

# Construire des formulaires

Les formulaires sont une partie essentielle d'Internet, car ils permettent aux sites Web de récolter les informations des utilisateurs afin de traiter les demandes. Les formulaires peuvent demander une petite quantité d'informations, faire une requête de recherche ou saisir un nom d'utilisateur et un mot de passe, ou une grande quantité d'informations comme des informations d'expédition et de facturation ou remplir une demande d'emploi.

Nous avons besoin de savoir comment construire des formulaires afin d'interagir avec les utilisateurs. Dans cette leçon, nous allons voir comment utiliser le langage HTML pour créer un formulaire, quels éléments utiliser pour récolter différents types de données et comment styler les formulaires en CSS. Nous n'entrerons pas trop dans de détail pour ce qui est de la façon dont les informations d'un formulaire sont traitées une fois que celui-ci est validé par l'internaute.

Le traitement des formulaires est un sujet plus complexe, en dehors du domaine de ce cours.

Pour l'instant nous nous en tiendrons à la création des ceux-ci et à la façon de les styler.

## Initialisation d'un formulaire

Pour ajouter un formulaire à une page, nous utiliserons l'élément `<form>`. L'élément `<form>` identifie l'endroit où les éléments de formulaire apparaîtront dans la page et l'élément `<form>` enveloppera tous les éléments inclus dans le formulaire.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains three lines of HTML code:

```
1 <form action="/login" method="post">
2   ...
3 </form>
```

```
1 <form action="/login" method="post">
2   ...
3 </form>
```

Une poignée de différents attributs peuvent être appliqués à `<form>`, les éléments les plus communs sont `action` et `method`. L'attribut `action` contient l'URL vers laquelle les informations incluses dans le formulaire seront envoyées pour traitement par le serveur. L'attribut `method` est la méthode "HTTP" que le navigateur doit utiliser pour soumettre les données de formulaire.

Ces deux attributs se rapportent à la façon dont sont envoyées et traitées les données vers le serveur.

## Les champs text & textarea

Lorsqu'il s'agit de recueillir la saisie de texte des utilisateurs, il existe quelques éléments différents disponibles pour obtenir des données dans les formulaires. Plus précisément, les champs de texte et les zones de texte sont utilisés pour collecter des données textuelles ou basées sur des chaînes de caractères. Ces données peuvent comprendre des passages de contenu textuel, des numéros de téléphone etc...

### Input text

Un des éléments principaux utilisés pour obtenir le texte des utilisateurs est l'élément `<input>`. L'élément `<input>` utilise l'attribut `type` pour définir quel type d'information doit être insérée dans le contrôle. La valeur d'attribut de `type` la plus populaire est `text`, qui désigne une seule ligne d'entrée de texte.

Avec la définition d'un attribut `type`, il faut aussi donner à un élément `<input>` un attribut `name`. La valeur d'attribut `name` est utilisée comme nom du contrôle et est soumise avec les données entrées au serveur.



```
1 <input type="text" name="nom_utilisateur">
```

L'élément `<input>` est autonome, ce qui signifie qu'il n'utilise qu'une seule balise et qu'il n'a aucun autre contenu. La valeur de l'élément est fournie par ses attributs et leurs valeurs correspondantes.

A l'origine, les deux seules valeurs d'attribut de type texte étaient "text" et "password" (pour les entrées de mot de passe). Cependant, HTML5 a apporté une poignée de nouvelles valeurs d'attribut de type.

Ces valeurs ont été ajoutées pour fournir une signification sémantique plus claire pour les `<input>` ainsi que pour fournir de meilleurs contrôles pour les utilisateurs. Si un navigateur ne comprend pas l'une de ces valeurs d'attribut de type HTML5, il retournera automatiquement à la valeur d'attribut de type texte.

Vous trouverez ci-dessous la liste des nouveaux types d'entrée HTML5.

- *color* : permet de définir une couleur
- *date* : permet de saisir une date (composé d'un jour, d'un mois et d'une année).
- *datetime-local* : permet de saisir une date et une heure (sans fuseau horaire).
- *email* : permet de saisir une adresse électronique.
- *month* : permet de saisir un mois et une année (sans fuseau horaire).
- *number* : permet de saisir un nombre.
- *range* : permet de saisir un nombre dont la valeur exacte n'est pas importante.
- *search* : un champ texte sur une ligne pour des termes de recherche. Les sauts de ligne sont automatiquement retirés.
- *tel* : pour saisir un numéro de téléphone.
- *time* : permet de saisir une heure (sans fuseau horaire).
- *url* : permettant de saisir une URL.
- *week* : permet de saisir une date représentée par un numéro de semaine et une année (sans indication de fuseau horaire).

Certains types sont désormais obsolètes :

- *datetime* : contrôle pour saisir une date et une heure selon un fuseau horaire UTC. Cette fonctionnalité a été retirée du standard WHATWG HTML.

Les éléments `<input>` ci-dessous montrent quelques unes de ces valeurs d'attribut de type HTML5 en cours d'utilisation.

