Chapter 07: Finding the Path

▼ Some bad practices

- Never create a component inside component (React documentation). Think how many times inside component get renderred.
- Never write useState inside use if else statement
  - leads to inconsistency as react might not know about state etter
  - React likes concrete things and needs to know exactly number of state setters for component (so it should not be based on conditionals). Also don't use useState() in loop
- Never use useState outside of functional component, and we never have to do that as it doesn't make sense. useState is to create local state variable for component

Refer:

Browser History api and how react-router-dom uses it for client-side routing

Home v6.14.1

https://reactrouter.com/en/main

React Router

▼ React Router

- There are different library to handle routing in react applications. React-Router is mostly used. Its not developed by React team

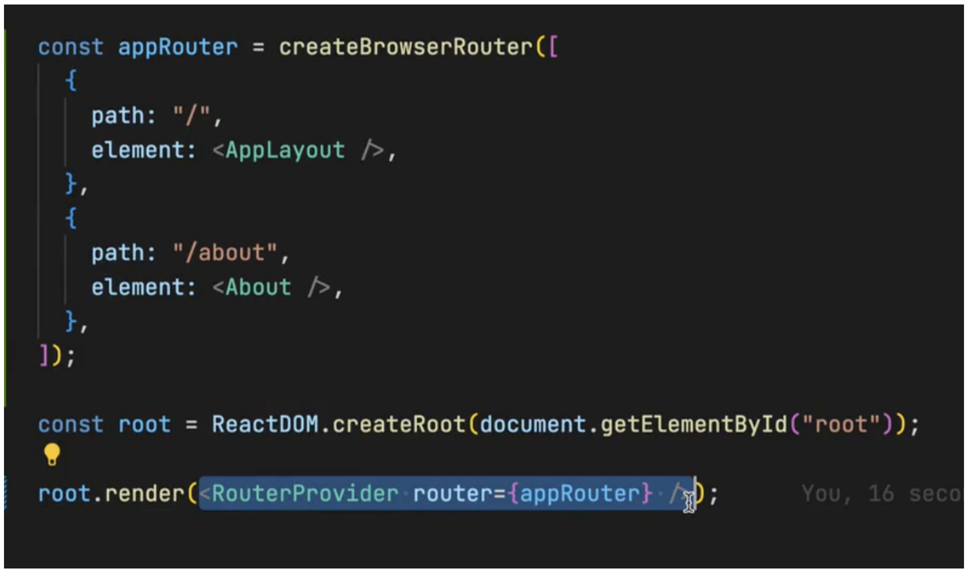
`createBrowserRouter`

- There are many types of Routers in react-route-dom library. We will be using `createBrowserRouter` as this is most preffered one for normal usecase

- Create routing configuration -using `createBrowserRouter`

`RouterProvider`

- This uses above configuration to provide routing facility
- Instead of passing `<App />` to `render()`, pass `RouterProvider`



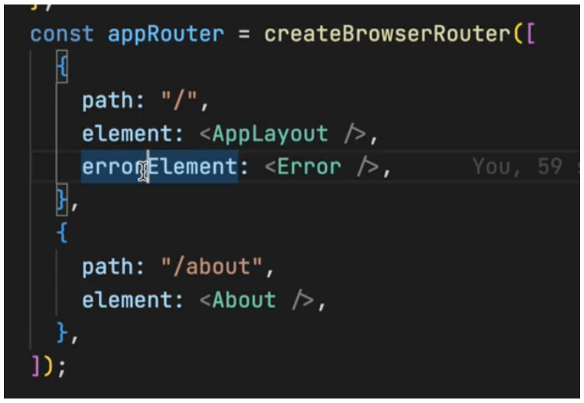
```
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
  },
  {
    path: "/about",
    element: <About />,
  },
]);

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<RouterProvider router={appRouter} />);        You, 16 seco
```

`errorElement`

- This field can be used to specify Error component to be rendered if any errors (like route not found 404)

