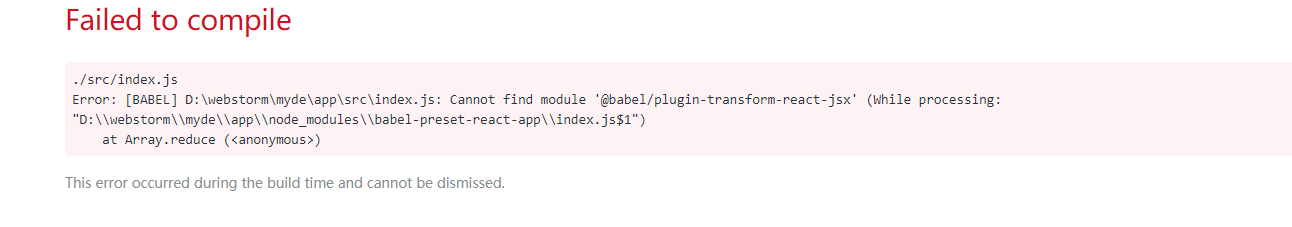
React实战

# 轮播图

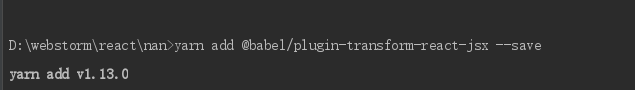
## 轮播图实现原理

Container

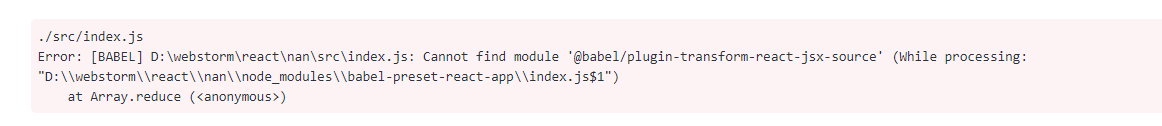


第一个坑,缺少什么?

解决办法

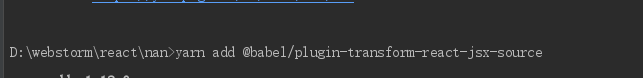


第二个错误

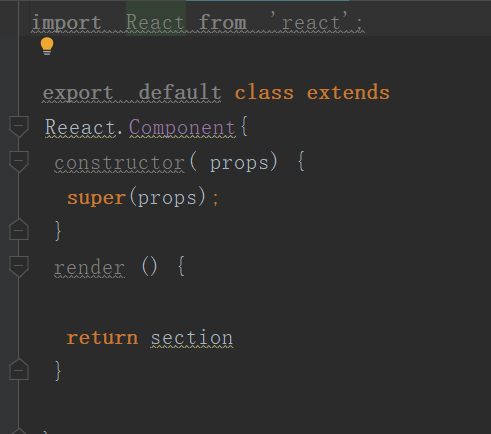


解决办法

2:

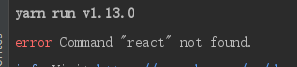


缺谁就安装谁



明明已经安装 react . 报错

这是没有设置 jsx语法.将webstorm语言框架设置一下.



解决:重新安装:

npm install -g create-react-app

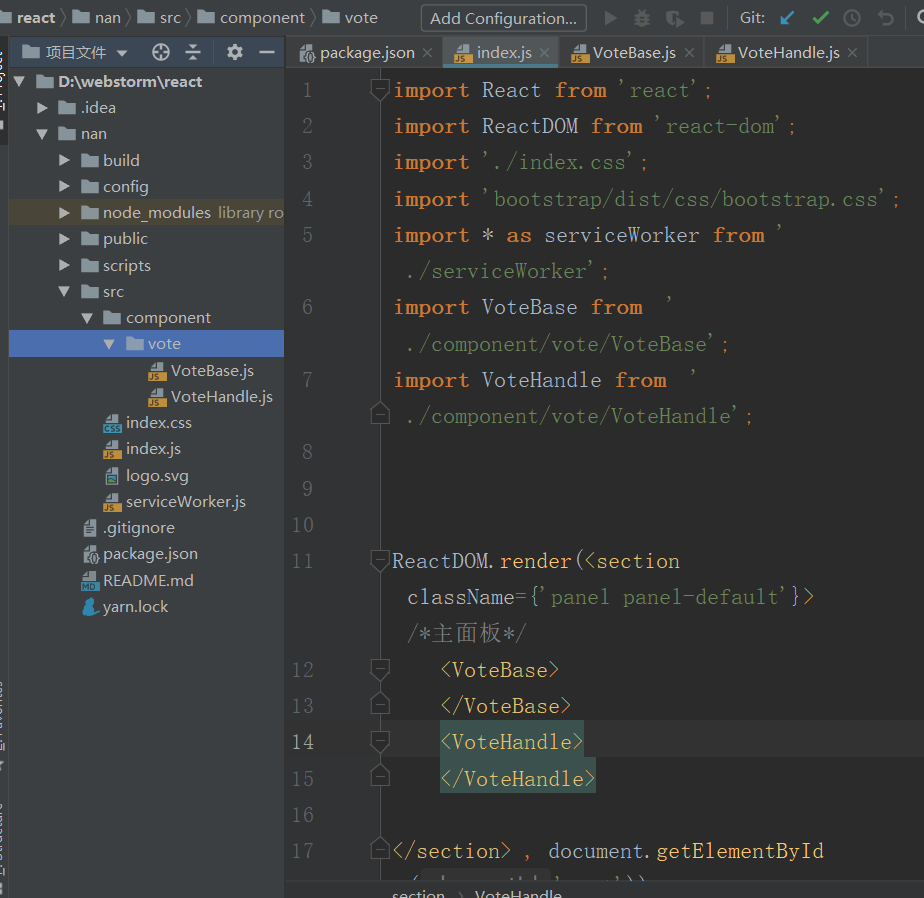
create-react-app hello-world

直接引bootstrap

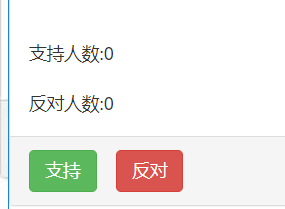


创建组件

在主index.js下面调取组件 即是src目录下的index.js



排除一上坑,终于解决



要看在哪个地方操作,就在那个地方对应的组件写方法 这里开始引入redux.

3.



