

❗此MD/PDF/HTML文档由 [@竹梦者也](#) 编辑制作，不可贩卖 ❌。仅供学习交流使用，下载后切勿随意传播。转载麻烦加上出处~

👉 欢迎在留言区提供修改建议~

📁 本文内容和代码来自于 [李强](#) 老师的 [axios](#) 教程

✅ 2021.7.17 @竹梦者也(<https://www.zjgsuzjx.top>)

# 一、axios入门

## 一、前期准备工作

### 1. 安装

快速安装虚拟服务器: <https://github.com/typicode/json-server>

安装axios: <https://github.com/axios/axios>

### 2. 虚拟服务器说明

```
1 获取: GET /posts
2  GET /posts/1
3 添加: POST /posts
4 更新: PUT /posts/1
5  PATCH /posts/1
6 删除: DELETE /posts/1
```

创建 `db.json` 至文件夹根目录:

```
1 {
2   "posts": [
3     { "id": 1, "title": "json-server", "author": "typicode" }
4   ],
5   "comments": [
6     { "id": 1, "body": "some comment", "postId": 1 }
7   ],
8   "profile": { "name": "typicode" }
9 }
```

npm安装: `npm install -g json-server`

启动服务器: `json-server --watch db.json`

## 二、axios 的理解和使用

### 1. 概念

axios 前端最流行的 ajax 请求库, react / vue 官方都推荐使用 axios 发 ajax 请求。

官方中文文档: <https://axios-http.com/zh/docs/intro>

### 2. 特点

1. 基于 xhr + promise 的异步 ajax 请求库
2. 浏览器端 / node 端都可以使用
3. 支持请求 / 响应拦截器
4. 支持请求取消
5. 请求/响应数据转换
6. 批量发送多个请求

### 3. 引入

使用时:

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

研究源码时: `npm install axios`

## 4. 快速使用

注意：下述方法都是基于前面提到的虚拟服务器环境。

基本语法格式：`axios({请求类型}).then(response => {响应结果})`

和Promise语法基本一致。

请求类型里可以填入 `method`、`url`、`data` 等配置。

响应内容 `response` 是一个对象，里面含有配置对象、响应头、响应结果以及原生AJAX的异步XMLHTTP请求。

### # 发送GET请求

```
1 btns[0].onclick = function(){
2   //发送 AJAX 请求
3   axios({
4     //请求类型
5     method: 'GET',
6     //URL
7     url: 'http://localhost:3000/posts/2',
8   }).then(response => {
9     console.log(response);
10  });
11 }
```

### # 发送POST请求

```
1 //添加一篇新的文章
2 btns[1].onclick = function(){
3   //发送 AJAX 请求
4   axios({
5     //请求类型
6     method: 'POST',
7     //URL
8     url: 'http://localhost:3000/posts',
9     //设置请求体
10    data: {
11      title: "今天天气不错, 还挺风和日丽的",
12      author: "张三"
13    }
14  })
15 }
```

```
14     }).then(response => {
15         console.log(response);
16     });
17 }
```

### # 发送PUT请求

```
1 //更新数据
2 btns[2].onclick = function(){
3     //发送 AJAX 请求
4     axios({
5         //请求类型
6         method: 'PUT',
7         //URL
8         url: 'http://localhost:3000/posts/3',
9         //设置请求体
10        data: {
11            title: "今天天气不错, 还挺风和日丽的",
12            author: "李四"
13        }
14    }).then(response => {
15        console.log(response);
16    });
17 }
```

### # 发送DELETE请求

```
1 //删除数据
2 btns[3].onclick = function(){
3     //发送 AJAX 请求
4     axios({
5         //请求类型
6         method: 'delete',
7         //URL
8         url: 'http://localhost:3000/posts/3',
9     }).then(response => {
10        console.log(response);
11    });
12 }
```

## 5. 常用语法

### # axios.request

等同于 axios(config)

```
1 //发送 GET 请求
2 btns[0].onclick = function(){
3   // axios()
4   axios.request({
5     method:'GET',
6     url: 'http://localhost:3000/comments'
7   }).then(response => {
8     // 成功之后的结果
9     console.log(response);
10  })
11 }
```

### # axios.post

发 post 请求

```
1 //发送 POST 请求
2 btns[1].onclick = function(){
3   // axios()
4   axios.post(
5     'http://localhost:3000/comments',
6     {
7       "body": "喜大普奔",
8       "postId": 2
9     }).then(response => {
10    console.log(response);
11  })
12 }
```

## 6. response分析

```
▼ (data: Array(1), status: 200, statusText: "OK", headers: {}, config: {}, ...)
  ▶ config: {url: "http://localhost:3000/comments", method: "get", headers: {}, transformRequest: Array(1), transformResponse: Array(1), ...} 配置对象
  ▶ data: [{}] 响应结果
  ▶ headers: {cache-control: "no-cache", content-length: 68, content-type: "application/json; charset=utf-8", expires: "-1", pragma: "no-cache"} 响应头
  ▶ request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, onreadystatechange: f, ...} AJAX请求对象
    status: 200
    statusText: "OK"
  ▶ __proto__: Object
```

