TD5 / TME 5 – React (2)

TD5 / TME 5 : React (2)

Ce TD / TME a pour objectif d'expérimenter quelques *hooks* vus lors du cours sur React :
- un retour sur `userState`
- `useEffect`
- `useRef`
- Gestion des formulaires

Il n'est pas directement lié au projet Organiz'Asso mais a pour but de vous familiariser avec les *hooks* dans des cas simples, afin d'être plus efficaces pour le projet.

5.1 Les états

Revenons sur les états, que vous avez pu expérimenter la semaine dernière.
Nous allons créer une mini-bibliothèque pour afficher quelques livres avec leur titre, le nom de leur auteur, et leur statut d'emprunt, et permettre d'en ajouter.

5.1.1 Initialisation du *state*

Dans le composant principal :
- créer deux variables `dateAuj` et `dateE` initialisées respectivement à la date courante et au 19 février 2023
- créer deux *states* nommés `titreCollection` et `livres`, initialisés respectivement à une chaîne de caractères vide et au tableau

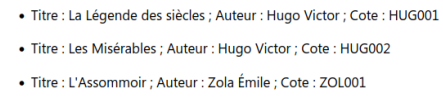
```
[
        {auteur: "Hugo Victor", titre: "La Légende des siècles", emprunt: {statut: false,
        ↪  dateEmprunt:dateAuj}, cote: "HUG001"},
        {auteur: "Hugo Victor", titre: "Les Misérables", emprunt: {statut: false,
        ↪  dateEmprunt:dateAuj}, cote: "HUG002"},
        {auteur: "Zola Émile", titre: "L'Assommoir", emprunt: {statut: true, dateEmprunt:dateE}, cote:
        ↪  "ZOL001"}
]
```

5.1.2 Utilisation de `titreCollection`

Insérer un champ de saisie permettant de changer dynamiquement le titre de la page (stocké dans `titreCollection` et affiché dans un élément `h1`) comme sur cette capture d'écran :

5.1.3 Affichage des livres

Créer un composant Livre, appelé avec trois props `auteur`, `titre` et `cote`, et permettant d'obtenir le résultat suivant :



```
import './Livre.css';

function Livre(props) {

  return (
                <li><p>Titre : {props.titre} ; Auteur : {props.auteur} ; Cote :
                ↪  {props.cote}</p>        </li>
  );
}

export default Livre;
```

... et pour l'utiliser :

```
<ul>
        {livres.map( (livre) => <Livre auteur={livre.auteur} titre={livre.titre} cote={livre.cote}
        ↪  emprunt={livre.emprunt} />
        ) }
</ul>
```

5.1.4 Affichage conditionnel

Compléter le composant `Livre` pour afficher si le livre est disponible (statut de l'emprunt à `false`) ou s'il a été emprunté et si c'est le cas, depuis quelle date. Utiliser une classe pour différencier visuellement les deux états possibles :

- Titre : La Légende des siècles ; Auteur : Hugo Victor ; Cote : HUG001

  Livre disponible
- Titre : Les Misérables ; Auteur : Hugo Victor ; Cote : HUG002

  Livre disponible
- Titre : L'Assommoir ; Auteur : Zola Émile ; Cote : ZOL001

  Livre emprunté le 19/02/2023

Livre.jsx :

```
import './Livre.css';

function Livre(props) {

  return (
                <li><p>Titre : {props.titre} ; Auteur : {props.auteur} ; Cote : {props.cote}</p>
                        <p className={"emprunt"+props.emprunt.statut}>{props.emprunt.statut? "Livre
                        ↪  emprunté le "+props.emprunt.dateEmprunt.toLocaleDateString():"Livre
                        ↪  disponible"}</p>
                </li>

  );
}

export default Livre;
```

Livre.css. Utiliser directement `emprunttrue` et `empruntfalse` est certes inélégant, mais efficace !

```
.empruntfalse  {color: green;}

.emprunttrue {color: red;}
```

5.1.5 Ajout d'un livre

Ajouter un formulaire permettant d'ajouter un nouveau livre (auteur, titre et cote), non emprunté.

Les laisser chercher un peu avant de trouver ou leur donner en expliquant raison et syntaxe de `setLivres([...livres, {...` Il « suffit » de saisir la bonne requête sur un moteur de recherche et en tant que développeurs, c'est une compétence qu'ils doivent travailler.

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

//import Livre from "./Livre1";
import Livre from "./Livre2";

function App() {
        const dateAuj = new Date();
        const dateE = new Date(2023, 1, 19);

        const [titreCollection, setTitreCollection] = useState("");
        const [livres, setLivres] = useState([
                {auteur: "Hugo Victor", titre: "La Légende des siècles", emprunt: {statut: false, dateEmprunt:dateAuj}, cote: "HUG001"},
                {auteur: "Hugo Victor", titre: "Les Misérables", emprunt: {statut: false, dateEmprunt:dateAuj}, cote: "HUG002"},
                {auteur: "Zola Émile", titre: "L'Assommoir", emprunt: {statut: true, dateEmprunt:dateE}, cote: "ZOL001"}
        ]);

        const changeTitre = (evt) => setTitreCollection(evt.target.value);

        const addLivre = (evt) =>{
                evt.preventDefault();
                // pour tester avec des données statiques
//                setLivres([...livres, {auteur: "Barjavel René", titre: "La Nuit des Temps", emprunt: {statut: false, dateEmprunt:dateAuj}, cote: "BAR001"}]);
                setLivres([...livres, {auteur: document.getElementById("chp_auteur").value, titre: document.getElementById("chp_titre").value, emprunt: {statut: false, dateEmprunt:dateAuj}, cote: document.getElementById("chp_cote").value}]);
        }

