Form Validation in React.js

By Saksham Arya
10 mins read
Last updated: 27 Mar 2024
649 views
Topics Covered

Overview

We have seen many web or mobile applications that need the user's information and messages which are to be entered by a user only. These applications take the user's input as a form that contains several input fields. But sometimes, it is necessary to check whether the information inserted by the user is correct or not. For example, if a user is entering his or her name in the input field, it must not be an empty string. This is where form validation comes into play.

Introduction to Form Validation in ReactJS

Form validation in React Js is a process of validation of information that has been entered by a user who is using the application. This is done to ensure that the information entered by the user is valid. For example, the first name of anyone can never be an empty string, the password which has been created by the user can not be an empty string, etc.

There can be several constraints on the input data which have to be checked. If the user has entered any data without following its respective constraint (s), we have to notify the user and ask the user to re-enter the data while following the constraint (s).

For example, some application wants the user to enter a strong password. So there can be many constraints in this case. Some of these are given below.

- Password must be greater than 8 characters in length.
- Password must contain at least one uppercase character.
- Password must contain at least one lowercase character.
- Password must contain at least one digit.
- Password must contain at least one special character.

In the form validation process, we try to check whether the data inserted by the user in the input field is following all the constraints or not. After this checking process, if the input data passes all the constraints, we proceed to the next steps. Otherwise, if any of the constraints fail, we notify the user that the input data is not valid.

How to Create a Form in React?

Let us create a simple form in React Js which takes the user's first name, last name, email address, mobile number, age, and password.

1. **Creating React Application:**
   Go to a folder where you want to create the React Js application. Open the folder inside a code editor and the terminal, and write the following command to create react application with the name 'reactapplication'.

```
npx create-react-app reactapplication
```

3. **Running out React Js application:**
   To run the React application created in the previous step, open the terminal inside the newly create reactapplication folder and write npm start. This will run the React application in local host 3000.

   App.js:

```
import logo from './logo.svg'
import './App.css'

function App() {
  return <div className="main">This is the React JS Application</div>
}

export default App
```

   This is what our app looks like.

4. **Creating Form:**
   Let us create a form inside the App.js file of the react application which is present inside the src folder of the React application.

This form will take the user's first name, last name, email address, mobile number, age, and password as input.

We will use the useState hooks in React to store the information that is inserted by the user.

App.js:

```
import logo from './logo.svg'
import './App.css'
import { useState } from 'react'

function App() {
  const [firstName, setFirstName] = useState('') // useState to store First
Name
```

```
  const [lastName, setLastName] = useState('') // useState to store Last
Name
  const [mobile, setMobile] = useState('') // useState to store Mobile
Number
  const [age, setAge] = useState('') // useState to store Age
  const [email, setEmail] = useState('') // useState to store Email address
of the user
  const [password, setPassword] = useState('') // useState to store
Password

  return (
    <div className="main">
      <form>
        {/* Input Field to insert First Name */}
        <input
          placeholder="First Name"
          onChange={(e) => setFirstName(e.target.value)}
        />
        {/* Input Field to insert Last Name */}
        <input
          placeholder="Last Name"
          onChange={(e) => setLastName(e.target.value)}
        />
        {/* Input Field to insert Mobile Number */}
        <input
          placeholder="Mobile Number"
          onChange={(e) => setMobile(e.target.value)}
        />
        {/* Input Field to insert Age */}
        <input placeholder="Age" onChange={(e) => setAge(e.target.value)}
/>
        {/* Input Field to insert Email Address of the user */}
        <input placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} />
        {/* Input Field to insert Password */}
        <input
          placeholder="Password"
          onChange={(e) => setPassword(e.target.value)}
        />
        <button type="submit">Submit</button>
      </form>
    </div>
  )
}

export default App
```

App.css:

```
.main {
  padding: 5%;
  text-align: center;
}

input {
  padding: 1%;
  width: 50%;
  margin: 1% 2%;
}
```

```css
button {
  font-weight: bold;
  background: red;
  border: none;
  color: white;
  padding: 1%;
}
```

This is what our form looks like.

Quiz Pop

Quiz Type
SCQ
100
Success Rate:35%

What is the purpose of form validation in ReactJS?

To ensure that all forms are submitted successfully
To verify the correctness of user-entered data
To enhance the appearance of input fields
To speed up the rendering process of forms
Submit

How to Do Simple Form Validation in React.js?

Let us try to implement Form validation in React Js application.

