TP Contrôle JavaScript

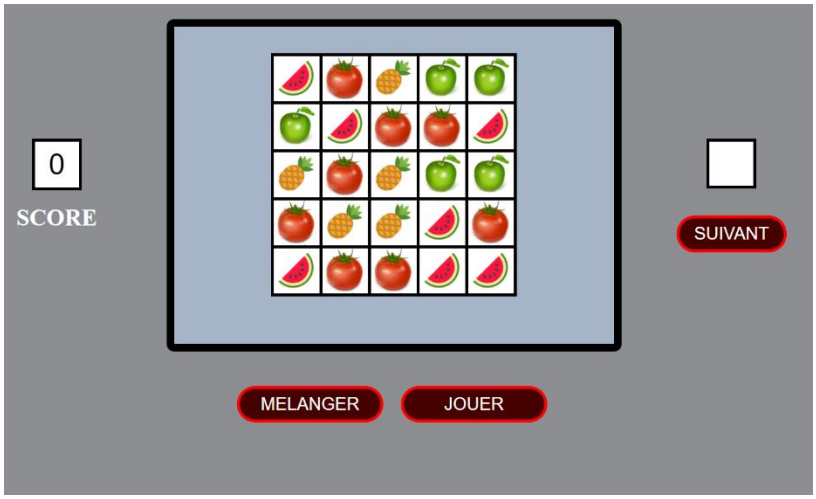
MEMORY

Le travail proposé consiste à coder quelques fonctions javascript participant au fonctionnement d' un jeu de "memory" dont le code html et le CSS sont fournis.

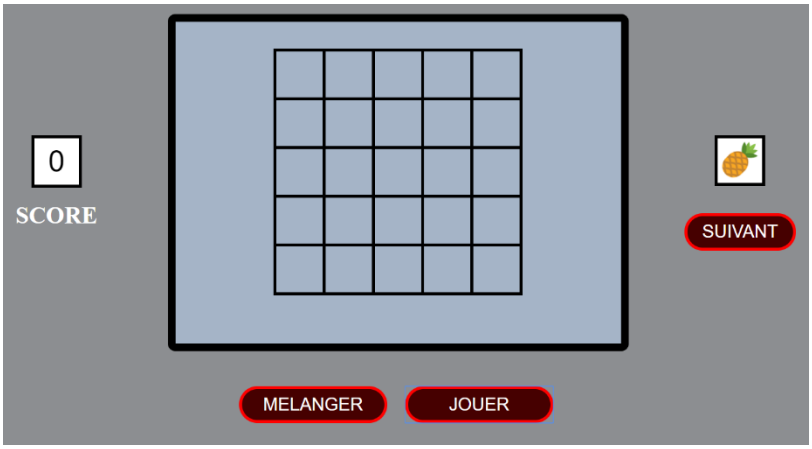
Principe du jeu :

Des fruits sont répartis aléatoirement dans un plateau de jeu de 5 x 5. Il faut mémoriser leurs positions respectives avant d'appuyer sur "jouer"

Vue au lancement du jeu :

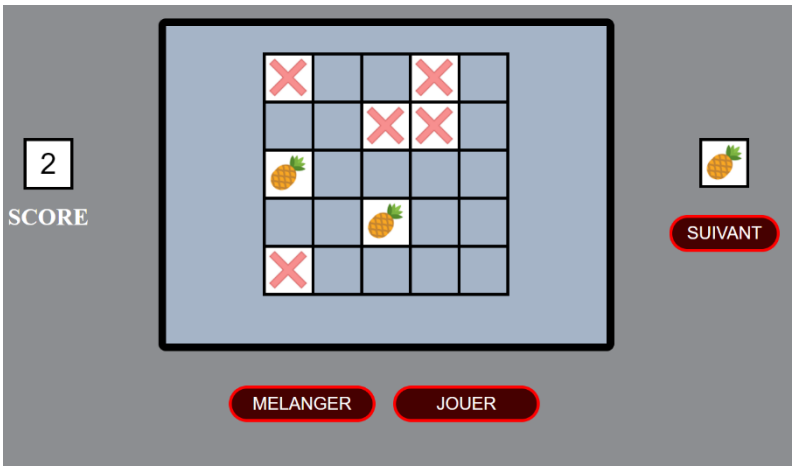


L'appui sur le bouton "JOUER" a pour effet de faire disparaitre les fruits du plateau et d'en faire apparaitre un aléatoirement dans la case située au-dessus du bouton 



