

*Web : principes
de base
(HTML/CSS)*

DA1
Henallux

Module 5

Formulaires

HTML

Au programme

➤ Les formulaires HTML

- *Qu'est-ce ? À quoi ça sert ?*
- *Protocoles GET et POST*

➤ HTML et CSS pour les formulaires

- *Structure d'un formulaire HTML*
- *Données dans un formulaire*
- *Mise en forme d'un formulaire*

➤ Validation côté client

- *Options de validation*
- *Un mot (important) sur la sécurité de la validation*

Les formulaires HTML

Au programme de ce chapitre...

- **Qu'est-ce que c'est ?**
- **Objectifs associés à un formulaire**
 - *3 tâches qu'un formulaire est censé effectuer*
- **Protocoles GET et POST**
 - *Protocoles pour l'envoi de données*

Ensuite : *HTML et CSS pour les formulaires*

Qu'est-ce que c'est ?

Un **formulaire HTML** permet (généralement) de

- 1) **recueillir** des informations
- 2) **vérifier** certaines contraintes sur ces informations, et
- 3) **envoyer** ces infos via le web.

Il s'accompagne d'au moins un bouton de soumission (**submit**) permettant l'envoi des informations.

The image shows a screenshot of an HTML form with three main sections:

- Données personnelles**: Contains input fields for 'Nom', 'Prénom', 'Âge' (with a value of 18 and a spinner), and 'Date d'anniversaire (JJ/MM)'.
- Études**: Contains a dropdown menu for 'Section' (set to 'Comptabilité'), radio buttons for 'Année d'étude' (1re, 2e, 3e), and checkboxes for 'Langues connues' (Anglais, Néerlandais, Allemand).
- Commentaires**: Contains a text area labeled 'Commentaires éventuels'.

At the bottom of the form, there is a button labeled 'Submit Query'. A green arrow points from the text 'Il s'accompagne d'au moins un bouton de soumission' to this button, which is also circled in green.

Objectifs d'un formulaire

Recueillir des infos

Vérifier les infos

Envoyer les infos

Objectif 1 : **Recueillir des informations** de l'utilisateur

- Informations de **natures diverses** :
 - données textuelles (ou mot de passe ou...)
 - données numériques (ou date ou...)
 - éléments à cocher
 - fichier à télécharger
 - choix d'une couleur...

- Chaque information est un couple **clef/valeur**. (clef = nom)

Exemples : **nom**/Simpsons
âge/45

Nom :

Prénom :

Âge :

Section :

Année d'étude : ☒ 1re ☐ 2e ☐ 3e

Langues connues :

☐ Anglais

☐ Néerlandais

☐ Allemand

Objectifs d'un formulaire

Recueillir des infos

Vérifier les infos

Envoyer les infos

Objectif 2 : **Vérifier certaines contraintes** sur ces informations

- Exemples de **contraintes** :

- Information requise (champ ne peut pas être vide)
- Minimum/maximum
- Liste de valeurs valides
- Format à respecter (email)



Prénom :

Âge : 18

Date d'anniversaire (JJ/MM) :

- C'est un type de **validation**.

- Validation **locale** (sur le client) : réponse immédiate mais pas du tout sécurisée vu que tout reste sous le contrôle de l'utilisateur !
- Validation **par le serveur** : plus sûre, utilisant potentiellement des bases de données (login, autorisations/rôles...)

Objectifs d'un formulaire

Recueillir des infos

Vérifier les infos

Envoyer les infos

Objectif 3 : **Envoyer les informations**

Cela nécessite certaines informations :



• **Où ?** *Vers quelle adresse web envoyer les données ?*

Les données sont généralement envoyées vers un serveur pour y être traitées par un script (PHP par exemple) qui va

- consulter/modifier une base de données,
- construire la réponse (page HTML) à renvoyer

• **Comment ?** *Sous quel format communiquer les données ?*

Deux protocoles utilisés : GET et POST


• **Et ensuite ?** *Que faire de la réponse reçue (du serveur) ?*

L'afficher dans la même page ? Dans un nouvel onglet ?

Protocoles GET et POST

= comment les infos (couples clef/valeur) sont envoyées :

- **Méthode GET** : données incluses dans l'adresse web.
 - Exemple (recherche Google) :



clef valeur

```
https://www.google.be/search?q=html+tree
```
 - Informations visibles dans l'URL (dans l'historique aussi)
 - Conversion automatique des caractères problématiques (?&=...)
- **Méthode POST** : données transmises au sein de la demande.
 - Exemple (horaire Portail) : en cas de rafraîchissement, le navigateur indique qu'il doit envoyer à nouveau certaines infos.
 - Données non reprises dans l'historique du navigateur.
 - Moins de contrainte sur la taille des données

HTML et CSS pour les formulaires

Au programme de ce chapitre...

