


DESCRIPTION D'UNE MISSION BTS SIO		
Prénom – Nom	Quentin Coqueran	N° mission
Option	SLAM	
Situation		Formation
Lieu de réalisation	Campus Montsouris 2 Rue Lacaze – 75014 PARIS	
Période de réalisation	Mars 2020	Mars 2020
Modalité de réalisation	VÉCUE	

Intitulé de la mission	Titre de la mission
	Création d'un jeu « pendu »
Description du contexte de la mission	Description en 2 à 3 lignes maxi
	Créer une page où l'utilisateur doit trouver le mot caché, généré aléatoirement à partir d'un array en javascript et compter son nombre de tentatives.

Ressources et Outils utilisés	Liste des ressources disponibles et outils utilisés (Documentations, Matériels et Logiciels)
	<ul style="list-style-type: none"> - WampServer - Visual Studio Code - Documentation sur les sites internet comme MDN, W3C... pour trouver les fonctions nécessaires à la réalisation de ce projet
Résultat attendu	Résultat attendu avec la réalisation de cette mission Mot généré aléatoirement en JS et compter son nombre de tentatives pour modifier l'image du pendu en fonction du nombre d'erreur.
Contraintes	Contraintes : techniques budgétaires temps O.S. ou outils imposés...

	<ul style="list-style-type: none">• Vérifier si la lettre envoyée par l'utilisateur existe bien dans le mot et l'afficher.• Compter le nombre de tentative de l'utilisateur.• Afficher les mauvaises lettres choisies par l'utilisateur.
--	--

Compétences associées (Voir tableau)	Liste des intitulés du tableau de compétences (avec les références)
	<p>A1.1.1 Analyse du cahier des charges d'un service à produire</p> <p>A1.2.2 Rédaction des spécifications techniques de la solution retenue</p> <p>A1.3.4 Déploiement d'un service</p> <p>A1.4.1 Participation à un projet</p> <p>A1.4.2 Évaluation des indicateurs de suivi d'un projet et justification des écarts</p> <p>A2.3.2 Proposition d'amélioration d'un service</p> <p>A4.1.1 Proposition d'une solution applicative</p> <p>A4.1.4 Définition des caractéristiques d'une solution applicative</p> <p>A4.1.7 Développement, utilisation ou adaptation de composants logiciels</p>

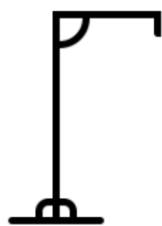
Description simplifiée des différentes étapes de réalisation de la mission en mettant en évidence la démarche suivie, les méthodes et les techniques utilisées
<p><u>Chapitre 1 : Développement de la partie HTML du projet</u></p> <p><u>Etape :</u></p> <p>1.1/ Dans ce projet il y'a très peu de HTML, tout d'abord nous créons le fichier index.html qui sera la seule page html de ce projet. Ensuite nous relions notre page au fichier style.css et script.js afin de relier le css et le javascript sur la page index.html.</p> <pre data-bbox="220 1489 1011 1568"><link rel="stylesheet" type="text/css" href="style.css"> <script src="script.js" type="text/javascript"></script></pre> <p>1.2/ Une fois le fichier crée et les fichiers css et js relié, nous mettons l'image du pendu par défaut et on écrit le nombre d'essai restant par défaut aussi, c'est-à-dire 7.</p> <pre data-bbox="220 1680 1018 1787"><div id="img"></div> <div id="text">Vous avez 7 chances</div> <p id="mot"></p></pre> <p>Ensuite nous créons le formulaire afin que l'utilisateur puisse saisir une lettre.</p> <pre data-bbox="220 1859 1375 1953"><form> <label for="letter">Saisissez une lettre :</label>
 <input type="text" class="letter" oninput="remplacerLettres(motSelectionner, document.getElementById('letter').value);" id="letter" maxlength="1"> <!-- active la fonction remplacerLettres() qui va prendre en parametre la valeur du champ "letter" et le motSelectionner --> </form></pre> <p>Le champs input possède l'attribut « oninput » qui va exécuter le code javascript a chaque nouvelle entrée dans le champ, ce qui convient parfaitement pour l'utilisation de</p>

notre pendu. En effet a chaque fois que l'utilisateur va rentrer une lettre, la fonction « remplacerLettres » qui prend en paramètre le mot sélectionner aléatoirement et la valeur contenue dans le champ va s'activer et l'attribut maxlength permet de bloquer le nombre de lettre a 1 pour que l'utilisateur ne puisse saisir qu'une lettre a la fois.

