Module 3: Handling Navigation with Routes

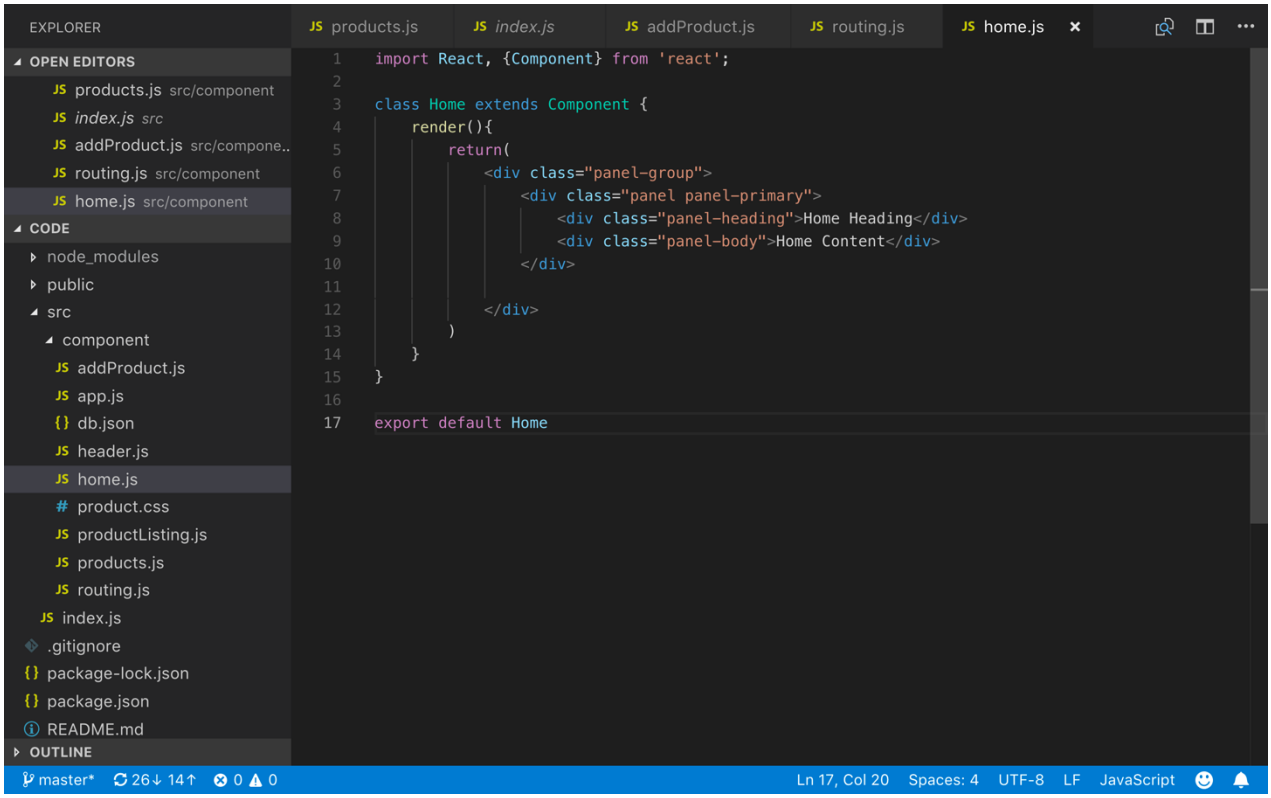
Demo Document 4

edureka!

Dynamic Music Store Application with Routing and API connectivity

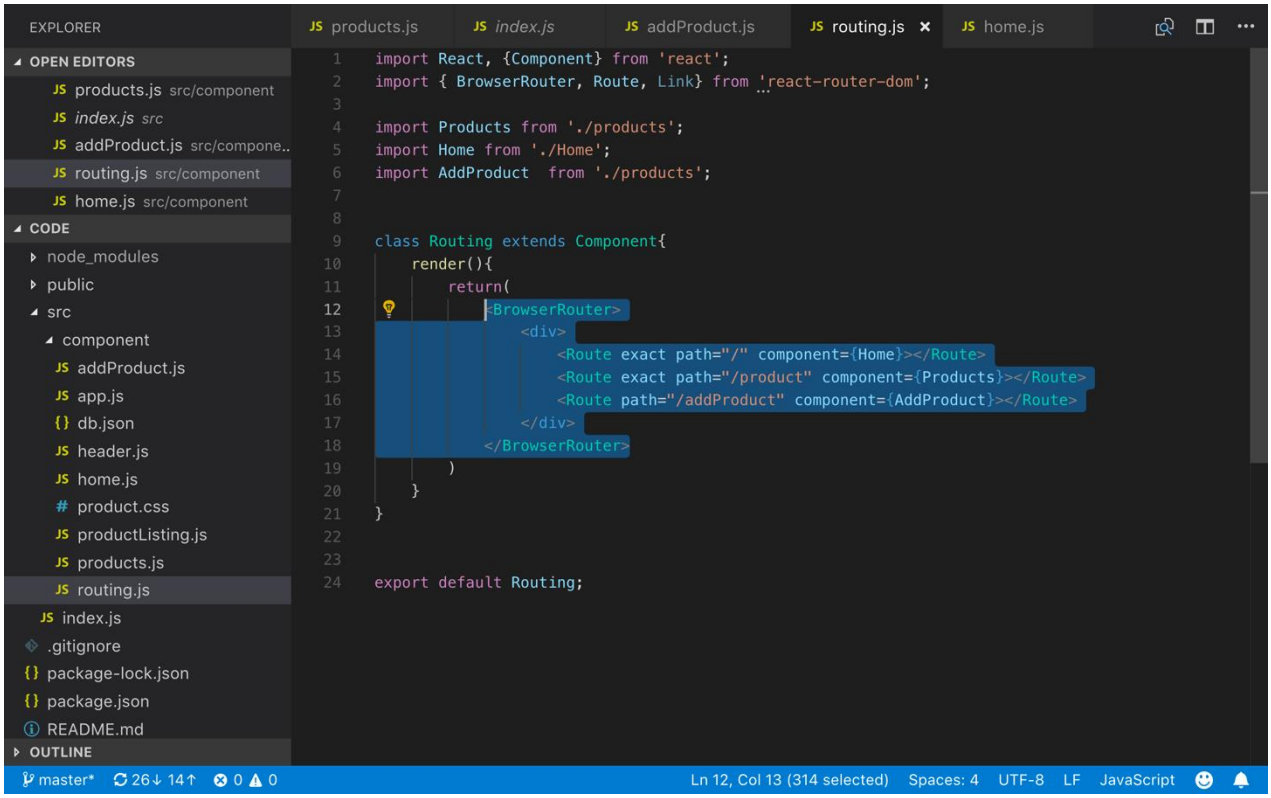
Step 1: To understand routing we need to create different components and navigate between these components. So, let us create the first component: ***Home***.

Open the component folder and create ***home.js*** as shown below.

