TASK

React - Form Validation

Visit our website

Introduction

WELCOME TO THE REACT - FORM VALIDATION TASK!

Up to this point, you have learnt quite a bit about the ins and outs of React. After a brief overview, you learnt about the rendering of elements, JSX, and then you worked with functional components. You have covered local state management, events and hooks. You also delved into routing. It is now time to have a look at an important element of user interaction: form input. We will be focussing on client-side validation of form input received from the user. Let's go!

A note from the

HyperionDev Team

The folks at **MDN Webdocs** have provided us with a helpful description of what form validation is: "When you enter data, the browser and/or the web server will check to see that the data is in the correct format and within the constraints set by the application. Validation done in the browser is called client-side validation, while validation done on the server is called server-side validation.

BUILT-IN HTML VALIDATION

There are several built-in validation attributes in HTML. This forms part of the most basic type of form validation that is available to us.

The following is a non-exhaustive list of the most common types of built-in validation attributes that belong to the input and other form elements:

- **`required`** - creates a compulsory field
- **`type`** - determines the type of input (E.g., text, email, password etc.)
- **`minlength`** and **`maxlength`** - determines the required length of the input
- **`min`** and **`max`** - the smallest or biggest number that can be entered
- **`multiple`** - allows the user to enter multiple values in an input field
- **`pattern`** - allows the matching of the input against a regular expression

As a reminder, a basic example of validation attributes in regular HTML is included below:

```
<form>
      <label for="email">Email:</label>
      <input type="email" name="email" id="email" required/>
</form>
```

In React, it is very similar, except that the **`for`** attribute cannot be used and is replaced by **`htmlFor`** instead.

Although these are very useful attributes to have in one's toolbox, they often do not produce a good user interface (UI) or user experience (UX). They are also often notoriously difficult (and laborious) to style in an acceptable and/or pleasing way. Another limitation is you can't set up custom error messages with these attributes.

As you have probably guessed by now there are better ways to deal with form validation and this is by using JavaScript or React libraries that were written especially for this purpose.

FORMIK

Formik is one of the world's most popular, open-source form libraries for React that provides functions to handle the creation and validation of forms. To quote from their website:

"Formik takes care of the repetitive and annoying stuff—keeping track of values/errors/visited fields, orchestrating validation, and handling submission—so you don't have to. This means you spend less time wiring up state and change handlers and more time focusing on your business logic."

Formik is basically a small group of hooks and React components that handle the nitty gritty of building forms. To get started, visit the very short **tutorial** posted on their site.

Once you have your React project up and running, you can install Formik by using the following command:

```
npm install formik -save
```

Once you have installed formik you can experiment with some code such as the example below. We'll use a simple newsletter signup form as an example. This form, like most newsletter signups, has two elements: an email input field and a submit button. The following code is explained using comments in the code blocks.

```
import { useFormik } from 'formik';

const SignupForm = () => {
    // Pass the initial form value and a submit function to the
    // useFormik() hook, which returns an object of form state and
    // helper methods
    const formik = useFormik({
    initialValues: {
      email: '',
    },
    onSubmit: values => {
      // Submit handling code goes here
    },
  });
  // Rendering of the form
  return (
    // Note the use of the formik object below
    <form onSubmit={formik.handleSubmit}> // Submit handler
      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onChange={formik.handleChange} // Change handler
        value={formik.values.email} // Current value of input field
      />

      <button type="submit">Submit</button>
    </form>
  );
};

export default SignUpForm;
```

Some important features to take note of:

1. If there is more than one form field, the same **handleChange** function is used.
2. The **id** and **name** properties must match the property defined in the **initialValues** object.
3. The field's value is accessed via the same name **value={formik.values.email}** in this case.

