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 le pattern namespace et le pattern module.

Préliminaires :
Récupérer l'archive "monster" dans arche. Vérifiez que son contenu s'affiche correctement.

Etape 1 :
Dans le fichier javascript :

1. déclarer sous forme de litéral (notation json) un l'objet monster qui va servir de conteneur pour l'ensemble des modules, variables et objet de notre application. Cet objet sert à réaliser le pattern namespace. Pour l'instant, il contient une seule propriété `modules` qui est un objet vide destiné à contenir les différents modules de l'application.
2. créer le module `monster.modules.actions` qui contient 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 "`showme`" qui affiche les propriétés du monstre dans une alerte.
    3. la fonction "`init`" qui initialise l'état du monstre avec les valeurs reçues en paramêtres.
3. créer le module `monster.modules.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 "`run`" qui initialise le monstre en appelant l'action `init` du module `monster.modules.actions` puis déclare la fonction `monster.modules.actions.showme` en tant que handler de l'événement click sur le bouton correspondant.
4. Déclarer cette fonction "`run`" comme handler de l'événnement window.onload afin de lancer l'application.

Etape 2 :

1. dans le module `monster.modules.app` , déclarer 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(life,money,awake )` qui permet d'afficher l'état du monstre reçu en paramètre dans la liste `.status` de l'interface.
3. reprogrammer la fonction `monster.modules.actions.showme` afin qu'elle

n'utilise plus une alerte mais les fonction `log et  displayStatus.`

Etape 3 :

1. Compléter les actions que peut réaliser le monstre en contruisant les fonctions "`run`", "`fight`", "`work`" et "`eat`", puis associer ces actions aux boutons correspondant :
   - 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 fonction "`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.