Module – 7 React-JS

❖ Component [Function & Class Components]: -

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

Function – Component: -

- In react, Components are the building blocks of any react application.
- A component can manage its own state and it interact with other components through props.
- Components allow developers to split the UI into independent, reusable sections making the code more maintainable and easier to debug.

Difference between functional & class components: -

- Functional component defined as plain JavaScript functions. Class component defined using Es6 classes.
- **Syntax** – **functional** component is **simple** and concise function syntax. And **Class** component requires class keyword with extends React.component.
- **Functional** component uses for state [usestate, useEffect] and lifecycle methods. **Class** component manages state using this.state and life cycle methods like componentDidMount.
- **Functional** component does not provide **this** keyword.
- **Class** component requires this to access props and state.
- **Functional component is Preferred** in modern react development.
- **Class** component is still supported but less commonly used in modern react.

2. How do you pass data to a component using props?

- In React, props are used to pass data from one component to another component, typically from parent to child component.
- Props are read-only meaning that child component cannot modify them.

Steps to pass data using props: -

1. **Pass data from parent to child component: -**

    - When rendering a child component, you can pass data as attributes in the JSX syntax.

2. **Access props in the child component: -**

    - In a functional component, props accessed using this prop.

3. what is the role of render () in class component?

- The purpose of the render() function is to display the specified HTML code inside the specified HTML element.
- In the render method, we can read props and state and return our JSX code to the root component of our app.

❖ State & Props: -

1. What are props in React-JS? How are props different from state?

- Props are a way to pass data from a parent component to a child component in React.
- They are immutable, meaning a child component cannot modify the props directly.
- Props help making components reusable by providing them with dynamic data.

❖ **How are props different from state: -**

- Props passed from parent to child. State is managed within the component.
- Props are immutable (read-only). States are mutable (can be updated).
- Props are used to communicate data. And States are used to manage component data.
- Props are controlled by parent. States are controlled by the component itself.

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

- State is a JavaScript Object or Variable that holds data about a component and determines how component renders and behave.
- It allows React components to manage dynamic data and update the user interface in response to user interactions and other events.

❖ **How manage component-data: -**

1. **Dynamic Rendering: -**
   - State allows components to display data that can change over time, such as user input, fetched data, or UI elements like a dropdown menu.
2. **User Interaction: -**
   - React uses state to handle events like button clicks, form inputs, and hover effects.
3. **Conditional Rendering: -**
   - State enables rendering specific content or components based on current conditions.

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

- In React class components, this.setState() is used to update the component's state.
- It is the only way to modify the state in a class component while ensuring that React knows about the changes and re-renders the component accordingly.

❖ **How does it work: -**

1. Partial State Update: -

- this.setState() merges the provided state update with the current state. It only updates the specified keys, leaving the rest unchanged.

2. Asynchronous Nature: -

- Updates via this.setState() are asynchronous. This means the state might not be updated immediately after calling this.setState().

- To perform an action after the state update, a callback function can be passed as the second argument to this.setState().