How to perform form validation in React?

Last Updated : 15 Feb, 2024

Form validation in React involves ensuring that the data entered into a form meets certain criteria before submission. In this, we will see the form validation in React.

Pre-requisites:

- NodeJS and NPM
- ReactJS
- React useState hook
- HTML, CSS, and JavaScript

Steps to Create React Application And Installing Module:

Step 1: Create a React application using the following command:

```
npx create-react-app react-form-validation
```

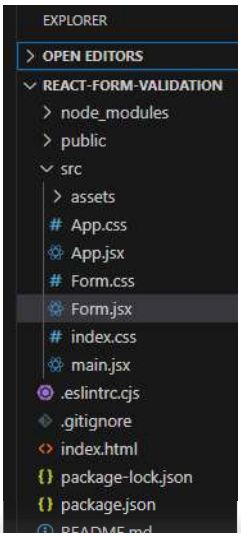
Step 2: After creating your project folder(i.e. react-form-validation), move to it by using the following command:

```
cd react-form-validation
```

Step 3: After creating the React application, Install the required package using the following command:

```
npm install
```

Project Structure:



- **Controlled Components:** React treats form elements as controlled components. This means that the React state is used to manage the values of the form elements. It's like React is in charge of the form, keeping everything organized.
- **State Management:** The `useState` hook in React is like a memory tool. It helps the app remember what users type in each form field and keeps an eye out for any mistakes. Think of it as giving each field its memory to recall what's been typed, making sure everything stays organized and error-free.
- **Event Handling:** In Event Handling when you type something into a box on a website, React pays attention. It uses this special skill called `onChange` notice when you change what's written. There's a helper named `handleChange()` that catches this change and makes sure to remember the recent content you typed.
- **Form Submission:** When you click "**submit**," React checks for errors using the `validateForm()` function. If there are no errors, it logs the submitted data; if there are errors, it shows error messages.
- **Validation Function:** A function called `validateForm()` checks each form field against specific rules and returns an object with error messages if any.

Example: Let's take one example to understand the form validation in React using controlled components.

Javascript

```
// Form.js
import React, { useState } from 'react';
import './Form.css';

function Form() {
    const [formData, setFormData] = useState({
        username: '',
        email: '',
```

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

const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
        ...formData,
        [name]: value,
    });
};

const handleSubmit = (e) => {
    e.preventDefault();
    const newErrors = validateForm(formData);
    setErrors(newErrors);

    if (Object.keys(newErrors).length === 0) {
        // Form submission logic here
        console.log('Form submitted successfully!');
    } else {
        console.log(`Form submission failed
         due to validation errors.`);
    }
};

const validateForm = (data) => {
    const errors = {};

    if (!data.username.trim()) {
        errors.username = 'Username is required';
    }

    if (!data.email.trim()) {
        errors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(data.email)) {
        errors.email = 'Email is invalid';
    }

    if (!data.password) {
        errors.password = 'Password is required';
    } else if (data.password.length < 8) {
        errors.password = `Password must be at
        least 8 characters long`;
    }
```

```
        return errors;
    };

    return (
        <div className="form-container">
            <h2 className="form-title">Form Validation</h2>
            <form onSubmit={handleSubmit}>
                <div>
                    <label className="form-label">
                        Username:
                        </label>
                    <input
                        className="form-input"
                        type="text"
                        name="username"
                        value={formData.username}
                        onChange={handleChange}
                    />
                    {errors.username &&
                        <span className="error-message">
                            {errors.username}
                        </span>
                    }
                </div>
                <div>
                    <label className="form-label">
                        Email:
                        </label>
                    <input
                        className="form-input"
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                    />
                    {errors.email &&
                        <span className="error-message">
                            {errors.email}
                        </span>
                    }
                </div>
                <div>
                    <label className="form-label">
```

```
                        className="form-input"
                        type="password"
                        name="password"
                        value={formData.password}
                        onChange={handleChange}
                    />
                    {errors.password &&
                        <span className="error-message">
                            {errors.password}
                        </span>
                    }
                </div>
                <div>
                    <label className="form-label">
                        Confirm Password:
                    </label>
                    <input
                        className="form-input"
                        type="password"
                        name="confirmPassword"
                        value={formData.confirmPassword}
                        onChange={handleChange}
                    />
                    {errors.confirmPassword &&
                        <span className="error-message">
                            {errors.confirmPassword}
                        </span>
                    }
                </div>
                <button className="submit-button"
                    type="submit">Submit</button>
            </form>
        </div>
    );
}

