# VUE环境安装

npm是node包的一个管理工具 很多前端项目都会用到npm工具

window需要手动安装git

首先，列出来我们需要的东西：

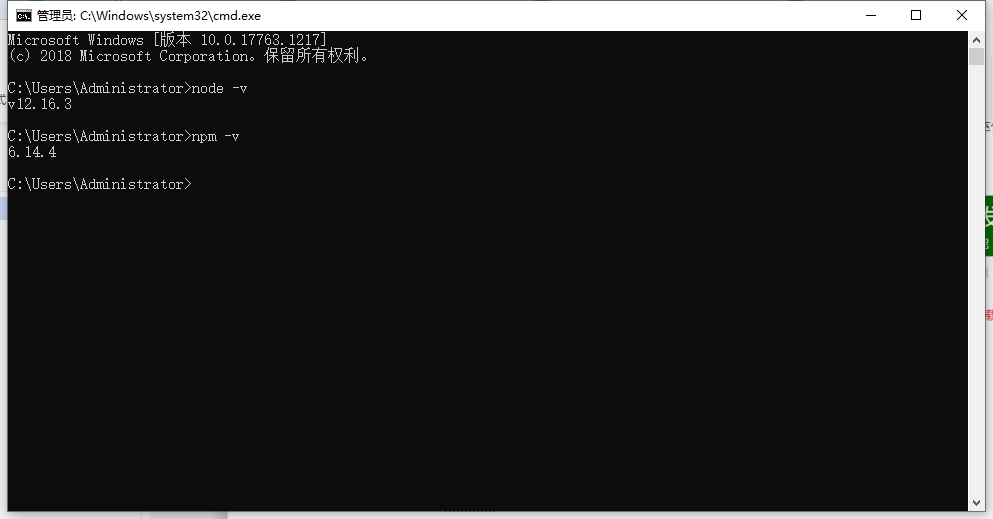
* node.js环境（npm包管理器）
* vue-cli 脚手架构建工具
* cnpm  npm的淘宝镜像

node官网中下载node 直接默认安装即可（window）



安装完用cmd模式下查询版本号。

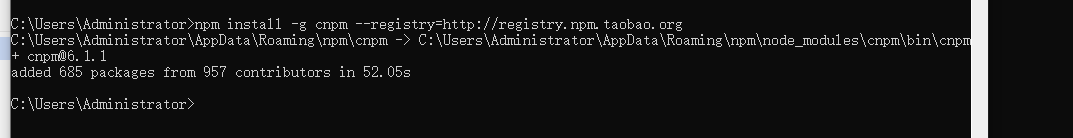
npm包管理器，是集成在node中的，所以，直接输入 npm -v就会如下图所示，显示出npm的版本信息。



OK！node环境已经安装完成，npm包管理器也有了。由于有些npm有些资源被屏蔽或者是国外资源的原因，经常会导致用npm安装依赖包的时候失败，所有我还需要npm的国内镜像---cnpm。

**安装cnpm**

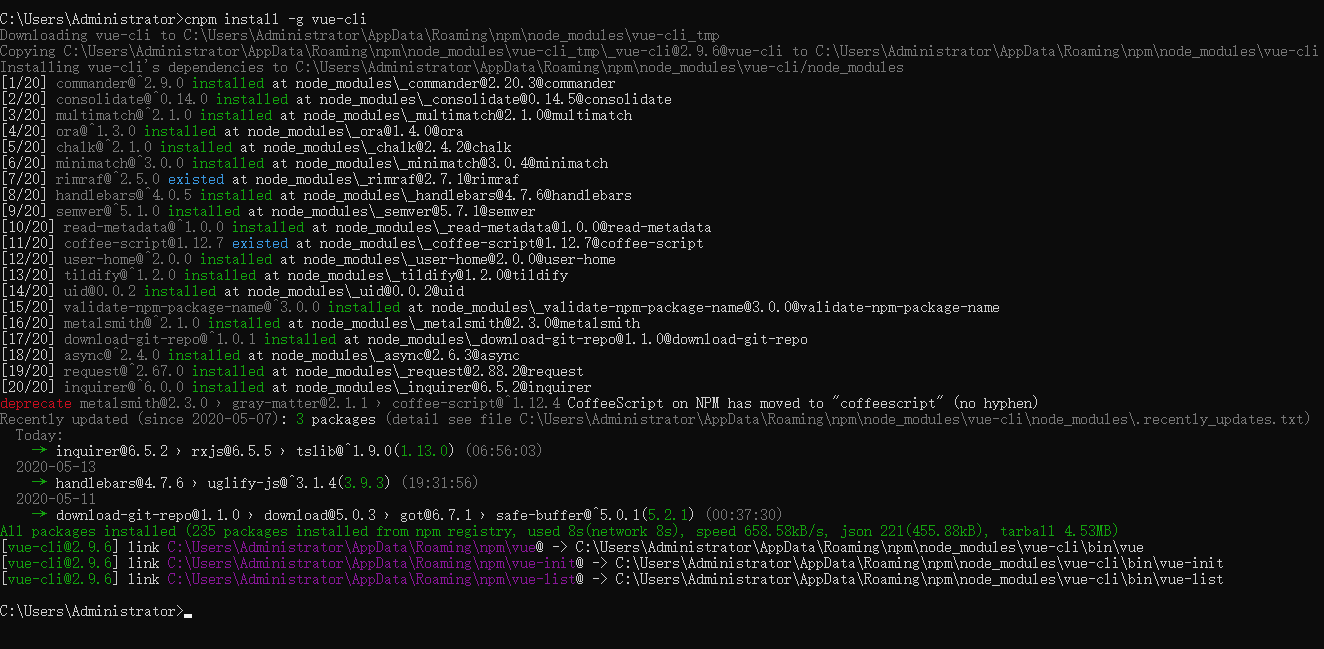
在命令行中输入 npm install -g cnpm --registry=http://registry.npm.taobao.org 然后等待，安装完成如下图。



**安装vue-cli脚手架构建工具**

在命令行中运行命令 cnpm install -g vue-cli ，然后等待安装完成。(注意，这里使用cnpm来替代npm，不然速度超级慢，会导致卡在那)

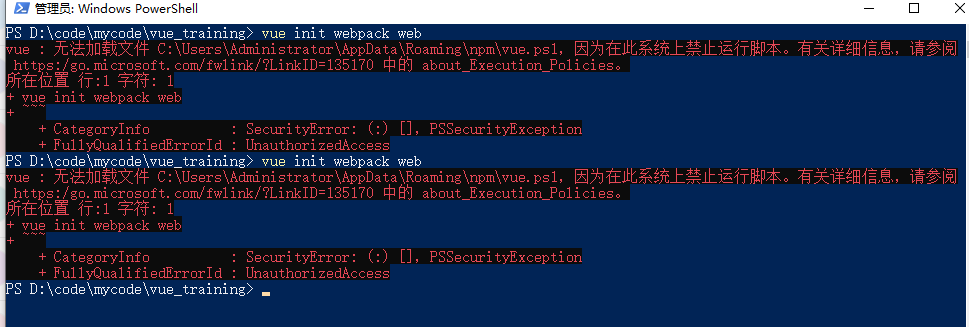
通过以上三部，我们需要准备的环境和工具都准备好了，接下来就开始使用vue-cli来构建项目。



