**前端新手开发流程**

1. **初始化环境配置**

|  |
| --- |
| 1. 安装nodejs 和 git  nodejs： https://nodejs.org/zh-cn/download/  git: <https://git-scm.com/>  2. 打开git 建立自己的文件夹目录，并用cd查看  cd desktop  cd work  cd webfront  创建完成之后的文件目录为： desktop/work/webfront   1. 从git上面下载代码（在自己的当前文件目录下）    1. 配置git的环境变量   git config --global user.name "xuyan"  git config --global user.email "xuyan@winsense.ai"  git config -l //查看自己已完成的配置   * 1. 生成自己的公钥并添加到git里面   网址：<https://git-scm.com/book/zh/v2/服务器上的-Git-生成-SSH-公钥>  默认没有公钥的git bash操作命令：  ssh-keygen -o // 生成公钥  cat ~/.ssh/id\_rsa.pub // 获取自己生成的公钥    生成公钥后，进入本人的 git 服务的 Settings，将生成的公钥填入，点击 add key    从git上面下拉项目  git clone ssh://git@192.168.50.66:30001/Front-end/jc-car-pcweb.git  注意： 从git复制的地址里面的localhost修改成 本地的ip地址：192.168.50.66  查看自己的文件目录  cd jc-car-pcweb/  git checkout -b dev origin/dev  gitk (git 的 gui 工具，可以看一下现在分支的情况)     1. 安装npm依赖   4.1 安装vscode （默认以管理员的身份运行）  注意：在自己的nodejs安装目录下面创建两个文件夹：node\_global 和 node\_cache , 然后打开自己的控制面板，配置自己的用户环境变量，变量名： NODE\_PATH 变量值： nodej的安装目录  4.2 打开终端，执行：  npm install //安装依赖包  ./project.sh dev //运行项目  ./project.sh build //打包文件  Tips：  常用指令：  ls：查看当前目录下的所有子目录  git clone 代码地址  cd 进入代码目录  git checkout -b 子分支名 origin/子分支名 (切换到开发分支)  gitk (git 的 gui 工具，可以看一下现在分支的情况)  如果有子代码库执行 git submodule update --init --recursive，然后 cd common，会  发现 common 目录下已经有内容  常见的react文件：  project.sh 配置文件 |

1. **刚接手项目的时候阅读common/README.md文件，了解当前项目的项目代码结构和业务模块逻辑，以及一些常用的类...**



**一： 现有框架下，新页面开发的流程（最好贴一点代码举例）**

1. 按照prd配置路由和接口

主要的文件有三个：

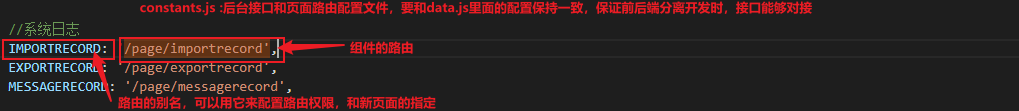
mock/data.Js :自己伪造的假数据（按照接口来配置，和constants.js文件里面配置 的接口保持一致）

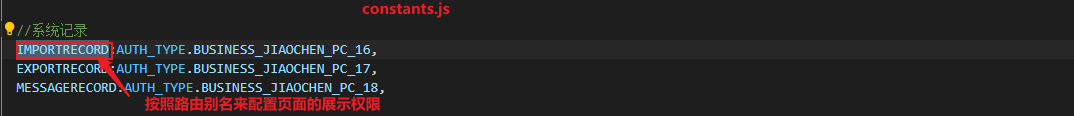
Src/project/until/constants.js: 前后端对接的文件（一些常量），我们在里面配置路 由和接口，新页面的展示也在该文件里面配置

