Ctrl+c 停止运行

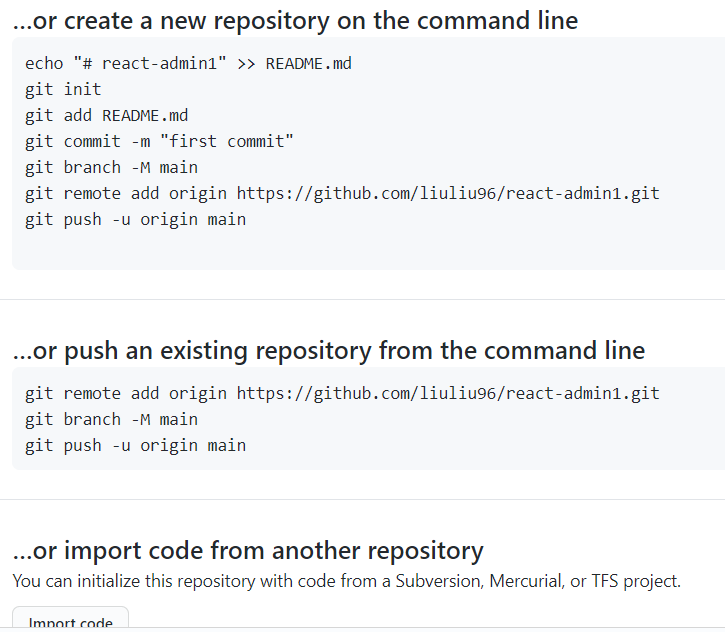
上下键 返回命令

npm （run）eject 显示webpack, babel等隐藏文件

创建git仓库并初始化

git init

git add .之后和下图一样

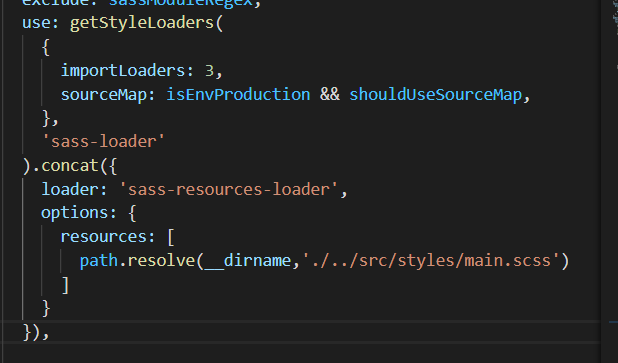


react-router-dom

import {Switch, Route, Router, HashHistory, Link} from "react-router-dom";

npm install sass-loader node-sass --save-dev 安装SaaS依赖

npm install sass-resources-loader –D



.concat({

                loader: 'sass-resources-loader',

                options: {

                  resources: [

                    path.resolve(\_\_dirname,'./../src/styles/main.scss')

                  ]

                }

              }),

antd 按需加载

npm install babel-plugin-import –save-dev

// Process application JS with Babel.

            // The preset includes JSX, Flow, TypeScript, and some ESnext features.

            {

              test: /\.(js|mjs|jsx|ts|tsx)$/,

              include: paths.appSrc,

              loader: require.resolve('babel-loader'),

              options: {

                customize: require.resolve(

                  'babel-preset-react-app/webpack-overrides'

                ),

                plugins: [

                  [

                    require.resolve('babel-plugin-named-asset-import'),

                    {

                      loaderMap: {

                        svg: {

                          ReactComponent:

                            '@svgr/webpack?-svgo,+titleProp,+ref![path]',

                        },

                      },

                    },

                  ],

                  [

                    "import",

                     {libraryName: "antd", style:'css'}

                  ]

                ],

                // This is a feature of `babel-loader` for webpack (not Babel itself).

                // It enables caching results in ./node\_modules/.cache/babel-loader/

                // directory for faster rebuilds.

                cacheDirectory: true,

跨域

1. cnpm install http-proxy-middleware
2. 新建文件夹

在src/目录新建setupProxy.js

const { createProxyMiddleware } = require('http-proxy-middleware');

module.exports=function(app){

    app.use(createProxyMiddleware("/devapi",{

        target:"http://www.web-jshtml.cn/api/react",//配置你要请求的服务器地址

        changeOrigin:true,

    }))

    // app.use(proxy("manage/api",{

    //     target:"http://admintest.happymmall.com:7000",

    //     changeOrigin:true,

    // }))

};

1. 修改路径

修改config/path.js

proxySetup路径

1. 重写路径
2. const { createProxyMiddleware } = require('http-proxy-middleware');
3. module.exports=function(app){
4. app.use(createProxyMiddleware("/devapi",{
5. target:"http://www.web-jshtml.cn/api/react",//配置你要请求的服务器地址
6. changeOrigin:true,
7. pathRewrite:{
8. "^/devapi":"",
9. }
10. }))
11. // app.use(proxy("manage/api",{
12. //     target:"http://admintest.happymmall.com:7000",
13. //     changeOrigin:true,
14. // }))
15. };

拦截器

npm install axios

import axios from "axios";

//第一步创建实例

const service = axios.create({

    baseURL: 'devapi',

    timeout: 5000,

    headers: {'X-Custom-Header': 'foobar'}

  });

  //第二步请求拦截

  service.interceptors.request.use(function (config) {

    // 在发送请求之前做些什么

    return config;

  }, function (error) {

    // 对请求错误做些什么

    return Promise.reject(error);

  });

 //第三步响应拦截

 service.interceptors.response.use(function (response) {

    // 对响应数据做点什么

    return response;

  }, function (error) {

    // 对响应错误做点什么

    return Promise.reject(error);

  });

export default service;

create-react-app 创建的项目有内置的环境变量NODE\_ENV,可通过process.env. NODE\_ENV读取变量

NODE\_ENV 默认三个值

1 开发：development- npm start

2 测试：test- npm run test

3 生产：production- npm run build

项目打包配置环境变量

依赖：npm install –g dotenv-cli

package.json中配置（暂时有问题）

  "scripts": {

    "start": "node scripts/start.js",

    "build": "node scripts/build.js",

    "test": "node scripts/test.js",

    "build:dev":"dotenv -e .env.development react-app-rewried build",

    "build:pro":"dotenv -e .env.production react-app-rewried build",

    "build:test":"dotenv -e .env.test react-app-rewried build"

  },

组件类型：

1. 无状态组件
2. 有状态组件
3. UI组件（不能复用）
4. 容器组件（对数据，业务处理）
5. 高阶组件
6. 回调渲染

密码加密

npm install crypto-js

import Cryptojs from ‘crypto-js’

//md5加密

const pwd= Cryptojs.MD5(password).toString()

//SHA1

let pwd= Cryptojs.SHA1. (password).toString()

白名单

import {withRouter} from ‘react-router-dom’;

this.props.history.push(‘home’);

withRouter

将react-router的history、location、match三个对象传入props对象上

React-cookies

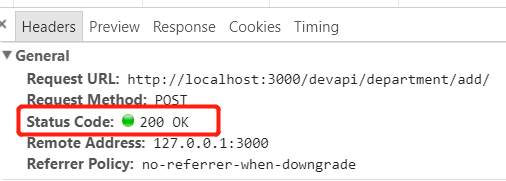
npm install react-cookies --save-dev （--save-dev 本地；--save线上 ）

响应数据业务逻辑

resCode属性的约定

一般都是后台定义好的变量，告知接口请求情况， 如通过该值做一些成功的事情，否则就是失败。约定某一个值为请求成功，定义为0



http响应状态

.then方法里的第二个回调不存在的时候，会调catch， 如果存在第二个回调，就不会执行catch