AGRANE Aurélie 22000445

PROGRAMMATION HUMANITE NUMERIQUE
DEVOIRS 5

Pour ce devoir, j'ai choisi un poème de Paul Verlaine intitulé « A celle qu'on dit froide ».

Exercice 1 :

```
>> let poem= `A celle qu'on dit froide

    Tu n'es pas la plus amoureuse
    De celles qui m'ont pris ma chair ;
    Tu n'es pas la plus savoureuse…
<- undefined
```

Pour commencer, il nous faut déclarer le poème. Pour cela il faut encadrer notre poème avec « ` » pour garder sa mise en forme.

Exercice 2 :

Pour la suite, il nous a fallu trouver les 5 mots les plus fréquents dans le poème.

```
>> let mots = poem.toLowerCase().match(/\b\w+\b/g);
<- undefined
>> let frequence = {};
<- undefined
>> mots.forEach(mots => {
       frequence[mots] = (frequence[mots] || 0) + 1;});
<- undefined
>> let tris = Object.keys(frequence).sort((a, b) => frequence[b] - frequence[a]);
<- undefined
>> let top5 = 5;
   for (let i = 0; i < top5; i++) {
       console.log(`Mots: ${tris[i]}, Fréquence: ${frequence[tris[i]]}`);}
   Mots: de, Fréquence: 12
   Mots: d, Fréquence: 12
   Mots: et, Fréquence: 9
   Mots: jusqu, Fréquence: 9
   Mots: les, Fréquence: 6
<- undefined
```

Voici les étapes :

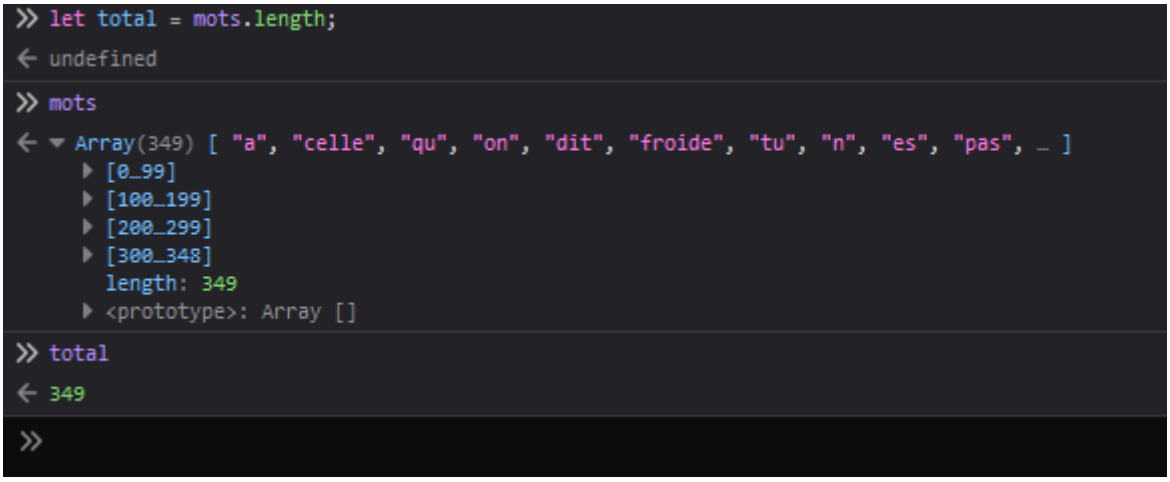
- On place notre poème sous forme de tableau en le déclarant avec « mots ».
- Déclarer « fréquence », qui est un ensemble vide pour nous permettre de stocker ce qu'il va suivre.

- On utilise « forEach » qui va nous permettre de répéter tous les éléments du tableau « mots ».
- On va ensuite trier les mots du poème en fonction de leur fréquence d'apparition. Cela va du plus fréquent au moins fréquent.
- On utilise ensuite la boucle « for » afin d'avoir un affichage des mots les plus fréquent que l'on stocke dans un tableau. On affiche alors les 5 premiers mots de ce tableau.

