### Exercise 10

### Define JSX

**JSX** stands for JavaScript XML. It is a syntax extension for JavaScript that allows you to write HTML-like code within your JavaScript files. JSX is not a new language but a syntactic sugar for React.createElement(). It makes the process of creating React elements much more intuitive and readable, as it closely resembles the final HTML structure.

### Explain about ECMA Script

**ECMAScript** (ES) is a standard for scripting languages, with JavaScript being the most popular implementation. It is standardized by Ecma International in the document ECMA-262. ECMAScript defines the syntax, semantics, and core features of the language. JavaScript is an implementation or "dialect" of ECMAScript. Each new version of ECMAScript (e.g., ES6, ES7, ES2020) introduces new features and improvements to the language, which are then adopted by browser engines and other JavaScript environments.

### Explain React.createElement()

React.createElement() is the fundamental function used to create React elements. When you write JSX, it is transpiled (converted) by a tool like Babel into a series of React.createElement() function calls. This function takes three primary arguments:

1. **Type:** The type of the element. This can be a string for an HTML tag (e.g., 'h1', 'div') or a reference to a React component.
2. **Props:** An object containing the properties or attributes of the element (e.g., { className: 'title' }).
3. **Children:** The children of the element, which can be other React elements, strings, or numbers.

**Example without JSX:**

JavaScript

const element = React.createElement('h1', { className: 'greeting' }, 'Hello, world!');

### Explain how to create React nodes with JSX

JSX allows you to create React elements (which represent nodes in the virtual DOM) using a syntax that looks like HTML. You can simply write the tag with its attributes and children directly in your JavaScript code.

**Example with JSX:**

JavaScript

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

This JSX code is equivalent to the React.createElement() example above. The transpiler handles the conversion for you, making the code cleaner and easier to read.

### Define how to render JSX to DOM

To display your React elements in a web browser, you need to render them into the Document Object Model (DOM). This is typically done using the ReactDOM library. In modern React (version 18+), you first create a "root" to manage the DOM, and then you call the render() method on that root.

**Example:**

JavaScript

import ReactDOM from 'react-dom/client';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
const element = <h1>Hello, world!</h1>;  
  
root.render(element);

This code finds the HTML element with the ID root and renders your h1 element inside it, replacing any existing content.

### Explain how to use JavaScript expressions in JSX

You can embed any valid JavaScript expression inside JSX by enclosing it in curly braces {}. This is how you make your UI dynamic.

**Examples:**

* **Variables:**  
  JavaScript  
  const name = 'John Doe';  
  const element = <h1>Hello, {name}</h1>;
* **Function calls:**  
  JavaScript  
  function formatName(user) {  
    return user.firstName + ' ' + user.lastName;  
  }  
    
  const user = { firstName: 'Jane', lastName: 'Doe' };  
  const element = <h1>Hello, {formatName(user)}</h1>;
* **Arrays:**  
  JavaScript  
  const numbers = [1, 2, 3];  
  const listItems = numbers.map((number) => <li>{number}</li>);  
  const element = <ul>{listItems}</ul>;

### Explain how to use inline CSS in JSX

In JSX, you apply inline styles by passing a JavaScript object to the style attribute. The keys of the object are the CSS property names written in **camelCase**, and the values are strings representing the style values.

**Example:**

JavaScript

const myStyle = {  
  color: 'blue',  
  fontSize: '20px',  
  backgroundColor: 'lightgray'  
};  
  
const element = <h1 style={myStyle}>Hello, world!</h1>;

You can also pass the object directly into the style attribute using double curly braces:

JavaScript

const element = <h1 style={{ color: 'blue', fontSize: '20px' }}>Hello, world!</h1>;

This is because the first set of curly braces indicates that you are entering a JavaScript expression, and the second set creates the JavaScript object itself.

