什么是状态管理

React.js 是一个 ui 渲染库，其核心概念就是 data -> UI 的逻辑。在复杂的业务逻辑下，组件的状态，需要被其他组件所共享，由此发展出了状态管理，在 react 生态中，有两个状态管理方案是最有名的，即 redux 和 mobx。

Redux

Redux 的核心概念

介绍 action、dispatch、reducer、store 的概念

action

action 是一种用户端发出的动作，可以是 button 的 onClick，也可能是组件首次渲染发出的请求数据动作，等等

dispatch

dispatch 是一个方法，他的作用就是把 action 发送到 store 中，同时，会经历所有的 middleware，来为 redux 提供更多的扩展功能

reducer

在 redux 中有一个强约束，所有对 store 的修改比如是经过 reducer 来完成的。每一次经历了 reducer 改动的 store，必须是 immutable 的改动，基于这个约束，我们就可以用"时间旅行"的方式来进行调试

store

store 是所有状态的中心，并且提供 api 给外部调用，以此完成中心化的状态管理过程

Redux Demo

使用 Redux 进行状态管理，进行一个 Demo 项目。通过这个项目，了解如何借用 redux 来实现产品的业务逻辑。

Redux 原理解析

讲解 Redux 的原理、模式

Mobx

Mobx 的核心概念

介绍 observable、action、computed 的概念

observable

在 mobx 中，observable 修饰过的对象，就可以被 mobx 监控，一旦该对象有任何变化，都会通知所有他的观察者。

action

所有对 observable 对象改变方法都应该被 action 修饰，mobx 在对 observable 对象修改时，有机会增加日志记录、批量更新等优化

computed

computed 修饰过的对象是一种计算属性，这也是 mobx 的优势，其概念就对应 vue.js 中的 computed，一旦 computed 内部的依赖更新了，computed 就会被重新计算

Mobx Demo

使用 Mobx 进行状态管理，进行一个 Demo 项目。借用 mobx 进行状态管理，来实践一个产品的业务逻辑。

Mobx 原理解析

React vs Mobx 工程实践对比

经过了两个 Demo 项目，通过对两个 Demo 项目进行对比，来展示两种状态方案的优劣。