Ajax

1. 使用方法

1、引入

<!-- 第一步：引入axios库 -->

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

1. 在JavaScript里调用axios库

<script>

        axios({

            url: 'https://hmajax.itheima.net/api/province' //将想访问的服务器网址传入

        }).then(result=>{  //用回调函数输出看看有没有获得这些数据

            console.log(result)

//不知道它这个对象有哪些属性，就多试几遍，比如先控制台输出这个对象

            console.log(result.data.list)

//现在就可以知道【省份数据】是在这个对象的“data”里的“list”的

            console.log(result.data.list.join('<br>'))

//.join()可以把对象内容转化成字符串文本内容

         document.querySelector('.myP').innerHTML = result.data.list.join('<br>')

//获得html标签，把字符串文本赋给到HTML标签内容里

        })

</script>

二、分析axios库

1、基本格式：axios({

url: ‘要访问的服务器网站’ ——> 就是给它地址

}).then(result => {

要做什么事 ——> 就是要干什么

})

注意！！！.then()方法是axios库的方法，而不是括号里的参数！！

1. 当查询具体信息时需要传一些查询参数给服务器，然后返回对应的信息结果，就要用params

格式：axios({

url: ‘要访问的服务器网站’,

params: {

自定义参数名 : 要传的参数值

}

}).then(result => {

要做什么事

})

例子：

axios({

//某个省某个市的所有地区数据资源所在服务器的地址

url: 'https://hmajax.itheima.net/api/area',

//完整网址形式就应该是“https://hmajax.itheima.net/api/area?pname=XXX&cname=XXX”

    params: {

        pname: pName, //pname是省份或直辖市参数

        cname: cName //cname是市级城市参数

    }

     }).then(result=>{

  ......

    })

拓展url：

(1)、URL,就是获取网络数据资源的“统一资源定位符”，简称“网址”

(2)、然后分析网址：https://hmajax.itheima.net/api/province

    这里【https:】就是“钥匙”【协议】，指定了浏览器和服务器之间传输数据必须按照一致

的格式，协议范围有：http、https....；【hmajax.itheima.net】就是“家的地址”【域名】，就是这个服务器在互联网的位置；【api/province】是“物品的地址”【资源路径】，这是资源所在这个服务器的位置

(3)、那还有这种：“https://music.163.com/#/song?id=1814904616”

   这里【song?id=1814904616】是啥玩意？这是URL的查询参数，标准格式：http://xxxx.com/xxx/xxx?参数名1=值1&参数名2=值2

     比如这个“https://hmajax.itheima.net/api/city?pname=河北省”，我们只想访问的是河北省数据而不是所有省份，【city?pname=河北省】就传入参数【pname】

四、请求配置和提交数据

提交一般两种方式：post提交、get获取数据（不过method默认就是get，可以省略）

注意：post搭配data和method提交，而get搭配params获取

格式：axios({

url: ‘要访问的服务器网站’,

data: {

参数名 : 值

},

method: ‘请求方法’

}).then(result => {

要做什么事

})

例子：

     axios({

        url: 'https://hmajax.itheima.net/api/register',  //用户注册信息录入的服务器网站

//请求的方式

        method: 'post',

//要传的参数值

        data: {

            username : 'itheima110', //该网站设置的获取account账号的参数名

            password : '7654321' //该网站设置的获取密码的参数名

        }

    }).then(result=>{

        console.log(result)//检查一下

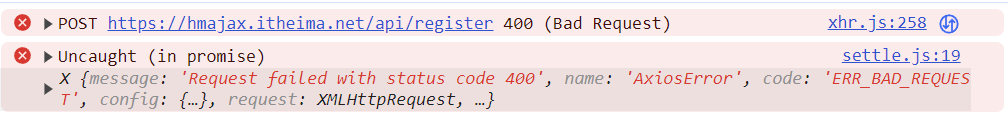
    })

结果：



五、捕捉报错信息

But，当我们这个账户已经注册了，再次注册时axios库就是捕捉到这个错误并及时返回“错误提示”：



其中：



然后具体的错误信息可以在这看到：



但是我们程序员看得明白，普通用户并不知道，要通过我们处理成直观的报错

那么axios的报错方式就是.catch{......}

格式：axios({

//请求选项

}).then(result => {

//处理数据

}).catch(error=>{

//处理错误

})

