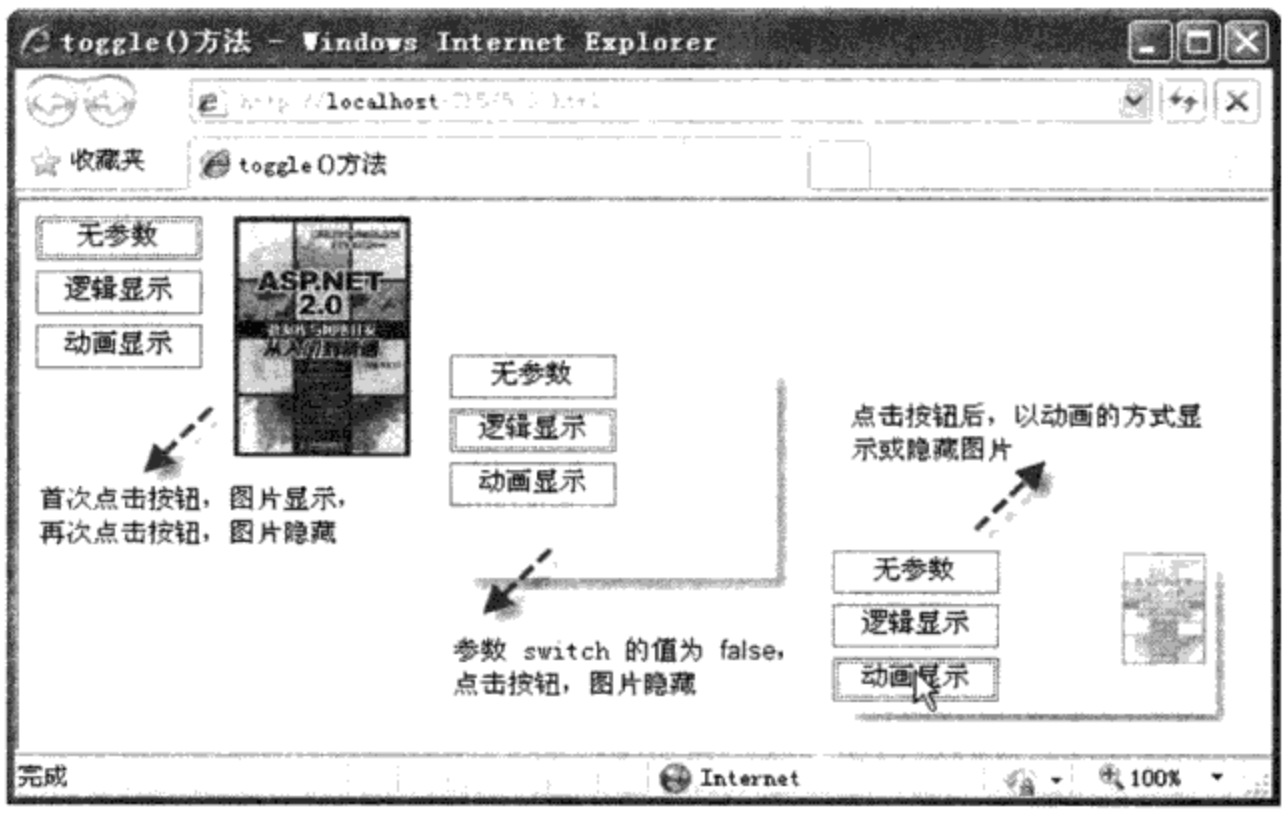


图 5-3　toggle() 方法

说明　无论是 show() 和 hide() 还是 toggle() 方法，当以动画效果切换页面元素可见状态时，其元素的 width、height、padding 和 margin 属性都将以动画的效果展示。

5.2　滑动

在 jQuery 中，还有一种滑动的动画效果改变元素的高度，即“拉窗帘”的效果。

5.2.1　slideDown() 与 slideUp 方法

要实现元素的滑动效果，需要调用 jQuery 中的两个方法，一个是 slideDown()，另一个是 slideUp()，其调用的语法格式介绍如下。

slideDown() 方法的格式如下：

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

其功能是以动画的效果将所选择元素的高度向下增大，使其呈现一种“滑动”的效果，而元素的其他属性并不发生变化；参数 speed 为动画显示的速度，可选项 [callback] 为动画显

示完成后，执行的回调函数。

slideUp () 方法的格式如下：

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

其功能是以动画的效果将所选择元素的高度向上减小，同样也是仅改变高度属性，其包含的参数作用与 slideDown() 方法一样。

下面通过示例来介绍这两种方法。

示例 5-4　slideDown() 与 slideUp() 方法

（1）功能描述

在页面中，单击“标题”栏时，通过 slideUp() 方法，以动画的效果将“内容”栏中的图片向上滑动，直到完全看不见，并改变“标题”栏中的内容；再次单击“标题”栏时，通过 slideDown() 方法，将“内容”栏中的图片向下滑动，直到全部显示，“标题”栏中的内容也同时发生相应改变。

（2）实现代码

新建一个 HTML 文件 5-4.html, 加入如代码清单 5-4 所示的代码。

代码清单 5-4　slideDown() 与 slideUp() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>slideDown() 与 slideUp() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:86px;border:solid 1px #666}
           .divFrame .divTitle{padding:5px;background-color:#eee}
           .divFrame .divContent{padding:8px;}
           .divFrame .divContent img{border:solid 1px #ccc;padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            var blnShow = false; //初始化一个布尔变量值
            var $Title = $(".divTitle"); //定义变量获取标题部分
            var $Tip = $("#divTip"); //定义变量获取提示元素

            $Title.click(function() { //点击标题部分事件
                if (!blnShow) {
                    //图片高度向上减小，执行完成后，回调一个函数
                    $("img").slideUp(3000, function() {
                        $Tip.html(" 关闭成功！ ");
```

```
                    })
                    $(this).html(" 显示图片 "); // 改变标题内容
                    blnShow = true; // 改变布尔变量值
                }
                else {
                    $Tip.html(""); // 清空提示内容
                    $("img").slideDown(3000); // 图片高度向下增大
                    $(this).html(" 隐藏图片 ");
                    blnShow = false;
                }
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle"> 隐藏图片 </div>
        <div class="divContent">
            <img src="Images/img02.jpg" alt=""/>
            <div id="divTip"></div>
        </div>
    </div>
</body>
</html>
```

(3) 页面效果

代码执行后的效果如图 5-4 所示。

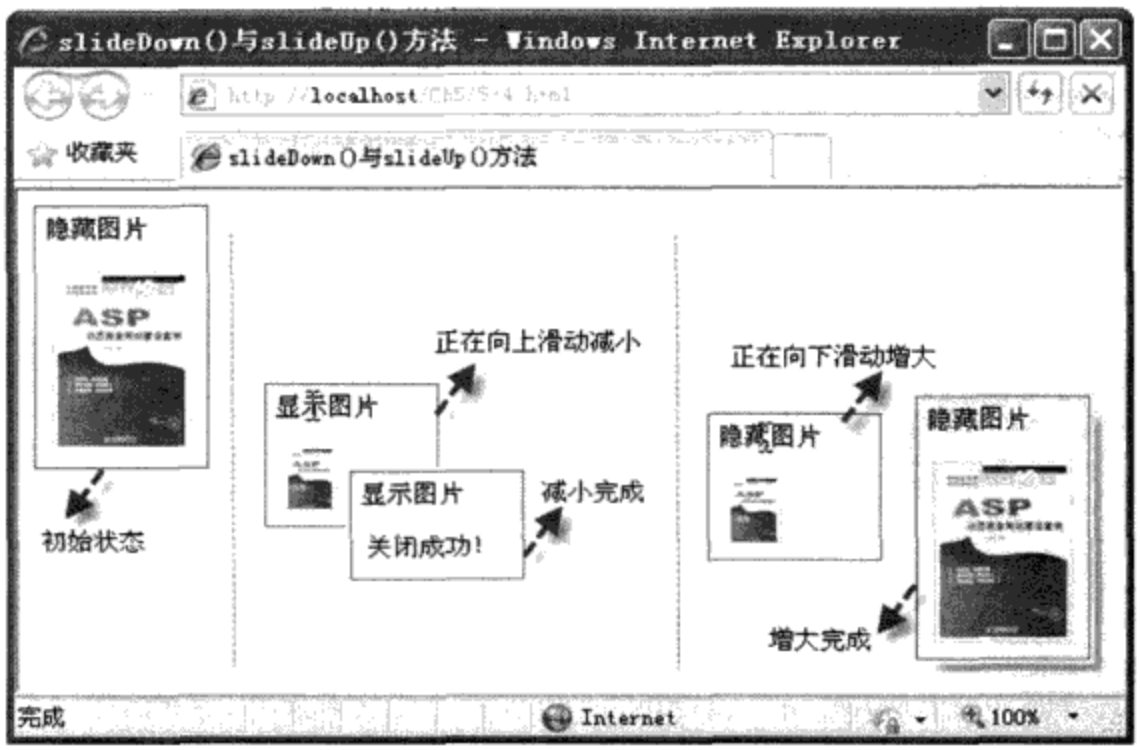


图 5-4　slideDown() 与 slideUp 方法

说明 无论是 slideDown() 还是 slideUp() 方法，它们的动画效果仅是减小或增加元素的高度，同时，如果元素有 margin 或 padding 值，这些属性也会以动画效果一起发生改变。

5.2.2 slideToggle() 方法

在示例 5-4 中，为了判断当前元素的显示状态，先定义了一个变量值 blnShow，根据这个变量值，决定是执行 slideDown() 还是 slideUp() 方法，即元素是向上减小还是向下增大。在 jQuery 中，通过 slideToggle() 方法，无需定义变量，该方法可以根据当前元素的显示状态，自动进行切换，其调用的语法格式如下所示：

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

该方法的功能是以动画的效果切换所选择元素的高度，即：如果高，则减小；如果低，则增大。同时，在每次执行动画完成后，可执行一个用于回调的函数。其包含的参数功能与方法 slideDown() 或 slideUp 一样，在此不再赘述。下面通过示例来介绍这个方法。

示例 5-5 slideToggle() 方法

（1）功能描述

在页面中，使用一个 标记包含一个 <img> 图片，当单击 元素时，通过 slideToggle() 方法，以动画的效果自动切换图片高度状态。

（2）实现代码

新建一个 HTML 文件 5-5.html，加入如代码清单 5-5 所示的代码。

代码清单 5-5 slideToggle() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>slideToggle() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #666;background-color:#eee;
           padding:5px;width:149px}
           .divFrame img{border:solid 1px #eee;padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".divFrame").click(function() { //div 元素点击事件
                // 图片高度状态自动切换，并执行一个回调函数
                $("img").slideToggle(3000, function() {
                    $("img").css("border", "solid 1px #ccc");
                })
```

```
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <img src="Images/img01.jpg" alt="" title="风景图" />
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-5 所示。

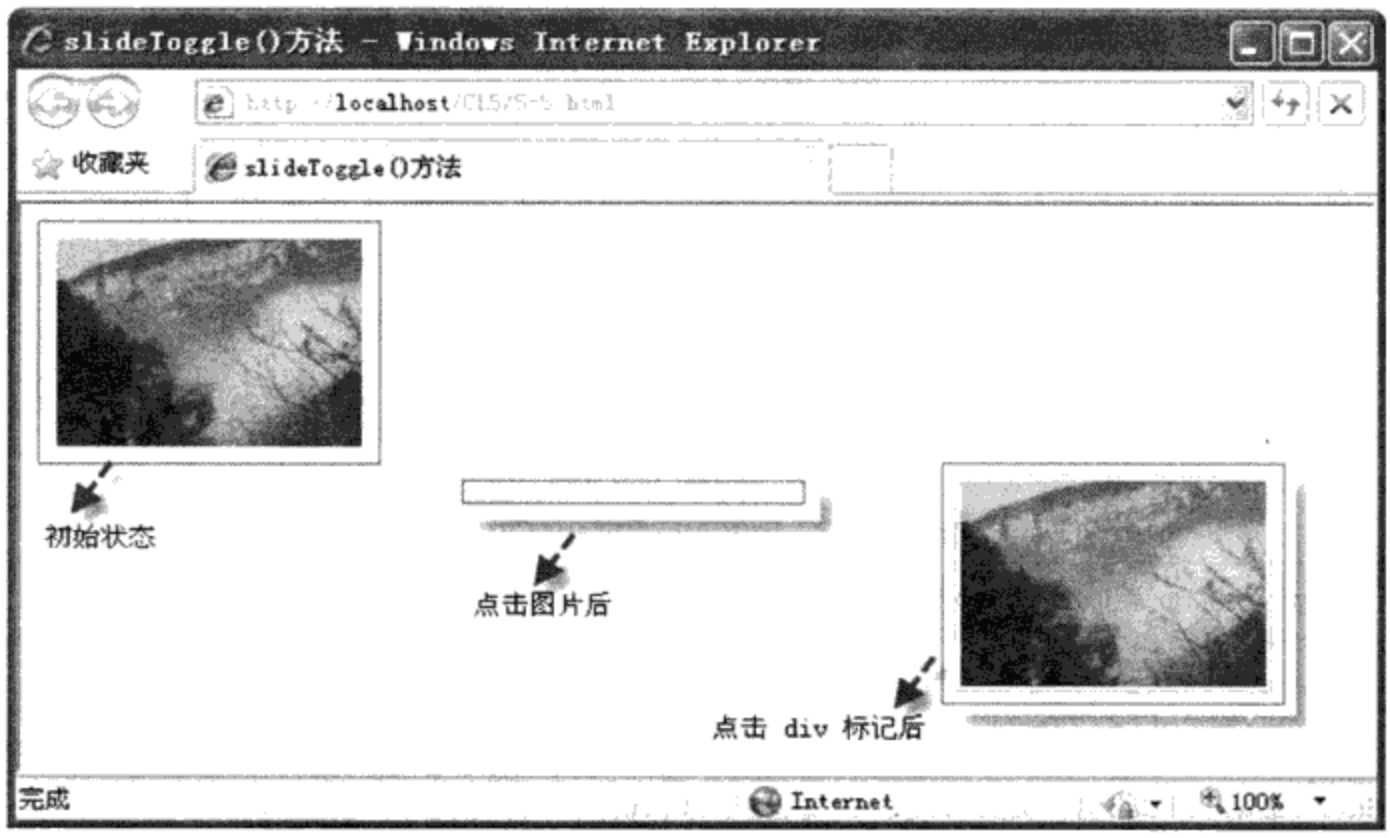


图 5-5　slideToggle() 方法

5.3　淡入淡出

在前两节中，我们介绍了通过动画效果切换元素显示状态和改变元素高度，除上述效果外，在 jQuery 中，还可以实现通过元素渐渐变换背景色的动画效果来显示或隐藏元素，即淡入淡出效果。

5.3.1　fadeIn() 与 fadeOut() 方法

show() 和 hide() 方法与 fadeIn() 和 fadeOut() 方法相比较，相同之处是都切换元素的显示

状态，不同之处在于，前者的动画效果使用元素的 width 与 height 属性都发生了变化，而后者仅是改变元素的透明度，并不修改其他属性。fadeIn() 和 fadeOut() 方法调用格式介绍如下。

fadeIn() 方法的格式如下：

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

该方法的功能是通过改变所选元素透明度，实现淡入的动画效果，并在完成时，可执行一个回调的函数。参数 speed 为动画效果的速度，可选项参数 [callback] 为动画完成时执行的函数。

fadeOut() 方法的格式如下：

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

该方法的功能是通过改变所选元素透明度，实现淡出的动画效果，其包含参数的功能与 fadeIn() 方法一样，在此不再赘述。下面通过示例来讲解这两个方法。

示例 5-6　fadeIn() 和 fadeOut() 方法

（1）功能描述

在页面中，设置两个按钮，一个用于淡入图片，另一个用于淡出图片；同时，无论是淡入还是淡出图片，效果完成后，都执行一个回调函数，显示当前操作的结果。

（2）实现代码

新建一个 HTML 文件 5-6.html, 加入如代码清单 5-6 所示的代码。

代码清单 5-6　fadeIn() 和 fadeOut() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> fadeIn() 和 fadeOut() 方法 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #666;
           width:188px;text-align:center;}
           .divFrame .divTitle{background-color:#eee;
           padding:5px 0px 5px 0px}
           .divFrame .divContent{padding:5px 0px 5px 0px}
           .divFrame .divContent img{border:solid 1px #eee;
           padding:2px}
           .divFrame .divTip{position:absolute;
           padding:90px 0px 0px 60px;font-size:13px;
           font-weight:bold}
           .btn {border:#666 1px solid;padding:2px;width:80px;
```

```
            filter: progid:DXImageTransform.Microsoft
            .Gradient(GradientType=0,StartColorStr=#ffffff,
             EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            $img = $("img"); // 获取图片元素对象
            $tip = $(".divTip"); // 获取提示元素对象
            $("input:eq(0)").click(function() { // 第一个按钮单击事件
                $tip.html(""); // 清空提示内容
                // 在 3000 毫秒中淡入图片，并执行一个回调函数
                $img.fadeIn(3000, function() {
                    $tip.html(" 淡入成功！ ");
                })
            })
            $("input:eq(1)").click(function() { // 第二个按钮单击事件
                $tip.html(""); // 清空提示内容
                // 在 3000 毫秒中淡出图片，并执行一个回调函数
                $img.fadeOut(3000, function() {
                    $tip.html(" 淡出成功！ ");
                })
            })
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <input id="Button1" type="button"
                   value="fadeIn" class="btn" />
            <input id="Button2" type="button"
                   value="fadeOut" class="btn" />
        </div>
        <div class="divContent">
            <div class="divTip"></div>
            <img src="Images/img05.jpg" alt="" title=" 风景图片 " />
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-6 所示。

5.3.2　fadeTo() 方法

在 jQuery 中，fadeIn() 和 fadeOut() 方法通过动画效果，改变元素的透明度切换元素显示状态，其透明度从 0.0 到 1.0 淡出或从 1.0 到 0.0 淡入，从而实现淡入淡出的动画效果；如果

要将透明度指定到某一个值，则需要调用 fadeTo() 方法。其调用的语法格式为：

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

该方法的功能是将所选择元素的不透明度以动画的效果调整到指定的不透明度值，动画完成时，可以执行一个回调函数，参数 speed 为动画效果的速度，参数 opacity 为指定的不透明值，取值范围是 0.0 ～ 1.0，可选项参数 [callback] 为动画完成时执行的函数。下面通过示例来讲解。

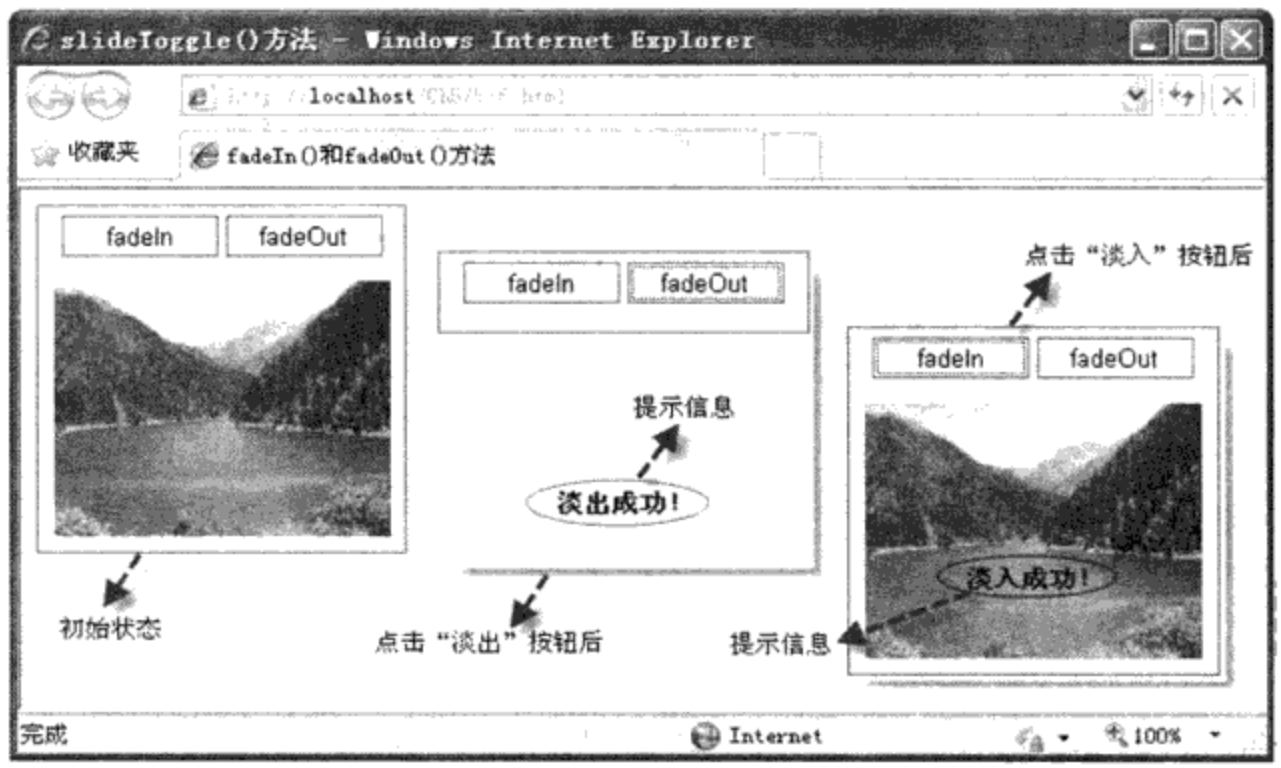


图 5-6　fadeIn() 和 fadeOut() 方法

示例 5-7　fadeTo() 方法

（1）功能描述

为了更好地展示各种透明度值的效果，在页面中创建一个下拉列表框，用于保存各种透明度值，当选择下拉列表框中某选项时，页面中图片的透明度便指定为该选项值。

（2）实现代码

新建一个 HTML 文件 5-7.html, 加入如代码清单 5-7 所示的代码。

代码清单 5-7　fadeTo() 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>fadeTo() 方法 </title>
    <script type="text/javascript"
```

```
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
            .divFrame{border:solid 1px #666;
            width:197px;text-align:center;}
            .divFrame .divTitle{background-color:#eee;
            padding:5px 0px 5px 0px}
            .divFrame .divContent{padding:5px 0px 5px 0px}
            .divFrame .divContent img{border:solid 1px #eee;
            padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            var $img = $("img"); // 获取图片元素对象
            var $sel = $("select");// 获取下拉列表框对象
            $sel.change(function() { // 下拉列表框选项改变事件
                var fltValue = $sel.val(); // 获取选中的值
                $img.fadeTo(3000, fltValue); // 改变图片的透明度
            })
        })
    </script>
</head>
<body>
     <div class="divFrame">
            <div class="divTitle">
                <select id="Select1">
                    <option value="0.2">0.2</option>
                    <option value="0.4">0.4</option>
                    <option value="0.6">0.6</option>
                    <option value="0.8">0.8</option>
                    <option value="1.0"
                            selected="selected">1.0</option>
                </select>
            </div>
            <div class="divContent">
                <img src="Images/img06.jpg" alt="" title=" 风景图片 " />
            </div>
        </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-7 所示。

5.4　自定义动画

上面几节介绍的动画效果都是元素局部属性发生变化，如高度、宽度、可见性等。在

jQuery 中，也允许用户自定义动画效果，通过使用 animate() 方法，可以制作出效果更优雅、动作更复杂的页面动画效果。

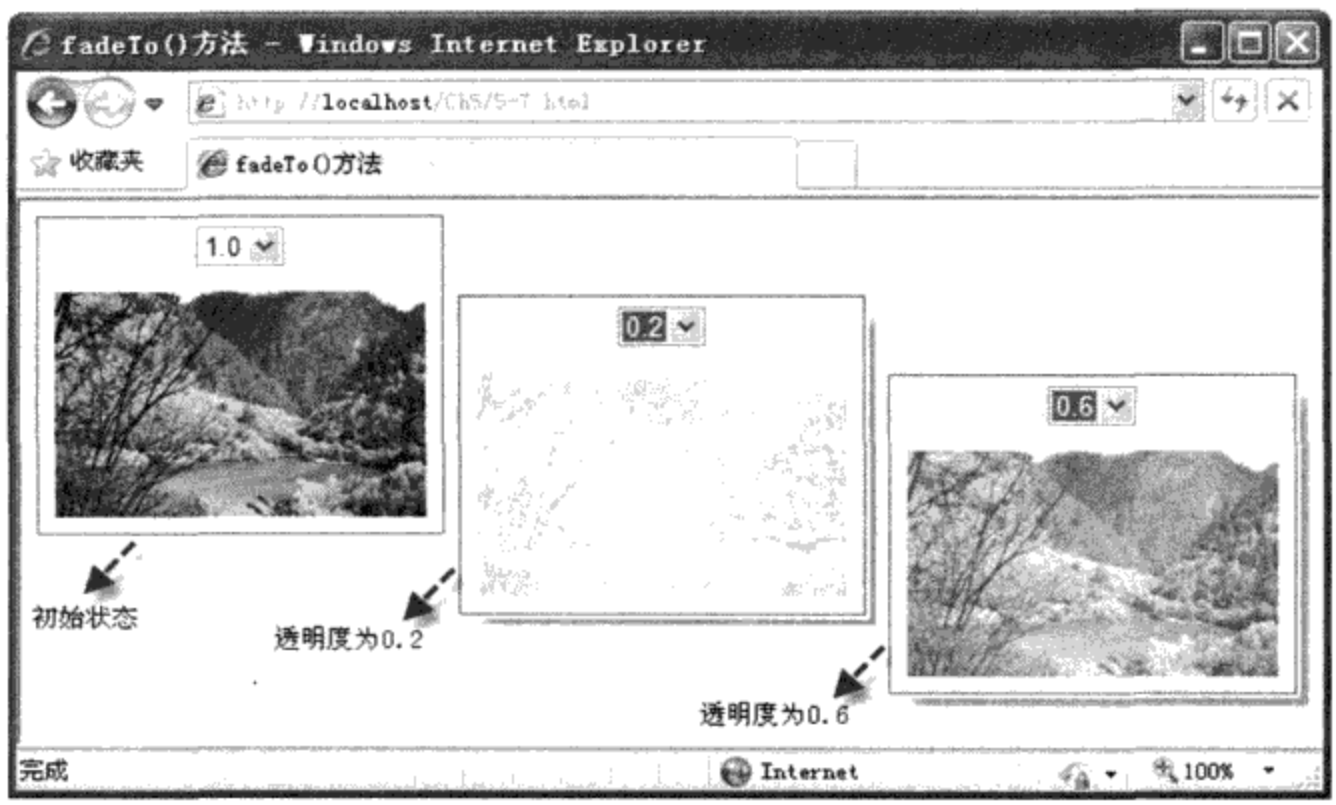


图 5-7 fadeTo() 方法

5.4.1 简单的动画

animate() 方法给开发者自定义各种复杂、高级的动画提供了极大的方便和空间，其调用的语法格式为：

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

其中，参数 params 表示用于制作动画效果的属性样式和值的集合。可选项 [duration] 表示三种默认的速度字符“slow”、“normal”、“fast”或自定义的数字。可选项 [easing] 为动画插件使用，用于控制动画的表现效果，通常有“linear”和“swing”字符值。可选项 [callback] 为动画完成后，执行的回调函数。下面举例说明。

示例 5-8 简单的动画

（1）功能描述

在页面中，单击某块 元素，其自身的高度与宽度以动画的效果增大。动画完成后，元素的边框加粗，并且边框颜色及 元素内容发生变化。

（2）实现代码

新建一个 HTML 文件 5-8.html，加入如代码清单 5-8 所示的代码。

代码清单 5-8　简单的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>简单的动画</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           .divFrame{border:solid 1px #ccc;background-color:#eee;
           width:60px;height:60px;
           font-size:13px;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $(".divFrame").click(function() { //div元素点击事件
                $(this).animate({ //宽与高变化的动画效果
                    width: "20%",
                    height: "70px"
                },
                 3000, function() { //动画完成后执行的回调函数
                     $(this).css({ "border": "solid 3px #666" })
                     .html("变大了！");
                 })
            })
        </script>
</head>
<body>
     <div class="divFrame">
          点击变大
     </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-8 所示。

5.4.2　移动位置的动画

通过 animate() 方法，不仅可以用动画效果增加元素的长与宽，还能以动画效果移动页面中的元素，即改变其相对位置。如，将方法中的参数 params，加入如下代码：

```
$("p").animate({
        left: "20px",
        top: "70px"
```

```
        },
3000)
```

这段代码执行后，页面中的 <p> 元素，在 3000 毫秒内，其对应的位置以动画的效果向右移动 20 像素、向下移动 70 像素。除上述方式移动元素位置外，还可以根据元素当前的位置进行累加或累减的移动。下面通过一个简单的示例说明其使用的方法。

图 5-8　简单的动画

说明　在动画方法 animate() 中，第一个参数 params 在表示动画属性时，需要采用“骆驼”写法，即如果是“font-size”，必须写成“fontSize”才有效，否则报错。

示例 5-9　移动位置的动画

（1）功能描述

在页面中，创建两个按钮，单击第一个“左移”按钮后，将页面中的 元素在当前的位置上，以动画的效果向左移动 52 个像素；单击第二个“右移”按钮后，页面中的 元素在当前的位置上，以动画的效果向右移动 52 个像素。

（2）实现代码

新建一个 HTML 文件 5-9.html, 加入如代码清单 5-9 所示的代码。

代码清单 5-9　移动位置的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>移动位置的动画 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
```

```
</script>
<style type="text/css">
        body{font-size:13px}
        .divFrame{border:solid 1px #666;
        width:168px;text-align:center;}
        .divFrame .divTitle{background-color:#eee;
        padding:5px 0px 5px 0px}
        .divFrame .divContent{width:108px;height:52px;
        padding:5px 0px 5px 0px;margin:0px 30px 0px 30px;
        overflow:hidden}
        .divFrame .divContent .divList{width:162px;
        position:relative}
        .divFrame .divContent .divList span{
        border:solid 1px #ccc;background-color:#eee;width:50px;
        height:50px;float:left;margin-right:2px}
        .btn {border:#666 1px solid;padding:2px;width:60px;
        filter: progid:DXImageTransform.Microsoft
        .Gradient(GradientType=0,StartColorStr=#ffffff,
        EndColorStr=#ECE9D8)}
</style>
<script type="text/javascript">
    $(function() {
        $("input:eq(0)").click(function() { //左移按钮点击事件
            //在3000毫秒内，以动画的形式向左移动52个像素
            $(".divList").animate({ left: "-=52px" }, 3000);
        })
        $("input:eq(1)").click(function() { //右移按钮点击事件
            //在3000毫秒内，以动画的形式向右移动52个像素
            $(".divList").animate({ left: "+=52px" }, 3000);
        })
    })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <input id="Button1" type="button"
                value="左移" class="btn" /> 
            <input id="Button2" type="button"
                value="右移" class="btn" />
        </div>
        <div class="divContent">
            <div class="divList">
                <span>1</span>
                <span>2</span>
                <span>3</span>
            </div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-9 所示。

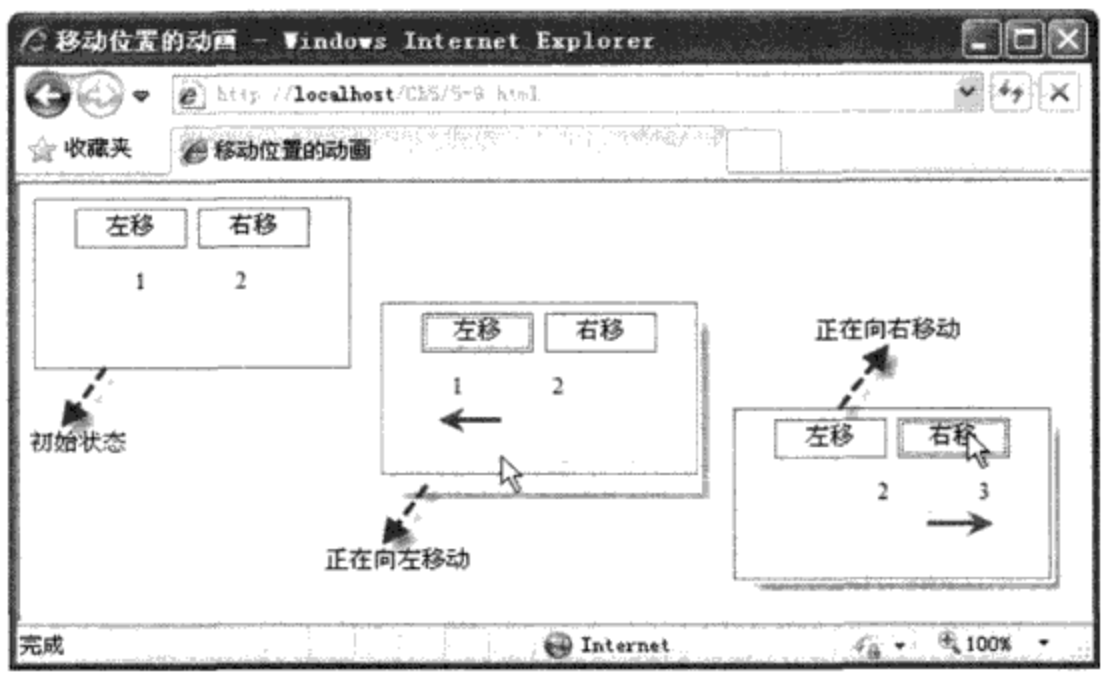


图 5-9　移动位置的动画

说明　要使页面中的元素以动画效果移动，必须首先将该元素的“position”属性设置成为“relative”或“absolute”，否则，无法移动该元素的位置。

5.4.3　队列中的动画

所谓“队列”动画，是指在元素中执行一个以上的多个动画效果，即有多个 animate() 方法在元素中执行，因此，根据这些 animate() 方法执行的先后顺序，形成了动画“队列”，产生“队列”后，动画的效果便按“队列”的顺序进行展示。下面举例说明。

示例 5-10　队列中的动画

（1）功能描述

在页面中，单击某个固定宽、高的 标记，首先，以动画的效果，将宽、高在原有基础上加大 1 倍；然后，再以动画的效果，将宽、高在原有基础上减小 1 倍。

（2）实现代码

新建一个 HTML 文件 5-10.html, 加入如代码清单 5-10 所示的代码。

代码清单 5-10　队列中的动画

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>队列中的动画 </title>
    <script type="text/javascript"
```

```
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           div{border:solid 1px #666;background-color:#eee;
               width:50px;height:50px;font-size:13px;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("div").click(function() { //div 块单击事件
                $(this)
             .animate({ height: 100 }, "slow") // 第 1 列
             .animate({ width: 100 }, "slow") // 第 2 列
             .animate({ height: 50 }, "slow") // 第 3 列
             .animate({ width: 50 }, "slow"); // 第 4 列
            })
        })
    </script>
</head>
<body>
     <div> 队列 </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-10 所示。

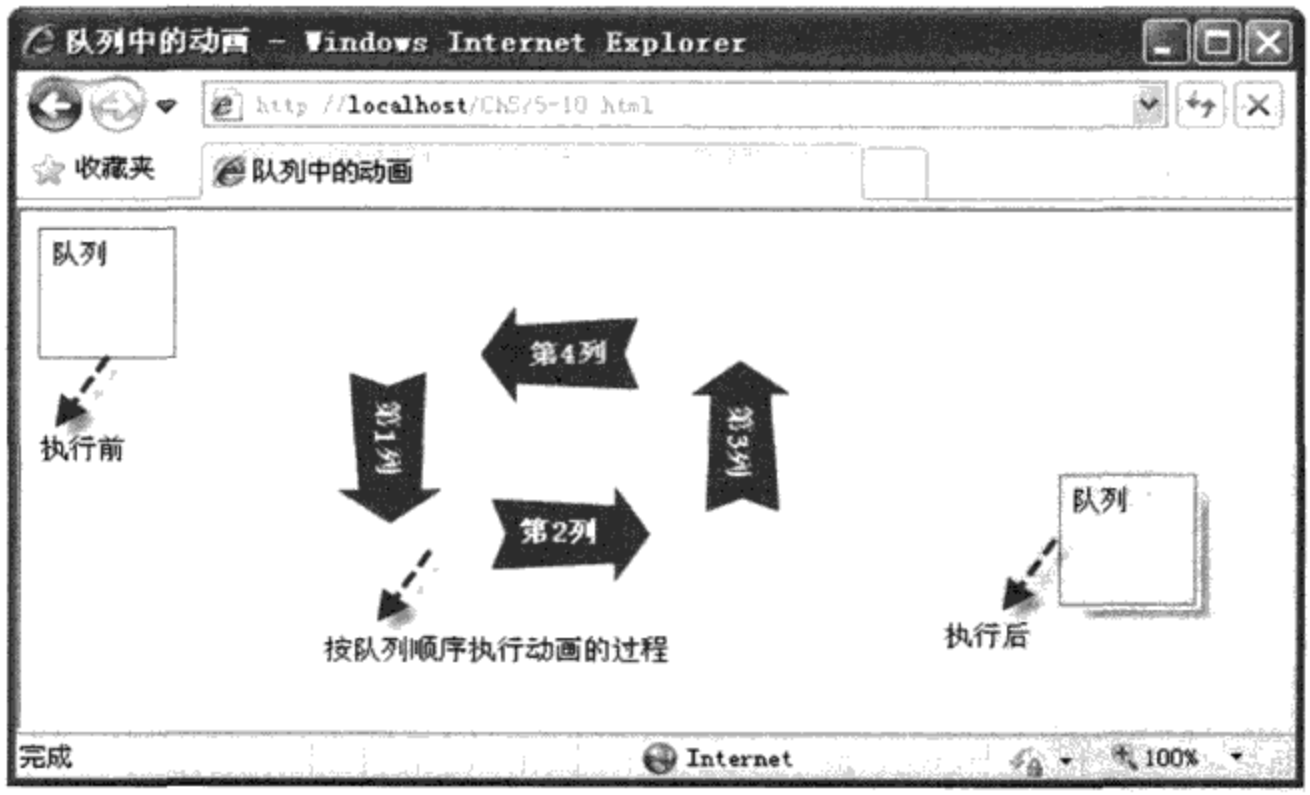


图 5-10　队列中的动画

说明　理解动画队列执行的过程后，可以清楚分析各队列中各动画的效果，并可以在指定的某队列中插入其他方法，如队列延时方法 delay()。该方法的运用将在下节中介绍。

5.4.4 动画停止和延时

在 jQuery 中，通过 animate() 可以实现元素的动画显示，但在显示的过程中，必然要考虑各种客观因素和限制性条件的存在，因此，在执行动画时，可通过 stop() 方法停止或 delay() 方法延时某个动画的执行。stop() 与 delay() 方法的语法调用格式介绍如下。

stop() 方法的格式如下：

```
stop([clearQueue], [gotoEnd])
```

该方法的功能是停止所选元素中正在执行的动画，其中可选参数 [clearQueue] 是一个布尔值，表示是否停止正在执行的动画，另外一个可选参数 [gotoEnd] 也是一个布尔值，表示是否立即完成正在执行的动画。

delay () 方法的格式如下：

```
delay(duration, [queueName])
```

该方法的功能是设置一个延时值来推迟后续队列中动画的执行，其中参数 duration 为延时的时间值，单位是毫秒。可选参数 [queueName] 表示队列名词，即动画队列。

下面举例说明。

示例 5-11 动画停止和延时

（1）功能描述

在页面中，设置三个超级链接，分别为“开始”、“停止”、“延时”，单击“开始”后，页面中的图片以“拉窗帘”的方式动画切换显示状态；单击“停止”后，立刻停止了正在执行中的动画效果；单击“延时”后，动画切换显示效果在延时 2000 毫秒后，再执行。

（2）实现代码

新建一个 HTML 文件 5-11.html，加入如代码如清单 5-11 所示的代码。

代码清单 5-11 动画停止和延时

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画停止和延时</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .divFrame{border:solid 1px #666;
          width:233px;text-align:center;}
          .divFrame .divTitle{background-color:#eee;
          padding:5px 0px 5px 0px}
          .divFrame .divContent{padding:5px 0px 5px 0px}
```

```
            .divFrame .divContent img{border:solid 1px #eee;
            padding:2px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("a:eq(0)").click(function() { // 点击 " 开始 " 链接事件
                $("img").slideToggle(3000); //" 拉窗帘 " 方式切换图片
            })
            $("a:eq(1)").click(function() { // 点击 " 停止 " 链接事件
                $("img").stop(); // 停止正在执行的动画
            })
            $("a:eq(2)").click(function() { // 点击 " 延时 " 链接事件
                $("img") // 延时切换图片
                .delay(2000)
                .slideToggle(3000);
            })
        })
    </script>
</head>
<body>
     <div class="divFrame">
           <div class="divTitle">
                <a href="javascript:void(0)"> 开始 </a>
                  |  
                <a href="javascript:void(0)"> 停止 </a>
                  |  
                <a href="javascript:void(0)"> 延时 </a>
           </div>
           <div class="divContent">
               <img src="Images/img07.gif" alt="" title="Log 图片 " />
           </div>
      </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 5-11 所示。

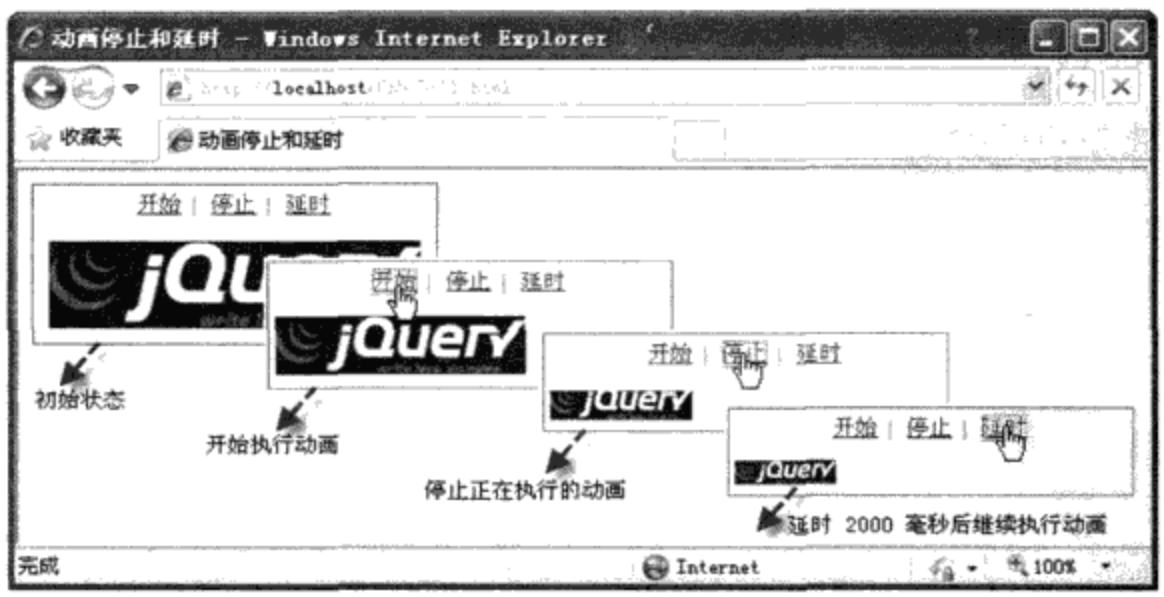


图 5-11　动画停止和延时

5.5 动画效果综述

在 jQuery 中，虽然有很多动画方法，实现各种各样的动画效果，但综合起来，绝大部分的动画方法仅仅是改变元素的属性或样式，如高度、宽度、透明度和 CSS 样式属性，详细说明如下：

5.5.1 各种动画方法说明

1）show() 与 hide() 方法，元素以动画效果实现显示与隐藏，可以同时改变元素的多个属性，如宽度、高度、透明度。

2）fadeIn() 与 fadeOut() 方法，元素以动画的效果淡入与淡出，仅改变元素的透明度。

3）slideUp() 与 slideDown() 方法，元素以“卷窗帘”的动画效果显示与隐藏，仅改变元素的高度，其他属性不发生变化。

4）fadeTo() 方法，元素按指定的透明度进行渐进式调整，从而达到一种动画效果，该方法改变的是元素的透明度，宽度与高度都不发生变化。

5）toggle() 方法，可以代替 show() 与 hide() 两个方法，因此，其改变的元素属性也与 show() 与 hide() 方法一样。

6）slideToggle() 方法，可以代替 slideUp() 与 slideDown() 两个方法，改变的元素也与 slideUp() 和 slideDown() 方法一样。

7）animate() 方法，自定义元素的动画效果，可以实现上述 6 种方法中全部属性改变的功能，同时，还可以用动画的效果，改变其他的元素属性，该方法是上述 6 种方法的核心。

5.5.2 使用 animate() 方法代替其他动画效果

animate() 方法不仅可以使元素实现各种各样的动画效果，还可以代替其他的动画方法，详细代码如下所示：

1）animate() 方法代替 hide() 方法，代码如下：

```
$("页面元素").animate({ height: "hide", width: "hide", opacity: "hide" }, 600);
```

等价于下列代码：

```
$("页面元素").hide(600);
```

2）animate() 方法代替 fadeOut () 方法，代码如下：

```
$("页面元素").animate({ opacity: "hide" }, 600);
```

等价于下列代码：

```
$("页面元素").fadeOut(600);
```

3）animate() 方法代替 slideUp () 方法，代码如下：

```
$("页面元素").animate({ height: "hide" }, 600);
```

等价于下列代码：

```
$("页面元素").slideUp(600);
```

4）animate() 方法代替 fadeTo () 方法，代码如下：

```
$("页面元素").animate({ opacity: "0.8" }, 600);
```

等价于下列代码：

```
$("页面元素").fadeTo(600, "0.8");
```

5.6　综合案例分析——动画效果浏览相册中的图片

5.6.1　需求分析

经分析，该案例的需求如下：

1）将尺寸不同的相片统一成宽度与高度相同的缩略图片，展示在页面中。

2）当单击“点击放大”链接时，以动画的效果放大至其原始图片，同时，隐藏“点击放大”链接，显示该图的基本信息。

3）当点击原始图片中的“关闭”按钮时，以动画的效果将图片缩小成单击前的缩略图，即返回到图片初始状态。

4）在浏览放大后的原始图片时，又单击其他缩略图片，那么，处于放大状态的原始图片自动以动画的效果进行关闭，使得整个相册始终只有一个图片处于放大状态。

5.6.2　效果界面

该案例的效果界面如下所示：

1）将不同尺寸的相片形成统一的缩略图片，显示在页面中，即相册进行初始化，如图 5-12 所示。

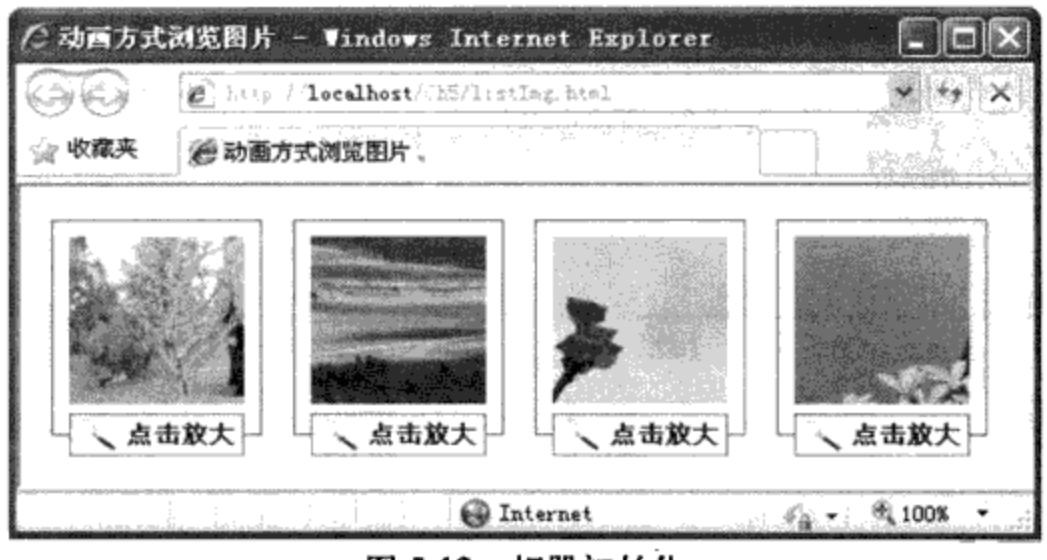


图 5-12　相册初始化

2）当单击某一个图片的“点击放大”链接后，该图片以动画的效果进行放大，并展示该图片的基本信息，如图 5-13 所示。

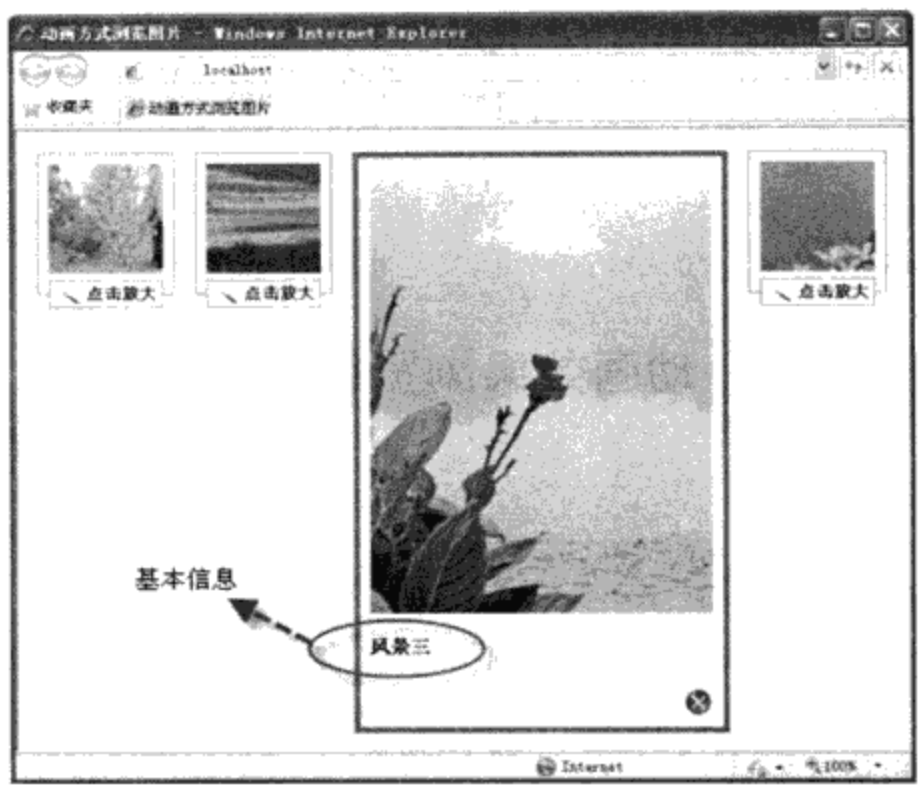


图 5-13　图片放大后的效果

3）在图 5-13 中，如果再单击其他图片的“点击放大”链接时，将自动以动画的效果缩小当前已放大的图片，并同时放大单击过的图片，其切换状态如图 5-14 所示。

图 5-14　自动以动画效果缩小已放大的图片

4）如果单击放大后的图片中“关闭”链接时，将以动画效果缩小该图片，其效果如图 5-15 所示。

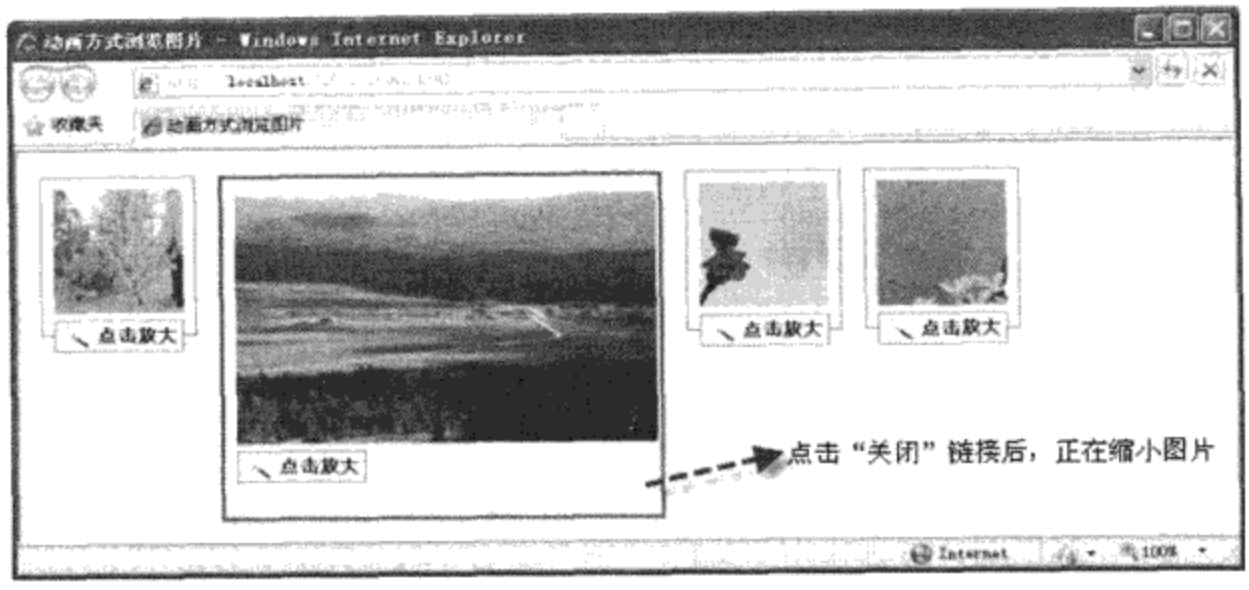


图 5-15　单击“关闭”链接后，以动画效果缩小图片

5.6.3　功能实现

在该项目中，新建一个 HTML 文件 listImg.html, 加入如代码清单 5-12 所示的代码。

代码清单 5-12　相册中的动画效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>动画方式浏览图片</title>
    <link type="text/css" href="Css/css_Animate.css"
          rel="Stylesheet" />
    <script type="text/javascript"
            src="Jscript/js_Animate.js"></script>
</head>
<body>
    <div class="p_Lst">
            <img  alt=""src="images/list_1.jpg" />
            <div class="p_Alt">
                    <h3>风景一</h3>
            </div>
    </div>
    <div class="p_Lst">
         <img  alt=""src="images/list_2.jpg"/>
         <div class="p_Alt">
                 <h3>风景二</h3>
         </div>
     </div>
     <div class="p_Lst">
            <img  alt=""src="images/list_3.jpg"/>
            <div class="p_Alt">
```

```
                    <h3> 风景三 </h3>
            </div>
    </div>
    <div class="p_Lst">
            <img  alt=""src="images/list_4.jpg"/>
            <div class="p_Alt">
                    <h3> 风景四 </h3>
            </div>
    </div>
</body>
</html>
```

为了代码的调用方便和后续维护的快捷，在 HTML 中，单独调用了两个辅助文件，一个是用于控制页面样式的 CSS 文件 css_Animate.css，另外一个是用于实现页面功能的 JS 文件 js_Animate.js。这两个文件的代码参见代码清单 5-13 和 5-14。

代码清单 5-13　CSS 文件 css_Animate.css

```
body {font-size:13px}
/* 图片外框样式 */
.p_Lst {
   position: relative;
   float: left;
   width: 90px;
   height: 98px;
   padding: 8px;
   border: 1px solid #666;
   margin: 10px 8px 20px 8px;
}
/* 图片最近外框样式 */
.p_Img {
   width: 90px;
   height: 90px;
   margin-bottom: 5px;
   overflow: hidden;
}
/* 图片信息样式 */
.p_Alt {
   display:none;
}
/* 缩略图片中 " 点击放大 " 链接样式 */
.p_Big {
   display: block;
   width: 90px;
   height: 23px;
   background: url(../Images/imgLarge.jpg);
   cursor: pointer;
}
/* 原始放大图片中 " 关闭 " 按钮样式 */
```

```
.p_Cls {
    position: absolute;
   right: 10px;
   bottom: 10px;
   display: block;
   width: 20px;
   height: 21px;
   background: url(../Images/imgClose.jpg);
   text-indent: -9999px;
}
```

