React Main Topics

@frontend_in_depth

Components:

- **Functional Components:** Stateless components defined as JavaScript functions.
- **Class Components:** Components defined as ES6 classes, allowing for state and lifecycle methods.
- **JSX:** Syntax extension for JavaScript, allowing the embedding of HTML-like syntax within JavaScript code to define React elements.

Props and State:

- **Props (Properties):** Data passed to a component from its parent, immutable within the component.
- **State:** Data managed within a component, mutable and triggers UI updates when changed using setState().

Lifecycle Methods:

- **Mounting Lifecycle:** Methods invoked when a component is being created and inserted int the
- **DOM.**
- **render()**
- **componentDidMount()**

- **Updating Lifecycle:** Methods invoked when a component is being re-rendered as a result of changes to props or state.
- **should ComponentUpdate()**
- **render()**
- **componentDidUpdate()**

  Unmounting Lifecycle: Method invoked when a component is being removed from the DOM.
- **componentWillUnmount()**

Hooks (introduced in React 16.8):

- **useState:** Hook for adding state to functional Components.
- **useEffect:** Hook for handling side effects in functional components (e.g., data fetching, subscriptions).
- **useContext:** Hook for accessing context in functional components.
- **useReducer:** Hook for managing state transitions with actions in functional components.
- **useRef:** Hook for accessing DOM elements or React elements.

JSX Elements:

- **HTML Elements:** HTML-like elements representing the UI structure.
- **React Components:** Custom components defined by developers.
- **Expressions:** JavaScript expressions enclosed in curly braces {} for dynamic content rendering.

Event Handling:

- **onClick, onChange, onSubmit, etc.:** Event handlers for handling user interactions.
- **Synthetic Events:** Cross-browser wrapper for native events, providing consistent event handling.

Conditional Rendering:

- **Conditional Statements:** if statements or ternary operators for conditional rendering.
- **Logical && Operator:** Conditional rendering based on a logical condition.

Lists and Keys:

- **Lists:** Rendering arrays of data as a list of elements using map) function.
- **Keys:** Unique identifiers for list items, aiding React in efficient re-rendering.

Forms:

- **Controlled Components:** Form inputs whose value is controlled by React state.
- **Uncontrolled Components:** Form inputs that maintain their own state.

Context:

- **React Context:** Mechanism for passing datathrough the component tree without manuallypassing props at every level.

Fragments:

- **React Fragments:** Wrapper for groupingmultiple elements without adding extra nodesto the DOM.

Error Boundaries:

- **Error Boundary:** A component that catchesJavaScript errors in its child componentsubtree.

High-Level Concepts:

- **Virtual DOM:** React's abstraction of thebrowser's DOM, enabling efficient updates.
- **Reconciliation:** Process of updating the DOMto match React's virtual DOM representation.
- **Component Composition:** Building UIs bycombining smaller, reusable components.
- **One-Way Data Flow:** Data flows down thecomponent hierarchy, simplifying statemanagement.

Additional Concepts:

- **Hooks Customization:** Writing custom hooks for reusing stateful logic.
- **Code Splitting:** Splitting the code into smaller chunks for better performance.
- **Server-Side Rendering (SSR):** Rendering React components on the server side before sending them to the client.
- **Static Type Checking:** Using tools like PropTypes or TypeScript for static type checking.
- **State Management Libraries:** Integration with state management libraries like Redux or MobX for managing complex application state.

What new did you learn today?

LET ME KNOW IN COMMENTS !!