Chapter – 7 - Routing

React Guidelines

* Never create a component inside a component. Why? Because If we create a component inside a component, every time parent component renders or rerenders, the child component will be created which is not memory efficient.

* Never create useState variables inside if else block. Why? Because If we define UseState variables inside if-else block and if the block does not get executed React will not be able to know whether the state variable really exist or not. React will not be able to trace that variable and React does not like inconsistency. React should know what your component is doing.

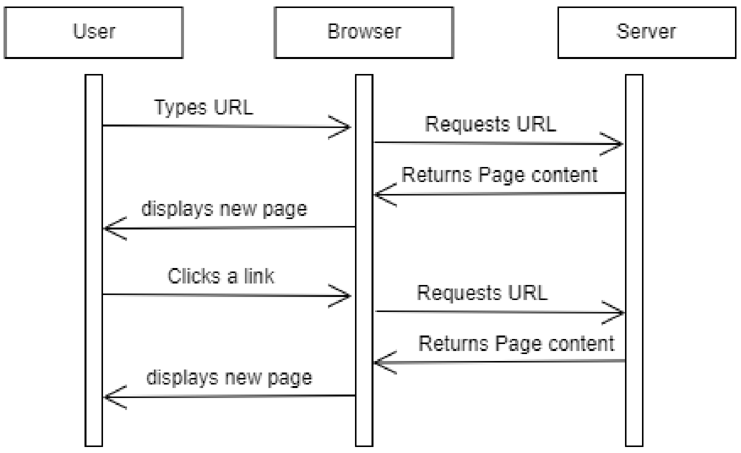
* Never create useState variables inside a for loop. Why? Because If we do, it will create state variables as many times as the number of iteration.one state variable was needed, many were created.

* Never create useState variables outside of a function component. Why? Because if we do, state variables will not be a part of that component.

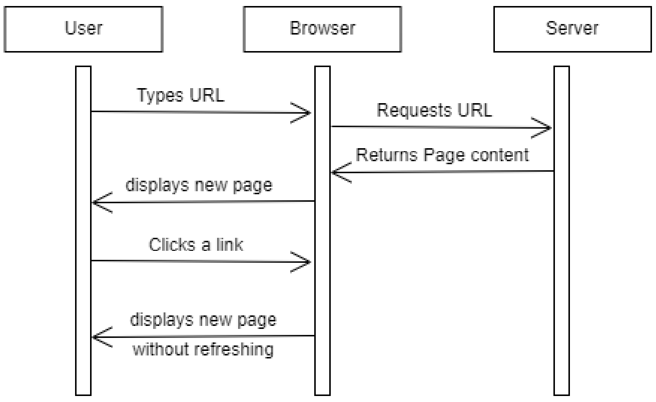
Routing in general

There are two types of routing. Server-side routing & Client-side Routing.

Server-side routing is the traditional approach to handling routes in web applications. The process involves requesting a new page from the server and providing it to the user every time a link is clicked. One major issue with server-side routing is that every request result in a full-page refresh, which is not performance efficient in most cases.



Client-side routing involves JavaScript which handles the routing process internally. In client-side routing, a request to the server is prevented when a user clicks a link, hence no page refresh even when the URL changes. Because all of the available data and components are already present in the local machine.



What is a SPA?

SPA stands for Single Page Application. In older days, when we used to navigate from pages to pages network calls were made, data were loaded from the server and then displayed in UI. Basically, we were reloading the entire page on every network call. But with SPA, our application does not reload for every action that user takes. SPA does not make network calls, when we navigate from pages to pages or some portion of the pages.

Can I have more than one use effect?

We can create as many use effects as per the need.

Why CDN is a great place to host Images?

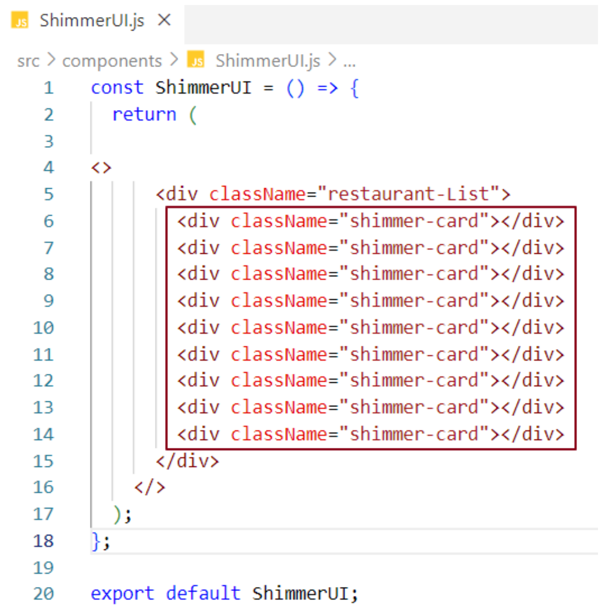
* CDN is fast and it returns images very fast.

* CDN has 100% Up time.

* CDN optimises images before sending them to browser. CDN caches images.

Big applications like swiggy Zomato uses CDN to host images because of these advantages.

In precious chapter we had shown a text in shimmer component. Let's build actual shimmer UI.



```
const ShimmerUI = () => {
  return (

<>
      <div className="restaurant-List">
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
        <div className="shimmer-card"></div>
      </div>
    </>
  );
};

export default ShimmerUI;
```

Again, we are hardcoding shimmer card components in UI. We need to use a loop to ease our task .

When you are building an app be conscious about what packages you are using. In an application development, use packages if it is actually required. For example – Formik is a package using which we can build react forms. Packages such as isOdd, isEven are not really required when you can write your own code for them.

How can I achieve routing in react?

