Module G - XHTML 1.0 et CSS 2

Préparé par Stéphane Brunet
Révisé par Luc Baron
Version du 25 août 2020

1 Introduction

1.1 Objectifs

Ce document a pour but de donner un aperçu du développement de sites Web à l'aide des standards les plus couramment employés. Bien que beaucoup de détails soient omis, l'accent est mis sur les concepts de base, les règles importantes et quelques "recettes" afin de réaliser des sites web conformes aux normes du World Wide Web Consortium (www.w3.org).
La matière est séparée en deux parties essentielles, le XHTML et les feuilles de style en cascade ou CSS, qui reflètent la façon dont on doit approcher la création de pages web. Cette introduction fournit uniquement le contenu, la structure et la sémantique. La mise en page et le style des différents éléments contenus sont appliqués grâce aux feuilles de style. La nette séparation du contenu et de la mise en page offre de nombreux avantages décrits dans la deuxième partie de ce document.

1.2 Quelques dates importantes de l'histoire du HTML/XHTML

Les personnes intéressées à l'histoire du HTML/XHTML devraient se référer par exemple à l'article sur Wikipédia.

- 1989, au CERN, un informaticien propose un système pour partager l'information entre les chercheurs (le document original est toujours disponible)
- 1991: un premier navigateur simple est mis à la disposition de la communauté
- 1993, la première version du navigateur MOSAIC par le NCSA qui a réellement accéléré l'adoption du Web
- 1995: création du W3C pour faire évoluer cette norme ouverte
- Décembre 1996: le W3C publie la version 1.0 des feuilles de styles CSS
- Décembre 1997: le W3C publie la version 4.0 de la norme HTML
- Janvier 2000: publication de la version 1.0 de la norme XHTML

1.3 Démystifier les normes et acronymes

Beaucoup de normes encadrent le Web. Bien qu'il ne soit pas nécessaire de les connaître tous pour réaliser un site web réussi, la liste non exhaustive suivante permet de les démêler...
Les standards majeurs sont:

HTTP
: HyperText Transfert Protocol - protocole de transfert d'information entre le serveur Web et le client Web.

SGML
: Standard Generalized Markup Language - norme ISO 8879-1986 qui permet de structurer l'information d'un document pour un champ d'applications très vaste. En raison de sa complexité, son usage reste très limité, contrairement aux standards auxquels il a donné naissance: HTML et XML.

HTML
: HyperText Markup Language - Langage de structuration/balisage de documents pour le World Wide Web. La dernière version de ce standard est 4.01.

XML
: Extensible Markup Language - Langage de balisage extensible, basé sur la norme SGML, mais d'usage beaucoup plus simple. La popularité de ce langage provient du fait qu'il est extensible. On le retrouve entre autres dans le domaine du Web, des documents bureautiques (OpenDocument), de la finance et de l'administration.

XHTML
: Réécriture du HTML conforme au standard XML. Ce document se base sur XHTML 1.0 qui est la réécriture du standard HTML 4.01. C'est le choix que nous avons fait pour le développement de nos sites Web.

XHTML5
: Réécriture du HTML 4.01 et XHMTL 1.0 pour se conformer explicitement au modèle de séparation des données : contenu et structure. Nous avons choisi de ne pas utiliser cette norme plus récente, mais plus complexe. Elle est aussi connue sous le nom de HTML5.

CSS
: Cascading Style Sheets - Feuilles de style en cascade. Elles permettent d'ajouter les informations de présentation aux documents (X)HTML et XML: style et position du contenu.

URL
: Uniform Resource Locator - cette norme permet de localiser des ressources sur un réseau ou internet (fichiers, images, ...). Elle permet de spécifier à la fois le protocole et l'adresse de la ressource.

2 XHTML

2.1 Concepts de base du XHTML

Un fichier HTML ou XHTML est un document texte lisible et éditable avec n'importe quel éditeur de texte. Word n'est pas un éditeur de texte pur, mais plutôt un système de traitement de texte qui utilise ses propres codes de formatage des différents éléments du document. Il suffit de demander à Word de sauvegarder un document en format HTML pour constater le grand nombre de codes qui est alors transformé en balises HTML. Ainsi, HTML définit les différents éléments d'un document avec des balises. Celles-ci sont facilement reconnaissables aux caractères `<` et `>` systématiquement placés au début et à la fin de chaque balise. Par exemple:

- la balise `<p>` indique le début d'un paragraphe;
- la balise `</p>` indique la fin du paragraphe (noter le caractère `/` juste après le `<` qui dénote une balise de fin);
- tout ce qui se retrouve entre ces deux balises (`<p>` et `</p>`) appartient au paragraphe en question.

Certains éléments (p. ex. un retour à la ligne ou bien l'insertion d'une image) ne nécessitent pas de balise de fin. Dans ce cas, la structure de la balise est légèrement différente. Par exemple, pour forcer un retour à la ligne, la balise utilisée est `<br />`. Le caractère `/` est cette fois-ci disposé à la fin de la balise, juste avant le `>`. Ainsi, le client Web est en mesure de comprendre qu'il n'y aura pas de balise de fin. Par ailleurs, notez l'espace contenu dans la balise qui n'est pas obligatoire dans XHTML mais qui permet de garder une compatibilité avec le standard HTML plus ancien.
Outre le contenu que des balises englobent, il est possible de modifier des attributs propres à chaque balise (e.g. donner un nom à la balise, ou changer une de ses caractéristiques). Par exemple, lorsqu'une image doit être insérée dans un document HTML, celle-ci n'est pas directement définie dans le document. Il faut indiquer l'URL de l'image afin que le client Web soit capable d'aller récupérer le fichier correspondant. Voici un exemple:

```
<img src="./imagesE/getfirefox_88x31.png" alt="Get Firefox"/>
```

Dans le cas de cette balise d'image, deux attributs `src` et `alt` sont définis. La valeur qui leur est affectée doit nécessairement se trouver entre guillemets.
Il est aussi possible de rajouter dans le fichier HTML des commentaires qui ne seront pas affichés à l'écran. Ceux-ci devront être contenus entre les balises `<!--` et `-->`.

2.2 Structure générale d'un fichier XHTML

Un fichier XHTML minimaliste est présenté ci-dessous.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>Fichier XHTML minimaliste</title>
   </head>
   <body>
     <p>Un premier document en XHTML!</p>
 </body>
</html>
```

La première ligne décrit dans quelle version de XML est écrit le document ainsi que l'encodage utilisé pour les caractères. Si cette ligne manque au fichier, l'encodage par défaut est UTF-8. Il est recommandé de préciser l'encodage utilisé afin d'éviter les mauvaises surprises avec les caractères accentués et/ou spéciaux.
Les lignes 2 et 3 précisent dans quel standard est écrit le document. C'est le *Document Type Definition (DTD)* qui correspond à la grammaire du langage utilisé pour décrire le contenu. Dans le cadre de ce cours, nous avons décidé d'utiliser du XHTML 1.0 strict. Cette ligne est absolument nécessaire afin que le client Web soit capable de déterminer de quelle manière il doit interpréter le document. Un fichier ne contenant pas de DTD n'est pas valide.
Chaque document XHTML contient un en-tête et un corps, tous deux contenus entre les balises `<html>` et `</html>`. L'en-tête est défini entre les balises `<head>` et `</head>` et contient des informations comme le titre du document, des mots-clés, etc. (voir le paragraphe plus détaillé à ce sujet). Le corps du document HTML, où se trouve tout le contenu à afficher, est défini entre les balises `<body>` et `</body>`.

2.3 Deux types de balises: bloc ou en ligne

Le standard XHTML distingue deux types de balises: celle de niveau bloc (*block-level*) ou celles en ligne (*inline/text-level*). Les différences sont:

- par défaut les éléments de bloc sont disposés sur une nouvelle ligne tandis que les éléments en ligne sont disposés à la suite du contenu les précédant, sans retour à la ligne.
- en général, les éléments de bloc peuvent contenir d'autres éléments de bloc ou des éléments en ligne tandis que les éléments en ligne ne peuvent contenir que d'autres éléments en ligne.

2.4 Formattage du texte - éléments de niveau bloc

2.4.1 Titres, paragraphes, retours à la ligne et séparateurs

Le standard XHTML définit 6 niveaux de titres définis par les balises **<h>**:

