**React 10**

### ****1. Define JSX****

**JSX (JavaScript XML)** is a **syntax extension** for JavaScript used with React.  
It lets you **write HTML-like code inside JavaScript**, which gets transpiled to React.createElement() calls behind the scenes.

#### Example:

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

JSX makes code more readable and easier to write React elements.

### ****2. Explain ECMA Script****

**ECMAScript (ES)** is the **standardized scripting language specification** that JavaScript follows.  
Versions like **ES5**, **ES6 (ES2015)**, etc., introduced new language features.

#### Example of ES6 Features:

let, const

Arrow functions

Classes

Template literals

Modules (import, export)

Promises

Destructuring

React heavily benefits from ES6 features for modern, clean code.

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

This is the **core method** used to create React elements (nodes). JSX compiles to React.createElement.

#### Syntax:

#### React.createElement(type, props, ...children);

#### Example:

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

This is equivalent to:

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

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

JSX makes it easy to create React nodes that represent DOM elements or components.

#### Example:

const node = <h1>Welcome to React</h1>;const list = <ul><li>Item 1</li><li>Item 2</li></ul>;

we can also embed components:

const MyComponent = () => <p>This is a custom component</p>;const app = <MyComponent />;

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

Use ReactDOM.render() or createRoot() (in React 18+) to render JSX elements into a real DOM node.

#### React 18+ (Preferred):

import React from 'react';import ReactDOM from 'react-dom/client';

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

root.render(<h1>Hello, React!</h1>);

#### Older React:

ReactDOM.render(<h1>Hello</h1>, document.getElementById('root'));

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

we can embed **any JavaScript expression** inside JSX using curly braces {}.

#### Example:

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

const sum = <p>{5 + 3}</p>; // 8

const isLoggedIn = true;const message = <h2>{isLoggedIn ? "Welcome!" : "Please login"}</h2>;

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

In JSX, inline styles are written as **JavaScript objects**, not as strings.

#### Syntax:

const style = {

color: 'blue',

fontSize: '24px',

backgroundColor: 'lightgray'

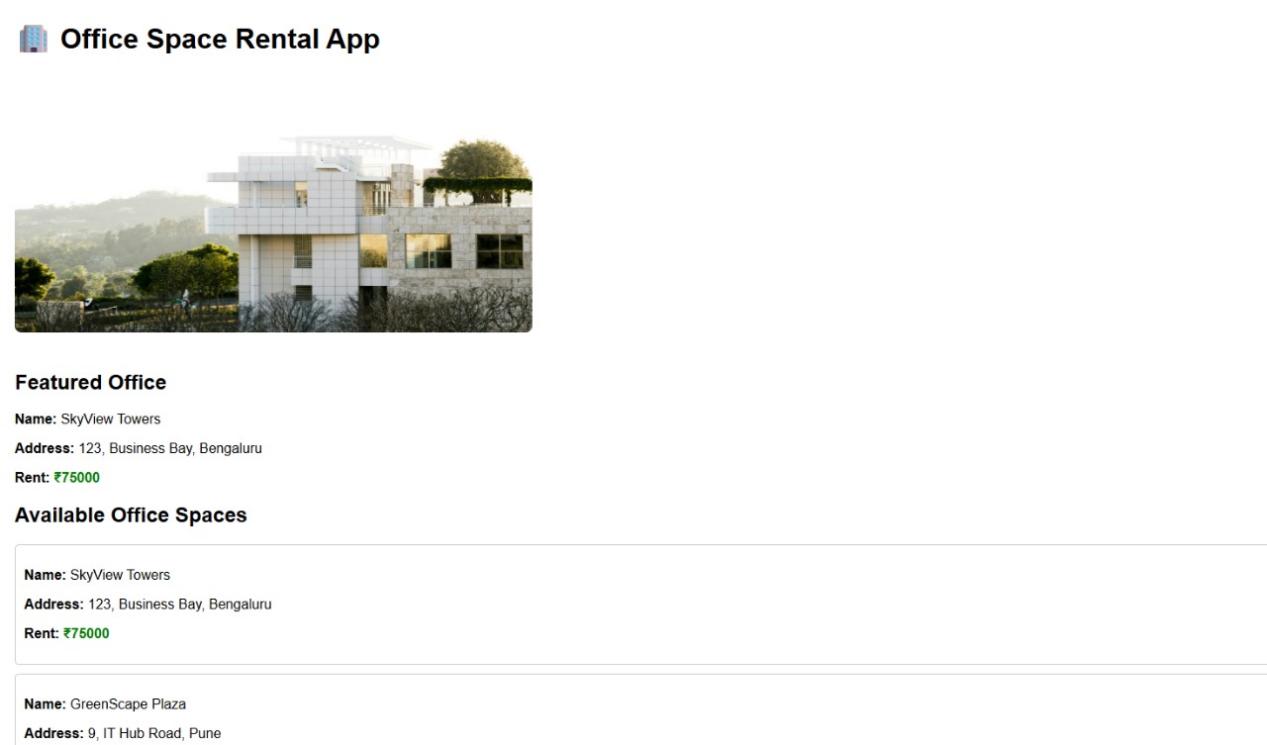
};

