HELHa
Haute École Louvain en Hainaut

Syllabus

Catégorie Économique

Michaël Anckaert

Concepts fondamentaux des technologies Internet

UE : EC1B05INDE1
AA : 17-EC1B05INDE1/3

Année académique 2014-2015

Première partie

Les bases du HTML 5

CHAPITRE 1

LA CRÉATION DES SITES WEB STATIQUES

1.1 L'HISTOIRE DES SITES WEB

A l'heure où la tendance est d'utiliser le HTML 5 pour la création des sites Web, il est à savoir que les sites Web en ont fait du chemin...

En effet, en observant la définition 1, nous pouvons comprendre que le web utilise le réseau Internet. Ce dernier permet d'exploiter différentes technologies Internet afin de fournir à l'être humain de multiples moyens de communications.

Définition 1 (Web)**.**

La toile ou encore le World Wide Web (www) est un système hypertexte utilisant le protocole http (hypertext transfer protocol), permettant de visiter des pages sur le réseau Internet.

Parmi les technologies Internet, nous retrouvons notamment :

- le Web
- les courriers électroniques
- les messageries instantanées
- les réseaux sociaux
- ...

Donc, il ne faut pas faire l'amalgame entre web et technologies Internet : l'un est une partie de l'autre et non l'inverse !

Le langage par excellence qui nous permet de communiquer sur le Web est le langage HTML que l'on retrouve en définition 2. Ce langage sera la base de notre étude du web. Grâce à lui, nous pourrons, par la suite, nous orienter vers des langages évolués comme le PHP, le Javascript...

Définition 2 (Langage HTML)**.**

HTML (hypertext markup language) est le langage universel de balisage utilisé pour communiquer sur le Web.

A l'origine, ce langage, inventé par Tim Berners-Lee [1] en 1991, est issu du langage SGML (Standard Generalized Markup Language) qui date de 1986. Ce dernier étant très lourd à adapter au monde du web fut donc allégé par sa version HTML.

De cette époque à nos jours, ce langage a considérablement évolué, voyons les principales évolutions :

- **HTML 1** : c'est la toute première version créée en 1991.
- **HTML 2** : la deuxième version du HTML apparaît en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML. Les règles et le fonctionnement de cette version sont donnés par le W3C mentionné à la définition 3 (tandis que la première version a été créée par un seul homme).
- **HTML 3** : version apparue en 1996, elle ajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.
- **HTML 4** : il s'agit de la version la plus répandue du HTML (plus précisément, il s'agit de HTML 4.01). Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style que nous étudierons plus loin dans notre cours.
- **HTML 5** : c'est La dernière version. Encore assez peu répandue, elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc.

FIGURE 1 – Tim Berners-Lee, créateur du HTML

1. né le 8 juin 1955 à Londres, est un citoyen britannique, principal inventeur du World Wide Web (WWW) au tournant des années 1990. En juillet 2004, il est anobli par la reine élisabeth II pour ce travail et son nom officiel devient Sir Timothy John Berners-Lee. Depuis 1994, il préside le World Wide Web Consortium (W3C), organisme qu'il a fondé.

Définition 3 (W3C).

Est aussi appellé le World Wide Web Consortium est un organisme de normalisation à but non-lucratif, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XML, CSS...

Le W3C fonctionne comme un consortium international, il regroupe au 26 février 2013, 387 entreprises partenaires. Il a comme leitmotiv : "un seul web partout et pour tous".

Afin de parvenir à ses objectifs, le créateur du langage Web avait besoin d'un lecteur pour pouvoir déchiffrer son système de codage et le présenter à un public lambda. Nous allons donc nous intéresser à présent au fonctionnement du Web.

1.2 Le fonctionnement du Web

HTML est donc le langage universel utilisé pour communiquer sur le Web. L'information est ainsi transportée sur cette gigantesque toile de réseaux interconnectés qu'est Internet, pour aboutir sur l'ordinateur de votre lecteur grâce à un programme appelé navigateur ou browser.

Définition 4 (Navigateur).

Est un logiciel conçu pour consulter le World Wide Web. Techniquement, c'est au minimum un client HTTP.

Nous avons donc deux interlocuteurs : le navigateur de votre lecteur et votre lecteur lui-même. Ce logiciel, que l'on appelle un browser, vous permet de surfer sur le Net et d'afficher sur votre écran les "pages" qu'il a interceptées. Il y a, hélas, beaucoup de marques et de types de browsers différents. Des simples, des archaïques ou des sophistiqués... Les plus connus sont Internet Explorer, Mozilla Firefox, Google Chrome, Safari et Opéra.

FIGURE 2 – Les principaux navigateurs

1.2.1 Les particularités

Nous nous devons de bien comprendre les particularités suivantes pour la bonne utilisation du langage que nous étudions :

- Chaque navigateur a sa propre façon de travailler. A la différence de votre traitement de texte préféré qui restitue exactement votre document sur une feuille de papier avec votre police de caractères et votre mise en page, nous ne saurons jamais exactement ce que le navigateur de votre lecteur du bout du monde affichera sur l'écran de celui-ci.

- En HTML, nous n'avons pas la maîtrise totale de votre document. Pour transiter le plus rapidement possible sur les lignes téléphoniques, on a adopté un format de texte très compact mais aussi (par conséquence) peu sophistiqué. C'est le bon vieux format de texte pur et dur, sans fioritures du Bloc-notes ou Notepad de Windows par exemple. Et de plus ce format ASCII a été amputé d'un bit (7 bits au lieu de 8) ! Nous serons donc privé de certains caractères spéciaux comme le é pour lesquels il faudra passer par des codes particuliers (voir annexes).

- Mais récompense suprême... HTML est un langage universel qui s'adapte à toutes les plateformes (Windows, Macintoch, Unix, OS/2...).

- En plus du texte adressé à votre lecteur, il vous faudra inclure des instructions pour le navigateur de celui-ci.

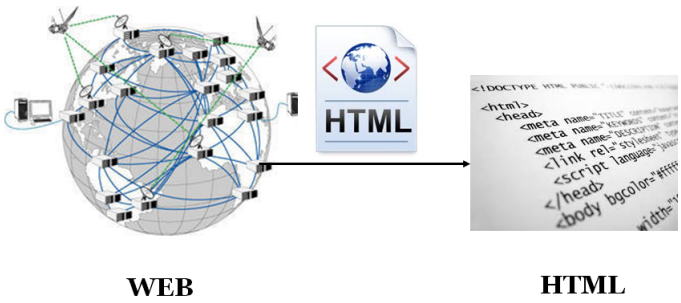


FIGURE 3 – Le Web parle HTML !

Astuce 1 (Le jeu de caractères utf-8)**.**

UTF-8 (abréviation de l'anglais Universal Character Set Transformation Format - 8 bits) est un codage de caractères informatiques conçu pour coder l'ensemble des caractères du "répertoire universel de caractères codés", initialement développé par l'ISO dans la norme internationale ISO/CEI 10646, aujourd'hui totalement compatible avec le standard Unicode, en restant compatible avec la norme ASCII limitée à l'anglais de base (et quelques autres langues beaucoup moins fréquentes), mais très largement répandue depuis des décennies. L'UTF-8 est utilisé par 78,1 % des sites web en novembre 2013. De par sa nature, UTF-8 est d'un usage de plus en plus courant sur Internet, et dans les systèmes devant échanger de l'information. Il s'agit également du codage le plus utilisé dans les systèmes GNU, Linux et compatibles pour gérer le plus simplement possible des textes et leurs traductions dans tous les systèmes d'écritures et tous les alphabets du monde.

1.2.2 Le système de balisage

HTML est donc un système utilisant des balises. Voyons de quoi il s'agit avec la définiton 5.

Définition 5 (Les balises)**.**

La balise ou le tag est une instruction qui est différenciée du texte par les signes **<** et **>**. Quand nous écrivons les balises de notre page HTML, nous devons garder à l'esprit :

1. qu'une balise marque une action pour le navigateur (ce qu'il doit faire...).
2. que les attributs précisent les modalités de cette action (comment il doit le faire...).

De manière générale en HTML, tout contenu, c'est-à-dire : le texte, les images ou les media les plus divers doivent être enfermés dans une balise. Cette balise est fixée par la version du HTML que nous utilisons, contrairement au XML, où là, c'est l'utilisateur qui définit ses balises. Les balises ne sont pas "case sensitive" donc il est équivalent d'écrire **<HTML>**, **<html>**, **<Html>**, etc.

Exemple 1. *Syntaxe d'une balise*

```
1 <b> mon texte en gras </b>
2 <br/>
3 <img src="aragorn.png" alt="Aragorn"/>
```

Nous pouvons remarquer à l'exemple 1 que nous avons un cas particulier au niveau de la syntaxe de nos balises. En effet, aux lignes 2 et 3, il n'y a aucune manipulation de texte. La balise de la ligne 2 fait un saut de ligne et la balise de la ligne 3 insère une image. Ce cas particulier de balisage, appelé balise orpheline, sera fermé dès l'ouverture. Cependant nous retrouvons notre cas général à la ligne 1 soit une balise en paires.

CHAPITRE 2

LA PAGE WEB DE BASE

2.1 LES OUTILS DE BASE

Afin de réaliser des pages HTML et de tester le résultat de celles-ci, nous avons besoin d'une série d'outils tels qu' :

- un **éditeur de texte** (DreamWeaver - voir annexes).
- un **navigateur**.

Astuce 2 (Le choix du navigateur)**.**

Chaque utilisateur lambda navigue avec son navigateur préféré, afin de toucher un maximum d'utilisateurs, il faut s'assurer que notre site Web soit accessible par plusieurs navigateurs. Il est clair que pour un site statique comme le notre, la portabilité n'est pas aussi importante que pour un site dynamique où là, il y a une clientèle, un public cible, une part de marché à conquérir. Donc, il est conseillé de tester notre site sur plusieurs navigateurs.

2.1.1 L'éditeur de texte

Incroyable mais vrai : nous pouvons tout à fait créer un site web uniquement avec le bloc-Notes, le logiciel d'édition de texte intégré par défaut à Windows. Notons qu'à l'époque du HTML 1.0, il n'y avait que ce genre d'éditeur pour développer les sites web.

Il y a cependant des logiciels plus puissants aujourd'hui et personne n'utilise vraiment plus le bloc-Notes. On peut classer

ces logiciels de création de site web en deux catégories :

- **Les WYSIWYG** (What You See Is What You Get - Ce Que Vous Voyez Est Ce Que Vous Obtenez) : ce sont des programmes qui se veulent très faciles d'emploi, ils permettent de créer des sites web sans apprendre de langage particulier. Parmi les plus connus d'entre eux : Mozilla Composer, Microsoft Expression Web, Dreamweaver... et même Word ! Le principal défaut de certains est la qualité souvent assez mauvaise du code HTML et CSS qui est automatiquement généré par ces outils.

- **Les éditeurs de texte** : ce sont des programmes dédiés à l'écriture de code. On peut en général les utiliser pour de multiples langages, pas seulement HTML et CSS. Voici une liste non exhaustive par plateforme :
  - ▷ Windows : NotePad++, jEdit, PSpad, ConTEXT...
  - ▷ Mac OS X : jEdit, Smultron, TextWrangler...
  - ▷ Linux : gEdit, Kate, vim, Emacs, jEdit...

- **Les SGC** : ce sont des systèmes de gestion de contenu ou SGC (Content Management System ou CMS) est une famille de logiciels destinés à la conception et à la mise à jour dynamique de sites Web ou d'applications multimedia. Au travers d'interfaces web, les SGC sont accessibles quel que soit le type de système d'exploitation au moyen d'un navigateur Web. Ainsi, les utilisateurs n'ont pas besoin d'installer de logiciels spécifiques supplémentaires. Grâce aux standards du web, les SGC offrent donc un format de données lisible (HTML et ses dérivés RIA), imprimable et stockable par tous, ce qui facilite l'échange et l'accessibilité des documents. Un SGCD permet de gérer et de générer le code source des projets pour l'exporter. Nous retrouvons notamment : WordPress, Joomla, Prestashop...

Astuce 3 (Le choix de l'éditeur)**.**

Dreamweaver à la chance d'appartenir à une suite de logiciels du type Adobe tout comme Photoshop. Bien qu'un développeur Web n'est pas un infographiste, ce dernier sera tout de même amené à effectuer des modifications mineures sur des images. La combinaison Dreamweaver - Photoshop permet de mettre en oeuvre un résultat rapide et fiable. Dreamweaver à l'avantage de gérer tous les langages web : HTML, CSS, JS, PHP, C#, XML..., ce qui lui confère un atout non négligeable. De plus, il permet de visualiser directement le résultat sur les navigateurs même celui obtenu sur des smartphones. Pour l'aspect dynamique, nous pouvons connecter une base de données ou directement répercuter les changements de code sur l'hébergeur.

2.1.2 Les navigateurs sur ordinateur

Sans s'en douter, un navigateur est un programme extrêmement complexe. En effet, comprendre le code HTML n'est pas simple pour un novice. Le principal problème est que les différents navigateurs n'affichent pas le même site exactement de la même façon ! Il faudra prendre notre mal en patience et prendre l'habitude de vérifier régulièrement que notre site fonctionne correctement sur la plupart des navigateurs. Il est très important de comprendre les différences entre navigateurs car les navigateurs n'affichent pas toujours un même site web exactement de la même façon. La cause est issue du fait que les navigateurs ne connaissent pas toujours les dernières fonctionnalités de HTML et CSS (feuilles de style que nous étudierons plus tard). Par exemple, Internet Explorer a longtemps été en retard sur certaines fonctionnalités CSS et paradoxalement, il a aussi été en avance sur quelques autres voire en être le précurseur.

Pour compliquer les choses, plusieurs versions des navigateurs coexistent :

- Firefox 2 ... Firefox 30
- Internet Explorer 6 ... Internet Explorer 11
- Chrome 8 ... Chrome 35
- etc.

Navigateur	OS
Google Chrome Google Chrome	Windows Mac Linux
Mozilla Firefox Firefox	Windows Mac Linux
Internet Explorer IE	Windows
Safari Safari	Windows Mac
Opera Opera	Windows Mac Linux

Chaque version prend en charge de nouvelles fonctionnalités mais, si les utilisateurs ne mettent pas à jour leur(s) navigateur(s), cela devient un problème pour les webmasters comme vous qui créent des sites web. Chrome a résolu en grande partie le problème en mettant en place des mises à jour automatiques, sans intervention de l'utilisateur. Les utilisateurs de Firefox ne pensent pas toujours à mettre à niveau leur navigateur ; quant à Internet Explorer, les utilisateurs sont d'autant moins incités à mettre à jour leur navigateur que les dernières versions nécessitent aussi de passer à une version récente de Windows (Internet Explorer 9 n'est pas disponible pour Windows XP, par exemple).

Les sites `www.normansblog.de` et `www.caniuse.com` tiennent notamment à jour une liste des fonctionnalités CSS prises en charge par les différentes versions de chaque navigateur.

La plupart des soucis viendra le plus souvent des anciennes versions d'Internet Explorer (IE6, IE7, IE8). Paradoxalement, avec sa version IE11, Microsoft a réussi le pari de corriger ses défaillances passées. Cependant, nous devrons vérifier comment le site s'affiche sous ses anciennes versions... Nous devons nous attendre à des surprises ! Nous devons vérifier surtout que notre site s'affiche sans erreur, sans chercher à obtenir exactement le même rendu sur les vieilles versions de ces navigateurs.

Astuce 4 (IETester).

Il existe sous Windows un programme appelé IETester : `http://www.my-debugbar.com/wiki/IETester/HomePage`. Il permet de vérifier le rendu de son site sous différentes versions d'Internet Explorer. à noter que ce programme est relativement instable (il plante souvent) mais il a le mérite d'exister.

2.1.3 Les navigateurs sur mobile

En plus des navigateurs sur ordinateur, il faut savoir qu'il existe des variantes de ces navigateurs conçues pour les téléphones portables, en particulier pour les smartphones. De plus en plus de personnes consultent aujourd'hui des sites web sur leur portable, nous devons donc connaître un minimum le fonctionnement des navigateurs des téléphones.

En fait, nous n'allons pas être dépaysés : la plupart des navigateurs sur smartphones sont les mêmes que sur ordinateur, dans une version plus légère adaptée aux mobiles. Tout dépend du type de téléphone :

- **iPhone** : sur l'iPhone d'Apple, le navigateur utilisé est Safari Mobile. Il s'agit d'une version light et néanmoins très complète de Safari pour ordinateur.

- **Android** : les portables sous Android bénéficient du navigateur Chrome Mobile. Là encore, il s'agit d'une version adaptée aux mobiles.

- **Windows Phone** : sous Windows Phone, on retrouve... Internet Explorer Mobile ! Le principe est le même que pour les précédents navigateurs : il s'agit d'une version dédiée aux mobiles.

- **Blackberry** : les Blackberry font exception car ils ont leur propre navigateur (il n'existe pas d'équivalent sur ordinateur). Néanmoins, les versions les plus récentes de ce navigateur se basent sur un noyau commun à Safari et Chrome (il s'agit du moteur de rendu Webkit). Par conséquent, l'affichage est en général proche de celui proposé par Safari et Chrome.

Les navigateurs pour mobiles prennent en charge la plupart des dernières fonctionnalités de HTML et CSS. De plus, le systçme de mise à jour automatisé des mobiles nous garantit que les utilisateurs auront le plus souvent les dernières versions.

Notons néanmoins que des différences existent entre ces différents navigateurs mobiles et qu'il est conseillé de tester son site sur ces appareils aussi ! En particulier, l'écran étant beaucoup moins large, il faudra vérifier que votre site s'affiche correctement.

2.2 Les balises et leurs attributs

2.2.1 Les balises

Nous avons vu que les pages HTML sont remplies de ce qu'on appelle des balises. Celles-ci sont invisibles à l'écran pour nos visiteurs, mais elles permettent à l'ordinateur de comprendre ce qu'il doit afficher. Nous avons dit également que les balises se repèrent facilement car elles sont entourées de "chevrons", c'est-à-dire des symboles **<** et **>**.

A quoi est-ce qu'elles servent ? Elles indiquent la nature du texte qu'elles encadrent. Elles veulent dire par exemple : "Ceci est le titre de la page", "Ceci est une image", "Ceci est un paragraphe de texte", etc.

Nous distinguons deux types de balises : les balises en paires et les balises orphelines.

2.2.1.1 Les balises en paires

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Voici la syntaxe :

Exemple 2. *Une balise en paires*

```
1 <title>Ceci est un titre</title>
```

Nous distinguons une balise ouvrante **<title>** et une balise fermante **</title>** qui indique que le titre se termine. Cela signifie pour l'ordinateur que tout ce qui n'est pas entre ces deux balises... n'est pas un titre.

2.2.1.2 Les balises orphelines

Ce sont des balises qui servent le plus souvent à insérer un élément à un endroit précis (par exemple une image). Il n'est pas nécessaire de délimiter le début et la fin de l'image, nous voulons juste dire à l'ordinateur "Insère une image ici".

Exemple 3. *Une balise orpheline*

```
1 <img src="aragorn.png" alt="Aragorn"
```

2.2.2 Les attributs

Comme nous l'avons déjà vu avant, les attributs sont d'une certaine manière les options des balises : ils viennent les complèter pour donner des informations supplémentaires. L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur. Quel en est l'utilité ? Prenons la balise de l'exemple 3 que nous venons de voir. Seule, elle ne servirait pas à grand chose, c'est pourquoi on ajoute un attribut qui indique le nom de l'image à afficher. L'ordinateur comprend alors qu'il doit afficher l'image contenue dans le fichier aragorn.png. Dans le cas d'une balise fonctionnant "par paire", on ne met les attributs que dans la balise ouvrante et pas dans la balise fermante.

2.3 LA STRUCTURE DE LA PAGE

La structure de base d'une page web en HTML5 se limite aux instructions suivantes :

Exemple 4. *La structure de la page*

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="utf-8" />
5         <title>Titre</title>
6     </head>
7     <body>
8     </body>
9 </html>
```

Nous notons que les balises s'ouvrent et se ferment dans un ordre bien précis. Par exemple, la balise **<html>** est la première que l'on ouvre et c'est aussi la dernière que l'on ferme (tout à la fin du code, avec **</html>**). Les balises doivent être fermées dans le sens inverse de leur ouverture. Un exemple :

- **<html><body></body></html>** : correct. Une balise qui est ouverte à l'intérieur d'une autre doit aussi être fermée à l'intérieur.
- **<html><body></html></body>** : incorrect, les balises s'entremêlent.

Astuce 5 (Document bien formé)**.**

Notre éditeur n'est peut être pas conçu pour déceler ce genre d'erreur, ou malgré nos vérifications, ils restent des erreurs de syntaxe. Il est possible de valider son document afin de savoir s'il ce dernier est bien formé. Pour cela nous pouvons utiliser l'adresse suivante : `http://html5.validator.nu`

Astuce 6 (Indentation).

Nous remarquons à l'exemple 4 qu'il y a des espaces au début de certaines lignes pour "décaler" les balises. Ce n'est pas obligatoire et cela n'a aucun impact sur l'affichage de la page, mais cela rend le code source plus lisible. On appelle cela l'indentation. Dans votre éditeur, il suffit d'appuyer sur la touche <Tab> pour avoir le même résultat.

2.3.1 Le doctype

Le langage HTML 5 est une amélioration du langage HTML 4, avec des simplifications par rapport à la version XHTML qui était le standard avant lui. Tout document peut débuter de la même manière par la déclaration du **<!doctype>**. Cette déclaration est indispensable car c'est elle qui indique qu'il s'agit bien d'une page web HTML. Ce n'est pas vraiment une balise comme les autres (elle commence par un point d'exclamation), nous pouvons considérer que c'est un peu l'exception qui confirme la règle. Cette ligne du doctype était autrefois incroyablement complexe. Il était impossible de la retenir de tête. Pour XHTML 1.0, il fallait écrire : **<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http ://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**. Dans le cadre de HTML5, il a été décidé de la simplifier, pour le plus grand bonheur des webmasters. Quand nous voyons une balise doctype courte **<!DOCTYPE html>**, cela signifie que la page est écrite en HTML5.

2.3.2 La balise <html>

La balise **<html>** est la racine de notre document, toutes les balises qu'elle englobe sont ses filles. Notons que ses deux balises filles sont la balise **<head>** et la balise **<body>** comme présenté à la figure 4.

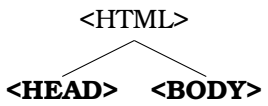


FIGURE 4 – Document HTML de base sous forme d'arbre

2.3.3 L'en-tête <head> et le corps <body>

Une page web est donc constituée de deux parties :

– **L'en-tête <head>** : cette section donne quelques informations générales, appelées métadonnées, sur la page. Cette section est généralement assez courte. Les informations que contient l'en-tête ne sont pas affichées sur la page, elles sont cependant

très importantes et elles sont de 6 types : **<base/>**, **<link/>**, **<meta/>**, **<script>**, **<style>** et **<title>**. Nous y reviendrons dans les chapitres suivants.

– **Le corps <body>** : c'est là que se trouve la partie principale de la page. Tout ce que nous écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

Pour le moment, le corps est vide (nous y reviendrons plus loin). Intéressons-nous par contre aux deux balises contenues dans l'en-tête...

2.3.4 L'encodage (charset)

Cette balise indique l'encodage utilisé dans votre fichier .html. Sans rentrer dans les détails, car cela pourrait vite devenir compliqué, l'encodage indique la façon dont le fichier est enregistré. C'est lui qui détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, caractères arabes, etc.). Il y a plusieurs techniques d'encodage portant des noms bizarres et utilisées en fonction des langues : ISO-8859-1, OEM 775, Windows-1253... Une seule cependant devrait être utilisée aujourd'hui autant que possible : UTF-8. Cette méthode d'encodage permet d'afficher sans aucun problème pratiquement tous les symboles de toutes les langues de notre planète ! C'est pour cela que que le jeu de caractères UTF-8 est utilisé. Il faut aussi que notre fichier soit bien enregistré en UTF-8. C'est le cas le plus souvent sous Linux par défaut mais, sous Windows, il faut généralement le dire au logiciel.

Astuce 7 (Jeu de caractères de la page)**.**

Si nous avons un problème d'affichage des accents plus tard dans votre page web, c'est qu'il y a un problème avec l'encodage. Vérifions que la balise **<charset>** indique bien UTF-8 et que votre fichier est enregistré en UTF-8 (votre éditeur de texte est capable de vous le dire).

2.3.5 Le titre principal de la page

C'est le titre de notre page, probablement l'élément le plus important ! Toute page doit avoir un titre qui décrit ce qu'elle contient. Il est conseillé de garder le titre assez court (moins de 100 caractères en général). Le titre ne s'affiche pas dans notre page mais en haut de celle-ci (souvent dans l'onglet du navigateur). Il faut savoir que le titre apparaît aussi dans les résultats de recherche comme sur Google.

Nous pouvons représenter notre page de base entière comme l'arbre de la figure 6.

2.4 LES COMMENTAIRES

Un commentaire en HTML est un texte qui sert simplement de mémo. Il n'est pas affiché, il n'est pas lu par l'ordinateur, cela ne change rien à l'affichage de la page. L'utilité d'un commentaire est d'informer d'autres personnes ou encore nous permettre de nous rappeler de notre code source. Nous pouvons utiliser les commentaires pour laisser des indications sur le fonctionnement de notre page.

Exemple 5. *Les commentaires*

```
1  <!DOCTYPE html>
2  <html>
3      <!-- En-tte de la page -->
4      <head>
5          <meta charset="utf-8" />
6          <title>Titre</title>
7      </head>
8      <body>
9      <!-- Corps de la page -->
10     </body>
11 </html>
```

FIGURE 5 – Représentation du titre lors d'une recherche dans google

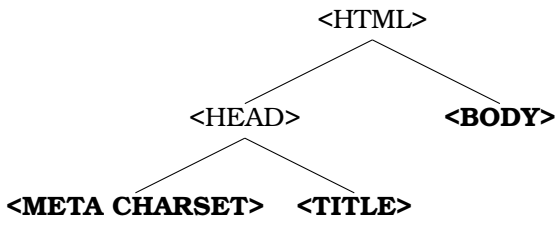


FIGURE 6 – Document HTML de base complet sous forme d'arbre

Astuce 8 (Tout le monde peut voir vos commentaires... et tout votre code HTML!)**.**

Astuce importante : tout le monde peut voir le code HTML de notre page une fois celle-ci mise en ligne sur le Web. Il suffit de faire un clic droit sur la page et de sélectionner "Afficher le code source de la page" (l'intitulé peut changer selon notre navigateur). Notons que la loi sur les droits d'auteur reste d'application !

2.5 L'ENREGISTREMENT DE LA PAGE

Toute page HTML, lors de son enregistrement, doit porter l'extension .html[2] et porter le nom index s'il s'agit de l'entrée principale de notre site. Donc, si notre page est l'entrée principale de notre site, son nom et son extension seront : index.html.

2. .htm est l'ancienne extension à l'époque du DOS.

CHAPITRE 3

LES STRUCTURES DE L'INFORMATION

Jusqu'à présent notre page HTML n'affiche qu'une page blanche avec un titre. Nous n'irons pas très loin avec ce résultat. En effet, il faut organiser sa page afin de transmettre un certain nombre d'informations. Nous allons découvrir de nombreuses balises HTML dans ce chapitre. Certaines existent depuis la toute première version de HTML, d'autres ont été introduites plus récemment dans HTML5.

Nous allons voir successivement dans ce chapitre :
- comment rédiger des paragraphes ;
- comment structurer sa page avec les titres ;
- comment donner de l'importance à certains mots ;
- comment organiser les informations sous forme de listes.

3.1 LES PARAGRAPHES

La plupart du temps, lorsqu'on écrit du texte dans une page web, on le fait à l'intérieur de paragraphes. Le langage HTML propose justement la balise **<p>** pour délimiter les paragraphes.

Exemple 6. *La syntaxe de la balise <p>*

```
1 <p>Bonjour et bienvenue sur notre site!</p>
```

Comme nous l'avons vu dans le chapitre précédent, nous écrivons le contenu du site web entre les balises **<body></body>**. Il nous suffit donc de mettre notre paragraphe entre ces deux balises et nous aurons enfin notre première vraie page web avec du texte !

Reprenons donc exactement le même code qu'au chapitre précédent et ajoutons un paragraphe :

Exemple 7. *La balise <p> dans notre page*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>
    <body>
        <p>Bonjour et bienvenue sur notre site!</p>
    </body>
</html>
```

Avec l'ajout de cette balise, notre arbre devient :

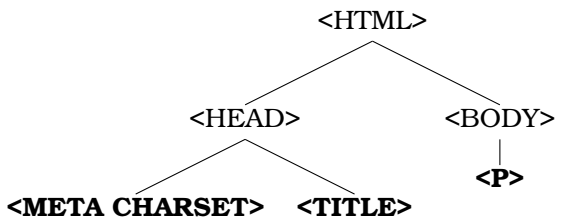


FIGURE 7 – Document HTML avec balise <p> sous forme d'arbre

3.1.1 Les sauts de ligne

Le saut de ligne permet de séparer le contenu de notre page html. Il est possible d'y parvenir de deux manières différentes. La première consiste à structurer le contenu en plusieurs paragraphes comme dans l'exemple 8 ou alors implémenter un retour à la ligne comme dans l'exemple 9.

Exemple 8. *Le saut de ligne avec plusieurs paragraphes dans notre page*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>
    <body>
        <p>Bonjour et bienvenue sur notre site! Ceci est notre
           premier test alors soyez indulgents s'il vous plaît, nous
            apprennons petit à petit comment cela marche.</p>
```

```
9           <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3
                jours quand nous aurons appris un peu plus de choses,
                nous vous assurons que vous allez être surpris!</p>
10      </body>
11  </html>
```

Ce qui donne dans notre arbre :

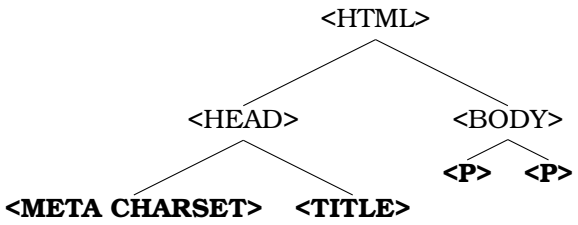
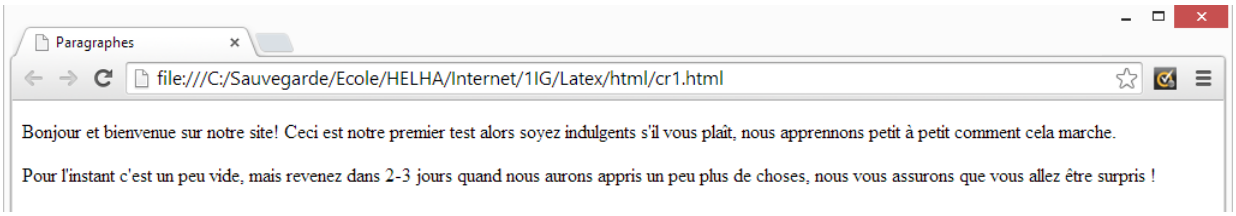


FIGURE 8 – Document HTML avec plusieurs balises <p> sous forme d'arbre

Et de manière visuelle dans un navigateur :



Exemple 9. *Le retour à la ligne dans notre page*

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <meta charset="utf-8" />
5           <title>Paragraphes</title>
6       </head>
7       <body>
8           <p>Bonjour et bienvenue sur notre site! <br/>
9              Ceci est notre premier test alors soyez indulgents s'il
                   vous plaît, nous apprennons petit à petit comment cela
                    marche.</p>
10          <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3
               jours quand nous aurons appris un peu plus de choses,
               nous vous assurons que vous allez être surpris!</p>
11      </body>
12  </html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

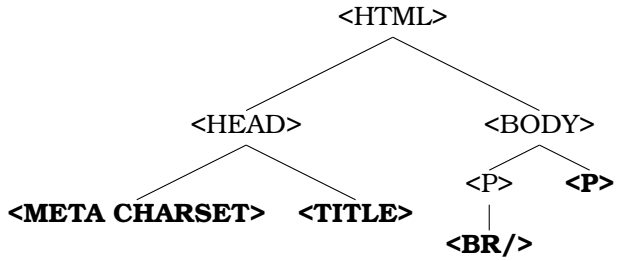
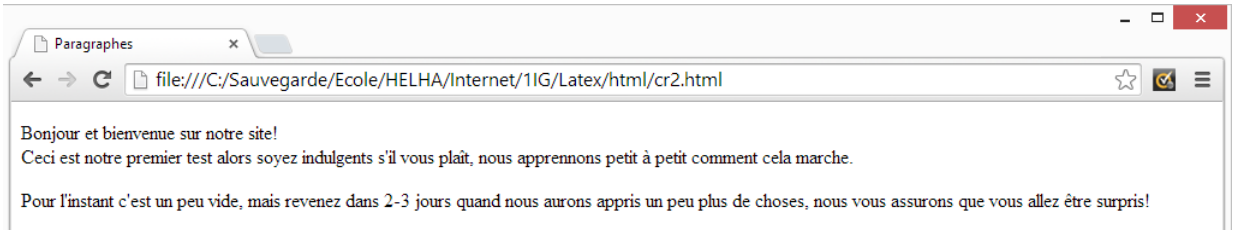


FIGURE 9 – Document HTML avec retour à ligne sous forme d'arbre



3.1.2 Les séparateurs

Il est parfois utile d'insérer une ligne horizontale pour séparer deux parties dans un document. Pour cela, nous utiliserons la balise **<hr />**. A titre d'exemple, nous allons insérer une ligne séparatrice entre deux paragraphes déjà définis dans un document HTML :

Exemple 10. *Les séparateurs dans notre page*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Paragraphes</title>
6      </head>
7      <body>
8          <p>Bonjour et bienvenue sur notre site! <br/>
9             Ceci est notre premier test alors soyez indulgents s'il
                  vous plaît, nous apprennons petit à petit comment cela
                   marche.</p>
10         <hr/>
11         <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3
              jours quand nous aurons appris un peu plus de choses,
              nous vous assurons que vous allez être surpris!</p>
12     </body>
13 </html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

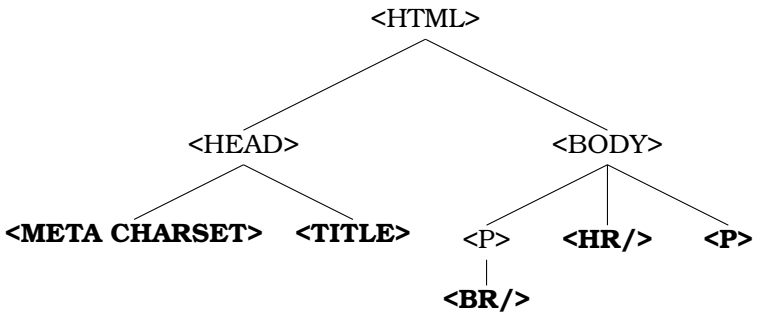
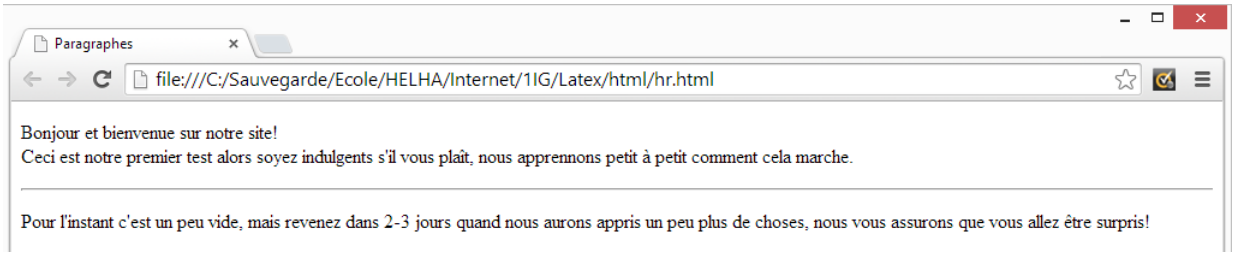


FIGURE 10 – Document HTML avec balise <hr/> sous forme d'arbre



Si nous voulons que la barre ait une largeur plus faible, utilisons l'attribut width et précisons, entre guillemets, le pourcentage de la largeur du navigateur à utiliser.

Il est également possible de modifier l'épaisseur de la barre avec l'attribut size. Affectons-lui la largeur en pixels entre guillemets.

Enfin, il est possible d'annuler l'ombrage dans la barre avec l'attribut noshade.

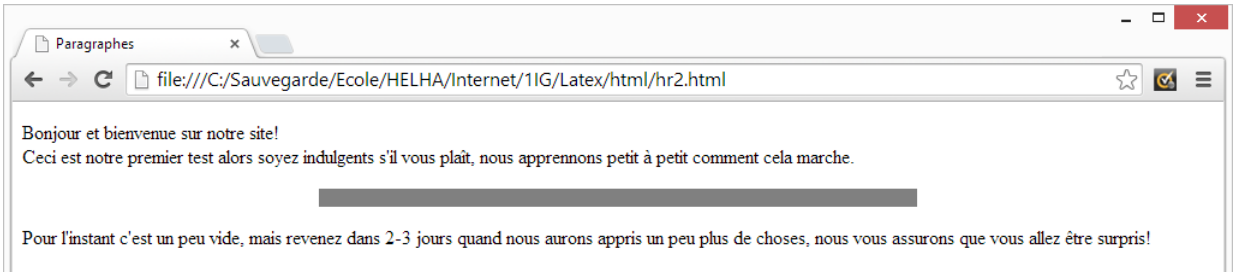
Les attributs **width**, **size** et **noshade** peuvent être utilisés conjointement comme dans l'exemple qui suit :

Exemple 11. *Les séparateurs personnalisés dans notre page*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Paragraphes</title>
6      </head>
7      <body>
8          <p>Bonjour et bienvenue sur notre site! <br/>
9             Ceci est notre premier test alors soyez indulgents s'il
                 vous plaît, nous apprennons petit à petit comment cela
                  marche.</p>
10         <hr width="50%" size="15" noshade />
```

```
11          <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3
                jours quand nous aurons appris un peu plus de choses,
                nous vous assurons que vous allez être surpris!</p>
12      </body>
13  </html>
```

Soit visuellement dans un navigateur :



Cependant, il est préférable d'utiliser les propriétés des CSS.

