TD4 – React

Introduction : installation et utilisation de React

Pour créer et initialiser un projet React, en ligne de commande :

1. tapez `npm init vite@latest app-react -- --template react` (les doubles `--` ne sont pas une faute de frappe)
2. `cd app-react`
3. `npm install`
4. `npm run dev`

Attention ! Sur les machines de la PPTI, il faut systématiquement saisir non pas npm, mais /bin/npm, et /bin/node au lieu de node.
Par exemple, vous taperez `/bin/npm init vite@latest app-react -- --template react`

Cela lancera un serveur qui fera tourner React. Vous avez alors accès à quelques commandes, dont vous pourrez consulter la liste en tapant `h`. Les deux plus utiles sont `o`, qui ouvre une fenêtre de votre navigateur sur le site sur lequel vous travaillez, et `q` qui arrête le serveur.
Le fichier `index.html` contient un appel vers `src/main.jsx` , qui lui-même appelle le composant stocké dans `src/App.jsx`. Vous pouvez ensuite soit modifier le composant App par défaut, soit en créer un autre que vous appellerez dans main.jsx.

1. Créer un mini-projet nommé react-intro
2. Créer un composant nommé Test, dans un fichier Test.jsx. Ce composant affiche uniquement le paragraphe

   ```
   <p>Ceci est un paragraphe.</p>
   ```

3. Modifier le fichier main.jsx pour qu'il appelle et affiche le composant Test
4. Ajouter dans Test une propriété `texte` qui permet d'ajouter une phrase au paragraphe
5. Dans main.jsx, utiliser cette propriété pour obtenir le code HTML suivant :

   ```
   <p>Ceci est un paragraphe. Et une nouvelle phrase.</p>
   ```

Fichier main.jsx :

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import Test from './Test.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <Test texte=" Et une nouvelle phrase."/>
  </StrictMode>,
)
```

Fichier Test.jsx :

```
function Test(props){
        return (
                <p>Ceci est un paragraphe.{props.texte}</p>
        )
}

export default Test;
```

Exercice 1 – Mon premier composant React : Le jeu de cartes

Q 1.1 On souhaite créer un composant `Cards` qui permet d'afficher le symbole de la carte. Ce composant renvoie un élément `div` possédant la classe "Cards". Son contenu sera la valeur de la propriété `symbol` des `props`.

!!!! `class` est le mot réservé à la définition des classes en JavaScript. Pour la classe CSS, on utilise alors « `className` ».

```
function Card(props) {
    return <div className="Cards"> {props.symbol} </div>;
}

export default Card;
```

Ne pas oublier de signaler la nécessité d'exporter le composant.

Q 1.2 À quoi correspond `props` ?

Ce sont les valeurs des attributs que l'on va récupérer quand on appelle le composant Card :
`<Card symbol="toto"/>`.

Q 1.3 On souhaite créer une variable associée au composant `Card` qui permet de dire si on affiche ou non la carte (variable `affichage` de valeur `visible` ou `hidden` récupérée des propriétés). Quelle est la meilleure façon de faire ? Écrire le code permettant de créer cette variable. Pour le moment, ne pas gérer cet affichage conditionnel.

On crée un état car cette propriété va être modifiée et l'affichage aussi. L'affichage est seulement mis à jour quand l'état du composant change.

```
import { useState } from 'react':
function Card(props) {
    const [affichage, setAffichage] = useState(props.affichage)

    return <div className="Cards"> {props.symbol} </div> ;
}
```

Q 1.4 Modifier le code afin de conditionner l'affichage du composant selon la règle suivante : si l'affichage est `'visible'`, alors on affiche le symbole, sinon on affiche "-".

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

function Card(props) {
    const [affichage, setAffichage] = useState(props.affichage)

    return <div className="Cards"> {affichage === 'visible' ? props.symbol : '-'} </div>;
}
```

Q 1.5 On souhaite rendre le code un petit peu interactif. Quand on clique sur une carte retournée (affichage "-"), on doit pouvoir changer son état et afficher son symbole. Modifier le code du composant `Card`.

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

function Card(props) {
    const [affichage, setAffichage] = useState(props.affichage)
```

```

    function handleCardClick () {
        setAffichage('visible');
    }

    return <button className="Cards" onClick={handleCardClick}> {affichage === 'visible' ? props.symbol
    ↪  : '-'}</button>;
}
```

Exercice 2 – Les listes en React

Q 2.1 Créer un composant `Cardlist` qui permet d'afficher 4 cartes. La première doit être visible, les trois suivantes cachées.

