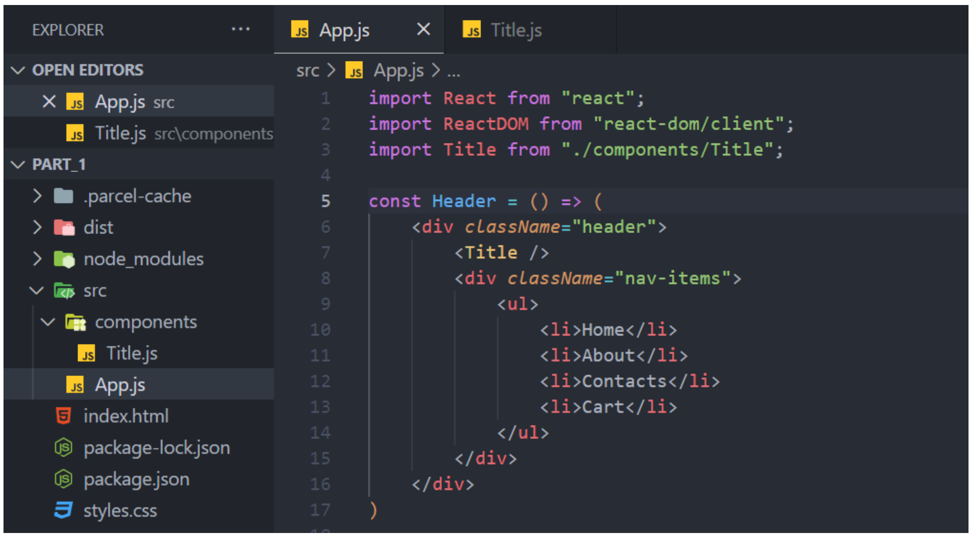
#NOTES :-

- An interesting convo why React Fragments are used :- Link
- A good webapp to share your code :- Link. Just copy your code, paste it and export it to get the image.
- Some JS concepts that are important in React :- Link
- Destructuring of props in both Functional and Class Components :- Link
- Learn Git visually from Link
- If you have agreat understanding of the React Fiber, then read this Link to get a good idea about how ReactDOM render function works.
- This article offers a great visual representation of how Virtual DOM works. Read it only if you have time.
- Read more about React Fiber from this Link.

Food Villa App

Now, we made another folder named "**src**" in the root directory, because we want to wrap our code in a proper folder structure. This make our code modular i.e. it gives us a more modular approach. Remember that this is just a convention.

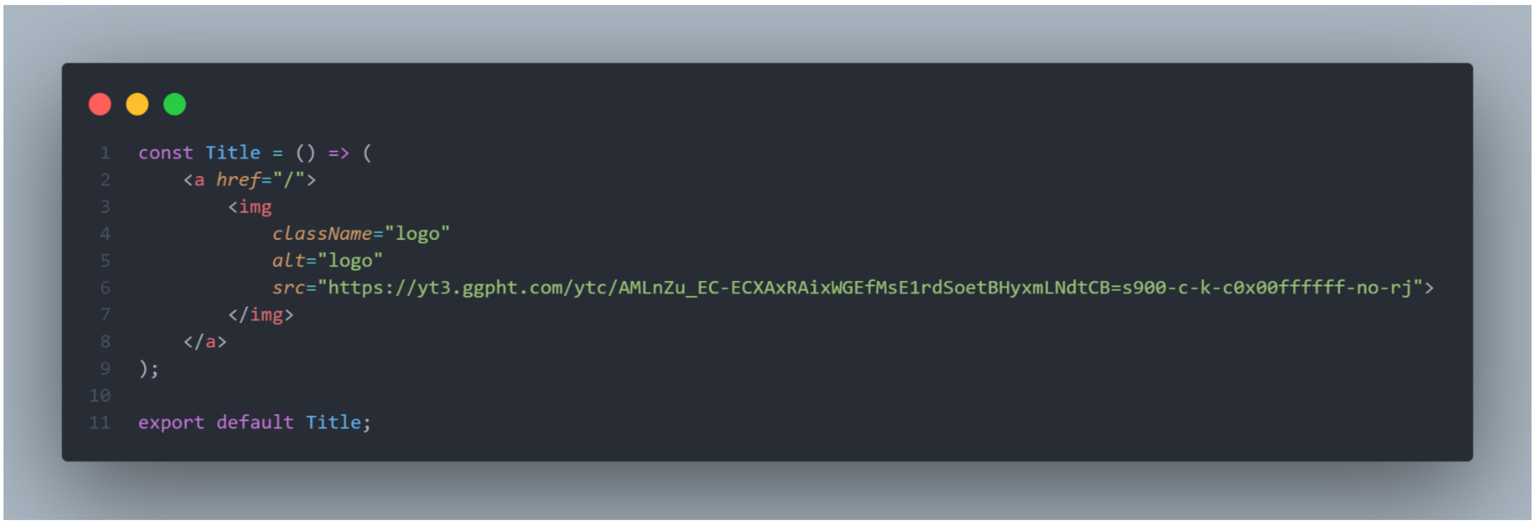
Moreover, we create a "**components"** folder to store all the different component files. So, need to import each component file correctly in App.js because they are in different directories now.



```
import React from "react";
import ReactDOM from "react-dom/client";
import Title from "./components/Title";

const Header = () => (
    <div className="header">
        <Title />
        <div className="nav-items">
            <ul>
                <li>Home</li>
                <li>About</li>
                <li>Contacts</li>
                <li>Cart</li>
            </ul>
        </div>
    </div>
)
```

