React项目-后台管理系统

尚硅谷前端研究院

# 第1章: 准备

## 项目描述

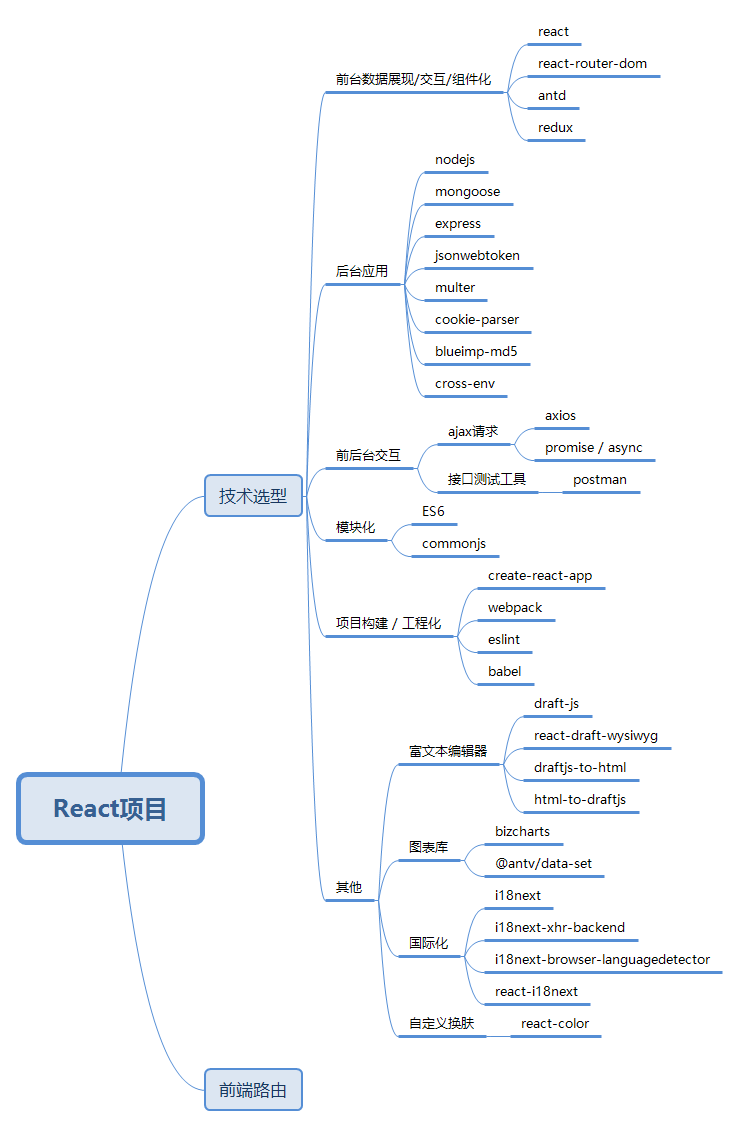
1. 此项目为一个前后台分离的后台管理的SPA, 包括前端PC应用和后端应用
2. 包括用户管理 / 商品分类管理 / 商品管理 / 权限管理 / 国际化 / 自定义换肤等功能模块
3. 前端: 使用React全家桶 + Antd + Axios + ES6 + Webpack + Echarts等技术
4. 后端: 使用Node + express + mongoose等技术
5. 采用模块化、组件化、工程化的模式开发

## 项目功能界面

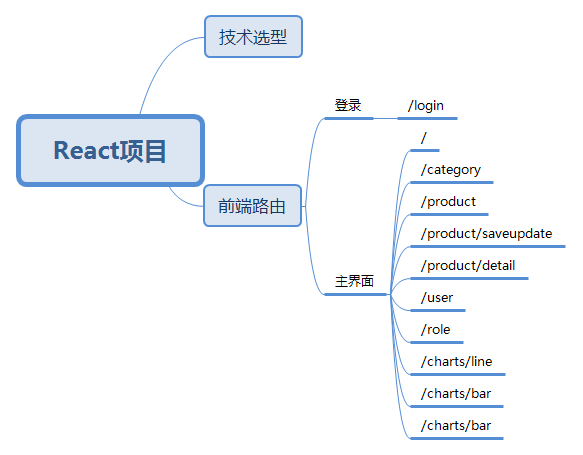


其它的界面功能运行final版应用查看

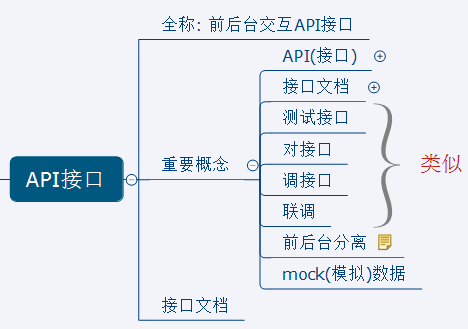
## 技术选型



## 前端路由



## API接口



## 你能从此项目中学到什么?

### 流程及开发方法

1. 熟悉一个项目的**开发流程**
2. 学会**模块化、组件化、工程化**的开发模式
3. 掌握使用**create-react-app**脚手架初始化react项目开发
4. 学会使用**node+express+mongoose**搭建后台开发

### React插件或第三方库

1. 掌握使用**react-router-dom**开发单页应用
2. 掌握使用**axios**与后端进行数据交互
3. 掌握使用**antd**组件库构建界面
4. 学会使用**redux+react-redux+redux-thunk**管理应用组件状态
5. 学会使用**mongoose**操作mongodb数据库
6. 学会使用**express**搭建后台路由
7. 学会使用**blueimp-md5**对密码进行MD5加密处理

### 功能

1. 学会开发登录/token验证功能
2. 学会开发权限管理功能
3. 学会开发国际化功能
4. 学会开发自定义换肤功能
5. 学会开发商品/分类管理功能

## npm/yarn常用命令

|  |
| --- |
| ## 初始化项目:      yarn init -y     npm init -y  ## 下载项目的所有声明的依赖:      yarn     npm install  ## 下载指定的运行时依赖包:      yarn add webpack@3.2.1     npm install webpack@3.2.1 -S  ## 下载指定的开发时依赖:      yarn add webpack@3.2.1 -D     npm install webpack@3.2.1 -D  ## 全局下载指定包:      yarn global add webpack     npm install webpack -g  ## 删除依赖包:      yarn remove webpack     npm remove webpack -S     yarn global remove webpack     npm remove webpack -g  ## 运行项目中配置的script:      yarn run xxx     npm run xxx      ## 查看某个包的信息:      yarn info xxx     npm info xxx  ## 设置淘宝镜像:      yarn config set registry https://registry.npm.taobao.org     npm config set registry https://registry.npm.taobao.org |

## git常用基本命令

|  |
| --- |
| **\*** git config --global user.name "username" //配置用户名 **\*** git config --global user.email "xx@gmail.com" //配置邮箱  **\*** git init //初始化生成一个本地仓库  **\*** git add \* //添加到暂存区  **\*** git commit –m “message” //提交到本地仓库  **\*** git remote add origin url //关联到远程仓库  **\*** git push origin master //推送本地master分支到远程master分支  **\*** git checkout -b dev //创建一个开发分支并切换到新分支  **\*** git push ogigin dev //推送本地dev分支到远程dev分支  **\*** git pull origin dev //从远程dev分支拉取到本地dev分支  **\*** git clone url //将远程仓库克隆下载到本地  \* git checkout -b dev origin/dev // 克隆仓库后切换到dev分支 |

# 第2章: 应用开发详解

## 2.1. 开启项目开发

### 2.1.1. 使用create-react-app(脚手架)搭建项目

1. create-react-app是react官方提供的用于搭建基于react+webpack+es6项目的脚手架
2. 操作:

npm install -g create-react-app : 全局下载工具

create-react-app react-admin :下载模板项目

cd gzhipin

npm start

访问: localhost:3000

### 2.1.2. 编码测试与打包发布项目

1. 编码测试

npm start

访问: [http://localhost:3000](http://localhost:8080)

编码, 自动编译打包刷新(live-reload), 查看效果

1. 打包发布

npm run build

npm install -g serve

serve build

访问: [http://localhost:5000](http://localhost:9000)

## 2.2. 功能需求分析

演示项目功能, 对功能模块进行分析说明

## 2.3. 项目源码基本目录设计

### 2.3.1. 基本结构



### 2.3.2. App.js

|  |
| --- |
| **import** React, {Component} **from 'react'** */\* 应用根组件  \*/* **class** App **extends** Component {  render() {  **return** (  <**div**>App</**div**>  )  } }  **export default** App |

### 2.3.3. index.js

|  |
| --- |
| **import** React **from 'react' import** ReactDOM **from 'react-dom'  import** App **from './App'** ReactDOM.render(<**App** />, **document**.getElementById(**'root'**)) |

## 2.4. 引入antd

### 2.4.1. 下载组件库包

|  |
| --- |
| **yarn add antd** |

### 2.4.3. 实现组件的按需打包

1. 下载依赖模块

|  |
| --- |
| **yarn add react-app-rewired customize-cra babel-plugin-import -D** |

1. 定义加载配置的js模块: config-overrides.js

|  |
| --- |
| **const** { *override*, *fixBabelImports*} = *require*(**'customize-cra'**);  ***module***.**exports** = *override*(  *fixBabelImports*(**'import'**, {  **libraryName**: **'antd'**,  **libraryDirectory**: **'es'**,  **style**: **true**,  }), ); |

* 修改配置: package.json

|  |
| --- |
| **"scripts"**: {  **"start"**: **"react-app-rewired start"**,  **"build"**: **"react-app-rewired build"**,  **"test"**: **"react-app-rewired test"**,  **"eject"**: **"react-scripts eject"** } |

### 2.4.4. 在应用中使用antd组件

|  |
| --- |
| **import** React **from 'react' import** ReactDOM **from 'react-dom' import** {Button} **from 'antd'** ReactDOM.render(  <**Button type='primary'**>学习</**Button**>,  **document**.getElementById(**'root'**) ) |

### 2.4.5. 自定义antd主题

需求:

使antd的默认基本颜色从Blue变为Green

实现:

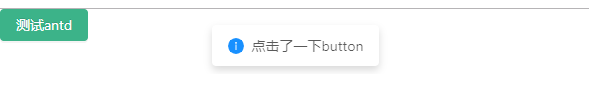
下载工具包:

|  |
| --- |
| **yarn add less less-loader -D** |

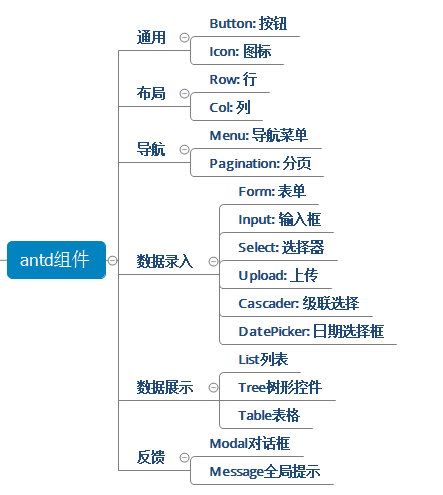
修改config-overrides.js

|  |
| --- |
| **const** { *override*, *fixBabelImports*, *addLessLoader* } = *require*(**'customize-cra'**);  ***module***.**exports** = *override*(  *fixBabelImports*(**'import'**, {  **libraryName**: **'antd'**,  **libraryDirectory**: **'es'**,  **style**: **true**,  }),  *addLessLoader*({  **javascriptEnabled**: **true**,  **modifyVars**: { **'@primary-color'**: **'#1DA57A'** },  }), ); |

### 2.4.6. 应用中使用的组件



### 2.4.7. 项目中用到的antd组件

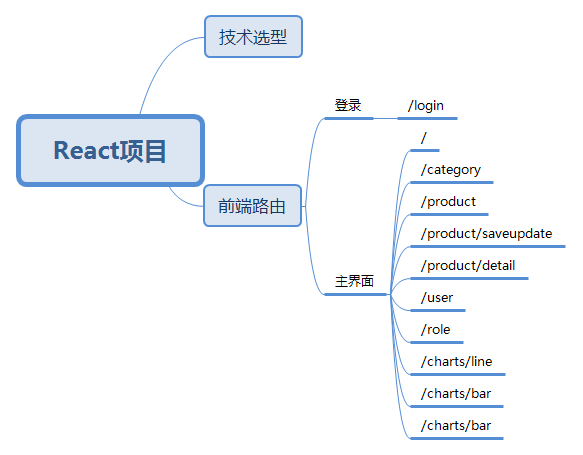


## 2.5. 引入路由

### 2.5.1. 下载路由包: react-router-dom

|  |
| --- |
| **yarn add react-router-dom** |

### 2.5.2. 前台应用路由



### 2.5.3. 路由的配置文件

|  |
| --- |
| */\*  所有路由配置文件  \*/* **import** Home **from '../components/home'**; **import** Login **from '../components/login'**;  **const** routes = [  {  **path**: **'/'**, *// 路由路径* **component**: Home, *// 路由组件* **exact**: **true** *// 是否全匹配路由路径* },  {  **path**: **'/login'**, *// 路由路径* **component**: Login, *// 路由组件* **exact**: **true** *// 是否全匹配路由路径* }, ];  **export default** routes; |

### 2.5.4. home路由组件: components/home/index.jsx

|  |
| --- |
| **import** React, { Component } **from 'react'**;  **export default class** Home **extends** Component {  render() {  **return** <**div**>  Home  </**div**>;  } } |