Notez comment les différentes valeurs fournissent des contrôles différents, qui rendent la collecte des entrées des utilisateurs plus facile. Cependant les navigateurs se comportent encore très différemment en fonction des type définis dans le standard HTML5.

Voici un exemple avec du code et le rendu dans plusieurs navigateurs.

```
1 <input type="date" name="anniv">
2 <input type="time" name="temps-jeu">
3 <input type="email" name="adresse-mail">
4 <input type="url" name="site-url">
5 <input type="number" name="cout">
6 <input type="tel" name="num-tel">
```

## Chrome

Date	<input type="text" value="jj/mm/aaaa"/>
Temps	<input type="text" value="--:--"/>
Email	<input type="text" value="aaa@"/>
Url	<input type="text" value="aa"/>
Nombre	<input type="text" value="3"/>
Téléphone	<input type="text" value="06"/>

## Firefox

Date	<input type="text"/>
Temps	<input type="text"/>
Email	<input type="text" value="lau"/>
Url	<input type="text" value="aa"/>
Nombre	<input type="text" value="4"/>
Téléphone	<input type="text" value="06"/>

## Internet Explorer / Edge

Date	<input type="text" value="15"/>
Temps	<input type="text" value="15"/>
Email	<input type="text" value="lau"/>
Url	<input type="text" value="http"/>
Nombre	<input type="text" value="15"/>
Téléphone	<input type="text" value="06"/>

Pour consulter la liste de tous les "input" possibles consultez le site [MDN](#).

## Textarea

Un autre élément utilisé pour capturer des données textuelles est l'élément `<textarea>`. L'élément `<textarea>` diffère de l'élément `<input>` en ce qu'il peut accepter de plus grands passages de texte couvrant plusieurs lignes. L'élément `<textarea>` contient également des balises de début et de fin qui peuvent envelopper du texte brut. L'élément `<textarea>` n'acceptant qu'un seul type de valeur, l'attribut `type` ne s'applique pas ici, mais l'attribut `name` est toujours utilisé.



L'élément `<textarea>` a deux attributs de dimensionnement: `cols` pour la largeur visible à l'écran de la zone de saisie de texte et `rows` pour la hauteur à savoir le nombre de lignes de texte visible. La taille d'une zone de texte, cependant, est plus communément paramétrée en utilisant les propriétés css `width` et `height`.

## Choix à entrées multiples et menus

Outre les contrôles d'entrées basés sur le texte, HTML permet également aux utilisateurs de sélectionner des données à l'aide de choix multiples et de listes déroulantes. Il existe quelques options et éléments différents pour ces contrôles de formulaire, dont chacun présente des avantages distincts.

### Boutons radio

Les boutons radio sont un moyen facile de permettre aux utilisateurs de faire un choix rapide à partir d'une petite liste d'options. Les boutons radio permettent aux utilisateurs de sélectionner une seule option parmi plusieurs options.

Pour créer un bouton radio, l'élément `<input>` est utilisé avec une valeur d'attribut de type `radio`. Chaque élément de bouton radio doit avoir la même valeur d'attribut `name` pour que tous les boutons d'un groupe soient liés les uns aux autres.

Avec les entrées textuelles, la valeur d'une entrée est déterminée par ce qu'un utilisateur tape dans le champs. Avec des boutons radio un utilisateur effectue une sélection dans une liste de choix multiple. Aussi nous devons définir la valeur d'entrée. En utilisant l'attribut `value`, nous pouvons définir une valeur spécifique pour chaque élément `<input>`.

En outre, pour présélectionner un bouton radio pour les utilisateurs, nous pouvons utiliser l'attribut booléen `checked`.

```
1 <input type="radio" name="jour" value="ven" checked> Vendredi  
2 <input type="radio" name="jour" value="sam"> Samedi  
3 <input type="radio" name="jour" value="dim"> Dimanche
```

☒ Vendredi ☐ Samedi ☐ Dimanche

## Cases à cocher

Les cases à cocher sont très semblables aux boutons radio. Ils utilisent les mêmes attributs et modèles, à l'exception de la valeur d'attribut de type `checkbox`. La différence entre les deux est que les cases à cocher permettent aux utilisateurs de sélectionner plusieurs valeurs et de les lier à un nom de contrôle -attribut `name`-, alors que les boutons radio limitent les utilisateurs à une seule valeur.

```
1 <input type="checkbox" name="jour" value="friday" checked> Vendredi  
2 <input type="checkbox" name="jour" value="saturday"> Samedi  
3 <input type="checkbox" name="jour" value="sunday"> Dimanch
```

☒ Vendredi ☐ Samedi ☐ Dimanche

## Les listes déroulantes

Les listes déroulantes sont un moyen parfait de fournir aux utilisateurs une longue liste d'options de manière pratique. Une longue colonne de boutons radio à côté d'une liste de différentes options est non seulement

visuellement peu attrayante mais aussi difficile à utiliser en particulier sur un appareil mobile.

