Chapitre 7 : React et les formulaires

Table des matières

I. Rappel les formulaires en HTML

En HTML, un formulaire a pour principale objectif l'envoie des informations d'un poste client vers un serveur web.

La déclaration d'un formulaire se fait à l'aide de la balise form :

```
<form> et</form>
```

A l'intérieur de la balise **form** va contenir **les champs de formulaire** : Ce sont des objets comme des zones de texte ou des boutons.

a.Balise <input>

La zone de texte est le champ de formulaire le plus célèbre. Il permet d'écrire un texte sur une seule ligne (comme le nom ou le login par exemple). Pour créer une zone de texte on fait appel à la balise <input>. Il s'agit d'une balise orpheline (pas de </input>).

Les principaux attributs de cette balise sont :

Attribut type : L'attribut type permet de définir le type de champ à intégrer. Voici les valeurs à mettre pour cet attribut et qui permettent d'avoir les champs les plus fréquents:
text: Permet d'avoir une zone de texte normale (adapté pour les noms, emails, login...)

- **password**: Permet d'avoir une zone de mot de passe où les caractères sont masqués.
- **radio** : Pour créer un bouton radio. Il s'agit d'un petit cercle qu'on peut cocher et qui permet de faire un choix unique. Le fait de cocher un autre bouton radio décoche le premier.
- **checkbox**: Permet de créer une case à cocher. C'est un petit carré qu'on peut cocher ou décocher. Il est destiné à faire des choix multiples.
- **file**: Permet de créer le champ de chargement de fichier (comme une pièce jointe d'un email).
- **hidden**: Pour créer un champ caché. C'est comme une zone de texte mais qui n'est pas visible sur le navigateur. Il sert à stocker provisoirement des données jusqu'à la phase de traitement.

- **button**: Permet de créer un bouton simple. A la base, ce bouton ne fait aucune opération. Il faut donc le programmer (généralement avec Javascript).
- **reset**: Permet de créer un bouton d'annulation. Si le formulaire contient des champs qu'on a déjà saisi, ce bouton permet de tout initialiser.
- **submit**: C'est le bouton d'envoi. C'est lui qui permet de poster le formulaire vers la page définie dans l'attribut action de la balise <form>.

Attribut name : L'attribut name permet de donner un nom au champ de formulaire. Le nom sert d'identifiant du champ et doit être unique.

L'attribut value : sert à définir la valeur par défaut d'un champ. S'il s'agit d'une zone de texte, le fait de déclarer l'attribut value force le navigateur à initialiser ce champ avec le texte faisant office de valeur de l'attribut :

- Si le champ est un bouton, alors le texte de l'attribut value constitue l'étiquette (ou label), C'est le texte visible sur le bouton.
- Pour les boutons radio ou les cases à cocher, le texte de l'attribut value n'est pas visible sur le navigateur mais il est très utile (si on fait appel à des programmes comme Javascript ou PHP).

Attribut size : L'attribut size contient un entier comme valeur. Il définit la largeur du champ en caractères. Par défaut, une zone de texte mesure 20 caractères (cela veut dire qu'on peut voir jusqu'à 20 caractères à la fois même si le texte est plus grand).

Attribut tabindex : permet de définir l'ordre de tabulation. En pratique, si on veut saisir un grand formulaire, on se déplace d'un champ à un autre en appuyant sur la touche "tabulation" du clavier. Par défaut le curseur se déplace dans l'ordre où sont déclarés les champs. On peut rompre cet ordre en définissant les valeurs de tabindex. La valeur est un entier qui commence de 1. Le curseur se déplacera dans l'ordre croissant de tabindex.

Attribut checked : est un attribut booléen. Explicitement il est déclaré comme ceci: checked="true", mais il suffit de déclarer checked tout seul pour que le navigateur sache qu'il est activé. Cet attribut s'applique uniquement sur les boutons radio (type="radio") et les cases à cocher (type="checkbox"). S'il est déclaré alors le bouton est automatiquement coché au chargement de la page.

b.Balise <textarea>

La balise **<textarea>** permet de définir un espace de texte. Un espace de texte est une grande zone de texte qui permet d'écrire des paragraphes entiers.

