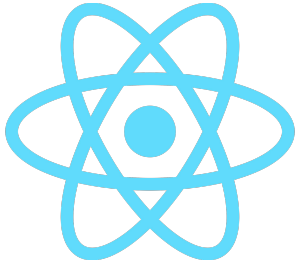


React.js

Lecture 3

Agenda

- Recap the last lecture.
- Routing
- HTTP Requests using Axios
- Questions!

Routing

Routing is the capacity to show different pages to the user. That means the user can move between different parts of an application by entering a URL or clicking on an element.

React at its core is a very simple library, and it does not dictate anything about routing, so you need to install it first.

`npm install react-router`

https://reactrouter.com/start/library/installation

Routing

<BrowserRouter>

A <BrowserRouter> stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in history stack.

<Routes>

Rendered anywhere in the app, <Routes> will match a set of child routes from the current location, Whenever the location changes, <Routes> looks through all its child routes to find the best match and renders that branch of the UI.

<Route>

Routes are perhaps the most important part of a React Router app. They couple URL segments to components, each route should specify 2 mandatory props the path and element which takes JSX component.

Routing

```
<BrowserRouter>
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about-us" element={<AboutUs />} />
        <Route path="/list" case element={<List />} />
    </Routes>
</BrowserRouter>
```

Routing

- To Define **not found** page use the * for the path

  <Route path="*" element={<h1>Not found page</h1>} />

- To Define **dynamic segments** page use the **:property** for the path

  <Route path="/details/**:id**" case element={<Details />} />

Routing-Link

A <Link> is an element that lets the user navigate to another page by clicking or tapping on it.a <Link> renders an accessible <a> element with a real href that points to the resource it's linking to.

<Link to="/">Home**</Link>**

<Link to="/about-us">About us**</Link>**

Routing - Active links [Extra]

Most web apps have persistent navigation sections at the top of the UI, the sidebar, and often multiple levels. Styling the active navigation items so the user knows where they are

Using NavLink: is a special kind of <Link> that knows whether or not it is "active" or "pending"

```
<NavLink
        to="/"
        className={({ isActive, isPending }) => {
            return isActive ? "active" : isPending ? "pending" : "";
        }}>
        Home
</NavLink>
```

Routing-Routing hooks

- useNavigate:
  - This hook tells you everything you need to know about a page navigation, can use the navigate function to go from route to other like **navigate("/products")**
- useParams :
  - The useParams hook returns an object of key/value pairs of the dynamic params from the current URL that were matched by the route.
- useLocation:
  - This hook returns the current location object.
- useSearchParams:
  - The useSearchParams hook is used to read and modify the query string in the URL for the current location.
  - const [searchParams, setSearchParams] = **useSearchParams()**;
  - **searchParams.get('paramName')**

Routing- Having Layout [Extra]

We will need to wrap the routes with a parent wrapper that holds the layout element with the navbar and footer for example.

```
<BrowserRouter>
    <Routes>
      <Route element={<Layout />}>
          <Route path="/" element={<Home />} />
      </Route>
      <Route path="/about-us" element={<AboutUs />} />
    </Routes>
</BrowserRouter>
```

Routing- Having Layout [Extra]

And inside the layout component we can use the navbar and the <Outlet /> element from react-router-dom to be as a placeholder for the components that match the route.

An <Outlet> should be used in parent route elements to render their child route elements. This allows nested UI to show up when child routes are rendered.

```
function Layout() {
    return (
        <>
            <Header />
            <Outlet />
        </>
    )
}
```

React Hooks

useEffect

React Hooks: useEffect

- Declare an effect. By default, your effect will run after every render.
- Specify the effect dependencies. Most effects should only rerun when needed rather than after every render. For example, a fade-in animation should only trigger when a component appears. Connecting and disconnecting to a chat room should only happen when the component appears and disappears or when the chat room changes. You will learn how to control this by specifying dependencies.
- Add cleanup if needed. Some effects need to specify how to stop, undo, or clean up whatever they were doing. For example, “connect" needs "disconnect," “subscribe" needs "unsubscribe," and “fetch" needs either “cancel" or "ignore." You will learn how to do this by returning a cleanup function.

https://www.freecodecamp.org/news/react-useeffect-absolute-beginners/

React Hooks: useEffect

Every React component goes through the same lifecycle:

- A component mounts when it's added to the screen.
- A component updates when it receives new props or state, usually in response to an interaction.
- A component unmounts when it's removed from the screen.

