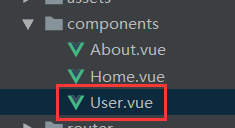
# Vue-router动态路由

## vue-router动态路由的使用

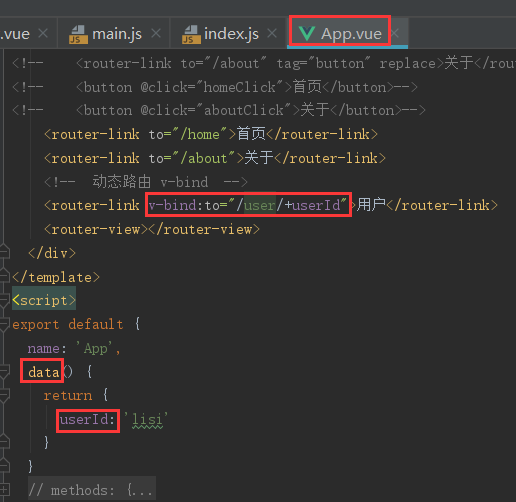


1. 创建组件，并添加内容

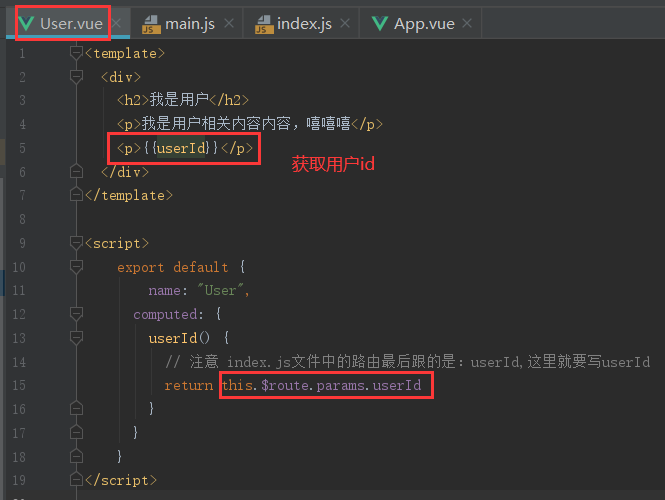
 

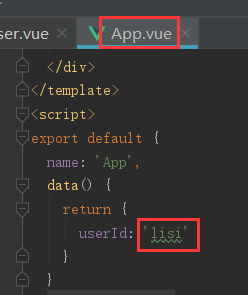
1. 动态路由，在URL后面添加用户id，使用 v-bind: ，可以简写一个冒号:

路径注意单双引号



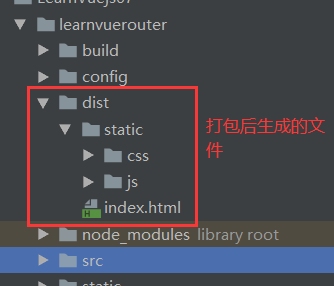
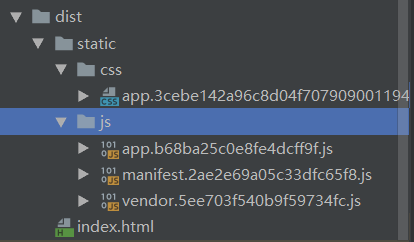
1. 获取用户id并渲染，使用$route

 或这样写 

页面运行结果： 

## vue-router打包文件的解析

打包项目命令npm run build,比较未使用脚手架创建的项目，打包后的文件都放在bundle.js文件中，而且文件大，页面加载耗时，而使用脚手架创建的项目，打包后的文件分成了css和js两个文件夹

