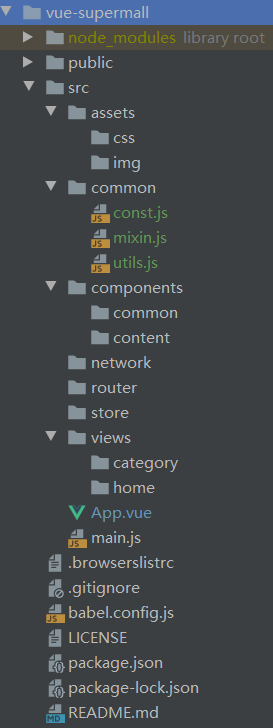
1、github：

新建仓库后，（选择1/2）

（1）在本地把github上的仓库git clone，然后把本地已经由vue create内的文件复制到clone下来的文件夹内（除了.git），然后git add . ，git commit -m ‘xxx’

（2）在本地的项目文件夹下，执行新建仓库的时候给的git remote add origin https…，然后git push -u origin master

2、新建各类文件夹，搭好目录结构：



assets：静态资源文件

common：公共js

components/common：公共组件

components/content：业务组件

network：axios网络请求

router：前端路由

store：状态管理

views：各页面视图

3、引用两个css文件：

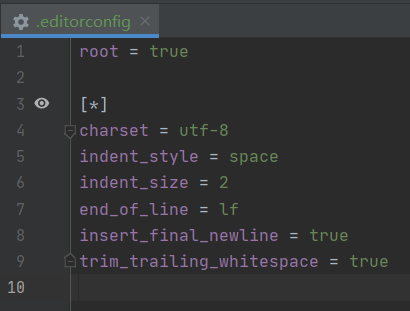
在assets/css中加入初始化css：<https://github.com/necolas/normalize.css>，还有base.css

4、配置别名：

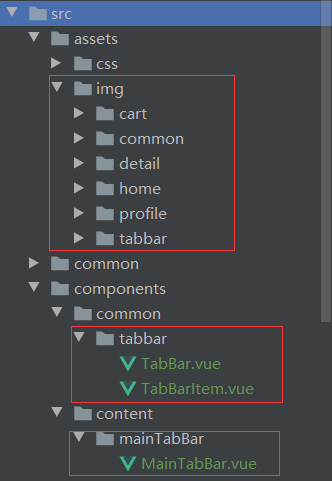
新建vue.config.js：



5、拷贝编辑规范： 这个文件（公司）项目内有自带的，不需要自己创建



6、引入各独立组件及其静态资源：



7、配置路由：

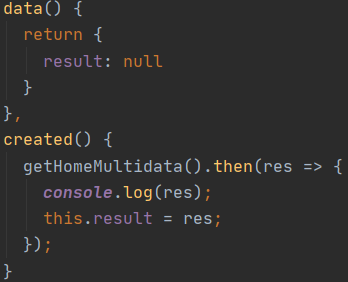
npm install vue-router，router文件夹内创建index.js

8、请求首页（或其他页面数据）数据：

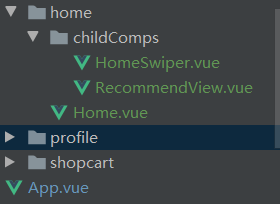
安装axios，配置request.js。

每个view再封装一层请求数据的模块以解耦，Home.vue对应Home.js，达到组件只负责展示的效果，最后在组件里import数据请求模块，在created()钩子里请求到数据后，用data里预设的变量接受。

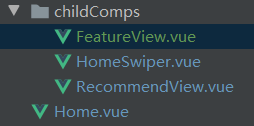
ps.垃圾回收：指内存块中彻底没有变量指向它，就会被回收，这时候数据才真正消失了。



9、设计首页：（包括轮播图）



10、FeatureView：



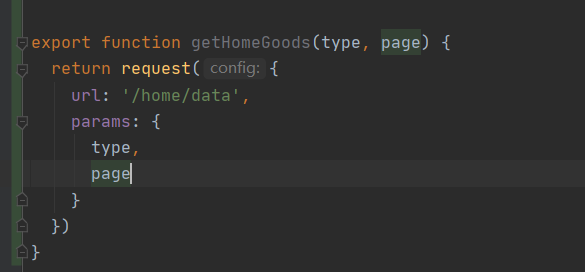
11、TabControl：



12、处理首页不同tab加载不同的内容：

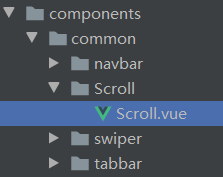


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

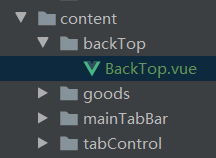


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13、封装better-scroll组件，让它在每个页面上都能调用：



