**Objectives**

**1. JSX (JavaScript XML)**

JSX is a syntax extension for JavaScript used in React to describe UI elements.

* It allows writing HTML-like code directly inside JavaScript.
* Though it looks like HTML, it’s compiled into React.createElement() calls.

**Example:**

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

**2. ECMA Script**

ECMAScript is the standardized specification of JavaScript defined by ECMA International.

* It outlines core features of the language (syntax, types, objects, etc.).
* Versions include ES5, ES6 (major update), ES7, and beyond.
* ES6 introduced features like let, const, arrow functions, classes, modules, promises, and more.

**3. React.createElement()**

This function creates a React element manually without using JSX.

React.createElement(

'div',

{ className: 'container' },

'Hello World!'

);

* First parameter: tag name (div)
* Second: props object
* Third: children (text or nested elements)

**4. Creating React Nodes with JSX**

JSX lets you declare React nodes in a more readable, declarative way.

**Example:**

const element = <button className="btn">Click Me</button>;

* This will internally convert to:  
  React.createElement("button", { className: "btn" }, "Click Me")

**5. Rendering JSX to the DOM**

You use ReactDOM.render() to inject JSX into the browser’s DOM.

import ReactDOM from 'react-dom';

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

ReactDOM.render(element, document.getElementById('root'));

* The root is typically a div in your HTML file.

**6. Using JavaScript Expressions in JSX**

You can embed expressions by wrapping them in {}.

**Example:**

const user = 'Sangamesh';

const element = <h2>Welcome, {user}!</h2>;

* You can use any valid JS expression: functions, math, conditionals.

**7. Using Inline CSS in JSX**

React uses camelCase properties and JS objects for inline styles.

**Example:**

const style = {

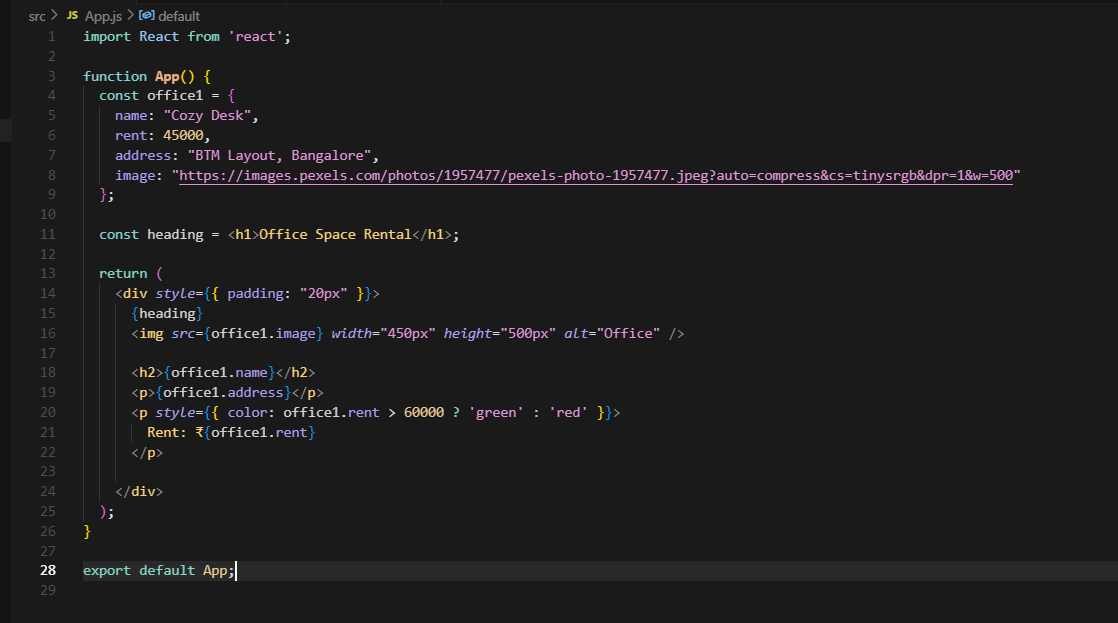
backgroundColor: 'lightblue',

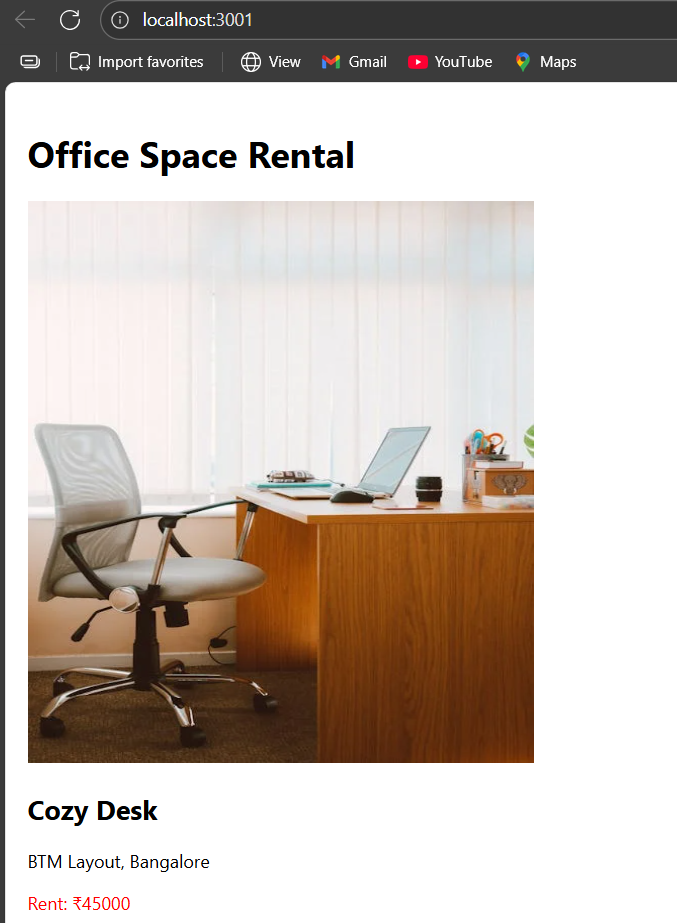
fontSize: '20px'

};

const element = <div style={style}>Styled Div</div>;

* Avoid string-based CSS like "background-color" or "font-size" — use camelCase and object notation.



****