```
<h1> Titre niveau 1 </h1>
<h2> Titre niveau 2 </h2>
<h3> Titre niveau 3 </h3>
<h4> Titre niveau 4 </h4>
<h5> Titre niveau 5 </h5>
<h6> Titre niveau 6 </h6>
```

Ce qui donne le résultat suivant:

Titre niveau 1
Titre niveau 2
Titre niveau 3
Titre niveau 4
Titre niveau 5
Titre niveau 6

Pour créer des paragraphes dans un document XHTML, il faut insérer le texte entre des balises ** ** et ** **:

```
<p> Un premier paragraphe contenant du texte sur plusieurs lignes.
Un premier paragraphe contenant du texte sur plusieurs lignes. Un premier
paragraphe contenant du texte sur plusieurs lignes. Un premier paragraphe
contenant du texte sur plusieurs lignes.</p>
<p> Le deuxième paragraphe est inséré entre ses propres balises. Il ne faut
pas oublier d'ajouter la balise de fin lorsque le paragraphe est terminé.</p>
```

Donnant le résultat suivant:

Un premier paragraphe contenant du texte sur plusieurs lignes. Un premier paragraphe contenant du texte sur plusieurs lignes. Un premier paragraphe contenant du texte sur plusieurs lignes. Un premier paragraphe contenant du texte sur plusieurs lignes.
Le deuxième paragraphe est inséré entre ses propres balises. Il ne faut pas oublier d'ajouter la balise de fin lorsque le paragraphe est terminé.

Vous pouvez constater que les retours à la ligne dans le document XHTML ne sont pas nécessairement les mêmes que dans le résultat affiché à l'écran. Afin de forcer un retour à la ligne, il faut utiliser la balise **
**.
De la même manière, les espaces dans le document ne sont pas reproduits exactement de la même manière (sauf dans le cas particulier de l'élément **<pre>**). Voici un exemple illustrant le phénomène:

```
<p> Voici un retour à la ligne dans le document HTML...
qui n'est pas nécessairement celui à l'écran. Pour forcer un retour à la ligne
il faut donc nécesairement utiliser la balise correspondante: <br />
Une autre ligne<br />
Une autre ligne<br />
Bon! Vous avez compris...
</p>
<p> Voici beaucoup d'espaces dans le document HTML: [             ] qui sont
affichés comme un seul espace par le client Web.</p>
```

Donnant le résultat suivant:

Voici un retour à la ligne dans le document HTML... qui n'est pas nécessairement celui à l'écran. Pour forcer un retour à la ligne, il faut donc nécesairement utiliser la balise correspondante:
Une autre ligne
Une autre ligne
Bon! Vous avez compris...
Voici beaucoup d'espaces dans le document HTML: [ ] qui sont affichés comme un seul espace par le client Web.

Il est aussi possible d'ajouter un séparateur horizontal dans le document à l'aide de la balise **<hr />** (*horizontal rule*). Voici un exemple:

```
<p> Un premier paragraphe contenant du texte sur plusieurs lignes.
Un premier paragraphe contenant du texte sur plusieurs lignes.</p>
<hr /> <!-- Le séparateur horizontal -->
<p> Le deuxième paragraphe en dessous du séparateur horizontal. Le deuxième
paragraphe en dessous du séparateur horizontal. Le deuxième paragraphe en
dessous du séparateur horizontal.</p>
```

Donnant le résultat suivant:

Un premier paragraphe contenant du texte sur plusieurs lignes. Un premier paragraphe contenant du texte sur plusieurs lignes.

Le deuxième paragraphe en dessous du séparateur horizontal. Le deuxième paragraphe en dessous du séparateur horizontal. Le deuxième paragraphe en dessous du séparateur horizontal.

2.4.2 Liste ordonnées, non ordonnées et définitions

Les listes ordonnées et non ordonnées sont définies respectivement par les balises `<ol>...</ol>` et `<ul>...</ul>`. Chaque item de la liste doit être inséré entre les balises `<li>...</li>` nécessairement à l'intérieur d'une liste ordonnée ou non ordonnée. Voici un exemple:

```
<p>Ceci est une liste non ordonnée:</p>
<ul>
 <li>premier élément,</li>
 <li>deuxième élément,</li>
 <li>dernier élément.</li>
</ul>
<p>Ceci est une liste ordonnée:</p>
<ol>
  <li>premier élément,</li>
  <li>deuxième élément,</li>
  <li>dernier élément.</li>
</ol>
```

Donnant le résultat suivant:

Ceci est une liste non ordonnée:

- premier élément,
- deuxième élément,
- dernier élément.

Ceci est une liste ordonnée:

1. premier élément,
2. deuxième élément,
3. dernier élément.

Il est aussi possible d'imbriquer plusieurs listes ordonnées ou non comme le montre l'exemple suivant:

```
<p>Ceci est une liste non ordonnée avec une sous-liste imbriquée:</p>
<ul>
  <li>premier élément, contenant la sous-liste suivante:
    <ol>
       <li> premier élément de la sous-liste ordonnée,</li>
      <li> deuxième élément de la sous-liste ordonnée,</li>
      <li> troisième élément de la sous-liste ordonnée,</li>
    </ol>
  </li>
  <li>deuxième élément,</li>
  <li>dernier élément.</li>
</ul>
```

Donnant le résultat suivant:

Ceci est une liste non ordonnée avec une sous-liste imbriquée:

- premier élément, contenant la sous-liste suivante:
    1. premier élément de la sous-liste ordonnée,
    2. deuxième élément de la sous-liste ordonnée,
    3. troisième élément de la sous-liste ordonnée,
- deuxième élément,
- dernier élément.

2.4.3 Tableaux

Avant de décrire en détail le fonctionnement des tableaux en XHTML, voici un exemple simple:

```
<table>
  <caption> Tableau 1 - Prix de quelques véhicules neufs ou usagés </caption>
  <tr> <!-- ligne 1 -->
    <th> Marque </th> <!-- Cellule-titre - colonne 1-->
    <th> Modèle </th> <!-- Cellule-titre - colonne 2-->
    <th> Année </th> <!-- Cellule-titre - colonne 3-->
    <th> Kilométrage </th> <!-- Cellule-titre - colonne 4-->
    <th> Prix de vente </th> <!-- Cellule-titre - colonne 5-->
```

```
  </tr>
  <tr> <!-- ligne 2 -->
    <td> Chevrolet </td> <!-- Cellule de données - colonne 1-->
    <td> Camaro </td> <!-- Cellule de données - colonne 2-->
    <td> 1978 </td> <!-- Cellule de données - colonne 3-->
    <td> 230000 km </td> <!-- Cellule de données - colonne 4-->
    <td> 3000$ </td> <!-- Cellule de données - colonne 5-->
  </tr>
  <tr> <!-- ligne 3 -->
    <td> Toyota </td>
    <td> Tercel</td>
    <td> 1998</td>
    <td> 100000 km </td>
    <td> 5000$</td>
  </tr>
  <tr> <!-- ligne 4 -->
    <td> Lotus</td>
    <td> Elise</td>
    <td> 2006</td>
    <td> 0 km</td>
    <td> 45000$</td>
  </tr>
</table>
```

Donnant le résultat suivant:

Tableau 1 - Prix de quelques véhicules neufs ou usagés

Marque	Modèle	Année	Kilométrage	Prix de vente
Chevrolet	Camaro	1978	230000 km	3000$
Toyota	Tercel	1998	100000 km	5000$
Lotus	Elise	2006	0 km	45000$

Plusieurs balises sont nécessaires afin de définir un tableau:

- **<table>**...**</table>** (lignes 1 et 31): définition du tableau qui contient ensuite des lignes et éventuellement un titre.
- **<tr>**...**</tr>** (*table row*): une ligne du tableau. Chaque ligne contient ensuite des cellules de données ou de titre. Les lignes sont insérées du haut vers le bas dans le tableau.
- **<td>**...**</td>** (*table data*): une cellule de données. Les cellules de données sont empilées de la gauche vers la droite dans une ligne du tableau.
- **<th>**...**</th>** (*table header*): une cellule de titre.
- **<caption>**...**</caption>**: définition du titre pour le tableau.

Tout comme dans un logiciel de traitement de texte, il est possible de fusionner des cellules afin qu'elles prennent plusieurs lignes et/ou colonnes. En XHTML, il faut utiliser les attributs **colspan** et **rowspan** pour indiquer sur combien de colonnes ou de lignes la cellule s'étend. Voici un exemple montrant différentes combinaisons:

```
<table>
  <caption> Tableau 2 - Tableau plus complexe </caption>
  <tr> <!-- ligne 1 -->
    <td colspan="3"> 1.1 à 1.3</td> <!-- cellule disposée sur 3 colonnes -->
    <td> 1.4</td>
    <td> 1.5</td>
  </tr>
  <tr> <!-- ligne 2 -->
    <td> 2.1</td>
    <td rowspan="2"> 2.2 et 3.2</td><!-- cellule disposée sur 2 colonnes -->
    <td> 2.3</td>
    <td> 2.4</td>
    <td> 2.5</td>
  </tr>
  <tr> <!-- ligne 3 -->
    <td> 3.1</td>
    <td> 3.3</td>
    <td rowspan="3" colspan="2"> 3.4 à 5.5</td> <!-- cellule sur 2 colonnes
                                                    et 3 lignes -->
  </tr>
  <tr> <!-- ligne 4 -->
    <td> 4.1</td>
    <td> 4.2</td>
    <td> 4.3</td>
  </tr>
  <tr> <!-- ligne 5 -->
```

```
    <td> 5.1</td>
    <td> 5.2</td>
    <td> 5.3</td>
  </tr>
</table>
```

Ce qui donne le résultat suivant:

Tableau 2 - Tableau plus complexe

<table>
<tr><td colspan="3">1.1 à 1.3</td><td>1.4</td><td>1.5</td></tr>
<tr><td>2.1</td><td rowspan="2">2.2 et 3.2</td><td>2.3</td><td>2.4</td><td>2.5</td></tr>
<tr><td>3.1</td><td>3.3</td><td colspan="2" rowspan="3">3.4 à 5.5</td></tr>
<tr><td>4.1</td><td>4.2</td><td>4.3</td></tr>
<tr><td>5.1</td><td>5.2</td><td>5.3</td></tr>
</table>

D'autres éléments existent pour définir un tableau. Parmi ceux-ci, les éléments **<col />** et **<colgroup>** permettent de définir des colonnes et groupes de colonnes. Ils sont particulièrement utiles pour définir un style pour une colonne complète. Un exemple détaillé se trouve dans la section sur les tableaux dans la partie consacrée aux feuilles de styles CSS.

2.4.4 Autres éléments de niveau bloc

D'autres éléments de niveau bloc sont présentés dans le tableau ci-dessous. Ils fonctionnent de la même manière que l'élément de paragraphe mais leur signification est légèrement différente.

Élément	Description
<pre>	Texte préformaté (généralement en caractères d'imprimerie). Les espaces apparaissent exactement comme spécifiés dans le fichier HTML.
<address>	Adresse/coordonnées de contact d'une personne, d'une organisation.
<blockquote>	Un bloc de citation long, généralement inséré entre des guillemets par le client Web.

2.5 Formattage du texte - éléments en ligne

2.5.1 Les éléments sémantiques

Comme leur nom indique, les balises sémantiques donnent une certaine signification au contenu qu'elles englobent. Celles-ci sont résumées dans le tableau suivant. Notez que la balise de fin est obligatoire dans tous les cas mais qu'elle n'est pas indiquée dans le tableau.

Élément	Description
****	Mettre de l'emphase sur une portion de texte.
****	Mettre de l'emphase plus forte sur une portion de texte.
<cite>	Un extrait ou une référence à une autre source.
<dfn>	Englobe une définition (manque de précision dans le standard).
<code>	Un fragment de code (source) informatique.
<samp>	Un exemple de sortie d'un programme informatique.
<kbd>	Une entrée au clavier par l'utilisateur.
<var>	Une variable ou un argument (programmation informatique).
<abbr>	Une abréviation.
<acronym>	Un acronyme.
<q>	Une citation courte.

L'usage systématique de l'élément correspondant (e.g. **<abbr>** dès qu'il y a une abbréviation dans le texte) n'est pas une obligation mais plus un choix de l'auteur en fonction du niveau de détail recherché. Historiquement, les éléments définis dans le standard sont particulièrement orientés vers un usage dans le domaine informatique, d'où l'existence de **<code>**, **<samp>**, **<kbd>**, **<var>**, etc.

2.5.2 Hyperliens

Les hyperliens sont à la base du XHTML. Sans eux, il ne serait pas possible de naviguer aussi simplement d'une page à Web l'autre. L'élément **<a>**...**** (*anchor*, en anglais) permets de créer un hyperlien à partir de la partie de document qu'elle englobe. Par exemple:

```
<p> <a href="http://www.mozilla.com"
     title="lien vers le site mozilla.com">Téléchargez Firefox</a>
et diffusez-le dans votre entourage! </p>
```

Ce qui donne le résultat suivant:

Téléchargez Firefox et diffusez-le dans votre entourage!

Le texte "Téléchargez Firefox" devient un hyperlien vers le site www.mozilla.com grâce à l'élément `<a>`. Deux attributs sont ajoutés dans la balise de début:

- `href` est l'attribut qui contient l'adresse de la ressource (l'URL) de l'hyperlien. Plus de détails sont donnés à ce sujet plus loin dans ce document.
- `title` est un attribut donnant un titre à l'hyperlien. Dans certains clients Web, celui-ci s'affiche sous forme d'une "infobulle" (*tooltip*) à l'écran.

2.5.3 Images

Les images sont couramment employées sur les sites Web. L'élément `<img />` permet d'insérer une image à la position de la balise dans le document. Notez qu'il n'y a pas de balise de fin pour cet élément. Comme l'image ne peut être directement copiée dans le fichier XHTML, l'élément `<img />` prends obligatoirement un attribut `src` qui définit l'URL de l'image à afficher.
Bien que la grande majorité des utilisateurs d'internet utilisent des clients Web permettant de visualiser les images, il est conseillé de fournir une information alternative sous forme de texte à l'aide de l'attribut `alt`. La description fournie doit correspondre au mieux au contenu de l'image. Cette description est essentielle pour les aveugles! Un non-voyant utilisant un client Web avec synthèse vocale entendra la description textuelle de l'image en consultant votre site web.
Les attributs `height` et `width` permettent de donner les dimensions de l'image. Deux remarques concernant l'usage de ces attributs:

- Ils ne sont pas obligatoires, mais permettent au client Web de réserver l'espace nécessaire à l'image avant même qu'elle soit chargée.
- Lorsque les dimensions de l'image ne correspondent pas à celles données dans les attributs, celle-ci est étirée ou réduite.

Voici un exemple montrant l'insertion d'une image avec les trois paramètres insérés:

```
<img src="http://www.w3.org/Icons/valid-xhtml10"
alt="Valid XHTML 1.0 Strict" height="31" width="88" />
```

Donnant le résultat suivant:

2.6 Les éléments génériques

Dans un contexte où les éléments sémantiques ne sont pas suffisants pour spécifier à quoi correspond le contenu, il est possible d'utiliser des éléments génériques `<div>` et `<span>` qui sont respectivement des éléments blocs et en ligne. Ces éléments sont essentiels à la construction d'un site Web valide selon W3C. D'ailleurs le XHTML5 est formé de nouvelles balises, basée sur un usage intensif de celles-ci. Des exemples de l'usage de `<div>` seront détaillés au laboratoire.

2.7 Les attributs `id` et `class`

Ces deux attributs sont particulièrement importants, dans la mesure où ils servent à plusieurs utilisations. Ils peuvent être ajoutés à n'importe quel élément. Il y a une différence importante entre les deux attributs qui est expliquée dans les paragraphes suivants.
L'attribut `id` permet de désigner un élément d'un document XHTML de manière **unique**. Cet identifiant est principalement utilisé dans deux situations:

- identifier un élément en particulier pour un usage dans les feuilles de styles et/ou dans des scripts du côté client Web (JavaScript par exemple);
- faire pointer un hyperlien vers cet élément en particulier du document.

L'attribut `class` permet d'assigner un ou plusieurs noms de classe à un élément. Contrairement à l'identifiant `id`, ces noms de classe peuvent être partagés entre plusieurs éléments, de même nature comme de nature différente. Ses deux rôles sont les suivants:

- ajouter une information sémantique quant au contenu de l'élément. Ceci est d'autant plus important lorsqu'on utilise un élément générique `<div>` ou `<span>` qui ne porte aucune information sémantique au départ.
- utiliser comme sélecteur pour les feuilles de styles.

2.8 L'attribut de titre `title`

L'attribut de titre peut être rajouté à tous les éléments afin de fournir des informations concernant l'élément. Par exemple, dans le cas d'un lien hypertexte, le titre permettrait de décrire la destination du lien.
Dans le cas d'un acronyme ou d'une abbréviation l'attribut de titre permet de donner la signification sous forme textuelle, comme dans l'exemple ci-dessous:

```
<p> Le <acronym title="RAdio Dectecting and Ranging">radar</acronym> était
installé en arrière de la bâtisse.</p>
```

Ce qui donne le résultat suivant:

> Le radar était installé en arrière de la bâtisse.

Notez que sous Firefox, une "infobulle" (*tooltip*) apparaît avec le contenu de l'attribut de titre lorsqu'on pointe le curseur sur l'élément.

2.9 La (re)construction des URL par le client Web

Dans le paragraphe concernant les liens hypertextes, un exemple d'URI a été donné. Le but de cette partie est d'expliquer un peu plus précisément comment est construit un URL, et surtout, comment inscrire un URI dans un document XHTML. L'image suivante montre un URL absolu pointant vers un élément particulier (dont l'attribut `id` a la valeur `id_element`) d'une page XHTML nommée `page.html`.

Lors de la rédaction des pages Web, il serait possible d'utiliser systématiquement des URL absolus pour identifier chacun des fichiers d'un site Web mais ceci a plusieurs inconvénients:

- L'URL absolu est long à écrire.
- Il suffit de déplacer un site Web vers un autre domaine (e.g. une nouvelle adresse symbolique) pour que tous les liens soient à réécrire.
- Il suffit de vouloir déplacer une partie d'un site vers un autre sous-répertoire pour que tous les URL correspondants soient à changer.

Partant de ce constat, dans le but de simplifier/alléger l'écriture des URL, les créateurs du XHTML ont inclus la possibilité d'écrire des URL relatifs à la page courante ou bien par rapport à une référence déterminée par l'auteur de la page (voir le paragraphe sur l'élément `<base />` dans la section suivante). Afin d'illustrer les URL relatifs par un exemple, considérons l'arborescence suivante:

Les URL absolus de tous les éléments représentés, dans l'ordre, sont:

- `http://www.exemple.com/index.html`
- `http://www.exemple.com/repertoire1/index.html`
- `http://www.exemple.com/repertoire1/page1.html`
- `http://www.exemple.com/repertoire1/images/image1.png`
- `http://www.exemple.com/repertoire1/images/image2.png`

Dans le document HTML `page1.html`, un hyperlien pointant vers la page `index.html` dans le même répertoire pourrait utiliser les URL suivants:

- `index.html`
- `./index.html`
- `/repertoire1/index.html`

L'image `image1.png` est insérée dans le document HTML `page1.html`. L'élément `<img />` pourrait utiliser les URL relatifs suivants:

- `images/image1.png`
- `./imagesE/image1.png`
- `/repertoire1/images/image1.png`

Dans le document HTML `page1.html`, un hyperlien pointant vers la page `http://www.exemple.com/index.html` pourrait utiliser les URL relatifs suivants:

- `../index.html`
- `/index.html`

Dans le document XHTML `page1.html` se trouve un élément dont l'attribut `id` a pour valeur *id_element*. Un hyperlien dans le document `http://www.exemple.com/index.html` voulant pointer vers cet élément en particulier pourrait utiliser les URL suivants:

- `http://www.exemple.com/repertoire1/page1.html#id_element` (URI absolu)
- `/repertoire1/page1.html#id_element`
- `./repertoire1/page1.html#id_element`
- `repertoire1/page1.html#id_element` (Notez l'absence de caractère `/` au début de cet URI relatif)

Dans le document html `page1.html`, un hyperlien voulant pointer vers cet élément en particulier pourrait utiliser les URL suivants:

- `http://www.exemple.com/repertoire1/page1.html#id_element` (URL absolu)
- `#id_element`

2.10 L'en-tête d'un document XHTML

L'en-tête d'un document XHTML contient des informations importantes concernant le document XHTML: le titre du document, des mots-clés, des métadonnées. Aucune information présente dans l'en-tête n'est directement affichée. Comme présenté dans la section "Structure générale d'un fichier XHTML", l'en-tête est défini entre les balises `<head>...</head>`.

2.10.1 Le titre du document

Contrairement à l'attribut `title` qui définit un titre à un élément en particulier, l'élément `<title>...</title>` englobe le titre du document. Dans les navigateurs Web, cette information s'affiche dans la barre de titre de l'application.

2.10.2 Métadonnées

Les méta-données sont plutôt variées. Il ne serait pas utile de décrire en détail tout ce qui peut y être mis dedans. L'élément `<meta />` contient généralement les deux attributs `name` et `content` correspondant respectivement au nom et au contenu de la méta-information décrite par l'élément. Voici un exemple expliqué juste après:

```
<meta name="author" content="Stephane Brunet" />
<meta name="keywords" content="introduction,XHTML,MEC1310" />
<meta name="description" content="Une introduction au XHTML pour le cours MEC1310" />
```

Bien qu'il soit possible d'utiliser n'importe quels mots pour l'attribut `name`, il en existe un certain nombre plus courant:

- *author*: l'auteur du document.
- *keywords*: une liste de mots-clés sur le contenu du document. Traditionnellement, les moteurs de recherche utilisaient cette information pour l'indexation du site. Actuellement, l'indexation est faite à l'aide du contenu complet du document HTML.
- *generator*: le logiciel qui a généré la page HTML.
- *description*: une description textuelle du document.

Dans certaines situations, l'attribut `http-equiv` peut remplacer l'attribut `name`. Ceci donne alors l'information au serveur Web d'ajouter/modifier l'information contenue dans l'en-tête de la requête HTTP.

2.10.3 L'élément `<base />`

L'attribut `href` contenu dans l'élément `<base />` permet de donner une référence explicite à partir de laquelle les URI absolus sont construits depuis les URI relatifs. Dans le cas de l'exemple donné à la section "La (re)construction des URI par le client Web", il aurait été possible de rajouter la ligne suivante dans l'en-tête de chaque document XHTML:

