



# **Apprenez à créer votre site web avec HTML5 et CSS3**

**Par M@teo21**



[www.siteduzero.com](http://www.siteduzero.com)

*Dernière mise à jour le 17/10/2011*

## Sommaire

Sommaire .....	1
Informations sur le tutoriel .....	3
Apprenez à créer votre site web avec HTML5 et CSS3 .....	5
Informations sur le tutoriel .....	5
Partie 0 : .....	6
Comment fait-on pour créer des sites web ? .....	6
Comment fonctionnent les sites web ? .....	6
HTML et CSS : deux langages pour créer un site web .....	10
Le rôle de HTML et CSS .....	10
Les différentes versions de HTML et CSS .....	11
L'éditeur de texte .....	12
Sous Windows .....	13
Sous Mac OS X .....	15
Sous Linux .....	15
Les navigateurs .....	16
Pourquoi le navigateur est important .....	17
Les navigateurs sur ordinateur .....	17
Les navigateurs sur mobile .....	19
Votre première page web en HTML .....	20
Créer une page web avec l'éditeur .....	20
Les balises et leurs attributs .....	26
Les balises .....	26
Les attributs .....	27
Structure de base d'une page HTML5 .....	28
Le doctype .....	29
La balise <html> .....	29
L'en-tête <head> et le corps <body> .....	29
Les commentaires .....	32
Insérer un commentaire .....	32
Tout le monde peut voir vos commentaires... et tout votre code HTML ! .....	32
Organiser son texte .....	34
Les paragraphes .....	34
Sauter une ligne .....	35
Les titres .....	37
La mise en valeur .....	39
Mettre un peu en valeur .....	40
Mettre bien en valeur .....	40
Marquer le texte .....	41
N'oubliez pas : HTML pour le fond, CSS pour la forme .....	41
Les listes à puces .....	43
Liste non ordonnée .....	43
Liste ordonnée .....	44
Créer des liens .....	44
Un lien vers un autre site .....	45
Un lien vers une autre page de son site .....	47
Deux pages situées dans un même dossier .....	47
Deux pages situées dans des dossiers différents .....	48
Résumé en images .....	49
Un lien vers une ancre .....	51
Lien vers une ancre située dans une autre page .....	52
Cas pratiques d'utilisation des liens .....	52
Un lien qui affiche une infobulle au survol .....	53
Un lien qui ouvre une nouvelle fenêtre .....	53
Un lien pour envoyer un e-mail .....	54
Un lien qui de téléchargement d'un fichier .....	54
Les images .....	54
Les différents formats d'images .....	56
Le JPEG .....	56
Le PNG .....	57
Le GIF .....	58
Il existe un format adapté à chaque image .....	58
Les erreurs à éviter .....	58
Insérer une image .....	60
Insertion d'une image .....	60
Ajouter une infobulle .....	60
Miniature cliquable .....	61

Les figures .....	62
Création d'une figure .....	63
Bien comprendre le rôle des figures .....	64
<b>Partie 1 : Les joies de la mise en forme avec CSS .....</b>	<b>66</b>
<b>Mettre en place le CSS .....</b>	<b>66</b>
Où mettre du CSS ? .....	66
Résumé visuel .....	70
Appliquer un style : sélectionner une balise .....	73
Appliquer un style à plusieurs balises .....	75
Des commentaires dans du CSS .....	76
Appliquer un style : class et id .....	77
Les balises universelles .....	80
<b>Formatage du texte en CSS (partie 1/2) .....</b>	<b>81</b>
Taille du texte .....	83
Polices .....	85
Alignement .....	87
Alignements simples .....	88
L'indentation .....	89
<b>Formatage du texte en CSS (partie 2/2) .....</b>	<b>90</b>
Effets de style .....	91
Mettre en italique .....	91
Mettre en gras .....	92
Les majuscules en CSS .....	92
Un peu de déco ? .....	93
Les couleurs .....	95
Indiquer le nom de la couleur .....	95
La notation hexadécimale .....	96
La méthode RGB .....	97
Et en Bonus Track... ..	98
Le fond .....	99
La couleur de fond .....	100
L'image de fond .....	101
<b>Les pseudo-formats .....</b>	<b>105</b>
Des liens sympas .....	106
Au passage de la souris .....	106
Au moment du clic .....	107
Lorsque le lien est sélectionné .....	108
Quand la page a déjà été vue... ..	109
Première lettre et première ligne .....	110
La première lettre .....	111
La première ligne .....	111
Avant / Après .....	112
Une image au lieu du texte ? .....	113
<b>Partie 2 : XHTML &amp; CSS, toujours plus forts ! .....</b>	<b>114</b>
<b>Les listes à puces .....</b>	<b>115</b>
Différents types de listes (XHTML) .....	116
Liste non ordonnée .....	116
Liste ordonnée .....	117
Liste de définitions .....	117
Décorez vos listes (CSS) .....	118
Retrait des listes .....	119
Représentation de la puce .....	120
Changer l'image de la puce .....	123
<b>Mise en boîte (partie 1/2) .....</b>	<b>124</b>
Block et Inline c'est pas pareil ! .....	125
Quelques exemples .....	126
Les balises universelles .....	126
Respectez la sémantique ! .....	127
La taille .....	127
Minimum et maximum .....	129
"Couper" un block avec un overflow .....	129
Les bordures .....	131
En haut, à droite, à gauche, en bas... ..	132
Ça marche aussi sur des balises inline ! .....	133
Les marges .....	135
En haut, à droite, à gauche, en bas... Et on recommence ! .....	136
Centrer des blocks .....	138
<b>Mise en boîte (partie 2/2) .....</b>	<b>138</b>
Transformer un inline en block (et vice-versa) .....	140
Les flottants .....	142
Faire flotter une image .....	142
Créer une lettrine (exercice) .....	143
Stopper un flottant .....	145

Positionnement absolu, fixe et relatif .....	146
Le positionnement absolu .....	147
Le positionnement fixe .....	149
Le positionnement relatif .....	150
<b>Créons le design de votre site web ! .....</b>	<b>151</b>
Bon, par quoi on commence ? .....	152
Design fixe ou extensible ? .....	153
Primo, le XHTML .....	154
La structure générale .....	155
L'en-tête .....	156
Menu et sous-menus .....	157
Le corps .....	160
Allez le dernier : le pied de page .....	160
Le code XHTML final .....	161
Secundo, le CSS .....	164
Centrer le design .....	164
L'en-tête .....	165
Le menu .....	166
Le corps .....	169
Le pied de page (et on a fini !) .....	173
Attendez, c'est pas fini ! .....	175
Rappel : insérer un CSS dans le XHTML .....	176
Changez le CSS, changez le design ! .....	176
Laissez le visiteur choisir le design .....	178
<b>Les tableaux .....</b>	<b>179</b>
Un tableau simple .....	179
La ligne d'en-tête .....	181
Titre du tableau .....	183
Des tableaux plus élaborés .....	183
Diviser un gros tableau .....	184
3, 2, 1... Fusioooooon ! .....	185
Et avec un peu de CSS... ..	188
Rien qu'avec des propriétés CSS connues .....	188
Quelques nouvelles propriétés CSS .....	190
<b>Les formulaires .....</b>	<b>192</b>
Créer un formulaire .....	192
Les zones de saisie .....	194
Zone de texte à une ligne .....	195
Les labels .....	195
Quelques attributs supplémentaires .....	196
Zone de mot de passe .....	197
Zone de texte multiligne .....	197
Éléments d'options .....	200
Les cases à cocher .....	200
Les zones d'options .....	201
Les listes déroulantes .....	202
Un formulaire accessible et design ? .....	205
Définir un ordre de tabulation .....	205
Définir des touches de raccourci .....	206
Organiser le formulaire en plusieurs zones .....	207
Gouache time ! .....	208
Y'a plus qu'à appuyer sur le bouton ! .....	209
<b>Et maintenant, on fait quoi ? (Conclusion) .....</b>	<b>211</b>
L'heure du bilan a sonné .....	213
Résumons ce qu'on a vu .....	213
Ce qu'on n'a pas vu .....	214
On va quand même pas s'arrêter là ? .....	215
Un langage client : le Javascript .....	216
Un langage serveur : le PHP .....	217
<b>Partie 3 : Annexes .....</b>	<b>219</b>
<b>Envoyez votre site sur le web .....</b>	<b>220</b>
Le nom de domaine .....	220
Réserver un nom de domaine .....	221
L'hébergeur .....	222
Le rôle de l'hébergeur .....	224
Trouver un hébergeur .....	224
Commander un hébergement pour votre site web .....	227
Utiliser un client FTP .....	228
Installer un client FTP .....	229
Configurer le client FTP .....	230
Transférer les fichiers .....	231
<b>Gagner de l'argent grâce à la publicité .....</b>	<b>232</b>
Qu'est-ce qu'une bonne publicité ? .....	234
Principe n°1 : ne pas gêner la navigation .....	234

Principe n°2 : une publicité ciblée .....	235
Installation d'une bannière publicitaire .....	236
Les avantages d'AdFever .....	236
S'inscrire à AdFever .....	236
L'interface d'AdFever .....	237
Le vocabulaire de la pub .....	239
Vos questions courantes .....	241
<b>Le W3C et les standards du web .....</b>	<b>242</b>
L'histoire du web .....	243
Du HTML au XHTML .....	245
Et le CSS ? .....	245
Votre site est-il valide ? .....	245
Le validateur XHTML .....	246
Le validateur CSS .....	248
<b>Liste des balises XHTML .....</b>	<b>249</b>
Balises de premier niveau .....	249
Le code minimal d'une page XHTML .....	250
Balises d'en-tête .....	251
Balises de structuration du texte .....	253
Balises de liste .....	254
Balises de tableau .....	256
Balises de formulaire .....	256
Balises génériques .....	258
<b>Liste des propriétés CSS .....</b>	<b>259</b>
Propriétés de formatage de texte .....	260
Police, taille et décorations .....	260
Alignement .....	261
Propriétés de couleur et de fond .....	262
Couleur .....	262
Image de fond .....	262
Propriétés des boîtes .....	264
Dimensions .....	264
Marges extérieures .....	264
Marges intérieures .....	265
Bordures .....	265
Propriétés de positionnement et d'affichage .....	266
Affichage .....	267
Positionnement .....	267
Propriétés des listes .....	268
Propriétés des tableaux .....	270
Autres propriétés .....	270

# Apprenez à créer votre site web avec HTML5 et CSS3

Vous rêvez d'apprendre à créer des sites web ?  
(mais vous avez peur que ce soit compliqué car vous débutez ?)

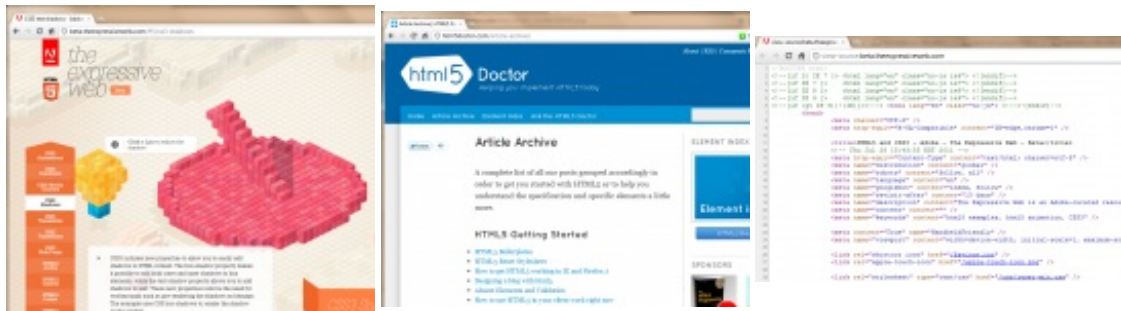
Vous êtes au bon endroit ! Ce cours est destiné aux **débutants** qui ne connaissent rien à la création de sites web et qui n'attendent qu'une chose : qu'on leur explique pas à pas comment tout cela fonctionne avec des mots simples et des exemples concrets !

Nous découvrirons dans ce cours les célèbres langages **HTML5** et **CSS3** que l'on utilise aujourd'hui pour concevoir des sites web.

Même si ces "langages" ne signifient pas encore grand chose pour vous, rassurez-vous : tout ce que vous avez besoin de savoir sera expliqué dans ce cours ! Vous découvrirez notamment comment :

- Insérer du texte, des images et des vidéos
- Faire des liens entre vos pages
- Mettre en forme en modifiant la couleur, la taille, le fond, la police...
- Positionner les éléments du site comme bon vous semble : en-tête, menus...
- ... et bien plus encore ! 😊

Vous n'avez qu'une chose à faire : lisez les chapitres dans l'ordre et découvrez le monde fascinant de la création de sites web avec HTML5 et CSS3 !



*Aperçu de sites web créés à l'aide des langages HTML5 et CSS3 que nous allons découvrir*

## Informations sur le tutoriel

Auteur :

- [Mathieu Nebra \(M@teo21\)](#)

Difficulté :

Temps d'étude estimé : 20 jours

Licence :



## Partie 0 :

---

### Comment fait-on pour créer des sites web ?

Bonjour et bienvenue à toutes et à tous !

Voici donc le premier chapitre de ce cours pour débutants, qui va vous apprendre à créer votre site web !

Nous allons passer quelques temps ensemble, tout dépendra de la vitesse à laquelle vous apprendrez. Si vous lisez ce cours régulièrement et à une bonne vitesse, vous l'aurez terminé en une à deux semaines. Mais si vous avez besoin d'un peu plus de temps, ne vous inquiétez pas : le principal est que vous y alliez à votre rythme, en prenant du bon temps de préférence. 😊

Je me doute que vous vous posez mille questions :

- Quels logiciels faut-il pour créer des sites web ?
- Que signifient les langages HTML5 et CSS3 ?
- Est-ce que tout cela n'est pas trop compliqué et plutôt fait pour les programmeurs ?

Je vous propose de commencer par la question la plus simple mais aussi la plus importante : *comment fonctionnent les sites web ?*

---

## Comment fonctionnent les sites web ?



Comment fonctionnent les sites web ?

Non, n'ayez pas peur de poser des questions même si vous pensez qu'elles sont "bêtes". Il est très important que nous en parlions un peu avant de nous lancer à fond dans la création de sites ! 😊

Je suis certain que vous consultez des sites web tous les jours. Pour cela, vous lancez un programme appelé le navigateur web en cliquant sur l'une de ces icônes :



*Les navigateurs web, des programmes qui permettent d'afficher des sites web*

Avec le navigateur, vous pouvez consulter n'importe quel site web. Voici par exemple un navigateur affichant le célèbre site web Wikipédia :



*Un site web : Wikipédia*

Je suis sûr que vous avez l'habitude d'utiliser un navigateur web ! Aujourd'hui, tout le monde sait aller sur le Web... mais qui sait vraiment comment le Web fonctionne ? Comment créer des sites web comme celui-ci ?



J'ai entendu parler de HTML, de CSS, est-ce que ça a un rapport avec le fonctionnement des sites web ?

Tout à fait ! 😊

Il s'agit de *langages informatiques* qui permettent de créer des sites web. Tous les sites web sont basés sur ces langages, ils sont incontournables et universels aujourd'hui. Ils sont à la base même du Web. Le langage HTML a été inventé le premier par un certain Tim Berners-Lee en 1991...

Tim Berners-Lee suit encore aujourd'hui avec attention l'évolution du Web. Il a créé le [World Wide Web Consortium \(W3C\)](http://www.w3.org/)



qui définit les nouvelles versions des langages. Il a par ailleurs créé plus récemment la [World Wide Web Foundation](#), qui analyse et suit l'évolution du Web.



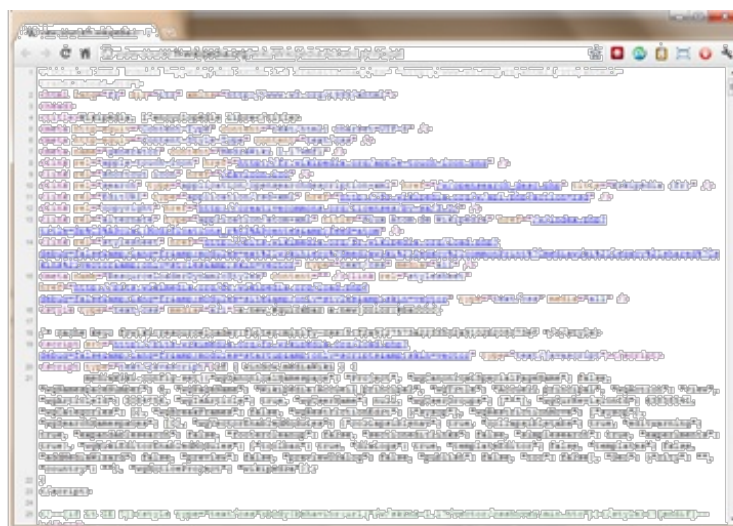
*Tim Berners-Lee, inventeur du Web*



De nombreuses personnes confondent (à tort) Internet et le Web. Il faut savoir que le *Web fait partie* d'Internet. Internet est donc un grand ensemble qui comprend, entre autres : le Web, les e-mails, la messagerie instantanée, etc. Tim Berners-Lee n'est donc pas l'inventeur d'Internet, c'est "seulement" l'inventeur du Web. 🤨

Les langages HTML et CSS sont à la base du fonctionnement de tous les sites web. Quand vous visitez un site avec votre navigateur, il faut savoir que derrière des rouages s'activent pour permettre au site web de s'afficher. L'ordinateur se base sur ce qu'on lui a expliqué en HTML et CSS pour savoir ce qu'il doit afficher :

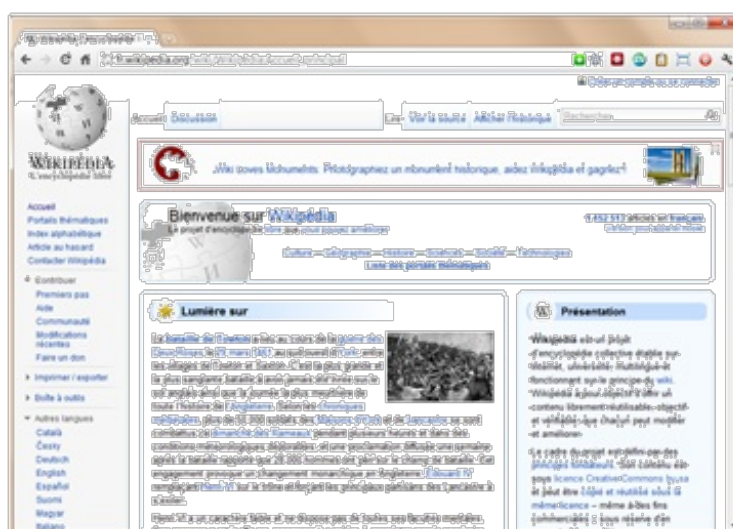
## Langages HTML et CSS



*Traduction par  
l'ordinateur*



## Résultat visible à l'écran



HTML et CSS sont deux "langues" qu'il faut savoir parler pour créer des sites web. C'est le navigateur web qui fera la traduction entre ces langages informatiques et ce que vous verrez s'afficher à l'écran.

Vous vous demandez sûrement pourquoi il faut connaître 2 langages pour créer des sites web ? Je vous réponds sans plus tarder !

## HTML et CSS : deux langages pour créer un site web

Pour créer un site web, on doit donner des instructions à l'ordinateur. Il ne suffit pas simplement de taper le texte qu'il y aura dans son site (comme on le ferait dans un traitement de texte Word par exemple), il faut aussi indiquer où placer ce texte, insérer des images, faire des liens entre les pages, etc.

### Le rôle de HTML et CSS

Pour expliquer à l'ordinateur ce que vous voulez faire, il va falloir utiliser un langage qu'il comprend. Et c'est là que les choses se corsent, parce qu'il va falloir apprendre deux langages !



Pourquoi avoir créé deux langages ? Un seul aurait suffi non ? 🤔

Vous devez vous dire que manipuler deux langages va être deux fois plus complexe et deux fois plus long à apprendre... mais non ! Je vous rassure, s'il y a deux langages c'est au contraire pour faciliter les choses. Nous allons avoir affaire à deux langages qui se *complètent* car ils ont des rôles différents :

- **HTML** (*HyperText Markup Language*) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce que vous souhaitez que la page affiche : du texte, des liens, des images... Vous direz par exemple : "Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc.".
- **CSS** (*Cascading Style Sheets*, aussi appelées *Feuilles de style*) : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleur, taille du texte...). Ce langage est venu compléter le HTML en 1996.

Vous avez peut-être aussi entendu parler du langage XHTML. Il s'agit d'une variante du HTML qui se veut plus rigoureuse et qui est donc un peu plus délicate à manipuler.



Pour faire simple, le HTML est apparu le premier en 1991. Début 2000, le W3C a lancé le XHTML en indiquant que ce serait l'avenir... mais le XHTML n'a pas percé comme on l'espérait. Retour aux sources en 2009 : le W3C abandonne le XHTML et décide de revenir au HTML pour le faire évoluer.

Il y a beaucoup de confusion autour de ces langages, alors qu'ils se ressemblent beaucoup. Aucun n'est vraiment meilleur que l'autre, il s'agit de deux façons de faire différentes. Dans ce cours, nous allons travailler sur la dernière version de HTML (HTML5) qui est aujourd'hui le langage d'avenir que tout le monde est incité à utiliser.

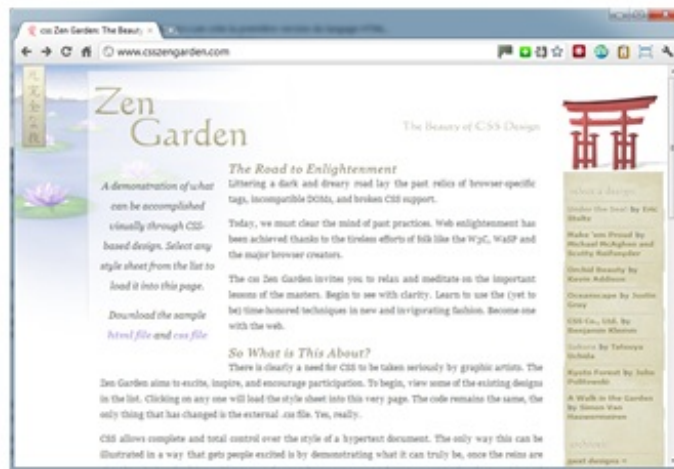
Vous pouvez très bien créer un site web uniquement en HTML, mais celui-ci ne sera pas très beau : l'information apparaîtra "brute". C'est pour cela que le langage CSS vient toujours le compléter.

Pour vous donner une idée, voici ce que donne la même page sans CSS puis avec le CSS :

HTML  
(pas de CSS)



HTML + CSS



La même page sans CSS et avec CSS

Le HTML définit le contenu (comme vous pouvez le voir, c'est brut de décoffrage !). Le CSS permet, lui, d'arranger le contenu et de définir la présentation : couleur, image de fond, marges, taille du texte...

Comme vous vous en doutez, le CSS a besoin d'une page HTML pour fonctionner. C'est pour cela que nous allons d'abord apprendre les bases du HTML avant de nous occuper de la décoration en CSS. 😊

Vos premières pages ne seront donc pas les plus esthétiques, mais qu'importe ! Ca ne durera pas longtemps.

## Les différentes versions de HTML et CSS

Au fil du temps, les langages HTML et CSS ont beaucoup évolué. Dans la toute première version de HTML (HTML 1.0) il n'était même pas possible d'afficher des images !

Voici un très bref historique des langages pour votre culture générale. 😊

### Les versions de HTML

- **HTML 1** : c'est la toute première version créée par Tim Berners-Lee en 1991.
- **HTML 2** : la deuxième version du HTML apparaît en 1994 et se finira en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des prochaines version du HTML. Les règles et le fonctionnement de cette version sont données par le W3C (tandis que la première version a été créée par un seul homme).
- **HTML 3** : apparue en 1996, cette nouvelle version du HTML rajoute 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 de 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 l'utilisation de feuilles de style, notre fameux CSS !

- **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. C'est cette version que nous allons découvrir ensemble.

### *Les versions de CSS*

- **CSS 1** : dès 1996, la première version du CSS est utilisable. 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 ensuite par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut 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.



Notez que HTML5 et CSS3 ne sont pas des versions encore totalement finalisées par le W3C. Cependant, même s'il peut y avoir des changements mineurs dans ces langages, je vous recommande chaudement de commencer dès aujourd'hui avec ces nouvelles versions. Leurs apports sont nombreux et valent vraiment le coup. D'ailleurs, de nombreux sites web professionnels se construisent aujourd'hui sur ces dernières versions.

---

## L'éditeur de texte

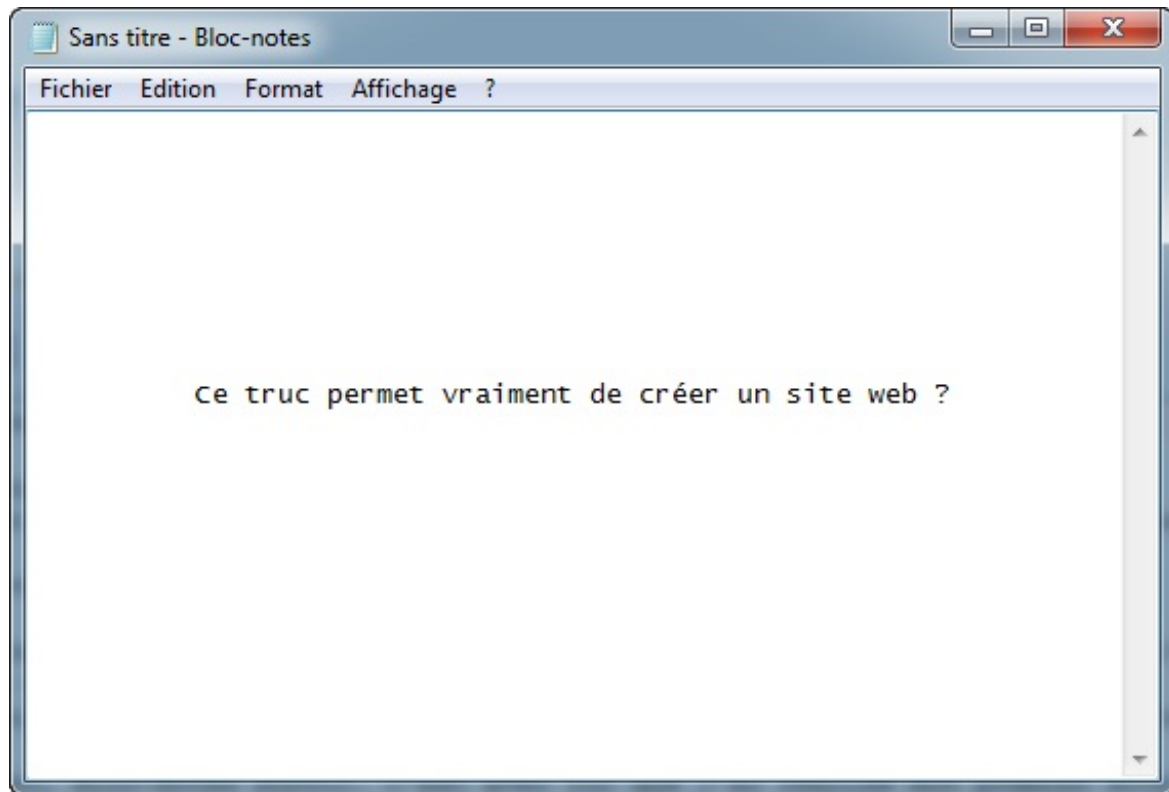


De quel logiciel je vais avoir besoin pour créer mon site web ?

Vais-je devoir casser ma tirelire pour acheter un logiciel très complexe que je vais mettre des mois à comprendre ?

Il existe effectivement de nombreux logiciels dédiés à la création de sites web. Mais, je vous rassure, vous n'aurez pas à déboursier un seul centime. Pourquoi aller chercher un logiciel payant et compliqué, alors que vous avez déjà tout ce qu'il faut chez vous ?

Eh oui, accrochez-vous bien parce qu'il suffit de... Bloc-Notes !



*Le logiciel Bloc-Notes*

Incroyable mais vrai : on peut tout à fait créer un site web juste avec Bloc-Notes, le logiciel d'édition de texte intégré par défaut sur Windows. D'ailleurs, j'avoue, c'est comme cela que j'ai commencé moi-même il y a quelques années. 😊

Il y a cependant des logiciels plus puissants aujourd'hui et personne n'utilise vraiment 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 programmer. Parmi les plus connus d'entre eux : Nvu, Microsoft Expression Web, Dreamweaver... et même Word ! Leur principal défaut est la qualité du code HTML et CSS qui est automatiquement généré par ces outils, souvent d'assez mauvaise qualité. Un bon créateur de site web doit tôt ou tard connaître HTML et CSS, c'est pourquoi je ne recommande pas l'usage de 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. Ils se révèlent être de puissants alliés pour les créateurs de sites web !

Vous l'aurez compris, je vais vous inviter à utiliser un éditeur de texte dans ce cours. Voici quelques conseils, selon que vous êtes sous Windows, Mac OS X ou Linux.

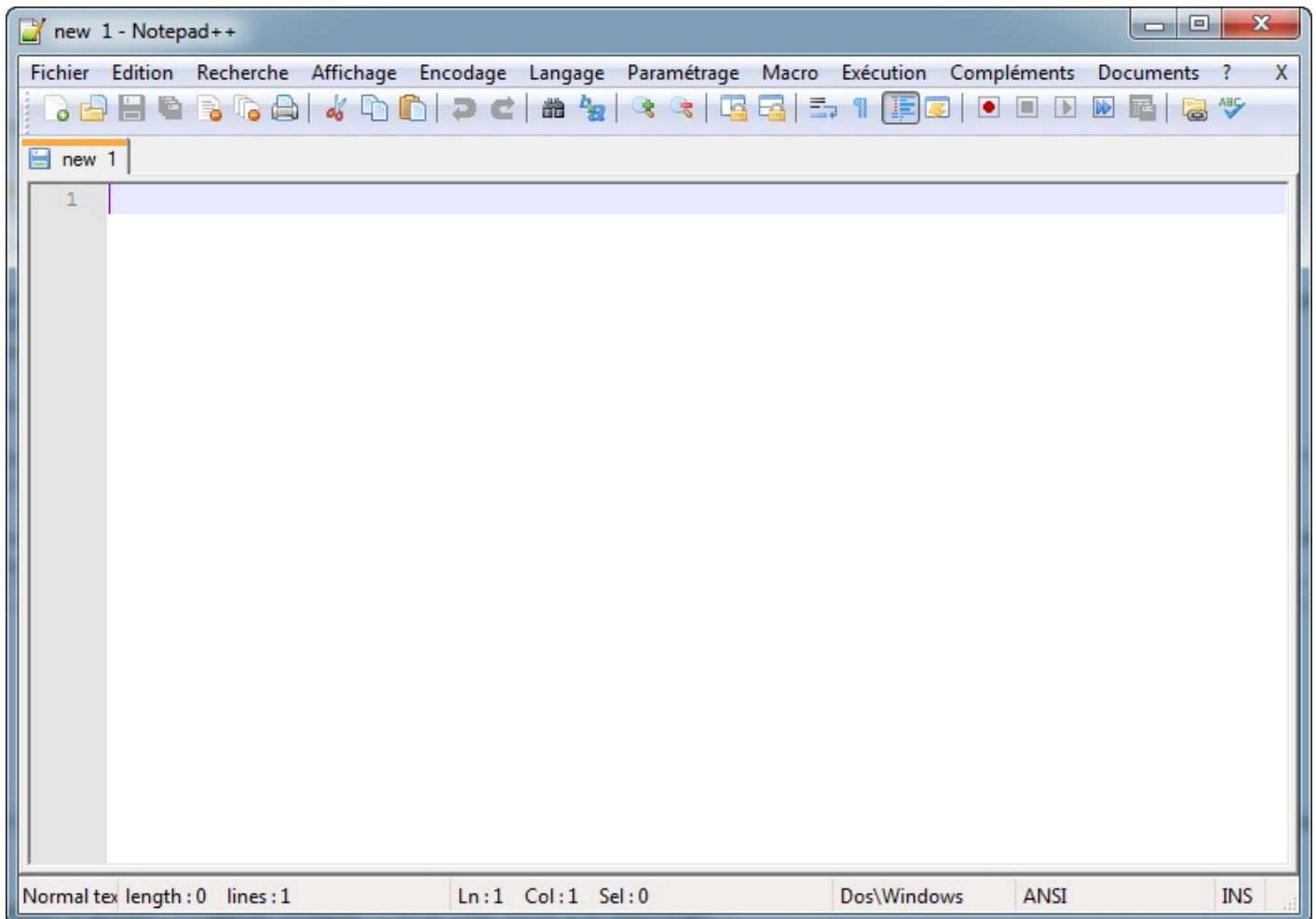
### Sous Windows

Il existe une grande quantité d'éditeurs de texte, je ne pourrai pas tous vous les présenter. Néanmoins, je vous invite à vous pencher sur Notepad++, l'un des plus utilisés d'entre eux sous Windows. Ce logiciel est simple, en français et gratuit.

### Site web de Notepad++

Prenez la version *Installer* et non le Zip

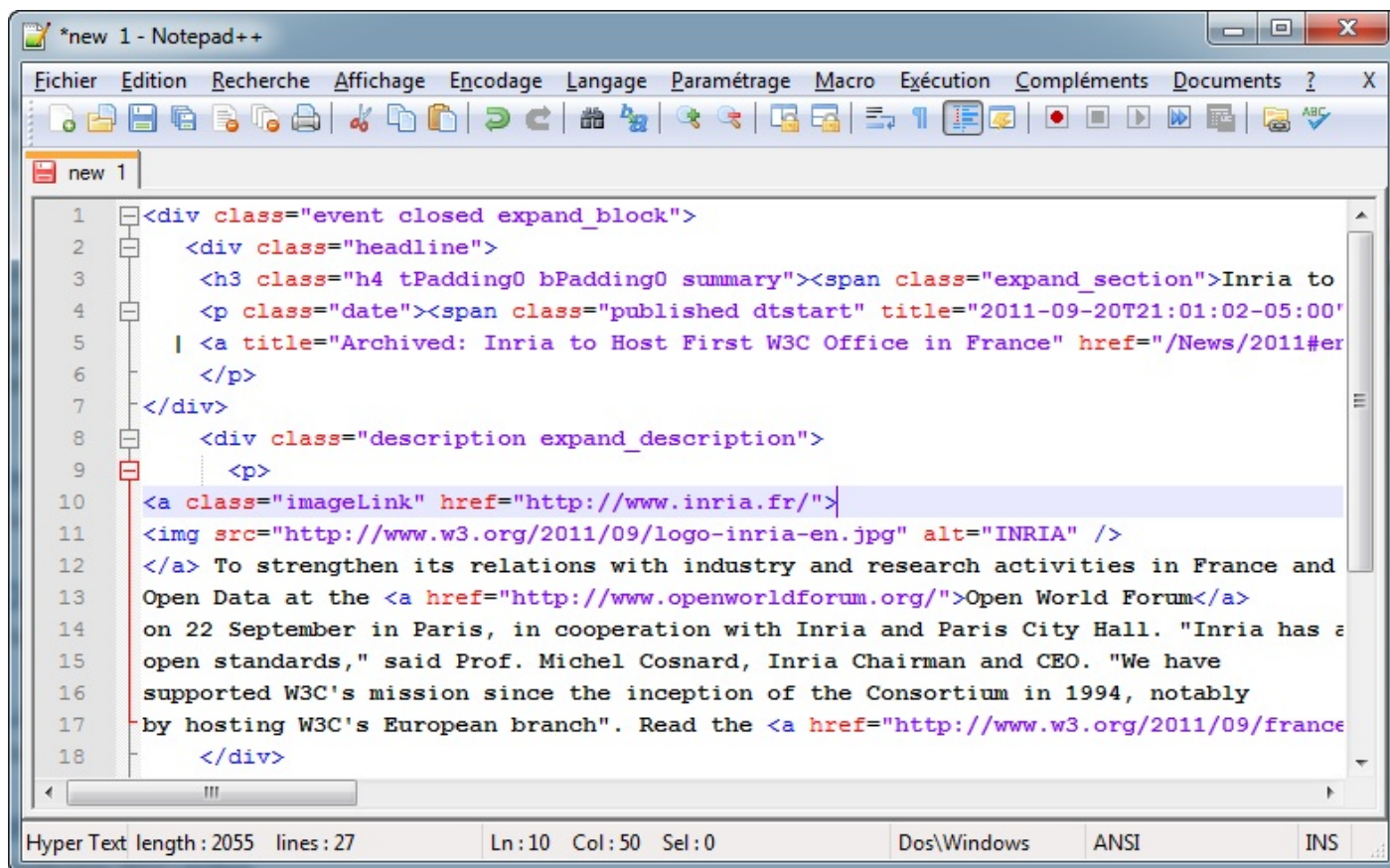
Voici à quoi ressemble Notepad++ lorsque vous le lancez :



Je vous conseille de faire la manipulation suivante : allez dans le menu "Langage" / "H" / "HTML". Cela permettra au logiciel de savoir que l'on va taper du HTML.

Lorsque vous utiliserez le logiciel, il colorera votre code ce qui vous permettra de vous repérer plus facilement :





```

1 <div class="event closed expand_block">
2   <div class="headline">
3     <h3 class="h4 tPadding0 bPadding0 summary"><span class="expand_section">Inria to
4     <p class="date"><span class="published dtstart" title="2011-09-20T21:01:02-05:00"
5     | <a title="Archived: Inria to Host First W3C Office in France" href="/News/2011#er
6     </p>
7   </div>
8   <div class="description expand_description">
9     <p>
10    <a class="imageLink" href="http://www.inria.fr/">
11    
12    </a> To strengthen its relations with industry and research activities in France and
13    Open Data at the <a href="http://www.openworldforum.org/">Open World Forum</a>
14    on 22 September in Paris, in cooperation with Inria and Paris City Hall. "Inria has a
15    open standards," said Prof. Michel Cosnard, Inria Chairman and CEO. "We have
16    supported W3C's mission since the inception of the Consortium in 1994, notably
17    by hosting W3C's European branch". Read the <a href="http://www.w3.org/2011/09/france
18    </div>

```



Pour l'instant, ne vous préoccupez pas de savoir ce que signifie tout ce charabia que vous pouvez voir. Je souhaitais simplement vous donner un aperçu des possibilités du logiciel.

D'autres éditeurs disponibles sous Windows existent. Si Notepad++ ne vous convient pas, vous pouvez essayer :

- [jEdit](#)
- [PSPad](#)
- [ConTEXT](#)
- ... et bien d'autres si vous recherchez "Editeur de texte" sur le Web. 😊

## Sous Mac OS X

Vous pouvez essayer l'un des logiciels suivants :

- [jEdit](#)
- [Smultron](#)
- [TextWrangler](#)

## Sous Linux

Les éditeurs de texte sont légion sous Linux. Certains d'entre eux sont installés par défaut, d'autres peuvent être téléchargés facilement avec le centre de téléchargement (sous Ubuntu notamment) ou via des commandes comme `apt-get` et `aptitude`. Voici quelques logiciels que vous pouvez tester :



- 
- gEdit
  - Kate
  - vim
  - Emacs
  - jEdit
- 
-

## Les navigateurs

### Pourquoi le navigateur est important

Le navigateur est le programme qui nous permet de voir les sites web. Si vous lisez ces lignes, c'est donc que votre navigateur est ouvert et que vous l'avez sous les yeux. 😊






Comme je vous l'ai expliqué plus tôt, le travail du navigateur est de lire le code HTML et CSS pour afficher un résultat visuel à l'écran. Si votre code CSS dit "Les titres sont en rouge", alors le navigateur affichera les titres en rouge. Le rôle du navigateur est donc essentiel !

On ne dirait pas, mais un navigateur est un programme extrêmement complexe. Comprendre le code HTML et CSS n'est en effet pas une mince affaire. Le principal problème, vous vous en rendez vite compte, c'est que **les navigateurs n'affichent pas tous les sites exactement de la même façon** ! Il faudra vous y faire et prendre l'habitude de vérifier régulièrement que votre site fonctionne correctement sur la plupart des navigateurs.

### Les navigateurs sur ordinateur

#### Téléchargez les navigateurs

Il existe de nombreux navigateurs différents. Voici les principaux à connaître :

Navigateur	OS	Téléchargement	Commentaires
<b>Google Chrome</b> 	Windows Mac Linux	<a href="#">Téléchargement</a>	Le navigateur de Google, simple d'emploi et très rapide. <b>C'est le navigateur que j'utilise au quotidien.</b>
<b>Mozilla Firefox</b> 	Windows Mac Linux	<a href="#">Téléchargement</a>	Le navigateur de la fondation Mozilla, célèbre et réputé. Je l'utilise fréquemment pour tester mes sites web.
<b>Internet Explorer</b> 	Windows	<a href="#">Téléchargement</a> (Déjà installé sur Windows)	Le navigateur de Microsoft, qui équipe tous les PC Windows. Je l'utilise fréquemment pour tester mes sites web.
<b>Safari</b> 	Windows Mac	<a href="#">Téléchargement</a> (Déjà installé sur Mac OS X)	Le navigateur d'Apple, qui équipe tous les Mac.
<b>Opera</b> 	Windows Mac Linux	<a href="#">Téléchargement</a>	L'éternel <i>outsider</i> . Il est moins utilisé mais propose de nombreuses fonctionnalités.

**Il est conseillé d'installer plusieurs navigateurs sur son ordinateur** pour s'assurer que son site fonctionne correctement sur chacun d'eux. De manière générale, je conseille de tester son site web régulièrement au moins sur Google Chrome, Mozilla Firefox et Internet Explorer.

Notez que Safari et Google Chrome affichent les sites web quasiment de la même façon. Il n'est pas forcément nécessaire de tester son site sur Safari et Google Chrome, même si c'est toujours plus sûr. 😊

Voici un aperçu de quelques-uns de ces principaux navigateurs sur la page d'accueil de Google :



### *Comprendre les différences entre navigateurs*

Comme je vous le disais plus tôt, les navigateurs n'affichent pas toujours les sites web *exactement* de la même façon. Pourquoi ? Cela est dû au fait que les navigateurs ne connaissent pas toujours les dernières fonctionnalités de HTML et CSS. Par exemple, Internet Explorer a longtemps été en retard sur certaines fonctionnalités CSS (et paradoxalement, il a aussi été en avance sur quelques autres).






Pour compliquer les choses, plusieurs versions des navigateurs co-existent :

- Firefox 2, Firefox 3.5, Firefox 3.6, Firefox 4
- Internet Explorer 6, Internet Explorer 7, Internet Explorer 8, Internet Explorer 9
- Chrome 8, Chrome 9, Chrome 10
- ...

Chaque version prend en charge de nouvelles fonctionnalités, mais si les utilisateurs ne mettent pas à jour leurs navigateurs cela devient un problème pour les *webmasters* comme vous qui créez 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. Firefox a des utilisateurs qui ne pensent pas à se mettre à jour, et Internet Explorer a du mal à inciter à se mettre à jour car les dernières versions nécessitent aussi de mettre à jour Windows (Internet Explorer 9 n'est pas disponible pour Windows XP par exemple).

