Fiche d’ investigation de fonctionnalités

Fonctionnalité: Moteur de recherche

Problématique: Offrir une expérience utilisateur fluide et efficace lors de la recherche de recettes.

Option 1: Implémentation avec méthodes de l'objet array (forEach, filter)

Cette implémentation utilise les méthodes de l'objet array pour parcourir et manipuler les données de manière fonctionnelle. La méthode forEach permet d'itérer sur chaque élément d'un tableau et d'effectuer une action spécifique, tandis que la méthode filter permet de créer un nouveau tableau contenant uniquement les éléments qui répondent à une condition donnée.

Avantages

Expressivité :
permettent d'écrire un code plus concis et lisible.

Manipulation des données :
offrent des fonctionnalités avancées pour la manipulation des données.

Composition :
peuvent être facilement combinées pour effectuer des opérations complexes sur les données.

Inconvénients

Performance :
peut entraîner une légère perte de performance par rapport aux boucles natives, car elles impliquent des appels de fonctions supplémentaires.

Compatibilité :
Certaines anciennes versions de navigateurs ou d'environnements JavaScript peuvent ne pas prendre en charge toutes les méthodes de boucles fonctionnelles.

Option 2: Implémentation avec des boucles natives (for)

Cette implémentation consiste à parcourir chaque recette dans notre base de données de manière itérative à l'aide de boucles natives for et à vérifier si la recherche correspond au nom, à la description ou aux ingrédients de la recette. Si une correspondance est trouvée, la recette est ajoutée à un tableau qui sera renvoyé en tant que résultats de la recherche.

Avantages

Performance :
peuvent être plus performantes car elles évitent les appels de fonctions supplémentaires.

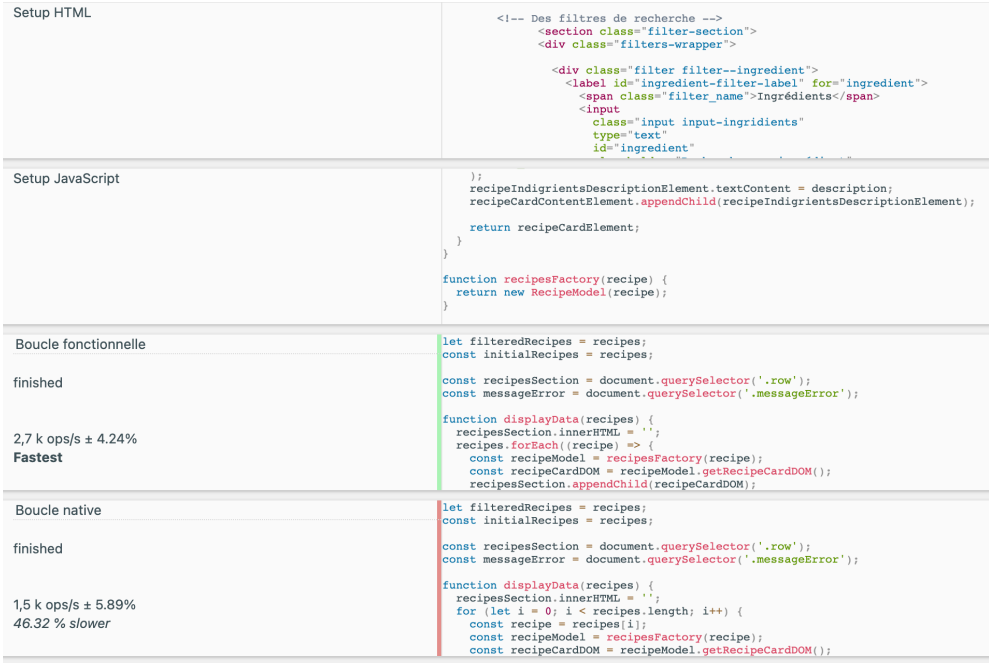
Compatibilité :
prises en charge par tous les navigateurs et environnements JavaScript.

Inconvénients

Syntaxe plus complexe :
nécessitent l'utilisation de variables d'index et peuvent être moins lisibles que les boucles fonctionnelles.

Manipulation des données :
offrent moins de fonctionnalités intégrées pour la manipulation des données, ce qui peut nécessiter plus de code pour réaliser des opérations complexes.

Comparaison de performance de la recherche principale (JSBench.me)



Pour 50 recettes:

	Opération par seconde	Avantage en %
Boucle fonctionnelle	2,7 k ops/s	2.24%
Boucle native	1,5 k ops/s	5.89%

La boucle fonctionnelle est plus performante que la boucle native dans ce cas spécifique de recherche principale sur 50 recettes. La boucle fonctionnelle est capable d'exécuter plus d'opérations par seconde, ce qui se traduit par une meilleure performance et une recherche plus rapide des recettes.

Pour 100 recettes:

	Opération par seconde	Avantage en %
Boucle fonctionnelle	1,4 k ops/s	3.97%
Boucle native	1,2 k ops/s	5.13%

La boucle fonctionnelle est plus performante que la boucle native pour la recherche de 100 recettes.

Pour 500 recettes:

	Opération par seconde	Avantage en %
Boucle fonctionnelle	396 ops/s	0.49%
Boucle native	392 ops/s	1.27%

Dans ce cas, les performances des deux boucles sont très proches, avec des différences minimes. Les deux boucles sont donc relativement similaires en termes de vitesse d'exécution pour la recherche de 500 recettes. Cependant, la boucle fonctionnelle conserve un léger avantage en termes de performance, bien que ce soit une différence très faible.