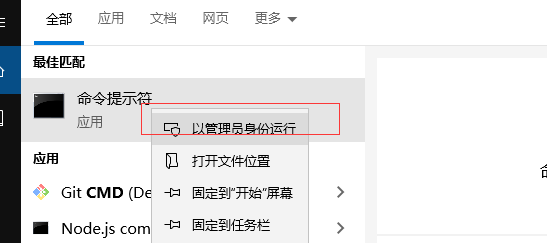
# 创建项目

## vue init webpack web

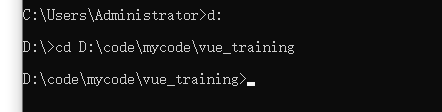
初始化web项目有可能会出现以下提示



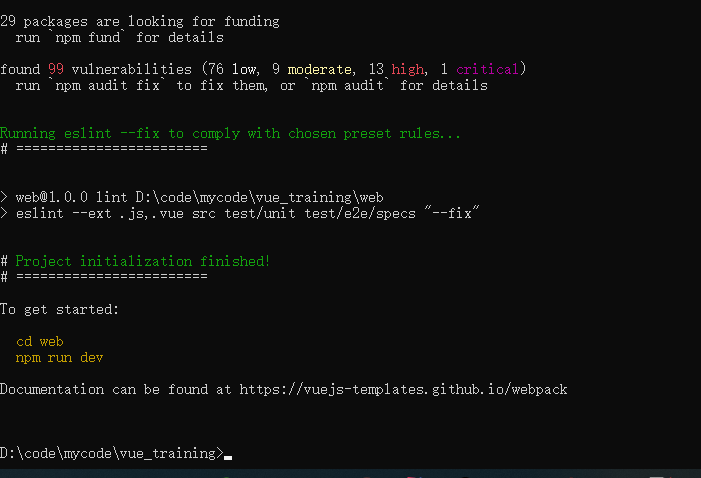
cmd需要管理员权限才可以初始化



切换到对应目录下



重新执行初始化方法



## cd web

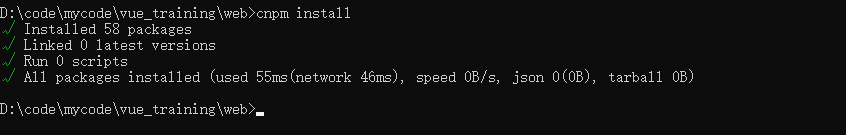
进入项目中 。



## cnpm install

刚初始化的项目是没有依赖的。运行会报错。

所以在这之前需要解决项目的依赖问题，使用下面的命令安装项目的依赖。

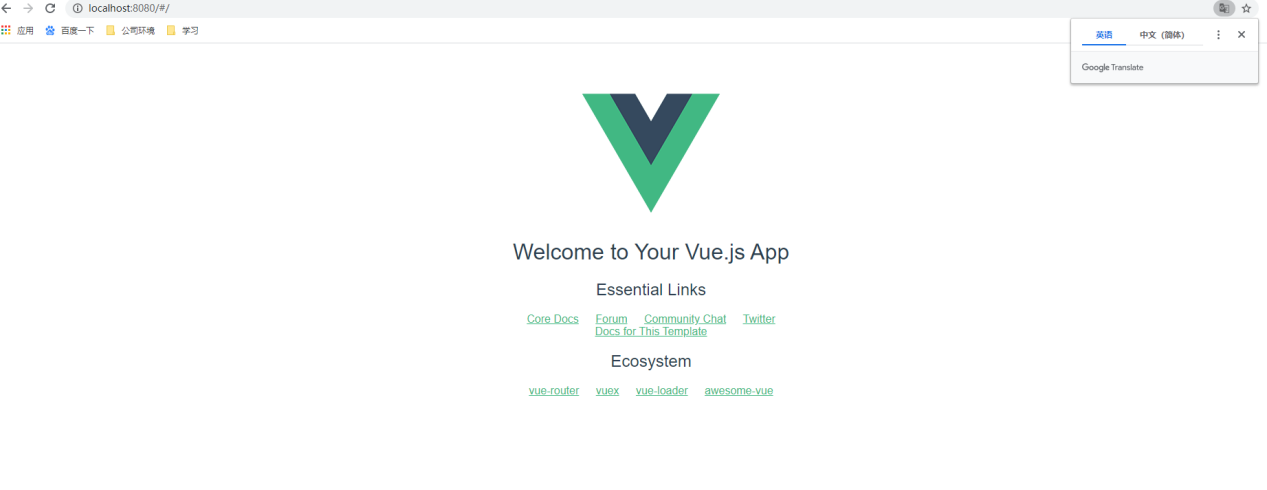


## npm run dev

启动项目



启动成功



## vue的结构目录

build —-最终发布的代码存在位置

node-module—npm加载的项目依赖模块

src —-开发目录（App.vue 项目入口文件，main.js项目的核心文件）

static—静态资源目录，如图片、字体等

test—初始测试目录，可删除

.xxx—-文件是配置文件，包括语法配置，git配置等

index.html—首页入口文件，你可以添加一些meta信息或者统计代码的

