

第10章 jQuery 基础

随着 JavaScript、CSS、DOM 和 Ajax 等技术的不断进步，越来越多的开发者将一个又一个丰富多彩的功能进行封装，供更多的人在遇到类似情况时使用，jQuery 就是其中的优秀一员。从本章开始，陆续介绍 jQuery 的相关知识，本章作为介绍 jQuery 的第 1 章，将重点讲解 jQuery 的概念以及一些简单的基础运用。

10.1 jQuery 概述

本节重点介绍 jQuery 的概念，并通过实例来展示页面中引入 jQuery 的优势，以及如何下载和使用 jQuery。

10.1.1 jQuery 是什么

就像 9.4 节中提到的 Ajax 框架一样，简单地说，jQuery 是一个优秀的 JavaScript 框架，它能使用户更方便地处理 HTML 文档、events 事件、动画效果和 Ajax 交互等。它的出现极大地改变了开发者使用 JavaScript 的习惯，掀起了一场新的网页革命。

jQuery 由美国人 John Resig 于 2006 年最初创建，至今已吸引了来自世界各地的众多 JavaScript 高手加入其团队。最开始的时候，jQuery 所提供的功能非常有限，仅仅可以增强 CSS 的选择器功能。但随着时间的推移，jQuery 的新版本一个接一个地发布，它也越来越受到人们的关注。

如今 jQuery 已经发展到集各种 JavaScript、CSS、DOM 和 Ajax 功能于一体的强大框架，可以用简单的代码轻松地实现各种网页效果。它的宗旨就是让开发者写更少的代码，做更多的事情（Write less, do more）。

目前 jQuery 主要提供如下功能。

（1）访问页面框架的局部。这是第 5 章介绍的 DOM 模型所完成的主要工作之一，通过前面章节的示例也可以看到，DOM 获取页面中某个节点或者某一类节点有固定的方法，而 jQuery 则大大地简化了其操作的步骤。

（2）修改页面的表现（Presentation）。CSS 的主要功能就是通过样式风格来修改页面的表现。然而由于各个浏览器对 CSS3 标准的支持程度不同，使得很多 CSS 的特性没能很好地体现。jQuery 的出现很好地解决了这个问题，它通过封装好的 JavaScript 代码，使得各种浏览器都能很好地使用 CSS3 标准，极大地丰富了 CSS 的运用。

（3）更改页面的内容。通过强大而方便的 API，jQuery 可以很方便地修改页面的内容，包括文本的内容、插入新的图片、表单的选项，甚至整个页面的框架。

（4）响应事件。在第 6 章中介绍了 JavaScript 处理事件的相关方法，而引入 jQuery 之后，可以更加轻松地处理事件，而且开发人员不再需要考虑讨厌的浏览器兼容性问题。

（5）为页面添加动画。通常在页面中添加动画都需要开发大量的 JavaScript 代码，而 jQuery

大大简化了这个过程。jQuery 的库提供了大量可自定义参数的动画效果。

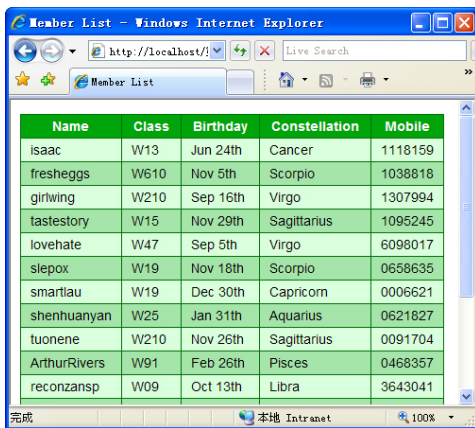
(6) 与服务器异步交互。类似 9.4 节中介绍的 Ajax 框架可以简化代码的编写, jQuery 也提供了一整套 Ajax 相关的操作, 大大方便了异步交互的开发和使用。

(7) 简化常用的 JavaScript 操作。jQuery 还提供了很多附加的功能来简化常用的 JavaScript 操作, 例如数组的操作、迭代运算等。

10.1.2 jQuery 的优势

下面通过“隔行变色的表格”来具体说明 jQuery 的优势。

页面中常常会遇到各种各样的数据表格, 例如学校的人员花名册, 公司的年度收入报表, 股市的行情统计, 等等。当表格的行列都很多, 并且数据量很大的时候, 单元格如果采用相同的背景色, 用户在实际使用时会感到凌乱。通常的解决办法就是采用隔行变色, 使得奇数行和偶数行的背景颜色不一样, 达到数据一目了然的目的, 如图 10.1 所示。



Name	Class	Birthday	Constellation	Mobile
isaac	W13	Jun 24th	Cancer	1118159
fresheggs	W610	Nov 5th	Scorpio	1038818
girlwing	W210	Sep 16th	Virgo	1307994
tastestory	W15	Nov 29th	Sagittarius	1095245
lovehate	W47	Sep 5th	Virgo	6098017
slepox	W19	Nov 18th	Scorpio	0658635
smartlau	W19	Dec 30th	Capricorn	0006621
shenhuanyan	W25	Jan 31th	Aquarius	0621827
tuonene	W210	Nov 26th	Sagittarius	0091704
ArthurRivers	W91	Feb 26th	Pisces	0468357
reconzansp	W09	Oct 13th	Libra	3643041

图 10.1 隔行变色的表格

对于纯 CSS 页面, 要实现隔行变色的效果, 通常是给偶数行的<tr>标记都添加上单独的 CSS 样式, 代码如下:

```
<tr class="altrow">
```

而奇数行保持表格本身的背景颜色不变, 完整代码如例 10.1 所示。

【例 10.1】CSS 实现隔行变色的表格 (光盘文件: 第 10 章\10-1.html)

```
<html>
<head>
<title>CSS 实现隔行变色的表格</title>
<style>
<!--
.datalist{
    border:1px solid #007108;
    font-family:Arial;
    border-collapse:collapse;
    background-color:#d9ffdc; /* 表格背景色 */
    font-size:14px;
}
.....
.datalist tr.altrow{
```

```

background-color:#a5e5aa; /* 隔行变色 */
}
-->
</style>
</head>
<body>
<table class="datalist" summary="list of members in EE Studay" id="oTable">
  <tr>
    <th scope="col">Name</th>
    <th scope="col">Class</th>
    <th scope="col">Birthday</th>
    <th scope="col">Constellation</th>
    <th scope="col">Mobile</th>
  </tr>
  <tr>
    <td>isaac</td>
    <td>W13</td>
    <td>Jun 24th</td>
    <td>Cancer</td>
    <td>1118159</td>
  </tr>
  <tr class="altrow">
    <td>fresheggs</td>
    <td>W610</td>
    <td>Nov 5th</td>
    <td>Scorpio</td>
    <td>1038818</td>
  </tr>
  ....
</table>
</body>
</html>

```

以上代码的运行结果如图 10.2 所示。可以看出纯 CSS 设置隔行变色需要将表格中所有的偶数行都手动加上单独的 CSS 类别，如果表格数据量大，则十分麻烦。另外，如果希望在某两行数据中间插入一行，则改动将更大。

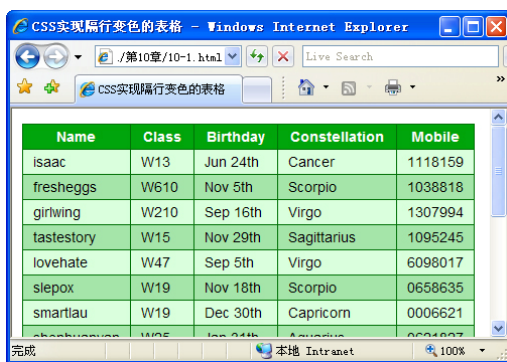


图 10.2 纯 CSS 实现隔行变色的表格

当页面中引入 JavaScript 时，上述的情况得到了很好的改观，不再需要一行行手动添加 CSS 类别。只需要用 for 循环遍历所有表格的行，当行号为偶数时则添加单独的 CSS 类别，如例 10.2 所示。

【例 10.2】JavaScript 实现隔行变色的表格（光盘文件：第 10 章\10-2.html）

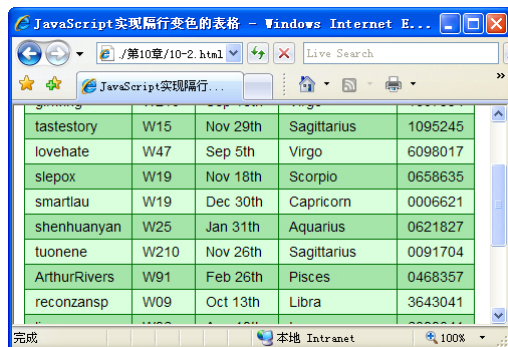
```

<script language="javascript">
window.onload = function(){