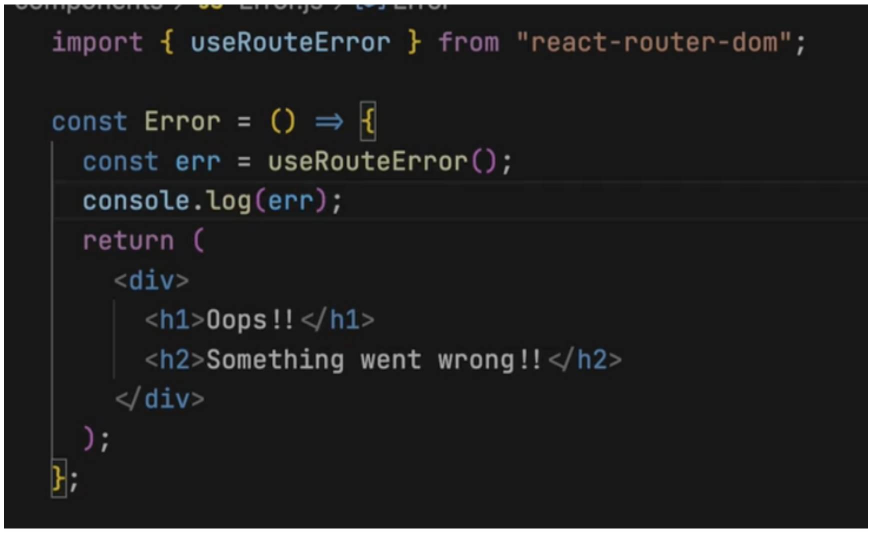


```
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,          You, 59
  },
  {
    path: "/about",
    element: <About />,
  },
]);
```

`useRouterError` **hook**

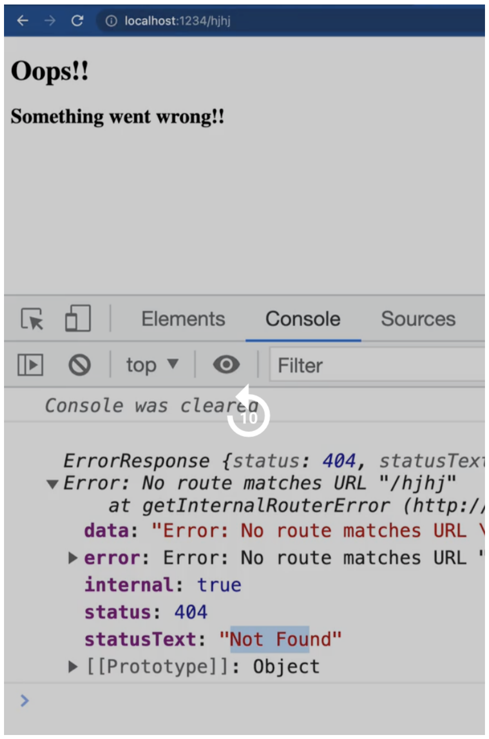
- This hook is from `react-router-dom`

- This can be used to capture infomation on any errors, using which we can render accurrate Error information to user



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

const Error = () => {
  const err = useRouteError();
  console.log(err);
  return (
    <div>
      <h1>Oops!!</h1>
      <h2>Something went wrong!!</h2>
    </div>
  );
};
```



Oops!!

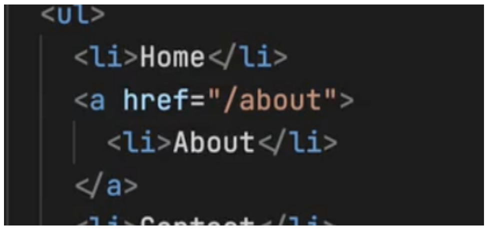
Something went wrong!!