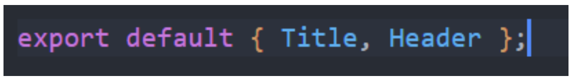
But this alone will not work because we also have to export the functional component ,"Title", from the Title.js file. We can export a component in 2 ways :-

i. Using **export default funcCompName;**



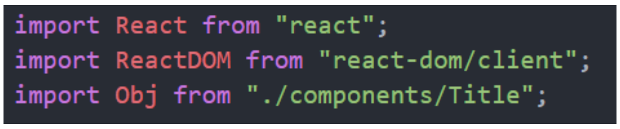
```
const Title = () => (
    <a href="/">
        <img
            className="logo"
            alt="logo"
            src="https://yt3.ggpht.com/ytc/AMLnZu_EC-ECXAxRAixWGEfMsE1rdSoetBHyxmLNdtCB=s900-c-k-c0x00ffffff-no-rj">
        </img>
    </a>
);

export default Title;
```

Now why are we using the keyword "default" or what does it mean? This is because there is another way to export components (discussed later). Using default, we can import a file by **Default import.** Also, if we name the imported file as "MyTitle" instead of "Title", then also it works fine, since it is a default export. We can only export 1 thing by default. So, to **export multiple Functional components** we can export like :-

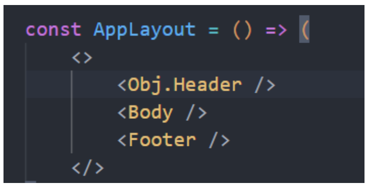


```
export default { Title, Header };
```

and import and use it like :-

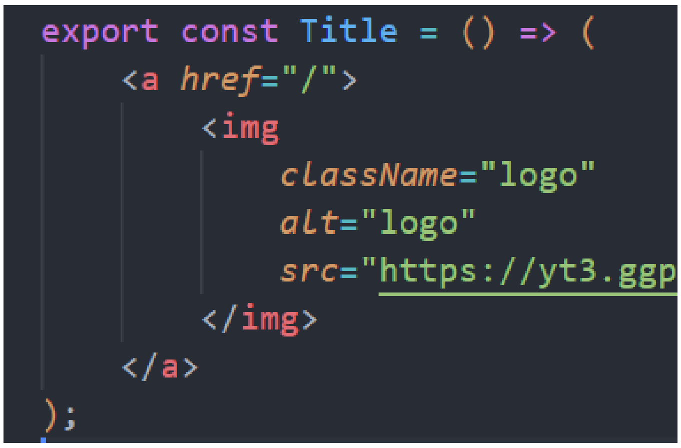


```
import React from "react";
import ReactDOM from "react-dom/client";
import Obj from "./components/Title";
```

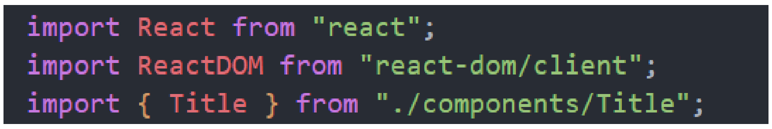


```
const AppLayout = () => (
  <>
    <Obj.Header />
    <Body />
    <Footer />
  </>
```

ii. Using **named export and named import :-**



```
export const Title = () => (
  <a href="/">
    <img
      className="logo"
      alt="logo"
      src="https://yt3.ggp
    </img>
  </a>
);
```



```
import React from "react";
import ReactDOM from "react-dom/client";
import { Title } from "./components/Title";
```

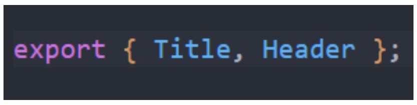
DO NOT IMPORT WITHOUT USING CURLY BRACES

Here we have to import the component with same name as it was in the file from which it was exported. Note that when we are doing "{ Title }" during import, its not the same as destructuring an object because if we import like :-

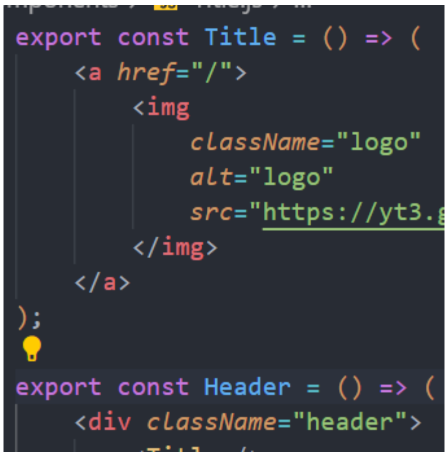
import Obj from "./components/Title";

Then doing Obj.Title will give us an error

To **export multiple functional components**, we do either of the below 2 :-



```
export { Title, Header };
```

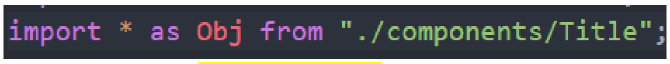


```
export const Title = () => (
  <a href="/">
    <img
      className="logo"
      alt="logo"
      src="https://yt3.
    </img>
  </a>
);

export const Header = () => (
  <div className="header">
```

However, during importing them, we cannot just say :-

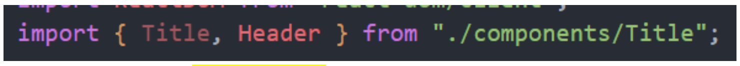
import Obj from "./components/Title";

Because as I said earlier, during named exports, we are not exporting components as objects, so we do any one of the below 2:-



```
import * as Obj from "./components/Title";
```

and use it like :- <Obj.Header />

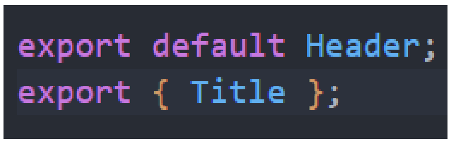


```
import { Title, Header } from "./components/Title";
```

