**HTML5 : Formulaires**

* Objectifs
* Construction d'un formulaire
* Elément <form>
* Méthodes GET et POST
* Elément <input>, ses types et attributs
* Les autres éléments

# Objectif des formulaires

* La plupart des éléments HTML permettent d'**afficher** sur le navigateur **une information reçue** du serveur. Cela suffit avec les sites statiques

Il n’y avait que de l’information affichée mais pas d’interaction possible entre site et utilisateurs.

* Mais **comment envoyer une demande** ?

Comment interagir avec le site ?

Comment avoir une réponse personnalisée ?

* Les **formulaires** permettent au visiteur du site d'**entrer des informations** qui, une fois validées, sont **envoyées au serveur.** La réponse est alors plus ciblée à la demande.

# Construction d'un formulaire

* Un **formulaire** est contenu dans un élément

**<form>** et souvent est composé de **champs**

<input>, <textarea> quand l’utilisateur doit mettre du texte (genre remarque, commentaire), ... et d'un **bouton d'envoi**

* HTML5 permet, via un formulaire, d'**entrer des données** mais son but n'est **pas de les traiter** (le serveur peut le faire avec PHP, MySQL pour conserver les données côté serveur, ...) **ni de les valider** vérification qu’il n’y a pas de caractères fautifs (lettres quand on demande un âge) ou si les conditions sont remplies (par exemple plus de 18 ans) ou encore si une adresse mail contient bien @ etc. (le client peut le faire avec JavaScript, s'il n'est pas désactivé).
* On définit **quelles sont les données** à envoyer au serveur, **où** et **par quelle méthode**

Tout ça peut être validé avant d’être envoyé vers le serveur. La première validation sera donc faite côté utilisateur (avec JS) puis deuxième validation côté serveur avec PHP.

# Elément <form>

L'élément **<form>** contient les autres éléments du formulaire. Il est défini par ses attributs : certains attributs sont obligatoires

* **action** : URL où sont envoyées les données « les données vont sur telle page ». S’il est vide, la page du formulaire est envoyée à elle-même.
* **method** : méthode d'envoi HTTP au serveur pour envoyer des données au serveur il faut une méthode, il y en a 2 (GET et POST)
* **name** : nom du formulaire <https://developer.mozilla.org/fr/docs/Web/API/HTMLFormElement/name>
* **enctype** : type MIME type de fichiers qu’on envoie associé au contenu lors de l'envoi au serveur (données encodées) « encoding type » – le type d’encodage. On va dire comment les données vont être encodées pour être en
* voyées au serveur.

Quand on fera de l’upload, il faudra impérativement préciser le type d’encodage.

* **novalidate** : le navigateur ne doit pas valider les données avant l'envoi au serveur
* Et aussi : **accept-charset**, **autocomplete** (suggestion pour compléter ce qu’on demande comme les adresses etc.), **target**

# Méthodes GET et POST

* La méthode par défaut est **GET**
  + Les données envoyées sont **écrites dans l'URL** demandée (via paires de clé/valeur)
  + **Longueur limitée** (varie selon le navigateur ou le serveur : 255, 2048, 64k,)
  + **À éviter** pour envoyer des **données sensibles**
  + Utile pour **débugger** et pour les **bookmarks** (favoris, marques-page, etc…)
* La méthode **POST** **est privilégiée**
  + N'apparaît pas à l'écran donc plus sûre les données sont cachées dans le message envoyé au serveur
  + Pas de taille limite

**Élément <label>**

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/Label>

*L'élément HTML <label> représente une légende pour un objet d'une interface utilisateur. Il peut être associé à un contrôle en utilisant l'attribut for ou en plaçant l'élément du contrôle à l'intérieur de l'élément <label>. Un tel contrôle est appelé contrôle étiqueté par l'élément <label>.*

Ça signifie en fait que sans l’élément « for », je pourrais cocher oui ou non, mais je ne saurais pas pour quoi je clique, cela ne sera pas afficher

* Ne pas hésiter à faire le test directement sur le lien de mdn, c’est très parlant et beaucoup plus clair.

# Elément <input>

● L'élément **<input>** balise auto-fermante a plusieurs fonctionnalités et apparences selon son attribut **type :** il faut toujours préciser le type= « … »

* **text** : champ de texte
* **password** : champ mot de passe
* **submit** : bouton d'envoi
* **reset** : bouton de remise à zéro revenir aux valeurs initiales du formulaire
* **radio** : bouton radio (un seul choix)
* **checkbox** : case à cocher (plusieurs choix)
* **button** : bouton
* **hidden** : champ caché (champ de texte invisible) sert pour stocker dans la page des infos que l’utilisateur ne verra pas mais que la machine a besoin. Par exemple quand on a un formulaire sur plusieurs pages, quand on passe d’un page à l’autre, il oublie tout. Du coup le champ caché va servir pour le stockage et la récupération des données.
* **image** : bouton graphique cliquable
* **file** : sélection d'un fichier (pour l’upload) quand on veut télécharger un fichier, il faut une fenêtre qui s’ouvre et qui permet le choix
* **required** : rend un champ obligatoire. Un message s'affichera au passage de la souris et l'envoie du formulaire sera bloqué. C'est un attribut Booléen - il se suffit à lui-même, pas besoin de précision
* **placeholde**r va permettre d'afficher un petit texte dans mon input pour que niveau ergonomie cela soit plus évident.