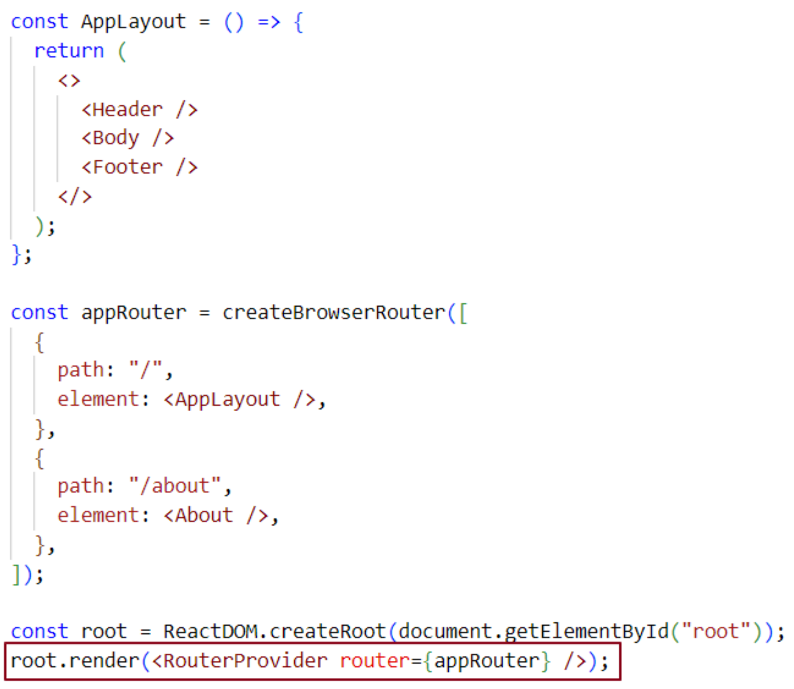
The library react-router-dom is used to achieve routing in react.

* react-router-dom package provides a function createBrowserRouter which helps us creating a route.

* createBrowserRouter function takes an array of router configuration objects of which each object has their own configuration set. We provide path and element to each configuration object so that when we navigate to the path, component or element associated with that path will be rendered in UI. createBrowserRouter creates a router for us.

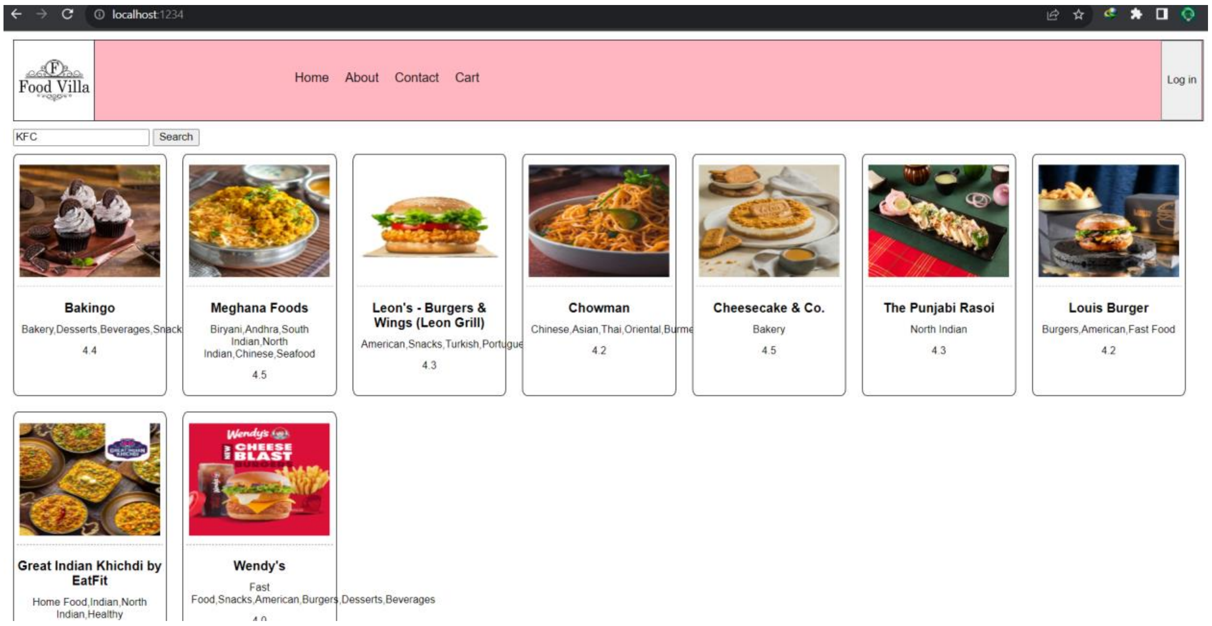
* Creating a router will not be enough, we need to provide this router to our application so that our app can use it for routing. For this to happen RouterProvider is the component that we import from react-router-dom.

* Now, instead of rendering a component inside render function. we provide RouterProvider component with a prop routing configuration array as an argument to render function which will render component as per the route configuration.

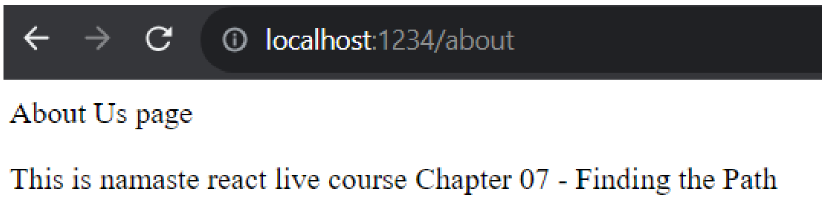


```
const AppLayout = () => {
  return (
    <>
      <Header />
      <Body />
      <Footer />
    </>
  );
};

const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
  },
  {
    path: "/about",
    element: <About />,
  },
]);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<RouterProvider router={appRouter} />);
```