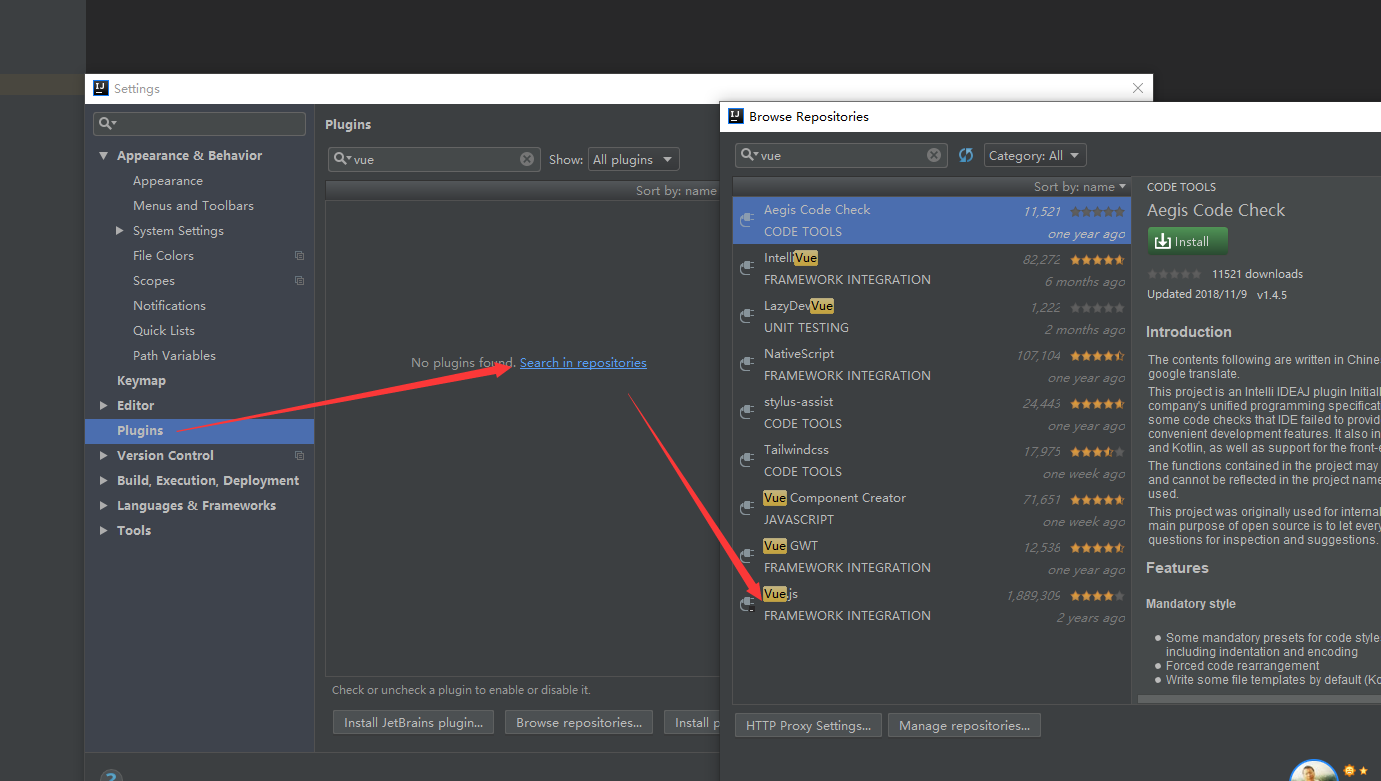
package.json—项目配置文件

READMDE.md—项目的说明文档

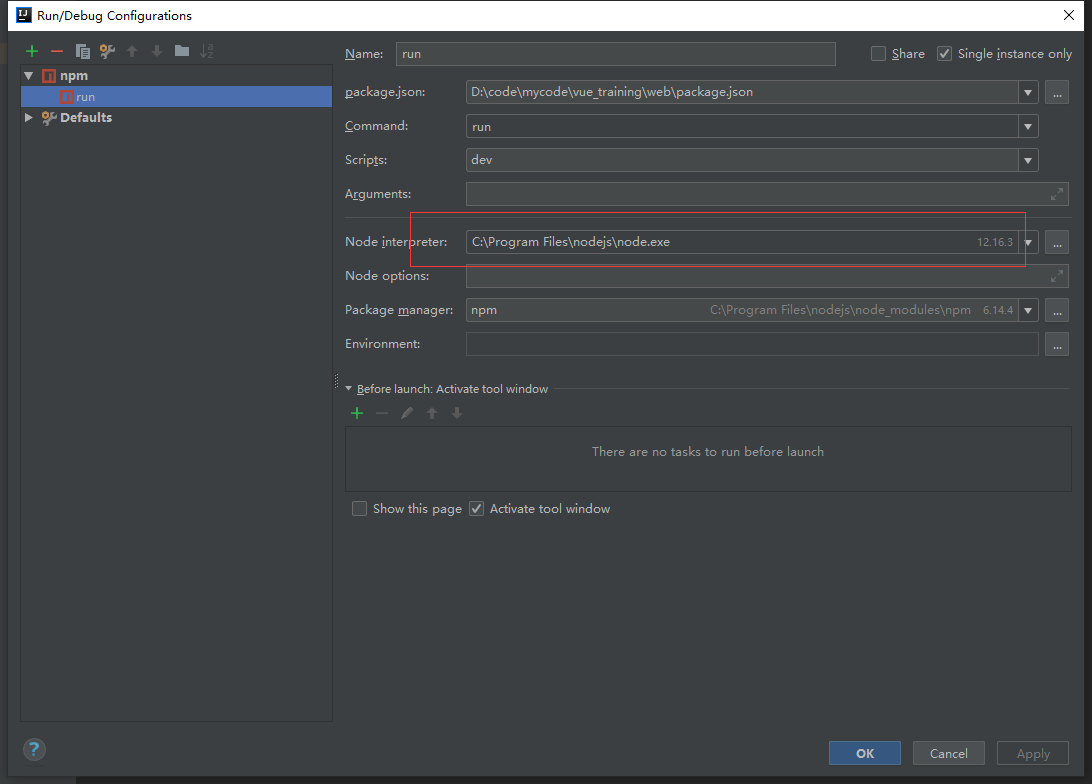
## 打包命令

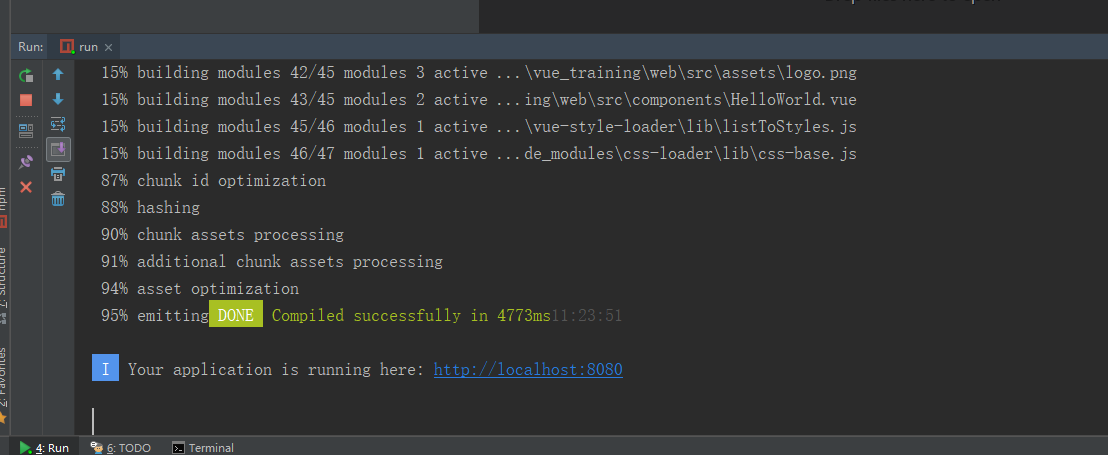
npm run build

# Idea引入vue插件



install安装然后重启即可。





# Google浏览器 vue调试插件

方法1 需要翻墙，推荐方法2

安装方法1: 需正常打开chrome商店，搜索vuejs devtools 安装。chrome://extensions/ 开发者工具-扩展程序下启用；

方法2： github下载插件，npm包安装依赖，拖入浏览器扩展程序

具体操作：

1 . **下载chrome扩展插件。**

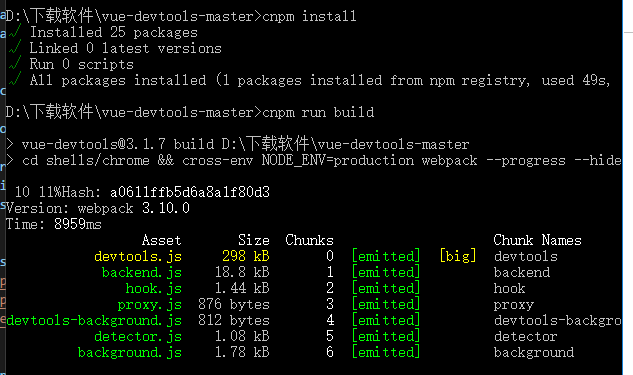
在github上下载压缩包并解压到本地，github下载地址：[https://github.com/vuejs/vue-devtools](https://github.com/vuejs/vue-devtools" \t "https://www.cnblogs.com/alice-fee/p/_blank)

