// 安装完成以后自带的一个createStore方法

import { createStore } from "redux";

// createStore 接收一个函数  可以在里面设置初始值  先当于vuex的state

const store = createStore(() => {

    const state = 0;

    return state;

});

export default store;

// 安装完成以后这个store就会有以下方法

// dispatch

// store.dispatch()可以改变值

// subscribe

// 用来重新rander

// store.subscribe(() => {

    // this.forceUpdate(); 强制刷新页面

    // this.setState({});

// });

// getState

// store.getState()可以获取到state里面的值

// replaceReducer

// Symbol

# React Redux

**class MyComp extends Component {**

**// content...**

**}**

**const Comp = connect(...args)(MyComp);**

**connect() 接**收四个参数，它们分别是 **mapStateToProps**，**mapDispatchToProps**，**mergeProps**和**option**

**第一个参数是一个函数 这个函数能够将store中的数据作为props绑定到组件上所以这个函数主要参数就是 Redux 的store还有组件自己的props**

**第二个参数是可以将action作为props绑定到组件上 由于mapDispatchToProps方法返回了increase和decrease属性的对象 这两个属性也会成为组件的props**

**第三个参数是用来做marge的 一般不需要传**

**第四个参数直接忽略**

<https://www.cnblogs.com/passkey/p/9910760.html>

**redux中间件种类和各自的作用**

**redux-thunk 中间件  改造store.dispatch,解决异步操作**

### 它的引入是为了解决action里面无法进行异步操作的问题。它可以使action返回的是函数。

**redux-promise 中间件**

### 这个的引入是解决action里面异步写法过于复杂的问题，使得store.dispatch方法可以接受 Promise 对象作为参数

**redux-logger中间件 日志中间件**

### 能打印出先前的state 改变后的state和action给开发带来一些便利

### ****redux-saga中间件****

### 它本身是基于es6的generator来实现的，它的实现需要一个监视saga我们可以用takeEvery或者takeLatest。takeEvery是注册监视action不用等着异步完成就可以更新视图同步性比较高。然后takeLatest是防止重复提交用的。然后就是call和fork了call是阻塞的，fork是非阻塞的。

### thunk 和 saga的区别，他们都能实现异步，不同点在于thunk是强耦合的。必须要写在action里面这样的话不便于维护，所以现在更多的是用saga。

### https://www.jianshu.com/p/b17d8bec13f3

# Redux框架之applyMiddleware

Middleware是无需引用大量代码第三方库实现异步 actions。这种方式可以让你像 dispatch 一般的 actions 那样 dispatch 异步 actions

每个 middleware 接受 Store 的 dispatch 和 getState 函数作为命名参数，并返回一个函数

import logger from "redux-logger";

import thunk from "redux-thunk";

applyMiddleware(logger, thunk)

https://www.jianshu.com/p/caff049d576c