Now if we navigate to the base URL http://localhost:1234/ we get AppLayout component page


localhost:1234
Food Villa
Home
About
Contact
Cart
Log in
KFC
Search
Bakingo
Bakery,Desserts,Beverages,Snack
4.4
Meghana Foods
Biryani,Andhra,South Indian,North Indian,Chinese,Seafood
4.5
Leon's - Burgers & Wings (Leon Grill)
American,Snacks,Turkish,Portugue
4.3
Chowman
Chinese,Asian,Thai,Oriental,Burme
4.2
Cheesecake & Co.
Bakery
4.5
The Punjabi Rasoi
North Indian
4.3
Louis Burger
Burgers,American,Fast Food
4.2
Great Indian Khichdi by EatFit
Home Food,Indian,North Indian,Healthy
Wendy's
Fast Food,Snacks,American,Burgers,Desserts,Beverages

If we navigate to about page URL http://localhost:1234/about we get about component page


localhost:1234/about

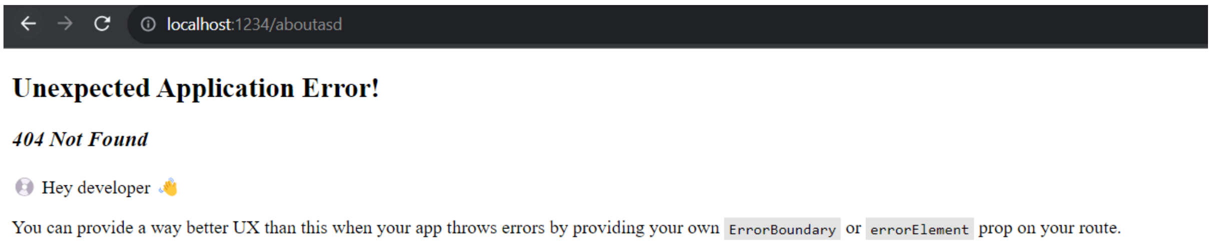
About Us page

This is namaste react live course Chapter 07 - Finding the Path

This way we can achieve routing in react.

What happens when user navigates to a URL that does not exist?

react-router-dom is a powerful Library that also handles the error in case we navigate to a route that does not even exist. we get a default 404 error page.


localhost:1234/aboutasd

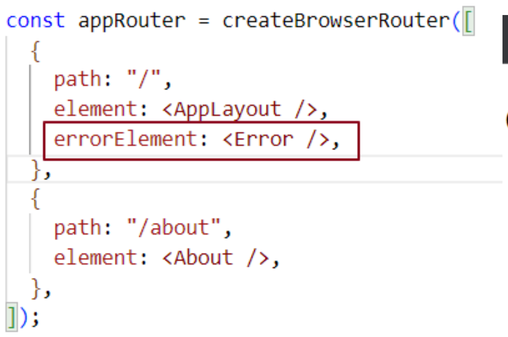
Unexpected Application Error!

404 Not Found

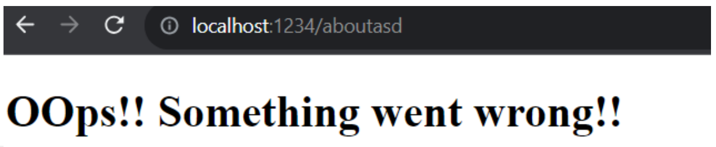
💿 Hey developer 👋

You can provide a way better UX than this when your app throws errors by providing your own `ErrorBoundary` or `errorElement` prop on your route.

However, we can show our custom error page and to show our custom error page on UI, we add one more attribute errorElement to route configuration object for which the path provided did not match with the URL provided in UI. errorElement calls the error component when we provide invalid URL on the UI.



```
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,
  },
  {
    path: "/about",
    element: <About />,
  },
]);
```

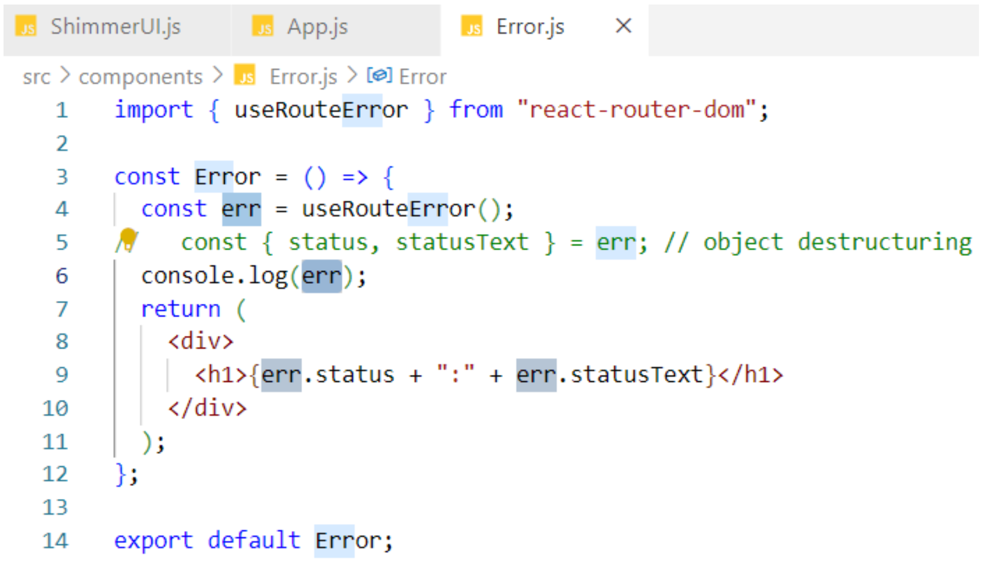


localhost:1234/aboutasd

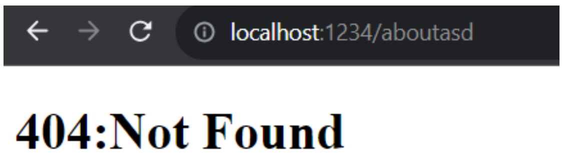
OOps!! Something went wrong!!

How do we show more information about the error in case user redirects to an Invalid URL?

react-router-dom provides a hook useRouteError which returns an error object which states what types of error did we encounter.



