

# HTML - CSS

Ivan Kurzweg

29 mars 2007

**HTML - CSS**

by Ivan Kurzweg

Copyright © 2006 Ivan KURZWEG, *ivan.kurzweg@wanadoo.fr*

Permission to use, copy, modify, and distribute this documentation for any purpose with or without fee is here by granted, provided that the above copyright notice and this permission notice appear in all copies.

## Table des matières

<b>1</b>	<b>HTML / XHTML - Hypertext Markup Languages</b>	<b>3</b>
1.1	XHTML vs HTML vs SGML vs XML vs CSS!	3
1.1.1	Historique	3
1.2	Le W3C	4
1.2.1	Historique	4
1.2.2	Des standards?	4
1.2.3	Validation du code	5
1.3	La structure du langage	5
1.3.1	Langage de balisage	5
1.3.2	Attributs	6
1.3.3	Identifiant / Classe	6
1.3.4	Emboitements	6
1.3.5	La documentation du W3C	7
1.4	Squelette d'un fichier XHTML	7
1.4.1	La DTD	8
1.4.2	Les balises d'entêtes	8
1.4.3	Convention de nommage des fichiers	8
1.5	Exercices d'applications	9
1.5.1	Fiche stagiaire	9
1.5.2	Page de liens	9
<b>2</b>	<b>Formatage des données en XHTML</b>	<b>9</b>
2.1	Formatage du texte	9
2.1.1	Phrases	9
2.1.2	Les paragraphes	9
2.1.3	Les titres	10
2.2	Liens hypertextes	10
2.2.1	URL, URI	10
2.2.2	La balise anchor	11
2.3	Images et multimedia	11
2.4	Listes	12
2.4.1	Listes à puces et ordonnées	12
2.4.2	Listes de définitions	12
2.5	Tables	13
2.6	Cadres	14
2.7	Exercices d'applications	14
<b>3</b>	<b>CSS</b>	<b>14</b>
3.1	Présentation des feuilles de style	14
3.2	Syntaxe et propriétés des sélecteurs	15
3.2.1	Portée d'une feuille de style	15
3.2.1.1	Utilisation locale	15
3.2.1.2	Utilisation globale	16
3.2.1.3	Feuilles de style externes	16
3.2.2	Propriétés des sélecteurs	17
3.2.2.1	propriétés liées aux polices de caractères	17
3.2.2.2	Propriétés liées à la mise en page des textes	17
3.2.2.3	Propriétés liées aux arrière-plans	17
3.2.2.4	Propriétés liées aux listes	18
3.2.2.5	Propriétés liées aux bordures	18
3.2.2.6	Propriétés liées à la définition des marges	18
3.2.3	Classes de style	18
3.2.3.1	Les classes régulières	18
3.2.3.2	Les classes génériques	19
3.3	Division d'un document	19
3.3.1	La balise <span>	20
3.3.2	La balise <div>	20

3.4	Placement d'un objet . . . . .	20
3.4.1	Placer un texte . . . . .	20
3.4.2	Placer une image . . . . .	21
3.5	Exercices d'applications . . . . .	21
<b>4</b>	<b>Formulaires HTML</b>	<b>21</b>
4.1	Balises <form> . . . . .	22
4.1.1	Un premier exemple . . . . .	22
4.1.2	Balise <form> . . . . .	22
4.1.3	Balises de formulaires . . . . .	23
4.1.3.1	Boutons . . . . .	23
4.1.3.2	La balise INPUT . . . . .	23
4.2	Evénements intrinsèques . . . . .	24
<b>5</b>	<b>Evaluation</b>	<b>25</b>
<b>6</b>	<b>Références</b>	<b>25</b>
6.1	Références . . . . .	25

## Table des figures

1	Tableau . . . . .	14
---	-------------------	----

### Résumé

Ce document propose un premier aperçu du langage HTML et des feuilles de styles CSS. IL doit pouvoir être suivi en 4 demi-journées, et ne se veut bien sûr pas exhaustif. Toute information complémentaire devrait être trouvée sur le site du W3C.

## 1 HTML / XHTML - Hypertext Markup Languages

### 1.1 XHTML vs HTML vs SGML vs XML vs CSS !

HTML pour *Hyper Text Markup Language* est un langage de description de documents. C'est une application du langage de balisage SGML pour *Standard Generalized Markup Language*. HTML en est à sa version 4.01, la dernière.

SGML est un méta langage de balisage, utilisé pour concevoir d'autres langages (HTML, RTF, etc..). Unanimement reconnu comme étant trop complexe à mettre en oeuvre, SGML est aujourd'hui peu à peu abandonné pour XML.

XML pour *Extensible Markup Language* est une norme servant à décrire des documents structurés, créé donc à partir de SGML. Il s'agit en fait d'une version simplifiée de SGML (« profil » figé de la norme, DTD optionnelle, syntaxe particulière pour les éléments vides) et plus adaptée au Web (support natif des différents codages internationaux). La version 2 de XHTML est en cours de finalisation.

XHTML est donc un dérivé de HTML, se conformant non plus aux exigences de SGML, mais à celle de XML.

Enfin, si HTML se veut un langage de description de documents (une tentative de spécifier le fond/la structure du document sans en fixer la forme/l'affichage), CSS pour *Cascading Style Sheets* est un langage cette fois-ci destiné à la mise en forme du document : la façon de le présenter.

#### 1.1.1 Historique

Pour bien comprendre l'évolution de ces langages, un petit retour historique (Source [Wikipedia](#)) :

- 1989 - 1992 : l'HTML a été inventé pour le World Wide Web, afin de pouvoir écrire des documents hypertextes liant les différentes ressources d'Internet. En août 1991, lorsque Tim Berners-Lee annonce publiquement le Web sur Usenet, il ne cite que le langage SGML, mais donne l'URL d'un document ayant l'extension de fichier html. Les premiers éléments du langage HTML sont le titre du document, les hyperliens, la structuration du texte en titres, sous-titres, listes ou texte brut, et un mécanisme rudimentaire de recherche par index. La description d'HTML est alors assez informelle et principalement définie par le support des divers navigateurs Web contemporains.

- 1993 : L'état de HTML correspond alors à ce que l'on pourrait appeler HTML 1.0. Il n'existe cependant aucune spécification portant ce nom, notamment parce que le langage était alors en pleine évolution. Un effort de normalisation était cependant en cours. À partir de fin 1993, le terme HTML+ sera utilisé pour désigner la version future de HTML. Avec le navigateur NCSA Mosaic, HTML connaît deux inventions majeures : l'invention de l'élément IMG permet d'intégrer des images (GIF ou XBM) aux pages Web (Mosaic 0.10); les formulaires rendent le World Wide Web interactif (Mosaic 2.0pre5) et l'ouvre aux commandes par Internet.
- 1994 : Avec l'apparition de Netscape Navigator 0.9 le 13 octobre, le support de nombreux éléments de présentation est ajouté : styles de texte, clignotement, centrage... Le développement de HTML prend alors deux voies divergentes. D'une part, les développeurs de navigateurs s'attachent à maximiser l'impact visuel des pages Web. D'autre part, les concepteurs du Web proposent d'étendre les capacités de description sémantiques (logos, notes de bas de page...) et les domaines d'applications (formules mathématiques, tables) de HTML. En ceci, ils suivent les principes de SGML consistant à laisser la présentation à un langage de style. En l'occurrence, les feuilles de style en cascade (CSS) sont prévues pour HTML. Seul le support des tables est rapidement intégré aux navigateurs, notamment parce qu'il permet une très nette élaboration de la présentation. Le manque de structure du HTML alors mis en oeuvre par Netscape Navigator, puis Microsoft Internet Explorer, est parfois dénoncé comme étant de la « soupe de balises » (tag soup en anglais).
- 1995 - 1996 : En mars 1995, le W3C nouvellement fondé propose le résultat de ses recherches sur HTML+ : le brouillon HTML 3.0. Il comprend notamment le support des tables, des figures et des expressions mathématiques. Ce brouillon expire le 28 septembre 1995 sans donner de suites directes. Fin 1995, la RFC 1866 décrivant HTML 2.0 est finalisée. Ce document décrit HTML tel qu'il existait avant juin 1994, donc sans les nombreuses additions de Netscape Navigator.
- 1997 : Le 14 janvier, le W3C publie la spécification HTML 3.2. Elle décrit la pratique courante observée début 1996, donc avec une partie des additions de Netscape Navigator et Internet Explorer. Ses plus importantes nouveautés sont la standardisation des tables et de nombreux éléments de présentation. Le 18 décembre, le W3C publie la spécification HTML 4.0 qui standardise notamment le support des styles, les cadres (frames) et les objets (généralisation des images et des applets).
- 1998 - 1999 : La dernière spécification de HTML est la 4.01 datant du 24 décembre 1999. Elle n'apporte que des corrections mineures à la version 4.0.
- 2000 à nos jours : Le développement de HTML en tant qu'application de SGML est abandonné au profit de XHTML, application de XML. La première étape est la spécification XHTML 1.0, publiée le 26 janvier 2000. Il s'agit d'une reformulation de HTML 4.01 basée sur XML au lieu de SGML. La seconde étape est la spécification XHTML 1.1, publiée le 31 mai 2001. Il s'agit d'un classement des fonctionnalités de XHTML 1.0 en modules.