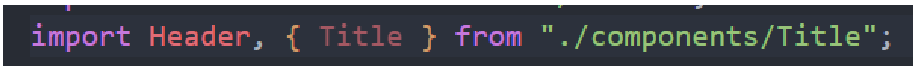
and use it like :- <Header />

#NOTE :-

We can also export by default and name in the same file, and import them in the same line BUT the default import should be placed before the named import.



```
export default Header;
export { Title };
```



```
import Header, { Title } from "./components/Title";
```

Search Bar creation :-

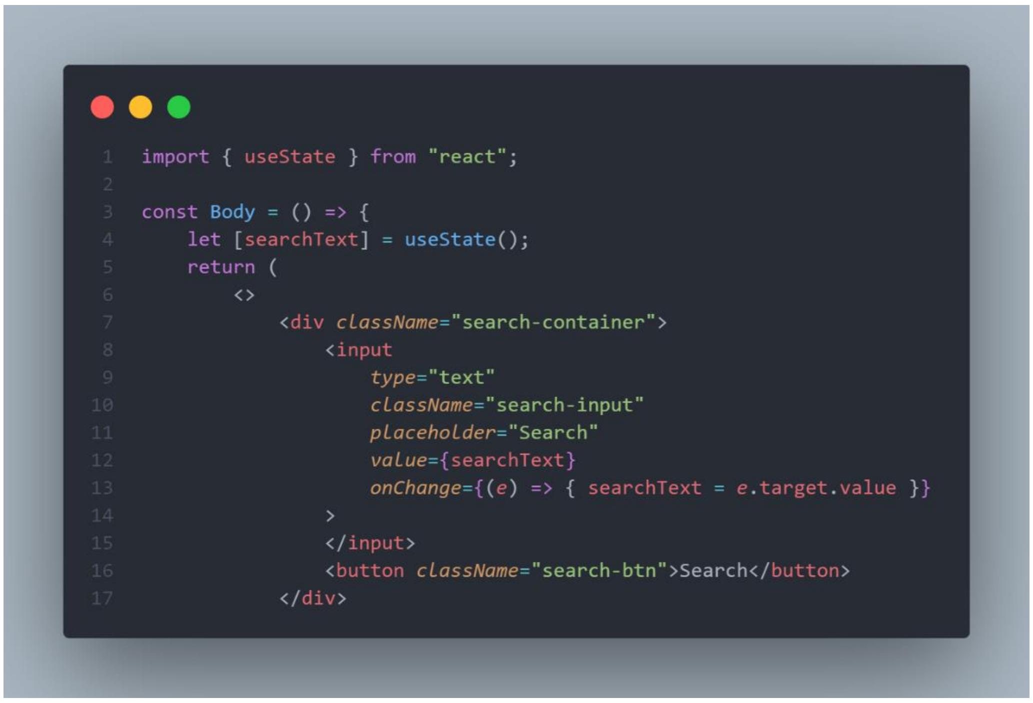
See the video from 00:52:14 to 1:11:49. Also, get a overview from this Link
While creating a Search Bar, we can use the input. But you will see that if we wanted to write in that input box formed by the normal input tag, then it won't work, even that's how we used to do it in HTML because form elements like <input>, <textarea>,<select> has their own states and update it based on user inputs. But, in React the state is handled differently because every component in React maintains a state and we can put a variable onto that state and everytime we use a local variable in React, we need to use state variables and these state variables are created using a useState Hook.

React Hooks :- They are nothing but functions. But every hook function has a special use-case. For ex :- useState() hook is used to create state variables. We can all the useState hook by :- import { useState } from "react";
The function useState(), returns an array with it's first variable as our local state variable. We use it like :-

const [local_variable] = useState(default_val);

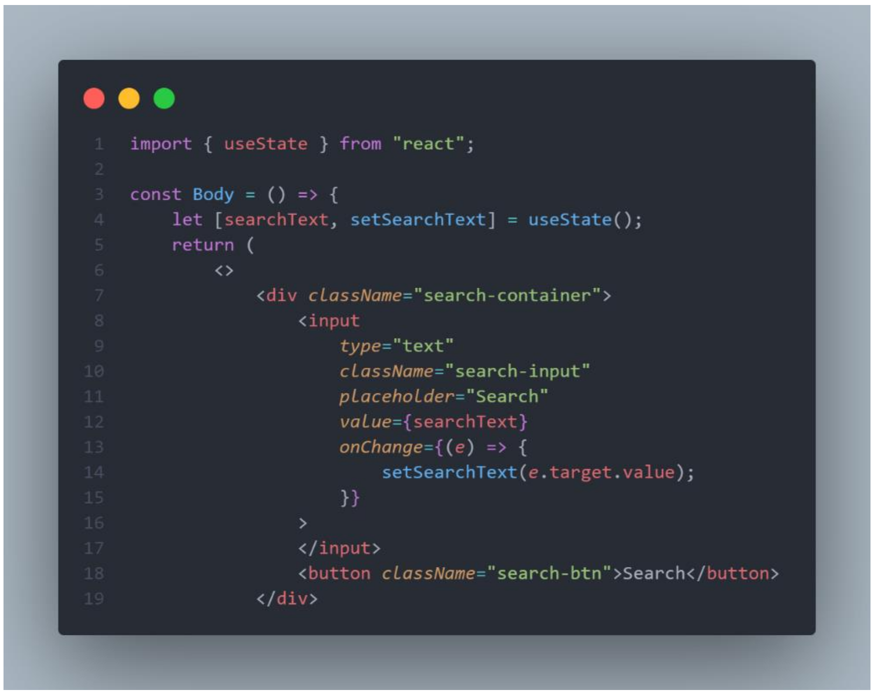
(The default value is optional.)

