Module-14 Components, State, Props

- Components (Functional & Class Components)

THEORY EXERCISE:

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

Components in React:

Components are the building blocks of a React application. They help break the UI into smaller, reusable pieces that manage their own structure and behavior.

Difference between Functional and Class Components:

Functional Component	Class Component
Defined using JavaScript functions.	Defined using ES6 classes.
Use **React Hooks** (like `useState`, `useEffect`) to manage state and lifecycle.	Use `this.state` for state and lifecycle methods like `componentDidMount()`.
Shorter and easier to write and understand.	More code and complex syntax.
Cannot use `this` keyword.	Use `this` to access props and state.
Example: `function MyComponent() { return <h1>Hello</h1>; }`	Example: `class MyComponent extends React.Component { render() { return <h1>Hello</h1>; } }`

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

Passing Data with Props in React:
In React, props (short for properties) are used to send data from a parent component to a child component. Props are read-only and cannot be changed by the child component.

How it works:

1. In the parent component, you pass props like attributes in HTML.
2. In the child component, you receive them using props.

Example:

```
// Parent Component
function App() {
  return <Greeting name="Dev" />;
}

// Child Component
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
```

Output:
Hello, Dev!

Here, the name = "Dev" is passed from the parent (App) to the child (Greeting) using props.

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

Role of render() in Class Components:
In React class components, the render() method is used to return the JSX (UI) that should be displayed on the screen.

- It is a required method in every class component.
- It runs automatically when the component mounts or updates.
- It must return a single JSX element.

Example:

```
class MyComponent extends React.Component {
  render() {
    return <h1>Hello, World!</h1>;
  }
}
```

Here, render() returns the UI (<h1>Hello, World!</h1>) for the component.

- Props and State

THEORY EXERCISE:

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

What are props in React.js?
Props (short for properties) are used to pass data from one component to another, usually from a parent to a child component. They are read-only and cannot be changed by the receiving component.

How are props different from state?

- Props are used to pass data, while state is used to manage data within a component.
- Props are set by the parent component; state is managed inside the component.
- Props are read-only; state can be changed using setState or hooks like useState.
- Props make components reusable; state is used for dynamic behavior and interactivity.

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

Concept of state in React:
State is a built-in object in React used to store data that can change over time. It helps make components dynamic and interactive.

How it is used to manage component data:

- In class components, state is managed using this.state and updated with this.setState().
- In functional components, state is managed using the useState hook.

- When state changes, the component re-renders automatically to reflect the new data.

Example using useState:

```
import React, { useState } from 'react';

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

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

In this example, count is the state, and it updates when the button is clicked.

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

Why is this.setState() used in class components:
this.setState() is used to update the state in class components. You cannot change the state directly using this.state, so you must use this.setState() to tell React that the state has changed.

How it works:

- It takes an object or a function and updates the component's state.
- After the state is updated, React automatically re-renders the component to show the new state.

Example:

```
class Counter extends React.Component {
  constructor() {
    super();
    this.state = { count: 0 };
  }

  increaseCount = () => {
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <p>{this.state.count}</p>
        <button onClick={this.increaseCount}>Add</button>
      </div>
    );
  }
}
```

Here, this.setState() updates the count and re-renders the component.