



LES FORMULAIRES EN JAVASCRIPT



MEMBRES DU GROUPE

KOUADIO AMANI YANNICK IVAN
KOUADIO DIBIE ELISEE JULES CEDRIC
KONE GNINFLAN CHARLY
KOUAME GUY MARC AXEL
SAOURE KOUAME FIDELE
SIKA BEDEL

Professeur

M. SANE ARNAUD

SOMMAIRE

I- Introduction

A-Rappels général sur les formulaires

- 1-Utilité des formulaires
- 2-Outils de création
- 3-Fichier complémentaire du traitement

B-Structure et fonctionnement des formulaires

II- Validation des données

A-Attributs HTML

 **Required**

 **Maxlength & Minlength**

B-Objet et méthode JavaScript

III- L'accès au formulaire et les objets liés au formulaire

A-Accès au formulaire

- 1-Accès par la collection forms
- 2-Accès par les méthodes de DOM

B-Objet et propriétés liés au formulaire

- 1-Objet text
 - a)Propriétés de l'objet text
 - b)Méthodes de l'objet text
- 2-Objet checkbox
 - a)Propriétés de l'objet checkbox
- 3-Objet radio
 - b)Propriétés de l'objet radio
- 4-Objet select
 - a)Propriétés de l'objet radio

IV- Conclusion

I. INTRODUCTION

Imaginez un monde où les sites web seraient de simples pages statiques, incapables d'interagir avec les utilisateurs. Heureusement, grâce à JavaScript et aux formulaires, ce monde est devenu un lointain souvenir. Les formulaires en JavaScript sont les piliers de l'interaction dynamique entre les utilisateurs et les sites web. Ils permettent de collecter des données de types différents, de gérer des transactions et de créer des expériences utilisateur personnalisées.

Dans cet exposé, plongeons dans l'univers captivant des formulaires en JavaScript, où chaque saisie devient une action concrète. Préparez-vous à explorer les mécanismes sous-jacents, à découvrir des fonctionnalités puissantes et à percer les mystères des interactions en ligne. Soyez prêts car les formulaires en JavaScript vous réservent bien des surprises !

A. Rappel général sur les formulaires

1. Utilité des formulaires

Les formulaires constituent un moyen incontournable de collecter des informations auprès des utilisateurs sur les sites web. Ils permettent de recueillir des données telles que les noms, les adresses, les préférences, les commentaires, et bien plus encore. Les formulaires jouent un rôle crucial dans diverses situations, que ce soit pour des inscriptions, des sondages, des commandes en ligne ou des formulaires de contact. Ils facilitent l'échange d'informations entre les utilisateurs et les propriétaires de sites web.

2. Outils de création

En HTML, les formulaires sont créés à l'aide de l'élément `<form>`. Cet élément doit être configuré avec les attributs `method` et `action`. L'attribut `method` spécifie la méthode d'envoi des données (`GET` ou `POST`), tandis que l'attribut `action` indique la page chargée de traiter les données. Pour créer les champs du formulaire, l'élément principal utilisé est `<input>`, avec différents types d'attributs. Par exemple, le type `text` est utilisé pour les champs de texte, le type `email` pour les adresses e-mail, et le type `tel` pour les numéros de téléphone. Chaque champ `<input>` doit également avoir les attributs `name` pour identifier le champ et `id` pour faciliter la liaison avec les balises `<label>`.

Les balises `<label>` permettent de lier un libellé à un champ de formulaire. Cela améliore l'accessibilité en fournissant des indications aux utilisateurs sur les données attendues. L'utilisation de l'attribut `for` dans la balise `<label>` et de l'attribut `id` dans la balise `<input>` permet d'établir cette relation...

En ce qui concerne JavaScript, il joue un rôle crucial dans l'ajout d'interactivité aux formulaires. Grâce à JavaScript, nous pouvons valider les saisies utilisateur en temps réel,

afficher des messages d'erreur, proposer des suggestions automatiques et même dynamiquement modifier le contenu des formulaires en fonction des actions de l'utilisateur.