Then we can use it like :-



```
import { useState } from "react";

const Body = () => {
    let [searchText] = useState();
    return (
        <>
            <div className="search-container">
                <input
                    type="text"
                    className="search-input"
                    placeholder="Search"
                    value={searchText}
                    onChange={(e) => { searchText = e.target.value }}
                >
                </input>
                <button className="search-btn">Search</button>
            </div>
```

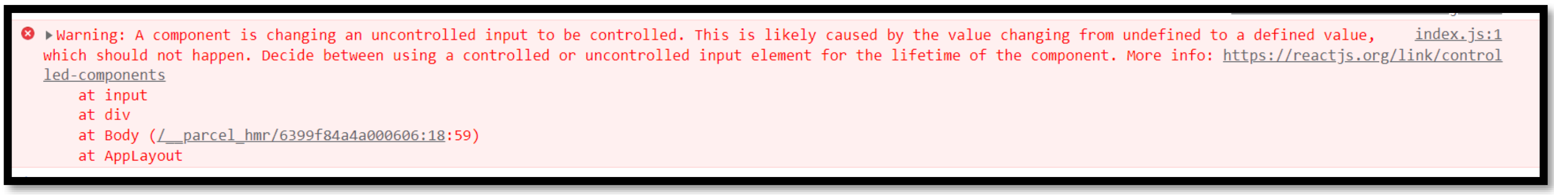
The onChange attribute is used to determine what to do when there is a change in the text in the input-box using the event variable(e) provided by that attribute itself. (you can also do "console.log(e.target.value)" to see that what the user gives input in the input-box, gets printed in the console). To reflect the change in the user input in the search-box too, we store that change in the searchText variable and that variable is passed to the value attribute.
Now, the above code will also give desirable results. But, we modify the state variables using a set function provided to us by the useState() and in the on onChange attribute using that function to actually modify the state variable like :-



```
import { useState } from "react";

const Body = () => {
    let [searchText, setSearchText] = useState();
    return (
        <>
            <div className="search-container">
                <input
                    type="text"
                    className="search-input"
                    placeholder="Search"
                    value={searchText}
                    onChange={(e) => {
                        setSearchText(e.target.value);
                    }}
                >
                </input>
                <button className="search-btn">Search</button>
            </div>
```

Important points regarding useState Hooks :-

- The normal industry convention for creating a set function for the state variable is using the prefix “set”.
- If we do not give a default value to a state variable the moment we change the state variable in the UI, it will throw a warning :-

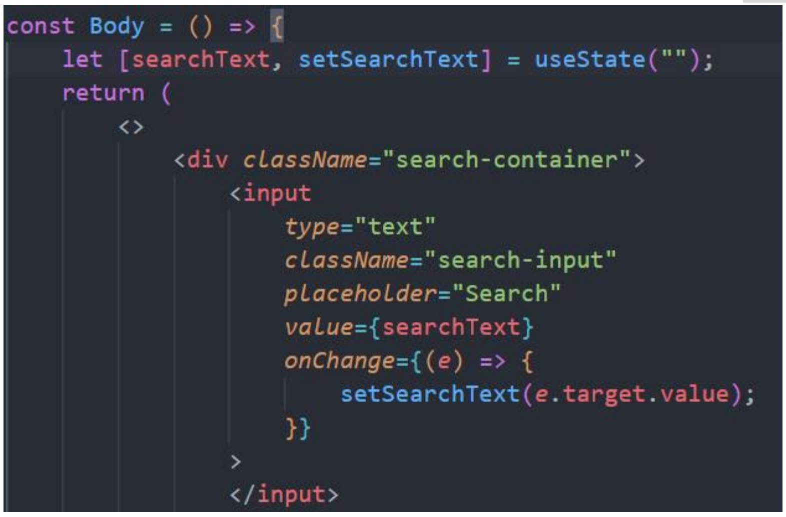


```
▸Warning: A component is changing an uncontrolled input to be controlled. This is likely caused by the value changing from undefined to a defined value,    index.js:1
which should not happen. Decide between using a controlled or uncontrolled input element for the lifetime of the component. More info: https://reactjs.org/link/control
led-components
    at input
    at div
    at Body (/__parcel_hmr/6399f84a4a000606:18:59)
    at AppLayout
```

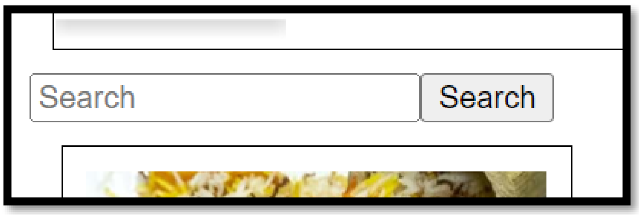
So, it’s always better to give a default value (even if it’s a null/empty value).

- Earlier, we said that useState(), returns an array with the 1st element being the local state variable. Now, we also know that the second variable is a set function which helps to modify the state variable.
- If there’s an empty/null value given as the default value of a state-variable, and there is already a placeholder attribute with another value, then the latter will take precedence, else, if the default value isn’t null, then the default value passed during declaration of the variable will be prioritized.