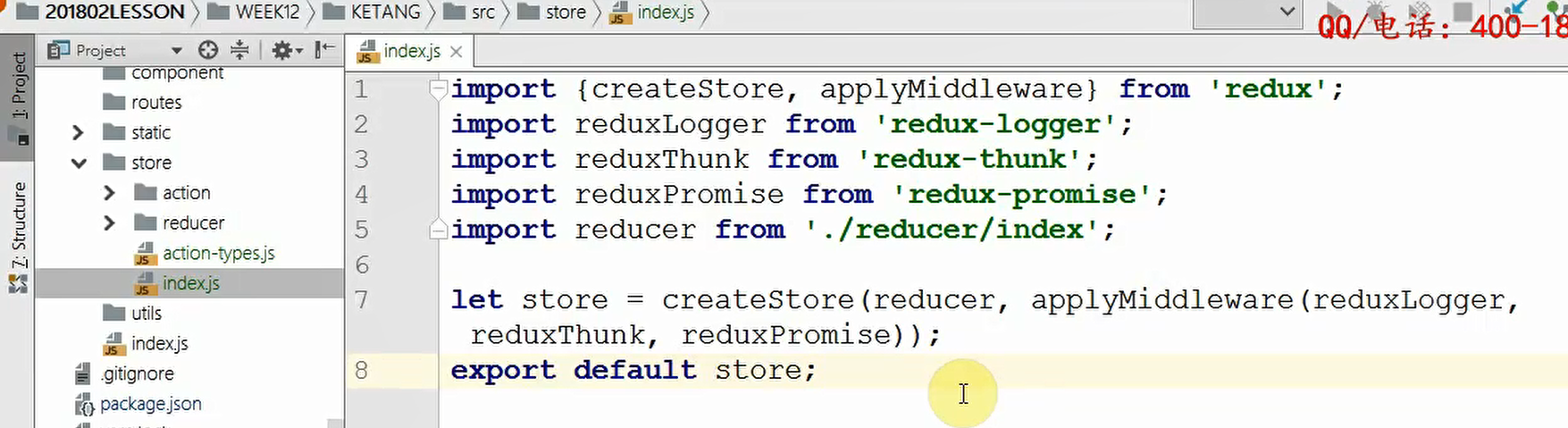
style属性 中 .20px 和autu 写在一个引号之间

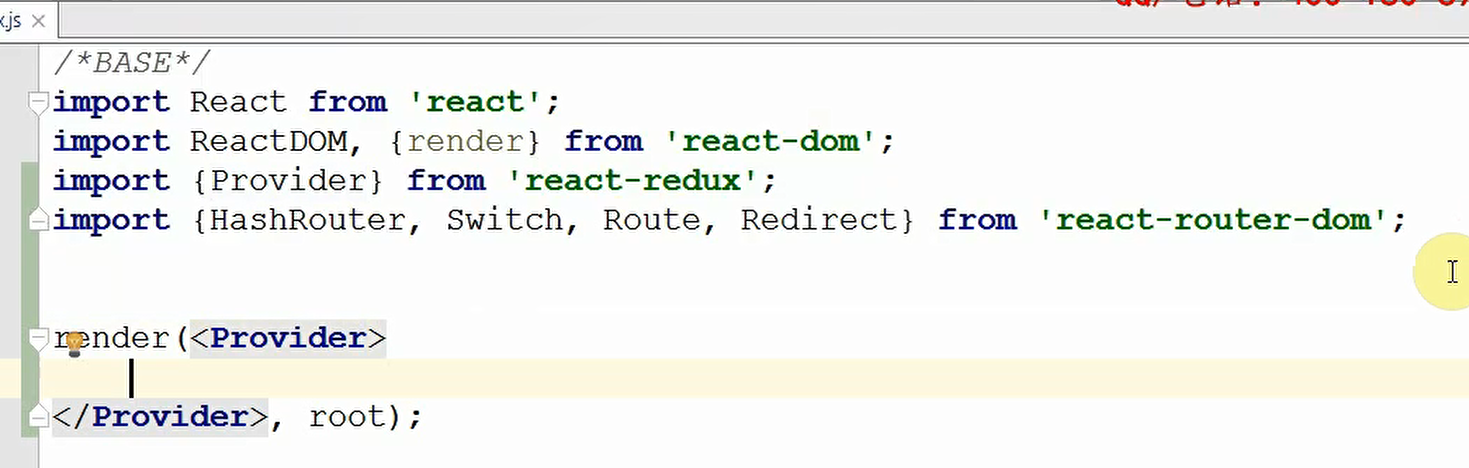
# 设计

1. reducer

| - course.js

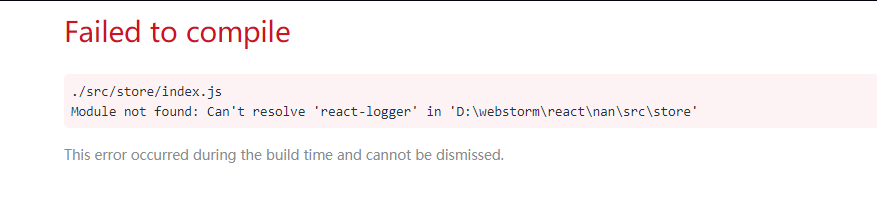
中间件 : 写在store的reducer/index.js中 有时间就去补一下.





