TP3 PARTIE 2 : EXERCICES SUR JAVASCRIPT

Exercice 1 : une calculatrice primitive

Écrivez une page HTML `calculette.html` on retrouvera 2 éléments `<input>` de type texte, une `<div>` sans contenu, et le bouton suivant :

```
<button id="bouton" onclick="calculer()">Calculer</button>
```

Dans un fichier `calcul.js`, vous implémenterez la fonction `calculer()` qui devra :

1. récupérer la valeur de chaque `<input>` ;
2. vérifier que ces valeurs sont bien des entiers ou des nombres flottants ;
3. récupérer l'élément `<div>` ;
4. insérer dans la `<div>` l'addition entre les deux valeurs récupérées.

Exercice 2 : un petit dessin

Écrivez une page HTML `etoiles.html` avec dans le corps seulement une `<div>`, et vous écrirez la balise `<body>` ainsi :

```
<body onload="dessiner_etoiles()">
```

Dans un fichier `dessiner.js`, vous implémenterez la fonction `dessiner_etoiles()`, qui devra récupérer l'élément `<div>` de la page, et y insérer des étoiles sous cette forme :

```
*
**
***
****
*****
******
*******
********
*********
**********
```

> **Note**

> Le caractère pour revenir à la ligne dans une chaîne de caractère est le caractère `\n`.

Exercice 3 : lister les nombres premiers

Écrivez une page HTML `premiers.html` avec un élément `<input>` de type texte, et le bouton suivant :

```
<button onclick="afficher_premiers()">Lister les nombres
premiers</button>
```

Pour savoir si un nombre est premier, il suffit de tester tout ses diviseurs. Si on ne trouve aucun diviseur autre que 1 et le nombre lui-même, alors il est premier.

Question 1

Écrivez en pseudo-code l'algorithme qui prend un nombre et qui retourne vrai si le nombre est premier, faux sinon.

Question 2

Dans un fichier `nombres_premiers.js`, vous implémenterez la fonction `afficher_premiers()` dans laquelle vous allez :

1. implémenter l'algorithme définit à la question précédente ;
2. récupérer le contenu de l'`<input>` et vérifier que son contenu est bien un entier ;
3. pour chaque entier entre 1 et la valeur entrée dans le `<input>`, appliquez l'algorithme de nombre premier ;
4. si un nombre est premier, alors on l'affiche sur la page.

Question 3

Faites en sorte que les nombres premiers soient affichés sous forme de liste.

Exercice 4 : compter les mots

Écrivez une page HTML `compteur.html` avec un élément `<textarea>`, une `<div>` vide, et le bouton suivant :

```
<button onclick="compter_mots()">Compter les mots</button>
```

Dans un fichier `compteur.js`, implémentez la fonction `compter_mots()` qui comptera le nombre de mots dans le texte entré dans le `<textarea>`, et qui affiche le résultat dans la

`<div>`.

Exercice 5 : tableau en table

Dans un fichier `tableau_en_table.js`, implémentez la fonction `tableau_en_table` qui prend un argument un tableau de type `Array`, et qui la transforme en une `<table>` d'une ligne.

Vous créerez votre propre page HTML pour tester votre fonction.

Exercice 6 : initialiser le tableau de manière aléatoire

Pour rappel, pour générer une valeur aléatoire en JavaScript, on écrit :

```
let valeur_alea = Math.ceil(Math.random() * (valeur_max));
```

Écrivez une fonction `init_tableau` qui prend en argument un entier `n`, et qui crée un `Array` avec `n` valeurs générées aléatoirement.

Testez votre nouvelle fonction avec votre travail de l'exercice 5.

Exercice 7 : table en tableau

Dans cet exercice, on va faire le chemin contraire de celui de l'exercice 5, c'est-à-dire qu'on veut récupérer le contenu d'une ligne d'un élément `<table>` et convertir en un `Array`.

Implémentez la fonction `table_en_tableau` qui récupère la table dans la page HTML, et qui renvoie son contenu sous forme de `Array`.

> **Note**
>
> Dans le HTML, vous allez récupérer le contenu sous forme de chaînes de caractère. Pour convertir en entier, vous devez utiliser la fonction `parseInt`.

Exercice 8 : tri par insertion

Voici le pseudo-code du tri par insertion :

```
tri_insertion(tableau tab):
        pour i de 1 à taille(tab)-1:
                tmp = tab[i]
                j = i
```

```
                tant que j > 0 ET tab[j-1] > tmp:
                        tab[j] = tab[j-1]
                        j = j - 1
                        tab[j] = tmp

        retourner tab
```

Question 1

À la main, faites le tri du tableau `[3, 5, 1, 2]` avec l'algorithme de tri par insertion.

Question 2=

Implémentez en JavaScript l'algorithme de tri par insertion dans une fonction `tri_insertion` qui prend en argument un tableau.

Question 3

Testez votre algorithme en affichant le résultat avec `console.log`.

Exercice 9 : affichage d'un tableau trié

Pour finir ce TP, on va réunir tout le travail qu'on a fait depuis l'exercice 5 pour créer une page qui, initialement, affiche un table avec des nombres générés aléatoirement, et un bouton "Trier la table" qui, quand on clique dessus, affiche la table triée.

Pour cela, votre page HTML devra ressembler à quelque chose comme ça :

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8" />
    <title>Tri par insertion</title>
    <script type="text/javascript" src="tri-insertion.js"></script>
  </head>
  <body onload="init_tableau();">
    <div id="tableau-div"></div>
    <button onclick="trier_tableau();">Trier la table</button>
  </body>
</html>
```

À vous de jouer pour écrire la fonction `trier_tableau()` qui récupère la table dans la page, la transforme en `Array`, trie cet array, et finit par transformer l'array en table pour la réintroduire dans le fichier HTML à la place de l'ancienne.