INTRODUCTION AU XHTML

OBJECTIF DU COURS

L'objectif du cours est de présenter les **bases de la conception de documents structurés avec XHTML**
NB : Ce cours est lié au cours sur les feuilles de style CSS.
Des connaissances de base du HTML et/ou du XHTML peuvent être utiles, ainsi qu'une première pratique de la composition de documents Web. Toutefois, des rappels sont faits sur les principes de base et un débutant peut profiter de l'exposé pour initier sa pratique en étant en phase avec les standards actuels.
Ce cours représente également un socle utile avant d'aborder **HTML5**.

PRÉSENTATION DES LANGAGES

HTML

HTML : HyperText Markup Language
Langage de marquage hypertexte qui sert principalement à la structuration de documents dédiés aux sites Web.
Il s'agit d'un langage au format **texte**, enrichi de **balises** (ou **tags**, ou **marqueurs**).
Les fichiers contenant du code HTML reçoivent l'extension **.html** ou .htm

Le HTML est un standard sous l'autorité du **W3C** (**World Wide Web Consortium**).
http://www.w3.org
Il s'agit d'un langage **non propriétaire**.

Il s'agit d'un standard ancien.
Le point de départ peut être fixé l'année au cours de laquelle Tim Berners-Lee propose un document nommé Hypertext and CERN, en 1989.
Le premier « navigateur » daterait de 1990.
En juin 1993, il existait environ 130 sites Web !

HTML a connu plusieurs évolutions :
HTML 1.0 : ?
HTML 2.0 : 1995 (1ere spécification par IETF Internet Engineering Task Force)
HTML 3.0 : 1995 (W3C – Voir ci-après)
HTML 3.2 : 1996
HTML 4.0 : 1997
HTML 4.01 : 1999
HTML5 : ???

XHTML

XHTML : eXstensible HyperText Markup Language
Très proche de HTML, XHTML est une reformulation de celui-ci en XML. C'est à dire que XHTML est un langage « appartenant à » XML, au même titre que d'autres langages comme MathML (formules mathématiques), SVG (description de figures)…
Il répond aux mêmes principes que HTML (hiérarchies de balises), mais dans une **écriture beaucoup plus rigide (ou rigoureuse)**.
Comme pour le HTML, les fichiers contenant du XHTML reçoivent l'extension **.html** ou htm.

Le XHTML est un standard sous l'autorité du **W3C** (**World Wide Web Consortium**).
http://www.w3.org
Il s'agit d'un langage **non propriétaire**.

Recommandation XHTML 1.0 : 2000 (révisée en 2002)
Recommandation XHTML 1.1 : 2001

HTML5

Nouvelle spécification du langage, encore en statut de brouillon, mais qui commence à être implémenté partiellement par les navigateurs et utilisé par les concepteurs. Nous y reviendrons dans une extension du cours, dédiée à HTML5.

BASES DES LANGAGES HTML & XHTML

Balises (ou tags ou marqueurs)

XHTML, comme HTML est un langage de **balisage**.
Les balises permettent de **définir ou délimiter des éléments dans le document**.
Les balises sont encadrées par des chevrons ouvrant et fermant **<** et **>**.

Elles se présentent :

- Par **paires**, une balise ouvrante étant suivie d'une balise fermante :
  `<p>Un paragraphe</p>`
  `<h1>Titre de niveau 1</h1>`
  NB : le marqueur fermant est identique au marqueur ouvrant, précédé d'un /
- Comme des **balises uniques** (ou **simples**, ou « **vides** »)
  `<br />` (retour à la ligne)
  `<hr />` (ligne de séparation horizontale)
  Les balises simples sont fermées en XHTML (d'ou le /> en fin de balise).

A noter :

- Les noms des balises ne sont pas libres, mais fixés par la grammaire du langage.
- Les balises sont écrits en minuscules.

Remarque :
De nombreuses balises du HTML sont obsolètes en XHTML (voire proscrites dans la syntaxe stricte du XHTML), par exemple <font>, , <u>…

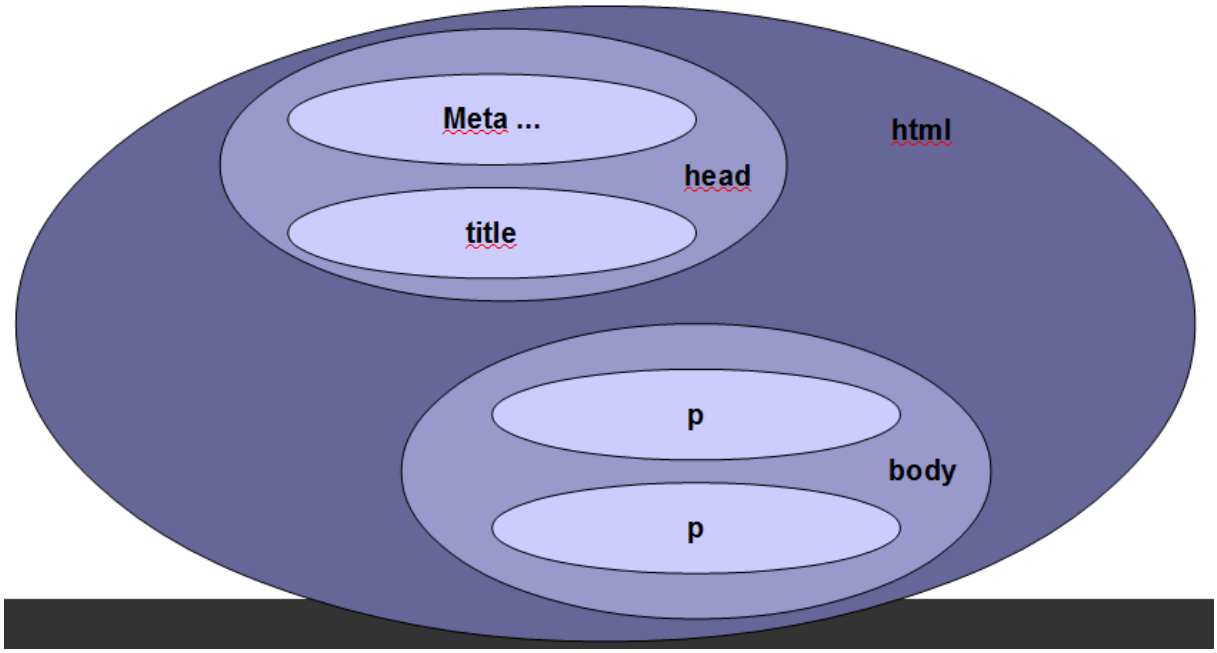
Liste des éléments :