```
<base href="http://www.exemple.com/" />
```

Le résultat aurait été le même pour les URL absolus ou relatifs commençant par un caractère `/`. Par contre, dans le cas du document `page1.html`, à l'intérieur du répertoire `repertoire1`, l'URL relatif `images/image1.png` ne pointerait plus sur l'URL absolu `http://www.exemple.com/repertoire1/images/image1.png` mais plutôt sur l'URL absolu `http://www.exemple.com/images/image1.png`.
L'élément `<base />` peut être utile pour simplifier l'écriture des URL relatifs mais peut compliquer le développement du site sur une arborescence temporaire du disque dur (ce qu'on fait effectivement dans le cadre du cours). Il faudrait en effet modifier le contenu de l'élément de chaque document XHTML lors de la publication sur le serveur Web.
Pour reprendre l'exemple du site `www.exemple.com`, si l'arborescence du site est copiée sur un disque dur local et qu'on essaye de visualiser une page hors connexion, aucun lien URL ne fonctionnerait, car ils pointeraient tous vers un fichier sur internet et non sur le disque dur.

2.11 Où trouver de l'information supplémentaire ?

La première source d'information concernant le XHTML est l'organisme W3C qui s'occupe de gérer de nombreux standards, dont le (X)HTML:

- le standard HTML 4.01, dernière version du HTML avant l'avènement du XML et du XHTML.
- le standard XHTML 1.0, qui définit les différences entre le HTML 4.01 et le XHTML 1.0 sans décrire en détail l'usage des différents éléments.

3 Feuilles de styles en cascade (CSS)

3.1 Concepts de base du CSS

L'intérêt majeur des feuilles de styles CSS est de séparer le contenu de la présentation. À l'origine, le HTML comportait des balises pour modifier le style de texte (police, graisse, taille, couleur, arrière-plan) mélangé au contenu. Avec une telle approche, deux problèmes majeurs peuvent apparaître:

- Les attributs et balises de style sont complètement noyés dans le contenu, ce qui rend la tâche de mise à jour du style plus difficile.
- Un site complet doit nécessairement avoir une certaine homogénéité dans sa conception visuelle. Lorsqu'une page est rajoutée, il faut s'assurer d'utiliser les mêmes éléments de style que les pages existantes. Ou bien, lorsque la conception graphique doit être modifiée, toutes les pages doivent nécessairement être mises à jour. La tâche est d'autant plus complexe et laborieuse que le site est grand...

Qu'est-ce que le contenu ? Essentiellement, toute entité qui donne de l'information à l'usager peut être considérée comme du contenu. La structure, l'organisation de l'information sont aussi quelque chose qui appartient au contenu. Normalement, quel que soit le style de présentation qu'on impose, le contenu devrait rester inchangé.
Le style, au contraire, n'est pas essentiel à la compréhension du document. Par exemple, si on supprimait le style sur un document, tout le contenu, tant les illustrations que le texte restent compréhensibles. Par contre, la disposition du texte et des illustrations, le style des paragraphes, l'image de fond qui sert à "embellir" la présentation ne sont pas complètement indispensables.

3.2 Un style CSS appliqué à un document XHTML

Un document XHTML simple est présenté ci-dessous:.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>Fichier XHTML pour exemple CSS</title>
    <link href="exemple1-CSS.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <p> Ceci est un premier paragraphe du document XHTML. Ceci est un premier
    paragraphe du document XHTML. Ceci est un premier paragraphe du document XHTML.
    Ceci est un premier paragraphe du document XHTML. Ceci est un premier paragraphe
    du document XHTML.</p>
    <p> Le second paragraphe introduit une liste ordonnée:</p>
    <ol>
      <li> Une fonction matlab: <code>plot3d</code></li>
      <li> Une fonction C: <code>sprintf()</code></li>
    </ol>
  </body>
</html>
```

La seule différence entre ce fichier et ceux présentés dans la partie du document consacrée au XHTML se situe dans l'en-tête du document, à la ligne 7. Un élément **`<link />`** est ajouté afin de faire référence à la feuille de style à appliquer à ce document. Les attributs utilisés sont les suivants:

- **`href`**: L'URI de la feuille de style.
- **`rel`**: La relation entre le document pointé par l'URI et ce document XHTML. Comme c'est une feuille de style, la valeur **`stylesheet`** lui est attribuée.
- **`type`**: la nature du document pointé. C'est une chaîne de caractères standard décrivant le type de fichier (*MIME content type*). Dans le cas d'une feuille de style CSS, le type est **`text/css`**.

Le fichier CSS **`exemple1-CSS.css`** pourrait contenir, par exemple, les styles suivants:

```
@charset "ISO-8859-1";

body {
  font-family: Trebuchet, Helvetica, sans-serif;
  font-size: 15px;
}

p {
  color: #5500FF;
  text-align: justify;
  text-indent: 2em;
}

ol {list-style-type: lower-roman; }

code {font-weight: bold;}
```

La signification des lignes du fichier exemple1-CSS.css est donnée ci-dessous:

- La ligne 1 désigne l'encodage des caractères utilisé pour le fichier.
- Ligne 3 à 6 - style appliqué à l'élément **`body`**:
  - **`font-family`**: Liste de polices de caractères à utiliser. Selon ce qui est disponible sur l'ordinateur, le navigateur va choisir dans la liste, de gauche à droite.
  - **`font-size`**: donne la hauteur de caractères à utiliser. Ici, elle est fixée à 15 pixels.
- Ligne 8 à 12 - style appliqué aux éléments **`p`**:
  - **`color`**: couleur du contenu (p.ex. du texte) donné sous forme d'un code RGB (composantes rouge, vert et bleu), expliqué plus loin dans ce document.
  - **`text-align`**: alignement du texte du paragraphe (ici, justifié).
  - **`text-indent`**: indentation de la première ligne du paragraphe. Ici fixée à 2em (2 fois la taille de caractères).
- Ligne 14 - style appliqué aux éléments **`ol`**:
  - **`list-style-type`**: style de numérotation (ici, en chiffres romains minuscules).

- Ligne 16 - style appliqué aux éléments `code`:
  - `font-weight`: choix de la graisse d'une police (ici, en gras).

3.3 La sélection des éléments

Dans l'exemple précédent, les propriétés sont appliquées à un ensemble d'éléments de même nature, mais le standard CSS permet de raffiner la sélection de plusieurs manières. Par exemple, il est possible de sélectionner tous les éléments appartenant à une classe commune (p.ex. les éléments ayant un attribut `class` commun). Le tableau ci-dessous résume les principales possibilités (se référer à la spécification CSS2.1 pour plus de détails):

Séquence	Signification
E	Sélectionne tous les éléments E
E F	Sélectionne tous les éléments F qui sont des *descendants* de E
E > F	Sélectionne tous les éléments F qui sont des *enfants* de E
E:link E:visited	Dans le cas d'hyperliens, sélectionne les éléments E qui n'ont pas été visités (`E:link`) ou déjà visités(`E:visited`)
E:hover	Cette séquence est utilisé pour sélectionner un élément E pointé avec le curseur mais pas sélectionné avec la souris.
E:active	Cette séquence est utilisé pour sélectionner un élément E sélectionné avec la souris.
E.nom_classe	Sélectionne tous les éléments E appartenant à la classe `nom_classe` (attribut `class`).
E#mon_id #identifiant	Sélectionne l'élément E ayant comme identifiant `mon_id` (attribut `id`).

Parmi les séquences présentées ci-dessus, il faut bien comprendre la nuance entre les *descendants* et les *enfants*:

- Les *descendants* d'un élément E sont tous les éléments contenus dans cet élément E.
- Les *enfants* d'un élément E sont tous les descendants directs de cet élément E.

En reprenant l'exemple de la section précédent, tous les éléments `code` et `li` présents sont des descendants de l'élément `ol`. Tous les éléments `li` présents sont des enfants de l'élément `ol`. Mais, par contre, les éléments `code` ne sont pas des enfants de l'élément `ol` car ils sont à l'intérieur des éléments `li`.

3.4 Unités de longueur, couleurs, pourcentages et URI

Dans un fichier CSS, il faut être en mesure d'affecter des valeurs aux propriétés voulues. Non seulement il existe des chaînes de caractères, des nombres entiers ou décimaux (dont le séparateur décimal est le point), mais il est parfois indispensable de spécifier une unité, une couleur, un URI vers une image, par exemple.

3.4.1 Les unités de longueur

Parmi les unités de longueur, il faut considérer deux types: les relatives et les absolues. Parmi les unités absolues, il y a celles couramment utilisées dans la vie de tous les jours et dans l'imprimerie:

- les pouces, représentés par les caractères `in`;
- les centimètres et millimètres, représentés respectivement par les caractères `cm` et `mm`;
- Les points, unités d'imprimerie, correspondants à 1/72 pouce et représentés par les caractères `pt`;
- Les picas, unités d'imprimerie, correspondants à 12 points et représentés par les caractères `pc`;

Parmi les unités relatives se retrouvent une unité informatique et aussi des unités d'imprimerie:

- les pixels, contraction de *picture element*, correspondent à un point lumineux d'un écran et sont représentés par les caractères `px` C'est une unité relative car l'écran a une certaine résolution (96 pixels par pouce généralement). Attention à une petite subtilité de la norme CSS: logiquement, si on parle d'un pixel pour une imprimante à 300ppp, un pixel serait un point de 1/300 pouce. Afin de garder une certaine cohérence, le standard CSS recommande que le navigateur ajuste la taille pour représenter sur l'imprimante une dimension *correspondant à ce qu'on voit à l'écran*.
- les `em`, correspondant à la taille de la police de caractère. Par exemple, pour un élément ayant une police de taille 14pt, 1em=14pt. 1.5em correspondrait à 150% de la taille de la police.
- les `ex`, définis comme la hauteur du caractère `x` pour la police de l'élément en question.

