

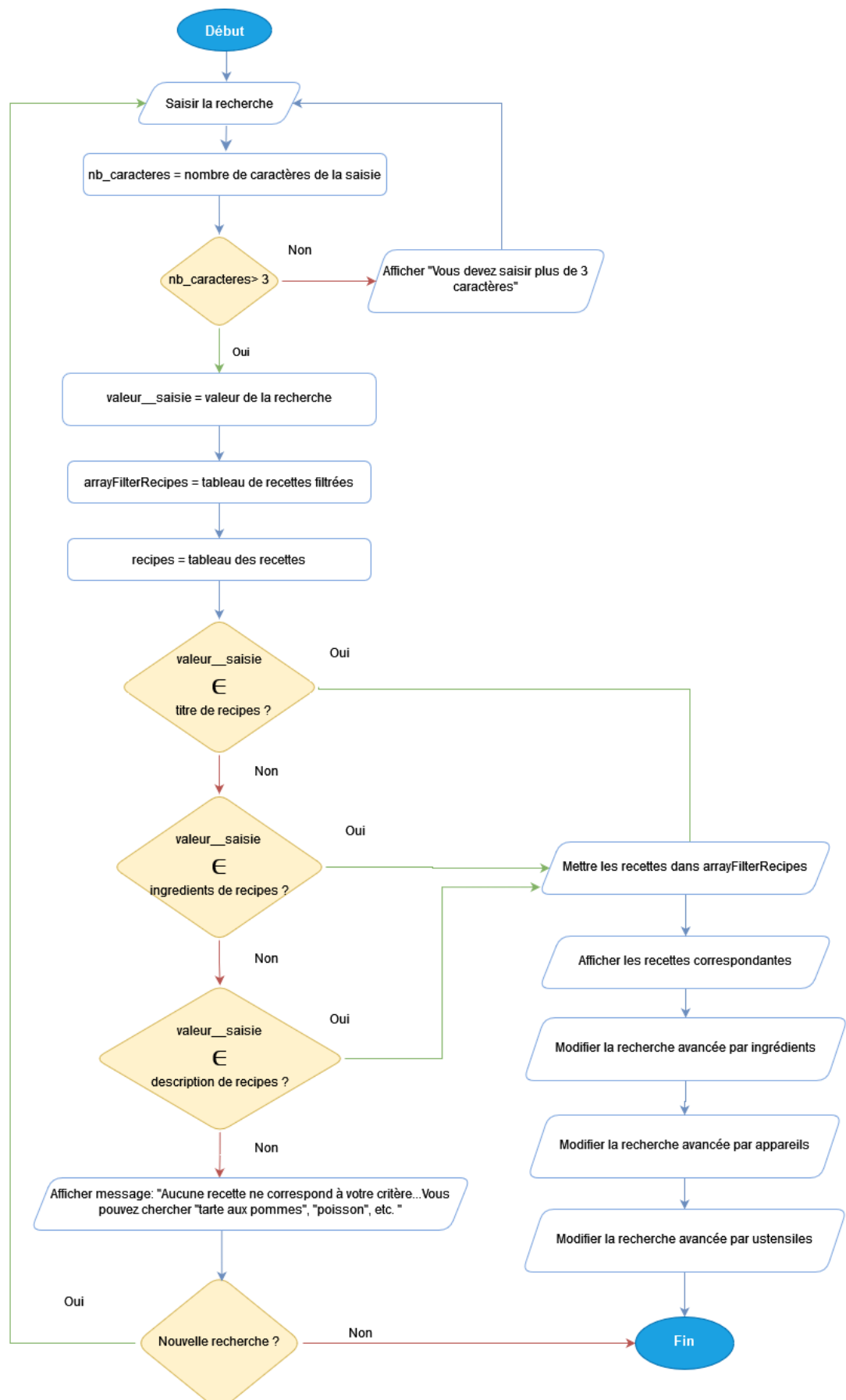
Fiche d'investigation de fonctionnalité

Fonctionnalité : Moteur de recherche	Fonctionnalité #3
Problématique : Afin de se différencier des concurrents sur le marché et retenir un maximum d'utilisateurs, le site « Les petits plats » souhaite mettre en place un moteur de recherche fluide et rapide.	

