[AJAX+AXIOS+

WebSocket]

[2021.9.27]

文档标题作为文字占位，只需单击占位更换文本添加内容

摘要

[前后端数据交互]

[日期及邮编地址]

[ID名称]

[原生Ajax（异步javascript和xml） 4](#_Toc7194)

[服务器软件： 4](#_Toc4192)

[ajax也分get和post 4](#_Toc11856)

[ajax前端部分 4](#_Toc21675)

[ajax后端部分 4](#_Toc20423)

[ajax流程 4](#_Toc2014)

[XMLHttpRequest对象 4](#_Toc18310)

[//创建xmlhttprequest对象 5](#_Toc25653)

[//设置请求行，参数1：提交方式：get/post，参数2：url，参数3：是否为异步，默认为true异步 5](#_Toc230)

[//回调函数用onreadystatechange①所有浏览器都支持 5](#_Toc390)

[//发送请求报文，若方式为post则写入键值对，否则为空或null 5](#_Toc11965)

[前端使用ajax 5](#_Toc16450)

[发送数据之get请求 5](#_Toc3298)

[发送数据之post请求 5](#_Toc32105)

[接收数据 6](#_Toc23248)

[1. xml格式：xxx.xml，相当于一个document 6](#_Toc24345)

[XML缺点： 7](#_Toc16502)

[2. json格式：xxx.json 7](#_Toc18083)

[后端（Django）使用Ajax 8](#_Toc5753)

[接收数据 8](#_Toc7250)

[简单数据： 8](#_Toc5478)

[复杂数据：json 8](#_Toc2622)

[返回数据 8](#_Toc19375)

[简单数据：使用HttpResponse(‘xxx’)直接返回，如单个字符串，数字 8](#_Toc9953)

[json数据（常用）：多组数据，json格式在python中相当于字典 8](#_Toc30829)

[方法1：json.dumps() 8](#_Toc21843)

[方法2：JsonResponse() 8](#_Toc6450)

[方法3：serializers序列化（重点） 9](#_Toc4176)

[ajax跳转页面 9](#_Toc11126)

[获取其他网站的ajax数据 9](#_Toc15986)

[ajax封装 9](#_Toc26212)

[跨域问题（跟ajax无关） 10](#_Toc24908)

[jQuery的Ajax 10](#_Toc6159)

[axios的Ajax 10](#_Toc15940)

[axios简介 10](#_Toc26970)

[安装和导入axios 11](#_Toc22371)

[html文件内导入axios 11](#_Toc9754)

[组件内导入axios 11](#_Toc31599)

[axios使用 11](#_Toc21299)

[axios传参（get、post） 11](#_Toc12300)

[获取axios接收到的数据（结合async和await） 12](#_Toc27333)

[axios.get() 13](#_Toc3514)

[axios.post() 13](#_Toc5800)

[websocket 13](#_Toc9448)

[websocket简介 13](#_Toc17341)

[websocket使用 14](#_Toc31858)

原生Ajax（异步javascript和xml）

服务器软件：

apache：开源的web服务器软件，免费，目前使用人数最多

IIS：微软的web服务器，之前不开源，目前开源

tomcat：web服务器软件

Nginx：web服务器软件

ajax也分get和post

后端决定ajax提交方式（get/post）、数据变量名

前端写请求报文

ajax前端部分

1. 注册事件函数
2. 创建XMLHttpRequest对象
3. 设置请求行
4. 设置请求头
5. 注册回调函数
6. 发送请求

ajax后端部分

1. 接受提交的数据
2. 处理请求
3. 返回响应报文

ajax本质：使用js和xml异步发送http请求和处理返回的数据

通过触发js事件函数，创建xmlhttprequest对象（该对象模拟http协议的get和post请求）发送该模拟请求报文到服务器，服务器处理完数据发送响应报文回浏览器，再通过js回调函数处理返回的数据展示到浏览器上

ajax流程

前端：

发送ajax请求

后端：

读取

返回格式化的数据（如xml、json）

前端：

回调函数解析并展示数据

XMLHttpRequest对象

//创建xmlhttprequest对象

var xhr=XMLHttpRequest();