- http://www.la-grange.net/w3c/html4.01/index/elements.html

L'imbrication des éléments

Un document XHTML est structuré selon une **hiérarchie d'éléments**, comparable à une arborescence (dont la racine est la balise <html>). L'image des « poupées russes » peut également illustrer la structure d'imbrication des éléments.

Exemple de hiérarchie d'éléments (incomplète) :



Exemple de document XHTML simple : Hello World

```
<?xml version="1.0" encoding="UTF-8"?>
```

Prologue

Nous détaillerons ces points ultérieurement

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
```

Doctype

```
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="fr" lang="fr">
```

Ouverture de la balise html

```
<head>
 <meta http-equiv="Content-Type"
content="text/html; charset= UTF-8"/>
 <title>Hello</title>
</head>
```

En-tête

```
<body>
<p>Salut tout le monde</p>
</body>
```

Corps du document
Contenant un paragraphe

```
</html>
```

Fermeture de la balise html

Les attributs

Les marqueurs acceptent des **attributs** (ou propriétés) :
Les attributs sont des **paires nom="valeur"**.
Les attributs sont placés dans les balises ouvrantes (jamais les balises fermantes) ou les balises uniques.
Les valeurs d'attributs sont encadrées par des apostrophes doubles (le plus courant) ou simples :

Exemples :
```
<h1 id="untitre">paragraphe</h1>
```
ou
```
<p class="paragraphespecial">paragraphe</p>
```

Remarque :
De nombreux attributs du HTML sont obsolètes en XHTML (voire proscrits dans la syntaxe stricte du XHTML), ainsi qu'en HTML strict, par exemple face, color, align, valign, bgcolor, border…

Listes des attributs :

- http://www.la-grange.net/w3c/html4.01/index/attributes.html

Les commentaires

Il est possible (et conseillé) d'inclure des commentaires (non affichés par les navigateurs) dans le code XHTML.
Les commentaires sont encadrés par **< !--** et **-->**

Exemple :
```
< !-- Un commentaire -->
```

RÈGLES DE SYNTAXES OBLIGATOIRES EN XHTML

Le chevauchement des balises est illégal

```
<h1>Texte <i>sans chevauchement</i></h1>
```
est valide.

~~`<h1>Texte <i>avec chevauchement</h1></i>`~~
n'est pas valide.

Toutes les balises doivent être fermées

Toutes les balises doivent être fermées, même les balises vides.

```
<p>un paragraphe</p>
<p>un autre paragraphe</p>
<p>encore un paragraphe</p>
```
est valide.

~~`<p>un paragraphe`~~
~~`<p>un autre paragraphe`~~
~~`<p>encore un paragraphe`~~
n'est pas valide.

```
<img src="une_image.jpg" width="32" height="32" alt="Mon image" />
```
est valide.
Noter la barre / à la fin de la balise img (la balise est donc fermée). L'espace précédant / est utile pour les navigateurs conçus avant XHTML, ne gène en rien les navigateurs actuels et ne remet pas en cause la validité du code.

~~`<img src="une_image.jpg" width="32" height="32" alt="Mon image">`~~

n’est pas valide.

XHTML est sensible à la casse

Tous les noms d’éléments et d’attributs doivent être saisis en minuscules (ceci est facultatif pour les valeurs d’attributs et bien sûr le contenu).

```
<img src="uneImage.jpg" width="32" height="32" alt="Mon image" />
<img src="une_image.jpg" width="32" height="32" alt="Mon image" />
```

sont valides

~~<IMG SRC="une_image.jpg" WIDTH="32" HEIGHT="32" ALT="Mon image" />~~

n’est pas valide

Attention au code produits par les éditeurs WYSIWYG notamment (voir les « onMouseOver » ou autres déclencheurs JavaScript par exemple).

Les valeurs d’attributs sont toujours spécifiées entre apostrophes simples ou doubles (le plus courant)

```
<img src="une_image.jpg" width="32" height="32" alt="Mon image" />
```

est valide.

~~~~

n’est pas valide.

NB : lorsqu’une balise comporte plusieurs attributs, ceux ci sont séparés d’un espace.

Tout attribut nécessite une valeur

```
<input type="checkbox" name="rouge" id=”elt1” value="ok"
checked="checked">
```

est valide.

~~<input type="checkbox" name="rouge" id=”elt1” value="ok" checked>~~

n’est pas valide.

L'attribut name doit être remplacé par l'attribut id (identifiant)

```
<form id="ok">...</form>
```

est valide

~~<form name="pasOK">...</form>~~

n’est pas valide

Remarque : Pour les éléments de formulaire qui conservent leur attribut name, il est fréquent de faire cohabiter les attributs name et id.

Les éléments script et style sont déclarés comme possédant un contenu de données textuelles analysées (PCDATA : Parsed Character DATA)

```
<script language="javascript" type="text/javascript">
<![CDATA[
var val = 100 + 50;
]]>
</script>
```

On ne met pas de tiret double à l'intérieur d'un commentaire

Les tirets doubles n'apparaissent qu'au début et à la fin d'un commentaire.

```
<!-- Un commentaire -->
<!--============-->
```
sont valides.