Pour 1000 recettes:

	Opération par seconde	Avantage en %
Boucle fonctionnelle	187 ops/s	3.74%
Boucle native	197 ops/s	2.78%

Dans ce cas, la boucle native affiche une légère amélioration de performance par rapport à la boucle fonctionnelle pour la recherche de 1000 recettes. La boucle native est plus rapide de 3,74% par rapport à la boucle fonctionnelle.

Dans l'ensemble, la boucle native présente une légère amélioration de performance par rapport à la boucle fonctionnelle pour la recherche de 1000 recettes. Cependant, il est important de prendre en considérations des facteurs telles que la lisibilité et la maintenabilité du code.

Solution retenue :

La solution retenue utilisant les boucles fonctionnelles a montré une performance supérieure lors des tests.

Pour le traitement des 50 recettes, la boucle fonctionnelle a traité les recettes à une vitesse de 2,7 k ops/s, soit 2,24% plus rapide que la boucle native qui a atteint 1,5 k ops/s, soit 46,32% plus lente.

La boucle fonctionnelle nous permet de rechercher les recettes plus rapidement, améliorant ainsi l'expérience utilisateur en réduisant les temps d'attente. De plus, son code est plus clair et plus concis, facilitant la lecture et la compréhension.

En tenant compte de ces avantages, nous avons choisi d'adopter la boucle fonctionnelle pour notre recherche de recettes. Cela nous permettra d'offrir une expérience utilisateur plus fluide et réactive, avec des temps de réponse rapides lors de la recherche des recettes.

Diagramme boucle fonctionnelle

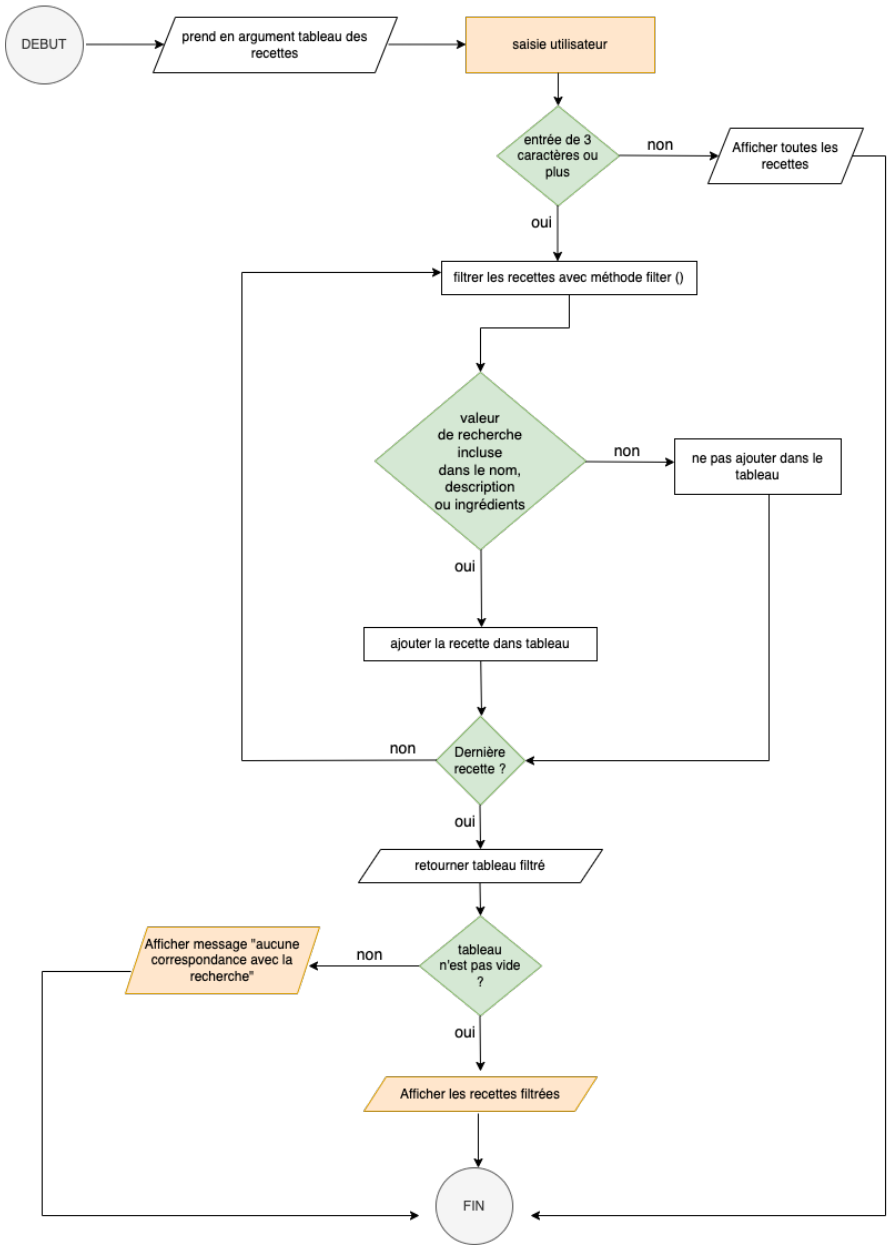


Diagramme boucle native