用更少的代码 做更多的事情

巧用jQuery

吴超 张帅 编著

体验jQuery非凡魅力
感受前台编程乐趣

人民邮电出版社
POSTS & TELECOM PRESS

巧用jQuery

● 吴超 张帅 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

巧用JQuery / 吴超，张帅编著. —北京：人民邮电出版社，2009.8
ISBN 978-7-115-20970-2

Ⅰ. 巧… Ⅱ. ①吴…②张… Ⅲ. JAVA语言－程序设计
Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第090450号

内 容 提 要

jQuery 是近年来非常流行的 JavaScript 框架，它优雅、简洁、高效，深受开发者的青睐。本书为拥有 JavaScript 基础的读者介绍了如何利用 jQuery 框架编写 Web 页面。本书前 5 章为学习 jQuery 提供了必要的基础，包括 jQuery 核心操作、Ajax 传输、插件用法和页面动态性等。第 6～19 章分别介绍了如何利用 jQuery 来实现各种页面操作和效果，并通过大量的实例和用法演示，对读者提高自己的 Web 页面访问效果有很大的帮助。

本书适合具备基本 HTML 和 CSS 知识，并熟悉 JavaScript 的 Web 界面开发人员阅读。

巧用 jQuery

◆ 编　　著　吴　超　张　帅
责任编辑　刘　浩

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：20.75
字数：522 千字　　2009 年 8 月第 1 版
印数：1－3 500 册　　2009 年 8 月河北第 1 次印刷

ISBN 978-7-115-20970-2/TP

定价：42.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

jQuery 并不复杂，它的设计者是崇尚简洁的，这种设计思路在你开始写第一行 jQuery 代码的时候就能体会到。很多开发者在使用了 jQuery 之后，都对它赞不绝口，甚至宣称不可能有比它更快速的 JavaScript 编写方式了。

编写程序是一件苦差事，需要程序员查阅大量文档并订正修改。特别是 JavaScript 的开发，语言本身虽然简单，但是要实现某一简单的功能往往也需要冗长的代码，再加上调试的不便，所以很多程序员并不太喜欢 JavaScript 语言。

但是在 Web 的时代中，JavaScript 又如此重要，甚至对于某些以用户体验为第一目标的 Web 站点来说，客户端的 JavaScript 要比服务器端的代码更为重要。此时将 JavaScript 做更高一个层次上的封装和改进，对开发者来说非常有必要。

于是，就出现了各种 JavaScript 库，比如 Prototype、DWR 以及 jQuery 等。这些库各有所长，有些更擅长 UI，有些更擅长 Ajax。而 jQuery 能在众多库中脱颖而出的最大原因，在于它的简洁，这种简洁给代码带来了一种优雅的气质，同时对于编写者来说，也使得 JavaScript 的开发成为了一种享受。

当然，简洁的设计并不代表功能的匮乏，通过本书，你可以发现 jQuery 用简单的方式就实现了极丰富的效果。在 jQuery.js 中几十 KB 的代码就实现了这样的功能，实在让人惊叹设计者高超的设计能力。而在这几十 KB 的代码之外，jQuery 还有众多的支持者为它开发各式各样的插件，使得它几乎无所不能。

想要体会 jQuery 的简洁而又强大的功能吗？想要享受编写代码的快感吗？请继续翻阅本书。

本书由吴超、张帅主编，参加编写的还有杨轶、苏啸鸣、程达、徐元浩、潘贤敏、李斌、赵静、金雯斌、王璇、段坤、吴戈、龚中华、刘兆宏、张兵、季建华、钟晓媛、汪洪、陈功杰、刘福刚、何伟、石霞、刘梨平等，在此表示感谢！由于时间仓促，书中难免存在不足，请广大读者批评指正（电子函件：book_better@sina.com）。

