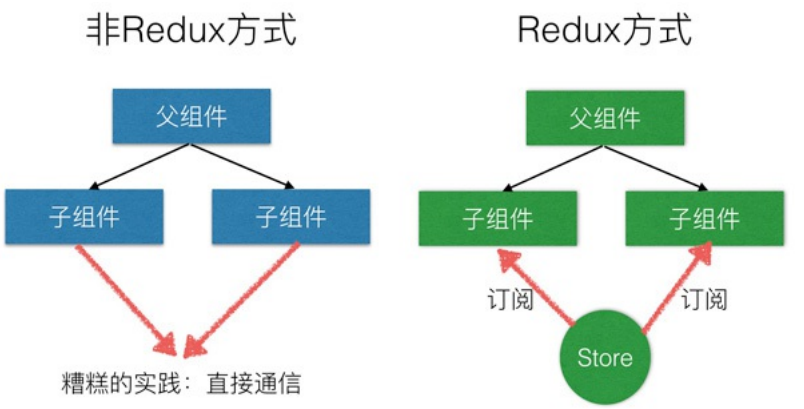
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=156 sentences=335, words=2341

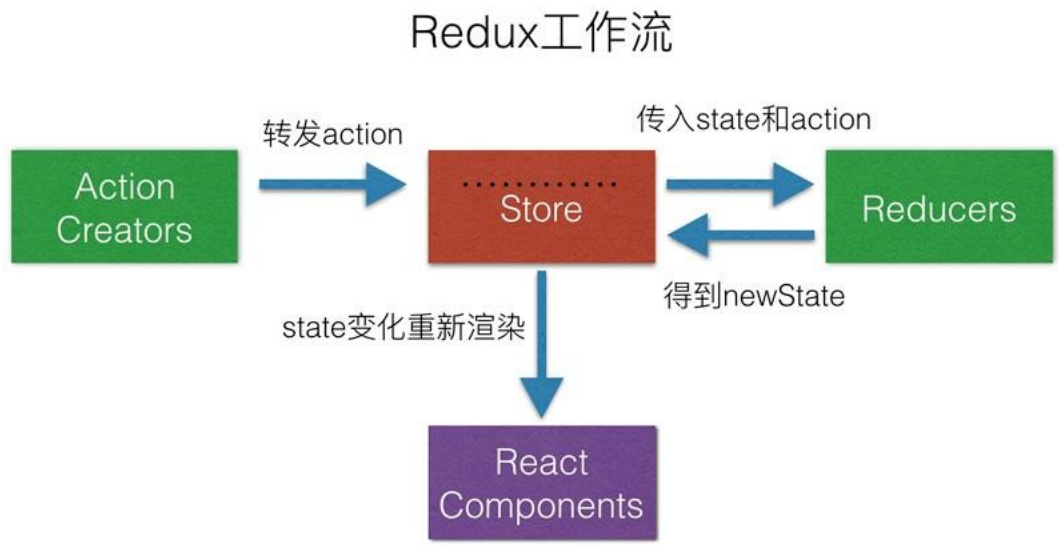
1. Redux应用场景

- 随着 JavaScript 单页应用开发日趋复杂,管理不断变化的 state 非常困难
- Redux的出现就是为了解决state里的数据问题
- 在React中，数据在组件中是单向流动的
- 数据从一个方向父组件流向子组件(通过props)，由于这个特征，两个非父子关系的组件（或者称作兄弟组件）之间的通信就比较麻烦



2. Redux设计思想

- Redux是将整个应用状态存储到到一个地方，称为store
- 里面保存一棵状态树state tree
- 组件可以派发dispatch行为action给store,而不是直接通知其它组件
- 其它组件可以通过订阅store中的状态(state)来刷新自己的视图



3. Redux三大原则

- 整个应用的 state 被储存在一棵 object tree 中，并且这个 object tree 只存在于唯一一个 store 中
- State 是只读的，惟一改变 state 的方法就是触发 action，action 是一个用于描述已发生事件的普通对象 使用纯函数来执行修改，为了描述action如何改变state tree ，你需要编写 reducers
- 单一数据源的设计让React的组件之间的通信更加方便，同时也便于状态的统一管理