Les attributs **name** et **tabindex** sont aussi applicables sur l'espace de texte mais, en plus, il dispose des attributs suivants :

- **cols**: définit la largeur en caractères de l'espace de texte. Elle désigne combien de caractère une ligne peut-elle contenir. Le fait de dépasser la largeur définie entraîne un retour automatique à la ligne.
- **rows**: définit la hauteur de l'espace de texte. La hauteur désigne combien de ligne on peut voir à la fois. Si le texte dépasse le nombre de ligues défini, alors une barre de défilement qui permet d'atteindre le reste des lignes apparaît.

c.Balise <select>

La balise **<select>** permet de déclarer une liste de sélection (ou liste déroulante). Elle dispose également des attributs **name** et **tabindex**. Pour remplir cette liste avec les valeurs (appelées options) on doit déclarer une autre balise à l'intérieur, c'est la balise **<option>**. Dans la balise **<option>** on déclare le mot à afficher (en tant qu'élément de la liste).

La balise **<option>** dispose d'un attribut booléen du nom de **selected**. Si cet attribut est déclaré, c'est l'option qui le contient qui sera automatiquement sélectionnée au chargement de la page.

a.Exemple

Voici un exemple de déclaration d'une formulaire en HTML :

```
<form name="inscription" method="post" action="">
  Civilité<br />
  <input type="radio" name="civilite" /> Mlle <br/>
  <input type="radio" name="civilite" /> Mme <br/>
  <input type="radio" name="civilite" checked /> M. <br/>
  Nom et prénom <br/>
  <input type="text" name="nom" value="" /> <br/>
```

```
  Email <br/>
  <input type="text" name="email" value="" /> <br/>
  Login<br />
  <input type="text" name="login" value="" /> <br/>
  Mot de passe <br/>
  <input type="password" name="pass" value="" /> <br/>
  Vous être <br/>
  <select name="niv">
    <option>Etudiant</option>
    <option selected>Fonctionnaire</option>
    <option>Employé au secteur privé</option>
    </select> <br />
  <input type="submit" name="envoyer" value="S'inscrire" />
</form>
```

II. Les formulaires en React

a. Introduction

React utilise des formulaires pour permettre aux utilisateurs d'interagir avec la page Web. React supporte deux types de composants : **les composants contrôlés** et **les composants non contrôlés**. La documentation de React indique :

Dans la plupart des cas, nous recommandons d'utiliser des composants contrôlés pour mettre en œuvre des formulaires. Dans un composant contrôlé, les données du formulaire sont gérées par un composant React. L'alternative est l'utilisation de composants non contrôlés, où les données du formulaire sont gérées par le DOM lui-même.

b. Les composants contrôlés

Dans un composant contrôlé, les données du formulaire sont traitées **par l'état du composant**. L'état du composant sert de "source unique de vérité" pour les éléments d'entrée qui sont rendus par le composant.

Voici un exemple qui récupère les données d'une zone de texte dans l'état du composant :

```
class Form extends React.Component {

  constructor(props) {
    super(props);
```

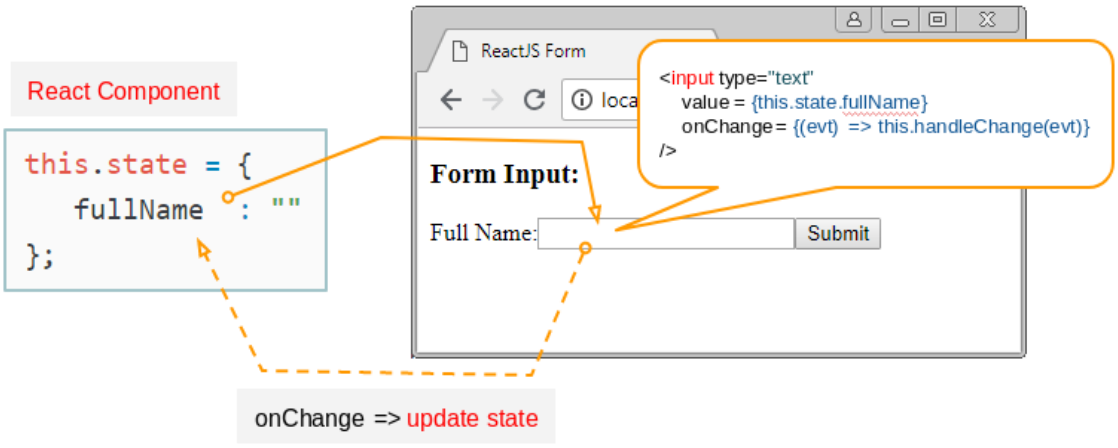
```
this.state = {
      fullname:''
    }

    this.handleSubmit = this.handleSubmit.bind(this);

  }
  handleSubmit(e){
    e.preventDefault();
    console.log("Le texte saisi est : " + this.state.value);
  }

  render(){
    return <div>
      <form onSubmit={this.handleSubmit}>
        <label htmlFor=''>Full name : </label> <br/>
        <input type="text" value={this.state.fullname }
               onChange={(e)=>{this.setState({fullname:
e.target.value})}}/>
        <input type="submit" value="Envoyer les données"/>
      </form>
    </div>
  }
}
```