Il est possible de mélanger les unités dans un même fichier mais il est conseillé d'utiliser au maximum les unités relatives (ou les pourcentages, comme expliqué plus loin). Voici un exemple:

```
body {
  font-size: 15px;
  margin-top: 5ex;
  margin-bottom: .125in;
}
p {
  font-size: 2em;
  margin: 10mm;
}
```

Seules les lignes 2 et 8 utilisent des unités relatives. Dans le cas de la ligne 2, la marge du dessus (voir section suivante) est fixée à 5 fois la taille du caractère.

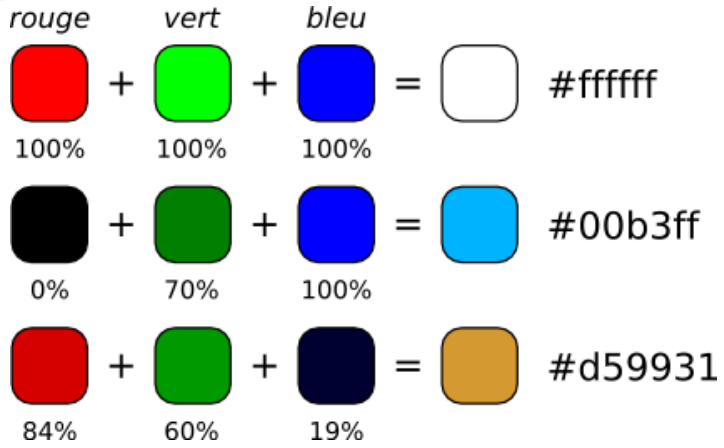
La seule particularité de cet exemple se trouve à la ligne 8. Une unité *relative à la taille de caractères* est utilisée pour justement fixer la taille de la police des éléments `p`. Dans ce cas, la taille de caractère de l'élément parent est utilisée comme référence. C'est-à-dire que pour les paragraphes se trouvant directement dans l'élément `body`, la taille sera 2 fois plus grande.

3.4.2 Définir une couleur

La manière de définir une couleur dans une feuille de style est calquée directement sur un concept courant un informatique: la synthèse additive de couleurs à partir des trois couleurs de base rouge, vert et bleu. En dosant la quantité de chaque, il est possible de générer une infinité de couleurs. Notez qu'il existe aussi quelques couleurs prédéfinies.
Plusieurs possibilités sont offertes pour définir la quantité de chaque composante pour une couleur donnée:

- `rgb(`*rouge*`,`*vert*`,`*bleu*`)` où *rouge*, *vert* et *bleu* sont respectivement l'intensité des composantes rouge, verte et bleue sous forme d'une valeur entière décimale entre 0 (intensité mini.) et 255 (intensité maxi)
- `#rgb` où `r`, `g` et `b` sont respectivement l'intensité des composantes rouge, verte et bleue sous forme d'une valeur hexadécimale entre 0 (mini) et F (maxi). Il y a donc 16 niveaux d'intensité pour chaque composante d'où 4096 couleurs différentes.
- `#rrggbb` où `rr`, `gg` et `bb` sont respectivement l'intensité des composantes rouge, verte et bleue sous forme d'une valeur hexadécimale entre 0 (mini) et FF (maxi). Il y a donc 256 niveaux d'intensité pour chaque composante d'où plus que 16 milions de couleurs différentes.

La figure ci-dessous montre quelques exemples:



3.4.3 L'usage de pourcentages

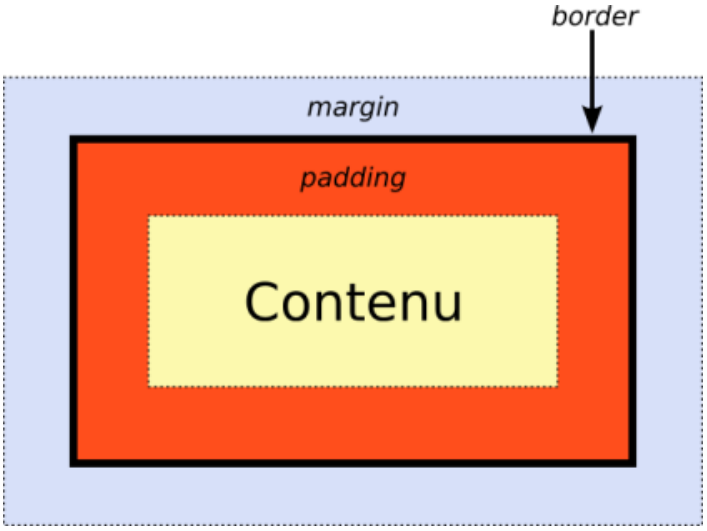
Dans n'importe quel cas où une grandeur relative peut être utilisée, il est possible d'utiliser des pourcentages. Par exemple, pour une longueur, au lieu de `1.5em` il serait tout à fait acceptable d'écrire `150%`. De même pour les intensités de couleurs, au lieu de `rgb(0,128,255)`, il serait tout à fait acceptable d'écrire `rgb(0%,50%,100%)`.

3.4.4 Définir un URI

Il est parfois nécessaire de référer à l'intérieur de la feuille de style à un fichier séparé. Par exemple, plutôt que de donner une couleur de fond à la page, on choisit de disposer une image. De la même manière que pour le (X)HTML, c'est un URI qu'il faut écrire, de la manière suivante: `url(`*adresse*`)` où *adresse* est un URI relatif à la page CSS ou un URI absolu.

3.5 La mise en page selon CSS - un empilement de boîtes

Tous les éléments d'un fichier (X)HTML sont considérés comme des boîtes disposées les unes par rapport aux autres. Dans la partie sur le XHTML, la notion d'élément en bloc (*block-level*) ou en ligne (*in-line*) était abordée. Par défaut, les éléments en blocs s'empilent les uns en dessous des autres (comme les paragraphes, par exemple) tandis que les éléments en ligne (le texte contenu dans le paragraphe, les éléments `a`, etc.) sont empilés dans le sens de lecture, c'est-à-dire de gauche à droite puis de haut en bas, dans le cas de nos langues officielles. Noter qu'il est possible de modifier la manière dont les éléments sont disposer les uns les autres à l'aide de CSS (voir la section sur le positionnement CSS).
Les blocs ont des caractéristiques communes présentées sur l'image ci-dessous:



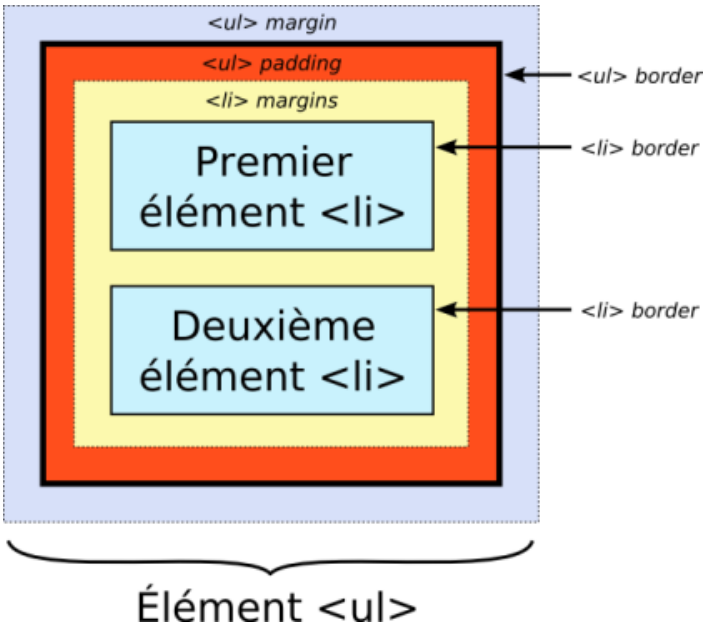
Chaque boîte a les caractéristiques suivantes:

- Le contenu est tout ce qui est à l'intérieur de l'élément.
- Entre le contenu et la bordure (visible ou non) se trouve une marge intérieure appelée *padding*. Il est possible de spécifier une couleur ou une image de fond à l'intérieur de ce cadre, y compris sous le contenu.
- De l'autre côté de la bordure se trouve une marge extérieure transparente qui permet de distancer la boîte des boîtes adjacentes.

