Université de Montréal

SCI6005 Information numérique et informatique documentaire (A2021)

Christine Dufour, EBSI, UdeM

A2021 2021-10-12

Cours 6 - Développement de sites Web : formats de documents structurés, HTML

Christine Dufour, EBSI, UdeM

Table des matières

Cours 6 - Développement de sites Web : Documents structurés & HTML

1. + Au programme aujourd'hui

- Formats de documents structurés
- HTML

2. + Alignement pédagogique

Objectifs visés, matériel du cours et évaluation : Examen mi-session

Lien entre les objectifs, les compétences à développer et le matériel du cours 6

Objectif général : Comprendre la place des technologies et de l'information numérique en contexte documentaire		
Objectifs spécifiques	**Compétences à développer**	**Matériel associé**
Décrire les principales technologies utilisées en milieux documentaires	Définir ce qu'est un format de document structuré	Section *Documents structurés*
	Connaître la syntaxe de base d'un élément HTML (balise d'ouverture, de fermeture, attribut)	Section *Éléments du langage HTML > Syntaxe HTML*
	Décrire la structure logique de base d'une page HTML	Section *Éléments du langage HTML > Structure logique globale*
	Connaître les éléments présents dans l'entête (title, meta, link)	Section *Éléments du langage HTML > Structure logique globale > Éléments charpente : head*
	Savoir utiliser les principaux éléments HTML (contenu structuré)	Section *Éléments du langage HTML > Contenu structuré* TP Site Web
	Expliquer la différence entre une adresse absolue et une adresse relative	Section *Éléments du langage HTML > Contenu structuré > Élément de niveau texte : a*
	Pouvoir « construire » l'adresse absolue et relative d'un fichier sur un serveur Web	Section *Éléments du langage HTML > Contenu structuré > Élément de niveau texte : a*
	Expliquer l'importance de la validation des pages Web	Section *Éléments du langage HTML > Validation HTML*

Objectifs visés, matière du cours et activités associées

Lien entre les objectifs, la matière du cours 6 et les activités associées

Objectif général : Concevoir et implanter des systèmes et services d'information numérique		
Objectifs spécifiques	**Matière**	**Activités**
Exploiter la structuration de l'information numérique dans différents contextes	*Éléments du langage HTML*	*TP Site Web*
	Éléments du langage HTML	Exercices HTML sur StudiUM : • Dictée trouée HTML - niveau 1 • Dictée trouée HTML - niveau 2 • Dictée trouée HTML "spécial chemins"

3. Introduction

Comme une grande majorité de milieux, les milieux documentaires ont adopté le **Web** comme **plateforme** pour offrir des contenus et des services à leurs utilisateurs. Bien que tous les professionnels et toutes les professionnelles de l'information n'ont pas comme tâche de développer des sites Web, une **compréhension de base du développement Web** est utile à toutes et tous afin de bien comprendre la philosophie derrière un développement Web qui privilégie entre autres l'*accessibilité*.

4. Documents structurés

Remarque

Matériel adapté de Marcoux, Yves. 2007. Notes de cours du SCI6052 Information documentaire numérique. EBSI, FAS, UdeM.

Parmi les formats de fichiers qui sous-tendent le Web se trouvent les **formats de documents structurés**. HTML est un exemple de ce type de format comme l'est XML. L'idée de base d'un format de documents structurés est qu'il s'agit d'un format qui permet de décrire la **structure logique d'un document**.

Exemple de la structure logique d'un courriel ***Exemple***

Prenons l'exemple du **courriel** qui suit :

COURRIEL

De: ana.conda@escie.uestenciel.ca

À: ali.gator@escie.uestenciel.ca; rhino.c.ross@escie.uestenciel.ca

Objet: Réunion du COSP

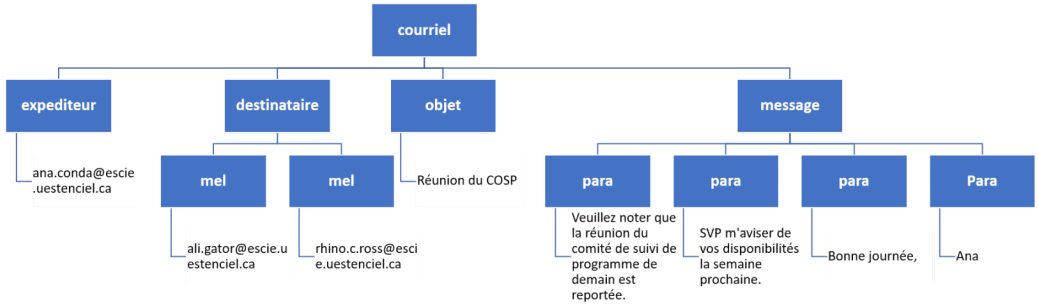
Veuillez noter que la réunion du comité de suivi de programme de demain est reportée.

SVP m'aviser de vos disponibilités la semaine prochaine.

