**Créez des formulaires**

Toute page HTML peut être enrichie de formulaires interactifs, qui invitent vos visiteurs à renseigner des informations : saisir du texte, sélectionner des options, valider avec un bouton… tout est possible !

Nous arrivons cependant aux limites du langage HTML car il faut ensuite pouvoir analyser les informations que le visiteur a saisies… et cela ne peut pas se faire en langage HTML. Comme nous allons le voir, le traitement des résultats doit s'effectuer dans un autre langage, par exemple le PHP.

En attendant, nous avons un grand nombre de nouvelles balises HTML à découvrir. Bienvenue dans le monde merveilleux des formulaires, un monde où les boutons, les cases à cocher et les listes déroulantes vivent en harmonie (enfin presque).

**Créer un formulaire**

Lorsqu'il vous prend subitement l'envie d'insérer un formulaire dans votre page HTML, vous devez pour commencer écrire une balise <form> </form>  . C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

<p>Texte avant le formulaire</p>

<form>

<p>Texte à l'intérieur du formulaire</p>

</form>

<p>Texte après le formulaire</p>

Notez qu'il faut obligatoirement mettre des balises de type block (comme <p> </p>  ) à l'intérieur de votre formulaire si vous souhaitez y faire figurer du texte.

Voilà pour la structure de base. Maintenant, soyez attentif : ce que j'ai à vous dire n'est pas évident, parce qu'on est à la limite du HTML.

On va prendre un exemple pour que les choses soient claires. Supposons que votre visiteur vienne de taper un commentaire dans votre formulaire, par exemple un message qu'il aimerait publier sur vos forums. Ce message doit être *envoyé* pour que vous puissiez le recevoir (logique, non ?) et l'afficher pour vos autres visiteurs.

Eh bien c'est là le problème, ou plutôt les problèmes, que l'on va se poser :

* **Problème n° 1** : comment envoyer le texte saisi par le visiteur ? Par quel moyen ?
* **Problème n° 2** : une fois que les données ont été envoyées, comment les traiter ? Souhaitez-vous recevoir le message automatiquement par mail ou préférez-vous qu'un programme se charge de l'enregistrer quelque part, puis de l'afficher sur une page visible par tout le monde ?

Pour fournir les réponses à ces deux problèmes, vous devez ajouter deux attributs à la balise <form>  :

* method  : cet attribut indique par quel moyen les données vont être envoyées (réponse au **problème n° 1**). Il existe deux solutions pour envoyer des données sur le Web :
  + method="get"  : c'est une méthode en général assez peu adaptée car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page ( http://…  ), mais ce détail ne nous intéresse pas vraiment pour le moment. La plupart du temps, je vous recommande d'utiliser l'autre méthode : post  ,
  + method="post"  : c'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse ;
* action  : c'est l'adresse de la page ou du programme qui va *traiter* les informations (réponse au **problème n° 2**). Cette page se chargera de vous envoyer un e-mail avec le message si c'est ce que vous voulez, ou bien d'enregistrer le message avec tous les autres dans une base de données.  
  Cela ne peut pas se faire en HTML et CSS ; on utilisera en général un autre langage dont vous avez peut-être entendu parler : PHP.

On va donc maintenant compléter la balise <form>  avec les deux attributs qu'on vient de voir.

Pour method  , vous l'aurez deviné, je vais mettre la valeur post  .  
Pour action  , je vais taper le nom d'une page fictive en PHP ( traitement.php  ). C'est cette page qui sera appelée lorsque le visiteur cliquera sur le bouton d'envoi du formulaire.

<p>Texte avant le formulaire</p>

<form method="post" action="traitement.php">

<p>Texte à l'intérieur du formulaire</p>

</form>

<p>Texte après le formulaire</p>

Pour le moment, on ne sait pas ce qu'il se passe à l'intérieur de la page traitement.php  : je vous demande de me faire confiance et d'imaginer que cette page existe et fonctionne.

