Ajax（Asynchronous(异步)JavaScript and XML）是一种用于创建快速动态网页的技术。并非编程语言。

AJAX 仅仅组合了：

浏览器内建的 XMLHttpRequest 对象（从 web 服务器请求数据）

JavaScript 和 HTML DOM（显示或使用数据）

Ajax 是一个令人误导的名称。Ajax 应用程序可能使用 XML 来传输数据，但将数据作为纯文本或 JSON 文本传输也同样常见。

Ajax 允许通过与场景后面的 Web 服务器交换数据来异步更新网页。这意味着可以更新网页的部分，而不需要重新加载整个页面。

Ajax如何工作:

网页中发生一个事件（页面加载、按钮点击）

由 JavaScript 创建 XMLHttpRequest 对象

XMLHttpRequest 对象向 web 服务器发送请求

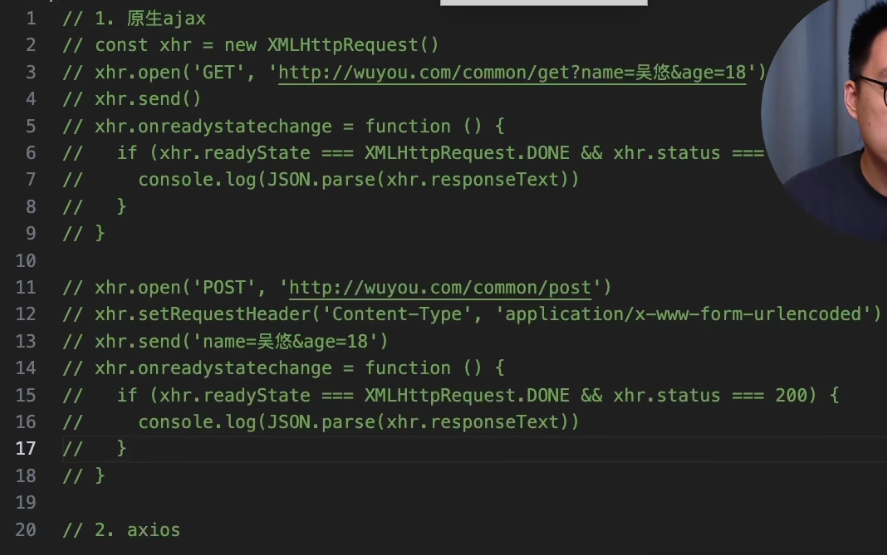
服务器处理该请求

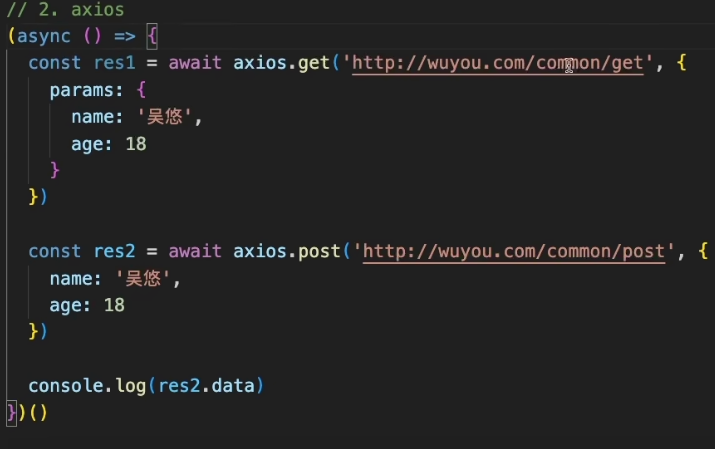
服务器将响应发送回网页

由 JavaScript 读取响应

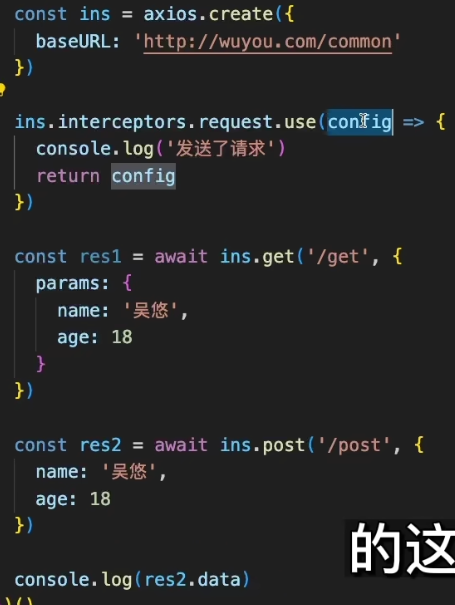
由 JavaScript 执行正确的动作（比如更新页面）

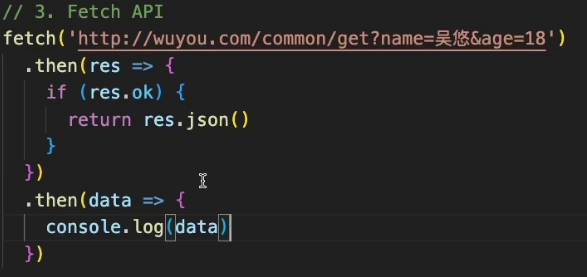
Ajax数据请求

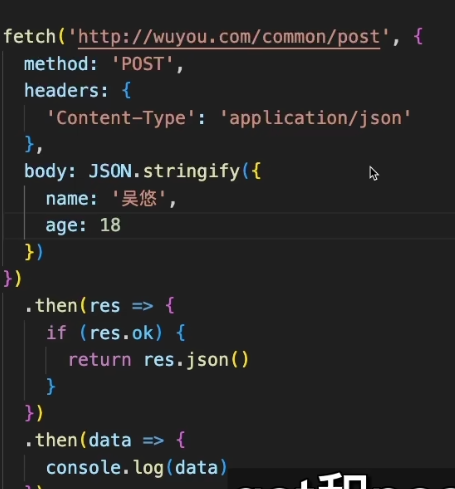




拼接 创建基础ins,然后拼接







JSON.stringify和parse是相互转换的

Axios 是一个基于 [promise](https://javascript.info/promise-basics" \t "/Users/liujiongzhi/Documents\\x/_blank) 网络请求库，作用于[node.js](https://nodejs.org/" \t "/Users/liujiongzhi/Documents\\x/_blank) 和浏览器中。

axios.get('/user?ID=12345')  
.then(function (response) {  
// 处理成功情况  
console.log(response);  
})  
.catch(function (error) {  
// 处理错误情况  
console.log(error);  
})  
.finally(function () {  
// 总是会执行  
});  
  
// 上述请求也可以按以下方式完成（可选）  
axios.get('/user', {  
params: {  
ID: 12345  
}  
})  
.then(function (response) {  
console.log(response);  
})  
.catch(function (error) {  
console.log(error);  
})  
.finally(function () {  
// 总是会执行  
});   
  
// 支持async/await用法  
async function getUser() {  
try {  
const response = await axios.get('/user?ID=12345');  
console.log(response);  
} catch (error) {  
console.error(error);  
}  
}

try {  
 let response = await fetch("www.baidu.com");  
 let res = await response.json();  
 console.log(res);  
} catch (error) {  
 console.log(error);  