Pour créer une liste déroulante, nous utiliserons les éléments `<select>` et `<option>`. L'élément `<select>` enveloppe toutes les options de menu et chaque option de menu est marquée à l'aide de l'élément `<option>`.

L'attribut `name` réside sur l'élément `<select>` et l'attribut `value` réside dans les éléments `<option>` qui sont imbriqués dans l'élément `<select>`. L'attribut `value` de chaque élément `<option>` correspond alors à l'attribut `name` de l'élément `<select>`.

Chaque élément `<option>` enveloppe le texte (visible pour les utilisateurs) d'une option individuelle dans la liste.

Tout comme l'attribut booléen `checked` pour les boutons radio et les cases à cocher, les menus déroulants peuvent utiliser l'attribut booléen `selected` pour présélectionner une option pour les utilisateurs.

```
1 <select name="jour">
2   <option value="Vendredi">Vendredi</option>
3   <option value="Samedi" selected>Samedi</option>
4   <option value="Dimanche">Dimanche</option>
5 </select>
```



## Sélection multiple

L'attribut booléen `multiple`, lorsqu'il est ajouté à l'élément `<select>` pour une liste déroulante standard, permet à un utilisateur de choisir plus d'une option de la liste à la fois. En outre, l'utilisation de l'attribut booléen `selected` sur plus d'un élément `<option>` dans le menu présélectionnera plusieurs options.



La taille de l'élément `<select>` peut être contrôlée à l'aide de CSS et doit être ajustée de manière appropriée pour permettre plusieurs sélections. Il peut être utile d'informer les utilisateurs que pour choisir plusieurs options, ils devront maintenir la touche Maj enfoncée tout en cliquant pour faire leurs sélections.

```
1 <select name="jour" multiple>
2   <option value="Vendredi">Vendredi</option>
3   <option value="Samedi" selected>Samedi</option>
4   <option value="Dimanche">Dimanche</option>
5 </select>
```



## Les boutons

Une fois qu'un utilisateur a saisi les informations demandées, un bouton de validation permet à l'utilisateur d'envoyer ces informations, de les valider, pour les faire parvenir à l'url précisée dans l'attribut `action` de la balise `form`. Ce bouton, `button`, doit avoir le type `submit`. On peut aussi utiliser un élément, `input`, aussi avec le type `submit` afin de pouvoir déclencher le traitement des données.

### Soumettre une valeur avec "input"

Les utilisateurs cliquent sur le bouton soumettre pour traiter les données après avoir rempli un formulaire. Le bouton de soumission est créé en utilisant l'élément `<input>` avec une valeur d'attribut de type `submit`. L'attribut `value` est utilisé pour spécifier le texte qui apparaît dans le bouton.

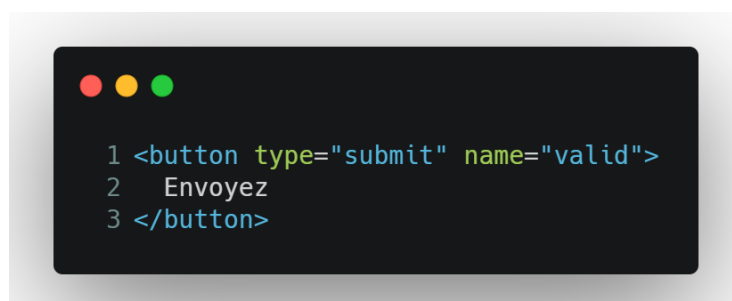


## Soumettre une valeur avec un bouton “submit”

L'élément `<input>` est autonome et n'a aucun autre contenu. Si vous désirez plus de contrôle sur la structure et la conception du bouton - notamment la possibilité d'y mettre d'autres éléments-, l'élément `<button>` peut être utilisé.

L'élément `<button>` fonctionne de la même manière que l'élément `<input>` avec la valeur d'attribut `type` de `submit`. Cependant, il comprend des balises d'ouverture et de fermeture, qui peuvent contenir d'autres éléments. Par défaut, l'élément `<button>` agit comme s'il avait une valeur d'attribut de type `submit`, donc l'attribut `type` et la valeur peuvent être omis dans l'élément `<button>` si vous le souhaitez.

Plutôt que d'utiliser l'attribut `value` pour contrôler le texte dans le bouton de soumission, le texte qui apparaît entre les balises d'ouverture et de fermeture de l'élément `<button>` apparaîtra.




## Autres input

Outre les applications dont nous venons de parler, l'élément `<input>` comporte quelques autres cas d'utilisation. Cela inclut le transfert de données cachées et l'ajout de fichiers pendant le traitement du formulaire.

## Input hidden

Les entrées cachées fournissent un moyen de transmettre des données au serveur sans les afficher aux utilisateurs. Les entrées cachées sont généralement utilisées pour le suivi des codes, des clés ou d'autres informations qui ne sont pas pertinentes pour l'utilisateur mais qui sont utiles lors du traitement du formulaire. Si ces informations ne sont pas affichées sur la page, il est cependant facile de les consulter en visualisant le code source d'une page. Elles ne doivent donc pas être utilisées pour des informations sensibles ou sécurisées.

