

Forms and Validation in ReactJs

In this article, we'll explore the fundamentals of Forms and Validation in ReactJs. We'll dive into the different ways you can handle form submission and validation using popular libraries and hooks.



by Curious Coder (CodeWithCurious.com)

Introduction to Forms in ReactJs

Handling forms is a common requirement for any web application. In ReactJs, forms are controlled components meaning that the component state manages the input values rather than the DOM. This section will provide an overview of form handling in ReactJs.

Uncontrolled Components

Uncontrolled Components are traditional HTML form controls. ReactJs retrieves their current values using a ref. These components feel more natural and browser-native to some developers.

Controlled Components

Controlled components are React components that provide all of their form data through props to the parent component. This is React's recommended approach to handling form data.

Form elements in ReactJs

Forms in ReactJs are composed of various form elements that differ in their behavior and can be customized. In this section, we'll explore the basic and complex form elements included in ReactJs and their distinctive features.

Input

The Input component is a basic form element that receives most user inputs. It can accept a variety of attributes such as 'type', 'value', and 'placeholder' and emit various events such as 'onChange', 'onBlur', and 'onKeyDown'.

Select

The Select component provides a dropdown list of selectable items. React wraps the native Select component in two other components, the Option component to wrap each option and the OptGroup component to group the options.

Checkbox

The Checkbox component is a basic form element that is used for selecting one or more options from a set of options. ReactJs provides a rich interface to customize Checkbox functionality.

Form Submitting in ReactJs

Submitting forms in ReactJs can be achieved through a variety of methods, such as sending an HTTP POST request, interfacing with an external API, etc. In this section, we'll explore how to handle form submission in ReactJs.

Using Formik, developers can have an opportunity to handle and analyze complex forms efficiently. It abstracts away many of the inconveniences associated with handling form submission and provides a means to incorporate complex form validation without having to reinvent the wheel.

Validation using Formik library

Invalid form data can cause issues for your application and your users. The Formik library provides a simple means of validating users' inputs and displaying error messages.

1 Formik Yup validation Schema

Formik uses the Yup library to handle validation. Formik Yup validation schema is a powerful and flexible way to validate user inputs. It offers a simple schema definition via a chainable interface and a comprehensive list of validation rules.

2 Formik Helper Functions

Formik provides helper functions to help you test and validate form inputs. It simplifies handling onSubmit and onSubmitting events and provides many additional features to assist with rendering form state, form events, and errors.

Handling errors and displaying error messages

Error messages help users know why their input has been rejected. The Formik library provides a simple interface to create and display form errors. In this section, we'll explore how to manage errors and display them properly.

Managing error on a field

Formik provides tools to set, remove or access data from individual form fields. To display an error message, we can associate each error message with its corresponding field using the 'errors' property of Formik.

Displaying an Error Message

In Formik, a touched property is set to true if the user has interacted with the field. By default, error messages are only displayed if the corresponding field is touched, and an error exists for the field.

Form Validation using React Hooks

The React Hooks library provides a simple interface to share stateful logic between different components of your application. It is an efficient way to handle data and lifecycle operations. In this section, we'll explore form validation using React Hooks and how to apply them to different areas of your application.

1 **useState and useEffect Hooks**

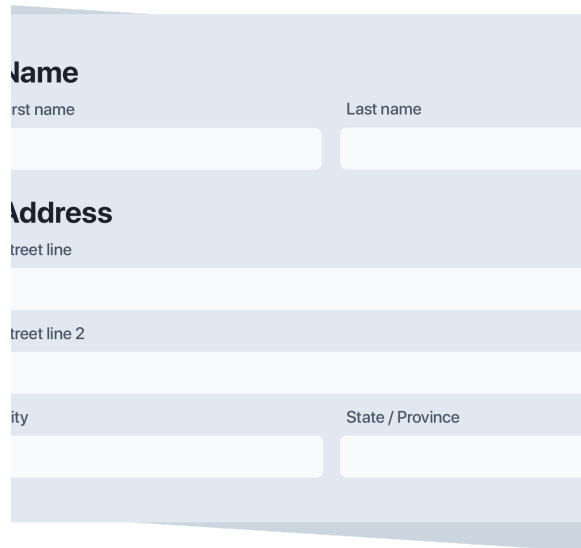
The useState and useEffect Hooks provide you with a convenient way to manage component state and perform data side effects. They're perfect for managing form validation and can be used in combination with other Hooks or traditional React components.

2 **useForm Hook**

The useForm Hook is a custom Hook that simplifies working with a form. It abstracts away many of the inconveniences associated with individual form fields and provides various functions to interact with form data.

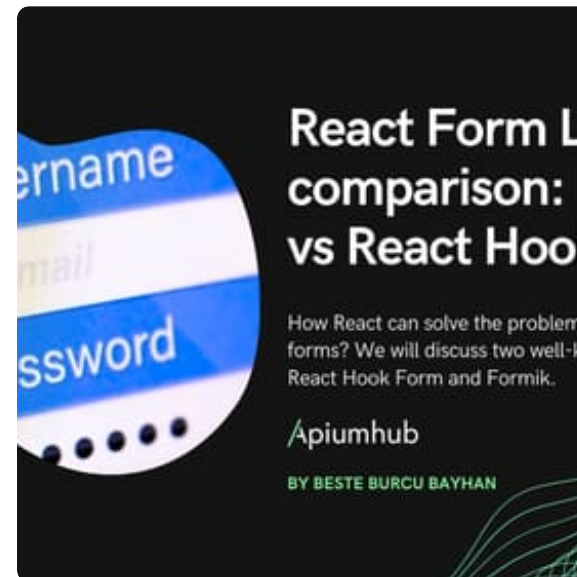
Conclusion and Key Takeaways

Handling forms and validation in ReactJs can seem daunting, but it doesn't have to be. With ReactJs libraries like Formik and Hooks, you can easily create, validate, and manage your data. In this article, we've covered the basics of form handling and validation and explored various methods to make the process simpler. Remember to keep your design concise, engage users, and make sure they get answers to their questions efficiently.

A mockup of a form with a light blue header and footer. The main content area is white. It features three sections: 'Name' with 'First name' and 'Last name' labels and input fields; 'Address' with 'Street line' and 'Street line 2' labels and input fields; and 'City' with 'City' and 'State / Province' labels and input fields.

Forms in ReactJs

Understand the basic and complex form elements in ReactJs. Learn how to manage data and control user inputs to improve application performance and simplify data handling.



Formik Library

Learn how to use Formik to manage form data and validation easily. Formik offers a simple schema definition via a chainable interface and a comprehensive list of validation rules.