# JSX and React Fundamentals

## Define JSX

JSX stands for JavaScript XML.

It allows writing HTML elements in JavaScript and placing them in the DOM without using methods like createElement().

JSX makes code easier to write and understand for building React components.

Example:  
const element = <h1>Hello, world!</h1>;

## Explain ECMA Script

ECMAScript (often abbreviated as ES) is the standardized scripting language specification on which JavaScript is based.

Each new version of ECMAScript adds new features to JavaScript.

ES6 (ECMAScript 2015) introduced major features like let/const, arrow functions, classes, modules, promises, etc.

## Explain React.createElement()

`React.createElement()` is a method used to create a virtual DOM element.

It is the foundation of JSX behind the scenes.

Syntax:  
React.createElement(type, [props], [...children])

Example:  
React.createElement('h1', null, 'Hello World')

## Explain how to create React nodes with JSX

React nodes can be created using JSX syntax.

JSX allows you to describe what the UI should look like in a declarative way.

Example:  
const heading = <h1>Welcome to React</h1>;

## Define how to render JSX to DOM

ReactDOM.render() is used to render JSX to the actual DOM.

It takes a JSX element and a DOM node where it should be mounted.

Example:  
ReactDOM.render(<App />, document.getElementById('root'));

## Explain how to use JavaScript expressions in JSX

You can use JavaScript expressions in JSX by wrapping them in curly braces `{}`.

This allows dynamic values to be embedded into the UI.

Example:  
const name = 'Alice';  
const greeting = <h1>Hello, {name}!</h1>;

## Explain how to use inline CSS in JSX

Inline CSS can be applied using the `style` attribute in JSX.

The value must be a JavaScript object with camelCased property names.

Example:  
const headingStyle = {  
 color: 'blue',  
 fontSize: '20px'  
};  
const element = <h1 style={headingStyle}>Styled Text</h1>;