Notre priorité, pour le moment, est de découvrir en HTML/CSS comment faire pour insérer des zones de texte, des boutons et des cases à cocher dans votre page web. C'est ce que nous allons voir maintenant.

**Les zones de saisie basiques**

Bien, retour au concret.

Nous allons passer en revue les différentes balises HTML permettant de saisir du texte dans un formulaire.  
Il faut savoir qu'il y a deux zones de texte différentes :

* **la zone de texte monoligne** : comme son nom l'indique, on ne peut y écrire qu'une seule ligne. Elle sert à saisir des textes courts, par exemple un pseudo ;
* **la zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes, par exemple une dissertation sur l'utilité du HTML dans le développement des pays d'Asie du Sud-Est (ce n'est qu'une suggestion, hein…).

**Zone de texte monoligne**

La figure suivante montre à quoi ressemble une zone de texte monoligne.

Une zone de texte monoligneUne zone de texte monoligne

Pour insérer une zone de texte dans une ligne, on va utiliser la balise <input />  .

On retrouvera cette balise plusieurs fois dans la suite de ce chapitre. À chaque fois, c'est la valeur de son attribut type  qui va changer.

Pour créer une zone de texte à une ligne, on doit écrire :

<input type="text" />

Ce n'est pas encore suffisant : il faut donner un nom à votre zone de texte. Ce nom n'apparaît pas sur la page, mais il vous sera indispensable par la suite. En effet, cela vous permettra (en PHP par exemple) de reconnaître d'où viennent les informations : vous saurez que tel texte est le pseudo du visiteur, tel texte est le mot de passe qu'il a choisi, etc.

Pour donner un nom à un élément de formulaire, on utilise l'attribut name  . Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :

<input type="text" name="pseudo" />

Essayons donc de créer un formulaire très basique avec notre champ de texte :

<form method="post" action="traitement.php">

<p><input type="text" name="pseudo" /></p>

</form>

Comme d'habitude, je vous invite fortement à essayer ce code chez vous afin d'observer le résultat.

**Les libellés**

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise <label>  :

<form method="post" action="traitement.php">

<p>

<label>Votre pseudo</label> : <input type="text" name="pseudo" />

</p>

</form>

Ce code donne exactement le résultat que vous avez pu observer à la figure précédente.

Mais cela ne suffit pas. Il faut lier le label à la zone de texte.  
Pour ce faire, on doit donner un nom à la zone de texte, non pas avec l'attribut name  mais avec l'attribut id  (que l'on peut utiliser sur toutes les balises).

Un name  et un id  sur le champ ? Cela ne va-t-il pas faire double emploi ?

Pas du tout ! En fait, l'id est utilisé pour **identifier** l'élément HTML pour pouvoir y accéder et le manipuler. Il est donc unique pour cet élément.

Le name , lui, réfère à la variable du formulaire que l’élément concerne. Ici, il n’y a qu’un seul élément qui pourra référer à la variable *pseudo* ;name et id auront donc la même valeur. Mais lorsque nous utiliserons des checkbox ou des radio , plusieurs éléments correspondront à la même variable. Par exemple, la variable *couleur*avec un élément pour *rouge*, un pour *bleu* et un pour *vert*. Ils auront donc le même name , mais pas le même id  .

Pour lier le label au champ, il faut lui donner un attribut for  qui a la même valeur que l' id  du champ… Le mieux est de le voir sur un exemple :

<form method="post" action="traitement.php">

<p>

<label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo" id="pseudo" />

</p>

</form>

Essayez de cliquer sur le texte « Votre pseudo » : vous allez voir que le curseur se place automatiquement dans la zone de texte correspondante.

**Quelques attributs supplémentaires**

On peut ajouter un certain nombre d'autres attributs à la balise <input />  pour personnaliser son fonctionnement :

* on peut agrandir le champ avec size  ;
* on peut limiter le nombre de caractères que l'on peut saisir avec maxlength  ;
* on peut préremplir le champ avec une valeur par défaut à l'aide de value  ;
* on peut donner une indication sur le contenu du champ avec placeholder  . Cette indication disparaîtra dès que le visiteur aura cliqué à l'intérieur du champ.

