第5章 jQuery基础

由于JavaScript本身存在两个缺点：一个是复杂的DOM操作，另一个是不一致的浏览器实现。因此，为了简化JavaScript开发工作，解决浏览器之间的兼容性问题，业界出现了许多的JavaScript库。JavaScript库中封装了很多预定义的对象和使用函数，能够帮助开发人员轻松搭建具有高难度交互功能的客户端页面，并且可以完美兼容各种浏览器。其中，jQuery在经历了若干次版本更新后，逐渐从各种JavaScript库中脱颖而出，成为Web开发人员的最佳选择。本章主要讲解jQuery的基本语法、如何使用jQuery操作DOM元素以及正则表达式的应用。

5.1 jQuery的作用

jQuery是一种轻量级的JavaScript库，它的设计主旨是“write less，do more”。在开发中jQuery的作用主要包括5个方面。

1. 访问和操作DOM元素

jQuery提供了一套方便、快捷的API来操作DOM元素，因此在开发中可以在很大程度上减少代码的编写，并且提高用户对网页的体验度。

2. 控制页面样式

通过引入jQuery，开发人员可以便捷地控制页面的CSS样式，并且可以很好地兼容各种浏览器。

3. 对页面事件的处理

通过jQuery的事件绑定机制可以使得页面在处理事件时能将表现层和功能开发分离。这样一来，开发人员更多地专注于程序的逻辑与功能，页面设计人员可以侧重于页面的优化与用户体验。

4. 方便地使用jQuery插件

引入jQuery可以使用大量的jQuery插件来完善页面的功能和效果，如jQuery UI插件、Form插件、Validate插件等。

5. 便捷地使用AJAX

实际开发中，利用AJAX异步读取服务器数据是常见的需求，使用原生的JavaScript来操作AJAX十分烦琐，而引入jQuery后，不仅完善了功能，还大大简化了代码的编写，通过其内部对象和函数，简单几行代码就可以实现复杂的功能。

5.2 开发环境搭建

搭建 jQuery 开发环境十分简单，只需简单的几个步骤。

1. 下载 jQuery

进入 jQuery 官网(https://jquery.com/)，单击页面右侧的 Download jQuery 进入下载页面。jQuery 库的类型有两种：开发版(未压缩)和发布版(压缩)，它们的对比如表 5-1 所示。

表 5-1 jQuery 版本区别

名　　称	说　　明
jQuery-版本号.js	完整无压缩版本，主要用于测试、学习和开发
jQuery-版本号.min.js	经过压缩后的版本，主要用于发布的产品和项目

本教程采用的版本为无压缩版，版本号为 jQuery-3.4.1.js。

2. 引入 jQuery

在实际开发中，只需把下载的 jQuery 库文件放到工程中的一个公共位置，然后在相应页面引用即可。例如，首先在工程 jQuery 中新建文件夹 js，并将 jQuery 的库文件放入该文件夹中，然后创建页面 5-1.html，如图 5-1 所示。

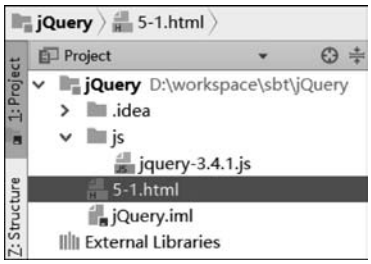

图 5-1 配置 jQuery

在 5-1.html 页面中便可通过如下代码引入 jQuery 库。

```
<script src = "js/jquery - 3.4.1.js"></script>
```

其中，src 属性引用时使用的是相对路径，在实际项目中，可根据需求调整 jQuery 库的路径。

3. 使用 jQuery

在页面中引入 jQuery 后，就可以在< script >标签中使用它了。

【例 5-1】 在 5-1.html 中加入代码，引入 jQuery。

```
<script type = "text/javascript">
    $(document).ready(function () {
       alert("Hello jQuery...")
    })
</script>
```

例 5-1 所示代码会在浏览器中弹出一个警告框，如图 5-2 所示。

页面 5-1.html 代码中的一条关键语句为 $(document).ready()，这条语句可分成 3 个部分：$()、document、ready()，在 jQuery 中分别称为工厂函数、DOM 对象和 jQuery 方法。

工厂函数 $()的作用是将 DOM 对象转化为 jQuery 类型的对象，从而使得该对象能调用相应的 jQuery 方法。例如，document 对象是一个 DOM 对象，将它作为参数传给工厂函数 $()转化为 jQuery 对象类型后，就能使用 jQuery 封装好的 ready()方法，该方法可以将

图 5-2　jQuery 的 Hello World

函数作为参数传入，表示在页面加载完成后才执行该函数，类似于 window 对象的 onload 功能。

除了可以给工厂函数传具体的 DOM 对象外，还可以给它传 jQuery 选择器来选择页面相应的元素。jQuery 选择器与 CSS 选择器类似，但又有所不同，我们将在 5.4 节中详细讲述它。

jQuery 对象提供了一系列的方法。其中，一类重要的方法就是事件处理方法，主要用来绑定 DOM 元素的事件和事件处理函数。一般情况下，事件处理函数会作为参数传给 jQuery 对象的事件方法，例如页面 5-1.html 中的 ready()方法。在实际开发中，为了简化开发，通常省略 ready()方法，而将事件处理函数直接传给工厂函数。例如，例 5-1 所示代码可简化如下：

```
<script type="text/javascript">
   $(function () {
      alert("Hello jQuery...")
   })
</script>
```

