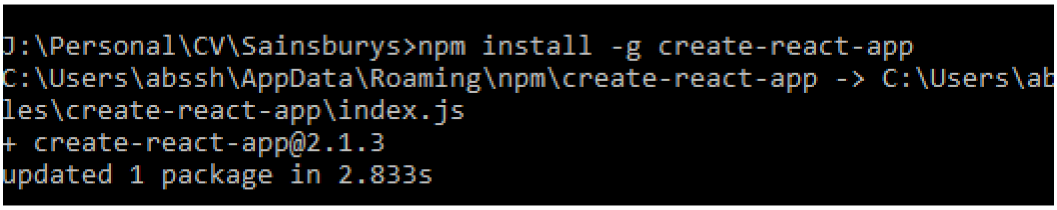
NASA Image Search using ReactJS

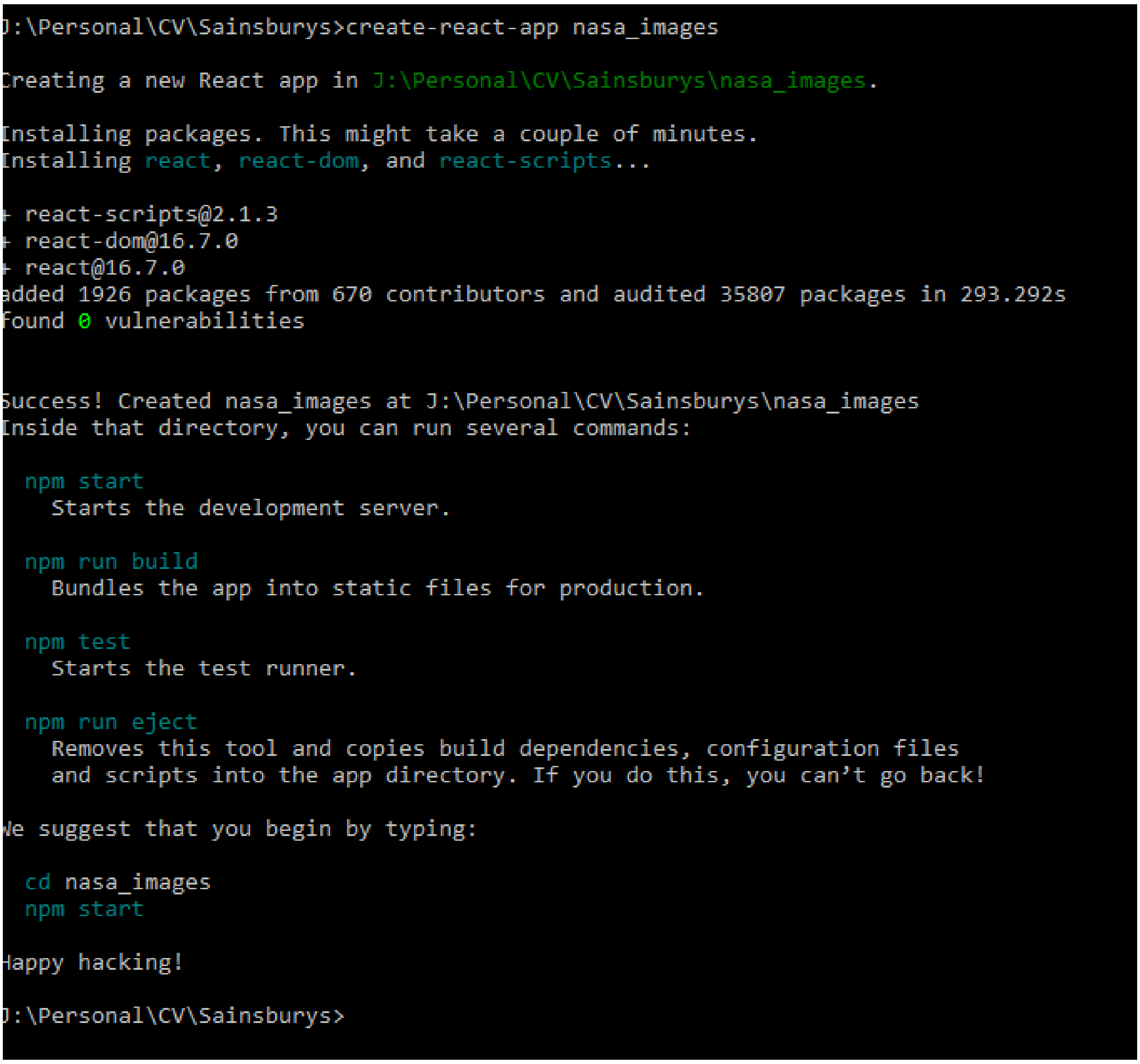
Page - https://aik1979.github.io/nasa-image-search/
Repository - https://github.com/aik1979/nasa-image-search

1. Install React Webpack using Node
    a. Run → npm install -g create-react-app



```
J:\Personal\CV\Sainsburys>npm install -g create-react-app
C:\Users\abssh\AppData\Roaming\npm\create-react-app -> C:\Users\ab
les\create-react-app\index.js
+ create-react-app@2.1.3
updated 1 package in 2.833s
```

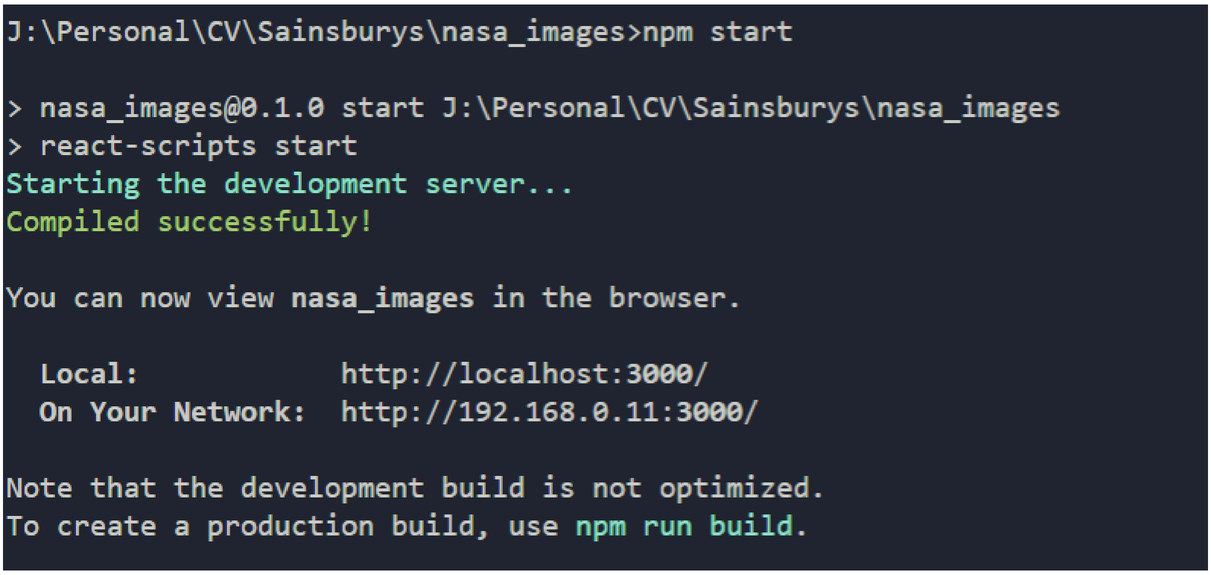
2. Create your project
    a. Run → create-react-app nasa_images



```
J:\Personal\CV\Sainsburys>create-react-app nasa_images

Creating a new React app in J:\Personal\CV\Sainsburys\nasa_images.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

+ react-scripts@2.1.3
+ react-dom@16.7.0
+ react@16.7.0
added 1926 packages from 670 contributors and audited 35807 packages in 293.292s
found 0 vulnerabilities


Success! Created nasa_images at J:\Personal\CV\Sainsburys\nasa_images
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd nasa_images
  npm start

Happy hacking!

J:\Personal\CV\Sainsburys>
```

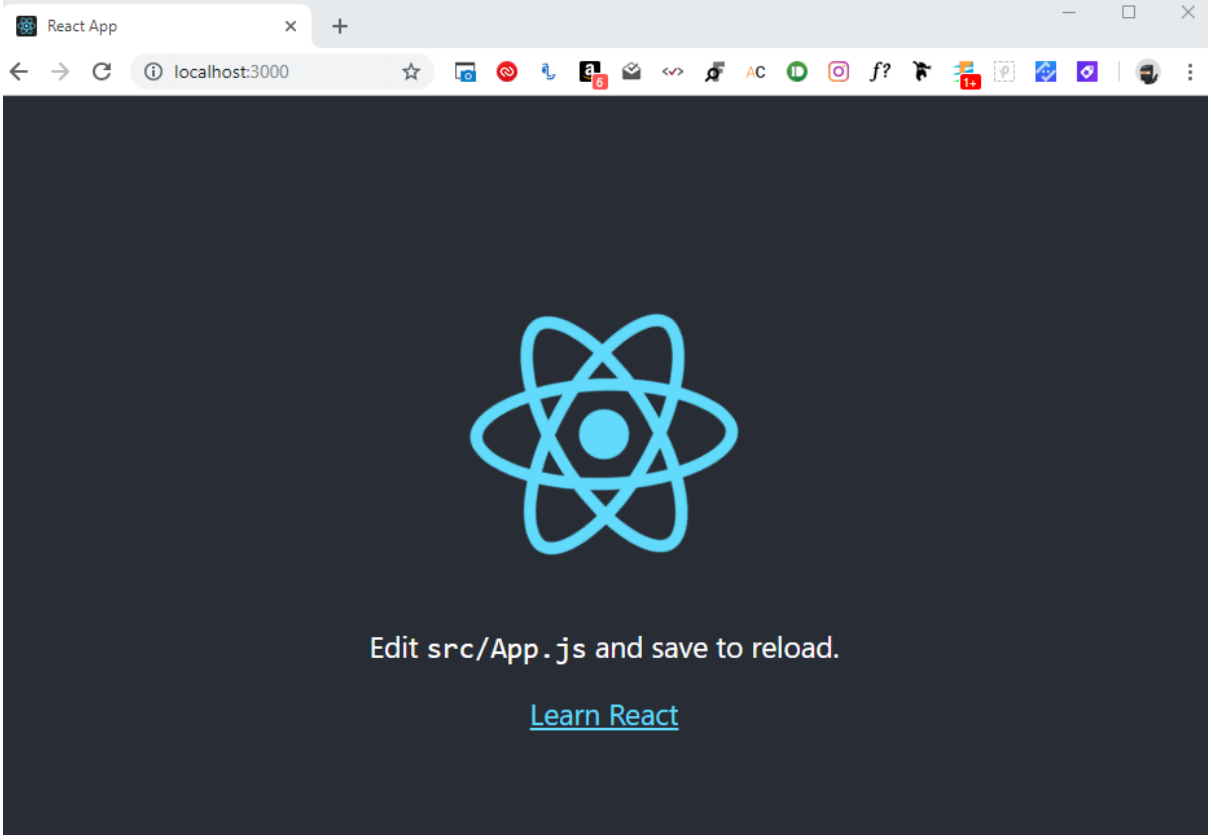
3. I use Visual Studio Code or Atom for my coding, for this project I will use Visual Studio Code (VSC).
4. Open the project folder created in step 2 into VSC.

