**Components (Functional & Class Components**)

**THEORY EXERCISE**

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

* **Components** are reusable building blocks in React that define UI elements. They can accept inputs (props) and return React elements to be rendered on the screen.
* **Functional Components:** These are simple JavaScript functions that return JSX. They are stateless (before hooks) and easier to write.
* **Class Components:** These are ES6 classes extending React.Component. They have lifecycle methods and can hold internal state.

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

* Props are passed as attributes in JSX. For example: <Greeting name="Alice" /> passes a prop called name with value "Alice" to the Greeting component, which can access it as props.name.

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

* The render() method is required in class components. It returns the JSX that defines the UI to be rendered on the screen.

**LAB EXERCISE**

**Task 1:**  
Create a functional component Greeting that accepts a name as a prop and displays "Hello, [name]!".

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

// Usage example:

// <Greeting name="Alice" />

**Task 2:**  
Create a class component WelcomeMessage that displays "Welcome to React!" and has a render() method.

import React, { Component } from 'react';

class WelcomeMessage extends Component {

render() {

return <h1>Welcome to React!</h1>;

}

}

export default WelcomeMessage;

**Props and State**

**THEORY EXERCISE**

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

* **Props** (short for properties) are read-only inputs passed from a parent component to a child component. They help configure or customize the child component.
* **State** is a component's internal data storage that can change over time, usually in response to user actions or events. Unlike props, state is managed within the component itself and can be updated.

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

* **State** is an object that holds data or information about the component that can change over time.
* React re-renders the component whenever the state changes, allowing dynamic and interactive UIs. State is used to track values like user input, toggles, counters, etc.

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

* this.setState() is used to update the state object in class components.
* It merges the new state with the current state and triggers a re-render of the component.
* Directly modifying this.state won't re-render the component, so setState() must be used.

**LAB EXERCISE**

**Task 1:**  
Create a React component UserCard that accepts name, age, and location as props and displays them in a card format.

function UserCard(props) {

return (

<div style={{border: '1px solid #ccc', padding: '10px', width: '200px'}}>

<h2>{props.name}</h2>

<p>Age: {props.age}</p>

<p>Location: {props.location}</p>

</div>

);

}

**Task 2:**  
Create a Counter component with a button that increments a count value using React state. Display the current count on the screen.

import React, { useState } from 'react';

function Counter() {

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

return (

<div>

<h3>Count: {count}</h3>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

export default Counter;