```

```
var oTable = document.getElementById("oTable");
for(var i=0;i<oTable.rows.length;i++){
    if(i%2==0) //偶数行时
        oTable.rows[i].className = "altrow";
    }
}
</script>
</head>
<body>
<table class="datalist" summary="list of members in EE Studay" id="oTable">
    <tr>
        <th scope="col">Name</th>
        <th scope="col">Class</th>
        <th scope="col">Birthday</th>
        <th scope="col">Constellation</th>
        <th scope="col">Mobile</th>
    </tr>
    <tr>
        <td>isaac</td>
        <td>W13</td>
        <td>Jun 24th</td>
        <td>Cancer</td>
        <td>1118159</td>
    </tr>
    <tr>
        <td>fresheggs</td>
        <td>W610</td>
        <td>Nov 5th</td>
        <td>Scorpio</td>
        <td>1038818</td>
    </tr>
    ..... <!-- 不再需要手动添加 CSS 类别 -->
</table>
</body>
</html>
```

以上代码不再需要手动为偶数行逐一添加 CSS 类别，所有的操作都在 JavaScript 中完成。JavaScript 代码是由开发者自己设计的，运行结果如图 10.3 所示。另外，当需要对表格添加或删除某行时，也不需要修改变色的代码。



tastestory	W15	Nov 29th	Sagittarius	1095245
lovehate	W47	Sep 5th	Virgo	6098017
slepox	W19	Nov 18th	Scorpio	0658635
smartlau	W19	Dec 30th	Capricorn	0006621
shenhuanyan	W25	Jan 31th	Aquarius	0621827
tuonene	W210	Nov 26th	Sagittarius	0091704
ArthurRivers	W91	Feb 26th	Pisces	0468357
reconzansp	W09	Oct 13th	Libra	3643041

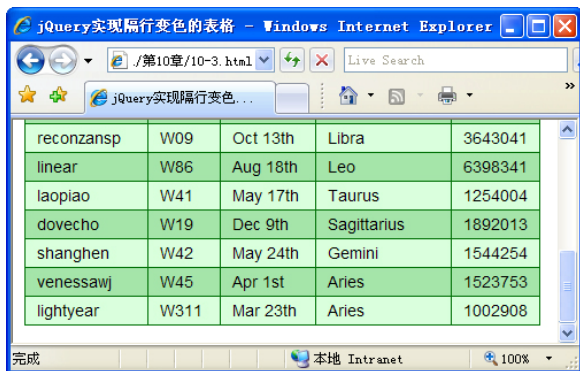
图 10.3 JavaScript 实现隔行变色的表格

当 jQuery 引入到页面中时，则不再需要开发者设计类似例 10.2 的算法，jQuery 中的选择器可以直接选中表格的奇数行或者偶数行。选中之后再添加 CSS 类别即可，如例 10.3 所示。

【例 10.3】jQuery 实现隔行变色的表格（光盘文件：第 10 章\10-3.html）

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("table.data1 tr:nth-child(odd)").addClass("altrow");
});
</script>
```

以上 JavaScript 代码首先调用 jQuery 框架，然后使用 jQuery，只需要一行代码便可以轻松实现表格的隔行变色，其语法也十分简单，后面的章节会陆续介绍，运行结果如图 10.4 所示，与例 10.1 和例 10.2 的运行结果完全相同。



reconzansp	W09	Oct 13th	Libra	3643041
linear	W86	Aug 18th	Leo	6398341
laopiao	W41	May 17th	Taurus	1254004
dovecho	W19	Dec 9th	Sagittarius	1892013
shanghen	W42	May 24th	Gemini	1544254
venessawj	W45	Apr 1st	Aries	1523753
lightyear	W311	Mar 23th	Aries	1002908

图 10.4 jQuery 实现隔行变色的表格

10.1.3 下载并使用 jQuery

jQuery 的官方网站 (<http://jquery.com/>) 提供了最新的 jQuery 框架下载，如图 10.5 所示。通常只需要下载最小的 jQuery 包（Minified）即可。

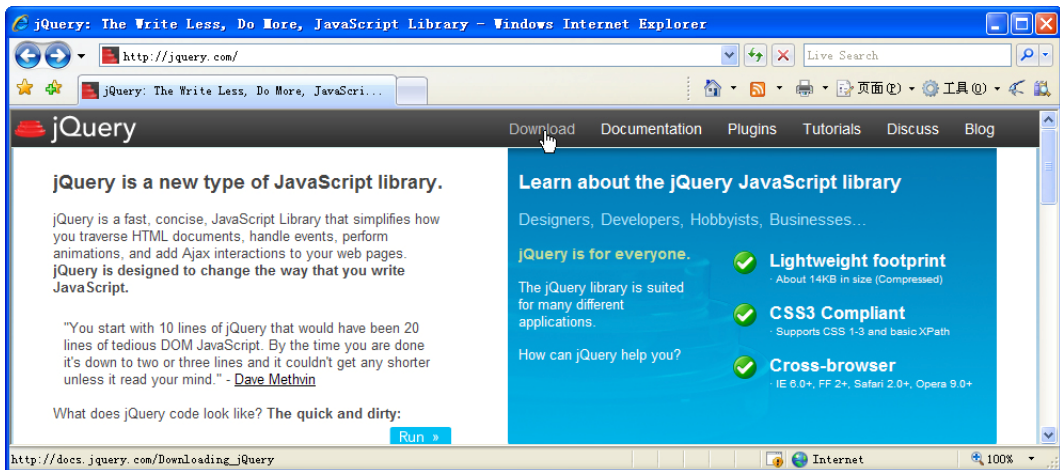
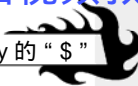


图 10.5 jQuery 官方网站

下载完成后不需要任何安装过程，直接将下载的.js 文件用<script>标记导入到自己的页面中即可，代码如下：



```
<script language="javascript" src="jquery.min.js"></script>
```

在导入 jQuery 框架后，便可以按照它的语法规则随意地使用了。

10.2 jQuery 的 “\$”

在 jQuery 中，最频繁使用的莫过于美元符号 “\$”，它提供了各种各样丰富的功能，包括选择页面中的一个或是一类元素、作为功能函数的前缀、`window.onload` 的完善、创建页面的 DOM 节点等。本节主要介绍 jQuery 中 “\$” 的使用方法，作为以后的章节的基础。

10.2.1 选择器

在 CSS 中选择器的作用是选择页面中某一类（类别选择器）元素或者某一个元素（id 选择器），而 jQuery 中的 “\$” 作为选择器，同样是选择某一类或某一个元素，只不过 jQuery 提供了更多更全面的選擇方式，并且为用户处理了浏览器的兼容问题。

例如，在 CSS 中可以通过如下代码来选择 `<h2>` 标记下包含的所有子标记 `<a>`，然后添加相应的样式风格。

```
h2 a{  
    /* 添加 CSS 属性 */  
}
```

而在 jQuery 则可以通过如下代码来选中 `<h2>` 标记下包含的所有子标记 `<a>`，作为一个对象数组，供 JavaScript 调用。

```
$("#h2 a")
```

如例 10.4 所示，文档中有两个 `<h2>` 标记，分别包含了 `<a>` 子元素。

【例 10.4】使用 “\$” 选择器（光盘文件：第 10 章\10-4.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html>  
<head>  
<title>$选择器</title>  
<script language="javascript" src="jquery.min.js"></script>  
<script language="javascript">  
    window.onload = function(){  
        var oElements = $("#h2 a");        //选择匹配元素  
        for(var i=0;i<oElements.length;i++){  
            oElements[i].innerHTML = i.toString();  
        }  
    }  
</script>  
</head>  
  
<body>  
<h2><a href="#">正文</a>内容</h2>  
<h2>正文<a href="#">内容</a></h2>  
</body>  
</html>
```

运行结果如图 10.6 所示。可以看到 jQuery 很轻松地实现了元素的选择。如果使用 DOM，

类似这样的节点选择将需要大量的 JavaScript 代码。

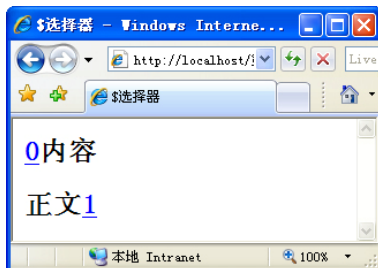


图 10.6 \$选择器

jQuery 中选择器的通用语法如下：

```
$(selector)
```

或者：

```
jQuery(selector)
```

其中 selector 符合 CSS3 标准，这在后面的章节会详细介绍，下面列出了一些典型的 jQuery 选择元素的例子：

```
$("#showDiv")
```

id 选择器，相当于 JavaScript 中的 `document.getElementById("#showDiv")`，可以看到 jQuery 的表示方法简洁很多。

```
$(".SomeClass")
```

类别选择器，选择 CSS 类别为 “SomeClass” 的所有节点元素，在 JavaScript 中要实现相同的选择，需要用 for 循环遍历整个 DOM。

```
$(“p:odd”)
```

选择所有位于奇数行的 <p> 标记。几乎所有的标记都可以使用 “:odd” 或者 “:even” 来实现奇偶的选择。

```
$("td:nth-child(1)")
```

