## Objective Answers

### 1. Define JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used in React to describe how the UI should appear. It allows developers to write HTML-like code within JavaScript, making component structures more readable and declarative. Under the hood, JSX is transpiled into React.createElement() calls which create virtual DOM elements. JSX enhances code clarity by resembling HTML while still having the full power of JavaScript, such as embedding expressions, functions, or variables inside {}. This seamless blend of markup and logic makes UI development with React highly efficient and expressive.

### 2. Explain About ECMAScript

ECMAScript (often abbreviated as ES) is the scripting language standard upon which JavaScript is based. Managed by ECMA International, it ensures consistency across different implementations of JavaScript. ES6 (ECMAScript 2015) was a significant update that introduced features like let, const, arrow functions, classes, modules, and promises. These additions made JavaScript more powerful, modular, and easier to maintain, especially for large-scale web applications. React development heavily relies on ES6+ features to build modern, component-based UIs.

### 3. Explain React.createElement()

React.createElement() is a core API method in React that constructs React elements manually. It takes three arguments: the element type (e.g., 'div', 'h1'), optional properties (as an object), and children (which can be strings or other React elements). For example:

React.createElement('h1', { className: 'title' }, 'Hello World');

This creates a virtual DOM node which React uses to render or update the actual DOM. Although JSX is preferred for convenience, it is essentially compiled into React.createElement() under the hood.

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

React nodes using JSX can be created by writing HTML-like tags inside a JavaScript file. JSX supports self-closing tags, attributes, and nesting just like HTML. For example:

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

This creates a React node of type h1. Components can return such nodes to define what should appear on the screen. JSX also supports conditionals, loops, and expressions inside curly braces, which enhances its flexibility.

### 5. Define How to Render JSX to DOM

JSX elements or React components are rendered to the DOM using the ReactDOM.render() function (in older versions) or the root.render() method (in React 18+). The function takes a React element and a DOM node as arguments:

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

root.render(<App />);

This tells React to render the <App /> component into the HTML element with the ID root. React updates only the parts of the DOM that change using its virtual DOM diffing algorithm, making rendering efficient.

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

JavaScript expressions can be embedded inside JSX using curly braces {}. These expressions can include variables, functions, ternary operators, or any valid expression:

const name = "React";

return <h1>Welcome to {name}</h1>;

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

However, JSX does not support full statements like if, for, or while directly; instead, use expressions such as ternaries or map functions.

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

Inline styles in JSX are defined using JavaScript objects, where CSS property names are written in camelCase instead of kebab-case. The style attribute receives this object:

const headingStyle = {

color: 'blue',

fontSize: '24px',

backgroundColor: 'lightgray'

};

return <h1 style={headingStyle}>Office Rentals</h1>;

const style = {

color: rent > 60000 ? 'green' : 'red'

};

Inline styles are useful for quick, component-scoped styling, especially for dynamic values.