手写一个十分精简的 react-redux

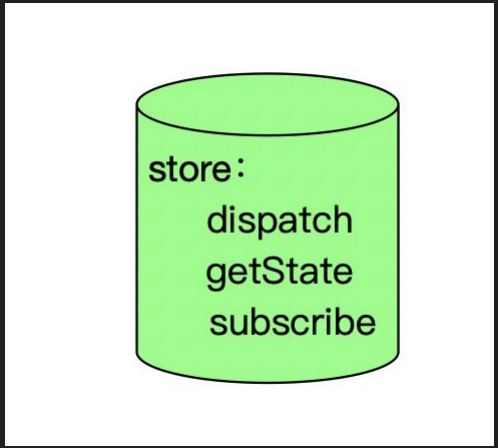
从数据流说起：为什么react需要redux？

react只是一个构建页面的工具，而非一个框架。react自身并不带有数据流处理方案，其自身只有一个单向数据流，数据只能从祖先组件流向后代组件，兄弟组件之间的通信只能通过状态提升实现，这在项目中显然是没法忍受的。

于是，redux就应运而生

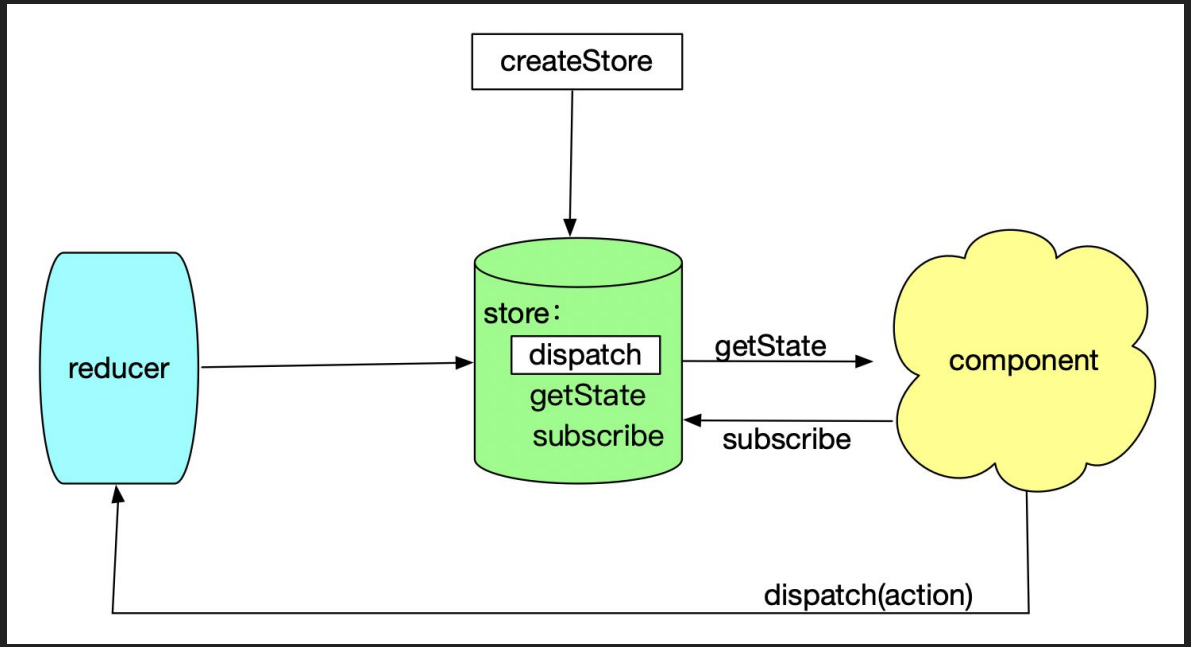
redux解决了哪些问题

redux将数据统一保存到store中，并规定数据的修改只能通过特定的操作完成，这使得数据的不会被随意修改，修改可追溯，方便预测，并且由于有了统一的数据中心store，组件之间的通信可以顺利完成。