### 2.5.5. login路由组件: components/login/index.jsx

|  |
| --- |
| **import** React, { Component } **from 'react'**;  **export default class** Login **extends** Component {  render() {  **return** <**div**>  Login   </**div**>;  } } |

### 2.5.6. not-match路由组件: components/not-match/index.jsx

|  |
| --- |
| **import** React, { Component } **from 'react'**;  **export default class** NotMatch **extends** Component {  render() {  **return** <**div**>  NotMatch   </**div**>;  } } |

### 2.5.7. 映射路由: App.js

|  |
| --- |
| **import** React, { Component } **from 'react'**; **import** { BrowserRouter **as** Router, Route, Switch } **from 'react-router-dom'**;  **import** routes **from './config/routes'**; **import** NotMatch **from './components/not-match'**;  **export default class** App **extends** Component {  render() {  **return** <**Router**>  <**Switch**>  {  routes.map((route, index) => {  **return** <**Route** {...route} **key=**{index}/>  })  }  {*/\* 不写path就默认匹配所有路径 \*/*}  <**Route component=**{NotMatch}/>  </**Switch**>  </**Router**>;  } } |

## 2.6.引入redux

### 2.6.1. redux/store.js

|  |
| --- |
| */\*  用来集中式管理所有状态数据模块  \*/* **import** { createStore, *applyMiddleware* } **from 'redux'**; **import** thunk **from 'redux-thunk'**; **import** { **composeWithDevTools** } **from 'redux-devtools-extension'**;  **import** reducers **from './reducers'**;  **let** store; **if** (process.env.**NODE\_ENV** === **'development'**) {  store = createStore(reducers, **composeWithDevTools**(*applyMiddleware*(thunk))); } **else** {  store = createStore(reducers, *applyMiddleware*(thunk)); }  **export default** store; |

### 2.6.2. redux/reducers.js

|  |
| --- |
| */\*  包含n个 根据prevState和action来生成newState函数 模块  \*/* **import** { *combineReducers* } **from 'redux'**;  **function** *user*(prevState = {}, action) {  **switch** (action.**type**) {  **default** :  **return** prevState;  } }  **export default** *combineReducers*({  *user* }) |

### 2.6.3. redux/action-creators.js

|  |
| --- |
| */\*  包含n个 生成action对象的工厂函数 模块  action对象： { type: 要执行的操作类型, data: 要操作的数据 }  \*/* |

### 2.6.4. redux/action-types.js

|  |
| --- |
| */\*  action的type类型常量模块  \*/* |

### 2.6.2. index.js 入口文件

|  |
| --- |
| **import** React **from 'react'**; **import** ReactDOM **from 'react-dom'**; **import** { *Provider* } **from 'react-redux'**;  **import store from './redux/store'**; **import** App **from './App'**;  ReactDOM.render(  <**Provider store=**{**store**}>  <**App** />  </**Provider**>, ***document***.getElementById(**'root'**)); |

## 2.7. Login组件(不与后台交互)

### 2.7.1. 静态组件

1. assets/images/logo.png



1. login/index.less

|  |
| --- |
| .**login**{  **width**: 100%;  **height**: 100%;  **background-image**: **url**(**"./bg.jpg"**);  **background-repeat**: **no-repeat**;  **background-size**: **cover**;  .**login-header**{  **height**: 80**px**;  **background-color**: **rgba**(21, 20, 13, 0.5);  **display**: **flex**;  **align-items**: **center**;  **img**{  **margin**: 0 15**px** 0 30**px**;  **width**: 45**px**;  **height**: 45**px**;  }  **h1**{  **font-size**: 25**px**;  **color**: **#fff**;  }  }  .**login-content**{  **margin**: 100**px auto** 0;  **padding-top**: 20**px**;  **width**: 400**px**;  **height**: 310**px**;  **background-color**: **#fff**;  **h2**{  **margin**: 20**px** 0;  **font-size**: 30**px**;  **font-weight**: **bold**;  **text-align**: **center**;  }  .**login-form**{  **margin**: 0 **auto**;  **width**: 80%;  .**login-form-button**{  **width**: 100%;  }  }  } } |

1. login/index.jsx

|  |
| --- |
| */\* 用户登陆的路由组件  \*/* **import** React, {Component} **from 'react'**; **import** {  Form, Icon, Input, Button, } **from 'antd'**;  **import** logo **from './logo.png'**; **import './index.less'**;  **const** Item = Form.*Item*;  **export default class** Login **extends** Component {    *handleSubmit* = (e) => {  e.preventDefault();  }    render () {  **return** (  <**div className="login"**>  <**header className="login-header"**>  <**img src=**{logo} **alt="logo"**/>  <**h1**>React项目: 后台管理系统</**h1**>  </**header**>  <**section className="login-content"**>  <**h2**>用户登录</**h2**>  <**Form onSubmit=**{**this**.*handleSubmit*} **className="login-form"**>  <**Item**>  <**Input prefix=**{<**Icon type="user" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **placeholder="请输入用户名"** />  </**Item**>  <**Item**>  <**Input prefix=**{<**Icon type="lock" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **type="password" placeholder="请输入密码"** />  </**Item**>  <**Item**>  <**Button type="primary" htmlType="submit" className="login-form-button"**>  登录  </**Button**>  </**Item**>  </**Form**>  </**section**>  </**div**>  )  } } |

### 2.7.2. 前台表单验证与数据收集

|  |
| --- |
| **import** React, {Component} **from 'react'**; **import** {  Form, Icon, Input, Button, message } **from 'antd'**;  **import** logo **from './logo.png'**; **import './index.less'**;  **const** Item = Form.*Item*;  **class** Login **extends** Component {    *login* = (e) => {  *// 禁止表单的默认行为* e.preventDefault();  *// 检验表单是否验证通过。 通过才能发请求* **this**.**props**.**form**.validateFields((error, values) => {  **if** (!error) {  *// 发送请求，请求登录* ***console***.log(values);  } **else** {  *// 表单校验错误，提示错误* message.error(**'请正确填写用户名和密码'**, 2);  }  })    }    *// 校验密码的方法  validator* = (rule, value, callback) => {  *// 校验密码的正则* **const** pwdReg = /^[a-zA-Z0-9\_]+$/;  **if** (!value) {  callback(**'请输入密码'**);  } **else if** (value.**length** < 5) {  callback(**'密码必须大于5位'**);  } **else if** (value.**length** > 12) {  callback(**'密码必须小于12位'**);  } **else if** (!pwdReg.test(value)) {  callback(**'密码只能包含英文、数字、下划线'**);  } **else** {  *// 必须调用callback函数* callback();  }  }    render () {  **const** { getFieldDecorator } = **this**.**props**.**form**;    **return** (  <**div className="login"**>  <**header className="login-header"**>  <**img src=**{logo} **alt="logo"**/>  <**h1**>React项目: 后台管理系统</**h1**>  </**header**>  <**section className="login-content"**>  <**h2**>用户登录</**h2**>  <**Form onSubmit=**{**this**.*login*} **className="login-form"**>  <**Item**>  {  getFieldDecorator(  **'username'**,  {  *// 校验用户名的配置对象* **rules**: [  {**required**: **true**, **whitespace**: **true**, **message**: **'请输入用户名'**},  {**min**: 5, **message**: **'用户名必须大于5位'**},  {**max**: 12, **message**: **'用户名必须小于12位'**},  {**pattern**: /^[a-zA-Z0-9\_]+$/, **message**: **'用户名只能包含英文、数字、下划线'**}  ]  }  )(  <**Input prefix=**{<**Icon type="user" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **placeholder="请输入用户名"** />  )  }  </**Item**>  <**Item**>  {  getFieldDecorator(  **'password'**,  {  **rules**: [  {**validator**: **this**.*validator*}  ]  }  )(  <**Input prefix=**{<**Icon type="lock" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **type="password" placeholder="请输入密码"** />  )  }  </**Item**>  <**Item**>  <**Button type="primary" htmlType="submit" className="login-form-button"**>  登录  </**Button**>  </**Item**>  </**Form**>  </**section**>  </**div**>  )  } }  *// Form.create是高阶组件（HOC）的用法* **export default** Form.*create*()(Login); |

## 2.7. 运行server端项目

### 2.7.1. 说明

1. 咱们的项目是一个前后台分离的项目: 前台应用与后台应用
2. 后台应用负责处理前台应用提交的请求, 并给前台应用返回json数据
3. 前台应用负责展现数据, 与用户交互, 与后台应用交互

### 2.7.2. 运行后台应用

1. 确保启动mongodb服务
2. 启动服务器应用: npm start

### 2.7.3. API接口文档

### 2.7.4. 使用postman工具测试接口

1. postman是用来测试API接口的工具
2. postman可以看作活接口文档

## 2.8. 前后台交互ajax

### 2.8.1. 下载依赖包

|  |
| --- |
| **yarn add axios** |

### 2.8.2. 封装ajax请求模块

1. api/ajax.js

|  |
| --- |
| */\*\*  \* 封装ajax请求的模块  \* 要求：  \* 1. 返回值为promise  \* 2. 成功直接返回数据（去掉其它无用的）  \* 3. 失败统一处理错误  \*/* **import** axios **from 'axios'**; **import** { message } **from 'antd'**;  **export default function** ajax(url, data, method = **'GET'**) {  **let** promise;  *// 转化为大写* method = method.toUpperCase();    **if** (method === **'GET'**) {  *// GET 请求* promise = axios.get(url, {  **params**: data  });  } **else if** (method === **'POST'**) {  *// POST 请求* promise = axios.post(url, data);  } **else** {  *// 不处理，提示错误* message.error(**`**${method} **请求方式，暂不支持，请使用 GET / POST 代替`**);  }  *// 统一处理成功和失败* **return** promise  .then((res) => {  **return** res.**data**;  })  .catch((err) => {  ***console***.log(**'\*\*\*\*\*\*\*\*\*\* 请求失败啦 TnT \*\*\*\*\*\*\*\*\*\*'**);  ***console***.log(err);  ***console***.log(**'\*\*\*\*\*\*\*\*\*\* 赶紧处理上面错误啊 \*\*\*\*\*\*\*\*\*\*'**);  message.error(**`请求失败，具体错误请看控制台~~`**);  }) } |

1. api/index.js

|  |
| --- |
| */\*\*  \* 包含n个接口的请求函数模块  \* 返回值是promise  \*/* **import** *ajax* **from './ajax'**;  *// 请求登录函数* **export const** *reqLogin* = (username, password) => *ajax*(**'/login'**, {username, password}, **'POST'**); |

1. login/index.jsx

|  |
| --- |
| **import** React, {Component} **from 'react'**; **import** {  Form, Icon, Input, Button, message } **from 'antd'**;  **import** { *reqLogin* } **from '../../api'**;  **import** logo **from './logo.png'**; **import './index.less'**;  **const** Item = Form.*Item*;  **class** Login **extends** Component {    *login* = (e) => {  *// 禁止表单的默认行为* e.preventDefault();  *// 检验表单是否验证通过。 通过才能发请求* **this**.**props**.**form**.validateFields(**async** (error, values) => {  **if** (!error) {  *// 发送请求，请求登录* **const** { username, password } = values;  *// 请求登录* **const** result = **await** *reqLogin*(username, password);  *// 判断登录是否成功* **if** (result.**status** === 0) {  *// 登录成功* message.success(**'登录成功~'**, 1);  *// 跳转到admin页面* **this**.**props**.history.replace(**'/'**);  } **else** {  *// 登录失败* message.error(result.msg);  }  } **else** {  *// 表单校验错误，提示错误* message.error(**'请正确填写用户名和密码'**, 2);  }  })    }    *// 校验密码的方法  validator* = (rule, value, callback) => {  *// 校验密码的正则* **const** pwdReg = /^[a-zA-Z0-9\_]+$/;  **if** (!value) {  callback(**'请输入密码'**);  } **else if** (value.**length** < 5) {  callback(**'密码必须大于5位'**);  } **else if** (value.**length** > 12) {  callback(**'密码必须小于12位'**);  } **else if** (!pwdReg.test(value)) {  callback(**'密码只能包含英文、数字、下划线'**);  } **else** {  *// 必须调用callback函数* callback();  }  }    render () {  **const** { getFieldDecorator } = **this**.**props**.**form**;    **return** (  <**div className="login"**>  <**header className="login-header"**>  <**img src=**{logo} **alt="logo"**/>  <**h1**>React项目: 后台管理系统</**h1**>  </**header**>  <**section className="login-content"**>  <**h2**>用户登录</**h2**>  <**Form onSubmit=**{**this**.*login*} **className="login-form"**>  <**Item**>  {  getFieldDecorator(  **'username'**,  {  *// 校验用户名的配置对象* **rules**: [  {**required**: **true**, **whitespace**: **true**, **message**: **'请输入用户名'**},  {**min**: 5, **message**: **'用户名必须大于5位'**},  {**max**: 12, **message**: **'用户名必须小于12位'**},  {**pattern**: /^[a-zA-Z0-9\_]+$/, **message**: **'用户名只能包含英文、数字、下划线'**}  ]  }  )(  <**Input prefix=**{<**Icon type="user" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **placeholder="请输入用户名"** />  )  }  </**Item**>  <**Item**>  {  getFieldDecorator(  **'password'**,  {  **rules**: [  {**validator**: **this**.*validator*}  ]  }  )(  <**Input prefix=**{<**Icon type="lock" style=**{{ **color**: **'rgba(0,0,0,.25)'** }} />} **type="password" placeholder="请输入密码"** />  )  }  </**Item**>  <**Item**>  <**Button type="primary" htmlType="submit" className="login-form-button"**>  登录  </**Button**>  </**Item**>  </**Form**>  </**section**>  </**div**>  )  } }  *// Form.create是高阶组件（HOC）的用法* **export default** Form.*create*()(Login); |

