TP numéro 4 de Langages Web

JavaScript

1. Ouvrir le fichier `exJS.html` dans un navigateur afin de comprendre le fonctionnement du code JavaScript qu'il contient.
2. Créer une page HTML puis écrire un script JavaScript de façon à avoir au chargement de la page une fenêtre de dialogue demandant le nom de l'utilisateur et affichant dans la console le nom saisi.
3. Écrire une fonction JavaScript `table(n)` qui affiche dans la page, à l'aide de l'API DOM, la table de multiplication de son argument (de $n \times 1$ à $n \times 10$). Par exemple, pour $n = 5$, on affichera :

   $5 \times 1 = 5$
   $5 \times 2 = 10$
   $5 \times 3 = 15$
   $5 \times 4 = 20$
   $5 \times 5 = 25$
   $5 \times 6 = 30$
   $5 \times 7 = 35$
   $5 \times 8 = 40$
   $5 \times 9 = 45$
   $5 \times 10 = 50$

   Remarque : l'entité HTML `&times;` permet d'afficher le signe $\times$.

   Ajouter du code qui appelle cette fonction avec un argument choisi par l'intermédiaire d'une fenêtre de dialogue.
4. Écrire en JavaScript le jeu « devine ». Le programme choisit un nombre au hasard entre 0 et 100 que l'utilisateur doit deviner en un nombre minimum de coups. Pour cela, ce dernier propose des nombres et à chaque proposition le programme répond si la valeur à trouver est plus petite ou plus grande. Quand la valeur est trouvée, on affiche dans une fenêtre le nombre de tentatives effectuées.

   Veillez à bien découper votre script en fonctions et à utiliser le bon format de boîte de dialogue en fonction des cas.

   Pour la génération aléatoire, vous pourrez utiliser la fonction :

   ```
   // renvoie un nombre pseudo-aléatoire entre min et max
   function nbAlea(min, max)
   {
       var nb = min + (max-min+1)*Math.random();
       return Math.floor(nb);
   }
   ```

   Attention : les arguments de cette fonction doivent être des entiers. Faites la conversion si nécessaire.

   Vous pourrez ensuite améliorer le jeu :

(a) en laissant l'utilisateur choisir l'intervalle de la valeur ;

(b) en permettant d'effectuer plusieurs parties avec le même intervalle et en ajoutant une gestion du meilleur score obtenu dans la série de parties ;

(c) en permettant au joueur de quitter une partie en cours.

Pour finir, vous utiliserez vos connaissances en HTML et CSS pour améliorer la mise en page et gérer l'affichage, dans un tableau par exemple, des 5 meilleurs scores.

5. Le but de cet exercice est de coder le jeu de la vie (`http://fr.wikipedia.org/wiki/Jeu_de_la_vie`). Pour cela, vous coderez des fonctions JavaScript pour insérer une table dans un document HTML et la faire évoluer selon des règles précises.

   (a) Créer un fichier HTML contenant un bloc `div` d'identifiant `jeuDeLaVie`.

   (b) Créer un fichier JS contenant un constructeur d'objet `JeuDeLaVie` qui prend un paramètre `n` en entrée : il s'agira de la dimension de la table. Les objets créés par ce constructeur possèdent un attribut `jeuVie` qui est un tableau à 2 dimensions (de taille $n \times n$) de booléens. Le constructeur remplit ce tableau par des valeurs aléatoires.

   (c) Ajouter une méthode `creerTabDansId` au constructeur `JeuDeLaVie`, ayant pour paramètre `id`, construisant une table dans l'élément HTML d'identifiant `id`, où la case à la `i`$^{\text{e}}$ ligne et à la `j`$^{\text{e}}$ colonne a pour classe `vivant` si `this.jeuVie[i][j]` est vrai.

   (d) Utiliser le constructeur `JeuDeLaVie` pour insérer une table dans la page HTML.

   (e) Créer une feuille de style pour que la table soit centrée, que chaque case soit un carré de 0,5 cm de côté, et que les éléments de la classe `vivant` soient colorés.

   (f) Ajouter une méthode `voisinsVivants` au constructeur `JeuDeLaVie`, ayant pour paramètres `x` et `y`, retournant le nombre de voisins vivants (c'est-à-dire de cases voisines [1] ayant la valeur `true`) pour la case `this.jeuVie[x][y]`.

   (g) Ajouter une méthode `oneTurn` au constructeur `JeuDeLaVie`, modifiant `this.jeuVie` de la façon suivante : la case `this.jeuVie[x][y]` est vraie à la fin de la méthode si et seulement si deux ou trois de ses cases voisines sont vivantes avant le début de la méthode.

   (h) Ajouter une méthode `unTour` au constructeur `JeuDeLaVie`, faisant avancer d'un tour le jeu de la vie à l'aide de `oneTurn` et mettant à jour la table, puis l'appeler à intervalles réguliers (en dehors du constructeur) en utilisant `setInterval`.

1. Une case qui n'est pas sur le bord de la table a en tout huit cases voisines.