4. 原生计数器

- redux (https://github.com/reduxjs/redux)
- createStore.ts (https://gitee.com/zhufengpeixun/redux/blob/master/src/createStore.ts)

```
create-react-app zhufeng_redux_prepare
cd zhufeng_redux_prepare
cnpm install redux -S
yarn start
```

4.1 public\index.html

public\index.html

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <title>React Apptitle>
  head>
  <body>
    <div id="root">div>
    <div id="counter">
      <p id="counter-value">0p>
      <button id="add-btn">+button>
      <button id="minus-btn">-button>
    div>
  body>
html>
```

4.2 src\index.js

src\index.js

```
import { createStore} from './redux';
let counterValue = document.getElementById('counter-value');
let addBtn = document.getElementById('add-btn');
let minusBtn = document.getElementById('minus-btn');

const ADD = 'ADD';
const MINUS = 'MINUS';
let initState = { number: 0 };

const reducer = (state = initState, action) => {
    switch (action.type) {
        case ADD:
            return { number: state.number + 1 };
        case MINUS:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let store = createStore(reducer);
function render() {
    counterValue.innerHTML = store.getState().number + '';
}
store.subscribe(render);
render();
addBtn.addEventListener('click', function () {
    store.dispatch({ type: ADD });
});
minusBtn.addEventListener('click', function () {
    store.dispatch({ type: MINUS });
});
```

4.3 redux\index.js

src\redux\index.js

```
import createStore from './createStore'
export {
    createStore
}
```

4.4 redux\createStore.js

src\redux\createStore.js

```
const createStore = (reducer, preloadedState) => {
  let state=preloadedState;
  let listeners = [];
  function getState() {
    return state;
  }
  function dispatch(action) {
    state = reducer(state, action);
    listeners.forEach(l => l());
    return action;
  }
  function subscribe(listener) {
    listeners.push(listener);
    return () => {
      listeners = listeners.filter(l => l !== listener);
    }
  }
  dispatch({ type: '@@REDUX/INIT' });
  return {
    getState,
    dispatch,
    subscribe
  }
}
export default createStore;
```

5. React计数器

5.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';

ReactDOM.render(<Counter1 />, document.getElementById('root'));
```

5.2 Counter1.js

src\components\Counter1.js

```
import React, { Component } from 'react';
import { createStore} from '../redux';
const ADD = 'ADD';
const MINUS = 'MINUS';
const reducer = (state = initState, action) => {
    switch (action.type) {
        case ADD:
            return { number: state.number + 1 };
        case MINUS:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let initState = { number: 0 };
const store = createStore(reducer, initState);
export default class Counter extends Component {
    unsubscribe;
    constructor(props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (
            <div>
                <p>{this.state.number}p>
                <button onClick={() => store.dispatch({ type: 'ADD' })}>+button>
                <button onClick={() => store.dispatch({ type: 'MINUS' })}>-button>
                <button onClick={
                    () => {
                        setTimeout(() => {
                            store.dispatch({ type: 'ADD' });
                        }, 1000);
                    }
                }>1秒后加1button>
            div>
        )
    }
}
```

6. bindActionCreators

- bindActionCreators (https://gitee.com/zhufengpeixun/redux/blob/master/src/bindActionCreators.ts)

6.1 Counter1.js

src\components\Counter1.js

```
import React, { Component } from 'react';
+import { createStore,bindActionCreators} from '../redux';
const ADD = 'ADD';
const MINUS = 'MINUS';
const reducer = (state = initState, action) => {
    switch (action.type) {
        case ADD:
            return { number: state.number + 1 };
        case MINUS:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let initState = { number: 0 };
const store = createStore(reducer, initState);
+function add() {
+    return { type: 'ADD' };
+}
+function minus() {
+    return { type: 'MINUS' };
+}
+const actions = { add, minus };
+const boundActions = bindActionCreators(actions, store.dispatch);
export default class Counter extends Component {
    unsubscribe;
    constructor(props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (

                {this.state.number}
+               +
+               -
                 {
                        setTimeout(() => {
+                         boundActions.add();
                        }, 1000);
                    }
                }>1秒后加1

        )
    }
}
```