Pour créer une entrée cachée, vous utilisez la valeur `hidden` pour l'attribut `type`. De plus, incluez les valeurs d'attribut de `name` et de `value` appropriées.




```
1 <input type="hidden" name="code" value="123">
```

## Envoyer un fichier

Pour permettre aux utilisateurs d'ajouter un fichier à un formulaire, à peu près comme attacher un fichier à un courrier électronique, utilisez la valeur `file` pour l'attribut `type`.



```
1 <input type="file" name="file">
```



Choisissez un fichier    Aucun fichier choisi

Malheureusement, le style d'un élément `<input>` qui a une valeur d'attribut de type de fichier est une tâche difficile avec CSS. Chaque navigateur a son propre style de bouton par défaut et aucun ne fournit beaucoup de contrôle pour remplacer le style par défaut. JavaScript peut être utilisé pour permettre de styler le bouton de type `file`, mais c'est un peu plus difficile à mettre en oeuvre.

## Organiser les éléments de formulaire


Savoir comment envoyer des données avec les "input" est la moitié de la bataille. Organiser les éléments de formulaires d'une manière utilisable est l'autre moitié. Lors de l'interaction avec les formulaires, les utilisateurs doivent comprendre ce qui leur est demandé et comment fournir les informations demandées.

En utilisant des étiquettes `-label-`, des encadrés `-fieldset-` et des légendes, nous pouvons mieux organiser les formulaires et guider les utilisateurs afin qu'ils les complètent correctement.

### Label

Les étiquettes fournissent des sous-titres ou des en-têtes pour les contrôles de formulaire, les liant sans ambiguïté et créant un formulaire accessible pour tous les utilisateurs et technologies d'assistance. Créés à l'aide de l'élément `<label>`, les étiquettes doivent inclure du texte qui décrit les entrées ou les contrôles auxquels ils appartiennent.

Les labels doivent inclure un attribut `for`. La valeur de l'attribut `for` doit être la même que la valeur de l'attribut `id` sur le contrôle de formulaire auquel le label correspond. L'association des valeurs d'attribut `for` et `id` lie les deux éléments, ce qui permet aux utilisateurs de cliquer sur l'élément `<label>` pour mettre le focus au bon contrôle de formulaire.



```
1 <label for="username">Nom utilisateur</label>
2 <input type="text" name="nom" id="username">
```



Nom

Si vous le souhaitez, l'élément `<label>` peut contenir des contrôles de formulaire, tels que des boutons radio ou des cases à cocher. Il est alors autorisé d'omettre les attributs `for` et `id`.

```
1 <label>
2   <input type="radio" name="jour" value="vendredi" checked> Vendredi
3 </label>
4 <label>
5   <input type="radio" name="jour" value="Samedi"> Samedi
6 </label>
7 <label>
8   <input type="radio" name="jour" value="Dimanche"> Dimanche
9 </label>
```



☒ Vendredi ☐ Samedi ☐ Dimanche

## Fieldset

Les groupes de champs regroupent les contrôles de formulaire et les labels en sections organisées. Tout comme un élément `<section>` ou un autre élément structurel, le `<fieldset>` est un élément au niveau du bloc qui enveloppe les éléments liés, spécifiquement dans un élément `<form>`, pour une meilleure organisation.

Les champs, par défaut, comprennent également un contour de bordure, qui peut être modifié à l'aide de CSS.

```
1 <fieldset>
2   <label>
3     Nom <input type="text" name="nom">
4   </label>
5   <label>
6     Mot de passe <input type="text" name="motpasse">
7   </label>
8 </fieldset>
```

Nom  Mot de passe

## Legend

L'élément `<legend>` fournit une légende ou un en-tête pour l'élément `<fieldset>`. L'élément `<legend>` enveloppe le texte décrivant les contrôles de formulaire qui appartiennent au regroupement. L'élément `<legend>` doit directement être inclu après la balise `<fieldset>` d'ouverture. Sur la page, la légende apparaîtra dans la partie supérieure gauche de la bordure du regroupement.

```

1 <fieldset>
2   <legend>Connexion</legend>
3   <label>
4     Nom <input type="text" name="nom">
5   </label>
6   <label>
7     Mot de passe <input type="password" name="motpasse">
8   </label>
9 </ fieldset>

```

Connexion

Nom  Mot de passe

## Datalist

L'élément HTML `<datalist>` contient un ensemble d'éléments `<option>` qui représentent les valeurs possibles pour d'autres contrôles.

```

1 <label for="monNavigateur">Veuillez choisir un navigateur parmi ceux-ci :</label>
2 <input list="navigateurs" id="monNavigateur" name="monNavigateur">
3 <datalist id="navigateurs">
4   <option value="Chrome">
5   <option value="Firefox">
6   <option value="Internet Explorer">
7   <option value="Opera">
8   <option value="Safari">
9 </datalist>

```

Veillez choisir un navigateur parmi ceux-ci :

- Chrome
- Firefox
- Internet Explorer
- Opera
- Safari

## Formulaires & attributs des inputs

Pour tenir compte de tous les différents éléments de formulaire, d'entrée et de contrôle, il existe un certain nombre d'attributs et de valeurs correspondantes. Ces attributs et valeurs servent à une poignée de fonctions différentes, telles que la désactivation des contrôles et l'ajout de validation de formulaire. Voici quelques-uns des attributs les plus fréquemment utilisés et les plus utiles.