2. **npm install**

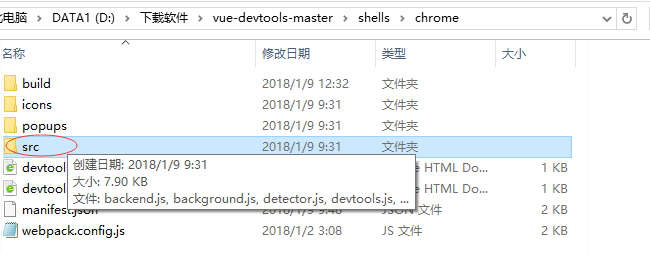
下载完成后打开命令行cmd进入vue-devtools-master文件夹，

1. npm install，安装依赖包；如果安装太慢，请参照文章末尾说明进行操作。

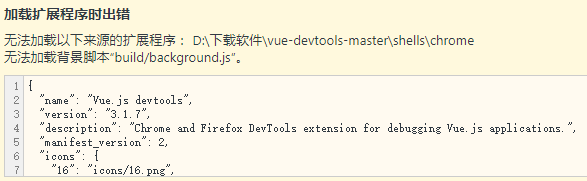
2. npm run build



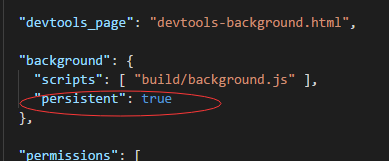
npm run build 执行完，会在shells>chrome下的src文件夹里生产如上图所示的几个js文件；



若不执行以上命令会报错，无法加载背景脚本"build/background.js"，如下图：



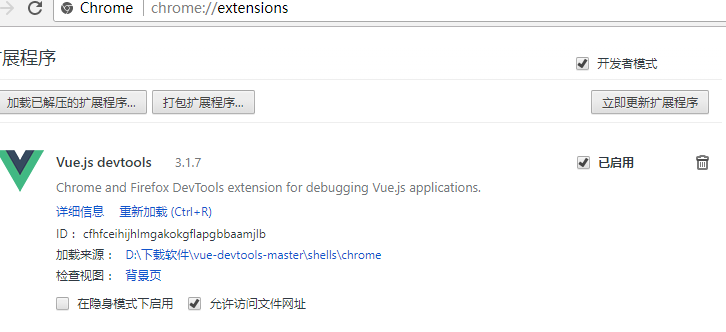
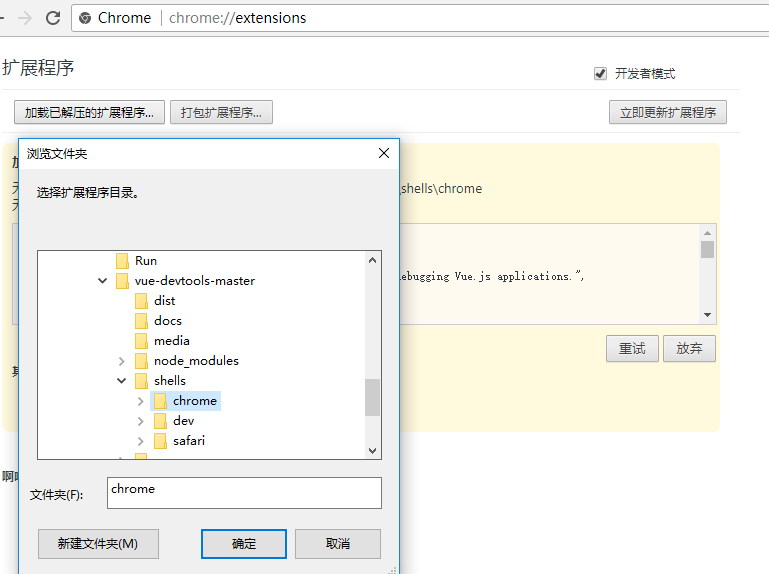
3. 打开shells>chrome>manifest.json并把json文件里的"persistent":false改成true



