Manipulation des formulaires

Les formulaires permettent aux utilisateurs d'interagir avec une page web en envoyant des informations. Manipuler les formulaires avec JavaScript permet de contrôler, valider, et réagir aux données saisies avant l'envoi.

Manipuler les Champs de Texte

Les champs de texte **(<input type="text">)** sont utilisés pour recevoir des données textuelles de l'utilisateur. JavaScript offre des méthodes pour lire et définir leur contenu, ainsi que des événements pour détecter les changements effectués par l'utilisateur.

```
<form>
  <label for="name">Nom :</label>
  <input type="text" id="name" name="name">
</form>
```

Pour interagir avec le contenu d'un champ de texte en JavaScript, on utilise la **propriété value.** Celle-ci permet à la fois de lire la valeur actuelle saisie par l'utilisateur et de la modifier si nécessaire.

```
// Sélectionner le champ de texte
let input = document.getElementById('name');

// Lire la valeur du champ de texte
let nom = input.value;
console.log("Nom saisi :", nom);

// Définir une nouvelle valeur pour le champ
input.value = "Jean Dupont";
```

Dans cet exemple, **input.value** lit la valeur actuelle du champ, tandis que l'affectation **input.value = "Jean Dupont";** modifie cette valeur pour afficher "Jean Dupont" dans le champ.

Utilisation des Événements input et change

Pour réagir aux actions de l'utilisateur, JavaScript offre deux événements utiles :

- **input** : Cet événement se déclenche chaque fois que l'utilisateur modifie le contenu du champ, a chaque frappe au clavier.

```
input.addEventListener('input', function() {
   console.log(input.value);
});
```

- **change** : Cet événement se déclenche uniquement lorsque l'utilisateur quitte le champ de texte après avoir effectué une modification.

```
input.addEventListener('change', function() {
   console.log(input.value);
});
```

Dans cet exemple, l'événement **input** capture chaque modification en temps réel, tandis que **change** se déclenche lorsque l'utilisateur sort du champ.

Manipuler les Champs select

Les menus déroulants **(<select>)** permettent à l'utilisateur de choisir une option parmi plusieurs possibilités. JavaScript permet d'accéder à la liste des options, de détecter quelle option est sélectionnée et même d'en ajouter de nouvelles.

```
<label for="country">Pays :</label>
<select id="country" name="country">
  <option value="fr">France</option>
  <option value="us">États-Unis</option>
</select>
```

Accéder aux Options et à l'Option Sélectionnée

Dans un champ select, il est possible d'accéder à toutes les options disponibles avec la propriété **options**, et de déterminer laquelle est sélectionnée en utilisant **selectedIndex** qui est une propriété de l'element select.

Liste des options : La propriété options permet d'accéder à la liste complète des options d'un select.

```
let select = document.getElementById('country');

let options = select.options;
console.log("Options disponibles :", options);
```

Option sélectionnée : **select.selectedIndex** renvoie l'index de l'option choisie, tandis que **select.options[select.selectedIndex]** permet d'accéder directement à cette option.

```
// Obtenir l'option sélectionnée
let selectedOption = options[select.selectedIndex];
console.log("Option actuellement sélectionnée :", selectedOption.text);
```

Dans cet exemple, **select.selectedIndex** donne l'index de l'option sélectionnée, et **options[select.selectedIndex]** permet de lire son texte.

Ajouter une Nouvelle Option à un select

Pour ajouter une option, il est possible de créer un nouvel élément option, puis de l'ajouter à la liste avec la méthode **select.add()**.

```
let newOption = document.createElement("option");
newOption.value = "de";
newOption.text = "Allemagne";
select.add(newOption);
```

Cet exemple ajoute l'option "Allemagne" avec une valeur "de" à la liste de sélection.

Utiliser l'Événement change sur un select

L'événement **change** sur un select est déclenché lorsque l'utilisateur choisit une nouvelle option dans la liste.

```
select.addEventListener('change', function() {
   console.log("Nouvelle option sélectionnée :", select.value);
});
```

Ce code permet de détecter et de traiter les changements dans le choix de l'utilisateur.

Manipuler les Cases à Cocher

Les cases à cocher **(<input type="checkbox">)** offrent un moyen simple pour l'utilisateur de faire un choix binaire, souvent pour des options d'activation ou de désactivation. JavaScript permet de lire, modifier et surveiller leur état.

```
<label for="subscribe">S'abonner :</label>
<input type="checkbox" id="subscribe" name="subscribe">
```

Lire et Définir l'État d'une Case à Cocher

Pour savoir si une case est cochée, on utilise la propriété **checked**. Cette même propriété permet de modifier l'état de la case en lui attribuant une valeur true ou false.

```
let checkbox = document.getElementById('subscribe');

// Lire l'état de la case à cocher
let isChecked = checkbox.checked;
console.log("Case cochée :", isChecked);

// Définir la case comme cochée
checkbox.checked = true;
```

Ici, **checkbox.checked** retourne **true** si la case est cochée, et l'attribution **checkbox.checked = true;** permet de la cocher.