另外，在 jQuery 中，符号“$”实际代表的是 jQuery 对象本身（这里注意区别其他 jQuery 类型的对象），因此工厂函数 $()也可写成 jQuery()，$后面也可以直接调用方法。例如，如下代码调用了 jQuery 对象的 trim()方法会将字符串“ my university ”两边的空格去除并在控制台打印“my university”。

```
jQuery(function () {
       var str = ' my university  '
       console.log('---'+ $.trim(str) + '---')
})
```

5.3　jQuery 对象和 DOM 对象

在实际开发中，容易混淆的是 jQuery 对象和 DOM 对象的概念和用法。因此，在详细讲述 jQuery 语法之前，有必要弄清楚两者的区别和联系。

1. DOM 对象

使用原生 JavaScript 方法如 getElementById()或 getElementsByTagName()等方法获

取到的 DOM 元素就是 DOM 对象，DOM 对象有其独有的属性和方法。例如下述代码将获取 id 为 my 的 DOM 对象，并访问 DOM 对象独有的 innerHTML 属性。

```
var domObj = document.getElementById("my") // 获取 DOM 对象
var objHTML = domObj.innerHTML             // 使用 DOM 对象的 innerHTML 属性
```

2. jQuery 对象

jQuery 对象是指通过工厂函数将 DOM 对象转换后的对象，只有 jQuery 对象能够使用 jQuery 方法。例如下述代码将 id 为“my”的元素转换为 jQuery 对象后，调用 jQuery 对象独有的 html()方法。

```
$("#my").html()                            //获取 id 为 my 的元素的 html 内容
```

其中，给工厂函数传的 #my 表示 jQuery 选择器(用单引号或双引号引用来表示)，该选择器选择了页面中 id 属性为 my 的 DOM 元素并将之转换为 jQuery 元素，进而调用 html()这一 jQuery 方法，功能等价于 document.getElementById(“my”).innerHTML。

需要注意的是，jQuery 对象无法使用 DOM 对象的任何属性和方法，同样，DOM 对象也无法使用 jQuery 对象的任何属性和方法。例如，$(“#my”).innerHTML 和 domObj.html()都是错误的使用方法。

3. 相互转换

实际开发中，往往使用 jQuery 对象操作 DOM 更加快捷、方便，但在某些特定场景下也需要将 jQuery 对象转化成原生的 DOM 元素进行操作，可以使用 get()方法将 jQuery 对象转换成 DOM 对象，例如：

```
var myDOM = $("#my").get()                 //将 jQuery 对象转换为 DOM 对象
```

当 jQuery 对象是数组或者集合时，则需要给 get()方法指定相应的索引。将 DOM 对象转换成 jQuery 直接使用工厂函数即可，例如：

```
var $myJQuery = $(domObj)                  //将 DOM 对象转换为 jQuery 对象
```

实际开发中，通常给 jQuery 对象类型的变量名前加“$”符号以区分 DOM 对象类型的变量，例如这段代码中的变量 $myJQuery。

5.4 jQuery 选择器

选择器是 jQuery 的核心之一，其主要作用是为方便获取页面中的元素，然后为该元素添加相应的行为，使页面交互变得快捷、丰富。根据 jQuery 选择器获取元素方式的不同，可以分为通用 CSS 选择器和过滤选择器两种。

5.4.1 通用 CSS 选择器

我们将 CSS 选择器加上单引号或者双引号作为 jQuery 的工厂函数的参数，称为 jQuery 的通用 CSS 选择器，语法如下：

```
$('CSS selector')
```

该函数会返回相应页面 DOM 元素转化后的 jQuery 对象。jQuery 支持大多数 CSS 选择器，包括基本选择器、层次选择器和属性选择器，其语法与 CSS 选择器完全相同。下面给出这 3 种选择器的基本用法和简单示例。

1. 基本选择器

基本选择器包括标签选择器、类选择器、ID 选择器、并集选择器、交集选择器和全集选择器，通过基本选择器可以实现大多数页面元素的查找，关于基本选择器的说明如表 5-2 所示。

表 5-2 jQuery 基本选择器

名　称	返回值	示　例
标签选择器	元素集合	$('h1')选取所有<h1>元素
类选择器	元素集合	$(".title")选择所有 class 属性为 title 的元素
id 选择器	单个元素	$("#my")选择 id 为 my 的元素
并集选择器	元素集合	$('div,span,p')选取所有 、<p>和元素
交集选择器	单个或者多个	$('div#my')选择 id 为 my 的 元素
全局选择器	元素集合	$('*')选择所有页面元素

