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

实戰

基于最新版 jQuery

陶国荣 著

jQuery: The Definitive Guide

jQuery 权威指南

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
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

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
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

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。

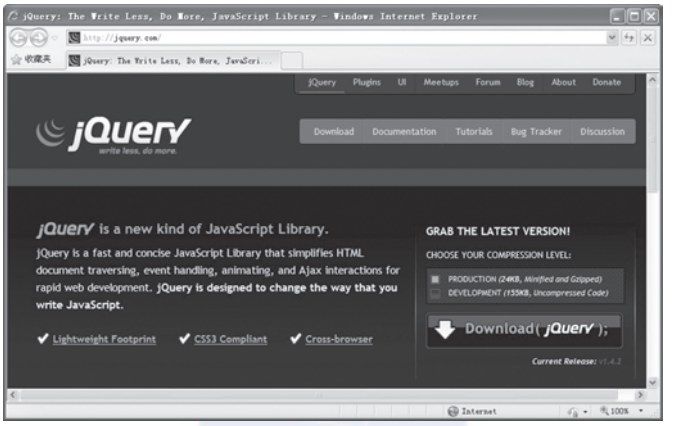


图 1-1　jQuery 的官方网站

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

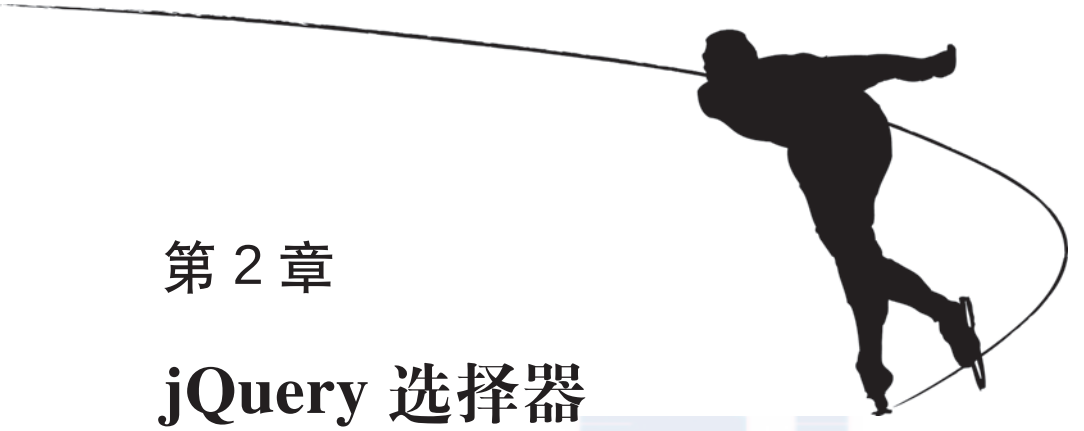
HTML 文件 1-4.html 执行后，最终实现的页面效果如图 1-5 所示。