3. Fichier complémentaire du traitement

Outre la création des formulaires, nous devons également considérer le traitement des données soumises. Lorsqu'un utilisateur remplit un formulaire et le soumet, les données doivent être gérées de manière sécurisée et efficace. C'est là que les fichiers complémentaires entrent en jeu. En utilisant des technologies côté serveur comme PHP, Node.js ou Python, nous pouvons traiter les données soumises, les valider, les stocker dans une base de données, les envoyer par e-mail, générer des rapports, ou effectuer toute autre action requise en fonction des besoins spécifiques de l'application.

B. Structure et fonctionnement du formulaire

Un formulaire en HTML est généralement défini à l'aide de la balise `<form>`. À l'intérieur de cette balise, vous pouvez inclure différents éléments de formulaire tels que `<label>`, `<input>`, `<select>`, `<textarea>`, etc. Ces éléments permettent à l'utilisateur de saisir et de sélectionner des données.

Voici un exemple simple de structure d'un formulaire en HTML :

```
1  <!DOCTYPE html>
2  <html Lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href="style.css">
7      <title>Simple formulaire</title>
8  </head>
9  <body>
10     <h2>Formulaire de contact</h2>
11     <form action="traitement.php" method="post">
12         <label for="nom">Nom :</label>
13         <input type="text" id="nom" name="nom" required>
14
15         <label for="prenom">Prénom :</label>
16         <input type="text" id="prenom" name="prenom" required>
17
18         <label for="email">Adresse email :</label>
19         <input type="email" id="email" name="email" required>
20
21         <label for="telephone">Numéro de téléphone :</label>
22         <input type="tel" id="telephone" name="telephone" required>
23
24         <input type="submit" value="Envoyer">
25         <input type="reset" value="Annuler">
26     </form>
27 </body>
28 </html>
29
```

Voici quelques aspects importants du fonctionnement d'un formulaire en JavaScript :

- ✚ **Validation des données** : Vous pouvez utiliser JavaScript pour valider les données saisies par l'utilisateur avant de soumettre le formulaire. Par exemple, vous pouvez vérifier si les champs obligatoires sont remplis, si le format de l'adresse e-mail est correct, ou si les données correspondent à des critères spécifiques.
- ✚ **Gestion des événements** : JavaScript permet de gérer les événements associés au formulaire. Par exemple, vous pouvez écouter l'événement "**submit**" qui se déclenche lorsque l'utilisateur clique sur le bouton "**Envoyer**".
- ✚ **Envoi et traitement des données** : Lorsque l'utilisateur soumet le formulaire, vous pouvez utiliser JavaScript pour intercepter cet événement et gérer l'envoi des données côté serveur. Cela peut impliquer l'utilisation de techniques telles que les requêtes AJAX pour envoyer les données en arrière-plan, ou la navigation vers une autre page pour effectuer le traitement côté serveur.

II. VALIDATION DES DONNEES

A. Attributs HTML

✚ Required

L'attribut "**required**" est utilisé pour indiquer qu'un champ de formulaire est obligatoire. Lorsqu'il est ajouté à un élément HTML tel que **<input>** ou **<select>**, l'utilisateur doit remplir ce champ avant de pouvoir soumettre le formulaire. Cela permet de garantir que les informations essentielles sont fournies.



✚ Maxlength & Minlength

Les attributs "**maxlength**" et "**minlength**" définissent respectivement le nombre maximal et minimal de caractères autorisés dans un champ de texte. Ils sont souvent utilisés pour imposer des limites spécifiques, telles que les mots de passe qui doivent comporter un

nombre minimum de caractères ou les commentaires qui doivent rester concis. Ces attributs permettent de guider les utilisateurs lors de la saisie et de prévenir les entrées incorrectes.

```
1 <label for="pwd">Mot de passe :</label>
2 <input type="password" name="pwd" id="pwd" maxLength="16" minLength="8">
3
```

B. Objet et méthode JavaScript

Objet JavaScript : les expressions régulières

Les expressions régulières, également appelées regex, sont des motifs de texte utilisés pour effectuer des correspondances et des validations avancées. Elles permettent de définir des critères précis pour les données saisies par les utilisateurs.

