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

**JSX in React.js**

JSX (JavaScript XML) is a syntax extension for JavaScript that allows you to write HTML-like code inside JavaScript. It is commonly used in React.js to describe the UI structure in a more readable and declarative way.

**For example, instead of writing:**

**JavaScript:**

const element = React.createElement("h1", {}, "Hello, World!");

**jsx:**

**const element = <h1>Hello, World!</h1>;**

**Why is JSX Used?**

1. Simplifies UI Development – JSX allows developers to write HTML-like code inside JavaScript, making the structure of components more intuitive.
2. Improves Readability – The syntax is more readable compared to using React.createElement().
3. Faster Performance – JSX is compiled into optimized JavaScript code before rendering.
4. Prevents Injection Attacks – React automatically escapes values to prevent cross-site scripting (XSS) attacks.
5. Better Debugging – Provides helpful error messages when syntax issues arise.

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

**How is JSX Different from Regular JavaScript?**

JSX (JavaScript XML) differs from regular JavaScript in the following ways:

1. HTML-like Syntax
   * JSX allows writing elements like <div>, <h1>, and <p> inside JavaScript, which is not possible in regular JavaScript.
   * **Example:**

const element = <h1>Hello, World!</h1>;

**In plain JavaScript, you would write:**

**const element = React.createElement("h1", {}, "Hello, World!");**

1. **Requires a Compiler (Babel)**
   * JSX is not understood by browsers directly and needs to be compiled into JavaScript using Babel before execution.
2. **Uses className Instead of class**
   * Since class is a reserved keyword in JavaScript, JSX uses className for assigning CSS classes.
   * **Example:**

<div className="container">Hello</div>

1. **Self-Closing Tags**
   * JSX enforces self-closing tags for elements like <img />, <br />, and <input />, while in HTML, you may or may not close them explicitly.

**Can You Write JavaScript Inside JSX?**

**Examples:**

1. **Embedding Variables:**

const name = "John";

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

1. **Using Expressions:**

const age = 25;

const element = <p>{age >= 18 ? "Adult" : "Minor"}</p>;

1. **Calling Functions:**

function greet() {

return "Welcome to React!";

}

const element = <h1>{greet()}</h1>;

1. **Inline Styling (Using JavaScript Objects):**

const style = { color: "blue", fontSize: "20px" };

const element = <p style={style}>Styled Text</p>;

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

**Importance of Using Curly Braces {} in JSX Expressions**

In JSX, curly braces {} are used to embed JavaScript expressions inside the markup. They allow us to dynamically insert values, execute functions, and manipulate data within the JSX structure.

**Key Uses of Curly Braces {} in JSX**

**1️. Embedding JavaScript Variables**

Curly braces allow inserting JavaScript variables inside JSX.

const name = "Dharmik";

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

**2️. Using Expressions Inside JSX**

You can place **mathematical operations, string concatenation, and function calls** inside {}.

const price = 100;

const discount = 20;

const finalPrice = price - (price \* discount) / 100;

const element = <p>Final Price: ${finalPrice}</p>;

**3️. Calling Functions Inside JSX**

You can call JavaScript functions directly within JSX.

function greet() {

return "Welcome to React!";

}

const element = <h1>{greet()}</h1>;