所有表格行的第一个单元格，就是第一列。这在修改表格的某一列的属性时是非常有用的，不再需要一行行遍历表格。

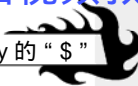
```
$("li > a")
```

子选择器，返回 标记的所有子元素 <a>，不包括孙标记。

```
$("a[href$=pdf]")
```

选择所有超链接，并且这些超链接的 href 属性是以 “pdf” 结尾的。有了属性选择器，可以很好地选择页面中的各种特性元素。

关于 jQuery 的选择器的使用还有很多技巧，在后面的章节都会陆续介绍。

**注意：**

在 jQuery 中美元符号 “\$” 其实就等同于 “jQuery”，从 jQuery 的源码中也可以看到这一点，例如：

```
if ( window.jQuery )
    var _jQuery = window.jQuery;

var jQuery = window.jQuery = function( selector, context ) {
    // The jQuery object is actually just the init constructor 'enhanced'
    return new jQuery.prototype.init( selector, context );
};

// Map over the $ in case of overwrite
if ( window.$ )
    var _$ = window.$;

// Map the jQuery namespace to the '$' one
window.$ = jQuery;
```

为了编写代码的方便，通常都采用 “\$” 来代替 “jQuery”。

10.2.2 功能函数前缀

在 JavaScript 中，开发者经常需要编写一些小函数来处理各种操作细节，例如在用户提交表单时，需要将文本框中的最前端和最末端的空格清理掉。JavaScript 没有提供类似 trim() 的功能，而引入 jQuery 后，便可以直接使用 trim() 函数，例如：

```
$.trim(sString);
```

以上代码相当于：

```
jQuery.trim(sString);
```

即 trim() 函数是 jQuery 对象的一个方法，用例 10.5 做简单的检验。

【例 10.5】 jQuery 中去除首尾空格的 \$.trim() 方法（光盘文件：第 10 章\10-5.html）

```
<head>
<title>$.trim()</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
var sString = " 1234567890 ";
sString = $.trim(sString);
alert(sString.length);
</script>
</head>
```

以上代码的运行结果如图 10.7 所示，字符串 sString 首尾的空格都被 jQuery 去掉了。



图 10.7 去除首尾空格



jQuery 中类似这样的功能函数很多，而且涉及 JavaScript 的方方面面，在后续的章节中都会陆续介绍。

10.2.3 解决 window.onload 函数的冲突

由于页面的 HTML 框架需要在页面完全加载后才能使用，因此在 DOM 编程时 window.onload 函数频繁被使用。倘若页面中有多处都需要使用该函数，或者其他.js 文件中也包含 window.onload 函数，冲突问题将十分棘手。

jQuery 中的 ready() 方法很好地解决了上述问题，它能够自动将其中的函数在页面加载完成后运行，并且同一个页面中可以使用多个 ready() 方法，而且不相互冲突。例如：

```
$(document).ready(function(){
    $("table.datalist tr:nth-child(odd)").addClass("altrow");
});
```

对上述代码 jQuery 还提供了简写，可以省略其中的 “(document).ready” 部分，代码如下：

```
$(function(){
    $("table.datalist tr:nth-child(odd)").addClass("altrow");
});
```

这就是例 10.3 中使表格隔行变色的代码。

10.2.4 创建 DOM 元素

利用 DOM 方法创建元素节点，通常需要将 document.createElement()、document.createTextNode()、appendChild() 配合使用，十分麻烦，而 jQuery 使用 “\$” 则可以直接创建 DOM 元素，例如：

```
var oNewP = $("<p>这是一个感人肺腑的故事</p>")
```

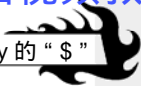
以上代码等同于 JavaScript 中的如下代码：

```
var oNewP = document.createElement("p"); //新建节点
var oText = document.createTextNode("这是一个感人肺腑的故事");
oNewP.appendChild(oText);
```

另外，jQuery 还提供了 DOM 元素的 insertAfter() 方法，因此 5.3.6 节中的例 5.15 可用 jQuery 重写，如例 10.6 所示。

【例 10.6】jQuery 中创建 DOM 元素（光盘文件：第 10 章\10-6.html）

```
<html>
<head>
<title>创建 DOM 元素</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){ //ready()函数
    var oNewP = $("<p>这是一个感人肺腑的故事</p>"); //创建 DOM 元素
    oNewP.insertAfter("#myTarget"); //insertAfter()方法
});
</script>
</head>
<body>
    <p id="myTarget">插入到这行文字之后</p>
    <p>也就是插入到这行文字之前，但这行没有 id，也可能不存在</p>
</body>
</html>
```



运行结果如图 10.8 所示。可以看到利用 jQuery 大大缩短了代码长度，节省了编写时间，为开发者提供了便利。

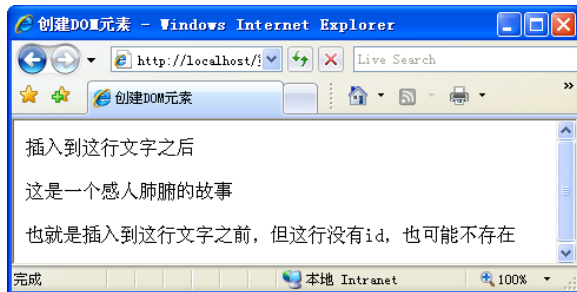


图 10.8 jQuery 创建 DOM 元素

10.2.5 自定义添加“\$”

从上面的几小节中已经看到 jQuery 的强大，但无论如何，jQuery 都不可能满足所有用户的所有需求，而且有一些特殊的需求十分小众，也不适合放到整个 jQuery 框架中。正是意识到了这一点，因此在 jQuery 中提供了用户自定义添加“\$”的方法。

例如 jQuery 中并没有将一组表单元素设置为不可用的 `disable()` 方法，用户可以自定义该方法，代码如下：

```
$.fn.disable = function(){  
    return this.each(function(){  
        if(typeof this.disabled != "undefined") this.disabled = true;  
    });  
}
```

以上代码首先设置“`$.fn.disable`”，表明为“\$”添加一个方法“`disable()`”，其中“`$.fn`”是扩展 jQuery 时所必需的。

然后利用匿名函数定义这个方法，即用 `each()` 将调用这个方法的每个元素的 `disabled` 属性均设置为 `true`（如果该属性存在）。

例 10.7 为 7.3.2 节的例 7.11 直接修改而来，添加了 `$.fn.disable` 和 `$.fn.enable` 两个 jQuery 方法。

【例 10.7】扩展 jQuery 的功能（光盘文件：第 10 章\10-7.html）

```
<script language="javascript" src="jquery.min.js"></script>  
<script language="javascript">  
$.fn.disable = function(){  
    //扩展 jQuery，表单元素统一 disable  
    return this.each(function(){  
        if(typeof this.disabled != "undefined") this.disabled = true;  
    });  
}  
$.fn.enable = function(){  
    //扩展 jQuery，表单元素统一 enable  
    return this.each(function(){  
        if(typeof this.disabled != "undefined") this.disabled = false;  
    });  
}  
</script>
```

并且在复选框旁设置了按钮，对这两个方法进行调用，代码如下：

```
function SwapInput(oName,oButton){
    if(oButton.value == "Disable"){
        //如果按钮的值为 Disable，则调用 disable()方法
        $("input[name="+oName+"]").disable();
        oButton.value = "Enable";
    }else{
        //如果按钮的值为 Enable，则调用 enable()方法
        $("input[name="+oName+"]").enable();
        oButton.value = "Disable"; //然后设置按钮的值为 Disable
    }
}

<p>你喜欢做些什么:</p>
<input type="button" name="btnSwap" id="btnSwap" value="Disable" class="btn" onclick="SwapInput('hobby',this)"><br>
<input type="checkbox" name="hobby" id="book" value="book"><label for="book">看书</label>
<input type="checkbox" name="hobby" id="net" value="net"><label for="net">上网</label>
<input type="checkbox" name="hobby" id="sleep" value="sleep"><label for="sleep">睡觉</label>
</p>
```

方法 SwapInput(oName,oButton)根据按钮的值进行判断，如果是不可用“Disable”，则调用 disable()将元素设置为不可用，同时修改按钮的值为“Enable”。反之则调用 enable()方法。运行结果如图 10.9 所示。

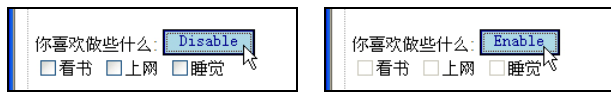


图 10.9 扩展 jQuery 的功能

10.2.6 解决“\$”的冲突

与上一节的情况类似，尽管 jQuery 已经非常强大，但有些时候开发者需要使用其他的类库框架。这时则需要很小心，因为其他框架中可能也使用了“\$”，从而发生冲突。jQuery 同样提供了 noConflict()方法来解决“\$”的冲突问题，例如：

```
jQuery.noConflict();
```

以上代码便可以使“\$”按照其他 JavaScript 框架的方式运算。这时在 jQuery 中便不能再使用“\$”，而必须使用“jQuery”，例如\$("#div p")必须写成 jQuery("#div p")。

10.3 jQuery 与 CSS3

在 3.3 节中对 CSS 选择器做了详细的介绍，但由于浏览器兼容性问题，很多 CSS3 的新特性没有介绍。当 jQuery 引入到页面中后，有必要重新探讨一下 CSS3 的选择器。