4. **扩展chrome插件**

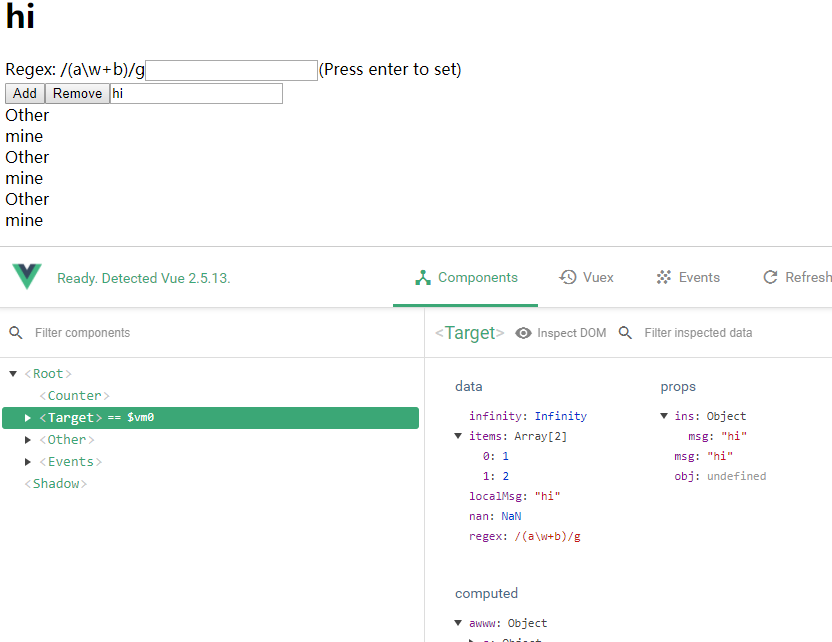
1.打开chrome浏览器，打开更多工具>扩展程序；

2.再点击加载已解压的扩展程序，然后把shells>chrome文件夹放入



**5， 测试安装成功**

在插件的目录下执行npm run dev，这个时候我们的插件就可以运行了,打开localhost:8080可以看到插件已经安装并运行了。



# 技术讲解学习

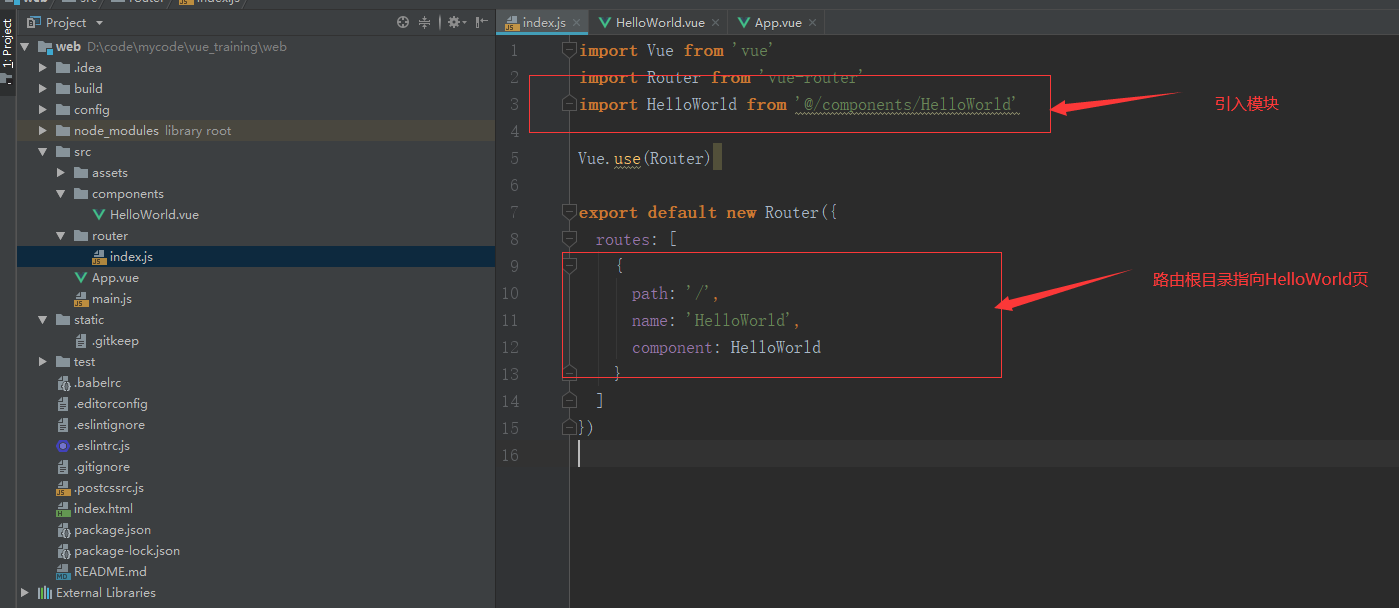
Vue和普通的html开发不一样，vue的统一入口是 index.html，然后通过路由功能进行页面跳转非html直接访问页面模式。