Des sites comme [normansblog.de](http://normansblog.de) tiennent notamment à jour une liste des fonctionnalités CSS supportées par les différentes versions de chaque navigateur :

Browser Rendering Engine	 Firefox Gecko					 Safari Webkit			 Chrome Webkit		 Internet Explorer Trident			 Opera Presto	
Version	3.6	4	5	6	7b – 8a	4	5	5.1	9	10 – 14	6 – 8	9	10a3	10.6 – 11	11.1 – 11.5
Animations	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗
Background Gradients	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	▲
Background Size	✓	✓	✓	✓	✓	▲	✓	✓	✓	✓	✗	✓	✓	✓	✓
Border Image	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Border Radius	▲	▲	▲	▲	▲	▲	✓	✓	✓	✓	✗	✓	✓	✓	✓
Box Shadow	✓	✓	✓	✓	✓	▲	✓	✓	▲	✓	✗	✓	✓	✓	✓
Columns	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓
Font Face	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	▲	✓	✓	✓	✓
HSLa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Hyphens	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Multiple Backgrounds	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Opacity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

Liste des fonctionnalités supportées par les navigateurs

Comme vous le voyez, c'est... compliqué.

Le plus gros des soucis viendra le plus souvent des anciennes versions d'Internet Explorer (IE6, IE7, IE8). Il faudra vérifier sous ces anciennes versions comment son site s'affiche... Attendez-vous à des surprises ! Vérifiez surtout que votre site s'affiche sans erreurs, sans chercher à obtenir exactement le même rendu sur les vieilles versions de ces navigateurs.



Il existe un programme appelé **IETester** sous Windows. Il permet de tester son site sous différentes versions d'Internet Explorer. A noter que ce programme est relativement instable (il plante souvent) mais il a le mérite d'exister.

## Les navigateurs sur mobile

En plus des navigateurs que je vous ai présentés, il faut savoir qu'il existe des variantes de ces navigateurs que l'on retrouve sur les téléphones portables, en particulier les *smartphones*.

De plus en plus de personnes consultent aujourd'hui des sites web sur leur portable, il faut donc connaître un minimum le fonctionnement des navigateurs des téléphones.

En fait, vous n'allez pas être dépayés : 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* mais 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 Safari et Chrome.



Safari Mobile sur iPhone

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

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



Les tablettes tactiles sont équipées des mêmes navigateurs, l'écran étant simplement plus large. Ainsi, l'iPad est équipé de Safari Mobile par exemple.

---

Ainsi se termine notre premier chapitre 😊

Nous avons fait tous les préparatifs nécessaires : nous sommes maintenant prêts à rédiger notre première page en HTML dans le prochain chapitre !



## Votre première page web en HTML

Ca y est, vous avez installé tous les logiciels ?

Vous devriez maintenant avoir un éditeur de texte pour *créer votre site* (comme Notepad++) et plusieurs navigateurs pour le *tester* (Mozilla Firefox, Google Chrome, Internet Explorer...).

Dans ce chapitre, nous allons commencer à pratiquer ! Nous allons découvrir les bases du langage HTML et enregistrer notre toute première page web ! 😊

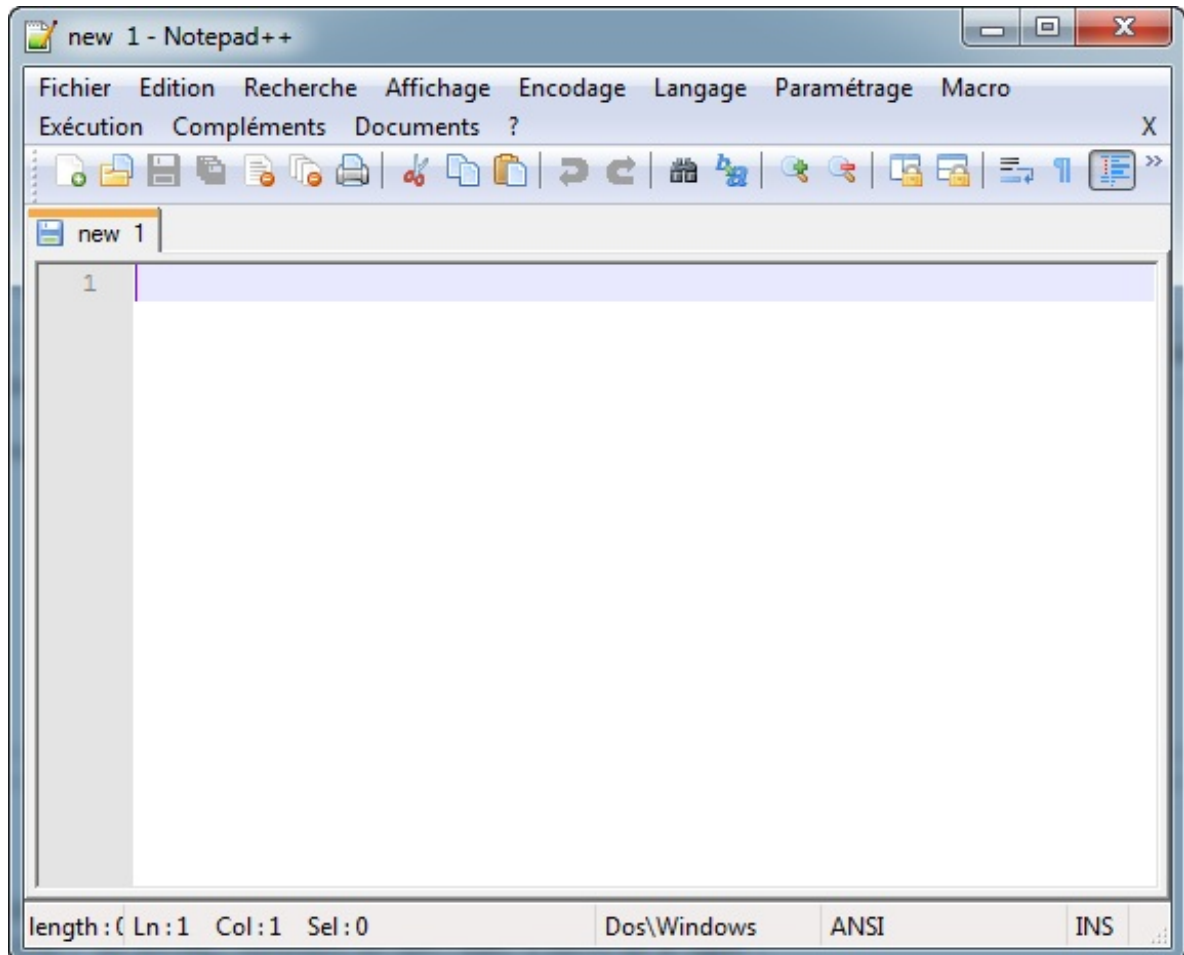
Alors oui, bien sûr, ne vous attendez pas encore à réaliser une page web exceptionnelle dès ce second chapitre, mais patience... ça viendra ! 😊

---

## Créer une page web avec l'éditeur

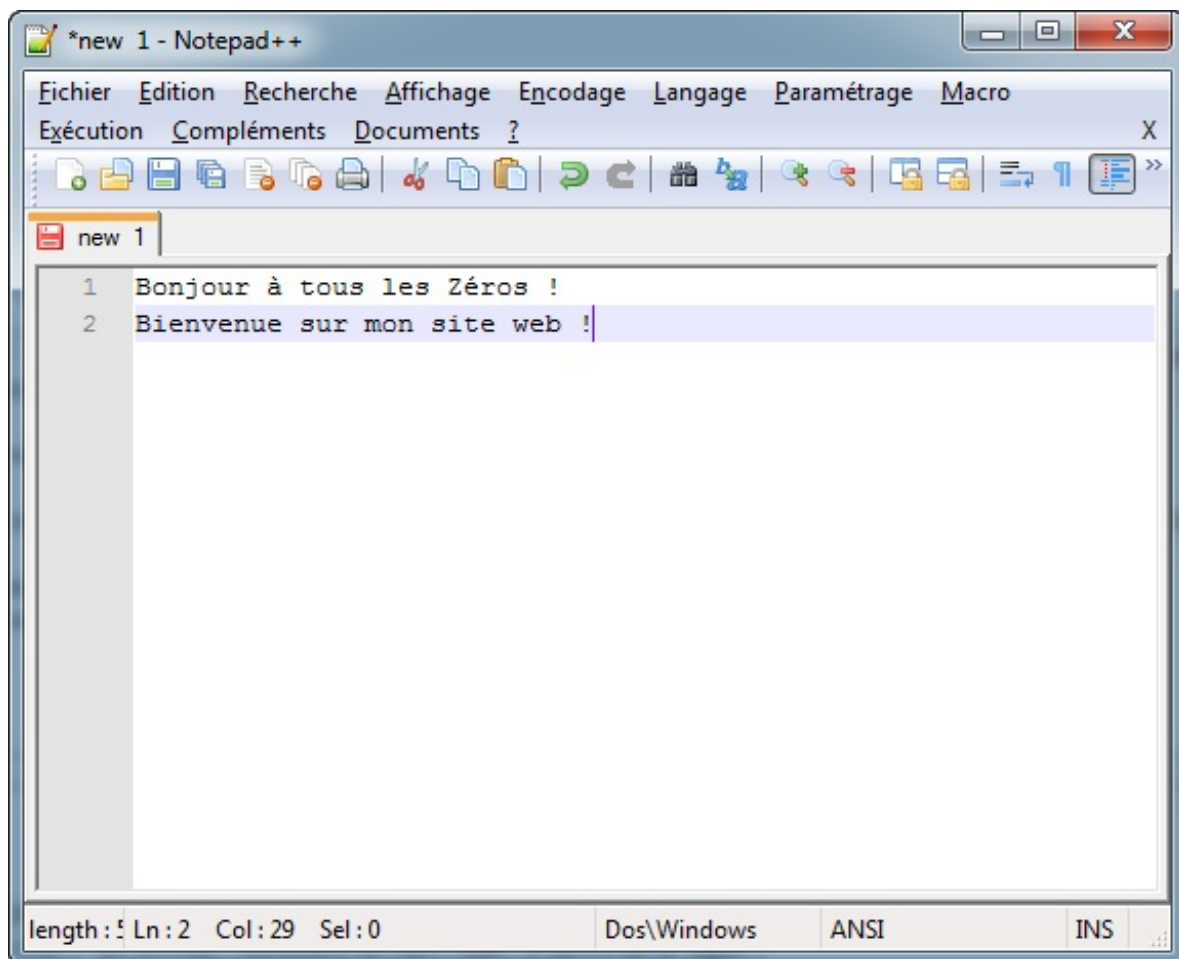
Allez, mettons-nous en situation ! Comme je vous l'ai dit, nous allons créer notre site dans un éditeur de texte. Vous avez dû en installer suite à mes conseils dans le premier chapitre : qu'il s'appelle Notepad++, PSpad, jEdit, vim, TextWrangler... peu importe. Ces logiciels ont un but très simple : vous permettre d'écrire du texte !

Dans la suite de ce cours, je travaillerai sous Notepad++. Je vais donc l'ouvrir :



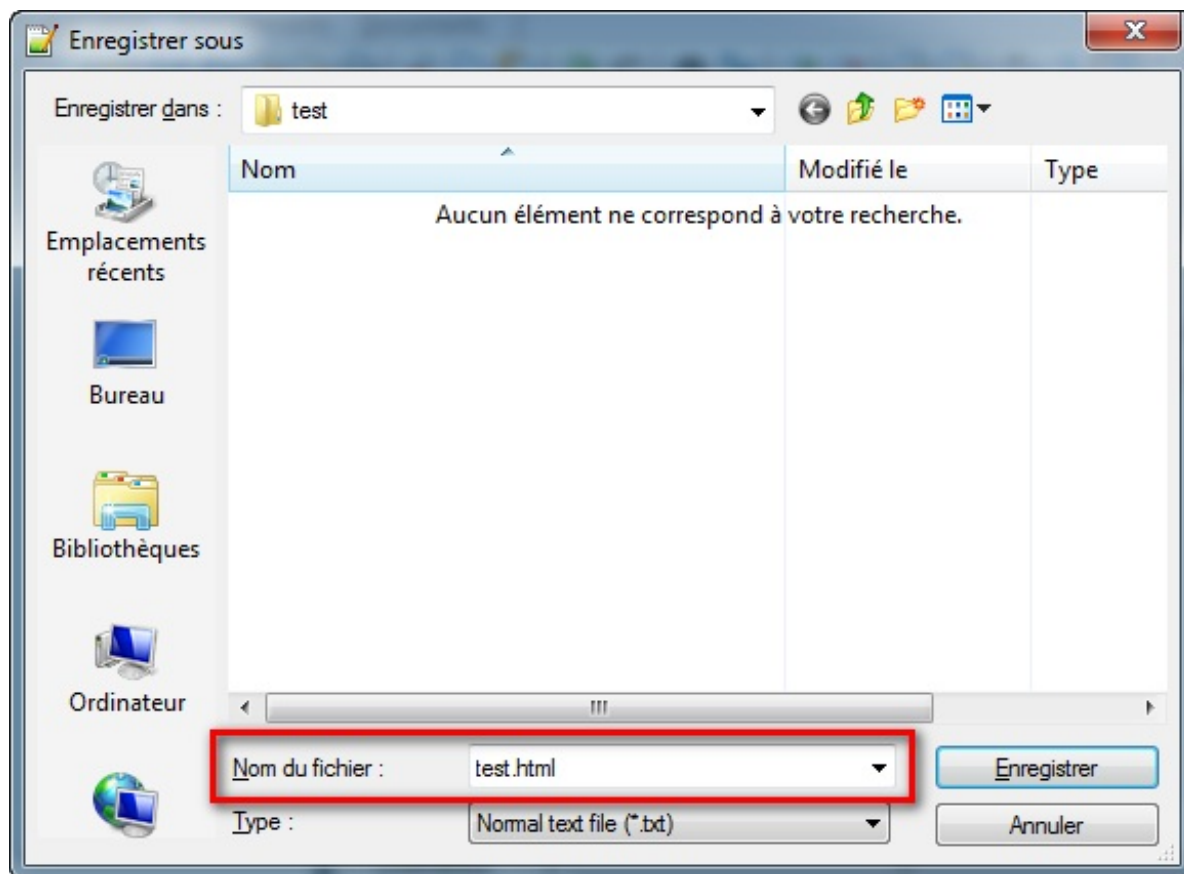
Bon, qu'est-ce qu'on fait maintenant ? Qu'est-ce qu'on écrit sur cette feuille blanche ?

On va faire un petit essai. Je vous invite à écrire ce qui vous passe par la tête :



Vous pouvez écrire les mêmes phrases que moi ou ce que vous voulez ; le but est d'écrire quelque chose. 😊

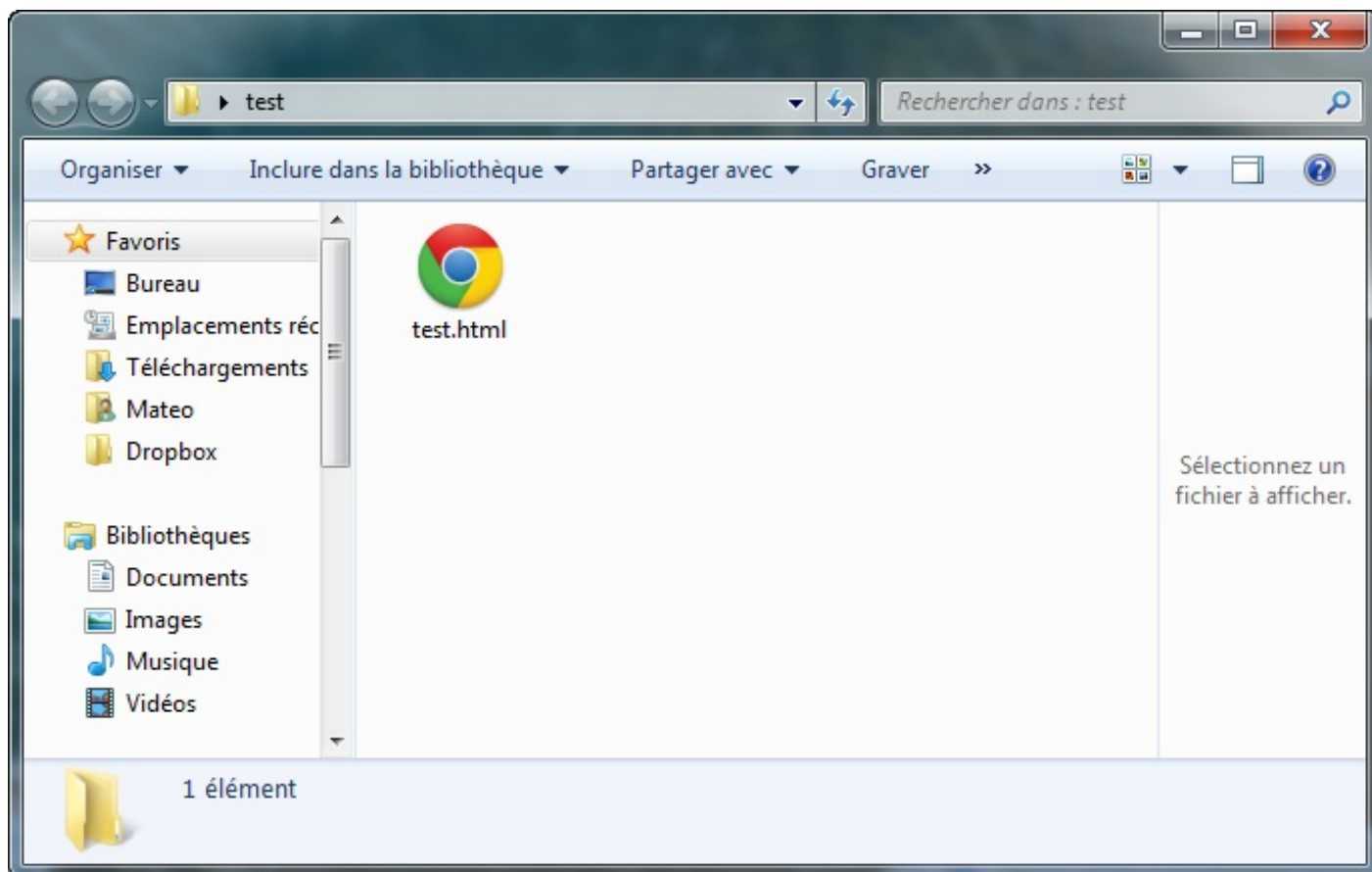
Maintenant, enregistrons ce fichier. Pour ça, c'est très simple : comme dans tous les programmes, vous avez un menu **Fichier / Enregistrer**. Une boîte de dialogue vous demande où enregistrer le fichier et sous quel nom. Enregistrez-le où vous voulez. Donnez au fichier le nom que vous voulez, en terminant par `.html`, par exemple : `test.html`.



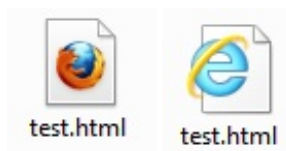
Je vous recommande de créer un nouveau dossier dans vos documents qui contiendra les fichiers de votre site. Pour ma part j'ai créé un dossier `test` dans lequel j'ai mis mon fichier `test.html`.

Ouvrez maintenant l'explorateur de fichiers dans le dossier où vous avez enregistré votre page. Vous y verrez le fichier que vous venez de créer :

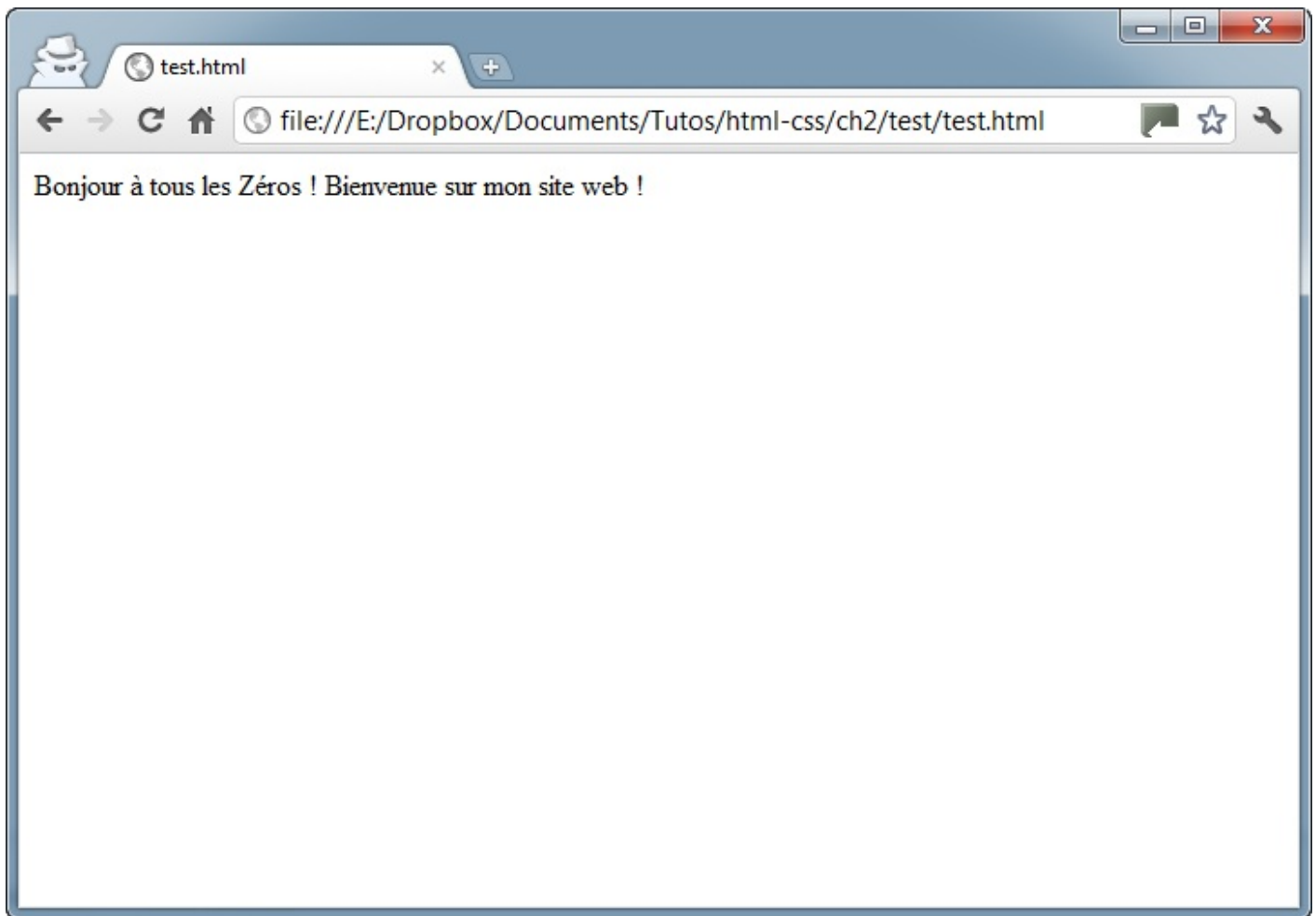




L'apparence du fichier dépend de votre navigateur web par défaut. Ici, l'icône est celle de Google Chrome, mon navigateur par défaut, mais le fichier a peut-être une autre icône chez vous. Voici par exemple l'icône qui apparaît selon que votre navigateur principal est Firefox ou Internet Explorer :



Double-cliquez simplement sur ce fichier et... votre navigateur s'ouvre en affichant le texte que vous avez écrit !



Ca ne marche pas bien on dirait ! Tout le texte s'affiche sur la même ligne alors qu'on avait écrit 2 lignes de texte différentes !?

En effet, bien vu ! 😏

Le texte s'affiche sur la même ligne alors qu'on avait demandé à l'écrire sur 2 lignes différentes. Que se passe-t-il ?

En fait, pour créer une page web il ne suffit pas simplement de taper du texte comme on vient de le faire. En plus de ce texte, il faut aussi écrire ce qu'on appelle des **balises**, qui vont donner des instructions à l'ordinateur comme "aller à la ligne", "afficher une image", etc.

## Les balises et leurs attributs

Bon, tout ça était trop facile. Evidemment, il a fallu que ces satanés informaticiens s'en mêlent et compliquent les choses. Il ne suffit pas "simplement" d'écrire du texte dans l'éditeur, il faut aussi donner des instructions à l'ordinateur. En HTML, on passe pour cela par des **balises**.

### Les balises

Les pages HTML sont remplies de ce qu'on appelle des balises. Celles-ci sont invisibles à l'écran pour vos visiteurs, mais elles permettent à l'ordinateur de comprendre ce qu'il doit afficher.

Les balises se repèrent facilement. Elles sont entourées de "chevrons", c'est-à-dire des symboles `<` et `>`, comme ceci :

`<balise>`

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

On distingue deux types de balises : les balises en paires et les balises orphelines.

#### *Les balises en paires*

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Voici à quoi elles ressemblent :

Code : HTML

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

On distingue une balise ouvrante (`<titre>`) et une balise fermante (`</titre>`) 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.

Code : HTML

```
Ceci n'est pas un titre <titre>Ceci est un titre</titre> Ceci n'est pas un titre
```

#### *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, on veut juste dire à l'ordinateur "Insère une image ici".

Une balise orpheline s'écrit comme ceci :

Code : HTML

```
<image />
```



Notez que le `/` de fin n'est pas obligatoire. On pourrait écrire seulement `<image>`. Néanmoins, pour ne pas confondre avec le premier type de balise, les webmasters recommandent de rajouter ce `/` (slash) à la fin de la balise. Vous devrez donc mettre un `/` aux balises orphelines et je vous recommande de faire de même, c'est une bonne pratique.

## Les attributs

Les attributs sont un peu les options des balises. Ils viennent les compléter pour donner des informations supplémentaires. L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

Code : HTML

```
<balise attribut="valeur">
```

A quoi ça sert ? Prenons la balise `<image />` que nous venons de voir. Seule, elle ne sert pas à grand chose. On pourrait rajouter un attribut qui indique le nom de l'image à afficher :

Code : HTML

```
<image nom="photo.jpg" />
```

L'ordinateur comprend alors qu'il doit afficher l'image contenue dans le fichier `photo.jpg`.

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. Par exemple, ce code indique que la citation est de Neil Armstrong et qu'elle date du 21 Juillet 1969 :

Code : HTML

```
<citation auteur="Neil Armstrong" date="21/07/1969">  
C'est un petit pas pour l'homme un bond de géant pour l'humanité.  
</citation>
```



Toutes les balises que nous venons de voir sont fictives. Les vraies balises ont des noms en anglais (eh oui !), nous allons les découvrir dans la suite de ce cours. 😊

## Structure de base d'une page HTML5

Reprenons notre éditeur de texte (dans mon cas Notepad++). Je vous invite à copier-coller le code source ci-dessus dans Notepad++. Ce code correspond à la base d'une page web en HTML5 :

### Code : HTML

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

  <body>

</body>
</html>
```



J'ai mis des espaces au début de certaines lignes pour "décaler" les balises. Ce n'est pas obligatoire et ça n'a aucun impact sur l'affichage de la page, mais ça permet de rendre le code source plus lisible. On appelle ça l'indentation. Dans votre éditeur, il suffit d'appuyer sur la touche Tab pour avoir le même résultat.

Copié dans Notepad++, cela donne :

The screenshot shows the Notepad++ application window with the file 'test.html' open. The code is as follows:

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

The status bar at the bottom indicates 'length: 11 Ln: 11 Col: 8 Sel: 0'. The menu bar includes 'Fichier', 'Edition', 'Recherche', 'Affichage', 'Encodage', 'Langage', 'Paramétrage', and 'Macro'. The toolbar contains various icons for file operations and editing.

Vous noterez que les balises s'ouvrent et se ferment dans un ordre 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.



Euh, on pourrait avoir des explications sur toutes les balises que l'on vient de copier dans notre éditeur m'sieur ?

Bien sûr, c'est demandé si gentiment. 😊

Ne prenez pas peur en voyant toutes ces balises d'un coup, je vais vous expliquer leur rôle !

## Le doctype

Code : HTML

```
<!DOCTYPE html>
```

La toute première ligne s'appelle le *doctype*. Elle 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), vous pouvez considérer que c'est un peu l'exception qui confirme la règle (ça commence bien 😊).

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">
```

Pour HTML5, il a été décidé de la simplifier, pour le plus grand bonheur des webmasters. Quand vous voyez une balise doctype courte (`<!DOCTYPE html>`), cela signifie que la page est écrite en HTML5.

## La balise <html>

Code : HTML

```
<html>

</html>
```

C'est la balise principale du code. Elle englobe tout le contenu de votre page. Comme vous pouvez le voir, la balise fermante `</html>` se trouve tout à la fin du code !

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

Une page web est constituée de 2 parties :

- L'en-tête **<head>** : cette section donne quelques informations générales sur la page, comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que l'en-tête contient ne sont pas affichées sur la page, ce sont simplement des informations générales à destination de l'ordinateur. Elles sont cependant très importantes !
- Le corps **<body>** : c'est là que se trouve la partie principale de la page. Tout ce que nous écrivons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrivons 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...

### *L'encodage (charset)*

#### Code : HTML

```
<meta charset="utf-8" />
```

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, symboles arabes, etc.).

Il y a plusieurs techniques d'encodages aux noms bizarres utilisées en fonction des langues : ISO-8859-1, OEM 775, Windows-1253... Un seul cependant devrait être utilisé 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 j'ai indiqué utf-8 dans cette balise.

Il faut aussi que votre 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.



Sous Notepad++, allez dans le menu Encodage > Encoder en UTF-8 (sans BOM) pour que votre fichier soit enregistré en UTF-8 dès le début. Cela ne s'applique qu'au fichier actuellement ouvert.

Pour ne pas avoir à le faire pour chaque nouveau fichier, je vous conseille d'aller dans le menu Paramétrage > Préférences, onglet Nouveau document/Dossier. Sélectionnez UTF-8 sans BOM dans la liste.



Si vous avez un problème d'affichage des accents plus tard sur votre page web, c'est qu'il y a un problème avec l'encodage. Vérifiez que la balise indique bien UTF-8 et que votre fichier est enregistré en UTF-8 (votre éditeur de texte est capable de vous le dire, Notepad++ le fait dans le menu Encodage).

### *Le titre principal de la page*

#### Code : HTML

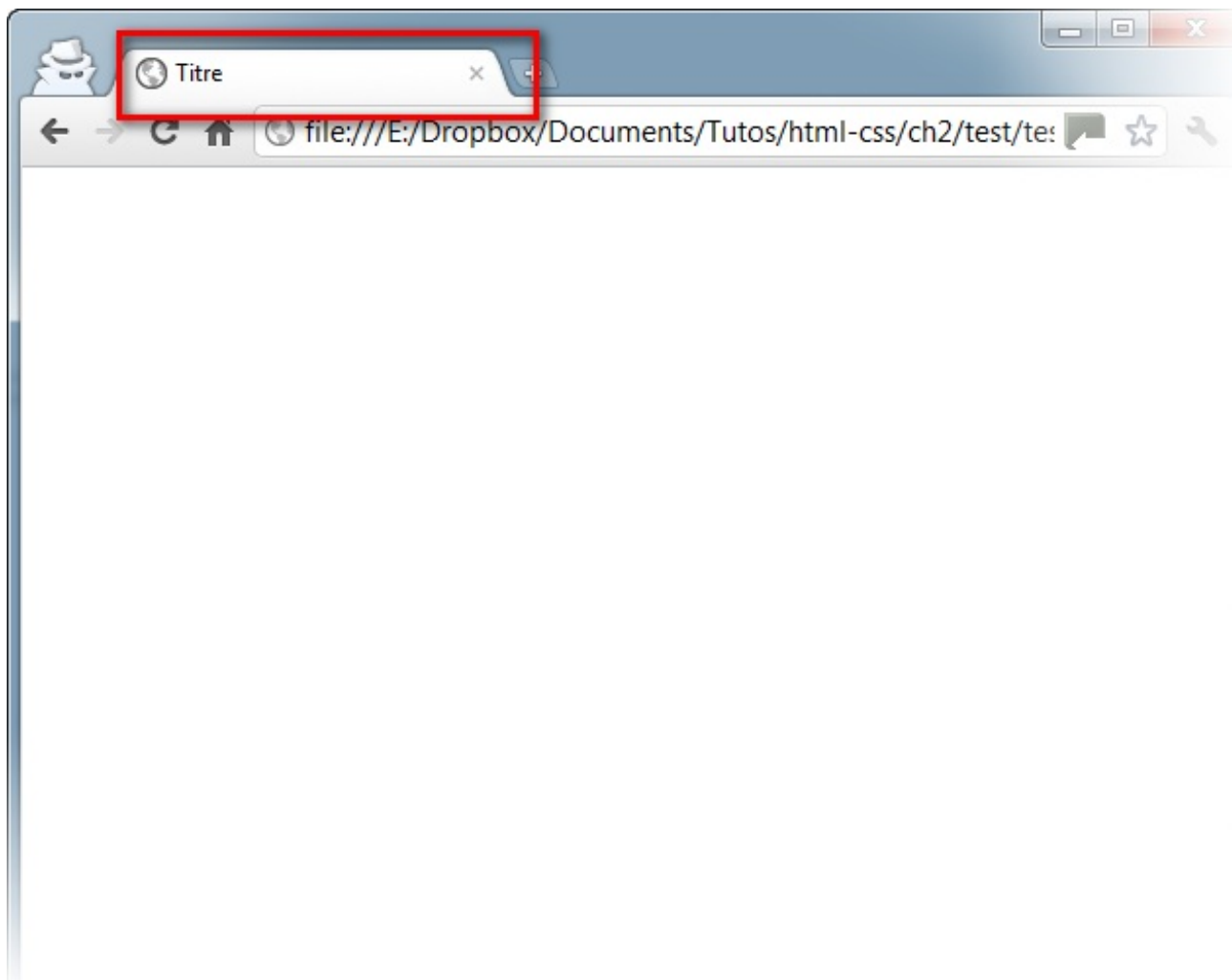
```
<title>
```

C'est le titre de votre page, probablement l'élément le plus important ! Toute page doit avoir un titre qui décrit ce qu'elle

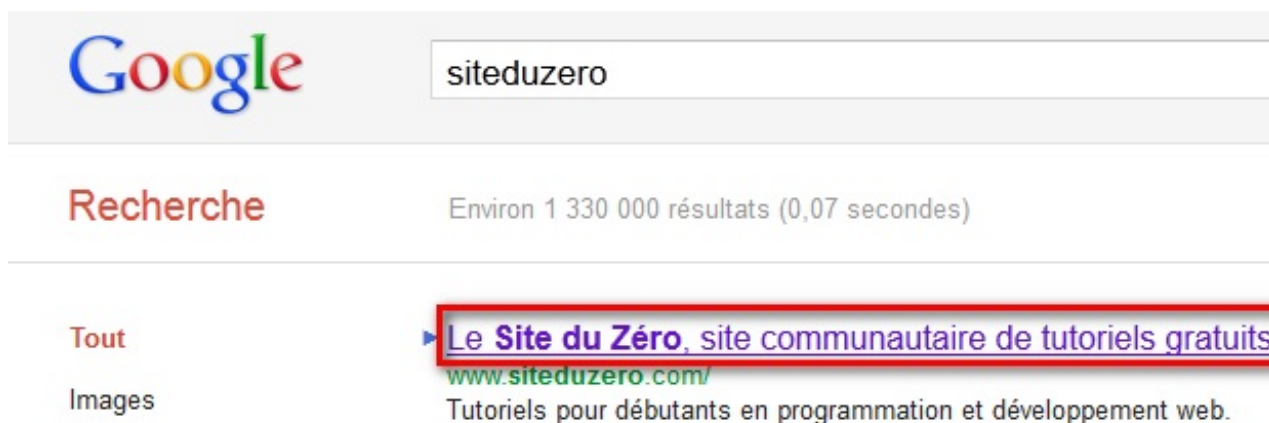
contient.

Il est conseillé que le titre soit assez court (moins de 100 caractères en général).

Le titre ne s'affiche pas dans votre page mais en haut de celle-ci (souvent dans l'onglet du navigateur). Enregistrez votre page web et ouvrez-la dans votre navigateur. Vous verrez que le titre s'affiche dans l'onglet :



Il faut savoir que le titre apparaît aussi dans les résultats de recherche, comme ici sur Google :



Autant vous dire que bien choisir son titre est important ! 😊



## Les commentaires

Nous avons appris à créer notre première *vraie* page HTML dans ce chapitre. Avant de terminer, j'aimerais vous présenter le principe des 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, ça ne change rien à l'affichage de la page.



Bref, ça ne sert à rien ?

Eh bien si ! 😊

Ca sert pour *vous* et les personnes qui liront le code source de votre page. Vous pouvez utiliser les commentaires pour laisser des indications sur le fonctionnement de votre page.

Quel intérêt ? Cela vous permettra de vous souvenir comment fonctionne votre page si vous revenez sur votre code source après un long moment d'absence. Ne rigolez pas, ça arrive à tous les webmasters. 😊

### Insérer un commentaire

Un commentaire est une balise HTML avec une forme bien spéciale :

Code : HTML

```
<!-- Ceci est un commentaire -->
```

Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source (surtout s'il est long).

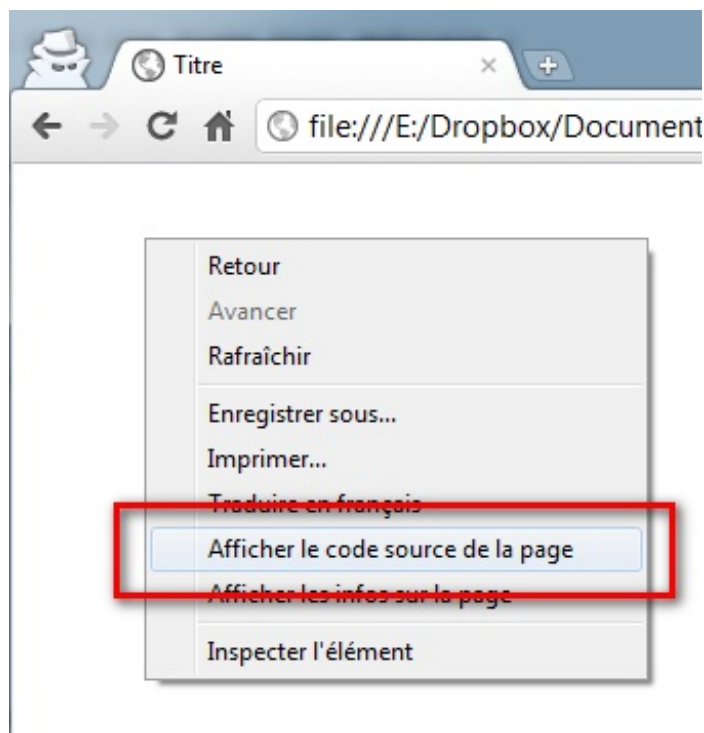
Code : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <!-- En-tête de la page -->
    <meta charset="utf-8" />
    <title>Titre</title>
  </head>

  <body>
    <!-- Corps de la page -->
  </body>
</html>
```

### Tout le monde peut voir vos commentaires... et tout votre code HTML !

Terminons par une remarque importante : **tout le monde peut voir le code HTML de votre 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 votre navigateur) :



Le code source s'affiche alors :



Vous pouvez tester sur n'importe quel site web, ça marche ! Garanti à 100%. Cela s'explique assez facilement : le navigateur *doit* obtenir le code HTML pour savoir ce qu'il faut afficher. Le code HTML de tous les sites est donc public.

La morale de l'histoire ? Tout le monde pourra voir votre code HTML et vous ne pouvez pas l'empêcher. Par conséquent, ne mettez pas d'informations sensibles comme des mots de passe dans les commentaires... et soignez votre code source, car je pourrai venir vérifier si vous avez bien suivi mon tutoriel à la lettre ! 😊



Ne prenez pas peur en regardant le code de certains sites web s'il vous paraît long ou ne pas respecter les mêmes règles que celles que je vous présente dans ce tutoriel. Tous les sites ne sont pas écrits en HTML5 (loin de là), et parfois certains webmasters écrivent très mal leur code, ce ne sont pas toujours des exemples à suivre !

Nous avons créé une toute première page web, mais pour le moment celle-ci est blanche.  
Dans le prochain chapitre, nous allons commencer à rédiger le contenu de notre page !

## Organiser son texte

Bon, la page blanche c'est bien joli, mais votre site web risque d'avoir un succès mitigé si vous le laissez comme ça. 😊

Nous allons voir comment rédiger le contenu de notre page web dans ce chapitre. Comme nous l'avons vu, il ne faudra pas faire ça n'importe comment : il ne faut pas oublier qu'une page HTML est composée de balises. Ces balises indiquent à l'ordinateur le sens du texte : ceci est un paragraphe, ceci est un titre, etc.

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 de son texte.
- Comment organiser les informations sous forme de liste à puces.

Motivés ? Allez, vous allez voir, ce n'est pas compliqué. 😊

---

## 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.

### Code : HTML

```
<p>Bonjour et bienvenue sur mon site !</p>
```

- `<p>` signifie "Début du paragraphe"
- `</p>` signifie "Fin du paragraphe"

Comme je vous l'ai dit dans le chapitre précédent, on écrit le contenu de notre 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 !

Je reprends donc exactement le même code que dans le chapitre précédent, et j'y ajoute mon paragraphe :

### Code : HTML

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

  <body>
    <p>Bonjour et bienvenue sur mon site !</p>
  </body>
</html>
```

[Essayer !](#)

Essayez, vous allez voir le résultat !

Bon, ok c'est pas encore le nirvana, mais c'est un bon début. 😊

Mais ne nous arrêtons pas en si bon chemin. Nous allons voir maintenant quelque chose d'un peu particulier en HTML : le saut de lignes. Ça a l'air simple, mais pourtant ça ne fonctionne pas vraiment comme dans un traitement de texte habituel...

## Sauter une ligne

En HTML, si vous appuyez sur la touche Entrée, ça ne crée pas une nouvelle ligne comme vous en avez l'habitude. Essayez donc ce code :

### Code : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Essais de sauts de ligne</title>
```

```
</head>

<body>
  <p>
    Bonjour et bienvenue sur mon site !
    Ceci est mon premier test, alors soyez indulgents s'il
vous plaît, j'apprends petit à petit comment ça marche.
    Pour l'instant c'est un peu vide, mais revenez dans 2-3
jours quand j'aurai appris un peu plus de choses, je vous assure que
vous allez être surpris !
  </p>
</body>
</html>
```

[Essayer !](#)

Tout est sur la même ligne alors qu'on est pourtant allé à la ligne dans notre code ! Taper frénétiquement sur la touche Entrée dans l'éditeur de texte ne sert donc strictement à rien. 🤪

Comme vous devez vous en douter, il y a pourtant bien un moyen de faire des sauts de ligne en HTML.

En fait, si vous voulez écrire un deuxième paragraphe, il vous suffit d'utiliser une deuxième balise `<p>`. Votre code HTML devrait donc être au final rempli de balises de paragraphe !

Un exemple :

#### Code : HTML

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Paragraphe</title>
  </head>

  <body>
    <p>
      Bonjour et bienvenue sur mon site !
      Ceci est mon premier test, alors soyez indulgents s'il
vous plaît, j'apprends petit à petit comment ça marche.
    </p>

    <p>
      Pour l'instant c'est un peu vide, mais revenez dans 2-3
jours quand j'aurai appris un peu plus de choses, je vous assure que
vous allez être surpris !
    </p>
  </body>
</html>
```

[Essayer !](#)



Oui, mais si je veux juste aller à la ligne dans un paragraphe, et non pas sauter une ligne ?

