

HTML et XHTML.

Patrick Poulingeas.

16 mars 2005.

1. Introduction.

HTML (*HyperText Markup Language*) a été inventé par Tim Berners-Lee en 1990 alors qu'il effectuait des recherches au CERN (Centre Européen de Recherche Nucléaire). HTML est un élément fondamental du World Wide Web, conçu lui aussi par Berners-Lee. Le Web a pour objectif de relier les différentes ressources d'Internet à l'aide de documents hypertextes. Auparavant, on disposait de serveurs Gopher présentant leurs ressources sous une forme arborescente. Malgré l'existence de moteurs de recherche Gopher, l'exploitation efficace d'informations sur Internet restait délicate car des documents portant sur une même thématique n'étaient pas reliés entre eux.

HTML est le langage avec lequel sont écrits les documents hypertextes qui sont au cœur du Web. Il s'agit d'un langage de type déclaratif : le concepteur d'une page Web décrit celle-ci avec HTML qui met à sa disposition pour cela différents types de balises (*tags*). C'est au navigateur à analyser et produire un affichage correct correspondant à la description HTML.

D'un point de vue syntaxique, HTML se conforme aux règles de SGML (*Standard Generalized Markup Language*). SGML est un méta-langage permettant de définir des langages structurés par des balises. La spécification de HTML est donc décrite à l'aide de SGML.

Les documents HTML sont des documents au format texte, composés d'éléments qui sont structurés par des balises. Les balises peuvent avoir des options que l'on appelle « attributs ». Quand on examine l'ensemble des balises proposées par HTML, on note qu'il y a un mélange entre des balises liées à la structure du document et des balises liées à mise en forme du document. Le mélange obtenu rend difficile l'extraction d'informations d'une page HTML (à partir d'un programme par exemple). Dans l'esprit de Berners-Lee, seules des balises en rapport avec la structure du document devaient intégrer HTML. Les balises de mise en forme supposent que le navigateur qui va afficher les pages HTML dispose de capacités d'affichage permettant de les prendre en charge. Or cette supposition est en contradiction avec l'universalité recherchée pour le Web. Ces balises de mise en forme ont été le plus souvent introduites par des navigateurs voulant mettre en avant les possibilités d'affichage « avancées » qu'ils possédaient.

Par ailleurs, un processus de normalisation était conduit par le W3C (World Wide Web Consortium), organisme dont Tim Berners-Lee est aujourd'hui le directeur. Les diverses normes produites ont été le résultat de consensus entre les principaux acteurs du Web. Elles ont contribué à limiter la prolifération des extensions HTML spécifiques à tel ou tel navigateur. La spécification courante est HTML 4.01. Il s'agit aussi de la dernière : HTML devant progressivement s'effacer devant XHTML (cf. chapitre 12) dont l'un des buts est de séparer rigoureusement contenu et présentation dans un document.

2. Structure générale d'un document HTML et remarques diverses.

Un document HTML suivant le standard 4.01 du W3C doit commencer par une déclaration de type de document qui sert à spécifier les balises autorisées et la grammaire que respecte le document (Cette exigence est issue de SGML).

Exemple :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Il y a 3 types de documents HTML normalisés :

- HTML Strict : c'est une spécification de laquelle ont été retirées des balises liées à l'aspect visuel d'une page et les balises liées aux cadres (*frames*). Il s'agit de la définition de type de document (DTD : *Document Type Definition*) la plus restrictive des trois.
- HTML Transitional : il s'agit d'une spécification conservant des balises permettant essentiellement des effets graphiques (balises ayant souvent été introduites par des navigateurs sans passer par un processus de normalisation). Cette norme doit permettre aux webmasters de faire progressivement une transition de leurs « vieilles » pages HTML à la norme HTML Strict. Cette spécification ne contient pas les balises en rapport avec la gestion des cadres.
- HTML Frameset : Il s'agit de la norme Transitional à laquelle les balises concernant les cadres ont été ajoutées.

Après avoir indiqué le type de document, la structure générale (simplifiée) d'un document HTML se présente sous la forme :

```
<html>
  <head>
    <title>
      Titre de la page
    </title>
  </head>
  <body>
    Corps du document HTML
    contenant les balises décrivant le document.
  </body>
</html>
```

Le titre indiqué s'affiche le plus souvent dans la barre de titre du navigateur (mais la norme HTML 4.01 n'oblige pas un tel comportement).

Remarques :

- Dans une partie textuelle d'un document HTML, plusieurs blancs (c'est-à-dire des espaces, des tabulations, des retours à la ligne) seront reproduits par le navigateur comme un seul espace (Les blancs suivant une balise n'étant même pas pris en compte). Les tabulations et les retours à la ligne servent donc essentiellement à rendre le source HTML plus lisible.