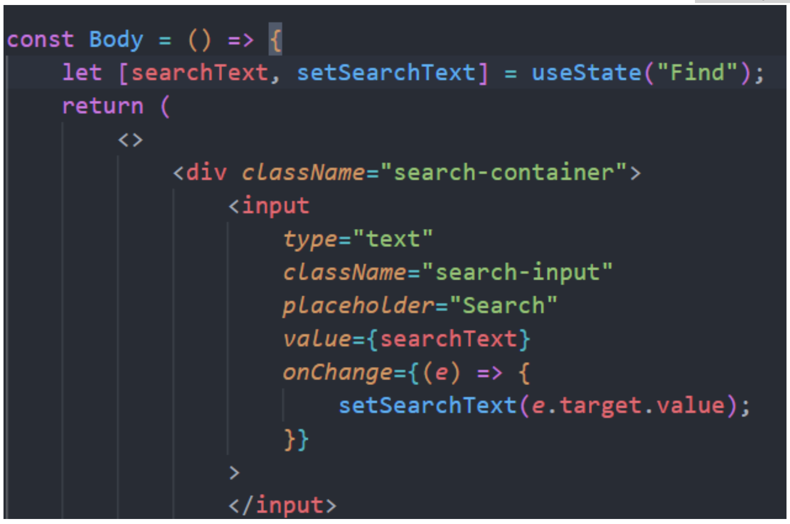
Example :- 1



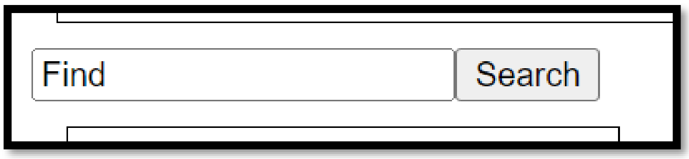
```
const Body = () => {
  let [searchText, setSearchText] = useState("");
  return (
    <>
      <div className="search-container">
        <input
          type="text"
          className="search-input"
          placeholder="Search"
          value={searchText}
          onChange={(e) => {
            setSearchText(e.target.value);
          }}
        >
        </input>
```



Example :- 2



```
const Body = () => {
  let [searchText, setSearchText] = useState("Find");
  return (
    <>
      <div className="search-container">
        <input
          type="text"
          className="search-input"
          placeholder="Search"
          value={searchText}
          onChange={(e) => {
            setSearchText(e.target.value);
          }}
        >
        </input>
```



- So, use State() helps us to do Two-Way Binding because we can now read the state variable and write it too. (See the example in video from 1:28:24 to 1:29:30

#NOTE :-

Why do we need to use useState in React?

Because React does One-way Data Binding. Read from here and here. To read more about useState, go here.

Also, when we change a local-variable in React through our UI, then React has no idea that the variable got changed and hence it will not re-render the required components. To overcome this, we use useState. (also see video from 1:31:10 to 1:35:40).

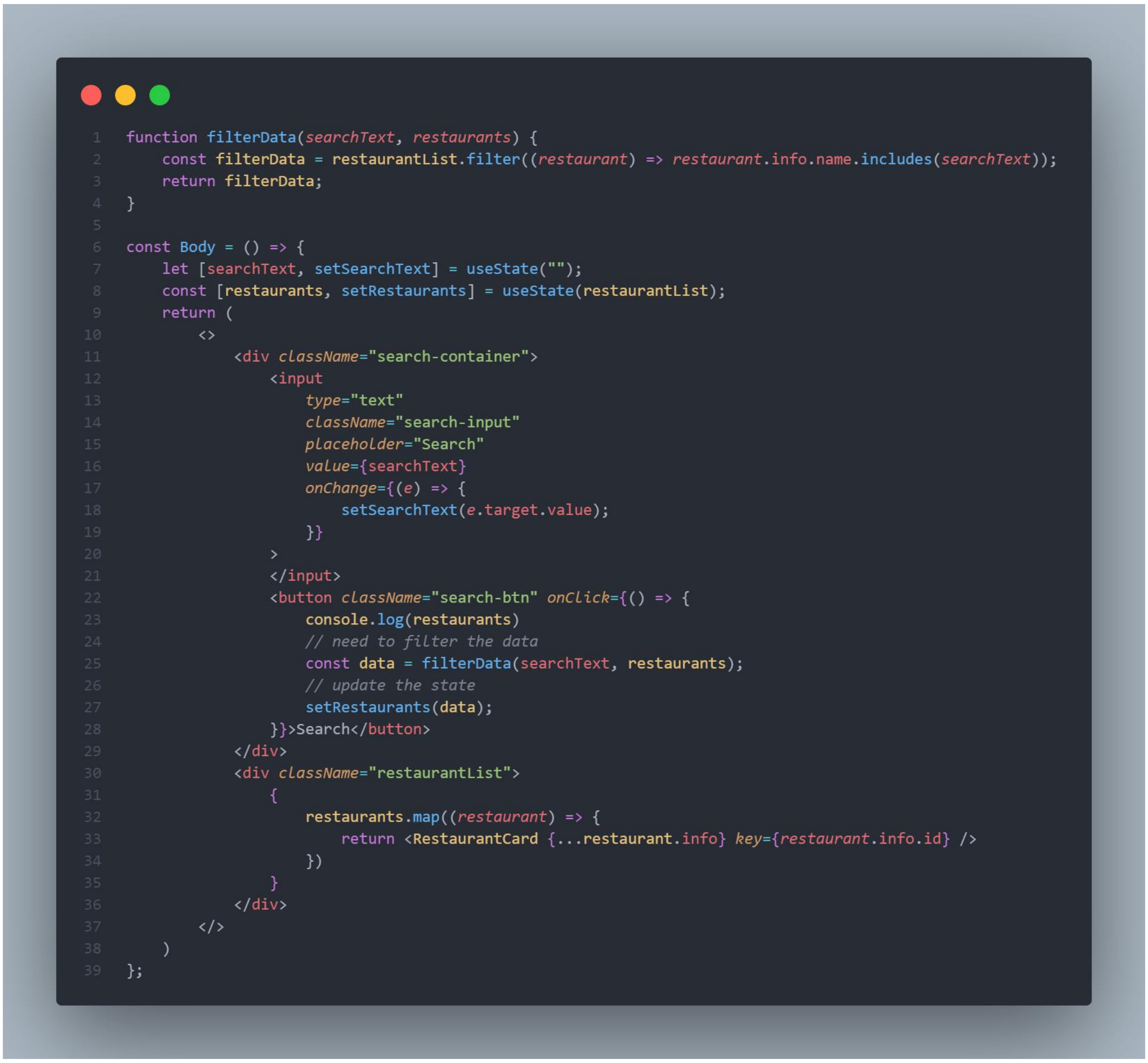
When the value/state of that state-variable is changed, React used Reconciliation to rerender the changed components.