Bonne journée,

Ana

Par "*structure logique*" on entend les **éléments qui le composent** soit ici un *expéditeur*, des *destinataires* (une ou plusieurs adresses courriels), un *objet* et le *message* comme tel (composé d'un ou plusieurs paragraphes). Cette structure logique peut être représentée sous forme d'un **organigramme** comme suit où chaque type d'éléments est dans un rectangle bleu, éléments qui s'emboîtent pour représenter la hiérarchie de la structure, les éléments les plus spécifiques étant associés aux contenus :



Structure logique d'un courriel

Ainsi un **format de documents structurés** permet de représenter cette *structure logique*. Par exemple, le courriel pourrait être représenté de la manière suivante en **XML** :

```xml
<courriel>
  <expediteur>ana.conda@escie.uestenciel.ca</expediteur>
  <destinataire>
    <mel>ali.gator@escie.uestenciel.ca</mel>
    <mel>rhino.c.ross@escie.uestenciel.ca</mel>
  </destinataire>
  <objet>Réunion du COSP</objet>
  <message>
    <para>Veuillez noter que la réunion du comité de suivi de programme de demain est reportée.</para>
    <para>SVP m'aviser de vos disponibilités la semaine prochaine.</para>
    <para>Bonne journée,</para>
    <para>Ana</para>
  </message>
</courriel>
```

XML est un format dit structuré, ou *format de documents structurés*, car il permet le **balisage descriptif logique** des documents. Par *balisage descriptif logique*, on entend le fait que le format permet l'ajout de *balises descriptives* au texte pour en indiquer la structure logique. Examinons de plus près la ligne suivante :

```xml
<expediteur>ana.conda@escie.uestenciel.ca</expediteur>
```

Dans la **syntaxe XML**, les *balises descriptives* sont des chaînes de caractères entourées par les caractères < et > comme, par exemple, `<expediteur>`. Le texte est encadré par deux balises descriptives, une qui en indique le début et l'autre la fin. Comme on peut le voir dans l'exemple, la balise de fin se distingue dans sa syntaxe par l'ajout du caractère / devant la chaîne de caractères de la balise.

Au **niveau sémantique**, la chaîne de caractères de la balise descriptive est un *identificateur générique* qui représente la *nature du texte* qu'elle encadre. Dans l'exemple ci-dessous, elle indique que *ana.conda@escie.uestenciel.ca* est l'expéditrice du courriel.

L'extrait ci-dessous permet de comprendre une autre caractéristique d'un format structuré, soit le fait que certains éléments peuvent être **imbriqués** :

```xml
<message>
  <para>Veuillez noter que la réunion du comité de suivi de programme de demain est reportée.</para>
  <para>SVP m'aviser de vos disponibilités la semaine prochaine.</para>
  <para>Bonne journée,</para>
  <para>Ana</para>
</message>
```

On remarque en effet ici que l'on retrouve quatre éléments `para` à l'intérieur d'un élément `message`. Cette imbrication permet de représenter la structure logique du courriel. L'organigramme présenté précédemment montre en effet que le message d'un courriel se compose de paragraphes.

Une autre caractéristique importante d'un format structuré est qu'il s'agit d'un **format texte**, c'est-à-dire qu'on peut le lire à partir d'un éditeur de texte simple comme *Bloc-notes* ou *TextEdit*.

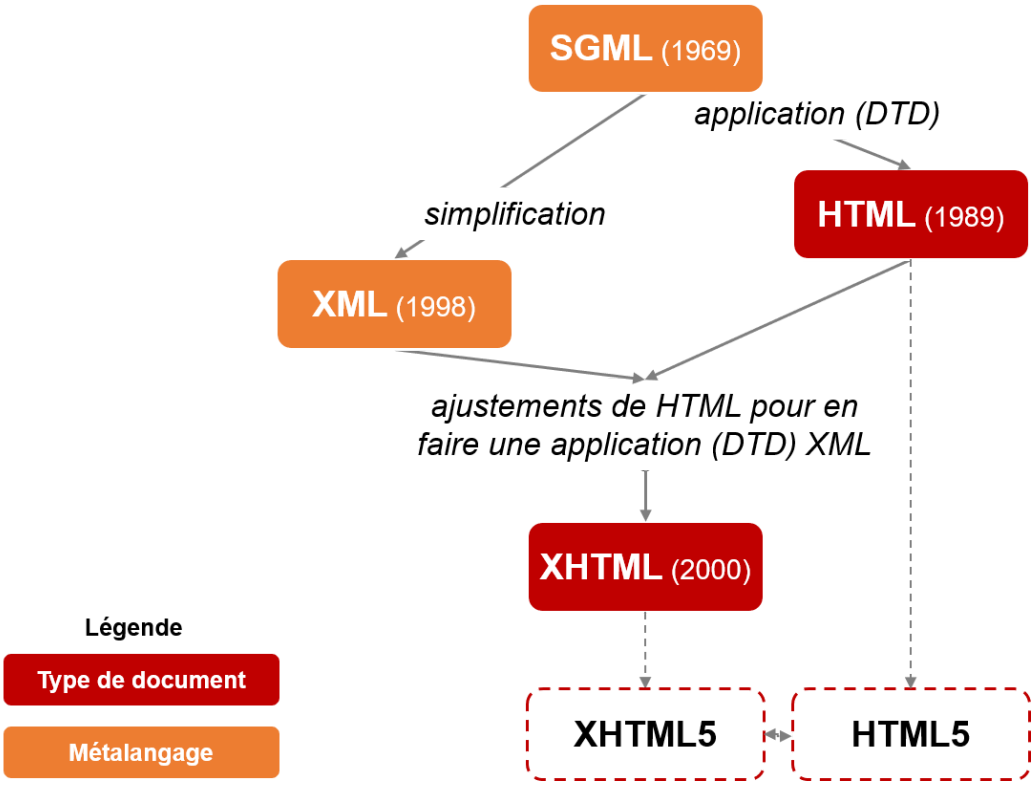
Finalement, il est à noter que le **balisage descriptif logique** en XML ne véhicule explicitement **aucune information quant à la mise en forme des contenus**. Dans l'exemple, l'élément `objet` ne précise en rien quel visuel l'objet du courriel doit prendre dans le logiciel de messagerie. On cherche en effet à séparer le contenu comme tel de son traitement (par exemple de sa mise en forme). La mise en forme dans XML se fera dans un autre fichier, une *feuille de styles* par exemple.