## 1.2 Le W3C

### 1.2.1 Historique

Le W3C, le *World Wide Web Consortium* est fondé par Tim Berner-Lee. Depuis sa création, le W3C émet des directives et des standards web ; ces standards, appelés recommandations, sont actuellement au nombre de quelques dizaines (plus de 90).

Le W3C est un consortium international dont la mission est de : "lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web". Il est administré conjointement par le MIT (USA), l'ERICIM (France) et l'université Keyo (Japon). Il est financé par des dons provenant de particuliers, d'entreprises et de fondations.

Le W3C a contribué à fournir au Web des standards permettant ainsi à n'importe quel navigateur sur n'importe quel OS de pouvoir lire les pages Internet. Le W3C insiste également sur l'intérêt d'offrir un Internet accessible et multi langages.

### 1.2.2 Des standards ?

Le W3C a établi depuis quelques années déjà des normes : HTML, CSS, DOM, XML etc... Malheureusement, de nombreux développeurs (designers) web ne connaissent pas bien ces langages et ont encore en tête les balises propriétaires des navigateurs les plus célèbres d'avant la standardisation : Internet Explorer 4, Netscape 4.x. Et les logiciels WYSIWYG de pages HTML produisent souvent du code non standard.

Ainsi, on rencontre encore des sites "spécifiques IE" et une partie de plus en plus importante d'internautes, de la bonne accessibilité à leur site. En effet, de l'eau a coulé sous les ponts et on assiste depuis 3-4 ans à la sortie régulières de nouveaux navigateurs, sur tout les systèmes (Windows, Linux, Mac, Unix, PalmOS, PocketPC...), parfois spécifiques à certaines utilisations (pour les handicapés physiques ou visuels comme les navigateurs "brailles") et qui plus est, conforme pour la plupart aux spécifications des Standards.

Il y a ainsi de nombreuses raisons de respecter les standards du W3C :

- une interopérabilité et une portabilité certifiée
- une accessibilité universelle aux contenus
- la production de contenu Web à moindre coût
- tirer profit de la technologie XML
- un contrôle qualité optimisé du travail fourni par les prestataires
- une réduction considérable du volume des documents
- un référencement plus efficace dans les moteurs de recherche
- la pérennité des documents
- une rétro compatibilité avec les anciens navigateurs
- pour l'avenir du Web

### 1.2.3 Validation du code

Si la plupart des sites se préoccupent peu du respect des recommandations du W3C (exemple : [www.clicanoo.com](http://www.clicanoo.com), [www.orange.re](http://www.orange.re)), il est bien sûr souhaitable de produire du code valide. La difficulté réside dans l'utilisation de plus en plus importante d'objets de toutes provenance au coeur d'une page, et surtout dans les appels à des scripts de programmation (Php, Perl, Python, Ruby, ..) qui peuvent parfois rendre la validation complexe.

Le W3C propose un moteur de validation de page HTML en ligne, à l'adresse <http://validator.w3c.org>. Je vous encourage vivement à produire du code propre et valide ! Le site [www.proze.net](http://www.proze.net) propose également un moteur de validation capable de valider l'ensemble d'un site.

## 1.3 La structure du langage

### 1.3.1 Langage de balisage

Les langages de balisage représentent une classe de langages spécialisés dans l'enrichissement d'information textuelle. Ils opèrent grâce aux balises, unités sémantiques délimitant chacune un ensemble à l'intérieur d'un fichier texte, souvent en unicode.

L'inclusion de balises permet de transférer à la fois la structure du document et son contenu. Cette structure est compréhensible par un programme informatique, ce qui autorise un affichage personnalisé selon des règles pré-établies ; la typographie (en premier lieu la fonte) et d'autres éléments de présentation peuvent changer. On peut de plus inclure des éléments non-textuels.

---

#### Exemple 1.1 Une balise html

```
<p> Voici donc un paragraphe délimité par ses 2 balises - ouvrante et ↔  
fermante </p>
```

Il existe également des balises qui n'encadrent pas de texte :

---

#### Exemple 1.2 Une balise html

```
<hr> La balise "ligne horizontale"
```

La liste des balises autorisées par le langage est fonction de sa DTD - *Document Type Definition*. Ce jeu de tags varie donc dans un même langage, d'une version à une autre.

Si en HTML, la casse des mots clefs n'a pas d'importance, en XHTML les noms et les valeurs des balises doivent être écrites en minuscule. Et toutes les balises doivent être fermées en XHTML, même si elles ne le sont pas en HTML.

---

**Exemple 1.3** Une balise xhtml

---

```
<hr></hr> La balise "ligne horizontale" en XHTML  
On peut choisir de l'écrire <hr />
```

---

### 1.3.2 Attributs

Les balises HTML peuvent être accompagnés d'attributs qui prennent différentes valeurs selon les besoins. Ainsi, on pouvait (on peut toujours encore !) spécifier certaines propriétés d'affichage directement en HTML, alors que c'est normalement le rôle des feuilles de styles CSS.

Néanmoins, certains attributs sont utiles en HTML, et permettent de donner certaines spécificités. Un exemple peut-être celui d'une liste à puce que l'on souhaite faire commencer au numéro 4 :

---

**Exemple 1.4** Attributs d'une liste ordonnée

---

```
<ol>  
<li value="4"> cet item de liste a le numéro 4.</li>  
<li> cet item de liste a le numéro 5.</li>  
<li> cet item de liste a le numéro 6.</li>  
</ol>
```

---

La liste des attributs possibles pour une balise HTML est définie dans la recommandation du [W3C](#). Certains attributs sont notés obsolètes, et ne doivent plus être utilisés (mais sont tolérés dans le cadre d'une DTD transitoire

### 1.3.3 Identifiant / Classe

Deux attributs particuliers sont applicables à toutes les balises HTML : **id** et **class**. **id** assigne un identifiant unique à un élément (qui peut être vérifié par un analyseur SGML). L'attribut **id** a plusieurs rôles dans HTML :

- comme sélecteur dans une feuille de style ;
- comme ancre cible de liens hypertextes ;
- comme moyen d'appeler un élément particulier à partir d'un script ;
- comme nom d'un élément OBJECT déclaré ;
- pour un traitement universel par les agents utilisateurs (par exemple, pour identifier les champs lors de l'extraction des données des pages HTML pour peupler une base de données, pour traduire des documents HTML dans d'autres formats, etc.).

L'attribut **class**, au contraire, assigne un ou plusieurs noms de classe à un élément ; on peut dire de l'élément qu'il appartient à ces classes. Un nom de classe peut être partagé par plusieurs instances d'éléments. L'attribut **class** a plusieurs rôles dans HTML :

- comme sélecteur dans une feuille de style (quand l'auteur souhaite assigner une information de style à un ensemble d'éléments) ;
- pour un traitement universel par les agents utilisateurs.

### 1.3.4 Emboitements

Bien que HTML soit un langage de description de documents plus qu'un langage de programmation, on retrouve certaines règles propre à la programmation dans l'écriture de code : la lisibilité, l'indentation, les commentaires et l'emboîtement des blocs.

L'emboîtement des blocs consiste à fermer les blocs fils avant les blocs parents. L'indentation du code peut aider à retrouver facilement la hiérarchie des blocs, et les commentaires de retrouver la construction de la page.

