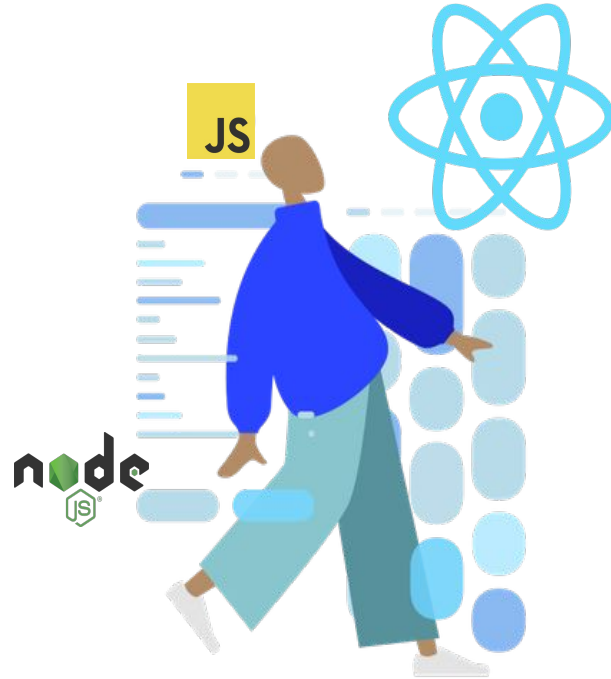


A taste of React.js 🔥



Facilitators:

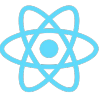
Collins Koech - @collinskoech11

Paul Onteri - @paulonteri



Developer Student Clubs
Catholic University of Eastern Africa

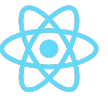
Microsoft Student Partners



Prerequisites

- Know very basic react
- Install npm and node.js
- All of the above can be found here:
https://docs.google.com/presentation/d/1NSV_Hcx2mDhIMyUlhDi0t4McQwpeHjSqmwGMwSetKFM/edit#slide=id.gbe6933289c_0_98



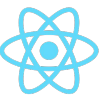


Create a new React application

Run ``npx create-react-app mywebapplication`` to generate a new react app using `create-react-app` named `mywebapplication`.

Proceed to open the application folder named `mywebapplication` in your text editor.

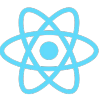




Created a new app and opened it on VSCode

```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
```





Run your project

In your project folder/directory run the terminal command **`npm start`**

Terminal output after running **npm start**.

```
Compiled successfully!
```

```
You can now view mywebapplication in the browser.
```

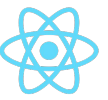
```
Local: http://localhost:3000
```

```
On Your Network: http://192.168.43.195:3000
```

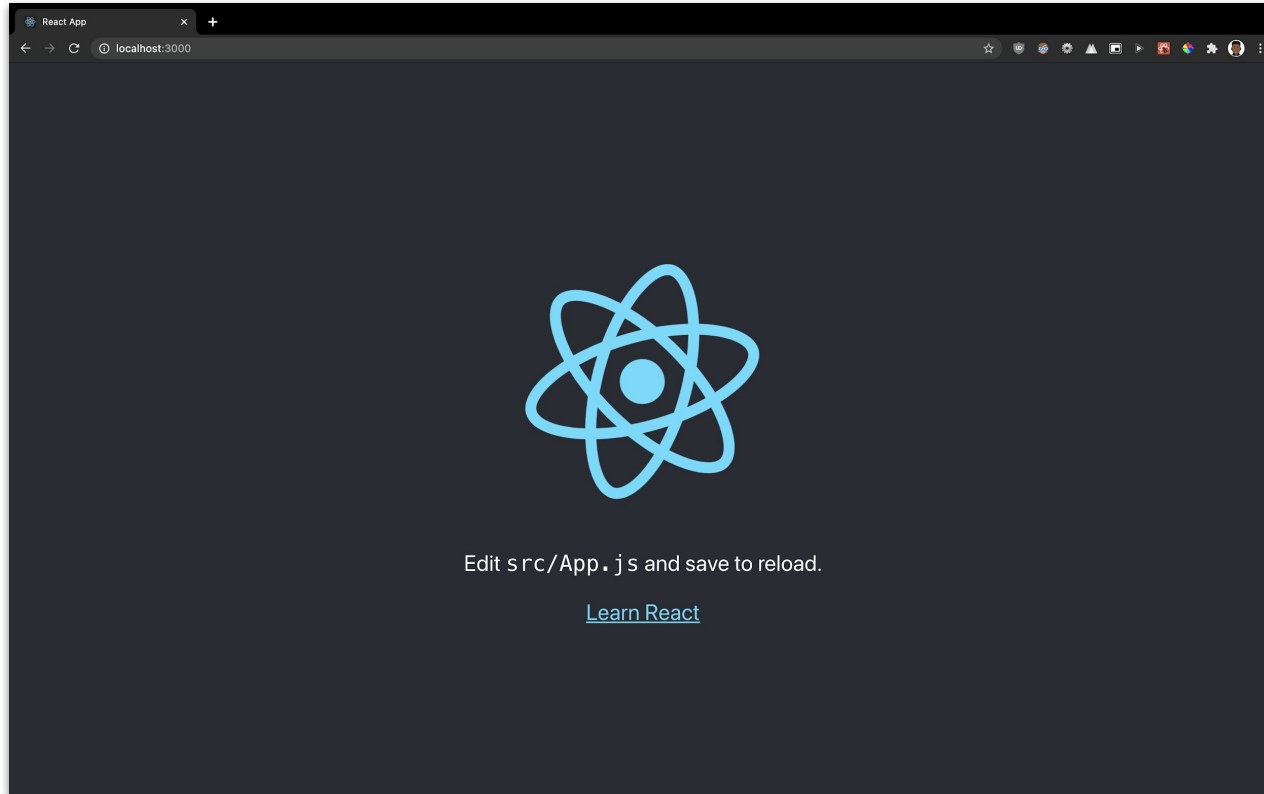
```
Note that the development build is not optimized.
```

