

## React.js useState exercise 8 : DragDropList

1ère étape : Créer le composant DragDropList dont le code est fourni dans l'énoncé de l'exercice. Puis, créer un state pour stocker la liste des items et l'afficher à l'aide de la fonction map dans l'élément ul prévu à cet effet

```
src > exercices > DragDropList.jsx > DragDropList
1  import { useState } from "react";
2
3  const initialItems = ["Item 1", "Item 2", "Item 3", "Item 4", "Item 5"];
4
5  function DragDropList() {
6    const [items] = useState(initialItems);
7
8    return (
9      <div>
10        <ul>
11          {items.map((item, index) => (
12            <li key={index}>{item}</li>
13          ))}
14        </ul>
15      </div>
16    );
17  }
18
19  export default DragDropList;
```

N'oubliez pas d'importer et d'ajouter le composant SearchFilter dans App.jsx

```
src > App.jsx > App
1  import DragDropList from "../exercices/DragDropList"
2
3  function App() {
4
5    return (
6      <>
7        <DragDropList/>
8      </>
9    )
10  }
11
12  export default App
```

### Résultat



- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

2ème étape : Nous créons un state (et son setter) pour y stocker l'index de l'élément en cours de drag&drop

Nous ajoutons la méthode `handleDragStart()` qui permettra d'initialiser l'index de l'item en cours de drag&drop lorsque l'événement `onDragStart` sera déclenché (quand le drag&drop commence)  
Pour rendre le drag&drop possible, il ne faut pas oublier d'ajouter l'attribut `draggable` sur chaque élément li

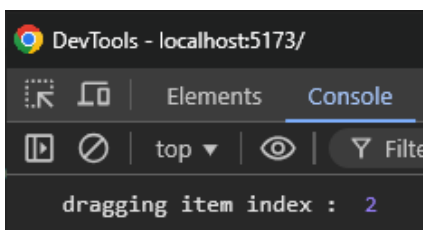
```
src > exercices > DragDropList.jsx > DragDropList
5  function DragDropList() {
6      const [items] = useState(initialItems);
7      const [draggingItem, setDraggingItem] = useState(null);
8
9      function handleDragStart(index) {
10         console.log("dragging item index : ", index)
11         setDraggingItem(index);
12     }
13
14     return (
15         <div>
16             <ul>
17                 {items.map((item, index) => (
18                     <li key={index} draggable
19                         onDragStart={() => handleDragStart(index)}
20                     >
21                         {item}
22                     </li>
23                 ))}
24             </ul>
25         </div>
26     );
27 }
```

Test : déplaçons un (draggable) item sur la page



- Item 1
  - Item 2
  - Item 3
  - Item 4
  - Item 5
- 

Résultat en console



3ème étape : Ajouter la méthode `handleDragEnd()` pour "capturer" la fin du drag&drop (événement `onDragEnd`) et pour réinitialiser le state `draggingItem`

```
src > exercices > DragDropList.jsx > DragDropList
5  function DragDropList() {
13
14      function handleDragEnd(){
15          console.log("dragging ends");
16          setDraggingItem(null);
17      }
18
19      return (
20          <div>
21              <ul>
22                  {items.map((item, index) => (
23                      <li key={index} draggable
24                          onDragStart={() => handleDragStart(index)}
25                          onDragEnd={handleDragEnd}
26                      >
27                          {item}
28                      </li>
29                  ))}
30              </ul>
31          </div>
32      );
33  }
```

Tester le drag&drop et voir le résultat en console.

4ème étape : Ajouter le setter du state `items` pour pouvoir le mettre à jour

```
src > exercices > DragDropList.jsx > DragDropList
5  function DragDropList() {
6      const [items, setItems] = useState(initialItems);
7      const [draggingItem, setDraggingItem] = useState(null);
8  }
```

Puis ajouter la méthode `handleDragOver()` qui permet de mettre à jour le state contenant la liste (`items`) lorsqu'un (draggable) élément (`li`) "passe au-dessus d'un autre" (`over`)

-> voir page suivante

```

src > exercices > DragDropList.jsx > DragDropList
5  function DragDropList() {
19      function handleDragOver(index) {
20          console.log("dragging over item with index : ", index)
21
22          if (draggingItem === null) return;
23          if (draggingItem === index) return;
24
25          const newItems = [...items];
26          const draggingItemValue = newItems[draggingItem];
27          newItems.splice(draggingItem, 1);
28          newItems.splice(index, 0, draggingItemValue);
29
30          setDraggingItem(index);
31          setItems(newItems);
32      }
33
34      return (

```

Dans la fonction `handleDragOver()`, si l'index de l'item en cours de drag&drop correspond à l'index de l'item survolé (ou s'il est null) nous quittons la fonction (ligne 22 et 23) car il n'est pas nécessaire de mettre à jour le state `items` (la liste des items)

Dans le cas contraire, l'index de l'item en cours de drag&drop est différent de l'index de l'item survolé, il faut donc mettre à jour le state `items`.

