react

React view 复杂应用 react redux

redux

dispatch action reducer state 单身边数据流

实时聊天功能的app

登录注册 信息完善 个人中心 牛人列表 BOSS列表

技术栈

前端页面 前段支撑 Antd React16 Redux Reducer 后端支撑 Express Socket.io MongoDB

react react.js node.js npm webpack es6基础

create-react-app 脚手架 第三方库 redux react-router 定制化配置

yarn start 启动

yarn add redux

Express+mongodb 开发web后台接口

React

- react 帮助你构建UI的库 MVC的view层
- facebook 出品 专注view层
- 一切皆组件
- 全是用es6语法 16 核心代码重写的版本
- 错误处理 npm install --save react react-dom

React 是什么

如何使用react实现组件化

react 进阶

React 基础语法

- import React
- class 组件 render直接使用
- render函数返回值jsx 转成js执行

React View层语法

- Js直接写成html
- Class 写成 className
- 变量用 {}

API

- 一切皆是组件
- 组件间通信通过属性传递
- 类实现组件 使用jsx语法

组件之间用props传递数据

- 使用 <组件 数据= "值" > 的形式传递
- 组件里使用 this.props获取值
- 如果只有render函数 用函数的形式写组件

父组件 子组件 return (

{this.props.name}
) 函数式组件 function Zujian(props){ return (
{props.name}
) }

组件内部 state 状态

组件内部通过state管理状态

- JSX本质是js 所以直接数组.map渲染列表 映射
- Constructor 设置组件的初始状态，构造函数 实例化执行 记得执行super(props)
- State 不可变对象 this.state 获取值 this.setSate 修改值 不能直接对state赋值

constructor(props) { super(props) this.state = { actor:['a','b','c] } } render() { return (

{this.state.actors.map(v=> { return
- {v}
})}

) }

修改state状态

方法一 constructor(props){ super(props) this.state={

```
}
this.addChange = this.addChange.bind(this)
```

} addChange() { this.setState={ actors:[...this.state.actors,'小明'+Math.random()] }

} render() {

} 添加 关于this指向的问题 箭头函数 <button onClick={()=>this.addChange()}>添加

或者 添加 addChange=()=>{

}

React 生命周期

React周期有若干钩子函数 在组件不同的状态执行

- 初始化周期
- 组件重新渲染生命周期
- 组件卸载生命周期

componentWillMount componentDidMount componentRecieveMount componentWillUpdate componentShouldUpdate componentDidUpdate

antd-mobile --save

npm install antd --save

常用组件

- Layout

按需加载

import {Button, List} from 'antd-mobile'; import '../node_modules/antd-mobile/dist/antd-mobile.css';

render() { return (<List renderHeader={()='演员列表}>
this.state.actors.map((v)=> { return <List.Item key={v}>{v}</List.Item> })) }

Redux

Redux 是什么

- 专注于状态管理的库 和react解耦
- 单一状态 single state 单向数据流
- 核心概念： store state action reducer

Redux 核心概念

Redux 实战

项目越来越大 管理不过来

- 人少数据少 setState
- 状态 redux管理 store 记录 state
- 需要改变 告诉 dispatch 做什么 action
- 处理变化的人 reducer 拿到 state和action 生成新的 state

Redux使用方法

- 通过reducer 新建store， 随时通过store.getState获取状态
- 需要状态变更， store.dispatch(action)
- Reducer 函数接受state和action，返回新的state，可以用store.subscribe监听每次修改

import {createStore} from 'redux';

function counter(state=0, action) { switch(action.type) { case '加演员' return state+1 case '减演员' return state-1 default return 10 } }

const store = createStore(counter) const init = store.getState()

console.log(init)

function listener() { const current =store.getState() console.log(现在有${current}演员) } store.subscribe(listener)

store.dispatch({type:'加演员'}) console.log(store.getState())

Redux如何和React使用

状态管理 订阅发布的设计模式

使用

- 把store.dispatch 传给组件 内部可以修改状态
- Subscribe 订阅render函数 每次修改都渲染
- Redux下官内容 移到单独的index.js reducer.js action.type.js index.redux.js