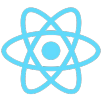
```
To create a production build, use yarn build.
```





Browser output on <http://localhost:3000/> after running npm start





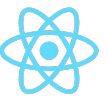
Clear contents of src/App.js

The screenshot shows the Visual Studio Code editor interface. On the left, the Explorer sidebar displays the file structure of a project named 'MYWEBAPPL...'. The 'src' directory is expanded, showing files like 'App.css', 'App.js' (selected), 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'reportWebVitals.js', 'setupTests.js', '.gitignore', 'package.json', 'README.md', and 'yarn.lock'. The main editor area shows the content of 'src > JS App.js > ...'. The file contains a single function definition:

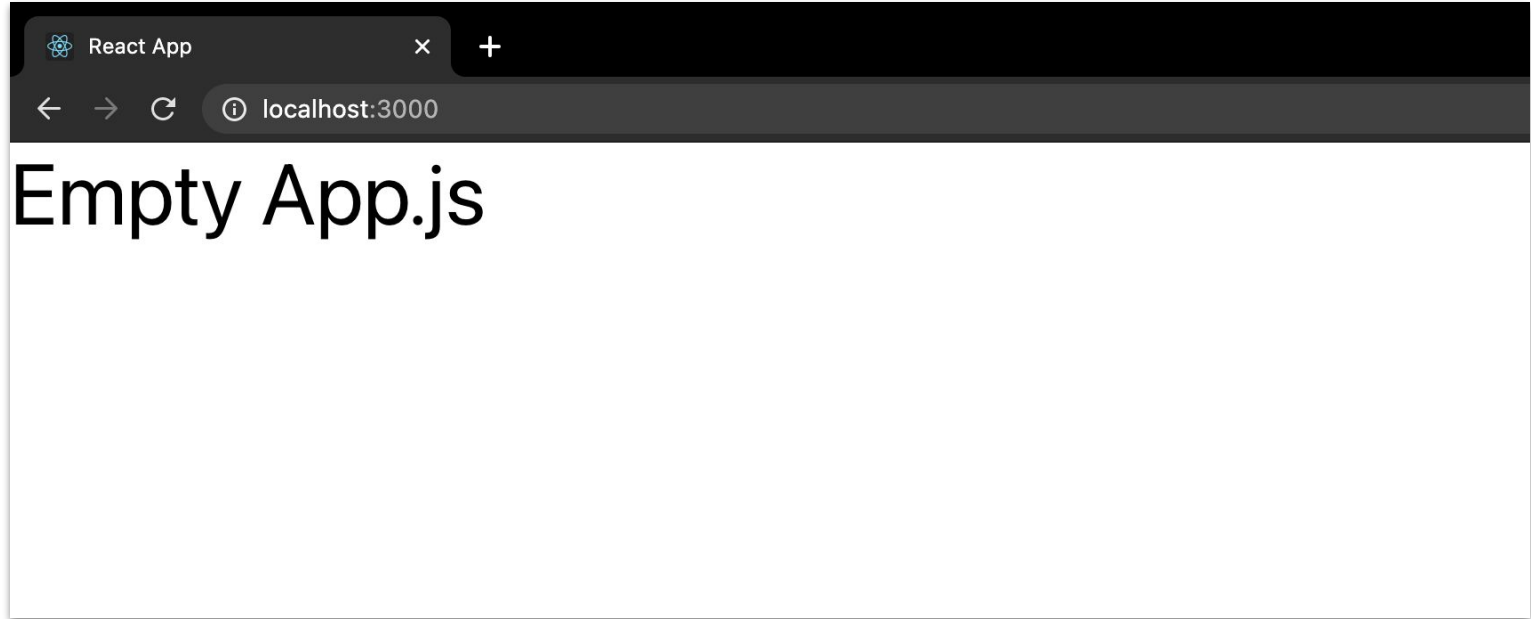
```
1 function App() {  
2   return <div> Empty App.js</div>;  
3 }  
4  
0 references  
5 export default App;  
6
```