Pour illuster l'interaction entre les boîtes correspondant à des éléments d'un document XHTML, voici une portion de code XHTML représentant une liste non ordonnée:

```
<ul>
  <li> Premier élément li </li>
  <li> Deuxième élément li </li>
</ul>
```

L'empilement des boîtes des éléments `li` à l'intérieur de l'élément `ul` donne:



Pour n'importe quel élément (à quelques exceptions près) il est possible d'ajuster les propriétés suivantes d'une boîte:

- Le type, l'épaisseur, la couleur pour la bordure de la boîte, soit pour les quatre côtés à la fois ou bien chaque côté individuellement.
- La marge extérieure (*margin*), soit pour les quatre côtés à la fois ou bien chaque côté individuellement.
- La marge intérieure (*padding*), soit pour les quatre côtés à la fois ou bien chaque côté individuellement.
- Le type d'arrière-plan de la boîte: transparent, une couleur unie, une image.
- Le type d'arrière-plan de la boîte: transparent, une couleur unie, une image.

Voici un exemple complet. D'abord le fichier XHTML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>Fichier XHTML pour exemple CSS</title>
    <link href="exemple2-CSS.css" rel="stylesheet" type="text/css" />
```

```
  </head>
  <body>
    <div id="ex1">Contenu</div>
    <div id="ex2">Contenu</div>
    <div id="ex3">Contenu</div>
  </body>
</html>
```

Et le fichier CSS **exemple2-CSS.css**:

```
@charset "ISO-8859-1";
body {
  font-family: Trebuchet, Helvetica, sans-serif;
  font-size: 15px;
  background-image: url(exemple2-CSS-fond.png);
}
div {
  width: 15em;
  text-align: left;
  color: black;
  font-weight: bold;
}
div#ex1 {
  padding: 5mm;
  border-style: solid;
  border-width: 8px;
  border-color: #d59931;
  background-color: #00b3ff;
}
div#ex2 {
  padding: 5mm;
  border-style: dotted;
  border-width: 5px;
  border-color: black;
  background-image: url(exemple2-CSS.png);
  margin: 0.2in;
}
div#ex3 {
  padding: 5mm;
  border-style: solid;
  border-width: 1px;
  border-color: red;
  background: transparent;
}
```

Le résultat obtenu est approximativement le suivant (le contenu du document a été transféré dans un élément **div** ayant les mêmes propriétés que l'élément **body** de l'exemple).

Contenu
Contenu
Contenu

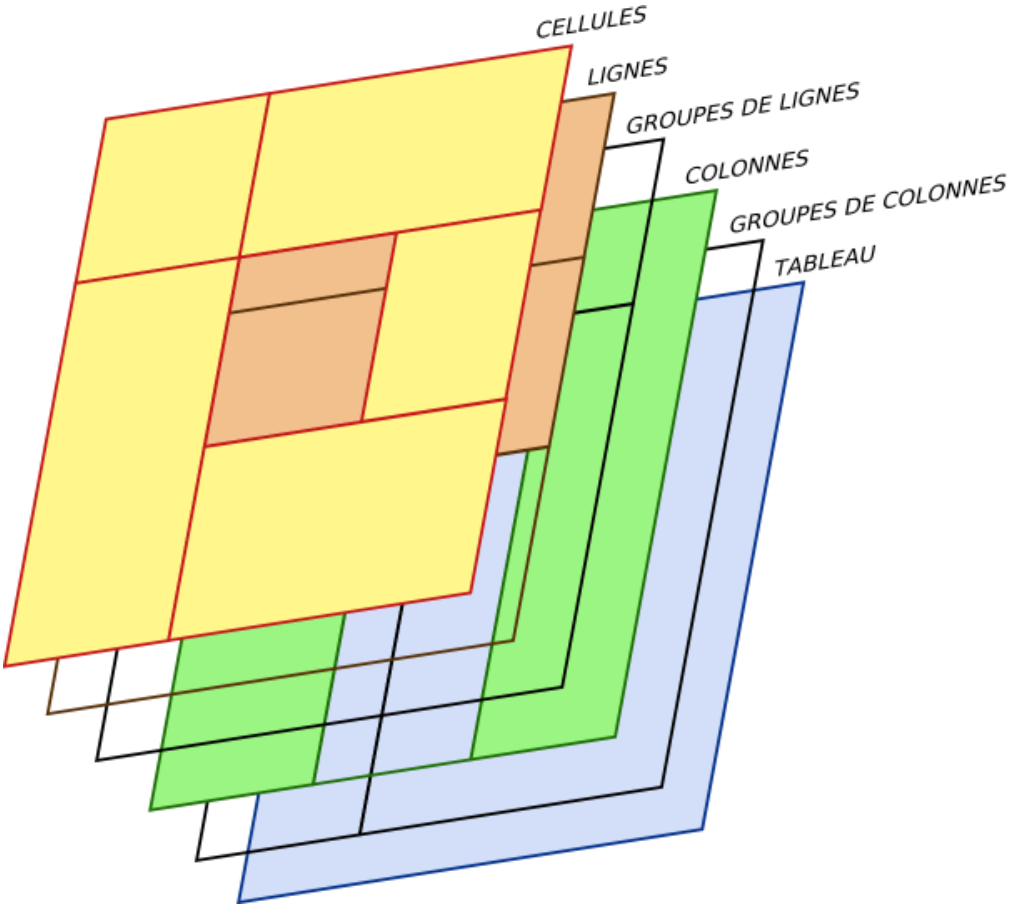
3.6 Mise en forme du texte

Les paramètres de style qui influencent la mise en forme du texte sont les suivants:

- style de police de caractère (**font-***);
- couleur de l'élément contenant (**color**);
- le soulignement, texte barré ou avec une ligne au-dessus (**text-decoration**);
- l'espacement entre les mots ou les lettres (**letter-spacing** et **word-spacing**);
- transformation du texte - capitalisé, tout en majuscules ou minuscules (**text-transform**);
- caractéristiques d'un paragraphe (pas au sens de l'élément **p**):
    - l'indentation de la première ligne (**text-indent**);
    - la position du texte - à gauche, à droite, centré ou justifié (**text-align**);
    - l'interligne (**line-height**).

3.7 Mise en forme des tableaux

La mise en forme des tableaux peut paraître compliquée au premier abord, mais suit une procédure très logique. Le résultat final peut être vu sous forme d'un empilement de six couches comme le présente la figure suivante:



Pour chacune des couches, lorsqu'une partie du fond est transparente, les couches en dessous peuvent apparaître. L'exemple suivant a été repris et adapté du document officiel sur le CSS 2.1. Voici tout d'abord la portion XHTML:

```
<table>
  <col id="colonne1" /><col id="colonne2" /><col id="colonne3" />
  <tr>
    <td> 1 </td>
    <td> 2 </td>
    <td> 3 </td>
  </tr>
  <tr>
    <td> 4  </td>
    <td class="cellule5"> 5 </td>
    <td class="cellule6"> 6 </td>
  </tr>
  <tr>
    <td> 7 </td>
    <td> 8 </td>
    <td> 9 </td>
  </tr>
  <tr>
    <td> 10 </td>
    <td> 11 </td>
    <td> 12 </td>
  </tr>
  <tr>
    <td> 13 </td>
    <td> 14 </td>
    <td> 15 </td>
  </tr>
</table>
```

Et la partie CSS pour appliquer le style:

```
table {
  border-collapse: collapse;
  border: 5px solid aqua;
  background-color: silver;
}
```

```
#colonne1 { border: 3px solid black; }
td { border: 1px solid red; padding: 1em; }
td.cellule5 {
  border: 5px dashed blue;
  background-color: yellow;
}
td.cellule6 { border: 5px solid green; }
```

Le résultat est le suivant:

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Tous les éléments entrant dans la composition d'un tableau ont les propriétés d'une simple boîte. Toutefois, il existe quelques variantes dans un tableau dont quelques-unes nécessitent une attention particulière:

- La propriété **`border-collapse`** d'un élément **`table`** permet de choisir entre deux types de disposition des bordures des cellules:
  - *separate* où chaque cellule a des bordures séparées des cellules voisines (dans la mesure où la marge extérieure est non nulle).
  - *collapse* où les cellules partagent la même bordure que les cellules contiguës. Le standard CSS décrit comment gérer les conflits entre deux éléments contiguës essayant de définir une propriété différente pour leur bordure commune.
- Des bordures ayant leur propriété **`border-style`** fixé à *hidden* ne seront jamais affichée, même si elles sont communes à plusieurs éléments ayant des styles de bordures différentes.
- Il est possible de choisir si les cellules vides doivent ou non être affichées en modifiant la propriété **`empty-cells`** aux éléments **`table`**, **`th`** et/ou **`td`**. Les deux options sont *show* (montrer) et *hide* (cacher).
- Les positions horizontale et verticale du texte dans une cellule sont ajustées avec les propriétés **`text-align`** et **`vertical-align`** respectivement.
- La position du titre du tableau est définie avec la propriété **`caption-side`** prenant soit la valeur *top* (titre au-dessus du tableau) ou *bottom* (titre en dessous du tableau).

3.8 Positionnement CSS

Le positionnement CSS est la partie du standard CSS qui est la plus compliquée à mettre en oeuvre. Il est en effet possible de modifier complètement la disposition des éléments d'une page à l'aide de quelques propriétés particulières dans le fichier CSS.
Le positionnement peut être fait de deux manières:

- *relatif* où la boîte contenant l'élément est d'abord disposée normalement puis déplacée selon les propriétés définies dans le fichier CSS. Les éléments autours ne changent pas de position.
- *absolu* où la boîte contenant l'élément positionné est disposée sur l'écran en donnant les coordonnées de l'objet. Ce type de positionnement n'est pas traité dans ce document.
- *flottant* où la boîte contenant l'élément positionné est extraite du contenu pour être disposée à droite ou à gauche.

3.8.1 Exemple de positionnement relatif

L'exemple qui suit montre un positionnement relatif. Voici la portion XHTML:

```
<p> Ceci est un paragraphe contenant plein de texte, toujours du texte.
Ceci est un paragraphe contenant plein de texte, toujours du texte.
Ceci est un paragraphe contenant plein de texte, toujours du texte.
<span id="particulier"> Texte spécial!</span>
Ceci est un paragraphe contenant plein de texte, toujours du texte.
Ceci est un paragraphe contenant plein de texte, toujours du texte.
</p>
```

Et la partie CSS pour appliquer le style:

```
#particulier {
  position: relative;
  top: -10px;
  left: 15px;
  color: red;
  border: 1px solid red;
  font-weight: bold;
}
```

Le résultat est le suivant:

Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte. Texte spécial! Ceci est un

paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte.

Notez la position du texte contenu dans l'élément **span** qui a été décalé et superposé au reste du texte. Afin d'obtenir ce résultat, il faut utiliser les propriétés suivantes:

- ligne 2: la propriété **position** a la valeur *relative* afin d'informer le navigateur qu'il doit effectuer un positionnement relatif.
- lignes 3 et 4: les propriétés **top** et **left** donnent la position relative de l'élément par rapport à sa position initiale. On aurait aussi pu utiliser les propriétés **bottom** ou **right**.

3.8.2 Exemple d'élément flottant

L'exemple suivant présente un élément **div** rendu flottant. La partie XHTML est présentée ci-dessous:

```
<p> <img src="./imagesE/image.png" id="boite" /> Ceci est un paragraphe contenant
plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de
texte, toujours du texte. Ceci est un paragraphe contenant plein de texte,
toujours du texte. Ceci est un paragraphe contenant plein de texte,
toujours du texte. Ceci est un paragraphe contenant plein de texte.
</p>
```

Et la partie CSS pour appliquer le style:

```
#boite {
  float: left;
  border: 2px solid red;
  padding: 4px;
  margin: 6px;
}
```

Le résultat est le suivant:


Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte, toujours du texte. Ceci est un paragraphe contenant plein de texte.

La seule propriété indispensable dans ce cas est **float** qui doit prendre la valeur *left* ou *right* selon que l'élément doit être positionné à gauche ou à droite. Il est quelquefois nécessaire d'ajuster la largeur de l'élément flottant à l'aide de la propriété **width**.

3.9 Quelques conseils

Ce document sur CSS 2.1 est un survol des possibilités offertes. Pour avoir une meilleure idée de ce qui est faisable, le document officiel sur le CSS 2.1 publié par le W3C est la source d'information à privilégier. Pour les personnes trouvant le document officiel un peu difficile d'approche, le site brainjar.com a une excellente introduction à CSS, en plus de fournir des introductions à des outils non abordés dans le cadre de ce cours.
Comme toutes les propriétés existantes n'ont pas été décrites, il est utile de se référer directement à l'index des propriétés disponible, à l'annexe F du document officiel sur le CSS 2.1, puis d'utiliser les hyperliens pour aller lire la section particulière du document où une description plus détaillée est fournie.
L'usage de CSS en respectant rigoureusement les standards ne garantit malheureusement pas qu'un affichage convenable de la page Web sur tous les navigateurs du marché. Il faut donc la tester, dans la mesure du possible, avec un nombre maximum de navigateurs différents (et aussi des différentes versions du même navigateur) pour vérifier si le résultat est acceptable.
Le site web CSS Zen Garden présente un exemple d'utilisation poussé à l'extrême de CSS. Pour la même page, plusieurs styles peuvent être choisis, présentant chacun une disposition graphique différente. Le niveau de complexité est tel qu'il demande beaucoup d'expérience avant d'en arriver à un résultat semblable.
Il est quand même possible de réaliser des sites tout à fait acceptables sans être un designer Web chevronné. Beaucoup de designers d'expérience écrivent des articles sur le bon usage du standard CSS mis gratuitement à la disposition de la communauté internaute.

4 Validation des documents Web

4.1 Valideur XHTML

Le W3C offre un outil en ligne permettant de vérifier si une page HTML ou XHTML est conforme au standard utilisé. C'est le HTML Validator qui propose trois options:

- Entrer l'URL d'une page, qui doit donc être publiée afin d'être accessible par le HTML Validator.
- Transférer un fichier vers le HTML Validator. C'est la méthode à employer pour vérifier une page qui est enregistrée sur un disque local.

- Entrer le code (X)HTML dans la zone de texte (Méthode à laisser tomber au profit des précédentes).

4.2 Valideur CSS

Le CSS Validator est l'équivalent du HTML Validator pour les fichiers CSS. Notez que la vérification des deux est indépendante! Les options du service de validation CSS sont tout à fait semblables au service pour le (X)HTML.

4.3 Que faire s'il y a des erreurs ?

Les messages fournis par les services de validation ne sont pas toujours clairs. L'erreur affichée ne concerne pas toujours l'élément pointé par le valideur mais bien souvent quelque chose juste avant. Par exemple, si le fichier XHTML comporte une nouvelle nouvelle balise `<p>` alors que le paragraphe précédent n'a pas été terminé par sa balise de fin `</p>`, le valideur va afficher l'erreur au niveau du nouveau paragraphe qui semble débuter à l'intérieur du paragraphe précédent.
Il se peut que corriger une erreur au début de la liste d'erreurs fournies par le valideur en élimine quelques-unes par la suite. Il ne faut donc pas paniquer si de nombreuses erreurs apparaissent, mais plutôt essayer de les corriger une à une dans l'ordre chronologique.