React And Redux

Context API

Context provides a way to pass arguments to the component tree without passing props at every level. In the React application data is passed through props from parent to child at every level but such use of props can be cumbersome for some types of props that are required by the application. So, context solves this problem out and helps to avoid the passing of props manually at every level in the application.

When to use context?
The problem of **prop drilling** *(passing props at every level)* is solved by context. Context helps to access your data everywhere. Bit it fails when it comes to global state management. Context allows you to get data from store anywhere in the component tree.

High Order Components

High order components are JavaScript functions used to add additional functionality to an existing component. These functions are pure, meaning they receive data and return values according to that data. If the data changes, the high order functions are re-enabled with different data inputs. If we want to update our returning part, we do not need to change the HOC. We need to change the data that uses our function.
The High Order Component (HOC) **wraps** around the "normal" component and provides additional data input. It actually returns a function and another component that wraps around the source.

Note: High order components are **pure functions** used for different functionalities. As you start using these function you will feel that your app is becoming easier to maintain.

Redux Connect

Redux helps us to connect the components and bind them together to make the code easier to maintain. There are two types of React components based on their connection with redux:
Presentational Components: They are only concerned with how things look alike and are not aware of the Redux state. They get their data through props and trigger callbacks.
Container Components: They are fully aware of how things work and know the functionality of the Redux state. They are created using React Redux and can dispatch and subscribe functions.

What is connect() function?
The Redux React component **connect()** is used as a **container component** to wrap the dispatch functions. A higher-order component is returned by React.
Connect has four arguments:

- mapStateToProps
- mapDispatchToProps
- mergeProps
- options

mapStateToProps: This is used to select the part of data from a store that is needed by the connected component. It has two arguments state and ownProps(optional). It is called every time the state changes and then returns that part of data that a component needs.

mapDispatchToProps: This is to dispatch functions from the store. It is basically used for a state change. It creates functions that are dispatched on call and then further pass them as props to components.

mergeProps: It determines how final props will look-alike for the wrapped component in the application.

Summarising It

Let's summarise what we have learnt in this module:

- Learned about context API.
- Learned about high order components and connect().

Some References:

● Context API
https://reactjs.org/docs/context.html
● Connect()
https://react-redux.js.org/api/connect
● High Order Components
https://reactjs.org/docs/higher-order-components.html