5. Start the server
    a. Run → npm start



```
J:\Personal\CV\Sainsburys\nasa_images>npm start

> nasa_images@0.1.0 start J:\Personal\CV\Sainsburys\nasa_images
> react-scripts start
Starting the development server...
Compiled successfully!

You can now view nasa_images in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.0.11:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.
```

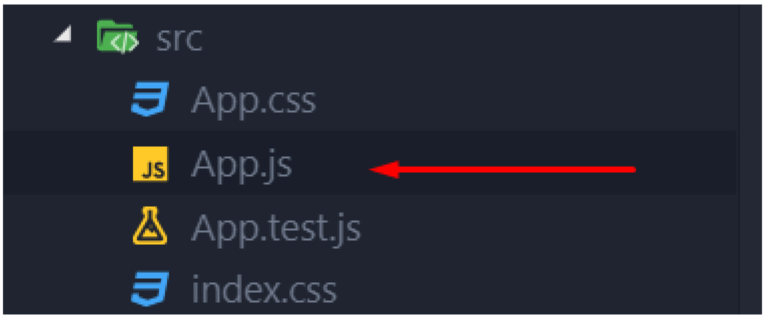
6. The server will start and a browser will open with the project showing



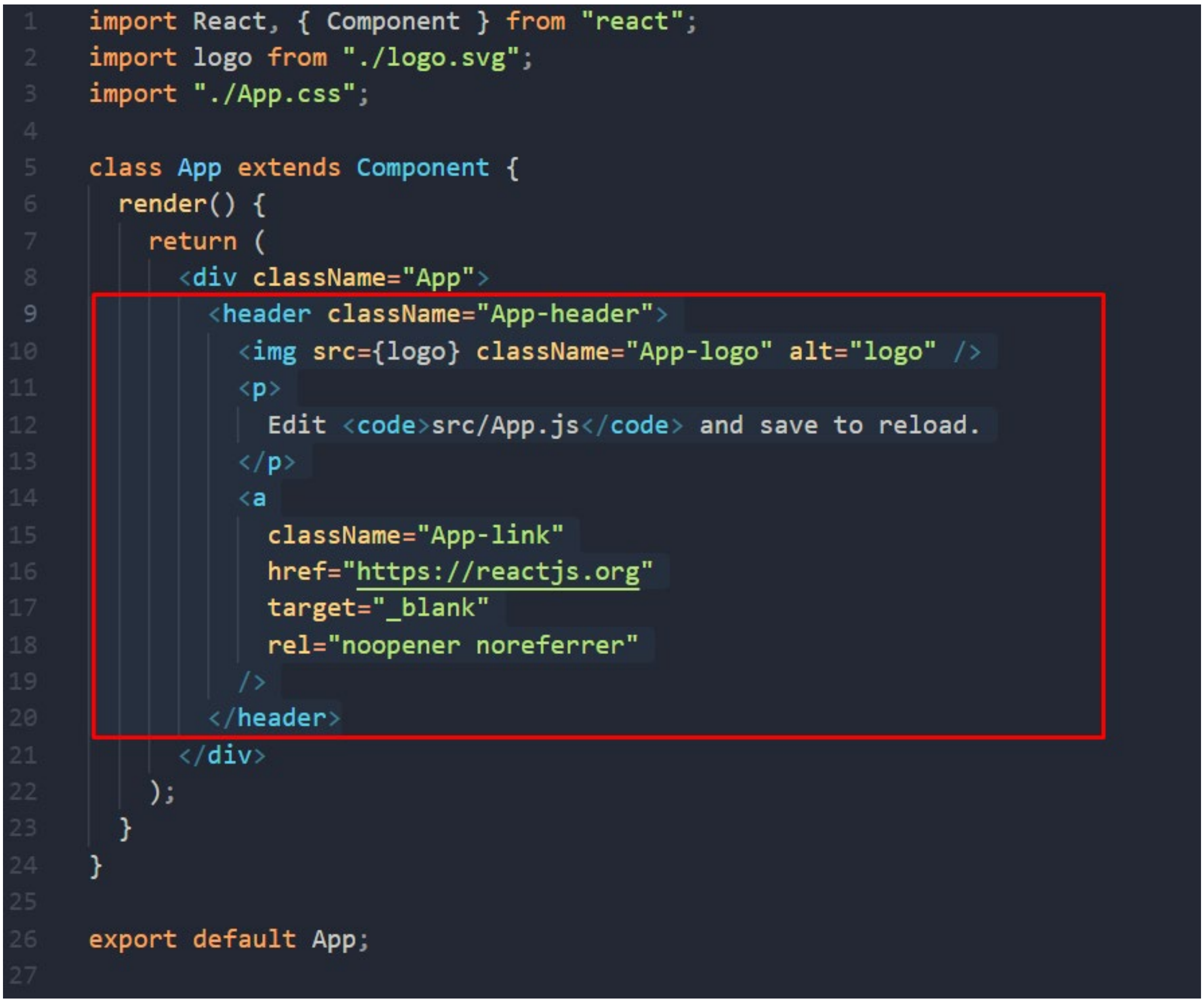
Edit `src/App.js` and save to reload.

Learn React

7. Open src/App.js

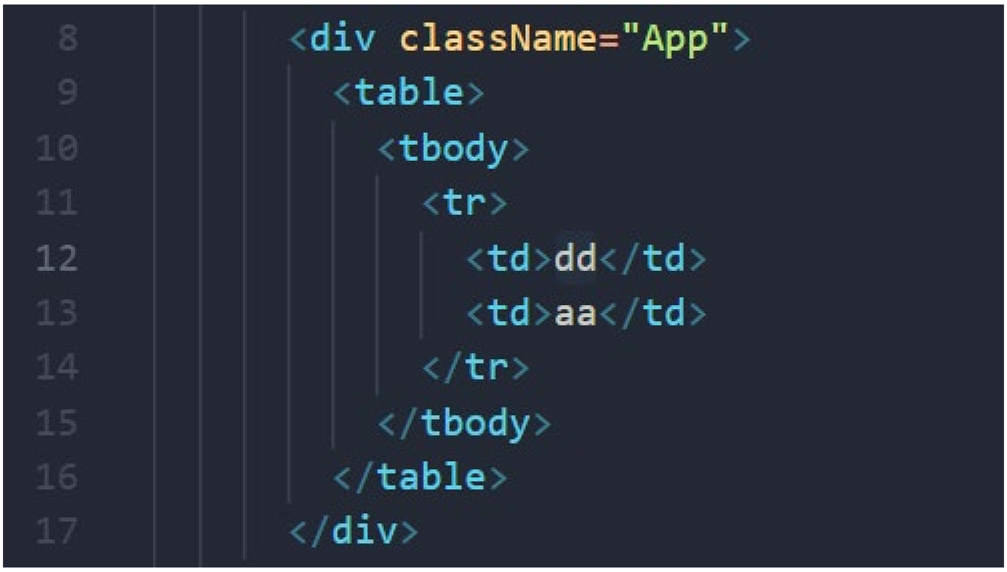


8. Delete everything within the DIV tag with className “App”



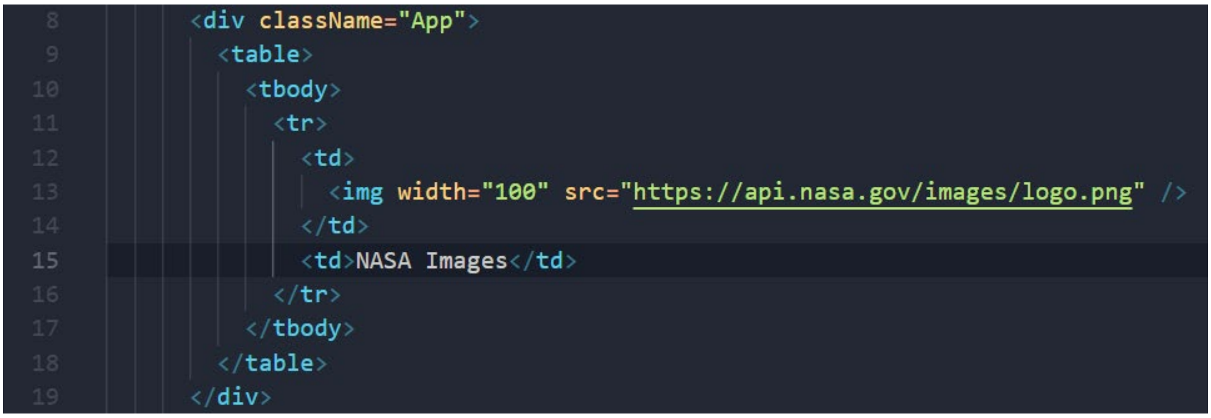
```
import React, { Component } from "react";
import logo from "./logo.svg";
import "./App.css";

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Edit <code>src/App.js</code> and save to reload.
          </p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          />
        </header>
      </div>
    );
  }
}

export default App;
```

9. To keep everything clean and aligned I have opted to use <table>
10. Back in the App.js file, create the header for the page using the table element that will hold the logo and title



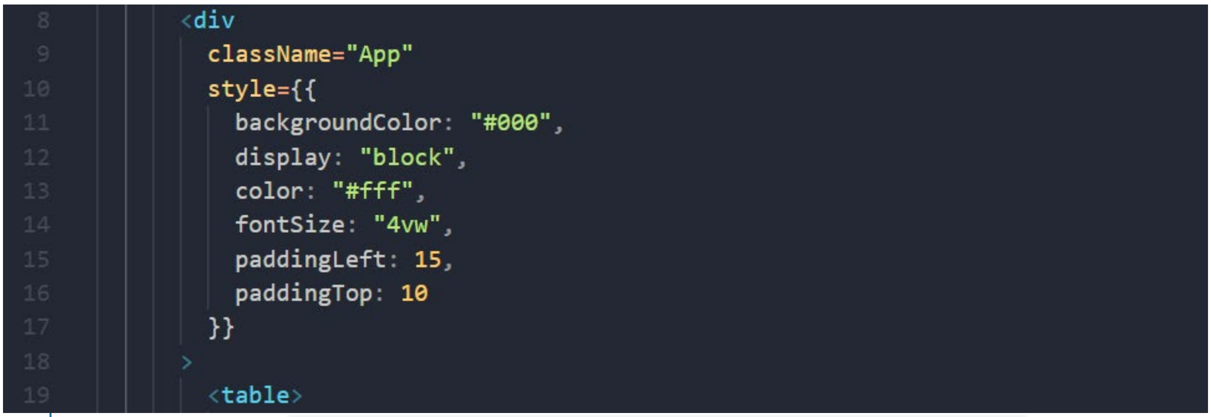
```
<div className="App">
  <table>
    <tbody>
      <tr>
        <td>dd</td>
        <td>aa</td>
      </tr>
    </tbody>
  </table>
</div>
```

11. Add the logo and the title in their respective cells

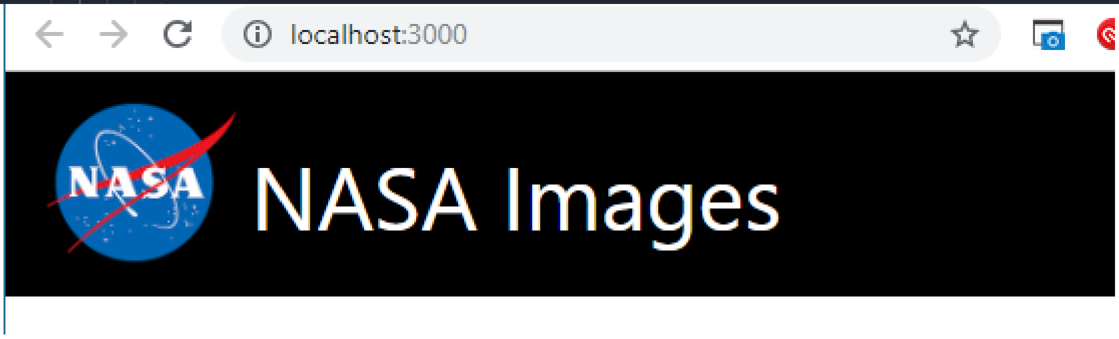


```
<div className="App">
  <table>
    <tbody>
      <tr>
        <td>
          <img width="100" src="https://api.nasa.gov/images/logo.png" />
        </td>
        <td>NASA Images</td>
      </tr>
    </tbody>
  </table>
</div>
```