### Disabled

L'attribut booléen `disabled` désactive un élément ou un contrôle afin qu'il ne soit pas disponible pour l'interaction ou l'entrée. Les éléments désactivés n'émettront aucune valeur vers le serveur pour le traitement du formulaire.

Appliquer l'attribut `disabled` à un élément `<fieldset>` désactive tous les contrôles de formulaire dans le champ. Si l'attribut `type` a une valeur `hidden`, le champs `hidden` est ignoré.



```
1 <label>
2   Nom <input type="text" name="name" disabled>
3 </label>
```

Nom

## Placeholder

L'attribut HTML5 `placeholder` fournit une aide ou information, dans le contrôle de formulaire d'un élément `<input>` ou `<textarea>` qui disparaît une fois que l'on cliqué ou que l'on prend le focus de l'élément. Ceci est utilisé pour donner aux utilisateurs des informations supplémentaires sur la manière dont l'entrée de formulaire doit être remplie, par exemple, le format d'adresse de courrier à utiliser. Attention cependant, l'information disparaissant, il ne faut pas qu'elle remplace le label ou tout aide importante à comprendre pour un utilisateur.

```
1 <label>
2   Email
3   <input type="email" name="email" placeholder="nom@adresse.com">
4 </label>
```

Email

La principale différence entre les attributs `placeholder` et `value` est que le texte de valeur reste en place lorsqu'un contrôle a le focus, à moins que l'utilisateur ne le supprime manuellement. Cela est idéal pour

pré-remplir des données, telles que des informations personnelles, pour un utilisateur, mais pas pour fournir des suggestions.

## Required

L'attribut booléen HTML5 `required` impose qu'un contrôle d'élément ou de formulaire doit contenir une valeur lors de sa soumission au serveur. Si un élément ou un contrôle de formulaire ne possède pas de valeur, un message d'erreur s'affiche pour demander à l'utilisateur de remplir le champ requis. Actuellement, les styles de message d'erreur sont contrôlés par le navigateur et ne peuvent pas être stylés avec CSS. Les éléments et les contrôles de formulaire non valides, par contre, peuvent être créés en utilisant les pseudo-classes CSS `:optional` et `:required`.

L'attribut `required` est pris en charge pour tous les types d'éléments `<input>` exceptés :

- `color`
- `hidden`
- `range`
- `submit`
- `image`
- `reset`
- `button`

La validation est également spécifique au type d'un contrôle. Par exemple, un élément `<input>` avec une valeur d'attribut de type `email` exigera non seulement qu'une valeur existe dans le contrôle, mais également qu'une adresse de courrier électronique valide.

```
1 <label>
2   Email *
3   <input type="text" name="email" required>
4 </label>
```

Email \*

Veuillez compléter ce champ.

## Attributs supplémentaires

D'autres attributs de contrôle de formulaire comprennent, sans être exhaustif, les éléments suivants. N'hésitez pas à rechercher ces attributs si nécessaire.

- accept
- autocomplete
- autofocus
- min
- max
- maxlength
- pattern
- readonly
- step...

## Exemple de formulaire de connexion

Ce qui suit est un exemple d'un formulaire de connexion complet qui comprend plusieurs éléments et attributs différents pour illustrer ce que nous avons vu jusqu'à présent. Ces éléments sont alors stylés à l'aide de CSS.

HTML

```
1 <form action="" method="">
2 <fieldset class="account-info">
3   <label>
4     Utilisateur
5     <input type="text" name="username">
6   </label>
7   <label>
8     Mot de Passe
9     <input type="password" name="password">
10  </label>
11 </fieldset>
12 <fieldset class="account-action">
13   <input class="btn" type="submit" name="submit" value="Connexion">
14   <label>
15     <input type="checkbox" name="remember"> Rester connecté
16   </label>
17 </fieldset>
18 </form>
```

CSS

```
1 form{
2   border: 1px solid #c6c7cc;
3   border-radius: 5px;
4   font: 14px/1.4 "Helvetica Neue", Helvetica, Arial, sans-serif;
5   overflow: hidden;
6   width: 240px;
7 }
8
9 fieldset{
10  border: 0;
11  margin: 0;
12  padding: 0;
13 }
14
15 input{
16  border-radius: 5px;
17  font:14px/1.4 "Helvetica Neue", Helvetica, Arial, sans-serif;
18  margin: 0;
19 }
20
21 .account-info {
22   padding: 20px 20px 0 20px;
23 }
24
25 .account-info label {
26   color: #395870;
27   display: block;
28   font-weight: bold;
29   margin-bottom: 20px;
30 }
31
32 .account-info input {
33   background: #fff;
34   border: 1px solid #c6c7cc;
35   box-shadow: inset 0 1px 1px rgba(0, 0, 0, .1);
36   color: #636466;
37   padding: 6px;
38   margin-top: 6px;
39   width: 100%;
40 }
41
42 .account-action {
43   background: #f0f0f2;
44   border-top: 1px solid #c6c7cc;
45   padding: 20px;
46 }
47
48 .account-action .btn {
49   background: linear-gradient(#49708f, #293f50);
50   border: 0;
51   color: #fff;
52   cursor: pointer;
53   font-weight: bold;
54   float: left;
55   padding: 8px 16px;
56 }
57
58 .account-action label {
59   color: #7c7c80;
60   font-size: 12px;
61   float: left;
62   margin: 10px 0 0 20px;
63 }
```

