TP N°2 : Révisions sur JavaScript et Manipulation du DOM

Tous les fichiers de travail sont à télécharger depuis Ecampus, ressource : Développement Web/Section Git/Collaborer sur un projet avec Git/Github.

Exercice 1 : Analyse Dynamique des Critiques Clients

Dans le fichier « **index.html** », vous disposez d'une section identifiée par l'ID **`div1`**, qui regroupe une liste de commentaires des clients après l'achat d'un produit donné. Chaque commentaire est noté de 1 (très mauvais) à 5 (excellent) et est suivi d'un texte descriptif. Vous allez implémenter une fonction **`filtrerCommentairesCritiques()`** qui parcourt cette liste pour identifier automatiquement les critiques urgentes nécessitant une attention immédiate. Cette fonction s'arrête dès qu'elle trouve un commentaire indiquant une situation critique, met ce commentaire en évidence et le supprime de la liste des autres commentaires.

1- Ajouter un gestionnaire d'événements dans le fichier « **index.html** » pour déclencher la filtration du premier commentaire critique lors du clic sur un bouton.

2- Implémenter le code de la fonction **`filtrerCommentairesCritiques()`** dans le fichier « **mesScripts.js** » afin de simuler le comportement décrit.

3- Modifier le code de la fonction pour qu'il affiche toutes les critiques sous forme de liste numérotée

Voici un aperçu du résultat final :

Exercice 2 :Animation Dynamique des Destinations de Voyage

Dans le fichier « **index.html** », vous trouverez une bande d'information interactive d'ID **`infoBande`**, qui affiche un message de bienvenue suivi d'une liste de villes. L'objectif de cet exercice est de retirer la première ville de la liste chaque fois que la souris entre ou sort de la zone de la bande, simulant ainsi le passage d'une ville lors d'un voyage. Ce

processus se poursuit jusqu'à ce que toutes les villes aient été passées, après quoi le message final : « **Toutes les villes sont passées ! Au revoir !** » sera affiché.

1- Ajouter les gestionnaires d'événements appropriés dans le fichier « **index.html** » pour détecter l'entrée et la sortie de la souris sur la bande d'information.

2- Implémenter le code de la fonction **`miseAJourVilles()`** dans le fichier « **mesScripts.js** » afin de simuler le comportement décrit. Assurez-vous que la fonction retire la première ville de la liste et met à jour l'affichage chaque fois qu'un événement de souris est détecté.

On veut cette fois-ci retirer automatiquement la première ville de la liste et la rajouter à la fin, créant ainsi une simulation continue de passage des villes comme lors d'un voyage. Ce processus doit continuer indéfiniment sans interaction de la souris.

3-Supprimer tous les gestionnaires d'événements de souris précédemment attachés à la bande d'information. Ajouter un gestionnaire d'événement au chargement de la page qui active le défilement des villes.

4-Implémenter le code de la fonction **`demarrerDefilementVilles()`** qui utilise **`setInterval`** pour mettre à jour la liste des villes toutes 2 secondes. Assurez-vous que la fonction retire la première ville de la liste, l'ajoute à la fin et met à jour l'affichage en conséquence.

Aide :

`setInterval` est une fonction JavaScript qui permet d'exécuter une fonction ou un bloc de code à des intervalles de temps spécifiés, exprimés en millisecondes. Par exemple, **`setInterval(fn, 500)`** exécutera la fonction **`fn`** toutes les 500 millisecondes. C'est particulièrement utile pour des opérations qui doivent être répétées à des intervalles réguliers, comme le rafraîchissement de données ou la création d'animations simples.

Exercice 3 : Chargement dynamique de données

Dans le fichier « **index.html** », vous trouverez une interface interactive permettant d'afficher des données dynamiques dans un tableau. L'objectif de cet exercice est de charger et d'afficher différentes catégories de données telles que les taux de change, les indices boursiers et la consommation d'énergie, en fonction de la sélection de l'utilisateur dans un menu déroulant. Ces données sont initialisées dans des tableaux HTML cachés et doivent être affichées dans un tableau visible lorsque l'utilisateur change sa sélection.

1- Initialiser les éléments de l'interface dans le fichier « **index.html** » : Assurez-vous d'inclure un menu déroulant pour la sélection des catégories de données, ainsi qu'un tableau pour afficher les données sélectionnées.

2- Ajouter un gestionnaire d'événements dans le fichier « **index.html** » pour gérer les changements de la liste déroulante et charger les données correspondantes.

3- Compléter le code de la fonction **`chargerDonnees()`** dans le fichier « **mesScripts.js** ». Cette dernière doit récupérer les données du tableau correspondant à la sélection de l'utilisateur, effacer les données précédemment affichées et remplir le tableau visible avec les nouvelles données. Chaque changement de sélection dans le menu déroulant doit déclencher cette mise à jour.

Aide :

-Pour ajouter une nouvelle ligne à un tableau existant **`tab`**, on utilise la méthode **`insertRow()`** qui renvoie un nouvel objet **`tr`** : **`var nouvelleLigne = tableau.insertRow();`**

-Pour ajouter des cellules (éléments **`td`** ou **`th`**) à cette ligne, on utilise la méthode **`insertCell(indice)`** sur l'objet ligne : **`var cellule1 = nouvelleLigne.insertCell(0);`**

-Pour ajouter un vrai **`th`** :

```
var th = document.createElement('th');
nouvelleLigne.appendChild(th);
```

-La propriété **`.rows`** est utilisée pour accéder à une collection des lignes d'un tableau HTML ou d'une section spécifique du tableau :

```
var rows = table.rows;
console.log(rows.length);
```

-La propriété **`.cells`** est utilisée pour accéder à toutes les cellules d'une ligne de tableau :

```
var row = table.rows[0]; // Accés à la 1re ligne du tableau
console.log(row.cells.length); //affiche le nombre de cellules
```

Exercice 4 : Analyse Dynamique de Données de Transactions

Dans cet exercice, vous allez développer une application web interactive pour simuler et analyser des transactions financières. L'application permettra aux utilisateurs de générer des transactions de manière aléatoire, d'enregistrer ces transactions dans une liste visible et de calculer des statistiques en temps réel, telles que le montant total des transactions, le montant moyen, et le pourcentage de transactions dépassant un seuil spécifique.

1- Dans le fichier « **mesScripts.html** », vous trouverez la fonction **`genererTransaction()`** qui génère un montant aléatoire de transaction entre 100 et 2000 unités. Implémenter la fonction **`ajouterTransaction()`** qui doit :

- •Générer un montant de transaction aléatoire.
- •Ajouter ce montant à la liste des transactions affichée. Si le montant est supérieur à 1000 unités, lui appliquer un style distinctif à l'aide de la classe **`.supMille`** (définie dans le fichier **styles.css**).
- •Mettre à jour les statistiques globales basées sur les nouvelles transactions.

2- Implémenter la fonction **`afficherStatistiques()`** qui affichera, via une alerte, les statistiques suivantes :

- •Total des montants des transactions.
- •Montant moyen des transactions.
- •Pourcentage de transactions supérieures à 1000 unités.

3- Compléter le fichier « **index.html** » en ajoutant les gestionnaires d'événements appropriés pour déclencher l'affichage des transactions et de leurs statistiques lorsque l'utilisateur clique sur un bouton dédié.

Voici un aperçu du résultat à obtenir :