- Les balises HTML ne sont pas sensibles à la casse. <HEAD>, <Head> ou <head> sont valides pour la norme HTML 4.01.
- Un document HTML étant de type texte (au sens MIME), on a vu que selon la norme HTTP, le codage des caractères devrait être par défaut ISO-8859-1 (Latin1) pour le navigateur. On devrait donc pouvoir insérer des lettres accentuées comme « é » ou « à » directement dans le source HTML. Mais la norme pour HTML 4.01 est **Unicode** et non pas Latin1. De plus, le navigateur peut proposer à l'utilisateur de fixer le jeu de caractères utilisé pour l'affichage (*Affichage/Encodage des caractères* dans Firefox, *Affichage/Codage* dans Internet Explorer). Afin d'être certain d'obtenir les caractères que l'on désire, HTML définit des « entités de caractères ». Celles-ci commencent par le symbole « & » et se terminent par un « ; ». Le principe de construction de ces entités est résumé dans le tableau ci-dessous :

	Lettre de base	Modificateur	
&	a A	acute	accent aigu
	e E	grave	accent grave
	i I	circ	accent circonflexe
	o O	uml	tréma
	u U	cedil	cédille
	c C		
			;

Exemple : « carré » s'écrit « carré ».

- Il est possible d'insérer des commentaires dans une page HTML. Les commentaires en HTML prennent la forme suivante :

```
<!-- note -->
```

ou, sur plusieurs lignes :

```
<!-- remarquez
      bien cela -->
```
- De façon générale, les navigateurs sont **très** tolérants avec les erreurs contenues dans les pages HTML. Ils préfèrent afficher ce qu'ils peuvent « comprendre » du document plutôt que de produire des messages d'erreurs.

3. Balises permettant de structurer le corps d'un document HTML.

Les balises de cette section sont liées à la structure d'un document. Le style appliqué à ces balises dépend des navigateurs.

Les paragraphes sont placés entre les balises <p> et </p>.

Les titres des paragraphes sont encadrés par une paire de balises de la forme <h_n>...</h_n> avec n ∈ [1,6]. <h1>...</h1> correspond au titre de plus au niveau et <h6>...</h6> au titre de plus bas niveau. Le titre <h1>...</h1> sera généralement affiché avec une taille de fonte plus élevée que <h2>...</h2>, etc.

On dispose d'un certain nombre de balises correspondant à des attributs de texte logiques :

- `<em>...</em>` : mise en valeur « normale » (em : *emphasys*)
- `<strong>...</strong>` : mise en valeur « forte »
- `<cite>...</cite>` : citation
- `<samp>...</samp>` : exemple
- `<dfn>...</dfn>` : définition
- `<acronym>...</acronym>` : acronyme
- `<code>...</code>` : code source
- `<var>...</var>` : noms de variables

Un retour à la ligne est produit grâce à la balise : `<br>` (Il n'y a pas de balise de fermeture)

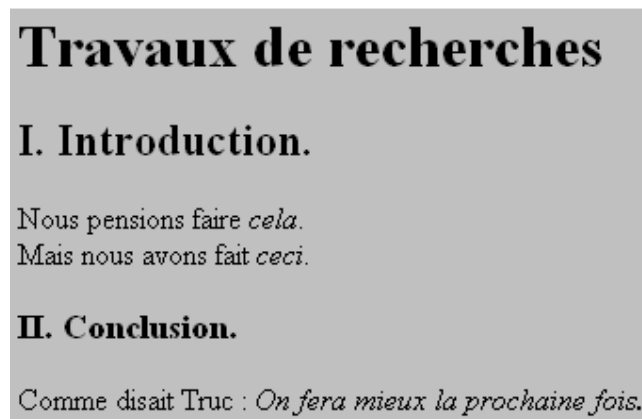
Exemple :

```
<h1>Travaux de recherches</h1>

<h2>I. Introduction.</h2>
<p>
    Nous pensions faire <em>cela</em>.<br>
    Mais nous avons fait <em>ceci</em>.<br>
</p>

<h3>II. Conclusion.</h3>
<p>
    Comme disait Truc :
    <cite>On fera mieux la prochaine fois.</cite>
</p>
```

Résultat obtenu :



Travaux de recherches

I. Introduction.

Nous pensions faire *cela*.
Mais nous avons fait *ceci*.

