Les formulaires

Informations sur le tutoriel

Ajouter à mes tutoriels favoris (20 fans)

Auteur : M@teo21
Licence :

Plus d'informations

Popularité

Visualisations : 181 547 348

Appréciation des lecteurs : 15
15
60
888

Publicité

Historique des mises à jour

- *Le 08/08/2010 à 03:23:12*
  Correction conjugaison.
- *Le 03/08/2010 à 01:07:48*
  Correction orthographique, #2605
- *Le 31/07/2010 à 18:20:55*
  Correction orthographique, ticket n°2593

Le temps passe, les amis. Plus on avance dans le cours, plus vous allez voir qu'on commence à atteindre les limites.

Comment ça ? On m'aurait menti ?!
Le XHTML et le CSS ont des limites ? On ne peut pas tout faire avec ?!

Ben quoi, j'ai rien dit moi

Plus sérieusement, oui le XHTML et le CSS ont des limites. Vous pouvez déjà faire un bon site avec tout ce que je vous ai appris, mais imaginez qu'un jour vous vouliez faire un *SUPER MEGA site hyper-génial* ?
On y arrive. Ce que je vais vous apprendre, c'est toujours du XHTML, mais vous allez vous rendre compte par vous-mêmes que vous vous heurtez aux "limites" du langage.

En fin de compte, c'est bon signe. Ca veut peut-être dire que vous êtes devenus bons et que le XHTML ne vous suffit plus

De quoi va-t-on parler dans ce chapitre ? A force de divaguer, j'allais oublier quel était le thème du jour !
Nous allons parler des **formulaires**. L'idée est simple : vous avez créé un site, et vous aimeriez par exemple que vos visiteurs puissent vous donner leur avis dessus, en cochant des cases, en entrant leurs commentaires, leurs suggestions....

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)
Sommaire du chapitre :

Créer un formulaire

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

Code : HTML

```
<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) à l'intérieur de votre formulaire si vous avez besoin d'écrire du texte dedans.

Donc ça, c'était la structure de base.
Maintenant restez attentifs, parce que ce que j'ai à vous dire n'est pas évident vu qu'on est à la limite du XHTML.

Il faut savoir qu'un formulaire est fait pour être envoyé. On va prendre un exemple pour que les choses soient claires.
Supposons que votre visiteur vienne de taper un commentaire dans votre formulaire, comme par exemple "Ouah ton site il déchiiiiire". Ce message doit être *envoyé* pour que vous puissiez le recevoir (logique non ?), et afin que vous sachiez ce que le visiteur pense de votre site.

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

- **Problème n°1 :** comment envoyer le texte rentré 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 ?
  Cela revient d'ailleurs à faire un livre d'or pour votre site si vous avez bien suivi

Vous devez indiquer 2 attributs à la balise <form> afin de donner les réponses à ces 2 problèmes :

- method : cet attribut indique par quel moyen les données vont être envoyées (**problème n°1**). Il existe 2 moyens 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 on peut rentrer un grand nombre d'informations grâce à elle.
- action : c'est l'adresse de la page ou du programme qui va *traiter* les informations (**problème n°2**). Cette page se chargera de vous envoyer un 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 XHTML / CSS, on utilisera en général un autre langage dont vous avez peut-être entendu parler : le **PHP**. On aura l'occasion d'y revenir par la suite, ne vous en faites pas 

On va donc maintenant compléter la balise <form> avec les 2 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 spéciale en PHP ("traitement.php"). C'est cette page qui sera appelée lorsque le visiteur cliquera sur le bouton "Envoyer le formulaire".

Code : HTML

```
<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. Nous verrons par la suite comment cette page PHP fait pour analyser les données du formulaire, mais ce n'est pas notre priorité.

Notre priorité là, c'est de voir comment en XHTML / CSS on fait pour insérer des zones de texte, des boutons et des cases à cocher dans votre page web.
C'est ce qu'on va apprendre maintenant 

Les zones de saisie

Bon, retour au concret 

Nous allons voir quelles sont les balises XHTML qui nous permettent de rentrer du texte dans un formulaire.
Pour commencer, il faut savoir qu'il y a 2 zones de texte différentes :

- **La zone de texte à une ligne** : comme son nom l'indique, on ne peut écrire qu'une seule ligne à l'intérieur :p. Elle sert pour rentrer des textes courts, comme par exemple : "Entrez votre pseudo"
- **La zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes, comme par exemple : "Faites une dissertation sur l'utilité du XHTML dans le développement des pays d'Asie du Sud-Est"

Zone de texte à une ligne

Vous ne savez pas ce qu'est une zone de texte ?
Ca tombe bien, j'en ai pris une en photo :

Votre pseudo :

Pour insérer une zone de texte à une ligne, on va utiliser la balise <input />.
On la retrouvera plusieurs fois par la suite dans ce chapitre. A chaque fois, c'est son attribut type qui va changer.