- **Format de base d'un formulaire HTML**
 - Balise `<form>` et boutons d'actions
- **Organisation interne d'un formulaire HTML**
 - Libellés dans formulaire
 - Structuration / découpe en sections
- **Champs d'un formulaire HTML**
 - Balises `<input/>`, `<textarea>` et `<select>`
- **Mise en forme (CSS) d'un formulaire**
 - Pseudo-classes utiles

Ensuite : *Validation des données*

Format de base

Balise principale : **<form>**

```
<form  
  action="http://monsite.be/monscript.php"  
  
  method="get"  
  
  target="_blank">  
  ...  
</form>
```

Cible où soumettre les informations

Protocole (get ou post)

Où afficher la réponse

À l'intérieur d'un formulaire, on peut trouver entre autres :

- des **sections** (**<fieldset>**);
- des **déclencheurs de soumission/remise à zéro** (**<input/>** ou **<button>**);
- les **champs** du formulaires (**<input/>**, **<textarea>**, **<select>**);
- des **étiquettes** textuelles ou graphiques (**<label>**).

Format de base

Le formulaire devrait contenir un **déclencheur de soumission** pour l'envoi des données.



- Via **<input/>** :
`<input type="submit" value="Envoyer !" />`
- Via **<button>** :
`<button type="submit">Envoyer !</button>`

Remarques

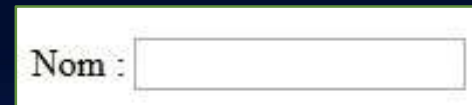
- Un formulaire peut avoir plusieurs boutons de soumission.
Exemples : méthodes d'envoi ou des adresses différentes.
« Envoyer le mail » et « Sauvegarder comme brouillon »
- On peut également créer des boutons de remise à zéro/réinitialisation en utilisant comme type **reset** au lieu de **submit**.

Organisation interne

Pour structurer le contenu du formulaire, on peut utiliser des balises standards comme `<p>`, `<div>`, ``... mais aussi :

- balise `<label>` pour spécifier un libellé / une étiquette

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



Établir un lien avec le champ : attributs `for` et `id`.

- Confort d'utilisation : en cas de clic sur l'étiquette, le champ est activé (coché/décoché ou reçoit le focus).
- Accessibilité : le text-to-speech lit le libellé lire quand on clique dans le champ.

- balise `<fieldset>` pour spécifier des sections (slide suivant).

Organisation interne

Sections dans un formulaire :

```
<fieldset>
  <legend>Données
    personnelles</legend>
  ...contenu...
</fieldset>
<fieldset>
  <legend>Études</legend>
  ...contenu...
</fieldset>
```

- La balise `<legend>` permet d'ajouter un titre à une section de formulaire.

Données personnelles

Nom :

Prénom :

Âge :

Date d'anniversaire (JJ/MM) :

Études

Section :

Année d'étude : ☒ 1re ☐ 2e ☐ 3e

Langues connues :

☐ Anglais

☐ Néerlandais

☐ Allemand

Champs d'un formulaire

`<input/>``<select>``<textarea>`

- **Balises** pour spécifier les champs d'un formulaire :

- `<input/>` pour la plupart des champs
 - Champs textuels (dont email, password, url...)
 - Champs numériques (dont dates, glissière...)
 - Options à cocher
 - ...

Selon l'attribut type

- `<select>` pour une liste d'options déroulante
- `<textarea>` pour un texte plus long

- Quelques **attributs** communs aux champs :

- **name** pour spécifier la clef associée au champ ;
- **value** pour spécifier la valeur (initiale) ;
- **id** pour lier le champ à son libellé.

Le nom et l'id sont souvent identiques, mais ce n'est pas obligatoire !