例子：

axios({

            url: 'https://hmajax.itheima.net/api/register',

            //请求的方式

            method: 'post',

            //要传的参数值

            data: {

                username : 'itheima110',//这个账号我之前已经注册过了

                password : '7654321'

            }

        }).then(result=>{

            //成功

            //处理数据

        }).catch(error=>{

            //失败报错

            console.log(error)

//检查一下报错的对象有什么信息

            console.log(error.response)

//我们发现报错信息在response这个对象里，再检查一下

            console.log(error.response.data)

//我们有发现具体报错信息在response对象的data数组里

            alert(error.response.data.message)

//那么用弹窗的形式将data里的message字符串信息输出，更加直观

        })

1. HTTP协议，请求报文（浏览器发给服务器）



第一行（请求行）：请求方式、URL网址、协议

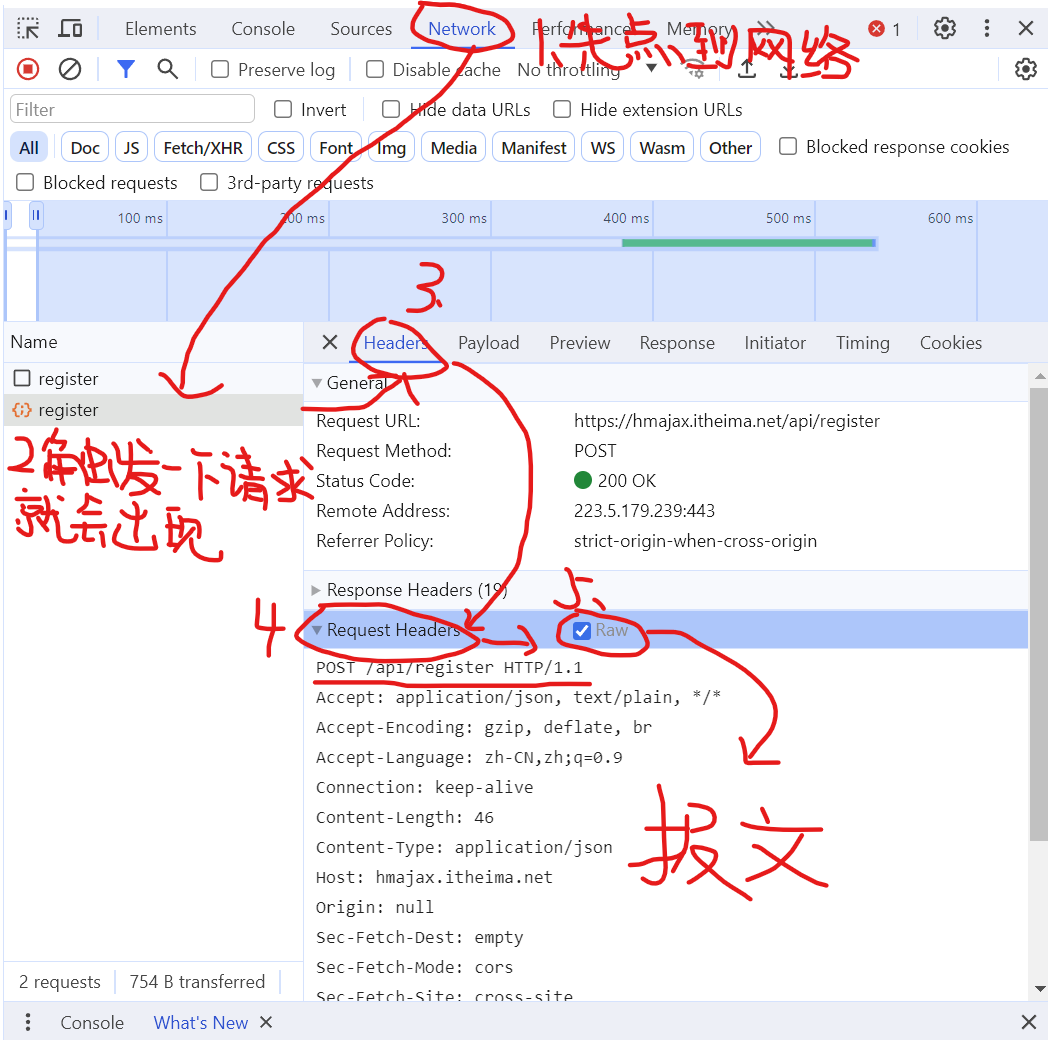
中间（请求头）：以键值对的格式携带的附加信息，图中Content-Type:application/json就是意思规定要按照json文件形式与服务器交换数据资源

