

第 1 章

◀ jQuery入门 ▶

当前的网页开发,几乎所有的项目都依赖于 jQuery 框架,它是一个开源的 JavaScript 库。jQuery 的创始人是美国的 John Resig, 它于 2006 年 1 月创建了 jQuery 项目。jQuery 库的目的是使网站开发人员用较少的代码完成更多的功能(即 Write less, do more)。它具有极其简洁的语法,并且克服了不同浏览器平台之间的兼容性,极大地提高了程序员编写网站代码的效率。随着人们对 jQuery 的了解以及其开源特性,越来越多的人开始用 jQuery 创建项目,并且对 jQuery 进行完善和优化。

本章主要内容:

- 认识 jQuery 与 JavaScript 的关系
- 学习利用浏览器的开发工具调试 jQuery
- 了解 jQuery 库的核心方法 \$()
- 学习创建一个带 jQuery 库的网页

1.1 什么是 jQuery

JavaScript 发展了这么多年,却因为很多浏览器有自己的标准而让人使用起来非常头疼。随着技术进步, jQuery 横空出世了,它到底有什么优势,又为什么在当前会这么流行呢?本节来揭开 jQuery 流行的真相。

1.1.1 下载并配置 jQuery 运行环境

为了使用 jQuery,首先必须从 jQuery 官网下载最新的 jQuery 库, jQuery 的官方网站网址如下:

```
http://jquery.com
```

进入官网后,位于右上角的位置可以看到“Download jQuery”按钮,如图 1.1 所示。单击这个下载按钮后,官方提供了 3 个下载文件,如图 1.2 所示。

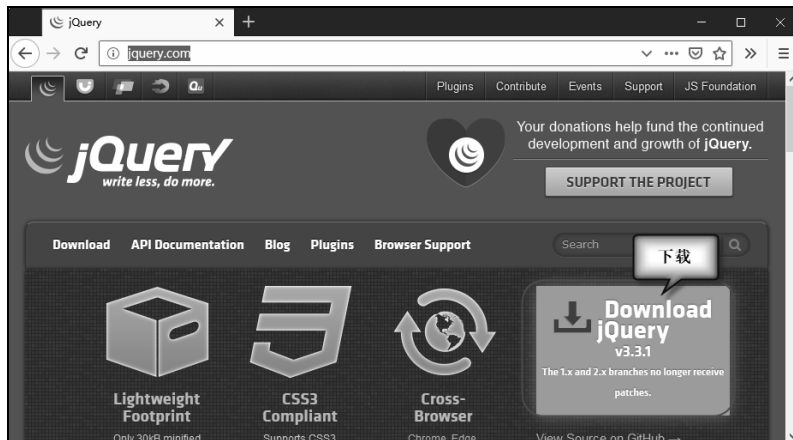


图 1.1 下载 jQuery 库

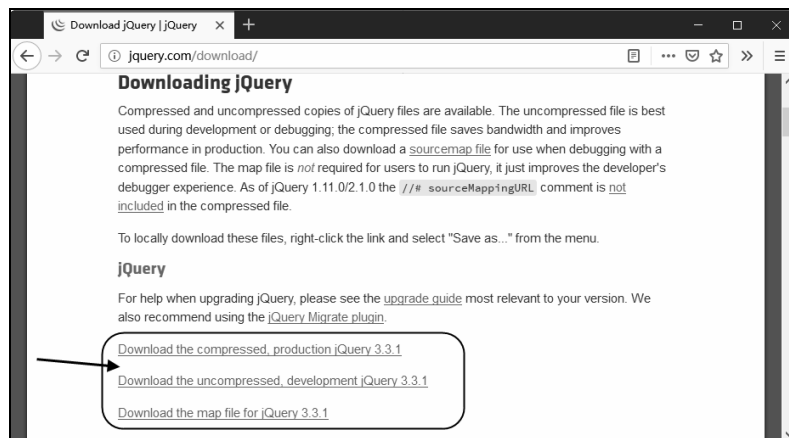


图 1.2 jQuery 不同的版本下载页面

有 3 个可供下载的文件，分别是：

- **Production jQuery 版：**优化压缩后的版本，具有较小的体积，主要用于部署网站时使用。
- **Development jQuery 版：**未压缩版本，有 266KB 的大小，一般用于在网站建设时使用这个版本以便调试。
- **jQuery map 文件：**map 文件能够被用来在源代码感知的浏览器上调试压缩后的 jQuery 文件，比如 Google Chrome，它可以增强调试的体验，对于使用 jQuery 的用户来说，一般不需要下载该文件。

建议同时下载这 3 个文件，放在一个统一的位置，这样可以在需要进行切换，将鼠标悬停在要下载的连接上，右击鼠标从弹出的菜单中选择“从链接另存文件为”（Firefox），即可将选中的 jQuery 文件保存起来，保存文件名自动为 jquery-3.3.1.js。



如果是 Chrome 浏览，右击菜单后选择“链接另存为”，保存后的文件名都是一致的。

从下载的 jQuery 库名字可以看出，其扩展名为.js，与自行编写的其他 js 文件一样，jQuery

库实际上就是一个扩展 JavaScript 功能的外部 js 文件。因此，引用 jQuery 库的方式与引用其他的外部 js 文件相似，在网页上引用 jQuery 库的代码如下所示：

```
<!--引用 jQuery 脚本库-->
<script src="jQuery/jquery-3.3.1.js" ></script>
```

在网站开发阶段，可以直接引用开发版，即 jquery-3.3.1.js 版本，当网站要部署到正式环境时，可以引用压缩后的 jquery-3.3.1.min.js 版本，这个压缩版本只有 84.8KB 大小，可以减少网页代码大小，并提高页面加载速度。

1.1.2 jQuery1.x、2.x 和 3.x 的区别

虽然目前官方主页已经只支持 3.x 的下载，但是因为一些旧代码的维护或公司的要求，很多读者可能依然使用的是 3.x 以下的版本。这里我们就简单说明一下三者的区别：

- 1.x：兼容 IE6、7、8（原来是国内首选），是使用最为广泛的，目前官方只做 BUG 维护，功能不再新增。因此一般项目来说，使用 1.x 版本就可以了，最终版本为 1.12.4（2016 年 5 月 20 日截止）。
- 2.x：不兼容 IE6、7、8，很少有人使用，目前官方只做 BUG 维护，功能不再新增。如果不考虑兼容低版本的浏览器可以使用 2.x，毕竟很多网站已经公开说不再支持 IE6，最终版本为 2.2.4（2016 年 5 月 20 日截止）。
- 3.x：不兼容 IE6、7、8，只支持最新的浏览器，目前该版本是官方主要更新维护的版本。最新版本为 3.3.1（2018 年 1 月 20 日更新）。

具体的使用上差别不是特别大，读者可通过 <https://api.jquery.com/> 官方文档来了解。不同版本所支持的浏览器也可以通过 <https://jquery.com/browser-support/> 来了解。

1.1.3 jQuery 与 JavaScript 的区别

由于 JavaScript 属于一门动态编程语言，因此在学习与使用时极易引起错误，并且目前也没有特别好的代码检查工具，而且编码时最重要的是要兼顾各种不同浏览器之间的代码兼容性，比如同样的代码在 IE 中可以运行，在 Firefox 中却无法显示，这常常令程序员们抱怨不已。jQuery 的出现恰恰解决了这些问题。

为了了解 jQuery 代码的简洁易用性，我们编写一个网页，对比用 JavaScript 和用 jQuery 实现同样的功能需要几行代码。新建一个名为 JavaScript01.html 的网页，实现表单颜色的更改，页面的效果如图 1.3 所示。

这个页面包含了一个 HTML 的表单，在表单外面有两个按钮，用来更改表单中的 input 元素和 textarea 元素的背景色，HTML 的定义如下：



图 1.3 JavaScript 代码和 jQuery 库代码的示例页面

```

01 <body>
02 使用 JavaScript 代码更改 DOM 元素
03 <!-- 表单元素 -->
04 <form action="" id="contacts-form">
05   <fieldset>
06     <label><span>姓名:</span><input type="text" /></label></br>
07     <label><span>电子邮件:</span><input type="text" /></label></br>
08     <div class="wrapper"><span>留言:</span><textarea></textarea></div>
09   </fieldset>
10 </form>
11 <!-- 操作按钮 -->
12 <div class="wrapper">
13 <a href="#" class="button" onClick="javascript:setColorByJs();" >
JavaScript 更改表单颜色</a>
14 <a href="#" class="button" onClick="javascript:setColorByjQuery();" >
jQuery 更改表单颜色
15 </a>
16 </div>
17 </body>

```

HTML 页面上放置了一个表单标签 `form`，在 `form` 内部有两个 `input` 元素和一个 `textarea` 元素，在 `form` 元素的外面放置了两个按钮，分别为这两个按钮定义了 `onClick` 事件，“JavaScript 更改表单颜色”按钮将调用 `setColorByJs` 函数，而“jQuery 更改表单颜色”将调用 `setColorByjQuery` 函数，这两个函数在 HTML 的 `head` 部分实现，如下所示。

```

01 <head>
02 <meta charset=utf-8">
03 <!-- 添加对 jQuery 库的引用 -->
04 <script src="../jquery-3.3.1.js"></script>
05 <title>JavaScript 示例1</title>
06 <script type="text/javascript">
07   //使用 javascript 更改表单背景色
08   function setColorByJs() {
09     //获取 input 元素集合
10     var inputs=document.getElementsByTagName("input");
11     //循环元素集合，为每一个元素设置背景色
12     for(var i=0;i<inputs.length;i++){
13       inputs[i].style.background="#efefef";
14     }
15     //获取 textarea 元素集合
16     var textareas=document.getElementsByTagName("textarea");
17     //循环元素集合，为每一个元素设置背景色
18     for(var i=0;i<textareas.length;i++){
19       textareas[i].style.background="#efefef";

```

```

20     }
21 }
22 //使用 jQuery 更改表单背景色
23 function setColorByjQuery() {
24     $("input").css("background","#efefef"); //更改 input 元素的背景色
25     $("textarea").css("background","#efefef");
26                                     //更改 textarea 元素的背景色
27 }
28 </script>
29 </head>

```

通过比较 JavaScript 代码和 jQuery 的代码(jQuery 的语法后面会解释),会发现使用 jQuery 只需要极其精简的代码(第 23~26 行)来完成。用 JavaScript 需要数行代码(第 8~21 行)完成的工作,JavaScript 代码使用了 `getElementsByTagName` 函数,返回了一个数组,然后通过循环这个数组从而得到每个元素,在得到了元素之后,为其 `style` 属性指定背景色。而 jQuery 通过其表单选择器,可以用非常简单的语句来实现 `getElementsByTagName` 实现的类似功能,其 `css` 方法可以针对一个选中的集合进行操作,这大大简化了需要循环执行的操作。



第 4 行代码 `src="../jquery-3.3.1.js"` 中的 `../` 表示是当前目录的上一级目录,因为 `jquery-3.3.1.js` 没有在当前目录。

jQuery 使用了 CSS 的选择器,并且具有隐式迭代功能,这就简化了原本需要循环执行的相关代码。从功能性上来说, jQuery 提供了如下特色来完成对网页的操作:

- 快速获取文档元素: jQuery 的选择机制构建于 CSS 的选择器,它提供了快速查询 DOM 文档中元素的能力,而且大大强化了 JavaScript 中获取页面元素的方式。
- 提供漂亮的页面动态效果: jQuery 中内置了一系列的动画效果,可以开发出非常漂亮的网页,目前许多知名的网站都使用了 jQuery 内置的效果,比如淡入淡出、元素移除等动态特效。
- 创建 AJAX 无刷新网页: AJAX 是异步的 JavaScript 和 XML 的简称,可以开发出非常灵敏无刷新的网页,特别是开发服务器端网页时,比如 PHP 网站,需要往返地与服务器沟通,如果不使用 AJAX,每次数据更新不得不重新刷新网页,而使用了 AJAX 特效后,可以对页面进行局部刷新,提供更好的页面交互效果。
- 提供对 JavaScript 语言的增强: jQuery 提供了对基本 JavaScript 结构的增强,比如元素迭代和数组处理等操作。
- 增强的事件处理: jQuery 提供了各种页面事件,它可以避免程序员在 HTML 中添加太多事件处理代码,最重要的是,它的事件处理器消除了各种浏览器兼容性问题。
- 更改网页内容: jQuery 可以修改网页中的内容,比如更改网页的文本、插入或者是翻转网页图像,简化了原本需要编写大量 JavaScript 代码的工作。

jQuery 之所以如此优秀,是因为它整合了非常多优秀的特征,其中主要有如下几个方面:

- 利用 CSS 的选择器提供高速的页面元素查找行为。

- 提供了一个抽象层来标准化各种常见的任务，可以解决各种浏览器的兼容性问题。
- 将复杂的代码精简化，提供连缀编程模式，大大简化了代码的操作。

提示

连缀编程模式（Chaining Pattern），允许我们在相同的元素上运行多条 jQuery 命令，一条接着另一条。这样的话，浏览器就不必多次查找相同的元素。

以上列出的只是 jQuery 的主要功能，此外它还为 JavaScript 语言增加了不少完善的特性，读者可以通过 jQuery 完善的文档获取 jQuery 更多的功能信息。

1.1.4 编写第一个 jQuery 网页

为方便读者学习，这里我们先简单用记事本来写一个 HTML 5 网页。

```
01 <!DOCTYPE html>
02 <html lang="zh-CN">
03 <head>
04   <meta charset="UTF-8">
05   <title>HELLO</title>
06 <body>
07   <div id="hi">Hello jQuery, 我来了</div>
08 </body>
09 </html>
```

这个网页代码结构比较简单，估计学习过 HTML 的人都能一眼看懂，网页中只有一个 div，会在网页中显示一行文字“Hello jQuery，我来了”。

此时我们要为 div 增加一个单击事件，首先要获取 div，使用 JavaScript 代码应该是：

```
document.getElementById("hi")
```

如果使用 jQuery，代码是：

```
$("#hi")
```

这样一比较，是不是 jQuery 书写更简单？下面使用 jQuery 为 div 增加事件。

(1) 首先在第 05 行后面添加对 jQuery 库的引用。这里要注意 js 文件存放的位置，如果在当前目录中，则不需要../，这个是指 js 文件在上一级目录中。

```
<script src="../jquery-3.3.1.js" type="text/javascript" ></script>
```

(2) 在文档的加载事件中，为 div 增加事件。

```
<script type="text/javascript">
$(document).ready(function(e) {
    $("#hi").click(function() {
        alert("hello");
    });
});
```

```
});  
</script>
```

首先使用\$("#hi")获取到 div，然后添加 click 事件。本例效果如图 1.4 所示。



图 1.4 第一个 jQuery 网页

1.2 jQuery 3 的特色

本节介绍一下 jQuery 3 的一些特色，这是 jQuery 更新的主要原因。

1.2.1 jQuery 3 的 Strict Mode

现在 jQuery 3 支持的大多数浏览器都有“use strict”（严格模式），顾名思义，这种模式使得 Javascript 在更严格的条件下运行。

严格模式的优点是：

- 消除 Javascript 语法的一些不合理、不严谨之处，减少一些怪异行为；
- 消除代码运行的一些不安全之处，保证代码运行的安全；
- 提高编译器效率，增加运行速度；
- 为未来新版本的 Javascript 做好铺垫。

如果使用 JavaScript，我们的代码可能要根据严格模式做很多修订，而 jQuery 新版本是用这个指令构建的，所以我们的代码不需要再设计严格模式，因此大多数现有代码不会做任何更改。

1.2.2 支持 for...of 遍历

jQuery 3 支持 for...of 语句，这是由 ECMAScript 6 中引进的一种 for 循环语句，为 Arrays、Maps 和 Sets 这样的可迭代对象提供了一种更直接的遍历方法。

在 jQuery 中，`for...of` 循环可以取代以前的 `$.each(...)` 语法，并且更容易通过 jQuery 的元素集合进行循环。对比代码如下所示：

```
var elems = $(".someclass");
// 传统的 jQuery 方式
$.each(function(i, elem) {

});
// ECMAScript 6 方式
for ( let elem of elems ) {

}
```