### 2.8.3. 配置代理

package.json

|  |
| --- |
| **"proxy"**: **"http://localhost:5000"** |

## 2.9. Login组件(完成登陆功能)

### 2.9.1. 下载依赖

|  |
| --- |
| **yarn add store** |

### 2.9.2. utils/storageUtils.js

|  |
| --- |
| */\* local存储工具模块 使用兼容所有浏览器的local存储的工具库: store  \*/  /\*\*  \* 本地存储的工具模块  \* 使用了兼容所有浏览器的工具库 store  \*/* **import store from 'store'**; **import** { message } **from 'antd'**;  *// 保存user的标识key* **const** USER\_KEY = **'user\_key'**;  *// 保存本地存储内容* **export function** *setStorage*(data) {  **if** (data && **typeof** data !== **'function'**) {  store.set(USER\_KEY, data);  } **else** {  message.error(**'该类型数据不能用于本地存储'**);  } }  *// 读取本地存储内容* **export function** *getStorage*() {  **return** store.*get*(USER\_KEY) || **''**; }  *// 删除本地存储内容* **export function** *removeStorage*() {  store.*remove*(USER\_KEY); } |

### 2.9.3. login/index.jsx

|  |
| --- |
| *login* = (e) => {  *// 禁止表单的默认行为* e.preventDefault();  *// 检验表单是否验证通过。 通过才能发请求* **this**.**props**.**form**.validateFields(**async** (error, values) => {  **if** (!error) {  *// 发送请求，请求登录* **const** { username, password } = values;  *// 请求登录* **const** result = **await** *reqLogin*(username, password);  *// 判断登录是否成功* **if** (result.**status** === 0) {  *// 登录成功* message.success(**'登录成功~'**, 1);  *// 保存用户信息* **const** data = result.**data**;  *// 内存储存：用于多次操作使用，优化（因为读取本地数据更慢）* memory.**user** = data;  *// 本地存储：用于刷新浏览器后使用  setStorage*(data);  *// 跳转到admin页面* **this**.**props**.history.replace(**'/'**);  } **else** {  *// 登录失败* message.error(result.msg);  }  } **else** {  *// 表单校验错误，提示错误* message.error(**'请正确填写用户名和密码'**, 2);  }  }) } |

## 2.10. Admin组件(登陆检查)

### **2.10.1. utils/memory-utils.js**

|  |
| --- |
| */\*\*  \* 用于内存中存储用户信息的模块  \*/* **export default** {  **user**: {} } |

### **2.10.2. login/login.jsx**

|  |
| --- |
| *login* = (e) => {  *// 禁止表单的默认行为* e.preventDefault();  *// 检验表单是否验证通过。 通过才能发请求* **this**.**props**.**form**.validateFields(**async** (error, values) => {  **if** (!error) {  *// 发送请求，请求登录* **const** { username, password } = values;  *// 请求登录* **const** result = **await** *reqLogin*(username, password);  *// 判断登录是否成功* **if** (result.**status** === 0) {  *// 登录成功* message.success(**'登录成功~'**, 1);  *// 保存用户信息* **const** data = result.**data**;  *// 内存储存：用于多次操作使用，优化（因为读取本地数据更慢）* memory.**user** = data;  *// 本地存储：用于刷新浏览器后使用  setStorage*(data);  *// 跳转到admin页面* **this**.**props**.history.replace(**'/'**);  } **else** {  *// 登录失败* message.error(result.msg);  }  } **else** {  *// 表单校验错误，提示错误* message.error(**'请正确填写用户名和密码'**, 2);  }  })   } |

### **2.10.3. admin/index.jsx**

|  |
| --- |
| constructor(props) {  **super**(props);  *// 登录校验* **const** userData = *getStorage*();  **if** (!userData) {  **this**.**props**.history.replace(**'/login'**);  } **else** {  memory.**user** = userData;  } } |

## 2.11. Admin组件(搭建整体结构)

### **2.11.1. 整体组件组成**



### **2.11.2. LeftNav组件**

1. **left-nav.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react'  import './left-nav.less'** */\* 左侧导航组件  \*/* **export default class** LeftNav **extends** Component {  render() {  **return** (  <**div className='left-nav'**>  LeftNav  </**div**>  )  } } |

### **2.11.3. Header组件**

1. **header.less**

|  |
| --- |
| .**header** {  **height**: 100**px**; } |

1. **header.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react'  import './header.less'** */\* 头部组件  \*/* **export default class** Header **extends** Component {  render() {  **return** (  <**div className='header'**>  Header  </**div**>  )  } } |

### **2.11.4. Footer组件**

1. **footer.less**

|  |
| --- |
| .**footer** {  **height**: 100**px**;  **padding**: 40**px** 0;   **text-align**: **center**;  **color**: **#cccccc**; } |

1. **footer.jsx**

|  |
| --- |
| **import** React **from 'react'  import './footer.less'** */\* 底部组件  \*/* **export default function** Footer(props) {  **return** (  <**div className="footer"**>  推荐使用谷歌浏览器，可以获得更佳页面操作体验  </**div**>  ) } |

### **2.11.5. Admin组件**

1. **admin.less**

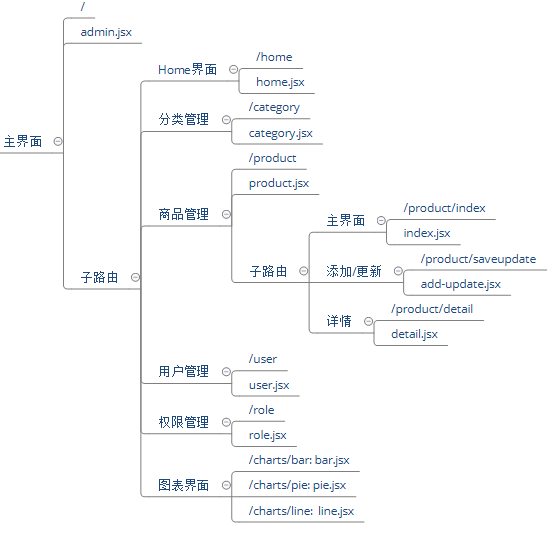
|  |
| --- |
| .**container** {  .**main** {  **height**: **calc**(100**vh**);  **background-color**: **#f1f3f5**;  **overflow**: **auto**;  }  .**content** {  **padding**: 20**px**;  **background**: **white**;  **height**: 300**px**;  } } |

1. **admin.jsx**

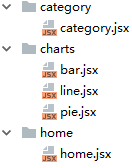
|  |
| --- |
| **import** React, { *Component* } **from 'react'**; **import** { Route, Link } **from 'react-router-dom'**; **import** {  Layout, Breadcrumb } **from 'antd'**;  **import** LeftNav **from '../../components/left-nav'**; **import** { *getStorage* } **from '../../utils/storage-utils'**; **import** memory **from '../../utils/memory-utils'**;  **import** logo **from '../../assets/images/logo.png'**; **import './index.less'**;  **const** {  Header, Content, Footer, Sider, } = Layout;  **export default class** Admin **extends** *Component* {  constructor(props) {  **super**(props);  *// 登录校验* **const** userData = *getStorage*();  **if** (!userData) {  **this**.**props**.history.replace(**'/login'**);  } **else** {  memory.**user** = userData;  *// 初始化状态* **this**.**state** = {  **collapsed**: **false**,  **opacity**: 1  };  }  }    *onCollapse* = (collapsed) => {  **this**.setState({ collapsed, **opacity**: 1 - **this**.**state**.**opacity** });  }    render() {  **const** { opacity, collapsed } = **this**.**state**;    **return** (  <**Layout style=**{{ **minHeight**: **'100vh'** }}>  <**Sider  collapsible  collapsed=**{collapsed}  **onCollapse=**{**this**.*onCollapse*}  >  <**Link to="/home" className="logo"**>  <**img src=**{logo} **alt="logo"**/>  <**h1 style=**{{opacity}}>硅谷后台</**h1**>  </**Link**>  <**LeftNav** />  </**Sider**>  <**Layout**>  <**Header style=**{{ **background**: **'#fff'**, **padding**: 0 }} />  <**Content style=**{{ **margin**: **'0 16px'** }}>  <**Breadcrumb style=**{{ **margin**: **'16px 0'** }}>  <**Breadcrumb.Item**>User</**Breadcrumb.Item**>  <**Breadcrumb.Item**>Bill</**Breadcrumb.Item**>  </**Breadcrumb**>  <**div style=**{{ **padding**: 24, **background**: **'#fff'**, **minHeight**: 360 }}>  Bill is a cat.  </**div**>  </**Content**>  <**Footer style=**{{ **textAlign**: **'center'** }}>  Ant Design ©2018 Created by Ant UED  </**Footer**>  </**Layout**>  </**Layout**>  );  } } |

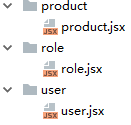
## 2.12. Admin的子路由

### **2.12.1. 组成**



### **2.12.2. 定义各个子路由组件**





### **2.12.3. 注册路由: admin.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {Row, Col} **from 'antd' import** {  **Switch**,  **Route**,  **Redirect** } **from 'react-router-dom'  import** MemoryUtils **from '../../utils/MemoryUtils' import** LeftNav **from '../../components/left-nav/left-nav' import** Header **from '../../components/header/header' import** *Footer* **from '../../components/footer/footer' import** Home **from '../home/home' import** Category **from '../category/category' import** Product **from '../product/product' import** User **from '../user/user' import** Role **from '../role/role' import** Bar **from '../charts/bar' import** Line **from '../charts/line' import** Pie **from '../charts/pie'  import './admin.less'** */\* 后台管理主路由组件  \*/* **export default class** Admin **extends** Component {   render() {  *// 检查是否已经登陆, 如果没有登陆, 自动跳转到登陆页面* **if** (!MemoryUtils.*user*.**\_id**) {  **return** <**Redirect to='/login'**></**Redirect**>  }   **return** (  <**Row className="container"**>  <**Col span=**{4}>  <**LeftNav**/>  </**Col**>  <**Col span=**{20} **className="main"**>  <**Header**/>  <**div className="content"**>  <**Switch**>  <**Route path='/home' component=**{Home}/>  <**Route path='/category' component=**{Category}/>  <**Route path='/product' component=**{Product}/>  <**Route path='/user' component=**{User}/>  <**Route path='/role' component=**{Role}/>  <**Route path="/charts/bar" component=**{Bar}/>  <**Route path="/charts/pie" component=**{Pie}/>  <**Route path="/charts/line" component=**{Line}/>  <**Redirect to='/home'**/>  </**Switch**>  </**div**>  <**Footer**/>  </**Col**>  </**Row**>  )  } } |

## 2.13. LeftNav组件

### **2.13.1. 导航菜单配置: config/menuConfig.js**

|  |
| --- |
| **const** menuList = [  {  **title**: **'首页'**, *// 菜单标题名称* **key**: **'/home'**, *// 对应的path* **icon**: **'home'**, *// 图标名称* },  {  **title**: **'商品'**,  **key**: **'/products'**,  **icon**: **'appstore'**,  **children**: [ *// 子菜单列表* {  **title**: **'品类管理'**,  **key**: **'/category'**,  **icon**: **'bars'** },  {  **title**: **'商品管理'**,  **key**: **'/product'**,  **icon**: **'tool'** },  ]  },   {  **title**: **'用户管理'**,  **key**: **'/user'**,  **icon**: **'user'** },  {  **title**: **'权限管理'**,  **key**: **'/role'**,  **icon**: **'safety'**,  },  {  **title**: **'图形图表'**,  **key**: **'/charts'**,  **icon**: **'area-chart'**,  **children**: [  {  **title**: **'柱形图'**,  **key**: **'/charts/bar'**,  **icon**: **'bar-chart'** },  {  **title**: **'折线图'**,  **key**: **'/charts/line'**,  **icon**: **'line-chart'** },  {  **title**: **'饼图'**,  **key**: **'/charts/pie'**,  **icon**: **'pie-chart'** },  ]  }, ]  **export default** menuList |

### **2.13.2. left-nav.less**

|  |
| --- |
| .**left-nav** {  **height**: **calc**(100**vh**);  **background-color**: **#001529**;  **color**: **#ffffff**;  .**logo** {  **line-height**: 80**px**;  **padding-left**: 20**px**;  **background-color**: **#002140**;  **img** {  **height**: 35**px**;  }  **h1** {  **color**: **#ffffff**;  **font-size**: 20**px**;  **display**: **inline-block**;  **vertical-align**: **middle**;  **margin**: 0 0 0 10**px**;  }  } } |