空行：分隔请求头，空行之后是发送给服务器的资源

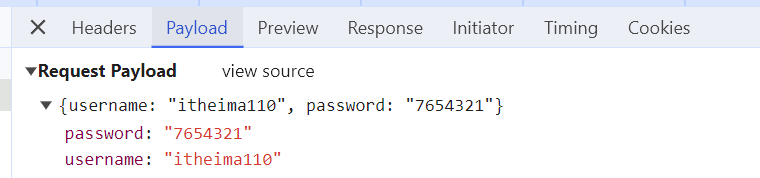
最后一行（请求体）：发送的资源



去浏览器哪里看：



那么请求体去哪看？在旁边



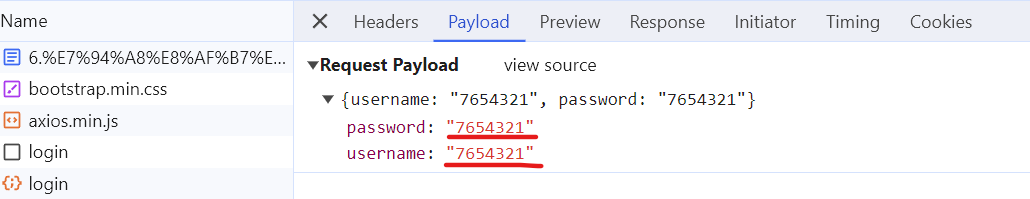
如何利用“请求报文”查看自己的错误？

例子：

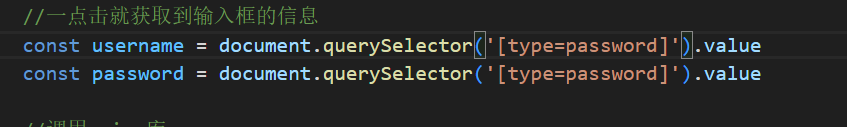
用登陆界面做例子，当我们输入正确的账号密码确报错



那就到请求报文那看看传入资源有无问题：



发先账号密码都一样，找到代码处：



这样修改一下就好了，全部代码在这：

<!-- 第一步：引入axios库 -->

    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

    <script>

        //思路：第一步先获取两个元素，提示框元素是用来随时更改样式的，按钮元素是为了点击用的

        const Alert = document.querySelector('.alert')

        const Button = document.querySelector('button')

    //第二步：给提示框改变样式设置对应的函数，然后就可以在按钮点击事件的时候调用它了

    //这个函数需要传两个参数：1、从服务器返回的信息“是否有登录成功？”

    //                      2、成功和失败情况各自对应的bootstrap里的对应【样式】的“类名”

        function showAlert(message,bootstrap\_className){

            //首先让它的文字内容变成“服务器返回的‘是否登陆成功’的信息”

            Alert.innerHTML = message

            //然后把成功和失败情况各自对应的bootstrap里的对应【样式】的“类名”赋给它

            Alert.classList.add(bootstrap\_className)

            //最后，因为前面css部分，我们把提示框设置成了透明不显示，现在要把它改变后的样式

显示出来，就要添加上我们已经准备好的.show类名的样式

            Alert.classList.add('show')

            //最后最后，不能一直显示这玩意，出现个两秒提示就够了

            setTimeout(function(){

                //把bootstrap的样式取消掉

                Alert.classList.remove(bootstrap\_className)

                //把css的show样式取消掉，就透明了

                Alert.classList.remove('show')

            },2000)

