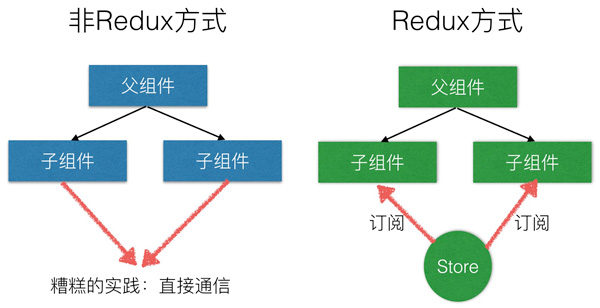
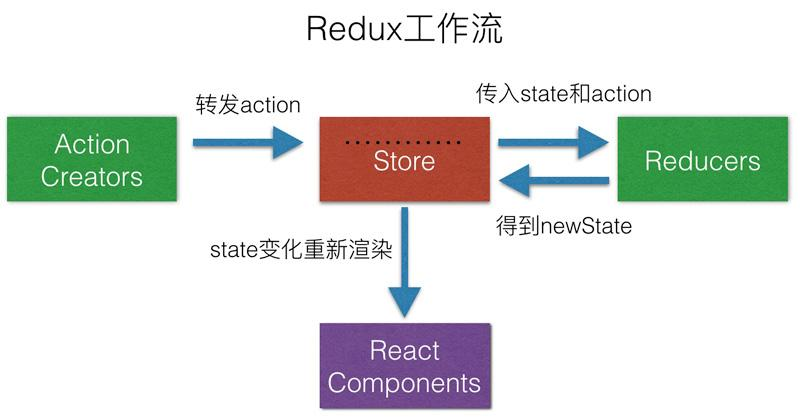
## **1. Redux应用场景**

* 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)

### **4.1 index.html**

<div id="counter">

<p id="counter-value">0</p>

<button id="increment-btn">+</button>

<button id="decrement-btn">-</button>

</div>

### **4.2 index.js**

**import** {createStore} **from** 'redux';**let** counterValue = document.getElementById('counter-value');**let** incrementBtn = document.getElementById('increment-btn');**let** decrementBtn = document.getElementById('decrement-btn');

**const** INCREMENT='INCREMENT';**const** DECREMENT = 'DECREMENT';**let** initState = 0;**function** **reducer**(state=initState,action){

**switch**(action.type){

**case** INCREMENT:

**return** state + 1;

**case** DECREMENT:

**return** state - 1;

**default**:

**return** state;

}

}**let** store=createStore(reducer);**function** **render**() {

counterValue.innerHTML=store.getState();

}

store.subscribe(render);

render();

incrementBtn.addEventListener('click',**function** () {

store.dispatch({type:INCREMENT});

});

decrementBtn.addEventListener('click',**function** () {

store.dispatch({type:DECREMENT});

});

### **4.3 redux**

#### **4.3.1 index.js**

src\redux\index.js

**import** createStore **from** './createStore'**export** {

createStore

}

#### **4.3.2 createStore.js**

src\redux\createStore.js

**export** **default** **function** **createStore**(reducer, preloadedState) {

**let** currentState = preloadedState;

**let** currentListeners = [];

**function** **getState**() {

**return** currentState;

}

**function** **subscribe**(listener) {

currentListeners.push(listener);

**return** **function** **unsubscribe**() {

**const** index = currentListeners.indexOf(listener);

currentListeners.splice(index, 1);

};

}

**function** **dispatch**(action) {

**if** (Object.getPrototypeOf(action) !== Object.prototype) {

**throw** **new** Error(`动作必须是一个纯对象，如果想进行异步操作请使用中间件`);

}

**if** (**typeof** action.type === "undefined") {

**throw** **new** Error(`动作不能一个值为undefined的type属性`);

}

currentState = reducer(currentState, action);

**for** (**let** i = 0; i < currentListeners.length; i++) {

**const** listener = currentListeners[i];

listener();

}

**return** action;

}

dispatch({ type:'@@redux/INIT' });

**return** {

dispatch,

subscribe,

getState

};

}

## **5. React计数器**

* 使用React实现一个计数器

**import** React, { Component } **from** 'react';**import** { createStore } **from** '../redux';**function** **reducer**(state=0,action){