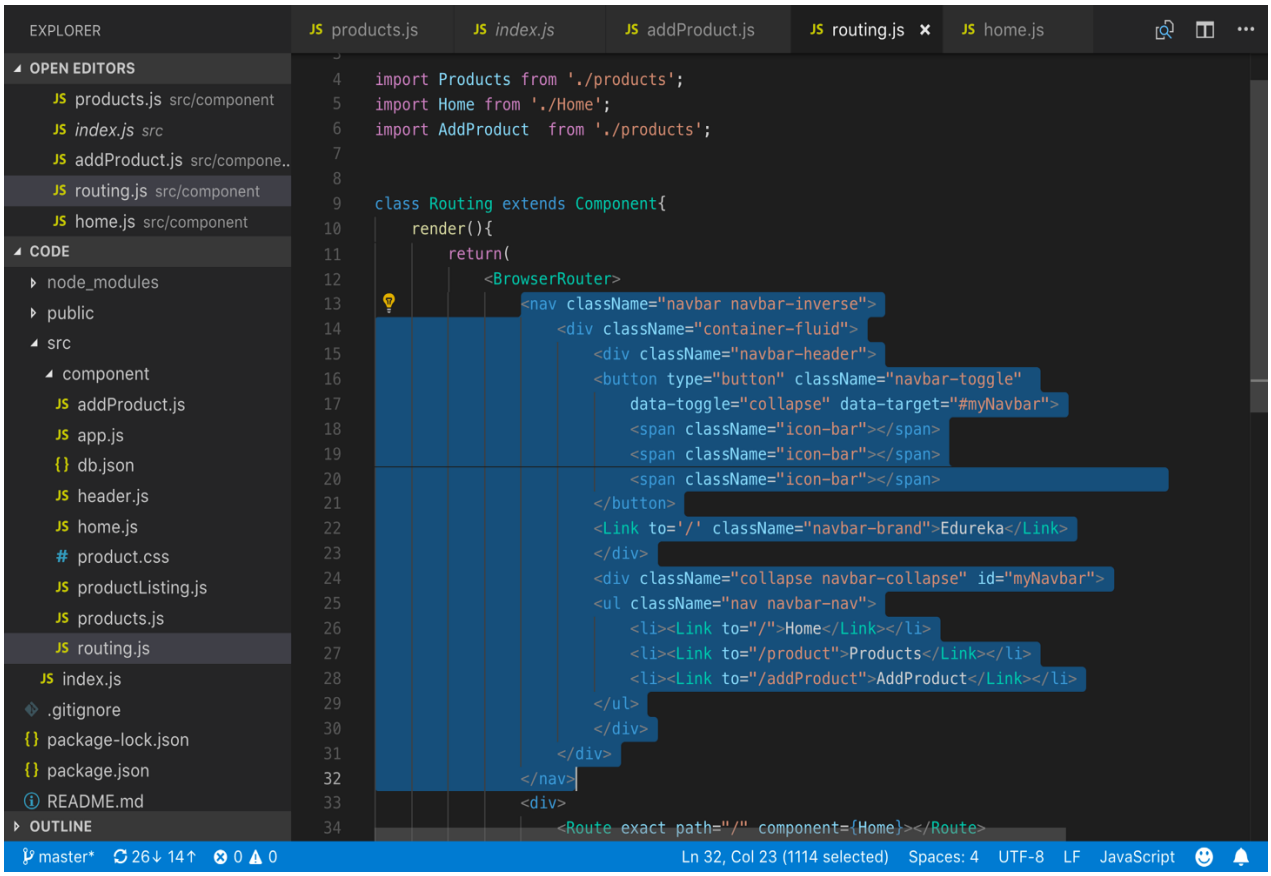


Step 2: Create a folder: ***routing.js***.

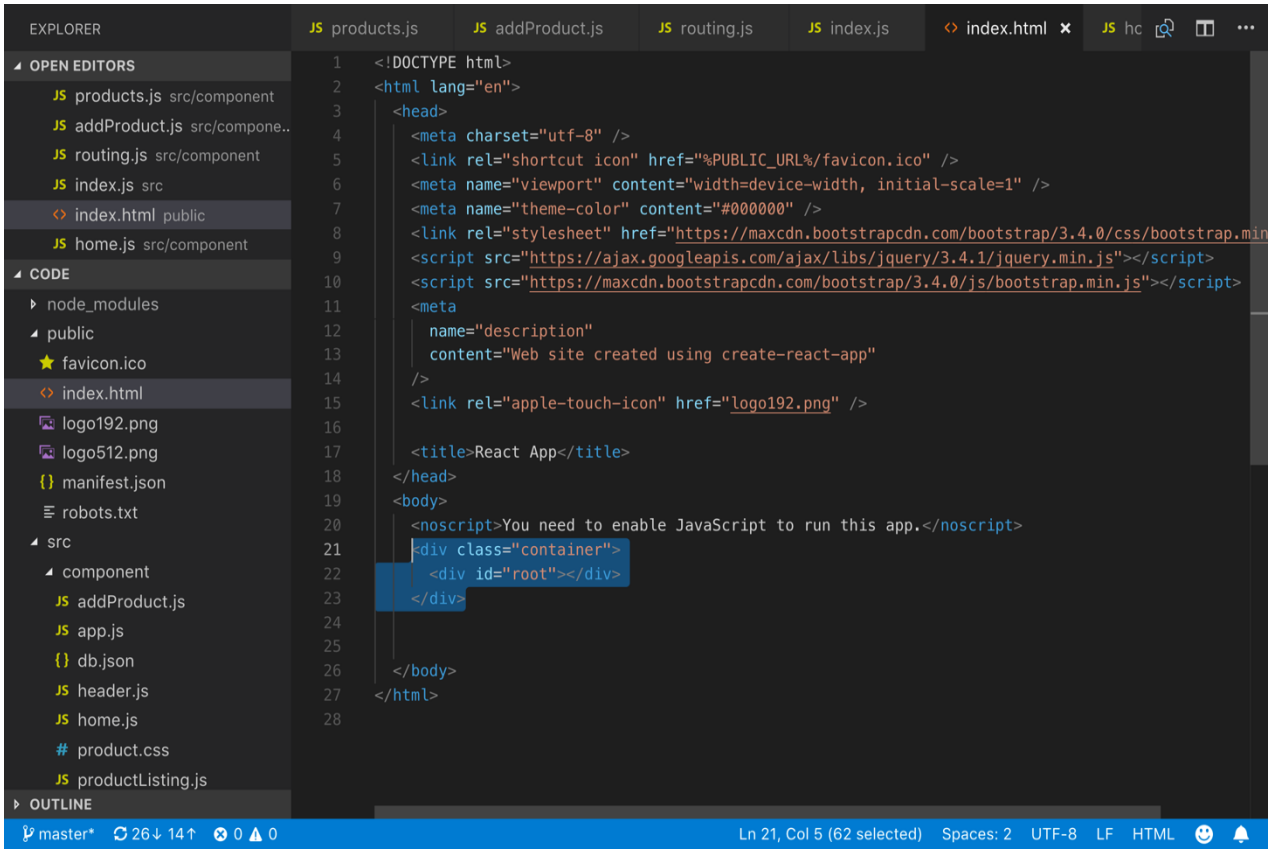
To implement routing install, ***react-router-dom*** using ***"npm i react-router-dom"*** and import ***browserRouter*** from ***react-router-dom.***



