**Index: React JS Topics**

1. **React JS Introduction**  
   Overview of React and its benefits in building UIs.
2. **Introduction to NPM**  
   Tool for managing Node.js packages used in React projects.  
   *Example:* Installing libraries like react or react-dom.
3. **React Components**
   * **Function-Based Components:** Simple components using functions.  
     *Example:* function Header() { return <h1>Hello</h1>; }
   * **Class-Based Components:** Components using class.  
     *Example:* class Header extends React.Component { render() { return <h1>Hello</h1>; } }
4. **JSX (JavaScript XML)**  
   Combines HTML and JavaScript.  
   *Example:* <h1>{5 + 5}</h1> renders 10.
5. **Single Page Application (SPA)**  
   React creates dynamic, fast-loading web pages without refreshing the page.
6. **Styling in React JS**  
   Apply CSS using classes, inline styles, or CSS modules.  
   *Example:* <h1 style={{ color: "blue" }}>Hello</h1>.
7. **Event Handlers in React JS**  
   Handle user actions like clicks.  
   *Example:* <button onClick={() => alert("Clicked!")}>Click Me</button>.
8. **Data Flow in React JS**  
   Unidirectional data flow: Parent passes data to children.
9. **Data Management**  
   Use **states** and **props** to manage and share data in components.
10. **States in React JS**  
    Tracks dynamic data within a component.  
    *Example:* const [count, setCount] = useState(0);
11. **Props in React JS**  
    Pass data from parent to child components.  
    *Example:* <ChildComponent name="React" />.
12. **Hooks in React JS**
    * **useState:** Manages state in function components.  
      *Example:* const [count, setCount] = useState(0);
    * **useEffect:** Runs side effects like API calls.  
      *Example:* useEffect(() => { console.log("Mounted"); }, []);
    * **useRef:** Accesses DOM elements.  
      *Example:* const inputRef = useRef(null);
    * **useContext:** Shares data across components.  
      *Example:* const value = useContext(MyContext);
13. **Routing in React JS**  
    Enable navigation between pages without reloading.
14. **Installing React Router DOM**  
    Install library for routing:  
    *Command:* npm install react-router-dom.
15. **Browser Router**  
    Wraps the app to enable routing.  
    *Example:* <BrowserRouter><App /></BrowserRouter>.
16. **useNavigate and useLocation**
    * **useNavigate:** Navigate programmatically.  
      *Example:* navigate('/home');
    * **useLocation:** Access current URL.  
      *Example:* const location = useLocation();
17. **Axios in React JS**  
    Fetch data from APIs.  
    *Example:* axios.get('/api/data').then(response => console.log(response.data));
18. **CRUD in React JS**  
    Create, Read, Update, and Delete data in a React app.
19. **JSON Server in React JS**  
    Mock backend for testing CRUD operations.  
    *Command:* npx json-server --watch db.json.

**Detailed answers**

**React JS Topics - Detailed Points for Interview**

1. **Introduction to React JS**
   * React is maintained by Facebook.
   * Based on **Component-Based Architecture** for building UIs.
   * Uses a **virtual DOM** for efficient rendering.
   * React applications are declarative, making code predictable and easier to debug.
   * React can be combined with libraries like Redux for state management.
2. **Introduction to NPM**
   * NPM is used for installing libraries, managing dependencies, and running scripts in a React project.
   * package.json keeps track of dependencies.
   * Alternatives: Yarn, pnpm.
   * Command: npm start to run a React app.
3. **Components in React JS**
   * Components can be **reusable** and **independent**.
   * Functional components support **React Hooks**; class components use lifecycle methods.
   * Example: <Header /> represents a self-contained piece of UI.
   * Types: Presentational (UI-focused) and Container (logic-focused).
4. **JSX Component**
   * Allows writing HTML-like syntax in JavaScript files.
   * JSX gets transpiled to React.createElement.
   * Supports embedding expressions using {}.  
     *Example:* <h1>Hello, {name}</h1>.
5. **Single Page Application (SPA)**
   * SPAs improve performance by loading resources once and updating content dynamically.
   * React uses **React Router** for SPA navigation.
   * Example: Gmail is an SPA.
6. **Styling in React JS**
   * Inline styles: { color: 'blue' }
   * External CSS: Link stylesheet using <link />.
   * CSS Modules: Scoped to components to avoid conflicts.
   * Styled Components: Write styles as JavaScript.
7. **Event Handlers in React JS**
   * React uses synthetic events for cross-browser compatibility.
   * Event binding can be done inline or in the constructor.
   * Example:
   * function handleClick() { console.log('Clicked'); }
   * <button onClick={handleClick}>Click</button>;
8. **Data Flow in React JS**
   * Data flows **unidirectionally** from parent to child using props.
   * Child-to-parent communication uses **callback functions** passed as props.
9. **Data Management**
   * Local state using useState or class state.
   * Global state using **context** or libraries like **Redux**.
   * Example:
   * const [data, setData] = useState([]);
10. **States in React JS**
    * Represents the component's dynamic data.
    * Changes in state trigger re-renders.
    * Example:
    * const [counter, setCounter] = useState(0);
    * setCounter(counter + 1);
11. **Props in React JS**
    * Props are **immutable** and passed from parent to child.
    * They help components stay reusable.  
      *Example:*
12. <ChildComponent name="React" />;
13. **Hooks in React JS**
    * **useState:** Manages state in functional components.
    * **useEffect:** Handles side effects (e.g., API calls).
    * **useRef:** Maintains references to DOM nodes or variables.
    * **useContext:** Simplifies global state sharing.
14. **Routing in React JS**
    * Helps create navigable views in SPAs.
    * Key components: <Route>, <Switch>, <Link>, and <NavLink>.
    * Example:
    * <Route path="/about" element={<About />} />;
15. **Installing React Router Dom**
    * Install using: npm install react-router-dom.
    * Import BrowserRouter to wrap your application for routing support.
16. **Browser Router in React JS**
    * Acts as the parent wrapper for routing.
    * Example:
    * <BrowserRouter>
    * <App />
    * </BrowserRouter>;
17. **useNavigate and useLocation in React JS**
    * **useNavigate:** Navigate programmatically.  
      Example:
    * const navigate = useNavigate();
    * navigate('/home');
    * **useLocation:** Fetch current URL details.  
      Example:
    * const location = useLocation();
    * console.log(location.pathname);
18. **Axios in React JS**
    * A library for HTTP requests.
    * Simplifies GET, POST, PUT, DELETE operations.
    * Example:
    * axios.get('/api/users').then((response) => console.log(response.data));
19. **CRUD Using React JS**
    * **Create:** Form to add data.
    * **Read:** Display data from an API.
    * **Update:** Edit existing data with forms.
    * **Delete:** Remove items from the list.
20. **JSON Server in React JS**
    * A lightweight mock backend for testing APIs.
    * Example:
    * npx json-server --watch db.json