Dans l'exemple suivant, la zone de texte contient une indication permettant de comprendre ce qu'il faut saisir ; le champ fait 30 caractères de long mais on ne peut écrire que 10 caractères maximum à l'intérieur :

<form method="post" action="traitement.php">

<p>

<label for="pseudo">Votre pseudo :</label>

<input type="text" name="pseudo" id="pseudo" placeholder="Ex : Zozor" size="30" maxlength="10" />

</p>

</form>

Testez vous-mêmes le résultat pour observer le comportement du champ. En attendant, voici le rendu du champ dans son état initial en figure suivante.

Un champ de texte avec une indication (placeholder)

Un champ de texte avec une indication (placeholder)

**Zone de mot de passe**

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une « zone de mot de passe », c'est-à-dire une zone où on ne voit pas à l'écran les caractères saisis. Pour créer ce type de zone de saisie, utilisez l'attribut type="password"  .

Je complète mon formulaire. Il demande maintenant au visiteur son pseudo *et* son mot de passe :

<form method="post" action="traitement.php">

<p>

<label for="pseudo">Votre pseudo :</label>

<input type="text" name="pseudo" id="pseudo" />

<br />

<label for="pass">Votre mot de passe :</label>

<input type="password" name="pass" id="pass" />

</p>

</form>

Testez la zone de mot de passe : vous verrez que les caractères ne s'affichent pas à l'écran, comme sur la figure suivante.

Une zone de saisie de mot de passe

Une zone de saisie de mot de passe

**Zone de texte multiligne**

Pour créer une zone de texte multiligne, on change de balise : nous allons utiliser <textarea> </textarea>  .

Comme pour tout autre élément du formulaire, il faut lui donner un nom avec name  et utiliser un label  qui explique de quoi il s'agit.

<form method="post" action="traitement.php">

<p>

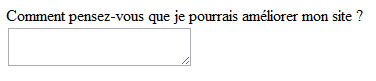
<label for="ameliorer">Comment pensez-vous que je pourrais améliorer mon site ?</label><br />

<textarea name="ameliorer" id="ameliorer"></textarea>

</p>

</form>

Et voici le résultat en image (figure suivante) !

Une (petite) zone de saisie multiligne

Comme vous pouvez le constater, c'est un peu petit ! Heureusement, on peut modifier la taille du <textarea>  de deux façons différentes.

* **En CSS** : il suffit d'appliquer les propriétés CSS width  et height  au <textarea>  .
* **Avec des attributs** : on peut ajouter les attributs rows  et cols  à la balise <textarea>  . Le premier indique le nombre de lignes de texte qui peuvent être affichées simultanément, et le second le nombre de colonnes.

Pourquoi ouvre-t-on la balise <textarea>  pour la fermer juste après ?

Vous pouvez préremplir le <textarea>  avec une valeur par défaut. Dans ce cas, on n'utilise pas l'attribut value  : on écrit tout simplement le texte par défaut entre la balise ouvrante et la balise fermante !

<form method="post" action="traitement.php">

<p>

<label for="ameliorer">

Comment pensez-vous que je puisse améliorer mon site ?

</label>

<br />

<textarea name="ameliorer" id="ameliorer" rows="10" cols="50">

Améliorer ton site ?!

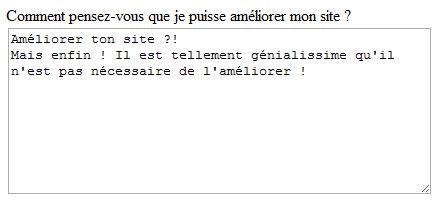
Mais enfin ! Il est tellement génialissime qu'il n'est pas nécessaire de l'améliorer !

</textarea>

</p>

</form>

Et voici le résultat à la figure suivante.

