Chapter 08 - Let's get Classy

**🔹 A) <div>**

* **Full form:** *Division*
* A **generic container element** in HTML.
* Doesn’t have any special meaning → just groups other elements together.
* In React, <div> is commonly used as a **wrapper**.

**Example:**

<div>

<h1>Hello</h1>

<p>This is inside a div</p>

</div>

**🔹 B) <nav>**

* **Semantic HTML element** introduced in HTML5.
* Specifically represents a **section of navigation links** (like menus, headers, sidebars).
* Helps **SEO** and **Accessibility** because search engines & screen readers know it’s navigation.

**Example:**

<nav>

<a href="/home">Home</a>

<a href="/about">About</a>

</nav>

**🔹 C) <hr>**

* **Horizontal Rule** → draws a **horizontal line** across the page.
* Used to separate content sections.
* Self-closing tag → doesn’t need </hr>.

**Example:**

<p>Section 1</p>

<hr />

<p>Section 2</p>

✅ **Interview one-liners:**

* *“<div> is a generic container with no semantic meaning.”*
* *“<nav> is a semantic HTML5 tag for navigation links, useful for SEO & accessibility.”*
* *“<hr> creates a horizontal line to visually separate content.”*

**1] Tell me about createBrowserRouter function in configuring a Router?**

* createBrowserRouter is a function, that is the **recommended way in React Router v6.4+** to create a router.
* It uses the **HTML5 History API** (no # in URL).
* It allows you to define routes as an **array of objects**, making config more structured.

**Example:**

A screen shot of a computer program

AI-generated content may be incorrect.

✅ **Interview one-liner:**  
*“createBrowserRouter is a function in React Router v6.4+ that creates a router using the browser’s history API, letting us configure routes as objects.”*

**2] How do we use RouterProvider component to provide a router to our App?**

* RouterProvider is a special component that **makes the router available to the entire React app**.
* Without it, routing won’t work.
* You pass your created router (from createBrowserRouter) into it.

**Example:**

A computer screen shot of a program code

AI-generated content may be incorrect.

✅ **Interview one-liner:**  
*“RouterProvider is the top-level component that takes a router object and makes routing available throughout the app.”*

**3] Explain how Outlet is used in React.**

* Outlet is a **placeholder component** provided by React Router where child routes are rendered.
* It tells React Router: *“When a child route is active, render that child’s UI here.”*

**🔹 Layout with Parent + Child Routes**

In real apps, we often have a **layout** (ex: Navbar, Footer, Sidebar) that stays the same across pages.  
Only the **middle content** changes based on the route.  
That middle content is where <Outlet /> comes in.

**🔹 What Happens Here?**

* The Layout (header, nav, footer) **always renders**. But child components like about page, contact page etc don’t render always, they render when ever we go to that route. So Outlet is the place where these child components render.
* <Outlet /> dynamically shows whichever child route matches:
  + /home → renders <Home /> inside <Outlet />
  + /about → renders <About /> inside <Outlet />

**Example:**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

✅ **Interview one-liner:**  
*“Outlet is a placeholder in parent routes where child route components get rendered. It’s used for layouts with nested routing.”*

⚡ Quick Recap:

* **createBrowserRouter** → defines routes as config objects.
* **RouterProvider** → provides router to the app.
* **Outlet** → renders child routes inside parent layout.

**4] How do you create Nested Routes in react-router-dom?**

👉 Nested routes allow us to render **child components inside parent routes**.

A screenshot of a computer program

AI-generated content may be incorrect.

🡪 "<BrowserRouter> and <Routes> are components provided by the react-router-dom library, not by React itself. They enable routing inside a React app."

🡪 Difference between BrowserRouter & createBrowserRouter

* "<BrowserRouter> is a component-based way of setting up routing. It’s simple and good for small apps. ((Old way, React Router v5/v6.0 → still supported))
* createBrowserRouter is a function-based configuration introduced in React Router v6.4, which allows advanced features like data loading, actions, and error handling.
* Both achieve routing, but createBrowserRouter is more modern and scalable."