代码清单 5-14　JS 文件 js_Animate.js

```
/// <reference path="jquery-1.4.2.js"/>

$(function() {
    var curIndex = -1; // 初始化当前打开图片值
    var intImgL = "-120px";
    var intImgT = "-120px";
    // 带参数 index 遍历图片外框 Div
    $(".p_Lst").each(function(index) {
        var $this = $(this); // 获取每个外框 Div
        var $img = $this.find("img"); // 查找其中的图片元素
        var $info = $this.find(".p_Alt"); // 查询其中的图片信息元素
        var arrPic = {}; // 定义一个空数组保存初始的长与宽
        arrPic.imgw = $img.width();
        arrPic.imgh = $img.height();
        arrPic.orgw = $this.width();
        arrPic.orgh = $this.height();
        $img.css({ // 设置初始时的图片外边距位置
            marginLeft: intImgL,
            marginTop: intImgT
        });
        // 将图片、点击放大链接、关闭按钮放入外框 Div 中
        var $drag = $("<div class='p_Img'>")
                    .append($img).prependTo($this);
        var $open = $("<a href='javascript:void(0)' class='p_Big'
                    title=' 点击放大 '></a>").appendTo($this);
        var $clos = $("<a href='javascript:void(0)' class='p_Cls'
                    title=' 点击关闭 '></a>").appendTo($info);
        // 保存放入元素后的外框 Div 的长与宽
        arrPic.dragw = $drag.width();
        arrPic.dragh = $drag.height();

        // 放大按钮单击事件
        $open.click(function() {
            $this.animate({ // 外框动画
                width: arrPic.imgw,
                height: (arrPic.imgh + 85), //85 是图片信息的高度，
                borderWidth: "5"
```

```
        }, 3000);
        $open.fadeOut(); // 点击放大链接淡出
        $(".p_Alt", $this).fadeIn(); // 图片提示信息淡入
        $drag.animate({ // 加入图片后的 Div 框动画
            width: arrPic.imgw,
            height: arrPic.imgh
        }, 3000);
        $img.animate({ // 以动画的形式自动调整位置
            marginTop: "0px",
            marginLeft: "0px"
        }, 3000);
        // 获取当前被放大成原始图的图片各组成部分
        var $f_this = $(".p_Lst:eq(" + curIndex + ")");
        var $f_open = $(".p_Big:eq(" + curIndex + ")");
        var $f_drag = $(".p_Img:eq(" + curIndex + ")");
        var $f_larg = $(".p_Alt:eq(" + curIndex + ")");
        var $f_imgs = $("img:eq(" + curIndex + ")");
        if (curIndex != -1) { // 如果当前有已放大的图片
            // 自动以动画的形式关闭该图片
            cls_Click($f_this, $f_open, $f_drag, $f_imgs,$f_larg);
        }
        // 重新获取当前放大图片的索引号
        curIndex = index;
    });

    // 关闭按钮单击事件
    $clos.click(function() {
        // 以动画的形式缩小当前所点击的图片
        cls_Click($this, $open, $drag, $img, 1);
        // 初始化索引号
        curIndex = -1;
    });

    /*
    *@param 参数 pF 表示图片最外层 Div
    *@param 参数 pO 表示图片点击前的放大按钮
    *@param 参数 pW 表示紧邻图片层 Div
    *@param 参数 pI 表示紧选中的图片元素
    *@param 参数 blnS 表示图片中的说明内容
    */
    function cls_Click(pF, pO, pW, pI, blnS) {
        var $strInit;
        pF.animate({
            width: arrPic.orgw,
            height: arrPic.orgh,
            borderWidth: "1"
        }, 3000);
        pO.fadeIn();
        if (blnS) {
            $strInit = $(".p_Alt", pF);
```

```
                } else {
                    $strInit = blnS;
                }
                $strInit.fadeOut();
                pW.animate({
                    width: arrPic.dragw,
                    height: arrPic.dragh
                }, 3000);
                pI.animate({
                    marginTop: intImgT,
                    marginLeft: intImgL
                }, 3000);
            }
        });
    })
```

5.6.4　代码分析

在 JS 文件 js_Animate.js 中，为了实现各相册中的图片单击放大查看的动画效果，先带参数 index 遍历整个图片最外框的 Div 元素，其代码如下：

```
$(".p_Lst").each(function(index) {
// 执行代码部分
...
})
```

在遍历过程中，先获取设置好的外框与图片的长与宽，并将改变后的图片和相关元素先放入类别为“p_Img”的 Div 中，然后全部放入最外框的 Div 元素中，从而完成图片初始化的页面效果。其代码如下：

```
... 省略部分代码
var $this = $(this); // 获取每个外框 Div
var $img = $this.find("img"); // 查找其中的图片元素
var $info = $this.find(".p_Alt"); // 查询其中的图片信息元素
var arrPic = {}; // 定义一个空数组保存初始的长与宽
arrPic.imgw = $img.width();
arrPic.imgh = $img.height();
arrPic.orgw = $this.width();
arrPic.orgh = $this.height();
$img.css({ // 设置初始时的图片外边距位置
    marginLeft: intImgL,
    marginTop: intImgT
});
// 将图片、点击放大链接、关闭按钮放入外框 Div 中
var $drag = $("<div class='p_Img'>")
            .append($img).prependTo($this);
var $open = $("<a href='javascript:void(0)' class='p_Big'
            title=' 点击放大 '></a>").appendTo($this);
var $clos = $("<a href='javascript:void(0)' class='p_Cls'
```

```
            title=' 点击关闭 '></a>").appendTo($info);
//保存放入元素后的外框 Div 的长与宽
arrPic.dragw = $drag.width();
arrPic.dragh = $drag.height();
... 省略部分代码
```

当单击图片初始状态中的“点击放大”链接时，首先，以 3000 毫秒的速度放大图片最外框 Div 元素，接着以同样的速度放大最近一层包含图片的 Div，与此同时，以同样的速度显示或隐藏相应的页面内容，其代码如下：

```
... 省略部分代码
$this.animate({ //外框动画
    width: arrPic.imgw,
    height: (arrPic.imgh + 85), //85 是图片信息的高度，
    borderWidth: "5"
}, 3000);
$open.fadeOut(); //点击放大链接淡出
$(".p_Alt", $this).fadeIn(); //图片提示信息淡入
$drag.animate({ //加入图片后的 Div 框动画
    width: arrPic.imgw,
    height: arrPic.imgh
}, 3000);
$img.animate({ //以动画的形式自动调整位置
    marginTop: "0px",
    marginLeft: "0px"
}, 3000);
... 省略部分代码
```

为了在单击“点击放大”链接时，缩小正在放大的图片，使仅有一个图片处于放大显示状态中，首先，需要获取当前处于放大显示图片的索引号，然后，根据该索引号获取图片的各个组成部分，再将这些部分以动画效果进行缩小。

根据上述原理，定义一个公用变量 curIndex，用于记录当前放大显示的图片索引号，并赋初始值为 −1，当首次单击“点击放大”链接时，由于 curIndex 等于 −1，因此不执行缩小的操作。这时，再通过下面代码记录当前放大的图片索引号：

```
curIndex = index;
```

当再次单击另一个“点击放大”链接时，将根据该索引号，获取图片的各个组成部分，并执行缩小的动画效果操作，其代码如下：

```
... 省略部分代码
//获取当前被放大成原始图的图片各组成部分
var $f_this = $(".p_Lst:eq(" + curIndex + ")");
var $f_open = $(".p_Big:eq(" + curIndex + ")");
var $f_drag = $(".p_Img:eq(" + curIndex + ")");
var $f_larg = $(".p_Alt:eq(" + curIndex + ")");
var $f_imgs = $("img:eq(" + curIndex + ")");
if (curIndex != -1) { //如果当前有已放大的图片
```

```
        // 自动以动画的形式关闭该图片
        cls_Click($f_this, $f_open, $f_drag, $f_imgs, $f_larg);
    }
    ... 省略部分代码
```

当在原始放大图片中，单击“关闭”按钮时，先执行缩小操作，再将变量 curIndex 赋值为 −1，表示没有放大的原始图片，其代码如下：

```
... 省略部分代码
// 以动画的形式缩小当前所点击的图片
cls_Click($this, $open, $drag, $img, 1);
// 初始化索引号
curIndex = -1;
... 省略部分代码
```

由于图片在放大与缩小时，都执行某项同样的操作，因此，将该项操作自定义为一个函数 cls_Click()。该函数的功能是根据图片的各个组成部分，以同样的速度，采用动画的效果，缩小所选的图片，其代码如下所示：

```
... 省略部分代码
/*
*@param 参数 pF 表示图片最外层 Div
*@param 参数 pO 表示图片点击前的放大按钮
*@param 参数 pW 表示紧邻图片层 Div
*@param 参数 pI 表示紧选中的图片元素
*@param 参数 blnS 表示图片中的说明内容
*/
function cls_Click(pF, pO, pW, pI, blnS) {
    var $strInit;
    pF.animate({
        width: arrPic.orgw,
        height: arrPic.orgh,
        borderWidth: "1"
    }, 3000);
    pO.fadeIn();
    if (blnS) {
        $strInit = $(".p_Alt", pF);
    } else {
        $strInit = blnS;
    }
    $strInit.fadeOut();
    pW.animate({
        width: arrPic.dragw,
        height: arrPic.dragh
    }, 3000);
    pI.animate({
        marginTop: intImgT,
        marginLeft: intImgL
    }, 3000);
```

```
    }
    ...省略部分代码
```

5.7 本章小结

动画效果无疑是众多 jQuery 爱好者最爱之一，在本章中，通过介绍 jQuery 中常用动画效果的基本语法、简单实例，使读者了解和掌握 jQuery 动画的基本方法，为开发出更优雅、高效的页面打下扎实的语言基础。

第 6 章

Ajax 在 jQuery 中的应用

本章内容

- 加载异步数据
- 请求服务器数据
- $.ajax() 方法
- Ajax 中的全局事件
- 综合案例分析——用 Ajax 实现新闻点评即时更新
- 本章小结

Ajax 是 Asynchronous JavaScript and XML 的缩写，其核心是通过 XMLHttpRequest 对象，以一种异步的方式，向服务器发送数据请求，并通过该对象接收请求返回的数据，从而完成人机交互的数据操作。这种利用 Ajax 技术进行的数据交互并不局限于 Web 动态页面，在普通的静态 HTML 页面中同样可以实现，因此，在这样的背景下，Ajax 技术在页面开发中得以广泛使用。在 jQuery 中，同样有大量的函数与方法为 Ajax 技术提供支持。

6.1 加载异步数据

在页面开发过程中，为了加快整体页面打开的速度，对于某局部的数据采用异步读取（Ajax 技术）的方法获取，这一方法的应用，极大地优化了用户的体验，优化了页面的执行。

6.1.1 传统的 JavaScript 方法

抛开 jQuery，使用传统的 JavaScript 方法，基于 XMLHttpRequest 对象，也可以将数据加载到页面中。下面通过一个简单的示例来说明这一方法实现的过程。

示例 6-1 传统的 JavaScript 方法实现 Ajax 功能

（1）功能描述

创建两个新页面 a.html 和 b.html，前者用于加载页，在该页中，单击“加载”按钮后，在不刷新该页面的情况下，将 b.html 页中的内容显示在 a.html 页面的 标记中；后者用于被加载页，在该页中，通过 标记包含一些文字内容。

（2）实现代码

新建一个 HTML 文件 a.html 和 b.html，加入如代码清单 6-1a 和代码清单 6-1b 所示的代码。

代码清单 6-1a 加载页面，用于显示内容

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>传统的 JavaScript 方法实现 Ajax 功能 </title>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:260px;border:solid 1px #666}
           .divFrame .divTitle{padding:5px;background-color:#eee}
           .divFrame .divContent{padding:8px}
           .divFrame .divContent .clsShow{font-size:14px}
           .btn {border:#666 1px solid;padding:2px;width:80px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
```

```
<script type="text/javascript">
    var objXmlHttp = null; // 声明一个空的 XMLHTTP 变量
    function CreateXMLHTTP() {
        // 根据浏览器的不同，返回该变量的实体对象
        if (window.ActiveXObject) {
           objXmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        else {
           if (window.XMLHttpRequest) {
                objXmlHttp = new XMLHttpRequest();
           }
           else {
                alert("初始化 XMLHTTP 错误！ ");
           }
        }
    }
    function GetSendData() {
        document.getElementById("divTip").innerHTML =
        "<img alt='' title='正在加载中 ...'
        src='Images/Loading.gif' />"; // 初始化内容
        // 设置发送地址变量并赋初始值
        var strSendUrl = "b.html?date="+Date();
        CreateXMLHTTP(); // 实例化 XMLHttpRequest 对象
        //open 方法初始化 XMLHttpRequest
        objXmlHttp.open("GET", strSendUrl, true);
        objXmlHttp.onreadystatechange =
            function() { // 回调事件函数
            if (objXmlHttp.readyState == 4) { // 如果请求完成加载
                if (objXmlHttp.status == 200) { // 如果响应已成功
                    // 显示获取的数据
                    document.getElementById("divTip")
                    .innerHTML = objXmlHttp.responseText;
                }
            }
        }
        objXmlHttp.send(null); //send 发送设置的请求
    }
    </script>
</head>
<body>
  <div class="divFrame">
        <div class="divTitle">
            <input id="Button1" type="button"
            onclick="GetSendData()" class="btn" value=" 获取数据 " />
        </div>
        <div class="divContent">
            <div id="divTip"></div>
        </div>
    </div>
</body>
</html>
```

代码清单 6-1b　被加载页面用于设置内容

```
<div class="clsShow">
姓名：陶国荣 <br />
性别：男 <br />
邮箱：tao_guo_rong@163.com
</div>
```

（3）页面效果

代码执行后的效果如图 6-1 所示。

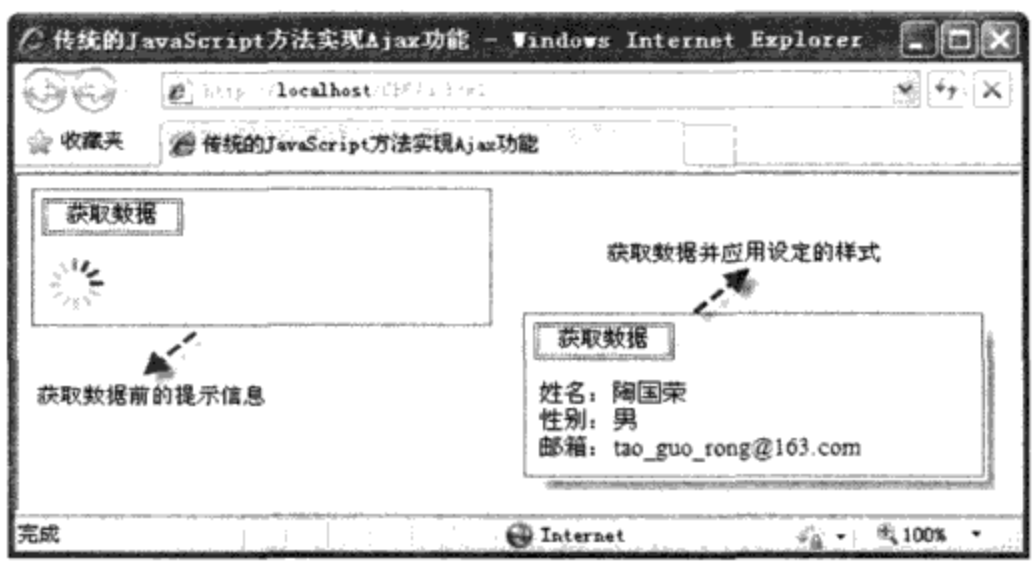


图 6-1　传统的 JavaScript 方法实现 Ajax 功能

说明　在 a.html 页面的 JavaSrcipt 代码中，为了能即时获取被加载页面 b.html 变化了的数据，在设置发送地址 Url 时，后面跟有一参数 date，即 "b.html?date="+Date()，功能就是清空缓存中已加载的变量数据信息，重新获取新的即时数据。

6.1.2　jQuery 中的 load() 方法

在传统的 JavaScript 中，使用 XMLHttpRequest 对象异步加载数据；而在 jQuery 中，使用 load() 方法可以轻松实现获取异步数据的功能。其调用的语法格式为：

```
load(url, [data], [callback])
```

其中参数 url 为被加载的页面地址，可选项 [data] 参数表示发送到服务器的数据，其格式为 key/value。另一个可选项 [callback] 参数表示加载成功后，返回至加载页的回调函数。下面举例说明。

示例 6-2　load() 方法实现异步获取数据

（1）功能描述

使用 load() 方法实现示例 6-1 的功能。

（2）实现代码

新建一个 HTML 文件 6-2.html, 加入如代码清单 6-2 所示的代码。

代码清单 6-2　load() 方法实现异步获取数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>load() 方法实现 Ajax 功能 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮点击事件
                $("#divTip").load("b.html"); //load() 方法加载数据
            })
        })
    </script>
</head>
<body>
   <div class="divFrame">
         ... 省略主体部分代码
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 6-2 所示。

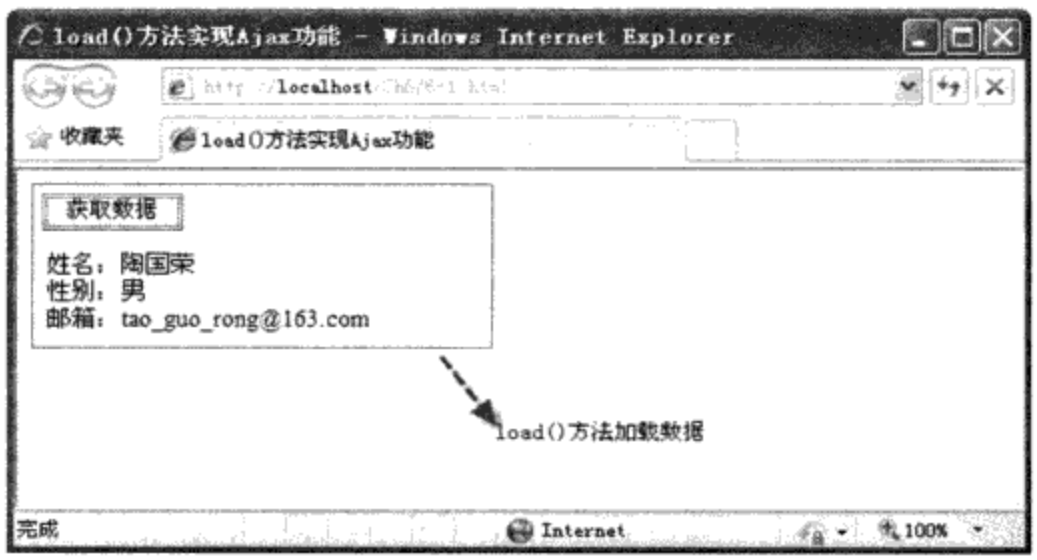


图 6-2　load() 方法实现异步获取数据

说明　在 load () 方法中，参数 url 还可以用于过滤页面中的某类别的元素，如代码“$("#divTip") .load("b.html .divContent")”，则表示获取页面 b.html 中类别名为“ivContent”的全部元素。

6.1.3 jQuery 中的全局函数 getJSON()

虽然使用 load() 方法可以很快地加载数据到页面中，但有时需要对获取的数据进行处理，如果将用 load() 方法获取的内容进行遍历，也可以进行数据的处理，但这样获取的内容必须先插入页面中，然后才能进行，因此，执行的效率不高。

为了解决这个问题，我们采用将要获取的数据另存为一种轻量极的数据交互格式，即 JSON 文件格式，由于这种格式很方便计算机的读取，因而颇受开发者的青睐。在 jQuery 中，专门有一个全局函数 getJSON()，用于调用 JSON 格式的数据，其调用的语法格式为：

```
$.getJSON(url, [data], [callback])
```

参数 url 表示请求的地址，可选项 [data] 参数表示发送到服务器的数据，其格式为 key/value。另外一个可选项 [callback] 参数表示加载成功时执行的回调函数。下面举例说明。

示例 6-3 全局函数 getJSON() 实现异步获取数据

（1）功能描述

创建一个 JSON 格式的文件 UserInfo.json，用于保存人员资料信息，另外，新建一个页面，当单击页面中的“获取数据”按钮时，将通过全局函数 getJSON() 获取文件 UserInfo.json 中的全部数据，并展示在页面中。

（2）实现代码

新建一个 HTML 文件 6-2.html, 加入如代码清单 6-3 所示的代码。

代码清单 6-3 全局函数 getJSON() 实现异步获取数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>getJSON 函数获取数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数处理获取的数据
                $.getJSON("UserInfo.json", function(data) {
                    $("#divTip").empty(); // 先清空标记中的内容
                    var strHTML = ""; // 初始化保存内容变量
                    // 遍历获取的数据
                    $.each(data, function(InfoIndex, Info) {
```

```
                    strHTML += "姓名：" + Info["name"] + "<br>";
                    strHTML += "性别：" + Info["sex"] + "<br>";
                    strHTML += "邮箱：" + Info["email"] + "<hr>";
                })
                $("#divTip").html(strHTML); // 显示处理后的数据
            })
        })
    })
    </script>
</head>
<body>
    <div class="divFrame">
        ...省略主体部分代码
    </div>
</body>
</html>
```

另外，创建一个文本文件，另存为 .json 格式，在文件 UserInfo.json 中，加入如下代码：

```
[
  {
    "name": "陶国荣",
    "sex": "男",
    "email": "tao_guo_rong@163.com"
  },
  {
    "name": "李建洲",
    "sex": "女",
    "email": "xiaoli@163.com"
  }
]
```

（3）页面效果

代码执行后的效果如图 6-3 所示。

图 6-3　全局函数 getJSON() 实现异步获取数据

说明 在示例 6-3 中，在单击按钮“获取数据”时，使用全局函数 $.each() 遍历所获取的数据 data，在遍历数据前，先清空 ID 号为“divTip”元素中的内容，以确保重新构建 HTML 内容，然后通过当前项 [‘元素名称’] 的方式获取每一项的数据，最后将叠加后的数据显示在 ID 号为“divTip”的元素中。

6.1.4 jQuery 中的全局函数 getScript()

在 jQuery 中，除通过全局函数 getJSON 获取 .json 格式的文件内容外，还可以通过另外一个全局函数 getScript() 获取 .js 文件的内容。其实，在页面中获取 .js 文件的内容有很多方法，如下列代码：

```
<script type="text/javascript"
src="Jscript/xx.js"></script>
```

动态设置的代码如下：

```
$("<script type='text/javascript' src='Jscript/xx.js' />")
.appendTo("head");
```

但这样的调用方法并不是最理想的。在 jQuery 中，通过全局函数 getScript() 加载 .js 文件后，不仅可以像加载页面片段一样轻松地注入脚本，而且所注入的脚本自动执行，大大提高了页面的执行效率。函数 getScript() 的调用格式如下所示：

```
$.getScript(url, [callback])
```

参数 url 为等待加载的 JS 文件地址，可选项 [callback] 参数表示加载成功时执行的回调函数。

下面通过示例来解释这种方法。

示例 6-4 全局函数 getScript() 实现异步获取数据

（1）功能描述

创建一个 UserInfo.js 文件，在该文件中，以数组的方式保存人员资料信息，然后，通过 $.each() 方法遍历各元素，将每次遍历元素的内容，以叠加的方式保存至变量 strHTML，并将该变量的值作为 ID 号为“divTip”元素的内容显示在页面中。另外，新建一个页面，在该页中，单击“获取数据”按钮后，打开 UserInfo.js 文件。

（2）实现代码

新建一个 HTML 文件 6-4.html, 加入如代码清单 6-4 所示的代码。

代码清单 6-4 全局函数 getScript() 实现异步获取数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>getScript 函数获取数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开已获取返回数据的文件
                $.getScript("UserInfo.js");
            })
        })
    </script>
</head>
<body>
   ... 省略主体部分代码
</body>
</html>
```

另外，新建一个 JS 文件 UserInfo.js，该文件的代码内容如下所示：

```
var data = [
  {
      "name": "陶国荣",
      "sex": "男",
      "email": "tao_guo_rong@163.com"
  },
  {
      "name": "李建洲",
      "sex": "女",
      "email": "xiaoli@163.com"
  }
];

var strHTML = ""; // 初始化保存内容变量
$.each(data, function() { // 遍历获取的数据
    strHTML += "姓名：" + this["name"] + "<br>";
    strHTML += "性别：" + this["sex"] + "<br>";
    strHTML += "邮箱：" + this["email"] + "<hr>";
})
$("#divTip").html(strHTML); // 显示处理后的数据
```

(3) 页面效果

代码执行后的效果如图 6-4 所示。

图 6-4　全局函数 getScript() 实现异步获取数据

说明　与所有全局函数一样，getScript() 也有一个回调函数，该回调函数在文件加载成功后执行，如代码“$.getScript('UserInfo.js', function() {alert(" 加载成功！ ");});”表示在文件加载成功后，将弹出一个内容是“加载成功！”的窗口。

6.1.5　jQuery 中异步加载 XML 文档

在前几节中，我们通过 jQuery 中的各种全局函数，实现了不同格式数据的异步加载，如 html、json、js 格式数据。在日常的页面开发中，有时也会遇到使用 XML 文档保存数据的情况，对于这种格式的数据，jQuery 中使用全局函数 $.get() 进行访问，其调用的语法格式为：

```
$.get(url, [data], [callback], [type])
```

其中参数 url 表示等待加载的数据地址，可选项 [data] 参数表示发送到服务器的数据，其格式为 key/value，可选项 [callback] 参数表示加载成功时执行的回调函数，可选项 [type] 参数表示返回数据的格式，如 html、xml、js、json、text 等。下面举例说明。

示例 6-5　全局函数 get() 实现异步获取 XML 文档数据

（1）功能描述

将人员资料信息保存到一个 XML 格式的文档 UserInfo.xml 中，另外，新建一个页面，在页面中，单击“获取数据”按钮，通过全局函数 $.get() 打开 XML 格式的文档 UserInfo.xml，并获取文档中的全部数据，显示在页面中。

（2）实现代码

新建一个 HTML 文件 6-5.html, 加入如代码清单 6-5 所示的代码。

代码清单 6-5　全局函数 get 实现异步获取 XML 文档数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
```

```
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>get 实现异步获取 xml 文档数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数处理获取的数据
                $.get("UserInfo.xml", function(data) {
                    $("#divTip").empty(); // 先清空标记中的内容
                    var strHTML = ""; // 初始化保存内容变量
                    $(data).find("User")
                        .each(function() { // 遍历获取的数据
                        var $strUser = $(this);
                        strHTML += " 姓名：" +
                        $strUser.find("name").text() + "<br>";
                        strHTML += " 性别：" +
                        $strUser.find("sex").text() + "<br>";
                        strHTML += " 邮箱：" +
                        $strUser.find("email").text() + "<hr>";
                        })
                        $("#divTip").html(strHTML); // 显示处理后的数据
                    })
                })
            })
    </script>
</head>
<body>
   ... 省略主体部分代码
</body>
</html>
```

另外，新建一个 XML 文件 UserInfo.xml，该文件的代码内容如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<Info>
  <User id="1">
    <name> 陶国荣 </name>
    <sex> 男 </sex>
    <email>tao_guo_rong@163.com</email>
  </User>

  <User id="2">
```

```
    <name> 李建洲 </name>
    <sex> 女 </sex>
    <email>xiaoli@163.com</email>
  </User>
</Info>
```

（3）页面效果

代码执行后的效果如图 6-5 所示。

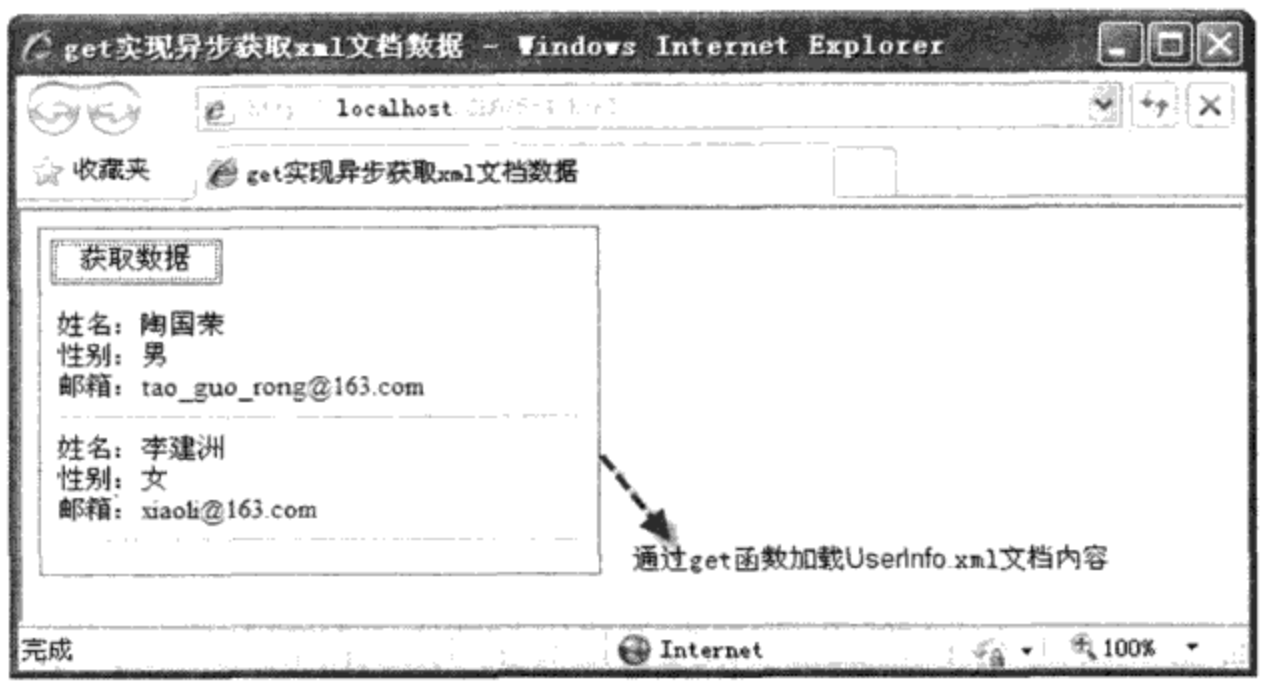


图 6-5　全局函数 get 实现异步获取 XML 文档数据

说明　在示例的 JS 代码中，先通过 each() 方法遍历文档中的 User 节点，然后在遍历的过程中使用 find 方法，查询该节点中的 name、sex、email 选项，并通过 text() 方法获取各选项的值，最后，将各值以叠加的形式保存到变量 strHTML，并显示在页面中。

6.2　请求服务器数据

前面 6.1 节介绍的是如何在 HTML 页面中加载异步数据的方法，即如何从服务器上取得静态的数据。但页面的应用远不仅局限于此，Ajax 技术最终体现在与服务器的动态数据实现人机交互中，即客户端传递带有参数的请求，服务器接收后，分析所传递来的请求，并做出相应的响应，发送对应数据至客户端，客户端接收请求返回的数据，从而实现了数据的双向传递。下面就介绍交互式函数的应用。

6.2.1　$.get() 请求数据

在上面的 6.1.5 小节中，通过调用全局函数 $.get()，实现了 XML 文档的加载。除加载数据外，$.get() 函数还可以实现数据的请求。下面通过一个示例介绍 $.get() 函数带参请求服务器中的数据。

示例 6-6　全局函数 get () 向服务器请求数据

（1）功能描述

创建一个服务器页面 UserInfo.aspx，该页面的功能是获取客户端发送的请求，并分析传来的参数值，返回对应的结果。另外，在客户端 HMTL 页面的文本框中，输入传递的参数值，单击“请求数据”按钮后，显示服务器返回的数据结果。

（2）实现代码

新建一个 HTML 文件 6-6.html, 加入如代码清单 6-6 所示的代码。

代码清单 6-6　全局函数 get() 向服务器请求数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.get 发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数返回服务器响应后的数据
                $.get("UserInfo.aspx",
                { name: encodeURI($("#txtName").val()) },
                function(data) {
                    $("#divTip")
                    .empty() // 先清空标记中的内容
                    .html(data); // 显示服务器返回的数据
                })
            })
        })
    </script>
</head>
<body>
    ... 省略主体部分代码
</body>
</html>
```

另外，新建一个服务器端文件 UserInfo.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"
         ResponseEncoding="gb2312" %>
<%
```

```
    string strName = System.Web.HttpUtility
                     .UrlDecode(Request["name"]);// 解码姓名字符
    string strHTML = "<div class='clsShow'>"; // 初始化保存内容变量
    if (strName == "陶国荣")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        strHTML += "邮箱：tao_guo_rong@163.com<hr>";
    }
    else if (strName == "李建洲")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        strHTML += "邮箱：xiaoli@163.com<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

代码执行后的效果如图 6-6 所示。

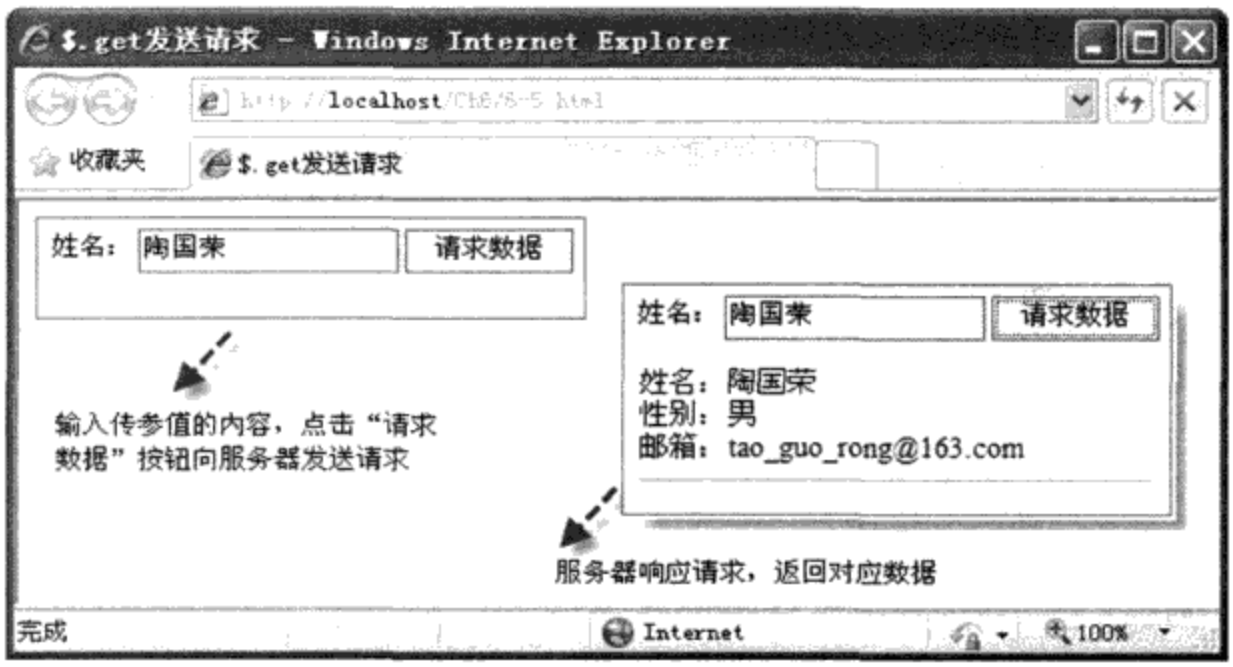


图 6-6　全局函数 get () 向服务器请求数据

说明　客户端向服务器传递参数时，使用的格式是 {key0:value0,key1:value1,...}，“key0”为参数名称，“value0”为参数的值。如果参数的值是中文格式，必须通过使用 encodeURI() 进行转码，当然，在客户端接收时，使用 decodeURI() 进行解码即可。

6.2.2　$.post() 请求数据

$.post() 也是带参数向服务器发出数据请求。全局函数 $.post() 与 $.get() 在本质上没有太大的区别，所不同的是，GET 方式不适合传递数据量较大的数据，同时，其请求的历史信息

会保存在浏览器的缓存中，有一定的安全风险。而以 POST 方式向服务器发送数据请求，则不存在这两个方面的不足。

$.post() 函数调用的语法格式如下所示：

```
$.post(url, [data], [callback], [type])
```

其中参数与 $.get() 函数参数代表的意义完全相同，在此不再赘述。下面举例说明。

示例 6-7　全局函数 post () 向服务器请求数据

（1）*功能描述*

在示例 6-6 的基础上，在客户端页面中增加一个下拉列表框，用于“性别”的选择，单击“请求”按钮后，传递两个参数 name 与 sex 的值，向服务器页面 User_Info.aspx 发出数据请求，服务器页面接收参数值后，响应请求，将相应数据发送到客户端。

（2）*实现代码*

新建一个 HTML 文件 6-7.html, 加入如代码清单 6-7 所示的代码。

代码清单 6-7　全局函数 post() 向服务器请求数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.post 发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { //按钮单击事件
                //打开文件，并通过回调函数返回服务器响应后的数据
                $.post("User_Info.aspx",
                { name: encodeURI($("#txtName").val()),
                   sex: encodeURI($("#selSex").val())
                },
                function(data) {
                    $("#divTip")
                    .empty() //先清空标记中的内容
                    .html(data); //显示服务器返回的数据
                })
            })
        })
    </script>
</head>
```

```
<body>
    ...省略主体部分代码
</body>
</html>
```

另外，新建一个服务器端文件 User_Info.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"
 ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility
       .UrlDecode(Request["name"]);// 解码姓名字符
    string strSex = System.Web.HttpUtility
       .UrlDecode(Request["sex"]);// 解码性别字符
    string strHTML = "<div class='clsShow'>"; // 初始化保存内容变量
    if (strName == "陶国荣" && strSex=="男")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        strHTML += "邮箱：tao_guo_rong@163.com<hr>";
    }
    else if (strName == "李建洲" && strSex == "女")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        strHTML += "邮箱：xiaoli@163.com<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

执行后的效果如图 6-7 所示。

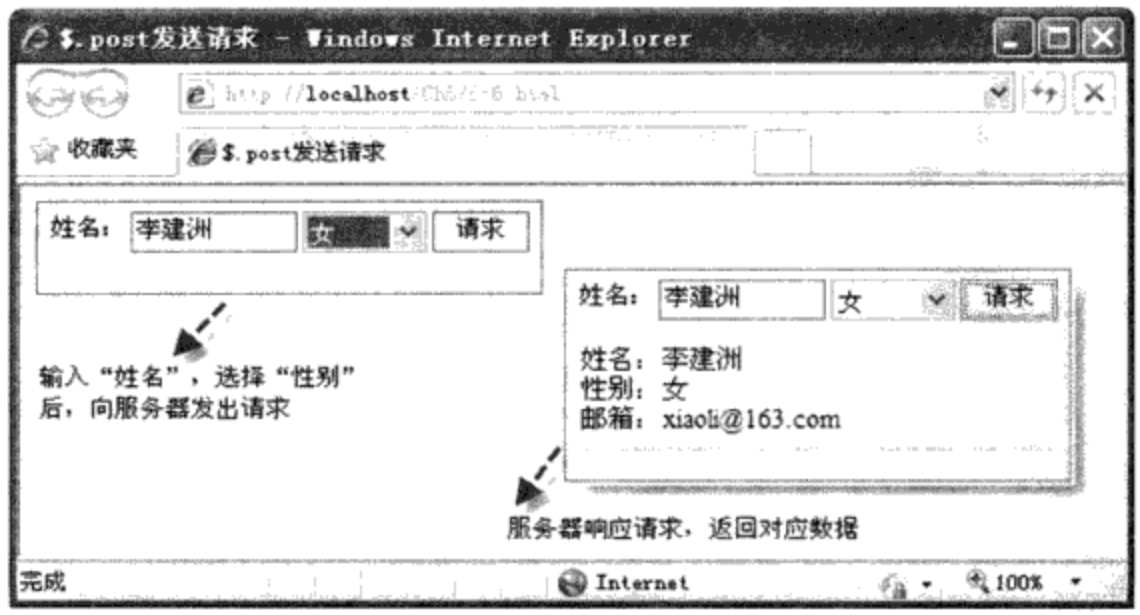


图 6-7　全局函数 post 向服务器请求数据

6.2.3　serialize() 序列化表单

在使用全局函数 $.get() 和 $.post() 向服务器传递参数时，其中的参数是通过名称属性逐个搜索输入字段的方式进行传输的，如果表单的输入字段过多，这种方式就比较麻烦，而且不利于拓展。为了解决这个问题，jQuery 引入 serialize() 方法，该方法可以简化参数传值的方式，其调用的语法格式如下：

```
serialize()
```

该方法的功能是将所选择的 DOM 元素转换成能随 AJAX 传递的字符串，即序列化所选择的 DOM 元素。下面举例说明。

示例 6-8　serialize() 序列化表单

（1）功能描述

在示例 6-7 的基础上，在客户端页面中，再增加一个“邮箱”复选框，用于控制页面是否显示邮箱地址，当单击“请求”按钮时，通过调用表单的 serialize() 方法，获取全部的输入字段值，并向服务器页面 User-Info.aspx 发出数据请求，服务器页面接收参数值后，响应请求，将相应数据发送到客户端。

（2）实现代码

新建一个 HTML 文件 6-8.html, 加入如代码清单 6-8 所示的代码。

代码清单 6-8　serialize() 序列化表单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>serialize() 序列化表单 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() { // 按钮单击事件
                // 打开文件，并通过回调函数返回服务器响应后的数据
                $.post("User-Info.aspx",
                $("#frmUserInfo").serialize(),// 序列化表单数据
                function(data) {
                    $("#divTip")
                    .empty() // 先清空标记中的内容
                    .html(data); // 显示服务器返回的数据
                })
```

```
            })
        })
    </script>
</head>
<body>
  <form id="frmUserInfo">
     ...省略主体部分代码
  </form>
</body>
</html>
```

另外，新建一个服务器端文件 User-Info.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility
      .UrlDecode(Request["txtName"]); //解码姓名字符
    string strSex = System.Web.HttpUtility
      .UrlDecode(Request["selSex"]); //解码性别字符
    string strEmail = Request["chkEmail"]; //是否显示邮件字符
    string strHTML = "<div class='clsShow'>"; //初始化保存内容变量
    if (strName == "陶国荣" && strSex=="男")
    {
        strHTML += "姓名：陶国荣<br>";
        strHTML += "性别：男<br>";
        if (strEmail=="1")
        {
            strHTML += "邮箱：tao_guo_rong@163.com";
        }
        strHTML += "<hr>";
    }
    else if (strName == "李建洲" && strSex == "女")
    {
        strHTML += "姓名：李建洲<br>";
        strHTML += "性别：女<br>";
        if (strEmail == "1")
        {
            strHTML += "邮箱：xiaoli@163.com<hr>";
        }
        strHTML += "<hr>";
    }
    strHTML += "</div>";
    Response.Write(strHTML);
%>
```

（3）页面效果

代码执行后的效果如图 6-8 所示。

图 6-8　serialize() 序列化表单

说明　虽然 serialize() 方法可以很完美地模拟浏览器提交表单的操作，同时自动解决了中文编码的问题，但它自身有很多不足，如表单中有多项被选中时，该方法只能传递一项的值，因此，在选择传递参数时，须慎重考虑。

6.3　$.ajax() 方法

除了可以使用全局性函数 load()、get()、post() 实现页面的异步调用和与服务器交互数据外，在 jQuery 中，还有一个功能更为强悍的最底层的方法 $.ajax()，该方法不仅可以很方便地实现上述三个全局函数完成的功能，还更多地关注实现过程中的细节。下面我们详细介绍该方法。

6.3.1　$.ajax() 的基本概念

在 jQuery 中，$.ajax() 是最底层的方法，也是功能最强的方法，前几节中的 $.get()、$.post()、$.getScript()、$.getJSON() 都是在此方法的基础之上建立的。其调用的语法格式为：

```
$.ajax([options])
```

其中可选项参数 [options] 为 $.ajax() 方法中的请求设置，其格式为 key/value，既包含发送请求的参数，也含有服务器响应后回调的数据，其全部名称如表 6-1 所示。

表 6-1　$.ajax() 方法中的参数列表

参数名	类型	描　述
url	String	发送请求的地址（默认为当前页面）
type	String	数据请求方式（post 或 get），默认为 get
data	String 或 Object	发送到服务器的数据。如果不是字符串则自动转成字符串格式，如果是 get 请求方式，那么，该字符串将附在 url 的后面

（续）

参数名	类型	描　述
dataType	String	服务器返回的数据类型，如果没有指定，jQuery 将自动根据 HTTP 包 MIME 信息自动判断，服务器返回的数据根据自动判断的结果进行解析，传递给回调函数，其可用类型为： html：返回纯文本的 HTML 信息，包含的 Script 标记会在插入页面时被执行 script：返回纯文本 JavaScript 代码 text：返回纯文本字符串 xml：返回可被 jQuery 处理的 XML 文档 json：返回 JSON 格式的数据
beforeSend	Function	该函数用于发送请求前修改 XMLHttpRequest 对象，其中的参数就是 XMLHttpRequest 对象，由于该函数本身是 jQuery 事件，因此，如果函数返回 false，则表示取消本次事件
complete	Function	请求完成后调用的回调函数，该函数无论数据发送成功或失败都会调用，其中有两个参数，一个是 XMLHttpRequest 对象，另外一个是 strStatus，用于描述成功请求类型的字符串
success	Function	请求成功后调用的回调函数，该函数有两个参数，一个是根据参数 dataType 处理后服务器返回的数据，另外一个是 strStatus，用于描述状态的字符串
error	Function	请求失败后调用的回调函数，该函数有三个参数，第一个是 XMLHttpRequest 对象，第二个是出错信息 strError，第三个是捕捉到的错误对象 strObject
timeout	Number	请求超时的时间（毫秒），该设置将覆盖 $.ajaxSetup() 方法中的同样设置
global	Boolean	是否响应全局事件，默认是 true，表示响应，如果设置成 false，表示不响应，那么，全局事件 $.ajaxStart 等将不响应
async	Boolean	是否为异步请求，默认是 true，表示是异步，如果设置成 false，表示是同步请求
cache	Boolean	是否进行页面缓存，true 表示进行缓存，false 表示不进行页面缓存

下面通过一个简单示例介绍 jQuery 中 $.ajax() 方法在数据交互过程中的应用。

示例 6-9　用 $.ajax() 方法发送请求

（1）功能描述

创建一个用于登录的 HTML 页 login.html，在页面中设置用于输入“用户名”和“密码”的文本框及“登录”和“取消”按钮，该页面不做任何数据处理。

另外，创建一个服务器端页面 login.aspx，用于处理静态页发送来的登录请求；同时，再创建一个 HTML 页 6-9.html。

在页面 6-9.html 中，先使用 $.ajax() 方法请求调用 login.html 页面，然后，通过页面中“登录”按钮的单击事件，将获取的用户名称和密码发送到服务器，服务器响应请求，向客 户端发送处理后的数据，客户端根据回调的数据，显示不同的页面效果。

（2）实现代码

新建一个 HTML 文件 6-9.html, 加入如代码清单 6-9 所示的代码。

代码清单 6-9　$.ajax() 方法发送请求

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.ajax() 方法发送请求 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $.ajax({ // 请求登录页
                url: "login.html", // 登录静态页
                dataType: "html",
                success: function(HTML) { // 返回页面内容
                    // 将页面内容置入表单
                    $("#frmUserLogin").html(HTML);
                    //" 登录 " 按钮单击事件
                    $("#btnLogin").click(function() {
                        // 获取用户名称
                        var strTxtName = encodeURI($("#txtName")
                                         .val());
                        // 获取输入密码
                        var strTxtPass = encodeURI($("#txtPass")
                                         .val());
                        // 开始发送数据
                        $.ajax({ // 请求登录处理页
                            url: "login.aspx", // 登录处理页
                            dataType: "html",
                            // 传送请求数据
                            data: { txtName: strTxtName,
                                    txtPass: strTxtPass },
                            // 登录成功后返回的数据
                            success: function(strValue) {
                                // 根据返回值进行状态显示
                                if (strValue == "True") {
                                    $(".clsShow")
                                    .html(" 操作提示，登录成功！ ");
                                }
                                else {
                                    $("#divError")
                                    .show().html(" 用户名或密码错误！ ");
                                }
                            }
                        })
                    })
                }
            })
        })
```

```
    </script>
</head>
<body>
  <form id="frmUserLogin"></form>
</body>
</html>
```

另外，再新建一个 HTML 页面 Login.html，该文件的代码内容如下所示：

```
<div class="divFrame">
     <div class="divTitle">
          <span>用户登录</span>
     </div>
     <div class="divContent">
          <div class="clsShow">
               <div id="divError" class="clsError"></div>
               <div>名称：<input id="txtName" type="text"
                                class="txt" /></div>
               <div>密码：<input id="txtPass" type="password"
                                class="txt" /></div>
               <div>
                   <input id="btnLogin" type="button"
                          value="登录" class="btn" />  
                   <input id="btnReset" type="reset"
                          value="取消" class="btn" />
               </div>
          </div>
     </div>
</div>
```

再有，新建一个服务器端文件 Login.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
<%
    string strName = System.Web.HttpUtility
      .UrlDecode(Request["txtName"]); //解码姓名字符
    string strPass = System.Web.HttpUtility
      .UrlDecode(Request["txtPass"]); //解码密码字符
    bool blnLogin = false;
    if (strName == "admin" && strPass == "123456")
    {
        blnLogin = true;
    }
    Response.Write(blnLogin);
%>
```

（3）页面效果

代码执行后的效果如图 6-9 所示。

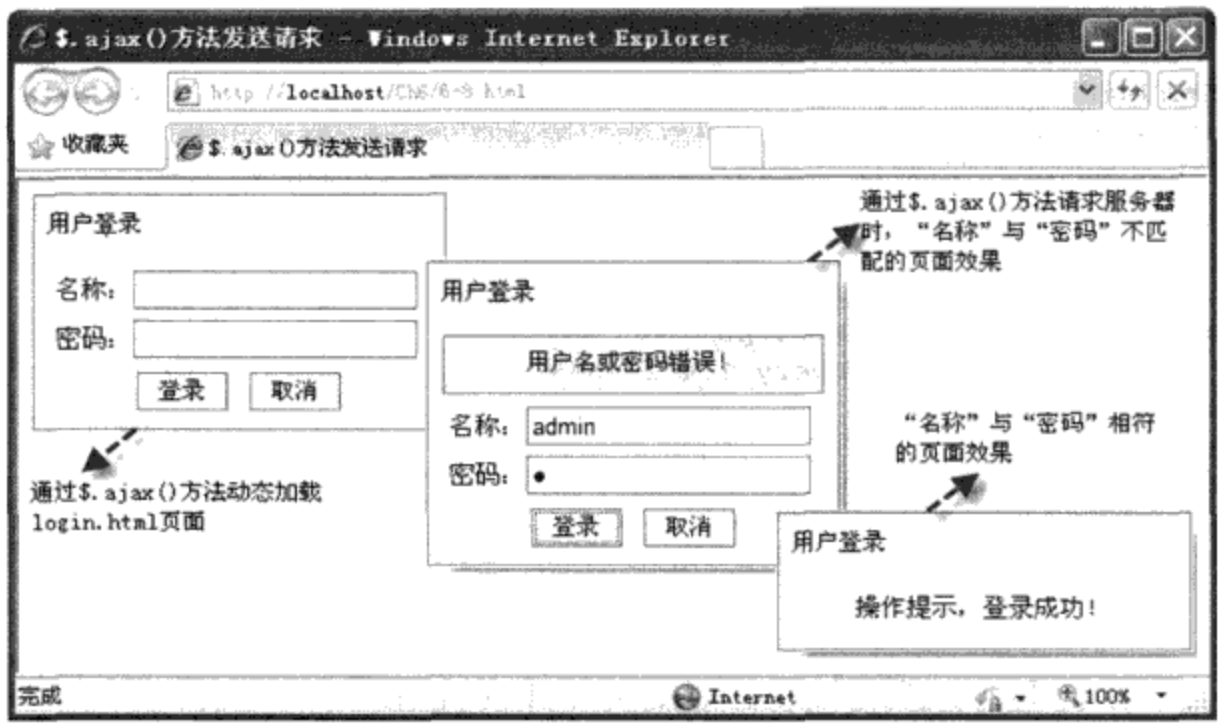


图 6-9　$.ajax() 方法发送请求

说明　由于 $.ajax() 方法是 jQuery 中最底层的方法，因此，凡使用全局函数 $.getScript()、$.get()、$.post()、$.getJSON() 调用的示例，都可以使用 $.ajax() 方法进行代替，限于篇幅，这里不再赘述，读者可自行测试。

6.3.2　$.ajaxSetup() 设置全局 Ajax

在使用 $.ajax() 方法时，有时需要调用多个 $.ajax() 方法，如果每个方法都设置其中的请求细节，将是一件十分麻烦的事。为了简化这种工作，在 jQuery 中，可以使用 $.ajaxSetup() 方法设置全局性的 Ajax 默认选项，一次设置，全局有效，这样大大简化了 $.ajax() 方法中细节的编写，该方法的调用格式为：

```
$.ajaxSetup([options])
```

其中的可选项参数 [options] 是一个对象，通过该对象可以设置 $.ajax() 方法中的参数。下面举例说明。

示例 6-10　$.ajaxSetup() 方法全局设置 Ajax

（1）功能描述

在页面中，设置三个按钮，分别通过 $.ajax() 方法请求一个 XML 文档中的某部分数据，并将回调的数据展示在页面中，在请求前，使用 $.ajaxSetup() 方法进行一些参数项的全局性设置。

（2）实现代码

新建一个 HTML 文件 6-10.html, 加入如代码清单 6-10 所示的代码。

代码清单 6-10 $.ajaxSetup() 方法全局设置 Ajax

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>$.ajaxSetup() 方法全局设置 Ajax</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $.ajaxSetup({ // 设置全局性的 Ajax 选项
                type: "GET",
                url: "UserInfo.xml",
                dataType: "xml"
            })
            $("#Button1").click(function() { //" 姓名 " 按钮的单击事件
                $.ajax({
                    success: function(data) { // 传回请求响应的数据
                     ShowData(data, " 姓名 ", "name"); // 显示 " 姓名 " 部分
                    }
                })
            })
            $("#Button2").click(function() { //" 性别 " 按钮的单击事件
                $.ajax({
                    success: function(data) { // 传回请求响应的数据
                     ShowData(data, " 性别 ", "sex"); // 显示 " 性别 " 部分
                    }
                })
            })
            $("#Button3").click(function() { //" 邮箱 " 按钮的单击事件
                $.ajax({
                    success: function(data) { // 传回请求响应的数据
                     ShowData(data, " 邮箱 ", "email"); // 显示 " 邮箱 " 部分
                    }
                })
            })
            /*
            * 根据名称与值，获取请求响应数据中的某部分
            *@Param d 为请求响应后的数据
            *@Param n 为数据中文说明字符
            *@Param d 为数据在响应数据中的元素名称
            */
            function ShowData(d, n, v) {
                $("#divTip").empty(); // 先清空标记中的内容
                var strHTML = ""; // 初始化保存内容变量
                $(d).find("User").each(function() { // 遍历获取的数据
                    var $strUser = $(this);
```

```
                    strHTML += n + ":" +
                    $strUser.find(v).text() +
                    "<hr>";
                })
                $("#divTip").html(strHTML); // 显示处理后的数据
            }
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <span><input id="Button1" type="button"
                         value="姓名" class="btn" /></span>
            <span><input id="Button2" type="button"
                         value="性别" class="btn" /></span>
            <span><input id="Button3" type="button"
                         value="邮箱" class="btn" /></span>
        </div>
        <div class="divContent">
            <div id="divTip" class="clsShow"></div>
        </div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 6-10 所示。

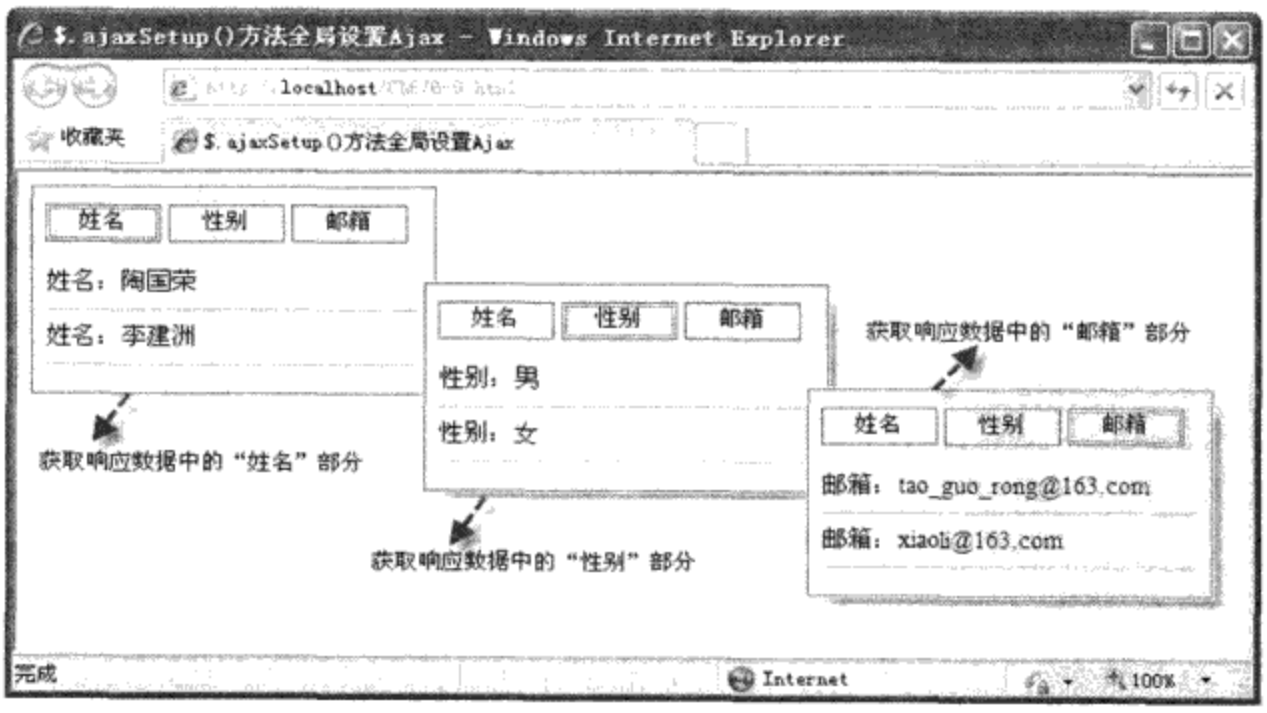


图 6-10　$.ajaxSetup() 方法全局设置 Ajax

说明　在 JS 代码中，由于使用了 $.ajaxSetup() 方法设置部分全局性的 Ajax 参数选项，使后续代码中的异步数据请求非常简单，避免了重复编写相同的代码。由于每次请求都要分析

响应后的数据，因此通过一个自定义的函数 ShowData，在每次调用时，根据不同的数据元素名称，返回对应的数据。

6.4 Ajax 中的全局事件

在 jQuery 中，除了全局性的函数外，还有 6 个全局性的 Ajax 事件。由于在使用 $.ajax() 方法时，其中的选项参数 global 的值默认情况下为 true，这也就意味着，所有在执行的 Ajax 的数据请求，都绑定了全局事件。

6.4.1 Ajax 全局事件的基本概念

jQuery 中的 6 个全局性 Ajax 事件的详细说明见表 6-2。

表 6-2 Ajax 中的全局事件

事件名称	参数	功能描述
ajaxComplete(callback)	callback	Ajax 请求完成时执行函数
ajaxError(callback)	callback	Ajax 请求发生错误时执行函数，其中捕捉到的错误可以作为最后一个参数进行传递
ajaxSend(callback)	callback	Ajax 请求发送前执行函数
ajaxStart(callback)	callback	Ajax 请求开始时执行函数
ajaxStop(callback)	callback	Ajax 请求结束时执行函数
ajaxSuccess(callback)	callback	Ajax 请求成功时执行函数

说明 在 jQuery 中，所有的全局事件都是以 XMLHttpRequest 对象和设置作为参数传递给回调函数，在处理回调函数时，只要分析其传回的参数值即可。

6.4.2 ajaxStart 与 ajaxStop 全局事件

在 jQuery 中，使用 Ajax 获取异步数据时，ajaxStart 与 ajaxStop 这两个全局事件的使用频率非常高。前者是当请求开始执行时触发，往往用于编写一些准备性的工作，如提示“正在获取数据……”字样；后者是当请求结束时触发，在这一事件中，常常与前者配合，说明请求的最后进展状态，如将显示中的“正在获取数据……”字样修改为“已成功获取数据！”，后渐渐消失掉。

在事件绑定时，Ajax 中的全局事件可以绑定在页面的任何一个元素中，如下列代码将全局事件 ajaxStart 绑定在一个 标记中：