On retrouve **deux "familles" de formats de documents structurés** :

- Certains formats de documents structurés peuvent servir à *décrire n'importe quel type de document comportant une structure logique*. Ils peuvent aussi bien servir à décrire la structure d'un courriel, que celle d'un livre ou celle d'une fiche de recette! On parlera en ce cas d'un **format de balisage généralisable**. XML est un exemple de format de balisage généralisable comme il propose un *métalangage* permettant de décrire n'importe quelle structure logique. Il permet en effet de définir l'ensemble des balises d'une structure soit dans une définition de documents (*document type definition DTD*) ou un schéma XML.
- D'autres formats de documents structurés décrivent la *structure d'un seul type de document*. C'est le cas par exemple du format HTML que nous explorerons un peu plus tard. On retrouve dans le format HTML un *ensemble prédéfini d'éléments* qui permettent de représenter la *structure logique d'une page Web*.

Historique des formats de documents structurés

XML n'est pas le seul format de documents structurés et ce n'est pas non plus le premier à être apparu ou le plus récent! Le **schéma** ci-dessous trace l'*évolution des formats de documents structurés* et le **tableau** qui suit décrit les *caractéristiques des formats*.



Évolution des formats de documents structurés

Caractéristiques des principaux formats de documents structurés

Date de création	Format	Normalisation	Type	Commentaires
1969	**SGML** (*Standard Generalized Markup Language*)	ISO/IEC 8879:1986	Métalangage	• Développé à partir d'un produit d'IBM GML (auteurs Goldfarb, Mosher, Lorie) • Peut être utilisé pour représenter la structure logique de n'importe quel type de documents

Date de création	Format	Normalisation	Type	Commentaires
1989	**HTML** (*HyperText Markup Language*)	ISO/IEC 15445:2000	Type de document spécifique : page Web	• Application SGML c'est-à-dire que SGML a été utilisé pour décrire la structure logique d'une *page Web*. Cette description correspond à HTML. • Plusieurs versions HTML. La plus récente est HTML 5 (recommandation finale en 2014)
1998	**XML** (*eXtensible Markup Language*)	Recommandation du W3C	Métalangage	• La trop grande complexité de SGML en ayant ralenti l'adoption, XML a été créé. Il s'agit d'un métalangage moins complexe et un peu moins puissant • Peut être utilisé pour représenter la structure logique de n'importe quel type de documents
2000	**XHTML** (*eXtensible HyperText Markup Language*)	Recommandation du W3C	Type de document spécifique : page Web	• Ré-écriture du format HTML à partir de XML. XML a ainsi été utilisé pour décrire la structure logique d'une page web. Cette description correspond à XHTML • XHTML possède ainsi certaines caractéristiques XML absentes de HTML comme une syntaxe plus stricte à plusieurs égards • Plusieurs versions XHTML. La plus récente est XHTML5

Avantages et désavantages des formats de documents structurés

Les **avantages** à utiliser des formats de documents structurés sont **nombreux**, comme illustré dans le tableau ci-dessous. Si les **désavantages** semblent moins nombreux, il n'en demeure pas moins un **obstacle important** dans certains contextes à leur adoption.

Avantages et désavantages des formats de documents structurés

Avantages	Désavantages

• Balisage logique qui permet, en distinguant le contenu de ses traitements (par exemple la mise en forme) de :
  ◦ faciliter la réutilisation de l'information
  ◦ diviser le travail entre auteurs, typographes, informaticiens, etc.
• Balisage logique qui permet, en identifiant la nature des contenus, de faciliter leur indexation automatique et de faire des recherches d'information plus efficaces
• Formats normalisés qui offrent une meilleure garantie quant à leur pérennité et leur interopérabilité
	• Surtout un changement « culturel » pour les auteurs qui passent d'un logiciel de traitement de texte

Conséquences pour le professionnel ou la professionnelle de l'information

L'utilisation des formats de documents structurés demande de développer des **habiletés** entre autres sur le plan de la **modélisation** de l'information. En effet, s'il faut développer une chaîne de traitement XML pour un nouveau type de document, il faut être en mesure d'en *modéliser la structure*. Il faut bien comprendre ces chaînes de traitement, en particulier les possibilités de traitement automatique. Finalement, cela demande aussi la **maîtrise des outils** nécessaires pour leur traitement.

Remarque

Si vous êtes tout particulièrement intéressé.e aux documents structurés, le cours INU3011 Documents structurés[1] est pour vous!

5. Éléments du langage HTML

Nous nous attarderons maintenant sur un format de documents structurés très présents sur le Web, **HTML**, pour en comprendre les **fondements** (*syntaxe*, *structure*, éléments, validation). Il est à noter qu'un **glossaire interactif des éléments HTML** est accessible à l'URL https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005 et doit être utilisé en **complément des notes de cours**.

5.1. Syntaxe HTML

Attention

HTML5 (que nous utiliserons dans le cours) permet une *syntaxe plus laxiste* que d'autres formats de balisage pour les pages Web comme XHTML qui adoptent la syntaxe plus stricte XML. En HTML5, par exemple, les balises peuvent être écrites en majuscule ce qui n'est pas le cas en XHTML.

Toutefois, nous **privilégierons dans le SCI6005** une **syntaxe stricte même pour HTML5**. Il est en effet important d'acquérir de bonnes habitudes; ainsi, si un jour vous avez à faire du HTML avec une version plus stricte, vous serez déjà habitué(e) à le faire. Dans les notes de cours, un astérisque (*) indiquera les éléments de syntaxe plus stricts.

