Formulaires

Les formulaires sont la base de l'interactivité d'un site web.
Le HTML vous permet de créer un formulaire, mais en ce qui concerne l'envoi, c'est en général un script d'un langage de programmation type PHP qui permettra de le traiter (si le formulaire est correctement rempli). En conséquence, l'envoi et le traitement d'un formulaire sont l'affaire du développeur et non de l'intégrateur.
L'intégrateur, dans tous les cas, devra savoir préparer un formulaire, c'est-à-dire le coder correctement et le mettre en forme en CSS. C'est ce que nous allons voir dans cette partie.

I. Formulaire et méthode d'envoi

Un formulaire se définit par une balise de type **bloc**, tandis que les objets de formulaire qui le composent (champ texte, menu déroulant, etc.) sont de type **inline**.

A. La balise form

La balise utilisée pour délimiter un formulaire est la balise **<form>**. Tout ce qui se trouve dans cette balise fera partie du formulaire. Les formulaires étant généralement destinés à être interactifs, ils doivent être traités. Pour ce faire, il faut que les données puissent être envoyées, soit par e-mail, soit à une page web dynamique qui sera chargée de les traiter. **Cette page ne pourra être en HTML seul, car le HTML ne permet pas de récupérer des informations sur les formulaires.**

L'attribut qui permet de spécifier l'action qui sera faite en cas de clic sur le bouton « Envoyer » (par défaut) est l'attribut action qui prend pour valeur soit une adresse e-mail précédée de l'attribut « mailto: » (cette méthode n'est quasiment plus utilisée) ou bien l'adresse d'une page web chargée de traiter le formulaire.

Voyons le cas où une page web (on la supposera en PHP) traite les données du formulaire :

```
<form action="url-de-la-page.php">
</form>
```

Fig. 1 Balise «form» © Fotolia

Ensuite, il va falloir indiquer à notre formulaire quelle méthode adopter pour l'envoi des données à la destination. Il existe deux méthodes :

La méthode GET, qui fait passer toutes les données par l'URL (adresse) de la page cible. Vous vous retrouverez donc avec une adresse de ce style : **page.php?nom=dupont&prenom=pierre**. Cette méthode a plusieurs inconvénients : outre le fait qu'un visiteur peut très bien mettre n'importe quoi dans la barre d'adresse, cette méthode est également limitée en nombre de caractères. La limite varie en fonction du navigateur, elle peut aller de 255 caractères à 65 536 en général. Bref, il ne faut pas utiliser cette méthode pour des raisons de sécurité.

La méthode POST, bien plus adaptée, va tout envoyer à la page web dans les en-têtes HTTP que renvoie votre navigateur. Vous n'aurez pas à vous en préoccuper. Cette méthode est plus sécurisée que la méthode GET contre les pirates, mais n'espérez pas vous dispenser de vérifications côté serveur, qui doivent **toujours** être faites. N'ayez jamais confiance en une donnée provenant du visiteur.

L'attribut permettant de spécifier la méthode d'envoi est **method**. Il peut prendre deux valeurs : **get** ou **post**.

Voici le code de notre formulaire de tout à l'heure que nous choisirons d'envoyer en méthode post :

```
<form action="url-de-la-page.php" method="post">
</form>
```

Fig. 2 Balise «form» avec la méthode «post» © Fotolia

Il existe d'autres attributs : **name** (définit le nom du formulaire), **id** (définit l'identifiant du formulaire), etc.

Enfin, la balise <form> ne peut pas contenir d'autres balises <form>. Mais vous pouvez avoir plusieurs formulaires dans une même page web.

B. Les balises fieldset, legend et label

Les balises **<fieldset>**, **<legend>** et **<label>** ont été introduites avec le HTML 4.0 pour des raisons d'accessibilité, dans l'optique d'améliorer l'accès au Web par les malvoyants, les handicapés. Si vous voulez que votre formulaire soit validé W3C, il est conseillé de les utiliser, mais rien n'est obligatoire en la matière votre formulaire fonctionnera tout de même.

La balise **<fieldset>** définit un regroupement dans un formulaire d'une page web HTML (balise de type **bloc**). Cette balise permet de regrouper des champs et informations de même thème ou de même nature.

