**Explain React Forms validation**

**Validation** ensures the user provides correct and complete input before form submission. In React, you handle validation using state and logic in the component.

**Types of Validation:**

**Client-side** (before data is sent)

**Synchronous** (runs instantly)

**Custom logic** (e.g., required fields, length, pattern, etc.)

**Example :-**

const [email, setEmail] = useState('');

const [error, setError] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

if (!email.includes('@')) {

setError('Invalid email address');

} else {

setError('');

// Proceed with form submission

}

};

**Identify the differences between React Form and HTML Form**

An **HTML form** is a standard way of collecting user input on web pages using native HTML elements like <form>, <input>, <textarea>, and <button>. It relies on the browser’s default behavior for handling input and submission. When a user submits an HTML form, the browser typically reloads the page and sends the form data to the server using HTTP methods like GET or POST. State management in HTML forms is handled by the DOM, and form validation can be performed using built-in attributes like required, pattern, or minlength. This makes HTML forms relatively simple to implement but limited in customization and control.

In contrast, a React form is built using React components and JSX. It allows developers to manage form data explicitly through component state using hooks like useState or state management libraries like Redux. Rather than relying on the browser’s default submission behavior, React forms usually handle data submission using JavaScript functions and send data asynchronously to the server using tools like fetch or axios, which prevents page reloads. Validation in React forms is often done manually or with the help of libraries such as Formik or Yup, allowing for more flexible and complex validation logic. This approach offers more control and interactivity but requires more setup and JavaScript knowledge compared to traditional HTML forms.

**Explain about controlled components**

A **controlled component** is a form input element whose value is **controlled by React state**. The React component handles both the rendering and the updates of the input.

Example :-

const [name, setName] = useState('');

<input type="text" value={name} onChange={(e) => setName(e.target.value)} />

The input field is tied to the name state.

Any changes update the state via onChange.

**Identify various React Form input controls**

React supports all the standard HTML input types, including:

| **Input Type** | **React Example** |
| --- | --- |
| Textbox | <input type="text" value={val} onChange={...} /> |
| Password | <input type="password" value={val} onChange={...} /> |
| Textarea | <textarea value={val} onChange={...}></textarea> |
| Checkbox | <input type="checkbox" checked={val} onChange={...} /> |
| Radio | <input type="radio" value="X" checked={val === "X"} /> |
| Dropdown/Select | <select value={val} onChange={...}>...</select> |
| Submit button | <button type="submit">Submit</button> |

**Explain how to handle React Forms**

Handling a form in React involves:

1. State management for input fields
2. onChange handlers to update state
3. Form validation (optional)
4. onSubmit handler for submission logic

Example :-

function MyForm() {

const [username, setUsername] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

alert(`Hello ${username}`);

};

return (

<form onSubmit={handleSubmit}>

<input value={username} onChange={(e) => setUsername(e.target.value)} />

<button type="submit">Submit</button>

</form>

);

}

**Explain about submitting forms in React**

* Use <form onSubmit={handleSubmit}>
* Call e.preventDefault() to stop default behavior (page reload)
* Do something with the data (e.g., API call, alert, local storage)

**Sample Code:**

const handleSubmit = (e) => {

e.preventDefault();

console.log('Form submitted with:', name, email);

};

**Create a React App named “mailregisterapp” which will have a component named “register.js”.**

**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();

**App.js :-**

import React from 'react';

import Register from './register';

function App() {

  return (

    <div className="App">

      <Register />

    </div>

  );

}

export default App;

**Register.js :-**

import React, { useState } from 'react';

const Register = () => {

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    password: ''

  });

  const [errors, setErrors] = useState({

    name: '',

    email: '',

    password: ''

  });

  // Validate input on change

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value

    });

    // Validate the field on change

    let error = '';

    if (name === 'name') {

      if (value.length < 5) {

        error = 'Name must be at least 5 characters';

      }

    } else if (name === 'email') {

      if (!value.includes('@') || !value.includes('.')) {

        error = 'Email is not valid';

      }

    } else if (name === 'password') {

      if (value.length < 8) {

        error = 'Password must be at least 8 characters';

      }

    }

    setErrors({

      ...errors,

      [name]: error

    });

  };

  // Validate all fields on form submit

  const handleSubmit = (e) => {

    e.preventDefault();

    let formIsValid = true;

    let newErrors = {};

    if (formData.name.length < 5) {

      newErrors.name = 'Name must be at least 5 characters';

      formIsValid = false;

    }

    if (!formData.email.includes('@') || !formData.email.includes('.')) {

      newErrors.email = 'Email is not valid';

      formIsValid = false;

    }

    if (formData.password.length < 8) {

      newErrors.password = 'Password must be at least 8 characters';

      formIsValid = false;

    }

    setErrors(newErrors);

    if (formIsValid) {

      alert('Registration Successful!');

      // Here you can also reset form or send data to backend

      setFormData({ name: '', email: '', password: '' });

      setErrors({});

    } else {

      alert('Please fix the errors before submitting.');

    }

  };

  return (

    <div>

      <h2 style={{ color: 'red' }}>Register Here!!!</h2>

      <form onSubmit={handleSubmit}>

        <div>

          <label>Name: </label>

          <input

            type="text"

            name="name"

            value={formData.name}

            onChange={handleChange}

          />

          <div style={{ color: 'red' }}>{errors.name}</div>

        </div>

        <div>

          <label>Email: </label>

          <input

            type="text"

            name="email"

            value={formData.email}

            onChange={handleChange}

          />

          <div style={{ color: 'red' }}>{errors.email}</div>

        </div>

        <div>

          <label>Password: </label>

          <input

            type="password"

            name="password"

            value={formData.password}

            onChange={handleChange}

          />

          <div style={{ color: 'red' }}>{errors.password}</div>

        </div>

        <button type="submit">Submit</button>

      </form>

    </div>

  );

};

export default Register;

Output :-