3.2 Les titres

Lorsque le contenu de notre page va s'étoffer avec de nombreux paragraphes, il va devenir difficile pour nos visiteurs de se repérer. C'est là que les titres deviennent utiles. En HTML, nous sommes veinards, nous avons le droit d'utiliser six niveaux de titres différents avec un degré d'importance pour chacun.

Nous avons donc six balises de titres différentes :

- **<h1></h1>** : titre très important qui sert à afficher le titre de la page au début de celle-ci.
- **<h2></h2>** : titre important.
- **<h3></h3>** : pareil, c'est un titre un peu moins important (nous pouvons le considérer comme un "sous-titre").
- **<h4></h4>** : titre anodin.
- **<h5></h5>** : titre très anodin.
- **<h6></h6>** : titre vraiment très anodin.

Astuce 9 (Hiérarchie des titres)**.**

Il ne faut pas choisir la balise de titre en fonction de la taille qu'elle applique au texte ! Il faut impérativement bien structurer sa page en commençant par un titre de niveau 1 **<h1>**, puis un titre de niveau 2 **<h2>**, etc. Il ne devrait pas y avoir de sous-titre sans titre principal ! Si nous voulons modifier la taille du texte, sachons que nous apprendrons à faire cela en CSS.

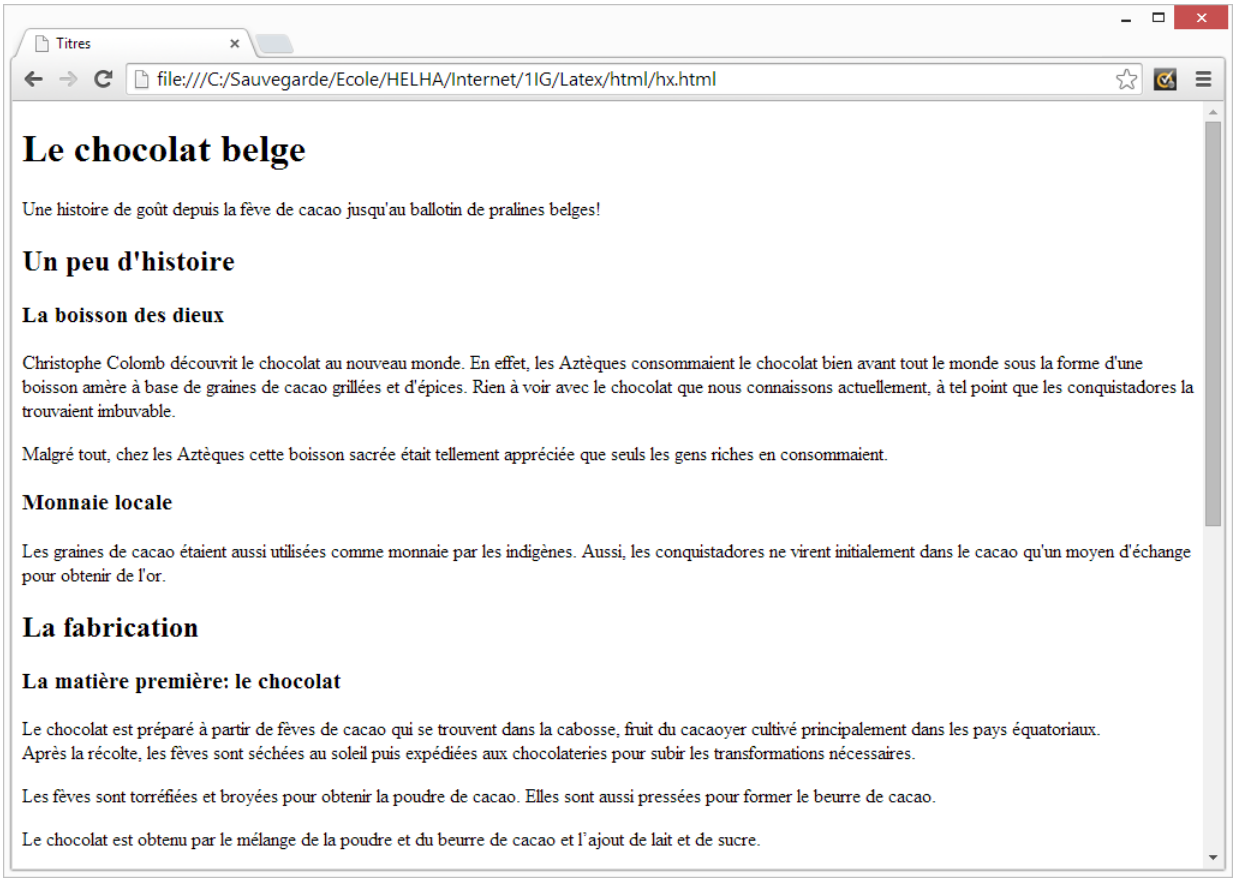
Afin de nous mettre l'eau à la bouche, voici un exemple qui séduira plus d'un :

Exemple 12. *Les titres dans notre page*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titres</title>
    </head>
    <body>
        <h1>Le chocolat belge</h1>
        <p>Une histoire de goût depuis la fève de cacao jusqu'au
           ballotin de pralines belges!</p>
        <h2>Un peu d'histoire</h2>
        <h3>La boisson des dieux</h3>
        <p>Christophe Colomb découvrit le chocolat au nouveau monde.
           En effet, les Aztèques consommaient le chocolat bien
           avant tout le monde sous la forme d'une boisson amère à
           base de graines de cacao grillées et d'épices. Rien à
           voir avec le chocolat que nous connaissons actuellement,
           à tel point que les conquistadores la trouvaient
           imbuvable.</p>
        <p>Malgré tout, chez les Aztèques cette boisson sacrée était
           tellement appréciée que seuls les gens riches en
           consommaient.</p>
        <h3>Monnaie locale</h3>
        <p>Les graines de cacao étaient aussi utilisées comme monnaie
            par les indigènes. Aussi, les conquistadores ne virent
           initialement dans le cacao qu'un moyen d'échange pour
           obtenir de l'or.</p>
        <h2>La fabrication</h2>
        <h3>La matière première: le chocolat</h3>
        <p>Le chocolat est préparé à partir de fèves de cacao qui se
           trouvent dans la cabosse, fruit du cacaoyer cultivé
           principalement dans les pays équatoriaux.<br/>
       Après la récolte, les fèves sont séchées au soleil puis
           expédiées aux chocolateries pour subir les
           transformations nécessaires.</p>
        <p>Les fèves sont torréfiées et broyées pour obtenir la
           poudre de cacao. Elles sont aussi pressées pour former le
            beurre de cacao.</p>
        <p>Le chocolat est obtenu par le mélange de la poudre et du
           beurre de cacao et lajout de lait et de sucre.</p>
        <p>La proportion de chaque élément déterminera la couleur du
           chocolat:<br/>
       le chocolat noir ou fondant peut contenir jusqu'à 70% de
```

```
            cacao,<br/>
        le chocolat au lait contient une part importante de poudre de
            lait,<br/>
        le chocolat blanc ne gardera du cacao que son beurre ajouté
            au sucre et lait en poudre.</p>
        <h2>Les vertus du chocolat</h2>
        <h3>Energétique</h3>
        <p>Les qualités nutritives et énergétiques du chocolat sont
            fort appréciées, déjà par les Espagnols au temps de la
            colonisation de l'Amérique mais aussi par les sportifs
            pendant l'effort.<br/>
        Ses constituants (magnésium , fer, lipides, glucides...) font
            du chocolat un excellent reconstituant.</p>
        <h3>Aphrodisiaque</h3>
        <p>Depuis le temps des Aztèques et des Mayas qui ont initié
            la consommation du cacao, le chocolat a toujours été
            considéré comme un excitant sexuel. L'Empereur aztèque
            Moctezuma lui-même en buvait "pour avoir accès aux femmes
            " . Le théologien Fransiscus Rauch écrit en 1624: "Ce
            breuvage bu dans les couvents y enflamme les passions".</
            p>
        <h3>Antidépresseur</h3>
        <p>Au seizième siècle, les dames de la noblesse espagnole en
            consommaient jusque dans les églises. Cela les aidaient-
            elles à supporter la longueur et la monotonie des offices
            ?<br/>
        On sait aujourd'hui que le chocolat contient de la
            phényléthylamine qui aurait une action positive en cas de
            dépression nerveuse.</p>
    </body>
</html>
```

Nous pouvons voir le résultat visuel dans un navigateur à la page suivante.



3.3 L'IMPORTANCE DES MOTS

Au sein de nos paragraphes, certains mots sont parfois plus importants que d'autres et nous aimerions les faire ressortir. HTML nous propose différents moyens de mettre en valeur le texte de notre page.

3.3.1 La mise en valeur légère

Pour appliquer une mise en valeur légère à notre texte, nous devons utiliser la balise ****. Son utilisation est très simple : encadrons les mots à mettre en valeur avec ces balises et le tour est joué ! Utiliser la balise **** a pour conséquence de mettre le texte en italique. En fait, c'est le navigateur qui choisit comment afficher les mots. Nous lui disons que les mots sont assez importants et, pour faire ressortir cette information, il change l'apparence du texte en utilisant l'italique.

3.3.2 La mise en valeur lourde

Pour mettre un texte bien en valeur, nous utilisons la balise **** qui signifie "fort" ou "important". Là encore, le gras n'est qu'une conséquence, le navigateur a choisi d'afficher en gras les mots importants pour les faire ressortir davantage. La balise **** ne signifie pas "mettre en gras" mais "important". Nous pourrons décider plus tard, en CSS, d'afficher les mots "importants" d'une autre façon que le gras si nous le souhaitons.

3.3.3 Le marquage du texte

La balise **<mark>** permet de faire ressortir visuellement une portion de texte. L'extrait n'est pas forcément considéré comme important mais on veut qu'il se distingue bien du reste du texte. Cela peut être utile pour faire ressortir un texte pertinent après une recherche sur notre site par exemple. Par défaut, **<mark>** a pour effet de surligner le texte. Nous pourrons changer l'affichage en CSS (décider de surligner dans une autre couleur, d'encadrer le texte, etc.). C'est le même principe : elles indiquent le sens des mots et non pas comment ceux-ci doivent s'afficher.

Voici un exemple qui reprend l'ensemble des balises vues dans cette section :

Exemple 13. *L'importance des mots dans notre page*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Importance des mots</title>
6      </head>
7      <body>
8          <p>Nous pouvons constater que dans cette phrase certains mots
               <em>sont légèrment importants</em>.</p>
9          <p>Par contre, d'autres <strong>sont très importants</strong>
              .</p>
10         <p>Néanmoins quelques uns <mark>marque l'attention</mark>.</p
              >
11     </body>
12 </html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

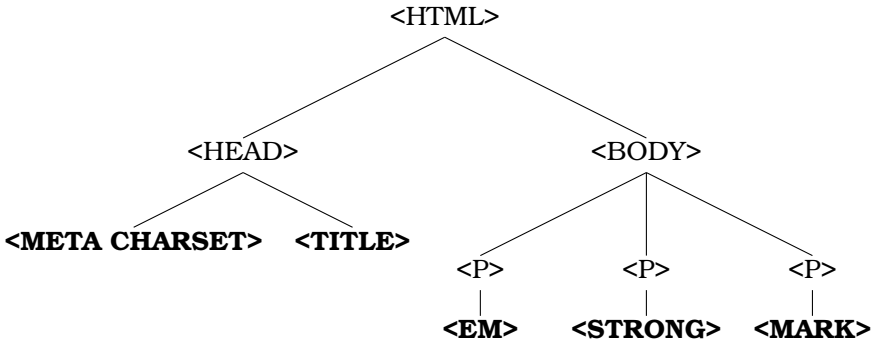
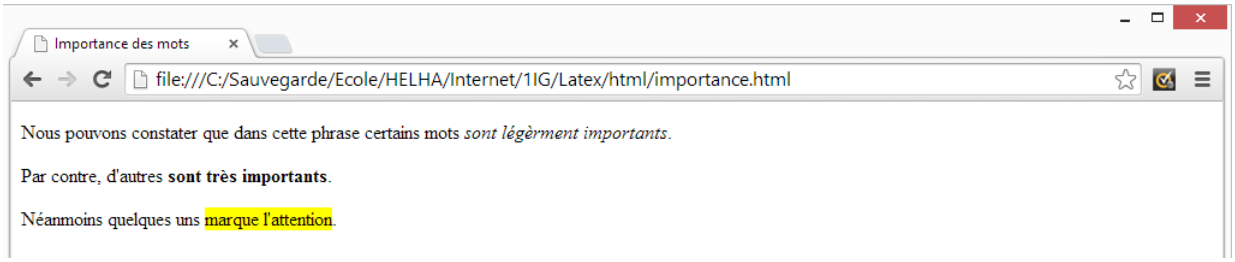


FIGURE 11 – Document HTML avec mise en valeur des mots sous forme d'arbre



3.4 LES LISTES

Les listes nous permettent souvent de mieux structurer notre texte et d'ordonner nos informations. Nous allons découvrir ici deux types de listes :

- les listes non ordonnées ou listes à puces ;
- les listes ordonnées ou listes numérotées ou encore énumérations.

3.4.1 Les listes non ordonnées

Une liste non ordonnée peut ressembler à ceci :

- 3 oeufs ;
- 100 g chocolat (noir ou au lait) ;
- 1 sachet de sucre vanillé.

C'est une structure qui nous permet de créer un ensemble d'éléments sans notion d'ordre (il n'y a pas de "premier" ni de "dernier"). Pour créer une liste non ordonnée, il suffit d'utiliser la balise **
** (unordered list).

Chacun des éléments de la liste est représenté par une balise **
** (list item).

Voici la structure qui correspond à notre exemple :

Exemple 14. *Les listes non ordonnées*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Liste non ordonnée</title>
    </head>
    <body>
        <p>
            <ul>
                <li>3 ufs;</li>
                <li>100 g chocolat (noir ou au lait);</li>
                <li>1 sachet de sucre vanillé.</li>
            </ul>
        </p>
    </body>
</html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

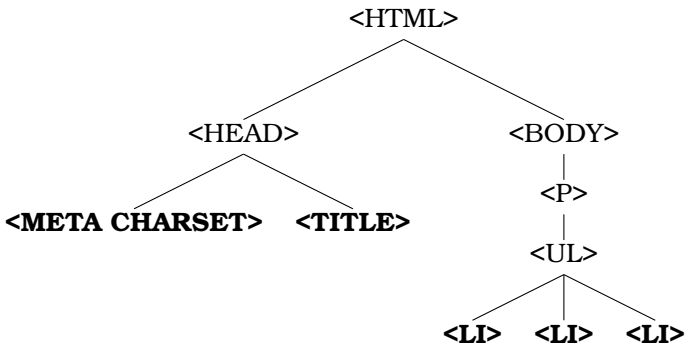
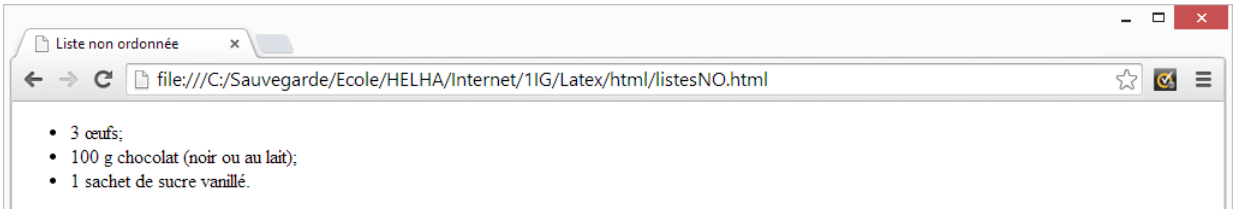


FIGURE 12 – Document HTML avec une liste non ordonnée sous forme d'arbre



Astuce 10 (Imbriquer des listes).

Pour les listes complexes, nous pouvons imbriquer ces listes à puces, c.-à-d. : créer une liste à puces dans une liste à puces. Pour cela, nous devons ouvrir une seconde balise **
** à l'intérieur d'un élément **
** de la première balise **<ul>**. Néanmoins, nous devons porter une attention particulière à l'ordre de fermeture de ces balises.

3.4.2 Les listes ordonnées

L'utilisation des listes ordonnées est similaire à celle des listes non ordonnées à la seule exception qu'il faut respecter un ordre déterminé pour que la structure ait du sens. Il nous suffit de remplacer la balise **
** par la balise **
** (ordered list).

Observons la liste de l'exemple suivant :

Exemple 15. *Les listes ordonnées*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Liste ordonnée</title>
    </head>
    <body>
        <p>
            <ol>
                <li>Faire ramollir le chocolat dans une terrine;</
                    li>
                <li>Séparer le blanc et le jaune d'uf;</li>
                <li>Incorporer les jaunes et le sucre;</li>
                <li>Battre les blancs en neige ferme et les ajouter
                    délicatement au mélange à l'aide d'une spatule
                    ;</li>
                <li>Mettre au frais de 1 à 2 heures minimum.</li>
            </ol>
        </p>
    </body>
</html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

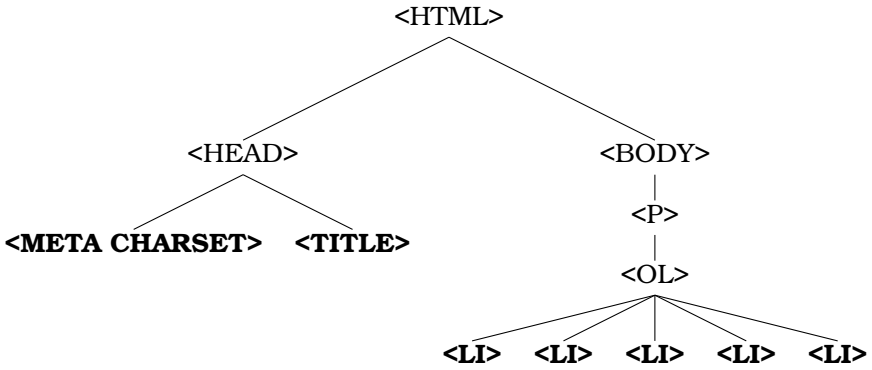


FIGURE 13 – Document HTML avec une liste ordonnée sous forme d'arbre

Nous pouvons avoir dans un document plusieurs types de liste, prenons les listes de nos deux derniers exemples et combinons les :

Exemple 16. *La combinaison de listes*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Listes</title>
    </head>
    <body>
        <h1>La recette de la mousse au chocolat</h1>
        <h2>Les ingrédients</h2>
        <p>
            <ul>
                <li>3 ufs;</li>
                <li>100 g chocolat (noir ou au lait);</li>
                <li>1 sachet de sucre vanillé.</li>
            </ul>
        </p>
        <h2>La procédure</h2>
        <p>
```

```
19                <ol>
20                    <li>Faire ramollir le chocolat dans une terrine;</
                          li>
21                    <li>Séparer le blanc et le jaune d'uf;</li>
22                    <li>Incorporer les jaunes et le sucre;</li>
23                    <li>Battre les blancs en neige ferme et les ajouter
                          délicatement au mélange à l'aide d'une spatule
                          ;</li>
24                    <li>Mettre au frais de 1 à 2 heures minimum.</li>
25                </ol>
26            </p>
27        </body>
28    </html>
```

Ce qui donne dans notre arbre et de manière visuelle dans un navigateur :

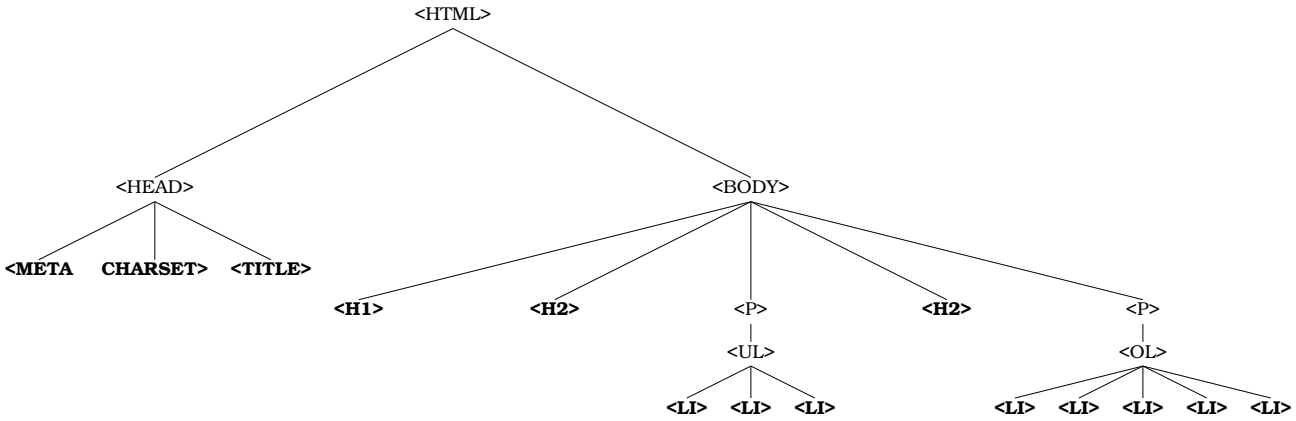
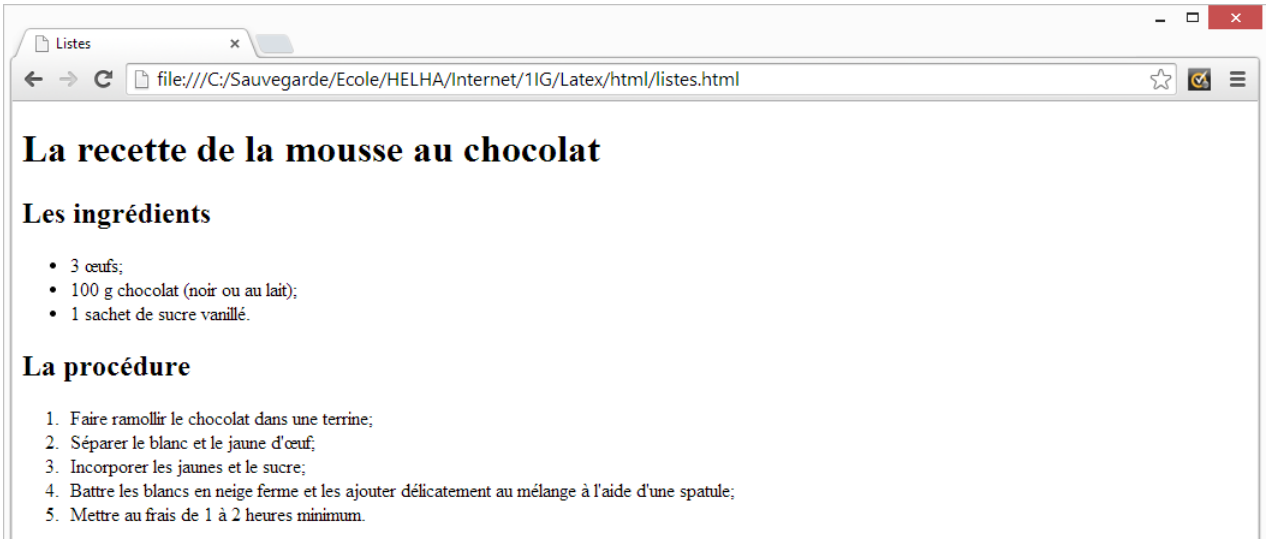


FIGURE 14 – Document HTML avec différents types de liste sous forme d'arbre

Astuce 11 (La combinaison de listes différentes imbriquées)**.**

Nous pouvons étendre nos combinaisons avec nos différents types de liste.



3.4.3 Les listes de définitions

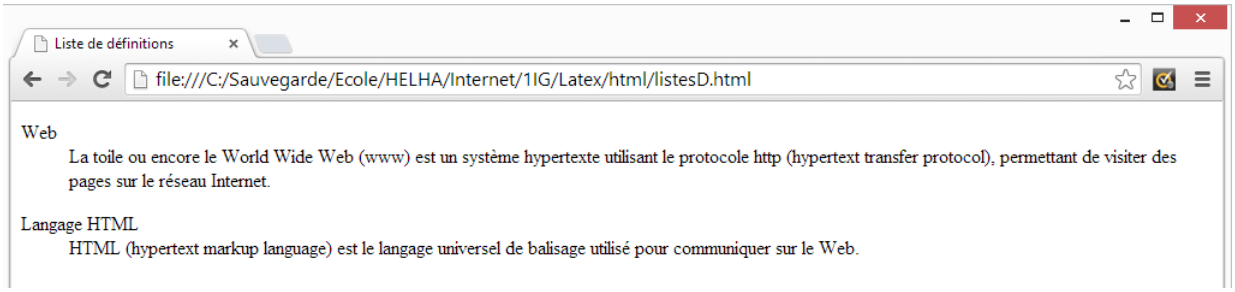
Les listes de définitions comportent deux parties, un terme et une description. Pour marquer une liste de définitions, il vous faut trois éléments HTML : un **<dl>** conteneur, un terme de définition **<dt>**, et une description de définition **<dd>**.

Exemple 17. *Les listes de définitions*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Liste de définitions</title>
    </head>
    <body>
        <p>
            <dl>
                <dt>Web</dt>
                <dd>La toile ou encore le World Wide Web (www) est un
                    système hypertexte utilisant le protocole http (
                    hypertext transfer protocol), permettant de
                    visiter des pages sur le réseau Internet.</dd>
            </dl>
            <dl>
                <dt>Langage HTML</dt>
                <dd>HTML (hypertext markup language) est le langage
                    universel de balisage utilisé pour communiquer
                    sur le Web.</dd>
```

```
16                </dl>
17            </p>
18        </body>
19    </html>
```

Ce qui donne de manière visuelle dans un navigateur :



Astuce 12 (Extension des listes de définitions)**.**

Nous pouvons utiliser plusieurs **<dt>** et **<dd>** dans la même liste de définitions. Nous pouvons aussi utiliser des éléments de type bloc dans une description de définition, comme un **<p>** ou un **<ul>**. Nous ne pouvons pas utiliser d'éléments de type bloc dans un terme de définition.

3.5 Les autres balises de structure

3.5.1 Les citations

La balise **<blockquote>** définit une citation. Elle spécifie une section qui est citée par une autre source via l'attribut cite. Notons que pour des citations courtes la balise **<q>** est conseillée.

3.5.2 Les adresses

La balise **<address>** définit les informations de l'auteur ou du propriétaire d'un document.

La balise **<address>** permet de mettre en évidence des adresses, coordonnées et URL. Elles sont généralement affichées dans un paragraphe spécifique et en italique (mise en page modifiable en CSS). La représentation peut varier en fonction du navigateur. Si

l'élément **<address>** est à l'intérieur de l'élément **<body>**, il représente les informations de contact pour le document.

Si l'élément **<address>** est dans un élément **<article>**(voir partie 3), alors il représente les coordonnées de l'auteur/propriétaire de cet article. Habituellement l'élément **<address>** est ajoutée en en-tête ou bas d'une page Web.

Nous pouvons utiliser la balise **<address>** en l'incluant, avec d'autres informations, dans un élément **<footer>** (voir partie 3).

3.5.3 L'indice et l'exposant

Les balises **** et **** placent le texte respectivement en indice et en exposant.

3.5.4 Le préformatage du texte

Les navigateurs ne reconnaissent qu'un espace entre les mots, ce qui peut se révéler gênant dans certaines situations. La balise **<pre></pre>** affiche un texte dit préformaté. Le navigateur prend ainsi en compte tous les espaces et sauts de ligne définis à l'écran. Avant que les tableaux ne soient reconnus par les navigateurs, les braves pionniers du HTML devaient employer cette balise pour faire des tableaux.

Prenons un exemple pour illustrer ces nouvelles balises :

Exemple 18. *Les autres balises de structure*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Les autres balises de structure</title>
6      </head>
7      <body>
8          <h1>Les autres balises de structure</h1>
9          <h2>La citation</h2>
10         <p>Une des citations de Socrate:<blockquote>La seule chose
               que je sais, c'est que je ne sais rien.</blockquote></p>
11         <h2>L'adresse</h2>
12         <p><address>Innovation
13                     rue Neuve,1
14                     1000 Bruxelles</address></p>
15         <h2>L'indice et l'exposant</h2>
16         <p><sub>escalier</sub>escalier<sup>escalier</sup></p>
17         <h2>Le préformatage de texte</h2>
18         <p>Cette chaine n'est pas préformatée.</p>
19         <p><pre>Cette    chaine    est   préformatée.</pre></p>
20     </body>
21 </html>
```

Ce qui donne de manière visuelle dans un navigateur :

Nous sommes capables à présent de construire une page HTML et de structurer son contenu. Un site web est composé de plusieurs pages HTML et donc nous devons les lier afin de pouvoir naviguer à travers notre site. C'est pourquoi nous abordons la notion de liens au chapitre suivant.

CHAPITRE 4

LES LIENS

HTML est un langage hypertexte (et hypergraphique) qui nous permet en cliquant sur un mot, généralement souligné, ou une image de nous transporter :

- vers un autre endroit du document ;
- vers un autre fichier HTML de notre site ;
- vers un autre ordinateur situé sur le Web.

Ce système d'hypertexte peut nous être familier car il est également utilisé par les fichiers d'aide de Windows. Ce sont ces liens qui nous permettent de surfer de page en page et qui constituent l'essence des documents HTML. La syntaxe de ces liens entre plusieurs pages est simple : ****.

Définition 6 (URL)**.**

Le sigle URL (de l'anglais Uniform Resource Locator, littéralement "localisateur uniforme de ressource"), auquel se substitue informellement le terme adresse web, désigne une chaîne de caractères utilisée pour adresser les ressources du World Wide Web : document HTML, image, son, forum Usenet [a], boîte aux lettres électroniques, entre autres. Les URL constituent un sous-ensemble des identifiants uniformisés de ressourcse [b].

a. Usenet est un système en réseau de forums, inventé en 1979. Pour fonctionner dans un environnement Unix, il utilise alors le protocole UUCP. Il devient accessible depuis Internet grâce à l'utilisation du protocole NNTP. Il est encore régulièrement utilisé au XXIe siècle.

b. Un URI, de l'anglais Uniform Resource Identifier, soit littéralement identifiant uniforme de ressource, est une courte chaîne de caractères identifiant une ressource sur un réseau (par exemple une ressource Web) physique ou abstraite, et dont la syntaxe respecte une norme d'Internet mise en place pour le World Wide Web.

4.1 LE CHEMIN ABSOLU ET LE CHEMIN RELATIF

Ces deux types de lien visent le même but : accéder à la ressource demandée. Cependant, leur syntaxe diverge quelque peu. En effet, d'une part nous avons le chemin absolu qui comme son nom l'indique mentionne l'adresse complète de la ressource sollicitée. D'autre part, le chemin relatif, quant à lui, renverra le chemin de la ressource demandée en fonction de sa position par rapport au répertoire courant.

Prenons un exemple concret pour illustrer ces deux notions :

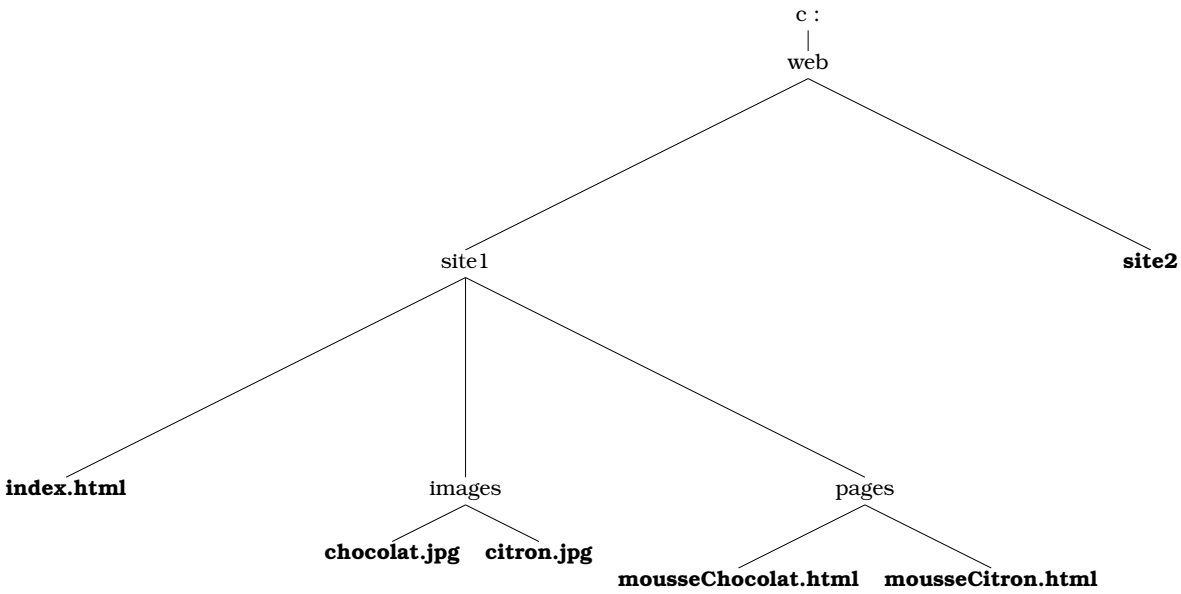


FIGURE 15 – Chemin absolu et chemin relatif

Soit l'arborescence de la figure 15, si notre position courante est le fichier index.html alors :

– le chemin absolu pour atteindre la ressource chocolat.jpg est : $c : \backslash web \backslash site1 \backslash images \backslash chocolat.jpg$

– le chemin relatif pour atteindre la ressource chocolat.jpg est : $images \backslash chocolat.jpg$

En gardant notre arborescence, si notre position courante est le fichier mousseCitron.html alors :

– le chemin absolu pour atteindre la ressource citron.jpg est : $c : \backslash web \backslash site1 \backslash images \backslash citron.jpg$

– le chemin relatif pour atteindre la ressource citron.jpg est : $.. \backslash images \backslash citron.jpg$

Nous pouvons remarquer que si la ressource se trouve dans le dossier site2, alors dans le cas d'un chemin relatif, il faudra remonter jusqu'au niveau du dossier web et puis descendre jusqu'à la ressource souhaitée.