Pour une zone de texte à une ligne, on doit taper :
```
<input type="text" />
```

Mais ce n'est pas tout ! Il manque un attribut qui sera très important : c'est le nom de votre zone de texte. En effet, cela vous permettra plus tard (dans le langage PHP) de reconnaître que tel texte est le pseudo du visiteur, tel texte est son mot de passe etc...
Il faut donc donner un nom à cette zone de texte, grâce à l'attribut name. Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :
```
<input type="text" name="pseudo" />
```

Allez, on va tester ça déjà :

Code : HTML

```
<form method="post" action="traitement.php">
   <p><input type="text" name="pseudo" /></p>
</form>
```

Essayer !

Pensez à entourer votre balise <input /> par une balise block comme car sinon votre page web ne sera pas valide (cela vous est expliqué dans l'annexe "Le W3C et les standards du web").

Les labels

Cette zone de texte est bien jolie, mais si votre visiteur tombe dessus il ne sait pas trop ce qu'il doit mettre dedans. On va justement lui expliquer qu'il doit rentrer son pseudo.
Pour indiquer ce que signifie une zone de texte au visiteur, on utilise la balise <label> qui entoure le libellé, comme ceci :

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       <label>Votre pseudo</label> : <input type="text" name="pseudo" />
   </p>
</form>
```

Mais ça ne suffit pas. Il faut lier le label avec la zone de texte.
Pour ce faire, il faut 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 ? Ca ne va pas faire double emploi ça ?

Si, un peu. Personnellement, je donne la même valeur au name et à l'id, ça ne pose pas de problème.
Pour lier le label au champ, il faut lui donner un attribut for qui a la même valeur que l'id du champ...
En un mot comme en cent, ça donne ça :

Code : HTML

```
<form method="post" action="traitement.php">
    <p>
        <label for="pseudo">Votre pseudo</label> : <input type="text" name="
    </p>
</form>
```

Essayer !

Essayez de cliquer sur le texte "Votre pseudo" : vous allez voir que le curseur se place automatiquement dans la zone de texte correspondante.
On a "lié" le label avec sa zone de texte pour qu'on sache à quoi il correspond.

Mais... Ca sert juste à ce que la zone de texte soit sélectionnée si on clique sur "Votre pseudo" ?

Non, justement
Cela permet aussi aux personnes atteintes d'un handicap de pouvoir se repérer plus facilement dans votre formulaire. On n'y pense pas assez souvent, mais parfois les formulaires sont tellement gros et mal construits qu'il est difficile de savoir à quoi se rapporte chaque zone de texte.
Ici, les non-voyants par exemple sauront que la zone de texte correspond au label "Votre pseudo" à coup sûr, et ils vous seront reconnaissants d'avoir pris le temps de rendre votre formulaire plus clair grâce aux labels.

Quelques attributs supplémentaires

Il existe quelques autres attributs sur la balise <input /> qui vous intéresseront sûrement.

Il est ainsi possible, si vous en avez besoin, de donner une valeur par défaut à votre zone de texte.
Pour faire cela, il vous suffit d'ajouter l'attribut *value* à <input /> en indiquant quelle valeur vous voulez mettre au départ. Exemple :
`<input type="text" name="pseudo" value="M@teo21" />`

Autre chose : vous pouvez modifier la largeur de votre zone de texte ainsi que le nombre maximal de caractères que l'on peut mettre dedans.
La largeur se définit avec *size*.
Le nombre maximal de caractères se définit avec *maxlength*.

Dans l'exemple suivant, la zone de texte contient le pseudo "M@teo21" par défaut, elle fait 30 caractères de long mais on ne peut mettre que 10 caractères maximum à l'intérieur :

Code : HTML