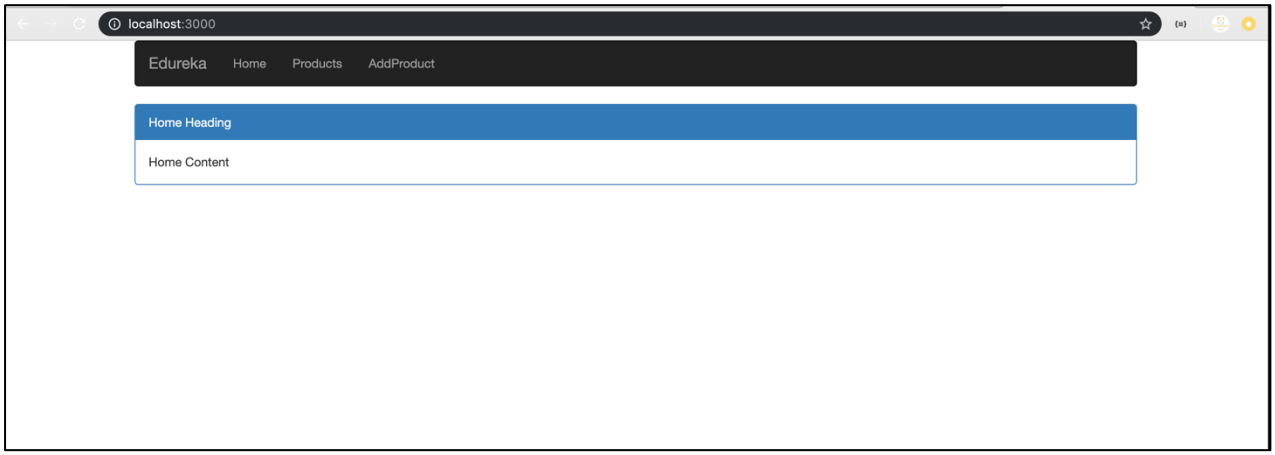
Step 3: Now we need to use bootstrap navbar, to implement the routes we ***will not use <a> tag,*** but we will use ***<link> tag*** and map the path to the link.



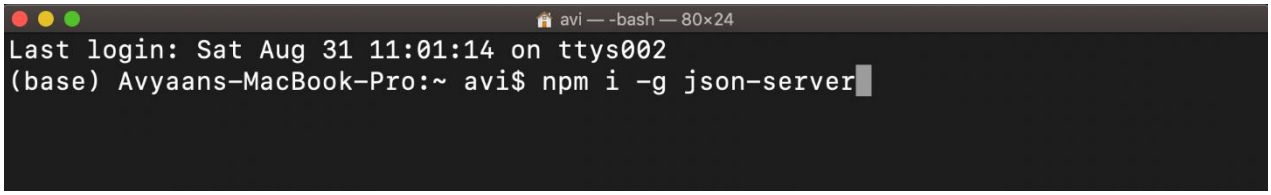
Step 4: In the ***index.html*** we have to wrap the ***root div*** with container class of bootstrap to make it responsive.



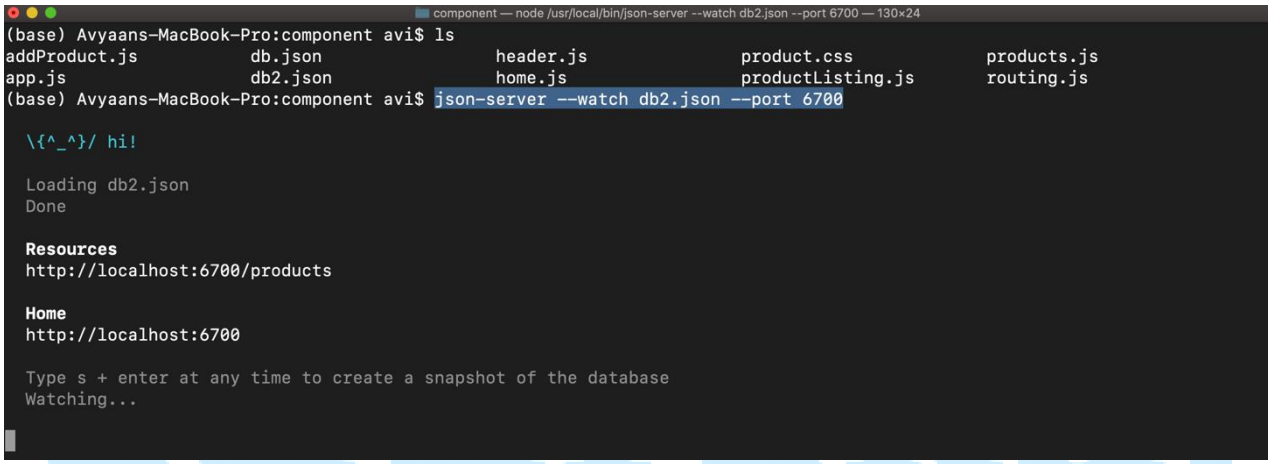
Step 5: If you check the browser, you will find out a new navbar. when you click on any menu item, you can see the page content changing in SPA manner without refreshing the page.



Step 6: For creating static API we have a package: ***json-server***. Install json-server as a global package.

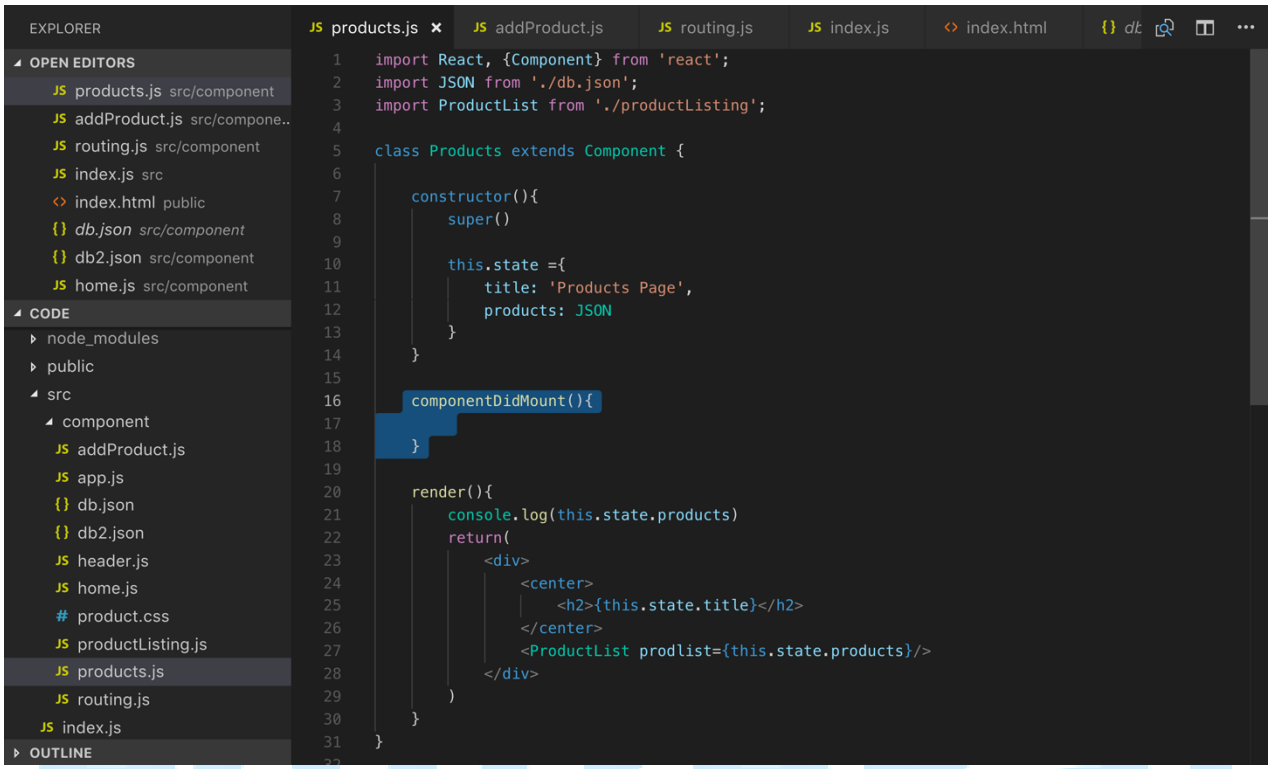


Step 7: Run the json-server with ***db.json*** file and provide a port number. By this you will be creating an API.



