Introduktion

Formulär används typiskt för att samla in data som sen skickas till servern

- **Textinmatningar:** För att skriva in data, t.ex. titel, beskrivning och regissör.
- **Nummerinmatningar:** För att ange numeriska värden, som produktionsår.
- **Knappar:** För att skicka in formuläret.
- **Felmeddelanden:** För att visa valideringsfel eller andra problem vid inmatning.

Introduktion till react-hook-form

Några viktiga begrepp:

`register` - kopplar ett inmatningsfält till react-hook-form

`handleSubmit` - funktion som anropas när formuläret ska skickas in

`formState` - innehåller information om formulärets state, ev errors m.m.

`reset` - återställer formuläret till ursprungligt skick

register

Kopplar varje inmatningsfält till formulärets state. Detta gör att react-hook-form kan hålla koll på värden, validering och ändringar.

Exempel:

```
const { register } = useForm();
//senare i formuläret
<input {...register("titel")} />;
```

Resultatet blir:

```
<input
  name="title"
  ref={register("title").ref}
  onChange={register("title").onChange}
  onBlur={register("title").onBlur}
/>
```

Validering med register

Olika typer av validering:

```
// Obligatoriskt fält
<input {...register("title", { required: "Titel är obligatorisk" })} />
// Minsta antal tecken
<input {...register("title", { minLength: { value: 3, message: "Minst 3 tecken" } })} />
// Högsta antal tecken
<input {...register("title", { maxLength: { value: 50, message: "Max 50 tecken" } })} />
// Endast bokstäver och mellanslag (Regex)
<input {...register("title", { pattern: { value: /^[A-Za-z\s]+$/, message: "Endast bokstäver tillåtna" } })} />
// Anpassad validering: Måste innehålla "film"
<input {...register("title", { validate: value => value.includes("film") || "Måste innehålla 'film'" })} />
```

handleSubmit

En funktion som skickar med den insamlade formulärdatan till din callback-funktion

```
<form onSubmit={handleSubmit(onSubmit)}>...</form>
```

formState

Innehåller statusinformation om formuläret, som felmeddelanden (errors), vilket gör det enkelt att visa feedback till användaren.

Exempel:

```
const {
  formState: { errors },
} = useForm();
//senare i formuläret:
{
  errors.titel && <span>Fältet är obligatoriskt</span>;
}
```

reset

En funktion som återställer formuläret till dess ursprungliga värden. Efter inskickande av data anropas typiskt reset()

Steg-för-steg: Bygga filmregistreringskomponenten

Installera och importera

Börja med att installera react-hook-form:

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

Skapa komponent-fil för registrering av ny film

Importera hooks i komponenten:

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

Definiera typ för formulärdata

Skapa en typ som beskriver de värden formuläret ska hantera:

```
type MovieFormValues = {
  title: string;
  description: string;
  productionYear: number;
  director: string;
};
```

Använd useForm-hooken

Skapa själva komponent-koden (`const RegisterMovie = () =>` ..)

Anropa sedan useForm först i komponenten för att hantera formulärets logik.

```
const {
  register,
  handleSubmit,
  formState: { errors },
  reset,
} = useForm<MovieFormValues>();
```

Skapa en optional useState för fel vid registrering

```
const [message, setMessage] = useState("");
```

`message` skriver felmeddelande om registringen misslyckas

Skapa callback-funktion

```
const handleRegisterMovie = async (data: MovieFormValues) => {
  const token = localStorage.getItem("token");
  if (!token) {
    setMessage("Ingen giltig token hittades. Logga in igen.");
    return;
  }
  try {
    const response = await fetch("http://localhost:8080/movies", {
      method: "POST",
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "application/json",
      },
      body: JSON.stringify(data),
    });
    if (!response.ok) {
      throw new Error(
        "Kunde inte registrera filmen. Kontrollera att du har rätt behörighet."
      );
    }
    setMessage("Filmen registrerades framgångsrikt!");
    reset(); // Återställ formuläret vid lyckad registrering
  } catch (err) {
    if (err instanceof Error) {
      setMessage(err.message);
    }
  }
};
```

Bygg formuläret med JSX

```
return(
    {/* skriv först ut ev meddelande om  fel vid registrering */}
    {message && <div>{message}</div>}
    <form onSubmit={handleSubmit(handleRegisterMovie)}>
    <div>
        <input type="text" placeholder="Titel" {...register("title", { required: "Titel är obligatorisk" })}/>
        {errors.title && <p>{errors.title.message}</p>}
    </div>
    <div>
        <input type="number" placeholder="Produktionsår" {...register("productionYear",
                {required: "Produktionsår är obligatoriskt", valueAsNumber: true})} />
        {errors.productionYear && <p>{errors.productionYear.message}</p>}
    </div>

    {/* NOT: fortsätt att skapa input-fält omgärdade av en div för varje attribut i Movie */}

    <button type="submit">
        Registrera film
    </button>
    </form>
)
```

Exportera till sist komponenten

```
export default RegisterMovie;
```