**Exemple 1.5** Erreurs de blocs

```
<p>Un exemple de paragraphe avec une liste ordonnée
<ol>
<li value="4"> cet item de liste a le numéro 4.</li>
<li> cet item de liste a le numéro 5.</li>
<li> cet item de liste a le numéro 6.</li>
</p></ol>
```

**1.3.5** La documentation du W3C

Vous l'aurez compris, les recommandations de W3C vont rapidement devenir votre bible du développeur Web ! Les recommandations sont disponibles en plusieurs langues, et voici les liens pour :

- [HTML 4.0](#)
- [XHTML 1.0](#)
- [CSS 2.0](#)

La documentation propose l'ensemble des balises et des attributs disponibles dans le langage. C'est donc LA source à consulter en cas de doute sur la syntaxe d'une balise. Voici un exemple de la documentation :

**Exemple 1.6** Documentation de la balise IMG

```
<!-- Pour éviter les problèmes avec les agents utilisateurs
en mode texte seul ainsi que pour rendre le contenu d'une image
compréhensible et navigable pour les utilisateurs d'agents
utilisateurs non-visuels, il est nécessaire de fournir une
description avec ALT, et pour éviter les images cliquables
côté serveur -->
<!ELEMENT IMG - O EMPTY          -- image incorporée -->
<!ATTLIST IMG
  %attrs;                          -- %coreattrs, %i18n, %events --
  src          %URI;               #REQUIRED -- URI de l'image à incorporer --
  alt          %Text;              #REQUIRED -- description brève --
  longdesc    %URI;               #IMPLIED  -- lien vers une description ←
    longue
                                     (en complément de l'attribut ←
                                     alt) --
  name        CDATA                #IMPLIED  -- nom de l'image destiné aux ←
    scripts --
  height     %Length;              #IMPLIED  -- surclasse la hauteur --
  width      %Length;              #IMPLIED  -- surclasse la largeur --
  usemap     %URI;                 #IMPLIED  -- utiliser une image cliquable ←
    côté client --
  ismap      (ismap)               #IMPLIED  -- utiliser une image cliquable ←
    côté serveur --
>
```

**1.4** Squelette d'un fichier XHTML

Nous avons donc vu que HTML est un langage de description. Mais il en existe plusieurs versions sur Internet, que chaque navigateur doit pouvoir lire. De plus, il est important pour les différents moteurs de recherches d'avoir des informations sur les documents analysés par les robots. Ainsi, une page web est composée de plusieurs parties :

- Définition de type de document : quels sont la version du langage, et le codage des caractères
- Entête : des méta-informations sur la page (auteur, date, mots clefs, ...)
- Corps : ce qui s'affichera dans le navigateur

Globalement, un fichier HTML est donc de ce type :

---

**Exemple 1.7 Squelette HTML**

---

```
<?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">
<head>
</head>
<body>
<!-- le contenu de ma page -->
</body>
</html>
```

---

**1.4.1 La DTD**

Il est nécessaire d'indiquer en tout début de page quel est le langage que l'on va utiliser. Pour cela nous disposons pour chaque langage (HTML, XHTML) 3 types de DTD :

- *transitoire* : cette DTD permet certains abus d'écriture de code. On peut utiliser des balises obsolètes, utiliser des raccourcis (certaines balises n'ont par exemple pas besoin d'être fermées), utiliser des attributs de présentation.
- *stricte* : cette DTD demande un plus grand respect des recommandations.
- *cadre* : DTD utilisée pour les cadres (fractionnement de fenêtre)

---

**Exemple 1.8 Exemples de déclaration de DTD**

---

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

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

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

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

---

**1.4.2 Les balises d'entêtes**

L'élément **head** contient des informations sur le document courant, tels que son titre, des mots-clés que les moteurs de recherche peuvent exploiter et d'autres données qui ne sont pas considérées comme faisant partie du contenu du document. Les agents utilisateurs navigateurs ne restituent généralement pas les éléments qui apparaissent comme contenu de l'élément **head**. Il peuvent cependant rendre les informations contenues par l'élément **head** disponibles aux utilisateurs au moyen d'autres mécanismes.

La balise **title** est une balise obligatoire, elle contient le titre de la page. Il apparaîtra par exemple dans la barre de titre de la fenêtre du navigateur, ou dans les moteurs de recherche.

Les autres balises d'entête sont pour la plupart des balises **meta**, qui contiennent des informations complémentaires sur la page et son contenu. Ces informations sont spécifiées en renseignant des valeurs d'attributs. En particulier on retiendra les suivants :

- `http-equiv="Content-Type" content="text/html; charset=iso-8859-1"` : indique le jeu de caractère utilisé
- `name="keywords" content=" .... "` : donne une liste de mots clefs

---

**1.4.3 Convention de nommage des fichiers**

Une erreur fréquemment commise concerne le nommage des fichiers d'un site Internet. En effet, les personnes travaillant sous Windows s'autorisent tout genre de fantaisies pour nommer les fichiers. Un

exemple courant est l'utilisation d'espace et de majuscules dans les noms de fichiers. Ceci n'a pas grande importance sous Windows, mais une fois le site mis en ligne, peut apporter quelques surprises ...

Il faut donc prendre dès le début de bonnes habitudes pour nommer les fichiers :

- la page par défaut d'un répertoire est nommée `index.html`
- les noms de fichiers (`html`, `images`, `php`, ...) ne comportent pas d'espaces, pas de caractères spéciaux, pas de majuscules
- les noms des répertoires respectent les mêmes règles

## 1.5 Exercices d'applications

Tout au long de ce cours, nous allons utiliser comme exemple une application web présentant les stagiaires de la formation. Dans cette première partie, vous allez vous attacher à la création de deux pages distinctes, *from scratch* c'est-à-dire en utilisant tout simplement un éditeur de texte.

### 1.5.1 Fiche stagiaire

La première page est une page de présentation d'un stagiaire, elle contiendra plus tard un certain nombre d'informations sur chacun d'entre vous. Vous devez la créer en utilisant la *DTD XHTML STRICT*. Une fois la page validée, faites en sorte de suivre les indications du **W3C** pour faire apparaître le logo XHTML dans la page.

Dans un premier temps, elle ne devra contenir qu'une seule phrase de présentation. Vous vous attacherez à soigner les balises d'entête. Bien sûr, la page devra être valide sur le **W3C**.

### 1.5.2 Page de liens

La seconde page est une page dans laquelle vous collerez tous les liens utiles que nous rencontrerons pendant la formation. Vous devez la créer en utilisant la *DTD XHTML TRANSITIONAL*.

Dans un premier temps, elle ne devra contenir qu'une seule phrase de présentation. Vous vous attacherez à soigner les balises d'entête. Bien sûr, la page devra être valide sur le **W3C**. Une fois la page validée, faites en sorte de suivre les indications du **W3C** pour faire apparaître le logo XHTML dans la page.

## 2 Formatage des données en XHTML

### 2.1 Formatage du texte

Dans cette section, nous allons examiner quelques balises permettant de structurer le texte d'un document. Nous ne nous intéressons pas à la mise en forme.

#### 2.1.1 Phrases

Un certain nombre de balises permettent d'enrichir les informations contenues dans une phrase. Même si elles ne sont pas souvent employées, il peut être intéressant de les connaître :

- **EM** : Indique une mise en exergue.
- **STRONG** : Indique une mise en exergue plus forte.
- **CITE** : Contient un extrait ou une référence vers une autre source
- **DFN** : Indique qu'il s'agit de l'instance définissante du terme englobé.
- **CODE** : Désigne un fragment de code informatique.
- **SAMP**: Désigne un exemple des sorties d'un programme, d'un script, etc.
- **KBD**: Indique un texte que doit saisir l'utilisateur.
- **VAR**: Indique l'instance d'une variable ou le paramètre d'un programme.
- **ABBR**: Indique une forme abrégée (par exemple, « `WWW` », « `HTTP` », « `i.e.` », etc.).
- **ACRONYM**: Indique un acronyme (par exemple, « `radar` », « `LAN` », etc.).

#### 2.1.2 Les paragraphes

Les phrases sont contenues généralement dans des paragraphes. Nous avons vu dans le premier chapitre un exemple de l'utilisation de la balise `p`. Il est possible de sauter une ligne dans un même

paragraphe en utilisant la balise **br**. Une fois de plus, il est important de ne pas se soucier de la présentation du document lors du découpage du contenu en paragraphe : c'est le rôle des feuilles de style CSS, et non d'HTML.

### 2.1.3 Les titres

Enfin, tout document devrait posséder une hiérarchisation de titres : titre de chapitre, titre de paragraphe, titre de sous-paragraphe, etc... Cette hiérarchie est représentée en HTML par les balises H1, H2, H3, H4, H5, H6, du plus important au moins important. Il est important de souligner que c'est l'agent utilisateur (le navigateur internet par exemple), qui s'occupe de la numérotation des titres ! Cette numérotation est généralement du style 1, 1.1, 1.1.1, 1.1.2, 1.2 ...

Voici un exemple de document utilisant quelques unes des balises précédentes :

---

#### Exemple 2.1 Squelette HTML

---

```
<h1>Titre du chapitre</h1>
<p>Un exemple de hiérarchie de paragraphes</p>
<h2>Titre de niveau 2</h2>
<p>Voici <abbr>qqs</abbr> exemples de balises <acronym>HTML</acronym>, ↔
    enrichissant le contenu d'un
paragraphe. Il n'agit ici de ne préciser que le <strong>fond</strong> du ↔
    document, et non sa forme. </p>
```

---

## 2.2 Liens hypertextes

### 2.2.1 URL, URI

Une URI - *Universal Resource Identifier* - est un pointeur qui identifie de façon certaine une ressource disponible quelque part. Une URI peut être :

- URN : Qui permet d'appeler une ressource par son nom
- URC : Qui décrit les caractéristiques d'une ressource
- URL : Qui donne sa localisation (ou plutôt le mode d'emploi pour l'atteindre).