本节简单地介绍 CSS3 的相关知识，包括 CSS3 的新特性、浏览器的兼容性问题，以及 jQuery 所带来的变革等。

10.3.1 CSS3 标准

CSS 早在 1996 年的时候就作为标准被 W3C 推出，当时的版本是 CSS1。紧接着，为了适

应快速发展的 Web 页面，1998 年 W3C 推出了新的 CSS 2 版本，并陆续得到大多数浏览器的支持，其中 Microsoft 公司 2006 年发布的 Internet Explorer 7 浏览器便以 CSS2 为基础。

随着 Ajax 的流行以及 jQuery 的出现，CSS 也在不断地完善自我。CSS3 各个细节的发布则经历了很多过程，例如早在 1999 年，关于颜色的设置就发布了 CSS3 的版本，而关于 CSS 的盒子模型，最新的 CSS3 标准则是 2007 年 8 月才发布的。

**注意：**

关于 CSS3 的版本细节，有兴趣的读者可以参考 W3C 的官方网站关于 CSS3 的部分，网址是 <http://www.w3.org/Style/CSS/current-work.html#CSS3>。

CSS3 充分吸取了多年来 Web 发展的需求，提出了很多新颖的 CSS 特性，例如很受欢迎的圆角矩形 border-radius 属性，读者可以在 Firefox 或者 Safari 3 中尝试运行例 10.8。

【例 10.8】CSS3 标准中的圆角矩形（光盘文件：第 10 章\10-8.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>圆角矩形</title>
<style type="text/css">
<!--
div{
width:200px;height:120px;
border:2px solid #000000;
background-color:#FFFF00;
border-radius:20px;           /* CSS3 中的圆角矩形 */
-moz-border-radius:20px;     /* mozilla 中的圆角矩形 */
-webkit-border-radius:20px;  /* Safari 3 中的圆角矩形 */
}
-->
</style>
</head>
<body>
<div></div>
</body>
</html>
```

在 Firefox 和 IE 7 中的显示效果如图 10.10 所示。



图 10.10 圆角矩形

10.3.2 浏览器的兼容性

CSS3 的新特性中最令人兴奋的莫过于选择器的增强,例如属性选择器可以根据某个属性值来选择标记,位置选择器可以根据子元素的位置来选择标记,等等。很可惜的是,由于 IE 7 主要基于 CSS2,大部分属性选择器都支持得不够理想。

读者可以通过 <http://www.css3.info/selectors-test/test.html> 来测试自己的浏览器对各种选择器的支持情况。图 10.11 为 IE 7 浏览器的测试结果,其中红色的块非常多,表明 IE 7 对选择器支持得很不好。

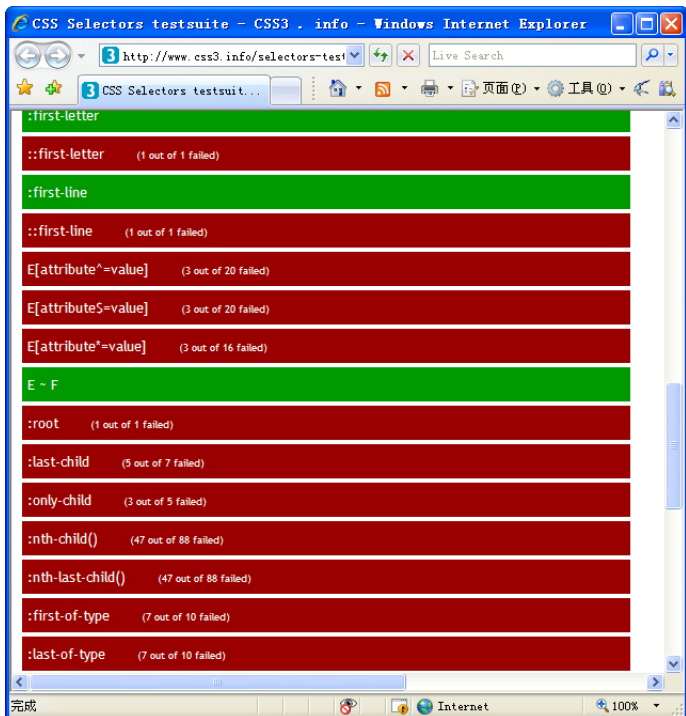


图 10.11 IE 7 的测试结果

正因为浏览器对 CSS3 的兼容性问题,使得很多方便的 CSS 属性没能推广。我们也期待着各个浏览器的新版本能提供更好的标准支持。

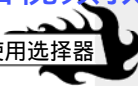
10.3.3 jQuery 的引入

jQuery 的框架中通过预先的 JavaScript 编程,提供了几乎所有 CSS3 标准下的选择器,开发者可以利用这些选择器轻松地选择各种元素,供 JavaScript 编程调用。

更重要的是, jQuery 的这些选择器的兼容性非常好,在目前主流的浏览器中都可以使用,这就使得这些理论上的 CSS3 选择器一下子变成了事实。开发者可以首先按照以前的方法定义各种 CSS 类别,然后通过 jQuery 的 `addClass()` 方法或者元素的 `className` 属性将其添加到指定元素集中,如例 10.9 所示。

【例 10.9】jQuery 带来的变革 (光盘文件:第 10 章\10-9.html)

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



```
<html>
<head>
<title>属性选择器</title>
<style type="text/css">
<!--
.myClass{
  /* 设定某个 CSS 类别 */
  background-color:#005890;
  color:#4eff00;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
  //先用 CSS3 的选择器，然后添加样式风格
  $("li:nth-child(2)").addClass("myClass");
});
</script>
</head>
<body>
<ul>
<li>糖醋排骨</li>
<li>圆笼粉蒸肉</li>
<li>泡菜鱼</li>
<li>板栗烧鸡</li>
<li>麻婆豆腐</li>
</ul>
</body>
</html>
```

以上选择器 `li:nth-child(2)` 表示选中 `` 中位于第 2 个子元素的标记，在 IE 7 和 Firefox 2.0 上的运行结果如图 10.12 所示，兼容性很棒。

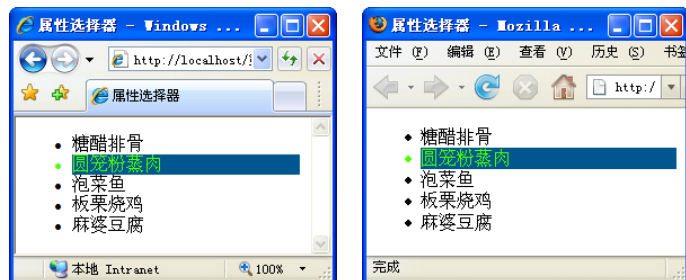


图 10.12 jQuery 的引入

10.4 使用选择器

10.3 节中已经介绍了 jQuery 带来的选择器的变化，本节重点讲解 jQuery 中丰富多彩的选择器以及它们的基本用法。

10.4.1 属性选择器

在 3.3 节中介绍的 CSS 选择器均可以用 jQuery 的 “\$” 进行选择，从例 10.4 也可以看到这

一点，这里特别需要指出的是，那些目前浏览器还没有支持或者支持的还不充分的选择器，使用 jQuery 来进行操作是很方便的。例如 CSS3 中定义属性选择器，目前的主流浏览器都不支持，本节就来介绍如何用 jQuery 实现属性选择器。

属性选择器的语法是在标记的后面用中括号 “[” 和 “]” 添加相关的属性，然后再赋予不同的逻辑关系，如例 10.10 所示。

【例 10.10】使用 jQuery 中的属性选择器（光盘文件：第 10 章\10-10.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>属性选择器</title>
<style type="text/css">
<!--
a{
    text-decoration:none;
    color:#000000;
}
.myClass{
    /* 设定某个 CSS 类别 */
    background-color:#d0baba;
    color:#5f0000;
    text-decoration:underline;
}
-->
</style>
</head>
<body>
<ul>
<li><a href="http://picasaweb.google.com/isaacshun">isaac photo</a>
<ul>
<li>10-6.html</li>
<li><a href="10-7.html">10-7.html</a></li>
<li><a href="10-8.html" title="圆角矩形">10-8.html</a></li>
<li><a href="10-9.html">10-9.html</a></li>
<li><a href="Pageisaac.html" title="制作中...">isaac</a></li>
</ul>
</li>
</ul>
</body>
</html>
```

以上代码定义了 HTML 框架结构，以及相关的 CSS 类别，供测试使用，此时的显示效果如图 10.13 所示。

在页面中如果希望选择设置了 title 属性的标记，则直接采用如下语法：

```
$("#a[title]")
```

这在 3.3.7 节中也已经提到，本例中如果使用如下代码将使得“10-8.html”和“isaac”所在的超链接增加 myClass 样式风格。

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("#a[title]").addClass("myClass");
});
</script>
```

其显示效果如图 10.14 所示，设置了 title 属性的两个超链接被添加了 myClass 样式。



图 10.13 页面框架



图 10.14 属性选择器 a[title]

如果希望根据属性的值进行判断,例如为 href 属性值等于“10-9.html”的超链接添加 myClass 样式风格,则可以使用如下代码:

```
$("#a[href=10-9.html]").addClass("myClass");
```

其运行结果如图 10.15 所示。

以上两种是比较简单的属性选择器, jQuery 中还可以根据属性值的某一部分进行匹配,例如下面的代码表示 href 属性值以“http://”开头的所有超链接。

