Apprendre le JavaScript | Contrôler les formulaires

Les formulaires sont un élément essentiel du web. Ils permettent aux utilisateurs d'envoyer des données au serveur, de se connecter à un site, de remplir des informations, etc. En JavaScript, nous pouvons contrôler les formulaires pour valider les données, les pré -remplir, les soumettre, etc.

Rappel

Les éléments de formulaires sont des éléments HTML qui permettent de saisir des données. Les éléments de formulaires les plus courants sont les champs de texte, les cases à cocher, les boutons radio, les listes déroulantes, les boutons, etc.

Les attributs essentiels des formulaires sont :

- `action` : l'URL du script qui traite les données du formulaire.
- `method` : la méthode HTTP utilisée pour envoyer les données (GET ou POST).

Les éléments de formulaire ont également des attributs spécifiques qui permettent de contrôler la valeur de l'élément. Ce sont ces données qui sont envoyées au serveur lors de la soumission du formulaire.

Un élément qui n'a pas de name, ne sera pas envoyé lors de la soumission du formulaire.

- `value` : la valeur de l'élément.
- `name` : le nom de l'élément.

Types d'éléments de formulaires

Les éléments de formulaires peuvent être de différents types. Les types de champs sont:

- `text` : un champ de texte.
- `password` : un champ de texte pour les mots de passe. // Les caractères sont masqués.
- `checkbox` : une case à cocher. // Plusieurs cases peuvent être cochées
- `radio` : un bouton radio. // Un seul bouton peut être sélectionné. Les boutons radio doivent avoir le même `name` pour former un groupe.
- `search` : un champ de recherche. // Permet de saisir une recherche et d'activer un bouton de recherche.
- `date` : un champ de date.
- `time` : un champ d'heure.
- `number` : un champ numérique. Prends l'attribut `step` pour définir le pas de l'incrément.
- `range` : un champ de plage. Prends l'attribut `min` et `max` pour définir les valeurs minimales et maximales.

- `file` : un champ de fichier.
- `color` : un champ de couleur.
- `tel` : un champ de téléphone.
- `email` : un champ d'email.
- `url` : un champ d'URL.
- `hidden` : un champ caché.
- `reset` : un bouton de réinitialisation.
- `button` : un bouton. // ne soumet pas le formulaire
- `submit` : un bouton de soumission. // soumet le formulaire

Les select et les textarea sont des éléments de formulaires qui permettent de sélectionner une option dans une liste déroulante ou de saisir du texte dans une zone de texte.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#input_types

Accessibilité des éléments de formulaires

Les éléments de formulaires doivent être accessibles pour les utilisateurs qui utilisent des technologies d'assistance. Les attributs utilisés pour améliorer l'accessibilité des éléments de formulaires sont :

- `id` : l'identifiant de l'élément. À placer sur l'élément `input, textarea, select,` etc.
- `for` : l'identifiant de l'élément auquel l'élément label est associé. À placer sur l'élément `label`.

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

Les attributs spécifiques aux éléments de formulaires

Les éléments HTML de type input, textarea et select possèdent des attributs spécifiques qui permettent de contrôler la valeur de l'élément.

- `value` : la valeur de l'élément.
- `name` : le nom de l'élément.
- `type` : le type de l'élément.
- `id` : l'identifiant de l'élément.
- `placeholder` : le texte d'aide de l'élément.
- `checked` : l'état de la case à cocher ou du bouton radio.
- `selected` : l'état de l'option sélectionnée.
- `disabled` : désactive l'élément. Pour accéder ou modifier un attribut, on utilise la méthode par point ou par crochet de l'objet `element`.

```
const champ = document.querySelector("input[type='text']");
console.log(champ.value); // Affiche la valeur du champ champ.value = "Nouvelle
valeur"; // Modifie la valeur du champ
champ.value = "Nouvelle valeur"; // Modifie la valeur du champ
```