  先在index.rudex.js 封装 import React from 'react'; const ADD_GUN ='演员+1' const REMOVE_GUN = '演员-1' export function counter(state=0,action) { switch() { case ADD_GUN: return state+1 case REMOVE_GUN: return state-1 default: return 10 } } export function addGun() { return { type:'演员+1' } } export function removeGun() { return { type:'演员-1' } }

  store.dispatch({type:'演员+1'})

  index.js import {createStor} from 'redux'; import {couter} from './index.redux.js'; const store = createStore(counter) function render() { } 执行render render() subscriobe(render)

  App.js const store = this.props.store const num = store.getState render(){ return (

  {num} <button onClick={()=>store.dispatch(addGun())}>add
) }

组件解耦

内部组件不依赖于外部状态 外部只负责传进参数

处理异步

处理异步 调试工具 更优雅的和react结合

- Redux处理异步 需要redux-thunk插件
- npm i redux-devtools-extension 并开启
- 使用react-redux优雅的连接react和redux

redux默认值处理同步 异步任务需要redux-thunk中间件

- yarn add redux-thunk --save
- 使用applyMiddleware开启thunk中间件
- Action 返回函数 使用dispatch提交action

redux处理异步 首先 安装redux-thunk中间件 index.js

import {applyMiddleware} from 'redux'

import thunk from 'redux-thunk'

import {addGunAsync,removeGunAsync} from './index.redux.js'

index.redux.js

export function addGunAsync() { return dispatch => { setTimeout(()=>{ dispatch(addGun()) },2000) } }

App.js

const addGunAsync = this.props.addGunAsync <button onClick={()=>store.dispatch(addGunAsync())}>添加

redux 扩展程序 chorme

- 新建store判断window.devToolsExtension
- 使用compose结合thunk和window.devToolsExtension
- 调试窗的redux选项卡 实时看到state index.js import {compose} from 'redux'

const store = createStore(counter,compose(applyMiddleware(thunk), window.devToolsExtension ? window.devToolsExtension():f=>{}))

react-redux

方便管理 连接

- yarn add react-redux --save
- 忘记subscribe 记住 reducer action dispatch即可
- React-redux提供Provider和connect两个接口来连接
- Provider组件在应用最外层，传入store即可 用一次
- connect 负责从外部组件获取组件需要的参数
- connect装饰器

高阶组件

index.js

删除引入的其他函数 import {Provider} from 'react-redux'; ReactDom.render(( ))

App.js

import {connect} from 'react-redux' import {addGun,removeGun,addGunAsync,removeGunAsync} from './index.redux.js';

添加

const mapStatetoProps=(state)=>{ const num ={state} } const actionsCreators = {addGun,removeGun,addGunAsync,removeGunAsync}

App = connet(mapStatetoProps,actionsCreators)(App)

装饰器 优化代码

- yarn eject
- yarn add babel-plugin-transform-decorators-legacy --save-dev
- plugins

@connect(state=>({num=state}) {addGun,removeGun,addGunAsync,removeGunAsync})

进阶

- 什么数据放到react里
- Redux管理ajax
- Redux管理聊天数据

React4

开发单页应用

安装

yarn add react-router-dom --save

yarn add roadhog@2.5.0-beta.1 "plugins": [["@babel/plugin-proposal-decorators", { "legacy": true }]]

入门组件

- BrowserRouter 包裹整个应用
- Router 路由对应渲染的组建 可嵌套
- Link跳转
- 动态路由 router index.js import {BrowserRouter,Router,Route,Link} from 'react-router-dom'; function Home() { return

  # home

  } (
  - Frist
  -

- home
-
- pass
-

)

exact 精准匹配

招聘需求分析

用户中心 牛人 boss

登录 求职信息 个人信息 管理职位

注册 职位列表 查看信息

信息完善 聊天 socke.io

项目骨架

- src 前端源码目录
- server express目录
- component组件 container reducers 等 路由

页面骨架

- router划分页面
- 用户信息 聊天
- mongodb
- axios异步请求
- redux管理所有数据 antd-mobile 弱化css

前后端联调

yarn add axios --save

axios 如何发送异步请求

- 如何发送 使用proxy配置转发
- axios 拦截器 同意loading处理
- redux使用异步数据 渲染页面

简洁好用的请求库

“proxy":"http://localhost:909