第五节 jQuery

jQuery是一个javascript库，核心理念是write less,do more(写得更少,做得更多)，他内部帮我们把几乎所有功能都做了封装，相比上一节基于DOM、BOM操作会更加简单。例如：

```
// DOM根据ID选择标签对象
document.getElementById("xx")

// jQuery根据ID选择标签对象
$('#xx')
```

jQuery封装了非常多的功能，我们课程会结合案例将最常用的功能做详细讲解。

学前必备

1. 快速应用

在使用jQuery时，需要提前下载并应用jQuery之后，才能在代码中使用。

- 下载jQuery http://jquery.com/download/
- 应用jQuery

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'/>
    <title>jQuery</title>
</head>
<body>
    <div class="body">
        <ul>
            <li id="login">登录</li>
            <li id="register">注册</li>
        </ul>
    </div>

    <!--引入jQuery-->
    <script type="text/javascript" src="js/jquery-3.4.1.min.js"></script>
    <script>
        // 利用jQuery提供的功能获取标签文本
        var value = $('#login').text();
        console.log(value);
    </script>
</body>
</html>
```

2. DOM对象和jQuery对象

DOM对象和jQuery对象都为标签提供了各种各种功能，并且两者之间可以进行相互转换。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'/>
    <title>jQuery</title>
</head>
<body>
    <div id="content">人生如烟，烟如雾。</div>

    <!--引入jQuery-->
    <script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.min.js"></script>

    <script>
        // DOM操作
        // 获取文本
        var txt = document.getElementById('content').innerText;
        document.getElementById('content').innerText = '沙雕';

        // jQuery操作
        var text = $('#content').text();
        $('#content').text('二货');

        // Dom对象转换jQuery对象：$(dom对象)
        $(document.getElementById('content'))

        // jQuery对象转换成Dom对象：jQuery对象[0]
        $('#content')[0]
    </script>
</body>
</html>
```

5.1 选择器

jQuery提供了大量选择器，用于帮助开发者快速找到HTML中的指定标签。

5.1.1 id 选择器

HTML代码：

```
<div id="notMe">
    <p>id="notMe"</p>
</div>
<div id="myDiv">白日依山尽</div>
```

jQuery代码：

```
$("#myDiv");
```

结果：

```
[<div id="myDiv">id="myDiv"</div> ]
```

5.1.2 class 选择器

HTML代码：

```
<div class="notMe">窗前明月光</div>
<div class="myClass">疑是地上霜</div>
<span class="myClass">举头望明月</span>
```

jQuery代码：

```
$(".myClass");
```

结果：

```
[ <div class="myClass">窗前明月光</div>, <span class="myClass">举头望明月</span> ]
```

5.1.3 标签选择器

HTML代码：

```
<div>DIV1</div>
<div>DIV2</div>
<span>SPAN</span>
```

jQuery代码：

```
$("div");
```

结果：

```
[ <div>DIV1</div>, <div>DIV2</div> ]
```

5.1.4 多选择器

HTML代码：

```
<div>鸡</div>
<p class="myClass">我顶你个肺</p>
<span>你太美</span>
<p class="notMyClass">哈哈哈哈哈哈</p>
```

jQuery代码：

```
$("div,span,p.myClass")
```

结果：

```
[ <div>鸡</div>, <p class="myClass">我顶你个肺</p>, <span>你太美</span> ]
```

5.1.5 层级选择器

HTML代码：

```
<form>
  <label>Name:</label>
  <input name="name" />
  <div>
      <label>Newsletter:</label>
      <input name="newsletter" />
 </div>
</form>
<input name="none" />
```

jQuery代码：

```
$("form input")
```

结果：

```
[ <input name="name" />, <input name="newsletter" /> ]
```

5.1.6 属性选择器

HTML代码：

```
<input type="checkbox" name="newsletter" value="Hot Fuzz" />
<input type="checkbox" name="newsletter" value="Cold Fusion" />
<input type="checkbox" name="accept" value="Evil Plans" />
```

jQuery代码：

```
$("input[name='newsletter']")
```

结果：

```
[ <input type="checkbox" name="newsletter" value="Hot Fuzz" />, <input
type="checkbox" name="newsletter" value="Cold Fusion" /> ]
```

