Programmation web : TP
JavaScript - DOM

Exercice 1: Manipulation du DOM avec JavaScript

Cet exercice a pour but de vous familiariser avec les manipulations de base du Document Object Model (DOM) en JavaScript. Vous ajouterez des éléments à une page web, modifierez leur contenu et leur style, et réagirez à des événements utilisateur.

- Créez un fichier HTML (index.html) avec la structure de base
- Créez un fichier JavaScript (script.js) dans le même répertoire que votre fichier HTML.
- Créez une (‘div’) et un paragraphe (‘p’) :
  - Commencez par créer une nouvelle div dans le document HTML.
  - À l'intérieur de cette div, créez un paragraphe p.
  - Ajoutez du texte initial à ce paragraphe, par exemple : « Ceci est un paragraphe ».
- Modification le texte:
  - Après avoir créé le paragraphe, modifiez son texte pour dire : « Le texte a été modifié ».
- Modification du style CSS :
  - Changez le style du paragraphe pour qu'il ait une couleur de fond « lightblue » et que le texte soit centré.
- Ajout d'un événement:
  - Ajoutez un événement « click » à la div. Lorsqu'un utilisateur clique sur la div, le texte du paragraphe doit changer pour « Un clic a été détecté ».

Exercice 2: To-do list

Créez une simple application de liste de tâches (to-do list) en utilisant HTML, CSS et JavaScript. L'application devrait permettre à l'utilisateur d'ajouter des tâches, de les marquer comme accomplies et de les supprimer.

Étapes à suivre :

1. Créer la structure HTML :
   - Créez un fichier HTML contenant une liste <ul> vide pour afficher les tâches.
   - Ajoutez un formulaire avec un champ de texte pour saisir une nouvelle tâche et un bouton pour l'ajouter à la liste.
2. Styler avec CSS :
   - Utilisez CSS pour styliser la liste de tâches et le formulaire. Vous pouvez ajouter des bordures, des marges, des polices, etc., pour améliorer l'apparence de votre application.
3. Manipuler le DOM avec JavaScript :
   - Utilisez JavaScript pour manipuler le DOM et rendre l'application interactive.
   - Lorsque l'utilisateur soumet le formulaire, récupérez la valeur du champ de texte et ajoutez-la à la liste de tâches.

- Chaque tâche devrait avoir un bouton pour la marquer comme accomplie et un bouton pour la supprimer de la liste.
- Lorsque l'utilisateur clique sur le bouton pour marquer une tâche comme accomplie, ajoutez une classe CSS pour indiquer visuellement que la tâche est accomplie.
- Lorsque l'utilisateur clique sur le bouton pour supprimer une tâche, supprimez-la de la liste.

4. Gérer les interactions utilisateur :
   - Utilisez des gestionnaires d'événements pour détecter les clics sur les boutons d'ajout, de marquage comme accomplie et de suppression de tâches.
   - Assurez-vous de mettre à jour le DOM en conséquence après chaque interaction de l'utilisateur.
5. Tester et déboguer :
   - Testez votre application en ajoutant, en marquant comme accomplie et en supprimant des tâches.
   - Utilisez les outils de développement de votre navigateur pour déboguer les éventuels problèmes.

Guide JavaScript

Le DOM est une interface de programmation pour les documents HTML qui représente la page de sorte que les programmes peuvent changer la structure du document, son style et son contenu. Le DOM représente l'arbre hiérarchique des éléments HTML de la page. Lorsqu'une page est chargée, le navigateur crée un arbre DOM. Chaque élément, attribut et texte dans le code HTML devient un nœud dans l'arbre DOM.

1. **Les éléments de la page: window, document**
   - **window:** L'objet global représentant la fenêtre contenant le DOM document.
   - **document:** L'objet principal du DOM, il permet d'accéder à tous les autres nœuds.

   Exemple :
   ```
   console.log(document.title); // Affiche le titre du document
   console.log(document.URL); // Affiche l'URL du document
   ```

2. **Les sélecteurs DOM**
   - **getElementById:** Sélectionne un élément par son ID.
   - **getElementsByClassName:** Sélectionne des éléments par leur classe.
   - **getElementsByTagName:** Sélectionne des éléments par leur nom de balise.
   - **querySelector:** Sélectionne le premier élément qui correspond au sélecteur CSS spécifié.
   - **querySelectorAll:** Sélectionne tous les éléments qui correspondent au sélecteur CSS spécifié.

