HTML : Les formulaires

INTRODUCTION

Les formulaires permettent à l'internaute d'effectuer une saisie et d'envoyer les informations saisies vers le serveur web afin d'y être traitées (grâce à un script CGI, ASP.Net ou PHP).

Les saisies dans un formulaire peuvent s'effectuer par :

- Des champs de saisie de texte,
- zones de texte multilignes,
- listes de choix (*combobox ou listes déroulantes*),
- des cases à cocher (checkbox) pour des choix multiples de réponses,
- des boutons radio, pour un choix unique de réponse (Exemples : Oui/Non, Monsieur/Madame),
- il est aussi possible de télécharger un fichier

HTML 5 a ajouté de nouveaux types de champs :

- datalist
- output

DECLARER UN FORMULAIRE

Un formulaire s'ouvre avec la balise `<form>` et se termine par `</form>`.

Entre ces 2 balises, on placera tout le code permettant de saisir des données (champs textes, listes déroulantes, cases à cocher…) et, à ne pas oublier, au moins un bouton d'envoi (soumission du formulaire), lequel permettra de soumettre le formulaire.

La balise `<form>` possède obligatoirement 2 attributs : `action` et `method`. D'autres attributs plus traditionnels peuvent être ajoutés de façon facultative : `name`, `id`…

Exemple

```
<form action="destination.php" method="post">
   <input type="text" name="nom">
   <input type="submit" value="envoyer">
</form>
```

Explications :

Attribut	Utilité
action	Indique le nom/url du fichier qui sera exécuté lors de l'envoi du formulaire (au clic sur le bouton de soumission. Ce fichier de destination recevra et traitera les données : contrôles des informations saisies, enregistrement en base de données… Ce fichier sera côté serveur (PHP, ASP/.Net, Ruby…) l'envoi du formulaire.
method	L'attribut `method` peut recevoir 2 valeurs possibles : ▪ `get` : les données saisies sont ajoutées à l'url. ⚠ L'utilisation de cette valeur est à éviter. ▪ `post` : les données saisies dans le formulaire sont envoyées avec l'entête http. Elles n'apparaissent pas dans l'url. ***⚠ L'utilisation de cette valeur est recommandée.***

Exercice

Pour tester les deux méthodes `get` et `post`, reproduisez l'exemple ci-dessus puis :

- Modifier l'attribut `action` pour la valeur `http://bienvu.net/script.php`, saisir des données et envoyez le formulaire. Observez ce qui est affiché : on voit des paires clés/valeurs correspondant respectivement aux valeurs des attributs `name` (ici : « nom » et « prenom » mais aussi le bouton d'envoi) et les valeurs saisies.
- Puis modifier l'attribut `method` pour la valeur `get`, saisir des données et envoyez le formulaire. Observez alors cette fois qui se passe dans la barre d'adresse du navigateur : on voit les paires clés/valeurs séparées par le caractère `&`.

On peut en déduire que les données sont transmises sous forme d'un tableau qui pourra être manipulé par un langage de script côté serveur (ici, le fichier de destination est écrit en PHP).

Un 3ème attribut, `enctype`, facultatif, sert à spécifier le format des données envoyées. Cet attribut peut prendre 2 valeurs :

- `text/plain`, qui est la valeur par défaut et donc permet, comme dans l'exemple ci-dessus, d'omettre l'attribut `enctype`
- `multipart/form-data` qui doit être présent lorsque le formulaire comprend un champ de type fichier à télécharger (nous en reparlerons).

CHAMPS DE SAISIE DE TEXTE

L'internaute peut saisir une information dans un champ de saisie par l'intermédiaire de la balise `<input type="text" name="nom">`

Exemple d'un champ de saisie :


```
<input type="text" name="nom" size="50" maxlength="50" value="Julie DU-
BOIS">
```

Ici, `Julie DUBOIS` s'affiche par défaut dans le champ car cette valeur est pré-renseignée dans l'attribut `value`.

> **En HTML 5, la balise `<input>` ne nécessite pas de slash de fermeture : `<input type="text">` et non `<input type="text" />`**

Cette balise possède les attributs principaux suivants : seuls les attributs `type` et `name` sont obligatoires. Si l'attribut `name` est manquant pour un champ, les données saisies dans ce champ ne seront pas transmises au fichier de destination.