//设置请求行，参数1：提交方式：get/post，参数2：url，参数3：是否为异步，默认为true异步

xhr.open(‘get/post’, ‘url?key=value/url&key2=value2’, true/false);

//回调函数用onreadystatechange**①**所有浏览器都支持

xhr.onreadystatechange = function() {

if (xhr.readyState==4 && xhr.statu==200) {

数据处理代码;

}

}

//发送请求报文，若方式为post则写入键值对，否则为空或null

xhr.send(null/ ’key=value&key2=value2’);

注：①onreadystatechange事件所有浏览器都支持，但是需要写判断，readyState状态码分为五种：0/1/2/3/4，其中4为完成状态，所以判断if xhr.readyState==4，但是只判断readyState还不够，还需要判断xhr.status==200（页面存在且成功连接）再执行数据处理代码

前端使用ajax

发送数据之get请求

document.querySelector(‘xxx’).onclick = function() { //请求触发事件函数

var xhr=new XMLHttpRequest(); //创建异步对象

xhr.open(‘get’, ’url?name=xxx&age=20’); //设置请求行，get，get请求将数据拼接 在url中

xhr.setRequestHeader(‘cookie’, ‘xxxxxxxx’); //请求头setRequestHeader，‘参数1’， ‘值1’，get请求可以省略设置请求头， post不发数据也可省略

xhr.send(null) //请求主体发送，不发送数据直接空或者null

xhr.onload=function() { //注册回调函数，响应回来后触发，处理返回的数据

console.log(‘请求响应回来了’);

xhr.responseText; //取用返回的数据，字符串格式

}

}

发送数据之post请求

document.querySelector(‘xxx’).onclick = function() { //请求触发事件函数

var xhr=new XMLHttpRequest(); //创建异步对象

xhr.open(‘post’, ‘url’); //设置请求行，post，url

xhr.setRequestHeader(‘Content-Type’, ‘application/x-www-form-urlencoded’);

//设置请求头，让ajax像html表单那样发送数据

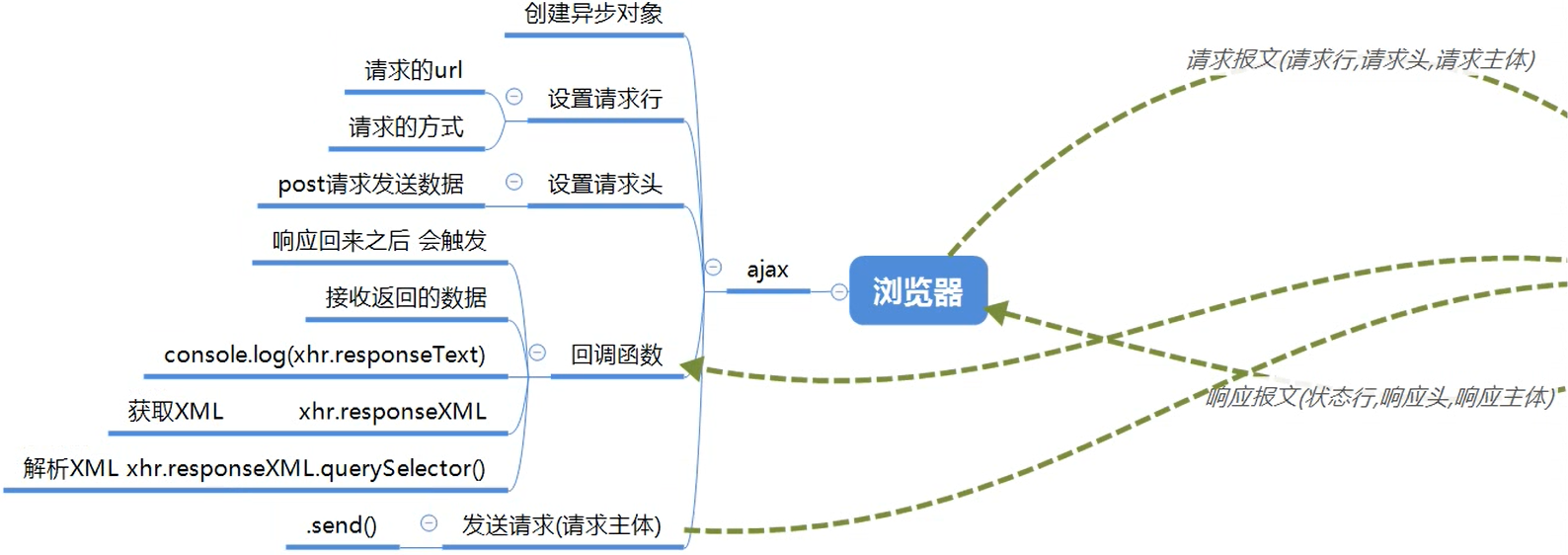
xhr.send(‘key=value&key2=value2’); //请求主体发送，将数据写在send中，格式为 字符串‘key=value&key2=value2’内容不要加空格