```
<form method="post" action="traitement.php">
    <p>
        <label for="pseudo">Votre pseudo :</label>
        <input type="text" name="pseudo" id="pseudo" value="M@teo21" size="3
    </p>
</form>
```

Essayer !

Zone de mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une zone "mot de passe", c'est-à-dire une zone où on ne voit pas à l'écran ce qui est tapé dedans.
La seule chose que vous avez besoin de changer, c'est l'attribut *type* de <input />. Mettez type="*password*", et le tour est joué !

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

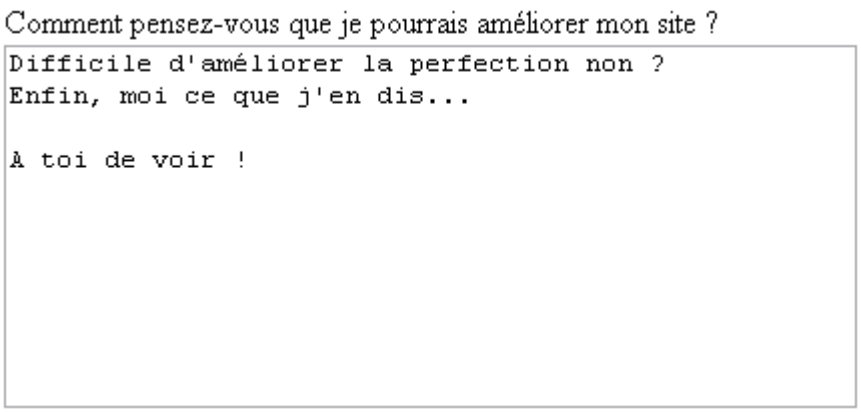
Code : HTML

```
<form method="post" action="traitement.php">
   <p>
        <label for="pseudo">Votre pseudo :</label>
        <input type="text" name="pseudo" id="pseudo" value="M@teo21" />

        <br />
        <label for="pass">Votre mot de passe :</label>
        <input type="password" name="pass" id="pass" />

   </p>
</form>
```

Essayer !

Zone de texte multiligne

On arrive (enfin) aux zones de texte multilignes. Rassurez-vous, ça va aller plus vite maintenant que vous connaissez les labels
Voici une zone de texte multiligne :



La zone de texte multiligne est une balise existant par paire (contrairement à <input />). C'est la balise <textarea></textarea>
A elle aussi, comme à tout autre élément du formulaire, il faut lui donner un nom avec name, et utiliser un label qui explique de quoi il s'agit.

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
        <label for="ameliorer">Comment pensez-vous que je pourrais améliorer
        <textarea name="ameliorer" id="ameliorer"></textarea>
   </p>
</form>
```

Essayer !

On peut modifier la taille du textarea de 2 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 du textarea, et le second le nombre de colonnes.

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       <label for="ameliorer">Comment pensez-vous que je pourrais améliorer
       <textarea name="ameliorer" id="ameliorer" rows="10" cols="50"></text
   </p>
</form>
```

Essayer !

Mais euh, au fait : pourquoi on ouvre <textarea> pour le fermer juste après ? Une balise simple comme <input /> aurait suffit non ?

En fait, cela sert à pré-remplir le textarea. C'est un peu ce qu'on faisait tout à l'heure avec *value*, sauf que là on peut le faire sur plusieurs lignes
Vous pouvez donc mettre la valeur par défaut du textarea comme ceci :

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       <label for="ameliorer">Comment pensez-vous que je puisse améliorer
       <textarea name="ameliorer" id="ameliorer" rows="10" cols="50">
       Améliorer ton site ?!
       Mais enfin ! Il est tellement génialissime qu'il n'y a pas besoin d
       </textarea>

   </p>
</form>
```

Essayer !

Eléments d'options

En plus des zones de saisies, le XHTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire.
On va voir dans cette partie :

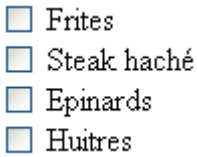
- Les cases à cocher, que vous connaissez sûrement...
- Les zones d'options, que vous connaissez aussi...
- Les listes déroulantes, que vous avez déjà dû voir... Bon en fait vous connaissez déjà tout

Ou plutôt : vous avez déjà vu tous ces éléments, mais je parie que vous ne savez pas comment on les crée en XHTML !
... J'ai gagné ?

Je suis décidément trop fort

Les cases à cocher

Ceci, mes amis, ce sont des cases à cocher :



Bonne nouvelle : ça va aller vite

En effet, la balise à utiliser vous la connaissez déjà : c'est <input />
On va seulement changer la valeur de son attribut *type* pour mettre "checkbox" :
`<input type="checkbox" name="choix" />`

Rajoutez un <label> bien placé, et le tour est (déjà ?) joué !

