实现动态获取数据，点击+号按钮数字增加，点击-号按钮数字减少

# 安装依赖

yarn add redux react-redux --dev

### **新建redux配套文件**

在src/store/目录下新建 以下文件

index.js store核心文件

reducer/index.js 负责记录操作的reducer文件

reducer.js

// 1 定义默认数据，后期可以从接口中获取

const defaultState = {

num: -1

};

// 2 创建和对外暴露一个函数 返回state

export default (state = defaultState, action) => {

return state;

}

store/index.js

// 1 引入 store生成器

import { createStore } from "redux";

// 2 引入reducer

import reducer from "./reducer";

// 3 创建和对外暴露store

export default createStore(reducer);

src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

// 1 引入 react-redux 负责将store和组件连接起来

import { Provider } from "react-redux";

// 1 引入 store

import store from "./store";

// 2 将App用 Provider 标签包裹起来

// 2 将store通过属性的方式传递到App组件上

ReactDOM.render(<Provider store={store} ><App /></Provider>, document.getElementById('root'));

App.js

import React, { Component } from 'react';

// 1 引入 react-redux 中 链接 组件和store的对象 connect

import { connect } from "react-redux";

class App extends Component {

render() {

return (

// 4 使用store中的数据

<div className="App">

{this.props.num}

<hr />

<button> + </button>

<button> - </button>

</div>

);

}

}

// 2 将 store中的数据传递到 App的props上

const mapStateToProps = (state) => {

return {

num: state.num

}

}

// 3 用 connect 将store中的数据通过props的方式传递到App上

export default connect(mapStateToProps, null)(App)

# 抽离组件，绑定事件

编辑App.js

将 两个按钮 抽离出来变成两个组件，这才满足组件共享数据的设计理念

// 加

class AddBtn extends Component {

render() {

return <button onClick={this.props.numAdd} >+</button>

}

}

// 减

class SubstraBtn extends Component {

render() {

return <button onClick={this.props.numSubStra}>-</button>

}

}

class App extends Component {

render() {

return (

<div className="App">

{this.props.num}

<hr />

{/\* 使用组件 传递props \*/}

<AddBtn {...this.props}></AddBtn>

{/\* 使用组件 传递props\*/}

<SubstraBtn {...this.props}></SubstraBtn>

</div>

);

}

}

<button onClick={this.props.numAdd} >+</button>

<button onClick={this.props.numSubStra}>-</button>

// 2 将行为action 链接到store和组件上

const mapDispatchToProps = (dispatch) => {

return {

// 点击事件中的加

numAdd: () => {

// 创建一个action，负责将行为类型和数据交给reducer

const action = {

// type是一个自定义的字符串

type: "NUM\_ADD",

value: 1

};

// 派发行为- 会将action 派发到 reducer中

dispatch(action);

},

// 点击事件中的减

numSubStra: () => {

const action = {

type: "NUM\_SUBSTRA",

value: 1

};

dispatch(action);

}

}

}

// 3 用 connect 将store中的数据通过props的方式传递到App上

export default connect(mapStateToProps, mapDispatchToProps)(App)

# 编辑 reducer逻辑

// 1 定义默认数据，后期可以从接口中获取

const defaultState = {

num: -1

};

// 2 创建和对外暴露一个函数 返回state

export default (state = defaultState, action) => {

// 当 action被派发时(dispatch)，会触发

if (action.type === "NUM\_ADD") {

// 复制一份旧的state

let newState = Object.assign({}, state);

newState.num += action.value;

// 将新的state返回，即可触发store的更新

return newState;

}

if (action.type === "NUM\_SUBSTRA") {

// 复制一份旧的state

let newState = Object.assign({}, state);

newState.num -= action.value;

return newState;

}

return state;

}

将state中的数据修改为对象形式

编辑 reducer/index.js

const defaultState = {

// 修改为对象形式

payload: {

num: -1

}

};

export default (state = defaultState, action) => {

if (action.type === "NUM\_ADD") {

let newState = Object.assign({}, state);

// 修改为对象形式

newState.payload.num += action.value;

return newState;

}

if (action.type === "NUM\_SUBSTRA") {

let newState = Object.assign({}, state);

// 修改为对象形式

newState.payload.num -= action.value;

return newState;

}

return state;

}

修改App.js中使用的state的代码

const mapStateToProps = (state) => {

return {

{/\* 修改为对象的形式 \*/}

num: state.payload.num

}

}

将action的type类型提取成常量的形式

新建文件 src/store/actionType/index.js

export const NUM\_ADD = "NUM\_ADD";

export const NUM\_SUBSTRA = "NUM\_SUBSTRA";

修改 使用到了 NUM\_ADD的文件

编辑 src/store/reducer/index.js

// 1 导入 type常量

import { NUM\_ADD, NUM\_SUBSTRA } from "../actionType";

export default (state = defaultState, action) => {

// 2 修改为常量的方式

if (action.type === NUM\_ADD) {

......

