Redux Notes

So, what are the alternatives for managing the state of a React component? **Redux** is one of them.

Why? **Redux solves a problem that might not be clear in the beginning**: it helps giving **each React component** the **exact piece of state** it needs. Redux holds up the **state** within a **single location**.

What's the inconvenient? Redux add another **layer of abstraction** to your application.

Use? React project can be **easily refactored** to include Redux later.

What's reducers?

Function that's 2 parameters (previousState, actions) => new state

Step to setup Redux:

- Have a React component
- Connect your component with Redux
    - Import {connect} from "react-redux"
        - Map the state to props
            - Const mapState = () => ({key: value})
    - Export default connect(mapState)(Component)
- Create the store
    - Import {createStore} from "redux"
    - Const store = createStore(reducer)
- Create the provider
    - Import {Provider} from "react-redux"
- Initialize the state
    - Const exState = {key: value}
- Create a reducer
    - Function reducer (state, action) {return state}
- Create a stateless component
    - Const App = () => ()
        - Next
            - Add tag <Provider store={store}>here any</Provider>
- Render your app as usual with ReactDOM