React Validate Hook: Validation Without the Baggage

Kabui, Charles

2025-10-05

Table of Contents

> A lightweight, flexible, and type-safe React form validation hook with zero dependencies (except React).

Most React form libraries force you into an all-or-nothing relationship. Want validation? You get form state management, field registration, submit handling, and a kitchen sink of features you didn't ask for.

What if you just want validation?

The Problem with Current Solutions

```
// react-hook-form: Great, but brings everything
const { register, handleSubmit, formState: { errors } } = useForm();

// Formik: Powerful, but heavyweight
<Formik initialValues={...} validationSchema={...} onSubmit={...}>

// Yup/Zod alone: No React integration
const schema = z.string().email();
```

Each forces you into their world. But sometimes you already have state management. Sometimes you just need to validate a search box, or a single field, or a multi-step wizard where each step has different validation rules.

A Different Approach

react-validate-hook does one thing: **validates React components**. That's it.

```
import { useValidator } from 'react-validate-hook';

function LoginForm() {
  const { ValidateWrapper, validate, errors } = useValidator();
  const [email, setEmail] = useState('');

  return (
    <ValidateWrapper
      setValue={setEmail}
      fn={(value) => {
        if (!value) return "Email required";
        if (!/\S+@\S+\.\S+/.test(value)) return "Invalid email";
        return true;
      }}
    >
      {({ error, setValue }) => (
        <input
          value={email}
          onChange={(e) => setValue(e.target.value)}
          className={error ? 'border-red-500' : 'border-gray-300'}
        />
      )}
    </ValidateWrapper>
```

```
  );
}
```

Your state, your components, your rules. The hook just handles validation.

Why This Matters

For Design System Builders: Provide validation as a primitive, not a framework. Your components stay flexible.

For Existing Apps: Drop validation into forms without refactoring your entire state management.

For Complex UIs: Multi-step wizards, conditional validation, non-form validation (search, filters, settings).

For Schema Users: Bring your own validation library:

Factory Validation (Schema-Based)

For complex validation rules using Zod, Yup, or custom schemas:

```
// Works with Zod, Yup, Joi, or custom validators

import { useValidator } from 'react-validate-hook';
import { z } from 'zod';

const emailSchema = z.string().email("Invalid email");
const passwordSchema = z.string().min(8, "Min 8 characters");

function SignupForm() {
  const { ValidateWrapper, validate, errors } = useValidator(
    (data, schema) => {
      const result = schema.safeParse(data);
      return result.success ? true : result.error.errors[0].message;
    }
  );

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

  return (
```

```
    <form onSubmit={(e) => { e.preventDefault(); validate(); }}>
      <ValidateWrapper setValue={setEmail} fn={emailSchema}>
        {({ error, setValue }) => (
          <input
            type="email"
            onChange={(e) => setValue(e.target.value)}
            aria-invalid={!!error}
          />
        )}
      </ValidateWrapper>

      <ValidateWrapper setValue={setPassword} fn={passwordSchema}>
        {({ error, setValue }) => (
          <input
            type="password"
            onChange={(e) => setValue(e.target.value)}
            aria-invalid={!!error}
          />
        )}
      </ValidateWrapper>

      <button type="submit" disabled={errors.length > 0}>
        Sign Up
      </button>
    </form>
  );
}
```

The Numbers

- **2KB minified** (vs 9KB for react-hook-form, 13KB for Formik)
- **Zero dependencies** except React
- **Full TypeScript support** with generics
- **18 test cases** covering edge cases

When NOT to Use This

Be honest about what you need:

- **Building standard CRUD forms?** -> Use react-hook-form

- **Need complex field dependencies?** -> Use Final Form
- **Want everything handled for you?** -> Use Formik

But if you need validation that fits *your* architecture instead of forcing you into theirs, this might be exactly what you're looking for.

Get started: `npm install react-validate-hook`

GitHub: https://github.com/ToKnow-ai/react-validate-hook **npm:** https://www.npmjs.com/package/react-validate-hook

Sometimes the best tool is the one that doesn't try to do everything.