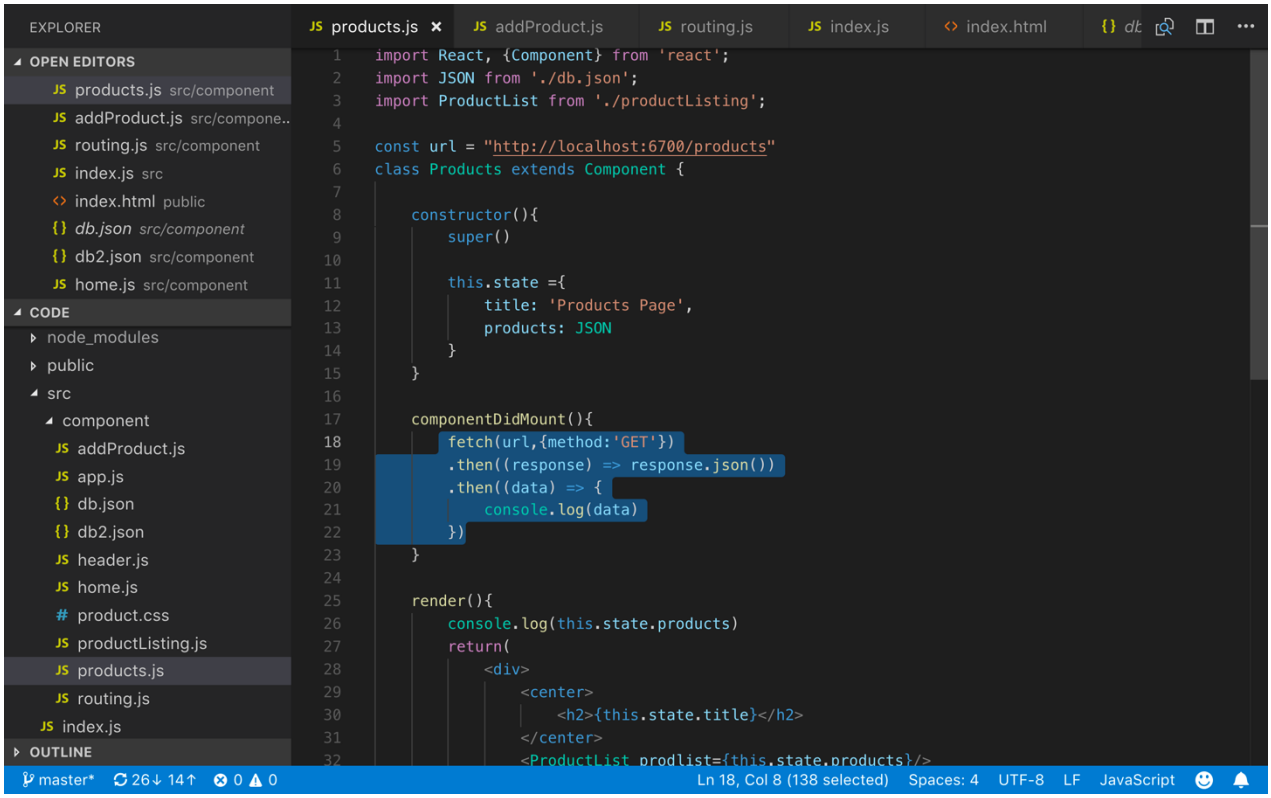
Step 8: Enter the endpoint in browser, you will receive json data present at the API.

Step 9: To call API, make use of lifecycle component: ***componentDidMount(),*** which will be executed after render part.

