

Développement front

MMI 2 – TP#3 S4

Danielo **JEAN-LOUIS**
Michele **LINARDI**

Patterns – Patrons de conception

- Ensemble de syntaxes et façons de travailler propre à React
 - Conditions
 - Boucles
 - Requêtes asynchrones
- Permettent d'afficher du contenu

Conditions

- Permet d'afficher des éléments en fonction de conditions définies
- Plusieurs syntaxes possibles

Source(s) :

- <https://fr.reactjs.org/docs/conditional-rendering.html>

Conditions – if classique

```
function Formulaire(props) {  
  const isUserRegistered = props.isUserRegistered;  
  if (isUserRegistered) {  
    return <LogInForm />;  
  }  
  
  return <SignInForm />;  
}
```

On affiche un
composant spécifique
en fonction de la valeur
de la props
"isUserRegistered"

Source(s) :

- <https://fr.reactjs.org/docs/conditional-rendering.html>

Conditions – if ternaire

```
function Message(props) {  
  const isUserRegistered = props.isUserRegistered;  
  return (  
    <div>  
      L'utilisateur <b>{isUserRegistered ? 'a un compte' : 'n'a pas de compte'}</b>.  
    </div>  
  );  
}
```

Le texte s'affiche en fonction de la props "isUserRegistered".
Note : cette syntaxe doit impérativement être dans une balise.

Source(s) :

- <https://fr.reactjs.org/docs/conditional-rendering.html>

Conditions – if avec l'opérateur &&

```
function Message(props) {  
  const [isUserConnected, setIsUserConnected] = React.useState(false);  
  
  const changeStatus = () => {  
    // Cette syntaxe inverse un booléen.  
    // Si c'est vrai, c'est faux. Si c'est faux, c'est vrai.  
    setIsUserConnected(!isUserConnected)  
  }  
  
  return (  
    <>  
      {isUserConnected &&  
        <h2>  
          Bonjour {props.name}.  
        </h2>  
      }  
      <button onClick={changeStatus}>Changer</button>  
    </>  
  );  
}
```

Le h2 ne s'affichera que si et seulement si le state "isUserConnected" est égal à "true"

Source(s) :

- <https://fr.reactjs.org/docs/conditional-rendering.html>

Pratiquons ! - Découvrons ReactJS (Partie 1)

Pré-requis :

- Avoir la ressource `ressources/patterns/if`

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-4/ressources>

Boucles

- Permet d'afficher une liste d'éléments
- Utilisation de la méthode `.map()`
 - Renvoie un nouveau tableau modifié

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- <https://fr.reactjs.org/docs/lists-and-keys.html>

Boucles

Ici on affiche une liste de nombre, chacun est contenu dans une balise ``

```
function NumberList() {  
  const numbers = [1, 2, 3, 4, 5];  
  
  return (  
    <ul>  
      {numbers.map((number) =>  
        <li>  
          {number}  
        </li>  
      )}  
    </ul>  
  );  
}
```

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- <https://fr.reactjs.org/docs/lists-and-keys.html>

Boucles

```
function EmployeesList() {  
  const listEmployees = [{  
    name: "Patrick",  
    job: "Sr. Graphic Designer"  
  }, {  
    name: "Joanne",  
    job: "CEO"  
  }];  
  
  return (  
    <ul>  
      {listEmployees.map((employee) =>  
        <li>  
          {employee.name} - {employee.job}  
        </li>  
      )}  
    </ul>  
  );  
}
```

Ici, notre tableau contient des objets, on accède donc à chaque propriété de chaque objet pour en afficher le contenu

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- <https://fr.reactjs.org/docs/lists-and-keys.html>

Boucles

- Possibilité de faire des boucles dans des boucles

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- <https://fr.reactjs.org/docs/lists-and-keys.html>

Pratiquons ! - Découvrons ReactJS (Partie 1)

Pré-requis :

- Avoir la ressource `ressources/patterns/boucles`

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-4/ressources>

Boucles

! ▶ Warning: Each child in a list should have a unique "key" prop.

Check the render method of `NumberList`. See <https://reactjs.org/link/warning-keys> for more information.

li

NumberList@file:///E:/projects/cours/developpement-front-s4/travaux-pratiques/numero-3/ressources/patterns/boucles/index.temp.html line 16 > injectedScript:4:17

React lève un avertissement

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- <https://fr.reactjs.org/docs/lists-and-keys.html>

Algorithme de résolution

- React ne remplace que les éléments qui ont été modifiés
- Dans les boucles, React a besoin d'un identifiant unique pour chaque élément
 - Identifiant crée par la props "key"

Algorithme de résolution – Props key

- Utilisable sur n'importe quelle balise ou composant
- Permet d'identifier un élément dans une boucle
- Dans chaque boucle la valeur de la props doit être unique

Algorithme de résolution – Props key

```
function NumberList(props) {  
  const numbers = props.numbers;  
  const listItems = numbers.map((number) =>  
    <li key={number.toString()}>  
      {number}  
    </li>  
  );  
  return (  
    <ul>{listItems}</ul>  
  );  
}
```

La props "key" a une valeur unique, ici un numéro. Ceci permet à React d'identifier les éléments.

Algorithme de résolution – Props key

- Préférez la clé "id" de votre objet comme valeur pour la props "key"
 - Si votre jeu de données possède la clé

Pratiquons ! - Découvrons ReactJS (Partie 2)

Pré-requis :

- Avoir la ressource `ressources/patterns/boucles`

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-4/ressources>

Requêtes asynchrones

- Permet de récupérer des ressources externes de façon asynchrones
 - Ne bloque pas l'ensemble lors du chargement
- Utilisation du hook `React.useEffect()` => `{}`

React.useEffect(() => {})

- Hook permettant de créer des effets de bord
- Idéal pour utiliser la fonction fetch()
- Appelé à l'initiation de l'application

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
- <https://fr.reactjs.org/docs/hooks-effect.html>

React.useEffect(() => {})

```
function Example() {  
  const [listItems, setListItems] = useState([]);  
  
  React.useEffect(() => {  
    fetch('monurl.com')  
      .then(function (response) {  
        return response.json();  
      })  
      .then(function (json) {  
        setListItems(json)  
      });  
  }, []);  
  
  /* [...] */  
}
```

Notre React.useEffect fait une requête vers l'url fictive "monurl.com". On récupère un JSON et on le met dans notre state "setListItems".

Le tableau vide indique que notre React.useEffect ne doit être appelé qu'une seule fois. En absence, le contenu serait appelé indéfiniment.

Pratiquons ! - Découvrons ReactJS (Partie 1)

Pré-requis :

- Avoir la ressource `ressources/patterns/fetch`

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-4/ressources>

Requêtes asynchrones – Bonne pratique

- Avoir un state qui indique le chargement
 - Au début le state indique qu'on charge des données
 - Quand les données sont chargées, le state devient faux

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
- <https://fr.reactjs.org/docs/hooks-effect.html>

Pratiquons ! - Découvrons ReactJS (Partie 2)

Pré-requis :

- Avoir la ressource `ressources/patterns/fetch`

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-4/ressources>

Questions ?

