# 之前的笔记链接

react全家桶学习笔记1：[路由基础配置](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记1-路由基础配置.docx)

react全家桶学习笔记2：[首页布局和菜单渲染](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记2-首页布局和菜单渲染.docx)

react全家桶学习笔记3：[二级路由配置](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记3-二级路由配置.docx)

react全家桶学习笔记4： [配置菜单默认项](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记4-设置菜单默认项.docx)

react全家桶学习笔记5： [axios配置网络请求](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记5-axios设置网络请求.docx)

react全家桶学习笔记6：[用户列表渲染](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记6-antd用户列表渲染.docx)

react全家桶学习笔记7：[路由传参](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记7-路由传参.docx)

react全家桶学习笔记8: [状态机初始化配置](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记8-状态机初始配置.docx)

react全家桶学习笔记9: [redux获取分类数据](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记9-redux获取分类数据.docx)

react全家桶学习笔记10：[渲染一二级分类数据以及优化](file:///G:\C以及其他教程\Web开发2023\web开发2023视频教程\凡云1\react零基础学习资料+学习笔记\react全家桶笔记\react全家桶学习笔记10-redux渲染一二级分类以及优化处理.docx)，

react全家桶学习笔记11：[redux获取用户数据](react全家桶学习笔记11-获取用户数据.docx)

react全家桶学习笔记12：自定义[Hook](react全家桶学习笔记12-自定义Hook-封装面包屑.docx)和封装面包屑

react全家桶学习笔记13：[登录功能实现](react全家桶学习笔记13-登录功能.docx)

react全家桶学习笔记14 ：[筛选权限菜单](react全家桶学习笔记14-筛选权限菜单.docx)

# 我们来实现登录和未登录提示功能

# 上一节我们学习了登录功能的实现，其实这个是远远不够的，一个网站的每一个链接都应该有判断用户是否登录，只有登录了才能访问，没有登录需要把他重定向到登录页面，如果他还不是注册用户，需要引导他去注册。

# 下面我们就来学习这个功能，老师偷偷写了一个组件，我根据推导也写了一个src/router/AddRouth.jsx组件，大概是下面的代码

|  |
| --- |
| import {Navigate, Route, Routes, *useLocation*, useNavigate} from "react-router-dom"; import *UsersPage* from "../pages/index/users/UsersPage.jsx"; import *RolesPage* from "../pages/index/roles/RolesPage.jsx"; import *ProductsCategory* from "../pages/index/products/ProductsCategory.jsx"; import *ProductsList* from "../pages/index/products/ProductsList.jsx"; import *FinancesSale* from "../pages/index/finances/FinancesSale.jsx"; import {*lazy*, *useState*} from "react"; import { *Modal* } from 'antd' import *IndexPage* from "../pages/index/IndexPage.jsx"; import *HomePage* from "../pages/index/home/HomePage.jsx"; import *ProductsUpdateDynamic* from "../pages/index/products/ProductsUpdateDynamic.jsx"; import *ProductsUpdate* from "../pages/index/products/ProductsUpdate.jsx"; import *LoginPage* from "../pages/login/LoginPage.jsx"; const NotFound = *lazy*(()=>import("../pages/404/NotFound.jsx")) ;  let routes = [  {path:"home" ,element :<HomePage/>,authPath:'/home'},  {path:"users" ,element :<UsersPage/>,authPath:'/home/user'},  {path:"roles" ,element :<RolesPage/>,authPath:'/home/role'},  {path:"products/category" ,element :<ProductsCategory/>,authPath:'/home/product/category'},  {path:"products/list" ,element :<ProductsList/>,authPath:'/home/product/list'},  {path:"finances/sale" ,element :<FinancesSale/>,authPath:'/home/datav/sale'}, ]  const *AddRoute* = () => {  const [authRoutes,setAuthRoutes] = *useState*([])  const navigate = useNavigate()  let { pathname } = *useLocation*()  *console*.log("当前路径",pathname)  //1.判断为登录用户访问的利用是否是当前下面在存在的路由  const isRoute = routes.some(item=>"/"+ item.path === pathname)  // console.log(isRoute)  // 如果用户访问的路由是我们项目的路由，isRoute就是true  if(isRoute){  //2.如果用户访问的是我们项目的路由，需要判断他是否已经登录  let userInfo = *localStorage*.userInfo  //2.1进入if，表明用户已经登录  if(userInfo){  //2.2获取用户能够访问的权限路径  if(authRoutes.length == 0){  let {role:{menus}} = *JSON*.parse(userInfo)  setAuthRoutes(routes.filter(item=>menus.includes(item.authPath)))  }  } else { //3.进入这里说明用户没有登录，引导他去登录  *Modal*.warning({  title:'未登录提示',  content:'你还未登录，请先登录',  afterClose:()=>{  navigate('/login')  }  })  }  }    return (  <Routes>  <Route path="/" element ={<Navigate to="/home"/>}></Route>  <Route path="home" element={<HomePage/>}></Route> {/\*这里不能使用<Navigate to="/home"/>，会渲染2次\*/}  <Route path="login" element={<LoginPage/>}></Route>  <Route path="/" element={<IndexPage/>}>  <Route path="users" element={<UsersPage/>}></Route>  <Route path="roles" element={<RolesPage/>}></Route>  <Route path="products/category" element={<ProductsCategory/>}></Route>  <Route path="products/list" element={<ProductsList/>}></Route>  {/\*动态路由\*/}  <Route path="products/update/:id" element={<ProductsUpdateDynamic/>}></Route>  {/\*state传参\*/}  <Route path="products/update" element={<ProductsUpdate/>}></Route>  {/\*配置财务统计的销售金额路由\*/}  <Route path="finances/sale" element={<FinancesSale/>}></Route>  </Route>  {/\*<Route path='\*' element={<NotFound/>}></Route>\*/}  </Routes>  ); };  export default *AddRoute*; |