Eh bien devinez quoi : il existe une balise "Aller à la ligne" !

C'est une balise *orpheline* qui sert juste à indiquer qu'on doit aller à la ligne : `<br />`. Vous devez obligatoirement la mettre à l'intérieur d'un paragraphe.

Voici comment l'utiliser dans un code :

#### Code : HTML

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Sauts de ligne</title>
  </head>

  <body>
    <p>
      Bonjour et bienvenue sur mon site !<br />
      Ceci est mon premier test, alors soyez indulgents s'il
      vous plaît, j'apprends petit à petit comment ça marche.
    </p>

    <p>
      Pour l'instant c'est un peu vide, mais revenez dans 2-3
      jours quand j'aurai appris un peu plus de choses, je vous assure que
      vous allez être surpris !
    </p>
  </body>
</html>
```

[Essayer !](#)



Vous pouvez théoriquement mettre plusieurs balises `<br />` d'affilée pour faire plusieurs sauts de lignes, mais on considère que c'est une mauvaise pratique qui rend le code délicat à maintenir. Pour décaler un texte avec plus de précision, on utilisera le CSS, ce langage qui vient compléter le HTML et dont je vous parlerai un peu plus loin.

Donc c'est compris ?

- `<p> </p>` : pour organiser son texte en paragraphes.
- `<br />` : pour aller à la ligne.

Maintenant qu'on sait écrire des paragraphes, voyons voir comment on crée des titres. 😊

## Les titres

Lorsque le contenu de votre page va s'étoffer avec de nombreux paragraphes, cela va devenir difficile pour vos visiteurs de se repérer. C'est là que les titres deviennent utiles.

En HTML on est verni, on a le droit d'utiliser 6 niveaux de titres différents. Je veux dire par là qu'on peut dire "Ceci est un titre très important", "Ceci est un titre un peu moins important", "Ceci est un titre encore moins important", etc. On a donc 6 balises de titre différentes :

- `<h1>` `</h1>` : signifie "titre très important". En général, on s'en sert pour afficher le titre de la page au début de celle-ci.
- `<h2>` `</h2>` : signifie "titre important".
- `<h3>` `</h3>` : pareil, c'est un titre un peu moins important (on peut dire un "sous-titre" si vous voulez).
- `<h4>` `</h4>` : titre encore moins important.
- `<h5>` `</h5>` : titre pas important.
- `<h6>` `</h6>` : titre vraiment, mais alors là vraiment pas important du tout.



Attention : ne confondez pas avec la balise `<title>` ! La balise `<title>` affiche le titre de la page dans la barre de titre du navigateur comme nous l'avons vu. Les titres `<h1>` et compagnie eux, servent à créer des titres qui seront affichés dans la page web.

Ne vous laissez pas impressionner par toutes ces balises. En fait, 6 niveaux de titres, c'est beaucoup. Dans la pratique, personnellement, je n'utilise que les balises `<h1>`, `<h2>` et `<h3>`, et très rarement les autres (je n'ai pas souvent besoin de 6 niveaux de titres différents 😊). Votre navigateur affiche le titre très important en très gros, le titre un peu moins important en un peu moins gros, etc.



Ne choisissez pas votre balise de titre en fonction de la taille qu'elle procure 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 vous voulez modifier la taille du texte, sachez que nous apprendrons à faire ceci en CSS un peu plus tard.

Essayez de faire une page web avec des titres pour voir ce que ça donne :

### Code : HTML

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Niveaux de titres</title>
  </head>

  <body>
    <h1>Titre super important</h1>
    <h2>Titre important</h2>
    <h3>Titre un peu moins important (sous-titre)</h3>

    <h4>Titre pas trop important</h4>
    <h5>Titre pas important</h5>
    <h6>Titre vraiment pas important du tout</h6>
  </body>
</html>
```

Essayer !

Allez, je vous donne un exemple d'utilisation des titres dans une page web (vous allez voir que je ne me sers pas de toutes les balises) :

**Code : HTML**

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Présentation du Site du Zéro</title>
</head>

<body>
  <h1>Bienvenue sur le Site du Zéro !</h1>

  <p>
    Bonjour et bienvenue sur mon site : le Site du Zéro.<br />
    Le Site du Zéro, qu'est-ce que c'est ?
  </p>

  <h2>Des cours pour débutants</h2>

  <p>
    Le Site du Zéro vous propose des cours (tutoriels) destinés
    aux débutants : aucune connaissance n'est requise pour lire ces
    cours !
  </p>
  <p>
    Vous pourrez ainsi apprendre, sans rien y connaître
    auparavant, à créer un site web, à programmer, à construire des
    mondes en 3D !
  </p>

  <h2>Une communauté active</h2>

  <p>
    Vous avez un problème, un élément du cours que vous ne
    comprenez pas ? Vous avez besoin d'aide pour créer votre site ?<br
    />
    Rendez-vous sur les forums ! Vous y découvrirez que vous
    n'êtes pas le seul dans ce cas, et vous trouverez très certainement
    quelqu'un qui vous aidera aimablement à résoudre votre problème.
  </p>
</html>
```

[Essayer !](#)

Voilà une page web qui prend forme ! 😊



Oui mais moi je veux centrer mon titre, l'écrire en rouge et le souligner !

Nous ferons tout cela lorsque nous apprendrons le CSS (dès la deuxième partie du cours). Il faut savoir que **<h1>** ne signifie pas "Times New Roman, taille 16 pt", mais "*Titre important*".

Grâce au langage CSS, vous pourrez dire "Je veux que mes titres importants soient centrés, rouges et soulignés". Pour le moment, en HTML, nous ne faisons que structurer notre page. **Nous rédigeons le contenu avant de nous amuser à le mettre en forme.**



## La mise en valeur

Au sein de vos paragraphes, certains mots sont parfois plus importants que d'autres et vous aimeriez les faire ressortir. HTML vous propose différents moyens de mettre en valeur le texte de votre page.

### Mettre un peu en valeur

Pour mettre *un peu* en valeur votre texte, vous devez utiliser la balise `<em>` `</em>`.

Son utilisation est très simple : entourez les mots à mettre en valeur par ces balises, et c'est bon ! Je reprends un peu l'exemple de tout à l'heure, et j'y mets quelques mots en évidence :

#### Code : HTML

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Emphase</title>
</head>

<body>
  <p>
    Bonjour et bienvenue sur mon site !

    Ceci est mon premier test, alors <em>soyez indulgents</em>
    s'il vous plaît, j'apprends petit à petit comment ça marche.
  </p>
</html>
```

[Essayer !](#)

Comme vous pouvez le voir, utiliser la balise `<em>` a pour conséquence de mettre le texte en *italique*. En fait, c'est le navigateur qui choisit comment afficher les mots. On lui dit que les mots sont assez importants et pour faire ressortir cette information, il change l'apparence du texte en utilisant l'italique.

### Mettre bien en valeur

Pour mettre un texte bien en valeur, on utilise la balise `<strong>` qui signifie "fort", ou "important" si vous préférez. Elle s'utilise exactement de la même manière que `<em>` :

#### Code : HTML

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Forte emphase</title>
</head>

<body>
  <p>
    Bonjour et bienvenue sur mon site !

    Ceci est mon premier test, alors <strong>soyez
    indulgents</strong> s'il vous plaît, j'apprends petit à petit
    comment ça marche.
  </p>
</html>
```

[Essayer !](#)

Vous voyez sûrement le texte s'afficher en gras. Là encore, le gras n'est qu'une *conséquence*. Le navigateur a choisi d'afficher en gras les mots importants pour les faire plus ressortir.

La balise `<strong>` ne signifie pas "mettre en gras" mais "important". On pourra décider plus tard en CSS d'afficher les mots "importants" d'une autre façon que le gras si on le souhaite.

## Marquer le texte

La balise `<mark>` permet de faire ressortir visuellement une portion de texte. Le texte 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 votre site par exemple.

### Code : HTML

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Marquage du texte</title>
</head>

<body>
  <p>
    Bonjour et bienvenue sur mon site !

    Ceci est mon premier test, alors <mark>soyez
    indulgents</mark> s'il vous plaît, j'apprends petit à petit comment
    ça marche.
  </p>
</html>
```

[Essayer !](#)

Par défaut, `<mark>` a pour effet de surligner le texte. On pourra changer l'affichage en CSS (décider de surligner dans une autre couleur, décider d'encadrer le texte, etc.). C'est le même principe que ce que je vous disais pour les balises précédentes : elles indiquent le *sens* des mots et non pas comment ceux-ci doivent s'afficher.

## N'oubliez pas : HTML pour le fond, CSS pour la forme

Je vais peut-être vous sembler un peu lourd mais il est très important qu'on se comprenne bien, car les débutants font souvent la même grosse erreur à ce stade. Ils ont vu les balises `<em>`, `<strong>`, `<mark>`... et ils se disent : "Chouette, j'ai découvert comment mettre en italique, en gras et comment surligner du texte en HTML !".

Et pourtant... ce n'est pas à ça que servent ces balises ! Je sais, je sais, vous allez me dire "Oui mais quand j'utilise `<strong>` le texte apparaît en gras, donc c'est pour mettre en gras.", et pourtant, c'est une erreur de croire que cette balise sert à ça.

Le rôle des balises est d'indiquer le *sens* du texte. Ainsi, `<strong>` indique à l'ordinateur "Ce texte est important". C'est tout. Et pour *montrer* que le texte est important, l'ordinateur décide de le mettre en gras (mais il pourrait aussi bien l'écrire en rouge !). La plupart des navigateurs affichent les textes importants en gras, mais rien ne les y oblige.



Je ne comprends pas. À quoi ça sert que l'ordinateur sache qu'un texte est important ?



Il n'est pas si intelligent pour comprendre !

Détrompez-vous ! De nombreux programmes analysent les codes source des pages web, à commencer par les robots de moteurs de recherche. Ces robots parcourent le web en lisant le code HTML de tous les sites. C'est le cas des robots de Google et de Bing par exemple. Les mots-clés "importants" ont tendance à avoir plus de valeur à leurs yeux, donc si quelqu'un fait une recherche sur ces mots il a plus de chances de tomber sur votre site.

Bien entendu c'est une explication grossière, et il ne faut pas croire qu'utiliser la balise `<strong>` à tout va améliorera votre référencement. Il faut simplement faire confiance aux ordinateurs : ils comprennent ce qu'un texte "important" veut dire et peuvent se servir de cette information.



Mais alors, comment on fait pour mettre spécifiquement en gras, pour écrire en rouge, et tout et tout ?

Tout cela se fait en CSS. Souvenez-vous :

- Le HTML définit le fond (contenu, logique des éléments)
- Le CSS définit la forme (apparence)

Nous verrons le CSS plus loin, pour l'instant nous nous concentrons sur le HTML et ses balises, qui ont chacune un sens particulier. 😊

---

## Les listes à puces

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

- Les listes non ordonnées
- Les listes ordonnées

### Liste non ordonnée

Une liste non ordonnée ressemble à ceci :

- Fraises
- Framboises
- Cerises

C'est un système qui nous permet de faire une liste d'éléments, sans notion d'ordre (il n'y a pas de "premier" ni de "dernier"). Créer une liste à puces non ordonnée est très simple. Il suffit d'utiliser la balise `<ul>` que l'on referme un peu plus loin avec un `</ul>`.

Commencez donc à taper ceci :

Code : HTML

```
<ul></ul>
```

Et maintenant, voilà ce qu'on va faire : on va écrire chacun des éléments de la liste entre 2 balises `<li></li>`. Toutes ces balises doivent se trouver entre `<ul>` et `</ul>`. Vous allez comprendre de suite avec cet exemple :

Code : HTML

```
<ul>
  <li>Fraises</li>
  <li>Framboises</li>
  <li>Cerises</li>
</ul>
```

*Notez que la liste doit être placée à l'intérieur de `<body></body>`. Je ne mets pas tout le code de la page à partir de maintenant pour rester lisible.*

[Essayer !](#)

Retenez donc ces deux balises :

- `<ul></ul>` délimite toute la liste.
- `<li></li>` délimite un élément de la liste (une puce).

Vous pouvez mettre autant d'éléments que vous voulez dans la liste à puces, vous n'êtes pas limités à 3 éléments bien entendu.

Et voilà, vous savez créer une liste à puce non ordonnée ! Pas si dur une fois qu'on a compris comment imbriquer les balises.



Pour ceux qui ont besoin de faire des listes complexes, sachez que vous pouvez *imbriquer* des listes à puces (créer une liste à puces *dans* une liste à puces). Si vous voulez faire ça, ouvrez une seconde balise `<ul>` à l'intérieur d'un élément `<li></li>`.

Si vous fermez les balises dans le bon ordre, vous n'aurez pas de problème. Attention néanmoins, cette technique est un peu compliquée à maîtriser.

## Liste ordonnée

Une liste ordonnée fonctionne de la même façon, seule une balise change : il faut remplacer `<ul></ul>` par `<ol></ol>`. À l'intérieur de la liste, on ne change rien : on utilise toujours des balises `<li></li>` pour délimiter les éléments.



L'ordre dans lequel vous mettez les éléments de la liste est important. Le premier `<li></li>` sera l'élément n°1, le second sera le n°2 etc...

Comme c'est particulièrement intuitif, je vous laisse admirer la simplicité de cet exemple :

### Code : HTML

```
<h1>Ma journée</h1>

<ol>
  <li>Je me lève</li>
  <li>Je mange et je bois</li>
  <li>Je retourne me coucher</li>
</ol>
```

[Essayer !](#)

Par rapport à l'exemple précédent, tout ce qu'on a eu à changer est donc la balise `<ol>`.



Pour information, il existe un troisième type de liste, beaucoup plus rare : la liste de définitions. Elle fait intervenir les balises `<dl>` (pour délimiter la liste), `<dd>` (pour délimiter un terme) et `<dt>` (pour délimiter la définition de ce terme).

Ce chapitre était un peu plus conséquent que les précédents mais il fallait bien rentrer dans le vif du sujet ! Je sais qu'il y a beaucoup de nouvelles balises à retenir, mais cela est nécessaire. En pratiquant un peu, ça viendra tout seul et vous n'aurez plus à faire le moindre effort pour vous souvenir d'une balise. 😊

Si vous trouvez que cela fait beaucoup pour vous d'un coup, n'ayez crainte. Personne ne vous oblige à retenir les balises par cœur. Vous pourrez toujours revenir sur cette page pour vous souvenir comment il faut faire.

Mêmes les webmasters expérimentés ne connaissent pas par cœur toutes les balises du langage HTML. Cependant, vous verrez, à force d'utiliser certaines balises tout le temps, cela rentrera naturellement dans votre tête. 😊



## Créer des liens

Dans le chapitre précédent, vous avez appris à créer une page HTML toute simple. D'accord, elle n'était pas franchement magnifique, mais c'était une vraie page HTML quand même.

Comme vous le savez, un site web est composé de plusieurs pages. Comment faire pour aller d'une page vers une autre ? A l'aide de liens pardi ! Dans ce chapitre, nous allons justement apprendre à créer des liens entre nos pages.

Je suppose que chacun d'entre vous sait ce qu'est un lien : il s'agit d'un texte sur lequel on peut cliquer pour se rendre sur une autre page.

On peut faire un lien d'une page `a.html` vers une page `b.html`, mais on peut aussi faire un lien vers un autre site (ex. : `http://www.siteduzero.com`). Dans les 2 cas, nous allons voir que le fonctionnement est le même.

---

## Un lien vers un autre site

Il est facile de reconnaître les liens sur une page : ils sont écrits d'une façon différente (par défaut en bleu souligné) et un curseur en forme de main apparaît lorsqu'on pointe dessus.

Je vous propose d'essayer de faire le lien suivant qui amène vers le Site du Zéro :

Bonjour. Souhaitez-vous visiter le [Site du Zéro](http://www.siteduzero.com) ?  
C'est un bon site ! ;-)

Pour faire un lien, la balise que nous allons utiliser est très simple à retenir : `<a>`. Il faut cependant lui ajouter un attribut, `href`, pour indiquer vers quelle page on souhaite amener.

Voici un lien qui amène vers le Site du Zéro, situé à l'adresse <http://www.siteduzero.com> :

### Code : HTML

```
<a href="http://www.siteduzero.com">Site du Zéro</a>
```

Nous allons placer ce lien au sein d'un paragraphe. Voici donc comment reproduire l'exemple de l'image précédente :

### Code : HTML

```
<p>Bonjour. Souhaitez-vous visiter le <a  
href="http://www.siteduzero.com">Site du Zéro</a> ?<br />  
C'est un bon site ! ;-)</p>
```

[Essayer !](#)



Par défaut, le lien s'affiche en bleu souligné. Si vous avez déjà visité la page, le lien s'affiche en violet. Nous verrons comment changer cette apparence lorsque nous étudierons le CSS.

Si vous voulez faire un lien vers un autre site, il suffit donc de copier son adresse (on parle d'URL) en `http://`. Notez que certains liens commencent parfois par `https://` (sites sécurisés) ou d'autres préfixes (`ftp://...`).

Si vous faites un lien vers un site qui comporte une adresse un peu bizarre avec des `&`, comme :

`http://www.site.com/?data=15&name=mateo21`

... Vous devrez remplacer tous les `&` par `&amp;` dans votre lien comme ceci :

`http://www.site.com/?data=15&amp;name=mateo21`

Vous ne verrez pas la différence, mais cela est nécessaire pour avoir une page web correctement construite en HTML5.

Les liens que nous venons de voir sont appelés **liens absolus**, car on indique l'adresse complète. Nous allons maintenant voir que l'on peut écrire les liens d'une façon un peu différente, ce qui va nous être utile pour faire des liens entre les pages de notre site.

## Un lien vers une autre page de son site

Nous venons d'apprendre à créer des liens vers des sites déjà existants. Mais je suis sûr que vous aimeriez faire des liens entre les différentes pages de votre site, non ?

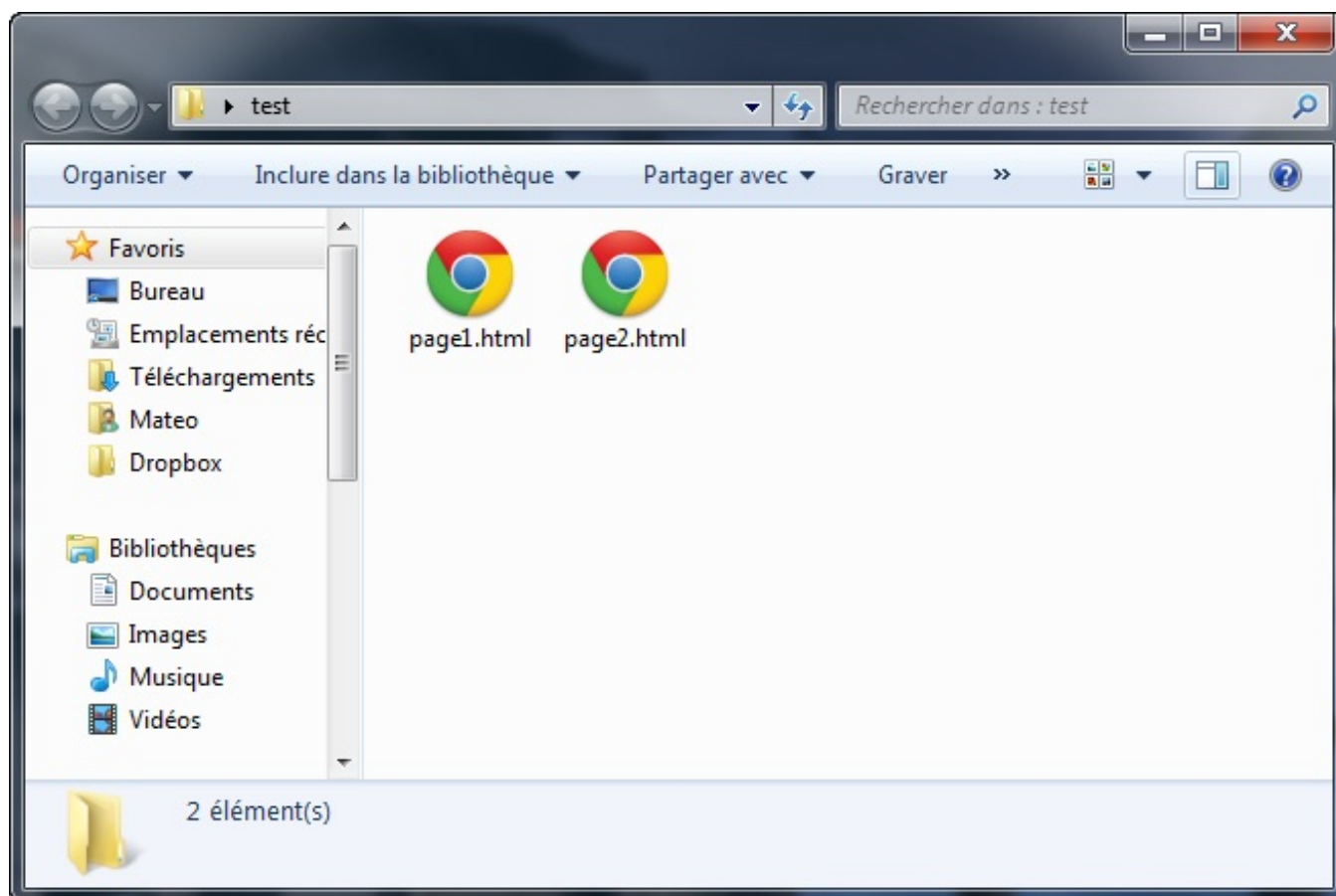


Oui, justement, comment je fais pour faire un lien vers une autre page de mon site ? Je ne connais pas son adresse en `http://...` je commence à peine à créer mon site là ! Je n'ai pas d'adresse. 😞

En effet, pour le moment vous êtes en train de créer votre site sur votre ordinateur. Vous êtes le seul à pouvoir le voir, et il n'a pas encore "d'adresse web" qui commence en `http://` comme la plupart des sites. Heureusement, cela ne va pas nous empêcher de travailler.

### Deux pages situées dans un même dossier

Pour commencer, nous allons créer 2 fichiers correspondant à 2 pages HTML différentes. Comme je suis très inspiré, je vous propose de les appeler `page1.html` et `page2.html`. Nous aurons donc ces 2 fichiers sur notre disque **dans le même dossier** :



Comment faire un lien de la page 1 vers la page 2, sans avoir d'adresse en `http://` ? En fait, c'est facile : si les deux fichiers sont situés dans le même dossier, il suffit d'écrire simplement le nom du fichier vers lequel on veut amener. Par exemple : `<a href="page2.html">`. On dit que c'est un **lien relatif**.

Voici le code que nous allons utiliser dans nos fichiers `page1.html` et `page2.html`.

*page1.html*

Code : HTML



```
<p>Bonjour. Souhaitez-vous visiter <a href="page2.html">la page  
2</a> ?</p>
```

### *page2.html*

La page 2 (page d'arrivée) affichera simplement un message pour indiquer que l'on est bien arrivé sur la page 2 :

**Code : HTML**

```
<h1>Bienvenue sur la page 2 !</h1>
```

### *Essayer !*

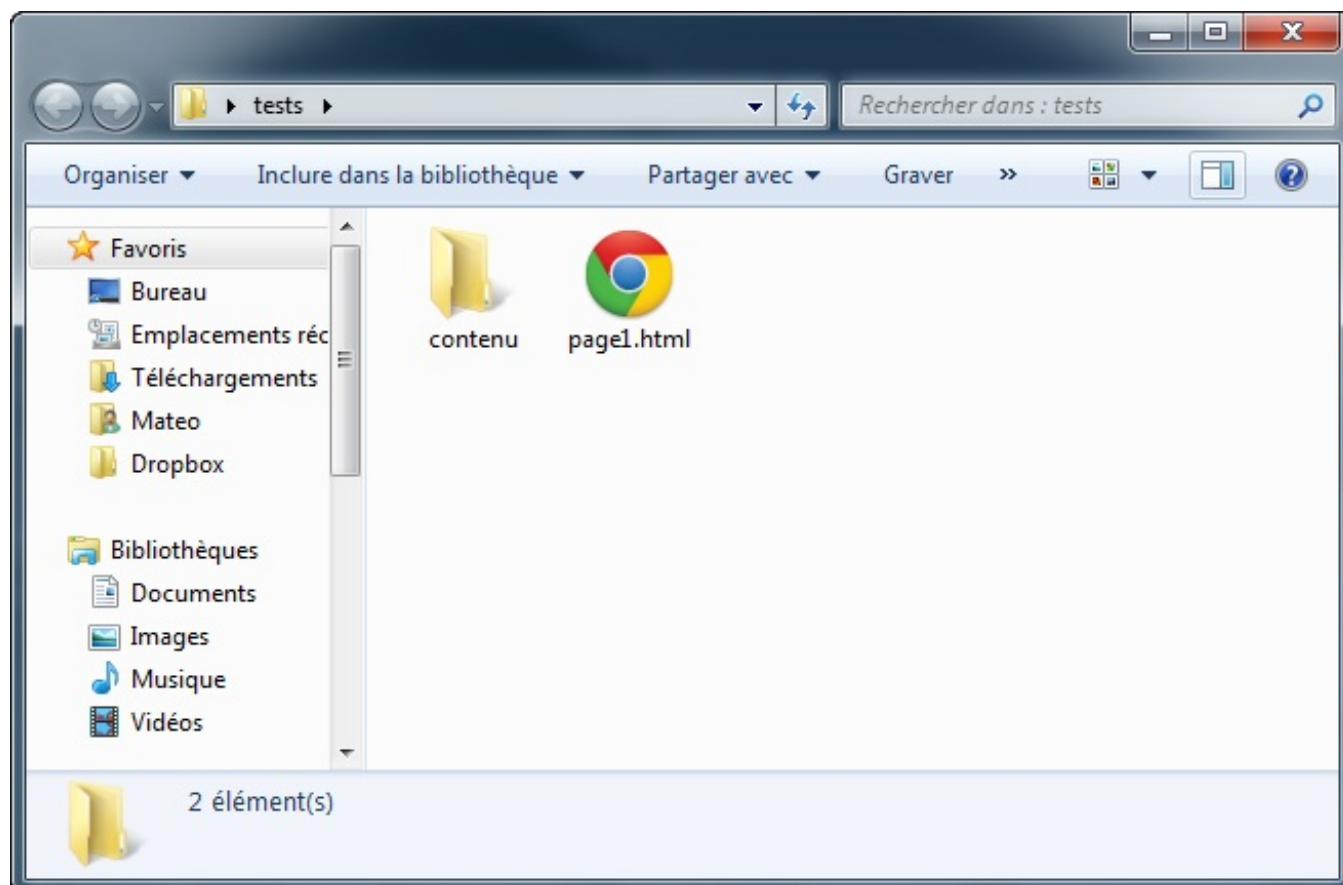
Le lien ci-dessous ouvre page1.html :

[Essayer !](#)

## Deux pages situées dans des dossiers différents

Les choses se corsent un petit peu si les pages sont situées dans des dossiers différents. Idéalement, elles ne devraient être être trop loin l'une de l'autre (dans un sous-dossier par exemple).

Imaginons que page2.html se trouve dans un sous-dossier appelé contenu :



*Le fichier page2.html se trouve à l'intérieur du dossier contenu*

Dans ce cas de figure, il va falloir faire un lien comme ceci :

**Code : HTML**

```
<a href="contenu/page2.html">
```

S'il y avait plusieurs sous-dossiers, on écrirait ceci :

**Code : HTML**

```
<a href="contenu/autredossier/page2.html">
```



Et si le fichier ne se trouve pas dans un sous-dossier mais dans un dossier "parent", on fait comment ?

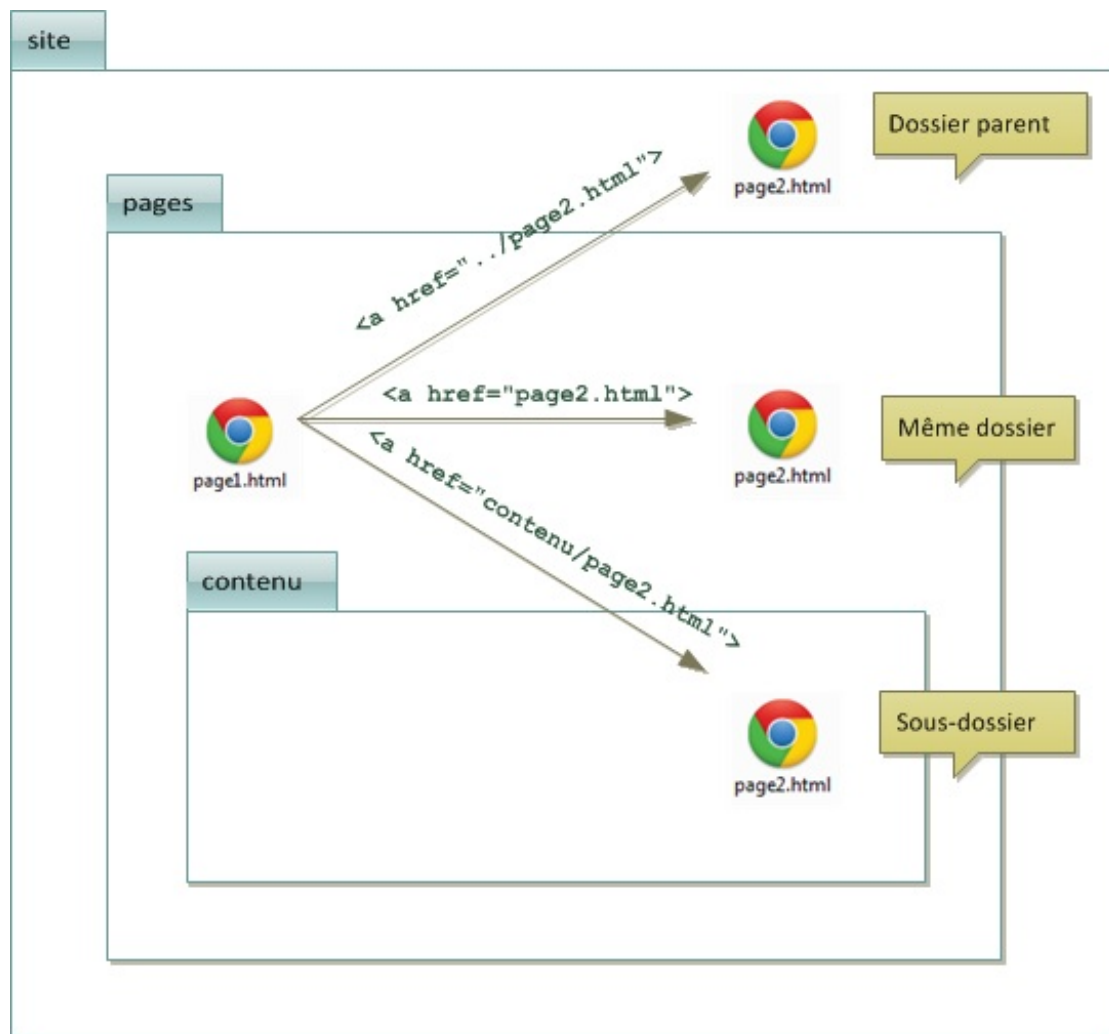
Si votre fichier cible est placé dans un dossier qui se trouve "avant" dans l'arborescence, il faut écrire deux points comme ceci :

**Code : HTML**

```
<a href="../page2.html">
```

## Résumé en images

Les liens relatifs ne sont pas bien compliqués à utiliser une fois qu'on a compris le principe. Il suffit de regarder dans quel "niveau de dossier" se trouve votre fichier cible pour savoir comment écrire votre lien :



## Un lien vers une ancre

Une **ancre** est une sorte de point de repère que vous pouvez mettre dans vos grosses pages HTML.

En effet, si votre page est très grande il peut être utile de faire un lien amenant plus bas dans la même page pour que le visiteur puisse sauter directement à la partie qui l'intéresse.

Pour créer une ancre, il suffit de rajouter l'attribut `id` à une balise qui va alors servir de repère. Ce peut être n'importe quelle balise, un titre par exemple.

Utilisez l'attribut `id` pour donner un nom à l'ancre. Cela nous servira ensuite pour faire un lien vers cette ancre. Par exemple :

### Code : HTML

```
<h2 id="mon_ancre">Titre</h2>
```

Ensuite, il suffit de faire un lien comme d'habitude, mais cette fois l'attribut `href` contiendra un dièse (#) suivi du nom de l'ancre. Exemple :

### Code : HTML

```
<a href="#mon_ancre">Aller vers l'ancre</a>
```

Normalement, si vous cliquez sur le lien, cela vous amènera plus bas dans la même page (à condition que la page comporte suffisamment de texte pour que les barres de défilement se déplacent automatiquement).

Voici un exemple de page comportant beaucoup de texte et utilisant les ancres (j'ai mis n'importe quoi dans le texte pour remplir 🤖) :

### Code : HTML

```
<h1>Ma grande page</h1>

<p>
  Aller directement à la partie traitant de :<br />
  <a href="#cuisine">La cuisine</a><br />
  <a href="#rollers">Les rollers</a><br />
  <a href="#arc">Le tir à l'arc</a><br />
</p>
<h2 id="cuisine">La cuisine</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="rollers">Les rollers</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="arc">Le tir à l'arc</h2>

<p>... (beaucoup de texte) ...</p>
```

[Essayer !](#)

S'il ne se passe rien quand vous cliquez sur les liens, c'est qu'il n'y a pas assez de texte. Dans ce cas, vous pouvez soit rajouter du blabla dans la page pour qu'il y ait (encore) plus de texte, ou bien réduire la taille de la fenêtre de votre navigateur pour faire apparaître les barres de défilement sur le côté.



L'attribut `id` sert à donner un nom "unique" à une balise, pour s'en servir de repère. Et, croyez-moi, vous n'avez pas fini d'entendre parler de cet attribut. Ici, on s'en sert pour faire un lien vers une ancre, mais en CSS cela nous sera très utile pour "repérer" une balise précise, vous verrez. 😊

Evitez cependant de créer des `id` avec des espaces ou des caractères spéciaux, utilisez simplement des lettres et chiffres dans la mesure du possible pour que cela soit reconnu par tous les navigateurs.

## Lien vers une ancre située dans une autre page

Alors là je vous préviens, on va faire le Mégamix ! 😊

L'idée, c'est de faire un lien qui ouvre une nouvelle page ET qui amène directement à une ancre située plus bas sur cette page. En pratique c'est assez simple à faire : il suffit de taper le nom de la page, suivi d'un dièse (`#`), suivi du nom de l'ancre.

Par exemple : `<a href="ancres.html#rollers">`

... vous amènera sur la page `ancres.html`, directement au niveau de l'ancre appelé "rollers".

Voici une page qui contient 3 liens, chacun amenant vers une des ancres de la page de l'exemple précédent :

### Code : HTML

```
<h1>Le Mégamix</h1>
<p>
  Rendez-vous quelque part sur une autre page :<br />
  <a href="ancres.html#cuisine">La cuisine</a><br />
  <a href="ancres.html#rollers">Les rollers</a><br />
  <a href="ancres.html#arc">Le tir à l'arc</a><br />
</p>
```

[Essayer !](#)

## Cas pratiques d'utilisation des liens

Je vais essayer de vous montrer ici quelques cas pratiques d'utilisation des liens. Par exemple, saviez-vous qu'il est très facile de faire des liens qui lancent un téléchargement ? Qui créent un nouvel e-mail ? Qui ouvrent une nouvelle fenêtre ?

Non ? Eh bien nous allons voir tout ça ici. 😊

### Un lien qui affiche une infobulle au survol

Vous pouvez utiliser l'attribut `title` qui affiche une bulle d'aide lorsqu'on pointe sur le lien. Cet attribut est facultatif.

Vous aurez un résultat ressemblant à ceci :



La bulle d'aide peut être utile pour informer le visiteur avant même qu'il n'ait cliqué sur le lien.

Voici comment reproduire ce résultat :

#### Code : HTML

```
<p>Bonjour. Souhaitez-vous visiter le <a  
href="http://www.siteduzero.com" title="Réservé aux débutants">Site  
du Zéro</a> ?</p>
```

[Essayer !](#)

### Un lien qui ouvre une nouvelle fenêtre

Il est possible de "forcer" l'ouverture d'un lien dans une nouvelle fenêtre. Pour cela, on rajoutera `target="_blank"` à la balise `<a>` :

#### Code : HTML

```
<p>Bonjour. Souhaitez-vous visiter le <a  
href="http://www.siteduzero.com" target="_blank">Site du Zéro</a>  
<br />  
Le site s'affichera dans une autre fenêtre.</p>
```

[Essayer !](#)



Selon la configuration du navigateur, la page s'affichera dans une nouvelle fenêtre ou un nouvel onglet. Vous ne pouvez pas choisir entre l'ouverture d'une nouvelle fenêtre ou d'un nouvel onglet.



Notez 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.

## Un lien pour envoyer un e-mail

Si vous voulez que vos visiteurs puissent vous envoyer un mail, vous pouvez utiliser des liens de type "mailto". Rien ne change au niveau de la balise, vous devez simplement modifier la valeur de l'attribut href comme ceci :

Code : HTML

```
<p><a href="mailto:votrenom@bidule.com">Envoyez-moi un e-mail  
</a></p>
```

[Essayer !](#)

Il suffit donc de faire commencer le lien par "mailto:" et d'écrire l'adresse e-mail où on peut vous contacter. Si vous cliquez sur le lien, un nouveau message vide s'ouvre, prêt à être envoyé à votre adresse e-mail. 😊

## Un lien qui de téléchargement d'un fichier

Beaucoup d'entre vous se demandent comment ça se passe pour le téléchargement d'un fichier... En fait, il faut faire exactement comme si vous faisiez un lien vers une page web, mais en indiquant cette fois le nom du fichier à télécharger.

Par exemple, supposez que vous vouliez faire télécharger monfichier.zip. Placez simplement ce fichier dans le même dossier que votre page web (ou dans un sous-dossier) et faites un lien vers ce fichier :

Code : HTML

```
<p><a href="monfichier.zip">Télécharger le fichier</a></p>
```

C'est tout ! Le navigateur, voyant qu'il ne s'agit pas d'une page web à afficher, va lancer la procédure de téléchargement lorsqu'on cliquera sur le lien.

En résumé donc, on distingue 2 types de liens :

- Les liens vers d'autres pages, de loin les plus courants.
- Les liens vers des ancres, pour amener plus bas sur une même page.

Il y a aussi, comme vous l'avez vu, la possibilité de faire des liens qui amènent vers une ancre située sur une autre page. Bref, vous avez l'embaras du choix. 😊

## Les images

Insérer une image dans une page web ? Vous allez voir, c'est d'une facilité déconcertante. 😊

Enfin presque. Il existe différents *formats* d'image que l'on peut utiliser sur des sites web, et on ne doit pas les choisir au hasard. En effet, les images sont parfois volumineuses à télécharger, ce qui ralentit le temps de chargement de la page (beaucoup plus que le texte !).

Pour faire en sorte que vos pages restent lisibles et rapides à télécharger, suivez donc activement mes conseils !



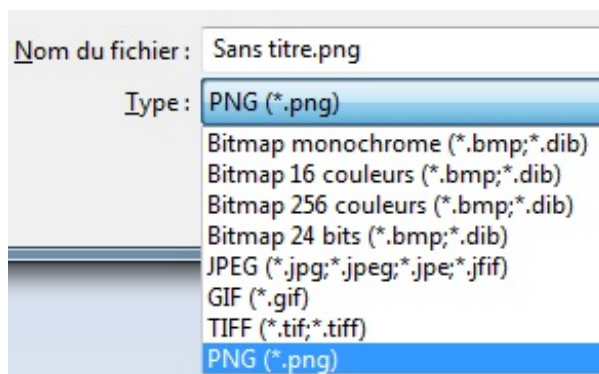


## Les différents formats d'images

Savez-vous ce qu'est un format d'image ?

Quand vous avez une image entre les mains, vous avez la possibilité de l'enregistrer dans plusieurs "formats" différents. Le poids (en Ko, voire en Mo) de l'image sera plus ou moins élevé selon le format choisi, et la qualité de l'image va changer.

Par exemple, le logiciel de dessin Paint (même si c'est loin d'être le meilleur) vous propose de choisir entre plusieurs formats lorsque vous enregistrez une image :



Certains formats sont plus adaptés que d'autres selon l'image (photo, dessin, image animée...). Notre but ici est de faire le tour des différents formats que l'on utilise sur le web, pour que vous les connaissiez et sachiez choisir celui qui convient le mieux à votre image. Rassurez-vous, il n'y a pas beaucoup de formats différents, ça ne sera donc pas bien long. 😊

Toutes les images diffusées sur Internet ont un point commun : elles sont **compressées**. Cela veut dire que l'ordinateur fait des calculs pour qu'elles soient moins lourdes et donc plus rapides à charger.

### Le JPEG

Les images au format JPEG (*Joint Photographic Expert Group*) sont très répandues sur le Web. Ce format est conçu pour réduire la taille des photos, qui peuvent comporter plus de 16 millions de couleurs différentes.

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

Voici un exemple d'image au format JPEG :

*montagne.jpg*

Notez 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.

Quand il s'agit d'une photo, on ne peut généralement pas détecter la perte de qualité. Par contre, si ce n'est pas une photo, vous risquez de voir l'image un peu "baver". Dans ce cas, il vaut mieux utiliser le format PNG.

## Le PNG

Le format PNG (*Portable Network Graphics*) est le plus récent de tous. Ce format est adapté à la plupart des graphiques (je serais tenté de dire "à tout ce qui n'est pas une photo"). Le PNG a deux gros avantages : il peut être rendu transparent et 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 c'est devenu le format le plus puissant pour enregistrer la plupart des images.

Le PNG existe en 2 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)

Voici une image PNG en 24 bits, la célèbre mascotte Zozor du Site du Zéro :



*zozor.png*



Une vieille version d'Internet Explorer (IE6) ne peut pas afficher correctement les images PNG 24 bits transparentes. Ce navigateur tend à être de moins en moins utilisé, mais gardez quand même cette information en tête.



Au fait, si le PNG 24 bits peut afficher autant de couleurs qu'une image JPEG, et qu'en plus il peut être rendu transparent sans modifier la qualité de l'image... quel est l'intérêt du JPEG ?

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. Je vous conseille donc toujours de réserver le format JPEG aux photos.

## Le 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 : les images sont le plus souvent plus légères et la transparence est de meilleure qualité. Je vous recommande donc d'utiliser le PNG autant que possible.

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. Exemple :



*manga.gif*

## Il existe un format adapté à chaque image

Si on résume, voici quel format adopter en fonction de l'image que vous avez :

- **Une photo** : utilisez un JPEG.
- **N'importe quel graphique avec peu de couleurs** (moins de 256) : utilisez un PNG 8 bits, ou éventuellement un GIF.
- **N'importe quel graphique avec beaucoup de couleurs** : utilisez un PNG 24 bits.
- **Une image animée** : utilisez un GIF animé.