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       Cochez les aliments que vous aimez manger :<br />
       <input type="checkbox" name="frites" id="frites" /> <label for="frit
       <input type="checkbox" name="steak" id="steak" /> <label for="steak"
       <input type="checkbox" name="epinards" id="epinards" /> <label for="
       <input type="checkbox" name="huitres" id="huitres" /> <label for="hu
   </p>
</form>
```

Essayer !

Qu'est-ce que je peux rajouter ?
Grâce aux label, vous n'êtes pas obligés de cliquer directement sur la case, vous pouvez aussi cliquer sur le texte juste à côté (enfin, ça ne marche pas sur IE comme je vous l'ai dit...).
N'oubliez pas aussi de donner un nom à chaque case à cocher, cela vous permettra d'identifier plus tard quelles cases le visiteur a coché.

Ah si, j'allais oublier. Vous pouvez faire en sorte qu'une case soit déjà cochée par défaut. Pour faire cela, il faut rajouter l'attribut *checked="checked"* (oui oui, même attribut et valeur).
Cela nous permet d'influencer les choix dans notre exemple

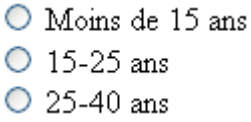
Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       Cochez les aliments que vous aimez manger :<br />
       <input type="checkbox" name="frites" id="frites" /> <label for="frit
       <input type="checkbox" name="steak" id="steak" /> <label for="steak"
       <input type="checkbox" name="epinards" id="epinards" checked="checke
       <input type="checkbox" name="huitres" id="huitres" /> <label for="hu
   </p>
</form>
```

Essayer !

Les zones d'options

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités :



Ca ressemble aux cases à cocher, avec juste une petite difficulté supplémentaire, vous allez voir.

La balise à utiliser est toujours un <input />, avec cette fois la valeur "radio" pour l'attribut type.
La grosse différence avec les cases à cocher, c'est que les zones d'options fonctionnent par "groupe". Tout un groupe d'options a le même nom, mais un attribut *value* différent à chaque fois.

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

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<
       <input type="radio" name="age" value="moins15" id="moins15" /> <labe
       <input type="radio" name="age" value="medium15-25" id="medium15-25"
       <input type="radio" name="age" value="medium25-40" id="medium25-40"
       <input type="radio" name="age" value="plus40" id="plus40" /> <label
   </p>
</form>
```

Essayer !

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache dans quel "groupe" le bouton fait partie.
Essayez d'enlever les attributs *name*, vous verrez que chaque élément d'option deviendra sélectionnable. Or, ce n'est pas ce que l'on veut, c'est pour ça 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 name étant identiques, on n'aurait pas pu les différencier (et vous savez bien qu'un id doit être unique !). Voilà donc pourquoi on a choisi de mettre à l'id la même valeur que value.

Si vous avez 2 zones d'options différentes, il faut donner un nom unique à chaque groupe comme ceci :

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :
       <input type="radio" name="age" value="moins15" id="moins15" /> <lab
       <input type="radio" name="age" value="medium15-25" id="medium15-25"
       <input type="radio" name="age" value="medium25-40" id="medium25-40"
       <input type="radio" name="age" value="plus40" id="plus40" /> <label
   </p>
   <p>
       Sur quel continent habitez-vous ?<br />
       <input type="radio" name="continent" value="europe" id="europe" />
       <input type="radio" name="continent" value="afrique" id="afrique" /
       <input type="radio" name="continent" value="asie" id="asie" /> <lab
       <input type="radio" name="continent" value="amerique" id="amerique"
       <input type="radio" name="continent" value="australie" id="australi
   </p>
</form>
```

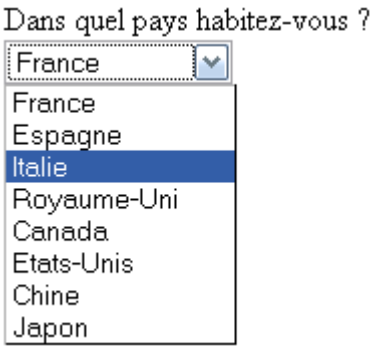
Essayer !

Si, au lieu de mettre *name="continent"* on avait continué à mettre des *name="age"*, le visiteur n'aurait pas pu sélectionner son âge ET son continent.
Essayez de changer les noms, vous verrez bien ce qu'il se passe

Dernière petite précision : si vous voulez qu'une des options soit cochée par défaut, vous rajoutez un *checked="checked"* comme pour les cases à cocher, et le tour est joué !

Les listes déroulantes

Les listes déroulantes sont un autre moyen élégant de faire un choix entre plusieurs possibilités :



Cette fois, ça fonctionne un peu différemment. 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. Par exemple "pays" :
`<select name="pays">`

Et maintenant, à l'intérieur du <select></select>, on va mettre plusieurs balises <option></option> (une par choix possible).
On rajoute un attribut *value* pour pouvoir identifier ce que le visiteur a choisi.

Embrouillé ?
Meuh non, c'est d'une simplicité enfantine !

Code : HTML

```
<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">Etats-Unis</option>
           <option value="chine">Chine</option>
           <option value="japon">Japon</option>
       </select>
   </p>
</form>
```

Essayer !