Une zone de saisie multiligne préremplie

**Les zones de saisie enrichies**

HTML5 apporte de nombreuses fonctionnalités nouvelles relatives aux formulaires. De nouveaux types de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut type  de la balise <input />  l'une des nouvelles valeurs disponibles. Faisons un petit tour d'horizon !

Tous les navigateurs ne connaissent pas encore ces zones de saisie enrichies. À leur place, les anciennes versions des navigateurs afficheront une simple zone de saisie monoligne (comme si on avait écrit type="text"  ). Entre nous, c'est parfait : les nouveaux navigateurs peuvent profiter des dernières fonctionnalités, tandis que les anciens affichent une zone de texte de remplacement qui convient tout aussi bien.  
Vous avez donc tout intérêt à utiliser ces nouvelles zones de saisie dès aujourd'hui ! Au mieux, vos visiteurs profiteront des nouvelles fonctionnalités, au pire, ils ne verront aucun problème.

**E-mail**

Vous pouvez demander à saisir une adresse e-mail :

<input type="email" />

Le champ vous semblera *a priori* identique, mais votre navigateur sait désormais que l'utilisateur doit saisir une adresse e-mail. Il peut afficher une indication si l'adresse n'est pas un e-mail ; c'est ce que fait Firefox, par exemple (figure suivante).

Un champ e-mail mal renseigné est entouré de rouge dans FirefoxUn champ e-mail mal renseigné est entouré de rouge dans Firefox

Sachez que certains navigateurs, comme les navigateurs mobiles sur iPhone et Android, affichent un clavier adapté à la saisie d'e-mail (figure suivante).

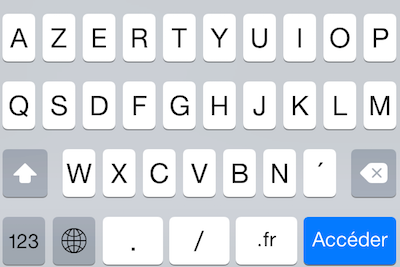
Clavier de saisie d'e-mail sur un iPhone

**Une URL**

Avec le type url  , on peut demander à saisir une adresse absolue (commençant généralement par http://  ) :

<input type="url" />

Même principe : si le champ ne vous semble pas différent sur votre ordinateur, sachez que celui-ci comprend bel et bien que le visiteur est censé saisir une adresse. Les navigateurs mobiles affichent par exemple un clavier adapté à la saisie d'URL (figure suivante).

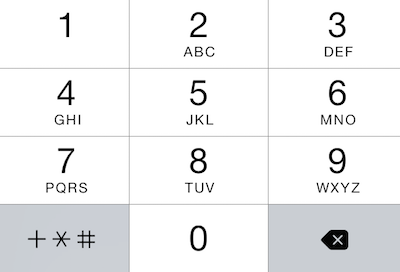
Clavier de saisie d'URL sur iPhone

**Numéro de téléphone**

Ce champ est dédié à la saisie de numéros de téléphone :

<input type="tel" />

Sur iPhone, par exemple, un clavier adapté s'affiche lorsqu'on doit remplir le champ (figure suivante).

Clavier de saisie de numéro de téléphone sur un iPhone

**Nombre**

Ce champ permet de saisir un nombre entier :

<input type="number" />

Le champ s'affichera en général avec des petites flèches pour changer la valeur (figure suivante).

Champ de saisie de nombre

Champ de saisie de nombre

Vous pouvez personnaliser le fonctionnement du champ avec les attributs suivants :

* min  : valeur minimale autorisée ;
* max  : valeur maximale autorisée ;
* step  : c'est le « pas » de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6…).