### **2.13.3. 导航菜单组件: left-nav.jsx**

|  |
| --- |
| **import** React **from 'react' import** {Menu, Icon} **from 'antd' import** {**NavLink**, **withRouter**} **from 'react-router-dom'  import** menuList **from '../../config/menuConfig' import** logo **from '../../assets/images/logo.png'  import './left-nav.less'  const** SubMenu = Menu.*SubMenu  /\* 左侧导航组件  \*/* **class** LeftNav **extends** React.Component {   */\*  得到当前用户需要显示的所有menu元素的列表  使用递归调用  \*/  getMenuNodes* = (menus) => {  **return** menus.reduce((pre, item) => {  **if** (item.**children**) {  **const** subMenu = (  <**SubMenu key=**{item.**key**}  **title=**{<**span**><**Icon type=**{item.**icon**}/><**span**>{item.**title**}</**span**></**span**>}>  {  **this**.*getMenuNodes*(item.**children**)  }  </**SubMenu**>  )  pre.push(subMenu)  } **else** {  **const** menuItem = (  <**Menu.Item key=**{item.**key**}>  <**NavLink to=**{item.**key**}>  <**Icon type=**{item.**icon**}/>{item.**title**}  </**NavLink**>  </**Menu.Item**>  )  pre.push(menuItem)  }  **return** pre  }, [])  }   componentWillMount() {  **this**.**menuNodes** = **this**.*getMenuNodes*(menuList)  }   render() {   *// 得到请求路径* **const** path = **this**.**props**.**location**.**pathname   return** (  <**div className='left-nav'**>  <**NavLink to="/home"**>  <**div className="logo"**>  <**img src=**{logo} **alt="logo"**/>  <**h1**>硅谷后台</**h1**>  </**div**>  </**NavLink**>   <**Menu  theme="dark"  mode="inline"  defaultSelectedKeys=**{[path]}  >  {**this**.**menuNodes**}  </**Menu**>  </**div**>  )  } }  **export default withRouter**(LeftNav) |

## 2.14. Header组件

### **2.14.1. 下载依赖**

|  |
| --- |
| **yarn add jsonp** |

### **2.14.2. api/index.js**

|  |
| --- |
| **import** jsonp **from 'jsonp'**  */\* 通过jsonp请求获取天气信息  \*/* **export function** *reqWeather*(city) {  **const** url = **`http://api.map.baidu.com/telematics/v3/weather?location=**${city}**&output=json&ak=3p49MVra6urFRGOT9s8UBWr2`  return new** Promise((resolve, reject) => {  jsonp(url, {  **param**: **'callback'** }, (error, response) => {  **if** (!error && response.**status** == **'success'**) {  **const** {dayPictureUrl, weather} = response.**results**[0].weather\_data[0]  resolve({dayPictureUrl, weather})  } **else** {  alert(**'获取天气信息失败'**)  }  })  }) } |

### **2.14.3. header/header.less**

|  |
| --- |
| .**header** {  **background-color**: **#ffffff**;  .**header-top** {  **height**: 60**px**;  **line-height**: 60**px**;  **padding**: 0 20**px**;  **text-align**: **right**;  .**logo** {  **line-height**: 60**px**;  **text-align**: **left**;  **font-size**: 18**px**;  **img** {  **height**: 40**px**;  **vertical-align**: **middle**;  }  **span** {  **margin-left**: 5**px**;  }  }  **a** {  **margin-left**: 40**px**;  }  }  .**breadcrumb** {  **height**: 40**px**;  **line-height**: 40**px**;  **padding**: 0 20**px**;  **border-top**: 1**px solid #1DA57A**;  .**breadcrumb-title** {  **text-align**: **center**;  **font-size**: 18**px**;  **&**:**after** {  **position**: **absolute**;  **content**: **''**;  **left**: 73**px**;  **top**: 39**px**;  **border-top**: 9**px solid #ffffff**;  **border-left**: 12**px solid transparent**;  **border-right**: 12**px solid transparent**;  }  }  .**weather** {  **text-align**: **right**;  **font-size**: 14**px**;  .**date** {  **margin-right**: 10**px**;  **vertical-align**: **middle**;  }  .**weather-img** {  **img** {  **height**: 15**px**;  }  }  .**weather-detail** {  **margin-left**: 5**px**;  **vertical-align**: **middle**;  }  }  } } |

### **2.14.4. utils/utils.js**

|  |
| --- |
| */\* 包含n个工具函数的模块  \*/  /\*  格式化日期 \*/* **export function** *formateDate*(time) {  **if** (!time) **return ''  let** date = **new** Date(time)  **return** date.getFullYear() + **'-'** + (date.getMonth() + 1) + **'-'** + date.getDate() + **' '** + date.getHours() + **':'** + date.getMinutes() + **':'** + date.getSeconds() } |

### **2.14.5. header/header.jsx**

|  |
| --- |
| **import** React **from 'react' import** {Row, Col, Modal} **from "antd" import** {**withRouter**} **from 'react-router-dom'  import** MemoryUtils **from '../../utils/MemoryUtils' import** storageUtils **from '../../utils/storageUtils' import** {formateDate} **from '../../utils/utils' import** {*reqWeather*} **from '../../api' import** menuList **from '../../config/menuConfig'  import './header.less'** */\* 头部组件  \*/* **class** Header **extends** React.Component {   **state** = {  **sysTime**: formateDate(**new** Date().getTime()), *// 当前时间* **dayPictureUrl**: **''**, *// 天气图片url* **weather**: **''** *// 天气文本* }   */\*  获取指定地址的天气信息  \*/  getWeather* = **async** () => {  **let** city = **'北京'  const** {dayPictureUrl, weather} = **await** *reqWeather*(city)  **this**.setState({  dayPictureUrl,  weather  })  }   */\*  登出  \*/  logout* = () => {  Modal.*confirm*({  **content**: **'确定退出登陆吗?'**,  onOk: () => {  MemoryUtils.*user* = {}  storageUtils.removeUser()  **this**.**props**.**history**.replace(**'/login'**)  }  })  }   */\*  每隔1s更新当前时间显示  \*/  updateTime* = () => {  **this**.**intervalId** = setInterval(() => {  **const** sysTime = formateDate(**new** Date().getTime())  **this**.setState({  sysTime  })  }, 1000)  }   */\*  根据请求的路由路径得到菜单名称  \*/* getMenuName(path) {  **let** menuName  **for** (**let** i = 0; i < menuList.**length**; i++) {  **let** menu = menuList[i]  **if** (menu.**key** == path) {  menuName = menu.**title  break** } **else if** (menu.**children**) {  **for** (**let** j = 0; j < menu.**children**.**length**; j++) {  **const** menu2 = menu.**children**[j]  **if** (path.indexOf(menu2.**key**) === 0) {  menuName = menu2.**title  break** }  }  **if** (menuName) {  **break** }  }  }  **return** menuName || **'首页'** }   componentDidMount() {  *// 异步获取天气* **this**.*getWeather*()  *// 更新时间* **this**.*updateTime*()  }   componentWillUnmount() {  clearInterval(**this**.**intervalId**)  }   render() {  **const** {**sysTime**, **dayPictureUrl**, **weather**} = **this**.**state  const** username = MemoryUtils.*user*.username  **const** path = **this**.**props**.**location**.**pathname  const** menuName = **this**.getMenuName(path)   **return** (  <**div className="header"**>  <**Row className="header-top"**>  <**Col span=**{24}>  <**span**>欢迎，{username}</**span**>  <**a href="javascript:" onClick=**{**this**.*logout*}>退出</**a**>  </**Col**>  </**Row**>  <**Row className="breadcrumb"**>  <**Col span=**{4} **className="breadcrumb-title"**>  {menuName}  </**Col**>  <**Col span=**{20} **className="weather"**>  <**span className="date"**>{**sysTime**}</**span**>  <**span className="weather-img"**>  <**img src=**{**dayPictureUrl**} **alt="weather"**/>  </**span**>  <**span className="weather-detail"**>  {**weather**}  </**span**>  </**Col**>  </**Row**>  </**div**>  )  } }  **export default withRouter**(Header) |

## 2.15. Home界面

### **2.15.1. home.less**

|  |
| --- |
| .**home-wrap**{  **background-color**: **#ffffff**;  **height**: **calc**(62**vh**);  **display**: **flex**;  **align-items**: **center**;  **justify-content**: **center**;  **font-size**: 30**px**; } |

### **2.15.2. home.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import './home.less'** */\* Home路由组件  \*/* **export default function** Home(props) {  **return** (  <**div className="home-wrap"**>  欢迎使用硅谷后台管理系统  </**div**>  ) } |

## 2.16. 分类管理

### **2.16.1. api/index.js**

|  |
| --- |
| *// 获取一级或某个二级分类列表* **export const** *reqCategorys* = (parentId = **'0'**) => *ajax*(**'/manage/category/list'**, {parentId})  *// 添加分类* **export const** *reqAddCategory* = (parentId, categoryName) => *ajax*(**'/manage/category/add'**, {  parentId,  categoryName }, **'POST'**)  *// 更新品类名称* **export const** *reqUpdateCategory* = (categoryId, categoryName) => *ajax*(**'/manage/category/update'**, {  categoryId,  categoryName }, **'POST'**) |

### **2.16.2. category.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** PropTypes **from 'prop-types' import** {  Button,  Table,  Icon,  Modal,  Input,  Form,  Select,  Card,  message } **from 'antd' import** {  *reqCategorys*,  *reqUpdateCategory*,  *reqAddCategory* } **from '../../api'  const** Option = Select.***Option*** */\* 分类管理的路由组件  \*/* **export default class** Category **extends** Component {   **state** = {  **parentId**: 0, *// 父级分类的id* **parentName**: **''**, *// 父级分类的名称* **categorys**: [], *// 所有一级分类的数组* **subCategorys**: [], *// 需要显示的二级分类的数组* **isShowUpdate**: **false**, *// 是否显示用于更新分类的对话框* **isShowAdd**: **false**, *// 是否显示用于添加的对话框* }   */\*  显示添加界面  \*/  showAdd* = () => {  **this**.setState({  **isShowAdd**: **true** })  }   */\*  添加分类  \*/  addCategory* = **async** () => {  **const** {parentId, categoryName} = **this**.**form**.getFieldsValue()  *// console.log('data', data)* **this**.setState({**isShowAdd**: **false**})  **this**.**form**.resetFields()   **const** result = **await** *reqAddCategory*(parentId, categoryName)  **if** (result.**status** === 0) {  message.success(**'添加分类成功'**)  **if** (parentId === **this**.**state**.**parentId**) {  **this**.*getCategorys*()  }  } **else** {  message.error(**'添加失败: '** + result.msg)  }  }   */\*  根据当前的parentId得到分类(一级/二级)列表并更新状态  \*/  getCategorys* = **async** () => {  **const** parentId = **this**.**state**.**parentId  const** result = **await** *reqCategorys*(parentId)  **if** (result.**status** === 0) {  **const** categorys = result.**data  if** (parentId === 0) {  **this**.setState({  categorys  })  } **else** {  **this**.setState({  **subCategorys**: categorys  })  }  }  }   */\*  显示指定分类的子分类列表  \*/  showSubCategorys* = **async** (category) => {  *// 更新父分类id和名称* **this**.setState({  **parentId**: category.**\_id**,  **parentName**: category.**name** }, () => {  **this**.*getCategorys*()  })  }    */\*  显示更新分类的框  \*/  showUpdateCategory* = (category) => {  **this**.**category** = category  **this**.setState({  **isShowUpdate**: **true** })  }   */\*  更新分类  \*/  updateCategory* = **async** () => {  **const** {category} = **this  const** categoryName = **this**.**form**.getFieldValue(**'categoryName'**)  **this**.setState({  **isShowUpdate**: **false**,  })  **const** result = **await** *reqUpdateCategory*(category.**\_id**, categoryName)  **if** (result.**status** === 0) {  **this**.*getCategorys*()  }  }   */\*  初始化表单的字段数组(只用执行一次)  \*/  initColumns* = () => {  **this**.**columns** = [  {  **title**: **'品类名称'**,  **dataIndex**: **'name'** },  {  **title**: **'操作'**,  **key**: **'operation'**,  **width**: 300,  render: (category) => {  **return** (  <**span**>  <**a href="javascript:;" onClick=**{() => **this**.*showUpdateCategory*(category)}>修改名称</**a**>**&nbsp;&nbsp;** {  **this**.**state**.**parentId** === 0  ? <**a href="javascript:;" onClick=**{() => **this**.*showSubCategorys*(category)}>查看其子品类</**a**>  : **null** }  </**span**>  )  }  },  ]  }   */\*  显示一级分类列表  \*/  showFirstCategorys* = () => {  **this**.setState({  **parentId**: 0,  **parentName**: **''**,  **subCategorys**: [],  })  }   componentWillMount() {  **this**.*initColumns*()  }   componentDidMount() {  **this**.*getCategorys*()  }   render() {   **const** {**parentId**, **parentName**, **categorys**, **subCategorys**, **isShowUpdate**, **isShowAdd**} = **this**.**state  const** category = **this**.**category** || {}   **const** header = (  <**Card style=**{{**fontSize**: 18}}>  {  **parentId** === 0 ? (  <**span**>一级分类列表</**span**>  ) : (  <**span**>  <**a href="javascript:" onClick=**{**this**.*showFirstCategorys*}>一级分类</**a**>  <**Icon type='arrow-right' style=**{{**marginLeft**: 10, **marginRight**: 10}}/>  {**parentName**}  </**span**>  )  }  <**Button type="primary"  onClick=**{**this**.*showAdd*}  **style=**{{**float**: **'right'**}}>  <**Icon type="plus"**/>添加品类  </**Button**>  </**Card**>  )   **return** (  <**div**>  {header}  <**Table  rowKey='\_id'  columns=**{**this**.**columns**}  **dataSource=**{**parentId** === 0 ? **categorys** : **subCategorys**}  **loading=**{**categorys**.**length** === 0}  **bordered  pagination=**{{**defaultPageSize**: 10, **showQuickJumper**: **true**, **showSizeChanger**: **true**}}  />   <**Modal visible=**{**isShowUpdate**}  **closable=**{**false**}  **title='更新分类'  width=**{300}  **onCancel=**{() => **this**.setState({**isShowUpdate**: **false**})}  **onOk=**{**this**.*updateCategory*}  >  <**UpdateForm categoryName=**{category.**name**} **setForm=**{form => **this**.**form** = form}/>  </**Modal**>   <**Modal visible=**{**isShowAdd**}  **title='添加分类'  onCancel=**{() => {  **this**.setState({**isShowAdd**: **false**})  **this**.**form**.resetFields()  }}  **onOk=**{**this**.*addCategory*}  >  <**AddForm categorys=**{**categorys**} **setForm=**{form => **this**.**form** = form}/>  </**Modal**>  </**div**>  )  } }   */\* 更新分类的Form组件  \*/* **class** UpdateForm **extends** Component {   **static** *propTypes* = {  **categoryName**: PropTypes.string.isRequired,  **setForm**: PropTypes.func.isRequired  }   componentWillMount() {  **const** form = **this**.**props**.**form  this**.**props**.**setForm**(form)  }   render() {  **const** {**categoryName**} = **this**.**props  const** {getFieldDecorator} = **this**.**props**.**form  return** (  <**Form**>  <**Form.Item**>  {  getFieldDecorator(**'categoryName'**, {  **initialValue**: **categoryName**,  })(<**Input**/>)  }  </**Form.Item**>  </**Form**>  )  } }  UpdateForm = Form.*create*()(UpdateForm)   */\* 添加分类的Form组件  \*/* **class** AddForm **extends** Component {   **static** *propTypes* = {  **categorys**: PropTypes.array.isRequired,  **setForm**: PropTypes.func.isRequired  }   componentWillMount() {  **const** form = **this**.**props**.**form  this**.**props**.**setForm**(form)  }   render() {  **const** {getFieldDecorator} = **this**.**props**.**form  const** {**categorys**} = **this**.**props  const** options = **categorys**.map(c => (  <**Option key=**{c.**\_id**} **value=**{c.**\_id**}>{c.**name**}</**Option**>  ))  **return** (  <**Form**>  <**Form.Item label="所属分类"**>  {  getFieldDecorator(**'parentId'**, {  **initialValue**: **"0"** })(  <**Select**>  <**Option value="0"**>一级分类</**Option**>  {options}  </**Select**>  )  }  </**Form.Item**>   <**Form.Item label="分类名称"**>  {  getFieldDecorator(**'categoryName'**, {  **initialValue**: **''** })(  <**Input placeholder='输入分类名称'**/>  )  }  </**Form.Item**>  </**Form**>  )  } }  AddForm = Form.*create*({})(AddForm) |

