Using the react-hook-form Library

React Hook Form (`react-hook-form`) is a **performance-optimized** library that helps manage forms in React by leveraging **uncontrolled components** and **direct DOM manipulation**.

It simplifies form handling by providing a declarative API, built-in validation, and seamless integration with third-party libraries like **Zod**.

How to use react-hook-form

1. Install using npm

```
npm install react-hook-form
```

2. Basic Usage with register

```
import React from "react";
import { useForm } from "react-hook-form";

export default function BasicForm() {
  const { register, handleSubmit } = useForm();

  const onSubmit = (data) => {
    console.log("Form Data:", data);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register("firstName")} placeholder="Name" />
      <input {...register("email")} placeholder="Email" />
      <button type="submit">Submit</button>
    </form>
  );
}
```

💡 `register()` binds the input field to React Hook Form. This also returns the following methods that can be passed as props to the `input`.

```
const { onChange, onBlur, name, ref } = register('firstName');
```

This tracks changes to the input via `onChange` and `onBlur` and It also attaches a `ref` to the input.

Unlike controlled components (which rely on `useState` for each input), React Hook Form **registers input fields directly to the DOM**, reducing re-renders and improving performance.

Key Concepts:

1. **Uncontrolled Inputs** – Uses `register()` instead of `useState()`.
2. **Ref-based Registration** – Directly attaches inputs to the form.
3. **Minimal Re-renders** – Only the affected fields re-render, not the whole form.
4. **Built-in Validation** – Uses HTML5 validation rules and custom validation.

Additional React Hook Form Features

1. Provide Default values

```
// set default value sync
useForm({
  defaultValues: {
    firstName: 'Anuj',
    lastName: 'Sharma'
  }
})
```

```
// set default value async
useForm({
  defaultValues: async () => fetch('/api-endpoint');
})
```

2. Reset the form

```
const { register, handleSubmit, reset } = useForm({
  defaultValues: {
    firstName: 'Anuj',
    lastName: 'Sharma'
  }
});

const onSubmit = (data) => {
  console.log(data);
  reset();  // Clears form
};
```

3. Form State

```
const {
  register,
  handleSubmit,
  formState: { errors, isDirty, isSubmitting, isValid },
} = useForm();
const onSubmit = (data) => console.log(data);

return (
  <form onSubmit={handleSubmit(onSubmit)}>
    <input {...register("test")} />
    <input type="submit" />
  </form>
);
```

4. Watch to monitor the value of form fields in real-time.

```
const { register, handleSubmit, watch } = useForm();

// Watch the 'email' field
const emailValue = watch("email");
const { email, password } = watch(["email", "password"]); // Watching mul
```

💡

- `watch` does **NOT** trigger re-renders (which makes it efficient).
- If you need a controlled input that re-renders, use `useState` instead.

5. Form Validation

```
<input {...register("firstName", { required: true })} />
<input {...register("lastName", { minLength: 2 })} />
```

```
<input
  {...register("username", {
    required: "Username is required",
    minLength: { value: 3, message: "At least 3 characters required" },
    maxLength: { value: 10, message: "Max 10 characters allowed" }
  })}
  placeholder="Enter Username"
/>
{errors.username && <p>{errors.username.message}</p>}
```

6. Using the Zod Resolver

   Zod can be used with React Hook Form for **schema-based validation**. This is especially useful for complex forms with advanced validation rules.

   1. Install zod resolver

      ```
      npm install zod @hookform/resolvers
      ```

   2. Create a Zod Schema

```
import { z } from 'zod';

const loginSchema = z.object({
  name: z.string().min(1, 'Name is required'),
  email: z.string().min(1, 'Email is required').email('Invalid email address'),
});
```

3. Use Zod with react-hook-form

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { z } from 'zod';

function ZodForm() {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm({
    resolver: zodResolver(loginSchema),
  });

...
}
```