II. Conclusion.

Comme disait Truc : *On fera mieux la prochaine fois.*

Les balises permettant de positionner des attributs physiques (choix d'une fonte, couleur, etc.) pour une partie d'un document sont présentées au chapitre 11. Il est préférable de ne pas les mélanger avec les balises que nous venons de voir qui sont en rapport avec la structure logique du document. Pour les attributs physiques, il vaut mieux recourir aux feuilles de style (CSS) pour définir ceux-ci.

4. Insertion d'images dans un document HTML.

L'insertion d'images dans une page HTML s'accomplit grâce à la balise `<img>`.

L'attribut `src` permet d'indiquer l'URI de l'image qui peut être aux formats JPEG, GIF ou PNG (ainsi que d'éventuels autres formats pris en compte par le navigateur).

L'attribut `alt` permet de spécifier un texte qui apparaît si le navigateur ne peut afficher l'image (parce qu'il s'agit d'un navigateur texte par exemple). Ce texte peut aussi s'afficher comme une info-bulle. Selon la norme HTML 4.01, l'attribut `alt` est **obligatoire**.

On peut spécifier la largeur et la hauteur de l'image avec les attributs `width` et `height`. On indique soit un pourcentage de la fenêtre du navigateur (Exemple : `width="50%"`), soit un nombre de pixels (Exemple : `width="150"`). Si les dimensions de l'image ne sont pas indiquées, comme dans l'exemple ci-dessous, le navigateur les détermine automatiquement à partir des informations contenues dans le fichier. Si le chargement d'une page est très long, comme le navigateur aura affiché le texte avant l'image, cela pourra entraîner une modification de la mise en page une fois que le navigateur connaîtra les dimensions de l'image.

Exemple :

```

```

Résultat obtenu avec Internet Explorer 6.0 :



5. Liens hypertextes.

Un lien hypertexte, dont l'affichage peut être un texte ou une image, est une référence à une autre ressource Web. En cliquant sur le lien, l'internaute est conduit à cette autre ressource. Les liens hypertextes constituent donc la base du concept de navigation sur le Web.

Un lien se définit avec la balise `<a>` (a : *anchor*). L'attribut `href` de cette balise indique la ressource cible. Entre la balise d'ouverture `<a>` et celle de fermeture `</a>` se trouve le texte ou l'image qui est la représentation visuelle du lien pour l'internaute.

Exemple :

```
<a href="http://www.msi.unilim.fr/">
    Vers le site du LMSI
</a>
```

L'URI indiquée avec href peut aussi être relative à la page contenant le lien. Par exemple, depuis la page stockée à l'URL : <http://www.msi.unilim.fr/index.html>, le lien :

```
<a href="themes.html">
référence la ressource associée à l'URL : http://www.msi.unilim.fr/themes.html
```

La ressource désignée par href n'est pas nécessairement une ressource HTTP (puisque'il s'agit d'une URI).

Exemples :

```
<a href="ftp://ftp.lip6.fr/pub/python/2.4/Python-2.4.tgz">
    T&eacute;l&eacute;chargez la derni&egrave;re version
    de Python.
</a>
```

```
<a href="mailto:hegel@berlin-uni.de">
    Ecrivez moi pour obtenir des renseignements
    sur le Syst&egrave;me du Savoir Absolu.
</a>
```

Il est aussi possible de faire des liens référençant un endroit d'une même page HTML. Ce sont des liens locaux (ou internes) au document HTML.

Pour cela, on place des définitions d'ancres dans le document. Ces ancres servent de repères à l'intérieur du document. Elles disposent d'un attribut name qui permet de les distinguer les unes des autres.

Exemple :

```
<a name="intro">
    <h1>Introduction</h1>
</a>
```

Un lien hypertexte vers une ancre déclarée dans le même document prend la forme suivante : `<a href="#nomAncre">`.

Exemple :

```
<a href="#intro">Lien vers l'introduction</a>
```

En cliquant sur le lien ci-dessus, la page sera réaffichée à partir de l'ancre « intro ».

On peut aussi faire référence à une ancre dans un document distant.

Exemple :

```
<a href="cours.htm#intro">Lien vers l'introduction du cours</a>
```

L'attribut name peut être remplacé par l'attribut id. Cet attribut id peut de plus être utilisé pour définir une ancre avec d'autres balises. Ainsi :

```
<h2 id="section2">Section 2</h2>
```

crée une ancre dans le document, ancre qui peut ensuite être référencée par :

```
<a href="#section2">Retour &agrave; la section 2 du document</a>
```

6. Listes d'items.

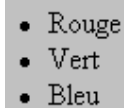
6.1. Listes non numérotées.

Pour obtenir une liste d'items non numérotés, on a recours aux balises `<ul>` et `<li>`.