## 2.17. 商品管理

### **2.17.1. 下载依赖**

|  |
| --- |
| yarn add react-draft-wysiwyg draftjs-to-html |

### **2.17.2. api/index.js: 接口请求函数**

|  |
| --- |
| *// 根据分类ID获取分类* **export const** *reqCategory* = (categoryId) => *ajax*(**'/manage/category/info'**, {categoryId})  *// 获取商品分页列表* **export const** *reqProducts* = (pageNum, pageSize) => *ajax*(**'/manage/product/list'**, {pageNum, pageSize})  *// 根据ID/Name搜索产品分页列表* **export const** *reqSearchProducts* = ({pageNum, pageSize, searchType, searchWord}) => *ajax*(**'/manage/product/search'**, {  pageNum,  pageSize,  [searchType]: searchWord, })  *// 添加/更新商品* **export const** *reqAddOrUpdateProduct* = (product) => *ajax*(**'/manage/product/'** + (product.**\_id** ? **'update'** : **'add'**), product, **'post'**)  *// 对商品进行上架/下架处理* **export const** *reqUpOrDownProduct* = (productId, status) => *ajax*(**'/manage/product/updateStatus'**, {  productId,  status }, **'POST'**)  *// 删除图片* **export const** *deleteImg* = (name) => *ajax*(**'/manage/img/delete'**, {name}, **'post'**) |

### **2.17.3. product/index.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {  Table,  Icon,  Input,  Select,  Button,  Card,  message } **from 'antd'  import** {  *reqProducts*,  *reqUpOrDownProduct*,  *reqSearchProducts*, } **from '../../api'  const** Option = Select.***Option*** */\* 商品管理的主二级路由组件  \*/* **export default class** ProductIndex **extends** Component {   **state** = {  **products**: [], *// 当前页的商品列表* **total**: 0, *// 后台商品的总数量* **searchType**: **'productName'**, *// 搜索的类型名称* **searchWord**: **''**, *// 搜索的关键字* }   */\*  异步获取指定页的商品列表  \*/  getProducts* = **async** (index) => {   **const** {**searchType**, **searchWord**} = **this**.**state  let** result  **if** (**searchWord**) {  result = **await** *reqSearchProducts*({**pageNum**: index, **pageSize**: 20, **searchType**, **searchWord**})  } **else** {  result = **await** *reqProducts*(index, 20)  }   **if** (result.**status** === 0) {  **const** {list, total} = result.**data  this**.setState({  **products**: list,  **total**: total  })  message.success(**'获取列表成功!'**)  } **else** {  message.error(**'获取失败: '** + result.**msg**)  }  }   */\*  显示指定商品的详情  \*/  showDetail* = (product) => {  **this**.**props**.**history**.push(**'/product/detail'**, product)  }   */\*  显示更新商品的界面  \*/  showUpdate* = (product) => {  **this**.**props**.**history**.push(**'/product/saveupdate'**, product)  }   */\*  显示添加商品的界面(与更新是同个的路由组件)  \*/  showAdd* = () => {  **this**.**props**.**history**.push(**'/product/saveupdate'**)  }   */\*  对指定商品进行上架或下架的异步更新处理  \*/  downOrUpProduct* = **async** (product) => {  **const** productId = product.**\_id  const** status = product.**status** === 1 ? 2 : 1  **const** result = **await** *reqUpOrDownProduct*(productId, status)  **if** (result.**status** === 0) {  product.**status** = status  **this**.setState({  **products**: **this**.**state**.**products** })  message.success(**'更新商品状态成功'**)  } **else** {  message.error(**'操作失败: '** + result.**msg**)  }  }   */\*  根据输入关键字搜索匹配的商品列表  \*/  search* = () => {  **const** searchWord = **this**.**state**.**searchWord**.trim()  **if** (searchWord) {  **this**.*getProducts*(1)  }  }   */\*  初始化表格的字段数组  \*/  initColumns* = () => {  **this**.**columns** = [  {  **title**: **'商品名称'**,  **dataIndex**: **'name'** },  {  **title**: **'商品描述'**,  **dataIndex**: **'desc'** },  {  **title**: **'价格'**,  **dataIndex**: **'price'**,  render: (price) => **'¥'** + price  },  {  **title**: **'状态'**,  **dataIndex**: **'status'**,  **width**: 200,  render: (status, product) => {  ***console***.log(**'render'**, status, product)  **let** statusText = **'在售'  let** opText = **'下架'  if** (status !== 1) {  statusText = **'已下架'** opText = **'上架'** }  **return** (  <**span**>  <**Button type='primary' onClick=**{() => **this**.*downOrUpProduct*(product)}>{opText}</**Button**>  **&nbsp;&nbsp;&nbsp;&nbsp;**<**span**>{statusText}</**span**>  </**span**>  )  }  },  {  **title**: **'操作'**,  **key**: **'operation'**,  **width**: 150,  render: (product) => {  **return** (  <**span**>  <**a href="javascript:;" onClick=**{() => **this**.*showDetail*(product)}>详情</**a**>**&nbsp;&nbsp;** <**a href="javascript:;" onClick=**{() => **this**.*showUpdate*(product)}>修改</**a**>  </**span**>  )  }  },  ]  }   componentWillMount() {  **this**.*initColumns*()  }   componentDidMount() {  **this**.*getProducts*(1)  }   render() {  **const** {**products**, **total**, **searchType**, **searchWord**} = **this**.**state   const** header = (  <**Card**>  <**Select value=**{**searchType**} **onSelect=**{val => **this**.setState({**searchType**: val})}>  <**Option value='productName'**>根据商品名称</**Option**>  <**Option value='productDesc'**>根据商品描述</**Option**>  </**Select**>  **&nbsp;&nbsp;** <**Input placeholder='关键字'  style=**{{**width**: 200}}  **value=**{**searchWord**}  **onChange=**{e => **this**.setState({**searchWord**: e.**target**.**value**})}/>  **&nbsp;&nbsp;** <**Button type='primary' onClick=**{**this**.*search*}>搜索</**Button**>  <**Button type="primary" onClick=**{**this**.*showAdd*} **style=**{{**float**: **'right'**}}>  <**Icon type="plus"**/>添加产品  </**Button**>  </**Card**>  )   **return** (  <**div**>  {header}   <**Table  rowKey='\_id'  columns=**{**this**.**columns**}  **dataSource=**{**products**}  **bordered  pagination=**{{  **defaultPageSize**: 20,  **total**,  **showQuickJumper**: **true**,  **onChange**: **this**.*getProducts* }}  />  </**div**>  )  } } |

### **2.17.4. product/add-update.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {Form, Input, Select, Button, Icon, message} **from 'antd'  import** PicturesWall **from './pictures-wall' import** RichTextEditor **from './rich-text-editor'  import** {*reqCategorys*, *reqAddOrUpdateProduct*} **from '../../api'  const** Option = Select.***Option*** */\* 商品添加/更新的二级路由组件  \*/* **class** ProductAddUpdate **extends** Component {   **state** = {  **categorys**: [], *// 一级分类列表* **subCategorys**: [], *// 某个二级分类的列表* **parentId**: **''**, *// 父级分类的ID* }   */\*  获取指定分类的子分类列表, 如果parentId为0获取一级分类列表  \*/  getCategorys* = **async** (parentId = **'0'**) => {  **const** result = **await** *reqCategorys*(parentId)  **if** (result.**status** === 0) {  **const** categorys = result.**data  if** (parentId === **'0'**) {  **this**.setState({  parentId,  categorys,  **subCategorys**: []  })  } **else** {  **this**.setState({  parentId,  **subCategorys**: categorys  })  }  }  }   */\*  处理一级列表项改变事件, 获取对应的子分类列表  \*/  onSelectChange* = value => {  **this**.*getCategorys*(value)  }   */\*  添加/更新商品  \*/  submit* = **async** (event) => {  *// 阻止事件的默认行为* event.preventDefault()   *// 收集用户输入的数据, 并封装为商品对象* **const** {name, desc, price, categoryId1, categoryId2} = **this**.**props**.**form**.getFieldsValue()  **const** detail = **this**.**refs**.**editor**.*getContent*()  **const** imgs = **this**.**refs**.**imgs**.*getImages*()  **const** newProduct = {name, desc, price, detail, imgs}  **if** (!categoryId2 || categoryId2 === **'未选择'**) {  newProduct.**categoryId** = categoryId1  newProduct.**pCategoryId** = **'0'** } **else** {  newProduct.**categoryId** = categoryId2  newProduct.**pCategoryId** = categoryId1  }   *// 如果是更新, 指定id* **const** product = **this**.**props**.**location**.**state  if** (product && product.**\_id**) {  newProduct.**\_id** = product.**\_id** }   *// 异步请求后台* **const** result = **await** *reqAddOrUpdateProduct*(newProduct)  *// 如果成功了, 返回到商品主界面显示新的列表* **if** (result.**status** === 0) {  **this**.**props**.**history**.**goBack**()  } **else** {  message.error(result.**msg** || **'操作失败'**)  }  }   componentDidMount() {  *// 获取一级列表显示* **this**.*getCategorys*()   *// 如果是更新, 且商品是二级分类下的, 获取相应的二级分类列表* **const** product = **this**.**props**.**location**.**state  if** (product && product.**pCategoryId** !== **'0'**) {  **this**.*getCategorys*(product.**pCategoryId**)  }  }   render() {  **const** {**categorys**, **subCategorys**, **parentId**} = **this**.**state  const** product = **this**.**props**.**location**.**state** || {}  **const** imgs = product.**imgs   let** options, subOptions  **if** (**categorys**.**length** > 0) {  options = **categorys**.map(category => (<**Option key=**{category.**\_id**} **value=**{category.**\_id**}>{category.**name**}</**Option**>))  }  **if** (**subCategorys**.**length** > 0) {  subOptions = **subCategorys**.map(category => (  <**Option key=**{category.**\_id**} **value=**{category.**\_id**}>{category.**name**}</**Option**>))  }   **let** initCategoryId1 = **'未选择'  if** (**categorys**.**length** > 0) {  **if** (product.**pCategoryId** === **'0'**) {  initCategoryId1 = product.**categoryId** } **else if** (product.**pCategoryId**) {  initCategoryId1 = product.**pCategoryId** }  }   **let** initCategoryId2 = **'未选择'  if** (**subCategorys**.**length** > 0) {  initCategoryId2 = product.**categoryId** || **'未选择'** }   **const** {getFieldDecorator} = **this**.**props**.**form   const** formItemLayout = {  **labelCol**: {  **sm**: {**span**: 2},  },  **wrapperCol**: {  **sm**: {**span**: 12},  },  }    **return** (  <**div style=**{{**paddingLeft**: 30}}>  <**h1**>  <**Icon type="arrow-left" onClick=**{() => **this**.**props**.**history**.**goBack**()}/>**&nbsp;&nbsp;** {product.**\_id** ? **'编辑商品'** : **'添加商品'**}  </**h1**>  <**Form**>  <**Form.Item label='商品名称'** {...formItemLayout}>  {  getFieldDecorator(**'name'**, {  **initialValue**: product.**name** })(  <**Input placeholder='请输入商品名称'**/>  )  }  </**Form.Item**>   <**Form.Item label='商品描述'** {...formItemLayout}>  {  getFieldDecorator(**'desc'**, {  **initialValue**: product.**desc** })(  <**Input placeholder='请输入商品描述'**/>  )  }  </**Form.Item**>   <**Form.Item label='所属分类'** {...formItemLayout}>  {  getFieldDecorator(**'categoryId1'**, {  **initialValue**: initCategoryId1  })(  <**Select style=**{{**width**: 200}} **onChange=**{**this**.*onSelectChange*}>  {options}  </**Select**>  )  }  **&nbsp;&nbsp;** {  **subCategorys**.**length** > 0 ? getFieldDecorator(**'categoryId2'**, {  **initialValue**: initCategoryId2  })(  <**Select style=**{{**width**: 200}}>  {subOptions}  </**Select**>  ) : **null** }  </**Form.Item**>   <**Form.Item label='商品价格'** {...formItemLayout}>  {  getFieldDecorator(**'price'**, {  **initialValue**: product.**price** })(  <**Input  type='number'  style=**{{**width**: 200}}  **placeholder='请输入商品价格'  addonAfter='元'** />  )  }  </**Form.Item**>   <**Form.Item label='商品图片'** {...formItemLayout}>  <**PicturesWall ref='imgs' imgs=**{imgs}/>  </**Form.Item**>   <**Form.Item label='商品详情' labelCol=**{{**sm**: {**span**: 2}}} **wrapperCol=**{{**sm**: {**span**: 19}}}>  <**RichTextEditor ref='editor' html=**{product.**detail**}/>  </**Form.Item**>   <**Form.Item**>  <**Button  type="primary"  htmlType="submit"  style=**{{**width**: 100}}  **onClick=**{**this**.*submit*}  >  提交  </**Button**>  </**Form.Item**>  </**Form**>   </**div**>  )  } }  **export default** Form.*create*({})(ProductAddUpdate) |

