WEB –TP12

Javascript 4 / Git

L'objectif de ce TP est de réaliser un jeu de pousse-pousse. Le suivi de version sera assuré par GIT et vous **nommerez impérativement votre dossier "TP12_VotreNom"**. L'aspect final de la page est donné en annexe en fin deTP.

I Mise en place du HTML et du CSS associé

-1- Test des liens

Créer un fichier html qui fera appel dans son entête à un fichier de script et un fichier css.

- ➔ Vérifier que l'appel du script est fonctionnel
- ➔ Vérifier que l'appel du CSS est fonctionnel

Commit n°1

-2- Squelette

La page sera composée de trois zones situées les unes en dessous des autres et dont les contenus seront centrés en largeur.

La première zone de titre contiendra le texte "Un jeu d'enfant "

La deuxième zone contiendra la table de 7 X 4. Chaque case de la table sera identifiée par un id représentant la ligne et la colonne de cette case (id="26" pour la case située sur la ligne 2, 6ème colonne). Un évènement onClick sera également associé à chaque case et appellera une fonction "move()" qui passera l'id de la case en paramètres. Dans un second temps, la création de cette table sera confiée à une fonction JavaScript.

La dernière zone contiendra le bouton "Mélanger" faisant appel à une fonction "mix()".

Commit n°2

-3- CSS

Pour les différentes zones les caractéristiques CSS suivantes sont retenues :

Body : dégradé gris foncé vers gris clair, de gauche à droite.

Titre : Gras, Italic, Gris clair, de taille 60 px, police au choix.

Table : fond jaune, coins arrondis, bordure de 20px noire, cases de 100px X 100px, coins arrondis, bordure jaune 1px.

Bouton : texte de couleur blanche sur fond noir, 40px avec coins arrondis et un espace de 10px entre le texte et les bordures.

Commit n°3

-4- JavaScript

Le jeu va être structuré en diverses fonctions :

- ➔ Fonction getAllId() : fonction chargée de créer un tableau référençant tous les id des différents <td>.
- ➔ Fonction display(arr) : fonction recevant un tableau d'image (tabLettres) et chargée d'affecter un backgroundImage différent à chaque case de la table HTML.
- ➔ Fonction mix() : fonction chargée d'organiser les images du tableau tabLettres de façon aléatoire.
- ➔ Fonction getIdCaseVide() : fonction chargée retourner l'id (donc les coordonnées) de la case vide.
- ➔ Fonction toggle(id1, id2) : fonction chargée d'inverser les backgroundImage des cases dont les id sont passés en paramètres.
- ➔ Fonction move(letter) : fonction chargée de déplacer les lettres en haut, en bas, à droite, à gauche en fonction de le la lettre cliquée dans le jeu.

En outre, une fonction anonyme sera appelée sur le window.onload afin d'initialiser le jeu

-4-1- **Codage de la fonction getAllId()**

Coder la fonction getAllId() qui n'attend aucun paramètre et retourne un tableau contenant les 28 id des différents <td>. Pour ce faire, déclarer un "new array()" dans la fonction, récupérer un tableau de td (de type collection d' HTMLElement), puis pour chacun de ces <td>, récupérer son attribut "id" et enregistrer celui-ci dans le "array" qu'il faudra retourner une fois complété.

Appeler cette fonction dans le windows.onload et récupérer son retour dans une variable nommée tableauId que vous afficherez grâce à un console.log

 **Commit n°4**

-4-2- **Codage de la fonction display(arr)**

Coder la fonction display(arr) qui reçoit en paramètre un tableau d'id et qui, pour chaque id, fait correspondre un backgroundImage du dossier images fournit.

Appeler cette fonction dans le window.onload -à la suite de la fonction getAllid() – en lui passant la variable "tableauId" en paramètre.