Notons que si nous hébergeons notre site, nous ne connaissons pas l'arborescence de notre hébergeur. Dans cet optique, il sera toujours préférable de travailler avec un adressage relatif.

Astuce 13 (Séparateur de niveaux dans un chemin)**.**

Vu que nous ne pouvons pas connaître le système d'exploitation de notre serveur web (hébergeur) à l'avance, afin d'éviter toute erreur d'adressage, il est fortement conseillé d'utiliser le caractère "/" plutôt que le caractère "\". En effet, un serveur web fonctionnant sur un système d'exploitation Linux ne reconnaît pas le caractère "\" dans une adresse qu'elle soit absolue ou relative.

4.2 Les liens externes

Tout ordinateur situé sur le réseau Internet possède une adresse ou une URL. HTML permet d'accéder à toutes les machines et toutes les ressources du Net. Pour peu qu'Internet nous soit un peu familier, ce sont les adresses du type :

- http ://serveur/chemin.../fichier
- ftp ://serveur/chemin.../fichier

4.3 Les liens locaux

L'organisation classique, et plus que conseillée, d'un site Web consiste à regrouper l'ensemble des éléments de celui-ci (fichiers HTML, images, ...) dans un répertoire respectif par thématique. Nous pourrons ainsi "transporter" aisément notre site pour le présenter sur un autre ordinateur et, but ultime, le charger sur un serveur. Cette façon de procéder est la plus sûre et nous évitera pas mal de problèmes. Nous pouvons nous inspirer de la figure 15 pour mettre en place notre arborescence. De plus, dans le cadre de notre étude future du langage PHP et du modèle de conception MVC, cette habitude nous permettra de modifier plus facilement et plus rapidement notre code.

4.4 Les liens mixtes

Nous entendons par là un lien vers un fichier situé à un autre endroit de notre ordinateur (et donc non situé dans le répertoire de notre site).

Attention Danger ! En effet, il est peu probable que le serveur Web qui hébergera notre site possède la même arborescence que notre disque local. L'adresse prendra la forme :

- file :///lecteur :/répertoire/fichier.html (en adressage absolu) ;
- ../../../fichier.html (en adressage relatif).

4.5 Les ancres

Des liens peuvent aussi pointer vers un endroit précis du même document ou d'un autre fichier. C'est ce qu'on appelle les ancres, ancrages ou pointeurs [Anchor].

Voici les instructions à utiliser :

- **Point d'ancrage** : **** ou **** = Ceci est une cible
- **Lien vers une ancre dans la même page** : **** = Lien vers la cible *** dans la même page
- **Lien vers une ancre dans une autre page** : **** = Lien vers la cible *** dans une autre page

Plusieurs liens à l'intérieur d'un même document supposent que ce document présente une certaine longueur et donc un temps de chargement assez long. Ainsi, nous allons préférer généralement à cette technique le découpage d'un longue page en un ensemble de plusieurs pages de dimension plus réduite.

4.6 Les autres usages des liens

4.6.1 Le lien qui affiche une infobulle au survol

Nous pouvons utiliser l'attribut générique title qui affiche une bulle d'aide lorsqu'on pointe sur le lien. Cet attribut est facultatif. La bulle d'aide peut être utile pour informer le visiteur avant même qu'il n'ait cliqué sur le lien.

4.6.2 Le lien qui ouvre une nouvelle fenêtre

Il est possible de "forcer" l'ouverture d'un lien dans une nouvelle fenêtre. Pour cela, nous ajoutons l'attribut et la valeur **target="_blank"** à la balise **<a>**. Selon la configuration du navigateur, la page s'affichera dans une nouvelle fenêtre ou un nouvel onglet. Nous ne pouvons pas choisir entre l'ouverture d'une nouvelle fenêtre ou d'un nouvel onglet. Notons cependant qu'il est déconseillé d'abuser de cette technique car elle perturbe la navigation. Le visiteur lui-même peut décider s'il veut ouvrir le lien dans une nouvelle fenêtre. Il fera Maj + Clic sur le lien pour ouvrir dans une nouvelle fenêtre ou Ctrl + Clic pour ouvrir dans un nouvel onglet.

4.6.3 Le lien pour envoyer un e-mail

Si nous voulons que nos visiteurs puissent nous envoyer un e-mail, nous pouvons utiliser des liens de type mailto. Rien ne change au niveau de la balise, nous devons simplement modifier la valeur de l'attribut href. Il suffit donc de faire commencer le lien par mailto : et d'écrire l'adresse e-mail où on peut nous contacter. Si nous cliquons sur le lien, un nouveau message vide s'ouvre, prêt à être envoyé à notre adresse e-mail.

4.6.4 Le lien pour télécharger un fichier

Beaucoup d'entre nous se demandent comment cela se passe pour le téléchargement d'un fichier... En fait, il faut procéder exactement comme si nous faisions un lien vers une page web, mais en indiquant cette fois le nom du fichier à télécharger ! Par exemple, supposons que nous voulions faire télécharger monfichier.rar, plaçons simplement ce fichier dans le même dossier que notre page web (ou dans un sous-dossier) et faisons un lien vers ce fichier.

Voici un exemple qui reprend les 4 usages de liens précédemment vus :

Exemple 19. *Les autres usages des liens*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Les autres usages de liens</title>
6      </head>
7      <body>
8          <h1>Les autres usages de liens</h1>
9          <h2>Le lien avec infobulle</h2>
10         <p>Bonjour. Souhaitez-vous visiter le <a href="http://www.
               webacademy.be" title="Réservé à l'élite">Site Webacademy<
               /a>?</p>
11         <h2>Le lien d'ouverture dans une autre page ou un autre
               onglet</h2>
12         <p>Bonjour. Souhaitez-vous visiter le <a href="http://www.
               webacademy.be" target="_blank">Site Webacademy</a>?<br />
13 Le site s'affichera dans une autre fenêtre.</p>
14         <h2>Le lien d'envoi d'e-mail</h2>
15         <p><a href="mailto:votrenom@domaine.com">Envoyez-moi un e-
               mail!</a></p>
16         <h2>Le lien de téléchargement</h2>
17         <p><a href="monfichier.rar">Télécharger le fichier</a></p>
18     </body>
19 </html>
```

Ce qui donne de manière visuelle dans un navigateur :

Maintenant que nous pouvons surfer entre nos pages, nous allons à présent les illustrer grâce aux images dans le chapitre suivant.

CHAPITRE 5

LES IMAGES

La thématique des images est l'une des plus importantes lors de l'élaboration d'un site. Le texte de nos pages est chargé assez rapidement jusque maintenant, l'ajout des images risque de ralentir le chargement de nos pages. En effet, en fonction du format de l'image, nous pouvons avoir des temps de réponse complètement différents voire extrêmes.

Le chargement d'une page ne doit jamais excéder 3 secondes, après ce délai, le visiteur a de fortes chances de zapper notre site.

Le choix du type d'image à utiliser va dépendre de deux éléments :

- **le poids** : il va avoir une incidence sur la rapidité de chargement de notre page. Plus une image sera lourde, plus le chargement de la page sera long ;
- **la qualité** : en fonction de l'objectif à atteindre, le choix du format de l'image va affecté cette qualité.

Comme il vient d'être dit, c'est l'objectif qui va définir le type de l'image à utiliser. Nous allons parcourir dans les sections suivantes les formats d'images les plus utilisés pour leur(s) avantage(s).

5.1 LES DIFFÉRENTS FORMATS

Il n'est pas inutile de prévoir dans notre panoplie de compositeur Web, un logiciel de retouches d'images (voir annexes). Il est très facile de se faire une bibliothèque d'images tout en surfant sur le Web. La preuve est qu'il suffit de se positionner sur l'image, de cliquer avec le bouton droit de la souris et de suivre les instructions du menu déroulant (Enregis-trer l'image sous...). A cet instant, le format de l'image enregistré n'est peut-être pas celui qu'il nous faut. Vu que nous avons notre logiciel de retouches d'images, le tour est joué. En effet, ce dernier est capable entre autres de modifier le format de l'image.

Analysons à présent ces différents types de format.

FIGURE 16 – Exemples de montages d'images

5.1.1 Le format JPEG

Les images au format JPEG (Joint Photographic Expert Group) sont très répandues sur le Web. Ce format est conçu pour réduire le poids des photos (c'est-à-dire ; la taille du fichier associé), qui peuvent comporter plus de 16 millions de couleurs différentes.

Les images JPEG sont enregistrées avec l'extension .jpg ou .jpeg.

Notons que le JPEG détériore un peu la qualité de l'image, d'une façon généralement imperceptible. C'est ce qui le rend si efficace pour réduire le poids des photos.

5.1.2 Le format PNG

Le format PNG (Portable Network Graphics) est le plus récent de tous.

Le PNG a deux avantages importants :

– il peut être rendu transparent ;

– il n'altère pas la qualité de l'image.

Le PNG a été inventé pour concurrencer un autre format, le GIF, à l'époque où il fallait payer des royalties pour pouvoir utiliser des GIF. Depuis, le PNG a bien évolué et est devenu le format le plus puissant pour enregistrer la plupart des images.

Le PNG existe en deux versions, en fonction du nombre de couleurs que doit comporter l'image :

– **PNG 8 bits** : 256 couleurs ;

– **PNG 24 bits** : 16 millions de couleurs (autant qu'une image JPEG).

Astuce 14 (JPEG ou PNG ?)**.**

La compression du JPEG est plus puissante sur les photos. Une photo enregistrée en JPEG se chargera toujours beaucoup plus vite que si elle était enregistrée en PNG. Il est conseillé donc toujours de réserver le format JPEG aux photos.

5.1.3 Le format GIF

C'est un format assez vieux, qui a été néanmoins très utilisé (et qui reste très utilisé par habitude). Aujourd'hui, le PNG est globalement bien meilleur que le GIF. Le format GIF est limité à 256 couleurs (alors que le PNG peut aller jusqu'à plusieurs millions de couleurs). Néanmoins, le GIF conserve un certain avantage que le PNG n'a pas : il peut être animé.

FIGURE 17 – Exemples d'images au format GIF

5.1.4 Les autres formats

Il existe deux autres formats : BMP et TIF qui offre une qualité d'image incomparable mais nécessite un poids trop volumineux. Ils ne permettent pas de charger rapidement la page.

Astuce 15 (Les miniatures).

Si nous devons tout de même afficher des images en BMP ou en TIF, il nous est possible d'utiliser la technique de la miniature. Cette technique consiste à afficher l'image dans un format léger de taille réduite et, une fois sélectionnée, l'afficher dans un format plus lourd. De cette manière, les deux objectifs sont atteints : la page est chargé rapidement et le visiteur peut contempler la qualité de son image. La règle des 3 secondes n'est pas d'application puisque c'est le visiteur lui-même qui consacre du temps pour bénéficier d'une image de meilleure qualité.

5.2 LA BALISE <IMG>

La syntaxe de cette balise est la suivante : ****. L'attribut **src** a la même fonction que l'attribut **href** de la balise **<a>**. La balise image possède de nombreux attributs, dont :

- **Le texte alternatif alt="****"** : pour les navigateurs n'ayant pas ou plus l'option "image" activée ou s'il y a un problème de chargement.
- **Les dimensions width=? et height=?** : hauteur et largeur (en pixels/en %). Elles seront plutôt définies avec une CSS.
- **La bordure border=?** (en pixels/en %) : bordure qui entoure l'image. Comme pour les dimensions, elle sera définie en CSS.

Il est possible également d'utiliser l'attribut title.

Prenons en considérations les remarques suivantes :

- En HTML, l'image ne fait pas partie de notre document car le navigateur va la chercher à l'adresse indiquée (utilisation d'un lien) ;
- Presque en conséquence logique du premier point, le fait d'utiliser la même image à plusieurs reprises dans un fichier HTML ne modifie en rien sa taille ;
- Prévoir un texte pour les navigateurs n'ayant pas ou plus l'option image activée, permet au lecteur de ne pas perdre le fil de l'exposé et peut-être d'activer cette option pour découvrir votre oeuvre. A l'heure actuelle, le problème viendrait d'un problème d'adressage ;
- Il est important pour la fluidité de l'affichage de préciser la taille en hauteur et largeur de l'image car le navigateur peut ainsi, connaissant l'emplacement à réserver pour celle-ci, continuer à afficher le texte.
- Pour le nom de l'image, il faut éviter les accentués et les espaces. De plus, il faudra faire attention à la casse. En effet, un serveur web fonctionnant sur un système d'exploitation Linux respectera la casse.

5.3 Les fonctionnalités dérivées

Nous pouvons pousser notre étude des images à des concepts plus évolués qui, selon le cas, peuvent être utiles.

5.3.1 Les images cliquables

Les balises sont : **** Notons que les images avec liens sont entourées d'une bordure. Dans notre cas l'entièreté de l'image est cliquable, nous pouvons délimiter une zone de l'image. C'est ce que nous allons voir à présent.

5.3.2 Les images cliquables composées

Les images cliquables évoluées sont divisées en zones sur lesquelles il est possible de cliquer et d'ouvrir une page située à une URL déterminée. On appelle cette technique aussi l'hypergraphique par rapport à l'hypertexte.

Les balises sont :

- **pour l'image** : **** ;
- **pour les zones cliquables** : **<map name="nom"><area shape="rect" coords="x1, y1, x2, y2" href="URL"/><area**...**></map>**. Les formes possibles sont rect, circle et polygon. En fonction de la forme choisie, la valeur de l'attribut coords peut varier.

5.3.3 Les figures

La balise **<figure></figure>** a un rôle avant tout sémantique. Le but est d'indiquer à l'ordinateur que l'image a de l'importance.

A l'intérieur de cette balise, nous pouvons inclure au moins une balise **<img/>** et au plus une balise **<figcaption><figcaption/>**. Cette dernière balise permet de commenter l'image afin d'être éventuellement répertoriée : c'est le principe d'une figure.

Astuce 16 (Placer une image dans une balise **<p>** ou **<figure>**?)**.**

Il est vrai que la balise **<figure>** peut remplacer la balise **<p>** pour une figure. Donc tout dépend de ce que votre image apporte au texte :

- si elle n'apporte aucune information (c'est juste une illustration pour décorer) : plaçons l'image dans un paragraphe ;
- si elle apporte une information : plaçons l'image dans une figure (comme dans l'exemple 20 aux lignes 14 à 19).

Voici un exemple reprenant diverses images :

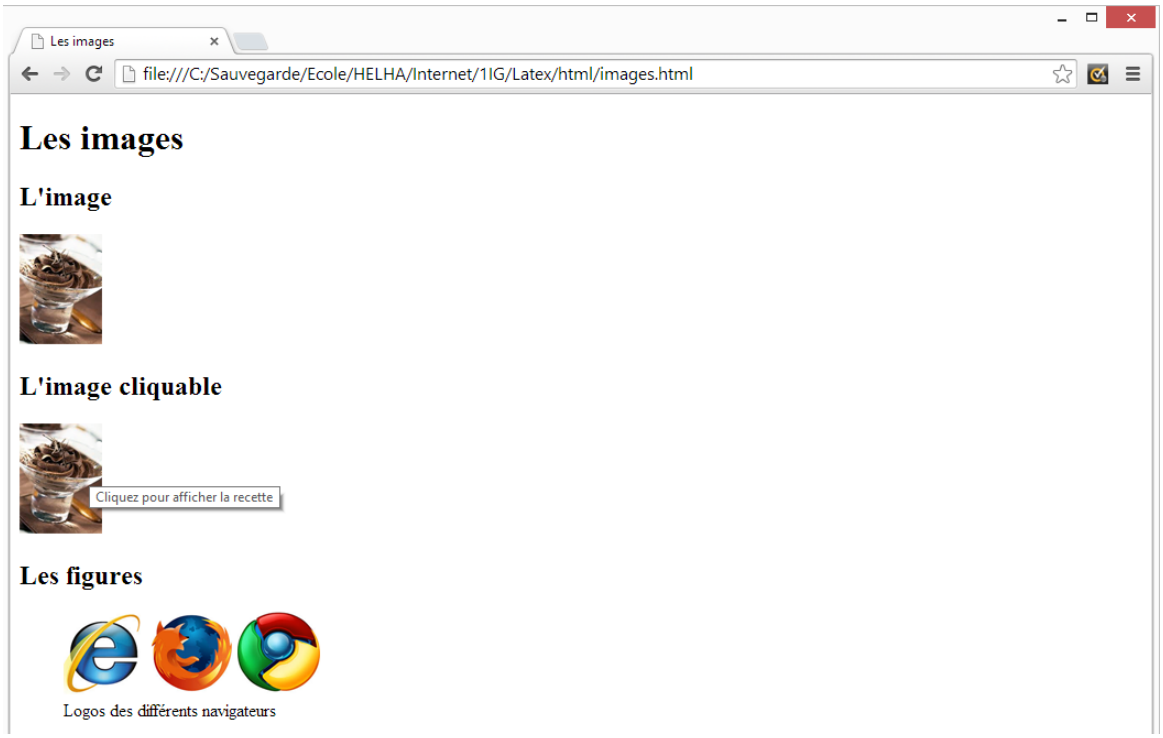
Exemple 20. *Les images*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Les images</title>
    </head>
    <body>
        <h1>Les images</h1>
        <h2>L'image</h2>
        <p><img src="../images/mousseChocolat.png" alt="La mousse au
            chocolat"/></p>
        <h2>L'image cliquable</h2>
        <p><a href="listes.html"> <img src="../images/mousseChocolat.
            png" alt="La mousse au chocolat" title="Cliquez pour
            afficher la recette"/></a></p>
        <h2>Les figures</h2>
        <figure>
        <img src="../images/logoIE.png" alt="Logo Internet Explorer"/
            >
        <img src="../images/logoFF.png" alt="Logo Mozilla Firefox"/>
        <img src="../images/logoGC.png" alt="Logo Google Chrome"/>
        <figcaption>Logos des différents navigateurs</figcaption>
        </figure>
    </body>
</html>
```

Nous pouvons observer dans l'exemple 20 que la balise **<figure>** est bien un conteneur est peut remplacer la balise **<p>** pour le traitement des images. La balise **figure** est l'une des nouvelles balises offertes par HTML 5. Il faudra donc implémenter des solutions alternatives pour les navigateurs qui les supportent pas.

Nous constatons que les dimensions des images n'ont pas été renseignées. En effet, nous laisserons ce rôle aux feuilles de style. L'idée maitresse est de **garder le HTML pour le fond et le CSS pour la forme**.

L'exemple 20 nous donne de manière visuelle dans un navigateur :



Nous venons d'illustrer nos pages grâce aux images et nous avons parcouru la première partie du cours, nous allons à présent donner du style à nos pages ! Pour parvenir à nos fins, nous allons utiliser les feuilles de style qui sont abordées dans la partie qui suit. Nous apporterons des éléments d'information complémentaires sur les anciennes balises qui ont été remplacées par les feuilles de style.

Deuxième partie

La mise en forme avec le CSS 3

CHAPITRE 6

LA MISE EN PLACE

A l'heure actuelle, nous ne pouvons pas concevoir des sites web sans utiliser les CSS pour les rendre "sexy".

Définition 7 (CSS)**.**

CSS est l'acronyme de Cascading Style Sheets, ou les "feuilles de styles en cascade", elles permettent de définir la présentation (la forme) des pages Web.

Le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte...). Ce langage est venu complèter le HTML en 1996.

6.1 L'ORIGINE DES FEUILLES DE STYLE

Les versions de CSS :

- **CSS 1** : dès 1996, nous disposons de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.
- **CSS 2** : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS ajoute de nombreuses options. nous pouvons désormais utiliser des techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.
- **CSS 3** : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc.

Astuce 17 (Langage de substitution).

Grâce à ses évolutions, le CSS a remplacé les veilles structures de mises en page et tente à rivaliser avec le javascript sur certains points.

6.2 LES MODES DE DÉCLARATION

6.2.1 L'ancienne méthode

Au début de l'histoire du Web, lorsque nous souhaitions modifier le type de caractère d'une portion de texte, nous devions utiliser la balise **<font>** en écrivant par exemple **<font face="TIMES">**. Si nous souhaitions définir la couleur de l'arrière-plan et la bordure d'un tableau (que nous verrons dans la partie 4) ainsi que le retrait (**padding**) de chacune de ses colonnes, nous aurions défini notre tableau par le code suivant : **<table border="1" bgcolor="silver" cellpadding="3" cellspacing="0">**.

Cette approche présentait cependant de nombreux inconvénients :
- Nos styles étant définis à l'intérieur même de notre contenu, ils étaient fastidieux à définir ;
- Nous devions répéter la meme définition de style à chaque fois que nous souhaitions l'utiliser dans notre page où dans les diverses pages de notre site ;
- A chaque fois que nous souhaitions modifier ce style, nous devions reparcourir le contenu de toutes nos pages pour mettre à jour, une par une, les diverses définitions qu'elles contenaient ;
- Ces répétitions augmentaient la taille de nos pages ;
- Nos styles étaient figés, les contenus se présentaient de la même façon pour tous les usages possibles : écran, imprimante, PDA, lecteurs vocaux, etc...

6.2.2 L'attribut style

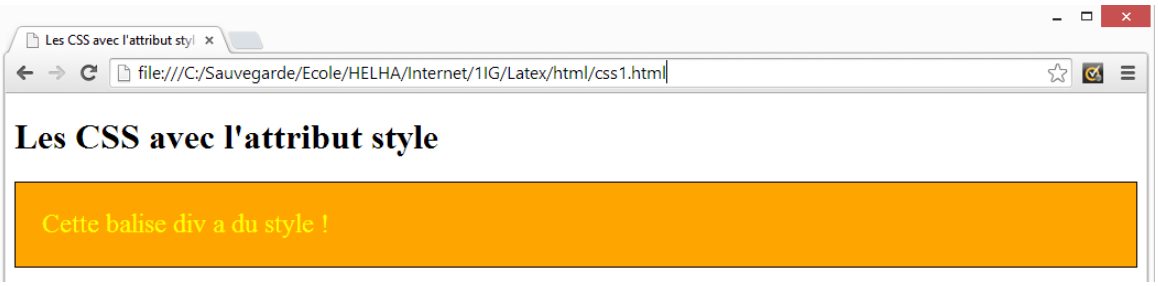
Nous pouvons définir des styles CSS directement dans la définition de l'attribut **style** d'une balise HTML. Dans l'exemple ci-dessous, nous utilisons une balise ** ** qui permet de définir une "boîte" à l'intérieur d'un contenu (nous analyserons cette balise de plus près dans la sous-section 6.3.

Exemple 21. *CSS défini dans la balise*

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="utf-8" />
5         <title>Les CSS avec l'attribut style</title>
```

```
    </head>
    <body>
        <h1>Les CSS avec l'attribut style</h1>
        <div style="background-color:orange; border:1px solid black;
            color:yellow; font-size:150%;padding:1em;">
        Cette balise div a du style !
        </div>
    </body>
</html>
```

Ce qui nous donne visuellement :



Nous pouvons constater que la valeur de l'attribut **style** est un ensemble de propriétés séparées par un caractère **;**. Ce caractère est très important car il représente la fin de la propriété.

6.2.3 La balise <style>

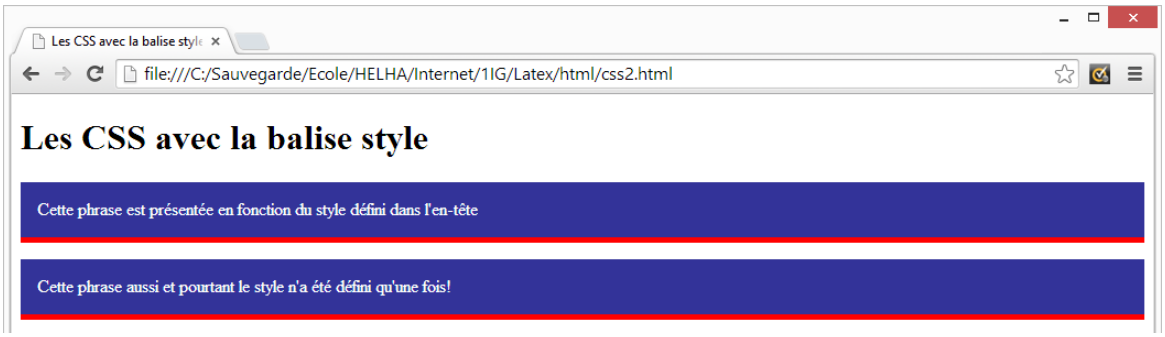
Plutôt qu'utiliser l'attribut style, il est préférable de définir nos styles CSS une fois pour toute dans une section particulière de notre page Web : la balise **<head>**.

Exemple 22. *CSS défini dans l'en-tête*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Les CSS avec la balise style</title>
        <style type="text/css">
            div
            {
                background-color:#339;
                color:#fff;
                padding:15px;
                border-bottom:5px solid red;
                margin-bottom:15px;
```

```
14                }
15            </style>
16        </head>
17        <body>
18            <h1>Les CSS avec la balise style</h1>
19            <div>
20            Cette phrase est présentée en fonction du style défini dans l
                  'en-tête
21            </div>
22            <div>
23            Cette phrase aussi et pourtant le style n'a été défini qu'une
                  fois!
24            </div>
25        </body>
26    </html>
```

Ce qui nous donne visuellement :



Grâce à cette nouvelle façon de procéder, nous n'avons besoin de définir notre style qu'une seule fois. Dans notre exemple, le style défini s'appliquera automatiquement à toutes les balises ** ** de la page. Avec cette méthode, nous pouvons appliquer le même style plusieurs fois dans la même page, mais pas à plusieurs pages d'un coup. Pour aller plus loin dans la standardisation de nos pages, nous devrons utiliser la méthode définie au point suivant.

6.2.4 La balise <link>

La façon idéale de définir les CSS consiste à les enregistrer dans un document indépendant de nos pages HTML. Grâce à cette méthode, toutes les pages qui font référence à cette feuille de style externe hériteront de toutes ses définitions. Un autre intérêt de cette méthode est de pouvoir définir plusieurs feuilles de styles pour le même contenu et de basculer d'une feuille à l'autre en fonction du support sur lequel le contenu est affiché (écran, imprimante, etc.). Nous reviendrons plus tard sur cet aspect. Une page

HTML peut faire référence à plusieurs feuilles de styles en même temps. Dans ce cas, les définitions contenues dans ces différentes feuilles seront combinées entre elles.

Voici un exemple de styles définis dans un document séparé :

Exemple 23. *CSS défini dans un fichier externe - 1ère façon*
Soit le fichier 'mes-styles.css' :

```
body
{
     background-color:#ccf;
     letter-spacing:1em;
}

p
{
     font-style:italic;
     font-family:times,sherif;
}
```

Soit le fichier html :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Les CSS avec la balise link</title>
        <link href="mes-styles.css" media="all" rel="stylesheet" type
            ="text/css" />
    </head>
    <body>
        <h1>Les CSS avec la balise link</h1>
        <p>Voici un exemple de paragraphe.</p>
        <p>Et voici un deuxième paragraphe.</p>
    </body>
</html>
```

Comme dans la méthode précédente (CSS dans l'en-tête de la page), le style n'a été défini qu'une seule fois mais peut être utilisé plusieurs fois. La différence entre cette méthode et la précédente, c'est que notre feuille de style peut être utilisée par un nombre illimité de pages. Il suffira d'ajouter la mention **<link href="mes-styles.css" media="all" rel="stylesheet" type="text/css" />** dans ces pages pour que notre feuille de style s'y applique.

Lorsque les utilisateurs du site chargeront une page, leur navigateur ira également lire la feuille de styles à laquelle cette page fait référence. Cette feuille de style sera gardée en mémoire par le navigateur et n'aura pas besoin d'être rechargée lors de la lecture des pages suivantes. Le résultat est un gain de temps de chargement global et une économie de bande passante pour le serveur de notre site.

Cette méthode permet également de mettre en place plusieurs feuilles de styles destinées aux différents media (imprimante, navigateurs de PDA, etc.). Nous pouvons en effet souhaiter de mettre en place une présentation particulière (sans les menus, par exemple) destinée à l'impression de nos documents.

Voici une liste des valeurs les plus couramment utilisées pour la balise **link** :

- **<link href="general.css" rel="stylesheet" type="text/css" media="all"/>**
  Remplaçons **general.css** par le nom que nous souhaitons donner à notre feuille de style. Cette définition nous permettra de mettre en place une feuille de style commune à **tous les medias**.
- **<link href="ecran.css" rel="stylesheet" type="text/css" media="screen, projection"/>**
  Remplaçons ecran.css par le nom que nous souhaitons donner à notre feuille de style. Cette définition nous permettra de mettre en place une feuille de style destinée **aux écrans**.
- **<link href="mobile.css" rel="stylesheet" type="text/css" media="handheld"/>**
  Remplaçons mobile.css par le nom que nous souhaitons donner à notre feuille de style. Cette définition nous permettra de mettre en place une feuille de style destinée **aux PDA et téléphones mobiles**.
- **<link href="impression.css" rel="stylesheet" type="text/css" media="print"/>**
  Remplaçons impression.css par le nom que nous souhaitons donner à notre feuille de style. Cette définition nous permettra de mettre en place une feuille de style destinée **aux imprimantes**.

Précisons enfin que la déclaration de la balise **<link ... />** n'est pas la seule façon de faire appel à une feuille de style séparée. Nous pouvons également utiliser la formulation suivante :

Exemple 24. *CSS défini dans un fichier externe - 2ème façon*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Les CSS avec la balise link</title>
6          <style type="text/css">
7                  @import url(mes-styles.css) all;
8          </style>
9      </head>
10     <body>
11         <h1>Les CSS avec la balise link</h1>
12         <p>Voici un exemple de paragraphe.</p>
13         <p>Et voici un deuxième paragraphe.</p>
14     </body>
15 </html>
```

Nous devrons remplacer **"mes-styles.css"** par le nom que nous souhaitons donner à notre feuille de style. Nous pouvons également remplacer **"all"** par le type de média auquel est destiné notre feuille de style.

Le résultat sera exactement le même que si nous avions déclaré **<link href="mes-styles.css" rel="stylesheet" type="text/css" media="all"/>**, à une nuance près : la règle **@import** permet d'importer une feuille de styles à l'intérieur d'une autre feuille de style (même syntaxe sans préciser le type de media).

Il y a de multiples avantages à séparer les feuilles de styles du contenu :

- La réduction de la taille des pages : Les définitions de style ne sont faites qu'une seule fois, même si elles sont utilisées plusieurs fois ;
- La réduction des temps de connexion : Les navigateurs garderont en mémoire (en cache) le contenu de la feuille de style CSS qui s'appliquera sur toutes les pages du site. Seuls les contenus des pages devront être chargés au cours de la navigation ;
- Une mise à jour plus facile : Nous n'aurons besoin que de changer la feuille de style pour mettre à jour la présentation de l'ensemble de notre site ;
- Scinder le travail de rédaction et le travail de présentation : Nous pouvons commencer à rédiger le contenu de nos pages sans nous soucier de leur présentation finale. Nous devons penser simplement à placer correctement nos balises sémantiques (titre, sous-titres, listes, etc.). Nous pourrons travailler notre mise en page et notre design plus tard.

6.3 Les balises universelles

Il arrivera parfois que nous ayons besoin d'appliquer un css à certains mots qui, à l'origine, ne sont pas entourés par des balises.

Nous ne pouvons donc en mettre que sur une balise. Si, par exemple, nous voulons modifier uniquement "bienvenue" dans le paragraphe suivant : ** **Bonjour et bienvenue sur mon site !** **.

Cela serait facile à faire s'il y avait une balise autour de "bienvenue" mais, malheureusement il n'y en a pas. Par chance, il existe des balises créées à cet effet.

En fait, il y a deux balises dites universelles, qui n'ont aucune signification particulière.

Il y a une différence minime mais significative entre ces deux balises :

- **** : c'est une balise de type **inline**, c'est-à-dire : une balise que l'on place au sein d'un paragraphe de texte, pour sélectionner certains mots uniquement. Les balises **** et **** sont de la même famille. Cette balise s'utilise donc au milieu d'un paragraphe et c'est celle dont nous allons nous servir pour colorer "bienvenue".
- ** ** : c'est une balise de type **block**, qui entoure un bloc de texte. Les balises ** **, **<h1>**... sont de la même famille. Ces balises ont quelque chose en commun : elles créent un nouveau "bloc" dans la page et provoquent donc obligatoirement un retour à la ligne. ** ** est une balise fréquemment utilisée dans la construction d'un design, comme nous le verrons plus tard.

Pour le moment donc, nous allons utiliser plutôt la balise ****. Nous la mettons autour de "bienvenue" et nous lui ajoutons un style.

Pour la partie HTML :

```
1 <html>
2 ...
3 <p>Bonjour et <span>bienvenue</span> sur mon site !</p>
4 ...
5 </html>
```

Pour la partie CSS :

```
1 p
2 {
3   color: blue;
4 }
```

Il demeure cependant une amélioration à apporter : identifier ou classifier la balise **span** afin de la distinguer de ses consoeurs. C'est la piste mentionnée aux sections 6.5 et 6.6.

6.4 Les sélecteurs

Néanmoins, nous pouvons peaufiner la cible de nos styles avec les sélecteurs.

Définition 8 (Le sélecteur)**.**

Un sélecteur W3C représente une structure. Cette structure peut être vue par exemple comme une condition, notamment dans une règle CSS, qui détermine quels éléments de l'arbre documentaire sont sélectionnés par le sélecteur, ou comme une description plate du fragment HTML ou XML correspondant à cette structure. Les sélecteurs W3C vont de la simple représentation du nom d'un élément jusqu'à des représentations contextuelles complexes.

Quelques rçgles suffisent pour pouvoir séparer le contenu et sa présentation. L'essentiel est de comprendre comment établir le lien entre un style et la phrase à laquelle s'applique le style.

6.4.1 Les sélecteurs simples

Un sélecteur de type d'élément est le nom d'une balise de notre page web. Un sélecteur de type d'élément représente une occurrence d'un élément (balise) du type donné dans l'arbre du document.