Le principe est un peu différent de ce qu'on a vu jusqu'ici, mais ça se comprend néanmoins très bien.

Autre nouveauté, on ne peut plus utiliser le *checked="checked"* ici, on doit utiliser à la place le... *selected="selected"*. Il nous permet comme le checked de sélectionner une valeur par défaut :

Code : HTML

```
<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" selected="selected">Canada</option>
            <option value="etats-unis">Etats-Unis</option>
            <option value="chine">Chine</option>
            <option value="japon">Japon</option>
        </select>
    </p>
</form>
```

Essayer !

Mais les listes d'options savent faire encore mieux !
On peut créer des **groupes d'options** à l'intérieur de la liste, grâce à la balise <optgroup></optgroup>. Vous devez lui ajouter l'attribut *label* qui permet de donner un nom au groupe (à ne pas confondre avec la balise <label> !).

Dans notre exemple, pourquoi ne pas séparer les pays en fonction de leur continent ?

Code : HTML

```
<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>
```

Essayer !

C'est assez pratique, surtout quand on a une graaaaaande liste déroulante 

Un formulaire accessible et design ?

Maintenant, on va essayer d'aller encore plus loin.

Notre objectif sera double : faire en sorte que notre formulaire soit *accessible* (= compréhensible) et *design* (= pas trop moche :-°)

On va faire ça en 4 étapes :

1. Définir un ordre de tabulation (*accessibilité*)
2. Définir des touches de raccourci (*accessibilité*)
3. Organiser le formulaire en plusieurs zones (*accessibilité et design*)
4. Rajouter du CSS (*design*)

Définir un ordre de tabulation

C'est le premier des points que nous allons voir censé nous faciliter la vie.
Comme vous le savez peut-être, on peut se déplacer dans un formulaire uniquement grâce à la touche "Tab" (tabulation) située à gauche de votre clavier. A chaque fois qu'on appuie sur Tab, on va au champ suivant. A chaque fois qu'on fait Maj + Tab, on retourne au champ précédent.

Le but est de dire en XHTML dans quel ordre on doit se déplacer dans le formulaire. Par exemple, après le champ "nom" si je tape Tab je dois tomber sur le champ "prénom", puis sur "e-mail" etc...

On va utiliser l'attribut *tabindex* qui peut se rajouter sur toutes les balises du formulaire qu'on a apprises.
On doit lui mettre un nombre pour valeur. Chaque champ du formulaire doit avoir un nombre différent.
Les nombres indiquent dans quel ordre on se déplace dans le formulaire : d'abord le n°1, puis le n°2, le n°3 etc...

Vous n'êtes pas obligés de mettre des nombres qui se suivent. Il est même conseillé de laisser des "espaces" entre les nombres au cas où vous auriez besoin de rajouter plus tard des champs.
Ainsi, il est tout à fait possible de compter 10 par 10 : n°10, n°20, n°30 etc... Ca ne coûte pas plus cher de compter de 10 en 10, et si plus tard on a besoin de créer un champ n°25, on n'aura aucun problème 

Sur ce formulaire, j'ai rajouté les tabindex à chaque champ. Comme on l'a vu, le premier champ est celui qui a le numéro le plus petit, et le dernier celui qui a le plus grand.

Code : HTML

```
<form method="post" action="traitement.php">
   <p>
       <label for="nom">Quel est votre nom ?</label><br />
       <input type="text" name="nom" id="nom" tabindex="10" /><br />

       <label for="prenom">Quel est votre prénom ?</label><br />
       <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

       <label for="email">Quel est votre e-mail ?</label><br />
       <input type="text" name="email" id="email" tabindex="30" /><br />

       <label for="pays">Dans quel pays habitez-vous ?</label><br />
       <select name="pays" id="pays" tabindex="40">
           <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>
```

Essayer !

Essayez de taper plusieurs fois d'affilée sur "Tab", vous allez voir que vous vous déplacerez dans l'ordre que vous avez défini avec *tabindex*.
Cela est particulièrement utile pour les personnes qui ne peuvent pas se servir d'une souris (eh oui, ça existe !).

Par défaut, si aucun *tabindex* n'est mis, le navigateur dira que le premier champ est celui tout en haut, et que le dernier celui tout en bas de la page.
Cependant, je vous conseille de toujours mettre vous-même les tabindex, car si votre formulaire se complexifie par la suite cela sera très utile.

Définir des touches de raccourci

Une touche de raccourci ("access key" en anglais) est une touche qui permet d'accéder directement à un champ de votre formulaire sans avoir à cliquer dessus avec la souris et sans avoir à appuyer plusieurs fois sur "Tab" comme un forcené avant de tomber sur le champ qui vous intéresse.

