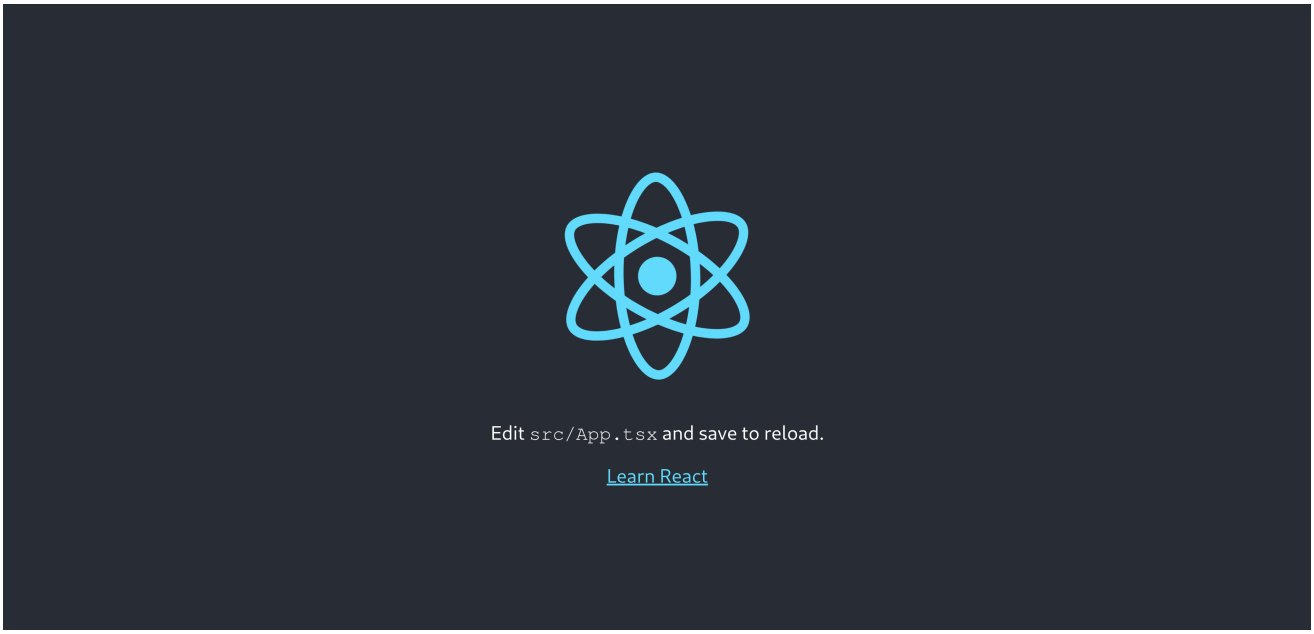
EPQ

To start off with, I create a typescript react app with the command `npx create-react-app artefact --template typescript`. This creates a folder called artefact containing all starting code for the project. This folder contains some boilerplate code, most of which will be removed. running the command `npm start` starts the boilerplate app, which looks like this:



The artefact folder looks like this:

```
artefact
├── node_modules
├── package.json
├── package-lock.json
├── public
├── README.md
├── src
└── tsconfig.json
```

`node_modules` contains the required modules for the project. `Public` contains the `index.html` document where the react app will be injected. And `src` is where the typescript files for the app will go.

I start off by removing all the boilerplate code from the src directory. This leaves me with four files,

```
artefact/src
├── App.css
├── App.tsx
├── index.css
└── index.tsx
```

`App.css` is empty and will be where all the component's CSS will go
`App.tsx` is the typescript file where the react components will go. Currently, it contains:

```
import React from "react";
import "./App.css";

function App() {
    return <div className="App"></div>;
}

export default App;
```

This code returns an empty div element to be injected into the app.

`index.css` contains some extra CSS for index.html, currently it just contains some font styling:

```
body {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
"Roboto",
        "Oxygen", "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans",
        "Helvetica Neue", sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

code {
    font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier
New",
        monospace;
}
```

`index.tsx` contains code that injects the empty app element created in `App.tsx`:

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

ReactDOM.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>,
    document.getElementById("root")
);
```

Sorting

The sorting algorithm page will contain these elements:

- The algorithm visuliser
- A select element to select the selected algorithm
- Buttons to start, stop and reset the visualizer
- Scrolling slide bars to select the number of bars to be sorted, and the sorting speed
- A panel containing metrics on the algorithm, which will display the time taken, number of comparisons and number of swaps
- A place to describe the selected algorithm

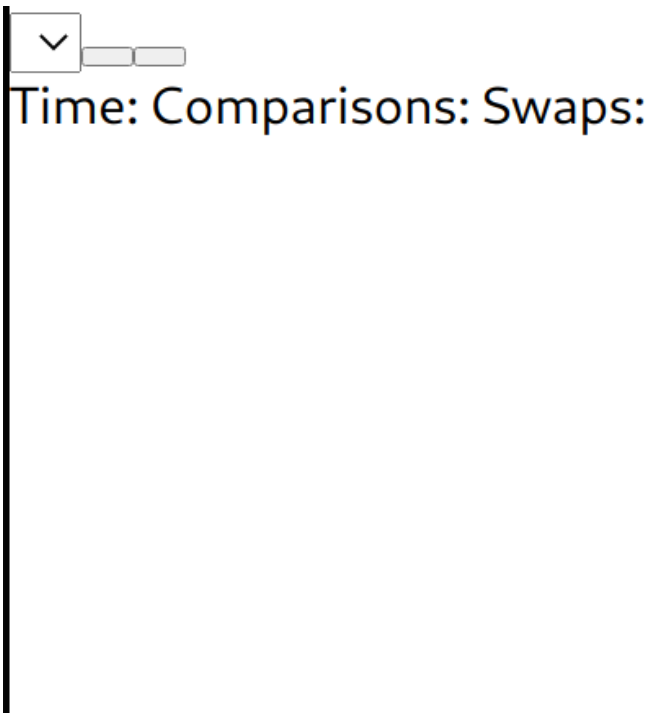
So to start off with, I need to create these components inside `App.tsx`. I have not decided on a final visual design for the application, so I will not style the components any more than required to get them working. Then, once I have decided on how the application will look, I will add the CSS to style the components. The file `App.css` now looks like this:

```
import React from "react";
import "./App.css";

const BarContainer = () => {
    return <div className="barContainer"></div>;
};
const Controlls = () => {
    return (
        <div className="controlls">
            <select className="algorithmSelect" />
            <button className="startstop" />
            <button className="reset" />
        </div>
    );
};
const Metrics = () => {
    return (
        <div className="metrics">
            <Time />
            <Comparisons />
            <Swaps />
        </div>
    );
};
const Time = () => {
    return <span>Time: </span>;
};
const Comparisons = () => {
    return <span>Comparisons: </span>;
};
const Swaps = () => {
    return <span>Swaps: </span>;
```

```
};
const Description = () => {
    return <div className="description"></div>;
};
function App() {
    return (
        <div className="App">
            <BarContainer />
            <Controlls />
            <Metrics />
            <Description />
        </div>
    );
}

export default App;
```

'App.tsx` now contains empty components for the elements of the page. I decided to put the speed, swaps, and comparisons in their own components to make things easier later when I have to update them with their values dynamically.

Now when I run the app it looks like this:



Because there's no styling at all, it is very hard to tell what is going on. Therefore I added some basic CSS styling for the bar container inside `App.css` :

```
.barContainer {
    width: 80vw;
    height: 20rem;
    background: black;
    margin: auto;
}
```

This, plus some added text inside the controls, makes the app now look like this: