Module	Développer en front-end

Listes et clés

Dans le code suivant, on utilise la méthode `map()` pour prendre un tableau de nombres et doubler leurs valeurs. On affecte le nouveau tableau retourné par `map()` à une variable `doubled` et on l'affiche dans la console :

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => {number * 2});

console.log(doubled);
```

Ce code affiche [2, 4, 6, 8, 10] dans la console.

On peut construire des collections d'éléments et les inclure dans du JSX en utilisant les accolades {}.

Ci-dessous, on itère sur le tableau de nombres en utilisant la méthode JavaScript map(). On retourne un élément <li> pour chaque entrée du tableau. Enfin, on affecte le tableau d'éléments résultant à listItems :

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

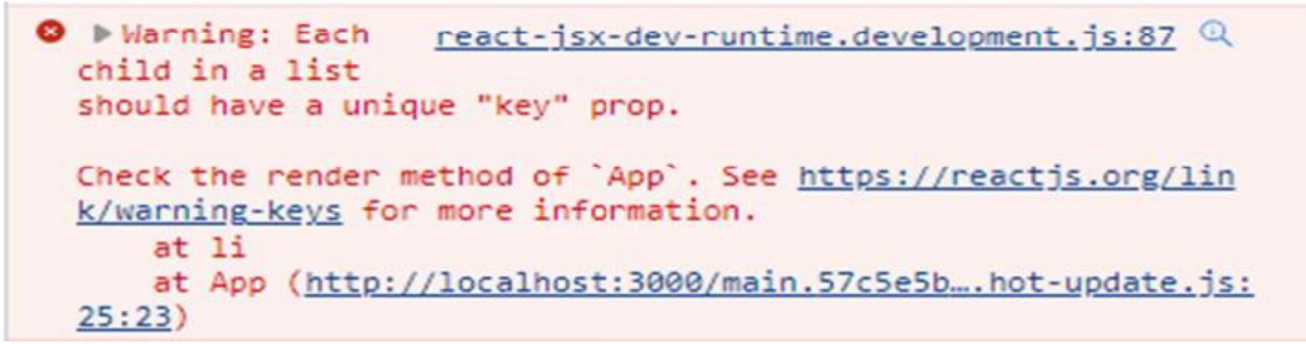
On inclut tout le tableau listItems dans un élément <ul>, et on l'affiche dans le DOM :

```
return(
  <ul>{listItems}</ul>

);
```

En exécutant ce code, vous obtiendrez un avertissement disant qu'une clé devrait être fournie pour les éléments d'une liste. Une « clé » (key, NdT), est un attribut spécial que vous devez inclure quand vous créez une liste d'éléments. Nous verrons pourquoi c'est important dans la prochaine section.

Assignons une key aux éléments de notre liste dans numbers.map() afin de corriger le problème de clés manquantes.



Les clés

Les clés aident React à identifier quels éléments d'une liste ont changé, ont été ajoutés ou supprimés. Vous devez donner une clé à chaque élément dans un tableau afin d'apporter aux éléments une identité stable :

Fichier :Index.js

```
const data=[
       {id:1,name:'name1',age:20},
       {id:2,name:'name2',age:18},
       {id:3,name:'name3',age:25}
      ]
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(

    <App users={data}/>  //function component

);
```

Le meilleur moyen de choisir une clé est d'utiliser quelque chose qui identifie de façon unique un élément d'une liste parmi ses voisins. Le plus souvent on utilise l'ID de notre donnée comme clé :

Fichier :App.js

```
function App(props) {

  const users=props.users;
 //console.log(users);
const myData = users.map(user => <li key={user.id}>
                                 {user.name + ' has '+ user.age+ + ' years old'}
                                 </li>);
//console.log(doubled);
  return (
    <div className="bg-black text-white container">
```

```
    <ul>{myData}</ul>
     </div>
  );
}

export default App;
```