

# *P7 - Développez un algorithme de recherche en Javascript*



*par Thomas RANQUE*

# Le projet


Réalisation d'un moteur de recherche performant pour le site "les petits plats".

Bootstrap pour l'UI.


2 algorithmes de recherche:

- Boucles natives
- Méthodes d'array

Fiche d'investigation de fonctionnalité + algorithme.




## Les petits plats

Rechercher une recette 

Ingredients ▾

Appareils ▾

Ustensiles ▾




### Pates Carbonara

🕒 30 min

**Tagliatelles:** 500 grammes  
**Lardons:** 150 grammes  
**Crème fraîche:** 200 grammes  
**Parmesan:** 100 grammes  
**huile d'olive:** 1 cuillère à soupe

Faire cuire les pates comme indiqué sur le paquet. Dorer les lardons dans une sauteuse avec l'huile d'olive. Ajouter la crème fraîche et baisser le feu au minimum. Quand les Tagliatelles sont prêtes les mettre dans la sauteuse et bien mélanger le tout en ajoutant le jaun...

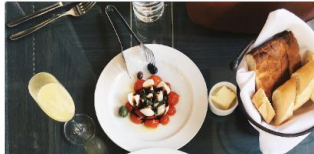


### Spaghettis à la bolognaise

🕒 30 min

**Spaghettis:** 400 grammes  
**Oignon:** 2  
**Coulis de tomate:** 300 grammes  
**V viande hachée 1% de matière grasse:** 400 grammes  
**Vin rouge:** 20 cl  
**Crème Fraîche:** 1 cuillère à soupe

Cuisiner la viande hachée dans une poêle à frire. Dans une autre faire cuire les oignons découpés en fins dés avec un peu de beurre. Ajouter du vin rouge. Mélanger les oignons avec la viande hachée. Faire cuire les pates le temps indiqué sur le paquet. Ajouter le...






### Fondant au chocolat

🕒 30 min

**Beurre:** 160 grammes  
**Chocolat noir:** 200 grammes  
**Farine:** 50 grammes  
**Oeuf:** 4  
**Sucre:** 150 grammes

Faire fondre le chocolat et le beurre au bain marie. Dans un saladier battre les oeufs avec le sucre jusqu'à obtenir une texture de type mousse. Ajouter la farine ainsi que le mélange de beurre et chocolat fondu. Beurrez le moule à gateaux. Mettre au four préchauffé à 200° pul...