La balise **<legend>** définit la légende du regroupement dans un formulaire d'une page web HTML (balise de type **inline**). La balise **<legend>** est obligatoirement associée à la balise **<fieldset>** qui la précède.

```
<form action="url-de-la-page.php" method="post">
    <fieldset>
        <legend>Informations sur vous</legend>
        Nom : <input type="text" name="monNom" >
        Prénom : <input type="text" name="monPrenom">
    </fieldset>
    <fieldset>
        <legend>Informations sur votre conjoint</legend>
        Nom : <input type="text" name="conjointNom">
        Prénom : <input type="text" name="conjointPrenom">
    </fieldset>
</form>
```

Fig.3 Fichier HTML © Fotolia

Résultat dans un navigateur :

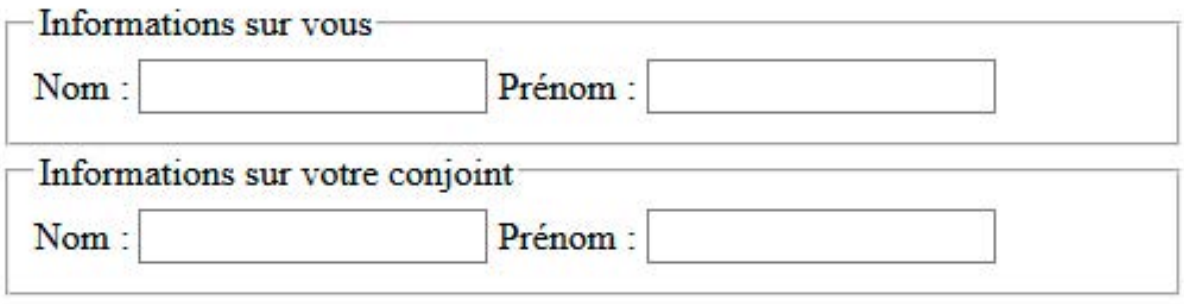

Fig.4 Résultat dans un navigateur © Fotolia

La balise **<label>** définit une étiquette d'un champ de formulaire d'une page web HTML (balise de type **inline**). La balise **<label>** possède un attribut **for** qui fait référence à l'attribut **id** d'un champ de formulaire. Cela permet ainsi d'associer la balise **<label>** avec le champ auquel elle correspond.

```
<form action="url-de-la-page.php" method="post">
    <fieldset>
        <legend>Informations sur vous</legend>
        <label for="nom">Nom :</label>
        <input type="text" name="monNom"  id="nom">
        <label for="prenom">Prénom :</label>
        <input type="text" name="monPrenom" id="prenom">
    </fieldset>
</form>
```

Fig. 5 Balise «label» © Fotolia

II. Les objets de formulaires

A. La balise input

1. Les types de champs

La plupart des objets que nous allons voir utilisent la balise **<input>**. Les attributs que va prendre cette balise vont déterminer le type de notre champ ainsi que d'autres options suivant la nature de ces champs. On appelle un champ une « zone » interactive du formulaire. L'attribut qui définit le type d'objet de formulaire est **type**, dont la syntaxe est la suivante :

```
<input type="valeur-de-l-attribut">
```

Fig. 6 Exemple type © Fotolia

Il en existe plus d'une vingtaine, dont une douzaine sont apparus avec la version 5 d'HTML, nous ne les passerons donc pas tous en revue de manière exhaustive, sachant qu'il est tout à fait possible d'en voir apparaître de nouveau. Commençons par un exemple.

2. Exemple d'un champ texte à une ligne

Pour un champ texte à une ligne, on indique la valeur **text** à **type**.

```
<form action="url-de-la-page.php" method="post" name="formulaire_01">
    <input type="text" >
</form>
```

Fig. 7 Type de valeur text © Fotolia

Cela n'est pas suffisant. Pour récupérer les données entrées par l'internaute, il faut bien pouvoir identifier vos différents champs, et donc leur donner des noms différents. L'attribut permettant de donner un nom à un champ est **name** et il prend la valeur du nom que vous souhaitez donner à votre attribut. Pour nommer le champ texte **prenom**, on aura ceci : **name=« prenom »**.

Caractères spéciaux

Attention : les noms d'attribut ne doivent pas comporter d'espace ni de caractères spéciaux (accents, @, etc.) et il est préférable d'utiliser les minuscules.

Quelques autres attributs :

L'attribut **maxlength** : il va vous permettre de spécifier le nombre de caractères maximum que peut rentrer l'internaute. Si l'entrée du texte n'est pas contrôlée, vous pouvez tomber sur des petits malins qui peuvent insérer tout un roman dans votre champ texte !