编者

2009 年 6 月

目　录

第 1 章　初识 jQuery ····· 1
1.1　优雅简洁高效的 jQuery ····· 1
1.2　体会 jQuery 的魅力 ····· 2
1.2.1　$选取符 ····· 2
1.2.2　操作 DOM ····· 3
1.2.3　链式写法 ····· 7
1.2.4　Document ready 事件 ····· 8
1.2.5　JavaScript 库共存 ····· 9
1.3　动手编写 jQuery ····· 10
1.3.1　下载 jQuery ····· 10
1.3.2　编写 jQuery 的 HelloWorld ····· 11
1.3.3　改变多个元素的行为 ····· 12
1.3.4　实现页面动态性 ····· 13
1.4　用 Firebug 调试 jQuery 代码 ····· 14
1.5　用 Eclipse 开发 jQuery ····· 18
1.6　Eclipse 的 jQueryWTP 插件 ····· 22

第 2 章　jQuery 核心操作 ····· 23
2.1　运行核心——$(document).ready() ···· 23
2.2　快速选择页面元素 ····· 25
2.3　灵活控制元素和属性 ····· 30
2.4　动态改变页面 CSS ····· 33
2.4.1　jQuery 处理页面 CSS 的方法 ····· 33
2.4.2　操作 CSS 的例子 ····· 35
2.5　巧用 jQuery 事件 ····· 37
2.5.1　jQuery 事件处理方法 ····· 37
2.5.2　jQuery 捕捉鼠标位置 ····· 41
2.5.3　jQuery 事件实例 ····· 42

第 3 章　简化 Ajax 开发 ····· 54
3.1　用 jQuery 实现 Ajax ····· 54
3.1.1　load()方法 ····· 55
3.1.2　post()、get()和 Ajax()方法 ····· 57
3.2　用 jQuery 实现聊天器 ····· 58

第 4 章　巧用 jQuery 插件 ····· 65
4.1　jQuery 插件的使用方法 ····· 65
4.2　编写 jQuery 插件 ····· 67
4.3　常用 jQuery 插件 ····· 71
4.3.1　插件类别 ····· 71
4.3.2　优秀插件推荐 ····· 72

第 5 章　实现动态效果 ····· 86
5.1　jQuery 实现页面动态效果 ····· 86
5.2　jQuery 动态效果 API ····· 89
5.2.1　基本方法 ····· 89
5.2.2　滑动 ····· 91
5.2.3　淡入淡出 ····· 93
5.2.4　自定义 ····· 94
5.3　jQuery UI 库 ····· 99

第 6 章　展现图像 ····· 101
6.1　图片动态弹出效果 ····· 101
6.2　实现图片顺序加载 ····· 105
6.3　绚丽的 Flash 效果相册 ····· 109
6.4　巧用插件实现华丽的相册 ····· 113
6.5　实现幻灯相册 ····· 115
6.6　动态的图文结合 ····· 118
6.7　让图片更像照片 ····· 121
6.8　图片剪切 ····· 122
6.9　图片预览效果 ····· 123
6.10　图片局部平移 ····· 124
6.11　图片动态切换效果 ····· 125
6.12　圆角效果 ····· 126
6.13　图片放大镜 ····· 127
6.14　图片三维视图 ····· 129

第 7 章　导航与菜单设计 ····· 130
7.1　用 jQuery 实现 Tab 标签 ····· 130

7.2 LavaLamp 导航效果 …… 132
7.3 绚丽的 Coda Slider …… 133
7.4 类苹果 Dock 菜单 …… 137
7.5 滑动效果菜单 …… 138
7.6 手风琴拉伸菜单 …… 140
7.7 Superfish 动态菜单 …… 141
7.8 多层次菜单 …… 142
7.9 折叠菜单 …… 143
7.10 TreeView 树形导航 …… 145
7.11 jGlideMenu 滑动导航 …… 146
7.12 使用 idTabs 插件 …… 147
7.13 鼠标右键弹出菜单 …… 149

第 8 章 巧用 jQuery 操作 Google 地图 …… 151
8.1 jQuery 操作地图 …… 151
8.2 结合 PHP 创建地图标记 …… 155
8.3 利用 jMaps 插件操作 Google 地图 …… 157