Par exemple, une expression régulière peut être utilisée pour vérifier si une adresse e-mail est au format correct, si un numéro de téléphone respecte une structure spécifique, ou si un code postal est valide. Les expressions régulières offrent une flexibilité et une puissance de validation inégalées.

```
1 // Numéro de téléphone à valider
2 const phoneNumber = "07 07 80 26 64";
3
4 // Expression régulière pour le format du numéro de téléphone
5 const phoneRegex = /^(\d{2}\s){4}\d{2}$/;
6
7 // Vérification de la validité du numéro de téléphone
8 if (phoneRegex.test(phoneNumber)) {
9   console.log("Le numéro de téléphone est valide.");
10 } else {
11   console.log("Le numéro de téléphone n'est pas valide.");
12 }
```

Dans cet exemple, nous utilisons l'expression régulière `/^(\\d{2}\\s){4}\\d{2}$/`. Voici ce qu'elle signifie :

- ⚡ `^` : Début de la chaîne.
- ⚡ `(\\d{2}\\s){4}` : Représente le motif répété quatre fois. `\\d{2}` correspond à deux chiffres consécutifs et `\\s` correspond à un espace.
- ⚡ `\\d{2}` : Représente les deux derniers chiffres du numéro de téléphone.
- ⚡ `$` : Fin de la chaîne.

Tableau des caractères spéciaux couramment utilisés dans les expressions régulières

Caractère Spécial	Description
\	Échappe le caractère suivant pour être utilisé littéralement
.	Correspond à n'importe quel caractère sauf le saut de ligne
^	Correspond au début de la chaîne
\$	Correspond à la fin de la chaîne
*	Correspond à 0 ou plusieurs occurrences du caractère précédent
+	Correspond à 1 ou plusieurs occurrences du caractère précédent
?	Correspond à 0 ou 1 occurrence du caractère précédent
{n}	Correspond à exactement n occurrences du caractère précédent
{n,}	Correspond à au moins n occurrences du caractère précédent
{n,m}	Correspond à entre n et m occurrences du caractère précédent
[...]	Correspond à un caractère parmi ceux spécifiés entre les crochets
[^...]	Correspond à un caractère qui n'est pas parmi ceux spécifiés
(...)	Groupe les caractères ensemble pour créer des sous-expressions
	Correspond à l'une des alternatives séparées par le symbole
\w	Correspond à un caractère alphanumérique (lettre, chiffre, _)
\W	Correspond à un caractère qui n'est pas alphanumérique
\d	Correspond à un chiffre
\D	Correspond à un caractère qui n'est pas un chiffre
\s	Correspond à un espace, une tabulation ou un saut de ligne
\S	Correspond à un caractère qui n'est pas un espace
\b	Correspond à une limite de mot (début ou fin de mot)
\B	Correspond à une position qui n'est pas une limite de mot

Méthode JavaScript : preventDefault()

Lorsqu'un formulaire est soumis, le comportement par défaut du navigateur est de recharger la page ou de naviguer vers une nouvelle URL.

Cependant, dans de nombreux cas, nous souhaitons contrôler ce comportement par défaut afin d'effectuer des actions personnalisées avant ou à la place du rechargement de la page.

La méthode `preventDefault()` permet d'annuler le comportement par défaut d'un événement, tel que la soumission d'un formulaire.

Cela permet de traiter les données du formulaire en utilisant JavaScript, d'effectuer des opérations supplémentaires et de fournir une expérience utilisateur plus fluide.



```
1 <form id="nameForm">
2   <label for="name">Nom :</label>
3   <input type="text" id="name" name="name">
4   <button type="submit">Valider</button>
5 </form>
6
7 <script>
8   document.getElementById('nameForm').addEventListener('submit', function(event) {
9     event.preventDefault(); // Empêche le comportement par défaut du formulaire
10    var nameInput = document.getElementById('name');
11    var name = nameInput.value.trim();
12    // Vérifie si le nom est valide
13    if (name === '') {
14      alert('Veuillez saisir un nom.');// Affiche un message d'erreur
15    } else {
16      alert('Nom valide : ' + name); // Affiche le nom saisi
17    }
18  });
19 </script>
```

III. L'ACCES AU FORMULAIRE ET LES OBJETS LIES AU FORMULAIRE

A. Accès au formulaire

1. Accès par la collection forms

La collection `forms` est une propriété de l'objet `document` en JavaScript qui représente une collection d'éléments de formulaire (`<form>`) présents dans le document HTML.