## Les erreurs à éviter

### *Bannissez les autres formats*

Les autres formats non cités ici, comme le format BITMAP (\*.bmp) sont à bannir car bien souvent ils ne sont pas compressés, donc trop gros. Ils ne sont pas du tout adaptés au web. On peut en mettre sur son site, mais le chargement sera vraiment *extrêmement* long !

### *Choisissez bien le nom de votre image*

Si vous voulez éviter des problèmes, prenez l'habitude d'enregistrer vos fichiers avec des noms en minuscules, sans espaces ni

accents. Par exemple : `mon_image.png`.

Vous pouvez remplacer les espaces par le caractère "underscore" comme je l'ai fait : `_`.

---

## Insérer une image

Revenons maintenant au code HTML pour découvrir comment placer des images dans nos pages web ! 😊

### Insertion d'une image

Quelle est la fameuse balise qui va nous permettre d'insérer une image ? Il s'agit de... `<img />` !

C'est une balise de type *orpheline* (comme `<br />`). Cela veut dire qu'on n'a pas besoin de l'écrire en deux exemplaires comme la plupart des autres balises que nous avons vues jusqu'ici. En effet, nous n'avons pas besoin de délimiter une portion de texte, nous voulons juste insérer une image à un endroit précis.

La balise doit être accompagnée de 2 attributs obligatoires :

- `src` : il permet d'indiquer où se trouve l'image que l'on veut insérer. Vous pouvez soit mettre un chemin en absolu (ex. : `http://www.site.com/fleur.png`), soit mettre le chemin en relatif (ce qu'on fait le plus souvent). Ainsi, si votre image est dans un sous-dossier `images` vous devrez taper : `src="images/fleur.png"`
- `alt` : cela signifie "texte alternatif". On doit *toujours* indiquer un texte alternatif à l'image, c'est-à-dire un court texte qui décrit ce que contient l'image. Ce texte sera affiché à la place de l'image si celle-ci ne peut pas être téléchargée (ça arrive), ou sur les navigateurs de personnes handicapées (non-voyants) qui ne peuvent malheureusement pas "voir" l'image. Cela aide aussi les robots des moteurs de recherche pour les recherches d'images. Pour la fleur, on mettrait par exemple : `alt="Une fleur"`.

Les images doivent se trouver obligatoirement à l'intérieur d'un paragraphe (`<p></p>`). Voici un exemple d'insertion d'image :

#### Code : HTML

```
<p>
    Voici une photo que j'ai prise lors de mes dernières vacances à
    la montagne : <br />
    
</p>
```

[Essayer !](#)

Bref, l'insertion d'image est quelque chose de très facile pour peu qu'on sache indiquer où se trouve l'image, comme on avait appris à le faire avec les liens. 😊

La plus grosse "difficulté" (si on peut appeler ça une difficulté) consiste à choisir le bon format d'image. Ici, c'est une photo donc c'est évidemment le format JPEG qu'on utilise.

Je le répète : évitez à tout prix les accents, majuscules et espaces dans vos noms de fichiers et de dossiers. Voici un chemin qui va poser problème :

"Images du site/Image toute bête.jpg"

Il faudrait supprimer les espaces (ou les remplacer par le symbole "\_"), supprimer les accents et tout mettre en minuscules comme ceci :

"images\_du\_site/image\_toute\_bete.jpg"

Sachez donc que si votre image ne s'affiche pas, c'est très certainement parce que le chemin est incorrect ! Simplifiez au maximum vos noms de fichiers et de dossiers et tout ira bien. 😊



### Ajouter une infobulle

L'attribut fait pour afficher une bulle d'aide est le même que pour les liens : il s'agit de `title`. Cet attribut est facultatif (contrairement à `alt`).

Voici ce que ça peut donner :

#### Code : HTML

```
<p>
  Voici une photo que j'ai prise lors de mes dernières vacances à
  la montagne :<br />
  
</p>
```

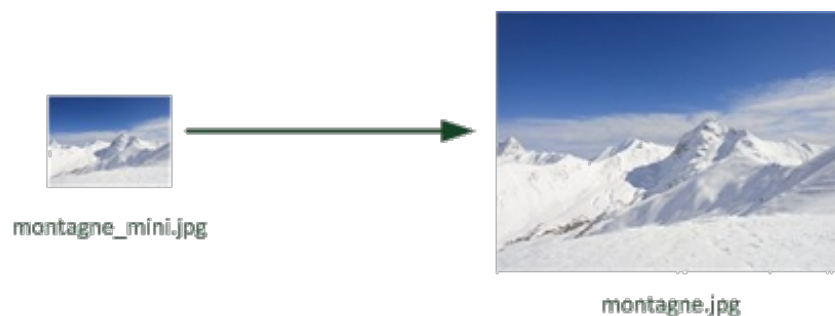
Survolez la photo avec la souris pour voir l'infobulle apparaître.

[Essayer !](#)

## Miniature cliquable

Si votre image est très grosse, il est conseillé d'en afficher la miniature sur votre site. Ajoutez ensuite un lien sur cette miniature pour que vos visiteurs puissent afficher l'image en taille originale.

Il existe des millions de logiciels permettant de créer des miniatures d'images. J'utilise personnellement [Easy Thumbnails](#). Je vais ainsi disposer de 2 versions de ma photo : la miniature et l'image d'origine.



Je les place toutes les deux dans un dossier appelé `img` par exemple. J'affiche la version `montagne_mini.jpg` sur ma page et je fais un lien vers `montagne.jpg` pour que l'image agrandie s'affiche lorsqu'on clique sur la miniature.

Voici le code HTML que je vais utiliser pour cela :

#### Code : HTML

```
<p>
  Vous souhaitez voir l'image dans sa taille d'origine ? Cliquez
  dessus !<br />
  <a href="img/montagne.jpg"></a>
</p>
```

[Essayer !](#)



Parfois, certains navigateurs choisissent d'afficher un cadre bleu (ou violet) pas très esthétique autour de votre image cliquable.  
Heureusement, nous pourrons retirer ce cadre grâce au CSS dans peu de temps. 😊

---

## Les figures

Au cours de la lecture de ce tutoriel, vous avez déjà rencontré plusieurs fois des **figures**. Ce sont des éléments qui viennent enrichir le texte pour compléter les informations de la page.

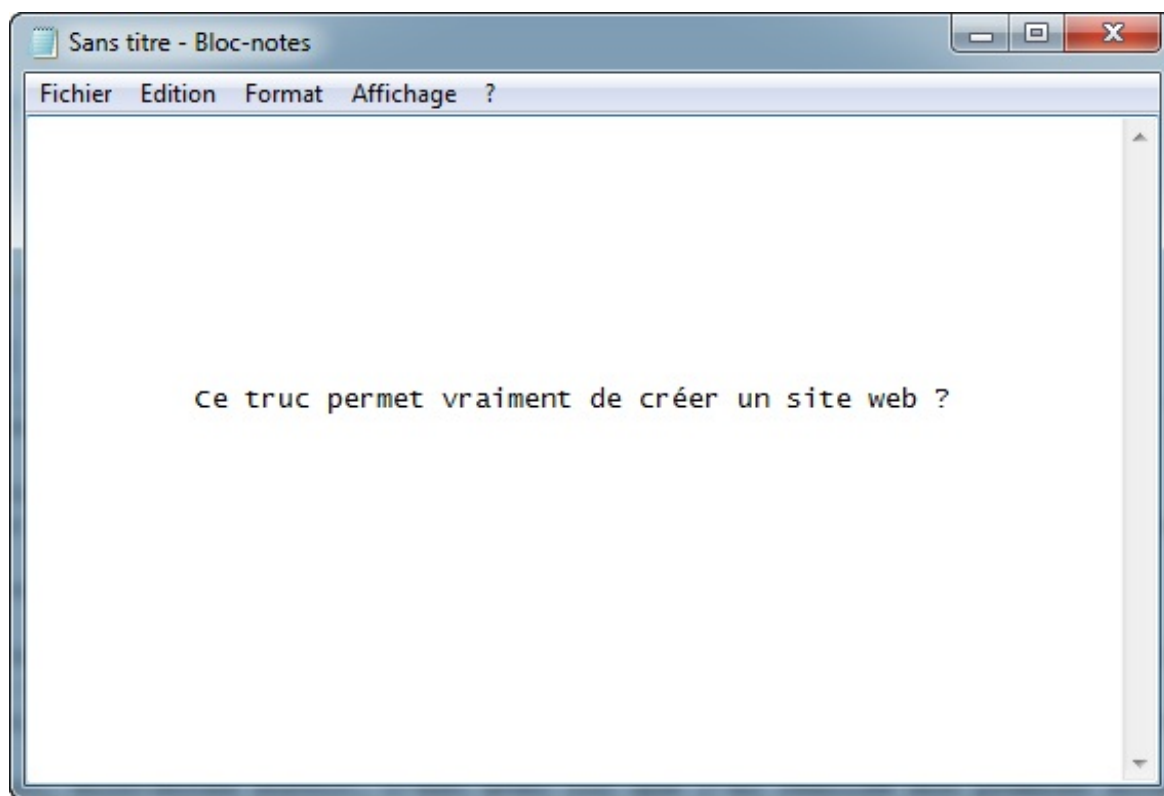
Les figures peuvent être de différents types :

- Images
- Codes source
- Citations
- ...

Bref, tout ce qui vient *illustrer* le texte est une figure. Nous allons ici nous intéresser aux images, mais contrairement à ce qu'on pourrait croire, les figures ne sont donc pas *forcément* des images : un code source aussi illustre le texte.

### Création d'une figure

Reprenons par exemple cette capture d'écran du premier chapitre :



*Le logiciel Bloc-Notes*

En HTML5, on dispose de la balise `<figure>`. Voici comment on pourrait l'utiliser :

**Code : HTML**

```
<figure>
  
</figure>
```



Une figure est le plus souvent accompagnée d'une légende. Pour ajouter une légende, utilisez la balise `<figcaption>` à l'intérieur de la balise `<figure>`, comme ceci :

**Code : HTML**

```
<figure>
  
  <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```

Essayer !

## Bien comprendre le rôle des figures

Un peu plus tôt dans ce chapitre, je vous ai dit que les images devaient être situées dans des paragraphes (placées à l'intérieur d'une balise `<p></p>`). Ce n'est pas tout à fait vrai. 🤔

Si vous faites de votre image une figure, l'image peut être située en-dehors d'un paragraphe.

**Code : HTML**

```
<p>Connaissez-vous le logiciel Bloc-Notes ? On peut faire des sites
web avec !</p>

<figure>
  
  <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```



Je ne vois pas vraiment de changement. Quand dois-je placer mon image dans un paragraphe et quand dois-je la placer dans une figure ?

Bonne question ! Tout dépend de ce que votre image apporte au texte :

- Si elle n'apporte aucune information (c'est juste une illustration pour décorer) : placez l'image dans un paragraphe.
- Si elle apporte une information : placez l'image dans une figure.

La balise `<figure>` a un rôle avant tout sémantique. Cela veut dire qu'elle indique à l'ordinateur que l'image a du sens et qu'elle est importante pour la bonne compréhension du texte. Cela peut permettre à un programme de récupérer toutes les figures du texte et de les référencer dans une table des figures par exemple.

Enfin, sachez qu'une figure peut très bien comporter plusieurs images. Voici un cas où cela se justifie :

**Code : HTML**

```
<figure>
  
  
  
```

```
<figcaption>Logos des différents navigateurs</figcaption>
</figure>
```

Vous savez maintenant insérer des images dans vos pages web !

Saviez-vous qu'avec HTML5 on peut aussi insérer très facilement des vidéos et des extraits audio dans ses pages ? Auparavant, il était obligatoire d'utiliser des plugins assez lourds comme Flash, mais désormais, c'est simple comme une balise. Vous allez voir tout ça dans le chapitre suivant ! 😊

## Partie 1 : Les joies de la mise en forme avec CSS

Maintenant que vous connaissez les bases du HTML... donnez du *style* à votre page grâce au CSS !

---

### Mettre en place le CSS

Nous y voici enfin. 😊

Après avoir passé toute une première partie du cours à ne travailler que sur le HTML, nous allons maintenant découvrir le CSS que j'avais volontairement mis à l'écart.

Le CSS n'est pas plus compliqué que le HTML. Il vient le compléter pour vous permettre de mettre en forme votre page web.

Le CSS comporte beaucoup de propriétés, qui sont un peu l'équivalent des balises en HTML. Heureusement, personne ne vous demande de tout retenir par coeur et un aide-mémoire sera à votre disposition en annexe pour vous aider (aide-mémoire dont j'ai moi-même besoin, je ne retiens pas tout 😊).

Dans ce premier chapitre sur le CSS, nous allons voir la théorie sur le CSS : qu'est-ce que c'est ? A quoi ça ressemble ? Où est-ce qu'on écrit du code CSS ? Cette théorie n'est pas bien compliquée, mais vous devez obligatoirement la connaître car c'est la base du CSS. C'est d'ailleurs la seule chose que je vous demanderai de retenir par coeur en CSS, le reste pourra être retrouvé dans le mémo en annexe. 😊

Allez, ne traînons pas, je vois que vous brûlez d'impatience !

---

## Où mettre du CSS ?

Vous vous souvenez ce que CSS veut dire ? Je vous en ai rapidement parlé dans le premier chapitre du cours. Cela signifie : "*Cascading Style Sheets*", ce qui peut se traduire en français par "Feuilles de style en cascade".

On dit "Feuille de style" car en règle générale, on écrit le code CSS dans un fichier à part (à l'extension .css au lieu de .html). C'est un fichier dans lequel on écrit l'apparence que notre site doit avoir : la couleur et la police du texte, la taille des titres, la position des menus, la couleur ou l'image de fond etc... Croyez-moi, le CSS permet de faire une foule de choses ! 😊

On peut écrire du code CSS à 3 endroits différents, selon ce qu'on préfère :

- **Méthode A** : dans un fichier .css (le moyen le plus recommandé)
- **Méthode B** : dans l'en-tête du fichier XHTML
- **Méthode C** : dans les balises

Voici une description de chacune de ces techniques. Si vous ne devez en retenir qu'une, retenez la première :

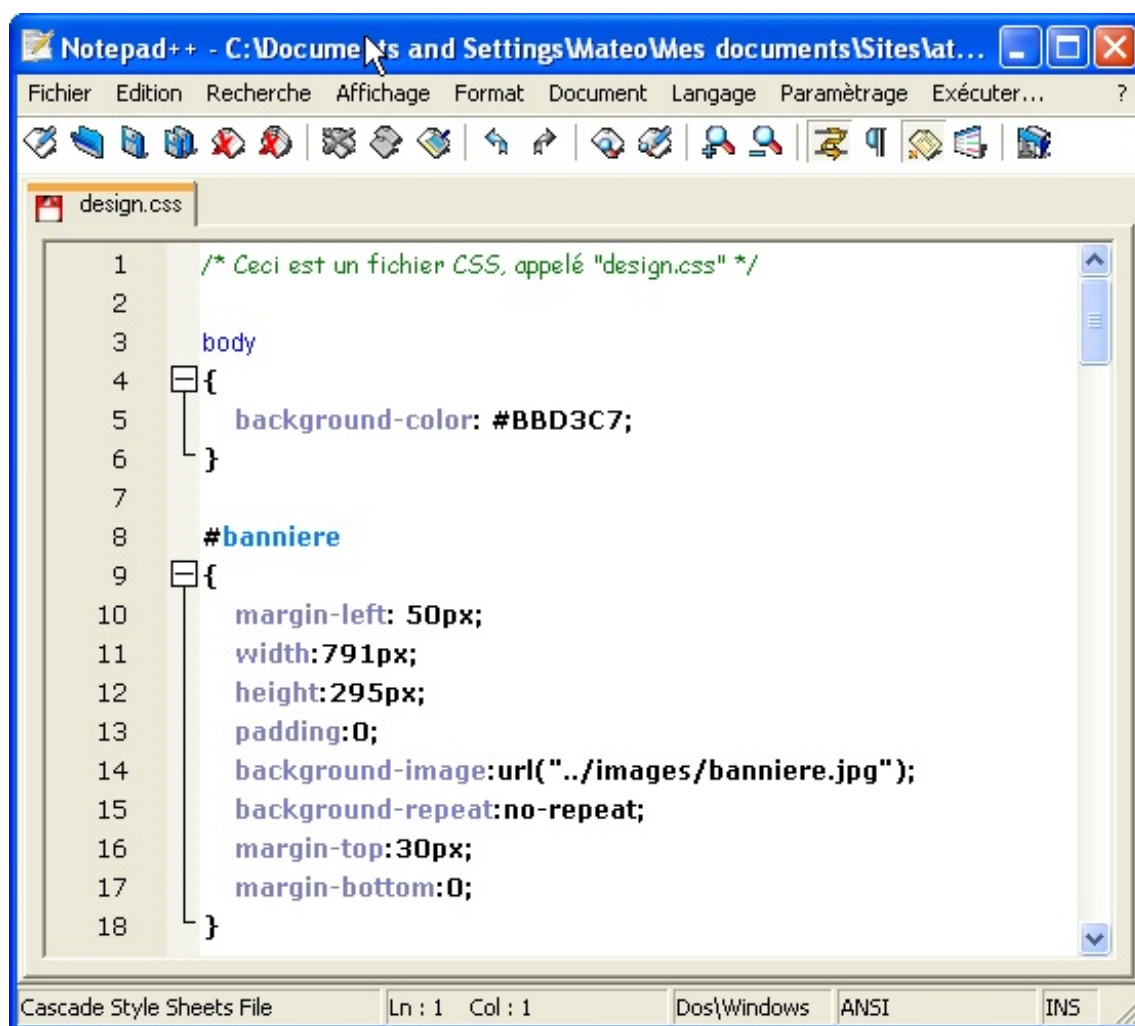
- Dans un fichier .css (Méthode A) :

Comme je viens de vous le dire, le plus souvent on écrit du code CSS dans un fichier spécial ayant l'extension .css (contrairement aux fichiers XHTML qui ont l'extension .html).

C'est la méthode la plus pratique que je vais utiliser quasiment tout le temps dans la suite du cours. Parmi les nombreux avantages que cette solution apporte, il y a la possibilité de pouvoir proposer facilement plusieurs designs différents à vos visiteurs.

Si vous utilisez Notepad++, pensez à aller dans le menu "Langage / CSS" pour activer la coloration du CSS. Vous penserez à enregistrer votre fichier en .css au lieu de .html.

Voici un exemple de fichier CSS sous Notepad++ :



Si vous voulez voir comment je procède dans une vidéo, cliquez sur le lien ci-dessous :

[Enregistrer un CSS sous Notepad++ \(170 Ko\)](#)

Ce que vous voyez est un aperçu du code CSS que nous allons écrire. Ne vous préoccupez pas de savoir ce que ça veut dire pour le moment, je vais vous expliquer ça un peu plus loin 😊

Maintenant, retournez dans votre fichier XHTML. Il faut y ajouter une ligne entre les balises <head> </head> qui va permettre de "dire" au fichier XHTML quel fichier CSS il doit charger :

#### Code : HTML

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
  <head>
    <title>Exemple d'utilisation de CSS externe</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    <link rel="stylesheet" media="screen" type="text/css"
title="Design" href="design.css" />
  </head>
  <body>
    <p>Cette page comporte une feuille de style externe.
C'est la meilleure méthode à utiliser quand on fait du CSS.</p>
  </body>
</html>

```

La balise `<link />` comporte plusieurs attributs. Vous pouvez en modifier deux d'entre eux pour le moment :

- *title* : c'est le nom que vous donnez à votre feuille de style (mettez ce que vous voulez)
- *href* : c'est l'emplacement où se trouve votre feuille de style sous forme de lien relatif. Dans cet exemple le CSS se trouve dans le même dossier.

- **Directement dans le header du fichier XHTML ( Méthode B ) :**

Il est aussi possible de taper du CSS directement à l'intérieur même du fichier XHTML, entre les balises `<head>` `</head>`. Vous devrez y mettre une balise `<style>` `</style>` à l'intérieur, comme ceci :

**Code : HTML**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
  <head>
    <title>Exemple de CSS dans le header</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    <style type="text/css">
      /* Vous taperez votre code CSS ici */
    </style>
  </head>
  <body>
    <p>Cette page comporte du CSS directement dans le
header.</p>
  </body>
</html>
```



Cette seconde méthode ressemble beaucoup à la première. Cependant, la première solution (utiliser un .css externe) est quand même bien plus pratique, car elle vous permettra de faire changer le design du site en un clin d'oeil. De plus, le fichier ne sera téléchargé qu'une seule fois pour toutes les pages de votre site, ce qui allègera la taille des fichiers .html et rendra donc votre site plus rapide !

Au risque de me répéter (faut bien que ça rentre :p), je vous recommande donc d'utiliser un fichier .css externe (la première solution proposée) plutôt que de mettre directement le CSS dans le fichier XHTML 😊

- **La méthode "à l'arrache" dans les balises (Méthode C) :**

ça c'est la troisième (et la moins recommandée) des méthodes. Elle consiste à ajouter un attribut *style* sur les balises pour leur appliquer un style particulier. Cet attribut fonctionne sur toutes les balises. Par exemple sur un titre :

**Code : HTML**

```
<h1 style="/* Votre style pour cette balise ici */">Titre</h1>
```

Cette méthode a tous les défauts : non seulement le CSS sera peu lisible à l'intérieur des balises, mais en plus ça ne nous permet pas de profiter de tous les avantages du CSS, comme la possibilité de changer la couleur de tous les titres du site en un clic.

Bref, je vous ai montré la méthode seulement pour que vous ne soyez pas perdus si vous la voyez.

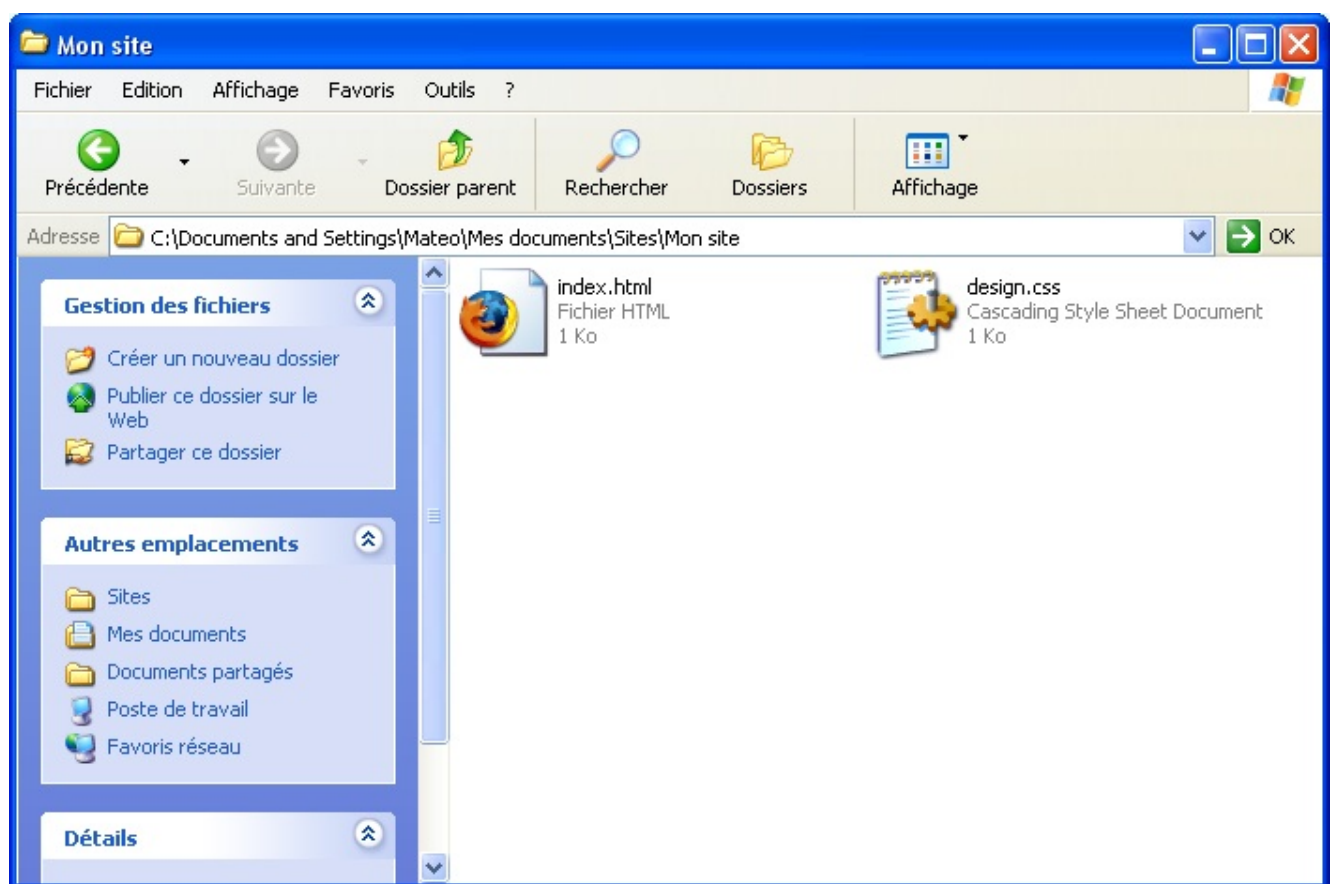
Voilà, vous venez de voir quelles sont les 3 méthodes que l'on peut utiliser pour insérer du CSS (de la meilleure à la pire). L'idéal est donc d'utiliser une feuille de style externe (= mettre son CSS dans un fichier .css) et c'est cette méthode que je vais utiliser dans la suite du cours.

## Résumé visuel

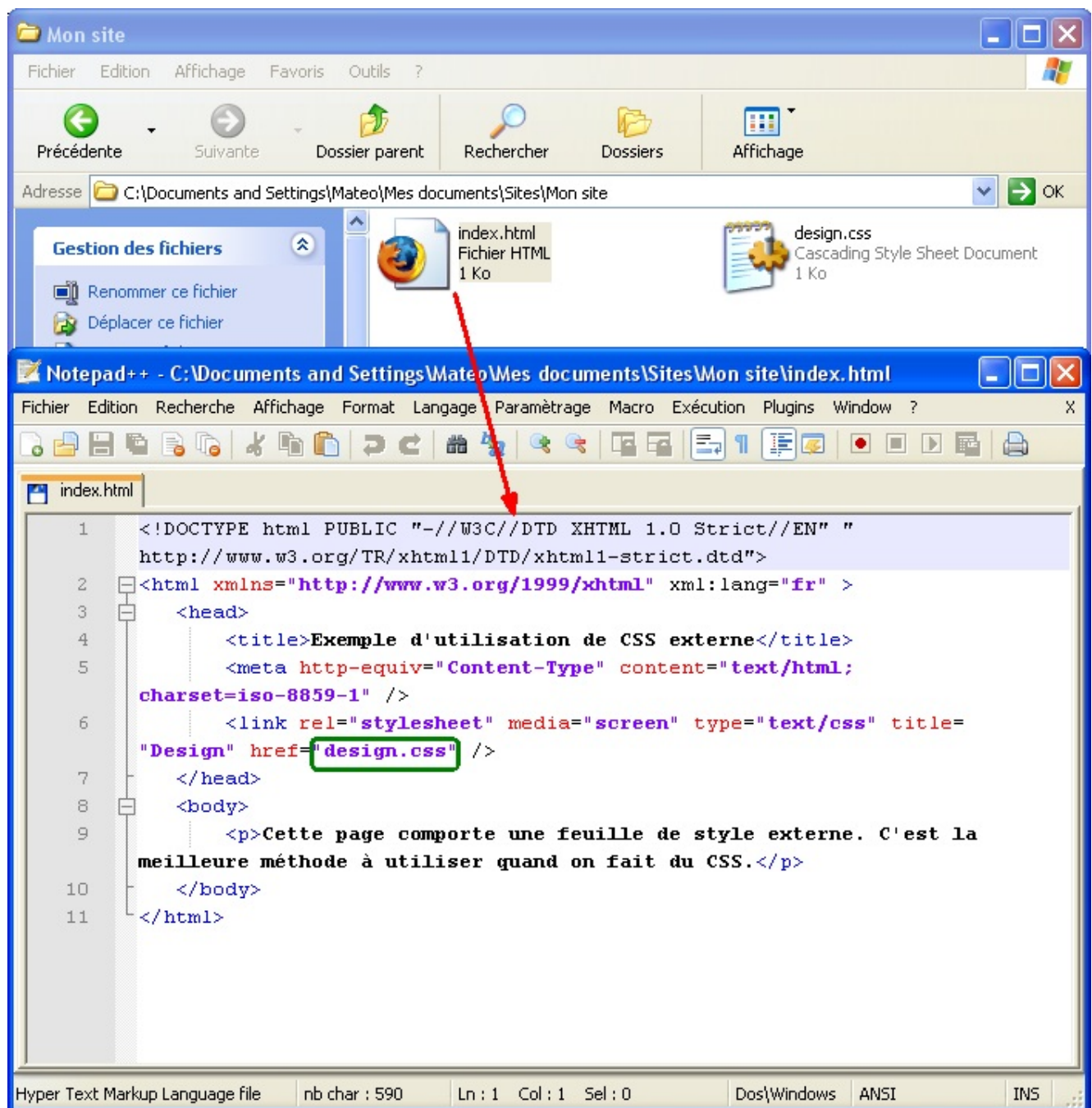
Je suppose ici que vous avez choisi la **Méthode A** que je vous ai conseillée. Jusqu'ici on travaillait sur un seul fichier, le .html. A partir de maintenant on va travailler sur 2 fichiers : un .css et toujours notre bon vieux .html.

Résumons donc les fichiers que nous devons avoir sous les yeux pour que ça soit clair pour tout le monde :

- Dans le dossier de votre site, il doit y avoir au moins 2 fichiers : un .html et un .css.

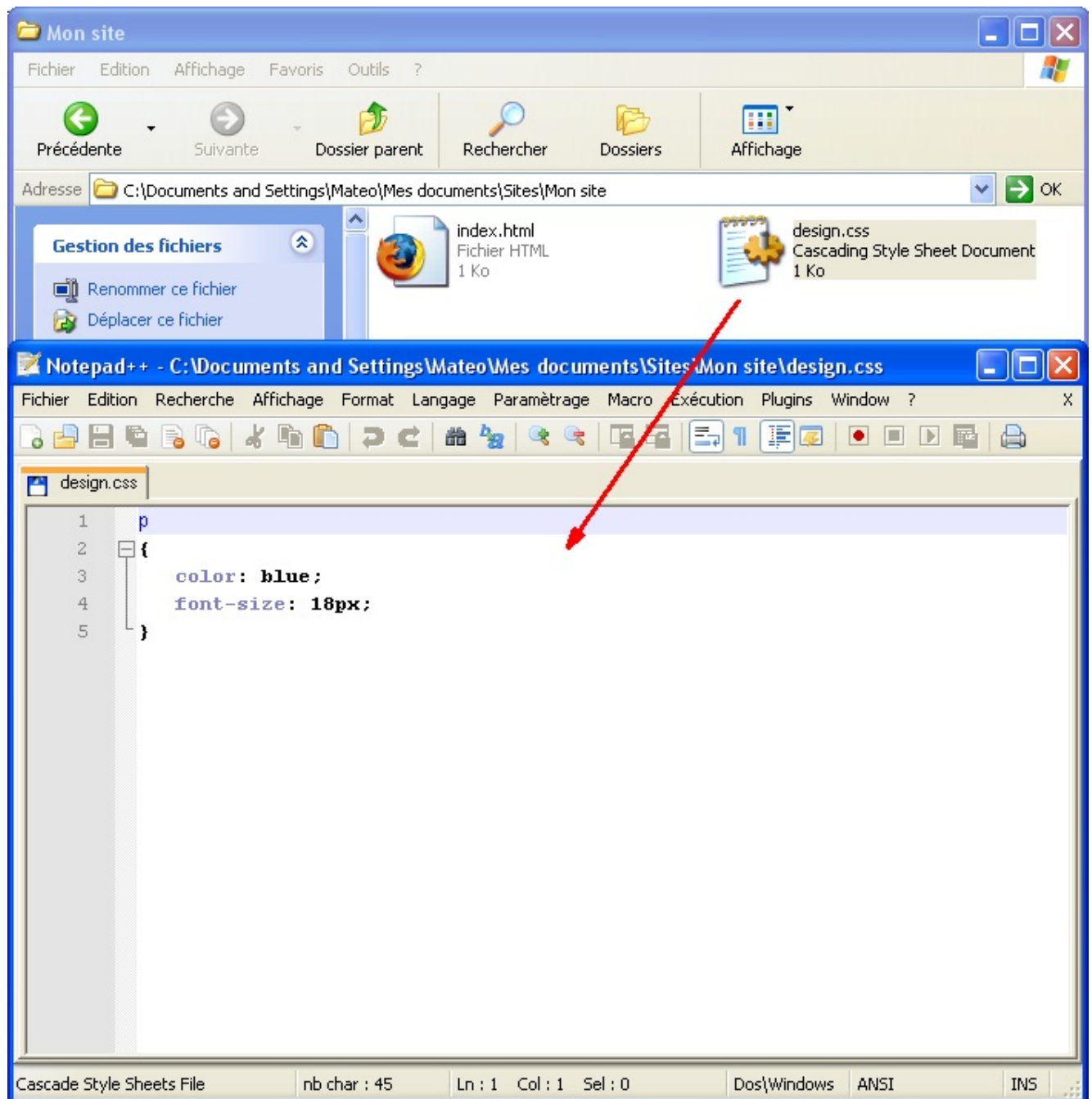


- Le fichier .html contient le code XHTML (qu'on a appris jusqu'ici) avec en particulier la ligne pour faire la liaison avec le fichier .css :



- Le fichier .css contient du code CSS (que nous allons apprendre à partir de maintenant) :





Si vous voulez voir le résultat, il faut ouvrir le fichier .html comme avant (double-cliquez sur l'icône du fichier index.html dans l'explorateur). N'essayez pas d'ouvrir le .css : seul, ce fichier ne sert à rien. C'est le .html qui, lors de son ouverture, va charger le .css et appliquer les informations de style.

Bien, maintenant que cela est clair pour tout le monde (du moins j'espère 😊), intéressons-nous de plus près au fonctionnement du code CSS.

## Appliquer un style : sélectionner une balise

Maintenant que nous savons où placer le code CSS, intéressons-nous au code de plus près. Je vous ai donné, sans vous l'expliquer, un premier bout de code CSS :

Code : CSS

```
p
{
    color: blue;
}
```

Dans un code CSS comme celui-ci, on trouve 3 éléments différents :

- Des noms de balises : on écrit les noms des balises dont on veut modifier l'apparence. Par exemple, si je veux modifier l'apparence de tous les titres `<h1>`, je dois écrire `h1`
- Des propriétés CSS : les "effets de style" de la page sont rangés dans des propriétés. Il y a par exemple la propriété `color` qui permet d'indiquer la couleur du texte, `font-size` qui permet d'indiquer la taille du texte, etc. Il y a beaucoup de propriétés CSS et, comme je vous l'ai dit, je ne vous obligerai pas à les connaître toutes par coeur (sauf s'il me prend une envie sadique de vous faire souffrir 😏).
- Les valeurs : à chaque propriété CSS on doit indiquer une valeur. Par exemple, pour la couleur, il faut indiquer le nom de la couleur. Pour la taille, il faut indiquer quelle taille on veut, etc.

Schématiquement, une feuille de style CSS ressemble à ça :

Code : CSS

```
balise1
{
    propriete: valeur;
    propriete: valeur;
    propriete: valeur;
}

balise2
{
    propriete: valeur;
    propriete: valeur;
    propriete: valeur;
    propriete: valeur;
}

balise3
{
    propriete: valeur;
}
```

Vous repérez sur ce schéma les balises, propriétés et valeurs dont je viens de vous parler.

Comme vous le voyez, on écrit le nom de la balise (par exemple `h1`), et on ouvre des accolades pour y mettre les propriétés et valeurs que l'on veut à l'intérieur.

On peut mettre autant de propriétés que l'on veut à l'intérieur des accolades. Chaque propriété est suivie du symbole "deux-points" (:) puis de sa valeur correspondante. Enfin, chaque ligne se termine par un point-virgule (;)



Le point-virgule n'est pas obligatoire s'il n'y a qu'une seule propriété pour la balise (comme c'est le cas pour la balise `p`). Toutefois, il vaut mieux prendre l'habitude de le mettre tout le temps, c'est plus sûr.

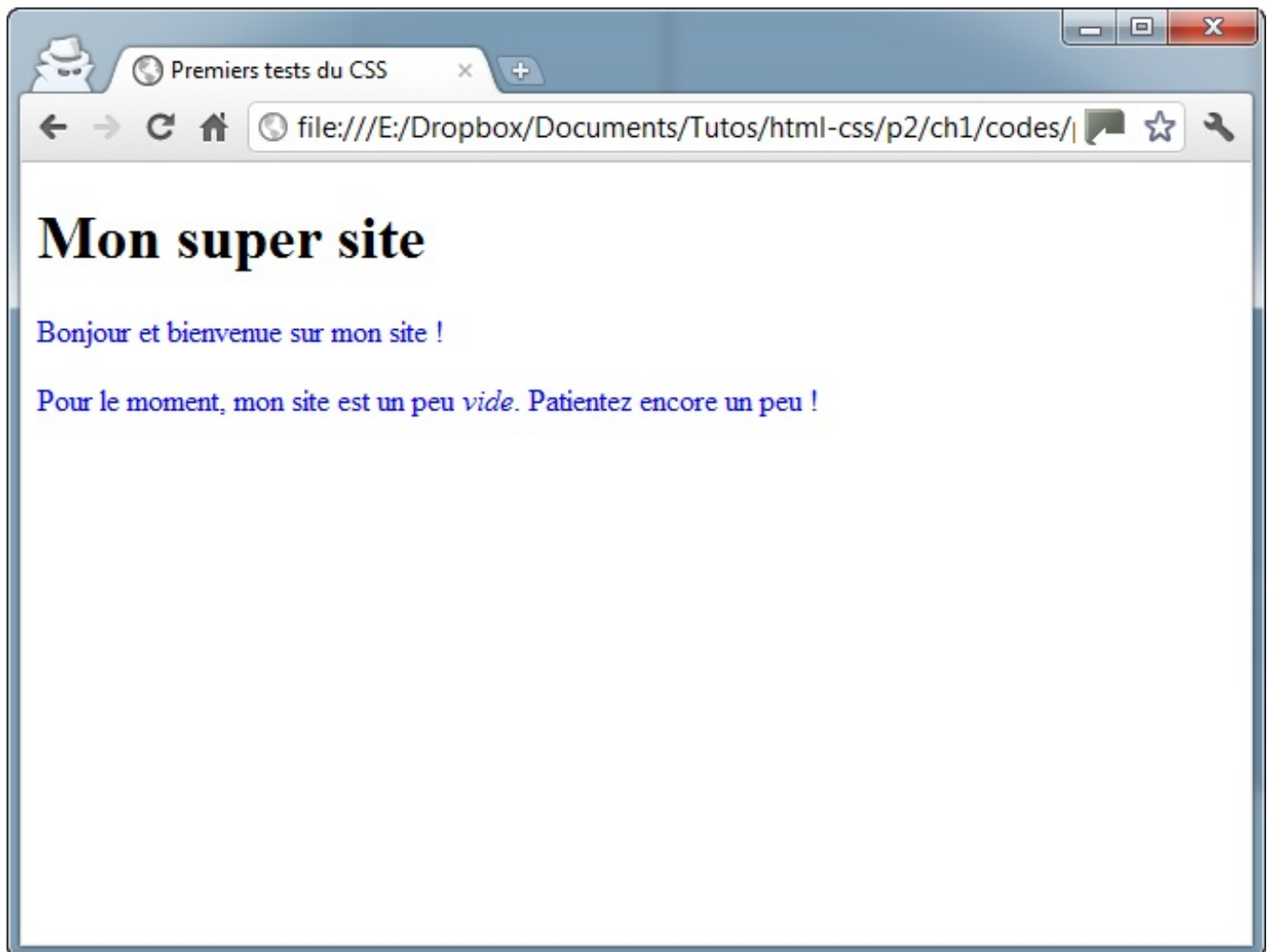
Je vous apprendrai de nombreuses propriétés dans les chapitres suivants. Pour le moment, on va juste changer la couleur pour s'entraîner dans nos exemples.

Le code CSS que nous avons utilisé jusqu'ici :

**Code : CSS**

```
p
{
    color: blue;
}
```

... signifie donc en français : *"Je veux que tous mes paragraphes soient écrits en bleu."* Il donne ce résultat à l'écran :



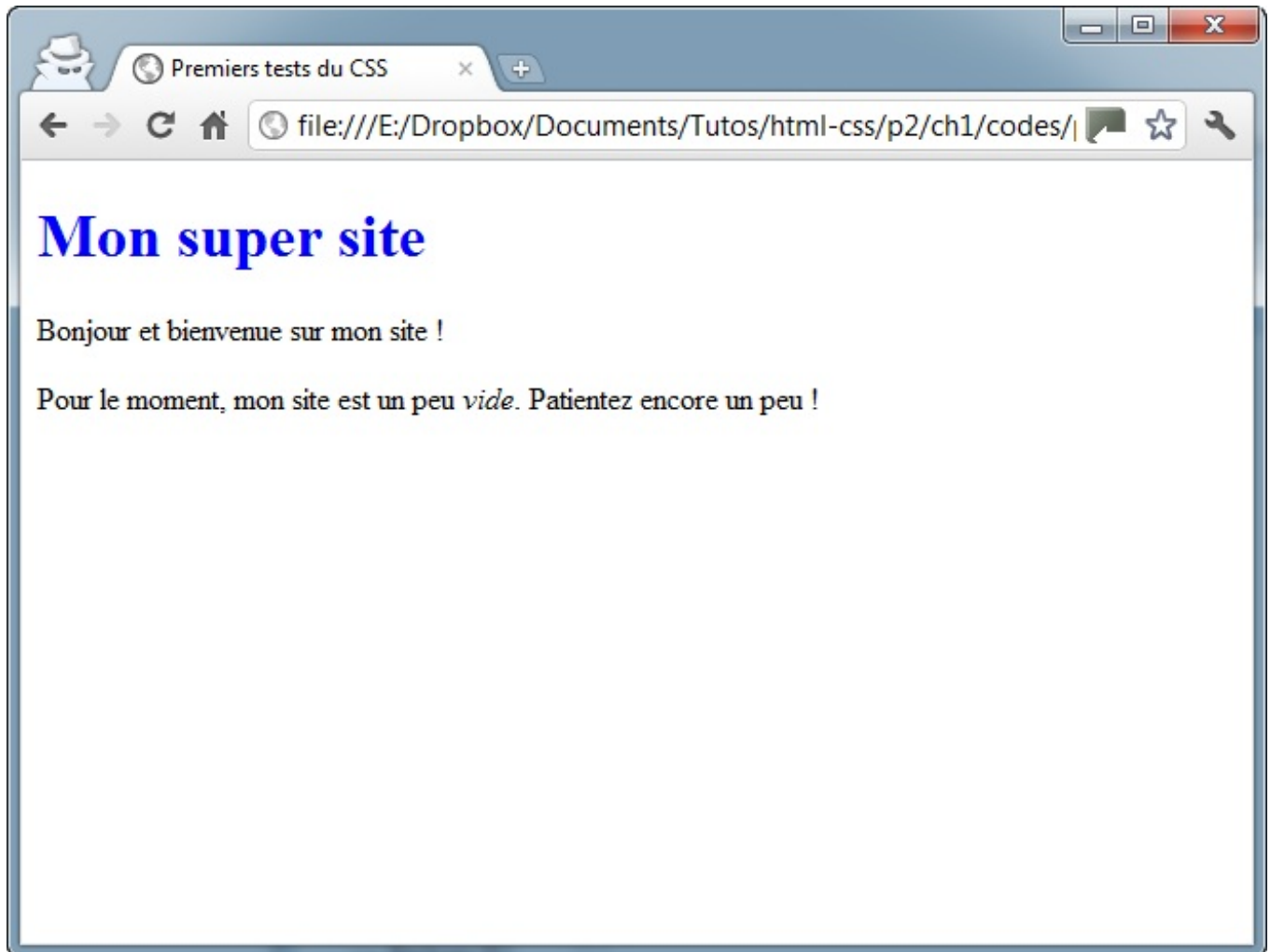
Essayez de changer le nom de la balise affectée par le code CSS. Par exemple, si j'écris `h1`, c'est le titre qui sera écrit en bleu. Modifiez votre fichier `style.css` comme ceci :

**Code : CSS**

```
h1
```

```
{  
    color: blue;  
}
```

Maintenant, ouvrez à nouveau votre page HTML (souvenez-vous, on ouvre la page HTML dans le navigateur, pas le fichier CSS !) : vous devriez voir son titre s'afficher en bleu !



## Appliquer un style à plusieurs balises

Prenons le code CSS suivant :

**Code : CSS**

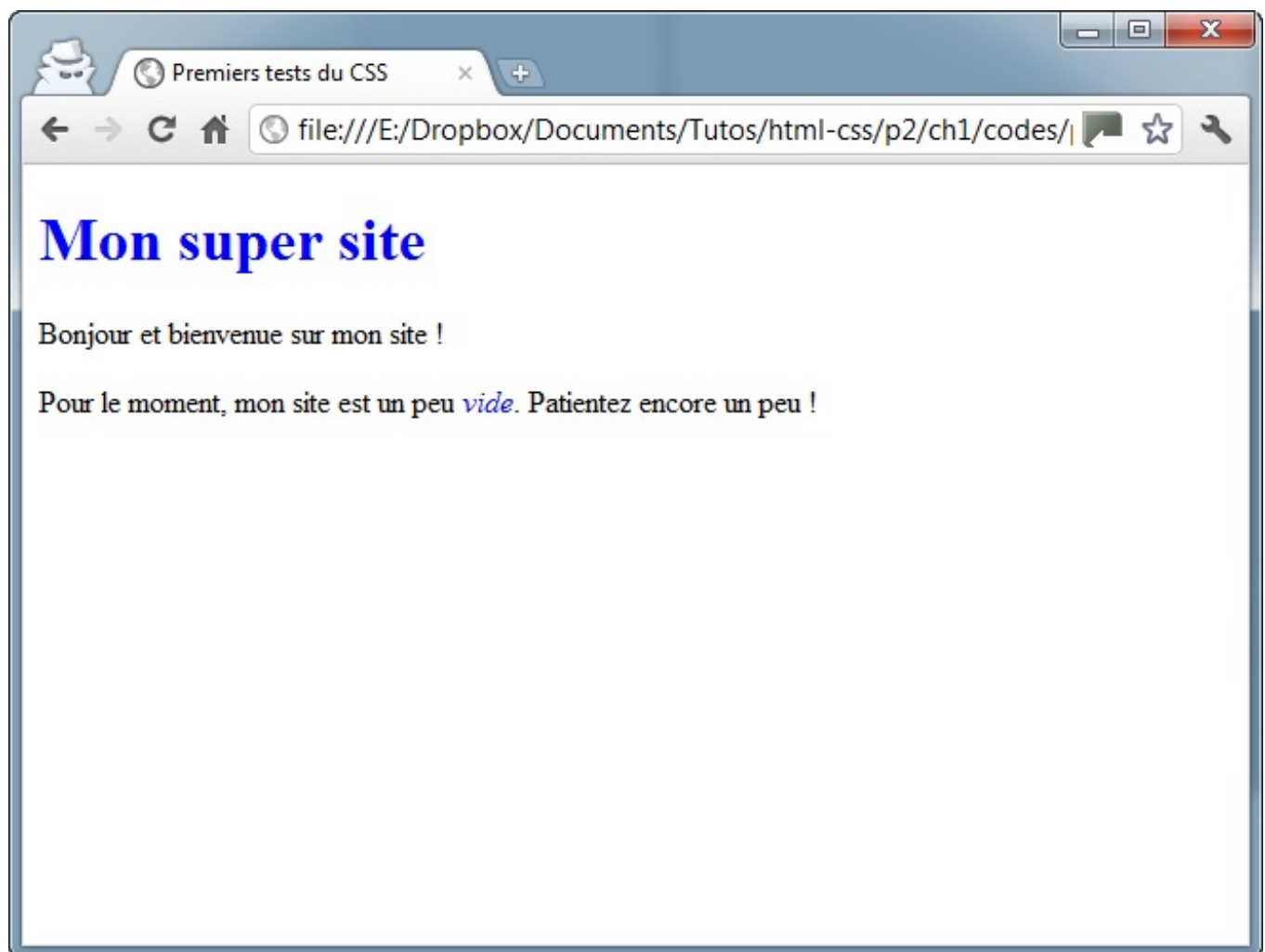
```
h1  
{  
    color: blue;  
}  
  
em  
{  
    color: blue;  
}
```

Il signifie que nos titres `<h1>` et nos textes importants `<em>` doivent s'afficher en bleu. Par contre, c'est un peu répétitif, vous ne trouvez pas ?

Heureusement, il existe un moyen en CSS d'aller plus vite si les styles de 2 balises doivent avoir la même présentation. Il suffit de combiner la déclaration en séparant les noms des balises par une virgule comme ceci :

#### Code : CSS

```
h1, em
{
    color: blue;
}
```



Cela signifie : "Je veux que le texte de mes `<h1>` et `<em>` soit écrit en bleu".

Vous pouvez indiquer autant de balises à la suite que vous le désirez.

## Des commentaires dans du CSS

Comme en HTML, il est possible de mettre des commentaires. Les commentaires ne seront pas affichés, ils servent simplement à indiquer des informations pour vous, par exemple pour vous y retrouver dans un long fichier CSS.

D'ailleurs, vous allez vous en rendre compte, en général le fichier HTML est assez petit, et la feuille CSS assez grande (si elle contient tous les éléments de style de votre site, c'est un peu normal). Notez qu'il est possible de créer plusieurs fichiers CSS pour son site si vous ressentez le besoin de séparer un peu votre code CSS (en fonction des différentes sections de votre site par exemple).

... De quoi on parlait déjà ? Ah oui, les commentaires en CSS. 😊

Donc, pour faire un commentaire, c'est facile ! Tapez */\**, suivi de votre commentaire, puis *\*/* pour terminer votre commentaire. Vos commentaires peuvent être écrit sur une ou plusieurs lignes. Par exemple :

#### Code : CSS

```
/*  
style.css  
-----  
  
Par Mathieu Nebra  
*/  
  
p  
{  
    color: blue; /* Les paragraphes seront bleus */  
}
```

Il est possible que j'utilise les commentaires dans la suite du cours pour vous donner des explications à l'intérieur même des fichiers `.css`.

---

## Appliquer un style : class et id

Ce que je vous ai montré jusqu'ici a quand même un défaut : cela implique par exemple que TOUS les paragraphes soient écrits par exemple en bleu.

Comment faire pour que seulement certains paragraphes (ou certains titres, ou certaines citations) soient écrits d'une manière différente ? On pourrait placer notre code CSS dans un attribut `style` sur la balise que l'on vise (c'est la technique que je vous ai présentée un peu plus tôt), mais comme je vous l'ai dit ce n'est pas recommandé (il vaut mieux utiliser un fichier CSS externe).

Pour résoudre le problème, on peut utiliser ces attributs spéciaux *qui fonctionnent sur toutes les balises* :

- L'attribut `class`
- L'attribut `id`

Que les choses soient claires dès le début : les attributs `class` et `id` sont quasiment identiques. Il y a seulement une petite différence que je vous dévoilerai plus bas.

Pour le moment et pour faire simple, on ne va s'intéresser qu'à l'attribut `class`.

Comme je viens de vous le dire, c'est un attribut que l'on peut mettre sur n'importe quelle balise, aussi bien titre que paragraphe, image, etc.

```
<h1 class=""></h1>
<p class=""></p>
<img class="" />
```



Oui mais que met-on comme valeur à l'attribut `class` ?

En fait, vous devez écrire un nom qui sert à identifier la balise. Ce que vous voulez.

Par exemple, je vais donner la classe *introduction* à mon premier paragraphe :

Code : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Premiers tests du CSS</title>
  </head>

  <body>
    <h1>Mon super site</h1>

    <p class="introduction">Bonjour et bienvenue sur mon site !</p>
    <p>Pour le moment, mon site est un peu <em>vide</em>.
    Patientez encore un peu !</p>
  </body>
</html>
```

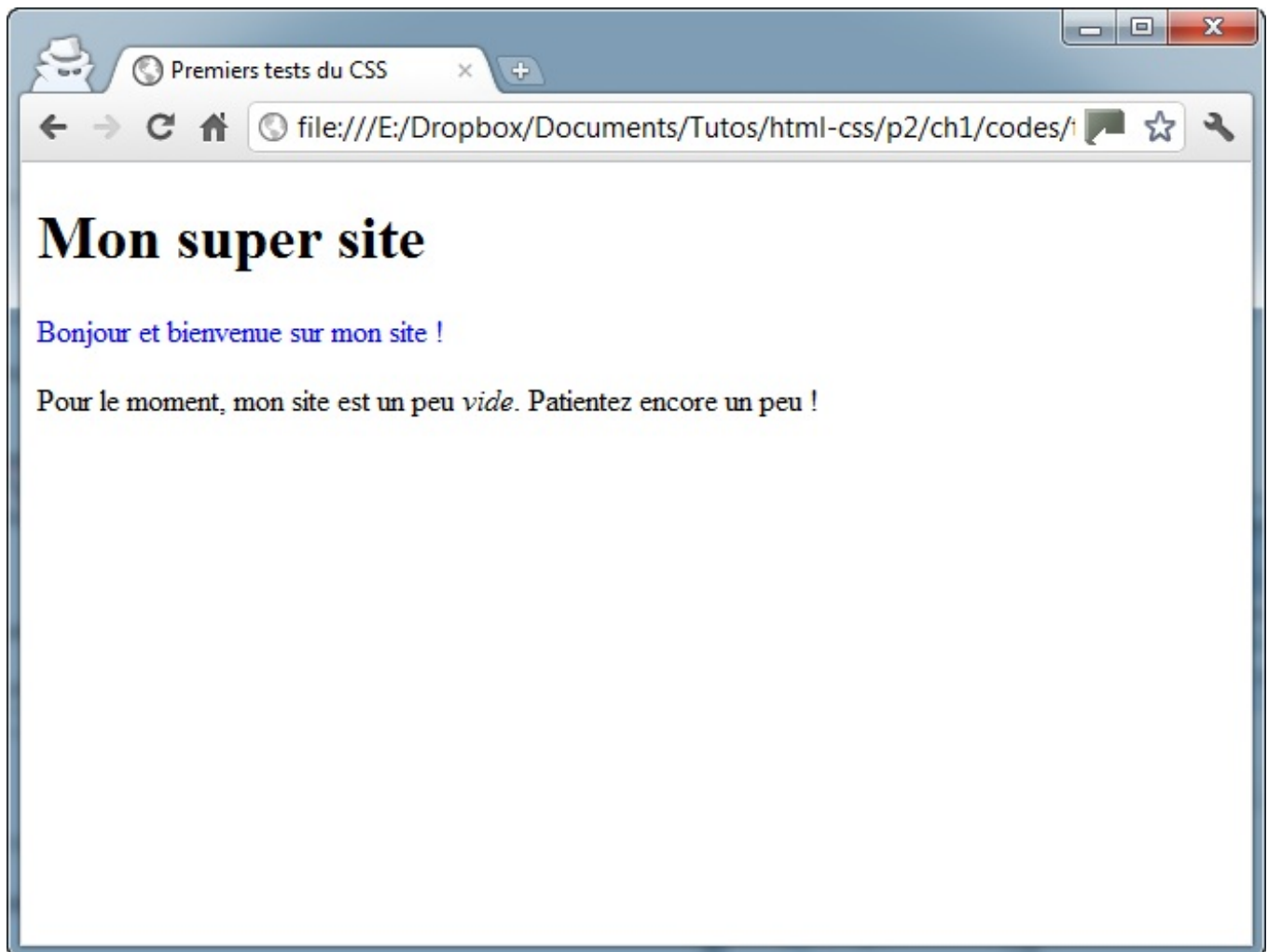
Maintenant que c'est fait, votre paragraphe est identifié. Il a un nom : *introduction*. Vous allez pouvoir réutiliser ce nom dans le fichier CSS pour dire : "Je veux que seules les balises qui ont comme nom *introduction* soient affichées en bleu".

Pour faire ça en CSS, indiquez le nom de votre classe en commençant par un point, comme ceci :

Code : CSS

```
.introduction
{
    color: blue;
}
```

Testez le résultat : seul votre paragraphe appelé "introduction" va s'afficher en bleu !



Et l'attribut `id` alors ?

Lui, il fonctionne exactement de la même manière que `class`, à un détail près : il ne peut être utilisé *qu'une fois*.

Quel intérêt ? Il y en a assez peu pour tout vous dire, cela vous sera utile si vous faites du Javascript plus tard pour reconnaître certaines balises. D'ailleurs, nous avons déjà vu l'attribut `id` dans le chapitre sur les liens (pour réaliser des ancrés).

En pratique, nous ne mettrons des `id` que sur des éléments qui sont uniques sur votre page, comme par exemple le logo :

Code : HTML

```

```



Si vous utilisez des id, dans le CSS il faudra faire précéder le nom de l'id par un dièse (#) :

#### Code : CSS

```
#logo
{
    /* Mettez les propriétés CSS ici */
}
```

Je ne vous propose pas de le tester, ça fonctionne exactement comme *class*.



Si vous vous emmêlez les pinceaux entre *class* et *id* (il ne devrait pas y avoir de raison, mais on sait jamais), vous pouvez utiliser tout le temps des *class* car ça marchera toujours.

## Les balises universelles

Il arrivera parfois que vous ayez besoin d'appliquer une *class* (ou un *id*) à certains mots qui ne sont pas à l'origine entourés par des balises.

En effet, le problème de *class*, c'est qu'il s'agit d'un attribut. Vous ne pouvez donc en mettre que sur une balise.

Par exemple, si je veux modifier uniquement "bienvenue" dans le paragraphe suivant :

#### Code : HTML

```
<p>Bonjour et bienvenue sur mon site !</p>
```

Ca serait facile à faire s'il y avait une balise autour de "bienvenue", malheureusement il n'y en a pas. Heureusement, on a inventé... la balise-qui-sert-à-rien. 😊

En fait, on a inventé 2 balises qui ne servent à rien, avec une petite (mais importante !) différence entre les deux :

- **<span> </span>** : c'est une balise de type *inline*. Vous vous souvenez ce qu'est une balise inline ? C'est une balise qui se met à l'intérieur d'un paragraphe, comme <strong>, <em>, <q>. Cette balise s'utilise donc au milieu d'un paragraphe, et c'est celle dont nous allons nous servir pour colorer "bienvenue".
- **<div> </div>** : c'est une balise de type *block*. Comme <p>, <h1> etc... Elle crée un nouveau "bloc" dans la page, et provoque donc obligatoirement un retour à la ligne. C'est une balise fréquemment utilisée dans la construction d'un design, comme nous le verrons plus tard

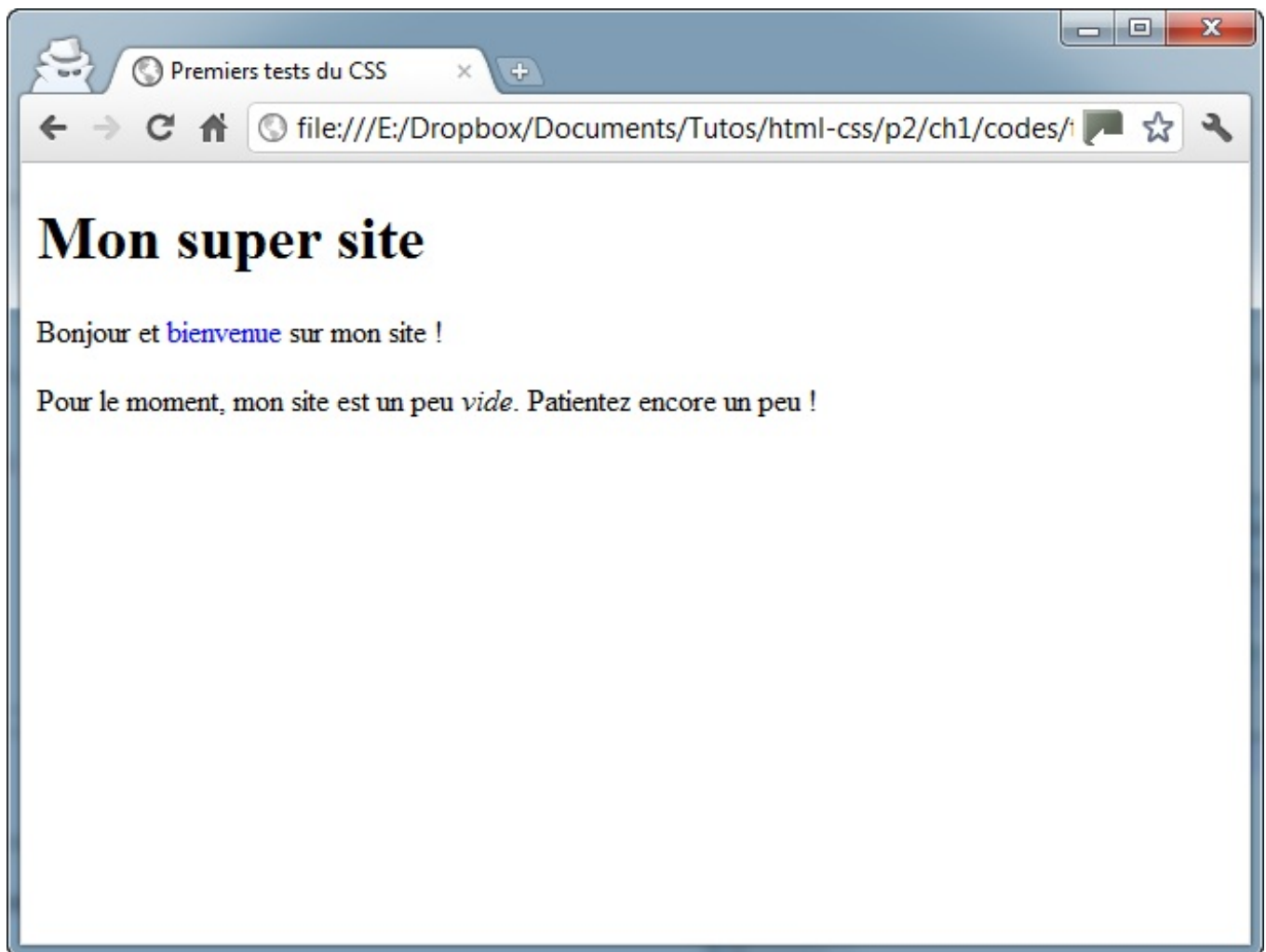
Pour le moment donc, nous allons utiliser plutôt la balise **<span>**. On la met autour de "bienvenue", on lui rajoute une *class* (du nom qu'on veut), on crée le CSS et c'est gagné ! 😊

#### Code : HTML

```
<p>Bonjour et <span class="salutations">bienvenue</span> sur mon
site !</p>
```

Code : CSS

```
.salutations  
{  
    color: blue;  
}
```



Que de nouveautés, que de nouveautés...

Eh oui, le CSS est un langage "à part", les règles ne sont pas les mêmes. Mais comme vous pouvez le voir, XHTML et CSS sont très liés, et on peut difficilement concevoir l'un sans l'autre 😊

Toutes les nouvelles notions que je vous ai apprises dans ce chapitre sont importantes, aussi prenez votre temps pour bien les assimiler. Rien ne presse, personne ne vous oblige à passer au chapitre suivant (à moins que vous ayez fait un pari stupide avec des amis du genre "Tu n'arriveras pas à apprendre à créer un site web en 2 jours" :p)

Si tant de nouveautés d'un seul coup vous font un peu peur, ne vous affolez pas pour autant : le plus dur c'est toujours d'apprendre "les bases". Or, c'est justement ce qu'on vient de faire. Tout le reste devrait couler de source à partir de maintenant 😊

Dans le prochain chapitre, nous allons découvrir un nombre important de propriétés CSS. Car pour le moment, à part écrire notre texte en bleu avec une taille de 18 pixels, on ne sait rien faire de bien palpitant 😊

## Formatage du texte en CSS (partie 1/2)

Nous arrivons maintenant à un chapitre qui devrait beaucoup vous intéresser 😊

Non, le "formatage du texte" n'a rien à voir avec la destruction de toutes les données présentes sur votre disque dur 😊 Cela signifie simplement que l'on va modifier l'apparence du texte (on dit qu'on le "met en forme"). Soyez rassurés donc, à la fin de ce chapitre votre disque dur sera toujours vivant 😊

Pas de surprise particulière : nous sommes toujours dans le CSS, et nous allons réutiliser ce que nous venons d'apprendre dans le chapitre précédent. J'espère donc, ceci dit en passant, que vous avez bien lu et compris comment on se servait d'un .css 😊

Cette fois, on passe au concret : nous allons voir successivement comment modifier la taille du texte, changer la police, aligner le texte etc... Vous avez une foule de choses à découvrir, et afin de ne pas vous assommer d'un chapitre trop gros j'ai décidé de le scinder en 2 parties.

La bonne nouvelle, c'est qu'à la fin de ces 2 chapitres votre site commencera (enfin) à ressembler à quelque chose 😊

---

## Taille du texte



Hé, mais modifier la taille du texte je sais déjà le faire ! Tu l'as expliqué dans le chapitre précédent !

En fait, dans le chapitre précédent je vous ai donné un exemple sans trop d'explications, afin d'illustrer un peu le fonctionnement d'un CSS. En réalité, les possibilités pour modifier la taille du texte sont nombreuses (et variées !).

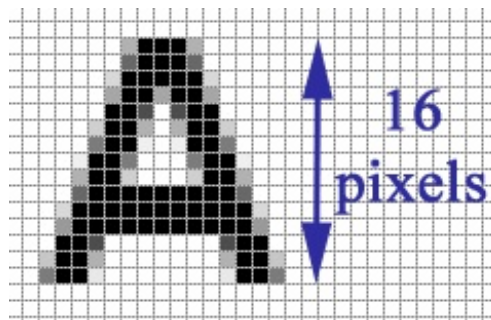
Ce que vous savez déjà, c'est que la propriété CSS qui permet de changer la taille du texte est *font-size*. Ca, ça ne change pas. Mais par contre, on peut indiquer la taille du texte de différentes manières :

- **En pixels** : c'est une façon très précise d'indiquer la taille du texte. C'est à vous de dire combien de pixels exactement doit faire le texte.

Pour avoir un texte de 16 pixels de hauteur, vous devez écrire :

```
font-size: 16px;
```

Les lettres auront une taille de 16 pixels, comme le montre l'image suivante :



Voici un exemple d'utilisation (je ne vous mets pas le fichier XHTML qui va avec, vous êtes des grands maintenant 😊) :

**Code : CSS**

```
p
{
    font-size: 14px; /* Paragraphes de 14 pixels */
}
h1
{
    font-size: 22px; /* Titres de 22 pixels */
}
```

Essayer !

Si vous êtes allergique aux pixels, sachez qu'il est aussi possible d'indiquer une taille en centimètres ("2cm" par exemple) et en millimètres ("14mm" par exemple).



Indiquer la taille du texte en pixels, centimètres et millimètres est très pratique et très précis. Certes. Mais il est préférable d'indiquer une taille relative (comme nous allons le voir) afin que la taille du texte s'adapte aux choix de tout le monde (certains préfèrent avoir du texte plutôt gros, d'autres plutôt petit etc...)

- **En donnant une valeur relative** : c'est-à-dire en écrivant carrément "gros", "très gros", "petit", "minuscule". Mais bien sûr, comme tout c'est en anglais :p. Voici la liste des différentes valeurs que vous pouvez mettre ainsi que leur signification :

- xx-small : minuscule
- x-small : très petit
- small : petit
- medium : moyen

- large : grand
- x-large : très grand
- xx-large : euh... gigantesque 😊

Voici un exemple de CSS qui utilise des valeurs relatives :

Code : CSS

```
.minuscule
{
    font-size: xx-small;
}
.trespetit
{
    font-size: x-small;
}
.petit
{
    font-size: small;
}
.moyen
{
    font-size: medium;
}
.grand
{
    font-size: large;
}
.tresgrand
{
    font-size: x-large;
}
.supermegagigatresgrand
{
    font-size: xx-large;
}
```

[Essayer !](#)



La taille du texte n'est peut-être pas exactement identique selon le navigateur... mais ce n'est pas bien grave, car dans tous les cas "xx-small" correspond à un texte très petit, et "xx-large" à un texte très grand. Et c'est tout ce qu'on a besoin de savoir 😊

Je vous encourage à utiliser cette méthode pour indiquer la taille du texte plutôt que d'utiliser des pixels. Votre site sera ainsi bien plus "adaptable" aux différentes configurations de vos visiteurs.

- **En em** : c'est une autre façon d'indiquer de manière relative la taille du texte. Un peu délicat à comprendre, je vous l'accorde, mais une fois qu'on a un peu testé ça vient tout seul, et on se rend compte que c'est une méthode vraiment pratique.

On doit indiquer la taille du texte par rapport à la taille normale de la police. Je m'explique : "1em" signifie "Taille normale". Si vous mettez un nombre supérieur (généralement un nombre décimal) comme "1.3em", le texte aura une taille 1,3 fois plus grande. De même pour réduire : "0.8em" et votre texte aura une taille 0,8 fois plus petite. C'est personnellement la méthode que j'utilise le plus souvent pour indiquer la taille du texte (la plupart de mes *font-size* sont donc en "em").



Faites attention, il faut mettre un point à la place de la virgule pour les nombres décimaux. Vous devez donc écrire "1.4em" et non pas "1,4em" !

Code : CSS

```
.petit_em
{
    font-size: 0.7em;
}
.grand_em
{
    font-size: 1.3em;
}
```

Essayer !

- **En pourcentage** : ça c'est facile. Si vous indiquez "100%", le texte aura une taille "normale". Si vous mettez "130%", le texte aura une taille correspondant à 130% de la taille normale. Ca ressemble énormément aux "em" (en fait c'est plus ou moins pareil). Après, tout est une question de goût 😊

Pfiou ! Comme quoi, il y a l'embaras du choix pour indiquer la taille du texte 😊

A tel point que l'on s'y perd un peu d'ailleurs...

Comme je vous l'ai dit, les pixels c'est très pratique pour être précis, mais ce n'est pas très "recommandé" car il se peut qu'une taille trop petite (ou trop grosse) gêne certaines personnes.

En ce qui me concerne, la méthode des "em" (ou des %) est la plus pratique : elle a l'avantage de s'adapter automatiquement aux préférences du visiteur et d'être facile à utiliser.

---

## Polices

Ah... La police... On touche un point sensible 😬

En effet, le problème c'est que, 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 vous, son navigateur prendra une police par défaut (une police standard) qui n'aura peut-être rien à voir avec ce à quoi vous vous attendiez.

Certes, tout le monde a "Times New Roman" et "Arial" me direz-vous... Et encore, sur Mac c'est "Time" et pas "Times New Roman".

Mais si vous aviez envisagé d'utiliser la dernière police gothique à la mode pour votre site de Heavy Metal, il va peut-être falloir revoir vos plans 😬



Bon, et concrètement comment on fait ?

La propriété CSS qui permet d'indiquer la police est *font-family*. Vous devez indiquer le nom de la police comme ceci :  
`font-family: police;`

Seulement, pour éviter qu'il n'y ait de problème si l'internaute n'a pas la même police que vous, on met en général **plusieurs** noms de police, séparés par des virgules :

`font-family: police1, police2, police3, police4;`

Le navigateur essaiera d'abord de mettre la police1. S'il ne l'a pas, il essaiera de mettre la police2. S'il ne l'a pas, il essaiera la police3 et ainsi de suite.

En général, on met en tout dernier "serif", ce qui correspond à une police standard (qui ne se met que si aucune autre police n'a été trouvée).

Oui, mais quelles sont les polices les plus courantes qu'on a le "droit" d'utiliser me direz-vous ?

Voici une liste de polices qui fonctionnent bien sur la plupart des navigateurs et que vous pouvez donc utiliser sans crainte :

- Arial
- Arial Black
- Comic Sans MS
- Courier New
- Georgia
- Impact
- Times New Roman
- Trebuchet MS
- Verdana

Si vous désirez tester les différentes polices listées ci-dessus, cliquez sur le bouton ci-dessous :

[Essayer !](#)

Ainsi, si j'écris :

`font-family: Impact, "Arial Black", Arial, Verdana, serif;`

... cela signifie : *"Mets la police Impact, ou, si elle n'y est pas, Arial Black, ou sinon Arial, ou sinon Verdana, ou si rien n'a marché mets une police standard (serif)"*

En général, il est bien d'indiquer un choix de 3-4 polices (+ serif) afin de s'assurer qu'au moins l'une d'entre elles aura été trouvée sur l'ordinateur du visiteur.



Si le nom de la police comporte des espaces, vous devez l'entourer de guillemets, comme je l'ai fait pour "Arial Black".

Allez, on se fait un petit CSS d'exemple pour récapituler tout ça et ça sera bon :

Code : CSS

```
h1
{
  font-family: "Arial Black", Arial, Verdana, serif; /* On essaie
d'avoir Arial Black en priorité */
}
p
{
  /* La police Comic Sans MS est agréable à lire pour les
paragraphe je trouve */
  font-family: "Comic Sans MS", "Trebuchet MS", Georgia, serif;
}
```

Essayer !



## Alignement

### Alignements simples

Le langage CSS nous permet de faire tous les alignements que l'on connaît : à gauche, centré, à droite et justifié.

C'est tout simple. On utilise la propriété `text-align`, et on indique l'alignement désiré :

- **left** : le texte sera aligné à gauche (c'est le réglage par défaut).
- **center** : le texte sera centré.
- **right** : le texte sera aligné à droite.
- **justify** : le texte sera "justifié". Justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes. Les textes des journaux, par exemple, sont tout le temps justifiés.

Regardez les différents alignements sur cet exemple :

#### Code : CSS

```
h1
{
    text-align: center; /* Pour centrer le titre */
    font-family: "Arial Black", Arial, Verdana, serif; /* Un titre en
    Arial Black c'est mieux :o;) */
}
blockquote
{
    text-align: justify; /* La citation sera justifiée */
}
.signature
{
    text-align: right; /* Pour aligner à droite ma signature */
    font-family: "Comic Sans MS", Georgia, "Times New Roman", serif;
    font-size: 80%;
}
```

N'oubliez pas qu'il y a toujours du XHTML derrière, bien entendu. D'ailleurs pour cet exemple, je vais vous montrer le code que j'utilise. Sachez toutefois qu'il est toujours possible d'afficher le code XHTML d'une page en faisant Clic droit / "Afficher le code source".

Voici donc le code XHTML de cet exemple permettant de tester les alignements :

#### Code : HTML

```
<h1>La parole du sage</h1>

<p>Un jour, un grand sage a dit :</p>

<blockquote><p>Morbi fermentum libero non libero. Nunc in turpis in
justo adipiscing scelerisque. Donec id elit non diam aliquet semper.
Maecenas pede. Maecenas fringilla. Fusce eleifend dui quis lectus.
Praesent facilisis, ligula a consequat posuere, metus purus porta
mi, at consectetur justo wisi id dui. Maecenas mattis. Ut imperdiet
pharetra enim. Suspendisse quis leo nec arcu interdum aliquam.
Vivamus dictum quam id tellus. Maecenas in quam sit amet risus
semper auctor. Integer leo dui, malesuada eu, fermentum at, vehicula
at, nisl. Pellentesque hendrerit. Proin ut libero. Curabitur sem
ipsum, porta non, feugiat vel, mollis ut, justo. Sed a orci id metus
pretium lobortis. Morbi ultrices, quam a facilisis faucibus, odio
nunc dignissim enim, eget rhoncus purus erat ac
quam.</p></blockquote>
```

```
<p class="signature">Signé : M@teo21</p>
```

Essayer !



**Vous ne pouvez pas modifier l'alignement du texte d'une balise *inline* (comme *span*, *a*, *em*, *strong*...). L'alignement ne fonctionne que sur des balises de type *block* (*p*, *div*, *blockquote*, *h1*, *h2*, ...), et c'est un peu logique quand on y pense : on ne peut 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.**

## L'indentation

Qu'est-ce que l'indentation ? C'est tout simplement la mise en retrait du texte. Cela permet par exemple de faire commencer un paragraphe un peu plus à droite, ce qui rend (je trouve) la lecture beaucoup plus facile et agréable. C'est un procédé que l'on retrouve dans la plupart des livres d'ailleurs.

Voici un exemple de paragraphe indenté :

Bien que cela :  
plus en plus inquié  
telles hostilités ven  
*Texte indenté*

On utilise la propriété *text-indent*. On doit indiquer quelle est la taille du retrait : on peut le faire en pixels, en centimètres, en millimètres...

Bref, là le mieux à mon avis c'est d'utiliser les pixels ^^ . Regardez comment je mets tous les textes des paragraphes en retrait avec *text-indent* :

### Code : CSS

```
p
{
  text-indent: 30px; /* Les paragraphes commenceront 30 pixels sur
la droite */
  text-align: justify; /* Ils seront justifiés */
  font-size: large; /* Allez, soyons fous, grossissons le texte
:0;) */
}
```

Essayer !

L'avantage en CSS, c'est qu'il vous suffit de dire une fois pour toutes : "*Je veux que tous mes paragraphes soient indentés de 30 pixels sur la droite*" et à chaque fois que vous écrirez un nouveau paragraphe, celui-ci sera automatiquement mis en retrait. Avant le XHTML (à l'époque du HTML) ce genre de chose était impossible, ou plutôt difficile et répétitif. Là, c'est un jeu d'enfant d'aligner son texte comme on veut 😊

A noter 2 propriétés similaires sur lesquelles je n'ai pas besoin de m'étendre :

- *word-spacing* : l'espace entre les mots (en pixels).
- *letter-spacing* : l'espace entre les lettres (en pixels).

Et voilà une moitié du morceau d'avalée 😊

Dès que vous vous sentez en forme, rendez-vous au chapitre suivant pour découvrir encore tout plein de propriétés CSS de formatage du texte 😊

---

## Formatage du texte en CSS (partie 2/2)

Bienvenue dans la seconde partie de ces 2 chapitres dédiés au formatage du texte 😊

Le CSS vous réserve encore bien des surprises, alors n'attendez pas : foncez ! 😊

---

## Effets de style

Il existe en CSS une série de propriétés étonnantes comme sympathiques qui permettent de donner un peu plus de style à vos pages web (et je crois que ça ne sera pas de refus 😊)

Cela va de la mise en gras, italique, souligné à la mise en capitales, en passant par la possibilité de faire clignoter le texte ! 😊

### Mettre en italique



Attends attends là ! On se calme ! Je croyais que la balise `<em>` permettait de mettre un texte en italique ?!

Je n'ai jamais dit ça 😊

Retournez voir les chapitres précédents si vous avez des doutes, mais je n'ai jamais dit que la balise `<em>` était faite pour mettre le texte en italique (de même que je n'ai jamais dit que `<strong>` était fait pour mettre en gras).

`<em>`, mettez-vous bien ça dans la tête, est fait pour insister sur des mots. Ca veut dire que les mots qu'il entoure sont assez importants.

Pour représenter cette importance, la plupart des navigateurs mettent le texte en italique (et ce n'est pas une obligation).

Si on a la possibilité de mettre en italique en CSS, c'est donc utile juste **à des fins de présentation**. Parce que, je vous le rappelle, le CSS sert uniquement à la présentation de vos pages web.

Concrètement, pour mettre en italique en CSS on utilise *font-style*, qui peut prendre 3 valeurs :

- **italic** : le texte sera mis en italique.
- **oblique** : le texte sera mis en italique. Quoi là aussi ? 😊 Euh, pour tout vous dire je n'en sais rien moi-même, mais tout ce que je constate c'est que "italic" et "oblique" reviennent au même. Choisissez donc l'un des deux 😊
- **normal** : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique. Par exemple, si vous voulez que les textes entre `<em>` ne soient plus en italique, vous devrez écrire :

Code : CSS

```
em
{
    font-style: normal;
}
```

Sur l'exemple suivant, je me sers par exemple de *font-style* pour mettre en italique tous mes titres `<h2>` :

Code : CSS

```
h1
{
    text-align: center;
    font-family: Arial, "Times New Roman", Verdana, serif;
}
h2
{
    font-style: italic; /* Les titres h2 seront en italique ! */
    text-indent: 30px; /* On décale un peu les sous-titres */
    font-family: Arial, "Times New Roman", "Arial Black", Verdana,
serif;
}
```

[Essayer !](#)

Bref, c'est facile à utiliser, pas la peine de traîner plus longtemps là-dessus 😊

## Mettre en gras

Et si nous passions à la mise en gras ?

Alors là, pareil pour `<strong>`, je ne vous refais pas le même speech que tout à l'heure. La mise en gras en CSS permet de mettre en gras par exemple les titres, certains paragraphes entiers etc... C'est à vous de voir.

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).

On n'a qu'à s'en servir pour mettre les paragraphes en gras, tenez :

Code : CSS

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

[Essayer !](#)

## Les majuscules en CSS

Le CSS permet d'appliquer des effets très intéressants sur du texte, en modifiant automatiquement les majuscules. Nous allons voir 2 propriétés CSS qui travaillent sur les majuscules.

Commençons par la propriété *font-variant*, toute simple, qui prend uniquement 2 valeurs différentes :

- [small-caps](#) : le texte sera écrit en petites capitales.
- [normal](#) : le texte sera écrit normalement (par défaut).

Code : CSS

```
p
{
    font-variant: small-caps;
}
```

[Essayer !](#)

Vous écrivez vos paragraphes normalement (comme d'habitude), et vous laissez CSS s'occuper de transformer ça pour vous automatiquement en petites capitales. C'est-y pas beau ? 😊

Mais attendez, il y a une seconde propriété CSS qui travaille sur les majuscules elle aussi : *text-transform*. Elle peut prendre ces valeurs :

- **uppercase** : tout le texte sera écrit en majuscules.
- **lowercase** : tout le texte sera en minuscules.
- **capitalize** : la première lettre de chaque mot sera en majuscule.
- **none** : pas de transformation (par défaut).

Allez, ça faisait un moment : pour cet exemple je vous mets le code XHTML qui va avec 😊

Regardez en particulier comment je me sers des class pour créer (entre autres) une class pousser\_une\_gueulante 😊

#### Code : HTML

```
<h1>Je suis très en colère...</h1>

<p>...mais je vais essayer de rester calme. Enfin, si possible, mais
je promets rien hein...<br />
<span class="pousser_une_gueulante">ahhhh !!! non cette fois je
craque ! qui c'est qui a mis de la mayo dans mes frites !??<br
/>sacrilège !</span></p>
<p class="chuchoter">CECI EST UN PARAGRAPHE QUE JE CHUCHOTE, ALORS
QUE POURTANT DANS LE CODE XHTML IL EST ECRIT EN MAJUSCULES</p>
```

#### Code : CSS

```
h1
{
    text-align: center;
    font-family: Arial, "Times New Roman", Verdana, serif;
    text-transform: capitalize; /* Les premières lettres des mots du
titre seront en majuscules */
}

.pousser_une_gueulante
{
    text-transform: uppercase; /* Si j'ai envie de me faire entendre,
je mets en majuscules */
}

.chuchoter
{
    text-transform: lowercase;
    font-style: italic; /* Le texte chuchoté sera en minuscules et
italique */
}
```

[Essayer !](#)

Cette propriété *text-transform* est vraiment pratique pour changer l'apparence de tout un texte en un clin d'oeil !

Comme quoi, le CSS n'est pas que bon à aligner le texte et changer la police. Il peut aussi agir **directement** sur un texte déjà écrit pour modifier ses majuscules / minuscules.

## Un peu de déco ?

Cette propriété CSS porte bien son nom : *text-decoration*

Elle permet, entre autres, de souligner le texte. Voici les différentes valeurs qu'elle peut prendre :

- **underline** : souligné.
- **line-through** : barré.

- **underline** : ligne au-dessus.
- **blink** : clignotant. Attention, cette propriété ne marche pas sous Internet Explorer ni sous Google Chrome. Elle fonctionne en revanche bien sur tous les autres navigateurs, dont Mozilla Firefox.
- **none** : normal (par défaut).

Ce CSS va vous permettre de tester les effets de text-decoration :

#### Code : CSS

```
h1
{
    text-align: center;
    font-family: "Arial Black", Arial, "Times New Roman", serif;
    text-decoration: blink; /* Le titre sera clignotant ! (ne
fonctionne pas sous Internet Explorer) */
}
.souligne
{
    text-decoration: underline;
}
.barre
{
    text-decoration: line-through;
}
.ligne_dessus
{
    text-decoration: overline;
}
```

Au passage, voici le XHTML qui va avec :

#### Code : HTML

```
<h1>A ne pas manquer !</h1>

<p>La propriété CSS <em>text-decoration</em> permet de décorer un
peu son texte :<br />
<span class="souligne">en le soulignant</span>...<br />
<span class="barre">en le barrant</span>...<br />
...ou encore <span class="ligne_dessus">en mettant une ligne au-
dessus</span>.</p>
```

[Essayer !](#)

N'oubliez pas que l'intérêt du CSS c'est aussi de pouvoir cumuler les styles. Là j'ai créé une class "souligne" qui ne fait que souligner pour l'exemple, mais dans la pratique on crée souvent des class combinant les styles, comme par exemple une class "important" qui souligne le texte, le met en gras, et l'écrit en plus gros 😊

## Les couleurs

Passons maintenant au vaste sujet de la **couleur** 🤔



Comment ça vaste ? 🤔

Eh bien on a plusieurs possibilités pour indiquer une couleur, comme c'était le cas avec la taille du texte. Nous allons ici voir quelles sont toutes ces possibilités qu'offre le CSS pour choisir une couleur.

Première chose à savoir : la propriété qui permet de changer la couleur du texte est *color* (facile à retenir ;)), vous l'avez d'ailleurs entraperçue dans notre introduction au CSS.

### Indiquer le nom de la couleur

La méthode la plus simple et la plus pratique pour choisir une couleur est de taper son nom (in english, of course). Le seul défaut de cette méthode est qu'il n'existe que 16 couleurs dites "standard". D'autres couleurs officieuses existent, mais comme elles ne fonctionneront pas forcément pareil sur tous les navigateurs, je vais éviter de vous les montrer.

Voici les 16 couleurs que vous pouvez utiliser en tapant simplement leur nom :

Couleur	Aperçu
white	<div></div>
silver	<div></div>
grey	<div></div>
black	<div></div>
red	<div></div>
maroon	<div></div>
lime	<div></div>
green	<div></div>
yellow	<div></div>
olive	<div></div>
blue	<div></div>
navy	<div></div>
fuchsia	<div></div>
purple	<div></div>
aqua	<div></div>
teal	<div></div>

Vous pouvez les apprendre par coeur si ça vous chante, en plus ça vous fera réviser votre anglais 🤪

Voici le CSS de test :



## Code : CSS

```

h1
{
  text-align: center;
  font-family: Arial, "Arial Black", "Times New Roman", serif;
  text-decoration: underline;
  color: green; /* Le titre en vert (pourquoi pas ?) */
}
p
{
  text-indent: 20px;
  color: blue; /* Les paragraphes en bleu */
}
strong /* ... et les mots importants en rouge clignotant ! */
{
  color: red;
  text-decoration: blink;
}

```

... et la page XHTML qui va avec :

## Code : HTML

```

<h1>De toutes les couleurs</h1>

<p>Salut et bienvenue dans cette page haute en couleurs ! J'utilise
des <strong>noms de couleurs standards</strong> dans mon CSS pour
égayer un peu la page. Ainsi, "red" signifie "rouge", "blue"
signifie "bleu" etc.</p>

<p>Grâce à l'attribut <em>color</em> du CSS, j'ai (entre autres) pu
convertir <strong>automatiquement</strong> tous mes mots importants
(dans une balise "strong" ;) ) en textes rouge clignotant ! Comme
ça, on ne risque pas de les louper ;o)</p>

```

[Essayer !](#)

Les textes importants en rouge clignotant... ben ouais, fallait y penser ! Allez, un peu d'imagination quoi 😊

## La notation hexadécimale

16 couleurs, c'est quand même un peu limite quand on sait que la plupart des écrans peuvent en afficher 16 millions. D'un autre côté remarquez, s'il avait fallu donner un nom à chacune des 16 millions de couleurs...

Heureusement, il existe plusieurs façons en CSS de choisir une couleur parmi toutes celles qui existent. La première que je vais vous montrer est la notation hexadécimale.

Je ne vais pas m'attarder dessus car elle n'est pas très pratique à manier, mais elle reste encore souvent utilisée par habitude. En effet, avant que le CSS n'apparaisse, c'était une méthode courante pour définir une couleur (en fait c'était la seule :p). Heureusement depuis, on a inventé des méthodes plus simples.

Un nom de couleur en hexadécimal, ça ressemble à ça : #FF5A28. Pour faire simple, c'est une combinaison de lettres et de chiffres qui indiquent une couleur.

On doit toujours commencer par écrire un dièse (#), suivi de 6 lettres ou chiffres allant de 0 à 9 et de A à F.

Ces lettres ou chiffres fonctionnent deux par deux. Les 2 premiers indiquent une quantité de rouge, les 2 suivants une quantité

de vert, et les 2 derniers une quantité de bleu. En mélangeant ces quantités (qui sont les composantes Rouge-Vert-Bleu de la couleur) on peut obtenir la couleur qu'on veut.

Ainsi, #000000 correspond à la couleur noire et #FFFFFF à la couleur blanche. Mais maintenant, me demandez pas quelle est la combinaison qui produit du orange couleur "coucher de soleil", je n'en sais strictement rien 😬



Il se peut que certains logiciels de dessin, comme Photoshop, vous indiquent les couleurs en hexadécimal. Il vous sera alors facile de copier-coller le code hexadécimal d'une couleur dans votre fichier CSS.

Voici par exemple comment on fait pour appliquer la couleur blanche en hexadécimal :

```
color:#FFFFFF;
```

## La méthode RGB

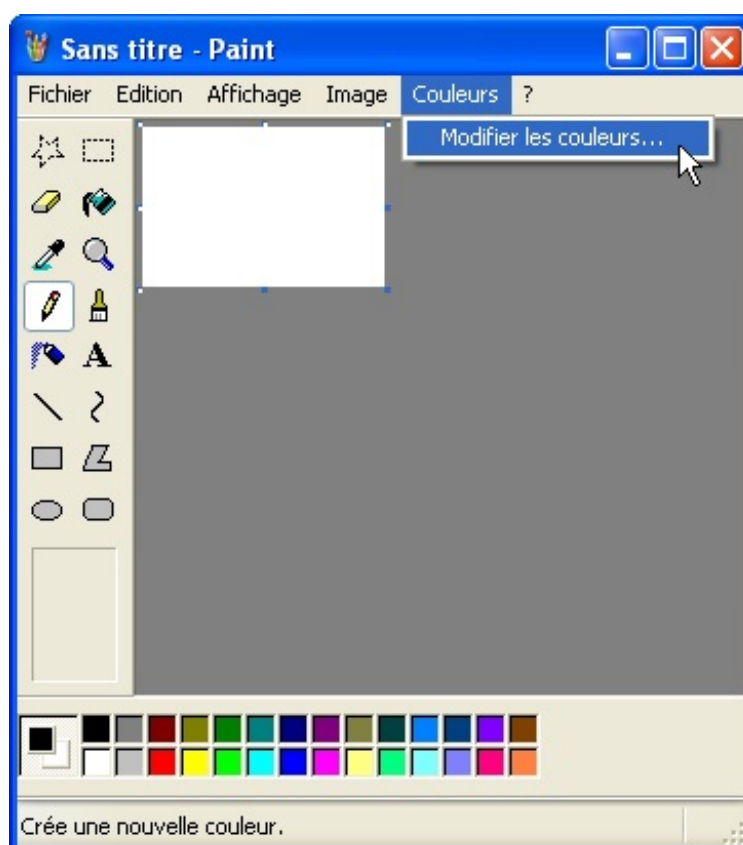
Que signifie RGB ? En anglais, Rouge-Vert-Bleu s'écrit Red-Green-Blue, ce qui s'abrège en "RGB". Comme pour la notation hexadécimale, on doit définir une quantité de rouge, de vert et de bleu pour choisir une couleur.



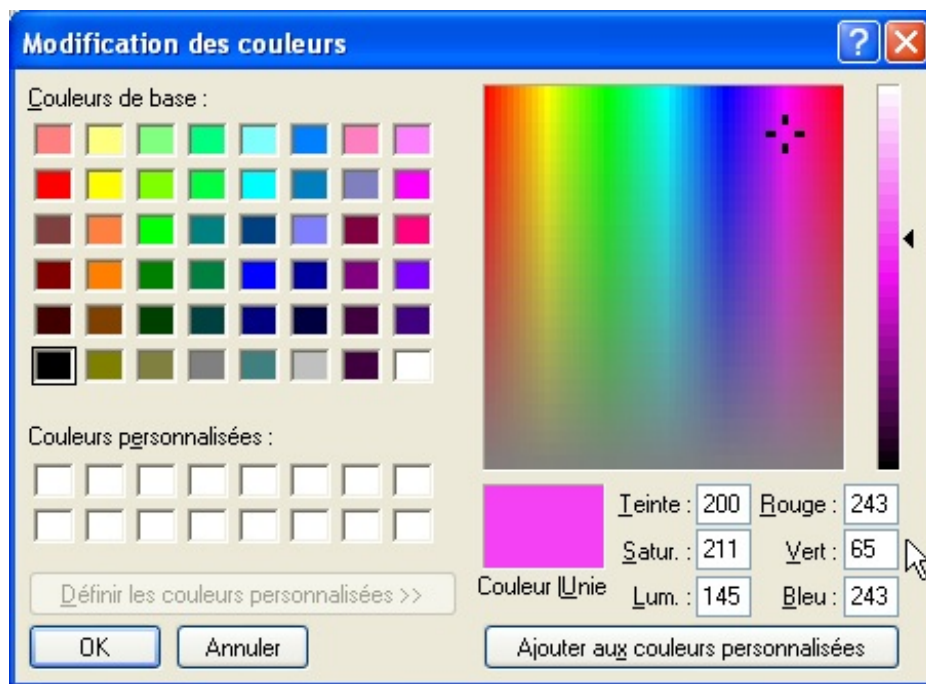
Quoi ?! Encore cette histoire tordue de quantités de rouge-vert-bleu ?

Oui, mais là vous allez voir que c'est beaucoup plus pratique et qu'avec un logiciel de dessin tout simple comme Paint, vous pouvez trouver la couleur que vous désirez. Voici la marche à suivre :

1. Lancez le logiciel Paint depuis le menu Démarrer.
2. Rendez-vous dans le menu Couleurs / Modifier les couleurs :



3. Une fenêtre s'ouvre. Cliquez sur le bouton "Définir les couleurs personnalisées" en bas. Dans la zone qui apparaît à droite, faites bouger les curseurs pour sélectionner la couleur qui vous intéresse.
4. Supposons que je sois pris d'une envie folle d'écrire mes titres <h1> en rose barbie (supposons seulement). Je sélectionne la couleur dans la fenêtre, comme ceci :



5. On relève les quantités de Rouge-Vert-Bleu correspondantes indiquées en bas à droite de la fenêtre (ici 243-65-243). Je recopie ces valeurs dans cet ordre dans le fichier CSS, comme ceci :

**Code : CSS**

```
h1
{
  text-align: center;
  color: rgb(243, 65, 243);
}
```

Essayer !

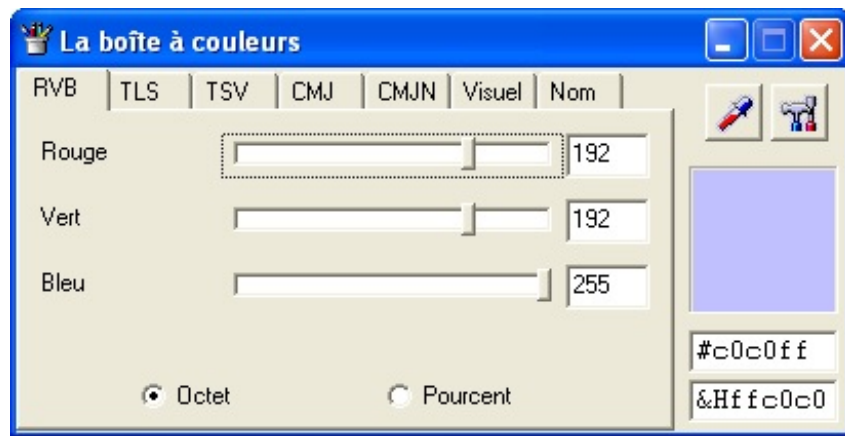
Et voilà le travail ! 😊

Comme vous avez pu le constater dans l'exemple, pour utiliser la méthode RGB il faut taper *rgb(Rouge, Vert, Bleu)* en remplaçant "Rouge, Vert, Bleu" par les nombres correspondants.

Pour votre information, ces nombres vont de 0 à 255. Si vous écrivez un jour une quantité de rouge de 327, c'est qu'il y a un problème 🤪

## Et en Bonus Track...

Je mets à votre disposition un petit logiciel tout simple, spécialisé dans le choix d'une couleur. Nul doute qu'il vous sera très utile pour vous aider à choisir vos couleurs. Ce logiciel s'appelle "La boîte à couleurs" (pas compliqué comme nom), et il ressemble à ceci :



Bien entendu, il est en français, totalement gratuit et il est téléchargeable directement depuis le Site du Zéro. Bref, ce serait bien bête de ne pas l'essayer 😊

[Télécharger "La Boîte à Couleurs" \(1,45 Mo\)](#)

Il y a plusieurs onglets comme vous pouvez le voir. Je vous recommande de rester sur le premier ("RVB") ou d'aller dans l'onglet "Visuel". Les autres ne nous concernent pas, et évitez en particulier l'onglet "Nom" qui propose des noms de couleurs parfois invalides (je vous rappelle qu'il existe seulement 16 noms de couleurs "standards").

Vous pouvez récupérer en bas à droite le numéro de la couleur en hexadécimal (le numéro commence toujours par un #), ou encore recopier les valeurs de Rouge-Vert-Bleu (RVB) dans le CSS.

Enfin, et c'est certainement la fonctionnalité la plus intéressante du logiciel, vous pouvez utiliser sur la pipette en haut à droite pour récupérer n'importe quelle couleur s'affichant sur votre écran !

Amusez-vous bien ! 😊

## Le fond

Contrairement à ce qu'on pourrait croire, le fond ne désigne pas forcément le fond de toute une page web. On peut aussi appliquer un fond uniquement aux titres, ou aux paragraphes, ou encore à certains mots d'un paragraphe.

Il faut tout d'abord savoir qu'il existe 2 types de fonds :

- Les fonds comportant une couleur
- Les fonds comportant une image de fond

Nous allons commencer à nous intéresser à la couleur de fond dans un premier temps, puis nous verrons comment faire pour avoir une image de fond.

### La couleur de fond

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

Pour indiquer la couleur de fond de la page web, il faut travailler sur la balise `<body>`. Eh oui, `<body>` correspond à toute la page web, c'est donc en modifiant sa couleur de fond que l'on changera la couleur de fond de la page web.

Regardez très attentivement ce fichier CSS :

#### Code : CSS

```
body /* On travaille sur la balise body, donc sur TOUTE la page */
{
  background-color: black; /* Le fond de la page sera noir */
  color: white; /* Le texte de la page sera blanc */
}
```

[Essayer !](#)



Hé, mais tu as indiqué une couleur de texte blanche à la balise `<body>`, et tous les paragraphes `<p>` et titres `<h1>` ont pris cette couleur. Comment cela se fait-il ?

Je voulais justement profiter de l'occasion pour vous en parler. Ce phénomène s'appelle **l'héritage**. Non non, rassurez-vous, il n'y a pas eu de morts 🤪

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur de cette balise prendront le même style.

Gné ? 🤔

C'est en fait simple à comprendre et intuitif. La balise `<body>`, vous le savez, contient entre autres les balises de paragraphe `<p>` et de titre `<h1>`.

SI j'applique une couleur de fond noire et une couleur de texte blanche à la balise `<body>`, tous mes titres et paragraphes auront eux aussi une couleur de fond noire et un texte de couleur blanche... C'est ce phénomène qui s'appelle l'héritage : on dit que les balises qui se trouvent à l'intérieur d'une autre balise "héritent" de ses propriétés.



Cela veut dire que TOUT le texte de ma page web sera forcément écrit en blanc ?

Non, pas obligatoirement. Si vous dites par la suite que vous voulez vos titres en rouge, ce style aura la priorité et vos titres seront donc en rouge. En revanche, si vous n'indiquez rien de particulier (comme on l'a fait tout à l'heure), alors vos titres

hériteront de la couleur blanche.

Cela ne fonctionne pas uniquement pour la couleur, entendons-nous bien. Toutes les propriétés CSS seront héritées : vous pouvez par exemple demander une taille de texte de "1.3em" dans la balise <body>, et tous vos titres et paragraphes seront de cette taille-là.

Voici un exemple où je vous montre comment on "annule" l'héritage pour que nos titres ne soient pas écrits en blanc. J'en ai profité pour créer une class "surligne" qui met le texte sur fond jaune pour donner une impression de surlignage.

#### Code : CSS

```
body
{
    background-color: black;
    color: white; /* Toutes les balises contenues dans body verront
leur texte coloré en blanc... */
}
h1
{
    color: red; /* ... sauf si je demande expressément de changer la
couleur par la suite */
}
.surligne /* Un style qui permet par exemple de surligner certains
mots d'un texte */
{
    background-color: yellow;
    color: black; /* Le texte surligné sera écrit en noir, parce que
le blanc sur fond jaune on voit rien ;o) */
}
```

[Essayer !](#)

Comme vous pouvez le constater, on n'a pas indiqué de couleur particulière pour les paragraphes (<p>), donc ils ont hérité de la couleur blanche. En revanche, le titre n'a pas hérité de la couleur blanche car on a écrit qu'on voulait qu'il soit en rouge. La class "surligne" vous montre qu'on peut sans problème appliquer une couleur de fond à certains mots d'un texte. Et l'effet est plutôt sympa non, qu'en dites-vous ? 😊

## L'image de fond

Tout comme pour la couleur de fond, l'image de fond ne s'applique pas forcément à la page entière, on peut tout aussi bien mettre une image de fond aux titres, paragraphes, citations etc...

La propriété permettant d'indiquer une image de fond est *background-image*. Comme valeur, on doit lui mettre `url("nom_de_l_image.png")`. Par exemple :

```
background-image:url("fond.png");
```

Bien entendu, votre fond n'est pas forcément en PNG, il peut aussi être en JPEG ou en GIF.

L'adresse indiquant où se trouve l'image de fond peut être en absolu (`http://...`) ou en relatif (`fond.png`).



Attention lorsque vous mettez une adresse en relatif dans le fichier CSS ! L'adresse de l'image doit être indiquée par rapport au fichier .css et non pas par rapport au fichier .html.

Ainsi, si votre site comporte 2 dossiers : "css" et "images", il vous faudra taper : `../images/fond.png` pour récupérer l'image de fond. Si vous ne mettez pas le chemin correct, votre image de fond ne s'affichera pas.

Si on veut appliquer une image de fond à toute la page, on doit là encore utiliser la balise <body> :

#### Code : CSS

```
body
```

```
{
    background-image: url("../images/neige.png");
}
h1
{
    font-style: italic;
    font-family: "Arial Black", Arial, Verdana, serif;
    text-align: center;
}
blockquote p /* Tous les paragraphes contenus dans un blockquote */
{
    text-align: justify;
    text-indent: 25px;
}
```

[Essayer !](#)

Il existe par ailleurs une propriété CSS qui permet de "fixer" le fond, pour ne pas qu'il bouge en même temps que le texte. L'effet obtenu est, je trouve, intéressant.

La propriété concernée répond au doux nom de `background-attachment` et peut prendre 2 valeurs :

- [fixed](#) : l'image de fond reste fixe.
- [scroll](#) : l'image de fond défile avec le texte (par défaut).

En réutilisant le même fichier XHTML que tout à l'heure, mais en changeant un peu le CSS pour y rajouter *background-attachment*, on obtient ceci :

#### Code : CSS

```
body
{
    background-image: url("../images/neige.png");
    background-attachment: fixed; /* Le fond restera fixe */
}
h1
{
    font-style: italic;
    font-family: "Arial Black", Arial, Verdana, serif;
    text-align: center;
}
blockquote p
{
    text-align: justify;
    text-indent: 25px;
}
```

[Essayer !](#)

Essayez de descendre plus bas dans la page avec les barres de défilement, vous verrez que le fond reste fixe

Il reste encore 2 propriétés en rapport avec les images de fond que je souhaite vous montrer.

La première d'entre elle est celle qui gère **la répétition de l'image de fond**. Cette propriété s'appelle *background-repeat* et peut prendre ces valeurs :

- [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 (par défaut).

Gardons encore notre même fichier XHTML, mais appliquons cette fois un fond dégradé qui se répète uniquement verticalement.

Voici l'image de fond que j'ai créée moi-même tout seul comme un grand sous Photoshop (oui ça mérite d'être souligné tant mon niveau en dessin est faible :p) :



Un dégradé

#### Code : CSS

```
body
{
    background-image: url("../images/degrade.png");
    background-repeat: repeat-y; /* Le fond ne se répètera que sur la
première colonne, verticalement */
}
h1
{
    font-style: italic;
    font-family: "Arial Black", Arial, Verdana, serif;
    text-align: center;
}
blockquote p
{
    text-align: justify;
    text-indent: 25px;
}
```

Essayer !

Enfin, la dernière des propriétés sur le fond que je tenais à vous montrer (comme ça on les aura toutes vues) concerne la position de l'image de fond.

On peut indiquer où doit se trouver l'image de fond avec *background-position*. Cette propriété n'est intéressante que si vous avez mis "background-repeat: no-repeat;" (un fond qui ne se répète pas).

Vous devez donner à *background-position* 2 valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page (ou du paragraphe si vous appliquez le fond à un paragraphe). Ainsi, si vous tapez :

`background-position: 30px 50px;`

... votre fond sera placé à 30 pixels de la gauche et à 50 pixels du haut. Il est aussi possible de mettre ces valeurs en anglais :

- top : en haut.
- bottom : en bas.
- left : à gauche.
- center : centré.
- right : à droite.

Il est possible de combiner ces mots. Par exemple, pour aligner une image en haut à droite, vous taperez `:background-position: top right;`

Allez, pour ce dernier exemple je vais réutiliser toutes les propriétés sur le fond qu'on a apprises 🤖

#### Code : CSS

```
body
{
```



```

    background-image: url("../images/skieur.gif"); /* Le fond est
l'image "skieur.gif" */
    background-repeat: no-repeat; /* Le fond ne se répète pas */
    background-position: top right; /* Le fond est aligné en haut à
droite */
    background-attachment: fixed; /* Le fond est fixé */
}
h1
{
    font-style: italic;
    font-family: "Arial Black", Arial, Verdana, serif;
    text-align: center;
}
blockquote p
{
    text-align: justify;
    text-indent: 25px;
}

```

[Essayer !](#)

Si vous utilisez beaucoup de propriétés en rapport avec le background (comme c'est le cas sur cet exemple), vous pouvez utiliser une sorte de "méga-propriété" *background* qui peut prendre plusieurs valeurs combinées des propriétés background-image, background-repeat, background-attachment et background-position.

C'est la première "méga-propriété" que je vous montre, il y en aura d'autres. Pour toutes les "méga-propriétés" comme *background*, il faut savoir que :

- L'ordre des valeurs n'a pas d'importance. Vous pouvez combiner les valeurs dans n'importe quel ordre :  
background: url("../images/skieur.gif") no-repeat top right fixed;  
background: no-repeat fixed top right url("../images/skieur.gif");
- Vous n'êtes pas obligés de mettre toutes les valeurs. Ainsi, si vous ne voulez pas mettre fixed, vous pouvez l'enlever sans problème :  
background:url("../images/skieur.gif") no-repeat top right;

La "méga-propriété" n'est intéressante que si vous avez plusieurs valeurs à combiner bien entendu 😊

L'exemple ci-dessous donne le même résultat que l'exemple précédent, mais il utilise *background* pour combiner les valeurs et rendre le fichier CSS plus petit :

#### Code : CSS

```

body
{
    background: url("../images/skieur.gif") no-repeat top right
fixed;
}
h1
{
    font-style: italic;
    font-family: "Arial Black", Arial, Verdana, serif;
    text-align: center;
}
blockquote p
{
    text-align: justify;
    text-indent: 25px;
}

```

[Essayer !](#)

Une dernière chose : dans tous ces exemples, j'ai appliqué un fond à la page entière (body). Mais cela ne doit pas vous faire oublier qu'on peut appliquer un fond à n'importe quel élément (un titre, un paragraphe, certains mots d'un paragraphe etc)... Je vous conseille donc pour vous entraîner d'essayer d'appliquer un fond à vos titres ou paragraphes. Si vous avez un peu de goût (ce que je n'ai pas) vous arriverez certainement à donner une très belle allure à votre page web 😊

---

Les possibilités en CSS sont larges, n'est-ce pas ? Pour tout vous dire, elles sont quasi-infinies et... sans vouloir vous décourager, vous n'avez pas tout vu 😊

La seule vraie petite difficulté dans cette affaire, c'est qu'il est délicat de retenir toutes ces propriétés CSS par cœur pour savoir laquelle utiliser au bon moment.

Enfin, moi je sais pas vous mais j'ai pas une mémoire d'éléphant. En pratiquant bien entendu, on finit par retenir sans s'en rendre compte, mais au début c'est un peu galère. Heureusement, [une annexe](#) est mise à votre disposition pour récapituler toutes les propriétés CSS que nous avons vues, afin que vous puissiez aller directement à l'essentiel 😊

Allez, et ne vous arrêtez pas en si bon chemin, vous n'avez pas encore vu le meilleur 😊

---

## Les pseudo-formats

Nous venons de passer en revue un grand nombre de propriétés CSS dans les chapitres précédents. Vous savez maintenant modifier la taille du texte, sa police, sa couleur etc etc...

Nous allons voir dans ce chapitre un nouvel aspect du langage CSS qu'on appelle... *les pseudo-formats*

Nous n'allons pas apprendre de nouvelles propriétés CSS (vous en savez déjà pas mal), nous allons plutôt voir comment les appliquer à des moments ou des endroits précis. Par exemple, nous allons apprendre à modifier l'apparence d'un lien au passage de la souris, à modifier automatiquement la première lettre d'un paragraphe etc...

Tous ces éléments devraient vous permettre d'ajouter encore plus de dynamisme à votre site web. Que du bon pour vous quoi 😊

---

## Des liens sympas

Si vous avez bien suivi les chapitres précédents, vous savez comment modifier l'apparence des liens. Il suffit d'appliquer des styles à la balise `<a>` et le tour est joué.

Par défaut, les liens s'affichent en bleu et sont soulignés. Supposons que vous ne vouliez pas du soulignement, vous allez utiliser :

```
text-decoration:none;
```

... ce qui aura pour effet d'annuler le soulignement. Vos liens ne seront alors plus soulignés.

Bon jusque-là, j'espère que je ne vous apprends rien de trop nouveau 😊

Allez, petite révision pour se chauffer un peu, après on passe aux choses sérieuses. Voici un CSS qui applique 2-3 styles aux liens pour changer des immondes liens bleus soulignés 🤢

### Code : CSS

```
a
{
    text-decoration: none; /* Les liens ne seront plus soulignés */
    color: red; /* Les liens seront en rouge au lieu de bleu */
    font-style: italic; /* Les liens seront en italique (pourquoi pas
?) */
}
```

[Essayer !](#)

Nous allons apprendre à modifier l'apparence des liens :

- ... lorsque le visiteur pointe dessus avec la souris.
- ... lorsque le visiteur clique dessus.
- ... lorsque le visiteur a sélectionné le lien.
- ... lorsque le visiteur a déjà vu la page.

## Au passage de la souris

Le premier pseudo-format que je vais vous montrer s'appelle `:hover`. Comme tous les autres pseudo-formats que nous allons voir, c'est une information que l'on rajoute après le nom de la balise (ou de la class) dans le CSS, comme ceci :

### Code : CSS

```
a:hover
{
}
```

`":hover"` signifie "Dessus".

`"a:hover"` signifie donc : "Quand la souris est sur le lien" (quand on pointe dessus).

A gauche, vous tapez comme d'habitude la balise concernée (en l'occurrence `<a>`, la balise de lien), et à droite vous mettez le pseudo-format.

A partir de là, c'est à vous de définir l'apparence que doivent avoir les liens lorsqu'on pointe dessus. Laissez libre cours à votre imagination, il n'y a pas de limite 😊

Voici un exemple de présentation des liens, mais n'hésitez pas à inventer le vôtre :

**Code : CSS**

```
a
{
    text-decoration: none; /* Les liens ne seront plus soulignés */
    color: red; /* Les liens seront en rouge au lieu de bleu */
    font-style: italic; /* Les liens seront en italique (pourquoi pas
?) */
}
a:hover /* Quand le visiteur pointe sur le lien */
{
    text-decoration: underline; /* Le lien deviendra souligné quand
on pointera dessus */
    color: green; /* Le lien sera écrit en vert quand on pointera
dessus */
}
```

Essayer !

Sympa , n'est-ce pas ? 😊

La bonne nouvelle, c'est que vous pouvez appliquer le pseudo-format ":hover" à n'importe quelle balise. La mauvaise nouvelle, c'est que pour Internet Explorer 6 le ":hover" ne marche que sur les liens. Sous Internet Explorer 7, il marche en théorie sur toutes les balises, mais d'après mes tests il y a quand même quelques bugs.

Sur tous les autres navigateurs (dont Mozilla Firefox, que j'espère que vous avez téléchargé depuis le temps 😊) ça fonctionne à la perfection.

Voici un exemple d'utilisation du :hover sur un paragraphe (à tester avec un autre navigateur que IE) :

**Code : CSS**

```
p:hover /* Quand on pointe sur un paragraphe */
{
    background-color: #CFE1EB; /* Le fond du paragraphe change de
couleur */
    text-indent: 20px;
}
```

Essayer !

Si vous pointez sur les paragraphes, vous voyez donc qu'ils changent de couleur. Pas très utile, mais après tout l'objectif du CSS c'est de décorer non ? 😊

## Au moment du clic

Le pseudo-format **:active** permet d'appliquer un style particulier au moment du clic. Le lien gardera cette apparence très peu de temps : juste pendant que vous cliquez avec le bouton de la souris. En clair, ce n'est pas forcément toujours bien visible.

Pour ma part, un effet que je trouve bon d'appliquer avec **:active**, c'est de changer la couleur de fond du lien. Comme ça, on voit bien qu'on a cliqué 😊 :

**Code : CSS**

```
a:active /* Quand le visiteur clique sur le lien */
{
    background-color: #FFCC66;
}
a:hover /* Quand le visiteur pointe sur le lien */
{
    text-decoration: underline;
    color: green;
}
a /* Lien normal */
{
    text-decoration: none;
    color: red;
    font-style: italic;
}
```

[Essayer !](#)

## Lorsque le lien est sélectionné

Là, c'est à peine différent. Le pseudo-format **:focus** applique un style lorsque le lien est sélectionné.



C'est-à-dire ?

C'est-à-dire... que ça revient un peu comme **:active**, c'est-à-dire au moment du clic (en tout cas pour un lien).

Ce pseudo-format, appliqué à d'autres balises XHTML que nous n'avons pas encore vues, permettra de créer des effets assez sympas, vous verrez 😊

Pour ce qui est des liens, je vous mets un exemple très similaire au précédent (on change le fond) pour que vous puissiez comparer :

Code : CSS

```
a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color: #FFCC66;
}
a:hover /* Quand le visiteur pointe sur le lien */
{
    text-decoration: underline;
    color: green;
}
a /* Lien normal */
{
    text-decoration: none;
    color: red;
    font-style: italic;
}
```

[Essayer !](#)

Vous le voyez : le lien garde sa couleur de fond un peu plus longtemps. Personnellement, je préfère utiliser un **:focus** à la place d'un **:active** car je trouve ça plus visible.

Mais comme IE (dans ses anciennes versions) ne comprend pas le focus, j'applique le même style au **:focus** et au **:active**, comme on a appris à le faire dans le premier chapitre CSS, en séparant les noms par une virgule :

**Code : CSS**

```
a:active, a:focus /* Appliquer le même style aux liens actifs et focus */
{
    background-color: #FFCC66;
}
```

L'avantage de cette technique, c'est que si le navigateur est IE, il prendra uniquement le :active (et le fond sera un peu coloré), tandis que si c'est un autre navigateur le fond restera plus longtemps grâce au :focus.

A vous de vous faire votre propre idée, je ne vous oblige à rien 😊

## Quand la page a déjà été vue...

Il est possible d'appliquer un style à un lien vers une page qui a déjà été vue. Par défaut, le navigateur colore le lien en un violet immonde (encore plus immonde que le bleu souligné :p)

Personnellement, je préfère éviter de modifier l'apparence des liens qui ont déjà été vus car je trouve que c'est assez peu agréable au final visuellement. Mais là encore, c'est une opinion personnelle, je ne veux pas vous influencer 😊

Le pseudo-format qui nous intéresse s'appelle **:visited** (qui signifie "visité"). Une application marrante (mais pas très utile) de ce pseudo-format pourrait être de barrer tous les liens qui ont été vus :

**Code : CSS**

```
a:visited /* Quand le visiteur a déjà vu la page concernée */
{
    text-decoration: line-through;
}
a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color: #FFCC66;
}
a:hover /* Quand le visiteur pointe sur le lien */
{
    text-decoration: underline;
    color: green;
}
a /* Lien normal */
{
    text-decoration: none;
    color: red;
    font-style: italic;
}
```

[Essayer !](#)

Si vous avez cliqué sur tous les liens, ils seront tous barrés et vous n'y verrez plus grand chose, forcément 😊

Bon, plus sérieusement, si vous voulez appliquer un style précis à des liens déjà visités, je peux vous conseiller de colorer les liens en légèrement plus clairs que la normale. Si vos liens sont verts en temps normal, mettez-les en vert clair lorsqu'ils ont été visités.

Enfin, c'est une suggestion comme une autre, mais c'est un effet qui passe plutôt bien je trouve.

Bon allez, j'arrête de vous proposer mes idées, sinon tous vos sites vont se ressembler 😊

D'ailleurs, on a fini de voir les pseudo-formats qu'on utilise généralement sur les liens. On va maintenant s'intéresser aux pseudo-formats modifiant la première lettre ou la première ligne.

---

## Première lettre et première ligne

En CSS, il est possible de modifier automatiquement l'apparence de la première lettre et de la première ligne du texte contenu dans une balise.

Il s'agit là encore de pseudo-formats. Si on a tendance à utiliser ceux-là plutôt sur des paragraphes, n'oubliez pas que ça fonctionne aussi bien sur d'autres balises (comme les titres).

### La première lettre

Pour modifier l'apparence de la première lettre, vous devez utiliser le pseudo-format **:first-letter**

On peut s'en servir pour écrire en gras et en plus gros la première lettre de tous les paragraphes de votre page.

Si en plus vous indentez votre texte avec *text-indent*, on va finir par se croire dans un roman ou un article de journal 😊

Code : CSS

```
p:first-letter /* La première lettre de chaque paragraphe */
{
  font-weight: bold; /* En gras */
  font-size: 1.2em; /* Ecrit légèrement plus gros que la normale */
  color: blue; /* En bleu */
}
p
{
  text-indent: 20px;
}
```

[Essayer !](#)

### La première ligne

Un autre pseudo-format intéressant permet de modifier cette fois la première ligne. Il s'agit de **:first-line**.

Appliqué aux paragraphes, il permet d'inciter à la lecture du texte. Par exemple, vous pourriez automatiquement écrire en petites capitales la première ligne de chaque paragraphe, afin de les rendre plus attrayants :

Code : CSS

```
p:first-line /* La première ligne de chaque paragraphe */
{
  font-variant: small-caps; /* En petites capitales */
}
p
{
  text-indent: 20px;
}
```

[Essayer !](#)

Bref, pas besoin d'y passer 3/4 d'heure non plus, c'est facile à utiliser donc je vous laisse faire joujou avec ces nouvelles possibilités 😊



## Avant / Après

Les pseudo-formats que nous allons voir maintenant sont très intéressants et changent un peu de ce qu'on a l'habitude de faire en CSS. C'est toutefois un peu plus difficile que le reste, aussi ouvrez grand vos oreilles parce que ça en vaut la peine.

Bon, de quoi parle-t-on ici ?

On parle de la possibilité d'ajouter automatiquement du texte au début et à la fin de certains paragraphes, afin de diminuer la quantité de texte que vous avez à écrire.

Par exemple, supposons que je pose beaucoup de questions dans ma page web. En temps normal, j'écrirai ceci en XHTML pour indiquer qu'une personne pose une question :

### Code : HTML

```
<p>Question : quel est l'âge du webmaster ?</p>
```

[Essayer !](#)

Maintenant, supposons que je sois un webmaster fainéant (et vous n'avez pas idée du nombre de webmasters fainéants qui existent, j'en fais d'ailleurs partie :p). Je souhaite automatiser l'insertion du texte "Question : " au début et du point d'interrogation à la fin, pour ne pas avoir à les répéter 50 fois dans mon code XHTML.

Je vais créer une class "question" qui permettra de mettre en forme automatiquement toutes les questions de mon texte.

Voici le code XHTML qui va vous permettre de comprendre le truc :

### Code : HTML

```
<h1>Plein de questions</h1>

<p>J'ai créé le Site du Zér0 parce que j'en avais marre de ne pas
trouver de tutoriels corrects sur Internet.</p>

<p class="question">De quand date ce site</p>

<p>Il a été créé aux alentours de 1999</p>

<p class="question">Quel a été le premier nom du site</p>

<p>Il n'a jamais changé de nom, il s'est toujours appelé "Site du
Zér0"</p>

<p class="question">Combien de temps as-tu passé pour trouver un nom
aussi ridicule</p>

<p>Environ 40 secondes. Oui je sais, c'est beaucoup vu le résultat,
mais que voulez-vous ;o)</p>

<p class="question">Quel est l'âge du webmaster</p>

<p>Eh quoi ? Qui ça moi ? Mon âge ne vous intéresse pas allons
allons.<br />Bon allez, une petite exclu, avec ça vous saurez
retrouver mon âge : j'ai commencé la création de ce site à l'âge de
14 ans.</p>
```

[Essayer !](#)

Utilisons maintenant les pseudo-formats suivants :

- **:before** : qui signifie "avant"
- **:after** : qui signifie "après"

On va utiliser une propriété CSS nouvelle et un peu particulière qui nous permettra d'insérer du texte : content  
Regardez comment j'utilise :before et :after sur les paragraphes équipés de la class "question" :

Code : CSS

```
.question:before /* Avant chaque question */
{
    content: "Question : "; /* Commencer par Question : */
}
.question:after /* Après chaque question */
{
    content: " ?"; /* Rajouter un point d'interrogation */
}
```

Essayer !

*content* est donc une propriété CSS vraiment particulière, vu qu'elle permet d'indiquer le texte que l'on veut mettre "Avant" (before) ou "Après" (after). C'est donc assez différent de ce qu'on a fait jusque-là (et je vous rassure, c'est la seule propriété CSS vraiment "à part" qui existe).

Vous n'êtes pas obligés de mettre uniquement la propriété *content* avec un :before ou :after, vous pouvez aussi utiliser tous les styles CSS que vous connaissez. Ces styles s'appliqueront au texte rajouté par le :before ou :after.

Code : CSS

```
.question:before
{
    content: "Question : ";
    font-weight: bold; /* "Question" sera en gras */
    color: blue; /* "Question" sera en bleu */
}
.question:after
{
    content: " ?";
}
```

Essayer !

Comme vous le voyez, la mise en gras et en bleu s'appliquera donc au texte rajouté par le :before.



Si vous appliquez le pseudo-format `:first-letter` à la class `question`, pour mettre en gras par exemple la première lettre, c'est la première lettre de "Question :" (le Q) qui sera mise en gras, et non pas la lettre qui se trouve juste après "Question :". `:first-letter` s'appliquera donc à la première lettre du texte écrit avec le `:before`.

## Une image au lieu du texte ?

Il est aussi possible d'indiquer une image à la place du texte avec le `:before` et le `:after`.

On utilise toujours la propriété *content*, mais au lieu de taper du texte entre guillemets, on va mettre une valeur "url" comme ceci :

## Code : CSS

```
.question:before
{
    content: url("../../images/qst.gif"); /* Mettre une image de
question */
}
.question:after
{
    content: " ?";
}
```

Essayer !



N'oubliez pas que le chemin relatif vers l'image se fait à partir de la position du fichier CSS et non pas du fichier XHTML. Si vous vous trompez dans le chemin, l'image ne s'affichera pas.

Ce que je vous montre là ne sont que des exemples d'utilisation, vous en trouverez certainement une utilité plus appropriée à votre site web 😊

Les pseudo-formats sont un aspect intéressant du CSS bien qu'ils amènent parfois à des confusions.

Si vous avez répondu tout juste au QCM c'est que vous avez tout compris. Si vous avez fait des erreurs, sachez que c'est un peu normal de se planter la première fois. Essayez de faire des tests avec les CSS proposés pour essayer de comprendre "le truc".

En pratique, on voit assez peu sur le web de pseudo-formats autres que ceux concernant les liens. Il est vrai qu'avoir des liens qui changent de couleur quand on pointe dessus par exemple, c'est assez sympa 😊

:first-line et :first-letter fonctionnent bien sur tous les navigateurs (dont Internet Explorer) mais sont assez peu connus et c'est dommage car ils permettent de donner un "petit plus" à la présentation de votre page web sans faire d'efforts.

Enfin, :before et :after sont très peu utilisés (ils ne sont pas toujours compatibles avec tous les navigateurs)

La partie II traitant uniquement du CSS s'arrête là. Eh oui, déjà 😊

Mais il faut reconnaître que ces 4 chapitres ont été riches en nouveautés pour vous 😊

Si on résume ce qu'on a fait jusqu'ici :

- I. La première partie vous a enseigné toutes les bases du XHTML, de quoi faire une première page web avec des liens et des images. La page était certes un peu moche, mais...
- II. on a introduit le CSS dans cette partie II. Désormais, sans rien changer au fichier XHTML, vous avez appris à donner une belle présentation à votre page web par le biais des fichiers .css. Les possibilités sont quasi-infinies, c'est à vous de mélanger les propriétés CSS à votre goût.
- III. Toutefois, on n'a pas encore tout vu sur la création de site web. Il reste des balises XHTML un peu plus complexes, notamment pour créer des tableaux ou des formulaires (mixés à du CSS pour leur donner une belle présentation). Il reste aussi à apprendre à manier les "calques", un passage obligatoire si vous voulez créer le design de votre site web.

En clair, la partie III mélangera XHTML et CSS et vous permettra de maîtriser enfin le design de votre site web.

C'est la dernière ligne droite et il vous reste encore plein de nouvelles choses à découvrir 😊

Quand vous êtes prêts, rendez-vous dans la partie III 😊

## Partie 2 : XHTML & CSS, toujours plus forts !

Non, vous n'avez pas encore tout vu,  
Combinez XHTML et CSS, c'est magique !

---

### Les listes à puces

Alors comme ça, vous voilà dans la partie III de ce tutoriel ? Bravo, on avance bien à ce que je vois 😊

Vous avez encore pas mal de choses à apprendre sur le XHTML et le CSS. Notamment, et c'est important, vous découvrirez dans cette partie comment réaliser le design de votre site web.

La particularité de cette partie III, c'est que désormais nous allons apprendre de nouvelles choses en XHTML et en CSS. Le plan de notre cours pourrait donc se résumer à ceci :

- I. XHTML
- II. CSS
- III. XHTML & CSS, la puissance combinée de ces deux langages

Ah, et tenez justement : vous voyez ce que je viens de faire ci-dessus ? Ca s'appelle une liste à puce, et c'est justement ce que nous allons apprendre à faire dans ce chapitre ! (ouah c'te transition de fou 🤪)

Ce n'est pas bien compliqué à comprendre, et ça fait un bon petit chapitre pour démarrer doucement la partie III qui va se compliquer un peu par la suite.

Car oui, pour ceux qui se poseraient des questions, la partie III est bien la partie la plus "difficile". Mais il n'y a pas de quoi s'en faire, du temps que vous restez attentifs ça ne peut que bien se passer 😊

---

## Différents types de listes (XHTML)

Ce chapitre a le mérite d'être très clair dans sa construction : dans un premier temps nous apprendrons à créer des listes à puces d'un point de vue XHTML, et dans un second temps nous apprendrons à décorer ces listes via de nouvelles propriétés CSS. Tout d'abord, il faut savoir qu'il existe 3 types de listes à puces :

- Les listes non ordonnées
- Les listes ordonnées
- Les listes de définitions (plus rares)

Nous allons voir comment on crée chacune de ces listes à puces. C'est facile, mais soyez attentifs quand même car vous allez avoir besoin des listes à puces dans les chapitres suivants. En effet, contrairement à ce qu'on pourrait croire les listes à puces sont très utilisées : elles servent notamment à créer le menu de votre site web !

### Liste non ordonnée

Une liste non ordonnée ressemble à ceci :

- Fraises
- Framboises
- Cerises

C'est un système qui nous permet de faire une liste d'éléments, sans notion d'ordre (il n'y a pas de "premier" ni de "dernier"). Créer une liste à puce non ordonnée est très simple. Il suffit d'utiliser la balise `<ul>` que l'on referme un peu plus loin avec un `</ul>`.

Commencez donc à taper ceci :

**Code : HTML**