2. 层次选择器

层次选择器主要用来选择当前元素的后代元素、子元素、相邻元素和兄弟元素。有关层次选择器的用法说明见表 5-3。

表 5-3 jQuery 层次选择器

名　称	返回值	示　例
后代选择器	元素集合	$('#my span')选择 id 为 my 元素下的所有后代元素
子选择器	元素集合	$('#my>span')选择 id 为 my 元素下的所有子元素
相邻元素选择器	元素集合	$('#my+span')选择紧邻 id 为 my 元素之后的兄弟元素
兄弟元素选择器	元素集合	$('#my~span')选择 id 为 my 元素之后的所有兄弟元素

3. 属性选择器

属性选择器通过 HTML 元素的属性来选择相应的元素，有关属性选择器的用法说明见表 5-4。

表 5-4 jQuery 属性选择器

语法构成	返回值	示　例
[attr]	元素集合	$("[href]")选择含有 href 属性的元素
[attr=val]	元素集合	$("[href='#']")选择 href 属性值为“#”的元素
[attr!=val]	元素集合	$("[href!='#']") 选择 href 属性值不为“#”的元素
[attr^=val]	元素集合	$("[href^='http']") 选择 href 属性值以 http 开头的元素
[attr$=val]	元素集合	$("[href$='com']") 选择 href 属性值以 com 结尾的元素
[attr*=val]	元素集合	$("[href*='wzu']") 选择 href 属性值包含 wzu 的元素
[attr1=val1][attr2=val2]…	元素集合	$("div[id][class='univ']")选择含有 id 属性并且 class 属性为 univ 的 元素

5.4.2 过滤选择器

过滤选择器的主要作用是在原有匹配的元素中进行二次筛选，有关过滤选择器的用法说明见表 5-5。

表 5-5 jQuery 过滤选择器

语法构成	返回值	示　例
:first	单个元素	$("li:first")选取第一个 元素
:last	单个元素	$("li:last")选取最后一个 元素
:not(selector)	集合元素	$("li:not(.my)")选取 class 属性不是 my 的 元素
:even	集合元素	$("li:even")选取索引为偶数所有 元素
:odd	集合元素	$("li:odd")选取索引为奇数所有 元素
:eq(index)	单个元素	$("li:eq(3)")选取索引为 3 的 元素
:gt(index)	集合元素	$("li:gt(3)") 选取索引大于 3 的 元素
:lt(index)	集合元素	$("li:lt(3)") 选取索引小于 3 的 元素
:hidden	集合元素	$(":hidden")选取所有隐藏的元素
:visible	集合元素	$(":visible")选取所有可见的元素

5.5 jQuery 事件处理机制

在了解了 jQuery 选择器后，就可以结合 jQuery 事件处理机制与页面进行交互。实际开发中常用的步骤为首先通过 jQuery 选择器选取 DOM 元素，然后给选定的元素绑定事件及相应的处理函数。jQuery 绑定事件的方式主要有两种，第一种语法如下：

```
$('selector').eventName(function(){函数体})
```

其中 selector 表示选择器，eventName 表示事件名称，function 为事件响应函数。

【例 5-2】 创建页面 5-2.html。

```
<script type="text/javascript">
    $(function () {
      $("#btn1").click(function () {
          alert("你好!")
      })
   })
</script>
<body>
  <button id="btn1">点我弹框</button>
</body>
```

例 5-2 所示代码为页面中 id 为 btn1 的按钮绑定了单击事件，单击该按钮会执行相应的事件处理函数，此处为弹出一个信息为“你好”的警告框。

另外一种通用的事件绑定的语法如下：

```
$('selector').on("eventName", function(){函数体})
```

该语法通过 jQuery 的 on()函数来绑定事件，on()函数的第一个参数为绑定的事件名称，第二个参数为事件响应函数，具体的事件名称可通过官方文档（https://jquery.cuishifeng.cn/on.html）进行查阅。例如，页面 5-2.html 中的代码可以修改为：

```
$("#btn1").on('click', function () {
    alert("你好!")
})
```

可以看出，第一种绑定事件的方式相对直观，编码方便，但一次只能添加一个监听，而且有的监听事件不支持这种方式。第二种方式则更加通用，且可以添加多个监听。因此，实际开发中推荐使用第二种方式。例如，在页面 5-2.html 中添加如下代码：

```
$('#div1')
    .on('mouseenter', function () {
        console.log('进入')
    })
    .on('mouseleave', function () {
        console.log('离开')
})
<div id = "div1" style = "height: 100px; width: 100px"> div1 </div>
```

以上代码的功能是为 id 为 div1 的< div >同时绑定了鼠标移进和移出事件，当鼠标指针移进该 DIV 区域时会触发事件处理函数 mouseenter，此处为在控制台打印“进入”二字；当鼠标移出该 DIV 区域时会触发事件处理函数 mouseleave，此处为在控制台打印“离开”二字。

尽管 jQuery 中还存在许多其他事件，如键盘事件、表单事件等，但用法与例 5-2 基本一致，读者只需理解并掌握事件处理的原理即可，以后在遇到实际问题时便可自行查阅 jQuery 官方文档。

