Resolving the Initial App Layout: The Main

The second primary component of our app is the Main. Let's work on its basic appearance so that we finally have a skeleton for our app.

Okay, now onto the Main component.

Like before, we'll keep this simple too.

Simply render a simple text within a `\<main>` element.

While developing apps, you want to be sure to build progressively i.e build in bits, and ascertain that the app works.

Below's the `<Main>` component.

```
import React from "react";
import "./Main.css";
const Main = () => {
  return <main className="Main">Main Stuff</main>;
};
export default Main;
```

Again, a corresponding stylesheet, `Main.css` has been imported.

With the rendered elements of both `<Main />` and `<Sidebar />`, there exists the CSS class names, `.Main` and `.Sidebar`.

Since the components are both rendered within `<App />` , the `.Sidebar` and `.Main` classes are children of the parent class, `.App`.

Remember that `.App` is a *flex-container*. Consequently, `.Main` can be made to fill the remaining space in the viewport. Here's the full code:

```
.Main {
  flex: 1 1 0;
  background-color: rgba(25, 25, 27, 1);
  height: 100%;
}
```

That was easy.

And all the code we've written up until this point.

Run it to see how our App looks.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

Not so exciting. Patience. We'll get there.

For now, the basic layout of the application is set. Well done!

Next, we'll handle the state object for the app.