**Nom d'utilisateur**

**Mot de passe**

**Login**

☐ Rester connecté

## En pratique

Maintenant que nous savons comment construire des formulaires, créons la page d'inscription pour notre site afin que nous puissions commencer à recueillir les demandes d'inscriptions pour le cours.

En entrant dans notre fichier "inscrire.html", nous commencerons par suivre le même schéma que nous avons utilisé sur nos pages "Programme" et "Lieu". Cela inclut l'ajout d'un élément `<section>` avec une valeur d'attribut de classe "row" juste au-dessous de la section de mise en avant -pilote- et l'imbrication d'un élément `<div>` avec une valeur d'attribut de classe "grid" directement dans l'élément `<section>`. Notre code juste au-dessous de la section principale de la page d'enregistrement devrait ressembler à ceci:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains five lines of HTML code:

```
1 <section class="row">
2   <div class="grid">
3     ...
4   </div>
5 </section>
```

Dans l'élément `<div>` avec une valeur d'attribut de classe de la grille, nous allons créer deux colonnes, l'une couvrant les deux tiers de la largeur de la page et l'autre couvrant un tiers de la largeur de la page. La colonne des deux tiers sera un élément `<section>` sur le côté gauche qui indique aux utilisateurs pourquoi ils doivent s'inscrire à notre conférence. La colonne d'un tiers sera alors un élément `<form>` sur le côté droit, ce qui permettra aux utilisateurs de s'inscrire à notre conférence.

Nous ajouterons ces deux éléments, ainsi que leurs classes "col-2-3" et "col-1-3", directement à l'intérieur de l'élément `<div>` avec une valeur d'attribut de classe "grid". Étant donné que ces deux éléments seront des éléments de blocs en ligne, nous devons ouvrir un commentaire directement après la balise de fermeture des deux tiers et fermer ce

commentaire directement avant la balise d'ouverture d'un tiers de la colonne.

Dans l'ensemble, notre code devrait ressembler ceci:

```
1 <section class="row">
2   <div class="grid">
3     <section class="col-2-3">
4       ...
5     </section><!--
6     --><form class="col-1-3">
7       ...
8     </form>
9   </div>
10 </section>
```

Maintenant, dans notre colonne des deux tiers, nous allons ajouter quelques détails sur notre cours et en quoi c'est une bonne idée que d'y assister. Nous allons le faire en utilisant une poignée de différents niveaux de titre (avec leurs styles préétablis), un paragraphe et une liste non ordonnée.


Dans notre élément `<section>` avec une valeur d'attribut de classe de "col-2-3", le code devrait ressembler à ceci:

```
1 <section class="col-2-3">
2   <h2>Venez prendre un bon cours !</h2>
3   <h5>45 heures à prix maîtrisé</h5>
4   <p>Venez prendre une bonne leçon de HTML et de CSS sans pour autant souffrir. Des cours mais aussi de la
   pratique, beaucoup de pratiques, des exercices réguliers pour exploiter chacune de notions étudiées.
   Alors n'hésitez plus, et n'hésitez pas à amener avec vous un ami ou deux...</p>
5   <h4>Quel contenu ?</h4>
6   <ul>
7     <li>Les bases du HTML</li>
8     <li>Approfondie avec le CSS</li>
9     <li>Des exemples de code</li>
10    <li>Un site complet réalisé en 45h</li>
11  </ul>
12 </section>
```



Actuellement, notre liste non ordonnée n'a pas de marqueur d'élément de liste. Tous les styles par défaut du navigateur ont été désactivés par la réinitialisation CSS que nous avons ajoutée au début de notre site. Nous allons créer certains styles personnalisés spécifiquement pour cette liste non ordonnée.

La nouvelle section au bas de notre fichier "style.css" devrait ressembler à ceci:



```
1 <ul class="pourquoi-inscrire">
2   ...
3 </ul>
```

Avec une classe disponible pour ajouter des styles, créez une nouvelle section pour les styles de page d'enregistrement au bas de notre fichier main.css. Dans cette section, nous allons utiliser la classe pour sélectionner la liste non ordonnée et ajouter une liste de style carré et quelques marges inférieure et gauche.