Ce qui est très pratique, c'est que le XHTML vous permet de choisir quelles touches du clavier serviront de raccourcis.
Ce qui l'est moins, c'est que les raccourcis s'utilisent de manière différente en fonction du navigateur :

- Firefox et Internet Explorer (Windows) : il faut faire la combinaison de touches Alt + Raccourci. Si ça ne marche pas, essayez Alt + Maj + Raccourci.
- Safari et IE-Mac (Macintosh) : il faut taper Ctrl + Raccourci.

Pour définir une touche de raccourci, vous utiliserez l'attribut *accesskey* qui, comme tabindex, peut se mettre sur tous les types de champs de formulaire qu'on a vus.
Vous devez lui mettre comme valeur la touche du clavier qui doit servir de raccourci pour le champ.

Sur cet exemple, le champ de recherche est accessible directement avec la touche R :

Code : HTML

```
<form method="post" action="traitement.php">
    <p>
        <label for="nom">Quel est votre nom ?</label><br />
        <input type="text" name="nom" id="nom" tabindex="10" /><br />

        <label for="prenom">Quel est votre prénom ?</label><br />
        <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

        <label for="email">Quel est votre e-mail ?</label><br />
        <input type="text" name="email" id="email" tabindex="30" /><br />

        <label for="recherche">Que recherchez-vous sur ce site ? <em>(racc
        <input type="text" name="recherche" id="recherche" tabindex="40" ac
    </p>
</form>
```

Essayer !

Sous Windows, il faut donc faire Alt + R pour arriver directement sur le champ de recherche.
Sous Mac, il faut faire Ctrl + R.

Le gros problème des touches de raccourci est que certains caractères sont déjà utilisés par le navigateur. Si vous utilisez les mêmes, il y aura un conflit et vos visiteurs ne pourront plus utiliser les raccourcis auxquels ils sont habitués.
L'idéal est d'utiliser des chiffres en raccourci, ils sont en général très peu utilisés par les navigateurs.

N'oubliez pas d'indiquer quelque part sur la page quels sont les raccourcis utilisables, parce que vos visiteurs ne pourront pas les deviner.
Enfin bon, si, ils peuvent regarder le code source de votre page pour repérer les attributs *accesskey*, mais j'ai déjà connu plus pratique personnellement

Organiser le formulaire en plusieurs zones

La technique que nous allons voir a 2 avantages :

- Elle permet de rendre le formulaire plus clair, donc plus accessible.
- Elle permet d'améliorer le design de votre formulaire.

Concrètement, quelle est l'idée ?
Si vous avez un formulaire assez gros (et en général ça sera le cas), il est facile que le visiteur se perde dans la masse d'informations qu'il a à entrer. Il est possible en XHTML de grouper plusieurs champs ayant un thème entre eux.

On utilisera la balise <fieldset></fieldset> pour délimiter un groupe de champs.
A l'intérieur de cette balise, vous mettrez vos champs (vos <input /> entre autres...) ainsi qu'une autre balise : <legend></legend>. Celle-ci permet de donner le nom du groupe.

Ca ressemble étrangement aux <optgroup> qu'on vient de voir ça... Mais pourquoi tout à l'heure on a utilisé un attribut (label) pour donner un titre au groupe, et là on utilise une balise spéciale <legend> ?

Alors ça, ça fait partie des grands mystères du XHTML. Je n'en sais strictement rien, et pour ma part j'aurais préféré qu'on se mette d'accord : soit on utilise des attributs pour donner un titre, soit on utilise des balises. Là, ça sera à vous de vous en souvenir. Comme d'hab, ça vient avec la pratique... enfin, essayez de le retenir

rapidement quand même, parce qu'on m'annonce que ça pourrait être une question de QCM

Allez, trêve de bavardages, voici un exemple concret d'utilisation des <fieldset> :

Code : HTML

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

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

        <label for="nom">Quel est votre nom ?</label><br />
        <input type="text" name="nom" id="nom" tabindex="10" /><br />

        <label for="prenom">Quel est votre prénom ?</label><br />
        <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

        <label for="email">Quel est votre e-mail ?</label><br />
        <input type="text" name="email" id="email" tabindex="30" /><br />
    </fieldset>

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

        <p>
            Faites un souhait que vous voudriez voir exaucé :<br />
            <input type="radio" name="souhait" value="riche" id="riche" tab
            <input type="radio" name="souhait" value="celebre" id="celebre"
            <input type="radio" name="souhait" value="intelligent" id="inte
            <input type="radio" name="souhait" value="autre" id="autre" tab
        </p>