L'attribut **placeholder** : il va vous permettre de spécifier une valeur par défaut à votre champ, cette valeur sera affichée dans le champ texte et le visiteur pourra la modifier bien entendu.

L'attribut **required** : rend la saisie du champ obligatoire.

Voilà maintenant un exemple combinant toutes les propriétés que nous venons d'énoncer. Il s'agit d'un champ texte intitulé « prenom ». Il pourra contenir 25 caractères maximum, possédera la valeur « Anthony » par défaut et sera obligatoire :

```
<form action="url-de-la-page.php" method="post" name="formulaire_01">
    <input type="text" name="prenom" placeholder="Anthony" maxlength="25"
    required>
</form>
```

Fig. 8 Exemple de champ texte © Fotolia

Vous obtenez ceci :

Fig. 9 Affichage sur un navigateur © Fotolia

3. Liste des types de champs

Tableau n°1 Types de champs

VALEUR	DESCRIPTION
button	Bouton cliquable (en général avec JavaScript)
checkbox	Cases à cocher
color	Choix d'une couleur
date	Date avec attributs (année, mois et jour), l'ordre dépend de la langue
datetime-local	Indique une date et une heure (année, mois, jour, minute, seconde, et fraction de seconde)
email	Une adresse e-mail
file	Pour sélectionner un fichier dans le but de l'uploader
hidden	Champ caché pour l'utilisateur (permet d'envoyer des données invisibles pour l'utilisateur mais dont on pourrait avoir besoin)
image	Pour personnaliser le bouton submit (envoyer) à l'aide d'une image
month	Définit un mois et une année
number	Pour indiquer un nombre
password	Champ de mot de passe (les caractères sont masqués et sont remplacés par des étoiles ou des points)
radio	Bouton radio
range	Un curseur glissant permet d'indiquer une valeur (par défaut, entre 0 et 100, mais paramétrable)

VALEUR	DESCRIPTION
reset	Bouton reset (réinitialise toutes les valeurs des champs du formulaire)
search	Champ de recherche
submit	Bouton submit (ou bouton « envoyer ») afin d'envoyer le formulaire
tel	Numéro de téléphone
text	Champ de texte comportant une seule ligne
time	Pour indiquer une heure
url	Une URL
week	Définit une semaine d'une année 2017-w52 = la 52e semaine de l'année 2017

Nous réaliserons plus loin un exemple de formulaire complet avec les types de champ input les plus courants.

En cas de doute, n'hésitez pas à consulter directement la page suivante :
https://www.w3schools.com/tags/att_input_type.asp

4. Liste des principaux attributs

Nous n'allons pas passer en revue tous les attributs de la balise **<input>** mais les principaux, certains pourront être utilisés avec d'autres balises de formulaire.

Tableau n°2 Principaux attributs

VALEUR	DESCRIPTION
autofocus	place le focus sur un champ spécifique
placeholder	suggestion de saisie apparaissant uniquement lorsque le champ est vide et non sélectionné
required	saisie obligatoire
maxlength	nombre maximal de caractères
min	valeur minimale
max	valeur maximale
step	valeur du pas d'incrément (exemple 2 par 2)
title	bulle d'aide et message d'erreur personnalisé
pattern	format défini par une expression régulière, exemple pattern=»<1-9>{4}», pour être valide doit comporter 4 caractère entre 1 et 9Plus d'infos sur le lien suivant https://www.w3schools.com/tags/att_input_pattern.asp

Nous verrons plus tard comment visualiser la vérification automatique de la validité des champs **<input>** avant validation du formulaire.

B. Zone de texte multiligne

La zone de texte multiligne va être utile dès que vous voudrez taper un long message, comme pour poster un message sur le forum par exemple.

```
<form action="url-de-la-page.php" method="post" name="formulaire_01">
    <textarea>Tapez ici votre message</textarea>
</form>
```

Fig. 10 Exemple de texte multiligne © Fotolia

Nous distinguerons la taille la zone **textarea** que nous piloterons à l'aide des règles CSS **width** (largeur) et **height** (hauteur) des attributs **cols** et **rows** qui indiquent respectivement le nombre de caractères pouvant être placés sur une ligne sans afficher l'ascenseur horizontal et le nombre de lignes pouvant être placées dans la **textarea** sans afficher l'ascenseur vertical.

