programation web en js

mini-projet 1

Exercice : the life 'n death a po' Monster

L'objectif du td est de programmer un simulateur de la vie d'un pauvre monstre qui passe son temps à dormir, courir, combattre, travailler et se nourrir. L'utilisateur doit surveiller son état et lui faire faire des actions qui lui font gagner ou perdre des points de vie ou de l'argent afin de le maintenir en vie.
Il est demandé de programmer l'application en mettant en œuvre les bonnes pratiques de programmation présentées en cours, en particulier il est demandé d'utiliser des modules ES6.

En particulier, on prévoiera :

- un module définissant le monstre, son état (points de vies, argent ...), et les actions modifiant son état (courir, manger, combattre …),
- un module gérant les affichages sur l'interface : log des actions, affichage de l'état,
- un module important les 2 précédents, et gérant l'exécution de l'application : initialisation, gestion des boutons.
- Le module principal, qui importe le précédant et lance l'exécution de l'application.

Etape 1 :

1. créer le module **`actions`** qui contient l'état et les actions que peut réaliser le monstre :
    1. des variables privées décrivant l'état du monstre :
        - name : le nom du monstre
        - life : nombre de point de vie du monstre
        - money : l'argent du monstre
        - awake : true ou false, indique si le monstre est réveillé ou non,
    2. la fonction exportée "`get`" qui retourne l'état courant du monstre.
    3. la fonction exportée "`init`" qui initialise l'état du monstre avec les valeurs reçues en paramètres.
2. créer le module **`app`** qui contient les fonctions de déroulement de l'application, et en particulier la déclaration des handlers associés aux événements produits par les actions de l'utilisateur sur l'interface. Ce module contient :
    1. des variables privées pour stocker les différents objets du dom recevant des événements,
    2. la fonction exportée "`start`" qui initialise l'application :
        1. en appelant l'action `init ;`
        2. déclare un handler de l'événement click sur le bouton "showme" (#b6) ; ce handler affiche l'état courant du monstre en utilisant une alerte.
3. créer le module principal, qui importe **`app`** et déclare cette fonction "`start`" comme handler de l'événement `window.onload` afin de lancer l'application. Tester.

Etape 2 :

1. créer le module **`ui`**, qui exporte la fonction `log(message)`qui permet d'afficher un message dans la boite `#actionbox` de l'interface. Elle le fait en insérant un nouveau <p> comme premier fils, afin de décaler les les anciens messages vers le bas.
2. dans le même module, déclarer la fonction `displayStatus()` qui permet d'afficher l'état du monstre reçu en paramètre dans la liste `.status` de l'interface.
3. Compléter le module app pour qu'il utilise ces 2 fonctions pour afficher l'état du monstre dans l'interface après son initialisation.

Etape 3 :

1. Dans le module **`actions`**, compléter les actions que peut réaliser le monstre en construisant les fonctions "`run`", "`fight`", "`work`" et "`eat`", puis, dans le module **`app`** associer ces actions aux boutons correspondant, et faites en sorte de mettre à jour l'interface après chaque action :
   - pour chaque méthode, le monstre doit être vivant, réveillé, et disposer de suffisamment de points de vie ou d'argent pour l'action,
   - chaque méthode affiche un message pour expliquer ce qui se passe, ainsi que le nouvel état du monstre,
   - run : perte de 1 point de vie,
   - fight : perte de 3 points de vie,
   - work : perte d'1 pt de vie et gain de 2 unités d'argent,
   - eat : perte de 3 unités d'argent et gain de 2 pts de vie
2. construire la fonctionnalité "`sleep`" qui endort le monstre et programme son réveil 10s plus tard, en utilisant un timer : voir la fonction setTimeOut dans la référence js. Vérifiez que lorsque le montre dort, il ne peut pas courir, manger ni combattre. Le monstre gagne 1 point de vie à son réveil. Bien sur, l'état affiché du monstre doit être affiché lorsqu'il s'endort et lorsqu'il se réveille.

Etape 4 :

1. construire une fonction qui s'exécute toute les 12s (voir pour cela la fonction `setInterval` dans la référence js) et qui retire 1 point de vie au monstre puis exécute une de ses actions au hasard (voir `Math.random()`).
2. programmer et associer les actions correspondant aux boutons kill (pour tuer le monstre) et newlife (pour lui redonner une nouvelle vie).
3. modifier la fonction showStatus pour ajouter un effet visuel lié à l'état du monstre :
   - faire varier la couleur de la boite #monster en fonction du nombre de points de vie (par ex. <5 : rouge, <10 : orange, <15 : bleu, >20 : vert etc …
   - faire varier l'épaisseur de la bordure de la boite #monster en fonction de la quantité d'argent possédée par le monstre.