```
Console was cleared

  ErrorResponse {status: 404, statusTex
 ▼Error: No route matches URL "/hjhj"
      at getInternalRouterError (http:/
    data: "Error: No route matches URL \
   ▶error: Error: No route matches URL '
    internal: true
    status: 404
    statusText: "Not Found"
   ▶[[Prototype]]: Object
```

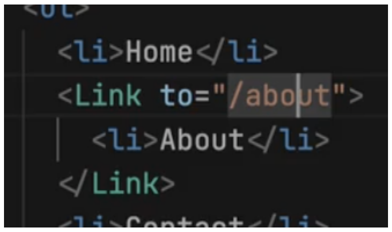
`<Link>`

- Problem with using directly <a> is it makes server call and releads whole page, which is not righ way for SPA (Single Page Applications)



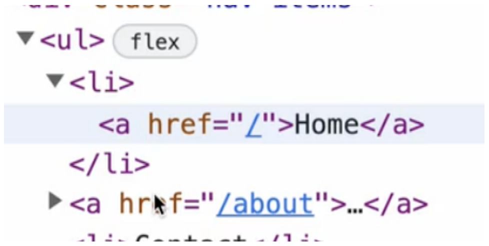
```
<ul>
  <li>Home</li>
  <a href="/about">
    <li>About</li>
  </a>
```

- There are two types of Routing
    - Server side routing : where all pages come from server
    - Client side routing: where we just load different component. No page reloads
- We are building Client side routing here
- react-router-dom provides Link component, which does same activity of <a>, but doesn't reload page



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

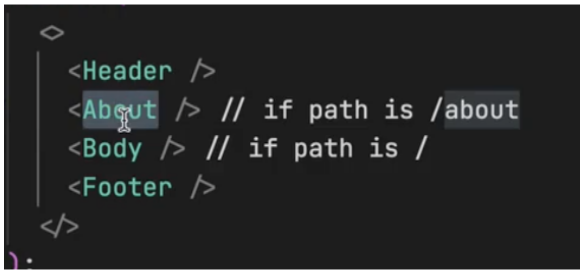
    - but if you see in UI, Its just an <a> at the end. react-router-dom does this black magic. Its awesome. Just like react keeps track of the states, react-router-dom keeps track of these Link. Browser understands only <a>, so its coverted to <a>. But clicking this react-router-dom loads component thats assoiciating with this Link (also adds entry in browser stack, and updates url in address bar, refer Browser History api and how react-router-dom uses it for client-side routing



```
▼<ul> flex
  ▼<li>
     <a href="/">Home</a>
   </li>
  ▶<a href="/about">…</a>
```

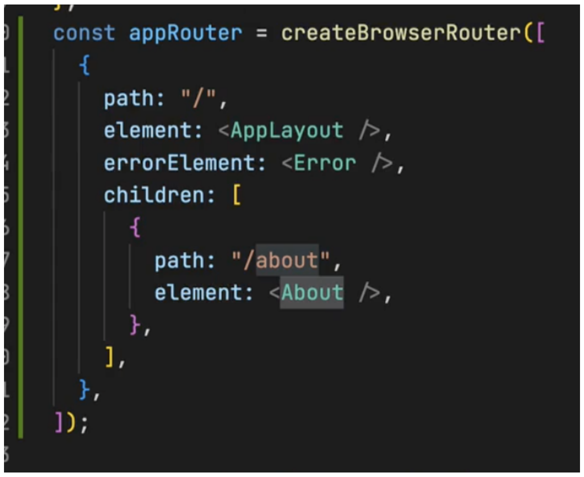
Nested Routing

- We want to keep header and footer always even when we change page. Something like this



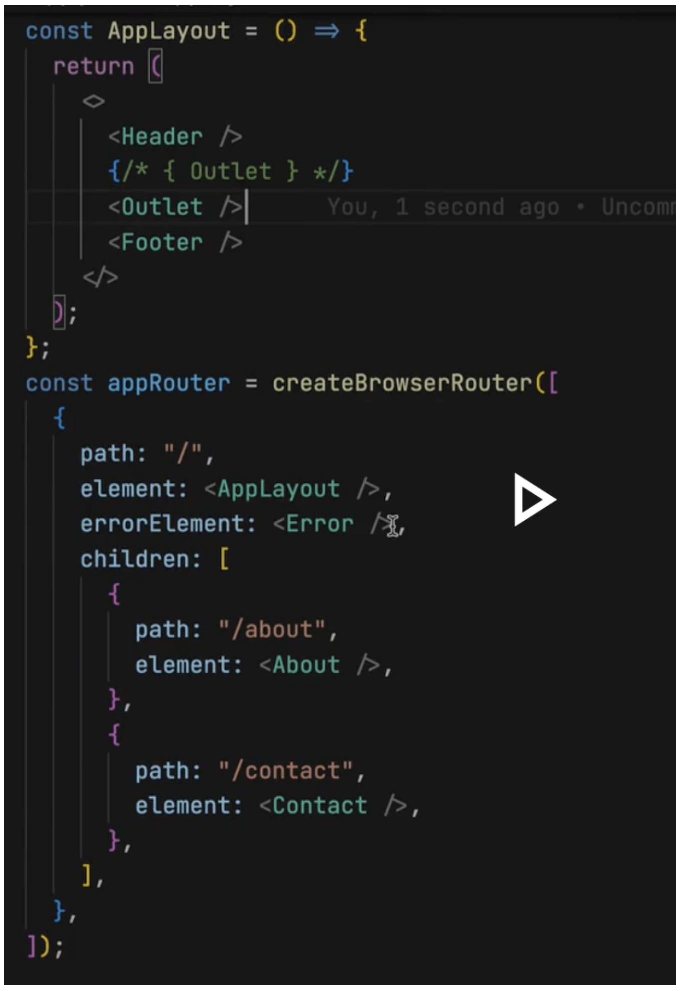
```
<>
  <Header />
  <About /> // if path is /about
  <Body /> // if path is /
  <Footer />
</>
);
```

- We can achieve this using Nested Routing. We can create children of route. Now `<About/>` children of `<AppLayout/>`



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

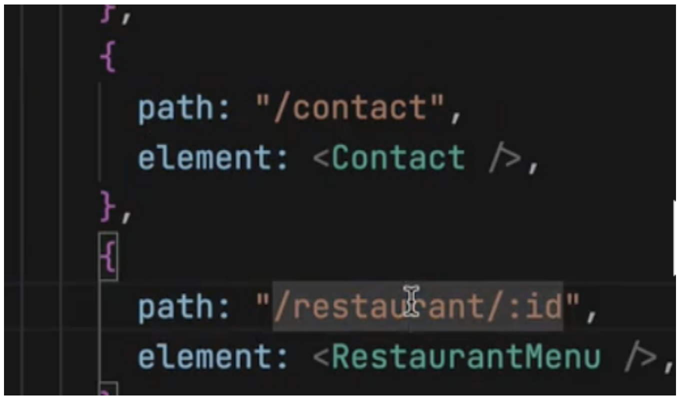
- `<Outlet/>`
    - react-router-dom gives us `Outlet` component, which will be filled by the children based on Route. One of the children will go into `Outlet` based on route. React does reconcilliation, will not rerender header and footer, but only rerenders html for children in `Outlet`



