**Question:React form Validation**

**Code:**

import React, { useState } from 'react';

function App() {

const [formData, setFormData] = useState({

name: '',

email: '',

});

const [errors, setErrors] = useState({});

const [submitted, setSubmitted] = useState(false);

const validate = () => {

const newErrors = {};

if (!formData.name.trim()) {

newErrors.name = 'Name is required';

}

if (!formData.email.trim()) {

newErrors.email = 'Email is required';

} else if (!/\S+@\S+\.\S+/.test(formData.email)) {

newErrors.email = 'Email format is invalid';

}

return newErrors;

};

const handleChange = (e) => {

const { name, value } = e.target;

setFormData((prev) => ({

...prev,

[name]: value,

}));

// Clear error on typing

setErrors((prevErrors) => ({

...prevErrors,

[name]: '',

}));

};

const handleSubmit = (e) => {

e.preventDefault();

const validationErrors = validate();

if (Object.keys(validationErrors).length > 0) {

setErrors(validationErrors);

setSubmitted(false);

} else {

setErrors({});

setSubmitted(true);

}

};

return (

<div style={styles.container}>

<h2 style={styles.heading}>🧪 React Form with Validation</h2>

<form onSubmit={handleSubmit} style={styles.form}>

<div style={styles.inputGroup}>

<label style={styles.label}>Name:</label>

<input

type="text"

name="name"

value={formData.name}

onChange={handleChange}

style={styles.input}

/>

{errors.name && <span style={styles.error}>{errors.name}</span>}

</div>

<div style={styles.inputGroup}>

<label style={styles.label}>Email:</label>

<input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

style={styles.input}

/>

{errors.email && <span style={styles.error}>{errors.email}</span>}

</div>

<button type="submit" style={styles.button}>Submit</button>

</form>

{submitted && (

<div style={styles.successBox}>

<h3>🎉 Form Submitted Successfully!</h3>

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

<p><strong>Email:</strong> {formData.email}</p>

</div>

)}

</div>

);

}

const styles = {

container: {

background: 'linear-gradient(to right, #ffecd2, #fcb69f)',

minHeight: '100vh',

padding: '40px',

fontFamily: 'Segoe UI, sans-serif',

},

heading: {

textAlign: 'center',

color: '#222',

marginBottom: '30px',

},

form: {

maxWidth: '400px',

margin: '0 auto',

background: '#fff',

padding: '30px',

borderRadius: '12px',

boxShadow: '0 0 10px rgba(0,0,0,0.1)',

},

inputGroup: {

marginBottom: '20px',

},

label: {

display: 'block',

fontWeight: 'bold',

marginBottom: '8px',

},

input: {

width: '100%',

padding: '10px',

borderRadius: '6px',

border: '1px solid #ccc',

},

button: {

width: '100%',

padding: '12px',

backgroundColor: '#00b894',

color: 'white',

border: 'none',

borderRadius: '8px',

fontSize: '16px',

cursor: 'pointer',

},

error: {

color: 'red',

fontSize: '13px',

marginTop: '5px',

display: 'block',

},

successBox: {

maxWidth: '400px',

margin: '30px auto 0',

background: '#dff9fb',

padding: '20px',

borderRadius: '10px',

textAlign: 'center',

border: '2px solid #00cec9',

},

};

export default App;

**Output:**