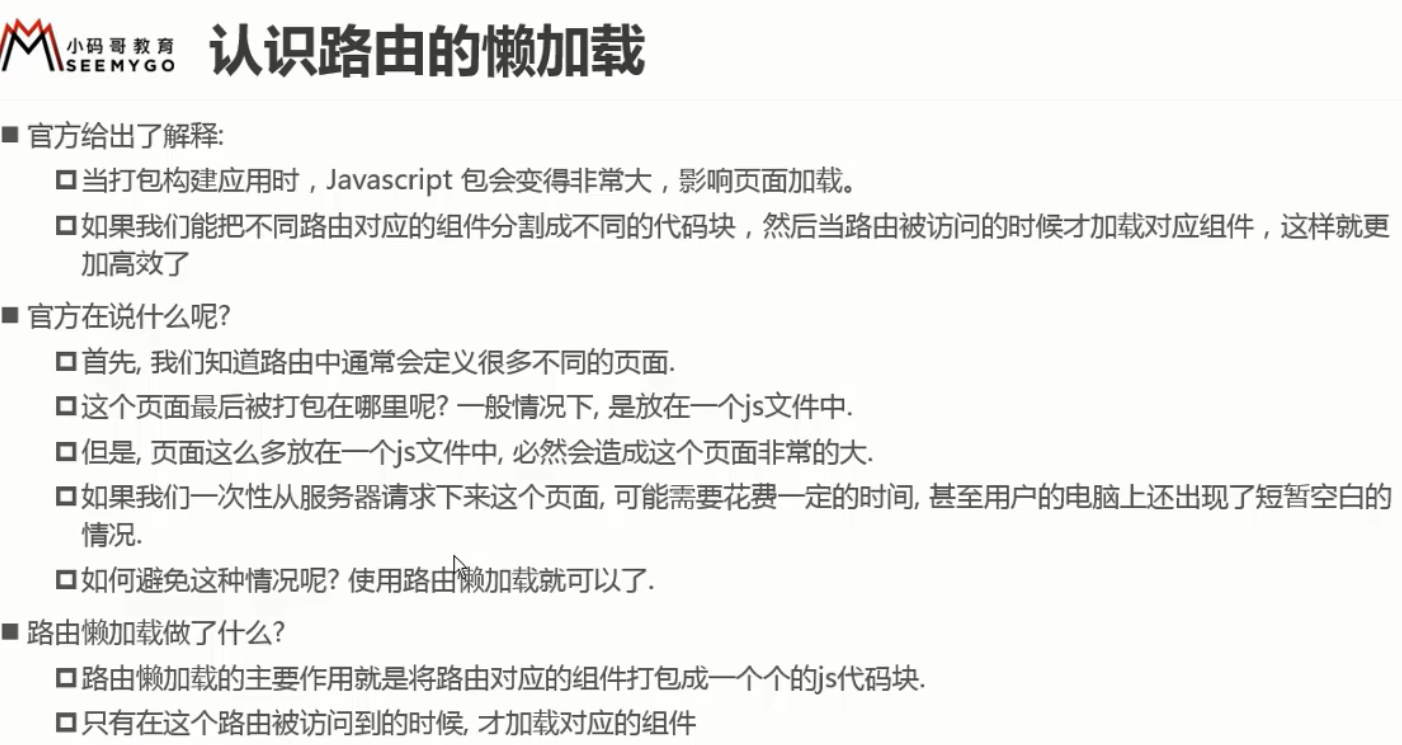
 

打包的文件解析：

* 【app.js文件】：当前应用程序开发的所有代码（或业务代码）都被打包放在此app.js文件
* 【vendor.js文件】：vendo（提供商、第三方，引用第三方的东西，如vue、vue-router、axios等）第三方的东西都被打包到vendor.js文件
* 【manifest.js文件】：为了打包的代码做底层支撑，如导入导出一些文件

## vue-router路由懒加载的使用

懒加载：用到时，再加载。



【懒加载的效果】



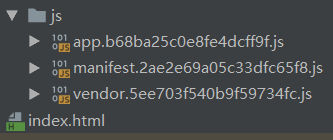
【懒加载的方式】

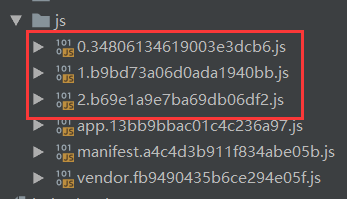




或者 

但是建议使用第一种，先定义变量，再使用变量。这样代码更清晰。

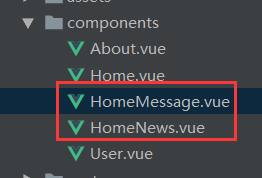
打包前有这三个js文件

npm run build打包后，有几个组件，打包后就会多生成几个js文件，因为一个组件对应一个懒加载，一个懒加载对应一个js文件。

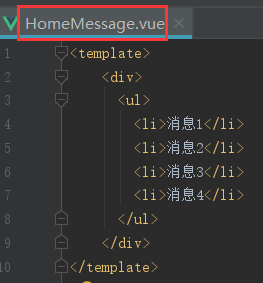
## vue-router路由的嵌套使用



1. 创建组件（即子组件）HomeNews.vue 和HomeMessage.vue2个组件。



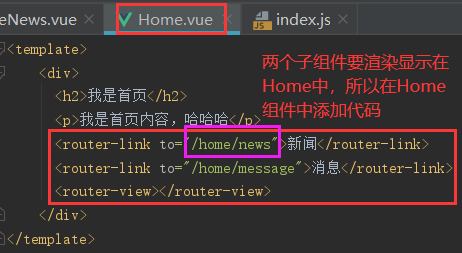
往2个子组件添加内容

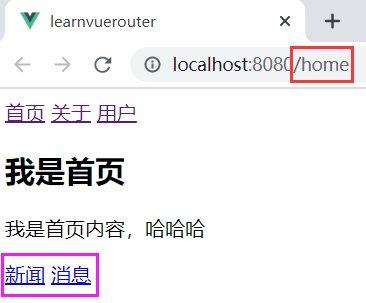
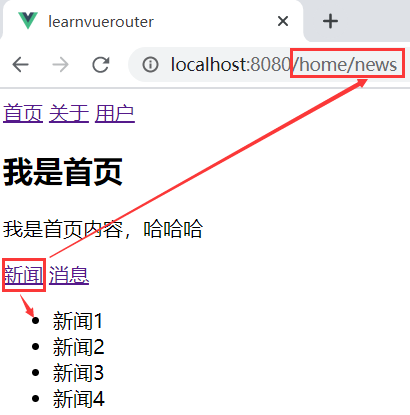
注意，因为新添加是Home组件的2个子组件，所以再配置路由也要在Home的里面