5.6 jQuery 中的 DOM 操作

jQuery 中提供了一系列的操作 DOM 的方法，它们不仅简化了使用传统 JavaScript 操作 DOM 时烦琐的代码，而且解决了跨平台浏览器兼容性问题，从而提高了开发效率并且令用户与浏览器的交互更加便捷。jQuery 中的 DOM 操作主要分为内容操作、节点操作和样式操作，下面分别对这三部分做详细介绍。

5.6.1 内容操作

jQuery 提供了对元素内容的操作方法，即对 HTML 代码、标签内容和属性值内容进行操作。

1. 操作 HTML 代码

jQuery 主要使用 html()方法来对 HTML 代码进行操作，该方法类似于 JavaScript 中的 innerHTML 属性，通常用于动态新增和替换页面的内容，其语法格式如下：

```
html([content])
```

其中，content 表示可选参数，当有值时表示设定被选元素的新内容，当没有参数时表示获取

被选元素的内容。

【例 5-3】 创建页面 5-3.html。

```
<script type="text/javascript">
    $(function () {
        $('#btn1').click(function () {
            $('#ans').html("<ul><li>Q1</li><li>Q2</li></ul>")
        })
        $('#btn2').click(function () {
            alert($('#ans').html())
        })
    })
</script>
<body>
  <button id="btn1">单击显示问题</button>
  <button id="btn2">单击弹出问题</button>
  <div id="ans"></div>
</body>
```

例 5-3 所示代码在单击“显示问题”按钮时会在 id 为 ans 的 中通过 html()方法设置页面内容，显示效果如图 5-3 所示。给元素设置完 HTML 内容后，单击“弹出问题”按钮会将该元素的 HTML 内容以对话框形式弹出，如图 5-4 所示。在页面初始阶段单击该按钮，弹出的内容为空。可以看出，html()方法返回的是该标签节点的所有内容。

图 5-3 设置元素的 HTML 内容

图 5-4 获取元素的 HTML 内容

2. 操作标签内容

使用 text()方法可以获取或设置元素的文本内容，而不含 html。例如在 5-3.html 页面中添加如下代码：

```
<script>
  …
  $('#btn3').click(function () {
      alert( $('#ans').text())
  })
</script>
<button id = "btn3">单击弹出文本</button>
```

单击完“显示问题”按钮后，单击“弹出文本”按钮，显示效果如图 5-5 所示。

图 5-5　获取元素的 text 内容

同样的，使用 text()方法设置元素内容时，html 也将会转义成普通字符来解析。例如，继续在 5-5.html 页面中添加如下代码：

```
<script>
  $('#btn4').click(function () {
      $('#ans').text("<ul><li>Q1</li><li>Q2</li></ul>")
  })
</script>
<button id = "btn4">单击设置文本</button>
```

直接单击该按钮，所弹出的内容如图 5-6 所示。

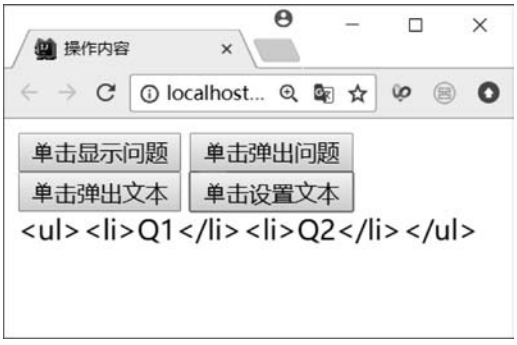

图 5-6　使用 text()方法设置元素文本

3. 操作 value 属性值

jQuery 对象的 val()方法常用于获取和设置 DOM 元素的 value 属性值。

【例 5-4】 创建页面 5-4.html。

```
<script>
   $('#search').
        on("focus", function () {
           if( $(this).val() == "请输入内容"){
            $(this).val("")
        }
    }).on("blur", function () {
           if( $(this).val() == ""){
               $(this).val("请输入内容")
        }
   })
</script>
<input type="text" value="请输入内容" id="search">
```

例 5-4 所述代码创建的初识页面显示文本框，并且文本框有默认值“请输入内容”，如图 5-7 所示。当文本框获得鼠标焦点时会触发 focus 事件，处理函数会获取文本框的 value 属性值并判断是否为默认值，如果是则清空内容，如图 5-8 所示。当文本框失去焦点时，如果当前文本框内容为空则恢复为默认值。

图 5-7 初始文本框状态

图 5-8 获取焦点时文本框状态

5.6.2 节点操作

DOM 中的节点类型分为元素节点、文本节点和属性节点，文本节点和属性节点又包含在元素节点中，其中，文本节点又属于内容，已在 5.5.1 节中介绍过。下面主要讲述元素节点的创建、查找、插入、删除、替换、遍历以及属性节点的获取和设置等。

1. 创建和插入节点

jQuery 在页面中创建新元素的语法为 $(html)，其中 $()为工厂函数，参数为标准 HTML 代码。例如以下代码创建了一个 id 为“city”、内容为“城市”的<li>元素节点。

```
$("<li id='city'>城市</li>")
```

