📚 Cours : Manipulation du DOM et Interactions avec l'Utilisateur

🎯 Objectif du cours :

Ce cours vous guide dans la manipulation du **DOM (Document Object Model)** à l'aide de JavaScript. Vous apprendrez comment interagir avec les éléments HTML d'une page web, comment modifier leur contenu, leur style, et gérer des événements comme les clics ou les saisies de l'utilisateur. À la fin de ce cours, vous serez capable de rendre vos pages web dynamiques et interactives.

---

1. 🌐 Qu'est-ce que le DOM ?

Le **DOM** est une interface de programmation qui permet aux langages de programmation comme JavaScript de manipuler la structure d'une page web. Il représente le document HTML sous forme d'un arbre, chaque élément HTML étant un **nœud** de cet arbre.

- **Éléments HTML** : `<div>`, `<p>`, `<h1>`, etc.
- **Attributs HTML** : `id`, `class`, `style`, etc.
- **Texte** : Le contenu textuel des éléments.

2. 🛠️ Sélectionner des éléments du DOM

Pour manipuler un élément du DOM, il faut d'abord le **sélectionner**. JavaScript propose plusieurs méthodes pour cela :

a. `getElementById()`

Cette méthode sélectionne un élément par son `id`.

```
let titre = document.getElementById("monTitre");
console.log(titre); // Affiche l'élément ayant l'ID "monTitre"
```

b. `getElementsByClassName()`

Cette méthode sélectionne tous les éléments ayant une certaine classe. Elle renvoie une **collection d'éléments**.

```
let paragraphes = document.getElementsByClassName("paragraphe");
console.log(paragraphes); // Affiche tous les éléments ayant la classe
"paragraphe"
```

c. querySelector()

Cette méthode permet de sélectionner un élément en utilisant un sélecteur CSS. Elle renvoie **le premier élément correspondant**.

```
let bouton = document.querySelector(".monButton");
console.log(bouton); // Sélectionne le premier élément avec la classe "monButton"
```

d. querySelectorAll()

Cette méthode sélectionne tous les éléments correspondant au sélecteur CSS. Elle renvoie une **NodeList**, qui est similaire à un tableau.

```
let boutons = document.querySelectorAll(".btn");
console.log(boutons); // Affiche tous les éléments avec la classe "btn"
```

---

3. 🔄 Modifier le contenu d'un élément

Une fois un élément sélectionné, vous pouvez facilement modifier son **contenu** avec la propriété **`innerHTML`** ou **`textContent`**.

a. Modifier le texte d'un élément

```
let titre = document.getElementById("monTitre");
titre.textContent = "Nouveau Titre"; // Modifie le texte sans HTML
```

b. Modifier le contenu HTML d'un élément

```
let section = document.getElementById("maSection");
section.innerHTML = "<h2>Nouveau contenu HTML</h2><p>Ceci est un nouveau paragraphe.</p>"; // Modifie le contenu HTML
```

---

4. 🎨 Modifier le style d'un élément

JavaScript vous permet de **modifier les styles CSS** d'un élément de manière dynamique. Vous pouvez accéder à la propriété `style` de l'élément sélectionné.

```
let titre = document.getElementById("monTitre");
titre.style.color = "red"; // Change la couleur du texte en rouge
titre.style.fontSize = "30px"; // Modifie la taille de la police
titre.style.backgroundColor = "yellow"; // Change la couleur de fond
```

---

5. ⚙ Modifier les attributs d'un élément

Vous pouvez également modifier les **attributs HTML** d'un élément avec les méthodes **`setAttribute()`** et **`getAttribute()`**.

a. Modifier un attribut

```
let bouton = document.getElementById("monButton");
bouton.setAttribute("disabled", true); // Désactive le bouton
```

b. Obtenir un attribut

```
let bouton = document.getElementById("monButton");
let type = bouton.getAttribute("type"); // Récupère la valeur de l'attribut
"type"
console.log(type);
```