        <p>
            <label for="precisions">Si "Autre", veuillez préciser :</label>
            <textarea name="precisions" id="precisions" cols="40" rows="4"
        </p>
    </fieldset>
</form>
```

Essayer !

A l'intérieur des <fieldset></fieldset>, l'utilisation de balises de paragraphe n'est plus obligatoire comme c'était le cas tout à l'heure.

Le résultat est de toute évidence plus clair : on voit de suite quelles sont les grandes parties du formulaire. On y gagne forcément en clarté, et le visiteur vous en sera reconnaissant.

Bien sûr, ce formulaire ne serait pas la perfection ultime (hum hum) si on n'y rajoutait pas une petite couche de peinture en CSS

Gouache time !

La bonne nouvelle maintenant, c'est qu'après avoir appris un nombre incalculable de nouvelles balises XHTML, vous n'allez pas apprendre de nouvelles propriétés CSS (pfiou :D)
Comme vous connaissez déjà tout, je vais vous faire l'affront de vous donner un code CSS sans explications préalables. Na !

Code : CSS

```
input, textarea
{
   font-family: "Times New Roman", Times, serif; /* On modifie la police d
}

input:focus, textarea:focus /* Quand le curseur est sur un champ */
{
   background-color: #FFFF99;
}

label
{
   color: blue; /* Colorer en bleu tous les labels (bah oui, pourquoi pas
}

legend /* On met un peu plus en valeur les titres des fieldset */
{
   font-family: Arial, "Arial Black", Georgia, "Times New Roman", Times, s
   color: #FF9933;
   font-weight: bold;
}

fieldset
{
   margin-bottom: 15px; /* Une marge pour séparer les fieldset */
   background-color: #FFFFCC;
}
```

Essayer !

Bon, c'est moche, mais c'est parce que c'est moi qui l'ai fait tout seul

L'idée, c'est surtout de vous montrer que vous en savez désormais assez en CSS pour créer n'importe quel habillage de formulaire. Vous avez tous les outils en main, vous n'avez plus qu'à improviser ! (enfin pas trop quand même)

On a réutilisé pour l'occasion le pseudo-format ":focus" qui, pour ceux qui s'en souviennent encore, permet d'appliquer des styles CSS lorsqu'un objet est sélectionné. En l'occurence, on s'en sert pour mettre un fond jaune aux champs lorsqu'on clique dessus, ce qui donne un effet assez intéressant

---

Y'a plus qu'à appuyer sur le bouton !

Nous y sommes presque.
Nous avons vu la plupart des éléments que l'on peut intégrer dans un formulaire, mais il manque le plus important : le bouton de validation !

Heureusement, c'est très facile à créer, d'autant plus que vous connaissez déjà la balise...
Quoi ? C'est pas encore <input /> dis-moi ?

Si
Décidemment, c'est la balise à tout faire !

Bon, bien sûr un seul type de bouton ne suffisait pas aux webmasters (faut pas rêver non plus), il a fallu en créer 3 :

- Le bouton d'envoi : il déclenche l'envoi du formulaire. Le visiteur se retrouve automatiquement télétransporté à la page indiquée dans l'attribut action du formulaire (on l'a vu au début de ce chapitre). Un bouton d'envoi se crée avec l'attribut *type="submit"*. Vous pouvez lui ajouter un attribut value pour

changer le texte à l'intérieur du bouton, mais vous pouvez laisser la valeur par défaut, c'est aussi clair :
<input type="submit" />

- Le bouton de remise à zéro : il remet à zéro automatiquement toutes les valeurs du formulaire. On doit utiliser cette fois l'attribut *type="reset"*
<input type="reset" />
- Le bouton qui-sert-à-rien : c'est un bouton "générique" qui n'effectue aucune action particulière. Le formulaire n'est pas envoyé, il n'est pas remis à zéro, non rien ne se passe.
Quel intérêt ? Ca vous servira principalement à lancer des scripts en Javascript (un autre langage qu'on peut utiliser sur une page web, oui je sais ça fait beaucoup de langages). Nous, on ne s'en servira pas, mais je vous en parle pour que vous sachiez que ça existe au cas où vous en auriez besoin un jour.
Cette fois, je vous recommande de mettre l'attribut *value* pour que l'on sache à quoi sert le bouton :
<input type="button" value="Je sers à rien" />

On va tester les 2 premiers boutons (envoi et remise à zéro) dans un petit formulaire fictif :

Code : HTML

```
<form method="post" action="cible_formulaire.php">

    <fieldset>
        <legend>Vos coordonnées</legend>

        <label for="nom">Quel est votre nom ?</label><br />
        <input type="text" name="nom" id="nom" tabindex="10" /><br />