```
const AppLayout = () => {
  return (
    <>
      <Header />
      {/* { Outlet } */}
      <Outlet />        You, 1 second ago • Uncomm
      <Footer />
    </>
  );
};
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,
    children: [
      {
        path: "/about",
        element: <About />,
      },
      {
        path: "/contact",
        element: <Contact />,
      },
    ],
  },
]);
```

Dynamic Routing

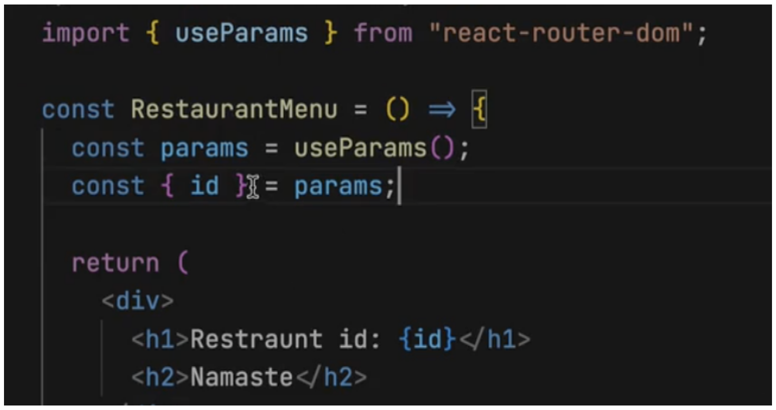
- If my route is `/restaurant/{id}` , here id can be dynamic value. If its rotiwala , router should route to / `restaurant/rotiwala`
- I want Router to render a component for any value of `id` i.e `/restaurant/*` , but pass `id` information to rendered component
- Configuration :



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

- So whatever url followed by `/restaurant/*` will render `<RestaurantMenu/>` component

- **`useParams() hook`**
  - We can read the dynamic url params using `useParams` hook from react-router-dom



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

const RestaurantMenu = () => {
  const params = useParams();
  const { id }= params;

  return (
    <div>
      <h1>Restraunt id: {id}</h1>
      <h2>Namaste</h2>
```

▼ Some Extra knowledge

- Cdn is better to save and access images because It caches image, its fast , and has good uptime
- Be concious when you import package. Don’t import package for every small things that you can do by yourself. Eg Adding Shimmer Effect
- FORMIK is best and most recommended library to create Forms in react . Lot of big companies are using it

Formik

React hooks and components for hassle-free form validation. The world's leading companies use Formik to build forms and surveys in React and React Native.

https://formik.org/

Homework

Object.values

extra homework

- build login page using Formik (default true)