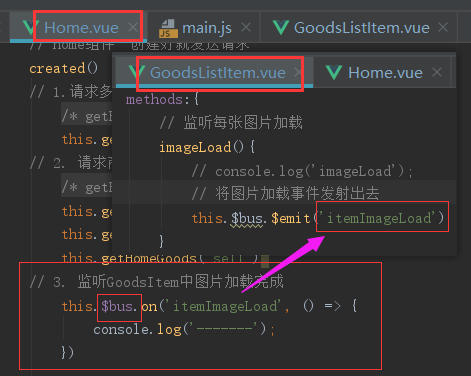
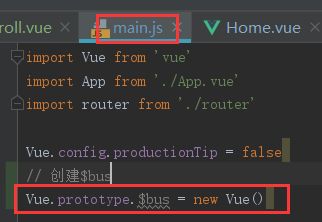
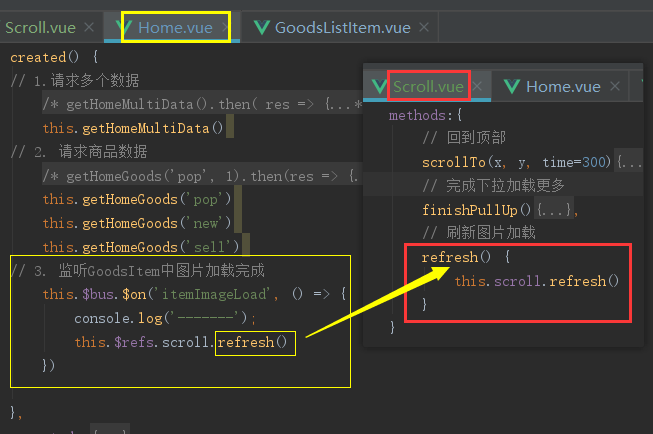
# 项目开发2

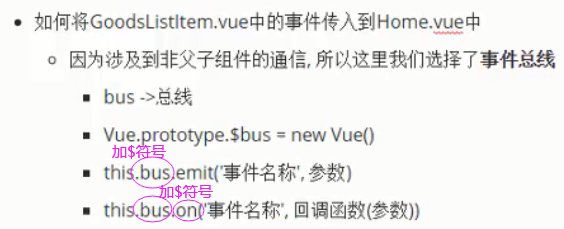
## 首页开发-滚动区域的Bug分



## Refresh函数找不到的bug处理



防止报错refresh（）找不到或者未定义



防止报错找不到scroll，因为在created（）中有可能找不到this.$refs.scroll

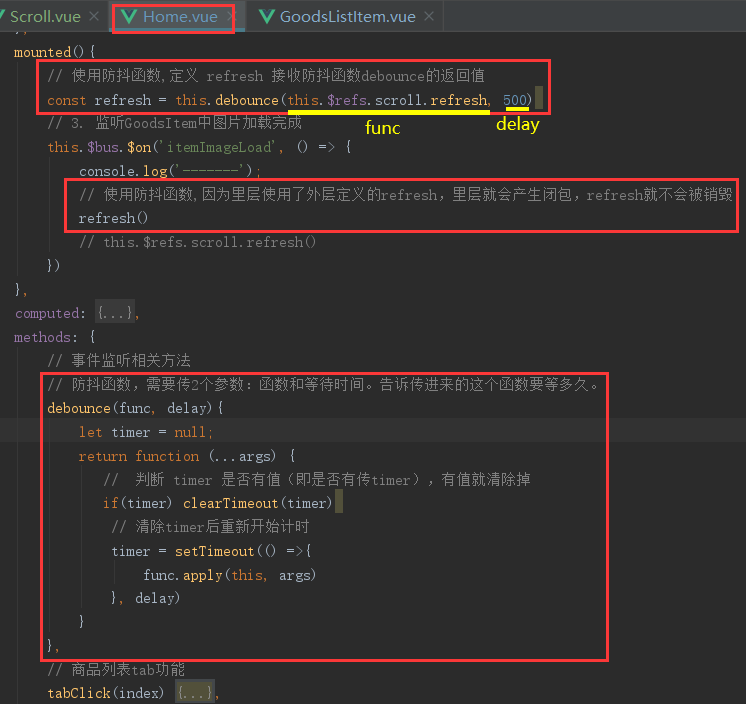


## 刷新频繁的防抖函数处理

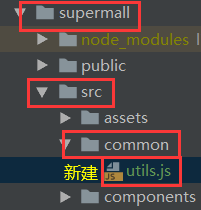


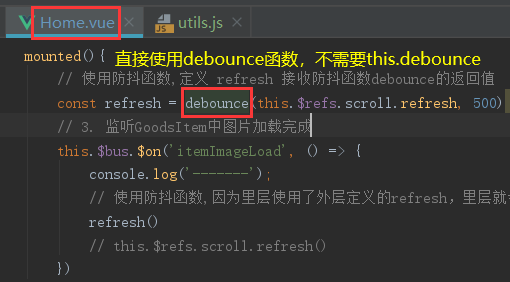
refresh()函数在每张图片加载好就会刷新一次，有多少张图片就刷新几次，这样太过于频繁，导致影响服务器性能。