## 7. 配置axios默认配置

```
1 //默认配置
2 axios.defaults.method = 'GET';//设置默认的请求类型为 GET
3 axios.defaults.baseURL = 'http://localhost:3000';//设置基础 URL
4 axios.defaults.params = {id:100}; // 设置&后面内容
5 axios.defaults.timeout = 3000; // 延迟时间
6
7 btns[0].onclick = function(){
8   axios({
9     url: '/posts'
10  }).then(response => {
11    console.log(response);
12  })
13 }
```

**优势**：通过默认配置，可以方便在后面使用同一请求时，不需要写多余的代码。

## 8. axios创建实例对象

```
1 // 这里创建的实例对象和axios几乎有一样的功能
2 const duanzi = axios.create({
3   baseURL: 'https://api.apiopen.top',
4   timeout: 2000
5 });
6 const another = axios.create({
7   baseURL: 'https://b.com',
8   timeout: 2000
9 });
10
11 duanzi.get('/getJoke').then(response => {
12   console.log(response.data)
13 })
14 another({
15   url: '/getJoke'
16 }).then(response=>{
17   console.log(response)
18 })
```

**好处**：如果要对多个不同地址发送请求，可以通过构建对象的方式，对多个不同的请求对象多次发送请求，节省代码，也就是默认配置的plus版。

## 9. 拦截器

**请求拦截器**：在发送请求前，进行一些预处理，满足则放行。

**响应拦截器**：在接受响应之前，进行一些预处理，满足则放行。

请求拦截器语法：

```
axios.interceptors.request.use(config=>{ },error=>{ })
```

响应拦截器语法：

```
axios.interceptors.response.use(response=>{ },error=>{ })
```

跟Promise语法极其相似

```
1 // 设置请求拦截器 config 配置对象
2 axios.interceptors.request.use(function (config) {
3   console.log('请求拦截器 成功 - 1号');
4   //修改 config 中的参数
5   config.params = {a:100};
6   return config;
7 }, function (error) {
8   console.log('请求拦截器 失败 - 1号');
9   return Promise.reject(error);
10 });
11 // 设置第二个请求拦截器
12 axios.interceptors.request.use(function (config) {
13   console.log('请求拦截器 成功 - 2号');
14   //修改 config 中的参数
15   config.timeout = 2000;
16   return config;
17 }, function (error) {
18   console.log('请求拦截器 失败 - 2号');
19   return Promise.reject(error);
20 });
21
22 // 设置响应拦截器
23 axios.interceptors.response.use(function (response) {
24   console.log('响应拦截器 成功 1号');
```

```

25     return response.data;
26     // return response;
27 }, function (error) {
28     console.log('响应拦截器 失败 1号')
29     return Promise.reject(error);
30 });
31 // 设置第二个响应拦截器
32 axios.interceptors.response.use(function (response) {
33     console.log('响应拦截器 成功 2号')
34     return response;
35 }, function (error) {
36     console.log('响应拦截器 失败 2号')
37     return Promise.reject(error);
38 });
39
40 //发送请求
41 axios({
42     method: 'GET',
43     url: 'http://localhost:3000/posts'
44 }).then(response => {
45     console.log('自定义回调处理成功的结果');
46     console.log(response);
47 });

```

上面的执行顺序为 **请求2-请求1-响应1-响应2**，请求拦截器是先进后出，响应拦截器是先进先出（原因在源码分析中会解释）

**备注**：在请求拦截器中可以设置 **config** 中的参数，如设置 **parms**；在响应拦截器中可以设置 **response** 的参数，如设置返回响应主体 **response.data**。

## 10. 请求取消

可以设置在请求到达本地之前取消本次请求。常常用在抢购商品时，避免用户多次请求造成服务器压力过大，因此可以设置在第一次请求到达之前，**自动取消** 发送的所有请求。

演示延迟：`json-server --watch db.json -d 2000`

```

1 //获取按钮
2 const btns = document.querySelectorAll('button');
3 //2.声明全局变量

```



```
4 let cancel = null;
5 //发送请求
6 btns[0].onclick = function(){
7     //检测上一次的请求是否已经完成
8     if(cancel !== null){
9         // 取消上一次的请求
10        cancel();
11        // 取消当前请求
12        return 0;
13    }
14    axios({
15        method: 'GET',
16        url: 'http://localhost:3000/posts',
17        //1. 添加配置对象的属性
18        cancelToken: new axios.CancelToken(function(c){
19            //3. 将 c 的值赋值给 cancel
20            cancel = c;
21        })
22    }).then(response => {
23        console.log(response);
24        //将 cancel 的值初始化
25        cancel = null;
26    })
27 }
28
29 //绑定第二个事件取消请求
30 btns[1].onclick = function(){
31     cancel();
32 }
```

可以通过 `cancel()` 取消上次请求。

## 二、axios源码分析

未完待续.....



