Dans cet exemple, la valeur de l'élément input est renseigné par la valeur this.state.fullname (Un état géré par le composant). Lorsqu'un utilisateur change la valeur de <input/>, cette valeur doit être mise à jour pour this.state.fullName via la méthode setState().



i. Exemples avec les cases à cocher

Les cases à cocher permet aux utilisateurs de sélectionner plusieurs choix à partir d'un ensemble d'éléments.

Voici un exemple d'utilisation d'une case à cocher dans React :

```
class ExempleCheckBox extends React.Component {
  constructor(props){
    super(props);

    this.handleSubmit = this.handleSubmit.bind(this);
    this.state = {
      isChecked : false
    }
  }

  handleSubmit(e){
    e.preventDefault();
    alert("L'etat de la case a cocher est : " + this.state.isChecked);
  }
  render() {
    return <div>
      <form onSubmit={this.handleSubmit}>
        <label>Choisir : </label>
        <input type="checkbox" checked={this.state.isChecked} onChange={() =>
this.setState({isChecked: !this.state.isChecked})} />
        <input type="submit" value="Soumettre les donnees"/>
      </form>
    </div>
  }
}
```

ii. Exemples avec les radios buttons

Le <input type="radio" /> fonctionne un peu différemment des autres entrées. La propriété **value** est statique et représente l'option à sélectionner. La propriété **name** est dupliqué et doit correspondre à tous les boutons radio qui composent le groupe de boutons radio. La propriété **checked** est introduit par une condition qui détermine si ce bouton particulier est affiché comme actif ou non.

```
class ExempleRadioButton extends React.Component {
  constructor(props) {
    super(props);
```

```
    this.state = {
      choix: "tri"
    }
  }
  render(){
    return <div>
      Choisir un filiere : <br/>
      <label>
        TDI
        <input type='radio' name="filiere" checked={this.state.choix ===
"tdi"} onChange={()=>this.setState({choix: "tdi"})}/>
      </label>

      <label>
        TRI
        <input type='radio' name="filiere" checked={this.state.choix ===
"tri"} onChange={()=>this.setState({choix: "tri"})}/>
      </label>

      <label>
        Inforgraphie
        <input type='radio' name="filiere" checked={this.state.choix ===
"info"} onChange={()=>this.setState({choix: "info"})}/>
      </label>
    </div>;
  }
}
```

iii. Exemples avec les listes déroulantes

Comme les balises <input/> et <textarea/>, nous utilisons l'évènement onChange pour récupérer la valeur sélectionnée à partir de la liste déroulante à travers l'objet Event.

Dans React, au lieu d'utiliser l'attribut selected, la propriété value est utilisée sur l'élément select racine. Ainsi, vous pouvez définir une valeur par défaut en passant la valeur de l'option dans la propriété value de l'élément select.