**Nouveaux types <input>** (depuis HTML5)

● Des **nouveaux types** ont été ajoutés avec HTML5 (certains navigateurs proposent un affichage particulier)

* **tel** : numéro de téléphone
* **url** : adresse, URL
* **email** : courriel
* **search** : champ de recherche
* **datetime-local** : date et heure
* **range** : valeur limitée entre 2 bornes
* **date** : date
* **time** : heure
* **month** : mois
* **week** : semaine
* **number** : nombre
* **color** : couleur

sélecteur de couleur qui s’affiche

# Attributs pour <input>

* Avec l'élément **<input>**, on a les attributs :
  + **name** : définit le nom du champ
  + **value** : définit la valeur transmise
* L'attribut **for** permet de **lier** un **<input>** avec un **<label>**
* Selon les types, les **autres attributs** sont :

**multiple** (email, file), **checked** (radio, checkbox), **accept** (file), **height**/**width**/**alt**/**src** (image), **min**/**max**/**step** (date, time, datetime-local, month, week, number, range)

# Exemple

**Côté CLIENT : test.html**

<form method="GET" name="login" action="test.php">

<label>Identifiant :</label><input type="text" name="nom">

<label>Mot de passe :</label><input type="password" name="mdp">

<input type="submit" value="Se connecter"> </form>

**Côté SERVEUR : test.php**

<?php

if (isset($\_GET['nom']) && isset($\_GET['mdp'])) {

echo "<h1>Bienvenue ".$\_GET['nom'].", sur le site !</h1>";

}

?>

# Zone de texte <textarea>

* Il permet une entrée de **texte multilignes** **entre** les balises **<textarea> et </textarea>** ce n’est donc pas une balise orpheline
* ● Les attributs spécifiques sont :
  + **rows** : nombre de lignes par défaut
  + **cols** : nombre de colonnes par défaut
  + **wrap** : retours à la ligne (hard) ou non (soft)

Si un style CSS est spécifié, les attributs rows et cols sont ignorés <https://www.w3schools.com/tags/tag_textarea.asp>

On peut redimensionner le textarea – ou on peut bloquer cette fonction (si on veut mettre un nombre de caractères maximum) etc. il y a plein d’attributs possibles au niveau CSS comme pour du texte classique.

**Liste de choix**

<https://www.w3schools.com/tags/tag_select.asp>

* On a une combinaison de l'élément **<select>** qu’on va beaucoup utiliser pour faire des listes déroulantes contenant des éléments **<option>**

🡪 autant que de choix dans la liste déroulante

* L'attribut **size** représente le nombre d'options visibles et on peut en choisir plusieurs avec **multiple** par exemple, si on a une liste avec 10 possibilités, il n’en affiche que 3 à la fois
* L'élément **<option>** est caractérisé par les attributs **value, name, label** et **selected** <https://www.w3schools.com/tags/tag_option.asp>
* On peut **regrouper visuellement** des options avec l'élément **<optgroup>** si on a une liste de 30 possibilités, on fait par ex 5 groupes avec dedans 5/6/7 possibilités qui s’afficheront quand on passera la souris dessus

# Radio-bouton et Case à cocher

* Retour sur certains éléments <input>
  + **Radio-bouton** : <input **type="radio"**>
  + **Case à cocher** : <input **type="checkbox"**>
* Souvent, on les utilise **en groupe** 🡪 attribut « name » à qui on mettra un id par la suite
  + Radio-bouton : **un seul coché** par groupe

Exemple <input type= « radio » name= « titre »><label>monsieur</label>

* + Case à cocher : **0, 1 ou plusieurs** cochés

Exemple <input type= « checkbox »>…

* Pour les grouper, l'attribut **name** doit **être le même**, on les distingue par l'attribut **value** etl'attribut **checked** indique celui qui est coché

# Bouton cliquable

Sert à mettre un bouton sur lequel on pourra cliquer mais qui n’enverra pas le formulaire. On peut par exemple avoir un bouton qui activera du son, affichera une image etc.

* Quand on a besoin d’un **bouton** qui n’a pas pour but d’envoyer les données vers le serveur

(type="submit"), on peut utiliser **<button>**

* On doit ajouter l’attribut **type="button"** pour qu’il soit pris en compte correctement
* Exemple :

<**button** **type="button"**

onclick="alert('Bonjour !')"**>**Cliquez ici !**</button>** quand on cliquera dessus, il exécutera du JS

# <fieldset> et <legend>

* On peut visuellement **regrouper** des éléments de formulaire dans un cadre avec **<fieldset>** genre mettre mes possibilités dans un cadre
* On ajoute ensuite une légende avec **<legend>**

<form action="login.php">

**<fieldset>**

**<legend>** Identification : **</legend>**

<label for="username">Utilisateur:</label><br>

<input type="text" id="username" name="username"><br>

<label for="pwd">Mot de passe :</label><br>

<input type="password" id="pwd" name="pwd"><br><br>

<input type="submit" value="Se connecter">

**</fieldset>**

</form>

# 