xhr.onload = function() { //注册回调函数，处理返回的数据

console.log(‘xxx’);

}

}



ajax流程图

接收数据

1. xml格式：xxx.xml，相当于一个document

语法：

<?xml version=’1.0’ encoding=’UTF-8’> //声明，可不写

<root> //根标签，必须有，名字自定义

<name>xxx</name>

<age>20</age>

...

</root>

注：

1. xml内标签必须成对
2. 标签名随意，但是最好有意义，类似于编程语言内的变量名

前端解析xml文件：

前端js操作xml文件类似js对html元素的操作，选取标签元素，读取标签内容获取数据

1. xhr.responseXML.querySelector(‘xxx’)
2. var xxx=标签.innerHTML //对于大量数据需要使用循环进行解析
3. 创建标签来装解析出来的数据

XML缺点：

传输的数据量大，但有效数据比不高

解析复杂

1. json格式：xxx.json

json简介：

json全称JavaScript Object Notion，发源于JS，是一种超轻量级数据交互格式。

json规定字符集是UTF-8，字符串必须使用双引号””，object的键也必须使用双引号””

数组或对象的最后一个成员不能加逗号

1. json的载体是字符串，解析数据用
2. 基本上所有的编程语言都支持json
3. 语法简洁，基本上所有编程语言都提供了方法解析json
4. json格式的字符串转化完毕之后会变成数组对象

5>var xxx=JSON.parse(xhr.responseText); //前端转换后端返回的json数据的代码

6>json格式在js中相当于object对象

json写法1：对象格式

字符串格式：用 ’ ’包裹

对象使用 { }

键名必须使用 双引号” ”包裹（类似python的字典）

属性值必须使用双引号 ” ”包裹

键值对形式 ”key”: ”value”

xhr.responseText=’{“key1”: “value1”, “key2”: ”value2”}’;

转化为对应的对象或数组

var xxx=JSON.parse(xhr.responseText); //前端转换后端返回的json数据的代码

取用数据

obj.key1;

json写法2：数组格式

var jsonarr = ’[“value1”, ”value2”, “value3”]’;

转化为对应的数组

var arr = JSON.parse(jsonarr);

取用数据

arr[x];

json写法3：对象数组混合

var jsonobjarr = ‘{“key1”: [“value1”, “value2”, “value3”]}’;

var objarr = JSON.parse(jsonobjarr);

或

var jsonobjarr = ‘[{“key1”: “value1”}, {“key2”: “value2”}]’;

var objarr = JSON.parse(jsonobjarr);

json文件：xxx.json

语法：

[

{

“xxx”: “value”,

“xxx2”: “value2”,

},

{

“xxx”: “value”,

“xxx2”: “value2”,

}

]

注：对于后端传来的json格式数据，有的需要使用JSON.parse()转换成json，有的不需要，原则：若传过来的本身就是正常的json格式，如{‘key1’: xxx}，就不用JSON.parse()，若含有字符串格式的json数据，则需要JSON.parse()。

后端（Django）使用Ajax

接收数据

简单数据：

前端：ajax发get/post类型请求，简单数据格式为’key1=value1&key2=value2’

后端：django接收get/post简单数据时直接使用request.GET.get(‘xxx’)、request.POST.get(‘xxx’)

复杂数据：json

前端：发送post请求，格式为json：var xxx=”{‘key1’: ‘nnn’, ‘key2’:’nnn’}”