        <label for="prenom">Quel est votre prénom ?</label><br />
        <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

        <label for="email">Quel est votre e-mail ?</label><br />
        <input type="text" name="email" id="email" tabindex="30" /><br />
    </fieldset>

    <fieldset>
        <legend>Votre souhait</legend>

        <p>
            Faites un souhait que vous voudriez voir exaucé :<br />
            <input type="radio" name="souhait" value="riche" id="riche" tab
            <input type="radio" name="souhait" value="celebre" id="celebre"
            <input type="radio" name="souhait" value="intelligent" id="inte
            <input type="radio" name="souhait" value="autre" id="autre" tab
        </p>

        <p>
            <label for="precisions">Si "Autre", veuillez préciser :</label>
            <textarea name="precisions" id="precisions" cols="40" rows="4"
        </p>
    </fieldset>


    <p>
        <input type="submit" /> <input type="reset" />
    </p>

</form>
```

Essayer !

Dans le cas présent, le formulaire ne fait strictement rien. Rassurez-vous donc, aucune information n'a été enregistrée

Lorsque vous cliquez sur "Envoyer", le formulaire vous amène donc à une page "cible_formulaire.php", qui est

une page fictive en PHP que vous ne savez pas faire.
Comme je vous l'ai dit, c'est un peu là la limite : on sait construire un formulaire en XHTML / CSS, mais pour traiter les données (les enregistrer ou les envoyer par mail) on est obligés de passer par le langage PHP... dont nous allons parler, pas d'inquiétude

---

Q.C.M.

Quelle balise permet d'indiquer le début et la fin d'un formulaire ?

- ◯ <form></form>
- ◯ <formul></formul>
- ◯ <formulaire></formulaire>

Quelle balise permet de rédiger du texte sur plusieurs lignes ?

- ◯ <input type="text" />
- ◯ <input type="multiline" />
- ◯ <textarea></textarea>

Quel attribut permet de limiter le nombre de caractères que l'on peut écrire dans un champ de texte à une ligne ?

- ◯ size
- ◯ maxchars
- ◯ maxlength

Si je mets l'attribut type="radio" à mon <input />, en quoi va-t-il se transformer ?

- ◯ En option
- ◯ En case à cocher
- ◯ En poste de radio

A quoi sert la balise <legend> ?

- ◯ A donner le titre d'un groupe de champs <fieldset>
- ◯ A donner le titre d'un groupe de valeurs d'une liste déroulante
- ◯ A indiquer à quoi sert un champ

Pourquoi faut-il donner un nom (name) identique à chaque zone d'options, avec une valeur (value) différente pour chacune des options ?

- ◯ Parce que sinon on pourrait sélectionner plusieurs valeurs là où un seul choix est normalement possible.
- ◯ Parce que le XHTML ne supporte pas un trop grand nombre de zones d'options avec des noms différents.
- ◯ Parce qu'on est censés pouvoir sélectionner plusieurs valeurs pour chaque zone d'options.

Quel pseudo-format CSS permet de modifier l'apparence d'un champ lorsqu'on est en train de l'éditer ?

- ○ :hover
- ○ :selected
- ○ :focus

Lequel de ces boutons permet d'envoyer le formulaire ?

- ○ <input type="submit" />
- ○ <input type="send" />
- ○ <input type="reset" />

Lequel de ces langages permet d'analyser et traiter les informations envoyées par le visiteur grâce au formulaire ?

- ○ XHTML
- ○ PHP
- ○ CSS

Correction !

Statistiques de réponses au QCM

Ladies and gentlemen, j'ai l'honneur de vous annoncer que vous venez de lire un chapitre entier sur les formulaires et que vous ne savez toujours pas vous en servir.

Quoi, pourquoi vous me regardez comme ça ?... 
... NooOOooOOoonnnn, pas les tomaaaaates !!! 

Avant de me lyncher sur la place publique, écoutez ce que j'ai à vous dire : vous avez ter-mi-né d'apprendre le XHTML et le CSS ! 
Oui oui, vous avez bien entendu !

Mais, comme j'ai (malheureusement) une conscience, je m'en voudrais de vous lâcher comme ça dans la nature. Je vais donc me farcir l'écriture d'un chapitre supplémentaire, qui sera un chapitre de **conclusion** 

Comme dans toute dissertation qui se respecte :

- Nous ferons le bilan de ce que nous avons appris (et de ce que nous n'avons pas appris)
- Et l'*ouverture* sur ce mystérieux langage "PHP" qui est, paraît-il, censé pouvoir traiter les informations du formulaire (et bien d'autres choses encore !)