Cette collection regroupe tous les formulaires présents sur la page et permet d'accéder à ces formulaires individuellement pour les manipuler ou obtenir des informations à leur sujet.

La collection `forms` est une collection indexée, ce qui signifie que chaque formulaire est accessible à travers un index numérique.

L'index commence à 0 pour le premier formulaire, puis 1 pour le deuxième formulaire, et ainsi de suite.

On peut aussi les manipuler grâce notamment à `id` et au `name`.

Accès à un formulaire par son index :

```
1 var form = document.forms[0]; // Accède au premier formulaire de la page
```

Accès à un formulaire par son attribut id :

```
1 var form = document.forms["myForm"]; // Accède au formulaire avec l'ID "myForm"
```

Accès à un formulaire par son attribut name :

```
1 var form = document.forms.namedItem("myForm"); // Accède au formulaire avec le nom "myForm"
```

2. Accès par les méthodes de DOM

Les formulaires sont étroitement liés au Document Object Model (DOM) en JavaScript. Lorsqu'un formulaire est présent dans une page HTML et que l'utilisateur interagit avec celui-ci, les données saisies sont reflétées et accessibles via le DOM. Chaque élément de formulaire, tel que les champs de saisie, les boutons, les cases à cocher, etc., est représenté par un nœud du DOM. Ces nœuds peuvent être accédés et manipulés à l'aide des méthodes et propriétés JavaScript.

Voici quelques exemples de manipulations courantes des formulaires en utilisant le DOM :

Accès aux éléments de formulaire :

Nous pouvons utiliser des méthodes DOM telles que `getElementById`, `getElementsByTagName`, `getElementsByClassName`, ou `querySelector` pour accéder aux éléments de formulaire spécifiques en utilisant leur ID, leur nom, leur balise ou leur classe.



Exemple :

```
1 var myInput = document.getElementById("myInput");
2 var myForm = document.querySelector("form");
3 var allInputs = myForm.getElementsByTagName("input");
4
```




B. Objet et propriétés liés aux éléments de formulaire en HTML

1. L'objet input de type text

a) Propriété de l'objet text

-  **name** : Indique le nom du champ de texte.
-  **Value** : Contient la valeur actuelle du champ de texte.




b) Méthodes de l'objet text

-  **focus()** : Définit le focus sur le champ de texte, ce qui signifie qu'il sera prêt à recevoir une entrée de l'utilisateur.
-  **blur()** : Retire le focus du champ de texte.
-  **select()** : Sélectionne tout le texte dans le champ de texte.

```
1 <input type="text" id="myInput" name="myInputName" value="Valeur initiale">
2
3 <script>
4   var inputElement = document.getElementById("myInput");
5
6   // Accéder à la valeur de l'élément de saisie de texte
7   var value = inputElement.value;
8   console.log(value);
9
10  // Définir le focus sur le champ de texte
11  inputElement.focus();
12
13  // Retirer le focus du champ de texte
14  inputElement.blur();
15
16  // Sélectionner tout le texte dans le champ de texte
17  inputElement.select();
18 </script>
```

2. L'objet input de type checkbox




a) Propriété de l'objet checkbox

-  **name** : Indique le nom de l'élément checkbox.
-  **Checked** : Indique l'état actuel de l'élément case à cocher (coché = True ou non = False).
-  **Value** : Contient la valeur actuelle de l'élément checkbox.

```
1 <input type="checkbox" id="myCheckbox" name="myCheckboxName" value="valeur1" checked>
2
3 <script>
4   var checkboxElement = document.getElementById("myCheckbox");
5
6   // Accéder à L'état de La case à cocher
7   var isChecked = checkboxElement.checked;
8   console.log(isChecked);
9 </script>
10
```

3. L'objet input de type radio

a) Propriété de l'objet radio

-  **name** : Indique le nom de l'élément radio.
-  **Checked** : indique l'état actuel d'un des boutons sélectionnés (coché = True ou non = False).
-  **Value** : Contient la valeur actuelle de l'élément radio.

```
1 <input type="radio" id="radio1" name="radioGroup" value="option1" checked>
2 <input type="radio" id="radio2" name="radioGroup" value="option2">
3
4 <script>
5   var radio1Element = document.getElementById("radio1");
6   var radio2Element = document.getElementById("radio2");
7
8   // Accéder à L'état des boutons radio
9   var isRadio1Checked = radio1Element.checked;
10  var isRadio2Checked = radio2Element.checked;
11  console.log(isRadio1Checked);
12  console.log(isRadio2Checked);
13 </script>
14
```


