**1. Fonction getRandomInt**

typescript

Copier le code

function getRandomInt(max: number): number {

return Math.floor(Math.random() \* max);

}

* **Description** : Cette fonction génère un entier aléatoire compris entre 0 (inclus) et max (exclus).
* **Utilisation** : Elle est utilisée pour sélectionner un mot aléatoire dans la liste des mots disponibles.

**2. Fonction principale JeuDuPendu**

typescript

Copier le code

function JeuDuPendu (): void {

// Déclaration et initialisation des variables

var motsAleatoire : string[] = ["Emma", "Liam","Sofia","Noah","Olivia","Mason","Ava","Elijah","Isabella","James","Mia","Lucas","Charlotte","Oliver","Amelia","Ethan","Harper","Benjamin","Evelyn","Alexander"];

let mots : string = motsAleatoire[getRandomInt(motsAleatoire.length)];

const mot: string = mots.toLowerCase();

const cache: string[] = new Array(mot.length).fill('-');

let essais: number = 6;

* **motsAleatoire** : Un tableau contenant une liste de prénoms.
* **mots** : Sélectionne un mot aléatoire dans motsAleatoire en utilisant getRandomInt.
* **mot** : Le mot sélectionné, converti en minuscules pour simplifier les comparaisons.
* **cache** : Un tableau représentant le mot caché, initialisé avec des tirets (-) pour chaque lettre du mot.
* **essais** : Nombre d'essais restants pour le joueur (initialisé à 6).

**3. Fonctions utilitaires**

**a. Vérifier si une lettre est dans le mot**

typescript

Copier le code

const isInside = (lettre: string): boolean => {

return mot.includes(lettre);

};

* **Description** : Retourne true si la lettre proposée par le joueur est présente dans le mot à deviner, sinon false.

**b. Vérifier si le joueur est pendu**

typescript

Copier le code

const isHanged = (): boolean => {

return essais <= 0;

};

* **Description** : Retourne true si le nombre d'essais est épuisé (c'est-à-dire que essais est inférieur ou égal à 0).

**c. Afficher le mot actuel**

typescript

Copier le code

const displayWord = (): void => {

alert("Le mot est: " + cache.join(''));

};

* **Description** : Affiche le mot actuel avec les lettres trouvées et les tirets pour les lettres restantes.

**d. Remplacer les tirets par les lettres trouvées**

typescript

Copier le code

const replaceChar = (lettre: string): void => {

for (let i = 0; i < mot.length; i++) {

if (mot[i] === lettre) {

cache[i] = lettre;

}

}

};

* **Description** : Parcourt le mot et remplace les tirets par la lettre trouvée à toutes les positions correspondantes.

**e. Vérifier si le joueur a gagné**

typescript

Copier le code

const win = (): boolean => {

return !cache.includes('-');

};

* **Description** : Retourne true si toutes les lettres du mot ont été trouvées (c'est-à-dire qu'il n'y a plus de tirets dans cache).

**4. Fonction guess pour gérer les suppositions**

typescript

Copier le code

const guess = (letter: string): void => {

if (isInside(letter)) {

replaceChar(letter);

alert(`Bonne reponse:'${letter}'!`);

} else {

essais--;

alert(`Desole, '${letter}' n'est pas dans le mot. Vous avez ${essais} essais restants.`);

}

displayWord();

if (win()) {

// Gagné

} else if (isHanged()) {

alert("Vous êtes pendu! Le mot était :" + mot);

}

};

* **Description** :
  + **Si la lettre est dans le mot** : Remplace les tirets par la lettre et affiche un message de réussite.
  + **Sinon** : Décrémente le nombre d'essais et affiche un message d'échec avec le nombre d'essais restants.
  + Affiche le mot actuel après chaque supposition.
  + Vérifie si le joueur a gagné ou perdu après chaque supposition et affiche les messages correspondants.

**5. Boucle principale du jeu**

typescript

Copier le code

while (!win() && !isHanged()) {

const letter: string | null = prompt("Deviner une lettre (ou appuyer sur annuler pour quitter):");

if (letter === null) {

alert("Jeu annulé. Merci d’avoir joué!");

return;

}

if (letter.length === 1) {

guess(letter.toLowerCase());

} else {

alert("Veuillez saisir une seule lettre.");

}

}

if (win()) {

alert("Félicitations! Vous avez deviné le mot:" + mot + " Bien joué!");

} else {

alert("Game over! Plus de chance la prochaine fois! Le mot était : " + mot);

}

console.log(mot);

* **Description** :
  + **Boucle while** : Continue de demander au joueur de deviner une lettre tant que le joueur n'a pas gagné (!win()) et qu'il n'est pas pendu (!isHanged()).
  + **prompt** : Demande au joueur de saisir une lettre ou d'annuler le jeu.
    - **Si le joueur annule** : Affiche un message d'annulation et termine le jeu.
    - **Si le joueur saisit une seule lettre** : Appelle la fonction guess avec la lettre saisie en minuscule.
    - **Sinon** : Demande au joueur de saisir uniquement une lettre.
  + **Après la boucle** :
    - **Si le joueur a gagné** : Affiche un message de félicitations.
    - **Sinon** : Affiche un message de défaite avec le mot correct.
  + **console.log(mot)** : Affiche le mot dans la console pour des fins de débogage.

**6. Lancement du jeu**

typescript

Copier le code

JeuDuPendu();

* **Description** : Appelle la fonction JeuDuPendu pour démarrer le jeu.

**Résumé du fonctionnement**

1. **Initialisation** :
   * Sélectionne un mot aléatoire dans la liste.
   * Initialise le mot caché avec des tirets et définit le nombre d'essais à 6.
2. **Boucle de jeu** :
   * Demande au joueur de saisir une lettre.
   * Vérifie si la lettre est dans le mot.
     + Si oui, remplace les tirets par la lettre trouvée.
     + Si non, décrémente le nombre d'essais.
   * Affiche le mot actuel après chaque supposition.
   * Vérifie si le joueur a gagné ou perdu.
3. **Fin du jeu** :
   * Affiche un message de félicitations si le joueur a gagné.
   * Affiche un message de défaite avec le mot correct si le joueur a perdu.

Ce code offre une implémentation simple et fonctionnelle du jeu du pendu en utilisant des fonctionnalités de base de TypeScript telles que les tableaux, les fonctions et les structures de contrôle.

²