类似的有搜索框：用户输入一个字就等100ms或500ms，看用户是否还继续输入，若在100ms或500ms前用户继续输入，就等用户输完所有字，再进行搜索；如果用户每输入一个字就发送一次请求，这样对服务器不好。

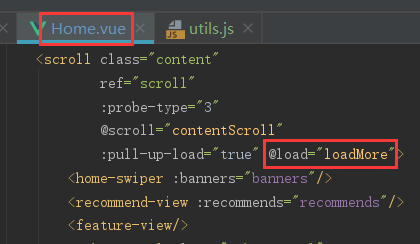
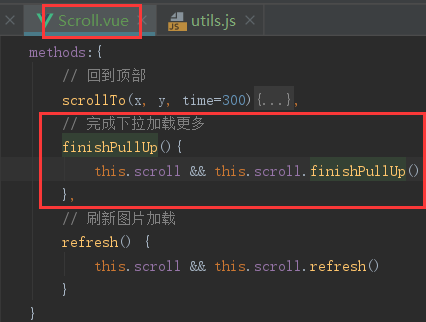
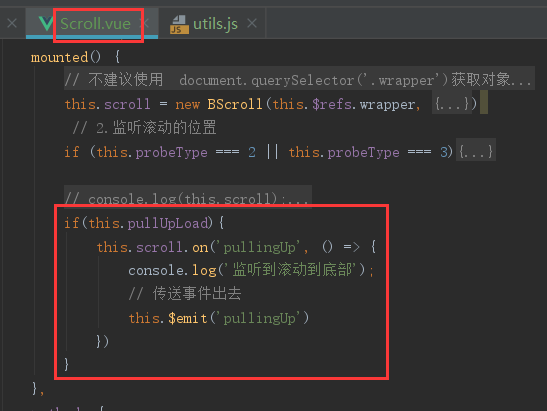
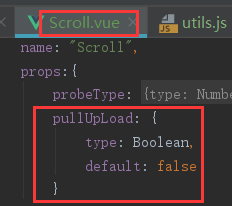
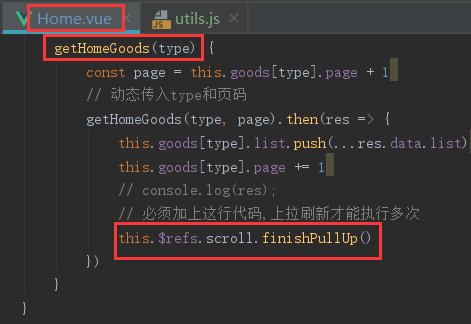


## 上拉加载更多的完成

新建工具类 

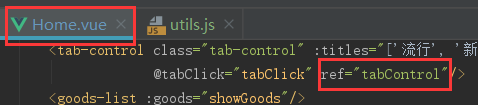
 

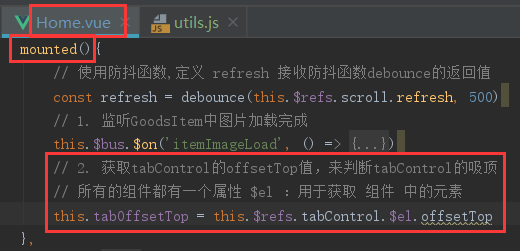
* 实现上拉加载更多

## tabControl的offsetTop获取



1. 首先要获取到是哪个元素
2. 定义一个变量来保存获取到的元素的offsetTop值
3. created()函数中的【this.$refs.tabControl】返回undefined，不能获取到元素，所以要在mounted()中获取



但是获取到的高度是图片加载完之前的高度，不是tabControl所在的高度。所以也不能在mounted()中获取offsetTop。

## tabControl的吸顶效果完成

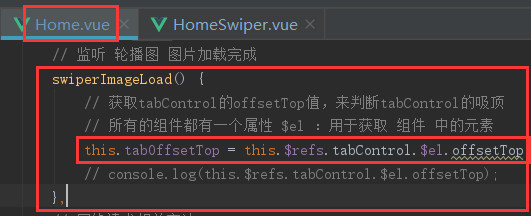


1. 先监听图片是否加载完，主要是轮播图的加载比较慢

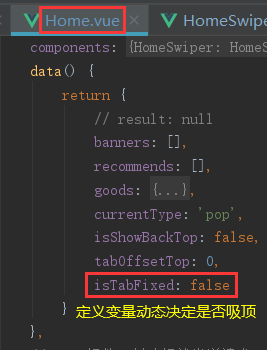
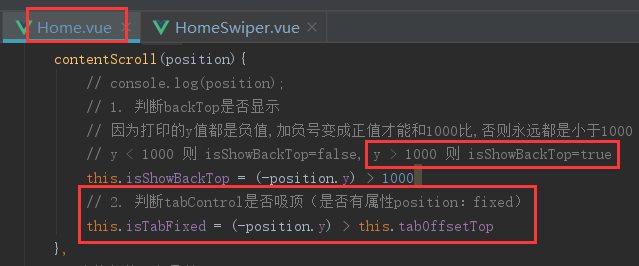
 

