# 跨域

同源策略

所谓同源是指"协议+域名+端口"三者相同，即便两个不同的域名指向同一个 ip 地址，也非同源


但是有三个标签是允许跨域加载资源：

```
<img  src=XXX>
<link  href=XXX>
<script  src=XXX>
```

**当协议、子域名、主域名、端口号中任意一个不相同时，都算作不同域**。不同域之间相互请求资源，就算作“跨域”

解决跨域：

## 1.jsonp

**JSONP 原理（只能解决get跨域）**

利用 script 标签没有跨域限制的漏洞(script标签的src属性不受同源策略限制），动态创建一个script标签，网页可以得到从其他来源动态产生的 JSON 数据。JSONP 请求一定需要对方的服务器做支持才可以。

优点是简单兼容性好，可用于解决主流浏览器的跨域数据访问的问题。

缺点是仅支持 get 方法具有局限性,不安全可能会遭受 XSS 攻击

步骤：

1.去创建一个script标签

2.script的src属性设置接口地址

3.接口参数，必须要带一个自定义函数名，要不然后台无法返回数据

4.通过定义函数名去接收后台返回数据

## 2.nginx 反向代理（最简单的跨域方式）

实现原理类似于 Node 中间件代理，需要你搭建一个中转 nginx 服务器，用于转发请求。

使用 nginx 反向代理实现跨域，是最简单的跨域方式。只需要修改 nginx 的配置即可解决跨域问题，支持所有浏览器，支持 session，不需要修改任何代码，并且不会

影响服务器性能。

步骤：通过 nginx 配置一个代理服务器（域名与 domain1 相同，端口不同）做跳板机，反向代理访问 domain2 接口，并且可以顺便修改 cookie 中 domain 信息，方便当前域 cookie 写入，实现跨域登录

3.cors
4.postMessage
5.websocket
6. Node 中间件代理(两次跨域)
7.window.name + iframe
8.location.hash + iframe
9.document.domain + iframe

备注：

- CORS 支持所有类型的 HTTP 请求，是跨域 HTTP 请求的根本解决方案
- JSONP 只支持 GET 请求，JSONP 的优势在于支持老式浏览器，以及可以向不支持 CORS 的网站请求数据。
- 不管是 Node 中间件代理还是 nginx 反向代理，主要是通过同源策略对服务器不加限制。
- 日常工作中，用得比较多的跨域方案是 cors 和 nginx 反向代理

# 跨域（webpack/vue-cli）

解决跨域问题的办法有三种，后端设置、JSONP、和代理

## webpack配置proxy

webpack-dev 自带的代理服务器

单页面开发流行的情况下，如果你的项目是用webpack开发的，那么久好办了，webpack有自带的代理机制；在webpack的config设置文件中有proxy这么一项配置

```
proxy:{
    "/api":{
        target:"http://127.0.0.1:88",
```

```
        secure: false,
        changeOrigin: true
      }
}
```

在config文件中加上这个配置的话，那么你的项目中要是“/api”的接口都会被代理到”http://127.0.0.1:88/api“这个地址，那么这里你可以写上后端接口的地址；从而在开发的时候就不用跑去麻烦后端人员，也不会有跨域的问题出现；

前端在本地起了服务，要访问后端服务(或者测试环境)联调接口。例如测试环境是：101.132.133.44:9003/api/bookRank，直接访问报跨域问题。

配置webpack的devServer

```
proxy: {
// '/api'是在请求的接口中正则匹配带有/api的接口替换到
101.132.133.44:9003的服务
"/api": {
target: {
"host": "101.132.133.44",        主机
"protocol": 'http://',              协议
"port": 9003,//端口
},
pathRewrite: {'^/api' : ''},//如果接口中没有api或者路径是其他，在这里修改
// ignorePath: true,
changeOrigin: true,//必配
secure: false
}
}
```

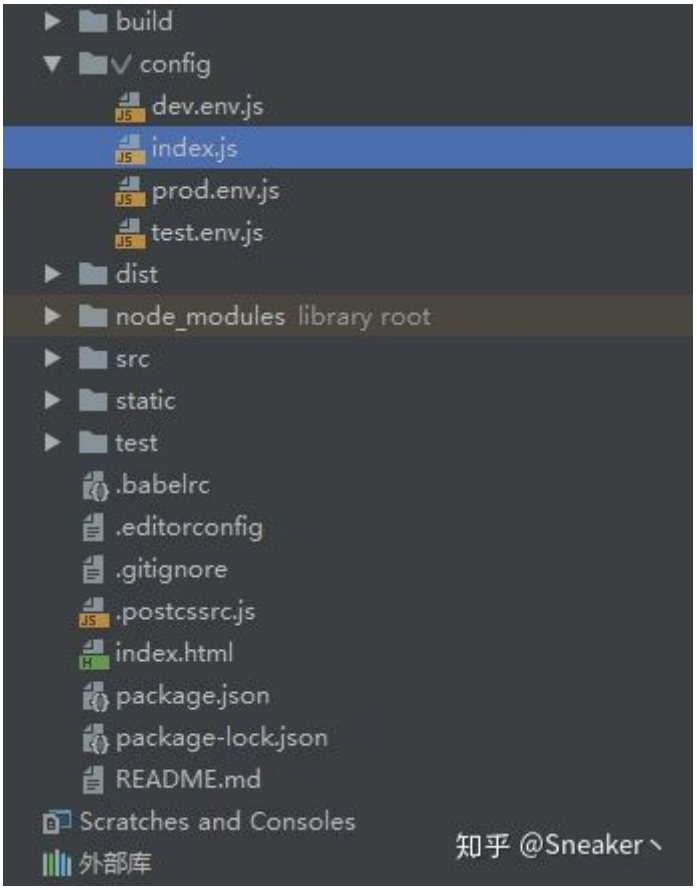
## vue-cli

如果是使用框架的脚手架进行项目搭建，也可以顺着文件的引用关系去找到这个配置在哪个位置，进行修改

vue-cli中npm start 的时候引用的webpack 配置文件是build文件下面的webpack.dev.conf； 而webpack.dev.conf 里面就有一个配置项 proxy: config.dev.proxyTable, config则是引用的 config文件下的index.js ;所以找到对应的位置进行设置就可以了

# Vue 在开发环境中如何解决axios跨域问题

在项目的config文件夹下的index.js文件里面




找到dev下proxyTable{ },在里面加入如下代码:

```
dev: {

  // Paths
  assetsSubDirectory: 'static',
  assetsPublicPath: '/',
  proxyTable: {
    '/api': {
      target: 'http://192.168.50.125:8888', //设置调用接口域名和端口号别忘了加http
      changeOrigin: true,
      pathRewrite: {
        '^/api': '' //这里理解成用'/api'代替target里面的地址，组件中我们调接口时直接用/api代替
        // 比如我要调用'http://0.0:300/user/add'，直接写'/api/user/add'即可 代理后地址栏显示/
      }}
},
```



就可以解决跨域访问的问题；

请求后台数据时，注意后台要求的数据格式。

如果要求是formData格式，则axios请求时加个请求头


在将提交的字段data改成qs.stringify(data)即可(PS：需要安装qs模块)。