子组件子路由的path不需要加/这就是嵌套路由。注意，使用懒加载就不能再导入组件，不然会报错。

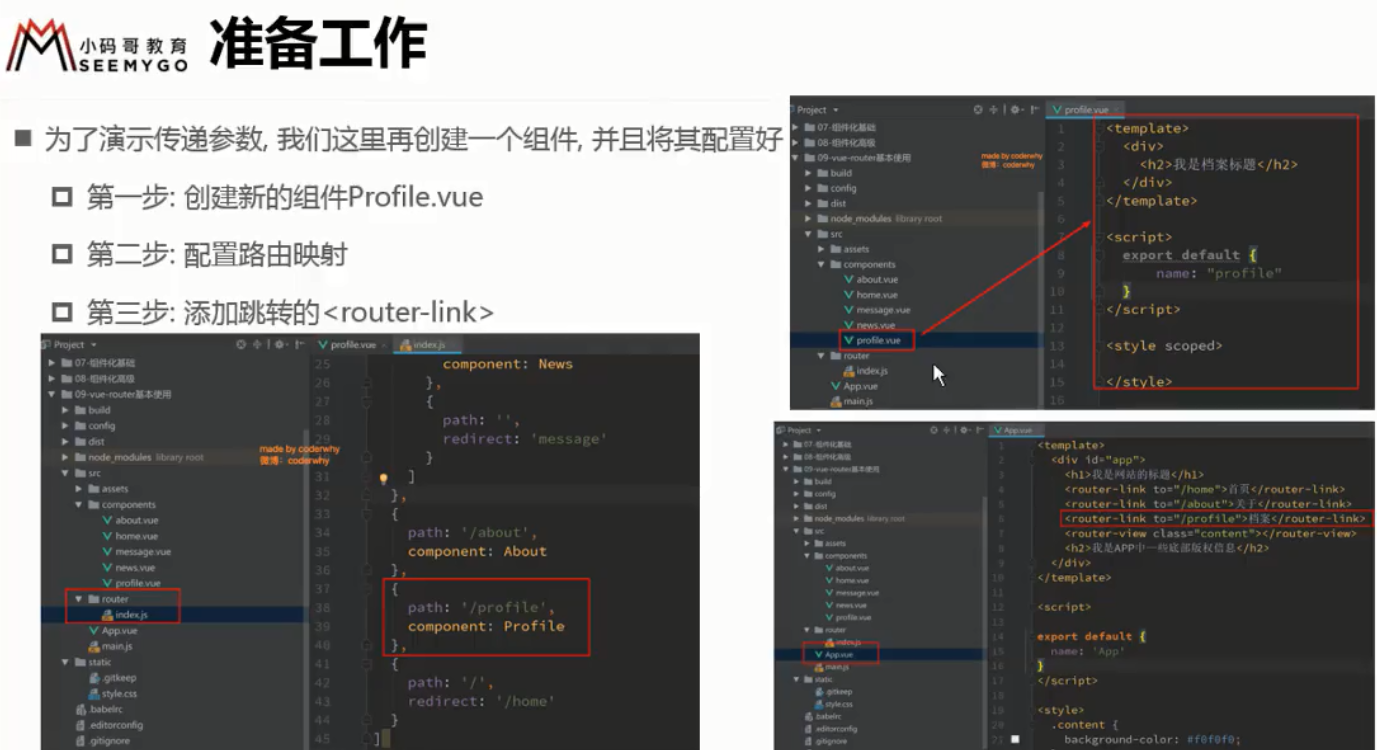