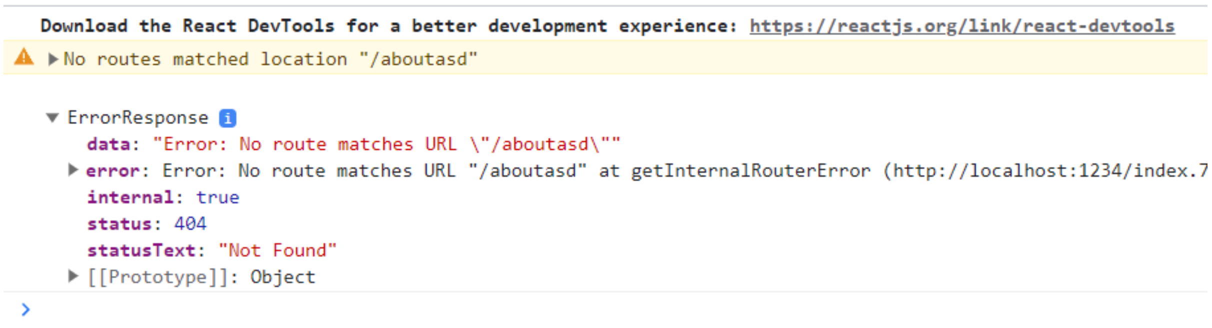
```
import { useRouteError } from "react-router-dom";

const Error = () => {
  const err = useRouteError();
    const { status, statusText } = err; // object destructuring
  console.log(err);
  return (
    <div>
      <h1>{err.status + ":" + err.statusText}</h1>
    </div>
  );
};

export default Error;
```



localhost:1234/aboutasd

404:Not Found

On the console:



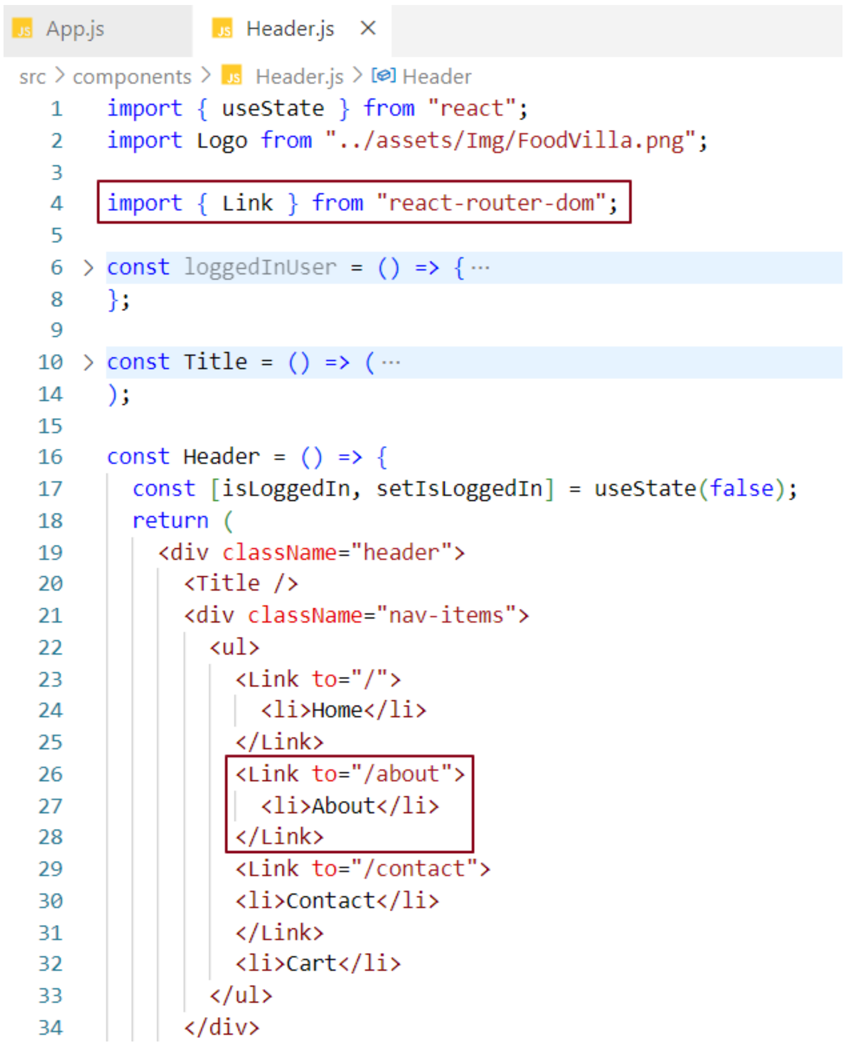
```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
⚠ ▶No routes matched location "/aboutasd"

▼ErrorResponse ℹ
    data: "Error: No route matches URL \"/aboutasd\""
  ▶error: Error: No route matches URL "/aboutasd" at getInternalRouterError (http://localhost:1234/index.7
    internal: true
    status: 404
    statusText: "Not Found"
  ▶[[Prototype]]: Object
>
```

What is a Link tag? Why do we use Link tag?

Anchor tag leads to Server-Side routing because in anchor tag we have an attribute href where we provide the URL and when we click on the link, a call is made to the URL hosted in a server and our page reloads until we get the data from the server. This is not a good user experience

* To achieve client-side routing, we use Links in React. We should enclose navigation elements inside a Link tag given by react-router-dom.

* In Link tab we define an attribute named **to** where we set a path for the link and **to** decides where to navigate when the link is clicked.