Formik keeps track of the form's values, validation, and error messages, so the next step is to set up a custom JavaScript validation function and then pass this function to the **useFormik()** hook. For example:

```
import { useFormik } from ‘formik’;

const validate = values => {
   const errors = {};

   if (!values.email) {
      errors.email = “Required”;
            // This is a regular expression of a valid email
   } else if (!/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/i.test(values.email)){
      errors.email = “Invalid email address”;
   }
   return errors;

};
const SignupForm = () => {
    // Pass the initial form value and a submit function to the
    // useFormik() hook which returns an object of form state and
    // helper methods
    const formik = useFormik({
    initialValues: {
      email: '',
    },
    validate, // -> Pass validate to the useFormik() hook
    onSubmit: values => {
      // Submit handling code goes here
    },
  });

<form onSubmit={formik.handleSubmit}> // Submit handler
      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onChange={formik.handleChange}
        onBlur={formik.handleBlur}
        value={formik.values.email}
      />
      // Logic for the display of the error message defined in validate
      {formik.errors.email ? <div>{formik.errors.email}</div> : null}
      <button type="submit">Submit</button>
    </form>
```

This code would work, but the user would not have a good experience because an error message would be generated on each keystroke. This is definitely not what we want. Instead, we want the error to appear only after the user is done typing in the field.

Formik has a solution for us. It keeps track of which fields have been visited and stores this information in an object called `touched`. The keys of this object are the field names, and the values are booleans (`true` or `false`). To use this object, we assign the value `formik.handleBlur` to the input element(s) `onBlur` property. Finally, we tweak the logic of the rendering of the error message slightly so that it only appears when the error exists and when the user has visited that particular field.

```
<form onSubmit={formik.handleSubmit}> // Submit handler
      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onChange={formik.handleChange}
        onBlur={formik.handleBlur}
        value={formik.values.email}
      />
      // Tweaked error message display logic
      {formik.touched.email && formik.errors.email ?
         <div>{formik.errors.email}</div> : null}
      <button type="submit">Submit</button>
    </form>
```

And that is all there is to Formik. It is a really useful library that will make future React form handling a breeze.

Extra resource

There are several other libraries available for the creation and/or validation of forms in React. Explore some **alternatives** in a blog article summarising some of the top React form libraries.

SPOT CHECK 1

Let's see what you can remember from this section.

1. What are some ways that you can do form validation with pure HTML?
2. What is Formik?
3. Why is a React form-handling library such as Formik preferred over native HTML form validation?

Instructions

Ensure that you have installed Formik **before** attempting this task.

Compulsory Task

Use Formik to:

- Add the following to the Online Store web application you created in the previous task:
  - A login page containing input for:
    - An email address
    - A password with a minimum length of 8 characters
  - A registration page containing input for the following:
    - First Name (Should not exceed 15 characters)
    - Surname (Should not exceed 20 characters)
    - A valid email address
    - A password (Must contain 8 characters or more, at least one uppercase and lowercase letter, a number and a special case character)
    - A confirm password field (To ensure both passwords match)
- Make sure that the data entered is in the correct format (i.e., validate the user input)

Refer to the Formik **tutorial** as a reference and for more detailed information.

Rate us

Share your thoughts

HyperionDev strives to provide internationally excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we’ve done a good job?

Click here to share your thoughts anonymously.

SPOT CHECK 1 ANSWERS

1. The following HTML attributes can be used to validate form input:
   a. required
   b. type
   c. minlength and maxlength
   d. min and max
   e. multiple
2. Formik is an open source library for React that provides functions for the creation and validation of forms.
3. Although HTML form validation is very useful it often does not produce a good user interface (UI) or user experience (UX). It is also often notoriously difficult (and laborious) to style in an acceptable and/or pleasing way. Another limitation is you can't set up custom error messages with these attributes. Formik bypasses all of these limitations.

REFERENCES

MDN Webdocs - Client-side form validation (2023) Retrieved on 20 July 2023
https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation

Dev.to website - Top 10 React form libraries for efficient form creation (March 2023) Retrieved on 20 July 2023
https://dev.to/femi_dev/top-10-react-form-libraries-for-efficient-form-creation-hp2

Formik library home page (2020) Retrieved on 20 July 2023
https://formik.org/

Formik Library - Overview and getting started (2020) Retrieved on 20 July 2023
https://formik.org/docs/tutorial