第5章 jQuery

5.1 jQuery 基础

jQuery 基础

5.1.1 jQuery 概述

jQuery 是一个快速、简洁的 JavaScript 函数库。jQuery 设计宗旨是 Write Less，Do More（写更少的代码做更多的事情）。jQuery 提供一种简便的 JavaScript 设计模式，在 jQuery 中优化了 HTML 文档操作、事件处理、动画设计和 AJAX 交互。jQuery 兼容各种主流浏览器，如 IE 6.0 以上、FF 1.5 以上、Safari 2.0 以上、Opera 9.0 以上版本的浏览器。

5.1.2 jQuery 功能

jQuery 的功能如下：

（1）查询 HTML 元素，修改 HTML 元素的属性和样式。

（2）动态生成网页元素，并插入到原来的布局中，读取和改变元素的内容、属性值以及样式。

（3）jQuery 动画特效。使用 jQuery 可以为网页元素添加显示、隐藏、上下滑动等动画效果。

（4）与 AJAX 进行交互，实现提交数据局部刷新网页。AJAX 是异步的 JavaScript 和 XML 的简称，可以开发出无刷新的网页。开发服务器端网页时，需要多次与服务器通信，不使用 AJAX 时数据更新需要重新刷新网页，而通过 AJAX 可以对页面进行局部刷新，减少整页刷新次数。

5.1.3 jQuery 下载

任何一款网页编辑工具都可以用来编辑和调试 jQuery 程序，如 HBuilder、SublimeText。

如果编写 jQuery 脚本，需要在官网下载 jQuery 脚本库。jQuery 的下载地址是 http://jquery.com/download/。最新的 jQuery 版本是 3.3.1，每个版本对应三种脚本库，compressed、uncompressed 和 source map。compressed 是压缩版，文件较小，适合项目开发，但是不方便调试。uncompressed 是没有经过压缩处理的版本，文件较大，但调试方便。

source map 是一个信息文件，里面存储信息转换后的代码的位置所对应的转换前的位置，方便进行 JavaScript 还原性调试。通过 source map，调试工具将直接显示原始代码，而不是转换后的代码。有关 source map，读者可以参考网址 http://www.ruanyifeng.com/blog/2013/01/javascript_source_map.html。

本书使用最新的版本 jQuery 3.3.1，在使用时需要将其复制到站点根目录下。

5.1.4 第一个 jQuery 程序

下面通过一个简单的 jQuery 程序，方便读者了解 jQuery 编程特点。

【例 5.1】 jQuery 程序示例。

```
1  <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5          $(document).ready(function(){
6                        alert("hello world!");})
7    </script>
8  </head>
9   <body>
10  </body>
11 </html>
```

当页面载入之后，弹出 hello world 消息框。页面显示效果如图 5.1 所示。

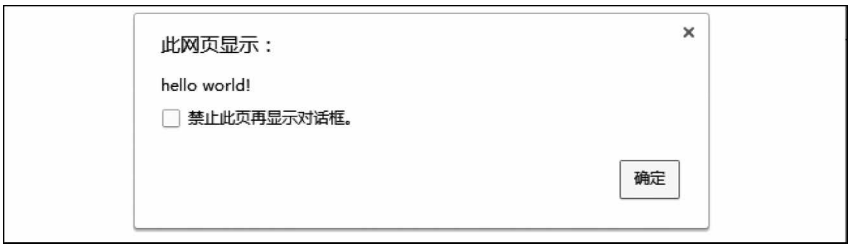

图 5.1 显示截图

说明：第 5 行代码中 $()表示 jQuery 对象，用来引用网页中指定的元素对象。如引用 p 对象的写法 $("p")，在 JavaScript 中的写法是 document.getElementById("p")。$(document)表示引用 HTML 文档的 document 对象。$(document).ready(){}表示当 HTML 文档载入(ready)之后执行匿名函数，即页面中出现提示框——alert()方法。

5.2 jQuery 选择器

通过 jQuery 可以对 HTML 元素进行动态管理，在操作网页元素之前需要通过 jQuery 选择器引用元素。在 jQuery 中，有基础选择器和层次选择器。

5.2.1 基础选择器


基础选择器

1. 标签选择器

标签选择器是通过 HTML 标签名称引用网页元素，如通过 $("table") 可以选取网页中的 table 元素。

语法：

```
$("标签名");
```

2. 类选择器

类选择器是根据网页元素类名来引用网页元素。

语法：

```
$(".类名");
```

例如，在网页中定义了一个名为 title 的 CSS 类，在 p 标记中应用了<p class="title">，如果要引用该对象，可以通过类选择器 $(".title")表示。

3. id 选择器

id 选择器是通过网页元素 id 号选择对应的 HTML 元素。

语法：

```
$("#id 号");
```

【例 5.2】 通过 id 号选择 HTML 元素实例。

```
1  <html><head>
2      <meta charset="utf-8">
3      <script type="text/javascript" src="jquery-3.3.1.js"></script>
4      <script type="text/javascript">
5          $(document).ready(function(){
6                      alert($("#mybox").html());     })
7  </script>
8   </head>
9   <body>
10     <div id="mybox">     aaaa      </div>
11   </body></html>
```