12. Format the contents using the style tags

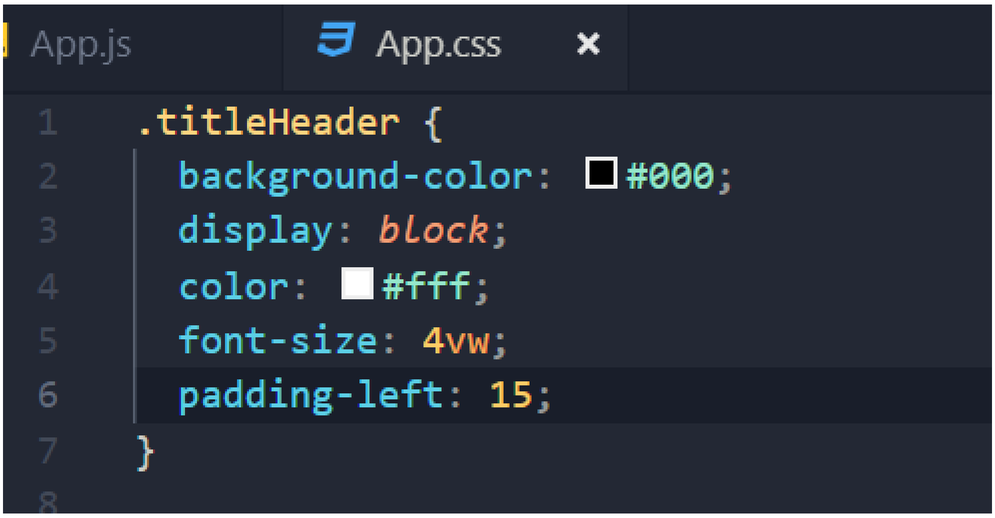


```
<div
  className="App"
  style={{
    backgroundColor: "#000",
    display: "block",
    color: "#fff",
    fontSize: "4vw",
    paddingLeft: 15,
    paddingTop: 10
  }}
>
  <table>
```



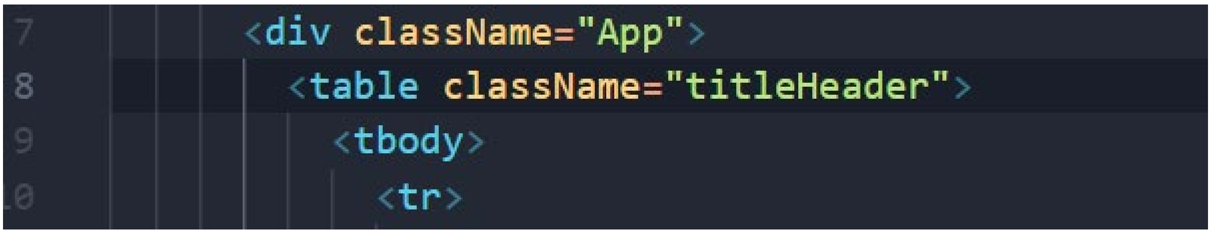
NASA Images

13. For aesthetics, the styling should be moved over to the App.css file and set it with the correct syntax



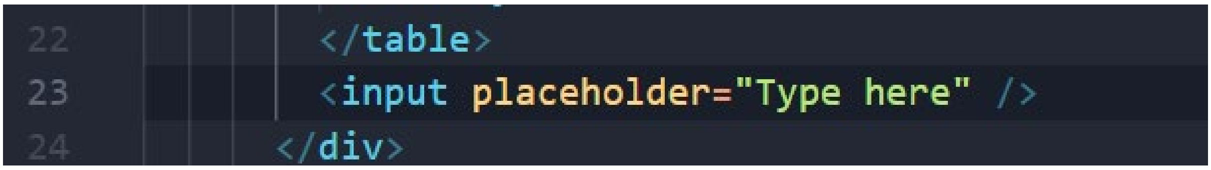
```css
.titleHeader {
  background-color: #000;
  display: block;
  color: #fff;
  font-size: 4vw;
  padding-left: 15;
}
```

14. Back in the App.js file, change the table class to the css class set in the css file, which is ‘titleHeader’



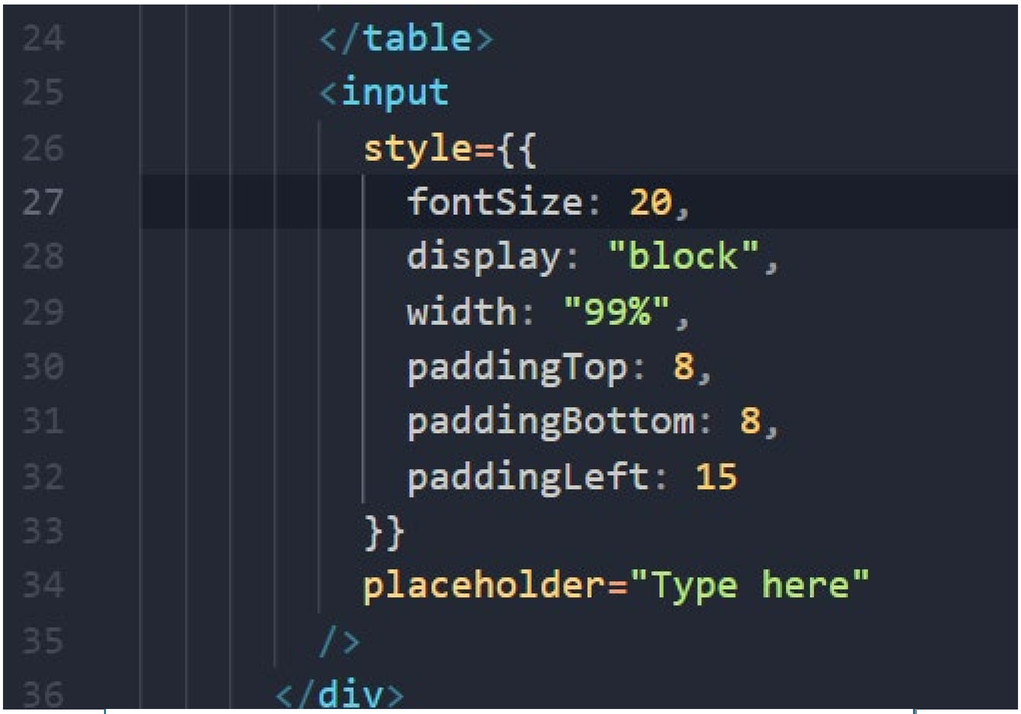
```jsx
<div className="App">
  <table className="titleHeader">
    <tbody>
      <tr>
```

15. To add a search bar, add input field under the table element

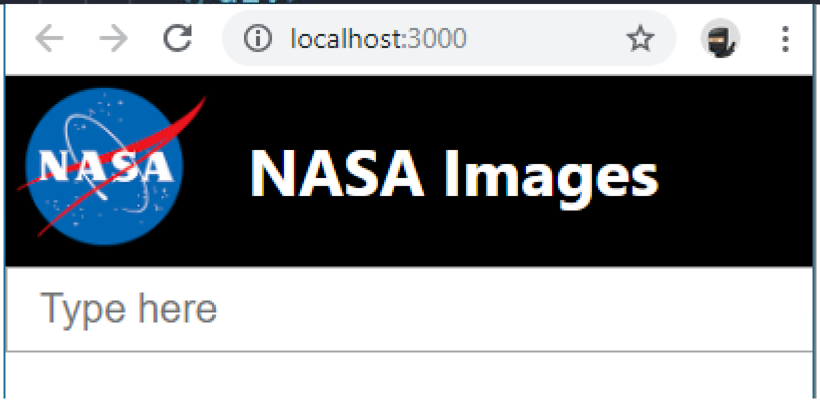


```jsx
    </table>
    <input placeholder="Type here" />
  </div>
```

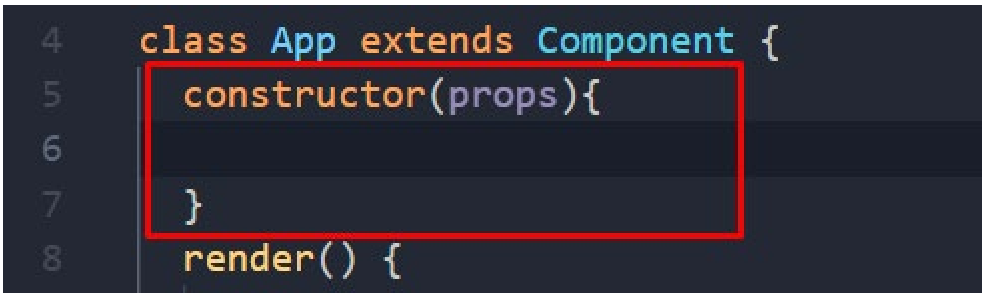
16. Add some styling to the search bar



```
          </table>
          <input
            style={{
              fontSize: 20,
              display: "block",
              width: "99%",
              paddingTop: 8,
              paddingBottom: 8,
              paddingLeft: 15
            }}
            placeholder="Type here"
          />
        </div>
```

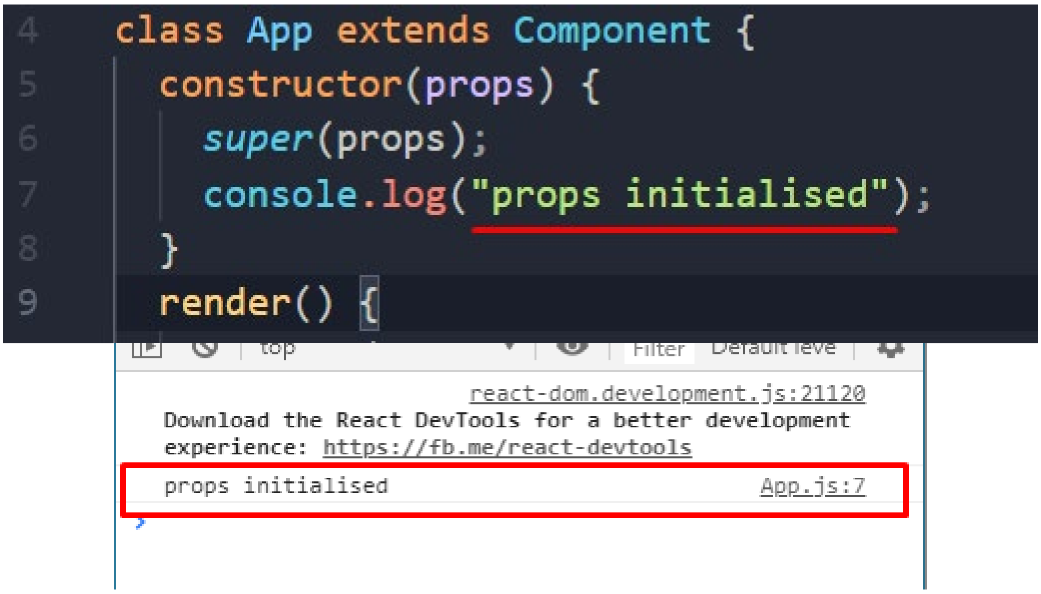


17. Now for the content, we need to start with some dummy content and create a component which will recreate each row for us according to the number of results.
    a. Create a new constructor here called props



```
class App extends Component {
  constructor(props){

  }
  render() {
```