第 9 章 提升页面体验 …… 159
9.1 jTip 动态提示 …… 159
9.2 jGrowl 提示效果 …… 161
9.3 添加社会化标签 …… 162
9.4 图片延迟加载 …… 163
9.5 动态新闻提示 …… 164
9.6 自动设定行高 …… 165
9.7 对链接进行预览 …… 166
9.8 链接区域扩展 …… 167
9.9 实现页面快捷键 …… 168
9.10 翻页效果 …… 169
9.11 禁止选择页面文字 …… 172
9.12 页面内容动态显示与隐藏 …… 172
9.13 页面自动滚动 …… 173
9.14 内容的下滑展示 …… 175
9.15 Web OS 交互界面 …… 176

第 10 章 处理表单 …… 185
10.1 jQuery 处理表单元素的技巧 …… 185
10.2 jQuery 实现多个文件上传 …… 187
10.3 页面投票 …… 188
10.4 处理电子表格 …… 189
10.5 定义输入格式 …… 191
10.6 限制输入类型 …… 192
10.7 表单验证 …… 193
10.8 Ajax 方式的文件上传 …… 195
10.9 jqUploader 文件上传 …… 196
10.10 为上传框增加样式 …… 198
10.11 页面进度条 …… 199
10.12 样式多样的下拉框 …… 200
10.13 级联选择框 …… 205
10.14 级联下拉框 …… 207
10.15 获得焦点时增加样式 …… 208
10.16 Ajax Form 表单 …… 209
10.17 良好样式的表单 …… 211
10.18 调整型输入 …… 216
10.19 为输入框加上水印效果 …… 217
10.20 Shift 快捷键的支持 …… 217
10.21 改进的取值方式 …… 219
10.22 Pin 标记 …… 219
10.23 用 jQuery 选取时间 …… 220
10.24 用 jQuery 选取颜色 …… 221
10.25 用 jQuery 选取日期 …… 223
10.26 用 jquery.jframe 实现 frame 行为 …… 225

第 11 章 实现绚丽的页面效果 …… 227
11.1 JSS 定义页面样式 …… 227
11.2 页面直播效果 …… 229
11.3 图片弹跳 …… 231
11.4 样式动态切换 …… 232
11.5 按需加载 JavaScript 和 CSS …… 234
11.6 操作样式规则 …… 234
11.7 边框效果 …… 236
11.8 Nifty 圆角 …… 236
11.9 实现定时器 …… 237
11.10 Easydrag 实现拖放 …… 239
11.11 实现 Portal 效果 …… 240

11.12 Draggable 插件 …… 241
11.13 jqDnR 拖放与拉伸 …… 242

第 12 章 修正浏览器 …… 245
12.1 PNG 透明 …… 245
12.2 修正链接样式 …… 246
12.3 提高页面可访问性 …… 246
12.4 处理鼠标滚轮事件 …… 247

第 13 章 轻松实现页面提示 …… 249
13.1 提示框效果 …… 249
13.2 内嵌确认框 …… 250
13.3 弹出层做提示 …… 251
13.4 SimpleModal 弹出层 …… 253
13.5 表单输入提示 …… 254
13.6 ClueTip 做 Ajax 提示 …… 255
13.7 BetterTip 提示 …… 255
13.8 简单提示 …… 256

第 14 章 实现页面编辑 …… 257
14.1 用 Jeditable 实现内嵌编辑 …… 257
14.2 用 tEditable 编辑单元格 …… 258
14.3 表格内容编辑 …… 259
14.4 Inplace 做内联编辑 …… 260
14.5 在线编辑器 …… 261

第 15 章 实现页面多媒体 …… 263
15.1 Multimedia Portfolio 展现多媒体内容 …… 263
15.2 用 jmedia 展现多媒体 …… 264
15.3 MP3 播放 …… 265
15.4 多媒体播放 …… 266
15.5 Flash 播放 …… 268
15.6 Quicktime 播放 …… 269
15.7 展现 SVG 图像 …… 270
15.8 强手棋游戏 …… 272

第 16 章 操作表格 …… 283
16.1 用 Ingrid 实现 Flash 效果表格 …… 283
16.2 树形表格 …… 285
16.3 可扩展表格 …… 286
16.4 动态表格样式 …… 288
16.5 表格排序 …… 289
16.6 表格位置样式 …… 291
16.7 调整表格列宽 …… 292

