

React.js useState exercice 7 : SearchFilter

1ère étape : Créer le composant SearchFilter dont le code est fourni dans l'énoncé de l'exercice et afficher la liste des items à l'aide de la fonction map dans l'élément ul prévu à cet effet

```
src > exercices > SearchFilter.jsx > SearchFilter
1  const items = ["Apple", "Banana", "Carrot", "Date", "Eggplant",
2
3  function SearchFilter() {
4
5      return (
6          <div>
7              <input type="text" />
8              <ul>
9                  {items.map((item, i) => (
10                     <li key={i}>{item}</li>
11                 ))}
12             </ul>
13         </div>
14     );
15 }
16
17 export default SearchFilter;
```

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

```
src > App.jsx > App
1  import SearchFilter from "../exercices/SearchFilter"
2
3  function App() {
4
5      return (
6          <>
7              <SearchFilter/>
8          </>
9      )
10 }
11
12 export default App
13
```

Résultat



- Apple
- Banana
- Carrot
- Date
- Eggplant
- Fig
- Grape

2ème étape : créer un state (search) pour stocker le texte qui sera saisi dans l'input

```
src > exercices > SearchFilter.jsx > SearchFilter
5 function SearchFilter() {
6   const [search] = useState("");
7
8   return (
9     <div>
10      <input type="text" value={search}/>
11      <ul>
12        {items.map((item, i) => (
13          <li key={i}>{item}</li>
14        ))}
15      </ul>
16    </div>
17  );
18 }
```

3ème étape : Créer une variable (filteredItems) pour stocker la liste filtrée des items en fonction de la valeur du state search (qui contiendra le texte saisi dans l'input)
Puis remplacer la liste originelle (items) par la liste filtrée dans le render

```
src > exercices > SearchFilter.jsx > SearchFilter
4
5 function SearchFilter() {
6   const [search] = useState("");
7
8   const filteredItems = items.filter((item) =>
9     item.toLowerCase().includes(search.toLowerCase())
10  );
11
12   return (
13     <div>
14      <input type="text" value={search}/>
15      <ul>
16        {filteredItems.map((item, i) => (
17          <li key={i}>{item}</li>
18        ))}
19      </ul>
20    </div>
21  );
22 }
```

4ème étape : tester en initialisant le state avec différentes valeurs

```
src > exercices > SearchFilter.jsx > SearchFilter
5 function SearchFilter() {
6   const [search] = useState("e");
7 }
```

le state étant initialisé à avec la valeur "e", il apparaît dans l'input

```
< > ↻ 🏠 localhost:5173
```

- Apple
- Date
- Eggplant
- Grape

Autre valeur par défaut pour le state

```
src > exercices > SearchFilter.jsx > SearchFilter
5  function SearchFilter() {
6    const [search] = useState("an");
7  }
```

Autre résultat

localhost:5173

an

- Banana
- Eggplant

5ème étape : Ajouter la gestion de l'événement onChange sur l'input pour pouvoir y saisir quelque chose et mettre à jour la liste filtrée "automatiquement"

```
src > exercices > SearchFilter.jsx > SearchFilter
5  function SearchFilter() {
6    const [search, setSearch] = useState("");
7
8    function handleInputChange(event) {
9      setSearch(event.target.value);
10   }
11
12   const filteredItems = items.filter((item) =>
13     item.toLowerCase().includes(search.toLowerCase())
14   );
15
16   return (
17     <div>
18       <input type="text" value={search} onChange={handleInputChange}/>
19       <ul>
```

Nous avons ajouté la méthode "setter" (setSearch) du useState (ligne 6)

Nous avons créé la méthode handleInputChange() qui met à jour le state (grâce au setter) en fonction de la valeur saisie dans l'input (ligne 7 à 9)