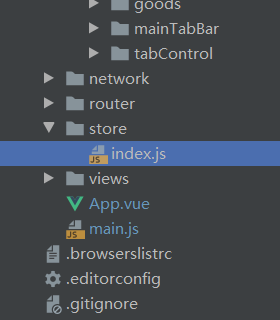
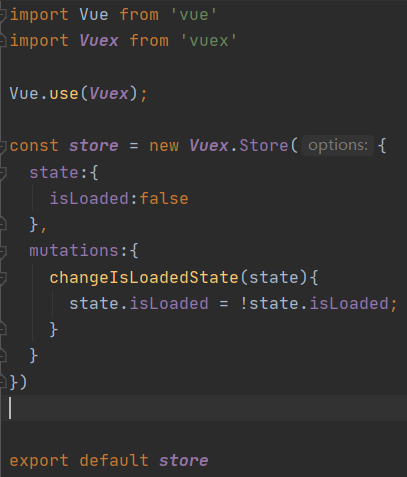
14、制作返回顶部组件：



15、使用vuex管理状态：

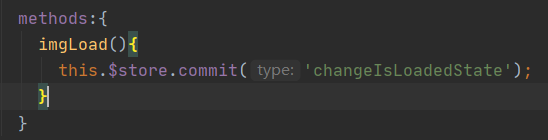
因为在使用betterscroll的时候，它是不知道图片是否加载完的，所以要在图片加载完成后立即刷新一次betterscroll要管理的内容，步骤如下：

1）在/store内创建index.js



2）当图片加载完之后，commit一次：





3）在使用了betterscroll的组件内创建监听器：（计算属性只是为了美观

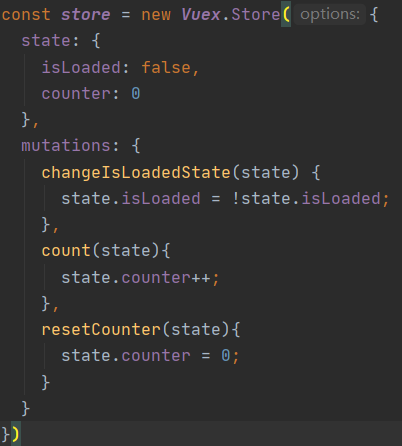


注意：watch属性内的函数名实际上就是要监听的会发生变化的值，一旦发生变化就会执行。刷新之后要把状态修改回false，务必使用commit。因为这里会让要监听的值发生改变，会造成监听函数死循环调用，所以刷新和修改属性操作包裹在if里，条件为true（即需要刷新时）才刷新。

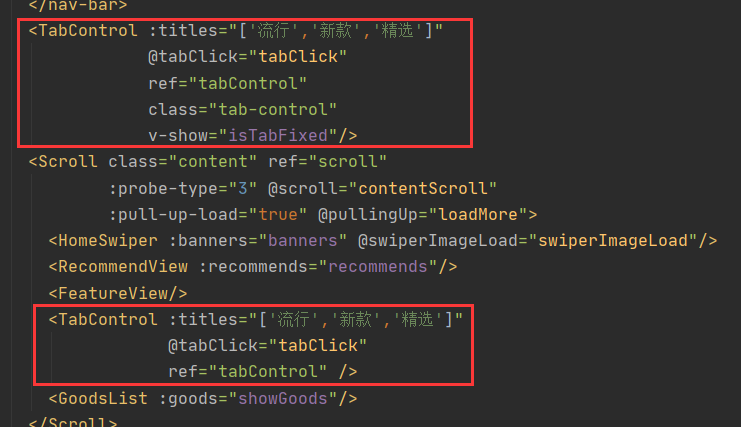
ps. 防抖函数：为了改善性能（因为每加载一次突破都要refresh一次），使用防抖函数，当数据加载到一定量时再refresh。还适用于搜索框的联想词条列表。

debounce(func, delay) {  
 let timer = null;  
 return function (...args) {  
 if (timer) clearTimeout(timer);  
 timer = setTimeout(() => {  
 func.apply(this, args);  
 }, delay);  
 }  
}