1.2.3 动画方面使用 requestAnimationFrame API

jQuery 3 使用 `requestAnimationFrame()` API 来执行动画，使动画运行得更加顺畅、快速。`requestAnimationFrame` API 只用于支持它的浏览器，对于那些很老的浏览器（如 IE9）jQuery 使用先前的 API 来作为显示动画的后备方案。

`requestAnimationFrame` 是 HTML5 中新增的 API，有点类似 `setTimeout` 和 `setInterval`，就是俗称的计时器。但与这两个计时器不同的是，`requestAnimationFrame` 不需要设置时间间隔，它采用系统时间间隔，保持最佳绘制效率，不会因为间隔时间过短造成过度绘制，增加开销；也不会因为间隔时间太长使动画卡顿不流畅，让各种网页动画效果能够有一个统一的刷新机制，从而节省系统资源，提高系统性能，改善视觉效果。

1.2.4 支持 SVG

SVG（Scalable Vector Graphics，可缩放矢量图形）是用于描述二维矢量图形的一种图形格式。它由万维网联盟制定，是一个开放标准。jQuery 对 SVG 的支持还不算完善，但从 jQuery 3 开始，操作类名的方法（例如 `.addClass()` 和 `.hasClass()`）将会支持 SVG。

也就是说，我们可以修改（添加、删除、切换）或者查找 SVG（可缩放矢量图形）下的 jQuery 类，然后使用 CSS 的类样式，这样就变相支持了 SVG。

1.2.5 :visible 和 :hidden 新改变

jQuery 3 还修改了一些 `:visible` 和 `:hidden` 的代码。只要元素具有任何布局盒，哪怕宽高为零，也会被认为是 `:visible`。比如，`br` 元素和不包含内容的行内元素现在都会被 `:visible` 这个过滤器选中。

因此，如果页面中包含如下的结构：

```
<div></div>
<br />
```


然后运行以下语句：

```
console.log($('body :visible').length);
```

在 jQuery 1.x 和 2.x 中得到的结果就会是 0，但在 jQuery 3 中会得到 2。

1.3 选择 jQuery 的开发工具

网站开发的工具多种多样，比如可以直接使用记事本或者是 Notepad++ 等工具来编写网页，但是这些工具没有代码提示功能，比如在编写 jQuery 代码时，如果能够有一款具有 jQuery 代码提示功能的工具，会使得网站开发人员的开发效率得到大幅提升，特别是对于网站开发的初学者来说，使用具有代码提示功能的编辑器，可以让初学者快速添加 jQuery API 的使用。Dreamweaver 是 Adobe 公司的一款可视化网页设计工具，它原生就附带了对 jQuery 的代码提示功能，因此笔者将在本书中选用 Dreamweaver 作为代码编写环境。

笔者使用的 Dreamweaver 版本为 CS 6，通过如下网址可以获取到关于 Dreamweaver 工具的更多详细信息：

```
http://www.adobe.com/cn/products/dreamweaver.html
```

接下来将通过一个使用 jQuery 的网站示例来演示如何在 Dreamweaver 中创建一个使用 jQuery 库的网页，步骤如下所示。

步骤 01 打开 Dreamweaver，单击主菜单中的“站点 | 新建站点”菜单项，Dreamweaver 将弹出如图 1.5 所示的新建站点对话框。

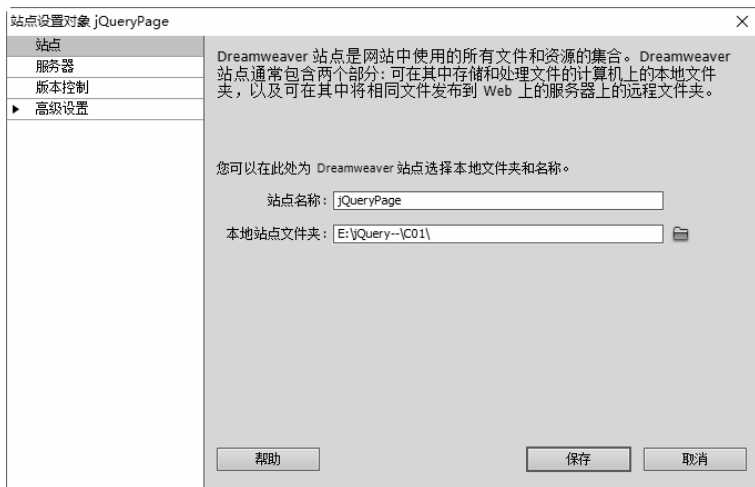


图 1.5 新建 Dreamweaver 网站

在“站点名称”文本框中，输入 jQueryPage 作为网站的名称，在本地站点文件夹文本框中，使用右侧的  按钮选择一个本地文件夹。然后单击“保存”按钮。

步骤 02 将下载的jQuery库复制到本地站点文件夹中，现在的站点管理器树状视图如图 1.6 所示。其中，JavaScript01.html是前面的例子。站点管理器在Dreamweaver的“文件|资源”视图中。

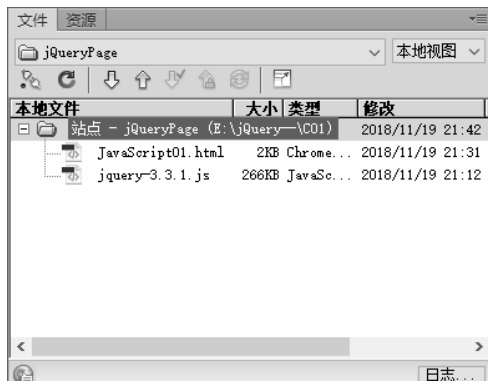


图 1.6 站点管理器视图

单击树状视图的根节点，即“站点”节点，从弹出的菜单中选择“新建文件”菜单项，在站点管理器中将新添加文件“untitled.html”，将其重命名为“index.html”，双击该文件，在Dreamweaver文档视图中将显示该文件的设计视图（此时是一个空白页）。

步骤 03 切换设计视图到源代码视图 **代码** **拆分** **设计** **实时视图**，将光标停在源代码的<head>和</head>之间的位置，从站点管理器中拖动jquery-3.3.1.js到源代码视图，Dreamweaver会自动添加对jQuery的引用，如图 1.7 所示。

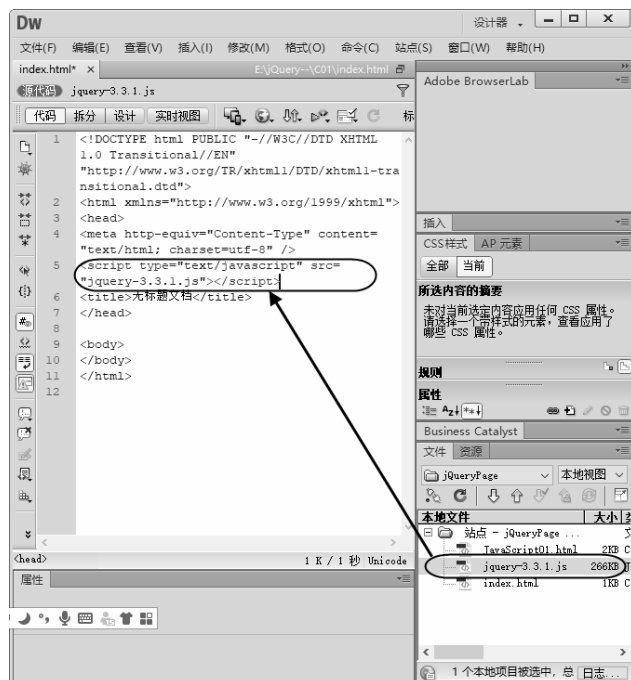


图 1.7 添加对 jQuery 库的引用

步骤 04 接下来通过一段 jQuery 的代码来看一看如何在页面上使用 jQuery 进行网页元素的控制。首先在页面的 `<body>` 和 `</body>` 之间放一个 `div` 元素，如下所示：

```
<body>
  <div id="msg">欢迎阅读 jQuery 从零开始学</div>
</body>
```

在 `<head>` 和 `</head>` 之间，添加如下代码来使用 jQuery 操纵这个 `div` 元素：

```
01 <head>
02 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
03 <title>第1个 jQuery 文档</title>
04 <script type="text/javascript" src="jQuery/jquery-3.3.1.js"></script>
05 <script type="text/javascript">
06   //jQuery 的页面加载事件
07   $(document).ready(function(e) {
08     $("#msg").css("font-size","9pt");    //更改 div 元素的字体
09     //向 div 中添加一个单击事件
10     $("#msg").click(function(e) {
11       alert($(this).html());
12     });
13     //向页面上添加一个新的 div 元素
14     $("<div>", {
15       style:"font-size:9pt",           //设置 div 的样式
16       text: "单击这里更改颜色",      //设置 div 的文本内容
17       //为文本添加单击事件
18       click: function(){
19         $(this).css("background","#9F3");
20       }
21     }).appendTo("body");              //将 div 添加到 body 中
22
23   });
24 </script>
25 </head>
```

`$`表示当前使用的是 jQuery 对象来操纵网页，在 `<script>` 区域，`$(document).ready` 是 jQuery 的页面加载事件，这个事件是传统 JavaScript 中的 `window.load` 事件的替代方法，当 DOM 载入就绪时，就会执行在括号中定义的代码，在页面加载事件中，完成了如下几个工作：

- 使用 jQuery 的选择器选择 `div` 元素，使用 jQuery 的函数 `css` 更改 `div` 的字体大小为 `9pt`。
- 为页面上的 `div` 元素添加 `click` 事件，当用户单击 `div` 元素时，就会弹出一个消息框。
- 向 HTML 页面上添加一个新的 `div` 元素，并关联了 `click` 事件。


至此这个示例就编写完了，在 Dreamweaver 中单击 ，会弹出一个菜单来选择浏览器，可根据自己熟悉的浏览器运行当前网页，运行效果如图 1.8 所示。



图 1.8 jQuery 网页示例运行效果

在编写 jQuery 代码时，可以发现 Dreamweaver 提供了方便的代码提醒功能，例如在创建了一个选择器之后，Dreamweaver 将自动跳出一系列可供操作的方法，如图 1.9 所示。

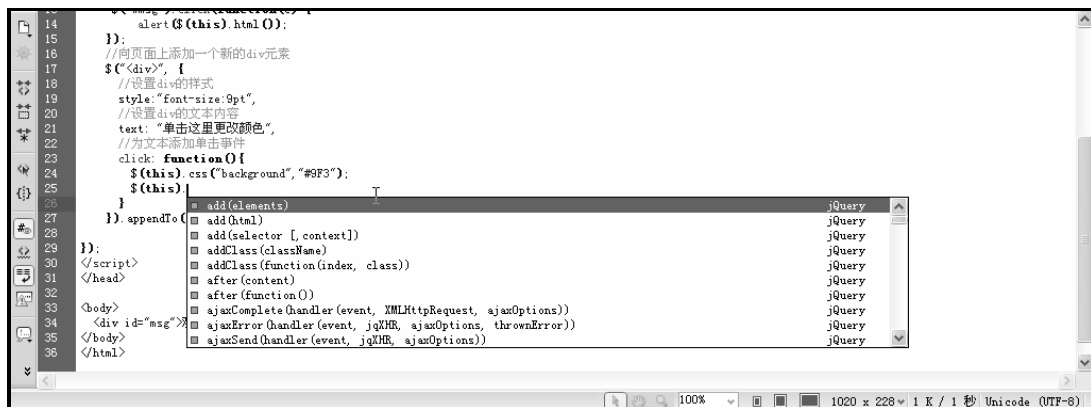


图 1.9 Dreamweaver 的代码提示功能

可以看到，像很多标准的代码编辑器一样，Dreamweaver 提供了 jQuery 的函数列表，这大大方便了对于 jQuery 不是特别熟悉的用户。

1.4 认识 jQuery 库的基础知识

由于 jQuery 的代码语法非常简洁，而实现的功能及其强大，因此在网页的代码中常见到对该库的引用。为了便于开发人员能够快速应用 jQuery 库，本节将简要介绍 jQuery 库的基础知识。

1.4.1 jQuery 库的核心方法——\$()

在 jQuery 程序代码中，不管是页面元素的选择，还是内置的功能方法，都是以一个美元符号“\$”和一对圆括号开始的。其实“\$()”方法是 jQuery 库中最重要、最核心的方法 jQuery() 的简写，主要用来选择页面元素或执行功能方法。因此如下代码：

```
$(function() {}); //执行一个匿名方法
$('#box'); //进行执行的 ID 元素选择
$('#box').css('color', 'red'); //执行功能方法
```

也可以写成如下形式:

```
jQuery(function () {});
jQuery('#box');
jQuery('#box').css('color', 'red');
```

查看 jQuery 的相关 API 资料, 可以发现 jQuery() 方法有 9 个重载, 分别如下:

(1) jQuery()

该方法返回一个空的 jQuery 对象, 不接受任何参数。

(2) jQuery(element)

该方法实现将一个 DOM 元素转化为 jQuery 对象。

(3) jQuery(elementArray)

该方法实现将多个 DOM 元素组成的数组转化为 jQuery 对象。

(4) jQuery(callback)

该方法等价于 jQuery(document).ready(callback), 主要用来实现绑定在 DOM 文档载入完成后执行的方法。

(5) jQuery(selector,[context])

该方法接收一个包含 jQuery 选择器的字符串, 在具体执行时, 会使用传入的字符串去匹配一个或多个元素。

(6) jQuery(object)

该方法将一个普通的对象包装成 jQuery 对象。

(7) jQuery(selection)

一个用于克隆的 jQuery 对象。

(8) jQuery(html,attributes)

该方法具体执行时, 不仅会根据传入的 html 标志代码动态创建由 jQuery 对象封装的 DOM 元素, 还会设置该 DOM 元素的属性、事件等。

(9) jQuery(html,[ownerDocument])

该方法具体执行时, 不仅会根据传入的 html 标志代码动态创建由 jQuery 对象封装的 DOM 元素, 还会指定该 DOM 元素所在的文档。

1.4.2 jQuery 代码的风格

了解了 jQuery 库的核心方法, 接着需要熟悉 jQuery 代码的风格, 例如:

```
$('#box').css('color', 'red'); //执行功能方法
```

在执行功能方法的时候要注意，其实 `css()` 这个功能方法并不是直接被 jQuery 对象调用执行，而是先获取元素，然后返回某个具体的对象，再调用 `css()` 这个功能方法。

不过值得注意的是，执行了 `css()` 这个功能方法后，最终返回的还是 jQuery 对象。这就是前面我们提到过的连缀方式，可以不停地连续调用功能方法，例如下面的代码：

```
$('#box').css('color','red').css('font-size','50px'); //连缀
```

最后要说明一下，jQuery 中的代码注释与 JavaScript 语言中的注释风格保持一致，即有两种最常用的注释，分别为：

- 单行注释：“//...”
- 多行注释：“/*...*/”

1.4.3 jQuery 库延迟等待加载模式

在 jQuery 程序代码中，为了让方法在浏览器加载完网页后执行，一般会使用 “`$()`” 将方法进行首尾包裹，即 `$(function){}`。为什么必须包裹所要执行的方法呢？

这是因为 jQuery 代码文件是在 `<body>` 标签元素之前加载，而 jQuery 代码文件里的方法一般需要操作 DOM 元素。为了让上述方法能够正常执行，必须等待所有的 DOM 元素加载后才能进行元素操作，于是就需要通过 “`$()`” 包裹方法来实现延迟等待加载功能。

在 JavaScript 原生代码里，原本是通过 `load` 事件来实现延迟等待加载，具体代码如下：

```
window.onload=function(){}; //JavaScript 等待加载
```

在 jQuery 代码里，为了实现上述功能，则需要通过如下代码：

```
$(document).ready(function(){}); //jQuery 等待加载
```

上述代码可以简写为：

```
$(function(){}); //jQuery 等待加载
```

那么上述两种等待加载方式有什么区别呢？具体区别请见表 1.1。

表 1.1 延迟等待加载区别

选 项	window.onload	\$(document).ready()
执行时机	必须等待网页全部加载完毕，然后再执行包裹代码	加载完毕，就能执行包裹代码
执行次数	只能执行一次，如果是第二次，那么第一次的执行会被覆盖	可以执行多次，第 N 次都不会被上一次覆盖
简写方案	无	\$(function){}

