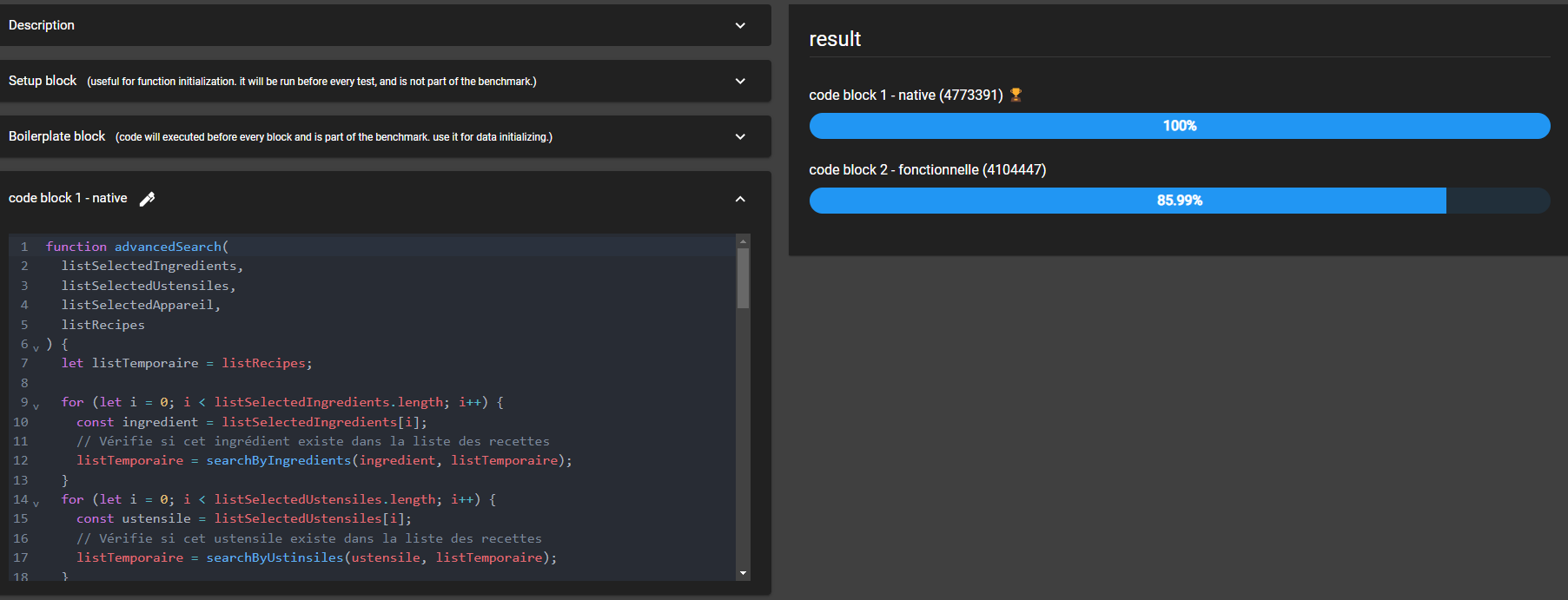
**Fiche d’investigation de fonctionnalité**

|  |  |
| --- | --- |
| **Fonctionnalité : Recherche Principale** | **Fonctionnalité #2** |
| **Problématique :**  Le défi principal est de développer un système de recherche quasi instantané, offrant une expérience utilisateur fluide. La recherche doit pouvoir filtrer les recettes en temps réel à partir du texte saisi par l'utilisateur, ainsi que par les tags sélectionnés (ingrédients, appareils, ustensiles). | |
| **Option 1 : Approche avec boucles natives**  Cette approche utilisera les boucles JavaScript natives (for, while) pour parcourir et filtrer les données des recettes.  **Branche sur le repository GitHub** : native | |
| **Avantage :**   * Contrôle précis sur l'itération des données * Gain de performance (+ rapide) | **Inconvénients :**   * Longueur du code * Maintenabilité du code difficile * Risque plus élevé d'erreurs de syntaxe (oubli d'incrément, condition de sortie mal définie) |

|  |  |
| --- | --- |
| **Option 2 : Approche avec programmation fonctionnelle**  Cette option utilise les méthodes forEach, filters de l'objet Array pour parcourir les données des recettes.  **Branche sur le repository GitHub** : main | |
| **Avantage :**   * Syntaxe plus concise et lisible * Moins de risques d'erreurs liées à la gestion de l'index | **Inconvénients :**  Parcourt toujours l'ensemble des données |

**Test de performance avec Jsbench**

* Code block 1 - native : 4773391 opérations par seconde (100% de performance relative)
* Code block 2 - fonctionnelle : 4104447 opérations par seconde (85.99% de performance relative)



**SOLUTION RETENUE :**Ces résultats montrent que l'implémentation native est légèrement plus performante que l'implémentation fonctionnelle. La différence de performance est d'environ 14.01%.

**Lien :**<https://jsben.ch/Zaf4d>