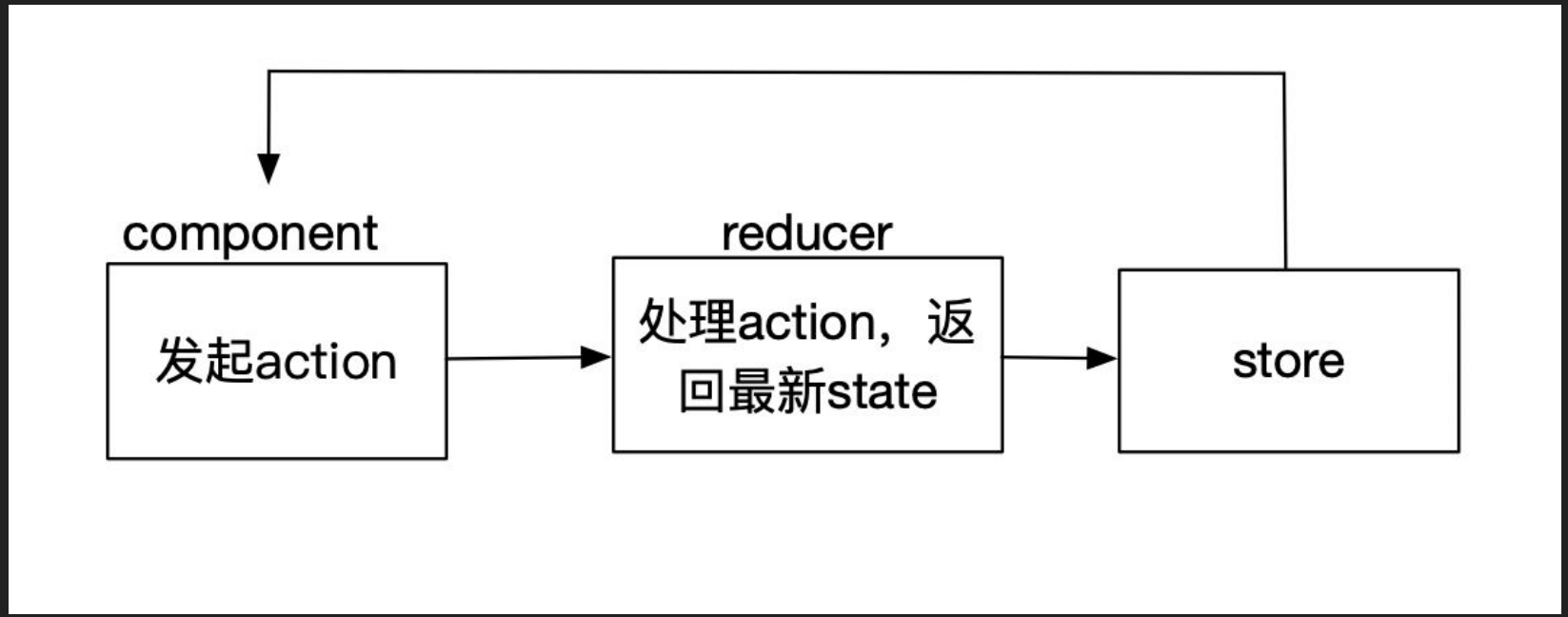
数据现在存储于store中，页面仅作为展示容器
或者说，页面由数据驱动展示

有了redux后页面的工作方式



组件通过getState()从store中获取数据，同时通过subscribe()方法订阅store的变化，这样store发生变化后组件能及时感知到。
组件对数据的修改通过dispatch(action)来实现；
action是一个js对象，描述了需要对state做的修改；
reducer会接收到action，根据action的类型及数据处理后返回最新的state并更新到store中存储

数据流修改的流程：



如何在react组件中使用redux

根据store的使用方法，我们要想在react组件中使用redux，首先，需要能获取到store，这样组件才能获取到store中存储的数据；其次，组件要订阅到store中，这样store中数据发生变化组件可以实时更新。

1. 组件要能获取到store，或者能获取到store中存储的state
2. 组件要订阅到store上，state更新时组件可以实时更新
3. 要有reducer处理方法

react-redux已经帮我们完成了上面这些事情，Provider会将store放入到上下文中供子组件使用，此外根组件包裹在Provider中，然后对于要使用store中数据的组件，需要用connect()方法处理下：

```
class App extends Component{
        ...
}
export default connect()(App)
```

```
<Provider store={store}>
        <App />
</Provider>
```

动手写一个react-redux

先看Provider：

Provider主要功能时将store放入上下文。实现起来比较简单：

```
const Context = React.createContext(null); // 公共上下文，所有组件共享

class Provider extends Component{

    render(){
        return (
            <Context.Provider store={store}>
                    {this.props.children}
            </Context.Provider>
        )
    }
}
```

connect方法

参看：

手写react-redux：1.基础版.md

手写react-redux：2.进阶版.md