目录

[npx create-react-app user-redux-thunk 1](#_Toc520411569)

[cd user-redux-thunk 1](#_Toc520411570)

[github.com创建仓库，粘贴代码git init等到命令行下 1](#_Toc520411571)

[index.html中添加bootstrap.css<link>地址 1](#_Toc520411572)

[命令行cnpm install –save redux react-redux redux-logger redux-thunk 1](#_Toc520411573)

[npm start 2](#_Toc520411574)

[Src下创建constants、components、actions、reducers目录 2](#_Toc520411575)

[入口文件src/index.js 2](#_Toc520411576)

[Store=createStore(rootReducer,{},applyMiddleware(logger,thunk)); 3](#_Toc520411577)

[<Provider store={store}><App /></Provider> 4](#_Toc520411578)

[要实现点击按钮，数字+1 4](#_Toc520411579)

[src/App.js 4](#_Toc520411580)

[(1)Ui:包含两个按钮和显示的数据 4](#_Toc520411581)

[(2)onclick={()=>increment()} action create函数 5](#_Toc520411582)

[Actions/index.js 5](#_Toc520411583)

[Reducers/counter,js（返回新state） 6](#_Toc520411584)

[Rootreducer 7](#_Toc520411585)

[src/app.js 7](#_Toc520411586)

[（4）之所以可以用this.props.counter 7](#_Toc520411587)

[（5）之所以可以用onclick={（）=>increment()} 8](#_Toc520411588)

[（6）最后升级App(connect) 8](#_Toc520411589)

[类型检查 8](#_Toc520411590)

## npx create-react-app user-redux-thunk

## cd user-redux-thunk

## github.com创建仓库，粘贴代码git init等到命令行下

## index.html中添加bootstrap.css<link>地址

## 命令行cnpm install –save redux react-redux redux-logger redux-thunk

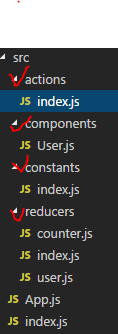
<https://redux.js.org/>

https://github.com/evgenyrodionov/redux-logger

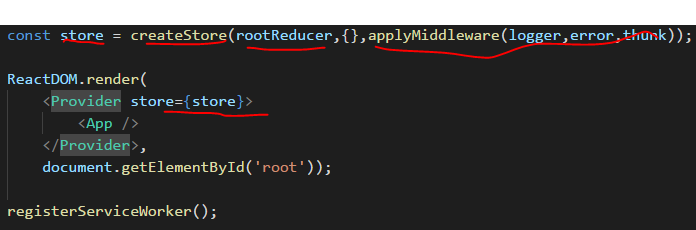
<https://github.com/reduxjs/redux-thunk>

## npm start

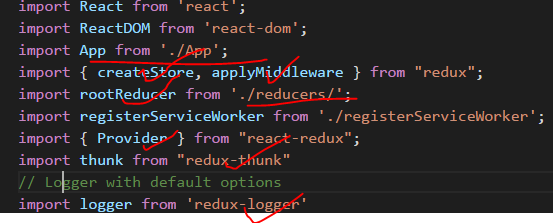
## Src下创建constants、components、actions、reducers目录



## 入口文件src/index.js



依赖



### Store=createStore(rootReducer,{},applyMiddleware(logger,thunk));

**st**ore：存放state的单一数据源,createStore创建，含getState.

State:状态/数据

Action:改变状态的唯一方法，触发action, action是一个对象，必须含type(描述意图)

Action creator:纯函数，

Reducer：（state,action）=> state:描述如何根据action改变state

RootReducer:通过combineReducers({reducer1,reducer2})组合多个reducer,

Provider:根组件，向下传递store

ApplyMiddleware(中间件)