Attribut	Utilité
type	Obligatoire Le type de champ *input*. Valeurs possibles : ▪ text : champ de saisie simple ▪ password : la saisie sera remplacée par le caractère « * », par exemple pour les mots de passe (△ cela ne sécurise en rien les valeurs saisies). ▪ file : pour télécharger un fichier (△ nécessite d'ajouter l'attribut *enctype* de la balise `<form>` avec la valeur). ▪ radio : bouton radio ▪ checkbox : case à cocher ▪ submit : pour le bouton de soumission (envoi) du formulaire ▪ button : le champ sera un bouton (équivalent à l'élément HTML `<button>`), cela peut être le cas des champs de soumission. ▪ hidden : le champ sera caché à l'utilisateur ▪ number : champ avec flèches montantes et descendantes pour augmenter ou baisser un nombre. ▪ Autres types (s'assurer de la compatibilité avec les navigateurs)
name	Obligatoire Nom de la donnée. C'est le nom sous lequel l'information saisie ici sera connue et récupérée dans le script serveur déclenché par le formulaire.
id	Facultatif Identifiant (doit être unique). Son usage diffère légèrement de l'attribut *name* bien que dans la pratique ils aient souvent la même valeur.

value	Facultatif Texte définit comme valeur par défaut.
size	Facultatif Permet de définir un nombre de caractère visible pour le champ.
maxlength	Facultatif Permet de définir un nombre de caractère maximum saisissable.

Attributs additionnels (donc facultatifs)

Attribut	Utilité
autocomplete	Active l'autocomplétion du champ (par rapport aux saisies effectuées précédemment dans le navigateur (cache) pour le même champ. Cet attribut peut recevoir de nombreuses valeurs.
autofocus	Donne la priorité (focus) au champ lors du chargement du formulaire (c'est-à-dire que le curseur sera positionné directement dans le champ)
readonly	Met le champ en lecture seule, aucune saisie ne sera possible.
disabled	Désactive le champ, aucune interaction possible (ni saisie ni évènements).
placeholder	affiche (grisé) un texte d'indication qui s'effacera lors de la saisie d'un caractère.

> **Il existe de nombreux autres attributs applicables à la balise `<input>` mais il convient de s'assurer de leur compatibilité avec les navigateurs.**

ETIQUETTES DE CHAMPS

Pour indiquer à l'internaute quelle est l'information à saisir dans le champ, il faut ajouter une étiquelle de champ via la balise `<label>` :

```
<label for="nom">Nom :</label><input type="text" name="nom" id="nom">
```

La balise `<label>` possède obligatoirement l'attribut `for` qui permet d'associer l'étiquette au champ de formulaire souhaité.

Du côté du champ (balise `<input>`), il est nécessaire d'ajouter l'attribut `id` dont la valeur doit correspondre avec celle du `for` pour que l'association fonctionne.

En pratique, on renseigne les attributs `id` et `for` avec la même valeur que l'attribut `name`, sauf pour les champs de types choix (cases à cocher et bouton radio, vus plus loin dans ce cours).

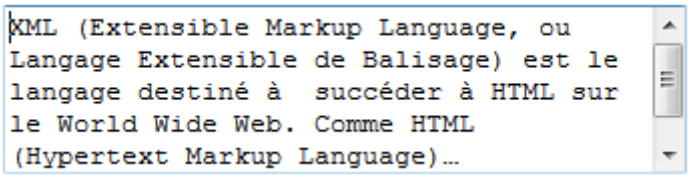
ZONES DE SAISIE MULTI-LIGNES

Lorsque l'on veut saisir du texte sur plusieurs lignes, on passe par des zones de saisie multi-lignes à l'aide de la balise `<textarea>...</textarea>`. Cette balise possède les attributs suivants :

Attribut	Utilité
name	Nom de la donnée. C'est le nom sous lequel l'information saisie ici sera connue et récupérée dans le script serveur déclenché par le formulaire.
id	Identifiant (doit être unique). Son usage diffère légèrement de l'attribut *name* bien que dans la pratique ils aient souvent la même valeur
rows	Nombre de lignes dans la zone de saisie.
cols	Nombre de caractères par ligne dans la zone de saisie.

La valeur par défaut contenue dans la zone de saisie multi-ligne est le texte compris entre les balises `<textarea>` et `</textarea>`.