        return (
        <div>
                <h1>{titreCollection}</h1>
                <label htmlFor="chp_collec">Titre de la collection ?</label><input id="chp_collec" onChange={changeTitre}/>
                <ul>
                        {livres.map( (livre) => <Livre auteur={livre.auteur} titre={livre.titre} cote={livre.cote} emprunt={livre.emprunt} />
                        ) }
                </ul>
                <form onSubmit={addLivre}>
                        <label htmlFor="chp_titre">Titre</label><input id="chp_titre" /><br />
                        <label htmlFor="chp_auteur">Auteur</label><input id="chp_auteur" placeholder="Nom Prénom" /><br />
                        <label htmlFor="chp_cote">Cote</label><input id="chp_cote" placeholder="INIxxx" /><br />
                        <button type="submit">Ajouter un livre</button>
                </form>
        </div>
        );
}

export default App;
```

5.2 Utilisation de `useEffect` et des références

5.2.1 Utilisation de `useEffect`

Créer un composant qui demande la saisie de deux chaînes de caractères, stockées ensuite dans le *state*, dans deux champs de formulaire, et les affiche au fur et à mesure de leur saisie dans la console du navigateur (par exemple `"Première chaîne : TexteSaisiDansLeChamp1 et deuxième chaîne : TexteSaisiDansLeChamp2"`).

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

function App() {
        const [chaine1, setChaine1] = useState ("");
        const [chaine2, setChaine2] = useState ("");

        const changeChaine1 = (evt) =>{
                setChaine1(evt.target.value);
        }

        const changeChaine2 = (evt) =>{
                setChaine2(evt.target.value);
        }

        useEffect ( () => {
                console.log(`Première chaîne : ${chaine1} et deuxième chaîne : ${chaine2}`);
        }, [chaine1, chaine2]);

  return (
    <div>
            <label htmlFor="chp_str1">Première chaîne</label>
            <input id="chp_str1" onChange={changeChaine1} />
                <label htmlFor="chp_str2">Deuxième chaîne</label>
            <input id="chp_str2" onChange={changeChaine2} />
    </div>
  );
}

export default App;
```

5.2.2 Références

Initialiser un compteur à 0 dans le *state*, et ne plus y stocker les deux chaînes de caractères. Ajouter un bouton. Au click sur le bouton, afficher dans la console la valeur du compteur incrémentée et les deux chaînes de caractères en utilisant `useRef`.

```
import { useState, useRef, useEffect } from 'react';

function App() {

        const [compteur, setCompteur] = useState(0);

        const chaine1 = useRef(null);
        const chaine2 = useRef(null);

        useEffect ( () => {
                console.log(`Itération ${compteur} : ${chaine1.current.value} et
                ↪  ${chaine2.current.value}`);
        }, [compteur]);

```

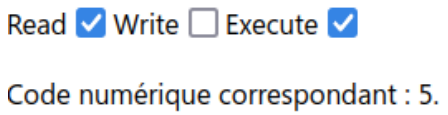
```
        const incrCompteur = (evt) => setCompteur(compteur+1);

        return (
                <div>
                        <label htmlFor="chp_chaine1">Première chaîne</label>
                        <input id="chp_chaine1" ref={chaine1}/>
                        <label htmlFor="chp_chaine2">Deuxième chaîne</label>
                        <input id="chp_chaine2" ref={chaine2}/>
                        <br/>
                        <button onClick={incrCompteur}>Nouvelles chaînes</button>
                </div>
        );
}

export default App;
```

5.3 Formulaires contrôlés

Créer trois cases à cocher permettant de calculer le code numérique correspondant aux trois bits *read*, *write* et *execute* dans la gestion de droits Unix. Le code numérique correspondant est ensuite affiché dynamiquement.



Utiliser un *state* contenant le code numérique d'une part, et un objet `RWX` d'autre part avec trois propriétés (par exemple, dans le cas précédent, `RWX={r: true, w: false, x: true}`).

```
import { useState, useRef, useEffect } from 'react';

function App() {
        const [code, setCode] = useState (0);
        const [RWX, setRWX] = useState({r: false, w: false, x: false});

        const read = useRef(null);
        const writ = useRef(null);
        const execute = useRef(null);

        const changeCode = (evt) => {
                setRWX({r: Boolean(read.current.checked), w: Boolean(writ.current.checked), x:
                ↪  Boolean(execute.current.checked)});
        }

        // Conversion dynamique de type en JS: true converti en 1 et false en 0 pour la
        ↪  multiplication
        useEffect ( () => {setCode(RWX.r*4+RWX.w*2+RWX.x); }, [RWX]);

        return (
        <div>
                <label htmlFor="read" lang="en">Read</label><input type="checkbox" id="read"
                ↪  checked={RWX.r} onChange={changeCode} ref={read}/>
                <label htmlFor="writ" lang="en">Write</label><input type="checkbox" id="writ"
                ↪  onChange={changeCode} checked={RWX.w} ref={writ}/>
                <label htmlFor="execute" lang="en">Execute</label><input type="checkbox" id="execute"
                ↪  onChange={changeCode} checked={RWX.x} ref={execute} />
            <p>Code numérique correspondant : {code}.</p>
        </div>
```

```
        );
}

export default App;
```

5.4 Organiz'Asso

Appliquer ces notions aux composants que vous avez définis lors du TME4. Il n'y a pour le moment pas d'initialisation des données avec la base de données, donc prévoir des initialisations « en dur » lors de la définition des états.