Balises d'ouverture et de fermeture

HTML étant un format de document structuré il en possède les caractéristiques. Il permet ainsi le **balisage descriptif logique**. Voici un exemple d'un contenu balisé en HTML :

```
1 <title>Page d'accueil de la BLSH</title>
```

Comme illustré dans l'exemple ci-dessus, les **règles de base** concernant sa syntaxe sont les suivantes :

- À une exception près décrite par la suite, le **contenu** est **encadré** par un élément HTML qui en indique la **nature**. Dans l'exemple ci-dessus, on comprend ainsi que *Page d'accueil de la BLSH* est un titre.
- L'**élément HTML** est constitué d'une **balise d'ouverture** où le nom de l'élément est en minuscule* et encadré par < et >.
- La **balise de fermeture** de l'élément HTML se distingue de la balise d'ouverture par l'ajout d'une barre oblique / devant le nom de l'élément.

Cependant, **certains éléments HTML n'encadrent pas de contenu** comme l'exemple ci-dessous :

```
1 Maître Corbeau sur un arbre perché<br/>
```

L'élément `br` permet d'insérer dans un texte un retour de ligne forcé. De par sa nature, il n'encadre pas de contenu textuel comme l'exemple précédent. C'est ce que l'on appelle un **élément vide**. Il y en a d'autres que nous verrons plus tard comme l'élément `img` pour insérer une image, `hr` pour afficher une ligne horizontale ou `meta` pour préciser des métadonnées. Il y a **trois syntaxes possibles** pour les éléments vides :

1. On peut *juxtaposer la balise d'ouverture et de fermeture** : `<br></br>`
2. On peut *contracter en une seule balise la balise d'ouverte et de fermeture** : `<br/>`
3. On peut *utiliser uniquement la balise d'ouverture* : `<br>`

Par souci de retenir une syntaxe stricte, la troisième option n'est pas recommandée. La deuxième option a l'avantage de demander moins d'effort à la saisie comme elle compte un plus petit nombre de caractères.

[1] https://admission.umontreal.ca/cours-et-horaires/cours/inu-3011/

Imbrication des éléments HTML

Il est possible d'**imbriquer des éléments HTML** comme, par exemple :

```
1 <p>Cet événement marqua le début de la <strong>Première guerre mondiale</strong> qui allait durer plus de quatre ans.</p>
```

Remarquez bien que l'élément `strong` est **complètement inclus** à l'intérieur de l'élément `p`, ce qui est une condition **nécessaire** pour que cela soit **valide**. Un chevauchement des éléments comme suit n'est pas valide : `<p> ... <strong> ... </p> ... </strong>`.

Toutefois, un **élément ne peut pas inclure n'importe quel autre élément**. L'exemple ci-dessous n'est pas valide :

```
1 <strong>Cet événement marqua le début de la <p>Première guerre mondiale</p> qui allait durer plus de quatre ans.</strong>
```

Il faut respecter la structure logique définie par HTML. Il n'y a pas à s'inquiéter, avec un peu de pratique, on vient à savoir ce que peut ou non inclure un élément. De plus, comme nous le verrons plus tard, il existe des sites permettant de valider le code HTML qui au besoin nous indiqueront les erreurs commises.

Attributs

Certains éléments peuvent avoir des **attributs** (parfois obligatoires, d'autres fois facultatifs) **à l'intérieur de leur balise d'ouverture**, afin d'en *préciser certaines caractéristiques*. En voici un exemple :

```
1 <a href="http://www.w3c.org">Page d'accueil du W3C</a>
```

L'exemple correspond à un lien hypertextuel (élément `a`). Pour un lien hypertextuel, il est nécessaire de préciser la cible du lien. Cela se fait à l'aide de l'attribut `href`.

On indique ainsi, dans la balise d'ouverture, le *nom de l'attribut en minuscule** à la suite du nom de l'élément. Le nom de l'attribut est suivi du *signe égal* = et, finalement, on précise *entre guillemets** la *valeur de l'attribut*. Il est possible d'avoir plus d'un attribut dans une balise d'ouverture. En ce cas, on sépare chaque couple *attribut-valeur* par une espace comme dans l'exemple qui suit :

```
1 <meta name="author" content="John Smith" />
```

Cet exemple correspond à une métadonnée (élément `meta`). Cet élément en fait est un élément vide comme l'indique à la fin de la balise d'ouverture la barre oblique. Une métadonnée possède deux caractéristiques (attributs) : (1) le type de métadonnées (`name="author"` indique qu'il s'agit d'une métadonnée sur l'auteur de la page) et (2) la valeur de la métadonnée (`content="John Smith"` précise que l'auteur est John Smith).

Traitement des espaces et de la casse

En XML et en XHTML, la **casse** (c'est-à-dire les lettres minuscules et majuscules) est **prise en compte**. On doit ainsi toujours mettre les noms des éléments et des attributs en **minuscules**. En HTML5, les éléments et les attributs peuvent être en majuscules ou en minuscules. Toutefois, il est conseillé de conserver la bonne pratique établie de XHTML de les garder en minuscules pour faciliter, au besoin, le passage à XHTML.