1. 在父组件Home中插入<router-view>标签

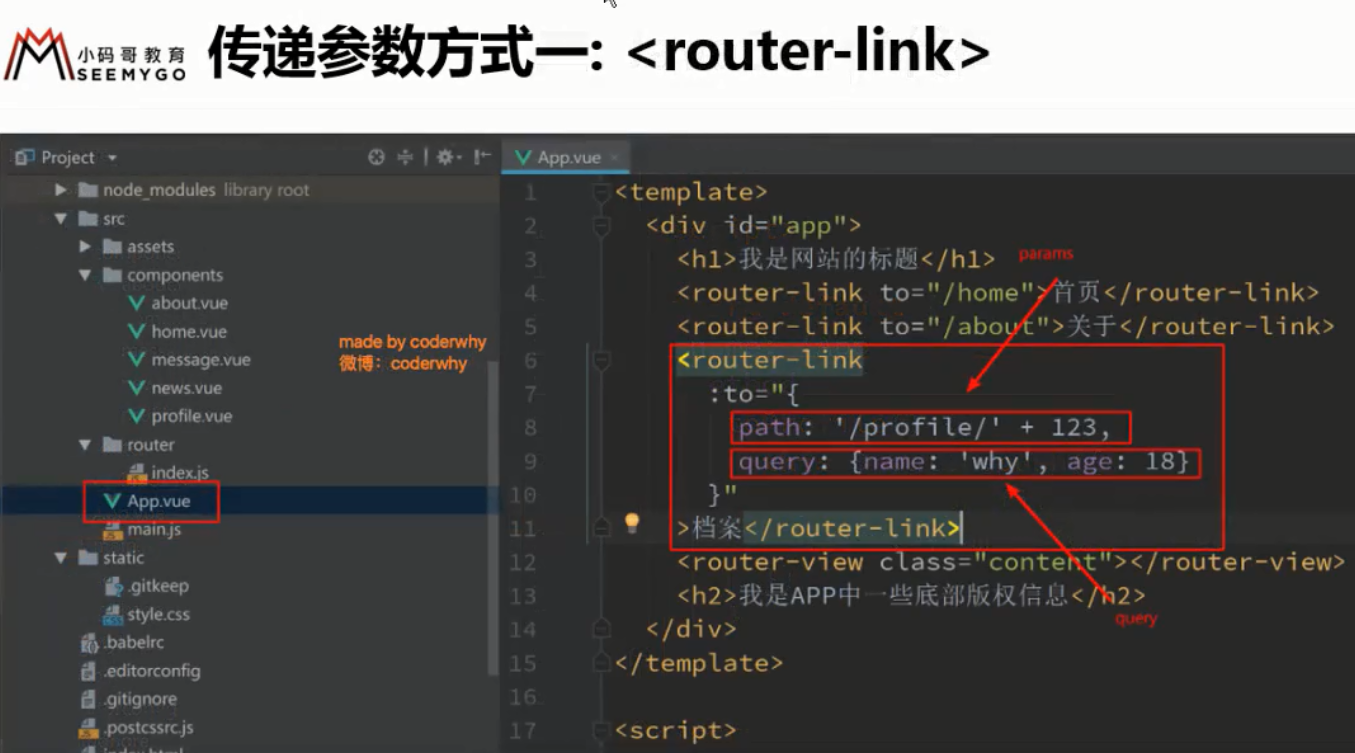
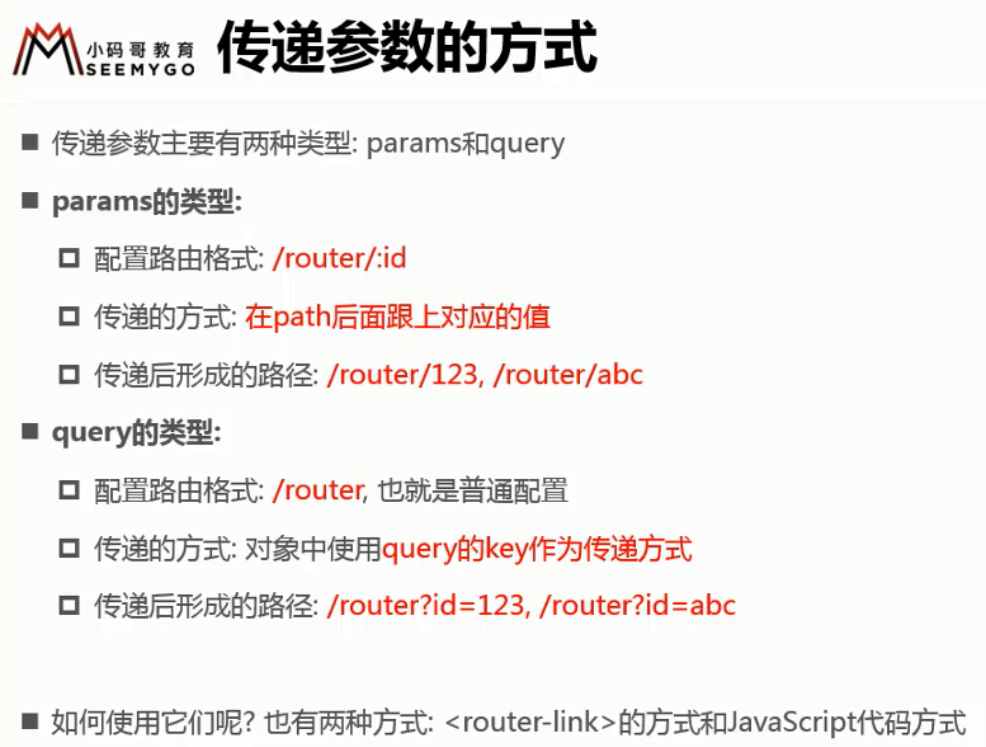


