3 minutes

trigger(), setFocus() et criteriaMode

Définir le focus manuellement sur un champ

`useForm()` met à disposition la méthode `setFocus()` permettant de donner manuellement le focus à un champ.

Le premier argument est le nom du champ enregistré à focus. Optionnellement, vous pouvez passer en deuxième argument `{ shouldSelect: true }` pour sélectionner le contenu du champ.

```
import { useEffect } from 'react';
import { useForm } from 'react-hook-form';

export default function App() {
  const { register, handleSubmit, setFocus } = useForm();
  const onSubmit = (data) => console.log(data);

  useEffect(() => {
    setFocus("firstName");
  }, [setFocus]);

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

Copier

Déclencher manuellement la validation

`useForm()` met à disposition la méthode `trigger()` permettant de manuellement déclencher la validation du formulaire ou d'un seul champ.

Vous pouvez ne passer aucun argument pour déclencher la validation de tout le formulaire ou passer le nom d'un champ enregistré ou plusieurs noms de champs enregistrés dans un tableau.

La plupart du temps c'est utile lorsque la validation d'un champ dépend de la valeur d'un autre champ.

Optionnellement, vous pouvez passer en deuxième argument `{ shouldFocus: true }` pour sélectionner le champ dont vous déclenchez la validation.

Afficher toutes les erreurs d'un champ

Par défaut, seule la première erreur de chaque champ est disponible sur l'objet `errors`.

Si vous souhaitez que toutes les erreurs de chaque champ soient disponibles, il faut utiliser l'option `criteriaMode: 'all'` de `useForm()` :

```
const {
    register,
    handleSubmit,
```

```
    formState: { errors, isSubmitting, submitCount },
  } = useForm({
    defaultValues: {
      name: '',
    },
    criteriaMode: 'all',
  });
```

Copier

Vous pourrez ensuite accéder à toutes les erreurs d'un champ en utilisant la propriété `types` :

```
{errors?.name && (
  <ul style={{ color: 'red' }}>
    {Object.keys(errors.name.types).map((k) => (
      <li key={k}>{errors.name.types[k]}</li>
    ))}
  </ul>
)}
```

Copier