Chapter 08 - Let's get Classy

Theory Assignment:

- How do you create Nested Routes react-router-dom configuration
  You nest <Route> components inside each other.
  const appRouter = createBrowserRouter([
  {
  path: "/",
  element: <AppLayout />,
  children: [
  { path: "about", element: <About /> },
  { path: "contact", element: <Contact /> }
  ]
  }
  ]);
  Now /about and /contact render **inside** AppLayout using <Outlet />.

- Read abt createHashRouter, createMemoryRouter from React Router docs.
  - It's a router that uses **# in the URL** to manage navigation.
  - Useful when you're **hosting on static servers** where real path-based routing doesn't work.
  - URL looks like: `example.com/#/about`

  CreateMemoryRouter

  - A router that keeps the navigation **in memory**, not in the browser URL.
  - Mostly used for **testing** or **non-browser environments**.

- What is the order of life cycle method calls in Class Based Components
  - `constructor()`
  - `render()`
  - `componentDidMount()`
  - [on state/prop change] → `render()`, then `componentDidUpdate()`
  - [when unmounted] → `componentWillUnmount()`

- Why do we use componentDidMount?
  - Called **once** after the component is rendered.
  - Ideal for:
    - API calls
    - Setting up listeners or timers
    - DOM manipulation

- Why do we use componentWillUnmount? Show with example
  - Called **just before** the component is removed.
  - Used for cleanup:
    - Clearing timers
    - Removing event listeners
    - Cancelling API requests

```
componentWillUnmount() {
  clearInterval(this.timer);
}
```

- (Research) Why do we use super(props) in constructor?
- Required when using a constructor in a class that extends `React.Component`.
- It lets you use `this.props` inside the constructor.
- It calls the parent class's constructor and passes `props` to it.

- (Research) Why can't we have the callback function of useEffect async?
  - Because React expects useEffect to return:
- Nothing, or
- A **cleanup function** (not a Promise).
  - An async function **returns a Promise**, which React can't use for cleanup.

Instead, define an async function **inside** useEffect and call it:

```
useEffect(() => {
  const fetchData = async () => {
    const res = await fetch(...);
  };
  fetchData();
}, []);
```

Coding Assignment:

- Create a Class Based Component
  - Create 2 class based child components
  - Pass props from Parent to child
  - Create a constructor
  - Create a state variable inside child
  - Use this.setState to update it
  - What if there are multiple state variables?
  - Write a console.log for each lifecycle method
  - Play with the console logs to find out the correct order of their execution
- Create interval inside componentDidMount?
  - Use clearInterval to fix the issue caused by that interval

React Life Cycle Method Diagram - https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Code Link - https://bitbucket.org/namastedev/namaste-react-live/src/master/