npm run dev 运行项目，页面结果：

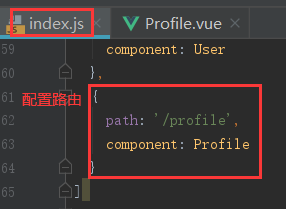
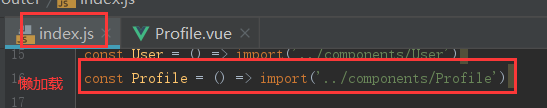
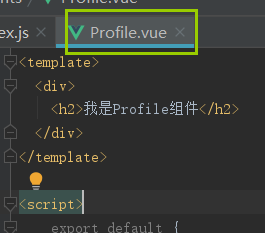
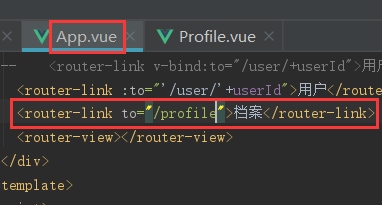
  

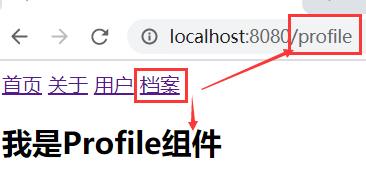
## vue-router参数传递（一）

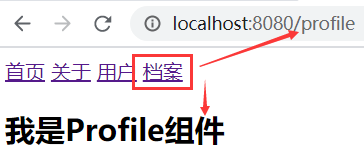




1. 新建组件profile

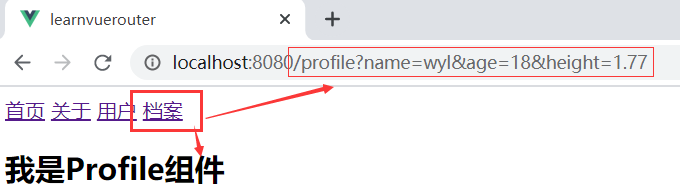
渲染到页面👉

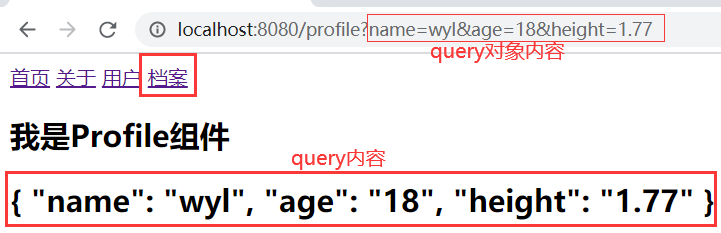
页面结果：

传递参数也能运行成功

Query类型：



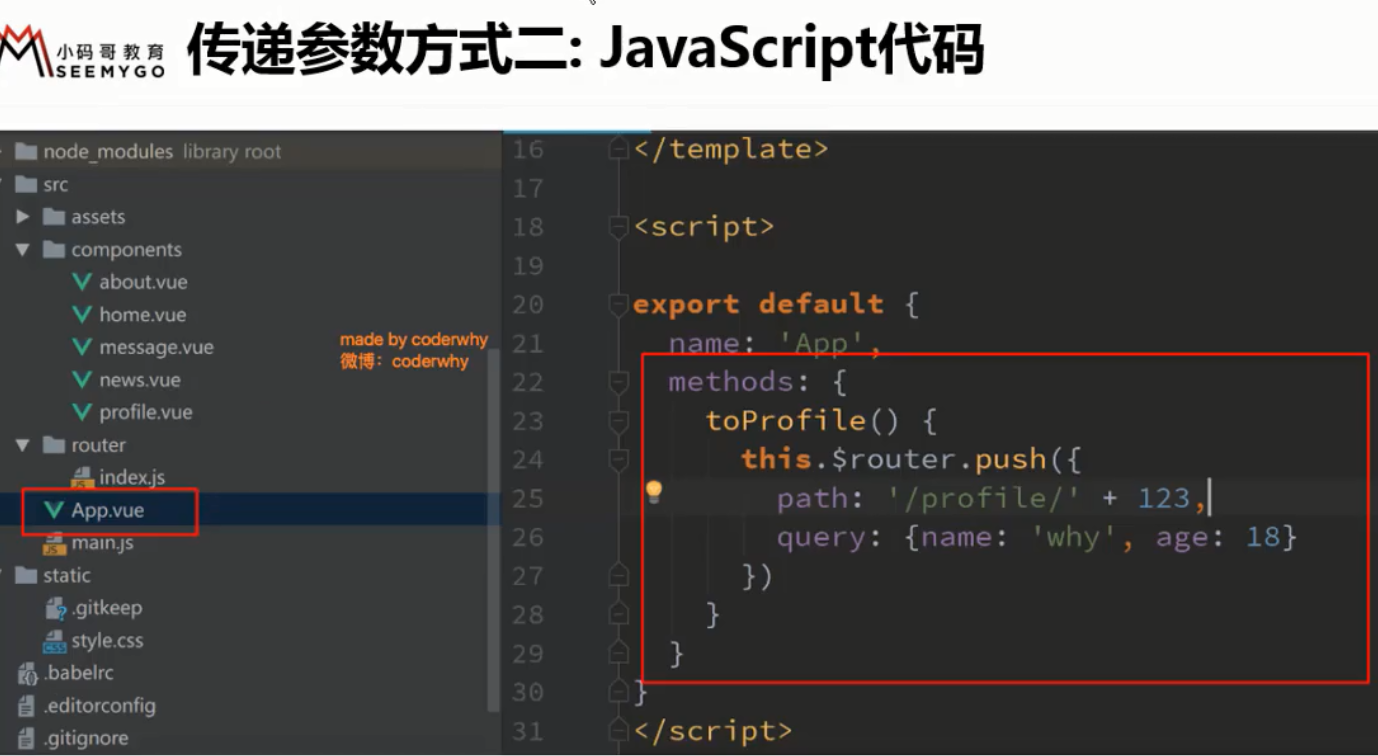
运行结果

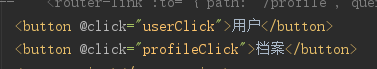
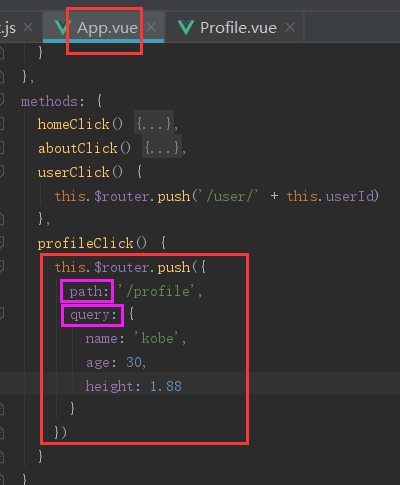
页面运行结果

取出query中的某个内容



## vue-router参数传递（二）



不使用router-link标签和 :to 来传参，使用button来传参

## vue-router-route和route的由来

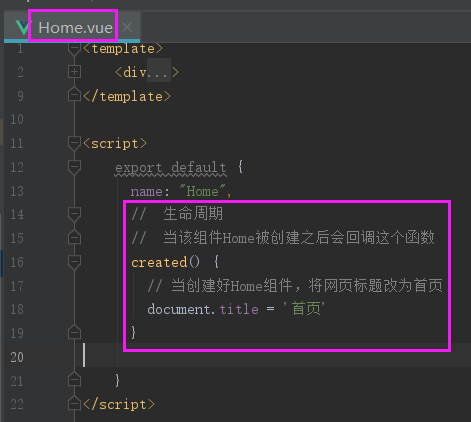
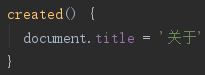
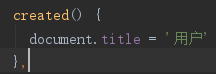
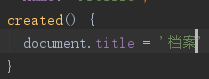




* 任何组件中的this.$router 都是index.js里创建的router对象，任何组件都可以使用this.$router。
* this.$route就是对应的组件创建的路由对象。
* 所有的组件都继承自Vue类的原型。