```
$("#a[href^=http://]");
```

这对于网站中选取所有外部的超链接十分有用,可以将页面中所有外部超链接赋予相同的操作,例如:

```
$("#a[href^=http://]").addClass("myClass");
```

运行结果如图 10.16 所示。



图 10.15 属性选择器 a[href=10-9.html]



图 10.16 属性选择器 a[href^=http://]

既然可以根据属性值的开头来匹配选择,自然也可以根据属性值的结尾来进行匹配,如下代码表示 href 值以“html”结尾的超链接集合。

```
$("#a[href$=html]");
```

这种方法通常用来选取网站中某些下载资源,例如所有的 jpg 图片、所有的 pdf 文件等,例如:

```
$("#a[href$=html]").addClass("myClass");
```

其运行结果如图 10.17 所示。

另外还可以利用“*”进行任意匹配,例如下面的代码选中 href 值中包含字符串“isaac”

的所有超链接，并添加样式风格。

```
$("a[href*=isaac]").addClass("myClass");
```

其运行结果如图 10.18 所示。



图 10.17 属性选择器 a[href\$=html]

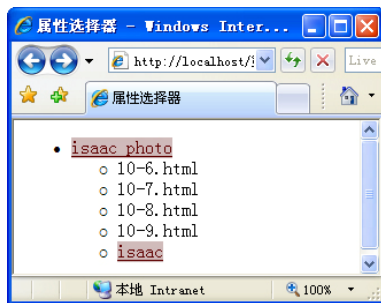


图 10.18 属性选择器 a[href*=isaac]

10.4.2 包含选择器

jQuery 中还提供了“包含选择器”，用来选择包含了某种特殊标记的元素。同样采用例 10.10 中的 HTML 框架，则下面的代码表示包含了超链接的所有标记。

```
$( "li:has(a)" )
```

下面的代码将选中二级项目列表中所有包含了超链接的标记，在 IE 7 和 Firefox 2.0 中的运行结果如图 10.19 所示。

```
$( "ul li ul li:has(a)" ).addClass("myClass");
```



图 10.19 包含选择器 "ul li ul li:has(a)"

从运行结果可以看出，IE 和 Firefox 都准确地选择了二级项目列表中的后 4 项，但是在设置下划线的样式风格上发生了分歧。这是因为在 CSS 中，a 的样式风格与 .myClass 样式风格都设置了 text-decoration，从而产生了冲突。



经验：

即便是 jQuery，也不能完全处理页面中本身存在的矛盾冲突。因此在编写 CSS、JavaScript、DOM、Ajax 等脚本时，应当分别规范各自的代码。

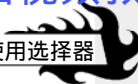


表 10.1 中罗列了 jQuery 支持的 CSS3 最基本的选择器，供读者需要时查询。

表 10.1 jQuery 支持的 CSS3 最基本的选择器

选 择 器	说 明
*	所有标记
E	所有名称为 E 的标记
EF	所有名称为 F 的标记，并且是 E 标记的子标记（包括孙、重孙等）
E > F	所有名称为 F 的标记，并且是 E 标记的子标记（不包括孙标记）
E + F	所有名称为 F 的标记，并且该标记紧接着前面的 E 标记
E ~ F	所有名称为 F 的标记，并且该标记前面有一个 E 标记
E:has(F)	所有名称为 E 的标记，并且该标记包含 F 标记
E.C	所有名称为 E 的标记，属性类别为 C，如果去掉 E，就是属性选择器.C
E#I	所有名称为 E 的标记，id 为 I，如果去掉 E，就是 id 选择器#I
E[A]	所有名称为 E 的标记，并且设置了属性 A
E[A=V]	所有名称为 E 的标记，并且属性 A 的值等于 V
E[A^=V]	所有名称为 E 的标记，并且属性 A 的值以 V 开头
E[A\$=V]	所有名称为 E 的标记，并且属性 A 的值以 V 结尾
E[A*=V]	所有名称为 E 的标记，并且属性 A 的值中包含 V

10.4.3 位置选择器

CSS3 中还允许通过标记所处的位置来进行选择，例如例 10.3 中位于奇数行的<tr>标记就是位置选择器的一种。

不光是<tr>标记，页面中几乎所有的标记都可以运用位置选择器，例如有例 10.11 所示的 HTML 框架。

【例 10.11】使用 jQuery 中的位置选择器（光盘文件：第 10 章\10-11.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>位置选择器</title>
<style type="text/css">
<!--
div{
    font-size:12px;
    border:1px solid #003a75;
    margin:5px;
}
p{
    margin:0px;
    padding:4px 10px 4px 10px;
}
.myClass{
    /* 设定某个 CSS 类别 */
    background-color:#c0ebff;
    text-decoration:underline;
}
-->
</style>
</head>
<body>
```

```
<div>
  <p>1. 大礼堂</p>
  <p>2. 清华学堂</p>
</div>
<div>
  <p>3. 图书馆</p>
</div>
<div>
  <p>4. 紫荆公寓</p>
  <p>5. C 楼</p>
  <p>6. 清华地下</p>
</div>
</body>
</html>
```

在上述代码中有 3 个<div>块，每个<div>块都包含文章段落<p>标记，其中第 1 个<div>块包含 2 个<p>，第 2 个<div>块包含 1 个<p>，最后一个包含 3 个<p>标记。在没有任何 jQuery 代码的情况下，显示结果如图 10.20 所示。

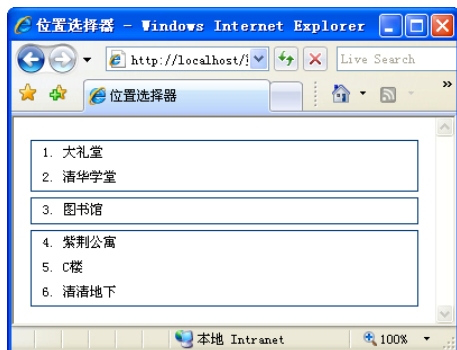


图 10.20 位置选择器

在页面中如果希望选择每个<div>块的第 1 个<p>标记，则可以通过:first-child 来选择，代码如下：

```
$("p:first-child")
```

以上代码表示选择所有的<p>标记，并且这些<p>标记是其父标记的第 1 个子标记。jQuery 会自动清除 Firefox 中的空格问题（5.3.3 节进行了详细说明），因此下面的代码在 IE 和 Firefox 中的运行结果如图 10.21 所示，兼容性非常好。

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
  $("p:first-child").addClass("myClass");
});
</script>
```

例 10.3 中所使用的隔行变色也是位置选择器的一种，在上面的例 10.11 中，同样可以通过类似的方法选中每个<div>块中的奇数行，例如：

```
$("p:nth-child(odd)").addClass("myClass");
```

以上代码的运行结果如图 10.22 所示。

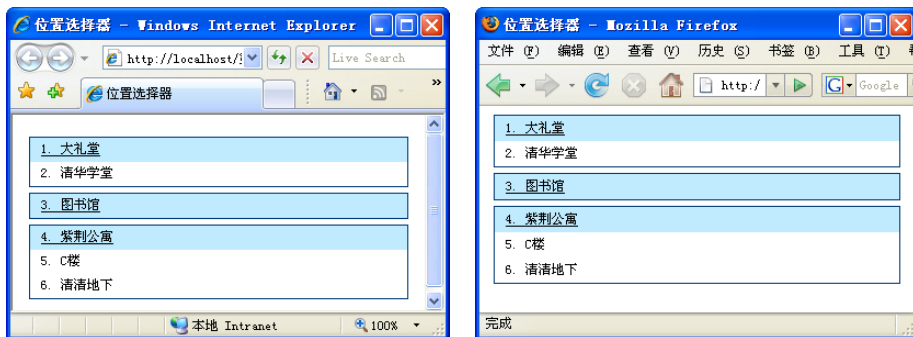


图 10.21 位置选择器 p:first-child

:nth-child(odd|even)中的奇偶顺序是根据各自的父元素单独排序的，因此上面的代码选中的是 1.大礼堂、3.图书馆、4.紫荆公寓、6.清华地下。如果希望将页面中整个<p>元素表进行排序，则可以直接使用:even 或者:odd，例如：

```
$("#p:even").addClass("myClass");
```

以上代码的显示结果如图 10.23 所示，可以从第 3 个<div>块中看到与:nth-child 的区别。

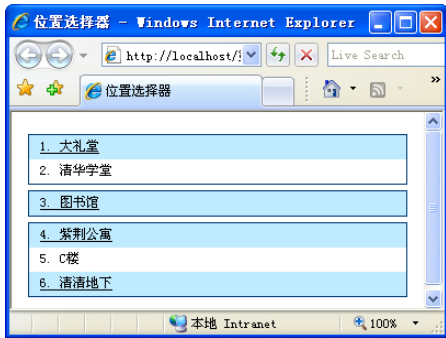


图 10.22 位置选择器 p:nth-child(odd)

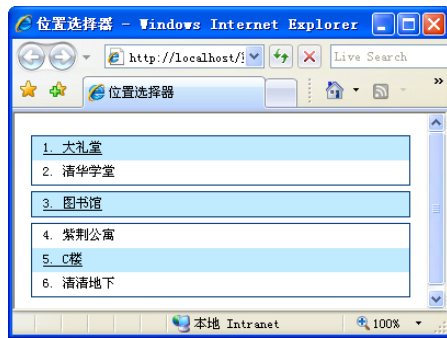


图 10.23 位置选择器 p:even

另外，可以从第 1 个<div>块中发现，:nth-child(odd)与 p:even 选择出的结果一致。这是因为:nth-child 相关的 CSS 选择器是从 1 开始计数的，而其他选择器是从 0 开始计数的。例如下面的代码匹配所有<p>列表中的第 5 个。

