

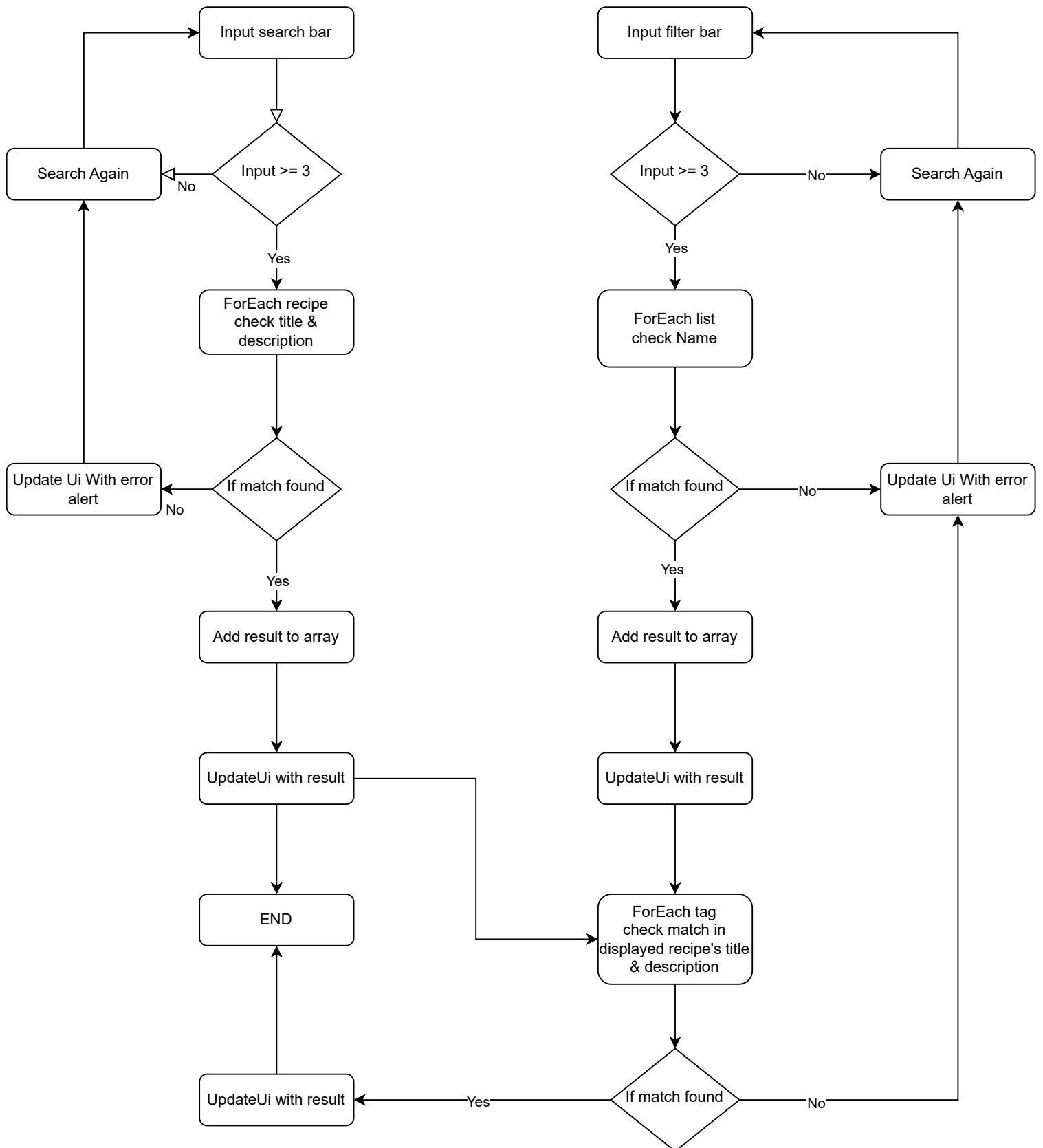
## Fiche d'investigation de fonctionnalité

<b>Fonctionnalité</b> : Filtrer les recettes dans l'interface utilisateur	<b>Fonctionnalité #1</b>
<b>Problématique</b> : Accéder rapidement à une recette correspondant à un besoin de l'utilisateur dans les recettes déjà reçues. La recherche peut s'effectuer en deux étapes via une barre de recherche principale et des tags	
<b>Option 1 : Boucle Native</b> Cette option consiste à utiliser une boucle for pour parcourir la liste de recettes et identifier celles qui correspondent aux critères de recherche de l'utilisateur.	
<b>Avantages</b> <ul style="list-style-type: none"> <li>⊕ Approche simple et facile à comprendre</li> <li>⊕ Bonne performance pour la recherche par saisie utilisateur</li> </ul>	<b>Inconvénients</b> <ul style="list-style-type: none"> <li>⊖ Moins performant pour la recherche par filtres dans les menus déroulants</li> </ul>

<b>Option 2 : Méthode "Filter"</b> Cette option consiste à utiliser la méthode filter() de JavaScript pour filtrer la liste de recettes en fonction des critères de recherche de l'utilisateur.	
<b>Avantages</b> <ul style="list-style-type: none"> <li>⊕ Bonne performance pour la recherche par filtres dans les menus déroulants</li> <li>⊕ Plus concis que l'utilisation d'une boucle for</li> </ul>	<b>Inconvénients</b> <ul style="list-style-type: none"> <li>⊖ Moins performant pour la recherche par input</li> </ul>

<b>Solution retenue :</b> Nous avons donc retenu la solution des boucles utilisant la programmation fonctionnelle en raison de sa lisibilité, de sa maintenabilité. La programmation fonctionnelle offre des avantages significatifs par rapport à l'utilisation des boucles natives, en particulier pour des projets de grande envergure nécessitant un code propre et maintenable. Il serait possible d'utiliser les boucles natives pour les recherches par input qui sont plus performante ici tant que le nombre d'objet à rechercher reste limité.
---

# Algorigramme



# Recherche Principale par Input

JSBEN.CH

BENCHMARKBROWSE

Recherche principale par Input

Description

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

Builerplate block (code will executed before every block and is part of the benchmark. use it for data initating.)

