1. express ---> js
   1. 引入express

const express = require('express');

* 1. 创建应用对象

const app = express();

* 1. 创建路由规则

// request 是对请求报文的封装

// response 是对响应报文的封装

app.get('/', (request, response)=> {

    response.send('HELLO EXPRESS');

});

app.post('/server', (request, response)=> {

    // 设置响应头  设置允许跨越

    response.setHeader('Access-Control-Allow-Origin','\*');

    // 设置响应体

    response.send('HELLO AJAX POST');

});

app.all('/json-server', (request, response)=> {

    response.setHeader('Access-Control-Allow-Origin','\*');

    response.setHeader('Access-Control-Allow-Headers','\*');

    // 响应一个数据

    const data = {

        name: 'atguigu'

    };

    // 对对象进行字符串转换

    let str = JSON.stringify(data);

    // 设置响应体

    response.send(str);

});

* 1. 监听端口启动服务

app.listen(8000, ()=> {

    console.log('服务已经启用,8000端口监听中....');

})

1. html

btn.onclick = function() {

            // console.log('test');

            // 1. 创建对象

            const xhr = new XMLHttpRequest();

            // 2. 初始化 设置请求方法和 url

            xhr.open('GET', 'http://127.0.0.1:8000/server');

            xhr.open('POST', 'http://127.0.0.1:8000/server');

            xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded')

            xhr.setRequestHeader('name', 'andy');

            // 设置响应体数据的类型 不设置自己手动数据转换

            xhr.responseType = 'json';

            // 超时设置

            xhr.timeout = 2000;

            // 超时回调

            xhr.ontimeout = function() {

                alert('网络异常, 请稍后重试!!');

            }

            // 网络异常回调

            xhr.onerror = function() {

                alert('你的网络似乎出了一些问题!');

            }

        // abort 取消请求

xhr.abort();

            // 3. 发送

            xhr.send();

            // xhr.send('a=100&b=200&c=300');

            // xhr.send('a:100&b:200&c:300');

            xhr.send('1233211234567');

        // 标识变量  重复发送请求解决

        let isSending = false; // 是否正在发送AJAX请求

        btns[0].onclick = function() {

            // 判断标识变量

            if (isSending) x.abort(); // 如果正在发送，则取消该请求，创建一个新的请求

            x = new XMLHttpRequest();

            // 修改  标识变量的值

            isSending = true;

            x.open('GET', 'http://127.0.0.1:8000/delay');

            x.send();

            x.onreadystatechange = function() {

               if (x.readyState === 4) {

                   isSending = false;

               }

            }

            // 4. 事件绑定 处理服务器端返回的结果

            // readystate 是 xhr 对象中的一个属性，表示状态

            //      0: 未初始化

            //      1: open方法已经调用完毕

            //      2: send方法已经调用完毕

            //      3: 服务端返回部分结果

            //      4: 服务器返回了所有的结果

            // change  改变

            xhr.onreadystatechange = function() {

                if (xhr.readyState === 4) {

                    if (xhr.status >= 200 && xhr.status < 300) {

                        // console.log(xhr.statusText);  // 状态字符串

                        // console.log(xhr.getAllResponseHeaders());  // 所有响应头

                        // console.log(xhr.response);  // 响应体

                        result.innerHTML = xhr.response;

                }

            }

        }

1. Axios

// 配置 baseURL

        axios.defaults.baseURL = 'http://127.0.0.1:8000';

        btns[0].onclick = function() {

            axios.get('/axios-server', {

                // url 参数

                params: {

                    id: 100,

                    vip: 7

                },

                // 请求头信息

                headers: {

                    name: 'atguigu',

                    age: 20

                }

            }).then(value => {

                console.log(value);

            })

        }

        btns[1].onclick = function() {

            axios.post('/axios-server', {

                // 请求体

                username: 'admin',

                password: 'admin'

            },{

                // url

                params: {

                    id: 200,

                    vip: 9

                },

                // 请求头参数

                headers: {

                    height: 100,

                    weight: 100

                }

            })

        }

        btns[2].onclick = function() {

            axios({

                // 请求方法

                method: 'post',

                // url

                url: '/axios-server',

                // url 参数

                params: {

                    vip: 10,

                    level: 30

                },

                // 头信息

                headers: {

                    a: 100,

                    b: 200

                },

                // 请求体参数

                data: {

                    username: 'admin',

                    password: 'admin'

                }

            }).then(response => {

                // console.log(response);

                // 响应状态码

                console.log(response.status);

                // 响应状态字符串

                console.log(response.statusText);

                // 响应头信息

                console.log(response.headers);

                // 响应体

                console.log(response.data);

            })

        }

1. jsonp

   <script>

        function handle(data) {

            // 获取 result 元素

            const result = document.getElementById('result');

            result.innerHTML = data.name;

        }

</script>

<script src="http://127.0.0.1:8000/jsonp-server"></script>

app.all('/jsonp-server', (request, response) => {

    // response.send("console.log('hello jsonp-server')")  要发送一段js代码

    const data = {

        name: '尚硅谷'

    }

    // 将数据转化为字符串

    let str = JSON.stringify(data);

    // 返回结果

    response.end(`handle(${str})`);

})