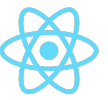
 A search bar at the bottom of the editor area contains the text 'Find related code in mywebapplication'. The status bar at the bottom indicates the current branch is 'master*', the encoding is 'UTF-8', the line ending is 'LF', and various extensions like Babel JavaScript, Go Live, kitematic, ESLint, and Prettier are active.





Bowser output after clearing App.js





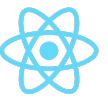
Created a new component in the src folder named **Student**

The screenshot shows the Visual Studio Code interface with the Explorer panel on the left and the Code editor on the right. The Explorer panel shows the project structure with the 'src' folder expanded, and 'Students.js' is highlighted. The Code editor shows the content of 'Students.js' with the following code:

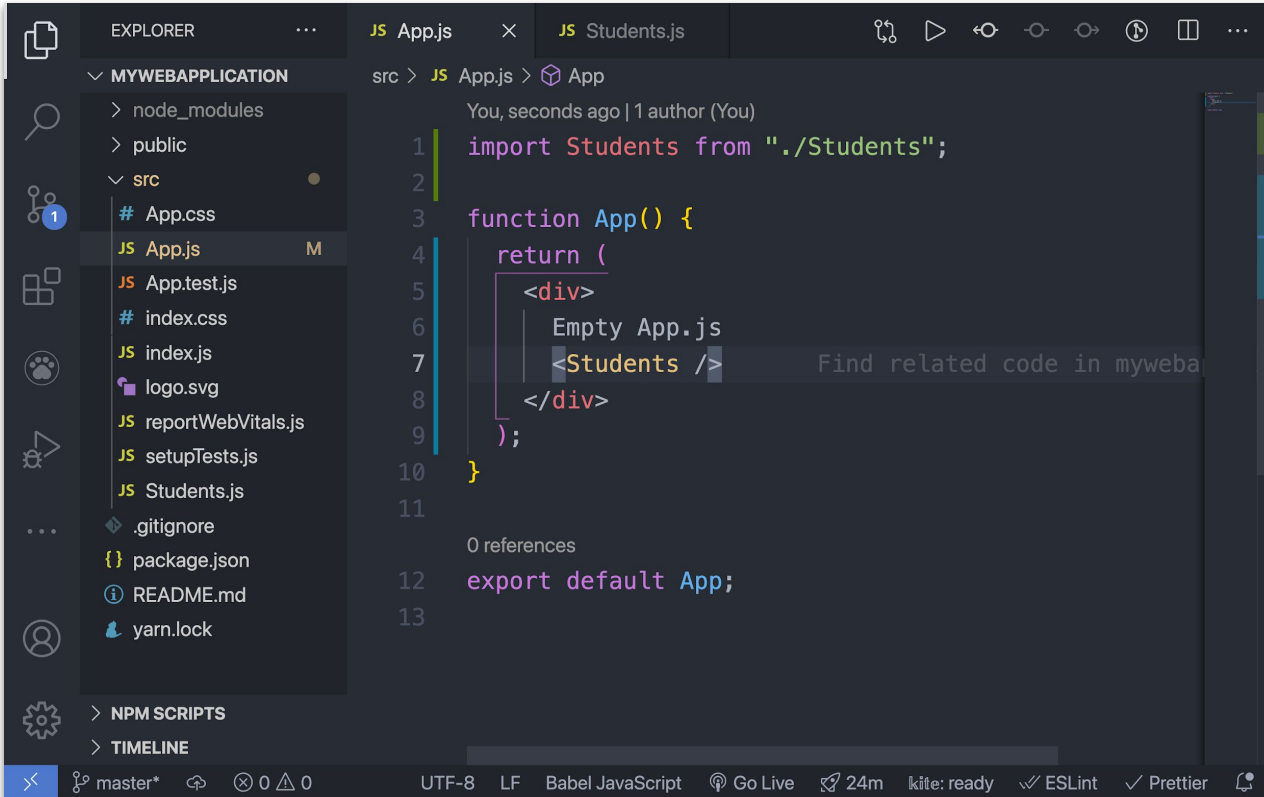
```
1 function Students() {  
2   return (  
3     <div>  
4       <p>Students in my school</p>  
5       <ul>  
6         <li>John Doe</li>  
7         <li>Jane Doe</li>  
8         <li>Student Three</li>  
9         <li>Mwanafunzi wa nne</li>  
10      </ul>  
11    </div>  
12  );  
13 }  
14  
15 export default Students;  
16
```

The status bar at the bottom shows the file is encoded in UTF-8, uses LF line endings, and is using Babel JavaScript. It also indicates that the file has 0 references and that the ESLint and Prettier extensions are active.





