3. Components (Functional &Class Components)

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

**Components** in React are **reusable, independent pieces** of code that return **HTML (JSX)** to be rendered on the web page. They help you break the UI into smaller, manageable parts.

There are two main types of components in React:

**🔹 1. Functional Components (Modern and Preferred Way)**

* **JavaScript functions** that return JSX.
* Introduced hooks (like useState, useEffect) to manage state and lifecycle.
* Simpler, cleaner, and easier to read.

jsx

CopyEdit

function Welcome(props) {

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

}

**🔹 2. Class Components (Older Way)**

* Use **ES6 classes** that extend React.Component.
* Must define a render() method that returns JSX.
* Use this.state and lifecycle methods like componentDidMount.

jsx

CopyEdit

class Welcome extends React.Component {

render() {

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

}

}

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

In React, **props** (short for **properties**) are used to **send data from a parent component to a child component**.

**🔹 Step-by-Step:**

**1. Define a child component that accepts props:**

jsx

CopyEdit

function Greeting(props) {

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

}

**2. Pass data from the parent component:**

jsx

CopyEdit

function App() {

return <Greeting name="Amit" />;

}

In this example:

* App is the parent component.
* Greeting is the child component.
* "Amit" is passed as a **prop** named name.

**🔄 Props Flow: One-way (Parent ➡️ Child)**

**More Example:**

jsx

CopyEdit

function Profile(props) {

return (

<div>

<p>Name: {props.name}</p>

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

</div>

);

}

function App() {

return <Profile name="Neha" age={28} />;

}

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

In **class components** in React, the render() method is **required** and plays a central role.

**🔹 Purpose of render():**

* It **returns JSX**, which defines **what should be displayed on the screen**.
* It is called **automatically by React** whenever:
  + The component mounts (appears on the screen).
  + The component updates (due to props or state change).

LAB EXERCISE

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>;

}

🔹 Example Usage:

jsx

Copy

Edit

function App() {

return <Greeting name="Amit" />;

}

✅ Output:

Copy

Edit

Hello, Amit!

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

import React from 'react';

class WelcomeMessage extends React.Component {

render() {

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

}

}

🔹 Example Usage:

jsx

Copy

Edit

function App() {

return <WelcomeMessage />;

}

✅ Output:

css

Copy

Edit

Welcome to React!

Props and State

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** in React.

They allow you to:

* **Customize** components
* **Reuse** components with different data
* Keep components **modular and dynamic**

| **Feature** | **Props** | **State** |
| --- | --- | --- |
| Definition | Data passed **from parent to child** | Data **managed within** the component |
| Mutability | **Immutable** (cannot be changed by child) | **Mutable** (can change using setState or useState) |
| Purpose | For **communication** between components | For **local data handling** |
| Access | props.name | this.state.name (class) / name (function) |
| Used In | **Both functional and class components** | **Both functional (with hooks) and class components** |

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

**State** in React is a **built-in object** that stores **dynamic data** inside a component.  
When the state **changes**, the component **automatically re-renders** to reflect the updated data.

**🔹 Why Use State?**

State is used when:

* You want your component to **remember data** (like a counter value).
* The UI should **update based on user interaction** (like form inputs, toggles, etc.).

**🔧 Example in Functional Component (using useState):**

jsx

CopyEdit

import React, { useState } from 'react';

function Counter() {

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

return (

<div>

<p>You clicked {count} times</p>

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

</div>

);

}

* useState(0) initializes the count with 0.
* setCount() updates the state.
* Component re-renders when count changes.

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

In **class components**, this.setState() is used to **update the component's state** and **trigger a re-render** of the UI with the new state values.

**🔹 How It Works:**

1. You **cannot update this.state directly** (e.g., this.state.count = 1 ❌).
2. Instead, you use this.setState() to **safely update** the state and let React **handle re-rendering**.

**🔧 Example:**

jsx

CopyEdit

class Counter extends React.Component {

constructor() {

super();

this.state = {

count: 0

};

}

increaseCount = () => {

this.setState({ count: this.state.count + 1 }); // ✅ correct way

};

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={this.increaseCount}>Increment</button>

</div>

);

}

}

Lab Work

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

**✅ UserCard Component:**

jsx

CopyEdit

import React from 'react';

function UserCard(props) {

const { name, age, location } = props;

return (

<div style={{

border: '1px solid #ccc',

borderRadius: '10px',

padding: '16px',

maxWidth: '250px',

boxShadow: '0 4px 8px rgba(0,0,0,0.1)',

margin: '10px',

fontFamily: 'Arial'

}}>

<h2>{name}</h2>

<p><strong>Age:</strong> {age}</p>

<p><strong>Location:</strong> {location}</p>

</div>

);

}

**🔹 Usage in a Parent Component:**

jsx

CopyEdit

function App() {

return (

<div>

<UserCard name="Amit Sharma" age={30} location="Mumbai" />

</div>

);

}

**✅ Output:**

A styled card displaying:

makefile

CopyEdit

Amit Sharma

Age: 30

Location: Mumbai

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

**✅ Counter Component (Functional):**

jsx

CopyEdit

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0); // Initialize count state to 0

const handleIncrement = () => {

setCount(count + 1); // Update count

};

return (

<div style={{ textAlign: 'center', marginTop: '50px', fontFamily: 'Arial' }}>

<h2>Count: {count}</h2>

<button onClick={handleIncrement}>Increment</button>

</div>

);

}

**Usage in a Parent Component (e.g., App.js):**

jsx

CopyEdit

function App() {

return (

<div>

<Counter />

</div>

);

}

**✅ Output:**

A screen with:

csharp

CopyEdit

Count: 0

[Increment] ← (Clicking this increases the count)