Taming the State

In this lesson you learn why you would consider using a third-party state management library such as Redux or MobX.

When do you need state management?

Previous chapters have shown you that state management can be a crucial topic in larger applications, as React and a lot of other SPA frameworks struggle with it. As applications get more complex, the big challenge in web applications is to tame and control the state.

How React Makes State Management simple

Compared to other solutions, React has already taken a big step forward. A unidirectional data flow and a simple API to manage state in components is indispensable. These concepts make it easier to reason about your state and your state changes. It also makes it easier to reason about it on a component level and on an application level to a certain degree.

It is possible to introduce bugs by operating on stale state when using an object over a function in `setState()`. We lift state around to share or hide necessary state across components. Sometimes a component needs to lift up state, because its sibling component depends on it. Perhaps the component is far away in the component tree, so the stated needs to be shared across the whole component tree. Components are more involved in state management, as the main responsibility of components is representing the UI.

Because of this, there are standalone solutions to take care of state management. Libraries like Redux or MobX are both feasible solutions in a React application. They come with extensions, react-redux and mobx-react, to integrate them into the React view layer. Redux and MobX are outside of the scope of this course, but I encourage you to study the different ways to handle scaling state management as your React applications become more complex.

Further Reading:

Further Reading:

- Read about external state management and how to learn it
- Check out my second book about state management in React called Taming the State in React