---

6. 🖱 Gestion des événements utilisateur

Les événements sont au cœur de l'interactivité d'une page web. En JavaScript, les événements peuvent être liés à différents éléments HTML pour déclencher des actions. Les événements les plus courants sont les **clics**, les **souris**, les **claviers**, etc.

a. Lier un événement à un élément

Vous pouvez utiliser **`addEventListener()`** pour ajouter un gestionnaire d'événements à un élément.

```
let bouton = document.getElementById("monButton");
bouton.addEventListener("click", function() {
    alert("Vous avez cliqué sur le bouton !");
});
```

b. Les événements les plus utilisés

- **click** : Lorsqu'un utilisateur clique sur un élément.
- **mouseover** : Lorsqu'un utilisateur survole un élément avec la souris.
- **keydown** : Lorsqu'une touche du clavier est enfoncée.
- **submit** : Lorsqu'un formulaire est soumis.

Exemple avec **mouseover** :

```
let bouton = document.getElementById("monButton");
bouton.addEventListener("mouseover", function() {
    bouton.style.backgroundColor = "lightblue"; // Change la couleur de fond
au survol
});
```

---

7. 📝 Manipulation des formulaires

JavaScript permet de **gérer** les **formulaires**, de **valider les entrées de l'utilisateur** et d'empêcher leur envoi si nécessaire.

a. Récupérer la valeur d'un champ de formulaire

```
let inputNom = document.getElementById("nom");
```

```
let nom = inputNom.value; // Récupère la valeur de l'input
console.log(nom);
```

b. Empêcher l'envoi d'un formulaire par défaut

Si vous souhaitez valider le formulaire via JavaScript avant de l'envoyer :

```
let formulaire = document.getElementById("monFormulaire");
formulaire.addEventListener("submit", function(event) {
    event.preventDefault(); // Empêche l'envoi par défaut du formulaire
    let nom = document.getElementById("nom").value;
    if (nom === "") {
        alert("Le nom est requis");
    } else {
        formulaire.submit(); // Soumet le formulaire si validé
    }
});
```

---

8. 📱 Création dynamique d'éléments

En plus de manipuler les éléments existants, JavaScript vous permet de créer de nouveaux éléments HTML à la volée.

a. Créer un élément et l'ajouter à la page

```
let nouvelleDiv = document.createElement("div"); // Crée un nouvel élément
<div>
nouvelleDiv.textContent = "Ceci est une nouvelle div"; // Ajoute du texte
document.body.appendChild(nouvelleDiv); // Ajoute la div au corps de la page
```

---

9. 🔄 Modifier le contenu dynamique en temps réel

Une fonctionnalité clé pour interagir avec l'utilisateur est de **mettre à jour le contenu en temps réel** sans recharger la page. Cela peut se faire en utilisant des événements tels que les **cliques**, **saisies clavier**, etc.

Exemple : afficher la saisie d'un utilisateur en temps réel dans un autre élément.

```
let input = document.getElementById("monInput");
let affichage = document.getElementById("affichage");

input.addEventListener("input", function() {
    affichage.textContent = input.value; // Met à jour le texte en temps réel
});
```

---

10. 🌟 Conclusion

La manipulation du DOM est une compétence essentielle pour le développement web interactif. Dans ce cours, nous avons couvert comment **sélectionner**, **modifier**, et **interagir avec** les éléments du DOM, ainsi que l'importance des **événements** dans l'interactivité. Avec cette base, vous pouvez créer des pages web dynamiques qui réagissent aux actions de l'utilisateur, telles que les clics, les entrées de texte, et les survols.

Prochaines étapes :

- Explorer la manipulation avancée du DOM avec des bibliothèques comme **jQuery**.
- Intégrer des **animations CSS** et JavaScript pour enrichir l'interactivité de vos pages.
- Étudier l'**AJAX** pour charger dynamiquement du contenu sans recharger la page.