Import the **Students** components to **App.js**

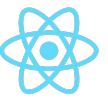


The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The Explorer sidebar shows the file structure of a project named 'MYWEBAPPLICATION'. The 'src' folder is expanded, showing files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'reportWebVitals.js', 'setupTests.js', and 'Students.js'. The 'App.js' file is selected. The main editor area shows the code for 'App.js'. The code is as follows:

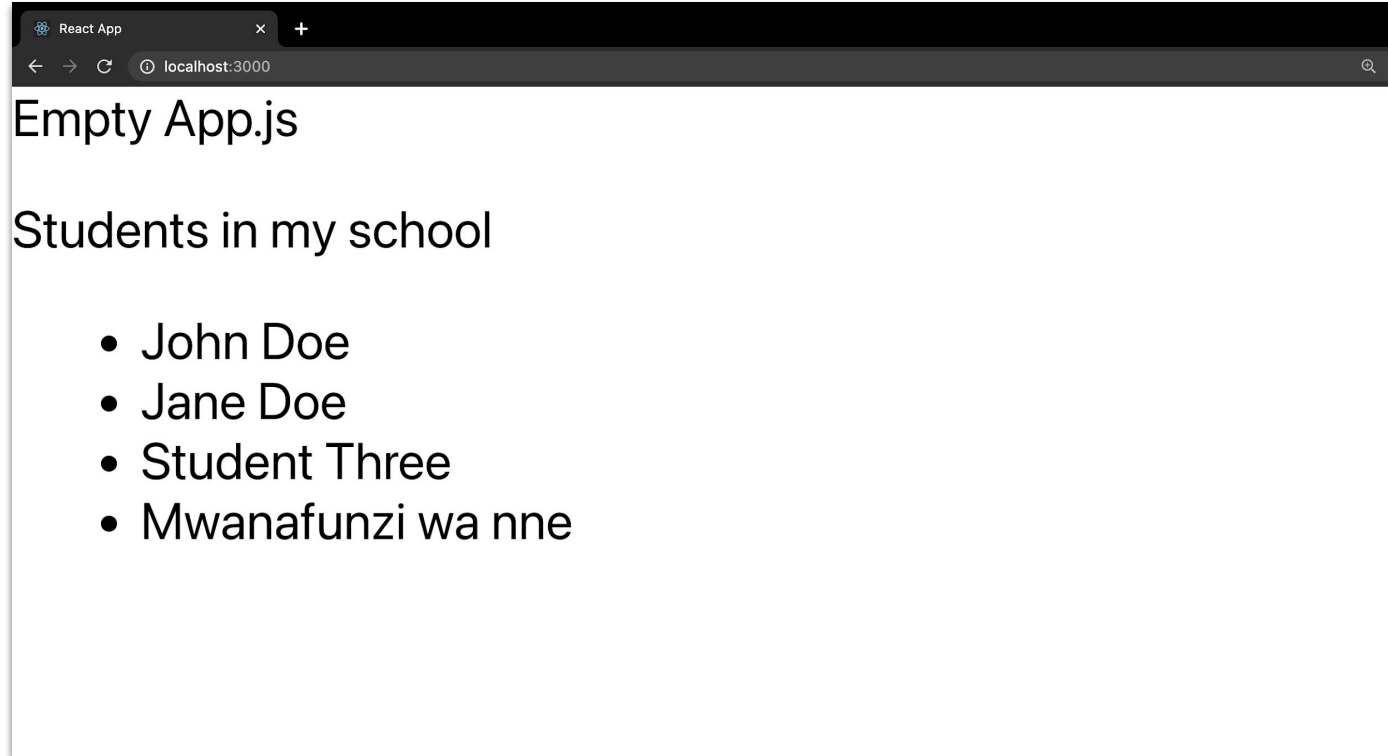
```
1 import Students from "./Students";
2
3 function App() {
4   return (
5     <div>
6       Empty App.js
7       <Students />
8     </div>
9   );
10 }
11
12 export default App;
```

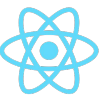
The code is highlighted with a blue selection bar on the left. The 'Students' component is imported from './Students'. The 'App' function returns a JSX element containing an 'Empty App.js' text and a 'Students' component. The 'Students' component is imported from './Students'. The 'App' function is exported as the default export.





