# 📘 React and ECMAScript Essentials

### 🔹 Define JSX

* JSX stands for **JavaScript XML**.
* It’s a syntax extension that allows writing HTML-like code within JavaScript.
* JSX improves readability and is transformed into React.createElement() calls.

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

### 🔹 What is ECMAScript?

* ECMAScript (often abbreviated as ES) is the **standardized scripting language** upon which JavaScript is based.
* ES versions (like ES6, ES7...) introduce new features and syntax.
* ES6 (ECMAScript 2015) was a major update with let/const, arrow functions, classes, modules, etc.

### 🔹 React.createElement()

* Core method that creates a React element.
* JSX is compiled into React.createElement() under the hood.

React.createElement("h1", { className: "greeting" }, "Hello, Sahil!");

Produces:

<h1 className="greeting">Hello, Sahil!</h1>

### 🔹 Creating React Nodes with JSX

* JSX allows nesting of elements and usage of components.

const App = () => (

<div>

<h1>Welcome</h1>

<p>This is your React app.</p>

</div>

);

### 🔹 Rendering JSX to the DOM

* Use ReactDOM.render() or the new createRoot() method with React 18+.

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

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

root.render(<App />);

### 🔹 Using JavaScript Expressions in JSX

* Expressions must be wrapped in {} inside JSX.
* You can embed variables, functions, and calculations.

const name = "Sahil";

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

### 🔹 Using Inline CSS in JSX

* Inline styles use the style attribute and require a JS object.
* CSS properties must be camelCased.

const styleObj = {

color: "blue",

fontSize: "20px"

};

const heading = <h1 style={styleObj}>Styled with Inline CSS</h1>;