}

return state;

}

编辑 src/App.js

// 1 导入 type 常量

import { NUM\_ADD, NUM\_SUBSTRA } from "./store/actionType";

const mapDispatchToProps = (dispatch) => {

return {

numAdd: () => {

const action = {

// 2 使用 type常量

type: NUM\_ADD,

value: 1

};

dispatch(action);

}

}

使用action生成器来创建action

新建文件 src/store/actionCreator/index.js

import { NUM\_ADD,NUM\_SUBSTRA} from "../actionType";

export const numAdd = () => ({

type: NUM\_ADD,

value: 1

})

export const numSubstra = () => ({

type: NUM\_SUBSTRA,

value: 1

})

修改 App.js

// 1 导入action

import { numAdd, numSubstra } from "./store/actionCreator";

const mapDispatchToProps = (dispatch) => {

return {

numAdd: () => {

// 2 修改为 生成器生成的action

dispatch(numAdd());

}

}

}

拆分和合并reducer

编辑 actionType/index.js

export const NUM\_ADD = "NUM\_ADD";

export const NUM\_SUBSTRA = "NUM\_SUBSTRA";

// 新增 增加 苹果action type

export const APPLE\_NUM\_ADD = "APPLE\_NUM\_ADD";

// 新增 减少 苹果action type

export const APPLE\_NUM\_SUBSTRA = "APPLE\_NUM\_SUBSTRA";

编辑 actionCreator/index.js

// 新增 添加苹果 action

export const appleNumAdd = () => ({

type: APPLE\_NUM\_ADD,

value: 1

})

// 新增 减少苹果 action

export const appleNumSubstra = () => ({

type: APPLE\_NUM\_SUBSTRA,

value: 1

})

新建文件 reducer/numReducer.js

将 以前 reducer/index.js 全部复制过去即可

import { NUM\_ADD, NUM\_SUBSTRA } from "../actionType";

const defaultState = {

payload: {

num: -1

}

};

export default (state = defaultState, action) => {

if (action.type === NUM\_ADD) {

let newState = Object.assign({}, state);

newState.payload.num += action.value;

return newState;

}

if (action.type === NUM\_SUBSTRA) {

let newState = Object.assign({}, state);

newState.payload.num -= action.value;

return newState;

}

return state;

}

新建文件 reducer/fruitReducer.js

import { APPLE\_NUM\_ADD, APPLE\_NUM\_SUBSTRA } from "../actionType";

const defaultState = {

payload: {

appleNum: 110

}

};

export default (state = defaultState, action) => {

if (action.type === APPLE\_NUM\_ADD) {

let newState = Object.assign({}, state);

newState.payload.appleNum += action.value;

return newState;

}

if (action.type === APPLE\_NUM\_SUBSTRA) {

let newState = Object.assign({}, state);

newState.payload.appleNum -= action.value;

return newState;

}

return state;

}

编辑 reducer/index.js 用来合并 两个reducer fruitReducer 和 numReducer

// 1 引入 合并reducer的对象

import { combineReducers } from "redux";

import fruitReducer from "./fruitReducer";

import numReducer from "./numReducer";

// 2 对象的形式传入 要合并的reducer

const rootReducer = combineReducers({ numReducer, fruitReducer });

export default rootReducer;

修改App.js

import React, { Component } from 'react';

import { connect } from "react-redux";

// 1 多导入两个action appleNumAdd 和 appleNumSubstra

import { numAdd, numSubstra, appleNumAdd, appleNumSubstra } from "./store/actionCreator";

class AddNumBtn extends Component {

render() {

return <button onClick={this.props.numAdd} >+</button>

}

}

class SubstraNumBtn extends Component {

render() {

return <button onClick={this.props.numSubStra}>-</button>

}

}

// 2 新增的组件

class AddFruitBtn extends Component {

render() {

// 2.1 新绑定的事件

return <button onClick={this.props.appleNumAdd} >+</button>

}

}

// 2 新增的组件

class SubstraFruitBtn extends Component {

render() {

// 2.1 新绑定的事件

return <button onClick={this.props.appleNumSubStra}>-</button>

}

}

class App extends Component {

render() {

// 3 修改过的页面代码

return (

<div className="App">

数量 {this.props.num}

<hr />

<AddNumBtn {...this.props}></AddNumBtn>

<SubstraNumBtn {...this.props}></SubstraNumBtn>

<hr />

水果数量 {this.props.appleNum}

<br />

{/\* 3.1 引入新组件 \*/}

<AddFruitBtn {...this.props}></AddFruitBtn>

{/\* 3.1 引入新组件 \*/}

<SubstraFruitBtn {...this.props}></SubstraFruitBtn>

</div>

);

}

}

const mapStateToProps = (state) => {

// 4 修改数据的获取方法

return {

num: state.numReducer.payload.num,

appleNum: state.fruitReducer.payload.appleNum

}

}

const mapDispatchToProps = (dispatch) => {

return {

numAdd: () => {

dispatch(numAdd());

},

numSubStra: () => {

dispatch(numSubstra());

},

// 5 新增的action

appleNumAdd: () => {

dispatch(appleNumAdd());

},

// 5 新增的action

appleNumSubStra: () => {

dispatch(appleNumSubstra());

},

}

}

// 3 用 connect 将store中的数据通过props的方式传递到App上

export default connect(mapStateToProps, mapDispatchToProps)(App)

添加异步action redux-thunk

安装依赖 redux-thunk

yarn add redux-thunk --dev

修改store/index.js

// 1 引入 redux的中间件连接器

import { createStore, applyMiddleware } from "redux";

import reducer from "./reducer";

// 1 引入 redux-thunk

import reduxThunk from "redux-thunk";

// 2 使用中间件连接器将redux-thunk传入 store构造器

export default createStore(reducer, applyMiddleware(reduxThunk));

修改 actionCreator/index.js

// 1 修改 减少苹果的action 为异步的形式

export const appleNumSubstra = () => {

// 2 返回一个函数

return (dispatch) => {

// 3 开启异步 后期将 setTimeout 替换成异步的方式即可

setTimeout(() => {

const action = {

type: APPLE\_NUM\_SUBSTRA,

value: 1

};

// 4 开启派发

dispatch(action);

}, 2000);

}

}