# Optimisations

Structure du site & responsive

## Séparation HTML/CSS/JS :

Index.html

css

scripts

index.js

factory.js

search.js

assets

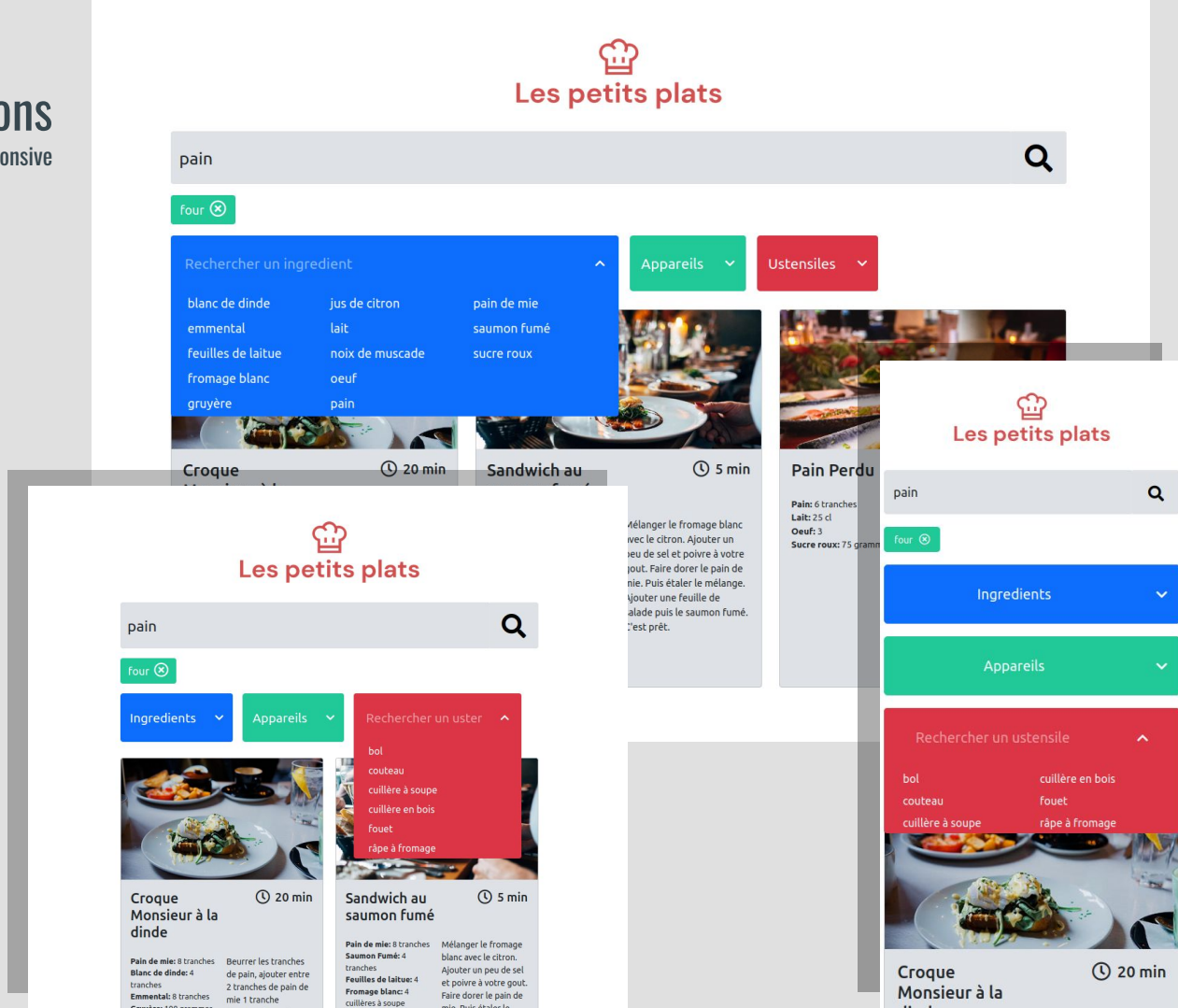
photos

logo.png

recipes.json

Breakpoints 992px / 768px

Mobile first



# Factory pattern design

Basé sur 3 classes:

- Tag
- BtnSubSearch
- Card

Méthodes de création:

- make()
- makeActive() / makeInactive()

Méthodes de traitement des données JSON:

- setPhoto()
- getIngredients()

Méthode statique:

- static #var

```
1 //////////////////////////////////////////////////
2 // TAGS Factory //
3 class Tag {
4   constructor(type, txt) {
5     this.type = type;
6     this.txt = txt;
7     this.color = this.defColor();
8   }
9   defColor() {
10    switch (this.type) {
11      case 'ingredients':
12        return 'primary';
13      case 'appliance':
14        return 'success';
15      case 'ustensils':
16        return 'danger';
17      default:
18        console.log('ERROR with TAG type constructor! Define new search type or check type...');
19    }
20  }
21  make() {
22    const tag = document.createElement('button');
23    tag.classList = `btn w-auto rounded d-flex align-items-center text-white bg-${this.color}`;
24    tag.id = this.txt;
25    tag.title = this.type;
26    tag.innerHTML = `${this.txt} <span><i class="far fa-times-circle ms-2"></i></span>`;
27    tag.addEventListener('click', closeTag);
28    return tag;
29  }
30 }
31
32
33 //////////////////////////////////////////////////
34 //BTN SUB SEARCH Factory //
35 const typeValues = {ingredients: 'ingredients', appliance: 'appareils', ustensils: 'ustensiles'};
36
37 > class BtnSubSearch {
38   //
39   //
40   //
41   //
42   //
43   //
44   //
45   //
46   //
47   //
48   //
49   //
50   //
51   //
52   //
53   //
54   //
55   //
56   //
57   //
58   //
59   //
60   //
61   //
62   //
63   //
64   //
65   //
66   //
67   //
68   //
69   //
70   //
71   //
72   //
73   //
74   //
75   //
76   //
77   //
78   //
79   //
80   //
81   //
82   //
83   //
84   //
85   //
86   //
87   //
88   //
89   //
90   //
91   //
92   //
93   //
94   //
95   //
96   //
97   //
98   //
99   //
100  //
101  //
102  // CARD Factory //
103  const fakePhotos = [
104    './assets/photos/jay-wennington-N_Y88TWmGwA-unsplash.jpg',
105    './assets/photos/louis-hansel-shotsoflouis-qNBGVy0CY8Q-unsplash.jpg',
106    './assets/photos/stil-u2Lp8tXicjw-unsplash.jpg',
107    './assets/photos/toa-heftiba-DQKerTsQwi0-unsplash.jpg'
108  ];
109
110 > class Card {
111   //
112   //
113   //
114   //
115   //
116   //
117   //
118   //
119   //
120   //
121   //
122   //
123   //
124   //
125   //
126   //
127   //
128   //
129   //
130   //
131   //
132   //
133   //
134   //
135   //
136   //
137   //
138   //
139   //
140   //
141   //
142   //
143   //
144   //
145   //
146   //
147   //
148   //
149   //
150   //
151   //
152   //
153   //
154   //
155   //
156   //
157   //
158   //
159   //
160   //
161   //
162   //
163   //
164   //
165   //
166   //
167   //
168   //
169   //
170   //
171   //
172   //
173   //
174   //
175   //
176   //
177   //
178   //
179   //
180   //
181   //
182   //
183   //
184   //
185   //
186   //
187   //
188   //
189   //
190   //
191   //
192   //
193   //
194   //
195   //
196   //
197   //
198   //
199   //
200   //
201   //
202   //
203   //
204   //
205   //
206   //
207   //
208   //
209   //
210   //
211   //
212   //
213   //
214   //
215   //
216   //
217   //
218   //
219   //
220   //
221   //
222   //
223   //
224   //
225   //
226   //
227   //
228   //
229   //
230   //
231   //
232   //
233   //
234   //
235   //
236   //
237   //
238   //
239   //
240   //
241   //
242   //
243   //
244   //
245   //
246   //
247   //
248   //
249   //
250   //
251   //
252   //
253   //
254   //
255   //
256   //
257   //
258   //
259   //
260   //
261   //
262   //
263   //
264   //
265   //
266   //
267   //
268   //
269   //
270   //
271   //
272   //
273   //
274   //
275   //
276   //
277   //
278   //
279   //
280   //
281   //
282   //
283   //
284   //
285   //
286   //
287   //
288   //
289   //
290   //
291   //
292   //
293   //
294   //
295   //
296   //
297   //
298   //
299   //
300   //
301   //
302   //
303   //
304   //
305   //
306   //
307   //
308   //
309   //
310   //
311   //
312   //
313   //
314   //
315   //
316   //
317   //
318   //
319   //
320   //
321   //
322   //
323   //
324   //
325   //
326   //
327   //
328   //
329   //
330   //
331   //
332   //
333   //
334   //
335   //
336   //
337   //
338   //
339   //
340   //
341   //
342   //
343   //
344   //
345   //
346   //
347   //
348   //
349   //
350   //
351   //
352   //
353   //
354   //
355   //
356   //
357   //
358   //
359   //
360   //
361   //
362   //
363   //
364   //
365   //
366   //
367   //
368   //
369   //
370   //
371   //
372   //
373   //
374   //
375   //
376   //
377   //
378   //
379   //
380   //
381   //
382   //
383   //
384   //
385   //
386   //
387   //
388   //
389   //
390   //
391   //
392   //
393   //
394   //
395   //
396   //
397   //
398   //
399   //
400   //
401   //
402   //
403   //
404   //
405   //
406   //
407   //
408   //
409   //
410   //
411   //
412   //
413   //
414   //
415   //
416   //
417   //
418   //
419   //
420   //
421   //
422   //
423   //
424   //
425   //
426   //
427   //
428   //
429   //
430   //
431   //
432   //
433   //
434   //
435   //
436   //
437   //
438   //
439   //
440   //
441   //
442   //
443   //