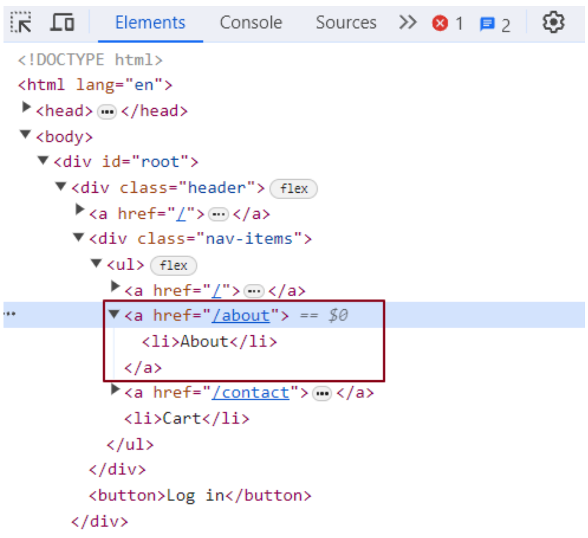
App.js | Header.js

src > components > Header.js > Header

```
import { useState } from "react";
import Logo from "../assets/Img/FoodVilla.png";

import { Link } from "react-router-dom";

const loggedInUser = () => {⋯
};

const Title = () => (⋯
);

const Header = () => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  return (
    <div className="header">
      <Title />
      <div className="nav-items">
        <ul>
          <Link to="/">
            <li>Home</li>
          </Link>
          <Link to="/about">
            <li>About</li>
          </Link>
          <Link to="/contact">
          <li>Contact</li>
          </Link>
          <li>Cart</li>
        </ul>
      </div>
```

Now if we click on About, it will lead me to about component page.

- At this moment If we inspect the page, we see anchor tag instead of Link tag. why? because the Link tag uses anchor tag behind the scene and this is done by react-router-dom library. To make the Link tag understandable by browser react-router-dom keeps a track of all the Link tags and convert them to anchor tag which browser understands.

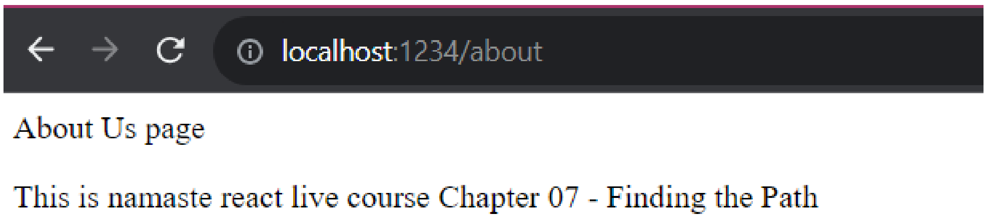


```
<!DOCTYPE html>
<html lang="en">
▶<head> ··· </head>
▼<body>
  ▼<div id="root">
    ▼<div class="header"> flex
      ▶<a href="/"> ··· </a>
      ▼<div class="nav-items">
        ▼<ul> flex
          ▶<a href="/"> ··· </a>
          ▼<a href="/about"> == $0
             <li>About</li>
           </a>
          ▶<a href="/contact"> ··· </a>
           <li>Cart</li>
         </ul>
       </div>
       <button>Log in</button>
     </div>
```

Nested Routing? Why Nested Routing? How do we achieve this in React?

Nested routing enables us placing multiple children components within the parent component and performs routing on child components without reloading the entire page.

Right now, if we notice about page component, we can't see the header and footer component loaded. But we want them to be loaded for about component and for other page components.


localhost:1234/about

About Us page

This is namaste react live course Chapter 07 - Finding the Path

In below screen shot, if my path is "/", I want my "Body" component to be inside header and footer component. If my path is "/about", I want my "About" component to be inside header and footer component.

```
const AppLayout = () => {
  return (
    <>
      <Header />
      <Body />  // If path is "/"
      <About /> // If path is "/about"
      <Footer />
    </>
  );
};
```

We want our header and footer component fixed and display different components inside header and footer on page navigation.

To make it happen we will have to make About and Body component, children of AppLayout. component. Also, we provide an Outlet in our Layout where we want our component to be loaded while navigated.

```
const AppLayout = () => {
  return (
    <>
      <Header />
      <Outlet />
      <Footer />
    </>
  );
};

const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,
    children: [
      {
        path: "/about",
        element: <About />,
      },
      {
        path: "/",
        element: <Body />,
      },
    ],
  },
]);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<RouterProvider router={appRouter} />);
```

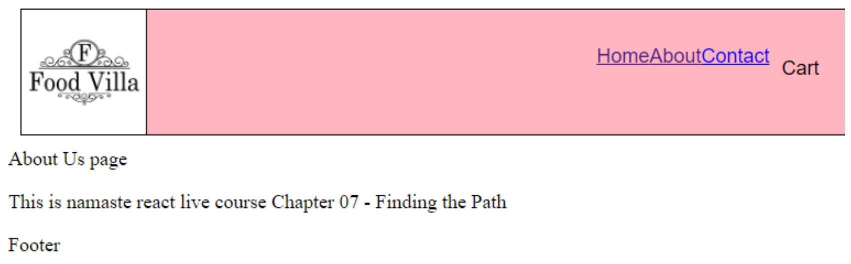
What is Outlet?

Outlet is a component that react-router-dom provides to plug in children components into layout component during page navigation.

Outlet is a place holder for children components configured in route configuration objects.

When we navigate from page to page react triggers reconciliation which updates only the updated portion in the dom which is the outlet.