1. 只需要知道轮播图加载一张的高度即可，所以修改代码

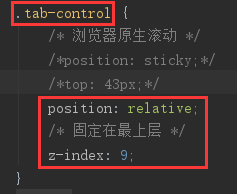
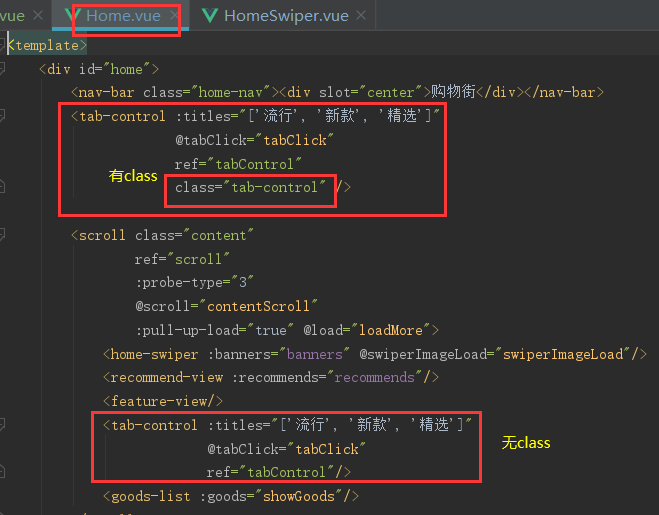
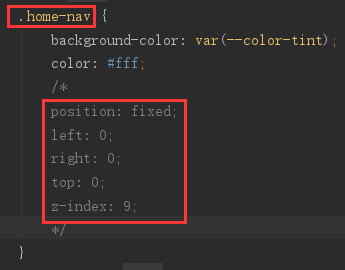
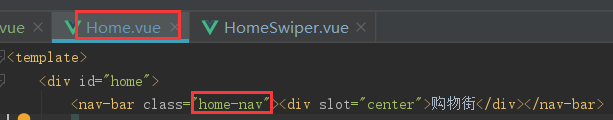


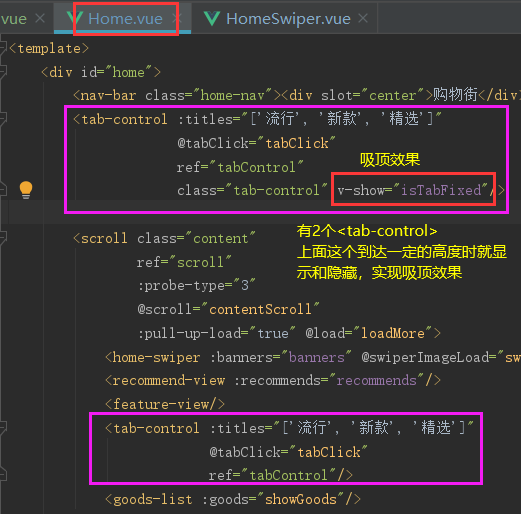
1. 动态决定是否吸顶

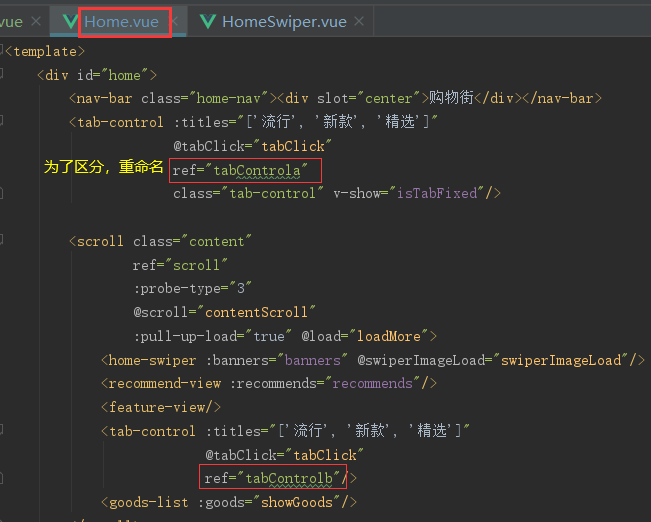
结果：tabControl不见了，不能加fixed，删除动态添加class：fixed。

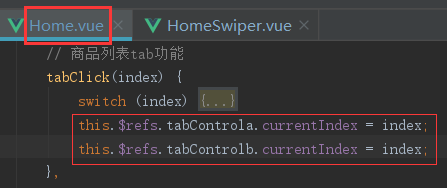
1. 复制【tabControl标签】，放到【顶部nav标签】后面，但是【顶部nav标签】设置【position：fixed】就会脱标不占位置，这时【tabControl标签】就会跑到顶部，但是被【顶部nav标签】盖住了，原来是因为除【顶部nav标签】以下的内容要滚动，所以原生的做法要把【顶部nav标签】设置【position：fixed】才不会随着内容一起滚动。现在使用【better-scroll】实现内容滚动，【顶部nav标签】对它并没有影响，所以把【顶部nav标签】的class中的fixed删除。



视觉误导，实现吸顶效果👉

1. 但是存在一个问题，列表内容和tabControl选项（pop/new/sell）不一致，比如内容是新款new的，但是tabControl选项却是pop流行。解决如下：



