* Vendredi 8 septembre
* Vendredi 12 octobre
* Mardi 6 novembre
* Mardi 13 novembre
* Vendredi 16 novembre
* Vendredi 23 novembre
* Vendredi 30 novembre
* Vendredi 7 décembre
* Planification
* 1er sprint
* 1er sprint (fin) et planification du 2e
* 2e sprint
* 2e sprint(fin) et planification du 3e
* 3e sprint
* 3e sprint (fin), Test (identification des bugs)
* Correction des bugs

Pour l’algorithmique des tous les jeux nous avons décidé de ne pas aller chercher des codes sources. Nous avons voulu relever le défi et trouver nous-mêmes les solutions.

On décide de tout commencer par la grille commune.

Dans HTML on crée une division contenant la grille.

Dans Java script on crée une fonction prenant en paramètre la première ou les 2 premières lettres du jeu, et selon ces dernières créera une grille de façon dynamique.

Function crGrille(type)

Pendu

Par exemple : crGrille(‘p’) qui ajoutera à la division grille un tableau d’une ligne et d’autant de colonnes qu’il y a de lettres dans le mot choisi au hasard dans le dictionnaire.

Une autre fonction se charge de créer une table de 27 valeurs contenant les lettres de l’alphabet et de l’ajouter à la division clavier.

Sudoku – sous-taches

1 Grille

Dans HTML on reste toujours dans la division grille pour chaque jeu.

Dans javascript :

Function crGrille(‘s’) qui ajoutera à la division grille une table de 9 colonnes et 9 lignes dont les cellules sont des champs de saisie d’un chiffre. Afin de simplifier la vérification des valeurs, on attribue des identifiants à chaque cellule selon la ligne et colonne.

Derniere cellule.id = 99

Premiere cellule.id = 11

2 vérification ligne colonne zone

Une fois qu’on a la grille en place, on va pouvoir saisir des valeurs aux champs.

A la saisie, une fonction se charge d’identifier la ligne, la colonne et la zone.

Function checkl(e,ligne,col) prend la cellule, la ligne et la colonne, crée un tableau de 9 valeurs et stocke toutes les valeurs de la ligne pour vérifier si la valeur saisie est présente. Ensuite on parcourt ce tableau jusqu’à la valeur saisie, et après la valeur saisie, seulement à ce moment on peut lancer la recherche. Elle retourne TRUE si la valeur saisie peut être retenue.

Function checkc(e,ligne,col) de même pour la colonne.

Function checkc(e,ligne,col) de même pour la zone, après avoir identifié la zone correspondante.

3 remplissage sans doublons

On créée une fonction qui prend des lettres dans le tableau contenant toutes les lettres et fait appel à la même fonction qu’à la saisie à la main. Pour simplifier les tests on remplit seulement 20 valeurs. Ainsi, on gagne lorsqu’on a saisi 71 valeurs correctes.

Motus – sous-taches

1 Grille

Function crGrille(‘m’) ajoutera une table de 6 lignes et 8 colonnes à la division grille.

2 Cellule rouge

Function getText() va chercher la mots saisi dont la taille est de 8, dans le cas contraire l’utilisateur est amené à ressaisir le mot.

Dans la fonction même on vérifie le mot caché et le mot saisi sur chaque lettres avec charAt().

Si les lettres coïncident alors la cellule devient rouge grâce aux attributs Style.

3 Cellule jaune

Plus compliqué que prévu car les lettres devenues rouges ne doivent pas être prises en compte.

La solution est d’extraire les lettres trouvées du mot cachées. Ainsi on aura un tableau de lettres non trouvées ou on va faire appel à Array.includes(lettres). Si le nouveau tableau contient la lettre alors la cellule devient jaune.

On gagne lorsque coup joués est inférieur à 6 et le mot lu est le même que le mot choisi au hasard dans le dictionnaire.

Mots Mêlés

Pour mots mêlés on a décidé une grille 12 x 12

Au clique sur une lettre on va l’ajouter dans un tableau virtuel qui représente le mot lu. Chaque lettre choisie devient rouge.

Dès que ce tableau correspond à l’un des mots cachés dans la grille, vous gagnez, celui-ci devenant vert.

Pour ce faire on a défini :

Fonction arrayEqual(l1, l2) qui renvoie vrai lorsque les 2 tableaux sont égaux.