Let us assume the following constraints for the input fields that we have in our form.

For First Name:

- First Name can not be an empty string.

For Email Address:

- Email Address can not be an empty string.

Password:

- Password must be greater than 8 characters in length.
- Password must contain at least one uppercase character.
- Password must contain at least one lowercase character.
- Password must contain at least one digit.
- Password must contain at least one special character.

Given below is the implementation of the Form Validation in React Js. The form will be validated when we click the submit button. If all the input data are according to our assumed constraints, we will alert Form is Valid. Otherwise, if any of the input data does not follow our assumed constraints, we will alert the user the form is not valid.

App.js:

```
import logo from './logo.svg'
import './App.css'
import { useState } from 'react'

function App() {
  const [firstName, setFirstName] = useState('') // useState to store First
Name
  const [lastName, setLastName] = useState('') // useState to store Last
Name
  const [mobile, setMobile] = useState('') // useState to store Mobile
Number
  const [age, setAge] = useState('') // useState to store Age
  const [email, setEmail] = useState('') // useState to store Email address
of the user
  const [password, setPassword] = useState('') // useState to store
Password

  // Function which will validate the input data whenever submit button is
clicked.

  function validateForm() {
    // Check if the First Name is an Empty string or not.

    if (firstName.length == 0) {
      alert('Invalid Form, First Name can not be empty')
      return
    }

    // Check if the Email is an Empty string or not.

    if (email.length == 0) {
      alert('Invalid Form, Email Address can not be empty')
      return
    }

    // check if the password follows constraints or not.

    // if password length is less than 8 characters, alert invalid form.

    if (password.length < 8) {
      alert(
        'Invalid Form, Password must contain greater than or equal to 8
characters.',
```

```
      )
      return
    }

    // variable to count upper case characters in the password.
    let countUpperCase = 0
    // variable to count lowercase characters in the password.
    let countLowerCase = 0
    // variable to count digit characters in the password.
    let countDigit = 0
    // variable to count special characters in the password.
    let countSpecialCharacters = 0

    for (let i = 0; i < password.length; i++) {
      const specialChars = [
        '!',
        '@',
        '#',
        '$',
        '%',
        '^',
        '&',
        '*',
        '(',
        ')',
        '_',
        '-',
        '+',
        '=',
        '[',
        '{',
        ']',
        '}',
        ':',
        ';',
        '<',
        '>',
      ]

      if (specialChars.includes(password[i])) {
        // this means that the character is special, so increment
countSpecialCharacters
        countSpecialCharacters++
      } else if (!isNaN(password[i] * 1)) {
        // this means that the character is a digit, so increment
countDigit
        countDigit++
      } else {
        if (password[i] == password[i].toUpperCase()) {
          // this means that the character is an upper case character, so
increment countUpperCase
          countUpperCase++
        }
        if (password[i] == password[i].toLowerCase()) {
          // this means that the character is lowercase, so increment
countUpperCase
          countLowerCase++
        }
      }
    }
```

```
    if (countLowerCase == 0) {
      // invalid form, 0 lowercase characters
      alert('Invalid Form, 0 lower case characters in password')
      return
    }

    if (countUpperCase == 0) {
      // invalid form, 0 upper case characters
      alert('Invalid Form, 0 upper case characters in password')
      return
    }

    if (countDigit == 0) {
      // invalid form, 0 digit characters
      alert('Invalid Form, 0 digit characters in password')
      return
    }

    if (countSpecialCharacters == 0) {
      // invalid form, 0 special characters characters
      alert('Invalid Form, 0 special characters in password')
      return
    }

    // if all the conditions are valid, this means that the form is valid

    alert('Form is valid')
  }

  return (
    <div className="main">
      <form>
        {/* Input Field to insert First Name */}
        <input
          placeholder="First Name"
          onChange={(e) => setFirstName(e.target.value)}
        />
        {/* Input Field to insert Last Name */}
        <input
          placeholder="Last Name"
          onChange={(e) => setLastName(e.target.value)}
        />
        {/* Input Field to insert Mobile Number */}
        <input
          placeholder="Mobile Number"
          onChange={(e) => setMobile(e.target.value)}
        />
        {/* Input Field to insert Age */}
        <input placeholder="Age" onChange={(e) => setAge(e.target.value)}
/>
        {/* Input Field to insert Email Address of the user */}
        <input placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} />
        {/* Input Field to insert Password */}
        <input
          placeholder="Password"
          onChange={(e) => setPassword(e.target.value)}
        />
        <button
          type="submit"
          onClick={() => {
```

```
            validateForm()
          }}
        >
          Submit
        </button>
      </form>
    </div>
  )
}

export default App
```

Use Case - 1:
If we Enter an Empty First Name and Submit, we will See Something Like this.

