

Technologies du Web

Master COMASIC

Le langage HTML

Antoine Amarilli¹

27 novembre 2014

1. Matériel de cours inspiré de notes par Pierre Senellart. Merci à Marc Jeanmougin, Antonin Delpeuch et Pierre Senellart pour leur relecture.

Présentation générale

- **HyperText Markup Language.**
- Décrit une page Web. (Autres documents sur le Web...)
- Normalisé par le **W3C** (industriels et académiques) et **WHATWG**.
- Format ouvert et **documenté** : HTML 5 specification (548 pages, 2014)
- Langage à **balises**, structure et contenu du document.
- Pas de **présentation** (CSS).
- Pas de **comportements dynamiques** (JavaScript).

Versions

- Tentative complexe de langage général : **SGML**.
 - **HTML** inspiré de SGML, versions 1 à 4.
 - Tentative plus simple de langage général : **XML**.
 - **XHTML** : HTML conforme à XML (1.0 et 1.1).
 - **HTML 5** : pas du XML, mais... XHTML 5.
 - **Autres formats** dérivés de XML.
- Contrairement à HTTP : plusieurs versions dans la nature, et **peu de respect pour les standards** !

Table des matières

1 Généralités

2 Structure

3 Texte, listes et tableaux

4 Multimédia

5 Formulaires

Principes du balisage

- Balise **ouvrante** (``) et **fermante** (``) :

Exemple `illustratif`

- La nature et répétition des **espaces** est ignorée :

Peu

importe !

- Balises **sans fermeture** : `
` pour un retour à la ligne.
(Ou `
` par compatibilité avec XML.)

- **Attributs** :

```
<p>Un <a href="http://example.com/">lien</a>.</p>
```

- **Commentaires** :

```
<!-- non affiché -->
```

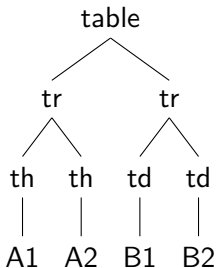
- **Entités HTML** : caractères rares et échappement

```
<p>&lt;p&gt;C'est m&eacute;ta&nbsp;!&lt;/p&gt;</p>
```

Imbrication

- Il faut ouvrir et fermer les balises dans le **bon ordre** :
<a> et non <a>
- **Règles** : pas possible de mettre n'importe quelle balise n'importe où. (En théorie du moins...)
- Interprétation **arborescente**.

```
<table>
  <tr>
    <th>A1</th>
    <th>B1</th>
  </tr>
  <tr>
    <td>A2</td>
    <td>B2</td>
  </tr>
</table>
```



Structure générale

- Le DOCTYPE précise la **version** de HTML utilisée.
→ Cet exemple est en HTML5.
- L'attribut lang (optionnel) précise la **langue**.

```
<!DOCTYPE html>  
<html lang="fr">  
  <head>  
    <!-- en-tête -->  
  </head>  
  <body>  
    <!-- corps -->  
  </body>  
</html>
```

Structure générale

- Le DOCTYPE précise la **version** de HTML utilisée.
→ Cet exemple est en HTML5.
- L'attribut lang (optionnel) précise la **langue**.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <!-- en-tête -->
  </head>
  <body>
    <!-- corps -->
  </body>
</html>
```

(Démonstration : afficher le source de <http://www.lemonde.fr/>.)

En-tête

- L'en-tête contient les **méta-données** :