浏览器中运行效果如图 5.2 所示。

网页中定义了一个 id 为 mybox 的 div，当网页载入后，将 div 中的网页内容出现在弹出的对话框中。html()方法表示获取被选中元素的内容。

4. 选择所有元素

在 jQuery 中，使用通配符“*”表示引用所有 HTML 元素。

语法：

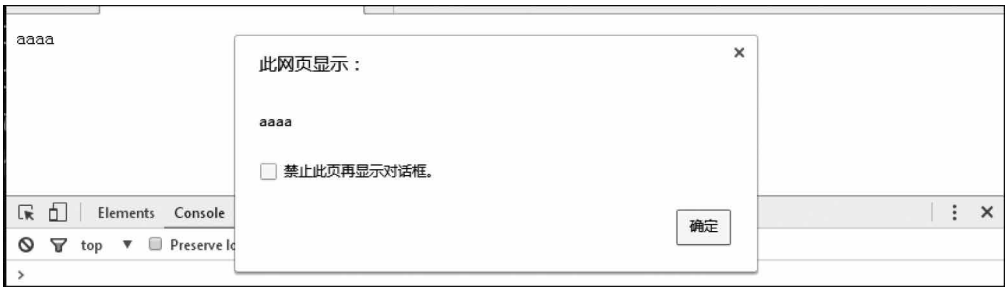

图 5.2　通过 id 号引用元素截图

```
$("*");
```

5. 同时选择多个元素

如果同时对多个 HTML 元素进行相同的操作，可以通过逗号运算符一次性引用多个元素。

语法：

```
$("选择器 1,选择器 2,选择器 3");
```

【例 5.3】 同时引用多个 HTML 元素实例。

```
1  <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5        $(document).ready(function(){
6                  $("#mybox,p").text("html");    })
7    </script></head>
8   <body>
9    <div id="mybox">    aaaa    </div>
10   <p>page</p>
11  </body></html>
```

网页中有两个元素，分别是 id 为"mybox"的 div 元素和 p 元素，通过 $("#mybox,p") 将这两个元素同时引用，通过 text() 方法设置这两个元素的值为"html"。运行效果如图 5.3 所示。

图 5.3　引用多个元素网页的运行结果

5.2.2　层次选择器

HTML 文件中的元素是有一定的层次关系的，如处于根部的是 HTML 元素，其下有头

部元素和主体元素，它们分别有自己的子元素，如 head 中包含 title 和 meta 等元素。

层次选择器

1. 祖先-后代选择器

利用祖先-后代选择器可以选取某个祖先元素的指定后代元素，如利用 $("div p ")可以选择 div 中所有的 p 元素。

语法：

```
$("祖先选择器 后代选择器");
```

【例 5.4】 祖先-后代选择器的应用。

```
1  <html><head>
2      <meta charset="utf-8">
3      <script type="text/javascript" src="jquery-3.3.1.js"></script>
4      <script type="text/javascript">
5         $(document).ready(function(){
6                   $("#mybox div").css("font-size","20px");})
7      </script></head>
8    <body>
9      <div id="mybox">
10           <div><span>hello world</span> everyone! </div>
11     </div>
12 </body></html>
```

2. 父>子选择器

使用父>子选择器可以选取指定父元素中的某些子元素，如通过 $("div>p")可以选取 div 中直接子元素<p>元素。

语法：

```
$("父选择器>子选择器");
```

【例 5.5】 父>子选择器的应用。

```
1  <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5        $(document).ready(function(){
6                $("#mybox>div").css("font-size","20px");})
7    </script></head>
8  <body>
9    <div id="mybox">
10     <div class="s1">
11         <span class="s1">hello world</span>
12         everyone!
13     </div>
```

```
14    </div>
15  </body></html>
```


$("#mybox>div")表示引用 id 号为"mybox"的网页元素的直接< div >子元素，也就是字符串"everyone."，通过 css("font-size","20px")方法，为其设置文字大小为 20px。对于< span >元素，由于不是"mybox"< div >的直接子元素，所以大小不受影响。

小贴士：祖先-后代选择器与父>子选择器的区别是什么？

两者的区别是，祖先-后代选择器能表示祖先元素中所有的某类后代元素；包括子元素和孙子元素；而父>子选择器只能表示父元素的直接子元素。

3. 前＋后选择器

前＋后选择器可以选择指定的前面元素后面的元素。

语法：

```
$("前元素 + 后元素");
```

【例 5.6】 前＋后元素选择器应用。

```
1  <html><head>
2        <meta charset = "utf - 8">
3        <script type = "text/javascript" src = "jquery - 3.3.1.js"></script>
4        <script type = "text/javascript">
5              $(document).ready(function(){
6                    $("#username + #mbutton").css("font - family","微软雅黑");})
7        </script></head>
8    <body>
9        <input type = "text" name = "username" id = "username" />
10       <input type = "button" name = "mybutton" id = "mbutton" value = "按钮" />
11   </body></html>
```

$("#username＋#mbutton")表示引用 id 号为"username"的元素后边的 id 号为"mbutton"的元素，通过 css()方法设置字体为微软雅黑。

