

演示项目-案例：<http://ihrm.itheima.net/hrsaas/dashboard>

模板别人开源的一个项目（它是用element+vue+pc管理系统 ）-地址https://panjiachen.gitee.io/vue-element-admin-site/zh/

完整版本参考：<https://panjiachen.gitee.io/vue-element-admin/#/nested/menu1/menu1-2/menu1-2-1>

完整版本功能：是一个后台前端解决方案，它基于 vue 和 element-ui实现。它使用了最新的前端技术栈，内置了 i18 国际化解决方案，动态路由，权限验证，提炼了典型的业务模型，提供了丰富的功能组件，它可以帮助你快速搭建企业级中后台产品原型。是一个后台集成方案, 集成了PC项目中很多的业务场景和功能, 尤其在当下SPA的趋势下,我们可以从中获得很多成熟的解决方案.

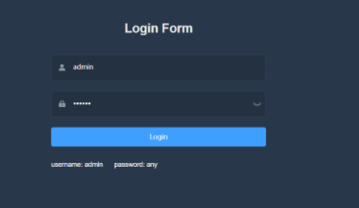
一个成熟的集成方案，里面包含了所有的业务功能和场景,并不适合直接拿来进行二次开发, 但是可以通过该项目中的一个案例来进行学习和使用。

集成方案并不适合我们直接拿来进行二次开发，基础模板则是一个更好的选择

项目目的：我们拿的基础模板, 包含了基本的 **登录 / 鉴权 / 主页布局** 的一些基础功能模板, 我们可以直接在该模板上进行功能的扩展和项目的二次开发

环境准备：1、$ npm -v #查看npm版本2、$ git --version #查看git安装版本3、$ npm config get registry #查看镜像地址

项目模板启动和目录介绍：1、**git拉取基础项目模板$ git clone https://gitee.com/PanJiaChen/vue-admin-template.git hrsaas #拉取基础模板到hrsaas目录 2、**安装项目依赖（定位到项目目录下）$ npm install #安装依赖 // 如果依赖有错误提示让安装 core-js $ npm i core-js -D -D:开发环境 -S:开发环境与生产环境都使用3、$ npm run dev #启动开发模式的服务启动项目



Mock数据：官网操作文档：<https://github.com/nuysoft/Mock/wiki/Getting-Started>

在线接口基地址:http://119.91.150.211:3000/api

接口文档：<https://www.apifox.cn/apidoc/shared-fa9274ac-362e-4905-806b-6135df6aa90e/api-23869528>

跨域浏览器(了解)：创建一个可以跨域的谷歌浏览器，去掉了浏览器的同源策略

1. 创建一个文件夹 c盘创建一个xxx文件夹
2. 创建一个谷歌的快捷方式，将谷歌复制粘贴 一份出来
3. 创建的谷歌快捷方式右键=>属性=>目标，在目标最后面添加一段代码
4. --disable-web-security --user-data-dir=C:\xxx

注意代码前面要有空格

点击确定再打开就是一个跨域浏览器了

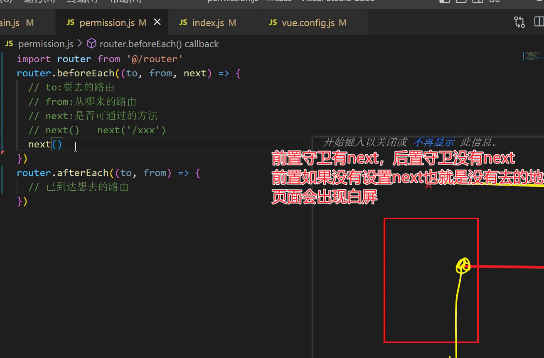
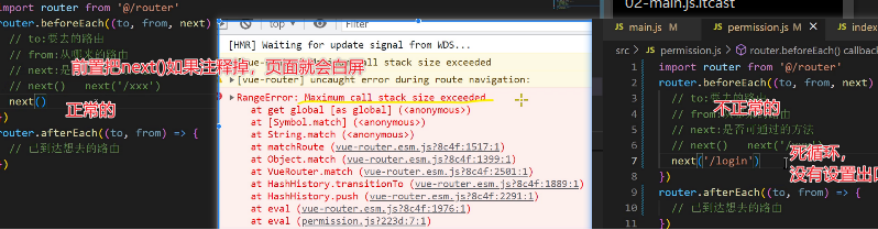
问题：还是插件配置的问题 vue2人资项目

很多公司移动端按需引入，pc端很多都是全局引入的模式。对于网络不是瓶颈。

优化可能是20-30k.人资是全局引入，

看到白屏：第一步先看导航守卫。第二步：看路由的地址是否存在。

导航守卫:前置守卫和后置守卫。



做一个项目有些可以了解，有些可以暂时不用了解。要了解路由，api,vuex。后面公共组件和公共方法。公共组件： ，公共方法：知道有什么方法。找一个页面模仿着写：看这位置怎么写。项目可能赶的话1-2天做事。有个缓冲器。进去企业一定要问有没有坑点。

**自定义规范**

插件:Prettier ESLint ,注意不要安装最新版本的，安装v3.1.0

* 配制vscode,直接在配制文件中粘贴下面代码

"[javascript]": {

"editor.defaultFormatter": "rvest.vs-code-prettier-eslint"

},

"[html]": {

"editor.defaultFormatter": "rvest.vs-code-prettier-eslint"

},

"[vue]": {

"editor.defaultFormatter": "rvest.vs-code-prettier-eslint"

