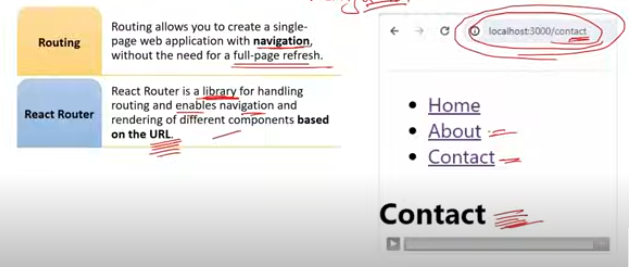
Q1)What is Routing and Router in React ?



In React, routing is the mechanism that allows you to navigate between different views or components in your application without reloading the entire page. It’s essential for creating single-page applications (SPAs) where different parts of the app are displayed based on the URL.

**Router** is a component that manages the navigation of your application. React Router is a popular library for handling routing in React applications. Here’s a brief overview:

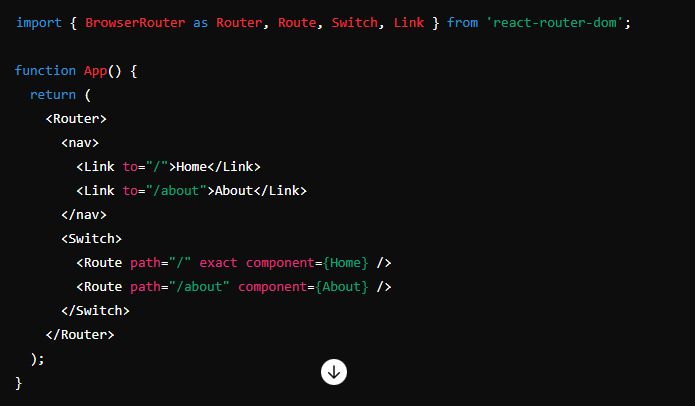
Router**:** This is the core component that keeps the UI in sync with the URL. It uses the browser's history API to manage the URL and navigation.

Route**:** This component is used to define which component should be rendered when a specific URL path is matched. For example, <Route path="/about" component={About} /> renders the About component when the URL is /about.

Switch**:** This component is used to group Route components and ensure that only the first matching route is rendered.

Link**:** This component is used to create navigation links that update the URL and render the corresponding component without refreshing the page.BrowserRouter**:** A common Router implementation that uses the HTML5 history API to keep the UI in sync with the URL.

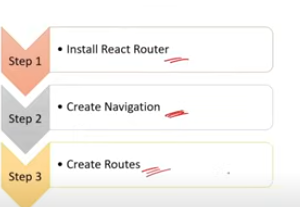
Example usage:



In this example, BrowserRouter wraps the application, Link components create navigation links, and Route components define which component should be displayed based on the URL.

Q2)How to implement Routing in React ?



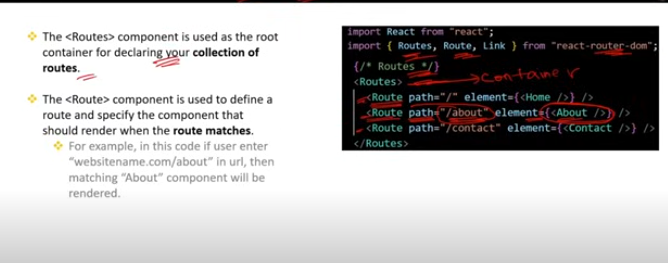


**Routing in React** involves using the react-router-dom library.

* **Router**: Manages navigation and URL synchronization.
* **Route**: Defines which component should render for a specific path.
* **Switch**: Ensures only one route is rendered at a time.
* **Link**: Creates navigation links that update the URL without reloading the page.

To implement routing, you install react-router-dom, wrap your app with BrowserRouter, define routes with Route, and use Link for navigation.

Q3)What are the roles of <Routes> and <Route> component in React Routing ?



### Roles of <Routes> and <Route> in React Routing

**React Router** is a popular library for handling routing in React applications. <Routes> and <Route> are key components used for defining and rendering routes.

### <Routes> Component

* **Role**: Acts as a container for all <Route> elements. It matches the URL with the defined routes and renders the appropriate component.
* **Usage**: You wrap all your <Route> components inside <Routes> to define the routing configuration.

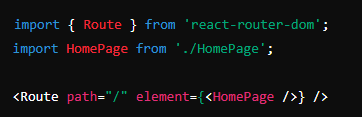
**Example**:



### <Route> Component

* **Role**: Defines a single route within the <Routes> container. It specifies the path and the component to render when the path matches the URL.
* **Attributes**:
  + path: The URL pattern that the route matches.
  + element: The component to render when the path matches.

**Example**:

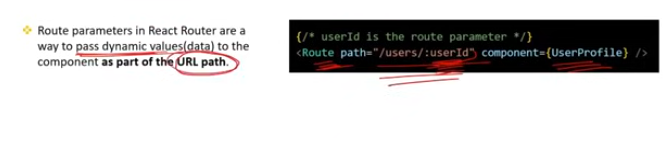


### Summary

* <Routes>: A container for defining multiple routes. It processes and matches the URL against all child <Route> components.
* <Route>: Defines a single route with a specific path and the component to render.

In modern React Router (v6+), <Routes> and <Route> are used together to handle routing in a more declarative way compared to previous versions.

Q4)What are Route Parameters in React Routing ?