```
<head>
  <meta charset="utf-8">
  <title>Titre de la page</title>
  <meta name="description" content="blah">
  <!-- ... -->
</head>
```

- Indiquer l'**encodage**.
- Indiquer le **titre**.
- Autres indications pour les moteurs de recherche, le cache...
- `<script>` : voir plus tard.

Soupe de balises

- Les gens écrivent souvent du HTML **n'importe comment**.
- Compatibilité avec **tous les sites** depuis les débuts du Web.
- Les navigateurs sont **très** tolérants.
- Émulation **non-standardisée** des défauts d'interprétations de certains anciens navigateurs! (quirks mode).

Soupe de balises

- Les gens écrivent souvent du HTML **n'importe comment**.
- Compatibilité avec **tous les sites** depuis les débuts du Web.
- Les navigateurs sont **très** tolérants.
- Émulation **non-standardisée** des défauts d'interprétations de certains anciens navigateurs! (quirks mode).

(Démonstration : soupe de tags.)

Soupe de balises

- Les gens écrivent souvent du HTML **n'importe comment**.
- Compatibilité avec **tous les sites** depuis les débuts du Web.
- Les navigateurs sont **très** tolérants.
- Émulation **non-standardisée** des défauts d'interprétations de certains anciens navigateurs! (quirks mode).

(Démonstration : soupe de tags.)

(Démonstration : validateur W3C.)

Table des matières

- 1 Généralités
- 2 Structure
- 3 Texte, listes et tableaux**
- 4 Multimédia
- 5 Formulaires

Texte

- Texte à mettre dans des **paragraphes** :
 - Paragraphes délimités avec `<p> ... </p>`.
 - Pas de texte directement dans `<body>` !
- **Retour à la ligne** avec `
`.
- Mettre **en valeur** dans un paragraphe :
 - ` ... `
 - ` ... `
 - `<mark> ... </mark>`

Titres et structure

- **Titres** : de `<h1>` à `<h6>` (importance décroissante).
- Autres façons (purement sémantiques) de **structurer** la page :
 - En-tête** avec `<header>`.
 - Pied-de-page** avec `<footer>`.
 - Élément** de navigation avec `<nav>`.
 - Article** avec `<article>`.

Listes

- A
- B
- C

- ① A
- ② B
- ③ C

- | | |
|---|---|
| X | A |
| Y | B |
| Z | C |

```
<ul>
  <li>A</li>
  <li>B</li>
  <li>C</li>
</ul>
```

```
<ol>
  <li>A</li>
  <li>B</li>
  <li>C</li>
</ol>
```

```
<dl>
  <dt>X</dt> <dd>A</dd>
  <dt>Y</dt> <dd>B</dd>
  <dt>Z</dt> <dd>C</dd>
</dl>
```


Tableaux

```
<table>
  <tr>
    <th>X1</th>
    <th>X2</th>
  </tr>
  <tr>
    <td>A1</td>
    <td>A2</td>
  </tr>
  <tr>
    <td>B1</td>
    <td>B2</td>
  </tr>
</table>
```

X1	X2
B1	B2
C1	C2

Tableaux complexes

- `<caption>` Donner une **légende** au tableau.
- `<thead>` Délimiter l'**en-tête** du tableau (en-têtes de colonnes).
- `<tfoot>` Délimiter le **pied de page** du tableau (somme...).
- `<tbody>` Délimiter le **corps** du tableau.
- `<col>` Délimiter les **colonnes** :

```
<table>
  <colgroup>
    <col id="cle">
    <col span="2" id="valeurs">
  <tr>
    <td>K1</td>
    <td>Va1</td>
    <td>Vb1</td>
  </tr>
  <!-- ... -->
</table>
```

Liens

Liens **hypertexte**, d'où **HTTP**, **HTML** !

```
<a href="http://www.telecom-paristech.fr/">Telecom</a>
```

Peut contenir un URL **absolu** (comme avant), mais aussi **relatif**.
Considérons la page `http://example.com/foo/bar.html`.

Lien	Résolution
/quux	<code>http://example.com/quux</code>
..	<code>http://example.com/</code>
bar2.html	<code>http://example.com/foo/bar2.html</code>
baz/toto.html	<code>http://example.com/foo/baz/toto.html</code>
#top	<code>http://example.com/foo/bar.html#top</code>

L'effet du **fragment** #top est de faire **défiler** jusqu'à l'élément portant l'attribut `id="top"`. Non transmis au serveur.

iFrames

- Afficher une autre page dans la page courante :