后端：接收使用request.GET.get(‘xxx’)、request.POST.get(‘xxx’)

返回数据

简单数据：使用HttpResponse(‘xxx’)直接返回，如单个字符串，数字

json数据（常用）：多组数据，json格式在python中相当于字典

方法1：json.dumps()

导入json库，使用json.dumps()处理数据，再用HttpResponse()返回数据

可以转换处理后的数组、元组、字典数据为json字符串

import json

jsondata=json.dumps(obj) # obj为python对象，如dict、list、tup(元组)

return HttpResponse(jsondata,)

方法2：JsonResponse()

导入JsonResponse类，使用JsonResponse({‘key1’: ‘xxx’, ‘key2’: ‘xxx’})

from django.http import JsonResponse #JsonResponse是HttpRespsonse的子类

jsondata={‘key1’: ‘xxx’, ‘key2’: ‘xxx’} #必须是字典格式的json数据

return JsonResponse(jsondata)

方法3：serializers序列化（重点）

导入serializers模块，使用serializers.serializer(QuerySet数据)，然后直接用HttpResponse返回结果即可

from django.core import serializers

data = xxx.objects.filter(...)

backjson = serializers.serializer(‘json’, data)

return HttpResponse(backjson)

注：该方法只能序列化model对象，即用get、filter直接查询出的对象，不能序列化dict对象，如.values()读取的部分字段数据

ajax跳转页面

返回url相对路径数据给前端，例’/mailbx/main/’，前端通过window.location.href=’url’跳转页面。

获取其他网站的ajax数据

开发者界面→network→XHR选项→挨个点开看看→复制到json格式化网站检验

ajax封装

将相同的部分写死，不同的部分写成参数：url、data、type、回调函数。

只传递一个参数：将参数写成对象格式：option.url、option.data、option.type、option.回调函数

判断数据类型：通过xhr.getResponseHeader(‘Content-Type’)判断回传的数据是何种格式（xml、json、普通字符串）工具函数：字符串. indexOf(‘xxx’)，没有包含’xxx’，则返回-1

var type = xhr.getResponseHeader(‘Content-Type’);

// 是否为json

if (type.indexOf(‘json’) != -1) {

回调函数(JSON.parse(xhr.responseText));

}

// 是否为xml

else if (type.indexOf(‘xml’) != -1) {

回调函数(xhr.responseXML);

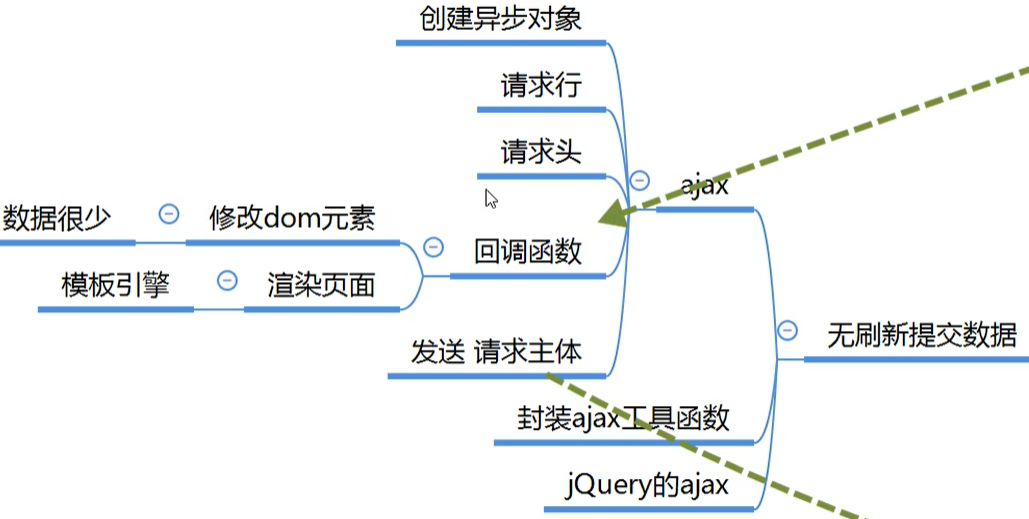
}

// 普通字符串

else {

回调函数(xhr.responseText);