Browser output after importing the **Students** components to **App.js**





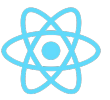
Create a new component named Library

The screenshot shows the Visual Studio Code editor interface. On the left, the Explorer sidebar displays the file structure of a project named 'MYWEBAPPLICATION'. The 'src' directory is expanded, showing files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'Library.js' (highlighted with a green 'U' icon), 'logo.svg', 'reportWebVitals.js', 'setupTests.js', 'Students.js', '.gitignore', 'package.json', 'README.md', and 'yarn.lock'. The main editor area shows the 'Library.js' file being edited. The code defines a 'Library' function that returns a JSX element. The JSX element contains a 'div' with a 'h1' tag, a 'p' tag, and a 'ul' tag with three 'li' tags. The third 'li' tag is currently selected. The code is as follows:

```
1 function Library() {
2   return (
3     <div>
4       <h1>Welcome to the library!</h1>
5       <p>Here are the books available</p>
6       <ul>
7         <li>The River and The Source</li>
8         <li>Damu Nyeusi</li>
9         <li>Harry Potter</li>
10      </ul>
11    </div>
12  );
13 }
14
15 export default Library;
16
```

Below the code, it says '0 references'. The status bar at the bottom shows 'master*' as the active branch, '0' errors, '0' warnings, and '0' info messages. It also displays 'UTF-8' encoding, 'LF' line endings, 'Babel JavaScript' as the language, 'Go Live' as the extension, '40m' as the time taken, 'kite: ready' as the status, and 'ESLint' and 'Prettier' as the linters.





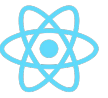
Import the Library component to App.js

The screenshot shows the VS Code editor interface. On the left, the Explorer sidebar displays the file structure of a project named 'MYWEBAPPLICATION'. The 'src' directory is expanded, showing files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'Library.js', 'logo.svg', 'reportWebVitals.js', 'setupTests.js', and 'Students.js'. The 'App.js' file is selected. The main editor area shows the 'App.js' file with the following code:

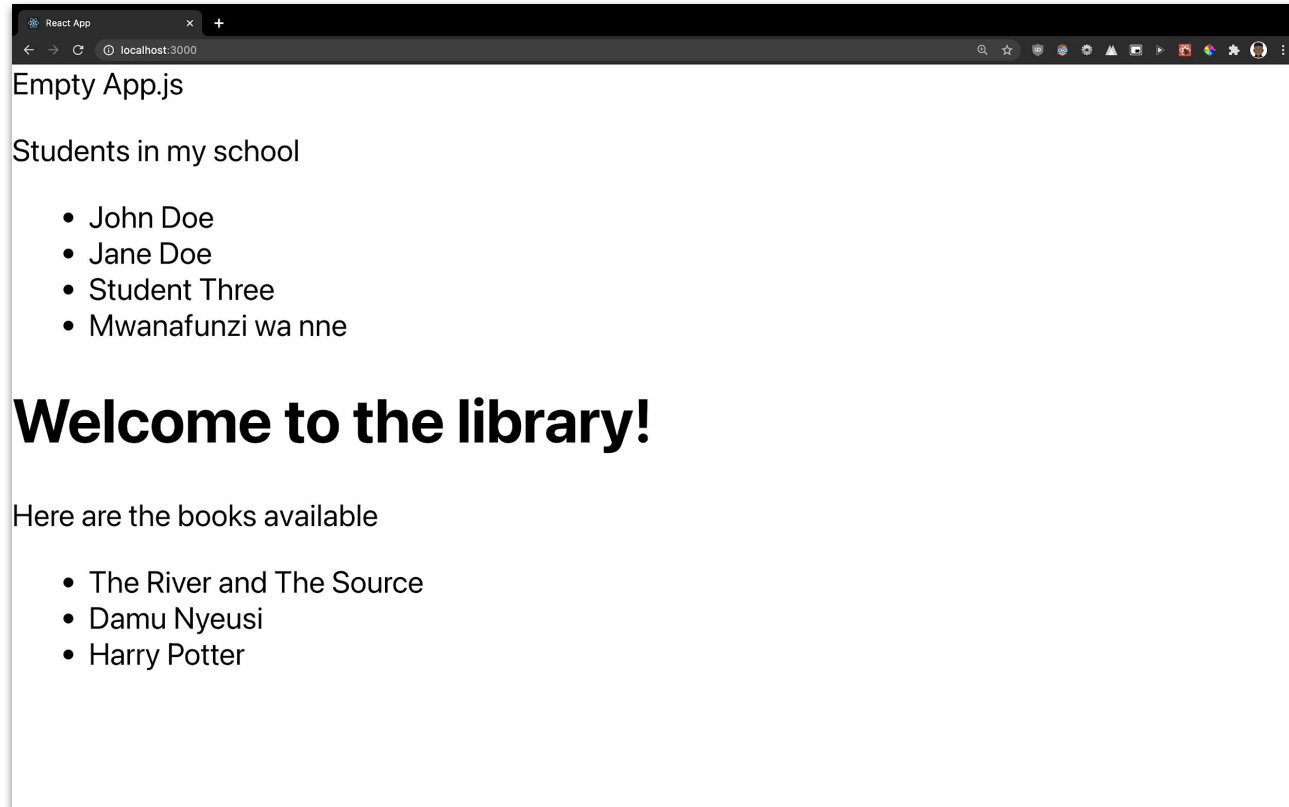
```
src > JS App.js > App
You, seconds ago | 1 author (You)
1  import Students from "./Students";
2  import Library from "./Library";
3
4  function App() {
5    return (
6      <div>
7        Empty App.js
8        <Students />
9        <Library />
10     </div>
11   );
12 }
13
14 export default App;
15
```

