Revisiting your Knowledge of Redux

Redux is a state container, and in this lesson, we'll take our first step in handling the app's state object with Redux.

Remember the quote from the official docs ?

> Redux is a predictable state container for JavaScript apps.

One key phrase in the above sentence is, **state container**.

Technically, you want the STATE of your application to be managed by REDUX.

This is what makes Redux a state container.

Your React component state still exists. Redux doesn’t take it away.

However, REDUX will efficiently manage your overall APPLICATION STATE. Like a bank vault, it’s got a STORE to do that.

For the simple <App/> component we’ve got here, the state object is simple.

Here it is:

```
{
  tech: “React”
}
```

We need to take this out of the <App /> component state, and have it managed by REDUX.

From an earlier explanation, you should remember the analogy between the BANK VAULT and the REDUX STORE. The Bank Vault keeps money, the Redux store keeps the application state object.

So, what is the first step to refactoring the <App /> component to use Redux?

Yeah, you got that right.

Remove the component state from within <App />.

The Redux STORE will be responsible for managing the App’s state. With that being said, we need to remove the current state object from <App/>

```
import React, { Component } from “react”;

import HelloWorld from “./HelloWorld”;

class App extends Component {
  // the state object has been removed.
  render() {
    return
  }
}

export default App;
```

The solution above is incomplete, but right now, <App/> has NO state.

Please install Redux by running **yarn add redux** from the command line interface (CLI). We need the **redux** package to do anything right.