Umirc.js: umi.js的配置文件，里面配置自己新页面的路由，配置好的路由要和 constants.js里面的路由保持一致

* 1. 路由和权限







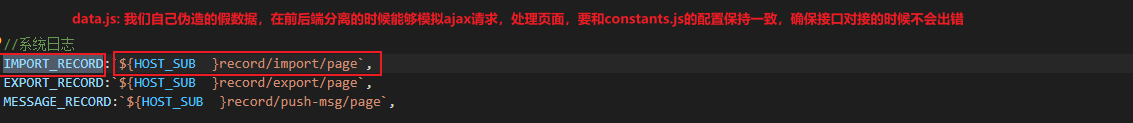
（2）权限配置



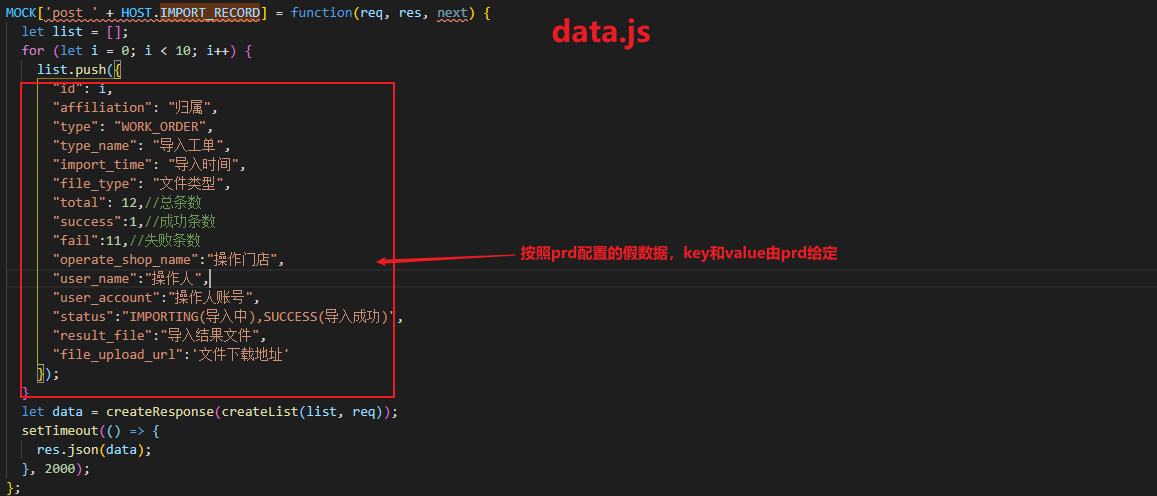


（3）接口配置：





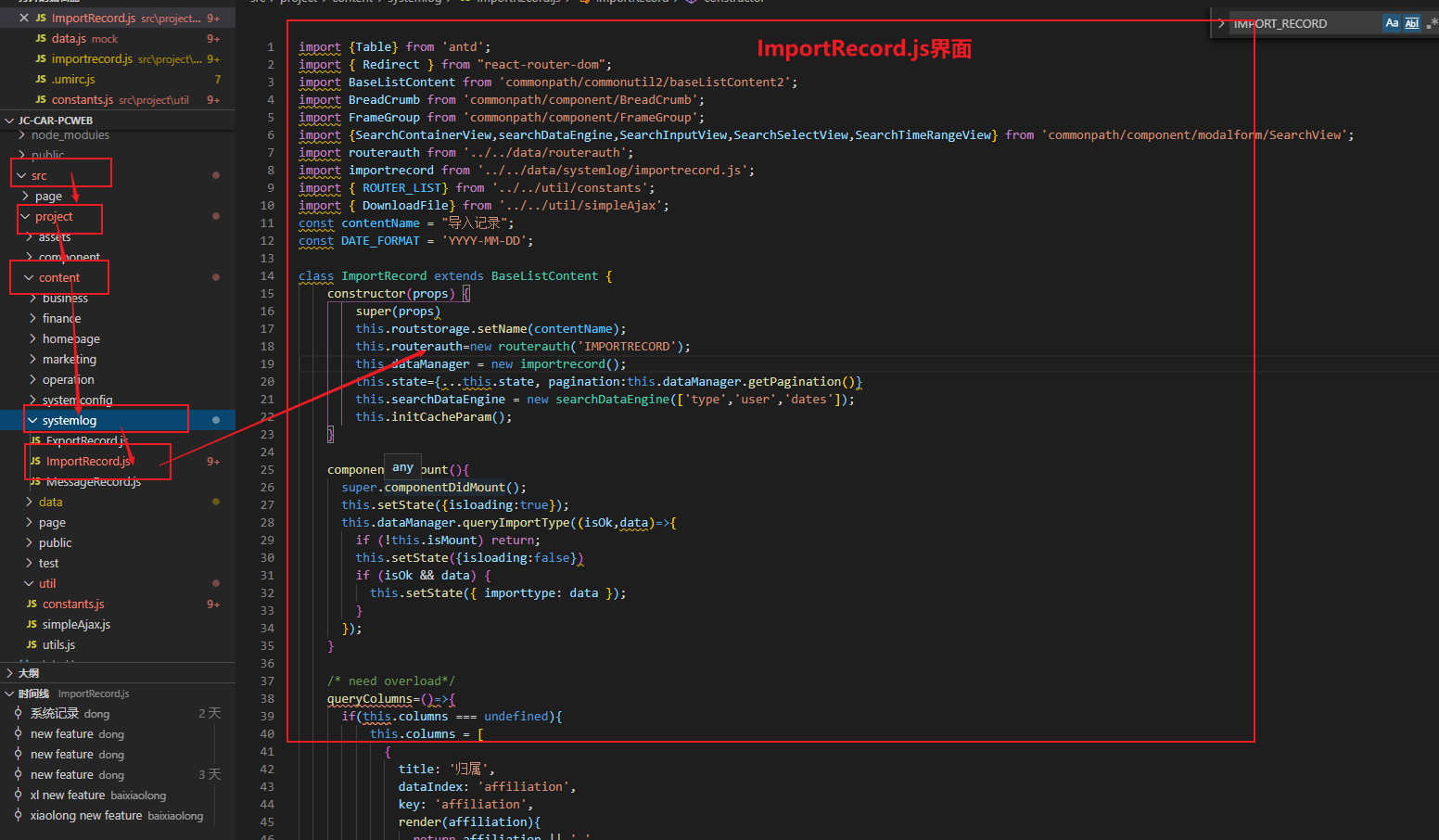
--自己造的假数据



2 按照自己在umirc里面配置的组件位置找到该组件，编辑该组件的代码

1. 页面编辑：

根据需求文档确认当前页面所使用的子组件，然后创建这些组件，并且导入他们所继承的基类组件和一些用到的方法（新手： 在项目里面找相同的已经实现的组件，捋清楚他们的逻辑，复制粘贴，然后按照自己的prd更改相关接口的参数和方法）