上述代码仅创建了一个新元素，尚未添加到 DOM 中。要想在页面中新增一个节点，则必须将创建的节点插入到 DOM 中。jQuery 提供了多种方法来实现节点的插入，从插入方式上来看主要分为两大类：内部插入和平行插入，具体方法见表 5-6。

表 5-6　插入节点方法

插入方式	方　法	案　例
内部插入	append(n)	$('F').append('c')表示将 c 作为子节点插入到 F 的尾部
	appendTo(n)	$('c').appendTo('F')表示将 c 作为子节点插入到 F 的尾部
	prepend(n)	$('F'). prepend('c')表示将 c 作为子节点插入到 F 的首部
	prependTo(n)	$('c'). prependTo('F')表示将 c 作为子节点插入到 F 的首部
平行插入	after(n)	$('A').after('B')表示将 B 作为兄弟节点插入到 A 之后
	insertAfter(n)	$('A').insertAfter('B')表示将 A 作为兄弟节点插入到 B 之后
	Before(n)	$('A').before('B')表示将 B 作为兄弟节点插入到 A 之前
	insertBefore(n)	$('A'). insertBefore('B')表示将 A 作为兄弟节点插入到 B 之前

下面以常见的 append()方法为例,讲述插入节点在开发中常见的应用场景。

【例 5-5】　创建页面 5-7.html。

```
<script type="text/javascript">
    $(function () {
        $('#btn1').click(function () {
            $('ul').append("<li>复旦大学</li>")

        })
    })
</script>
<body>
  <h2>中国著名大学</h2>
  <ul class="univ">
     <li>清华大学</li>
     <li>北京大学</li>
     <li>浙江大学</li>
  </ul>
  <button id="btn1" value="添加节点">添加</button>
</body>
```

例 5-5 所示代码的初始页面效果如图 5-9 所示,单击“添加”按钮将在节点<ul>内追加新创建的<li>节点,效果如图 5-10 所示。

appendTo()方法表示将前一个节点追加到后一个节点后面,调用方法的节点必须是 jQuery 对象。例如,在 5-5.html 中添加下述代码:

```
$('#btn2').click(function () {
    $('<li>南开大学</li>').appendTo( $('ul'));
})
<button id="btn2" value="添加节点 2"> appendTo </button>
```

利用 appendTo(),可以将节点“南开大学”作为最后一项追加到
列表的后面。需要注意的是,调用方法的对象必须是 jQuery 对象。appendTo()方法传入的参数可以是 jQuery 对象,也可以是普通的字符串,比如上述代码粗体部分也可改成“$('
南开大学
').appendTo('ul')”。表 5-6 中其余几个方法的用法同 append()和 appendTo()方法类似,请读者自行尝试。

图 5-9　追加节点初始页面

图 5-10　追加后页面

2. 删除、清空和替换节点

使用 jQuery 删除页面节点的方法为 remove()，该方法用于删除匹配元素及其包含的文本和子节点。例如，$('ul').remove()将会删除< ul >元素及其所有后代节点。不同于 remove()方法，empty()方法能清空元素中的所有后代节点。例如 $('ul').empty()将会清空< ul >标签内所有的节点，而< ul >元素本身不被删除。

jQuery 中替换节点的方法有 replaceWith()和 replaceAll()。前者的作用是将所有匹配的元素替换成指定的节点。例如，$('. ul li:eq(1)').replaceWith("< li >西湖大学
")将< ul >下第 2 个< li >元素(“北京大学”)替换成括号内的内容(“西湖大学”)。replaceAll()方法的作用相同，只是颠倒了 replaceWith()方法的操作顺序，类似于 append()方法和 appendTo()方法。继续在页面 5-7. html 中添加如下代码：

```
$('#btn3').click(function () {
    $('ul').remove();
})
```

```
$('#btn4').click(function () {
    $('ul').empty();
})
$('#btn5').click(function () {
    $('ul li:last').replaceWith("<li>中国科学院</li>");
})
$('#btn6').click(function () {
    $('<li>国防科大</li>').replaceAll('ul li:last');
})
<button id="btn3" value="删除节点">remove</button>
<button id="btn4" value="清空节点">empty</button>
<button id="btn5" value="替换节点 1">replaceWith</button>
<button id="btn6" value="替换节点 2">replaceAll</button>
```

在创建的页面中单击 id 为“btn4”的按钮后，页面所有<li>节点会被清空，但会保留<ul>节点。单击前，页面及其源码如图 5-11 所示；单击后，页面及其源码如图 5-12 所示。