Exemple :

```
<ul>
  <li>Rouge</li>
  <li>Vert</li>
  <li>Bleu</li>
</ul>
```

Résultat obtenu :



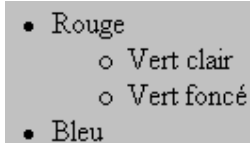
- Rouge
- Vert
- Bleu

On peut imbriquer des listes. Le style des puces employé peut alors varier.

Exemple :

```
<ul>
  <li>Rouge</li>
  <ul>
    <li>Vert clair</li>
    <li>Vert fonc&eacute;</li>
  </ul>
  <li>Bleu</li>
</ul>
```

Résultat :



- Rouge
 - Vert clair
 - Vert foncé
- Bleu

6.2. Listes numérotées.

Pour avoir une numérotation des listes, on emploie la balise `<ol>` à la place de la balise `<ul>`. L'imbrication des listes numérotées et leur mélange avec des listes non numérotées est possible.

Exemple :

```
<ol>
```

```

<li>Introduction</li>
<ul>
  <li>Bla bla </li>
  <li>Bla bla bla </li>
</ul>
<li>Conclusion</li>
</ol>

```

Résultat :

```

1. Introduction
   o Bla bla
   o Bla bla bla
2. Conclusion

```

6.3. Listes de termes avec définitions.

Des balises spécifiques permettent de définir une liste de termes. Ce sont `<dl>`, `<dt>` et `<dd>`. « dl » est l'acronyme de « *definition list* », « dt » celui de « *definition term* » et « dd » celui de « *definition description* ».

Exemple :

```

<dl>
  <dt>Amiga OS</dt>
  <dd>Système d'exploitation</dd>
  <dt>BCPL</dt>
  <dd>Langage de programmation</dd>
  <dt>68000</dt>
  <dd>Processeur</dd>
</dl>

```

Résultat obtenu :

```

Amiga OS
  Système d'exploitation
BCPL
  Langage de programmation
68000
  Processeur

```

7. Tableaux.

Les balises présentées dans cette section ne servent pas qu'à l'affichage de tableaux de données. Quand on n'utilise pas de feuille de style, les tableaux constituent l'un des moyens pour obtenir une mise en page d'une qualité convenable. Cette technique a souvent été employée par les webmestres, mais cette façon de procéder contribue à assimiler des éléments de structuration du document (les tableaux) à des éléments de présentation. Il vaut donc mieux dorénavant faire appel aux feuilles de style.

Un tableau se définit à l'intérieur d'un bloc `<table>...</table>`. L'attribut optionnel `border` permet de spécifier une bordure pour le tableau, bordure dont la taille est spécifiée en pixels.

Les lignes se définissent avec `<tr>` et `</tr>` (tr : *table row*). A l'intérieur d'une ligne, on crée des colonnes avec `<td>` et `</td>` (td : *table data*).

Exemple avec un tableau à 3 lignes et 2 colonnes :

```
<table border="1">
  <tr> <td>(1,1)</td> <td>(1,2)</td> </tr>
  <tr> <td>(2,1)</td> <td>(2,2)</td> </tr>
  <tr> <td>(3,1)</td> <td>(3,2)</td> </tr>
</table>
```

Résultat obtenu :

(1,1)	(1,2)
(2,1)	(2,2)
(3,1)	(3,2)

Il est possible d'ajouter une légende au tableau, légende placée au-dessus ou au-dessous du tableau. Ceci s'accomplit avec le couple `<caption>...</caption>` placé à l'intérieur des balises `<table>` et `</table>`. L'attribut `align` de `caption`, pouvant valoir « top » (par défaut) ou « bottom », indique la position de la légende.

Il est de plus possible d'introduire des légendes de colonnes. Elle interviennent dans la définition de la 1^{ère} ligne du tableau, avec les balises `<th>` (th : *table header*) et `</th>` à la place de `<td>` et `</td>`. Un style particulier est appliqué par le navigateur à ces légendes pour qu'elles se distinguent des autres valeurs contenues dans le tableau.

Exemple :

```
<table border="1">
  <caption>Années de naissance</caption>
  <tr> <th>Nom</th> <th>Année</th> </tr>
  <tr> <td>Dupont</td> <td>1964</td> </tr>
  <tr> <td>Durand</td> <td>1960</td> </tr>
</table>
```

Résultat obtenu :

Années de naissance	
Nom	Année
Dupont	1964
Durand	1960

Les cellules d'un tableau peuvent contenir du texte, des images, des vidéos, d'autres tableaux, etc.

