**Define JSX**

**JSX** stands for **JavaScript XML**.  
It is a syntax extension for JavaScript used in React to describe what the UI should look like.  
JSX looks like HTML but you write it inside JavaScript code.

Example:

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

React uses JSX to create React elements under the hood. JSX makes it easier to write and visualize UI components.

**Explain about ECMA Script**

ECMAScript (often abbreviated as ES) is the standardized scripting language specification that JavaScript follows.  
It defines the rules, syntax, and features for JavaScript.  
New versions are released every year, e.g., ES5, ES6 (also called ES2015), ES7, etc.  
ES6 introduced many modern features like arrow functions, classes, modules, let and const, promises, and more.

**Explain React.createElement()**

React.createElement() is a method React uses internally to create **React elements**.

It takes three arguments:

* The HTML tag or React component
* Props (attributes)
* Children (nested elements or text)

Example:

const element = React.createElement(

'h1',

{ className: 'greeting' },

'Hello, world!'

);

This creates a React element that represents <h1 class="greeting">Hello, world!</h1>.

**Explain how to create React nodes with JSX**

When you write JSX, like:

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

Babel (the JSX compiler) transforms this into:

const element = React.createElement('h1', null, 'Hello, world!');

So, writing JSX is syntactic sugar that compiles down to React.createElement() calls which produce React nodes.

**Define how to render JSX to DOM**

To render JSX elements to the actual webpage (DOM), React provides the ReactDOM.render() method.

Example:

import React from 'react';

import ReactDOM from 'react-dom';

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

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

This tells React to display the JSX element inside the DOM node with id root.

**Explain how to use JavaScript expressions in JSX**

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

Example:

const name = "Virat Kohli";

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

You can also use expressions like math, function calls, ternary operators:

const number = 10;

const element = <p>{number > 5 ? 'Greater than 5' : '5 or less'}</p>;

**Explain how to use inline CSS in JSX**

In JSX, **inline styles** are written as an object using **camelCase** for CSS properties.

Example:

const divStyle = {

color: 'blue',

backgroundColor: 'lightgray',

fontSize: '20px',

padding: '10px'

};

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

Or inline directly:

const element = <div style={{ color: 'red', fontWeight: 'bold' }}>Hello</div>;

Note: CSS properties use camelCase (backgroundColor instead of background-color).

**Index.js :-**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

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

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

reportWebVitals();

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

**App.js :-**

import React from 'react';

function App() {

  // Single office object

  const office = {

    name: "Central Business Tower",

    rent: 55000,

    address: "123 Main St, Metropolis"

  };

  // List of offices

  const offices = [

    { id: 1, name: "Central Business Tower", rent: 55000, address: "123 Main St, Metropolis" },

    { id: 2, name: "Downtown Office Plaza", rent: 72000, address: "456 Market Ave, Metropolis" },

    { id: 3, name: "Riverfront Workspace", rent: 48000, address: "789 River Rd, Metropolis" },

    { id: 4, name: "Skyline Center", rent: 80000, address: "101 Skyline Dr, Metropolis" }

  ];

  // Inline style function for rent color

  const rentStyle = (rent) => ({

    color: rent < 60000 ? 'red' : 'green',

    fontWeight: 'bold'

  });

  return (

    <div style={{ padding: '20px', fontFamily: 'Arial, sans-serif' }}>

      {/\* Heading \*/}

      <h1>Office Space Rental</h1>

      {/\* Image of office space \*/}

      <img

        src="https://images.unsplash.com/photo-1504384308090-c894fdcc538d?auto=format&fit=crop&w=800&q=60"

        alt="Office Space"

        style={{ width: '400px', borderRadius: '8px', marginBottom: '20px' }}

      />

      {/\* Display single office details \*/}

      <h2>Featured Office</h2>

      <p><strong>Name:</strong> {office.name}</p>

      <p><strong>Rent:</strong> <span style={rentStyle(office.rent)}>{office.rent}</span></p>

      <p><strong>Address:</strong> {office.address}</p>

      {/\* List of offices \*/}

      <h2>Available Offices</h2>

      <ul>

        {offices.map(({ id, name, rent, address }) => (

          <li key={id} style={{ marginBottom: '15px' }}>

            <p><strong>Name:</strong> {name}</p>

            <p>

              <strong>Rent:</strong> <span style={rentStyle(rent)}>{rent}</span>

            </p>

            <p><strong>Address:</strong> {address}</p>

          </li>

        ))}

      </ul>

    </div>

  );

}

export default App;

**Output :-**