1安装中间件 :

npm install redux-logger --save-dev

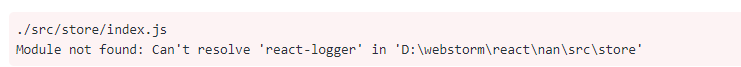


解决方法1 ;

仍旧报错



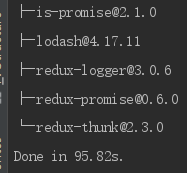


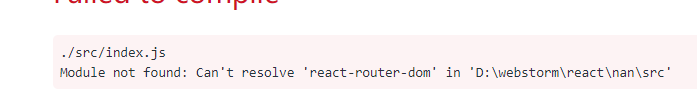


又报错

安装中间件.

yarn add redux-logger redux-thunk redux-promise





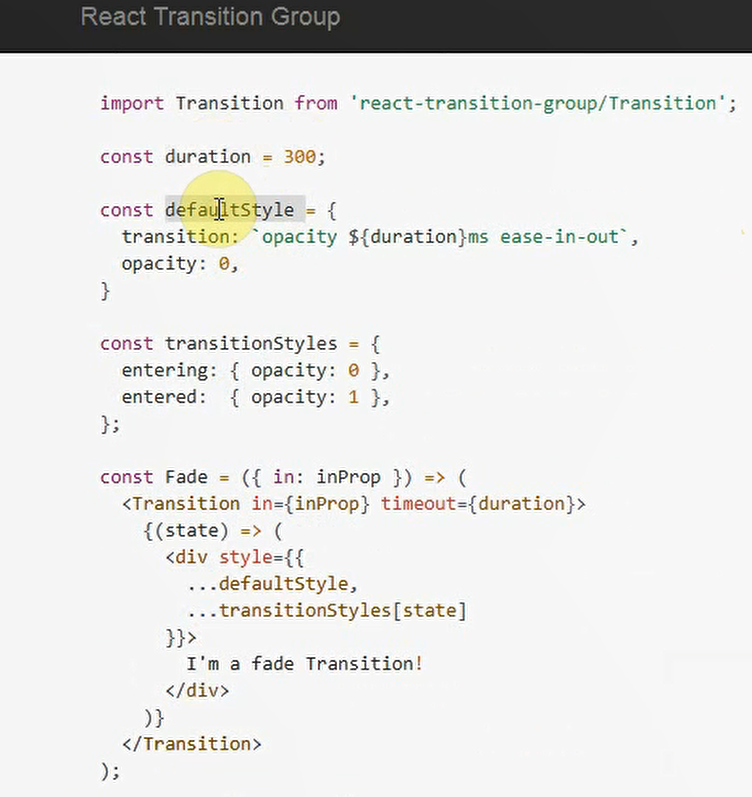


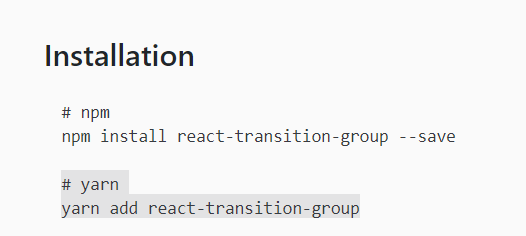
>yarn add less-loader

生成的less文件无法编译成css文件



隐藏插件



安装

# 安装React Transition Group

个人中心:

注册 效果 登录效果

地址:: /personal/login

方式:POST

参数: 用户名:

密码:

返回: { code: 0, 0:登录成功,1: 登录失败

msg: “xxx”

}

方式: get

参数

参数: 用户名:

密码:

返回: { code: 0, 0:已登录,1:未登录

msg: “xxx”

