REACT HOOK FORM

POR QUE O REACT HOOK FORM EXISTE?

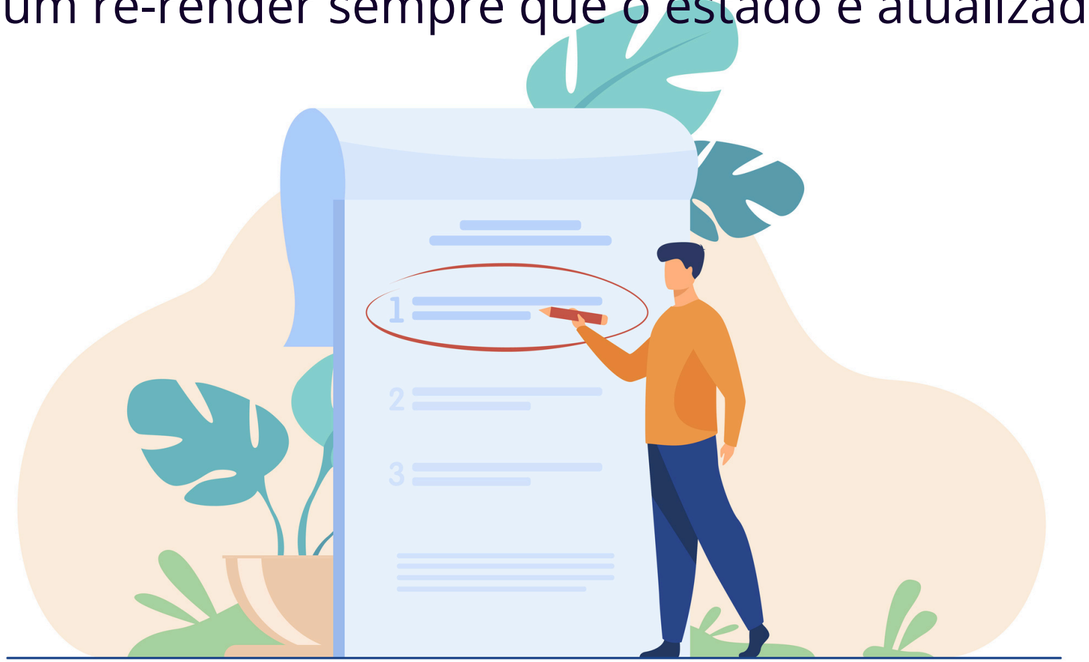
No React, os formulários são tradicionalmente gerenciados com o *useState*, onde cada alteração no campo dispara um re-render do componente. Isso pode impactar a performance em formulários grandes.
O React Hook Form surgiu para solucionar esse problema, permitindo um gerenciamento eficiente de formulários **sem re-renderizações** desnecessárias. Ele utiliza **Refs** ao invés de useState, armazenando os valores dos campos diretamente no DOM, o que melhora a performance e reduz o código necessário.

O QUE SÃO REFS?

No React, **refs** (referências) são usadas para acessar diretamente elementos do DOM ou armazenar valores que não disparam re-renderizações quando alterados. Elas são criadas usando o **useRef**.

Diferença entre *useRef* e *useState*:

- *useRef* armazena valores sem causar re-renderizações do componente.
- *useState d*ispara um re-render sempre que o estado é atualizado.



OUTROS USOS DE REFS

Refs no React são usadas em diversos cenários além do gerenciamento de formulários com *React Hook Form*. Aqui estão alguns exemplos:

01 Acessar Elementos do DOM

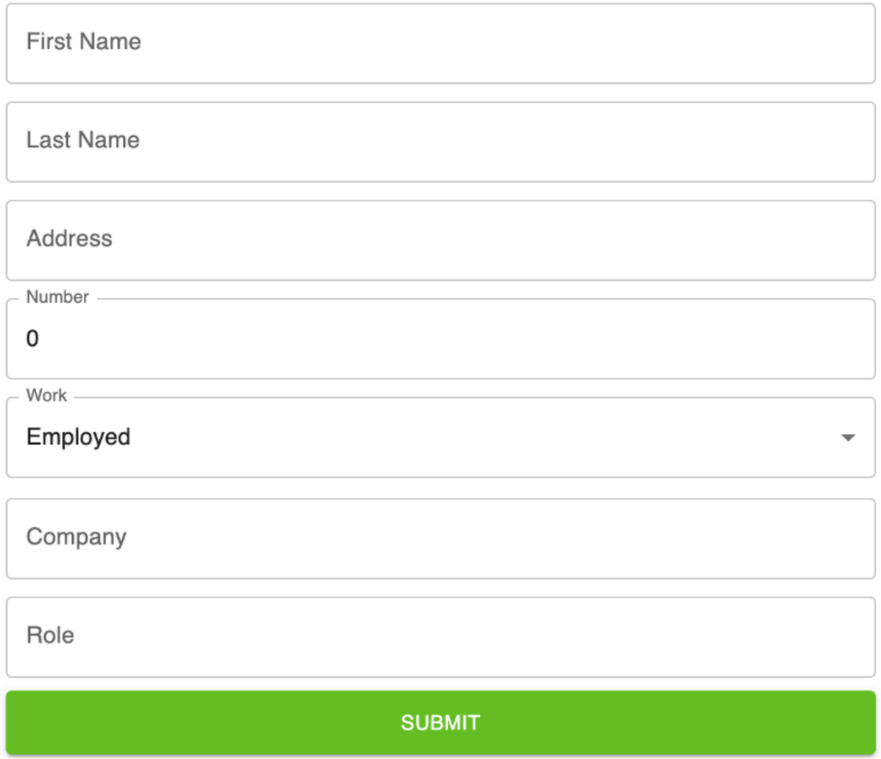
02 Manter Valores entre Re-renders sem Causar Re-renderizações

03 Manter Referência para um Intervalo ou Timeout

04 Referenciar um Componente sem Prop Drilling

05 Animações e Medidas do DOM

O QUE É REACT HOOK FORM E POR QUE USÁ-LO?



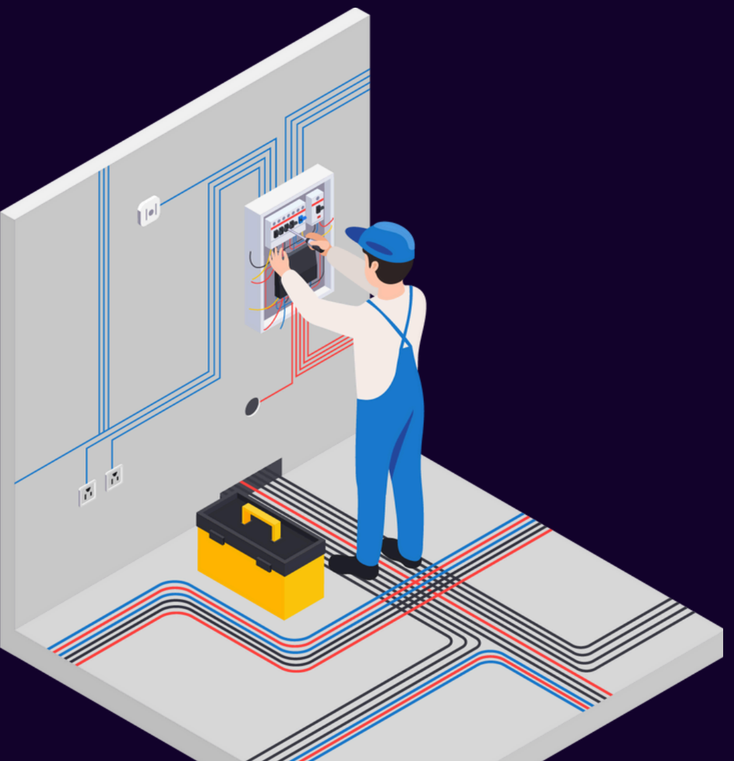
O **React Hook Form** é uma biblioteca leve para gerenciamento de formulários no React. Seus principais benefícios são:

- Menos re-renderizações.
- Validações eficientes e fáceis de configurar.
- Integração com bibliotecas de UI.

INSTALAÇÃO E CONFIGURAÇÃO

Instale o React Hook Form no projeto:

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



CRIANDO UM FORMULÁRIO BÁSICO

Criamos um formulário com campos para Nome, Cargo, Salário e Data de Admissão:

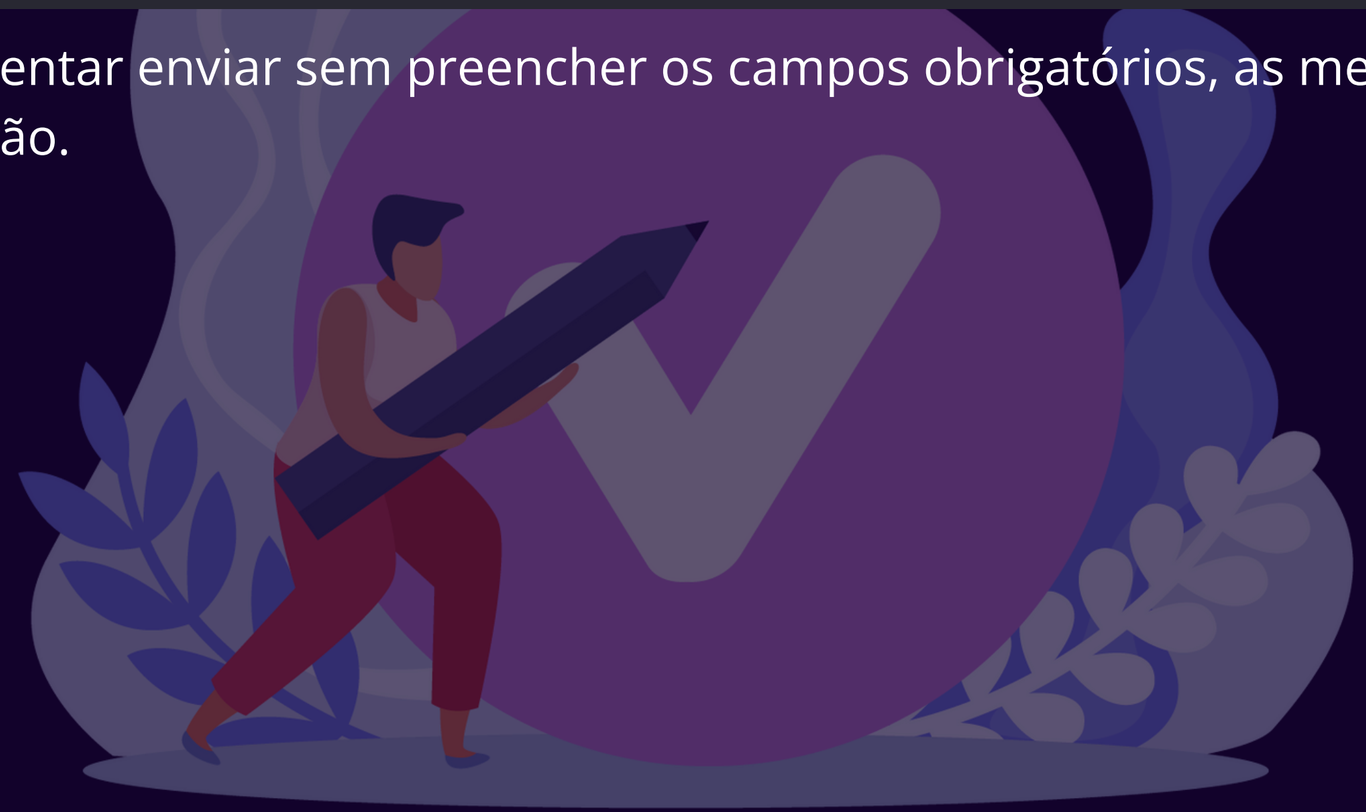
```tsx
import { useForm } from "react-hook-form";

interface Funcionario {
  nome: string;
  cargo: string;
  salario: number;
  dataAdmissao: string;
}

export default function FormularioCadastro() {
  const { register, handleSubmit } = useForm<Funcionario>();

  const onSubmit = (data: Funcionario) => {
    console.log(data);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register("nome")} placeholder="Nome" />
      <input {...register("cargo")} placeholder="Cargo" />
      <input {...register("salario")} type="number" placeholder="Salário" />
      <input {...register("dataAdmissao")} type="date" />
      <button type="submit">Enviar</button>
    </form>
  );
}
```

ADICIONANDO VALIDAÇÕES

Podemos adicionar validações nos campos:

```
1 <input {...register("nome", { required: "Nome é obrigatório" })} />
2 <input {...register("salario", { min: 1000, required: "Salário é obrigatório" })} />
```

Se o usuário tentar enviar sem preencher os campos obrigatórios, as mensagens de erro aparecerão.



EXIBINDO MENSAGENS DE ERRO

Podemos exibir os erros usando errors do React Hook Form:

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

<input {...register("nome", { required: "Nome é obrigatório" })} />
{errors.nome && <p>{errors.nome.message}</p>}
```

THANK YOU

@wallace027dev