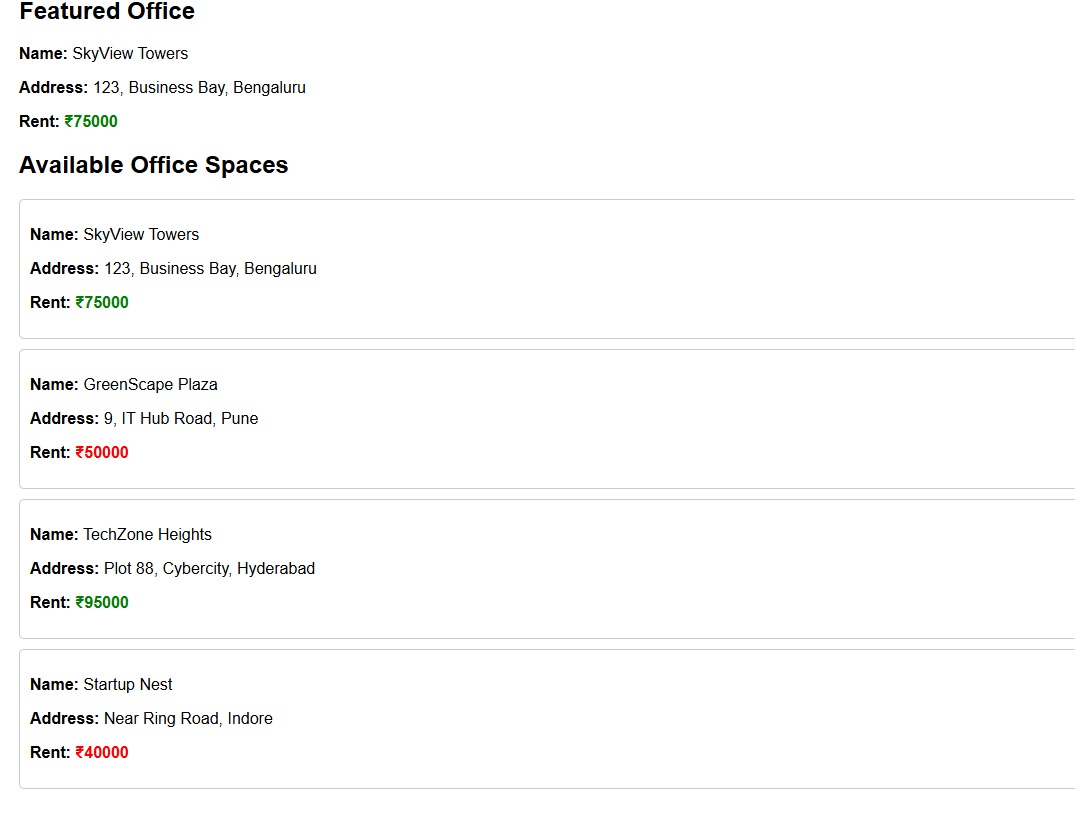
const element = <h1 style={style}>Styled Heading</h1>;

we can also apply directly:

<h1 style={{ color: 'red', fontWeight: 'bold' }}>Hello</h1>

**Output for the lab:**

****

****