目录

[React 2](#_Toc32728)

[环境搭建 2](#_Toc4179)

[useState 2](#_Toc16447)

[classnames工具优化类名库 3](#_Toc28559)

[表单受控绑定 4](#_Toc15062)

[获取DOM 5](#_Toc18955)

[组件通讯-父传子props 6](#_Toc4927)

[组件通讯-父传子chilren 7](#_Toc13660)

[组件通讯-子传父 8](#_Toc28493)

[组件通讯-contex机制顶层通讯 9](#_Toc21582)

[useEffect 11](#_Toc20927)

[自定义HOOK 12](#_Toc25924)

[Redux 13](#_Toc5019)

[ReactRouter 17](#_Toc26769)

# React

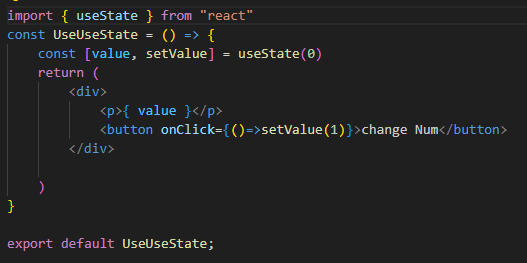
# 环境搭建

Node 环境

$ npx create-react-app react-basic

# useState

从核心react库引用useState

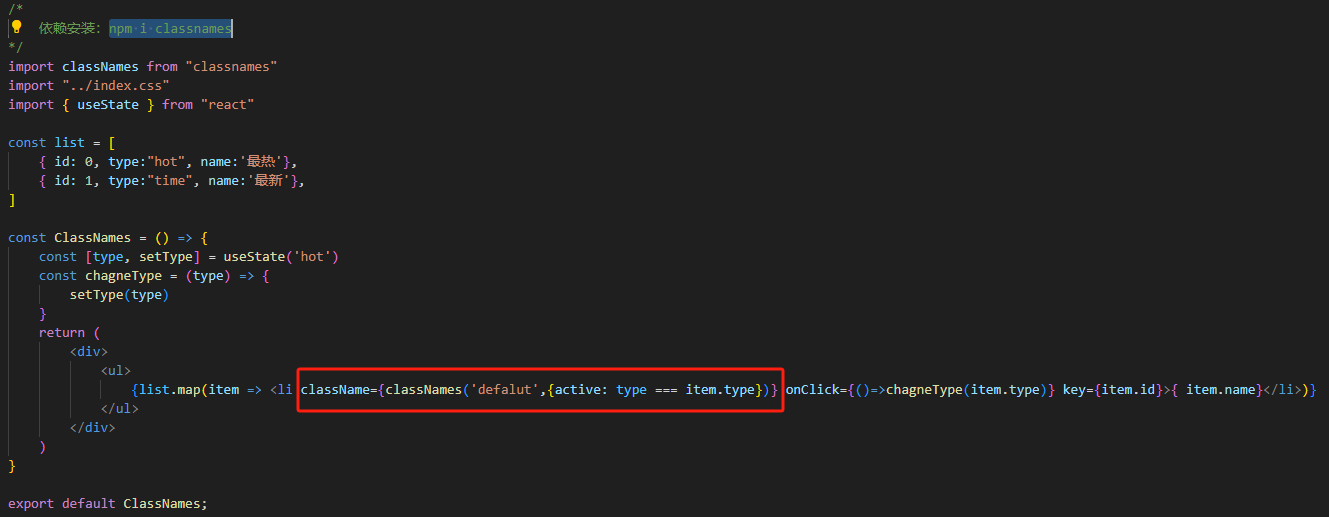


# classnames工具优化类名库

依赖安装： $ npm i classnames

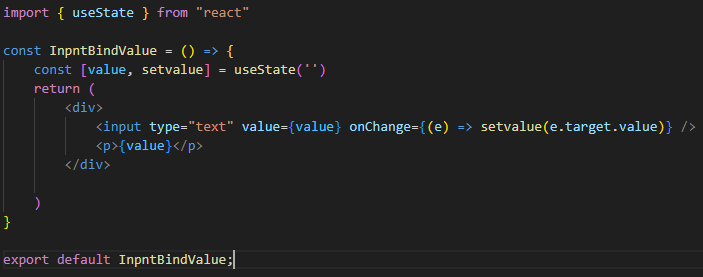
方法一：className={ classNames( 'defalut',{active: type === item.type})}

