

Fiche d'investigation de fonctionnalité

Fonctionnalité: filtrer les recettes	fonctionnalité #2
Problématique : Afin de se démarquer des sites concurrents et de pouvoir retenir un maximum d'utilisateurs, les résultats d'une recherche doivent être performants en termes de rapidité et de fluidité.	
Nombre de champ: 1 champ recherche (la barre) : optionnel	
Nombre de sélecteur: 3 sélecteurs de mots clés (ingrédients, appareil, ustensile) : optionnel	
Nombre de champ minimum: 0 champ et 0 sélecteur : affichage de toutes les recettes	

Option 1: Boucles natives (for)	
Pour cette option, utilisation de boucles natives proposées par javascript (for). 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 dont la méthode includes afin de déterminer pour chaque recette si elle doit être affichée par rapport aux filtres sélectionnés.	
Avantage :	Inconvénients :
<ul style="list-style-type: none"> • Marginalement plus compatible 	<ul style="list-style-type: none"> • Moins performant • Plus d'instructions

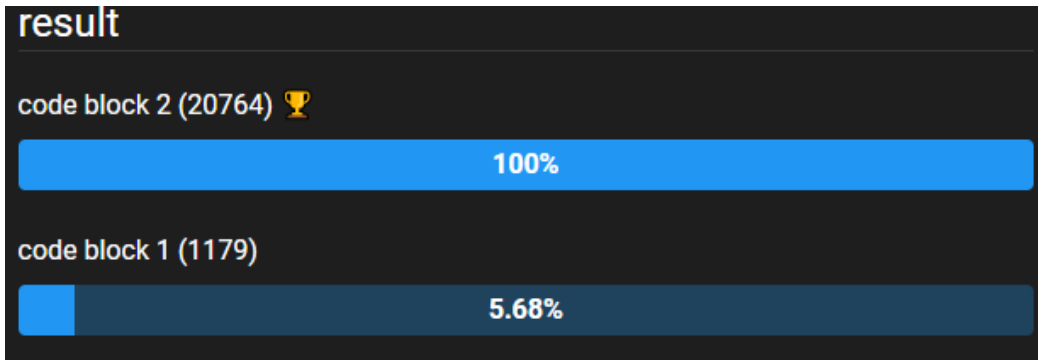
Option 2: Les méthodes de l'objet Array (foreach, map, every,...)	
Pour cette option, utilisation de méthodes de l'objet Array proposées par javascript. Il s'agit de parcourir le tableau généré depuis le fichier JSON faisant office de base de données à l'aide de boucles contenant plusieurs méthodes de l'objet Array dont «foreach, map, every... afin de déterminer pour chaque recette si elle doit être affichée par rapport aux filtres sélectionnés.	
Avantages :	Inconvénients :
<ul style="list-style-type: none"> • Plus performant • Moins d'instructions donc plus maintenable 	<ul style="list-style-type: none"> • Marginalement moins compatible

Solution retenue :
L'option 2 est retenue pour les deux raisons principales suivantes : <ul style="list-style-type: none"> • Les performances sont meilleurs (voir annexe) . • Plus maintenable : moins d'instructions et une syntaxe plus concise.

Annexe Les tests de performances.

1) Utilisation de l'outil de comparaison de performance en ligne jsben.ch.

Avec le fichier JSON comportant les 50 recettes et le mot «hachez» pour le champs de recherche principal.



Option 1 (block 1) : **1179** opérations par seconde
 Option 2 (block 2) : **17864** opérations par seconde

2) Test dans l'environnement de «production»

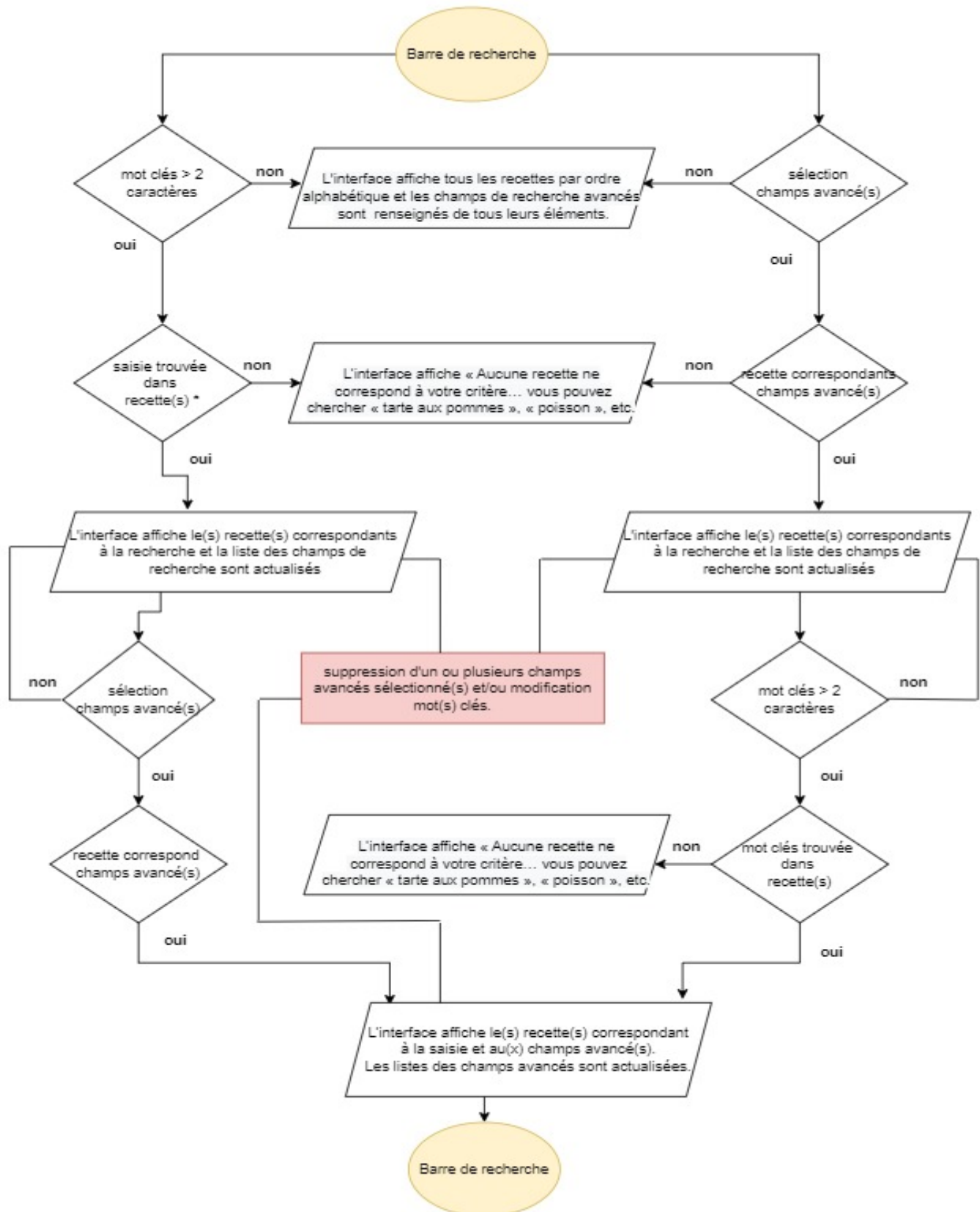
Pour cela, insertion dans le code des méthodes `console.time` en début et `console.timeEnd` en fin de chaque option pour chronométrer leur temps d'exécutions.

coco				Q
algo1:	utils.js:662	algo2:	utils.js:620	
4.996826171875 ms		3.936279296875 ms		
trie-sort:	utils.js:113	trie-quick_sort:	utils.js:113	
1.89208984375 ms		0.125 ms		

ananas				Q
algo1:	utils.js:662	algo2:	utils.js:620	
2.726806640625 ms		1.772216796875 ms		
trie-sort:	utils.js:113	trie-quick_sort:	utils.js:113	
0.809814453125 ms		0.14404296875 ms		

zzzzzzzzzzzz				Q
algo1:	utils.js:662	algo2:	utils.js:620	
5.80810546875 ms		2.18994140625 ms		
trie-sort:	utils.js:113	trie-quick_sort:	utils.js:113	
1.8408203125 ms		0.248046875 ms		

Algorithme de la barre de recherche



* Le système recherche des recettes correspondant au(x) mot(s) clés dans : le titre de la recette, la liste des ingrédients de la recette et la description de la recette.