React recall

**1. JSX (JavaScript XML)**

* Syntax similar to HTML used in React components.
* Embedding expressions using curly braces {}.
* Components must return a single element.

**2. Components**

* **Functional Components**: Written as JavaScript functions.
* **Class Components**: Written as ES6 classes (less commonly used after React hooks).
* **Props**: Passing data to components.
* **State**: Managing component-specific data.

**3. State and Lifecycle**

* **useState Hook**: Managing state in functional components.
* **useEffect Hook**: Handling side effects, similar to lifecycle methods like componentDidMount, componentDidUpdate, componentWillUnmount.
* State immutability and updating patterns.

**4. Handling Events**

* Handling user inputs and events like onClick, onChange, onSubmit.
* Event binding and synthetic events.

**5. Conditional Rendering**

* if statements and ternary operators for rendering UI conditionally.
* Rendering components based on state or props.

**6. Lists and Keys**

* Rendering lists of elements using .map().
* Using key props to help React identify items efficiently.

**7. Forms and Input Handling**

* Handling form inputs with controlled components.
* Capturing input values with onChange events and state.

**8. React Router**

* Setting up routing with react-router-dom.
* **Routes**: Defining routes for different components.
* **Link**: Navigating between pages.

**9. Context API**

* Prop drilling and passing data deeply through component trees.
* Sharing state globally using React.createContext and useContext hook.

**10. React Hooks**

* **useState**: Managing state in functional components.
* **useEffect**: Managing side effects like API calls.
* **useContext**: Consuming context values.
* **useRef**: Accessing DOM elements and storing mutable values.
* **useReducer**: Advanced state management similar to Redux.

**11. Higher-Order Components (HOCs)**

* Functions that take a component and return a new component to enhance its behavior.

**12. React Fragments**

* Grouping multiple elements without adding extra nodes to the DOM using <React.Fragment> or <>.

**13. Error Boundaries**

* Handling errors in component trees with class components (less common in hooks).

**14. Optimizing Performance**

* **Memoization** with React.memo and useMemo.
* **useCallback** to memoize functions.
* **Lazy Loading** and code splitting using React.lazy and Suspense.

**15. PropTypes**

* Validating props using prop-types package.

**16. React DevTools**

* Debugging and inspecting React components in the browser.

**17. State Management Libraries**

* **Redux**: Managing global state.
* **Recoil**, **Zustand**, **MobX**: Alternatives to Redux.