**Un curseur**

Le type range  permet de sélectionner un nombre avec un curseur (aussi appelé *slider*), comme à la figure suivante :

<input type="range" />

Un curseur grâce au type range

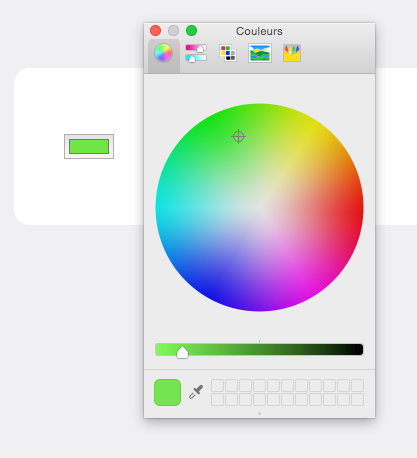
Un curseur grâce au type range

Vous pouvez utiliser là aussi les attributs min  , max  et step  pour restreindre les valeurs disponibles.

**Couleur**

Ce champ permet de saisir une couleur :

<input type="color" />

Un champ de type color

Attention, tous les navigateurs [ne connaissent pas encore ce type de champ](https://caniuse.com/#feat=input-color), mais la compatibilité progresse.

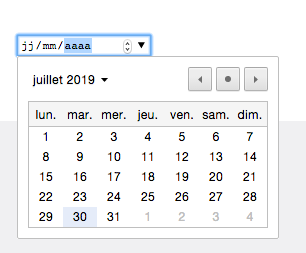
**Date**

Différents types de champs de sélection de date existent :

* date  : pour la date (05/08/1985 par exemple) ;
* time  : pour l'heure (13:37 par exemple) ;
* week  : pour la semaine ;
* month  : pour le mois ;
* datetime  : pour la date et l'heure (avec gestion du décalage horaire) ;
* datetime-local  pour la date et l'heure (sans gestion du décalage horaire).

Exemple :

<input type="date" />

Un champ de type date

Comme vous le voyez, il y a le choix !

Surveillez bien [quels navigateurs gèrent ce type de champ](https://caniuse.com/#feat=input-datetime), il n'est pas encore complètement reconnu.

**Recherche**

On peut créer un champ de recherche comme ceci :

<input type="search" />

Le navigateur décide ensuite comment afficher le champ de recherche. Ainsi, il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche, et éventuellement mémoriser les dernières recherches effectuées par le visiteur.

**Les éléments d'options**

HTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. Ce sont des éléments qui demandent au visiteur de faire un choix parmi une liste de possibilités. Nous allons passer en revue :

* les cases à cocher ;
* les zones d'options ;
* les listes déroulantes.

**Les cases à cocher**

Créer une case à cocher ? Rien de plus simple ! Nous allons réutiliser la balise <input />  , en spécifiant cette fois le type checkbox  :

<input type="checkbox" name="choix" />

Rajoutez un <label>  bien placé, et le tour est joué !

<form method="post" action="traitement.php">

<p>

Cochez les aliments que vous aimez manger :<br />

<input type="checkbox" name="frites" id="frites" /> <label for="frites">Frites</label><br />

<input type="checkbox" name="steak" id="steak" /> <label for="steak">Steak haché</label><br />

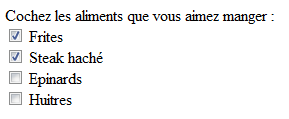
<input type="checkbox" name="epinards" id="epinards" /> <label for="epinards">Epinards</label><br />

<input type="checkbox" name="huitres" id="huitres" /> <label for="huitres">Huitres</label>

</p>

</form>

Et voici le résultat en figure suivante.

Des cases à cocher

N'oubliez pas de donner un nom différent à chaque case à cocher, cela vous permettra d'identifier plus tard lesquelles ont été cochées par le visiteur.

Enfin, sachez que vous pouvez faire en sorte qu'une case soit cochée par défaut, avec l'attribut checked  :

<input type="checkbox" name="choix" checked />

Normalement, tout attribut possède une valeur. Dans le cas présent, en revanche, ajouter une valeur n'est pas obligatoire : la présence de l'attribut suffit à faire comprendre à l'ordinateur que la case doit être cochée.  
Si cela vous perturbe, sachez que vous pouvez donner n'importe quelle valeur à l'attribut (certains webmasters écrivent parfois checked="checked"  , mais c'est un peu redondant !). Dans tous les cas, la case sera cochée.