L'URL - *Uniform Resource Location* - va donc permettre à un lien hypertexte d'atteindre une ressource. Une URL est composé de la manière suivante :

```
{scheme}://<nom d'utilisateur>:<mot de passe>@<hôte ou adresse IP>:<port de ↔
    connexion>/<extension spécifique au protocole>
```

on dispose ainsi de plusieurs formes de ressources hypertextes :

---

#### Exemple 2.2 URL's

---

```
http://www.kurzweg.info/InitLL_2007/asrest/index.htm
ftp://ftp.freebsd.org
../../asrest/index.htm
mailto:ivan.kurzweg@wanadoo.fr
```

---

Comme nous venons de le voir dans l'exemple précédent, il est possible d'écrire le lien vers la page `index.html` de deux manières :

- en utilisant la référence (*path, chemin*) *absolu*
- en utilisant la référence *relative*

Dans le cas de la référence relative, on cherche un objet dans le même espace de noms, en lui donnant le chemin complet depuis l'appelant. La remontée dans l'arborescence de fichier est possible en utilisant le raccourci `..` qui signifie *répertoire parent*. Ainsi, l'exemple précédent `../../asrest/index.html` signifie que l'on descend de deux répertoires depuis le répertoire courant (celui où la page en cours est stocké, qui apparaît dans la barre d'adresse), puis on remonte dans le répertoire `asrest` pour atteindre la page `index.html`.

**NOTE**

Attention, il est important de bien comprendre la nécessité de donner des chemins relatifs quand les objets appartiennent au même espace de noms. En effet, lors, par exemple, de la mise en ligne d'un site Internet, toutes les références changent : un chemin noté `/home/ikare/public_html/InitLL/images/logo.jpg` sur un poste en local n'aura plus aucune valeur chez un hébergeur Internet ! Par contre, `./images/logo.html` sera toujours valide.

**2.2.2 La balise anchor**

La balise **a** a deux fonctions :

- faire une ancre : spécifier dans la page grâce à l'attribut **name** un point d'accès hypertexte
- faire un lien : en spécifiant la valeur de l'attribut **href**

Il est à noter que la documentation HTML spécifie que la valeur de l'attribut **alt** (texte secondaire) doit être renseignée. Elle sert dans le cas de navigateur n'affichant pas d'images.

**Exemple 2.3 Liens**

```
<a name="acces">Ici, un point dans la page qui sera accessible grâce à un lien vers ↔
  la page suivi de #acces</a>

<a href="http://www.kurzweg.info/InitLL_2007/asrest/index.htm" alt="Vers une page ↔
  de cours">
Cliquer ici pour acceder au cours</a>

<a href="#acces" alt="Vers une ancre dans la page">Et ici pour remonter !</a>
```

**2.3 Images et multimedia**

Heureusement pour le Web, les pages HTML ne sont pas composées que de textes ! Malheureusement pour certains, il est possible d'y inclure des objets qui ne sont pas tous OpenSource, et loin d'être totalement portables (Flash, ActiveX, ...). Nous nous contenterons dans cette partie d'abord l'inclusion d'images dans une page HTML.

Les formats les plus couramment utilisés dans les images sont GIF, JPG, PNG. La partie de la formation consacrée aux outils bureautique vous donnera plus d'informations sur ces différents formats. De nombreux sites sur Internet propose des comparatif, en voici : <http://csrs.qc.ca/Goeland/proj/-envolee/sacdestic/html/optimiser.html>.

**NOTE**

Il est à noter que le format GIF est un format propriétaire, déposé par Unisys, PNG est par contre libre.

La balise **img** permet d'inclure une image dans une page HTML. La recommandation du W3C nous donne la définition suivante :

```
<!-- Pour éviter les problèmes avec les agents utilisateurs
  en mode texte seul ainsi que pour rendre le contenu d'une image
  compréhensible et navigable pour les utilisateurs d'agents
  utilisateurs non-visuels, il est nécessaire de fournir une
  description avec ALT, et pour éviter les images cliquables
```

```

    côté serveur -->
<!ELEMENT IMG - O EMPTY          -- image incorporée -->
<!ATTLIST IMG
  %attrs;                          -- %coreattrs, %i18n, %events --
  src          %URI;                #REQUIRED -- URI de l'image à incorporer --
  alt          %Text;                #REQUIRED -- description brève --
  longdesc     %URI;                #IMPLIED -- lien vers une description longue
                                     (en complément de l'attribut alt) --
  name         CDATA                 #IMPLIED -- nom de l'image destiné aux scripts --
  height       %Length;              #IMPLIED -- surclasse la hauteur --
  width        %Length;              #IMPLIED -- surclasse la largeur --
  usemap       %URI;                #IMPLIED -- utiliser une image cliquable côté client ←
  --
  ismap        (ismap)               #IMPLIED -- utiliser une image cliquable côté serveur ←
  --
>

```

Le renseignement de l'attribut **ALT** est une obligation. Indiquer les dimensions d'une image, même si elles sont celles d'origine peut permettre un affichage plus rapide de la page

## 2.4 Listes

### 2.4.1 Listes à puces et ordonnées

Il existe deux types de listes en HTML : listes à puces et liste ordonnées. Dans les deux cas, le principe de rédaction est le même :

- la balise globale **ul** ou **ol** (à puces ou ordonnée) englobe la liste
- chaque élément de la liste est encadré par la balise **li**

Une fois de plus, il faut éviter toute inclusion de code lié à la présentation dans HTML.

---

#### Exemple 2.4 Liste à puces

---

```

<li class="tocline1"><a href="conform.html" class="tocxref">3 La conformité : ←
  obligations et recommandations</a>
<ul class="toc">
  <li class="tocline2"><a href="conform.html#q1" class="tocxref">3.1 Définitions ←
    </a>
  </li><li class="tocline2"><a href="conform.html#conformance" class="tocxref" ←
    ">3.2 La conformité</a>
  </li><li class="tocline2"><a href="conform.html#q3" class="tocxref">3.3 Les ←
    conditions d'erreur</a>
  </li><li class="tocline2"><a href="conform.html#text-css" class="tocxref">
3.4 <span class="index-def" title="text/css">Le type de contenu "text/css"</span></ ←
  a>

</li></ul>
</li>

```

### 2.4.2 Listes de définitions

Le WEB est à l'origine un outil conçu pour les scientifiques. Ainsi on dispose d'une balise spéciale permettant de faire une liste de définitions :

**Exemple 2.5** Liste de définitions

```

<DL>
  <DT>Dweeb
  <DD>Une jeune personne motivée qui peut évoluer
    en <EM>nerd</EM> ou en <EM>geek</EM>

  <DT>Hacker
  <DD>Un programmeur futé

  <DT>Nerd
  <DD>Une personne techniquement brillante mais socialement inepte
</DL>

```

## 2.5 Tables

L'utilisation des tables en HTML est fréquente, même si certains designers les utilisent à mauvais escient (pour cadrer une mise en page par exemple). Le principe d'écriture du code est le suivant :

- encadrement du tableau avec la balise **table**
- déclaration des lignes avec la balise **tr**
- déclaration des cellules (colonnes) dans chaque ligne **td**

De ces principes découlent certaines contraintes : chaque ligne doit contenir le même nombre de colonnes. Dans des cas spéciaux, il est possible de fusionner plusieurs :

- ligne en utilisant l'attribut **rowspan**
- cellules en utilisant l'attribut **colspan**

Voici un exemple de déclaration de tableau (source W3C) :

**Exemple 2.6** Un tableau avec des cellules fusionnées

```

<TABLE border="1"
  summary="Cette table donne quelques statistiques sur les
    drosophiles : hauteur et poids moyens, et le
    pourcentage de celles ayant des yeux rouges
    (pour les mâcles et les femelles).">
<CAPTION><EM>Une table test avec des cellules fusionnées</EM></CAPTION>
<TR><TH rowspan="2"><TH colspan="2">Moyenne
  <TH rowspan="2">Yeux<BR>rouges
<TR><TH>hauteur<TH>poids
<TR><TH>Mâcles<TD>1.9<TD>0.003<TD>40%
<TR><TH>Femelles<TD>1.7<TD>0.002<TD>43%
</TABLE>

```

Ce qui nous donnerait le résultat suivant :

FIG. 1 Tableau

*Une table test avec des cellules fusionnées*

	Moyenne		Yeux rouges
	hauteur	poids	
<b>Mâles</b>	1.9	0.003	40%
<b>Femelles</b>	1.7	0.002	43%

## 2.6 Cadres

Les jeux de cadres - *frameset* - permettent de diviser la page du navigateur en plusieurs fenêtres indépendantes. Ils sont de moins en utilisés, car posent quelques problèmes dans le référencement opéré par les robots, et par l'affichage dans des navigateurs non-visuels. Par conséquent, je vous renvoie à la documentation W3C pour plus d'informations.