图 5-11　初始页面及源码

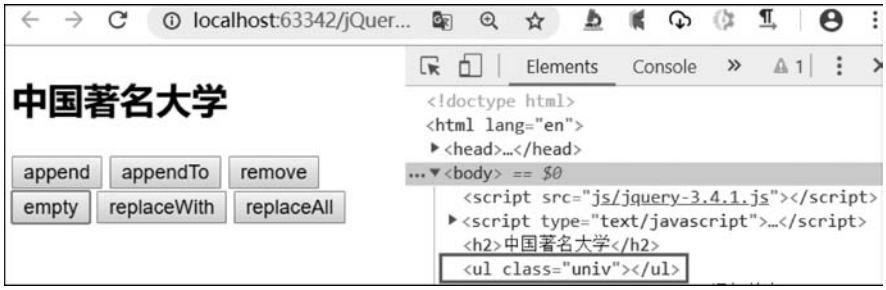

图 5-12　清空列表项后页面及源码

单击 id 为“btn5”的按钮会将列表的最后一项替换成“中国科学院”，单击 id 为“btn6”的按钮会用“国防科大”替换掉列表的最后一项，相应效果请读者自行尝试。

3. 查找和遍历节点

jQuery 中除了可以使用选择器来查找节点外，还能通过已选择到的元素获取与其相邻的兄弟节点、父子节点等进行二次操作。此类方法中，常见的有 children()、next()、prev()、

siblings()、parent()、parents()等。例如，$('ul').children().length 将会获取<ul>标签所有子元素的个数；$('ul li:eq(1)').next().html()将会获取<ul>下第 2 个<li>元素后面紧邻元素的 html 内容；$('ul li:eq(1)').siblings().length 将会获取<ul>下第 2 个<li>元素前后所有同辈元素的个数；$('ul li:eq(1)').parent().html()将会获取<ul>下第 2 个<li>元素的父元素的 html 内容。读者可以根据以上描述自行在 5-5.html 页面中添加代码并观察结果。

遍历节点是 Web 开发中一个非常重要的功能，常用于在前端页面中遍历服务端传来的数据。其语法如下所示：

```
$(selector).each(function(index, item))
```

其中，each()表示遍历节点集合方法；function()为每个节点的处理函数；参数 item 表示当前元素，其对象类型为 DOM 类型，因此需要转换成 jQuery 对象后才能调用 jQuery 方法；index 表示当前元素的索引。例如，在 5-5.html 中添加以下代码：

```
$('#btn7').click(function () {
    $("li").each(function (index, item) {
        console.log(item.innerHTML + " === " + index)
    })
})
<button id = "btn7" value = "遍历节点"> each </button>
```

上述代码中的 each()方法将遍历页面中所有<li>元素集合，并在控制台打印每个节点的 html 内容和索引值，效果如图 5-13 所示。

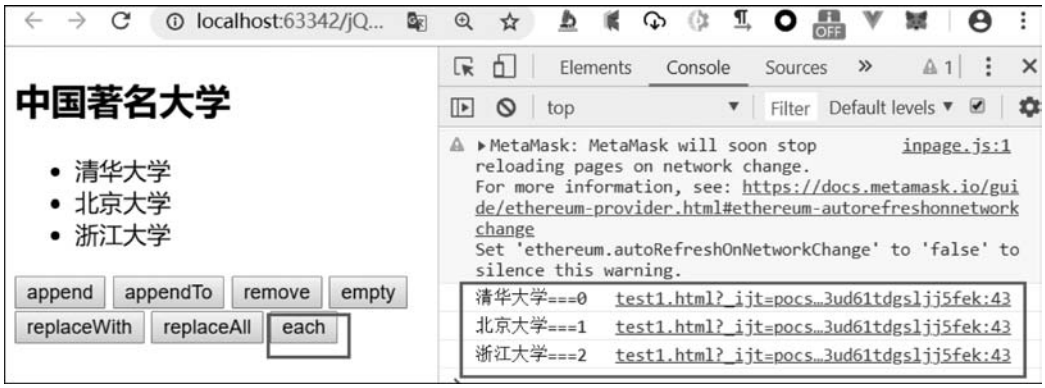

图 5-13　each()方法遍历集合元素

其中，item 对象为当前遍历的元素节点，是 DOM 类型，因此需要调用 innerHTML 属性才能访问节点内容。如要对 DOM 元素使用 jQuery 方法，需要将其转换为 jQuery 对象，例如上述代码也可改成如下形式：

```
$("li").each(function (index, item) {
   console.log( $(item).html() + " === " + index)
})
```

在实际开发中，一定要时刻注意 DOM 对象和 jQuery 对象的区分。

4. 属性节点的操作

除了操作元素节点本身，很多时候需要操作元素的属性。在 jQuery 中，有两种常用的操作元素属性的方法，分别为 attr()和 removeAttr()方法，前者可以用来获取和设置元素的属性，后者可以用来删除元素的属性。例如，$("#tab1").attr("border", 1)将为 id 为 tab1 的表格添加 1 个像素的边框，$("#img1").attr("width")将会获取 id 为 img1 的图片的宽度值，$("#img1").removeAttr("alt")将会删除 id 为 img1 的图片的 alt 属性。

5.6.3 样式操作

在 jQuery 中，对元素的样式操作主要包括直接设置样式值、获取样式值、追加样式、移出样式和切换样式。

1. 设置和获取样式

jQuery 中常用的设置样式的方法为 css()，其基本语法如下：

```
$(selector).css(name, value) //设置 css 属性
```

