**2. React js JSX**

**Question 1: What is JSX in React.js? Why is it used?**

Ans : JSX stands for JavaScript XML. It is a syntax extension for JavaScript that allows developers to write HTML-like code directly within JavaScript files. This blend of markup and logic is particularly central to React, where it defines the structure and appearance of user interface components.

Why it is used ? → Improves readability, Combines logic and layout:

**Question 2: How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?**

Ans : **Differences**: JSX looks like HTML, but it's not valid JavaScript by default.

#### ****Can you write JavaScript inside JSX?****

**Yes**, you can embed JavaScript expressions inside JSX using **curly braces** {}.

**Example:**

const name = "Jayesh paliya";

const element = <h1>Hello, {name}!</h1>;

Here, {name} is a JavaScript expression embedded within JSX.

**Question 3: Discuss the importance of using curly braces {} in JSX expressions.**

Ans : **Curly braces** {} in JSX are used to embed **JavaScript expressions** within the JSX markup.

#### ****Why they're important:****

* **Expression evaluation**: You can insert variables, function calls, expressions, or even ternary operations.
* **Seamless logic integration**: Allows React to render dynamic content directly within the HTML-like structure.

#### ****Examples:**** // Using a variable

const user = "Jayesh";

<p>Welcome, {user}!</p>

// Using an expression

<p>2 + 2 = {2 + 2}</p>

// Using a function

<p>{getGreeting(user)}</p>

**3. Components(Functional &Class Components)**

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

Ans : components are independent, reusable pieces of code that represent a part of the user interface. They can be thought of as building blocks that can be combined to create complex UIs. The two main types of components in React are functional components and class components:

**Functional Components:** These are simply JavaScript functions that accept props (properties) as an argument and return JSX (JavaScript XML) to describe what should be rendered.

EX: function MyFunctionalComponent(props) {

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

}

**Class Components:**

→ Require extending React.Component.

→ Have a render() method to return JSX.

Ex: class MyClassComponent extends React.Component {

render() {

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

}

}

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

Ans : You pass data to a component using props (short for properties) as attributes in the component tag.

Ex: function Greeting(props) {

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

}

<Greeting name="Jayesh" />

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

Ans : In **class components**, the render() method is **required** and is used to **return JSX**, which describes what should be displayed on the screen.

Ex : class Welcome extends React.Component {

render() {

return <h1>Welcome!</h1>;

}

}

**4. props and state**

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

Ans : **Props (Properties)** in React are used to **pass data from parent to child components**. They are **read-only** and help make components **reusable** and **customizable**.

**Props :**

- props is immutable(readonly)

- pass from the parent.

- configure componant

**state:**

- state is a nutable (can change overtime)

- managed with in the componant.

- handle internal data.

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

Ans : State is a built-in object in React that allows components to create and manage dynamic data. When state changes, the component re-renders automatically.

**Ex**: **Usage in Functional Components :**

import { useState } from "react";

function Example() {

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

return <button onClick={() => setCount(count + 1)}>Count: {count}</button>;

}

**Ex**: **Usage in Class Components**:

class Example extends React.Component {

constructor() {

super();

this.state = { count: 0 };

}

render() {

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

}

}

**Question 2 : Why is this.setState() used in class components, and how does it work?**

Ans : this.setState() is used to **update the component's state** in **class components**.  
It triggers **re-rendering** of the component with the updated state values.

#### 🔹 How it works:

* Takes an object or a function to update part of the state.
* React **merges the update** with the existing state and **re-renders** the component.

Ex: this.setState({ count: this.state.count + 1 });

**4. Handling Events in react**

**Question 2 : How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events?**

Ans : you use methods like addEventListener:

ex: document.getElementById("btn").addEventListener("click", handleClick);

In **React**, you use **JSX attributes** like onClick, directly on elements:

ex: <button onClick={handleClick}>Click Me</button>

React uses **Synthetic Events**, which are **cross-browser wrappers** around the native DOM events.

They have the **same interface** as native events but work consistently across browsers.

Example: event.preventDefault() works the same in all browsers with synthetic events.

**Question 2 : What are some common event handlers in React.js? Provide examples of onClick, onChange, and onSubmit?**

Ans : onClick – when an element is clicked:

Ex: <button onClick={() => alert("Clicked!")}>Click Me</button>

onChange – when an input changes:

EX: <input onChange={(e) => console.log(e.target.value)} />

onSubmit – when a form is submitted:

ex: <form onSubmit={(e) => { e.preventDefault(); alert("Form submitted"); }}>

<button type="submit">Submit</button>

</form>

**Question 3 : Why do you need to bind event handlers in class components?**

Ans : In class components, methods don't automatically bind this. You need to **bind this** so the method correctly refers to the component instance.

Ex: class MyComponent extends React.Component {

constructor() {

super();

this.state = { clicked: false };

this.handleClick = this.handleClick.bind(this); // binding here

}

handleClick() {

this.setState({ clicked: true });

}

render() {

return <button onClick={this.handleClick}>Click</button>;

}

}

**4. Conditional Rendering**

**Question 1 : What is conditional rendering in React? How can you conditionally render elements in a React component?**

Ans : **Conditional rendering** in React means **showing or hiding elements** in the UI based on certain conditions (like user login, age, data availability, etc.).

**Question 2 : How to use if-else, ternary operators, and && in JSX?**

Ans : if else : let message;

if (isLoggedIn) {

message = <p>Welcome back!</p>;

} else {

message = <p>Please log in.</p>;

}

return <div>{message}</div>;

Ternary Operator : {isLoggedIn ? <LogoutButton /> : <LoginButton />}

Logical && : {isLoggedIn && <p>You are logged in.</p>}