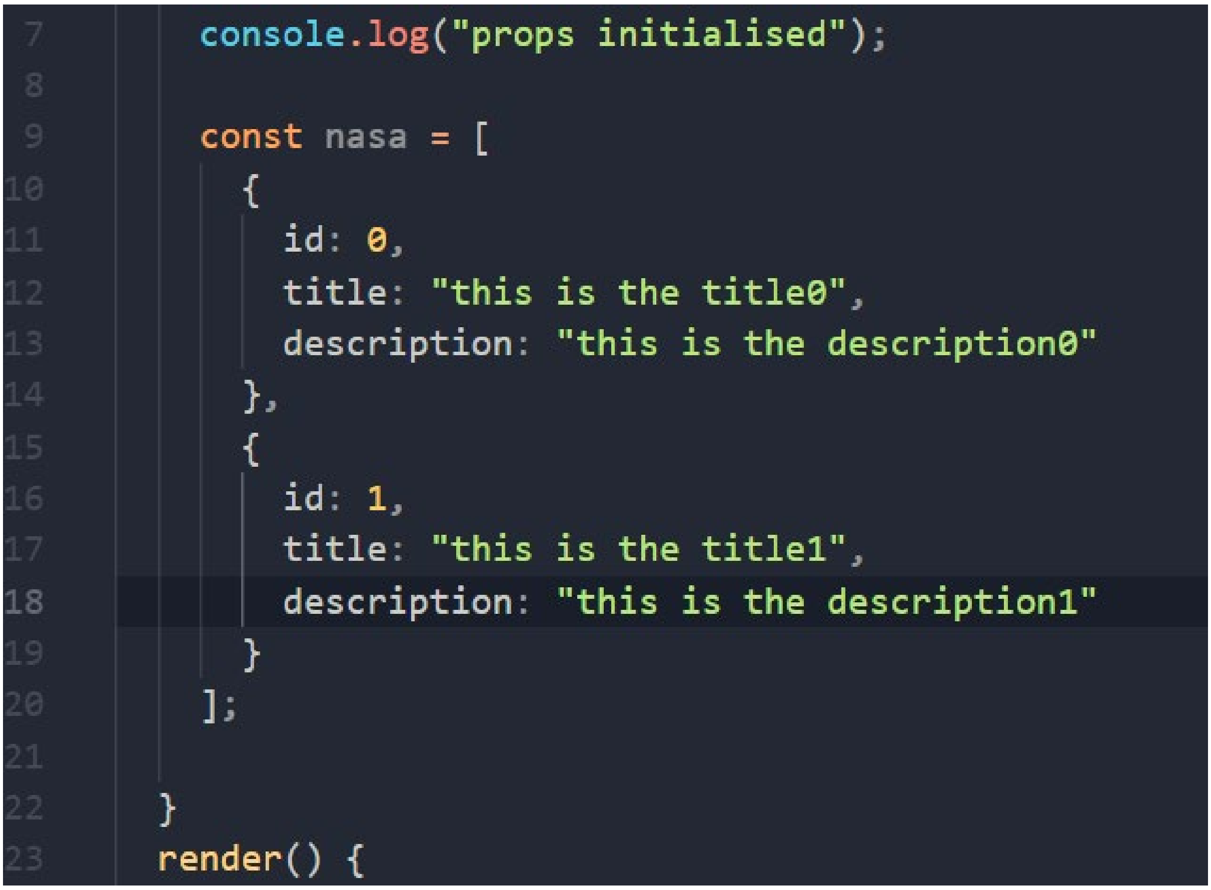
    b. Make sure what you have created works and appears on the console log



```
class App extends Component {
  constructor(props) {
    super(props);
    console.log("props initialised");
  }
  render() {
```

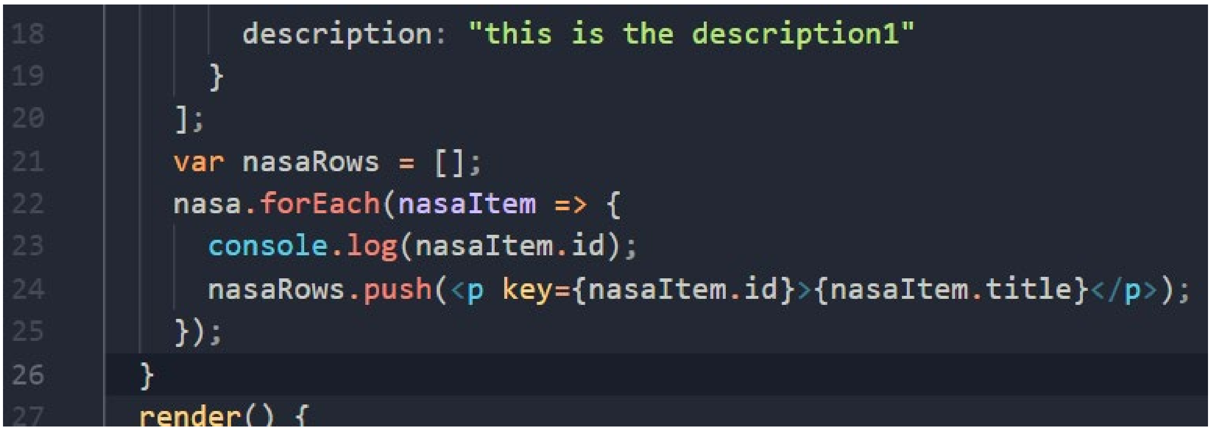
```
react-dom.development.js:21120
Download the React DevTools for a better development
experience: https://fb.me/react-devtools
props initialised                                App.js:7
```

c. Create a new variable for each item that will be pulled in via the API



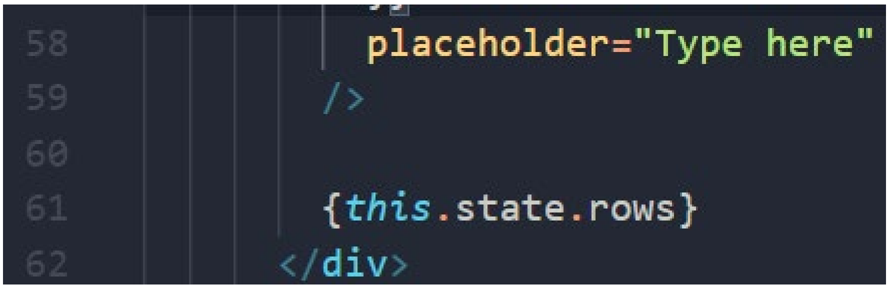
```
    console.log("props initialised");

    const nasa = [
      {
        id: 0,
        title: "this is the title0",
        description: "this is the description0"
      },
      {
        id: 1,
        title: "this is the title1",
        description: "this is the description1"
      }
    ];

  }
  render() {
```

d. Create a new variable to render each row for the results using the forEach loop



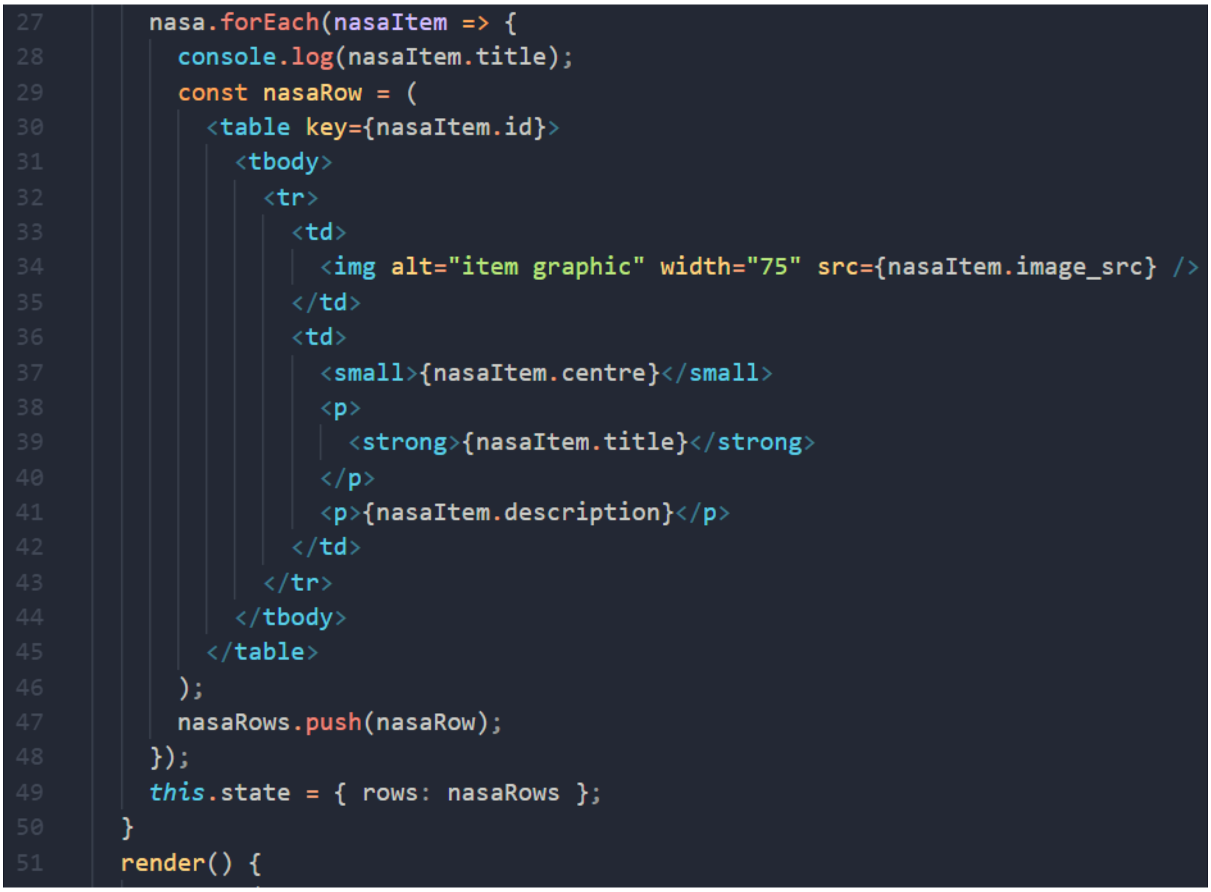
```
      description: "this is the description1"
    }
  ];
  var nasaRows = [];
  nasa.forEach(nasaItem => {
    console.log(nasaItem.id);
    nasaRows.push(<p key={nasaItem.id}>{nasaItem.title}</p>);
  });
}
render() {
```

e. Where I want the rows to appear in the body I add the following - {this.state.rows}



```
      placeholder="Type here"
    />

    {this.state.rows}
  </div>
```

