React State Management

Something about prop drilling...

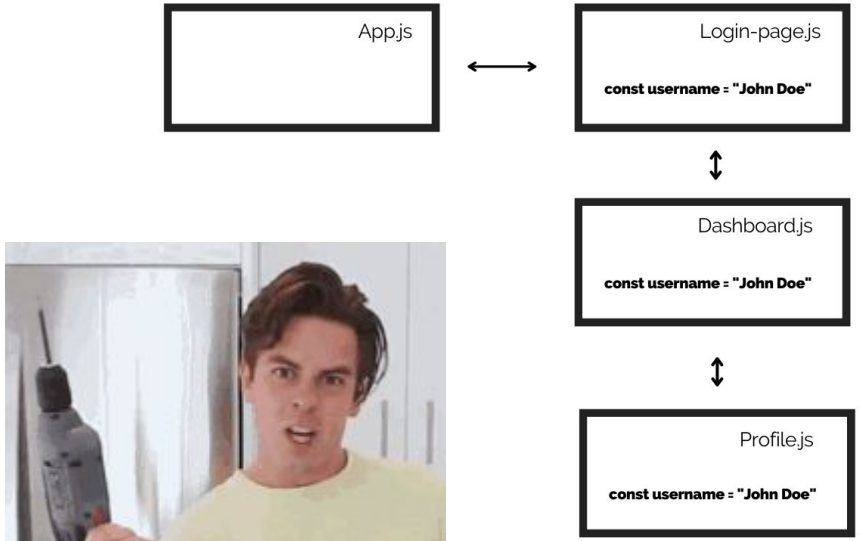
The Problem With State

State Management

State is great! But what about when we have a large application that passes data to different components up and down the component tree?

Props Drilling

We have to keep passing our property or data down the component tree until we reach the desired component. It can get confusing in larger applications and does not scale well.



State Management

State Management

We can solve this by having a library handle our state for us so it is available in our entire application without passing props.

- Pros
  - Separation of Concerns
  - Cleaner
  - Easier to manage
- Cons
  - Another thing to learn ugh!
  - More set up and planning in the beginning

2 Popular Way to Manage State

1. Redux Library
    a. A third party library built to help manage state in any application, but mostly used in React
    b. Companies that use this are Instagram, Amazon, Robinhood
2. React Context
    a. Built into React by Facebook, the creators of React. It is better for small to midsize applications

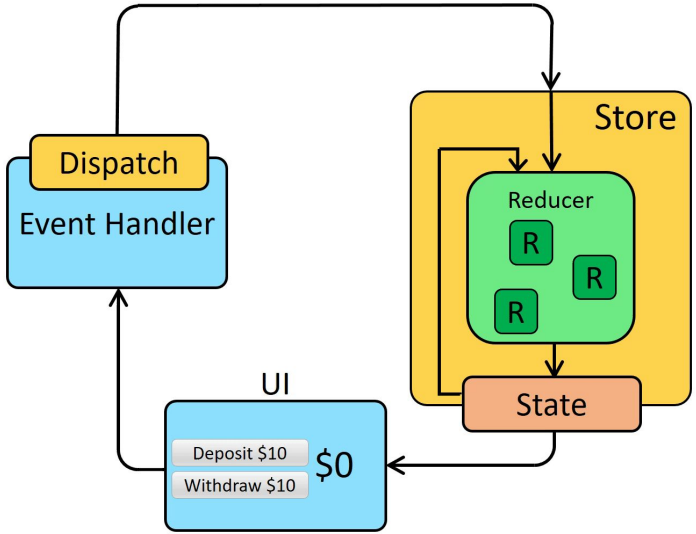
Redux

Introduction to Redux

Redux is a predictable state management library for JavaScript applications, commonly used with frameworks like React. It helps manage application state in a centralized and predictable manner.

But what is Redux?

Redux is a state management pattern and library that provides a predictable and centralized way to manage application state. It follows the principles of a single source of truth and immutability.



Core Concepts of Redux

Redux revolves around three core concepts: actions, reducers, and the store.

1. Actions: Plain JavaScript objects that describe a change in the application state.
2. Reducers: Pure functions that define how the state should change based on actions.
3. Store: A centralized container that holds the state of the application and provides methods for accessing and updating the state.

Setting Up Redux

To use Redux in an application, you need to install the Redux library, create reducers to handle state changes, and set up the store.

Simple as “npm install redux react-redux”

Connecting Redux with React

Redux is commonly used with React to manage state in React applications. You can connect React components to the Redux store using the connect function or hooks like useSelector and useDispatch.

Benefits of Using Redux

Redux provides several benefits, including centralized state management, predictable state changes, improved debugging, and easier testability. It also enables better collaboration among team members.

- Example
  - Improved code organization
  - Easier state tracking
  - Simplified debugging and testing.

React Context

Introduction to React Context

React Context is a feature in React that allows you to share data across the component tree without explicitly passing props at each level. It provides a way to manage global state and facilitates communication between components.

What is React Context?

React Context is a mechanism for sharing data between components without the need for prop drilling. It creates a "context" that holds the data and provides it to all child components that consume the context.

Context Provider and Consumer

React Context consists of two primary components: Provider and Consumer.

- Provider component allows you to define the data to be shared
- Consumer component consumes the data within its subtree.

Setting Up Context

To use React Context, you need to create a context using React.createContext and wrap the components that need access to the context with the Provider component. This establishes the context's scope.

Consuming Context with Consumer

Components that need to access the data from the context can use the Consumer component. The Consumer renders the content based on the context value provided by the nearest Provider in the component hierarchy.

Context with useContext Hook

React provides a useContext hook that simplifies the usage of Context. It allows functional components to consume context values directly without using the Consumer component.

Benefits of React Context

React Context offers benefits such as eliminating prop drilling, providing a centralized way to manage global state, and simplifying component communication. It improves code readability and maintainability.

- Example:
  - Reduced complexity
  - Improved code organization
  - Enhanced developer experience.

Conclusion

Recap: React Context allows data sharing and state management across the component tree without prop drilling.

Key Takeaway: Utilizing React Context in React applications eliminates the need for excessive prop passing, provides a centralized approach to manage shared data, and simplifies component communication, leading to improved code readability and maintainability.