## App.vue

vue的入口页面

通过<router-view/>实现的页面跳转，类似htm的iframe。

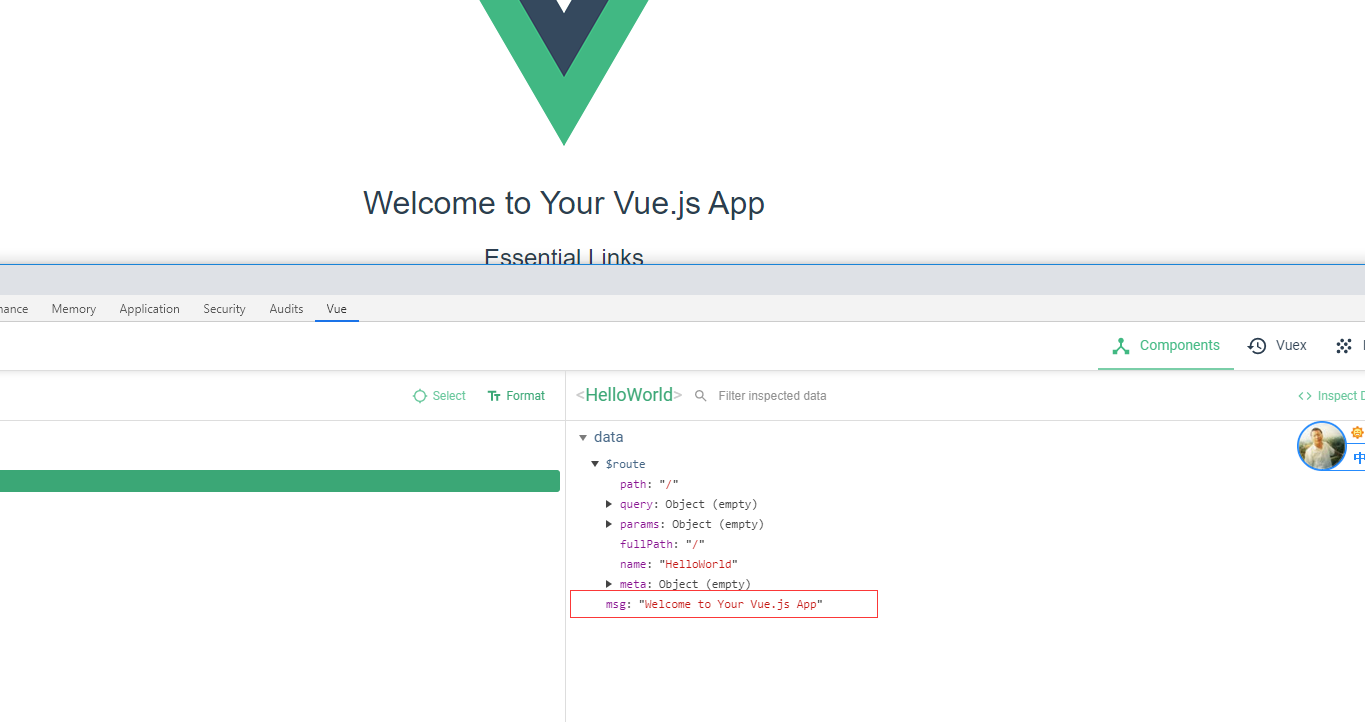
## 路由

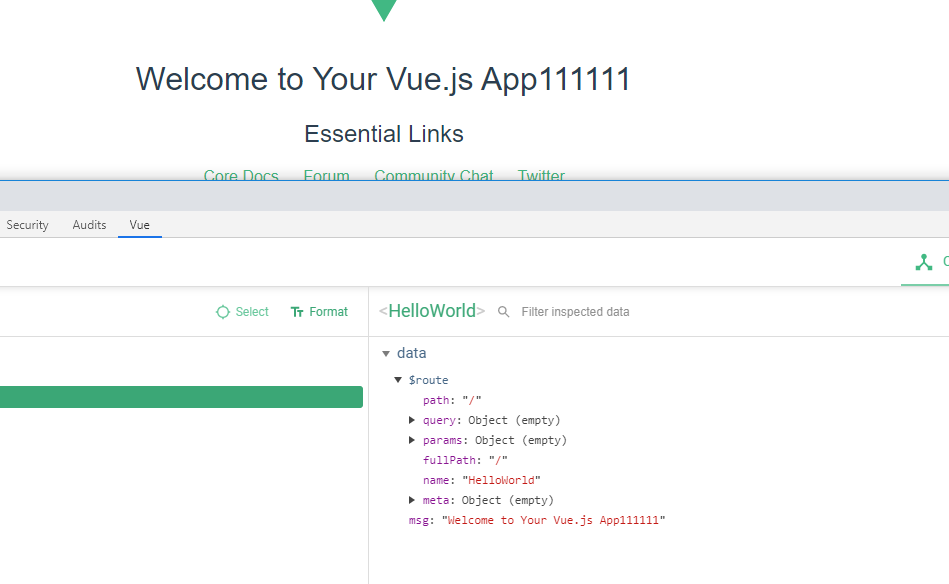


## 数据绑定

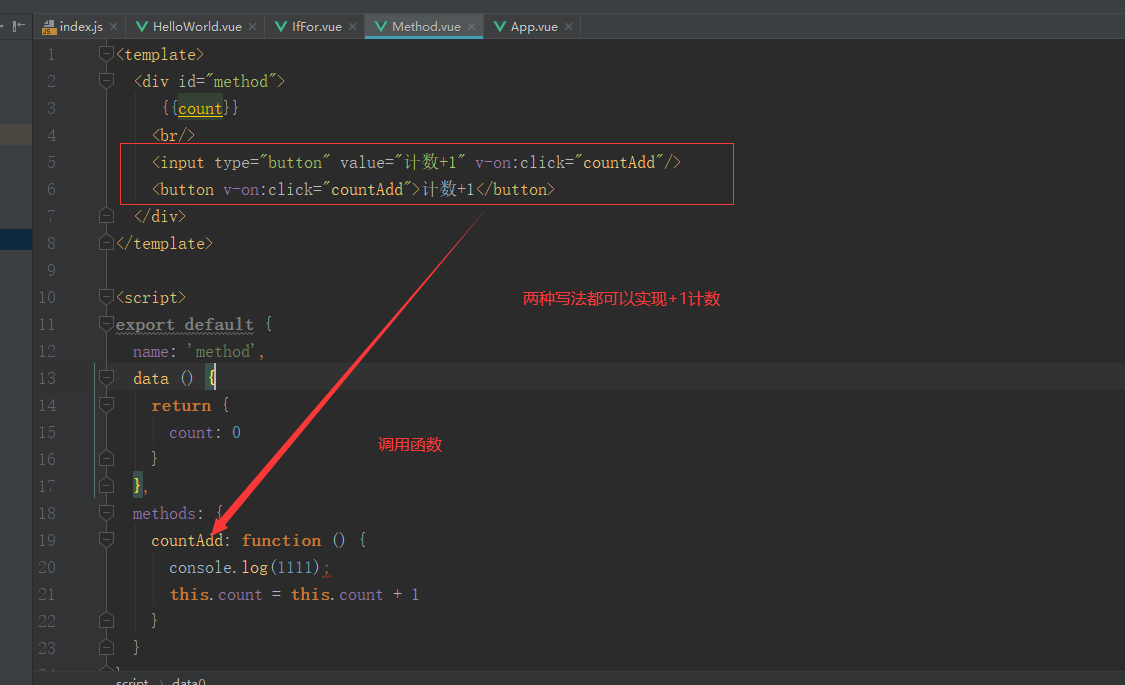
Vue是通过修改数据实现页面数据的变动，也就是只需要对数据对象进行修改，页面的内容就会发生变化，不同自己去对dom进行操作。