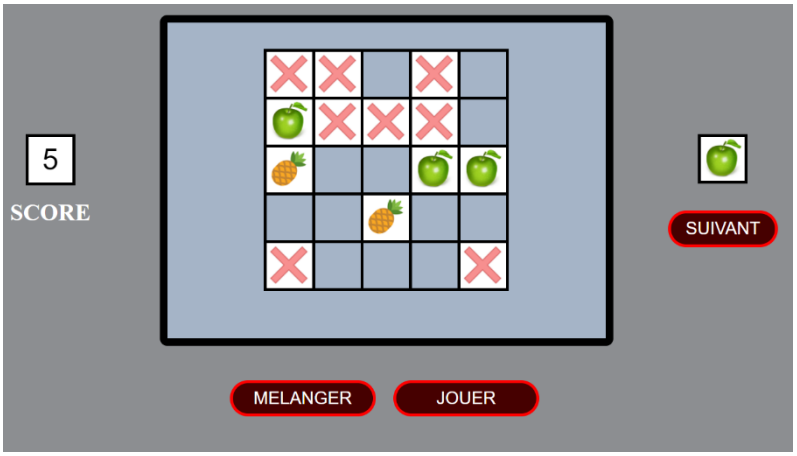
Le joueur doit alors cliquer sur les cases où le fruit proposé est caché. Si une bonne case est cliquée, le fruit apparait et le score augmente de 1 point, sinon, c'est une croix rouge qui apparait. Une case qui a été cliquée ne peut plus être modifiée.

Exemple d'évolution du jeu :

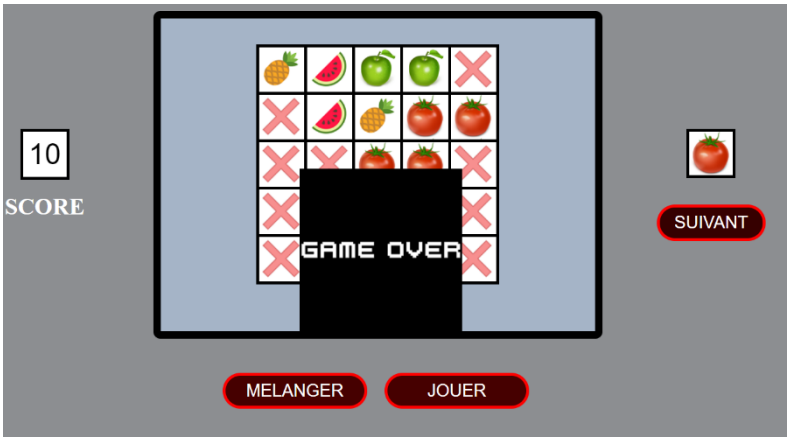


Lorsque le joueur pense avoir trouvé tous les emplacements du fruit proposé, il clique sur

et un nouveau fruit apparait :



Lorsque les 25 cases ont été cliquées, une image "game over" apparait et le jeu est bloqué :



Travail demandé

Votre travail va consister uniquement à compléter des fonctions Javascript. **Les fichiers HTML et CSS fournis ne doivent en aucun cas être modifiés.** La totalité du travail est à effectuer dans le fichier JSmemory.js. Les parties de code à créer, modifier, ou commenter sont repérées par des commentaires.

Pour commencer, ouvrez le fichier JSMemory.js **et commencez par écrire vos nom et prénom en commentaire sur la ligne 1**. Vous pouvez ensuite faire afficher la page memory.html et constater que l'image game over est déjà présente…

Git init !

1) Compléter la fonction init() de façon à faire disparaitre l'image "game over" contenue dans le bloc repéré par l'id "over" . Vous pourrez pour cela utiliser l'attribut de style "visibility".

Commit 1

2) Compléter la fonction init() de façon à faire apparaitre le contenu de la variable "score" (déjà initialisée à zéro au début de la fonction, ligne 6) dans l'input de type="text" d'id="score".

Commit 2

3) Compléter la fonction "alea()" qui doit retourner un nombre entier aléatoire compris entre 0 et 3.

Afin de vérifier que votre fonction retourne bien la valeur souhaitée, vous pouvez temporairement décommenter la ligne "alert(p)" de la fonction init() et actualiser plusieurs fois votre page pour vérifier que le contenu de l'alerte est bien compris entre [0 et 3]. Vous pouvez aussi utiliser la console des outils développeurs de votre navigateur et user d'un console.log(p) si vous préférez.

Commit 3

4) Compléter la fonction initSymbole(S) qui fera afficher aléatoirement, dans le background du d'id = "symbole" (situé au-dessus du bouton "suivant"), un des quatre fruits.

