

Docs

Components

Blocks

Charts

Themes

Colors

Search...

🔍

🔄

🌙

Getting Started

Introduction

Installation

components.json

Theming

Dark mode

CLI

Monorepo New

Next.js 15 + React 19

Typography

Open in v0

Figma

Changelog

Installation

Next.js

Vite

Remix

Astro

Laravel

Gatsby

Manual

Components

Accordion

Alert

Alert Dialog

Aspect Ratio

Avatar

Badge

Breadcrumb

Button

Calendar

Card

Carousel

Chart

Docs > React Hook Form

React Hook Form

Building forms with React Hook Form and Zod.

Docs

Forms are tricky. They are one of the most common things you'll build in a web application, but also one of the most complex.

Well-designed HTML forms are:

- Well-structured and semantically correct.
- Easy to use and navigate (keyboard).
- Accessible with ARIA attributes and proper labels.
- Has support for client and server side validation.
- Well-styled and consistent with the rest of the application.

In this guide, we will take a look at building forms with `react-hook-form` and `zod`. We're going to use a `<FormField>` component to compose accessible forms using Radix UI components.

Features

The `<Form />` component is a wrapper around the `react-hook-form` library. It provides a few things:

- Composable components for building forms.
- A `<FormField />` component for building controlled form fields.
- Form validation using `zod`.
- Handles accessibility and error messages.
- Uses `React.useId()` for generating unique IDs.
- Applies the correct `aria` attributes to form fields based on states.
- Built to work with all Radix UI components.
- Bring your own schema library. We use `zod` but you can use anything you want.
- You have full control over the markup and styling.**

Anatomy

```
<Form>
  <FormField
    control={...}
    name="..."
    render={() => (
      <FormItem>
        <FormLabel />
        <FormControl>
          { /* Your form field */ }
        </FormControl>
        <FormDescription />
        <FormMessage />
      </FormItem>
    )}
  />
</Form>
```

Example

```
const form = useForm()

<FormField
  control={form.control}
  name="username"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Username</FormLabel>
      <FormControl>
        <Input placeholder="shadcn" {... field} />
      </FormControl>
      <FormDescription>This is your public display name.</FormDescription>
      <FormMessage />
    </FormItem>
  )}
/>
```

Installation

CLIManual

1 Command

```
pnpm npm yarn bun

pnpm dlx shadcn@latest add form
```

Usage

1 Create a form schema

Define the shape of your form using a Zod schema. You can read more about using Zod in the [Zod documentation](#).

```
1 "use client"
2
3 import { z } from "zod"
4
5 const formSchema = z.object({
6   username: z.string().min(2).max(50),
7 })
```

2 Define a form

Use the `useForm` hook from `react-hook-form` to create a form.

```
1 "use client"
2
3 import { zodResolver } from "@hookform/resolvers/zod"
4 import { useForm } from "react-hook-form"
5 import { z } from "zod"
6
7 const formSchema = z.object({
8   username: z.string().min(2, {
9     message: "Username must be at least 2 characters.",
10  }),
11 })
12
13 export function ProfileForm() {
14   // 1. Define your form.
15   const form = useForm<z.infer<typeof formSchema>>({
16     resolver: zodResolver(formSchema),
17     defaultValues: {
18       username: "",
19     },
20   })
21
22   // 2. Define a submit handler.
23   function onSubmit(values: z.infer<typeof formSchema>) {
24     // Do something with the form values.
25     // ✅ This will be type-safe and validated.
26     console.log(values)
27   }
28 }
```

Since `FormField` is using a controlled component, you need to provide a default value for the field. See the [React Hook Form docs](#) to learn more about controlled components.

3 Build your form

We can now use the `<Form />` components to build our form.

```
1 "use client"
2
3 import { zodResolver } from "@hookform/resolvers/zod"
4 import { useForm } from "react-hook-form"
5 import { z } from "zod"
6
7 import { Button } from "@/components/ui/button"
8 import {
9   Form,
10  FormControl,
11  FormDescription,
12  FormField,
13  FormItem,
14  FormLabel,
15  FormMessage,
16 } from "@/components/ui/form"
17 import { Input } from "@/components/ui/input"
18
19 const formSchema = z.object({
20   username: z.string().min(2, {
21     message: "Username must be at least 2 characters.",
22   }),
23 })
24
25 export function ProfileForm() {
26   // ...
27
28   return (
29     <Form {...form}>
30       <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8">
31         <FormField
32           control={form.control}
33           name="username"
34           render={({ field }) => (
35             <FormItem>
36               <FormLabel>Username</FormLabel>
37               <FormControl>
38                 <Input placeholder="shadcn" {... field} />
39               </FormControl>
40               <FormDescription>
41                 This is your public display name.
42               </FormDescription>
43               <FormMessage />
44             </FormItem>
45           )}
46         />
47         <Button type="submit">Submit</Button>
48       </form>
49     </Form>
50   )
51 }
```

4 Done

That's it. You now have a fully accessible form that is type-safe with client-side validation.

Style: New York

Username

shadcn

This is your public display name.

Submit

Examples

See the following links for more examples on how to use the `<Form />` component with other components:

- [Checkbox](#)
- [Date Picker](#)
- [Input](#)
- [Radio Group](#)
- [Select](#)
- [Switch](#)
- [Textarea](#)
- [Combobox](#)

< Dropdown Menu

Hover Card >

Built by [shadcn](#). The source code is available on [GitHub](#).

PDF

Created by [JLKN700N6](#) [PAGG](#) [PDF](#)