About component page navigation -

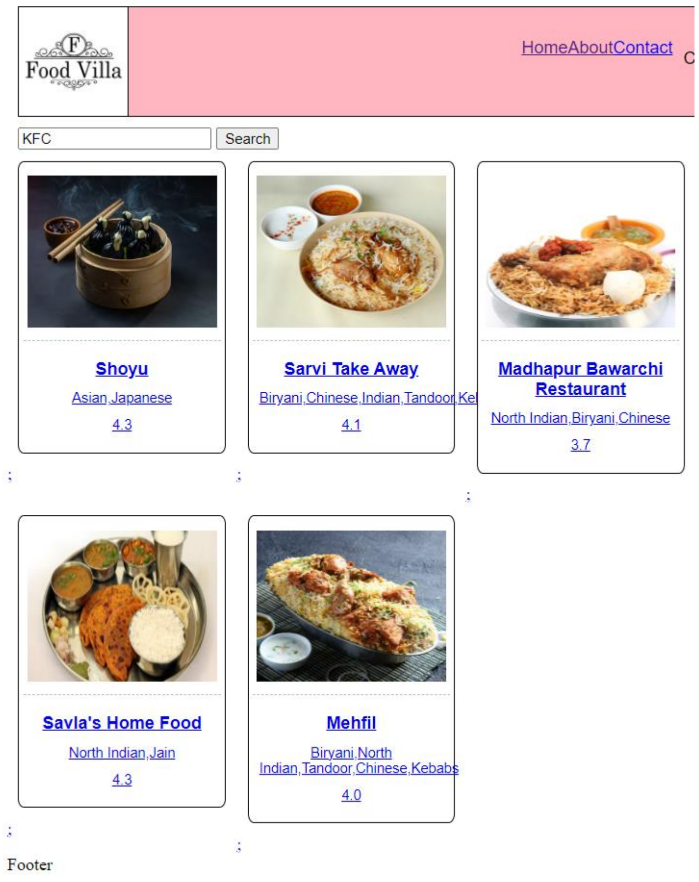


About Us page

This is namaste react live course Chapter 07 - Finding the Path

Footer

Body component page navigation -


Food Villa

HomeAboutContact C

KFC Search

Shoyu
Asian,Japanese
4.3

Sarvi Take Away
Biryani,Chinese,Indian,Tandoor,Ke
4.1

Madhapur Bawarchi Restaurant
North Indian,Biryani,Chinese
3.7

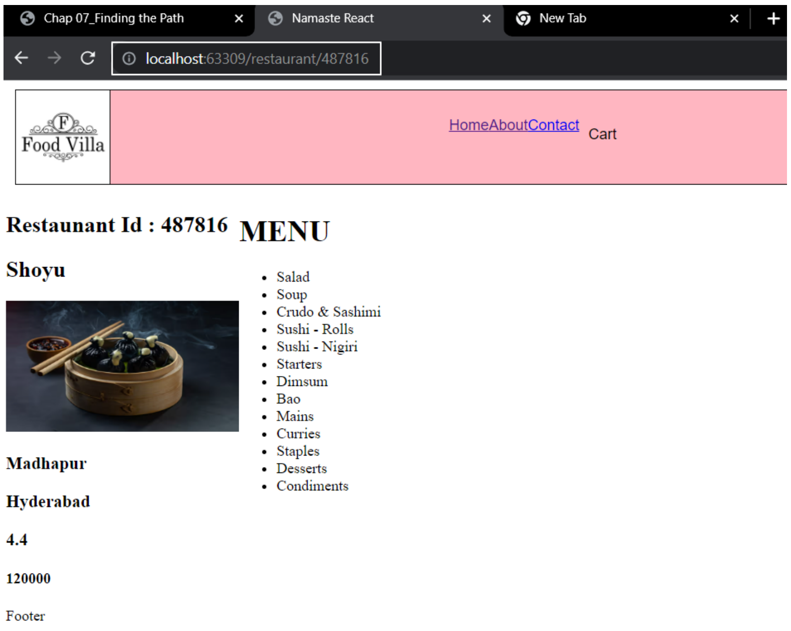
Savla's Home Food
North Indian,Jain
4.3

Mehfil
Biryani,North Indian,Tandoor,Chinese,Kebabs
4.0

Footer

Dynamic routing –

Dynamic routing is the time at which routing take place. In Dynamic routing, routes are initialised dynamically. In our case restaurant card is clicked and route for that card generated.


Chap 07_Finding the Path | Namaste React | New Tab

localhost:63309/restaurant/487816

Food Villa

HomeAboutContact Cart

Restaunant Id : 487816

Shoyu

Madhapur

Hyderabad

4.4

120000

MENU

- Salad
- Soup
- Crudo & Sashimi
- Sushi - Rolls
- Sushi - Nigiri
- Starters
- Dimsum
- Bao
- Mains
- Curries
- Staples
- Desserts
- Condiments

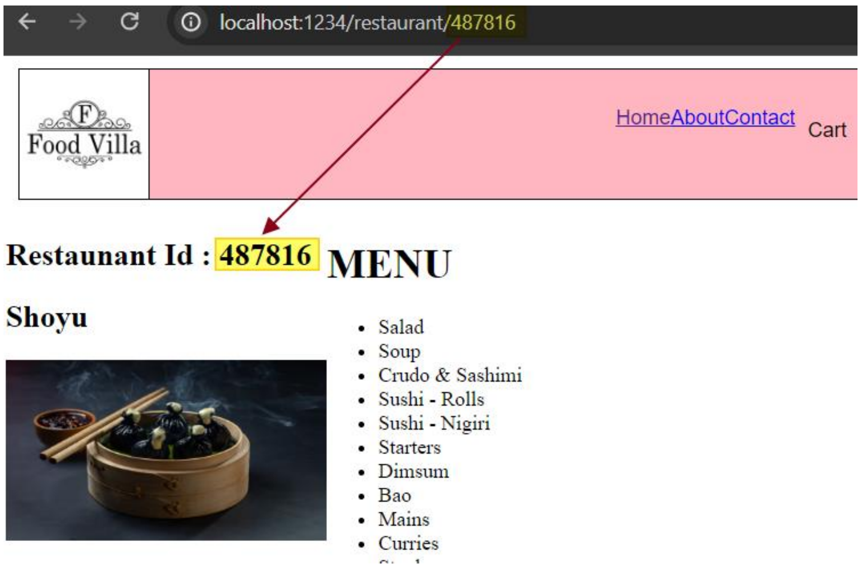
Footer

Behind the scene, when we click on a restaurant card from restaurant list, Id of that restaurant will be attached to the URL and this will be read by restaurant Menu component. Restaurant Menu component uses this ID for making API calls and displaying the required data in UI.

