React and JavaScript Interview Questions

Define JSX

JSX (JavaScript XML) is a syntax extension for JavaScript commonly used with React. It allows developers to write HTML-like code within JavaScript. JSX provides a more readable and declarative way to create React elements and UI components. Under the hood, JSX is transpiled into React.createElement() calls by tools like Babel.  
  
Example:  
const element = <h1>Hello, World!</h1>;  
This JSX code is transformed into:  
const element = React.createElement("h1", null, "Hello, World!");

Explain about ECMA Script

ECMAScript (often abbreviated as ES) is the standardized scripting language specification upon which JavaScript is based. It defines the core features of the language like variables, loops, data types, etc.  
  
ES5 introduced stricter parsing ('use strict'), Array methods.  
ES6 (ECMAScript 2015) introduced significant improvements: let, const, arrow functions, classes, promises, template literals, Map, Set, modules, etc.  
Newer versions (ES7 onward) continued to enhance functionality and syntax.  
  
ECMAScript keeps evolving, enabling developers to write more concise, readable, and efficient JavaScript code.

Explain React.createElement()

React.createElement() is a core method in React used to create React elements.  
  
Syntax:  
React.createElement(type, props, ...children)  
  
- type: A string (e.g., 'div') or React component.  
- props: An object containing element properties.  
- children: The content (can be text, elements, or components).  
  
Example:  
const element = React.createElement('h1', { className: 'greeting' }, 'Hello World');  
  
This creates a virtual DOM element like: <h1 class="greeting">Hello World</h1>  
JSX is a syntactic sugar over React.createElement().

Explain how to create React nodes with JSX

React nodes can be created using JSX by writing HTML-like tags directly in JavaScript. Each JSX element is compiled to a React.createElement() call.  
  
Example:  
const greeting = <h1>Hello, React!</h1>;  
  
You can also create nested elements or component trees using JSX:  
const element = (  
 <div>  
 <h1>Welcome</h1>  
 <p>This is a React app.</p>  
 </div>  
);  
  
Each JSX element becomes a node in the virtual DOM.

Define how to render JSX to DOM

To render JSX into the actual browser DOM, you use the ReactDOM.render() or createRoot().render() (in React 18+).  
  
React 18 example:  
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(<App />);  
  
This renders the App component (which uses JSX) into the <div id="root"></div> in your HTML.

Explain how to use JavaScript expressions in JSX

In JSX, you can embed JavaScript expressions by enclosing them in curly braces {}.  
  
Example:  
const name = 'Sreeja';  
const element = <h1>Hello, {name}</h1>;  
  
You can also use:  
- Function calls  
- Conditionals (ternary)  
- Array methods (like .map())  
  
Example with ternary:  
const isLoggedIn = true;  
const element = <p>{isLoggedIn ? 'Welcome Back!' : 'Please Log In'}</p>;  
  
Note: You can't use statements (like if, for) directly in JSX.

Explain how to use inline CSS in JSX

In JSX, inline CSS styles are provided using a JavaScript object, where the property names are camelCase versions of CSS property names.  
  
Example:  
const styleObj = {  
 color: 'blue',  
 fontSize: '24px'  
};  
  
const element = <h1 style={styleObj}>Styled Text</h1>;  
  
You can also define styles inline directly:  
<h1 style={{ color: 'green', backgroundColor: 'lightgray' }}>Welcome</h1>  
  
**officespacerentalapp**

**App.js**

import React from 'react';

import './App.css';

const officeList = [

{

id: 1,

name: 'DBS',

rent: 50000,

address: 'Chennai',

image: 'https://images.unsplash.com/photo-1570129477492-45c003edd2be?auto=format&fit=crop&w=800&q=60'

},

];

function App() {

return (

<div style={{ padding: "30px" }}>

<h1>Office Space , at Affordable Range</h1>

{officeList.map((office) => (

<div key={office.id} style={{ marginTop: "30px" }}>

<img src={office.image} alt="office space" style={{ width: '300px' }} />

<h2>Name: {office.name}</h2>

<p style={{ color: office.rent < 60000 ? 'red' : 'green', fontWeight: 'bold' }}>

Rent: Rs. {office.rent}

</p>

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

</div>

))}

</div>

);

}

export default App;

**App.css**

body {

font-family: Arial, sans-serif;

}

h1 {

font-weight: bold;

}

h2 {

margin-top: 10px;

}

**Output:**