Les attributs spécifiques à certains types d'éléments

Checkbox et Radio

Pour les éléments de type `checkbox` et `radio`, l'attribut `checked` permet de savoir si l'élément est coché ou non. De plus, on peut le modifier pour cocher ou décocher l'élément.

```
const checkbox = document.querySelector("input[type='checkbox']");
console.log(checkbox.checked); // Affiche true si la case est cochée

checkbox.checked = true; // Coche la case
checkbox.checked = false; // Décoche la case
```

Select

Pour les éléments de type `option`, l'attribut `selected` permet de savoir si l'option est sélectionnée ou non. De plus, on peut le modifier pour sélectionner ou désélectionner l'option.

```
const option = document.querySelector("option");
console.log(option.selected); // Affiche true si l'option est sélectionnée

option.selected = true; // Sélectionne l'option
option.selected = false; // Désélectionne l'option
```

L'objet select possède également une propriété `selectedIndex` qui permet de connaître l'index de l'option sélectionnée et une propriété `selectedOptions` qui permet de connaître les options sélectionnées dans un select multiple.

```
const select = document.querySelector("select");
console.log(select.selectedIndex); // Affiche l'index de l'option sélectionnée
console.log(select.selectedOptions); // Affiche les options sélectionnées
```

File

Pour les éléments de type `file`, l'attribut `files` permet de connaître les fichiers sélectionnés par l'utilisateur.

Le input file n'utilise pas la propriété `value` pour afficher le nom du fichier sélectionné. Pour afficher le nom du fichier, on utilise la propriété `files` de l'élément.

```
const file = document.querySelector("input[type='file']");
console.log(file.files); // Affiche les fichiers sélectionnés
```

Désactiver un champ

Pour désactiver un champ, on utilise l'attribut `disabled`. Cela empêche l'utilisateur d'interagir avec le champ. Lorsqu'un champ est désactivé, il ne peut pas être modifié, ni soumis avec le formulaire.

```
const champ = document.querySelector("input[type='text']");
champ.disabled = true; // Désactive le champ
champ.disabled = false; // Active le champ
```

Récupérer les données d'un formulaire

Pour récupérer les données d'un formulaire, on utilise la propriété `elements` de l'objet `form`. Cette propriété est une collection d'éléments de formulaire. On peut accéder à un élément de la collection par son nom ou son index.

```
const form = document.querySelector("form");
const champ = form.elements["nom"];
console.log(champ.value); // Affiche la valeur du champ
```

On peut également accéder à un élément de formulaire par son nom en utilisant la propriété `name` de l'objet `form`.

```
<form>
    <input type="text" name="nom" value="John Doe" />
</form>
```

```
const form = document.querySelector("form");
const champ = form.nom;
console.log(champ.value); // Affiche la valeur du champ
```

Les événements liés aux formulaires

Bloquer l'envoi d'un formulaire

Par défaut, lorsqu'un formulaire est soumis, la page est rechargée. De plus, si on clique sur un bouton de type `submit`, le formulaire est soumis ou si on appuie sur la touche `Enter` dans un champ de texte, le formulaire est soumis. Pour empêcher ce comportement par défaut, on peut utiliser la méthode `preventDefault()` de l'objet `event`.

```
form.addEventListener("submit", function (evenement) {
    evenement.preventDefault();
});
```

Envoyer un formulaire par JavaScript

Pour envoyer un formulaire par JavaScript, on utilise la méthode `submit()` de l'objet `form`.

```
form.submit();
```

Réinitialiser un formulaire par JavaScript

Pour réinitialiser un formulaire par JavaScript, on utilise la méthode `reset()` de l'objet `form`.

```
form.reset();
```

Changement de valeur d'un champ

Pour détecter le changement de valeur d'un champ, on utilise l'événement `change()`. Cet événement est déclenché lorsque la valeur du champ change et que le champ perd le focus.

```
champ.addEventListener("change", function () {
    console.log("La valeur du champ a changé");
});
```