Bringing Everything Together: Index Files

As we begin working on our app, we shall first index all our major actors into their own directories. This is a common Redux practice and will prove helpful in the future.

All important components required to build the more advanced Hello World app have been discussed in isolation in the earlier sections.

Now, let's put everything together and build the app. Excited?

Firstly, let's talk about folder structure.

When you get to a bank, the Cashier likely sits in their own cubicle/office. The Vault is also kept safe in a secure room. For good reasons, things feel a little more organised that way. Everyone in their own space.

The same may be said for Redux.

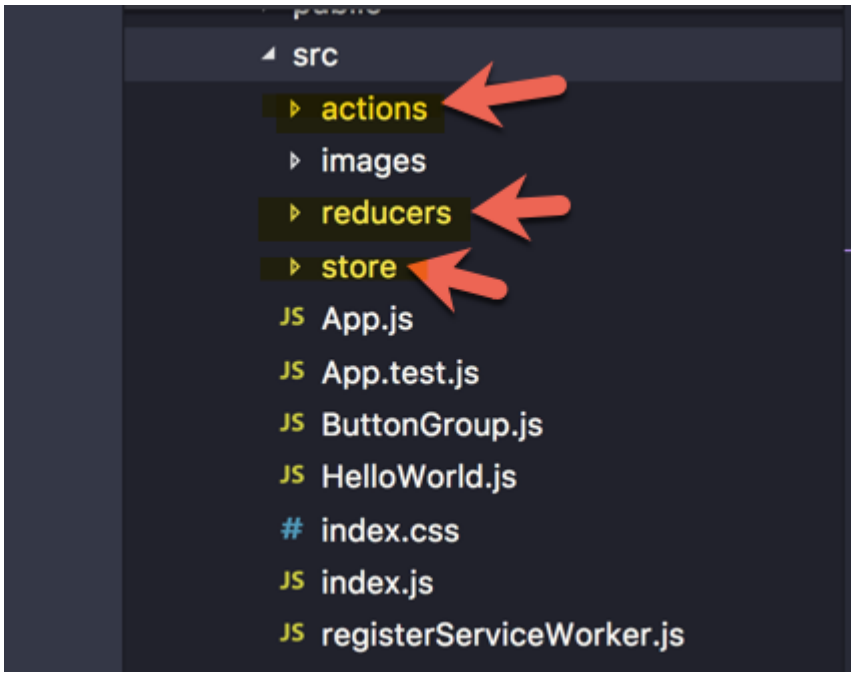
It is a common practice to have the major actors of a redux app live within their own folder/directory. By actors, I mean, the reducer, actions and store.

It is common to create 3 different folders within your app directory, and name it after these actors.

This isn't a must - and inevitably, you decide how you want to structure your project. For big applications though, this is certainly a pretty decent practice.

We'll now refactor the current app directories we have. Create a few new directories/folders. One called **reducers**, another, **store**, and the last one, **actions**.

You should now have a component structure that looks like this:



In each of the folders, create an **index.js** file. This will be the entry point for each of the Redux actors (reducers, store and actions). I call them actors - like movie actors. They are the major components of a Redux system.

Now, we'll refactor the previous app from the lesson *Your First Redux Application*, to use this new directory structure.

store/index.js:

```
import { createStore } from "redux";
import reducer from "../reducers";
const initialState = { tech: "React " };
export const store = createStore(reducer, initialState);
```

store/index.js

This is just like we had before. The only difference is that the store is now created in its own **index.js** file i.e. having separate cubicles/offices for the different Redux actors.

Now, if we need the store anywhere within our app, we can safely import the store as,

```
import store from "./store";
```

With that being said, the **App.js** file for this particular example is slightly different from the former.

App.js:

```
import React, { Component } from "react";
import HelloWorld from "./HelloWorld";
import ButtonGroup from "./ButtonGroup";
import { store } from "./store";
class App extends Component {
  render() {
    return [
      <HelloWorld key={1} tech={store.getState().tech} />,
      <ButtonGroup key={2} technologies={["React", "Elm", "React-redux"]} />
    ];
  }
}

export default App;
```

App.js

What is different?

In line 4, the store is imported from it's own 'cubicle'. Also, there's now a **ButtonGroup** component. We will learn more about this in the next lesson.