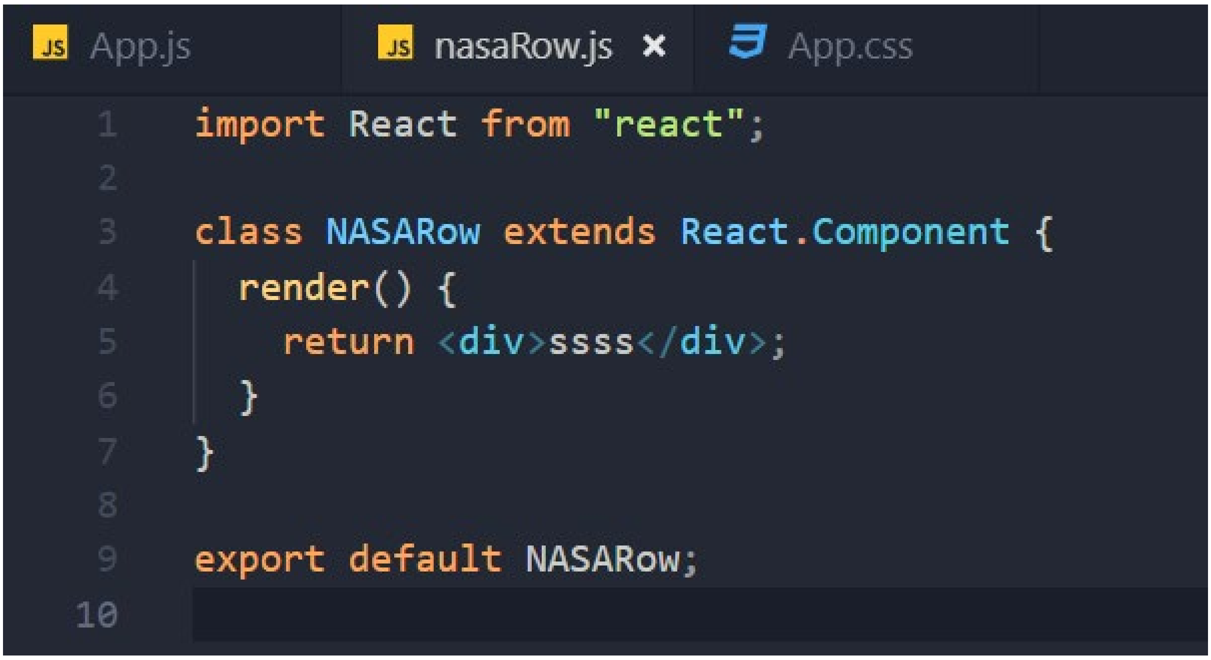
f. Format the rows so that they display the image, title, description, centre



```
  nasa.forEach(nasaItem => {
    console.log(nasaItem.title);
    const nasaRow = (
      <table key={nasaItem.id}>
        <tbody>
          <tr>
            <td>
              <img alt="item graphic" width="75" src={nasaItem.image_src} />
            </td>
            <td>
              <small>{nasaItem.centre}</small>
              <p>
                <strong>{nasaItem.title}</strong>
              </p>
              <p>{nasaItem.description}</p>
            </td>
          </tr>
        </tbody>
      </table>
    );
    nasaRows.push(nasaRow);
  });
  this.state = { rows: nasaRows };
}
render() {
```

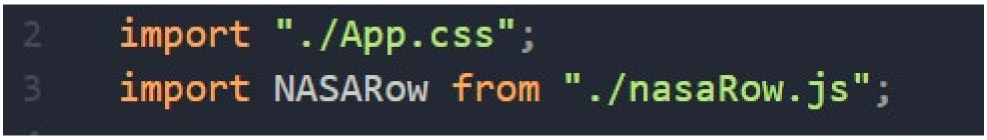
g. To tidy the file up create a new file called nasaRow.js

h. Import the dependencies and create a new class called NASARow add a render component with div tags and text in there



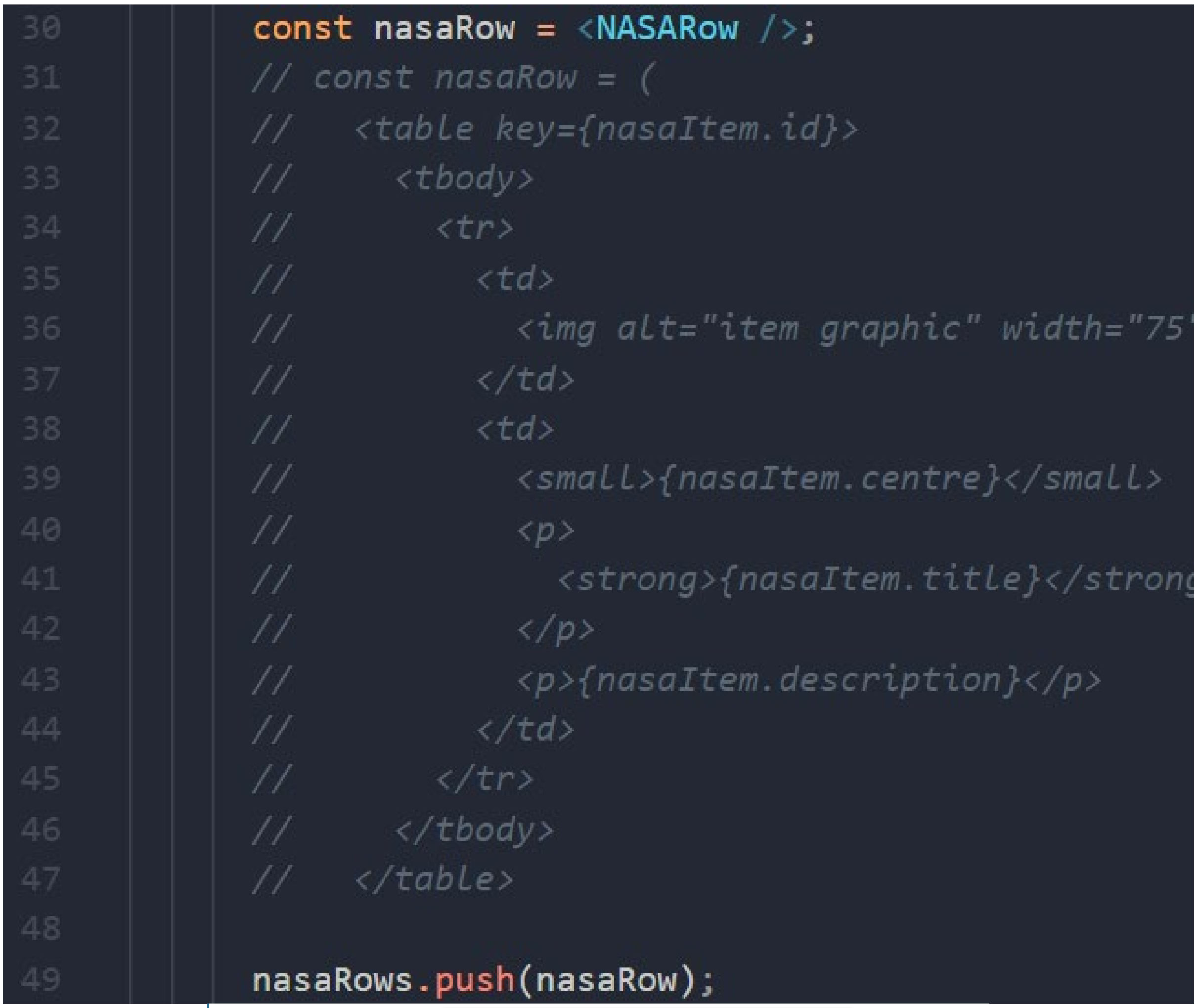
```
import React from "react";

class NASARow extends React.Component {
  render() {
    return <div>ssss</div>;
  }
}

export default NASARow;

```

i. Import the file created into the App.js file -
   i. Add → import NASARow from "./nasaRow.js";

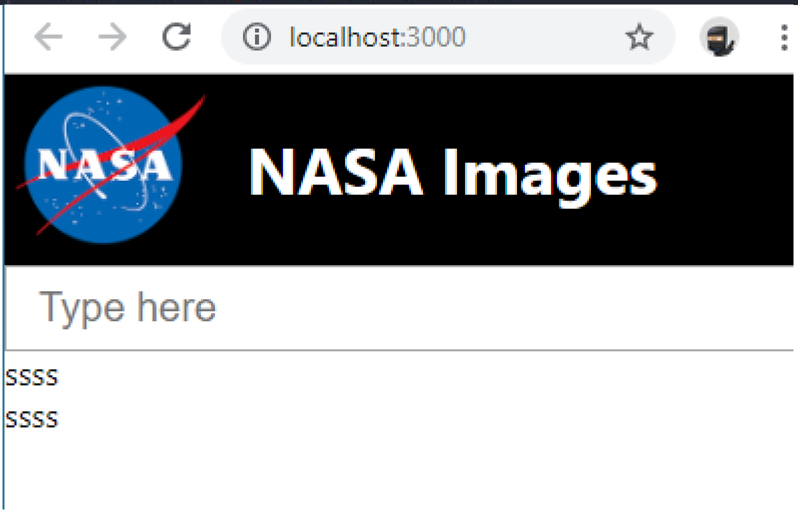


```
import "./App.css";
import NASARow from "./nasaRow.js";
```

j. Comment out the nasaRow const and replace it with the new NASARow and test to make sure it displays correctly and the file is rendered



```
      const nasaRow = <NASARow />;
      // const nasaRow = (
      //   <table key={nasaItem.id}>
      //     <tbody>
      //       <tr>
      //         <td>
      //           <img alt="item graphic" width="75'
      //         </td>
      //         <td>
      //           <small>{nasaItem.centre}</small>
      //           <p>
      //             <strong>{nasaItem.title}</strong
      //           </p>
      //           <p>{nasaItem.description}</p>
      //         </td>
      //       </tr>
      //     </tbody>
      //   </table>

      nasaRows.push(nasaRow);
```

