AJAX = 异步 JavaScript 和 XML。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

创建 XMLHttpRequest 对象的语法：

*variable*=new XMLHttpRequest();

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

*variable*=new ActiveXObject("Microsoft.XMLHTTP");

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject ：

## 实例

var xmlhttp; if (window.XMLHttpRequest) { // IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码 xmlhttp=new XMLHttpRequest(); } else { // IE6, IE5 浏览器执行代码 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); }

如需将请求发送到服务器，我们使用 **XMLHttpRequest** 对象的 **open()** 和 **send()** 方法：

xmlhttp.open("GET","ajax\_info.txt",true);

xmlhttp.send();

|  |  |
| --- | --- |
| **方法** | **描述** |
| open(*method*,*url*,*async*) | 规定请求的类型、URL 以及是否异步处理请求。   * *method*：请求的类型；GET 或 POST * *url*：文件在服务器上的位置 * *async*：true（异步）或 false（同步） |
| send(*string*) | 将请求发送到服务器。   * *string*：仅用于 POST 请求 |

## 服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据。 |
| responseXML | 获得 XML 形式的响应数据。 |

## responseText 属性

如果来自服务器的响应并非 XML，请使用 responseText 属性。

responseText 属性返回字符串形式的响应，因此您可以这样使用：

## 实例

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_first" \t "https://www.runoob.com/ajax/_blank)**

## responseXML 属性

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 responseXML 属性：

## 实例

请求 [cd\_catalog.xml](https://www.runoob.com/try/demo_source/cd_catalog.xml" \t "https://www.runoob.com/ajax/_blank) 文件，并解析响应：

xmlDoc=xmlhttp.responseXML; txt=""; x=xmlDoc.getElementsByTagName("ARTIST"); for (i=0;i<x.length;i++) { txt=txt + x[i].childNodes[0].nodeValue + "<br>"; } document.getElementById("myDiv").innerHTML=txt;

## onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

|  |  |
| --- | --- |
| **属性** | **描述** |
| onreadystatechange | 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。 |
| readyState | 存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。   * 0: 请求未初始化 * 1: 服务器连接已建立 * 2: 请求已接收 * 3: 请求处理中 * 4: 请求已完成，且响应已就绪 |
| status | 200: "OK" 404: 未找到页面 |

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

## 实例

xmlhttp.onreadystatechange=function() { if (xmlhttp.readyState==4 && xmlhttp.status==200) { document.getElementById("myDiv").innerHTML=xmlhttp.responseText; } }

1.

axios是一个基于Promise的HTTP库,而ajax是对原生XHR的封装;

2.

ajax技术实现了局部数据的刷新,而axios实现了对ajax的封装。

# **axios中文文档**

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中。

npm axios from 'axios'

**较科学的封装好的axios：（new-axios.js）**

[IMG_256](javascript:void(0);)

import axios from 'axios'

import { Notify } from 'vant';

// import Vue from 'vue'

// import store from '@/store'  // 我此项目没有用到vuex，所以vuex代码的都注释了，需要的自己打开使用

// import {ACCESS\_TOKEN} from '@/store/mutation-types'

// 创建 axios 实例

const requests = axios.create({

  baseURL: process.env.VUE\_APP\_API, // 基础url,如果是多环境配置这样写，也可以像下面一行的写死。

　　// baseURL: 'http://168.192.0.123',

  timeout: 6000 // 请求超时时间

})

// 错误处理函数

const err = (error) => {

  if (error.response) {

    const data = error.response.data

    // const token = Vue.ls.get(ACCESS\_TOKEN)

    if (error.response.status === 403) {

        Notify({ type: 'danger', message: data.message||data.msg });

    }

    if (error.response.status === 401) {

        Notify({ type: 'danger', message: '你没有权限。' });

      // if (token) {

      //   store.dispatch('Logout').then(() => {

      //     setTimeout(() => {

      //       window.location.reload()

      //     }, 1500)

      //   })

      // }

    }

  }

  return Promise.reject(error)

}

// request interceptor(请求拦截器)

requests.interceptors.request.use(config => {

//   const token = Vue.ls.get(ACCESS\_TOKEN)

  const token = localStorage.getItem('token')

  if (token) {

    config.headers['token'] = token // 让每个请求携带自定义 token 请根据实际情况自行修改

  }

  return config

}, err)

// response interceptor（接收拦截器）

requests.interceptors.response.use((response) => {

  const res = response.data

  if (res.code !== 0&&res.code!==200) {

    Notify({ type: 'danger', message: res.message||res.msg });

    // 401:未登录;

    if (res.code === 401||res.code === 403||res.code===999) {

      Notify({ type: 'danger', message: '请登录'});

    }

    return Promise.reject('error')

  } else {

    return res

  }

}, err)

export default {

  requests

}

[IMG_257](javascript:void(0);)

**main.js 引入，添加到vue原型**

import requests from '@s/basejs/new-axios.js' // 记得改为你的路径

Vue.prototype.rq = requests // 此处命名为rq,你可以改

**使用**

[IMG_258](javascript:void(0);)

this.rq.get('/api/product/get?productChannelId='+this.productChannelId).then(res=>{

console.log(res)

})

// 传对象参数

// get请求记得加paramsthis.rq.get('/api/product/get,{params:{name:'abc'}}).then(res=>{

console.log(res)

})

this.rq.post('/api/product/get,{name:'abc'}).then(res=>{ console.log(res)

})

[IMG_259](javascript:void(0);)

**以下步骤一般不需要**

**开发环境如果要跨域，解决跨域问题（设置代理）：vue-cli 3.0的在 package.json  同级目录新建一个 vue.config.js 文件，加入下面代码，其他版本找到配置文件的devServer加入代码**

**：**

[IMG_256](javascript:void(0);)

module.exports = {

//axios域代理，解决axios跨域问题

baseUrl: '/',

devServer: {

proxy: {

'': {

target: 'http://192.168.0.108:8090',

changeOrigin: true,

ws: true,

pathRewrite: {

}

}

}

}

}

[IMG_257](javascript:void(0);)

修改完后记得重启项目应用配置