6.2 bindActionCreators.js

src\redux\bindActionCreators.js

```
function bindActionCreator(actionCreator, dispatch) {
    return function (...args) {
        return dispatch(actionCreator.apply(this, args))
    }
}

export default function bindActionCreators(actionCreators, dispatch) {
    const boundActionCreators = {}
    for (const key in actionCreators) {
        const actionCreator = actionCreators[key]
        if (typeof actionCreator === 'function') {
            boundActionCreators[key] = bindActionCreator(actionCreator, dispatch)
        }
    }
    return boundActionCreators
}
```

6.3 src\redux\index.js

src\redux\index.js

```
export {default as createStore} from './createStore'
export {default as bindActionCreators} from './bindActionCreators'
```

7. combineReducers.js

src\redux\combineReducers.js -combineReducers (https://gitee.com/zhufengpeixun/redux/blob/master/src/combineReducers.ts)

7.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
ReactDOM.render(<div><Counter1/><Counter2/>div>, document.getElementById('root'));
```

7.2 action-types.js

src\store\action-types.js

```
export const ADD1 = 'ADD1';
export const MINUS1 = 'MINUS1';

export const ADD2 = 'ADD2';
export const MINUS2 = 'MINUS2';
```

7.3 counter1.js

src\store\reducers\counter1.js

```
import * as types from '../action-types';

let initialState = { number: 0 }

export default function (state=initialState, action) {
    switch (action.type) {
        case types.ADD1:
            return { number: state.number + 1 };
        case types.MINUS1:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

7.4 counter2.js

src\store\reducers\counter2.js

```
import * as types from '../action-types';

let initialState = { number: 0 };

export default function (state=initialState, action) {
    switch (action.type) {
        case types.ADD2:
            return { number: state.number + 1 };
        case types.MINUS2:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

7.5 reducers\index.js

src\store\reducers\index.js

```
import { combineReducers} from '../../redux';
import counter1 from './counter1';
import counter2 from './counter2';
let rootReducer = combineReducers({
    counter1,
    counter2
});
export default rootReducer;
```

7.6 store\index.js

src\store\index.js

```
import { createStore } from '../redux';
import reducer from './reducers';
const store = createStore(reducer, { counter1: { number: 0 }, counter2: { number: 0 } });
export default store;
```

7.7 actions\counter1.js

src\store\actions\counter1.js

```
import * as types from '../action-types';

const actions = {
    add1() {
        return { type: types.ADD1 };
    },
    minus1() {
        return { type: types.MINUS1 };
    }
}
export default actions;
```

7.8 actions\counter2.js

src\store\actions\counter2.js

```
import * as types from '../action-types';

const actions = {
    add2() {
        return { type: types.ADD2 };
    },
    minus2() {
        return { type: types.MINUS2 };
    }
}
export default actions;
```

7.9 Counter1.js

src\components\Counter1.js

```
import React, { Component } from 'react';
+import { bindActionCreators} from '../redux';
+import actions from '../store/actions/counter1';
+import store from '../store';
+const boundActions = bindActionCreators(actions, store.dispatch);
export default class Counter extends Component {
    unsubscribe;
    constructor(props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
+        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().counter1.number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (

                {this.state.number}
+                +
+                -

        )
    }
}
```

7.10 Counter2.js

src\components\Counter2.js

```
import React, { Component } from 'react';
+import { bindActionCreators} from '../redux';
+import actions from '../store/actions/counter2';
+import store from '../store';
+const boundActions = bindActionCreators(actions, store.dispatch);
export default class Counter extends Component {
    unsubscribe;
    constructor(props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
+        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().counter2.number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (

                {this.state.number}
+                +
+                -
                 {
                        setTimeout(() => {
+                            boundActions.add2();
                        }, 1000);
                    }
                }>1秒后加1

        )
    }
}
```

7.11 combineReducers.js

src\redux\combineReducers.js

```
function combineReducers(reducers){
    return function combination(state={},action){
       let nextState = {};
       for(let key in reducers){

        let nextStateForKey = state[key];

        let reducerForKey = reducers[key];

        nextState[key] = reducerForKey(nextStateForKey, action);
       }
       return nextState;
    }
}
export default combineReducers;
```

7.12 redux\index.js

src\redux\index.js