~~<!-- Un autre --- commentaire -->~~
~~ou~~
~~<!----------------------->~~
ne sont pas valides.

TYPOLOGIE DES ÉLÉMENTS

Les éléments répondent à différents types différents de rendu (ou d'affichage) dans le navigateur :

- block
- inline
- list-item
- inline-block
- table
- table-cell
- table-row
- none (pas d'affichage → retrait du flux du doucement)
- d'autres type de rendu existent...

L'appartenance à un type de rendu dicte le comportement en terme de positionnement et d'affichage des éléments.
Remarque : le positionnement et l'affichage peuvent être revus en utilisant les feuilles de style CSS, dans certaines limites.

Eléments de type block

Présentation

Le rendu de type block fait référence (le nom l'indique) à des blocs à l'intérieur d'un document.
C'est le cas des paragraphes (…), des listes (
…) par exemple.

Caractéristiques standards :

- Ces éléments, par défaut, **apparaissent les uns en dessous des autres**.
- Ils présentent des **dimensions et des marges externes ou internes fixées par défaut** (redéfinissables avec les feuilles de style CSS), à l'exception des blocs .
- Ils sont **positionnables** (avec les feuilles de style CSS).

A noter :

- **Un bloc** (à l'exception des blocs de paragraphes et de titres <h1>, <h2>, … <h6> **peut contenir d'autres blocs**.
- **Un bloc peut contenir des éléments inline**.

Exemples d'élements de la « famille » block

Balise ouvrante	Balise fermante	Commentaires	Exemples
<blockquote>	</blockquote>	Citation longue. Les navigateurs présentent souvent le bloc avec une marge gauche.	Auteur <blockquote> Une citation d'un auteur, citation plutôt longue </blockquote>
		Division Cette balise ne propose aucun sens (en terme de sémantique, donc). C'est une division, un bloc, un regroupement d'autres éléments. Cette balise sert notamment à structurer les documents.	 Un paragraphe dans une division Un autre paragraphe
<dl>	</dl>	Liste de définitions La liste comprend des titres (<dt>…</dt>) et des éléments de définition (<dd>…</dd>).	<dl> <dt>Mot à définir</dt> <dd>Définition</dd> <dd>Suite de la définition</dd> <dt>Autre mot</dt> <dd>Autre définition</dd> <dd>Autre suite</dd> </dl>
<h1>, <h2>, … <h6>	</h1>, </h2>, … </h6>	Titres Il existe 6 niveaux de titres hiérarchiques. Rappel : les titres, bien qu'ils soient des blocs, ne peuvent pas contenir d'autres éléments de type bloc.	Grand titre Sous titre Sous sous titre (etc.)
		Liste ordonnée La liste comprend des éléments de liste (…)	 - Un élément bien ordonné - Un autre élément
		Paragraphe Rappel : les paragraphes, biens qu'ils soient des blocs, ne peuvent pas	Je suis un paragraphe assez court. Moi aussi, je suis un paragraphe.

		contenir d'autres éléments de type bloc.	
<table>	</table>	Tableau (table de données) Les tableaux sont divisés en lignes (<tr>…</tr>), elles même divisées en cellules d'en tête (<th>…</th>) ou de cellules « standards » (<td>…</td>)	<table> <tr> <td>cellule A</td> <td>cellule B</td> </tr> <tr> <td>cellule C</td> <td>cellule D</td> </tr> </table>
		Liste non ordonnée (liste à puces) La liste comprend des éléments de liste (...)	 - Un élément de liste à puce - Un autre élément de liste à puce

Les différents rendus de type block

display	Exemples d'éléments	Commentaires
block	h1, p, ul, div...	Rendu de type block standard
list-item	li	Rendu de type block standard, avec quelques spécificités (CSS : propriétés spécifique pour les puces : list-style...)
table	table	Rendu de type block, qui n'occupe que la largeur du contenu
table-row	tr	Rendu de type block, avec une répartition automatique de la hauteur des éléments au sein de l'élément parent

Liste des éléments de type bloc :

- http://htmlhelp.com/reference/html40/block.html

Eléments inline

Présentation

Les éléments en ligne servent à modifier, enrichir (...) des portions de textes, apporter du sens.
On dit également éléments « au fil du texte » ou « éléments internes ».

Il peut s'agir, par exemple, de créer un lien (<a…>…), de renforcer une portion de texte (…)…

Caractéristiques standards :

- Ces éléments, par défaut, apparaissent **au fil du texte**, par opposition aux éléments de type block, ils ne **sont pas** placés les uns au dessus des autres (ils restent à l'emplacement défini).
- Ils peuvent présenter des **dimensions fixées** (avec les attributs width et height), ils sont alors **définis comme « remplacés »** : c'est le cas des images (<img width="32" height="64" …>), des champs de saisie de formulaires (<input>), des champs de saisie multilignes des formulaires (<textarea>), des champs de sélection (listes à sélection unique / multiple) des formulaires (<select>), des objets comme les applets java ou le flash (<object>).
- Les autres éléments en ligne, définis comme **« non remplacés »** n'ont **pas de dimension** et occupent la **place nécessaire à leur contenu**.
- Les éléments en ligne **n'ont pas de marges internes ou externes** par défaut.
- Ils ne sont **pas** dédiés à un positionnement précis (même si cela est possible avec les CSS).

A noter :

- **Un élément inline ne peut contenir que des éléments en ligne** (donc pas de bloc).
- **Un élément inline doit être contenu dans un élément de type block**.