Les **espaces**, **tabulations** et **sauts de ligne** entre les éléments balisés sont **ignorés** par les navigateurs. L'exemple ci-dessous est ainsi tout à fait compréhensible pour le navigateur.

```
1 <!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr-ca"><head><title>Module de formation continue</title></head><body><header><h1>Modules de formation continue</h1></header><section><article><p><strong>Responsable :</strong> John Smith</p><p><strong>Sujet :</strong> Macramé</p></article><article><p><strong>Responsable :</strong> Marie Poppins</p><p><strong>Sujet :</strong> L'art du biscuit</p></article></section><footer><p>Mise à jour le 13 novembre 2013</p></footer> </body></html>
```

Toutefois, les tabulations et sauts de ligne sont *fort utiles pour faciliter la lecture aux humains*! L'exemple ci-dessous correspond aux mêmes éléments HTML que le précédent, mais avec une structuration visuelle grâce aux retraits et sauts de ligne.

```
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr-ca">
3 <head>
4    <title>Module de formation continue</title>
5 </head>
6 <body>
```

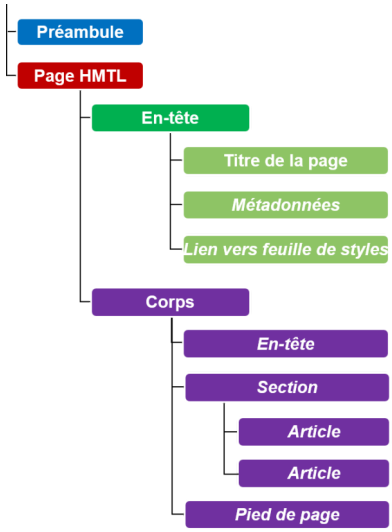
```
 7  <header><h1>Modules de formation continue</h1></header>
 8  <section>
 9   <article>
10     <p><strong>Responsable :</strong> John Smith</p>
11     <p><strong>Sujet :</strong> Macramé</p>
12   </article>
13   <article>
14     <p><strong>Responsable :</strong> Marie Poppins</p>
15     <p><strong>Sujet :</strong> L'art du biscuit</p>
16   </article>
17  </section>
18  <footer><p>Mise à jour le 13 novembre 2013</p></footer>
19 </body>
20 </html>
```

5.2. Structure logique globale

Un fichier HTML doit respecter la **structure logique globale d'une page Web**. Le tableau ci-dessous en explicite les différents éléments. Certains de ces éléments seront décrits dans des sous-sections subséquentes.

Description des éléments de la structure globale d'une page Web

Composantes	Définition	Exemples	Statut
Préambule	Indique la norme HTML retenue	`<!DOCTYPE html>` pour HTML5	Obligatoire
Éléments charpentes	Définissent la structure de plus haut niveau	`html`,`head`, `body`	Obligatoires
Métadonnées	Présentes dans l'en-tête de la page (`<head>`), elles permettent de documenter certains aspects de la page	`title`, `meta`, `link`	`title` est l'unique métadonnée obligatoire
Regroupements structurels	Présents dans le corps de la page (`<body>`), ils permettent de préciser la structure logique du corps de la page	`header`, `footer`, `section`, `article`, `nav`	Facultatifs
Titrage	Permet d'inclure une structure de titre et sous-titres dans le corps de la page (`<body>`)	`h1`, `h2`, ..., `h6`	Facultatifs
Contenu structuré	Permet d'inclure différents types de contenus dans le corps de la page (`<body>`)	`p`, `strong`, `table`, `ol`	Facultatifs



Structure logique globale d'un fichier HTML (les éléments en italique sont facultatifs)

Exemple d'un fichier HTML *Exemple*

L'exemple très simple ci-dessous illustre les différents éléments de la structure logique globale d'une page Web.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr-ca">
<head>
  <title>Module de formation continue</title>
</head>
<body>
 <header><h1>Modules de formation continue</h1></header>
 <section>
  <article>
    <p><strong>Responsable :</strong> John Smith</p>
    <p><strong>Sujet :</strong> Macramé</p>
  </article>
  <article>
    <p><strong>Responsable :</strong> Marie Poppins</p>
    <p><strong>Sujet :</strong> L'art du biscuit</p>
  </article>
 </section>
 <footer><p>Mise à jour le 13 novembre 2013</p></footer>
</body>
</html>
```

a) Éléments charpente : html

L'élément `html` doit **chapeauter tout le document**. On pourra y retrouver certains attributs pour, entre autres, préciser la langue. Par exemple (balise d'ouverture) :

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

L'attribut `xml:lang` est facultatif mais recommandé. Il indique dans quelle langue est le document. La langue est identifiée par son code selon la spécification RFC3066 (http://www.ietf.org/rfc/rfc3066.txt) : http://www.i18nguy.com/unicode/language-identifiers.html.

L'attribut `xmlns` est facultatif; il permettra, dans le cadre du TP, de s'assurer de respecter le balisage plus strict privilégié lors de la validation.

b) Éléments charpente : head

L'élément `head` peut contenir (entre autres) les éléments suivants :

- **Titre de la page** : `title` (*obligatoire*)
- **Liens vers d'autres ressources** : `link` (facultatif)
- **Métadonnées** : `meta` (facultatif)

Attention : Les éléments `link` et `meta` sont facultatifs dans le format HTML mais obligatoires pour le TP Site Web!

L'exemple ci-dessous illustre l'en-tête d'une page Web du site de Ms John Smith et Bill Brown, qui sont les auteurs de la page. La métadonnée "description" nous apprend qu'ils sont consultants en information. L'élément `link` permet de lier le fichier HTML à une feuille de styles externe où se trouveront définis les éléments visuels de la page.