4. 前～兄弟选择器

使用前～兄弟选择器可以引用前面元素后面的兄弟元素。

语法：

```
$("前元素～兄弟元素");
```

【例 5.7】 前～兄弟选择器应用。

```
1  <html><head>
2      <meta charset = "utf - 8">
3      <title>
4          jquery 演练 6
```

```
    </title>
    <script type="text/javascript" src="jquery-3.3.1.js"></script>
    <script type="text/javascript">
        $(document).ready(function(){
                        $("p ~ div").css("font-size","20px");})
    </script></head>
<body>
    <p>基本选择器</p>
    <div>
        <span>hello world</span>
    </div>
    <div>hello world!</div>
</body>
</html>
```

$("p ~ div")表示段落标记后续的 标记，由于标记是嵌套在 标记中，因此标记不在引用范围。css("font-size","20px")方法是对这些符合要求的元素设置 css 文字大小为 20 像素。

5.2.3 过滤器

过滤器

利用过滤器可以对选中的数据进行过滤。过滤器的使用方法是$("选择器：s 过滤器")。

1. 基本过滤器

：first 匹配找到的第一个元素。

例如，$("input：first")选择网页中的第一个<input>元素。

：last 匹配找到的最后一个元素。

例如，$("input：last")选择网页中的最后一个<input>元素。

：not 过滤与给定选择器匹配的元素。

例如，$("input：not(.one)")表示引用<input>中 class 不是 one 的元素。

：even 匹配所有索引值是偶数的元素。

例如，$("input：even")表示引用<input>元素中索引值为偶数的元素，索引值从 0 开始，even 表示奇数次出现的<input>元素，如网页中第 1 个、3 个、5 个 input 元素。

：odd 匹配所有索引值是奇数的元素。

例如，$("input：odd")表示引用<input>元素中索引值为奇数的元素，索引值从 0 开始表示，odd 表示偶数次出现的<input>元素。

：eq(index)匹配索引值是 index 的元素。

例如，$("input：eq(0)")匹配<input>中索引号为 0 的元素。

：gt(index)匹配索引值大于 index 的元素。

例如，$("input：gt(0)")匹配<input>元素索引值大于 0 的元素。

：lt(index)匹配索引值小于 index 的元素。

例如，$("input：lt(2)")匹配<input>元素中索引值小于 2 的元素。

：header 匹配 h1～h6 的元素。

语法：

```
$(": header")
```


【例 5.8】 基本过滤器应用实例。

```
1 <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5       $(document).ready(function(){
6         $("input:first").css("background-color","blue");
7         $("input:last").css("background-color","red");
8         $("input:even").css("background-color","blue");
9         $("input:gt(0)").css("background-color","blue");
10        })
11  </script></head>
12 <body>
13  <form>
14        <input type="text" name="uname">
15        <input type="password" name="pwd">
16        <input type="email" name="myemail">
17        <input type="submit" value="提交">
18  </form></body></html>
```

第 6 行代码设置网页中的第一个< input >元素的 CSS 属性，第 7 行代码设置网页中的最后一个< input >元素的 CSS 属性，第 8 行代码设置网页中的索引号为奇数的< input >元素 CSS 属性，第 9 行代码设置网页中的索引号为大于 0 的< input >元素 CSS 值。

说明：在书写代码时，为了能更好地查看代码运行效果，建议每次只应用一条语句，将其余过滤器代码注释执行。

例如：

```
$("input:first").css("background-color","blue");
//$("input:last").css("background-color","red");
//$("input:even").css("background-color","blue");
//$("input:gt(0)").css("background-color","blue");
```

2. 内容过滤器

内容过滤器是根据网页元素的内容对元素进行过滤。

：contains()表示包含指定内容的元素。例如，$("div：contains(hello world!)")表示匹配 div 中包含"hello world!"的字符串。

：empty 表示不包含子元素或内容为空的元素。例如，$("div：empty ()")表示匹配不含子元素或文本的 div 空元素列表。

：has()表示匹配指定子元素的元素。例如，$("div：has (span)")表示匹配含有< span >元素的< div >元素。

：parent()匹配包含子元素或内容的元素。例如，$("div：parent (span)")表示匹配

至少包含一个< span >元素的< div >元素。

【例 5.9】 内容过滤器应用实例。

```
1 < html >< head >
2        < meta charset = "utf - 8">
3        < script type = "text/javascript" src = "jquery - 3.3.1.js"></script >
4        < script type = "text/javascript">
5           $ (document).ready(function(){
6              $ ("div:contains(jquery)").css("font - size","20px");
7            $ ("div:empty()").css("border","1px solid black");
8            })
9     </script ></head >
10 < body >
11 < div ></div >
12   < div >       jquery   </div >
13 </body ></html >
```

第 6 行代码匹配 div 中包含 jquery 的字符串，设置包含“jquery”字符串的< div >中文字大小为 20 像素。第 7 行代码匹配不含子元素或文本的< div >，满足条件的< div >边框设置为 1 像素实线。

3. 可见性过滤器

jQuery 中有两个可见性过滤器，分别是 hidden 和 visible 过滤器。

例如，$ ("input：hidden")匹配所有隐藏域元素。在 HTML 中，当一个< input >元素的 type 属性值为 hidden 时，表示隐藏域元素，写法如下：

```
< input type = "hidden"/>
```