Use Case - 2:
If we Enter an Empty Email Address and Submit, we will See Something Like this.

Use Case - 3:
If we enter a password with a length < 8, we will see something like this.

Use Case - 4:
If we enter a password without any upper case character, we will see something like this.

Use Case - 5:
If we enter a password without any lowercase characters, we will see something like this.

Use Case - 6:
If we enter a password without any digit character, we will see something like this.

Use Case - 7:
If we enter a password without any special characters, we will see something like this.

Use Case - 8:
If all the input data is valid, we will see something like this.

Validation Using Yup Library

In React Js, we have several libraries which can speed up the process of form validation in React Js applications. One such library is the Yup Library. Let us use the Yup library to validate our form according to the constraints we have assumed.

For the convenience of the reader, our assumed constraints are given below.

For First Name:

- First Name can not be an empty string.

For Email Address:

- Email Address can not be an empty string.

Password:

- Password must be greater than 8 characters in length.
- Password must contain at least one uppercase character.
- Password must contain at least one lowercase character.
- Password must contain at least one digit.
- Password must contain at least one special character.

1. **Install the Yup Library:**
   In the terminal, write npm i yup. You can also refer to the official NPM documentation of the Yup Library. Yup NPM
2. **Creating Yup Schema which will validate our form data and validating input:**
   In Yup, we have the functionality to create a schema of the data which we will take from the user as input. This schema will help us to validate our form data.

Given below is the code in which, we have created the Yup schema and used it to validate the form data.

App.js:

```
import logo from './logo.svg'
import './App.css'
import { useState } from 'react'
import * as yup from 'yup' // importing functions from yup library

function App() {
  const [firstName, setFirstName] = useState('') // useState to store
First Name
  const [lastName, setLastName] = useState('') // useState to store Last
Name
  const [mobile, setMobile] = useState('') // useState to store Mobile
Number
  const [age, setAge] = useState('') // useState to store Age
  const [email, setEmail] = useState('') // useState to store Email
address of the user
  const [password, setPassword] = useState('') // useState to store
Password

  // defining yup schema to validate our form

  const userSchema = yup.object().shape({
    // name can not be an empty string so we will use the required
function
    firstName: yup.string().required(),
    lastName: yup.string(),
```

```
    // email can not be an empty string so we will use the required
function
    email: yup.string().email().required(),
    // password can not be an empty string so we will use the required
function. Also, we have used the `min` function to set the minimum length
of the password. Yup passwords by default handle the conditions of at
least one upper case, at least one lower case, and at least one special
character in the password
    password: yup.string().min(8).required(),
    age: yup.string(),
    mobile: yup.string(),
  })

  // Function which will validate the input data whenever submit button
is clicked.

  async function validateForm() {
    // creating a form data object

    let dataObject = {
      firstName: firstName,
      lastName: lastName,
      email: email,
      password: password,
      age: age,
      mobile: mobile,
    }

    // validating this dataObject concerning Yup userSchema

    const isValid = await userSchema.isValid(dataObject)

    if (isValid) {
      alert('Form is Valid')
    } else {
      alert('Form is Invalid')
    }
  }

  return (
    <div className="main">
      <form>
        {/* Input Field to insert First Name */}
        <input
          placeholder="First Name"
          onChange={(e) => setFirstName(e.target.value)}
        />
        {/* Input Field to insert Last Name */}
        <input
          placeholder="Last Name"
          onChange={(e) => setLastName(e.target.value)}
        />
        {/* Input Field to insert Mobile Number */}
        <input
          placeholder="Mobile Number"
          onChange={(e) => setMobile(e.target.value)}
        />
        {/* Input Field to insert Age */}
        <input placeholder="Age" onChange={(e) => setAge(e.target.value)}
/>
        {/* Input Field to insert Email Address of the user */}
```

```
        <input placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} />
        {/* Input Field to insert Password */}
        <input
          placeholder="Password"
          onChange={(e) => setPassword(e.target.value)}
        />
        <button
          type="submit"
          onClick={() => {
            validateForm()
          }}
        >
          Submit
        </button>
      </form>
    </div>
  )
}

export default App
```

Use Case - 1:
If all the input data is valid, we will see something like this.