# redux

## 了解 redux 么，说一下 redux

* redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 **view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据**
* flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。
* 主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰
* 新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们

## redux 有什么缺点

* 一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。
* 当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

## Redux的三大核心原则？

* 单一数据源：整个应用的state被存储在reducer中，并且这个reducer只存在于唯一一个store
* state是只读的：唯一能改变state的就是action,action是一个用于描述已发生事件的对象,通过store中的dispatch方法来发送action，store.dispatch(action)
* 使用纯函数（reducer）来执行修改:reducer是一个纯函数，它接受先前的state，action，并且返回一个新的state

## 为什么需要redux中间件？

默认情况下，redux只能处理同步数据流。但是实际项目开发中，状态的更新、获取、通常是使用异步操作来实现的。

## redux中的combineReducers是什么？有什么作用？

他其实就是redux给我们提供的一个函数，可以方便的让我们对多个reducer进行合并，

combineReducers() 接收一个对象，它的值对应不同的 reducer 函数，这些reducer 函数会被合并为一个。然后被引入到 store 中，放到 createStore() 中；

取值时，注意此时总的state是一个多层对象，如果需要器中某个页面（单元）的某个state值，则需要通过state.reducer名.字段名才行；

## 在Redux中，什么是Action？

在Redux中，Action是一个描述发生了什么事情的普通对象。

它是Redux中的一个核心概念，用于将数据从应用程序发送到store。

一个action对象通常包含一个type字段，用于指示将要执行的操作类型，以及可选的payload字段，用于携带相关的数据。

当应用程序中的某个事件发生时，我们通过调用一个action creator函数来创建一个action对象，然后将其分发到Redux store中。

Reducers可以根据action的类型来更新store中的状态。 通过使用actions，我们可以明确地追踪和记录应用程序中发生的各种事件，以及相应的状态变化。