：hidden 表示匹配所有不可见元素

：visible 用来匹配所有可见元素。

【例 5.10】 可见性过滤器应用实例。

```
1   < html >< head >
2       < meta charset = "utf - 8">
3       < script type = "text/javascript" src = "jquery - 3.3.1.js"></script >
4       < script type = "text/javascript">
5            $ (document).ready(function(){
6                alert( $ ("input:hidden").val());
7            })
8       </script ></head >
9   < body >
10      < form >< input type = "hidden" name = "myhidden" value = "userzhangsan"/></form >
11  </body ></html >
```

第 6 行代码通过对话框输出隐藏域中的文本信息。

4. 属性过滤器

属性过滤器是根据元素的属性或属性值过滤。

[属性名]匹配包含指定属性的元素。例如，$("input[name]")表示匹配包含 name 属性的 input 元素。

[属性名＝值]匹配指定属性值的元素。例如，$("input[name＝username]")表示匹配 name 属性值为 username 的< input >元素。

[属性名!＝值]匹配属性不等于指定属性值的元素。例如，$("input[name!＝username]")表示匹配 name 属性值不为 username 的< input >元素。

【例 5.11】 属性过滤器应用实例。

```
1  <html><head>
2      <meta charset="utf-8">
3      <script type="text/javascript" src="jquery-3.3.1.js"></script>
4      <script type="text/javascript">
5       $(document).ready(function(){
6           $('input[name]').val("默认值");
7           $('input[id=email]').val("abc@qq.com");
8       })
9    </script></head>
10  <body>
11    <form>
12       <input type="text" name="uname">
13       <input type="text" name="unumber">
14       <input type="email" id="email">
15    </form></body></html>
```

第 6 行代码设置包含 name 属性的< input >元素的值为“默认值”。第 7 行设置 id 值为 email 的< input >元素的默认值为“abc@qq.com”。

5.3 设置 HTML 属性及 CSS 样式

通过 jQuery 可以控制网页中的 DOM(Document Object Model，文档对象模型)元素。jQuery 提供一系列与 DOM 相关的方法，使访问和操作元素和属性变得容易。

5.3.1 获得及设置 HTML 元素内容

获得及设置< html >元素内容

jQuery 通过 text()、html()以及 val()方法可以获取或设置 DOM 元素的文本内容。

- text()：获取或设置所选元素的文本内容。
- html()：设置或获取所选元素的内容，其中包括< HTML >标记。
- val()：设置或获取表单字段的值，用在表单元素中。

【例 5.12】 获取 HTML 元素应用实例。

```
1  <html><head>
2     <meta charset="utf-8">
3     <script type="text/javascript" src="jquery-3.3.1.js"></script>
```

```
4    <script type="text/javascript">
5        $(document).ready(function(){
6                    $("#show").click(function(){
7                    alert("Message: " + $("#mes").text());
8                    alert("Message: " + $("#mes").html());
9             });
10       $("#set").click(function(){
11                $("#mes").html("<b>最新设置的文本</b>");
12           })
13     })
14   </script></head>
15 <body>
16      <p id="mes">这是段落中的<b>粗体</b>文本.</p>
17      <input type="button" id="show" value="显示文本">
18      <input type="button" id="set" value="设置文本">
19 </body></html>
```

第 7 行代码获取 id 为 mes 的标记中的文本，第 8 行代码获取 id 为 mes 标记中的内容，包括< HTML >标记。第 11 行代码设置 id 为“mes”标记中的 HTML 内容。

网页在浏览器中运行时，当单击“显示文本”按钮先后两次弹出对话框，分别显示段落标记< p >中的文本内容和包含标记的文本信息，单击“设置文本”按钮，通过 id 选择器 $("#mes")选取 mes 元素，并设置段落中的 HTML 内容，如图 5.4 所示。

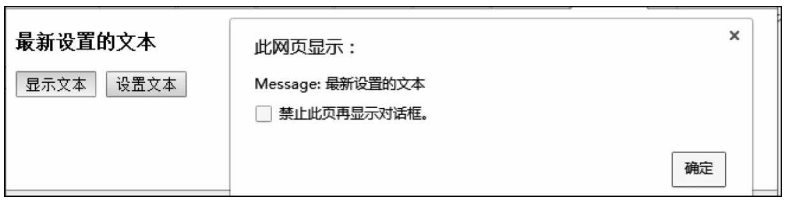

图 5.4　获取 HTML 元素截图

5.3.2　获取及设置获得 HTML 元素属性

在 jQuery 中，通过 attr()方法可以获取或设置所选元素的属性值。

【例 5.13】 attr()方法应用。

```
1 <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5        $(document).ready(function(){
6            $("#but").click(function(){
7                $("#uname").attr("value","李四"); })
8            })
9  </script></head>
10 <body>
11    <input type="text" id="uname" value="张三">
```

```
12     < input type = "button" value = "改变属性值" id = "but">
13 </body ></html >
```


第 5 行代码表示当文档载入之后执行匿名函数。第 6 行代码为按钮绑定单击事件，当单击按钮之后执行匿名函数。第 7 行代码设置 id 值为 uname 文本框的 value 值是李四。