1. 每个页面都有自己与之对应的数据来源文件（该文件是当前组件页面所私有的，其他的页面不能访问）



1. 配置自己的数据来源文件

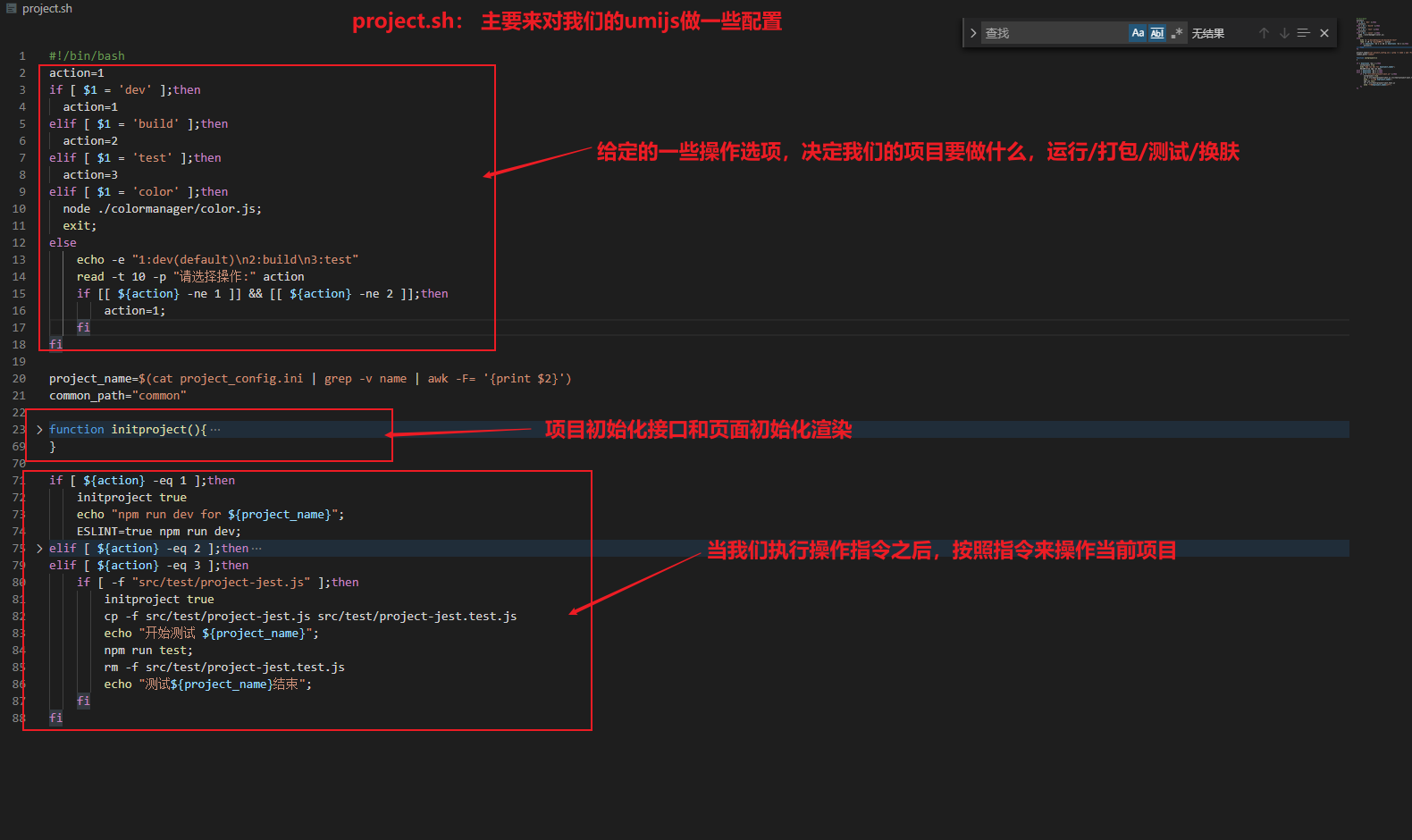


**二 project.sh里面做了些什么事情**

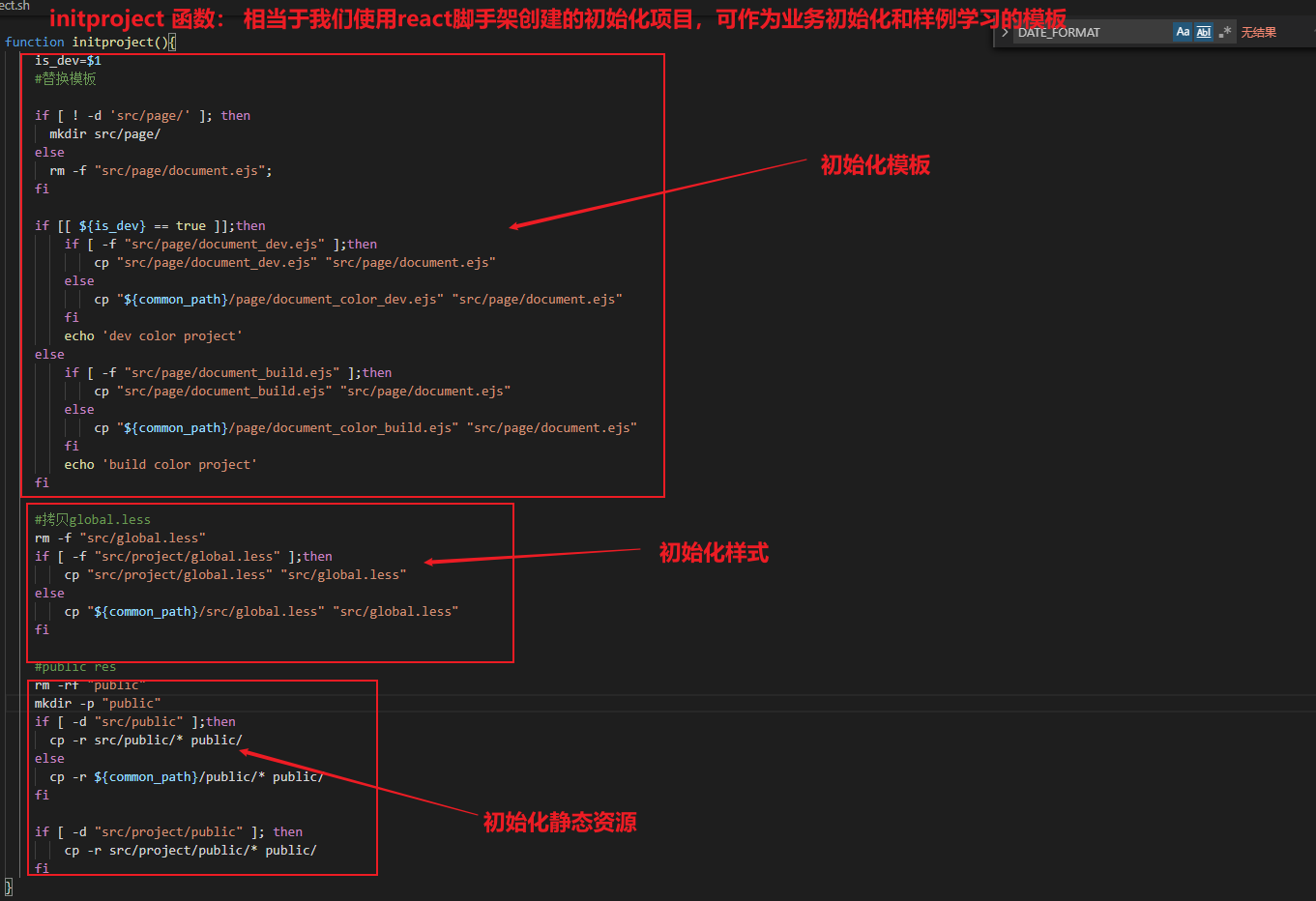
徐老师： project.sh主要是做umijs的一些配置的

我的理解：

配置一些操作选项，来决定如何处理我们的项目，（dev/bulid/test/color）



initProject函数： 可作为业务产品初始化和样例学习的模板



1. **接口数据传递路径**

|  |
| --- |
| （1）在componentDidMount（）生命周期函数里面里调用数据请求的方法     1. 数据来源文件里面通过ajax请求来获取数据      1. 使用当前取回来的数据   --通过生命周期函数获取的数据    ----通过依赖的数据文件获取的数据    ·难点：  配置下拉搜索框，数据请求的问题，  ·解决：  做一个判断，（如果没有请求到数据，就会报错，不会刷新页面），当请求到数据的话，再做渲染。 |