```
<head>
 <title>Site Web de John Smith et Bill Brown</title>
 <meta name="author" content="John Smith" />
 <meta name="author" content="Bill Brown" />
 <meta name="description" content="Site Web professionnel de John Smith et Bill Brown, consultants en information." />
 <link rel="stylesheet" type="text/css" href="css/styles.css" />
</head>
```

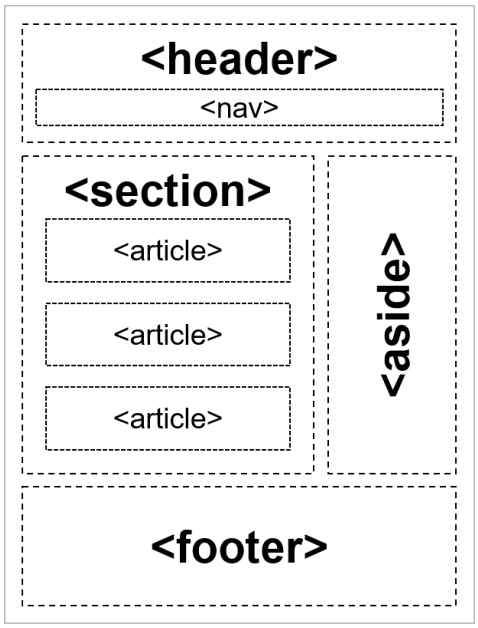
Il est à noter, qu'à une exception près, le **contenu** des éléments présents dans l'élément `head` **n'apparaît pas dans le navigateur**. L'exception est l'élément `title` dont le contenu s'affiche sur l'*onglet de la page*.

c) Éléments charpente : body

L'élément `body` inclut le **corps du document**, c'est-à-dire **tout ce qui est visible dans la fenêtre du navigateur**. On pourra y retrouver différents types d'éléments représentant des regroupements structurels, du titrage et des contenus structurés.

d) Regroupements structurels

Les regroupements structurels permettent de définir des **zones** à l'intérieur du corps d'une page Web. Bien que *facultatifs*, ces éléments sont fort utiles entre autres pour faciliter la définition du visuel de la page.



Principaux regroupements structurels (source : Rimelé, Rodolphe. 2011. HTML5 : Une référence pour le développeur Web. Paris : Eyrolles. Page 92)

5.3. Contenu structuré

Le **contenu structuré** correspond à tout ce que l'on peut vouloir afficher à l'intérieur d'une zone. Les types de contenu structuré sont nombreux; nous en verrons les principaux. On peut les séparer en **deux familles** :

- les éléments de contenu structuré qui définissent la *structure logique du contenu* du document (**éléments de niveau bloc**)
- des éléments qui servent à *qualifier le texte à l'intérieur d'un élément de niveau bloc* ou à *insérer du contenu multimédia* (**éléments de niveau texte ou en ligne**).

Ces deux familles seront brièvement décrites ci-dessous. Par la suite, l'élément de niveau texte `a` sera détaillé plus avant en raison de sa relative complexité. Pour les autres éléments de contenu structuré, veuillez vous référer au glossaire[1] pour de plus amples informations.

a) Éléments de niveau bloc

Les éléments de **niveau bloc** sont *conceptuellement bien séparés de ce qui les précède* et *de ce qui les suit*. Lors de leur restitution (*affichage*), un saut de ligne est généralement ajouté avant et après l'élément balisé. C'est le cas, par exemple, d'un paragraphe ou d'un tableau.

Voici ceux couverts dans le cadre du cours et qui sont expliqués dans le glossaire des éléments HTML[2] :

- *Titrage* : `h1` ... `h6`
- *Paragraphes, blocs d'adresse et citations longues* : `p`, `address`, `blockquote`
- *Éléments pour les listes* : `ul`, `ol`, `li`, `dl`, `dt`, `dd`
- *Éléments pour définir des tableaux* : `table`, `caption`, `th`, `tr`, `td`
- *Ligne horizontale* : `hr`

[1] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005
[2] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005

b) Éléments de niveau texte ou en ligne

Les éléments de **niveau texte**, lors de leur restitution (*affichage*), ne se voient pas ajouter un retour de ligne avant ou après. On peut les utiliser partout où l'on peut retrouver du contenu textuel. On y retrouve, par exemple, des éléments permettant de l'emphase sur certains mots dans un paragraphe.

Voici ceux couverts dans le cadre du cours et qui sont expliqués dans le glossaire des éléments HTML[1] :

- *Emphase* : `strong`, `em`
- *Image* : `img`
- *Citation courte* : `cite`
- *Lien hypertextuel* : `a`
- *Retour de ligne forcé* : `br`

En raison de ses caractéristiques, l'élément `a` est décrit plus avant dans la section suivante.

i) Élément de niveau texte : a

L'exemple suivant illustre la syntaxe d'un **lien hypertextuel** (élément `<a>`) :

```
1 <a href="uncertainendroit">Texte à cliquer pour se rendre à un certain endroit</a>
```

Les balises d'ouverture et de fermeture de l'élément *<a>* encadrent *ce qui sera cliquable*. On retrouve **obligatoirement** un attribut dans la balise d'ouverture, soit l'**attribut `href`** qui permet de préciser la *destination du lien*. La valeur de l'attribut peut prendre différentes formes *en fonction du type de lien hypertexte*.

On peut retrouver dans une page Web **différents types de liens hypertextes**, dont :

1. Des liens vers une *page d'un autre site Web*;
2. Des liens vers une *page du même site Web*;
3. Des liens vers un *endroit précis dans une page Web*;
4. Des liens vers une *adresse de courrier électronique*.

La différence entre les deux premiers cas réside dans le fait que l'endroit pointé se trouve sur un autre serveur Web (cas 1) ou sur le même serveur Web (cas 2). Le cas 3 quant à lui se distingue comme il ne pointe pas une page en général, mais plutôt un point précis de cette page. Finalement, le cas 4 montre que l'on peut faire des liens avec d'autres protocoles que le protocole http.

1 Cas 1 : Lien vers un autre serveur - Adresse URL absolue

Si on veut se rendre sur une *page d'un autre site Web*, il nous faut nécessairement utiliser une **adresse URL absolue** qui donne **tout le détail sur le chemin pour se rendre**. C'est une idée similaire aux chemins absolus que nous avons abordés plus tôt dans la session. Une adresse URL absolue sera ainsi formée (*les éléments entre crochets carrés sont facultatifs*) :

protocole://serveur[:port]/[chemin/]fichier[#position]

Plus précisément :

- *Protocole* : protocole utilisé par exemple http, https, ftp, telnet
- *Serveur* : adresse physique ou logique de l'ordinateur
- *Port* : numéro du port où le serveur est en attente (80 par défaut si non spécifié pour le protocole HTTP) (*facultatif*)
- *Chemin* : suite de dossiers pour se rendre au document (*facultatif*)
- *Fichier* : nom du document
  - Facultatif pour un lien vers une page Web nommée `index.htm` ou `index.html`
- *Position* : position dans le document (*facultatif*)

Voici un exemple de lien hypertextuel vers un autre site :