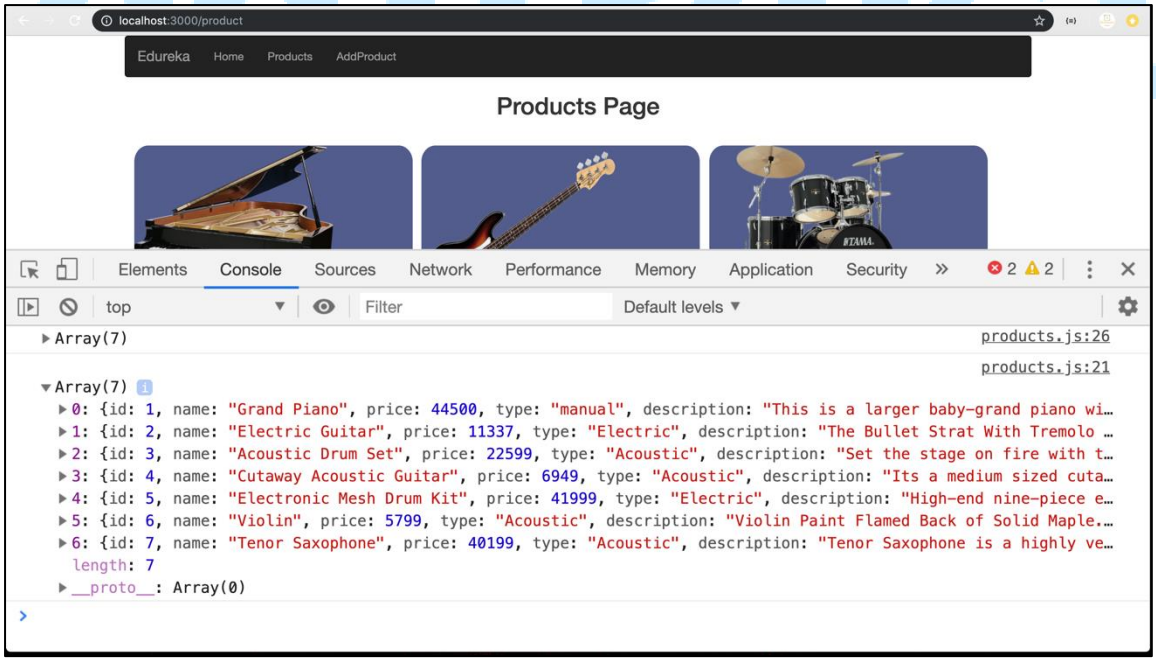


Step 10: Make a call to the API using ***fetch().***

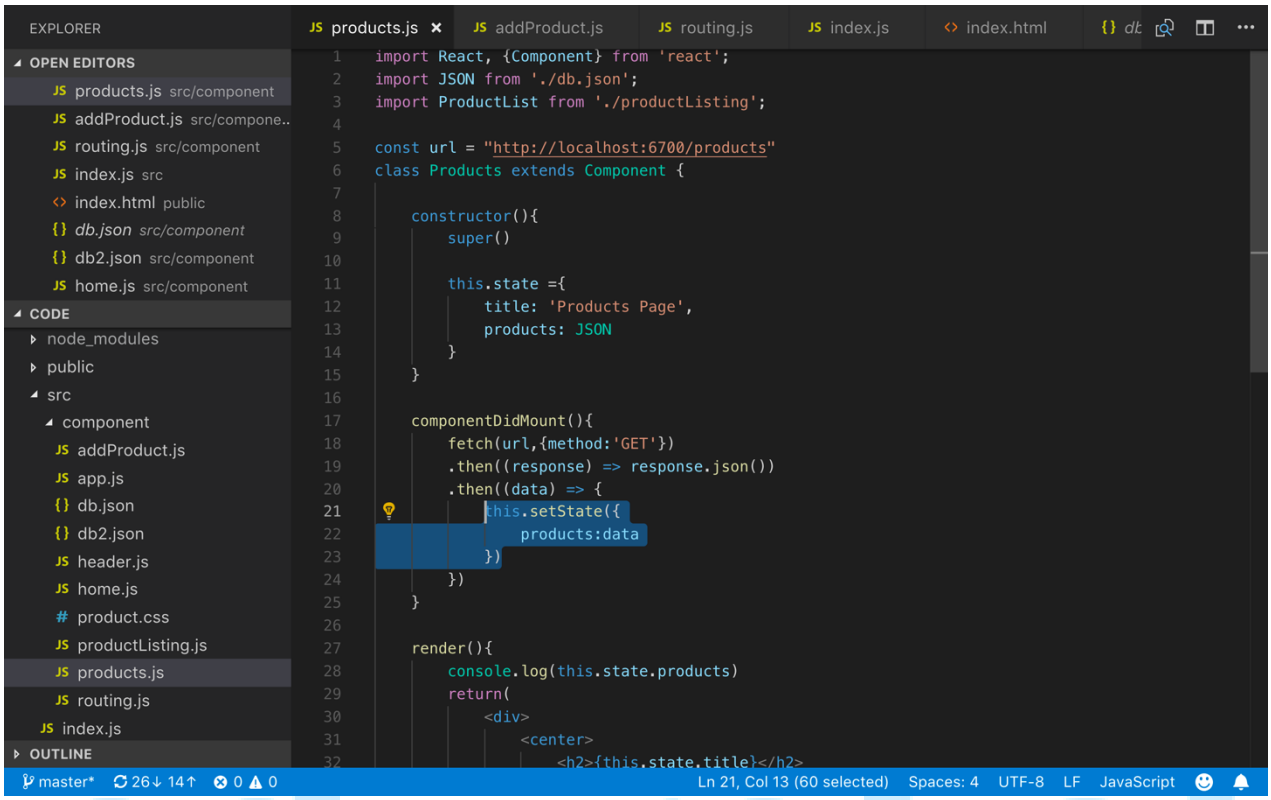
Using fetch() method we resolve a promise to get the data and the data is displayed in the console.log().



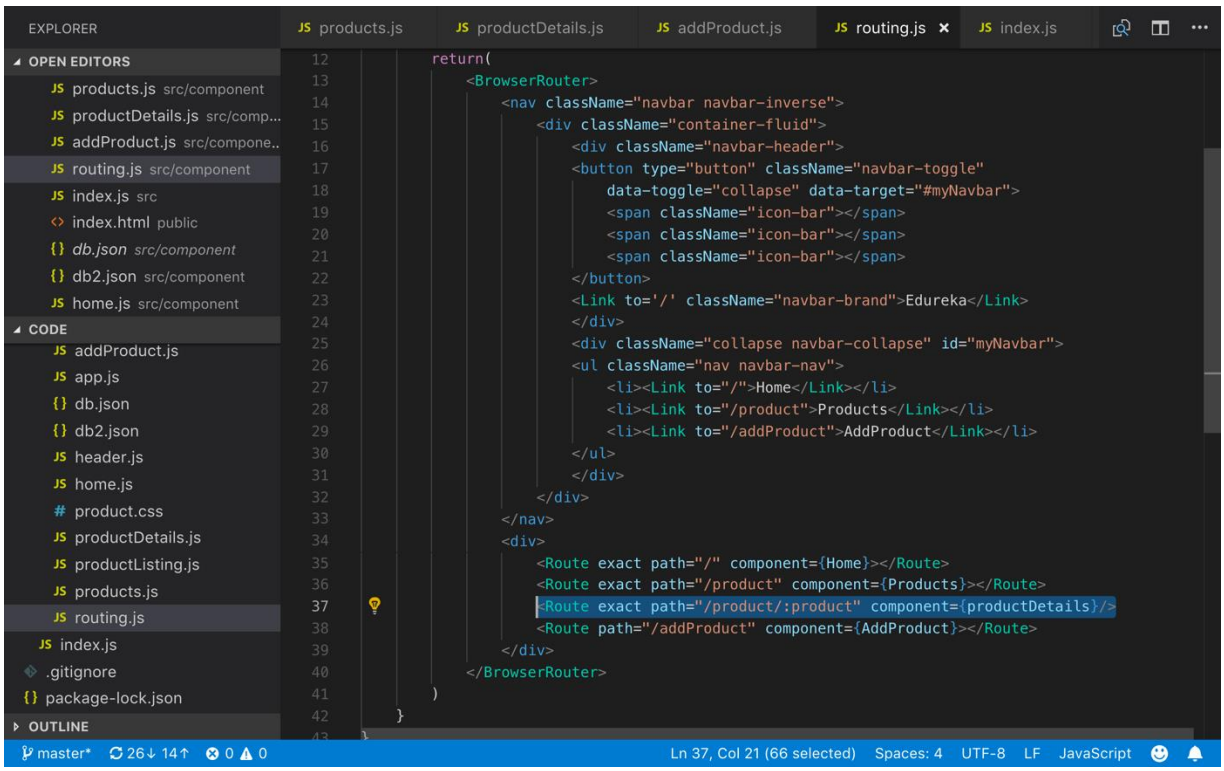
Step 11: Check the console, you can see the array of data returned from the API (as output).



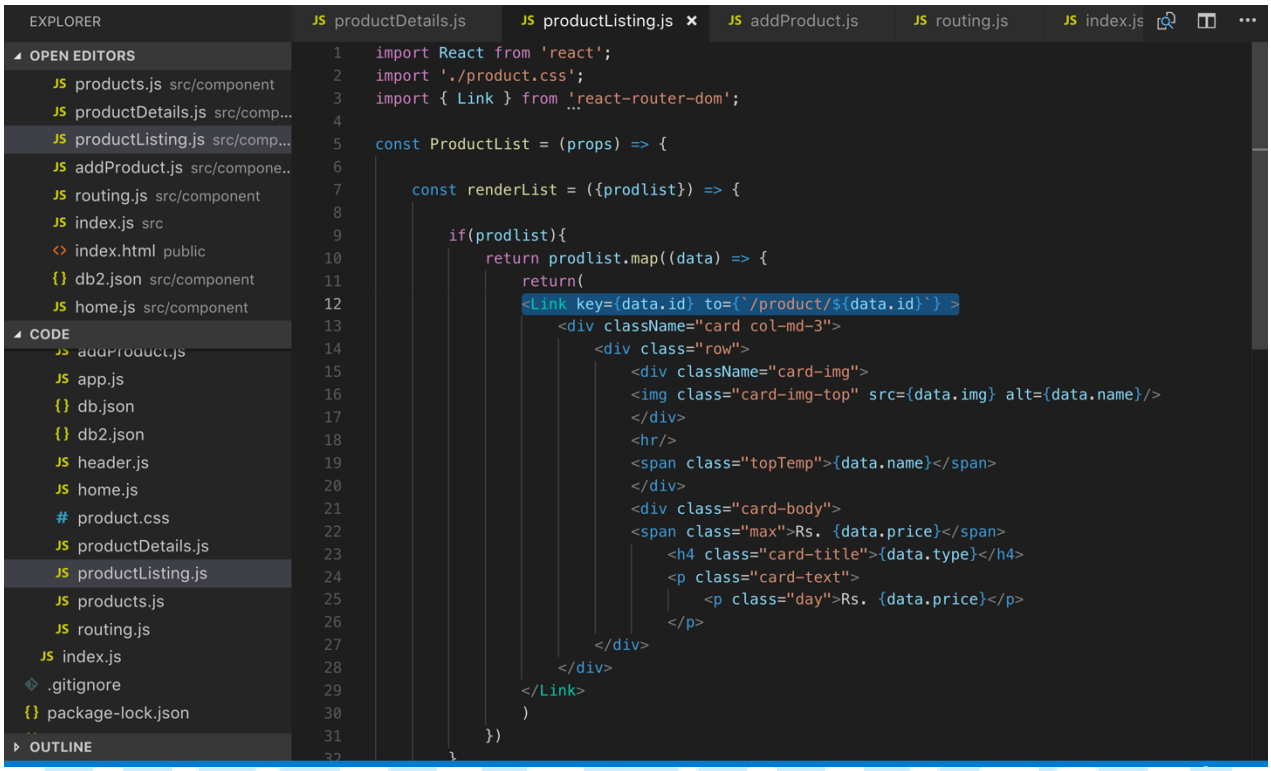
Step 12: Now save the received data to a state using ***this.setState.***



Step 13: To create a details page you have to create a ***route,*** here we will pass the param as id which will later act as a unique parameter.