Définition 9 (Les espaces de noms).

Les sélecteurs de type d'élément peuvent contenir un composant optionnel appelé espace de noms. Un préfixe d'espace de noms préalablement déclaré (via une règle : @namespace) peut être préfixé à un nom d'élément en les séparant par le séparateur d'espace de noms "|".

Le composant d'espace de noms peut être laissé vide pour indiquer que le sélecteur ne représente que des éléments sans espace de noms déclaré. De plus, une étoile peut être utilisée comme préfixe d'espace de noms, indiquant alors que le sélecteur représente des éléments dans quelques espaces de noms que ce soit (y compris les éléments sans espace de noms déclaré). Les sélecteurs de type d'élément qui n'ont pas de composant d'espace de noms (pas de séparateur d'espace de noms) représentent les éléments sans considération d'espace de noms (équivalent à "*|") sauf si un espace de noms par défaut a été déclaré, auquel cas le sélecteur représente les éléments dans cet espace de noms par défaut.

Il doit être noté que si un préfixe d'espace de noms utilisé dans un sélecteur n'a pas été préalablement déclaré, le sélecteur doit alors être considéré comme invalide et l'ensemble de la règle est alors ignorée en conformité avec les règles standard de gestion d'erreur.

En résumé :
- **ns|E** éléments E dans l'espace de nom ns
- ***|E** éléments E dans tout espace de noms, y compris ceux sans espace de noms déclaré
- **|E** éléments E sans espace de noms déclaré
- **E** si aucun espace de noms par défaut n'a été déclaré, ceci est équivalent à *|E. Sinon, c'est équivalent à ns|E où ns est l'espace de noms par défaut.

Exemple 25. *Exemples de sélecteurs simples avec noms d'espaces*

```
@namespace foo url(http://www.foo.com);
foo|h1 { color: blue; }
foo|* { color: yellow; }
|h1 { color: red; }
*|h1 { color: green; }
h1 { color: green; }
```

La première règle sélectionnera uniquement les éléments h1 dans l'espace de noms "http ://www.foo.com".

La seconde règle sélectionnera tous les éléments de l'espace de noms "http ://www.foo.com".

La troisième règle sélectionnera uniquement les éléments h1 elements sans déclaration d'espace de noms.

La quatrième règle sélectionnera les éléments h1 dans tout espace de noms (y compris ceux sans espace de noms déclaré).

La dernière règle est équivalente à la quatrième car aucun espace de noms par défaut n'a été défini.

Avec les CSS, nous pouvons donc changer la présentation de toutes les balises HTML standards. Il nous suffit d'identifier la balise et de faire figurer nos définitions comme suit :

Exemple 26. *La feuille de style avec les sélecteurs simples sans espace de noms*

Soit le code css suivant :

```
p
{
        font-weight:bold;
        line-height:1.3em;
}
```

Et le code HTML associé au CSS :

```
<html>
...
<p>Ce style va s'appliquer à moi car je suis un paragraphe.</p>
<div>Mais il ne s'appliquera pas à moi.</div>
...
</html>
```

Le code ci-dessus aura pour effet de mettre en gras et d'augmenter la hauteur de ligne de tous les paragraphes.

La syntaxe générale de définition est la suivante :
- Préciser le nom d'espaces éventuel ;
- Préciser le nom de la balise en premier avec le s ;
- Encadrer les définitions par des accolades ;
- Placer un point-virgule ; derrière chaque définition.

6.4.2 Les autres sélecteurs

Voyons sans tarder quelles sont les possibilités offertes :
- ***** : sélecteur universel ;
- **A B** : une balise B contenue dans une autre A ;
- **A + B** : une balise B qui en suit une autre A ;
- **A[attribut]** : une balise qui possède un attribut déterminé ;
- **A[attribut="Valeur"]** : une balise, un attribut et une valeur exacte. L'attribut doit en plus avoir la valeur exacte ;
- **A[attribut*="Valeur"]** : une balise, un attribut et une valeur. L'attribut doit cette fois contenir la valeur peu importe sa position ;
- ...

Il existe d'autres sélecteurs que nous pouvons retrouver sur le site du W3C : `http://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html`

6.5 LA CLASSIFICATION

Nous pouvons attribuer à chaque élément HTML une ou plusieurs classes. C'est nous qui définirons le nom de ces classes et qui déciderons de leurs styles. Les styles définis dans les classes remplaceront les styles "normaux" des éléments auxquels ils s'appliquent. Pour créer une classe, nous devons simplement faire figurer son nom précédé d'un point. Pour éviter toute ambiguïté, notre nom de classe ne doit pas comporter d'espace. Cette définition de style est à placer dans une feuille de styles ou dans la balise **<head>** de notre page.

Exemple 27. *Exemple de la classification*

Pour appliquer le style défini dans notre classe à un élément, ajoutons la mention ***class="nom-du style"*** *dans la définition de la balise. Cette façon de procéder est très pratique car elle permet d'appliquer les réglages de notre classe à de nombreux éléments, même s'ils ne sont pas du même type.*

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5      </head>
6      <style>
7          .mon-style
8          {
9          color:red;
10         }
11     </style>
12     <body>
13         <p class="mon-style">Le style s'applique à ce paragraphe</p>
14         <p>Mais pas à celui-là</p>
15
16         <p class="mon-style">Le style peut s'appliquer à ce
               paragraphe</p>
17         <div class="mon-style">Et aussi à cette balise!</div>
18     </body>
19 </html>
```

Chaque élément HTML peut avoir aucune, une ou plusieurs classes. Pour appliquer plusieurs classes au même élément, précisons simplement la liste de classes en séparant leurs noms par un espace.

Exemple 28. *Exemple de la classification avec plusieurs classes*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <style>
        .mon-style1
        {
        color:yellow;
        }
        .mon-style2
        {
        background-color:#A0A0A0;
        font-weight:bold;
        }
    </style>
    <body>
        <p class="mon-style1 mon-style2">Les styles des deux classes
           s'appliquent à ce paragraphe</p>
        <p class="mon-style2">Alors que ce paragraphe n'a qu'une
           seule classe </p>
    </body>
</html>
```

6.6 L'IDENTIFICATION

Les éléments HTML peuvent se voir attribuer un id (identification) ou, à la place, une classe.

Le principe de l'id est très similaire à celui de la classe à une exception près :

- Plusieurs éléments peuvent avoir la même classe ;
- Il ne doit y avoir qu'un seul élément ayant un id donné.

Nous pourrions parfaitement nous contenter d'utiliser les classes pour tous nos styles et oublier complètement l'existence des id.

Leur utilisation permet simplement de clarifier les choses et de mieux structurer nos pages :

- On utilise les classes pour définir l'aspect des mots, phrases et paragraphes ;
- On utilise les id pour placer des blocs dans la page (sachant qu'on n'a généralement jamais deux blocs placés exactement au même endroit).

Pour créer un id, nous devons simplement faire précéder son nom d'un dièse #. Pour éviter toute ambiguïté, notre nom d'id ne doit pas comporter d'espace. Cette définition de style est à placer dans une feuille de styles ou dans la balise **<head>** de votre page.

Exemple 29. *Exemple de l'identification Pour appliquer le style défini dans notre id à un élément, ajoutons la mention* **id="nom-du style"** *dans la définition de la balise.*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <style>
        #mon-style
        {
        color:red;
        }
    </style>
    <body>
        <p id="mon-style">Le style s'applique à ce paragraphe</p>
        <p>Mais pas à celui-là</p>
    </body>
</html>
```

6.7 L'ORDRE DE PRIORITÉ

6.7.1 Les styles standards, de classification et d'identification

Par défaut, chaque type de balise a une présentation particulière dans chaque navigateur. Si nous n'avons défini aucun style particulier pour la balise **<p>**, il est possible que le texte contenu dans ces balises ne se présente pas exactement de la même façon dans tous les navigateurs. Lorsque nous définissons nous-même le style d'une balise standard avec une définition telle que **p {blablabla}**, nous obligons tous les navigateurs à afficher notre texte de la même façon.

Si nous attribuons une classe à un paragraphe, les styles que nous aurons définis dans cette classe remplaceront également les styles standards du paragraphe et seront même prioritaires par rapport aux styles que nous aurons définis sous la forme **p {blablabla}**.

Si nous attribuons un id à un paragraphe, les styles définis dans l'id seront prioritaires par rapport à tous les autres styles pouvant s'appliquer à ce paragraphe.

Si nous définissons deux fois le même style dans 2 classes différentes et que les 2 classes s'appliquent à la même balise, c'est la dernière classe citée dans la déclaration **class="classe1 classe2"** qui sera prioritaire.

Si nous définissons 2 fois le même style pour la même balise standard, la même classe ou le même id, c'est la dernière définition qui sera prioritaire.

En résumé, voici l'ordre des priorités (ceci est valable pour toutes les balises HTML) :

1. Style standard défini par le navigateur

```
.mon-style1
{
color:brown;
font-weight:bold;
}
.mon-style2
{
color:green;
}
```

2. Style standard redéfini en CSS (la dernière définition est prioritaire)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <style>
        p
        {
        color:brown;
        }
        .mon-style
        {
        color:green;
        font-weight:bold;
        }
    </style>
    <body>
        <p class="mon-style">
        Ce paragraphe s'affiche en vert parce que la classe est
            prioritaire par rapport au style standard de la balise
        </p>
    </body>
</html>
```

3. Style de classe CSS (la dernière définition est prioritaire)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <style>
        .mon-style
```

```
        {
        color:brown;
        font-weight:bold;
        }
        .mon-style
        {
        color:green;
        }
    </style>
    <body>
        <p class="mon-style">
        Ce paragraphe s'affiche en vert parce que c'est la dernière
            (re)définition de style qui est prioritaire
        </p>
    </body>
</html>
```

4. Style d'id (la dernière définition est prioritaire)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <style>
        #mon-style1
        {
        color:green;
        }
        .mon-style2
        {
        color:brown;
        font-weight:bold;
        }
    </style>
    <body>
        <p id="mon-style1" class="mon-style2">
        Ce paragraphe s'affiche en vert parce que l'ID est
           prioritaire sur la classe
        </p>
    </body>
</html>
```

6.7.2 L'emboîtement des balises

Dans un cas d'emboîtement de balises tel que celui-ci :

```
<body>
<div>
<p>
bla bla bla
</p>
</div>
</body>
```

Les différents styles attribués aux balises **<body>**, ** ** et **<p>** vont se combiner pour définir quels sont les styles définitifs qui seront appliqués au texte "bla bla bla". Si le même style est défini, puis redéfini dans ces différentes balises (la couleur du texte, par exemple), c'est la dernière définition qui l'emportera sur les autres. Si un style particulier n'est défini que dans la balise **<body>**, il doit logiquement s'appliquer à l'ensemble du contenu de la balise **<body>**. En vérité, certains styles (tels que la couleur, par exemple) se transmettent automatiquement aux balises emboîtées, tandis que d'autres (tels que les marges, par exemple) ne se transmettent pas. Lorsqu'ils se transmettent, nous dirons que le contenu des balises ** ** et **<p>** aura "hérité" des propriétés de la balise **<body>**.

6.8 LES COMMENTAIRES

Comme en HTML, il est possible de mettre des commentaires. Les commentaires ne seront pas affichés, ils servent simplement à indiquer des informations pour nous, par exemple pour nous y retrouver dans un long fichier CSS. D'ailleurs, nous allons nous en rendre compte, en général le fichier HTML est assez court et la feuille CSS assez longue (si elle contient tous les éléments de style de notre site, c'est un peu normal). Notons qu'il est possible de créer plusieurs fichiers CSS pour notre site si nous ressentons le besoin de séparer un peu notre code CSS (en fonction des différentes sections de notre site, par exemple).

Donc, pour faire un commentaire, nous devons taper : /*, suivi de notre commentaire, puis */ pour terminer notre commentaire. Nos commentaires peuvent être écrits sur une ou plusieurs lignes. Par exemple :

```
@charset "utf-8";
/*style.css
  --------- */
p
{
color: blue; /* Les paragraphes seront en bleu */
}
```

CHAPITRE 7

LE FORMATAGE DU TEXTE

Nous avons déjà pu constater que le langage HTML calque les fonctionnalités de base d'un traitement de texte. L'histoire de l'HTML justifie cette constatation. En effet, les premiers sites Internet représentaient un sujet personnel à consonance diverse (profession, loisirs, référentiel historique, ...). Bref, l'aspect de ces sites étaient primordial, c'est pourquoi nous retrouvons des concepts communs à ceux d'un éditeur de textes.

7.1 LES UNITÉS DE MESURE

Les CSS offrent différentes unités pour exprimer la longueur. Certaines ont une histoire en typographie[3], comme le point (pt) et le pica (pc), d'autres sont connues pour leur usage quotidien, comme le centimètre (cm) et le pouce (in). Et il y a également une unité "magique" inventée spécialement pour CSS : le pixel px.

Les unités ne dépendent aucunement des propriétés mais du **média** de sortie : écran ou papier.

Il n'y a pas de restriction sur quelle unité peur être utilisée à tel ou tel endroit. Si une propriété accepte une valeur en px, elle accepte également une valeur en pouces ou en centimètres et vice-versa.

La relation entre **les unités absolues** est la suivante : 1in = 2.54cm = 25.4mm = 72pt = 6pc. Les unités appelées absolues (cm, mm, in, pt et pc) signifient la même chose en CSS que partout ailleurs. Une longueur exprimée dans l'une de ces unités apparaîtra avec cette taille précise (selon de la précision des composants matériels et du logiciel). Elles ne sont pas recommandées pour un écran, parce que les tailles d'écran varient beaucoup. Un grand écran peut avoir une largeur de 24in et un petit écran de portable une largeur de 8cm seulement. De plus, nous ne les regardons pas à la même distance.

3. (souvent abrégé en typo) désigne les différents procédés de composition et d'impression utilisant des caractères et des formes en relief, ainsi que l'art et la manière d'utiliser les différents types de caractères dans un but esthétique et pratique.

Les **unités em et ex** dépendent de la police de caractères et peuvent être différentes pour chaque élément du document. L'unité em est simplement la taille de la police de caractères. Pour un élément dont la police est de taille 2in, 1em signifie 2i. Exprimer des tailles, comme les margins (marges) et les paddings (espacements), en em signifie qu'elles sont relatives à la taille de la police, et si l'utilisateur a une large police de caractères (sur un grand écran par exemple) ou au contraire une petite police (sur un smartphone par exemple), les tailles seront en proportion.

Les **unités ex** sont rarement utilisées. Leur but est d'exprimer des tailles relatives à l'x-height de la police. Le x-height est, schématiquement, la hauteur des plus petites lettres comme le a, c, m, or o. Les polices qui ont la même taille (donc le même em) peuvent varier considérablement concernant la taille de leurs plus petites lettres, et lorsqu'il est important que certaines images par exemple aient le même x-height, l'unité ex est disponible.

L'**unité px** est l'unité magique en CSS. Elle n'est pas relative à la police courante ni relative aux unités absolues. L'unité px est définie pour être 'petite mais visible', et de telle façon qu'une ligne horizontale de 1px de largeur peut être affichée de faàon nette. Ce qui est net, petit et visible dépend du support et de la façon dont il est utilisé. Le px n'est donc pas défini comme une longueur constante, mais comme quelque chose qui dépend du type de matériel et de son usage courant.

Astuce 18 (Les prochaines unités)**.**

Pour rendre encore plus facile le fait de définir des règles CSS qui ne dépendent que de la taille par défaut de la police, une nouvelle unité est en développement : le rem. Le rem (pour "root em") est la taille de la police de l'élément racine du document. Contrairement au em, qui peut être différent pour chaque élément, le rem est constant dans tout le document.
D'autres unités en développement rendront possible de spécifier des tailles relatives à la fenêtre du lecteur. Ce sont les unités vw et vh. Le vw est égal à 1/100ème de la largeur de la fenêtre et le vh est égal à 1/100ème de la hauteur de la fenêtre.

7.2 LA TAILLE

Pour modifier la taille du texte, nous allons utiliser la propriété CSS **font-size**. Deux techniques nous permettent de fixer la taille :

- la **taille absolue** : en pixels, en centimètres ou millimètres. Cette méthode est très précise mais il est conseillé de ne l'utiliser que si c'est absolument nécessaire, car on risque d'indiquer une taille trop petite pour certains lecteurs.
- la **taille relative** : en pourcentage, "em" ou "ex", cette technique a l'avantage d'être plus souple. Elle s'adapte plus facilement aux préférences de taille des visiteurs.

7.2.1 La taille absolue

Elle est indiquée généralement par les pixels. Pour avoir un texte de 16 pixels de hauteur comme à la figure 18, nous devons donc écrire : **font-size** : **16px**;.

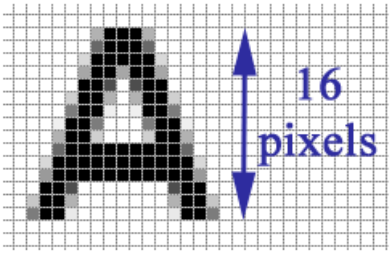


FIGURE 18 – Taille de 16 pixels

7.2.2 La taille relative

Elle adapte le texte aux préférences de tous les visiteur et donc elle est la méthode recommandée.

Il y a plusieurs moyens d'indiquer une valeur relative. Nous pouvons par exemple écrire la taille avec des mots en anglais comme ceux-ci :

- **xx-small** : minuscule ;
- **x-small** : très petit ;
- **small** : petit ;
- **medium** : moyen ;
- **large** : grand ;
- **x-large** : très grand ;
- **xx-large** : gigantesque.

Cette technique a un défaut : il n'y a que sept tailles disponibles (car il n'y a que sept noms). Heureusement, il existe d'autres moyens "em", "er" et les pourcentages. Nous pouvons interpréter cette technique de la manière suivante :

- Si nous écrivons 1em, le texte a une taille normale.
- Si nous voulons grossir le texte, nous pouvons inscrire une valeur supérieure à 1, comme 1.3em.
- Si nous voulons réduire le texte, inscrivons une valeur inférieure à 1, comme 0.8em.

7.3 LA POLICE

La notion de police est un sujet assez délicat dans les sites web. En effet, il se pose un problème : pour qu'une police s'affiche correctement, il faut que tous les internautes l'aient. Si un internaute n'a pas la même police que nous, son navigateur prendra une police par défaut (une police standard) qui peut être à l'antipode de ce que nous avions prévu.

Cependant, nous avons, depuis la version CSS 3, la possibilité de forcer le navigateur à télécharger automatiquement une police.

7.3.1 La personnalisation standard

La propriété CSS qui permet d'indiquer la police à utiliser est **font-family** :**"Times New Roman"**;.

Astuce 19 (Anticiper les polices installées chez l'internaute)**.**

Ppour éviter les problèmes si l'internaute n'a pas la même police que nous, nous précisons en général plusieurs noms de police, séparés par des virgules : **font-family** :**Impact, "Arial Black", Arial, Verdana, sans-serif**;.
Le navigateur essaiera d'abord d'utiliser la police Impact. S'il ne l'a pas, il essaiera la Arial Black. S'il ne l'a pas, il passera à la police Arial, et ainsi de suite. En général, nous indiquerons en tout dernier serif ou sans-serif, ce qui correspond à une police par défaut (qui ne s'applique que si aucune autre police n'a été trouvée).

Voici une liste de polices qui fonctionnent bien sur la plupart des navigateurs :
- Arial ;
- Arial Black ;
- Comic Sans MS ;
- Courier New ;
- Georgia ;
- Impact ;
- Times New Roman ;
- Trebuchet MS ;
- Verdana.

7.3.2 La personnalisation étendue

Pendant longtemps, cela n'était pas possible. Aujourd'hui, avec CSS 3, il existe heureusement un moyen d'utiliser n'importe quelle police sur son site et qui fonctionne bien avec la plupart des navigateurs.

Néanmoins, il y a des défauts :

- Il faudra que le navigateur de nos visiteurs télécharge automatiquement le fichier de la police, dont le poids peut atteindre, voire dépasser 1 Mo...
- La plupart des polices sont soumises au droit d'auteur, il n'est donc pas légal de les utiliser sur son site. Heureusement, il existe des sites comme `www.fontsquirrel.com` et `www.dafont.com` qui proposent en téléchargement un certain nombre de polices libres de droits.
- Il existe plusieurs formats de fichiers de polices et ceux-ci ne fonctionnent pas sur tous les navigateurs.

Voici les différents formats de fichiers de polices qui existent et qu'il faut connaître :

- **.ttf** : TrueType Font. Fonctionne sur IE9 et tous les autres navigateurs.
- **.eot** : Embedded OpenType. Fonctionne sur Internet Explorer uniquement, toutes versions. Ce format est propriétaire, produit par Microsoft.
- **.otf** : OpenType Font. Ne fonctionne pas sur Internet Explorer.
- **.svg** : SVG Font. Le seul format reconnu sur les iPhones et iPads pour le moment.
- **.woff** : Web Open Font Format. Nouveau format conçu pour le Web, qui fonctionne sur IE9 et tous les autres navigateurs.

En CSS, pour définir une nouvelle police et l'utiliser, nous devons déclarer :

Exemple 30. *Déclaration et utilisation d'une nouvelle police*
Le CSS :

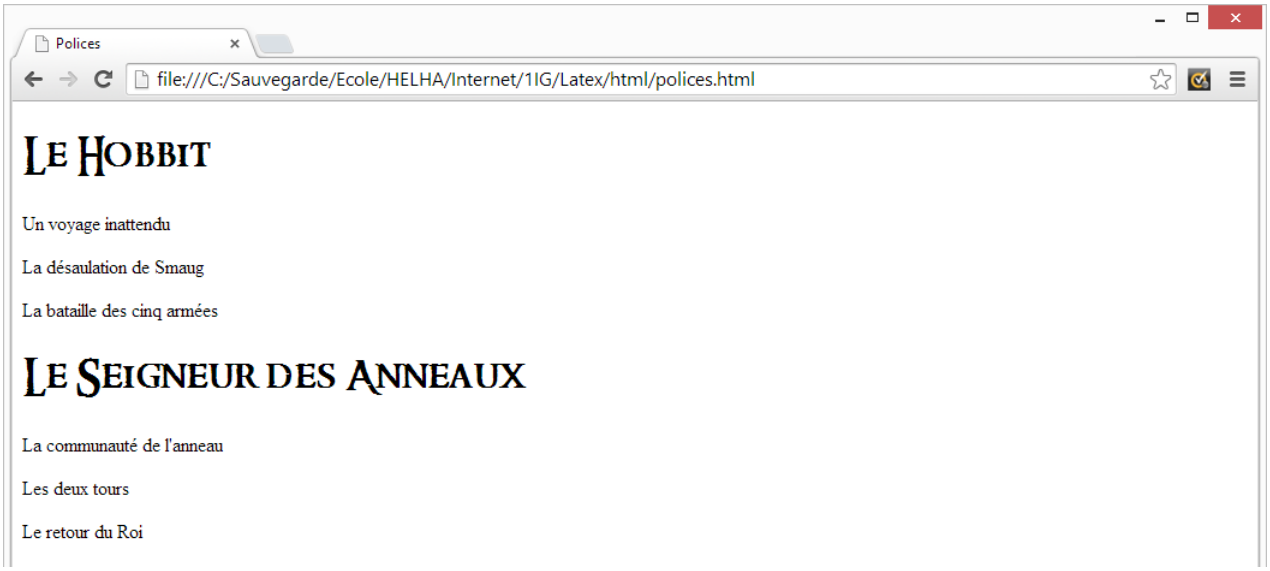
```
1  @charset "utf-8";
2  /* Définition d'une nouvelle police nommée maPoliceFavorite */
3  @font-face
4  {
5      font-family: 'ringbearermedium';
6      src: url('font/ringbearer-webfont.eot');
7      src: url('font/ringbearer-webfont.eot?#iefix') format('embedded-
           opentype'),
8           url('font/ringbearer-webfont.woff') format('woff'),
9           url('font/ringbearer-webfont.ttf') format('truetype'),
10          url('font/ringbearer-webfont.svg#ringbearermedium') format('
              svg');
11 }
12 /* Utilisation de la police qu'on vient de définir sur les titres */
13 h1
14 {
15     font-family: 'ringbearermedium', Arial, serif;
16 }
```

Le HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Polices</title>
        <link href="polices.css" media="all" rel="stylesheet" type="
           text/css" />
    </head>
    <body>
        <h1>Le Hobbit</h1>
        <p> Un voyage inattendu </p>
        <p> La désaulation de Smaug </p>
        <p> La bataille des cinq armées </p>
        <h1>Le Seigneur des Anneaux</h1>
        <p> La communauté de l'anneau </p>
        <p> Les deux tours </p>
        <p> Le retour du Roi </p>
    </body>
</html>
```

Le fichier de police doit être situé dans le même dossier que le fichier CSS (ou dans un sous-dossier, si nous utilisons un chemin relatif).

Ce qui nous donne graphiquement :



7.4 La mise en forme

Il existe en CSS une série de propriétés classiques de mise en forme du texte. Nous allons découvrir ici comment afficher le texte en gras, italique, souligné et au passage nous verrons qu'il est même possible d'aller jusqu'à le faire clignoter !

7.4.1 La mise en italique

Pour mettre en italique, nous utilisons **font-style** qui peut prendre trois valeurs :
- **italic** : le texte sera mis en italique.
- **oblique** : le texte sera passé en oblique (les lettres sont penchées, le résultat est légèrement différent de l'italique proprement dit).
- **normal** : le texte sera normal (par défaut). Cela nous permet d'annuler une mise en italique.

7.4.2 La mise en gras

La propriété CSS pour mettre en gras est **font-weight** et prend les valeurs suivantes :
- **bold** : le texte sera en gras ;
- **normal** : le texte sera écrit normalement (par défaut).

7.4.3 Le soulignement et les autres décorations

La propriété CSS associée porte bien son nom : **text-decoration**. Elle permet, entre autres, de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :
- **underline** : souligné.
- **line-through** : barré.
- **overline** : ligne au-dessus.
- **blink** : clignotant. Ne fonctionne pas sur tous les navigateurs (Internet Explorer et Google Chrome, notamment).
- **none** : normal (par défaut).

7.5 L'alignement

Le langage CSS nous permet de faire tous les alignements connus : à gauche, centré, à droite et justifié. Nous utilisons la propriété **text-align** et nous indiquons l'alignement désiré :
- **left** : le texte sera aligné à gauche (c'est l'alignement par défaut).
- **center** : le texte sera centré.
- **right** : le texte sera aligné à droite.

– **justify** : le texte sera justifié.

Nous ne pouvons pas modifier l'alignement du texte d'une balise inline (comme , <a>, , ...). L'alignement ne fonctionne que sur des balises de type block (<p>, , <h1>, <h2>, ...).

En effet, nous ne pouvons pas modifier l'alignement de quelques mots au milieu d'un paragraphe ! C'est donc en général le paragraphe entier qu'il vous faudra aligner.

En fait, grâce à l'alignement nous voyons apparaître les notions de contenant et de contenu. Le contenant et le contenu pouvant être soit de type block ou soit de type inline. Et donc, nous positionnons le contenu en fonction de son contenant.

7.6 Les flottants

Le CSS nous permet de faire flotter ou donner un habillage à un élément autour du texte. Pour ce faire, nous devons utiliser la propriété **float**. Cette propriété peut prendre deux valeurs :

– **left** : l'élément flottera à gauche ;
– **right** : l'élément flottera à droite.

7.6.1 L'image flottante

Afin de parvenir à faire flotter une image, nous devons respecter une règle simple : l'image doit toujours précéder le texte. Dans le cas contraire, l'image ne flottera pas. Bien entendu, le flottant sera lié à l'image dans ce cas.

7.6.2 La limite d'un flottant

Pour stopper l'effet d'un flottant pour du texte à placer en-dessous de celui-ci par exemple, nous devons utiliser la propriété **clear**.

Cette propriété propose les valeurs suivantes :

– **left** : le texte se poursuit en-dessous après un float : left ;
– **right** : le texte se poursuit en-dessous après un float : right ;
– **both** : le texte se poursuit en-dessous, que cela soit après un float : left ou right.

CHAPITRE 8

LA COULEUR ET LE FOND

8.1 LA COULEUR DU TEXTE

Nous avons déjà vu que la propriété qui permet de changer la couleur est **color**. Voyons à présent quelles sont les différentes valeurs admises par cette propriété.

8.1.1 Le nom de la couleur

La méthode la plus simple et la plus pratique pour choisir une couleur consiste à taper son nom en anglais. Nous retrouvons les couleurs disponibles à la figure suivante.

white		red		yellow		fuchsia	
silver		maroon		olive		purple	
gray		lime		blue		aqua	
black		green		navy		teal	

FIGURE 19 – Les 16 noms de couleur disponibles en CSS

Cette méthode a un seul défaut : il n'existe que seize couleurs dites "standards". D'autres couleurs officieuses existent mais, comme elles ne fonctionneront pas forcément de la même manière sur tous les navigateurs, nous éviterons de les utiliser.

8.1.2 La notation hexadécimale

Elle est couramment utilisée sur le Web. C'est un triplet précédé d'un dièse (#) et constitué de lettres (A à F) et de chiffres (0 à 9) qui indiquent une couleur. Ce triplet renseigne le poids de trois couleurs : red, green et blue.

Les valeurs varient de #000000 (couleur noire) à #FFFFFF (couleur blanche).

Nous pouvons rencontrer sa forme contractée comme #F3C qui correspond à #FF33CC.

Grâce à cette notation, nous passons de 16 couleurs à $2^{24} = 16777216$ couleurs soit plus ou moins 16 millions de couleurs.

Astuce 20 (Les palettes de couleurs)**.**

Afin d'avoir un aperçu de la couleur que donne une combinaison hexadécimale, nous pouvons utiliser un logiciel comme paint ou bien encore Adobe Photopshop. Ceux-ci possèdent une palette de couleurs qui renseigne ce rendu.

8.1.3 La méthode RGB

Elle est similaire à celle de la notation hexadécimale si ce n'est que les nombres sont renseignés comme décimaux. Encore une fois, les palettes peuvent nous être utiles.

Néanmoins, ce triplet de nombres décimaux est utilisé comme paramètre dans une fonction **rgb()** (par exemple rgb(240,96,204)).

8.2 LA COULEUR DE FOND

Pour indiquer une couleur de fond, nous utilisons la propriété **background-color**. Elle s'utilise de la même manière que la propriété **color**, c'est-à-dire : que nous pouvons taper le nom d'une couleur, l'écrire en notation hexadécimale ou encore utiliser la méthode RGB.

8.2.1 Le principe de base

Pour indiquer la couleur de fond de la page web, il faut travailler sur la balise **<body>**. Elle correspond à l'ensemble de la page web, c'est donc en modifiant sa couleur de fond que nous changerons la couleur d'arrière-plan de la page.

8.2.2 L'héritage

En CSS, si nous appliquons un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style. La balise <body>, nous le savons, contient entre autres les balises de paragraphe <p> et de titre <h1>. Si nous appliquons une couleur

de fond noire et une couleur de texte blanche à la balise **<body>**, tous nos titres et paragraphes auront eux aussi un arrière-plan de couleur noire et un texte de couleur blanche... Ce phénomène est appelé l'héritage : les balises qui se trouvent à l'intérieur d'une autre balise "héritent" de ses propriétés.

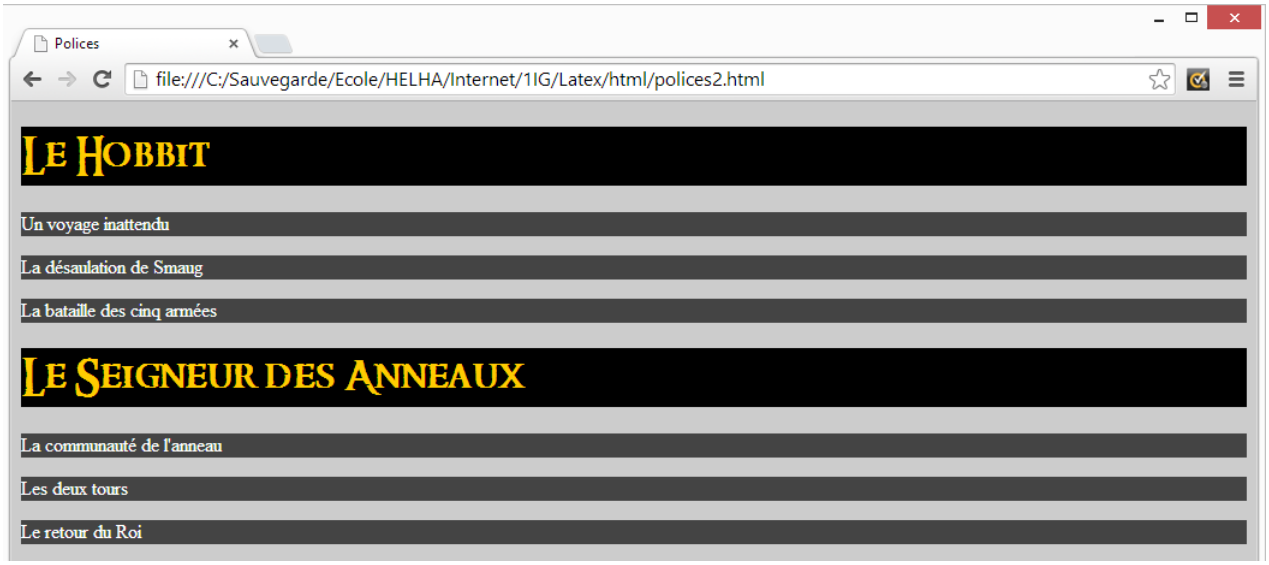
Exemple 31. *Application de couleurs*
Le CSS :

```
1  @charset "utf-8";
2
3  @font-face
4  {
5      font-family: 'ringbearermedium';
6      src: url('font/ringbearer-webfont.eot');
7      src: url('font/ringbearer-webfont.eot?#iefix') format('embedded-
           opentype'),
8           url('font/ringbearer-webfont.woff') format('woff'),
9           url('font/ringbearer-webfont.ttf') format('truetype'),
10          url('font/ringbearer-webfont.svg#ringbearermedium') format('
              svg');
11 }
12
13 body
14 {
15          background-color: #CCC;
16          color:#84004A;
17 }
18
19 h1
20 {
21     font-family: 'ringbearermedium', Arial, serif;
22          background-color:black;
23          color: #FC0;
24 }
25
26 p
27 {
28          color:white;
29          background-color:  #444;
30 }
```

Le HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Polices</title>
        <link href="polices2.css" media="all" rel="stylesheet" type="
            text/css" />
    </head>
    <body>
        <h1>Le Hobbit</h1>
        <p> Un voyage inattendu </p>
        <p> La désaulation de Smaug </p>
        <p> La bataille des cinq armées </p>
        <h1>Le Seigneur des Anneaux</h1>
        <p> La communauté de l'anneau </p>
        <p> Les deux tours </p>
        <p> Le retour du Roi </p>
    </body>
</html>
```

Ce qui nous donne visuellement :



8.3 L'IMAGE DE FOND

Outre le fait de changer la couleur de fond de notre page ou de ses éléments, il nous est possible d'appliquer une image personnalisée.