```
$("divTip").ajaxStart(function(){
  $(this).html("正在处理中 ...");
})
```

通过编写上述代码，使页面中的任何 Ajax 请求在开始执行时都会触发，即显示“正在处理中……”的字样。下面举例说明。

示例 6-11　jQuery 中的全局事件

（1）功能描述

创建一个 HTML 页面，设置一个文本框和一个按钮，单击“请求”按钮后，获取文本框数据，向服务器发送请求，服务器响应请求后，返回对应的数据，在数据交互过程中，请求开始时，触发绑定的 ajaxStart 全局事件，出现“正在发送数据请求……”的字样；请求结束后，触发绑定的 ajaxStop 全局事件，将“正在发送数据请求……”的字样改成“已成功获取数据。”，并隐藏起来。

（2）实现代码

新建一个 HTML 文件 6-11.html, 加入如代码清单 6-11 所示的代码。

代码清单 6-11　jQuery 中的全局事件

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Ajax 中的全局事件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
<script type="text/javascript">
    $(function() {
        // 元素绑定全局 ajaxStart 事件
        $("#divMsg").ajaxStart(function() {
            $(this).show(); // 显示元素
        })
        // 元素绑定全局 ajaxStop 事件
        $("#divMsg").ajaxStop(function() {
            $(this).html(" 已成功获取数据。").hide();
        })
        // 按钮的单击事件
        $("#Button1").click(function() {
            $.ajax({ // 带参数请求服务器
                type: "GET",
                url: "GetData.aspx",
                // 获取加码后的数据并作为参数传给服务器
                data: { txtData:
                        encodeURI($("#txtData").val()) },
                dataType: "html",
                success: function(data) { // 显示解码后的返回数据
                    $("#divTip").html(decodeURI(data));
                }
            })
```

```
            })
        })
        </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            <span>数据：<input id="txtData" type="text"
                              class="txt" /></span>
            <span><input id="Button1" type="button"
                              value="发送 " class="btn" /></span>
        </div>
        <div class="divContent">
            <div id="divMsg" class="clsTip">正在发送数据请求 ...</div>
            <div id="divTip" class="clsShow"></div>
        </div>
    </div>
</body>
</html>
```

另外，新建一个服务器端文件 GetData.aspx，该文件的代码内容如下所示：

```
<%@ Page Language="C#" ContentType="text/html"ResponseEncoding="gb2312" %>
<%
    string strName = Request["txtData"]; //返回传回的参数
    Response.Write(strName); //返回传回的参数
%>
```

（3）页面效果

代码执行后的效果如图 6-11 所示。

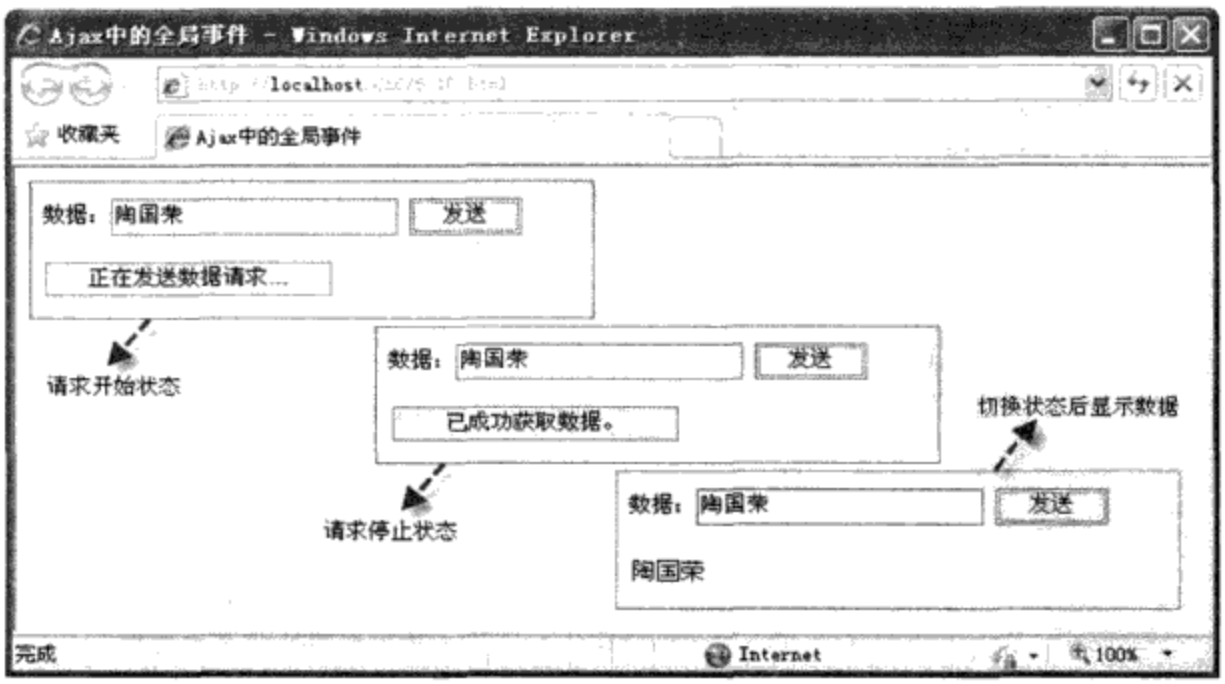


图 6-11　jQuery 中的全局事件

说明　由于是全局性的事件，因此，只要有 Ajax 请求发生，就会触发所设置的全局事件，该事件被绑定在某个元素上，通过一些很人性化的显示或隐藏进行切换，极大地丰富了提交数据时的用户体验，同时，也建立了页面加载反馈系统。

6.5　综合案例分析——用 Ajax 实现新闻点评即时更新

6.5.1　需求分析

经分析，该案例的需求如下：

1）使用 XML 文档保存新闻点评数据，在初始化页面和发表点评内容时，使用无刷新的方式调用该数据。

2）使用无刷新的方式，获取页面中提交的数据，通过服务器文件，写入保存新闻点评数据的 XML 文档中。

3）无论是获取数据，还是请求服务器响应，在页面中都有人性化的提示信息侦察进程的状态，即操作时显示，操作完成后则隐藏。

6.5.2　效果界面

该案例的效果界面如下：

1）初次加载点评数据时，在获取数据前，出现提示进程的状态信息，其页面效果如图 6-12 所示。

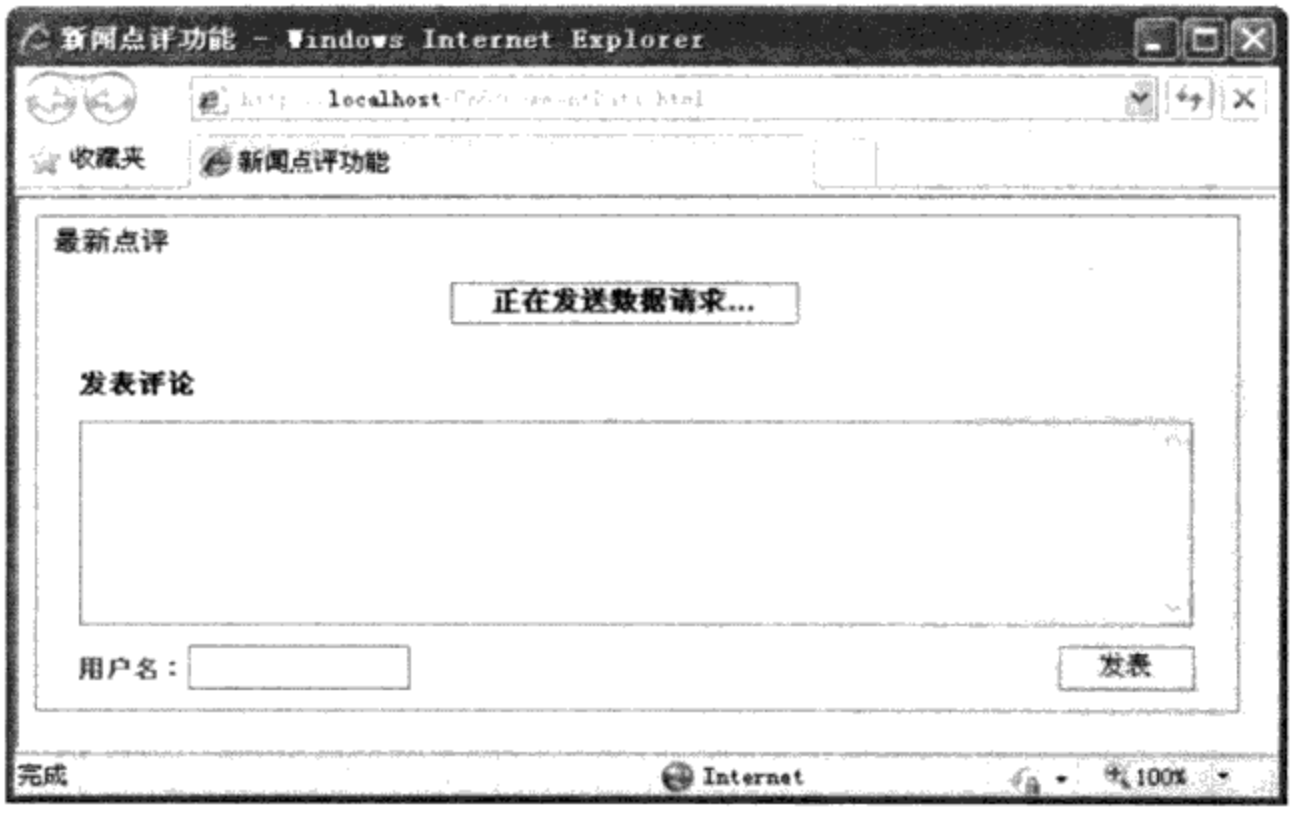


图 6-12　初始化新闻点评数据

2）输入点评内容与用户名称后，单击“发表”按钮，同样出现提示状态信息，并根据请求的进度切换状态信息显示的内容和效果，其页面效果如图 6-13 所示。

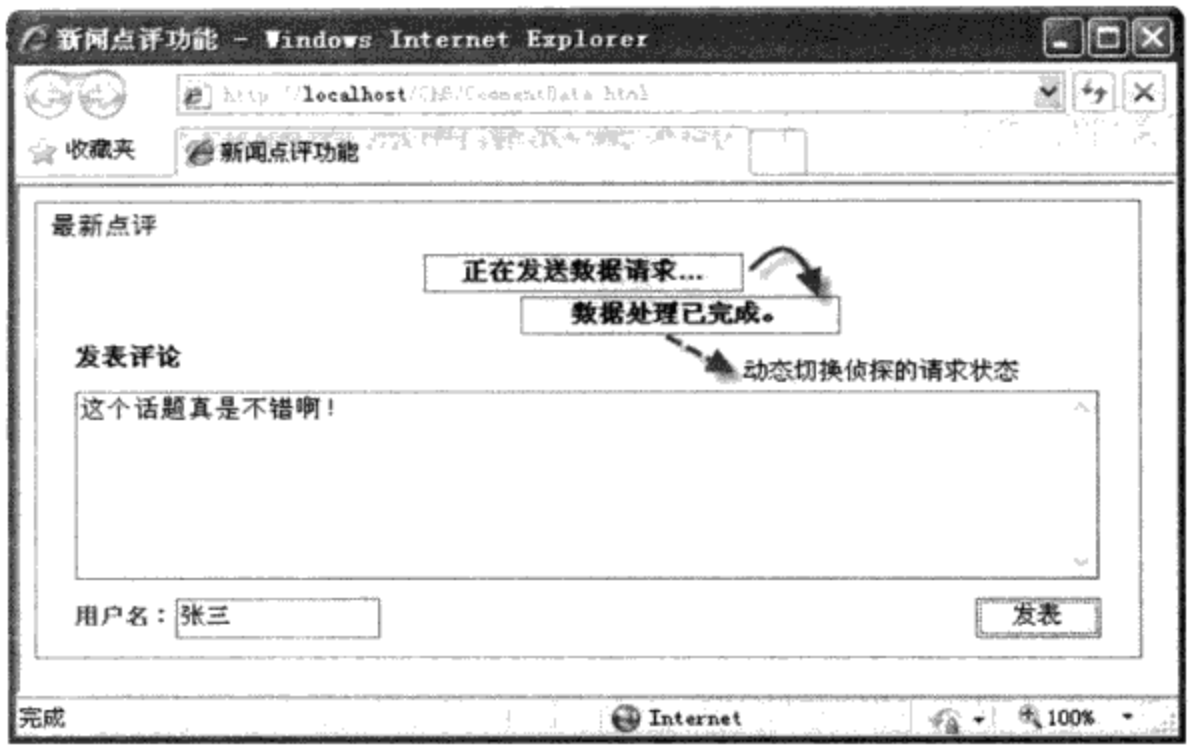


图 6-13　展示获取的点评数据

3）当服务器响应请求后，将传来的内容与用户名参数解码后写入 XML 中，同时，返回请求成功的信息，客户端根据这一信息，重新加载点评数据，将新增加的数据显示在页面中，其页面效果如图 6-14 所示。

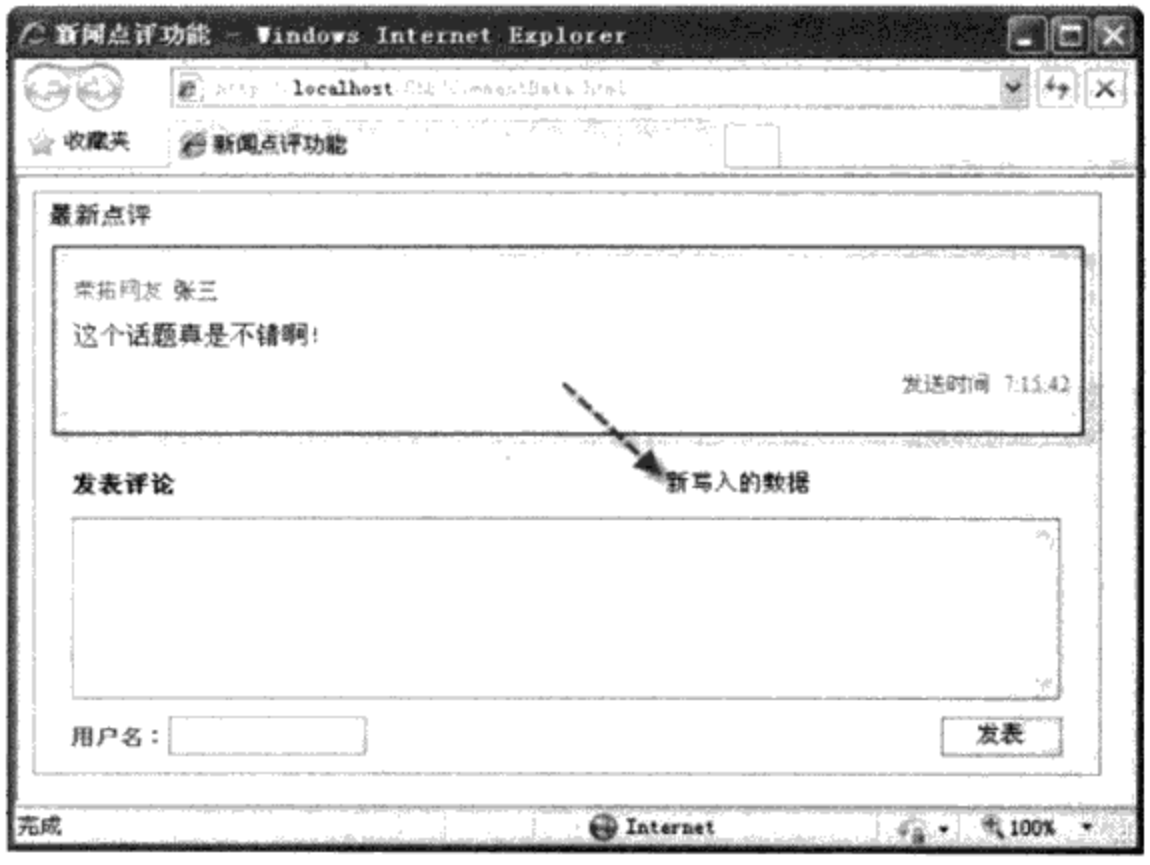


图 6-14　展示最新写入的数据

6.5.3　功能实现

新建一个 HTML 文件 CommentData.html，加入如代码清单 6-12 所示的代码。

代码清单 6-12　实现新闻点评即时更新

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>新闻点评功能</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <link type="text/css" href="Css/css_Ajax.css"
                          rel="Stylesheet" />
    <script type="text/javascript"
                          src="Jscript/js_Ajax.js"></script>
</head>
<body>
     <div class="divFrame">
         <div class="divTitle">
              <span>最新点评</span>
         </div>
         <div class="divContent">
         </div>
         <div class="divSubmit">
             <div class="SubTitle">发表评论
                  <span id="divMsg" class="clsTip">
                        正在发送数据请求...
                  </span>
             </div>
             <div class="SubContent">
                 <textarea id="txtContent" rows="6"
                           class="txt"></textarea>
                 <div class="SubBtn">
                     <span style="float:left">用户名：
                           <input id="txtName" type="text"
                                  class="txt" />
                     </span>
                     <span style="float:right">
                           <input id="Button1" type="button"
                                  value="发表" class="btn" />
                     </span>
                 </div>
             </div>
         </div>
    </div>
</body>
</html>
```

另外，为了便于模块化调用，针对该点评页面新建一个控制样式的 CSS 文件 css_Ajax.css，其代码如代码清单 6-13 所示。

代码清单 6-13　CSS 文件 css_Ajax.css

```
body{font-size:13px}
a
{
   text-decoration:none
}
/* 外框样式 */
.divFrame
{
   width:572px;
   border:solid 1px #666;
   background-color:#fafcff
}
/* 外框中主题样式 */
.divFrame .divTitle
{
   padding:5px;
   background-color:#eee
}
.divFrame .divTitle span
{
   padding:2px;
   padding-top:5px;
   font-family: 黑体 ;
   font-size:14px
}
/* 外框中内容样式 */
.divFrame .divContent
{
   padding:8px
}
... 省略样式部分代码
.divFrame .divContent .clsShow .ShowBottom
{
   text-align:right;
   font-size:12px;
   color:#555
}
/* 外框中提交点评内容样式 */
.divFrame .divSubmit
{
   padding:20px
}
... 省略样式部分代码
.divFrame .divSubmit .SubContent .SubBtn
{
```

```
    padding-top:10px;
    padding-bottom:12px;
    font-size:12px;
    color:#555;
    font-weight:bold
}
/* 侦察请求状态样式 */
.clsTip
{
    position:absolute;
     width:160px;
     text-align:center;
     font-size:13px;
     border:solid 1px #cc3300;
     padding:2px;
     margin-bottom:5px;
     background-color:#ffe0a3
 }
 ... 省略样式部分代码
```

用于实现页面功能的 JS 文件 js_Ajax.js，其代码如代码清单 6-14 所示。

代码清单 6-14　JS 文件 js_Ajax.js

```
/// <reference path="jquery-1.4.2.js"/>
$(function() {
// 元素绑定全局 ajaxStart 事件
$("#divMsg").ajaxStart(function() {
    $(this).show(); // 显示元素
})
// 元素绑定全局 ajaxStop 事件
$("#divMsg").ajaxStop(function() {
    $(this).html(" 数据处理已完成。").hide();
})
// 初始化点评数据
LoadData();
$("#Button1").click(function() { // 点击 " 发表 " 按钮事件
    // 获取加码后的用户名
    var strName = encodeURI($("#txtName").val());
    // 获取加码后的发表内容
    var strContent = encodeURI($("#txtContent").val());
    $.ajax(
       {
           type: "GET",
           url: "AddData.aspx", // 请求增加数据动态页
           dataType: "html",
           data: { name: strName, content: strContent },
           success: function(msg) {
               alert(msg);
               LoadData();
```

```
                $("#txtName").val("");
                $("#txtContent").val("");
            }
        })
})
/*
* 动态加载 XML 格式的点评数据
*/
function LoadData() {
    $.ajax(
        {
            type: "GET",
            url: "CommentData.xml", // 请求 XML 格式数据
            dataType: "xml",
            cache: false,
            success: function(data) {
                $(".divContent").empty(); // 先清空标记中的内容
                var strHTML = ""; // 初始化保存内容变量
                // 如果没有找到数据
                if ($(data).find("Data").length == 0) {
                    strHTML = "<div style='text-align:center'>
                              目前还没有找到点评数据! </div>";
                }
                // 遍历获取的数据
                $(data).find("Data").each(function() {
                    var $strUser = $(this);
                    strHTML += "<div class='clsShow'>";
                    strHTML += "<div class='ShowTitle'>荣拓网友
                      <a href=''>" +
                    $strUser.find("name").text() + "</a></div>";
                    strHTML += "<div class='ShowContent'>" +
                    $strUser.find("content").text() + "</div>";
                    strHTML += "<div class='ShowBottom'>发送时间
                      " +
                    $strUser.find("date").text() + "</div>"
                    strHTML += "</div>"
                })
                $(".divContent").html(strHTML); // 显示处理后的数据
            }
        })
    }
})
```

用于响应客户端请求，并处理数据的服务器端页面 AddData.aspx，其实现的代码如代码清单 6-15 所示。

代码清单 6-15　服务器端文件 AddData.aspx

```
<%@ Page Language="C#" ContentType="text/html"
```

```
ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.IO" %>
<%
    string strName = System.Web.HttpUtility
.UrlDecode(Request["name"]); //解码点评用户名称
    string strContent = System.Web.HttpUtility
.UrlDecode(Request["content"]); //解码点评提交内容
    string strFileName = "CommentData.xml";
    //定义 XML 文档变量
    XmlDocument xmlDoc = new XmlDocument();
    //打开指定的 XML 文档
    xmlDoc.Load(Server.MapPath(strFileName));
    //查找根节点元素
    XmlNode xmlN = xmlDoc.SelectSingleNode("Comment");
    //加入一个节点元素
    XmlElement xmlE = xmlDoc.CreateElement("Data");

    //创建一个子节点
    XmlElement xmlEn = xmlDoc.CreateElement("name");
    //设置节点文本
    xmlEn.InnerText = strName;
    //添加到节点中
    xmlE.AppendChild(xmlEn);
    //创建一个子节点
    XmlElement xmlEc = xmlDoc.CreateElement("content");
    //设置节点文本
    xmlEc.InnerText = strContent;
    //添加到节点中
    xmlE.AppendChild(xmlEc);
    //创建一个子节点
    XmlElement xmlEd = xmlDoc.CreateElement("date");
    //获取时间的时分秒
    string strSendTime = DateTime.Now.Hour + ":" +
    DateTime.Now.Minute + ":" + DateTime.Now.Second;
    xmlEd.InnerText =strSendTime;
    //添加到节点中
    xmlE.AppendChild(xmlEd);

    //将节点加入根节点中
    xmlN.AppendChild(xmlE);
    //保存创建好的 XML 文档
    xmlDoc.Save(Server.MapPath(strFileName));
    Response.Write("您的点评已成功发表！");
%>
```

6.5.4　代码分析

为了即时侦察出客户端各种 Ajax 请求的进展状态，使用全局事件 ajaxStart 与 ajaxStop 绑

定客户端页面中 ID 号为“divMsg”的元素，在请求触发和停止时，显示不同的内容和状态，其代码如下所示：

```
// 元素绑定全局 ajaxStart 事件
$("#divMsg").ajaxStart(function() {
    $(this).show(); // 显示元素
})
// 元素绑定全局 ajaxStop 事件
$("#divMsg").ajaxStop(function() {
    $(this).html(" 数据处理已完成。").hide();
})
```

由于页面在初始化与提交点评内容后，都要加载 XML 格式的点评数据，为了避免重复，我们自定义一个 JS 函数 LoadData，其功能就是使用 $.ajax() 方法，请求加载 XML 格式的点评数据，其代码如下：

```
/*
    * 动态加载 XML 格式的点评数据
    */
    function LoadData() {
        $.ajax(
            {
                type: "GET",
                url: "CommentData.xml", // 请求 XML 格式数据
                dataType: "xml",
                cache: false,
                success: function(data) {
                    // 请求数据成功后执行的代码
                    ...
                }
            })
    }
```

在自定义读取 XML 点评数据的函数中，当请求成功而返回数据时，执行一个回调函数，在该函数中，先清空显示加载数据的页面元素，然后，遍历返回的数据，在遍历过程中，通过 find() 方法定位各指定名称的元素内容，并以叠加的方式保存在字符变量 strHTML 中，最后，通过元素的 html() 方法将数据显示在页面中，其实现的代码如下：

```
...省略部分代码
$(".divContent").empty(); // 先清空标记中的内容
var strHTML = ""; // 初始化保存内容变量
if ($(data).find("Data").length == 0) {// 如果没有找到数据
     strHTML = "<div style='text-align:center'>
                目前还没有找到点评数据！ </div>";
    }
$(data).find("Data").each(function() { // 遍历获取的数据
  var $strUser = $(this);
```

```
  strHTML += "<div class='clsShow'>";
  strHTML += "<div class='ShowTitle'>荣拓网友
  <a href=''>" + $strUser.find("name").text()+ "</a></div>";
  strHTML += "<div class='ShowContent'>" +
$strUser.find("content").text() + "</div>";
  strHTML += "<div class='ShowBottom'>发送时间   " +
$strUser.find("date").text() + "</div>"
  strHTML += "</div>"
})
  $(".divContent").html(strHTML); // 显示处理后的数据
...省略部分代码
```

当用户单击“提交”按钮时，先以加码的方式获取页面中“内容”和“用户名”的值，并将该值作为请求服务器的参数，执行一个 $.ajax() 方法，向服务器端页面 AddData.aspx 发出请求，其实现的代码如下所示：

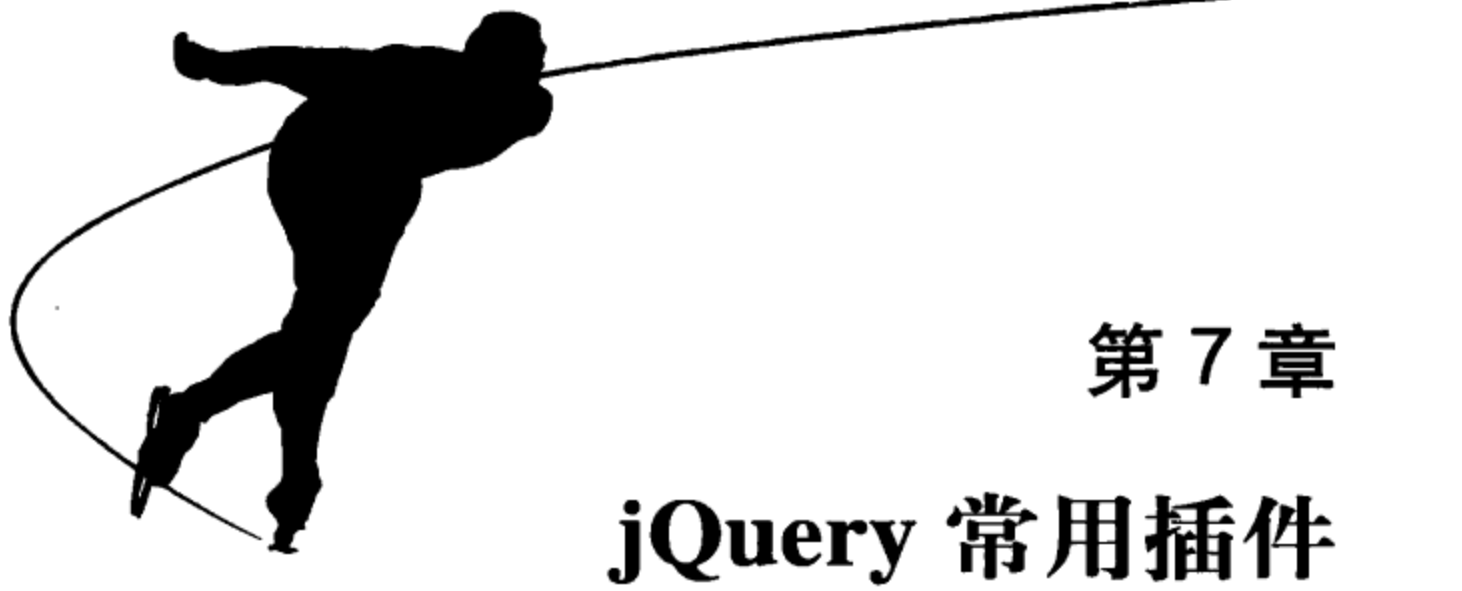
```
...省略部分代码
// 获取加码后的用户名
var strName = encodeURI($("#txtName").val());
// 获取加码后的发表内容
var strContent = encodeURI($("#txtContent").val());
$.ajax(
  {
    type: "GET",
    url: "AddData.aspx", // 请求增加数据动态页
    dataType: "html",
    data: { name: strName, content: strContent },
    success: function(msg) {
       // 请求数据成功后执行的代码
       ...
    }
})
...省略部分代码
```

在单击“提交”按钮，传递参数请求服务器并响应成功后，执行一个请求成功的回调函数，在该函数中，先弹出一个窗口，显示服务器传回的值，然后，重新执行自定义的函数 LoadData，即再次加载点评数据。最后，清空文本框“内容”和“用户名”中的值，其实现的代码如下所示：

```
...省略部分代码
alert(msg);
LoadData();
$("#txtName").val("");
$("#txtContent").val("");
...省略部分代码
```

6.6 本章小结

本章以理论与实例相结合的方式，介绍 jQuery 中支持 Ajax 的各种方法，然后，阐述了客户端与服务器端通过 Ajax 互相访问的方式，并详细介绍了核心方法 $.ajax() 的运用，最后，系统介绍了 Ajax 中的全局函数与事件，并结合一个新闻点评的应用示例，帮助巩固前面所学的全部知识。



第 7 章

jQuery 常用插件

本章内容

- jQuery 插件概述
- 验证插件 validate
- 表单插件 form
- Cookie 插件 cookie
- 搜索插件 AutoComplete
- 图片灯箱插件 notesforlightbox
- 右键菜单插件 contextmenu
- 图片放大镜插件 jqzoom
- 自定义 jQuery 插件
- 综合案例分析——使用 uploadify 插件实现文件上传功能
- 本章小结

虽然使用 jQuery 库可以满足绝大部分的应用需求，但是随着各种应用的层出不穷，我们就要在考虑主体程序库大小与通用性的前提下，如何丰富 jQuery 库中的功能，以满足人们对一些特定应用的需求。毫无疑问，jQuery 中的插件是解决这一问题的最好答案。

目前，有超过近百种的各类插件应用在全球的各种项目中。插件的使用，充分展示了 jQuery 中又一核心功能——强大的易展性（Extension），下面就开始我们的插件之旅吧！

7.1 jQuery 插件概述

插件是以 jQuery 的核心代码为基础，编写出符合一定规范的应用程序，并将程序进行打包，调用时，仅需要包含该打包后的 JS 文件即可。如需要使用表单插件，按下列步骤就可以实现插件的调用。

第一步：在页面中导入包含表单插件的 JS 文件，并确定它位于主 jQuery 库后，其代码如下：

```
...
<head>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery.form.js">
    </script>
</head>
...
```

第二步：在 JS 文件或页面 JS 代码中，编写如下代码，便完成了插件的调用：

```
$(function() {
     $("form").ajaxSubmit();
})
```

代码中 .ajaxSubmit() 就是调用插件中的方法，其作用就是获取表单中的字段值，向服务器发送请求。

当前，最新的插件可以从 jQuery 官方网站（http://plugins.jquery.com）中获取，其页面如图 7-1 所示。

7.2 验证插件 validate

validate 是一个十分优秀的表单验证插件之一，它广泛地使用在全球各个的项目中，并得到广大程序开发人员的认可，该插件具有以下几个功能。

1) 自带验证规则：其中包含必填、数字、URL 等众多验证规则。

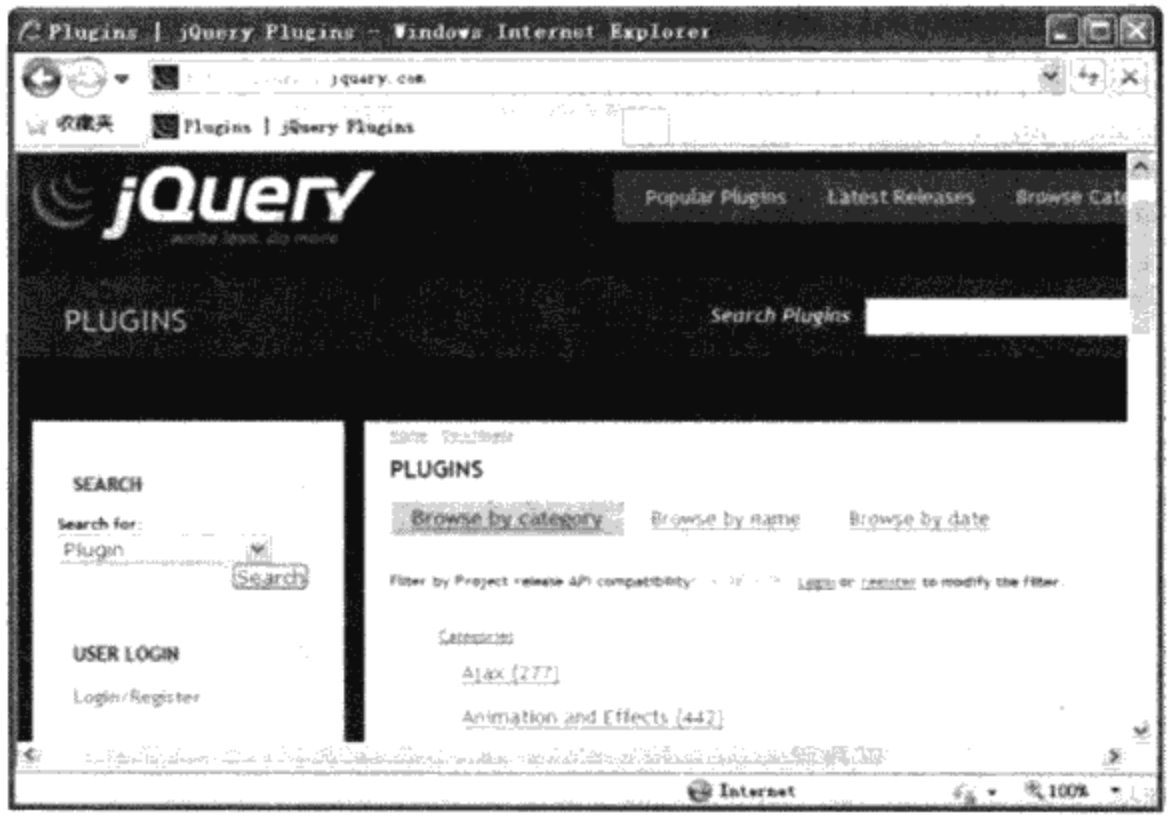


图 7-1　jQuery 插件下载官网页面

2）验证信息提示：可以使用默认的提示信息，也可以自定义提示信息，覆盖默认内容。

3）多种事件触发：不仅在表单提交时触发验证，而且在“keyup”或“blur”事件中也能触发。

4）允许自定义验证规则：除使用自带的验证规则外，开发者还可以很方便地自定义验证规则。

下面举例说明这个插件的使用方法。

示例 7-1　验证插件的使用

（1）插件文件

Js-7-1/jquery.validate.js

Js-7-1/jquery.validate.messages_cn.js

（2）下载地址

http://plugins.jquery.com/project/validate

（3）功能描述

在页面中，创建一个表单标记，Id 号为“frmV”，在表单中设置两个文本框，一个用于输入“姓名”，另一个用于输入“邮箱”，当单击“提交”按钮，提交表单数据时，通过 validate 插件，根据设置的验证规则，检测表单中的各个字段元素。

（4）实现代码

新建一个 HTML 文件 7-1.html, 加入如代码清单 7-1 所示的代码。

代码清单 7-1　使用 validate 插件验证表单提交时的基本信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>validate 验证插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-1/jquery.validate.js">
    </script>
    <script type="text/javascript"
            src="Js-7-1/jquery.validate.messages_cn.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#frmV").validate(
              {
                  /* 自定义验证规则 */
                  rules: {
                      username: { required: true, minlength: 6 },
                      email: { required: true, email: true }
                  },
                  /* 错误提示位置 */
                  errorPlacement: function(error, element) {
                      error.appendTo(element.siblings("span"));
                  }
              }
            );
        })
    </script>
</head>
<body>
     <form id="frmV" method="get" action="#">
     <div class="divFrame">
         <div class="divTitle">
              请输入下列资料
         </div>
         <div class="divContent">
              <div>
              用户名：<br />
              <input id="username" name="username"
                     type="text" class="txt" />
              <font color="red">*</font><br />
```

```
                <span></span>
                </div>
                <div>
                邮箱：<br />
                <input id="email" name="email"
                       type="text" class="txt" />
                <font color="red">*</font><br />
                <span></span>
                </div>
            </div>
            <div class="divBtn">
                <input id="sbtUser" type="submit"
                       value=" 提交 " class="btn" />
            </div>
        </div>
        </form>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图 7-2 所示。

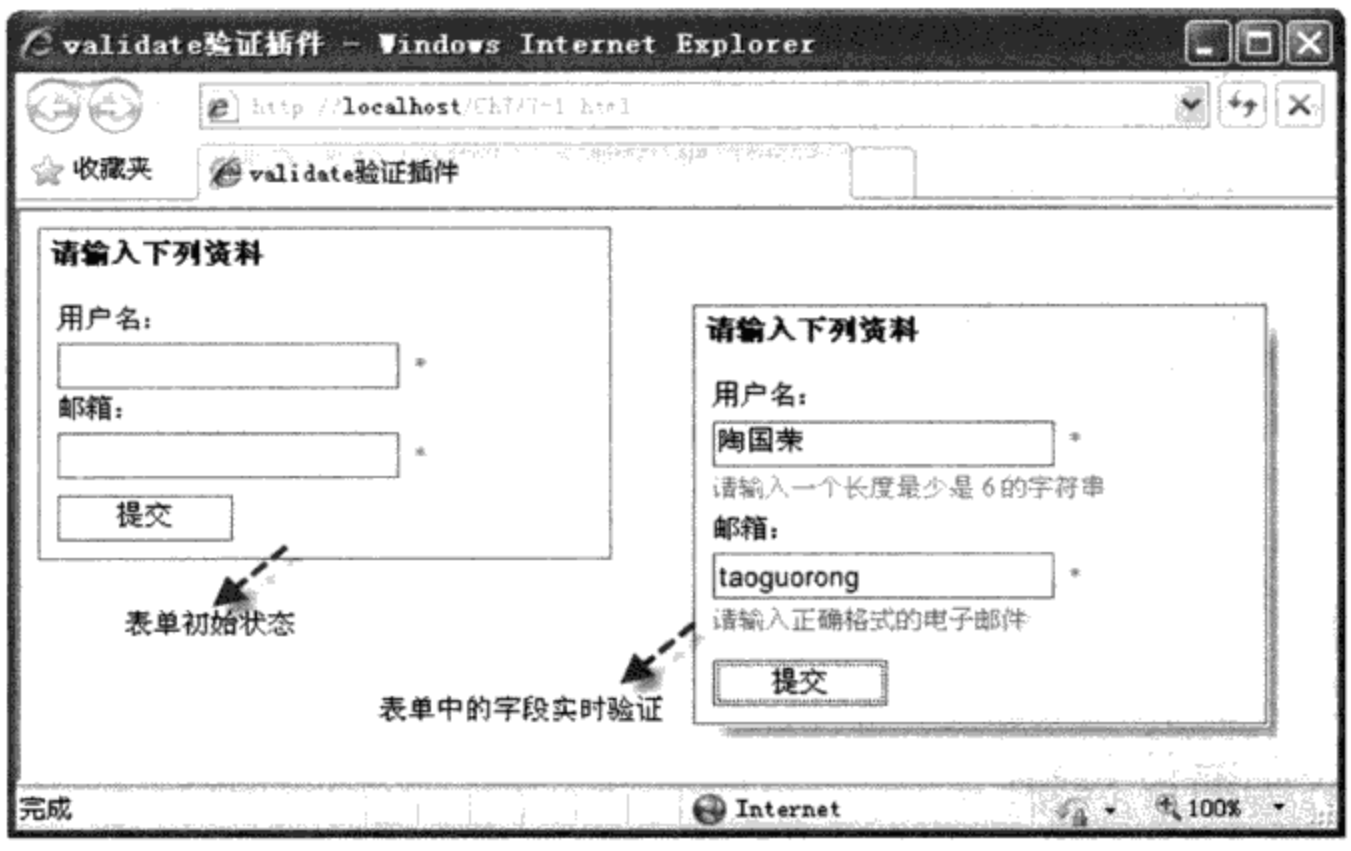


图 7-2　validate 验证插件

（6）代码说明

由于 validate 验证插件默认的提示信息是英文版的，因此为了汉化验证提示信息，需要页面部分引用一个中文验证信息库，其代码如下。

```
... 省略部分代码
<script type="text/javascript"
          src="Js-7-1/jquery.validate.messages_cn.js">
</script>
... 省略部分代码
```

导入该中文验证信息库后，就可以将显示的验证提示信息汉化。

验证插件 validate 导入完成后，如果要使表单在提交数据时触发验证，只需加入如下代码：

```
$(function() {
              $("#frmV").validate(
                {
                    // 表单验证时执行的代码
                    ...
                 }
              );
})
```

在本示例中，由于进行的是简单的“用户名”与“邮箱”格式的验证，因此，只要在 validate() 方法中增加一个规则（rules）属性，用表单字段的“name”属性与规则进行关联，声明默认的验证规则，其实现的代码如下所示：

```
/* 自定义验证规则 */
rules: {
          username: { required: true, minlength: 6 },
          email: { required: true, email: true }
}
```

为了使验证后的提示信息显示在页面指定的位置中，在 validate() 方法中，新增加一个提示信息位置属性，该属性执行一个回调函数，传递出错提示信息与执行验证字段两个参数，把提示信息的内容通过 appendTo() 的方法，增加到触发验证字段的“兄弟”元素 中，其实现的代码如下所示：

```
/* 错误提示位置 */
errorPlacement: function(error, element) {
  error.appendTo(element.siblings("span"));
}
```

7.3 表单插件 form

form 插件是专门为页面的表单而设计的，引入该插件后，通过调用 ajaxForm() 或 ajaxSubmit() 两个方法，可以很容易地实现 Ajax 方式提交数据，并通过方法中的 options 对象，设置参数、获取服务器返回的数据，同时，该插件还包含如下的一些重要方法。

1）formSerialize()：用于格式化表单中有用的数据，并将其自动整理成适合 Ajax 异步请求的 URL 地址格式。

2）clearForm()：清除表单中所有输入值的内容。

3）restForm()：重置表单中所有的字段内容，即将所有表单中的字段内容都恢复到页面加载时的默认值。

下面举例说明这个插件的使用方面。

示例 7-2　表单插件的使用

（1）插件文件

Js-7-2/jquery.form.js

（2）下载地址

http://plugins.jquery.com/project/form

（3）功能描述

在页面中创建一个 Id 号为“frmUserInfo”的表单，在表单中新建一个文本框，用于输入“用户名”，新建一个密码文本框，用于输入“密码”，当单击“提交”按钮后，将向服务器文件 Login.aspx 发送请求，提交表单中的各元素值，服务器响应请求，将返回的内容显示在 Id 号为“divData”页面元素中。

（4）实现代码

新建一个 HTML 文件 7-2.html, 加入如代码清单 7-2 所示的代码。

代码清单 7-2　使用 form 插件实现表单数据的提交

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>form 表单插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-2/jquery.form.js">
    </script>
    <style type="text/css">
          ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
        //定义一个表单提交时的对象
        var options = {
             //默认为 form 中的 action, 设置后便覆盖默认值
             url:"Login.aspx",
             //将服务器返回的数据显示在 Id 号为 divData 元素中
             target: "#divData"
            }
```

```
            // 以 Ajax 的方式提交表单
            $("#frmUserInfo").ajaxForm(options);
        })
    </script>
</head>
<body>
    <form id="frmUserInfo" method="get" action="#">
    <div class="divFrame">
        <div class="divTitle">
            用户登录
        </div>
        <div class="divContent">
            <div>
            用户名：<br />
            <input id="username" name="username"
                type="text" class="txt" />
            </div>
            <div>
            密码：<br />
            <input id="userpass" name="userpass"
                type="password" class="txt" />
            </div>
        </div>
        <div class="divBtn">
            <input id="sbtUser" type="submit"
                value=" 提交 " class="btn" />
        </div>
        <div id="divData"></div>
    </div>
    </form>
</body>
</html>
```

服务器端文件 Login.aspx 的代码如下列清单所示：

```
<%@ Page Language="C#" ContentType="text/html"ResponseEncoding="gb2312" %>
<%
    string strName = Request["username"]; // 姓名字符
    string strPass = Request["userpass"]; // 密码字符
    string strRetValue = " 用户名：" + strName + "<br> 密码：" + strPass;
    Response.Write(strRetValue);
%>
```

（5）页面效果

代码执行后的效果如图 7-3 所示。

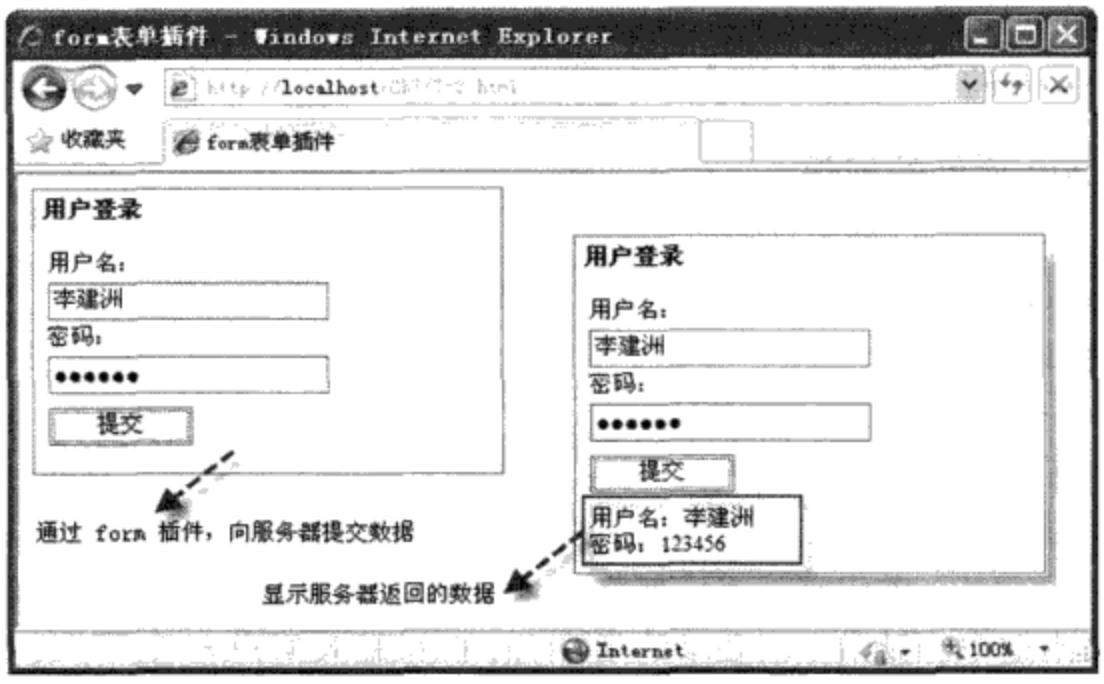


图 7-3　form 表单插件

（6）代码说明

在导入 form 表单插件后，提交数据变得十分轻松，仅仅一行代码：

```
$("#frmUserInfo").ajaxForm();
```

这行代码也等于如下代码：

```
$("#frmUserInfo").submit(function(){
  $("#frmUserInfo").ajaxSubmit();
  return false;
})
```

表单插件 form 中的 ajaxForm() 与 ajaxSubmit() 方法中，既可以没有参数，也能传递 1 个参数，该参数既可以是一个回调型函数，也可以是一个 options 对象，通过该对象可以实现更多的页面互动功能，该对象常用的属性如下所示：

```
var options={
      url:url, //form 提交数据的地址（url）
      type:type, //form 提交的方式（method）
      target:target, // 显示服务器返回数据的元素 Id 号
      beforeSubmit:function(), // 提交前执行的回调函数
      success:function(), // 提交成功后执行的回调函数
      dataType:null, // 服务器返回数据类型
      clearForm:true, // 提交成功后，清空表单中的字段值
      restForm:true, // 提交成功后，重置表单中的字段值
      timeout:6000 // 设置请求时间，超过该时间后，自动退出请求，单位（毫秒）
}
```

7.4　Cookie 插件 cookie

在 jQuery 中，引入 cookie 插件后，可以很方便地定义某个 cookie 名称，并设置 cookie

值。通过设置好的 cookie，可以很便利地保存用户的页面浏览记录，在用户选择保存的情况下，还可以保存用户的登录信息。下面举例说明。

示例 7-3　cookie 插件的使用

（1）插件文件

Js-7-3/jquery.cookie.js

（2）下载地址

http://plugins.jquery.com/project/cookie

（3）功能描述

在示例 7-2 的页面中，再增加一个是否保存用户名的复选框，当选中复选框时，单击“提交”按钮，将完成 cookie 的设置。当重新打开浏览器时，由于使用了 cookie 保存用户名，因此，在用户名文本框中，显示上次登录的用户名；如果输入用户名后，不选中复选框，则单击“提交”按钮，将销毁已存在的 cookie 对象。

（4）实现代码

新建一个 HTML 文件 7-3.html, 加入如代码清单 7-3 所示的代码。

代码清单 7-3　使用 cookie 插件保存表单中的用户名

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>cookie 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-3/jquery.cookie.js">
    </script>
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            if ($.cookie("UserName")) { // 如果有值，则显示在文本框中
                $("#UserName").val($.cookie("UserName"));
            }
            $("#sbtUser").submit(function() { // 表单提交事件
            // 如果选中了保存 " 用户名 " 项
            if ($("#chkSave").attr("checked")) {
                    // 设置 Cookie 值
                    $.cookie("UserName", $("#UserName").val(), {
                        path: "/", expires: 7
```

```
                    })
                }
                else {
                    // 销毁 Cookie 值
                    $.cookie("UserName", null, {
                        path: "/"
                    })
                }
                return false; // 表单不提交
            })
        })
    </script>
</head>
<body>
    <form id="frmUserInfo" method="get" action="#">
    <div class="divFrame">
        <div class="divTitle">
            用户登录
        </div>
        <div class="divContent">
            <div>
            用户名：<br />
            <input id="UserName" name="UserName"
                type="text" class="txt" />
            </div>
            <div>
            密码：<br />
            <input id="UserPass" name="UserPass"
                type="password" class="txt" />
            </div>
            <div><input id="chkSave" type="checkbox" />
                是否保存用户名
            </div>
        </div>
        <div class="divBtn">
            <input id="sbtUser" type="submit"
                value="提交" class="btn" />
        </div>
        <div id="divData"></div>
    </div>
    </form>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图 7-4 所示。

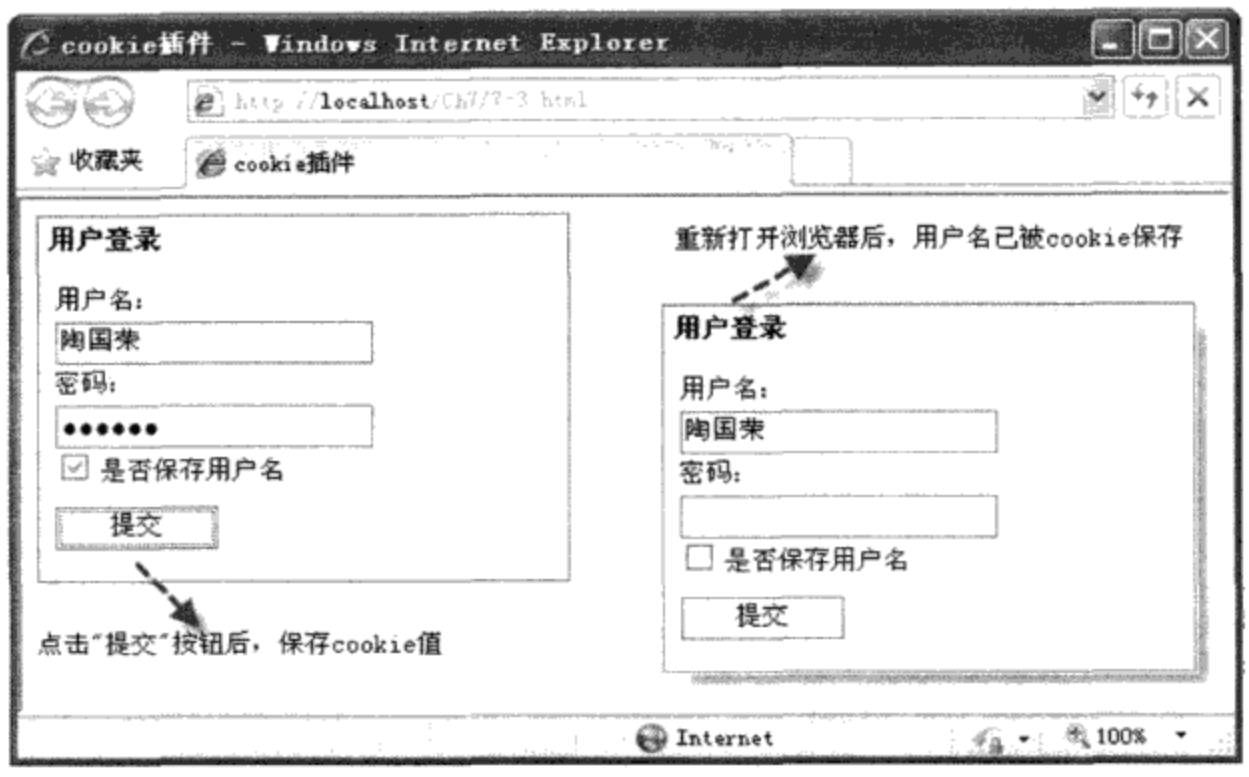


图 7-4　cookie 插件

（6）代码说明

在导入 cookie 插件后，可以通过一个全局性的方法管理客户端的 cookie 对象，格式如下：

```
$.cookie( 名称, 值, [option])
```