**Les zones d'options**

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités. Elles ressemblent un peu aux cases à cocher, mais il y a une petite difficulté supplémentaire : elles doivent être organisées en groupes. Les options d'un même groupe possèdent le même nom ( name  ), mais chaque option doit avoir une valeur ( value  ) différente.

La balise à utiliser est toujours un <input />  , avec cette fois la valeur radio  pour l'attribut type  .

Les choses seront plus claires sur l'exemple ci-dessous :

<form method="post" action="traitement.php">

<p>

Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />

<input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label><br />

<input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />

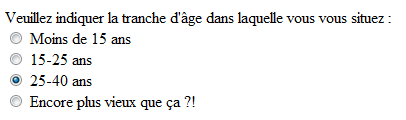
<input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40 ans</label><br />

<input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que ça ?!</label>

</p>

</form>

Ce qui nous donne la figure suivante.

Des boutons radio

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache de quel « groupe » le bouton fait partie.  
Essayez d'enlever les attributs name  , vous verrez qu'il devient possible de sélectionner tous les éléments d'options. Or, ce n'est pas ce que l'on veut ; c'est pour cela qu'on les « lie » entre eux en leur donnant un nom identique.

Vous noterez que cette fois on a choisi un id  différent de name  . En effet, les valeurs de name  étant identiques, on n'aurait pas pu différencier les *id* (et vous savez bien qu'un id  doit être unique !). Voilà donc pourquoi on a choisi de donner à l' id  la même valeur que value  .

Si vous avez deux zones d'options différentes, il faut donner un name  unique à chaque groupe, comme ceci :

<form method="post" action="traitement.php">

<p>

Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />

<input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label><br />

<input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />

<input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40 ans</label><br />

<input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que ça ?!</label>

</p>

<p>

Sur quel continent habitez-vous ?<br />

<input type="radio" name="continent" value="europe" id="europe" /> <label for="europe">Europe</label><br />

<input type="radio" name="continent" value="afrique" id="afrique" /> <label for="afrique">Afrique</label><br />

<input type="radio" name="continent" value="asie" id="asie" /> <label for="asie">Asie</label><br />

<input type="radio" name="continent" value="amerique" id="amerique" /> <label for="amerique">Amérique</label><br />

<input type="radio" name="continent" value="australie" id="australie" /> <label for="australie">Australie</label>

</p>

</form>

L'attribut checked  est, là aussi, disponible pour sélectionner une valeur par défaut.

**Les listes déroulantes**

Les listes déroulantes sont un autre moyen élégant de faire un choix parmi plusieurs possibilités. Le fonctionnement est un peu différent. On va utiliser la balise <select> </select>  qui indique le début et la fin de la liste déroulante. On ajoute l'attribut name  à la balise pour donner un nom à la liste.

Puis, à l'intérieur du <select> </select>  , nous allons placer plusieurs balises <option> </option>  (une par choix possible). On ajoute à chacune d'elles un attribut value  pour pouvoir identifier ce que le visiteur a choisi.

Voici un exemple d'utilisation :

<form method="post" action="traitement.php">

<p>

<label for="pays">Dans quel pays habitez-vous ?</label><br />

<select name="pays" id="pays">

<option value="france">France</option>

<option value="espagne">Espagne</option>

<option value="italie">Italie</option>

<option value="royaume-uni">Royaume-Uni</option>

<option value="canada">Canada</option>

<option value="etats-unis">États-Unis</option>

<option value="chine">Chine</option>

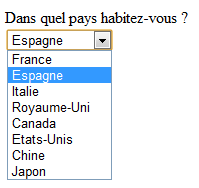
<option value="japon">Japon</option>

</select>

</p>

</form>

Le résultat obtenu est représenté à la figure suivante.



