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

Answer:

JSX (JavaScript XML) is a syntax extension for JavaScript that allows you to write HTML-like code directly within your JavaScript code. It is not a string nor HTML, but a special syntax that gets transformed into standard JavaScript function calls (React.createElement()) by a compiler like Babel.

**Why it is used:**

Readability and Expressiveness: JSX provides a clear and visual way to describe what the UI should look like. Writing <h1>Hello</h1> is much more intuitive and easier to understand for developers than the equivalent JavaScript:  
React.createElement('h1', null, 'Hello').

Declarative UI: It perfectly complements React's declarative nature. You describe the structure of the UI using a familiar HTML-like syntax, and React handles the underlying logic of how to build and update it.

Combining Markup and Logic: JSX allows you to seamlessly combine rendering logic (JavaScript) and markup (HTML-like tags) in the same place. This colocation often makes components easier to reason about because you can see the UI structure and the logic that affects it side-by-side.

Prevention of Injection Attacks: React DOM escapes any values embedded in JSX before rendering them. This means it's safe to embed user input in JSX, as it will never be executed as raw HTML, helping prevent XSS (cross-site-scripting) attacks.

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

Answer:

**Differences from Regular JavaScript:**

| **Feature** | **JSX** | **Regular JavaScript** |
| --- | --- | --- |
| **Syntax** | HTML-like syntax (e.g., <div></div>). | Standard JavaScript syntax. |
| **Purpose** | Used specifically for declaring UI elements in React. | A general-purpose programming language for logic, calculations, etc. |
| **Class Attribute** | Uses className (e.g., <div className="box">). | Uses class (a reserved keyword in JS). |
| **Self-Closing Tags** | Must be closed (e.g., <img />, <br />). | Not applicable. |

**Can you write JavaScript inside JSX?**

Yes, absolutely. This is one of the most powerful features of JSX. You can embed any valid JavaScript expression inside JSX by wrapping it in curly braces {}.

Examples of JavaScript inside JSX:

Variables: {firstName}

Object properties: {user.name}

Function calls: {formatDate(today)}

Arithmetic: {2 + 2}

Ternary operators (for conditional logic): {isLoggedIn ? 'Logout' : 'Login'}

You are essentially writing JavaScript "inside" the HTML-like structure, blurring the line between the two to create dynamic content.

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

Answer:

Curly braces {} in JSX are fundamentally important because they are the gateway for embedding dynamic content and logic into the markup. They serve a single, critical purpose: to tell the JSX parser, "The text between these braces is not HTML; it's a JavaScript expression. Please evaluate it and render its result here."

Their importance can be summarized in three key areas:

Dynamic Value Injection: They allow you to inject dynamic values from your component's logic (variables, state, props) directly into the UI. Without them, JSX would only be capable of rendering static text.

Without {} (Static): <p>Hello, user</p>

With {} (Dynamic): <p>Hello, {username}</p>

Execution of Logic: They enable you to execute JavaScript expressions inline to compute what should be displayed. This is crucial for conditionals, calculations, and transforming data for display.

**Example (Conditional):**

<h1>{isNew ? 'New Message!' : 'Inbox'}</h1>

**Example (Calculation):**

<p>Total: ${(price \* quantity).toFixed(2)}</p>

Passing Dynamic Props: They are essential for passing dynamic values (not static strings) as props to child components.

Static String Prop: <Button color="blue" /> (The prop color is the string "blue")

Dynamic Prop: <Button isActive={isUserActive} /> (The prop isActive gets the value of the isUserActive variable, which could be true or false).

In essence, curly braces {} are what transform JSX from a static template language into a powerful, dynamic tool for building modern user interfaces. They bridge the gap between the static world of markup and the dynamic world of JavaScript.