**1. Define JSX**

**JSX (JavaScript XML)** is a syntax extension for JavaScript used with React to describe the UI.  
It looks like HTML but gets compiled into React.createElement() calls by tools like Babel.

**Example:**

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

JSX is not required in React, but it makes code more readable and expressive.

**2. Explain about ECMAScript (ES)**

**ECMAScript (ES)** is the official standard for JavaScript, maintained by ECMA International.

* Modern JavaScript features like let, const, arrow functions, classes, modules, and more come from **ES6 (ECMAScript 2015)** and beyond.
* React heavily uses ES6+ features for cleaner, modular code.

**3. Explain React.createElement()**

React.createElement() is the function behind JSX. JSX is syntactic sugar for this method.

**Syntax:**

React.createElement(type, props, children)

**Example:**

const element = React.createElement('h1', { className: 'title' }, 'Hello');

This creates:

html

<h1 class="title">Hello</h1>

**4. Explain How to Create React Nodes with JSX**

In JSX, you create a **React node** just like writing HTML inside JavaScript.

**Example:**

const header = <h1>Welcome to React</h1>;

You can also create component nodes:

function Welcome() { return <h2>Hello User!</h2>; } const app = <Welcome />;

**5. Define How to Render JSX to the DOM**

To render JSX to the DOM, use the ReactDOM.createRoot().render() method:

import React from 'react';

import ReactDOM from 'react-dom/client';

const element = <h1>Hello React</h1>;

const root = ReactDOM.createRoot(document.getElementById('root')); root.render(element);

This tells React to mount the JSX inside the DOM node with id="root".

**6. Explain How to Use JavaScript Expressions in JSX**

You can use JavaScript **expressions** (not statements) inside JSX using curly braces {}.

**Examples:**

const name = "Durga"; const element = <h1>Hello, {name}</h1>; const sum = <p>{5 + 10}</p>;

**Allowed:** expressions, function calls, variables  
**Not Allowed:** if, for, etc. (use ternary or .map())

**7. Explain How to Use Inline CSS in JSX**

In JSX, inline styles are passed as an **object** with camelCase property names:

**Example:**

const styleObject = { color: 'blue', fontSize: '24px', backgroundColor: 'lightgray' };

const element = <h1 style={styleObject}>Styled Text</h1>;

Or inline directly:

<h2 style={{ color: 'green', padding: '10px' }}>Inline Style</h2>