在实际应用中，很少直接去使用 `window.onload` 事件来实现延迟等待加载，这是因为该事件所关联的方法需要等待图片之类的大型元素加载完毕后才能执行。最头疼的就是网速较慢的情况下，页面已经全面展开，图片还在缓慢加载，这时页面上任何的 JavaScript 交互功能全部处在假死状态，并且只能执行单次，在多次开发和团队开发中会带来困难。

1.5 调试 jQuery 程序

大部分复杂的工具都带有调试功能，如 Dreamweaver、Visual Studio 和 Eclipse，如果我们用这些工具来开发 jQuery 代码，调试起来难度不是很大，但很多人习惯用文本工具直接写代码，这就增加了调试的难度，目前 Chrome 和 Firefox 浏览器都支持在浏览器中直接调试和修订。下面我们来看看这两个浏览器的调试手法。

1.5.1 在 Chrome 中调试

由于 jQuery 库始终是脚本语言，因此没有一个开发工具提供调试功能。不过值得庆幸，Firefox 和 Chrome 浏览器都提供了程序调试的功能，本节先讲解 Chrome 浏览器下的 jQuery 调试。

找到任意一个引用了 jQuery 的 HTML 文件，或者找到我们上一节使用的 index.html 文件，右击文件，在弹出的快捷菜单中选择“打开方式|Google Chrome”，这个时候在 Chrome 浏览器中显示的是文件效果。单击 Chrome 右侧菜单中的“更多工具|开发者工具”命令，默认效果如图 1.10 所示。

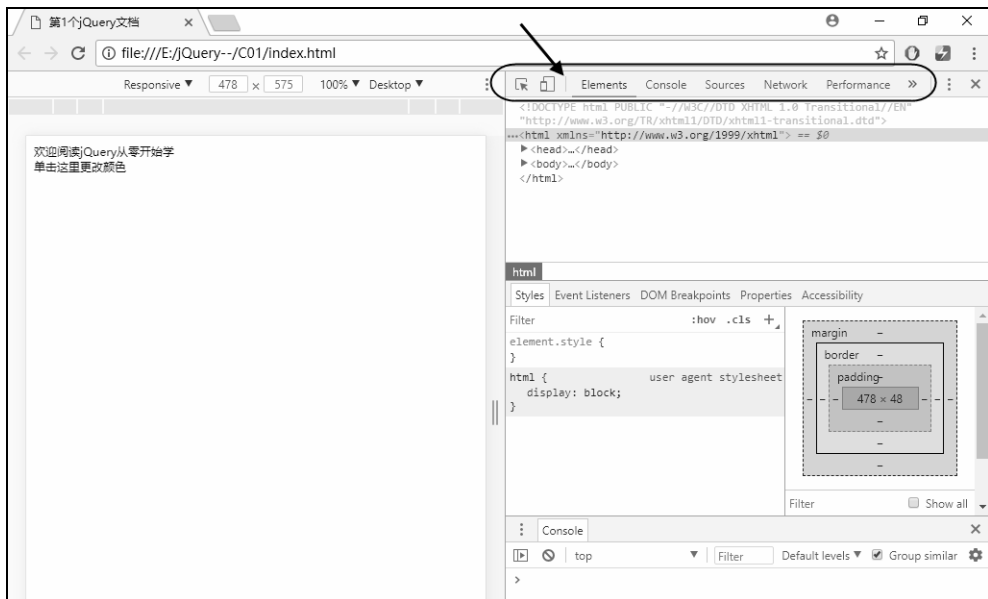


图 1.10 Chrome 中的开发者工具界面

右侧第一栏是 Chrome 的功能栏，其中 Sources 会显示本页面的源文件，包括所引用的 js 文件，单击 Sources 标签，可以看到左侧列出了本页面的源文件和 jQuery 库文件，如图 1.11 所示。双击 index.html 文件，就能看到该文件的所有代码。这里我们在第 21 行添加一个断点（单击该行的行号）。

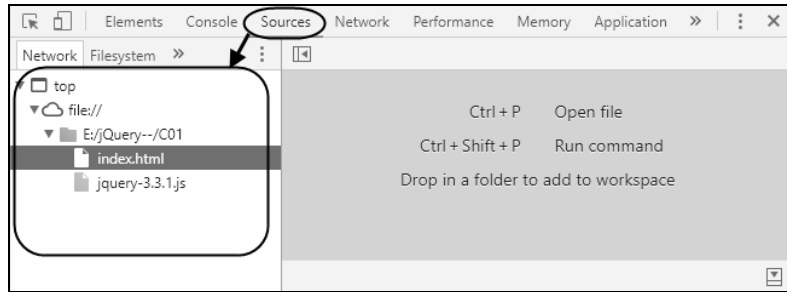


图 1.11 当前页面的源文件



断点，简单来说就是程序运行到这里的时候就会停下来。

单击浏览器的刷新按钮，因为我们将断点添加在“文本的单击事件”中，所以当我们单击第 2 行文本时，就会中断程序的执行，出现如图 1.12 所示的效果。

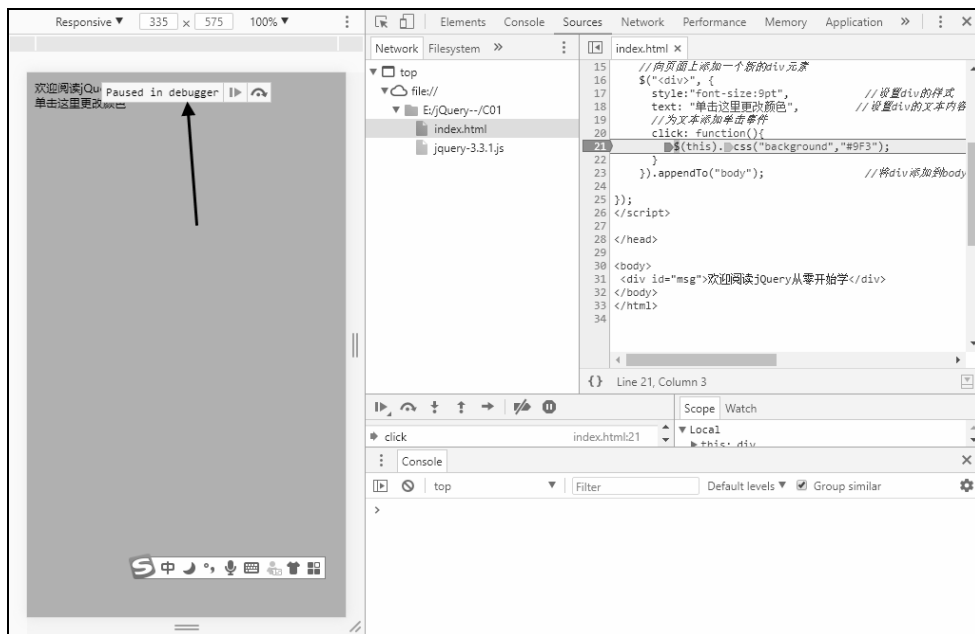


图 1.12 Chrome 中的断点

此时我们可以选中代码中的“\$(this)”，鼠标滑过时显示它的一些特性。如果你还不知道\$(this)是谁，也可以在 Chrome Console 视图中输出\$(this)的内容，如图 1.13 所示。



图 1.13 在程序中断时输出当前内容

Chrome 开发者工具非常强大，这里我们只认识了一个简单的设置断点，一些调试经验和技巧需要我们在反复的代码测试中持续总结和汲取。

1.5.2 在 Firefox 中调试

Firefox 浏览器也可以进行 jQuery 库的程序调试。打开 Firefox 浏览器，单击菜单栏中的“Web 开发者”|“切换工具箱”，或者使用快捷键 F12 都可以打开调试工具，如图 1.14 所示。

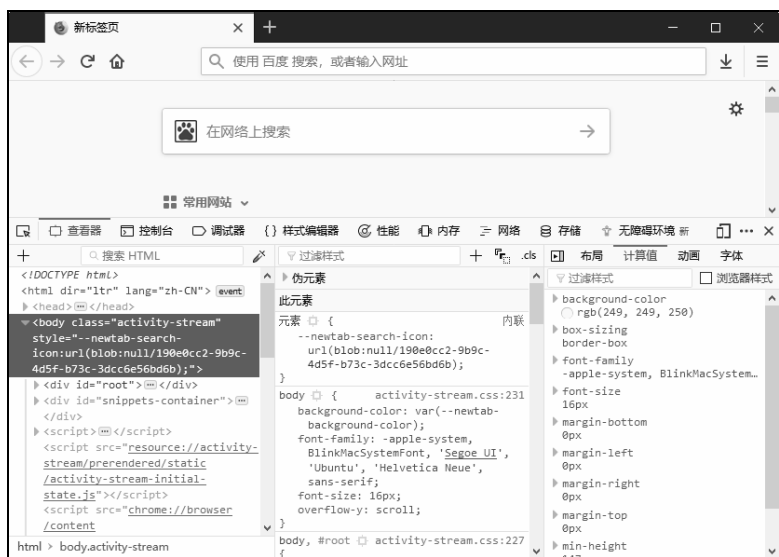


图 1.14 脚本调试界面

为了演示调试工具，通过浏览器打开 index.html。在该浏览器上按快捷键 F12，可以打开脚本调试界面，如图 1.15 所示。



图 1.15 脚本调试界面

在脚本调试界面中，选择“脚本”选项卡，在内容区域单击“启用”超级链接，即可启动对 jQuery 库程序的调试功能，如图 1.16 所示。



图 1.16 启用 jQuery 代码调试

启动 jQuery 代码调试后，选择功能栏中的“调试器”，单击第 21 行代码的行号“21”，在该行添加一个“断点”，如图 1.17 所示。如果行号变为蓝底白字，就说明断点添加成功。



图 1.17 添加断点

单击第 2 行文本，在调试窗口最右侧可以很方便地获取当前状态里一些变量或对象属性的信息，如图 1.18 所示。此时也可以单击功能栏中的“控制台”选项，输入 `$(this)` 来查看效果是否和 Chrome 测试的效果一致。

单击代码右上角工具栏 中的第 1 个按钮或者使用快捷键 F8，继续执行程序，第 2 行底色变绿，如图 1.19 所示。



图 1.18 监控视图

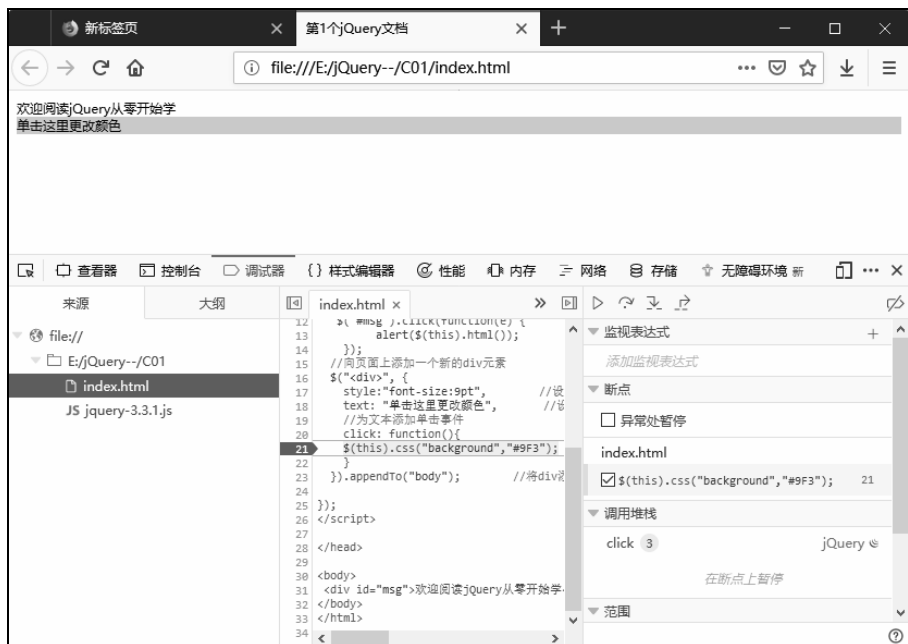


图 1.19 单步执行

从上面的执行结果可以发现，Firebug 调试工具也可以方便开发人员调试 jQuery 代码。

1.6 常见问题

1.6.1 为什么要使用一些著名公司的 CDN

jQuery 库文件既可以放在本地，也可以直接使用知名公司的 CDN（内容分发网络）。使用 CDN 的好处是这些大公司的 CDN 比较流行，用户访问你的网站之前很可能在访问别的网站时已经将库文件缓存在浏览器中了，所以能加快网站的打开速度。另外一个好处是显而易见的，就是节省了网站的流量带宽。

百度、新浪、谷歌和微软的服务器都存有 jQuery。如果要使用 CDN 上的 jQuery 库，只需要把引入代码：

```
<script type="text/javascript" src="../jquery-3.3.1.js"></script>
```

替换为：

```
<script src="http://libs.baidu.com/jquery/2.2.0/jquery.js"></script>
```



目前国内的 jQuery CDN 都比较落后，还没有发现 3.0 以上版本。所以如果要使用 CDN，推荐一个比较新的版本：<https://cdn.jsdelivr.net/npm/jquery@3.2/dist/jquery.min.js>。

为了提高 jQuery 库的加载速度，这些 CDN 一般会提供未压缩和压缩两个 jQuery 库版本，如果要使用压缩的 jQuery 库版本，则代码为：

```
<script src="http://libs.baidu.com/jquery/2.2.0/jquery.min.js"></script>
```

1.6.2 写 jQuery 和直接写 JavaScript 的区别

在第 1.1.2 节就举例演示过 jQuery 代码和 JavaScript 代码的区别，其中十几行的 JavaScript 代码用 2 行 jQuery 代码就可以替代了。本节详细地讲解一下两者的区别。

1. 定位元素

如果用 JavaScript 定位网页中 ID 为 div1 的元素，代码为：

```
document.getElementById("div1")
```

如果用 jQuery，则代码为：

```
$("#div1")
```



JavaScript 返回的结果是这个元素，而 jQuery 返回的结果是一个对象。

2. 改变元素的内容

改变页面中 div1 的内容时，JavaScript 代码为：

```
div1.innerHTML = "love";
```

jQuery 代码为：

```
div1.html("love ");
```

3. 显示/隐藏元素

显示/隐藏 div1 元素时，JavaScript 代码为：

```
div1.style.display = "none";  
div1.style.display = "block";
```

jQuery 代码为：

```
div1.hide();  
div1.show();
```

或

```
abc.toggle(); //在显示和隐藏之间切换
```

4. 为表单赋值

JavaScript 代码为：

```
div1.value = "love";
```

jQuery 代码为:

```
div1.val("love");
```

5. 设置元素不可用

JavaScript 代码为:

```
div1.disabled = true;
```

jQuery 代码为:

```
div1.attr("disabled", true);
```

1.6.3 jQuery 与其他 JavaScript 库的区别

目前流行的一些 JavaScript 库主要有 jQuery、Node.js、Vue.js、React.js。很多初学者会犹豫，我到底要怎么学，先学什么再学什么，或者说，是不是都要学。其实这些框架都有各自的功能，这里我们简单来了解一下。

jQuery 是一个运行在客户端的 JavaScript 库，专注于操纵 DOM，就是简化 JavaScript 对 DOM 的一些操作方式。

Node.js 是运行在服务器端的一个服务器程序。可以用 JavaScript 语言操作服务器层面的事务，比如创建 HTTP 链接、信息的 I/O 等。这些和 jQuery 一样都是用 JavaScript 语言进行操作执行。

