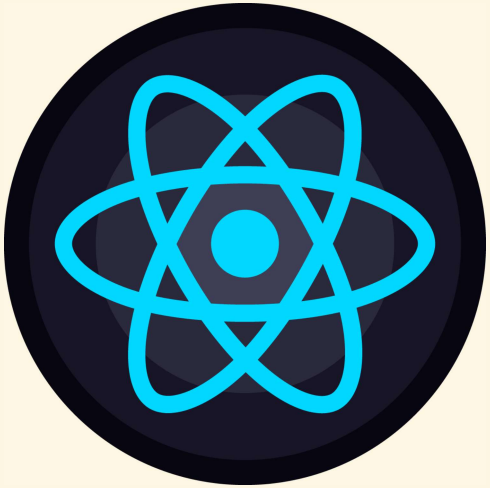
REACT : LES FORMULAIRES



LES FORMULAIRES

- Les formulaires sont des composants qui permettent de récupérer des données de l'utilisateur.
- Il existe deux types de formulaires :
  - Contrôlés : les données sont gérées par le composant.
  - Non contrôlés : les données sont gérées par le DOM.

LES FORMULAIRES

- Comme en HTML, Les champs peuvent posseder des attributs :
  - value : valeur par défaut du champ.
  - onChange : fonction appelée à chaque modification du champ.
  - required : champ obligatoire.

LES FORMULAIRES

- name : nom du champ, permet au serveur de récupérer la valeur du champ.
- type : type du champ. (text, email, password, number, date, checkbox, radio, file, submit, reset, button)
- placeholder : texte affiché dans le champ.

LES FORMULAIRES

- Les formulaires peuvent être soumis :
  - En cliquant sur un bouton submit.
  - En appuyant sur la touche entrée.

CONTRÔLÉ VS NON CONTRÔLÉ

- Regardons cet expemple:

```
<form>
  <input name="nonControllé" placeholder="nonControllé" />
  <input name="controllé" value="controllé" />
</form>
```

CONTRÔLÉ VS NON CONTRÔLÉ

- Le premier champ est **non contrôlé**, il est géré par le **DOM**.
- l'entrée dont le **name** est **non contrôlé** n'a pas d'attribut value et peut être modifiée librement

CONTRÔLÉ VS NON CONTRÔLÉ

- Le second champ est **contrôlé**, il est géré par le **composant**.
- l'entrée dont le **name** est **contrôlé** a un attribut value et ne peut pas être modifié

CONTRÔLÉ VS NON CONTRÔLÉ

- En **HTML**, on peut modifier un champ, que l'attribut **value** soit présent ou non.
- En **react** on ne peut modifier un champ que si l'attribut **value** est présent.

CONTRÔLÉ VS NON CONTRÔLÉ

- En react, il est recommandé d'utiliser des champs contrôlés.

CHAMP CONTRÔLÉ

- Pour créer un champ contrôlé, il faut utiliser l'attribut **value** et **onChange**.
- Nous allons créer un composant qui affichera un formulaire avec un champs nom.
- Exemple:

CHAMP CONTRÔLÉ

```tsx
import { useState } from "react";
import "./heroForm.css";

const TestForm: React.FC = () => {
  const [personne, setPersonne] = useState<string>("Luke skywalker");
  return (
    <form>
      <label htmlFor="nom">Nom :</label>
      <input type="text" id="name" name="nom" value={personne} />

      <input type="submit" value="Envoyer" />
    </form>
  );
};

export default TestForm;
```

CHAMP CONTRÔLÉ

- Nous pouvons voir que le champ est contrôlé car il possède un attribut **value**.
- Et nous ne pouvons pas changer la valeur du champ car il n'y a pas d'attribut **onChange**.

CHAMP CONTRÔLÉ

- Nous allons ajouter un attribut **onChange** pour pouvoir modifier la valeur du champ.
- On va pour cela créer une fonction qui sera appelée à chaque modification du champ.

```
const changePersonne = (event: React.ChangeEvent<HTMLInputElement>) => {
  setPersonne(event.target.value);
};
```

CHAMP CONTRÔLÉ

- Nous allons ensuite ajouter cet attribut **onChange** à notre champ.

```
<input
  type="text"
  id="name"
  name="nom"
  value={personne}
  onChange={changePersonne}
/>
```

CHAMP CONTRÔLÉ

- Par defaut, la page est rechargée à chaque soumission du formulaire.
- On va pour cela créer une fonction qui sera appelée à chaque soumission du formulaire, en utilisant `event.preventDefault();`

```
const soumettre = (event: React.FormEvent<HTMLFormElement>) => {
  event.preventDefault();
  console.log(personne);
};
```

CHAMP CONTRÔLÉ

- Nous allons ensuite ajouter cet attribut **onSubmit** à notre formulaire.

```
<form onSubmit={soumettre}>
```

DÉMONSTRATION

LA SUITE

Par ici !