

## Fiche d'investigation de fonctionnalité

Fonctionnalité : <code>Filtrer les recettes</code>	Fonctionnalité #1
<b>Problématique :</b> Accéder rapidement à une recette correspondant à un besoin de l'utilisateur dans les recettes déjà reçues. Le cas d'utilisation commence lorsque l'utilisateur entre au moins 3 caractères dans la barre de recherche principale.	

Option 1: <code>Approche fonctionnelle map() includes()</code>	
<b>Avantages :</b> <ul style="list-style-type: none"> <li>- Compatible avec tout les navigateurs</li> <li>- Plus facile à lire</li> <li>- Nécessite moins de code</li> <li>- permet de changer la nature des éléments lors du traitement : <code>map()</code> construit un nouveau tableau.</li> <li>- Idéal pour utiliser une valeur de retour</li> </ul>	<b>Inconvénients :</b> <ul style="list-style-type: none"> <li>- N'utilise pas les instructions « <code>break ;</code> » et « <code>continue ;</code> »</li> </ul>
Nombre de boucles nécessaires : 3 Nombre de boucles optionnelles : 2	

Option 2 : <code>Loop for...of</code>	
<b>Avantages :</b> <ul style="list-style-type: none"> <li>- Utilise les instructions « <code>break ;</code> » et « <code>continue ;</code> »</li> </ul>	<b>Inconvénients :</b> <ul style="list-style-type: none"> <li>- N'est pas supportée par Internet Explorer</li> <li>- Moins facile à lire</li> <li>- Nécessite plus de code</li> </ul>
Nombre de boucles nécessaires : 3 Nombre de boucles optionnelles : 2	

<b>Solution retenue :</b> <p>Nous avons donc retenu l'algorithme « <code>Loop for...of</code> » pour ces différentes raisons :</p> <ul style="list-style-type: none"> <li>- Parmi les boucles natives et fonctionnelles testées, la boucle « <code>for...of</code> » est apparue comme l'une des meilleures solutions selon le score obtenu au JavaScript Benchmark.</li> <li>- L'utilisation de la méthode « <code>indexOf()</code> » montre de meilleure performance que les méthodes « <code>include()</code> » et « <code>match()</code> » ;</li> <li>- Mettre la condition concernant le nom de la recette et la description dans le même « <code>if</code> » permet de gagner en performance ;</li> <li>- L'utilisation des instructions « <code>break ;</code> » ou « <code>continue ;</code> » permettent un gain de performance puisque dès qu'une condition est remplie, on passe à l'itération suivante.</li> <li>- La condition « <code>if</code> » placée avant la seconde boucle permet de gagner en performance puisque la boucle parcourt un tableau.</li> </ul>
--