Voici ce que cela donne pour créer une **textarea** portant le nom de « message », contenant la valeur par défaut « Tapez ici votre message » et qui permet de taper 3 lignes et 50 caractères par ligne, sans afficher les ascenseurs :

```
<form action="page.php" method="post" name="formulaire_01">
    <p>
        <label>Votre message :
            <textarea name="message" rows="3" cols="50">Tapez ici votre
            message</textarea>
        </label>
    </p>
</form>
```

Fig. 11 Exemple de zone de texte multiligne © Skill and You

Le rendu :

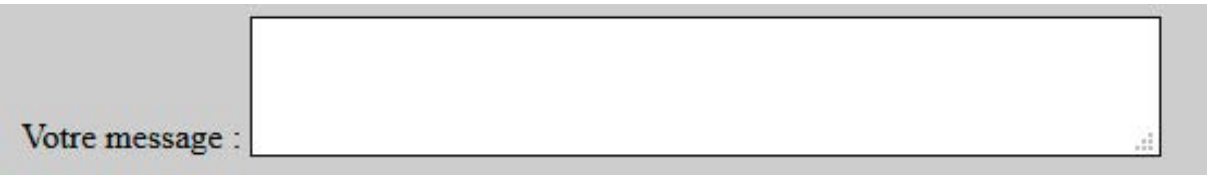

Fig. 12 Affichage sur un navigateur © Fotolia

Si vous ne souhaitez pas placer de valeur par défaut dans la zone de texte multiligne, les deux balises ouvrant et fermant la **textarea** pourront être accolées :

```
<form action="page.php" method="post" name="formulaire_01">
    <p>
        <label>Votre message : </label>
        <textarea name="message" rows="3" cols="50"></textarea>
    </p>
</form>
```

Fig. 13 Zone de texte sans valeur par défaut © Fotolia

Vérifier l'orthographe avec spellcheck

Un attribut bien pratique permet d'activer le vérificateur orthographique de votre navigateur, il s'agit de **spellcheck**.

```
<form action="page.php" method="post" name="formulaire_01">
    <p>
        <label>Votre message : </label selected="selected" >
        <textarea name="message" rows="3" cols="50"  spellcheck="true"></textarea>
    </p>
</form>
```

Fig. 14 Utilisation de spellcheck © Skill and You

En rendu, les mots mal orthographiés sont soulignés en rouge :

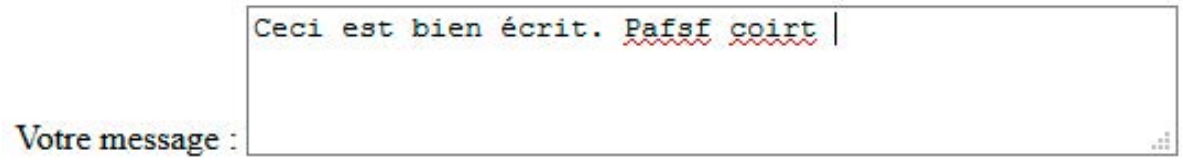

Fig. 15 Affichage sur un navigateur © Fotolia

C. Les listes déroulantes select

Les listes déroulantes sont utiles quand vous avez beaucoup de choix disponibles et une faible place par exemple. On utilise alors la balise **select**.

Les différents choix dans la liste déroulante sont définis par la balise **option + value**.

Enfin, si vous souhaitez utiliser un choix par défaut, on ajoute simplement **selected**.

Au final :

```
<form action="page.php" method="post" name="formulaire_01">
    <label>Votre couleur préférée : </label>
    <select name="couleur">
        <option value="rouge">Rouge</option>
        <option value="vert">Vert</option>
        <option value="bleu" selected>Bleu</option>
    </select>
</form>
```

Fig. 16 Exemple de liste déroulante © Fotolia

Ici, on demande quelle est la couleur préférée du visiteur. Par défaut, la couleur sélectionnée est le bleu. Voici ce que ça donne à l'écran :

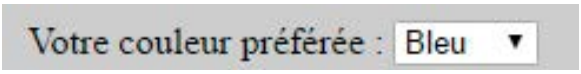

Fig. 17 Affichage sur un navigateur © Fotolia

Et si vous souhaitez que l'utilisateur puisse sélectionner plusieurs couleurs avec la touche Ctrl enfoncée, il va falloir utiliser l'attribut **multiple**. Voici un exemple :

```
<form action="page.php" method="post" name="formulaire_01">
    <label>Votre (ou vos) couleur(s) préférée(s) : </label>
    <select name="couleur" multiple="multiple">
        <option value="rouge">Rouge</option>
        <option value="vert">Vert</option>
        <option value="bleu" selected>Bleu</option>
    </select>
</form>
```