### Route Parameters in React Routing

**Route Parameters** are dynamic segments in the URL that can be used to pass data to a route. They are part of the URL path and can be accessed inside your route components.

### Key Points:

**Definition**:

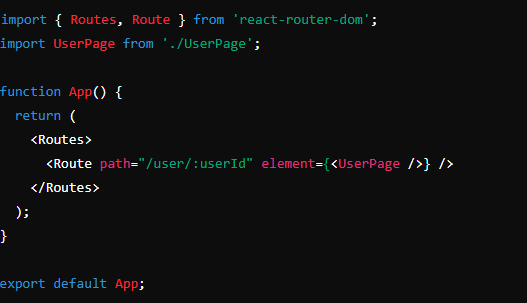
* 1. Route parameters are placeholders in the URL path that are defined using a colon (:) followed by the parameter name.

**Usage**:

* 1. They allow you to create dynamic routes where the path can include variable data, such as IDs or usernames.

### Example:

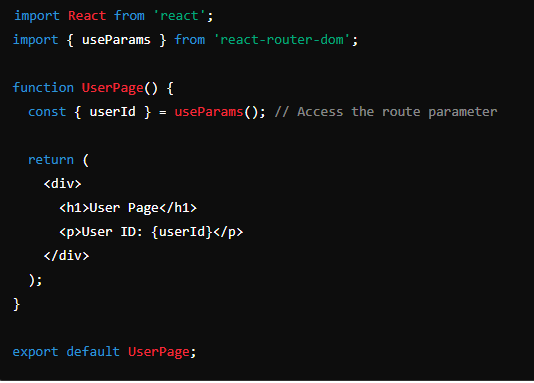
#### Defining Route with Parameters:



In this example, :userId is a route parameter.

#### Accessing Route Parameters:

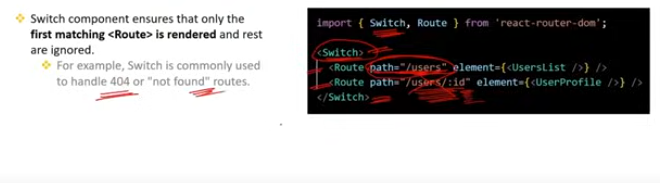
* Use the useParams hook to access the parameters inside the component.



### Summary

* **Route Parameters**: Dynamic segments in the URL path defined with a colon (e.g., :userId).
* **Access**: Use the useParams hook to retrieve parameter values in your component.

Q5)What is the role of Switch Component in React Routing ?



### Role of <Switch> Component in React Routing

<Switch> is a component used in React Router (v5 and earlier) to ensure that only one route is rendered at a time based on the current URL.

### Key Points:

**Exclusive Rendering**:

* 1. <Switch> renders the first <Route> or <Redirect> that matches the current URL. If a route is matched, the rest of the routes inside the <Switch> are ignored.

**Order Matters**:

* 1. Routes inside <Switch> are evaluated in the order they appear. The first matching route will be rendered, so the order of routes can affect which component is displayed.

**Usage**:

* 1. Wrap your <Route> components inside <Switch> to handle routing and ensure that only one route is active at a time.

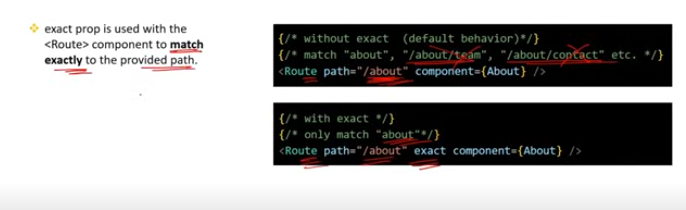
### Example:



### Summary

* **Purpose**: Ensures that only one route is rendered based on the URL, evaluating routes in order and stopping when a match is found.
* **Replacement**: In React Router v6 and later, <Switch> is replaced by <Routes>, which provides similar functionality with a new API.

Q6)What is the role of exact prop in React Routing ?



### Role of exact Prop in React Routing

In **React Router v5** and earlier, the exact prop is used with the <Route> component to control how route matching occurs.

### Key Points:

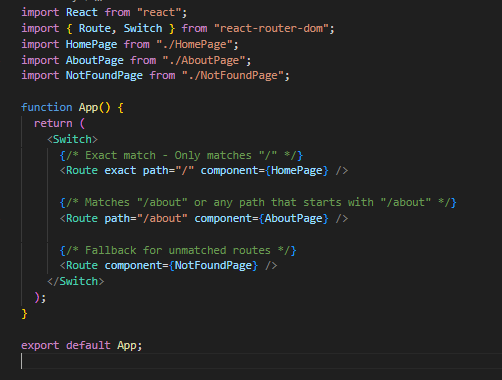
**Exact Matching:**

* 1. **Purpose**: Ensures that the route will only match if the URL exactly matches the path specified.
  2. **Usage**: Prevents the route from matching when the URL includes additional segments beyond the specified path.

**Default Behavior**:

* 1. Without the exact prop, a route will match any URL that starts with the specified path. This can lead to unintended matches when the URL has additional segments.

### Example:



In this example:

* The route with exact will match only / and not /about or any other path.
* The route without exact will match /about and any sub-path like /about/team, /about/company, etc.

### Summary:

* exact **Prop**: Ensures a route matches only when the URL exactly matches the specified path.
* **Default Matching**: Without exact, the route will match any URL that begins with the specified path.

In **React Router v6** and later, the exact prop is no longer needed because routes are matched exactly by default.