Boucles Natives

```
1 v function searchRecipes() {
2   let searchInputWords = searchHarderInput.value.toLowerCase().split(' ');
3   filteredRecipesByInput = [];
4   for (let i = 0; i < recipes.length; i++) {
5     let recipe = recipes[i];
6     let recipeTitleWords = recipe.name.toLowerCase().split(' ');
7     let recipeDescriptionWords = recipe.description.toLowerCase().split(' ');
8     let titleMatch = false;
9     let descriptionMatch = false;
10
11     for (let inputWord of searchInputWords) {
12       for (let titleWord of recipeTitleWords) {
13         if (titleWord.startsWith(inputWord)) {
14           titleMatch = true;
15           break;
16         }
17       }
18       if (titleMatch) break;
19     }
20   }
21 }
22
```

Boucles for

```
1 v function searchRecipes() {
2   let searchInputWords = escapeHTML(searchHarderInput.value.toLowerCase().split(' '));
3 v filteredRecipesByInput = recipes.filter(recipe => {
4   let recipeTitleWords = recipe.name.toLowerCase().split(' ');
5   let recipeDescriptionWords = recipe.description.toLowerCase().split(' ');
6   let titleMatch = searchInputWords.some(inputWord => recipeTitleWords.some(titleWord => titleWord.startsWith(inputWord)));
7   let descriptionMatch = titleMatch && searchInputWords.some(inputWord => recipeDescriptionWords.some(descriptionWord => descriptionWord.startsWith(inputWord)));
8   return titleMatch || descriptionMatch;
9 });
10
11 userChosenTags.length > 0 ? filterRecipesByTags() : displayFilteredRecipes(filteredRecipesByInput);
12
13 let errorMessage = document.getElementById('error-message');
14 const recipeContainer = document.querySelector('.recipe-container');
15 v if (filteredRecipesByInput.length === 0) {
16   if (errorMessage) {
17     errorMessage = document.createTextNode('p');
18     recipeContainer.id = 'error-message';
19   }
20 }
21
```

result

Boucles Natives (13574960)

100%

Boucles for (1164368)

85.77%

# Recherche de tag par input

Recherche tag par input

Description

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

Builerplate block (code will executed before every block and is part of the benchmark. use it for data initating.)

Boucles Natives

```
1
2 v /**
3  * Recherche les tags en fonction de l'input de l'utilisateur.
4  * @param (string) searchInput - la valeur de l'input de recherche.
5  * @param (string) tagClass - la classe CSS de la liste des tags.
6  * @param (string) dropdown - le type de dropdown (ingrédients, appareils, ustensiles).
7  */
8 v function searchTags(searchInput, tagClass, dropdown) {
9   if (searchInput.length < 3) {
10     let tags = document.querySelectorAll('.item_name');
11     for (let i = 0; i < tags.length; i++) {
12       if (tags[i].closest('.list' + dropdown)) {
13         tags[i].style.display = 'flex';
14       }
15     }
16     return;
17   }
18 }
19
```

Boucles For

```
1
2 v /**
3  * Recherche des tags en fonction de l'entrée de l'utilisateur.
4  * @param (string) searchInput - l'entrée de recherche de l'utilisateur.
5  * @param (string) tagClass - la classe CSS des tags à rechercher.
6  * @param (string) dropdown - le type de dropdown (ingrédients, appareils, ustensiles).
7  */
8 v function searchTags(searchInput, tagClass, dropdown) {
9   let searchInputLower = escapeHTML(searchInput.toLowerCase());
10   let uniqueTags = new Set();
11   let tags = document.querySelectorAll('.item_name');
12   let found = false;
13
14 v if (searchInput.length < 3) {
15   tags.forEach(tag => {
16     if (tag.closest('.list' + dropdown)) {
17       tag.style.display = 'flex';
18     }
19   });
20 }
21
```

result

Boucles Natives (11470425)

100%

Boucles For (11458215)

99.89%

# Recherche avancée par Tag

## Recherche par tag

RUN TESTS

GENERATE PAGE URL

NEW BENCHMARK

Description

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

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

Boucles For

```
1 √ /**
2  * Tableau des tags choisis par l'utilisateur.
3  * @type {Array<string>}
4  */
5  let userChosenTags = [];
6
7 √ /**
8  * Tableau des recettes filtrées.
9  * @type {Array<Object>}
10 */
11 let filteredRecipes = [];
12
13 √ /**
14  * Filtre les recettes en fonction des tags choisis par l'utilisateur.
15  */
16 √ function filterRecipesByTags() {
17     let sourceRecipes = (filteredRecipesByInput || filteredRecipesByInput.length === 0) ? recipes : filteredRecipesByInput;
18 }
```

Boucles Natives

```
1 √ /**
2  * Tableau global pour stocker les tags choisis par l'utilisateur.
3  * @type {Array}
4  */
5  let userChosenTags = [];
6
7 √ /**
8  * Tableau global pour stocker les recettes filtrées par tags.
9  * @type {Array}
10 */
11 let filteredRecipes = [];
12
13 √ /**
14  * Filtre les recettes en fonction des tags choisis par l'utilisateur.
15  */
16 √ function filterRecipesByTags() {
17     filteredRecipes = [];
18     resetDisplayAndPrintTV();
19 }
```

result

Boucles For (14057088) 🚩

100%

Boucles Natives (11381035)

80.96%