Rapport Projet Démineur

Bouha Maaye et Martín de las Heras

Janvier 2020

1 Introduction

Le projet est basé sur la création d'une jeux démineur avec une page organisées selon la consigne. L'objectif est de le faire en HTML, CSS et JavaScript, en utilisant ces trois outils pour mettre forme du contenu demandé, le style et le code du jeu.

2 Méthode

Pour la mise en forme du projet on veut commencer à vous expliquer tous les étapes.

2.1 HTML

Pour le HTML on a créé un fichier index.html, et dans ce fichier on a toutes les balises organisées hiérarchiquement.

<head>: Dans la balise <head>on trouve un titre (Démineur) et toutes les informations des auteurs, description... Après on a (<link rel="stylesheet" href="style.css">) qui lie la partie html avec la partie css qu'on expliquera, après on le lien pour charge le fichier JavaScript qu'on l'a noté script.

<body>: Dans le body on trouve titre de section, qui est démineur, de type <h1>, ensuite on a une que on l'a donné une classe qui s'appelle container. À l'intérieur on a une autre div avec la classe inputs, on trouve dans ce div les balises de type <input>qui permet à l'utilisateur de saisir le nombre de lignes et colonnes. Ensuite on a une autre div avec une class container inputs où on a une autre balise de type <input>qui permet de saisir le nombre de mines. Ensuite une autre div avec les mêmes classes où on a un autre <input>avec de type checkbox pour activer ou désactiver le mode Debug. Finalement on a une <button>pour commencer la partie, une div avec la classe partie pour l'espace avec le statut du jeu et une div avec un id grille.

2.2 CSS

Ensuite on a la partie de style avec CSS. Avec le sélecteur global * on a déterminé une police sans sérif. On utilise l'agencement Flexbox pour tout le document. Le titre h1 on l'a donné un background de couleur eeeeee on l'a centré avec justify-content et avec une taille de 80px. Après on a le classe container: on a mis la direction en colonne en utilisant flex-direction puis un arrière-plan gris, après pour bien le placer on a utilisé des padding et de margin avec des pixel différentes. Ensuite pour le classe inputs on a la direction en ligne en utilisant flex-direction pour qu'il soit l'un a côté de l'autre et en donnent un padding 10px. Après on a l'identifiant debug qu'on lui a donner la largeur et la taille. Dans l'identifiant newgame on lui a donner un arrière-plan vert et et couleur blanc, puis on la positionne avec une largeur 200px, une taille 50px et une couleur plus sombre avec le hover. Pour l'identifiant partie on l'a donné une margine automatique puis on l'a centré avec justify-content et avec un arrière-plan bleu et un couleur blanc. Dans l'identifiant grille on a on a mis de type colonne (parce que les éléments du grille sont les lignes) en utilisant flex-direction, ensuite on a la centre en utilisant justify-content puis un padding de 20px. Ensuite dans la classe ligne on l'a mis de type ligne et on l'a centré et dans la classe cellule on l'a donné un background gris et un bordure de taille 1px de couleur gris et solide et on l'a centré. Finalement on a le media query pour quand la largeur est moins de 769px le statut de la partie soit le 100% de la largeur.

2.3 JavaScript

En JavaScript on commence pour déclarer plusieurs variables : grille, jeu_fini, debug_actif que je l'initialise en faux... on déclare aussi le constructeur de Cellule avec deux paramètres (valeur et révélé). Puis on déclare la fonction getRandomInt comme dans l'énoncé. Après on ouvre l'espace pour utiliser jQuery où on trouve la fonction créer_grille qui crée la grille HTML dans notre index avec le nombre de mines et les lignes et colonnes et avec l'algorithme donné dans l'énoncé. Après on a la fonction placer_mines qui permet à placer les mines dans la grille en utilisant getRandomInt. On a suivant la fonction voisinage qui avec des boucles trouve le nombre des mines dans les cellules voisines (voisinage de Moore). Après on a mis les fonctions précédentes pour que quand l'utilisateur clique le bouton newgame la nouvelle partie se démarre, on crée la grille, on place les mines et on utilise le fonction voisinage. Après on a la fonction la_fonction qui avec le nombre de ligne et de colonne retourne l'élément du grille correspondent, avec $(".ligne") sélectionne toutes les lignes du grille, avec .eq(ligne) sélectionne la ligne qu'on a demandé et avec .children() on a une liste avec des cellules de la ligne demandé, finalement avec slice(colonne, colonne+1) on retrouve la cellule de la colonne qu'on a demandé dans la ligne qu'on avait déjà. On utilise cette fonction pour le mode debug, où on doit montrer la valeur de toutes les cellules que ne sont pas déjà révélés. Finalement on a la fonction à faire quand l'utilisateur clique sur une cellule, on révèle la cellule, s'il est une mine, le jeu est fini et l'utilisateur a perdu, et si ce n'est pas une mine, on trouve

si le nombre de mines est égal au nombre de cellules pour révéler et pourtant l'utilisateur aura gagné. Pour révéler les cellules, on obtient le nombre de ligne et de colonne avec l'index qu'il a dans la ligne, et si le valeur de la cellule est 0 (il n'y a pas des voisins qui sont mines, on révèle récursivement ses voisins et on retourne 0, si le valeur es inférieur à 9 on retourne 0 aussi et s'il est 9 c'est une mine donc on retourne 1.

3 Conclusion

Enfin le projet nous a permis d'améliorer notre connaissance de HTML, CSS et JavaScript. Ensuite pour ce projet on a utilisé les notions de HTML pour mettre en forme le contenu avec des balises simples. Avec CSS à l'aide de Flexbox on a mis en forme le style demandé et bien agencé tout dans la page. Finalement avec JavaScript on a fait la partie de code du jeux démineur, enfin on trouve une page bien organisée et un jeu qu'on peut le jouer.