Fiche d'investigation de fonctionnalité

Fonctionnalité : Filtrage par recherche de mots clefs sur un tableau de recettes.

Problèmatique : Il est impératif de trouver la méthode la plus optimisée pour afficher un grand nombre de recettes le plus rapidement possible.

Option 1 : Utilisation de la méthode forEach() + some().
Par cette méthode, on itère pour chaque élement du tableau. On va parcourir l'ensemble du tableau et vérifier dans chaque élement si il répond aux conditions.

Inconvenients :
• Impossible d'arreter la boucle. Elle commence et s'arrete seulement à la fin.
• La boucle ne retourne aucune valeur, elle ne crée aucun nouveau tableau.
• On ne peut pas traiter le même tableau une fois dans la boucle.
• Un peu lent.

Avantages :
• Très simple à utiliser.
• Facile à lire.
• Compatible avec les anciennes version d'EcmaScript

Option 2 : Utilisation de la méthode filter() + some().
Par cette méthode, on va retourner un nouverau tableau selon la condition en paramètre. Lorsque la condition est remplie (true), la méthode ajoute la valeur au nouveau tableau.

Inconvenients :
• Peut être gourmand sur les très grands tableaux.
• N'est pas compatible avant la version EcmaScript 5.

Avantages :
• Plus concis.
• Plus rapide.
• Crée automatiquement un nouveau tableau.

Option 3 : Utilisation de la méthode for() + for() (nested).
On itère sur chaque élement du tableau via une boucle for dans laquelle on va efffectuer plusieurs tests.

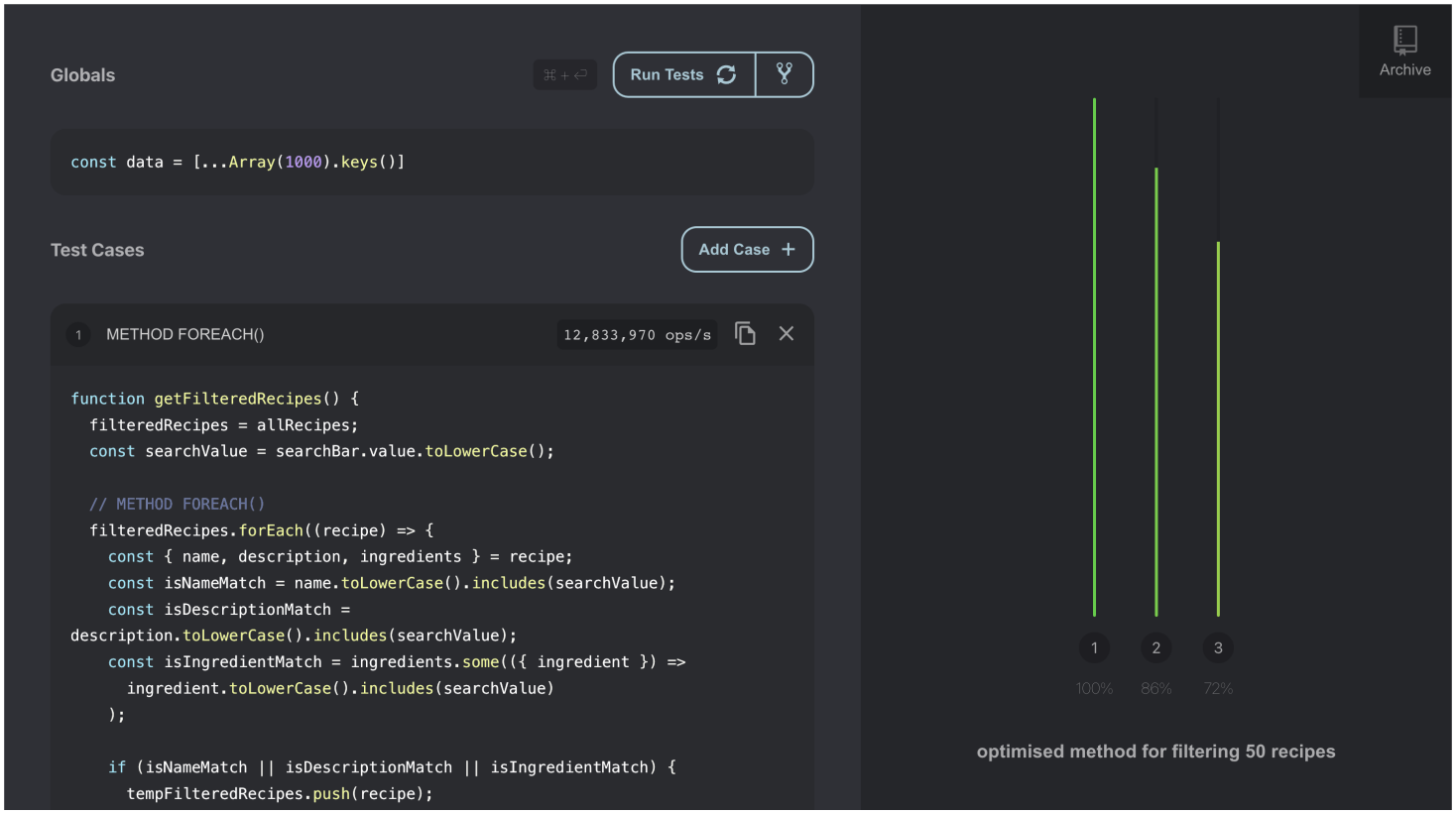
Inconvenients :
• plus verbeux que les autres méthodes.
• N'est pas déclarative. (au premier coup d'oeil on ne devine pas forcèment son utilité).
• Peut être plus lente qu'une méthode de programmation fonctionnelle dans certains cas.