Fig. 18 Liste déroulante à sélection multiple © Skill and You

Le rendu :

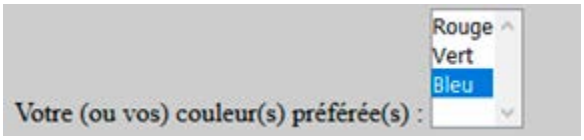

Fig. 19 Affichage sur un navigateur © Fotolia

D. La balise output

Champ dynamiquement mis à jour en fonction des entrées dans le formulaire.

***Exemple** : pour deux valeurs, on va obtenir le résultat d'une multiplication qui apparaît au remplissage des champs de valeur :*

```
<form action="scripts.php" method="post" id="total"
oninput="t.value = a.value * b.value ">
    <p>
        <label for="a">valeur a</label>
        <input type="number" name="a" id="a">
    </p>
    <p>
        <label for="b">valeur b</label>
        <input type="number" name="b" id="b">
    </p>
    <p>
        Résultat :
        <output for="t" name="t" form="total"></output>
    </p>
</form>
```

Fig. 20 Exemple de balise output © Fotolia

L'attribut **oninput** de la balise **form** donne la formule, ici la valeur de test égale à la valeur de **a** multipliée par la valeur de **b**. On a donc besoin de deux balises **input** de type **number** pour indiquer les valeurs de **a** et **b**.

La balise **output** renvoie le résultat, pour cela il faut indiquer dans les attributs de cette balise **for**=le nom de la variable (ici « t ») et **form**=identifiant de la balise form (ici « total »).

Au chargement de la page :

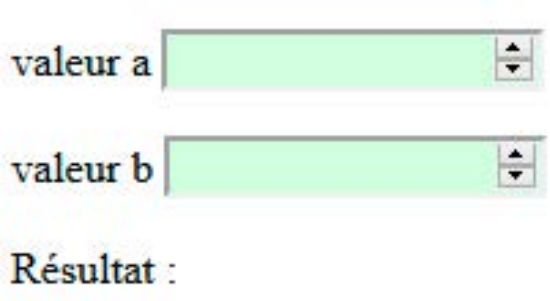

Fig. 21 Affichage sur un navigateur avant remplissage © Fotolia

Après remplissage des champs :

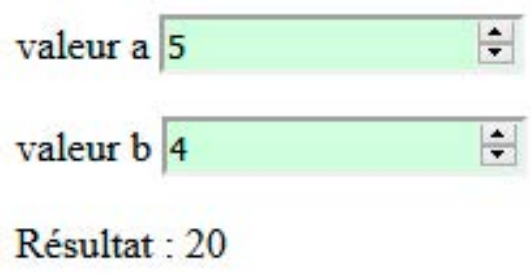

Fig. 22 Affichage sur un navigateur après remplissage des champs © Fotolia

Deuxième exemple : on peut également ajouter un bouton qui, activé, va renvoyer le résultat.

```
<form action="page.php" method="post" id="total-submit"
onsubmit="t.value = a.value * b.value; return false ;">
    <p>
        <label for="a">valeur a</label>
        <input type="number" name="a" id="a">
    </p>
    <p>
        <label for="b">valeur b</label>
        <input type="number" name="b" id="b">
    </p>
    <p>
        Résultat :
        <output for="t" name="t" form="total-submit"></output>
    </p>
    <input type="submit" value="Calculer">
</form>
```

Fig. 23 Ajout d'un bouton © Fotolia

On substitue **onsubmit** à **oninput** dans les attributs de la balise form (afin de lier le calcul au clic sur le bouton **submit**). On ajoute également « **return false** » afin de rester sur la même page web.