方法二：className={ classNames( 'defalut', type === item.type && ‘active’)}



# 表单受控绑定

使用useStatue,在input中的onChange方法使用set实现

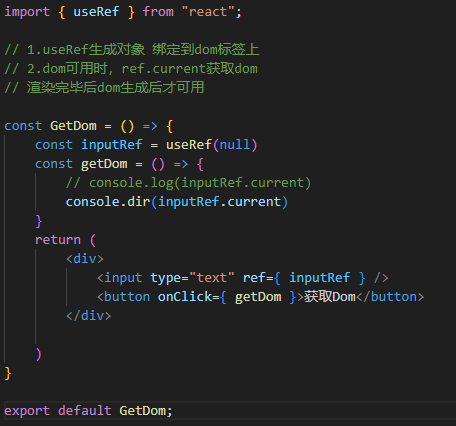


# 获取DOM

1.useRef生成对象 绑定到dom标签上

2.dom可用时，ref.current获取dom

渲染完毕后dom生成后才可用

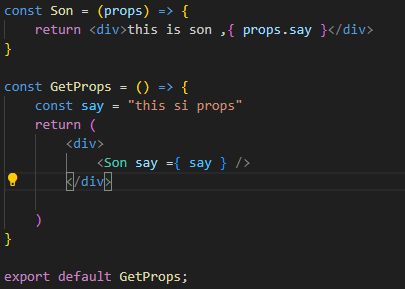


# 组件通讯-父传子props

也可以使用匿名属性，在子组件建构

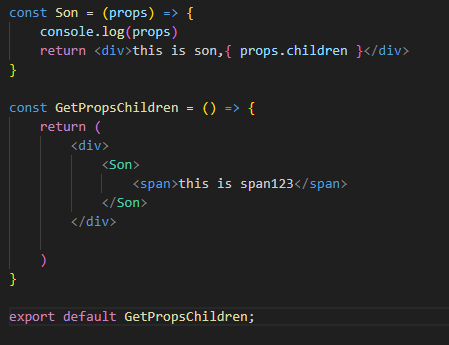
<Son {...item}>

const getProps = ({id,name,count})=>{return ()}



# 组件通讯-父传子chilren

props.chilren获取子组件中的HTML标签的值



# 组件通讯-子传父

自定义属性传值为父组件本身函数，通过子组件入参获取调用



# 组件通讯-contex机制顶层通讯

1. 引用createContext函数实例化

const Msgcontext = createContext()

1. 在顶层组件包裹子组件

<MsgContext.Provider value={msg}>

<A><B></B></A>

</MsgContext.Provider>

3.在通信组件使用hook useContext()获取属性值

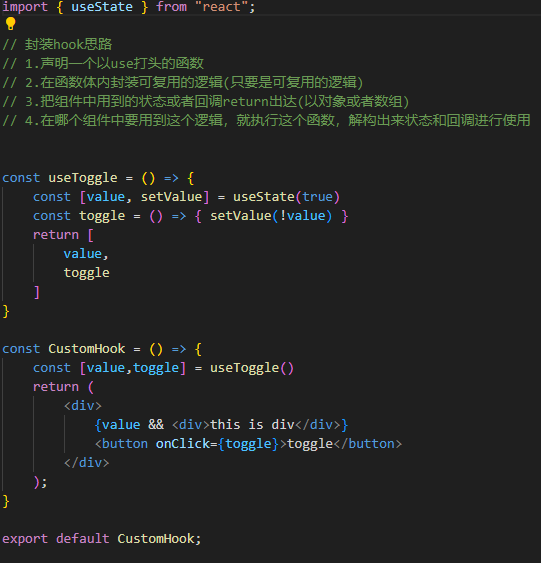


# useEffect

组件渲染完毕后调用



# 自定义HOOK



ReactHooks使用规则:

1. 只能在组件和其他自定义hook函数中使用
2. 只能在组件顶层调用，不能嵌套在if/for/其他函数

# Redux

1. 安装：

$ npm i @reduxjs/toolkit react-redux

调试工具谷歌商店安装devtools

项目index.js编写



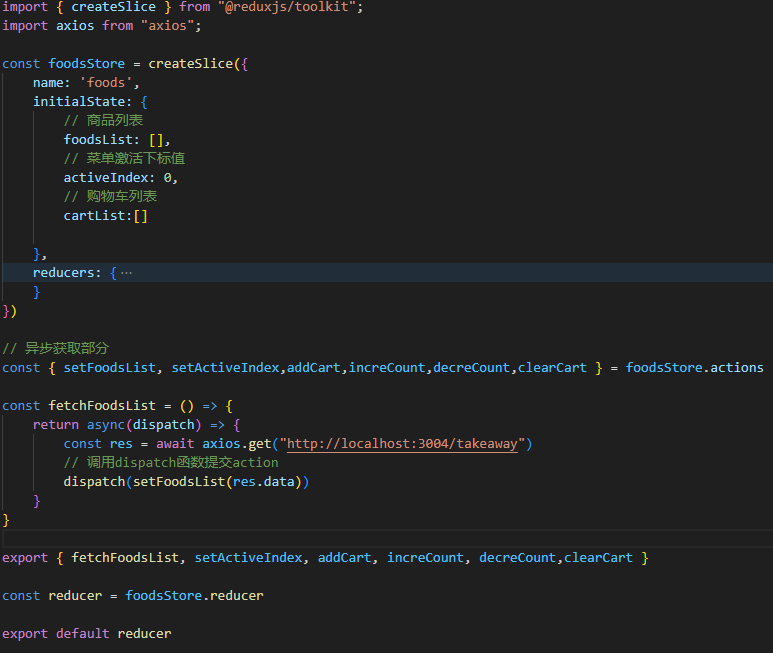
1. Store index.js文件编写



1. 子模块store编写

导出默认名给index.js使用

按需导出action函数给组件使用



4.组件引用



# ReactRoute

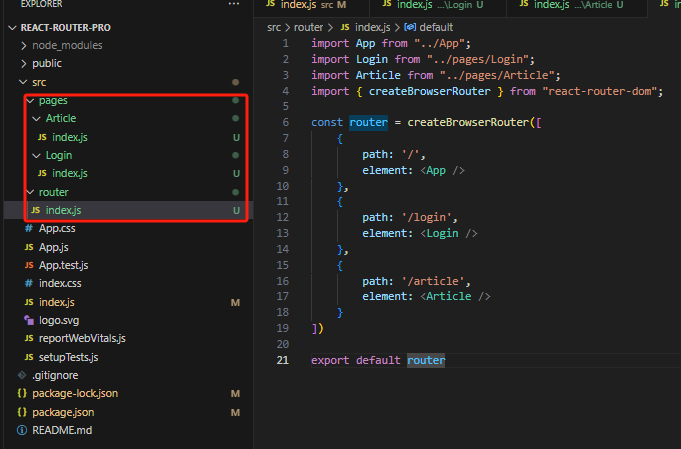
1. 依赖安装

$ npm i react-router-dom

1. index.js配置

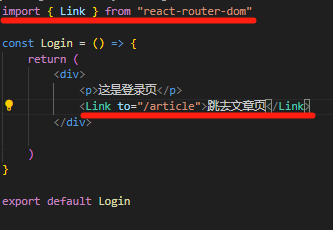


1. 抽象路由模块

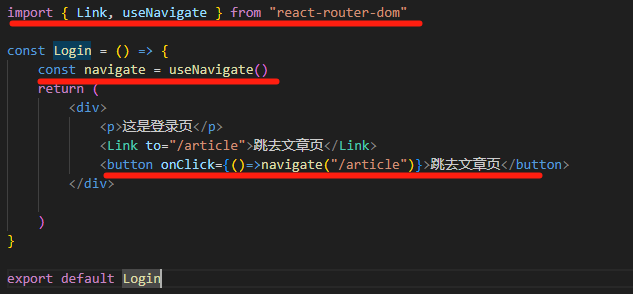


1. 路由导航

声明式导航：获取路由组件导航



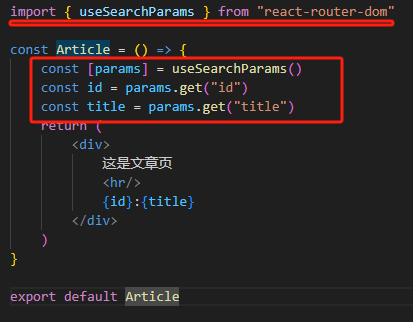
编程式导航：获取useNavigate 钩子导航



1. 导航传参

searchParams传参：

<button onClick={()=>navigate("/article?id=01&title=红楼梦")}>searchParams传参</button>



Params传参：不常用

/article/:id

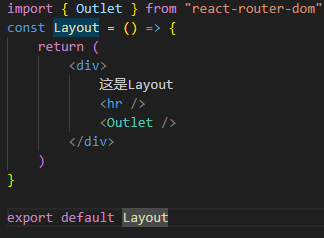
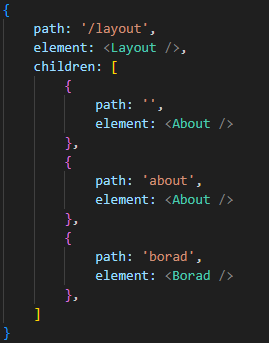
Import {useParams} from “react-router-dom”

const params = useParams() ;let id = params.id

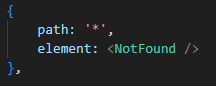
1. 嵌套路由配置

二级路由配置需要从react-router-dom引用<Outlet />组件作为渲染组件

配置：



7、404路由配置



8、路由的两种模式

History: url/login 原理：history对象 + pushState事件

Hash: url/#/login 原理：监听hashChange事件

# 记账本demo

1. 环境依赖

