**Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows developers to write HTML-like code inside JavaScript, making the UI structure more readable and easier to manage. Under the hood, JSX is converted into standard JavaScript using React.createElement().

**Explain about ECMA Script**

ECMAScript (ES) is the standard specification on which JavaScript is based. It defines the rules, syntax, and features for the language. New versions, such as ES6 and later, introduced features like let/const, arrow functions, classes, modules, and more, making JavaScript more powerful and developer-friendly.

**Explain React.createElement()**

React.createElement() is the core function React uses to create elements (nodes) without JSX. It takes three arguments:

* The type of element (e.g., 'div', 'h1'),
* The properties or attributes (e.g., { className: 'title' }),
* The child elements or text.

Example:

javascript

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

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

In JSX, you can create React nodes simply by writing HTML-like syntax inside JavaScript:

jsx

const heading = <h1>Hello World</h1>;

This is easier and cleaner than using React.createElement() and is automatically converted to it during compilation.

**Define how to render JSX to DOM**

To display JSX in the browser, React uses ReactDOM.createRoot() (or ReactDOM.render() in older versions) to attach the JSX to a DOM element, usually with an id="root":

jsx

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

root.render(<App />);

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

JSX allows you to embed JavaScript expressions inside curly braces {}. This can be used to display variables, function results, or calculations:

jsx

const name = "John";

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

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

In JSX, inline CSS is written using a JavaScript object with camelCase property names:

Jsx

const headingStyle = { color: 'blue', fontSize: '24px' };

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

You can also directly pass the object inside the style attribute:

jsx

<h1 style={{ color: 'red' }}>Inline Styled Text</h1>

**App.js**

import React from "react";

function App() {

// Single office object (example)

const office = {

name: "Tech Park - Tower A",

rent: 55000,

address: "123, MG Road, Bangalore"

};

// Multiple office list

const offices = [

{ name: "Tech Park - Tower A", rent: 55000, address: "MG Road, Bangalore" },

{ name: "Innovation Hub", rent: 75000, address: "Whitefield, Bangalore" },

{ name: "Startup Space", rent: 45000, address: "Koramangala, Bangalore" }

];

return (

<div style={styles.container}>

{/\* Heading \*/}

<h1>Office Space Rental App</h1>

{/\* Image \*/}

<img

src="https://via.placeholder.com/400x200.png?text=Office+Space"

alt="Office Space"

style={styles.image}

/>

{/\* Single Office Details \*/}

<h2>Office Details</h2>

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

<p

style={{

color: office.rent < 60000 ? "red" : "green",

fontWeight: "bold"

}}

>

Rent: ₹{office.rent}

</p>

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

<hr />

{/\* Multiple Office List \*/}

<h2>Available Office Spaces</h2>

<ul style={styles.list}>

{offices.map((item, index) => (

<li key={index} style={styles.item}>

<p>Name: {item.name}</p>

<p

style={{

color: item.rent < 60000 ? "red" : "green",

fontWeight: "bold"

}}

>

Rent: ₹{item.rent}

</p>

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

</li>

))}

</ul>

</div>

);

}

const styles = {

container: {

textAlign: "center",

fontFamily: "Arial",

padding: "20px"

},

image: {

width: "400px",

height: "200px",

margin: "20px 0"

},

list: {

listStyle: "none",

padding: 0

},

item: {

border: "1px solid #ccc",

padding: "10px",

margin: "10px auto",

width: "300px",

borderRadius: "8px"

}

};

export default App;

**Index.js**

import React from "react";

import ReactDOM from "react-dom/client";

import "./index.css";

import App from "./App";

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

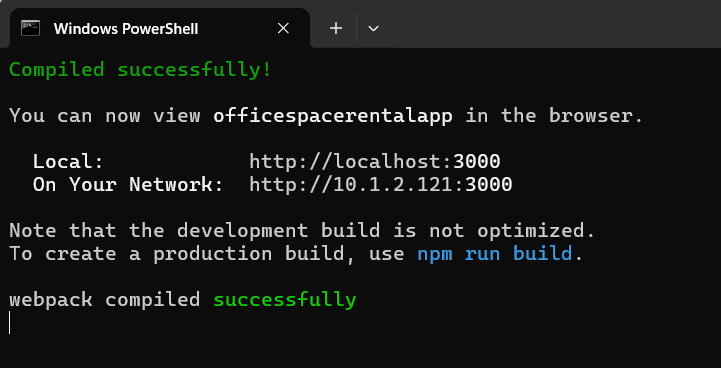
root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);



**OUTPUT ON THE BROWSER**

A computer screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.