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。

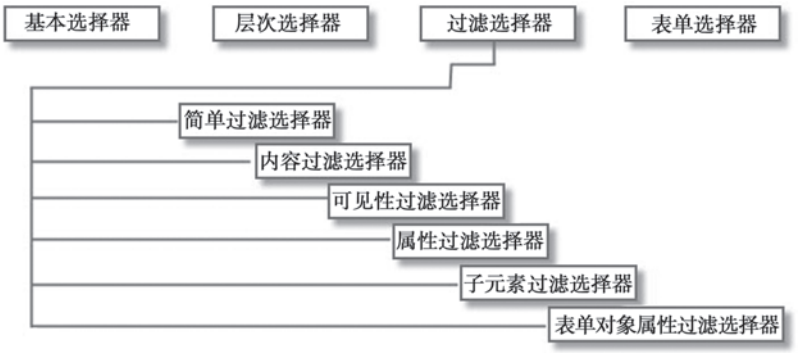


图 2-3　jQuery 选择器分类示意图

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

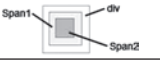
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<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>
```

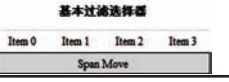
（3）页面效果

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	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
$("li:last") .addClass("GetFocus");	增加最后一个元素的类别	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
$("li:not(.NotClass)") .addClass("GetFocus");	增加去除所有与给定选择器匹配的元素类别	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
$("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	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
$("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	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move
$("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	基本过滤选择器 Item 0 Item 1 Item 2 Item 3 Span Move

（续）

关键代码	功能描述	页面效果
$("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

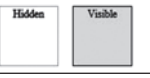
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 可见性过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></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	
$("div[title='A']") .show(3000);	显示所有属性 title 的值是 "A" 的元素	
$("div[title!='A']") .show(3000);	显示所有属性 title 的值不是 "A" 的元素	
$("div[title^='A']") .show(3000);	显示所有属性 title 的值以 "A" 开始的元素	
$("div[title$='C']") .show(3000);	显示所有属性 title 的值以 "C" 结束的元素	
$("div[title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 的元素	
$("div[id='divAB'][title*='B']") .show(3000);	显示所有属性 title 的值中含有 "B" 且属性 id 的值是 "divAB" 的元素	

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

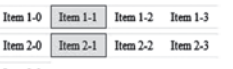
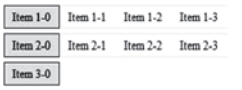
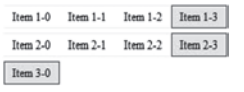
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用 jQuery 子元素过滤选择器 </title>
    <script language="javascript" type="text/javascript"
            src="Jscript/jquery-1.4.2.min.js"></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	
$("#form1 input:disabled") .addClass("GetFocus");	增加表单中所有属性为不可用的元素类别	
$("#form1 input:checked") .addClass("GetFocus");	增加表单中所有被选中的元素类别	
$("#Span2").html(" 选中项是：'+ $("select option:selected").text());	显示表单中所有被选中 option 的元素内容	

2.2.9　表单选择器

无论是提交还是传递数据，表单在页面中的作用是显而易见的。通过表单进行数据的提交或处理，在前端页面开发中占据重要地位。因此，为了使用户能更加方便地、高效地使用表单，在 jQuery 选择器中引入表单选择器，该选择器专为表单量身打造，通过它可以在页面中快速定位某表单对象。其详细的说明如表 2-17 所示。

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 write less, do more.
$("#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

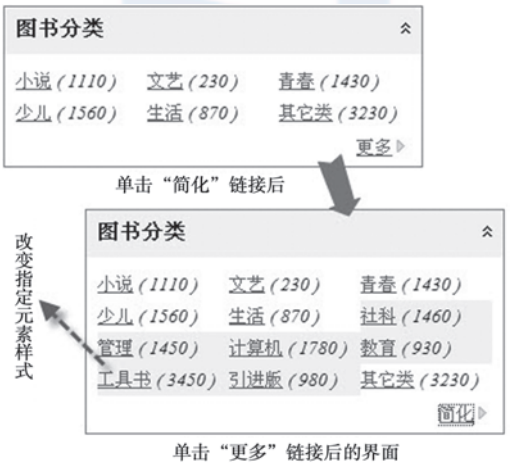


图 2-5 单击超级链接“简化”和“更多”后的界面

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。

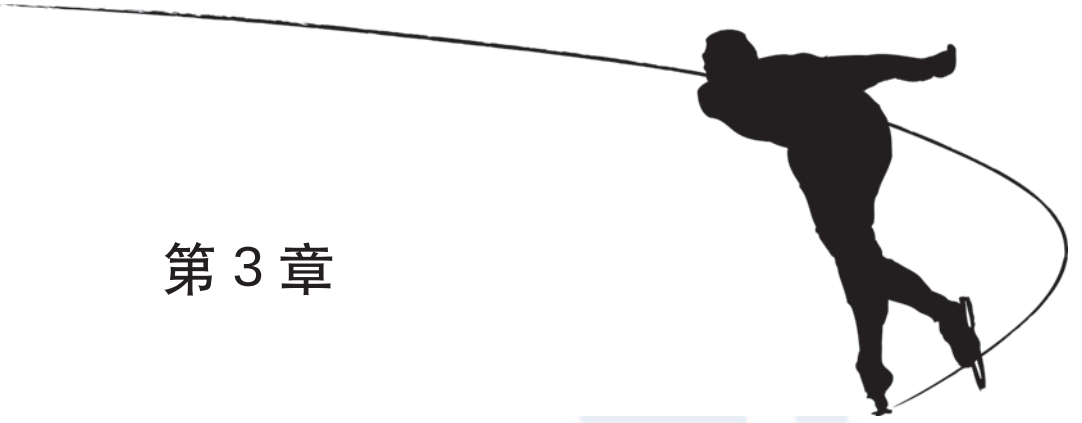
在代码中，通过如下的代码实现指定内容的隐藏：