},

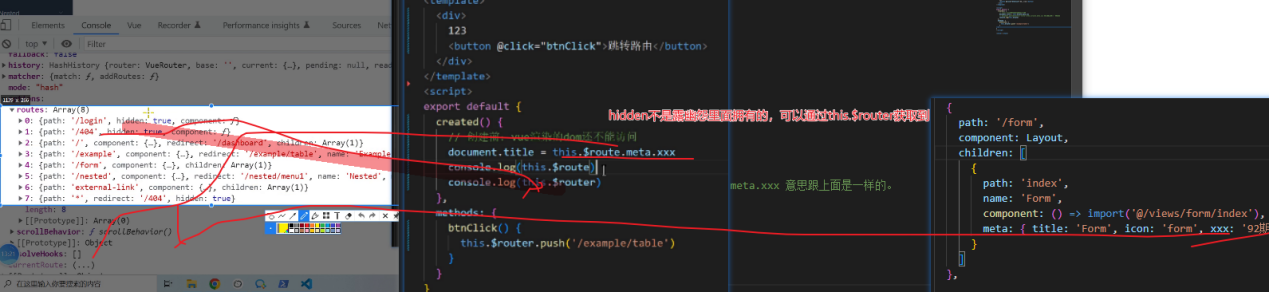
"[scss]": {

"editor.defaultFormatter": "rvest.vs-code-prettier-eslint"

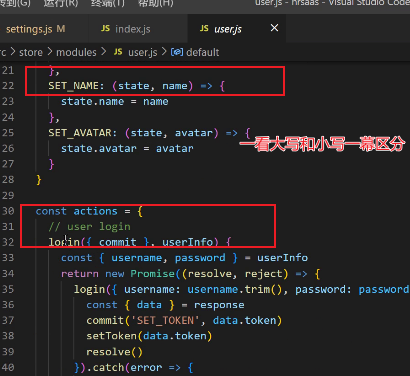
},

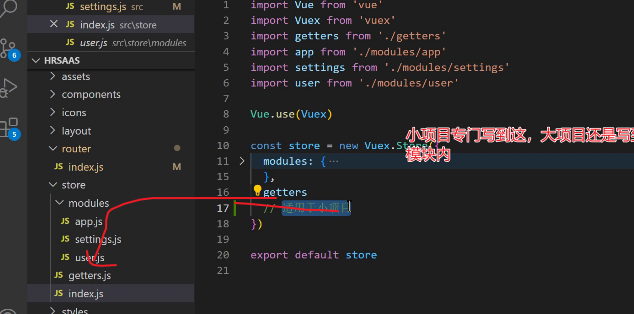
This.$router是获取整个路由配置

meta是直接在路由信息里面显示。



store写到模块块里面，避免了命名冲突。Namespaced:true主要是针对mutaion

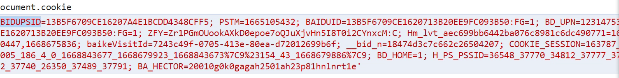




Mutatio：里面是方法大写。action里面是小写,区分开。

持久化：1、值修改了存储到浏览器2、初始值读取浏览器存储。

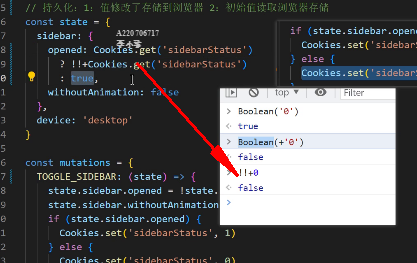
之前用的本地储存。现在用的cookie，有插件.cookie优势：相当于本地储存，区别：cookie有长度限制，本地储存很大的内存无限制2、本地储存有有自己的api，cookie修改很麻烦的，document.cookie，拿到的是一堆字符串。想要去cooie就需要分割，所以就需要一个加减cooie解决这个问题，另外cooie有有效期，可以通过浏览器看到，通过Js代码获取不到

、

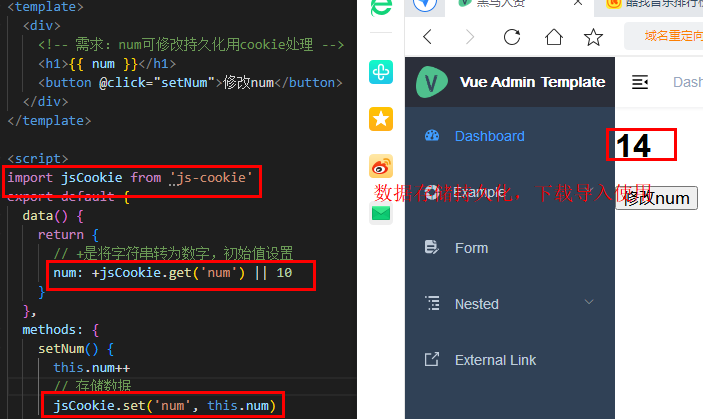
、cooie主动参与接口请求，同源的才会主动传，不同源的不会去传。

Js-cooie下载安装使用 js-cookie-----get(key) set(key,value) remove(key)

本地储存存的都是字符串，所以后期肯能需要转换。 数字转boolean



Js-cookie1、下载2、导入3、使用set,get,remove.