4. L'objet input de type select

a) Propriété de l'objet select

 **Value** : Contient la valeur actuelle de l'élément de la liste déroulante.



```
1 <select id="mySelect" name="mySelectName">
2   <option value="option1">Option 1</option>
3   <option value="option2">Option 2</option>
4   <option value="option3">Option 3</option>
5 </select>
6
7 <script>
8   var selectElement = document.getElementById("mySelect");
9
10  // Accéder à la valeur sélectionnée dans le menu déroulant
11  var selectedValue = selectElement.value;
12  console.log(selectedValue);
13 </script>
14
```

 **selectedIndex**: Cette propriété permet d'accéder à l'index de l'option sélectionnée dans le menu déroulant.



```
1 <select id="mySelect" name="mySelectName">
2   <option value="option1">Option 1</option>
3   <option value="option2">Option 2</option>
4   <option value="option3">Option 3</option>
5 </select>
6
7 <script>
8   var selectElement = document.getElementById("mySelect");
9
10  // Accéder à l'index de l'option sélectionnée
11  var selectedIndex = selectElement.selectedIndex;
12  console.log(selectedIndex);
13 </script>
14
```

- ✚ **options**: Cette propriété renvoie une collection d'objets "option" qui représentent les options disponibles dans le menu déroulant.



```
1 <select id="mySelect" name="mySelectName">
2   <option value="option1">Option 1</option>
3   <option value="option2">Option 2</option>
4   <option value="option3">Option 3</option>
5 </select>
6
7 <script>
8   var selectElement = document.getElementById("mySelect");
9   // Accéder à la collection des options
10  var options = selectElement.options;
11  // Accéder à la valeur et au texte de la première option
12  var firstOptionValue = options[0].value;
13  var firstOptionText = options[0].text;
14  console.log(firstOptionValue);
15  console.log(firstOptionText);
16 </script>
17
```

- ✚ **length**: Cette propriété renvoie le nombre d'options présentes dans le menu déroulant.



```
1 <select id="mySelect" name="mySelectName">
2   <option value="option1">Option 1</option>
3   <option value="option2">Option 2</option>
4   <option value="option3">Option 3</option>
5 </select>
6
7 <script>
8   var selectElement = document.getElementById("mySelect");
9
10  // Accéder au nombre d'options dans le menu déroulant
11  var optionsCount = selectElement.options.length;
12  console.log(optionsCount);
13 </script>
14
```

- ✚ **add()** et **remove()**: Ces méthodes permettent d'ajouter et de supprimer dynamiquement des options dans le menu déroulant.

```
1  <select id="mySelect" name="mySelectName">
2    <option value="option1">Option 1</option>
3    <option value="option2">Option 2</option>
4    <option value="option3">Option 3</option>
5  </select>
6
7  <script>
8    var selectElement = document.getElementById("mySelect");
9
10   // Ajouter une nouvelle option
11   var newOption = document.createElement("option");
12   newOption.value = "option4";
13   newOption.text = "Option 4";
14   selectElement.add(newOption);
15
16   // Supprimer une option existante
17   selectElement.remove(1); // Supprime la deuxième option
18 </script>
19
```

IV- Conclusion

En conclusion, dans ce document nous avons abordé divers aspects des formulaires HTML, en fournissant des informations essentielles sur leur utilité, leur structure et leur fonctionnement. Nous avons exploré les outils de création disponibles et examiné les méthodes de validation des données à l'aide des attributs HTML et des fonctionnalités JavaScript. Les formulaires HTML jouent un rôle crucial dans la collecte d'informations et l'interaction avec les utilisateurs sur les sites web. En comprenant leur structure, leur validation et leur manipulation à l'aide de JavaScript, les développeurs peuvent créer des expériences utilisateur fluides et efficaces.

On retiendra que, les formulaires HTML sont un élément clé du développement web, et une compréhension solide de leur utilisation et de leurs fonctionnalités permet aux développeurs de créer des interfaces interactives et conviviales pour les utilisateurs.