```
<iframe src="http://en.wikipedia.org/">  
  <p>Pas de support iFrame !</p>  
</iframe>
```

- Parfois pratique, mais **déconseillé** :

- Peut être **perturbant** pour l'utilisateur
- Pas toujours très **flexible**
- Complexe pour la **sécurité**

→ Préférer des solutions **côté serveur** (ou avec JavaScript)

Table des matières

- 1 Généralités
- 2 Structure
- 3 Texte, listes et tableaux
- 4 Multimédia**
- 5 Formulaires

Images

- Depuis la **première spécification** de HTML (1993) :
``
- src précise l'**URL** où chercher l'image (absolu ou relatif).
- alt indique un texte **obligatoire** pour remplacer l'image :
 - image non trouvée
 - malvoyants
 - navigateurs texte
 - robots

Note : l'image peut se trouver sur un **tout autre serveur** !

- Requêtes **multiples**.
- **Inline linking** et risques.
- **Vie privée ?** Pixel espion + Referer.

Aussi : `<figure>` et `<figcaption>` pour légender.

Formats d'image

BMP Aucune compression, historique, à éviter.

JPEG Compression appropriée aux photos.

WebP Alternative plus récente (2010) par Google.

PNG Compression sans perte adaptée aux dessins.

GIF Comme PNG, moins bien, mais support pour des images animées.

APNG PNG avec animation, alternative à GIF. Rare.

SVG Images vectorielles : rendu calculé à partir de formes géométriques, zoomables sans pixelisation.

Audio et vidéo

- Nouveau en **HTML5** :

```
<audio src="audio.ogg">
  <p>Pas de support audio :
  <a href="audio.ogg">fichier audio.ogg</a>.</p>
</audio>
```

controls Afficher des **contrôles** de lecture (pas par défaut!).

autoplay Lecture **automatique** (à éviter...).

loop Lecture en **boucle**.

preload Choisir le **préchargement** du fichier.

<source> Plusieurs **formats** alternatifs.

Élément **<video>** analogue pour la **vidéo**.

Formats audio

Complicqué ! Codecs versus conteneurs.

WAV. Sans compression, à éviter.

MID. Rendu logiciel d'une partition : ne pas confondre.

MP3. Format historique. Brevets, mauvaise qualité.

FLAC. Compression audio sans perte.

Speex. Compression audio optimisée pour la voix.

Ogg Vorbis. Alternative libre à MP3.

Opus. Alternative récente, IETF.

Formats vidéo

H.264. Format le plus répandu, mais problèmes de brevets.

Ogg Theora. Alternative libre, mais peu utilisée.

WebM. Format proposé par Google, libre. VP8 et VP9.

- Le standard HTML5 ne privilégie plus Theora.
- IE et Safari supportent uniquement H.264.
- Firefox, Chrome, Android supportent tout.
- **Controversé!** Le standard doit-il préférer les formats répandus ou les formats sans brevets ?

Table des matières

- 1 Généralités
- 2 Structure
- 3 Texte, listes et tableaux
- 4 Multimédia
- 5 Formulaires**

Bases

- Structure générale d'un formulaire :

```
<form action="action.php" method="get">  
  <!-- contenu du formulaire ici -->  
  <input type="submit">  
</form>
```

<input> Bouton pour **valider** le formulaire (ici).

action URL vers lequel **soumettre**.

method Quelle **méthode** utiliser pour la soumission.

enctype Comment **encoder** les données du formulaire.

- Le navigateur effectuera la requête demandée en fournissant les données des champs du formulaire.
- Traitement effectué par le **serveur**.

Retour à HTTP

- **GET** : les paramètres sont transmis dans le **chemin** :
`GET action.php?nom=Dupond&prenom=Jean HTTP/1.1`
`Host: example.com`
- **POST** et `application/x-www-form-urlencoded` (par défaut) : c'est pareil, mais les données sont dans le **corps** :
`POST action.php HTTP/1.1`
`Host: example.com`
`Content-Type: application/x-www-form-urlencoded`

`nom=Dupond&prenom=Jean`

Retour à HTTP (suite)

- **POST** et multipart/form-data : moins concis mais plus efficace pour de gros volumes :

```
POST action.php HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: multipart/form-data; boundary=--e06
```

```
----e06
```

```
Content-Disposition: form-data; name="nom"
```

```
Dupond
```

```
----e06
```

```
Content-Disposition: form-data; name="prenom"
```

```
Jean
```

```
----e06--
```

Champs texte

- Champ de base : le **champ texte** :