La propriété permettant d'indiquer une image de fond est **background-image**. Comme valeur, nous devons renseigner une url grâce à **url("nomImage.png")**. Prenons un exemple :

Exemple 32. *Application d'une image de fond*

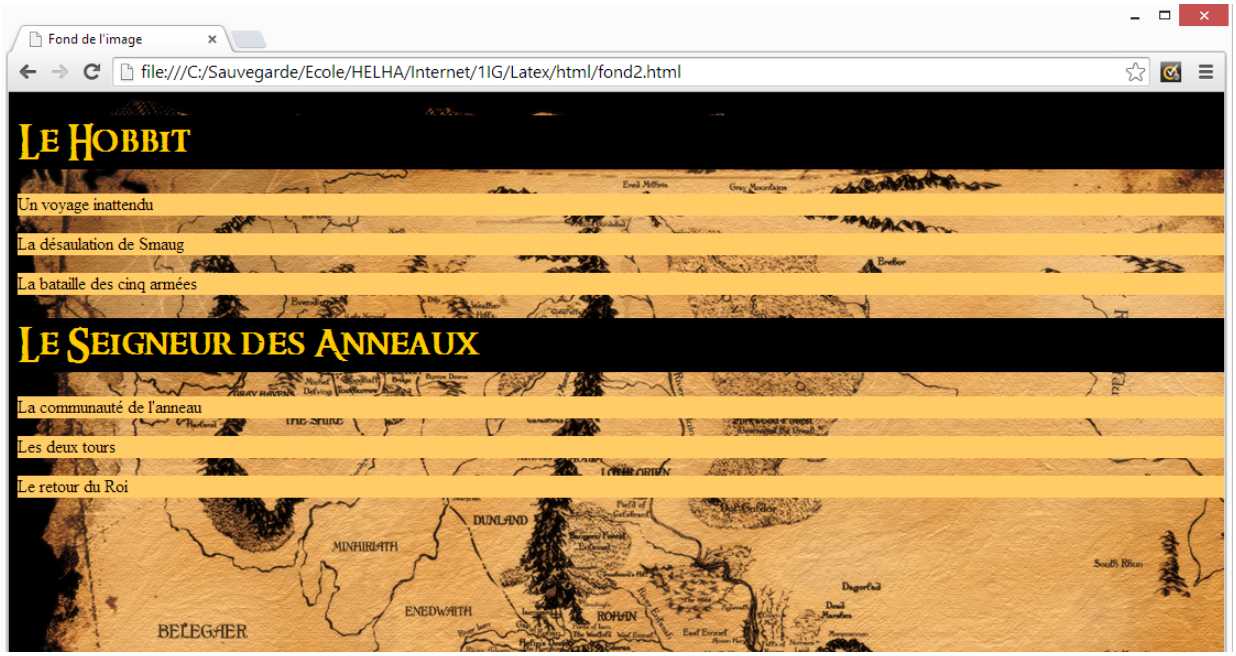
Le CSS :

```
@charset "utf-8";
@font-face
{
    font-family: 'ringbearermedium';
    src: url('font/ringbearer-webfont.eot');
    src: url('font/ringbearer-webfont.eot?#iefix') format('embedded-
        opentype'),
         url('font/ringbearer-webfont.woff') format('woff'),
         url('font/ringbearer-webfont.ttf') format('truetype'),
         url('font/ringbearer-webfont.svg#ringbearermedium') format('
            svg');
}

body
{
        background-image:url('images/carte.png');
        width:100%;
        color:#84004A;
}

h1
{
    font-family: 'ringbearermedium', Arial, serif;
        background-color:black;
        color: #FC0;
}

p
{
        color:black;
        background-color:  #FC6;
}
```

Le HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Fond de l'image</title>
        <link href="fond2.css" media="all" rel="stylesheet" type="
           text/css" />
    </head>
    <body>
        <h1>Le Hobbit</h1>
        <p> Un voyage inattendu </p>
        <p> La désaulation de Smaug </p>
        <p> La bataille des cinq armées </p>
        <h1>Le Seigneur des Anneaux</h1>
        <p> La communauté de l'anneau </p>
        <p> Les deux tours </p>
        <p> Le retour du Roi </p>
    </body>
</html>
```

Ce qui nous donne visuellement :



On peut compléter la propriété background-image que nous venons de voir par plusieurs autres propriétés qui permettent de changer le comportement de l'image de fond.

8.3.1 La taille du fond

Elle se traduit par la propriété **background-size** où sa valeur est constituée de sa largeur et de sa hauteur renseignées en pixels ou en %. Si une des deux valeurs n'est pas renseignée, alors la propriété applique celle par défaut c'est-à-dire : **auto**.

Nous pouvons également appliquer l'un des mots-clés suivants :
- **contain** : qui spécifie que l'image d'arrière-plan doit être mise à l'échelle pour être aussi grande que possible tout en assurant que les deux dimensions sont plus petites ou égales aux dimensions de la surface de positionnement ;
- **cover**, qui spécifie que l'image d'arrière-plan doit être mise à l'échelle pour être aussi petite que possible tout en assurant que les deux dimensions soient supérieures ou égales aux dimensions de la surface de positionnement.

8.3.2 La fixation du fond

La propriété **background-attachment** permet de "fixer" le fond. L'effet obtenu est intéressant car nous voyons alors le texte "glisser" par-dessus le fond.
Deux valeurs sont disponibles :
- **fixed** : l'image de fond reste fixe ;
- **scroll** : l'image de fond défile avec le texte (par défaut).

8.3.3 La répétition du fond

Par défaut, l'image de fond est répétée en mosaïque. Nous pouvons changer cela avec la propriété **background-repeat** :
- **no-repeat** : le fond ne sera pas répété. L'image sera donc unique sur la page ;
- **repeat-x** : le fond sera répété uniquement sur la première ligne, horizontalement ;
- **repeat-y** : le fond sera répété uniquement sur la première colonne, verticalement ;
- **repeat** : le fond sera répété en mosaïque (par défaut).

8.3.4 La position du fond

Nous pouvons indiquer où doit se trouver l'image de fond avec **background-position**. Cette propriété n'est intéressante que si elle est combinée avec **background-repeat** : **no-repeat** ; (un fond qui ne se répète pas).
Nous devons donner à **background-position** deux valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page (ou du paragraphe, si nous appliquons le fond à un paragraphe).

Il est aussi possible d'utiliser ces valeurs en anglais et de les combiner :
- **top** : en haut ;
- **bottom** : en bas ;
- **left** : à gauche ;
- **center** : centré ;
- **right** : à droite.

8.3.5 La combinaison des propriétés

Si nous utilisons beaucoup de propriétés en rapport avec le fond, nous pouvons utiliser une sorte de "super-propriété" appelée **background** dont la valeur peut combiner plusieurs des propriétés vues précédemment : **background-image**, **background-repeat**, **background-attachment** et **background-position**.

Il faut savoir que :
- L'ordre des valeurs n'a pas d'importance. Nous pouvons combiner les valeurs dans n'importe quel ordre ;
- Nous ne sommes pas obligés de mettre toutes les valeurs. Ainsi, si vous ne voulez pas écrire **fixed**, vous pouvez l'enlever sans problème.

8.3.6 La composition avec les images de fond

Depuis CSS3, il est possible de donner plusieurs images de fond à un élément. Pour cela, il suffit de séparer les déclarations par une virgule.

A noter que les images de fond multiples fonctionnent sur tous les navigateurs sauf sur les anciennes versions d'Internet Explorer, qui ne reconnaît cette fonctionnalité qu'à partir de la version 9 (IE9).

8.4 LA TRANSPARENCE

Le CSS nous permet de jouer très facilement avec les niveaux de transparence des éléments. Pour cela, nous allons utiliser des fonctionnalités de CSS3 : la propriété **opacity** et la notation RGBa.

8.4.1 L'opacité

La propriété **opacity** permet d'indiquer le niveau d'opacité (inverse de la transparence).
- avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut ;
- avec une valeur de 0, l'élément sera totalement transparent.

Il faut donc choisir une valeur comprise entre 0 et 1. Ainsi, avec une valeur de 0.6, notre élément sera opaque à 60%.

8.4.2 La notation RGBa

CSS3 nous propose une autre façon de jouer avec la transparence : la notation RGBa. Il s'agit en fait de la notation RGB que nous avons vue précédemment, mais avec un quatrième paramètre : le niveau de transparence (appelé « canal alpha »). De la même façon que précédemment, avec une valeur de 1, le fond est complètement opaque. Avec une valeur inférieure à 1, il est transparent.

Cette notation est connue de tous les navigateurs récents, y compris Internet Explorer (à partir de IE9). Pour les navigateurs plus anciens, il est recommandé d'indiquer la notation RGB classique en plus de RGBa. Pour ces navigateurs, le fond ne sera alors pas transparent mais, au moins, il y aura bien une couleur d'arrière-plan.

CHAPITRE 9

LES BORDURES ET LES OMBRES

Dans ce chapitre, nous allons mettre encore plus de panache à nos mise en forme. Les bordures et les ombrages donnent un effet relativement impressionnant.

9.1 LES BORDURES

Le CSS nous offre un large choix de bordures pour décorer notre page. De nombreuses propriétés CSS nous permettent de modifier l'apparence de nos bordures : **border-width**, **border-color**, **border-style**...

9.1.1 Les bordures standards

Afin de simplifier le nombre de propriété, nous pouvons utiliser la super-propriété **border** qui regroupe l'ensemble de ces propriétés. Pour border, nous pouvons utiliser jusqu'à trois valeurs pour modifier l'apparence de la bordure :

- la **largeur** : indique la largeur de notre bordure en pixels ;
- la **couleur** : indique la couleur de notre bordure ;
- le **type de bordure** : notre bordure peut être un simple trait, ou des pointillés, ou encore des tirets, etc. Voici les différentes valeurs disponibles :
  - **none** : pas de bordure (par défaut) ;
  - **solid** : un trait simple ;
  - **dotted** : pointillés ;
  - **dashed** : tirets ;
  - **double** : bordure double ;
  - **groove** : en relief ;
  - **ridge** : autre effet de relief ;
  - **inset** : effet 3D global enfoncé ;
  - **outset** : effet 3D global surélevé.

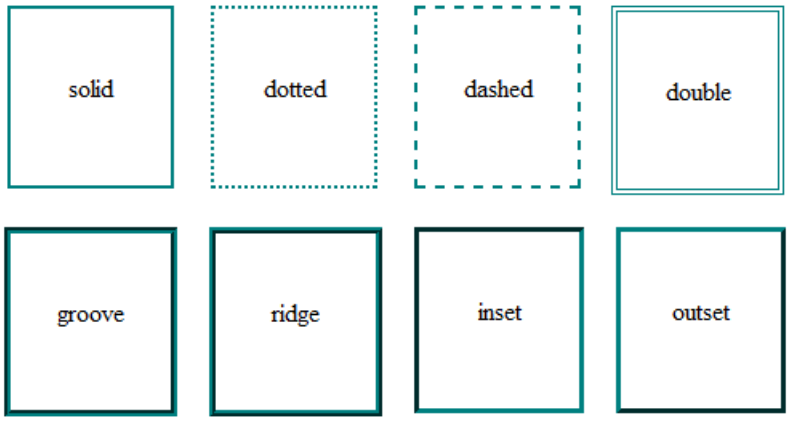


FIGURE 20 – Les différents types de bordure

Nous pouvons personnaliser chacune des 4 bordures qui entourent notre élément.
Voici les propriétés pour y accéder :
– **border-top** : bordure du haut ;
– **border-bottom** : bordure du bas ;
– **border-left** : bordure de gauche ;
– **border-right** : bordure de droite.

Il existe aussi des équivalents pour paramétrer chaque détail de la bordure si nous le désirons : **border-top-width** pour modifier l'épaisseur de la bordure du haut, **border-top-color** pour la couleur du haut, etc.

9.1.2 Les bordures arrondies

La propriété **border-radius** va nous permettre d'arrondir facilement les angles de n'importe quel élément. Il suffit d'indiquer la taille (« l'importance ») de l'arrondi en pixels.

De plus, nous pouvons personnaliser l'arrondi de chaque coin et leur appliquer un effet elliptique. D'une part, pour la personnalisation des coins, il faut renseigner les 4 coins. D'autre part, pour un effet elliptique, il faut utiliser une formule constituée de deux valeurs en pixels séparés par un /.

Astuce 21 (Les préfixes vendeurs).

Les bordures arrondies fonctionnent avec tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).
Pour les anciennes versions de Mozilla Firefox, Chrome et Safari, il était nécessaire d'utiliser ce qu'on appelle des « préfixes vendeurs », c'est-à-dire qu'il fallait écrire dans le code CSS différentes versions de la propriété (-moz-border-radius pour Firefox, -webkit-border-radius pour Safari, etc.). Ce n'est heureusement plus nécessaire aujourd'hui, sauf si nous voulons gérer les anciennes versions de ces navigateurs.

9.2 Les ombres

Nous allons ici découvrir deux types d'ombres :
– les ombres des boîtes ;
– les ombres du texte.

9.2.1 Les ombres des boîtes

La propriété **box-shadow** s'applique à tout le bloc et prend quatre valeurs dans l'ordre suivant :

1. le décalage horizontal de l'ombre ;
2. le décalage vertical de l'ombre ;
3. l'adoucissement du dégradé ;
4. la couleur de l'ombre.

Nous pouvons aussi ajouter une cinquième valeur facultative : **inset**. Dans ce cas, l'ombre sera placée à l'intérieur du bloc, pour donner un effet enfoncé.

9.2.2 Les ombres du texte

Avec **text-shadow**, nous pouvons ajouter une ombre directement sur les lettres de notre texte. Les valeurs fonctionnent exactement de la même façon que **box-shadow** : décalage, adoucissement et couleur.

Exemple 33. *Application de bordures et d'ombrages*

Le CSS :

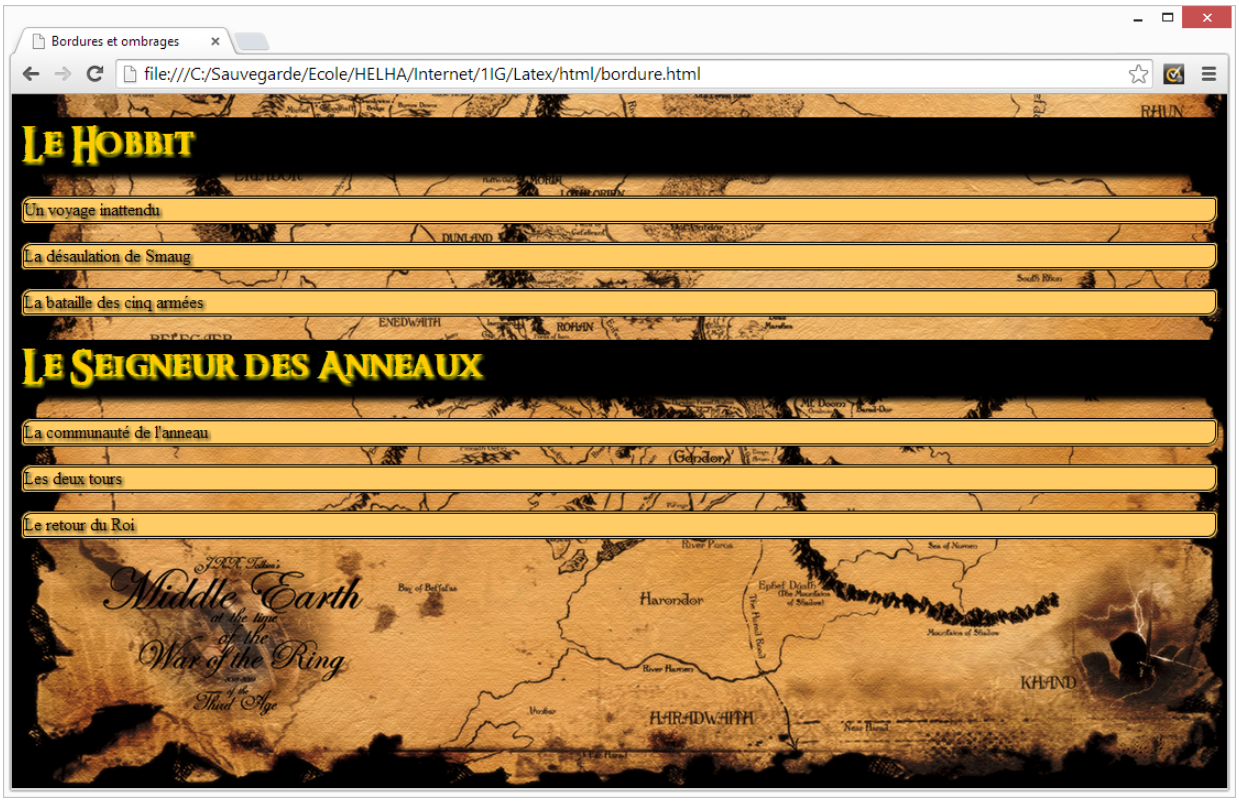
```
@charset "utf-8";
@font-face
{
    font-family: 'ringbearermedium';
    src: url('font/ringbearer-webfont.eot');
    src: url('font/ringbearer-webfont.eot?#iefix') format('embedded-
        opentype'),
         url('font/ringbearer-webfont.woff') format('woff'),
         url('font/ringbearer-webfont.ttf') format('truetype'),
         url('font/ringbearer-webfont.svg#ringbearermedium') format('
            svg');
}

body
{
        background: url('images/carte.png') fixed no-repeat bottom;
        background-size: 100%;
        color:#84004A;
}

h1
{
    font-family: 'ringbearermedium', Arial, serif;
        background-color:black;
        color: #FC0;
        border-radius: 10px 5px 10px 5px;
        box-shadow: 6px 6px 6px black;
        text-shadow: 2px 2px 4px yellow;
}

p
{
        color:black;
        background-color:  #FC6;
        border: double;
        border-radius: 10px 5px 10px 5px;
        text-shadow: 2px 2px 4px black;
}
```

Le HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
```

```
5           <title>Bordures et ombrages</title>
6           <link href="bordure.css" media="all" rel="stylesheet" type="
                text/css" />
7       </head>
8       <body>
9           <h1>Le Hobbit</h1>
10          <p> Un voyage inattendu </p>
11          <p> La désaulation de Smaug </p>
12          <p> La bataille des cinq armées </p>
13          <h1>Le Seigneur des Anneaux</h1>
14          <p> La communauté de l'anneau </p>
15          <p> Les deux tours </p>
16          <p> Le retour du Roi </p>
17      </body>
18  </html>
```

Ce qui nous donne visuellement :



CHAPITRE 10

LA CRÉATION D'APPARENCES DYNAMIQUES

Le CSS 3 introduit les concepts de pseudo-classes et de pseudo-éléments ce qui permet une mise en forme à partir d'informations absentes de l'arbre du document :

- Les **pseudo-éléments** créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document. Ainsi, certains langages n'offrent pas de mécanismes de correspondance avec la première lettre ou la première ligne du contenu d'un élément. Les pseudo-éléments de CSS permettent aux auteurs d'y accéder, ce qui serait autrement impossible. Ces pseudo-éléments leur permettent de donner un style à un contenu qui n'apparaît même pas dans le document source. Par exemple, les pseudo-éléments **:before** et **:after** autorisent une action sur un contenu généré.
- Les **pseudo-classes** classent les éléments selon des caractéristiques autres que leur nom, attribut ou contenu, celles-ci ne pouvant pas en principe être déduites de l'arbre du document. Les pseudo-classes peuvent être dynamiques, dans le sens où un élément peut les acquérir ou les perdre pendant l'interaction de l'utilisateur avec le document. Une exception, la pseudo-classe ' **:first-child**', qui peut se déduire de l'arbre. Dans certains cas, la pseudo-classe ' **:lang()**' peut également se déduire du document.

Ni les pseudo-éléments, ni les pseudo-classes n'apparaissent dans la source ou l'arbre du document.

Les pseudo-classes sont autorisées n'importe où dans le sélecteur, alors qu'un pseudo-élément ne peut être spécifié qu'après le sujet du sélecteur.

Les pseudo-éléments et pseudo-classes sont insensibles à la casse.

Certaines pseudo-classes s'excluent mutuellement, d'autres peuvent s'appliquer simultanément au même élément. Les conflits éventuels se résolvent selon l'ordre normal de cascade.

Afin de rester générique dans nos propos, nous parlerons d'agent utilisateur exécutant des actions. Cette notion sera approfondie plus tard au niveau du référencement de notre site.

Définition 10 (Agent utilisateur)**.**

Un agent utilisateur est une application cliente utilisée avec un protocole réseau particulier. L'expression est plus généralement employée comme référence pour celles qui accèdent au World Wide Web. Les agents utilisateur du Web vont de la gamme des navigateurs jusqu'aux robots d'indexation, en passant par les lecteurs d'écran ou les navigateurs braille pour les personnes ayant une incapacité.

10.1 LES PSEUDO-ÉLÉMENTS

10.1.1 Le pseudo-élément :first-line

Le pseudo-élément **:first-line** produit un style particulier sur la première ligne formatée d'un paragraphe. Par exemple : **P :first-line text-transform : uppercase**

La règle précédente signifie "mettre les lettres de la première ligne de chaque paragraphe en majuscule". Cependant, le sélecteur "**P :first-line**" ne correspond à aucun élément HTML réel. Celui-ci correspond en fait à un pseudo-élément que l'agent utilisateur va insérer au début de chaque paragraphe.

Notons que la longueur de la première ligne dépend de plusieurs facteurs, dont la largeur de la page, la taille de la police, etc. Quand un pseudo-élément coupe un élément réel, on peut souvent décrire l'effet désiré avec une séquence de balises fictives qui ferment puis réouvrent cet élément.

Le pseudo-élément **:first-line** ne peut s'attacher qu'à un élément de type block. Le pseudo-élément :first-line est similaire à un élément de type inline, avec certaines restrictions. Seules les propriétés suivantes peuvent lui être appliquées : les propriétés de police, les propriétés de couleur, les propriétés d'arrière-plan, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'text-shadow' et 'clear'.

10.1.2 Le pseudo-élément :first-letter

Le pseudo-élément **:first-letter** peut être employé pour faire des capitales initiales et des lettrines, ce sont des effets typographiques courants. Ce genre de lettre initiale est assimilé à un élément de type en-ligne quand la valeur de sa propriété 'float' est 'none', et assimilé à un élément flottant autrement.

Le pseudo-élément :first-letter admet ces propriétés : les propriétés de police, les propriétés de couleur, les propriétés d'arrière-plan, 'text-decoration', 'vertical-align' (seulement si la valeur de la propriété 'float' est 'none'), 'text-transform', 'line-height', les propriétés de marge, les propriétés d'espacement, les propriétés de bordure, 'float', 'text-shadow' et 'clear'.

Prenons un exemple afin d'illustrer ces pseudo-éléments :

Exemple 34. *Application de pseudo-éléments*
Le CSS :

```
@charset "utf-8";
@font-face
{
    font-family: 'ringbearermedium';
    src: url('font/ringbearer-webfont.eot');
    src: url('font/ringbearer-webfont.eot?#iefix') format('embedded-
        opentype'),
         url('font/ringbearer-webfont.woff') format('woff'),
         url('font/ringbearer-webfont.ttf') format('truetype'),
         url('font/ringbearer-webfont.svg#ringbearermedium') format('
            svg');
}

body
{
        background: url('images/carte.png') fixed no-repeat bottom;
        background-size: 100%;
        color:#84004A;
}

h1
{
    font-family: 'ringbearermedium', Arial, serif;
        background-color:black;
        color: #FC0;
        border-radius: 10px 5px 10px 5px;
        box-shadow: 6px 6px 6px black;
        text-shadow: 2px 2px 4px yellow;
}

p
{
        text-shadow: 2px 2px 4px black;
        background-color:  #FC6;
}

p, div
{
        color:black;
        border: double;
        border-radius: 10px 5px 10px 5px;
```

```
        padding:0.5%;
}

div
{
        background-color:  #DB7;
        font-size: 12pt;
        line-height: 12pt;
        opacity:0.8;
        text-align:justify;
        margin-left:5%;
}

div:first-letter
{
        font-size: 200%;
        font-style: italic;
    font-weight: bold;
        float: left;
}

span
{
        text-transform: uppercase;
}
```

Le HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Lettrine</title>
        <link href="lettrine.css" media="all" rel="stylesheet" type="
           text/css" />
    </head>
    <body>
        <h1>Le Hobbit</h1>
                <p> Un voyage inattendu </p>
        <div>Dans <span>un voyage inattendu</span>, Bilbon Sacquet
           cherche à reprendre le Royaume perdu des Nains d'Erebor,
           conquis par le redoutable dragon Smaug. Alors qu'il
           croise par hasard la route du magicien Gandalf le Gris,
           Bilbon rejoint une bande de 13 nains dont le chef n'est
           autre que le légendaire guerrier Thorin Écu-de-Chêne.
           Leur périple les conduit au cur du Pays Sauvage, où ils
           devront affronter des Gobelins, des Orques, des Ouargues
```

```
            meurtriers, des Araignées géantes, des Métamorphes et des
             Sorciers
</div>
                <p> La désaulation de Smaug </p>
        <div> Les aventures de Bilbon Sacquet, paisible hobbit, qui
            sera entraîné, lui et une compagnie de Nains, par le
            magicien Gandalf pour récupérer le trésor détenu par le
            dragon Smaug. Au cours de ce périple, il mettra la main
            sur l'anneau de pouvoir que possédait Gollum... </div>

                <p> La bataille des cinq armées </p>
        <div>Le troisième et dernier volet de la trilogie du Hobbit
            de Peter Jackson. </div>
        <h1>Le Seigneur des Anneaux</h1>
                <p> La communauté de l'anneau </p>
                <p> Les deux tours </p>
                <p> Le retour du Roi </p>
    </body>
</html>
```

Ce qui nous donne visuellement :

Nous remarquons la présence des propriétés **marge** et **padding**, elles seront étudiées dans les chapitres suivants.

10.1.3 Les pseudo-éléments :before et :after

Les pseudo-élément ' **:before**' et ' **:after**' servent à insérer un contenu (texte, compteur, image, ...) généré avant ou après celui d'un élément. Combinés aux pseudo-éléments **:before** et **:after**, le pseudo-élément **:first-letter**, ou **:first-line**, s'applique à la première lettre, ou à la première ligne, de l'élément, y compris le texte inséré.

10.2 Les pseudo-classes

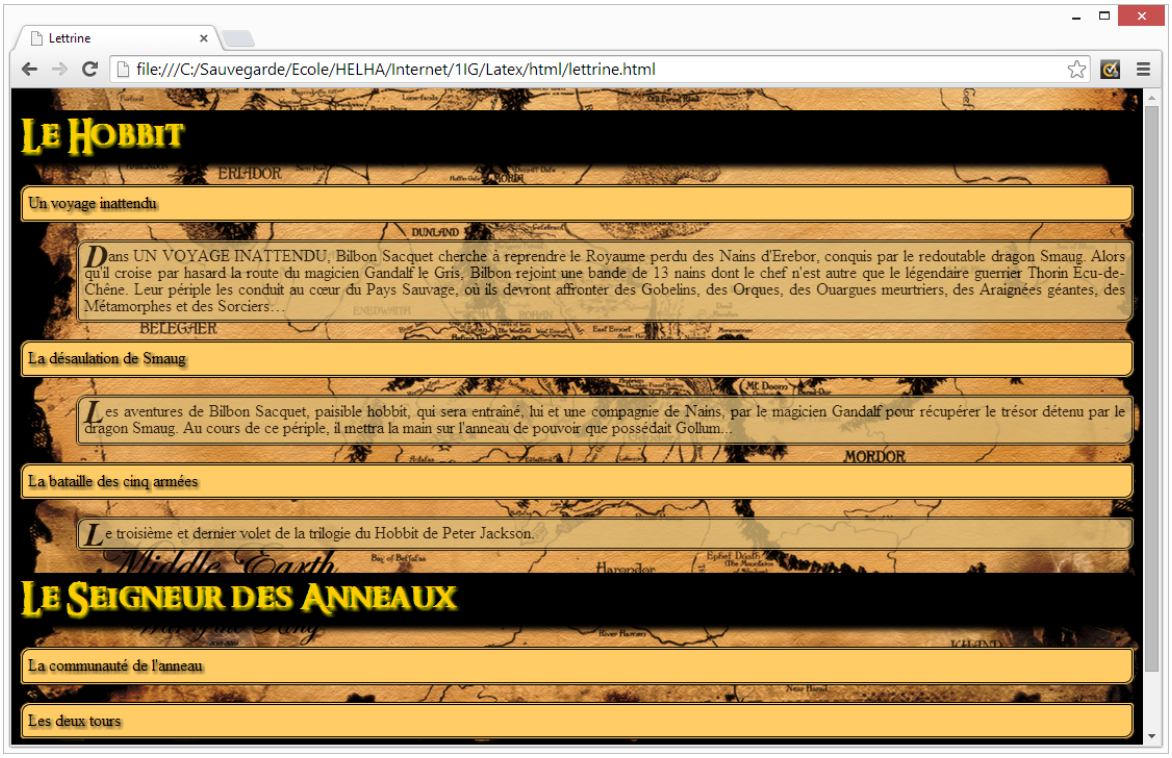
10.2.1 Les pseudo-classes des liens

En général, les agents utilisateurs représentent différemment les liens qui n'ont pas été visités de ceux qui l'ont déjà été. CSS en fournit un équivalent au travers des pseudo-classes ' **:link**' et ' **:visited**'.

– La pseudo-classe **:link** s'applique aux liens qui n'ont pas été visités ;

– La pseudo-classe **:visited** s'applique lorsque le lien a été visité par l'utilisateur.

Remarque : Après un certain temps, les agents utilisateurs peuvent revenir de l'état visité à l'état non-visité du lien. Ces deux états s'excluent mutuellement.



Le langage du document détermine quelles ancres ont des sources hyperliens. Ainsi, les pseudo-classes d'ancre s'appliquent aux éléments A avec un attribut "href".

Les deux déclarations CSS 3 suivantes produisent le même effet :

Exemple 35. *La similitude de pseudo-classes*

```
1 A:link { color: red }
2 :link  { color: red }
```

Pour ce lien :

```
1 <A class="external" href="http://out.side/">lien externe</A>
```

si celui est visité, la règle suivante :

```
1 A.external:visited { color: blue }
```

... rendra bleue la couleur du lien.

10.2.2 Les pseudo-classes d'interaction avec l'utilisateur

Les agents utilisateurs interactifs changent parfois l'aspect du rendu en réponse aux actions de l'utilisateur. En CSS 3, il y a trois pseudo-classes pour un usage courant :

- La pseudo-classe **:hover**, qui est appliquée quand l'utilisateur désigne un élément (au moyen d'un appareil de pointage) sans l'activer. Par exemple, un agent utilisateur visuel pourrait appliquer celle-ci quand le pointeur (d'une souris) survole la boîte produite par un élément. Les agents utilisateurs qui n'ont pas de capacités interactives ne sont pas tenus d'appliquer cette pseudo-classe. Certains agents utilisateurs conformes dotés de ces capacités interactives peuvent être incapables de l'appliquer (ex. un crayon optique) ;
- La pseudo-classe **:active**, qui est appliquée quand l'utilisateur active un élément. Par exemple, entre le moment où l'utilisateur presse le bouton de la souris et le relâche.
- La pseudo-classe **:focus**, qui s'applique quand un élément reçoit l'attention (celui-ci acceptant les événements du clavier ou d'autres formes d'entrées de texte).

Ces pseudo-classes ne s'excluent pas mutuellement. Un élément peut correspondre à plusieurs d'entre elles au même moment.

CSS ne définit pas lesquels des éléments peuvent être dans un de ces états ou comment ceux-ci entrent et sortent de ces états. L'écriture peut varier, selon que les éléments réagissent aux actions de l'utilisateur, ou non ; les divers appareils et agents utilisateurs peuvent avoir différentes façons de désigner ou d'activer les éléments.

Les agents utilisateurs ne sont pas tenus, en raison des transitions provoquées par les pseudo-classes, de remettre en forme le document en cours d'affichage. Par exemple, une feuille de style peut spécifier que la taille de la police ('font-size') d'un lien sous l'effet de la pseudo-classe :active soit plus grande que celle d'un lien inactif, et un agent utilisateur, comme cela peut modifier la position des lettres, peut en ignorer la règle.

Exemple 36. *L'ordre des peudo-classes*

```
1 @charset "utf-8";
2 A:link    { color: red }    /* lien non-visité */
3 A:visited { color: blue }   /* lien visité     */
4 A:hover   { color: yellow } /* lien survolé    */
5 A:active  { color: lime }   /* lien activé     */
```

*Notons que la règle **A :hover** doit être placé après **A :link** et **A :visited**, autrement les règles de cascade feront que la propriété 'color' spécifiée par celle-ci sera cachée. De la même façon, comme la règle **A :active** est placée après **A :hover**, la couleur spécifiée par celle-ci (lime) sera appliquée quand l'utilisateur active et survole l'élément A.*

Voici un exemple de combinaison de pseudo-classes dynamiques :

Exemple 37. *La combinaison de pseudo-classes dynamiques*

```
1 A:focus { background: yellow }
2 A:focus:hover { background: white }
```

Le dernier sélecteur correspond à un élément A qui a l'attention et qui est survolé.

10.2.3 Les autres pseudo-classes

La pseudo-classe **:first-child** correspond au premier élément enfant d'un autre élément.

La pseudo-classe **:lang(C)** est vérifiée pour un élément dans la langue C (fr, en, ...).

Troisième partie

La mise en page d'un site

CHAPITRE 11

LA STRUCTURE D'UNE PAGE

Dans ce chapitre, nous allons nous intéresser aux anciennes et nouvelles balises HTML dédiées à la structuration du site.

11.1 LES FRAMES

Les frames ont longtemps dominés le choix de structure des sites. A l'heure actuelle, plus aucun site n'est structurer avec des frames.

Cependant, il demeure un type de frame qui est toujours utilisé mais dont il ne faut pas en abuser : l'iframe.

11.1.1 Les frames standards

Pour diviser l'écran en plusieurs fenêtres, les balises sont peu nombreuses : **<frameset></frameset>**. Cette balise permet de délimiter :

- un début de zone avec des fenêtres ;
- une fin de zone avec des fenêtres ;
- un agencement des fenêtres.

Les attributs **rows** et **cols** permettent de scinder la fenêtre de manière horizontale ou verticale. La valeur indiquée représente le nombre de pixels ou le pourcentage de largeur des parties (2 à n) ainsi crées.

La balise **<frameset>** remplace le rôle de la balise **body**.

Pour l'instant, nos frames sont vides. Nous allons donc les garnir par l'attribut **src** d'une balise **frame**.

Nous pouvons imbriquer plusieurs balises **<frameset>** et les combiner avec des balises **<frame>**.

Les ascenseurs apparaissent automatiquement jusque maintenant. Si nous voulons changer ce comportement, il nous suffit d'utiliser l'attribut de la balise **<frame>** **scrolling**

prenant comme valeur **yes** ou **n**) ou **auto**.

L'attribut **name** indique le nom de la fenêtre de telle sorte que cette frame puisse être utilisée comme cible d'un lien hyper-texte. Donc, à un moment donné, nous devons utiliser l'attributtarget pour mentionner la cible à atteindre en fonction du lien et du nom de la frame définis.

L'attribut **target** peut aussi prendre certaines valeurs prédéfinies :

- **_blank** qui indique au navigateur qu'il doit créer une nouvelle fenêtre afin d'y afficher le fichier. Dans ce cas, nous ouvrons en fait un nouveau navigateur.
- **_self** qui indique que le fichier sera chargé dans la même fenêtre que celle dans laquelle se trouve le lien.
- **_top** qui implique l'affichage du fichier sur toute la surface de la fenêtre du navigateur.

L'un des inconvénients majeurs des frames est qu'il faut toujours vérifier les liens de nos frames.

Exemple 38. *Les jeux de frames*