Au chargement de la page :

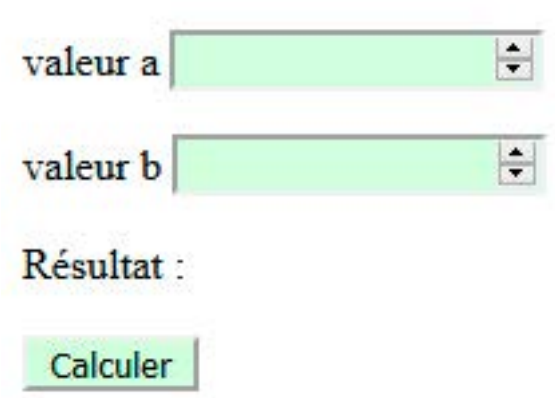

Fig. 24 Affichage sur un navigateur avant remplissage © Fotolia

Après remplissage des champs :

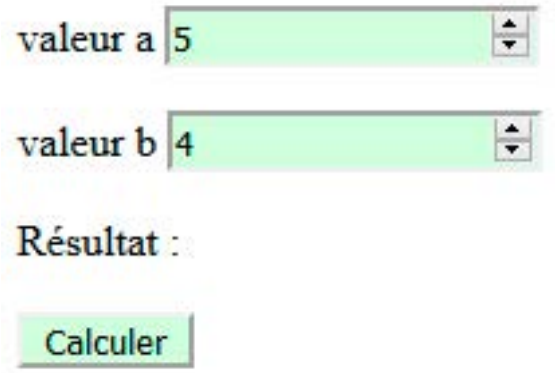

Fig. 25 Affichage sur un navigateur après remplissage © Fotolia

Après avoir cliqué sur « calculer » :

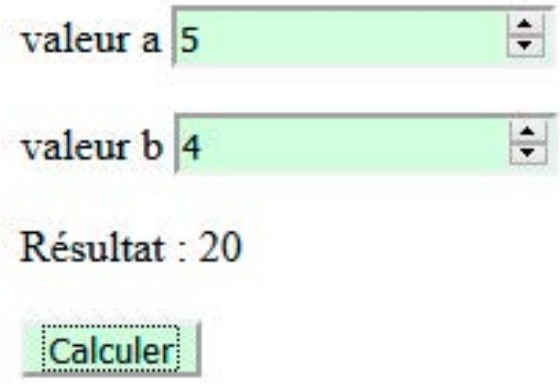

Fig. 26 Affichage sur un navigateur après avoir cliqué sur le bouton © Fotolia

E. La balise datalist

Permet d'ajouter une liste de suggestions à un champ de texte lors de la saisie ou au clic souris.

```
<form>
    <input list="cars" placeholder="indiquez votre voiture" >
    <datalist id="cars">
        <option value="BMW">BMW</option>
        <option value="Mercedes">Mercedes</option>
        <option value="traban">Traban</option>
    </datalist>
</form>
```

Fig. 27 Balise datalist © Fotolia

Le rendu :

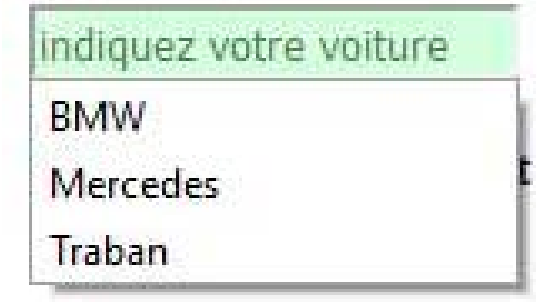

Fig. 28 Affichage sur un navigateur © Fotolia

F. Les barres de progression : progress et meter

Ces balises confortent la position de HTML5 comme langage applicatif et non plus seulement comme langage descriptif et permettent de donner l'état d'avancement d'une tâche par exemple ou d'avoir un aperçu d'un nombre en rapport avec un autre.

<progress>

value : attribut désignant la valeur actuelle de l'état d'avancement.

max : attribut indiquant la valeur maximale à atteindre.

<meter>

value : attribut désignant la valeur actuelle de l'élément.

max : attribut indiquant la valeur maximale.

min : attribut indiquant la valeur minimale.

Exemple

```
<p> Progress => état d'avancement :</p>
<progress id="devoirs" max="100" value="30" >30 %</progress>

<p>Meter => les inscrits à l'atelier :</p>
<meter id="inscrits" min="0" max="20" value="3">5 places</meter>
```

Fig. 29 Ajout d'une barre de progression © Fotolia

Donne :

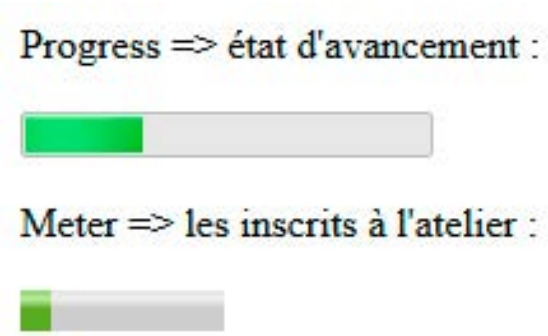

Fig. 30 Affichage sur un navigateur © Fotolia

III. Formulaire et CSS

A. Formulaire et CSS

Un formulaire peut complètement changer d'aspect avec des règles CSS. Nous pouvons par exemple donner des dimensions à nos objets de formulaire, leur donner une couleur de fond, une bordure, donner une taille et une couleur au texte entré…

Pseudo-classes CSS et formulaires

Certaines pseudo-classes permettent de styliser un champ de formulaire suivant qu'il soit optionnel, requis ou invalide.