444   //
445   //
446   //
447   //
448   //
449   //
450   //
451   //
452   //
453   //
454   //
455   //
456   //
457   //
458   //
459   //
460   //
461   //
462   //
463   //
464   //
465   //
466   //
467   //
468   //
469   //
470   //
471   //
472   //
473   //
474   //
475   //
476   //
477   //
478   //
479   //
480   //
481   //
482   //
483   //
484   //
485   //
486   //
487   //
488   //
489   //
490   //
491   //
492   //
493   //
494   //
495   //
496   //
497   //
498   //
499   //
500   //
501   //
502   //
503   //
504   //
505   //
506   //
507   //
508   //
509   //
510   //
511   //
512   //
513   //
514   //
515   //
516   //
517   //
518   //
519   //
520   //
521   //
522   //
523   //
524   //
525   //
526   //
527   //
528   //
529   //
530   //
531   //
532   //
533   //
534   //
535   //
536   //
537   //
538   //
539   //
540   //
541   //
542   //
543   //
544   //
545   //
546   //
547   //
548   //
549   //
550   //
551   //
552   //
553   //
554   //
555   //
556   //
557   //
558   //
559   //
560   //
561   //
562   //
563   //
564   //
565   //
566   //
567   //
568   //
569   //
570   //
571   //
572   //
573   //
574   //
575   //
576   //
577   //
578   //
579   //
580   //
581   //
582   //
583   //
584   //
585   //
586   //
587   //
588   //
589   //
590   //
591   //
592   //
593   //
594   //
595   //
596   //
597   //
598   //
599   //
600   //
601   //
602   //
603   //
604   //
605   //
606   //
607   //
608   //
609   //
610   //
611   //
612   //
613   //
614   //
615   //
616   //
617   //
618   //
619   //
620   //
621   //
622   //
623   //
624   //
625   //
626   //
627   //
628   //
629   //
630   //
631   //
632   //
633   //
634   //
635   //
636   //
637   //
638   //
639   //
640   //
641   //
642   //
643   //
644   //
645   //
646   //
647   //
648   //
649   //
650   //
651   //
652   //
653   //
654   //
655   //
656   //
657   //
658   //
659   //
660   //
661   //
662   //
663   //
664   //
665   //
666   //
667   //
668   //
669   //
670   //
671   //
672   //
673   //
674   //
675   //
676   //
677   //
678   //
679   //
680   //
681   //
682   //
683   //
684   //
685   //
686   //
687   //
688   //
689   //
690   //
691   //
692   //
693   //
694   //
695   //
696   //
697   //
698   //
699   //
700   //
701   //
702   //
703   //
704   //
705   //
706   //
707   //
708   //
709   //
710   //
711   //
712   //
713   //
714   //
715   //
716   //
717   //
718   //
719   //
720   //
721   //
722   //
723   //
724   //
725   //
726   //
727   //
728   //
729   //
730   //
731   //
732   //
733   //
734   //
735   //
736   //
737   //
738   //
739   //
740   //
741   //
742   //
743   //
744   //
745   //
746   //
747   //
748   //
749   //
750   //
751   //
752   //
753   //
754   //
755   //
756   //
757   //
758   //
759   //
760   //
761   //
762   //
763   //
764   //
765   //
766   //
767   //
768   //
769   //
770   //
771   //
772   //
773   //
774   //
775   //
776   //
777   //
778   //
779   //
780   //
781   //
782   //
783   //
784   //
785   //
786   //
787   //
788   //
789   //
790   //
791   //
792   //
793   //
794   //
795   //
796   //
797   //
798   //
799   //
800   //
801   //
802   //
803   //
804   //
805   //
806   //
807   //
808   //
809   //
810   //
811   //
812   //
813   //
814   //
815   //
816   //
817   //
818   //
819   //
820   //
821   //
822   //
823   //
824   //
825   //
826   //
827   //
828   //
829   //
830   //
831   //
832   //
833   //
834   //
835   //
836   //
837   //
838   //
839   //
840   //
841   //
842   //
843   //
844   //
845   //
846   //
847   //
848   //
849   //
850   //
851   //
852   //
853   //
854   //
855   //
856   //
857   //
858   //
859   //
860   //
861   //
862   //
863   //
864   //
865   //
866   //
867   //
868   //
869   //
870   //
871   //
872   //
873   //
874   //
875   //
876   //
877   //
878   //
879   //
880   //
881   //
882   //
883   //
884   //
885   //
886   //
887   //
888   //
889   //
890   //
891   //
892   //
893   //
894   //
895   //
896   //
897   //
898   //
899   //
900   //
901   //
902   //
903   //
904   //
905   //
906   //
907   //
908   //
909   //
910   //
911   //
912   //
913   //
914   //
915   //
916   //
917   //
918   //
919   //
920   //
921   //
922   //
923   //
924   //
925   //
926   //
927   //
928   //
929   //
930   //
931   //
932   //
933   //
934   //
935   //
936   //
937   //
938   //
939   //
940   //
941   //
942   //
943   //
944   //
945   //
946   //
947   //
948   //
949   //
950   //
951   //
952   //
953   //
954   //
955   //
956   //
957   //
958   //
959   //
960   //
961   //
962   //
963   //
964   //
965   //
966   //
967   //
968   //
969   //
970   //
971   //
972   //
973   //
974   //
975   //
976   //
977   //
978   //
979   //
980   //
981   //
982   //
983   //
984   //
985   //
986   //
987   //
988   //
989   //
990   //
991   //
992   //
993   //
994   //
995   //
996   //
997   //
998   //
999   //
1000  }
```

# Recherche d'algorithmes

Vue d'ensemble



## Les petits plats

### Fiche d'investigation de fonctionnalité

| Fonctionnalité: Recherche de recettes  | Fonctionnalité #2 |
|--|-------------------|
| <b>Problématique:</b> La recherche par mot-clé doit être la plus rapide possible. Elle doit se déclencher à partir de 3 lettres entrées dans le champ de recherche, puis doit s'actualiser à chaque nouvelle lettre. |                   |

#### Option 1: Les boucles natives

L'objet "recettes" du document JSON est itéré par une boucle "for" qui explore successivement les composants "name", "description" et "ingrédients" (itéré également par une boucle "for") de chaque recette. Les recettes sélectionnées sont ajoutées à un array qui sert ensuite à l'affichage du résultat de la recherche.

En cas de résultat positif sur un composant, un court-circuit empêche l'itération sur les composants suivants. (Optimisation du temps de traitement de l'algorithme)

#### Avantages:

Lisibilité du code.  
Construction du résultat par étape.  
Optimisations possibles.

#### Inconvénients:

Code répétitif.  
Code volumineux.

#### Test d'efficacité:

1621 opérations/sec (81.33%) - **Test #1** sans les court-circuits - *JSBench.ch*  
1998 opérations/sec (100%) - **Test #2** avec les court-circuits - *JSBench.ch*  
3160 ms - **Test #3** sans les court-circuits - *console.time*  
3145 ms - **Test #4** avec les court-circuits - *console.time*

#### Option 2: Les méthodes de l'objet "array"

La méthode "filter" est employée directement sur l'objet "recettes" pour supprimer les recettes qui ne correspondent pas à la recherche. L'array obtenu est alors itéré pour effectuer l'affichage des résultats.

#### Avantages:

Code compact.  
Lisibilité.

#### Inconvénients:

Optimisations limitées.  
Appel à une callback.

