## **Objectives**

* Define JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows developers to write HTML-like code inside JavaScript. Instead of using regular JavaScript functions to create elements, JSX provides a more readable and concise way to build UI components. JSX makes the code look cleaner and easier to understand. Behind the scenes, JSX is converted into React.createElement() calls by Babel before being rendered in the browser.

* Explain about ECMA Script

ECMAScript is the official standard that defines the rules, syntax, and features of the JavaScript language. JavaScript is based on ECMAScript. Over the years, new versions of ECMAScript have introduced useful features to simplify coding. For example:

* **ES5** introduced features like Array.forEach().
* **ES6 (also called ES2015)** added let, const, arrow functions, classes, template literals, promises, and modules.

These features improve the performance, readability, and maintainability of the code.

* Explain React.createElement()

React.createElement() is a core method in React used to create virtual DOM elements. It takes three arguments:

1. The type of element (e.g., 'div', 'h1', or a component),
2. The properties or attributes of the element (like id, className, etc.),
3. The children of the element (text, other elements, or components).

When you write JSX like <h1>Hello</h1>, it is actually converted to:  
React.createElement('h1', null, 'Hello').

This is what React uses to build the component tree before rendering it on the screen.

* Explain how to create React nodes with JSX

In React, nodes (or elements) represent the building blocks of the user interface. With JSX, we can easily create these elements using HTML-like syntax. For example:

* <h1>Hello, world!</h1> creates a heading node.
* <p>This is a paragraph.</p> creates a paragraph node.

JSX allows nesting elements, passing props, and writing cleaner UI code.

* Define how to render JSX to DOM

To display JSX on the web page, we use ReactDOM.render(). This function takes two arguments:

1. The JSX element or component to display,
2. The DOM node (usually a <div> with id="root") where the React component should be rendered.

This tells React what to show and where to show it on the webpage.

Example:

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

* Explain how to use JavaScript expressions in JSX

JavaScript expressions (like variables, math operations, or function calls) can be used inside JSX using curly braces {}. This is useful for displaying dynamic content. We can:

* Show a variable: {name}
* Do a calculation: {a + b}
* Call a function: {greetUser()}

This makes JSX powerful, as you can mix static HTML with dynamic JavaScript content easily.

* Explain how to use inline CSS in JSX

In JSX, inline styling is applied using the style attribute, but the value is an object. Instead of using traditional CSS syntax, we use camelCase property names and string values.

Example:

const myStyle = {

color: 'blue',

backgroundColor: 'lightgray',

fontSize: '18px'

};

Then applied as:

<h1 style={myStyle}>Styled Text</h1>

This helps in applying quick styles directly from the component.

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.

Create an element to display the heading of the page.

Attribute to display the image of the office space

Create an object of office to display the details like Name, Rent and Address.

Create a list of Object and loop through the office space item to display more data.

To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.

Output:



**Hint:**





**HandsOn:**