Exemples d'éléments de la « famille » inline

Balise ouvrante	Balise fermante	Commentaires	Exemples
<a>		Lien hypertexte L'attribut href précise la cible du lien	Autre page
		Met en emphase une portion de texte	Un paragraphe avec une portion importante.
<img>	rien	Inclusion d'une image L'attribut src indique le chemin vers le fichier image. L'attribut alt propose un texte alternatif (navigateurs en mode texte, malvoyants…)	
<q>	</q>	Citation courte (l'élément	Un paragraphe

		bloc blockquote est privilégié pour les citations longues)	`avec une <q> citation courte</q></p>`
`<samp>`	`</samp>`	Exemple	`<p>Un paragraphe avec un <samp>exemple</samp></p>`
`<span>`	`</span>`	Portion de texte Division qui ne propose pas de sens précis (comme div pour les blocs)	`<p>Un paragraphe avec <span>une portion particulière</span></p>`
`<strong>`	`</strong>`	Renforcement d'un morceau de texte.	`<p>Un paragraphe avec une <strong>portion renforcée</strong>.</p>`

Les différents rendus de type inline

display	Exemples d'éléments	Commentaires
inline	a, em, strong, span...	Rendu de type inline standard
inline-block	input, select	Rendu de type inline, les éléments pouvant être dimensionnés
table-cell	th, td	Rendu similaire au rendu inline-block, avec une répartition automatique de la largeur des éléments au sein de l'élément parent, une hauteur identique entre éléments frères et la possibilité d'utiliser la propriété vertical-align

Liste des éléments en ligne :

- http://htmlhelp.com/reference/html40/inline.html

Modifier le type de rendu

La propriété CSS **display** permet de passer d'un type de rendu à un autre.
La propriété display admet 18 valeurs, mais les plus couramment utilisées sont **présentées ci-dessus.**

Rendus spécifiques de formes tabulaires

display	Exemples d'éléments	Commentaires
table	table	Rendu de type block, qui n'occupe que la largeur du contenu
inline-table	table	Rendu de type inline, avec un comprtement de type table
table-row	tr	Rendu de type block, avec une répartition automatique de la hauteur des éléments au sein de l'élément parent
table-row-group	tbody	Regroupement d'une ou plusieurs rangées de cellules
table-header-group	thead	Comparable à table-row-group, mais l'élément est affiché avant les autres rangées et groupes de rangées
table-footer-group	tfoot	Comparable à table-row-group, mais l'élément est affiché après les autres rangées et groupes de rangées
table-column	col	Regroupement d'une colonnes de cellules
table-column-group	colgroup	Regroupement d'une ou plusieurs colonnes de cellules
table-cell	th, td	Rendu similaire au rendu inline-block, avec une répartition automatique de la largeur des éléments au sein de l'élément parent, une hauteur identique entre éléments frères et la possibilité d'utiliser la propriété vertical-align
table-caption	caption	Légende d'un tableau

LA STRUCTURE DES DOCUMENTS XHTML

Un document XHTML comprend plusieurs parties :

- Un prologue XML (facultatif, nous verrons pourquoi).
- Un DOCTYPE.
- Une déclaration de l'espace de noms et de la langue.
- Un en tête, précisant notamment le type d'encodage des caractères, présentant des méta données (voir ci après), ou faisant référence à d'autres fichiers.
- Un corps.

Remarque : HTML5 propose une simplification de certains éléments de structure.

Le prologue XML

Son rôle est de spécifier la **version de XML** et de préciser le **type d'encodage de caractères** employé.
Par exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Nous indiquons qu'ils s'agit de la version 1.0 de XML et que les caractères sont encodés en ISO-8859-1 (voir ci-après pour des précisions sur les encodages de caractères).

Remarque :
Le W3C conseille de débuter les documents XHTML par le prologue. Par contre, ce prologue n'est pas pris en charge par certains navigateurs et peut provoquer des fonctionnements non souhaités.
Certains choisissent de ne pas mentionner le prologue, l'encodage des caractères étant défini dans l'en-tête (voir ci-après).

Le DOCTYPE et les DTD XHTML

Le tag **DOCTYPE** précise la **DTD** (**Document Type Definition** ou **Définition de Type de Document**).
Elle est utile pour les navigateurs, les lecteurs de votre code ou les validateurs de code. Elle précise en quelque sorte la « grammaire » utilisée pour rédiger votre document.

Les différentes DTD XHTML

XHTML 1.0 Strict (formulation rigoureuse)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitionnel (formulation permissive autorisant l'emploi d'éléments HTML obsolètes)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset (formulation pour une utilisation des jeux de cadres / frames)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

DOCTYPES HTML

HTML 4.01 Strict (formulation rigoureuse)

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

HTML 4.01 Transitional (formulation permissive autorisant l'emploi d'éléments HTML obsolètes)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

HTML 4.01 Frameset (formulation pour une utilisation des jeux de cadres / frames)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">

La déclaration de l'espace de noms et de la langue

L'espace de noms et la langue du document sont indiqués dans la balise <html> du document.
Un espace de noms (namespace) est, en XML, un ensemble de types d'éléments et de noms d'attributs associés à une DTD.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
```

- Nous déclarons ici que l'espace de noms est celui proposé à l'URL suivant : "http://www.w3.org/1999/xhtml"
- Et la langue xml:lang="fr"
  NB : lang="fr" est supprimé en XHTML 1.1

L'entête

L'en-tête est délimité par les balises **<head>** et **</head>**
L'en-tête contient des informations non affichées par les navigateurs courants.

Le titre

Il fixe le titre du document.

