Date.......

Chapter-07 — Finding the path

Note

Never create a component inside a component. you can make that component above it and can do component composition in the present component. Because everytime your component render it will create a more component. If you write a component inside a component.

→ also don't use (never use) useState hook inside if or for loop/statement as react doesn't like inconsistency. (It is not preffered!!)

→ useState is a hook that react gives you to create local state variable inside your functional component., so never use useState outside your functional component.

→ you can create several useState according to the need.

Q. What are various ways to add images into our App? Explain with code.

Sol. There are various ways by which we can import the images in our app.

1) using URL

we can simply import an image using a URL e.g.

```
<img src="url-of-image" />
```

2) using default import

src k andar assets bna ke uske andar img waha rakhiyo images saare

If we have an image on our system, we can import using default import e.g

```
import image from "/path/of/image"
<img src={image} />
```

Spiral . . .

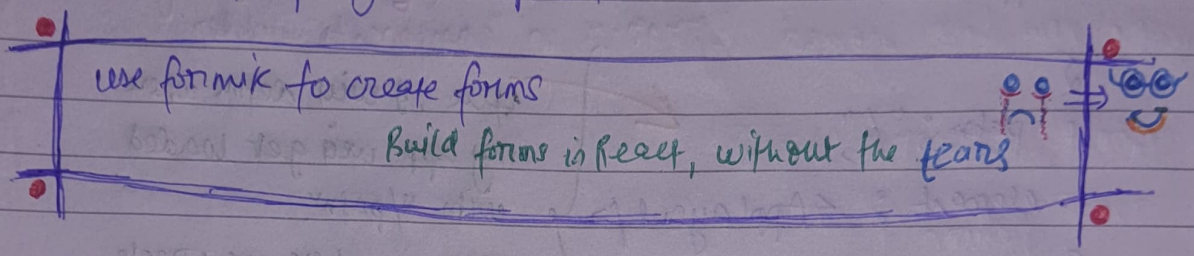
Teacher's Sign ....

→ we put optimised image on cdn (even swiggy use cdn)
↳ CDN is faster. It caches the image. It return very fast and have 100% uptime. It optimises (img are already optimised when we put.

Date .....

→ When you are building an app always be conscious about what packages you are using you don't have to import packages for every small thing.

So
when we should import a package?
→ when things got complicated..



Now we will move on to our aim of this chapter (finding the path) so we will use React routing through a npm package →

React Router

Installation

npm i react-router-dom (we will use latest React-router 6th version)

→ Now we will add a new component About.js in our component folder (as we want to make about us key work for us, ie when we click that button our react app will take us to that page).

So in our components folder

```
//make a new file About.js

const About = () {
    return ( <div>
                <h1> About us page </h1>
                <p> This is the Namaste React live course </p>
            </div>
    );
};
export default About;
```


Teacher's Sign .....

Date.....

// Now in our App.js

this is a function that we get from react-router-dom it will help us create routing

```
import { createBrowserRouter } from "react-router-dom"

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

i will create a router by using

my createBrowserRouter will take some configuration

a array list of paths

here, I will define where my app will get loaded with "/path"

This is where we create routing configuration.

createBrowserRouter is the most recommended route for all react router web projects.

There are multiple Routers. Read Docs

Now root.render() will render whatever will be given to it so If I want to render according of this configuration so we need to provide this router to my render for that we will use

```
import { createBrowserRouter, RouterProvider } from "react-router-dom"
```

a component used to provide router to the render function

So now to render our app

we need to pass this config

```
root.render(<RouterProvider router={appRouter} />);
```

So now our App.js will look like ->

// App.js

Date.....

```
import Header from "./components/Header";
import Body from "./components/Body";
import Footer from "./components/Footer";
import About from "./components/About";
import { createBrowserRouter, RouterProvider } from "react-router-dom";

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

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

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

Now in this implementation if we go on
http://localhost:1234/
or
http://localhost:1234/about
it works great, but if we go on different routes like
http://localhost:1234/some-randomtext

Spiral

Teacher's Sign .....

Date..................

it shows some unhandled default error page
Now if I want to have my own customised error page how I'll do this?

first we have to make a component in our "components" folder

// make new file named Error.js in "components" folder.

```
const Error = () => {

    return (
        <div>
            <h1> Oops!! </h1>
            <h2> Something went wrong!! </h2>
        </div>
    );
};

export default Error;
```

// Now in our App.js we need to import it and update our appRouter.