- See the video from 1:35:56 to 1:46:30 to see a good example of how React uses Reconciliation while using useState().

Filtering the Restaurant Data :-

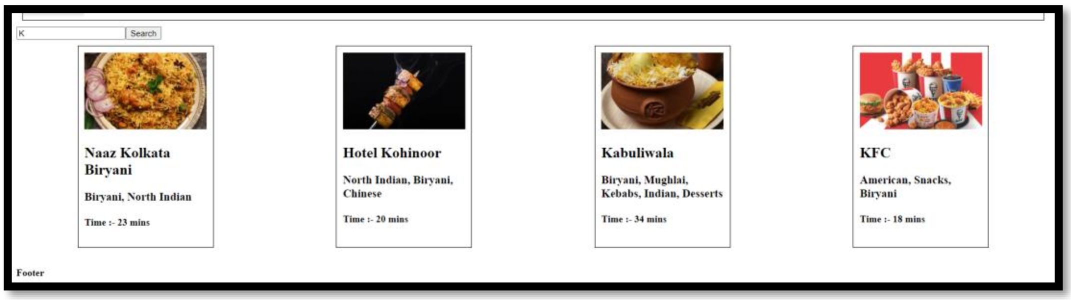
We know that when we click the “Search” button, an action should happen. So, we need the “onClick” attribute. Also, when we filter the data, we should only show only the filtered restaurants in our UI. So, the value of the entire restaurantList should be stored a state-variable as well with the initial restaurantList as its default value.
So, when we click that Search button, we should filter the data and update the state variable too.
Also, in the UI, the results shown will not be of all the objects in the restaurantList, but only the filteredData should be shown. So, the component where we were iterating over the restaurantList as a whole and displating its contents, there we should change the restaurantList to the state-variable restaurants (which has an initial value “restaurantList”).

Now, you will see that after these changes, the code looks something like this :-



```
function filterData(searchText, restaurants) {
    const filterData = restaurantList.filter((restaurant) => restaurant.info.name.includes(searchText));
    return filterData;
}

const Body = () => {
    let [searchText, setSearchText] = useState("");
    const [restaurants, setRestaurants] = useState(restaurantList);
    return (
        <>
            <div className="search-container">
                <input
                    type="text"
                    className="search-input"
                    placeholder="Search"
                    value={searchText}
                    onChange={(e) => {
                        setSearchText(e.target.value);
                    }}
                >
                </input>
                <button className="search-btn" onClick={() => {
                    console.log(restaurants)
                    // need to filter the data
                    const data = filterData(searchText, restaurants);
                    // update the state
                    setRestaurants(data);
                }}>Search</button>
            </div>
            <div className="restaurantList">
                {
                    restaurants.map((restaurant) => {
                        return <RestaurantCard {...restaurant.info} key={restaurant.info.id} />
                    })
                }
            </div>
        </>
    )
};
```

And you will see that the search bar is working fine and we are able to search any restaurant.



K Search

Naaz Kolkata Biryani
Biryani, North Indian
Time :- 23 mins

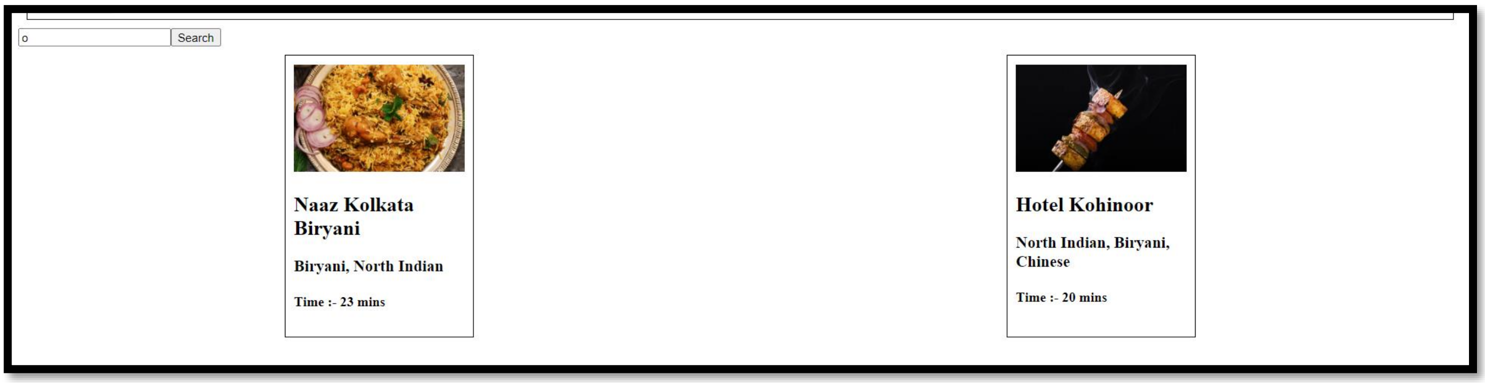
Hotel Kohinoor
North Indian, Biryani, Chinese
Time :- 20 mins