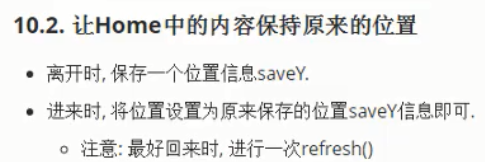
 

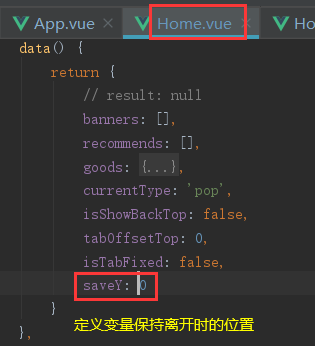
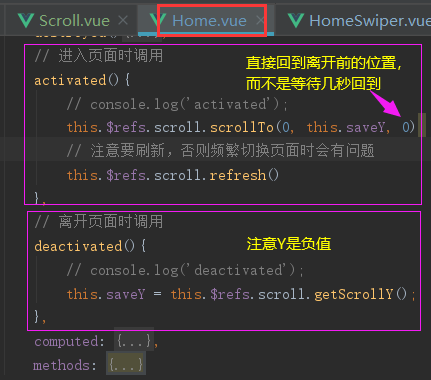
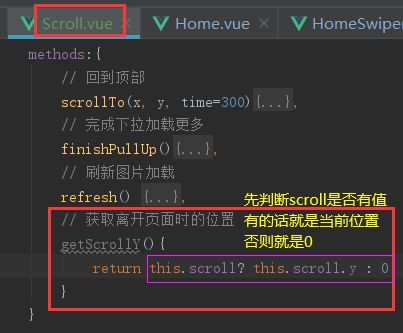
## Home离开时记录状态和位置

1. 让Home不要随意销毁掉，使用keep-alive



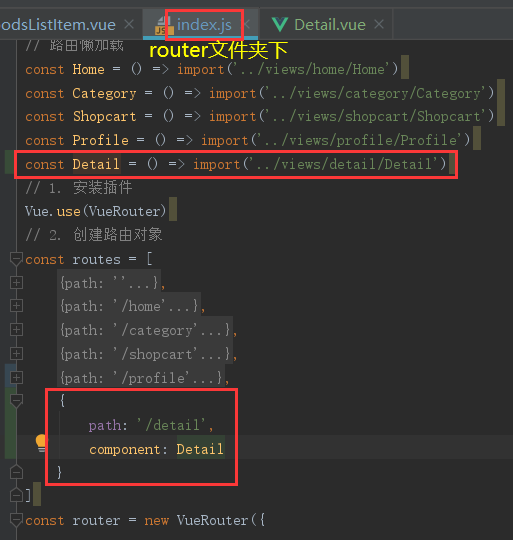
1. 底部的tabbar切换时，即离开Home页面，让Home中内容保持原来的位置不变