Avantages :
• Contrôle précis. (départ et fin de la boucle avec "i"), choix des conditions plus large,
• Possibilité d'arrêter la boucle avec break.
• Hautement retro-compatible, puisque for() existe depuis la création de JS.

Conclusion :
Puisqu'il faut choisir la méthode la plus optimisée, l'option 2 serait à retenir, même si les benchmarks réalisés sont un peu aléatoires. La moyenne renvoie à l'option 2.
Mais un test de plus grand ampleur avec un tableau beaucoup plus grand devrait être conduit pour s'assurer que la méthode filter() est toujours la plus optimisée.

Resultat benchmarks

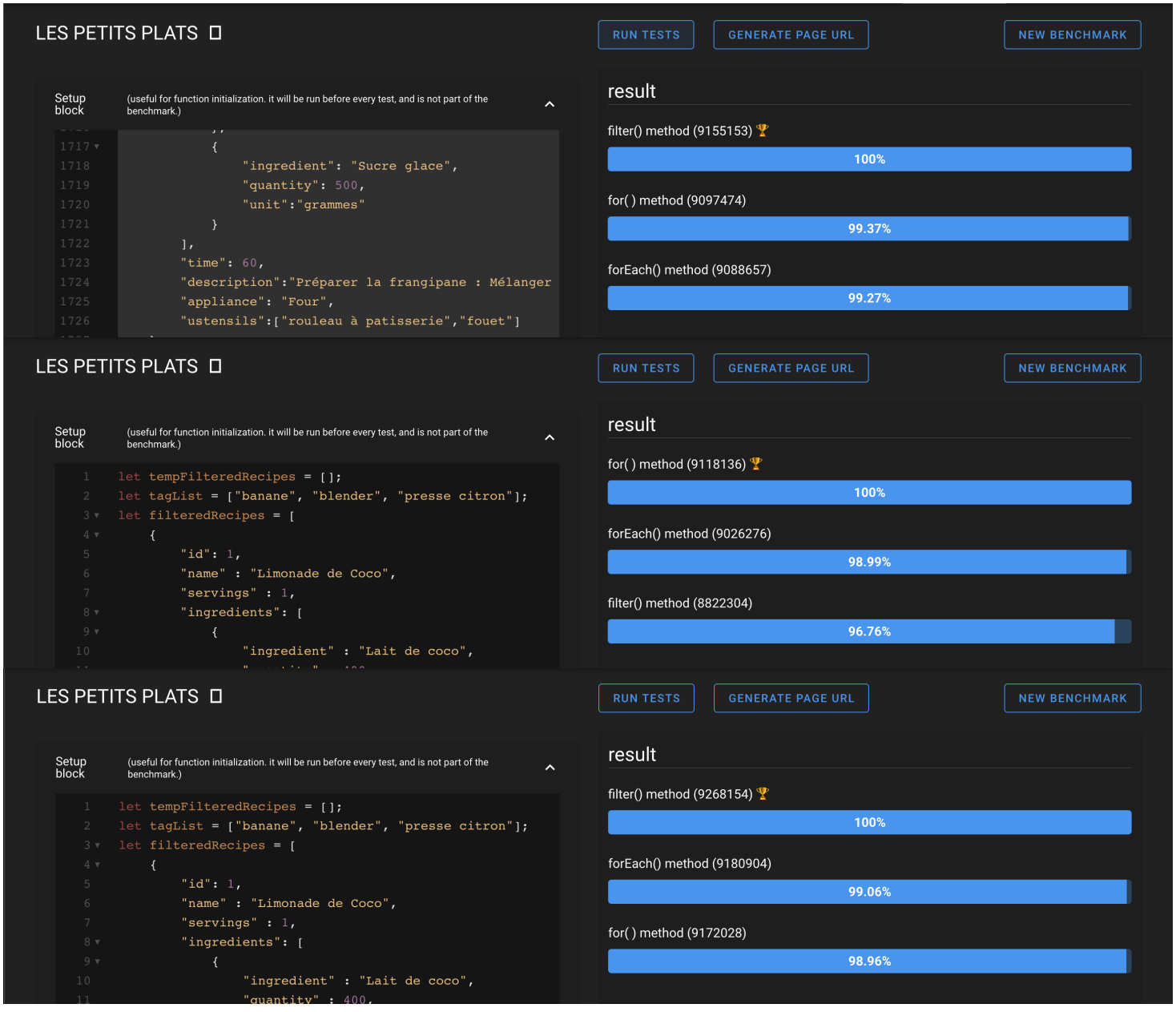