```
function CardList (props) {
        return
                <div className="cardlist">
                        <Card symbol="toto" affichage="visible" />
                        <Card symbol="tata" affichage="hidden" />
                        <Card symbol="titi" affichage="hidden" />
                        <Card symbol="tutu" affichage="hidden" />
                </div>;
}
```

Q 2.2 L'état du composant `Cardlist` stocke une liste de cartes `cartes`. Modifier le composant `Cardlist` pour ajouter son état, et l'utiliser pour afficher la liste des cartes. Pour cela, vous utiliserez la méthode `map()`.

```
import {useState} from 'react'

import Card from './card'

function CardList (props) {
        const [cartes, setCartes] = useState(props.cartes);

        return <div className="cardlist">
                        {
                        props.cartes.map ( (card, index) => (
                                <Card key={index} symbol={card.card} affichage={card.display} />
                        ))
                        }
                </div>;
}

export default CardList;
```

index est un attribut récupéré par défaut qui correspond à l'identifiant du composant dans React.
À ce stade, rappeler que l'élément Card va récupérer les attributs au travers de `props`, même si on a utilisé l'état de la liste pour faire naviguer les « variables ».
Il est généralement conseillé d'ajouter un attribut `key` aux composants pour pouvoir les distinguer. `<Card key={index} .../>`
Rappeler aussi qu'ils peuvent créer un fichier css pour chaque composant où ils vont pouvoir faire la mise en forme de leur composant.

Q 2.3 Comment ce code est-il utilisé dans une application React pour pouvoir être affiché dans le navigateur ?

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
        <React.StrictMode>
                <CardList />
        </React.StrictMode>
  );
```

Le fichier index.html :

```
<html lang="fr">
        <head>
                (...)
        </head>
        <body>
                <div id="root"><!-- React code --></div>
        </body>
</html>
```

Exercice 3 – Formulaire : Création d'une carte

Q 3.1 Créer le composant qui permet de créer une carte avec les champs suivants (il s'agit dans cette question de donner uniquement le code JSX, pas le code JavaScript permettant de créer effectivement une nouvelle carte) :

- Symbole (text)
- Affichage (menu déroulant)
- Bouton de soumission

De la même manière qu'on ne peut pas utiliser directement `class` pour l'attribut HTML du même nom dans le JSX, mais qu'on est obligé d'utiliser `className`, on ne peut pas utiliser directement `for`, mais plutôt `htmlFor`.

```
function FormAddCarte(props) {

    return(
                <form className="FormAddCard">
                        <label htmlFor="chp_symbol">Symbole</label><input type="text" id="chp_symbol" />
                        <label htmlFor="menu_visible">Visibilité</label>
                        <select id="menu_visible">
                                <option value="visible">Visible</option>
                                <option value="hidden">Hidden</option>
                        </select>
                        <input type="button" value="Ajouter une carte" onClick={props.handleAdd}/>
                </form>
    );
}

export default FormAddCarte;
```

Q 3.2 Que faut-il faire pour avoir une page qui affiche la liste des cartes et le formulaire de création de cartes ?

Il faut créer un « super-composant » incorporant les deux composants déjà créés. pageCard.js :

```
import { useState } from 'react';
import CardList from "./cardlist"
import FormAddCarte from "./formaddcarte"
```

```
4
5  function PageCard (props) {
6
7          const [cartes, setCartes] = useState([
8                  { id:1, card: "toto", display: 'visible'},
9                  { id:2, card: "tata", display: 'hidden'},
10                 { id:3, card: "titi", display: 'hidden'},
11                 { id:4, card: "tutu", display: 'visible'}]);
12
13         const addCarte = (evt) =>
14                 {
15                         let newCard = {};
16                         newCard.id=cartes.length+1;
17                         newCard.card=document.getElementById("chp_symbol").value;
18                         newCard.display=document.getElementById("menu_visible").value;
19                         setCartes([... cartes, newCard]);
20                 }
21
22         return(
23                 <div>
24                         <CardList cartes={cartes}/>
25                         <FormAddCarte cartes={cartes} handleAdd={addCarte}/>
26                 </div>
27     );
28 }
29
30 export default PageCard;
```