**App.js**

import React from 'react';

import './App.css';

import officeImage from './office.jpg';

function App() {

  const singleOffice = {

    Name: 'The Hub',

    Rent: 75000,

    Address: '123 Business Avenue, City'

  };

  const officeSpaces = [

    {

      id: 1,

      name: 'Creative Loft',

      rent: 55000,

      address: '456 Creative St, City'

    },

    {

      id: 2,

      name: 'Executive Suite',

      rent: 85000,

      address: '789 Corporate Blvd, City'

    },

    {

      id: 3,

      name: 'Startup Garage',

      rent: 45000,

      address: '101 Innovation Alley, City'

    }

  ];

  return (

    <div className="App">

      <header className="App-header">

        <h1>Office Space Rentals</h1>

      </header>

      <main>

        <div className="office-card">

          <img src={officeImage} alt="Office Space" className="office-image" />

          <div className="office-details">

            <h2>{singleOffice.Name}</h2>

            <p className={singleOffice.Rent < 60000 ? 'rent-red' : 'rent-green'}>

              Rent: ₹{singleOffice.Rent.toLocaleString()}

            </p>

            <p>Address: {singleOffice.Address}</p>

          </div>

        </div>

        <hr />

        {officeSpaces.map(office => (

          <div key={office.id} className="office-card">

            <img src={officeImage} alt={`Office Space ${office.name}`} className="office-image" />

            <div className="office-details">

              <h2>{office.name}</h2>

              <p className={office.rent < 60000 ? 'rent-red' : 'rent-green'}>

                Rent: ₹{office.rent.toLocaleString()}

              </p>

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

            </div>

          </div>

        ))}

      </main>

    </div>

  );

}

export default App;

**App.css**

.App {

  font-family: sans-serif;

  text-align: center;

  padding: 20px;

}

.App-header {

  background-color: #282c34;

  padding: 20px;

  color: white;

  margin-bottom: 20px;

}

main {

  display: flex;

  flex-direction: column;

  align-items: center;

}

.office-card {

  border: 1px solid #ccc;

  border-radius: 8px;

  padding: 20px;

  margin: 10px;

  width: 80%;

  max-width: 500px;

  text-align: left;

  display: flex;

  gap: 20px;

  background-color: #f9f9f9;

}

.office-image {

  width: 150px;

  height: 100px;

  object-fit: cover;

  border-radius: 4px;

}

.office-details {

  flex-grow: 1;

}

.rent-red {

  color: red;

  font-weight: bold;

}

.rent-green {

  color: green;

  font-weight: bold;

}

hr {

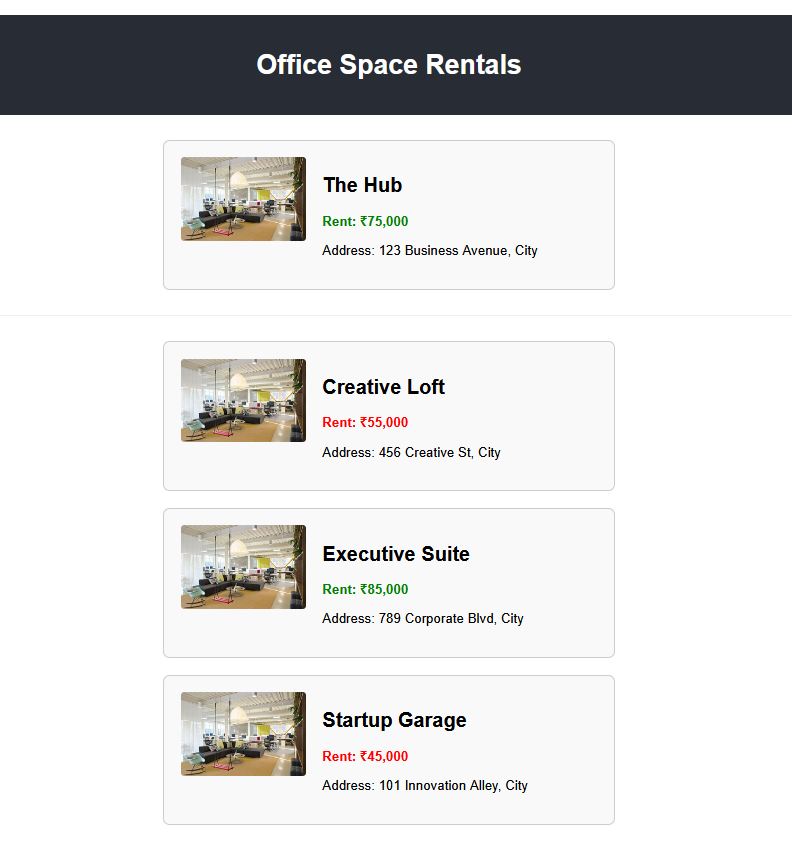
  width: 80%;

  margin: 20px 0;

  border: 0;

  border-top: 1px solid #eee;

}

**Output:**