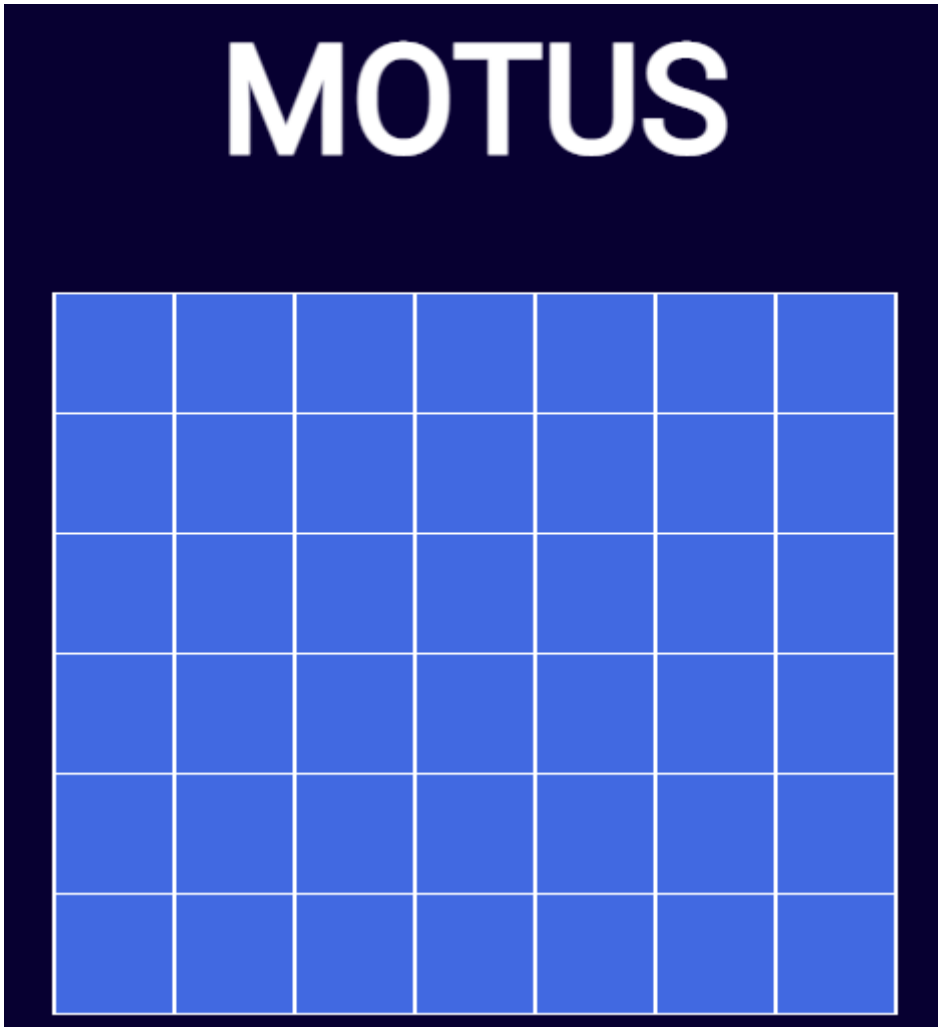
Aleksandar MILENKOVIC

Victor DE DOMENICO

Kickoff Motus JS



HTML :

- Réaliser une page HTML pour la page d'accueil du Motus avec ces éléments :
    - Titre du jeu en haut de la page
    - Liste de liens avec les difficultés (facile, moyen, difficile) qui redirige vers la page du jeu au clic
      Peut-être pourriez-vous envisager de ne pas changer de page ? par exemple le jeu serait masqué tant que l'utilisateur n'aurait pas choisi son niveau de difficulté ? Comme c'est un tout petit jeu une « SPA » me paraît plus adaptée.

- Réaliser une page HTML pour la page de jeu :
    - Titre du jeu en haut de la page
    - Bouton de validation
    - Div vide pour accueillir la grille

CSS :

- Mettre du style pour les deux pages HTML :
    - Bouton facile de couleur jaune
    - Bouton moyen de couleur verte
    - Bouton difficile de couleur rouge
    - Background-color identique pour les deux pages : bleu foncé
    - Les éléments textuels centré sur la page
    - Quelle technique CSS allez-vous utiliser pour la grille ? tableau ? flex ? grid ? autre… ce serait plus confortable de fixer ça avant car il est possible que cela influe sur la partie JS qui va fabriquer la grille. N'hésitez pas à faire un test avec un bout de code dans une sandbox pour valider la pertinence de vos idées, et si vous le faire mettez le code retenu dans le kick-off.

JS :

- Index :
    - Fichier js principal qui regroupera les fonctions dans l'ordre après les avoir importés
      Ici je pense que vous pourriez n'importer que la méthode qui instancie le jeu depuis un module. Exemple :

```
// fichier modules/game/main.js
import Game from './game.js';

export const initGame = (wrapper) => {
        const game = new Game(wrapper);
        game.start();
}

// fichier main.js (fichier js principal)
import { initGame } from 'lib/games.js' ;

document.querySelectorAll('.game-wrapper').forEach(wrapper => {
  initGame(wrapper);
});
```

- Niveau :
  - Fonction qui récupère la classe du lien (facile, moyen ou difficile) et la stocke dans une variable afin d'être exporté et utiliser dans d'autres fonctions
    Pourquoi pas. Dans ce cas je vous suggère que cette fonction prenne en charge tout ceci :
    - Exposer une méthode qui renvoie le niveau de difficulté courant
    - Exposer une méthode qui fabrique l'interface de chois du niveau (elle recevra comme paramètre la div dans laquelle générer les éléments).
    - En interne, la méthode ira chercher les différents niveaux de difficultés dans un autre fichier de configuration (par exemple game/settings.js).