La nouvelle section au bas de notre fichier "style.css" devrait ressembler à ceci:



```
1 /*
2  =====
3  S'inscrire
4  =====
5 */
6
7 .pourquoi-inscrire{
8   list-style: square;
9   margin: 0 0 22px 30px;
10 }
```

La section des détails de notre page d'inscription est complète, il est maintenant temps de répondre à notre formulaire d'inscription. Nous

allons commencer par ajouter les attributs `action` et `method` à l'élément `<form>`. Étant donné que nous n'avons pas mis en place notre traitement de formulaire, ces attributs serviront simplement d'espace réservé et devront être réexaminés.

Le code de notre élément `<form>` doit ressembler à ceci:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains three lines of HTML code:

```
1 <form class="col-1-3" action="#" method="post">
2   ...
3 </ form>
```

Ensuite, dans l'élément `<form>`, nous ajouterons un élément `<fieldset>`. Dans l'élément `<fieldset>`, nous ajouterons une série d'éléments `<label>` qui enveloppent un contrôle de formulaire donné.

Nous voulons recueillir le nom d'un utilisateur, l'adresse e-mail, son niveau de connaissance sur le sujet et tous les commentaires possibles. Le nom, l'adresse e-mail et le niveau de connaissance sont obligatoires, et nous voulons nous assurer que nous utilisons les éléments et attributs appropriés pour chaque contrôle de formulaire.

Avec un mélange de différents types d'entrée, sélectionnez des menus, des zones de texte et des attributs, le code de notre formulaire devrait ressembler à ce qui suit:

```

1 <form class="col-1-3" action="" method="post">
2 <fieldset>
3 <label>Nom
4 <input type="text" name="name" placeholder="Nom complet" required>
5 </label>
6 <label>Courriel
7 <input type="email" name="courriel" placeholder="Adresse mail / Courriel" required></label>
8 <label>
9 Niveau de connaissance
10 <select name="quantite" required>
11 <option value="1" selected>Débutant</option>
12 <option value="2">Faux débutant</option>
13 <option value="3">Bonne connaissance</option>
14 <option value="4">Ninja</option>
15 </select>
16 </label>
17 <label>
18 Commentaires
19 <textarea name="commentaires"></textarea>
20 </label>
21 </fieldset>
22 <input type="submit" name="submit" value="S'inscrire">
23 </form>

```

Ici, nous pouvons voir chaque contrôle de formulaire imbriqué dans un élément `<label>`. Le contrôle de formulaire `name` utilise un élément `<input>` avec une valeur d'attribut de type de texte, alors que le contrôle de formulaire nommé `courriel` utilise un élément `<input>` avec une valeur d'attribut de type `email`.

Les contrôles de nom et de courriel incluent l'attribut booléen `required` et un attribut `placeholder`.

Le contrôle de formulaire niveau de connaissance utilise les éléments `<select>` et les valeurs `<option>`. L'élément `<select>` contient lui-même l'attribut booléen `required` et le premier élément `<option>` inclura l'attribut booléen `selected`.

Le contrôle de formulaire Commentaires utilise l'élément `<textarea>` sans modifications particulières. Et enfin, en dehors de l'élément `<fieldset>`, il y a le contrôle de formulaire submit, qui est formé par un élément `<input>` avec une valeur d'attribut de type `submit`.

Avec le formulaire en place, il est temps d'y ajouter des styles. Nous allons commencer par quelques styles par défaut sur l'élément `<form>` lui-même et sur les éléments `<input>`, `<select>` et `<textarea>`.

Dans la section "S'inscrire" de notre fichier style.css, nous souhaitons ajouter les styles suivant:

```
1 form{
2   margin-bottom: 22px;
3 }
4
5 input,
6 select,
7 textarea{
8   font: 300 16px/22px "Lato", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
9 }
```

Nous allons commencer par placer une marge de 22 pixels en dessous de notre formulaire pour aider à l'espace verticalement des autres éléments. Ensuite, nous allons ajouter des styles basés sur les polices standard, y compris le poids, la taille, la hauteur de ligne et la famille, pour tous les éléments `<input>`, `<select>` et `<textarea>`.

Par défaut, chaque navigateur a sa propre interprétation de la façon dont les styles pour les contrôles de formulaire doivent apparaître. Dans cet esprit, nous avons répété les styles basés sur la police à partir de notre élément `<body>` pour nous assurer que nos styles restent cohérents.