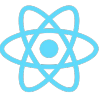
A tooltip is visible over the '<Library />' line, displaying the text 'Find related code in mywebap...'. The status bar at the bottom indicates the current file is 'App.js' and shows various tool icons like ESLint and Prettier.





Browser output after importing the Library component to App.js





Getting started with routing

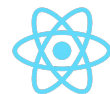
Routing is the process of selecting a path for traffic in a network/browser or between or across multiple networks.

react-router-dom is the de-facto React routing library, and it's one of the most popular projects built on top of React.

Routing links together your application navigation with the navigation features offered by the browser: the address bar and the navigation buttons. **react-router-dom** offers a way to write your code so that it will show certain components of your app only if the route matches what you define.

Install in your project by running the command: `npm install react-router-dom` command in your project directory.





React router reference:

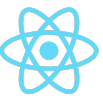
[BrowserRouter](#), usually aliased as [Router](#) wraps all your Route components.

[Link](#) components are used to generate links to your routes

[Route](#) components are responsible for showing - or hiding - the components they contain.

We'll come back to this.





Add **BrowserRouter** and create three **Routes** in App.js

```
1 import { BrowserRouter as Router, Route } from "react-router-  
2 import Students from "./Students";  
3 import Library from "./Library";  
4  
5 function App() {  
6   return (  
7     <div>  
8       <Router>  
9         <Route exact path="/" component={Students} />  
10        <Route exact path="/library" component={Library} />  
11        <Route exact path="/students" component={Students} />  
12      </Router>  
13    </div>  
14  );  
15 }  
16  
17 export default App;
```

We now have three routes in our application, *"/"* (home), *"/library"* & *"/students"*.

Let's learn how to navigate (change urls) next.



Add a link to “/” (home) and “/library” in our Students component.



We use the **Link** from react-router-dom to create clickable links to other parts of our react app.

```
<Link to=""> </Link>
```

