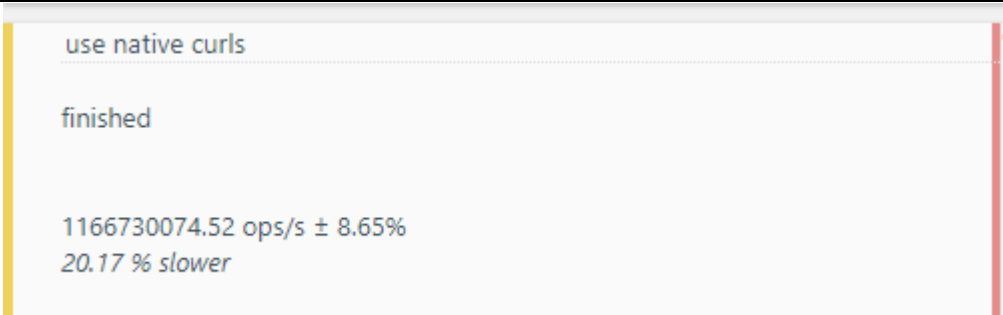
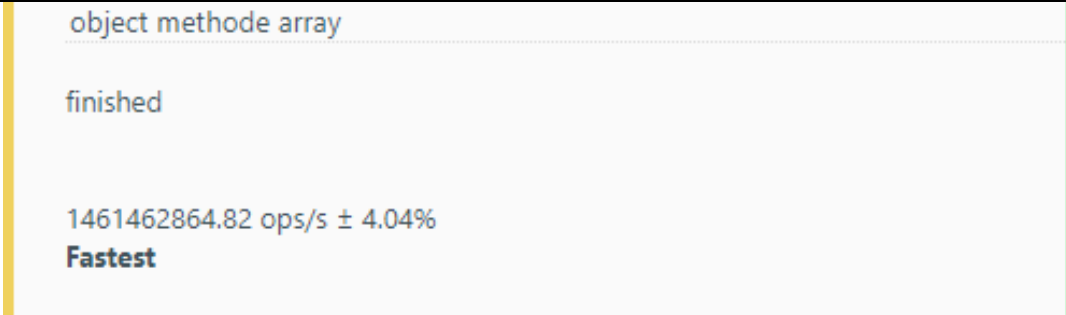


Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche d'une recette	Fonctionnalité #1
Problématique : Afin de pouvoir obtenir un résultat de recherche rapide, nous cherchons à évaluer la Meilleure façon de faire notre recherche.	

Option 1 : boucles natives (while, for...) Dans cette option, nous utilisons uniquement les fonctions natives de Java Script. L'avantage c'est la facilité de mise en place. Le principal inconvénient : c'est la répétition des inscriptions qui allonge la lecture du code.	
Avantages <ul style="list-style-type: none"> ⊕ simple à mettre en place ⊕ code viable dans le temps. 	Inconvénients <ul style="list-style-type: none"> ⊖ répétitions ⊖ Code source peu explicite
	

Option 2 : méthodes de l'objet array (foreach, filter, map, reduce). Dans cette option, nous utilisons uniquement les méthodes fournies par Java Script pour la manipulation des tableaux.	
Avantages <ul style="list-style-type: none"> ⊕ Code source plus lisible et compréhensible. 	Inconvénients <ul style="list-style-type: none"> ⊖ Danger que les méthodes utilisées deviennent obsolètes avec l'évolution du langage
	

Solution retenue : Après un test de performance, nous avons retenus l'approche moderne des méthodes de l'objet array 20% plus rapide que la méthode native
--