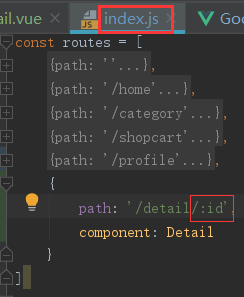
 

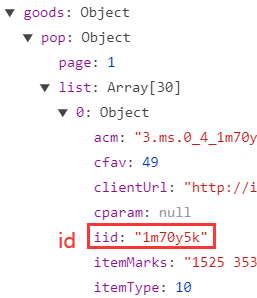
## 跳转到详情页并携带id

1. 新建文件夹来放详情页组件 
2. 配置路由，进行详情页跳转

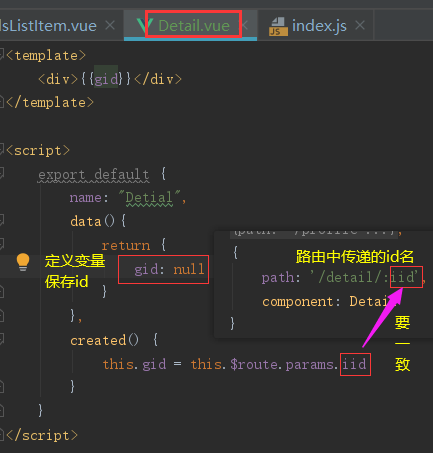


1. 页面跳转
2. 传递参数id

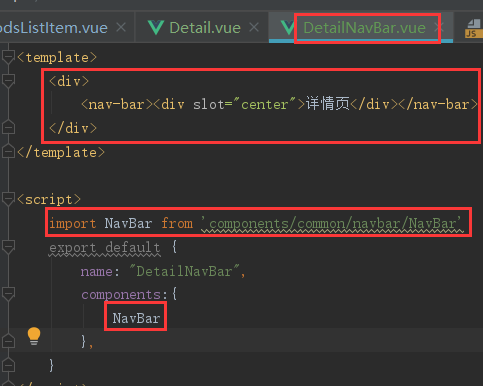
 

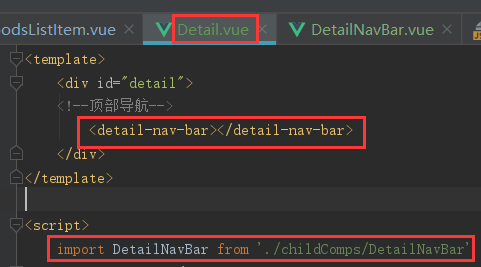
 

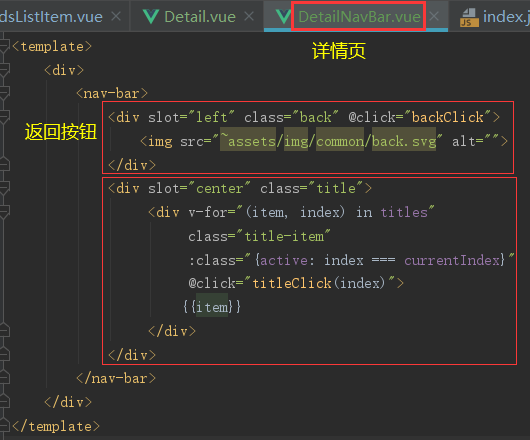
页面结果：

1. 获取iid 

## 详情页-导航栏封装

1. 封装详情页的navbar  

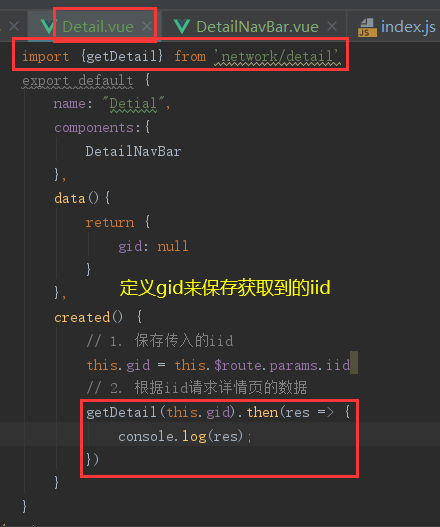


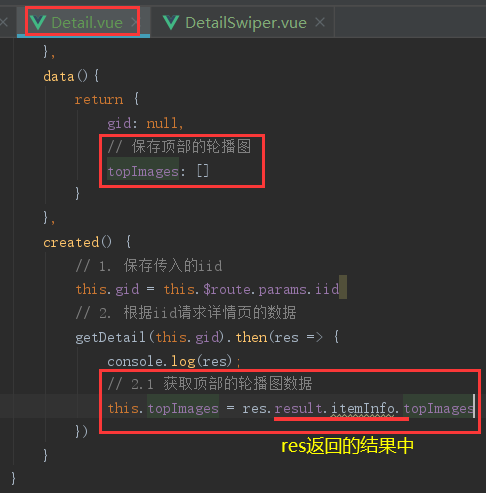
1. 

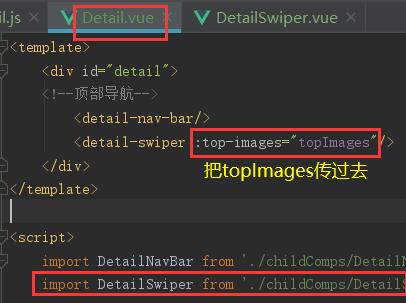
 

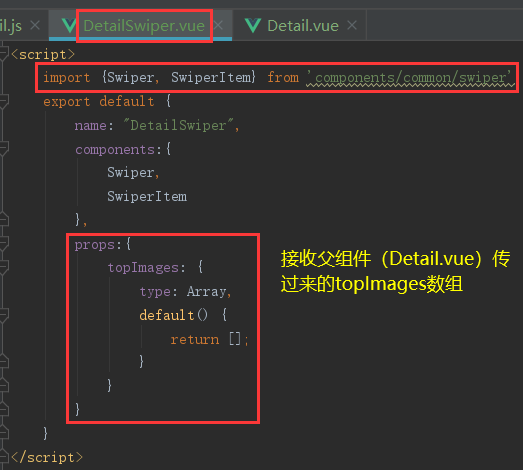
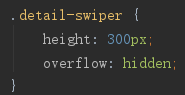
页面结果：

## 详情页-数据请求已经轮播图

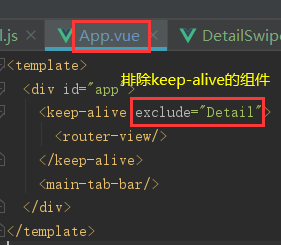
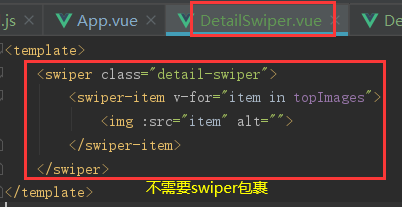
1. 封装详情页的相关请求 
2. 根据id请求详情页数据
3. 展示数据