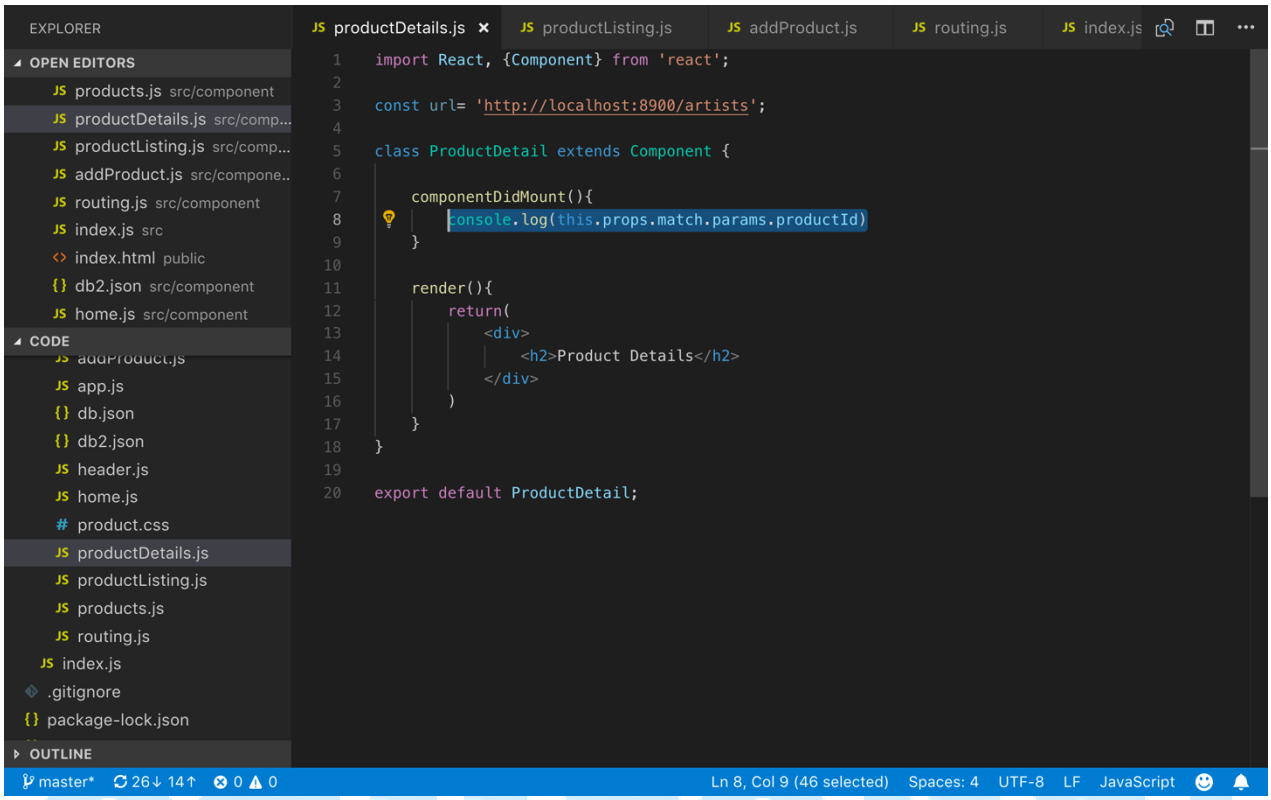


Step 14: Wrap all the element in the ***link tag*** and pass param as a unique id.

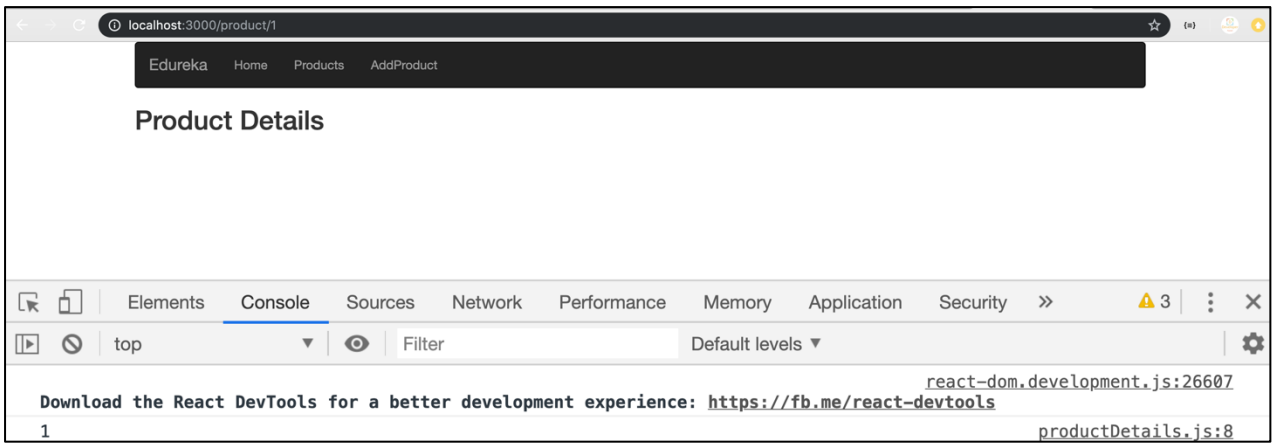


Step 15: Now on clicking any product user will be navigated to the details page of the respective product. Here you can see the params in the URL.