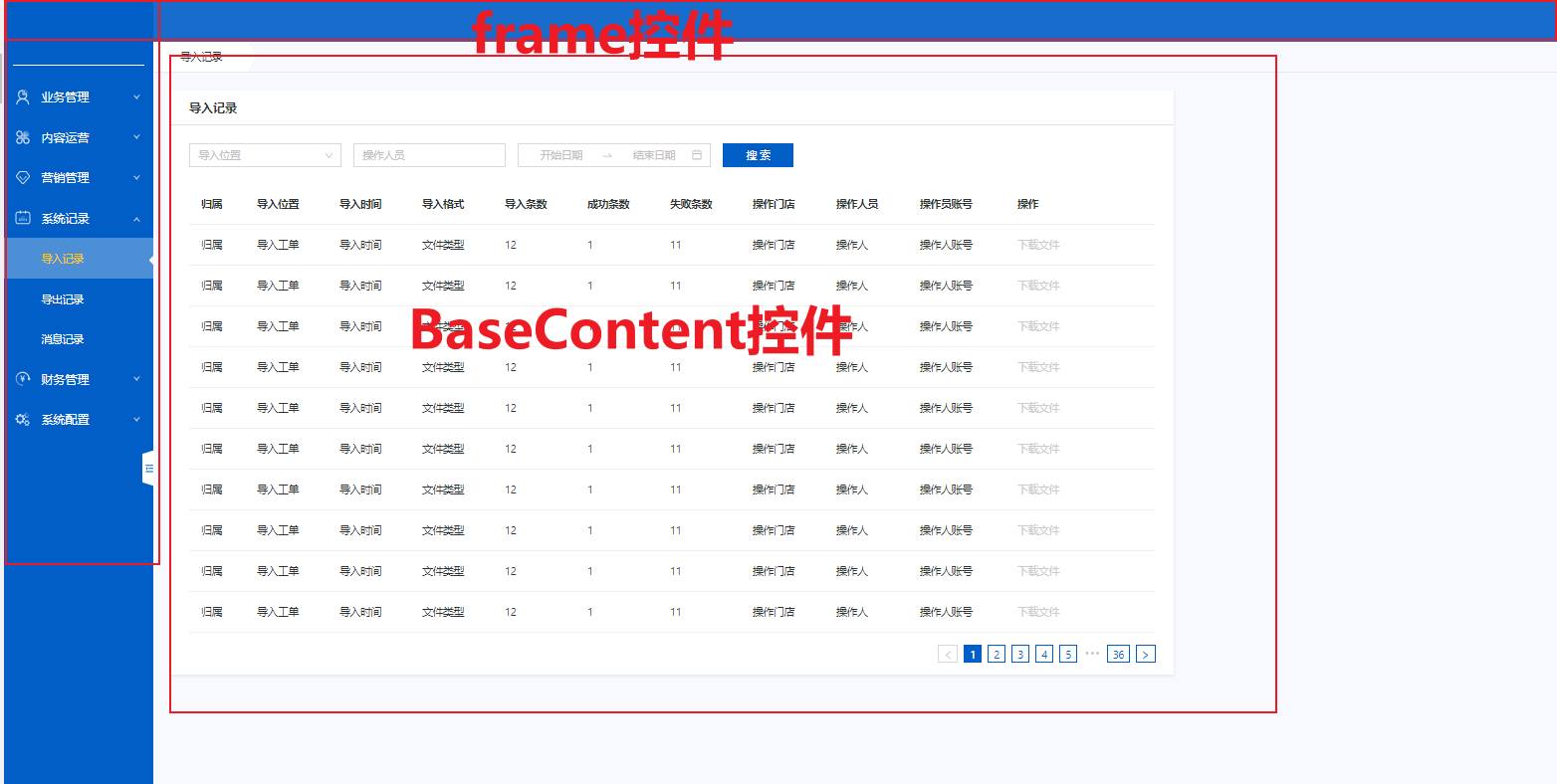
**四．自己写一个新控件的流程是什么**

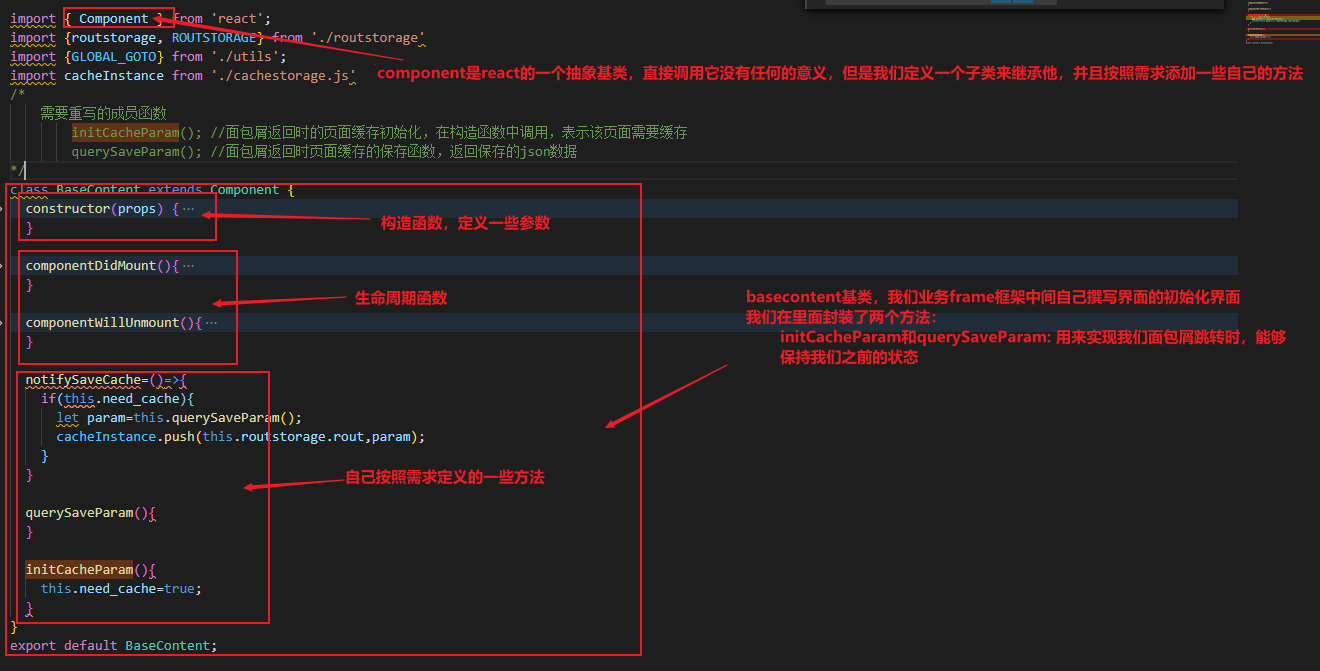
1. 从react继承 component ,
2. 按照自己的需求来添加方法，对react原生组件做一层封装
3. 当又有新的需求时，可以接着去添加新的方法，再去封装