第 17 章 做数据搜索 …… 295
17.1 快速搜索功能 …… 295
17.2 搜索自动提示 …… 297
17.3 搜索自动提示插件 …… 299

第 18 章 处理数据 …… 302
18.1 jquick 操作 DOM 树 …… 302
18.2 elementReady 响应元素事件 …… 303
18.3 jQuery Chart 做柱状图 …… 303
18.4 Yahoo Pipes …… 304
18.5 RSS 解析器 …… 309
18.6 XSLT 转换 …… 309
18.7 Taconite 操作页面数据 …… 310
18.8 将 XML 数据转化为 JSON 数据 …… 313
18.9 gFeed 解析 …… 313
18.10 AJAX SLT …… 314

第 19 章 jQuery 与其他技术结合 …… 316
19.1 Ajax+Java 登录 …… 316
19.2 使用 jQuery 来创建 Silverlight …… 320
19.3 Struts2 中使用 jQuery …… 323
19.4 Dreamweaver 的 jQuery 插件 …… 324

1 初识 jQuery

jQuery 是一个 JavaScript 库，它有助于简化 JavaScript 和 Ajax 编程。利用 jQuery 能够优雅、简洁、高效地编写 JavaScript 代码。优雅、简洁、高效这 3 个形容词用在 jQuery 身上一点都不夸张，不相信吗？请跟着我们进入 jQuery 的世界吧。

1.1 优雅简洁高效的 jQuery

2006 年 1 月 John Resig 等人创建了 jQuery，John Resig 一直在 Moziila 工作，对浏览器及 JavaScript 非常熟悉，也深知 JavaScript 存在一些内在不足，他希望开发出一套并非全能，但是在访问 DOM 元素等方面快速高效的 JavaScript 库，于是就有了 jQuery。

jQuery 不是第一个 JavaScript 库，在 jQuery 之前已经涌现了包括 Prototype、Scriptaculous 和 DWR 在内的优秀 JavaScript 库。利用 Prototype 能实现非常绚丽的页面效果，比如动画和拖放等。与这些 JavaScript 库相比，jQuery 最大的优点就是简洁实用。jQuery 的原理是独一无二的，它的目的就是保证代码简洁并可重用。它可以用很少的几行代码创建出漂亮的页面效果，从这一点上来说，jQuery 是一个了不起的 JavaScript 库。

jQuery 能保证代码简洁易读，开发者再也不必编写一大堆重复的循环代码和 DOM 脚本库调用。使用 jQuery 可以把握问题的要点，并使用最少的代码实现想要的功能。有人甚至说：采用 jQuery 之后，用大约 10 行代码就能完成 JavaScript 中 20 行左右的代码所能完成的功能，而这已达到一个编程语言效率的极限了。

下面举例说明 jQuery 的简洁性。比如，要将所有 id 为 external_links 的链接<a>加上鼠标点击事件，在这些链接被点击时显示当前链接的地址，如果用传统的 JavaScript，就需要如下的代码。

```
var external_links = document.getElementById('external_links');          //找到 id 为 external_links 的元素
var links = external_links.getElementsByTagName('a');                    //找到 external_links 下的<a>元素
for (var i=0;i < links.length;i++) {                                      //遍历这些元素
    var link = links.item(i);                                             //获取元素值
    link.onclick = function() {                                           //为元素增加鼠标点击事件响应
        return confirm('You are going to visit: ' + this.href);           //显示当前连接地址
    };
}
```

而如果用 jQuery 完成同样的功能，只需要以下代码就可以了。

```
$('#external_links a').click(function() { //找到 id 为 external_links 的<a>元素，并增加鼠标点击事件响应
    return confirm('You are going to visit: ' + this.href);
});
```

虽然 jQuery 的目标是简洁优雅，但是 jQuery 在设计上拥有良好的扩展性。可以通过插件（与 UI 相关的插件已经被打包成 jQuery UI：http://ui.jquery.com）实现类似 Yahoo UI 或者 ExtJS 一样的 UI 效果。