Champs d'un formulaire

`<input/>``<select>``<textarea>`

Champ textuel simple : `<input type="text" />`

- Attributs :
 - `size` : taille de la zone affichée (en nb de caractères)
 - `placeholder` : texte initial, disparaît dès que l'utilisateur entre une valeur
 - [V] `minlength` / `maxlength` : taille minimale/maximale (en nb de caractères)
 - [V] `pattern` : expression régulière décrivant le format attendu

[V] = attribut lié
à la validation

- Quelques autres types similaires :
 - `email` : pour une adresse mail
 - `password` : pour un mot de passe (affichage caché *****)
 - `url` : pour une url

- Exemple :

Prénom :

```
<label for="prenom">Prénom :</label>
<input type="text" id="prenom" name="firstName"
      placeholder="Tapez ici !" />
```


Champs d'un formulaire

```
<input/>
```

```
<select>
```

```
<textarea>
```

Champ numérique simple : `<input type="number" />`

- Attributs :
 - **placeholder** : texte initial, disparaît dès que l'utilisateur entre une valeur
 - [V] **min** / **max** : valeur minimale/maximale
 - [V] **step** : précision / pas d'incrément ou de désincrementation
= 1 par défaut (n'autorise que des entiers)
step="0.1" pour permettre des réels avec 1 décimale
- Quelques autres types similaires :
 - Pour dates/temps : **date**, **datetime-local**, **month**, **time**, **week**...
 - Pour une glissière : **range**
- Exemple (sans le label) :

```
<input type="number" min="16" max="99" value="18" name="age" />
```



Champs d'un formulaire

```
<input/>
```

```
<select>
```

```
<textarea>
```

Cases à cocher : `<input type="checkbox" />`

- Attributs :

- `checked` (booléen) : case cochée par défaut
- `value` : valeur renvoyée avec la clef si l'option est cochée (par défaut, "on")



Checkbox
choix
multiples

- Exemple (label placé après le champ) :

```
<input type="checkbox" name="nd" id="nd" />
<label for="nd">Néerlandais</label>
```

Langues connues :

- ☐ Anglais
☐ Néerlandais
☐ Allemand

Choix à cocher : `<input type="radio" />`

- Les champs d'un groupe ont le même attribut `name` (= la clef à laquelle sera associée la valeur sélectionnée). L'attribut `value` permet de les distinguer.



Bt radio
choix
unique

- Exemple :

```
<input type="radio" name="annee" id="an1" value="1" checked />
<label for="an1">1re</label>
<input type="radio" name="annee" id="an2" value="2" />
<label for="an2">2e</label>
```

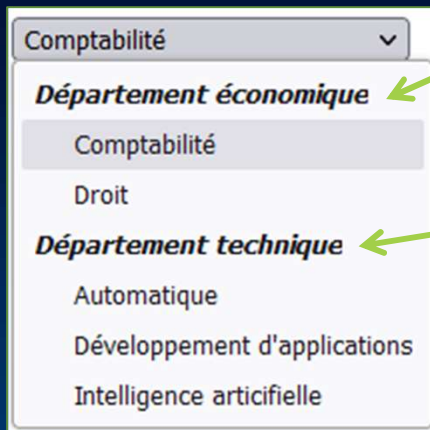
Année d'étude : ☒ 1re ☐ 2e ☐ 3e

Champs d'un formulaire

`<input/>``<select>``<textarea>`

Liste déroulante : `<select>...</select>`

- Attributs :
 - **multiple** (booléen) : autorise des choix multiples dans la liste
 - **size** : nombre d'options affichées / hauteur (si multiple)
 - **selected** (booléen, sur `<option>`) : sélectionnée par défaut
- Exemple :



```
<select name="sec" id="section">
  <optgroup label="Département économique">
    <option value="cp">Comptabilité</option>
    <option value="dr">Droit</option>
  </optgroup>
  <optgroup label="Département technique">
    <option value="au">Automatique</option>
    <option value="da">Développement d'apps</option>
    <option value="ia">Intell artificielle</option>
  </optgroup>
</select>
```

Valeur renvoyée (associée à la clef sec) si cette option est choisie

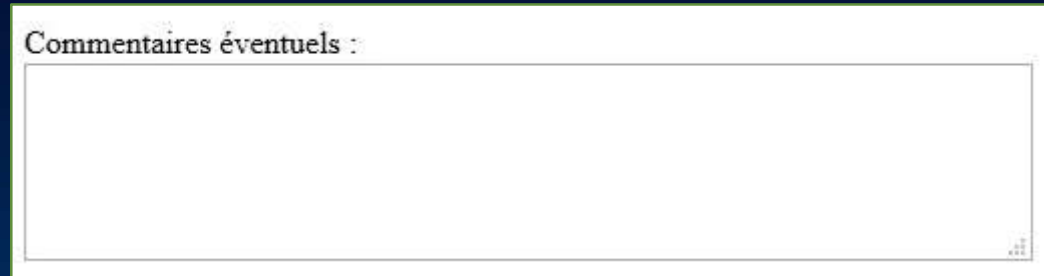
Champs d'un formulaire

`<input/>``<select>``<textarea>`

Bloc de texte : `<textarea>Texte initial</textarea>`

- Attributs :
 - **cols** : largeur initiale en nombre de caractères
 - **rows** : hauteur initiale en nombre de lignes