注意： 在添加新的方法时，如果要使用其他的方法库，需要用import去导入所依赖的文件

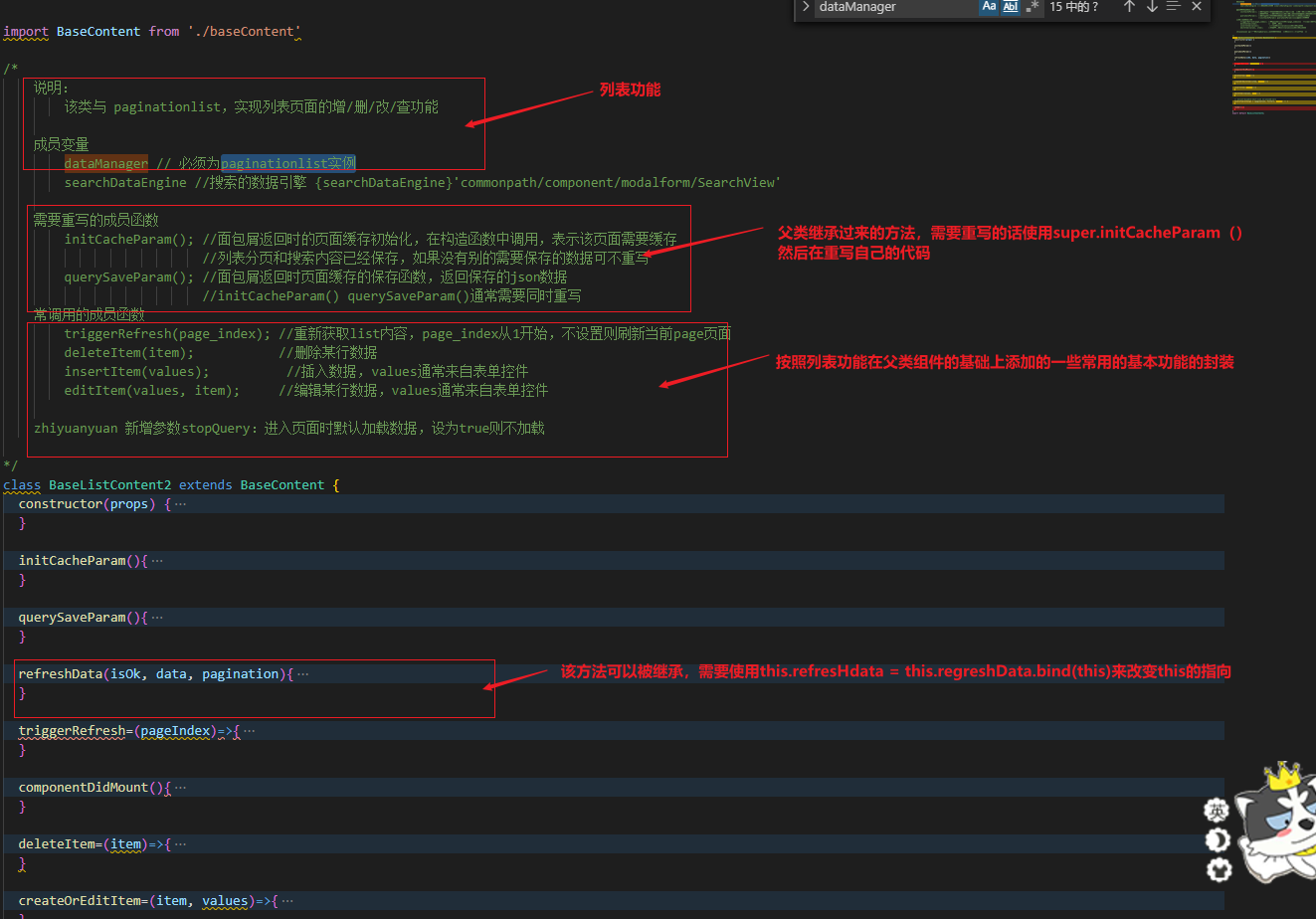
例如：导入页面的开始

1 basecontent.js：我们要编辑的所有组件的基类，继承自react的component组件