也可以给 css()方法传多个 name-value 对：

```
$(selector).css({name:value, name:value, …})
```

其中，name 用来规定 css 属性的名称，该参数可以是任何 css 属性，value 用来规定 css 属性的值。如果该方法没有 value 参数，则表示获取元素的 css 的值，语法如下：

```
$(selector).css(name)     //获取 css 属性
```

【例 5-6】 创建页面 5-6.html。

```
<style type="text/css">
    #div1{width: 50px; height: 50px}              //CSS 代码
</style>
<div id="div1">div1</div>                         //HTML 代码
<script>
    $(function () {
        $('#div1').mouseover(function () {        //jQuery 代码
            $('#div1').css("background-color", "red")
        })
    })
</script>
```

例 5-6 所示代码功能为，当鼠标指针移动到 div1 区域时，该区域背景变红色。如要获取 div1 的宽度值，可用 $('#div1').css("width")。

2. 追加和移除样式

jQuery 中可以使用 addClass()方法来追加样式，其语法格式如下：

```
$(selector).addClass(class) //追加单个样式
```

也可以为：

```
$(selector).addClass(class1 class2…) //追加多个样式
```

在页面 5-6.html 中添加以下代码：

```
.div2{
    background-color: yellow;
    border: 1px solid
}
<div id="div2">div2</div>
$('#div2').on('mouseover', function () {
        $('#div2').addClass("div1 div2")
})
```

上述代码功能为，当鼠标指针滑过 div2 时，为 div2 追加了样式 div1 和样式 div2（增加高度、宽度、背景颜色和边框）。追加样式本质上是为元素的 class 属性添加值。类似地，移出样式的语法如下：

```
$(selector).removeClass(class) //移除单个样式
```

也可以为：

```
$(selector). removeClass(class1 class2…) //移除多个样式
```

例如，在页面 5-6.html 中添加下述代码后，当鼠标指针移除 div2 的区域时，移除样式 div2（背景颜色和边框消失）。

```
$('#div2').on('mouseout', function () {
    $('#div2').removeClass("div2")
})
```

5.7 表单验证

表单是客户端向服务器端提交数据的主要媒介。为了保证表单数据的有效性和准确性，应用程序在处理业务之前通常需要先对数据进行验证，验证成功后再将数据发送给服务端。常用的验证方式有客户端验证和服务端验证，客户端验证本质上是在当前页面上调用脚本程序来对表单数据进行验证，而服务端验证则是将请求提交给服务端后，由服务端程序对提交的表单数据进行验证。这两种方式有各自的优势，客户端验证能在很大程度上减轻服务器的负担，而服务端验证能保证应用的安全性和有效性。因此，在实际开发中，通常将这两种验证方式结合使用。

本章主要讲述如何使用正则表达式和 jQuery 技术进行客户端验证。在讲解具体技术之前先看一个案例，在开发 HTML 表单时需要对用户输入的内容进行验证，例如，验证用户名、邮箱格式是否正确等。图 5-14 展示了网易邮箱注册页面，当输入的邮箱格式错误时，页面会直接给注册用户一定的提示。这是如何做到的呢？实际开发中通常由正则表达式结合 jQuery 来实现此类功能。

图 5-14 网易邮箱注册表单验证

正则表达式是一个描述字符模式的对象，它由一些特殊的符号组成。本节主要讲述如何在

JavaScript 中使用正则表达式。

1. 定义正则表达式

在 JavaScript 中，正则表达式有两种定义方式，一种是普通方式，另一种是构造函数方式。普通方式语法如下：

```
var reg = /模式/修饰符
```

其中，模式代表了某种规则，可以使用一些特殊符号来构成。模式是正则表达式的核心，本节后面会进行详细讲述。修饰符用来扩展表达式的含义，主要有 3 个，并且可以任意组合。

(1) G：表示全局匹配；

(2) I：表示不区分大小写匹配；

(3) M：表示可以进行多行匹配。

构造函数方式语法如下：

```
var reg = new RegExp("模式","修饰符")
```

创建完正则表达式后，可以使用 test()方法来检测一个字符串是否匹配该模式，语法格式为：

```
reg.test(字符串)
```

【例 5-7】 创建页面 5-7.html。

```
<script type = "text/javascript">
    var reg = /wzu/i
    var s = "wdfdfaawertrfwZudfadfadf"
    alert(reg.test(s))
</script>
```

上述代码在 JavaScript 中定义了一个模式 reg，该模式用来匹配目标字符串中是否包含子串 wzu，修饰符 i 表示匹配过程中忽略大小写。可以看出，此时 test()方法应该返回 true，而如果没有修饰符 i，则返回 false。再看一个例子：

```
var reg1 = /^wzu/gm
var s1 = " wzudfda\nwzu";
alert(s1.replace(reg1, "www"))
```

上述代码定义了一个模式 reg1，该模式用于匹配以 wzu 开头的字符串。其中字符串 s1 存在符号“\n”是多行字符串，由于加了全局匹配和多行匹配修饰符，该模式会将每一行作为一个单独的字符串处理，因此上述代码返回结果为 wwwdfda\nwww，如果不加修饰符 g，则首次匹配成功后即返回。如果不加修饰符 m，则不支持多行匹配。