Exemple d'une zone de saisie multi-ligne :




```
<textarea name="commentaire" rows="5" cols="30">XML (Extensible Markup Lan-
guage, ou Langage Extensible de Balisage) est le langage destiné à  succé-
der à HTML sur le World Wide Web. Comme HTML (Hypertext Markup Lan-
guage</textarea>
```

CASES A COCHER ET BOUTONS RADIO

Les données peuvent être sélectionnées par l'internaute dans des cases à cocher (plusieurs choix), ou dans des boutons radio (choix unique). Pour cela, on utilise les balises suivantes :

`<input type="checkbox" … >`, pour les cases à cocher,
`<input type="radio" … >`, pour les boutons radio.

Ces balises possèdent les attributs suivants :

Attribut	Utilité
name	Nom de la donnée. C'est le nom sous lequel l'information saisie ici sera connue et récupérée dans le script serveur déclenché par le formulaire.
value	Libellé correspondant à l'option à choisir.
checked	Indique que l'option sera sélectionnée par défaut.

Exemple de cases à cocher :

☐ **Bleu** ☑ **Rouge** ☐ **Vert**

```
<input type="checkbox" name="couleur1" value="Bleu" checked> Bleu
<input type="checkbox" name="couleur2" value="Rouge"> Rouge
<input type="checkbox" name="couleur3" value="Vert> Vert
```

Exemple de boutons radio :

○ **Bleu** ○ **Rouge** ◉ **Vert**

```
<input type="radio" name="couleur" value="Bleu"> Bleu
<input type="radio" name="couleur" value="Rouge"> Rouge
<input type="radio" name="couleur" value="Vert" checked="checked"> Vert
```

Les boutons radio doivent avoir le même nom (attribut `name`).

COMBOBOX (LISTES DE CHOIX)

L'internaute peut sélectionner les informations qu'il désire dans une liste pré-renseignée. La balise `<select>` permet de définir la liste des choix possibles. Chaque choix sera défini par une balise `<option>`, incluse dans le `<select>`.

Les attributs possibles de la balise `<select>` sont les suivants :

Attribut	Utilité
name	Nom de la donnée. C'est le nom sous lequel l'information saisie ici sera connue et récupérée dans le script serveur déclenché par le formulaire.
size	Si **size** vaut 1 (valeur par défaut), c'est une liste déroulante (un seul choix possible). Si **size** > 1, c'est une liste de choix avec ascenseur. (dans ce cas, on peut éventuellement avoir plusieurs choix : attribut MULTIPLE)
multiple	Indique que la sélection pourra être multiple (valable également sur les champs `input` de types `email` et `file`).

Les attributs possibles de la balise `<option>` sont les suivants :

Attribut	Utilité
value	Nom de la donnée pour l'option correspondante.
selected	Détermine le choix par défaut.

Exemple de liste déroulante :


```
<select name="couleur">
    <option value="Bleu">Bleu</option>
    <option value="Rouge">Rouge</option>
    <option value="Vert" selected>Vert</option>
</select>
```

Exemple de liste de choix :


```
<select name="couleur" size="3" multiple="multiple">
    <option value="Bleu">Bleu</option>
    <option value="Rouge">Rouge</option>
```

```
    <option value="Vert">Vert</option>
</select>
```

CHARGEMENT DE FICHIERS

Le type `file` sur une balise `<input>` permet le téléchargement d'un fichier du PC de l'utilisateur sur le serveur où est hébergée la page web.

L'utilisation de ce type requiert impérativement d'ajouter à la balise `<form>` :

- La valeur `post` à l'attribut `method` (le téléchargement de fichier en fonctionne pas avec une requête HTTP GET).
- L'attribut `enctype` avec la valeur `"multipart/form-data"` à la balise `<form>`.

Exemple :

```
<form action="destination.php" method="post" enctype="multipart/form-data">
<input type="file" name="fichier">
```

Le fichier sera alors stocké physiquement sur le serveur dans un répertoire dont le chemin sera à définir dans le traitement (langage de script côté serveur). En termes de sécurité, on recommande de stocker des fichiers sensibles dans des répertoires serveur non publics (c'est-à-dire qui ne sont pas accessibles via un navigateur web).

