Ajax 跨域问题

1. Ajax 是通过XMLHttprequest ajax 是不能跨域

优点：不刷新页面，就能异步更新数据，给用户的体验非常好

前后端相分离，减少了服务器的压力

（主域与子域之间也存在跨域问题）

主域名：

由两个或两个以上的字母构成，中间由点号隔开，整个域名只有1个点号 比如：csdn.net

子域名：

是在主域名之下的域名，域名内容会有多个点号 比如：blog.csdn.net

1. 子域与主域之间可以通过 document.domain=”a.com” 来解决
2. 服务器代理 后端做XMLHttprequest的代理文件

（优点：可以做到任何文件 缺点：增加服务器压力 主要是后端负责）

1. script 标签的形式 也就是jsonp json+padding

利用内填充原理

Jsop的做法：get方式请求

function createJs(sUrl){

var Oscript=document.createElement(‘script’);

Oscript.type=’text/javascript’;

Oscript.src=sUrl;

Document.getElementsByTagName(‘head’)[0].appendChild(Oscript)

}

CreateJs(‘json.js?callback=box’) //指定回调函数的名字为box

function box(json){

aleart({“name”:”miaov”})

}

何为组件化开发：

自己写的代码一般就自己能看懂，如果是一个只有你负责的项目

那是无所谓的,但是不可能（因为你会离职），在项目开发中最担心的

就是项目维护（协同开发）代码不统一，修改起来比较困难。

一些重复的功能每次都重复的开发，这样会造成资源浪费

组件化开发

功能类似的模块复用、达到快速开发的目的

容易维护

低耦合，高内聚

**Proxy： 解决react跨域问题**

用一个最简单的办法解决API资源请求跨域问题： http-proxy-middleware

http-proxy-middleware不需要自己安装，在安装webpack过程中，会自动依赖安装到你的node\_modules文件夹下

webpack的devServer来启动前端项目，

在你dwebpack.config.js里面添加proxy配置：

//导入proxy

**var** proxy = require('http-proxy-middleware')

//context可以是单个字符串，也可以是多个字符串数组，分别对应你需要代理的api,星号（\*）表示匹配当前路径下面的所有api

**const** context = [`/login`, `/admin/\*`]

module.exports = {

devServer: {

host: 'localhost',

port: '3011',

proxy: [

{

context: context,

target: 'https://www.xxx.com',

changeOrigin: true,

secure: false

}

]

}

}