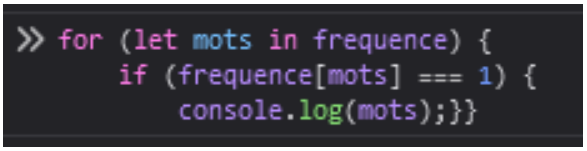
On obtient alors comme résultats :

1. de
2. d
3. et
4. jusqu
5. les

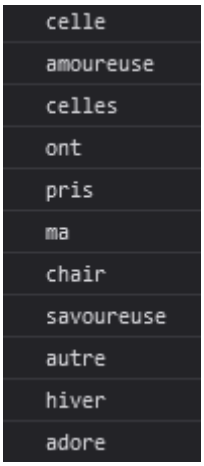
On veut ensuite calculer la richesse de notre poème, c'est-à-dire compter le nombre de mots total et ceux qui n'apparaissent qu'une seule fois.



Pour connaitre le nombre de mots on a le choix. Soit on entre tout simplement « mots » qui va nous montrer tous les éléments présents dans le tableau ou bien on utilise le code : « let total= mots.lenght ; ».
Notre poème est composé de 349 mots.



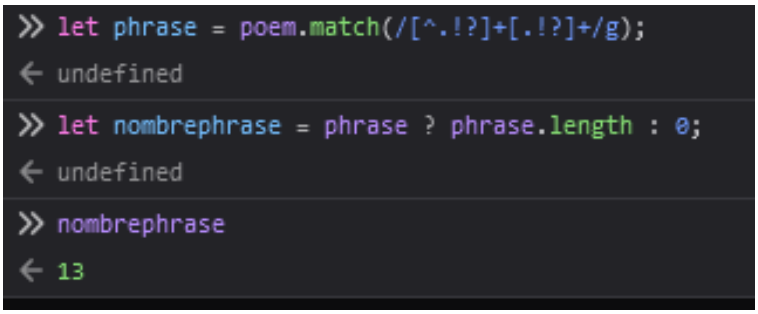
Pour compter les mots uniques, c'est-à-dire ceux qui n'apparaissent qu'une seule fois dans le texte, on utilise la boucle « for…in » qui va chercher dans notre tableau. « if …. ===1 » nous permets de vérifier si la condition de 1 fréquence est validé. Si c'est le cas, alors cela veut dire que le mot n'apparait qu'une seul fois dans notre poème.



Voici donc une partie de la liste de mots unique que l'on obtient.

Exercice 3 :

Pour cette partie, nous voulons connaitre le nombre de phrase que contient notre poème.



On fait appel à une expression régulière pour nous aider à trouver les phrases de notre poème.
Notre expression régulière nous permet de chercher une séquence de caractères qui possède les délimitateurs : «., !, ou ? ».

Par la suite, on utilise une expression ternaire qui va nous permettre de trouver les phrases dans notre poème.

Notre poème est donc composé de 13 phrases.

Exercice 4 :

Combien de strophe compose notre poème ?

```
>> let strophe = poem.trim().split('\n\n');
<- undefined
>> let stophetotal = strophe.length;
<- undefined
>> stophetotal
<- 16
```

Ce code nous permet de trouver les strophes de notre poème.
On commence par retirer tous les espaces en début et en fin de chaque chaîne. On indique que les strophes sont délimitées par « \n, \r ». (Que l'on a obtenu après avoir mit notre poème sous forme de tableau.)
On compte ensuite le nombres de strophes qui sont présentent.

Notre poème compte 16 strophes.

Typologie du poème :

Pour faire la typologie de notre poème nous aurons besoin du code suivant :

```
>> let typesStrophes = {};
<- undefined
>> let typesVers = {};
<- undefined
>> strophes.forEach((strophe, index) => {
        let nombreLignes = (strophe.match(/\n/g) || []).length + 1;
        typesStrophes[`Strophe ${index + 1}`] = nombreLignes;

        let syllabesParVers = strophe.split('\n').map(ligne => ligne.split(' ').reduce((acc, mot) => acc + (mot.match(/[aeiouy]/ig) || []).length, 0));
<- undefined
```

On définit des espaces vides « typesStrophes » et « typesvers ».
On analyse ensuite toutes les lignes de chaque strophe du poème pour en compter le nombre de syllabes.

(strophe.match(/\n/g) || []).length + 1; → Utilise une expression régulière pour compter le nombre de ligne en recherchant les sauts de ligne. Puis on stocke tout cela dans notre ensemble vide « typesStrophes ».

On divise ensuite tout en lignes puis en mots pour compter le nombre de syllabes. On stocke le tout dans « syllabesParVers ».

Pour finir, nous parcourons tous ce qui se trouve dans le tableau « syllabesParVers » que l'on transfers dans l'ensemble vide « typesvers » que l'on a créé un peu plus tôt.

Voici les résultats que nous obtenons :