Et enfin nous créons une div pour afficher les mauvaises lettres.

```
<div>
  <br>
  <span id="badLetter">Mauvaise lettres :</span>
  <span id="wrongLetter"></span>
</div>
```

Résultat :



Vous avez 7 chances

Saisissez une lettre :

Mauvaise lettres :

Chapitre 2 : Développement de la partie JS du projet

Etape :

2.1/ Dans un premier temps nous allons nous occuper de générer le mot à trouver, pour cela nous allons créer la fonction « choisirMot » qu'on va placer dans un attribut « onload » de la balise body afin d'activer cette fonction dès le chargement de la page.

```
<body onload="choisirMot();">
```

```
function choisirMot()
{
  var mots = ["pasteque", "poisson", "pomme", "garage", "chaise", "velo", "café", "poubelle", " tiroir", "carton", "table", "feuille", "verre"];
  motSelectionner = mots[Math.floor(Math.random() * mots.length)]; // mot généré aleatoirement
  motCacher(motSelectionner); // va lancer la fonction mot Cacher qui va nous permettre de définir le nombre de trait du mot
  return motSelectionner; // retourne le mot choisie
}
```

Nous créons un tableau rempli de mot, puis a l'aide de la fonction Math.floor() et Math.random() nous allons sélectionner un numéro au hasard pour retourner un index de la liste définie juste au-dessus. Une fois le mot sélectionner nous utilisons la fonction motCacher pour afficher le nombre de trait que contient le mot.

```
function motCacher(motRandom)
{
    for (i = 1; i <= motRandom.length; i++) // pour chaque lettre du mot selectionner
    {
        lettreCacher += "-"; // rajoute un tiret a chaque tours
        document.getElementById('mot').innerHTML = lettreCacher // rajoute le tiret a l'id "mot"
    }
}
```

Cette fonction comptabilise le nombre de lettre et va pour chaque lettre rajouter un « - » a la variable lettreCacher qui va ensuite être afficher dans la div contenant l'id « mot »

Résultat :

2.2/ Dans un second temps nous allons créer la fonction remplacerLettres() qui sera activé a chaque saisie dans l'input pour rajouter une lettre du mot, ou une erreur.

```
function remplacerLettres(mot, lettres)
{
    lettres = lettres.toLowerCase(); // transforme les majuscules en minuscules
    if(testerLettreDansMot(lettres, mot)) // si la lettre existe bien dans le mot
    {
        if(!finish)
        {
            for (var i = 0; i <= mot.length -1; i++) // pour chaque lettre du mot selectionner
            {
                if(lettres == mot.charAt(i))
                {
                    lettreTrouver = document.getElementById('mot').innerHTML;
                    if(lettres == lettreTrouver.charAt(i)) // si la lettre n'a pas déjà etais saisie
                    {
                        alert('cette lettre a déjà été proposée');
                    }

                    var index = mot.indexOf(lettres, compteur); // va recuperé le ou les index de la lettre saisie
                    compteur = index + 1; // utile uniquement pour les doubles lettres permet de placer l'indice de debut après la premiere lettre

                    document.getElementById('mot').innerHTML = document.getElementById('mot').innerText.replaceAt(index, lettres);
                    // pour pouvoir remplacer les tirets par des lettres nous devons changer le innerText pour pouvoir modifier uniquement le "string"
                }
            }
        }
    }
}
```

C'est cette fonction qui va faire fonctionner tout le mécanisme du jeu, tout d'abord nous transformons la lettre saisie par l'utilisateur en minuscule car cela pourrait poser un problème lors de la comparaison entre deux lettres, ensuite nous lançons une fonction booléenne testerLettreDansMot() qui renvoie true si la lettre existe bien dans le mot à trouver.

```
function testerLettreDansMot(lettre, chaine)
{
    document.getElementById('letter').value = "";
    for (var i = 0; i < chaine.length; i++) // fait tourner la boucle le nombre de fois qu'il y a de lettre dans la chaine
    {
        if(chaine.charAt(i) == lettre) // selectionne chaque lettre de la chaine individuellement
        {
            return true;
        }
    }
}
```

Tout d'abord nous effaçons le contenu de l'input pour que l'utilisateur puisse automatiquement mettre une nouvelle lettre sans avoir à supprimer le contenu, et ce que la lettre soit bonne ou mauvaise, ensuite nous faisons une boucle for qui va comparer la lettre de l'utilisateur à chaque lettre du mot à trouver grâce à la fonction charAt() et va renvoyer true si au moins une lettre correspondent.

Si la fonction testerLettreDansMot() renvoie bien true alors nous rentrons dans la boucle. Une fois dans la boucle nous faisons une condition *si finish = false*, la variable finish et une variable booléenne qui est définie par défaut en false et qui ne sera true que lorsque

la partie sera fini, c'est-à-dire lorsque l'utilisateur aura perdu donc obtenue 7 erreurs ou lorsque l'utilisateur aura gagné donc trouver toutes les lettres. Donc si la partie n'est pas fini nous créons une boucle qui va tourner autant de fois que le mot a trouvé possède de lettre et va vérifier de la même façon que la fonction testerLettreDansMot() si la lettre fournis a déjà était proposé, car si la lettres a déjà était proposé alors il peut y avoir un problème il faut donc empêcher l'utilisateur de mettre plusieurs fois une bonne lettre. Si un mot possède plusieurs fois la même lettre il faut également que les lettres s'affichent à tous les endroits où ils sont, donc nous allons récupérer l'index de la première lettre et l'additionner a 1 puis la stocker dans la variable compteur, cela va permettre lors du tour suivant de récupérer l'index de cette deuxième lettre, et enfin grâce à la variable replaceAt nous remplaçons le tiret par la lettre choisie à partir de l'index précédemment récupéré.