```
$("#p:eq(4)").addClass("myClass");
```

以上代码的运行结果如图 10.24 所示，“5.C 楼”被添加了 myClass 样式风格。

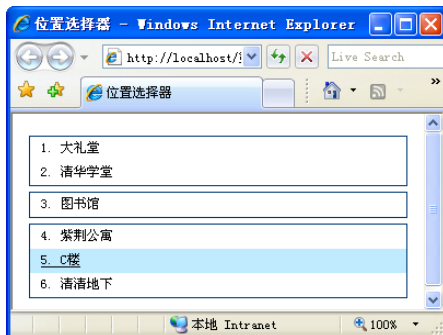


图 10.24 位置选择器 p:eq(4)



表 10.2 中罗列了所有 jQuery 支持的 CSS3 位置选择器，读者可以自己试验其中的每一项，这里不再一一重复。

表 10.2 jQuery 支持的 CSS3 位置选择器

选 择 器	说 明
:first	第 1 个元素，例如 div p:first 选中页面中所有 p 元素的第 1 个，且该 p 元素是 div 的子元素
:last	最后一个元素，例如 div p:last 选中页面中所有 p 元素的最后一个，且该 p 元素是 div 的子元素
:first-child	第一个子元素，例如 ul:first-child 选中所有 ul 元素，且该 ul 元素是其父元素的第一个子元素
:last-child	最后一个子元素，例如 ul:last-child 选中所有 ul 元素，且该 ul 元素是其父元素的最后一个子元素
:only-child	所有没有兄弟的元素，例如 p:only-child 选中所有 p 元素，如果该 p 元素是其父元素的唯一子元素
:nth-child(n)	第 n 个子元素，例如 li:nth-child(2) 选中所有 li 元素，且该 li 元素是其父元素的第 2 个子元素（从 1 开始计数）
:nth-child(odd even)	所有奇数号或者偶数号子元素，例如 li:nth-child(odd) 选中所有 li 元素，且这些 li 元素为其父元素的第奇数个元素（从 1 开始计数）
:nth-child($nX+Y$)	利用公式来计算子元素的位置，例如 li:nth-child($5n+1$) 选中所有 li 元素，且这些 li 元素为其父元素的第 $5n+1$ 个元素（1, 6, 11, 16...）
:odd 或者 :even	对于整个页面而言的奇数号或偶数号元素，例如 p:even 为页面中所有排在偶数的 p 元素（从 0 开始计数）
:eq(n)	页面中第 n 个元素，例如 p:eq(4) 为页面中的第 5 个 p 元素
:gt(n)	页面中第 n 个元素之后的所有元素（不包括第 n 个本身），例如 p:gt(0) 为页面中第 1 个 p 元素之后的所有 p 元素
:lt(n)	页面中第 n 个元素之前的所有元素（不包括第 n 个元素本身），例如 p:lt(2) 为页面中第 3 个 p 元素之前的所有 p 元素

10.4.4 过滤选择器

除了 CSS3 中的一些选择器外，jQuery 还提供了很多自定义的过滤选择器，用来处理更复杂的选择。例如很多时候希望知道用户所选中的复选框，如果通过属性的值来判断，那么只能获得初始状态下的选中情况，而不是真实的选择情况。利用 jQuery 的 :checked 选择器则可以轻松获得用户的选择，如例 10.12 所示。

【例 10.12】使用 jQuery 中的过滤选择器（光盘文件：第 10 章\10-12.html）

```
<html>
<head>
<title>jQuery 的过滤选择器</title>
<style type="text/css">
<!--
form{
    font-size:12px;
    margin:0px; padding:0px;
}
input.btn{
    border:1px solid #005079;
    color:#005079;
    font-family:Arial, Helvetica, sans-serif;
    font-size:12px;
}
.myClass{
    /* 设定某个 CSS 类别 */
    background-color:#FF0000;
    text-decoration:underline;
}
```

```
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
function ShowChecked(oCheckBox){
    //使用:checked 过滤出被用户选中的
    $("input[name="+oCheckBox+"]:checked").addClass("myClass");
}
</script>
</head>
<body>
<form name="myForm">
<input type="checkbox" name="sports" id="football"><label for="football">足球</label><br>
<input type="checkbox" name="sports" id="basketball"><label for="basketball">篮球</label><br>
<input type="checkbox" name="sports" id="volleyball"><label for="volleyball">排球</label><br>
<br><input type="button" value="Show Checked" onclick="ShowChecked('sports')" class="btn">
</form>
</body>
</html>
```

以上代码中有 3 个复选框，通过 jQuery 的过滤选择器:check 便可以很容易地筛选出用户选择的项，并赋予特殊的 CSS 样式，运行结果如图 10.25 所示。

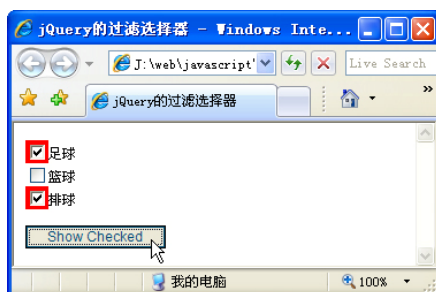


图 10.25 jQuery 的过滤选择器

另外，过滤选择器之间可以迭代使用，例如：

```
:checkbox:checked:enabled
```

表示<input type="checkbox">中所有被用户选中而且没有被禁用的。表 10.3 罗列了 jQuery 中常用的过滤选择器。

表 10.3 jQuery 中常用的过滤选择器

选 择 器	说 明
:animated	所有处于动画中的元素
:button	所有按钮，包括 input[type=button]、input[type=submit]、input[type=reset]和<button>标记
:checkbox	所有复选框，等同于 input[type=checkbox]
:contains(foo)	选择所有包含了文本“foo”的元素
:disabled	页面中被禁用了的元素
:enabled	页面中没有被禁用的元素
:file	上传文件的元素，等同于 input[type=file]
:header	选中所有标题元素，例如<h1>~<h6>
:hidden	页面中被隐藏了的元素
:image	图片提交按钮，等同于 input[type=image]



续表

选 择 器	说 明
:input	表单元素, 包括 <input>、<select>、<textarea>、<button>< td=""></input>、<select>、<textarea>、<button><>
:not(filter)	反向选择
:parent	选择所有拥有子元素(包括文本)的元素, 空元素将被排除
:password	密码文本框, 等同于 input[type=password]
:radio	单选按钮, 等同于 input[type=radio]
:reset	重置按钮, 包括 input[type=reset]和 button[type=reset]
:selected	下拉菜单中被选中的项
:submit	提交按钮, 包括 input[type=submit]和 button[type=submit]
:text	文本输入框, 等同于 input[type=text]
:visible	页面中的所有可见元素

10.4.5 实现反向过滤

在上述过滤选择器中:not(filter)过滤器可以进行反向的选择, 其中 filter 参数可以是任意其他的过滤选择器, 例如下面的代码表示

```
input:not(:radio)
```

另外, 过滤选择器还可以迭代使用, 如例 10.13 所示。

【例 10.13】使用 jQuery 的反向过滤功能(光盘文件:第 10 章\10-13.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>反向过滤</title>
<style type="text/css">
<!--
form{
    font-size:12px;
    margin:0px; padding:0px;
}
p{
    padding:5px; margin:0px;
}
.myClass{
    background-color:#ffbf4;
    text-decoration:underline;
    border:1px solid #0000FF;
    font-family:Arial, Helvetica, sans-serif;
    font-size:12px;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    //迭代使用选择器
    $("input:not(:checkbox):not(:radio)").addClass("myClass");
});
```

```

</script>
</head>
<body>
<form method="post" name="myForm1" action="addInfo.aspx">
<p><label for="name">姓名:</label> <input type="text" name="name" id="name"></p>
<p><label for="passwd">密码:</label> <input type="password" name="passwd" id="passwd"></p>
<p><label for="color">最喜欢的颜色:</label>
<select name="color" id="color">
  <option value="red">红</option>
  <option value="green">绿</option>
  <option value="blue">蓝</option>
  <option value="yellow">黄</option>
  <option value="cyan">青</option>
  <option value="purple">紫</option>
</select></p>
<p>性别:
  <input type="radio" name="sex" id="male" value="male"><label for="male">男</label>
  <input type="radio" name="sex" id="female" value="female"><label for="female">女</label></p>
<p>你喜欢做些什么:<br>
  <input type="checkbox" name="hobby" id="book" value="book"><label for="book">看书</label>
  <input type="checkbox" name="hobby" id="net" value="net"><label for="net">上网</label>
  <input type="checkbox" name="hobby" id="sleep" value="sleep"><label for="sleep">睡觉</label></p>
<p><label for="comments">我要留言:</label><br><textarea name="comments" id="comments" cols="30"
rows="4"></textarea></p>
<p><input type="submit" name="btnSubmit" id="btnSubmit" value="Submit">
<input type="reset" name="btnReset" id="btnReset" value="Reset"></p>
</form>
</body>
</html>

