Redux

An Introduction

@manoj_nama

Combining React with Redux

React+Redux

- Two things we need to connect React with Redux
- *Wrapping* our App with a **HOC** called `Provider`
- Provider *listens* for store changes and **notifies** the App (i.e `subscribe`)
- Use the `connect` utility to connect store with the components
- It controls **what** & **how** the component will receive the state data.
- Any component attached with connect utility has access to the store's state

Provider HOC

```
import { Provider } from 'react-redux';
import { render } from 'react-dom';
import App from './our/root/App';
import store from './our/create.store';

render(
    <Provider store={store}>
        <App />
    </Provider>,
    rootEl
);
```

Connect

```
import { connect } from 'react-redux';

class App extends Component {
    ...
}

const mapStateToProps = state => state;

const AppContainer = connect(mapStateToProps)(App);
export default AppContainer
```

Container Components

- Container component are the one which have access to the `Redux` state, or Are having any business logic
- They only pass the required data to child components
- Container components are the ones with `connect` applied
- We need a strategy for how many levels we need to pass data. After a couple of levels, introduce a `connect`ed component

Presentational Components

- Components which are there **only** to *present* data to the user.
- They **Don’t** have any *business logic*, or any *complex computations*
- They get the data as *via* **props** and just display them.
- They are sometimes called `PURE` components, as they are just simple functions

PURE components

```
const Row = ({data, from, props}) => {
    return (
        <p>This is a row. {data}</p>
    )
}
```

Connect Functions

- Connect function takes some functions as arguments
- These functions determine how we want to subscribe to store changes
- There are multiple scenarios where these come in handy
- `mapStateToProps`
- `mapDispatchToProps`
- Existence of each of these functions will tell what the component will receive

Connect

```
// Don’t Subscribe to store changes
// inject only dispatch to App component

const Root = connect()(App);
```

Connect

```
// Subscribe to store changes
// inject entire state and dispatch to App
const mapStateToProps = (state) => {
    return state;
}

const Root = connect(mapStateToProps)(App);
```

Connect

```
// Subscribe to store changes
// inject users and dispatch to App
const mapStateToProps = (state) => {
    return state.users;
}

const Root = connect(mapStateToProps)(App);
```

Connect

```
// Don't Subscribe to store changes
// inject fetchUsers action creator to App
const mapDispatchToProps = (dispatch) => {
    return {getUser: (id) => dispatch(fetchUsers(id))};
}

const Root = connect(null, mapDispatchToProps)(App);
```

Connect

```
    // Subscribe to store changes
    // inject users and dispatch to App
    // inject fetchUsers action creator to App
    const mapStateToProps = (state) => {
        return state.users;
    }
    const mapDispatchToProps = (dispatch) => {
        return {getUser: (id) => dispatch(fetchUsers(id))};
    }

const Root = connect(mapStateToProps, mapDispatchToProps)(App);
```