## vue-router全局导航守卫

导航守卫：在页面切换时做一些监听，也是做一些回调函数（钩子），

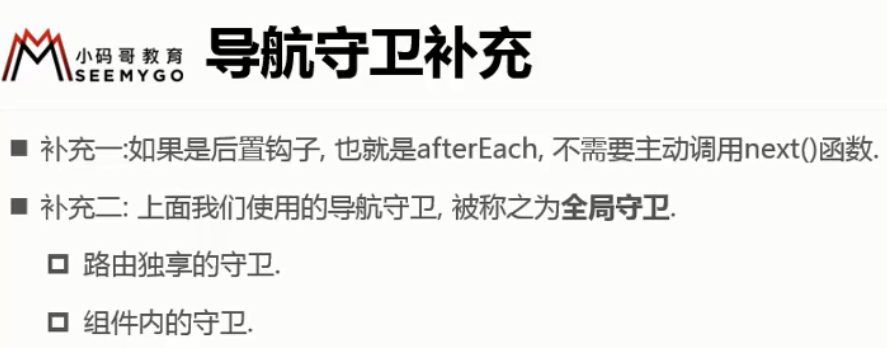
但是这样写每个页面都要添加这段代码，比较麻烦

高效做法是监听路由的变化，使用全局导航守卫

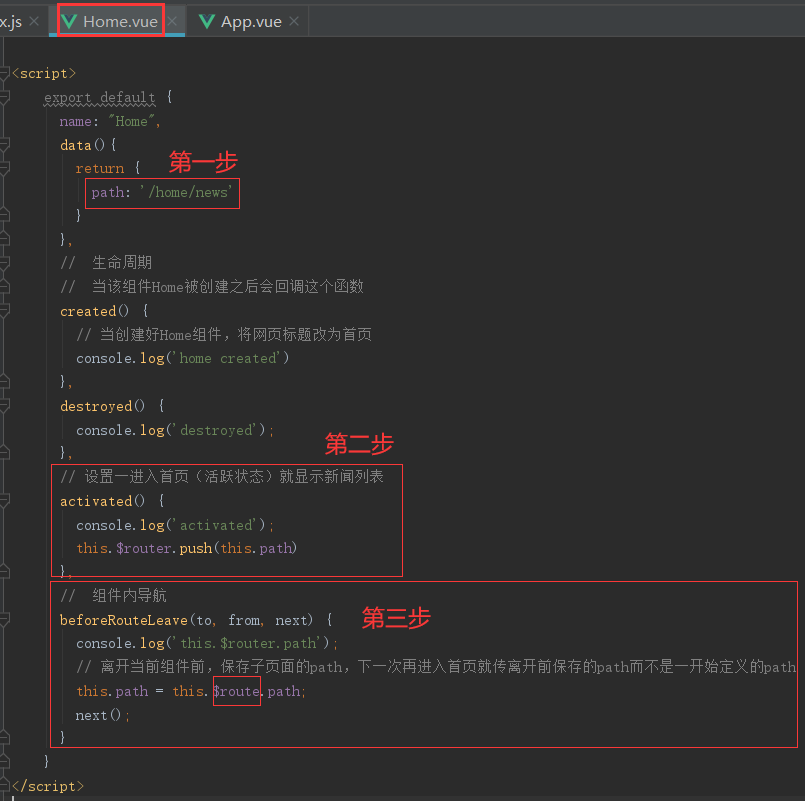
next()中可以跟URL，如next('/login)用户只有登录了才能进行下一步

## vue-router导航守卫的补充

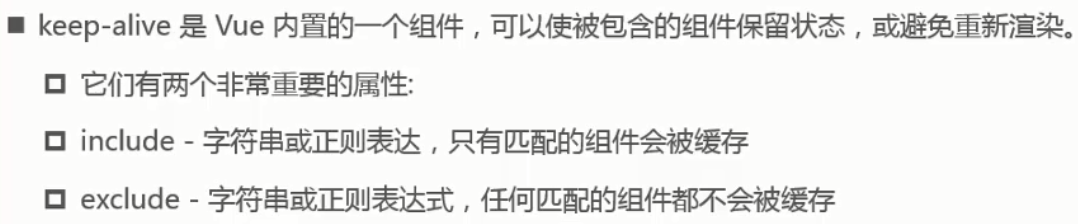
## keep-alive遇见vue-router

* 因为每次点击按钮切换页面，页面都会在创建后再销毁再创建新的一个（生命周期），而新闻列表被设置成默认的，所以每次都是显示新闻列表。修改：把【router-view】标签放在【keep-alive】标签中，提高了性能。使用keep-alive，created回调函数只会被调用一次。

进入到哪个页面，哪个页面就处于活跃状态

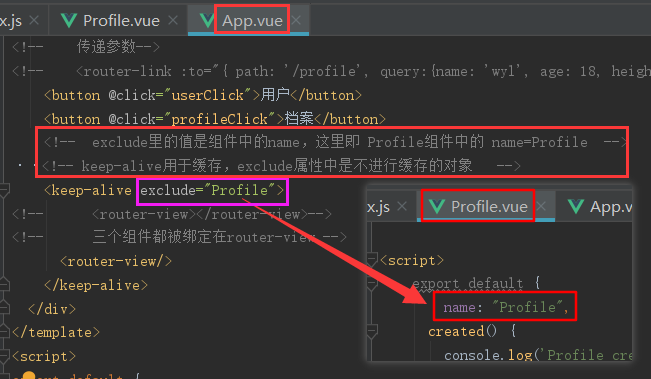
* activated和deactivated这两个方法跟keep-alive挂钩，如果没有写keep-alive，那么activated和deactivated这两个方法不会调用。
* 保留组件状态的操作，即当用户点击【首页】时显示的是【消息列表】，点击关于、档案后再点击首页时，页面还是保持显示【消息列表】。





## Vue-router-keep-alive属性介绍

需求：每次切换到【档案】，都会被重新建立，离开则销毁，再进来再创建。做法如下：使用exclude属性



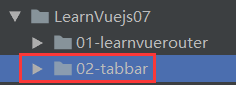
有多个不想被缓存，exclude中用逗号隔开，但是不要加空格

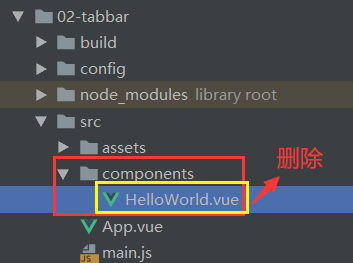
## tabbar基本结构的搭建

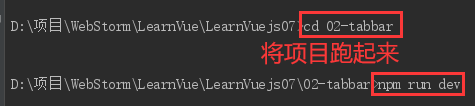
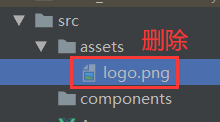
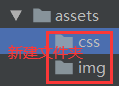