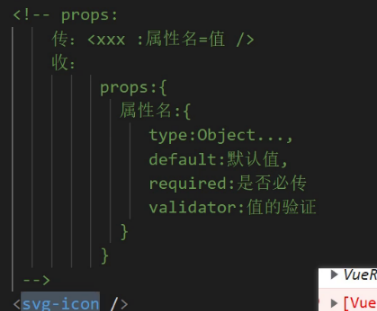


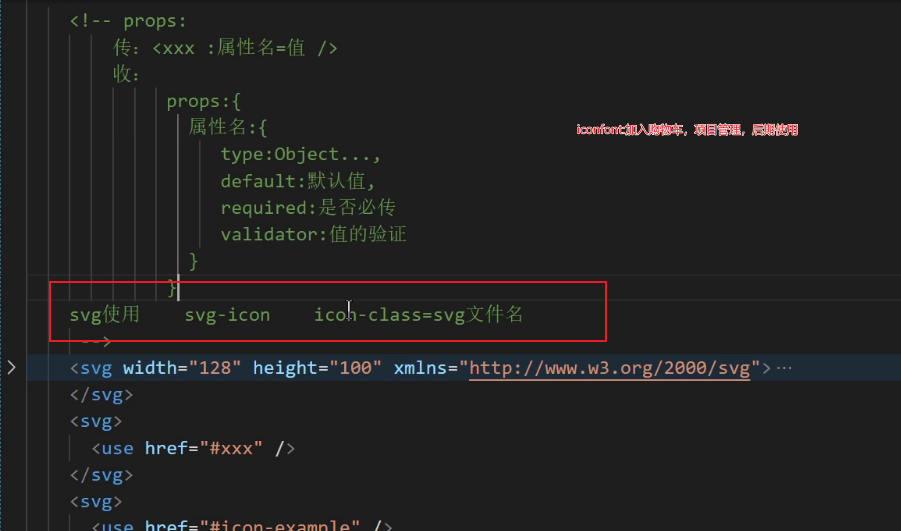
注释规范，代码规范，命名规范。



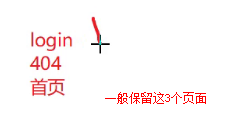
webpsck加载规则，首先找icon.js=>icons.vue==>icons文件夹（package.json，相关配置对应上）=》icons文件夹/index.js===>icons/index.vue

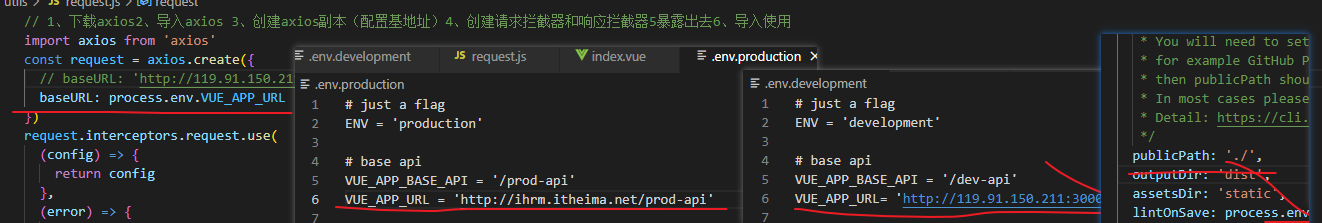






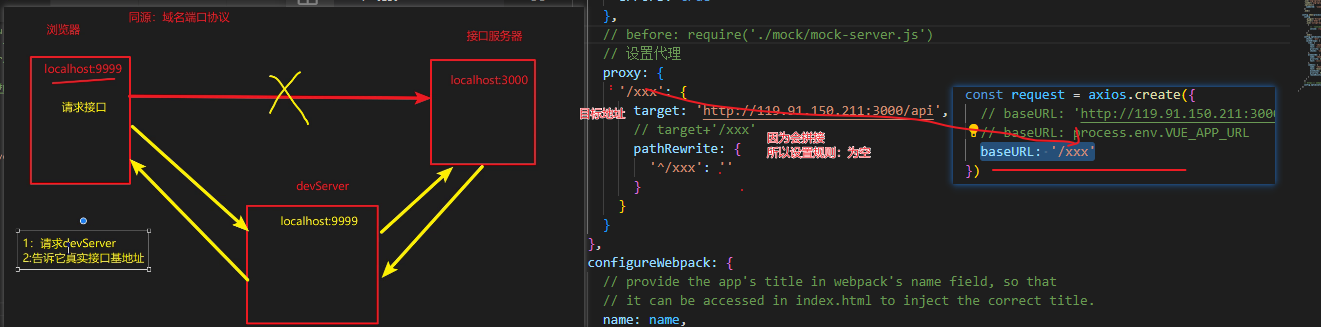
公司用过的东西没见过，先了解怎么用。





基地址设置环境变量：。

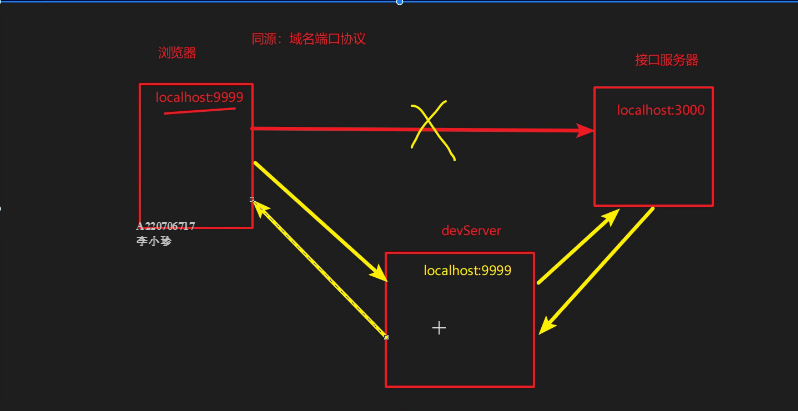
跨域：最好用的还是设置跨域浏览器，当然主流的方式还是开发环境配置proxy代理解决跨域。



