# Fiche d’investigation de fonctionnalité

| **Fonctionnalité :** Recherche principale | **Fonctionnalité #1** |
|---|---|
| **Problématique :** Déterminer l'algorithme le plus efficace pour vérifier la présence d'une sous-chaîne de charactères, parmis 3 chaines aléatoires de type et longueurs variables. | |

**Option 1 : Méthodes de l'Object Array (Built-in Library)**
La bibliothèque Js standard inclut de nombreuses méthodes différentes pour l'objet Array. Sur Node & Chrome, l'implémetation du moteur V8 permet de nombreuses optimisations bas niveaux améliorant les performances des constructions haut niveaux a travers différents composants du JIT (Just in Time) compiler.

| **Avantages** | **Inconvénients** |
|---|---|
| ⊕ Lisibilité humaine, maintenabilité ...<br>⊕ Rapidité | ⊖ Difficulté d'optimization en fonction du dataSet ?<br>⊖ Algorithme naïf de compléxité O(m*n) |

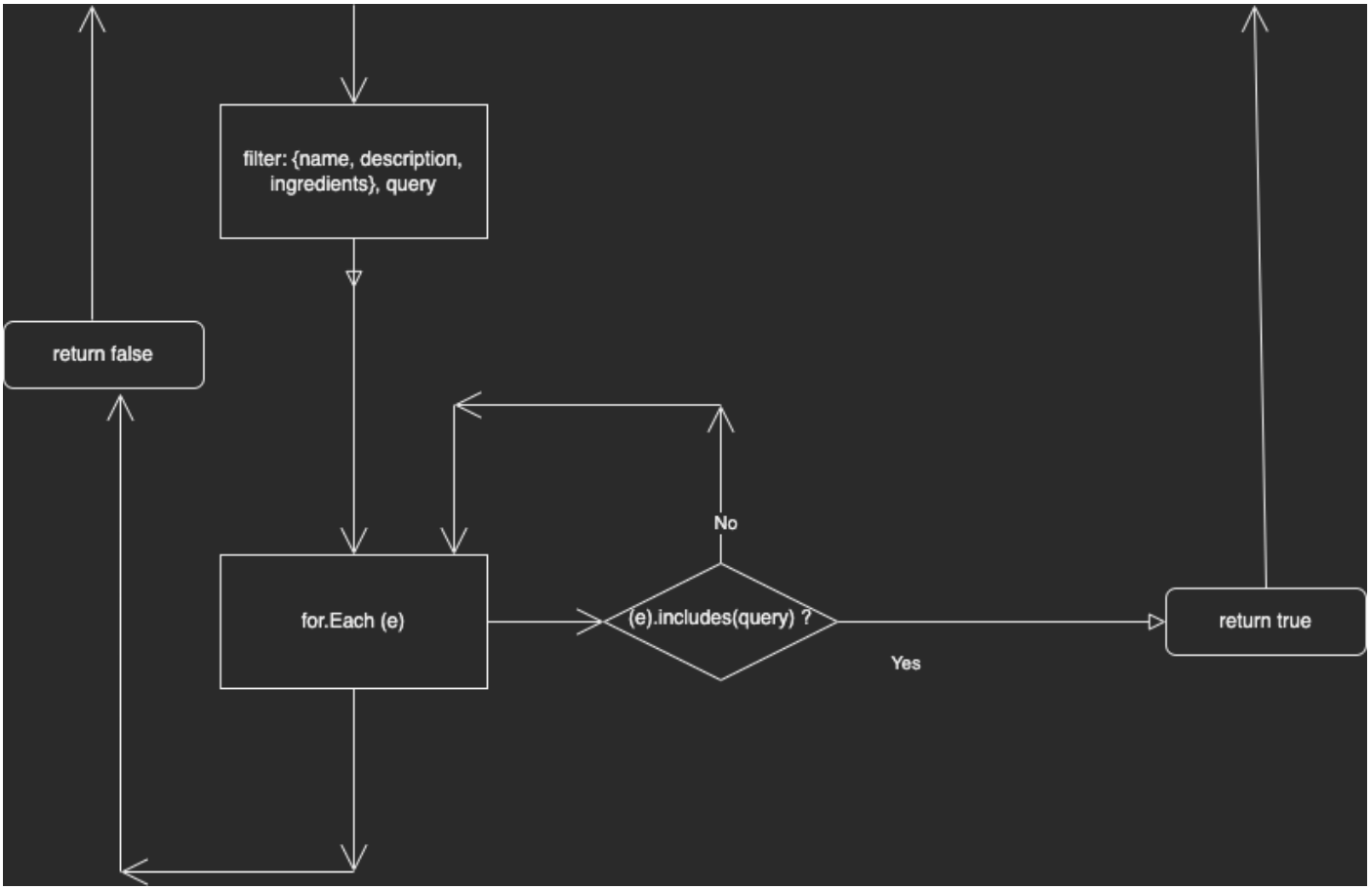
**Longueur minimal de P: 3**
Recherche exécuté sur 3 chaînes (T) de taille variable