1. 创建vue-cli2项目，创建命令：【vue init webpack tabbar】

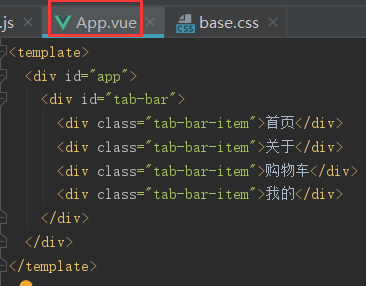
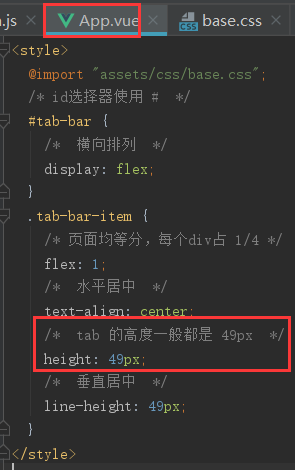


1. 重命名为02-tabbar

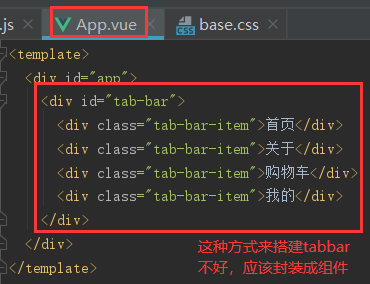
删除掉没用的标签和导入、组件、样式

1.   
2.   

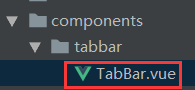
但是建议把base.css的引用写在App.vue中，不要在main.js中引用css文件

 让4个div在一行均匀分布

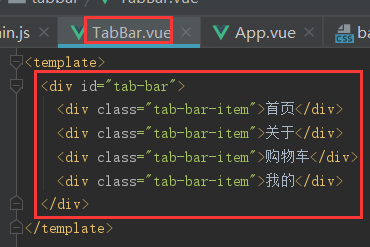
## tabbar-TabBar和TabBarItem组合

无复用性

1. 新建文件夹tabbar，便于管理。创建组件TabBar

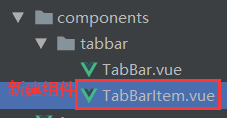
 

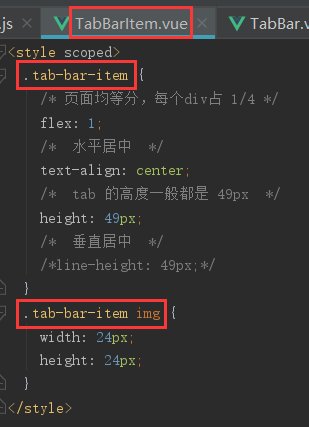
1. 把App.vue中写的组件剪切到TabBar

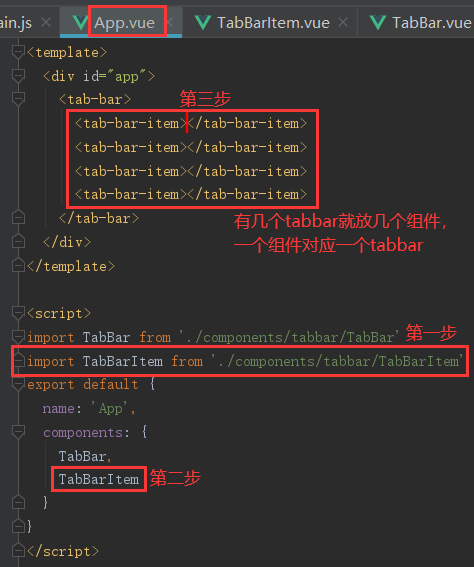


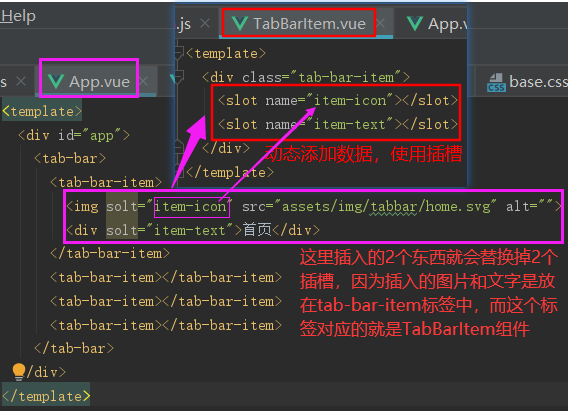
1. 使用组件，刷新网页正常显示。



1. 抽离繁琐的代码，写到另一个组件中。新建组件TabBarItem
2. 
3. tab-bar-item的标签和样式抽离到TabBarItem组件中

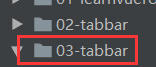
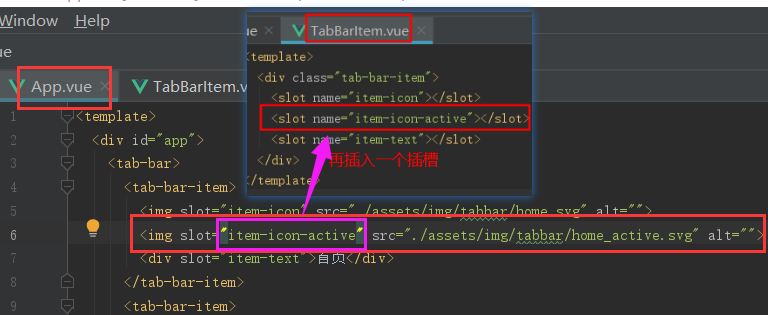
 

1. 
2. 动态插入东西🡺使用插槽<slot>

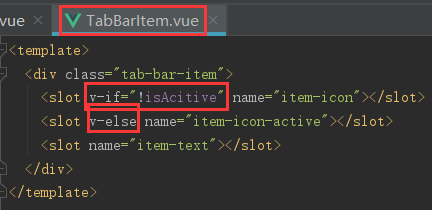
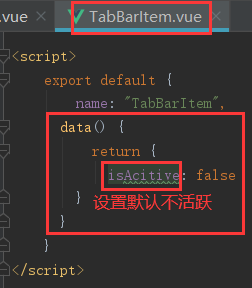
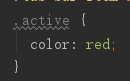
具名插槽会根据名字取找到对应的插槽