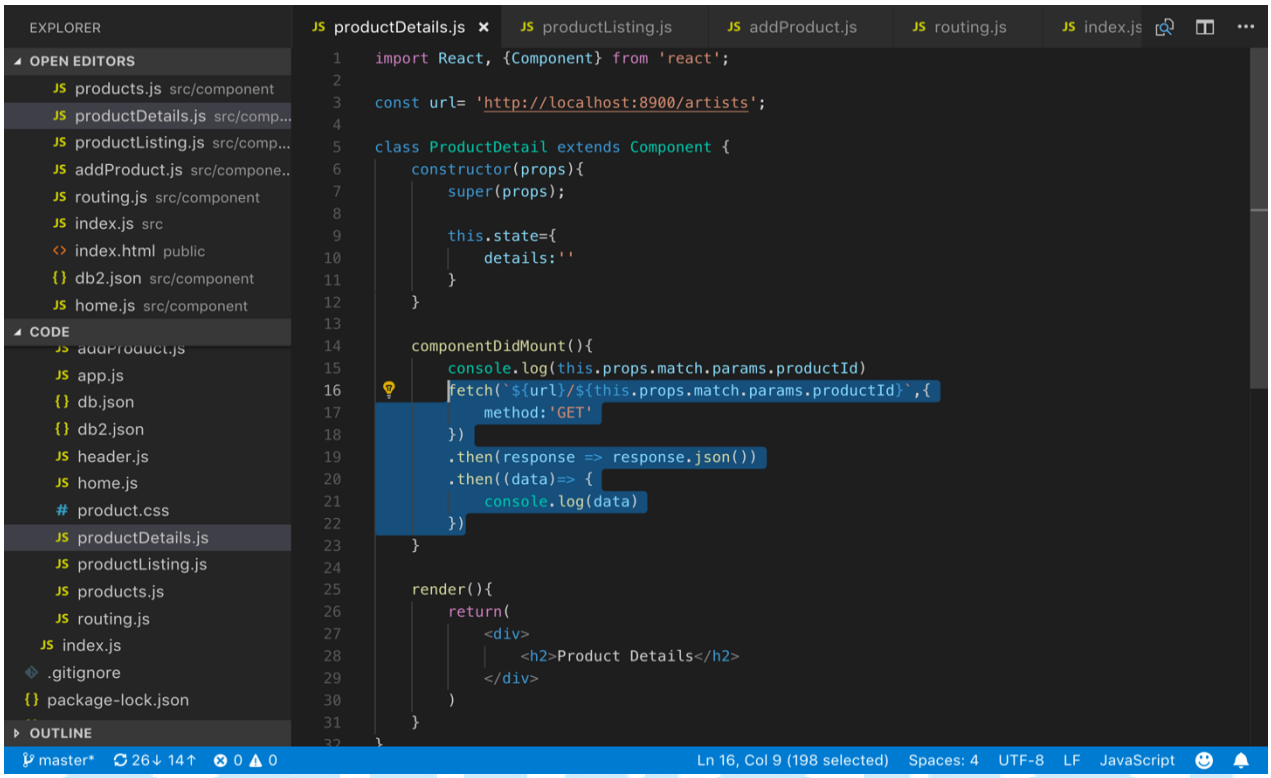
Step 16: This params value can be received via props, we will pass it to the API to get the data of respective item.



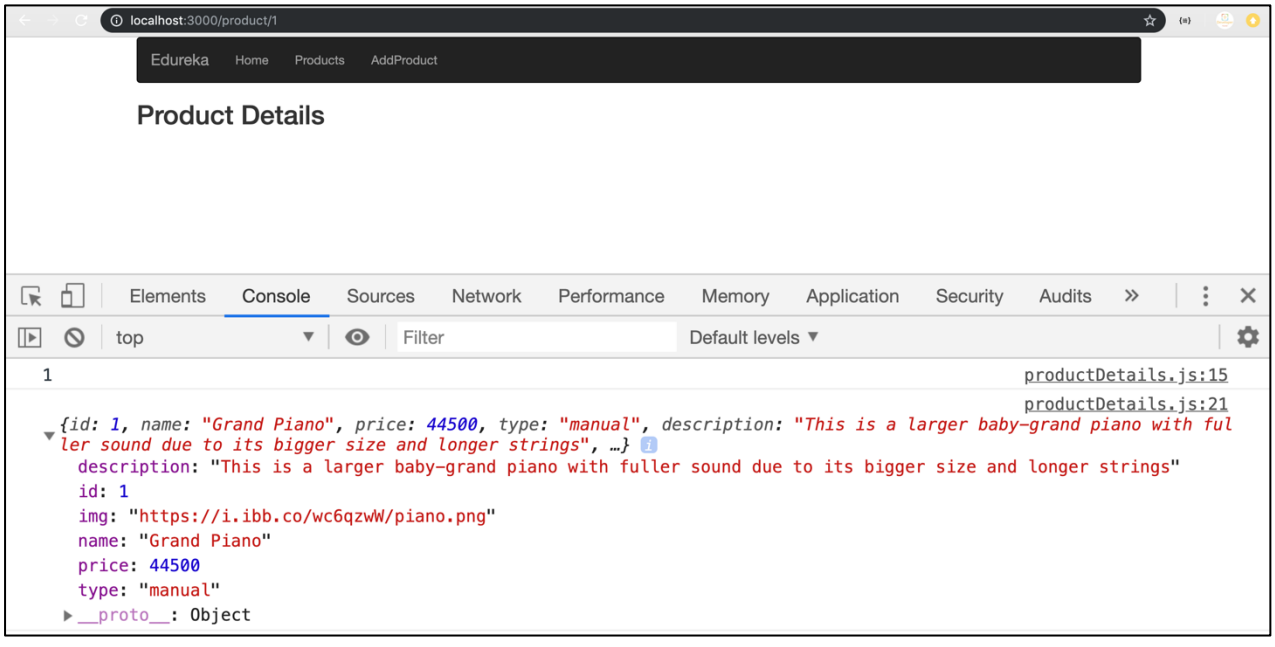
Step 17: If you check the console of the details page, you can see the param in the console.



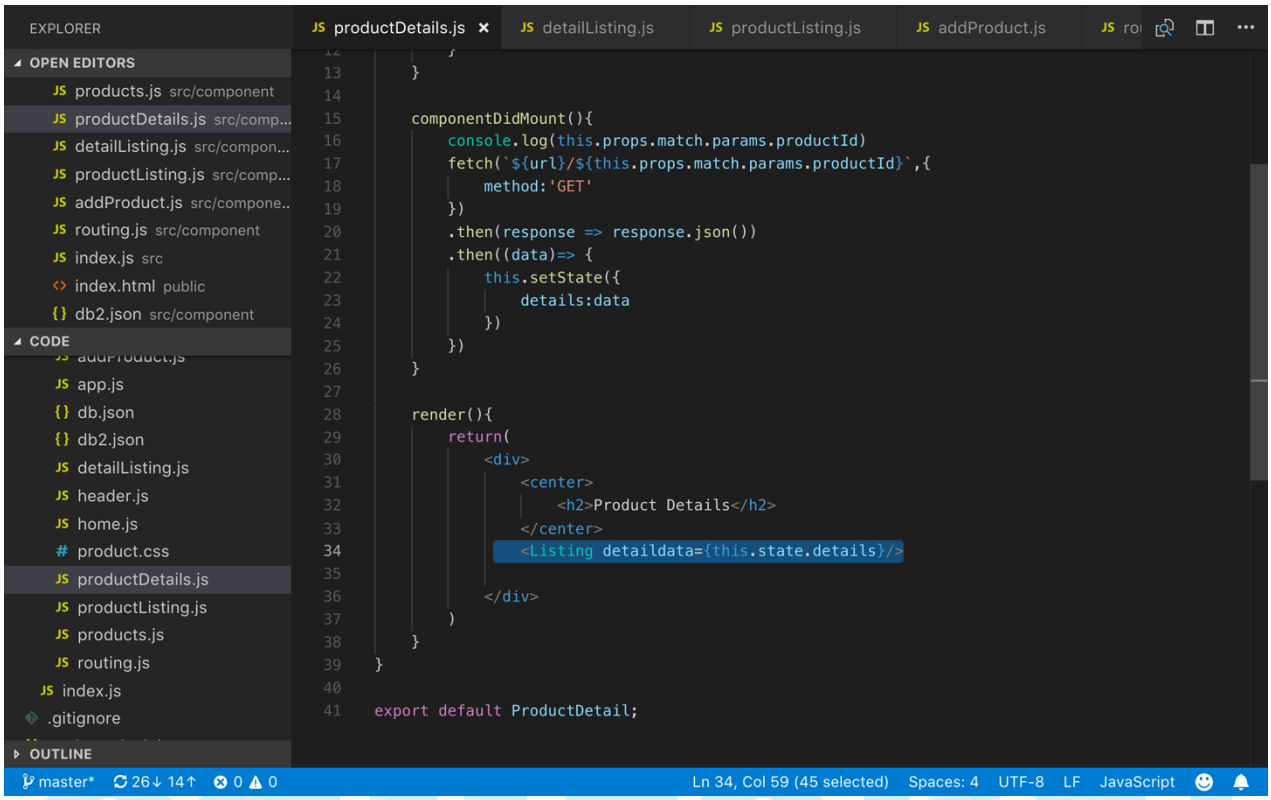
Step 18: Move to fetch() and this time pass the param in the URL, to retrieve particular data (later, set it to a state).