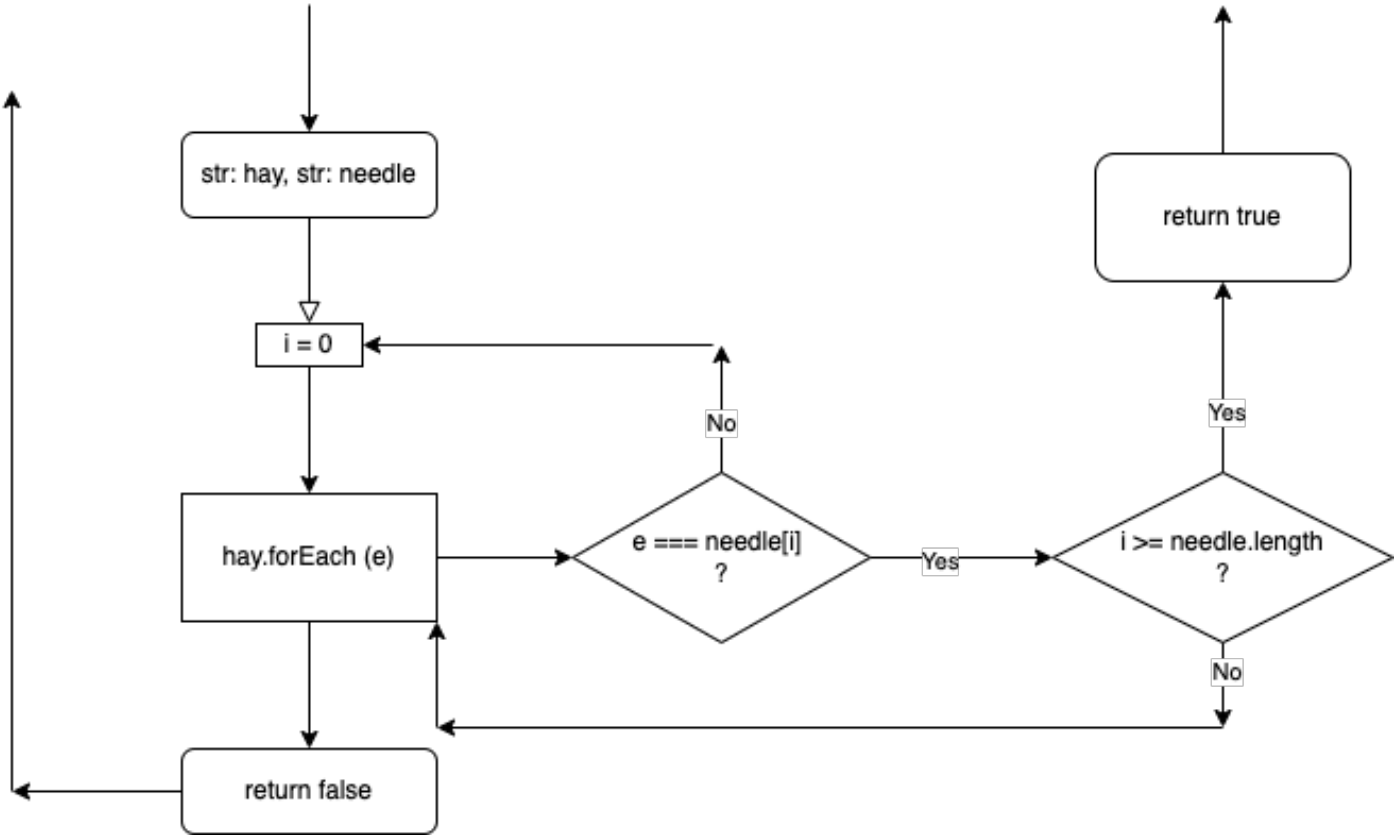
**Option 2 : Boucles natives**
Implémentation d'un algorithme sensiblement identique a la méthode includes mais non optimisé par V8
JS étant un langage de haut niveau, l'implémentation d'une construction bas niveau à son échelle du stack sera pour la majorité du temps, dépassé en performance par des methodes standards