```

以上代码中的选择器如下:

```
$(":input:not(:checkbox):not(:radio)").addClass("myClass");
```

表示所有<input>、<select>、<textarea>或<button>中非 checkbox 和非 radio 的元素,这里需要注意“input”与“:input”的区别。运行结果如图 10.26 所示,除了单选按钮和复选框,表单的其余元素均运用了 myClass 样式风格。

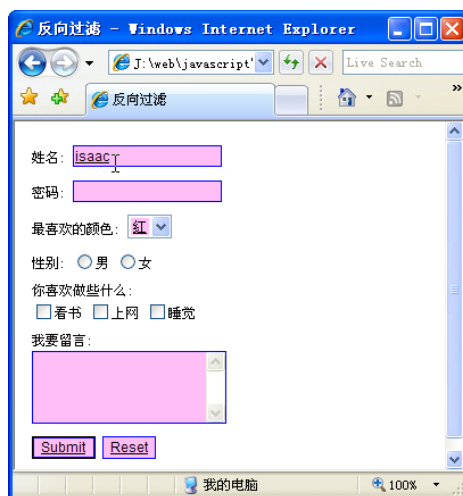


图 10.26 反向过滤

**注意：**

在:~not(filter)中, filter 参数必须是过滤选择器,而不能是其他的选择器,下面的代码是典型的错误语法：

```
div:~not(p:~hidden)
```

正确的写法为：

```
div p:~not(:~hidden)
```

10.5 管理选择结果

用 jQuery 选择出来的元素与数组非常类似,可以通过 jQuery 提供的一系列方法对其进行处理,包括获取长度、查找某个元素、截取某个段落等。

10.5.1 获取元素的个数

在 jQuery 中可以通过 size()方法获取选择器中元素的个数,它类似于数组中的 length 属性,返回整数,例如：

```
$("#img").size()
```

获得页面中所有图片的数目。例 10.14 是一个稍微复杂一点的示例,用来添加并计算页面中的<div>块。

【例 10.14】获取 jQuery 选择器中元素的个数（光盘文件：第 10 章\10-14.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
html1-transitional.dtd">
<html>
<head>
<title>size()方法</title>
<style type="text/css">
<!--
html{
  cursor:help; font-size:12px;
  font-family:Arial, Helvetica, sans-serif;
}
div{
  border:1px solid #003a75;
  background-color:#FFFF00;
  margin:5px; padding:20px;
  text-align:center;
  height:20px; width:20px;
  float:left;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
document.onclick = function(){
  var i = $("#div").size()+1; //获取 div 块的数目（此时还没有添加 div 块）
  $(document.body).append($("#div">"+i+"</div>")); //添加一个 div 块
  $("#span").html(i); //修改显示的总数
}
```

```

</script>
</head>
<body>
  页面中一共有<span>0</span>个 div 块。单击鼠标添加 div:
</body>
</html>

```

以上代码首先通过 `document.onclick` 为页面添加单击的响应函数。然后通过 `size()` 获取页面中 `<div>` 块的数目，并且使用 `append()` 为页面添加 1 个 `<div>` 块，然后利用 `html()` 方法将总数显示在 `` 标记中。运行结果如图 10.27 所示，随着鼠标的单击，`<div>` 块在不断地增加。



图 10.27 size()获取元素个数

10.5.2 提取元素

在 jQuery 的选择器中，如果想提取某个元素，最直接的方法是采用方括号加序号的形式，例如：

```
$("#img[title]")[1]
```

获取所有设置了 `title` 属性的 `img` 标记中的第 2 个元素。jQuery 同样提供了 `get(index)` 方法来提取元素，以下代码与上面的完全等效。

```
$("#img[title]").get(1)
```

另外，`get()` 方法在不设置任何参数时，可以将元素转化为一个元素对象的数组，如例 10.15 所示。

【例 10.15】jQuery 提取选择器中的元素（光盘文件：第 10 章\10-15.html）

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>get()方法</title>
<style type="text/css">
<!--
div{
  border:1px solid #003a75;
  color:#CC0066;
  margin:5px; padding:5px;
  font-size:12px;
  font-family:Arial, Helvetica, sans-serif;
  text-align:center;
  height:20px; width:20px;
  float:left;
}
-->
</style>

```



```

<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
function disp(divs){
    for(var i=0;i<divs.length;i++){
        $(document.body).append($("#div
style='background:'"+divs[i].style.background+"';>" +divs[i].innerHTML+"</div>"));
    }
    $(function(){
        var aDiv = $("div").get();    //转化为 div 对象数组
        disp(aDiv.reverse());        //反序，传给处理函数
    });
</script>
</head>
<body>
<div style="background:#FFFFFF">1</div>
<div style="background:#CCCCCC">2</div>
<div style="background:#999999">3</div>
<div style="background:#666666">4</div>
<div style="background:#333333">5</div>
<div style="background:#000000">6</div>
</body>
</html>

```

以上代码将页面中本身的 6 个 div 块用 get()方法转换为数组，然后用数组的反序 reverse()方法（数组的 reverse()方法在 2.3.5 节的例 2.19 中有详细介绍），并传给 disp()函数，再将其一个个显示在页面中。运行结果如图 10.28 所示。

get(index)方法可以获取指定位置的元素，反过来 index(element)方法可以查找元素 element 所处的位置，例如：

```
var iNum = $("li").index($("#li[title=isaac]")[0])
```

以上代码将获取<li title="isaac">标记在整个标记列表中所处的位置，并将该位置返回给整数 iNum。例 10.16 为 index(element)方法的典型运用。

【例 10.16】用 index()方法获取元素的序号（光盘文件：第 10 章\10-16.html）

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>index(element)方法</title>
<style type="text/css">
<!--
body{
    font-size:12px;
    font-family:Arial, Helvetica, sans-serif;
}
div{
    border:1px solid #003a75;
    background:#fcff9f;
    margin:5px; padding:5px;
    text-align:center;
    height:20px; width:20px;
    float:left;
    cursor:help;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>

```

```

<script language="javascript">
$(function(){
    //click()添加单击事件
    $("div").click(function(){
        //将块本身用 this 关键字传入，从而获取自身的序号
        var index = $("div").index(this);
        $("span").html(index.toString());
    });
});
</script>
</head>
<body>
<div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
单击的 div 块序号为: <span></span>
</body>
</html>

```

以上代码将块本身用 `this` 关键字传入 `index()` 方法中，获取自身的序号。并且利用 `click()` 添加事件，将序号显示出来，运行结果如图 10.29 所示。



图 10.28 get()方法



图 10.29 index(element)方法

10.5.3 添加、删除、过滤元素

除了获取选择元素的相关信息外，jQuery 还提供了一系列方法来修改这些元素的集合，例如可以利用 `add()` 方法来添加元素，代码如下：

```
$( "img[alt]").add("img[title]")
```

以上代码将所有设置了 `alt` 属性的 `` 和所有设置了 `title` 属性的 `` 组合在了一起，供别的方法统一调用，它完全等同于：

```
$( "img[alt],img[title]")
```

例如，可以将组合后的元素集统一设置添加 CSS 属性，代码如下：

```
$( "img[alt]").add("img[title]").addClass("myClass");
```

与 `add()` 方法相反，`not()` 方法可以去除元素集中的某些元素，例如：

```
$( "li[title]").not("[title*=isaac]")
```

表示选中所有设置了 `title` 属性的 `` 标记，但不包括 `title` 值中任意匹配字符串 “isaac” 的那些。例 10.17 为 `not()` 方法的一种典型运用。

【例 10.17】使用 not() 方法（光盘文件：第 10 章\10-17.html）

```

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

```

```

<head>
<title>not()方法</title>
<style type="text/css">
<!--
div{
    background:#fcff9f;
    margin:5px; padding:5px;
    height:40px; width:40px;
    float:left;
}
.green{ background:#66FF66; }
.gray{ background:#CCCCCC; }
#blueone{ background:#5555FF; }
.myClass{
    border:2px solid #000000;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("div").not(" .green, #blueone").addClass("myClass");
});
</script>
</head>
<body>
    <div></div>
    <div id="blueone"></div>
    <div></div>
    <div class="green"></div>
    <div class="green"></div>
    <div class="gray"></div>
    <div></div>
</body>
</html>

```

以上代码中共有 7 个<div>块，其中 3 个没有设置任何类型或者 id，一个设置了 id 为“blueone”，两个设置了样式风格“green”，另外一个设置了样式风格“gray”。jQuery 代码首先选中所有的<div>块，然后通过 not()方法去掉样式风格为“green”和 id 为“blueone”的<div>块，给剩下的添加 CSS 样式 myClass，运行结果如图 10.30 所示。

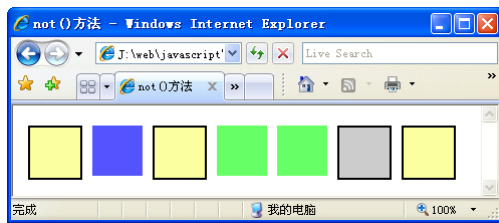


图 10.30 not()方法

注意：

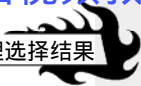
not()方法所接受的参数都不能包含特定的元素，只能是通用的表达式，例如下面是典型的错误代码：