> **Le téléchargement d'un fichier sur un serveur via un formulaire présente des failles de sécurité importantes et graves qu'il convient de contrôler impérativement avec un langage côté serveur.**

Les attributs possibles :

Attribut	Utilité
name	Nom du champ.
id	Identifiant unique du champ.
accept	Permet de n'autoriser que certains types (MIME) de fichiers (ne constitue en rien une sécurité). Par exemple : `<input type="file" name="fichier" accept="image/png">` n'autorisera que les images de type *png.*
multiple	Sélection multiple de fichiers.
required	Champ requis (encore une fois, ne valide en rien la sécurité !).

Différentes informations sur le fichier (nom, taille, type MIME etc.) seront transmises par la page web au serveur. Ces informations seront à récupérer, contrôler et traiter via un langage de script côté serveur (PHP, ASP, Java, Python, Ruby...).

INCLUSION DE DONNEES CACHEES

Il est possible d'envoyer des données cachées au serveur (données non saisies par l'utilisateur mais créées dans le code HTML du formulaire). Ceci se fait en affectant la valeur `hidden` à l'attribut `type` :

```
<input type="hidden" …>.
```

> **L'attribut `hidden` ne sécurise en rien un formulaire. En affichant le code source (touches `CTRL+U` ou inspecteur de code des outils de développement des navigateurs – `F12`), la valeur saisie sera visible en clair.**

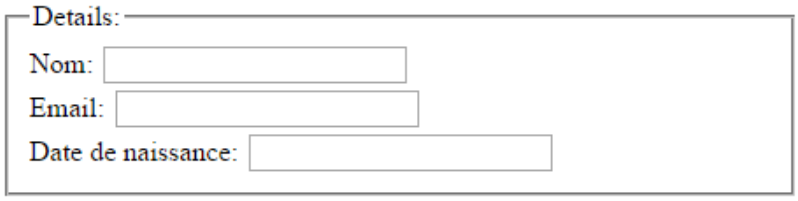
GROUPER DES CHAMPS

Il est possible de regrouper visuellement certains champs pour clarifier la présentation de votre formulaire par l'intermédiaire de la balise `<fieldset>`.

L'utilisation de cette balise nécessite obligatoirement la balise fille `<legend>` à placer immédiatement après et qui permet de donner un nom au groupe de champs.

Exemple :

```
<fieldset>
    <legend>Détails :</legend>
    Nom: <input type="text"><br>
    Email: <input type="text"><br>
    Date de naissance: <input type="text">
</fieldset>
```



NOUVEAUX TYPES DE CHAMPS

Datalist

HTML5 a ajouté l'élément `<datalist>` qui permet de cibler une liste d'éléments via un champ `input`. La saisie sera alors autosuggérée au fur et à mesure.

Exemple :

```
<label>Sélectionnez votre métier</label>
<input type="text" name="metiers1" id="metiers2" list="metiers3"><br>
<datalist id="metiers3">
<option value="webmaster">
<option value="développeur">
<option value="administrateur B.D.D.">
</datalist>
```

Output

La balise `<output>` (= sortie) permet l'affichage du résultat d'un calcul lors de la soumission d'un formulaire grâce à l'instruction HTML d'événement `onsubmit`.

Exemple :

```
<form method="post" action="post.php" id="monform" onsubmit="resultat.value
= parseInt(nombre1.value) + parseInt(nombre2.value); return false;">
  <input type="number" name="nombre1" id="nombre1"> + <input type="number"
name="nombre2" id="nombre2"> = <output for="number1 number2"
name="resultat" form="monform"></output><br><br>
  <input type="submit" name="envoi" value="Calculer">
</form>
```

> **Notez l'utilisation de la fonction javascript `parseInt()` afin de convertir en entier. Sans celle-ci, le résultat affiché, par exemple pour 2 + 1, serait 21, c'est-à-dire une simple concaténation de chaînes.**

VALIDATION DE FORMULAIRES

HTML5 propose de nouveaux attributs pour aider à la validation des formulaires.