```
...
<frameset rows="30%,70%">
        <frame src="a.html" />
        <frameset cols="30%,70%">
                <frame src="b.html" />
                <frame src="c.html" name="fenetreC" />
        </frameset>
</frameset>
...
```

Nous avons dans le fichier b.html :

```
...
<a href="a.html" target="fenetreC"><h1>B</h1></A>
...
```

Nous retrouvons à la page suivante un exemple illustré.

11.1.2 Les iframes

Une iframe est un cadre qui eut être placé à l'intérieur de nos pages avec le contenu de notre choix. Contrairement à la frame normale, l'iframe peut se placer n'importe où sur la page de manière totalement indépendante.

L'iframe conserve les propriétés des frames simples : nous pouvons en faire changer le contenu grâce aux liens externes et à l'attribut **target**.

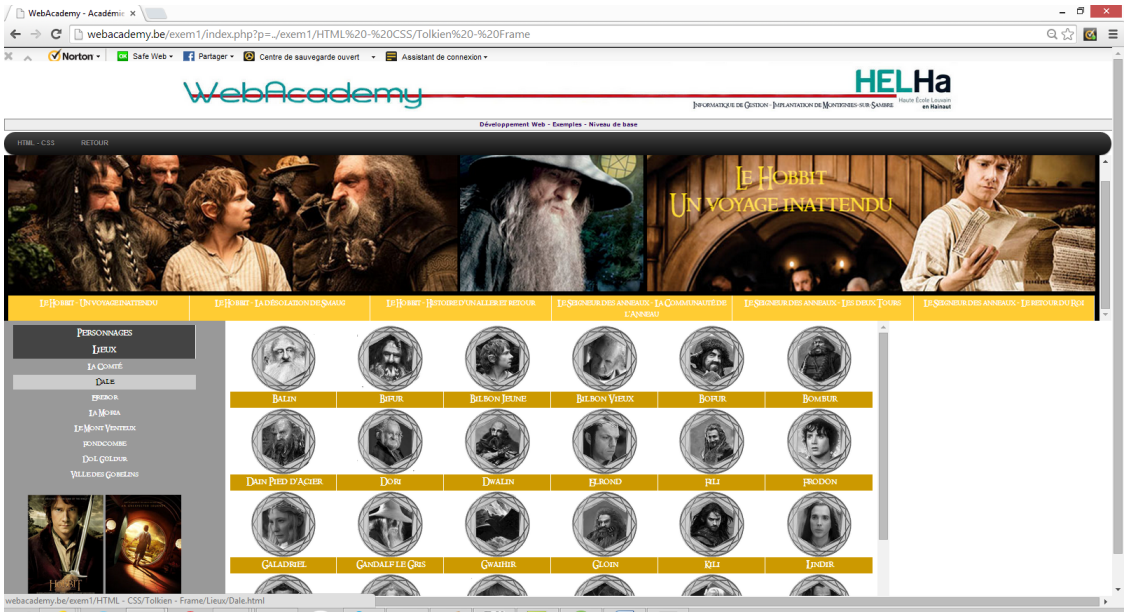


FIGURE 21 – Exemple de structure avec des frames

Exemple 39. *L'iframe*

Code de l'iframe :

```
...
<iframe align="right" width="500" height="500" frameborder="1"
    scrolling="auto" src="page1.htm" name="iframe"></iframe>
...
```

Code du lien qui change le contenu de la iframe :

```
...
<a href="page2.htm" target="iframe">Page 2</a>
...
```

Les iframes ont pour avantage de constituer un contenu séparé de la page hôte, c'est-à-dire : la page affichée dans l'iframe peut avoir ses propres scripts et styles qui n'interagissent pas avec la page hôte, et cette isolation est parfois utile.

Nous pourrions tenter de faire sensiblement la même chose avec la balise **<object>** mais ça n'a pas vraiment d'intérêt puisque cette dernière est en osmose avec la page hôte.

11.2 LES BALISES DE STRUCTURE DE LA PAGE

11.2.1 L'en-tête

La plupart des sites web possèdent en général un en-tête, appelé header en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de notre site... Nous devrons placer ces informations à l'intérieur de la balise **<header>** :

Exemple 40. *La balise <head>*

```
<header>
<!-- Plaçons ici le contenu de l'en-tête de notre page -->
</header>
```

Nous pouvons illustrer cette balise par l'ancien en-tête du site du zéro (open classroom maintenant).

FIGURE 22 – Exemple d'en-tête de page

11.2.2 Le pied de page

À l'inverse de l'en-tête, la balise **footer** se trouve en général tout en bas du document. Nous y trouvons des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

Exemple 41. *La balise <footer>*

```
<footer>
<!-- Plaçons ici le contenu du pied de page -->
</footer>
```

Nous pouvons illustrer cette balise par l'ancien pied de page du site du zéro (open classroom maintenant).

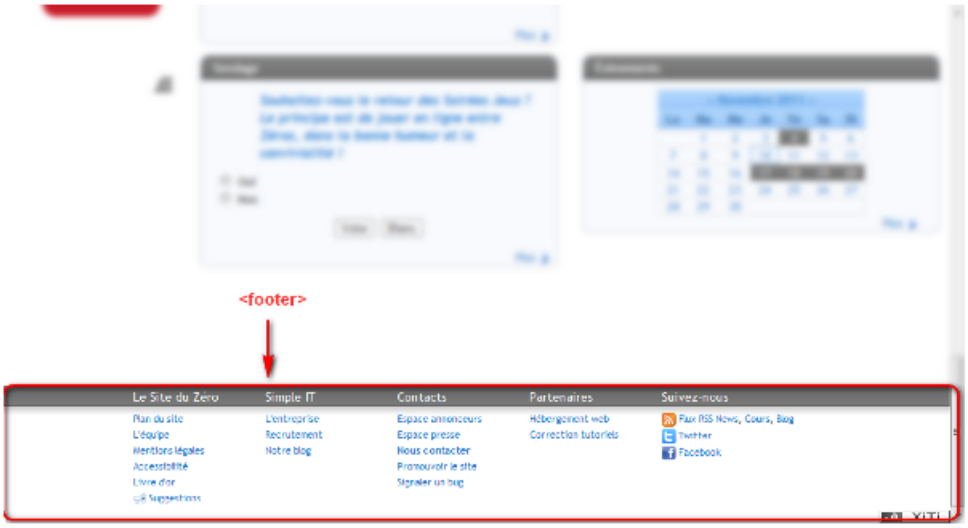


FIGURE 23 – Exemple de pied de page de page

11.2.3 Les liens de navigation

La balise **<nav>** doit regrouper tous les principaux liens de navigation du site. Nous y placerons par exemple le menu principal de notre site. Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise **<nav>** :

Exemple 42. *La balise <nav>*

```
<nav>
        <ul>
                <li><a href="index.html">Accueil</a></li>
                <li><a href="forum.html">Forum</a></li>
                <li><a href="contact.html">Contact</a></li>
        </ul>
</nav>
```

A la figure suivante, nous pouvons observer le menu principal de navigation de l'ancien site du Zéro.

FIGURE 24 – Exemple de navigation de page

11.2.4 La section de page

La balise <section> sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page. <section>

Exemple 43. *La balise <section>*

```
<section>
<h1>Ma section de page</h1>
<p>Mon texte y afférent</p>
</section>
```

Sur la page d'accueil du portail Free.fr à la figure de la page suivante, nous trouvons plusieurs blocs qui pourraient être considérés comme des sections de page.

11.2.5 Les informations complémentaires

La balise **<aside>** est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ce ne soit pas une obligation).

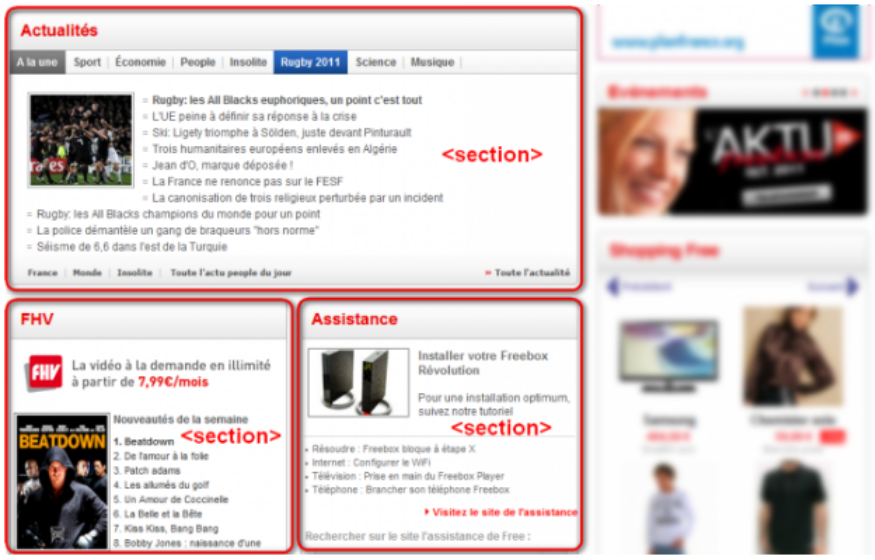


FIGURE 25 – Exemple de section de page

Exemple 44. *La balise <aside>*

```
<aside>
        <!-- Placez ici des informations complémentaires -->
</aside>
```

Il peut y avoir plusieurs blocs <aside> dans la page.

Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que nous visualisons. Ainsi, sur la page présentant la planète Saturne à la figure de la page suivante, nous trouvons dans ce bloc les caractéristiques de la planète (dimensions, masse, etc.).

11.2.6 L'article indépendant

La balise **<article>** sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

Exemple 45. *La balise <article>*

```
<article>
    <h1>Mon article</h1>
    <p>Mon texte s'y afférent</p>
</article>
```

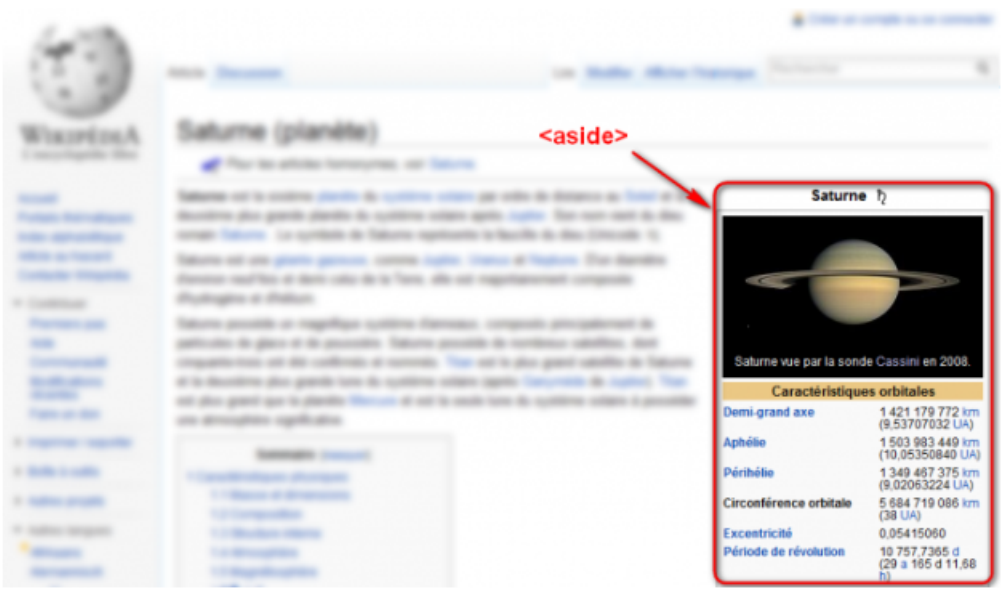


FIGURE 26 – Exemple d'informations complémentaires de page

FIGURE 27 – Exemple de d'article de page

Par exemple, voici un article issu de l'ancien site du Zéro à la figure suivante. Nous pouvons avec ces 6 balises structurer de manière plus détaillée notre page.

Prenons l'exemple de la figure suivante afin d'illustrer l'harmonie de ces balises.

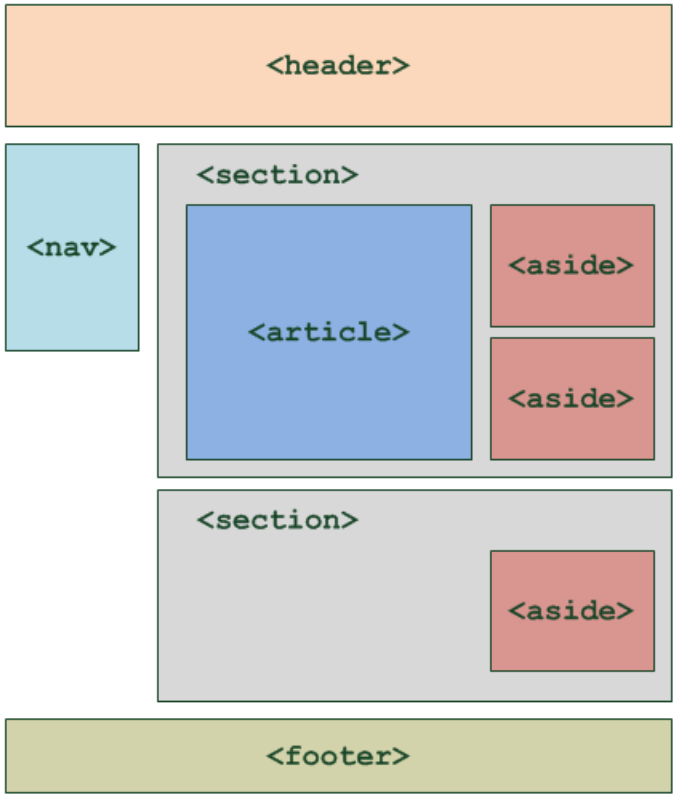


FIGURE 28 – Exemple de structure de page

Astuce 22 (La compatibilité avec IE).

Sans rentrer dans le détail, sachons que <!–[if lt IE 9]> est un commentaire conditionnel. C'est un commentaire spécial qui n'est lu que par Internet Explorer. Il permet de faire en sorte que le script s'exécute uniquement sur les versions d'Internet Explorer inférieures à IE9 (les autres navigateurs n'en ont pas besoin et ignoreront le commentaire).
Par exemple :
<!–[if lt IE 9]>
<script src="http ://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]–>
HTML5shiv permet de faire en sorte que les balises que nous venons de voir s'affichent correctement sur les anciennes versions d'Internet Explorer.

Voici un exemple de structure d'une page en HTML 5 :

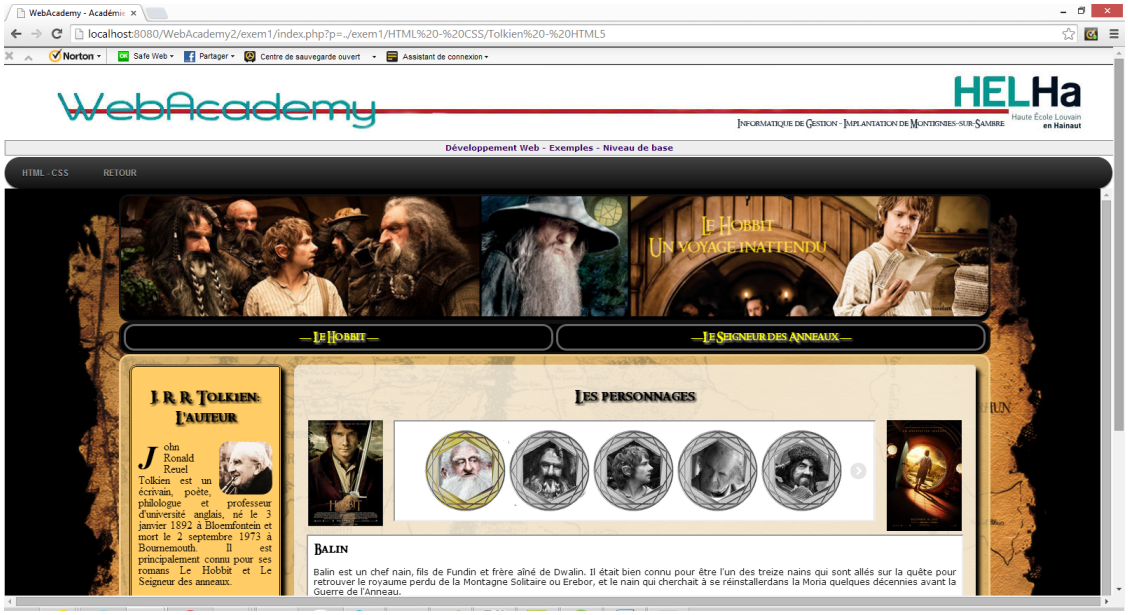


FIGURE 29 – Exemple de structure d'une page en HTML 5

CHAPITRE 12

LE MODÈLE DES BOÎTES

Dans ce chapitre nous allons apprendre à dimensionner nos balises. Pour rappel, nous avons vu jusque maintenant deux familles de balises : block et inline. Notre étude actuelle de ces familles portera sur les dimensions, les marges et l'ajustement à apporter à celles-ci. Pour une question de simplicité, nous utiliserons l'appellation francophone : bloc au lieu de block.

12.1 LES DIMENSIONS

Les balises de type bloc possède deux dimensions :
- **width** : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).
- **height** : c'est la hauteur du bloc. Nous l'exprimons soit en pixels (px), soit en pourcentage (%).

12.1.0.1 Les dimensions standards

Par défaut, un bloc prend 100% de la largeur disponible de son conteneur nonobstant ses marges ou celles de son conteneur.

Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution d'écran du visiteur. Toutefois, il se peut que nous ayons besoin de créer des blocs ayant une dimension précise en pixels. Peu importe notre choix, ses propriétés doivent figurer dans une feuille de style.

12.1.0.2 Les dimensions minimales et maximales

Nous pouvons fixer des dimensions minimales et maximales à un bloc. Cela est très pratique car nous avons la possibilité de définir des dimensions « limites » pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs.

Les propriétés sont :
- **min-width** : largeur minimale ;
- **min-height** : hauteur minimale ;
- **max-width** : largeur maximale ;
- **max-height** : hauteur maximale.

A l'exemple suivant, nous pouvons constater que les paragraphes occupent 50% de la largeur et qu'ils ont au moins 400 pixels de large en cas de résolutions trop petites.

Exemple 46. *Les dimensions limites*

```
p
{
    width: 50%;
    min-width: 400px;
}
```

12.2 LES MARGES

Nous devons savoir que tous les blocs possèdent des marges et qu'il existe deux types de marges :
- les **marges intérieures** ;
- les **marges extérieures**.

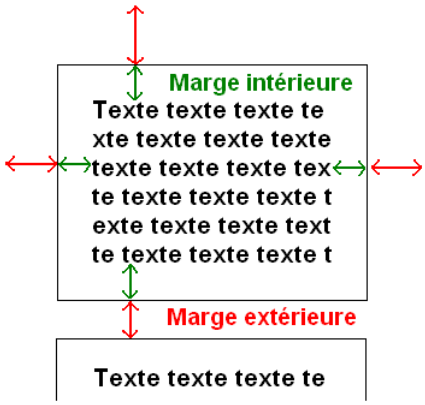


FIGURE 30 – Exemple de marges

Sur le bloc de la figure précédente, nous remarquons une bordure pour mieux repérer ses frontières.

L'espace entre le texte et la bordure est la marge intérieure (en vert).

L'espace entre la bordure et le bloc suivant est la marge extérieure (en rouge).

En CSS, nous pouvons modifier la taille des marges avec les deux propriétés suivantes :
– **padding** : indique la taille de la marge intérieure.
– **margin** : indique la taille de la marge extérieure.

Les balises de type inline possèdent également des marges. Nous pouvons donc aussi essayer ces manipulations sur ce type de balises.

Nous pouvons peaufiner notre travail en appliquant une mesure qu'à certaines marges. En effet, pour cela nous pouvons utiliser les suffixes suivants :
– **-top** : haut ;
– **-bottom** : bas ;
– **-left** : gauche ;
– **-right** : droite.

Astuce 23 (Les autres notations de valeurs)**.**

Il y a d'autres façons de spécifier les marges avec les propriétés margin et padding. Par exemple : **margin** : **2px 0 3px 1px** ; signifie « 2 px de marge en haut, 0 px à droite (le px est facultatif dans ce cas), 3 px en bas, 1 px à gauche ».
Autre notation raccourcie : **margin** : **2px 1px** ; signifie « 2 px de marge en haut et en bas, 1 px de marge à gauche et à droite ».

Astuce 24 (Centrer des blocs)**.**

Il est tout à fait possible de centrer des blocs. Cela est même très pratique pour réaliser un design centré quand nous ne connaissons pas la résolution du visiteur.
Pour centrer, il faut respecter les règles suivantes :
– donnons une largeur au bloc avec la propriété **width** ;
– indiquons que nous voulons des marges extérieures (marge(s) opposée(s)) automatiques, comme ceci : **margin** : **auto** ;.

12.3 L'AJUSTEMENT

Lorsque nous commençons à définir des dimensions précises pour nos blocs, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que nous allons voir ici ont justement été créées pour contrôler les dépassements... et décider du comportement à adopter le cas échéant.

12.3.1 La coupure de blocs

Si nous voulons que le texte ne dépasse pas les limites du bloc, il va falloir utiliser la propriété **overflow**.

Voici les valeurs qu'elle peut accepter :

- **visible** (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- **hidden** : si le texte dépasse les limites, il sera tout simplement coupé. Nous ne pourrons pas voir tout le texte.
- **scroll** : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour que nous puissions lire l'ensemble du texte.
- **auto** : c'est le mode « pilote automatique ». C'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire).

12.3.2 La coupure de mots

Si nous devons placer un mot très long dans un bloc, qui ne tient pas dans la largeur de ce bloc, nous allons pouvoir utiliser la propriété **word-wrap**. Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

Cette propriété sera couplée avec la valeur **break-word** qui forcera la césure du texte.

Bien entendu, il existe d'autres propriétés comme :

Cette propriété est souvent utilisée dans les forums.

Astuce 25 (Couper les mots en fonction de la langue)**.**

Certains navigateurs sont capables de couper un mot en fonction de ses syllabes. Pour cela, nous devons ajouter à la balise **<html>** l'attribut **lang** avec la valeur **fr**.

CHAPITRE 13

LE POSITIONNEMENT

Nous nous approchons tout doucement de notre objectif : après l'étude de ce chapitre, nous aurons toutes les notions nécessaires pour construire un site statique.

Cependant, il nous reste à voir les règles de positionnement. Celles-ci sont primordiales pour le design de notre site.

13.1 LE POSITIONNEMENT AVEC LES FLOTTANTS

Nous avons déjà abordé ce sujet lors du chapitre traitant des images. Les règles vues sont les mêmes pour des balises de type bloc. Souvenons-nous que le texte excédant l'élément flottant est placé en dessous de lui par défaut. Pour aligner les blocs, nous devrons utiliser les marges vues précédemment.

13.2 LE POSITIONNEMENT PAR AFFICHAGE

Il existe en CSS une propriété très puissante : **display**. Elle est capable de transformer n'importe quel élément de notre page d'un type vers un autre. Avec cette propriété, nous pouvons par exemple imposer à nos liens (originellement de type inline) d'apparaître sous forme de blocs.

Voici un liste non exhaustive des valeurs possibles :

- **inline** : éléments d'une ligne qui se placent les uns à côté des autres. Ex : **<a>**, ****, ****...
- **block** : éléments en forme de blocs qui se placent les uns en-dessous des autres et peuvent être dimensionnés. Ex : **<p>**, ** **, **<section>**...
- **inline-block** : éléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être dimensionnés (comme les blocs) (pratique pour la création de

menu déroulant). Ex : <select> et <input> (voir chapitre sur les formulaires).
- **none** : éléments non affichés. Ex : **<head>**...
- **list-item** : éléments qui se comportent comme une balise **<li>** (pratique pour la création de menu déroulant).
- ...

Étant donné que nous pouvons travailler avec des blocs, il serait parfois judicieux d'utiliser un alignement vertical. En effet, les blocs peuvent ne pas avoir la même hauteur et sont alignés, avec la propriété **inline-block**, les uns à côté des autres sur une ligne commune appelée : **baseline**.

Le positionnement vertical utilise la propriété **vertical-align** et prend comme valeur :
- **baseline** : aligne de la base de l'élément avec celle de l'élément parent (par défaut) ;
- **top** : aligne en haut ;
- **middle** : centre verticalement ;
- **bottom** : aligne en bas ;
- **valeur en px ou %** : aligne à une certaine distance de la ligne de base (baseline).

Voici un exemple, à la figure suivante, d'un menu construit à partir des valeurs **inline-block** et **list-item**.

FIGURE 31 – Exemple d'un menu

Les éléments du menu principal sont organisés avec la valeur **inline-block** tandis que les sous-éléments et leurs descendants sont affichés avec la valeur **list-item**.

13.3 LES POSITIONNEMENTS CLASSIQUES

Il faut d'abord faire son choix entre les trois modes de positionnement disponibles. Pour cela, nous utilisons la propriété **position** à laquelle nous donnons une de ces valeurs :
- **absolute** : positionnement absolu ;
- **fixed** : positionnement fixe ;
- **relative** : positionnement relatif.

13.3.1 Le positionnement absolu

Le positionnement absolu permet de placer un élément (réellement) n'importe où sur la page à partir du point d'origine de celle-ci.

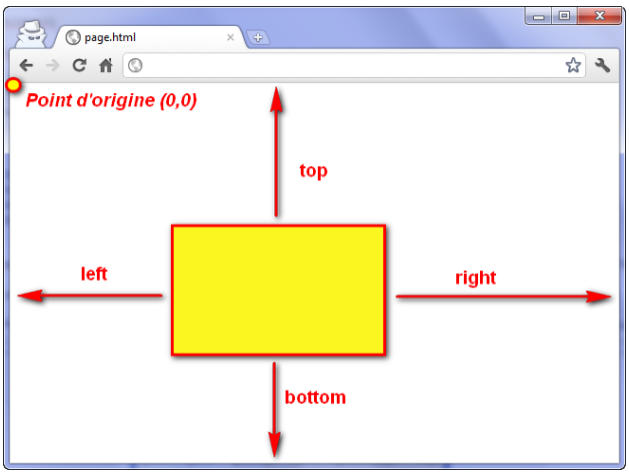


FIGURE 32 – Le point d'origine d'une page et positionnement d'un élément en absolu

Nous constatons que nous disposons de 4 valeurs pour ajuster le positionnement de l'élément :
- **left** : position par rapport à la gauche de la page ;
- **right** : position par rapport à la droite de la page ;
- **top** : position par rapport au haut de la page ;
- **bottom** : position par rapport au bas de la page.

Les éléments positionnés en absolu sont placés par-dessus le reste des éléments de la page. Par ailleurs, si nous plaçons deux éléments en absolu vers le même endroit, ils risquent de se chevaucher. Dans ce cas, utilisons la propriété **z-index** pour indiquer quel élément doit apparaître au-dessus des autres.

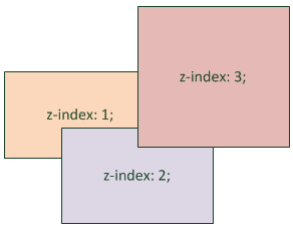


FIGURE 33 – Le chevauchement d'éléments

Nous constatons à la figure précédente que plus la valeur de la propriété est élevée, au plus l'élément est en avant-plan.

13.3.2 Le positionnement fixe

Le principe est exactement le même que pour le positionnement absolu sauf que, cette fois, le bloc reste fixe à sa position, même si on descend plus bas dans la page.

Cette technique est pratique pour conserver un menu à portée de main ou encore fixé un image en arrière-plan comme à la figure 29.

13.3.3 Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir difficile à utiliser. Ce positionnement permet d'effectuer des « ajustements » : l'élément est décalé par rapport à sa position initiale.

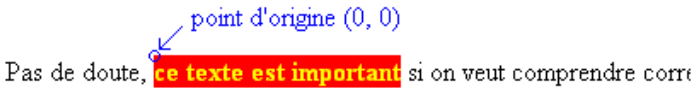


FIGURE 34 – Le point d'origine d'un élément

Appliquons les modifications de l'exemple suivant :

Exemple 47. *Le positionnement relatif*

```
strong
{
   background-color: red;
   color: yellow;
   position: relative;
   left: 55px;
   top: 10px;
}
```

Ce qui nous donne :

Le texte est décalé

FIGURE 35 – Positionnement relatif

Quatrième partie

Les fonctionnalités évoluées

CHAPITRE 14

LES TABLEAUX

14.1 LA NOTION DE TABLEAU

Nous pouvons commencer ce chapitre en abordant la notion de tableau.

Définition 11 (Tableau)**.**

Un tableau est constitué de lignes (formation horizontale) et de colonnes (formation verticale) dont chaque intersection forme une cellule.

Ce tableau est défini par la balise **<table></table>**. Au sein de ce tableau, chaque ligne est renseignée par **<tr></tr>** où tr signifie "table row". Pour marquer une intersection, donc une cellule, nous utilisons une balise **<td></td>** où td signifie "table data".

Nous pouvons représenter un tableau comme à la figure suivante.

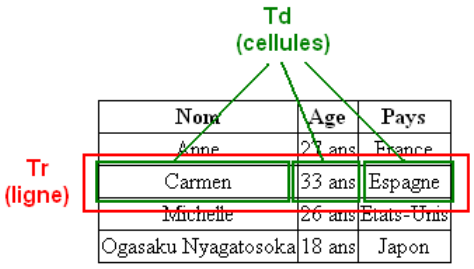


FIGURE 36 – Exemple de tableau

Nous pouvons retrouver l'équivalent HTML :

Exemple 48. *Un tableau*

```
...
<body>
        <table>
                <tr>
                        <td>Nom</td>
                        <td>Age</td>
                        <td>Pays</td>
                </tr>
                <tr>
                        <td>Anne</td>
                        <td>27 ans</td>
                        <td>France</td>
                </tr>
                <tr>
                        <td>Carmen</td>
                        <td>33 ans</td>
                        <td>Espagne</td>
                </tr>
                <tr>
                        <td>Michelle</td>
                        <td>26 ans</td>
                        <td>Etats-Unis</td>
                </tr>
                <tr>
                        <td>Ogazsaku Nyagatosoka</td>
                        <td>18 ans</td>
                        <td>Japon</td>
                </tr>
        </table>
</body>
...
```

Le contenu des cellules ne se limite pas qu'au texte. En effet, les cellules peuvent contenir tous les éléments HTML déjà passés en revue soit :

- du texte
- des images
- des liens
- des arrière-plans
- ...
- et même des tableaux !

14.2 LE TABLEAU SIMPLE

Ce tableau est constitué des trois éléments vus précédemment : **<table>**, **<tr>** et **<td>**. Nous pouvons également :

- gérer les espacements entre les bordures et les cellules ;
- ajouter une ligne d'en-tête ;
- ajouter un titre ;
- fusionner des cellules ;
- aligner les valeurs ;
- dimensionner les lignes et les colonnes.

14.2.1 La gestion des espacements

Celle-ci se gère par les propriétés CSS **border-spacing**, **padding**, et **border-collapse**.

Nous avons déjà étudier la propriété CSS **padding** précédemment à la section 12.2.

La propriété **border-spacing** va gérer l'espace entre la bordure du tableau et celle de la cellule. Elle reprend les valeurs de la propriété **spacing**.

La propriété **border-collapse** va permettre de coller les bordures internes des cellules. Les valeurs possibles sont :

- **collapse** : les bordures seront collées entre elles ;
- **separate** : les bordures seront dissociées (valeur par défaut)

Les propriétés **border-spacing** et **border-collapse** interagissent entre elles. Donc, si la propriété **border-collapse** à la valeur collapse, il est impossible de pouvoir préciser, en même temps, un espacement avec la propriété **border-spacing** et inversement.

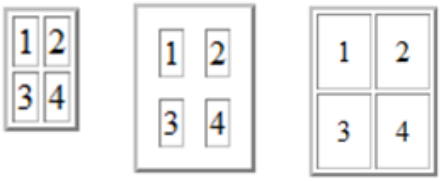


FIGURE 37 – Exemple de jeux d'espacements

14.2.2 La ligne d'en-tête

Il est possible de mentionner des cellules comme en-tête du tableau. Pour cela deux conditions doivent être respectées :

- les cellules d'en-tête doivent être comprises dans la **première balise <tr>** du tableau ;

– les cellules d'en-tête doivent être identifiées par une balise **<th>**, où th signifie "table header", au lieu d'une balise **<td>**.

14.2.3 Le titre

Nous avons la possibilité de documenter un tableau grâce à la balise **<caption></caption>**. La position du titre peut être définie par la propriété CSS /attributcaption-side qui possède les valeurs :

– **top** : le titre est placé au-dessus du tableau (par défaut) ;
– **bottom** : le titre est placé en dessous du tableau.

14.2.4 La fusion de cellules

Le principe de la fusion, qu'elle soit verticale ou horizontale, est de créer une cellule à partir d'au moins deux cellules successives.

La fusion horizontale est effectuée par l'attribut **colspan** auquel on renseigne le nombre de cellules à fusionner horizontalement à partir de sa déclaration. Le même principe est utilisé pour la fusion verticale en utilisant l'attribut **rowspan**.

cellule 1		
cellule 1	cel 2	3

cellule 1	cel 2	3
	cel 2	3

FIGURE 38 – Exemple de fusions

14.2.5 L'alignement des valeurs

Il est possible d'aligner les valeurs des cellules de manières horizontale et verticale.

Pour rappel, l'alignement horizontal s'effectue via la propriété CSS**text-align** et l'alignement vertical s'effectue via la propriété CSS **vertical-align**.

14.2.6 Les dimensions d'une cellule

Les deux dimensions possibles sont les propriétés CSS **width** pour la largeur et **height** pour la hauteur. Nous avons déjà vu ces propriétés précédemment.

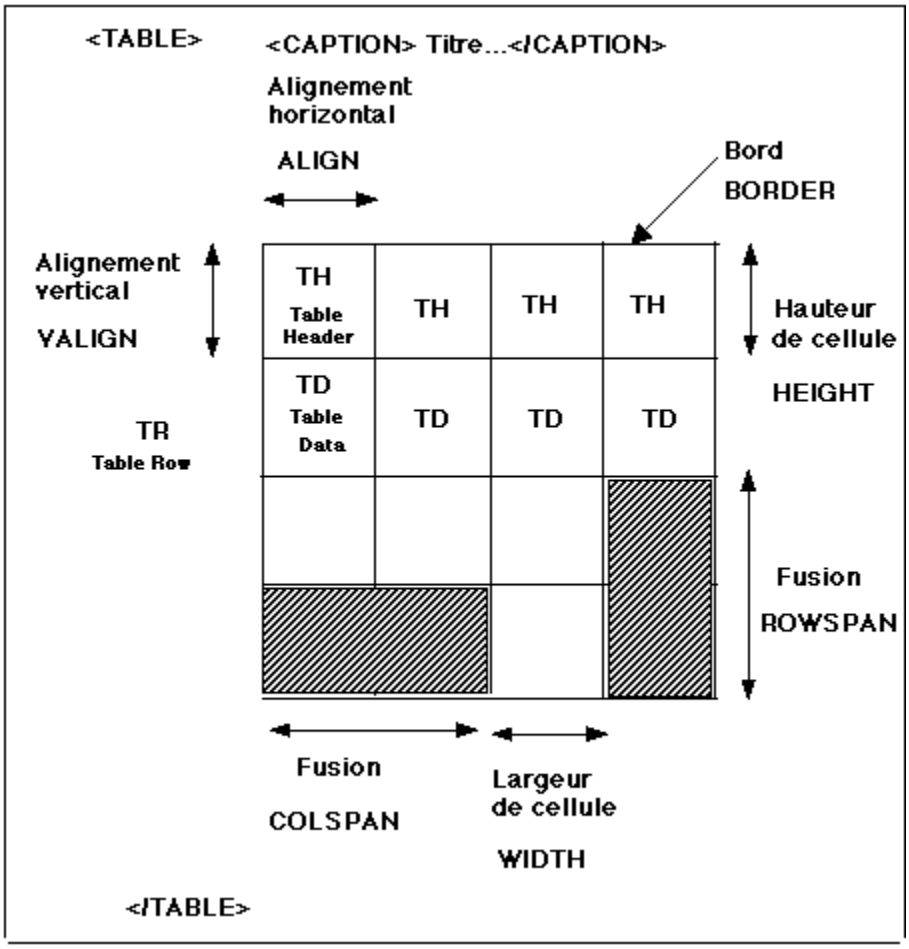


FIGURE 39 – Résumé de la structure d'un tableau simple

14.3 LE TABLEAU STRUCTURÉ

L'idée de structurer un tableau résulte de la taille de celui-ci. Le but est de placer des limites fictives dans le tableau comme présenté à la figure de la page suivante.

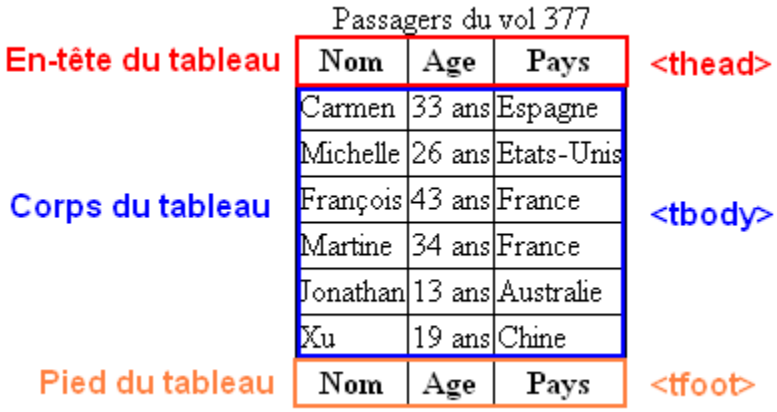


FIGURE 40 – Exemple d'un tableau structuré

Nous allons donc regrouper des lignes en trois catégories suivant les balises :
- **<theader></theader>** : zone d'en-tête ;
- **<tfoot></tfoot>** : zone de pied de page ;
- **<tbody></tbody>** : zone de corps.

Notons que cet ordre est fortement recommandé pour la compatibilité des navigateurs.

Nous retrouvons le code HTML relatif à la figure précédente :
Exemple 49. *Un tableau structuré*