```
$("li[title]").not("img[title*=isaac]")
```

正确的写法为：

```
$("li[title]").not("[title*=isaac]")
```





除了 add() 和 not() 外, jQuery 还提供了更强大的 filter() 方法来筛选元素。filter() 可以接受两种类型的参数, 一种与 not() 方法一样, 接受通用的表达式, 代码如下:

```
$("#li").filter("[title*=isaac]")
```

以上代码表示在标记的列表中筛选出那些属性 title 值任意匹配字符串“isaac”的标记。这看上去与下面的代码相同。

```
$("#li[title*=isaac]")
```

但 filter() 主要用于 jQuery 语句的链接, 如例 10.18 所示。

【例 10.18】filter() 方法的基础运用 (光盘文件: 第 10 章\10-18.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>filter()方法</title>
<style type="text/css">
<!--
div{
    margin:5px; padding:5px;
    height:40px; width:40px;
    float:left;
}
.myClass1{
    background:#fcff9f;
}
.myClass2{
    border:2px solid #000000;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("#div").addClass("myClass1").filter("[class*=middle]").addClass("myClass2");
});
</script>
</head>
<body>
<div></div>
<div class="middle"></div>
<div class="middle"></div>
<div class="middle"></div>
<div class="middle"></div>
<div></div>
</body>
</html>
```

以上代码中有 6 个<div>块, 其中中间 4 个设置了 class 属性为 middle。在 jQuery 代码中首先给所有的<div>块都添加“myClass1”样式风格, 然后通过 filter() 方法, 将 class 属性匹配“middle”的选择出来, 再添加“myClass2”样式风格。

其运行结果如图 10.31 所示, 可以看到所有的<div>块都运用了 myClass1 的背景颜色, 而只有被筛选出来的中间<div>块运用了 myClass2 的边框。

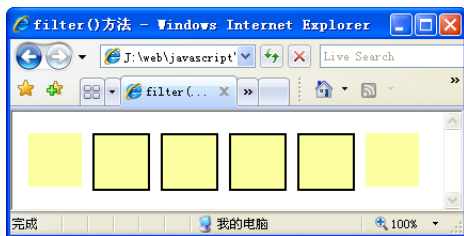


图 10.31 filter()方法

**注意：**

在 filter()的参数中,不能使用直接的等于匹配(=),只能使用前匹配(^=)、后匹配(&=)或者任意匹配(*=),例如例 10.18 中的 filter(),如果写成如下的语句将得不到想要的过滤效果:

```
filter("[class=middle]")
```

filter()另外一种类型的参数是函数。函数参数的功能非常强大,它可以让用户自定义筛选函数。该函数要求返回布尔值,对于返回值为 true 的元素则保留,否则去除。例 10.19 展示了该方法的使用。

【例 10.19】filter()方法接受函数作为参数(光盘文件:第 10 章\10-19.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>filter()方法</title>
<style type="text/css">
<!--
div{
margin:5px; padding:5px;
height:40px; width:40px;
float:left;
}
.myClass1{
background:#fcff9f;
}
.myClass2{
border:2px solid #000000;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("div").addClass("myClass1").filter(function(index){
        return index == 1 || $(this).attr("id") == "fourth";
    }).addClass("myClass2");
});
</script>
</head>
<body>
<div id="first"></div>
<div id="second"></div>
<div id="third"></div>
<div id="fourth"></div>
```

```
<div id="fifth"></div>
</body>
</html>
```

以上代码首先将所有的<div>块赋予 myClass1 样式风格，然后利用 filter()返回的函数值将<div>列表中位于第 1 个的，以及 id 为“fourth”的元素筛选出来，并赋予 myClass2 样式风格，运行结果如图 10.32 所示。

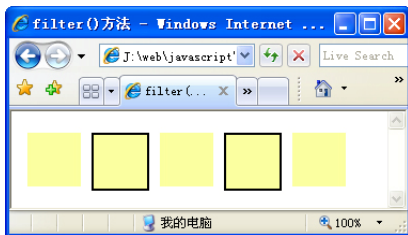


图 10.32 filter()方法

10.5.4 查询过滤新元素集合

jQuery 还提供了一些很实用的小方法，通过查询来获取新的元素集合。例如 find()方法，通过匹配选择器来筛选元素，代码如下：

```
$("p").find("span")
```

以上代码表示在所有<p>标记的元素中搜索标记，获得一个新的元素集合，它完全等同于以下代码：

```
$("span",$("p"))
```

实际运用如例 10.20 所示。

【例 10.20】用 find()方法查找元素（光盘文件：第 10 章\10-20.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>find()方法</title>
<style type="text/css">
<!--
.myClass{
    background:#ffde00;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("p").find("span").addClass("myClass");
});
</script>
</head>
<body>
<p><span>Hello</span>, how are you?</p>
```

```
<p>Me? I'm <span>good</span>.</p>
<span>What about you?</span>
</body>
</html>
```

例 10.20 的运行结果如图 10.33 所示,可以看到位于<p>标记中的被运用了新的样式风格,而最后一行没有任何变化。

另外,还可以通过 is()方法来检测是否包含指定的元素,例如可以通过以下代码来检测页面的<div>块中是否包含图片。

```
var bHasImage = $("div").is("img");
```

is()方法返回布尔值,当至少包含一个匹配项时为 true,否则为 false。

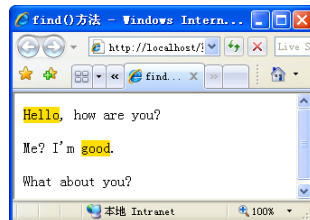


图 10.33 find()方法

10.6 采用 jQuery 链

从前面的例中也可以反复看到, jQuery 的语句可以链接在一起。这不仅可以缩短代码的长度,而且很多时候可以实现特殊的效果。如例 10.19 中的如下代码:

```
$("#div").addClass("myClass1").filter(function(index){
    return index == 1 || $(this).attr("id") == "fourth";
}).addClass("myClass2");
```

以上代码为整个<div>列表增加样式风格“myClass1”,然后进行筛选,再为筛选出的元素单独增加样式风格“myClass2”。如果不采用 jQuery 链,实现上述效果将非常麻烦。

在 jQuery 链中,后面的操作都是以前面的操作结果为对象的。如果希望操作对象为上一步的对象,则可以使用 end()方法,如例 10.21 所示。

【例 10.21】用 end()方法控制 jQuery 链 (光盘文件:第 10 章\10-21.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>end()方法</title>
<style type="text/css">
<!--
.myClass1{
    background:#ffde00;
}
.myClass2{
    border:1px solid #0000FF;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("p").find("span").addClass("myClass1").end().addClass("myClass2");
});
</script>
</head>
<body>
```

```
<p>Hello, <span>how</span> are you?</p>
<span>very nice,</span> thank you.
</body>
</html>
```

以上代码在<p>标记中搜索标记，然后添加样式风格“myClass1”，利用 end()方法将操作对象往回设置为\$("p")，并添加样式风格“myClass2”，运行结果如图 10.34 所示。

另外，还可以通过 andSelf()将前面两个对象进行组合后共同处理，如例 10.22 所示。

【例 10.22】用 andSelf()方法控制 jQuery 链 (光盘文件：第 10 章\10-22.html)

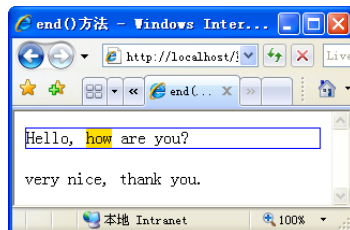


图 10.34 end()方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html>
<head>
<title>andSelf()方法</title>
<style type="text/css">
<!--
.myBackground{
    background:#ffde00;
}
.myBorder{
    border:2px solid #0000FF;
}
p{
    margin:8px; padding:4px;
    font-size:12px;
}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("div").find("p").addClass("myBackground").andSelf().addClass("myBorder");
});
</script>
</head>
<body>
<div>
    <p>第一段</p>
    <p>第二段</p>
</div>
</body>
</html>
```

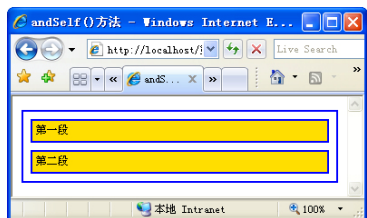


图 10.35 andSelf()方法

以上 jQuery 代码首先在<div>块中搜索<p>标记，添加背景相关的样式风格“myBackground”，这个风格只对<p>标记有效。然后利用 andSelf()方法将<div>和<p>组合在一起，添加边框相关的样式风格“myBorder”，这个风格对<div>和<p>均有效。运行结果如图 10.35 所示。