Kabuliwala
Biryani, Mughlai, Kebabs, Indian, Desserts
Time :- 34 mins

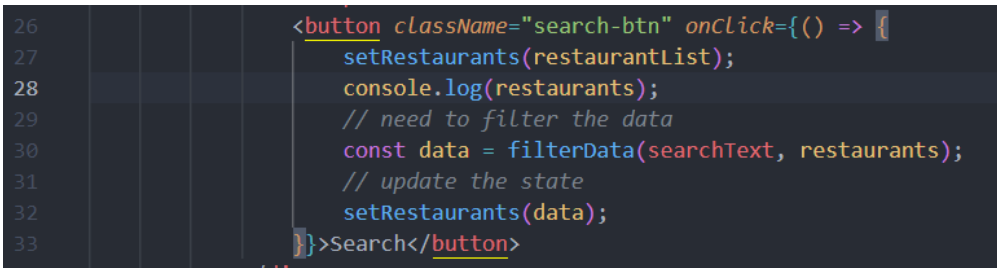
KFC
American, Snacks, Biryani
Time :- 18 mins

Footer

So, when we are searching for restaurants with "K", they are showing desirable results. But, when to try to search for restaurants containing "o" in their name, it will show only few restaurants and not all.



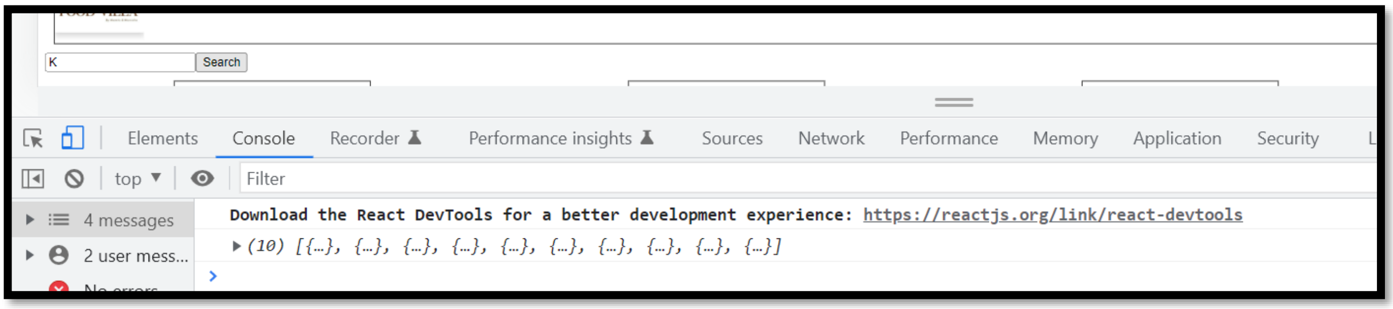
This happens because, once we are clicking the search button, the value of the state variable changes to the filtered Data and upon further use of the search button, the filter is only performed on the modified data and not the original restaurantList. So, we do something like this :-



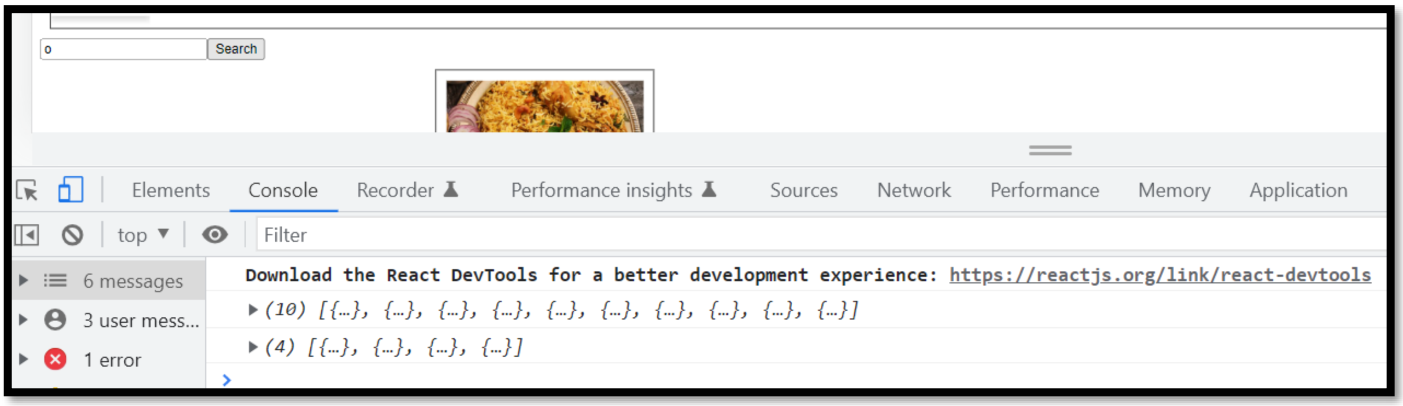
```
26  <button className="search-btn" onClick={() => {
27      setRestaurants(restaurantList);
28      console.log(restaurants);
29      // need to filter the data
30      const data = filterData(searchText, restaurants);
31      // update the state
32      setRestaurants(data);
33  }}>Search</button>
```

(Even, though line 28 isn't needed, I did it to show an error). Now, this should show desirable results because every time I click on Search button, the restaurants state variable should change it's value to the original restaurantList (it's default value) and then it should be filtered. But it will give the same undesirable results.