: optional : style un champ optionnel ;

: required : style un champ obligatoire ;

: invalid : style pris lorsque le champ devient invalide.

Par exemple :

```
input:valid {
    background-color: #ddffdd;
}
input:invalid {

    background-color: #f00;
}
input:required {
    background-color: #df0;
}
```

Fig. 31 Exemple d'utilisation des pseudo-classes © Skill and You

Soit si l'on regarde chaque couleur :

Fig. 32 Couleurs sélectionnées pour chaque pseudo-classe © Skill and You

Un champ obligatoire :

Fig. 33 Affichage sur un navigateur © Skill and You

Une écriture invalide :

Fig.34 Affichage sur un navigateur © Skill and You

Une écriture valide :

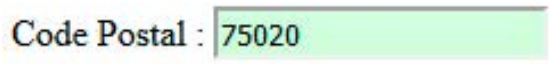

Fig.35 Affichage sur un navigateur © Skill and You

Voici un exemple de code HTML que l'on va styliser :

```
<form action="page.php" method="post" class="formulaire_01"  name="formulaire_01"
enctype="multipart/form-data">
    <input name="date" type="hidden" value="" >
    <fieldset>
        <p>
            <label for="nom"  >Nom :</label>
            <input type="text" name="nom"  id="nom" placeholder="écrivez votre nom"
            autocomplete="on" autofocus="true" required>

        </p>
        <p>
            <label for="inscription_newsletter">S'inscrire à la newsletter ?</label>
            <input name="inscription_newsletter" type="checkbox" id="
            inscription_newsletter">
        </p>

        <p>
            <label for="couleur">Quellle est votre couleur préférée ?</label>
            <select name="couleur" id="couleur" placeholder="votre couleur préférée
            email" >
                <option  value="blanc">Blanc</option>
                <option  value="noir" >Noir</option>
                <option  value="rouge">rouge</option>
                <option  value="bleu" >bleu</option>
            </select>
        </p>
        <p>
            <label for="pub">Voulez vous re cevoir notre pub:</label>
            Oui<input name="pub" type="radio" value="oui">
            Non<input name="pub" type="radio" value="non">
        </p>
        <p>
            <label for="email">Email</label>
            <input type="email" name="email"  id="fichier"  placeholder="votre email"
            autocomplete="on">
        </p>
        <p>
            <label for="message">Votre message : </label>
            <textarea name="message" rows="3" cols="30" id="message"></textarea>
        </p>
        <p id="envoyer">
            <input name="envoyer" type="submit" value="Envoyer">
        </p>
    </fieldset>
</form>
```

Fig.36 Exemple de fichier HTML © Skill and You

Résultat dans un navigateur :

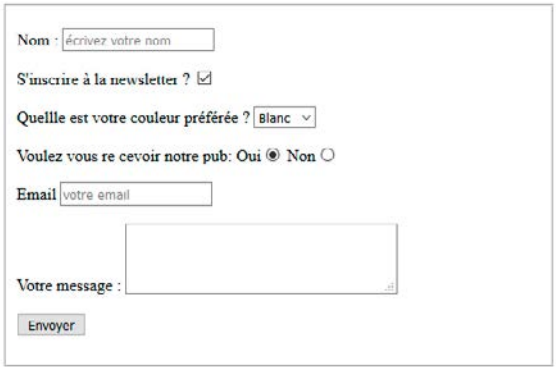

Fig. 37 Affichage sur un navigateur © Skill and You

Ce n'est déjà pas si mal, mais modifions :

1) La dimension, la position du formulaire, la couleur de la bordure et du fond

Fichier CSS

```css
/* reset */

* {
    margin: 0;
    padding: 0;
    box-sizing:border-box;
}
/* fin reset */

.formulaire_01 {
    width: 60%;
    max-width: 1240px;
    margin: 50px auto;

}
.formulaire_01 fieldset {
    border:2px dotted #F90;
    background-color: #D6D6D6;
}
```

Fig. 38 Fichier CSS © Skill and You

Commentaires : Les règles sur **formulaire_01** sont là essentiellement pour rendre celui-ci plus harmonieux dans une page vide.