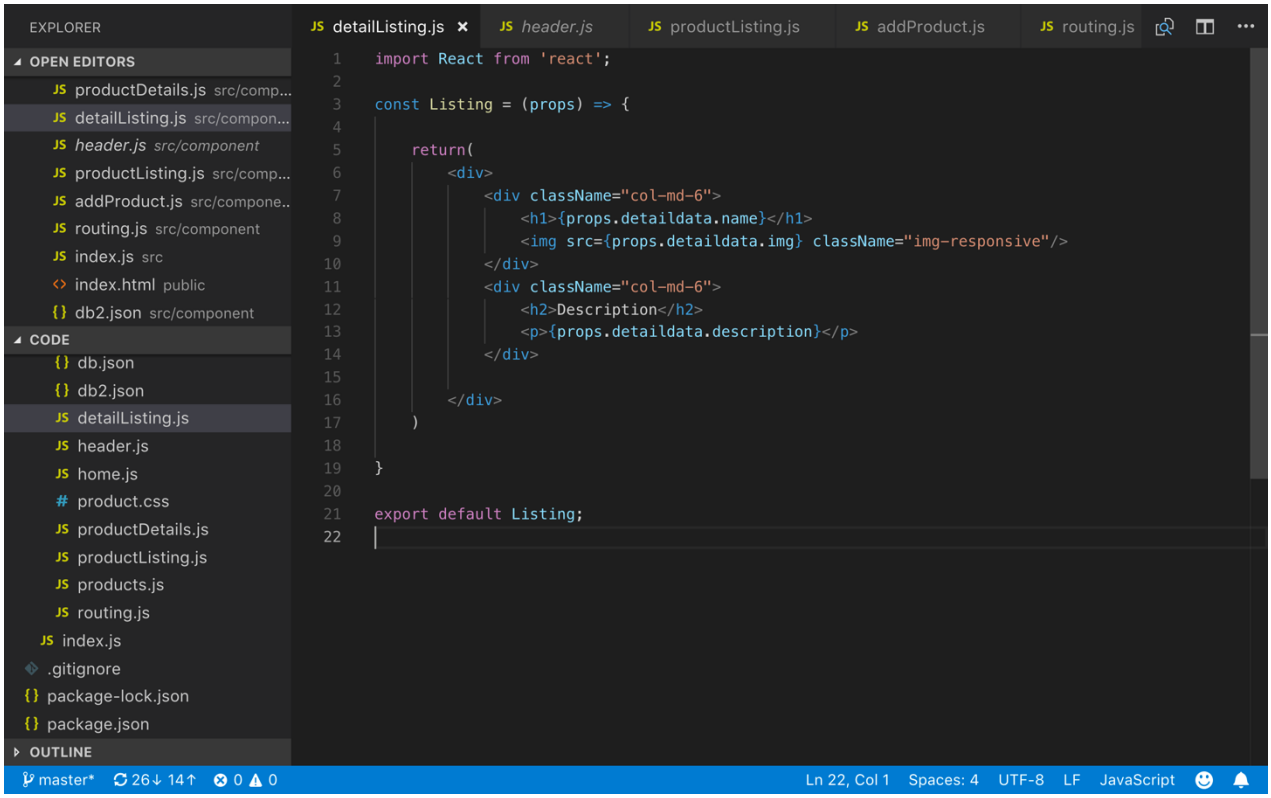
Step 19: Now, when you check the console of details page, you can see the data which we have to display on the browser.



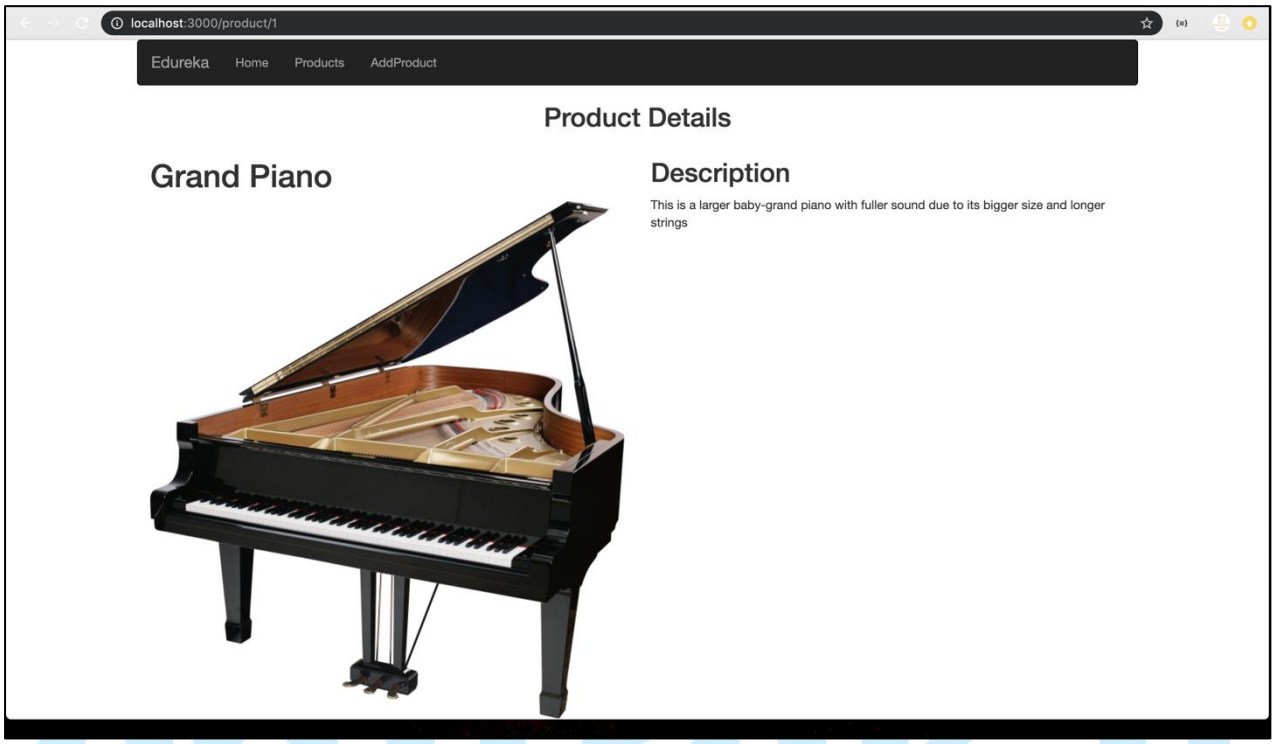
Step 20: Create a file called ***detail_listing.js***, to send the output of the details page.



Step 21: Perform the ***data binding*** to display the data of respective product in the details page.



Step 22: Finally, by clicking on any particular product, the details of the respective product is displayed on the screen.



Conclusion:

We have successfully displayed the details of the respective product from an API with the help of routing methods.