Vue.js 偏重于前端的 UI，也就是前端页面，只关注视图层。

React.js 也是偏重于 UI，而且要熟悉 JSX，听起来会觉得复杂一些，它实现了单向响应的数据流，从而减少了重复代码。

简单来说也可以这么理解，jQuery 专注于操纵 DOM，Vue 和 React 偏重于 UI，而 Node.js 则偏重于后台的服务器端。

第 2 章

◀ jQuery 选择器 ▶

jQuery 的选择器是其核心功能，可以说是使用 jQuery 的重中之重，只有灵活地掌握了选择器，才能游刃有余地操纵 jQuery。在 jQuery 中，选择器按照选择的元素类别可以分为如下 4 种：

- 基本选择器：基于元素的 id、CSS 样式类、元素名称等使用基于 CSS 的选择器机制查找页面元素。
- 层次选择器：通过 DOM 元素间的层次关系获取页面元素。
- 过滤选择器：根据某类过滤规则进行元素的匹配。它又可以细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，以及表单对象属性过滤选择器。
- 表单选择器：可以在页面上快速定位某类表单对象。

2.1 基本选择器

jQuery 的基本选择器与 CSS 的选择器相似，它可以有如下 3 种：

- 标签选择器：按 HTML 元素的标签名称进行选择。
- id 选择器：取得文档中指定 id 的元素。
- 类选择器：根据 CSS 类来进行选择。

jQuery 还包含一个使用*的通配符选择器，用于选择所有的页面元素，几个元素之间还可以进行组合，jQuery 的基本选择器的描述参见表 2.1。

表 2.1 jQuery 基本选择器说明

名称	说明	举例
id 选择器	根据元素 id 选择	<code>\$("#divId")</code> 选择 id 为 divId 的元素
标签选择器	根据元素的名称选择	<code>\$("#a")</code> 选择所有<a>元素
CSS 样式类选择器	根据应用到 DOM 元素的 CSS 类进行选择	<code>\$(".bgRed")</code> 选择所用 CSS 类为 bgRed 的元素
*通配符选择器 selector1, selector2, selectorN	选择所有元素，使用通配符*可以将几个选择器用“,”分隔开，然后再拼成一个选择器字符串，会同时选中这几个选择器匹配的内容	<code>\$("#**")</code> 选择页面所有元素 <code>\$("#divId, a, .bgRed")</code>

现在创建一个新的页面，名为 `base_selector.html`，添加对 jQuery 库的引用，接下来通过示例来查看 jQuery 基本选择器的作用，HTML 元素定义如下：

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
04 <title>基本选择器</title>
05 <script type="text/javascript" src="../jquery-3.3.1.js"></script>
06 <style type="text/css">
07   body{
08     font-size:9pt;
09   }
10   .divclass{
11     font-style:italic;
12   }
13   .spanclass{
14     font-weight:bold;
15   }
16 </style>
17 </head>
18 <body>
19 <div id="div1">我是第1个 div</div>
20 <div id="div2">我是第2个 div</div>
21 <div class="divclass">我是第3个 div</div>
22 <span id="span1">我是第1个 span</span>
23 <span id="span2">我是第2个 span</span>
24 <span class="spanclass">我是第3个 span</span>
25 </body>
26 </html>
```

HTML 代码定义了 3 个 `div` 和 3 个 `span`，并且定义了两个 CSS 样式，接下来看一看如何通过 jQuery 基本选择器来实现选择效果。

2.1.1 标签选择器

首先必须在页面的 `head` 区添加对 jQuery 库的引用，也就是上面的代码第 5 行，后面的代码不再单独说明。

接下来使用 jQuery 的标签选择器，选中所有的 `div` 标签，并更改其字体大小为 18px。实现这个功能的代码如下：

```
//使用标签选择器更改字体大小
$("div").css("font-size","18px");
```

这里的 `$("div")` 就是选择元素名称为 `div` 的标签选择器，上述代码会同时更改 3 个 `div`（也

就是当前页面中所有的 div) 的字体大小, 因此运行时可以看到 3 个 div 的字体变成了 18px 的大小。

2.1.2 id 选择器

id 一般用来表示某个事物的唯一性, 这里的 id 选择器也就是只选择某一个具体的元素。使用 id 选择器选择示例中 id 为 div2 的 div, 将其背景色更改为红色, 代码如下所示:

```
//使用 id 选择器更改背景色
$("#div2").css("background","red");
```

可以看到, 运行之后果然第 2 个 div 已经更改了背景色, 如图 2.1 所示。



图 2.1 使用 id 选择器更改背景色



在 id 选择器中, id 前面必须跟一个 “#”, 以表明这是一个 jQuery 的 id 选择器。

2.1.3 类选择器

在 CSS 样式中, 可以为某一类元素设计统一的样式, 设计的代码如下:

```
.center {text-align: center}
```

上面是 CSS 的类选择器的代码。在 HTML 页面中, 可以用以下代码应用这个类:

```
<p class="center">
用了这个样式我就居中了。
</p>
```

如果要选中所有应用了 center 样式的元素, 在 jQuery 中需要使用 \$(".center") 形式。

这里还是使用前面的示例代码, 选择 CSS 类为 spanclass 的所有元素, 将其字体样式更改为斜体, 实现如下:

```
//使用类选择器设置字体样式
$(".spanclass").css("font-style","italic");
```


类选择器与 id 选择器之间的不同在于使用前缀“.”表示这是一个类选择器，无论是类选择器还是 id 选择器，都与 CSS 选择器具有相同的语法。

2.1.4 使用选择器组合

通过使用多个选择器的组合，可以同时更改选中标签的样式或内容，比如要更改 id 为 div2 和 span2 的元素，可以使用如下组合选择器：

```
//使用选择器组合
$("#div2,#span2").css("background","#9F0");
```

通过在括号内包含两个不同的选择器，就可以同时选中两个不同的元素进行样式设置，效果如图 2.2 所示。



图 2.2 使用选择器组合效果

2.1.5 使用*通配符选择器

“*”通配符选择器表示一次性选中页面上的所有元素，比如可以通过通配符选择器选中所有的元素，将其字体颜色更改为红色，代码如下：

```
//通配符选择器
$("*").css("color","red");
```

使用了通配符选择器后，果然所有的元素字体都变成了红色，读者可以亲自尝试下。

2.2 层次选择器

网页的 DOM 结构表现为树状结构，在选择元素时，通过 DOM 元素之间的层次关系，可以获取到需要的元素，比如当前节点的后代节点、父子关系的节点、兄弟关系的节点等，层次选择器的选择规则如表 2.2 所示。

表 2.2 层次关系的选择规则

名 称	说 明	举 例
ancestor descendant 后代选择器	使用"form input"的形式选中 form 中的所有 input 元素。即 ancestor(祖先)为 form、descendant(子孙)为 input	\$(".bgRed div") 选择 CSS 类为 bgRed 的元素中的所有<div>元素
parent > child 父子选择器	选择 parent 的直接子节点 child。child 必须包含在 parent 中并且父类是 parent 元素	\$(".myList>li") 选择 CSS 类为 myList 元素中的直接子节点对象
prev + next 相邻选择器	prev 和 next 是两个同级别的元素。选中在 prev 元素后面的 next 元素	\$("#hibiscus+img")选择 id 为 hibiscus 元素后面的 img 对象
prev ~ siblings 同级选择器	选择 prev 后面的根据 siblings 过滤的元素	\$("#someDiv~[title]")选择 id 为 someDiv 的对象后面所有带有 title 属性的元素

注：siblings 是过滤器。

新建一个名为 level_selector.html 的网页，在该页面添加几个具有层次关系的 HTML 元素，如下所示。

```

01 <body>
02 <ul id="nav">
03 <li><a href="#">产品介绍</a>
04   <ul id="product">
05     <li><a href="#">产品一</a></li>
06     <li><a href="#">产品二</a></li>
07     <li><a href="#">产品三</a></li>
08     <li><a href="#">产品四</a></li>
09     <li><a href="#">产品五</a></li>
10     <li><a href="#">产品六</a></li>
11   </ul>
12 </li>
13 <li><a href="#">服务介绍</a>
14   <ul id="services">
15     <li><a href="#">服务一</a></li>
16     <li><a href="#">服务二</a></li>
17     <li><a href="#">服务三</a></li>
18     <li><a href="#">服务四服务五</a></li>
19     <li><a href="#">服务四服务五服务六</a></li>
20     <li><a href="#">服务七</a></li>
21   </ul>
22 </li>
23 </ul>
24 </body>

```

在 HTML 中使用 ul、li 和 CSS 构建了一个下拉菜单，下拉菜单的 CSS 代码可以参考本书配套源代码，菜单效果如图 2.3 所示。



图 2.3 HTML+CSS 菜单效果

在示例 HTML 中，使用 ul 和 li 构建了层次结构的菜单项，接下来演示层次选择器的用法。

2.2.1 后代选择器

使用后代选择器，可以选择祖先下面的所有子元素，比如示例中构建了一个 2 层嵌套的 ul 和 li 菜单结构，如果要使得所有的 li 字体都变为粗体，无论是嵌套在哪一个层次，都可以使用后代选择器，如下所示：

```
<script type="text/javascript" src="../../jquery-3.3.1.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    //根据 ul 元素匹配所有的 li 元素，设置所有 li 元素的字体为粗体
    $("ul li").css("font-weight","bold");
    $("#services li").css("background","#9F9"); //让服务介绍的背景 li 为绿色
});
</script>
```

示例中使用了后代选择器，第 1 个 jQuery 选择器选中所有的 li 元素，更改 CSS 使其字体为粗，第 2 个后代选择器祖先使用了 id 选择器，后代指定为 li，祖先可以指定不同的选择器选择元素，而后代只能指定要选择的标签。

2.2.2 父子选择器

后代选择器会匹配所有的后代元素，而父子选择器只会匹配当前父元素下的所有子元素，比如要使菜单的主菜单项显示 14px 的字体，可以使用如下父子选择器：

```
//为了避免设置父元素的 CSS 继承到子元素，这里先单独设置了子元素的字体
$("#product,#services").css("font-size","9pt");
//根据父子元素规则设置子元素
$("#nav>li").css("font-size","14px");
```

第 1 行是为了避免设置了父类的 li 之后，CSS 会继承到子元素，因此为子元素单独指定了 CSS，这样在设置了 id 为 nav 的子元素 li 之后，就可以看到顶层菜单果然已经变成了 14 号字体，如图 2.4 所示。



图 2.4 父子选择器的效果



与后代选择器不同的是，父子选择器只会选择其父子关联的元素，而后代选择器会选择所有的子元素。

2.2.3 相邻选择器

相邻选择器允许选择相邻的元素，它匹配指定元素后面的元素，比如图 2.5 中“产品三”后面紧跟的是“产品四”，要选中“产品四”，可以用“产品三”的相邻选择器来进行选择，代码如下：

```
$("#prod1+li").css("font-style","italic"); //使用相邻选择器选择元素
```

示例将相邻的元素字体样式设置为斜体，结果如图 2.5 所示。

与相邻元素选择器相似的是 next 函数，它用来选中当前元素的下一个元素，因此可以使用 next 函数进行替换，如下所示：

```
$("#prod1").next().css("font-style","italic");
```



图 2.5 使用相邻选择器

2.2.4 平级选择器

与相邻选择器不同的是，平级选择器会选择当前元素的平级元素，下面通过一个例子来说明。要选择 id 为 srv2 的所有平级元素，可以使用如下语句：

```
$("#srv2~li").css("font-style","italic"); //使用平级选择器选择元素
```

通过将 id 为 srv2 的所有元素进行平级选择，可以看到所有出现在“服务二”后面的菜单项都变成了斜体，如图 2.6 所示。



图 2.6 使用平级选择器选择元素

使用~的平级选择器类似 nextAll 函数的效果，因此上面示例的替代语法如下：

```
$("#srv2").nextAll().css("font-style","italic");
```

如果要选择所有的相邻元素，包含前面的和后面的，可以使用 siblings 函数，如下所示：

```
//选择所有的相邻元素
$("#srv2").siblings("li").css("font-style","italic");
```

这一次，除了服务 2 没有变成斜体之外，可以看到所有的菜单项都变成了斜体，如图 2.7 所示。



图 2.7 所有相邻元素选择器

2.3 过滤选择器

除了基本选择器和层次选择器之外，jQuery 的强大之处还包括可以通过特定的过滤规则来筛选出所需的 DOM 元素，这种选择器称为过滤选择器，类似于 CSS 中的伪类选择器的语法。过滤选择器以冒号开头，过滤选择器根据其过滤规则的种类，又可以分为基本过滤选择器、内容过滤选择器、可见性过滤选择器、属性过滤选择器、子元素过滤选择器以及表单对象属性过滤器。下面分别对这几种不同的过滤选择器进行介绍。

2.3.1 基本过滤选择器

基本过滤选择器也可以称为简单过滤选择器，它是过滤选择器中使用最为广泛的一种，主要用来选择首、尾、指定索引、奇数或偶数位的元素等。表 2.3 所示为基本过滤器的规则列表。

表 2.3 基本过滤选择器规则列表

名称	说明	举例
:first	匹配找到的第一个元素	查找表格的第一行: <code>\$("#tr:first")</code>
:last	匹配找到的最后一个元素	查找表格的最后一行: <code>\$("#tr:last")</code>
:not(selector)	去除所有与给定选择器匹配的元素	查找所有未选中的 input 元素: <code>\$("#input:not(:checked)")</code>
:odd	匹配所有索引值为奇数的元素, 从 0 开始计数	查找表格的 1、3、5 等奇数行: <code>\$("#tr:odd")</code>
:even	匹配所有索引值为偶数的元素, 从 0 开始计数	查找表格的 2、4、6 等偶数行: <code>\$("#tr:even")</code>
:eq(index)	匹配一个给定索引值的元素, index 从 0 开始计数	查找第二行: <code>\$("#tr:eq(1)")</code>
:gt(index)	匹配所有大于给定索引值的元素, index 从 0 开始计数	查找第二、第三行, 即索引值是 1 和 2, 也就是比 0 大: <code>\$("#tr:gt(0)")</code>
:lt(index)	选择结果集中索引小于 N 的 elements, index 从 0 开始计数	查找第一、第二行, 即索引值是 0 和 1, 也就是比 2 小: <code>\$("#tr:lt(2)")</code>
:header	选择所有 h1、h2、h3 一类的 header 标签	给页面内所有标题加上背景色: <code>\$("#:header").css("background", "#EEE");</code>
:animated	匹配所有正在执行动画效果的元素	只有对不在执行动画效果的元素执行一个动画特效: <code>\$("#run").click(function){\$("#div:not(:animated)").animate({ left: "+=20" }, 1000);}</code>

在日常工作中, 基本过滤选择器比较常用在表格类型的选择上, 创建一个名为 `simple_filter_selector.html` 的网页, 在网页上添加一个 6 行 2 列的表格, 初始效果如图 2.8 所示。

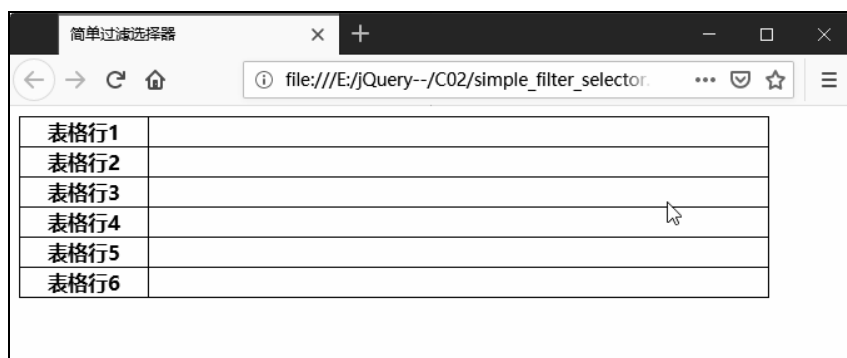


图 2.8 在应用 jQuery 选择器之前的效果

先使用 `first` 和 `last` 选中表格行的首尾, 并设置不同的颜色, 代码如下:

```
$("#tr:first").css("background", "#FF0"); // 表格第一行显示黄色
$("#tr:last").css("background", "#FCF"); // 表格的最后一行显示暖红
```

通过 `first` 和 `last` 设置了首尾行不同的颜色后，运行效果如图 2.9 所示。

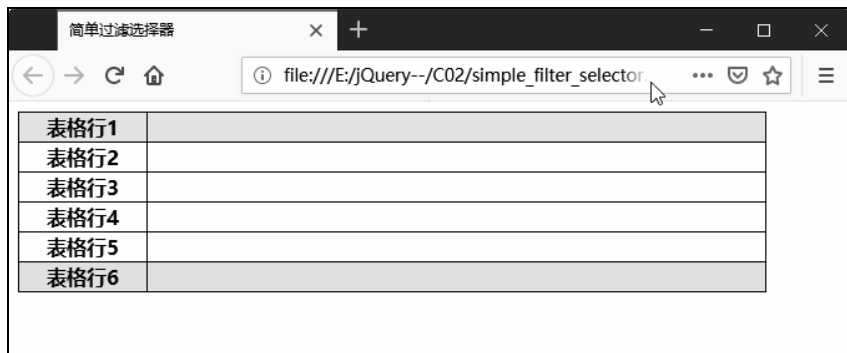


图 2.9 设置首尾行的颜色

在设置表格隔行颜色效果时，`even` 和 `odd` 是另外两个非常有用的过滤器，可以过滤出偶数行和奇数行的元素，比如要对表格的奇数行和偶数行显示不同的颜色，则可以使用如下代码：

```
$("#tr:odd ").css("background","#BBBBFF"); //表格的奇数行显示蓝色
$("#tr:even ").css('background', '#DADADA'); //表格的偶数行显示灰色
```

运行效果如图 2.10 所示。



图 2.10 隔行颜色效果



因为表格的索引 `index` 是从 0 开始，所以索引为偶数的实际上在表格中是奇数行。也就是索引 0 表示表格的第 1 行，索引 1 表示表格的第 2 行。所以看上去表格的显示效果与设计效果相反。如果表格添加一个表头可能看起来会更舒服一些。

在应用了 `even` 和 `odd` 选择器之后，发现它们将前面使用 `first` 和 `last` 过滤器设置的颜色也覆盖了，为了保留首尾行的颜色，可以使用 `not` 过滤器，它可以过滤指定的行，首尾行过滤的示例如下：

```
$("#tr:even:not(:first)").css("background","#BBBBFF"); //偶数行，但滤除第一行
$("#tr:odd:not(:last)").css("background","#DADADA"); //奇数行，但滤除最后一行
```

使用了 `not` 选择器后，可以看到现在运行时首尾行的颜色被忽略掉了，如图 2.11 所示。

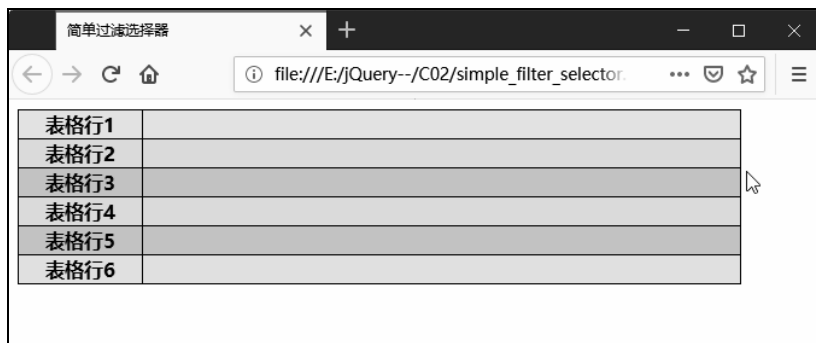


图 2.11 not 过滤器的效果

除了 first、last、even 和 odd 这类相对比较固定的过滤规则之外，还可以使用 eq 等于规则，选择特定索引位置的元素，gt 和 lt 分别返回大于或小于指定索引值的元素。

举例来说，想让表格中的第 5 行背景为红色、小于第 2 行的显示黄色、大于第 5 行的显示黑色，可以使用如下语句：

```

$("tr:eq(5)").css("background", "#F00"); //让第5行的背景为红色
$("tr:gt(5)").css("background", "#000"); //大于第5行的显示黑色
$("tr:lt(2)").css("background", "#FFC"); //小于第2行显示黄色

```

示例运行效果如图 2.12 所示。

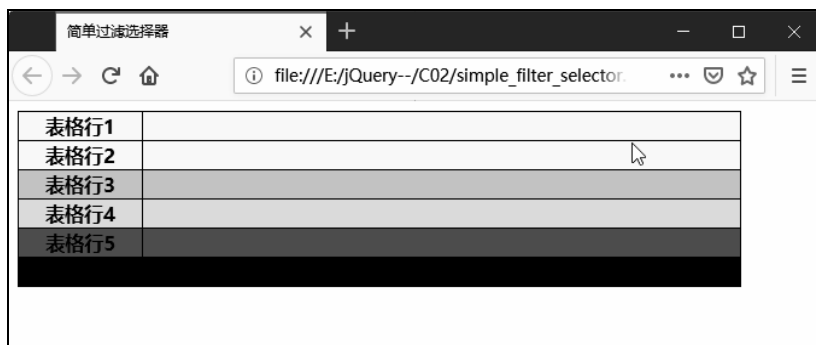


图 2.12 eq、gt 和 lt 运行效果

2.3.2 内容过滤选择器

内容过滤选择器可以根据 HTML 文本内容进行过滤选择，包含的过滤规则如表 2.4 所示。

表 2.4 内容过滤器规则列表

名称	说明	举例
:contains(text)	匹配包含给定文本的元素	查找所有包含"John"的 div 元素: \$("div:contains('John')")
:empty	匹配所有不包含子元素或者文本的空元素	查找所有不包含子元素或者文本的空元素: \$("td:empty")

(续表)

名称	说明	举例
:has(selector)	匹配含有选择器所匹配的元素	给所有包含 p 元素的 div 元素添加一个 text 类: \$("div:has(p)").addClass("text");
:parent	匹配含有子元素或者文本的元素	查找所有含有子元素或者文本的 td 元素: \$("td:parent")

为了演示内容过滤选择器，新建一个名为 content_filter_selector.html 的网页，在该 HTML 网页中添加一个 6 行 3 列的表格，并且加入一些内容，初始效果如图 2.13 所示。

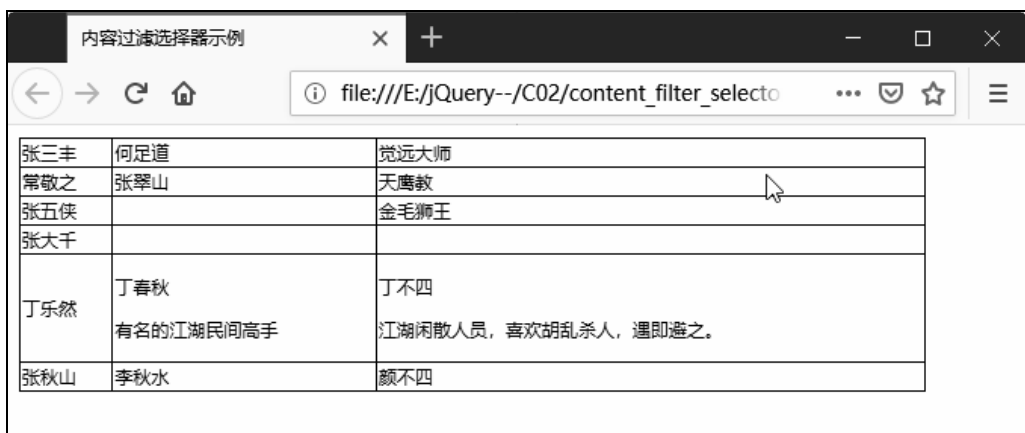


图 2.13 内容过滤选择器的初始网页

接下来添加如下内容过滤选择器的代码，读者可以打开本书配套的源代码，用注释的方式一次保留一行来查看其效果，限于本章的篇幅，这里列出了示例的主要代码：

```
<script type="text/javascript">
  $(document).ready(function(e) {
    $("td:contains('张')").css("background","#FFC");
                                                                    //将文字中含“张”的背景设置为淡黄
    $("td:empty").css("background","#060");
                                                                    //单元格中不包含内容的颜色，也不包含 (空格)的空单元格
    $("td:has(p)").css("background","#9F0"); //单元格中包含子元素<p>的颜色
    $("td:parent").css("color","#060"); //单元格中包含文本的前景色
  });
</script>
```

第 1 行使用 contains 查找表格中张姓的人，设置背景为淡黄，第 2 行设置单元格中为空的单元格的颜色，第 3 行设置单元格中包含段落标记 p 的颜色，第 4 行中设置单元格中包含文本的前景色，运行效果如图 2.14 所示。

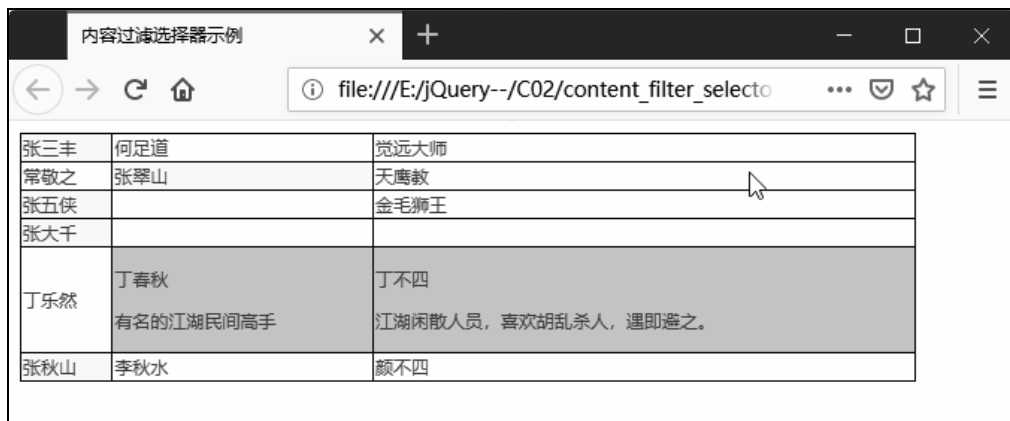


图 2.14 内容过滤选择器运行效果

2.3.3 可见性过滤选择器

可见性过滤器根据元素是否可见来查找元素，主要是用 `hidden` 查找隐藏的元素和 `visible` 查找可见的元素，其选择规则如表 2.5 所示。

表 2.5 可见性选择器规则列表

名称	说明	举例
<code>:hidden</code>	匹配所有的不可见元素	查找所有不可见的 <code>tr</code> 元素: <code>\$("tr:hidden")</code>
<code>:visible</code>	匹配所有的可见元素	查找所有可见的 <code>tr</code> 元素: <code>\$("tr:visible")</code>

`:hidden` 会匹配以下几种格式的元素：

- 具有 CSS 属性 `display` 属性值为 `none` 的元素。
- HTML 表单元素中的隐藏域，即 `type="hidden"` 的元素。
- 宽度和高度被显式设置为 0 的元素。
- 由于祖先元素为隐藏而导致无法显示在页面上的元素。

`:visible` 是指在屏幕上占用布局空间的元素，可见性元素的宽度和高度大于 0 时显示。



CSS 属性 `visibility:hidden` 或者是 `opacity:0` 被认为可见，这是由于它们仍然会占用布局空间。如果在动画期间隐藏一个元素，元素会被考虑为可见，直到动画终止，在动画期间显示一个元素，元素在动画开始时被认为可见。

新建一个名为 `hidden_filter_selector.html` 的网页，然后添加几个隐藏和显示的元素，HTML 代码如下：

```
01 <body>
02 <span></span>
03 <div></div>
04 <div style="display:none;">隐藏的元素</div>
```

```

05 <div></div>
06 <div class="starthidden">隐藏的页面元素</div>
07 <div></div>
08 <form>
09     <input type="hidden" />
10     <input type="hidden" />
11     <input type="hidden" />
12 </form>
13 <span>     </span>
14 <button>显示隐藏元素</button>
15 </body>

```

其中，`starthidden` 类指定了 `div` 的 `display` 属性为 `none`，表示一个隐藏的 `div`，接下来添加如下所示的可见性过滤选择器代码：

```

01 <script type="text/javascript">
02 $(document).ready(function(e) {
03     //在一些浏览器中，隐藏元素也包含 <head>、<title>、<script>等元素
04     //获取隐藏元素但排除<script>
05     var hiddenEls = $("body").find(":hidden").not("script");
06     $("span:first").text("找到" + hiddenEls.length + "个隐藏元素");
07     //$("div:hidden").show(3000); //动态地显示隐藏元素
08     $("span:last").text("找到" + $("input:hidden").length + "个表单隐藏域");
09     //为可见的按钮元素关联事件处理代码
10     $("div:visible").click(function () {
11         $(this).css("background", "yellow");
12     });
13     //为按钮关联事件处理代码，显示隐藏页面元素
14     $("button").click(function () {
15         $("div:hidden").show("fast");
16     });
17 });
18 </script>

```

代码中的实现步骤如下：

- 步骤 01** 第 5~6 行代码选中了页面上所有的隐藏元素，但是不包含 `script` 元素，这样就可以选取页面上所有非页面元素的隐藏元素，然后在第 1 个 `span` 中显示找到的隐藏元素，这里使用了 `:first` 基本过滤选择器。
- 步骤 02** 第 7 行代码选取隐藏的 `div` 元素，调用 jQuery 的 `show` 方法动态地显示隐藏元素。
- 步骤 03** 第 8 行代码显示隐藏的表单元素个数。
- 步骤 04** 第 10~11 行代码为当前显示出来的 `div` 元素关联单击事件，在单击时将背景色设为黄色。
- 步骤 05** 第 14~15 行代码为按钮关联事件，在事件处理代码中，将隐藏的 `div` 元素调用 `show` 函数动态地显示出来。

示例的运行效果如图 2.15 所示。

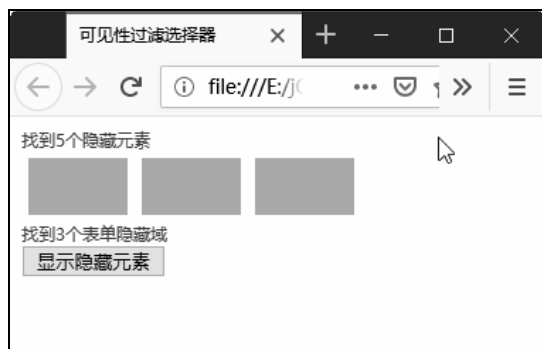


图 2.15 可见性过滤器示例效果

2.3.4 属性过滤选择器

属性过滤选择器是 jQuery 中非常有用的一种选择器，可以基于 HTML 元素的属性来选择特定的元素，除了根据不同的属性来选择元素，还可以根据不同的属性值来选择元素，属性过滤选择器的选择规则如表 2.6 所示。

表 2.6 属性过滤选择器规则列表

名称	说明	举例
[attribute]	匹配包含给定属性的元素	查找所有含有 id 属性的 div 元素: <code>\$("#div[id]")</code>
[attribute=value]	匹配给定的属性是某个特定值的元素	查找所有 name 属性是 newsletter 的 input 元素: <code>\$("#input[name='newsletter']").attr("checked", true);</code>
[attribute!=value]	匹配给定的属性是不包含某个特定值的元素	查找所有 name 属性不是 newsletter 的 input 元素: <code>\$("#input[name!='newsletter']").attr("checked", true);</code>
[attribute^=value]	匹配给定的属性是以某些值开始的元素	查找所有 name 以 news 开始的 input 元素: <code>\$("#input[name^='news']")</code>
[attribute\$=value]	匹配给定的属性是以某些值结尾的元素	查找所有 name 以 'letter' 结尾的 input 元素: <code>\$("#input[name\$='letter']")</code>
[attribute*=value]	匹配给定的属性是包含某些值的元素	查找所有 name 包含 'man' 的 input 元素: <code>\$("#input[name*='man']")</code>
[attributeFilter1] [attributeFilter2] [attributeFilterN]	复合属性选择器，需要同时满足多个条件时使用	找到所有含有 id 属性并且它的 name 属性是以 man 结尾的元素: <code>\$("#input[id][name\$='man']")</code>

从表 2.6 中可以知道，不仅可以根据属性名称进行选择，还可以根据属性与属性值的匹配规则来选择元素。接下来创建一个页面 `attribute_filter_selector.html`，在该页面上添加几个 HTML 元素，然后在 JavaScript 代码块中使用属性过滤器来选择元素，如下所示。