Une liste déroulante

Si vous voulez qu'une option soit sélectionnée par défaut, utilisez cette fois l'attribut selected  :

<option value="canada" selected>Canada</option>

Vous pouvez aussi grouper vos options avec la balise <optgroup> </optgroup>  . Dans notre exemple, pourquoi ne pas séparer les pays en fonction de leur continent ?

<form method="post" action="traitement.php">

<p>

<label for="pays">Dans quel pays habitez-vous ?</label><br />

<select name="pays" id="pays">

<optgroup label="Europe">

<option value="france">France</option>

<option value="espagne">Espagne</option>

<option value="italie">Italie</option>

<option value="royaume-uni">Royaume-Uni</option>

</optgroup>

<optgroup label="Amérique">

<option value="canada">Canada</option>

<option value="etats-unis">Etats-Unis</option>

</optgroup>

<optgroup label="Asie">

<option value="chine">Chine</option>

<option value="japon">Japon</option>

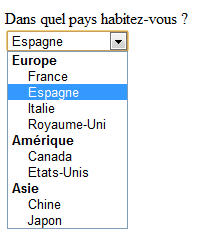
</optgroup>

</select>

</p>

</form>

Le résultat obtenu est représenté à la figure suivante.

[](https://course.oc-static.com/ftp-tutos/cours/html-css/p4/ch2/optgroup)

[Les options sont regroupées par continent](https://s3-eu-west-1.amazonaws.com/course.oc-static.com/ftp-tutos/cours/html-css/p4/ch2/liste_deroulante/index.html)

Les groupes ne peuvent pas être sélectionnés. Ainsi, dans notre exemple, on ne peut pas choisir « Europe » : seuls les noms de pays sont disponibles pour la sélection.

**Finaliser et envoyer le formulaire**

Nous y sommes presque. Il ne nous reste plus qu'à agrémenter notre formulaire de quelques dernières fonctionnalités (comme la validation), puis nous pourrons ajouter le bouton d'envoi du formulaire.

**Regrouper les champs**

Si votre formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises <fieldset>  . Chaque <fieldset>  peut contenir une légende avec la balise <legend>  . Regardez cet exemple :

<form method="post" action="traitement.php">

<fieldset>

<legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

<label for="nom">Quel est votre nom ?</label>

<input type="text" name="nom" id="nom" />

<label for="prenom">Quel est votre prénom ?</label>

<input type="text" name="prenom" id="prenom" />

<label for="email">Quel est votre e-mail ?</label>

<input type="email" name="email" id="email" />

</fieldset>

<fieldset>

<legend>Votre souhait</legend> <!-- Titre du fieldset -->

<p>

Faites un souhait que vous voudriez voir exaucé :

<input type="radio" name="souhait" value="riche" id="riche" /> <label for="riche">Etre riche</label>

<input type="radio" name="souhait" value="celebre" id="celebre" /> <label for="celebre">Etre célèbre</label>

<input type="radio" name="souhait" value="intelligent" id="intelligent" /> <label for="intelligent">Etre <strong>encore</strong> plus intelligent</label>

<input type="radio" name="souhait" value="autre" id="autre" /> <label for="autre">Autre...</label>

</p>

<p>

<label for="precisions">Si "Autre", veuillez préciser :</label>

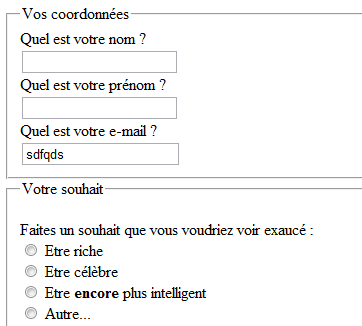
<textarea name="precisions" id="precisions" cols="40" rows="4"></textarea>

</p>

</fieldset>

</form>

Le résultat obtenu est représenté à la figure suivante.