```
...
<body>
<table>
   <caption>Passagers du vol 377</caption>

   <thead> <!-- En-tête du tableau -->
       <tr>
           <th>Nom</th>
           <th>Âge</th>
           <th>Pays</th>
       </tr>
   </thead>

   <tfoot> <!-- Pied de tableau -->
       <tr>
           <th>Nom</th>
           <th>Âge</th>
```

```
            <th>Pays</th>
        </tr>
    </tfoot>

    <tbody> <!-- Corps du tableau -->
        <tr>
            <td>Carmen</td>
            <td>33 ans</td>
            <td>Espagne</td>
        </tr>
        <tr>
            <td>Michelle</td>
            <td>26 ans</td>
            <td>États-Unis</td>
        </tr>
        <tr>
            <td>François</td>
            <td>43 ans</td>
            <td>France</td>
        </tr>
        <tr>
            <td>Martine</td>
            <td>34 ans</td>
            <td>France</td>
        </tr>
        <tr>
            <td>Jonathan</td>
            <td>13 ans</td>
            <td>Australie</td>
        </tr>
        <tr>
            <td>Xu</td>
            <td>19 ans</td>
            <td>Chine</td>
        </tr>
    </tbody>
</table>
</body>
...
```

CHAPITRE 15

LES FORMULAIRES

L'utilité d'un formulaire résulte dans le but de communiquer un ensemble d'informations à destination d'une ressource (autre page, base de données ...).

Deux problèmes fondamentaux se posent :

– Comment envoyer les informations ?

– Comment traiter les informations envoyées ?

Lors de la création du formulaire nous allons répondre à ces deux questions.

15.1 LA CRÉATION

La définition d'un formulaire s'opère par la balise **<form></form>**.

Nous allons répondre à nos deux questions laissées en suspens grâce à deux attributs.

15.1.1 La méthode d'envoi des données

L'attribut **method** permet de définir la méthode de transfert des données vers la ressource qui les traitera. Les deux valeurs possibles sont **get** et **post**.

La méthode "get" envoie les données via une url limitée à 255 caractères. Elle a une faille majeure au niveau de la sécurité. Cette faille sera analysée au cours de Technologies Internet.

La méthode "post" n'a aucune limite et envoie les données via un tableau associatif à la ressource visée pour le traitement de ces données.

15.1.2 Le traitement des données envoyées

L'attribut **action** indique l'action à exécuter lors de l'envoi des données. Nous pouvons donc envoyer les données à un serveur Web.

Exemple 50. *La déclaration d'un formulaire*

```
...
<body>
<p>Texte avant le formulaire</p>
<form method="post" action="traitement.php">
   <p>Texte à l'intérieur du formulaire</p>
</form>
<p>Texte après le formulaire</p>
</body>
...
```

15.2 Les éléments de saisie simples

HTML permet de saisir du texte dans un formulaire via deux zones de texte différentes :

- L'élément de **saisie de base** : il sert à saisir des textes courts, par exemple un nom, un pseudo...
- L'élément de **saisie multiligne** : c'est une zone de texte qui permet d'écrire une quantité importante de texte sur plusieurs lignes, par exemple une description, une annotation...

15.2.1 L'élément de saisie de base

Pour insérer une zone de texte dans une ligne, on utilise la balise **<input />** à laquelle on ajoute l'attribut **type** avec la valeur **text**.

Pour donner un nom à un élément de formulaire, on utilise l'attribut **name**. Il est possible de préfixer un élément de formulaire par une étiquette. Pour cela, on utilise la balise **<label></label>**. Pour identifier une étiquette, on utilise l'attribut **for**.

Exemple 51. *L'élément de saisie de base*

```
...
<body>
<form method="post" action="traitement.php">
   <label for="nom">Nom</label>:<input type="text" name="nom"/>
</form>
</body>
...
```

FIGURE 41 – Exemple d'un élément de saisie

On peut ajouter un certain nombre d'autres attributs à la balise <input /> pour personnaliser son fonctionnement :
- On peut adapter la longueur du champ avec l'attribut **size**.
- On peut limiter le nombre de caractères que l'on peut saisir avec l'attribut **maxlength**.
- On peut pré-remplir le champ avec une valeur par défaut à l'aide de l'attribut **value**.
- On peut donner une indication sur le contenu du champ avec l'attribut **placeholder**. Cette indication disparaît dès que le visiteur clique à l'intérieur du champ.

15.2.2 L'élément mot de passe

Nous pouvons facilement faire en sorte que la zone de texte se comporte comme une « zone de mot de passe », c'est-à-dire : une zone où les caractères saisis sont remplacer par une étoile (le caractère peut varier en fonction du navigateur utilisé). Pour créer ce type de zone de saisie, nous devons donner la valeur **password** à l'attribut **type**.

Exemple 52. *L'élément mot de passe*

```
...
<body>
<form method="post" action="traitement.php">
   <label for="login">Login</label>:<input type="text" name="login"/>
   <label for="pwd">Pwd</label>:<input type="password" name="pwd"/>
</form>
</body>
...
```

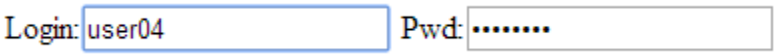


FIGURE 42 – Exemple d'un élément mot de passe

15.2.3 L'élément de saisie multiligne

Pour créer ce type d'élément, nous allons utiliser la balise **<textarea></textarea>**. Nous pouvons modifier la taille de l'élément **<textarea>** de deux façons différentes :

- En CSS : il suffit d'appliquer les propriétés CSS **width** et **height** à l'identifiant du **<textarea>**.
- En HTML avec des attributs : on peut ajouter les attributs **rows** et **cols** à la balise **<textarea>**. Le premier indique le nombre de lignes de texte qui peuvent être affichées simultanément et le second le nombre de colonnes.

Nous pouvons pré-remplir la balise **<textarea>** avec une valeur par défaut. Dans ce cas, nous n'utilisons pas l'attribut **value** : nous écrivons tout simplement le texte par défaut entre la balise ouvrante et la balise fermante.

Exemple 53. *L'élément textarea*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="remarque">Remarque</label>:<br/>
5         <textarea name="remarque" rows="10" cols="50">Nos factures
              sont payables au grand comptant.</textarea>
6 </form>
7 </body>
8 ...
```

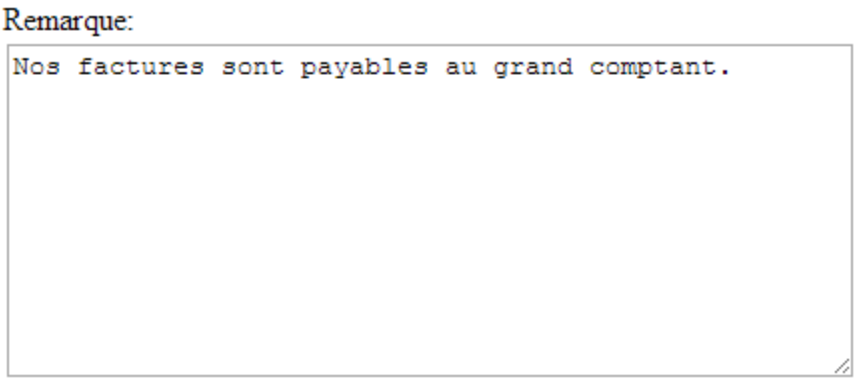


FIGURE 43 – Exemple d'un élément textarea

15.3 Les éléments de saisie évolués

HTML5 apporte de nombreuses nouvelles fonctionnalités relatives aux formulaires. De nouveaux types de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut **type** de la balise **<input />** l'une des nouvelles valeurs disponibles.

Astuce 26 (Comptabilité avec les navigateurs).

Tous les navigateurs peuvent ne pas connaître ces zones de saisie évoluées. À leur place, les anciennes versions des navigateurs afficheront une simple zone de saisie monoligne (comme si on avait écrit **type="text"**). Les nouveaux navigateurs peuvent profiter des dernières fonctionnalités, tandis que les anciens affichent une zone de texte de remplacement qui convient tout aussi bien.

15.3.1 L'élément e-mail

Nous pouvons demander à saisir une adresse e-mail.
Le champ vous semblera a priori identique mais votre navigateur sait désormais que l'utilisateur doit saisir une adresse e-mail. Il peut afficher une indication si l'adresse n'est pas un e-mail, c'est ce que fait Firefox par exemple.

Exemple 54. *L'élément e-mail*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="email">E-mail</label>:<input type="email" />
5 </form>
6 </body>
7 ...
```

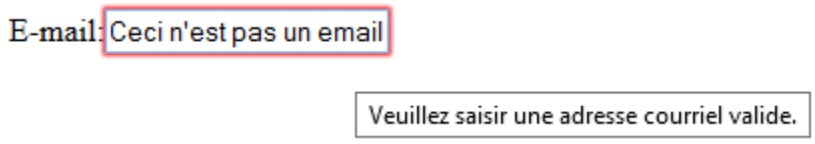


FIGURE 44 – Exemple d'un élément e-mail

15.3.2 L'élément url

Avec le type **url**, on peut demander à saisir une adresse absolue (commençant généralement par http ://). Si le champ ne nous semble pas différent sur notre navigateur, sachons que celui-ci comprend bel et bien que le visiteur est censé saisir une URL. Les navigateurs mobiles affichent par exemple un clavier adapté à la saisie d'URL.

Exemple 55. *L'élément url*

```
...
<body>
<form method="post" action="traitement.php">
        <label for="url">URL</label>:<input type="url" />
</form>
</body>
...
```

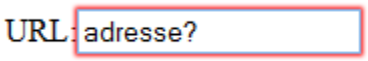


FIGURE 45 – Exemple d'un élément url

15.3.3 L'élément n^o de téléphone

Ce champ est dédié à la saisie de numéros de téléphone. Sur iPhone, par exemple, un clavier adapté s'affiche lorsqu'on doit remplir le champ.

Exemple 56. *L'élément tel*

```
...
<body>
<form method="post" action="traitement.php">
        <label for="gsm">GSM</label>:<input type="tel" />
</form>
</body>
...
```

15.3.4 L'élément nombre

Ce champ permet de saisir un nombre entier. Le champ s'affiche en général avec des petites flèches pour changer sa valeur.

Exemple 57. *L'élément number*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="NbEnfants">Nb Enfants</label>:<input type="number
              " />
5 </form>
6 </body>
7 ...
```

FIGURE 46 – Exemple d'un élément number

Nous pouvons personnaliser le fonctionnement du champ avec les attributs suivants :
- **min** : valeur minimale autorisée.
- **max** : valeur maximale autorisée.
- **step** : c'est le « pas » de déplacement. Si nous indiquons un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

15.3.5 L'élément curseur

Le type range permet de sélectionner un nombre avec un curseur (aussi appelé slider). Nous pouvons utiliser là aussi les attributs **min**, **max** et **step** pour restreindre les valeurs disponibles.

Exemple 58. *L'élément range*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="valeur">Valeur</label>:<input type="range" min="0
              " max="100" step="5" />
5 </form>
6 </body>
7 ...
```

FIGURE 47 – Exemple d'un élément range

15.3.6 L'élément couleur

Ce champ permet de saisir une couleur. En pratique, il reste assez peu mis en œuvre par les navigateurs à l'heure actuelle. Ne nous étonnons pas si nous voyons seulement un champ de texte classique.

Exemple 59. *L'élément color*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="couleur">Couleur</label>:<input type="color" />
5 </form>
6 </body>
7 ...
```

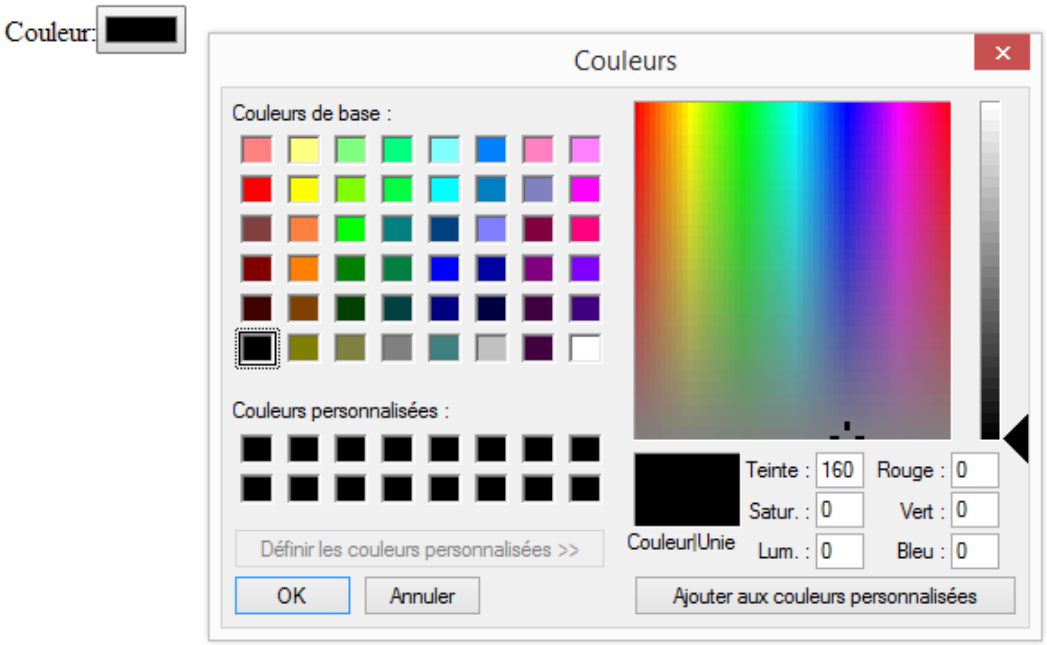


FIGURE 48 – Exemple d'un élément color

15.3.7 L'élément date

Différents types de champs de sélection de date existent :
- **date** : pour la date (05/08/1985 par exemple) ;
- **time** : pour l'heure (13 :37 par exemple) ;
- **week** : pour la semaine ;
- **month** : pour le mois ;
- **datetime** : pour la date et l'heure (avec gestion du décalage horaire) ;
- **datetime-local** pour la date et l'heure (sans gestion du décalage horaire).

Exemple 60. *L'élément date*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4         <label for="birthday">Date de naissance</label>:<input type="
          date" />
5 </form>
6 </body>
7 ...
```

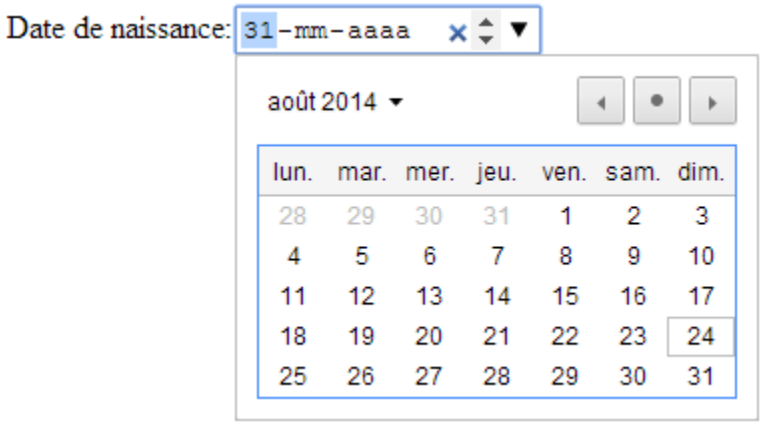


FIGURE 49 – Exemple d'un élément date

15.3.8 L'élément recherche

Le navigateur décide de l'affichage du champ de recherche. Ainsi, il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche et éventuellement mémoriser les dernières recherches effectuées par le visiteur.

Exemple 61. *L'élément search*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4 <label for="recherche">Recherche</label>:<input type="search" />
5 </form>
6 </body>
7 ...
```

15.4 LES ÉLÉMENTS D'OPTIONS

HTML nous offre une gamme d'éléments d'options à utiliser dans notre formulaire. Ce sont des éléments qui demandent au visiteur de faire un choix parmi une liste de possibilités.

Nous allons passer en revue :
- les cases à cocher ;
- les zones d'options ;
- les listes déroulantes.

15.4.1 L'élément à cocher

Nous allons réutiliser la balise **<input />**, en spécifiant cette fois à l'attribut **type** la valeur **checkbox**. Nous devons savoir que nous pouvons faire en sorte qu'une case soit cochée par défaut avec l'attribut **checked** et la valeur **checked**.

Exemple 62. *L'élément à cocher*

```
1 ...
2 <body>
3 <form method="post" action="traitement.php">
4 <p>
5     <label for="series">Cochez les séries que vous aimez regarder :</
          label><br />
6     <input type="checkbox" name="walkingDead" checked="checked" /> <
          label for="vWalkingDead">The walking dead</label><br />
7     <input type="checkbox" name="onceUponATime" /> <label for="
          vOnceUponATime">Once upon a time</label><br />
```

```
8      <input type="checkbox" name="strain" /> <label for="epinards">The
           strain</label><br />
9      <input type="checkbox" name="teenWolf"  /> <label for="vTeenWolf"
           >Teen wolf</label> <br />
10     <input type="checkbox" name="gameOfThrones" /> <label for="
           vGameOfThrones">Game of thrones</label>
11 </p>
12 </form>
13 </body>
14 ...
```

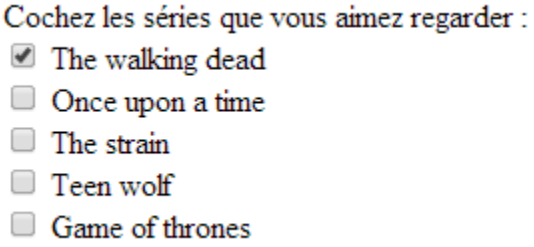


FIGURE 50 – Exemple d'un élément à cocher

15.4.2 L'élément de zone à options

Les zones d'options nous permettent de faire un seul choix parmi une liste de possibilités. Elles doivent être organisées en groupes. Les options d'un même groupe possèdent le même nom (**name**), mais chaque option doit avoir une valeur (**value**) différente.

La balise à utiliser est toujours **<input />** avec cette fois la valeur **radio** pour l'attribut **type**.

L'attribut **checked** est disponible pour sélectionner une valeur par défaut.

Exemple 63. *L'élément de zone à options*

```
1  ...
2  <body>
3  <form method="post" action="traitement.php">
4  <p>
5         Sur quel continent habitez-vous ?<br />
6         <input type="radio" name="continent" value="europe" /> <label
              for="europe">Europe</label><br />
7         <input type="radio" name="continent" value="afrique" /> <label
              for="afrique">Afrique</label><br />
8         <input type="radio" name="continent" value="asie" /> <label
              for="asie">Asie</label><br />
```

```
9           <input type="radio" name="continent" value="amerique" /> <
               label for="amerique">Amérique</label><br />
10          <input type="radio" name="continent" value="oceanie" /> <label
                for="oceanie">Océanie</label><br />
11          <input type="radio" name="continent" value="antarctique" /> <
               label for="antarctique">Antarctique</label>
12  </p>
13  </form>
14  </body>
15  ...
```

Sur quel continent habitez-vous ?
- Europe
- Afrique
- Asie
- Amérique
- Océanie
- Antarctique

FIGURE 51 – Exemple d'un élément de zone à options

15.4.3 L'élément de sélection

Les listes déroulantes sont un autre moyen de faire un choix parmi plusieurs possibilités. Le fonctionnement est un peu différent, nous allons utiliser la balise **<select> </select>** qui indique le début et la fin de la liste déroulante. Nous y ajoutons l'attribut **name** à la balise pour donner un nom à la liste.

A l'intérieur de cette balise, nous allons placer plusieurs balises **<option> </option>** (une par choix possible). Nous ajoutons à chacune d'elles un attribut **value** pour pouvoir identifier ce que le visiteur a choisi.

Si nous voulons qu'une option soit sélectionnée par défaut, nous devons utiliser cette fois l'attribut **selected** et la valeur **selected**. Nous pouvons aussi grouper nos options avec la balise **<optgroup> </optgroup>**.

Exemple 64. *L'élément de sélection*

```
...
<body>
<form method="post" action="traitement.php">
   <p>
       <label for="pays">Dans quel pays habitez-vous ?</label><br />
       <select name="pays">
           <optgroup label="Europe">
               <option value="belgique">Belgique</option>
               <option value="france">France</option>
               <option value="espagne">Espagne</option>
               <option value="italie">Italie</option>
               <option value="allemagne">Allemagne</option>
               <option value="royaume-uni">Royaume-Uni</option>
           </optgroup>
           <optgroup label="Amérique">
               <option value="canada">Canada</option>
               <option value="etats-unis">Etats-Unis</option>
               <option value="colombie">Colombie</option>
           </optgroup>
           <optgroup label="Asie">
               <option value="chine">Chine</option>
               <option value="japon">Japon</option>
               <option value="coreeSud">Corée du Sud</option>
           </optgroup>
       </select>
   </p>
</form>
</body>
...
```

Il est possible d'effectuer une recherche multiple grâce à l'attribut **multiple**. L'affichage de la liste sera fixe mais il est possible d'afficher un nombre d'éléments visibles grâce à l'attribut **size**.

Exemple 65. *L'élément de sélection multiple*

```
...
<body>
<form method="post" action="traitement.php">
   <p>
       <label for="pays">Quelles sont vos marques préférées ?</label>
          <br />
        <select name="cars" multiple size="5">
          <option value="volvo">Volvo</option>
          <option value="saab">Saab</option>
          <option value="opel">Opel</option>
```

```
10            <option value="audi">Audi</option>
11            <option value="peugeot">Peugeot</option>
12            <option value="seat">Seat</option>
13          </select>
14      </p>
15  </form>
16  </body>
17  ...
```

FIGURE 52 – Exemple d'un élément de sélection

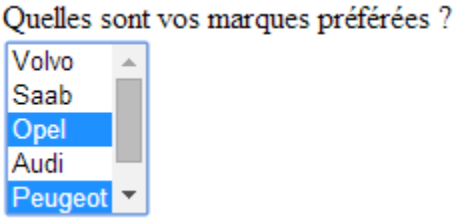


FIGURE 53 – Exemple d'un élément de sélection multiple

15.5 LA FINALITÉ ET L'ENVOI DU FORMULAIRE

Pour finaliser notre formulaire, il ne nous reste plus qu'à agrémenter celui-ci avec quelques fonctionnalités (comme la validation).

15.5.1 Le regroupement de champs

Si notre formulaire est conséquent car il comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises **<fieldset>**. Chaque balise **<fieldset>** peut contenir une légende avec la balise **<legend>**.

Exemple 66. *Le regroupement des champs*

```
...
<body>
<form method="post" action="traitement.php">
    <fieldset>
        <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->
        <label for="nom">Quel est votre nom ?</label>
        <input type="text" name="nom" id="nom" /> <br/>
        <label for="prenom">Quel est votre prénom ?</label>
        <input type="text" name="prenom" id="prenom" /> <br/>
        <label for="email">Quel est votre e-mail ?</label>
        <input type="email" name="email" id="email" />
    </fieldset>

    <fieldset>
        <legend>Votre souhait</legend> <!-- Titre du fieldset -->
        <p>
            Faites un souhait que vous voudriez voir exaucé :
            <input type="radio" name="souhait" value="riche" id="riche
                " /> <label for="riche">Etre riche</label>
            <input type="radio" name="souhait" value="celebre" id="
                celebre" /> <label for="celebre">Etre célèbre</label>
            <input type="radio" name="souhait" value="intelligent" id=
                "intelligent" /> <label for="intelligent">Etre <strong
                >encore</strong> plus intelligent</label>
            <input type="radio" name="souhait" value="autre" id="autre
                " /> <label for="autre">Autre...</label>
        </p>
        <p>
            <label for="precisions">Si "Autre", veuillez préciser :</
                label><br/>
            <textarea name="precisions" id="precisions" cols="40" rows
                ="4"></textarea>
```

```
26          </p>
27      </fieldset>
28  </form>
29  </body>
30  ...
```

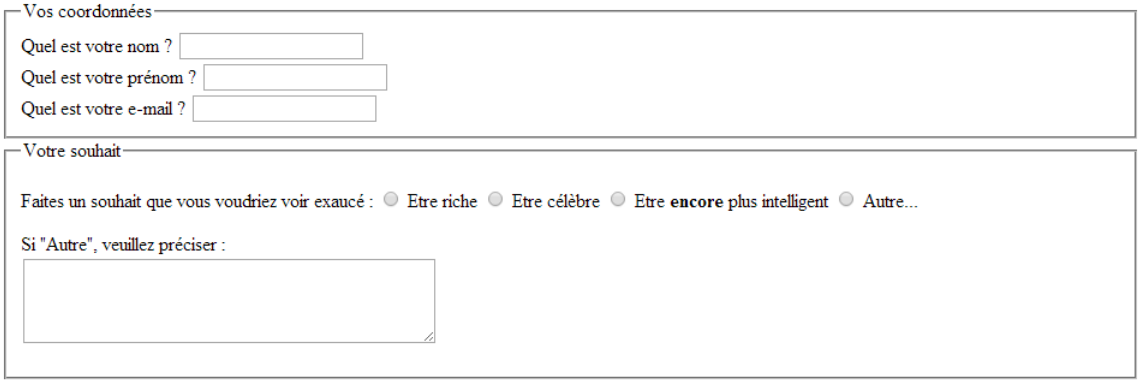


FIGURE 54 – Exemple d'un regroupement d'élément

15.5.2 La sélection automatique

Nous pouvons placer automatiquement le curseur dans l'un des champs de notre formulaire avec l'attribut **autofocus**. Dès que le visiteur chargera la page, le curseur se placera dans ce champ.

Exemple 67. *La sélection automatique*

```
1  ...
2  <body>
3  <form method="post" action="traitement.php">
4  <input type="text" name="prenom" autofocus />
5  </form>
6  </body>
7  ...
```

15.5.3 L'obligation de saisie d'un champ

Nous pouvons faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut **required**. Le navigateur indiquera alors au visiteur, si le champ est vide au moment de l'envoi, qu'il doit impérativement être rempli.

Exemple 68. *La sélection automatique*

```
...
<body>
<form method="post" action="traitement.php">
<input type="text" name="prenom" required />
</form>
</body>
...
```

Astuce 27 (Pseudo-formats)**.**

Nous disposons de pseudo-formats en CSS pour changer le style des éléments requis (**:required**) et invalides (**:invalid**). N'oublions pas non plus que nous disposons du pseudo-format **:focus** pour changer l'apparence d'un champ lorsque le curseur se trouve à l'intérieur.

15.5.4 L'élément bouton

Il ne nous reste plus qu'à créer le bouton d'envoi via la balise **<input />**. Il existe en quatre versions :

- **type="submit"** : c'est le principal bouton d'envoi de formulaire. C'est celui que nous allons utiliser le plus souvent car, grâce à lui, le visiteur est conduit à la page indiquée dans l'attribut **action** du formulaire.
- **type="reset"** : il permet la remise à zéro (à blanc) du formulaire.
- **type="image"** : c'est l'équivalent du bouton **submit** qui se présente sous forme d'image. Nous devons cependant ajouter l'attribut **src** pour indiquer l'URL de l'image.
- **type="button"** : Il s'agit d'un bouton générique, qui n'a (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page. Nous ne l'utiliserons pas dans ce cours.

On peut changer le texte affiché à l'intérieur des boutons avec l'attribut **value**.

Exemple 69. *L'élément bouton*

```
...
<body>
<form method="post" action="traitement.php">
<input type="submit" name="envoyer" value="Valider mes données!" /> <
    br />
<input type="image" name="logoIE" src="../images/logoIE.png" />
<input type="image" name="logoFF" src="../images/logoFF.png" />
<input type="image" name="logoGC" src="../images/logoGC.png" />
</form>
</body>
...
```

FIGURE 55 – Exemple d'éléments bouton

CHAPITRE 16

LA VIDÉO ET L'AUDIO

Depuis l'arrivée de Youtube et Dailymotion, il est devenu courant aujourd'hui de regarder des vidéos sur des sites web. Il faut dire que l'arrivée du haut débit a aidé à démocratiser les vidéos sur le Web.

16.1 LES FORMATS AUDIO ET VIDÉO

Le fonctionnement des vidéos est tellement complexe que nous pourrions faire un cours entier à ce sujet ! Étant donné que nous apprenons le HTML, nous n'allons pas nous borner à étudier les subtilités de l'encodage vidéo. Nous allons donc simplifier les choses et utiliser ce que nous avons réellement besoin.

16.1.1 Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF) ce qui permet de réduire leur poids :

- **MP3** : C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui.
- **AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème.
- **OGG** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet.
- **WAV** (format non compressé) : évitons autant que possible de l'utiliser car le fichier est très volumineux avec ce format. C'est un peu l'équivalent du Bitmap (BMP) pour l'image.

Aucun navigateur ne gère tous ces formats à la fois. Retenons surtout la compatibilité pour les MP3 et OGG :

Navigateur	MP3	OGG
Internet Explorer	Oui	-
Chrome	Oui	Oui
Firefox	Oui	Oui
Safari	Oui	-
Opera	-	Oui

FIGURE 56 – Compatibilité entre les formats audio (MP3/OGG) et les principaux navigateurs

16.1.2 Les formats vidéo

Le stockage de la vidéo est beaucoup plus complexe car nous avons besoin de trois éléments :

- Un **format conteneur** : c'est un genre de boîte qui va servir à contenir les deux éléments présentés ci-dessous. On le reconnaît en général à l'extension du fichier : AVI, MP4, MKV...
- Un **codec audio** : c'est le format du son de la vidéo qui est généralement compressé et qui fait partie de ceux que nous vus précédemment.
- Un **codec vidéo** : c'est le format qui compresse les images. Cependant ces formats sont complexes et nous ne pouvons pas toujours les utiliser gratuitement.
  Les principaux à connaître pour le Web sont :
  - **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui mais il n'est pas tout à fait gratuit. En fait, nous pouvons l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur un site web personnel), mais il y a un litige juridique qui fait qu'il est risqué de l'utiliser à profusion.
  - **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux mais, sous Windows, il faut installer des programmes supplémentaires pour pouvoir le lire.
  - **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux de H.264 à l'heure actuelle.

Navigateur	H.264	Ogg Theora	WebM
Internet Explorer	Oui	-	Oui
Chrome	-	Oui	Oui
Firefox	-	Oui	Oui
Safari	Oui	-	-
Opera	Oui	Oui	Oui

FIGURE 57 – Compatibilité entre les formats vidéo et les principaux navigateurs

Il est conseillé de proposer chaque vidéo dans plusieurs formats pour qu'elle soit lisible sur un maximum de navigateurs.

Astuce 28 (Outil de conversion vidéo)**.**

Pour convertir une vidéo dans ces différents formats, il existe l'excellent logiciel gratuit Miro Video Converter.

16.2 L'INSERTION D'UN ÉLÉMENT AUDIO

La balise **<audio></audio>** que nous allons découvrir est reconnue par tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

Exemple 70. *La balise audio*

```
1 ...
2 <body>
3 <audio src="musique.mp3"></audio>
4 </body>
5 ...
```

Si nous testons ce code... nous ne verrons rien ! En effet, le navigateur va seulement télécharger les informations générales sur le fichier (métadonnées) mais il ne se passera rien de particulier.

Nous pouvons compléter la balise des attributs suivants :

- **controls** : pour ajouter les boutons "Lecture", "Pause" et la barre de défilement. Cela peut sembler indispensable, et nous nous demandons peut-être pourquoi cela n'y figure pas par défaut. Certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript.
- **width** : pour modifier la largeur de l'outil de lecture audio.
- **loop** : pour jouer la musique en boucle.
- **autoplay** : pour jouer la musique dès le chargement de la page. Évitons d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !
- **preload** : indique si la musique peut être préchargée dès le chargement de la page ou non.

  Cet attribut peut prendre les valeurs :
  - ▷ **auto** (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
  - ▷ **metadata** : charge uniquement les métadonnées (durée, etc.).
  - ▷ **none** : pas de préchargement. Cela est utile si nous ne voulons pas gaspiller de bande passante sur votre site.

Astuce 29 (Le préchargement pour les navigateurs mobiles).

On ne peut pas forcer le préchargement de la musique, c'est toujours le navigateur qui décide.
Les navigateurs mobiles, par exemple, ne préchargent jamais la musique pour économiser la bande passante (le temps de chargement étant long sur un portable).

Exemple 71. *La balise audio visible*

```
1 ...
2 <body>
3 <audio src="musique\Indochine - Mao Boy.mp3" controls>Veuillez mettre
      à jour votre navigateur !</audio>
4 </body>
5 ...
```

Astuce 30 (L'utilisation du Flash).

