The Structure of the React Hello World Application

We'll look at the App.js implementation of the React "Hello World!" app. There is a single component which is using a tech prop to display "React".

The React app we'll be working with has been bootstrapped with create-react-app. Thus, the structure of the app is one you're already used to.

You may grab the repo from Github if you want to follow along - which I recommend. Move into the "hello-redux" and run the application using:

```
npm start
```

There's an `index.js` entry file that renders an `< App />` component to the **DOM**. The main **App** component comprises of a certain `< HelloWorld />` component.

This `< HelloWorld />` component that takes in a tech prop, and this prop is responsible for the particular technology displayed to the user.

For example, `< HelloWorld tech="React" />` will yield the following:

Hello World React!

Also, a `< HelloWorld tech="Redux" />` will yield the following.

Now, you get the gist.

Here’s what the App implementation looks like:

src/App.js

```
<svg id="d573c3dd-b80a-4aff-95aa-deaabb7cdddd" data-name="Layer 1" xmlns="http://www.w3.org/2
```

Have a good look at the state object.

There’s just one field, **tech**, in the state object and it is passed down as prop into the **HelloWorld** component as shown below (Line 15 in src/App.js):

< HelloWorld tech={this.state.tech}/>

Don’t worry about the implementation of the **HelloWorld** component - yet. It just takes in a tech prop and applies some fancy CSS. That’s all.

Since this is focused mainly on Redux, I’ll skip the details of the styling.

So, here’s the challenge.

How do we refactor our **App** to use **Redux** ?

How do we take away the state object and have it entirely managed by Redux? Remember that Redux is the **state manager** for your app.

Let’s begin to answer these questions in the next section.