获取轮播图数据

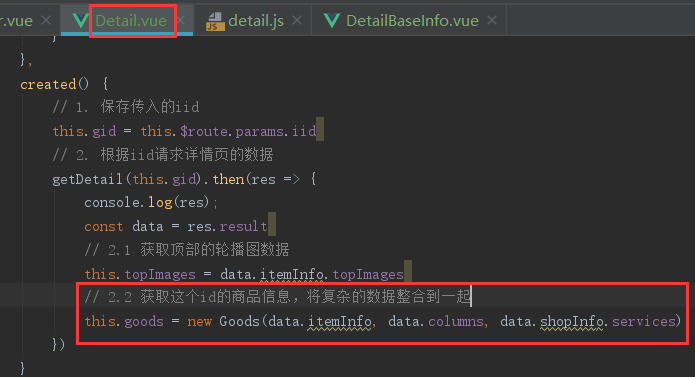
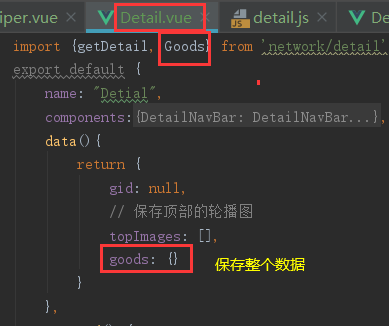
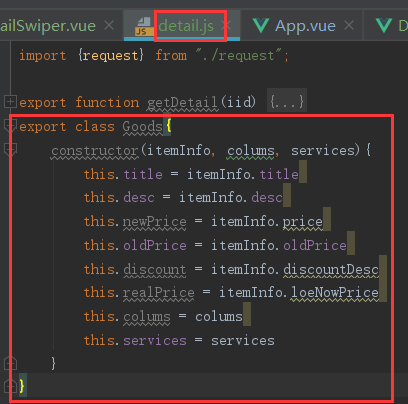
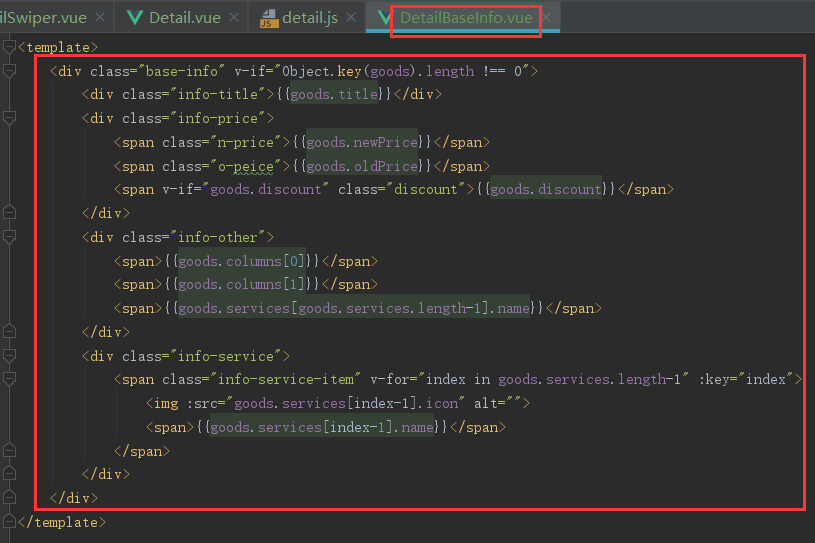
展示轮播图 

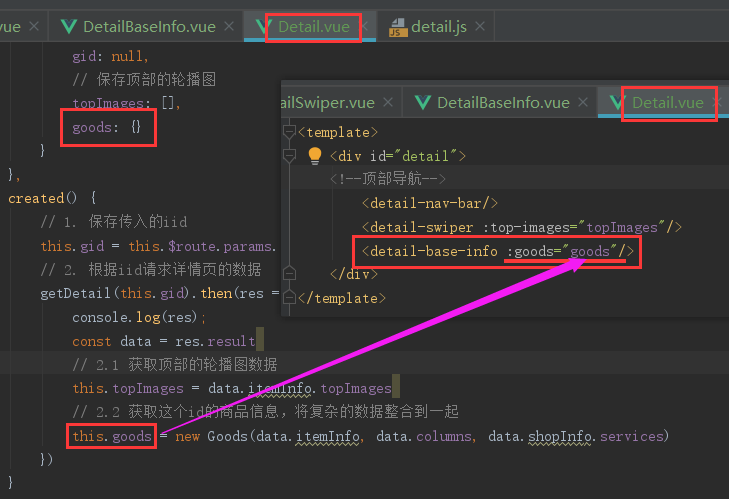
详情页轮播图组件  

但是存在一个问题，因为App.vue中使用了keep-alive，导致点击首页每个列表项，传过来的id都是同一个，页面的轮播图请求到的都是一样的。解决如下：

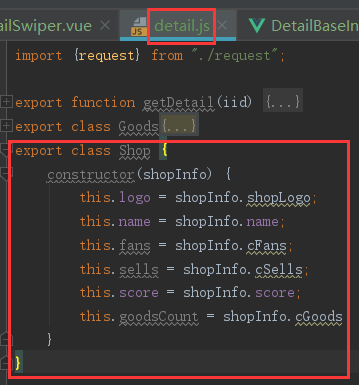
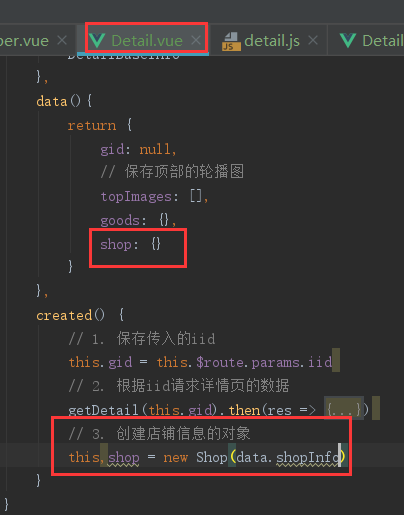
更正