**switch**(action.type){

**case** 'INCREMENT':

**return** state + 1;

**case** 'DECREMENT':

**return** state - 1;

**default**:

**return** state;

}

}**const** store = createStore(reducer,0);**export** **default** **class** **Counter** **extends** **Component** {

**constructor**(props) {

**super**(props);

**this**.state = { value: 0 };

}

componentDidMount() {

**this**.unsubscribe = store.subscribe(() => **this**.setState({ value: store.getState() }));

}

componentWillUnmount() {

**this**.unsubscribe();

}

render() {

**return** (

<div>

<p>{this.state.value}</p>

<button onClick={() => store.dispatch({ type: 'INCREMENT' })}>+</button>

<button onClick={() => store.dispatch({ type: 'DECREMENT' })}>-</button>

<button onClick={

() => {

setTimeout(() => {

store.dispatch({ type: 'INCREMENT' })

}, 1000);

}

}>1秒后加1</button>

</div>

)

}

}

## **6. bindActionCreators.js**

### **6.1 Counter.js**

**import** React, { Component } **from** 'react';**import** { createStore,bindActionCreators} **from** '../redux';**function** **reducer**(state=0,action){

**switch**(action.type){

**case** 'INCREMENT':

**return** state + 1;

**case** 'DECREMENT':

**return** state - 1;

**default**:

**return** state;

}

}**const** store = createStore(reducer,0);**function** **increment**(){

**return** {type:'INCREMENT'};

}**function** **decrement**(){

**return** {type:'DECREMENT'};

}**const** actions = {increment,decrement};//const boundIncrement = bindActionCreators(increment,store.dispatch);//可以传一个函数**const** boundActions = bindActionCreators(actions,store.dispatch);//也可以传对象

**export** **default** **class** **Counter** **extends** **Component** {

**constructor**(props) {

**super**(props);

**this**.state = { value: 0 };

}

componentDidMount() {

**this**.unsubscribe = store.subscribe(() => **this**.setState({ value: store.getState() }));

}

componentWillUnmount() {

**this**.unsubscribe();

}

render() {

**return** (

<div>

<p>{this.state.value}</p>

<button onClick={boundIncrement}>+</button>

<button onClick={boundIncrement}>-</button>

</div>

)

}

}

### **6.2 bindActionCreators.js**

bindActionCreators.js

**function** **bindActionCreator**(actionCreator, dispatch) {

**return** **function**() {

**return** dispatch(actionCreator.apply(**this**, arguments))

}

}**export** **default** **function** **bindActionCreators**(actionCreators, dispatch) {

**if** (**typeof** actionCreators === 'function') {

**return** bindActionCreator(actionCreators, dispatch)

}

**const** boundActionCreators = {}

**for** (**const** key **in** actionCreators) {

**const** actionCreator = actionCreators[key]

**if** (**typeof** actionCreator === 'function') {

boundActionCreators[key] = bindActionCreator(actionCreator, dispatch)

}

}

**return** boundActionCreators

}

## **7. combineReducers**

### **7.1 src/index.js**

**import** React **from** 'react';**import** ReactDOM **from** 'react-dom';**import** Counter1 **from** './components/Counter1';**import** Counter2 **from** './components/Counter2';

ReactDOM.render(<><Counter1/><hr/><Counter2/></>,document.getElementById('root'));

### **7.2 redux/index.js**

src\redux\index.js

**import** createStore **from** './createStore'**import** bindActionCreators **from** './bindActionCreators'**import** combineReducers **from** './combineReducers'**export** {

createStore,

bindActionCreators,

combineReducers

}

### **7.3 combineReducers.js**

src\redux\combineReducers.js

**export** **default** **function** **combineReducers**(reducers) {

**const** reducerKeys = Object.keys(reducers)

**return** **function** **combination**(state = {}, action) {

**const** nextState = {}

**for** (**let** i = 0; i < reducerKeys.length; i++) {

**const** key = reducerKeys[i];

**const** reducer = reducers[key];

**const** previousStateForKey = state[key];

**const** nextStateForKey = reducer(previousStateForKey, action);

nextState[key] = nextStateForKey;