   Pour ce faire, il faut:

   - ➔ Sélectionner une image aléatoirement dans de tableau de symboles(défini ligne 7) passé en paramètre, en vous aidant de votre fonction alea(). Si la fonction alea() ne fonctionne pas, choisissez le premier symbole du tableau.
   - ➔ Reconstituer la chaine de caractère que le backgroundImage devra interpréter. Par exemple, pour ananas.png, la chaine devra de présenter sous la forme "url('Images/ananas.png')"
   - ➔ Associer l'url ainsi constituée au correspondant

Commit 4

5) Coder la fonction creerTabSymb(T,S) qui retournera un tableau de 5x5, et dans laquelle la position des fruits, aléatoirement distribuée, sera mémorisée. Pour remplir cette table, on passe en paramètres à cette fonction le tableau vide (tabSymbolesVide) et le tableau de 1x4 (symboles) contenant les images des fruits; Le tableau "tabSymbolesVide" est un tableau à 2 dimensions défini dans la fonction init(), de sorte que tabSymbolesVide[0][0] désigne la case située sur la première ligne et première colonne. Le tableau "symboles" est toujours celui défini ligne 7.

Pour coder cette fonction, il faut :

➔ Balayer la totalité du tableau vide et…

- Pour chaque case du tableau vide, associer aléatoirement un élément de la table symbole (pour le remplir !) en s'aidant de la fonction aléa(). (Si la fonction alea() ne fonctionne pas, remplissez la première et la deuxième ligne du tableau avec le premier symbole et les trois lignes suivantes avec les trois autres symboles.)

Remarque : à ce stade ce code est sans effet sur la page html. Toutefois, vous pouvez toujours utiliser le mode console de votre navigateur pour vérifier que votre fonction s'est bien comportée. Sinon, vous aurez la surprise après la question suivante.


Commit 5

6) Coder la fonction affichage(tabHTML,tabSymb) qui reçoit en paramètre le tableau html défini dans la fonction init() et le tableau de symboles, et qui, à chaque case du tableau html associe le backgroundImage avec l'image correspondante contenue dans tabSymbole. (Attention, il faut reconstituer l'url !)


Commit 6

7) Coder la fonction play() qui doit

➔ cacher tous les fruits du plateau. Cela revient à effacer le backgroudImage de toutes les cases que vous venez de faire apparaitre…

- Remarque : la fonction play() ne reçoit pas de paramètres mais pourra utiliser la variable globale tabYX qui contient le tableau HTML

➔ Faire apparaitre un fruit dans la case symbole. Il suffit pour cela d'appeler la fonction initSymbole(symboles) précédemment codée. A ce stade, il faut commenter l'appel à initSymbole() dans la fonction init();


Commit 7

8) Coder la fonction next(), appelée par un clic sur le bouton suivant, qui fait afficher le fruit (et non un fruit choisi aléatoirement) suivant dans la case symbole.

- Remarque : la fonction next() ne reçoit pas de paramètres mais pourra utiliser les variables globales symboles et p


Commit 8

9) BONUS : Codage des fonctions verif() et gameOver()

Théoriquement, la fonction verif() doit permettre de vérifier si la case cliquée correspond bien au fruit présent dans la case symbole. Si c'est le cas, le score augmente et on affiche le fruit, sinon, on affiche une croix rouge. Dans les 2 cas, on comptabilise le clic afin de déterminer si la partie est finie ou pas. Dans le cas ou le joueur clique sur une case précédemment cochée, le clic ne doit pas être comptabilisé. Enfin, après 25 clics validés, la fonction gameOver() est appelée

9-1 Codez la fonction verif(pos) appelée lorsqu'une case est cliquée et qui reçoit l'id de cette case en paramètre. Cette fonction devra

- Vérifier que le symbole contenu de la case cliquée est le même que celui affiché au-dessus du bouton "suivant"
    - Si oui :
        - augmenter le score
        - Afficher le symbole
    - Si non :
        - Afficher une croix

Dans tous les cas, il faudra comptabiliser le nombre de click et appeler la fonction gameOver() après le 25ème click


Commit 9

9-2 Codez la fonction gameOver() qui devra faire apparaitre, après 2 secondes, l'image que vous vous êtes évertués à faire disparaitre à la question 1


Commit 10

Quel que soit le nombre de fonctions codées, vous devez déposer votre dossier sur Github et le partager avec votre professeur… Le dossier devra également être déposé dans votre compte perso sur le serveurir1