}

在路由里面 routes: {

person 中 新建信息 :Info .js ,Login.js Reginster.js ,Tip.js

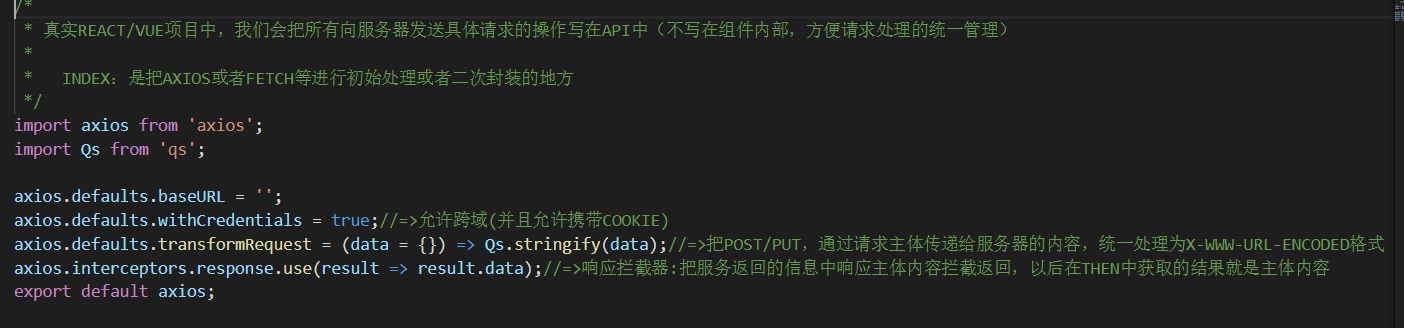
<switich>Route path :</switch> 每种情况都route ,,如果请求哪个 地址,一共四个 ,info和tip只能存其一.登录和失败 .path 地址都是info .



在真实react/vue项目中,所有的发送请求都放在api下, 我们会把所有向服务器发送具体请求的操作写在api中.(不写在组件内部) ,方便请求统一管理

index : 是把axios 或者 fetch 进行初始处理或者二次封装的地方

安装: npm isntall axios --save



import axios from 'axios';

import Qs from 'qs';

axios.defaults.baseURL = '';

axios.defaults.withCredentials = true;//=>允许跨域(并且允许携带COOKIE)

axios.defaults.transformRequest = (data = {}) => Qs.stringify(data);//=>把POST/PUT，通过请求主体传递给服务器的内容，统一处理为X-WWW-URL-ENCODED格式

axios.interceptors.response.use(result => result.data);//=>响应拦截器:把服务返回的信息中响应主体内容拦截返回，以后在THEN中获取的结果就是主体内容

export default axios;

qs : 插件的安装:

安装

npm install qs

使用

let data = qs.stringify({

    "username":this.username,

    "password":this.password

});