```
<title>Conception de documents Web</title>
```

NB : le titre n'est pas directement affiché dans le document (souvent par le navigateur, dans l'interface).

Les méta données

Les méta données permettent de spécifier différentes informations à propos du document :

- Certaines méta données (les **mots clés** et la **description**) sont utilisées pour décrire le contenu du document et faciliter sa « visibilité » dans les moteurs de recherche, tout comme le **titre** (la question du positionnement dans les moteurs de recherche est toutefois beaucoup plus complexe).
- Certaines méta données permettent de donner des informations / instructions aux moteurs de recherche, par exemple leur indiquer s'il faut suivre les liens lors de l'indexation d'un site, la fréquence souhaitée pour la relecture des informations par les moteurs de recherche.
- D'autres méta données permettent de préciser le nom de l'auteur, la version…

```
Exemples :
<meta name="author" content="Jacques Bandet" />
<meta name="keywords" content="xhtml,web,……" />
<meta name="robots" content="ALL" />
Informations sur les méta données :
```

- http://peccatte.karefil.com/Software/Metadata.htm#Meta

Les références à d'autres ressources

L'en-tête permet de faire référence à d'autres ressources utilisées par le document : feuilles de style CSS, fichiers de scripts externes JavaScript par exemple.
Exemple :

```
<link rel="stylesheet" type="text/css" href="../style/main.css" />
```

(Ici on fait référence à une feuille de style située dans un autre répertoire)

Le type d'encodage des caractères

Par exemple :

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

Le type d'encodage indique le jeu de caractères utilisé par le document.
Pour un document en français on dispose des encodages suivants :

- iso-8859-1 : encodage classique pour les langues de l'Europe occidentale (aussi appelé Latin-1).
- iso-8859-15 : même encodage comportant quelques caractères supplémentaires comme le signe €…
- utf-8 : encodage pour les caractères de la majorité des langues mondiales.

Pour en savoir plus :

- http://openweb.eu.org/articles/jeux_caracteres/

A noter :
En utilisant l'encodage iso-8859-1 (le plus couramment utilisé, actuellement, pour les documents français) ou iso-8859-15 les caractères ASCII 7-BIT (codes 32 à 127) sont valides, avec 4 exceptions.
Ces exceptions sont codées avec des entités :
" (codé ")
& (codé &)
< (codé <)
> (codé >)
Remarque : ce sont des caractères utilisés par le balisage XHTML.

Les autres caractères, en dehors de la classification ASCII 7-BIT (donc les codes de 128 à 255), sont codés par des **entités** ou des **références numériques** :
Par exemple :
é (codé é é)
è (codé è ou è)
à (codé à ou à)
ô (codé ô ou ,)
etc.

La liste des entités :

- http://www.w3schools.com/tags/ref_entities.asp

***A noter : le blanc insécable (pas de césure) s'écrit ***

Le type de contenu

Par exemple :

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

```
ou
<meta http-equiv="content-type" content="application/xhtml+xml;
charset=ISO-8859-1" />
```

Le HTML est servi en tant que HTML (donc text/html).
Selon le W3C, le XHTML 1.0 peutêtre servi en tant que HTML (donc text/html) ou en tant que XHTML (donc application/xhtml+xml).
Le XHTML 1.1, lui doit être servi uniquement en tant que XHTML.

Le corps du document

Le corps d'un document peut contenir divers éléments :

- du texte (titres, paragraphes, listes, etc.)
- des images
- des hyperliens
- des tableaux
- des formulaires
- des cadres
- des objets externes (applets Java, Flash …)
- etc.

Remarque : HTML5 propose de nouveaux types d'éléments.

Le corps est délimité par les balises **<body>** et **</body>**

PRÉSENTATION DE QUELQUES ÉLÉMENTS XHTML

Les balises div et span

Les éléments, div et span (en association avec les id et les classes des CSS), se présentent comme un **mécanisme générique pour structurer** des documents Web.
Rappels : div est un élément de type bloc. span est un élément en ligne. Aucun d'entre eux n'apporte de contrainte de présentation, ils sont « neutres » à cet égard. **Ils servent à « ajouter » de la structure**.
NB : Attention, ces éléments n'ont pas de sens particulier, ils sont neutres également sur le plan de la sémantique. Par conséquent, ils ne devraient pas remplacer systématiquement les autres éléments.

Les tableaux

Présentation

Marqueurs <table> </table>
Chaque ligne est encadrée par <tr> </tr>
Les cellules d'en-tête sont encadrées par <th> </th>
Les cellules de valeur sont encadrées par <td> </td>
Les marrqueurs <thead></thead> <tfoot></tfoot> <tbody></tbody> permettent de structurer les tableaux.
La balise <caption></caption> permet d'indiquer la légende du tableau.
Il est souhaitable d'ajouter à la balise <table> l'attribut summary = indiquer un résumé du tableau (<table summary="ce que contient le tableau">).

Exemple :

```
<table>
<caption>L&eacute;gende du tableau</caption>
<thead>
    <tr>
        <th>Cellule d'en t&ecirc;te A</th>
        <th>Cellule d'en t&ecirc;te B</th>
    </tr>
</thead>
<tfoot>
    <tr>
        <td>Cellule de pied de tableau A</td>
        <td>Cellule de pied de tableau B</td>
    </tr>
</tfoot>
<tbody>
    <tr>
        <td>Valeur A ligne 1</td>
        <td>Valeur B ligne 1</td>
    </tr>
    <tr>
        <td>Valeur A ligne 2</td>
        <td>Valeur B ligne 2</td>
   </tr>
</tbody>
</table>
```

Légende du tableau

Cellule d'en tête A	Cellule d'en tête B
Valeur A ligne 1	Valeur B ligne 1
Valeur A ligne 2	Valeur B ligne 2
Cellule de pied de tableau A	**Cellule de pied de tableau B**

Colonnes et lignes étendues

L'attribut **colspan** crée des cellules qui s'étendent sur plusieurs colonnes d'un tableau

Exemple :

```
<table>
<tr>
<th colspan="2">Cellule d'en t&ecirc;te &eacute;tendue en
largeur</th>
</tr>
<tr>
<td>Valeur A ligne 1</td><td>Valeur B ligne 1</td>
</tr>
<tr>
<td>Valeur A ligne 2</td><td>Valeur B ligne 2</td>
</tr>
</table>
```

Cellule d'en tête étendue en largeur	
Valeur A ligne 1	Valeur B ligne 1
Valeur A ligne 2	Valeur B ligne 2

L'attribut **rowspan** crée des cellules qui s'étendent sur plusieurs lignes d'un tableau

Exemple