如果是写入或设置 cookie 值，其调用的格式如下：

```
$.cookie(cookieName,cookieValue)
```

其中，参数“cookieName”表示要设置的 cookie 名称，“cookieValue”表示相对应的值。

如果是读取 cookie 值，其调用的格式如下：

```
$.cookie(cookieName)
```

其中，参数“cookieName”表示要读取的 cookie 名称。

如果是销毁 cookie 值，其调用的格式如下：

```
$.cookie(cookieName, NULL)
```

其中，参数“cookieName”表示要销毁的 cookie 名称，在销毁的 cookie 时，其可选项参数中的路径 (path) 和域名 (domain) 必须与设置时一样，否则不能销毁。

另外，在方法 $.cookie 中，可选项参数 [option] 以对象的形式展示，用于补充说明设置的 cookie 对象，其常用的属性如下：

```
$.cookie(cookieName,cookieValue, {
   expires: // 有限日期，可以是一个整数或一个日期（单位：天）
   path: //cookie 值被保存的路径，默认值与创建页路径一致
   domin: //cookie 域名属性，默认值与创建页域名一样
   secure: // 一个布尔值，表示传输 cookie 值时，是否需要一个安全协议
})
```

7.5　搜索插件 AutoComplete

AutoComplete 为自动填充、展示之意。在 jQuery 中，引入该插件后，用户在使用文本框搜索信息时，使用插件中的 autocomplete 方法绑定文本框。当在文本框中输入某个字符时，通过该方法中的指定的数据 URL，返回相匹配的数据，自动显示在文本框下，提醒用户进行选择。下面举例说明。

示例 7-4　搜索插件的使用

（1）插件文件

Js-7-4/jquery.autocomplete.js

Css-7-4/jquery.autocomplete.css

Images-7-4/indicator.gif

（2）下载地址

http://jquery.bassistance.de/autocomplete/jquery.autocomplete.zip

（3）功能描述

在页面中，创建一个文本框，用于输入查询信息，当在文本框中输入字符时，通过 AutoComplete 插件，绑定设置好的数组信息，并与文本框中的字符匹配，将匹配后的结果序列化后，展示在文本框的底部，供用户选择。

（4）实现代码

新建一个 HTML 文件 7-4.html, 加入如代码清单 7-4 所示的代码。

代码清单 7-4　AutoComplete 插件提示用户搜索信息

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>autocomplete 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-4/jquery.autocomplete.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-7-4/jquery.autocomplete.css" />
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            var arrUserName = [" 张三 ", " 王小五 ", " 张才子 ",
            " 李四 ", " 张大三 ", " 李大四 ", " 王五 ", " 刘明 ",
```

```
            "李小四", "刘促明", "李渊", "张小三", "王小明"];
            $("#txtSearch").autocomplete(arrUserName, {
                minChars: 0 // 双击空白文本框时显示全部提示数据
            });
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            搜索用户
        </div>
        <div class="divContent">
            <span style="padding:0 5px 0 10px">
                <a href="#">新闻</a></span>
            <span style="padding:0 5px 0 5px">
                <b>用户</b></span>
            <div>
                <input type="text" id="txtSearch" class="txt" />
                <input type="button" id="btnSearch"
                    value="查一下" class="btn" />
            </div>
        </div>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图 7-5 所示。

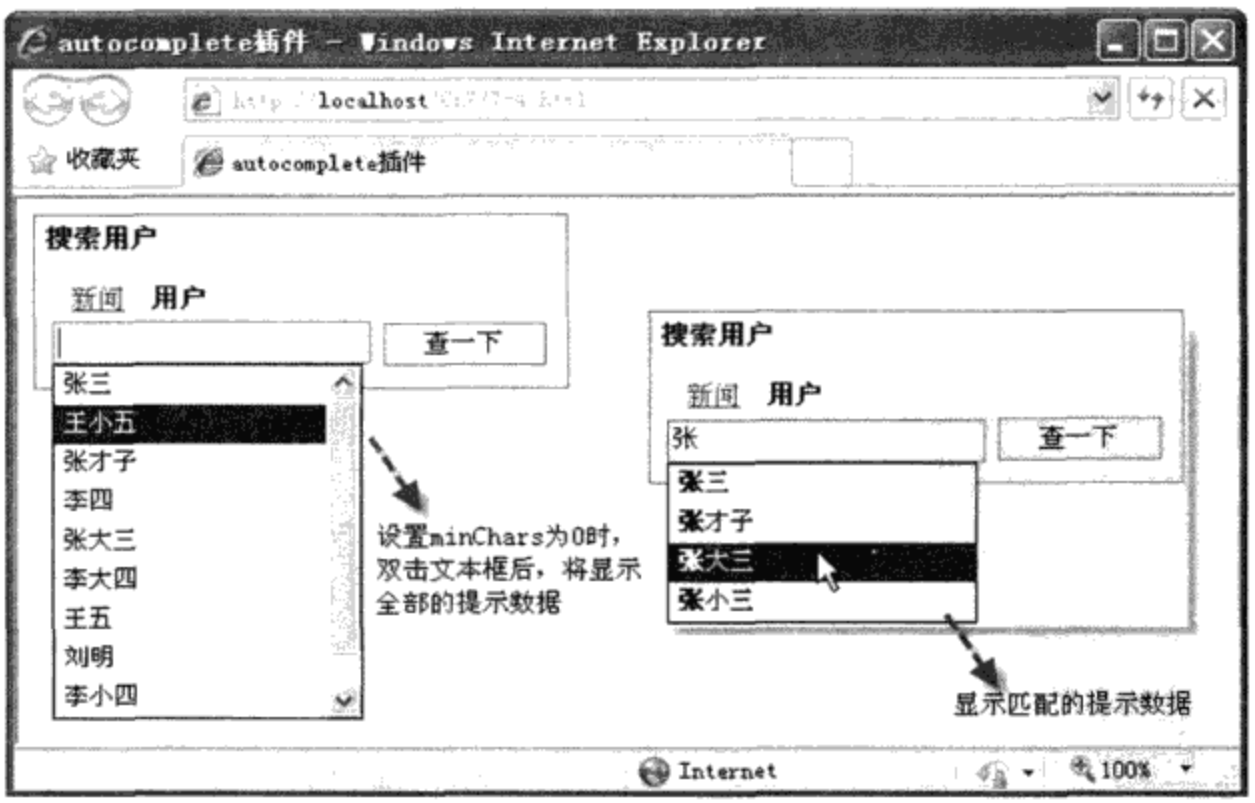


图 7-5 AutoComplete 插件提示搜索数据

（6）代码说明

在页面的 JS 代码中，由于导入了 AutoComplete 插件，使用的代码十分简单，首先自定义一个数组，用于与文本框中的数据相匹配，其代码如下：

```
var arrUserName = ["张三", "王小五", "张才子",
     "李四", "张大三", "李大四", "王五", "刘明",
     "李小四", "刘促明", "李渊", "张小三", "王小明"];
```

在实际的开发中，这部分数据可以与某个页面的 Url 绑定获取，同样也可以实现提示效果，在设置完数据源后，调用插件的 AutoComplete 方法，完成数据与文本框间的绑定。其代码如下：

```
$("#txtSearch").autocomplete(arrUserName, {
    minChars: 0 // 双击空白文本框时显示全部提示数据
 });
```

AutoComplete 插件的重要方法是 autoComplete，其调用的语法格式如下：

```
$(文本框元素).autocomplete(urlOrData, [option])
```

其中参数 urlOrData 表示绑定数据的路径或名称，可选项参数 [option] 为一个对象，其所需配置的属性名称如下所示：

```
$(文本框元素).autocomplete(urlOrData,
{
    minChars:// 表示自动完成前填入的最小字符，如果为 0，则双击空白文本时，显示全部的提示数据信息
    max:// 表示显示的提示数据总条数
    autoFill:// 布尔值，表示是否选中时自动填充至文本框
    mustMatch:// 布尔值，表示是否匹配数据，设为 True 时，则文本框中内容必须是匹配数据，如果不是，则清空文本框。
    matchContains:// 布尔值，表示是否包含匹配数据，设置为 True 时，则文本框中内容只要被匹配数据包含则显示提示数据，否则不显示。
       scrollHeight：// 设置匹配数据滚动显示的高度
   formatItem: function(data, i, total) {
          return "<I>"+data[0]+"</I>";
    }, // 格式化匹配数据显示的格式，如使用 <I> 标记
  formatMatch: function(data, i, total) {
          return data[0];
  },  // 未格式化的源匹配数据
  formatResult: function(data) {
          return data[0];
  }  // 返回最终匹配的结果
})
```

在示例 7-5 中，我们对 JS 部分的代码再次进行修改，增加两个功能，一则是改变匹配数据显示的格式，另外，单击“查一下”按钮后，显示所选中的匹配结果。

```
...省略部分代码
$("#txtSearch").autocomplete(arrUserName, {
```

```
        minChars: 0, //双击空白文本框时显示全部提示数据
        formatItem: function(data, i, total) {
            return "<I>" + data[0] + "</I>"; //改变匹配数据显示的格式
        },
        formatMatch: function(data, i, total) {
            return data[0];
        },
        formatResult: function(data) {
            return data[0];
        }
    }).result(SearchCallback);//选中匹配数据中的某项数据时，调用插件的 result 方法
    //自定义返回匹配结果函数
    function SearchCallback(event, data, formatted) {
        $("#divData").html("您的选择是：" + (!data ? "空" : formatted));
    }
    //点击"查一下"按钮后，触发插件的 search() 方法
    $("#btnSearch").click(function() {
        $("#txtSearch").search();
    });
    ...省略部分代码
```

通过 JS 代码改选后，其执行后的效果如图 7-6 所示。

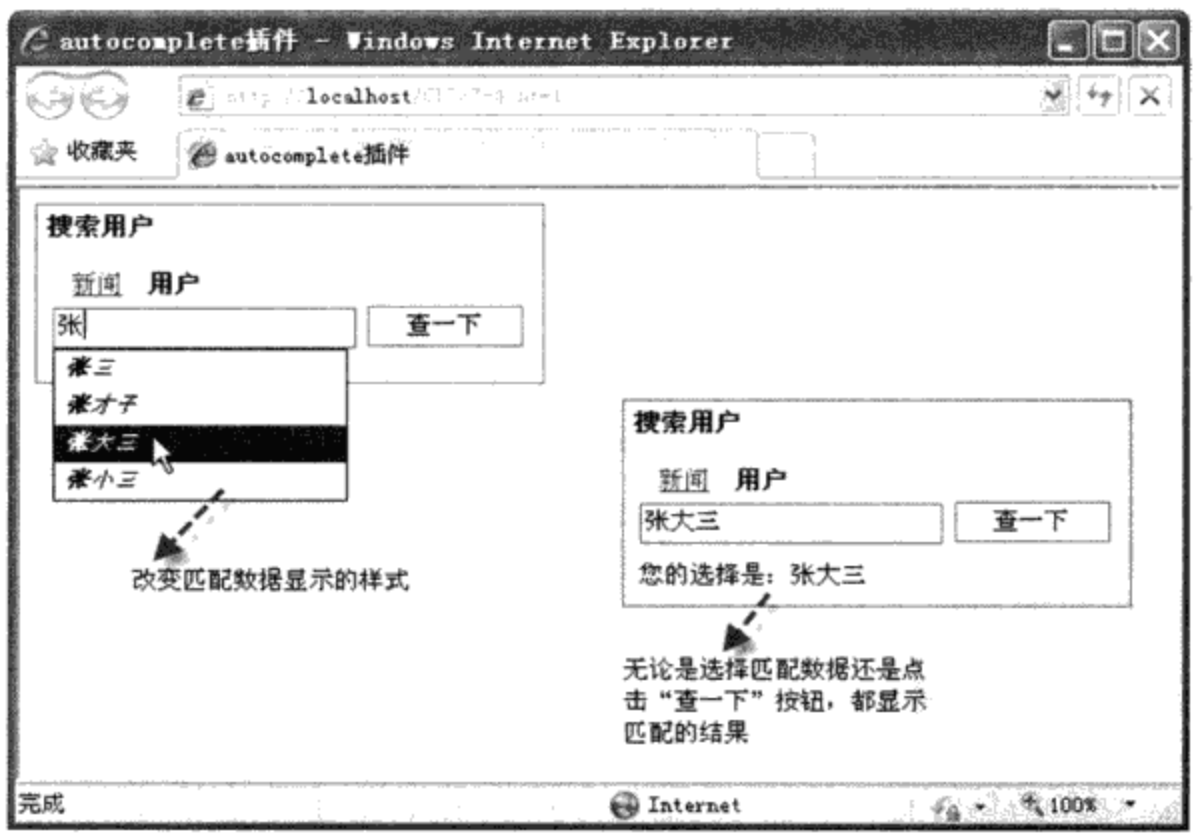


图 7-6 调用 AutoComplete 插件中的方法

说明 autoComplete 插件中有两个重要的方法，一个是 result(handler) 方法，用于响应查看匹配的结果，其中参数 handler 为自定义的函数；另一个是 search() 方法，一般与“查询”按钮关联，调用绑定的 result(handler) 方法。

7.6　图片灯箱插件 notesforlightbox

Notesforlightbox 是一个基于 jQuery 基础开发的图片放大浏览插件，它支持绝大部分浏览器，广泛应用在图片查看的项目中，该插件有以下几个强大的功能：

1）以圆角的方式展示选中的图片。

2）以按钮式查看“上一张”或“下一张”图片。

3）加载图片时带有进度条，显示加载进度。

4）可以采用自动播放的方式浏览图片。

5）多个样式属性可以随意设置。

下面通过示例来演示其功能。

示例 7-5　图片灯箱插件的使用

（1）插件文件

Js-7-5/jquery.notesforlightbox.js

Css-7-5/jquery.notesforlightbox.css

Images-7-5/ 全部图片

Pic-7-5/ 全部图片

（2）下载地址

http://www.notesfor.net/file.axd?file=NFLightBox.zip

（3）功能描述

在页面中，以列表的方式，展示某个相册中的全部图片，当用户单击其中某张图片时，通过引入的 notesforlightbox 插件，采用“灯箱”式显示所选中的图片。

（4）实现代码

新建一个 HTML 文件 7-5.html, 加入如代码清单 7-5 所示的代码。

代码清单 7-5　notesforlightbox 插件浏览相册中的图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>notesforlightbox 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-5/jquery.notesforlightbox.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-7-5/jquery.notesforlightbox.css" />
    <style type="text/css">
```

```
            ...省略样式部分代码
        </style>
        <script type="text/javascript">
            $(function() {
                $('.divPics a').lightBox({
                    overlayBgColor: "#666", // 浏览图片时的背景色
                    overlayOpacity: 0.5, // 背景色的透明度
                    containerResizeSpeed: 600 // 图片切换时的速度;
                })
            })
        </script>
    </head>
    <body>
        <div class="divFrame">
            <div class="divTitle">
                我的相册
            </div>
            <div class="divContent">
                <div class="divPics">
                    <ul>
                        <li><a href="Pic-7-5/img01.jpg"
                               title="第1篇风景图片">
                               <img src="Pic-7-5/img01.jpg" alt="" />
                        </a></li>
                        <li><a href="Pic-7-5/img02.jpg"
                               title="第2篇风景图片">
                               <img src="Pic-7-5/img02.jpg" alt="" />
                        </a></li>
                        <li><a href="Pic-7-5/img03.jpg"
                               title="第3篇风景图片">
                               <img src="Pic-7-5/img03.jpg" alt="" />
                        </a></li>
                        <li><a href="Pic-7-5/img04.jpg"
                               title="第4篇风景图片">
                               <img src="Pic-7-5/img04.jpg" alt="" />
                        </a></li>
                        <li><a href="Pic-7-5/img05.jpg"
                               title="第5篇风景图片">
                               <img src="Pic-7-5/img05.jpg" alt="" />
                        </a></li>
                        <li><a href="Pic-7-5/img06.jpg"
                               title="第6篇风景图片">
                               <img src="Pic-7-5/img06.jpg" alt="" />
                        </a></li>
                    </ul>
                </div>
            </div>
        </div>
    </body>
    </html>
```

（5）页面效果

代码执行后，出现相册的效果如图 7-7 所示。

图 7-7　在相册中选中某张照片时

在相册中，当用户单击某张照片后，通过 notesforlightbox 插件，以一种“灯箱”的方式浏览该单击的图片，其效果如图 7-8 所示。

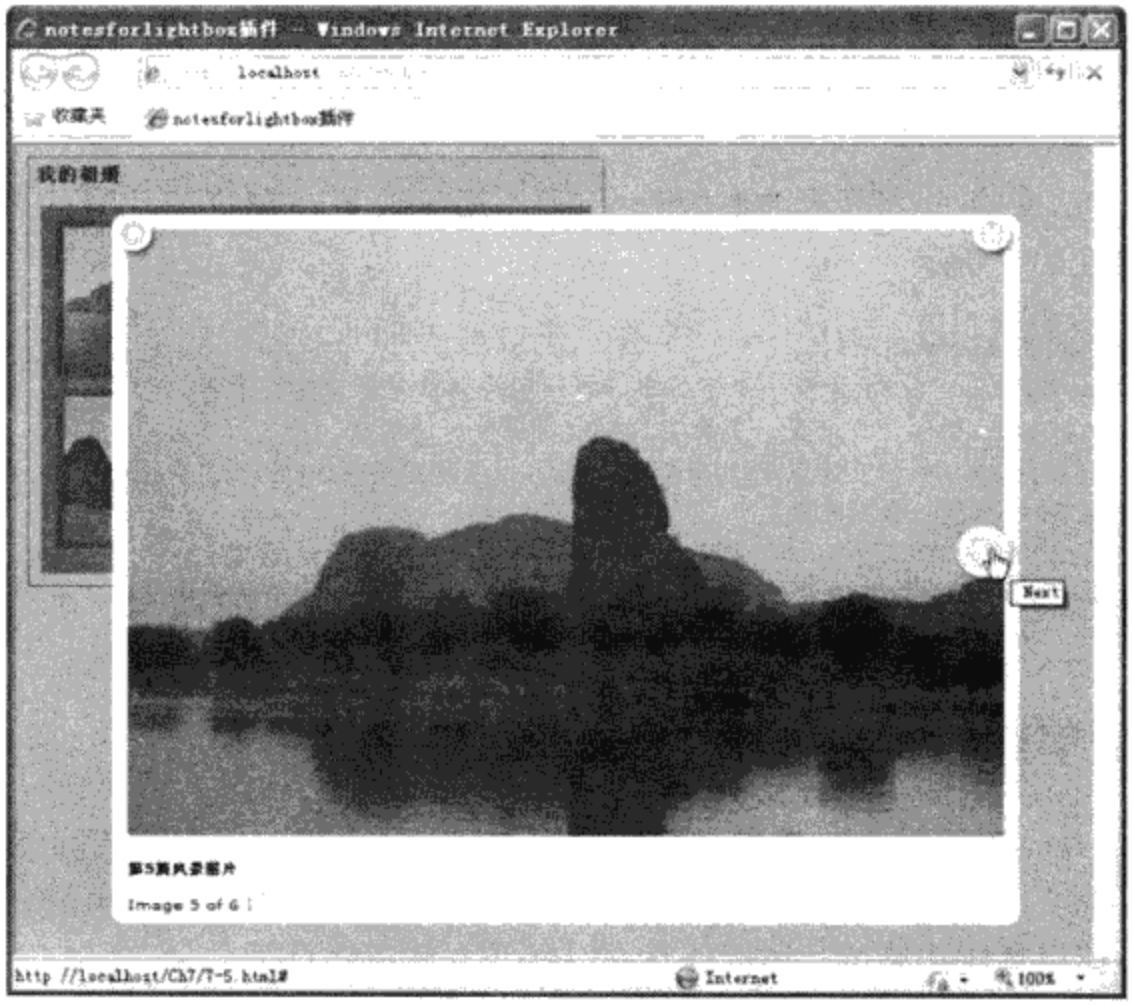


图 7-8　浏览选中的某张图片时

（6）代码说明

为了更好地展现 notesforlightbox 插件浏览图片的效果，除了在页面中引入 JS 插件库文件外，还应包含一个该插件所固有的 CSS 文件 jquery.notesforlightbox.css，其代码如下：

```
<link rel="stylesheet" type="text/css"
      href="Css-7-5/jquery.notesforlightbox.css" />
```

当然，用户可以对该 CSS 文件进行修改，以实现更好的页面效果。同时，该插件的一些功能性图片，如“上一张”、“下一张”图片默认是保存在文件夹 Image 中，如果其保存地址发生了变化，应打开 JS 插件文件 jquery.notesforlightbox.js，找到下列代码，进行设置：

```
...
// Configuration related to images
imageLoading: 'Images-7-5/loading.gif',
imageBtnPrev: 'Images-7-5/prev.png',
imageBtnNext: 'Images-7-5/next.png',
imageBtnClose: 'Images-7-5/close.png',
imageBlank: 'Images-7-5/lightbox-blank.gif',
imageBtnBottomPrev: 'Images-7-5/btm_prev.gif',
imageBtnBottomNext: 'Images-7-5/btm_next.gif',
imageBtnPlay: 'Images-7-5/start.png',
imageBtnStop: 'Images-7-5/pause.png',
...
```

全部设置完成以后，接下来的工作就是将页面中的图片与插件相关联，首先找到页面中的图片，然后通过插件中的 lightBox() 方法，绑定页面中找到的图片，其实现的代码如下：

```
$('.divPics a').lightBox();
```

lightBox() 方法中有一个可选项参数 [option]，用于设置在浏览图片时的各种页面属性，其常用的各种属性如下所示：

```
$('.divPics a').lightBox({
      overlayBgColor：// 设置一个字符串，表示图片浏览时页面的背景色
      overlayOpacity: // 设置一个数字，表示背景色的透明度，范围（0.0-0.9）
      fixedNavigation:// 设置一个布尔值，表示是否隐藏图片中的导航按钮，默认为 true，表示只有
      鼠标移动时才出现，否则隐藏。
      containerBorderSize：// 设置一个数字，表示图片的内容边框宽度。
      containerResizeSpeed：// 设置一个数字，表示图片间相互切换的速度，默认为 500，单位是毫秒
t});
```

7.7 右键菜单插件 contextmenu

contextmenu 是一款轻量型、功能完善的插件，利用该插件可以在页面的任何位置，设置一个触发右键事件的元素，当选中该元素，单击鼠标右键时，通过插件中的 contextMenu 方法，弹出一个设计精美快捷菜单。该插件具有以下几个显著特点：

1）可以在同一个页面中设置多个不同样式的菜单。

2）一个菜单可以绑定页面中的多个元素。

3）可随意设置菜单样式。

4）轻松访问与绑定菜单中的各选项。

下面通过示例来演示其用法。

示例 7-6　右键菜单插件的使用

（1）插件文件

Js-7-6/jquery.contextmenu.js

Css-7-6/jquery.contextmenu.css

Images-7-6/ 全部图片

（2）下载地址

http://www.trendskitchens.co.nz/jquery/contextmenu/jquery.contextmenu.r2.js

（3）功能描述

在页面中，创建一个文本编辑框，选中该文本框，单击鼠标右键，弹出一个设置好的快捷菜单，单击菜单选项时，显示所选择的选项内容。

（4）实现代码

新建一个 HTML 文件 7-6.html, 加入如代码清单 7-6 所示的代码。

代码清单 7-6　contextmenu 插件弹出快捷菜单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>ContextMenu 插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-6/jquery.contextmenu.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-7-6/jquery.contextmenu.css" />
    <style type="text/css">
          ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $('#txtContent').contextMenu('sysMenu',
              { bindings:
                 {
                     'Li1': function(Item) {
                         alert(" 在 ID 号为 : " + Item.id +
```

```
                                    "编辑框中，您点击了"新建"项");
                        },
                        'Li2': function(Item) {
                            alert("在ID号为：" + Item.id +
                                    "编辑框中，您点击了"打开"项");
                        }
                        //设置其他选项事件
                        //...
                    }
                });
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">点击右键</div>
        <div class="divContent">
            <textarea id="txtContent" cols="30"
                                    rows="5"></textarea>
        </div>
    </div>
    <div class="contextMenu" id="sysMenu">
        <ul>
            <li id="Li1"><img src="Images-7-6/new.png" /> 新建</li>
            <li id="Li2"><img src="Images-7-6/folder.png" /> 打开</li>
            <li id="Li3"><img src="Images-7-6/disk.png" /> 保存</li>
            <hr />
            <li id="Li4"><img src="Images-7-6/cross.png" /> 退出</li>
        </ul>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图 7-9 所示。

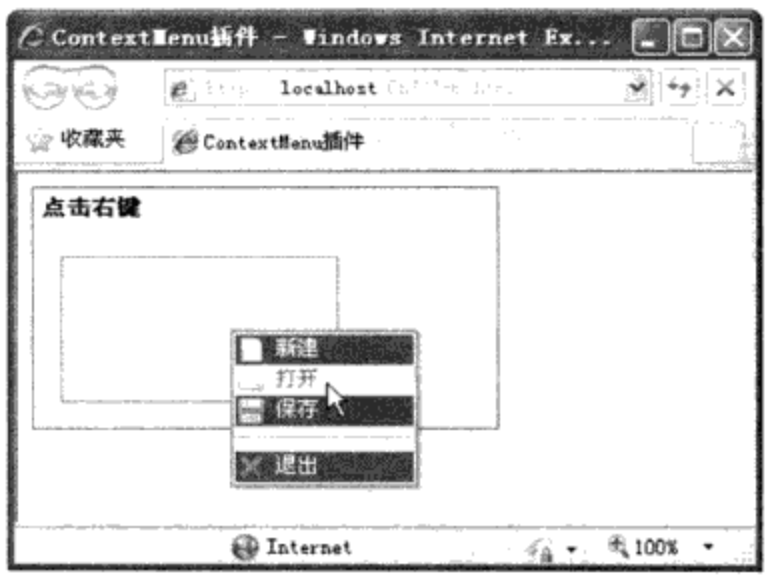


图 7-9　contextmenu 插件弹出快捷菜单

（6）代码说明

为了更好地实现页面效果，首先下载插件的 JS 文件，并重新命名为 jquery.contextmenu.js，然后在页面的 <title> 标记中导入，其执行的代码如下：

```
<script type="text/javascript"
          src="Js-7-6/jquery.contextmenu.js">
</script>
```

在 HTML 页面中，调用插件的方法如下：

```
$(element).contextMenu(Mid [,Option])
```

其中字符参数 Mid 表示与菜单绑定的元素 ID 号，可选项对象 Option 可以设置元素与菜单绑定时的属性或事件，其包含的属性和事件代码如下：

```
$(element).contextMenu(Mid ,
      {
        bindings{// 绑定菜单时，根据 ID 号获取或设置各选项属性和事件
            " 选项 ID 号 ", function(T){
                // 参数 T 为被绑定的元素
                // 获取或设置该选项的属性
                ...
            }
        }
      }
)
```

在上述示例中，单击菜单中“打开”选项时，将弹出一个对话框，显示绑定菜单元素的 ID 号和设置的选项内容，其代码如下：

```
...
bindings:
 {
   'Li1': function(Item) {
      alert(" 在 ID 号为：" + Item.id + " 编辑框中，您点击了 " 新建 " 项 ");
    },
   'Li2': function(Item) {
      alert(" 在 ID 号为：" + Item.id + " 编辑框中，您点击了 " 打开 " 项 ");
    }
   // 设置其他选项事件
   //...
}
...
```

弹出的对话框如图 7-10 所示。

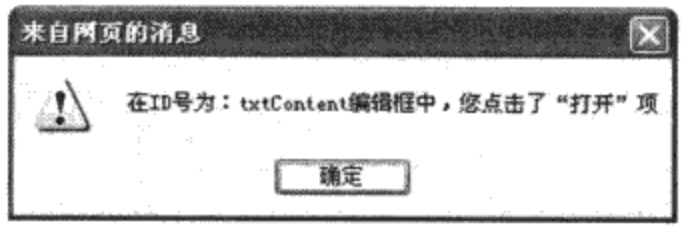


图 7-10　点击菜单中的“打开”选项时，弹出的对话框

在可选项 option 参数中，除了设置绑定菜单的选项外，还可以设置菜单的默认样式，其样式可以设置如下 3 部分，代码如下：

```
menuStyle: {
      // 菜单外框样式
      ...
}
itemStyle: {
      // 菜单选项样式
      ...
}
itemHoverStyle: {
      // 选项选中样式
}
```

在示例 7-7 中，为了改变弹出快捷菜单的样式，加入如下的代码：

```
$(function() {
    $('#txtContent').contextMenu('sysMenu',
       { bindings:{
            ...
          },
       menuStyle:{
            border: '2px solid #999'
          },
       itemStyle:{
            fontFamily: 'verdana',
            backgroundColor: '#666',
            color: 'white',
            border: 'none',
            padding: '1px'
          },
       itemHoverStyle: {
            color: '#666',
            backgroundColor: '#f7f7f7',
            border: 'none'
          }
     });
})
```

弹出菜单样式改变后的页面效果如图 7-11 所示。

为了使读者对 contextmenu 插件能有更加进一步的了解，下面列出该插件在使用时所调用的全部属性和事件，其完整的调用格式如下所示：

```
$(element).contextMenu(Mid
    {
       bindings{// 绑定菜单时，根据 ID 号获取或设置各选项属性和事件
          " 选项 ID 号 ", function(T){
              // 参数 T 为被绑定的元素
              // 获取或设置该选项的属性
```

```
            ...
        }
    },
menuStyle: {
      // 菜单外框样式
      ...
},
itemStyle: {
      // 菜单选项样式
      ...
},
itemHoverStyle: {
      // 选项选中样式
},
Shadow：// 设置一个布尔值，默认为 true，即显示背景阴影
eventPosY: // 指定菜单弹出时，在页面中的 Y 值
eventPosX: // 指定菜单弹出时，在页面中的 X 值
onContextMenu: function(e) {// 菜单内容显示事件
       // 其中通过回调函数中的参数 e，可以获取被绑定菜单的元素，即 e.target；如果该函数返回
          false，将不会弹出快捷菜单
},
onShowMenu: function(e, menu){// 菜单显示事件
       // 其中通过回调函数中的参数 e，可以获取被绑定菜单的元素，即 e.target；参数 menu 是显示的
          菜单，通过该参数可以访问其中的某一个选项，如移除第 1 个选项，代码如下：
          $(' 第 1 个选项 ID 号 ', menu).remove();
          return menu;
    }
)
```

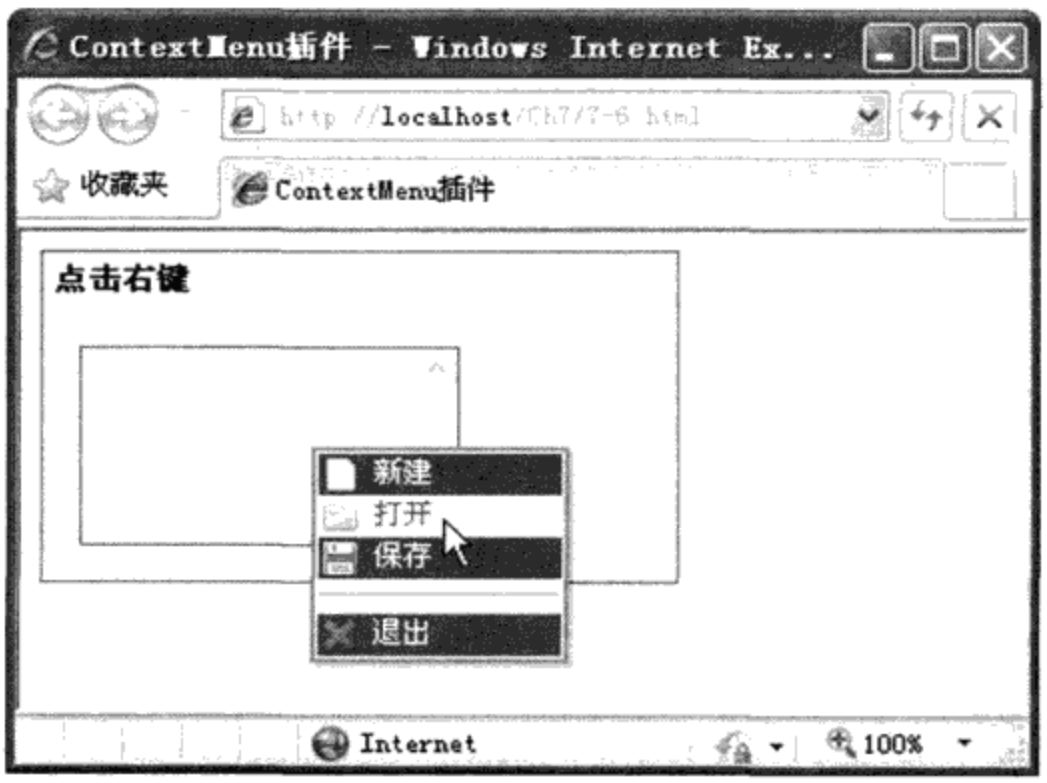


图 7-11　样式改变后的弹出菜单

7.8 图片放大镜插件 jqzoom

Jqzoom 是一款基于 jQuery 库的图片放大插件，在页面中实现放大的方法是：先准备 2 张一大一小的相同图片，在页面打开时，展示小图片，当鼠标在小图片的任意位置移动时，调用插件中的 jqzoom() 方法，绑定另外一张相同的大图片，在指定位置显示与小图片所选区域相同的大图片区域，从而实现逼真的放大效果。该插件非常适合在产品展示时使用。下面举例说明。

示例 7-7 图片放大镜插件的使用

（1）插件文件

Js-7-7/jquery.jqzoom.js

Css-7-7/jquery.jqzoom.css

Images-7-7/ 全部图片

（2）下载地址

http://www.mind-projects.it/projects/jqzoom/archives/jqzoom_ev1.0.1.zip

（3）功能描述

在页面中，放置一张图片，当鼠标在图片中移动时，出现一块选择区域，在图片的右边，显示放大后的所选区域。

（4）实现代码

新建一个 HTML 文件 7-7.html, 加入如代码清单 7-7 所示的代码。

代码清单 7-7 jqzoom 插件实现图片放大效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>JQZoom 放大镜插件 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-7-7/jquery.jqzoom.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css-7-7/jquery.jqzoom.css" />
    <style type="text/css">
           ... 省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#jqzoom").jqzoom( // 绑定图片放大插件 jqzoom
```

```
                {
                    zoomWidth: 230,
                    zoomHeight: 230
                }
            );
        });
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            图片放大镜
        </div>
        <div class="divContent">
            <a href="Images-7-7/bag.jpg" id="jqzoom"
                title="我的背包">
                <img src="Images-7-7/bagsmall.jpg">
            </a>
        </div>
    </div>
</body>
</html>
```

（5）页面效果

代码执行后的效果如图 7-12 所示。

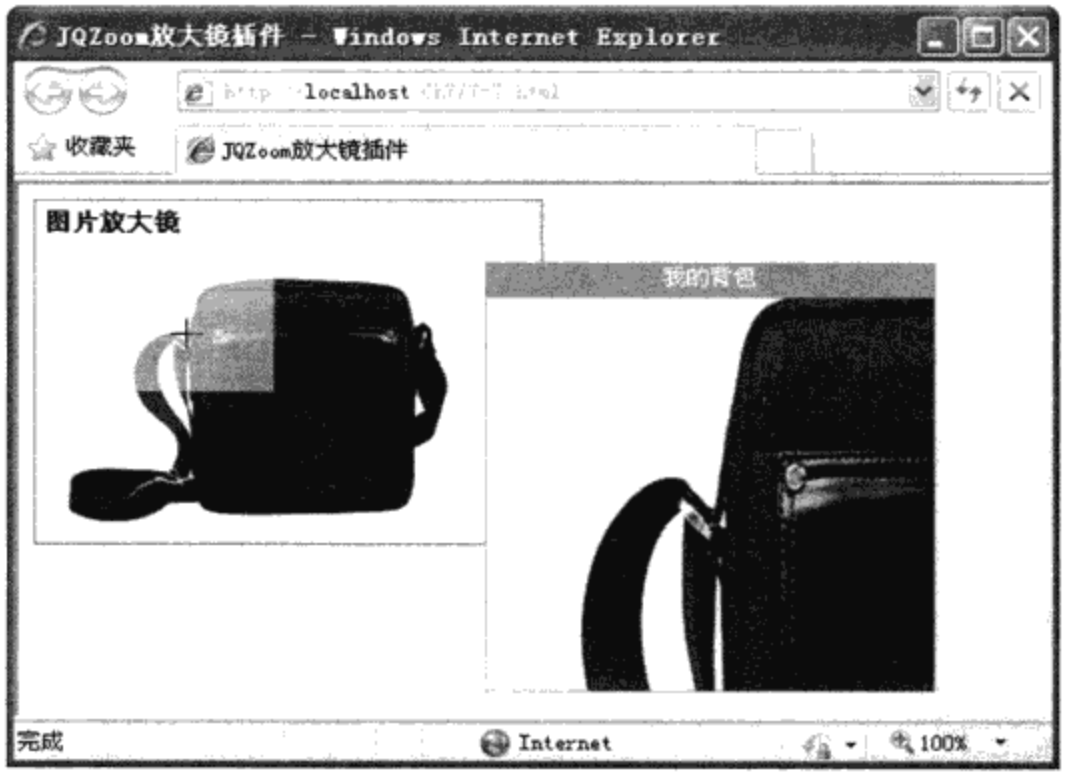


图 7-12　jqzoom 插件实现图片的放大效果

（6）代码说明

在页面中，为了实现图片放大的效果，除引用插件的 JS 文件外，还必须导入与之匹配的

CSS 文件，因此在页面的头文件中，加入如下代码：

```
<script type="text/javascript" src="Js-7-7/jquery.jqzoom.js">
</script>
<link rel="stylesheet" type="text/css"
       href="Css-7-7/jquery.jqzoom.css" />
```

引用完 JS 和 CSS 文件后，接下来的工作就是将图片与插件绑定。为实现这一目的，首先，在页面中增加一个 <img> 标记用于显示小图片，并将图片用一个 <a> 标记包裹起来，同时，将 <a> 标记的 href 属性设置为大图的 URL，并设置 title 属性。其实现的代码如下：

```
<a href="Images-7-7/bag.jpg" id="jqzoom" title=" 我的背包 ">
   <img src="Images-7-7/bagsmall.jpg">
</a>
```

然后，在 JS 文件中通过插件的 jqzoom() 方法，绑定对应的放大元素，其实现的代码如下：

```
$("#jqzoom").jqzoom( // 绑定图片放大插件 jqzoom
   {
     zoomWidth: 230, // 小图片中所选区域的宽
     zoomHeight: 230 // 小图片中所选区域的高
     }
);
```

在插件调用的 jqzoom([Option]) 方法中，可选项参数 Option 是一个对象，除上述的 zoomWidth 与 zoomHeight 属性外，还有如下的常用属性：

```
$(element).jqzoom( // 绑定图片放大插件 jqzoom
   {
    zoomType:// 放大镜类型，默认为 "standard"，如果设为 "reverse"，在小图片中，移动鼠标时，
                所选区域将高亮显示，非选中区域为淡灰色
        xOffset:// 放大后的图片与小图片间的 X( 横坐标 ) 距离
        yOffset:// 放大后的图片与小图片间的 y( 纵坐标 ) 距离
        position:// 放大后的图片相对原图片的位置，默认为 "right"，可设置为 "left","top",
                    "bottom"
    lens:// 一个布尔值，表示是否显示小图片中的选择区域，默认值为 "true"，如果设为 "false"，则
            鼠标在小图片中移动时，不会出现选择区域
    title:// 一个布尔值，是否显示放大图片的主题，默认值为 "true"，如果设为 "false"，则放大后的
             图片上面不会出现主题字样
    imageOpacity:// 当 zoomType 的值为 "reverse" 时，用来设置非选中区域透明度的值 ( 取值范围
                    (0.0-1.0) 间 )
   }
);
```

7.9 自定义 jQuery 插件

虽然有大量优秀的插件可供用户免费下载使用，但开发人员更希望自己能编写根据自身

项目特点使用的插件，提供给团队或他人使用。在 jQuery 中，编写插件也不是一件很难的事，只要编写的代码符合插件的各项功能要求，就可以实现自定义各项目功能的插件，在介绍如何编写自定义的插件前，先来了解插件的基础知识。

7.9.1　插件的种类

从广义上来说，插件分为 3 类，即封装方法插件、封闭函数插件、选择器插件，但最后一种很少人去开发使用，自定义的插件种类多数属于前面两种。

1. 封装方法插件

封装方法插件在本质上来说，是一个对象级别的插件，该类插件首先通过 jQuery 选择器获取对象，并为对象添加方法，然后，将方法进行打包，封装成一个插件，这种类型的插件编写简单，极易调用，也很方便地使用了 jQuery 中功能强大的选择器，因此，成为开发插件的首选。

2. 封闭函数插件

封闭函数插件是一个类级别的插件，该类插件最大的特点，就是可以直接给 jQuery 添加静态方法，并且可以将函数置于 jQuery 命名空间中，如最为常见的就是 $.ajax()、$.trim() 全局性函数，都是以内部插件的形式植入 jQuery 内核中。

7.9.2　插件开发要点

在了解完插件的种类后，接下来我们介绍插件所具有的特点，这些特点也是我们在开发插件时必须注意的事项。

1）插件的文件命名必须严格遵循 jQuery.[插件名].js 的规则，以便于与其他 js 文件的区分，如新插件文件 jquery.newplugin.js。

2）如果是对象级别插件，所有的方法都应依附于 jquery.fn 主体对象；如果是类级别插件，所有的方法都应依附于 jquery 对象。

3）无论是对象级别还是类级别插件，结尾都必须以分号结束，否则，在文件被压缩时，会出现错误提示信息。

4）在插件内部的代码中，如果要访问每个元素，可以使用 this.each 方法来遍历全部元素。

5）需要说明的是在插件的内部，this 所代表的是通过 jQuery 选择器所获取的对象，而非传统意义上的对象的引用。

6）由于 jQuery 代码在调用方法时，可以采用链写的方法同时调用多个方法，因此，为了保证这个功能的实现，插件本身必须返回一个 jQuery 对象。

7）虽然“$”美元符，可以与“jQuery”字符相代替，但在编写插件的代码中，尽量不要使用“$”符号，以避免与别的代码冲突。

8）在编写对象级别的插件时，使用 jQuery.fn.extend() 方法进行功能扩展；而针对类级别

的插件，则使用 jQuery.extend() 方法进行扩展。

7.9.3 开发插件示例

示例 7-8 对象级别插件的开发

（1）功能描述

在列表 <ul> 元素中，鼠标在表项 <li> 元素移动时，可以自定义其获取焦点 (focus) 时的背景颜色，即设置表项 <li> 元素选中时的背景色。

（2）搭建框架

新建一个 js 文件，命名为 jquery.lifocuscolor.js，并在文件中使用 $.fn.extend() 方法完成框架的搭建。其实现的代码如下：

```
/*-------------------------------------------------------------*/
功能：设置列表中表项获取鼠标焦点时的背景色
参数：li_col【可选】鼠标所在表项行的背景色
返回：原调用对象
示例：$("ul").focusColor("red");
/-------------------------------------------------------------*/
;(function($){
   $.fn.extend({
         "yourPluginName": function(pram_value){
         // 各种默认属性或参数的设置
         this.each(function(){
          // 插件实现的代码
          })
       }
    })
})(jQuery);
```

（3）代码编写

根据功能描述，在搭建的框架中，首先，设置插件的默认属性值，由于允许用户设置自己的颜色值，因此，创建一个颜色参数，并对该值进行初始化设置，同时，根据参数是否为空，赋予该参数不同的颜色值；另外，设置一个变量，保存丢失焦点时的颜色值，其实现的代码如下所示：

```
;(function($) {
    $.fn.extend({
      "focusColor": function(li_col) {
         var def_col = "#ccc"; // 默认获取焦点的色值
         var lst_col = "#fff"; // 默认丢失焦点的色值
         // 如果设置的颜色不为空，使用设置的颜色，否则为默认色
         li_col = (li_col == undefined) ? def_col : li_col;
         // 遍历表项 <li> 中的全部元素
         $(this).find("li").each(function() {
                    ...
```

```
                    });
                }
            });
        })(jQuery);
```

然后，在遍历表项 <li> 中，需针对象编写两个事件，一个是鼠标移入事件 mouseover()，在该事件中，将传回的变量 def_col 值，设置为对象的背景色，其代码如下：

```
$(this).find("li").each(function() {
    $(this).mouseover(function() { // 获取鼠标焦点事件
        $(this).css("background-color", li_col); // 使用设置的颜色
    })
})
```

最后，编写另外一个鼠标移出事件 mouseout()，在该事件中，将背景色还原成鼠标移入前，被变量 lst_col 保存的颜色值，其代码如下：

```
$(this).find("li").each(function() {
    $(this).mouseout(function() { // 鼠标焦点移出事件
        $(this).css("background-color", lst_col); // 恢复原来的颜色
    })
})
```

由于这两个事件可以进行合并链写，因此，最终的代码如下：

```
$(this).find("li").each(function() { // 遍历表项 <li> 中的全部元素
    $(this).mouseover(function() { // 获取鼠标焦点事件
        $(this).css("background-color", li_col); // 使用设置的颜色
      }).mouseout(function() { // 鼠标焦点移出事件
        $(this).css("background-color", "#fff"); // 恢复原来的颜色
      })
})
```

在代码最后结果时，必须返回一个 jQuery 对象，以方便其调用对象的链写操作，因此，最后加上一行返回 jQuery 对象的如下代码：

```
return $(this); // 返回 jQuery 对象，保持链式操作
```

经过上述分析，js 文件 jquery.lifocuscolor.js 最终完整的代码如下所示：

```
;(function($) {
    $.fn.extend({
    "focusColor": function(li_col) {
      var def_col = "#ccc"; // 默认获取焦点的色值
      var lst_col = "#fff"; // 默认丢失焦点的色值

      // 如果设置的颜色不为空，使用设置的颜色，否则为默认色
      li_col = (li_col == undefined) ? def_col : li_col;
      // 遍历表项 <li> 中的全部元素
      $(this).find("li").each(function() {
         $(this).mouseover(function() { // 获取鼠标焦点事件
```

```
            $(this).css("background-color", li_col); // 使用设置的颜色
                  })
                .mouseout(function() { // 鼠标焦点移出事件
            $(this).css("background-color", "#fff"); // 恢复原来的颜色
                  })
            });
            return $(this); // 返回 jQuery 对象，保持链式操作
        }
    });
})(jQuery);
```

(4) 引用插件

自己编写的插件保存为 js 文件后，就可以像其他插件一样，被需要使用的文件所调用，其使用的方法也是一样，先引入 js 文件，然后，在 js 代码中调用该插件中的各种方法。为了验证文件 jquery.lifocuscolor.js 中插件功能，新创建一个 html 文件 pl_LiGetFocus.html，引入该插件文件，并调用该插件的方法，其加入的代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>编写一个对象级别的插件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Plugin/jquery.lifocuscolor.js">
    </script>
    <style type="text/css">
            ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#u1").focusColor();// 调用自定义的插件
            //$("#u1").focusColor("red");// 调用自定义的插件
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            对象级别的插件
        </div>
        <div class="divContent">
            <ul id="u1">
                <li><span>张三</span><span>男</span></li>
                <li><span>李四</span><span>女</span></li>
                <li><span>王五</span><span>男</span></li>
```

```
            </ul>
        </div>
    </div>
</body>
</html>
```

（5）页面效果

执行 html 文件 pl_LiGetFocus.html 后，其实现的页面效果如图 7-13 所示。

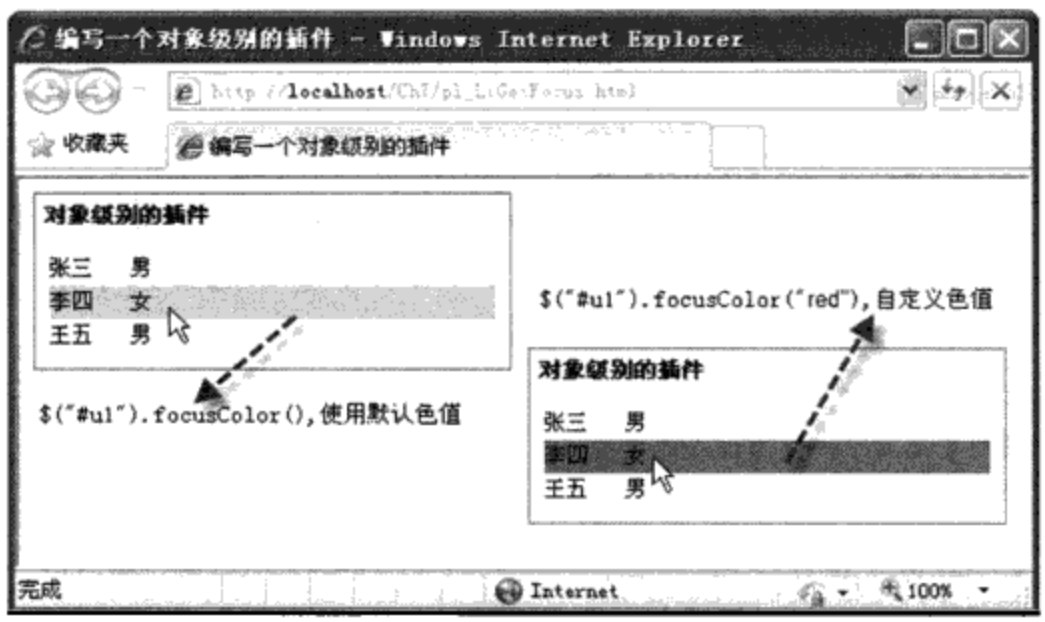


图 7-13　引用自定义的对象级别插件

示例 7-9　类级别插件的开发

（1）功能描述

新增两个类级别的全局函数，分别用于计算两数之和与两数之差，并将结果返回调用的页面中。

（2）搭建框架

新建一个 js 文件，命名为 jquery.twoaddresult.js，并在文件中，使用 $.extend() 方法完成框架的搭建，其实现的代码如下：

```
/*-------------------------------------------------------------------/
功能：计算二个数字相加或相减的结果
参数：数字 p1,p2
返回：两数相加后的结果
示例：$.AddNum(1,2);
/-------------------------------------------------------------------*/
; (function($){
    $.extend({
        "yourPluginName": function(pram_value){
          // 插件实现的代码
        }
    });
})(jQuery);
```

（3）代码编写

根据功能描述，编写一个用于计算两数之和的全局函数，该函数先对用户传来的两个参数进行检测，判断其是否为 undefined 类型，以获取其最终用于计算的值，然后，通过 return 语句，返回其最终的计算结果，其实现的代码如下：

```
"addNum": function(p1, p2) {
   //如果传入的数字不为空，使用传入的数字，否则为 0
   p1 = (p1 == undefined) ? 0 : p1;
   p2 = (p2 == undefined) ? 0 : p2;
   var intResult = parseInt(p1) + parseInt(p2);
   return intResult;
 }
```

然后，使用同样的方式，增加一个用户计算两数之差的全局函数，由于最后返回的一个两数相减的结果，因此，在进行基本检测后，判断第一个参数是否大于第二个参数，如果大于，则返回前者减去后者的结果，其实现的代码如下：

```
"subNum": function(p1, p2) {
   //如果传入的数字不为空，使用传入的数字，否则为 0
   var intResult = 0;
   p1 = (p1 == undefined) ? 0 : p1;
   p2 = (p2 == undefined) ? 0 : p2;
   if (p1 > p2) {//如果传入的参数前者大于后者
       intResult = parseInt(p1) - parseInt(p2);
   }
   return intResult;
}
```

最后，使用 jQuery.extend() 方法，直接对 jQuery 对象进行拓展，以扩充其全局函数，其最终实现的完整代码如下：

```
; (function($) {
    $.extend({
        "addNum": function(p1, p2) {
            //如果传入的数字不为空，使用传入的数字，否则为 0
            p1 = (p1 == undefined) ? 0 : p1;
            p2 = (p2 == undefined) ? 0 : p2;
            var intResult = parseInt(p1) + parseInt(p2);
            return intResult;
        },
        "subNum": function(p1, p2) {
            //如果传入的数字不为空，使用传入的数字，否则为 0
            var intResult = 0;
            p1 = (p1 == undefined) ? 0 : p1;
            p2 = (p2 == undefined) ? 0 : p2;
            if (p1 > p2) { //如果传入的参数前者大于后者
                intResult = parseInt(p1) - parseInt(p2);
        }
```

```
            return intResult;
        }
    });
})(jQuery);
```

（4）引用插件

与引用对象级别插件一样，类级别的插件也是先在 <title> 标记中导入插件的 js 文件，然后，编写 js 代码，调用插件中的公用方法或函数。

为检测插件的功能，新建一个 HTML 文件 pl_TwoCalculate.html，在页面中，单击上下两个“等于”按钮后，分别调用插件中的全局函数 addNum 与 subNum，计算文本框中的两数之和与两数之差。其实现的页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>编写一个类级别的插件</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Plugin/jquery.twoaddresult.js">
    </script>
    <style type="text/css">
        ...省略样式部分代码
    </style>
    <script type="text/javascript">
        $(function() {
            $("#Button1").click(function() {
                $("#Text3").val(
                $.addNum($("#Text1").val(),
                        $("#Text2").val()));
            }); // 调用自定义的插件计算两数之和
            $("#Button2").click(function() {
                $("#Text6").val(
                $.subNum($("#Text4").val(),
                        $("#Text5").val()));
            }) // 调用自定义的插件计算两数之差
        })
    </script>
</head>
<body>
    <div class="divFrame">
        <div class="divTitle">
            类级别的插件
        </div>
        <div class="divContent">两数相加：
```

```
            <input id="Text1" type="text" class="txt" />
            +
            <input id="Text2" type="text" class="txt" />
            =
            <input id="Text3" type="text" class="txt" />
            <input id="Button1" type="button"
                   value=" 等于 " class="btn" />
            <hr /> 两数相减：
            <input id="Text4" type="text" class="txt" />
            - 
            <input id="Text5" type="text" class="txt" />
            =
            <input id="Text6" type="text" class="txt" />
            <input id="Button2" type="button"
                   value=" 等于 " class="btn"/>
        </div>
    </div>
    </body>
</html>
```

