

## 题目

- ◆ 手写一个简易的 ajax
- ◆ 跨域的常用实现方式

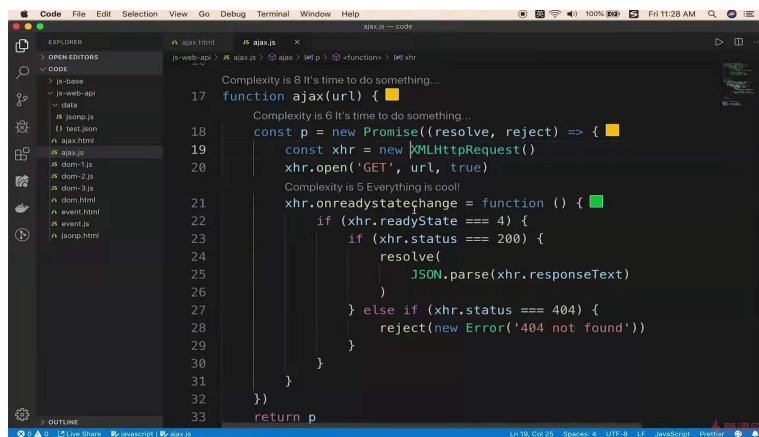
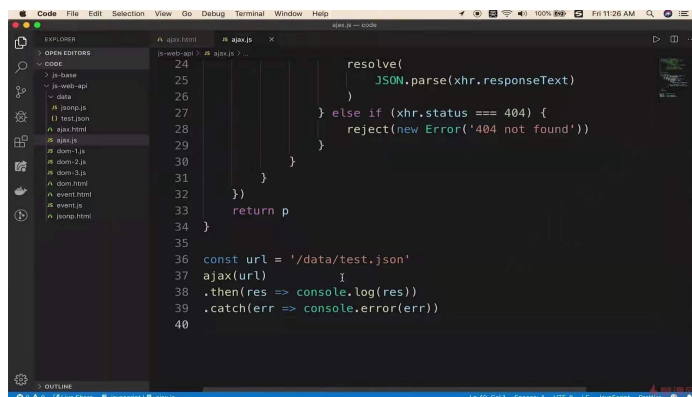
慕课网

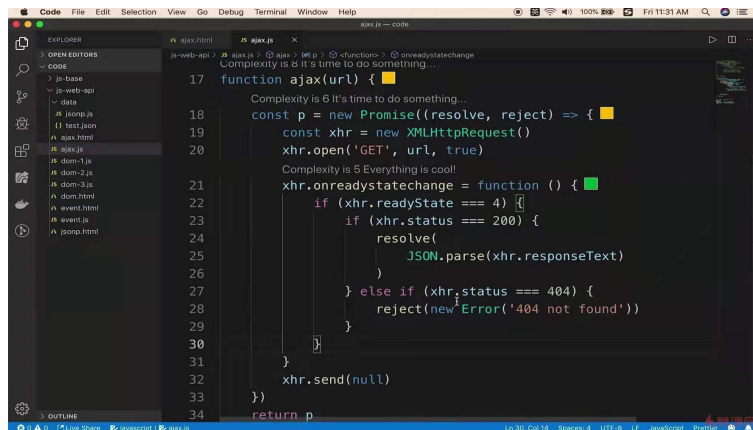
## 手写一个简易的 ajax

```
function ajax(url, successFn) {
  const xhr = new XMLHttpRequest()
  xhr.open("GET", url, true)
  xhr.onreadystatechange = function () {
    if (xhr.readyState === 4) {
      if (xhr.status === 200) {
        successFn(xhr.responseText)
      }
    }
  }
  xhr.send(null)
}
```

慕课网

慕课网

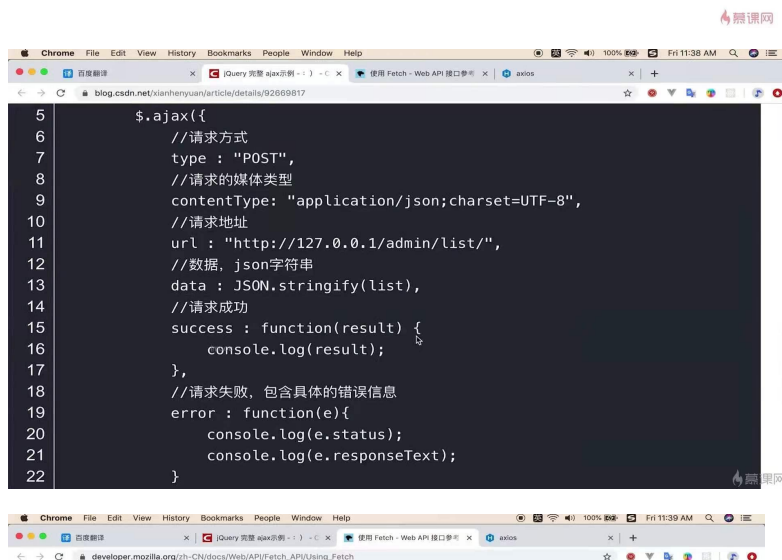




```
17 function ajax(url) {
18     // Complexity is 6 It's time to do something...
19     const p = new Promise((resolve, reject) => {
20         const xhr = new XMLHttpRequest()
21         xhr.open('GET', url, true)
22         // Complexity is 5 Everything is cool!
23         xhr.onreadystatechange = function () {
24             if (xhr.readyState === 4) {
25                 if (xhr.status === 200) {
26                     resolve(
27                         JSON.parse(xhr.responseText)
28                     )
29                 } else if (xhr.status === 404) {
30                     reject(new Error('404 not found'))
31                 }
32             }
33         }
34         xhr.send(null)
35     })
36     return p
37 }
```