## 2.7 Exercices d'applications

Nous allons maintenant compléter les pages créées dans le premier chapitre.

- Page "stagiaire" : vous allez devoir entrer toutes les informations nécessaires dans cette page. Vous vous attacherez à ne mettre aucune information de présentation, mais à structurer correctement le contenu.
- Page liens : sans utiliser de tableau, proposez une classification des liens en 4 parties : HTML, CSS, PHP, MySQL. Entrez quelques liens dans la page

Créez ensuite une page de garde, nommée index.html, qui possède une image et deux liens vers les pages précédentes.

### NOTE



Bien sûr, vos pages doivent être toujours valides W3C.

## 3 CSS

### 3.1 Présentation des feuilles de style

La séparation du fond de la forme est une recommandation récurrente au W3C. En effet, le concepteur est responsable du fond du document : il organise son discours suivant une hiérarchie simple : un document est composé d'un ensemble de paragraphes qui possèdent éventuellement des titres. Les paragraphes peuvent prendre différentes formes : un texte simple, une citation, une liste à puces, une liste ordonnée, un tableau, ... Les titres sont définis en appliquant une structuration à six niveaux (accessibles par les balises `<h1>` à `<h6>`).

La responsabilité du navigateur est alors de réaliser une mise en page adaptée au document défini par le concepteur. C'est le navigateur qui est chargé de faire ressortir les titres des paragraphes (en changeant la police par exemple), de définir la largeur des colonnes des tableaux afin d'obtenir un affichage du document agréable à lire. Pour s'acquitter de cette tâche, les navigateurs appliquent un ensemble de règles par défaut. Par exemple, le texte compris entre les balises `<h1>` et `</h1>` sera affiché en gras avec une taille de police supérieure à la taille d'un texte classique. Les feuilles de style (*Cascading Style Sheets* (CSS)) renforcent les concepteurs dans cette philosophie. Elles ont été introduites en 1996 par le W3C. Aujourd'hui, tous les navigateurs internet supportent la définition de feuilles de style. Elles permettent de définir l'ensemble des règles par défaut applicables par un navigateur à une balise html quelconque.

Ainsi, les feuilles de style permettent de préciser, par exemple, la couleur de fond d'un paragraphe introduit par les balises `<p>` et `</p>`. Il existe deux recommandations du W3C concernant les feuilles de style : c'est pourquoi nous parlons des recommandations CSS1 et CSS2. Ces recommandations sont disponibles en français à l'adresse suivante : <http://www.yoyodesign.org/doc/w3c/css.html.fr>

Dans ce chapitre, nous verrons dans un premier temps *la syntaxe et les propriétés des sélecteurs* : comment nous pouvons modifier l'apparence d'une balise en précisant la valeur associée à chaque propriété liée à cette balise. Ensuite, nous examinerons *la division d'un document*, ou comment modifier l'apparence d'une partie d'un document en le séparant explicitement (à l'aide des balises `<div>` et `<span>`) du reste du document. Enfin, nous verrons comment la recommandation CSS2 permet de *fixer précisément la position d'une image* ou de tout objet au sein d'une page HTML.

## 3.2 Syntaxe et propriétés des sélecteurs

La définition des éléments d'une feuille de style respecte une syntaxe relativement simple. Dans un premier temps, nous allons définir la syntaxe utilisée dans une feuille de style externe (c'est à dire faisant l'objet d'un fichier distinct du document html). On appelle *sélecteur* toute balise HTML (`<h1>`, `<p>`, ...) privée des caractères `<` et `>`. Chaque sélecteur possède un ensemble de propriétés qui peuvent être définies par une (ou plusieurs) feuilles de style. Par exemple, la police de caractères utilisée, sa taille, sa couleur ou bien encore la couleur du fond,... Les recommandations CSS1 et CSS2 définissent 53 propriétés. Pour définir un ensemble de propriétés associées à une balise HTML, la syntaxe est la suivante : **sélecteur {propriété1 :valeur1;propriété2 :valeur2;...;propriétén :valeurn}** Par exemple, si nous souhaitons fixer la couleur d'arrière-plan d'un document, il suffit de définir : **body {background-color :white}** Enfin, il est possible de définir une (ou plusieurs) propriété(s) pour un groupe de sélecteurs. La syntaxe à utiliser dans ce cas est assez simple, il nous suffit de séparer chaque sélecteur par des virgules avant de définir les valeurs associées aux propriétés. Voici par exemple comment justifier à droite les titres de niveau 1 et 2 : **h1, h2 {text-align : right}**

### 3.2.1 Portée d'une feuille de style

Les recommandations du W3C concernant les feuilles de style autorisent non seulement la définition de feuilles de style externes, mais aussi la définition de propriétés directement à l'intérieur d'une page HTML. Dans ce cas, la syntaxe utilisée est légèrement différente que celle déjà présentée. En fait, nous pouvons modifier localement les propriétés liées à une balise particulière d'un document HTML ou bien définir, toujours à l'intérieur de la page, l'ensemble des propriétés à appliquer à toutes les balises du document.

**3.2.1.1 Utilisation locale** Pour définir au niveau d'une balise précise les valeurs de ses propriétés, il est nécessaire de définir l'attribut `style` de cette balise. L'utilisation de l'attribut `<style>` à l'intérieur du code HTML (entre `<body>` et `</body>`) permet de modifier l'apparence d'un élément :

```
<sélecteur style="propriété :valeur">
```

Voici un exemple où nous définissons la couleur de fond de la page et du premier paragraphe (respectivement à blanc et à bleu), nous modifions la police de caractères utilisée pour le titre du premier paragraphe (police helvetica) et enfin, nous modifions également la couleur du texte du second paragraphe (en vert foncé) :

**Exemple 3.1** Feuille de style locale

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title> Utilisation locale des feuilles de style </title>
</head>
<body style="background-color : white">
<h1 style=" font-family : helvetica ; border-style : inset ">
CE TITRE UTILISE LA POLICE HELVETICA </h1>
<p style="background-color : blue">
La couleur du texte de ce paragraphe est le bleu. </p>
<h1> CE TITRE UTILISE TOUTES LES VALEURS PAR DEFAULT DES PROPRIETES </h1>
<p style="color : darkgreen"> Ce deuxième paragraphe utilise la police par défaut ←
'
la couleur choisie pour le texte est le vert foncé. </p>
</body>
</html>
```

**3.2.1.2 Utilisation globale** L'utilisation locale des feuilles de style se montre très rapidement limitée. En effet dans la majorité des cas, la définition des attributs est suffisante pour obtenir les mêmes effets. C'est pourquoi, il est recommandé de définir des feuilles de style globales. En utilisant des feuilles de style globales, nous obtenons la garantie d'une mise en page uniforme de toute la page ; garantie que nous ne pouvons obtenir en utilisant des feuilles locales ou bien la définition d'attributs. La définition d'une feuille de style globale a lieu dans la partie entête du document (entre les balises **<head>** et **</head>**). Une feuille de style globale est introduite par la balise **<style>** et terminée par la balise **</style>**. Il est conseillé de placer entre commentaires les propriétés définies : ainsi les navigateurs ne supportant pas les feuilles de style les ignoreront. Dans l'exemple suivant, nous définissons la couleur par défaut du fond de la page, des paragraphes ainsi que la police des titres de niveau 1.

**Exemple 3.2** Utilisation globale des feuilles de styles

```
<!DOCTYPEHTMLPUBLIC"-//W3C//DTD HTML4.01Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type"content="text/html; charset=iso-8859-1">
<title>Utilisation globale des feuilles de style</title>
<style type="text/css">
<!--body{background-color:white}
p{background-color:lightgreen;color:darkgreen}
h1{font-family:helvetica}
-->
</style>
</head>
<body>
<h1>Propriétés de la balise &lt;body></h1>
<p>Nous redéfinissons la propriété <i>background-color</i> du sélecteur
<i>body</i> à~ blanc.</p>
<h1>Propriétés de la balise &lt;p></h1>
<p>Chaque paragraphe a pour couleur de fond le vert clair et le texte
est écrit en vert foncé.</p>
<h1>Propriétés de la balise &lt;h1></h1>
<p>La police de caractères de la balise &lt;h1>est <i>helvetica.</i></p>
</body>
</html>
```

&gt;

**3.2.1.3 Feuilles de style externes** Comme vous l'avez sans doute compris, une feuille de style regroupe un ensemble de règles qui indique au navigateur comment il doit interpréter (ou afficher) chaque

balise.

Ces règles peuvent être définies directement dans le document HTML (utilisation locale ou globale), ou bien (et c'est beaucoup plus intéressant) à l'extérieur du document. L'avantage des feuilles de style externes réside dans le fait de pouvoir appliquer un même ensemble de règles à toutes les pages d'un site internet, ce qui est impossible avec une utilisation locale ou globale. Supposons définie une feuille de style nommée `style.css`, dont voici le contenu :

