


Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche principal	Fonctionnalité#1
<p>Problématique : Filtrer les recettes qui correspondent à un besoin de l'utilisateur dans les recettes déjà reçues. La barre principale permet de rechercher des mots ou groupes de lettres dans le titre, les ingrédients ou la description.</p>	
<p>option 1: Native Dans cette méthode, pour manipuler les tableaux nous allons utiliser les boucles itératives natives comme for</p>	
<p>Avantages</p> <ul style="list-style-type: none"> ★ Utiliser les connaissances de base, pour créer un algorithme de recherche. 	<p>Inconvénients</p> <ul style="list-style-type: none"> ★ Réinventer la roue, et refaire quelque chose qui est déjà implémentée. ★ Code plus long
<p>Nombre de champs : 1 champ de recherche Nombre de champs minimum à remplir : 0 (afficher toutes les recettes)</p>	
<p>Option 2 : Fonctionnelle Dans cette méthode, nous allons utiliser les méthodes avancées pour manipuler les tableaux.</p>	
<p>Avantages</p> <ul style="list-style-type: none"> ★ Ecrire du code plus lisible. ★ Efficacité et gain en productivité. 	<p>Inconvénients</p> <ul style="list-style-type: none"> ★ Avoir une connaissance des méthodes de manipulation des tableaux (filter, some, sort...) en js et comprendre leur fonctionnement.
<p>Nombre de champs: 1 champ de recherche Nombre de champs minimum à remplir : 0 (afficher toutes les recettes)</p>	
<p>Solution retenue : En se basant sur les résultats obtenus par les outils de comparaison de performance, les deux méthodes sont performantes. Pour des raisons d'efficacité et de productivité, je vais garder la deuxième option qui est la méthode de l'objet array (Fonctionnelle)</p>	

Mesure de performance des deux méthodes

Pour mesurer les performances d'un code js, on peut faire le test manuel avec du code en utilisant `console.time` et `console.timeEnd`.

Terme recherchée : banane  127.0.0.1:5501/?search=banane

Calcul de temps d'exécution en ms		
	Native	Fonctionnelle
Essai 1	Algorithme 1 : 87 ms - chronomètre arrêté	Algorithme 2 : 73 ms - chronomètre arrêté
Essai 2	Algorithme 1 : 115 ms - chronomètre arrêté	Algorithme 2 : 82 ms - chronomètre arrêté
Essai 3	Algorithme 1 : 65 ms - chronomètre arrêté	Algorithme 2 : 61 ms - chronomètre arrêté

Le temps d'exécution du code est variable pour la même méthode, les résultats sont peu concluants.

On peut aussi utiliser des outils de comparaison de performance tels que :

★jsBen.ch

Test de performance avec jsben.ch

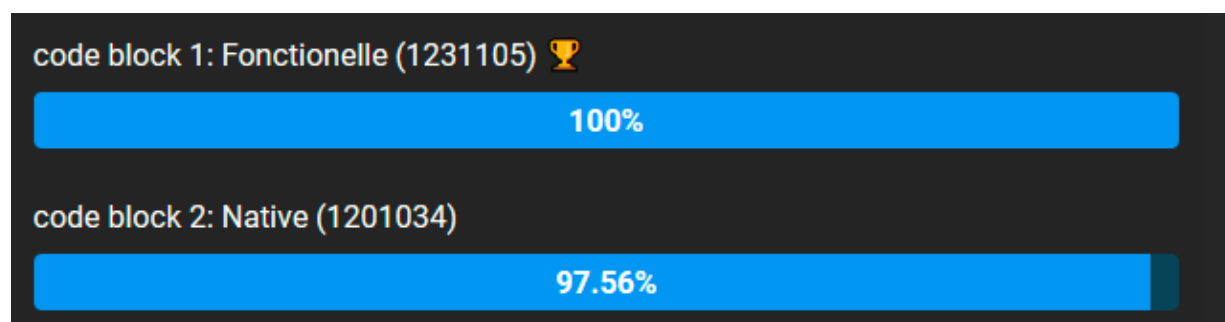


Diagramme de la recherche avancée