Logger:每次dispatch action都会打印日志https://github.com/evgenyrodionov/redux-logger

Thunk: <https://github.com/reduxjs/redux-thunk>

### <Provider store={store}><App /></Provider>

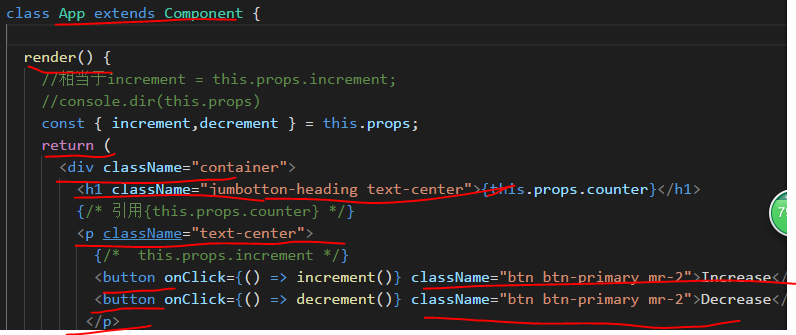
## 要实现点击按钮，数字+1

### src/App.js

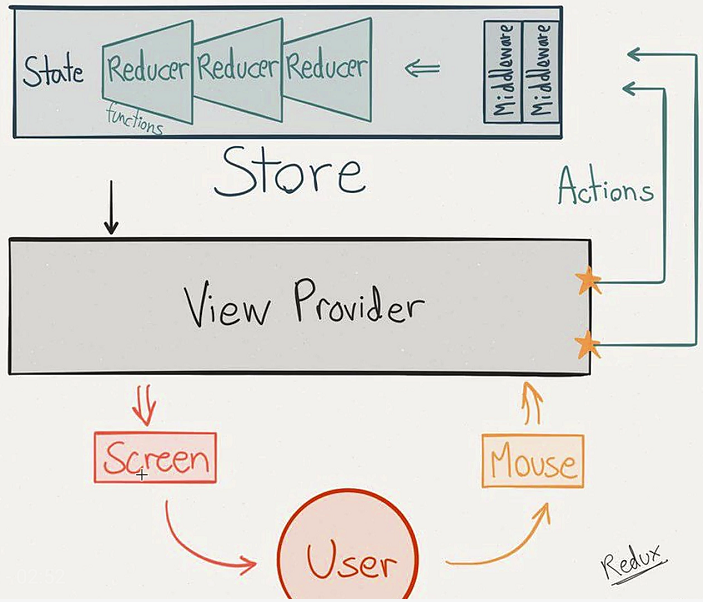
### (1)Ui:包含两个按钮和显示的数据



先实现以下红色划线





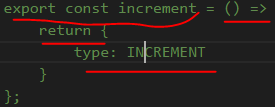


### (2)onclick={()=>increment()} action create函数

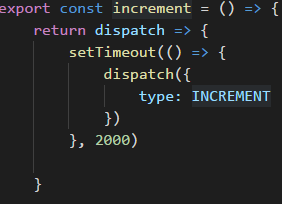


### **Actions/index.js(意图)**





改为dispatch+异步模式，点击后2秒再数字+1



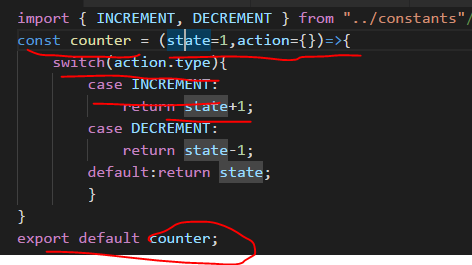
constants/index.js

其中INCREMENT是常量，表示type(action意图)，定义在

### Reducers/counter,js（返回新state）

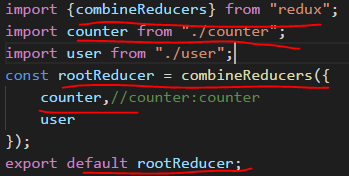




State=1初始值，所以启动项目后浏览器显示1

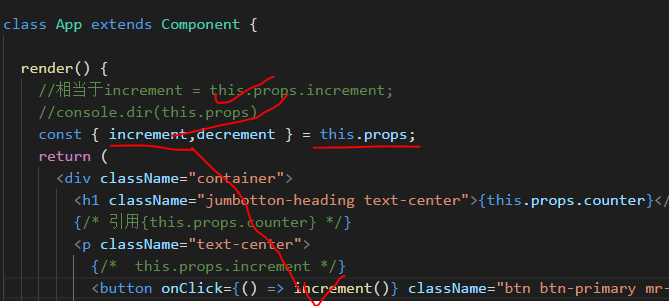


### Rootreducer

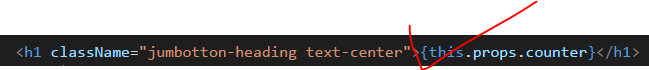


### src/app.js

实现onclick=数字+1方法



### （4）之所以可以用this.props.counter



是因为

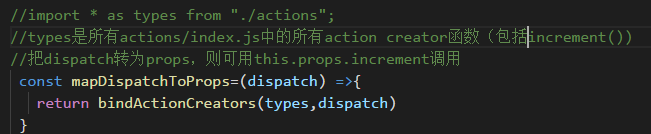


（5）之所以可以用onclick={（）=>increment()}

方法名加（）表示调用

这里的increment是this.props.increment的别称，用const {increment} = this.props定义

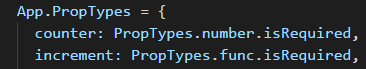
之所以可以用this.props.increment



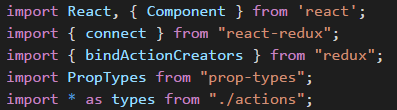
### （6）最后升级App(connect)



