Architecture et langages Web
(NFA040)

Formulaires (tutoriel)

Remarque. Ce tutoriel est extrait de la section de MDN intitulée : Mon premier formulaire HTML. Les sources des exemples accompagnent ce sujet.

Un formulaire HTML, qu'est-ce ?

Les formulaires HTML sont un des vecteurs principaux d'interaction entre un utilisateur et un site web ou une application. Ils permettent à l'utilisateur d'envoyer des données au site web. La plupart du temps, ces données sont envoyées à des serveurs web mais la page peut aussi les intercepter et les utiliser elle-même.

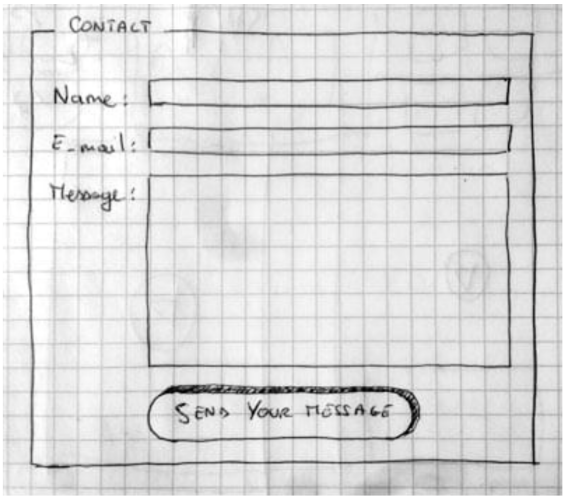
Un formulaire HTML est composé d'un ou plusieurs widgets. Ceux-ci peuvent être des zones de texte (sur une seule ligne ou plusieurs lignes), des boîtes à sélection, des boutons, des cases à cocher ou des boutons radio. La plupart du temps, ces items sont associés à un libellé qui décrit leur rôle — des étiquettes correctement implémentées sont susceptibles d'informer clairement l'utilisateur normal ou mal-voyant sur ce qu'il convient d'entrer dans le formulaire.

La principale différence entre un formulaire HTML et un document HTML habituel réside dans le fait que, généralement, les données collectées par le formulaire sont envoyées vers un serveur web. Dans ce cas, vous avez besoin de mettre en place un serveur web pour récupérer ces données et les traiter. La mise en place d'un tel serveur ne fait pas partie des sujets abordés dans ce guide. Si vous souhaitez toutefois en savoir plus, voyez « Envoi des données de formulaire » plus loin dans ce module.

Concevoir le formulaire

Avant de passer au code, il est souhaitable de prendre un peu de recul et accorder quelques instants de réflexion à votre formulaire. Dessiner un rapide croquis vous permettra de définir les informations que vous souhaitez demander à l'utilisateur. Du point de vue de l'expérience utilisateur, il est important de garder à l'esprit que plus vous demandez d'informations, plus vous risquez que votre utilisateur s'en aille. Restez simple et ne perdez pas votre objectif de vue : ne demandez que ce dont vous avez absolument besoin. La conception de formulaires est une phase importante de la construction d'un site internet ou d'une application.

Dans ce guide, nous allons concevoir un formulaire de contact simple. Posons les premières pierres.



Notre formulaire contiendra trois champs de texte et un bouton. Nous demandons simplement à notre utilisateur son nom, son adresse électronique et le message qu'il souhaite envoyer. En appuyant sur le bouton, le message sera envoyé au serveur web.

Apprentissage actif : mise en œuvre de notre formulaire HTML

Très bien, nous sommes maintenant prêts à passer au HTML et à coder notre formulaire. Pour construire notre formulaire, nous aurons besoin des éléments HTML suivants : `<form>`, `<label>`, `<input>`, `<textarea>` et `<button>`.

Avant de poursuivre, faites une copie locale de notre simple modèle HTML — vous y incorporerez votre formulaire.

L'élément `<form>`

Tous les formulaires HTML débutent par un élément `<form>` comme celui-ci :

```
<form action="/my-handling-form-page" method="post">

</form>
```

Cet élément définit un formulaire. C'est un élément conteneur au même titre que les éléments `<div>` ou `<p>`, mais il accepte aussi quelques attributs spécifiques afin de contrôler la manière dont il se comporte. Tous ses attributs sont optionnels mais définir au moins les attributs `action` et `method` est considéré comme de bonne pratique.

- L'attribut `action` définit l'emplacement (une URL) où doivent être envoyées les données collectées par le formulaire.
- L'attribut `method` définit la méthode HTTP utilisée pour envoyer les données (cela peut être « get » ou « post »).