5.1.7 表单选择器

HTML代码：

```
<form>
    <input type="button" value="Input Button"/>
    <input type="checkbox" />

    <input type="file" />
    <input type="hidden" />
    <input type="image" />

    <input type="password" />
    <input type="radio" />
    <input type="reset" />

    <input type="submit" />
    <input type="text" />
    <select>
        <option>Option</option>
    </select>
    <textarea></textarea>
    <button>Button</button>
</form>
```

jQuery代码：

```
$(":text")      // 找到所有input标签
// $(":input")    找到所有input标签
// $(":password")   找到所有input且type=password的标签
// $(":radio")    找到所有input且type=radio的标签
// $(":checkbox") 找到所有input且type=checkbox的标签
```

结果：

```
[ <input type="text" /> ]
```

5.2 筛选器

筛选器一般用于对选择器选中的标签进行再次筛选。

5.2.1 parent 父标签

HTML代码：

```
<div><p>Hello</p><p>Hello</p></div>
```

jQuery代码：

```
$("p").parent()
```

结果：

```
[<div><p>Hello</p><p>Hello</p></div> ]
```

5.2.2 children 所有子标签

HTML代码：

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

jQuery代码：

```
$("div").children()
```

结果：

```
[ <span>Hello Again</span> ]
```

5.2.3 next 下一个兄弟标签

HTML代码：

```
<p>Hello</p><p>Hello Again</p><div><span>And Again</span></div>
```

jQuery代码：

```
$("p").next()
```

结果：

```
[ <p>Hello Again</p>, <div><span>And Again</span></div> ]
```

5.2.4 prev 上一个兄弟标签

HTML代码：

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

jQuery代码：

```
$("p").prev()
```

结果：

```
[ <div><span>Hello Again</span></div> ]
```

5.3.5 siblings 所有兄弟标签

HTML代码：

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

jQuery代码：

```
$("div").siblings()
```

结果：

```
[ <p>Hello</p>, <p>And Again</p> ]
```

5.3.6 find 子孙中查找标签

HTML代码：

```
<p><span>Hello</span>, how are you?</p>
```

jQuery代码：

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

结果：

```
[ <span>Hello</span> ]
```

5.3.7 first 匹配的第一个标签

HTML代码：

```
<ul>
    <li>list item 1</li>
    <li>list item 2</li>
    <li>list item 3</li>
    <li>list item 4</li>
    <li>list item 5</li>
</ul>
```

jQuery代码：

```
$('li').first()
```

结果：

```
[ <li>list item 1</li> ]
```

5.3.8 last 匹配的最后一个标签

HTML代码：

```
<ul>
    <li>list item 1</li>
    <li>list item 2</li>
    <li>list item 3</li>
    <li>list item 4</li>
    <li>list item 5</li>
</ul>
```

jQuery代码：

```
$('li').last()
```

结果：

```
[ <li>list item 5</li> ]
```

5.3 属性

5.3.1 addClass 添加样式

HTML代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>Title</title>
    <style>
        .c1{
            height:200px;
            width: 200px;
            border-radius: 50%;
            background-color: red;
        }
        .green{
            background-color: green;
        }
    </style>
</head>
<body>
<div class="c1"></div>
</body>
</html>
```

jQuery代码：

```
$('.c1').addClass('green')
```

结果：

```
div标签背景颜色变成了绿色
```

5.3.2 removeClass 删除样式

HTML代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        .c1{
            height:200px;
            width: 200px;
            border-radius: 50%;
            background-color: red;
        }
        .green{
            background-color: green;
        }

    </style>
```

```
</head>
<body>

<div class="c1 green"></div>


</body>
</html>
```

jQuery代码：

```
$('.c1').removeClass('green')
```

结果：

```
div标签背景又变成了红色
```

案例：加载框/Tab菜单切换

5.3.3 html、text文本

HTML代码：

```
<div class="a1">
    <a href="">百度</a>
</div>
<div class="a2"></div>
```

jQuery代码：

```
取值：
  $('.a1').html()
  $('.a1').text()

设置值：
  $('.a2').html('<a href="">京东</a>')
  $('.a2').text('<a href="">京东</a>')
```