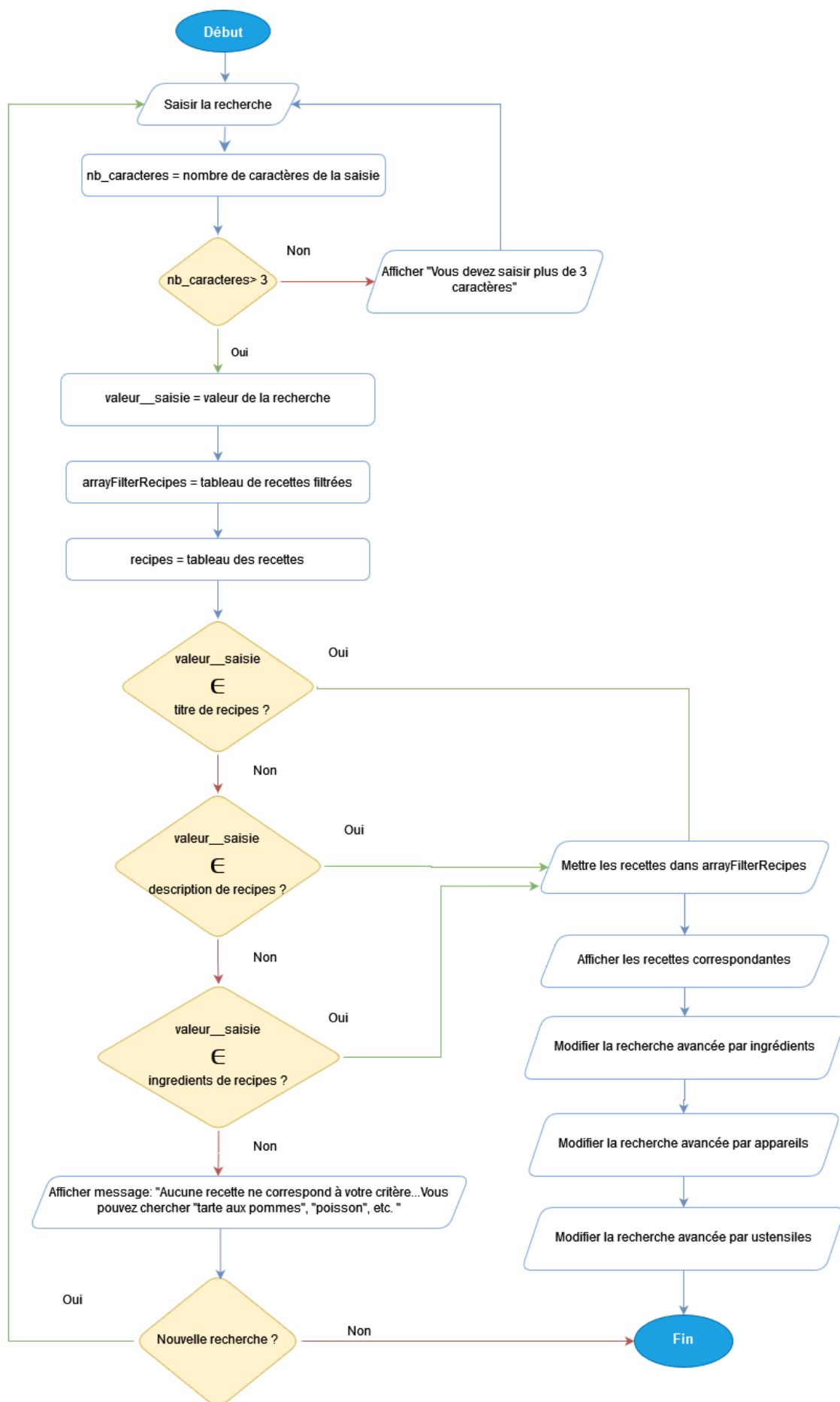
Option 1 : Algorithme basé sur la programmation fonctionnelle (cf. Annexe 1 – Algorithme 1) Cette option utilise les standards contemporains de Javascript (ES6) basés sur les méthodes de l'objet Array (filter, some, every, etc.).	
Avantages <ul style="list-style-type: none"> ⊕ Lisibilité du code ⊕ Moins de risques d'erreurs accidentelles ⊕ Code plus facile à étendre et à maintenir 	Inconvénients <ul style="list-style-type: none"> ⊖ Compatibilité avec les anciens navigateurs ⊖ Moins performant
Nombre de caractères minimum à saisir dans la barre recherche principale : 3 Nombre de champs optionnels : 3 (ingrédients, appareils et ustensiles)	

Option 2 : Algorithme basé sur les boucles natives (cf. Annexe 2 – Algorithme 2) Cette option utilise les boucles natives (for) de Javascript pour parcourir les données. Il s'agit de parcourir le tableau généré depuis le fichier JSON, faisant office de base de données à l'aide de boucles contenant plusieurs instructions afin de déterminer pour chaque recette si elle doit être affichée par rapport aux filtres sélectionnés	
Avantages <ul style="list-style-type: none"> ⊕ Compatibilité avec les anciens navigateurs ⊕ Meilleure performance 	Inconvénients <ul style="list-style-type: none"> ⊖ Lisibilité du code ⊖ Code plus complexe à étendre et à maintenir ⊖ Risques d'erreurs accidentelles
Nombre de caractères minimum à saisir dans la barre recherche principale : 3 Nombre de champs optionnels : 3 (ingrédients, appareils et ustensiles)	