Exemple :

```
<table>
  <tr>
    <td>
      
    </td>
    <td>
      
    </td>
  </tr>
</table>
```

Résultat obtenu :



On peut procéder à une fusion de lignes ou de colonnes dans un tableau. On dispose pour cela des attributs rowspan et colspan pour les balises <td> et <th>.

Exemples :

On a un tableau de 3 lignes et 2 colonnes dont veut fusionner les lignes constituant la première colonne. On va donc employer l'attribut rowspan :

```
<table border="1">
  <tr> <td rowspan="3">La</td>    <td>Chartreuse</td> </tr>
  <tr>                                <td>de</td>          </tr>
  <tr>                                <td>Parme</td>        </tr>
</table>
```

Affichage obtenu :

La	Chartreuse
	de
	Parme

On dispose maintenant d'un tableau de 2 lignes et 2 colonnes dont on souhaite fusionner les colonnes formant la seconde ligne. C'est l'attribut `colspan` que l'on va à présent utiliser :

```
<table border="1">
  <tr> <td>Mort</td> <td>&agrave;</td> </tr>
  <tr> <td colspan="2">Cr&eacute;dit</td> </tr>
</table>
```

Affichage obtenu :

Mort	à
Crédit	

La taille d'une cellule est calculée de façon automatique par le navigateur en tenant compte de son contenu. Il est cependant possible de forcer le navigateur à donner une taille à une cellule. Les attributs `height` et `width` indiquent la hauteur et la largeur souhaitées, soit en pixels, soit en pourcentage de l'espace libre dans la fenêtre d'affichage du navigateur. Si l'on fixe la hauteur d'une cellule, toutes les autres cellules de la même ligne auront la même hauteur. Respectivement, si l'on fixe la largeur d'une cellule, toutes les autres cellules de la même colonne auront la même largeur. Les attributs `height` et `width` peuvent être appliqués aux balises `<td>` et `<th>`.

La taille d'un tableau est déterminée par le navigateur à partir de la taille des cellules. On dispose cependant des attributs `height` et `width` pour la balise `<table>` afin d'indiquer une hauteur et une largeur au tableau.

L'alignement des données à l'intérieur des cellules se fait grâce aux attributs `align` (pour l'alignement horizontal) et `valign` (pour l'alignement vertical).

Exemple :

```
<table border="1">
<tr> <td rowspan="3" valign="top" >La</td>      <td>Chartreuse</td> </tr>
<tr>                                           <td align="right" >de</td> </tr>
<tr>                                           <td>Parme</td> </tr>
</table>
```

Résultat :

La	Chartreuse
	de
	Parme

Les valeurs possibles pour l'attribut `align` sont : « left » (valeur par défaut), « center » et « right ». Les valeurs possibles pour l'attribut `valign` sont : « top », « middle » (valeur par défaut) et « bottom ».

8. Formulaires.

Les formulaires sont des éléments fondamentaux pour l'interactivité d'un site Web. Ils permettent aux internautes d'envoyer des données par e-mail ou au serveur Web. Fréquemment, le serveur effectue un traitement de ces données avec un script s'appuyant sur la spécification CGI (*Common Gateway Interface*) et renvoie un document (HTML ou autre) comme résultat de son traitement.

Un formulaire est délimité dans le code HTML par les balises : `<form>` et `</form>`. La balise `<form>` dispose des attributs suivants :

- `name` : pour une gestion du formulaire avec JavaScript
- `action` : indique l'URL ou l'adresse e-mail où sont envoyées les données
- `method` : choix entre les méthodes GET ou POST
- `enctype` : type MIME indiquant l'encodage des données du formulaire

Exemples :

```
<form name="donneesClient"
      action="cgi/verifClient.pl"
      method="GET">
  <!-- Contenu du formulaire -->
</form>
```

```
<form name="saisieDonnees"
      action="mailto:machin@domaine.org"
      method="POST"
      enctype="text/plain">
  <!-- Contenu du formulaire -->
</form>
```

Remarque : on peut aussi utiliser l'attribut `target` dans la balise `<form>` pour indiquer un cadre de destination qui contiendra le résultat de la réponse à un GET ou un POST.

Diverses balises permettent ensuite de composer le formulaire :

- `<input type="text" ...>`
permet d'introduire une zone de saisie de texte sur une ligne.
L'attribut `size` permet de spécifier la taille du champ (tel qu'il apparaît à l'écran) en nombre de caractères.
L'attribut `maxlength` indique le nombre de caractères maximal que l'internaute peut entrer.
L'attribut `value` donne une valeur par défaut au champ de saisie.
L'attribut `name` lui affecte un identifiant.
- `<input type="password" ...>`
est similaire à la balise précédente, sauf que les données fournies par l'internaute sont masquées par des astérisques.
- `<textarea ...>` Texte affiché; par défaut, faut `</textarea>`
permet d'avoir un champ de saisie multi-lignes.
L'attribut `cols` indique le nombre de colonnes du champ, l'attribut `rows` le nombre de lignes. En cas de dépassement de ces valeurs, des ascenseurs sont automatiquement ajoutés.

