Javascript : exercices séance 3

Ex 1

Nous allons porter le jeu Puissance 4 en HTML :

- Créez une page qui contient une table de 7x7 cellules
- Sur les six premières rangées, chaque cellule contient un dont le width et height est de 100%.
- Sur la dernière rangée, chaque cellule contient un <button> dont le width et height est de 100%. De plus le onclick de chaque bouton appelle la fonction joue avec en paramètre le numéro de colonne. Donc onclick="joue(1)" pour le premier bouton, onclick="joue(2)" pour le second bouton etc.
- A la fin de la page, le fichier puissance4.js est chargé. Vous pouvez récupérer votre code de puissance4 en console comme base de travail.
- L'affichage de l'état du jeu passe maintenant par le DOM, écrivez une fonction pour le réaliser :
    - Il faut trouver la table, p.ex. en utilisant https://developer.mozilla.org/en-US/docs/Web/API/document/getElementsByTagName
    - Il faut parcourir cette table pour trouver les cellules. Utilisez l'inspecteur du browser pour bien visualiser la structure de l'arbre. Utilisez p.ex. http://www.w3schools.com/dom/prop_element_childnodes.asp
    - Les div dans chacunes des cellules doivent avoir leurs backgroundColor misent à blanc, bleu ou jaune en fonction de l'état du jeu.
- Les tours de jeu ne peuvent plus être gérés par une boucle utilisant un console.readln() : cette fonction n'existe pas dans un navigateur. Par contre, nous avons placé des boutons qui appelle la fonction joue et donnant en paramètre le numéro de colonne qui doit être joué. Implémentez cette fonction.
- Bonus 1 : améliorez le programme en soignant la présentation avec le css, ajoutez un texte qui indique qui doit jouer, gérer de la fin du jeu avec la possibilité de relancer une partie, etc…
- Bonus 2 : animons la descente d'un pion dans la grille. La manière d'effectuer cela en Javascript est contre-intuitive : on ne peut pas faire une boucle qui passe le pion dans chacune des cases intermédiaires. Le résultat sera que le navigateur reste « gelé » durant toute la période de l'animation et seule la dernière étape est affichée. La raison est qu'un navigateur agit d'une manière mono-threadée : quand la boucle Javascript s'exécute, rien d'autre ne se passe : pas d'interaction avec l'utilisateur, pas de rafraîchissement de l'interface. Il faut donc « rendre » la main au navigateur, en terminant le traitement en cours (en laissant la fonction s'achever). Mais l'animation demande de reprendre la main après un certain temps pour afficher l'étape suivante. Nous utiliserons la fonction setTimeout(f,t) pour effectuer cela : f est une fonction qui sera appelée après t millisecondes. Ainsi, si nous avons besoin d'une boucle, nous pouvons écrire :

```
function boucle(etape,max) {
   if (etape>=max) return ;
   // gérer le traitement de l'étape n° etape
   setTimeout(function() {
         boucle(etape+1,max);
   },100) ;
}
boucle(1,10) ;
```

Ainsi boucle sera appelée jusqu'à ce que l'étape soit >= au max. Chaque traitement s'effectue à la place du commentaire, c'est similaire au corps d'une boucle for. Pour que l'itération suivante se fasse, la fonction setTimeout est appelée. Le premier paramètre est une fonction qui appelle boucle en incrémentant le numéro d'étape. Le second paramètre dit de le faire dans 100ms.

Ex 2

Portez le jeu de bataille navale pour qu'il fonctionne dans un navigateur. A vous de voir comment faire, inspirez-vous de l'exercice précédent…