}

ajax数据多时用模板

跨域问题（跟ajax无关）

同源：协议名一样，地址一样，端口号一样称之为同源，即同一服务器内的同一文件夹下

不同源：上述三条存在一条不一样即为不同源

跨域：不同源的网站之间发送请求称为跨域，浏览器默认限制跨域访问

解决办法：

1. cors（cross origin resource sharing）html5才支持，最新方法，未来将普及，需在接收请求方的服务器端设置请求头’Access-Control-Allow-Origin: \*’
2. jsonp（json with padding）dom元素的src属性支持跨域获取资源，jsonp就是利用<script>标签的src属性支持跨域获取资源，跟ajax没关系。老方法，只能用get，存在安全问题，未来将被取代
3. 先写好一个用来处理返回的数据的callback函数
4. 前端：<script src=’url?callback=xxx’></script>
5. 服务器收到请求后拼接数据和函数名字符串返回给前端

fcname=get(‘callback’)

return fcname+’(’+data+’)’

4>前端收到响应报文后会将返回的数据按js代码解析执行

jQuery的Ajax

$.ajax()

axios的Ajax

axios简介

axios是一个专注于网络请求的js库，vue教程的P76-P79学到

安装和导入axios

下载axios.js文件，并在要使用的文件内导入axios.js

用npm安装axios

npm install axios -S

html文件内导入axios

每个使用axios 的html文件都要导入一次

<script src=’../xxx/axios.js’></script>

组件内导入axios

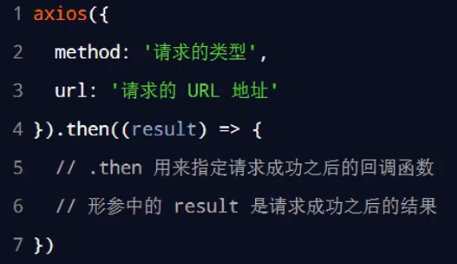
每个使用axios的组件都要导入一遍

<script>

import axios from ‘axios’ //通过npm安装的这么写即可

</script>

axios使用



返回值是一个promise对象，后面可以直接加 .then()

axios传参（get、post）

axios({

method: ‘get’,

url: ‘xxxxx’,

params: { //get用params传参

key: ‘xxx’,

}

})

axios({

method: ‘post’,

url: ‘xxxxx’,

data: { //post用data传参

key: ‘xxx’,

}

})

获取axios接收到的数据（结合async和await）

用来简化axios返回值的处理步骤

前提：包裹axios的函数需要加上async关键字

async get\_data() {

const result = await axios({

method: ‘get’,

url: ‘xxxxxx’,

params: {

key: ‘xxx’,

}

})