结果：

```
取值结果：
  html：<a href="">百度</a>
  text：百度

设置值结果：
  html中的内容会生成一个标签
  text中的内容还是一个文本内容显示，不能识别成标签
```

5.3.4 val 值

HTML代码：

```
<input type="text" id="username">

<input type="text" class="a1" name="sex">男
<input type="text" class="a1" name="sex">女

<input type="text" class="a2" name="hobby">抽烟
<input type="text" class="a2" name="hobby">喝酒
<input type="text" class="a2" name="hobby">烫头

<select name="city" id="s1">
    <option value="1">北京</option>
    <option value="2">上海</option>
    <option value="3">深圳</option>
</select>

<select name="lover" id="s2">
    <option value="1">波多</option>
    <option value="2">苍井</option>
    <option value="3">小泽</option>
</select>
```

jQuery代码：

```
获取值：
  文本输入框：$('#username').val();
  单选radio框：$('.a1:checked').val();
  多选checkout框：$('.a2:checked').val()是不行的;需要循环取值，如下：
    var d = $(':checkbox:checked');
    for (var i=0;i<d.length;i++){
      console.log(d.eq(i).val());
    }

  单选select框：$('#city').val();
  多选select框：$('#lover').val();
```

```
设置值：
  文本输入框：$('#username').val('you are my love');
  单选radio框：$('.a1').val([2]);  #注意里面必须是列表，写的是value属性对应的值
  多选checkout框：$('.a2').val(['2','3'])
  单选select框：$('#city').val('1');
  多选select框：$('#lover').val(['2','3'])
```

结果：

```
```

案例：模态框添加和编辑功能

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        .cover {
            position: fixed;
            top: 0;
            right: 0;
            bottom: 0;
            left: 0;
            background-color: rgba(0, 0, 0, 0.3);
            z-index: 99;
        }

        .modal {
            width: 300px;
            height: 200px;
            background-color: white;
            position: absolute;
            top: 50%;
            left: 50%;
            margin-top: -100px;
            margin-left: -150px;
            z-index: 1000;
        }
        .hide {
            display: none;
        }
    </style>
</head>
<body>

<button id="add">新增</button>
```

```
<table border="1">
    <thead>
    <tr>
        <th>#</th>
        <th>姓名</th>
        <th>爱好</th>
        <th>操作</th>
    </tr>
    </thead>
    <tbody>
    <tr>
        <td><input type="checkbox"></td>
        <td>金老板</td>
        <td>开车</td>
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>景女神</td>
        <td>茶道</td>
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>苑昊（苑局）</td>
        <td>不洗头、不翻车、不要脸</td>
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    </tbody>
</table>

<div class="cover hide"></div>
<div class="modal hide">
    <div>
        <label>姓名:
            <input type="text" id="name">
        </label>
    </div>
    <div>
        <label>爱好:
            <input type="text" id="hobby">
        </label>
    </div>
```

```
    <button id="cancel" type="button">取消</button>
    <button id="submit" type="button">提交</button>

</div>
<script src="jquery.js"></script>
<script>
    // 定义一个清空输入框并且隐藏模态框的方法
    function hideModal(){
         // 1. 清空input的值
        $("#name,#hobby").val('');
        // 2. 隐藏起来
        $(".cover,.modal").addClass('hide');
    }
    // 开除按钮的功能
    $("table").on('click', '.fire', function () { //我们先去学冒泡事件、事件委托然后再回来学这个例子，事件里面都是用的匿名函数，这里用on是因为我
        //们要将新添加进来的每行里面的button标签能够继承这个点击删除的事件
        // 点击开除按钮要做的事儿
        // 把当前行移除掉
        //this  --> 触发当前点击事件的DOM对象
        $(this).parent().parent().remove();  // 链式操作
    });
    // 新增按钮的功能
    $("#add").click(function () {
        // 点击新增按钮要做的事儿
        // 1. 移除cover和modal的hide样式
        $(".cover,.modal").removeClass('hide');
    });
    // 点击modal中的cancel按钮
    $("#cancel").click(function () {
       hideModal();
    });