```
h1,h2{text-align:center}
h3,h4{text-align:right;color:rgb(50%,50%,0)}
```

Vous remarquerez comment il est aisé de créer une nouvelle couleur à l'aide d'une feuille de style (ici, un jaune clair). Il est également possible de préciser l'intensité de chaque composante non pas en pourcentage mais par une valeur comprise entre 0 et 255. Pour lier (appliquer) cette feuille de style à un document html, il nous suffit de définir la balise `<link>` dans ce document :

```
<!DOCTYPEHTMLPUBLIC"-//W3C//DTDHTML4.01Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type"content="text/html;charset=iso-8859-1">
<title>Feuille de style externe</title>
<link rel=stylesheet type="text/css" href="v21.css" title="v21">
</head>
<body>
<h1>Ce titre est centré</h1>
<h2>Ce titre est également centré</h2>
<h3>Ce titre est aligné à~ droite et de couleur jaune</h3>
<h4>Idem</h4>
</body>
</html>
```

Voici quelques explications concernant les attributs utilisés :

- **rel** : précise que nous lions une feuille de style à ce document.
- **type** : la feuille de style est conforme au standard CSS.
- **href** : l'URL de la feuille de style.
- **title** : nous nommons la feuille de style au cas où<sup>1</sup> le navigateur devrait ultérieurement y faire référence (facultatif dans la majorité des cas).

### 3.2.2 Propriétés des sélecteurs

**3.2.2.1 propriétés liées aux polices de caractères** **font-family** : la police de caractères utilisée.

**font-size** : la taille de la police de caractères. Cette valeur peut être définie en points ou bien à l'aide d'une valeur prédéfinie : **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**, **smaller**, **larger**.

**font-style** : définit l'inclinaison de la police : **normal**, **oblique**, **italic**.

**font-weight** : épaisseur du trait : **bold**, **bolder**, **lighter**, **normal**. Il est également possible de préciser une valeur (multiple de 100) comprise entre 100 (le trait le plus fin) et 900 (le trait le plus épais). La valeur 400 correspond à normal.

**3.2.2.2 Propriétés liées à la mise en page des textes** **color** : la couleur de la police de caractères utilisée.

**text-align** : précise l'alignement du texte, quatre valeurs sont possibles : **left**, **center**, **justify**, **right**.

**text-decoration** : définit l'enrichissement dont fait l'objet le texte, cinq valeurs sont possibles : **none** (valeur par défaut) **underline**, **overline**, **line-through**, **blink**.

**text-transform** : autorise une modification automatique des textes, quatre valeurs possibles : **capitalize**, **uppercase**, **lowercase**, **none** (valeur par défaut).

**text-indent** : retrait du premier mot du paragraphe. La valeur associée à cette propriété peut être exprimée en points (pt), en centimètres (cm), en millimètres (mm) en pouces (in) ou bien en pourcentage. Par défaut, la valeur est égale à zéro.

**3.2.2.3 Propriétés liées aux arrière-plans** **background-color** : la couleur de fond de l'objet.

**background-image** : définit l'image de fond utilisée, par défaut la valeur est **none** (aucune).

**background-repeat** : précise si la répétition d'une image de fond est autorisée, les valeurs possibles sont **repeat**, **repeat-x**, **repeat-y**, **no-repeat** (valeur par défaut).

**3.2.2.4 Propriétés liées aux listes** **list-style-type** : précise le type de puces utilisées, les valeurs possibles sont : **disc** (valeur par défaut), **circle**, **square**, **decimal**, **lower-roman**, **upper-roman**, **lower-alpha**, **upper-alpha**, **none**.

**list-style-image** : permet l'utilisation d'une image comme puce, la valeur attendue est l'adresse de l'image. Par défaut, la valeur est **none**.

**list-style-position** : précise si la position de la puce par rapport au texte, deux valeurs sont autorisées **inside** et **outside**, valeur par défaut.

**3.2.2.5 Propriétés liées aux bordures** **border-style** : définit les quatre bordures d'un élément. Les valeurs possibles sont : **none**, **dashed**, **dotted**, **double**, **groove**, **inset**, **outset**, **ridge**, **solid**.

**border-color** : définit la couleur de la bordure. Une couleur peut être définie de trois façons différentes : 1. Par son nom (par exemple **blue**). 2. Par sa définition suivant le modèle RGB en hexadécimal, la notation est alors la suivante : **color : #rrvvbb**. 3. Par sa définition suivant le modèle RGB en décimal, une couleur est dans ce cas définie ainsi : **color : rgb(rrvvbb)**. Les valeurs peuvent également être des pourcentages, par exemple : **color : rgb(100%,50%,0)**.

**border-width** : définit l'épaisseur du trait de la bordure. Les valeurs autorisées sont : **thin**, **medium** (valeur par défaut), **thick** ou bien une valeur.

**3.2.2.6 Propriétés liées à la définition des marges** **margin-top** : définit la marge en haut. Par exemple, la définition suivante est autorisée : **h2 { margintop : 1em }**. Ici, la marge en haut d'un titre de niveau 2 est fixée égale à la taille de la police utilisée.

**margin-bottom** : définit la marge en bas.

**margin-right** : définit la marge à gauche.

**margin-left** : définit la marge à droite.

**padding-left** : définit l'espacement à gauche. Les propriétés **padding-right**, **padding-bottom**, **padding-top** sont également définies.

### 3.2.3 Classes de style

Quand il s'agit de rédiger un rapport, il est appréciable de pouvoir appliquer une présentation du résumé du document différente du reste du document. La définition de classes au niveau des feuilles de style autorise ce type de modification.

**3.2.3.1 Les classes régulières** Les classes régulières permettent la définition de plusieurs règles d'affichage pour un même sélecteur.

**Exemple 3.3** Classes régulières

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
<meta http-equiv = "Content-Type" content="text/html;charset=iso-8859-1">
<style type="text/css">
<!--body{background-color:white}
p{color:darkgreen}
p.resume{background-color:lightyellow;
text-align:justify;margin-left:5em;margin-right:5em}
-->
</style>
</head>
<body>
<p class= "resume">Voici le résumé du document.
Un retrait à~ gauche et à~ droite est appliqué.
La couleur de fond est un jaune clair.
Le texte est justifié.
Vous remarquerez que les propriétés éfinies pour un paragraphe <i>classique</i> ←
s'appliquent également ici.</p>
<p>Voici un paragraphe du corps du document.
Ici, seule la couleur de la police de couleur a été définie en vert foncé.</p>
</body>
</html>

```

Nous sommes libres de définir un nombre quelconque de classes pour chaque sélecteur. Ces définitions peuvent également intervenir au niveau des feuilles de style externes. L'utilisation d'une classe au niveau d'un document HTML est très simple, il suffit de préciser le nom **classe** utilisée par l'intermédiaire de l'attribut **class** qui existe pour toutes les balises HTML.

**3.2.3.2 Les classes génériques** Dans l'exemple précédent, nous avons défini une classe qui s'appliquait à la balise **<p>**. Le standard CSS autorise également des définitions de classes génériques qui pourront être appliquées à toutes balises.

**Exemple 3.4** Classe générique

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
<meta http-equiv = "Content-Type" content="text/html;charset=iso-8859-1">
<title> Classe générique </title>
<style type="text/css">
<!-- .bleu {color : blue }
-->
</style>
</head>
<bdy>
<p class= " bleu "> Ce paragraphe est en bleu. </p>
<p> Mais pas ici !!! </p>
</body>
</html>

```

**3.3** Division d'un document

Dans certains cas, il est nécessaire de pouvoir "oublier" pendant quelques mots ou lignes une feuille de style. C'est l'objectif des balises **<span>** et **<div>** qui autorisent une modification locale de la feuille de style.

### 3.3.1 La balise <span>

Supposons, que nous souhaitions appliquer une classe générique à plusieurs mots d'un paragraphe. L'utilisation de la balise <span> est alors fortement recommandée.

---

#### Exemple 3.5 Exemple d'emploi de <span>

---

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>La balise span</title>
<style type="text/css">
<!--.fondvert{background-color:lightgreen}
-->
</style>
</head>
<body>
<p>Voici un paragraphe.<span class="fondvert">Cette partie du texte est
affichée en utilisant la classe <i>fondvert</i>.</span>
Ici, nous appliquons de nouveau les propriétés d'affichage par défaut.</p>
</body>
</html>
```

---

### 3.3.2 La balise <div>

Comme la balise <span>, la balise <div> autorise une modification locale de la feuille de style. Cette modification portera non plus sur un ensemble de mots, mais sur un ou plusieurs paragraphes.