1.2 体会 jQuery 的魅力

上节说了 jQuery 这么多的好处，那么它究竟有多么简洁和优雅呢？让我们来看看 jQuery 的用法，体会使用 jQuery 开发的便捷吧。

1.2.1 $选取符

下面是一句用 jQuery 写的 JavaScript 语句，功能是选取本页面中所有的 元素，然后将这些 div 都加上一个名为“special”的 CSS 样式。

```
$("div").addClass("special");
```

可以想象一下，如果用纯粹的 JavaScript，需要多少行代码才能完成这项工作，而在 jQuery 中，一条简单的语句就完成了。

$是 jQuery 的特殊字符，用于声明 jQuery 对象。$也是 jQuery 选取元素的符号，这里选取的是 div 元素。选取是多元素的，而不仅仅只取第一个。如果网页上有 5 个 div 元素，那么在这里都会被选取出来。addClass()是 jQuery 对象的一个方法，其功能是在被选取的元素对象上加上 CSS 样式。

选取标记$可能是 jQuery 知识中最重要的内容，它是 jQuery 对象的缩写，上述代码也可以写成如下样式。

```
jQuery("div").addClass("special");
```

一般都用$来代替 jQuery，让代码看起来非常简洁。

jQuery 的选取符采用了与 CSS 同样的选取符语法，下面再举几个例子来进行说明。

下面是一段原始的 HTML：

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

采用一系列的选取符来选择页面内的不同元素，选中的部分会用粗体显示出来。

（1）$("div")：将选取所有的 元素。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
```

```
<p>...</p>
<p>...</p>
</div>
</div>
```

（2）$("#body")：选取 id 为 body 的元素。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

（3）$("div #body")：选取 id 为 body 的 。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

（4）$("div .contents p")：选取 class 为 contents 的 所有的下层<p>元素。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

（5）$("div > div")：选取被 包裹的下一层 。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

（6）$("div :has(div)")：选取至少包住一个子 的 元素。

```
<div id="body">
<h2>Some Header</h2>
<div class="contents">
<p>...</p>
<p>...</p>
</div>
</div>
```

选取了这些元素之后，可以利用 jQuery 对象的方法对这些元素做一些改变。

1.2.2 操作 DOM

jQuery 能够方便地对 DOM 进行操作，比如添加、修改或删除节点，为节点添加事件处理等。

下面将展示一些 jQuery 操作 DOM 以改变网页元素的例子。

（1）选取所有包含 target 属性的<a>，并且在其节点下添加一段文字。

```
$("a[target]").append("(Opens in New Window)");
```

- $("a[target]")选取了所有包含 target 属性的<a>；
- append()方法用于在节点之下加入新的文本。

原始的 HTML 代码如下：

```
<a href="http://jsgears.com">jsGears</a>
<a href=http://google.com target="_blank">Google</a>
<a href=http://amazon.com target="_blank">Amazon</a>
```

执行以上的 jQuery 之后，所有包含 target 属性的<a>都会添加文字“(Opens in New Window)”。

```
<a href="http://jsgears.com">jsGears</a>
<a href=http://google.com target="_blank">Google(Opens in New Window)</a>
<a href=http://amazon.com target="_blank">Amazon(Opens in New Window)</a>
```

（2）选取 id 为 body 的元素，并且修改两个 CSS 属性。

```
$("#body").CSS({
border: "1px solid green",
height: "40px"
});
```

- $("#body")选取 id 为 body 的元素；
- CSS()方法用于修改页面元素的 CSS 样式。在这里定义了新的 CSS 内容。

```
{
border: "1px solid green",
height: "40px"
}
```

下面是一段原始的 HTML 代码：

```
<div id="body">
...
</div>
```

选取 id 为 body 的元素并修改 CSS 后的结果如下。

```
<div id="body" style="border:1px solid green; height: 40px">
...
</div>
```

（3）在提交表单的时候，判断 username 字段是否为空，如果为空，则显示 help 区块内的文字。

```
$("form").submit(function(){
if($("input #username").val()=="")
$("span.help").show();
});
```

- $("form")找到 form 元素；
- submit(function(){…});定义了此 form 提交之后（即发生 submit 事件的时候）执行的行为。

在方法中，首先取出名为 username 的 input 元素，判断其值是否为空。

```
if($("input #username").val()=="")
```

如果是的话，则找出 span.help 元素，然后用 show()方法将其显示，代码如下。