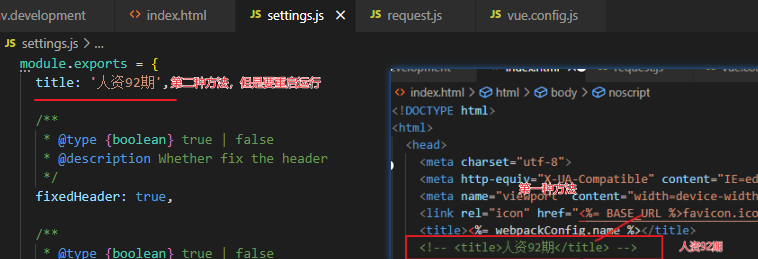
跨域proxy开发环境服务器配置解决2部曲：1、设置基地址2、基地址proxy设置目标地址,规则改变。每次改变记得重启运行。

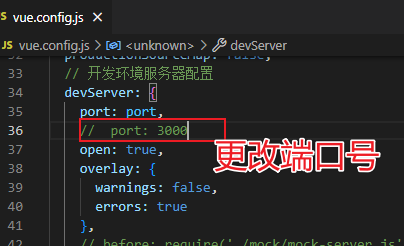
只是设置开发环境，到生产环境是后端解决的。

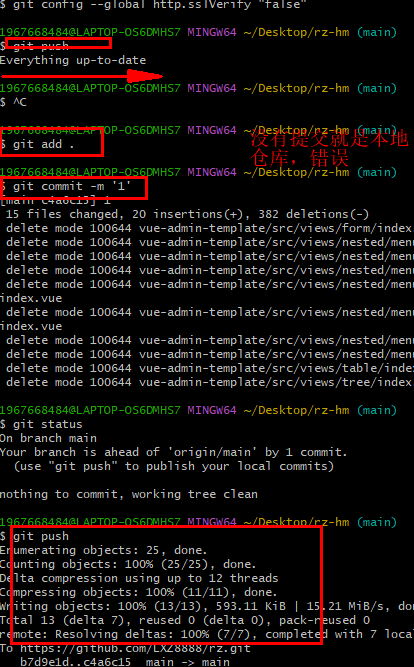
开发环境打包：改路径’./’然后运行得出dist文件给运维。



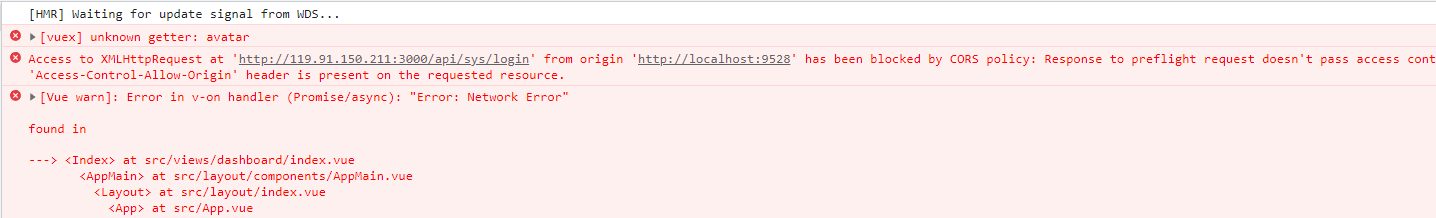








1、下载axios2、导入axios 3、创建axios副本（配置基地址）4、创建请求拦截器和响应拦截器5暴露出去6、导入使用

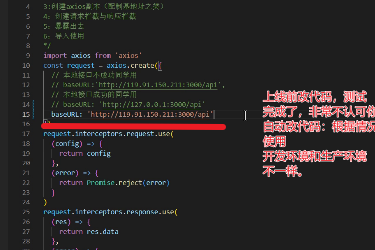


后端加：cors

前端：jsonp,代理中转，跨域浏览器

404请求写错了。

上线前把接口基地址换成真实接口基地址。





设置环境变量：

开发环境：

根目录创建一个文件.env.development

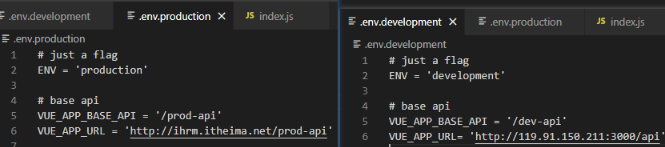
VUR\_APP\_开头 变量名VUR\_APP\_NUM=1

生产环境

根目录创建一个文件.env.production

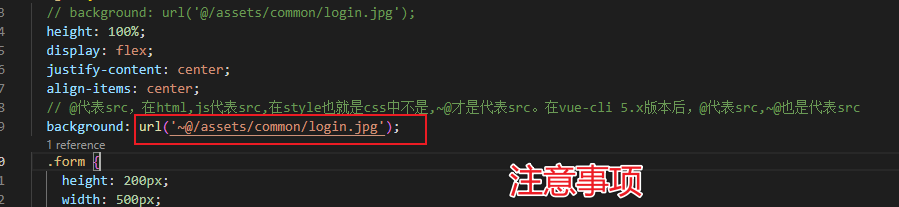
VUE\_APP\_NUM=2

Process.env.xxx



配置基地址:生产环境和开发环境。

有个问题：跨域浏览器时灵时不灵



Scoped:只管当前组件和子组件最外层，所以需要::v-deep

第二天：cookie操作，持久化，环境变量，代理proxy

基本表单验证：1、表单的值绑定 el-input绑定 2、el-form model:整个 表单的数据 rules：规则 3、el-form-item prop=”需要验证项目的字符名”

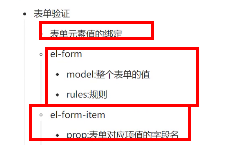
Validator:(rule,value,callback)=>{}

全局表单验证：

:model="form"和v-model的区别

1、v-[model](https://so.csdn.net/so/search?q=model&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_42333361/article/details/_blank):