```
01 <html>
02 <head>
```

```

03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
04 <title>属性过滤选择器</title>
05 <script type="text/javascript" src="../jquery-3.3.1.js"></script>
06 <script type="text/javascript">
07   $(document).ready(function(e) {
08     $("div[id]").css("background","#0F0");      //具有 id 属性的元素的背景色
09     $('div[id="hey"]').css("font-size","14px"); //id 属性为 hey 元素的字体
10     $('div[id!="hey"]').css("font-size","16px");
                                                    //id 属性不为 hey 元素的字体
11     $('div[id^="the"]').css("color","#090");  //id 属性以 the 开头的前景色
12     $('div[id$="be"]').css("color","#C00");   //id 属性以 be 结束的前景色
13     $('div[id*="er"]').css("color","#360");  //id 属性值中包含 er 的前景色
14   });
15 </script>
16 </head>
17
18 <body>
19   <div id="hey">具有 id 属性 hey 的元素</div>
20   <div id="there">具有 id 属性 there 的元素</div>
21   <div id="adobe">具有 id 属性 adobe 的元素</div>
22   <div>无 id 属性</div>
23 </body>
24 </html>

```

在 HTML 的 body 区，定义了 4 个 div 元素，分别为前 3 个 div 指定了不同的 id，并且具有一个无任何属性的 div 元素，在 JavaScript 代码部分，分别使用了属性过滤选择器的不同设置来选择元素并且设置其颜色或字体，运行时的效果如图 2.16 所示。

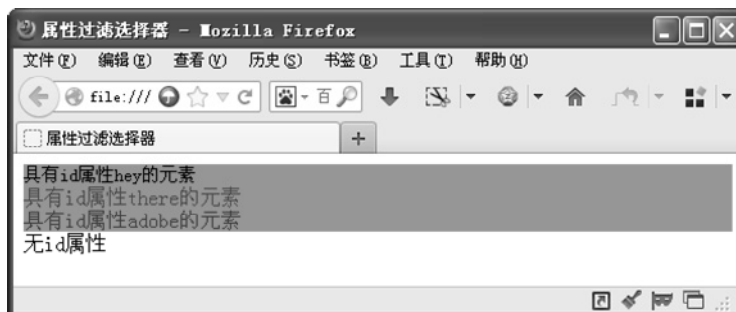


图 2.16 属性选择器运行效果

2.3.5 子元素过滤选择器

这个过滤器是指根据父元素中的某些过滤规则来选择子元素，例如可以选择父元素的第 1 个子元素 (:first-child) 或者是最后 1 个子元素 (:last-child)，或者是父元素中特定位置的子元素，其规则如表 2.7 所示。

表 2.7 子元素过滤器规则列表

名称	说明	举例
:nth-child(index/ even/odd/equation)	匹配其父元素下的第 N 个子元素或奇偶元素 ':eq(index)' 只匹配一个元素，将为每一个父元素匹配子元素： nth-child 是从 1 开始的，而:eq()是从 0 算起的 可以使用以下几项： <ul style="list-style-type: none"> • :nth-child(even) • :nth-child(odd) • :nth-child(3n) • :nth-child(2) • :nth-child(3n+1) • :nth-child(3n+2) 	在每个 ul 中查找第 2 个 li: \$("ul li:nth-child(2)")
:first-child	匹配第一个子元素 ':first' 只匹配一个元素，为每个父元素匹配一个子元素	在每个 ul 中查找第一个 li: \$("ul li:first-child")
:last-child	匹配最后一个子元素 ':last'只匹配一个元素，为每个父元素匹配一个子元素	在每个 ul 中查找最后一个 li: \$("ul li:last-child")
:only-child	如果某个元素是父元素中唯一的子元素，就将会被匹配；如果父元素中含有其他元素，就不会被匹配	在 ul 中查找是唯一子元素的 li: \$("ul li:only-child")

nth_child 可以根据指定的索引位置、奇数位、偶数位等来匹配元素，这个选择规则常用来选择某些特定集合性质的元素中的子元素，接下来创建一个名为 child_filter_selector.html 的网页，在其中添加一个 5 行 4 列的 HTML 表格。然后看一下 jQuery 的子元素过滤器如何选择其中的元素：

```

01 <script type="text/javascript">
02   $(document).ready(function(e) {
03     $("tr td:nth-child(2)").css("background","#090");
                                //让表格单元格第2列显示绿色背景
04     $("tr td:nth-child(even)").css("background","#CCC");
                                //偶数单元格显示灰色
05     $("tr td:nth-child(odd)").css("background","#9F0");
                                //奇数单元格显示淡绿
06     $("table tr:first-child").css("background","#F00");
                                //让表格第一行显示红色背景
07     $("table tr:last-child").css("background","#99F");
                                //让表格最后一行显示紫色背景
08     $("td p:only-child").css("background","#0F0");
                                //单元格中含有唯一元素<p>的背景设置
09   });
10 </script>

```

第 1 个选择器使用的是索引选择器，这将使得它选择表格行的第 2 个单元格，也就是第 2 列显示为绿色；第 2 个和第 3 个使用偶数和奇数选择器选择偶数和奇数单元格设置颜色；第 4 个和第 5 个选择器选择表格的第 1 行和最后一行设置背景色；最后一个选择器选择具有 p 元素的单元格，运行效果如图 2.17 所示。

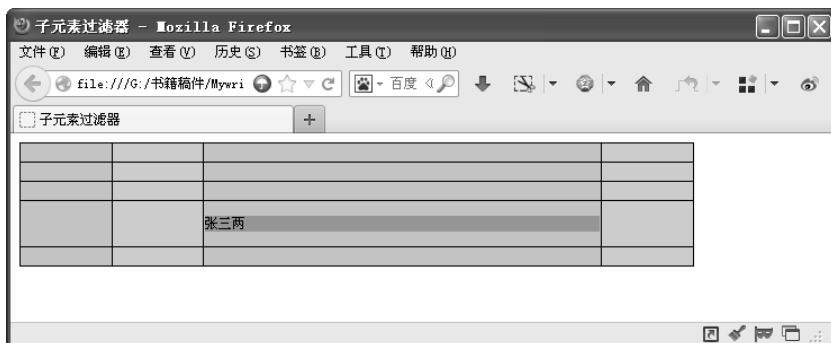


图 2.17 子元素过滤器的示例效果

如果注释掉奇数和偶数选择器，则可以看到第 4 个和第 5 个选择器的效果，如图 2.18 所示。

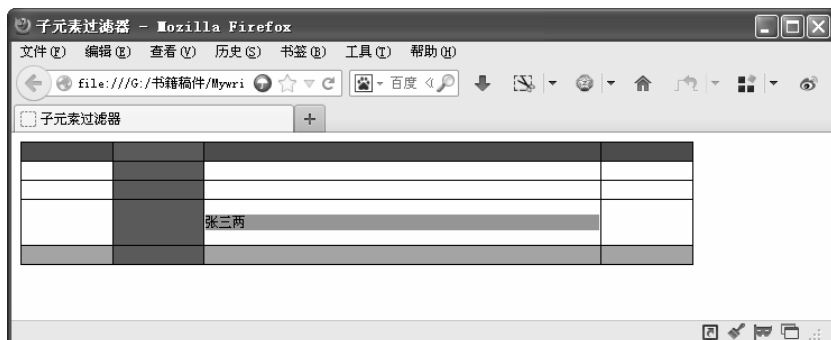


图 2.18 首尾行的选择效果

2.3.6 表单对象属性过滤器

这种类型的过滤器可以根据表单中某对象的属性特征来获取表单元素，比如表单元素的 enabled、disabled、selected 以及 checked 属性，其过滤规则如表 2.8 所示。

表 2.8 表单对象属性过滤器规则列表

名称	说明	解释
:enabled	匹配所有可用元素	查找所有可用的 input 元素: \$("input:enabled")
:disabled	匹配所有不可用元素	查找所有不可用的 input 元素: \$("input:disabled")
:checked	匹配所有被选中元素（复选框、单选按钮等，不包括 select 中的 option）	查找所有选中的复选框、单选按钮元素: \$("input:checked")
:selected	匹配所有选中的 option 元素	查找所有选中的选项元素: \$("select option:selected")

可以看到，使用表单对象属性过滤器，可以对表单中的控件元素的可用（enabled）、不可用（disabled），Checkbox 控件的选择（checked）与 select 控件的选中（selected）这些属性进行选择，这样使得在开发表单时可以快速地选中所需要的控件。

新建一个名为 form_filter_selector.html 的网页，然后在该网页中构建一个表单，效果如图 2.19 所示。

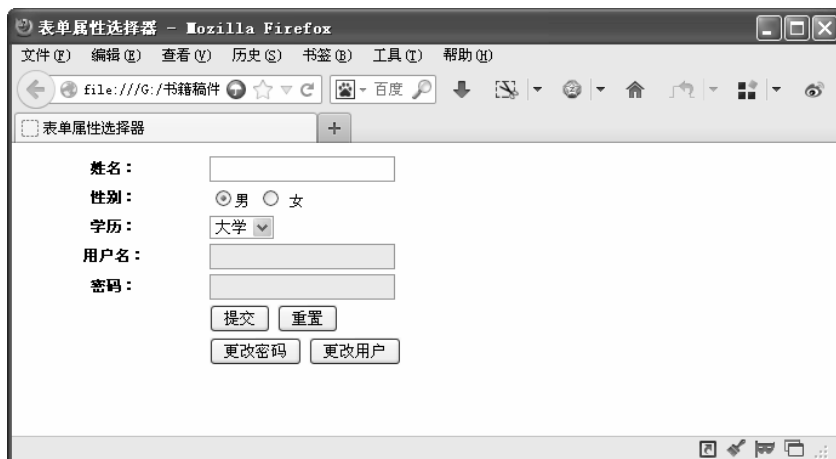


图 2.19 表单界面

由图 2.19 可以看到，这个表单包含了两个单选按钮，用来供用户选择性别；一个 select 下拉列表框，供用户选择学历；两个禁用掉的 input 控件，接下来看一看如何使用表单属性过滤器来选择元素，如下所示。

```

01 <script type="text/javascript">
02   $(document).ready(function(e) {
03     $("input:enabled").css("background","#FFF"); //已启用控件的背景色设置
04     $("input:disabled").css("background","#CFF"); //已禁用控件的背景色设置
05     $("input:disabled").attr("disabled",false);
06                                     //将禁用的文本框更改为 enabled
07     $("input:checked").click(
08       function() {
09         alert("我被选中了");
10       }
11     );
12     $("select option:selected").css("background","#FF0");
13                                     //选中的列表框背景变色
14   });
15 </script>

```

在 ready 事件主体中，代码完成的功能分别如下所示：

(1) 第 3 行和第 4 行代码，分别使用 enabled 和 disabled 来选中禁用和启用的 input 控件，然后使用 css 函数来设置其背景色。

(2) 第 5 行代码使用 `attr` 将已经被禁用掉的 `input` 控件设置为 `enabled`，即将 `disabled` 属性设置为 `false`。

(3) 第 6 行代码为具有 `checked` 属性的控件关联了 `click` 事件。

(4) 第 11 行代码将 `select` 控件中 `option` 集合具有 `selected` 属性的元素的背景色更改为黄色。

应用了表单属性过滤器应用效果如图 2.20 所示。

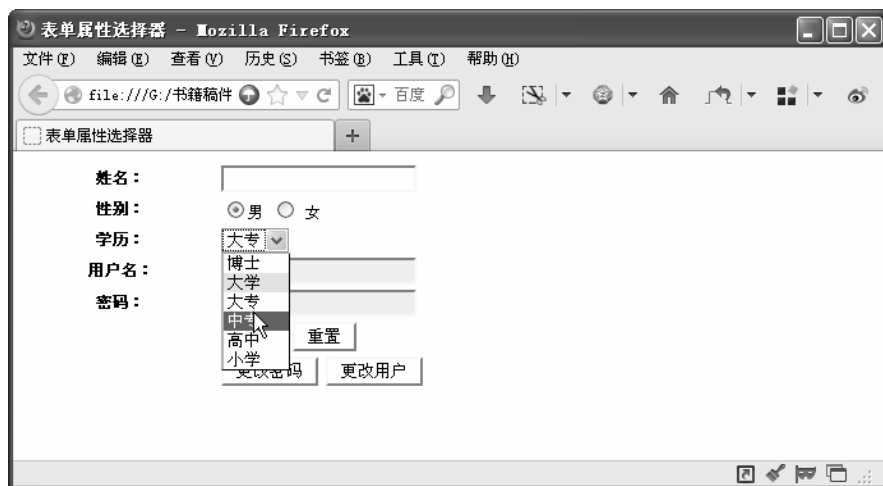


图 2.20 表单属性过滤器应用效果

可以看到表单属性在根据表单的属性设置来选择表单方面确实比较强大。

2.4 表单选择器

在学过了表单属性过滤器之后，接下来看一看 jQuery 的表单选择器，表单选择器提供了灵活的方法来选择表单中的元素。举例来说，如果要统一为表单中的 `input` 控件设置样式或者属性，使用表单选择器可以快速一次到位地进行设置，jQuery 中可供使用的表单选择器如表 2.9 所示。

表 2.9 表单选择器规则列表

名称	说明	解释
<code>:input</code>	匹配所有 <code>input</code> 、 <code>textarea</code> 、 <code>select</code> 和 <code>button</code> 元素	查找所有的 <code>input</code> 元素： <code>\$(":input")</code>
<code>:text</code>	匹配所有的文本框	查找所有文本框： <code>\$(":text")</code>
<code>:password</code>	匹配所有密码框	查找所有密码框： <code>\$(":password")</code>
<code>:radio</code>	匹配所有单选按钮	查找所有单选按钮： <code>\$(":radio")</code>
<code>:checkbox</code>	匹配所有复选框	查找所有复选框： <code>\$(":checkbox")</code>

(续表)

名称	说明	解释
:submit	匹配所有提交按钮	查找所有提交按钮: \$(" :submit")
:image	匹配所有图像域	匹配所有图像域: \$(" :image")
:reset	匹配所有重置按钮	查找所有重置按钮: \$(" :reset")
:button	匹配所有按钮	查找所有按钮: \$(" :button")
:file	匹配所有文件域	查找所有文件域: \$(" :file")

可以看到, 表单选择器可以匹配当前文档或者是某一个表单内部的所有表单元素, 比如可以同时选中所有的按钮或者是输入框。下面以上一小节中创建的表单为例, 演示一下表单选择器的使用效果。新建一个名为 `form_selector.html` 的网页, 然后复制在上一小节中创建的表单 HTML 代码, 接下来添加如下代码, 使用表单选择器来选择表单中的元素。

```

01 <script type="text/javascript" src="../jquery-3.3.1.js"></script>
02 <script type="text/javascript">
03   $(document).ready(function(e) {
04     $(" :input").css("background", "#FFC"); //设置所有 input 元素的背景色
05     $(" :text").hide(3000); //隐藏所有文本框对象
06     $(" :text").show(3000); //显示所有文本框对象
07     $(" :password").hide(3000); //隐藏所有密码框对象
08     $(" :password").show(3000); //显示所有密码框对象
09     $(" :button").css("font-weight", "bold"); //显示按钮对象的字体
10     $(" :radio").css("background", "#0F0"); //设置单选按钮的背景色
11   });
12 </script>

```

整个代码由如下几个选择器组成:

- (1) 选中文档界面中的所有 `input` 元素, 设置其背景色为黄色。
- (2) 用了两个 `text` 选择器, 选中所有的文本框对象, 先使用 `hide` 函数让其动态地隐藏, 然后使用 `show` 函数让其慢慢地显示。
- (3) 使用两个 `password` 选择器, 先隐藏所有的密码文本框元素, 然后显示所有的密码框元素。
- (4) 为网页上所有的按钮指定字体为加粗显示。
- (5) 为网页上所有的单选按钮设置背景色。

使用了表单选择器的页面效果如图 2.21 所示。

在运行时可以看到, 文本框会慢慢地隐藏和显示, 这是 jQuery 的 `hide` 和 `show` 这两个函数的效果, 这两个函数可以动态地显示和隐藏页面上的元素, 在实际的工作中非常有用。



图 2.21 表单选择器效果

2.5 常见问题

2.5.1 \$("input")和\$(".input")两个选择器的区别

\$("input")是标签选择器，选择页面中所有的 input 元素。

\$(".input")是表单选择器，选择表单中的 input、select、textarea、button 元素。

比如以下代码中，\$("input")会包含两个元素，而\$(".input")会包含 4 个元素。

