axios总结

基础描述

(1).axios基础 promise 的 http 请求库，可用于浏览器和node.js中
(2).返回的是Promise对象

请求逻辑:

当请求axios.post等请求时会放回axios 类的post等方法，然后请求request 方法(在此方法中通过添加链chain 进行了请求拦截) 在request 方法中请求getAddter 方法 getAddter 方法中去判断当前请求是否为浏览器端，如果是浏览器端就访问封装的xhr方法，返回一个xhr，然后返回数据。

特征

* 从浏览器中创建 XMLHttpRequest
* 从node.js发出http请求
* 支持Promise API
* 拦截请求和响应
* 转换请求和响应数据
* 取消请求
    自动转换JSON数据
    客户端支持防止CSRF/SXRF

示例

```
axios.post(url,data,[config])
axios.get(url,data,[config])
```

原理

通过封装一个axio类，axios类中包含post、get、request等方法；
request：为总入口函数，在post、get等请求方法中需要用到，接收一个config配置项(接收的配置会与原有默认配置进行合并，传入的url会覆盖原有url配置，headers会与原有配置项进行合并)，当调用request时会首先调用getAddter方法 返回一个addter对象；

(1).axios类
    封装一个axios类 其中包含post、get等方法，在执行post或者get时，调用axios中的request方法，将config配置项参数传入到request中，然后再request中调用 getAddter 请求拦截需要在request 方法中实现

```
    class Axios {
        constructor(){
            // 拦截器(interceptors)
            this.interceptors = {
                request: new InterceptorManager(),
                response: new InterceptorManager()
            }
        }
        post(url, data, config = {}){
```

```
            config.method = 'post';
            config.url = url;
            config.data = data;
            return this.request(config);
        }
        get () {} // 与post相似
        // 通用请求方法
        request(config){
            // 声明一个chain 防止addter 立即执行

            let addter = getAddter(config);
            return addter;
        }
    }
```

(2).拦截器(interceptors类)

在axios 的request 请求中定义一个链 chain 然后将 addter 添加到链中 并且循环 axios interceptors.request.handers 获取到存入其中的包含成功以及失败的拦截函数 使用unshift 将此对象推到链chain 前面 接着循环执行链 chain中的方法 起到拦截作用

```
    // 请求拦截器
    class Interceptors(){
        constructor() {
            // 定一个一个数组存放事件
            this.handers = [];
        }
        // addevenListener 添加绑定事件 一个成功事件 一个失败事件
        use(fulfilled, rejected) {
            // 将事件push 进入 到定义好的数组中
            this.handers.push({
                fulfilled: fulfilled,
                rejected: rejected
            });
        }
    }
```

(3).getAddter

调用axios的resquest方法时调用getAddter 在getAddter方法中判断访问类型 根据不同的访问类型 调用不同方法返回结果 (例：如果是浏览器端就调用xhr 方法 返回一个promise)

```
    function xhr (config){
        new Promise((resolve, reject) => {
            let xhr = new XMLHttpRequest();
            xhr.onload = function () {
                // 此处返回数据
                resolve(xhr.response);
            }
            let fd = new FormData();
            for (let key in config.data) {
                fd.append(key, config.data[key]);
            }
            xhr.open(config.method, config.url, true);
            xhr.send(fd);
            return xhr;
        })
    }
    // 返回addter
    function getAddter(config) {
        // 接受一个config 配置
        // 判断请求类型（浏览器或者node.js
        // 如果是浏览器 就调用一个xhr 方法 返回一个promise对象
        xhr(config);
```

```
    }

    axios
```