## 3.4 Placement d'un objet

Les feuilles de style apporte une solution facile d'emploi quand il s'agit de placer précisément un objet dans un document HTML. En effet, nous pouvons préciser les coordonnées relatives ou absolues de n'importe quel objet composant la page.

Les coordonnées absolues sont définies par rapport au coin supérieur gauche de la fenêtre du navigateur qui a pour coordonnées (0,0). La première coordonnée est associée à la largeur de la fenêtre, la seconde à la hauteur.

Supposons que la fenêtre de votre navigateur possède une taille de 400 x 300 pixels, alors : le sommet haut gauche a pour coordonnées (0,0); le sommet haut droit a pour coordonnées (400,0) ; le sommet bas droit a pour coordonnées (400,300); le sommet bas gauche a pour coordonnées (0,300).

Le positionnement relatif se détermine par rapport à d'autres objets de la page. Il est beaucoup plus difficile à maîtriser et est par conséquent beaucoup moins utilisé. L'utilisation des balises <span> et <div> est fortement conseillé dans ce cas. Nous allons présenter rapidement comment utiliser le positionnement absolu à l'aide de trois exemples : placement d'un texte, d'une image et superposition d'un texte sur une image.

#### 3.4.1 Placer un texte

Pour placer un texte précisément, nous pouvons introduire une division du document (en utilisant la balise <div>) et fixer les coordonnées du texte à l'aide d'une feuille de style locale. Voici un exemple où nous plaçons en (200,50) le texte *HYPertext MARKUP LANGUAGE*. Donc ce texte sera décalé de 200 pixels sur la droite et de 50 pixels vers le bas.

**Exemple 3.6** Placement du texte

```
<!DOCTYPE HTML PUBLIC "-//W3C //DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Placement de texte</title>
</head>
<body>
<span style="position:absolute;left:200px;top:50px;color:navy">
HYPERTEXT<br>
MARKUP<br>
LANGUAGE
</span>
</body>
</html>
```

**3.4.2 Placer une image**

Le placement d'une image n'est pas différent du placement d'un texte. Nous pouvons tout simplement remplacer le texte par l'affichage de l'image avec la balise **<img>**. Voici cependant une autre façon de procéder : nous allons utiliser, non pas la balise **<span>**, mais la balise **<div>**. De plus, nous allons définir une classe au niveau d'une feuille de style globale pour placer l'image

**Exemple 3.7** Placement d'image

```
<!DOCTYPE HTML PUBLIC "-//W3C //DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Placement d'image</title>
<style>
.v21 {position:absolute;left:70px;top:80px}
</style>
</head>
<body>
<div class="v21">

</div>
</body>
</html>
```

**3.5 Exercices d'applications**

Il est temps de mettre en oeuvre vos talents artistiques ! Mettez un peu de gaieté aux pages précédemment créées, en utilisant une feuille de style CSS externes. Si vous êtes en manque d'inspiration, le site <http://www.oswd.org/> peut sans doute vous être utile.

**4 Formulaires HTML**

Les formulaires sont un point important du langage HTML : ils représentent en effet un moyen simple et efficace pour permettre l'échange de données entre un site internet et un internaute. L'utilisateur peut remplir les champs de formulaires, et ensuite poster ces données au serveur du site en validant ce formulaire.

Les données acceptées par les formulaires sont multiples, on compte parmi elles des zones de texte, des cases à cocher, des listes de sélection, etc...

Nous allons étudier dans ce chapitre ces différentes zones de saisies, et préciser les modalités de transfert des données depuis le navigateur vers le serveur Internet.

## 4.1 Balises <form>

La balise <form> encadre la définition d'un formulaire. Ses attributs permettront de spécifier par exemple la méthode de passage des données vers le serveur Web.

### 4.1.1 Un premier exemple

Voici un exemple de formulaire exploitant les différentes zones interactives disponibles pour le programmeur.

---

#### Exemple 4.1 Exemple de formulaire

---

```
<form method="post" action="" enctype="multipart/form-data">
  <p>Exemple de formulaire HTML</p>
  <p> Zone de texte :
    <input type="text" name="zonetexte">
    <br>

  Case &agrave; cocher :
  <input type="checkbox" name="case" value="checkbox">
  <br>
  Bouton radio :
  <input type="radio" name="radio" value="radiobutton">
  <br>
  Liste d&eacute;roulante :
  <select name="liste">
    <option>Choix 1</option>

    <option>Choix 2</option>
    <option>Choix 3</option>
  </select>
  <br>
  T&eacute;l&eacute;chargement de fichier :
  <input type="file" name="fichier">
  <br>

  Champ cach&eacute; :
  <input type="hidden" name="cache">
</p>
<p>
  <input type="submit" name="Submit" value="Envoyer">
  <input type="reset" name="Submit2" value="Effacer">
</p>
</form>
```

---

### 4.1.2 Balise <form>

La balise <form> a pour attributs :

- **method** : précise la méthode d'envoi choisie pour les informations recueillies par le formulaire, il existe deux possibilités : les méthodes **GET** et **POST** liées au protocole HTTP. Par défaut, cet attribut prend la valeur **get**. Cependant, la méthode **post** est plus généralement utilisée car elle permet de traiter un plus grand nombre d'informations. Nous verrons dans les exercices d'application les différences entre les deux méthodes.
- **action** : cet attribut définit l'URL de destination du formulaire. Il existe plusieurs possibilités parmi lesquelles demander un traitement par le serveur ou envoyer un courrier électronique. Dans le premier cas, le serveur pourra réaliser un traitement des informations avec un script CGI par exemple. Cet attribut ressemblera alors à : **<form action="http://www.monserveur.com/cgi-bin/moncgi.pl" ...>** Pour un envoi par courrier électronique : **<form action="mailto:monmail@monserveur.com" ...>**. **Le destinataire du courrier recevra** alors un message contenant la valeur saisie par l'internaute dans chaque champ du formulaire est envoyé.

- **enctype** : spécifie la méthode d’encodage du formulaire. Ce champ est optionnel, la valeur **text/plain** permet une transmission des informations dans un format texte lisible directement par le destinataire.

### 4.1.3 Balises de formulaires

**4.1.3.1 Boutons** Les boutons permettent de valider (d’envoyer) ou d’effacer les champs du formulaire. Il existe la possibilité d’utiliser la représentation graphique par défaut du HTML, ou de placer une image à la place de ces boutons, en conservant les mêmes comportements. L’exemple précédent utilise les boutons standards du HTML, alors que l’exemple ci-dessous fait appel à des images :

---

#### Exemple 4.2 Boutons "images"

---

```
<FORM action="" method="post">
  <P>
    Nom : <INPUT type="text" name="nom"><BR>
    Prénom : <INPUT type="text" name="prenom"><BR>
    email : <INPUT type="text" name="email"><BR>
    <INPUT type="radio" name="sexe" value="Homme">
    Homme <INPUT type="radio" name="sexe" value="Femme"> Femme
  <P><BR>
    <BUTTON name="submit" value="submit" type="submit">
    <IMG src="bouton_submit.jpg" alt="envoyer" width="102" height="30">
  </BUTTON>
  <BUTTON name="reset" value="submit" type="reset">
  <IMG src="bouton_cancel.jpg" alt="Annuler" width="102" height="30">
  </BUTTON>
</FORM>
```

---

**4.1.3.2 La balise INPUT** La balise **INPUT** permet de placer des champs de formulaire. Elle accepte les attributs suivants :

- **type = text | password | checkbox | radio | submit | reset | file | hidden | image | button** Cet attribut spécifie le type de contrôle de saisie de données à créer. Les types de champs de saisie sont décrits ci-dessous. Cet attribut vaut **text** par défaut.
- **name** Cet attribut permet d’assigner un nom au contrôle. Ce nom sera appairé avec la valeur courante du contrôle, le couple ainsi constitué étant alors transmis si le formulaire est soumis.
- **value** Cet attribut spécifie la valeur initiale du contrôle. Il est optionnel sauf lorsque le type de contrôle est **radio**, auquel cas il devient obligatoire.
- **size** Cet attribut indique à l’agent utilisateur la largeur initiale du contrôle. La largeur est donnée en pixels, sauf pour les champs de saisie de texte et de mot de passe pour lesquels elle est exprimée en nombre de caractères (entier).
- **maxlength** Lorsque le type de champ est **text** ou **password**, cet attribut spécifie le nombre maximum de caractères qui peut y être entré. Ce nombre peut être supérieur à la valeur de l’attribut **size**, auquel cas l’agent utilisateur fournira un mécanisme de défilement à l’intérieur du champ. La valeur par défaut correspond à un nombre illimité de caractères.
- **checked** Lorsque le type de champ est un bouton **radio**, cet attribut booléen spécifie que le bouton radio est sélectionné. Cet attribut devra être ignoré pour tous les autres types de contrôles.
- **src = url** Lorsque le type de champ est **image**, cet attribut spécifie la localisation de l’image utilisée pour "décorer" graphiquement le bouton de soumission.