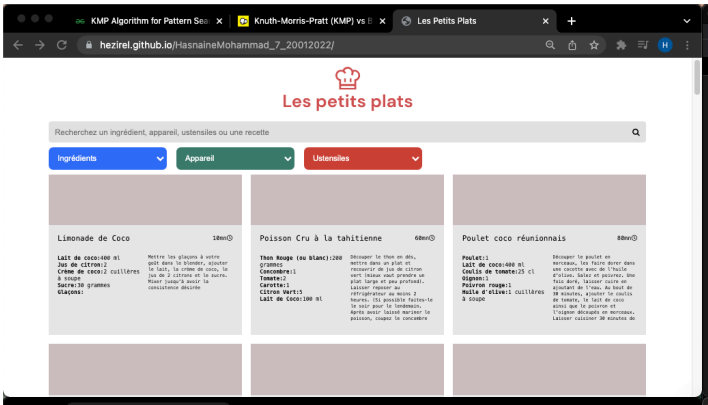
| **Avantages** | **Inconvénients** |
|---|---|
| ⊕ Personnalisation de l'algorithme de recherche strStr<br>⊕ Pré Optimisation par adaptation du dataSet ?<br>⊕<br><br>⊕ | ⊖Moins lisible pour un humain |

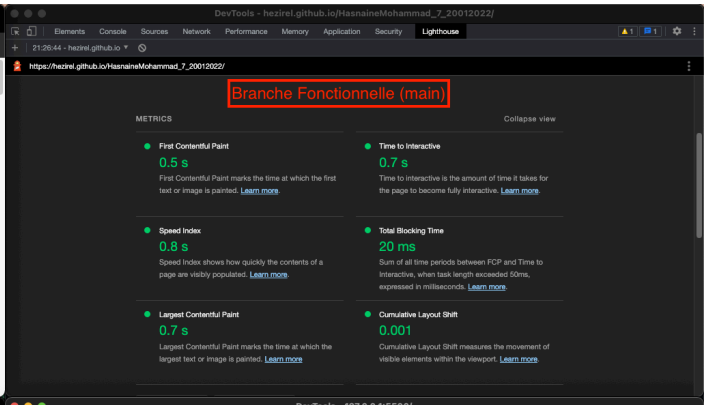
**Même variables d'expérimentation que pour l'option 1**

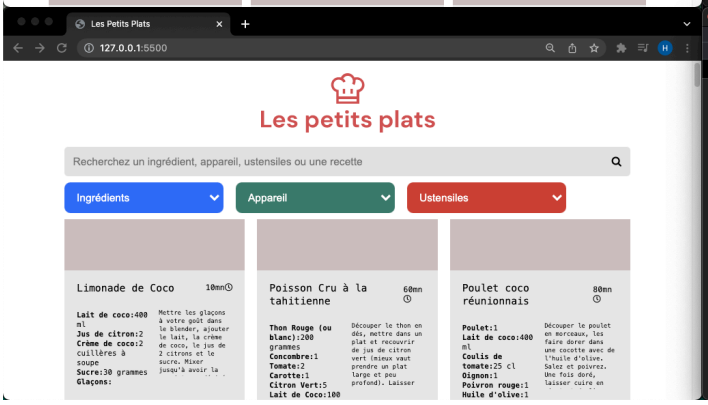
**Solution retenue :**
Le choix s'est porté sur l'option 1 après analyse des benchmarks Lighthouse sur un env Chrome.
L'implémentation de différents algorithmes plus bas niveaux comme KMP ou Boyers Moores peut se présenter comme voie d'amélioration possible.