export default Form;
```

CSS

```
/* From.css */
.form-container {
```

```
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}


.form-title {
    margin-bottom: 20px;
    font-size: 24px;
    color: #333;
}


.form-label {
    display: block;
    margin-bottom: 8px;
    color: #333;
}


.form-input {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
}


.error-message {
    color: #ff0000;
    font-size: 14px;
    margin-top: 5px;
}


.submit-button {
    padding: 10px 20px;
    background-color: #0d427a;
    color: #fff;
    border: none;
```

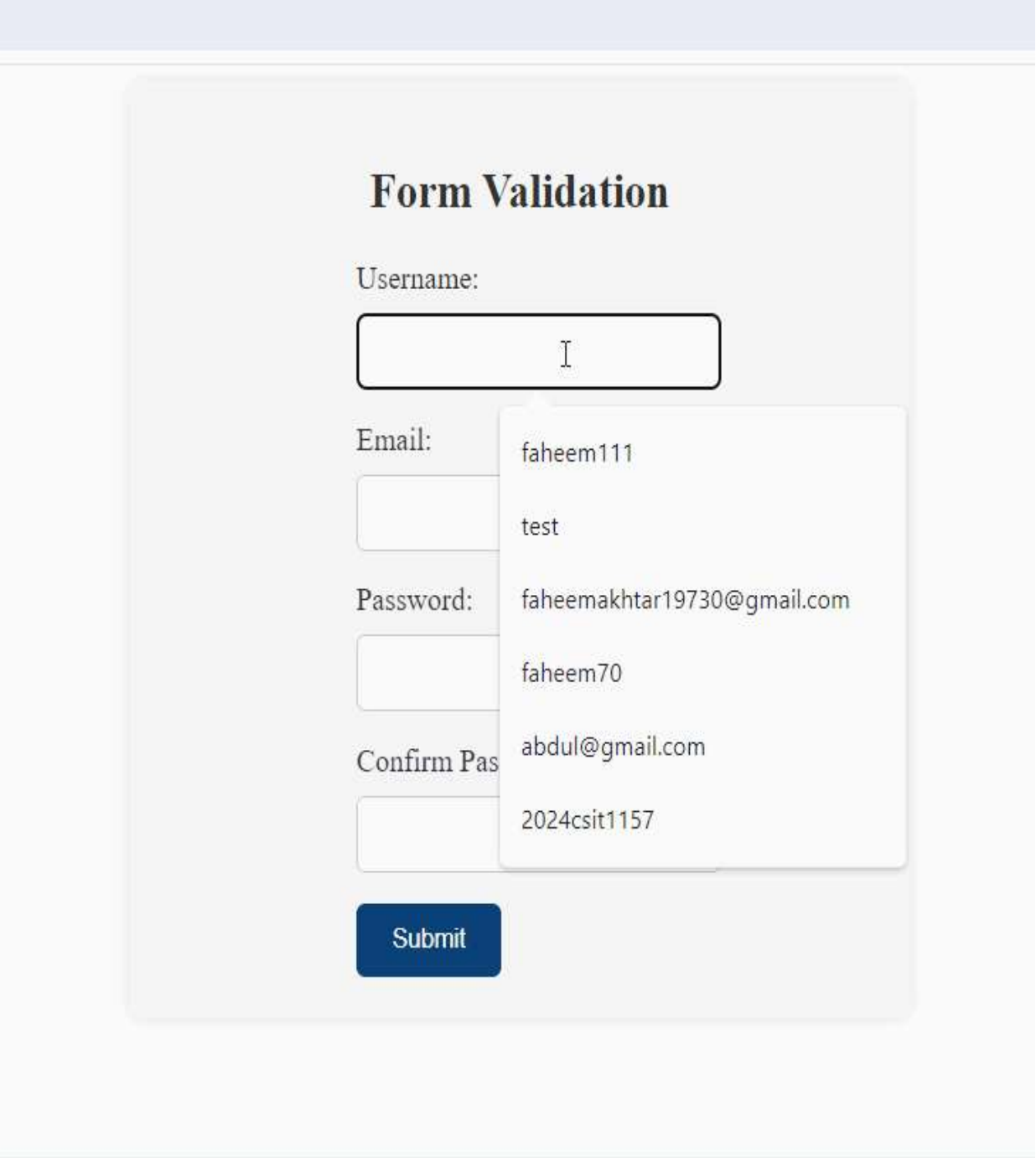
```
}

.submit-button:hover {
    background-color: #082647;
}
```

Start Your application using the following command:

```
npm start
```

Output:



Output

"This course was packed with amazing and well-organized content! The project-based approach of this course made it even better to understand concepts faster. Also the instructor in the live classes is really good and knowledgeable."- **Tejas | Deutsche Bank**

So get ready for salary hike only with our Full Stack Development Course.

Suggest improvement

Previous

How to Make Text Blink in ReactJS ?

Next

What is the use of the Array.shift() method in JavaScript?

Share your thoughts in the comments

Add Your Comment

Similar Reads

How to perform form validation for a required field in HTML ?

React Suite Form Validation Default Check

How to Implement Form Validation in React Native ?

React Suite Components Form validation Input

React.js Chakra UI Form Validation

React Suite Components Form validation

React Bootstrap Form Validation

How to Add Form Validation In Next.js ?

AngularJS Form Validation

Form required attribute with a custom validation message in HTML5