例如对msg数据进行修改





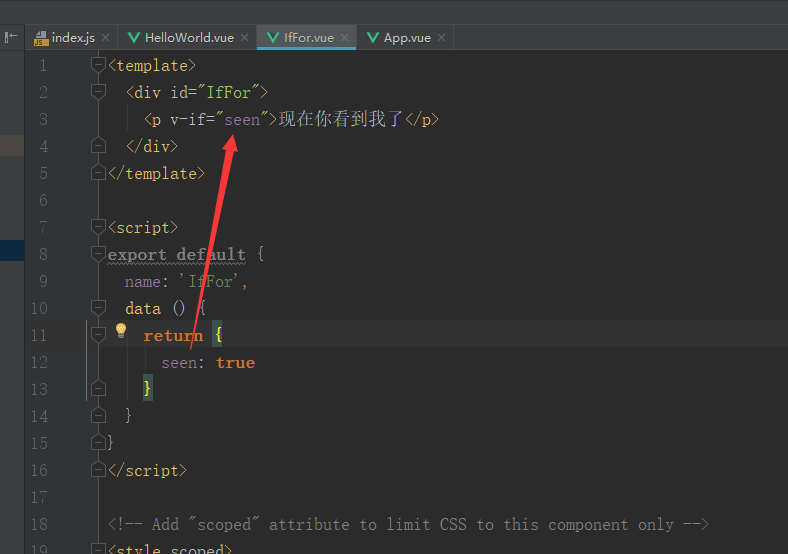
## 函数调用

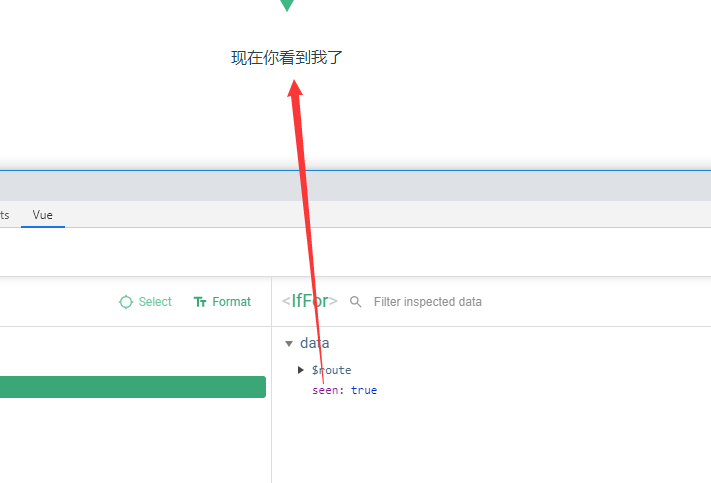


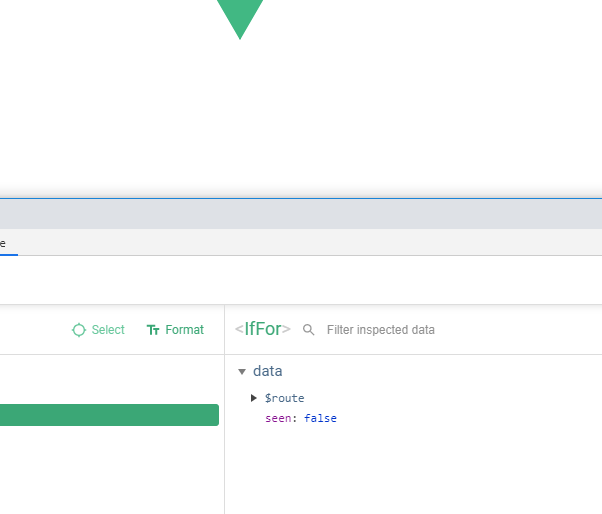
## 条件与循环

对HelloWorld做拷贝命名为 IfFor.vue文件

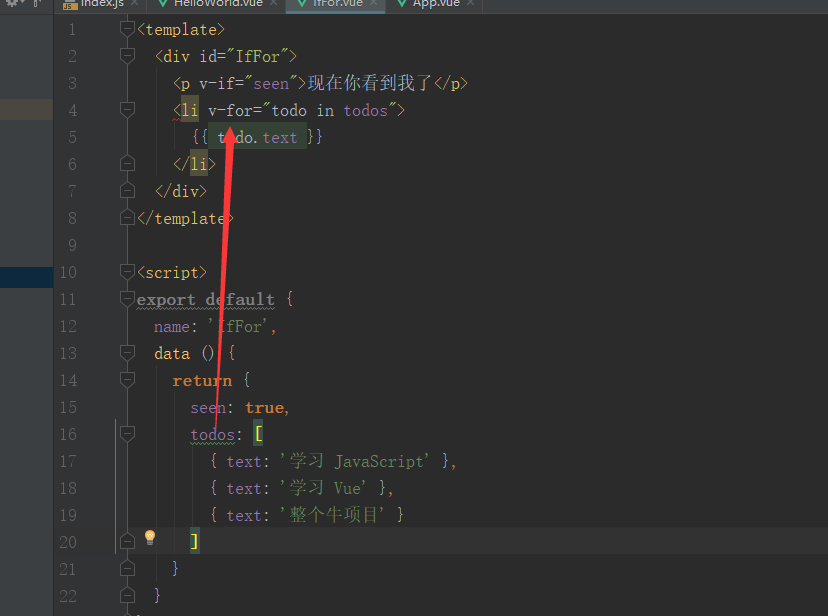
内容如下

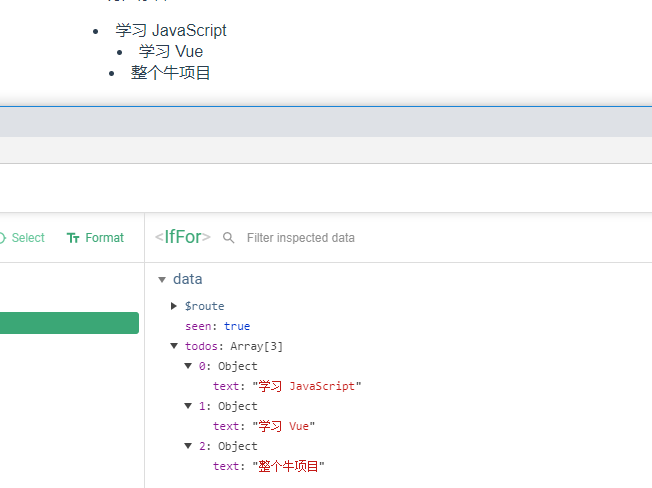






v-for会将数据循环呈现





如果是表格的话，我们只需要将从后台查询的数据复制todos，页面数据就会变化。

## 样式绑定

通过绑定数据来控制样式是否生效。

<template>

<div id="css">

<div v-bind:class="{ red: isRed,orange:isOranage }">字体颜色</div>

<button v-on:click="updateColor">更新字体颜色</button>

</div>

</template>

<script>

export default {

name: 'css',

data () {

return {

isRed: false,

isOranage: false,

count: 0

}

},

methods: {

updateColor: function () {

this.count = this.count + 1

if (this.count % 3 === 0) {

this.isRed = false

this.isOranage = false

}

if (this.count % 3 === 1) {

this.isRed = true

this.isOranage = false

}

if (this.count % 3 === 2) {

this.isRed = false

this.isOranage = true

}

}

}

}

</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->

<style scoped>

.red{

color: red;

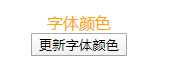
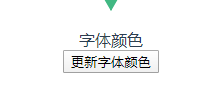
}

.orange{

color: orange;