### 类型检查

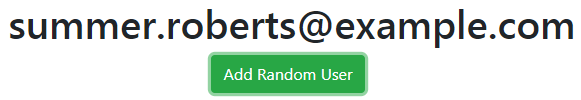


依赖



## 实现点击按钮显示随机用户邮箱

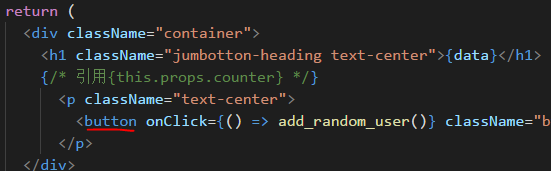
## 



### 新建components/User.js组件

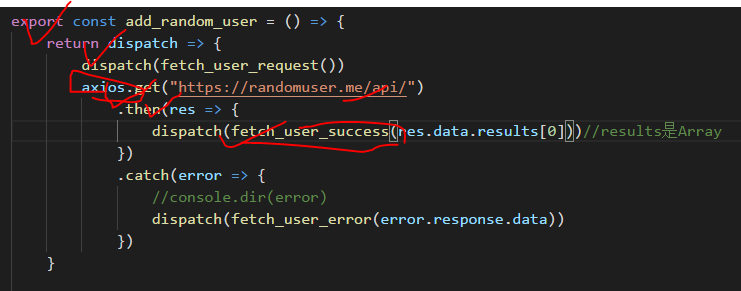


### （1）Ui按钮



### (2)actions/index.js

### 定义add\_random\_user() fetch\_user\_success()



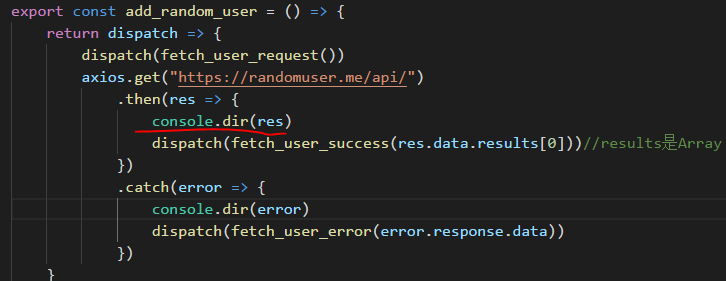
Axios:持久化数据的库：<https://www.npmjs.com/package/axios>

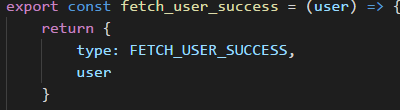
Axios.get(“url”).then(res=>{}).catch(error=>{})

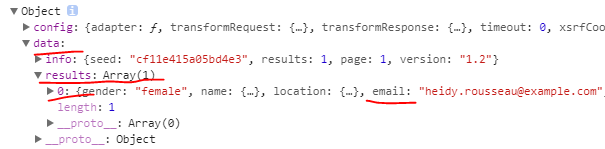
<https://...是个api>网址,返回的res.data.results是个数组，[0]取第一个，.email取邮箱数据



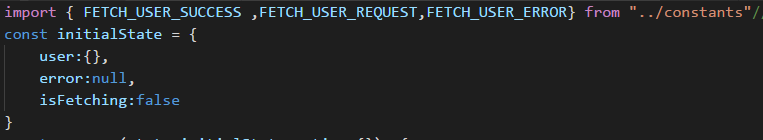
请求成功：fetch\_user\_success(参数为请求成功返回的user)

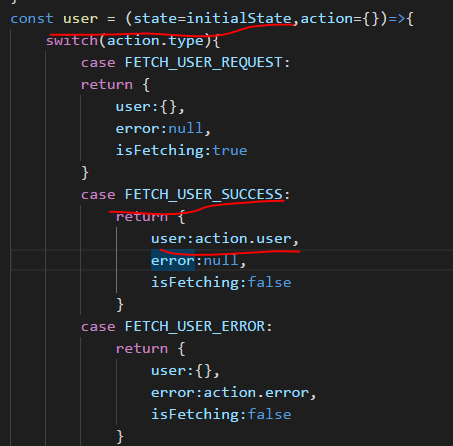






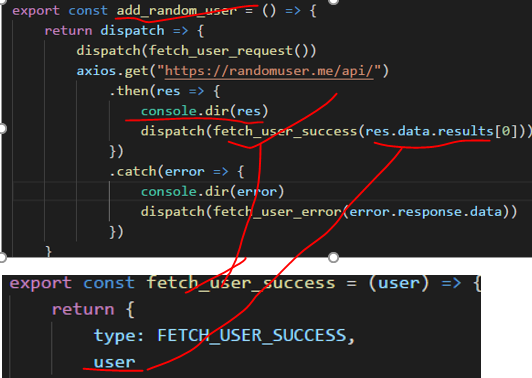
### (3)Reducers/user.js





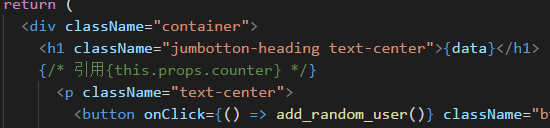
上图之所以action.user可获取api的返回对象

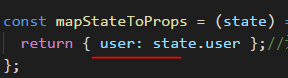
是因为下图actions/index.js中当type= success时，传递了user参数



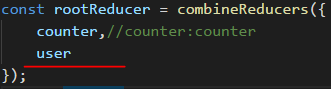
### （3）components/User.js

### onclick= add\_random\_user（）





### Reducers/index.js



### 扩展功能：页面显示请求中 Loading…,请求成功的邮箱，请求错误的错误提示

