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

```
获取：GET　/posts
GET　/posts/1
添加：POST　/posts
更新：PUT　/posts/1
PATCH　/posts/1
删除：DELETE /posts/1
```

创建db.json至文件夹根目录：

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "profile": { "name": "typicode" }
}
```

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：

```
<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

```
btns[0].onclick = function(){
    //发送 AJAX 请求
    axios({
        //请求类型
        method: 'GET',
        //URL
        url: 'http://localhost:3000/posts/2',
    }).then(response => {
        console.log(response);
    });
}
```

### # 发送POST请求

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

```
    }).then(response => {
        console.log(response);
    });
}
```

**# 发送PUT请求**

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

**# 发送DELETE请求**

```
//删除数据
btns[3].onclick = function(){
    //发送 AJAX 请求
    axios({
        //请求类型
        method: 'delete',
        //URL
        url: 'http://localhost:3000/posts/3',
    }).then(response => {
        console.log(response);
    });
}
```

## 5. 常用语法

### # axios.request

等同于 axios(config)

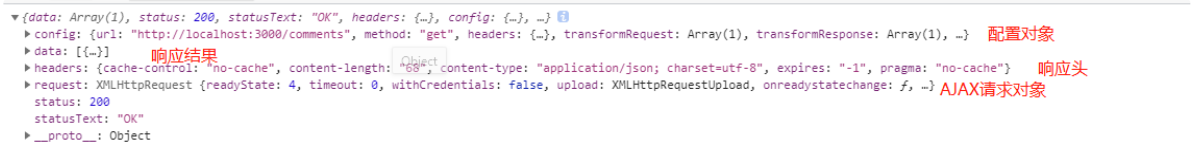
```
//发送 GET 请求
btns[0].onclick = function(){
    // axios()
    axios.request({
        method:'GET',
        url: 'http://localhost:3000/comments'
    }).then(response => {
        // 成功之后的结果
        console.log(response);
    })
}
```

### # axios.post

发 post 请求

```
//发送 POST 请求
btns[1].onclick = function(){
    // axios()
    axios.post(
        'http://localhost:3000/comments',
        {
            "body": "喜大普奔",
            "postId": 2
        }).then(response => {
        console.log(response);
    })
}
```

## 6. response分析



## 7. 配置axios默认配置

```
//默认配置
axios.defaults.method = 'GET';//设置默认的请求类型为 GET
axios.defaults.baseURL = 'http://localhost:3000';//设置基础 URL
axios.defaults.params = {id:100}; // 设置&后面内容
axios.defaults.timeout = 3000; // 延迟时间

btns[0].onclick = function(){
    axios({
        url: '/posts'
    }).then(response => {
        console.log(response);
    })
}
```

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

```
// 这里创建的实例对象和axios几乎有一样的功能
const duanzi = axios.create({
    baseURL: 'https://api.apiopen.top',
    timeout: 2000
});
const another = axios.create({
    baseURL: 'https://b.com',
    timeout: 2000
});

duanzi.get('/getJoke').then(response => {
    console.log(response.data)
})
another({
    url:'/getJoke'
}).then(response=>{
    console.log(response)
})
```

> 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

```
// 设置请求拦截器  config 配置对象
axios.interceptors.request.use(function (config) {
    console.log('请求拦截器 成功 - 1号');
    //修改 config 中的参数
    config.params = {a:100};
    return config;
}, function (error) {
    console.log('请求拦截器 失败 - 1号');
    return Promise.reject(error);
});
// 设置第二个请求拦截器
axios.interceptors.request.use(function (config) {
    console.log('请求拦截器 成功 - 2号');
    //修改 config 中的参数
    config.timeout = 2000;
    return config;
}, function (error) {
    console.log('请求拦截器 失败 - 2号');
    return Promise.reject(error);
});

// 设置响应拦截器
axios.interceptors.response.use(function (response) {
    console.log('响应拦截器 成功 1号');
```

```
    return response.data;
    // return response;
}, function (error) {
    console.log('响应拦截器 失败 1号')
    return Promise.reject(error);
});
// 设置第二个响应拦截器
axios.interceptors.response.use(function (response) {
    console.log('响应拦截器 成功 2号')
    return response;
}, function (error) {
    console.log('响应拦截器 失败 2号')
    return Promise.reject(error);
});

//发送请求
axios({
    method: 'GET',
    url: 'http://localhost:3000/posts'
}).then(response => {
    console.log('自定义回调处理成功的结果');
    console.log(response);
});
```

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

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

```
let cancel = null;
//发送请求
btns[0].onclick = function(){
    //检测上一次的请求是否已经完成
    if(cancel !== null){
        // 取消上一次的请求
        cancel();
        // 取消当前请求
        return 0;
    }
    axios({
        method: 'GET',
        url: 'http://localhost:3000/posts',
        //1. 添加配置对象的属性
        cancelToken: new axios.CancelToken(function(c){
            //3. 将 c 的值赋值给 cancel
            cancel = c;
        })
    }).then(response => {
        console.log(response);
        //将 cancel 的值初始化
        cancel = null;
    })
}

//绑定第二个事件取消请求
btns[1].onclick = function(){
    cancel();
}
```

> 可以通过cancel()取消上次请求。

## 二、axios源码分析

未完待续.....