Voici quelques exemples d'utilisation de la liste déroulante avec react :

Exemple 1 : Stocker une valeur dans l'état du composant et l'associe à la propriété value de l'élément select.

```
class ExempleFormSelect extends React.Component {
  constructor(props){
    super(props);
```

```
    this.handleSubmit = this.handleSubmit.bind(this);

    this.state = {
      filiere:'RI'
    }
  }

  handleSubmit(e){
    e.preventDefault();
    console.log(this.state.filiere);
  }

  render(){
    return <div>
      <form onSubmit={this.handleSubmit}>
        <label>Choisir votre filiere : </label>
        <select      value={this.state.filiere}
                  onChange={(e) => {this.setState({filiere:
e.target.value}); }}>
          <option value="DI">Developpement informatique</option>
          <option value="RI">Reseaux informatique</option>
          <option value="GE">Gestion des entreprise</option>
          <option value="CO">Comptabilite</option>
          <option value="CG">Controle de gestion</option>
        </select>
        <input type="submit" value="Envoyer les donnees"/>
      </form>
    </div>
  }
}
```

Exemple 2 : Remplir l'élément select en utilisant un tableau d'objets.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      choix : ''
    }

    this.options = [
      { label: "Apple", value: "apple" },
      { label: "Mango", value: "mango" },
      { label: "Banana", value: "banana" },
      { label: "Pineapple", value: "pineapple" },
    ];
```

```
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleSubmit(e){
    e.preventDefault();
    alert(this.state.choix);
  }

  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <select value={this.state.choix} onChange={(e) =>
{this.setState({choix:e.target.value})}}>
            {this.options.map(item => <option key={item.value}
value={item.value}>{item.label}</option>)}
          </select>
          <input value="Envoyer les donnees" type="submit"/>
        </form>
      </div>
    );
  }
}
```

iv. Gérer Multiple saisies

Voici un exemple qui montre comment nous pouvois traiter tous les entrees d'un utilisateur avec une seule fonction handleOnChange :

```
class ExempleMultipleEvent extends React.Component {
  constructor(props){
    super(props);

    this.handleOnSubmit = this.handleOnSubmit.bind(this);
    this.handleOnchange = this.handleOnchange.bind(this);

    this.state = {
      nom:'',
      prenom:'',
      age:'',
      filiere:'',
      vaccin:false
    }
  }
```

```
  handleOnchange(e){
    const value = e.target.value;
    const name = e.target.name;
    const type = e.target.type;

    this.setState({
      [name]: type==="checkbox" ? e.target.checked : value
    });
  }

  handleOnSubmit(e){
    e.preventDefault();
  }

  render(){
    return (
      <div>
        <form onSubmit={this.handleOnSubmit}>
          <label>
            Nom :
            <input name="nom" value={this.state.nom}
onChange={this.handleOnchange}/>
          </label><br/>
          <label>
            Nom :
            <input name="prenom" value={this.state.prenom}
onChange={this.handleOnchange}/>
          </label><br/>
          <label>
            Age :
            <input name="age" value={this.state.age}
onChange={this.handleOnchange}/>
          </label><br/>
          <label>
            Filiere :
            <input name="filiere" type="radio" value="TDI"
checked={this.state.filiere==="TDI"} onChange={this.handleOnchange}/> TDI
            <input name="filiere" type="radio" value="TRI"
checked={this.state.filiere==="TRI"} onChange={this.handleOnchange}/> TRI
          </label><br/>
          <label>
            Vaccine :
            <input type="checkbox" name="vaccin" checked={this.state.vaccin}
onChange={this.handleOnchange}/>
          </label><br/>
          <input type="submit" value="Soumettre les donnees"/>
        </form>
      </div>
    );
```

```
  }
}

class ExempleRadioButton extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      choix: "tri"
    }
  }
  render(){
    return <div>
      Choisir un filiere : <br/>
      <label>
        TDI
        <input type='radio' name="filiere" checked={this.state.choix ===
"tdi"} onChange={()=>this.setState({choix: "tdi"})}/>
      </label>

      <label>
        TRI
        <input type='radio' name="filiere" checked={this.state.choix ===
"tri"} onChange={()=>this.setState({choix: "tri"})}/>
      </label>

      <label>
        Inforgraphie
        <input type='radio' name="filiere" checked={this.state.choix ===
"info"} onChange={()=>this.setState({choix: "info"})}/>
      </label>
    </div>;
  }
}
```

c. Les composants non contrôlés

Les composants non contrôlés se comportent davantage comme des éléments de formulaire HTML traditionnels. Les données de chaque élément de saisie sont stockées dans le DOM, et non dans le composant. Au lieu d'écrire un gestionnaire d'événements pour toutes vos mises à jour d'état, vous utilisez une référence pour récupérer les valeurs du DOM.