Solution retenue : Suite aux tests réalisés (Annexe 3) entre les 2 algorithmes sur JSBench.me, l'algorithme 2 a été retenu. Deux tests ont été réalisés sur 2 navigateurs différents : Chrome et Firefox. Les résultats des tests donnent l'algorithme 2 comme plus performant. Sous Chrome, l'algorithme 1 est 13,57 % plus lent. Sur Firefox, l'algorithme 1 est 5,95 % plus lent. L'algorithme 2 se démarque par une meilleure performance et une meilleure compatibilité avec les anciens navigateurs.




Annexe 2 : Algorithme 2



Annexe 3 : Résultat du test de performance

Les résultats indiquent que l'algorithme 2 est plus performant.

Résultats du test sur Firefox


Run Please login/register to save & publish tests

enter test suite name v1 - by -

enter test suite description

+ Setup HTML - click to add setup HTML

Setup JavaScript	<pre> }, "time": 60, "description": "Préparer la frangipane : Mélanger le sucre la poudre d'amandes, le beurre et les feuilletée. Mettre au four 30 minutes", "appliance": "Four", "ustensils": ["rouleau à pâtisserie", "fouet"] }] const word = "chocolat"; const valueSearch = word.toLowerCase(); const filterRecipes = (valueSearch) => { const wordToFind = new RegExp("\\b" + valueSearch + "\\b", "gi"); const arrayFilterRecipes = recipes.filter((recipe) => { // Je teste chaque recette return (//Je recherche le mot dans le titre recipe.name.match(wordToFind) // Je recherche le mot dans les ingrédients recipe.ingredients.some((ingredient) => ingredient.ingredient.match(wordToFind))) }) } </pre>
algo1 finished <div> 30962.65 ops/s ± 1.44% 5.95 % slower </div>	
algo2 finished <div> 32923.21 ops/s ± 4.46% Fastest </div>	<pre> function filterRecipes(valueSearch) { const wordToFind = new RegExp("\\b" + valueSearch + "\\b", "gi"); let arrayFilterRecipes = []; for (let i = 0; i < recipes.length; i++) { //Je recherche le mot dans le titre if (wordToFind.test(recipes[i].name)) { arrayFilterRecipes.push(recipes[i]); } // Je recherche le mot dans la description } else if (wordToFind.test(recipes[i].description)) { arrayFilterRecipes.push(recipes[i]); } } } </pre>

+ Test Case - click to add another test case
 + Teardown JS - click to add teardown JavaScript
 + Output (DOM) - click to monitor output (DOM) while test is running

▶ RUN again

[Wiki](#) | [Report issue](#) | [Become a sponsor!](#)

Inspired by [Benchmark.js](#), [Jsperf.com](#) and [JsFiddle.com](#).



Résultats du test sur Chrome

JSBench.Me Run Please login/register to save & publish tests

enter test suite name

v1

- by

enter test suite description

+ Setup HTML - click to add setup HTML

Setup JavaScript

```
    ],  
    "time": 60,  
    "description": "Préparer la frangipane : Mélanger le sucre la poudre d'amandes, le beurre et l  
pate feuilletée. Mettre au four 30 minutes",  
    "appliance": "Four",  
    "ustensils": ["rouleau à pâtisserie", "fouet"]  
  }  
]  
  
const word = "coco";  
const valueSearch = word.toLowerCase();  
  
const filterRecipes = (valueSearch) => {  
  const wordToFind = new RegExp("\\b" + valueSearch + "\\b", "gi");  
  const arrayFilterRecipes = recipes.filter((recipe) => {  
    // Je teste chaque recette  
  
    return (  
      //Je recherche le mot dans le titre  
      recipe.name.match(wordToFind) ||  
      // Je recherche le mot dans les ingrédients  
      recipe.ingredients.some((ingredient) =>  
        ingredient.ingredient.match(wordToFind)  
      )  
    )  
  }  
}
```

algo1

finished

50170.19 ops/s ± 1.51%
13.57 % slower

algo2

finished

58044.44 ops/s ± 4.3%
Fastest

+ Test Case - click to add another test case

+ Teardown JS - click to add teardown JavaScript

+ Output (DOM) - click to monitor output (DOM) while test is running

RUN again

[Wiki](#) | [Report issue](#) | [Become a sponsor!](#)

Inspired by [Benchmark.js](#), [Jsperf.com](#) and [JsFiddle.com](#).