```

01 <form id="form1" action="#" method="post">
02   <label for="userName">用户名: </label>
03   <input type="text" id="userName"/>
04   <label for="pwd">密码: </label>
05   <input type="password" id="pwd"/>
06   <select><option>第一行</option>
07 <option>第二行</option>
08 <option>第三行</option></select>
09   <textarea cols="13" rows="20"></textarea>
10 </form>

```

2.5.2 子选择器和后代选择器的区别

在层次选择器中，我们介绍了子选择器和后代选择器，因为两个比较像，所以很多人表示看不懂。

比如\$("#div1 > p")和\$("#div1 p")，请读者先思考下这两个分别代表什么意思。

- 前者表示查找到 id 为 div1 的元素中子标签为 p 的元素，这里只需要查找一个就不再查找了。
- 后者表示查找 id 为 div1 中所有标签为 p 后代的元素，即子元素、孙子元素、曾孙子元素等都可以。

2.5.3 获得 class 为 sub 的元素的子节点下的所有<a>标签

这里出了一个动手题，主要是考察读者对本节的理解，答案是很简单的一句代码：

```
$(".sub > a");
```

这里用到了类选择器和父子选择器。

第 3 章

◀ 用jQuery来操作DOM ▶

在使用 JavaScript 编写网页代码的过程中，多数时间都在操纵 DOM，比如 Ajax 返回的 Json 数据、动态向 DOM 添加显示节点，或者是动态更改页面上元素的 CSS、属性等。DOM 的全称是“Document Object Model”，即文档对象模型，是一种与浏览器、平台和语言无关的接口，它可以让用户代码访问任何浏览器中呈现的元素，可以将 DOM 看作是网页呈现的一种标准。

本章主要内容：

- 修改元素属性
- 修改元素内容
- 动态创建内容
- 动态插入节点
- 动态删除节点

3.1 修改元素属性

要使用 jQuery 操纵 DOM，必须先使用选择器选中一个或多个元素，由于 jQuery 是对结果集进行隐式迭代的操作，因此一个 jQuery 对象可以同时多个元素进行属性更改。

3.1.1 获取元素的属性

获取和设置属性使用 jQuery 的 `attr` 方法，而移除属性使用 `removeAttr` 方法，其中获取元素属性的 `attr` 语法如下所示：

```
$(selector).attr(attribute)
```

其中，`selector` 是 jQuery 的选择器，`attr` 中的参数 `attribute` 是指定要获取的元素的属性名称。举个简单的例子，要想获取图像的地址，可以使用如下语句：

```
$("#img").attr("src");
```

下面新建一个 `get_set_attributes.html` 的网页，在这个网页中来演示如何获取 DOM 元素的属性值，HTML 元素如下所示。

```

01 <body>
02 <ul id="nav">
03 <li><a href="http://www.xxx.com/companyinfo" id="company_info" title="
介绍公司的相关资讯
04 ">
05 公司信息</a></li>
06 <li><a href="http://www.xxx.com/productinfo" id="product_info" title="
公司的产品信息">
07 产品简介</a></li>
08 <li><a href="http://www.xxx.com/companyculture" id="culture_info"
title="公司的文化信息">
09 公司文化</a></li>
10 <li><a href="http://www.xxx.com/contactus" id="contactus" title="联系方
式">联系我们</a>
11 </li></ul>
12 <div id="content"></div>
13 <!--属性的信息显示如下-->
14 <div id="attr_info">
15 <input id="btn_getAttr" type="button" value="显示属性信息">
16 </div>
17 </body>

```

在这里构建了一个菜单,用作网站的导航栏, id 为 btn_getAttr 的按钮将获取页面上的 DOM 不同的属性值,代码如下所示:

```

<script type="text/javascript">
$(document).ready(function(e) {
    $("#btn_getAttr").click(function(e) {
        var str="<br>"+"("#company_info").attr("title");
            //显示 id 为 company_info 的 title 属性值
        str+="<br>"+"("#product_info").attr("href");
            //显示 id 为 product_info 的 href 属性值
        str+="<br>"+"("#culture_info").attr("id");
            //显示 id 为 culture_info 的 id 属性值
        str+="<br>"+"("#btn_getAttr").attr("value");
            //显示 id 为 btn_getAttr 的 value 属性值
        $("#attr_info").append(str);        //在 div 中显示属性的值
    });
});
</script>

```

在示例代码中,使用 attr 分别获取了 4 个 HTML 元素的属性值,保存到 str 字符串中,通过运行可以看到不同的属性值已经成功地显示到了页面上,如图 3.1 所示。



图 3.1 获取 DOM 元素的属性值

3.1.2 设置元素的属性

要设置元素的属性，同样使用 `attr` 函数，语法如下：

```
$(selector).attr(attribute,value)
```

其中 `attribute` 用来指定属性的名称，`value` 用来指定属性的值。下面在 3.1.1 小节的 `get_set_attributes.html` 页面中添加一个新的按钮，在 jQuery 的页加载事件中通过如下代码来设置 DOM 元素的属性：

```
$("#btn_setAttr").click(function(e) {
    $("#company_info").attr("title","公司的发展历程和发展经验");
    //设置 title 属性

    $("#product_info").attr("href","http://www.microsoft.com");
    //设置 href 属性

    $("#culture_info").attr("id","btn_culture_info"); //设置 id 属性
    $("#contactus").attr("title","欢迎联系我们来获取更多信息");
    //设置联系人的 title 属性

});
```

可以看到，使用 `attr` 设置属性是使用“属性名称：属性值”匹配的语句，`attr` 还可以同时设置两个以上的属性值，如下代码所示：

```
//同时设置两个属性的值
$("#company_info").attr({
    "href":"http://www.microsoft.com/",
    "title":"欢迎进入微软公司网站"
});
```

可以看到，通过属性名/值对的方式，同时为 `href` 和 `title` 这两个属性设置了属性值。

3.2 修改元素内容

有3个方法可以用于获取HTML元素的内容，分别是：

- `text()`：设置或返回所选元素的文本内容。
- `html()`：设置或返回所选元素的内容（包括HTML标记）。
- `val()`：设置或返回表单字段的值。

`text` 和 `html` 的明显区别是，`text` 只返回元素的文本内容，而 `html` 返回的是将HTML解析后的内容。`val` 返回的是表单的内容。新建一个名为 `get_set_content.html` 的网页，在该网页中添加如下HTML代码：

```

01 <body>
02 <p id="test">
03     有3个方法可以用于获取<strong>HTML元素</strong>的内容，分别是：<br/>
04     <strong>text(): 设置或返回所选元素的文本内容</strong><br/>
05     <strong>html(): 设置或返回所选元素的内容（包括HTML标记）</strong><br/>
06     <strong>val(): 设置或返回表单字段的值</strong><br/>
07 </p>
08 <textarea name="textvalue" cols="80" rows="5"></textarea>
09 <div>
10 <button id="btn1">显示文本</button>
11 <button id="btn2">显示HTML</button>
12 </div>
13 </body>

```

在页面中放置了一个 `id` 为 `test` 的 `p` 元素，在段落内部设置了一些HTML代码，在段落下面添加一个 `textarea` 元素，用于显示文本的 `btn1` 和显示HTML的 `btn2`。接下来对 `btn1` 编写代码，使其获取 `p` 元素内部的文本内容，并显示到 `textarea` 中。`btn2` 将显示HTML内容到 `textarea` 元素，这两个按钮的事件处理实现如下：

```

01 <script type="text/javascript">
02     $(document).ready(function(e) {
03         $("#btn1").click(function(e) {
04             var textStr=$("#p").text(); //获取段落的文本内容
05             $("#textvalue").text(textStr); //在textarea中显示文本内容
06         });
07         $("#btn2").click(function(e) {
08             var htmlStr=$("#test").html(); //获取段落的HTML内容

```

```

09     $("#textvalue").text(htmlStr); //在 textarea 中显示 HTML 内容
10     });
11 });

```

按钮 btn1 使用 text 获取了段落的文本内容，并显示到 textarea 中，显示效果如图 3.2 所示。

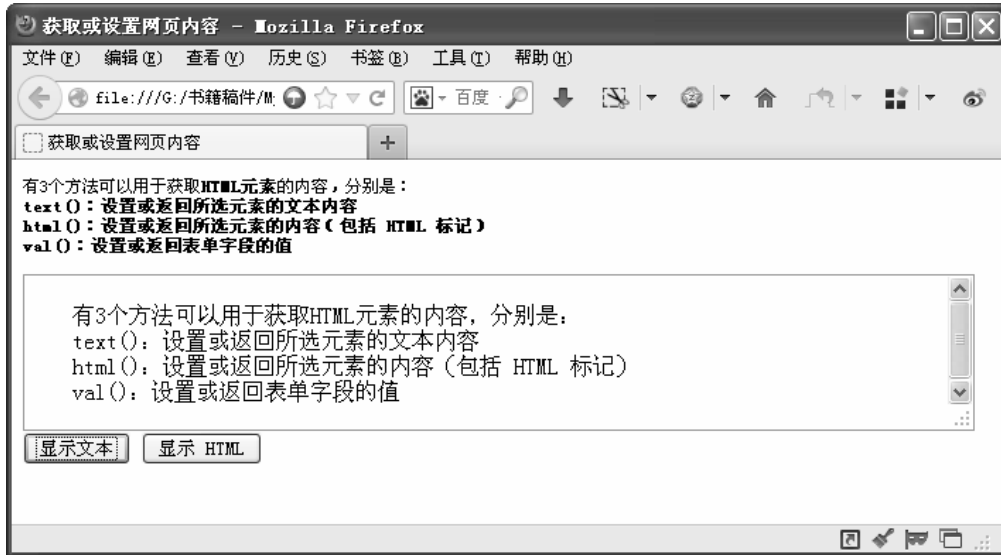


图 3.2 显示文本内容

可以看到，即便段落标记内部包含了 HTML 字符串，但是 text() 仅仅只是取出其中的文本内容，在为 textarea 赋值时也使用了带参数的 text 函数，这个参数将作为文本内容设置给 textarea，因此在 textarea 中显示了 HTML 文本内容。

btn2 按钮使用了 html() 方法，用来获取 HTML 格式的内容，其输出结果如图 3.3 所示。

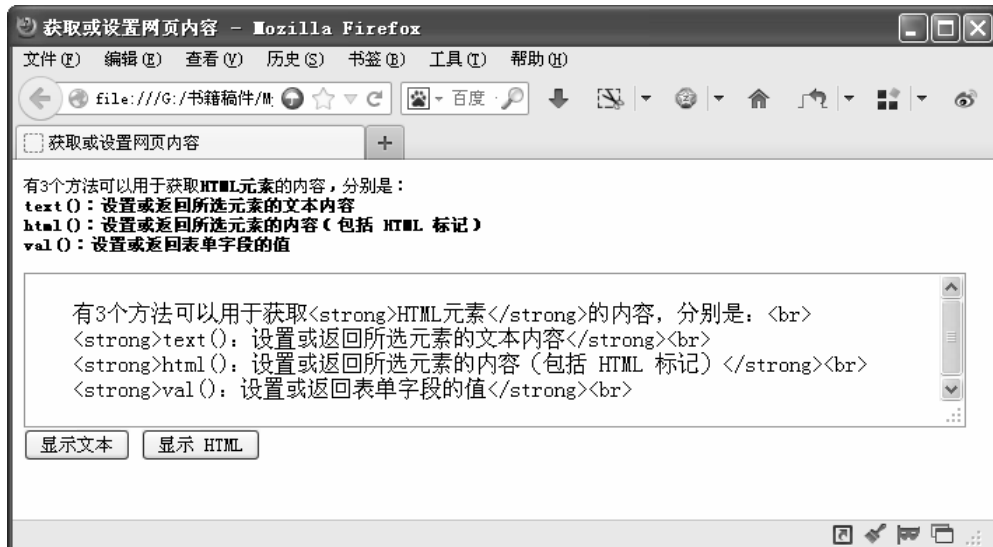


图 3.3 显示 HTML 内容

html()方法显示了段落标签中的 HTML 元素，可以看到它包含了 HTML 标记。同样的，如果为 html()方法带了一个参数，就表示将为指定的目标元素设置 HTML 内容，比如可以编写如下代码：

```
$("#test2").html(htmlStr); //将HTML内容设置到id为test2的div中
```

这就使得 HTML 代码的内容设置给了名为 test2 的 div，这样就可以动态地为 div 添加新的 HTML 内容了。

3.3 动态创建内容

jQuery 还允许开发人员动态地为页面添加内容，类似于 JavaScript 语言中的 createElement，jQuery 动态创建 HTML 元素使用工厂函数 \$() 实现，语法如下：

```
$(html)
```

其中参数 html 是要动态创建的 HTML 标记，它会动态创建一个 DOM 对象，但是这个 DOM 对象并没有添加到 DOM 对象树中，可以使用如下几个 jQuery 函数来将其添加到 DOM 对象树：

- append(): 在被选元素的结尾插入内容。
- prepend(): 在被选元素的开头插入内容。
- after(): 在被选元素之后插入内容。
- before(): 在被选元素之前插入内容。

在下一节会介绍这些方法的具体使用方式，本节主要关注如何使用工厂方法 \$() 来动态创建页面元素。举个例子，要向页面上插入一个新的 div 元素，可以使用如下语句：

```
$("#<div>", {
  text: "这是动态创建的页面元素",
  click: function(){
    $(this).toggleClass("test"); //设置其 toggleClass 为 test
  }
}).appendTo("body"); //将其添加到 body 元素中其他元素的后面
```

可以看到，在工厂函数 \$() 中不仅可以指定要创建的标签，还可以为其设置各种不同的属性，最后的 appendTo 将这个新创建的 div 元素添加到页面上。

3.4 动态插入节点

动态创建的节点如果不插入到 DOM 对象树中，那么是不会在页面上呈现的。想要动态插入节点，可以使用表 3.1 所示的几种方法。

表 3.1 动态插入方法列表

方法名称	方法描述
append()	方法在被选元素的结尾（仍然在内部）插入指定内容
appendTo()	方法在被选元素的结尾（仍然在内部）插入指定内容
prepend()	方法在被选元素的开头（仍位于内部）插入指定内容
prependTo()	方法在被选元素的开头（仍位于内部）插入指定内容
after()	在被选元素后插入指定的内容
before()	在被选元素前插入指定的内容
insertAfter()	把匹配的元素插入到一个指定的元素集合的后面
insertBefore()	把匹配的元素插入到一个指定的元素集合的前面



append 和 appendTo、prepend 和 prependTo 的不同之处在于内容和选择器的位置。

接下来新建一个名为 insert_elements.html 的页面，在其中添加如下 HTML 代码：

```

01 <style type="text/css">
02 body,td,th,input {
03     font-size: 9pt;
04 }
05 </style>
06 </head>
07 <body>
08 <div id="idbtn">
09 <input type="button" name="idAppend" id="idAppend" value="append 方法" />
10 &nbsp;
11 <input type="button" name="idappendTo" id="idappendTo" value="appendTo
方法" />
12 &nbsp;
13 <input type="button" name="idprepend" id="idprepend" value="prepend 方
法" />
14 &nbsp;
15 <input type="button" name="idprependTo" id="idprependTo"
value="prependTo 方法" />
16 &nbsp;
17 <input type="button" name="idbefore" id="idbefore" value="before 方法" />
18 &nbsp;
19 <input type="button" name="idafter" id="idafter" value="after 方法" />
20 &nbsp;
21 <input type="button" name="idinsbefore" id="idinsbefore"
value="insertBefore 方法" />
22 &nbsp;

```