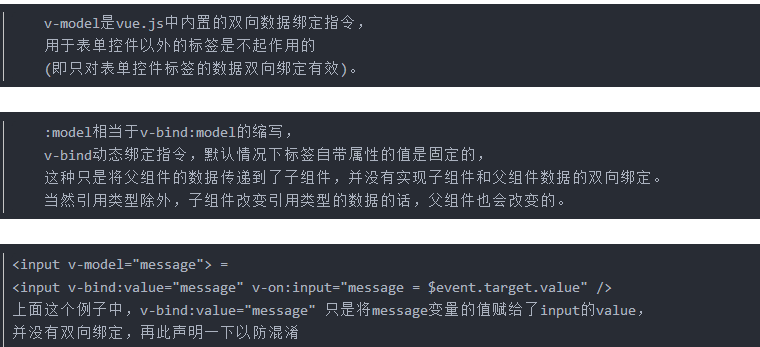
通常用于****表单上双向数据****的绑定，如果除了表单其他组件使用时，起不到任何效果。它还可以实现****子组件到父组件****的****双向数据动态绑定****。



2、:model

:model是****v-bind:model****的缩写。

只是将****父组件的值传递给和子组件****，但是并****未实现****子组件  
和父组件之间的双向数据绑定，当然****引用类型除外****，子组件改变了引用类型的数据的话，父组件的数据也会跟着改变。

对象参数命名最好使用接口文档里面的。

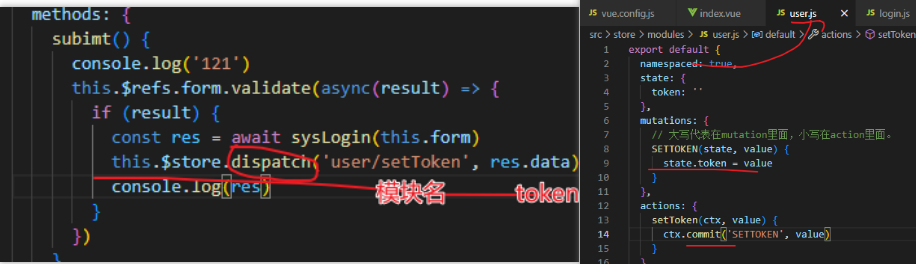


正则表达式一个插件：Fehelper,；路径别名插件 path-alias

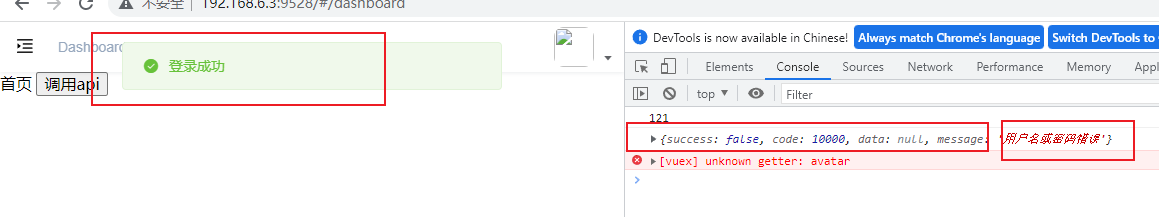


完成登录：1、先搞定静态页面然后搞定校验规则和数据2、定义接口api方法发送请求。2、导入api方法，3、点击登录在表单验证通过后调用api方法。（保存token）跳转到首页。

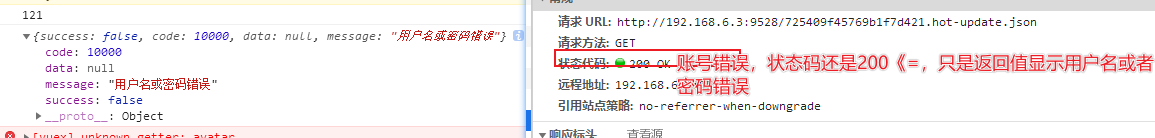
正规的网站不是直接保存到localstorage或者cookie上面，而是保存到vuex。优点：vuex有相关变量，vuex一刷新数据就没了，所以在vuex持久化localstorage。token保存到vuex.