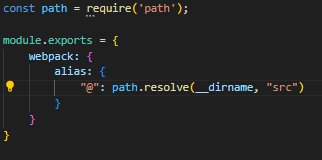
$ npm i @reduxjs/toolkit react-redux react-router-dom dayjs classnames antd-mobile axios

# 配置路径别名

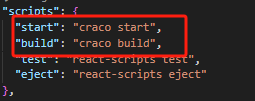
1）路径解析配置：craco插件

npm i -D @craco/craco

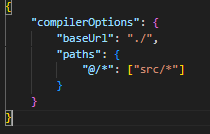
根目录增加craco.config.js



修改根目录package.json文件



2）路径联想配置：vsCode增加jsconfig.json



1. 数据Mock实现

Json-server实现数据Mock

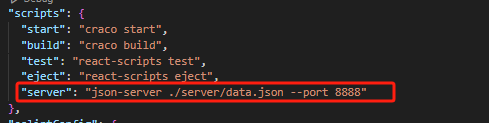
1. 项目安装

$ npm i -D json-server

1. 准备一个json文件

server/data.json

1. 修改根目录package.json添加启动命令



1. 、访问接口进行测试

启动项目：npm run server

1. 路由设计：确定一级路由和二级路由
2. Ant Design Mobile(移动端)主题定制

$ npm install --save antd-mobile

增加theme.css,入口文件index.js导入

1. 全局定制

:root:root{ --adm-color-primary: #a062d4 }

1. 局部定制

.purple\_theme{ --adm-color-primary: #a062d4 }

1. Redux管理账目列表
2. 实现统计区域dayjs库使用
3. Lodash 使用（数据处理依赖）

npm i lodash

1. 功能实现

# 使用CRA

1. 项目搭建

$ npx create-react-app 项目名称

$ npm i sass -D

1. 安装Ant Design（PC端）组件库

$ npm install antd --save

1. 安装路由

$ npm i react-router-dom

1. 配置路径别名

# 封装request请求模块

6、样式reset

$ npm i normalize.css

7、处理token失效，监控后端返回401状态码

账号：13888888888 验证码：246810

1. echarts使用
2. API封装