```
window.onload = function() {
    let fileInput = document.getElementById('fileInput');
    let fileDisplayArea = document.getElementById('fileDisplayArea');

    // On "écoute" si le fichier donné a été modifié.
    // Si on a donné un nouveau fichier, on essaie de le lire.
    fileInput.addEventListener('change', function(e) {
        // Dans le HTML (ligne 22), fileInput est un élément de tag "input" avec un attribut type="file".
        // On peut récupérer les fichiers données avec le champs ".files" au niveau du javascript.
        // On peut potentiellement donner plusieurs fichiers,
        // mais ici on n'en lit qu'un seul, le premier, donc indice 0.
        let file = fileInput.files[0];
        // on utilise cette expression régulière pour vérifier qu'on a bien un fichier textString.
        let textType = new RegExp("text.*");

        if (file.type.match(textType)) { // on vérifie qu'on a bien un fichier textString
            // lecture du fichier. D'abord, on crée un objet qui sait lire un fichier.
            var reader = new FileReader();

            // on dit au lecteur de fichier de placer le résultat de la lecture
            // dans la zone d'affichage du textString.
            reader.onload = function(e) {
                fileDisplayArea.innerText = reader.result;
            }

            // on lit concrètement le fichier.
            // Cette lecture lancera automatiquement la fonction "onload" juste au-dessus.
            reader.readAsText(file);

            document.getElementById("logger").innerHTML = '<span class="infolog">Fichier chargé avec succès</span>';
        } else { // pas un fichier textString : message d'erreur.
            fileDisplayArea.innerText = "";
            document.getElementById("logger").innerHTML = '<span class="errorlog">Type de fichier non supporté !</span>';
        }
    });
}
function segmentation() {

    const texte = document.getElementById(`fileDisplayArea`).innerText;
    const delim = document.getElementById(`delimID`).value;

    // on remplace tous les caractères de fin de ligne ou tabulation (éventuellement à compléter) par des espaces
    texte0 = texte.replace(RegExp("(\\n|\\r|\\t)","g"),' ');

    /* la suite tient compte de la casse; si on ne le veut pas, il suffit d'ajouter :
    texte0 = texte0.toLowerCase();                                                    */

    mots = multipleSplit(texte0, delim); // voir explications fonction ci-dessous
    nMots = mots.length;

    // on met les mots dans un objet, pour calculer la fréquence des mots
    freqMots = [];
    mots.forEach (m => {
        if (freqMots[m]) {
            freqMots[m]++;
        } else {
            freqMots[m]=1;
        }
    });

    // on met freqMots dans longMots pour y ajouter la longueur de chaque mot
    let longMots = Object.entries(freqMots);
    while (longMots[i] != undefined) {longMots[i][2]=longMots[i][0].length;i++;}

    // on trie longMots en fonction de la longueur croissante, puis alphabétique
```

```
66      longMots.sort((a, b) => ((100+a[2]+a[0]) > (100+b[2]+b[0]) ? 1 : -1));
67
68      analyse = `Le texte comprend ${nMots} mots.`;
69
70      analyse += `\n\nListe des mots triés par longueur :`;
71
72      /* pour construire un tableau bien colonné, j'ai développé une fonction
73      ajoutant des espaces insécables (Ascii 160) à gauche (nombres) ou à droite
        (texte),
74      car les espaces simples ( Ascii 32) sont "tassés" dans html;
75      j'ai également chois la police Courier New dans le fichiers css.
76      NB : je n'ai pas réussi à gérer un tableau HTML dans 'page-analysis' en
        javascript */
77      mot = ajoutEspace('Mot', 15, 'droite') ;
78      longueur = ajoutEspace('Longueur', 10, 'gauche') ;
79      frequence = ajoutEspace('Fréquence', 10, 'gauche') ;
80
81      analyse += '\n' + '-'.repeat(42);
82      analyse += '\n| ' + mot + '|' + longueur + ' |' + frequence + ' |';
83      analyse += '\n|' + '-'.repeat(40) + '|';
84      i = 0;
85      while (longMots[i] != undefined) {
86          mot = ajoutEspace(longMots[i][0], 15, 'droite') ;
87          longueur = ajoutEspace(longMots[i][2], 10, 'gauche') ;
88          frequence = ajoutEspace(longMots[i][1], 10, 'gauche') ;
89          analyse += '\n| ' + mot + '|' + longueur + ' |' + frequence + ' |';
90          i++;
91      };
92      analyse += '\n' + '-'.repeat(42);
93
94      let pageAnalysis = document.getElementById(`page-analysis`);
95      pageAnalysis.innerText = analyse
96
97  }
98  function multipleSplit(textString, separators) {
99  /* je n'ai pas réussi à mettre au point un regex, à travers la fonction RegExp,
100 permettant de faire un split sur plusieurs séparateurs, ceci à cause des caractères
    génériques;
101 j'ai donc développé une fonction sans
    regex                                                    */
102
103     let text0 = textString;
104
105     separators = ' ' + separators.replace(' ',''); // la boucle ci-dessous doit finir
        par le caractère espace
106     for (i=textString.length-1;i>0;i--) {text0=text0.split(separators[i]).join(
        separators[i-1]);};
107
108     /* remplacement de toutes les séquences de plusieurs espaces par un seul espace,
109     suppression des espaces à gauche et à droite, puis split sur le caractère espace
        */
110     words = text0.replace(/ +/g, ' ').trim().split(' ');
111
112     return words;
113
114 }
115 function ajoutEspace(texte, longueur, cote) {
116
117     nbsp = String.fromCharCode(160);
118     if (cote[0].toLowerCase() === 'g') {
119         texte0=nbsp.repeat(longueur)+texte;
120         texte0=texte0.slice(texte0.length-longueur);}
121     else {
122         texte0=texte+nbsp.repeat(longueur);
123         texte0=texte0.slice(0, longueur);}
124
125     return texte0;
126 }
```