持久化：持久化：1、值修改了存储到浏览器2、初始值读取浏览器存储。cookie和localstorage差不多作用。vuex的token处理，token的存储

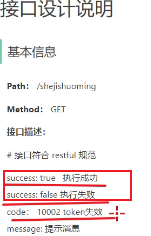


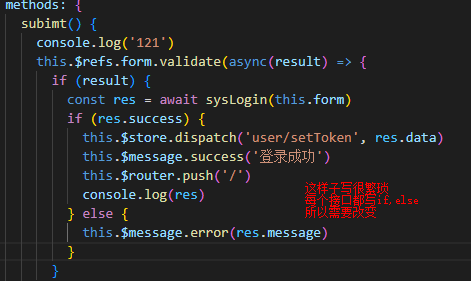
理想状态下能登录成功。密码错误也能登录所以需要进行处理。

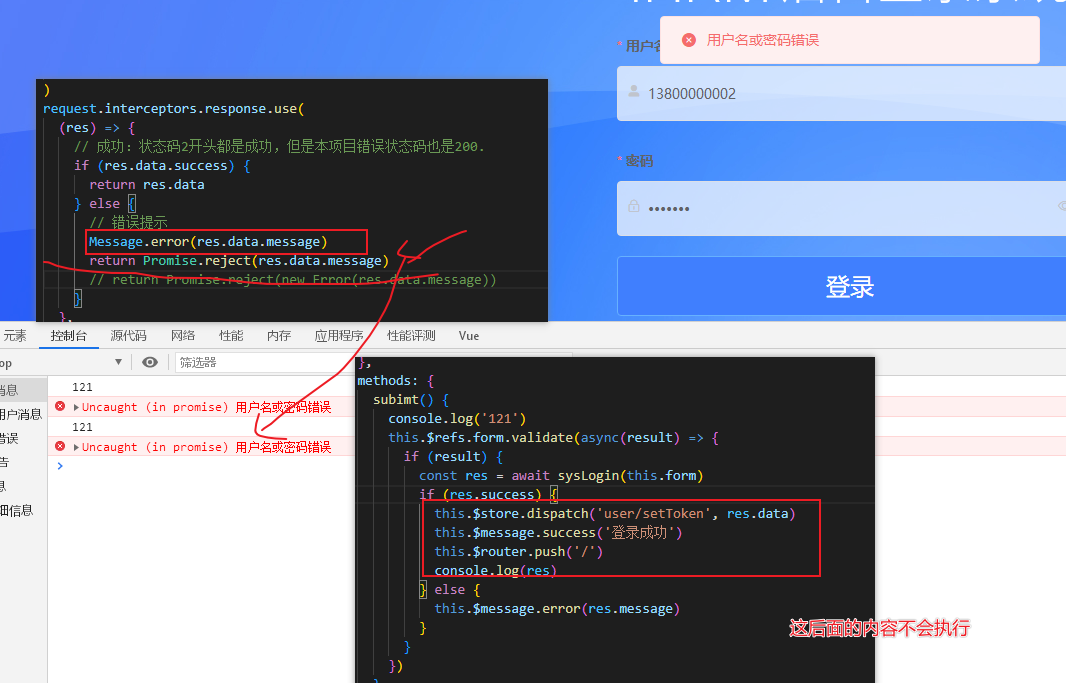


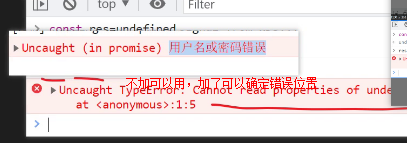


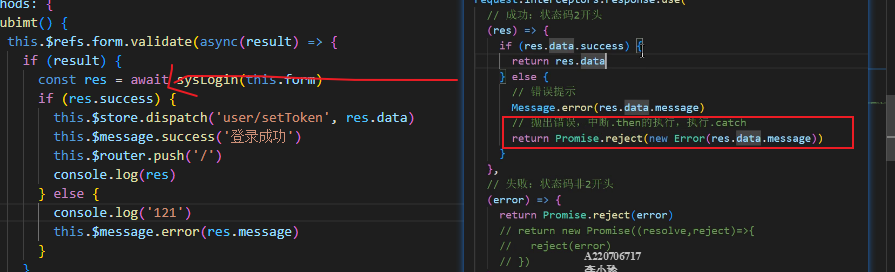
看请求或者接口文档说明会设置请求规范。

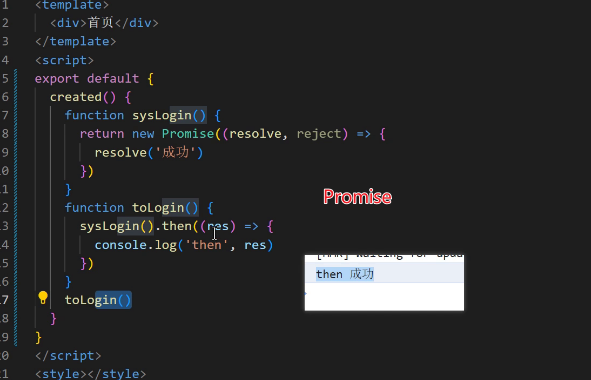


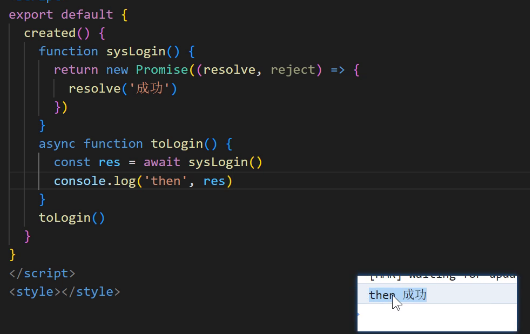




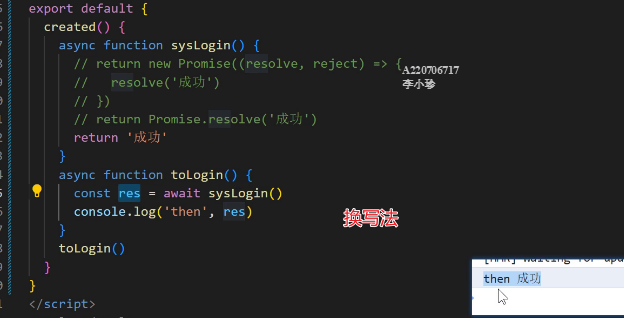


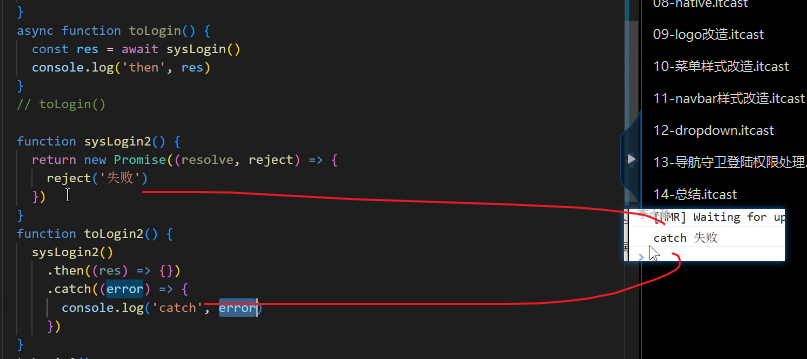












promise还是需要多加练习。

登陆回车处理。原生：@keyup.enter，但是不能在组件作用.原生有些能用，有些不能用。

组件身上使用原生事件可以通过.native修饰符使用。

如果在组件标签上加上.[native](https://so.csdn.net/so/search?q=native&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/xiaolinlife/article/details/_blank)修饰符，就相当于直接拿组件标签当做原生的html标签在使用了。



