第八章-jQuery框架Ajax模块

8.1 jQuery框架中的Ajax简介

Ajax技术的核心是XMLHTTPRequest对象 ，该对象是Ajax实现的关键，发送异步请求、接收服务器端的响应以及执行回调等操作都是通过XMLHTTPRequest对象来完成的。

jQuery框架对Ajax操作进行了封装，在jQuery框架的Ajax模块中提供了很多方法用于网络编程。我们主要从Ajax网络请求、Ajax事件以及序列化等方面讲解。

8.2 jQuery框架中的Ajax网络请求

jQuery框架中为我们提供的发送网络请求方法主要有：

- load方法
- ajax方法
- get方法
- post方法
- getScript方法
- getJSON方法

① load方法

语法 `jQ.load(url,[data],[callback])`

参数

- url 资源的请求路径
- data 发送请求时提交的参数，支持查询字符串和JSON对象格式
- callback 加载完成的回调函数，3形参为：响应体 + 状态 + 请求对象

说明 该方法请求远程的资源，并插入到选中的jQ实例对象中。

注意

- 默认发送GET请求，如果传递参数(JSON对象)则发送POST请求。
- 支持对返回的数据进行筛选

代码示例

```
//01 直接加载文件中的数据
//默认发送的是GET请求
//$("#demo").load("php/test.html");

//02 加载文件中的数据，获取其中的某一部分（进行筛选）
//$("#demo").load("php/test.html span");
```

```
//参数在请求体中进行传递
$("#demo").load("php/test.html span",{"namme":"zs"});
```

② ajax方法

语法 $.ajax(url,[settings]) | $.ajax(settings)

常用参数说明

- url 资源的请求路径
- data 发送请求时提交的参数，支持查询字符串和JSON对象格式
- async 是否异步发送网络请求
- cache 是否进行缓存处理
- context 指定回调函数中的this指针
- dataType 预期服务器返回的数据类型
- timeout 请求的超时时间
- beforeSend 请求发送前执行的回调函数
- complete 请求完成后执行的回调函数
- error 请求失败执行的回调函数
- success 请求成功执行的回调函数

说明 该方法是jQuery框架中最底层的Ajax实现，用于发送网络请求。

注意

- 最简单的情况下，$.ajax()可以不带任何参数直接使用。
- 所有的选项都可以通过$.ajaxSetup()函数来进行全局设置。

代码示例

```
$.ajax({
            "url":"php/03-ajax.php",  //设置请求路径
            "type":"get", //设置请求方法，不区分大小写
            "success":function (res,status,xhr) {
                    //请求成功的回调
                $("#demo").html(res); //获取响应状态码
                console.log(status);  //获取请求的状态
                console.log(xhr);     //获取请求对象本身
                console.log(this);    //获取上下文
            },
            "error":function (res) {
                //请求失败的回调函数
                console.log("失败");
                console.log(res);
            },
            //"data":"name=ls"    //参数：查询字符串形式
            "data":{"name":"ls"}, //参数：JSON对象形式
            "timeout":10,         //设置请求超时的时间
            statusText: timeout
            "context":obj,    //设置回调函数中this的指向
            "complete":function (res) {
```

```
                console.log("请求完成");
                console.log(res);
            }
        });
```

③ get和post方法

语法

[1]$.get(url,[data],[callback],[type])

[2]$.post(url,[data],[callback],[type])

参数

- url 资源的请求路径
- data 发送请求时提交的参数
- callback 请求成功的回调函数
- type 服务器端返回内容的格式：包括xml、html、json、script等

GET和POST对比

- GET请求参数跟在URL后，POST请求参数作为请求体发送。
- GET请求对参数大小有限制，而POST请求理论上不受限制。
- GET请求的数据会被浏览器缓存，存在严重的安全性问题。
- 服务器端读取数据的方式不同。在PHP中，区分为\$_GET和\$_POST。

