**\*\* INTRODUCTION TO REACT JS \*\***

**Question 1: What is React.js? How is it different from other JavaScript frameworks and libraries?**

React.js is an open-source JavaScript library developed by Facebook for building user interfaces, especially for single-page applications. It allows developers to create reusable UI components and efficiently update and render UI elements based on state changes.

Differences from other frameworks/libraries:

*Virtual DOM:* React uses a virtual DOM to optimize rendering, making UI updates faster than traditional DOM manipulation.

*Component-Based Architecture*: React applications are built using reusable components, making development modular and efficient.

*Unidirectional Data Flow*: React enforces a one-way data flow, making state management more predictable.

*Declarative UI:* Instead of directly manipulating the DOM, developers describe how the UI should look, and React updates it accordingly.

**Question 2: Explain the core principles of React such as the virtual DOM and component-based architecture.**

Virtual DOM: React creates a lightweight copy of the actual DOM, known as the virtual DOM.

When a component's state changes, React updates the virtual DOM first, compares it with the previous version, and efficiently updates only the changed parts in the real DOM (diffing and reconciliation).

Component-Based Architecture: UI is broken into independent, reusable components.Each component manages its own state and can be combined to form complex interfaces.

Components can be functional (stateless) or class-based (stateful).

**Question 3: What are the advantages of using React.js in web development?**

Performance: The virtual DOM improves rendering speed.

Reusability: Components can be reused across different parts of an application.

Flexibility: Can be used with other libraries and frameworks like Redux, Next.js, or even in non-SPA projects.

SEO-Friendly: Faster rendering improves search engine rankings compared to traditional client-side frameworks.

**\*\* COMPONENTS \*\***

**Question 1: What are components in React? Explain the difference between functional components and class components.**

Components in React are reusable, self-contained pieces of UI that allow developers to build complex interfaces efficiently. They help break down the UI into smaller, manageable parts.

There are two types of components:

Functional Components – These are simple JavaScript functions that return UI elements. They are easier to write and maintain. Since React 16.8, functional components can manage state and side effects using Hooks.

Class Components – These are ES6 classes that extend React's Component class. They include lifecycle methods and use this.state to manage data changes within the component.

The main difference is that functional components rely on Hooks for state management, whereas class components use this.state and lifecycle methods. Functional components are preferred in modern React development due to their simplicity and performance benefits.

**Question 2: How do you pass data to a component using props?**

Props (short for properties) are used to pass data from a parent component to a child component in React. They allow components to be dynamic and reusable. Props are read-only, meaning a child component cannot modify the props it receives.

To pass props, the parent component provides them as attributes, and the child component accesses them. This mechanism enables data sharing between components, making them more flexible and customizable.

**Question 3: What is the role of render() in class components?**

The render() method in class components is responsible for defining the component's UI by returning React elements. It is a required method in every class component and is automatically called whenever there is a change in the component's state or props.

React relies on render() to update the UI efficiently by comparing the new output with the previous one using the Virtual DOM. The method should always return a single parent element containing the component's structure.

**\*\* PROPS AND STATE \*\***

**Question 1: What are props in React.js? How are props different from state?**

Props are external data passed from a parent component to a child component. They make components reusable by allowing them to receive dynamic values. Since props are immutable, they cannot be modified by the child component that receives them.

State, on the other hand, is internal to a component and represents data that can change over time. It is managed within the component and updated using functions like setState() in class components or useState() in functional components.

The key difference is that props are controlled by the parent and remain unchanged in the child, while state is managed within the component and can be modified based on user interactions or other events.

**Question 2: Explain the concept of state in React and how it is used to manage component data.**

State is an object that holds dynamic data in a React component. Unlike props, which are passed from parent to child, state is local to a component and can change over time.

State is used to manage interactive elements, such as user input, form values, toggles, and counters. When the state changes, React automatically re-renders the component, ensuring that the UI remains in sync with the latest data.

In class components, state is defined using this.state and updated using this.setState(). In functional components, state is managed using the useState Hook.

**Question 3: Why is this.setState() used in class components, and how does it work?**

In class components, this.setState() is used to update the state of a component. Instead of modifying this.state directly, React provides setState() to ensure efficient updates and re-renders.

setState() works asynchronously, meaning React batches multiple updates for performance optimization. It also triggers the render() method, allowing the component to update with the new state values.

By using setState(), developers ensure that React efficiently manages UI changes without unnecessary re-renders, keeping the application smooth and responsive.