## 详情页-商品信息的展示

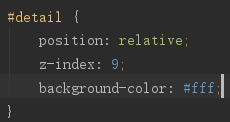
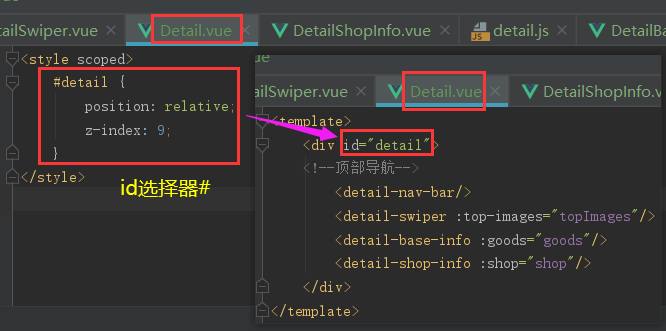
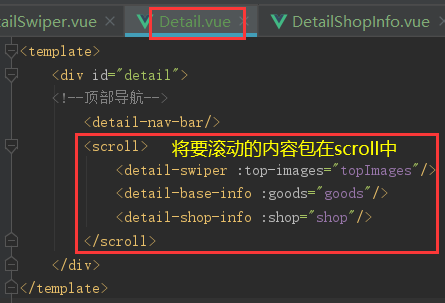
1. 对数据的整合
2.  

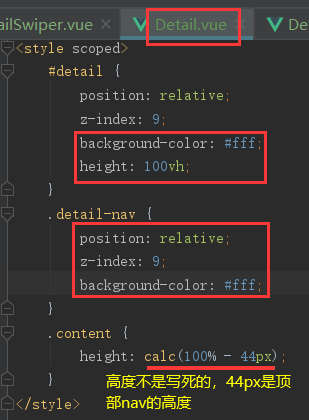
展示数据

## 详情页-店铺信息的解析和展示

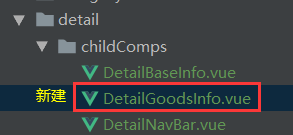
 

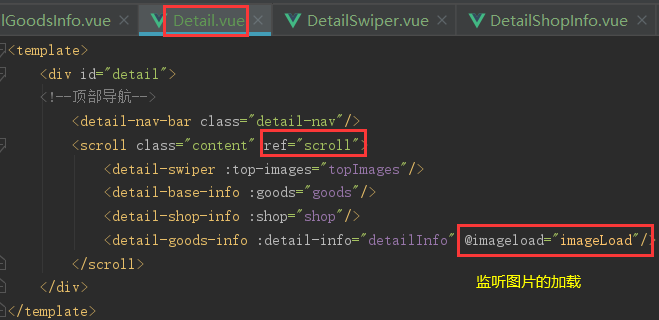
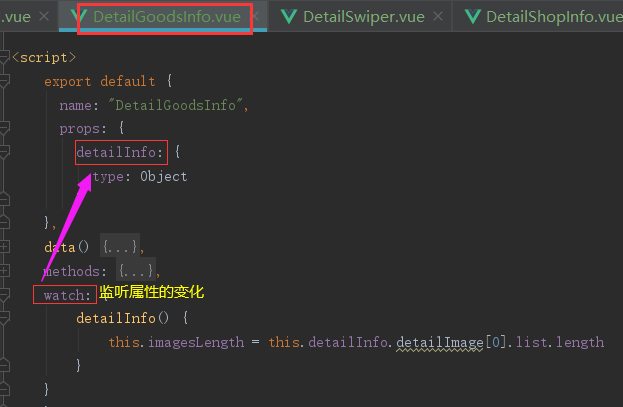
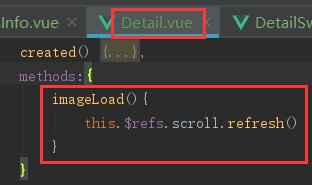
## 详情页-加入滚动的效果scroll

1. 隐藏tabbar 
2. 使用better-scroll而且scroll必须要有固定的高度

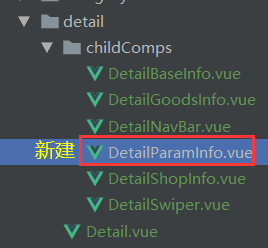
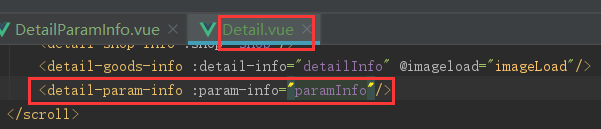
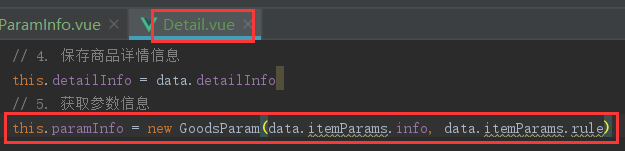
 

