TP n°4 – programmation événementielle en Javascript

Exercice 1 – Les formulaires

Récupérer le fichier tp4.html

1- Fonctionnement par défaut

a) Remplir les 3 champs du formulaire et valider ; que se passe-t-il ?

b) Ajouter un gestionnaire d'événement lors de la soumission (submit) du formulaire ; ouvrir une popup qui affiche les nom, prénom et mail.

type d'événement	
cible	
action	

2- **Champs obligatoires**

a) Ajouter le code Javascript pour vérifier que les champs nom et prenom soient remplis (champs obligatoires) ; si ce n'est pas le cas, empêcher la validation du formulaire grâce à la méthode preventDefault() de l'objet event et afficher un message d'erreur dans une popup :

b) Modifier le code précédent pour que les messages d'erreur s'affichent dans des span correspondantes (le message « nom obligatoire » doit s'afficher à coté de la zone de saisie du nom, dans une span). Ajouter un style (dans le document html) pour que les messages s'affichent en rouge.

3- **Type des champs**

Vérifier également que le mail ne comporte qu'un seul arobase ; sinon afficher un message d'erreur. *(remarque : utiliser pour cela la fonction indexOf).* Procéder comme à l'étape 2.b pour afficher un message d'erreur.

Exercice 2 – Le jeu du serpent

Récupérer le fichier jeu.html.

Le but du jeu est de manger tous les fruits dans un laps de temps donné. Le serpent se déplace tout seul dans une direction donnée. Le joueur peut changer la direction du déplacement à l'aide des 4 flèches de direction.

1- Déplacement du serpent

a) Ecrire une fonction *avancer()* qui fait avancer le serpent de 10px dans la direction courante (défini par la variable globale *direction*)

Pour que le serpent avance tout seul, il faut que la fonction avancer() soit exécutée automatiquement toutes les x millisecondes. Pour cela, utiliser la fonction setInterval(nomdefonction, délai_en_millisecondes) :

```
var jeu ;  // à déclarer dans les variables globales en début de script.
jeu =setInterval(avancer, 100) ; // en fin de script
```

b) Pour gérer les changements de direction, utiliser les événements liés au clavier (cf tp3) ; l'action consistera seulement à affecter à la variable globale direction le code de la flèche.

type d'événement	
cible	
action	

c) Modifier la fonction `déplacer` pour que le serpent ne sorte pas de l'aire de jeu. Le serpent reste bloqué dès qu'il touche une bordure. (Utiliser les constantes HAUTEUR et LARGEUR).

2- Start / pause

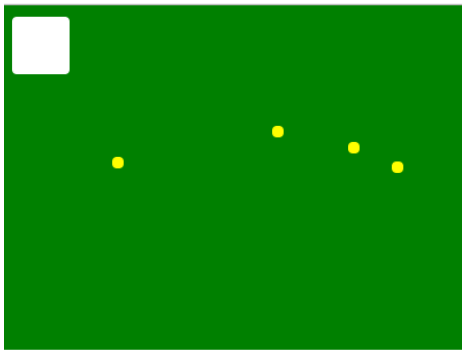
a) Rajouter dans le code HTML deux boutons de type button

b) Au chargement de la page, le serpent doit rester immobile. Le serpent commence à bouger que lorsque l'utilisateur clique sur le bouton start. Ajouter l'évènement.

c) Ajouter le code JS pour faire fonctionner le bouton *pause*. Pour mettre le jeu en pause, il faut empêcher le serpent d'avancer, donc il faut arrêter la fonction *setInterval*. Utiliser pour cela la fonction *clearInterval* avec en paramètre la variable *jeu*.

3- Les fruits

a) Tout d'abord il faut créer les fruits ; pour cela écrire la **fonction *fruits*** ayant comme paramètre le nombre de fruits à créer.



(le style des fruits est défini dans la balise <style> du document HTML)

Créer un fruit revient à créer un nouvel élément (**createElement**)

- de type *div*,
- de **classe CSS *fruit***,
- d'id *f1 (f2, f3, ...)*
- et positionné (*style.top et syle.left*) au hasard.

Une fois créé, il faut ajouter l'élément comme fils de la div *terrain* .(appendChild)

b) Test de collision entre le serpent et un fruit :

Dans la fonction avancer, après avoir déplacé le serpent, il faut tester si la div du serpent est au-dessus d'une div correspondant à un fruit.

Il faut donc passer en revue tous les fruits (nœuds enfants de la div terrain) et pour chaque fruit, vérifier si ses coordonnées offsetTop et offsetLeft sont comprises dans les coordonnées des quatre coins de la div serpent.

Si le serpent mange un fruit, il faut le supprimer du DOM (removeChild) et incrémenter un compteur de fruits mangés *nbFruitsManges*. Si le nombre de fruits mangés est égal au nombre de fruits initial, la partie est gagnée ! Modifier le code pour afficher une popup.

c) Compléter l' HTML avec un champ de type text, affichant en temps réel le nombre de fruits mangés.