```
<table>
<tr>
<th>Cellule d'en t&ecirc;te A</th><th>Cellule d'en t&ecirc;te B</th>
</tr>
<tr>
<td rowspan="2">Valeur A ligne 1 et 2 (&eacute;tendu)</td><td>Valeur
B ligne 1</td>
</tr>
<tr>
<td>Valeur B ligne 2</td>
</tr>
</table>
```

Cellule d'en tête A	Cellule d'en tête B
Valeur A ligne 1 et 2 (étendu)	Valeur B ligne 1
	Valeur B ligne 2

Les liens

Les **ancres** ou **liens hypertextes** permettent de naviguer de documents en documents ou à l'intérieur d'un document.

Chemins relatifs

Pour exprimer la destination, il est possible d'utiliser des chemins **absolus** ou **relatifs**.

Chaque document est positionné dans une arborescence de répertoires et de fichiers.
Le répertoire de plus haut niveau (contenant tous les autres documents ou fichiers), est appelé la « racine ».

On peut exprimer la destination d'un lien de manière **absolue** (c'est une mauvaise solution dans la plupart des cas).
Par exemple :
~~E:\undossier\projets\cours\exemples\index.html~~
Le problème est que si je déplace l'ensemble des documents, par exemple pour les inclure sur un autre disque dur de mon ordinateur, les chemins ne seront plus valides.

On peut exprimer la destination d'un lien de manière **relative** à la position courante

Pour descendre dans l'arborescence, vers un sous répertoire du répertoire courant :
nom_sous_repertoire/
ou
./nom_sous_repertoire/
(./ représentant le positionnement courant)

Pour remonter dans l'arborescence, vers un répertoire « parent » :
../
Il est possible de remonter de plusieurs niveaux :
../../../

Il est possible de remonter puis de redescendre dans l'arborescence :
../../autre_repertoire/autre_sous_repertoire

Types de liens

Lien textuel vers une autre page du même site et du même répertoire
```
<a href="page1.html">Lien</a>
```

Lien textuel vers une autre page du même site et d'un autre répertoire
```
<a href="../../rep1/sousrep2/page1.html">Lien</a>
```

Lien textuel vers une autre page dans une nouvelle fenêtre du navigateur
```
<a href="exemples/page1.html" target="_blank">Lien</a>
```

Attention, l'attribut target n'est pas valide pour les versions strictes de XHTML (et HTML). Il ne semble pas y avoir d'alternative proposée par le langage actuellement. Il est possible d'utiliser le JavaScript pour ouvrir une nouvelle fenêtre sur le click d'un lien, mais cette alternative présente d'autres inconvénients.

Lien textuel vers un document dans une nouvelle fenêtre
```
<a href="text.txt" target="_blank">Document text</a>
```
Même remarque que précédemment pour l'attribut target.

Lien textuel vers un document d'un autre site
```
<a href="http://www.unsite.ext/">Un site</a>
```

Lien textuel vers un fragment du document courant
```
<a href="#sommet">Lien</a>
```

Chaque marqueur de "fragment" est identifié par un **signet**
de la forme **Sommet**

Lien textuel vers un fragment d'un autre document
```
<a href="exemples/page1.html#bas">Lien</a>
```

Lien sous forme d'image vers une autre page
```
<a href="index.html"><img border="0" src="../../images/ampoule.gif"
width="16" height="18" /></a>
```

Les images cliquables (ou réactives)
On peut définir des zones cliquables (un lien) dans une image
```
<map id="map1">
<area href="exemples/red.html" shape="rect" coords="0, 0, 100,
100" />
<area href="exemples/green.html" shape="rect" coords="100, 0, 200,
100" />
<area href="exemples/yellow.html" shape="rect" coords="0, 100, 100,
200" />
<area href="exemples/blue.html" shape="rect" coords="100, 100, 200,
200" />
</map>
<img border="4" src="../../images/map.gif" usemap="#map1"
width="200" height="200" />
```

Liens vers une adresse E-mail
Il est possible de faire des liens vers des adresses E-mail : mailto:someone@domain.ext
Exemple :
```
<a href="mailto:someone@domain.ext">Lien vers une adresse E-mail</a>
```

Les images

On utilise le marqueur **<img>** pour inclure une image.
Ses principaux attributs sont :

- **src** : indique l'emplacement du fichier source de l'image
- **width** : largeur
- **height** : hauteur
- **alt** : permet d'afficher un texte qui apparaît lorsque l'image ne s'affiche pas et comme info bulle de l'image

Exemple
```
<img src="../../images/ampoule.gif" width="16" height="18"
alt="exemple d'image" />
```

Principaux formats d'images :

GIF (Graphic Interchange Format) :
Le plus répandu pour les graphismes aux tracés simples, sans dégradé de couleurs
Limité à une palette de 256 couleurs (choisies parmi des millions).
Peut être animé.
Gère la transparence.

JPG ou JPEG (Joint Picture Expert Group) :
Le plus adapté aux photos ou aux images présentant de nombreux dégradés de couleurs
Il gère la compression (à perte).

PNG (Portable Network Graphic) :
Format alternatif au GIF, qui pourrait remplacer le GIF (et le JPG)
Sa méthode de compression est améliorée
Attention, il n'est pas supporté par tous les navigateurs (versions anciennes), sa transparence pose également quelques problèmes avec les versions anciennes de certains navigateurs

Les formulaires

Les formulaires sont délimités par les marqueurs **<form> </form>**

Principaux attributs :

- **method** indique sous quelle forme seront envoyées les réponses POST | GET.
- **action** indique l'adresse d'envoi (script "traitement.php" ou adresse E-mail "mailto:adresse@domaine").
- **enctype** (facultatif) spécifie le codage des données dans l'URL. Généralement, il n'est pas nécessaire de le préciser car la valeur habituelle est attribuée par défaut (application/x-www-form-urlencoded). Pour des **formulaires d'upload** de fichiers (téléchargement de fichier depuis le PC du visiteur vers le serveur), par exemple, il faudra indiquer un **enctype="multipart/form-data"**.

Exemples de balises form
```
<form method="post" action="mailto:adresse@domaine.ext"
enctype="text/plain">
<form method="post" action="http://domaine/cgi-bin/script.cgi">
<form method="post" action="../rep/traitement.php">
```

Les balises <form> </form> servent de **conteneurs** permettant de regrouper des éléments qui vont permettre à l'utilisateur de choisir ou de saisir des données, qui seront envoyées à l'URL indiqué dans l'attribut "action" selon la méthode indiquée.

Il est possible d'insérer n'importe quel élément XHTML de base entre les balises <form> </form> (textes, tableaux, liens,...)
Il est également possible (c'est fait pour), d'inclure des éléments "interactifs", de saisie (et d'affichage) de données ou de choix :

- Elément **input** : différents boutons et champs de saisie.
- Elément **textarea** : zone de saisie de texte.
- Elément **select** : liste à choix unique ou multiples.

La balise input

Balise essentielle des formulaires
Syntaxe :

