

## Analyse des Performances JS

### 1. Benchmark JSBen.ch

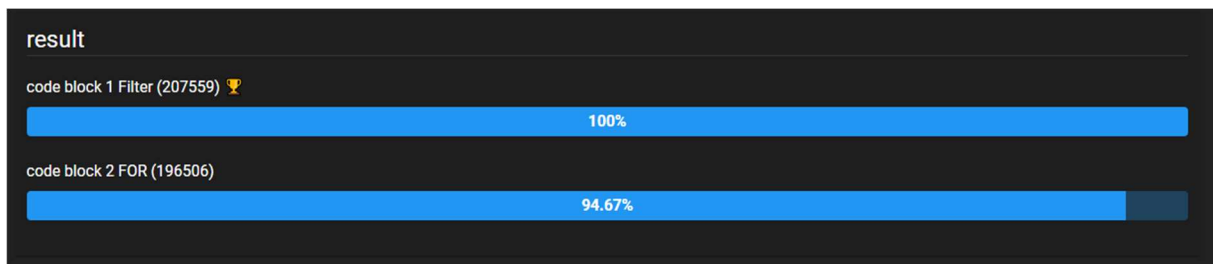
Lien : <https://jsben.ch/ZD9id>

On test les performances JS suivant plusieurs scénarios. Les nombres du tableau indiquent le nombre total d'opérations/seconde. Plus la valeur est grande plus l'algorithme est performant :

	Algo1 Filter	Algo2 For
Scenario1 (occurrence: ex: poulet) data: 50	191023	190174
Scenario2 (occurrence: ex: "") data: 50	1561076	332013
Scenario3 (occurrence: ex: poulet) data: 500	16007	16007
Scenario4 (occurrence: ex: poulet) data: 1000	9894	9670
Scenario5 (occurrence: ex: poulet) data: 5000	6821	6612

On remarque ici que l'algorithme avec Filter est plus performance que celui avec les boucles FOR.

Par exemple on obtient pour le scénatio1 avec 50 données :



## Outils tiers :

<https://jsben.ch/>

The screenshot shows the JS Benchmarks website interface. At the top, it says "filter vs for" and "v1". Below this, there's a section for "Setup HTML" and "Setup JavaScript". The "Setup JavaScript" section contains a code snippet for a benchmark. The benchmark compares two methods: "FILTER" and "FOR". The "FILTER" method is marked as "Fastest" with a performance of 102599.83 ops/s ± 0.87%. The "FOR" method has a performance of 101946.8 ops/s ± 0.51%, which is 0.64% slower. The code snippet for the benchmark is as follows:

```
const recipes = [
  {
    "id": 1,
    "name": "Limonade de Coco",
    "servings": 1,
    "ingredients": [
      {
        "ingredient": "Lait de coco",
        "quantity": 400,
        "unit": "ml"
      }
    ]
  }
];

let value = "Poulet";
let recipesMatched = [];
recipes.filter((recipe) => {
  if (
    recipe.name.includes(value) ||
    recipe.description.includes(value) ||
    recipe.ingredients.some((elt) => elt.ingredient.includes(value))
  ) {
    recipesMatched.push(recipe);
  }
});
return /
```

The "FOR" method code is similar but uses a for loop to iterate through the recipes array and check for matches.

## Outils internes (code JS) :

console.time(), console.timeEnd()

## Liens Comparaison boucles JS (ressources outils tiers/internes) :

[https://leanylabs.com/blog/js-forEach-map-reduce-vs-for-for\\_of/](https://leanylabs.com/blog/js-forEach-map-reduce-vs-for-for_of/)

<https://stackoverflow.com/questions/1003855/how-can-i-benchmark-javascript-code>

<https://stackoverflow.com/questions/44612083/which-js-benchmark-site-is-correct>

<https://stackoverflow.com/questions/111368/how-do-you-performance-test-javascript-code>

Développement à en cours/terminer :

- Multi tags pour une même catégorie ?
- Multi tags par chaque catégorie ?
- Reset tags ?
- Ustensiles, appareils