```
<ul></ul>
```

Et maintenant, voilà ce qu'on va faire : on va écrire chacun des éléments de la liste entre 2 balises `<li></li>`. Toutes ces balises doivent se trouver entre `<ul>` et `</ul>`. Vous allez comprendre de suite avec cet exemple :

**Code : HTML**

```
<ul>
  <li>Fraises</li>
  <li>Framboises</li>
  <li>Cerises</li>
</ul>
```

[Essayer !](#)

`<ul></ul>` indique le début d'une liste à puce  
`<li></li>` indique un nouvel élément de la liste à puce.

Vous pouvez mettre autant d'éléments que vous voulez dans la liste à puces, vous n'êtes pas limité à 3 éléments bien entendu



Et voilà, vous savez créer une liste à puce non ordonnée !

C'était dur hein ? 🤔

Pour ceux qui ont besoin de faire des listes complexes, sachez que vous pouvez *imbriquer* des listes à puces (créer



une liste à puce **DANS** une liste à puces). Si vous voulez faire ça, ouvrez une seconde balise `<ul>` à l'intérieur d'un élément `<li></li>`

Si vous fermez les balises dans le bon ordre, il ne devrait pas y avoir de problème 😊

## Liste ordonnée

Déjà que la liste à puces non ordonnée c'était de la rigolade... Eh bien là vous allez vous rendre compte que, décidemment, faire un site web c'est suuuuuper dur 😊

Pour faire une liste ordonnée, on va pas se fouler. On ne change que la balise `<ul></ul>` qui devient cette fois `<ol></ol>`. A l'intérieur, on ne change rien : on utilise toujours des balises `<li></li>`.



L'ordre dans lequel vous mettez les éléments de la liste est important. Le premier `<li></li>` sera l'élément n°1, le second sera le n°2 etc...

Comme c'est particulièrement intuitif, je vous laisse admirer la simplicité de cet exemple :

Code : HTML

```
<ol>
  <li>Je me lève</li>
  <li>Je mange et je bois</li>
  <li>Je retourne me coucher</li>
</ol>
```

[Essayer !](#)

Par rapport à l'exemple précédent, tout ce qu'on a eu à changer donc c'est la balise `<ol>`

## Liste de définitions

Là par contre, c'est différent des exemples précédents. C'est "un peu" plus difficile (mais alors franchement juste un peu ;)). Le principe de base est le même. On indique le début et la fin de la liste à l'aide d'une balise. Ici il ne s'agit pas de `<ul>`, ni de `<ol>` mais de... `<dl>` (c'est l'abréviation de "Definition List")

Commençons donc par taper ceci :

Code : HTML

```
<dl>

</dl>
```

Bien, maintenant la petite différence c'est qu'on n'utilise plus la balise `<li>` pour indiquer les différents éléments de la liste. En effet, dans le cas d'une liste de définitions, il y a 2 types d'éléments différents :

- Les mots
- Et leur définition

Les mots se trouvent entre `<dt></dt>`, et les définitions entre `<dd></dd>`.

Ce qu'il faut bien comprendre, c'est qu'on doit mettre d'abord le mot (`<dt>`), ensuite sa définition (`<dd>`). Si vous voulez mettre une seconde définition, il faut recommencer : un *dt* et ensuite un *dd*.

Regardez bien cet exemple :

#### Code : HTML

```
<dl>
  <dt>Chat</dt>
  <dd>Animal à 4 pattes qui fait "Miaou !"</dd>
  <dt>Chien</dt>
  <dd>Animal à 4 pattes qui fait "Ouaf Ouaf !"</dd>
  <dt>Prof de maths</dt>
  <dd>Extraterrestre venu d'une autre planète qui enseigne des
choses que personne ne comprend</dd>
</dl>
```

[Essayer !](#)

Comme vous pouvez le voir, les définitions sont mises légèrement en retrait par rapport aux mots. C'est pas très joli pour le moment, mais tout le monde sait qu'avec un petit coup de CSS on peut faire en sorte que les mots soient écrits en gras et en bleu et que les définitions soient clignotantes sur fond rouge. Si vous ne trouvez pas tout seul comment faire ça en CSS, c'est que vous êtes passés trop vite sur la partie II du tutoriel 😊

Les éléments de la liste à puce fonctionnent donc ici par paire. Une fois qu'on a compris qu'il faut taper d'abord le mot, puis sa définition, bah on sait se servir des listes de définitions 😊

Tenez, si c'est pas beau ça, on a déjà fini de voir la partie "XHTML" des listes à puces ! 😊

Que diriez-vous de pimenter tout ça avec un peu de CSS, mmh ? Ca devrait avoir encore meilleur goût ensuite 😊

## Décorez vos listes (CSS)

Non non, je ne vais pas vous apprendre à faire en sorte que vos définitions soient clignotantes sur fond rouge 🤪 Ca, vous savez déjà le faire :

Code : CSS

```
dd
{
background-color:red;
/* ... */
}
```

Bref ça c'est du vu et revu, en plus je viens de vous mettre sur la piste 😊

Ce que nous allons voir maintenant ce sont des propriétés CSS spécialisées dans la présentation des listes à puces. Il en existe 3. Nous allons étudier le fonctionnement de chacune d'elles.

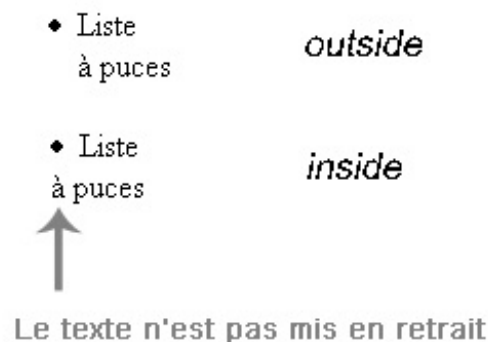
### Retrait des listes

La première des propriétés que je veux vous montrer est très facile à utiliser. Elle permet d'indiquer si on veut que la liste soit mise en retrait ou non.

Cette propriété s'appelle *list-style-position*, et peut prendre 2 valeurs :

- inside : la liste n'est pas mise en retrait.
- outside : la liste est mise en retrait (par défaut).

Ce schéma qui vous explique la différence entre inside et outside :



Voici le CSS qui nous permettra de tester le retrait :

Code : CSS

```
.pas_de_retrait
{
list-style-position: inside;
}

.retrait
{
list-style-position: outside;
```



```
}
```

Et un code XHTML banal de liste à puces :

#### Code : HTML

```
<p>Voici une liste avec retrait (par défaut) :</p>

<ul class="retrait">
  <li>Liste<br />à puces</li>
  <li>Ligne 1<br />Ligne 2</li>
  <li>Vous pouvez voir<br />que le texte est décalé sur la
droite</li>
</ul>

<p>Voici une liste sans retrait :</p>

<ul class="pas_de_retrait">
  <li>Liste<br />à puces</li>
  <li>Ligne 1<br />Ligne 2</li>
  <li>Vous pouvez voir<br />que le texte n'est pas décalé sur la
droite</li>
</ul>
```

[Essayer !](#)

Pour bien voir la différence au niveau du retrait, il faut que les éléments de la liste comportent au moins 2 lignes (c'est pour ça que j'ai mis un `<br />`). En pratique, on ne désactive pas le retrait des listes à puces, sauf quand on se sert des listes pour créer le menu de son site comme on le verra plus tard.

## Représentation de la puce

Nettement plus intéressante, la propriété *list-style-type* vous permet de changer l'apparence de la puce. En effet, vous n'êtes pas obligés d'avoir une puce sous forme de rond noir, pas plus que vous n'êtes obligés d'avoir des chiffres pour les listes numérotées (on peut utiliser des lettres : a, b, c, d...)

La propriété *list-style-type* peut prendre de nombreuses valeurs. Certaines d'entre elles concernent uniquement les listes non ordonnées, d'autre concernent uniquement les listes ordonnées :

- Pour les listes non ordonnées (`<ul>`) :
  - [disc](#) : un disque noir (par défaut).
  - [circle](#) : un cercle.
  - [square](#) : un carré.
  - [none](#) : aucune puce ne sera utilisée.
- Pour les listes ordonnées (`<ol>`), le choix est vaste :
  - [decimal](#) : des nombres 1, 2, 3, 4, 5... (par défaut)
  - [decimal-leading-zero](#) : des nombres commençant par zéro (01, 02, 03, 04, 05...).
  - [upper-roman](#) : numérotation romaine, en majuscules (I, II, III, IV, V...)
  - [lower-roman](#) : numérotation romaine, en minuscules (i, ii, iii, iv, v...)
  - [upper-alpha](#) : numérotation alphabétique, en majuscules (A, B, C, D, E...)
  - [lower-alpha](#) : numérotation alphabétique, en minuscules (a, b, c, d, e...)
  - [lower-greek](#) : numérotation grecque.

Il existe en réalité d'autres valeurs possibles pour les listes ordonnées, comme la numérotation arménienne, géorgienne, hébraïque, japonaise etc... Mais je vous en fais grâce (et je ne pense pas que vous m'en voudrez !), ce que je vous propose là devrait amplement vous suffire 😊

Et si on testait quelques-unes de ces listes à puces modifiées ? Voici le code XHTML qui va nous servir de test (un peu gros, mais ne prenez pas peur c'est un peu répétitif c'est tout 😊)

#### Code : HTML

```
<h2>Quelques listes non ordonnées</h2>

<p>Voici une liste à puce non ordonnée sans modification (=
<em>disc</em>) :</p>
<ul>
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>

<p>Avec <em>circle</em> :</p>
<ul class="cercle">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>

<p>Avec <em>square</em> :</p>
<ul class="carre">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>

<p>Avec <em>none</em> (sans puces) :</p>
<ul class="rien">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>

<h2>Quelques listes ordonnées</h2>

<p>Liste à puces ordonnée sans modification (= <em>decimal</em>)</p>
<ol>
  <li>Un</li>
  <li>Deux</li>
  <li>Trois</li>
  <li>Quatre</li>
</ol>

<p>Avec <em>decimal-leading-zero</em> (<strong>Ne fonctionne pas
sous IE)</strong>) :</p>
<ol class="commence_a_zero">
  <li>Un</li>
  <li>Deux</li>
  <li>Trois</li>
  <li>Quatre</li>
</ol>

<p>Avec <em>lower-alpha</em> :</p>
<ol class="lettres_minuscules">
  <li>Un</li>
```

```

<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>

<p>Avec <em>upper-roman</em> :</p>
<ol class="chiffres_romains">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>

<p>Avec <em>lower-greek</em> (<strong>Ne fonctionne pas sous
IE</strong>) :</p>
<ol class="lettres_grecques">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>

```

... Et le CSS qui va avec :

#### Code : CSS

```

/* Listes à puces non ordonnées */

.cercle
{
    list-style-type: circle;
}
.carre
{
    list-style-type: square;
}
.rien
{
    list-style-type: none;
}

/* Listes à puces ordonnées */

.commence_a_zero
{
    list-style-type: decimal-leading-zero;
}
.lettres_minuscules
{
    list-style-type: lower-alpha;
}
.chiffres_romains
{
    list-style-type: upper-roman;
}
.lettres_grecques
{
    list-style-type: lower-greek;
}

/* Quelques styles supplémentaires juste pour la présentation
En plus ça vous fait quelques rappels ;) */

h2

```

```
...
{
    text-indent: 20px;
    font-family: Arial, Verdana, "Times New Roman", serif;
}
em
{
    background-color: yellow;
}
strong
{
    color: red;
}
```

[Essayer !](#)

Je n'ai pas mis tous les styles possibles pour ne pas trop alourdir le code source, mais je pense que vous êtes assez grands pour tester les autres styles tous seuls non ? 🤖

## Changer l'image de la puce

Si aucune des représentations de puces ci-dessus ne vous convient, pourquoi ne pas créer la vôtre ? C'est justement ce que vous propose la propriété *list-style-image*, qui vous permet d'utiliser n'importe quelle image à la place d'une puce.

Vous devez lui indiquer comme valeur "url" suivi de l'adresse de l'image à utiliser. Ça fonctionne exactement comme la propriété *background* (qui permet d'indiquer une image de fond, si vous vous souvenez bien).

On va par exemple utiliser une petite image de dossier à la place de la puce, vous allez voir que ça fait beaucoup plus joli !



### Code : CSS

```
ul /* Ma liste aura des puces en forme de dossiers */
{
    list-style-image: url("dossier.png");
}

/* Juste pour améliorer la présentation : */
a
{
    color: blue;
    text-decoration: none;
}

a:hover
{
    text-decoration: underline;
}
```

[Essayer !](#)

Sympatoche non ? 😊



L'image de puce peut être de n'importe quel type (PNG, GIF ou JPEG).



Attention cependant, l'image ne doit pas être trop grande sinon elle sera coupée. Je vous conseille d'utiliser une taille d'environ 15x15 pixels.

---

Avec ça, on vient de faire le tour de tout ce qu'on pouvait voir et savoir sur les listes à puces. Certes les noms des balises ne sont pas très clairs (ul, ol, li, dl, dd, dt), mais au moins ils sont courts et il n'y en a pas des centaines non plus ! Vous verrez que vous les retiendrez rapidement avec l'habitude... 😊

Eh, ce n'est pas le moment de vous endormir ! Les chapitres suivants sont super super importants. Si vous voulez réussir votre site et apprendre à en créer le design, soyez très attentifs à partir de maintenant ! En effet, vous allez découvrir pas mal de nouvelles choses, surtout sur le CSS 😊

---



## Mise en boîte (partie 1/2)

Vous êtes en forme ? Tant mieux, c'est maintenant que tout se joue. Eh oui, jusqu'ici vous pouviez au mieux suivre mes petits exemples et inventer les vôtres, mais pas encore créer votre site web de toutes pièces.

Les 2 chapitres que vous allez lire vont être un véritable tournant dans ce cours : grâce à eux, vous saurez créer pour de bon votre site web, avec son design et tout et tout ! 😊 Et que ceux qui auraient peur de se retrouver tout seuls se rassurent : je ne vous abandonnerai pas. J'ai prévu de vous montrer toute la marche à suivre dans un TP 😊

Le titre de ce chapitre est assez évocateur, nous allons nous préparer à "mettre en boîte" notre site, ce qui veut dire qu'il sera bientôt prêt !

Et, comme j'ai repéré parmi vous quelques étourdis qui n'écoutaient pas quand j'ai parlé de balises *inline* / *block*, je pense qu'un petit rappel ne ferait de mal à personne 😊

---

## Block et Inline c'est pas pareil !

Tout au long des chapitres précédents, vous avez entendu à plusieurs reprises les mots "inline" et "blocks". J'ai insisté sur le fait que c'était important à comprendre, car tout se jouerait là-dessus par la suite...

Eh ben devinez quoi, c'est maintenant que tout se joue sur cette différence. Si vous n'êtes pas capables de faire la différence entre une balise inline et une balise block, vous allez être facilement largués. Vous ne m'en voudrez pas, donc, d'avoir pris la précaution de vous faire ce rappel 🤔



Bon, de quoi parle-t-on ici ?

On parle des balises. On peut classer les balises XHTML dans une des 2 catégories suivantes : **inline et block** (en français : "En ligne" et "Bloc").

Ce que j'attends de vous, c'est que vous sachiez reconnaître si une balise est inline ou block.

Par exemple, `<p></p>` ?

Oui, c'est une balise **block**.

Et `<a></a>` ?

Ca c'est une balise **inline**.



Mais comment je reconnais une balise inline d'une balise block ?

C'est en fait assez facile :

- **block** : une balise de type "block" sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. Votre page web sera en fait constituée d'une série de blocks à la suite les uns des autres. Mais vous verrez qu'en plus, il est possible de mettre un block à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
- **inline** : une balise de type "inline" se trouve obligatoirement à l'intérieur d'une balise "block". Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise "en ligne").

Est-ce que `<p></p>` crée un retour à la ligne ?

Oui, c'est donc une balise de type block.

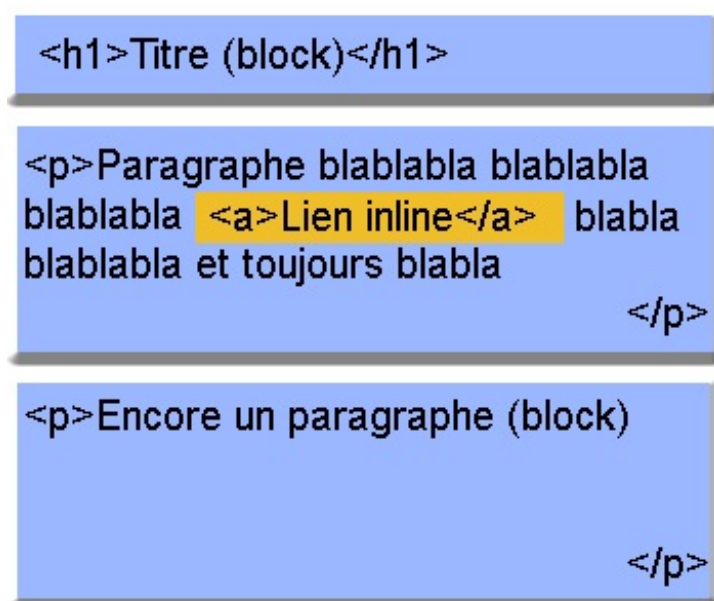
Est-ce que `<a></a>` crée un retour à la ligne ?

Non, le texte contenu dans cette balise s'écrit à la suite du texte précédent, il reste sur la même ligne. C'est donc une balise de type inline.

Fastoche, isn't it ? 🤔

Pour les sceptiques, voici un petit schéma que je vous ai concocté (c'est donc moche à souhait, mais c'est pas le problème 😊)

C'est grâce à ce schéma que je visualise les choses dans ma tête, donc si vous voulez rentrer dans ma tête, regardez-le bien :



Sur fond bleu, vous avez tout ce qui est de type block.  
Sur fond jaune, vous avez tout ce qui est de type inline.

Comme vous pouvez le voir, les blocks sont les uns en-dessous des autres. J'aurais pu imbriquer des blocks mais je ne veux pas trop compliquer les choses pour le moment 😊

La balise inline `<a></a>` se trouve à l'intérieur d'une balise block (je vous avais dit que c'était obligatoire), et le texte vient s'insérer sur la même ligne.

## Quelques exemples

Afin de mieux vous aider à assimiler quelles balises sont inline et quelles balises sont block, voici un petit tableau listant quelques balises courantes.

Balises blocks	Balises inline
<code>&lt;p&gt;</code>	<code>&lt;em&gt;</code>
<code>&lt;blockquote&gt;</code>	<code>&lt;strong&gt;</code>
<code>&lt;h1&gt;</code>	<code>&lt;q&gt;</code>
<code>&lt;h2&gt;</code>	<code>&lt;a&gt;</code>
<code>&lt;ul&gt;</code>	<code>&lt;sup&gt;</code>
<code>&lt;ol&gt;</code>	<code>&lt;img /&gt;</code>
...	...

Ce tableau n'est pas complet, loin de là. Si vous voulez avoir la liste complète des balises qui existent, et savoir si elles sont inline ou block, reportez-vous à l'annexe donnant la liste de toutes les balises XHTML.

## Les balises universelles

Vous les connaissez déjà car je vous les ai présenté il y a quelques chapitres. Ce sont des balises qui n'ont aucun sens particulier (contrairement à `<p>` qui veut dire "paragraphe", `<strong>` "important" etc...).

Le principal intérêt de ces balises c'est de pouvoir appliquer une class (ou un id) pour le css quand aucune autre balise ne convient.

Il existe 2 balises génériques, et comme par hasard la seule différence entre les deux c'est que l'une d'elle est inline, l'autre est block :

- `<span></span>` (*inline*) : on l'a déjà utilisée, si vous vous souvenez bien. Je m'en suis servi pour vous montrer quelques exemples de propriétés CSS au milieu d'une ligne. Vous souvenez-vous du code ci-dessous ?

**Code : HTML**

```
<p>Comme l'a dit <span class="nom">Neil Armstrong</span> un  
certain 20 juillet 1969...</p>
```

S'il existait une balise XHTML `<nom></nom>`, je l'aurais utilisée. Mais comme ce n'était pas le cas, j'ai dû me rabattre sur un `<span>` et lui appliquer une class que j'ai inventée ("nom").

- `<div></div>` (*block*) : c'est aussi une balise qui n'a aucun sens, comme span, sauf que celle-là est de type block. On va pas mal s'en servir dans les chapitres qui suivent, notamment pour créer le design de votre site. Vous allez voir qu'un design, en fait, c'est une série de blocks qu'on dispose comme on veut.

## Respectez la sémantique !

Les balises universelles sont "pratiques" dans certains cas, certes, mais attention à ne pas en abuser. Je tiens à vous avertir de suite : beaucoup de webmasters mettent des `<div>` et des `<span>` trop souvent, et oublient que d'autres balises plus adaptées existent.

Voici 2 exemples :

- **Exemple d'un span inutile :**

Je ne devrais jamais voir dans un de vos codes :

```
<span class="important">
```

... alors qu'il existe la balise `<strong>` qui sert à ça !

- **Exemple d'un div inutile :**

De la même manière, ceci est complètement absurde :

```
<div class="titre">
```

... puisqu'il existe des balises faites spécialement pour les titres (`<h1><h2><h3>...`)

Oui, vous allez me dire qu'au final le résultat (visuel) est le même. Je suis tout à fait d'accord.

Mais, tandis que les balises `span` et `div` ne signifient rien de particulier, les balises `strong` et `h1` veulent dire quelque chose !

`Strong` signifie "important" et `h1` signifie "titre principal".

En XHTML, on vous demande d'utiliser tant que possible des balises qui ont un sens : on appelle ça respecter la sémantique.

L'avantage c'est que si vous respectez ceci, votre code aura une certaine "logique". Lorsque le robot de Google viendra

référencer votre site, il "comprendra" le sens des balises et cela pourra améliorer votre position dans le moteur de recherche 😊



## La taille

Nous allons pour le moment travailler uniquement sur des balises de type "block".

Pour commencer, nous nous intéressons à la taille des blocks. Contrairement à un inline, un block a des dimensions précises. Il a une largeur et une hauteur.

Ce qui fait, Ô surprise, qu'on dispose de 2 propriétés CSS :

- [width](#) : c'est la largeur du block. A exprimer en pixels (px) ou en pourcentage (%).
- [height](#) : c'est la hauteur du block. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

Par défaut, et c'est très important à savoir, un block prend 100% de la largeur disponible. Regardez par exemple ce code XHTML qui n'utilise aucune propriété CSS :

### Code : HTML

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam lacinia leo ac nunc. Nulla aliquet. Nullam quam leo, volutpat vitae, venenatis eget, blandit eu, augue. Proin ac libero nec magna laoreet tempus. Nullam felis. Nam et libero. Sed velit dui, tempus in, nonummy venenatis, accumsan id, justo. Ut ullamcorper. Cras sem diam, vulputate nec, tempor at, pretium at, magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur sapien pede, malesuada ac, ultricies in, dignissim eget, mauris. Nullam varius diam ac ligula. Morbi mattis posuere odio. Ut mattis risus ac erat. Nam volutpat, nisl vitae venenatis mollis, ante erat tincidunt purus, nec ornare felis tellus sed purus. Phasellus orci.</p>
```

[Essayer !](#)

Rien de nouveau sur cet exemple. Cependant, il est très important de noter que le paragraphe (comme toutes les balises block) prend 100% de la largeur disponible par défaut.

Maintenant, pimenterons d'un peu de CSS pour modifier la largeur des paragraphes. Le CSS suivant dit : "Je veux que tous mes paragraphes aient une largeur de 50%".

### Code : CSS

```
p{  
  width: 50%;  
  text-align: justify; /* Texte justifié pour mieux voir la largeur du block */}
```

Testons l'effet sur le paragraphe de l'exemple précédent :

[Essayer !](#)

Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution du visiteur. Toutefois, il se peut que vous ayez besoin de créer des blocks ayant une dimension précise en pixels :

### Code : CSS

```
p{  
  width: 250px;
```

```
text-align: justify;
}
```

[Essayer !](#)

## Minimum et maximum

Il vous sera aussi très pratique de demander à ce qu'un block ait une taille minimale ou maximale. C'est là qu'entrent en jeu les 4 propriétés CSS suivantes :

- `min-width` : largeur minimale
- `min-height` : hauteur minimale
- `max-width` : largeur maximale
- `max-height` : hauteur maximale

Attention cependant, ces propriétés ne fonctionnent pas sous IE 6 et fonctionnent partiellement sous IE 7. Evitez de les utiliser tant qu'IE 6 est encore assez répandu.

## "Couper" un block avec un overflow

Supposons que vous ayez un long paragraphe et que vous vouliez (pour une raison qui ne regarde que vous) qu'il fasse 250px de large et 100px de haut.

Code : CSS

```
p{
  width: 250px;
  height: 100px;
  text-align: justify;
}
```

[Essayer !](#)

Attends, je croyais que le paragraphe ferait 100px de haut moi !?

Oui, mais si le texte n'a pas assez de place, le block va s'agrandir de manière à ce que tout soit visible. Si vous voulez "couper" votre paragraphe pour qu'il soit d'une dimension exacte de 250x100, vous allez devoir utiliser la propriété CSS *overflow*.

Les valeurs que peut prendre *overflow* sont les suivantes :

- visible (par défaut) : si le texte dépasse les limites de taille, le block est agrandi de manière à ce que tout soit visible. C'est ce que nous venons de constater.
- hidden : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra 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 qu'on puisse voir tout le texte. C'est un peu comme un cadre à l'intérieur de la page.
- auto : mode "pilote automatique" :p. En gros, c'est le navigateur qui décide ou pas de mettre des barres de défilement (il en mettra si c'est nécessaire).

Avec ce CSS, nous allons pouvoir tester *overflow* :

#### Code : CSS

```
p /* Tous les paragraphes auront ces propriétés CSS */{  
    width: 250px;  
    height: 100px;  
    text-align: justify;  
}/* Et chacun des paragraphes aura en plus une valeur d'overflow  
différente */  
.coupe{  
    overflow: hidden;  
}  
.barres_defilement{  
    overflow: scroll;  
}  
.auto{  
    overflow: auto;  
}
```

[Essayer !](#)

En hidden, on ne peut pas voir tout le texte.

En scroll, le navigateur a mis des barres de défilement verticales et horizontales (alors qu'on n'a pas besoin de la barre horizontale).

En auto, le navigateur s'est dit "Bah tiens, j'ai pas besoin de la barre horizontale, je vais l'enlever".

En pratique, vous vous en doutez, on se sert plutôt de *overflow: auto*;

L'overflow vous sera très utile si vous avez besoin de créer des "cadres" à l'intérieur de votre page web.



Il existe une ancienne balise HTML, `<iframe>`, qui donne à peu près le même résultat. Toutefois, cette balise n'existe plus en XHTML (elle est dépassée). Il vous faudra donc utiliser un *overflow* à la place.

## Les bordures

Le CSS vous offre un large choix de bordures pour décorer votre page. Tellement large qu'il existe pas mal de propriétés CSS, et que si je vous en faisais la liste complète maintenant vous sauriez plus trop où donner de la tête.

Alors, pour aller plus vite à l'essentiel, j'ai décidé de vous montrer uniquement la méga-propriété qui regroupe tout. Vous vous souvenez d'une méga-propriété ? Oui, on en a vu une : background, c'était pour le fond de votre page web. Retournez voir le chapitre correspondant si vous avez besoin de vous rafraîchir la mémoire sur les méga-propriétés.

Là, on va directement étudier la méga-propriété, car en pratique c'est ce qu'on fait le plus souvent.

La méga-propriété en question, c'est border. C'est elle qui permet d'indiquer quelle bordure on applique au block.

Comme toute méga-propriété qui se respecte, vous pouvez lui mettre plusieurs valeurs (ici, il y a 3 valeurs à utiliser). Vous n'êtes pas obligés de mettre toutes les valeurs (vous pouvez vous contenter de 2, ou d'une seule), et l'ordre dans lequel vous mettez ces valeurs n'a aucune importance.

N'oubliez pas cela, car beaucoup de gens croient qu'il y a un ordre à respecter et se prennent la tête, alors qu'on s'en fout totalement 😊

Comme je viens de vous le dire, pour border on peut utiliser jusqu'à 3 valeurs pour modifier l'apparence de la bordure :

- **La largeur** : indiquez la largeur de votre bordure. Mettez une valeur en pixels (comme 2px), ou utilisez un des mots-clé suivants :
  - **thin** : bordure fine
  - **medium** : bordure moyenne
  - **thick** : bordure épaisse

C'est le navigateur qui choisit le nombre de pixels qui correspond à thin, medium et thick. En général, c'est à peu près pareil sur tous les navigateurs.

Personnellement, j'ai plutôt tendance à utiliser des valeurs en pixels au lieu de "thin" et compagnie... Mais c'est une question de goût après tout 😊

- **La couleur** : c'est la couleur de votre bordure. Utilisez, comme on l'a appris, soit un nom de couleur ("black", "red"...), soit une valeur hexadécimale (#FF0000), soit une valeur rgb (rgb(198, 212, 37))
- **Le type de bordure** : là, vous avez pas mal de choix. Votre 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** : effet 3D.
  - **inset** : autre effet 3D (on a l'impression que le block forme un creux).
  - **outset** : encore un autre effet 3D (on a l'impression que le block est surélevé).

Testons maintenant un peu tout ça 😊

### Code : CSS

```
p
{
    width: 300px; /* Tous les paragraphes font 300px */
    text-align: justify; /* Tous les paragraphes sont justifiés */
}

.rien
{
    border: none;
}

.solid
{
    border: 1px solid black;
}
```

```
border: 2px solid black;
}
.dotted
{
border: 2px dotted green;
}
.dashed
{
border: 2px dashed red;
}
.double
{
border: 4px double maroon;
}
.groove
{
border: 4px groove teal;
}
.ridge
{
border: 4px ridge fuchsia;
}
.inset
{
border: 3px inset black;
}
.outset
{
border: 3px outset black;
}
```

[Essayer !](#)

Il y a matière à s'amuser, comme vous pouvez le constater 😊

## En haut, à droite, à gauche, en bas...

Qui a dit que vous étiez obligés de mettre la même bordure aux 4 côtés de votre block ?

Taratata, si vous voulez mettre des bordures différentes en fonction du côté (haut, bas, gauche ou droite), vous pouvez le faire sans problème. Dans ce cas, vous devrez utiliser ces 4 propriétés :

- [border-top](#) : bordure en haut.
- [border-bottom](#) : bordure en bas.
- [border-left](#) : bordure à gauche.
- [border-right](#) : bordure à droite.

Ce sont aussi des méga-propriétés, elles fonctionnent comme border mais ne s'appliquent donc qu'à un seul côté. Voici quelques tests de bordures.

Code : CSS

```
p
{
width: 350px;
}
h2
{
```

```
border-bottom: 2px solid black;
}

.haut_bas
{
border-top: 1px dashed red;
border-bottom: 1px dashed red;
}

.points
{
border-top: 3px dotted blue;
border-left: 2px solid green;
border-right: 2px solid green;
border-bottom: 3px dotted blue;
}

.tres_solide
{
border-left: 6px solid black;
border-right: 6px solid gray;
}
```

[Essayer !](#)

En particulier, vous remarquerez que j'ai appliqué une bordure en bas du titre. Ça n'a rien à voir avec un soulignement. Je vous l'ai dit plus haut : par défaut, un block (comme le titre) prend 100% de la largeur disponible. En l'occurrence, le block prend la largeur de toute la fenêtre. Le fait de mettre une bordure en bas du titre révèle qu'il prend effectivement toute la largeur.

C'est une technique décorative assez "classe" que d'utiliser un border-bottom sur un titre. Le titre crée ainsi une ligne de séparation, ce qui rend votre page plus structurée. En plus, c'est facile à utiliser.

Après, on aime ou on n'aime pas...

... Moi j'aime 🤔



Au fait, si vous voulez appliquer la même bordure aux 4 côtés de votre block, ne vous fatiguez pas à utiliser ces propriétés. Utilisez plutôt border qui s'applique à tous les 4 côtés simultanément.

## Ca marche aussi sur des balises inline !

Je vous ai dit que toutes les propriétés CSS que nous voyons dans ces chapitres s'appliquent le plus souvent aux blocks... Mais pas toujours. Par exemple, on peut utiliser border sur des balises inline (même si on le fait un peu plus rarement).

Il y a une balise inline sur laquelle on utilise fréquemment border : c'est <img /> (pour les images). En effet, vous vous souvenez que si vous transformez une image en lien, elle se voit dotée d'une immonde bordure bleue :

### Code : HTML

```
<p>
  Vous souhaitez vous rendre vers un endroit magique ? Rien de plus
  simple, cliquez sur l'image ci-dessous :<br />
  <a href="http://www.siteduzero.com"></a>
</p>
```

[Essayer !](#)

Maintenant, avec la propriété `border` du CSS vous allez pouvoir éviter ce petit désagrément :

**Code : CSS**

```
a img /* Toutes les images contenues dans un lien */
{
    border: none; /* Pas de bordure */
}
```

[Essayer !](#)

Le CSS, c'est magique 😊

Pour ceux qui auraient (déjà) oublié, j'ai utilisé l'imbrication. Si je tape "a img", cela signifie "Toutes les balises <img> contenues dans un lien <a>"... Et si vous réfléchissez bien, c'est exactement ce qu'on veut : enlever la bordure qui apparaît autour des images qui sont contenues dans des liens.



A la place de "border:none", j'aurais pu utiliser "border:0px" ou encore "border:0" (ça marche aussi). D'ailleurs, la plupart des webmasters ont tendance à mettre "border:0" tout simplement parce que c'est le plus court. N'oubliez pas : les webmasters sont des humains comme les autres. Moins ils en font, mieux ils se portent 😊

## Les marges

Ah, les joies des marges 😊

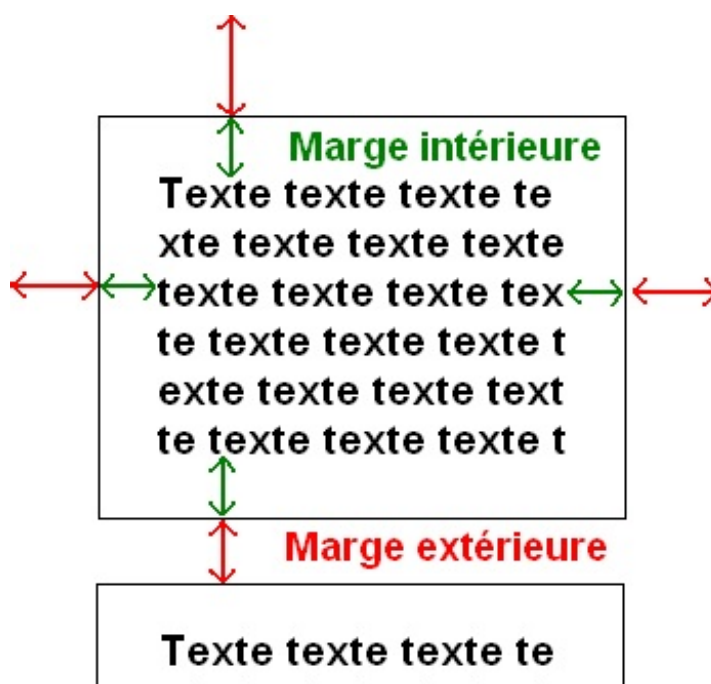
C'est pas le moment de vous endormir les amis 😊

Bien comprendre les marges, c'est vital. Si vous ne savez pas précisément comment fonctionnent les marges des blocks, vous serez bien embêtés lors de la création de votre design !

Tous les blocks possèdent des marges. Ce qui est important, c'est de savoir qu'il existe **2 types de marges** :

- Les marges intérieures
- Les marges extérieures

Regardez bien ce schéma :



Sur ce block, j'ai mis une bordure pour qu'on repère mieux ses bords.

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

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

En CSS, on peut modifier la taille des marges avec les 2 propriétés suivantes :

- padding : indique la taille de la marge intérieure. A exprimer en général en pixels (px).
- margin : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.

Pour bien voir les marges, prenons 2 paragraphes auxquels je mets simplement une petite bordure :

Code : CSS

```
P
{
  width: 350px;
  border: 1px solid black;
  text-align: justify;
}
```



Essayer !

Comme vous pouvez le constater, il n'y a par défaut pas de marge intérieure (padding). En revanche, il y a une marge extérieure (margin). C'est cette marge qui fait que 2 paragraphes ne sont pas collés et qu'on a l'impression de "sauter une ligne".



Les marges par défaut ne sont pas les mêmes pour toutes les balises block. Essayez d'appliquer ce CSS à des balises `<div>` qui contiennent du texte par exemple, vous verrez que là il n'y a par défaut ni marge intérieure, ni marge extérieure !

Supposons que je veuille rajouter une marge intérieure de 12px aux paragraphes :

Code : CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px; /* Marge intérieure de 12px */
}
```

Essayer !

Maintenant, je veux que mes paragraphes soient plus espacés entre eux. Je rajoute la propriété margin pour demander à ce qu'il y ait 50px de marge entre 2 paragraphes :

Code : CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin: 50px; /* Marge extérieure de 50px */
}
```

Essayer !



Mais ??? Une marge s'est mise à gauche aussi !

Bah oui, *margin* (comme padding d'ailleurs) s'applique aux 4 côtés du block.

Si vous voulez indiquer une marge en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises... Et vous allez voir que les créateurs du CSS se sont pas foulés, ça fonctionne comme pour border

**En haut, à droite, à gauche, en bas... Et on recommence !**

Ce qui serait bien, ce serait que vous reteniez les traductions suivantes en anglais :

- top : haut

- bottom : bas
- left : gauche
- right : droite

Comme ça, vous pouvez retrouver toutes les propriétés de tête.

Je vais quand même vous faire la liste des propriétés pour margin et padding, histoire que vous soyez sûrs que vous avez compris le principe.

Voici la liste pour margin :

- [margin-top](#) : marge extérieure en haut.
- [margin-bottom](#) : marge extérieure en bas.
- [margin-left](#) : marge extérieure à gauche.
- [margin-right](#) : marge extérieure à droite

Et la liste pour padding :

- [padding-top](#) : marge intérieure en haut.
- [padding-bottom](#) : marge intérieure en bas.
- [padding-left](#) : marge intérieure à gauche.
- [padding-right](#) : marge intérieure à droite.

Dites, c'est moi ou j'ai l'impression de me répéter ? 🤔

Bon, pour tester on va utiliser margin-bottom pour indiquer un espace en bas de chaque paragraphe, ce qui nous évitera d'avoir une marge à gauche comme tout à l'heure.

Et aussi, pour le fun, je vais rajouter une marge à gauche pour le titre (h1) afin qu'il soit un peu décalé.