native 是什么？

.native - 监听组件根元素的原生事件。

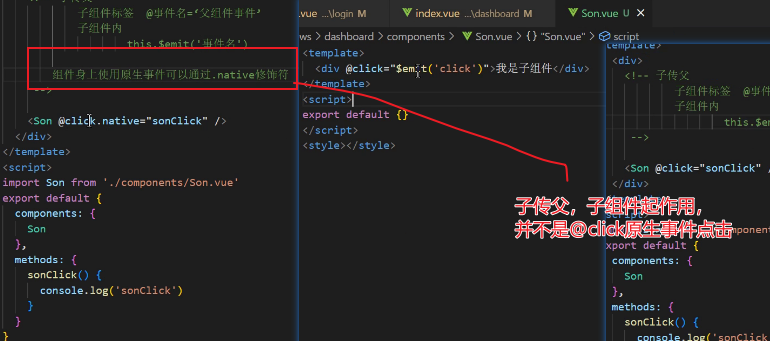
主要是给自定义的组件添加原生事件。

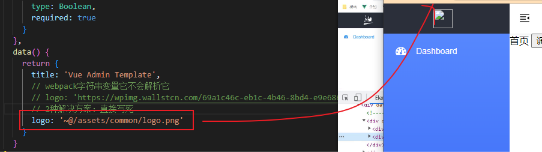
比方说:你想在某个组件的根元素上绑定事件，直接使用 @click=''function' 是不生效的，我们可以添加.native修饰符 @click.native=''function''

————————————————

版权声明：本文为CSDN博主「chengxu猿」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

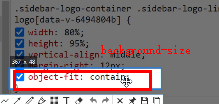
原文链接：https://blog.csdn.net/qq\_21013249/article/details/120852335





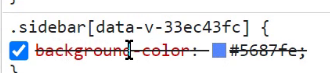


在线图片不需要解析，本地图片需要解析。require和import的使用方法。



  object-fit: contain;相当于background-sizing:contain

样式不生效原因一般两种:1、权重不够2、没选中。权重不够。



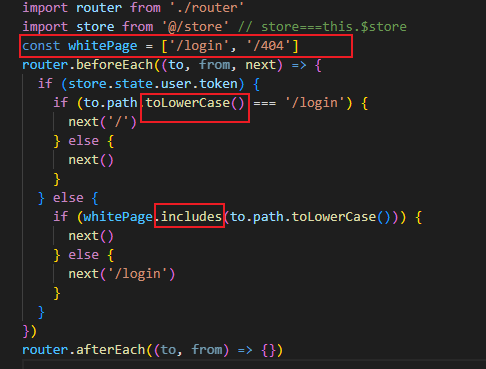
svg调颜色：

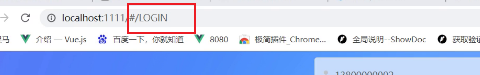
native还是很重要的，组件采用原生事件.native变。



url不认大小写。浏览器不区分大小写。

白名单:[‘121’,’1212’] includes()是否包含 true 或者是false。转换为小写.tolowercase()