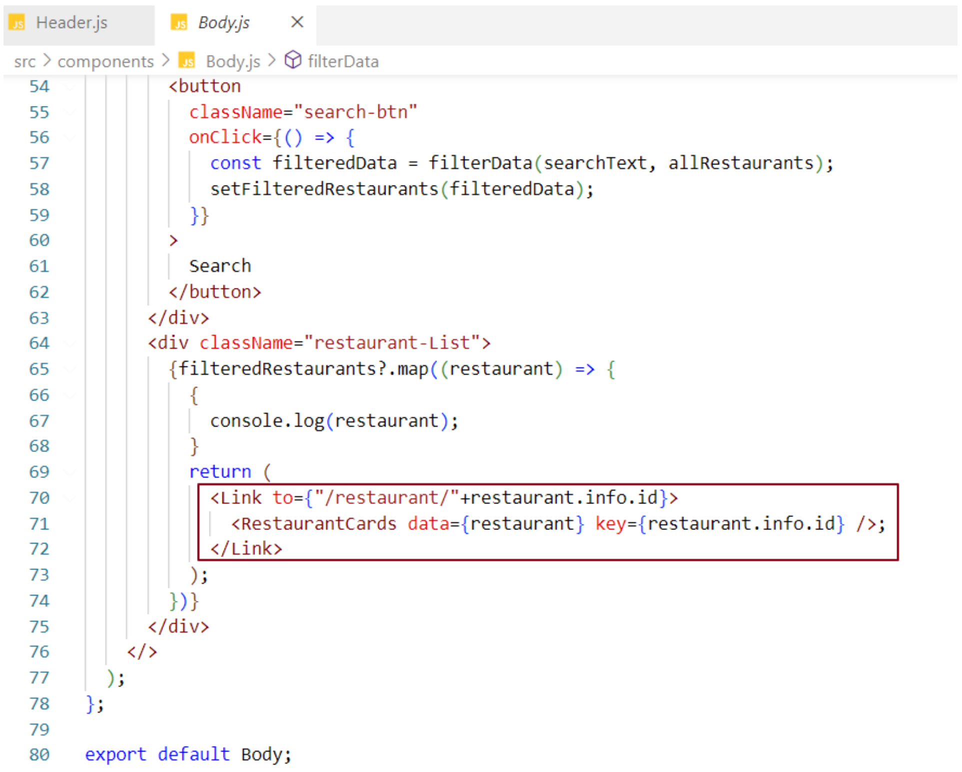
What is useParam hook in react?

useParam is one of the hooks given by react-router-dom library. We can use this hook to retrieve route parameters from a route URL as use it in the component we navigated to.

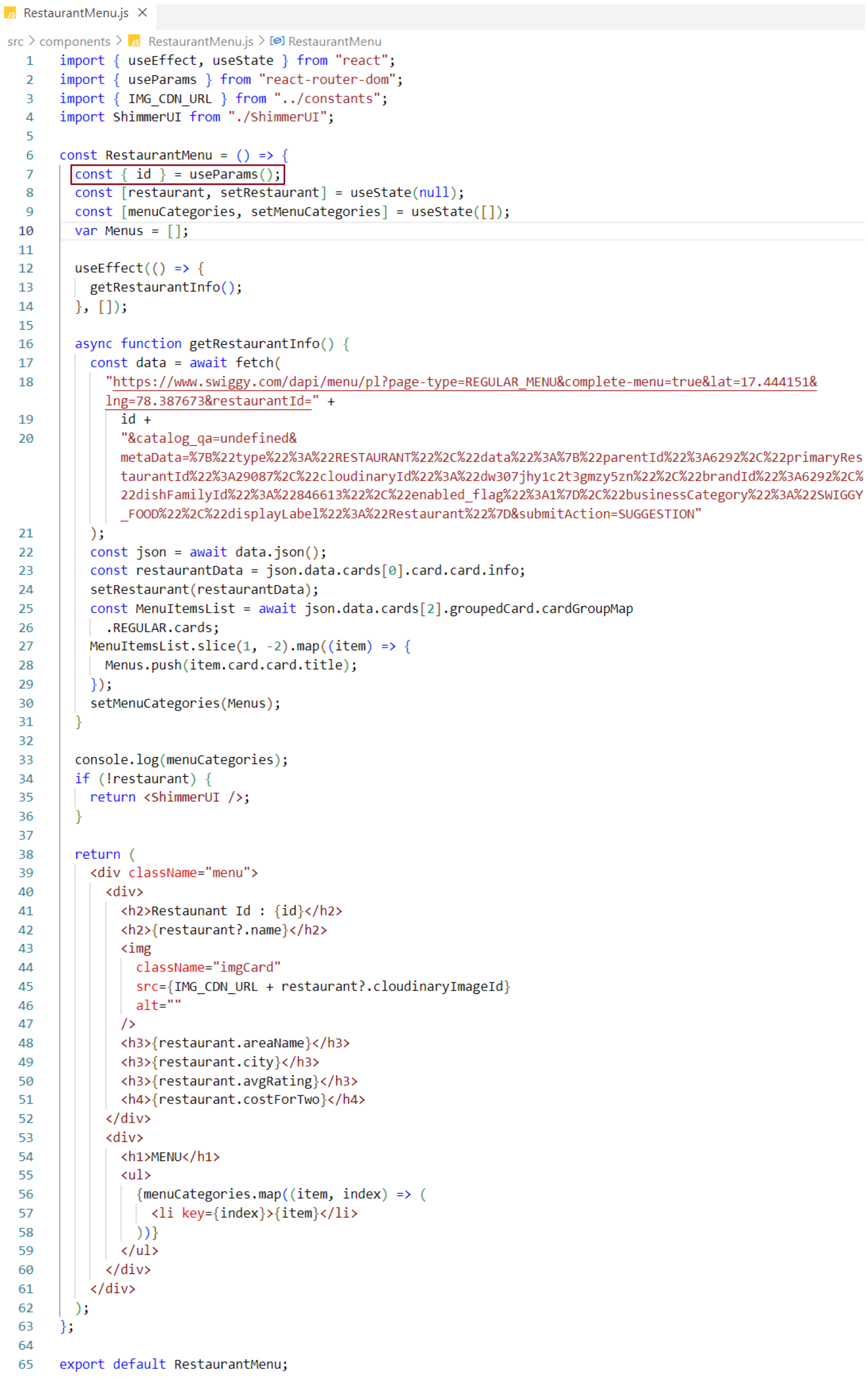
useParam enables us to read data from the route URL.