```

23 <input type="button" name="idinsafter" id="idinsafter"
value="insertAfter 方法" />
24 </div>
25 <div id="idcontent">使用不同的按钮，用不同的方法插入页面<br/></div>
26 </body>

```

代码中构建了多个不同的按钮，其中每个按钮将对应到一种不同的插入方法。为每个按钮关联的事件处理语句如下所示。

```

01 <script type="text/javascript">
02   $(document).ready(function(e) {
03     $("#idAppend").click(
04       function() {
05         //追加内容
06         $("#idcontent").append("<b>使用 append 添加元素</b><br/>");
07       }
08     );
09     $("#idappendTo").click(
10       function() {
11         //追加内容，语法与 append 颠倒
12         $("<b>使用 appendTo 添加元素</b><br/>").appendTo("#idcontent");
13       }
14     );
15     $("#idprepend").click(
16       function() {
17         //插入前置内容
18         $("#idcontent").prepend("<b>使用 prepend 插入前置内容</b><br/>");
19       }
20     );
21     $("#idprependTo").click(
22       function() {
23         //在元素中插入前缀元素，与 prepend 的操作语法颠倒
24         $("<b>使用 prependTo 添加元素</b><br/>").prependTo("#idcontent");
25       }
26     );
27     $("#idbefore").click(
28       function() {
29         //在指定元素的前面插入内容
30         $("#idcontent").before("<b>使用 before 添加元素</b><br/>");
31       }
32     );
33     $("#idafter").click(
34       function() {
35         //在指定元素的后面插入内容

```

```

36         $("#idcontent").after("<b>使用 after 添加元素</b><br/>");
37     }
38 );
39 $("#idinsbefore").click(
40     function() {
41         //在指定元素前面插入内容，与 before 语法颠倒
42         $("#<b>使用 insertBefore 添加元素</b><br/>").insertBefore
($("#idcontent");
43     }
44 );
45 $("#idinsafter").click(
46     function() {
47         //在指定元素的后面插入内容，与 after 的语法颠倒
48         $("#<b>使用 insertAfter 添加元素</b><br/>").insertAfter
($("#idcontent");
49     }
50 );
51 });
52 </script>

```

可以看到，在每个按钮的事件处理代码中，分别调用了不同的插入方法。通过这个示例可以看到各种不同的插入语句的使用方式和语法结构，比如 `append` 和 `appendTo`、`prepend` 和 `prependTo` 就只是选择器的不同。这个示例的运行效果如图 3.4 所示。

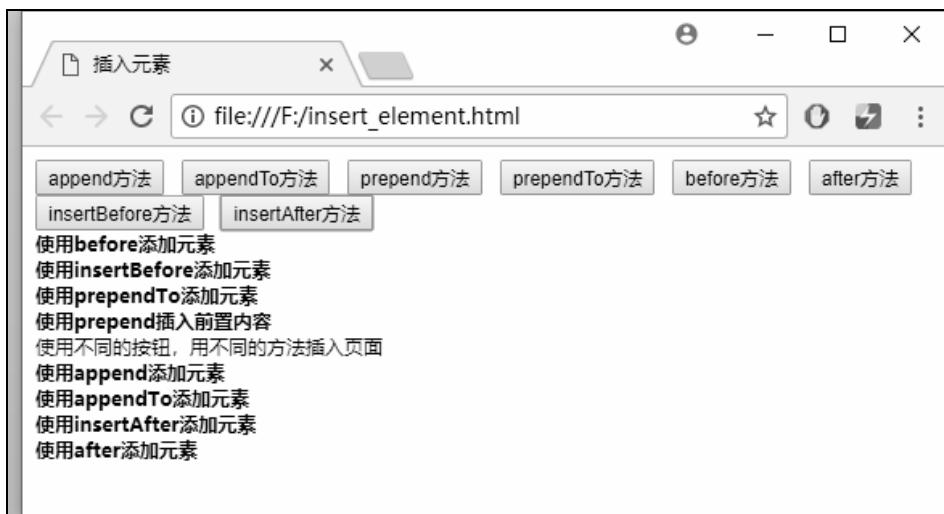


图 3.4 不同的插入语句的示例效果

3.5 动态删除节点

从网页上删除节点也是日常工作中经常遇到的一种操作，jQuery 提供了两个可以用来从 DOM 元素树中移除节点的方法，分别是：

- `remove()`方法：用来删除指定的 DOM 元素，它会将节点从 DOM 元素树中移除，但是会返回一个指向 DOM 元素的引用，因此它并不是真正地将 jQuery 引用到的元素对象删除，而是可以通过这个引用来继续操作元素。
- `empty()`方法：该方法也不会删除节点，只是清空节点中的内容，DOM 元素依然保持在 DOM 元素树中。

`remove()`方法会把元素从 DOM 对象树中移除，但是不会把引用了这些对象的 jQuery 对象删除，因此还是可以使用 jQuery 对象来进行一些操作。而 `empty` 只是将元素中的内容进行清空。接下来，我们创建一个名为 `dynamic_remove.html` 的网页，向其中插入一些 HTML 元素，然后分别演示使用 `remove` 和 `empty` 的效果，HTML 定义如下所示。

```
01 <body>
02 <div id="idwelcome">演示使用 remove 和 empty 方法<br/></div>
03 <div id="idtip"><b>remove 方法会从 DOM 树中移除节点</b><br/></div>
04 <div id="idsenc"><b>empty 方法只是清除元素的内容</b><br/></div>
05 <div><input name="btnremove" type="button" id="btnremove" value="remove
方法" />
06 &nbsp;
07 <input name="btnempty" type="button" id="btnempty" value="empty 方法" />
08 </div>
09 </body>
```

在 `body` 区，可以看到放了 3 个用来显示消息的 `div`，另外两个 `div` 中放置了两个按钮，分别用来调用 `remove` 方法和 `empty` 方法，这两个按钮的事件处理代码如下所示。

```
01 <script type="text/javascript" src="../jquery-3.3.1.js"></script>
02 <script type="text/javascript">
03   $(document).ready(function(e) {
04     $("#btnremove").click(
05       function(){
06         var id1=$("#idtip").remove();           //移除 DOM 对象
07         $("body").append(id1);                 //重新添加已被移除的 DOM 对象
08       });
09     $("#btnempty").click(
10       function(){
11         var id1=$("#idsenc").empty();          //清除 DOM 对象
12         //重新添加 DOM 对象的内容
```

```

13         id1.append("这是重新添加的内容哦，原来的内容已被清除了!");
14     });
15 });
16 </script>

```

remove 按钮内部调用了 remove 方法，尽管这个元素已经从 DOM 中移除了，但是 jQuery 仍然引用着这个对象，因此仍然可以将其添加到 body 中，使之经历了删除后又重新添加的过程。empty 方法只是清除了 DOM 中的内容，之后又重新向 div 中添加了元素。单击两个按钮后的效果如图 3.5 所示。

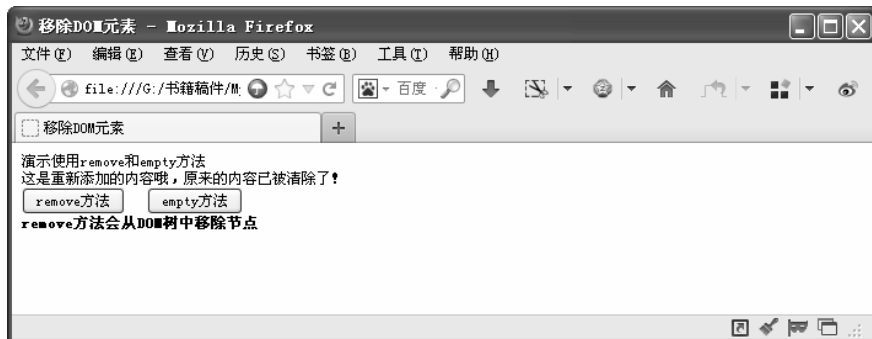


图 3.5 移除元素后的效果

3.6 实例 1：超链接提示效果

在项目的所有页面中，经常会看到超级链接的影子。如果要让超级链接自带提示，只需要在超级链接标签里设置 title 属性就可以了，具体语法如下：

```
<a href="#" title="超级链接提示信息">提示</a>
```

上述代码虽然可以实现提示效果，但是提示效果的响应速度非常缓慢。为了实现良好的人机交互，需要手动实现提示效果。

具体要求：当鼠标移动到超级链接上时，快速地出现提示。设计一个包含两个超级链接对象的页面 two_a.html，代码如下：

```

<body>
<!--超级链接-->
<p><a href="#" class="tooltip" title="超链接提示1">提示1.</a></p>
<p><a href="#" class="tooltip" title="超链接提示2">提示2.</a></p>
</body>

```

设置关于超级链接的类样式 tooltip，修改超级链接的相关样式，具体代码如下：

```

#tooltip{
    position:absolute;

```



```
border:1px solid #333;
background:#f7f5d1;
padding:1px;
color:#333;
display:none;
}
```

编写 jQuery 代码，实现超级链接提示功能，具体代码如下：

```
01 $(function(){
02     var x = 10;
03     var y = 20;
04     $("a.tooltip").mouseover(function(e){
05         this.myTitle = this.title;
06         this.title = "";
07         var tooltip = "<div id='tooltip'>"+ this.myTitle +"</div>";
                                //创建 div 元素
08         $("body").append(tooltip);           //把它追加到文档中
09         $("#tooltip")
10             .css({
11                 "top": (e.pageY+y) + "px",
12                 "left": (e.pageX+x) + "px"
13             }).show("fast");           //设置 x 坐标和 y 坐标，并且显示
14     }).mouseout(function(){
15         this.title = this.myTitle;
16         $("#tooltip").remove();           //移除
17     }).mousemove(function(e){
18         $("#tooltip")
19             .css({
20                 "top": (e.pageY+y) + "px",
21                 "left": (e.pageX+x) + "px"
22             });
23     });
24 })
```

在上述代码中，第 4~13 行设置鼠标滑入超级链接时的处理方法，其中第 7 行创建一个包含 title 属性值的提示框（<div>标签元素对象），第 8 行将所创建的提示框对象追加到文档中，剩下的代码主要用来设置 x 和 y 坐标，使得提示框显示在鼠标位置的旁边。第 14~16 行设置鼠标滑出超级链接时的处理方法，即主要是移除提示框。第 17~23 行设置鼠标在超级链接上移动时的处理方法，即通过 css()方法设置提示效果的坐标，以达到提示效果跟随鼠标一起移动的效果。

在浏览器中运行页面，效果如图 3.6 所示；当鼠标滑入超级链接时，就会快速出现提示，效果如图 3.7 所示；当鼠标滑出超级链接时，提示效果就会消失。

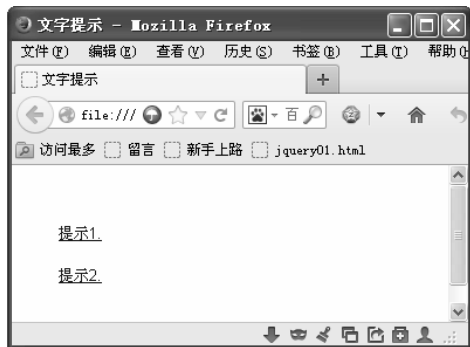


图 3.6 浏览页面



图 3.7 鼠标滑入时效果

3.7 实例 2：图片预览效果

为了让项目中的页面更漂亮，我们经常会用到图片。对于页面中的图片来说，一个常见的功能就是图片的预览效果。

这个具体要求是：如果将鼠标移动到图片上，将在该图片的右下角出现一张与之相对应的大图片，以达到图片预览的效果。设计一个包含 4 张图片对象的页面 `picture_CTP.html`，代码如下：

```
<body>
<ul>
  <!--插入4张图片-->
  <li><a href="images/apple_1_bigger.jpg" class="tooltip" title="苹果 iPod">
</a></li>
  <li><a href="images/apple_2_bigger.jpg" class="tooltip" title="苹果 iPod
nano"></a></li>
  <li><a href="images/apple_3_bigger.jpg" class="tooltip" title="苹果
iPhone"></a></li>
  <li><a href="images/apple_4_bigger.jpg" class="tooltip" title="苹果
Mac"></a></li>
</ul>
</body>
```

在上述代码中，用超级链接标签包含 4 张图片。

接下来，设置列表和图片的相关样式，以达到预期的排列顺序，具体代码如下：

```
ul,li{
  margin:0;
  padding:0;
}
li{
```

```

list-style:none;
float:left;
display:inline;
margin-right:10px;
border:1px solid #AAAAAA;
}
img{border:none;
}

```

继续编写 jQuery 代码，实现图片预览功能，具体代码如下：

```

01 $(function(){
02     var x = 10;
03     var y = 20;
04     $("a.tooltip").mouseover(function(e){
05         this.myTitle = this.title;
06         this.title = "";
07         var imgTitle = this.myTitle? "<br/>" + this.myTitle : "";
08         //创建 div 元素
09         var tooltip = "<div id='tooltip'><img src='"+ this.href +'"
alt='产品预览图' />"+imgTitle+"</div>";
10         $("body").append(tooltip);           //把它追加到文档中
11         $("#tooltip")
12             .css({
13                 "top": (e.pageY+y) + "px",
14                 "left": (e.pageX+x) + "px"
15             }).show("fast");           //设置 x 坐标和 y 坐标，并且显示
16     }).mouseout(function(){
17         this.title = this.myTitle;
18         $("#tooltip").remove();           //移除
19     }).mousemove(function(e){
20         $("#tooltip")
21             .css({
22                 "top": (e.pageY+y) + "px",
23                 "left": (e.pageX+x) + "px"
24             });
25     });
26 })

```

在上述代码中，第 4~15 行设置鼠标滑入图片时的处理方法，其中第 9 行创建一个包含大图片的提示框（<div>标签元素对象），第 10 行将所创建的提示框对象追加到文档中，剩下的代码主要用来设置 x 和 y 坐标，使得提示框显示在鼠标位置的旁边。第 16~18 行设置鼠标滑出图片时的处理方法，即主要是移除提示框。第 19~25 行设置鼠标在图片上移动时的处理方法，即通过 css() 方法设置提示效果的坐标，以达到提示效果跟随鼠标一起移动的效果。

在浏览器中运行页面，效果如图 3.8 所示。当鼠标滑过小图片时，就会快速出现图片的预览提示效果，如图 3.9 所示。当鼠标离开小图片时，图片预览提示效果就会消失。



图 3.8 浏览页面



图 3.9 鼠标滑入图片时效果

3.8 常见问题

3.8.1 tagName 和 attribute 的区别

在第 3.1 节曾经讲到，jQuery 中使用 `attr` 方法来获取或设置元素的属性，而这里的属性指的是 `attribute`，不是 `tagName`，`tagName` 是标签的名称，不是标签的属性。

例如:

```
<a href="http://www.xxx.com/contactus" id="contactus" title="联系方式">
```

其中, a 就是 tagName, href 和 id 则是标签 a 的 attribute。

3.8.2 attr 方法和 prop 方法都用于获取元素的属性吗

本章讲解了使用 attr 方法来获取或修改元素的属性, 但是针对 Boolean 值的元素, 在获取属性时, 有一些区别, 建议使用 prop 方法, 尤其是没有为单元按钮或复选框设置初始值的情况下。这里做一个实验, 创建一个 check_prop.html 文件。代码如下:

```
01 <html>
02 <head>
03   <title>prop 示例</title>
04   <script type="text/javascript" src="../jquery-3.3.1.js"></script>
05   <script type="text/javascript">
06     function check()
07     {
08       var s=$("#check1").attr("checked");
09       //var s=$("#check1").prop("checked");
10       alert(s);
11     }
12   </script>
13 </head>
14
15 <body>
16   <input id="check1" type="checkbox" onclick="check()">来选我吧
17 <p></p>
18 </body>
19 </html>
```

在第 16 行我们没有为 checkbox 设置初始值。分别用第 8 行的 attr 方法和第 9 行的 prop 方法来调用复选框的 checked 属性值, 会得出不一样的结果。读者可以亲自测试一下。