```
$("span.help").show();
```

看一下目标的 HTML 代码，一开始 span.help 是隐藏的，如果没有输入 username，则会显示如下内容。

```
<style type="text/CSS">
.help{display: none}
</style>
<form>
<label for="username">请输入名字</label>
<input type="text" id="username" name="username"/>
<span class="help">错误！  请输入您的名字</span>
</form>
```

（4）当用户单击 id 为 open 的链接时，显示 id 为 menu 的区块，并返回 false。

```
$("a #open").click(function(){
$("#menu").show();
return false;
});
```

作用于以下的 HTML 代码。

```
<style type="text/CSS">
#menu{display :none}
</style>
<a id="open" href="#">控制面板</a>
<ul id="menu">
<li><a href="#1">控制面板首页</a></li>
<li><a href="#2">编辑个人资料</a></li>
<li><a href="#3">个人空间管理</a></li>
</ul>
```

首先选取 id 为 open 的<a>：

```
$("a #open")
```

指定其单击事件发生时的方法，代码如下。

```
click(function(){});
```

此方法中，找出 id 为 menu 的元素，然后显示此元素。

```
$("#menu").show();
```

最后返回 false。

```
return false;
```

（5）将 id 为 menu 的区块以动态下拉效果显示出来。

```
$("#menu").slide Down("fast");
```

作用于以下的 HTML 代码，将原本隐藏的菜单用动态下拉的方式显示出来。

```
<style type="text/CSS">
#menu{display: none}
</style>
```

```
<ulid="menu">
<li><a href="#1">控制面板首页</a></li>
<li><a href="#2">编辑个人资料</a></li>
<li><a href="#3">个人空间管理</a></li>
</ul>
```

- $("#menu")选取 id 为 menu 的 HTML 元素；
- slideDown("fast")方法是 jQuery 的方法，用于动态显示，参数 fast 表示快速展示。

（6）将所有的 渐变为 300px 宽，文字与边界 20px 宽。

```
$("div").animate({
width:'300px',
padding:'20px'
},'slow');
```

作用于以下的 HTML 代码。

```
<div style="width:100px;border:solid1pxred;">
Hello world!
</div>
```

- $("div")选取所有的 ；
- Animate()方法提供动态的动画效果。这里指定了一个 CSS 样式，并指定显示速度为 slow。

jQuery 本身的 animate 函数并没有提供很多的动画效果，但是可以通过插件的形式，得到更多的动画效果。

（7）具有动态效果的回调函数，将所有的 以 0.5s 的动态效果隐藏之后，再以 0.5s 的动态效果显示。其中$(this)是方法调用方的元素。

```
$("div").hide(500,function(){
//$(this)是每一个各别的<div>
$(this).show(500);
});
```

可以作用于以下的 HTML。

```
<div style="width: 100px; border: solid 1px red;">
Hello world!
</div>
<div style="width:100px;border:solid1pxred;">
jsGears.com!
</div>
```

- $("div")选取所有的 ；
- hide 方法用于隐藏页面元素。方法有两个参数，第一个参数指定了隐藏的时间（单位是 ms），第二个参数为回调方法，在这个方法中重新对隐藏的元素进行显示。

（8）获取 sample.html 并且找出其中所有 下一层中的<h1>，将<h1>中的内容写入 id 为 body 的元素中。

```
$("#body").load("sample.html div>h1");
```

下面是一段原始的 HTML 代码。

```
<div id="body"></div>
sample.html 的片段：
<div>
<h1>Hello world!</h1>
<h2>ThisisH2</h2>
<h1>jsGears.com!</h1>
</div>
```

执行之后的结果如下所示。

```
<div id="body">
<h1>Hello world!</h1>
<h1>jsGears.com!</h1>
</div>
```

$("#body")选取 id 为 body 的元素，sample.html div>h1 选择了 sample.html 下 div 下的 h1，load 方法将选取出来的内容写入到#body 中。

（9）通过 getJSON()获得 JSON 格式的数据，并通过回调方法处理这些数据，代码如下。