Résultat dans un navigateur

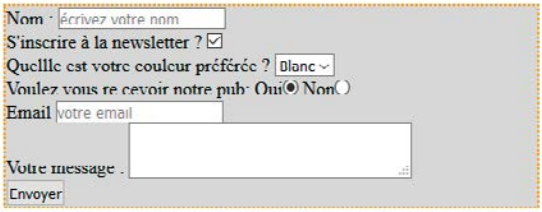

Fig. 39 Affichage sur un navigateur © Skill and You

2) Occupons-nous maintenant des textes et d'aérer les éléments

Fichier CSS

```css
.formulaire_01 fieldset {
    border:2px dotted #F90;
    background-color: #D6D6D6;
    padding: 15px;
    font: 1em arial, sans-serif ;
}

.formulaire_01 label {
    font-weight:bold
}
.formulaire_01 select, .formulaire_01 input {
    color:#983400;
    font-weight:bold
}
.formulaire_01 p {
    font: 1em arial, sans-serif ;
    padding: 5px;
    color: #4d4858;

}
```

Fig. 40 Fichier CSS © Skill and You

Résultat dans un navigateur

Fig. 41 Affichage sur un navigateur © Skill and You

Enfin occupons-nous du bouton **envoyer**, pour ceci on va ajouter des classes dans le HTML.

Fichier HTML

```
<p id="envoyer" class="right" >
    <input class="bt" name="envoyer" type="submit"
     value="Envoyer">
</p>
```

Fig. 42 Fichier HTML © Skill and You

Fichier CSS

```
.right {
    text-align:right
}
.bt {
    border:#F00 thin solid;
    width:120px;
    height:35px;
    cursor:pointer
}
```

Fig. 43 Fichier CSS © Skill and You

Règle Cursor

La nouveauté dans ce cas est la règle cursor : pointer qui permet d'avoir un pointeur de souris différent de la flèche par défaut.

Résultat dans un navigateur

Fig. 44 Affichage sur un navigateur © Skill and You

On peut également styliser la validité ou non de l'adresse e-mail ainsi que les champs requis :

Fichier CSS

```
.formulaire_01 input:valid {
    background-color: #ddffdd;
}
.formulaire_01 input:invalid {
    background-color: #f00;
}
.formulaire_01 input:required {
    background-color: #f1c4c4;
}
```

Fig. 45 Fichier CSS © Skill and You

On est également obligé d'augmenter le poids du ciblage pour le bouton **envoyer** en utilisant l'identifiant du champ qui est « envoyer » :

```
#envoyer .bt {
    border:#F00 thin solid;
    width:120px;
    height:35px;
    cursor:pointer;
    background-color: #f1c4c4;
}
#envoyer .bt:hover {
    background-color: #fff;
    color:#222;
}
```

Fig. 46 Fichier CSS © Skill and You

Résultat dans un navigateur

Fig. 47 Affichage sur un navigateur © Skill and You

On peut également, à défaut de travailler plus avant le positionnement des éléments ce que vous ferez par vous-mêmes dans les exercices associés, au moins aérer quelque peu l'ensemble. On ajoute une petite marge sur **input** et **select**.

```
.formulaire_01 select, .formulaire_01 input {
    color:#983400;
    font-weight:bold;
    margin: 3px;
}
```

Fig. 48 Fichier CSS © Skill and You

Le rendu :

Fig. 49 Affichage sur un navigateur © Skill and You

« Votre message » se trouve au pied de la zone de texte correspondante. Heureusement, **label** étant un élément **inline**, il est soumis à la règle **vertical-align**, on peut dès lors lui demander de se placer en haut.

Notre CSS devient :

```
.formulaire_01 label {
    font-weight:bold;
    margin: 4px;
    vertical-align: top;
}
```

Fig. 50 Fichier CSS © Skill and You

Le rendu :

Fig. 51 Affichage sur un navigateur © Skill and You

Les formulaires sont des éléments incontournables sur site web, ils permettent non seulement de recueillir des données sur les internautes, mais aussi d'interagir avec eux. Grâce à cette leçon, vous êtes désormais capables d'élaborer un formulaire.