L’attribut **type** de l’élément **INPUT** détermine quel contrôle doit être créé.

- **text** Ce type crée un champ de texte monoligne. La valeur émise lors de la soumission du formulaire est le texte entré.
- **password** Comme le type **text**, mais le texte entré est visualisé à l’écran de sorte que les caractères ne puissent être reconnus (par exemple une série d’étoiles). Ce contrôle est utilisé pour l’entrée de données sensibles telles que des mots de passe. La valeur émise lors de la soumission du formulaire est le texte tapé (et non pas le texte affiché (!)).
- **checkbox** Une case à cocher est une sorte de commutateur bipolaire. Lorsqu’elle est cochée, la case est dite "active". Lorsqu’elle est vide, la case est "inactive". La valeur de la case à cocher n’est émise que si cette dernière est active. Plusieurs cases à cocher du même formulaire pourront partager

le même nom. Au moment de la soumission, toute case à cocher "activée" y compris celles de nom semblable émet une paire nom/valeur dans laquelle le nom sera identique. Ceci permettra aux utilisateurs de choisir des valeurs multiples pour une propriété unique.

- **radio** Un bouton radio est aussi un commutateur bipolaire. Lorsqu'il est marqué, le bouton radio est dit "active". Lorsqu'il est vide, il est réputé "inactive". La valeur du bouton radio n'est émise que si ce dernier est actif. Plusieurs boutons radio du même formulaire pourront partager le même nom. Cependant, seul l'un d'entre eux pourra être actif à la fois. Lorsque l'un des bouton radio est marqué, tous les autres portant le même nom sont automatiquement désélectionnés. Pour cet ensemble de boutons radio, il ne peut donc être émis qu'une seule paire nom/valeur.
- **submit** Crée un bouton de soumission. Lorsque l'utilisateur clique sur ce bouton, le contenu du formulaire est soumis au programme spécifié par la localisation définie par l'attribut **action** de l'élément **FORM** englobant.
- **image** Crée un bouton de soumission graphique. La valeur de l'attribut **src** spécifie l'URL de l'image qui servira de représentation graphique du bouton. Certains utilisateurs ne pourront visualiser cette image. Nous recommandons fortement d'adjoindre la définition d'un attribut **alt** valant pour alternative textuelle de l'image
- **reset** Crée un bouton de réinitialisation. Lorsque ce bouton est activé par l'utilisateur, les valeurs de tous les contrôles du formulaire sont remises à leur valeur initiale, telle que mentionnée dans l'attribut **value**. Le couple nom/valeur d'un bouton de réinitialisation n'est jamais envoyé lors de la soumission d'un formulaire.
- **button** Crée un bouton poussoir qui n'a aucun comportement prédéfini. Le comportement de ce bouton sera défini par ailleurs, en lui associant des scripts clients déclenchés lorsque certains événements affectant ce bouton surviennent (par exemple lorsqu'on clique dessus). La valeur de l'attribut **value** est utilisée comme libellé du bouton.
- **hidden** Crée un élément qui ne doit pas être représenté par l'agent utilisateur. Cependant, le couple nom/valeur de cet élément sera joint aux données envoyées lors de la soumission du formulaire. Ce type de contrôle sera en général utilisé pour enregistrer des données d'échanges client/serveur qui seraient autrement perdues du fait de la nature volatile des processus HTTP. Les valeurs des contrôles **INPUT** de type **hidden** sont envoyées lors de la soumission du formulaire. Ceci resterait vrai pour tout contrôle qui ne pourrait être affiché pour des raisons de paramétrage de style. Le contrôle suivant, bien qu'invisible aux yeux de l'utilisateur, émettra sa valeur vers le serveur lorsque le formulaire sera soumis.
- **file** Demande à l'utilisateur de désigner un fichier. Lorsque le formulaire est soumis, le contenu de ce fichier sera transmis au serveur comme une valeur de n'importe quel autre contrôle.

## 4.2 Événements intrinsèques

Un script client est un programme qui peut accompagner un document HTML ou y être directement inséré. Le programme s'exécute sur la machine cliente au moment où le document se charge, ou à d'autres moments particuliers, par exemple lorsqu'un lien est activé. Le support des scripts par HTML est indépendant du langage utilisé pour l'écriture de ces derniers. Les scripts offrent aux auteurs de documents électroniques une méthode pour étendre considérablement les possibilités du HTML en lui ajoutant une interactivité et une dynamisation importante.

Un document HTML peut se voir attaché deux types de scripts :

- Ceux qui ne sont exécutés qu'une fois au moment du chargement du document par l'agent utilisateur. Ils sont codés dans un élément **SCRIPT** et sont exécutés au chargement du document. Les agents utilisateurs qui ne supportent pas les scripts, ou pour lesquels les scripts ont été désactivés pourront inclure des alternatives de contenu spécifiées dans un élément **NOSCRIPT**.
- Ceux qui sont exécutés à chaque fois qu'un événement précis survient. Ceux-ci peuvent être associés à un certain nombre d'éléments via des attributs d'événements intrinsèques.

Les formulaires HTML, et plus précisément les champs qui les composent, ont les événements intrinsèques suivants :

- **onfocus** L'événement onfocus lorsque l'élément reçoit le "focus" soit par action de l'organe de pointage, soit par la navigation tabulée. Cet attribut est reconnu par les éléments suivants : **LABEL**, **INPUT**, **SELECT**, **TEXTAREA**, et **BUTTON**.
- **onblur** L'événement onblur intervient lorsque l'élément perd le "focus", soit suite à une action de l'organe de pointage, soit par navigation tabulée. Il sera utilisable sur les mêmes éléments que l'événement **onfocus**.

- **onkeypress** L'événement **onkeypress** intervient lorsqu'une touche du clavier est enfoncée puis relâchée immédiatement alors que l'élément a le "focus". Cet attribut est reconnu par la plupart des éléments
- **onsubmit** L'événement **onsubmit** intervient lorsque le formulaire est soumis. Il n'est applicable que dans le contexte d'un élément **FORM**.
- **onreset** L'événement **onreset** intervient lorsqu'un formulaire est réinitialisé. Il n'est applicable que dans le contexte d'un élément **FORM**.
- **onselect** L'événement **onselect** intervient lorsque l'utilisateur sélectionne du texte dans un champ de texte. Cet attribut est exploitable pour des éléments **INPUT** et **TEXTAREA**.
- **onchange** L'événement **onchange** lorsque l'élément perd le "focus" ET sa valeur a été modifiée depuis le dernier instant où il a été activé. Cet attribut est applicable aux éléments suivants : **INPUT**, **SELECT**, et **TEXTAREA**.

La programmation des scripts à proprement parler n'est pas du ressort de ce cours.

## 5 Evaluation

Vous allez maintenant devoir commencer à construire le site "projet", celui qui est présenté sur la page de garde de la formation. Il s'agit de concevoir des pages (statiques dans un premier temps, dynamique ensuite), qui présente la formation *Initiation Logiciels Libres*. Des exemples de pages vous sont proposés.

Ces pages ne contiennent à l'origine aucun tableau, toutes les mises en page sont faites avec CSS. En vous inspirant de ce modèle, concevez en totalité les pages suivantes :

- la page `index.html` qui contient les textes d'accueil
- la page de style `style.css`
- la page `stagiaire.html` qui contient votre fiche stagiaire
- la page `module.html` qui contient le module HTML
- la page `entreprise.html` qui contient en faite vos liens

## 6 Références

### 6.1 Références

- [1] *openweb.eu.org - Pour les standards du Web*, <http://openweb.eu.org/> .
- [2] *Alsacreation.com - Comment choisir sa DTD*, <http://css.alsacreation.com/Bases-et-indispensables/DTD-comment-choisir> .
- [3] *w3C - Recommandations HTML*, <http://www.w3c.org> .
- [4] *Open Source Web Design*, <http://www.oswd.org/> .