#### Test d'efficacité: JSbench.js

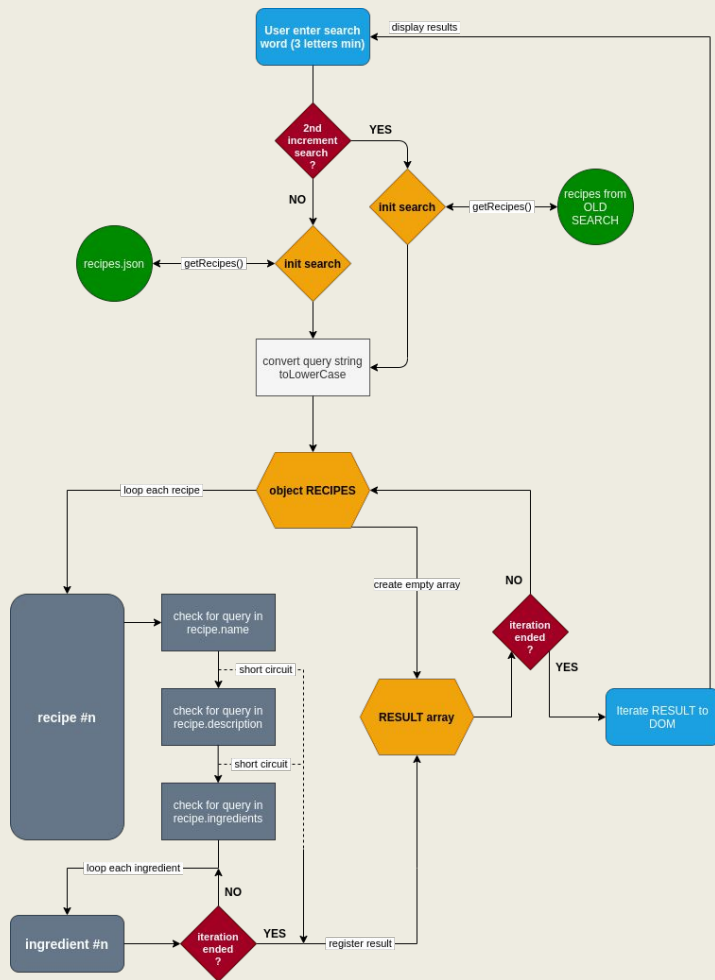
1993 opérations/sec (100%) - **Test #1** - *JSBench.ch*  
1415 opérations/sec (70.82%) - **Test #2** - *JSBench.ch*  
3193 ms - **Test #3** - *console.time*  
3210 ms - **Test #4** - *console.time*

#### Solution retenue:

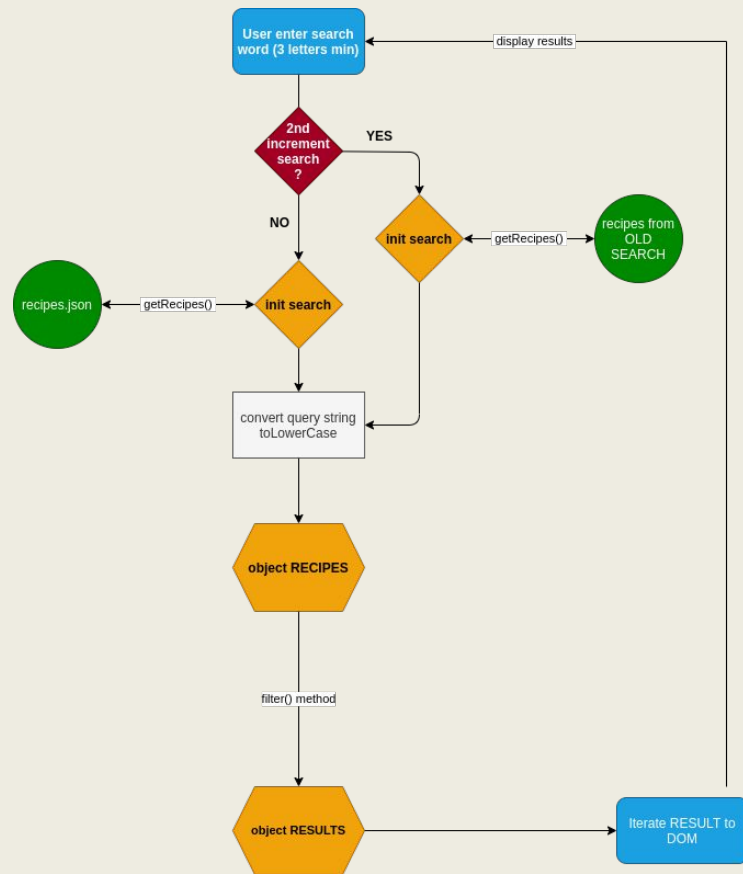
L'option 1 est la plus efficace. Il est nécessaire d'employer les court-circuits afin de ne pas avoir plusieurs fois la même recette dans les résultats. Les recettes des résultats précédents sont conservées comme base de recherche si l'utilisateur n'efface pas de lettre, afin d'améliorer davantage les performances de l'algorithme choisi.