Il est conseillé de proposer une solution de secours en Flash, comme Dewplayer. Nous placerons le code correspondant à Flash entre les balises **<audio>** et **</audio>** : ainsi, les anciens navigateurs afficheront le lecteur Flash, tandis que les nouveaux afficheront le lecteur natif.

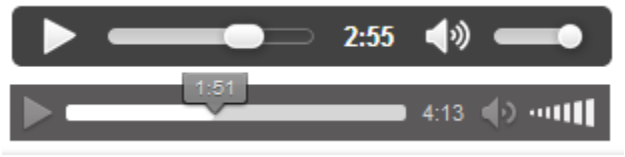


FIGURE 58 – Les lecteurs Audio de Google Chrome et FireFox

Nous avons vu que tous les navigateurs ne prennent pas en charge tous les formats, c'est pourquoi nous pouvons utiliser la balise **source** qui se place à l'intérieur de la balise **audio**.

Le navigateur prendra automatiquement le format qu'il reconnaît.

Exemple 72. *La balise source avec l'audio*

```
...
<body>
<audio controls>
        <source src="musique/Indochine - Mao Boy.mp3"></source>
        <source src="musique/Indochine - Mao Boy.ogg"></source>
</audio>
</body>
...
```

16.3 L'INSERTION D'UNE VIDÉO

Pour insérer une vidéo, il suffit d'une balise **<video></video>**.

Exemple 73. *La balise video*

```
...
<body>
<video src="video/webm"></video>
</body>
...
```

Cependant, nous risquons d'être déçus si nous utilisons seulement ce code car aucun contrôle ne permet de lancer la vidéo !

Nous allons ajouter quelques attributs (la plupart sont les mêmes que pour la balise **<audio>**) :

- **poster** : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, il est conseillé d'en créer une. Nous pouvons tout simplement faire une capture d'écran d'un moment de la vidéo.
- **controls** : pour ajouter les boutons "Lecture", "Pause" et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !
- **width** : pour modifier la largeur de la vidéo.
- **height** : pour modifier la hauteur de la vidéo.
- **loop** : la vidéo sera jouée en boucle.
- **autoplay** : la vidéo sera jouée dès le chargement de la page. Là encore, évitez d'en abuser, c'est en général irritant d'arriver sur un site qui lance quelque chose tout seul !
- **preload** : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - ▷ **auto** (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
  - ▷ **metadata** : charge uniquement les métadonnées (durée, dimensions, etc.).
  - ▷ **none** : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

On ne peut pas forcer le préchargement de la vidéo, c'est toujours le navigateur qui décide.

Les proportions de la vidéo sont toujours conservées. Si nous définissons une largeur et une hauteur, le navigateur fera en sorte de ne pas dépasser les dimensions indiquées mais il conservera les proportions.

Exemple 74. *La balise video avec image de lancement*

```
1 ...
2 <body>
3 <video src="videos/Le hobbit 3 BAVF.webm" controls poster="images/Le
    hobbit 3.jpg" width="600">
4 Il est temps de mettre à jour votre navigateur!
5 </video>
6 </body>
7 ...
```

FIGURE 59 – Le lecteur vidéo de Google Chrome

Nous utilisons la balise **<source>** à l'intérieur de la balise **<video>** pour proposer différents formats. Le navigateur prendra celui qu'il reconnaît.

Exemple 75. *La balise source avec une vidéo*

```
...
<body>
<video controls poster="images/Le hobbit 3.jpg" width="600">
        <source src="videos/Le hobbit 3 BAVF.mp4" ></source>
        <source src="videos/Le hobbit 3 BAVF.webm" ></source>
        <source src="videos/Le hobbit 3 BAVF.ogv" ></source>
</video>
</body>
...
```

Au niveau de la sécurité, les balises ne sont pas conçues pour limiter ou empêcher le téléchargement. C'est assez logique quand on y pense : pour que le visiteur puisse voir la vidéo, il faut bien de toute façon qu'il la télécharge d'une manière ou d'une autre.

CHAPITRE 17

LES MEDIA QUERIES

La première préoccupation des webmasters qui mettent en place le design de leur site est la résolution d'écran de leurs visiteurs. En effet, selon les écrans, la résolution peut varier et donc l'espace disponible aussi.

La solution à ce problème est d'utiliser les media queries. Ce sont des règles à appliquer pour changer le design d'un site en fonction des caractéristiques de l'écran du visiteur. Grâce à cette technique, nous pouvons créer un design qui s'adapte automatiquement à l'écran de chaque visiteur.

17.1 LA MISE EN PLACE

Les media queries font partie des nouveautés de CSS 3. Il ne s'agit pas de nouvelles propriétés mais de règles que l'on peut appliquer dans certaines conditions. Concrètement, nous allons pouvoir dire "si la résolution de l'écran du visiteur est inférieure à x, alors applique les propriétés CSS y". Cela nous permet de changer l'apparence du site dans certaines conditions : nous pouvons augmenter la taille du texte, changer la couleur de fond, positionner différemment notre menu dans certaines résolutions, etc.

Contrairement à ce que l'on pourrait penser, les media queries ne concernent pas que les résolutions d'écran. Nous pouvons changer l'apparence de notre site en fonction d'autres critères comme le type d'écran (smartphone, télévision, projecteur...), le nombre de couleurs, l'orientation de l'écran (portrait ou paysage), etc. Les possibilités sont très nombreuses !

17.1.1 L'application d'une media query

Il y a deux façons de l'utiliser :

– en chargeant une feuille de style .css différente en fonction de la règle (ex : "si la résolution est inférieure à 1280px de large, charge le fichier petiteResolution.css") ;

– en écrivant la règle directement dans le fichier .css habituel (ex : "si la résolution est inférieure à 1280px de large, charge les propriétés CSS ci-dessous").

17.1.2 Le chargement d'une feuille de style différente

Nous devons pour cela utiliser notre balise **<link />** et valoriser son attribut **media**, dans lequel, nous allons écrire la règle qui doit s'appliquer pour que le fichier soit chargé. On dit qu'on fait une "requête de media" (media query en anglais).

Exemple 76. *La media query avec l'attribut media*

```
<link rel="stylesheet" media="screen and (max-width: 1280px)" href="
    petiteResolution.css" />
```

Au final, notre code HTML pourrait proposer plusieurs fichiers CSS : un par défaut (qui est chargé dans tous les cas) et d'autres qui seront chargés en supplément uniquement si la règle correspondante s'applique.

Exemple 77. *Plusieurs media queries avec l'attribut media*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <!--Pour tout le monde-->
        <link rel="stylesheet" href="style.css" />
        <!--Pour ceux qui ont une résolution inférieure à 1280px-->
        <link rel="stylesheet" media="screen and (max-width: 1280px)"
            href="petiteResolution.css" />
        <title>Media queries</title>
    </head>
        <body>
        </body>
</html>
```

17.1.3 Le chargement des règles directement dans la feuille de style

Une autre technique consiste à écrire ces règles dans un même fichier CSS.

Exemple 78. *La media query dans un fichier CSS*

```
@media screen and (max-width: 1280px)
{
    /* Rédigeons nos propriétés CSS ici */
}
```

17.2 Les règles disponibles

Il existe de nombreuses règles permettant de construire des media queries.
Voici les principales :
- **color** : gestion de la couleur (en bits/pixel).
- **height** : hauteur de la zone d'affichage (fenêtre).
- **width** : largeur de la zone d'affichage (fenêtre).
- **device-height** : hauteur du périphérique.
- **device-width** : largeur du périphérique.
- **orientation** : orientation du périphérique (portrait ou paysage).
- **media** : type d'écran de sortie dont voici quelques-unes des valeurs possibles :
  - **screen** : écran « classique » ;
  - **handheld** : périphérique mobile ;
  - **print** : impression ;
  - **tv** : télévision ;
  - **projection** : projecteur ;
  - **all** : tous les types d'écran.

Nous pouvons ajouter le préfixe min- ou max- devant la plupart de ces règles. Ainsi, **min-width** signifie "largeur minimale", max-height "hauteur maximale", etc.

La différence entre **width** et **device-width** se perçoit surtout sur les navigateurs mobiles des smartphones.

Les règles peuvent être combinées à l'aide des mots suivants :
- **only** : "uniquement" ;
- **and** : "et" ;
- **not** : "non".

Exemple 79. *Ensemble de media queries*

```
1  /* Sur les écrans, quand la largeur de la fenêtre fait au maximum
       1280px */
2  @media screen and (max-width: 1280px)
3
4  /* Sur tous types d'écran, quand la largeur de la fenêtre est
       comprise entre 1024px et 1280px */
5  @media all and (min-width: 1024px) and (max-width: 1280px)
6
7  /* Sur les téléviseurs */
8  @media tv
9
10 /* Sur tous types d'écran orientés verticalement */
11 @media all and (orientation: portrait)
```

Pour les anciens navigateurs, nous utilisons le mot-clé **only** car ces vieilles versions ne le connaissent pas contrairement à **media** (ex : **@media only tv**).

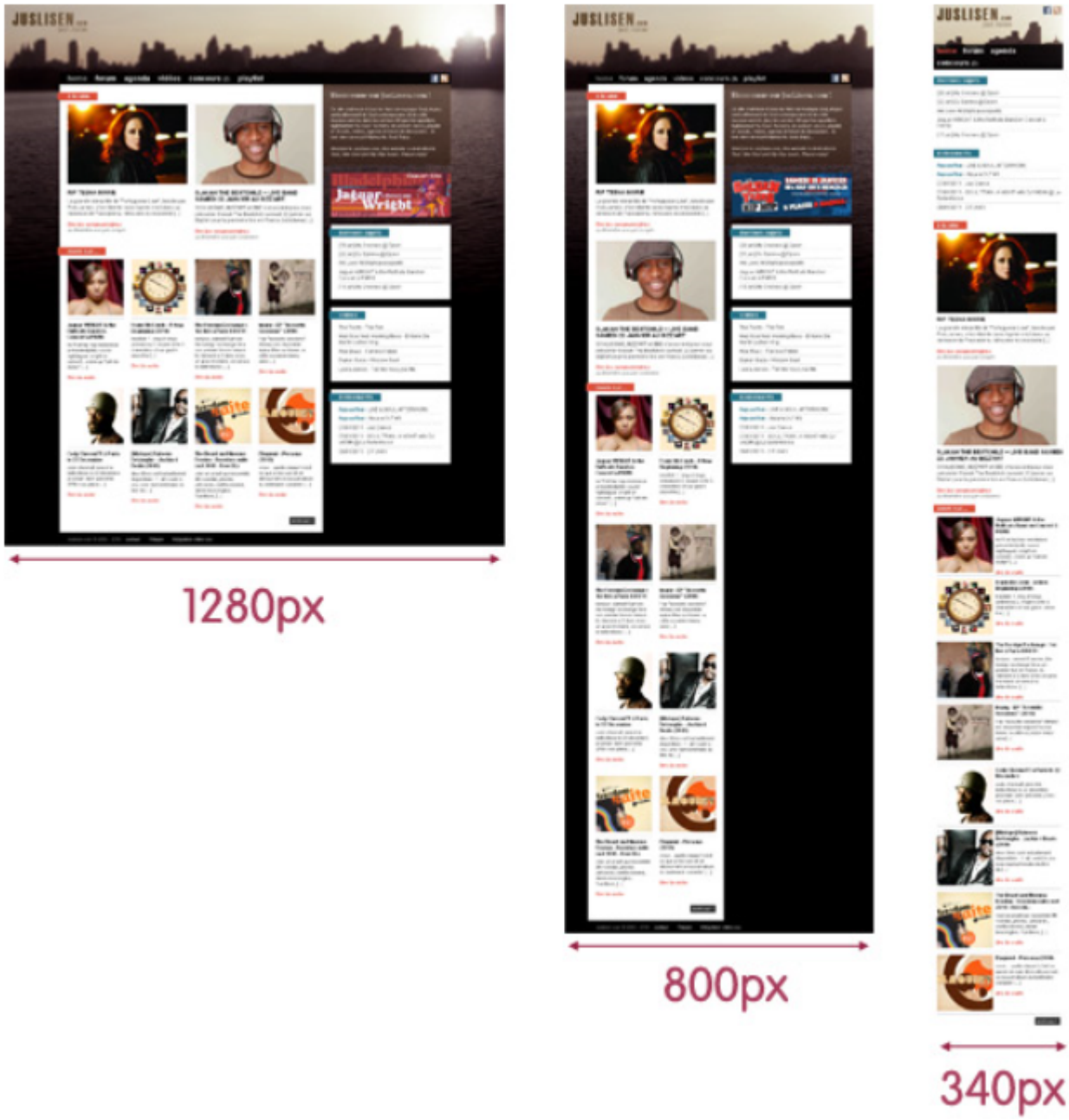


FIGURE 60 – Influence des media queries

17.3 TESTER LES MEDIA QUERIES

Les media queries sont surtout utilisées pour adapter le design du site aux différentes largeurs d'écran.

Faisons un test tout simple : nous allons changer la couleur et la taille du texte si la fenêtre fait plus ou moins de 1024 pixels de large. Pour ce test, nous allons utiliser la seconde méthode qui consiste à écrire la règle directement dans un même fichier .css.

Exemple 80. *Tester les media queries*

```
/* Paragraphes en bleu par défaut */
p
{
    color: blue;
}

/* Nouvelles règles si la fenêtre fait au plus 1024px de large */
@media screen and (max-width: 1024px)
{
    p
    {
        color: red;
        background-color: black;
        font-size: 1.2em;
    }
}
```

CHAPITRE 18

LE RÉFÉRENCEMENT

Il y a 3 étapes importantes lors du référencement d'un site Web :
– La définition de la stratégie ;
– La mise en place de la stratégie ;
– L'analyse et le suivi des résultats.

18.1 LES NOTIONS DE BASE

– **Indexation** : Inscrire son site sur les moteurs de recherche afin qu'il ressorte dans les résultats.
– **Positionnement** : Un site est indexé sur Google et est positionné à une certaine place. L'objectif des méthodes que nous allons voir n'est pas d'améliorer le référencement de son site (comme on peut le lire partout) mais le positionnement de son site.
– **Référencement** : Quand on parle de référencement, on parle généralement de toutes les techniques qui vont nous permettre d'améliorer le positionnement de notre site en fonction des moteurs de recherche. Search Engine Optimization" est la même chose que le "Référencement" mais en anglais.
– **SERP** : "Search Engine Result Page". Il s'agit de la page de résultat des moteurs de recherche.

Le référencement est un travail long et fastidieux. Il nous faudra plusieurs mois pour arriver à un classement honorable sur un mot clé ayant un peu de concurrence. Les algorithmes évoluant régulièrement avec la technologie, le positionnement n'est jamais acquis. Il nous faudra en permanence le travailler. Par exemple, les réseaux sociaux ont de plus en plus d'importance dans l'algorithme de Google. On appelle ça le SMO (Social Media Optimization).

18.2 LES FORMES DE RÉFÉRENCEMENT

Le référencement peut prendre 3 formes :
- Le **référencement naturel** : Celui que nous allons étudier.
- Les **liens sponsorisés** : Publicité sur les moteurs de recherche.
- Le **référencement payant** : Référencement de notre site par une société experte.

Le référencement naturel est un ensemble de techniques (que nous allons voir dans les sections suivantes) permettant de positionner le mieux possible notre site web sur les moteurs de recherches sur certains mots-clés définis par nos soins.

Les liens sponsorisés permettent d'afficher notre site sur certaines requêtes suivant les mots-clés que nous avons achetés.

Sur l'image ci dessous, Nous voyons la position des résultats du référencement naturel (en vert) et des liens sponsorisés (en rouge) sur une page Google.

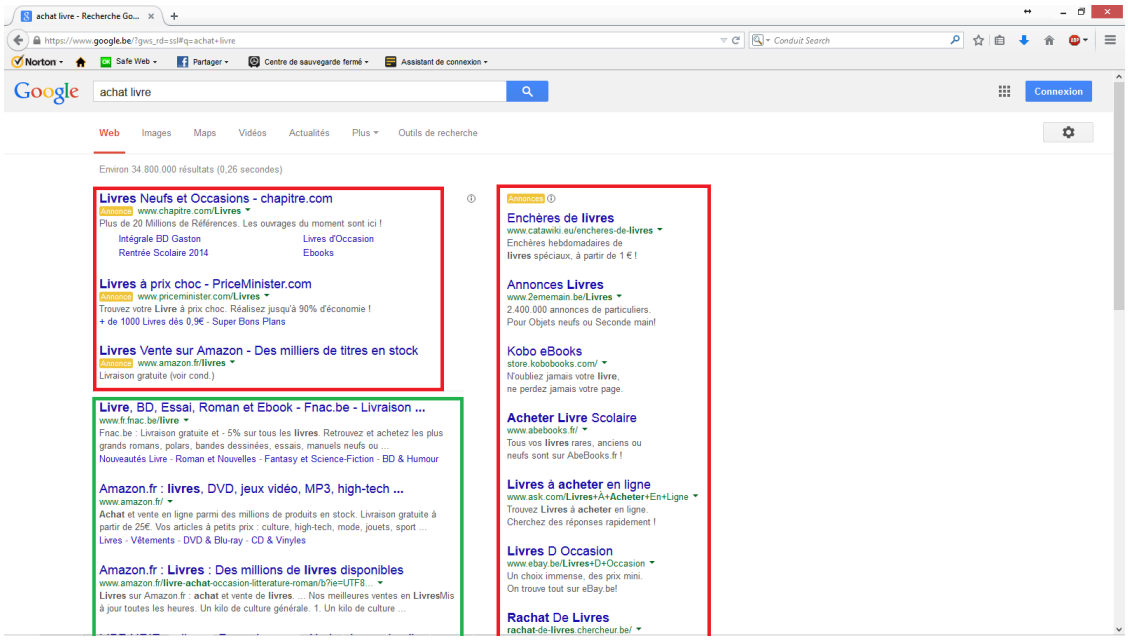


FIGURE 61 – Les formes de référencement

18.3 Les moteurs de recherche

Définition 12 (Le moteur de recherche)**.**

Le moteur de recherche permet à l'internaute de trouver un ou plusieurs sites web qui répondra à ses attentes. L'internaute demande au moteur de recherche des informations sur ce qu'il souhaite trouver, et le moteur de recherche lui fournit une liste de sites classés par pertinence.

La pertinence des résultats est un calcul complexe calculé par l'algorithme du moteur de recherche. L'algorithme analyse plusieurs centaines d'éléments sur tous les sites web afin de les classer, et de proposer à l'internaute le meilleur résultat possible selon une requête.

Il existe différents moteurs de recherche. La liste suivante est non exhaustive :
- Google
- Bing
- Yahoo
- Orange
- Ask
- Voila

18.3.1 L'histoire des moteurs de recherche

Il faut savoir que Google n'est pas présent depuis le début. Le premier vrai moteur de recherche était Excite, puis détrôné rapidement par Lycos en 1994 et Altavista en 1995.

En parallèle, Yahoo avait lancé en 1994 sont annuaire qu'il impose comme vecteur de qualité. A l'époque, les algorithmes de recherche n'étaient pas aussi complexes. De bonnes balises **<meta />** correctement optimisées avec les mots clés, présents dans le contenu, suffisaient pour positionner une page correctement.

Tout se passait bien pour Altavista qui ne se doutait pas qu'en 1998, tout allait basculer. Google faisait son arrivée. Il est souvent difficile de se lancer sur un marché ayant déjà des concurrents bien positionnés. Mais cela n'a pas fait peur à Google. Nous connaissons tous la suite...

L'algorithme est la clé de la réussite de Google. Il prenait en compte et analysait les backlinks (liens pointant vers un site) pour critère de pertinence. A l'époque, le critère était très pertinent car les webmasters ne manipulaient pas les backlinks. La quantité des liens était le critère principal de Google.

18.3.2 Le fonctionnement des moteurs de recherche

Pour un internaute lambda, un moteur de recherche est uniquement un site qui fournit une liste d'autres sites. Nous allons donc voir comment fonctionne un moteur de recherche.

1. Des robots de moteurs de recherche (spiders) parcourent le web de page en page afin de collecter les pages web.
2. Ces informations sont stockées dans une base de données appelée index.
3. Grâce à cette base de données, les moteurs de recherche vont pouvoir traiter les requêtes demandées par les internautes selon un algorithme.
4. Les résultats sont affichés sur le moteur de recherche après qu'une requête ait été demandée par un internaute.

18.3.3 Les robots

Les robots, nommés GoogleBot pour Google et BingBot pour Bing, parcourent le web 24h/24, 7 jours sur 7. Le robot, une fois sur un site, va stocker le code source de la page, puis va suivre les liens figurant sur la page et y accéder.

Mais le robot ne ré-indexe pas les sites déjà visités qui n'ont pas évolué. Le robot arrive sur une page web. S'il ne connaît pas cette pages web, il l'indexe. Sinon il vérifie s'il y a eu des changements majeurs. Si c'est le cas, il remplace l'ancienne version indexée par la nouvelle, sinon il ne fait rien et continue son chemin.

Le robot va se baser sur plusieurs critères :
– La date de dernière modification du document ;
– Le poids de la page ;
– La modification du contenu ;
– Qu'est ce qui a été modifié (pied de page, menu, contenu, etc...).

Mais il faut savoir que les robots reviennent plus rapidement sur des sites avec du contenu mis à jour régulièrement, comme des sites d'actualités, que des sites avec des mises à jour peu fréquentes.

18.3.4 Les stop words

Les stops words sont les petits mots sans importance, à première vue, tels que "le", "la", "les", "un", etc... mais finalement ils ne sont pas si inutiles que ca. Il faut faire attention car même si la plupart n'y font pas attention dans leur stratégie de référencement, car Google ne les prends pas en compte, ils ont leur importance.

Les requêtes sur "joueur de foot" et "joueur foot" ne sont pas les mêmes contrairement à ce que certains peuvent penser. Pourquoi ? Tout simplement car Google tient compte de la place des mots. Il ignore le mot "de" mais il se rappelle qu'il y a un mot entre "joueur" et "foot"

Si nous souhaitons lors d'une recherche, que les stops words soient pris en compte, nous devons saisir notre requête entre guillemets : "joueur de foot" Les guillemets disent à Google que nous souhaitons une recherche sur les 3 mots saisis, dans l'ordre saisis.

18.4 Les annuaires

Il y a quelques années encore, les annuaires avaient beaucoup plus d'importance qu'aujourd'hui. Maintenant, la source de trafic se partage en 3.

- ceux qui nous trouvent par les moteurs de recherche ;
- ceux qui arrivent sur notre site en tapant l'URL dans la barre d'adresse ;
- ceux qui cliquent sur un lien les redirigeant vers notre site via un autre un site web.

Il y a quelques années, on pouvait ajouter ceux qui visitaient la fiche de votre site sur les annuaires. Les annuaires étaient une source de trafic important. Les internautes descendaient dans la hiérarchisation des catégories afin de trouver des sites précis pour leur recherche.

18.5 Les mots-clés

Les mots-clés sont très importants dans le référencement de notre site. Un mauvais choix de mots clés et c'est toute notre stratégie de référencement qui devient bancale. Nous allons voir comment définir les bons mots clés pour notre site à l'aide de différents outils.

Choisir les mots clés sur lesquels baser son référencement n'est pas si facile que cela en à l'air.

18.5.1 Le choix

Il y a 3 critères à prendre en compte :

- l'**activité** : Les mots clés doivent décrire aux mieux votre activité pour recevoir des visiteurs qualifiés qui seront intéressés par le contenu de votre site.
- le **volume de recherche** : Les mots clés doivent être saisis dans les moteurs de recherche par de nombreux internautes.
- la **concurrence** : Difficile à évaluer, mais plus il y a de sites présents sur ce mot clé, plus il est difficile d'apparaitre en tête de liste.

Il faudra donc trouver le bon compromis entre ces trois critères.

18.5.2 Le principe de la longue traine

Le principe de la longue traine en référencement repose sur le contenu d'un site Internet. Pour être bien positionné, il faut un contenu de qualité. Mais le plus important sera l'analyse et le suivi qui en est fait derrière.

Lors d'une stratégie de référencement, la première chose qui est faite c'est le choix de ses mots-clés. Ceux ci vont permettre de créer la tête de la longue traîne. Contrairement à la queue qui elle est constituée du contenu de nos pages Web.

FIGURE 62 – Le principe de la longue traine

Astuce 31 (Tester ses mots-clés)**.**

En utilisant le générateur Google Adwords, il est possible de tester la pertinence de nos mots-clés. Pour cela, il faut saisir nos mots-clés les uns en dessous des autres, cliquer sur "Options Avancées" où un menu se déroule afin d'affiner nos recherches. Notons qu'une option importante est le pays et la langue. Une fois lancé, le générateur va répertorier toute une liste de mots-clés. Il y aura de fortes chances de trouver des mots-clés auxquels nous n'avions pas pensé.

18.6 LES BALISES AU SERVICE DU RÉFÉRENCEMENT

18.6.1 La balise title

Nous retrouvons cette balise dans plusieurs endroits :
- dans notre **navigateur** ;
- dans le **résultat des moteurs de recherches** ;
- dans nos **marques pages** ou **favoris**.

Les noms de titre à éviter :
- Les **symboles** ;
- Les **titres non optimisés** (ex : Bienvenue sur mon site...) ;
- Les **titres inexistants**.

18.6.2 La balise meta

Les balises **<meta />**, rappelons-le, permettent d'envoyer des informations (Metadonnées). Ces informations peuvent être traitées par :
- les **moteurs de recherche** ;
- les **navigateurs** ;
- les **outils d'aide au référencement**.

Il existe beaucoup de balises **<meta />** :
- **meta content-Type** : jeu d'encodage ;
- **meta description** : description ;
- **meta keywords** : mots-clés ;
- **meta generator** : spécifie les logiciels utilisés lors de la création du site Internet ;
- **meta author** : auteur ;
- **meta copyright** : droit d'auteur ;
- **meta dublin Core** : organisation de données propres au site ;
- **meta http-equiv** fournit un en-tête HTTP ;
- **meta refresh** : rafraîchir la page à intervalle régulier ;
- **meta robots** : spécifie les répertoires à référencer ou à exclure pour l'analyse des robots.

La balise **<meta />** sera particulièrement utile pour faire reconnaître notre page par les robots de recherche du genre Google.

Indiquons au robot de recherche que le contenu de l'attribut **description** est la description de notre page HTML. Ce contenu pourra être affiché comme résultat.

Exemple 81. *La description de la page*

```
1 <meta name="description" content="description de notre page"/>
```

Indiquons au robot de recherche que le contenu de l'attribut **content** est une série de mots-clés qui définira plus finement notre page. Il peut être utile de prévoir quelques mots-clés en anglais si nos pages sont en français.

Exemple 82. *Les mots-clés de la page*

```
1 <meta name="keywords" content="mot-clé, mot-clé, mot-clé... "/>
```

Nous pouvons appeler automatiquement une autre page (située à l'URL indiquée) après un délai de x secondes et établir le rafraîchissement de la page afin de permettre l'affichage à intervalle régulier de différentes informations (publicitaires ou autres).

Exemple 83. *Le rafraîchissement de la page*

```
1 <meta http-equiv="refresh" content="x" url="adresse"/>
```

Voici un lien pour générer ses balises **<meta />** : `http://www.outils-referencement.com/outils/pages-web/balises-meta`

18.6.3 Les balises h(x) et nav

Nous allons simplement rappeler que ces balises jouent un rôle essentiel dans le référencement d'un site.

CHAPITRE 19

LES CONCEPTS ÉVOLUÉS

19.1 LE LANGAGE JAVASCRIPT

JavaScript est un langage qui existe depuis de nombreuses années maintenant et que l'on utilise fréquemment sur le Web en plus de HTML et CSS. C'est probablement l'un des premiers langages que nous voudrions apprendre maintenant que nous avons des connaissances en HTML et CSS.

Définition 13 (Le langage Javascript)**.**

Javascript est un langage structuré sous la forme d'un ensemble de codes (scripts) directement incorporés dans HTML, qui permet de faire réaliser au navigateur certaines fonctions déterminées.

On peut faire déjà beaucoup de choses en HTML et CSS mais, lorsqu'on veut rendre sa page plus interactive, un langage comme JavaScript devient indispensable.

Voici quelques exemples de ce à quoi peut servir JavaScript :
- On l'utilisera le plus souvent pour modifier des propriétés CSS sans avoir à recharger la page. Par exemple, vous pointez sur une image et le fond de votre site change de couleur (ce n'est pas possible à faire avec un :hover car cela concerne deux balises différentes, c'est bien là une limite du CSS).
- On peut l'utiliser aussi pour modifier le code source HTML sans avoir à recharger la page, pendant que le visiteur consulte la page.
- Il permet aussi d'afficher des boîtes de dialogue à l'écran du visiteur. . .
- ... ou encore de modifier la taille de la fenêtre.

JavaScript est un langage qui se rapproche des langages de programmation tels que le C, C++, Python, Ruby... À l'inverse, HTML et CSS sont davantage des langages de description : ils décrivent comment la page doit apparaître mais ils ne donnent pas d'ordres directs à l'ordinateur (« fais ceci, fais cela... »), contrairement à JavaScript.

Exemple 84. *Exemple d'un script*

```
<html>
<head>
<script language="JavaScript">
function hello()
{
        alert("Bienvenue dans ce script!");
}
</script>
</head>
<body>
...
<input type="button" name="bienvenue" value="Voir message" onClick="
    hello()" />
...
</body>
</html>
```

Les navigateurs sont de plus en plus efficaces dans leur traitement de JavaScript, ce qui fait que les pages qui utilisent JavaScript sont de plus en plus réactives. On peut ainsi arriver aujourd'hui à créer des sites qui deviennent littéralement des applications web, l'équivalent de logiciels mais disponibles sous forme de sites web !

Un exemple célèbre : Google Docs, la suite bureautique de Google, disponible sur le Web.

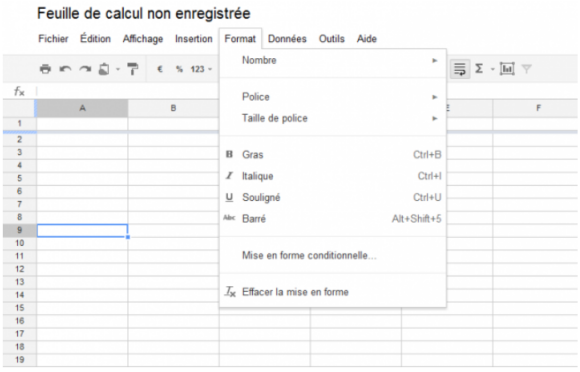


FIGURE 63 – Le tableur Google Docs

19.2 Le langage Java

Le langage Java est un langage de programmation orienté objet qui permet de compiler des programmes exécutables mais aussi de petites applications, indépendantes de toute plate-forme, appelées des applets. Ces applets, pourvu que le navigateur sache les interpréter, sont exécutées en local sur la machine affichant le document.

Cette intégration des applets est réalisée par la balise **<applet></applet>**.

Exemple 85. *Exemple d'une applet*

```
<html>
<head>
</head>
<body>
<applet code="clock.class" width="200" height="200">
        <param ... les paramètres transmis par l'auteur ... >
</applet>
...
</body>
</html>
```

Notons que "clock.class" est le fichier objet du programme Java qui se trouve dans le répertoire de ce site.

19.3 Les technologies liées à HTML 5

Le W3C ne travaille pas que sur les langages HTML et CSS. Ce sont certes les plus connus, mais le W3C cherche aussi à définir d'autres technologies qui viennent compléter HTML et CSS. Elles sont nombreuses et on les confond d'ailleurs souvent avec HTML5.

Voici une liste non exhaustive de ces nouvelles technologies introduites en parallèle de HTML5 (notons que certaines ne sont pas vraiment "nouvelles" mais elles reviennent sur le devant de la scène) :

- **Canvas** : permet de dessiner au sein de la page web, à l'intérieur de la balise HTML <canvas>. On peut dessiner des formes (triangles, cercles...) mais aussi ajouter des images, les manipuler, appliquer des filtres graphiques... Au final, cela nous permet de réaliser aujourd'hui de véritables jeux et des applications graphiques directement dans des pages web !
- **SVG** : permet de créer des dessins vectoriels au sein des pages web. À la différence de Canvas, ces dessins peuvent être agrandis à l'infini (c'est le principe du vectoriel). Le logiciel Inkscape est connu pour permettre de dessiner des SVG.
- **Drag & Drop** : permet de faire un "glisser-déposer" des objets dans la page web, de la même façon qu'on peut faire glisser-déposer des fichiers sur son bureau. Gmail l'utilise pour permettre d'ajouter facilement des pièces jointes à un e-mail.

- **File API** : permet d'accéder aux fichiers stockés sur la machine du visiteur (avec son autorisation). On l'utilisera notamment en combinaison avec le Drag & Drop.
- **Géolocalisation** : pour localiser le visiteur et lui proposer des services liés au lieu où il se trouve (ex. : les horaires des salles de cinéma proches). La localisation n'est pas toujours très précise, mais cela peut permettre de repérer un visiteur à quelques kilomètres près (avec son accord).
- **Web Storage** : permet de stocker un grand nombre d'informations sur la machine du visiteur. C'est une alternative plus puissante aux traditionnels cookies. Les informations sont hiérarchisées, comme dans une base de données.
- **Appcache** : permet de demander au navigateur de mettre en cache certains fichiers, qu'il ne cherchera alors plus à télécharger systématiquement. Très utile pour créer des applications web qui peuvent fonctionner même en mode « hors ligne » (déconnecté).
- **Web Sockets** : permet des échanges plus rapides, en temps réel, entre le navigateur du visiteur et le serveur qui gère le site web (c'est une sorte d'AJAX amélioré). C'est un peu l'avenir des applications web, qui pourront devenir aussi réactives que les vrais programmes.
- **WebGL** : permet d'introduire de la 3D dans les pages web, en utilisant le standard de la 3D OpenGL. Les scènes 3D sont directement gérées par la carte graphique.

19.4 Les sites web dynamiques

Les sites web dynamique utilisent un langage de programmation qui s'exécute sur le "serveur" qui contient notre site web. Contrairement au langage javascript qui s'exécute sur le poste client.

Nous retrouvons comme langages de programmation côté serveur :

- **PHP** : l'un des plus connus. Facile à utiliser et puissant, il est utilisé notamment par Facebook.
- **JEE (Java)** : très utilisé dans le monde professionnel, il s'agit d'une extension du langage Java qui permet de réaliser des sites web dynamiques, puissants et robustes. Au début, il est un peu plus complexe à prendre en main que PHP.
- **ASP .NET (C#/VB)** : assez semblable à JEE, c'est le langage de Microsoft. On l'utilise en combinaison avec d'autres technologies Microsoft (Windows Server...). Il utilise le puissant framework .NET, véritable couteau suisse des développeurs, qui offre de nombreuses fonctionnalités.
- **Django (Python)** : une extension du langage Python qui permet de réaliser rapidement et facilement des sites web dynamiques. Il est connu pour générer des interfaces d'administration prêtes à l'emploi.
- **Ruby on Rails (Ruby)** : une extension du langage Ruby, assez similaire à Django, qui permet de réaliser des sites web dynamiques facilement et avec une grande souplesse.

BIBLIOGRAPHIE

[1] J. Engels. *HTML 5 et CSS 3*. 2012.

[2] M. Nebra. Apprenez à créer votre site web avec html 5 et css 3. http://fr.openclassrooms.com/informatique/cours/apprenez-a-creer-votre-site-web-avec-html5-et-css3 (site consulté le 30/04/2014).

TABLE DES MATIÈRES

TABLE DES FIGURES