```
1 <a href="https://www.ulaval.ca/">Université Laval</a>
```

[1] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005

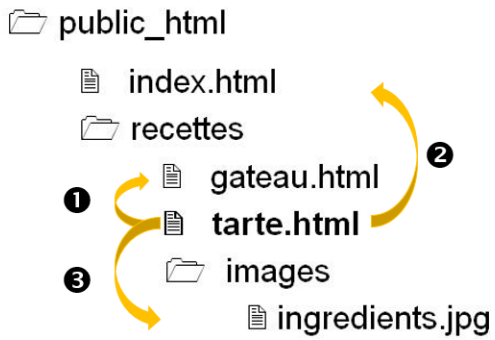
2 Cas 2 : Lien vers une page sur le même serveur - Adresse URL relative

Si vous voulez faire un lien vers une **autre page de votre propre site Web**, il est préférable de ne pas utiliser l'adresse URL absolue. En effet, si jamais vous changiez votre site de serveur, toutes les adresses utilisées pour lier les pages de votre site Web entre elles seraient à corriger. Il est préférable en ce cas d'utiliser une **adresse URL relative**. Tout comme pour les chemins relatifs que nous avons rencontrés plus tôt dans la session, une **adresse URL relative** donne le *chemin d'accès au fichier de destination par rapport à l'endroit où le lien est inclus* (le fichier source). Voici des exemples d'adresse URL relatives :

- index.html
- images/logo.gif
- ../recettes/tarte.html

Ainsi, si vous changez votre site Web de serveur en conservant la même structure de dossiers et fichiers, les liens entre les pages du site demeureront fonctionnels.

Pour déterminer l'**adresse URL relative** d'un fichier, on spécifie le chemin d'accès du fichier en fonction de l'emplacement du fichier source, en omettant l'adresse URL de base de l'espace Web sur le serveur. On retrouve **trois cas possibles** qui sont illustrés dans l'exemple ci-dessous. Il est à noter qu'il s'agit de la même logique que celle des chemins sur un disque dur vus en début de session.



Vous êtes à développer la page `tarte.html` :

1. Pour pointer vers un **fichier situé dans le même dossier** (par ex.: `gateau.html`) on donne seulement le nom du fichier :

   ```
   <a href="gateau.html">Recette de gâteau blanc</a>
   ```

2. Pour pointer vers un **fichier situé dans un dossier parent** (par ex. : `index.html`), on donne le nom du fichier, précédé de « ../ » :

   ```
   <a href="../index.html">Retour à la page d'accueil</a>
   ```

3. Pour pointer vers un **fichier situé dans un dossier placé dans le même dossier** (par ex.: `ingredients.jpg`), on donne le nom du dossier suivi de « / » et du nom du fichier :

   ```
   <img src="images/ingredients.jpg" alt="Photo des ingrédients" />
   ```

3 Cas 3 : Lien vers un point précis dans une page

Il est possible de faire un **lien vers un point précis dans une page**. Par exemple, on retrouve sur certaines pages Web un menu de navigation en haut de la page qui permet d'atteindre certaines sections de cette même page. Pour y arriver, il y a en fait deux étapes :

1. il faut définir les "**points de chute**" dans la page, c'est-à-dire les endroits où on veut que le lien mène,
2. il faut définir le **lien** (*point de départ*).

Définir le point de chute est simple : il suffit d'ajouter l'**attribut `id` à la balise d'ouverture de l'élément où l'on veut arriver et lui attribuer un nom unique**. Par exemple, si vous voulez pour atteindre la section *Recettes du mois* dans une page Web :

```
1 <h2 id="recettes">Recettes du mois</h2>
```

L'attribut `id` peut être ajouté à n'importe quelle balise d'ouverture. Il est important que sa valeur soit bien **unique** (c'est-à-dire qu'elle n'apparaît qu'une seule fois dans une même page).

Une fois le point de chute défini, il ne reste plus qu'à faire le lien. Son **attribut `href`** aura comme *valeur le nom de l'attribut `id`* que vous avez préalablement défini, *précédé du dièse #*. Par exemple :

```
1 <p>Consultez la section <a href="#recettes">Recettes du mois</a> pour faire de belles découvertes!</p>
```

Il est à noter que l'on peut exploiter les attributs `id` des pages d'un autre site Web pour faire un lien plus ciblé. Il s'agit d'ajouter le nom de l'attribut `id` à la fin de l'adresse URL absolue comme suit :