Pour le moment, ajoutez l'élément `<form>` ci dessus dans le corps de votre HTML.

Les éléments `<label>`, `<input>` et `<textarea>`

Notre formulaire de contact est très simple et ne contient que trois champs de texte, chacun ayant une étiquette. Le champ d'entrée pour le nom est un champ de texte sur une seule ligne, le champ pour l'adresse électronique est un champ de texte sur une ligne qui n'accepte que des adresses électroniques et enfin le champ pour le message est un champ de texte sur plusieurs lignes.

En terme de code HTML, nous avons besoin de quelque chose qui ressemble à ceci pour mettre en œuvre nos widgets de formulaire.

```
<form action="/ma-page-de-traitement" method="post">
    <div>
        <label for="name">Nom :</label>
        <input type="text" id="name" name="user_name">
    </div>
    <div>
        <label for="mail">e-mail :</label>
        <input type="email" id="mail" name="user_mail">
    </div>
    <div>
        <label for="msg">Message :</label>
        <textarea id="msg" name="user_message"></textarea>
    </div>
</form>
```

Les éléments `<div>` sont ici pour structurer notre code et rendre la mise en page plus facile (voir ci-dessous). Veuillez noter l'utilisation de l'attribut `for` sur tous les éléments `<label>`. C'est une manière formelle de lier un libellé à un élément du formulaire. Cet attribut fait référence à l'`id` de l'élément correspondant. Il y a plusieurs avantages à faire ainsi. Le plus évident de permettre à l'utilisateur de cliquer sur l'étiquette pour activer le bloc correspondant. Si vous souhaitez mieux comprendre les bénéfices de cet attribut, tout est détaillé dans cet article : Comment structurer un formulaire HTML.

Concernant l'élément `<input>`, l'attribut le plus important est l'attribut `type`. Ce dernier est extrêmement important puisqu'il définit le comportement de l'élément `<input>`. Il peut radicalement changer le sens de l'élément, faites-y attention. Si vous voulez en savoir plus à ce propos, vous pouvez lire l'article au sujet des widgets natifs pour formulaire.

- Dans notre exemple nous n'utilisons que la valeur `text` — qui est la valeur par défaut de cet attribut et représente un champ de texte basique sur une seule ligne acceptant n'importe quel type de texte.

- Pour la deuxième entrée, nous utilisons la valeur `email` qui définit un champ de texte sur une seule ligne n'acceptant que des adresses électroniques valides. Cette dernière valeur transforme un champ basique en une sorte de champ « intelligent » qui réalise des vérifications sur les données fournies par l'utilisateur. Vous trouverez plus de détails sur la validation des formulaires dans l'article Validation des données de formulaire.

Last but not least, remarquez la syntaxe de `<input>` vs `<textarea></textarea>`. C'est une des bizarreries du HTML. La balise `<input>` est un élément vide, ce qui signifie qu'il n'a pas besoin de balise fermante. Au contraire, `<textarea>` n'est pas un élément vide, il faut donc le fermer avec la balise fermante appropriée. Cela a un effet sur une caractéristique spécifique des formulaires HTML : la manière dont vous définissez la valeur par défaut. Pour définir une valeur par défaut d'un élément `<input>` vous devez utiliser l'attribut `value` de la manière suivante :

```
<input type="text" value="par défaut cet élément sera renseigné avec ce
texte">
```

A contrario, si vous souhaitez définir la valeur par défaut d'un élément `<textarea>`, il suffit simplement de mettre la valeur par défaut entre les balises ouvrantes et fermantes de l'élément `<textarea>` de la manière suivante :

```
<textarea>par défaut cet élément sera renseigné avec ce texte</textarea>
```

L'élément `<button>`

Notre formulaire est presque terminé. Il nous suffit seulement d'ajouter un bouton pour permettre à l'utilisateur de nous envoyer les données renseignées dans le formulaire. Ceci se fait simplement en ajoutant d'un élément `<button>` ; ajoutez-le juste avant la balise fermante `</form>` :

```
<div class="button">
    <button type="submit">Envoyer le message</button>
</div>
```

Note : Vous pouvez aussi utiliser l'élément `<input>` avec le type approprié pour produire un bouton, par exemple `<input type="submit">`. Le principal avantage de `<button>` par rapport à l'élément `<input>` est que ce dernier ne permet d'utiliser que du texte comme étiquette tandis que l'élément `<button>` permet d'utiliser n'importe quel contenu HTML, autorisant ainsi des textes de bouton plus complexes et créatifs.