}

此时result就是返回的对象包含六个属性，而不是一个完整的promise对象，其中data属性是真正需要的数据，还可以通过解构赋值简化数据对象

async get\_data() {

const {data: res} = await axios({ //解构赋值，data: res是将data重命名为res

method: ‘get’,

url: ‘xxxxxx’,

params: {

key: ‘xxx’,

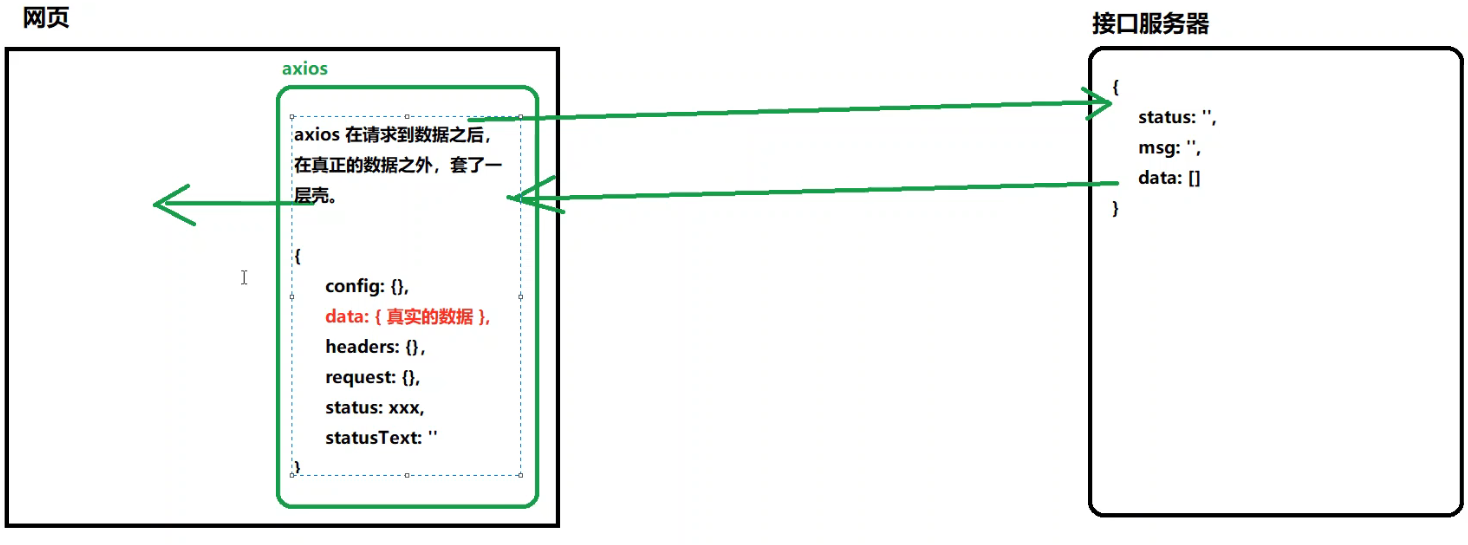
}

})

}

console.log(res.data);

此时data是服务器返回的数据



axios.get()

axios的get版本

axios.get(‘url’, {

params: {

key: ‘xxx’, //在params对象里写键值对

}

})

axios.post()

axios的post版本

axios.post(‘url’, {

key: ‘xxx’, //post直接在对象里面写键值对就可以了

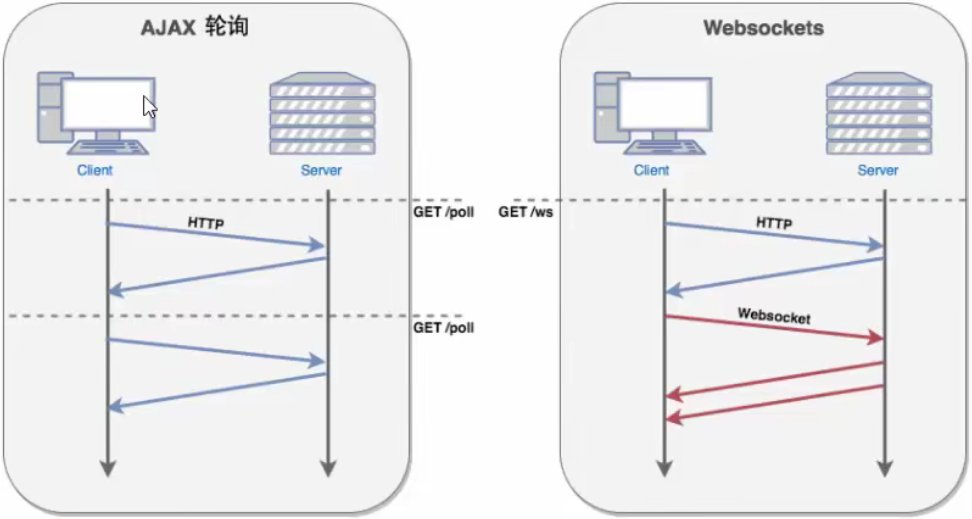
})

websocket

websocket简介

websocket协议是基于TCP的一种新的网络协议，它实现了浏览器与服务器全双工通信——允许服务器主动发消息给浏览器。

websocket是一种持久协议，http是非持久协议。没有websocket协议的时候靠ajax轮询实现相同效果



websocket使用

websocket在H5中加入了API，可以直接使用

1. 创建websocket对象

var socket = new websocket(‘url’, [protocol]);

1. websocket事件



1. websocket方法

socket.send() //使用连接发送数据

socket.close() //关闭连接