

## Fiche d'investigation de fonctionnalité

<b>Fonctionnalité :</b> Recherche	<b>Recherche principale</b>
-----------------------------------	-----------------------------

**Problématique :**

Pour tout algorithme important qu'on développe, on a pour habitude d'en faire deux implémentations différentes pour pouvoir comparer leurs performances et choisir la meilleure.

**Option 1 : Input Search V1 - méthode fonctionnelle**

Dans cette option, on utilise une programmation fonctionnelle en manipulant des objets array(filter, map, some ...) afin d'optimiser le fonctionnement des algorithmes, ainsi que pour avoir plus de réactivité lors des recherches

**Avantages :**

Algorithme plus rapide  
Fichiers plus léger  
Maintenance plus facile

**Inconvénients :**

Moins lisible

**Informations :**

Une saisie dans la barre de recherche déclenche une recherche parmi : le nom, les ingrédients et la description de la recette. La sélection d'un ou plusieurs tag(s) parmi trois catégories (ingrédients, ustensiles, appareils), déclenche la recherche dans les clés "ingredient", "ustensils" et "appliances" de chaque objet recette.

**Option 2 : Input Search V2 - méthode impérative**

Dans cette option on utilise des boucles natives (for) pour la recherche principale

**Avantages :**

Code plus lisible

**Inconvénients :**

Moins rapide

**Informations :**

Séquence d'instructions composée :

de boucles `for`

de structures conditionnelles `if`

**Solution retenue :**

Bien que l'échantillon de 50 recettes soit assez réduit pour comparer de manière fiable les différences de performances entre les deux algorithmes, l'approche fonctionnelle (option 1) semble plus efficace que l'approche impérative. Nous retenons donc l'option 1,

JSBEN.CH

BENCHMARKBROWSEDONATE

Les petits plats

Setup block (useful for function initialization, it will be run before every test, and is not part of the benchmark)

```

1817     quantity: 500,
1818     unit: "grammes",
1819   },
1820 ],
1821   time: 60,
1822   description:
1823     "Préparer la frangipane : Mélanger le sucre la poudre d'amande
1824   applique: "Four",
1825   ustensils: ["rouleau à pâtisserie", "fouet"],
1826 },
1827 ];
1828 const value="mix";

```

ADD LIBRARY

boilerplate block (code will be executed before every block and is part of the benchmark, use it for data initializing.)

Search V1

```

1+ function inputSearch(value, recipes) {
2+   // Algorithme V1
3+
4+   let recipesfiltred = [...recipes];
5+
6+   if (value.length >= 3) {
7+     recipesfiltred = recipes.filter(
8+       (recipe) =>
9+         recipe.name.toLowerCase().match(new RegExp(value, "i")) ||
10+        recipe.description.toLowerCase().match(new RegExp(value, "i")) ||
11+        recipe.ingredients.some(ingredient =>
12+          ingredient.toLowerCase().match(new RegExp(value, "i"))

```

Search V2

```

1+ function inputSearch(value, recipes) {
2+   //Algorithme V2
3+   //deuxième algorithme sans filter, find, map, forEach, findindex, include
4+
5+   let recipesfiltred = [];
6+
7+   if (value.length >= 3) {
8+     for (let i = 0; i < recipes.length; i++) {
9+       const recipe = recipes[i];
10+      const nameMatch = recipe.name.toLowerCase().match(new RegExp(value, "
11+      const descriptionMatch = recipe.description
12+      .toLowerCase()

```

result

Search V1 (3014500)

100%

Search V2 (2866331)

95.08%

If you like to donate (Thank you!):

Ethereum (ETH)

Chia (XCH)

Cardano (ADA)

Ravencoin (RVN)

Bitcoin (BTC)

Ripple (XRP)

Litecoin (LTC)

Monero (XMR)

Dogecoin (DOGE)

SOLANA (SOL)

2/3