```
export {default as createStore} from './createStore'
export {default as bindActionCreators} from './bindActionCreators';
+export {default as combineReducers} from './combineReducers';
```

8. react-redux

- Provider.js (https://gitee.com/zhufengpeixun/react-redux/blob/master/src/components/Provider.js)
- connect.js (https://gitee.com/zhufengpeixun/react-redux/blob/master/src/connect/connect.js)

8.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
import store from './store';
import { Provider } from './react-redux';
ReactDOM.render(
<Provider store={store}>
    <Counter1 />
    <Counter2 />
Provider>, document.getElementById('root'));
```

8.2 Counter1.js

src\components\Counter1.js

```
import React, { Component } from 'react';
import actions from '../store/actions/counter1';
import { connect } from '../react-redux';
class Counter1 extends Component {
    render() {
        let { number, add1, minus1 } = this.props;
        return (
            <div>
                <p>{number}p>
                <button onClick={add1}>+button>
                <button onClick={minus1}>-button>
                <button onClick={
                    () => setTimeout(() => add1(), 1000)
                }>1秒后加1button>
            div>
        )
    }
}
let mapStateToProps = (state) => state.counter1;
export default connect(
    mapStateToProps,
    actions
)(Counter1)
```

8.3 Counter2.js

src\components\Counter2.js

```
import React from 'react';
import { useSelector,useDispatch} from '../react-redux';
const Counter2 =  (props) => {
   const counter2 = useSelector(state => state.counter2);
   const dispatch = useDispatch();
        return (
            <div>
                <p>{counter2.number}p>
                <button onClick={()=>dispatch({type:'ADD2'})}>+button>
                <button onClick={()=>dispatch({type:'MINUS2'})}>-button>
            div>
        )
}
export default Counter2;
```

8.4 Provider.js

src\react-redux\Provider.js

```
import React from 'react'
import ReactReduxContext from './ReactReduxContext';

export default function(props){
  return (
    <ReactReduxContext.Provider value={{ store: props.store }}>
      {props.children}
    ReactReduxContext.Provider>
  )
}
```

8.5 ReactReduxContext.js

src\react-redux\ReactReduxContext.js

```
import React from 'react';
export const ReactReduxContext = React.createContext(null)
export default ReactReduxContext;
```

8.6 connect.js

src\react-redux\connect.js

```
import React, { useContext, useMemo, useReducer, useLayoutEffect } from 'react';
import ReactReduxContext from './ReactReduxContext';
import { bindActionCreators } from '../redux';

function connect(mapStateToProps, mapDispatchToProps) {
    return function (OldComponent) {
        return class extends React.Component {
            static contextType = ReactReduxContext;
            constructor(props, context) {
                super(props);
                const { store } = context;
                const { getState, subscribe, dispatch } = store;
                this.state = mapStateToProps(getState());
                this.unsubscribe = subscribe(() => {
                    this.setState(mapStateToProps(getState()));
                });
                let dispatchProps;
                if (typeof mapDispatchToProps === 'function') {
                    dispatchProps = mapDispatchToProps(dispatch);
                } else if (typeof mapDispatchToProps === 'object') {
                    dispatchProps = bindActionCreators(mapDispatchToProps, dispatch);
                } else {
                    dispatchProps = { dispatch };
                }
                this.dispatchProps = dispatchProps;
            }
            componentWillUnmount() {
                this.unsubscribe();
            }
            render() {
                return <OldComponent {...this.props} {...this.state} {...this.dispatchProps} />
            }
        }
    }
}

function connect(mapStateToProps, mapDispatchToProps) {
    return function (OldComponent) {
        return function (props) {
            const { store } = useContext(ReactReduxContext);
            const { getState, dispatch, subscribe } = store;
            const prevState = getState();
            const stateProps = useMemo(() => mapStateToProps(prevState), [prevState]);
            let dispatchProps = useMemo(() => {
                console.log('dispatchProps render');
                let dispatchProps;
                if (typeof mapDispatchToProps === 'function') {
                    dispatchProps = mapDispatchToProps(dispatch);
                } else if (typeof mapDispatchToProps === 'object') {
                    dispatchProps = bindActionCreators(mapDispatchToProps, dispatch);
                } else {
                    dispatchProps = { dispatch };
                }
                return dispatchProps;
            }, [dispatch]);
            const [, forceUpdate] = useReducer(x => x + 1, 0);
            useLayoutEffect(() => {
                return subscribe(forceUpdate);
            }, [subscribe]);
            return <OldComponent {...props} {...stateProps} {...dispatchProps} />
        }
    }
}
export default connect;
```