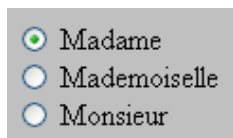
L'attribut name sert, comme précédemment, à donner un identifiant à la zone de saisie.

- `<input type="radio" ...>`
permet d'introduire un bouton radio dans le formulaire.
L'attribut name est commun à même groupe de boutons radio. C'est ce qui permet de regrouper des boutons radio. Cet attribut prendra la valeur associée au bouton finalement sélectionné.
L'attribut value fournit la valeur qui sera affectée à l'identifiant indiqué par l'attribut name si le bouton est choisi.
L'attribut checked, quand il est présent et égal à « checked », signale quel bouton est sélectionné par défaut dans un groupe de boutons radio.

Exemple :

```
<input type="radio"
      name="qualite" value="Mme" checked="checked">
Madame<br>
<input type="radio"
      name="qualite" value="Mlle">
Mademoiselle<br>
<input type="radio"
      name="qualite" value="M.">
Monsieur<br>
```

Affichage obtenu :

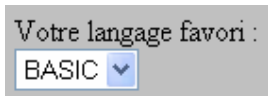


- `<input type="checkbox" ...>`
sert à introduire des cases à cocher.
Les attributs sont les mêmes que pour un bouton radio.
- `<select ...>...</select>`
permet de faire intervenir une liste de choix.
L'attribut name fournit un identifiant à la liste.
Chaque élément de la liste est indiqué avec la balise `<option>` qui a comme attribut value (la valeur assignée à l'identifiant de la liste si l'élément est choisi par l'internaute).

Exemple :

```
Votre langage favori :<br>
<select name="langage">
  <option value="BASIC">BASIC</option>
  <option value="Forth">Forth</option>
  <option value="Lisp">Lisp</option>
</select>
```

Affichage obtenu :



- `<input type="hidden" ...>`
Il s'agit d'un champ qui sera envoyé quand l'internaute soumettra le formulaire au serveur mais qui n'apparaît pas à l'affichage.
Il dispose des attributs name et value.
C'est un moyen de fournir des informations au serveur (pour maintenir une session par exemple) sans que l'internaute soit gêné par un affichage « parasite ».
- `<input type="submit" ...>`
Il s'agit du bouton de soumission du formulaire sur lequel clique l'internaute après avoir rempli celui-ci.
L'attribut value fournit le texte s'affichant sur le bouton.
- `<input type="reset" ...>`
introduit un bouton permettant de réinitialiser le contenu du formulaire.
Il dispose d'un attribut value ayant la même fonction que pour le bouton « submit ».

9. Cadres (*frames*).

Les cadres (en anglais : *frames*) permettent de diviser la fenêtre du navigateur en différentes zones. Chacune des zones recueille une page HTML susceptible d'être remplacée (à la suite d'un clic sur un lien hypertexte par exemple).

La technique de composition d'un site à base de frames a été souvent employée pour avoir une zone contenant un menu et une autre correspondant à la navigation dans les pages du site.

Le mécanisme des frames a le défaut de pouvoir être contourné en indiquant l'URL d'une page associée à un cadre. Un moteur de recherche peut ainsi fournir comme résultat l'URL d'un cadre, ce qui fait qu'en suivant ce lien l'internaute ne bénéficiera pas de l'architecture complète du site.

Afin de pallier à ce défaut, les webdesigners ont eu recours aux tableaux pour structurer la mise en page de leurs sites. La mise en page définie par les tableaux est généralement recopiée dans chacune des pages du site. Outre le fait que les tableaux sont des éléments de structuration et non de gestion de la présentation d'un document, cette copie de code HTML rend délicate la modification du style du site.

L'usage des tableaux comme éléments de mise en page devrait lui aussi s'achever avec le recours aux feuilles de style.

Pour employer des cadres, on crée une page principale servant à définir ceux-ci. Cette page n'accepte que des balises associées aux cadres. Elle prend la forme suivante :

```
<html>
  <head>
    <title>
      Titre significatif
    </title>
  </head>
  <body>
    <frameset ...>
```

```

        <!-- Définition des frames -->
    </frameset>
</body>
</html>

```

La balise <frameset> dispose d'attributs permettant de définir les cadres, leurs tailles et le fait qu'ils soient le résultat d'un partage horizontal ou vertical de la fenêtre du navigateur.