Activer et Désactiver une Case à Cocher

Il est aussi possible de désactiver temporairement une case pour empêcher l'utilisateur de la modifier en utilisant la propriété disabled.

```
// Désactiver la case
checkbox.disabled = true;

// Activer la case
checkbox.disabled = false;
```

Utiliser l'Événement change sur une Case à Cocher

L'événement change permet de détecter quand l'utilisateur coche ou décoche une case.

```
checkbox.addEventListener('change', function() {
   console.log("État de la case modifié :", checkbox.checked);
});
```

Exercices

Exercice 1 : Manipuler un Champ de Texte avec JavaScript

- **Création du formulaire de base** : Créez un formulaire HTML contenant un champ de type texte pour le nom, avec un libellé "Nom", et ajoutez un bouton de soumission.
- **Lecture de la saisie utilisateur** : Récupérez le champ de texte et configurez-le pour lire la valeur saisie par l'utilisateur à chaque modification **(événement input)**. Affichez la valeur actuelle du champ dans la console à chaque frappe.
- **Confirmation de modification** : Utilisez l'événement **change** pour afficher un message de confirmation dans la console lorsque l'utilisateur quitte le champ de texte.

Exercice 2 : Gérer un Menu Déroulant (select) avec des Interactions Dynamiques

- **Ajout du menu déroulant** : Dans le même formulaire, ajoutez un menu déroulant **(<select>)** pour que l'utilisateur puisse choisir son pays. Créez des options pour "France", "États-Unis" et "Canada".
- **Affichage des options** : Accédez au champ select en et affichez toutes les options disponibles dans la console au chargement de la page.

- **Écoute des changements** : Ajoutez un événement **change** pour afficher l'option choisie par l'utilisateur chaque fois qu'il modifie sa sélection. Dans cet événement, affichez dans la console le texte de l'option sélectionnée ainsi que sa valeur.

- **Ajout dynamique d'options** : En dehors du formulaire, ajoutez deux nouveaux champs de type texte et un bouton pour permettre à l'utilisateur d'ajouter dynamiquement une nouvelle option au menu déroulant select. L'utilisateur doit pouvoir saisir le nom du pays ainsi que la valeur de l'option.

```
<option value="fr">France</option>
```

- **Événement d'ajout** : Ajoutez un événement au nouveau bouton permettant de créer dynamiquement un nouvel élément option et de l'ajouter au champ select. Utilisez l'exemple de code fourni dans le cours.

Exercice 3 : Manipuler et Gérer les Interactions avec une Case à Cocher

- **Création de la case à cocher** : Dans le formulaire, ajoutez une case à cocher avec le label "S'abonner à la newsletter".

- **Suivi de l'état** : Récupérez la case à cocher et affichez son état (cochée ou non cochée) dans la console chaque fois que l'utilisateur modifie la case.

- **Paramétrage initial** : Configurez la case pour qu'elle soit cochée par défaut lorsque la page est chargée.

- **Message dynamique selon l'état** : Utilisez l'événement change pour afficher un message spécifique sous la case selon son état :

    - "Vous êtes abonné(e) !" si la case est cochée.
    - "Vous n'êtes pas abonné(e)." si elle est décochée.

Exercice 4 : Validation du Formulaire avec JavaScript

- **Création des éléments d'erreur** : Sous chaque champ du formulaire, créez des éléments HTML pour afficher les messages d'erreur associés.

- **Bouton de validation** : Ajoutez un bouton "Valider" à la fin du formulaire pour déclencher une vérification des données avant l'envoi.

- **Validation des champs** :
    - Vérifiez que le champ "Nom" n'est pas vide. Si le champ est vide, affichez un message d'erreur : "Veuillez saisir un nom."

    - Vérifiez que l'utilisateur a sélectionné un pays dans le champ select. Si aucune option n'est sélectionnée, affichez un message d'erreur : "Veuillez choisir un pays."

    - Vérifiez que l'utilisateur a coché la case d'abonnement. Si elle est décochée, affichez un message d'erreur : "Veuillez cocher la case pour confirmer l'abonnement."

- **Affichage d'un message de succès** : Si tous les champs sont correctement remplis, affichez un message de succès sous le formulaire : "Formulaire envoyé avec succès !"

Exercice 5 : Personnalisation et Interactions Visuelles du Formulaire

- **Indication de saisie en cours** : Lorsque l'utilisateur commence à saisir son nom dans le champ, changez la couleur du texte en bleu pour indiquer que la saisie est en cours. Lorsqu'il quitte le champ, remettez la couleur d'origine.

- **Changement de couleur du champ select** : Configurez le menu déroulant (select) pour que la couleur de fond du champ change selon le choix de l'utilisateur :
    - France : bleu
    - États-Unis : rouge
    - Canada : vert

- **Message dynamique sous la case à cocher** : Créez un paragraphe sous la case à cocher pour afficher un message en fonction de son état :
    - Lorsque la case est cochée, affichez : "Merci de vous être abonné(e) !"
    - Lorsque la case est décochée, affichez : "Abonnez-vous pour recevoir nos actualités."

- **Validation instantanée du champ "Nom"** : Configurez une validation instantanée pour le champ "Nom" afin qu'une erreur apparaisse en rouge dès qu'une saisie incorrecte est détectée, et disparaisse dès qu'elle est corrigée. Assurez-vous que le champ contient au moins 2 caractères pour être considéré valide.