```
1 <p>Vous retrouverez sur le site du Chef Groleau de <a
  href="https://chefgroleau.name/accueil.html#recettes">savoureuses recettes du mois</a>.</p>
```

4 Cas 4 : Lien vers une adresse de courrier électronique

Si vous voulez inclure un lien vers une adresse de courrier électronique, votre lien ressemblera à ce qui suit :

```
1 <a href="mailto:john.smith@gmail.com">John Smith</a>
```

Il faut toutefois réaliser que vous indiquez ainsi en clair l'adresse de courriel. Un robot programmé pour ramasser toutes les adresses de courrier électronique sur des sites pourra ainsi facilement la récupérer et l'ajouter à une liste d'envoi de pourriels.

c) Autres éléments

En sus des éléments de niveau bloc et texte présentés précédemment, on peut retrouver dans le code HTML des **entités** ainsi que des **commentaires**.

i) Entités

On peut retrouver dans une page Web des **entités**, c'est-à-dire des **codes** permettant de représenter certains **caractères spéciaux** (par exemple, l'espace insécable) ou **réservés** (par exemple les chevrons ouvrant et fermant < et > inclus dans la syntaxe des éléments HTML). Dans les débuts du Web, lorsque le jeu de caractères ASCII pur était le plus utilisé, les entités servaient aussi à représenter les caractères accentués. Cette pratique est de moins en moins utilisée comme les jeux de caractères ASCII étendus ou unicodes sont maintenant utilisés. Voici quelques **exemples de caractères spéciaux** et de leur représentation comme entité :

Exemples d'entités

Caractère représenté	Entité littérale	Entité numérique
"	&quot;	&#39;
&	&amp;	&#38;
Espace insécable		
<	&lt;	&#60;
>	&gt;	&#62;

On remarque dans ce tableau :

- La **syntaxe** des entités : Une entité débute par l'*esperluette (&)* et se termine par un *point-virgule (;)*. Ce qui se trouve entre ces deux caractères désigne le caractère à représenter.
- Le **type** d'entités : On retrouve pour chaque caractère représenté une *version littérale* et une *version numérique*. Bien que la version littérale soit plus facile à mémoriser, certains environnements de validation comme celui utilisé pour le TP Site Web **n'acceptent que la version numérique**.

Voici un exemple incluant une entité :

```
1 <p>Je rêve d'une glace Ben &amp;amp; Jerry!</p>
```

ii) Commentaires

Il est possible, et c'est même souhaitable!, d'**ajouter des commentaires** à votre code HTML afin de *documenter par exemple sa structure*. Advenant que le site Web change de main, la personne qui prendra votre relève en sera très reconnaissante. Ces **annotations** sont **ignorées par le navigateur** lors de la visualisation de la page. En voici un exemple :

```
1 <!-- Début du pied de page -->
```

Ainsi, un commentaire est précédé par `<!--` et suivi de `-->`. Les commentaires peuvent aussi être utilisés pour cacher temporairement certains contenus d'une page Web. Il faut toutefois réaliser que si un internaute regarde le code source de votre page, il les verra.

5.4. Validation HTML

Afin d'assurer une **accessibilité optimale** à nos pages Web, il est important de s'assurer qu'elles soient **conformes à la norme HTML utilisée**. La validation peut se faire de différentes manières :

- **Validation humaine** : Les concepteurs et conceptrices des pages Web valident les pages sur la base de leur expertise. La norme HTML étant complexe, il est possible que certaines erreurs leur échappent.
- **Validation via le logiciel d'édition de pages Web utilisé** : Certains environnements de développement intègrent des capacités de validation. Elles ne sont toutefois pas toujours efficaces à 100%.
- **Validation à partir du site du W3C** : Le W3C offre un site qui permet d'effectuer la validation des pages Web (http://validator.w3.org/). Il est à noter qu'il ne permet pas de faire une validation stricte du HTML5 comme souhaité dans le cadre du cours.
- **Validation à partir du service validator.nu** : Le service de validator.nu, en sus d'une validation comme l'offre le W3C, permet de valider de manière stricte HTML (https://validator.nu). Cette validation stricte est possible en ajoutant l'attribut `xmlns="http://www.w3.org/1999/xhtml"` dans la balise d'ouverture de l'élément `html` d'une page. C'est ce que vous ferez dans le cadre du TP Site Web (le paramétrage fin est précisé dans le protocole).

Afin d'être efficace, la validation doit se faire **au fur et à mesure** de la construction des pages. De plus, il est conseillé d'**alterner** entre validation et corrections des erreurs détectées. Une erreur peut en effet avoir des incidences multiples! Sa correction peut faire en sorte d'éliminer plusieurs erreurs dans certains cas.

6. Ressources en lien avec le cours

Matériel de cours

- *Notes de cours (cf. sci6005_a2021_c06_notes_cours)*

Ressources complémentaires

- Glossaire des éléments HTML vus dans le SCI6005[1]
- Exercices HTML sur StudiUM
  - Dictée trouée HTML - niveau 1[2]
  - Dictée trouée HTML - niveau 2[3]
  - Dictée trouée HTML "spécial chemins"[4]

Protocole du TP Site Web[5]

[1] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005
[2] https://studium.umontreal.ca/mod/quiz/view.php?id=3611946 [accès restreint]
[3] https://studium.umontreal.ca/mod/quiz/view.php?id=3611947 [accès restreint]
[4] https://studium.umontreal.ca/mod/quiz/view.php?id=3611948 [accès restreint]
[5] https://studium.umontreal.ca/course/view.php?id=202998#tp_web [accès restreint]

Index

Crédits des ressources

p. 4

http://creativecommons.org/licenses/publicdomain/4.0/fr/, johnny_automatic

p. 5

http://creativecommons.org/licenses/publicdomain/4.0/fr/, maoriveros