Résultat :

po-ss-

2.3/ Si l'utilisateur a trouvé toutes les lettres du mot alors il gagne nous devons donc changer l'image actuelle du pendu et écrire « Gagné ! » puis laisser l'utilisateur recommencer en lui affichant un bouton « recommencer la partie ».

```
if(document.getElementById('mot').innerHTML == motSelectionner) // si toute les lettres trouvé son egal au mot
{
    document.getElementById('img').innerHTML = "<img src='images/gagner.png'>";
    document.getElementById('text').innerHTML = "<span style='color: green'>Gagné !</span>";
    document.getElementById('again').style.display = "block"; // fait apparaître le bouton "recommencer"
    finish = true;
    if(nbErreur == 0) // si l'utilisateur n'a aucune erreur
    {
        document.getElementById('badLetter').style.display = "none";
        document.getElementById('wrongLetter').innerHTML = "Sans aucune faute ! Bravo !!!";
    }
}
```

Nous vérifions si l'accumulation de toutes les lettres trouvées par l'utilisateur est égale au mot a trouver, si c'est le cas alors nous changeons l'image et nous affichons en vert « Gagné ! », nous faisons apparaître le bouton « recommencer la partie » qui était caché jusqu'à maintenant, puis nous définissons la variable finish en true pour que la partie soit considéré comme fini et nous allons même un peu plus loin, supposons que l'utilisateur ne fasse aucune faute et que la partie « mauvaises lettres » est vide, nous pouvons lui supprimer cette partie, et même lui rajouter une phrase en plus « Sans aucune faute ! Bravo !!! ».

Résultat :

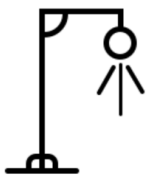


2.4/ Si l'utilisateur rentre une mauvaise lettre et que nous n'avons pas encore utilisé nos 7 chances alors la fonction testerLettreDansMot() va renvoyer false et nous allons rentrer dans une boucle else de la fonction remplacerLettres().

```
else if(nbErreur < 7) // si la fonction testerLettreDansMot renvoie false et que le nombre d'erreur est inférieur à 7
{
    if(!finish) // si la partie n'est pas finie
    {
        nbErreur++;
        nbChance--;
        wrongLetter += lettres;
        document.getElementById('img').innerHTML = "<img src='images/pendu"+nbErreur+".png'>";
        document.getElementById('text').innerHTML = "Plus que " + nbChance + " chance";
        document.getElementById('wrongLetter').innerHTML = " " + wrongLetter;
    }
}
```

Une fois dans la boucle nous faisons une condition *si finish = false*, donc si la partie n'est pas finie nous incrémentons la variable nbErreur qui va comptabiliser le nombre d'erreur, décrétons la variable nbChance qui va comptabiliser le nombre de chance de l'utilisateur et enfin nous rajoutons la lettre à la variable wrongLetter qui elle va stocker toutes les mauvaises lettres saisies par l'utilisateur et enfin nous affichons chaque variable à l'endroit où elle doit être positionnée.

Résultat :



Plus que 3 chance

Saisissez une lettre :


Mauvaise lettres : abcd

2.5/ Si l'utilisateur a atteint 7 erreurs alors il perd nous devons donc écrire « Perdu ! » puis laisser l'utilisateur recommencer en lui affichant un bouton « recommencer la partie ».

```
if(nbErreur == 7)
{
    document.getElementById('text').innerHTML = "<span style='color: red'>Perdu !</span>";
    document.getElementById('mot').innerHTML = "le mot à trouver était : " + motSelectionner;
    document.getElementById('again').style.display = "block";
    finish = true;
}
```

On affiche « Perdu ! » et on affiche le mot qui était à trouver puis nous faisons apparaître le bouton « recommencer la partie » qui était caché jusqu'à maintenant, et enfin nous définissons la variable finish en true pour que la partie soit considérée comme finie.

Résultat :



Perdu !

Perdu !


le mot a trouvé etais : verre

Saisissez une lettre :

Mauvaise lettres : abcd fgh

Recommencer la partie

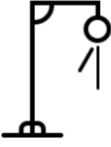
Résultat final :



Vous avez 7 chances

Saisissez une lettre :

Mauvaise lettres :

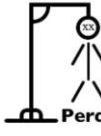


Plus que 4 chance

-able

Saisissez une lettre :

Mauvaise lettres : cso



Perdu !


Perdu !

le mot a trouvé etais : table

Saisissez une lettre :

Mauvaise lettres : csonjhx

Recommencer la partie



Bravo !


Gagné !

pasteque

Saisissez une lettre :

Mauvaise lettres : k

Recommencer la partie



Bravo !

Gagné !

poisson

Saisissez une lettre :

Sans aucune faute ! Bravo !!!

Recommencer la partie

Conclusion	Que pouvez-vous dire de cette mission : apport personnel, expérience, etc.
	<p>Cette mission m’a permis d’apprendre les bases en javascript et m’a permis de bien comprendre le fonctionnement de JavaScript et ses fonctions.</p>