## 知识点

- ◆ XMLHttpRequest
- ◆ 状态码：readyState status
- ◆ 跨域：同源策略（如何绕过），JSONP，CORS



```
5 $.ajax({
6     //请求方式
7     type : "POST",
8     //请求的媒体类型
9     contentType: "application/json;charset=UTF-8",
10    //请求地址
11    url : "http://127.0.0.1/admin/list/",
12    //数据, json字符串
13    data : JSON.stringify(list),
14    //请求成功
15    success : function(result) {
16        console.log(result);
17    },
18    //请求失败, 包含具体的错误信息
19    error : function(e){
20        console.log(e.status);
21        console.log(e.responseText);
22    }
23 })
```

一个基本的 fetch请求设置起来很简单。看看下面的代码：



```
1 fetch('http://example.com/movies.json')
2 .then(function(response) {
3     return response.json();
4 })
5 .then(function(myJson) {
6     console.log(myJson);
7 });
```

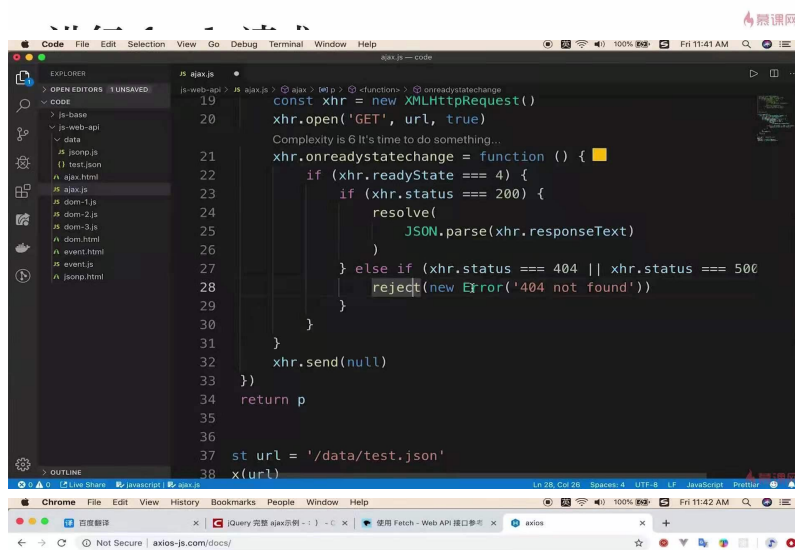
这里我们通过网络获取一个JSON文件并将其打印到控制台。最简单的用法是只提供一个参数用来指明想 fetch() 到的资源路径，然后返回一个包含响应结果的promise(一个 Response 对象)。

当然它只是一个 HTTP 响应，而不是真的JSON。为了获取JSON的内容，我们需要使



请注意，`fetch` 规范与 `jQuery.ajax()` 主要有两种方式的不同，牢记：

- 当接收到一个代表错误的 HTTP 状态码时，从 `fetch()` 返回的 Promise 不会被标记为 **reject**，即使该 HTTP 响应的状态码是 404 或 500。相反，它会将 Promise 状态标记为 **resolve**（但是会将 `resolve` 的返回值的 `ok` 属性设置为 `false`），仅当网络故障时或请求被阻止时，才会标记为 `reject`。
- 默认情况下，`fetch` 不会从服务端发送或接收任何 **cookies**，如果站点依赖于用户 session，则会导致未经认证的请求（要发送 cookies，必须设置 `credentials` 选项）。自从 2017 年 8 月 25 日后，默认的 `credentials` 政策变更为 `same-origin`。Firefox 也在 61.0b13 中改变默认值。



## axios #

npm v0.19.0  
build passing  
coverage 94%  
install size 409 kB  
downloads 25M/month  
chat on glitter  
code helpers 113

Promise based HTTP client for the browser and node.js

### Features

- Make `XMLHttpRequests` from the browser
- Make `http` requests from node.js

Requ

axi

axi

axi

axi

axi

axi

axi

axi

axi

axi

axi

axi

axi