- Mot :
  - Fonction qui permet de sélectionner un mot au hasard parmi un dictionnaire de mots en fonction de la difficulté choisis. Le mot sera choisi en fonction de sa longueur par rapport à la difficulté sélectionné (variable de Niveau.js):
    - Facile : inférieur à 6 lettres
    - Moyen : entre 6 et 9 lettres
    - Difficile : supérieur à 9 lettres
  - On utilisera des conditions if , else if et else pour faire correspondre la longueur du mot avec la difficulté
    N'hésitez pas à choisir des noms et à indiquer dans le kick-off toute la signature. Par exemple : export const getWord = (complexityLevel) => { … }
- Tableau :
  - Fonction qui va générer un tableau de 6 lignes avec des colonnes dont le nombre de case dépendra de longueur du mot importer depuis Niveau.js
  - Chaque case sera un input qui contiendra une lettre
  - On utilisera une boucle for pour générer le tableau
    Avez-vous envisagé d'autres pistes que d'utiliser des inputs ? (ce n'est pas forcément une mauvaise solution mais il y a d'autres possibilités).
- Jeu :
  - Dès qu'un input contient une lettre, l'input suivant est sélectionné jusqu'au dernier de la ligne
  - Lorsqu'on appuie sur la touche entrée : validation du mot

- Lorsqu'on appuie sur la touche retour : suppression du contenu du dernier input rempli de la ligne active
  Comment tout ceci sera-t-il piloté ? via une classe ?

- Lettres
  - L'alphabet apparaitra en dessous du tableau
    Gardez ça pour la fin, vous le ferez si vous avez le temps.

- Validation :
  - Lorsque le joueur valide son mot :
    - On vérifie à l'aide d'une API si le mot existe :
      - Si oui on procède à la validation
      - Sinon un message d'erreur apparait et le mot est supprimé
        Quelle partie de votre code fera le contrôle ? idéalement il faudrait exporter une méthode distincte depuis un autre fichier (par exemple export const checkWord = (word) => {} dans game/api.js)
    - On décompose le mot à trouver lettre par lettre et on met le tout dans une variable array.
    - On récupère la valeur de chaque input afin de les mettre dans une autre variable array
      Là je vois un truc sympa à base de querySelectorAll et de reduce 😊
    - On va faire, pour chaque lettre entrée par le joueur, une vérification avec toute les lettres du mot à trouver :
      - Si les deux lettres correspondent et ont le même indice (même emplacement) : sur la ligne validée, la lettre aura un fond rouge et sur la ligne suivante, la lettre apparaitra au bon endroit dans l'input (qui deviendra non modifiable et sera ignoré par la suite) avec un fond bleu
      - Si les deux lettres correspondent mais n'ont pas le même indice (emplacement différent) : sur la ligne validée, la lettre aura un fond jaune et sur la ligne suivante, la lettre n'apparaitra pas et l'input sera donc vide et modifiable.
      - Si les deux lettres ne correspondent pas : sur la ligne validée, la lettre gardera un fond bleu et sur la ligne suivante, la lettre n'apparaitra pas et l'input sera donc vide et modifiable
        - Vous décrivez bien la logique mais ce qui nous intéresse c'est aussi le code. Comme cet algo est un des éléments complexe de votre appli il faudrait l'écrire dans le kick-off, même si c'est du pseudo code. Vous devez aussi indiquer dans quel partie l'algo se trouvera : un fichier à part, une méthode d'une classe, une méthode dans un fichier exporté...
    - Lors de la validation, la ligne validée devient inactive et on passe à la suivant. Ensuite chaque case va changer en fonction du résultat de la modification, l'une après l'autre (c'est-à-dire : changement de la couleur de fond et de la valeur de l'input). Tout cela accompagné d'un son qui sera différent en fonction du résultat (voir section Autres)
    - Le fond de couleur des lettres changera pour rappeler au joueur quelles lettres de l'alphabet ne sont pas dans le mot
      Comment ferez-vous pour changer les styles ? Via des classes CSS ce serait peut-être

l'idéal ? Bricoler les styles en JS ce n'est pas forcément recommandé : à chacun son expertise, et le métier de CSS, c'est le style.

- Résultat :
    - Si le joueur trouve le mot : Un message affichant la victoire du joueur apparaitra
    - Si échec à la 6ème tentative : Un message affichant la défaite du joueur apparaitra
        - Comment gérez-vous les messages ? au niveau du code et au niveau du HTML ? Est-ce un petit module à part qui fonctionnerait de façon indépendante du reste ?

Autres :

- Dossier js qui comportera les fichiers js et leurs fonctions
- Dossier sons qui comportera les bips
- Pour la sélection du mot :
    - Soit une API
    - Soit un dictionnaire de données au format .txt