```
<form action="action.php" method="get">
  <label for="nom">Nom</label>
  <input name="lastname" id="nom" type="text"><br>
  <label for="prenom">Prénom</label>
  <input name="firstname" id="prenom" type="text">
  <input type="submit">
</form>
```

- L'élément `<label>` **explique** le champ et y est lié par son `id`.
- L'attribut `name` définit le **nom** dans la requête HTTP :

```
POST action.php HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
lastname=Dupond&firstname=Jean
```

Attributs du champ texte

autofocus Champ à **sélectionner** au chargement.

placeholder Texte **explicatif** affiché quand le champ est vide.

disabled Champ **non modifiable**. (Aussi readonly.)

autocomplete Tenter d'autocompléter avec valeurs **passées**.

value Contenu **par défaut**.

required Impose que le champ soit **rempli**.

type Multiples valeurs :

→ password (étoiles)

→ email

→ range avec attributs min et max

→ Autres : tel, number, color...

pattern Valider avec une **expression régulière** :

→ pattern=" [0-9]{5}" pour un code postal).

→ Validation plus complexe, compatibilité : **JavaScript**.

→ **Aucune garantie !** À **revalider** côté serveur.

Boutons radio

- Sélectionner parmi **plusieurs choix** :

```
<input type="radio" name="choix" id="oui" value="oui">  
<label for="oui">Oui</label><br>  
<input type="radio" name="choix" id="pe" value="pe">  
<label for="pe">Peut-être</label><br>  
<input type="radio" name="choix" id="non" value="non">  
<label for="non">Non</label><br>
```

- Partagent le même name mais ont des value **différentes**.
- **Non exclusifs** si name différents.

Cases à cocher

- Sélectionner des choix **non exclusifs** :

```
<input type="checkbox" name="lu" id="lu">  
<label for="lu">Lu</label><br>  
<input type="checkbox" name="approuve" id="approuve">  
<label for="approuve">Approuvé</label>
```

- Autre possibilité :

```
<input type="checkbox" name="c[]" value="lu" id="lu">  
<label for="lu">Lu</label><br>  
<input type="checkbox" name="c[]" value="ap" id="ap">  
<label for="ap">Approuvé</label>
```

- Attribut checked : coché **par défaut**.
- Le [] est pour indiquer que le nom contient **plusieurs valeurs**, nécessaire pour certains langages serveur (PHP...).

Listes déroulantes

- Sélectionner parmi une **liste** de choix :

```
<label for="affiliation">Affiliation</label>
<select name="affiliation" id="affiliation">
  <option value="ecp">École centrale de Paris</option>
  <option value="x">École polytechnique</option>
  <option value="ese">ESE</option>
  <option value="instn">INSTN</option>
  <option value="telecom">Télécom ParisTech</option>
  <option value="sud">Université Paris-Sud</option>
</select>
```

selected Sélectionné par défaut.

size Liste de **hauteur fixe** (non déroulante).

multiple Sélectionner **plusieurs éléments** (utiliser []).

<optgroup> Former des **groupes d'options**.

Upload de fichiers

- Joindre un **fichier** au formulaire :

```
<label for="fichier">Fichier à soumettre</label>  
<input type="file" name="fichier" id="fichier">
```

- **Impérativement** utiliser post et multipart/form-data !
- Spécifier le **type de fichier** attendu (MIME) avec accept.
- multiple pour **plusieurs fichiers**.
- **Pas de garanties !** À revalider.
- **Taille maximale** à imposer côté serveur.

Boutons

`<input type="submit">` Valide le formulaire.

`<input type="image" src="submit.png" alt="Submit">`
Valide le formulaire (image).

`<input type="reset">` Réinitialise le formulaire.

`<button>` Ne fait rien (JavaScript).

`value="texte"` Précise le texte du bouton.

Autres

`<fieldset>` Regrouper des champs par groupes, annotés avec `<legend>`.

`<datalist>` Spécifier des autocomplétions possibles.

`type="hidden"` Champs cachés :

- Forcer l'**ajout** d'un champ à la requête.
- Pourquoi faire ? Voir **plus tard**.
- À **revalider** côté serveur !

`<textarea>` Champ texte multilignes :

```
<textarea name="adresse" id="adresse" rows="3" cols="42">  
  (Valeur par défaut.)  
</textarea>
```