```
<input type="TypeDeChamp" value="ValeurParDefautOuVide"
name="NomDuChamp" [ProprieteSupplementaire] />
```

L'attribut **type** permet de préciser le type d'élément que représente la balise input :

- **text** : saisie d'une ligne de texte. La taille maximale du texte saisi est fixée grâce à l'attribut **maxlength**
- **password** : champ de saisie de texte dans lequel les caractères saisis apparaissent sous forme d'astérisques afin de camoufler la saisie de l'utilisateur
- **checkbox** : cases à cocher pouvant admettre deux états : coché (checked) et non coché. Lorsque la case est cochée la paire nom/valeur est envoyée
- **radio** : choix parmi plusieurs boutons radios proposés (l'ensemble des boutons radios devant porter le même attribut name). La paire nom/valeur du bouton radio sélectionné sera envoyée. Un attribut checked pour un des boutons permet de préciser le bouton sélectionné par défaut
- **file** : fichier qui sera envoyé avec le formulaire
- **hidden** : champ caché. Ce champ non visible sur le formulaire (mais visible dans le code source) permet de préciser un paramètre non interfacé
- **reset** : bouton de remise à zéro permettant de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut
- **submit** : bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut value
- **image** : image jouant le rôle d'un bouton de soumission

Exemple

```
<form id="form1" method="post" action="demo.php">
```

Text

```
<input type="text" id="id1" name="V1"
value="" />
OU
<input type="text" id="id1" name="V1"
value="valeur initiale" />
```

Textarea

```
<textarea rows="2" id="id2" name="V2"
cols="20"></textarea>
OU
<textarea rows="2" id="id2" name="V2"
cols="20">Valeur initiale</textarea>
```

Checkbox

☒
OK

```
<input type="checkbox" id="id3"
name="V3" value="ok"
checked="checked" /> OK
<input type="checkbox" id="id3"
```

	☐ NON OK	`name="V3" value="ok"/> NON OK`
Radio	◉ OUI ○ NON	`<input type="radio" id="id4" name="V4" value="Oui" checked="checked" /> OUI` `<input type="radio" id="id4" name="V4" value="Non" /> NON`

Select	Choix 1	`<select size="1" id="id5" name="V5">`
		`<option value="1" selected="selected">Choix 1</option>`
		`<option value="2">Choix 2</option>`
		`<option value="3">Choix 3</option>`
		`</select>`

File	Parcourir...	`<input type="file" id="id6" name="V6" />`
Password		`<input type="password" id="id7" name="V7" />`
Hidden		`<input type="hidden" id="hid1" name="champ_valeur_cachee" value="utile" />`
Submit	Envoyer	`<input type="submit" value="Envoyer" id="b1" name="B1" />`
Image	OK!	`<input type="image" src="chemin/envoyer.gif" width="24" height="12" alt="Envoyer" />`
Reset	Rétablir	`<input type="reset" value="Reset" id="b2" name="B2" />`

`</form>`

Envoi des données

Lorsque le formulaire est soumis, les données présentes dans le formulaire sont envoyées au script (ou à l'adresse Email) sous forme de **paires nom/valeur** :
nom de l'élément de formulaire = valeur associée

L'ensemble des paires nom/valeur sont séparées par des **esperluettes** (caractère &) :
champ1=valeur1&champ2=valeur2&champ3=valeur3

Selon la méthode, les modalités d'envoi au serveur sont différentes :

- La méthode **GET** permet d'envoyer les éléments du formulaire au travers de l'URL du script, en ajoutant l'ensemble des paires nom/valeur à l'URL du script, séparé de celui-ci par un point d'interrogation, ce qui donne un URL du type: http://nom_du_serveur/rep /script.php?champ1=valeur1&champ2=valeur2... Toutefois, la longueur de la chaîne URL est limitée

- La méthode **POST** est une bonne alternative à la méthode GET. Cette méthode code les informations de la même façon que la méthode GET (encodage URL et paires nom/valeur) mais elle envoie les données à la suite des en-têtes HTTP, dans un champ appelé corps de la requête. De cette façon la quantité de données envoyées n'est plus limitée de la même manière, elle est connue du serveur grâce à l'en-tête permettant de connaître la taille du corps de la requête.

Frameset (jeu de cadres) et frame (cadre)

La technologie des frames permet d'afficher **plusieurs pages au sein d'une même fenêtre, dans des cadres (frames)**.
Cette technique n'est généralement pas recommandée pour des question d'ergonomie, d'accessibilité, de maintenance…

Un fichier "parent" contient l'agencement des cadres
Il ne possède pas de <body> (corps de document), mais un **<frameset>** (jeu de cadres)

Exemple avec 2 cadres verticaux :