} finally {  
 setLoading(false);  
 console.log("结束");  
}

获取url参数方法1

window.location.search:是js里面window.location对象的一个属性,获取url查询参数部分,包括问号?,可用.split(1)去掉?

http://www.baidu.com/path/to/page?name=dasd

js标准库里面有URLSearchParams这个内置构造函数,输入去掉?的字符串后可以返回一个对象用于操作和查询

const queryParams=new URLSearchParams(“name=john&age=18”)

方法有:

1. queryParams.append(name,value) 增
2. queryParams.delete(name) 删
3. queryParams.set(name,value) 改,覆盖,没有就是添加
4. queryParams.get(name) 查
5. queryParams.getAll(name) 查,获取指定名称的所有查询参数的值,返回一个数组
6. queryParams.has(name) 检查是否存在指定名称的查询参数
7. queryParams.sort( ) 按照名称对查询参数排序
8. queryParams.toString( ) 再次转成经过URL编码的查询字符串,且会处理特殊字符

获取url参数方法2--使用正则表达式

已知url=window.location.href.split(“?”)[1]

const params=/[?&]([^=#])=([ ^&#]\*)/g

解释:

1.[?&]表示匹配?或者&,表示参数开始或者之间的分隔符

2.不匹配#号是不包括哈希部分

3.([^=#])匹配参数名,用()捕获

4.匹配参数值,不包括#或者&的字符,使用()捕获

5.g是全局匹配标志

然后

const params={}//创建对象存储匹配的参数

let match;

while ((match=regex.exec(url))!==null){

Params[decodeURIComponent(match[1])]=decodeURIComponent(match[2]);}

解释:

1. 在循环中,使用exec方法执行正则表达式的匹配,然后将匹配到的参数名和参数值存储params对象中,最终params对象将包含url的所有查询参数
2. Exec方法如果匹配成功,返回一个数组,第一个元素是整个匹配的字符串,其余的元素是与正则表达式的捕获组匹配的部分,如果没有匹配,exec返回null
3. 对于exec方法,匹配到第一个方法就会返回,然后将正则表达式对象的lastIndex属性设置为下一个匹配的起始位置
4. decodeURIComponent是js的一个全局函数,用于解码由encodeURIComponent函数编码的url组件,得到原始字符串如%20表示空格 %2F表示斜杠

window.location.port获取端口,如果没有明确指定端口号,将返回空字符串,这种情况下需要检查是否为空,然后使用默认端口号

http://www.baidu.com:80/page?name=dasd

window.location.pathname 用于获取当前界面url中的路径部分

http://www.baidu.com/path/to/page?name=dasd

window.location.hostname 返回主机名部分,不包括端口

http://www.baidu.com:8080/path/to/page?name=dasd

window.location.host返回主机名部分,包括域名和端口

http://www.baidu.com:8080/path/to/page?name=dasd