Annexes

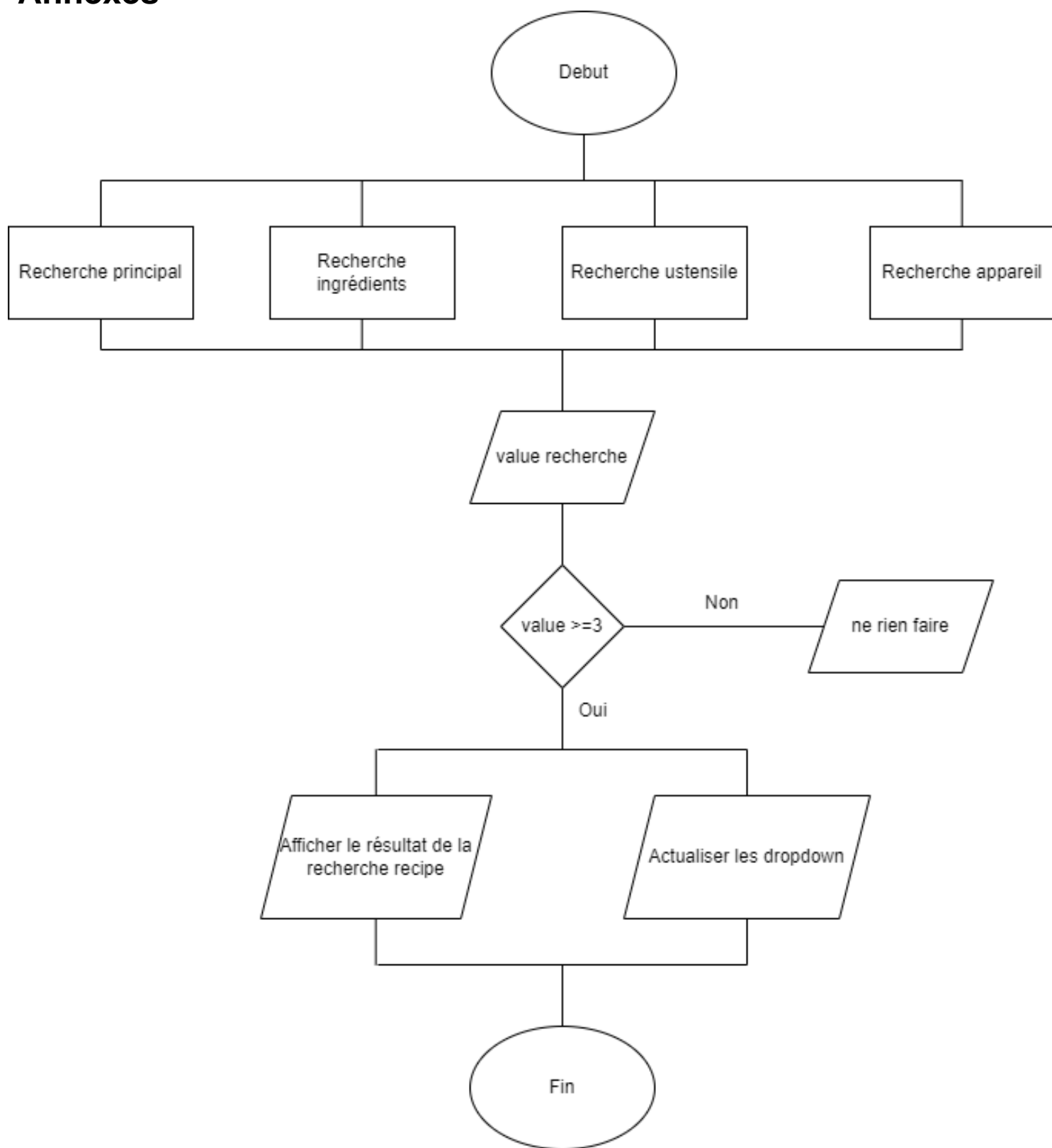


Figure 1 – Algorithme de fonctionnement global

Setup HTML	<pre> <script src= "../scripts/factories/recipes.js"></script> <script src= "../scripts/factories/dropdown.js"></script> <script src= "../scripts/factories/filter.js"></script> <!--Utils--> <script src= "../scripts/utills/sorting.js"></script> <script src= "../scripts/utills/remove-child.js"></script> <!--Page--> <script src= "../scripts/index.js"></script> </body> </html> </pre>
Setup JavaScript	<pre> input.value = ; }; async function init() { await makeAllRecipes(DATA); await creatDropdown(DATA); await addFilter(); await listenInputsearchValue(); await listenInputIngredientsValue(); } init(); </pre>
use native curls finished 1166730074.52 ops/s ± 8.65% 20.17 % slower	<pre> const dataFilter = (valueSearch, data) => { valueSearch = valueSearch.toLowerCase(); const result = []; for (let i = 0; i < data.length; i++) { const nameRecipe = data[i].name.toLowerCase(); const indexRecipe = nameRecipe.indexOf(valueSearch); const description = data[i].description.toLowerCase(); const indexDescription = description.indexOf(valueSearch); if (indexRecipe !== -1) result.push(data[i]); else if (indexDescription !== -1) result.push(data[i]); } } </pre>
object methode array finished 1461462864.82 ops/s ± 4.04% Fastest	<pre> const dataFilter = (valueSearch, data) => { valueSearch = valueSearch.toLowerCase(); let recipes = data.filter(({ name }) => name.toLowerCase().includes(valueSearch)); let recipesByIngredient = data.filter(({ ingredients }) => { let ingredientInRecipe = ingredients.filter(({ ingredient }) => ingredient.toLowerCase().includes(valueSearch)); if (ingredientInRecipe.length > 0) return data; }); } </pre>

Figure 2 : résultat du JSBench