8.7 hooks

8.7.1 useDispatch.js

src\react-redux\hooks\useDispatch.js

```
import {useContext} from 'react';
import ReactReduxContext from '../ReactReduxContext';

const  useDispatch = ()=>{
    const {store} = useContext(ReactReduxContext);
    return store.dispatch;
}
export default  useDispatch
```

8.7.2 shallowEqual.js

src\react-redux\shallowEqual.js

```
export function shallowEqual(obj1, obj2) {
    if (obj1 === obj2) {
        return true;
    }
    if (typeof obj1 != "object" || obj1 === null || typeof obj2 != "object" || obj2 === null) {
        return false;
    }
    let keys1 = Object.keys(obj1);
    let keys2 = Object.keys(obj2);
    if (keys1.length !== keys2.length) {
        return false;
    }
    for (let key of keys1) {
        if (!obj2.hasOwnProperty(key) || obj1[key] !== obj2[key]) {
            return false;
        }
    }
    return true;
}
export default shallowEqual;
```

8.7.3 useSelector.js

src\react-redux\hooks\useSelector.js

```
import {useContext,useLayoutEffect,useReducer,useRef} from 'react';
import { shallowEqual } from '../';
import ReactReduxContext from "../ReactReduxContext";
function useSelector(selector,equalityFn=shallowEqual){
    const {store} = useContext(ReactReduxContext);
    let lastSelectedState = useRef(null);

    let state = store.getState();
    let selectedState = selector(state);

    let [,forceUpdate] = useReducer(x=>x+1,0);
    useLayoutEffect(()=>store.subscribe(()=>{

        let selectedState = selector(store.getState());
        if(!equalityFn(lastSelectedState.current,selectedState)){
            console.log('重新渲染');
            forceUpdate();
            lastSelectedState.current = selectedState;
        }
    }),[]);

    return selectedState;
}
export default useSelector;
```

8.7.4 useBoundDispatch.js

src\react-redux\hooks\useBoundDispatch.js

```
import React from 'react';
import { bindActionCreators } from '../../redux';
import ReactReduxContext from '../ReactReduxContext';

function useBoundDispatch(actions){
    const {store} = React.useContext(ReactReduxContext);
    let boundActions = bindActionCreators(actions,store.dispatch);
    return boundActions;
}
export default useBoundDispatch;
```

8.7.5 hooks\index.js

src\react-redux\hooks\index.js

```
export {default as useDispatch} from './useDispatch';
export {default as useSelector} from './useSelector';
export {default as useBoundDispatch} from './useBoundDispatch';
```

8.7.6 react-redux\index.js

src\react-redux\index.js

```
export {default as Provider} from './Provider';
export {default as connect} from './connect';
export {default as shallowEqual} from './shallowEqual';
export {useDispatch,useSelector} from './hooks';
```

8.7.7 Counter1.js

src\components\Counter1.js

```
import React from 'react';
import {useDispatch,useSelector} from '../react-redux';
function Counter1(){
    let state = useSelector(state=>state.counter1);
    let dispatch = useDispatch();
    return (
        <div>
               <p>{state.number}p>
               <button onClick={()=>dispatch({type:'ADD1'})}>+button>
           div>
    )
}
export default Counter1;
```

8.7.8 Counter3.js

src\components\Counter6.js

```
import React from 'react';
import actions from '../store/actions/counter2';
import {useSelector,useBoundDispatch} from '../react-redux';
function Counter3(){
    let {number} = useSelector((state)=>state.counter2);
    let boundActions = useBoundDispatch(actions);
    return (
        <div>
            <p>{number}p>
            <button onClick={boundActions.add2}>+button>
            <button onClick={boundActions.minus2}>-button>
        div>
    )
}
export default Counter3;
```