（5）页面效果

执行 html 文件 pl_TwoCalculate.html 后，其实现的页面效果如图 7-14 所示。

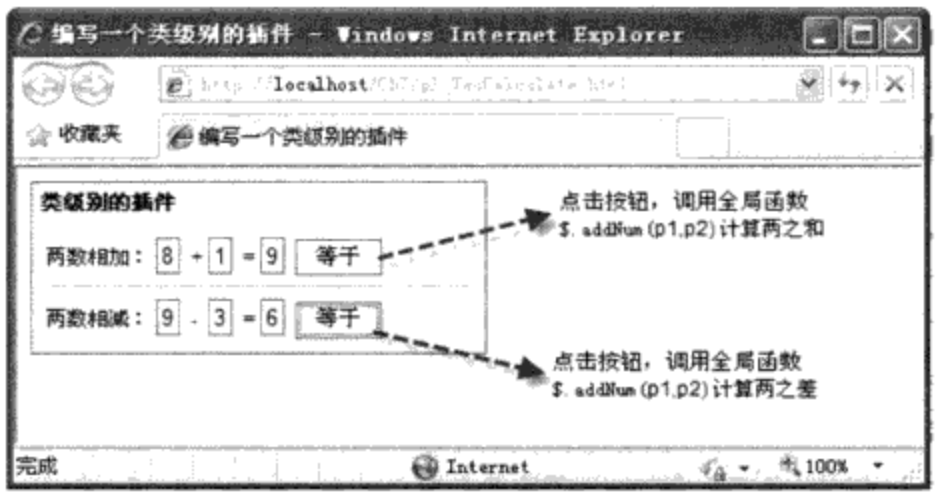


图 7-14　引用自定义的类级别插件

说明　无论是编写对象级别插件，还是编写类级别插件，都要严格遵守插件开发的要素，先搭框架，后进行开发，不同的插件使用不同的拓展方法。

7.10　综合案例分析——使用 uploadify 插件实现文件上传功能

7.10.1　需求分析

经分析，该案例的需求如下：

1）可以选择上传文件类型、限制上传文件大小，实现单个或多个文件同时上传功能，在

上传过程中显示文件上传进度比例信息。

2）文件成功上传后，如果是图片文件，则可以进行图片预览功能。

3）无论是单个或多个文件成功上传后，可删除其队列中的某一个文件。

7.10.2　效果界面

该案例的效果界面如下所示：

1）选择单个或多个文件上传时，显示带百分比的进度条，提示文件上传进度信息，实现的效果如图 7-15 所示。

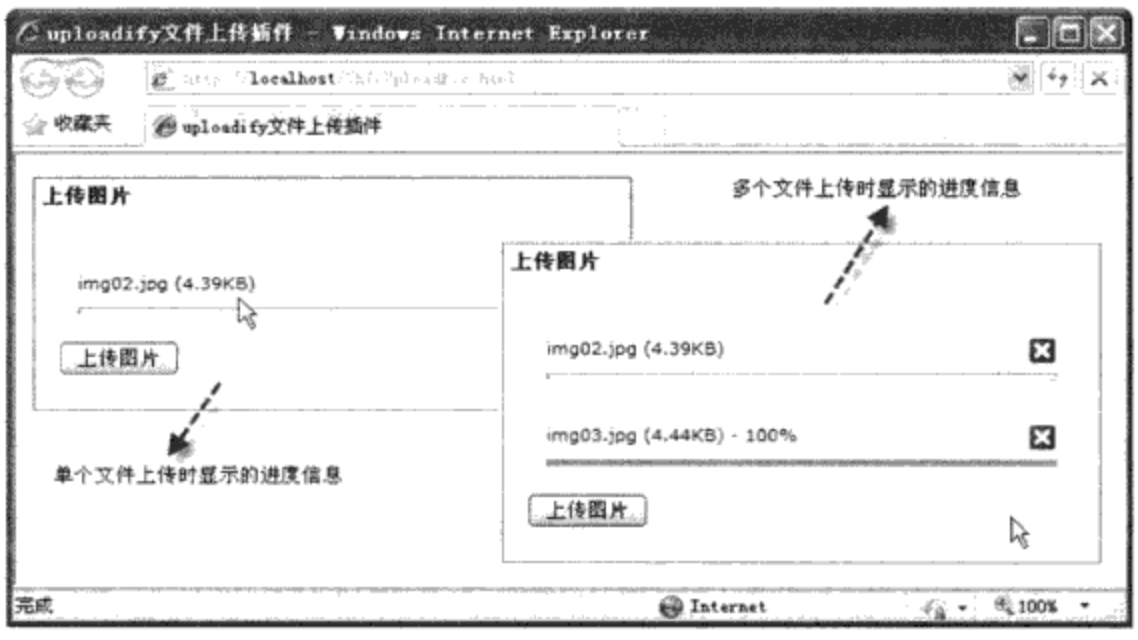


图 7-15　单个或多个文件上传时，提示的进度信息

2）文件上传成功后，通过回调函数，可以返回上传文件的 URL，根据这个 URL 地址，如果是图片，则可以实现图片预览的功能，其实现的页面效果如图 7-16 所示。

图 7-16　图片文件上传成功后的预览效果

3）在文件上传成功后，预览图片时，选择某个文件，单击“删除”链接，则可以删除指

定的某个上传后的文件，其实现的页面效果如图 7-17 所示。

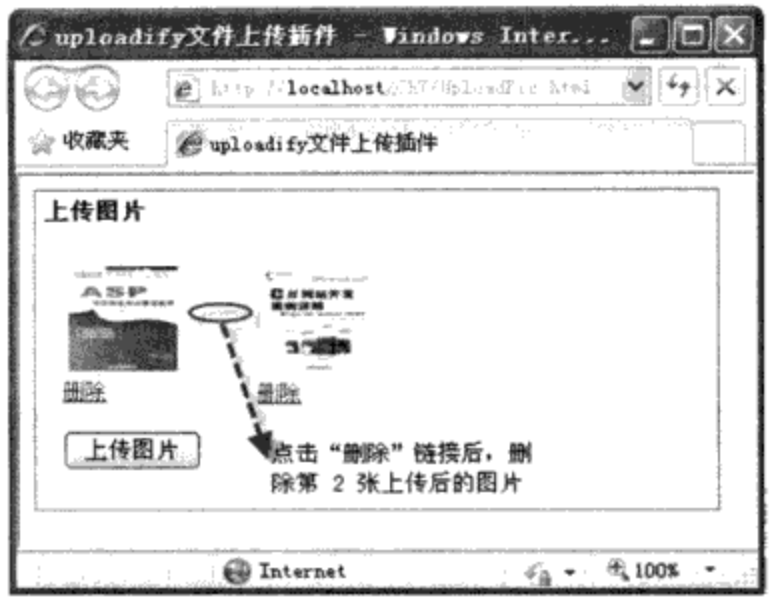


图 7-17　删除指定图片文件后的页面效果

7.10.3　功能实现

1. 插件介绍

在本案例中，使用了基于 jQuery 库开发的上传文件插件 Uploadify。该插件是众多插件中较为优秀的一款，支持多文件自动上传，带上传进度条和百分比例，并支持多种函数触发事件，如函数 onComplete（完成后）、onError（出错时）等，该插件广泛应用于各项目中，深受开发者的喜爱。

2. 插件下载

插件文件：Js/jquery.uploadify.v2.1.0.js
　　Js/swfobject.js
　　Images/ 全部文件
　　Css/uploadify.css

下载地址：http://www.uploadify.com/download/

3. 插件使用

上传文件插件 uploadify 下载完成后，其使用方法也十分简单。下面介绍其使用的步骤。

第一步：在调用文件中引用两个 js 文件，一个是 jquery.uploadify.v2.1.0.js，用于文件的上传，另一个是 swfobject.js 用于设置选择上传文件的按钮。其实现的代码如下：

```
<script type="text/javascript" src="Js/swfobject.js"></script>
<script type="text/javascript"
          src="Js/jquery.uploadify.v2.1.0.js">
</script>
```

第二步：在引用插件的页面中，编写 js 代码，设置 uploadify 插件上传文件时必须使用的

属性，如保存上传文件的地址、处理文件上传的服务器端文件名等。其实现的代码如下所示：

```
$(function() {
    $("#uploadify").uploadify({
        'uploader': 'Images/uploadify.swf',
        'script': 'Deal/UploadFile.ashx',
        'cancelImg': 'Images/cancel.png',
        'folder': 'Uploads/',
        'queueID': 'fileQueue',
        'buttonImg': 'Images/uploadify.jpg',
        'auto': true,
        'multi': true,
    })
})
```

在上述 js 代码中，为了不使用插件默认的选择上传文件的按钮，我们调用了一个“buttonImg”属性，并通过该属性设置自己定义的按钮地址。为确保文件的正确上传，其他属性所指定的文件路径或 ID 号必须准确无误。

第三步：当插件的 JS 文件被引用并且相关的基本属性设置完成之后，在引用的页面文件中加入如下的 HTML 代码，用于实现上传文件的选择和上传队列进度信息的展示。其代码如下：

```
<div id="fileQueue"></div>
<input type="file" name="uploadify" id="uploadify" />
```

其中 标记的 ID 号与 js 代码中“queueID”属性的值相同的，该属性表示上传文件对列的 ID 号，即 ID 号为“fileQueue”的 标记用于存放上传文件的队列，并通过下列代码：

```
$("#uploadify").uploadify({...})
```

使页面中 ID 号为“uploadify”的文件上传元素与插件中的 uploadify() 方法相绑定。

第四步：编写处理文件上传的服务器端代码，要实现文件的真正上传，还需要编写一个简单的服务器端文件，接受传来的文件队列，逐个保存在服务器中。在本案例中，通过一个程序处理文件 UploadFile.ashx 实现上述的功能。其实现的代码如下：

```
<%@ WebHandler Language="C#" Class="UploadFile" %>
using System;
using System.IO;
using System.Web;
public class UploadFile : IHttpHandler {
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "text/plain";
        context.Response.Charset = "utf-8";
        HttpPostedFile file = context.Request.Files["Filedata"];
        // 获取传回的保存文件地址
        string uploadPath = HttpContext.Current
        .Server.MapPath(@context.Request["folder"]) + "\\";
```

```
            //如果有文件
            if (file != null)
            {
                if (!Directory.Exists(uploadPath))//目录不存在
                {
                    Directory.CreateDirectory(uploadPath);//新创建目录
                }
                file.SaveAs(uploadPath + file.FileName);//逐个保存文件
                //返回一个值，确保上传成功后，在前台的文件队列自动消失
                context.Response.Write("1");
            }
            else
            {
                context.Response.Write("0");
            }
        }
        ...
    }
```

7.10.4 代码分析

经过上述 4 个步骤，就可以在一个 HTML 页面中，通过引用 uploadify 插件中的方法，实现一个或多个文件的上传，完成了需求中的第一项功能。

在通过 uploadify 插件实现文件上传过程中，有一个 onComplete() 函数，该函数在每个文件上传成功后便触发，并且在触发时传回 5 个参数，分别为 event、queueId、fileObj、response、data，其各自代码的意义如下：

event：表示触发事件的对象。

queueId：上传文件中队列的标识，用于区分每一个上传文件，它由六位随机数组成。

fileObj：选择上传的文件对象，通过该对象，可以访问上传文件的相关属性，如 name（名称）、size（大小）、creationDate（创建时间）、modificationDate（修改时间）、type（文件类型）、filePath（保存路径）。

response：服务器端文件上传处理程序返回的值，即文件 UploadFile.ashx 返回值，在本案例中，文件上传成功后，返回值为 1。

data：表示文件上传数据流，该对象有二个重要属性，一个是 fileCount，表示还没有上传完的文件总量，另一个是 speed 表示文件的平均上传速度。

根据上面对 onComplete() 函数中各个返回参数的描述，可以通过第三个参数 fileObj 中的 filePath 属性值，返回上传文件后的 URL，如果是图片上传，则将返回的图片 URL 设置为客户端中 <img> 标记的 src 属性值，从而实现图片上传后预览的功能。其实现的 JS 代码如下：

```
function SetFileContent(objFile) { //根据上传对象返回预览的图片
    var sLi = "";
    sLi += "<li>";
    sLi += "<img src='" + objFile.filePath + "' width='100'
height='100'>";
```

```
    sLi += "<input type='hidden' value='" + objFile.filePath + "'>";
    sLi += "<br />";
    sLi += "<a href='javascript:void(0)'>删除</a>";
    sLi += "</li>";
    return sLi;
}
```

其中，自定义的函数 SetFileContent 中的参数 objFile 就是 onComplete() 函数中的第三个参数 fileObj，将上述函数 SetFileContent 放入 onComplete() 事件中，即实现了图片文件上传成功后可以进行预览的功能，从而完成了第二项需求。其实现的代码如下：

```
$(function() {
$("#uploadify").uploadify({
   ...
'onComplete': function(event, queueID, fileObj, response, data) {
            $('ul').append(SetFileContent(fileObj));
     }
})
})
```

当然，为了在页面中展示预览的图片，需要加入一个列表 <ul> 标记。其页面代码如下：

```
<ul></ul>
<div id="fileQueue"></div>
<input type="file" name="uploadify" id="uploadify" />
```

为了在图片上传成功后进行预览时，删除不太理想的图片文件，首先，需要确定被删除表项 <li> 的 ID 号。然后，使用 jQuery 中的 remove() 方法移除指定的表项。最后，在增加新表项内容时，重新分配表项的 ID 号，从而保证 ID 号的不重复。

根据上面的分析，首先，通过 each 方法遍历全部的表项 ，并设置对应的 ID 号。其代码如下：

```
// 设置各表项 ID 号
$("ul li").each(function(l_i) {
      $(this).attr("id", "li_" + l_i);
})
```

然后，通过 each 方法遍历全部表项中“删除”链接，设置对应的“rel”属性，用于传递删除表项的 ID 号。其代码如下：

```
// 设置各链接的 rel 属性值
$("ul li a").each(function(a_i) {
      $(this).attr("rel", a_i);
})
```

最后，根据表项中“删除”链接中的“rel”属性，设置删除某个表项的单击事件。其实现的代码如下：

```
//设置各链接的单击事件
$("ul li a").click(function() {
      //通过自身的 rel 值，获取表项的 ID 号
      var li_id = "#li_" + this.rel;
      //根据 ID 号，删除某个表项
      $(li_id).remove();
 })
```

由于上述的三项操作都在同一函数中触发，并且可以进行链写，因此，将其封装到另外一个自定义的函数 SetUploadFile 中。其最后完整的代码如下：

```
function SetUploadFile() {
    //设置各表项 ID 号
    $("ul li").each(function(l_i) {
        $(this).attr("id", "li_" + l_i);
    })

    //设置各链接的 rel 属性值
    $("ul li a").each(function(a_i) {
        $(this).attr("rel", a_i);
    }).click(function() {//设置各链接的单击事件
        //通过自身的 rel 值，获取表项的 ID 号
        var li_id = "#li_" + this.rel;
        //根据 ID 号，删除某个表项
        $(li_id).remove();
    })
}
```

最后，将上述自定义的函数 SetUploadFile 放入 onComplete() 函数中，即实现了在预览上传图片时，可以进行删除的操作，从而完成了第三项需求。其实现的代码如下：

```
$(function() {
  $("#uploadify").uploadify({
        ...
  'onComplete': function(event, queueID, fileObj, response, data) {
            $('ul').append(SetFileContent(fileObj));
             SetUploadFile();
     }
  })
})
```

为了更好地实现页面与代码的分离，在本案例中，创建一个 HTML 页面文件 UploadPic.html，用于用户选择上传的文件。其完整的代码下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>uploadify 文件上传插件 </title>
```

```
    <script type="text/javascript"
            src="Js/fileUpload.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css/uploadify.css" />
    <style type="text/css">

    </style>
</head>
<body>
   <div class="divFrame">
        <div class="divTitle">
              上传图片
        </div>
        <div class="divContent">
             <ul></ul>
             <div id="fileQueue"
                  style="clear:both;padding-top:5px"></div>
             <div style="padding-top:5px">
                  <input type="file" name="uploadify"
                          id="uploadify" />
             </div>
        </div>
   </div>
</body>
</html>
```

另外，新建一个 js 文件 fileUpload.js，用于实现页面中的文件上传和预览，同时，删除上传后的文件。其完整代码如下：

```
/// <reference path="../Jscript/jquery-1.4.2-vsdoc.js"/>
/// <reference path="../Jscript/jquery-1.4.2.js"/>
/// <reference path="../Js/swfobject.js"/>
/// <reference path="../Js/jquery.uploadify.v2.1.0.js"/>
$(function() {
    $("#uploadify").uploadify({
        'uploader': 'Images/uploadify.swf',
        'script': 'Deal/UploadFile.ashx',
        'cancelImg': 'Images/cancel.png',
        'folder': 'Uploads/',
        'queueID': 'fileQueue',
        'buttonImg': 'Images/uploadify.jpg',
        'auto': true,
        'multi': true,
        'fileExt': '*.jpg;*.jpeg;*.gif;*.png',
        'onComplete': function(event, queueID,
                               fileObj, response, data) {
          $('ul').append(SetFileContent(fileObj));
          SetUploadFile();
        }
```

```
    })
})
function SetFileContent(objFile) { //根据上传对象返回预览的图片
    var sLi = "";
    sLi += "<li>";
    sLi += "<img src='" + objFile.filePath + "' width='100'
 height='100'>";
    sLi += "<input type='hidden' value='" + objFile.filePath + "'>";
    sLi += "<br />";
    sLi += "<a href='javascript:void(0)'>删除</a>";
    sLi += "</li>";
    return sLi;
}
function SetUploadFile() {
    //设置各表项ID号
    $("ul li").each(function(l_i) {
        $(this).attr("id", "li_" + l_i);
    })
    //设置各链接的rel属性值
    $("ul li a").each(function(a_i) {
        $(this).attr("rel", a_i);
    }).click(function() {//设置各链接的单击事件
        //通过自身的rel值，获取表项的ID号
        var li_id = "#li_" + this.rel;
        //根据ID号，删除某个表项
        $(li_id).remove();
    })
}
```

在本案例中，使用了 uploadify 插件的一些属性和函数，该插件的核心方法是 uploadify()，其调用的格式如下：

```
$(上传文件元素).uploadify(options)
```

其中参数 options 是一个对象，它所包含的常用属性见表 7-1。

表 7-1　options 对象包含的常用属性

名　称	说　明
uploader	上传控件的主体文件，一个 flash 控件，默认值为“uploadify.swf”
script	相对路径的服务器端文件名，它将处理上传的文件，绝对路径前缀或“/”或“http”的路径，默认值为 "uploadify.php"
folder	上传文件保存在服务器端的路径
queueID	上传文件的队列 ID 号，与客户端页的 ID 号相同
queueSizeLimit	设置在一次队列中的文件总数，单击“浏览”按钮后可以同时选择多少个文件
multi	是否允许同时上传多文件，可设定 true 或 false。默认值为 false
auto	选定文件后是否自动上传，可设定 true 或 false。默认值为 false
fileDesc	出现在上传对话框中，对文件类型描述，必须与 fileExt 结合使用，才有效果

（续）

名　称	说　明
fileExt	上传文件所支持的格式，启用本项时需同时声明 fileDesc。如："*.jpg、*.bmp、*gif"
sizeLimit	设置上传文件的最大值，单位为字节
simUploadLimit	在多个文件上传时，设置同时上传文件的数目。默认值为 1
buttonText	默认选择上传文件按钮的文字。默认值为 BROWER
buttonImg	默认选择上传文件按钮的图片地址
onInit	函数：用于检测上传文件时的初始状态，如下列代码： onInit: function() { $("#id").html(" 正在初始化 ...");}
onComplete	函数：用于文件上传成功后触发，可传回 5 个参数，其详细使用见本案例
onSelect	函数：用于选择文件后触发，可传回 3 个参数 event、queueID、fileObj，各参数代表的意义与 onComplete 函数一样
onError	函数：用于文件上传出错后触发，可传回 4 个参数 event、queueID、fileObj、rrorObj，其中前三个参数与 onComplete 函数一样，errorObj 参数有二个属性，一个是 type，表示错误类型，有三种“HTTP”、“IO”、“Security”；另一个参数是 info，表示出错信息的描述

7.11　本章小结

本章先从 jQuery 中插件的下载讲起，然后，通过示例与插件文件相结合的方式，详细介绍了目前比较常用插件的使用方法，然后，介绍如何自定义编写对象和类级别插件，再通过一个完整的案例介绍最为常用的上传文件插件 uploadify 的详细使用方法。由于插件在 jQuery 中占据重要位置，下一章节我们将继续介绍另外一种全新的 jQuery UI 插件。

第 8 章

jQuery UI 插件

本章内容

- 认识 jQuery UI
- jQuery UI 交互性插件
- jQuery UI 微型插件
- 综合案例分析——使用 jQuery UI 插件以拖动方式管理相册
- 本章小结

jQuery UI 是一个以 jQuery 为基础的用户体验代码库，它的本质源于一个名为 interface 的 jQuery 插件，后来对该插件内部的 API 进行重构，并升级了版本，重新取名为 jQuery UI。由于 jQuery 库注重于后台，没有很好的前台界面，而 jQuery UI 则补充了其不足之处，两者相互交织。随着 jQuery 技术的发展壮大，也同时确定了 jQuery UI 作为 jQuery 官方插件的地位。

8.1　认识 jQuery UI

jQuery UI 则重于用户界面的体验，根据其体验角度的不同，主要分为以下三个部分：

1）交互（Interactions）在该部分中，展示一些与鼠标操作相关的插件内容，如拖动（Draggable）、放置（Droppable）、缩放（Resizable）、复选（Selectable）、排序（Sortable）等。

2）微件（Widgets）该部分包括一些可视化的细小控件，通过这些小控件，可以极大优化用户在页面中的体验度，如折叠面板（Accordion）、日历（Datepicker）、对话框（Dialog）、进度条（Progressbar）、滑动模块（Slider）、标签页（Tabs）等。

3）效果或动画（Effects）该部分包括一些动画效果插件，使我们的动画不再拘泥于 animate() 方法，可以通过该部分的插件，实现一些复杂的动画效果。在该部分中，改进后的动画方法如 show()、hide()、toggle 等。

目前，jQuery UI 的最新版本可以在地址 http://jqueryui.com/download 的页面中进行下载，其下载界面如图 8-1 所示。

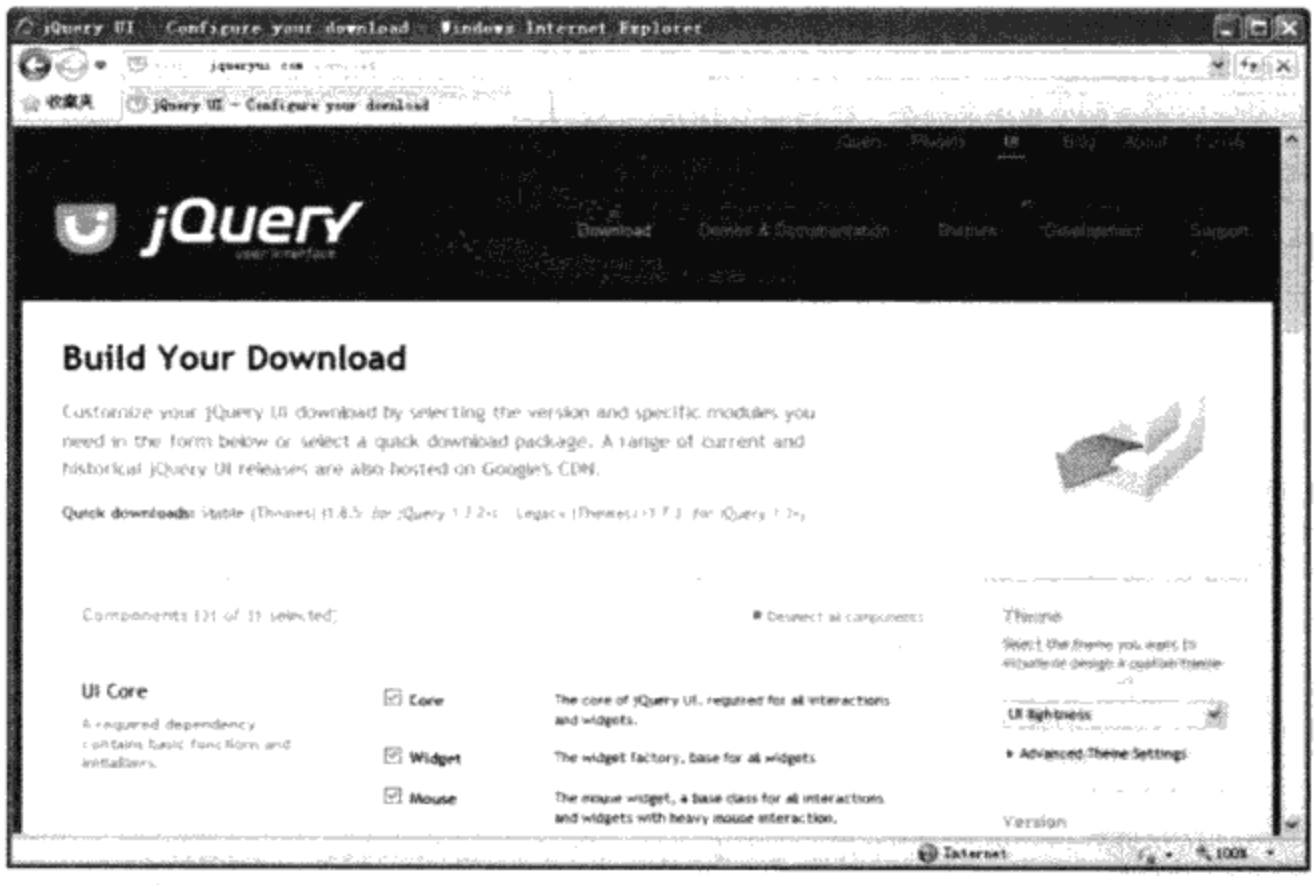


图 8-1　jQuery UI 官方下载页面

在该页面中的右侧，首先，在版本（Version）处，选中单选按钮 1.8.5，然后，单击

“Download”按钮，就可以下载最新的 jQuery UI 库 jquery-ui-1.8.5.custom.rar 压缩包文件。在该压缩包中，包括最新的 jQuery UI 库文件和 demos、themes、docs 等文件包，用户可以选择性解压缩。

本章所介绍的示例基于 jQuery UI 版本 1.8.5 和 jQuery 版本 1.4.2。

8.2 jQuery UI 交互性插件

8.2.1 拖曳插件

draggable（拖曳）插件能使请求的对象拖动，通过这个插件，可以使用 DOM 元素跟随鼠标进行移动，通过设置方法中的 option 选项，实现各种各样的拖动需求，其调用的语法格式为：

```
draggable(options)
```

其中选项 options 接受各种各样的参数值，用于控制拖动时的页面效果。其常用的参数值如表 8-1 所示。

表 8-1 选项 options 可接受的常用参数

参数名称	说　明
helper	表示被拖曳的对象，默认值为 original，即拖动自身；如果设置为 clone，那么，以复制的形式进行拖动
handle	表示触发拖曳的对象，常用一个 DOM 元素
dragPrevention	设置不触发拖曳的元素
start	当拖曳启动时触发的回调函数 function(e,ui)，其中参数 e 表示 event 事件，e.target 表示被拖曳的对象；参数 ui 表示与拖曳相关的对象
stop	停止拖曳是触发的回调函数，其中的参数说明与 start 中一样
drag	在拖动过程中触发的回调函数，其中的参数说明与 start 中一样
zIndex	设置被拖曳时，helper 对象的 z-index 值
axis	设置拖曳时的坐标，可以设为“x”或“y”值
containment	设置拖曳时的区域，可以设为“document”、“parent”和其他指定的元素和对象
grid	设置拖曳时的步长，如 grid:[50,60]，表示 x 坐标每次移动 50px，y 坐标每次移动 60px
opacity	设置对象在拖曳过程中的透明度，范围是（0.0~1.0）
revert	设置一个布尔值，如果为 true，则表示对象被拖曳结束后，又会自动返回原地，如果为 false，则不会返回原地，默认值为 false
scroll	设置一个布尔值，如果为 true，则表示对象在拖曳时，容器自动滚动，默认为 true 值
disable	临时性禁用拖动功能
enable	重新开启对象的拖动功能
destroy	彻底移除对象上的拖动功能

下面通过一个简单的示例介绍 DOM 元素通过 draggable 插件实现拖曳的过程。

示例 8-1　使用 draggable 插件实现对象的拖曳操作

（1）插件文件

Js-Pub/ jquery.ui.core.js

Js-Pub/ jquery.ui.widget.js

Js-Pub/ jquery.ui.mouse.js

Js-8-1/ jquery.ui.draggable.js

（2）功能描述

在页面中，创建三个可供拖曳的 块，分别实现透明度为 0.35 的任意拖曳和在 X 轴中任意拖曳及在父窗口中任意拖曳的操作后，返回原来的初始位置。

（3）实现代码

新建一个 HTML 文件 8-1.html，加入如代码清单 8-1 所示的代码。

代码清单 8-1　使用 draggable 插件实现对象的拖曳操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 draggable 插件实现对象的拖曳操作 </title>
    <script type="text/javascript"
        src="Jscript/jquery-1.4.2.js">
</script>
<script type="text/javascript"
        src="Jscript/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-8-1/jquery.ui.widget.js">
</script>
<script type="text/javascript"
        src="Js-8-1/jquery.ui.mouse.js">
</script>
<script type="text/javascript"
        src="Js-8-1/jquery.ui.draggable.js">
</script>
<style type="text/css">
       body{font-size:13px}
       div{margin:5px}
       .divFrame{border:dashed 1px #ccc;background-color:#eee;
            height:68px;width:200px}
       .divDrag{border:solid 1px #ccc;background-color:#eee;
                cursor:move;width:40px;
                padding:20px;text-align:center}
</style>
```

```
    <script type="text/javascript">
        $(function() {
            // 以透明度 0.35 随意拖动
            $("#divDragD").draggable({ opacity: 0.35 });
            // 以透明度 0.35 随意在 X 轴方向拖动
            $("#divDragX").draggable({ axis: "x", opacity: 0.35 });
            // 以透明度 0.35 随意在父窗口中拖动，拖动后返回原地
            $("#divDragC").draggable(
            { containment: "parent",
                opacity: 0.35,
                revert: true
            });
        })
    </script>
</head>
<body>
    <div id="divDragD" class="divDrag">随意</div>
    <div class="divFrame">
        <div id="divDragX" class="divDrag">X 轴</div>
    </div>
    <div class="divFrame">
        <div id="divDragC" class="divDrag">父窗口</div>
    </div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-2 所示。

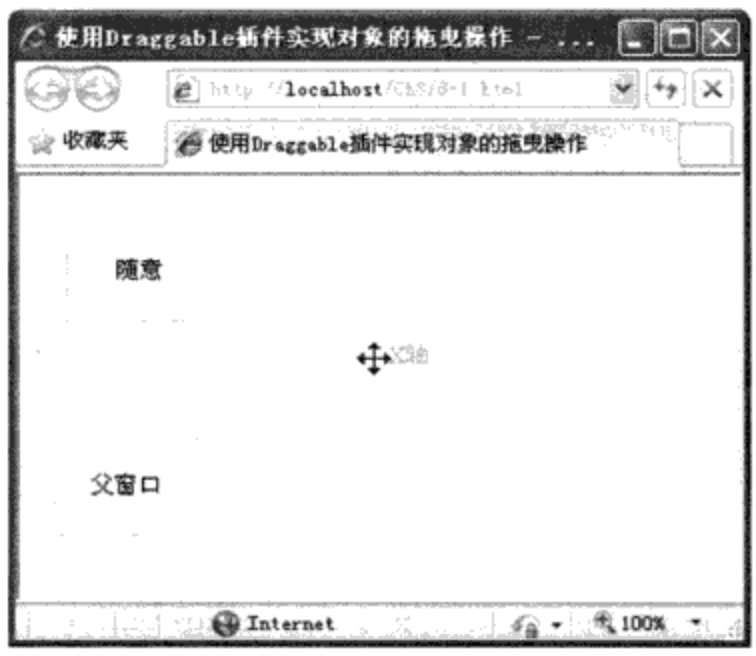


图 8-2　使用 draggable 插件实现对象的拖曳操作

（5）代码分析

在示例 8-2 中，通过 draggable 插件，开发人员可以很方便地设置页面中的某个元素进行

拖动的操作，如果要禁止拖动动作，只要加入如下代码：

```
$("div").draggable("diable");
```

禁止后，如果想开启元素的拖动操作，需要加入如下代码：

```
$("div").draggable("enable");
```

8.2.2　放置

在 jQuery UI 中，除使用 draggable 插件进行拖曳对象外，还可以通过 droppable（放置）插件“存放”拖曳的对象，即类似网上商城中购物车的效果，其调用的语法格式为：

```
droppable(options)
```

其中选 options 可接收的常用参数如表 8-2 所示。

表 8-2　选项 options 可接受的常用参数

参数名称	说　明
accept	可以为字符串或函数：如果是字符串，表示通过字符串获取的元素允许接收；如果是函数，表示只有执行函数后，返回 true 时，才能允许接收
activeClass	被接收的对象在拖曳时，接收容器的 CSS 样式
hoverClass	被接收的对象在进入接收容器时，容器的 CSS 样式
active	被接收的对象在拖曳时，调用的函数 function(e,ui)。其中参数 e 表示 event 事件，e.target 表示被拖曳的对象，参数 ui 表示与拖曳相关的对象
deactive	被接收的对象停止拖曳时，调用的函数，其中的参数说明与 active 中一样
over	被接收的对象拖曳到接收容器上方时，调用的函数，其中的参数说明与 active 中一样
out	被接收的对象拖出接收容器时，调用的函数，其中的参数说明与 active 中一样
drop	被接收的对象拖曳后，完全进入接收容器时，调用的函数，其中的参数说明与 active 中一样

下面通过一个简单的示例介绍 DOM 元素通过 droppable 插件实现放置的过程。

示例 8-2　使用 droppable 插件实现对象的放置操作

（1）插件文件

Js-Pub/ jquery.ui.core.js

Js-Pub/ jquery.ui.widget.js

Js-Pub/ jquery.ui.mouse.js

Js-8-1/ jquery.ui.draggable.js

Js-8-2/ jquery.ui.droppable.js

（2）功能描述

在页面中，创建一个产品区与购物车区，通过使用 droppable 插件将产品区中的产品拖动并放入购物车区中，同时，改变购物车的背景色。

（3）实现代码

新建一个 HTML 文件 8-2.html，加入如代码清单 8-2 所示的代码。

代码清单 8-2　使用 droppable 插件实现对象的置放操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 draggable 插件实现对象的放置操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.mouse.js">
    </script>
    <script type="text/javascript"
            src="Js-8-1/jquery.ui.draggable.js">
    </script>
    <script type="text/javascript"
            src="Js-8-2/jquery.ui.droppable.js">
    </script>
    <style type="text/css">
     body{font-size:13px}
     #divFrame{width:400px}
     .divLeft{float:left;border:solid 1px #666;width:100px}
     .divRight{float:right;border:solid 1px #666;width:200px}
     .divTitle{background-color:#ccc;padding:5px}
     .divDrag{padding:5px;cursor:move}
     .divGet{background-color:#eee}
     #divCart div{padding:5px;height:85px;font-size:11px}
</style>
<script type="text/javascript">
    $(function() {
        // 以透明度 0.35 随意拖动
        $(".divDrag").draggable({ opacity: 0.35 });
        $("#divCart").droppable({
            drop: function() {
                $(this)
            .addClass("divGet") // 改变购物车的 CSS
            .append($("<div></div>"))
            .find("#divTip").remove(); // 删除原有内容
            }
        })
    })
</script>
```

```
</head>
<body>
     <div id="divFrame">
          <div class="divLeft">
               <div class="divTitle">产品 </div>
               <div class="divDrag">
                    <img alt="" src="Images-8-2/img03.jpg" />
               </div>

          </div>
          <div class="divRight">
               <div class="divTitle">购物车 </div>
               <div id="divCart">
                    <div id="divTip">还没有产品 </div>
               </div>
          </div>
     </div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-3 所示。

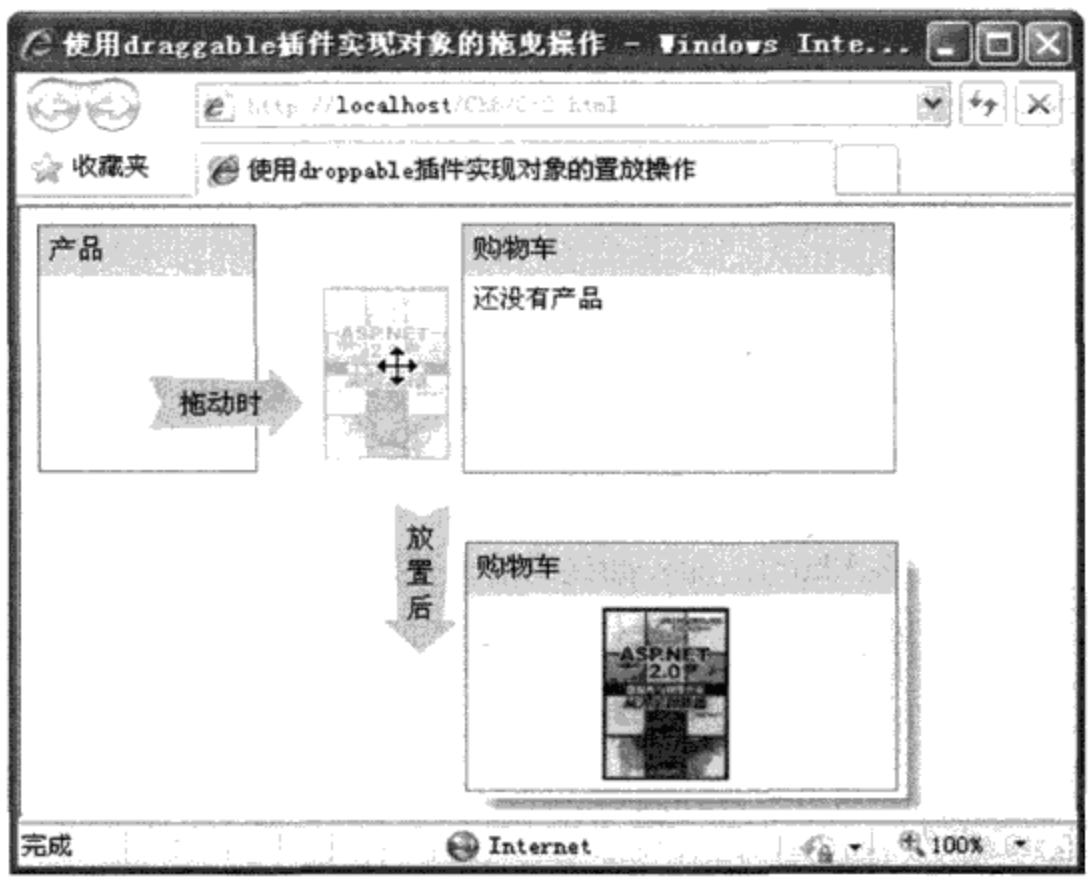


图 8-3　使用 droppable 插件实现对象的放置操作

（5）代码分析

在示例 8-2 中，产品从产品区拖至购物车时，通过 find("#divTip").remove() 代码，将购物车中原有提示信息消失，通过 append($(" ")) 代码将产品放入购物车中，另外，通

过 addClass("divGet") 代码改变购物车中的背景色。

8.2.3 排序插件

在 jQuery UI 中，除拖曳、放置指定元素外，还可以通过 sortable（排序）插件将有序列的标记，按照用户自己的想法任意拖曳其所在位置，形成一个新的序列，从而实现拖曳排序的功能。该插件的调用语法格式如下：

```
sortable (options)
```

其中，选项 options 所调用的参数与插件 draggable 中的 options 参数有很多相似之处。需要说明的是参数 item，该参数用于申明在页面中哪些元素以拖曳的方式进行排序。如下列代码：

```
$(element).sortable("option", "items", 'li' );
```

则说明，在页面中，有 <option>、<items>、 标记的元素可以通过拖曳的方式进行排序，如果不声明，则默认全部的表项型元素都可以。

下面通过一个简单的示例演示在列表中，通过 sortable 插件实现表项拖曳排序的过程。

示例 8-3 使用 sortable 插件实现列表中表项的拖曳排序操作

（1）插件文件

Js-Pub/ jquery.ui.core.js

Js-Pub/ jquery.ui.widget.js

Js-Pub/ jquery.ui.mouse.js

Js-8-3/jquery.ui.sortable.js

（2）功能描述

在页面中，创建 1 个列表 ，并在列表中增加 5 个表项 <li> 标记，通过 jQuery UI 中的排序插件——sortable，实现每个表项拖曳排序的功能。

（3）实现代码

新建一个 HTML 文件 8-3.html，加入如代码清单 8-3 所示的代码。

代码清单 8-3 使用 sortable 插件实现列表中表项的拖曳排序操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 sortable 插件实现列表中表项的拖曳排序操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
```

```
        src="Js-Pub/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.widget.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.mouse.js">
</script>
<script type="text/javascript"
        src="Js-8-3/jquery.ui.sortable.js">
</script>
<style type="text/css">
       body{font-size:13px}
       ul{padding:0px;margin:0px;
       list-style-type:none;width:260px}
       ul li{margin: 0 3px 3px 3px;border:solid 1px #ccc;
             background-color:#eee;padding:0.4em;cursor:move;
             font-size: 1.4em;height: 18px;}
</style>
<script type="text/javascript">
    $(function() {
        $("ul").sortable(
    { delay: 2, //为防与点击事件冲突，延时2秒
        opacity: 0.35 //以透明度0.35随意拖动
    })
    });
</script>
</head>
<body>
        <ul>
           <li>Item 1</li>
           <li>Item 2</li>
           <li>Item 3</li>
           <li>Item 4</li>
           <li>Item 5</li>
        </ul>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-4 所示。

（5）代码分析

在示例 8-3 的 JS 代码中，仅是通过 sortable 插件在页面中展示拖曳排序的操作，而实际的数据并没有进行保存，如果要保存页面中拖曳后的排列顺序，可以在 sortable 插件的选项 options 中设置一个 stop 参数，该参数执行一个回调函数，通过 $.post 方法与后台代码进行关联，相关的代码如下：

```
stop: function() {
        $.post("后台文件",
        $("ul").sortable("serialize"),
        function(data) {
             alert(data);
         });
}
```

图 8-4　使用 sortable 插件实现列表中表项的拖曳排序操作

在执行上述代码前，先给列表 <ul> 中的各个表项 <li> 设置相应的 Id 号，那么，通过代码 $("ul").sortable("serialize") 就可以获取各 Id 号排列的顺序，将这些顺序数据发送给后台，后台文件根据获取的顺序进行保存，从而真正实现表项的拖曳排序。

8.3　jQuery UI 微型插件

8.3.1　折叠面板插件

jQuery UI 插件折叠面板 (accordion) 可以实现页面中指定区域的折叠效果，这种效果俗语“手风琴”，即通过单击某块面板中的标题栏，就会展开相应的内容，当单击其他面板标题栏时，已展开的内容会自动关闭，通过这种方式，实现多个面板数据在一个页面中有序展示。其调用的语法格式为：

```
accordion(options)
```

其中选项 options 常用的参数如表 8-3 所示。

表 8-3　选项 options 中的常用参数

参数名称	说　明
animated	设置折叠时的效果，默认为“slide”；也可以自定义动画。如果设置为 false，表示不要设置折叠时的动画效果

（续）

参数名称	说　明
active	设置默认展开的主题选择，默认值为“1”
autoHeight	内容高度是否设置为自动增高，默认值为“true”
event	设置展开选项的事件，默认值为“click”，也可以设置双击、鼠标滑过事件
fillSpace	设置内容是否充满父元素的高度，默认值为“false”，如果设置为 true，那么，autoHeight 参数设置的值无效
icon	设置小图标，其设置的格式为 { "header": " 主题默认图标类别名 ", "headerSelected": " 主题选中时图标类别名 " }

下面通过一个简单的示例演示在页面中创建多个相同的区域，实现按主题折叠的效果。

示例 8-4　使用 accordion 插件实现页面中多区域的折叠操作

（1）插件文件

Js-Pub/ jquery.ui.core.js

Js-Pub/ jquery.ui.widget.js

Js-8-4/jquery.ui.accordion.js

（2）功能描述

在页面中，通过 accordion 插件展示几个相同区域的折叠效果。

（3）实现代码

新建一个 HTML 文件 8-4.html，加入如代码清单 8-4 所示的代码。

代码清单 8-4　使用 accordion 插件实现页面中多区域的折叠操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 accordion 插件实现区域块的折叠操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-8-4/jquery.ui.accordion.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .divFrame{width:260px}
```

```
            .divFrame h3{padding:5px;font-size:12px;
            border:solid 1px #ccc;background-color:#eee}
            .divFrame .divOpt{padding:5px;border:solid 1px #ccc}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#accordion").accordion();
        })
    </script>
</head>
<body>
  <div class="divFrame">
   <div id="accordion">
   <h3><a href="#">Section 1</a></h3>
   <div class="divOpt">
           <p>
             Item1
           </p>
   </div>
   <h3><a href="#">Section 2</a></h3>
   <div class="divOpt">
           <p>
             Item2
           </p>
   </div>
   <h3><a href="#">Section 3</a></h3>
   <div class="divOpt">
           <p>
             Item3
           </p>
   </div>
   <h3><a href="#">Section 4</a></h3>
   <div class="divOpt">
           <p>
             Item4
           </p>
   </div>
</div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-5 所示。

（5）代码分析

如果 accordion 插件在调用后，还需要获取或设置选项 options 中的值，那么可以通过下列方法进行。如获取初始化后的 active 值，代码如下所示：

```
var active = $("#accordion").accordion("option", "active" );
```

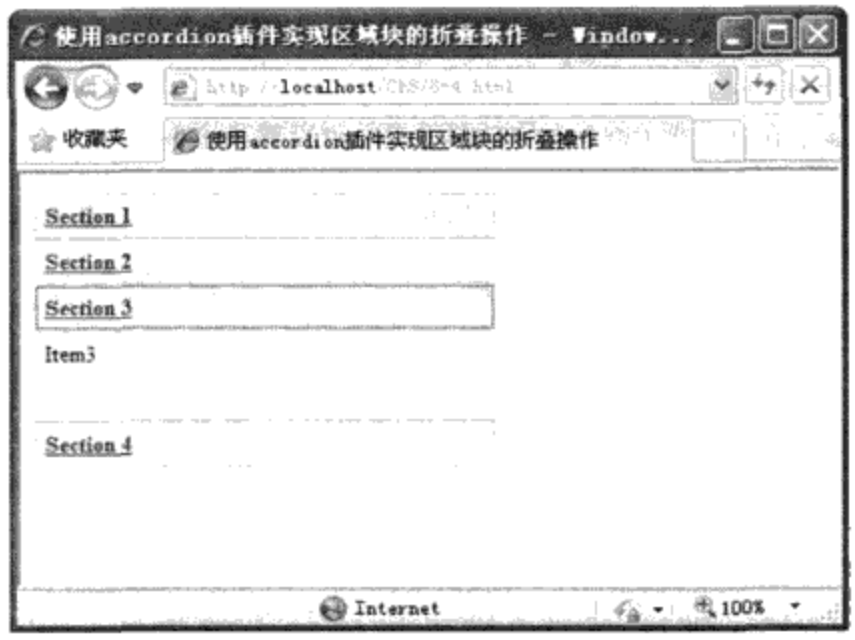


图 8-5　使用 accordion 插件页面中多区域的折叠操作

设置的代码如下：

```
var active = $("#accordion").accordion("option", "active",2);
```

其他参数在 accordion 插件初始化后的获取与设置与上述方法基本一样。

8.3.2　日历

在页面开发中，经常遇到需要用户输入日期的操作。通常的做法是，提供一个文本框（text），让用户输入，然后，编写代码验证输入的数据，检测其是否是日期型。这样比较麻烦，同时，用户输入日期的操作也不是很方便，影响用户体验。如果使用 jQuery UI 中的 datepicker（日历）插件，这些问题都可以迎刃而解。该插件调用的语法格式如下：

```
$(".selector").datepicker(options);
```

其中“.selector”表示 DOM 元素，一般指文本框，由于该插件的作用是提供日期选择，因此，常与一个文本框绑定，将选择后的日期显示在该文本框中。选项 options 是一个对象，与前面章节中插件的 options 一样，通过改变其参数对应的值，从而实现插件功能的变化，在 datepicker 插件中，选择 options 常用参数如表 8-4 所示。

表 8-4　选项 options 中的常用参数

参数名称	说　明
changeMonth	设置一个布尔值，如果为 true，则可以在标题处出现一个下拉选择框，可以选择月份，默认值为 false
changeYear	设置一个布尔值，如果为 true，则可以在标题处出现一个下拉选择框，可以选择年份，默认值为 false
showButtonPanel	设置一个布尔值，如果为 true，则在日期的下面显示一个面板，其中有两个按钮：一个为“今天”，另一个按钮为“关闭”，默认值为 false，表示不显示
closeText	设置关闭按钮上的文字信息，这项设置的前提是，showButtonPanel 的值必须为 true，否则显示不了效果

（续）

参数名称	说　明
dateFormat	设置显示在文本框 (text) 中的日期格式，可设置为 { dateFormat: 'yy-mm-dd' }，表示日期的格式为年 - 月 - 日，如 2012-10-1
defaultDate	设置一个默认日期值，如 {defaultDate: +7}，表示，弹出日期选择窗口后，默认的日期是在当前日期加上 7 天
showAnim	设置显示弹出或隐藏日期选择窗口的方式。可以设置的方式有，“show”、“slideDown”、“fadeIn”，或者为“”，表示没有弹出日期选择窗口的方式
showWeek	设置一个布尔值，如果为 true，则可以显示每天对应的星期，默认值为 false
yearRange	设置年份的范围，如 {yearRange: '2000:2010'}，表示年份下拉列表框的最小值为 2000 年，最大值为 2010 年，默认值为 c-10:c+10，当前年份的前后 10 年

下面通过一个示例，展示 datepicker 插件在页面中的基本用法。

示例 8-5　使用 datepicker 插件实现日期选择的基本操作

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-8-5/jquery.ui.datepicker.js

Css-Pub/demos.css

（2）功能描述

在页面中，当单击文本框时，通过 datepicker 插件弹出一个日期选择窗口，该窗口可以使用下拉列表框的方式选择年份与月份，并显示与日期相对应的星期，并且可以单击面板中的“关闭”按钮，关闭弹出的窗口。

（3）实现代码

新建一个 HTML 文件 8-5.html，加入如代码清单 8-5 所示的代码。

代码清单 8-5　使用 datepicker 插件实现日期选择的基本操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 datepicker 插件实现选择日期的操作一 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <link rel="stylesheet" type="text/css"
          href="Css-Pub/ui.all.css" />
    <script type="text/javascript"
          src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
          src="Js-8-5/jquery.ui.datepicker.js">
```

```
    </script>
     <link rel="stylesheet" type="text/css"
             href="Css-Pub/demos.css" />
     <style type="text/css">
            .txt{border:#666 1px solid;
           padding:2px;width:100px;margin-right:3px}
     </style>
     <script type="text/javascript">
        $(function() {
            $("#txtDate").datepicker(
            { changeMonth: true, // 显示下拉列表月份
                changeYear: true, // 显示下拉列表年份
                showWeek: true, // 显示日期对应的星期
                showButtonPanel: true, // 显示 " 关闭 " 按钮面板
                closeText: 'Close'// 设置关闭按钮的文本
            });
        })
    </script>
</head>
<body>
<div class="demo-description">
      <div>
          输入日期：<input name="txtDate" id="txtDate" class="txt" />
      </div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-6 所示。

图 8-6　使用 datepicker 插件实现选择日期的操作

通过示例 8-5 的演示，展示了 datepicker 插件基本的用法，但该插件的功能远不止这些，例如，在页面中，有时需要分时间段查询，这时，要求用户输入开始与结束时间，并且结束时间必须大于或等于开始时间。如果使用传统的 JavaScript 的方法编写，代码相对复杂；而使用 datepicker 插件则可以轻松选择两个符合条件的时间。下面通过一个示例介绍该功能实现的过程。

示例 8-6　使用 datepicker 插件实现分段时间的选择

（1）功能描述

在页面中，设置两个文本框 (text)，通过 datepicker 插件，分别获取开始时间与结束时间，同时，在选择开始时间后，再选择结束时间时，结束时间在大于或等于开始时间中选择。

（2）实现代码

新建一个 HTML 文件 8-6.html，加入如代码清单 8-6 所示的代码。

代码清单 8-6　使用 datepicker 插件实现分段时间的选择

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>使用 datepicker 插件实现选择日期的操作二 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
 <link rel="stylesheet" type="text/css"
         href="Css-Pub/ui.all.css" />
<script type="text/javascript"
        src="Js-Pub/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-8-5/jquery.ui.datepicker.js">
</script>
<script type="text/javascript"
        src="Js-8-6/jquery.ui.datepicker-zh-CN.js">
</script>
 <link rel="stylesheet" type="text/css"
         href="Css-Pub/demos.css" />
 <style type="text/css">
        .txt{border:#666 1px solid;padding:2px;
        width:100px;margin-right:3px}
 </style>
<script type="text/javascript">
    $(function() {
        $("#txtStart").datepicker( //绑定开始日期
          { changeMonth: true, //显示下拉列表月份
            changeYear: true, //显示下拉列表年份
            showWeek: true, //显示日期对应的星期
            firstDay: "1",
```

```
            onSelect: function(dateText, inst) {
             //设置结束日期的最小日期
             $('#txtEnd').datepicker('option',
             'minDate', new Date(dateText.replace('-', ',')))
            }
        })

        $("#txtEnd").datepicker( //绑定结束日期
        { changeMonth: true, //显示下拉列表月份
            changeYear: true, //显示下拉列表年份
            showWeek: true, //显示日期对应的星期
            firstDay: "1",
            onSelect: function(dateText, inst) {
             //设置开始日期的最大日期
             $('#txtStart').datepicker('option',
             'maxDate', new Date(dateText.replace('-', ',')))
            }
        })
    })
  </script>
</head>
<body>
<div class="demo-description">
    <div>
       开始日期：<input name="txtStart" id="txtStart" class="txt" />
       结束日期：<input name="txtEnd" id="txtEnd" class="txt" />
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 8-7 所示。

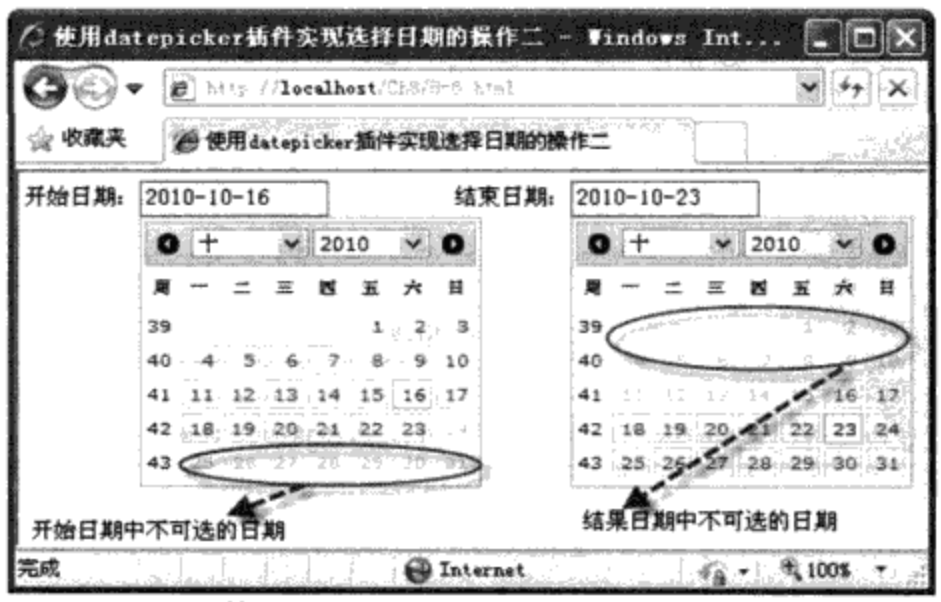


图 8-7　使用 datepicker 插件实现分段时间的选择

（4）代码分析

从示意图 8-7 中我们可以看出，通过对 datepicker 插件中 onSelect 事件的设置，使选择开始时间时，改变了结束日期的最小日期的选择范围。同理，在选择结束日期时，改变了开始日期的最大日期的选择范围，从而有效地规避了时间段中不规范数据的出现。

另外，为了更好地展示 datepicker 插件弹出日期选择窗口的原始效果，在页面中需引用与该插件配套的两个官方提供的样式文件，代码如下：

```
<link rel="stylesheet" type="text/css"
          href="Css-Pub/ui.all.css" />
<link rel="stylesheet" type="text/css"
          href="Css-Pub/demos.css" />
```

同时，为了使弹出的日期选择窗口支持中文，需要在页面再引入一个 js 文件，该文件将插件中的英文部分翻译成中文，其实现的代码如下：