Ajoutons quelques styles aux éléments de l'élément `<fieldset>`. Puisque nous pouvons ajouter d'autres éléments `<fieldset>` plus tard, ajoutons une valeur d'attribut de classe `inscrire-groupe` à notre élément `<fieldset>` existant, et à partir de là, nous pouvons appliquer des styles uniques aux éléments nichés en elle.

```
1 <fieldset class="inscrire-groupe">
2   ...
3 </fieldset>
```

Une fois que la valeur de l'attribut de la classe `inscrire-groupe` est en place, nous allons ajouter quelques styles aux éléments imbriqués dans l'élément `<fieldset>`. Ces styles apparaîtront dans notre fichier style.css, sous les styles de formulaire existants.



```
1 .inscrire-groupe label {
2   color: #648880;
3   cursor: pointer;
4   font-weight: 400;
5 }
6
7
8 .inscrire-groupe input,
9 .inscrire-groupe select,
10 .inscrire-groupe textarea {
11   border: 1px solid #c6c9cc;
12   border-radius: 5px;
13   color: #888;
14   display: block;
15   margin: 5px 0 27px 0;
16   padding: 5px 8px;
17 }
18
19 .inscrire-groupe label,
20 .inscrire-groupe input,
21 .inscrire-groupe select,
22 .inscrire-groupe textarea{
23   width: 100%;
24 }
25
26
27 .inscrire-groupe select {
28   height: 34px;
29 }
30
31 .inscrire-groupe textarea {
32   height: 78px;
33 }
```

Vous remarquerez que la plupart de ces propriétés et valeurs tournent autour du modèle de boîte, que nous avons déjà abordé. Nous faisons principalement la mise en place de la taille des différents contrôles de formulaire, en veillant à ce qu'ils soient disposés de manière appropriée. Mis à part l'ajout de styles de modèles de boîtes, nous adaptons la couleur et la police de caractères de quelques éléments.

A présent notre formulaire présente bien. Le seul élément restant à désigner est le bouton de soumission. Comme c'est un bouton, nous avons en fait déjà quelques styles existants que nous pouvons appliquer ici. Si nous revenons à notre page d'accueil, notre section `leader` contient un bouton qui a reçu certains styles par le biais de la classe `btn`.

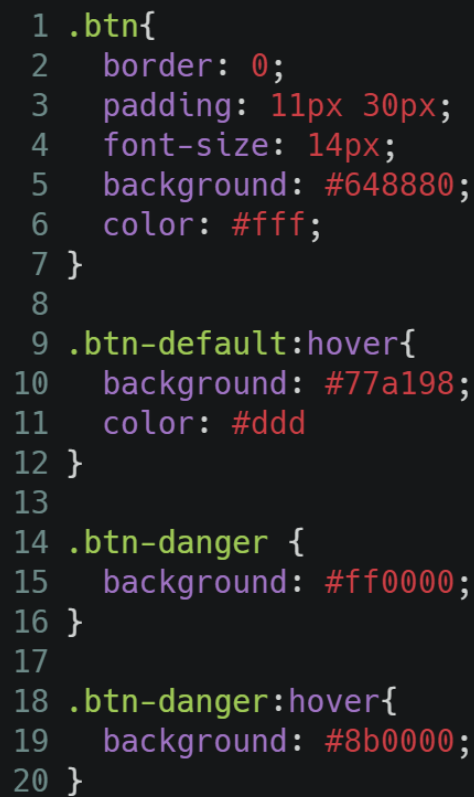
Ajoutons cette classe, `btn`, ainsi qu'une nouvelle valeur d'attribut de classe `btn-default` à notre bouton de soumission. Nous utiliserons le nom de classe `btn-default` puisque ce bouton apparaît sur un fond blanc et sera le style par défaut pour les boutons avancés.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains a single line of HTML code.

```
1 <input class="btn btn-default" type="submit" name="submit" value="S'inscrire">
```

Maintenant, notre bouton de soumission a certains styles partagés avec le bouton de la page d'accueil. Nous utiliserons les classes `btn` et `btn-default` pour ensuite appliquer spécifiquement quelques nouveaux styles à notre bouton de soumission.

Revenons à la section des boutons de notre fichier "style.css", ajoutons ce qui suit:



```
1 .btn{
2   border: 0;
3   padding: 11px 30px;
4   font-size: 14px;
5   background: #648880;
6   color: #fff;
7 }
8
9 .btn-default:hover{
10  background: #77a198;
11  color: #ddd
12 }
13
14 .btn-danger {
15   background: #ff0000;
16 }
17
18 .btn-danger:hover{
19   background: #8b0000;
20 }
```

Ces nouveaux styles, qui définissent la taille et l'arrière-plan de notre bouton de soumission, sont ensuite combinés avec les styles de classe `btn` existants pour créer la présentation finale de notre bouton de soumission.

Notre page d'inscription est terminée ! Les volontaires peuvent maintenant commencer à s'inscrire au cours.

## S'inscrire

L'inscription est ouverte du XX au YY...

Venez prendre une bonne leçon de HTML et de CSS à un prix défiant toute concurrence !

### Venez prendre un bon cours !

#### 45 HEURES À PRIX MAÎTRISÉ

Venez prendre une bonne leçon de HTML et de CSS sans pour autant souffrir. Des cours mais aussi de la pratique, beaucoup de pratiques, des exercices réguliers pour exploiter chacune de notions étudiées. Alors n'hésitez plus, et n'hésitez pas à amener avec vous un ami ou deux...

#### Quel contenu ?

- Les bases du HTML
- Approfondie avec le CSS
- Des exemples de code
- Un site complet réalisé en 45h

#### Nom

#### Courriel

#### Niveau de connaissance

#### Commentaires

S'INSCRIRE