React Context API and Router Components

# 1. Explain the Need and Benefits of React Context API

In React, data is often passed from parent to child using props. This can become cumbersome in larger applications where many components require access to the same data. This is referred to as "prop drilling," and it makes the code harder to manage and debug.  
  
The React Context API provides a way to share values like theme, user authentication, and settings between components without having to pass props manually at every level of the component tree.

## Benefits of React Context API:

1. Avoids prop drilling.  
2. Makes state and data sharing easy between nested components.  
3. Improves code readability and maintainability.  
4. Centralized management of app-wide data (such as themes, languages, or user info).  
5. Easily combined with useReducer or other state management techniques for global state control.

# 2. Working with createContext()

The `createContext()` function is used to create a Context object. This object can then be used to provide and consume values across the component tree.

## Steps to Use createContext():

1. Import createContext from React:  
 `import React, { createContext, useContext, useState } from 'react';`  
   
2. Create a Context:  
 `const MyContext = createContext();`  
  
3. Create a Provider component:

const MyProvider = ({ children }) => {  
 const [value, setValue] = useState('Hello from Context!');  
 return (  
 <MyContext.Provider value={{ value, setValue }}>  
 {children}  
 </MyContext.Provider>  
 );  
 };

4. Use the Provider in the component tree:  
 `<MyProvider><App /></MyProvider>`  
  
5. Consume the context using useContext:  
 `const { value } = useContext(MyContext);`

# 3. List the Types of Router Components

React Router is used to handle routing in React applications. It helps to navigate between components, manage URLs, and display the appropriate view for the current URL.

## Types of Router Components:

1. \*\*BrowserRouter\*\* – Uses HTML5 history API to keep the UI in sync with the URL.  
 `import { BrowserRouter } from 'react-router-dom';`  
  
2. \*\*HashRouter\*\* – Uses the hash portion of the URL (`window.location.hash`) to keep the UI in sync.  
 Useful for static sites where server configuration isn’t possible.  
  
3. \*\*MemoryRouter\*\* – Keeps the history of your “URL” in memory (does not read or write to the address bar).  
 Commonly used for testing or non-browser environments like React Native.  
  
4. \*\*StaticRouter\*\* – Used for server-side rendering. It doesn't change the location or interact with the browser.  
  
5. \*\*NativeRouter\*\* – Used with React Native applications to handle routing.  
  
6. \*\*Routes and Route\*\* – Define and render individual routes inside the Router.  
 `import { Routes, Route } from 'react-router-dom';`  
  
7. \*\*Link and NavLink\*\* – Used for navigation. NavLink can apply styles to the active link.  
 `import { Link, NavLink } from 'react-router-dom';`