通过请求主体传递给服务器的内容.统一处理

转换成x-www-url-encoded 格式 promise then 就是返回成功 .主体内容

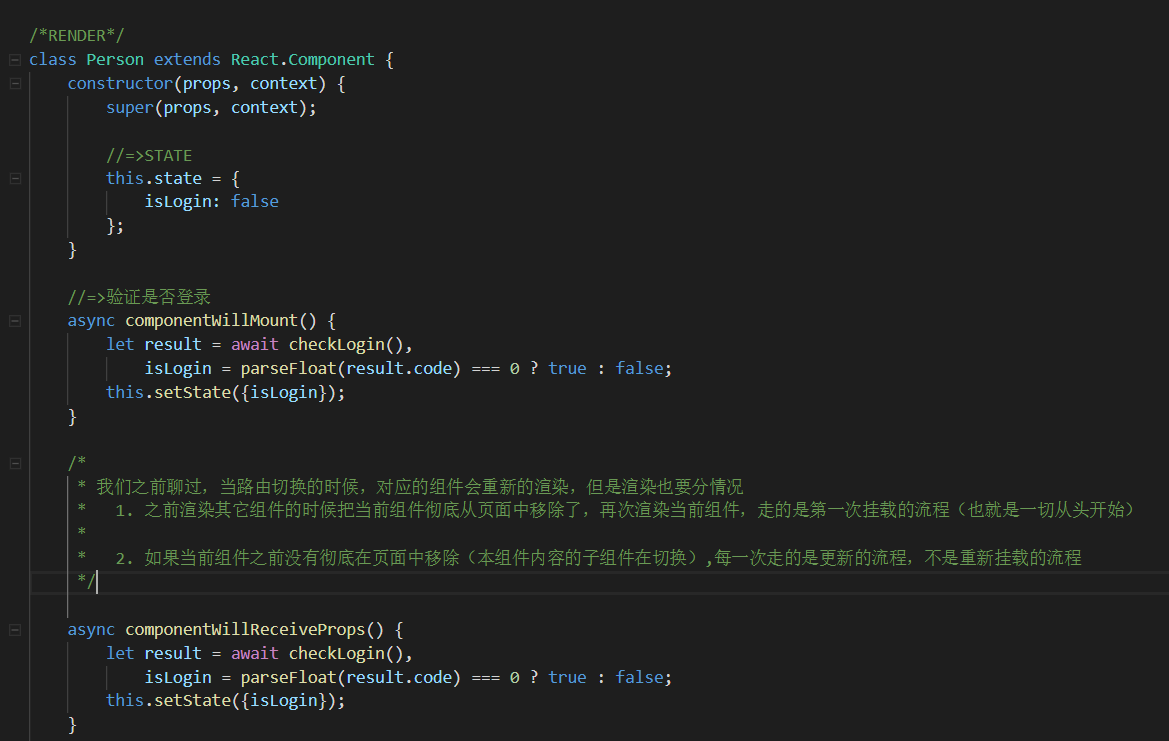
常用4个: 以上四个.

在api 中 写person请求 . 执行这个方法就知道登录或者失败

到路由中去

{/\*  
//=>路由的验证和渲染是同步的，不允许在校验中出现异步，因为这样在异步没有完成之前，根本不知道渲染谁，语法不支持这样的操作  
<Route path='/person/info' render={async () => {  
 //=>是否登录的权限校验  
 let result = await checkLogin();  
 if (parseFloat(result.code) === 0) {  
 return <Info/>;  
 }  
 return <Tip/>;  
}}/>  
\*/}

为什么这么做不行:因为校验的时候不允许出现异步,必须是同步.路由不会等你.瞬间.路由发现你没有返回.他就不知道渲染什么. 路由的验证和渲染是同步的.



/\*  
 \* 我们之前聊过，当路由切换的时候，对应的组件会重新的渲染，但是渲染也要分情况  
 \* 1. 之前渲染其它组件的时候把当前组件彻底从页面中移除了，再次渲染当前组件，走的是第一次挂载的流程（也就是一切从头开始）  
 \*  
 \* 2. 如果当前组件之前没有彻底在页面中移除（本组件内容的子组件在切换）,每一次走的是更新的流程，不是重新挂载的流程  
 \*/

个人中心的 样式

在static 里创建less 文件 .person.less 文件



效果隐藏界面与登录注册页面发生冲突 .

原因:flixed

display: **this**.state.in ? 'block' : 'none'  
}} onClick={**this**.handleClick}>  
 {/\*display 给ul隐藏加特点\*/}

所有受路由控制的都有history

基于render返回的组件不是受;路由管控的组件

routes下的

person,js





获取个人信息:

无需传参

地址 : /person/info

方法:get

返回: { code : 0

msg:”xxx” ,

data : {

id,

name

email

phone}

}

只要服务器发出请求,就要重新渲染.