### **2.17.5. product/detail.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {List, Icon} **from 'antd'  import** {*reqCategory*} **from '../../api' import** {BASE\_IMG\_PATH} **from '../../utils/constant'** */\* 商品详情组件  \*/* **export default class** ProductDetail **extends** Component {   **state** = {  **cName1**: **''**,  **cName2**: **''** }   */\*  异步获取当前产品对应的分类名称  \*/  getCategoryName* = **async** () => {  **const** {categoryId, pCategoryId} = **this**.**props**.**location**.**state  let** cName1, cName2  **if** (pCategoryId === **'0'**) {  **const** result = **await** *reqCategory*(categoryId)  cName1 = result.**data**.**name** } **else** {   */\*  const result1 = await reqCategory(pCategoryId)  const result2 = await reqCategory(categoryId)  cName1 = result1.data.name  cName2 = result2.data.name  \*/* **const** results = **await** Promise.all([*reqCategory*(pCategoryId), *reqCategory*(categoryId)])  ***console***.log(**'results'**, results)  cName1 = results[0].**data**.**name** cName2 = results[1].**data**.**name** }  ***console***.log(cName1, cName2)  **this**.setState({  cName1,  cName2  })  }   componentDidMount() {  **this**.*getCategoryName*()  }   render() {  **const** {name, desc, price, categoryId, pCategoryId, imgs, detail} = **this**.**props**.**location**.**state  const** {**cName1**, **cName2**} = **this**.**state  return** (  <**div className='product-detail'**>  <**h1**>  <**Icon type="arrow-left" onClick=**{() => **this**.**props**.**history**.**goBack**()}/>**&nbsp;&nbsp;** 商品详情  </**h1**>   <**List**>  <**List.Item**>  <**span className='left'**>商品名称:</**span**>  <**span**>{name}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品描述:</**span**>  <**span**>{desc}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品价格:</**span**>  <**span**>{price + **'元'**}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>所属分类:</**span**>  <**span**>{**cName1** + **'-->'** + **cName2**}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品图片:</**span**>  <**span**>  {  imgs.map(img => (  <**img src=**{BASE\_IMG\_PATH + img} **alt="img" key=**{img}  **style=**{{**width**: 150, **height**: 150, **marginRight**: 10}}/>  ))  }  </**span**>  </**List.Item**>   <**List.Item**>  <**span className='left'**>商品详情:</**span**>  <**div dangerouslySetInnerHTML=**{{**\_\_html**: detail}}></**div**>  </**List.Item**>  </**List**>  </**div**>  )  } } |

### **2.17.6. product/pictures-wall.jsx**

|  |
| --- |
| **import** React **from 'react' import** PropTypes **from 'prop-types' import** {Upload, Icon, Modal, message} **from 'antd'  import** {*deleteImg*} **from '../../api' import** {BASE\_IMG\_PATH} **from '../../utils/constant'** */\* 图片墙(上传)组件  \*/* **export default class** PicturesWall **extends** React.Component {   **static** *propTypes* = {  **images**: PropTypes.array  }   **state** = {  **previewVisible**: **false**, *// 是否显示图片预览框* **previewImage**: **''**, *// 预览图片的url* **fileList**: [], *// 包含所有图片文件信息的数组(显示)* }   *// 给父组件调用的用来获取所有图片文件名数组  getImages* = () => {  **return this**.**state**.**fileList**.map(file => file.**name**)  }   *// 取消查看图片大图  handleCancel* = () => **this**.setState({**previewVisible**: **false**})   *// 响应点击预览查看图片大图  handlePreview* = (file) => {  **this**.setState({  **previewImage**: file.**url** || file.**thumbUrl**,  **previewVisible**: **true**,  });  }   */\*  上传文件状态发生改变的回调  文件3个状态: loading, down, removed  \*/  handleChange* = **async** ({file, fileList, event}) => {  **const** status = file.**status  if** (status == **'done'**) {  **const** result = file.**response  if** (result.**status** === 0) {  **const** data = result.**data  const** url = data.**url  const** name = data.**name** file = fileList[fileList.**length** - 1]  file.**url** = url  file.**name** = name  ***console***.log(**'handleChange'**, file.**percent**, file.**status**, file)  } **else** {  message.error(**'上传图片失败'**)  }  } **else if** (status === **'removed'**) {  ***console***.log(**'removed'**, file, fileList.**length**)  **const** name = file.**name  const** result = **await** *deleteImg*(name)  **if** (result.**status** === 0) {  message.success(**'图片删除成功'**)  } **else** {  message.error(**'图片删除失败'**)  }  }   **this**.setState({fileList})  }   componentWillMount() {  **const** images = **this**.**props**.**imgs** || []  **const** fileList = images.map((image, index) => ({  **uid**: -index,  **name**: image,  **status**: **'done'**,  **url**: BASE\_IMG\_PATH + image,  }))  **this**.**state**.**fileList** = fileList  }   render() {  **const** {**previewVisible**, **previewImage**, **fileList**} = **this**.**state**;  **const** uploadButton = (  <**div**>  <**Icon type="plus"**/>  <**div className="ant-upload-text"**>Upload</**div**>  </**div**>  );  **return** (  <**div className="clearfix"**>  <**Upload  action="/manage/img/upload"  accept='image/\*'  listType="picture-card"  name='image'  fileList=**{**fileList**}  **onPreview=**{**this**.*handlePreview*}  **onChange=**{**this**.*handleChange*}  **onRemove=**{**this**.handleRemove}  >  {**fileList**.**length** >= 4 ? **null** : uploadButton}  </**Upload**>   <**Modal visible=**{**previewVisible**}  **footer=**{**null**}  **onCancel=**{**this**.*handleCancel*}  >  <**img alt="img" style=**{{**width**: **'100%'**}} **src=**{**previewImage**}/>  </**Modal**>  </**div**>  );  } } |

### **2.17.7. product/rich-text-editor.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** PropTypes **from 'prop-types' import** {EditorState, *convertToRaw*, ContentState} **from 'draft-js' import** {Editor} **from 'react-draft-wysiwyg' import** draftToHtml **from 'draftjs-to-html' import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css' import** htmlToDraft **from 'html-to-draftjs'** */\* 富文本编辑器组件  \*/* **export default class** RichTextEditor **extends** Component {   **static** *propTypes* = {  **html**: PropTypes.string  }   constructor(props) {  **super**(props)  **const** {**html**} = **this**.**props** *// 如果指定的页面文本, 需要初始化编辑状态数据中* **if** (**html**) {  **const** blocksFromHtml = htmlToDraft(**html**)  **const** {contentBlocks, entityMap} = blocksFromHtml  **const** contentState = ContentState.createFromBlockArray(contentBlocks, entityMap)  **const** editorState = EditorState.createWithContent(contentState)  **this**.**state** = {  **editorState**: editorState  }  *// 指定一个空的编辑状态数据* } **else** {  **this**.**state** = {  **editorState**: EditorState.createEmpty()  }  }  }   *// 给父组件调用的方法:用来获取输入的html文本  getContent* = () => {  **return** draftToHtml(*convertToRaw*(**this**.**state**.**editorState**.getCurrentContent()))  }   *// 当输入发生改变时的回调: 更新编辑状态数据  onEditorStateChange* = (editorState) => {  **this**.setState({  editorState,  })  }   render() {  **const** {editorState} = **this**.**state  return** (  <**div**>  <**Editor  editorState=**{editorState}  **wrapperClassName="demo-wrapper"  editorClassName="demo-editor"  onEditorStateChange=**{**this**.*onEditorStateChange*}  />  </**div**>  )  } } |

### **2.17.8. product/detail.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {List, Icon} **from 'antd'  import** {*reqCategory*} **from '../../api' import** {BASE\_IMG\_PATH} **from '../../utils/constant'** */\* 商品详情组件  \*/* **export default class** ProductDetail **extends** Component {   **state** = {  **cName1**: **''**,  **cName2**: **''** }   */\*  异步获取当前产品对应的分类名称  \*/  getCategoryName* = **async** () => {  **const** {categoryId, pCategoryId} = **this**.**props**.**location**.**state  let** cName1, cName2  **if** (pCategoryId === **'0'**) {  **const** result = **await** *reqCategory*(categoryId)  cName1 = result.**data**.**name** } **else** {   */\*  const result1 = await reqCategory(pCategoryId)  const result2 = await reqCategory(categoryId)  cName1 = result1.data.name  cName2 = result2.data.name  \*/* **const** results = **await** Promise.all([*reqCategory*(pCategoryId), *reqCategory*(categoryId)])  ***console***.log(**'results'**, results)  cName1 = results[0].**data**.**name** cName2 = results[1].**data**.**name** }  ***console***.log(cName1, cName2)  **this**.setState({  cName1,  cName2  })  }   componentDidMount() {  **this**.*getCategoryName*()  }   render() {  **const** {name, desc, price, categoryId, pCategoryId, imgs, detail} = **this**.**props**.**location**.**state  const** {**cName1**, **cName2**} = **this**.**state  return** (  <**div className='product-detail'**>  <**h1**>  <**Icon type="arrow-left" onClick=**{() => **this**.**props**.**history**.**goBack**()}/>**&nbsp;&nbsp;** 商品详情  </**h1**>   <**List**>  <**List.Item**>  <**span className='left'**>商品名称:</**span**>  <**span**>{name}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品描述:</**span**>  <**span**>{desc}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品价格:</**span**>  <**span**>{price + **'元'**}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>所属分类:</**span**>  <**span**>{**cName1** + **'-->'** + **cName2**}</**span**>  </**List.Item**>  <**List.Item**>  <**span className='left'**>商品图片:</**span**>  <**span**>  {  imgs.map(img => (  <**img src=**{BASE\_IMG\_PATH + img} **alt="img" key=**{img}  **style=**{{**width**: 150, **height**: 150, **marginRight**: 10}}/>  ))  }  </**span**>  </**List.Item**>   <**List.Item**>  <**span className='left'**>商品详情:</**span**>  <**div dangerouslySetInnerHTML=**{{**\_\_html**: detail}}></**div**>  </**List.Item**>  </**List**>  </**div**>  )  } } |

### **2.17.9. product/index.less**

|  |
| --- |
| .**demo-editor** {  **background**: **#ffffff**;  **min-height**: 200**px**; }  .**product-detail** {  **font-size**: 18**px**;  **background**: **white**;  .**left** {  **width**: 100**px**;  **font-size**: 18**px**;  } } |

### **2.17.10. utils/constant.js**

|  |
| --- |
| *// 图片的基础路径* **export const** BASE\_IMG\_PATH = **'http://localhost:5000/upload/'** |