    // 点击modal中的submit按钮
    $("#submit").click(function () {
        // 1. 获取用户输入的值
        var name = $("#name").val();
        var hobby = $("#hobby").val();
        // 2. 隐藏模态框，清空输入框
        hideModal();
        // 3. 创建一个tr标签，把数据塞进去
        var trEle = document.createElement("tr");
        $(trEle).append('<td><input type="checkbox"></td>');
        $(trEle).append('<td>' + name + '</td>');
        var tdTmp = document.createElement('td');
        tdTmp.innerText = hobby;
        $(trEle).append(tdTmp);
        $(trEle).append('<td><button class="fire">开除</button></td>')
        // 4. 把上一步的tr追加到表格的tbody后面
```

```
        $('tbody').append(trEle);
    });


</script>

</body>
</html>
```

5.3.4 prop 属性值

HTML代码：

```
<input type="checkbox" id="i1" value="1">
```

jQuery代码：

```
$("#i1").prop("checked")
```

结果：

```
false
```

案例：表格全选、反选、取消

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<button id="all">全选</button>
<button id="reverse">反选</button>
<button id="cancel">取消</button>
<table border="1">
   <thead>
    <tr>
        <th>#</th>
        <th>姓名</th>
        <th>爱好</th>
    </tr>
   </thead>
    <tbody>
```

```
    <tr>
        <td><input type="checkbox"></td>
        <td>金老板</td>
        <td>开车</td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>景女神</td>
        <td>茶道</td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>苑昊（苑局）</td>
        <td>不洗头、不翻车、不要脸</td>
    </tr>
    </tbody>
</table>

<script src="jquery.js"></script>
<script>
    // 点击全选按钮 选中所有的checkbox
    // DOM绑定事件方法
    // $("#all")[0].onclick = function(){}
    // jQuery绑定事件方法
    $("#all").click(function () {
        $(":checkbox").prop('checked', true);
    });
    // 取消
    $("#cancel").on("click", function () {
         $(":checkbox").prop('checked', false);
    });
    // 反选
    $("#reverse").click(function () {
        // 1. 找到所有选中的checkbox取消选中
        // $("input:checked").prop('checked', false);
        // 2. 找到没有选中的checkbox选中
        // $("input:not(:checked)").prop('checked', true);
        //你会发现上面这么写，不行，为什么呢？因为你做了第一步操作之后，再做第二步操作的时候，所有标签就已经全部取消选中了，所以第二步就把所有标签选中了

        // 方法:for循环所有的checkbox,挨个判断原来选中就取消选中，原来没选中就选中
        var $checkbox = $(":checkbox");
        for (var i=0;i<$checkbox.length;i++){
            // 获取原来的选中与否的状态
            var status = $($checkbox[i]).prop('checked');
            $($checkbox[i]).prop('checked', !status);
        }
    })
```

```
</script>
</body>
</html>
```

5.4 文档处理

5.4.1 append 内部插入

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
```

jQuery代码：

```
$('#d1').append('<a href="http://www.jd.com">京东</a>');
```

结果：

```
<div class="d1">
    <span>波多</span>
    <a href="http://www.jd.com">京东</a>
</div>
```

5.4.2 prepend 内部插入

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
```

jQuery代码：

```
$('#d1').prepend('<a href="http://www.jd.com">京东</a>');
```

结果：

```
<div class="d1">
    <a href="http://www.jd.com">京东</a>
    <span>波多</span>
</div>
```

5.4.3 after 外部插入

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
```

jQuery代码：

```
$('#d1').after('<a href="http://www.jd.com">京东</a>');
```

结果：

```
<div class="d1">
    <span>波多</span>
</div>
<a href="http://www.jd.com">京东</a>
```

5.4.4 before 外部插入

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
```

jQuery代码：

```
$('#d1').before('<a href="http://www.jd.com">京东</a>');
```

结果：

```
<a href="http://www.jd.com">京东</a>
<div class="d1">
    <span>波多</span>
</div>
```