代码示例

```
//发送请求获取服务器返回的文本,把div的内容替换掉
//第一个参数:url请求路径(必传)
//第二个参数:参数 支持两种形式[查询字符串][JSON对象]
//第三个参数:success(response,status,xhr)
//          请求成功的回调
//          response:服务器返回的响应体
//          status:状态说明[success-error]
//          xhr:请求对象
//第四个参数:预期返回的数据类型:json|script|jsonP等
$.get("php/03-get.php",{"param1":"value1"},
    function (response,status,xhr) {
    console.log(response);
    console.log(status);
    console.log(xhr);
})
//注意点:GET请求请求路径一样会缓存,POST请求不会缓存
$.post("php/04-post.php",{"param1":"value1"},
    function (response,status,xhr) {
    console.log(response);
    console.log(status);
    console.log(xhr);
})
```

④ getScript和getJson方法

jQuery框架提供了getScript和getJson方法来直接加载js文件和JSON数据

语法

```
[1] $.getScript(url,[callBack])
[2] $.getJson(url,[callBack])
```

说明

- getScript方法用于加载js文件，并自动执行。
- getJson方法用于加载JSON数据。

代码示例

```
$.getScript("test.js", function(){
          //加载完成后执行的回调函数
           alert("加载并执行JS文件");
});
```

8.3 jQuery框架中的Ajax事件方法

jQuery框架中除了提供上述方法来发送网络请求外，还提供了一些事件方法来对调用Ajax方法过程中的HTTP请求进行精细控制。通过jQuery提供的一些自定义全局函数，能够为各种与Ajax相关的事件注册回调函数。jQuery的Ajax全局事件方法如下：

```
[1] ajaxStart(callBack) =>检测到网络请求开始发送会触发，1次
[2] ajaxStop(callBack) =>检测到网络请求结束会触发，1次
[3] ajaxSend(callBack) =>检测到网络请求开始发送会触发，N次
[4] ajaxComplete(callBack) =>检测到网络请求结束会触发，N次
[5] ajaxError(callBack) => 网络请求失败会触发
[6] ajaxSuccess(callBack) => 网络请求成功会触发
```

```
$(document).ajaxStart(function () {
     console.log("第一个已经开始+++++");
     $("#demoID").show(1000);
})
$(document).ajaxStop(function () {
     console.log("最后一个已经结束+++++");
     $("#demoID").hide(1000);
})
$(document).ajaxSend(function () {
     console.log("开始发送网络请求___");
})
$(document).ajaxComplete(function () {
    console.log("发送网络请求完成___");
})
$.ajax({ //发送网络请求--A
    "url":"php/06-other.php",
    "type":"GET",
    "success":function (res,status,xhr) {
          console.log("网络请求成功--1");
```

```
      }
})
$.ajax({  //发送网络请求--B
    "url":"php/06-other.php",
    "type":"GET",
    "success":function (res,status,xhr) {
          console.log("网络请求成功--2");
      }
})
$(document).ajaxError(function () {
    console.log("请求失败");
})
$(document).ajaxSuccess(function () {
    console.log("请求成功");
})
```

8.4 jQuery框架中的序列化方法

在开发的时候，经常需要把表单中的数据作为网络请求的参数，如果一个一个的获取再拼接成查询字符串那么相当的麻烦，jQuery框架中为我们提供了两个对应的方法，可以方便解决该需求。

`serialize方法` 能够将DOM元素内容序列化为查询字符串。
`serializeArray方法` 可以将DOM元素序列化后返回JSON格式的数据。

代码示例

HTML

```
<body>
<form>
    <input type="text" name="username" id="demo1">
    <input type="text" name="password" id="demo2">
</form>
<button>按钮</button>
<script>
    $("button").click(function () {
        $.ajax({
            "url":"php/07-get.php",
            //"data":"username="+$("#demo1").val()+"&password=" +$("#demo2").val(),
            "data":$("form").serialize(),
            "success":function (res) {
                console.log(res);
            }
        })
        //把表单中的key-value按照固定的格式拼接为查询字符串:username=lisi&password=abcd
        console.log($("form").serialize());
        console.log($("form").serializeArray());
        //[{"username":"zhangsan"},{"password":"123"}];
    })
</script>
</body>
```

- 原创文章，版权声明：自由转载-非商用-非衍生-保持署名 | 文顶顶