## 2.18. 权限管理

### **2.18.1. api/index.js: 接口请求函数**

|  |
| --- |
| *// 添加角色* **export const** *reqAddRole* = (roleName) => *ajax*(**'/manage/role/add'**, {  roleName }, **'POST'**)  *// 获取角色列表* **export const** *reqRoles* = () => *ajax*(**'/manage/role/list'**)  *// 更新角色(给角色设置权限)* **export const** *reqUpdateRoles* = (role) => *ajax*(**'/manage/role/update'**, role, **'POST'**) |

### **2.18.2. role.jsx**

|  |
| --- |
| **import** React, {PureComponent} **from 'react' import** {  Card,  Button,  Table,  Form,  Input,  Select,  Tree,  Modal, } **from 'antd'  import** {  *reqRoles*,  *reqAddRole*,  *reqUpdateRoles* } **from '../../api' import** menuConfig **from '../../config/menuConfig' import** MemoryUtils **from '../../utils/MemoryUtils' import** {*formateDate*} **from '../../utils/utils'  const** FormItem = Form.*Item* **const** Option = Select.***Option* const** TreeNode = Tree.*TreeNode   /\* 权限(角色)管理路由组件  \*/* **export default class** Role **extends** PureComponent {   **state** = {  **roles**: [], *// 所有角色的列表* **role**: {}, *// 当前选中的角色* **menus**: [], *// 所有权限标识的数组* **isShowAdd**: **false**, *// 是否显示角色添加的框* **isShowRoleAuth**: **false**, *// 是否显示设置角色权限的框* **isShowUserRole**: **false**, *// 是否显示设置用户角色的框* }   */\*  异步获取角色列表  \*/  getRoles* = **async** () => {  **const** result = **await** *reqRoles*()  **if** (result.**status** === 0) {  **const** roles = result.**data  this**.setState({  roles  })  }  }   */\*  初始化Table的字段数据  \*/  initColumns* = () => {  **this**.**columns** = [  {  **title**: **'角色名称'**,  **dataIndex**: **'name'** },  {  **title**: **'创建时间'**,  **dataIndex**: **'create\_time'**,  **render**: *formateDate* },  {  **title**: **'授权时间'**,  **dataIndex**: **'auth\_time'**,  **render**: *formateDate* },  {  **title**: **'授权人'**,  **dataIndex**: **'auth\_name'**,  }  ]  }   */\*  处理某个角色的Radio选中时, 更新role和menus  \*/  handleSelectChange* = (selectedRowKeys, selectedRows) => {  **console**.log(**'handleSelectChange()'**, selectedRowKeys, selectedRows)  **const** role = selectedRows[0]  **this**.setState({  role,  **menus**: role.**menus** })  }   */\*  点击'创建角色'回调  \*/  showAddRole* = () => {  **this**.setState({  **isShowAdd**: **true** })  }   */\*  点击'设置权限'回调  \*/  showRoleAuth* = () => {  **this**.setState({  **isShowRoleAuth**: **true** })  }   */\*  给table的每行绑定点击监听  \*/  onRow* = (role) => ({  onClick: () => {  **console**.log(**'onClick()'**)  *// 如果点击的是当前选中的, 直接结束  /\*if (this.state.role === role) {  return  }\*/* **this**.setState({  role,  **menus**: role.**menus** })  }  })   */\*  异步添加角色  \*/  addRole* = **async** () => {  **const** roleName = **this**.**addForm**.getFieldValue(**'roleName'**)  **this**.**addForm**.resetFields()  **const** result = **await** *reqAddRole*(roleName)  **const** role = result.**data  const** roles = **this**.**state**.**roles**.slice()  roles.push(role)  **this**.setState({  roles,  **isShowAdd**: **false** })  }   */\*  更新角色  \*/  updateRole* = **async** () => {  **const** {role, **menus**} = **this**.**state** role.**menus** = **menus** role.**auth\_name** = MemoryUtils.*user*.**name** || **'admin'  this**.setState({  role,  **isShowRoleAuth**: **false** })  **const** result = **await** *reqUpdateRoles*(role)  **if** (result.**status** === 0) {  **this**.*getRoles*()  }  }   */\*  更新menus状态  \*/  setRoleMenus* = (menus) => {  **this**.setState({  menus  })  }   componentWillMount() {  **this**.*initColumns*()  }   componentDidMount() {  **this**.*getRoles*()  }   render() {  **console**.log(**'render()'**)  **const** {**roles**, **isShowAdd**, role, **isShowRoleAuth**, **isShowUserRole**, **menus**} = **this**.**state   const** rowSelection = {  **type**: **'radio'**,  **selectedRowKeys**: [role.**\_id**],  **onChange**: **this**.*handleSelectChange* }   **return** (  <**div**>  <**Card**>  <**Button type="primary" onClick=**{**this**.*showAddRole*}>创建角色</**Button**>**&nbsp;&nbsp;** <**Button type="primary" onClick=**{**this**.*showRoleAuth*} **disabled=**{!role.**\_id**}>设置角色权限</**Button**>**&nbsp;&nbsp;** </**Card**>   <**Table  columns=**{**this**.**columns**}  **rowKey='\_id'  dataSource=**{**roles**}  **bordered  rowSelection=**{rowSelection}  **onRow=**{**this**.*onRow*}  **pagination=**{{**defaultPageSize**: 10, **showQuickJumper**: **true**}}  />   <**Modal  title="创建角色"  visible=**{**isShowAdd**}  **onCancel=**{() => {  **this**.setState({**isShowAdd**: **false**})  **this**.**addForm**.resetFields()  }}  **onOk=**{**this**.*addRole*}  >  <**AddRoleForm setAddForm=**{(form) => **this**.**addForm** = form}/>  </**Modal**>    <**Modal  title="设置角色权限"  visible=**{**isShowRoleAuth**}  **onCancel=**{() => **this**.setState({**isShowRoleAuth**: **false**, **menus**: role.**menus**})}  **onOk=**{**this**.*updateRole*}  >  <**RoleAuthForm setAddForm=**{form => **this**.**roleAuthForm** = form}  **role=**{role}  **menus=**{**menus**}  **setRoleMenus=**{**this**.*setRoleMenus*}/>  </**Modal**>  </**div**>  )  } }  */\* 用来添加角色的form组件  \*/* **class** AddRoleForm **extends** PureComponent {   componentWillMount() {  **this**.**props**.setAddForm(**this**.**props**.**form**)  }   render() {  **console**.log(**'AddRoleForm render()'**)  **const** {getFieldDecorator} = **this**.**props**.**form  const** formItemLayout = {  **labelCol**: {**span**: 5},  **wrapperCol**: {**span**: 16}  };   **return** (  <**Form**>  <**FormItem label="角色名称"** {...formItemLayout}>  {  getFieldDecorator(**'roleName'**, {  **initialValue**: **''** })(  <**Input type="text" placeholder="请输入角色名称"**/>  )  }  </**FormItem**>  </**Form**>  )  } }  AddRoleForm = Form.*create*()(AddRoleForm)  */\* 用来设置角色权限的form组件  \*/* **class** RoleAuthForm **extends** PureComponent {   */\*  渲染树状节点列表  \*/  renderTreeNodes* = (data, key = **''**) => {  **return** data.map((item) => {  **let** parentKey = key + item.**key  if** (item.**children**) {  **return** (  <**TreeNode title=**{item.**title**} **key=**{parentKey} **dataRef=**{item}>  {**this**.*renderTreeNodes*(item.**children**, parentKey)}  </**TreeNode**>  )  }  **return** <**TreeNode** {...item} />  })  }   */\*  设置选中的节点，通过父组件方法再传递回来  \*/  onCheck* = (checkedKeys) => {  **this**.**props**.patchMenuInfo(checkedKeys);  }   componentWillMount() {  **this**.**props**.setAddForm(**this**.**props**.**form**)  }   render() {  **console**.log(**'RoleAuthForm render()'**)   **const** {role, menus, setRoleMenus} = **this**.**props   return** (  <**Form layout="horizontal"**>  <**FormItem label="角色名称："**>  <**Input disabled placeholder=**{role.**name**}/>  </**FormItem**>   <**Tree  checkable  defaultExpandAll  checkedKeys=**{menus || []}  **onCheck=**{(checkedKeys) => setRoleMenus(checkedKeys)}  >  <**TreeNode title="平台权限" key="platform\_all"**>  {**this**.*renderTreeNodes*(menuConfig)}  </**TreeNode**>  </**Tree**>  </**Form**>  )  } }  RoleAuthForm = Form.*create*()(RoleAuthForm) |

## 2.19. 用户管理

### **2.19.1. api/index.js: 接口请求函数**

|  |
| --- |
| *// 添加/更新用户* **export const** *reqAddOrUpdateUser* = (user) => *ajax*(**'/manage/user/'**+(user.**\_id** ? **'update'** : **'add'**), user, **'POST'**)  *// 获取用户列表* **export const** *reqUsers* = () => *ajax*(**'/manage/user/list'**)  *// 删除用户* **export const** *reqDeleteUser* = (userId) => *ajax*(**'/manage/user/delete'**, {userId}, **'POST'**) |

### **2.19.2. user.jsx**

|  |
| --- |
| **import** React, {Component} **from 'react' import** {  Card,  Button,  Table,  Form,  Input,  Select,  Modal,  Icon,  message } **from 'antd'  import** {  *reqUsers*,  *reqAddOrUpdateUser*,  *reqDeleteUser* } **from '../../api' import** {*formateDate*} **from '../../utils/utils'  const** FormItem = Form.*Item* **const** Option = Select.***Option*** */\* 用户管理路由组件  \*/* **export default class** User **extends** Component {   **state** = {  **users**: [], *// 用户列表* **roles**: [], *// 角色列表* **isShow**: **false**, *// 是否显示添加/更新用户的Modal* }   */\*  异步获取用户列表显示  \*/  getUsers* = **async** () => {  **const** result = **await** *reqUsers*()  **if** (result.**status** === 0) {  **const** {users, roles} = result.**data  this**.*initRoleMap*(roles)  **this**.setState({  users,  roles  })  }  }   */\*  初始化产生一个保存角色名的容器对象, 属性名为角色的id  \*/  initRoleMap* = (roles) => {  **const** roleMap = roles.reduce((pre, role) => {  pre[role.**\_id**] = role.**name  return** pre  }, {})  **this**.**roleMap** = roleMap  }   */\*  初始化Table的字段列表  \*/  initColumns* = () => {  **this**.**columns** = [  {  **title**: **'用户名'**,  **dataIndex**: **'username'** },  {  **title**: **'邮箱'**,  **dataIndex**: **'email'** },  {  **title**: **'电话'**,  **dataIndex**: **'phone'** },  {  **title**: **'注册时间'**,  **dataIndex**: **'create\_time'**,  **render**: *formateDate* },  {  **title**: **'所属角色'**,  **dataIndex**: **'role\_id'**,  render: (roleId) => {  **return this**.**roleMap** ? **this**.**roleMap**[roleId] : **''** }  },  {  **title**: **'操作'**,  render: (user) => {  **return** (  <**span**>  <**a href="javascript:;" onClick=**{() => **this**.*showUpdateUser*(user)}>修改</**a**>  **&nbsp;&nbsp;** <**a href="javascript:;" onClick=**{() => **this**.*clickDeleteUser*(user)}>删除</**a**>  </**span**>  )  }  },  ]  }   */\*  显示添加用户的Modal  \*/  showAddUser* = () => {  **this**.**user** = **null  this**.setState({  **isShow**: **true** })  }   */\*  显示更新用户的Modal  \*/  showUpdateUser* = (user) => {  **this**.**user** = user  **this**.setState({  **isShow**: **true** })  }   */\*  点击删除用户  \*/  clickDeleteUser* = (user) => {  Modal.*confirm*({  **content**: **`你确定要删除此**${user.username}**吗?`**,  onOk: **async** () => {  **const** result = **await** *reqDeleteUser*(user.**\_id**)  **this**.*getUsers*()  }  })  }   */\*  添加或更新用户  \*/  AddOrUpdateUser* = **async** () => {  **const** formUser = **this**.**form**.getFieldsValue()  **if** (**this**.**user**) {  formUser.**\_id** = **this**.**user**.**\_id** }  **this**.**form**.resetFields()  **this**.setState({  **isShow**: **false** })  **const** result = **await** *reqAddOrUpdateUser*(formUser)  **if** (result.**status** === 0) {  **this**.*getUsers*()  } **else** {  message.error(result.msg)  }  }    componentWillMount() {  **this**.*initColumns*()  }   componentDidMount() {  **this**.*getUsers*()  }   render() {  **console**.log(**'render()'**)  **const** {**users**, **roles**, **isShow**} = **this**.**state  const** {user} = **this  const** isAdd = user ? **false** : **true   return** (   <**div**>  <**Card**>  <**Button type="primary" onClick=**{**this**.*showAddUser*}>创建用户</**Button**>  </**Card**>   <**Table  columns=**{**this**.**columns**}  **rowKey='\_id'  dataSource=**{**users**}  **bordered  pagination=**{{**defaultPageSize**: 10, **showQuickJumper**: **true**}}  />   <**Modal  title=**{isAdd ? **'添加用户'** : **'更新用户'**}  **visible=**{**isShow**}  **onCancel=**{() => **this**.setState({**isShow**: **false**})}  **onOk=**{**this**.*AddOrUpdateUser*}  >  <**UserForm setForm=**{(form) => **this**.**form** = form} **user=**{user} **roles=**{**roles**}/>  </**Modal**>   </**div**>  )  } }  */\* 用来添加或更新的form组件  \*/* **class** UserForm **extends** Component {   */\*  初始化<Option/>的列表  \*/  initOptions* = () => {  **const** roles = **this**.**props**.roles  **return** roles.map(role => (<**Option key=**{role.**\_id**} **value=**{role.**\_id**}>{role.**name**}</**Option**>))  }   componentWillMount() {  **this**.**props**.**setForm**(**this**.**props**.**form**)  }   render() {  **const** user = **this**.**props**.**user** || {}  **const** {getFieldDecorator} = **this**.**props**.**form  const** formItemLayout = {  **labelCol**: {**span**: 5},  **wrapperCol**: {**span**: 16}  }   **const** options = **this**.*initOptions*()   **return** (  <**Form**>  <**FormItem label="用户名"** {...formItemLayout}>  {  getFieldDecorator(**'username'**, {  **initialValue**: user.username  })(  <**Input type="text" placeholder="请输入用户名"**/>  )  }  </**FormItem**>   {  !user.**\_id** ? (  <**FormItem label="密码"** {...formItemLayout}>  {  getFieldDecorator(**'password'**, {  **initialValue**: **''** })(  <**Input type="passowrd" placeholder="请输入密码"**/>  )  }  </**FormItem**>  )  : **null** }    <**FormItem label="手机号"** {...formItemLayout}>  {  getFieldDecorator(**'phone'**, {  **initialValue**: user.**phone** })(  <**Input type="phone" placeholder="请输入手机号"**/>  )  }  </**FormItem**>   <**FormItem label="邮箱"** {...formItemLayout}>  {  getFieldDecorator(**'email'**, {  **initialValue**: user.**email** })(  <**Input type="email" placeholder="请输入邮箱"**/>  )  }  </**FormItem**>   <**FormItem label="角色"** {...formItemLayout}>  {  getFieldDecorator(**'role\_id'**, {  **initialValue**: user.role\_id || **'未选择'** })(  <**Select style=**{{**width**: 200}}>  {options}  </**Select**>  )  }  </**FormItem**>  </**Form**>  )  } }  UserForm = Form.*create*()(UserForm) |

