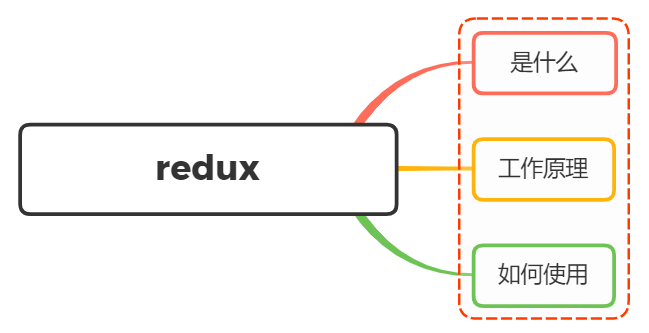
# 说说你对Redux的理解？其工作原理？



## 一、是什么

React是用于构建用户界面的，帮助我们解决渲染DOM的过程

而在整个应用中会存在很多个组件，每个组件的state是由自身进行管理，包括组件定义自身的state、组件之间的通信通过props传递、使用Context实现数据共享

如果让每个组件都存储自身相关的状态，理论上来讲不会影响应用的运行，但在开发及后续维护阶段，我们将花费大量精力去查询状态的变化过程

这种情况下，如果将所有的状态进行集中管理，当需要更新状态的时候，仅需要对这个管理集中处理，而不用去关心状态是如何分发到每一个组件内部的

redux就是一个实现上述集中管理的容器，遵循三大基本原则：

* 单一数据源
* state 是只读的
* 使用纯函数来执行修改

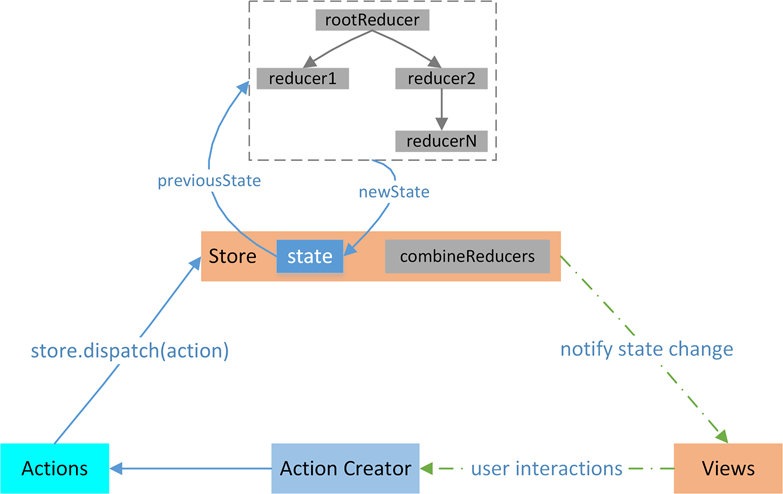
注意的是，redux并不是只应用在react中，还与其他界面库一起使用，如Vue

## 二、工作原理

redux要求我们把数据都放在 store公共存储空间

一个组件改变了 store 里的数据内容，其他组件就能感知到 store的变化，再来取数据，从而间接的实现了这些数据传递的功能

工作流程图如下所示：



根据流程图，可以想象，React Components 是借书的用户， Action Creactor 是借书时说的话(借什么书)， Store 是图书馆管理员，Reducer 是记录本(借什么书，还什么书，在哪儿，需要查一下)， state 是书籍信息

整个流程就是借书的用户需要先存在，然后需要借书，需要一句话来描述借什么书，图书馆管理员听到后需要查一下记录本，了解图书的位置，最后图书馆管理员会把这本书给到这个借书人

转换为代码是，React Components 需要获取一些数据, 然后它就告知 Store 需要获取数据，这就是就是 Action Creactor , Store 接收到之后去 Reducer 查一下， Reducer 会告诉 Store 应该给这个组件什么数据

## 三、如何使用

创建一个store的公共数据区域

import { createStore } from 'redux' // 引入一个第三方的方法  
const store = createStore() // 创建数据的公共存储区域（管理员）

还需要创建一个记录本去辅助管理数据，也就是reduecer，本质就是一个函数，接收两个参数state，action，返回state

// 设置默认值  
const initialState = {  
 counter: 0  
}  
  
const reducer = (state = initialState, action) => {  
}

然后就可以将记录本传递给store，两者建立连接。如下：

const store = createStore(reducer)

如果想要获取store里面的数据，则通过store.getState()来获取当前state

console.log(store.getState());

下面再看看如何更改store里面数据，是通过dispatch来派发action，通常action中都会有type属性，也可以携带其他的数据

store.dispatch({  
 type: "INCREMENT"  
})  
  
store.dispath({  
 type: "DECREMENT"  
})  
  
store.dispatch({  
 type: "ADD\_NUMBER",  
 number: 5  
})

下面再来看看修改reducer中的处理逻辑：

const reducer = (state = initialState, action) => {  
 switch (action.type) {  
 case "INCREMENT":  
 return {...state, counter: state.counter + 1};  
 case "DECREMENT":  
 return {...state, counter: state.counter - 1};  
 case "ADD\_NUMBER":  
 return {...state, counter: state.counter + action.number}  
 default:   
 return state;  
 }  
}

注意，reducer是一个纯函数，不需要直接修改state

这样派发action之后，既可以通过store.subscribe监听store的变化，如下：

store.subscribe(() => {  
 console.log(store.getState());  
})

在React项目中，会搭配react-redux进行使用

完整代码如下：

const redux = require('redux');  
  
const initialState = {  
 counter: 0  
}  
  
// 创建reducer  
const reducer = (state = initialState, action) => {  
 switch (action.type) {  
 case "INCREMENT":  
 return {...state, counter: state.counter + 1};  
 case "DECREMENT":  
 return {...state, counter: state.counter - 1};  
 case "ADD\_NUMBER":  
 return {...state, counter: state.counter + action.number}  
 default:   
 return state;  
 }  
}  
  
// 根据reducer创建store  
const store = redux.createStore(reducer);  
  
store.subscribe(() => {  
 console.log(store.getState());  
})  
  
// 修改store中的state  
store.dispatch({  
 type: "INCREMENT"  
})  
// console.log(store.getState());  
  
store.dispatch({  
 type: "DECREMENT"  
})  
// console.log(store.getState());  
  
store.dispatch({  
 type: "ADD\_NUMBER",  
 number: 5  
})  
// console.log(store.getState());

### 小结

* createStore可以帮助创建 store
* store.dispatch 帮助派发 action , action 会传递给 store
* store.getState 这个方法可以帮助获取 store 里边所有的数据内容
* store.subscrible 方法订阅 store 的改变，只要 store 发生改变， store.subscrible 这个函数接收的这个回调函数就会被执行

## 参考文献

* https://cn.redux.js.org/docs/introduction/
* https://www.redux.org.cn/docs/basics/Actions.html
* https://lulujianglab.com/posts/大白话解析 Redux 、 redux-thunk 、redux-saga 和 react-redux