}

**return** nextState;

}

}

### **7.4 store\index.js**

src\store\index.js

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

### **7.5 action-types.js**

src\store\action-types.js

**export** **const** INCREMENT1 = 'INCREMENT1';**export** **const** DECREMENT1 = 'DECREMENT1';

**export** **const** INCREMENT2 = 'INCREMENT2';**export** **const** DECREMENT2 = 'DECREMENT2';

### **7.6 actions**

#### **7.6.1 counter1.js**

src\store\actions\counter1.js

**import** \* **as** types **from** '../action-types';**export** **default** {

increment1(){

**return** {type:types.INCREMENT1};

},

decrement1(){

**return** {type:types.DECREMENT1};

}

}

#### **7.6.2 counter2.js**

src\store\actions\counter2.js

**import** \* **as** types **from** '../action-types';**export** **default** {

increment2(){

**return** {type:types.INCREMENT2};

},

decrement2(){

**return** {type:types.DECREMENT2};

}

}

### **7.7 reducers**

#### **7.7.1 index.js**

src\store\reducers\index.js

**import** {combineReducers} **from** '../../redux';**import** counter1 **from** './counter1';**import** counter2 **from** './counter2';**export** **default** combineReducers({

counter1,

counter2

});

#### **7.7.2 counter1.js**

src/store/reducers/counter1.js

**import** \* **as** types **from** '../action-types';**export** **default** **function** (state=0,action){

**switch**(action.type){

**case** types.INCREMENT1:

**return** state + 1;

**case** types.DECREMENT1:

**return** state - 1;

**default**:

**return** state;

}

}

#### **7.7.3 counter2.js**

src/store/reducers/counter2.js

**import** \* **as** types **from** '../action-types';**export** **default** **function** (state=0,action){

**switch**(action.type){

**case** types.INCREMENT2:

**return** state + 1;

**case** types.DECREMENT2:

**return** state - 1;

**default**:

**return** state;

}

}

### **7.8 Component**

#### **7.8.1 Counter1.js**

src\components\Counter1.js

**import** React, { Component } **from** 'react';**import** actions **from** '../store/actions/counter1';**import** store **from** '../store';**import** {bindActionCreators} **from** '../redux';**const** boundActions = bindActionCreators(actions,store.dispatch);**export** **default** **class** **Counter** **extends** **Component** {

**constructor**(props) {

**super**(props);

**this**.state = {value:0}

}

componentDidMount() {

**this**.unsubscribe = store.subscribe(() => **this**.setState({ value: store.getState().counter1 }));

}

componentWillUnmount() {

**this**.unsubscribe();

}

render() {

**return** (

<div>

<p>{this.state.value}</p>

<button onClick={boundActions.increment1}>+</button>

<button onClick={boundActions.decrement1}>-</button>

</div>

)

}

}

#### **7.8.2 Counter2.js**

src\components\Counter2.js

**import** React, { Component } **from** 'react';**import** actions **from** '../store/actions/counter2';**import** store **from** '../store';**import** {bindActionCreators} **from** '../redux';**const** boundActions = bindActionCreators(actions,store.dispatch);**export** **default** **class** **Counter** **extends** **Component** {

**constructor**(props) {

**super**(props);

**this**.state = {value:0}

}

componentDidMount() {

**this**.unsubscribe = store.subscribe(() => **this**.setState({ value: store.getState().counter2 }));

}

componentWillUnmount() {

**this**.unsubscribe();

}

render() {

**return** (

<div>

<p>{this.state.value}</p>

<button onClick={boundActions.increment2}>+</button>

<button onClick={boundActions.decrement2}>-</button>

</div>

)

}

}

## **8. react-redux**

### **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/><hr/><Counter2/></Provider>,document.getElementById('root'));

### **8.2 Counter.js**

src/components/Counter.js

**import** React, { Component } **from** 'react';**import** actions **from** '../store/actions/counter1';**import** {connect} **from** '../react-redux'**class** **Counter** **extends** **Component** {

**constructor**(props) {

**super**(props);

}

render() {

**return** (

<div>

<p>{this.props.value}</p>

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

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

</div>

)

}

}

**let** mapStateToProps = state=>({value:state.counter});**export** **default** connect(

mapStateToProps,

actions

)(Counter)

### **8.3 react-redux\index.js**

src\react-redux\index.js [react-redux](https://github.com/reduxjs/react-redux/blob/master/src/index.js)

**import** Provider **from** './Provider';**import** connect **from** './connect';**export** {

Provider,

connect

}

### **8.4 react-redux\Provider.js**

src\react-redux\Provider.js [Provider.js](https://github.com/reduxjs/react-redux/blob/master/src/components/Provider.js)