## tabbar给TabBarItem传入active

1. 复制02-tabbar，粘贴03-tabbar
2. TabBarItem组件中再放一个插槽来放活跃时的图标，App组件中则传入活跃时图标的路径

但是页面显示的时这样的，还需调整



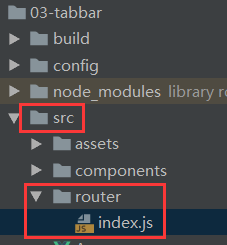
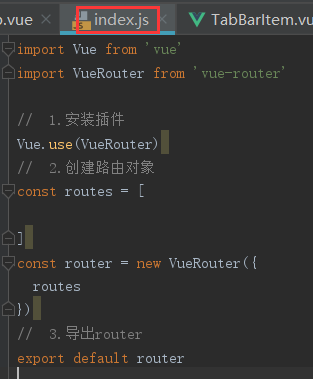
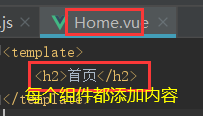
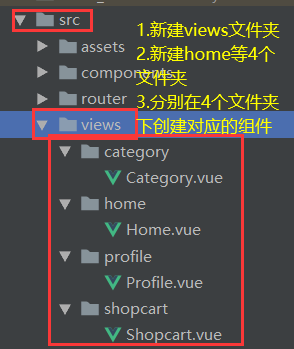
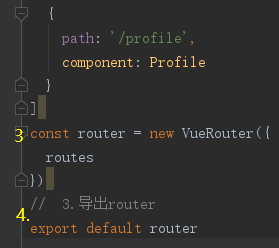
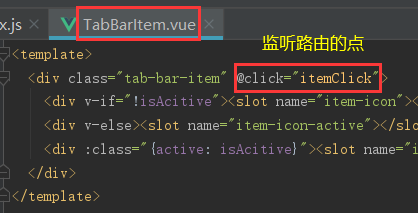


## tabbar-TabBarItem和路由结合

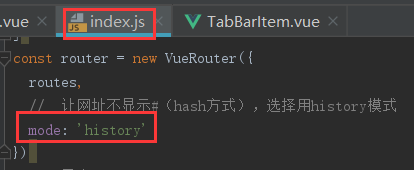


1. 前面创建项目时忘记勾选安装路由，所以先安装路由。npm install vue-router --save



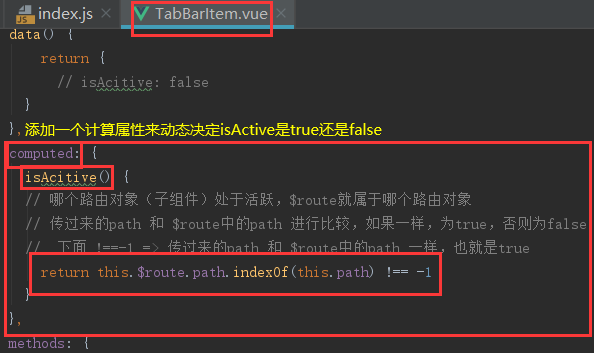
1. 新建router文件夹、index.js文件components文件夹放公共的组件
2.  
3. 
4.  
5.  

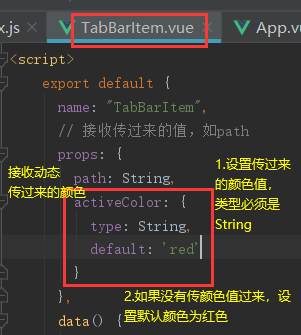
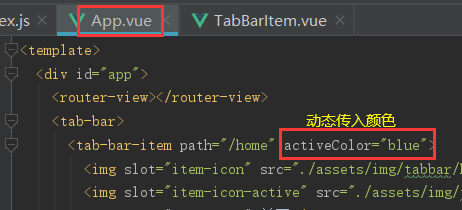
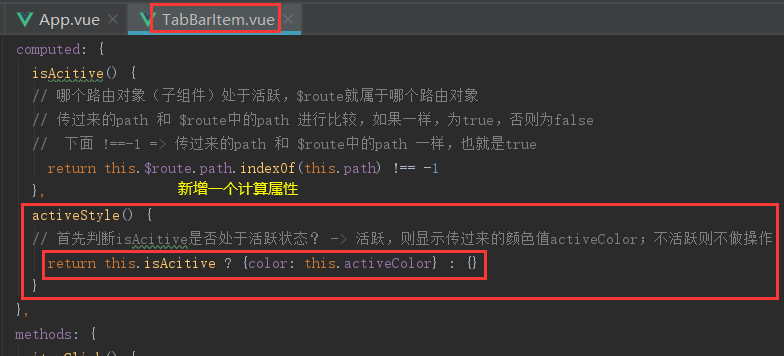
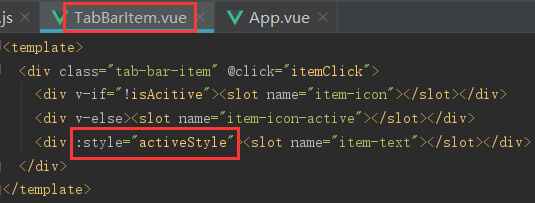


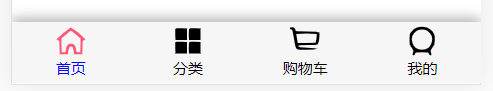
## tabbar-TabBarItem的颜色动态

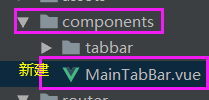
底部Tab颜色不会变化，所以要【动态决定isActive】，而不是写死



页面结果👉

1. 动态决定active样式中，活跃时显示的颜色
2. 
3.  
4. 

页面结果：

1. 
2. 移过去后注意修改路径