 **Commit n°5**

La première case du jeu est censée être vide pour permettre le déplacement des lettres mais ce n'est pas le cas actuellement. Modifier votre fonction display(arr), de façon à avoir un backgroundImage="" à la place de la petite abeille…

 **Commit n°6**

4-3 **Codage de la fonction mix()**

Cette fonction a pour but de mélanger l'ordre des id du tableau "tableauId". Pour ce faire, elle commence par récupérer le tableau d'id grâce à la fonction getAllId() puis échange au hasard l'ordre des id dans tableauId. Une solution consiste à créer 2 nombres aléatoires n1 et n2 compris entre [1 et 27], puis à échanger le contenu de tableauId[n1] avec celui de tableauId[n2] (permutation circulaire). Il suffit ensuite de répéter cette opération une bonne centaine de fois.

Coder cette fonction (appelée par le bouton "mélanger") et la tester en pensant à ajouter l'appel à la fonction display(arr) à la fin.

 **Commit n°7**

4-4 **Codage de la fonction getIdCaseVide()**

Au fil du jeu, la case vide va se déplacer. Or pour savoir comment pousser les lettres, il est nécessaire de déterminer l'emplacement de cette case dans le jeu. La fonction getIdCaseVide() va donc tester le backgroundImage de chaque <td> jusqu'à trouver celui correspondant à la case vide et retourner l'id correspondant. Cette recherche nécessite l'utilisation de 2 boucles imbriquées : une première permettant de balayer les lignes, la seconde permettant de balayer chaque colonne dans chacune des lignes.

Coder et tester cette fonction en l'appelant à la fin de la fonction mix() et en effectuant un console.log de l'id retourné.

 **Commit n°8**

4-5 **Codage de la fonction toggle(id1, id2)**

Coder cette fonction qui ne fait qu'inverser les backgroundImage des cases dont les id sont passés en paramètre. Cette fonction se code très simplement par permutation circulaire.

Pour tester votre fonction appeler cette dernière dans la fonction move(letter). Le premier Id que vous passerez à votre fonction sera celui de la case cliquée (paramètre "letter") et le second sera celui de la case vide que vous récupérerez grâce à la fonction getIdCaseVide();

 **Commit n°9**

4-6 **Codage de la fonction move(letter)**

Le codage du jeu se termine… Il va maintenant falloir mobiliser les quelques neurones encore actifs qu'il vous reste pour coder cette fonction. Avant toute chose, vous pouvez mettre en commentaire le code que vous avez inséré dans cette fonction à la question précédente.

Pour vous guider, il est possible de suivre les étapes suivantes:

-1- déterminer si la case cliquée est sur la même ligne ou sur la même colonne que la case vide.

-2- déterminer le sens du décalage (vers la droite, la gauche, le haut ou le bas) en comparant les indices de colonnes des 2 cases (pour sens droite – gauche) ou les indices des lignes (pour le sens haut-bas)

-3- déterminer le nombre n de décalage à effectuer (en ligne ou en colonne selon)

-4- boucler n fois sur la fonction toggle(Id1, Id2) en commençant par passer l'indice de la case vide et l'indice de la case adjacente, et en incrémentant ces 2 indices à chaque itération.

Remarque : pour vous aider, il est conseillé de commencer par coder des fonctions getCol(id) et getRaw(Id) qui retourneront respectivement le numéro de colonne et le numéro de ligne de la case dont l'id est passé en paramètre.

4-6-1 Coder la fonction getCol(id) et la tester dans la fonction move(letter) en affichant dans un console.log la colonne de la case cliquée.

Commit n°10

4-6-2 Selon le même principe, coder et tester la fonction getRaw(id)

Commit n°11

4-6-3 Après avoir commenté les 2 questions précédentes, coder la fonction move(letter) pour 1 cas de figure uniquement (par exemple décalage à gauche)

Commit n°12

4-6-4 A partir de la question précédente, étendre le raisonnement à l'ensemble des cas de figures

Commit n°13

4-7 Amélioration

On peut imaginer que ce jeu soit décliné par la suite en un jeu de puzzle avec un nombre de case différent. Une fonction javaScript qui serait en mesure de réaliser la table automatiquement (en modifiant juste les paramètres "nombreDeLignes" et "nombreDeColonnes") serait idéale…

Commenter le code HTML compris entre les balises <table> et </table> et faire générer cette table dans le window.onload à l'aide d'une fonction createTable(nbRaw,nbCol)

Commit n°14

--- Annexe ---