5.3.3 添加新元素/内容

添加新元素/内容

jQuery 中使用 append()、prepend()、after()、before()等方法向网页中添加新元素或内容。

- 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。

【例 5.14】 添加新元素实例。

```
1  < html >< head >
2     < meta charset = "utf - 8">
3          < script type = "text/javascript" src = "jquery - 3.3.1.js"></script >
4          < script type = "text/javascript">
5               $ (document).ready(function(){
6                    $ ("＃but").click(function(){
7                                   $ ("＃oldm").append("< br >< b >增加后的文本</b >");})
8                         })
9        </script ></head >
10  < body >
11       < p id = "oldm">原始文本</p >
12       < input type = "button" value = "增加文本" id = "but">
13  </body ></html >
```

第 7 行代码向段落 p 中现有文本之后插入新的文本“增加后的文本”。

浏览器中运行效果如图 5.5 所示，当单击“增加文本”按钮之后，显示如图 5.6 所示的效果。

图 5.5 增加文本前的截图

图 5.6 单击“增加文本”按钮之后网页截图

程序中通过 append()方法向指定的元素中追加内容，append(content)方法中 content 参数表示被追加的数据，可以是字符、< HTML >标记，还可以是一个返回字符串内容的函数。

jQuery 中通过 appendTo 方法也可以创建节点：

语法：

```
$(content).appendTo(selector);
```

其中，参数 content 表示需要插入的内容，参数 selector 表示被选的元素，即把 content 内容插入 selector 元素内，默认插入到尾部。

【例 5.15】 创建节点实例。

```
1  <html><head>
2     <meta charset="utf-8">
3     <script type="text/javascript" src="jquery-3.3.1.js"></script>
4     <script type="text/javascript">
5          $(document).ready(function(){
6            $("<input type='text'/>").appendTo($("form"));
7          })
8     </script></head>
9   <body>
10    <form><input type="button" value="增加文本" id="but"></form>
11  </body></html>
```

第 6 行代码用来创建一个 input 节点，并将其动态添加到 form 尾部。

5.3.4 删除元素/内容

jQuery 使用 remove()和 empty()方法删除网页元素和网页内容。

- remove()：删除被选元素及其子元素。
- empty()：从被选元素中删除子元素。

【例 5.16】 remove()删除元素实例。

```
1  <html><head>
2     <meta charset="utf-8">
3     <script type="text/javascript" src="jquery-3.3.1.js"></script>
4     <script type="text/javascript">
5          $(document).ready(function(){
6              $("#but").click(function(){
7                  $("p").remove();});
8          })
9     </script></head>
10 <body>
11       <p style="height:200px;width:200px;border:1px solid black;">
12           <span>段落中文字</span>
```

```
13        </p>
14        <form><input type="button" value="删除元素" id="but"></form>
15 </body></html>
```

第 7 行代码用来删除<p>标记及其中的标记。

【例 5.17】 empty()删除元素实例。

```
1  <html><head>
2     <meta charset="utf-8">
3     <script type="text/javascript" src="jquery-3.3.1.js"></script>
4     <script type="text/javascript">
5          $(document).ready(function(){
6              $("#but").click(function(){
7                  $("p").empty();});
8          })
9     </script></head>
10 <body>
11     <p style="height:200px;width:200px;border:1px solid black;">
12         <span>段落中文字</span>
13     </p>
14    <form><input type="button" value="删除元素" id="but"></form>
15 </body></html>
```

第 7 行代码用来删除<p>标记中嵌套的标记(标记),保留<p>标记。

5.3.5 操作 CSS

操作 CSS

jQuery 通过以下 4 种方法对 CSS 样式进行操作：

- addClass()：向被选元素添加一个或多个 CSS 类。
- removeClass()：从被选元素删除一个或多个类。
- toggleClass()：对被选元素进行添加、删除类的切换操作。
- css()：设置或返回样式属性。

【例 5.18】 addClass()方法和 toggleClass()方法应用。

```
1  <html><head>
2     <meta charset="utf-8">
3     <script type="text/javascript" src="jquery-3.3.1.js"></script>
4     <script type="text/javascript">
5          $(document).ready(function(){
6              $("#but").click(function(){
7                  $("#p1").addClass("s1");
8                  $("#p2").addClass("s2");
9              });
10             $("#switch").click(function(){
11                 $("#p1").toggleClass("s1");
12                 $("#p2").toggleClass("s2");});
13         })
```

```
14    </script>
15    <style type="text/css">
16      .s1{
17          color:red; }
18      .s2{
19          color:blue; }
20     </style>
21  </head>
22  <body>
23        <p id="p1">第一段段落文字</p>
24        <p id="p2">第二段落文字</p>
25        <form>
26            <input type="button" value="为元素添加类" id="but">
27            <input type="button" value="切换 CSS 类" id="switch">
28        </form></body></html>
```

第 6 行代码表示当单击“为元素添加类”按钮时执行匿名函数。第 7 和第 8 行代码为 p1、p2 段落动态添加 s1、s2 样式。第 10 行代码定义单击“切换 CSS 类”按钮的事件。第 11 和第 12 行代码为 p1、p2 段落添加/删除 s1、s2 样式，并动态地在添加和删除状态之间切换。

