State Management in React Js

What is React State Management?

In React, State refers to the data or variable that determine a component's

Behavior and render information to the user. The state is managed within a

Component, and change can trigger re-render of the component. React provides

A setState () method to update the state, which cause a component to re-render

With the new state. In addition to managing state within a single component, React

Also provides a way to share state between components through context and state management libraries like Redux.

Why we need State Management?

As we all know React application is made with components like Navbar, Sidebar,

Buttons, Forms and Tags etc. Components in a web app are arranged in such a way that data flow from the *topmost* level to the *least* level. And In a case, I want to pass

Data from the top level component to the least level. I have to pass this data from top level to bottom one by one from all components and this process is known as Prop drilling and it is very time taking process and make our app more complex. That why

We need state management to handle such things.

Here is a simple example of an e-commerce application, in which the status of multiple components will change when purchasing a product.

- Add that product to the shopping list
- Add product to customer history
- trigger count of purchased products

If developers do not have scalability in mind then it is really hard to find out what is happening when something goes wrong. This is why you need state management in your application.

Different ways for State management In React

There are several ways to managing State in React. But some popular ways are

Following.

- Context API
- Redux
- Mobx
- React Hooks

Pros and Cons of State management In React

There are following Pros of using State managing in React Application.

1. **Improved Organization:** State management libraries like Redux and Mbox provide a centralized place to store application state, making it easier to keep track and update the application's state.
2. **Better Scalability:** State management libraries make it easier to handle the state of large applications, as they provide a way to manage the state of multiple components and share data between them.
3. **Improved Performance:** State management libraries provide ways to handle the state in efficient way, such as avoiding unnecessary re-rendering of the component.

There are following Cons of using State managing in React Application.

1. **Additional Complexity:** Using a state management library can add complexity to an application, as developer needs to learn and implement the concepts and principle of libraries.
2. **Overhead:** Implementing state management can add a layer of abstraction, which can increase the complexity of the codebase and make it harder to debug.
3. **Boilerplate Code:** Some libraries requires much boilerplate code to set up and manage state, and making it difficult to understand the codebase for new react-developers.

Conclusion

The above mentioned ways to manage state management each of these approaches have their own pros and cons. React built-in state management is straightforward and easy to use but can become unwieldy in large-scale application. The Context API is simpler alternative, but it may not be suitable for large-scale application.

In conclusion the approach you choose will be depend on application need and your own preferences. It always be good idea to consider Pros and Cons of each approach before making decision.

Reference Links

- https://www.loginradius.com/blog/engineering/react-state-management/
- https://solguruz.com/blog/state-management-in-react/
- https://www.etatvasoft.com/blog/react-state-management/
- https://dev.to/boasbabs/the-analogy-of-state-management-in-react-30hd

- https://www.freecodecamp.org/news/state-management-in-react-props-vs-context-api/
- https://www.youtube.com/@academind