但这里使用计数器的方式实现：

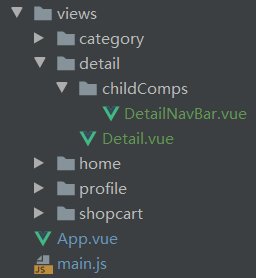
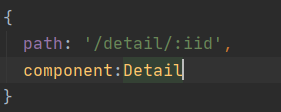
16、吸顶效果：



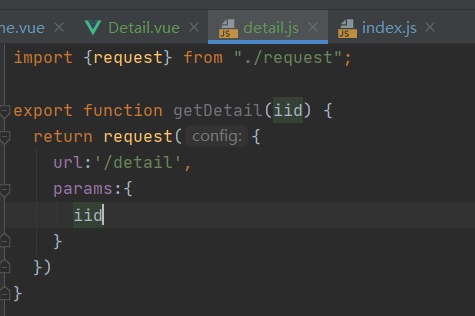
思路：当滑动到一定的距离的时候，控制在导航条下面的tabcontrol组件显示或者不显示，用vuex管理两个组件的状态。

17、保存路由切换进度：<keep-alive exclude="Detail" ></keep-alive>

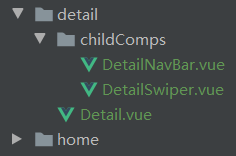
18、设置detail路由，新建detail组件，加入导航条组件



19、请求detail页面数据：在network下新建detail.js



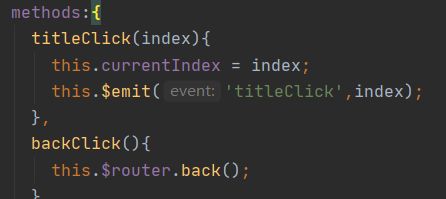
20、detail页面的轮播图：



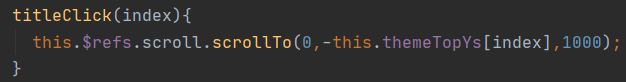
21、不同组件内有同样的代码（钩子函数等）时可以用mixin对它们进行复用。

22、详情页的导航栏与页面联动：

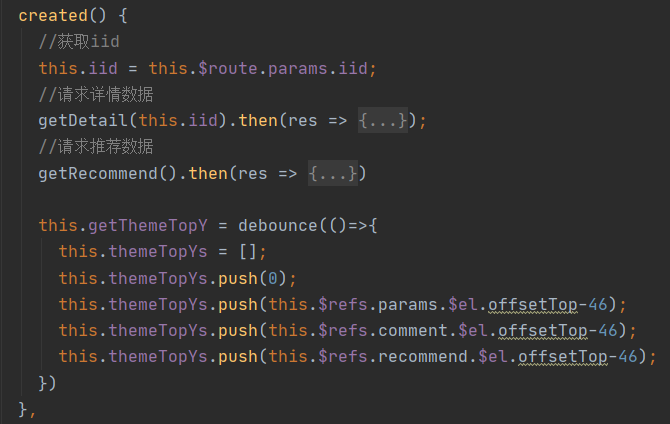
在navbar组件内监听点击事件，发送相应的index：



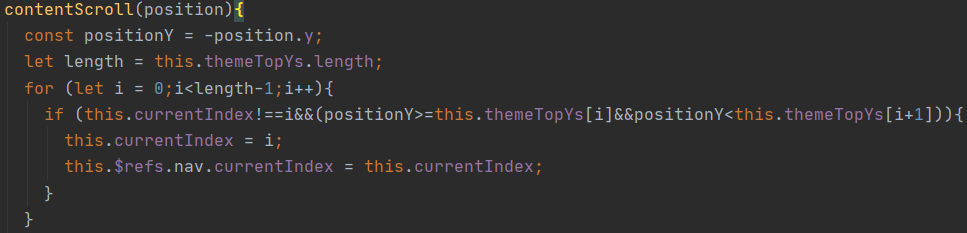
在detail组件内监听事件：

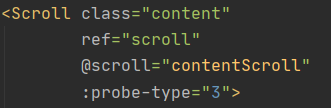


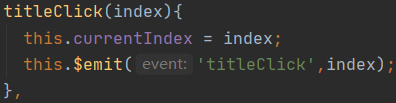
要滚动的距离用一个数组存放起来，数组的值就利用详情页内图片加载完后，获取每个组件内元素的offsetTop（然后微调-46什么的），用防抖函数包裹的原因是，这样就不用每张图片加载完就立马获取一次offsetTop，而是等全部图片加载完后获取一次即可



滚动联动：







23、在vuex中，mutation内一个函数尽量只做一件事情，便于vue插件进行跟踪，复杂的请求数据、逻辑判断放在action中：