2. 表达式的模式

正则表达式的模式一般分为简单模式和复合模式两种。简单模式是指通过普通字符的组合来表达的模式，例如 var reg=/wzu/等。简单模式只能匹配普通字符串，不能满足复杂需求。下面着重讲解复合模式。

复合模式是指通过利用通配符来表达语义的模式，常见的正则表达式符号有选择符和

量词符。其中，选择符如表 5-7 所示，量词符如表 5-8 所示。

表 5-7　正则表达式选择符

符号	含　义	示　例
[]	匹配指定集合内的任一个字符	/[234]/，匹配包含 2 或 3 或者 4 的字符串、/[0-9]/表示匹配任意数字、/[A-Z]/表示匹配任意大写字母、/[^A-z]/表示匹配非英文字母
^	匹配字符串的开始	/^wzu/，匹配以 wzu 开头的字符串
$	匹配字符串的结尾	/$wzu/，匹配以 wzu 结尾的字符串
\d	匹配一个数字序列	/\d/，等价于/[0-9]/
\D	匹配除了数字之外的任何字符	/\D/，等价于/^0-9/
\w	匹配一个数字、下画线或字母	/\w/，等价于[A-z0-9_]
\W	匹配任何非单字字符	/\W/，等价于[^A-z0-9_]
.	匹配除了换行符之外的任意字符	/./，等价于/[^\n\r]/

表 5-8　正则表达式量词符

符号	含　义	示　例
n?	匹配 0 次或 1 次字符 n	/a? /，表示匹配出现字符 a 零次或 1 次的字符串
n*	匹配 0 次或多次字符 n	/a*/，表示匹配出现字符 a 零次或多次的字符串
n+	匹配 1 次或多次字符 n	/a+/，表示匹配出现字符 a 一次或多次的字符串
n{x}	匹配字符 n 出现 x 次	/a{3}/，表示匹配出现字符 a 三次的字符串
n{x, y}	匹配字符 n 出现 x 次到 y 次	/ a{2, 4}/，表示匹配出现字符 a 二到四次的字符串
n{x,}	匹配字符 n 出现>=x 次	/ a{3, }/，表示匹配出现字符 a>=3 次的字符串

3. 实际案例

了解了正则表达式的基本知识后，在实际开发中就可使用正则表达式来验证表单数据了。以用户注册表单为例，需要验证的内容有用户名、密码、邮箱、手机号码、身份证等，主要验证输入的内容是否满足长度要求、是否含有特殊字符、是否符合一定的规则等。例如，我国身份证号码的规则为：

(1) 15 位或者 18 位；

(2) 18 位最后一位可能为 X 或者数字。

可定义正则表达式为：

```
var reg = /^\d{15}(\d{2}[0-9xX])?$/
```

其中，d{15}表示前 15 位是数字，? 表示(\d{2}[0-9xX])这一部分可出现 0 次或 1 次，该部分前 2 位是数字，最后 1 位是字符 x 或 X。

验证电子邮箱是否符合规则的正则表示可定义如下：

```
var reg = /^\w+@\w+(\.[A-z]{2,3}){1,2} $/
```

其中，^\w+表示邮箱必须以数字、字母或者下画线开头并且可以出现多次，@\w+表示必须出现@符号，并且之后必须跟 1 次或多次数字、字母或者下画线。\.[A-z]{2,3}表示“.”号后面跟 2～3 个字母，其中“.”是正则表达式保留字，因此需要使用“\”进行转义。

定义好规则后，便可以使用 jQuery 来对表单数据进行验证。在页面 5-7.html 中添加以下代码：

```
$(function () {
    $("input[name='identify']").blur(function () {
        var idf =  $(this).val()
        var reg =   /^\d{15}(\d{2}[0-9xX])?$/;
        if(reg.test(idf) == false){
            $("#idfInfo").html("身份证号码不正确,请重新输入")
        }else{
            $("#idfInfo").html("")
        }
    })
})
<input type="text" value="请输入身份证" name="identify"><span id="idfInfo"></span>
```

该函数表示当输入框失去焦点后，将触发事件处理函数。该函数首先获取用户输入的内容，然后测试该内容是否匹配正则表达式定义的模式，如果不匹配则在 id 为 idfInfo 的 标签（默认为空）中给出提示信息，效果如图 5-15 所示。输入正确的身份证号则不显示任何信息。

图 5-15　正则表达式判定身份证号

根据上述原理，根据不同的规则，可以对不同的数据进行表单验证，读者可自行尝试。

小　　结

本章介绍了 jQuery 技术的基础知识，包括如何搭建开发环境、jQuery 选择器的用法、使用 jQuery 操作 DOM 元素、利用 jQuery 和正则表达式进行表单验证等。jQuery 技术在前端开发中应用得较为广泛，也是后面学习 Ajax 技术的基础，读者需要重点掌握本章所讲述的核心基础部分。