浏览器中运行效果，如图 5.7 所示。在页面中单击“为元素添加类”按钮，会将内嵌的 CSS 样式中定义的 s1 选择器样式应用在 p1 段落文字上，s2 样式应用在 p2 文字上。单击“切换 CSS 类”按钮，会在添加和删除样式之间切换操作。

图 5.7　CSS 对网页样式设置截图

5.3.6　css()方法

css()方法可以设置、获取被选元素的一个或多个样式属性。

【例 5.19】 css()方法的应用实例。

```
1 <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5      $(document).ready(function(){
6            $("#b1").click(function(){
7            alert("第一段文字颜色是" + $("#p1").css("color"));
8            });
9            $("#b2").click(function(){
10           alert("第一段文字颜色是" + $("#p2").css("color"));
11           });    })
```

```
12   </script>
13   <style type="text/css">
14     .s1{color:red; }
15     .s2{color:blue;}
16   </style>
17 </head><body>
18   <p id="p1" class="s1">第一段段落文字</p>
19   <p id="p2" class="s2">第二段落文字</p>
20   <form>
21       <input type="button" value="显示第一段文字颜色" id="b1">
22       <input type="button" value="显示第二段文字颜色" id="b2">
23   </form></body></html>
```

第 6 行代码定义 b1 按钮单击后执行的事件。第 7 行代码定义在消息框中显示 p1 段落的 color 属性。第 9 行定义 b2 按钮的单击事件。第 10 行代码用来在消息框中显示 p2 段落的 color 属性。

浏览器中运行效果如图 5.8 所示。

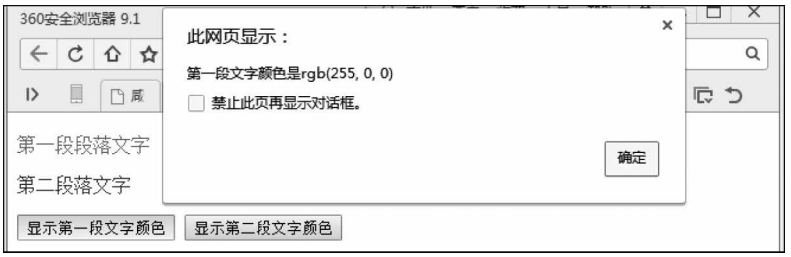

图 5.8　CSS 方法应用截图

5.3.7　处理 DOM 元素尺寸

通过 jQuery，可以轻松处理元素和浏览器窗口的宽度和高度。jQuery 提供设置尺寸的方法有 6 种。

(1) width()：设置或返回元素的宽度，不包括内边距、边框或外边距。

(2) height()：设置或返回元素的高度，不包括内边距、边框或外边距。

(3) innerWidth()：返回元素的宽度，包括内边距。

(4) innerHeight()：返回元素的高度，包括内边距。

(5) outerWidth()：返回元素的宽度，包括内边距和边框。

(6) outerHeight()：返回元素的高度，包括内边距和边框。

【例 5.20】 尺寸设置应用实例。

```
1  <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
5      $(document).ready(function(){
```

```
6            $("#b1").click(function(){
7                var txt ="宽度是:" + $("div").width() + "高度是:" + $("div").height();
8                $("div").html(txt); });
9            $("#b2").click(function(){
10               var txt ="包含内边距和边框的宽度是:" + $("div").outerWidth()";
11               $("div").html(txt); });
12           })
13   </script>
14   <style type="text/css">
15       div{
16           width:200px;
17       height: 150px;
18           border: 1px solid black;
19           }
20   </style></head>
21   <body>
22       <div></div>
23       <form>  <input type="button" value="显示 div 的大小" id="b1"/> <input type=
     "button"value="显示 div 的大小包括内边距和边框"id="b2"/> </form>
24   </body></html>
```

第 7 行代码获取 元素的宽度和高度。第 8 行代码将宽度高度等信息写到 中。第 10 行代码获取 元素的宽度,包括内边距和边框。

浏览器中运行效果如图 5.9 所示。

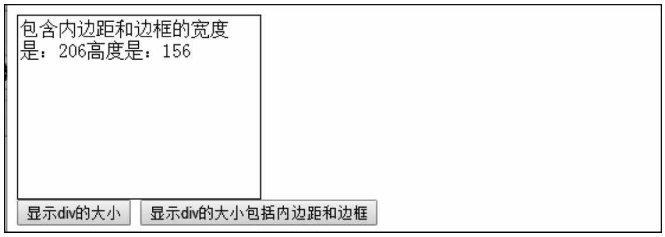

图 5.9　获取 DOM 元素大小图

5.4　常见事件方法

常见事件方法

事件方法将会触发元素的某个事件,执行一定的操作。jQuery 常见的事件方法如下:

(1) ready():定义 HTML 文档就绪事件。

语法:

```
$(document).ready(function(){…});
```

(2) load():当 HTML 元素载入时触发的事件。

语法:

```
$("selector").load(function(){…});
```


例如，$("img").load(function(){…})表示图片载入之后触发函数的执行。

(3) click()：将函数绑定到指定元素的单击事件。

语法：

```
$("selector").click(function(){…});
```

例如，$("input[type＝button]").click(function(){…})表示当单击 button 时触发 function 执行。

(4) dblclick()：定义当双击指定元素时触发的事件。

语法：

```
$(" selector ").dblclick(function(){…});
```

例如：$("input[type＝button]").dblclick(function(){…})表示当双击 button 时触发 function 事件。

(5) bind()：向匹配元素附加一个或更多事件处理器。

1. 添加事件处理器

语法：

```
$("selector").bind("click",function(){…});
```

例如，$("input[type＝button]").bind("click",function(){…})为按钮 button 绑定单击事件，当单击时 function 函数执行。

2. 添加多个事件处理器

语法：

```
$("selector").bind({
    click:function(){…},
    mouseover:function(){…},
    mouseout:function(){…}  });
```

例如：

```
$("img").bind({
    mouseover:function(){alert("mouseover ")},
    mouseout:function(){alert("mouseout ")} });
```

表示在 img(图像元素)上绑定了两个事件，分别是鼠标 over(悬停)事件和鼠标 out(移出)事件，当事件触发时执行相应的 function。

(1) blur()：指定元素的失去焦点事件。

语法：

```
$("selector").blur(function(){…});
```

例如，$("input").blur(function(){…})表示当 input 元素失去焦点时触发 function 函数执行。

(2) focus()：指定元素获取焦点事件发生时执行某个函数。

语法：

```
$("input").focus(function(){…});
```

例如，$("input[type=text]").focus(function(){…})表示当文本框获得焦点时触发 function 函数执行。

(3) change()：指定元素的值发生变化时执行某个函数。

语法：

```
$(selector).change(function(){…});
```

例如：

```
$("input[type=text]").change(function(){
$(this).css("background-color","blue"); });
```

表示当改变文本框输入值时修改文本框背景色。

(4) keydown()：当在某个元素中按下按键时执行的操作。

语法：

```
$(selector).keydown(function(){…});
```

例如：

```
$("input").keydown(function(){
$("input").css("background-color","yellow"); });
```

表示当在< input >元素中按下键盘按键时将< input >元素的背景色设置为黄色。

(5) keypress()：在某个元素上按下并松开键盘键时执行的函数，其中包含 keydown 和 keyup 两个事件。

语法：

```
$(selector).keypress (function(){…});
```

例如：

```
$("input").keypress(function(){…});
```

(6) keyup()：在某个元素上松开键盘键时执行的函数。

语法：

```
$(selector).keyup(function(){…});
```

例如，

```
$("input").keyup(function(){…});
```

(7) mousedown()：在某个元素上按下鼠标时执行的函数。

语法：

```
$(selector).mousedown(function(){…});
```

例如：

```
$("div").mousedown(function(){
$("div").css("color","yellow");});
```

表示在 元素上按下鼠标时改变 元素文字颜色。

(8) mouseout()：定义当鼠标从指定元素上移开时触发的函数。

语法：

```
$(selector).mouseout(function(){…});
```

例如：

```
$("div").mouseout(function(){
$("div").css("color","yellow");});
```

表示当从 元素上移开鼠标时改变 元素文字颜色。

(9) mouseover()：定义当鼠标悬停在指定元素上时触发的函数。

语法：

```
$(selector).mouseover(function(){…});
```

例如：

```
$("div").mouseover(function(){
$("div").css("color","yellow");});
```

表示当鼠标悬停在 上时 的文字颜色变为黄色。

(10) event.pageX：返回相对于文档左边缘的鼠标位置。

语法：

```
event.pageX;
```

例如：

```
$(document).mousemove(function(e){
alert("X: " + e.pageX + ", Y: " + e.pageY);});
```

表示当鼠标在网页中移动时(mousemove)，在对话框中显示鼠标所在位置的 X 坐标和 Y 坐标值。

(11) event.pageY：返回相对于文档上边缘的鼠标位置。

语法：

```
event.pageY;
```

(12) event.target：触发该事件的 DOM 元素。

语法：

```
event.target;
```

例如：

```
$("p h1").click(function(event){
alert(event.target.nodeName + " element.");});
```

当单击网页中的 p(段落)或 h1(标题 1)文字时，弹出消息框，提示触发 click 事件的事件源。

(13) event.type：描述事件的类型。

语法：

```
event.type;
```

例如：

```
$("p h1").click(function(event){
alert(event.type );});
```

当单击网页中的 p(段落)或 h1(标题 1)文字时，弹出消息框，提示触发的事件类型(click)。

【例 5.21】 常见事件方法应用实例。

```
1 <html><head>
2    <meta charset="utf-8">
3    <script type="text/javascript" src="jquery-3.3.1.js"></script>
4    <script type="text/javascript">
```

```
5        $(document).ready(function(){
6              $("img").bind({
7                  "mouseover":function(){
8                              $("img").attr("src","images/viewo.jpg");},
9                  "mouseout":function(){
10                             $("img").attr("src","images/view.jpg");}
12                 });
13         })
14    </script></head><body>
15     <img src="images/view.jpg">
16    </body></html>
```

第 6 行代码为<img>图像元素绑定事件；第 8 行代码定义鼠标悬停在图像上，图片切换为 viewo.jpg；第 10 行代码定义当鼠标移出图像区域切换为原始图像。