所有数据请求都写在store

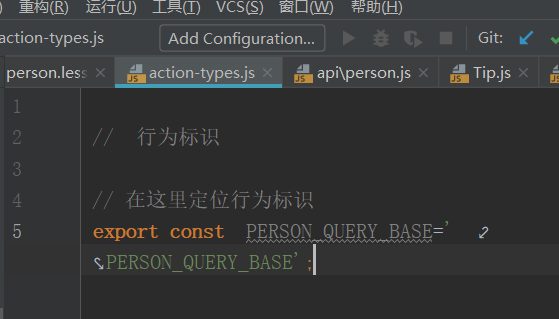
dispatch = > reducer => 改变容器中的状态.( baseinfo :初始值null) => 通知所有用到的状态信息的组件重新渲染 .

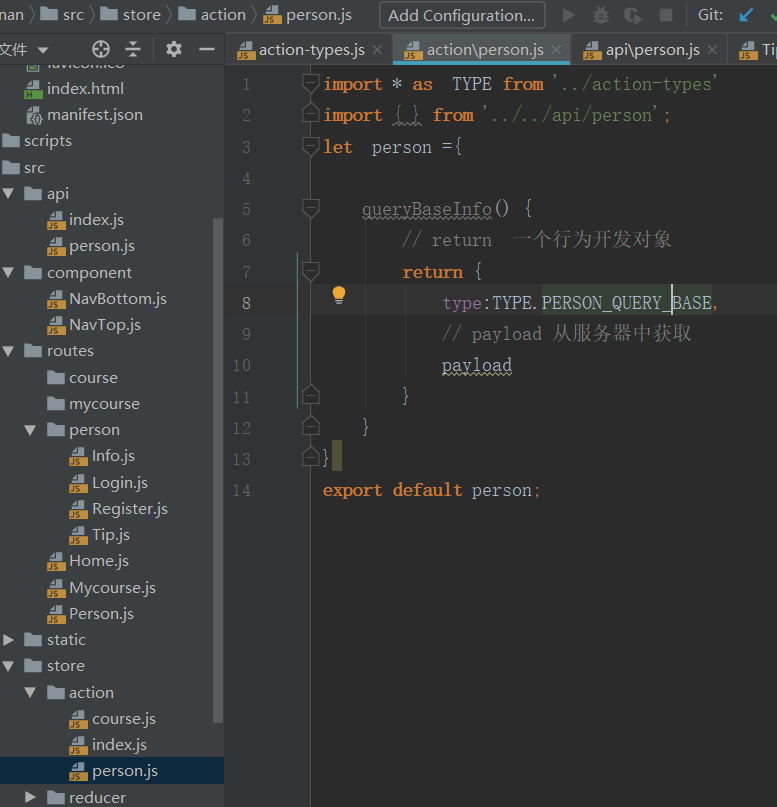
每一次加载组价:验证baseInfo的值.如果没有信息,基于dispatch派发一个任务.(获取服务器端的数据,发送给reducer,并且修改容器中的baseInfo,组件重新渲染) 如果有信息.直接渲染组件.而不需要在重新派发任务.(也就是不在重新发送ajax请求)

理解:通过dispatch开发把服务器中的数据存在redux容器.每次加载组件的时候都去验证rudux有无这个请求.没有就再去发请求

1. 临时存储,从服务器中获取的信息.保证在路由切换渲染组件的时候.防止重复发起请求,减轻服务器的压力
2. 多个平行组件的信息共享 \

在这里定位行为标识. action-types





在action找行为:

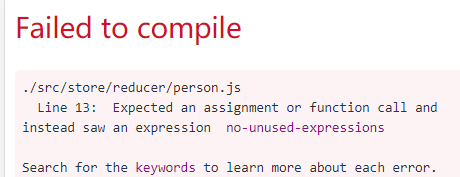
queryBaseInfo() {  
 // return 一个行为开发对象

要把从服务器中获取的信息传给reducer.r拿到修改容器中的状态

payload从服务器中获取 .

引入一个中间件 redux-promise 这里获取 get

先看redux有没有 ,



这个报错 意思是“希望是赋值或函数调用，不能是表达式”