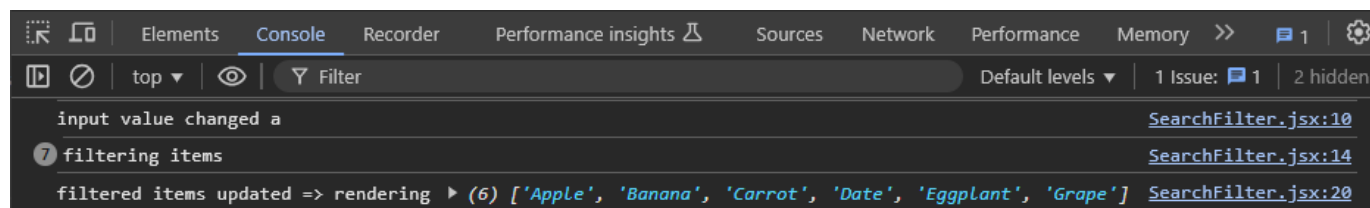
Nous avons ajouté l'événement onChange sur l'input (ligne 17)

Bonus : pour suivre l'ordre d'exécution du code, nous pouvons ajouter des console.log

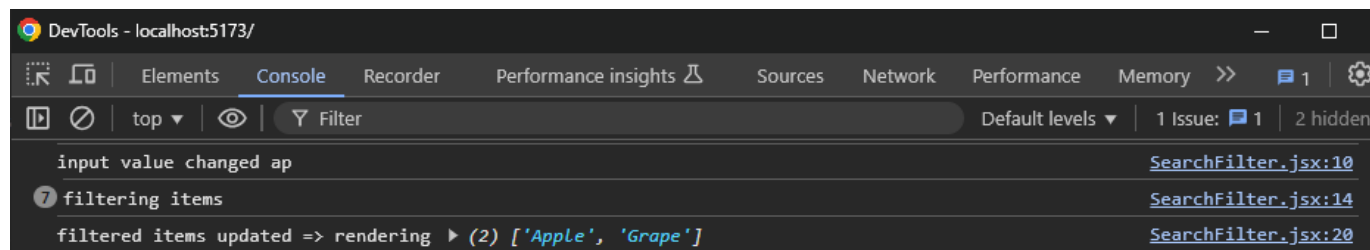
```
src > exercices > SearchFilter.jsx > SearchFilter
5  function SearchFilter() {
6    const [search, setSearch] = useState("");
7
8    function handleInputChange(event) {
9      setSearch(event.target.value);
10     → console.log("input value changed", event.target.value);
11   }
12
13   const filteredItems = items.filter((item) => {
14     → console.log("filtering items");
15     return item.toLowerCase().includes(search.toLowerCase());
16   });
17
18   return (
19     <div>
20     → {console.log("filtered items updated => rendering", filteredItems)}
21     <input type="text" value={search} onChange={handleInputChange} />
```

Tests :

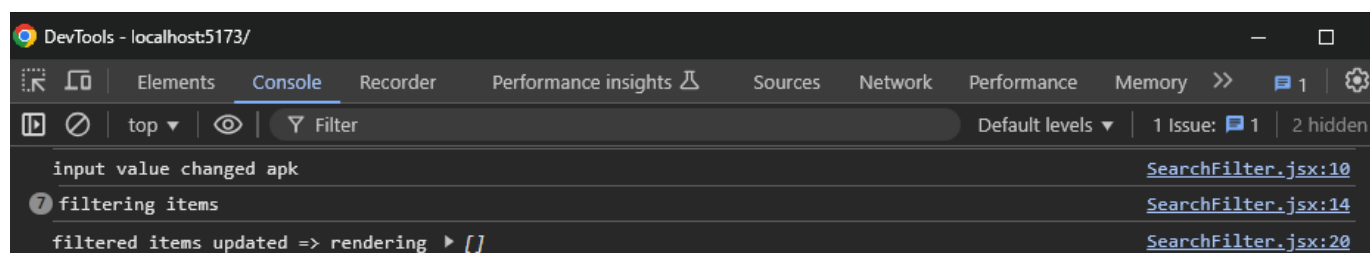
En tapant successivement les lettres a, p et k dans l'input "search" nous obtenons les résultats suivant dans la console



```
input value changed a
7 filtering items
filtered items updated => rendering ▶ (6) ['Apple', 'Banana', 'Carrot', 'Date', 'Eggplant', 'Grape']
```



```
input value changed ap
7 filtering items
filtered items updated => rendering ▶ (2) ['Apple', 'Grape']
```



```
input value changed apk
7 filtering items
filtered items updated => rendering ▶ []
```

git : https://github.com/DWWM-23526/REACT_EXERCICES