5.4.5 empty 删除标签内部的标签

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
```

jQuery代码：

```
$('.c1').empty()
```

结果：

```
<div class="d1">

</div>
```

5.4.6 remove 删除标签

HTML代码：

```
<div class="d1">
    <span>波多</span>
</div>
<div>你好</div>
```

jQuery代码：

```
$('.c1').remove()
```

结果：

```
<div>你好</div>
```

案例：表格数据删除

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        .cover {
            position: fixed;
            top: 0;
            right: 0;
            bottom: 0;
            left: 0;
            background-color: rgba(0, 0, 0, 0.3);
            z-index: 99;
        }
```

```
        .modal {
            width: 300px;
            height: 200px;
            background-color: white;
            position: absolute;
            top: 50%;
            left: 50%;
            margin-top: -100px;
            margin-left: -150px;
            z-index: 1000;
        }
        .hide {
            display: none;
        }
    </style>
</head>
<body>

<button id="add">新增</button>
<table border="1">
    <thead>
    <tr>
        <th>#</th>
        <th>姓名</th>
        <th>爱好</th>
        <th>操作</th>
    </tr>
    </thead>
    <tbody>
    <tr>
        <td><input type="checkbox"></td>
        <td>金老板</td>
        <td>开车</td>
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>景女神</td>
        <td>茶道</td>
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    <tr>
        <td><input type="checkbox"></td>
        <td>苑昊（苑局）</td>
        <td>不洗头、不翻车、不要脸</td>
```

```
        <td>
            <button class="fire">开除</button>
        </td>
    </tr>
    </tbody>
</table>

<div class="cover hide"></div>
<div class="modal hide">
    <div>
        <label>姓名:
            <input type="text" id="name">
        </label>
    </div>
    <div>
        <label>爱好:
            <input type="text" id="hobby">
        </label>
    </div>
    <button id="cancel" type="button">取消</button>
    <button id="submit" type="button">提交</button>

</div>
<script src="jquery.js"></script>
<script>

    // 开除按钮的功能
    $("table").on('click', function () {
        // 把当前行移除掉
        //this  --> 触发当前点击事件的DOM对象
        $(this).parent().parent().remove();  // 链式操作
    });

</script>
</body>
</html>
```

5.5 事件

jQuery中也可以为标签进行绑定事件，并且相比于DOM会更加方便。

5.5.1 jQuery绑定事件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>jQuery学习</title>
</head>
<body>
<ul>
    <li>王宝强</li>
    <li>陈羽凡</li>
    <li>谢霆锋</li>
</ul>

<script type="text/javascript">

    // 当页面框架加载完成之后（DOM结构），执行内部代码。
    $(function () {
        // 通过选择器找到指定标签
        $('li').onclick(function () {
            // 触发事件时，都会执行此匿名函数。  $(this)代表当前触发的标签。
        })
    })
</script>
</body>
</html>
```

案例：左侧菜单实现

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JQuery学习</title>
    <style>
        body {
            margin: 0;
        }

        .header {
            height: 48px;
            background-color: #499ef3;
        }

        .body .menu {
            position: fixed;
            top: 48px;
            left: 0;
            bottom: 0;
            width: 220px;
            border-right: 1px solid #dddddd;
            overflow: scroll;
        }
```

```
        .body .content {
            position: fixed;
            top: 48px;
            right: 0;
            bottom: 0;
            left: 225px;
            /* 超出范围的话，出现滚轮 */
            overflow: scroll;
        }

        .body .menu .title {
            padding: 8px;
            border-bottom: 1px solid #dddddd;
            background-color: #5f4687;
            color: white;

        }

        .body .menu .child {
            border-bottom: 1px solid #dddddd;
        }

        .body .menu .child a {
            display: block;
            padding: 5px 10px;
            color: black;
            text-decoration: none;
        }

        .body .menu .child a:hover {
            background-color: #dddddd;

        }

        .hide {
            display: none;
        }
    </style>
</head>
<body>
<div class="header"></div>
<div class="body">
    <div class="menu">
        <div class="item">
            <div class="title" >国产</div>
            <div class="child">
                <a href="#">少年的你</a>
                <a href="#">我不是药神</a>
                <a href="#">我和我的祖国</a>
```

```
            </div>
        </div>
        <div class="item">
            <div class="title" >欧美</div>
            <div class="child hide ">
                <a href="#">战争之王</a>
                <a href="#">华尔街之狼</a>
                <a href="#">聚焦</a>
            </div>
        </div>
        <div class="item">
            <div class="title" >韩国</div>
            <div class="child hide">
                <a href="#">坏家伙们</a>
                <a href="#">寄生虫</a>
                <a href="#">燃烧</a>
            </div>
        </div>
    </div>
    <div class="content"></div>
</div>

<script type="text/javascript">

    $(function () {
        // 给所有样式有 title 的标签绑定事件
        $('.title').click(function () {
            // 当前触发事件的标签
            $(this).next().removeClass('hide');
            $(this).parent().siblings().find('.title').addClass('hide');
        })
    })
</script>
</body>
</html>
```