```
$.getJSON("test.json", function(data){
for(var idx in data)
$("#menu").append("<li>"+data[idx]+"</li>");
});
```

下面是一段原始的 HTML 代码。

```
<ul id="menu">
<li>项目 1</li>
</ul>
```

test.json 的内容如下。

```
[
"Hello world!",
"jsGears.com!"
]
```

执行之后的结果如下所示。

```
<ul id="menu">
<li>项目 1</li>
<li>Hello world!</li>
<li>jsGears.com!</li>
</ul>
```

1.2.3 链式写法

jQuery 优雅的一个重要原因就是它具有独特的连续使用函数的写法（Chaining），当选取了一个或者一组元素之后，可以连续对这些元素进行多个处理。

例如，希望选取所有的 ，然后隐藏它们，修改文字颜色为蓝色，并将 以下拉的效果显示出来，那么就可以采用下面的 jQuery 代码。

```
$("div").hide();
$("div").CSS("color","blue");
$("div").slideDown();
```

这样的 3 行代码可以利用 jQuery 的连续函数的写法，用一行代码来替代，代码如下。

```
$("div").hide().CSS("color","blue").slideDown();
```

看起来似乎很神奇，其实这是因为 jQuery 的函数大多数的返回值是原本传入的页面元素本身，所以返回值就能一个接着一个的调用方法。

这种紧凑的写法，让 jQuery 的代码显得非常简洁。

在连续使用函数的写法中，有两个重要的函数需要特别注意。

（1）end()方法。这个方法执行之后，会返回“前一组找到的元素”。

（2）find()方法。与$()选择器类似，执行后找到所需要的元素，不同的是，$()会在整个文件内寻找元素，而 find()是在之前找到的元素中寻找此元素。

可以通过下面的代码来理解。

```
$("ul.open")          //找出文件内所有 class 为 open 的<ul>
.children("li")       //过滤出下一层所有的<li>
.addClass("open")     //对这些<li>新增一个 class
.end()                //回到前一次的搜索结果，就是所有的<ul>。
.find("a")            //再找出其中所有的<a>
.click(function(){    //对<a>新增事件处理
$(this).next().toggle();
return false;
})
.end();               //回到前一次搜索的结果
```

1.2.4 Document ready 事件

本小节介绍 jQuery 中一种很受好评的特性——jQuery 处理 document ready 事件的方法。在页面载入完成的时候，往往需要设定一些工作，比如设定输入焦点、显示一些信息等。此时一般是利用 JavaScript 的 window.onload 来处理，或是直接在 body 标签上加入 onload 事件处理函数。而在 jQuery 中提供了另外一种选择，请看以下的例子。

```
$(document).ready(function(){
alert('您好，欢迎您');
});
```

显示结果如图 1-1 所示。

图 1-1

先通过$()取得 document 对象，然后使用 ready 指定一个函数，这样就能在 document 的 ready 事件发生时，立即执行这个函数。

jQuery 的 document ready 事件模拟 DOM Content Loaded 事件。DOM Content Loaded 和 window.onload 的差别在于前者是在 DOM 载入完成之后触发，而后者是需要等到页面文件内的所有元素（包括图像等）全部载入完成之后才能触发。所以一般来说，window.onload 开始执行的时间要比 DOM Content Loaded 事件晚一些。

通常，并不需要在页面所有元素（比如图像）都载入完成之后再进行操作，这样会降低效率，所以在 DOM Content Loaded 事件内做处理是比较合适的做法。但是目前 Jnternet Explorer 的版本

（包括版本 6 和版本 7）均不支持 DOM Content Loaded 事件，所以 jQuery 使用了一些技巧来模拟 DOM Content Loaded 事件，以达到相同的效果。

DOM Content Loaded 和 window.onlad 的另一个差别在于，window.onload 并没有办法多次指定不同的方法来执行，最后指定的那个方法会覆盖之前的方法。比如，下面的代码在同一个网页中利用了多次 window.onload，结果是后面的方法会执行，而前面的方法被覆盖，没有执行。如果要两个方法都执行，则需要另外进行处理。

```
window.onload=function(){
alert('Hello world!');
};
window.onload=function(){
alert('您好！ ');
};
```

如果使用 jQuery，就会方便很多，看一下 jQuery 的写法：

