Les Formulaires

Composant React avec un état local

Les balises de formulaire

Pour interagir avec la saisie utilisateur en React, il faut utiliser les balises de formulaire :

- <form> ... </form>
- <input ... />
- <textarea ... />
- <select> ... </select>
- <label> ... </label>

Les balises de formulaire - HTML vs JSX

Quelques différences entre les balises formulaire HTML et JSX :

- L'événement « onChange » se déclenche dès qu'il y a un changement de valeur.
- La balise « textarea » utilise l'attribut "value" et s'écrit en mono-ligne.
- La balise « select » possède un attribut "value" pour sélectionner la valeur utiliser, cela permet d'éviter l'utilisation de l'attribut "selected" sur une des options.
- L'attribut "for" de la balise « label » est remplacé par "htmlFor".

Les balises de formulaire - Source de données

Un problème se pose avec les balises de formulaire :

- Les balises de formulaire possèdent leur propre valeur interne, qui est modifiée lorsque l'utilisateur interagit avec celle-ci.
- Un composant possède un "état local" qui représente les valeurs de l'élément.

⚠️ Il y a donc 2 sources de données possibles !

Composants contrôlés

Pour résoudre se problème, il faut synchroniser la valeur interne des balises avec les valeurs du « State », pour obtenir une seule source de vérité.

Comment réaliser cela :

- Utiliser l'événement « onChange » de la balise des formulaires pour modifier l'état d'une variable du « State » lors de la saisie de l'utilisateur.
- Définir la valeur de la balise de formulaire comme étant égale à la variable contenu dans le « State »

Exemple de composant contrôlé - Input

- La balise « Input » de type « text »

```
import { useState } from 'react';

const ExempleInput = function () {
    const [msg, setMsg] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }

    return (
        <form onSubmit={handleSubmit}>
            <label htmlFor='msg'>Entrer un message</label>
            <input id='msg' type='text' value={msg} onChange={(e) => setMsg(e.target.value)} />

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

export default ExempleInput
```

Exemple de composant contrôlé - Textarea

- La balise « Textarea »

```
import { useState } from 'react';

const ExempleTextarea = function () {
    const [msg, setMsg] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }

    return (
        <form onSubmit={handleSubmit}>
            <label htmlFor='msg'>Entrer un message</label>
            <textarea id='msg' value={msg} onChange={(e) => setMsg(e.target.value)} />

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

export default ExempleTextarea
```

Exemple de composant contrôlé - Select

- La balise « Select »

```
import { useState } from 'react';

const ExempleSelect = function () {
    const [choice, setChoice] = useState('firefox');
    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }
    return (
        <form onSubmit={handleSubmit}>
                <label htmlFor='nav'>Quel est votre navigateur</label>
                <select id='nav' value={choice} onChange={(e) => setChoice(e.target.value)}>
                    <option value="chrome">Google Chrome</option>
                    <option value="firefox">Mozilla Firefox</option>
                    <option value="other">Autre</option>
                </select>
                <input type='submit' value='Envoyer' />
            </form>
    );
}

export default ExempleSelect
```

Exemple de composant contrôlé - Checkbox

- La balise « Input » de type « checkbox »

```
import { useState } from 'react';

const ExempleCheckbox = function () {
    const [isCheck, setCheck] = useState(false);

    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }

    return (
        <form onSubmit={handleSubmit}>
            <label htmlFor='case'>Case à cocher</label>
            <input id='case' type='checkbox' checked={isCheck} onChange={(e) => setCheck(e.target.checked)} />

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

export default ExempleCheckbox
```

Formulaire avec plusieurs valeurs de saisie

Dans ce cas, il y a deux méthodes pour mettre en place la synchronisation :

- Utiliser une valeur d'état (State) pour chaque les valeurs du formulaire
  - Utiliser un hook d'état pour créer une valeur pour chacun des éléments.
  - Synchroniser chaque balise individuellement.

- Utiliser un state global pour stocker toutes les valeurs du formulaire
  - Utiliser un hook d'état pour créer une valeur de type objet.
  - Ajouter une propriété dans l'objet pour chaque élément.
  - Ajouter un nom *(identique à la propriété du state)* dans chaque balise du formulaire.
  - Créer une méthode qui synchronise la balise et la valeur de state sur base du nom.

Utiliser la méthode la plus adéquate par rapport au formulaire de votre composant.

Composant contrôlé - Plusieurs hook d'état

```jsx
import { useState } from 'react';

const MultiField = function () {
    const [name, setName] = useState('');
    const [nbGuest, setNbGuest] = useState('1');
    const [isPresent, setPresent] = useState(false);

    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }

    return (
        <form onSubmit={handleSubmit}>
            <label htmlFor='name'> Nom :</label>
            <input id="name" type="text" value={name} onChange={(e) => setName(e.target.value)} />

            <label htmlFor='nbGuest'> Nombre de personne :</label>
            <input id="nbGuest" type="number" min={1} value={nbGuest} onChange={(e) => setNbGuest(e.target.value)} />

            <label htmlFor='isPresent'>Soirée uniquement :</label>
            <input id="isPresent" type="checkbox" checked={isPresent} onChange={(e) => setPresent(e.target.checked)} />

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

export default MultiField
```

Composant contrôlé - Hook d'état global

```jsx
import { useState } from 'react';

const MultiField = function () {
    const [inputs, setInputs] = useState({ name: '', nbGuest: '1', isPresent: false });

    const handleSubmit = (e) => {
        e.preventDefault();
        // Do something
    }

    const handleInput = (e) => {
        const { name, type, checked, value } = e.target;
        setInputs({
            ...inputs,
            [name]: type === 'checkbox' ? checked : value
        });
    }

    return (
        <form onSubmit={handleSubmit}>
            <label htmlFor='name'> Nom :</label>
            <input id="name" name="name" type="text" value={inputs.name} onChange={handleInput} />

            <label htmlFor='nbGuest'> Nombre de personne :</label>
            <input id="nbGuest" name="nbGuest" type="number" min={1} value={inputs.nbGuest} onChange={handleInput} />

            <label htmlFor='isPresent'>Soirée uniquement :</label>
            <input id="isPresent" name="isPresent" type="checkbox" checked={inputs.isPresent} onChange={handleInput} />

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

export default MultiField
```

Valeur en lecture seul dans un composant contrôlé

Pour créer des balises en lecture seul, il faut utiliser l'attribut « readOnly ».

```
import { useState } from 'react';

const ExempleInput = function () {
    const [number, setNumber] = useState('0');
    const [result, setResult] = useState('0');

    const handleNumberInput = (e) => {
        const nb = e.target.value;
        setNumber(nb);
        setResult(parseInt(nb) * 2);
    }

    return (
        <form>
            <label htmlFor='number'>Entrer un nombre</label>
            <input id='number' type='number' value={number} onChange={handleNumberInput} />

            <label htmlFor='result'>Le double du nombre est</label>
            <input id='result' type='number' value={result} readOnly />
        </form>
    );
}

export default ExempleInput
```

Exercice • Manipulation des formulaires

Composant React avec un état local

Exercice

- Créer un composant « Calculatrice »
  - Il se compose de :
    - Deux champs de type “texte”, dont la saisie est limité au valeur numérique.
    - Un sélecteur pour l’opération
    - Un bouton de validation
    - Un champs texte en lecture qui affichera le résultat de l'opération