## 我暂时不知道它这么被调用和用在那里？经过思考一番，我觉得可能是把app.jsx里面的业务代码抽取出来。保存为AddRoute.jsx文件，然后在App.jsx文件里面导入这个组件，精简后的App.jsx代码如下，在精简之前，需要把原来的代码复杂一个备份。

|  |
| --- |
| import *AddRoute* from "./router/AddRoute.jsx"; import {BrowserRouter} from "react-router-dom"; import {*Suspense*} from 'react';  const *App* = () => {  return (  <div>  <Suspense fallback={<h1>**加载中...**</h1>}>  <BrowserRouter>  <AddRoute/>  </BrowserRouter>  </Suspense>   </div>  ); };  export default *App*; |

### 测试了一下，是可以正常运行的

## 虽然不清楚，但是处理逻辑大概如下，我们现在操作的是AddRoute.jsx组件

## 1. 判断为登录用户访问的利用是否是当前下面在存在的路由

|  |
| --- |
|  |

## 2.如果用户访问的是我们项目的路由，需要判断他是否已经登录

|  |
| --- |
|  |

## 3.如果用户没有登录，我们在else分支里面处理，需要用到antd的Modal组件

|  |
| --- |
|  |

### 注意：在路由那里，如果路由是/home，需要直接指定对应的组件而不要使用Navigate to={组件}的方式，这样子会渲染2次

## 4.为了避免重复渲染，我们把404的路由关闭了，但是我们又需要它，所以我们需要动态生成404路由

|  |
| --- |
|  |

### 还需要在下面动态控制404页面的显示

|  |
| --- |
|  |

## 5.还有一个问题，就是用户是登录了，而且我们又有对应的路由，但是这个用户没有权限访问这个路由，这个怎么办？需要修改一下判断404的条件，出来对isRoute取反，还需要3给条件，1是用户已经登录2是路径在项目中是存在的3是用户没有权限访问这个路径。所以要改为下面的代码

|  |
| --- |
|  |

## 6.这里有一个问题，如果你在AddRoute下面的渲染页面的路由的位置使用以前的代码，你好发现即使用户没有权限，他在浏览器上面输入路由一样可以访问他没有权限访问的路由，展示react的问题，所以我们需要在渲染页面路由的位置根据用户的权限来动态渲染路由

|  |
| --- |
|  |

### 测试一下，功能正常