# Algorigrammes

## Option 1 - Boucles natives



## Option 2 - Méthodes d'array



# Le versioning GitHub

Fork du repo fourni.

Respect du format de  
nommage du repo.

Commits réguliers et  
commentés.

Hébergement sur Pages  
(Github).

The screenshot displays the GitHub interface for the repository **Peanuts-83 / ThomasRanque\_7\_11022022**. The repository is public and has 4 branches and 0 tags. The file explorer shows the following structure:

- algo\_tests**: Ready for searchAlgo (3 days ago)
- assets**: main static results
- scripts**: Ready for searchAlgo
- scss**: UI responsive mobil tablet desk
- styles**: UI responsive mobil tablet desk
- .gitignore**: P7 NM
- favicon.ico**: header UI ok
- index.html**: Ready for searchAlgo
- package-lock.json**: project start
- package.json**: project start

A blue banner at the bottom of the file explorer encourages adding a README file to help people understand the project.

The commit history shows a series of commits on February 23, 2022, and February 22, 2022. The commits are as follows:

- Feb 23, 2022**
  - escape accents from tag search**: Peanuts-83 committed 27 minutes ago ✓
  - final loop version**: Peanuts-83 committed 1 hour ago ✓
  - cleaning**: Peanuts-83 committed 1 hour ago ✓
  - search & sub-search ok**: Peanuts-83 committed 2 hours ago
- Feb 22, 2022**
  - close subsearch & close tags & show/hide tag**: Peanuts-83 committed 18 hours ago
  - clicked tag filter ok**: Peanuts-83 committed 20 hours ago
  - responsive ok & tags display ok**: Peanuts-83 committed 22 hours ago
- Feb 21, 2022**
  - ingredients responsive ok**: Peanuts-83 committed 2 days ago



# Validations

Rigueur dans la structure des répertoire et la séparation des fichiers HTML/CSS/JS.

Commentaires dans les fichiers HTML/CSS/JS.

Valideurs W3C.

## Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

### Showing results for uploaded file index.html

Checker Input

Show ☐ source ☐ outline ☐ image report [Options...](#)

Check by [file upload](#) [Choisir un fichier](#) Aucun fichier choisi

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

[Check](#)

Document checking completed. 1

Used the HTML parser.

Total execution time 64 milliseconds.

[About this checker](#) • [Report an issue](#)



## Service de validation CSS du W3C

Résultats de la validation W3C CSS de section.css (CSS niveau 3 + SVG)

Aller à: [Avertissements \(5\)](#) [CSS valide](#)

### Résultats de la validation W3C CSS de section.css (CSS niveau 3 + SVG)

#### Félicitations ! Aucune erreur trouvée.

Ce document est valide conformément à la recommandation [CSS niveau 3 + SVG](#) !

Pour montrer à vos lecteurs votre souci d'interopérabilité lors de la création de cette page Web, vous pouvez afficher cette icône sur toutes les pages valides. Voici le fragment de XHTML que vous pouvez utiliser pour ajouter cette icône à votre page Web:



```
<a href="http://jigsaw.w3.org/css-validator/check/referer">
  
</a>
```



```
<a href="http://jigsaw.w3.org/css-validator/check/referer">
  
</a>
```

(fermez le tag img avec > au lieu de /> si vous utilisez HTML <= 4.01)



Interested in "developing" your developer skills? In W3C's hands-on Professional Certificate Program, learn how to code the right way by creating Web sites and apps that use the latest Web standards. [Find out more!](#)

[Donate](#) and help us build better tools for a better web.

Si vous le désirez, vous pouvez télécharger une copie de cette image dans votre répertoire Web local, et changer le fragment d'HTML décrit ci-dessus pour référencer le nouvel emplacement de celle-ci.