Exemples :

<frameset cols="30%,70%">	→ 2 cadres séparés verticalement
<frameset cols="10%,10%,80%">	→ 3 cadres séparés verticalement
<frameset rows="20%,80%">	→ 2 cadres séparés horizontalement
<frameset rows="30%,2*,1*">	→ « 2* » signifie que le 2 ^{ème} cadre occupera les 2/3 de l'espace restant, et « 1* » signifie que le 3 ^{ème} cadre occupera le tiers de l'espace restant.

Les balises <frame> servent ensuite à associer une page HTML originelle à un cadre et à donner un nom à ce cadre.

Exemple :

```

<frameset cols="20%,80%">
    <frame name="menu" src="menu.html">
    <frame name="article" src="introduction.html">
</frameset>

```

Dans l'exemple ci-dessus, la page menu.html contiendra probablement des liens vers des articles contenus sur le serveur. Si les liens ont la forme :

```
<a href="article1.html">Article 1</a>
```

la page de l'article s'affichera dans la frame du menu (ce qui n'est certainement pas l'effet souhaité). Pour que les pages HTML correspondant aux articles apparaissent dans l'autre frame, on va utiliser l'attribut target et le nom de la frame dans laquelle on désire que l'affichage ait lieu :

```
<a href="article1.html" target="article">Article 1</a>
```

On peut imbriquer des cadres.

Exemple :

```

<frameset cols="20%,80%">
    <frame name="menu" src="menu.html">
    <frameset rows="50%,50%">
        <frame name="articleHaut" src="entreeHaut.html">
        <frame name="articleBas" src="entreeBas.html">
    </frameset>
</frameset>

```

On peut aussi avoir des cadres locaux à une page HTML avec l'élément iframe.

Exemple :

```

<table border="2">
<tr>
    <td>

```

```

<iframe src="http://www.msi.unilim.fr/" width="250" height="250">
</iframe>
</td>
<td>
<iframe src="http://www.google.fr/" width="250" height="250">
</iframe>
</td>
</tr>

<tr>
<td>
<iframe src="http://www.alternatives87.eu.org/"
width="250" height="250">
</iframe>
</td>
<td>
<iframe src="http://www.python.org/" width="250" height="250">
</iframe>
</td>
</tr>
</table>

```

Résultat obtenu :



10. Méta-données d'un document HTML.

Il est possible de spécifier des informations concernant une page HTML (nom de l'auteur de la page, etc.) avec des balises <meta>. Ces balises sont généralement placées entre

les balises <head> et </head> du document. Ces méta-données peuvent être prise en compte par les robots indexeurs des moteurs de recherche.

Une balise <meta> a la forme suivante :

```
<meta name="identifiantInformation" content="information">
```

L'information sur le document HTML est identifiée par l'attribut name ; le contenu associé à cette information est fourni par l'attribut content.

Exemple :

```
<meta name="author" content="Karl Popper">
```

La norme HTML 4.01 ne définit aucun standard pour les noms et les contenus de la balise <meta>. Plusieurs balises de méta-données peuvent être présentes dans le document.

On peut aussi spécifier la langue utilisée par l'information de l'attribut content.

Exemple :

```
<meta name="keywords" lang="fr" content="Epistemologie, Falsifiabilité">
```

```
<meta name="keywords" lang="en" content="Epistemology, Falsifiability">
```

La balise <meta> a aussi un intérêt plus « pratique ». Elle permet d'insérer un en-tête HTTP dans la réponse du serveur. Pour cela, on substitue l'attribut http-equiv à l'attribut name.

Exemple :

```
<meta http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27 GMT">
```

a pour conséquence l'ajout de l'en-tête suivant à la réponse du serveur Web :

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

Remarque : l'en-tête HTTP ci-dessus peut être utile à des systèmes de cache (*proxy* par exemple) pour déterminer quand ils ont besoin de redemander un document.

Autre exemple :

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

11. Balises de mise en forme d'une page HTML.

Les balises présentées dans cette section permettent d'imposer au navigateur un style d'affichage pour une partie d'un document HTML. Avec l'introduction des feuilles de style, ces balises devraient progressivement disparaître.

Pour mettre en valeur une partie d'un texte, il est préférable d'utiliser la balise plutôt que la balise <i> qui permet de passer en italique. La mise en valeur est une propriété logique du texte, alors que le passage en italique est une propriété physique dont la signification est dépendante du contexte dans lequel on l'emploie. La reconnaissance et le traitement des informations contenues dans une page HTML sont facilités par l'usage **exclusif** de balises concernant la structuration du document.