Nous commençons par faire une copie de la liste `items` que nous stockons dans la variable `newItems` (ligne 25)

Nous récupérons l'élément correspondant à l'index de l'élément en cours de drag&drop dans cette nouvelle liste (ligne 26)

Puis nous mettons à jours l'ordre de la liste `newItems` (ligne 27 et 28) en fonction des index

Enfin, nous mettons à jours les states (ligne 30 et 31), ce qui déclenche un render et met à jours le DOM

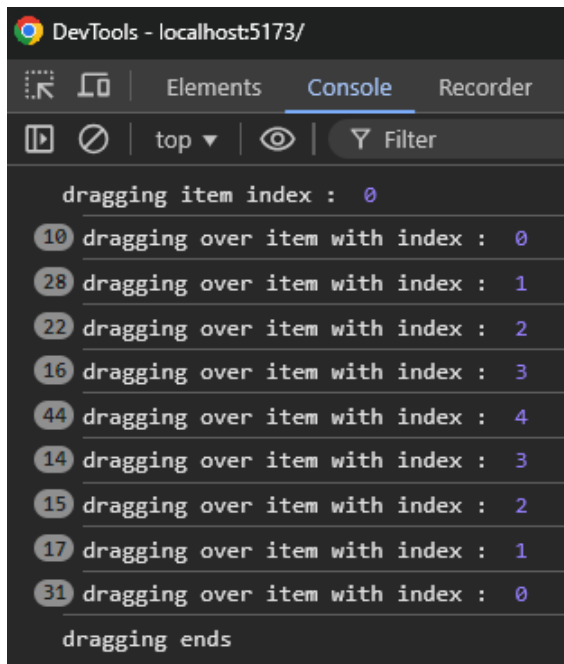
Avant de pouvoir tester, il faut ajouter l'événement `onDragOver` sur les éléments `li`

```

src > exercices > DragDropList.jsx > DragDropList
5  function DragDropList() {
31
32      return (
33          <div>
34              <ul>
35                  {items.map((item, index) => (
36                      <li key={index} draggable
37                          onDragStart={() => handleDragStart(index)}
38                          onDragOver={() => handleDragOver(index)}
39                          onDragEnd={handleDragEnd}
40                      >
41                          {item}
42                      </li>
43                  ))}
44              </ul>
45          </div>
46      );
47  }

```

En testant un drag&drop vous obtiendrez un résultat similaire dans la console

A screenshot of the Chrome DevTools Console. The title bar shows 'DevTools - localhost:5173/'. The 'Console' tab is selected, showing a list of log messages. The messages are: 'dragging item index : 0', '10 dragging over item with index : 0', '28 dragging over item with index : 1', '22 dragging over item with index : 2', '16 dragging over item with index : 3', '44 dragging over item with index : 4', '14 dragging over item with index : 3', '15 dragging over item with index : 2', '17 dragging over item with index : 1', '31 dragging over item with index : 0', and 'dragging ends'. Each message is preceded by a circular icon containing a number, representing the line number in the source code.

```
dragging item index : 0
10 dragging over item with index : 0
28 dragging over item with index : 1
22 dragging over item with index : 2
16 dragging over item with index : 3
44 dragging over item with index : 4
14 dragging over item with index : 3
15 dragging over item with index : 2
17 dragging over item with index : 1
31 dragging over item with index : 0
dragging ends
```

Pour plus de clarté :

Il est possible de supprimer l'événement onDragEnd et donc la méthode handleDragEnd() qui sert uniquement à réinitialiser le state à null et n'est pas indispensable dans le fonctionnement du drag&drop

Nous pouvons renommer les state draggingItem en draggingItemIndex (ainsi que son setter setDraggingItemIndex)

Nous pouvons aussi renommer les paramètres index dans les 2 méthodes handleDragStrat() et handleDragOver() qui correspondent à deux index différents, celui de l'élément déplacé (en cours de drag&drop) ou celui de l'élément survolé (over).

Code complet mis à jour à retrouver sur le git :-)

git : [https://github.com/DWWM-23526/REACT\\_EXERCICES](https://github.com/DWWM-23526/REACT_EXERCICES)