# Tests des algorithmes

JSBench.ch & Test perso

JSBENCH

BENCHMARKBROWSEDONATE

no title (put title and/or keywords here, which describes your test)

RUN TESTSGENERATE PAGE URLNEW BENCHMARK

Setup block (useful for function initialization, it will be run before every test, and is not part of the benchmark.)

bodyplate block (code will executed before every block and is part of the benchmark, use it for data initializing.)

```
1 // GLOBALS
2 let recipes = [];
3
4 // TODO: supprimer les accents et caracteres ,!%&-etc... avant la recherche!
5
6 // GET_JSON_DATA
7+ function getRecipes() {
8+   const recipes = [
9+     {
10+       "id": 1,
11+       "name": "Limonade de Coco",
12+       "servings": 1,
13+     },
14+   ];
15+ }
```

code block 1 (11311)

code block 2 (10562)

result

code block 1 (11311) 100%

code block 2 (10562) 93.38%

All ads have been removed. Please consider a donation. Thank you for your help!

Ethereum (ETH)

Chia (XCH)

Cardano (ADA)

Ravencoin (RVN)

Bitcoin (BTC)

Ripple (XRP)

Litecoin (LTC)

Monero (XMR)

Dogecoin (DOGE)

SOLANA (SOL)

code block 1

```
31 }
32 }
33+ for (let recipe of results) {
34+   showResults.innerText += recipe.name + '\n';
35+ }
36
37 const end = performance.now();
38 loopRes.innerText += `${end - start}ms - ${query}\n`;
39 loopTimes.push(end - start);
40 console.log('nbre de réponses:', results.length);
41 console.timeEnd('OPTION 1 - LOOP');
```

code block 2

```
13 return recipe.name.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "");
14 recipe.description.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "");
15 recipe.ingredients.filter(ingr => ingr.ingredient.toLowerCase().normalize("NFD").
```

```
43 // FOR LOOP SOLUTION
44 function loopAlgo(query) {
45   if (query == 'end') {
46     let res = loopTimes.reduce((sum, ms) => sum + ms, 0);
47     loopRes.innerText += `TOTAL: ${res}ms.\n\n`;
48     return;
49   }
50   console.time('OPTION 1 - LOOP');
51   const start = performance.now();
52
53   const showResults = document.querySelector('.for_loop .results');
54   showResults.innerText = '';
55   let results = [];
56   for (let recipe of recipes) {
57     let _added = false;
58     if (recipe.name.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "").includes(
59       query)) {
60       results.push(recipe);
61       // console.log('NAME', recipe.name);
62       _added = true;
63     } else if (recipe.description.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g,
64       "").includes(query) && !_added) {
65       results.push(recipe);
66       // console.log('DESCR', recipe.description);
67       _added = true;
68     } else if (!_added) {
69       for (let ingr of recipe.ingredients) {
70         if (ingr.ingredient.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "").
71           includes(query) && !_added) {
72           results.push(recipe);
73           _added = true;
74         }
75       }
76     }
77   }
78   showResults.innerText += results.map(recipe => recipe.name).join('\n');
79
80   const end = performance.now();
81   loopRes.innerText += `${end - start}ms - ${query}\n`;
82   loopTimes.push(end - start);
83   console.log('nbre de réponses:', results.length);
84   console.timeEnd('OPTION 2 - ARRAY');
85 }
86
87 // ARRAY SOLUTION
88 function arrayAlgo(query) {
89   if (query == 'end') {
90     let res = loopTimes.reduce((sum, ms) => sum + ms, 0);
91     arrayRes.innerText += `TOTAL: ${res}ms.\n\n`;
92     return;
93   }
94   console.time('OPTION 2 - ARRAY');
```

FOR LOOP SOLUTION

20ms - tomate  
14ms - citron  
6ms - salade  
7ms - oignon  
2ms - boeuf  
1ms - poulet  
43ms - des  
17ms - les  
7ms - coc  
30ms - pot  
2ms - gramma  
4ms - blanda  
30ms - four  
TOTAL: 353ms.

ARRAY METHODS SOLUTION

32ms - tomate  
25ms - citron  
12ms - salade  
14ms - oignon  
6ms - boeuf  
2ms - poulet  
43ms - des  
172ms - les  
7ms - coc  
30ms - pot  
2ms - gramma  
4ms - blanda  
30ms - four  
TOTAL: 404ms.