   Exemple :
   ```
   let elem = document.getElementById("monId");
   let classes = document.getElementsByClassName("maClasse");
   let tags = document.getElementsByTagName("div");
   let query = document.querySelector(".maClasse");
   let queryAll = document.querySelectorAll("div");
   ```

3. **Modification HTML**
   - **innerHTML:** Permet de modifier le contenu HTML d'un élément.
   - **textContent:** Permet de modifier uniquement le texte d'un élément sans les balises HTML.

   Exemple :
   ```
   let elem = document.getElementById("monId");
   elem.innerHTML = '<strong>Nouveau contenu</strong>'; // Change le contenu HTML
   elem.textContent = 'Juste du texte'; // Change le texte, en ignorant les balises HTML
   ```

4. **Sélection des attributs**
   - **getAttribute :** pour obtenir la valeur d'un attribut.
   - **setAttribute :** pour définir une nouvelle valeur pour un attribut.

   Exemple :
   ```
   let value = elem.getAttribute("href"); // Obtient la valeur de l'attribut href
   elem.setAttribute("href", "https://www.example.com"); // Définit la valeur de l'attribut href
   ```

5. **Modification des attributs**
   - **setAttribute:** Ajoute un nouvel attribut ou change la valeur d'un attribut existant.

Exemple :

```
elem.setAttribute("my-attribut', 'valeur'); // Ajoute un nouvel attribut ou
modifie la valeur d'un attribut existant
```

6. **La modification CSS**
   - **style:** Modifie le style d'un élément.

Exemple : changer la couleur de fond :

```
elem.style.color = 'red'; // Change la couleur du texte de l"élément Ã rouge
```

7. **Gestion des classes**
   - **classList.add:** Ajoute une classe à l'élément.
   - **classList.remove:** Supprime une classe de l'élément.
   - **classList.toggle:** Bascule la présence d'une classe.

Exemple :

```
elem.classList.add("nouvelle-classe'); // Ajoute une classe Ã l"élément
elem.classList.remove("ancienne-classe'); // Supprime une classe de l"élément
elem.classList.toggle("active"); // Bascule la classe 'active'
```

8. **Gestion des éléments**
   - **createElement:** Crée un nouvel élément.
   - **appendChild:** Ajoute un nouvel élément à un parent spécifié.
   - **removeChild:** Supprime un enfant d'un parent spécifié.
   - **replaceChild:** Remplace un enfant par un autre dans le parent spécifié.

Exemple :

```
let nouveauDiv = document.createElement("div"); // Crée un nouveau div
document.body.appendChild(nouveauDiv); // Ajoute le nouveau div Ã la fin du body

let ancienDiv = document.getElementById("ancienDiv");
document.body.removeChild(ancienDiv); // Supprime l'ancien div du body

let remplacementDiv = document.createElement("div");
document.body.replaceChild(remplacementDiv, ancienDiv); // Remplace l'ancien div par
le nouveau
```

9. **Gestion des événements**

Le DOM nous permet d'assigner des gestionnaires d'événements spécifiques à des éléments HTML en utilisant la méthode addEventListener. Cela permet de réagir à différents types d'événements, tels que les clics, le survol de la souris, ou encore les frappes de touches, sans avoir à modifier le HTML.

- **addEventListener** est utilisée pour ajouter un gestionnaire de clics à un bouton. Lorsque l'utilisateur clique sur le bouton, une alerte apparaît.

Exemple :

```
// Sélection de l"élément
var bouton = document.getElementById("monBouton");

// Ajout d'un écouteur d"événement pour réagir aux clics sur ce bouton
bouton.addEventListener('click', function () {
```

```
alert('Bouton cliqué!');
});
```

10. Gestion du temps

En JavaScript, la gestion du temps peut être effectuée à l'aide des fonctions **setTimeout** et **setInterval**. Ces fonctions permettent de définir des délais ou des intervalles pour l'exécution de code.

- Exemple : **setTimeout**

```
// Exécute une fonction après un délai spécifié (en millisecondes)
setTimeout(function () {
    console.log('Ce message s'affiche après 2 secondes');
}, 2000);
```

- Exemple : **setInterval**

```
// Exécute une fonction Ã répétition Ã chaque intervalle spécifié
var compteur = setInterval(function () {
    console.log('Ce message s'affiche toutes les 3 secondes');
}, 3000);

// flour arrêter l'intervalle, utilisez clearInterval
setTimeout(function () {
    clearInterval(compteur);
    console.log('Intervalle arrêté après 10 secondes');
}, 10000);
```