jQuery有很多事件，使用方法和click都是类似的，事件列表如下：

5.5.2 jQuery事件委托

jQuery的事件绑定是在页面加载完毕之后，找到相关标签并为其绑定事件，如果后期通过js代码有新增表现，那么新标签中模式是没有事件的，如：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery学习</title>
</head>
```

```
<body>

<input type="button" id="btn" value="添加元素">

<ul id="greenBoy">
    <li>王宝强</li>
    <li>陈羽凡</li>
    <li>谢霆锋</li>
</ul>

<script
src="https://cdn.bootcss.com/jquery/3.4.1/jquery.
min.js"></script>

<script type="text/javascript">

    $(function () {
        $('li').click(function () {
            alert($(this).text());
        });

        $('#btn').click(function () {
            var tag = $('<li>');
            tag.text('alex');
            $('#greenBoy').append(tag);
        })
    })
</script>
</body>
</html>
```

为了避免类似这种情况的发生，jQuery中引入了事件委托的概念，只需要基于on进行绑定即可：

```
<script type="text/javascript">

    $(function () {
        // on的第一个参数：事件名称
        // 第二个参数：选择器
        // 第三个参数：事件触发时执行的函数
        $('#greenBoy').on("click","li",function () {
            alert($(this).text());
        });

        $('#btn').click(function () {
            var tag = $('<li>');
            tag.text('alex');
            $('#greenBoy').append(tag);
        })
    })
```

```
</script>
```

5.6 Ajax

ajax作用：通过JavaScript代码向网络上的地址发送异步请求。

为了本地测试方便，我们通过ajax向本地json文件发送请求并获取数据。

- 本地创建 data.json文件

```
[
  {"id":1,"name":"武沛齐","age":18},
  {"id":2,"name":"Alex","age":18},
  {"id":3,"name":"吴老板","age":18}
]
```

- 编写页面 index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery学习</title>
</head>
<body>

<input type="button" id="btn" value="获取数据">

<script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.min.js"></script>
<script type="text/javascript">
    $(function () {
        $('#btn').click(function () {
            $.ajax({
                type: 'GET',
                // 也可以向网络地址 http://www.xxxx.com 发送请求。
                url: 'data.json',
                success: function (arg) {
                    console.log(arg);
                }
            })
        });
    })
</script>
</body>
</html>
```

案例：基于Ajax实现数据管理

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery学习</title>
</head>
<body>
<table border="1">
    <thead>
    <tr>
        <th>id</th>
        <th>姓名</th>
        <th>年龄</th>
    </tr>
    </thead>
    <tbody>

    </tbody>
</table>

<input type="button" id="btn" value="获取数据">

<script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.min.js"></script>
<script type="text/javascript">
    $(function () {
        $('#btn').click(function () {
            $.ajax({
                type: 'GET',
                // 也可以向网络地址 http://www.xxxx.com 发送请求。
                url: 'data.json',
                success: function (arg) {
                    console.log(arg);


                    // 1 先制作出所有的tr标签
                    var s = '';
                    for (var i in arg){
                        var a = '<tr><td>'+ arg[i]['id'] +'</td><td>'+ arg[i]
['name'] +'</td><td>'+ arg[i]['age'] +'</td></tr>';
                        s += a;
                    }
                    // 2 找到tbody标签,将标签添加进去
                    $('tbody').append(s);
                }
            })
        });
    })
</script>
</body>
</html>
```