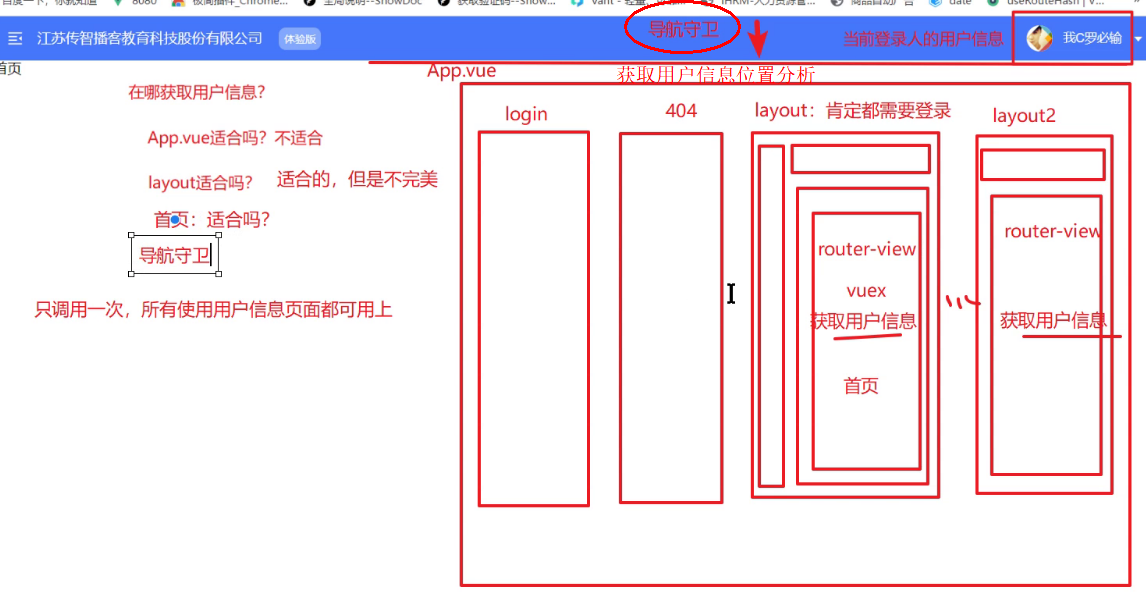




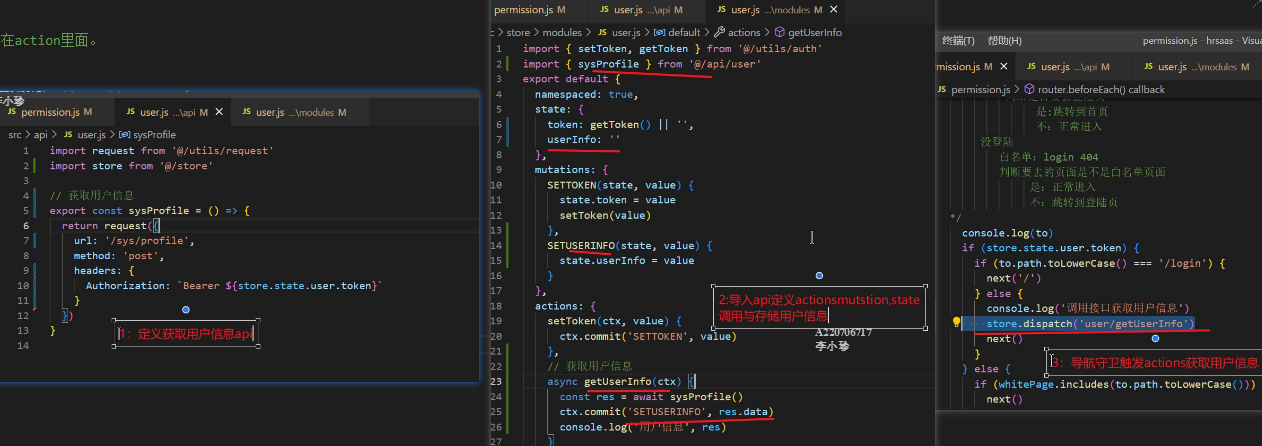
Promise有很多应用.async await都是基于promise应用的。 Prmoise.all(axios1(),axios2() )返回值是promise

特点：它对应所有内部代码执行都是一起进行的。如果都完成执行.then代码。如果有一个出错执行.catch。就算出错也希望执行.then(res=>res).the ((res2=>{})).catch((res)=>{})

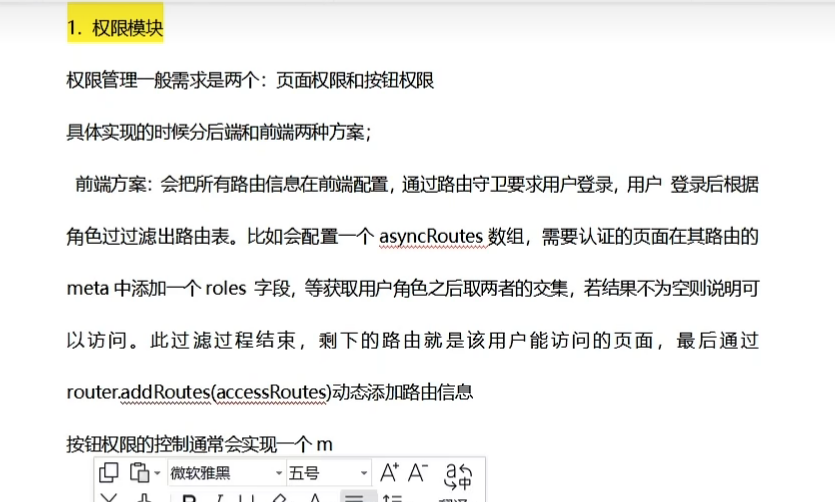
.catch返回值是promise。

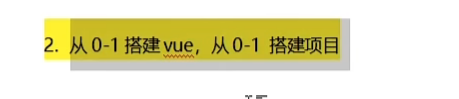


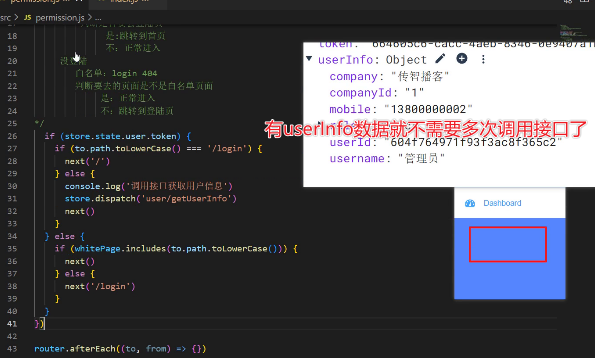
用户信息获取 1、定义接口api 2、在Vuex中导入api 3、在action中定义方法获取用户信息存储到state 4、在导航收尾中调用触发

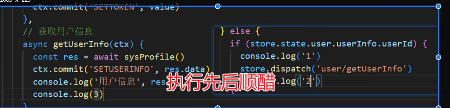


用户信息更改不适合做持久化。



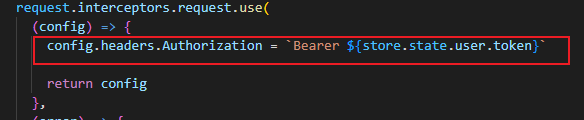






加了async返回值就变成了Promise，加了await就等它执行完才执行后面的东西。

是否所有接口都加了token，有些接口不想要加token。



接口分为2类。1、接口不需要token,你传了不报错。解决：统一全加token

2、接口不需要token,传了报错。解决：