Ajoutons que le navigateur ne dispose pas forcément des moyens d'assurer le style que l'on exige de lui avec des balises comme <i> (Pensez à un navigateur en mode texte ou s'exécutant sur un téléphone portable). Avec une balise comme , le navigateur choisit le style qu'il estime le plus approprié pour mettre en valeur le texte en question.

Nous pourrions introduire des balises comme ou des attributs comme bgcolor pour la balise <body>, mais ce sont des éléments du langage indiqués comme « *deprecated* » (désapprouvés) par la norme HTML 4.01. Pour des raisons de compatibilité avec les

précédentes versions de HTML, les navigateurs continuent à interpréter ces éléments mais il n'est pas recommandé de continuer à les employer.

Liste de balises influant sur le style de la fonte utilisée :

- `<i>...</i>` : met le texte entre ces balises en italique
- `<b>...</b>` : met le texte entre ces balises en gras
- `<small>...</small>` : utilise une fonte de « petite » taille pour le texte
- `<big>...</big>` : utilise une fonte de « grande » taille pour le texte

Insertion d'une ligne horizontale :

On utilise la balise `<hr>` (Elle n'a pas de balise de fin associée).

12. XHTML.

XHTML (*eXtensible HyperText Markup Language*) est l'évolution du HTML préconisée par le W3C. De façon très simplifiée, on peut voir XHTML comme une reformulation en XML (*eXtensible Markup Language*) de HTML, XML étant une sorte de SGML allégé. Les buts principaux de XHTML sont :

- de séparer plus clairement que ne le fait HTML contenu et forme d'un document,
- d'utiliser les outils standards pour éditer et travailler sur des documents XML,
- de permettre un traitement automatisé plus simple des documents, notamment avec l'aide des outils XML (parsers, etc.),
- de rationaliser et faciliter les extensions du langage.

Il y a aujourd'hui deux normes concernant XHTML : XHTML 1.0 et XHTML 1.1.

XHTML 1.0 regroupe en fait trois DTDs qui correspondent à celles de HTML : Strict, Transitional et Frameset. XHTML 1.0 permet aux webmasters de migrer progressivement leurs sites en HTML vers XHTML.

XHTML 1.1 est conçu comme un ensemble de modules. XHTML Basic est un ensemble minimal de modules qui permettent un affichage même sur des appareils limités en capacités matérielles (comme les smartphones ou certains PDA). On peut ensuite adjoindre à XHTML Basic des modules étendant ses fonctionnalités ou en apportant de nouvelles. On trouve ainsi un module de script (pour le JavaScript), un module de style (pour les CSS), etc.

Les principales exigences de XHTML par rapport à HTML sont :

- Le document doit être bien formé (c'est-à-dire respecter les conventions d'un document XML : imbriquer correctement des éléments, etc.).
- Toutes les balises sont en minuscules (XML est sensible à la casse).
- Toute balise ouverte doit être fermée (Par exemple : pas de `<p>` sans `</p>` l'accompagnant). Une balise sans contenu, comme `<br>`, s'écrit `<br/>` (Pour des raisons de compatibilité avec les anciens navigateurs, on écrira plutôt `<br />`).
- Les attributs prennent toujours la forme : `nom="valeur"`. On ne tolère plus la présence d'un attribut sans valeur (Exemple : `checked`), et l'on doit écrire alors : `attribut="attribut"` (Exemple : `checked="checked"`).
- L'attribut `name` ne doit plus être employé. C'est l'attribut `id` qui est utilisé à sa place.

Ces exigences résultent du fait qu'un document HTML doit être un document XML valide (c'est-à-dire respectant la DTD qui lui est associée). Si les navigateurs continuent à être permissifs avec le code (X)HTML qu'ils traitent, les outils liés à XML ne le sont pas.

Exemple de document XHTML 1.0 suivant la DTD Transitional :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Redirection</title>
    </head>
    <body>
        <p>
            Ce site est maintenant situ&eacute; &agrave; :
            <a href="http://bidon.org/">bidon.org</a>.
        </p>
    </body>
</html>
```

Commentaires sur l'exemple précédent :

- La première ligne est caractéristique d'un document XML. UTF-8 est une norme d'encodage Unicode.
- xmlns sert à déclarer un espace de nom (*namespace*). Avec cette déclaration, il sera appliqué par défaut à tous les éléments contenus par l'élément html.
- xml:lang et lang servent à spécifier la langue d'un élément. Le premier attribut est prioritaire.