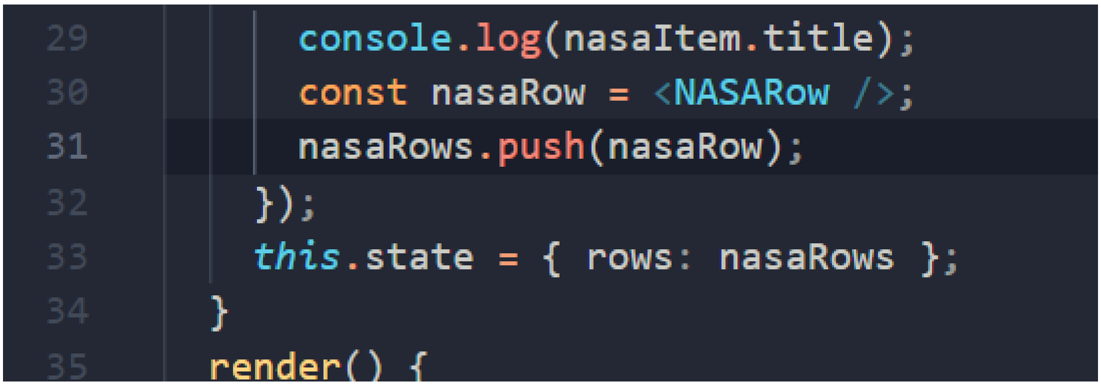


```
ssss
ssss
```

k. Uncomment the nasaRow const and cut the table element out

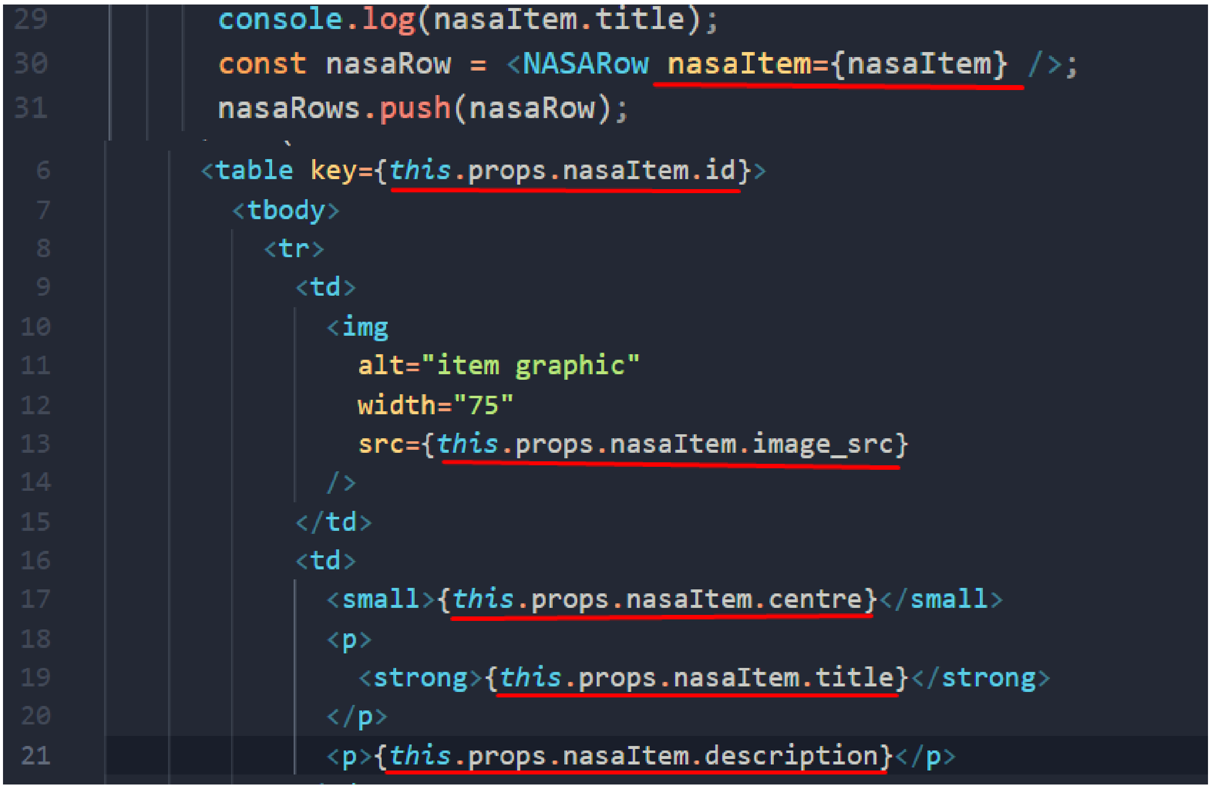
l. Paste it into nasaRow.js replacing the div tags

m. The nasaRow.js file classes now need to be linked to the App.js class by adding it to the const.



```
      console.log(nasaItem.title);
      const nasaRow = <NASARow />;
      nasaRows.push(nasaRow);
    });
    this.state = { rows: nasaRows };
  }
  render() {
```

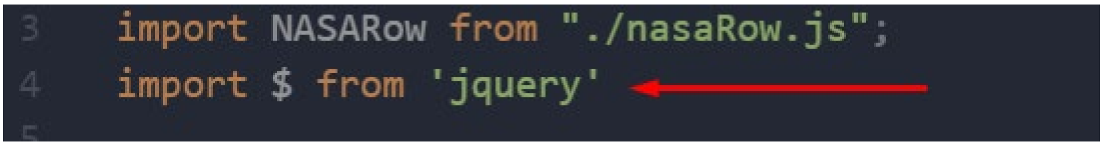
   n. Prefix ‘ this.props. ‘ in front of all ‘nasaItem.’ on the nasaRow.js file



```
      console.log(nasaItem.title);
      const nasaRow = <NASARow nasaItem={nasaItem} />;
      nasaRows.push(nasaRow);
```

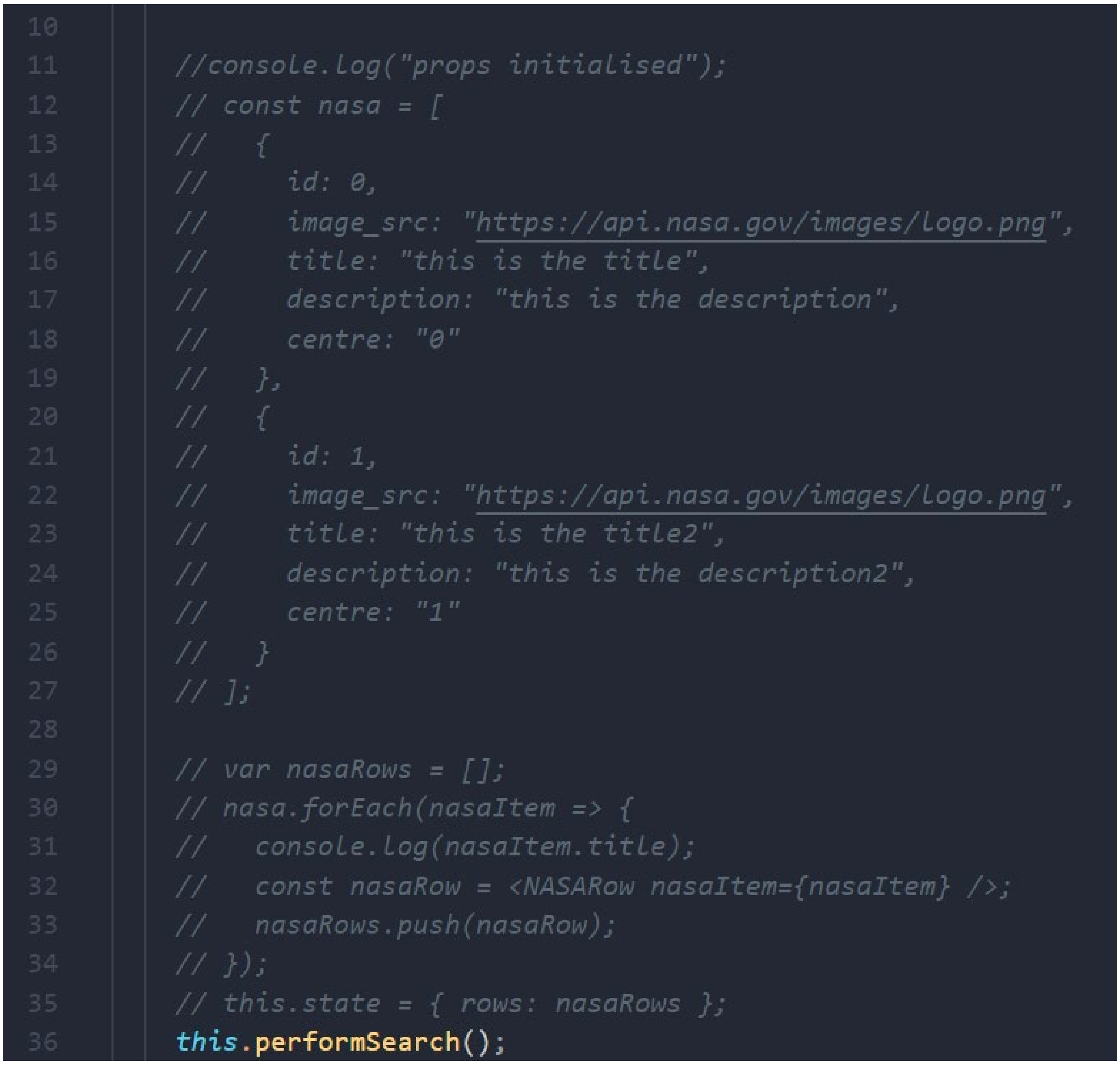
```
      <table key={this.props.nasaItem.id}>
        <tbody>
          <tr>
            <td>
              <img
                alt="item graphic"
                width="75"
                src={this.props.nasaItem.image_src}
              />
            </td>
            <td>
              <small>{this.props.nasaItem.centre}</small>
              <p>
                <strong>{this.props.nasaItem.title}</strong>
              </p>
              <p>{this.props.nasaItem.description}</p>
```

18. Now to hook up the API get the API search string such as for “apollo” - https://images-api.nasa.gov/search?q=apollo
19. To add the search component I will utilise Jquery, so I need to install the jquery package for React to use
    a. Run → npm i jquery
    b. Add → import $ from "jquery"; to the App.js header



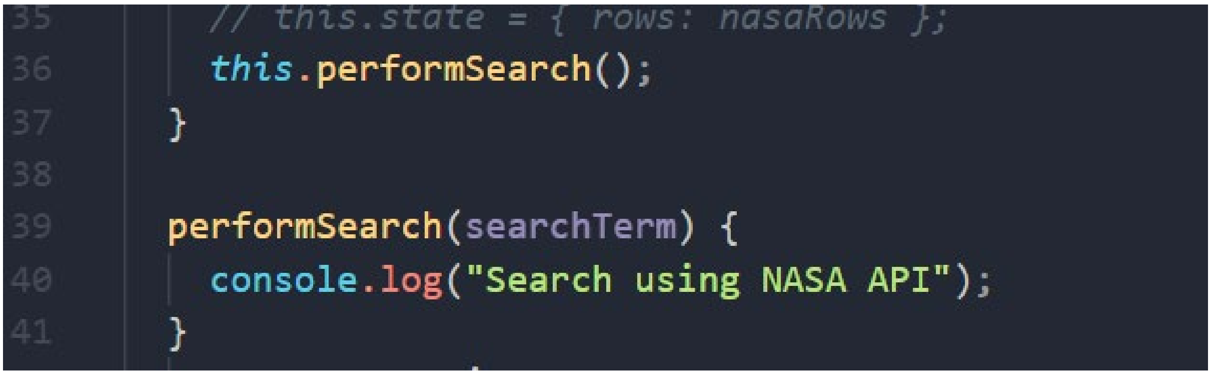
```
import NASARow from "./nasaRow.js";
import $ from 'jquery'
```

20. Next in the App.js file I need to comment out the dummy data and I need to create a new method called ‘performSearch’ within the props constructor



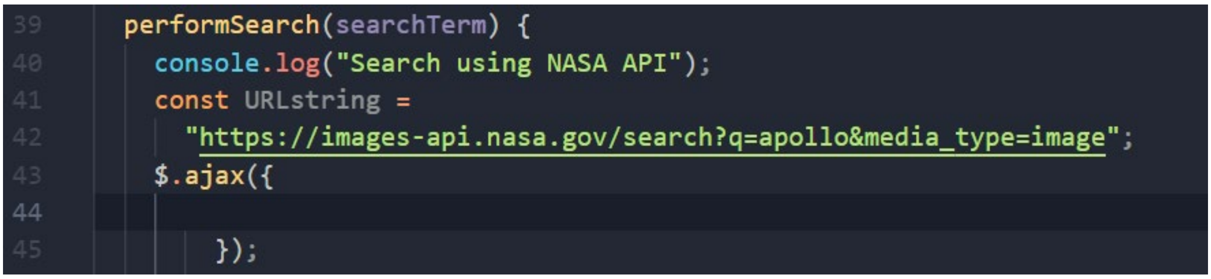
```
    //console.log("props initialised");
    // const nasa = [
    //   {
    //     id: 0,
    //     image_src: "https://api.nasa.gov/images/logo.png",
    //     title: "this is the title",
    //     description: "this is the description",
    //     centre: "0"
    //   },
    //   {
    //     id: 1,
    //     image_src: "https://api.nasa.gov/images/logo.png",
    //     title: "this is the title2",
    //     description: "this is the description2",
    //     centre: "1"
    //   }
    // ];

    // var nasaRows = [];
    // nasa.forEach(nasaItem => {
    //   console.log(nasaItem.title);
    //   const nasaRow = <NASARow nasaItem={nasaItem} />;
    //   nasaRows.push(nasaRow);
    // });
    // this.state = { rows: nasaRows };
    this.performSearch();
```