Attribut	Utilité
required	permet d'indiquer que le champ est obligatoire. Un message d'erreur s'affichera lors de la soumission du formulaire.
pattern	Motif d'expression régulière (nous aborderons ce point dans le cours Javascript) qui permet de spécifier le format attendu pour la saisie
title	description du contenu et du format. Ce message est affiché en cas d'erreur

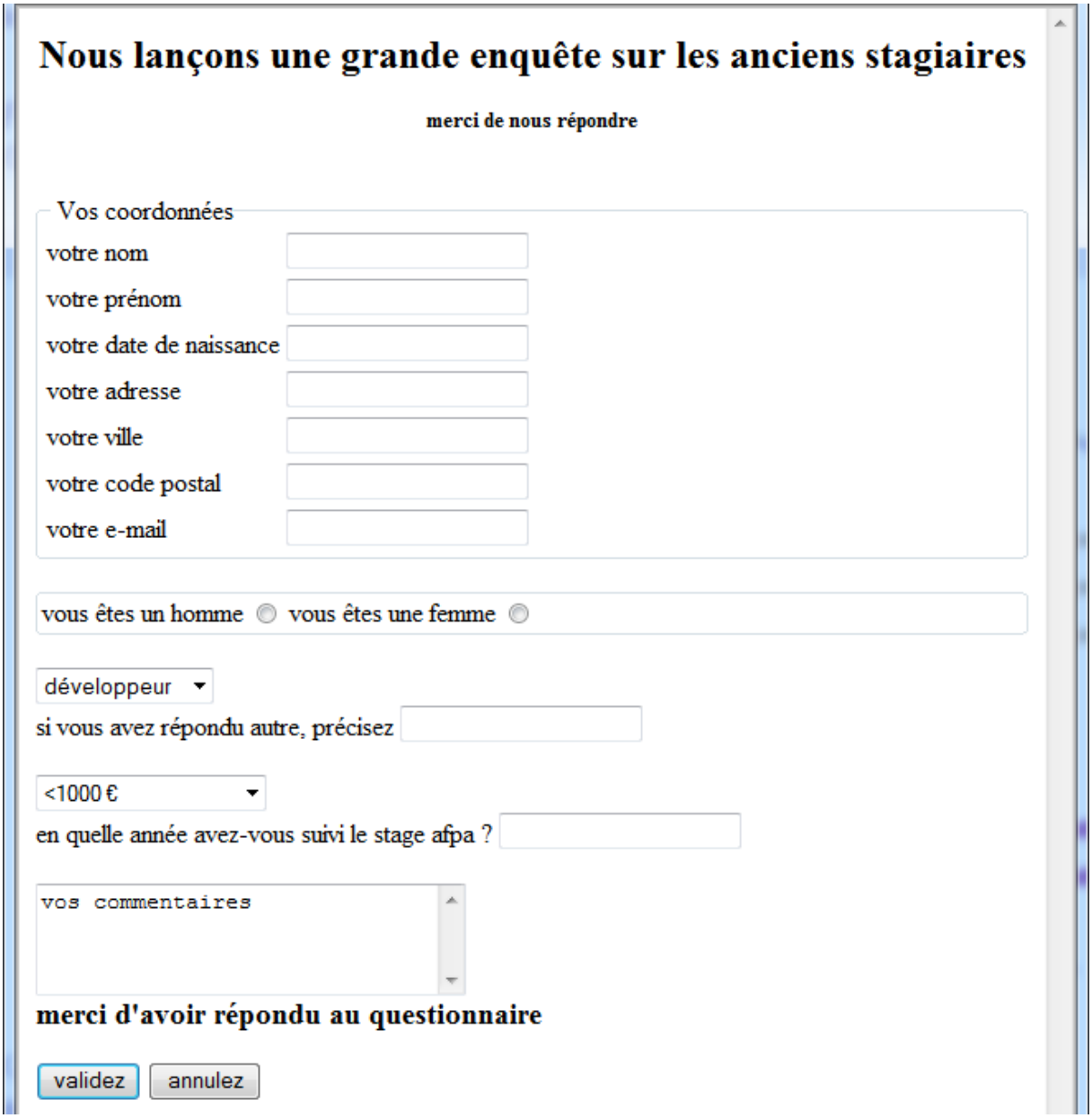
Exemples :

```
<input type="text" placeholder="Entrez votre nom" pattern="^[a-z]+$"
required title="Entrez votre nom (que des caractères entre a et z)" />
```

> **L'utilisation des attributs `pattern` et `required` ne sécurise en rien un formulaire, mais relève davantage de l'ergonomie (IHM). Il convient de toujours d'effectuer des contrôles côté serveur (avec un langage du type PHP, ASP, Python…).**

EXERCICES D'APPLICATION

1. **Réalisez la page HTML dont l'exemple est donné ci-dessous. Observez les données transmises avec la méthode `get` dans l'url.**



Utilisez `http://bienvu.net/script.php` comme valeur pour l'attribut action de votre balise `<form>`.