React Hooks: useEffect

useEffect :

accepts a function that can perform any side effects.

Usage :

```
useEffect(() => {
    return () => {
     // clean your code on unmount component
    }
},[arrayDependencies]);
```

[]: empty array means to fire on mount only

[dependencies]: means to fire on mount and with every change in this dependency value

React Hooks: useRef [self-study]

useRef :

It returns a ref object with a .current property. The ref object is mutable. Refs let a component hold some information that isn't used for rendering, like a DOM node or a timeout ID. Unlike with state, updating a ref does not re-render your component.

Usage :

Const inputRef = useRef();

<input ref={inputRef} ... />

Http Requests (Axios)

Axios

Axios is basically an external library, which is used to make promise-based HTTP calls

npm install axios

Creating an instance for shared config :

```
import axios from 'axios';

export const instance =
axios.create({
  baseURL: 'API BASE URL',
});
```

Axios

After initialize axios instance you can use it for API calling to integrate with backend server :

```
instance.get('/endpoint_path')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Let's check Axios Docs

Interceptors

[Extra]

Interceptors

Interceptors are functions that Axios calls for every request to transform the request before Axios sends it or transform the response before Axios returns the response to your code.

https://github.com/axios/axios

There are two types of interceptors:

- **Request interceptor:** this is called before the actual call to the endpoint is made.
- **Response interceptor:** this is called before the promise is completed and the data is received by the then callback.

Interceptors

Interceptors can be used, for example, for

- Show loader in request method and hide it in response one
- Set authorization header
- Send repeated params with the apis
- Set Accept-language key based on user-selected language
- Handling general errors returned from API

Thank you

Lap

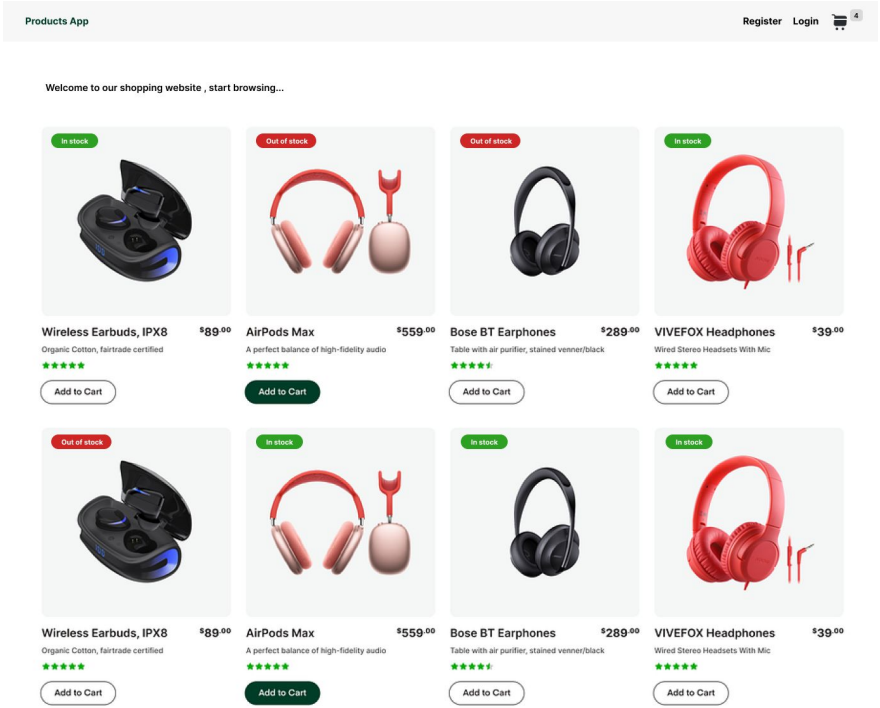
Task: Ecommerce App

1: Add navbar with links for different pages as shown in task (products list, Product details, cart , not found)

2: Products list screen will include products list, using this API

https://dummyjson.com/products

In the product list, if the stock key is 0, then it should show out of stock on the card; if it is greater than 0, then it should show in stock.



Task: Ecommerce App

3: When selecting a product, it should redirect to the product details screen and API to get data:

https://dummyjson.com/products/:id

4: [Bonus] Handle pagination for the products list using skip and limit to show 10 items per page.

Docs link for API:

https://dummyjson.com/docs/products#products-limit_skip