Les champs sont regroupés

**Sélectionner automatiquement un champ**

Vous pouvez placer automatiquement le curseur dans l'un des champs de votre formulaire avec l'attribut autofocus  . Dès que le visiteur chargera la page, le curseur se placera dans ce champ.

Par exemple, pour que le curseur soit par défaut dans le champ prenom  :

<input type="text" name="prenom" id="prenom" autofocus />

**Rendre un champ obligatoire**

Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut required  .

<input type="text" name="prenom" id="prenom" required />

Le navigateur indiquera alors au visiteur, si le champ est vide au moment de l'envoi, qu'il doit impérativement être rempli.

Les anciens navigateurs, qui ne reconnaissent pas cet attribut, enverront le contenu du formulaire sans vérification. Pour ces navigateurs, il sera nécessaire de compléter les tests avec, par exemple, des scripts JavaScript.

On dispose de pseudo-formats en CSS pour changer le style des éléments requis ( :required  ) et invalides ( :invalid  ). N'oubliez pas non plus que vous disposez du pseudo-format :focus  pour changer l'apparence d'un champ lorsque le curseur se trouve à l'intérieur.

:required

{

*background-color*: red;

}

**Le bouton d'envoi**

Il ne nous reste plus qu'à créer le bouton d'envoi. Là encore, la balise <input />  vient à notre secours. Elle existe en quatre versions :

* type="submit"  : le principal bouton d'envoi de formulaire. C'est celui que vous utiliserez le plus souvent. Le visiteur sera conduit à la page indiquée dans l'attribut action  du formulaire ;
* type="reset"  : remise à zéro du formulaire ;
* type="image"  : équivalent du bouton submit  , présenté cette fois sous forme d'image. Rajoutez l'attribut src  pour indiquer l'URL de l'image ;
* type="button"  : bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page. Nous ne l'utiliserons pas ici.

On peut changer le texte affiché à l'intérieur des boutons avec l'attribut value  .

Pour créer un bouton d'envoi, on écrira donc par exemple :

<input type="submit" value="Envoyer" />

Ce qui nous donne la figure suivante.

Un bouton d'envoiUn bouton d'envoi

Lorsque vous cliquez sur le bouton « Envoyer », le formulaire vous amène alors à la page indiquée dans l'attribut action  . Souvenez-vous, nous avions imaginé une page fictive : traitement.php  .

Le problème, c'est que vous ne pouvez pas créer cette page seulement en HTML. Il est nécessaire d'apprendre un nouveau langage, comme le PHP, pour pouvoir « récupérer » les informations saisies et décider quoi en faire. Cela tombe bien, j'ai aussi rédigé un [cours sur le langage PHP](https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql) pour ceux que cela intéresse !

**En résumé**

* Un formulaire est une zone interactive de la page, dans laquelle vos visiteurs peuvent saisir des informations.
* On délimite un formulaire avec la balise <form>  , à laquelle il faut ajouter deux attributs : method  (mode d'envoi des données) et action  (page vers laquelle le visiteur sera redirigé après envoi du formulaire, et qui traitera les informations).
* Une grande partie des éléments du formulaire peut s'insérer avec la balise <input />  . La valeur de son attribut type  permet d'indiquer quel type de champ doit être inséré :
  + text  : zone de texte ;
  + password  : zone de texte pour mot de passe ;
  + tel  : numéro de téléphone ;
  + checkbox  : case à cocher ;
  + etc.
* La balise <label>  permet d'écrire un libellé. On l'associe à un champ de formulaire avec l'attribut for  , qui doit avoir la même valeur que l' id  du champ de formulaire.
* On peut rendre un champ obligatoire avec l'attribut required  , faire en sorte qu'il soit sélectionné par défaut avec autofocus  , donner une indication dans le champ avec placeholder  …
* Pour récupérer ce que les visiteurs ont saisi, le langage HTML ne suffit pas. Il faut utiliser un langage « serveur » comme PHP… Si vous voulez aller plus loin, il va donc falloir apprendre un nouveau langage !