**🔥 React Basics (JSX & Hooks)**

19. JSX

\* JavaScript syntax extension that looks like HTML.

\* Used to define React components’ UI.

```jsx

const element = <h1>Hello, world!</h1>;

```

**Components**

\* Reusable UI building blocks.

\* Function components receive `props` as input.

**Props**

\*Data passed from parent to child components.

**State**

\* Internal data of a component that can change over time.

**23. Hooks: `useState` and `useEffect`**

\* `useState`: manages component state.

```jsx

const [count, setCount] = useState(0);

```

\* `useEffect`: runs side effects like data fetching or subscriptions.

```jsx

useEffect(() => {

// runs after component mounts or updates

}, [dependencies]);

```

**📙 React Advanced Concepts Notes (Expanded)**

**⚛️ React State Management & Hooks**

**`useContext`**

\* The Context API allows components to share values like themes, user info, or settings without manually passing props through every level.

\* This avoids “prop drilling,” which is passing props through many intermediate components unnecessarily.

\* Using `useContext` hook, any component within the context provider’s tree can consume the shared value directly, making the app easier to maintain and less cluttered.

Example:

```jsx

const ThemeContext = React.createContext('light');

function App() {

return (

<ThemeContext.Provider value="dark">

<Toolbar />

</ThemeContext.Provider>

);

}

function Toolbar() {

const theme = React.useContext(ThemeContext);

return <div className={`toolbar ${theme}`}>Current theme: {theme}</div>;

}

```

Here, `Toolbar` accesses the `theme` directly without prop passing.