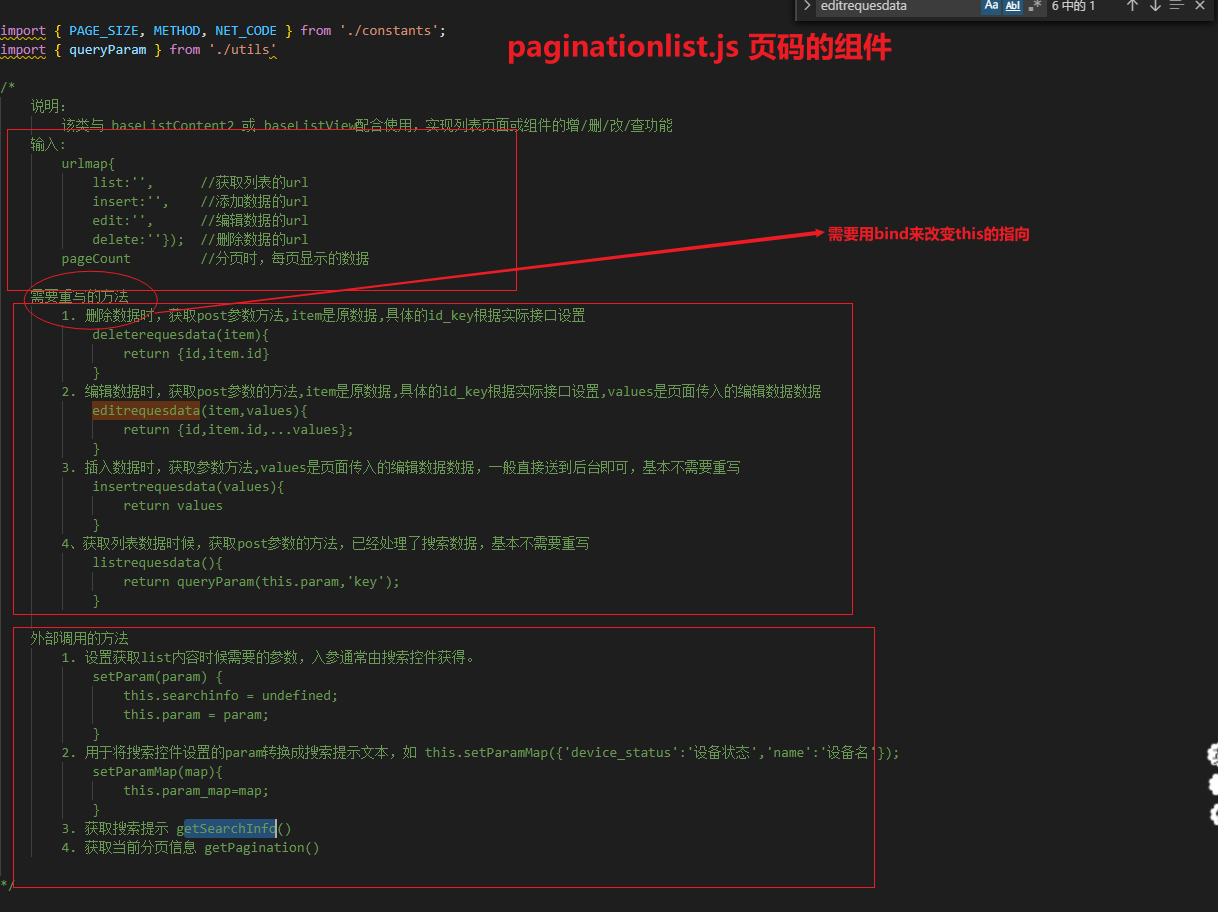


2 baseListContent.js 继承 baseContent.Js ： 为了实现列表页一个基类，为了更便捷的操作列表，封装了一些列表页的方法：（增删查改，列表刷新）