```
$("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。

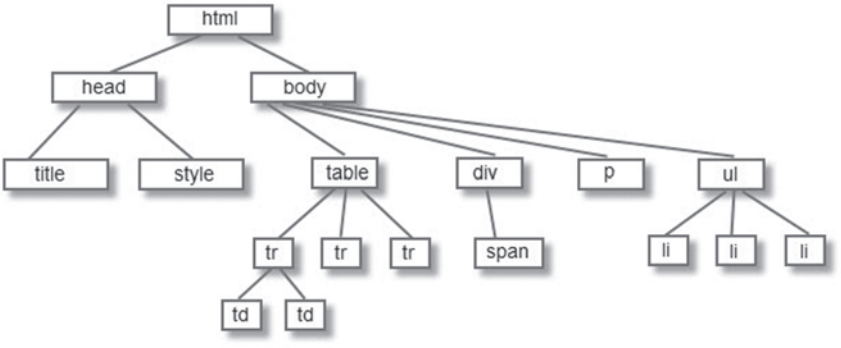


图 3-2　DOM 树状模型

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。

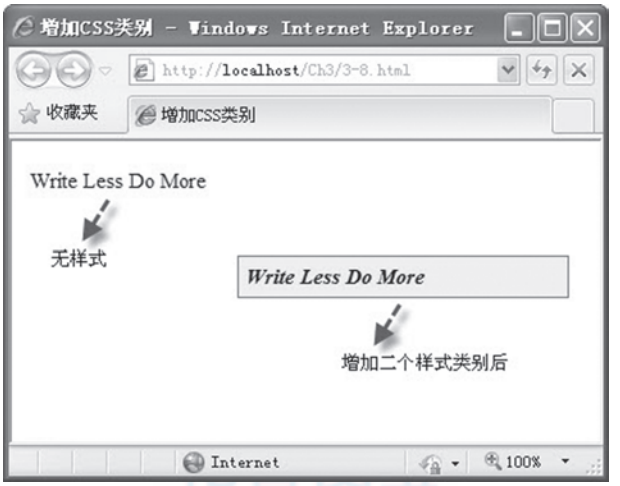


图 3-9　增加 CSS 类别

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");
```

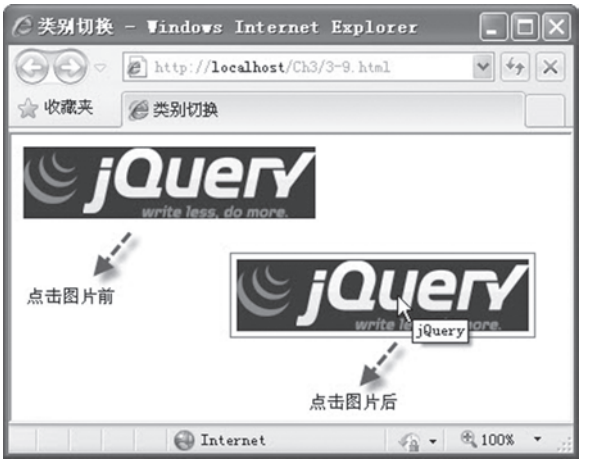


图 3-10 类别切换

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。

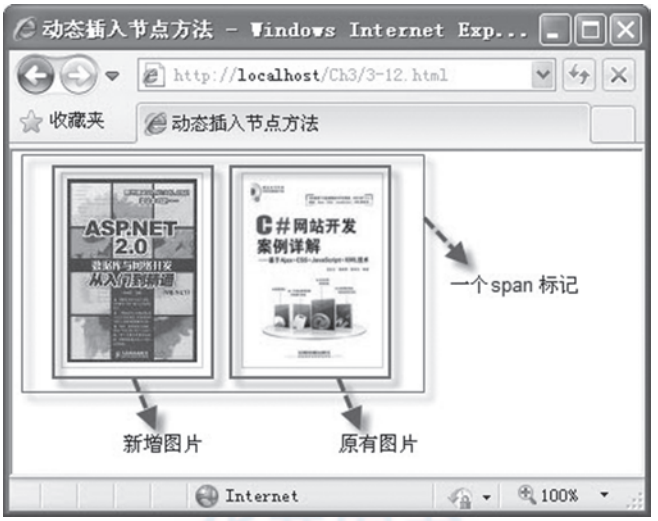


图 3-13　插入节点 (二)

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。