```
<frameset cols="10%,90%">
<frame src="f1.html" name="gauche" />
<frame src="f2.html" name="droite" />
</frameset>
<noframe>
Votre navigateur n'affiche pas les frames
</noframe>
```

La balise <frameset>

Principaux attributs (non exhaustif et attributs parfois non strictement valides) :

- **rows** : taille relative des cadres verticalement. Pourcentage (entre 1 et 100) ou valeur en pixels. Il est possible d'utiliser le caractère * pour spécifier que l'on prend la taille restante <frameset cols="10%,*">

cols : taille relative des cadres horizontalement. Pourcentage (entre 1 et 100) ou valeur en pixels. Il est possible d'utiliser le caractère * pour spécifier que l'on prend la taille restante <frameset cols="10%,*">

frameborder (1 ou 0) : indique si les cadres ont une bordure ou non

Exemple avec 2 cadres horizontaux

```
<frameset rows="10%,90%" frameborder="0">
<frame src="f1.html" name”haut” />
<frame src="f2.html" name="bas" />
</frameset>
<noframe>
Votre navigateur n'affiche pas les frames
</noframe>
```

La balise <frame>

Principaux attributs (non exhaustif et attributs parfois non strictement valides) :

- **src** : URL du fichier source du cadre
- **name** : nom du cadre
- **noresize** : interdit à l'utilisateur de redimensionner les cadres
- **scrolling** (yes no auto) : permet ou non l'affichage d'une barre de défilement (auto signifie que la barre de défilement est affichée uniquement lorsque nécessaire)

Exemple avec imbrication de deux framesets

```
<frameset rows="20%, 80%">
<frame src="f1.html" name="haut" noresize scrolling="no" />
<frameset cols="20%, 80%">
<frame src="f2.html" name="menu" noresize scrolling="no" />
<frame src="f3.html" name="principal" scrolling="yes" />
</frameset>
</frameset>
<noframe>
```

```
Votre navigateur n'affiche pas les frames
</noframe>
```

Les iframes

Au lien de travailler avec un jeu de cadres, on peut utiliser les iframe pour inclure un cadre dans le corps d'un document. On parle de cadre incorporé ou iframe (iframe = inline frame = cadre incorporé).

Principaux attributs (non exhaustif et attributs parfois non strictement valides) :

- **src** : URL du fichier source du cadre
- **width :** largeur du cadre
- **height** : hauteur du cadre
- **name** : nom du cadre

CONCLUSION : REFLEXIONS SUR LA PRATIQUE

Privilégier XHTML à HTML

Les documents XHTML sont **conformes à XML**, donc lisibles, ou éditables avec les outils XML.
XHTML est conçu pour fonctionner à la fois avec les anciens navigateurs et les navigateurs récents. La **compatibilité avec les futures standards** est bien plus probable.
La **rigidité** de XHTML est plutôt une qualité qu'un défaut. L'écriture en XHTML « pousse à » adopter les normes actuelles de conception Web.

Ces remarques sont valables si l'on considère que XHTML est la dernière version du langage (si l'on considère la dernière version normalisée entièrement).
Oui, mais **HTML5 arrive**...

Valider ses documents

Il est utile de **valider le code** des documents produits :

- afin d'être sûr de la **syntaxe** et de la **grammaire** utilisée et se conformer à la norme d'encodage ;
- indirectement, d'assurer une **meilleure interprétation du code** par les navigateurs (ils tendent à être de plus en plus en phase avec les normes de codage) ;
- indirectement, d'assurer une **meilleure pérennité du code** pour les navigateurs à venir.
- Un document valide sera sans doute plus facile à faire **évoluer vers les futures normes**.

Remarques :
Certains navigateurs n'interprètent pas correctement le XHTML (anciens navigateurs ou dans certains cas, les navigateurs actuels). La rigueur du codage peut parfois entraîner des difficultés (malgré tout, HTML présente aussi des difficultés).

La validation des documents, même si elle est **nécessaire**, n'est **pas suffisante** : seule la validité de la syntaxe est vérifiée, en aucun cas la qualité structurelle ou sémantique du document.

Le validateur du W3C :

- http://validator.w3.org

Séparer le contenu / la structure de la présentation

La combinaison XHTML et CSS permet de nettement séparer le contenu, sa structure de l'aspect présentation :

- **Le contenu et la structure sont gérés par XHTML.**
- **La présentation est gérée par les CSS.**

Il est alors possible de faire évoluer les 2 séparément.
Lorsque le style est géré dans des feuilles externes, appliquées à plusieurs documents XHTML, on peut modifier le style de plusieurs documents de manière simultanée.
La lecture des documents XHTML est grandement facilitée.
Le même document XHTML peut être lu sur différents supports.

Adopter un balisage structurel et sémantique

Avec l'utilisation des feuilles de style, il est possible de redéfinir et de personnaliser la présentation par défaut de l'ensemble des éléments proposés par XHTML.

Il est dès lors possible et souhaitable d'adopter un **balisage sémantique** (un paragraphe est un paragraphe, un titre est un titre…)
Cela permet de se conformer à **l'objectif de XHTML : décrire le contenu d'un document et le structurer**.

Cela permet de dépasser le seul aspect « présentation » dans les documents, qui sont alors interprétables autrement que dans un navigateur « standard » (par exemple pour les moteurs de recherche, qui « lisent » les documents en analysant le contenu, ou pour des lecteurs braille…). Cela permet d'établir des documents ouverts à d'autres type de supports.

De même, il est souhaitable de **clairement structurer les documents** (haut de page, menu, pied de page...)
Le code XHTML produit est alors généralement allégé, plus lisible et plus clair, la maintenance est plus facile. Les refontes graphiques, ou les changements de mise en page sont généralement largement facilités.