https://tinyurl.com/34jtyjcd



Le nombre d'opérations par seconde indique que la méthode forEach() serait la plus rapide.
Les résultats fluctuent beaucoup et une moyenne de plusieurs tests est à privilégier.

Resultat benchmarks

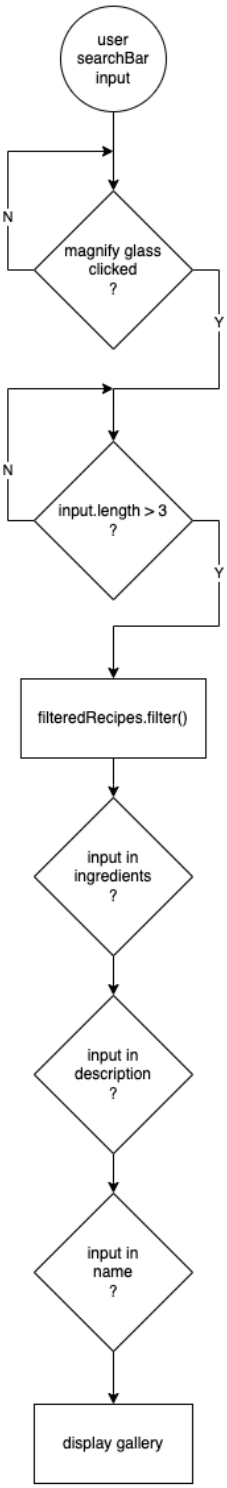
https://jsben.ch/uwVkmv



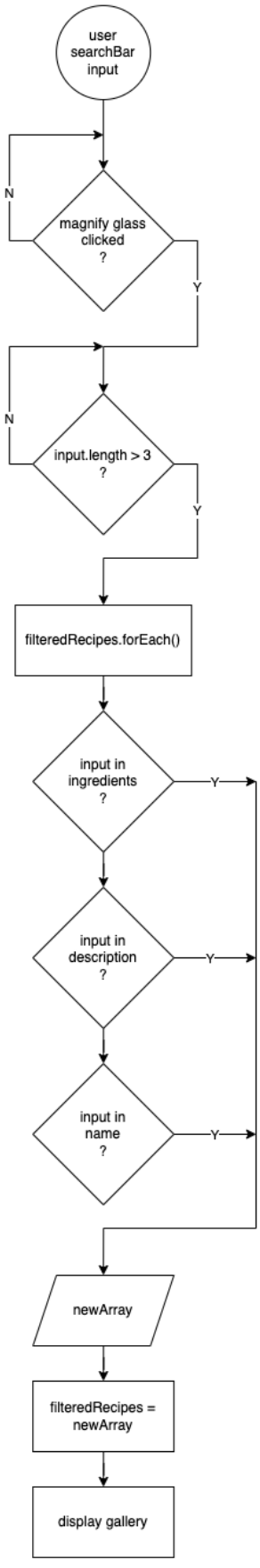
Les benchmarks peuvent être aléatoires c'est pourquoi il est conseillé de dégager une moyenne des résultats obtenus. Après plusieurs essais, la méthode filter() semble être la plus optimisée.

ALGORIGRAMME

filter() method



forEach() method



for() method