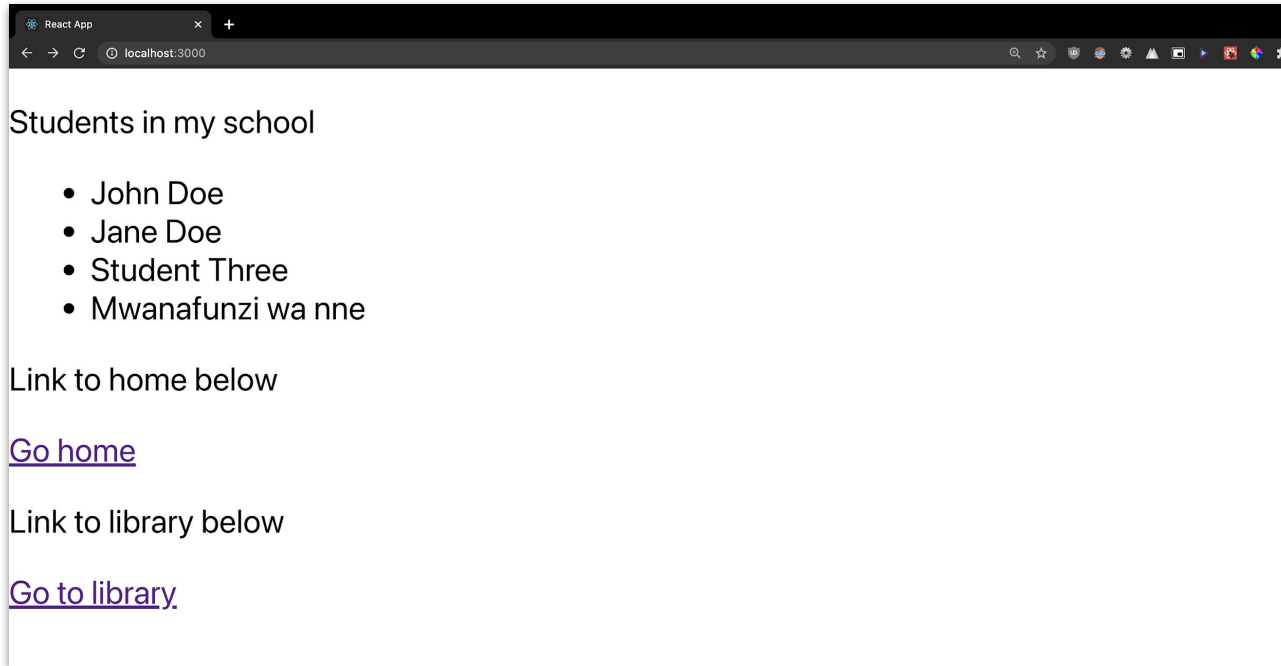
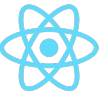
```
1 import { Link } from "react-router-dom";
2
3 function Students() {
4   return (
5     <div>
6       <p>Students in my school</p>
7       <ul>
8         <li>John Doe</li>
9         <li>Jane Doe</li>
10        <li>Student Three</li>
11        <li>Mwanafunzi wa nne</li>
12      </ul>
13
14      <p>Link to home below</p>
15      <Link to="/">Go home</Link>
16
17      <p>Link to library below</p>
18      <Link to="/library">Go to library</Link>
19    </div>
20  );
21 }
22
23 export default Students;
24 // Students component
```

We now have links to, “/” (home), & “/library” in our students component.

See how they look in the browser next ->



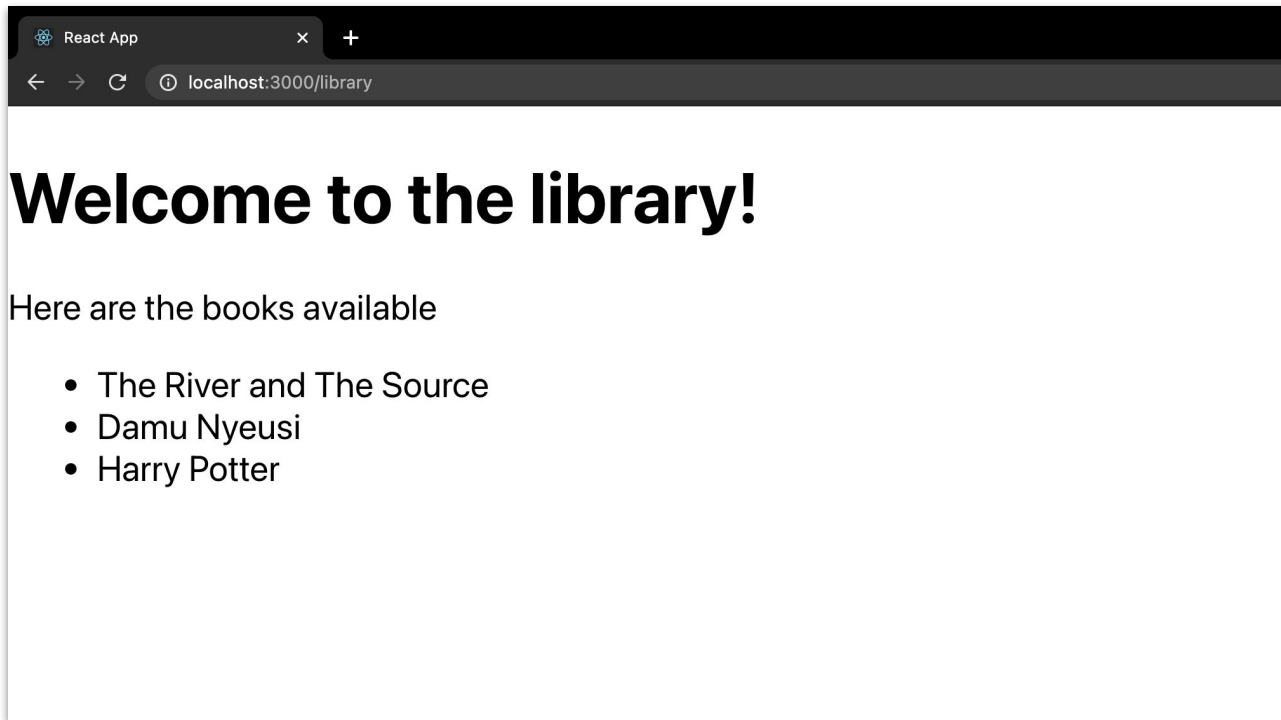
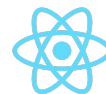
Browser output after adding a link to “/” (home) and “/library” in our Students component.