```
<script type="text/javascript"
           src="Js-8-6/jquery.ui.datepicker-zh-CN.js">
```

导入 JS 文件 jquery.ui.datepicker-zh-CN.js 后，在再次弹出的日期选择窗口中，所显示的英文字符全部转换成了中文，其示意图如图 8-7 所示。

8.3.3 选项卡插件

选项卡（tabs）在页面中的使用非常广泛，尤其是各大门户网站的首页，因为以选项卡的形式可以实现使用少量的空间展示更多内容的效果，同时，其快速切换的效果，也增加了用户的体验。在 jQuery UI 中，通过在页面中导入 tabs 插件，并调用插件中的 tabs() 方法直接针对列表生成对应菜单，轻松实现这种选项卡的功能。其调用的语法格式如下：

```
tabs(options)
```

其中选项 options 的常用参数如表 8-5 所示。

表 8-5 选项 options 中的常用参数

参数名称	说　明
collapsible	是否可折叠选项卡的内容，设置一个布尔值，如果为 true，那么，允许用户可折叠选卡的内容，即首次单击展开，再单击关闭，默认值为 false
disabled	设置不可用选项卡，如 {disabled: [1, 2] }，表示选项卡中，第 1、第 2 项不可用
event	设置触发切换选项卡的事件，默认值为“click”，也可以设置为“mousemove”
fx	设置切换选项卡时的一些动画效果
selected	设置被选中选项卡的 Index，如 {selected: 2}，表示第 2 项选项卡被选中

下面通过一个示例，展示 tabs 插件在页面中的基本用法。

示例 8-7　使用 tabs 插件展示选项卡的基本功能

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-8-7/jquery.ui.tabs.js

Css-Pub/demos.css

（2）功能描述

在页面中，创建 1 个列表 <ul> 标记，并在 <ul> 中创建 3 个 <li> 标记，通过 tabs 插件中的 tabs 方法使这 3 个 <li> 标记形成选项卡，其中前 2 个选项卡对应指定页面中的 内容，后 1 个选项通过 ajax 的方式，动态加载一个 HTML 页面内容，同时，在变换选项卡时，展示相应的切换效果。

（3）实现代码

新建一个 HTML 文件 8-7.html，加入如代码清单 8-7 所示的代码。

代码清单 8-7　使用 tabs 插件展示选项卡的基本功能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 tabs 插件展示选项卡的基本功能 </title>
     <script type="text/javascript"
             src="Jscript/jquery-1.4.2.js">
     </script>
     <link rel="stylesheet" type="text/css"
              href="Css-Pub/ui.all.css" />
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.core.js">
    </script>
    <script type="text/javascript"
            src="Js-Pub/jquery.ui.widget.js">
    </script>
    <script type="text/javascript"
            src="Js-8-7/jquery.ui.tabs.js">
    </script>
     <link rel="stylesheet" type="text/css"
              href="Css-Pub/demos.css" />
    <script type="text/javascript">
        $(function() {
            $("#tabs").tabs({
                // 设置各选项卡在切换时的动画效果
                fx: { opacity: "toggle", height: "toggle" },
                // 通过移动鼠标事件切换选项卡
```

```
                event: "mousemove"
            })
        })
    </script>
</head>
<body>
<div class="demo-description">
    <div id="tabs" style="width:360px">
    <ul>
        <li><a href="#tabs-1">One</a></li>
        <li><a href="#tabs-2">Two</a></li>
        <!-- 第三个选项卡的内容为 ajax 方法获取的静态页 -->
        <li><a href="Ajax-8-7/Content3.html">Three</a></li>
    </ul>
    <div id="tabs-1">
        <p>One 选项卡中的内容 </p>
    </div>
    <div id="tabs-2">
        <p>Two 选项卡中的内容 </p>
    </div>
</div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后的效果如图 8-8 所示。

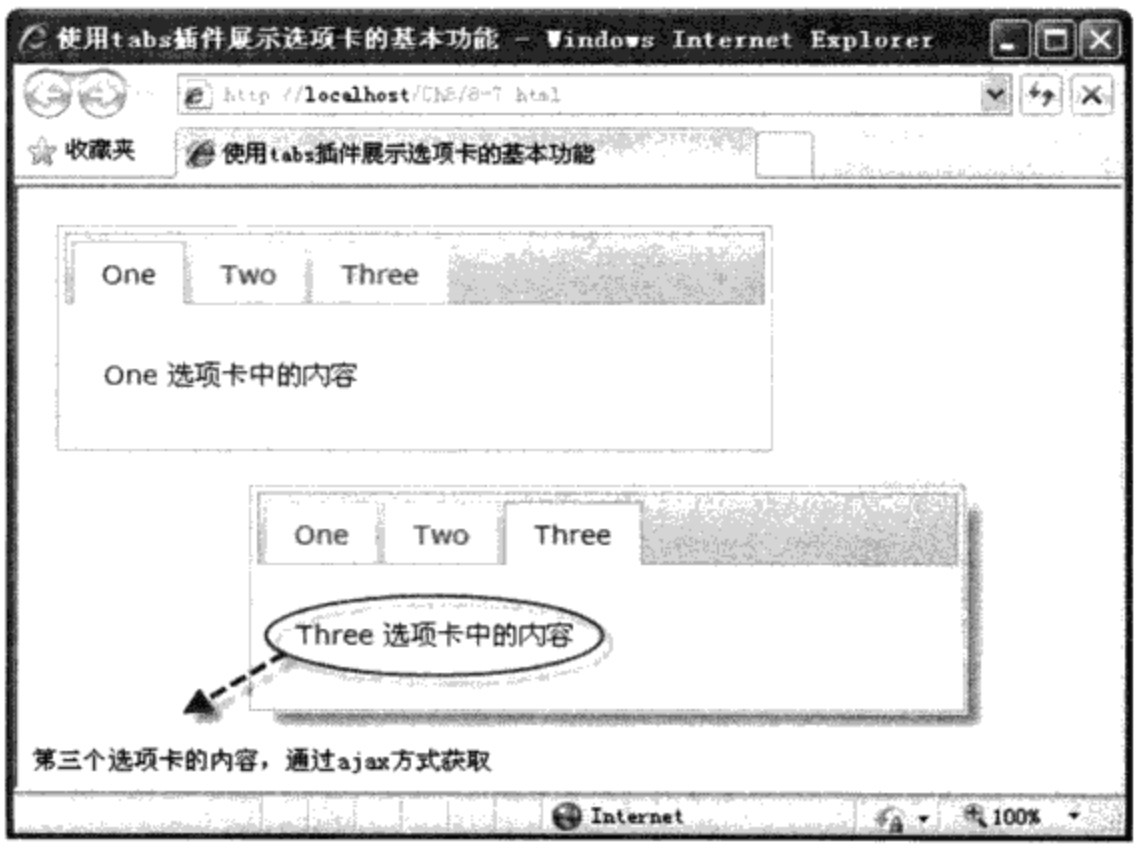


图 8-8　使用 tabs 插件展示选项卡的基本功能

在示例 8-7 中，第三个选项卡的内容通过 ajax 的方式调用一个 HTML 页面 Content3.html 获取，该页面的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <p>Three 选项卡中的内容 </p>
</body>
</html>
```

（5）代码分析

在上述示例 8-7 中，除直接通过超级链接的方式加载选项卡的内容外，还可以调用 tabs 插件中的 url() 方法，改变 ajax 方式加载的页面内容。如下列代码：

```
$("#tabs").tabs("url", 2, "8-1.html");
```

执行上述代码后，第三个选项卡的内容在使用 ajax 方式加载时，已经发生了变化，不再指向页面 content3.html，而是指向 HTML 页面 8-1.html。

另外，可以通过下列的代码，增加或删除一个选项卡，代码如下：

```
// 在选项卡后面，动态增加一个标签为 "four" 的选项卡，内容指向 "8-1.html"
$("#tabs").tabs('add', "8-1.html", "four");
// 删除选卡中的第三个选项卡，其中 "2" 为索引号，索引号从 0 开始
$("#tabs").tabs('remove', 2);
```

8.3.4　对话框插件

在页面开发过程中，经常要与用户进行交互，例如在提交表单时，如果文本框 (text) 中内容为空，需要提醒用户输入内容，一般的做法是使用传统的 JavaScript 语言中的 alert() 函数弹出一个信息窗口；另外，在删除某项记录时，也需要告知用户确定，可用 JavaScript 语言中的 confirm() 函数，虽然这两个函数都可以实现相应的功能，但没有动画效果，功能单一，用户体验差。在 jQuery UI 中，通过 dialog（对话）插件，不仅完成可以实现传统 JavaScript 语言中 alert() 函数与 confirm() 函数的功能，而且界面优雅，功能丰富，操作简便。该插件导入页面后，其调用语法格式代码如下：

```
$(".selector").dialog(options)
```

其中 .selector 表示 DOM 元素，一般指定一个 标记，用于显示弹出对话框的内容和设置的按钮，选项 options 是一个对象，它常用的参数如表 8-6 所示。

表 8-6 选项 options 中的常用参数

参数名称	说　明
autoOpen	设置一个布尔值，如果为 false，则不显示对话框，默认值为 true
bgiframe	设置一个布尔值，如果为 true，则表示如果在 IE 6 下，弹出的对话框可以遮盖住页面中类似于 <select> 标记的下拉列表框，默认值为 false
buttons	设置对话框中的按钮，如 { "buttons", { "Ok": function() { $(this).dialog("close"); }}} 则表示，设置了一个文本内容为“Ok”的按钮，单击该按钮，将关闭对话框
closeOnEscape	设置一个布尔值，如果为 false，则表示不使用 ESC 快捷键的方式关闭对话框，默认值为 true
draggable	设置一个布尔值，表示是否可以拖动对话框，默认值为 true
hide	设置对话框关闭时的动画效果，可以设置为“slide”等各种动画效果，默认值为 null
modal	设置对话框是否以模式的方式显示，模式指的是页面背景变灰，不允许操作，焦点锁定对话框的效果，默认值为 false
position	设置对话框弹出时，在页面中的位置，可以设置为“top”、“center”、“bottom”、“left”、“right”，默认值为 center
show	设置对话框显示时的动画效果，相关说明与 hide 参数一样
title	设置对话框中主题部分的文字，如“系统提示”，默认值为空

下面通过一个示例，展示 dialog 插件在页面中弹出提示信息对话框与确定信息对话框的过程。

示例 8-8 使用 dialog 插件弹出提示和确定信息对话框

（1）插件文件

Css-Pub/ui.all.css

Js-Pub/jquery.ui.core.js

Js-Pub/jquery.ui.widget.js

Js-Pub/ jquery.ui.mouse.js

Js-8-1/ jquery.ui.draggable.js

Js-Pub/jquery.ui.position.js

Js-8-8/jquery.ui.dialog.js

Css-Pub/demos.css

（2）功能描述

在页面中，设置 1 个文本框和 2 个按钮，同时，以 标记显示一条记录；当单击第一个“提交”按钮时，将检测文本框的内容是否为空，如果为空，则通过 dialog 插件弹出一个对话框，提示文本框内容不能为空；当单击第二个“删除”按钮时，先通过 dialog 插件弹出一个确定对话框，当用户单击“确定”按钮时，将删除页面中 标记的内容。

（3）实现代码

新建一个 HTML 文件 8-8.html，加入如代码清单 8-8 所示的代码。

代码清单 8-8　使用 dialog 插件弹出提示和确定信息对话框

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>使用 dialog 插件弹出提示和确定信息对话框 </title>
   <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
   </script>
   <link rel="stylesheet" type="text/css"
            href="Css-Pub/ui.all.css" />
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.core.js">
   </script>
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.widget.js">
   </script>
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.mouse.js">
   </script>
   <script type="text/javascript"
           src="Js-8-1/jquery.ui.draggable.js">
   </script>
   <script type="text/javascript"
           src="Js-Pub/jquery.ui.position.js">
   </script>
   <script type="text/javascript"
           src="Js-8-8/jquery.ui.dialog.js">
   </script>
   <link rel="stylesheet" type="text/css"
           href="Css-Pub/demos.css" />
   <style type="text/css">
           div{line-height:1.8em}
           .txt{border:#666 1px solid;padding:2px;width:180px;margin-right:3px}
           button,.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() {
           // 检测按钮事件
           $("#btnSubmit").bind("click", function() {
               if ($("#txtName").val() == "") { // 如果文本框为空
                   sys_Alert(" 姓名不能为空！请输入姓名 ");
               }
           });
           // 询问按钮事件
```

```
    $("#btnDelete").bind("click", function() {
        if ($("#spnName").html() !=null) { //如果对象不为空
            sys_Confirm("您真的要删除该条记录吗？");
            return false;
        }
    });
});
function sys_Alert(content) { //弹出提示信息窗口
    $("#dialog-modal").dialog({
        height: 140,
        modal: true,
        title: '系统提示',
        hide: 'slide',
        buttons: {
            Cancel: function() {
                $(this).dialog("close");
            }
        },
        open: function(event, ui) {
            $(this).html("");
            $(this).append("<p>" + content + "</p>");
        }
    });
}
function sys_Confirm(content) { //弹出询问信息窗口
    $("#dialog-modal").dialog({
        height: 140,
        modal: true,
        title: '系统提示',
        hide: 'slide',
        buttons: {
            '确定': function() {
                $("#spnName").remove();
                $(this).dialog("close");
            },
            '取消': function() {
                $(this).dialog("close");
            }
        },
        open: function(event, ui) {
            $(this).html("");
            $(this).append("<p>" + content + "</p>");
        }
    });
}
</script>
</head>
<body>
    <div class="demo-description">
    <div style="background-color:#eee;padding:5px;width:260px">
```

```
        <input id="txtName" type="text" class="txt" />
        <input id="btnSubmit" type="button" value="提交" class="btn" />
    </div>
    <div style="padding:5px;width:260px">
        <span id="spnName">张三</span>
        <input id="btnDelete" type="button" value="删除" class="btn" />
    </div>
    <div id='dialog-modal'></div>
</div>
</body>
</html>
```

（4）页面效果

代码执行后，单击“提交”按钮时，如果文本框中的内容为空，将弹出一个带模式的提示对话框，其实现的页面效果如图 8-9 所示。

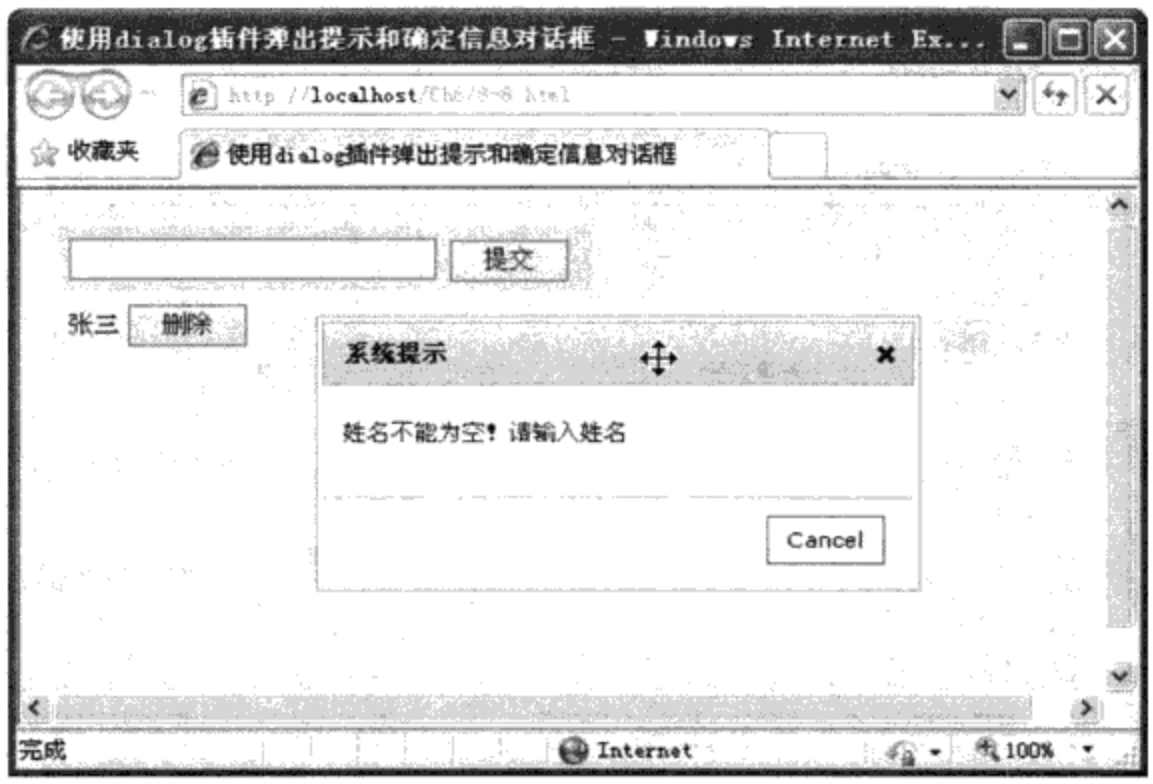


图 8-9　显示带模式的提示对话框

在示例 8-8 执行的页面中，如果单击“删除”按钮，将出现一个带模式的询问对话框，当单击对话框中的“确定”按钮后，才能将页面中的 标记内容删除，页面效果如图 8-10 所示。

（5）代码分析

Dialog 插件实现的对话框通常与一个 标记绑定，只要执行插件中的 dialog() 方法就可以弹出对话框，但在实际应用中，需要根据特定的条件，弹出不同内容与类型的对话框（如提示、询问），这时，需要将插件实现的弹出对话框的代码进行封装，即自定义一个函数，如下列代码：

```
function sys_Alert(content) { // 弹出提示信息窗口
    $("#dialog-modal").dialog({
        height: 140,
        modal: true,
        title: '系统提示',
        hide: 'slide',
        buttons: {
            Cancel: function() {
                $(this).dialog("close");
            }
        },
        open: function(event, ui) {
            $(this).html("");
            $(this).append("<p>" + content + "</p>");
        }
    });
}
```

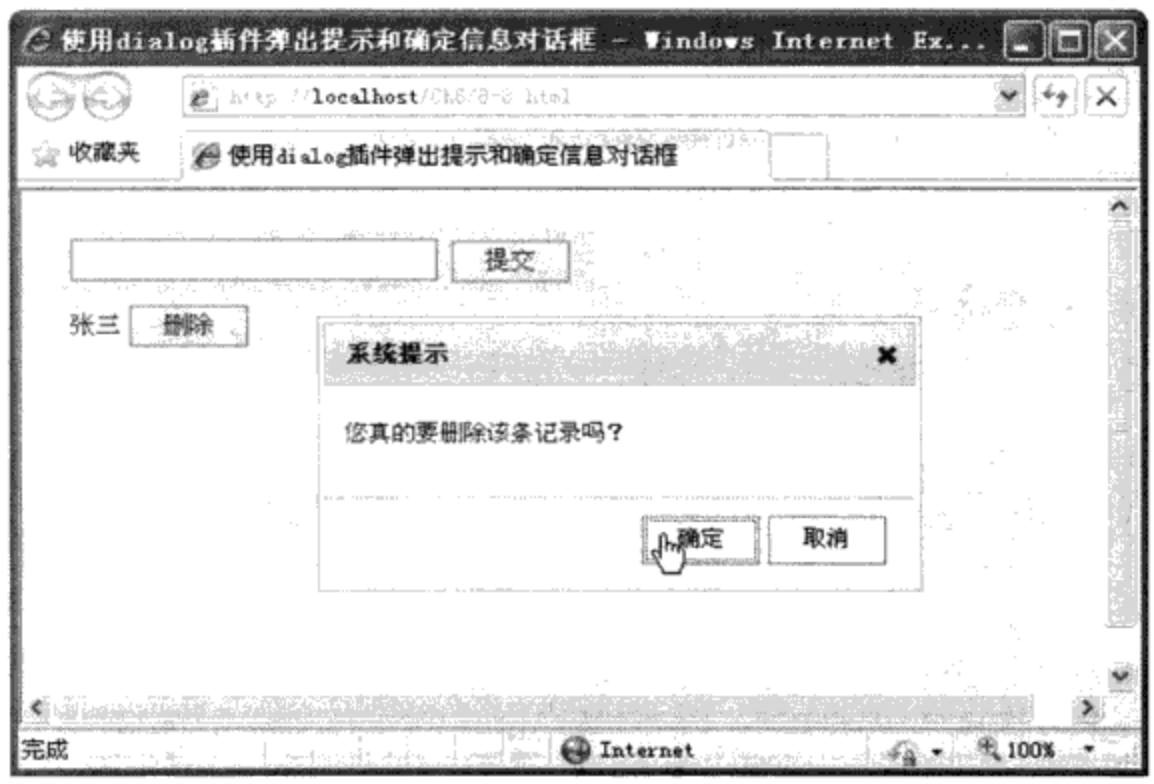


图 8-10　显示带模式的询问对话框

当页面在交互过程中，需要弹出对话框时，直接调用该函数即可，例如文本框内容为空时，弹出提示信息，代码如下所示：

```
... 省略部分代码
if ($("#txtName").val() == "") { // 如果文本框为空
      sys_Alert("姓名不能为空！请输入姓名");
}
... 省略部分代码
```

在删除时的询问对话框，也是先自定义一个函数，该函数执行时，通过 dialog 插件，弹出一个询问对话框，页面在交互时，如果要弹出询问对话框，直接调用该

函数即可。

8.4　综合案例分析——使用 jQuery UI 插件以拖动方式管理相册

8.4.1　需求分析

经分析，该案例的需求如下：

1）可以将列表中的某张图片，通过拖动或单击“删除”链接的方式，移至回收站。

2）在回收站中，可以将列表中的某张图片，通过拖动或单击“还原”链接的方式，返回相册。

3）在将图片拖动或还原时，需要有动画效果，以表示正在进行的动作。

8.4.2　效果界面

该案例的效果界面如下所示：

1）在图片列表中，单击某张图片，出现移动鼠标样式后就可以拖动该图片，直至回收站中。其实现的界面如图 8-11 所示。

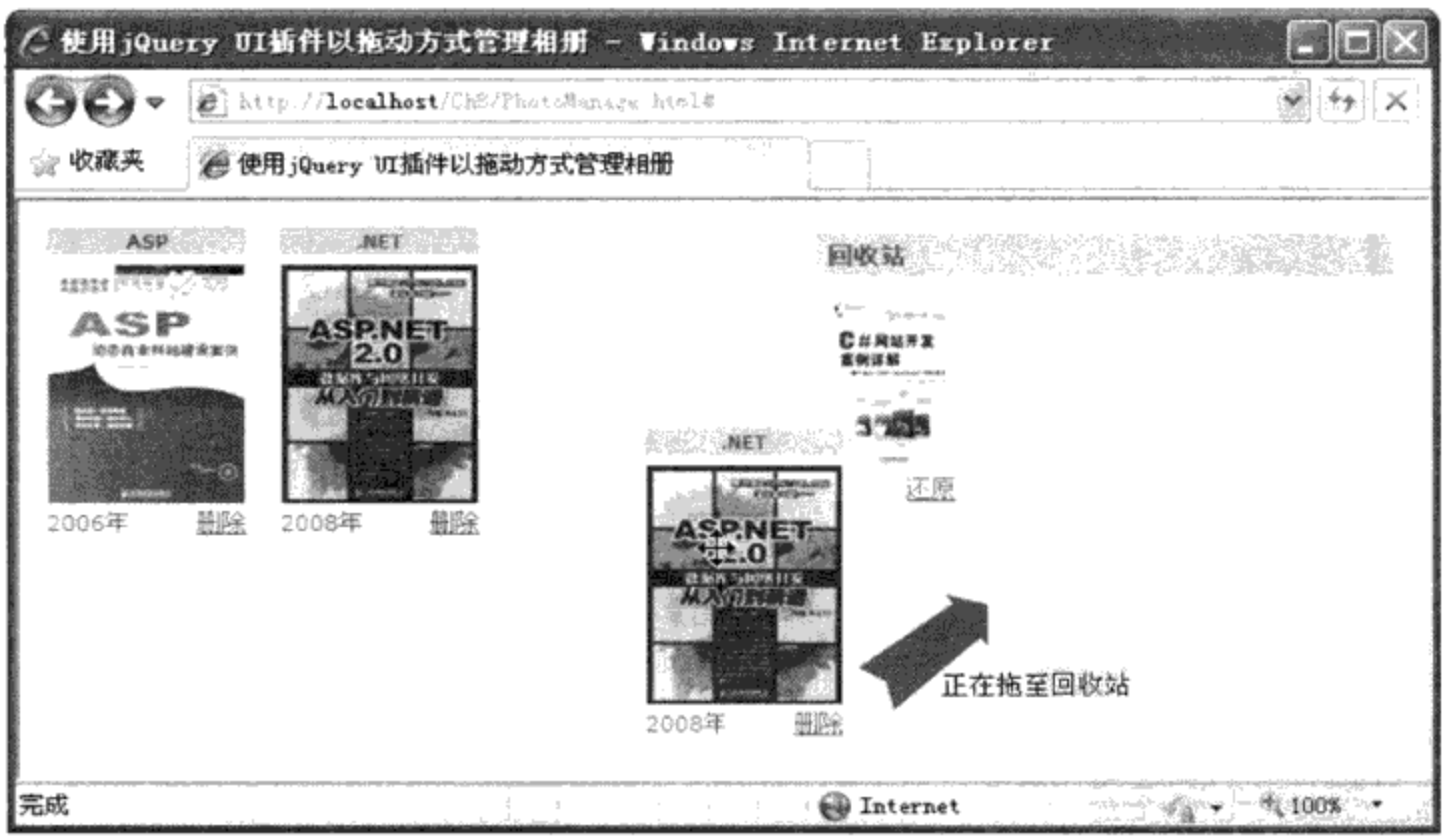


图 8-11　将列表中的图片拖至回收站

2）在回收站中，如果单击“还原”链接或拖动某图片，可以将回收站中的图片还原至列表中，其实现的界面如图 8-12 所示。

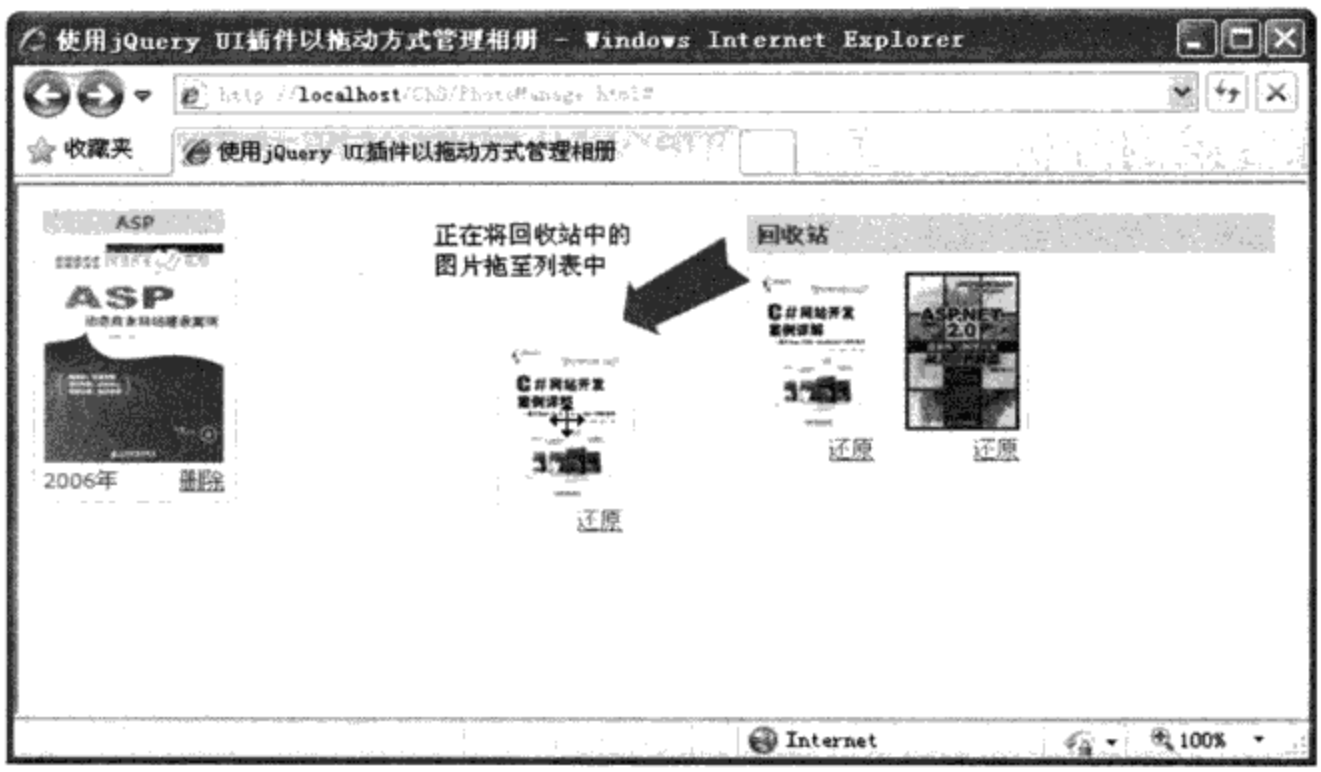


图 8-12　将回收站中的图片还原至列表

3）无论是“删除”还是“还原”图片，在拖动至板块后，以动画的效果进行排列，如将图片拖至回收站中，其排列的动画效果如图 8-13 所示。

图 8-13　将删除的图片以动画的效果排列至回收站

8.4.3　功能实现

在本案例中，运用了拖动 (draggable) 和放置 (droppable) 两款 jQuery UI 插件，实现图片

从列表拖至回收站中，然后从回收站还原到列表中的功能。

新建一个页面文件 PhotoManage.html，加入如代码清单 8-9 所示的代码。

代码清单 8-9　使用 jQuery UI 插件以拖动的方式管理图片

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0    Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>使用 jQuery UI 插件以拖动的方式管理相册 </title>
<script type="text/javascript"
        src="Jscript/jquery-1.4.2.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.core.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.widget.js">
</script>
<script type="text/javascript"
        src="Js-Pub/jquery.ui.mouse.js">
</script>
<script type="text/javascript"
        src="Js-8-1/jquery.ui.draggable.js">
</script>
<script type="text/javascript"
        src="Js-8-2/jquery.ui.droppable.js">
</script>
<link rel="stylesheet" type="text/css"
         href="Css/PhotoManage.css" />
<script type="text/javascript">
    $(function() {
      // 使用变量缓存 DOM 对象
      var $photo = $("#photo");
      var $trash = $("#trash");
      // 可以拖动包含图片的表项标记
      $("li", $photo).draggable({
          revert: "invalid", // 在拖动过程中，停止时将返回原来位置
          helper: "clone", // 以复制的方式拖动
          cursor: "move"
      });
      // 将相册中的图片拖动到回收站
      $trash.droppable({
          accept: "#photo li",
          activeClass: "highlight",
          drop: function(event, ui) {
              deleteImage(ui.draggable);
          }
      });
      // 将回收站中的图片还原至相册
```

```
$photo.droppable({
    accept: "#trash li",
    activeClass: "active",
    drop: function(event, ui) {
        recycleImage(ui.draggable);
    }
});
// 自定义图片从相册中删除到回收站的函数
var recyclelink = "<a href='#' title='从回收站还原'
                    class='phrefresh'>还原</a>";
function deleteImage($item) {
    $item.fadeOut(function() {
        var $list = $("<ul class='photo reset'/>")
                    .appendTo($trash);
        $item.find("a.phtrash").remove();
        $item.append(recyclelink)
        .appendTo($list).fadeIn(function() {
            $item
            .animate({ width: "61px" })
            .find("img")
            .animate({ height: "86px" });
        });
    });
}
// 自定义图片从回收站还原至相册时的函数
var trashlink = "<a href='#' title='放入回收站'
                  class='phtrash'>删除</a>";
function recycleImage($item) {
    $item.fadeOut(function() {
        $item
         .find("a.phrefresh")
         .remove()
         .end()
         .css("width", "85px")
         .append(trashlink)
         .find("img")
         .css("height", "120px")
         .end()
         .appendTo($photo)
         .fadeIn();
    });
}
// 根据图片所在位置绑定删除或还原事件
$("ul.photo li").click(function(event) {
    var $item = $(this),
        $target = $(event.target);
    if ($target.is("a.phtrash")) {
        deleteImage($item);
    } else if ($target.is("a.phrefresh")) {
        recycleImage($item);
```

```
            }
            return false;
        });
    });
</script>
</head>
<body>
<div class="phframe">
  <ul id="photo" class="photo">
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">ASP</h5>
          <img src="Images/img01.jpg" alt="2006 年图书作品 "
               width="85" height="120" />
          <span>2006 年 </span>
          <a href="#" title=" 放入回收站 " class="phtrash"> 删除 </a>
   </li>
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">.NET</h5>
          <img src="Images/img02.jpg" alt="2008 年图书作品 "
               width="85" height="120" />
          <span>2008 年 </span>
          <a href="#" title=" 放入回收站 " class="phtrash"> 删除 </a>
   </li>
   <li class="photoframecontent photoframetr">
          <h5 class="photoframeheader">C#</h5>
          <img src="Images/img03.jpg" alt="2010 年图书作品 "
               width="85" height="120" />
          <span>2010 年 </span>
          <a href="#" title=" 放入回收站 " class="phtrash"> 删除 </a>
    </li>
  </ul>
  <div id="trash" class="photoframecontent">
      <h4 class="photoframeheader"> 回收站 </h4>
  </div>
</div>
</body>
</html>
```

为了更好地展示图片在各个板块中的拖动页面效果，本案例中创建了一个 CSS 样式文件 PhotoManage.css，加入如代码清单 8-10 所示的代码。

代码清单 8-10　页面 PhotoManage.html 中导入的 CSS 文件

```
body{font-size:11px;font-family:Verdana,Arial,sans-serif;font-size:11px}
/* 图片样式设置 */
#photo {float:left;width:55%;  min-height:12em;padding:0;margin:0px;list-style-
        type:none}
.photo li { float: left; width: 96px; padding: 0.4em;
       margin: 0 0.4em 0.4em 0; text-align:center}
```

```
.photo li h5 { margin: 0 0 0.4em; cursor: move}
.photo li span { float:left}
.photo li a { float: right; }
.photo li img { width: 98%;cursor:move;border:solid 1px #eee;
      margin-bottom:3px}
/* 图片外框样式设置 */
.phframe {width:660px}
.photoframecontent {border: 1px solid #ccc;color:#666}
.photoframecontent a {color:#666}
.photoframeheader {border: 1px solid #ccc;background:#ccc;
     color:#666;font-weight:bold}
.photoframeheader a {color: #666}
.photoframetr {-moz-border-radius-topright:4px;
     -webkit-border-top-right-radius:4px}
/* 特殊样式设置 */
.active {background:#eee;}
.highlight {border:1px solid #fcefa1;background: #fbf9ee }
.reset {margin:0;padding:0; border:0;outline:0;line-height:1.3;
    text-decoration:none;font-size:100%; list-style:none}
.phtrash, .phrefresh{color:#666}
/* 回收站样式设置 */
#trash {float:right;width:42%;min-height:18em;padding:1%}
#trash h4 {line-height: 16px; margin: 0 0 0.4em;
    padding:4px 0px 0px 4px}
#trash .photo h5, #trash span { display:none;}
```

8.4.4 代码分析

本案例的核心功能是图片以拖动或单击链接的方式，从一个板块转移至另外一个板块，因此，首先在页面中，导入与拖动和放置相关的 jQuery UI 插件，两款功能插件的导入代码如下所示：

```
... 省略部分代码
<script type="text/javascript"
        src="Js-8-1/jquery.ui.draggable.js">
</script>
<script type="text/javascript"
        src="Js-8-2/jquery.ui.droppable.js">
</script>
... 省略部分代码
```

在页面头文件部分，除引入上面两款功能插件和 jQuery 库外，其他导入的 JS 文件中，jquery.ui.core.js 为 jQuery UI 的核心库，另外两个 JS 文件为辅助功能插件文件。

为了实现相册与回收站板块间图片的相互拖动和放置，先通过变量缓存这两个 jQuery 对象，为后续的操作提供方便和节省空间，代码如下：

```
// 使用变量缓存 DOM 对象
var $photo = $("#photo");
```

```
var $trash = $("#trash");
```

因为图片以表项 <li> 的形式展示，因此，在前者定义“$photo”对象中，查找表项 <li> 标记，并设置该对象可以进行拖动，其实现的代码如下所示：

```
// 可以拖动包含图片的表项标记
$("li", $photo).draggable({
   revert: "invalid", // 在拖动过程中，停止时将返回原来位置
   helper: "clone", // 以复制的方式拖动
   cursor: "move"
});
```

接下来要完成的代码是，拖放后的图片放置。由于有两种需求，一是从图片列表中拖入回收站，另外一种是从回收站还原至图片列表。因此，这两种情况要分别编写代码。实现第一种需求的代码如下所示：

```
// 将相册中的图片拖动到回收站
$trash.droppable({
   accept: "#photo li",
   activeClass: "highlight",
   drop: function(event, ui) {
         deleteImage(ui.draggable);
   }
});
```

上述代码设置了 $trash 元素接收对象为 "#photo li"，即图片列表中的表项，拖动时的样式类别名为“active”。当图片拖动后放入回收站时，触发一个回调函数 deleteImage ()，在该函数中，获取的表项“$item”在淡出时，先在回收站中增加一个列表 <ul> 标记，并移除原先列表中的“删除”链接“a.phtrash”，然后，将“还原”链接字符串“recyclelink”加入表项中，部分代码如下：

```
$item.fadeOut(function() {
   var $list = $("<ul class='photo reset'/>").appendTo($trash);
   $item.find("a.phtrash").remove();
   $item.append(recyclelink);
   ...
});
```

同时，在回收站中，以淡入的方式展示图片时，先将获取的表项“$item”，采用动画的方式缩小长度，然后，获取该表项中的 <img> 元素，并以动画的方式缩小 <img> 元素的高度，通过这样连贯的动作，使图片在放入回收站时动画效果更流畅。其代码如下：

```
$item.appendTo($list).fadeIn(function() {
   $item.animate({ width: "61px" })
   .find("img").animate({ height: "86px" });
});
```

第二种情况是，将图片从回收站还原至图片列表中，为此，图片列表绑定放置方法

droppable()，接收从回收站中拖放的图片，其实现的代码如下：

```
// 将回收站中的图片还原至相册
$photo.droppable({
    accept: "#trash li",
    activeClass: "active",
    drop: function(event, ui) {
          recycleImage(ui.draggable);
    }
});
```

在图片列表绑定 droppable() 方法中，声明接收对象为“#trash li”，拖动时的样式类别名称为“active”，当图片拖动至图片列表中时，触发一个回调函数 recycleImage()，该函数中先获取可以放置的对象“$item”，在该对象淡出回收站的过程中，移出“还原”链接字符“a.phrefresh”，新增“删除”链接字符串“trashlink”，其实现代码如下所示：

```
$item.fadeOut(function() {
    $item
    .find("a.phrefresh")
    .remove()
    .end()
    .css("width", "85px")
    .append(trashlink)
     ...
});
```

此外，在回调函数 recycleImage() 中，当图片从回收站中淡出时，获取放置对象“$item”中的 <img> 元素，并还原其高度，插入图片列表中，并以淡入的方式显示插入的图片，其实现的代码如下所示：

```
$item.fadeOut(function() {
    $item
     ...
    .find("img")
    .css("height", "120px")
    .end()
    .appendTo($photo)
    .fadeIn();
});
```

图片除拖动方式实现“删除”与“还原”功能外，还可以单击列表中图片下方的“删除”与“还原”字符链接，单击链接时，直接调用相应的自定义函数即可。因此，还需要设置图片列表 <li> 中字符链接的单击事件，其实现的代码如下：

```
// 根据图片所在位置绑定删除或还原事件
$("ul.photo li").click(function(event) {
    var $item = $(this),
    $target = $(event.target);
```

```
    if ($target.is("a.phtrash")) {
            deleteImage($item);
        } else if ($target.is("a.phrefresh")) {
            recycleImage($item);
        }
     return false;
});
```

代码中，“$target”为事件触发的对象，如果该对象为“a.phtrash”，表示单击了“删除”链接，则执行自定义函数 deleteImage($item)，否则，执行自定义函数 recycleImage($item)，最后，加入 return false 语句，避免单击链接时页面跳转。

另外，为了展示图片列表中的图片与回收站中的图片不同，案例中，通过一个 CSS 样式隐藏回收站中图片的 <h5> 与 标记，其代码如下所示：

```
#trash .photo h5, #trash span {display:none;}
```

8.5　本章小结

jQuery UI 插件在 jQuery 库中占有重要地位，本章先是通过一个个完整的示例，对 jQuery UI 常用插件的功能和运用进行了详细的介绍，最后，通过一个综合的案例，介绍拖动 (draggable) 与放置 (droppable) 插件在图片管理中的进阶应用，旨在使读者能够通过这些示例，掌握 jQuery UI 常用插件的基本用法，并能够运用到日常的页面开发中，减少代码的编写总量，提高项目开发速度。

第 9 章

jQuery 实用工具函数

本章内容

- 什么是工具函数
- 工具函数的分类
- 工具函数的扩展
- 其他工具函数——$.proxy()
- 综合案例分析——使用 jQuery 扩展工具函数实现对字符串指定类型的检测
- 本章小结

在前面的章节中，我们曾经使用过 $.each()、$.extend 函数，在 jQuery 中，它们是非常实用的全局性工具函数中的成员。除此之外，还有很多功能强大的工具函数，为我们在处理对象和操作数组方面提供便利。下面我们详细介绍这些工具函数的定义和调用方法。

9.1　什么是工具函数

在 jQuery 中，工具函数是指直接依附于 jQuery 对象、针对 jQuery 对象本身定义的方法，即全局性的函数，我们统称为工具函数，或 Utilities 函数。它们有一个明显的特征，一般情况下，采用如下的格式进行调用：

```
$.函数名() 或 jQuery.函数()
```

工具函数对应的官方网址是：http://api.jquery.com/category/utilities/

9.2　工具函数的分类

根据工具函数处理对象的不同，可以将其分为下列几大类别：浏览器的检测、数组和对象的操作、字符串操作、测试操作、URL 操作。

9.2.1　浏览器的检测

在浏览器检测中，又可分为浏览器类型与特征的检测，前者获取浏览器的名称或版本信息，后者检测浏览器是否支持标准的 W3C 盒子模型。

1. 浏览器名称或版本信息

虽然 jQuery 有很好的浏览器兼容性，但有时程序开发人员需要获取浏览器的相关信息，用于提供给用户或程序，在 jQuery 中，可以通过访问 $.browser 对象的属性获取。$.browser 对象即 jQuery.browser 对象，用于处理与浏览器相关的事务，该对象的属性如表 9-1 所示。

表 9-1　$.browser 对象的属性

属性名称	说　明
webkit	如果是 webkit 相关的浏览器，则为 true，否则为 false。该属性为 1.4.2 新增
mozilla	如果是 mozilla 相关的浏览器，则为 true，否则为 false
safari	如果是 safari 浏览器，则为 true，否则为 false
opera	如果是 opera 浏览器，则为 true，否则为 false
msie	如果是 IE 浏览器，则为 true，否则为 false
version	获取浏览器的版本号

下面通过一个示例介绍在页面中查看浏览器的相关信息。

示例 9-1 browser 对象的使用

（1）功能描述

根据浏览器的类型，在页面中显示浏览器的名称与版本号。

（2）实现代码

新建一个 HTML 文件 9-1.html, 加入如代码清单 9-1 所示的代码。

代码清单 9-1 browser 对象的使用

```
<!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>使用 browser 对象获取浏览器信息 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "您的浏览器名称是：";
            if ($.browser.msie) { //IE 浏览器
                strTmp += "IE";
            }
            if ($.browser.mozilla) { //火狐相关浏览器
                strTmp += "Mozilla FireFox";
            }
            strTmp += " 版本号是：" //获取版本号
                   + $.browser.version;
            $("#divTip").html(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

其执行后的效果如图 9-1 所示。

2. 盒子模型

盒子模型是 CSS 中的专术名词，用以描述页面设置中的各种属性，如内容（content）、填

充（padding）、边框（border）、边界（margin），由于这些属性拼合在一起，与日常生活中的“盒子”很相像，因此称为盒子模型。

图 9-1　在 IE 8 与火狐下显示浏览器的信息

说明　在 Mozilla FireFox 浏览器中，$.browser.version 获取的是其内核版本号，虽然版本是 3.6.10，但其内核依然是 1.9.2.10。

盒子模型分为二类，一类是 W3C 盒子模型，另一类是 IE 盒子模型。两者最根本的区别在于，属性 Height 与 Width 这两个值是否包含 padding 与 border。W3C 盒子模型不包含 padding 与 border，仅指内容 (content) 的 Height 和 Width，而 IE 盒子模型包含 padding 与 border。W3C 盒子模型示意图如图 9-2 所示。

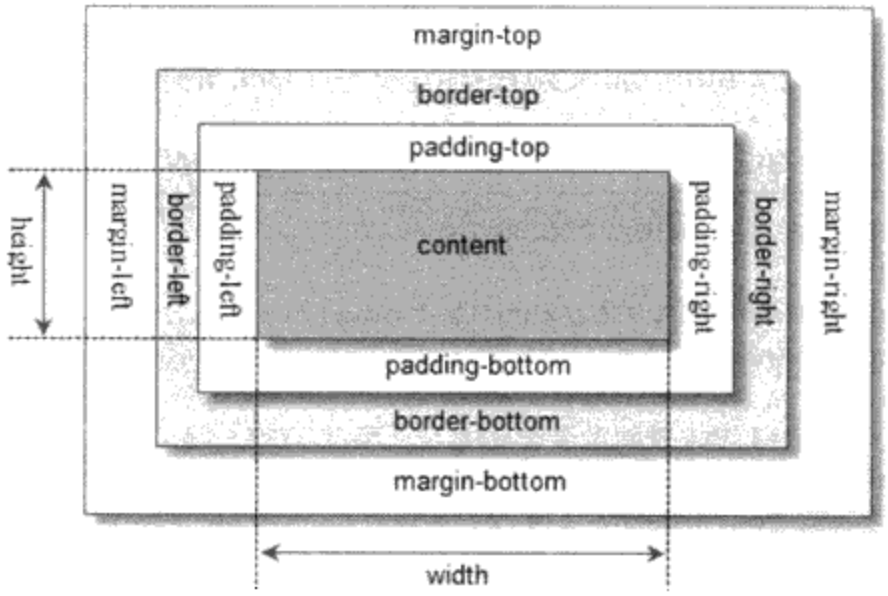


图 9-2　W3C 盒子模型

W3C 盒子模型的 height 和 width 的长度仅包含内容（content）的 height 和 width。而 IE 盒子模型的 height 和 width 的长度则包含 padding 与 border，示意图如图 9-3 所示。

在 jQuery 中，可以通过 jQuery. support .boxModel 对象返回的属性值，确定页面是否是标准的 W3C 盒子模型，其调用的方法如下所示：

```
$.support.boxModel 或 $.support.boxModel
```

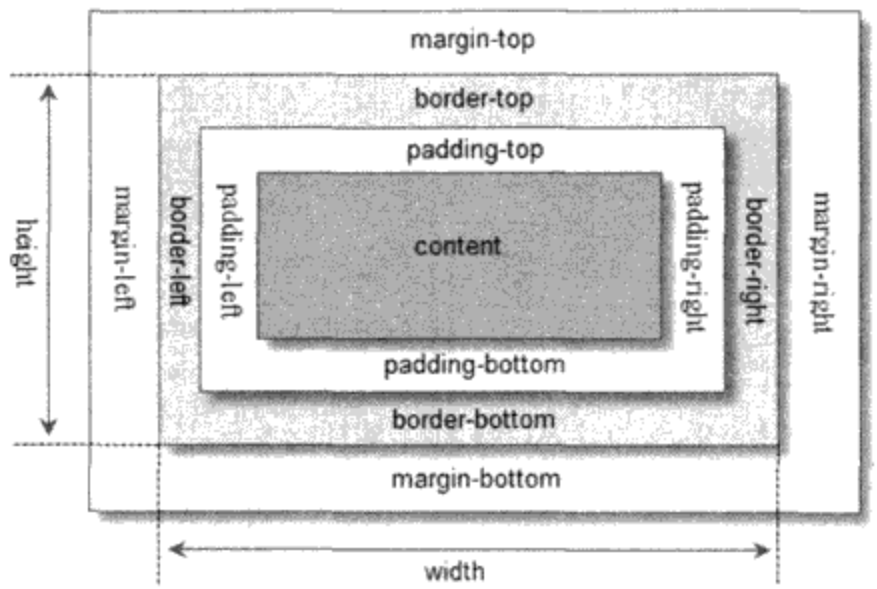


图 9-3　IE 盒子模型

该方法返回一个布尔值，如果是 true 表示是 W3C 盒子模型，否则，则不是 W3C 盒子模型。

下面通过一个示例介绍检测页面是否是 W3C 盒子模型。

示例 9-2　boxModel 对象的使用

（1）功能描述

根据页面的特征，显示是否是标准的 W3C 盒子模型。

（2）实现代码

新建一个 HTML 文件 9-2.html, 加入如代码清单 9-2 所示的代码。

代码清单 9-2　使用 boxModel 对象检测是否是 W3C 盒子模型

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 boxModel 对象检测是否是 W3C 盒子模型 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "您打开的页面是：";
            if ($.support.boxModel) { // 是 W3C 盒子模型
                strTmp += "W3C 盒子模型 ";
```

```
            }
            else { // 是 IE 盒子模型
                strTmp += "IE 盒子模型 ";
            }
            $("#divTip").html(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-4 所示。

图 9-4　W3C 盒子模型

在示例 9-2 的 HTML 代码中，如果删除页面中的下列代码：

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

再重新执行页，其实现的效果如图 9-5 所示。

图 9-5　IE 盒子模型

说明 在编写页面代码时，尽量先声明 DOCTYPE 类型，使用标准的 W3C 盒子模型，避免多个浏览器间的相互不兼容。

9.2.2 数组和对象的操作

1. 遍历数组

使用 $.each() 工具函数，不仅可以实现页面中元素的遍历，还可以完成指定数组的遍历，其调用的语法格式如下：

```
$.each(obj,fn(para1,para2))
```

其中参数 obj，表示要遍历的数组或对象，fn 为每个遍历元素执行的回调函数，该函数包含 2 个参数，para1 表示数组的序号或对象的属性，para2 表示数组的元素和对象的属性。

下面通过一个示例介绍 $.each() 工具函数遍历数组的方法。

示例 9-3 $.each() 函数遍历数组

（1）功能描述

定义一个数组，用于保存学生的相关资料，通过 $.each() 工具函数，获取数组中各元素的名称与内容，显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-3.html, 加入如代码清单 9-3 所示的代码。

代码清单 9-3 使用 $.each() 工具函数遍历数组

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.each() 工具函数遍历数组 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
       body{font-size:13px}
       ul{padding:0px;margin:0px;list-style-type:none;
          width:260px;border-top:dashed 1px #ccc;
          border-left:dashed 1px #ccc;border-right:dashed 1px #ccc}
          ul li{border-bottom:dashed 1px #ccc;padding:8px}
          .title{background-color:#eee; font-weight:bold}
    </style>
    <script type="text/javascript">
        $(function() {
            var arrStu = { "张三：": "60", "李四：": "70", "王二：": "80" }
```

```
            var strContent = "<li class='title'>姓名：分数</li>";
            $.each(arrStu, function(Name, Value) {
                strContent += "<li>" + Name + Value + "</li>";
            })
            $("ul").append(strContent);
        })
    </script>
</head>
<body>
    <ul></ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-6 所示。

图 9-6　使用 $.each() 工具函数遍历数组

2. 遍历对象

$.each() 函数除了遍历数组外，还可以遍历对象，获取对象的属性和值。常用于对一些未知对象的遍历，其使用方法与遍历数组基本一致。下面通过一个示例说明其遍历对象的使用方法。

示例 9-4　$.each() 函数遍历对象

（1）功能描述

通过 $.each() 工具函数，遍历 ajaxSettings 对象，获取该对象全部的属性和值，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-4.html, 加入如代码清单 9-4 所示的代码。

代码清单 9-4　使用 $.each() 工具函数遍历 ajaxSettings 对象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.each() 工具函数遍历 ajaxSettings 对象 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
       body{font-size:13px}
       ul{padding:0px;margin:0px;list-style-type:none;
          width:350px;border-top:dashed 1px #ccc;
          border-left:dashed 1px #ccc;border-right:dashed 1px #ccc}
          ul li{border-bottom:dashed 1px #ccc;padding:8px}
          .title{background-color:#eee; font-weight:bold}
    </style>
    <script type="text/javascript">
        $(function() {
            var strContent = "<li class='title'>属性：值 </li>";
            $.each($.ajaxSettings, function(Property, Value) {
            strContent += "<li>" + Property + ":" + Value + "</li>";
            })
            $("ul").append(strContent);
        })
    </script>
</head>
<body>
    <ul></ul>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-7 所示。

3. 数据筛选

在操作数组时，有时需要根据各种条件筛选元素，传统的 JavaScript 代码将遍历整个数组，在遍历中设置筛选规则，然后，获取符合规则的元素。而在 jQuery 中，可以使用工具函数 jQuery.grep（），很方便地筛选数组中的任何元素。该函数调用的语法格式如下：

```
$.grep(array, function(elementOfArray, indexInArray),[invert])
```

其中，参数 array 为要筛选的原数组，回调函数 fn 中可以设置两个参数，其中 elementOfArray 为数组中的元素，indexInArray 为元素在数组中的序列号；另外，可选项 [invert] 为布尔值，表示是否根据 fn 的规则取反向结果，默认值为 false，表示不取反，如果为 true，表示取反，即返回与回调函数 fn 规则相反的数据。

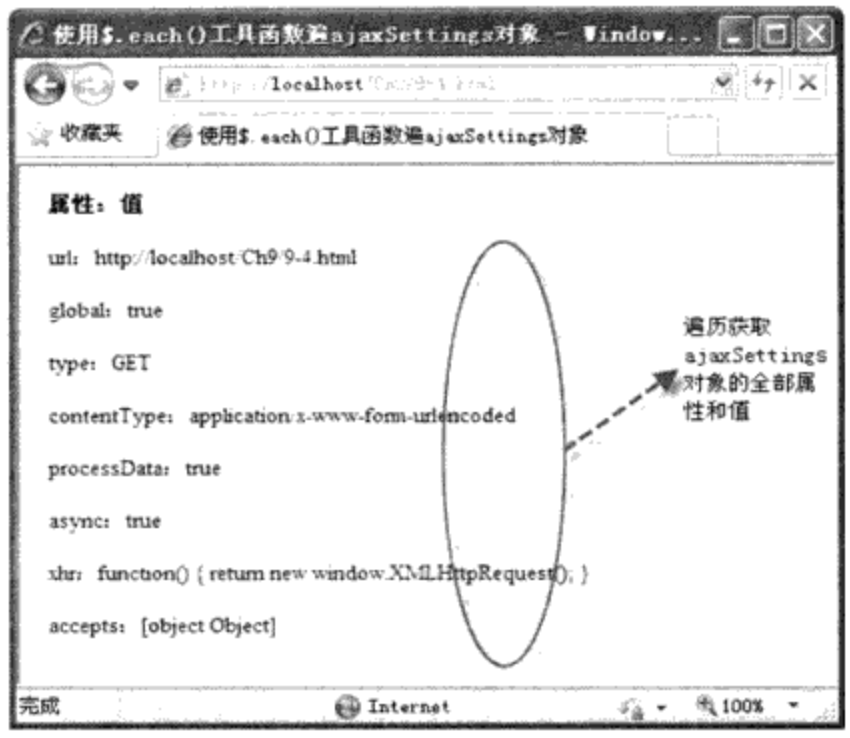


图 9-7　使用 $.each() 工具函数遍历 ajaxSettings 对象

下面通过一个示例介绍工具函数 jQuery.grep 筛选数组中的元素。

示例 9-5　$. grep() 函数筛选数据

（1）功能描述

通过 $.grep() 工具函数，查询数组中大于 5 且序号小于 8 的元素，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-5.html, 加入如代码清单 9-5 所示的代码。

代码清单 9-5　使用 $.grep() 工具函数筛选数组中的元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.grep() 工具函数筛选数组中的元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "筛选前数据：";
            var arrNum = [2, 8, 3, 7, 4, 9, 3, 10, 9, 7, 21];
```

```
            var arrGet = $.grep(arrNum, function(ele, index) {
                return ele > 5 && index < 8 // 元素值大于 5 且序号小于 8
            })
            strTmp += arrNum.join();
            strTmp += "<br/><br>筛选后数据："
            strTmp += arrGet.join();
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-8 所示。

图 9-8　使用 $.grep() 工具函数筛选数组中的元素

说明　在示例 9.5 中，由于数组的序号是从 0 开始的，因此，第 8 位元素是 9，尽管大于 5，符合第一个条件，但它不符合序列号小于 8 的条件，因此，筛选结果中没有出现该元素。

4. 数据变更

虽然可以通过 $.grep() 函数筛选数组中的元素，但如果要按指定条件修改数组中的所选元素，还需调用另外一个工具函数 $.map()，其调用的语法格式如下：