21. Next drop out of the props constructor and create a new function with the same name.



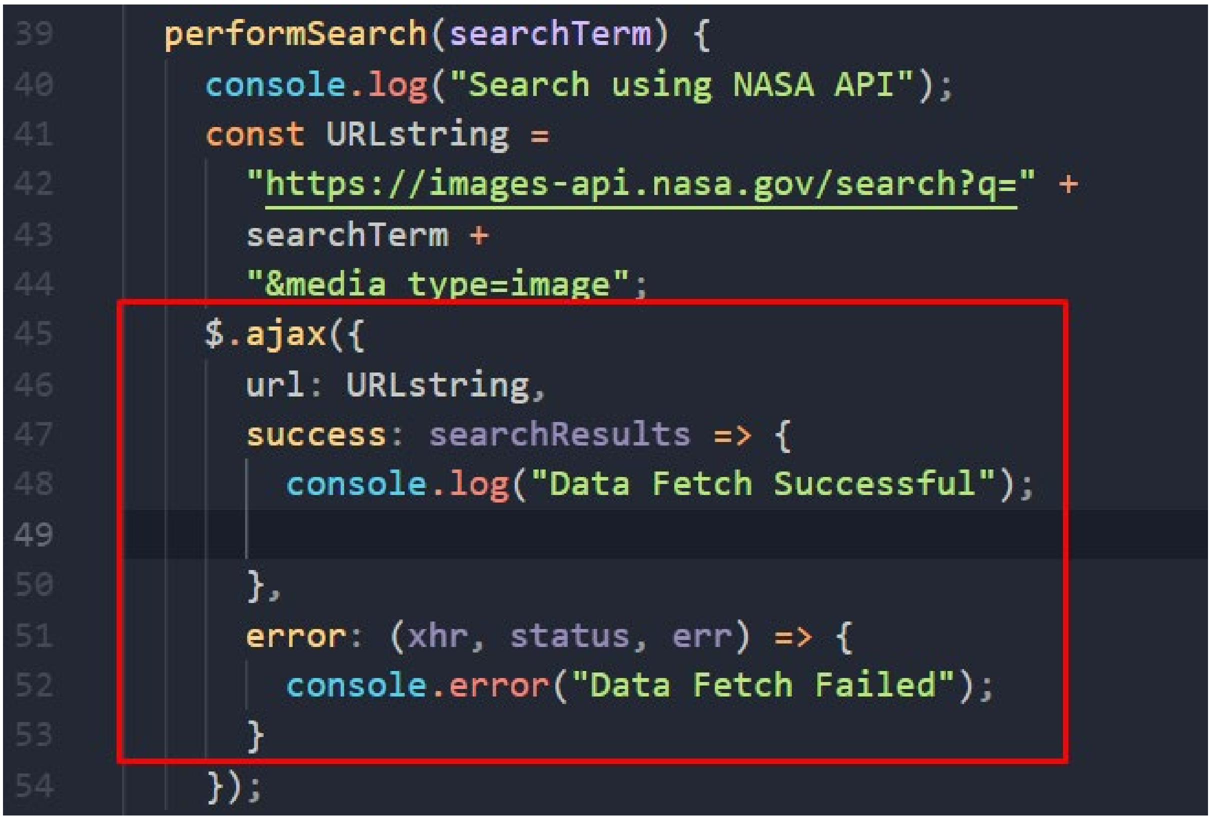
```
    // this.state = { rows: nasaRows };
    this.performSearch();
  }

  performSearch(searchTerm) {
    console.log("Search using NASA API");
  }
```

22. Next add a constant variable called URLstring and add the API link.



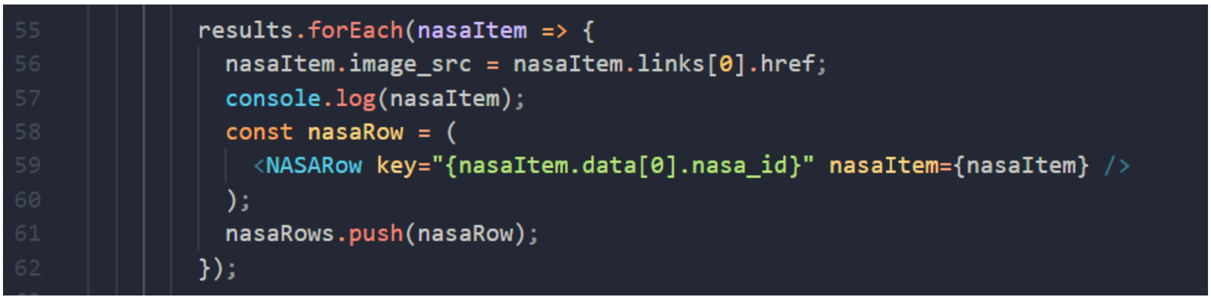
```
performSearch(searchTerm) {
  console.log("Search using NASA API");
  const URLstring =
    "https://images-api.nasa.gov/search?q=apollo&media_type=image";
  $.ajax({

    });
```

23. I am using some Ajax to make asynchronous calls via the search box so the results will be instantaneous
24. Within the ajax I also added data fetch status on the console log to be able to see if the search box is working



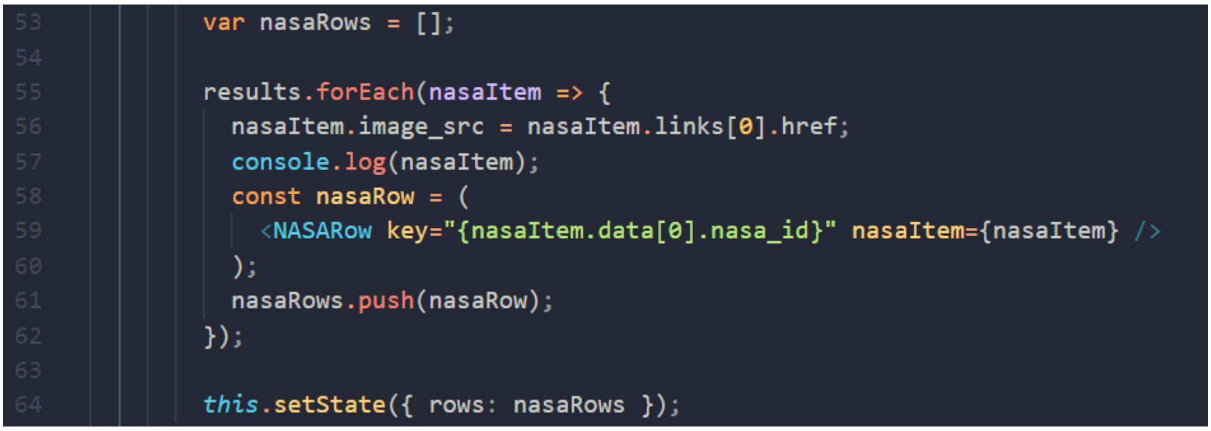
```
performSearch(searchTerm) {
  console.log("Search using NASA API");
  const URLstring =
    "https://images-api.nasa.gov/search?q=" +
    searchTerm +
    "&media_type=image";
  $.ajax({
    url: URLstring,
    success: searchResults => {
      console.log("Data Fetch Successful");

    },
    error: (xhr, status, err) => {
      console.error("Data Fetch Failed");
    }
  });
```

25. With the data coming through fine in the console log, I can now map the data correctly to my row data and create a forEach loop so each item is rendered separately.



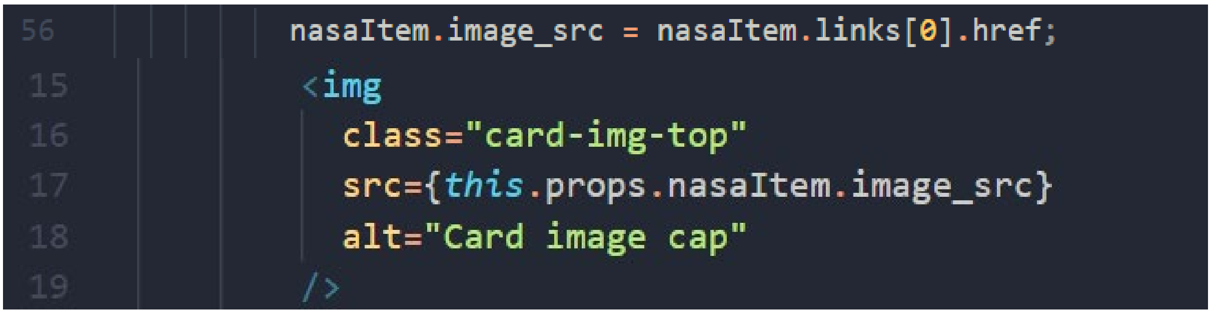
```
results.forEach(nasaItem => {
  nasaItem.image_src = nasaItem.links[0].href;
  console.log(nasaItem);
  const nasaRow = (
    <NASARow key="{nasaItem.data[0].nasa_id}" nasaItem={nasaItem} />
  );
  nasaRows.push(nasaRow);
});
```

26. To display each item separately I created another variable called nasaRows which pulled and pushed the data known as nasaRow to the relevant placeholders



```
var nasaRows = [];

results.forEach(nasaItem => {
  nasaItem.image_src = nasaItem.links[0].href;
  console.log(nasaItem);
  const nasaRow = (
    <NASARow key="{nasaItem.data[0].nasa_id}" nasaItem={nasaItem} />
  );
  nasaRows.push(nasaRow);
});

this.setState({ rows: nasaRows });
```

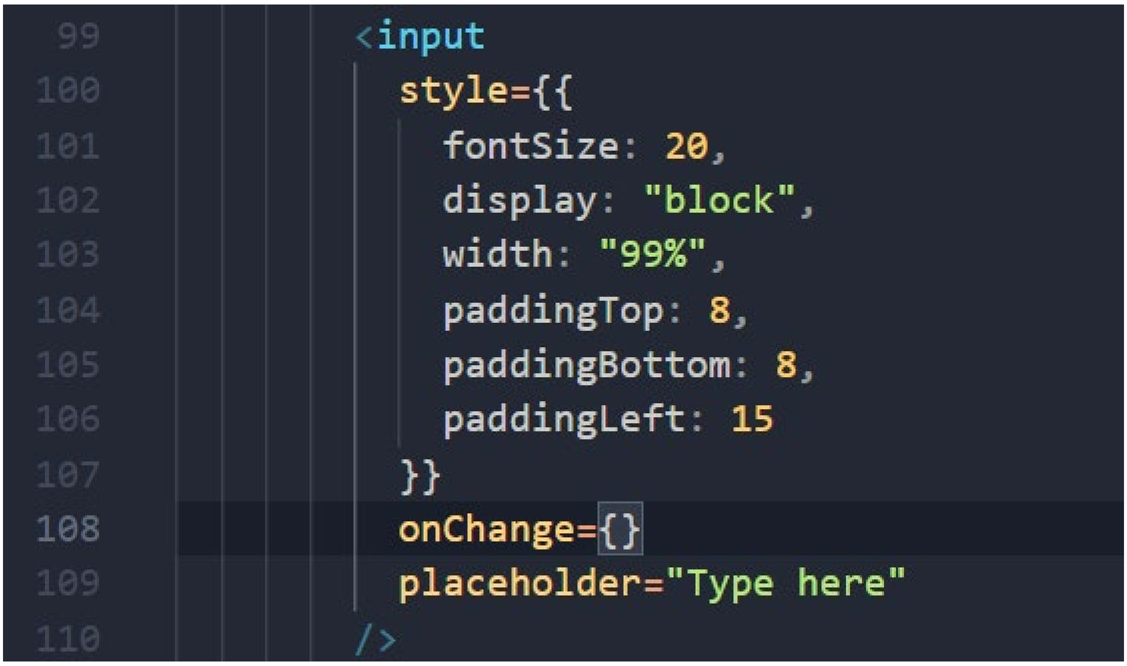
27. I noticed that the image thumbnail for each item was declared in a separate item, instead of it being in ‘data’ with the rest of the information, iot was in ‘links’ for this I had to declare the location of the image separately and call it back in



```
nasaItem.image_src = nasaItem.links[0].href;
```