## 详情页-商品详情数据展示

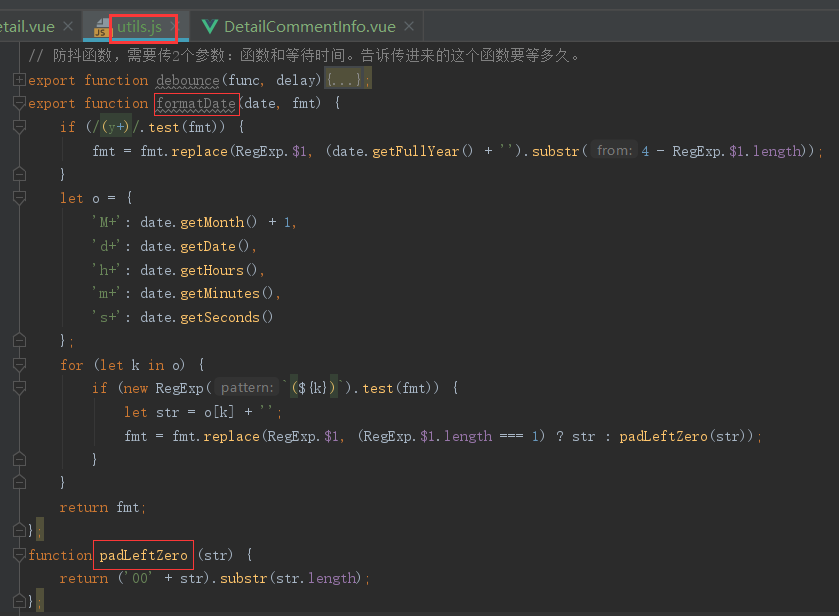
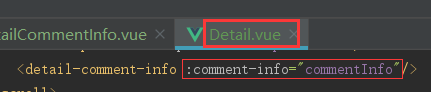
 

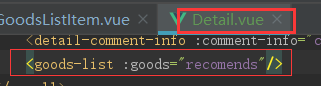
## 详情页-商品参数信息的展示

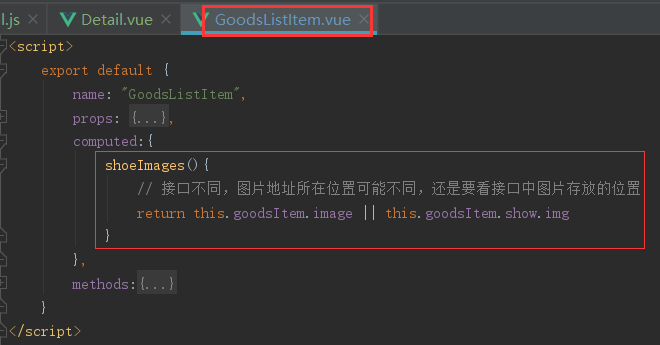
1.  
2. 

## 详情页-商品评论信息的展示

## 详情页-商品推荐数据的展示

1. 
2. 导入goodsList组件 

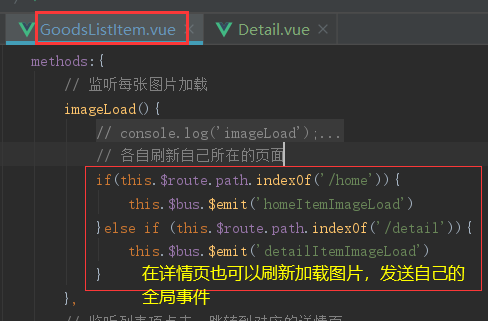
  

## 详情页-首页和详情页监听全局事件和mixin（混入）的使用

### 18.1 首页和详情页监听全局事件

详情页的推荐商品列表加载完（使用goodsList组件，图片加载完后会告诉首页进行刷新），但是在详情页中，并不需要首页进行刷新。

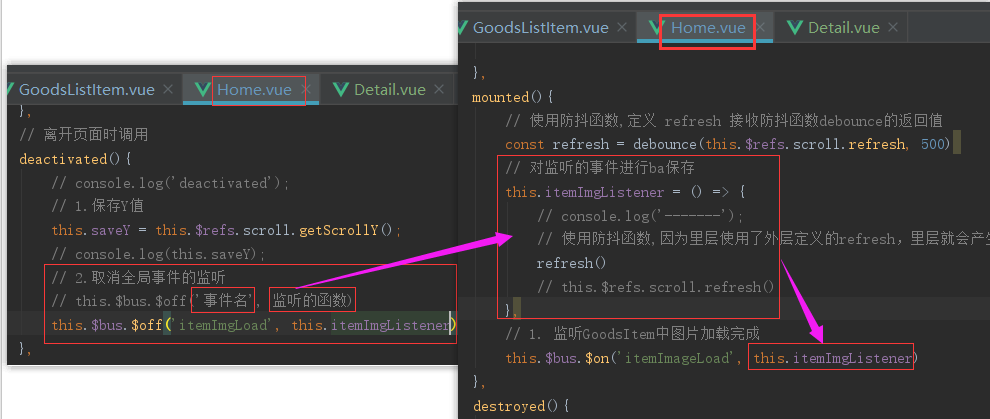
1. 方法一：通过路由来判断路径。首页和详情页一起写，不用分开到2个组件中去写。

1. 方法二：哪个组件需要这样做就按下面的格式写



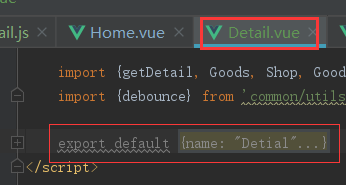
首页



详情页

### mixin（混入）的使用

抽取公共的、一样的代码。类可以使用继承，但是对象需要使用混入mixin（mounted和destroyed函数都是写在export default里面，而export default合起来就是一个对象）。

对监听全局事件使用混入