**import** React, { Component } **from** 'react'**import** PropTypes **from** 'prop-types'**import** { ReactReduxContext } **from** './Context'**export** **default** **class** **Provider** **extends** **Component** {

**static** propTypes = {

store: PropTypes.shape({

subscribe: PropTypes.func.isRequired,

dispatch: PropTypes.func.isRequired,

getState: PropTypes.func.isRequired

}),

children: PropTypes.any

}

**constructor**(props) {

**super**(props)

}

render() {

**return** (

<ReactReduxContext.Provider value={{store:this.props.store}}>

{this.props.children}

</ReactReduxContext.Provider>

)

}

}

### **8.5 react-redux\connect.js**

src\react-redux\connect.js [connect.js](https://github.com/reduxjs/react-redux/blob/master/src/connect/connect.js)

**import** React **from** "react";**import** { bindActionCreators } **from** "../redux";**import** { ReactReduxContext } **from** "./Context";**export** **default** **function**(mapStateToProps, mapDispatchToProps) {

**return** **function** **wrapWithConnect**(WrappedComponent) {

**return** **class** **extends** **React**.**Component** {

**static** contextType = ReactReduxContext;

**constructor**(props, context) {

**super**(props);

**this**.state = mapStateToProps(context.store.getState());

}

componentDidMount() {

**this**.unsubscribe = **this**.context.store.subscribe(() =>

**this**.setState(mapStateToProps(**this**.context.store.getState()))

);

}

shouldComponentUpdate() {

**if** (**this**.state === mapStateToProps(**this**.context.store.getState())) {

**return** false;

}

**return** true;

}

componentWillUnmount() {

**this**.unsubscribe();

}

render() {

**let** actions = bindActionCreators(

mapDispatchToProps,

**this**.context.store.dispatch

);

**return** <WrappedComponent {...this.state} {...actions} />;

}

};

};

}

### **8.6 react-redux\Context.js**

src\react-redux\Context.js [Context.js](https://github.com/reduxjs/react-redux/blob/master/src/components/Context.js)

**import** React **from** 'react'

**export** **const** ReactReduxContext = React.createContext(null)

**export** **default** ReactReduxContext

## **9. react-redux-old**

### **9.1 react-redux-old\Provider.js**

src\react-redux-old\Provider.js

**import** React, { Component } **from** 'react'**import** PropTypes **from** 'prop-types'**export** **default** **class** **Provider** **extends** **Component** {

**static** propTypes = {

store: PropTypes.shape({

subscribe: PropTypes.func.isRequired,

dispatch: PropTypes.func.isRequired,

getState: PropTypes.func.isRequired

}),

children: PropTypes.any

}

**constructor**(props) {

**super**(props);

}

**static** childContextTypes = {

store: PropTypes.shape({

subscribe: PropTypes.func.isRequired,

dispatch: PropTypes.func.isRequired,

getState: PropTypes.func.isRequired

})

}

getChildContext(){

**return** {store:**this**.props.store};

}

render() {

**return** **this**.props.children

}

}

### **9.2 react-redux-old\connect.js**

src\react-redux-old\connect.js

* connect方法将检查mapStateToProps方法返回的props对象是否变更以决定是否需要更新组件。为了提高这个检查变更的性能，connect方法基于Immutabe状态对象进行改进，使用浅引用相等性检查来探测变更。这意味着对对象或数组的直接变更将无法被探测，导致组件无法更新。

**import** React **from** 'react';**import** {bindActionCreators} **from** '../redux';**import** PropTypes **from** 'prop-types';**export** **default** **function**(mapStateToProps,mapDispatchToProps){

**return** **function** **wrapWithConnect**(WrappedComponent) {

**return** **class** **extends** **React**.**Component**{

**constructor**(props,context){

**super**(props);

**this**.state = mapStateToProps(context.store.getState());

}

**static** contextTypes = {

store: PropTypes.shape({

subscribe: PropTypes.func.isRequired,

dispatch: PropTypes.func.isRequired,

getState: PropTypes.func.isRequired

})

}

componentDidMount(){

**this**.unsubscribe = **this**.context.store.subscribe(

()=>**this**.setState(mapStateToProps(**this**.context.store.getState()))

);

}

componentWillUnmount(){

**this**.unsubscribe();

}

render(){

**let** actions = bindActionCreators(mapDispatchToProps,**this**.context.store.dispatch);

**return** <WrappedComponent {...this.state} {...actions}/>

}

}

}

}