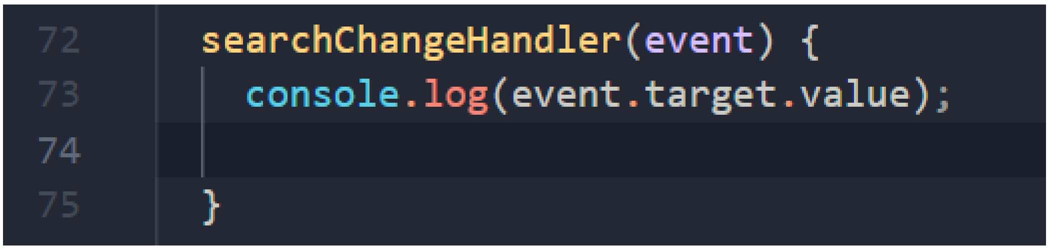
```
<img
  class="card-img-top"
  src={this.props.nasaItem.image_src}
  alt="Card image cap"
/>
```

28. With the row data now set, next thing to do is to get the search terms working, for this I added the onChange handler to the input box



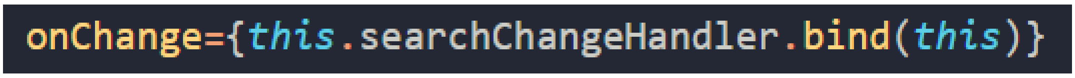
```
<input
  style={{
    fontSize: 20,
    display: "block",
    width: "99%",
    paddingTop: 8,
    paddingBottom: 8,
    paddingLeft: 15
  }}
  onChange={}
  placeholder="Type here"
/>
```

29. I created a new function called searchChangeHandler to handle this



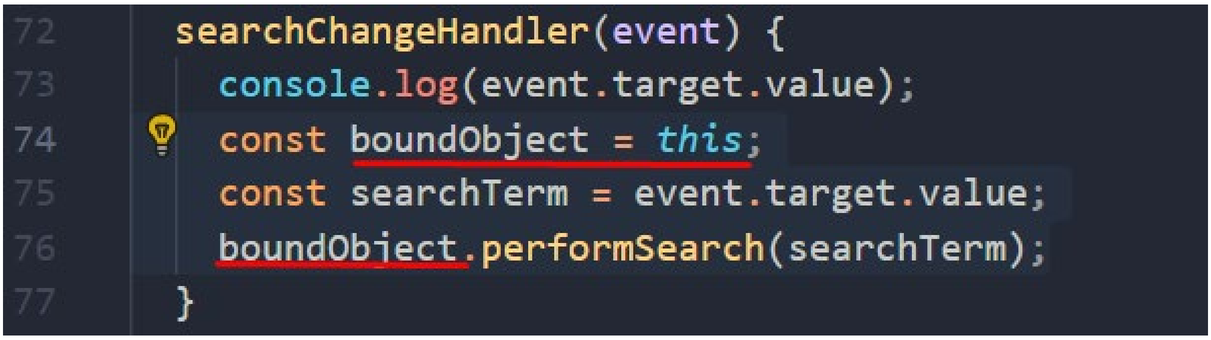
```
searchChangeHandler(event) {
  console.log(event.target.value);

}
```

30. The searchChangeHandler function and the onChange handler are not within the same method therefore I need to bind both these together. For this I used ‘.bind(this)



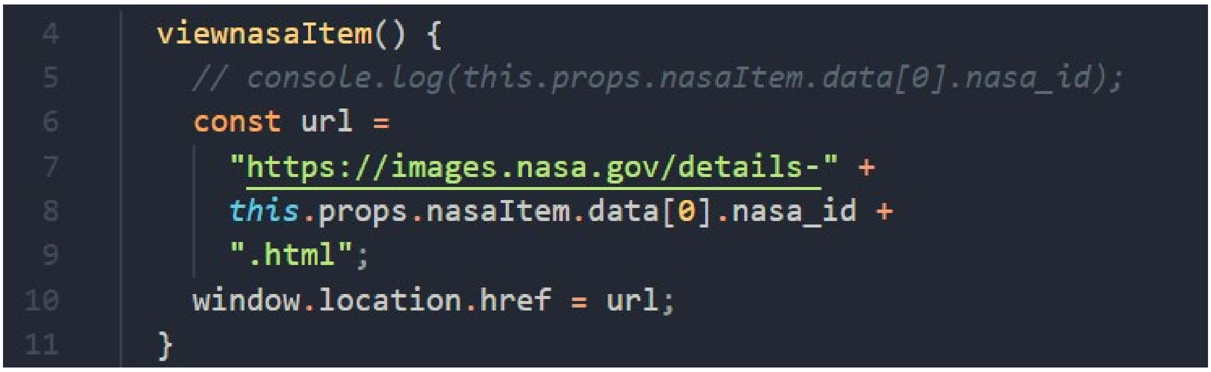
```
onChange={this.searchChangeHandler.bind(this)}
```

31. I added a new constant variable called boundObject and linked it to ‘this’



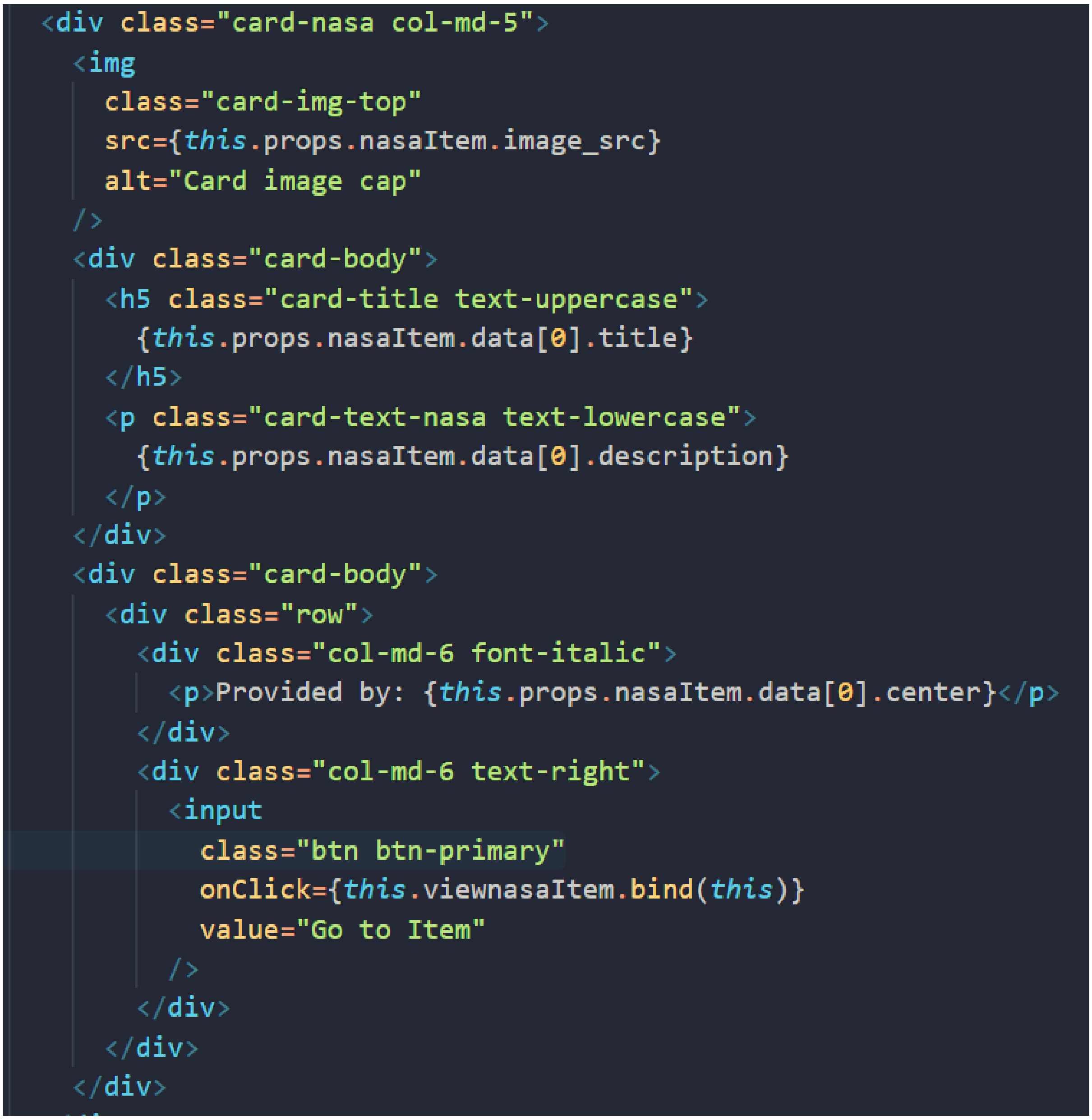
```
searchChangeHandler(event) {
  console.log(event.target.value);
  const boundObject = this;
  const searchTerm = event.target.value;
  boundObject.performSearch(searchTerm);
}
```

32. Now to add a button to go to the destination for this again I created a input box, added onClick with this.viewnasaItem.bind(this) and created another const to serve a window.location.href js



```
viewnasaItem() {
  // console.log(this.props.nasaItem.data[0].nasa_id);
  const url =
    "https://images.nasa.gov/details-" +
    this.props.nasaItem.data[0].nasa_id +
    ".html";
  window.location.href = url;
}
```

33. With the search now working, time to beautify the page.
34. I like working with Bootstrap and Material designs, therefore I used Bootstrap Card elements for this



```
<div class="card-nasa col-md-5">
  <img
    class="card-img-top"
    src={this.props.nasaItem.image_src}
    alt="Card image cap"
  />
  <div class="card-body">
    <h5 class="card-title text-uppercase">
      {this.props.nasaItem.data[0].title}
    </h5>
    <p class="card-text-nasa text-lowercase">
      {this.props.nasaItem.data[0].description}
    </p>
  </div>
  <div class="card-body">
    <div class="row">
      <div class="col-md-6 font-italic">
        <p>Provided by: {this.props.nasaItem.data[0].center}</p>
      </div>
      <div class="col-md-6 text-right">
        <input
          class="btn btn-primary"
          onClick={this.viewnasaItem.bind(this)}
          value="Go to Item"
        />
      </div>
    </div>
  </div>
```