// in App.js

```
- -- - - ---

import Error from "./components/Error";

:
:

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

This is a wildcard wise take care of all error

→ bas ek element pe kar dega for bhi sab par chalega
ie
http://localhost:1234/about/kuch bhi likh, ↳ type bhi apna custom error page aayega wala

Spiral Teacher's Sign..........

Date.....

Bill if we want to show some more ~~off~~ errors to the user, we can do it using a hook provided by "react-router-dom".

React-router-dom is very powerful it doesn't let you throw error in your console it catches all the routing errors. and we use this piece of error to show it to user properly It gives us a hook "useRouteError".

//in our Error.js

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

const Error = () => {
    const err = useRouteError();
    console.log(err) // and see the returned object (we can also destructure it)
    return (
      <div>
        <h1> Oops... </h1>
        <h2> Something went wrong!! </h2>
        <h2> {err.status + ": " + err.statusText} </h2>
      </div>
    );
};

export default Error;
```

Now, If I want that, If I click "About us" it guide us to "/About" page. So how we are going to do this.

① using anchor tags ✗.

⇒ problem with anchor tag

It will reload the entire page when it is clicked. It disrupts user experience and can result in a slower page load time.
This cause problems for single-page applications (SPAs).

Single Page Applications (SPA)

- React apps are SPAs
- Having SPAs will not reload, It will not make network call when we are changing pages.
- Loads a single HTML page and dynamically update the page in response to user interaction without reloading the entire page.
- This approach allows for faster navigation and a more seamless user experience.

Two types of Routing

1. client-side routing
2. Server-side routing

Server-side Routing (Normal web application jo hote hai)

↳ all our pages come from Server
- make a network call, get the Html, js, css and loads the whole page.

Client-side Routing (SPA)

- dynamically update content of SPA in response to change in URL
- don't do full page reload.

Date.....

So we cannot use anchor tag, then to solve this problem and to replace href we have {Link} that our "sweet big" react-router-dom gives us

Link is same to same as anchor tag, but just instead of "href" we use "to".

eg.

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

<Link to="/about">
    <li>About</li>
</Link>
```

→ To keep Header & footer stick on to every page, change the routing config. ie to make the about page children of <AppLayout/>

→
```
import {createBrowserRouter, RouterProvider, Outlet} from "react-router-dom"

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

a component given by react-router-dom

ye ek placeholder hai isski jagah koi aur component aa jaega jo ki children mein set karega.

→ React-router-dom gives access to Outlet. This Outlet will be filled by the configuration. So the children will go inside Outlet according to the route (path).

↳ kuch nai hoga to aise hi rahega invisible

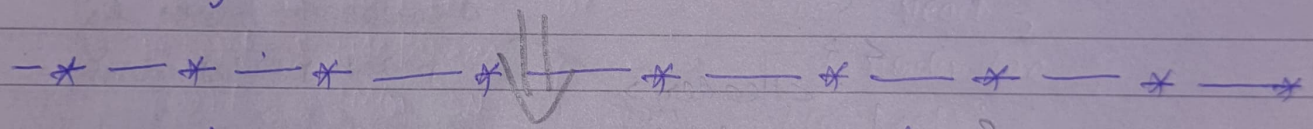
Spiral

Teacher's Sign.....

Date.....

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

initial



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

final

REMEMBER

Small c hai

Date.....

DYNAMIC ROUTING

→ Process of rendering component in response to a change in the application's URL.

→ If the route to restaurant menu page is like

```
{
    path: "/restaurant/:id",
    element: <RestaurantMenu/>
}
```

(id is dynamically (it can be anything)

→ To read the 'id' passed in URL,

'react-router-dom' gives us { useParams }

It's the routing parameters

Now to create a detailed page for each restaurant in our components we will make "RestaurantMenu.js".

→
```
import { useParams } from "react-router-dom";
const RestMenu = () => {
    const params = useParams();
    const { id } = params
    return (
        <div>
            <h1> Restaurant id: 1234 </h1>
            <h2> Namaste </h2>
        </div>
    );
};
export default RestMenu;
```

(doing destructuring of param)

{id}

→ we will be having the id inside param.

Spiral

Teacher's Sign .....

Date.....

Making an API call in Restaurant detail page :

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

async function getRestaurantInfo() {
    const data = await fetch("https://---" + id);
    const json = await data.json();
}
```

→ <RestMenu/> component, at last will be like :

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

  const RestaurantMenu = () => {
      const { resId } = useParams();
      const [restaurant, setRestaurant] = useState(null);

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

Date.....

```
async function getRestaurantInfo() {

  const data = await fetch(
    "https://www.url" + resId);

  const json = await data.json();
  setRestaurant(json.data);
}

return !restaurant ? (<Shimmer/>) : (

  <div>
    <h1> Restaurant id : {resId} </h1>
    <h2> {restaurant.name} </h2>
    <h3> {restaurant.area} </h3>
    <h3> {restaurant.city} </h3>
    <h3> {restaurant.avgRating} </h3>
  </div>
  <div>
    <h1> Menu </h1>
    <ul>
    {
      Object.values(
        restaurant?.menu?.items)
      .map(( ) => {
      return ( <li key={item.id}> {item.name} </li>
      iska array banaega.
    }
    }
    </ul>
  </div>

};

export default RestaurantMenu;
```

this will make array of all values of obj { key (values) (key - value) (key - value)

Spiral

Teacher's Sign .....