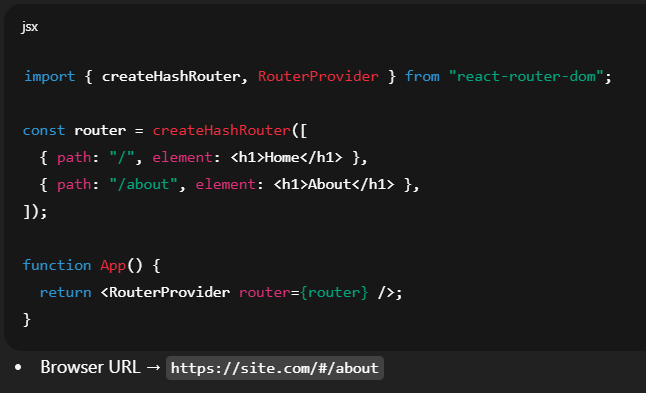
✅ **Interview one-liner:**  
*“Nested routes allow components to render inside parent routes, enabling layouts and hierarchical navigation.”*

**5] Read about createHashRouter, createMemoryRouter**

**🔹 createHashRouter**

* **What is it?** → It is a **function** provided by the react-router-dom library (NOT React itself).
* **What does it do?** → It creates a router that uses the **URL hash (#)** for routing.
* **Where is it used?** → In web apps where the server does not support client-side routing. (e.g., GitHub Pages).

✅ Example usage:



**🔹 createMemoryRouter**

* **What is it?** → It is also a **function** provided by the react-router-dom library.
* **What does it do?** → Creates a router that stores navigation history **in memory** (not in the browser URL).
* **Where is it used?** → In testing (React Testing Library), React Native, or Electron apps where you don’t have a browser address bar.

✅ Example usage:

A computer screen shot of a program

AI-generated content may be incorrect.

👉 So in an **interview** you should say:

“createHashRouter and createMemoryRouter are functions from the react-router-dom library. createHashRouter uses the hash part of the URL for routing, making it useful for static hosting like GitHub Pages. createMemoryRouter keeps the history in memory without touching the URL, mainly used in testing and non-browser environments like React Native.”

**6] What is the order of lifecycle method calls in Class Based Components?**

For **Mounting (when added to DOM):**

1. constructor()
2. render()
3. componentDidMount()

For **Updating (when props/state changes):**

1. render()
2. componentDidUpdate()

For **Unmounting (when removed):**

1. componentWillUnmount()

✅ **Interview one-liner:**  
*“Mount → constructor → render → componentDidMount. Update → render → componentDidUpdate. Unmount → componentWillUnmount.”*

**7] Why do we use componentDidMount?**

* Runs **once after the component is mounted**. (component is **created** and **inserted into the DOM** (the browser’s HTML tree).)
* Used for:
  + Fetch API calls
  + Start timers & Add event listeners
  + Do anything that requires the component to be present in the DOM

Example:

A computer code on a black background

AI-generated content may be incorrect.

**8] Why do we use componentWillUnmount? Show with example.**

* Runs **before component is removed**.
* Used for cleanup → remove event listeners, clear timers, cancel subscriptions.

Example:

A computer screen shot of a program code

AI-generated content may be incorrect.

**9] Why do we use super(props) in constructor?**

* In class components, we extend React.Component
* super(props) calls the parent constructor and makes this.props available inside constructor.
* Without it → this.props will be undefined in constructor.

Example:

A screen shot of a computer code

AI-generated content may be incorrect.

**10] Why can't we have the callback function of useEffect async?**

🡪**What is useEffect?**

* useEffect is a **React Hook** used to perform **side effects** in functional components.

**Side effects** are things that affect the outside world (not just returning JSX), like:

* Fetching data from an API
* Setting timers (setTimeout, setInterval)
* Subscribing/unsubscribing to events
* Updating the document title
* Why not callback functions !!
* useEffect expects the callback to **either return nothing** or a **cleanup function**.
* If we mark it async, it will return a **Promise** instead, which React doesn’t accept as cleanup.
* ✅ Solution → define an inner async function.

A screenshot of a computer program

AI-generated content may be incorrect.

✅ **Interview-ready one-liners:**

* *“componentDidMount is used for initialization tasks like API calls.”*
* *“componentWillUnmount is used for cleanup (remove listeners, clear timers).”*
* *“super(props) lets us access this.props inside constructor.”*
* *“useEffect cannot be async because it must return either nothing or a cleanup function, not a Promise.”*