### **2.19.3. 导航权限控制**

1. **left-nav.jsx**

|  |
| --- |
| */\* 判断当前用户是否有指定菜单的权限  \*/ hasAuth* = (menu) => {  **const** userMenuKeys = **this**.**userMenuKeys  const** isAdmin = MemoryUtils.*user*.username === **'admin'  return** isAdmin || menu.**isPublic** || userMenuKeys.has(menu.**key**) }  */\* 得到当前用户需要显示的所有menu元素的列表 使用递归调用  \*/ getMenuNodes* = (menus) => {  **return** menus.reduce((pre, item) => {   **if (this.*hasAuth*(item)) {**  **if** (item.**children**) {  **const** subMenu = (  <**SubMenu key=**{item.**key**}  **title=**{<**span**><**Icon type=**{item.**icon**}/><**span**>{item.**title**}</**span**></**span**>}>  {  **this**.*getMenuNodes*(item.**children**)  }  </**SubMenu**>  )  pre.push(subMenu)  } **else** {  **const** menuItem = (  <**Menu.Item title=**{item.**title**} **key=**{item.**key**}>  <**NavLink to=**{item.**key**}>  <**Icon type=**{item.**icon**}/>{item.**title**}  </**NavLink**>  </**Menu.Item**>  )  pre.push(menuItem)  }  }  **return** pre  }, []) } |

1. **config/menuConfig.js**

|  |
| --- |
| {  **title**: **'首页'**, *// 菜单标题名称* **key**: **'/home'**, *// 对应的path* **icon**: **'home'**, *// 图标名称* isPublic: true, *// 开放的* } |

## 2.20. 可视化图表

### **2.20.1. 下载依赖**

|  |
| --- |
| **yarn add echarts echarts-for-react** |

### **2.20.2. bar.jsx: 柱状图**

|  |
| --- |
| **import** React **from 'react' import** {Card} **from 'antd' import** ReactEcharts **from 'echarts-for-react'** *// 引入饼图和折线图* **import 'echarts/lib/chart/bar' import 'echarts/lib/component/tooltip' import 'echarts/lib/component/title' import 'echarts/lib/component/legend' import 'echarts/lib/component/markPoint'** */\* 显示柱状图的路由组件  \*/* **export default class** Bar **extends** React.Component {   **state** = {}   getOption() {  **let** option = {  **title**: {  **text**: **'用户骑行订单22'** },  **tooltip**: {  **trigger**: **'axis'** },  **xAxis**: {  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **yAxis**: {  **type**: **'value'** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'bar'**,  **data**: [  1000,  2000,  1500,  3000,  2000,  1200,  800  ]  }  ]  }  **return** option  }   getOption2() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'** },  **tooltip**: {  **trigger**: **'axis'** },  **legend**: {  **data**: [**'OFO'**, **'摩拜'**, **'小蓝'**]  },  **xAxis**: {  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **yAxis**: {  **type**: **'value'** },  **series**: [  {  **name**: **'OFO'**,  **type**: **'bar'**,  **data**: [  2000,  3000,  5500,  7000,  8000,  12000,  20000  ]  },  {  **name**: **'摩拜'**,  **type**: **'bar'**,  **data**: [  1500,  3000,  4500,  6000,  8000,  10000,  15000  ]  },  {  **name**: **'小蓝'**,  **type**: **'bar'**,  **data**: [  1000,  2000,  2500,  4000,  6000,  7000,  8000  ]  },  ]  }  **return** option  }   render() {  **return** (  <**div**>  <**Card title="柱形图表之一"**>  <**ReactEcharts option=**{**this**.getOption()} **notMerge=**{**true**} **lazyUpdate=**{**true**}  **style=**{{**height**: 500}}/>  </**Card**>  <**Card title="柱形图表之二" style=**{{**marginTop**: 10}}>  <**ReactEcharts option=**{**this**.getOption2()} **notMerge=**{**true**} **lazyUpdate=**{**true**}  **style=**{{**height**: 500}}/>  </**Card**>  </**div**>  )  } } |

### **2.20.3. line.jsx: 折线图**

|  |
| --- |
| **import** React **from 'react' import** {Card} **from 'antd' import** ReactEcharts **from 'echarts-for-react'** *// 引入饼图和折线图* **import 'echarts/lib/chart/line'** *// 引入提示框和标题组件* **import 'echarts/lib/component/tooltip' import 'echarts/lib/component/title' import 'echarts/lib/component/legend' import 'echarts/lib/component/markPoint'** */\* 显示折线图的路由组件  \*/* **export default class** Line **extends** React.Component {   **state** = {}   componentWillMount() {  *// echarts.registerTheme('Imooc', echartTheme)* }   getOption() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'** },  **tooltip**: {  **trigger**: **'axis'** },  **xAxis**: {  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **yAxis**: {  **type**: **'value'** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'line'**,  **data**: [  1000,  2000,  1500,  3000,  2000,  1200,  800  ]  }  ]  }  **return** option  }   getOption2() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'** },  **tooltip**: {  **trigger**: **'axis'** },  **legend**: {  **data**: [**'OFO订单量'**, **'摩拜订单量'**]  },  **xAxis**: {  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **yAxis**: {  **type**: **'value'** },  **series**: [  {  **name**: **'OFO订单量'**,  **type**: **'line'**,  **stack**: **'总量'**,  **data**: [  1200,  3000,  4500,  6000,  8000,  12000,  20000  ]  },  {  **name**: **'摩拜订单量'**,  **type**: **'line'**,  **stack**: **'总量'**,  **data**: [  1000,  2000,  5500,  6000,  8000,  10000,  12000  ]  },  ]  }  **return** option  }   getOption3() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'** },  **tooltip**: {  **trigger**: **'axis'** },  **xAxis**: {  **type**: **'category'**,  **boundaryGap**: **false**,  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **yAxis**: {  **type**: **'value'** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'line'**,  **data**: [  1000,  2000,  1500,  3000,  2000,  1200,  800  ],  **areaStyle**: {}  }  ]  }  **return** option  }   render() {  **return** (  <**div**>  <**Card title="折线图表之一"**>  <**ReactEcharts  option=**{**this**.getOption()}  **theme="Imooc"  notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{  **height**: 500  }}/>  </**Card**>  <**Card title="折线图表之二" style=**{{**marginTop**: 10}}>  <**ReactEcharts  option=**{**this**.getOption2()}  **notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{  **height**: 500  }}/>  </**Card**>  <**Card title="折线图表之三" style=**{{**marginTop**: 10}}>  <**ReactEcharts  option=**{**this**.getOption3()}  **notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{  **height**: 500  }}/>  </**Card**>  </**div**>  )  } } |

### **2.20.4. pie.jsx: 饼状图**

|  |
| --- |
| **import** React **from 'react' import** {Card} **from 'antd' import** ReactEcharts **from 'echarts-for-react' import 'echarts/lib/chart/pie' import 'echarts/lib/component/tooltip' import 'echarts/lib/component/title' import 'echarts/lib/component/legend' import 'echarts/lib/component/markPoint'** */\* 显示饼图的路由组件  \*/* **export default class** Bar **extends** React.Component {   getOption() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'**,  **x**: **'center'** },  **legend**: {  **orient**: **'vertical'**,  **right**: 10,  **top**: 20,  **bottom**: 20,  **data**: [**'周一'**, **'周二'**, **'周三'**, **'周四'**, **'周五'**, **'周六'**, **'周日'**]  },  **tooltip**: {  **trigger**: **'item'**,  **formatter**: **"{a} <br/>{b} : {c} ({d}%)"** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'pie'**,  **radius**: **'55%'**,  **center**: [  **'50%'**, **'60%'** ],  **data**: [  {  **value**: 1000,  **name**: **'周一'** },  {  **value**: 1000,  **name**: **'周二'** },  {  **value**: 2000,  **name**: **'周三'** },  {  **value**: 1500,  **name**: **'周四'** },  {  **value**: 3000,  **name**: **'周五'** },  {  **value**: 2000,  **name**: **'周六'** },  {  **value**: 1200,  **name**: **'周日'** },  ],  **itemStyle**: {  **emphasis**: {  **shadowBlur**: 10,  **shadowOffsetX**: 0,  **shadowColor**: **'rgba(0, 0, 0, 0.5)'** }  }  }  ]  }  **return** option  }   getOption2() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'**,  **x**: **'center'** },  **legend**: {  **orient**: **'vertical'**,  **right**: 10,  **top**: 20,  **bottom**: 20,  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **tooltip**: {  **trigger**: **'item'**,  **formatter**: **"{a} <br/>{b} : {c} ({d}%)"** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'pie'**,  **radius**: [**'50%'**, **'80%'**],  **center**: [  **'50%'**, **'60%'** ],  **data**: [  {  **value**: 1000,  **name**: **'周一'** }, {  **value**: 1000,  **name**: **'周二'** }, {  **value**: 2000,  **name**: **'周三'** }, {  **value**: 1500,  **name**: **'周四'** }, {  **value**: 3000,  **name**: **'周五'** }, {  **value**: 2000,  **name**: **'周六'** }, {  **value**: 1200,  **name**: **'周日'** }  ],  **itemStyle**: {  **emphasis**: {  **shadowBlur**: 10,  **shadowOffsetX**: 0,  **shadowColor**: **'rgba(0, 0, 0, 0.5)'** }  }  }  ]  }  **return** option  }   getOption3() {  **let** option = {  **title**: {  **text**: **'用户骑行订单'**,  **x**: **'center'** },  **legend**: {  **orient**: **'vertical'**,  **right**: 10,  **top**: 20,  **bottom**: 20,  **data**: [  **'周一'**,  **'周二'**,  **'周三'**,  **'周四'**,  **'周五'**,  **'周六'**,  **'周日'** ]  },  **tooltip**: {  **trigger**: **'item'**,  **formatter**: **"{a} <br/>{b} : {c} ({d}%)"** },  **series**: [  {  **name**: **'订单量'**,  **type**: **'pie'**,  **radius**: **'55%'**,  **center**: [  **'50%'**, **'50%'** ],  data: [  {  **value**: 1000,  **name**: **'周一'** }, {  **value**: 1000,  **name**: **'周二'** }, {  **value**: 2000,  **name**: **'周三'** }, {  **value**: 1500,  **name**: **'周四'** }, {  **value**: 3000,  **name**: **'周五'** }, {  **value**: 2000,  **name**: **'周六'** }, {  **value**: 1200,  **name**: **'周日'** }  ].sort(**function** (a, b) {  **return** a.**value** - b.**value** }),  **roseType**: **'radius'**,  **animationType**: **'scale'**,  **animationEasing**: **'elasticOut'**,  animationDelay: **function** (idx) {  **return *Math***.random() \* 200  }  }  ]  }  **return** option  }   render() {  **return** (  <**div**>  <**Card title="饼形图表之一"**>  <**ReactEcharts  option=**{**this**.getOption()}  **notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{**height**: 500}}/>  </**Card**>  <**Card title="饼形图之二" style=**{{**marginTop**: 10}}>  <**ReactEcharts  option=**{**this**.getOption2()}  **notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{**height**: 500}}/>  </**Card**>  <**Card title="饼形图之三" style=**{{**marginTop**: 10}}>  <**ReactEcharts  option=**{**this**.getOption3()}  **notMerge=**{**true**}  **lazyUpdate=**{**true**}  **style=**{{**height**: 500}}/>  </**Card**>  </**div**>  )  } } |