```
$.map(array, callback(elementOfArray, indexInArray)
```

其中，参数 array 为要变更的原数组，回调函数 fn 中可以设置两个参数，其中 elementOfArray 为数组中的元素，indexInArray 为元素在数组中的序列号。

下面介绍在示例 9.5 的基础上，调用 $.map() 函数，实现数组中元素变更的方法。

示例 9-6　$. map() 函数变更数据

（1）功能描述

通过 $.map() 工具函数，将数组中大于 5 且序号小于 8 的元素都增加 3，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-6.html, 加入如代码清单 9-6 所示的代码。

代码清单 9-6　使用 $.map() 工具函数变更数组中的元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.map() 工具函数变更数组中的元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          div{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "变更前数据：";
            var arrNum = [2, 8, 3, 7, 4, 9, 3, 10, 9, 7, 21];
            var arrGet = $.map(arrNum, function(ele, index) {
                if (ele > 5 && index < 8) { // 元素值大于 5 且序号小于 8
                    return ele + 1; // 元素增加 1
                }
            })
            strTmp += arrNum.join();
            strTmp += "<br/><br>变更后数据："
            strTmp += arrGet.join();
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-9 所示。

5. 数据搜索

在 jQuery 中，如果要在数组中搜索某个元素，可以使用工具函数 $.inArray()，该方法相当于用 JavaScript 中的函数 indexOf() 搜索字符串中的某个字符。在工具函数 $.inArray() 中，如果找到了指定的某个元素，则返回该元素在数组中的索引号，否则，返回 −1 值。其调用的

格式如下所示：

```
$.inArray(value, array)
```

其中，参数 value 表示要搜索的对象，array 表示搜索对象的数组。

图 9-9　使用 $.map() 工具函数变更数组中的元素

说明　在使用 $.map() 工具函数前，先通过 if (ele > 5 && index < 8) 语句筛选元素，然后，使用 ele + 1 语句实现每个符合条件元素的更新。

下面通过一个示例介绍如何通过 $.inArray() 函数，在指定的数组中，搜索某个字符的方法。

示例 9-7　$.inArray() 函数搜索数据

（1）功能描述

通过 $.inArray() 函数，在数组 arrStr 中，搜索内容为“2”的第一个匹配元素的位置，并将结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-7.html, 加入如代码清单 9-7 所示的代码。

代码清单 9-7　使用 $.inArray() 工具函数搜索数组中指定元素的位置

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.inArray() 工具函数搜索数组中指定元素的位置 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
```

```
            background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "待搜索数据：";
            var arrNum = [4, 21, 2, 12, 5];
            var arrPos = $.inArray(2, arrNum);
            strTmp += arrNum.join();
            strTmp += "<br/><br>搜索后结果："
            strTmp += arrPos;
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-10 所示。

图 9-10　使用 $.inArray() 工具函数搜索数组中指定元素的位置

说明　由于 $.inArray 函数在搜索过程中，搜索对象匹配的是单个元素，而非元素内容，因此，尽管第 1 号元素“21”中含有“2”，但它不是单个元素，仅是包含搜索对象，因此不匹配。

9.2.3　字符串操作

在 jQuery 中，如果要除掉字符中左右两边的空格符，可以使用工具函数 $.trim()。该函数常用于字符串操作中，也是 jQuery 核心库中唯一一个针对字符串操作的工具函数。其调用的语法格式如下所示：

```
$.trim(str)
```

其中，参数 str 为需要删除左右两边空格符的字符串。

下面通过一个示例介绍 $.trim() 函数除掉空格的方法。

示例 9-8 $.trim() 函数除掉字符串左右两边的空格符

（1）功能描述

通过 $.trim() 函数，除掉一个两边均有空格符的字符串，并将其执行前后的字符长度都显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-8.html, 加入如代码清单 9-8 所示的代码。

代码清单 9-8 使用 $.trim() 函数除掉字符串左右两边的空格符

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.trim() 工具函数除掉字符串的空格符 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:300px}
    </style>
    <script type="text/javascript">
        $(function() {
            var strTmp = "内容：";
            var strOld = "  jQuery,write less do more    ";
            var strNew = $.trim(strOld);
            strTmp += strOld;
            strTmp += "<br/><br>除掉空格符前的长度："
            strTmp += strOld.length;
            strTmp += "<br/><br>除掉空格符后的长度："
            strTmp += strNew.length;
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-11 所示。

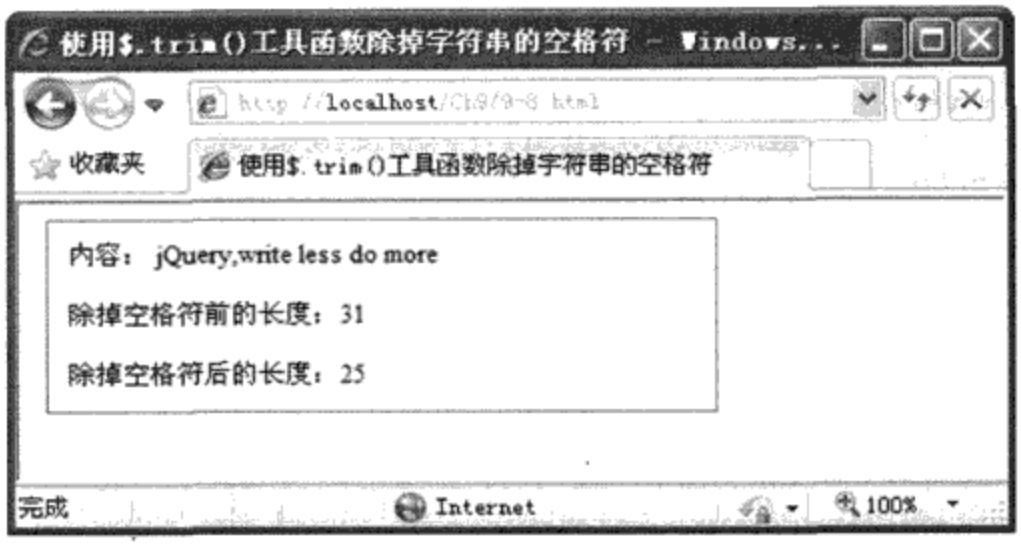


图 9-11　使用 $.trim() 工具函数除掉字符串的空格符

9.2.4　测试操作

在编写代码时，有时需要先获取对象或元素的各种状态，如是否为空、是否是对象等，然后，根据这些状态值，决定程序的下一步操作，因此，相关状态检测的工具函数，在代码编写中也十分重要。jQuery 1.4.2 版本中，有 5 个涉及状态检测的工具函数，其详细信息见表 9-2。

表 9-2　jQuery 中测试工具函数

函数名称	说　明	支持版本
$.isArray(obj)	返回一个布尔值，检测参数 obj 是否是一个数组对象，如果为 true，则表示是，否则，则表示不是	1.3 以上
$.isFunction(obj)	返回一个布尔值，检测参数 obj 是否是一个函数，如果为 true，则表示是，否则，则表示不是	1.2 以上
$.isEmptyObject(obj)	返回一个布尔值，检测参数 obj 是否是一个空对象，如果为 true，则表示是，否则，则表示不是	1.4 以上
$.isPlainObject(obj)	返回一个布尔值，检测参数 obj 是否是一个纯粹对象，如果为 true，则表示是，否则，则表示不是	1.4 以上
$.contains(container, contained)	返回一个布尔值，检测一个 DOM 节点是否包含另一个 DOM 节点，如果为 true，则表示是，否则，则表示不是	1.4 以上

下面通过 3 个示例，介绍在 1.4 版本中新增检测工具函数的使用方法。

示例 9-9　$.isEmptyObject() 函数的使用

（1）功能描述

通过 $.isEmptyObject() 函数，检测某个指定的对象是否为空，并将结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-9.html, 加入如代码清单 9-9 所示的代码。

代码清单 9-9　使用 $.isEmptyObject() 函数检测对象是否为空

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.isEmptyObject() 函数检测对象是否为空 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var obj0 = {};
            var obj1 = { "name": "taoguorong" };
            var strTmp = "obj0 是否为空：" + $.isEmptyObject(obj0);
            strTmp += "<br><br>obj1 是否为空："
                   + $.isEmptyObject(obj1);
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-12 所示。

图 9-12　使用 $.isEmptyObject() 函数检测对象是否为空

说明　因为对象 obj0 的内容为 {}，obj1 的内容为 {"name": "taoguorong"}，因此，$.isEmptyObject() 函数检测 obj0 时，返回 true，检测 obj1 时，返回 false。

示例 9-10　$.isPlainObject () 函数的使用

（1）功能描述

通过 $. isPlainObject() 函数，检测某个指定的对象是否为原始对象，即对象是否通过 {} 或 new Object() 关键字创建，并将结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-10.html, 加入如代码清单 9-10 所示的代码。

代码清单 9-10　使用 $.isPlainObject() 函数检测对象是否为原始对象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.isPlainObject() 函数检测对象是否为原始对象 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var obj0 = {};
            var obj1 = new Object();
            var obj2 = "null";
            var strTmp = "obj0 是否为原始对象："
                + $.isPlainObject(obj0);
            strTmp += "<br><br>obj1 是否为原始对象："
                + $.isPlainObject(obj1);
            strTmp += "<br><br>obj2 是否为原始对象："
                + $.isPlainObject(obj2);
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-13 所示。

图 9-13　使用 $.isPlainObject() 函数检测对象是否为原始对象

说明　在使用 $.isPlainObject() 函数检测对象是否为原始对象时，由于 obj1 定义时为 new Object()，因此，该对象为原始对象；如果 obj1 定义时带参数，如 new Object("name")，这时当传递参数生成对象时，构造器不一定指向 Object，因此，$.isPlainObject() 函数检测时，将返回 false。

示例 9-11　$.contains() 函数的使用

$.contains() 函数用于检测在一个 DOM 节点中是否包含另外一个 DOM 节点，其调用的语法格式如下：

```
$.contains(container, contained)
```

其中：参数 container 为 Object，是一个 DOM 元素，作为容器，可以包容其他 DOM 元素；参数 contained 也是一个 DOM 是一个节点，可能被其他元素所包含。整个函数返回一个布尔值，如果 container 对象包含 contained 对象，则结果为 true，否则，结果为 false。

（1）功能描述

通过 $. contains() 函数，检测指定的两个节点之间是否存在包含的关系，并将结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-11.html, 加入如代码清单 9-11 所示的代码。

代码清单 9-11　使用 $.contains() 函数检测两个节点是否存在包含的关系

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.contains() 函数检测两个节点是否包含 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
```

```
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
    <script type="text/javascript">
        $(function() {
            var node0 = document.documentElement;
            var node1 = document.body;
            var strTmp = "对象 node0 是否包含对象 node1："
            strTmp += $.contains(node0, node1);// 检测两者的包含关系
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-14 所示。

图 9-14　使用 $.contains() 函数检测两个节点是否包含

说明　在 DOM 中，documentElement 是文档的根节点，而 body 是根节点下的一个子节点，变量 node0 代表 documentElement，node1 代表 body。因此，如果执行 $.contains(node0, node1)，则返回 true，反之，则返回 false。

9.2.5　URL 操作

在前面 6.2.3 节 serialize() 序列化表单中，我们曾介绍过使用 serialize() 方法来序列化表单向服务器提交的数据，即 URL 操作。而 serialize() 方法的核心，则是工具函数 $.param()，通过该函数可以使数组或 jQuery 对象按照 name/value 的格式进行序列化，普通对象按照 key/value 的格式进行序列化。工具函数 $.param() 调用的语法格式如下所示。

```
$.param(obj, [traditional])
```

其中：参数 obj 表示需要进行序列化的对象，该对象可以是数组、jQuery 元素、普通对象；可选项参数 [traditional]，表示是否使用普通的方式浅层序列化，该函数返回一个序列化后的字符串。

下面通过一个完整的示例，介绍如何调用工具函数 $.param() 进行数组元素序列化的方法。

示例 9-12　使用函数 $.param() 对数组进行序列化

（1）功能描述

定义两个不同内容的数组：第一个数组 arrInfo 用于保存基本信息，如 ID 号、名称；第二个数组 arrScore 用于保存分数和汇总信息，内容见代码清单。使用函数 $.param() 分别对这两个数组进行序列化，使其各自成为可以执行传值的 URL，并将该 URL 分别显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-12.html, 加入如代码清单 9-12 所示的代码。

代码清单 9-12　使用函数 $.param() 对数组进行序列化

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 $.param() 进行数组元素序列化 </title>
<script type="text/javascript"
        src="Jscript/jquery-1.4.2.js">
</script>
<style type="text/css">
       body{font-size:13px}
       div{margin:5px;padding:10px;border:solid 1px #666;
           background-color:#eee;width:260px}
</style>
<script type="text/javascript">
    $(function() {
        //基本信息数组
        var arrInfo = { id: 101, name: "tao", sex: 0 };
        //分数和汇总信息数组
        var arrScore = { Score: { chinese: 90,
                                  maths: 100,
                                  english: 98 },
            SunNum: { Score: 288,
                      Num: 3 }
        };
        //序列化各数组
        var arrNewInfo = $.param(arrInfo);
        var arrNewScore = $.param(arrScore);
        var arrDecScore = decodeURIComponent(
```

```
                                $.param(arrScore));
            // 显示序列化后的数组
            var strTmp = "<b>arrInfo 数组序列化后 </b>：";
            strTmp += arrNewInfo;
            strTmp += "<br><br><b>arrScore 数组序列化后 </b>：";
            strTmp += arrNewScore;
            strTmp += "<br><br><b>arrScore 序列化解码后 </b>：";
            strTmp += arrDecScore;
            // 显示在页面中
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-15 所示。

图 9-15　使用 $.param() 进行数组元素序列化

说明　为了更加清晰展示 URL 在操作过程中每个参数对应值的状况，我们采用 ecodeURIComponent() 方法，对序列化后的数组 arrScore 的 URL 进行解码，通过解码后展示在页面中的字符串，可以很清楚地知道各参数传递时对应的值。

9.3　工具函数的扩展

工具函数的扩展，其实质就是自己编写类级别的插件，用于扩展 jQuery 对象本身。为实现这一目的，我们需要引入另外一个工具函数 $.extend()，通过该函数，可以很方便地定义自

己的工具函数。下面通过一个示例介绍函数 $.extend() 扩展函数的过程。

示例 9-13　使用函数 $.extend() 扩展工具函数

（1）功能描述

编写两个自定义的工具函数，一个用于返回两个数中最大值，另一个用于返回两个数中最小值；并在页面中进行调用，将返回的结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 9-13.html, 加入如代码清单 9-13 所示的代码。

代码清单 9-13　使用函数 $.extend() 扩展工具函数

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用函数 $.extend() 扩展工具函数 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
    </style>
<script type="text/javascript">
    /*-------------------------------------------------------------/
    功能：返回两个数中最大值
    参数：数字 p1,p2
    返回：最大值的一个数
    示例：$.MaxNum(1,2);
    /-------------------------------------------------------------*/
    ; (function($) {
        $.extend({
            "MaxNum": function(p1, p2) {
                return (p1 > p2) ? p1 : p2;
            }
        });
    })(jQuery);

    /*-------------------------------------------------------------/
    功能：返回两个数中最小值
    参数：数字 p1,p2
    返回：最小值的一个数
    示例：$.MinNum(1,2);
    /-------------------------------------------------------------*/
    ; (function($) {
        $.extend({
```

```
                "MinNum": function(p1, p2) {
                    return (p1 > p2) ? p2 : p1;
                }
            });
        })(jQuery);

        $(function() {
            var strTmp = "5 与 6 中最大的数是：";
            strTmp += $.MaxNum(5, 6);
            strTmp += "<br><br>7 与 8 中最小的数是：";
            strTmp += $.MinNum(7, 8);
            $("#divTip").append(strTmp);
        })
    </script>
</head>
<body>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-16 所示。

图 9-16　使用函数 $.extend() 扩展工具函数

（4）代码说明

工具函数 $.extend() 除可以扩展 jQuery 自身函数外，还有另外一个很强悍的功能，就是可以用于扩展已有的 Object 对象，其调用的语法格式为：

```
$.extend(target, object1,... [objectN])
```

其中参数 target 表示合并后的对象，object 为被合并对象，即可以将一个或多个对象合并成一个对象，最后返回该对象，如执行下列代码：

```
var objName = { name: "张三", sex: "男" };
```

```
var objInfo = { name: "李四", age: 30 };
var objLast = $.extend(objName, objInfo);
```

其最后返回的结果是：

```
objLast ={ name: "李四", sex: "男", age: 30 };
```

从返回结果不难看出，在 $.extend() 函数中，如果是合并对象，且存在相同参数的名称，那么，后面对象中的参数值将覆盖前面对象中的参数值。

说明 有关使用 $.extend() 开发插件的更多详细介绍，请参考第 7 章中的 7.9.3 节。

9.4 其他工具函数——$.proxy()

在 jQuery 中，除了上述章节中介绍的各种工具函数外，还有很多不错的函数，尤其在 jQuery 1.4.2 版本中，新增了很多简洁、高效的工具函数，其中，新增函数 $.proxy() 在处理不同作用域对象事件中相当实用。

$.proxy() 函数返回一个新的函数，并且这个函数始终保持特定的作用域。所谓的作用域，就是执行该函数对象的范围。当一个事件函数被元素绑定时，其作用域原则上应指向该元素，但有些事件函数的作用域，在被元素绑定时并不指向元素本身，而是指向另外一个对象。这时，为了使元素的绑定事件能正常执行，必须调用函数 $.proxy() 进行处理。经过处理后的事件函数，不仅可以被绑定的元素执行，而且还可以传递原先函数取消事件的绑定，该函数调用的语法格式为：

```
$.proxy(function,scope)
```

其中，参数 function 为要改变作用域的事件函数，参数 scope 为被事件函数设置作用域的对象，即事件函数的作用域将设置到该对象中。

该函数还有另外一种调用的语法格式，代码如下：

```
$.proxy(scope, name)
```

其中，参数 scope 为被事件函数设定的作用域对象，参数 name 为将要设置作用域的函数名，并且该参数必须是 scope 作用域对象的一个属性。

下面通过一个详细的示例介绍 $.proxy() 在页面中的应用。

示例 9-14 使用函数 $.proxy() 改变事件函数的作用域

（1）功能描述

强制性设置事件函数的作用域，使 this 指向对象函数 objMyInfo，而不是被绑定的元素对象 #Button1，但又可以执行对象 objMyInfo 中的 ShowEvent 事件，将设置的个人信息展示在页面中。

（2）实现代码

新建一个 HTML 文件 9-14.html, 加入如代码清单 9-14 所示的代码。

代码清单 9-14　使用函数 $.proxy() 改变事件函数的作用域

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用函数 $.proxy() 改变事件函数的作用域 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px;padding:10px;border:solid 1px #666;
               background-color:#eee;width:260px}
           input{margin:5px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
        $(function() {
            var objMyInfo = {
                name: "陶国荣",// 设置对象 name 属性.
                sex: "男", // 设置对象 sex 属性
                ShowEvent: function() { // 设置执行的事件
                    $("#divShow").html("姓名：" +
                     this.name + "<br><br>性别：" +
                     this.sex);
                }
            }
            $("#Button1").bind("click", // 通过 proxy 函数绑定设置的事件
            $.proxy(objMyInfo.ShowEvent, objMyInfo));
        })
    </script>
</head>
<body>
   <input id="Button1" type="button" value="显示" class="btn" />
   <div id="divShow"></div>
</body>
</html>
```

（3）页面效果

代码执行后的效果如图 9-17 所示。

（4）代码分析

在上述示例 9.14 的 JS 代码中，如果不使用 $.proxy() 函数改变事件函数的作用域，而是直接将对象的事件函数与执行元素进行绑定，即将下面代码：

```
$("#Button1").bind("click", // 通过 proxy 函数绑定设置的事件
```

```
    $.proxy(objMyInfo.ShowEvent, objMyInfo));
```

修改成如下代码：

```
$("#Button1").bind("click", // 直接绑定对象函数的事件
  objMyInfo.ShowEvent);
```

图 9-17　使用函数 $.proxy() 改变事件函数的作用域

执行后的页面效果如图 9-18 所示。

图 9-18　直接绑定对象函数的事件

从示意图中可以很明显地看出，在单击按钮时，由于传递的 this 作用域与事件函数中 this 的作用域不同，因此，无法访问事件函数中的 name 与 sex 属性，所以提示为“underfined”出错信息。同时，下列代码：

```
$("#Button1").bind("click", // 通过 proxy 函数绑定设置的事件
  $.proxy(objMyInfo.ShowEvent, objMyInfo));
```

等价于如下代码：

```
$("#Button1").bind("click", // 通过 proxy 函数绑定设置的事件
$.proxy(objMyInfo, "ShowEvent"));
```

两段代码只是调用的方式不同，实现的功能是相同的。

9.5　综合案例分析——使用 jQuery 扩展工具函数实现对字符串指定类型的检测

在 jQuery 中，仅有一个对字符串操作的全局函数——$.trim()。该函数可以删除指定字符串的前后空格，在前面的章节 9.2.3 中已作了详细的说明。而在实际的案例开发过程中，常常需要验证输入字符类型的有效性，如：输入“邮箱”时，需要验证用户输入的内容是否符合邮箱的格式等。下面介绍通过使用 jQuery 扩展工具开发一个全局函数 chkStrByType()，该函数可以根据指定的字符验证类型，检测待验证的字符串是否符合该类型的格式。

9.5.1　需求分析

1）在文本框中输入任意一个字符，然后，选择下拉列表框中的某个字符类型，单击“检测”按钮后，将检测文本框中的字符是否符合选择类型的格式。

2）将检测后的结果显示在页面中。

9.5.2　效果界面

在页面的文本框中，输入“陶国荣”三个字符，并选择“汉字”类型后，单击“检测”按钮，将在页面中显示检测后的结果，其实现的界面如图 9-19 所示。

图 9-19　验证通过时的页面效果

如果在文本框中输入的字符不变，而将选择类型改为“邮政编码”，那么单击“检测”按钮后，将显示字符与所选类型不符的提示结果，其实现的界面如图 9-20 所示。

图 9-20　验证失败时的页面效果

9.5.3　功能实现

为了实现字符串指定类型的验证并显示验证结果的功能，新建一个 HTML 页面 chkStr.html，加入如代码清单 9-15 所示的代码。

代码清单 9-15　使用 jQuery 扩展工具函数实现对字符串指定类型的检测

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 扩展工具函数实现对字符串指定类型的检测 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:11px}
           fieldset{width:410px}
           fieldset div{padding:8px}
           fieldset div select{font-size:9pt;padding:1px}
           #divTip{margin-top:10px;padding:10px;
           border:solid 1px #666;
           background-color:#eee;width:210px;display:none}
           .txt{border:#666 1px solid;padding:2px;width:120px;
           margin-right:3px}
           .btn {border:#666 1px solid;padding:2px;width:60px;
           filter: progid:DXImageTransform.Microsoft
           .Gradient(GradientType=0,StartColorStr=#ffffff,
           EndColorStr=#ECE9D8);}
    </style>
    <script type="text/javascript">
```

```
/*-------------------------------------------------------------------/
功能：返回检测字符串指定类型的结果
参数：checkType 为检测字符串的类型 ;strS 为待检测的字符串
返回：一个 bool 值 ,true 表示是指定的类型,
                false 表示不是指定的字符类型
示例：$.chkStrByType(" 陶国荣 ", "zh_cn");
/-------------------------------------------------------------------*/
 ; (function($) {
     $.extend({
         chkStrByType: function(strS, chkType) {
             var result;
             switch (chkType) {
                 case 'odd'://奇数型
                     var chkStr = arrRegExp['number'];
                     var reg = RegExp(chkStr, 'g');
                     var result = reg.test(strS);
                     if (true == result) {
                         var num = parseInt(strS) % 2;
                         if (1 == num) {
                             result = true;
                         } else {
                             result = false;
                         }
                     } else {
                         result = false;
                     }
                     break;
                 case 'even'://偶数型
                     var chkStr = arrRegExp['number'];
                     var reg = RegExp(chkStr, 'g');
                     var result = reg.test(strS);
                     if (true == result) {
                         var num = parseInt(strS) % 2;
                         if (num == 0) {
                             result = true;
                         } else {
                             result = false;
                         }
                     } else {
                         result = false;
                     }
                     break;
                 default://其他类型按正则表达式检测
                     var chkStr = arrRegExp[chkType];
                     var reg = RegExp(chkStr, 'g');
                     var result = reg.test(strS);
                     break;
             }
             return result;
         }
```

```
    });
    /*  正则验证字符串表达式  */
    var arrRegExp = {};
    arrRegExp['email'] = '\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*';
    arrRegExp['telephone'] = '(\\(\\d{3,4}\\)|\\d{3,4}-|\\s)?\\d{7,8}';
    arrRegExp['mobile'] = '(86)*0*1[3,5]\\d{9}';
    arrRegExp['postcode'] = '^\\d{6}$';
    arrRegExp['number'] = '^-?[0-9]\\d*$';
    arrRegExp['zh_cn'] = '[\\u4e00-\\u9fa5]';
    arrRegExp['url'] = '[a-zA-z]+://[^\\s]*';
})(jQuery);

$(function() {
    $("#btnChkStr").click(function() {
        // 获取待检测的字符串与指定的类型
        var $ChkStr = $("#txtChkStr").val();
        var $ChkType = $("#selStrType").val();
        // 保存检测后的结果值
        var blnResult = $.chkStrByType($ChkStr, $ChkType);
        // 返回检测后的结果
        var strTmpShow = "";
        var strTmpType = blnResult ? " 是 " : " 不是 ";
        strTmpShow = $ChkStr + strTmpType;
        strTmpShow = strTmpShow
                   + $("select :selected").text();
        strTmpShow = strTmpShow + " 类型 ";
        // 将返回后的结果显示在页面中
        $("#divTip").show().html("").append(strTmpShow);
        });
    });
</script>
</head>
<body>
  <fieldset><legend> 指定类型检测字符串 </legend>
    <div>
     <span> 检测内容：</span>
     <input id="txtChkStr" type="text" class="txt" />
     <span> 选择类型：</span>
     <select id="selStrType">
      <option value="email"> 邮箱 </option>
      <option value="telephone"> 电话号码 </option>
      <option value="mobile"> 手机号码 </option>
      <option value="postcode"> 邮政编码 </option>
      <option value="number"> 整数 </option>
      <option value="zh_cn"> 汉字 </option>
      <option value="url"> 网址 </option>
      <option value="odd"> 奇数 </option>
      <option value="even"> 偶数 </option>
     </select>
     <input id="btnChkStr" type="button"
```

```
                value=" 检测 " class="btn" />
          <div id="divTip"></div>
         </div>
      </fieldset>
</body>
</html>
```

9.5.4　代码分析

在本案例中，首先，将执行的代码使用 $.extend() 方法进行封装。这样可以将代码独立成为一个功能性的插件，用户只需导入该插件，通过使用 $. 方法名 () 就可以进行调用。其封装框架代码如下所示：

```
; (function($) {
     $.extend({
       chkStrByType: function(strS, chkType) {
          //... 执行代码
       }
       })
 })(jQuery);
```

在执行代码中，根据所传回的字符串类型格式，分别验证传回的字符串，当验证通过时，返回 true 值，否则，返回 false 值，其部分代码如下所示：

```
//... 省略部分代码
var result;
switch (chkType) {
   case 'odd':// 奇数型
      //... 验证字符是否为奇数类型
     break;
   case 'even':// 偶数型
      //... 验证字符是否为偶数类型
      break;
   default:// 其他类型按正则表达式检测
      //... 结合正则表达式验证相应类型
      break;
  }
   return result;
//... 省略部分代码
```

在本案例中，为了验证字符串的指定类型格式，先定义了一个数组，保存各种类型验证的正则表达式，其实现的代码如下所示：

```
//... 省略部分代码
/*  正则验证字符串表达式  */
var arrRegExp = {};
arrRegExp['email'] = '\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*';
arrRegExp['telephone'] = '(\\(\\d{3,4}\\)|\\d{3,4}-|\\s)?\\d{7,8}';
```

```
arrRegExp['mobile'] = '(86)*0*1[3,5]\\d{9}';
arrRegExp['postcode'] = '^\\d{6}$';
arrRegExp['number'] = '^-?[0-9]\\d*$';
arrRegExp['zh_cn'] = '[\\u4e00-\\u9fa5]';
arrRegExp['url'] = '[a-zA-z]+://[^\\s]*';
//...省略部分代码
```

如果需要验证其他的字符串格式，只要将验证的正则表达式加入到该数组中即可，通过数组保存字符类型正则表达式后，就可以在执行代码中调用该数组中的某个元素值，从而获取某类型的正则表达式。

例如：在验证是否是奇数时，先调用数组元素 arrRegExp['number']，获取验证是否为整数的正则表达式，然后，通过 RegExp 对象与正则表达式进行匹配，并将匹配的结果通过 RegExp 对象的 test() 方法进行检测，返回检测结果 true 或 false 值，如果返回值为 true，则将字符串通过 parseInt() 方法转成整数后与 2 求余数，如果有余数，则为奇数，否则，为偶数。其实现的部分代码如下所示：

```
//...省略部分代码
var chkStr = arrRegExp['number'];
var reg = RegExp(chkStr, 'g');
var result = reg.test(strS);
if (true == result) {
    var num = parseInt(strS) % 2;
    if (1 == num) {
        result = true;
    } else {
        result = false;
    }
} else {
  result = false;
}
//...省略部分代码
```

如果只要根据传回的类型，调出数组中类型的正则表达式，就可以检测字符串是否属于该类型，则执行下列代码：

```
//...省略部分代码
var chkStr = arrRegExp[chkType];
var reg = RegExp(chkStr, 'g');
var result = reg.test(strS);
//...省略部分代码
```

其中，参数 chkType 为调用 chkStrByType 函数时，输入的参数类型。

说明 在插件的执行代码中，RegExp 是 JavaScript 中的一个重要对象，该对象可以对字符串与正则表达式进行模式匹配，格式为 new RegExp(pattern, attributes)，其中参数 pattern 为正则表达式，参数 attributes 有三个值，“g”、“i”、“m”，分别表示全局、区分大小写、多行匹配。

9.6　本章小结

在 jQuery 中，众多实用的工具函数为程序开发人员开发程序带来方便，这也正是 jQuery 被众多爱好者热衷于的原因之一。本章从工具函数的概念与分类讲起，旨在通过一些简洁明了的应用示例，使读者能够快速上手使用这些常用的工具函数，并能够做到举一反三，开发出自己的工具函数，提高代码开发效率。对于 jQuery 1.4 新增工具函数，除了理解已经介绍的基础概念外，还要不断练习，才能真正掌握其实质。

第 10 章

jQuery 性能优化与最佳实践

本章内容

- 优化选择器执行的速度
- 处理选择器中的不规范元素标志
- 优化事件中的冒泡现象
- 使用 data() 方法缓存数据
- 解决 jQuery 库与其他库的冲突
- 使用子查询优化选择器性能
- 减少对 DOM 元素直接操作
- 正确区分 DOM 对象与 jQuery 对象
- 本章小结

jQuery 是继 prototype 后的又一个优秀的 JavaScript 框架，通过这个简洁、快速、灵活的框架，可以帮助我们在页面中实现文档操作、事件处理、Web 交互这样一些基本的事务。本章将介绍 jQuery 在日常开发中的一些应用技巧和注意事项，帮助开发者在编写代码时少走弯路，并提高代码执行的效率。

10.1　优化选择器执行的速度

由于选择器的使用在 jQuery 中占据十分重要的位置，优化选择器的执行速度，其实质就是优化代码的执行速度，从而加快页面的浏览速度。在 jQuery 中，为了提升选择器获取 DOM 元素的速度，笔者建议从下列几方面入手，结合自己的开发实际，提升代码质量。

10.1.1　优先使用 ID 与标记选择器

在 jQuery 中，访问 DOM 元素的最快方式是通过元素 ID 号，其次是通过元素的标记。因为前者源于 JavaScript 中的 document.getElementById()，而后者源于 document.getElementsByTagName() 方法，有一个页面，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>测试</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           .MyClass{padding:5px;margin:5px}
    </style>
</head>
<body>
    <div id="divTip" class="MyClass" title="tmp">测试文字</div>
</body>
</html>
```

有下列三种方法可以访问页面中的 元素，代码如下：

```
var eleName0 = $("#divTip");
var eleName1 = $("div");
var eleName2 = $(".MyClass");
```

在使用代码访问元素时，如果有 ID 号，建议使用 ID 号直接访问元素，这样的速度是最快的；如果没有 ID 号，可使用元素标记 (tag) 访问，其次就是使用类别 (class) 进行访问。

在 jQuery 中访问页面元素时，应尽量避免出现下列的访问语法，说明如下：

1）虽然用 ID 号访问页面中的元素最快，但应避免重复修饰，即避免使用 ID 号修饰 ID 号，错误代码如下所示：

```
var eleName0 = $("#divTip #divShow");
```

2）避免使用 tag 或 class 来修改 ID 号，这样的话，代码先执行遍历，后进行匹配，错误代码如下所示：

```
var eleName0 = $(".MyClass #divShow");
或
var eleName0 = $("div #divShow");
```

3）如果是通过元素的的属性访问，应尽量使用 tag 修饰进行访问，这样可以加快访问速度，正确代码如下：

```
var eleName3 = $("div[title='tmp']");
```

10.1.2 使用 jQuery 对象缓存

所谓对象缓存，就是在使用 jQuery 对象时，先尽量使用变量保存对象名，然后，通过变量进行相应的方法操作。如下列代码是欠妥的：

```
$("#divTip").bind("click", function() { alert("hello!"); })
$("#divTip").css("width", "200px");
$("#divTip").css("background-color", "red");
```

比较优化的代码是：

```
var objTmp = $("#divTip;") // 先使用变量进行保存
objTmp.bind("click", function() { alert("hello!"); })
objTmp.css("width", "200px");
objTmp.css("background-color", "red");
```

如果想让定义的变量在其他函数中也能使用，可以将该变量定义为全局性的变量，实现代码如下：

```
window.objPub = { // 在全局范围中，定义一个 windows 对象
        objTmp: $("#divTip")
 }
```

那么，通过全局的变量，可以在各个自定义的函数中访问该变量。通过变量，实现 DOM 元素的获取，实现代码如下：

```
function TestFun() {
   objPub.objTmp.bind("click", function() { alert("hello!"); })
   objPub.objTmp.css("width", "200px");
   objPub.objTmp.css("background-color", "red");
 }
```

以上代码最终实现的功能，与定义局部变量一样，但它却可以被不同的自定义函数所调

用，也可以当成普通的 jQuery 对象使用。

在使用变量缓存 jQuery 对象时，有以下两个地方需要我们在编写代码时注意。

1）无论是局部还是全局性的变量，为了避免与其他变量名相冲突，原则上，请尽量使用“$”符号进行命名，代码如下所示：

```
window.$objPub = { // 在全局范围中，定义一个 windows 对象
        $objTmp: $("#divTip")
 }
```

调用全局变量时，修改后的代码如下：

```
function TestFun() {
   $objPub.$objTmp.bind("click", function() { alert("hello!"); })
   $objPub.$objTmp.css("width", "200px");
   $objPub.$objTmp.css("background-color", "red");
}
```

2) 如果在同一个 DOM 对象中有多个对象的操作，应尽量采用链接式的写法优化调用的代码，因此，上述调用全局变量的代码最后可修改成下列代码：

```
function TestFun() {
   $objPub.$objTmp.bind("click", function() { alert("hello!"); })
   .css({ "width": "200px", "background-color": "red" });
}
```

10.1.3　给选择器一个上下文

在 jQuery 中，DOM 元素的查找可以通过 $(element) 方法实现。除此之外，还可以通过 $(expression, [context]) 方法，在指定的范围内查找某个 DOM 元素，这个方法的优势在于，缩小定位元素的范围，比一般的元素定位更快捷，效果更好。其调用的语法格式如下：

```
$(expression, [context])
```

其中，参数 expression 为需要查找的字符串，可选项 [context] 为等待查找的 DOM 元素集、文档或 jQuery 对象。

下面通过一个示例说明 $(expression, [context]) 方法与一般元素查找方法 $(elements) 的使用。

示例 10-1　在指定的查找范围内获取 DOM 元素

（1）功能描述

定义两个全局变量，其 $objTmp0 通过 $(expression, [context]) 方法获取 DOM 元素 div0，另外一个变量 $objTmp0 通过 $(element) 方法获取元素 div1，然后，自定义一个名为 TestFun 的函数，该函数的作用是通过定义的全局变量，设置 DOM 元素的内容，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-1.html, 加入如代码清单 10-1 所示的代码。

代码清单 10-1　在指定的查找范围内获取 DOM 元素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>在指定的查找范围内获取 DOM 元素 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .MyCls0{border:solid 1px #666;margin:5px;width:200px}
          .MyCls1{border:solid 1px #ccc;margin:5px;width:200px}
          .MyCls{background-color:#eee;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            window.$objPub = { // 在全局范围中，定义一个 windows 对象
                $objTmp0: $("#div0", ".MyCls0"),
                $objTmp1: $("#div1")
            }
            TestFun();
        })
        function TestFun() { // 自定义显示 div 内容的函数
            $objPub.$objTmp0.html("Tmp0");
            $objPub.$objTmp1.html("Tmp1");
        }
    </script>
</head>
<body>
    <div class="MyCls0">
        <div id="div0" class="MyCls" title="tmp0"></div>
    </div>
    <div class="MyCls1">
        <div id="div1" class="MyCls" title="tmp1"></div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-1 所示。

说明　从图 10-1 可以看出，虽然两种不同的方法都获取了 DOM 元素，并设置了其显示的内容，但从执行效果来说，代码 $objTmp0: $("#div0", ".MyCls0") 的执行速度明显优越于代码 $objTmp1: $("#div1")，因此，应尽量使用 $(expression, [context]) 方法访问 DOM 元素。

图 10-1　在指定的查找范围内获取 DOM 元素

10.2　处理选择器中的不规范元素标志

在 10.1.1 节中，我们介绍了如何优化选择器访问 DOM 元素的速度，但有时在自动生成的页面中有很多元素，其 ID 号并不是很规范的标记，常常含有特殊的符号或空格。下面将介绍像这样的 DOM 元素如何通过选择器获取。

10.2.1　选择器中含有特殊符号

在页面中，根据 W3C 的规定，元素的属性中不能包含类似于“#”、“(”、“[”等不规范的特殊字符，但在自动生成的有些页面的属性中，不可避免地出现了上述特殊符号。针对这样的元素，如果我们采用常规的方法，通过选择器获取元素，必然会报错，如下列 HTML 页面中的元素代码：

```
<div id="div#2#" class="MyCls" title="tmp1"></div>
<div id="div[3]" class="MyCls" title="tmp2"></div>
<div id="div(4)" class="MyCls" title="tmp3"></div>
```

采用下列代码方式获取元素，也不能成功获取，代码如下：

```
$("#div#2#").html("Tmp2");
$("#div[3]").html("Tmp3");
$("#div(4)").html("Tmp4");
```

为了正确获取这些属性中含有特殊字符的 DOM 元素，必须使用在特殊字符前添加转义符“\\”的方法。添加转义符后的代码如下：

```
$("#div\\#2\\#").html("Tmp2");
$("#div\\[3\\]").html("Tmp3");
$("#div\\(4\\)").html("Tmp4");
```

通过采用在特殊字符前添加转义符“\\”的方法，可以正确获取这些元素。

10.2.2 选择器中含有空格符号

在元素的属性中除了含有特殊符号外，有时还会含有空格符。空格符在元素的属性中虽然不起眼，但如果我们在编写通过选择器获取元素的代码时，选择器中含有空格符与不含有空格符将会出现两种完全不同的结果。下面通过一个示例介绍二者的差别。

示例 10-2 选择器中含有空格符与不含空格符的区别

（1）功能描述

分别用选择器获取含有空格符或不含有空格符的 DOM 对象，并用两个变量保存，然后分别计算两个对象的总个数，即 length 值，显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-2.html, 加入如代码清单 10-2 所示的代码。

代码清单 10-2 选择器中含有空格符与不含空格符的区别

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>选择器中含有空格符与不含空格符的区别</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
          .frame{margin:5px;padding:10px;border:solid 1px #666;
              background-color:#eee;width:300px}
          .MyCls{padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            var $objTmp0 = $(".MyCls :hidden");// 选择器中含有空格符
            var $objTmp1 = $(".MyCls:hidden");// 选择器中没有空格符
            var strTmp = "'.MyCls :hidden' 方式获取的元素个数是：";
            strTmp += $objTmp0.length;// 获取含有空格符的总个数
            strTmp += "<br><br>'.MyCls:hidden' 方式获取的元素个数是：";
            strTmp += $objTmp1.length; // 获取不含空格符的总个数
            $("#divTip").append(strTmp);// 显示在页面中
        })
    </script>
</head>
<body>
    <div class="frame">
        <div id="div0" class="MyCls">
```

```
            <div id="div1" class="MyCls"
            style="display:none">
            </div>
        </div>
        <div id="div2" class="MyCls" style="display:none"></div>
        <div id="div3" class="MyCls" style="display:none"></div>
        <div id="divTip"></div>
    </div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-2 所示。

图 10-2　选择器中含有空格符与不含空格符的区别

（4）代码分析

从图 10-2 中可以明显看出，在使用选择器获取元素时，其中是否有空格符虽然并不起眼，但最后获取的结果却截然不同。

在示例 10-2 中，变量 $objTmp0 保存的是类别为 MyCls 中的隐藏元素，因此，它的个数为 1；而变量 $objTmp1 保存的是隐藏元素中类别为 MyCls 的元素，所以，它的个数为 3。由此可见，在通过选择器获取元素时，其中的代码编写应格外注意。

10.3　优化事件中的冒泡现象

从前面的章节中我们知道，页面元素在互相嵌套时，如果都执行同一个事件，可能会触发事件的冒泡现象。在这种现象中，会出现一些意想不到的效果。为了规避这种现象的发生，在 jQuery 中，可以使用 stopPropagation() 方法来停止事件中的冒泡现象。

另外，在元素绑定事件的过程中，还有一个方法 target()，通过该方法可以获取触发事件的元素。如果是多个元素触发同一个事件，可以借助 target() 方法，获取这些元素的父级元素，

并将事件绑定到父级元素，通过冒泡现象，扩展到其子级元素中，这在某种程度上，比将事件绑定到每个子级元素执行效果更加优化。我们将通过示例介绍，通过事件绑定中的 target() 方法，优化多元素绑定同一事件时的冒泡现象。

示例 10-3　事件中的 target 方法优化冒泡现象

（1）功能描述

在一个表单 (form1) 中，包含一个 <fieldset> 标记，在该标记中设置三个文本框 (text)，为了使每个文本框在获取焦点 (focus) 时改变原有的样式，通过事件中的 target 方法，获取文本框的父级元素，并为该元素绑定一个事件，从而实现文本框改变样式的需求。

（2）实现代码

新建一个 HTML 文件 10-3.html, 加入如代码清单 10-3 所示的代码。

代码清单 10-3　事件中的 target 方法优化冒泡现象

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>事件中的 target 方法优化冒泡现象 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
            body{font-size:13px}
            div{line-height:1.8em}
            .txt{border:#666 1px solid;
            padding:2px;width:80px;margin-right:3px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("#frame").bind("click", function(e) {
                $objChild = $(e.target);// 捕捉触发事件的元素
                $objChild.addClass("txt");// 增加子元素的样式
            })
        });
    </script>
</head>
<body>
    <form id="form1">
        <fieldset id="frame" style="width:200px">
          <legend>输入信息 </legend>
          <div>姓名：<input id="Text1" type="text"/></div>
          <div>性别：<input id="Text2" type="text" /></div>
          <div>年龄：<input id="Text3" type="text" /></div>
        </fieldset>
```

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

（3）页面效果

代码执行后的页面效果如图 10-3 所示。

图 10-3　事件中的 target 方法优化冒泡现象

（4）代码分析

在示例 10-3 的 JS 代码中，有如下一段代码：

```
$("#frame").bind("click", function(e) {
    $objChild = $(e.target);// 捕捉触发事件的元素
    $objChild.addClass("txt");// 增加子元素的样式
})
```

其实质是通过 e.target 获取触发事件的元素，并将事件绑定到父级元素中，通过冒泡现象，传递给各个子元素，使其各个子元素也被同样的事件所绑定，因此，上述的 JS 代码与下列代码是等价的，都实现相同的功能。

```
$("#frame input").bind("click", function(e) {
    $(this).addClass("txt");// 增加元素的样式
})
```

但后面的 JS 代码，将遍历全部的 input 元素，逐个进行事件的绑定，如果有大量的 input 元素，执行效率明显低于示例 10-3 的 JS 代码，但考虑到冒泡现象的出现，如果对页面样式没有严格要求的话，采用这种方法是一种不错的优化速度的选择。

10.4　使用 data() 方法缓存数据

我们知道，缓存数据无论是在前端页面开发，还是在后台服务器脚本编写，都有十分重要的作用。同样，在 jQuery 中，也可以通过 data() 方法将数据缓存，虽然使用局部或全局的

变量可以保存数据，但变量并不能进行数据的缓存，而且并不依附于某元素自身；如果使用 data() 方法，可以针对元素定义数据，在元素中存取数据，从而避免数据被循环引用的风险。根据功能的不同，data() 方法有下列几种使用格式：

1）根据元素中的名称定义或返回存储的数据，其调用格式为：

```
data([name])
```

其中，可选项参数 [name] 为字符型，表示存储数据的名称。

2）根据元素中的名称在元素上存储或设置数据，其调用的格式为：

```
data(name, value)
```

其中，参数 name 表示存储数据的名称，value 表示将要被存储的数据。

3）除了定义和存储数据外，还可以移除元素中存放的数据，其调用格式为：

```
removeData(name)
```

其中，参数 name 表示将要被移除的元素上的数据名称。

下面通过一个简单的示例，介绍使用 data() 方法如何在元素上缓存或移除数据。

示例 10-4　使用 data() 方法在元素上存取移除数据

（1）功能描述

在页面中，创建一个 <p> 元素，在该元素上设置或保存数据，并移除数据，同时，将各种状态的缓存数据值显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-4.html, 加入如代码清单 10-4 所示的代码。

代码清单 10-4　使用 data() 方法在元素上存取移除数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 data() 方法在元素上存取移除数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
          body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").data("tmpData"); // 初始时

            // 显示初始化数据
            $("#divTip").append($("p").data("tmpData") == null ?
```

```
            "初始时：null" : $("p").data("tmpData"));

            $("p").data("tmpData", "陶国荣") // 设置

            // 显示设置后的数据
            $("#divTip").append("<br>赋值后：" +
                                $("p").data("tmpData"));

            $("p").removeData("tmpData") // 移除

            // 显示移除后的数据
            $("#divTip").append($("p").data("tmpData") == null ?
            "<br>移除时：null" : $("p").data("tmpData"));
        })
    </script>
</head>
<body>
    <p><b>数据状态</b></p>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-4 所示。

图 10-4　使用 data() 方法在元素上存取移除数据

使用 data() 方法除了可以存储一般数据外，还可以存储以“名称 / 值”格式展示的 JSON 数据。下面通过一个示例，介绍如何通过 data() 方法在元素中保存 JSON 格式的数据。

示例 10-5　使用 data() 方法在元素上存取移除 JSON 格式的数据

（1）功能描述

在示例 10-4 的基础上，将 <p> 元素中使用 data() 方法存储的数据，更改成 JSON 格式，

同样，将各种状态的缓存数据值显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-5.html, 加入如代码清单 10-5 所示的代码。

代码清单 10-5　使用 data() 方法在元素上存取移除 JSON 格式的数据

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 data() 方法在元素上存取移除 JSON 格式的数据 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2-vsdoc.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
    </style>
    <script type="text/javascript">
        $(function() {
            $("p").data("tmpData"); // 初始时

            // 显示初始化数据
            $("#divTip").append($("p").data("tmpData") == null ?
            "初始时：null" : $("p").data("tmpData"));

            $("p").data("tmpData",
           { name: "李建洲", sex: "女", score: "80" }); // 设置

            // 显示设置后的数据
            $("#divTip").append("<br>赋值后：" +
            $("p").data("tmpData").name + "/" +
            $("p").data("tmpData").sex + "/" +
            $("p").data("tmpData").score);

            $("p").removeData("tmpData") // 移除

            // 显示移除后的数据
            $("#divTip").append($("p").data("tmpData") == null ?
            "<br>移除时：null" : $("p").data("tmpData"));
        })
    </script>
</head>
<body>
    <p><b>数据状态 </b></p>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-5 所示。

图 10-5　使用 data() 方法在元素上存取移除 JSON 格式的数据

（4）代码分析

从示例 10-4 与示例 10-5 中，我们不难看出，无论是普通数据还是 JSON 格式的数据，都可以很方便地通过缓存使用 data() 方法进行存储，其数据的读取或移除也十分方便、简洁。尤其需要说明的是，由于缓存存储的数据是全局性的，因此，可以在各个自定义的函数中进行调用，如编写一个自定义的函数 test 调用示例 10-5 中的缓存数据，其代码如下：

```
function test() {
   var strName = $("p").data("tmpData").name;
   alert(strName);
}
```

执行上述代码后，将获取的数据显示在弹出的对话框中，效果如图 10-6 所示。

图 10-6　自定义函数调用存储在缓存中的数据

说明　虽然使用 data() 方法可以很方便地存取全局性的数据，但存储的数据随着操作的变化，会越来越大，如果不及时进行清理，将会影响原有程序的执行，这一点需引起程序开发人员的注意。除此之外，建议在存储针对元素的数据时，使用 data() 方法进行保存，可以优化执行代码。

10.5 解决 jQuery 库与其他库的冲突

在通常情况下，由于 jQuery 库良好的封装性，无论是全局变量（global），还是公用函数，都被无一例外地限定在其固有的默认空间中。基于这样的情况，在一般情况下 jQuery 库可以与其他的 JS 库（如 Prototype 等）并存，不会发生冲突现象。

虽然其他库与 jQuery 库不会发生冲突，但由于“$”是 jQuery 自身的快捷符，而其他 JS 库中也含有“$”符，如果多库共存，那么，就存在是哪个库使用“$”符的问题。为了解决这个问题，在 jQuery 中，可以通过函数 jQuery.noConflict()，将变量“$”的使用权过渡给需要使用的其他 JS 库，其调用的语法格式为：

```
jQuery.noConflict()
```

这个函数的作用是变更“$”变量的使用权，以确定 jQuery 库不与其他库相冲突，使用权变更后，就只能使用 jQuery 变量访问 jQuery 对象。

虽然通过函数 jQuery.noConflict() 可以很好地解决多库共存时变量符“$”的使用权问题，但在实际的应有中，又分为 jQuery 在其他库前导入与在其他库后导入两种情况，下面分别介绍这两种情况下应如何解决。

10.5.1 jQuery 在其他库前导入

如果 jQuery 在其他库前就已经导入了，那么可以直接使用 jQuery 符号处理相应的 jQuery 事务，而不必调用 jQuery.noConflict() 函数。下面通过一个示例来演示其实现的效果。

示例 10-6 解决 jQuery 库先于其他库导入时，变量“$”的使用权问题

（1）功能描述

在页面中，jQuery 库先于 Prototype 导入，然后，在页面中创建两个按钮：单击第一个按钮时，执行 jQuery 库中编写的代码；单击第二个按钮时，执行 Prototype 库中编写的代码。为两个不同库编写的代码功能，都是将获取到的元素内容显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-6.html, 加入如代码清单 10-6 所示的代码。

代码清单 10-6 解决 jQuery 库先于其他库导入时，变量“$”的使用权问题

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>解决 jQuery 库先于其他库导入时，变量 "$" 的使用权 </title>
    <!-- 先导入 jQuery 库 -->
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
```

```
</script>
<!-- 后导入 prototype 库 -->
<script type="text/javascript"
        src="Jscript/prototype-1.6.0.3.js">
</script>
<style type="text/css">
       body{font-size:13px}
       div{margin:5px}
       .btn {border:#666 1px solid;padding:2px;width:80px;
       filter: progid:DXImageTransform.Microsoft
       .Gradient(GradientType=0,StartColorStr=#ffffff,
       EndColorStr=#ECE9D8);}
</style>
<script type="text/javascript">
    // 直接使用 jQuery，无须使用 jQuery.noConflict() 函数
    jQuery(function() {
        jQuery("#Button1").click(function() {
            // 获取对象
            var $objTmp = jQuery("#txtName");
            // 显示内容
            jQuery("#divTmp").html("J_" + $objTmp.val());
        })
    })
    // 自定义一个测试函数
    function prototype_test() {
        // 使用 prototype 语法获取元素内容
        $("divTmp").innerHTML = "P_" + $F("txtName");
        }
    </script>
    </head>
    <body>
       <input type="text" id="txtName" value=" 测试内容 "
              style="display:none">
       <div>
           <input id="Button1" type="button" value="jQuery"
                  class="btn" />  
           <input id="Button2" type="button" value="prototype"
                  onclick="prototype_test();" class="btn" />
       </div>
       <div id="divTmp"></div>
    </body>
    </html>
```

（3）页面效果

代码执行后的页面效果如图 10-7 所示。

（4）代码分析

从图 10-7 中我们可以看出，两个按钮的单击事件都执行成功。这说明，在 jQurey 库先于其他库导入时，无须使用 jQuery.noConflict() 函数，就可以将变量“$”的使用权转让给其他

库。jQuery 库处理事务时，不再使用“$”符，全部修改成“jQuery”符号即可以执行。

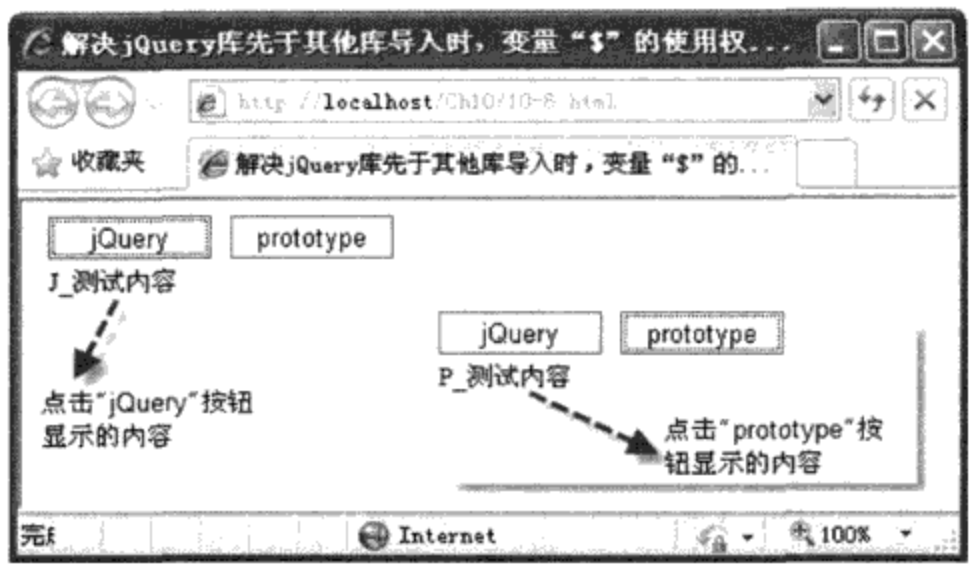


图 10-7　解决 jQuery 库先于其他库导入时，变量“$”的使用权问题

10.5.2　jQuery 在其他库后导入

如果 jQuery 库在其他库后面导入，若仍然采用示例 10-6 中 JS 代码的写法将不能正常执行。这时需要引入 jQuery.noConflict() 函数，声明转让变量“$”的使用权。这时，在 jQuery 代码中，如果还想使用变量“$”符号，那么，有两种方法可以实现：第一个方法是自定义一个含其他符号的快捷方式；第二个方法是在 jQuery 函数的内部继续使用变量“$”。下面通过一个示例来进行演示。

示例 10-7　解决 jQuery 库后于其他库导入时，变量“$”的使用权问题

（1）功能描述

基本功能与示例 10-6 相同，不同之处在于，jQuery 库在其他库后导入。

（2）实现代码

新建一个 HTML 文件 10-7.html, 加入如代码清单 10-7 所示的代码。

代码清单 10-7　解决 jQuery 库后于其他库导入时，变量“$”的使用权问题

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>解决 jQuery 库后于其他库导入时，变量 "$" 的使用权 </title>
    <!-- 先导入 prototype 库 -->
    <script type="text/javascript"
            src="Jscript/prototype-1.6.0.3.js">
    </script>
    <!-- 后导入 jQuery 库 -->
    <script type="text/javascript"
```