See, what is the output in console when we first search "K" and then "o" :-



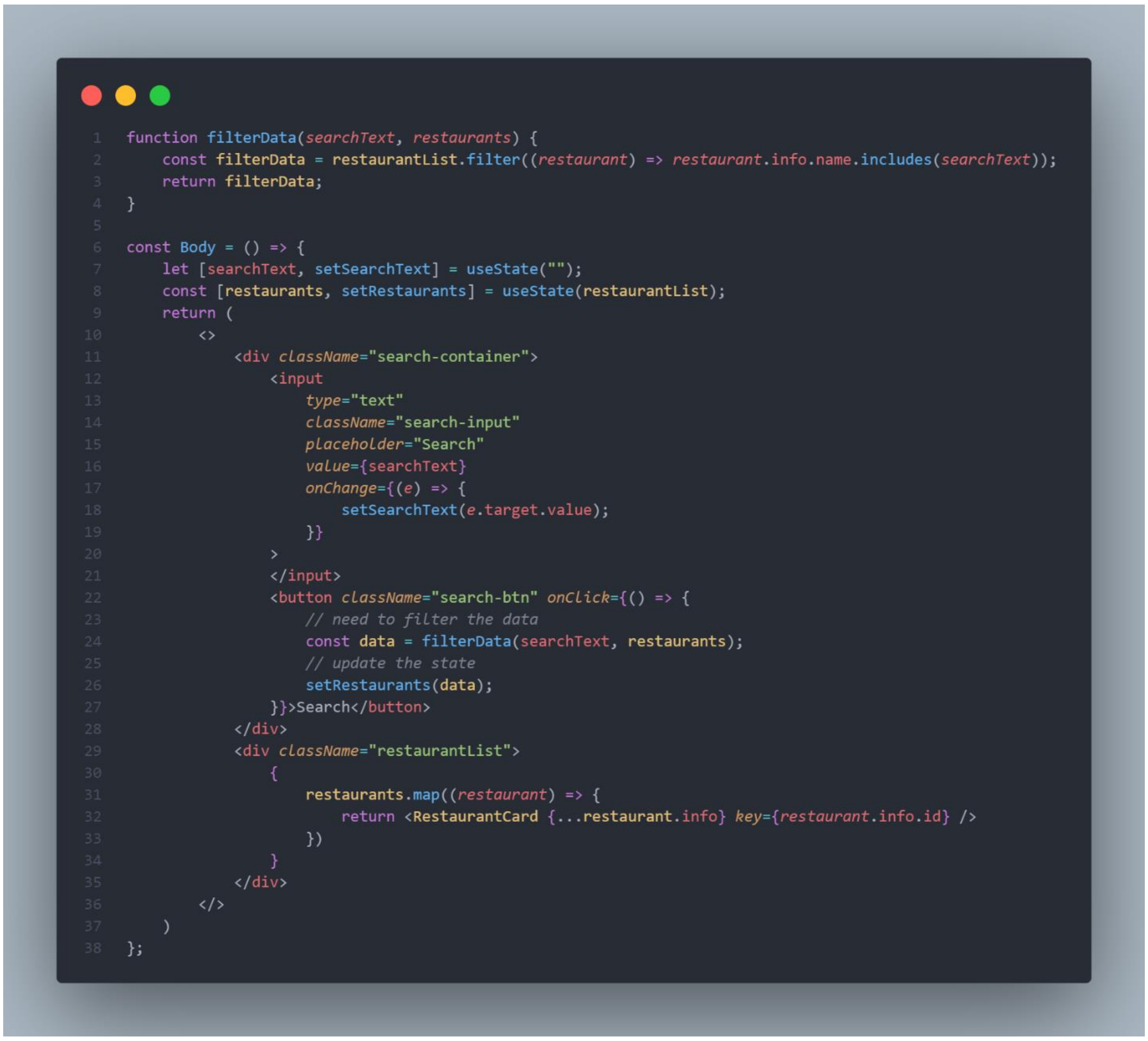
```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
▸(10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
```



```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
▸(10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
▸(4) [{…}, {…}, {…}, {…}]
```

The 1st output in console is fine because all the 10 initial items in the restaurantList is getting printed, but in the 2nd image, only the 4 items that was present after filtering with "K" are getting printed. This is because of the asynchronous nature of the useState() function. So, even before the value of restaurants gets updated to restaurantList in line 27 using the setRestaurants() function, restaurants gets passed to the filterData() function and we get undesirable results again.

To overcome this, instead of calling the setRestaurants() function in the onClick callback function, we can just perform filter on the original restaurantList data, instead of the restaurants parameter that we pass as an argument filterData() function, like :-



```jsx
function filterData(searchText, restaurants) {
    const filterData = restaurantList.filter((restaurant) => restaurant.info.name.includes(searchText));
    return filterData;
}

const Body = () => {
    let [searchText, setSearchText] = useState("");
    const [restaurants, setRestaurants] = useState(restaurantList);
    return (
        <>
            <div className="search-container">
                <input
                    type="text"
                    className="search-input"
                    placeholder="Search"
                    value={searchText}
                    onChange={(e) => {
                        setSearchText(e.target.value);
                    }}
                >
                </input>
                <button className="search-btn" onClick={() => {
                    // need to filter the data
                    const data = filterData(searchText, restaurants);
                    // update the state
                    setRestaurants(data);
                }}>Search</button>
            </div>
            <div className="restaurantList">
                {
                    restaurants.map((restaurant) => {
                        return <RestaurantCard {...restaurant.info} key={restaurant.info.id} />
                    })
                }
            </div>
        </>
    )
};
```

And now we will get desirable results.