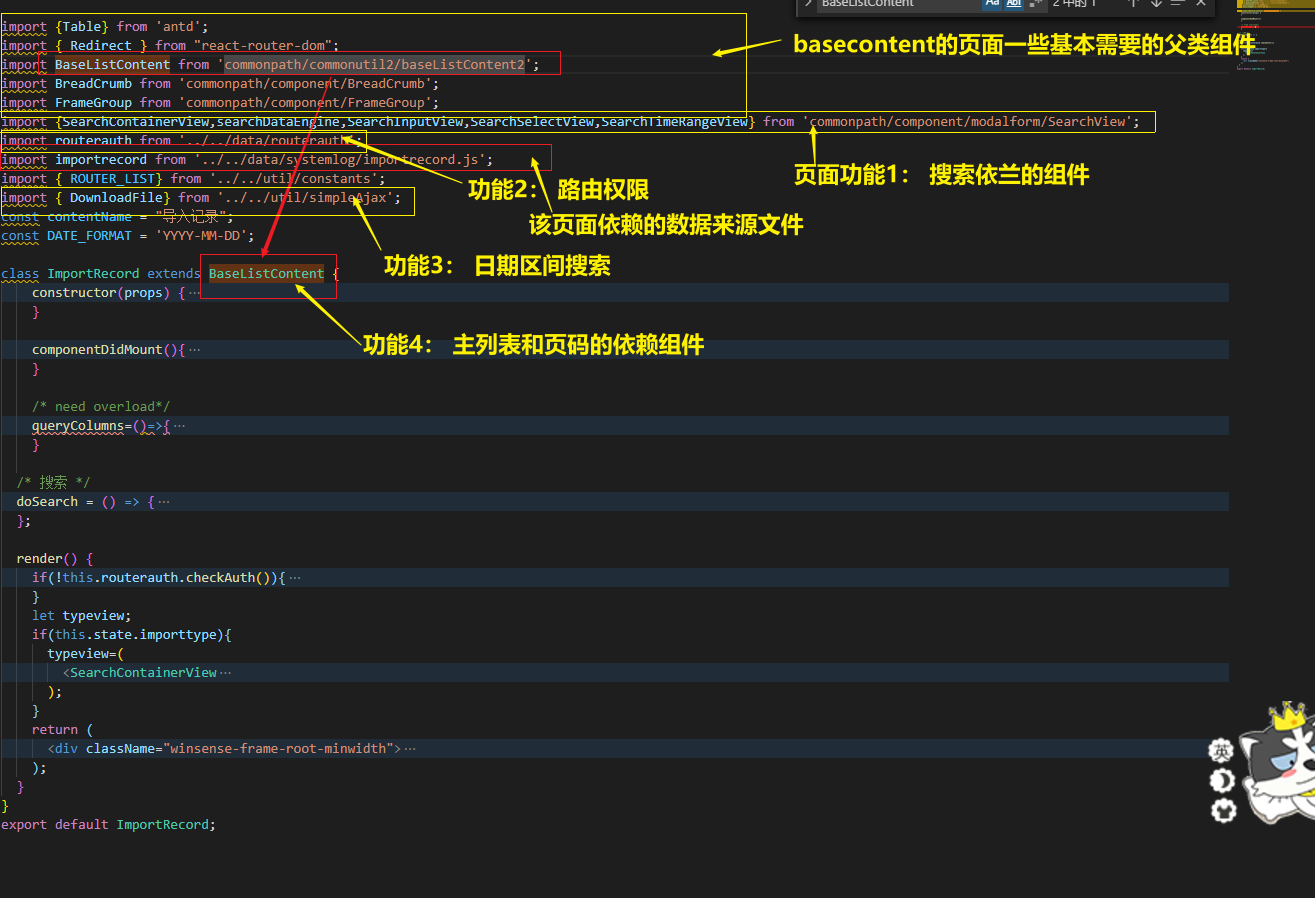


注意： 按照功能和需求，可以开发和他配套的组件，便于功能实现

页码组件： paginationlist.js



3 自己的导入记录页面importRecord继承了basecontentlist2.js :实现主要功能4，下拉搜索需要导入其他的一些依赖组件，如搜索类。。。



**五 设计控件接口需要考虑的问题有哪些**

1 对于基本组件，我们要考虑普适性和可继承性

例如： 我们的基础列表类，向外暴露的是对增删查改的一些方法，当我们开发其他组 件的时候，我们可以直接调用这些方法，就算重写的话，也只是一些轻微的改动

2 业务组件，更多的是业务逻辑和数据传输