}

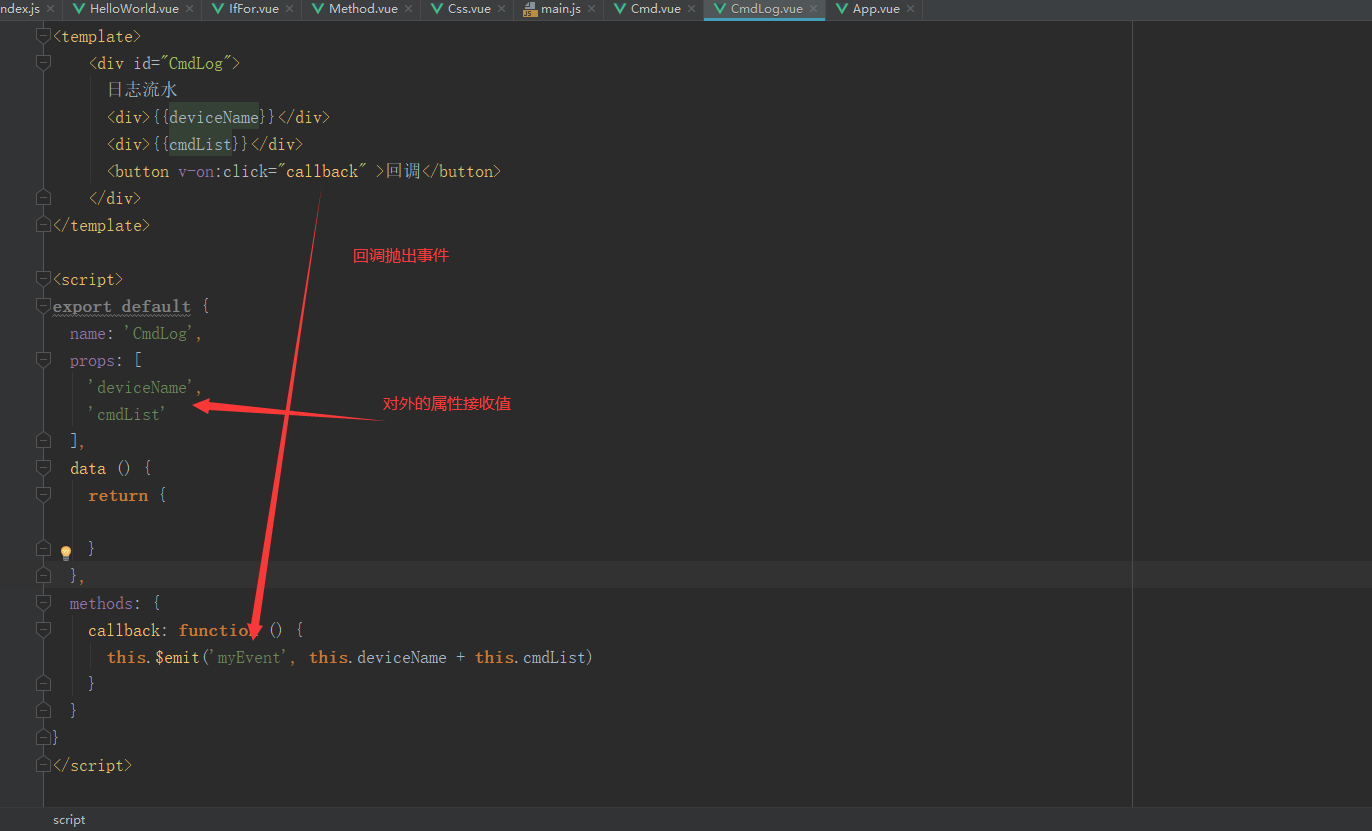
</style>



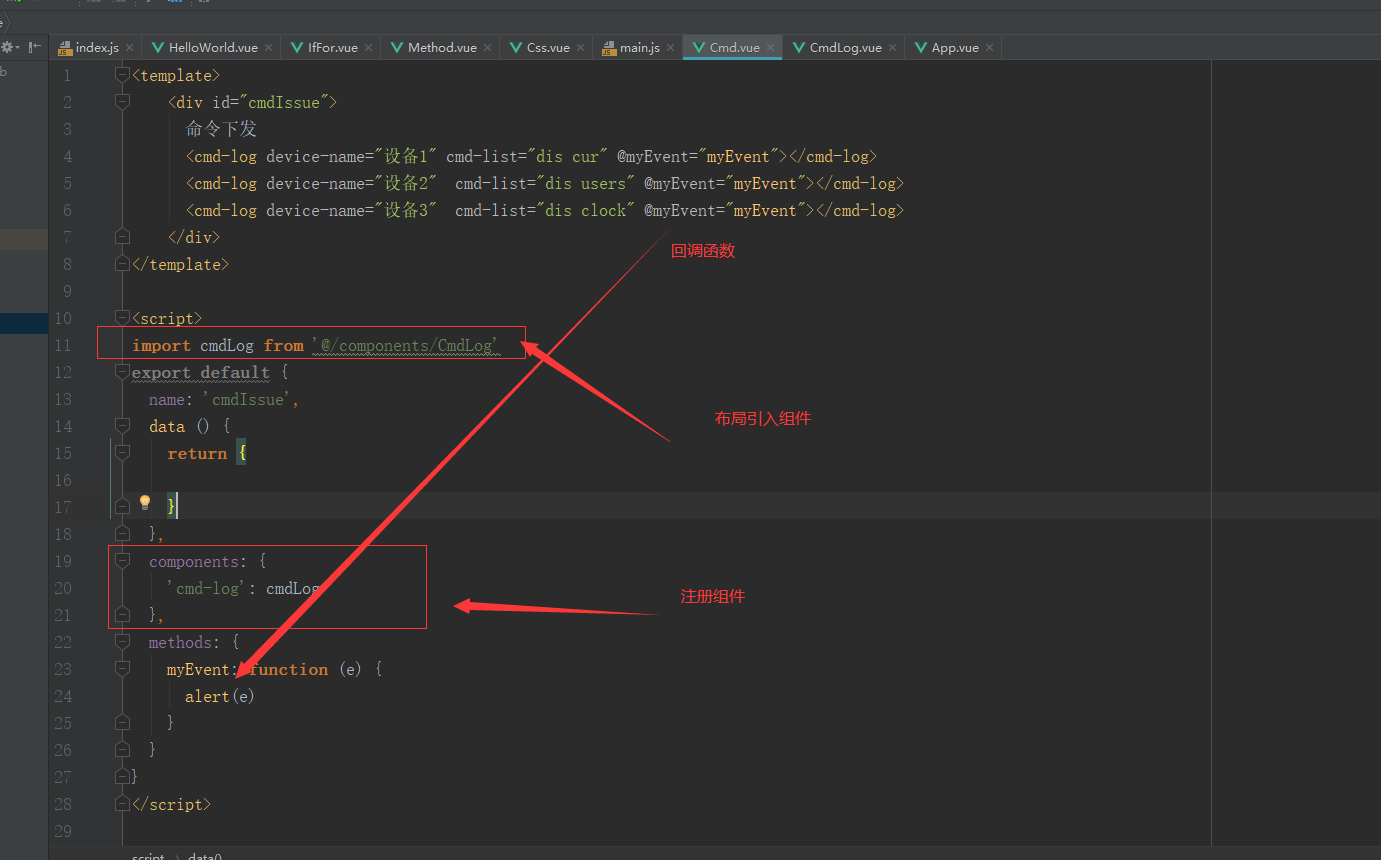
## 组件

把重复的常用的功能封装成组件供其他模块进行调用。

编写CmdLog文件



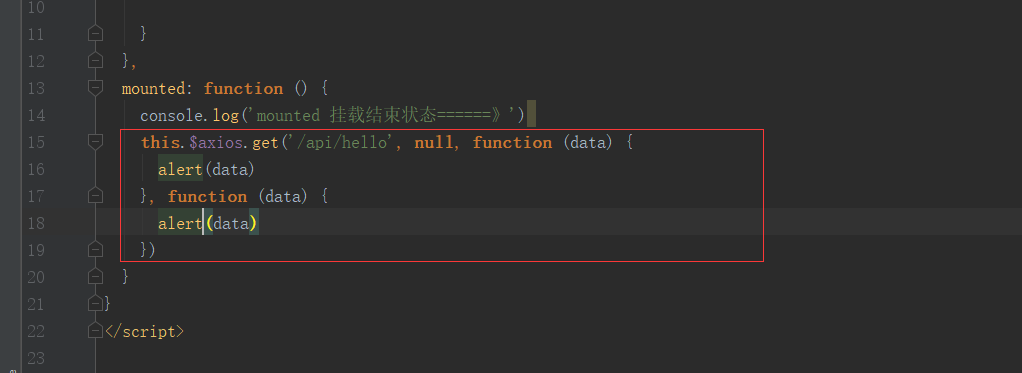
在Cmd中引入组件，并可以通过prop传递参数



## 前后端通讯

首先安装axio组件

cnpm install axios --save



axios.js

// 引用axios  
**import** axios **from** 'axios'  
**let** apiAxios = (method, url, params, success, failure) => {  
 axios({  
 method: method,  
 url: url,  
 data: (method === 'POST' || method === 'PUT' || method === 'DELETE') ? params : **null**,  
 params: method === 'GET' ? params : **null**,  
 headers: {  
 'Content-Type': 'application/json'  
 }  
 })  
 .then(res => {  
 console.log(res.data)  
 **if** (res.data.code === 200) {  
 **if** (success) {  
 success(res.data)  
 }  
 } **else** {  
 **if** (failure) {  
 failure(res.data)  
 } **else** {  
 window.alert(res.data.code + ':' + res.data.msg)  
 }  
 }  
 })  
 .catch(err => {  
 **if** (err) {  
 window.alert(err)  
 }  
 })  
}  
  
// 返回在vue模板中的调用接口  
**export default** {  
 get: (url, params, success, failure) => {  
 **return** apiAxios('GET', url, params, success, failure)  
 },  
 post: (url, params, success, failure) => {  
 **return** apiAxios('POST', url, params, success, failure)  
 },  
 put: (url, params, success, failure) => {  
 **return** apiAxios('PUT', url, params, success, failure)  
 },  
 **delete**: (url, params, success, failure) => {  
 **return** apiAxios('DELETE', url, params, success, failure)  
 }  
}