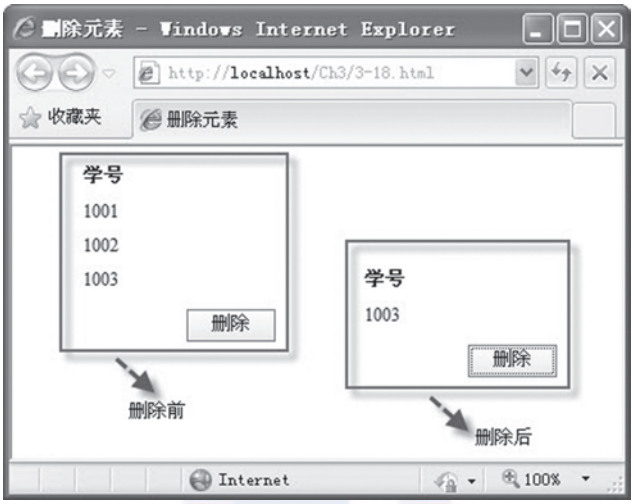


图 3-19　删除元素

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。

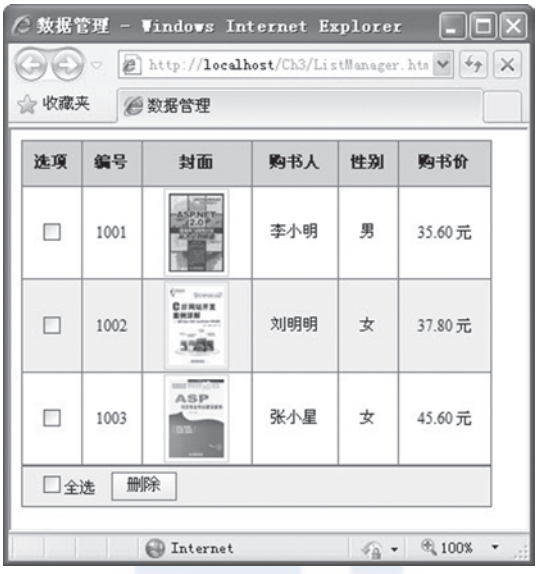


图 3-20　隔行变色展示数据

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。

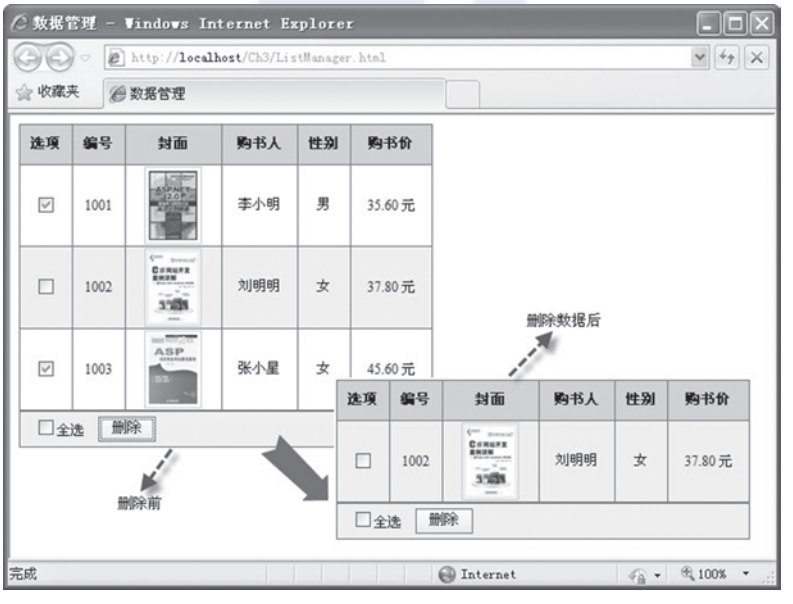


图 3-21　删除数据的前后对比

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。

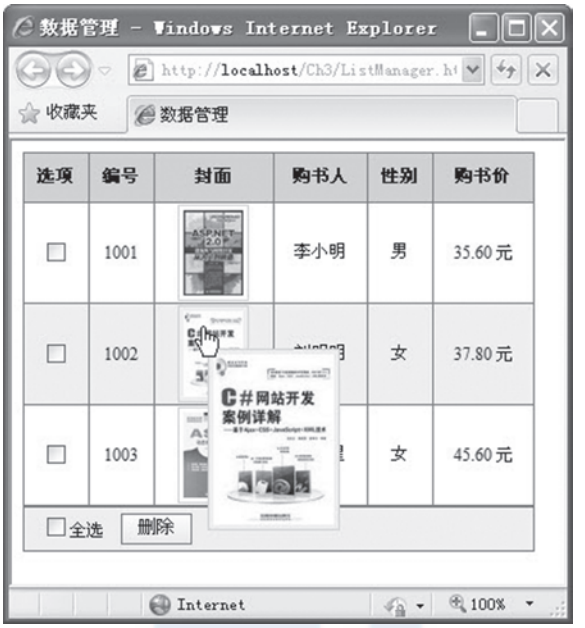


图 3-22　图片预览的效果

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。

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