- Exemple :



Commentaires éventuels :

Champs d'un formulaire

- Quelques attributs utilisables pour tous les champs :
 - **disabled** (booléen) : champ « grisé »/inactif
 - **autofocus** (booléen) : le champ reçoit le focus dès le chargement
 - **readonly** (booléen) : champ non modifiable par l'utilisateur
 - (voir aussi la section sur la validation)
- Quelques autres types pour **<input/>** :
 - **image** : pour un « bouton » de soumission prenant la forme d'une image et renvoyant la position du curseur lors du clic d'envoi
 - **file** : pour choisir un fichier à envoyer
 - **color** : pour choisir une couleur
 - **hidden** : champ caché (rempli automatiquement via JavaScript)

Mise en forme via CSS

On peut utiliser les sélecteurs déjà vus précédemment.

Données personnelles

Nom :

Prénom :

Âge :

Date d'anniversaire (JJ/MM) :

Études

Section :

Année d'étude : ☒ 1re ☐ 2e ☐ 3e

Langues connues :

☐ Anglais

☐ Néerlandais

☐ Allemand

Commentaires

Commentaires éventuels :

Submit Query

Données personnelles

Nom :

Prénom :

Âge :

Date d'anniversaire (JJ/MM) :

Études

Section :

Année d'étude : ☐ 1re ☒ 2e ☐ 3e

Langues connues :

☐ Anglais

☐ Néerlandais

☐ Allemand

Commentaires

Commentaires éventuels :

Submit Query

Mise en forme via CSS

Il existe également des pseudo-classes spécifiques :

- `:focus` \equiv l'élément qui a le focus
- `:disabled` \equiv les éléments désactivés (\leftrightarrow `:enabled`)
- `:checked` \equiv les éléments cochés (checkbox ou radio)
- `:read-only` \equiv les éléments en lecture seule (\leftrightarrow `:read-write`)
- `:placeholder-shown` \equiv les éléments avec un placeholder affiché

Par exemple (fond jaune sur l'élément qui a le focus) :

```
input:focus, textarea:focus {  
    background-color:yellow;  
}
```


Validation côté client

Au programme de ce chapitre...

➤ **Contraintes et validation des données en HTML**

- *Comment exprimer des contraintes de validation en HTML ?*

➤ **Sélecteurs CSS relatifs à la validation**

- *Encore plus de pseudo-classes !*

Contraintes de validation des données en HTML

Quelques **attributs relatifs à la validation** :

- **required** (booléen) : champ qui doit avoir une valeur
- Sur les champs textuels :
 - **minlength** et **maxlength** : nb de caractères minimal/maximal
 - **pattern** : expression régulière imposant un format
 - Les types email et url sont associés par défaut à un format.
- Sur les champs numériques :
 - **min** et **max** : valeur minimale / maximale
 - **step** : précision



- Remarque importante
Les validations côté client ne servent qu'au confort de l'utilisateur.
Il est très facile de passer outre !
- Par défaut, la soumission est bloquée si les données ne sont pas validées.
Pour enlever ce blocage : attribut booléen **novalidate** sur la balise **<form>** (ou l'attribut **formnovalidate** sur le bouton de soumission).

Contraintes de validation des données en HTML

Quelques exemples

- Nom requis :

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

- Âge entre 16 et 99 :

```
<input type="number" min="16" max="99" value="18" name="age" id="age" />
```

- Cote par incrément de .5 :

```
<input type="number" min="1" max="20" step="0.5" />
```

- Date anniversaire requise et au format JJ/MM :

```
<input type="text" name="anniv" id="anniv" required pattern="[0123][0-9]/[01][0-9]" />
```

Sélecteurs CSS pour la validation

Pour cibler les champs selon leur validation, on peut utiliser les **pseudo-classes**

- **:required** : champ qui doit avoir une valeur (↔ **:optional**)
- **:valid** : champ à valeur valide (↔ **:invalid**)
- **:in-range** : champ numérique valide (↔ **:out-of-range**)

Par exemple (fond jaune sur l'élément qui a le focus) :

```
input:invalid {  
    background-color: orange;  
}
```