```
        src="Jscript/jquery-1.4.2.js">
</script>
<style type="text/css">
       body{font-size:13px}
       div{margin:5px}
       .btn {border:#666 1px solid;padding:2px;width:80px;
       filter: progid:DXImageTransform.Microsoft
       .Gradient(GradientType=0,StartColorStr=#ffffff,
       EndColorStr=#ECE9D8);}
</style>
<script type="text/javascript">
    // 使用 jQuery.noConflict() 函数
    //jQuery.noConflict();
    //jQuery(function() {
        //jQuery("#Button1").click(function() {
            // 获取对象
            //var $objTmp = jQuery("#txtName");
            // 显示内容
            //jQuery("#divTmp").html("J_" + $objTmp.val());
        //})
    //})

    //**** 等价于方法一自定义快捷方式 ***//
    // 使用 jQuery.noConflict() 函数
    //var j=jQuery.noConflict();
    //j(function() {
        //j("#Button1").click(function() {
            // 获取对象
            //var $objTmp = j("#txtName");
            // 显示内容
            //j("#divTmp").html("J_" + $objTmp.val());
    //})
//})

//**** 等价于方法二在 jQuery 函数内容使用 $ 符 ***//
// 使用 jQuery.noConflict() 函数
jQuery.noConflict();
jQuery(function($) {
    $("#Button1").click(function() {
        // 获取对象
        var $objTmp = $("#txtName");
        // 显示内容
        $("#divTmp").html("J_" + $objTmp.val());
    })
})

// 自定义一个测试函数
function prototype_test() {
    // 使用 prototype 语法获取元素内容
    $("divTmp").innerHTML = "P_" + $F("txtName");
```

```
        }
</script>
</head>
<body>
    <input type="text" id="txtName" value=" 测试内容 "
                        style="display:none">
    <div>
        <input id="Button1" type="button" value="jQuery"
                            class="btn" />  
        <input id="Button2" type="button" value="prototype"
               onclick="prototype_test();" class="btn" />
    </div>
    <div id="divTmp"></div>
</body>
</html>
```

示例 10-7 最终实现的页面效果与示例 10-6 的效果一模一样。通过上述两种方法的介绍，相信在页面中与 jQuery 库同时并存的问题，都可以迎刃而解。

10.6 使用子查询优化选择器性能

在 jQuery 中，如果要查找一个元素，而这个元素被众多别的元素所包含或嵌套在其中。这时，如果直接使用 find() 方法进行查找，其操作性能比较低。由于 jQuery 允许在一个集合中附加其他的选择操作，这样的话，我们可以先查找最外层的元素，保存起来，再查找更近一层的元素，进行保存，最后，在最接近的外层中，使用 find() 方法，查找需要的元素。通过这种方式，可以在本地变量中保存上一级对象，减少选择器的性能开销。

下面通过一个简单示例来说明该方法实现的过程。

示例 10-8 使用子查询优化选择器性能

（1）功能描述

在页面中，创建一个 标记，并在其中创建一个列表 ，列表中包含两行表项 ，在其中的一个表项中设置 标记，通过子查询，获取 元素与另外一个表项中的内容，并显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-8.html, 加入如代码清单 10-8 所示的代码。

代码清单 10-8 使用子查询优化选择器性能

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
    <title>使用子查询优化选择器性能</title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{margin:5px}
           .ulFrame{padding:0px;margin:0px;list-style-type:none}
           .li0{font-size:11px}
           .li1{font-size:12px}
    </style>
    <script type="text/javascript">
        $(function() {
            var $divF = $("#divFrame");// 保存最外层对象
            var $ulF = $divF.find(".ulFrame");// 在外层对象中查找
            var $li0 = $ulF.find(".li0");
            var $spn = $li0.find("span");// 在最近一层中查找
            var $li1 = $ulF.find(".li1");
            var strTmp = "最终数据：" // 初始化显示内容
            strTmp += "<br>" + $spn.html(); // 获取内容
            strTmp += "<br>" + $li1.html(); // 获取内容
            $("#divTip").append(strTmp); // 显示在页面中
        })
    </script>
</head>
<body>
    <div id="divFrame">
        <ul class="ulFrame">
            <li class="li0"><span>测试元素一</span></li>
            <li class="li1">测试元素二</li>
        </ul>
    </div>
    <div id="divTip"></div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-8 所示。

（4）代码分析

如果仅仅是为了获取列表的第一个表项中的 标记内容和第二个表项中的内容，那么，初学者通常会编写下列代码：

```
$spn = $("div ul li span");
$li1 = $("div ul .li1");
```

上述代码同样也可以获得预期的结果，但代码没有缓存，性能较低，对选择器的开销很大，并且不利于查询同级数据，每次都是一个新开销。因此，在嵌套元素中，应尽量使用子查询访问元素。

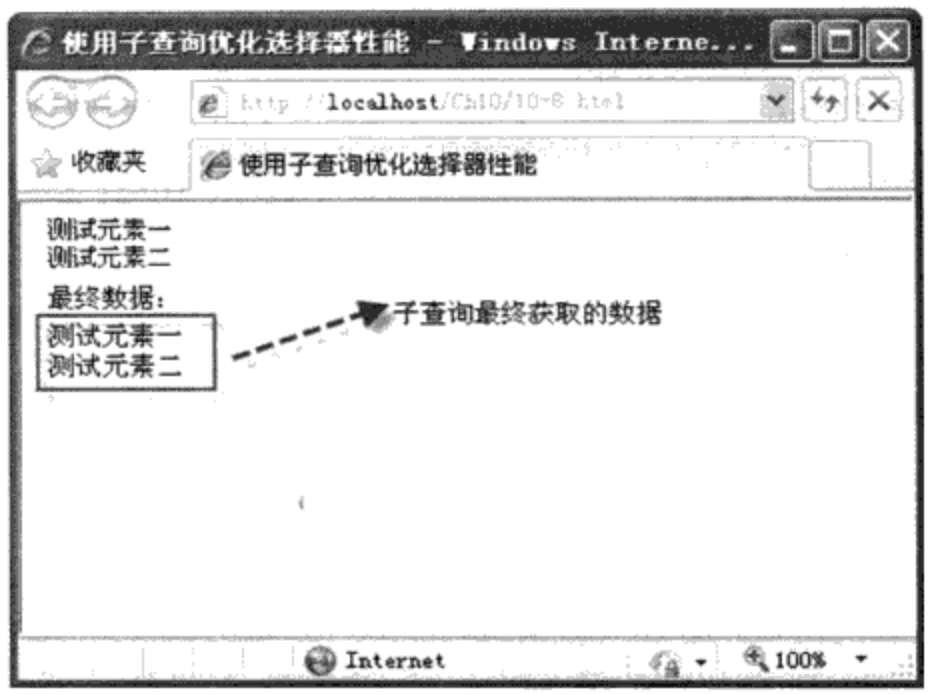


图 10-8　使用子查询优化选择器性能

10.7　减少对 DOM 元素直接操作

我们知道，DOM 元素操作的原理是：先在内存中创建 DOM 结构，然后，更新现有的 DOM 结构。如果直接对 DOM 进行操作，那么，其性能也是很低的，因此，为了减少这种对 DOM 元素直接操作的次数，有必要在操作前完善大部分的 DOM 操作，最后通过一次直接操作，更新其 DOM 结构。

下面通过一个简单的示例介绍该操作实现的过程。

示例 10-9　减少对 DOM 元素直接操作

（1）功能描述

在页面中创建一个列表 <ul> 标记，然后通过代码，在列表中动态添加 6 个含有内容的表项，并将最终结果显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-9.html, 加入如代码清单 10-9 所示的代码。

代码清单 10-9　减少对 DOM 元素直接操作

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>减少对 DOM 元素直接操作 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
```

```
    </script>
    <style type="text/css">
          body{font-size:13px}
          #ulFrame{padding:0px;margin:0px;
          list-style-type:none;width:200px}
          ul li{border-bottom:dashed 1px #ccc;padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            //定义数组
            var arrList = ["list0", "list1", "list2",
                           "list3", "list4", "list5"];
            var strList = "";//初始化字符
            $.each(arrList, function(index) {
                //遍历后累加数组元素
                strList += "<li>" + arrList[index] + "</li>";
            })
            //一次性完成 DOM 元素的增加
            $("#ulFrame").append(strList);
        })
    </script>
</head>
<body>
    <ul id="ulFrame"></ul>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-9 所示。

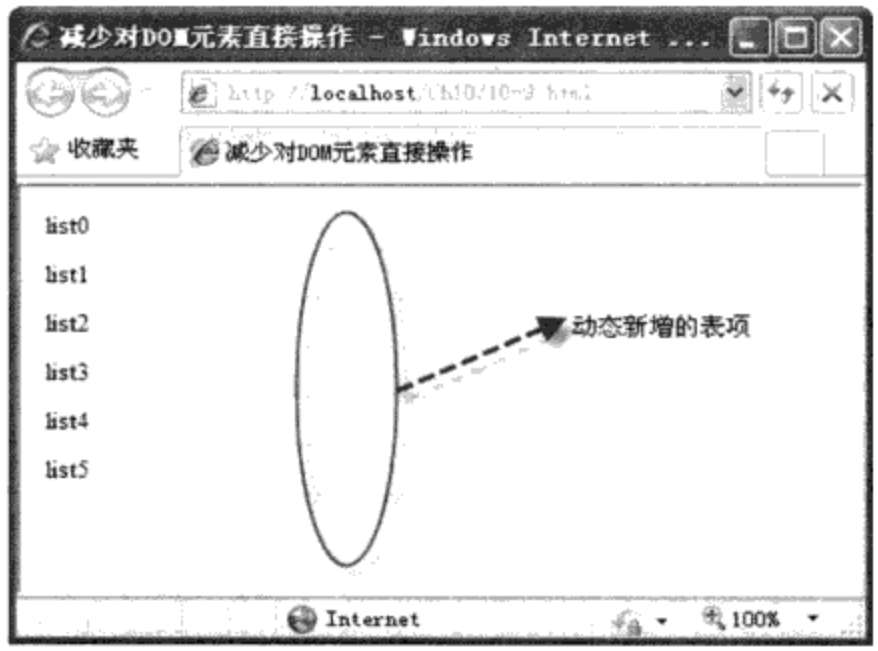


图 10-9　减少对 DOM 元素直接操作

（4）代码分析

在通过遍历的方式动态增加多个 DOM 元素时，许多初学者往往采用下列的代码：

```
...省略部分代码
$.each(arrList, function(index) {
    //遍历后累加数组元素
    $("#ulFrame").append("<li>" + arrList[index] + "</li>");
})
...省略部分代码
```

这样同样也可以实现动态增加 DOM 元素的效果，但每遍历一次，都直接对 DOM 元素进行操作，效率很低。因此，建议尽量减少直接对 DOM 元素进行操作，以优化操作性能。

10.8 正确区分 DOM 对象与 jQuery 对象

由于在 jQuery 中可以使用 JavaScript 语法，因此 jQuery 初学者容易混淆 DOM 与 jQuery 对象。一旦对象理解出错，那么，与对象相关的操作也就不会正确。下面将介绍如何区分 DOM 对象与 jQuery 对象，同时阐述二者将如何转换。

10.8.1 DOM 对象与 jQuery 对象的定义

所谓 DOM 对象，指的是通过传统的 JavaScript 方法获取的 DOM 元素对象，如下列代码：

```
var objDom = document.getElementById("txtTmp");
var DomValue = objDom.value;
```

其中，变量 objDom 保存的就是表单中的某个指定文本框对象，即 DOM 对象；变量 DomValue 获取了 Id 号为“txtTmp”的 DOM 对象的值。

所谓 jQuery 对象，指的是通过 jQuery 语法包装原始的 DOM 对象后生成的新对象。jQuery 对象在 jQuery 库中是特有的，只要是 jQuery 对象，就可以使用 jQuery 库中的所有方法与事件。如下列代码：

```
var $obj = $("#txtTmp");
var DomValue = $obj.val();
```

其中，变量 $obj 保存的就是表单中指定的文本框对象，由于被 $() 方法所包装，所以，该对象变成了 jQuery 对象；DomValue 变量保存通过 jQuery 中的 val() 方法获取的 jQuery 对象的值。

需要注意的是，不能使用 DOM 对象调用 jQuery 对象的方法，下列代码是错误的：

```
var objDom = document.getElementById("txtTmp");
var DomValue = objDom.val();
```

同理，也不能使用 jQuery 对象调用 DOM 对象的方法，下列代码也是错误的：

```
var $obj = $("#txtTmp");
var DomValue = $obj.value;
```

如果需要使 DOM 对象与 jQuery 对象之间的方法可以互相调用，必须先实现 DOM 对象与 jQuery 对象之间的类型转换。

10.8.2　DOM 对象与 jQuery 对象的类型转换

在 jQuery 中，可以很方便地完成 DOM 对象与 jQuery 对象之间的转换。只需调用 jQuery 中提供的 [index] 与 get(index) 方法即可将 jQuery 对象转换成 DOM 对象。DOM 对象只要通过 jQuery 方法 $() 进行包装，就可以转换成 jQuery 对象。下面通过一个简单的示例演示 DOM 对象与 jQuery 对象相互转换的过程。

示例 10-10　DOM 对象与 jQuery 对象的类型转换

（1）功能描述

在页面中，创建两个 标记，分别用于保存对象转换后设置的内容：Id 号为 div0 的 标记保存 DOM 对象转成 jQuery 对象后设置的内容；Id 号为 div1 的 标记保存 jQuery 对象转成 DOM 对象后设置的内容，同时，将设置的内容均显示在页面中。

（2）实现代码

新建一个 HTML 文件 10-10.html, 加入如代码清单 10-10 所示的代码。

代码清单 10-10　DOM 对象与 jQuery 对象的类型转换

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title> DOM对象与jQuery对象的类型转换 </title>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <style type="text/css">
           body{font-size:13px}
           div{padding:5px}
    </style>
    <script type="text/javascript">
        $(function() {
            //***** DOM对象转成jQuery对象 *****//
            //DOM对象
            var objDom0 = document.getElementById("div0");
            // 转成jQuery对象
            var $obj0 = $(objDom0);
            // 调用jQuery中的方法设置其中的内容
            $obj0.html("DOM对象转成jQuery对象后设置的内容 ");

            //***** jQuery对象转成DOM对象 *****//
            //jQuery对象
```

```
            var $obj1 = $("#div1");
            // 转成 DOM 对象
            var objDom1 = $obj1[0];
            // 调用 JavaScript 中的对象方法设置内容
            objDom1.innerHTML = "jQuery 对象转成 DOM 对象后设置的内容 ";
        })
    </script>
</head>
<body>
    <div id="div0"></div>
    <div id="div1"></div>
</body>
</html>
```

（3）页面效果

代码执行后的页面效果如图 10-10 所示。

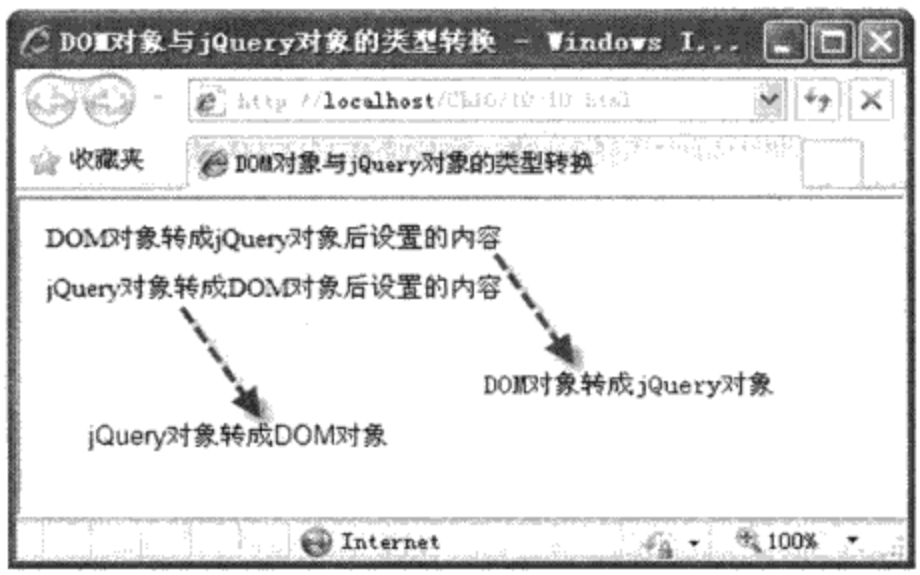


图 10-10　DOM 对象与 jQuery 对象的类型转换

（4）代码分析

从示例 10-10 中可以看出，DOM 对象与 jQuery 对象之间的类型转换十分方便，代码也很简单。在该示例的 JS 代码中，除使用 [index] 方法将 jQuery 对象转换成 DOM 对象外，还可以使用 get(index) 方法，因此，下面几行代码：

```
...省略部分代码
//jQuery 对象
var $obj1 = $("#div1");
// 转成 DOM 对象
var objDom1 = $obj1[0];
...省略部分代码
```

等价于

```
...省略部分代码
//jQuery 对象
```

```
var $obj1 = $("#div1");
// 转成 DOM 对象
var objDom1 = $obj1.get(0);
... 省略部分代码
```

通过上述示例的演示，让我们分清了 DOM 对象与 jQuery 对象之间的区别，为进一步学习和掌握对象间的方法打下了坚实的基础。

10.9　本章小结

作为一种新生的语言框架——jQuery，对于程序开发者，尤其是初学者来说，在使用过程中不可避免会出现语法或操作上的不当之处。本章先从优化选择器查询元素的功能讲起，列出在编写 jQuery 代码中各种欠妥或可优化的现象，通过理论与示例结合的方式，进行纠正与优化，同时，对一些技术技巧也进行了总结。读者可以结合自己开发过程的实际情况，针对本章节中提到的问题合理进行规避，以实现代码质量的最优化。

第 11 章

综合案例开发

本章内容

- 案例 1：切割图片
- 案例 2：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，但是如果使用 jQuery 插件，编写几行代码就可以轻松地实现该功能。另外，编写一个简单、高效的聊天室，用于日常的交流与商谈，将极大地方便日常的工作。本章将以案例的方式讲解如何使用 jQuery 插件实现图片的快速切割，以及如何基于 jQuery 库开发一个简洁、易操作的聊天室。

案例 1：切割图片

在电脑中的图片，我们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，就可以让用户在页面中将上传后的图片进行任意大小的切割。

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不限大小的切割。

2）对图片切割时，需要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，可以将所选择的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，进行切割时将提示出错信息。

效果界面

在本案例中，为实现上述用户需求，在 HTML 页面 ImgCropper.html 中导入了一款基本 jQuery 库的图片切割插件 jquery.imagecropper.js，并加载一张被切割的图片，该页面的初始状态如图 11-1 所示。

当用户在被切割图片中移动鼠标时，将出现一个边框与四个角都有小正方框的可拖动的块区。用户可以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大小预览所切割区的图片效果，其实现的页面如图 11-2 所示。

当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 11-3 所示。

图 11-1　图片切割页的初始状态

图 11-2　选择被切割区后的页面效果

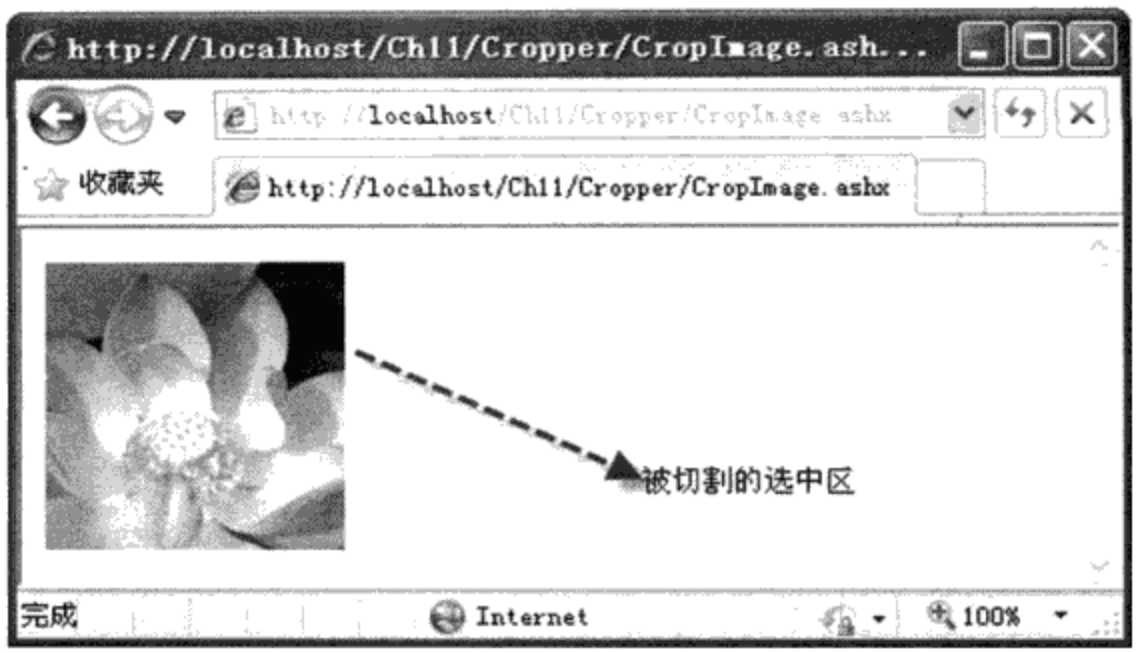


图 11-3　最终切割的图片

功能实现

为了实现对图片切割的功能，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，当单击“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。实现完整代码的过程如下。

新建一个 HTML 页面 ImgCropper.html，加入如代码清单 11-1 所示的代码。

代码清单 11-1　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-1.4.2-vsdoc.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery-1.4.2.js">
    </script>
    <script type="text/javascript"
            src="Jscript/jquery.imagecropper.js">
    </script>
    <script type="text/javascript">
        $(function() {
        $('#cropbox').Cropper({ // 调用切割方法
                aspectRatio: 1,// 设置所选区域的长宽比例
                onChange: showPreview,// 设置长宽变化后触发的事件
                onSelect: showPreview // 设置选择区域后触发的事件
            })
        });
```

```
        // 自定义提交切割数据时的检测函数
        function checkCoords() {
            if (parseInt($('#w').val())) return true;
            alert('请选择一个区域后，点击确定按钮！');
            return false;
        }
        // 根据预览图片的大小调用图片预览函数
        function showPreview(coords) {
            if (parseInt(coords.w) > 0) {
                imgPrev(coords, "#img100", 100);
                imgPrev(coords, "#img80", 80);
                imgPrev(coords, "#img60", 60);
            }
            // 获取所选择区域的坐标与宽高
            $('#x').val(coords.x);
            $('#y').val(coords.y);
            $('#w').val(coords.w);
            $('#h').val(coords.h);
        }
        // 自定义所选择区域的图片预览函数
        function imgPrev(coords,obj, w) {
            var rx = w / coords.w;
            var ry = w / coords.h;
            $(obj).css({ // 预览图片
                width: Math.round(rx * 504) + 'px',
                height: Math.round(ry * 378) + 'px',
                marginLeft: '-' + Math.round(rx * coords.x) + 'px',
                marginTop: '-' + Math.round(ry * coords.y) + 'px'
            });
        }
    </script>
    <link rel="stylesheet" type="text/css"
          href="Css/ImgCropper.css"/>
</head>
<body>
    <div id="outer">
     <div class="jcExample">
       <div class="article">
            <h2>图片切割示例</h2>
               <img src="Images/imgDemo.jpg" id="cropbox" />
               <form action="CropImage.ashx" method="post"
                     onsubmit="return checkCoords();">
                      <input type="hidden" id="x" name="x" />
                      <input type="hidden" id="y" name="y" />
                      <input type="hidden" id="w" name="w" />
                      <input type="hidden" id="h" name="h" />
                      <input type="hidden" id="p" name="p"
                             value="Images/imgDemo.jpg" />
                      <input type="hidden" id="l" name="l"
                                           value="0.9"/>
```

```
                    <input type="submit" value="确定"
                                        class="btn" />
                </form>
            </div>
            <div class="clsPre">
                <div class="img100"><img src="Images/imgDemo.jpg"
                                        class="imgPre" id="img100"/>
                    </div><div class="clsAlt">100*100 像素 </div>
                <div class="img80"><img src="Images/imgDemo.jpg"
                                        class="imgPre" id="img80"/>
                    </div><div class="clsAlt">80*80 像素 </div>
                <div class="img60"><img src="Images/imgDemo.jpg"
                                        class="imgPre" id="img60"/>
                    </div><div class="clsAlt">60*60 像素 </div>
            </div>
        </div>
    </div>
</body>
</html>
```

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，其完整源码如代码清单 11-2 所示。

代码清单 11-2　HTML 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 11-3 所示。

代码清单 11-3　处理 HTML 提交的图片切割文件 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;
            }
        }
    }
```

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，将图片与插件通过 Cropper 方法进行绑定，该图片就可以实现被任意切割的功能，其实现的代码如下所示：

```
$('#cropbox').Cropper({ // 调用切割方法
    aspectRatio: 1,// 设置所选区域的长宽比例
    onChange: showPreview,// 设置长宽变化后触发的事件
    onSelect: showPreview // 设置选择区域后触发的事件
    })
 });
```

在这里，需要对 imagecropper 插件进行简单的介绍：该插件是一款基于 jQuery 的图片切割插件，可以实现图片在线切割，达到和图像软件处理同样的效果，操作界面也十分人性化，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 11-1 所示。

表 11-1　选项 options 中的常用参数

参数名称	说　明
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、y 轴的宽与高
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。为了实现这项功能，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 与 y 轴的值，然后，根据这两个值，设置各类预览图片的 width 和 height 值、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'
   });
}
```

其中函数 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。由于在本案例中，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，则分别对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，将函数 showPreview() 与插件的 onChange 和 onSelect 事件绑定，从而实现当改变所选区域时，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、高与 x、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，为了防止在切割图片时提交无效的选中区域，在页面提交时，先通过自定义函数 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;
  }
```

案例 2：在线聊天室

虽然当前各类交流工具十分方便快捷，但不少企业或公司还是希望开发一个简单、高效便于公司内部交流的聊天室。下面将详细介绍一个基于 jQuery 库的小型聊天室开发的全过程。

需求分析

本案例的聊天室需要满足下列需求：

1）用户需要登录后，才能进入聊天室交流。

2）以无刷新的方式，动态显示交流后的内容。

3）以无刷新的方式，动态显示在线人员的基本信息。

4）登录后的用户可以提交文字与表情图标，并即时显示在内容区。

效果界面

为了显示当前的在线人员信息，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 11-4 所示。

图 11-4　用户登录聊天室

用户在登录聊天室时，调用 jQuery 中 Ajax 的全局函数 $.ajax()，将获取的用户名与密码数据向服务器发送请求，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。

当用户在登录时，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 11-5 所示。

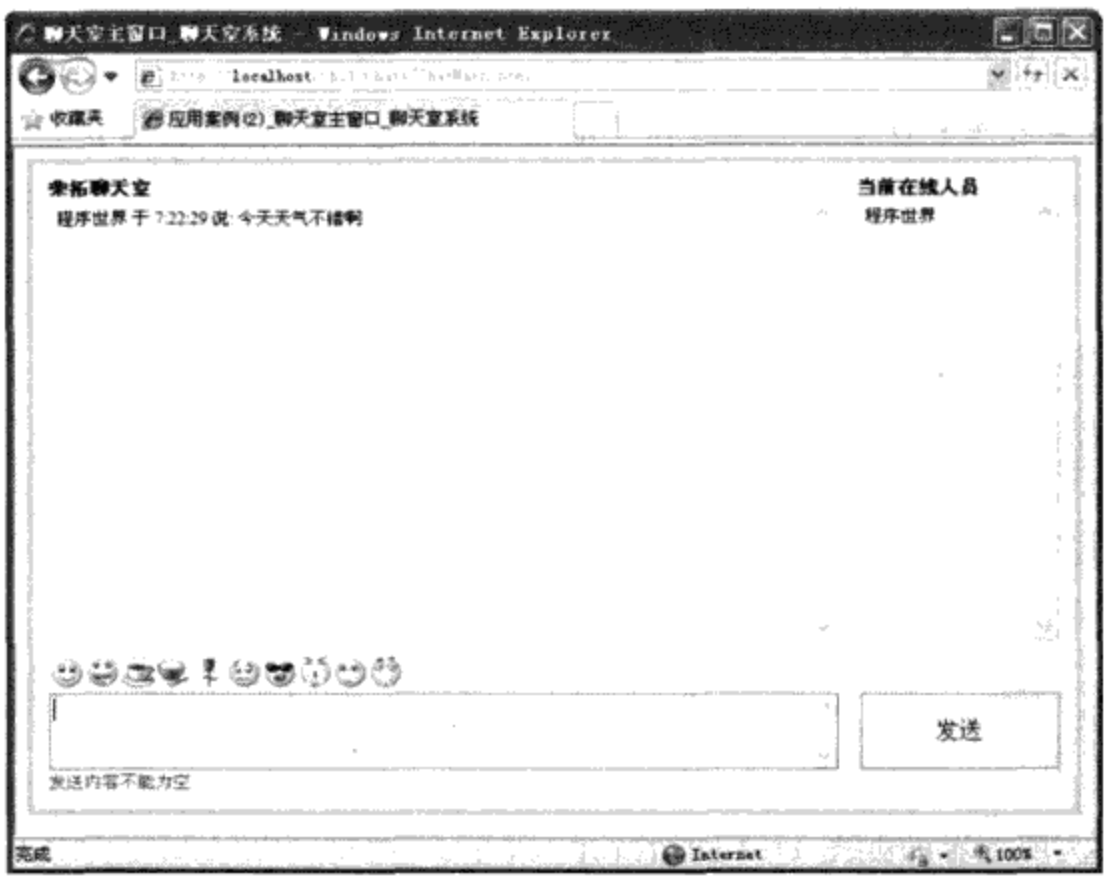


图 11-5　首次登录聊天主窗口的页面

首次登录聊天主页面时，左边的聊天内容区数据为空，右边显示登录时输入的“用户名”，在底部，可以在文本框中输入聊天内容，单击“发送”按钮后，将通过 jQuery 中 Ajax 的全局方法 $.ajax()，获取聊天内容，并向服务器提交请求，同时，服务器响应数据请求，写入完成后，将数据返回至主窗口页面，显示在内容区中。

聊天室主要的功能是实现多人在线聊天，另外，在发送内容时，还可以发送表情图标，以丰富发送内容。当多人在线并且发送表情图标显示在内容区中时，实现的界面如图 11-6 所示。

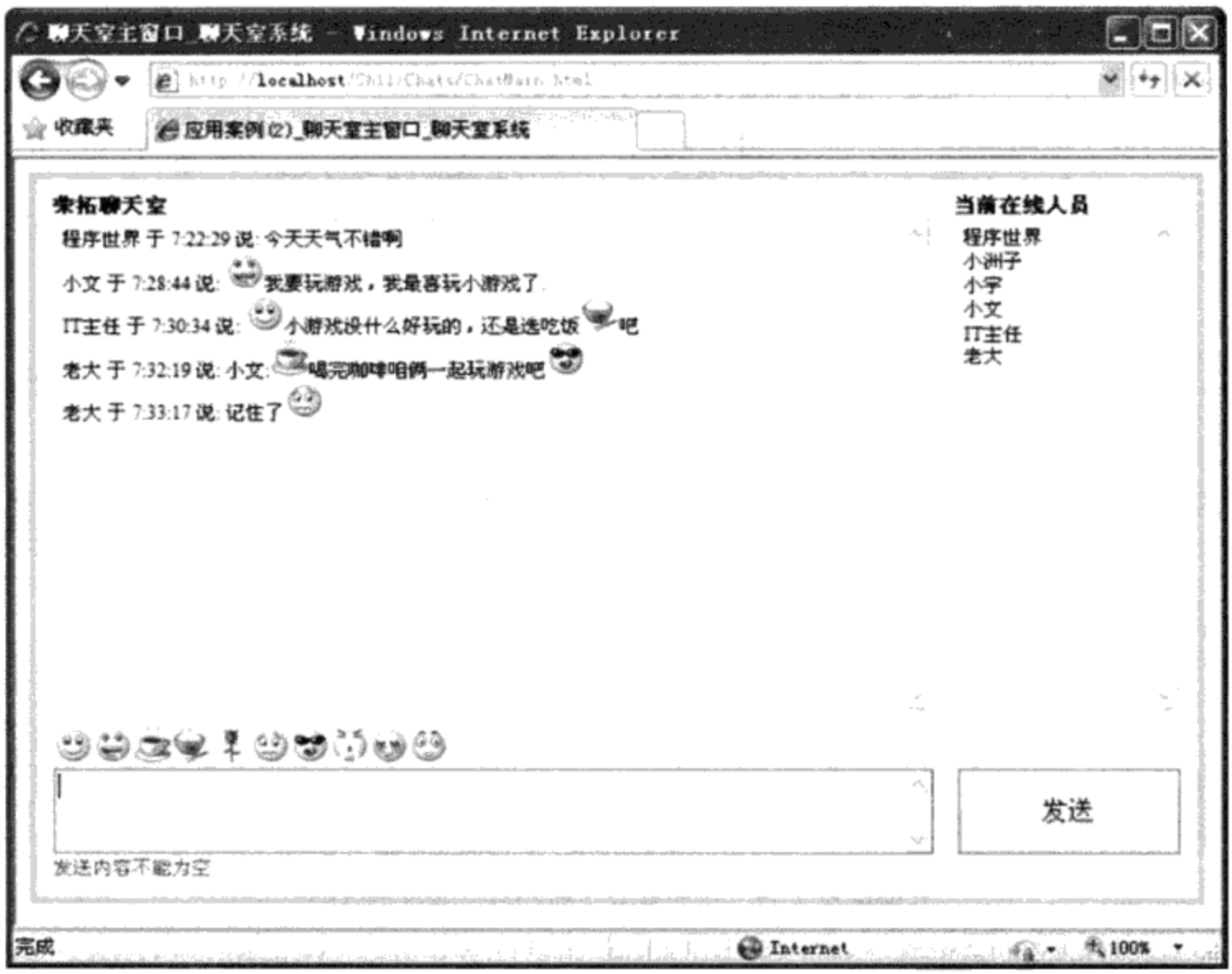


图 11-6　显示多人在线并发送表情图标

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 的方法 $.ajax() 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，则返回 true，否则，返回 false；客户端接收到 true 后，则进入聊天室页面，否则弹出“用户名或密码错误！”的窗口，提示用户登录失败。

当用户登录成功时，则进入聊天室页面 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()，分别用于时时获取最新的聊天内容与在线人员数据信息。

当用户在聊天室页面 ChatMain.html 底部的文本框中，输入发送内容并单击“发送”按钮后，将调用另外一个自定义的 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。本案例的操作流程图如图 11-7 所示。

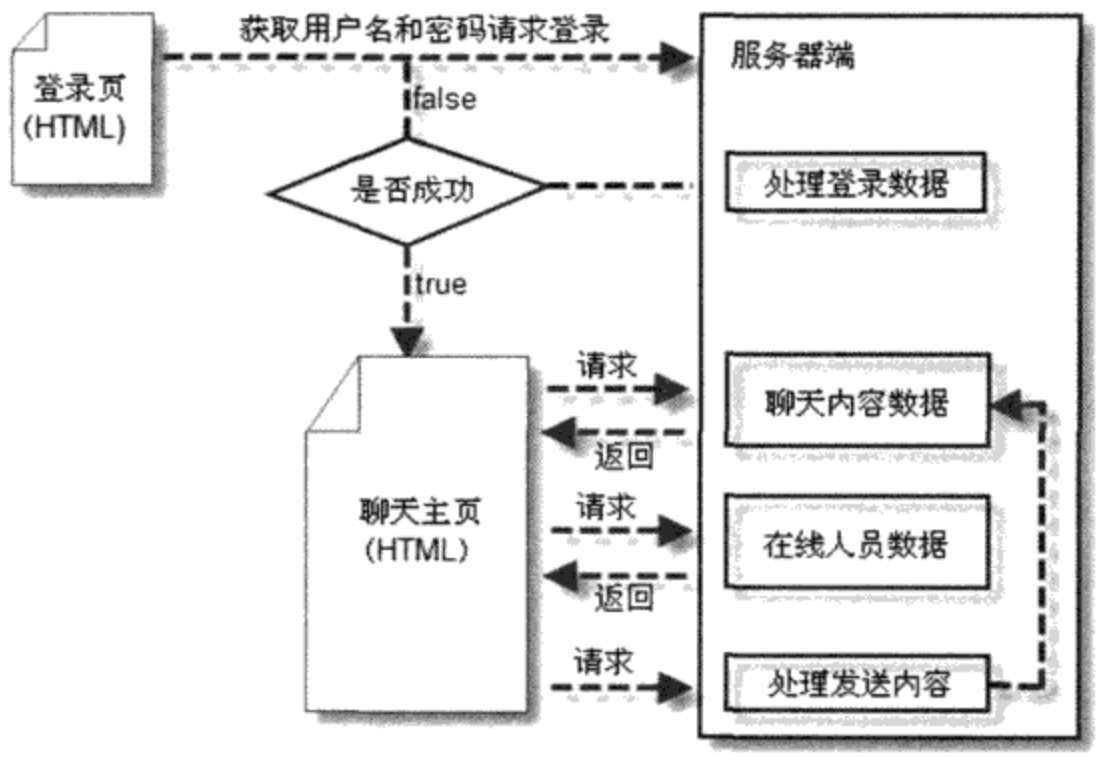


图 11-7　在线聊天室操作流程图

在案例中，用户要进入聊天室，首先进行登录验证。为实现该功能，新创建一个 HTML 页面 Login.html，加入如代码清单 11-4 所示的代码。

代码清单 11-4　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-1.4.2.js">
    </script>
    <link rel="stylesheet" type="text/css"
            href="Css/cssLogin.css" />
    <script type="text/javascript"
            src="Js/jsLogin.js">
```

```
    </script>
</head>
<body>
<div id="divLogin">
  <fieldset>
     <h3>用户登录</h3>
     <div class="content">
          <div>用户：<input id="txtName" type="text"
                                        class="txt" /></div>
          <div>密码：<input id="txtPass" type="password"
                                        class="txt"/></div>
          <div class="btnCenter">
              <input id="Button1" type="button" value="登录"
                                        class="btn" />  
              <input id="Reset1" type="reset" value="取消"
                                        class="btn" />
          </div>
          <span id="divMsg" class="clsTip">正在发送登录请求...</span>
     </div>
  </fieldset>
</div>
</body>
</html>
```

在登录页面 Login.html 中，为了便于以后代码的维护与重复使用，新建一个独立的 JS 文件 jsLogin.js。在该文件中，编写处理用户登录时的 JS 代码，该文件的完整代码如代码清单 11-5 所示。

代码清单 11-5　JS 文件 jsLogin.js 的完整代码

```
/// <reference path="../Jscript/jquery-1.4.2-vsdoc.js"/>
/// <reference path="../Jscript/jquery-1.4.2.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("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;
            }
        }
    });
}
```

同时，在登录页面 Login.html 中，还包含一个 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 11-6 所示。

代码清单 11-6　CSS 文件 cssLogin.css 的完整代码

```
body
{
   font-size:11px
}
#divLogin
{
   margin:15% 0 0 15%
}
#divLogin fieldset
{
   width:300px;
   padding:0px;
   margin:0px
}
```

```
#divLogin fieldset h3
{
   margin:0px;
   padding:5px;
   background-color:#eee
}
 #divLogin fieldset .content
 {
   padding:20px;
   line-height:2.6em
 }
#divLogin fieldset .content .btnCenter
{
   text-align:center
}
/*侦察请求状态样式*/
.clsTip
{
   position:absolute;
    width:160px;
    text-align:center;
    font-size:13px;
    border:solid 1px #cc3300;
    margin-top:5px;
    padding:2px;
    margin-bottom:5px;
    background-color:#ffe0a3;
    display:none;
 }
.txt
{
    border:#666 1px solid;
    padding:2px;width:180px;
    margin-right:3px
 }
.btn
{
   border:#666 1px solid;
   padding:2px;
   width:60px;
   filter: progid:DXImageTransform.Microsoft
   .Gradient(GradientType=0,StartColorStr=#ffffff,
   EndColorStr=#ECE9D8)
}
```

当用户登录成功后，则进入聊天室主页面，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 11-7 所示的代码。

代码清单 11-7　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-1.4.2.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，该文件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代码清单 11-8 所示。

代码清单 11-8　JS 文件 jsMain. js 的完整代码

```
/// <reference path="../Jscript/jquery-1.4.2-vsdoc.js"/>
/// <reference path="../Jscript/jquery-1.4.2.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);
    }
```

另外，在聊天室主页面 ChatMain.html 中，包含一个用于控制页面布局的 CSS 文件 cssMain.css，该文件用于设置页面的框架结构与元素样式，其完整的代码如代码清单 11-9 所示。

代码清单 11-9　聊天室主页面的 CSS 文件 cssMain.css 的完整代码

```
body
{
    font-size:11px
}
h3
{
    margin:0px
}
.divShow
{
    border:solid 1px #ccc;
    height:300px;
    padding:5px;
    font-size:12px;
    overflow-y:scroll
}
#divMain
{
    border:solid.5px #ccc
}
#divMain .divtop
{
    padding:10px
}
#divMain .divtop .divL
{
    float:left;
    width:78%
}
#divMain .divtop .divR
{
    float:right;
    width:20%
}
#divMain .divBot
{
    clear:both;
    padding:10px
}
#divMain .divBot .pb
```

```
{
   padding-bottom:3px
}
#divMain .divBot .pl
{
   padding-left:12px
}
#divMain .divBot .pt
{
   padding-top:3px;
   color:#555
}
/* 侦察请求状态样式 */
.clsTip
{
   position:absolute;
    width:160px;
    text-align:center;
    font-size:13px;
    border:solid 1px #cc3300;
    margin-top:5px;
    padding:2px;
    margin-bottom:5px;
    background-color:#ffe0a3;
    display:none;
 }
.txt
{
   border:#666 1px solid;
   padding:2px;
   margin-right:3px
 }
.btn
{
   border:#666 1px solid;
   padding:2px;
   width:135px;
   height:54px;
   font-size:16px;
   filter: progid:DXImageTransform.Microsoft
   .Gradient(GradientType=0,StartColorStr=#ffffff,
   EndColorStr=#ECE9D8)
}
```

同时，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 11-10 所示。

代码清单 11-10　处理客户端数据请求的 Web 页面文件 DealData.aspx 的完整代码

```
<%@ Import Namespace="System.Collections.Generic" %>
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        string strAction = System.Web.HttpUtility.UrlDecode(
                           Request["action"]);
        string strName = System.Web.HttpUtility.UrlDecode(
                           Request["name"]);
        string strPass = System.Web.HttpUtility.UrlDecode(
                           Request["pass"]);
        string strContent = System.Web.HttpUtility.UrlDecode(
                           Request["content"]);
        switch (strAction)
        {
            case "Login":
                Response.Clear();
                Response.Write(UserLogin(strName,strPass));
                Response.End();
                break;
            case "ChatList":
                Response.Clear();
                Response.Write(AllChatList());
                Response.End();
                break;
            case "OnLineList":
                Response.Clear();
                Response.Write(GetOnlineUserList());
                Response.End();
                break;
            case "SendContent":
                Response.Clear();
                Response.Write(AddSendContent(strContent));
                Response.End();
                break;
    }
}
/// <summary>
/// 用户登录
/// </summary>
/// <param name="strName"></param>
/// <returns></returns>
private bool UserLogin(string strName, string strPass)
{
    bool blnR = false;
    if (strPass == "123456")// 初始化密码
    {
        List<string> OnLineUserList =
```

```
                (List<string>)Application["OBJUSERLIST"];
            if (OnLineUserList == null)// 对象为 NULL
            {
                OnLineUserList = new List<string>();// 实例化对象
                OnLineUserList.Add(strName);// 新增对象元素
            }
            else
            {
                OnLineUserList.Add(strName);// 新增对象元素
            }
            Application["OBJUSERLIST"] = OnLineUserList;
            Session["LOGINUSER"] = strName;
            blnR = true;
        }
        return blnR;
    }
    /// <summary>
    /// 新增发送内容
    /// </summary>
    /// <param name="strContent"></param>
    /// <returns></returns>
    private bool AddSendContent(string strContent)
    {
        string name = Session["LOGINUSER"].ToString();
        string strSendConent = name + " 于 "
        + DateTime.Now.ToLongTimeString().ToString()
        + " 说 : " + strContent;
        List<string> strSendConentList =
                    (List<string>)Application["OBJMESSAGELIST"];
        if (strSendConentList == null)
        {
            strSendConentList = new List<string>();
        }
        strSendConentList.Add(strSendConent);
        Application["OBJMESSAGELIST"] = strSendConentList;
        return true;
    }
    /// <summary>
    /// 获取全部聊天记录
    /// </summary>
    /// <returns></returns>
    private string AllChatList()
    {
        string strSendConent = string.Empty;
        List<string> strSendConentList =
                    (List<string>)Application["OBJMESSAGELIST"];
        if (strSendConentList == null)
        {
            strSendConent = " 日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>
```

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 的方法 $.ajax() 进行数据的提交与接收，同时，将两个全局的 Ajax 事件 ajaxStart() 与 ajaxStop() 绑定到页面中元素。当触发 $.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，如下列代码：

```
... 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 的方法 $.ajax()，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，返回不了最新的数据。

在聊天室主页面中，当页面初次被加载时，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
});
```

由于在 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时时更新。

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，代码“$("#txtContent").val() + "<:" + this.id + ":>"”表示在已写入的聊天内容后面增加“<:”与“:>”，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替，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：

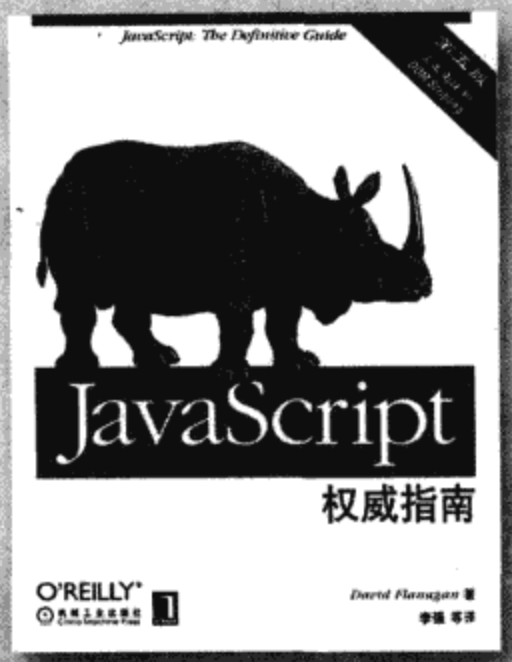
```
/// <summary>
/// 获取全部聊天记录
/// </summary>
/// <returns></returns>
private string AllChatList()
{
    //...省略部分代码
    strSendConent= strSendConent.Replace("<:", "<img src='Face/");
    strSendConent=strSendConent.Replace(":>", ".gif '/>");
    return strSendConent;
}
```

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时时的聊天数据。

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，学会如何运用所学，在 HTML 页面中，通过引入 jQuery 切割插件，实现任意大小图片切割方法。另外，通过介绍使用 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。



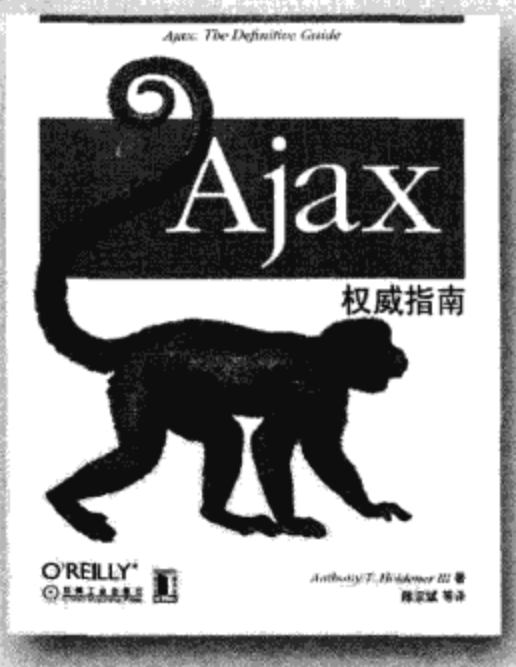
JavaScript权威指南（第五版）

作者：（美）费拉纳提
ISBN：978-7-111-21632-2
定价：109.00 元

《JavaScript权威指南第4版》中文版出版至今已有三年多的时间。这本《JaVaScript权威指南》连续印刷5次，销售数十万册，成为很多JavaScript学习者的必备宝典。由于其封面上是一只“爪哇犀牛”（封面上印上动物是原书出版公司O'Reilly一贯的风格），读者亲切地称其为“犀牛书”。

随着Ajax和Web 2.0技术的提出和流行，JavaScript再度受到广大技术人员的重视。但却没有一本从全新视角和层次来介绍JavaScript的参考书。《JavaScript高级程序设计》的出版填补了市场的空白，吸引了众多读者的目光，并且也获得相当不错的销售。而当时，《JavaScript权威指南》原书还处在改版之中。现在，《JaVascript权威指南第5版》虽然姗姗来迟，但必定会给众多期待本书的读者带来如沐春风的感觉。

Ajax权威指南

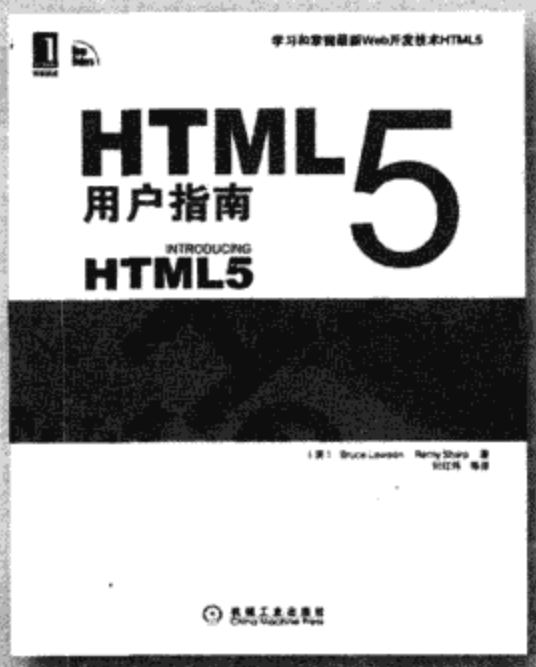


作者：（美）赫尔德尔
ISBN：978-7-111-27295-3
定价：125.00 元

Ajax是一种新技术，还是Web开发人员使用多年的老方法?实际上，二者兼而有之。本书不仅演示了经过检验而可靠的Web标准怎样使得Ajax成为可能，而且演示了这些老技术怎样给站点提供一种明确的现代Web 2. 0的感觉。

本书解释了如何使用像JavaScript、XML、CSS和XHTML这样的标准以及XMLHttpRequest对象来构建像桌面程序那样运行的、基于浏览器的Web应用程序。你将透彻理解今天的Web站点和应用程序的内部工作原理，并且学习怎样将Ajax知识用于高级浏览器搜索、Web服务、混搭等。通过学习本书，你将发现为什么利用Ajax从事开发更快速、更容易、代价更低。书中还包括以下主题：

- 把服务器端后端组件连接到浏览器中的用户界面。
- 加载和操纵XML文档，以及怎样用JSON替换XML。
- 操纵DOM(Document Object Model，文档对象模型)。
- 为可用性、功能性、可视化和可访问性设计Ajax界面。
- 站点导航布局。



HTML 5用户指南

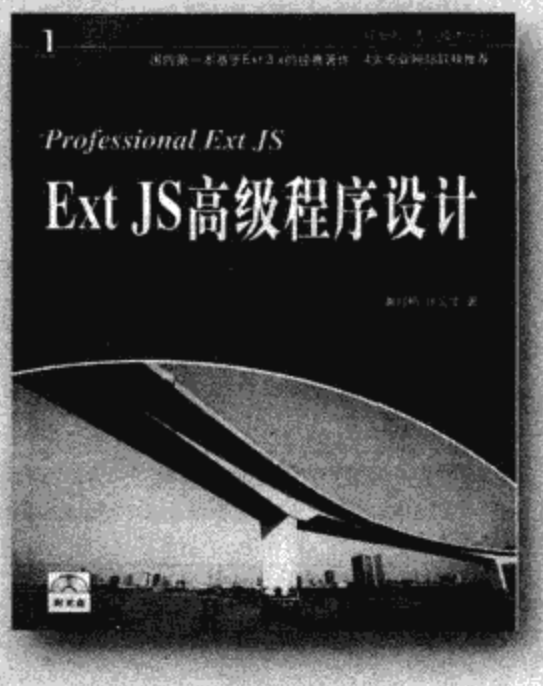
本书共分为10章，系统全面地介绍了HTML 5规范的核心内容，以及这些内容在当前浏览器中的支持情况，并告知开发者如何在当前的环境下应用这些功能，开发出漂亮的Web应用。本书短小精悍，但是信息量巨大；本书注重实践，其中的示例代码都具有很强的启发性和可操作性。对于初学者，本书是了解HTML 5的很好的入门材料；而对于想要了解HTML 5知识和应用的有经验的Web开发者，阅读本书更是轻车熟路，能进一步加深对Web开发和HTML 5的理解。

作者：（美）罗森，夏普
ISBN：978-7-111-32278-8
定价：39.00 元

Ext JS 高级程序设计

本书对Ext JS的核心知识以及中高级开发者在开发过程中会经常遇到的疑难问题进行了深入分析和探讨。本书不仅详细讲解了Ext JS 3.0中新增的用于Web 2.0网站开发的核心包Ext Core，以及给Ext JS带来革命性变化的Ext.Direct，而且还全面透彻地介绍了Ext.Data.Store、Ext UI、Ext扩展、Ext插件和调试等知识。实用性和可操作性强，各个知识点都配有实用的案例，并给出了最佳实践。全书最后以一个单页面的大型案例结束，以迭代的方式重现了该案例的实现过程，有助于读者融会贯通，将理论与实践完美结合。

尤为值得一提的是，本书中的主要实例同时包含.NET和Java两个版本，适合所有.NET开发者和Java软件开发人员阅读。



作者：黄灯桥，徐会生
ISBN：978-7-111-28769-8
定价：69.00 元

专业成就人生
立体服务大众

HZ BOOKS

www.hzbook.com

填写读者调查表　加入华章书友会
获赠精彩技术书　参与活动和抽奖

尊敬的读者：

感谢您选择华章图书。为了聆听您的意见，以便我们能够为您提供更优秀的图书产品，敬请您抽出宝贵的时间填写本表，并按底部的地址邮寄给我们（您也可通过www.hzbook.com填写本表）。您将加入我们的“华章书友会”，及时获得新书资讯，免费参加书友会活动。我们将定期选出若干名热心读者，免费赠送我们出版的图书。请一定填写书名书号并留全您的联系信息，以便我们联络您，谢谢！

书名：　　书号：7-111-(　　)　　.

姓名：	性别：☐男　☐女	年龄：	职业：
通信地址：		E-mail：	
电话：	手机：	邮编：	

1. 您是如何获知本书的：

☐朋友推荐　☐书店　☐图书目录　☐杂志、报纸、网络等　☐其他

2. 您从哪里购买本书：

☐新华书店　☐计算机专业书店　☐网上书店　☐其他

3. 您对本书的评价是：

技术内容	☐很好	☐一般	☐较差	☐理由_____
文字质量	☐很好	☐一般	☐较差	☐理由_____
版式封面	☐很好	☐一般	☐较差	☐理由_____
印装质量	☐很好	☐一般	☐较差	☐理由_____
图书定价	☐太高	☐合适	☐较低	☐理由_____

4. 您希望我们的图书在哪些方面进行改进？

_____

_____

5. 您最希望我们出版哪方面的图书？如果有英文版请写出书名。

_____

_____

6. 您有没有写作或翻译技术图书的想法？

☐是，我的计划是_____　☐否

7. 您希望获取图书信息的形式：

☐邮件　☐信函　☐短信　☐其他_____

请寄：北京市西城区百万庄南街1号　机械工业出版社　华章公司　计算机图书策划部收

邮编：100037　电话：（010）88379512　传真：（010）68311602　E-mail: hzjsj@hzbook.com