📓 Cours : Gestion des Événements et des Formulaires en JavaScript

🎯 Objectif du cours :

Ce cours vous introduit à la gestion des **événements utilisateur** et à la **gestion des formulaires** avec JavaScript. Vous apprendrez comment interagir avec les événements tels que le clic, la soumission de formulaire, ou la saisie clavier, et comment manipuler les formulaires pour valider et envoyer des données. À la fin de ce cours, vous serez capable de rendre vos pages web interactives et de gérer les informations saisies par les utilisateurs de manière dynamique.

---

1. 🖱️ Gestion des Événements en JavaScript

Un **événement** en JavaScript est une action que l'utilisateur effectue, comme un **clic**, une **saisie clavier**, ou un **mouvement de souris**. Les événements peuvent être capturés et manipulés grâce à JavaScript.

a. Écouter un événement

Pour **écouter** un événement, nous utilisons la méthode **`addEventListener()`**. Cette méthode prend deux arguments : l'événement à écouter (par exemple, `"click"`, `"mouseover"`, `"keydown"`, etc.) et une fonction de rappel qui sera exécutée lorsque l'événement se produira.

Exemple : écouter un clic sur un bouton :

```
let bouton = document.getElementById("monBouton");

bouton.addEventListener("click", function() {
    alert("Le bouton a été cliqué !");
});
```

b. Les événements courants en JavaScript

- **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.
- **focus** : Lorsqu'un élément, comme un champ de texte, reçoit le focus.
- **blur** : Lorsqu'un élément perd le focus.

Exemple avec **keydown** :

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

input.addEventListener("keydown", function(event) {
```

```
    console.log("Vous avez appuyé sur la touche : " + event.key);
});
```

---

2. 📝 Gestion des Formulaires en JavaScript

Les formulaires HTML sont utilisés pour collecter des données auprès des utilisateurs. JavaScript permet de **valider** ces données avant de les envoyer au serveur et d'empêcher l'envoi par défaut dans certains cas.

a. Récupérer la valeur d'un champ de formulaire

Pour récupérer la valeur d'un champ de formulaire, nous utilisons la propriété **`value`** de l'élément HTML.

Exemple : récupérer la valeur d'un champ de texte (input) :

```
let nomInput = document.getElementById("nom");
let nom = nomInput.value;
console.log(nom); // Affiche la valeur saisie dans le champ "nom"
```

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

Lorsque l'utilisateur soumet un formulaire, le comportement par défaut est d'envoyer les données au serveur et de recharger la page. Nous pouvons empêcher cela en utilisant la méthode **`preventDefault()`** sur l'événement `submit`.

Exemple de prévention de la soumission du formulaire :

```
let formulaire = document.getElementById("monFormulaire");

formulaire.addEventListener("submit", function(event) {
    event.preventDefault(); // Empêche la soumission par défaut
    console.log("Le formulaire ne sera pas envoyé.");
});
```

c. Valider un formulaire avec JavaScript

Avant d'envoyer un formulaire, il est souvent nécessaire de vérifier que l'utilisateur a rempli correctement les champs (par exemple, vérifier si un champ est vide). Vous pouvez effectuer cette validation avec JavaScript avant de soumettre le formulaire.

Exemple de validation de formulaire pour vérifier qu'un champ est rempli :

```
let formulaire = document.getElementById("monFormulaire");

formulaire.addEventListener("submit", function(event) {
    let nom = document.getElementById("nom").value;
    if (nom === "") {
        alert("Le nom est requis.");
        event.preventDefault(); // Empêche la soumission du formulaire
```

```
    }
});
```

d. Soumettre un formulaire via JavaScript

Si la validation réussit, nous pouvons soumettre le formulaire manuellement avec **`submit()`**. Cela permet de simuler une soumission sans passer par le bouton de soumission classique.

Exemple de soumission manuelle d'un formulaire :

```
let formulaire = document.getElementById("monFormulaire");

formulaire.addEventListener("submit", function(event) {
    event.preventDefault(); // Empêche la soumission par défaut
    // Validation des champs...
    formulaire.submit(); // Soumet le formulaire après la validation
});
```

---

3. 🔄 Manipulation Dynamique des Éléments de Formulaire

JavaScript vous permet de manipuler les **éléments de formulaire** de manière dynamique, comme ajouter des champs supplémentaires, modifier des valeurs, ou même supprimer des éléments du DOM.

a. Ajouter dynamiquement un champ de formulaire

Vous pouvez créer de nouveaux éléments de formulaire, comme des champs de saisie, et les ajouter au DOM en utilisant **`createElement()`** et **`appendChild()`**.

Exemple pour ajouter un nouveau champ de saisie :

```
let formulaire = document.getElementById("monFormulaire");

let nouvelInput = document.createElement("input");
nouvelInput.type = "text";
nouvelInput.name = "nouveauChamp";
nouvelInput.placeholder = "Saisissez un texte ici";

formulaire.appendChild(nouvelInput);
```

---

4. 🖱 Manipulation des Événements sur les Formulaires

Les événements liés aux formulaires vous permettent de capturer des actions spécifiques des utilisateurs, telles que la soumission, la mise au point (focus) ou la perte de focus (blur) d'un champ de formulaire.

a. Événement `focus` et `blur`

Les événements **`focus`** et **`blur`** sont utilisés pour détecter quand un champ de formulaire reçoit ou perd le focus.

Exemple d'utilisation de **`focus`** et **`blur`** :

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

inputNom.addEventListener("focus", function() {
    console.log("Le champ 'Nom' a le focus.");
});

inputNom.addEventListener("blur", function() {
    console.log("Le champ 'Nom' a perdu le focus.");
});
```

b. Événement `input`

L'événement **`input`** se déclenche chaque fois qu'un utilisateur modifie le contenu d'un champ de formulaire, qu'il soit en train de taper du texte ou de sélectionner une option.

Exemple de gestion de l'événement **`input`** :

```
let inputEmail = document.getElementById("email");

inputEmail.addEventListener("input", function() {
    console.log("L'email a été modifié : " + inputEmail.value);
});
```

---

5. 📤 Envoyer des Données de Formulaire avec Fetch

Après avoir validé et manipulé les données du formulaire, vous pouvez envoyer ces données à un serveur sans recharger la page en utilisant **`fetch()`** pour faire une requête HTTP.

Exemple d'envoi de données de formulaire via **`fetch()`** :

```
let formulaire = document.getElementById("monFormulaire");

formulaire.addEventListener("submit", function(event) {
    event.preventDefault(); // Empêche la soumission du formulaire par défaut

    let formData = new FormData(formulaire);

    fetch("votreScriptServeur.php", {
        method: "POST",
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        console.log("Réponse du serveur :", data);
```

```
    })
    .catch(error => {
        console.error("Erreur lors de l'envoi :", error);
    });
});
```

---

6. 🌟 Conclusion

La gestion des événements et des formulaires est essentielle pour créer des applications web interactives. Grâce à JavaScript, vous pouvez interagir avec les utilisateurs, valider leurs entrées, et gérer les données de manière dynamique sans avoir besoin de recharger la page. Vous êtes maintenant capable de manipuler les événements du DOM, gérer les formulaires, et effectuer des actions en fonction des interactions des utilisateurs.

Prochaines étapes :

- Explorer la gestion des formulaires avec des bibliothèques comme **jQuery**.
- Apprendre à utiliser **AJAX** pour des interactions avec le serveur sans rechargement de page.