        }

        Button.addEventListener('click',function(){

            //一点击就获取到输入框的信息

            const username = document.querySelector('[type=text]').value

            const password = document.querySelector('[type=password]').value

            //调用axios库

            axios({

                url: 'https://hmajax.itheima.net/api/login',

                method: 'post',

                data: {

                    // Uname : username,

                    // Psw : password

                    //这里我除了一个错误，因为在该服务器的axios库里固定的“账号参数名”是

username，而“密码参数名”是password，而我自己自定义的参数名人家认不得

                    // username,这样也可以

                    // password

                    username : username,

                    password : password

                }

            }).then(result => {

                //注意，思维转换一下，不需要我们自己判断字符串是否匹配，数据提交上去后服务器

自己会判断，并返回对应的信息【message】

                //成功的情况，信息将在当前对象的data的message里

                showAlert(result.data.message,'alert-success')

                console.log(result)

            }).catch(error => {

                //不成功的情况，信息则是在当前对象的response的data的message里

                showAlert(error.response.data.message,'alert-danger')

            })

        })

    </script>

1. 响应报文（服务器发给浏览器）

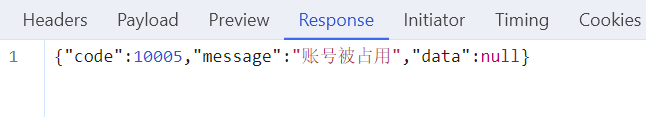


第一行（请求行）：协议、HTTP响应状态码、响应状态

中间（请求头）：以键值对的格式携带的附加信息，比如Content-Type

空行：分隔请求头，空行之后是返回给浏览器的资源

最后一行（请求体）：返回的资源



HTTP响应状态码：来表名请求是否成功

比如：404 服务器找不到资源

常见的几种：

1、1xx 信息

2、2xx 成功

3、3xx 重定向消息

4、4xx 客户端错误

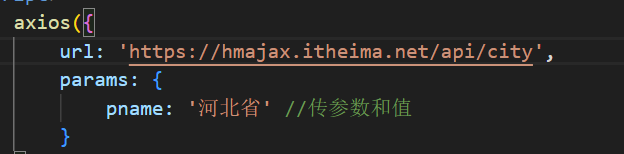
5、5xx 服务端错误

接口文档：后端工程师对接口描述的文章

（黑马Ajax接口文档网址：<https://apifox.com/apidoc/project-1937884）>

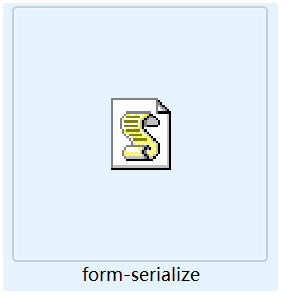


接口：用Ajax与服务器连接时所需要的URL、请求方式、参数



1. 有一个js插件叫：form-serialize，可以一次性获取所有表单数据

必须要有这个文件：



1. 然后第一步是引入这个文件

    <!-- 1、要用form-serialize必须先引入特有的JS文件，我已放在Ajax当前文件夹 -->

    <script src="./form-serialize.js"></script>

1. 获取页面的form表单元素

// 2、获取from表单元素先

        const form = document.querySelector('form')

1. 调用serialize()函数

const data = serialize(form, { hash: true, empty: true })

注意：

有两个参数需要传： 表单对象 、配置对象

1. 表单对象：注意一点，尽量让该表单里input输入框的name和接口文档里需要的参数名

字一模一样，因为后台认数据并封装进data对象里时，就是认【name】

1. 配置对象：

hash：传输的数据结构

~当是true时，传输JS数据【对象】，推荐，因为axios底层原理就是把像data、params这些对象自动转化成json文件的字符串提交服务器

~当是false时，传输的就是【查询字符串】："username=yjtlwkbz&password=7654321"类似这样的，就是URL的？后面那个玩意，但是axios库需要的是对象

empty：是否获取空值，true就是，false就不是，有脑子就能理解

1. 是获取到了表单数据，但是是对象数据，怎么分出里面的每个参数数据传参给axios库？

【解构】：定义 { 成员1 , 成员2 ......} = 对象

//利用【解构】对象：

const { username , password } = data //这样就反向把对象的值给到{}里的两个成员了

结果能看到，能获取值，也能提交给服务器，并判断数据正确否

