Module – 7 React JS

❖ React- Components (Functional & Class Components): -

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

- In React, components are the building blocks of a React application.
- They allow you to split the user interface into independent, reusable pieces that can be managed separately.
- Components are similar to JavaScript functions, but they work independently and return HTML.

❖ **Differences Between Functional and Class Components:-**

- **Functional component** is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX). **Class** component requires you to extend from React. Component and create a render function that returns a React element.

- A class component requires you to extend from React. Component and create a render function that returns a React element. In Class component It must have the render() method returning JSX (which is syntactically similar to HTML)

- Functional components are faster and lighter in performance but class components are slightly slower due to extra overhead compared to functional component

- Functional component managed with hooks like useEffect, useState etc. Class component use explicit lifecycle methods like componentDidMount.

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

- Props are used to pass data from a parent component to a child component in React. Props are read-only and cannot be changed by the child component.
- **Pass data using props: -**
- In the parent component pass data to the child component as an attribute.

- **Example -** function App() {
  return <Greeting name="Siddhraj" />;
  }
- **<u>In the child component: -</u>**
  Example - function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
  }

3. **What is the role of render() in class components?**
   - Render in React JS is a fundamental part of class components. It is used to display the component on the UI returned as HTML or JSX components.
   - 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 used to pass data from a parent component to a child component.
- Props allow components to be **dynamic and reusable** by enabling the parent to customize the child's behavior or appearance.
- **How props different from state: -**
- Props are used to pass data from parent to child components. State is used to manage data within a component.
- Props Immutable Cannot be changed by the child component. State Mutable: Can be updated using setState or useState.
- Props Passed from parent to child (unidirectional). State Exists within a single component.
- Example - Displaying a user's name passed as a prop. State Tracking whether a button is clicked or not.

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

- State is an object that stores data or information about the component.
- It is used to manage and track dynamic data that can change over time, such as user inputs, UI updates, or other variables that affect how the component renders.
- **How data manage from component: -**

1. **Initialize State: -**
   - State is initialized when the component is created.
   - Class Components: Define state in the constructor using this.state.
   - Functional Components: Use the useState hook to define the state.
2. **Update State: -**
   - **Class Components**: this.setState() method.
   - **Functional Components**: The setter function from useState.
3. **Use State in Rendering: -**
   - The component can use state data directly in its rendering logic to determine what to display.

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

- In React class components, this.setState() is the method used to update the component's state.
- Directly modifying this.state is not allowed because React needs to know when the state changes to re-render the component.
- Using this.setState() ensures that the component updates properly and triggers React's re-rendering mechanism.

❖ **How Does it Works :-**

1. **Merges the New State with the Old State:**
   - this.setState() only updates the properties you specify.
   - The rest of the state remains unchanged (it performs a shallow merge).

2. **Asynchronous Updates**:
   - this.setState() may batch multiple state updates for performance optimization.
   - The updated state is not available immediately after calling this.setState().
   - To work with the updated state, use the optional **callback function**.