```
$(document).ready(function(){
alert('Hello world!');
});
$(document).ready(function(){
alert('您好！ );
});
```

采用了 jQuery 的 document ready 函数之后，两个方法都会被执行，而且非常简单。

更让人惊喜的是，document ready 函数还有一种更为简洁的写法，如下所示。

```
$(function(){
alert('您好！ ');
});
```

直接把函数放到$()里面就行了。

前面很多地方用到了$()，包括刚才介绍的 document ready 和之前的选取功能都是通过$()来完成的。实际上，这个标记是 jQuery 对象的缩写，所以之前所有用到$()的地方都可以改写成 jQuery()，比如下面的代码所示。

```
jQuery(document).ready(function(){
jQuery("div").addClass("special");
});
```

1.2.5 JavaScript 库共存

一般来说，我们倾向于采用比较简便的$()来编写代码。但是在某种情况下，可能会无法使用$()，如在代码中使用了其他的 JavaScript 库，而且这个$已经被其他的 JavaScript 库定义和使用了。例如，在 prototype 中也使用了$()这样的方法名，这个时候在 jQuery 中使用$()就会产生冲突，而只能使用 jQuery()的形式，比如下面的代码：

```
jQuery.noConflict();
jQuery(document).ready(function(){
jQuery("div").addClass("special");
});
```

在使用 jQuery 之前，先使用 jQuery.noConflict()方法，就可以避免$()冲突的问题发生，然后再使用 jQuery 对象操作即可。此外，也可以自行给 jQuery 这个对象设定一个别名，代码如下所示。

```
var $j=jQuery.noConflict();
$j(document).ready(function(){
$j("div").addClass("special");
});
```

这里使用了一个变量来接收 jQuery.noConflict()的返回值，这个变量就成为了 jQuery 对象的一个别名。通过这个别名不仅避免了与其他 JavaScript 库的冲突，也可以将函数名称缩短。

1.3 动手编写 jQuery

编写 jQuery 代码的过程与编写普通 jQuery 的方法类似，区别在于：第一，需要引入 jQuery 库；第二，使用 jQuery 的特有写法编写程序，以下是具体的过程。

1.3.1 下载 jQuery

jQuery 目前的最新稳定版本为 1.2.6，在 http://docs.jquery.com/Release:jQuery_1.2.6 上可以找到 jQuery 1.2.6 的各种版本，如图 1-2 所示。

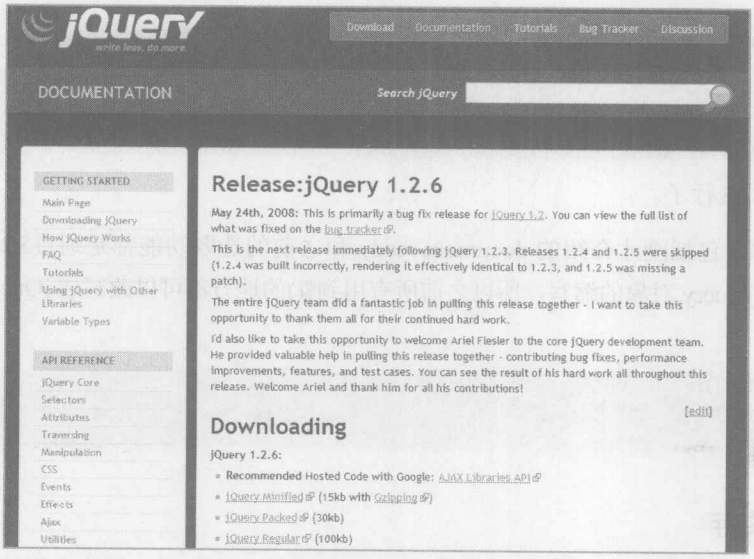

图 1-2

注意：在读者看到本书的时候，jQuery 又会有新的版本更新，但是版本之间会有比较好的兼容性，所以采用后续版本不会对本书的学习产生很大的影响。

在下载页面上，有以下 3 种不同的 jQuery 版本。

- **Minified**：适合在项目中使用。
- **Packed**：需要客户端来解压，但体积最小。