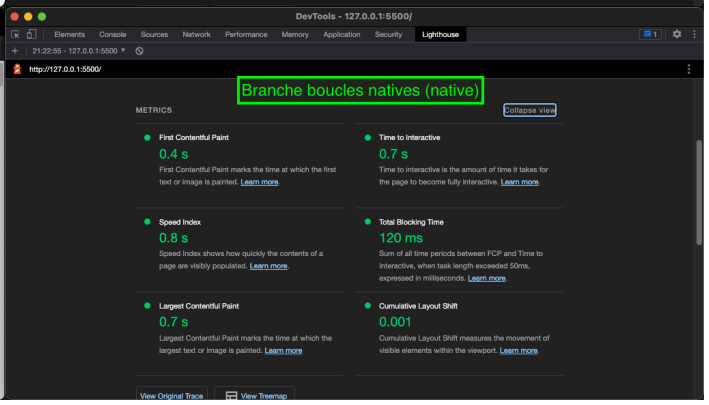

filter: {name, description, ingredients}, query
return false
No
for.Each (e)
(e).includes(query) ?
return true
Yes


str: hay, str: needle
i = 0
hay.forEach (e)
e === needle[i]
?
No
Yes
i >= needle.length
?
Yes
No
return true
return false


Les Petits Plats
hezirel.github.io/HasnaineMohammad_7_20012022/
Les petits plats
Recherchez un ingrédient, appareil, ustensiles ou une recette
Ingrédients
Appareil
Ustensiles
Limonade de Coco
10mn
Poisson Cru à la tahitienne
60mn
Poulet coco réunionnais
80mn


DevTools - hezirel.github.io/HasnaineMohammad_7_20012022/
Lighthouse
https://hezirel.github.io/HasnaineMohammad_7_20012022/
Branche Fonctionnelle (main)
METRICS
Collapse view
First Contentful Paint
0.5 s
Time to Interactive
0.7 s
Speed Index
0.8 s
Total Blocking Time
20 ms
Largest Contentful Paint
0.7 s
Cumulative Layout Shift
0.001


Les Petits Plats
127.0.0.1:5500
Les petits plats
Recherchez un ingrédient, appareil, ustensiles ou une recette
Ingrédients
Appareil
Ustensiles
Limonade de Coco
10mn
Poisson Cru à la tahitienne
60mn
Poulet coco réunionnais
80mn


DevTools - 127.0.0.1:5500/
Lighthouse
http://127.0.0.1:5500/
Branche boucles natives (native)
METRICS
Collapse view
First Contentful Paint
0.4 s
Time to Interactive
0.7 s
Speed Index
0.8 s
Total Blocking Time
120 ms
Largest Contentful Paint
0.7 s
Cumulative Layout Shift
0.001
View Original Trace
View Treemap

```
f2:Functionnal loop performance: 155.182ms
f3:kmp loop performance: 174.508ms
f4:bm loop performance: 159.554ms
[nodemon] clean exit - waiting for changes before restart
rs
[nodemon] starting `node main.js`
f0:Control loop performance: 476.623ms
f1:Native loop performance: 766.793ms
f2:Functionnal loop performance: 485.403ms
f3:kmp loop performance: 391.341ms
f4:bm loop performance: 273.289ms
[nodemon] clean exit - waiting for changes before restart
rs
[nodemon] starting `node main.js`
f0:Control loop performance: 603.511ms
f1:Native loop performance: 408.497ms
f2:Functionnal loop performance: 349.002ms
f3:kmp loop performance: 329.292ms
f4:bm loop performance: 248.453ms
[nodemon] clean exit - waiting for changes before restart
rs
[nodemon] starting `node main.js`
f0:Control loop performance: 278.105ms
f1:Native loop performance: 226.915ms
f2:Functionnal loop performance: 213.367ms
f3:kmp loop performance: 205.582ms
f4:bm loop performance: 249.771ms
[nodemon] clean exit - waiting for changes before restart
rs
[nodemon] starting `node main.js`
f0:Control loop performance: 206.73ms
f1:Native loop performance: 183.754ms
f2:Functionnal loop performance: 157.626ms
f3:kmp loop performance: 170.068ms
f4:bm loop performance: 164.184ms
```