Click on the “Go to library” link to navigate to “/library”.



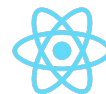
Browser output after clicking the “Go to library” link



We are now on the Library page! If you take a look at the url in the browser it should end with “/library”



Add a link to “/” (home) and “/school” in our Students component.



We use the **Link** from react-router-dom to create clickable links to other parts of our react app.

```
<Link to=""> </Link>
```

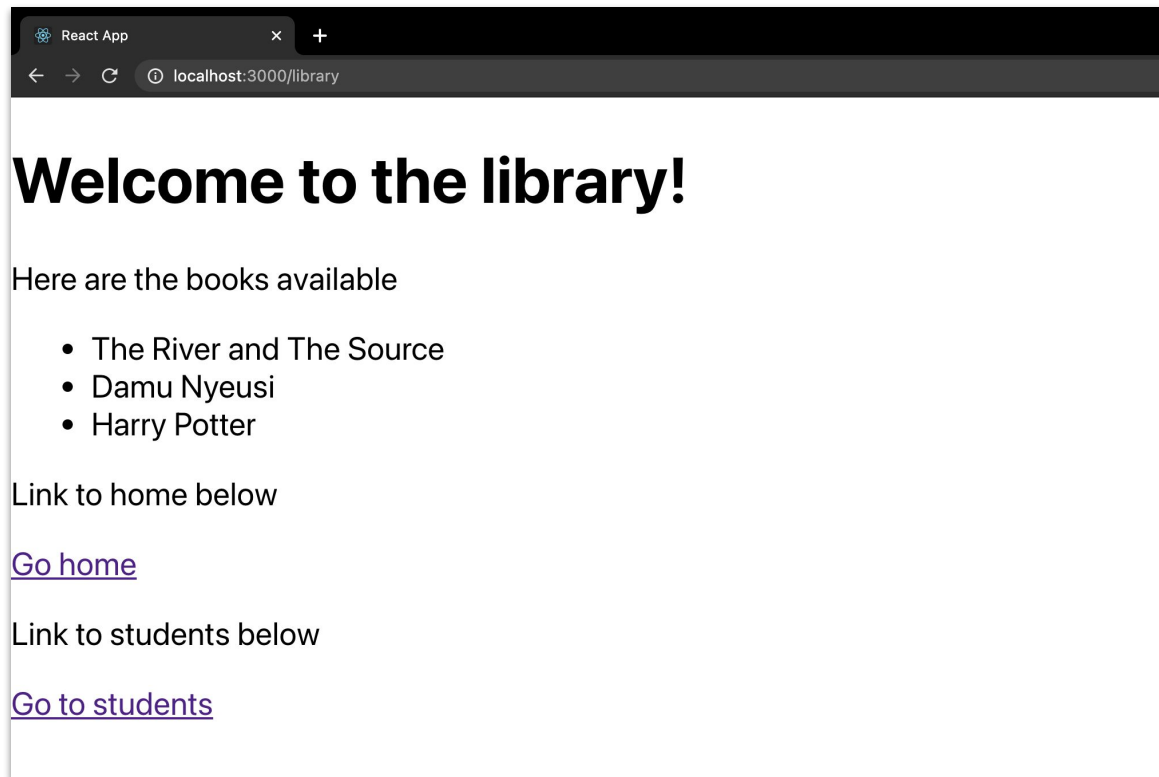
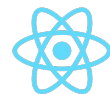
```
1 import { Link } from "react-router-dom";
2
3 function Library() {
4   return (
5     <div>
6       <h1>Welcome to the library!</h1>
7       <p>Here are the books available</p>
8       <ul>
9         <li>The River and The Source</li>
10        <li>Damu Nyeusi</li>
11        <li>Harry Potter</li>
12      </ul>
13
14      <p>Link to home below</p>
15      <Link to="/">Go home</Link>
16      <p>Link to students below</p>
17      <Link to="/students">Go to students</Link>
18
19    </div>
20  );
21 }
22
23 export default Library;
```

We now have links to, “/” (home), & “/school” in our students component.

See how they look in the browser next ->

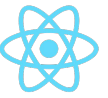


Browser output after adding a link to “/” (home) and “/students” in our Students component.



Click on the “Go to library” link to navigate to “/students”.





Build and deploy app to firebase

<https://dzone.com/articles/react-apps-firebase>