Mise en page élémentaire du formulaire

Nous avons désormais notre formulaire HTML, et si vous le regardez dans votre navigateur préféré, vous verrez qu'il est plutôt laid :

Note : Si vous pensez que vous n'avez pas écrit un code HTML correct, faites la comparaison avec celui de notre exemple terminé — voyez first-form.html (ou également directement).

Les formulaires sont notoirement embêtants à présenter joliment. Apprendre la mise en page ou la décoration des formulaires sort du cadre de cet article, donc pour le moment nous allons simplement ajouter quelques indications au CSS pour lui donner un air convenable.

Tout d'abord, ajoutons un élément `<style>` à notre page, dans l'en-tête HTML. Comme ceci :

```
   <style>

   </style>
```

Entre les balises style, ajoutons le CSS suivant, juste comme indiqué :

```
form {
  /* Uniquement centrer le formulaire sur la page */
  margin: 0 auto;
  width: 400px;
  /* Encadré pour voir les limites du formulaire */
  padding: 1em;
  border: 1px solid #CCC;
  border-radius: 1em;
}

form div + div {
  margin-top: 1em;
}

label {
  /* Pour être sûrs que toutes les étiquettes ont même taille et sont
correctement alignées */
  display: inline-block;
  width: 90px;
  text-align: right;
}

input, textarea {
  /* Pour s'assurer que tous les champs texte ont la même police.
     Par défaut, les textarea ont une police monospace */
  font: 1em sans-serif;

  /* Pour que tous les champs texte aient la même dimension */
  width: 300px;
  box-sizing: border-box;
```

```
  /* Pour harmoniser le look & feel des bordures des champs texte */
  border: 1px solid #999;
}

input:focus, textarea:focus {
  /* Pour souligner légèrement les éléments actifs */
  border-color: #000;
}

textarea {
  /* Pour aligner les champs texte multi-ligne avec leur étiquette */
  vertical-align: top;

  /* Pour donner assez de place pour écrire du texte */
  height: 5em;
}

.button {
  /* Pour placer le bouton à la même position que les champs texte */
  padding-left: 90px; /* même taille que les étiquettes */
}

button {
  /* Cette marge supplémentaire représente grosso modo le même espace que celui
     entre les étiquettes et les champs texte */
  margin-left: .5em;
}
```

Désormais notre formulaire a une bien meilleure allure.

Note : Il est sur GitHub dans first-form-styled.html (à voir aussi directement).

Envoyer les données au serveur Web

Finalement, gérer les données du formulaire côté serveur web est peut être le plus compliqué. Comme dit auparavant, un formulaire HTML est une manière pratique de demander de l'information à un utilisateur et de les adresser à un serveur web.

L'élément `<form>` définit où et comment les données sont envoyées, merci aux attributs `action` et `method`.

Mais ce n'est pas tout. Nous avons aussi besoin de donner un nom à nos données. Ces noms sont importants pour deux raisons. Du côté du navigateur, cela sert à définir le nom de chaque élément de donnée. Du côté du serveur, chaque information doit avoir un nom pour être manipulée correctement.

Pour nommer vos données vous devez utiliser l'attribut `name` pour identifier bien précisément l'élément d'information collecté par chacun des widgets. Regardons à nouveau le code de notre formulaire :

```
<form action="/my-handling-form-page" method="post">
  <div>
    <label for="name">Nom :</label>
    <input type="text" id="name" name="user_name">
  <div>
  <div>
    <label for="mail">E-mail :</label>
    <input type="email" id="mail" name="user_email">
  </div>
  <div>
    <label for="msg">Message :</label>
    <textarea id="msg" name="user_message"></textarea>
  </div>
</form>
  ...
```

Dans notre exemple, le formulaire enverra trois informations nommées respectivement « `user_name` », « `user_email` » et « `user_message` ». Ces informations seront envoyées à l'URL « `/my-handling-form-page` » avec la méthode HTTP POST.

Du côté du serveur, le script à l'URL « `/my-handling-form-page` » recevra les données sous forme d'une liste de trois éléments clé/valeur intégrés à la requête HTTP. À vous de définir comment ce script va manipuler les données. Chacun des langages serveurs (PHP, Python, Ruby, Java, C#, etc.) a son propre mécanisme pour traiter ces données. Il n'appartient pas à ce guide d'approfondir ce sujet, mais si vous souhaitez en savoir plus, nous avons mis quelques exemples dans l'article Envoi des données de formulaire.