浏览器中当鼠标进入图像区域，图片变换；鼠标移出图像区域，图片恢复。

5.5 jQuery＋AJAX

jQuery＋AJAX

AJAX 表示异步 JavaScript 和 XML(Asynchronous JavaScript and XML)。在不重载整个网页的情况下，AJAX 通过后台加载数据，并在网页上显示数据。AJAX 技术使用 HTTP Get 和 HTTP Post 方法从远程服务器上请求文本、HTML、XML 或 JSON 数据，同时能够把这些外部数据直接载入网页。

jQuery 提供 ajax()方法来通过 HTTP 请求加载远程数据。

语法：

```
$.ajax(url: "url",async:false, data, success);
```

说明：

- url 表示发送请求的地址。
- async 表示是否为异步请求，默认值为 true，即异步请求；如果设置为 false 则表示同步请求。
- data 表示发送到服务器的数据，要求为 Object 或 String 类型的参数，若是对象，必须为 key/value 格式，如{name：“zhangsan”，age：21}发送给后台被转换为 & name ＝ zhangsan & age ＝21。
- success 为匿名函数，表示请求成功后调用的回调函数。它有两个可以缺省的参数，data 表示由服务器返回根据 dataType 参数处理后的数据(类型可能是 xmlDoc、jsonObj、html、text 等)，textStatus 是描述状态的字符串。

```
function(data, textStatus){
…       }
```

$.ajax()函数返回 XMLHttpRequest 对象。该对象具有 responseText 和 responseXML 属性，responseText 表示响应的字符串形式文本信息，responseXML 表示获得 XML 形式的响应数据。

【例 5.22】 ajax()方法加载文本文件。

```
1 <html><head>
2    <script type="text/javascript" src="jquery-3.3.1.js"></script>
3    <script type="text/javascript">
4        $(document).ready(function(){
5            $("#change").click(function(){
6                    htmlobj=$.ajax({url:"txt/demo.txt",async:false});
7                    $("#mdiv").html(htmlobj.responseText);
8                });
9        });
10 </script>  </head>
11 <body>
12       <div id="mdiv">
13           <h2>原始文本</h2>
14       </div>
15       <form onsubmit="return false">
16           <button id="change" type="button">改变内容</button>
17       </form></body></html>
```

程序中第 6 行代码用来调用 ajax()方法，发送同步请求，加载远程数据 demo.txt。第 7 行代码是在 div 中显示文本文件中的数据。

在浏览器中运行效果如图 5.10 所示。单击“改变内容”按钮之后，div 中的文本改变为“通过 AJAX 改变文本”。

图 5.10　ajax()方法加载文本文件截图

【例 5.23】 ajax()方法加载 json 文件。

创建一个 myjson.json 文件，其中输入如下数据：

```
{
    "school":"Xianyang Normal University",
    "address":"Shannxi"
}
```

json 文件的创建方法：新建一个文本文件，在保存时设置保存类型为所有文件，输入文件名，注意扩展名为 json。json 文件中的数据以 key:value 的形式表示。

HTML 文件代码如下：

```
1 < html >  < head >
2    < meta charset = "utf - 8">
3    < script type = "text/javascript" src = "jquery - 3.3.1.js"></script >
4    < script type = "text/javascript">
5        $ (document).ready(function(){
6            $ ("＃change").click(function(){
7                        htmlobj = $ .ajax({url:"myjson.json",async:false});
8                        var json = JSON.parse(htmlobj.responseText);
9                        $ ("＃mdiv").html(json.school + json.address);
10             });
11           });
12    </script ></head >
13    < body >
14        < div id = "mdiv">
15            < h2 >原始文本</h2 >
16        </div >
17        < form onsubmit = "return false">
18            < button id = "change" type = "button">加载 json 中数据</button >
19        </form >
20    </body ></html >
```

程序中第 6 行代码定义 change 按钮单击事件。第 7 行代码调用 ajax()方法，发送同步请求，加载远程数据 myjson.json。第 8 行代码通过 JSON.parse()方法解析 json 数据。第 9 行代码用来在 div 中显示返回 json 中 school 和 address 的值。

在 ajax()方法中可以增加 success 参数，表示请求成功后的回调函数。

核心代码如下：

```
$ .ajax({url:"txt/myjson.json",async:false,success:function(data){
        var json = JSON.parse(data);
        alert(json.school);         }});
```

当 json 数据返回之后，执行 success()回调函数，data 参数表示的是返回的 json 格式的数据。

本 章 小 结

jQuery 是一个轻量级的 JavaScript 库，可以帮助用户快速地创建脚本，而且 jQuery 是跨浏览器的。通过 jQuery 的 API 方法可以轻松地获取网页中的元素，同时设置网页元素的外观属性，更改网页中的内容，捕捉网页中的事件并且对事件进行响应，实现淡入淡出、擦除等的动画效果，与 AJAX 进行交互。由于具有这些特点，jQuery 的应用已经越来越广泛。

课 后 习 题

（1）选择表单中所有< input >元素的方法是_____。

（2）描述 jQuery 时，使用_____方法获取和设置 CSS 属性。

（3）在 jQuery 中，使用_____方法可以切换 HTML 元素的显示和隐藏状态。