Code implementation for dynamic routing -

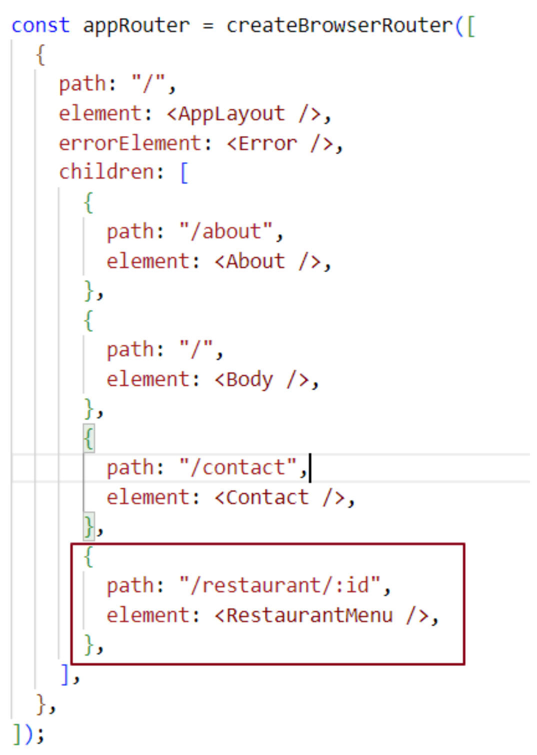


```
          <button
            className="search-btn"
            onClick={() => {
              const filteredData = filterData(searchText, allRestaurants);
              setFilteredRestaurants(filteredData);
            }}
          >
            Search
          </button>
        </div>
        <div className="restaurant-List">
          {filteredRestaurants?.map((restaurant) => {
            {
              console.log(restaurant);
            }
            return (
              <Link to={"/restaurant/"+restaurant.info.id}>
                <RestaurantCards data={restaurant} key={restaurant.info.id} />;
              </Link>
            );
          })}
        </div>
      </>
    );
  };

  export default Body;
```


RestaurantMenu.js ×

src > components > RestaurantMenu.js > RestaurantMenu

```
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { IMG_CDN_URL } from "../constants";
import ShimmerUI from "./ShimmerUI";

const RestaurantMenu = () => {
  const { id } = useParams();
  const [restaurant, setRestaurant] = useState(null);
  const [menuCategories, setMenuCategories] = useState([]);
  var Menus = [];

  useEffect(() => {
    getRestaurantInfo();
  }, []);

  async function getRestaurantInfo() {
    const data = await fetch(
      "https://www.swiggy.com/dapi/menu/pl?page-type=REGULAR_MENU&complete-menu=true&lat=17.444151&lng=78.387673&restaurantId=" +
        id +
        "&catalog_qa=undefined&metaData=%7B%22type%22%3A%22RESTAURANT%22%2C%22data%22%3A%7B%22parentId%22%3A6292%2C%22primaryRestaurantId%22%3A29087%2C%22cloudinaryId%22%3A%22dw307jhy1c2t3gmzy5zn%22%2C%22brandId%22%3A6292%2C%22dishFamilyId%22%3A%22846613%22%2C%22enabled_flag%22%3A1%7D%2C%22businessCategory%22%3A%22SWIGGY_FOOD%22%2C%22displayLabel%22%3A%22Restaurant%22%7D&submitAction=SUGGESTION"
    );
    const json = await data.json();
    const restaurantData = json.data.cards[0].card.card.info;
    setRestaurant(restaurantData);
    const MenuItemsList = await json.data.cards[2].groupedCard.cardGroupMap
      .REGULAR.cards;
    MenuItemsList.slice(1, -2).map((item) => {
      Menus.push(item.card.card.title);
    });
    setMenuCategories(Menus);
  }

  console.log(menuCategories);
  if (!restaurant) {
    return <ShimmerUI />;
  }

  return (
    <div className="menu">
      <div>
        <h2>Restaunant Id : {id}</h2>
        <h2>{restaurant?.name}</h2>
        <img
          className="imgCard"
          src={IMG_CDN_URL + restaurant?.cloudinaryImageId}
          alt=""
        />
        <h3>{restaurant.areaName}</h3>
        <h3>{restaurant.city}</h3>
        <h3>{restaurant.avgRating}</h3>
        <h4>{restaurant.costForTwo}</h4>
      </div>
      <div>
        <h1>MENU</h1>
        <ul>
          {menuCategories.map((item, index) => (
            <li key={index}>{item}</li>
          ))}
        </ul>
      </div>
    </div>
  );
};

export default RestaurantMenu;
```



```
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,
    children: [
      {
        path: "/about",
        element: <About />,
      },
      {
        path: "/",
        element: <Body />,
      },
      {
        path: "/contact",
        element: <Contact />,
      },
      {
        path: "/restaurant/:id",
        element: <RestaurantMenu />,
      },
    ],
  },
]);
```

What would happen if we do `console.log(useState())`?

If we do console.log(useState()), we get an array [undefined, function] where first item in an array is state is undefined and the second item in an array is setState function is bound dispatchSetState.

What is the difference between Client-Side Routing and Server-Side Routing?

In Server-side routing or rendering (SSR), every change in URL, http request is made to server to fetch the webpage, and replace the current webpage with the older one.

In Client-side routing or rendering (CSR), during the first load, the webapp is loaded from server to client, after which whenever there is a change in URL, the router library navigates the user to the new page without sending any request to backend. All Single Page Applications uses client-side routing.