#### Code : CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 50px;
}

h1
{
    margin-left: 30px;
}
```

[Essayer !](#)



Juste une question... Quelle est la différence entre un *margin-left* et un *text-indent* ?

Bonne question ça 😊

Vous vous souvenez que, pour décaler un peu le titre, on avait utilisé dans un chapitre de la partie II la propriété *text-indent*. Ici, *margin-left* semble donner le même résultat...

Visuellement le résultat est le même, certes. Si vous mettez une bordure, vous verrez une petite différence : la bordure de gauche est décalée avec un *margin-left*, alors que ce n'est pas le cas avec *text-indent*.

En pratique, je vous conseille de n'utiliser le *text-indent* que sur des paragraphes, pour décaler la première ligne. Le reste du temps, jouez avec les *margin* et les *padding* pour placer comme bon vous semble vos blocks.

## Centrer des blocks

Il est tout à fait possible de centrer des blocks, c'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

**Toutefois**, et c'est très important, cela ne fonctionne que si vous avez indiqué une largeur précise (*width*) au block. Si vous n'avez pas indiqué de largeur, le block ne sera pas centré !

Prenons le cas de nos petits paragraphes. On leur a déjà donné une largeur précise ; maintenant si on veut les centrer sur notre écran nous allons mettre la valeur "auto" à la propriété *margin*, comme ceci :

Code : CSS

```
p
{
    width: 350px; /* On a indiqué une largeur (obligatoire) */
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin: auto; /* On peut donc demander à ce que le block soit
centré avec "auto" */
    margin-bottom: 20px;
}
```

Essayer !

Ainsi, le navigateur centre automatiquement nos paragraphes. N'oubliez pas, je le dis et je le répète, que le *margin:auto* ne peut fonctionner que si vous avez indiqué une largeur précise pour votre block !

Toutes ces notions théoriques sur les blocks sont un peu barbant, je peux le comprendre.

Néanmoins, c'est un passage obligé et (bonne nouvelle) c'est la dernière barrière avant la création du design de votre site. En gros, une fois que vous aurez compris ça, vous pourrez enfin voler de vos propres ailes.

Bref, il est plus que jamais nécessaire d'être attentif. D'autant plus que ce n'est pas terminé : il vous reste à lire la seconde partie de ce chapitre.

Si j'ai un bon conseil à vous donner : ne vous pressez pas inutilement. Prenez le temps d'être certain d'avoir tout compris. Et surtout...

Faites des tests

Faites des tests

Faites des tests

Faites des tests

Faites des tests

Faites des tests

Faites des tests

Ca commence à rentrer là ? 😊

En effet, vous ne pouvez pas apprendre convenablement si vous ne vous exercez pas. Prenez des exemples de CSS de ce chapitre, et essayez de les modifier. C'est en pratiquant que les choses rentrent dans la tête, et pas autrement 😊

## Mise en boîte (partie 2/2)

Dans la première partie de ce chapitre "Mise en boîte", nous avons fait le tour des propriétés CSS permettant de modifier

l'apparence et la taille des blocks. Pour résumer rapidement, on a vu :

- Comment modifier la taille des blocks
- Comment modifier la bordure des blocks
- Comment modifier la marge des blocks

Dans cette seconde partie, nous allons apprendre à les positionner. Bah oui, il va bien falloir apprendre à dire : "Je veux que mon menu soit à gauche, que mon contenu soit à droite, qu'il y ait le logo de mon site en haut à droite à 12 pixels du bord droit" etc...

Autant que les choses soient claires : le positionnement en CSS, c'est pas franchement évident... Il y a la théorie, et il y a la pratique. Dans ce chapitre, nous verrons seulement la théorie (c'est un passage obligé), et dans le chapitre suivant nous passerons à la pratique (ce qui sera beaucoup plus concret pour vous)

Une chose est sûre, et vous devriez déjà vous mettre ça en tête, c'est que vous ne pourrez jamais avoir un site qui s'affiche pareil au pixel près sur tous les navigateurs. A la place, nous apprendrons à faire en sorte que notre site s'affiche "à peu près correctement" sur chacun de ces navigateurs, et ça sera déjà pas si mal 😊

---

## Transformer un inline en block (et vice-versa)

Dans le genre, "je suis là juste pour vous embrouiller l'esprit", je crois que je vais gagner le premier prix là 😊  
Je vais vous apprendre ici à ~~repousser les lois de la physique~~ à modifier les lois du CSS (brrr...).

Les images ( `<img />` ) sont des balises de type inline, et vous préféreriez que ce soient des blocks ? Pas de problème !  
Les titres ( `<h1>` ) sont de type block, et ça vous a toujours embêté, vous préféreriez que ça soient des balises inline ? Aucun problème !

Il existe une propriété CSS très puissante, elle s'appelle *display* (à manipuler avec précaution, sinon vous risquez de tout faire péter :p)  
Cette propriété peut prendre les valeurs suivantes :

- block : la balise concernée deviendra de type block.
- inline : la balise concernée deviendra de type inline.

En fait, *display* peut prendre beaucoup d'autres valeurs (c'est ça d'être une propriété puissante ;)), mais je vous en fais volontairement grâce et je vous montre juste les 2 plus utilisées.

Pour transformer une balise inline en balise de type block, vous allez devoir taper :  
`display: block;`

Par exemple, si je veux que TOUTES mes images soient de type block, j'écrirai donc ceci :

### Code : CSS

```
img
{
    display: block;
}
```



Attention, après avoir fait cela toutes vos images vont aller automatiquement à la ligne, comme le font les autres balises de type block.

Rien n'impose que TOUTES vos images soient de type block, hein. J'espère que vous l'avez compris depuis le temps : on peut très bien utiliser l'attribut "class" sur une balise pour qu'elle ait une présentation différente.

Par exemple :

### Code : HTML

```


```

Bien entendu, il faudra écrire le code CSS suivant pour que la deuxième image soit de type block :

### Code : CSS

```
.imageblock
{
    display: block;
}
```

J'espère que je ne vous apprend rien sur le fonctionnement de l'attribut *class*, sinon retournez voir le premier chapitre de la partie II au triple galop 🤪



Et l'inverse alors ? Pour transformer une balise de type block en balise de type inline ?

Bah c'est pareil, sauf qu'on utilise *display:inline*; 😊

Je ne vous refais pas un exemple de code, je pense que vous êtes assez grands pour deviner tous seuls comment il faut faire 😊


Petite remarque en passant : on transforme généralement des inline en block, mais on fait plus rarement l'inverse. Il faut dire que les balises block ont pas mal d'avantages : on peut modifier leur taille, leur bordure, leurs marges etc... Bref, tout ça vous le savez déjà 😊

---

## Les flottants

La première technique que nous allons voir, ce sont les **flottants**. C'est un nom un peu bizarre je vous l'accorde, mais c'est en fait pas si sorcier que ça.

Pour que vous voyiez bien de quoi on parle, voici ce que nous allons apprendre à faire :

 Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit. Donec  
vitae lorem imperdiet lacus molestie  
molestie. Cum sociis natoque penatibus  
et magnis dis parturient montes, nascetur ridiculus  
mus. Donec eu purus. Phasellus metus lorem,  
blandit et, posuere quis, tincidunt vitae, ante.  
Vivamus consequat mauris a diam. Vivamus nibh  
erat, hendrerit nec, aliquet ut, hendrerit quis, nunc.  
Vestibulum et turpis et elit tempor euismod.  
*Une image flottant à gauche*

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit. Donec  
vitae lorem imperdiet lacus molestie  
molestie. Cum sociis natoque penatibus  
et magnis dis parturient montes, nascetur ridiculus  
mus. Donec eu purus. Phasellus metus lorem,  
blandit et, posuere quis, tincidunt vitae, ante.  
Vivamus consequat mauris a diam. Vivamus nibh  
erat, hendrerit nec, aliquet ut, hendrerit quis, nunc.  
Vestibulum et turpis et elit tempor euismod.  
*Une image flottant à droite*

Cela permet en gros d'"entourer" un élément (ici une image) par du texte. Ca permet d'avoir une jolie présentation facilement.

J'imagine que la question qui vous brûle les lèvres maintenant est : "Mais quelle est donc la propriété magique qui fait flotter ? !".

La réponse est... float ("flottant" en anglais). Cette propriété peut prendre 2 valeurs, que vous pourriez d'ailleurs avoir deviné tous seuls 🤔 :

- **left** : l'élément flottera à gauche.
- **right** : l'élément flottera... à droite ! Oui bravo 😊

L'utilisation des flottants est très simple, tellement simple qu'on a parfois tendance à se compliquer la vie (et je parle en connaissance de cause). En fait, il n'y a que 2 choses à faire :

1. Vous appliquez un float à une balise.
2. Puis vous continuez à écrire du texte à la suite normalement.



On peut aussi bien utiliser la propriété *float* sur des balises block que sur des balises inline, vous allez voir...

## Faire flotter une image

Nous allons apprendre ici à faire flotter une image (qui est de type "inline" je vous rappelle). Voici le code XHTML que nous devons taper dans un premier temps :

### Code : HTML

```
<p></p>
```

<p>Ceci est un texte normal de paragraphe, écrit à la suite de  
l'image mais qui l'"entourera" car l'image est flottante.<br />  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum  
mollis scelerisque nulla. Donec feugiat augue et sem. Nulla ut  
purus. Pellentesque habitant morbi tristique senectus et netus et  
malesuada fames ac turpis egestas. Nulla volutpat leo quis magna.  
Donec accumsan lobortis ligula. Donec nec ante eu wisi tempus  
dictum. Sed nulla est, laoreet nec, consequat quis, iaculis non,  
tortor. Nunc sed purus. Sed metus. Donec posuere. Pellentesque  
porttitor tortor non eros. Ut rutrum erat a nunc. Duis non erat et  
orci viverra mollis. Sed lacinia velit a magna. Etiam aliquam, felis  
eu imperdiet auctor, ante augue dignissim odio, a pharetra tellus  
neque vel urna. Cras gravida adipiscing lectus. Nullam lorem ipsum,  
convallis eleifend, congue placerat, dictum non, leo. Vestibulum  
ante ipsum primis in faucibus orci luctus et ultrices posuere  
cubilia Curae.</p>



Une erreur courante que l'on a tendance à faire, c'est de mettre l'image flottante après le texte. Non, il ne faut pas : le flottant doit être avant le texte, qu'il flotte à droite ou qu'il flotte à gauche c'est pareil. Vous essaieriez de mettre la balise <img /> après le paragraphe, et vous verrez que ça ne marche plus 😊

Ce qu'il faut bien comprendre maintenant, c'est qu'on a juste besoin de modifier le CSS de l'image, et non pas du paragraphe. Le paragraphe n'a pas besoin d'être modifié, il sera automatiquement placé comme il faut. Voici le seul bout de code CSS qu'on a besoin de taper, qui permet de faire flotter l'image à gauche :

### Code : CSS

```
.imageflottante  
{  
    float: left;  
}
```

Essayer !

Amusez-vous aussi à faire flotter l'image à droite, c'est tout bête : mettez *float:right*; et le tour est joué ! 😊

## Créer une lettrine (exercice)

Parfois, il m'arrive de faire un cauchemar. J'imagine que je suis en train de rédiger un long cours sur les flottants en CSS, et que tout le monde bâille, s'endort, bref s'ennuie. Brrr... Cette vision me fait froid dans le dos à chaque fois.

Alors, pour éviter que le pire de mes cauchemars se produise, j'ai eu l'idée de vous faire travailler un peu sur un exercice amusant (du moins je l'espère :p).



Vous savez ce qu'est une lettrine ? Non.  
Bon c'est pas gagné 😊

Une lettrine, c'est la première lettre d'un paragraphe écrite beaucoup plus grosse. On en voit tout le temps dans les journaux (si vous lisez le journal ;o). Voici ce que vous devez arriver à faire :

**B**onjour tout le monde !  
commence par une lett  
pas plus joli comme ça  
trouve que ça invite à la lectur  
? Faire une lettrine en CSS, c

La seule consigne, c'est d'utiliser le code XHTML suivant, que vous n'avez pas le droit de changer :

**Code : HTML**

```
<p>Les visiteurs de l'hypermarché Carrefour de Haidian, un quartier  
nord de Pékin où sont installées les universités et les entreprises  
high-tech, ne sont pas accueillis en ce moment par le Père Noël mais  
par... la Vénus de Milo. Carrefour a décidé de donner dans le  
culturel en présentant, dans le cadre de l'année de la France en  
Chine, quarante reproductions de sculptures célèbres issues de  
collections des musées français. L'initiative remporte un grand  
succès, à en croire le nombre de personnes qui se prennent en photo  
devant les oeuvres, ou l'affluence aux visites commentées par des  
guides.</p>
```

Votre consigne, c'est de créer un fichier CSS qui permettra de créer cette présentation sous forme de lettrine. Vous ne pouvez pas modifier le fichier XHTML, mais par contre sur le CSS tous les coups sont permis : vous pouvez utiliser les propriétés que vous voulez.

Au boulot ! 😡

...  
...  
...

Drrring ! Allez c'est l'heure de la correction !

Bien évidemment il fallait utiliser un `float:left`, mais comment faire pour que la première lettre uniquement soit affectée ? Sans changer le XHTML ?

Si vous avez trouvé tout seul qu'il fallait utiliser le pseudo-élément `:first-letter` (= "première lettre"), eh bien bravo ! C'était la seule vraie difficulté, après le reste coule de source et on peut s'amuser à mettre les propriétés qu'on veut.

Il faut penser en particulier à utiliser `font-size`, pour modifier la taille de la première lettre pour qu'elle fasse disons... 3 hauteurs de lignes. Pour cela, j'ai mis la valeur "3em" (= 3 hauteurs de lignes), mais si vous avez tapé "300%" c'est tout aussi bon ! Par contre, il vaut mieux éviter de mettre une valeur en pixels comme "50px" parce qu'on ne connaît pas la taille d'une ligne (ça peut changer).

Voici le code que j'ai fait pour réaliser la lettrine :

**Code : CSS**

```
p:first-letter /* Je veux que la première lettre de mes  
paragraphe... */  
{  
    float: left; /* Flotte à gauche */
```

```
font-size: 3em; /* Fasse une hauteur de 3 lignes */
font-family: Arial, Georgia, "Times New Roman", Times, serif; /*
Soit mise en Arial si possible*/
font-weight: bold; /* Soit écrite en gras (c'est plus voyant) */
margin-right: 5px; /* Qu'il y ait une marge de 5px à droite pour
que ça colle pas trop au reste du texte */
}
```

Essayer !

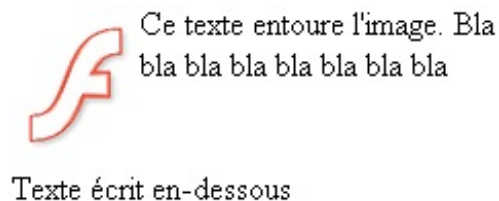
Ne vous arrêtez pas là ! Rien ne vous empêche d'écrire la lettrine en blanc sur fond noir, ou même d'utiliser une image de fond ! Laissez donc libre cours à votre imagination 😊

## Stopper un flottant

Quand vous mettez en place un flottant, le texte l'"entoure", ça on l'a compris.

Mais comment faire si vous voulez qu'au bout d'un moment le texte continue **en-dessous** du flottant ? On pourrait faire plusieurs `<br />` à la suite, mais ça serait pas pratique ni très propre...

En gros, on aimerait pouvoir faire ça :



Il existe en fait une propriété CSS (sans blague) qui permet de dire : "Stop, ce texte doit être en-dessous du flottant et non plus à côté". C'est la propriété *clear* qui peut grosso modo prendre 3 valeurs :

- **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 ce soit après un `float:left`; ou après un `float:right`;

Pour simplifier, on va utiliser tout le temps le `clear:both`, qui marche après un flottant à gauche et après un flottant à droite (ça marche à tous les coups quoi).

Pour illustrer son fonctionnement, on va prendre ce code XHTML :

### Code : HTML

```
<p></p>
<p>Ce texte est écrit à côté de l'image flottante.</p>
<p class="dessous">Ce texte est écrit sous l'image flottante.</p>
```

### Code : CSS

```
.imageflottante
{
float: left;
```

```
}  
.dessous  
{  
  clear: both;  
}
```

[Essayer !](#)

Et voilà le travail 😊

On applique un *clear:both*; au paragraphe que l'on veut voir continuer sous l'image flottante, et le tour est joué !

---

## Positionnement absolu, fixe et relatif

Hormis les flottants, il existe 3 façons de positionner un block en CSS :

- **Le positionnement absolu** : il nous permet de placer un block n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre etc...)
- **Le positionnement fixe** : c'est pareil que le positionnement absolu, mais cette fois le block reste toujours visible, même si on descend plus bas dans la page. C'est un peu comme *le background-attachment:fixed;* (si vous vous en souvenez encore ;))
- **Le positionnement relatif** : c'est le plus compliqué des trois. En gros, ça nous permet de déplacer un block par rapport à sa position normale.



Comme pour les flottants, le positionnement absolu, fixé et relatif fonctionne aussi sur des balises de type inline comme `<img />`

Toutefois, vous verrez qu'on l'utilise bien plus souvent sur des balises block que sur des balises inline.

Il faut d'abord faire son choix entre les 3 modes de positionnement disponibles. Pour cela, on utilise la propriété CSS `position` à laquelle on donne une de ces valeurs :

- **`absolute`** : positionnement absolu.
- **`fixed`** : positionnement fixe.
- **`relative`** : positionnement relatif.

Nous allons étudier chacun de ces positionnements un à un.

### Le positionnement absolu

Vous savez que, pour effectuer un positionnement absolu, il faut taper :

```
position: absolute;
```

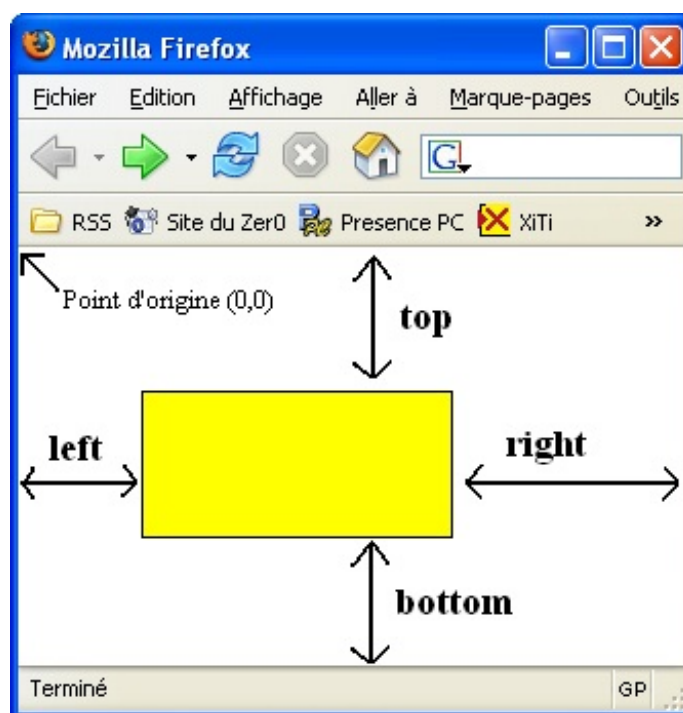
Mais ça ne suffit pas ! 💡

On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire **où on veut que le block soit positionné sur la page** . Pour faire cela, on va réutiliser 4 propriétés CSS que vous commencez certainement à bien connaître :

- **`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.

On peut leur donner une valeur en pixels, comme "14px", ou bien une valeur en pourcentage, comme "50%".

Si ce n'est pas très clair pour certains d'entre vous, ce schéma devrait vous aider à comprendre :



Avec ça, vous devriez être capables de positionner correctement votre block 😊

Il faut donc utiliser la propriété *position* et au moins une des 4 propriétés ci-dessus (top, left, right ou bottom). Si on écrit par exemple :

```
position: absolute;
right: 0px;
bottom: 0px;
```

... cela signifiera que le block doit être positionné tout en bas à droite (0 pixels par rapport à la droite de la page, 0 par rapport au bas de la page).

Pour le coup, on va utiliser la balise universelle <div> (de type block). Je vais rédiger un paragraphe, et je vais mettre dans le block <div> deux ou trois mots :

#### Code : HTML

```
<p>
Ceci est un paragraphe normal.<br />
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
vel libero. Cras dolor. Quisque quis odio eget justo pulvinar
aliquet. Morbi luctus mi. Fusce leo. Integer eleifend condimentum
felis. Phasellus vitae nibh. Mauris pellentesque porta magna.
Quisque at turpis. Praesent semper odio eget risus. Praesent
bibendum imperdiet massa. Quisque ac arcu ac augue dapibus
vestibulum. Pellentesque gravida. Mauris turpis. Aenean molestie.
Praesent at quam et ligula pellentesque luctus.
</p>

<div>Block positionné</div>
```

[Essayer !](#)

Rien d'extraordinaire, vous avez un paragraphe suivi d'un block div qui contient un peu de texte. Jusque là, rien de bien excitant.

Maintenant, ajoutons un peu de CSS pour positionner où on veut notre block div :

#### Code : CSS

```
div
{
    background-color: yellow;
    border: 1px solid green;

    position: absolute;
    left: 35px;
    top: 50px;
}
```

[Essayer !](#)

J'ai mis un fond jaune et une bordure pour qu'on repère mieux le block.

Comme vous pouvez le constater, le block se met au-dessus du paragraphe. Il le masque. Ca, c'est un des grands avantages et défauts du positionnement absolu : il n'y a aucun contrôle, vous pouvez mettre votre block vraiment **où vous voulez sur la page**, mais il faut faire attention à ce qu'il ne masque pas de texte.

Voici un autre exemple pour vous montrer qu'on peut vraiment mettre le block n'importe où :

Code : CSS

```
div
{
    background-color: yellow;
    border: 1px solid green;

    position: absolute;
    right: 50%;
    bottom: 20px;
}
```

[Essayer !](#)



Une petite précision technique pour ceux qui veulent aller plus loin (les autres fermez les yeux) : si vous positionnez un block A en absolu, et qu'à l'intérieur de ce block vous positionnez un autre block B en absolu, ce positionnement-là ne se fera plus par rapport au coin en haut à gauche de la fenêtre mais par rapport au coin en haut à gauche du block A.

Ah... Les joies du positionnement CSS 😊

## Le positionnement fixe

Le fonctionnement est **exactement le même** que pour le positionnement absolu, sauf que cette fois le block reste fixe, même si on descend plus bas dans la page.

Au lieu de faire *position: absolute;*, on va taper *position: fixed;* dans notre CSS. Comme tout à l'heure, on s'aide de top, left, right et bottom pour placer notre block où on veut sur la page.

Le *position: fixed;* est particulièrement utile pour faire un menu qui reste toujours visible :

Code : CSS

```
div
{
    background-color: yellow;
    border: 1px solid green;
    width: 150px;
    height: 250px;

    position: fixed;
    right: 40px;
    top: 60px;
}

p
{
    width: 300px;
}
```

[Essayer !](#)

## Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir une grosse prise de tête si on n'y fait pas très attention.

Personnellement, je ne l'utilise pas beaucoup.

Le positionnement relatif sert généralement à faire des "ajustements".

Prenons par exemple un texte important, situé entre 2 balises <strong>.

Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

### Code : CSS

```
strong
{
    background-color: red; /* Fond rouge */
    color: yellow; /* Texte de couleur jaune */
}
```

[Essayer !](#)

Cette fois, le schéma que je vous ai montré tout à l'heure pour les positions absolue et fixe ne marche plus. Pourquoi ? Parce que l'origine a changé : le point de coordonnées (0, 0) ne se trouve plus en haut à gauche de votre fenêtre comme c'était le cas tout à l'heure. Non, cette fois l'origine se trouve en haut à gauche de la position actuelle de votre élément.

Tordu n'est-ce pas ? Bah oui, c'est une position relative. Ce schéma devrait vous aider à comprendre où se trouve l'origine des points :


  
 Pas de doute, **ce texte est important** si on veut comprendre corri

Donc, si vous faites un *position: relative* et que vous appliquez une des propriétés *top*, *bottom*, *left*, *right*, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Je vous l'avais dit que c'était tordu 🤪 En fait, il faut faire des tests pour bien comprendre.

Prenons donc un exemple : je veux que mon texte se décale de 55 pixels vers la droite et de 10 pixels vers le bas. Je vais donc demander à ce qu'il soit décalé de 55 pixels par rapport au "bord gauche", et de 10 pixels par rapport au "bord haut" :

## Code : CSS

```
strong
{
    background-color: red;
    color: yellow;

    position: relative;
    left: 55px;
    top: 10px;
}
```

[Essayer !](#)

Le texte s'est décalé par rapport à sa position initiale, comme ceci :



Voilà le principe. A vous de voir si vous avez besoin de l'utiliser, maintenant vous savez comment ça marche 😊

Comme quoi, être attentif en cours de maths ça peut servir : au moins vous savez ce que c'est l'origine d'un repère (le point de coordonnées 0, 0).

Dès qu'on parle de positionnement absolu, il y a forcément une origine... Cela nous permet de placer un block (ou même une balise inline comme `img`) n'importe où sur la page !

Avec ces connaissances-là, vous êtes parés pour la suite...

D'ailleurs, la suite du programme c'est quoi déjà ?... Ah oui ! La suite, c'est un TP (= Travaux Pratiques)

En effet, le chapitre suivant ne va rien vous apprendre de nouveau, mais il va pourtant beaucoup vous intéresser : je vais vous montrer comment on crée le design d'un site de A à Z avec ce qu'on a appris ! 😊

Eh oui, vous savez tout ce qu'il faut, encore faut-il maintenant savoir s'y prendre... Quand vous êtes prêts, on y va 😊

## Créons le design de votre site web !

Enfin nous y voilà 😊

C'est un chapitre un peu particulier, assez différent de ce que vous avez lu jusqu'à maintenant. En fait, c'est ce que j'appelle un "TP" (= Travaux Pratiques). Maintenant, vous ne pouvez plus vous contenter de lire mes chapitres à moitié endormis, vous allez devoir mettre la main à la pâte en même temps que moi 😊

On va réutiliser tout ce qu'on a appris jusqu'ici. Si vous en ressentez le besoin, il est encore temps d'aller relire les chapitres précédents pour vous rafraîchir la mémoire.



Quel est l'objectif de ce TP ?

Je crois que le titre est assez clair pour le coup : nous allons créer le design de votre site web 😊

Comme je me doute bien que pour vous c'est la première fois, il va falloir être attentifs. Créer un design, ce n'est pas une mince affaire : il faut de la créativité (ça j'en ai pas trop), du goût pour les couleurs (ça j'en ai pas du tout), du talent (décidemment je suis mal barré...) et enfin de l'expérience.

Tout ce que je peux vous communiquer moi, vous l'aurez compris, c'est mon expérience. Après, la créativité et tout le reste



c'est inné : on l'a ou on l'a pas 🤔

Au boulot moussaillon ! 🧑🔧

---

## Bon, par quoi on commence ?

Vous le savez maintenant, une page web c'est une combinaison de 2 fichiers :

- **Un fichier XHTML** (.html ou .htm) : c'est dans ce fichier que se trouve le contenu de la page (le texte). Ce fichier est constitué de balises comme <strong> qui servent à indiquer si tel texte est important, si c'est un paragraphe ou une citation etc...
- **Un fichier CSS** (.css) : c'est le fichier qui permet de créer la présentation de notre page web. C'est lui qui indique qu'un texte est en rouge, qu'il est centré, dans la police "Arial" etc... En théorie, le CSS est facultatif. Ceci dit, sans CSS, nos pages web seraient bien moches 🤡

Si vous avez bien suivi jusqu'ici et que vous avez fait quelques tests, vous savez par quoi on commence :

1. En premier on tape le contenu de la page (XHTML)
2. Et ensuite on modifie la présentation de la page (CSS)

Nous allons donc suivre cet ordre-là.

Vous allez voir que la création du fichier XHTML sera (très) rapide et facile. En revanche, nous allons passer plus de temps sur le CSS. Parce que c'est délicat, qu'il y a pas mal de propriétés CSS à connaître, et aussi parce que des fois, on a beau être un pro, le navigateur n'affiche pas exactement les choses comme on voudrait.

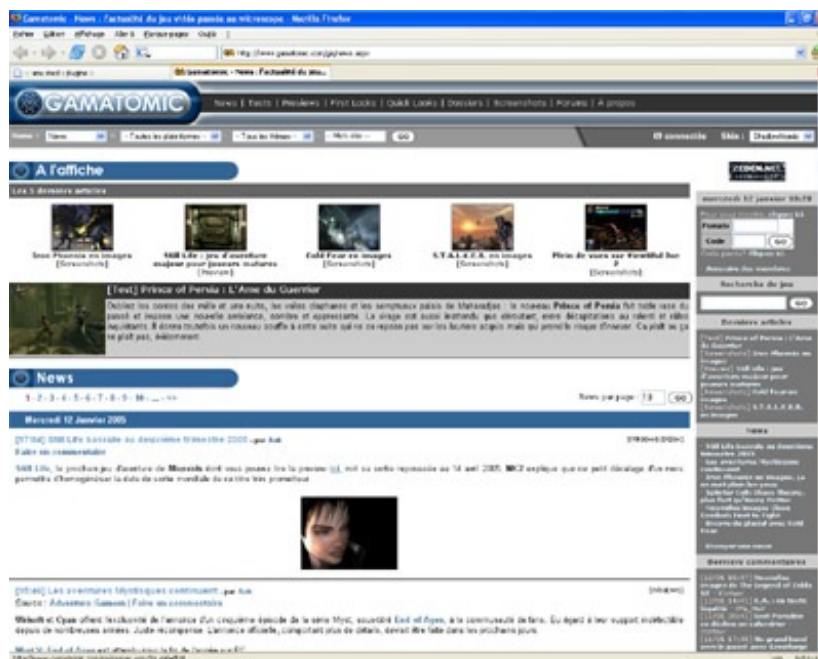
## Design fixe ou extensible ?

Un design fixe, c'est un design dont la largeur est de taille fixée (par exemple de 800 pixels). Il y a en général une marge à gauche et à droite, et le contenu du site se trouve au milieu. Voici un site au design fixe :



*Un design de taille fixe*

Un design extensible, c'est un design qui s'élargit automatiquement en fonction de la résolution du visiteur. Si le visiteur est en 1024\*768, le design fera 1024 pixels de large. Si le visiteur est en 1600\*1200, il fera 1600 pixels de large.



Un design de taille extensible

Personnellement, j'ai tendance à préférer les designs extensibles : je n'aime guère avoir une partie de mon écran inutilisée. Cependant, vous verrez si vous faites le tour des sites web que vous visitez régulièrement que nombre d'entre eux sont en design fixes. Certains disent trouver ça plus joli, mais je vais vous dire la vérité : c'est aussi parce que c'est plus facile à faire



Quel type de design allons-nous faire ?

Eh bien pour tout vous dire, la création d'un design extensible est assez complexe, et comme c'est votre premier design je crains que ça ne fasse trop à la fois. Nous allons donc réaliser un **design de taille fixe**, ce qui est déjà pas si mal. Lorsque vous vous serez améliorés et que vous aurez un peu plus d'expérience, vous pourrez vous lancer dans la création d'un design extensible. Vous aurez tout le temps de vous y intéresser



## Primo, le XHTML

Première étape : on ouvre notre éditeur de texte (Notepad++ si vous avez pris celui que je vous ai montré au début), et vous créez un nouveau fichier XHTML.

Allez-y allez-y, faites-le en même temps que moi ! 😊

Pour commencer, on va taper le code de "base" d'une page XHTML que je vous ai montré dans un des premiers chapitres. Le revoici à nouveau :

### Code : HTML

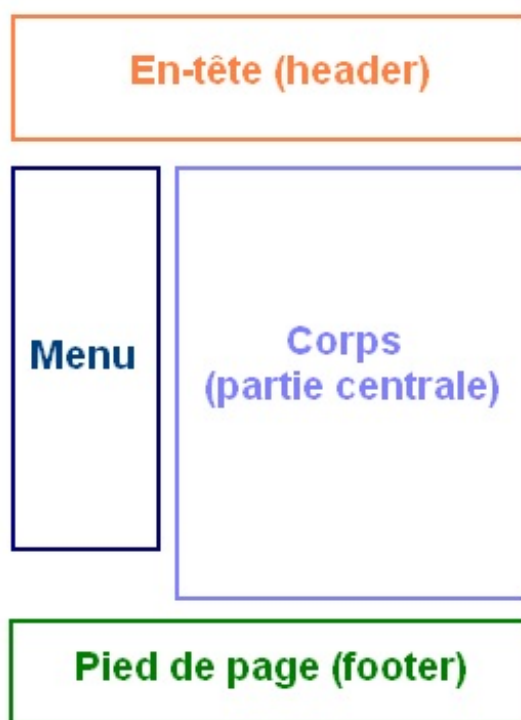
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
  <head>
    <title>Bienvenue sur mon site !</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>

  </body>
</html>
```

Comme vous le savez, nous allons écrire désormais uniquement entre les balises <body></body>.

## La structure générale

Bien souvent, on structure sa page web à l'aide de blocks <div>, la fameuse balise universelle de type "block" dont je vous ai parlé. On va avoir en général au moins 4 blocks, présentés comme ceci :



1. En haut, l'en-tête (aussi appelé "header"). Cet en-tête contient en général le titre de votre site, sous forme de bannière.

2. En-dessous, vous avez la partie principale de votre page avec :
  - A gauche ou à droite, le menu.
  - De l'autre côté, le corps de la page, c'est-à-dire la partie qui contiendra le texte de votre site.
3. Enfin, on met en général en bas un "pied de page" (aussi appelé "footer"), dans lequel on indique l'auteur du site, éventuellement son e-mail et un petit copyright pour faire classe 😊

Voilà pour les grandes lignes.

En XHTML, on va créer ces 4 blocks à l'aide de 4 <div> différents :

#### Code : HTML

```
<div id="en_tete">
  <!-- Ici on mettra la bannière -->
</div>

<div id="menu">
  <!-- Ici on mettra le menu -->
</div>

<div id="corps">
  <!-- Ici on mettra le contenu principal de la page (tout le texte
quoi) -->
</div>

<div id="pied_de_page">
  <!-- Enfin, on mettra en bas de la page le nom de l'auteur, un
copyright... -->
</div>
```

C'est tout simple, on a mis un block <div> pour chaque gros block de la page. On verra tout à l'heure ce qu'on met dedans.

Particularité qui a dû vous choquer : on utilise ici des attributs *id* et non des *class* comme on a eu l'habitude d'en utiliser jusqu'ici. Si vous ne vous souvenez pas de la différence entre un id et une class, retournez voir les premiers chapitres sur le CSS.

Grosso modo, un id et une class c'est quasiment la même chose, sauf qu'on ne peut mettre un id qu'une seule fois dans une page XHTML, contrairement aux class qu'on peut utiliser plusieurs fois. Il y a peu d'id dans une page en général, on en met surtout pour les grands "blocks" du design comme on vient de le faire ici.



**Vous devez écrire vos blocks dans l'ordre où ils seront affichés (de haut en bas) dans le code XHTML. En premier l'en-tête, en dernier le pied de page.**

**Pour ce qui est du menu et du corps qui se trouvent au même niveau, l'ordre entre eux deux n'importe pas. On pourra modifier leur position dans le CSS (menu à gauche, ou menu à droite).**

Bon, maintenant que fait-on ? Eh bien on va voir plus en détail ce qu'on met dans chacun des 4 <div> 😊

## L'en-tête

Alors ça, ça va être rapide. En général un en-tête c'est juste une bannière (une image), voire pour ceux qui n'ont pas envie de dessiner un simple titre <h1> (titre du site).

Pour ma part, je vais mettre une bannière, par exemple celle-ci :



Pour ceux qui croiraient que je leur aurais menti sur mon niveau en graphisme : non, cette image n'est pas de moi 😏

Toutes les images que vous allez voir dans ce chapitre sont copyright **zaz et venom** tous droits réservés (je leur ai promis de mettre leurs noms en lettres d'or dans le tuto, une promesse est une promesse 😊). Merci donc à eux pour l'aide importante qu'ils m'ont fournie pour ce chapitre !

Revenons à nos moutons. Comment mettre cette image dans le <div> de l'en-tête ? On a 2 possibilités :

- Soit on crée une balise <img /> à l'intérieur, afin d'insérer notre bannière. C'est une méthode facile pour insérer une image, ça on sait le faire.
- Soit on ne met rien à l'intérieur du <div id="en\_tete"> et on mettra la bannière sous forme d'image de fond tout à l'heure dans le CSS.

La seconde solution n'est pas particulièrement difficile mais un peu surprenante je dois l'avouer. En fait, on va créer un bloc "vide" qui va servir uniquement à afficher une image, et **cette image sera affichée par le CSS**.

Le résultat sera apparemment le même que si on avait utilisé <img /> mais nous verrons à la fin de ce chapitre que cette technique s'avèrera payante si vous voulez pouvoir changer le design de votre site facilement.

#### Code : HTML

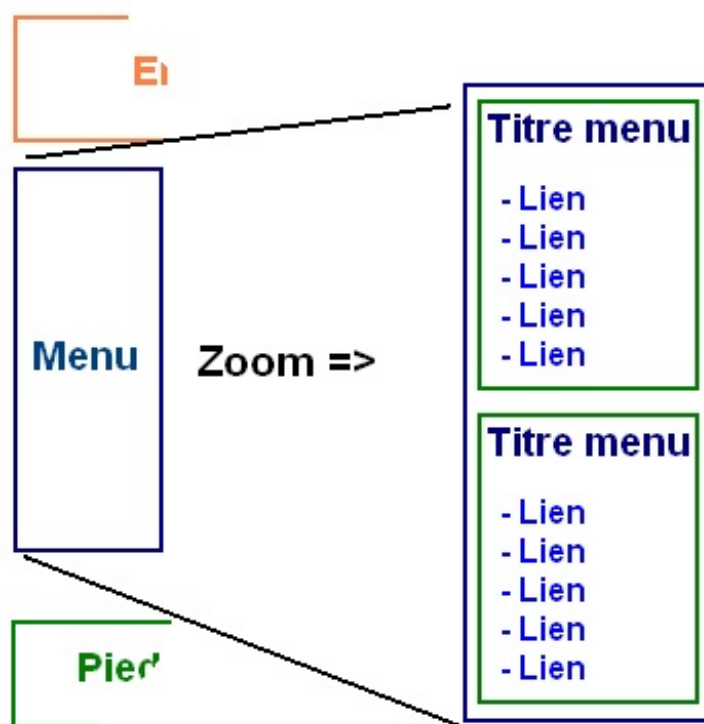
```
<div id="en_tete">  
  
</div>
```

On ne met donc rien entre le <div> et le </div> (juste un espace à la rigueur 😏)

## Menu et sous-menus

Dans le block du menu (en bleu marine), on va avoir à créer plusieurs sous-blocks (eh oui, ça s'emboîte) pour séparer les différents éléments du menu.

Regardez-bien le schéma ci-dessous : c'est un zoom sur le block "Menu" du schéma de tout à l'heure :



Comme vous le voyez, il contient 2 sous-blocks (cadres verts) qui vont servir de sous-menus. Comment faire cet emboîtement de blocs ? C'est très simple en XHTML :

#### Code : HTML

```
<div id="menu"> <!-- Cadre englobant tous les sous-menus (en bleu
marine sur le schéma) -->

  <div class="element_menu"> <!-- Cadre correspondant à un sous-
menu -->
    Texte du premier menu
  </div>

  <div class="element_menu">
    Texte du second menu
  </div>

</div>
```

Il nous suffit d'imbriquer des balises <div>, ces fameuses balises blocks universelles (on les appelle aussi "balises génériques" d'ailleurs).

J'ai mis des commentaires dans le code source pour que vous puissiez bien repérer à quoi correspond chaque <div>. Le premier englobe tous les autres sous-menus (j'ai donné à ces sous-menus une class "element\_menu" pour qu'on puisse les repérer tout à l'heure dans le CSS).

Bien, rajoutons maintenant le texte à l'intérieur du menu pour faire comme dans le dernier schéma.

On va mettre dans chaque "element\_menu" un titre de menu. Vous vous souvenez de la balise qui sert à faire des titres ? Oui, il y a 6 balises (selon l'importance du titre) : de <h1> à <h6>. Comme les titres de menus sont un peu moins importants, je vais les mettre en <h3>.

Ensuite, on va mettre une liste à puces (<ul>) pour avoir un menu bien organisé.

#### Code : HTML

```
<div id="menu">
```

```

<div class="element_menu">
  <h3>Titre menu</h3> <!-- Titre du sous-menu -->
  <ul>
    <li>Lien</li> <!-- Liste des liens du sous-menu -->
    <li>Lien</li>
    <li>Lien</li>
  </ul>
</div>

<div class="element_menu">
  <h3>Titre menu</h3>
  <ul>
    <li>Lien</li>
    <li>Lien</li>
    <li>Lien</li>
  </ul>
</div>

</div>

```

En général, on met souvent des listes à puces pour les menus car cela permet de créer une bonne organisation facilement. Vous avez peut-être remarqué que j'ai marqué des "Lien", mais que je n'ai pas mis de balise `<a href=...">`. C'était juste pour ne pas trop encombrer le schéma de suite, bien entendu qu'on va mettre des liens dans notre menu sinon il ne servirait pas à grand chose 😊

#### Code : HTML

```

<div id="menu">

  <div class="element_menu">
    <h3>Titre menu</h3>
    <ul>
      <li><a href="page1.html">Lien</a></li>
      <li><a href="page2.html">Lien</a></li>
      <li><a href="page3.html">Lien</a></li>
    </ul>
  </div>

  <div class="element_menu">
    <h3>Titre menu</h3>
    <ul>
      <li><a href="page4.html">Lien</a></li>
      <li><a href="page5.html">Lien</a></li>
      <li><a href="page6.html">Lien</a></li>
    </ul>
  </div>

</div>

```

Et voilà un menu tout beau tout propre, avec ses sous-menus !

Vous pouvez, si vous le désirez, ajouter d'autres sous-menus. Il n'y a pas de limite, mais attention à ne pas trop abuser de sous-menus sinon le visiteur risque de se perdre facilement.



Vous n'êtes pas obligés (même si c'est ce qu'on fait le plus souvent) de mettre des liens et une liste à puces dans un menu. Si vous voulez écrire un peu de texte dedans, utilisez la balise de paragraphe `<p></p>`.



## Le corps

Le corps, c'est la partie principale de la page. C'est là qu'il y aura tout le contenu de votre page.

Alors, comme je ne sais pas de quoi votre site va parler, je ne peux pas vous dire ce que vous devez mettre : seuls vous le savez



Je vais quand même vous donner quelques indications pour vous aider à démarrer.

Pour commencer, on va mettre un titre <h1> qui sera le titre principal de la page. En général, je mets le même titre dans la balise <title> (dans <head>) que dans la balise <h1>.

Ensuite, on écrira nos paragraphes <p> éventuellement séparés par des sous-titres <h2> pour un peu mieux structurer son texte



Cela nous donne un code relativement simple pour le corps :

### Code : HTML

```
<div id="corps">
  <h1>Mon super site</h1>

  <p>
    Bienvenue sur mon super site !<br />
    Vous allez adorer ici, c'est un site génial qui va parler
    de... heu... Je cherche encore un peu le thème de mon site :-D
  </p>

  <h2>A qui s'adresse ce site ?</h2>
  <p>
    A tout le monde ! Si je commence à privilégier certaines
    personnes, on va m'accuser de discrimination ;o)<br />
    Que vous soyez fans de fusils à pulsion plasma ou de Barbie
    et Ken, ce site est fait pour vous ! Si si !
  </p>

  <h2>L'auteur</h2>
  <p>
    L'auteur du site ? Bah, c'est moi, quelle question :-p<br />
    Je vais essayer de faire le meilleur site du monde (ça doit
    pas être bien compliqué). Mon objectif est d'attirer un maximum de
    visiteurs, de les rendre accros à mon site, puis de les mettre en
    mon pouvoir.<br />
    Je prendrai ensuite le contrôle du Monde. Une fois que ce
    sera fait, j'irai explorer les confins de l'Univers à la recherche
    de nouveaux peuples à soumettre à ma terrible puissance.
    MooUUuUuuUAhahHaaAhAAAAah !!! (rire diabolique).
  </p>
</div>
```

Ne faites pas attention au texte que j'ai écrit, c'est juste un petit délire perso pour "remplir" un peu la page 😄



N'oubliez pas que cette zone est libre. Si vous voulez mettre des images, des liens, d'autres paragraphes, faites-le !  
C'est le contenu de votre site après tout 😊

## Allez le dernier : le pied de page

Comme je vous l'ai dit, le pied de page sert à écrire le nom de l'auteur en général, et si ça vous chante vous pouvez mettre un copyright pour faire classe auprès de vos copains (mais ça servira qu'à ça hein 😊)

#### Code : HTML

```
<div id="pied_de_page">
  <p>Copyright "Tout pourri Corporation" 2005, tous droits réservés</p>
</div>
```

## Le code XHTML final

Nous avons vu en détail ce que nous devons mettre dans chacun des 4 gros blocks du design. Maintenant, on va assembler tous les codes sur lesquels on a travaillé, en n'oubliant pas de mettre en haut les balises DOCTYPE, <html>, <head>, <body> etc...

Ca nous donne ça :

#### Code : HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
  <head>
    <title>Mon super site</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>

  <body>

    <!-- L'en-tête -->

    <div id="en_tete">

    </div>

    <!-- Les menus -->

    <div id="menu">
      <div class="element_menu">
        <h3>Titre menu</h3>
        <ul>
          <li><a href="page1.html">Lien</a></li>
          <li><a href="page2.html">Lien</a></li>
          <li><a href="page3.html">Lien</a></li>
        </ul>
      </div>

      <div class="element_menu">
        <h3>Titre menu</h3>
        <ul>
          <li><a href="page4.html">Lien</a></li>
          <li><a href="page5.html">Lien</a></li>
          <li><a href="page6.html">Lien</a></li>
        </ul>
      </div>
    </div>

    <!-- Le corps -->
```

```

<div id="corps">
  <h1>Mon super site</h1>

  <p>
    Bienvenue sur mon super site !<br />
    Vous allez adorer ici, c'est un site génial qui va
    parler de... heu... Je cherche encore un peu le thème de mon site :-
    D
  </p>

  <h2>A qui s'adresse ce site ?</h2>
  <p>
    A tout le monde ! Si je commence à privilégier
    certaines personnes, on va m'accuser de discrimination ;o)<br />
    Que vous soyez fans de fusils à pulsion plasma ou de
    Barbie et Ken, ce site est fait pour vous ! Si si !
  </p>

  <h2>L'auteur</h2>
  <p>
    L'auteur du site ? Bah, c'est moi, quelle question :-
    p<br />
    Je vais essayer de faire le meilleur site du monde
    (ça doit pas être bien compliqué). Mon objectif est d'attirer un
    maximum de visiteurs, de les rendre accros à mon site, puis de les
    mettre en mon pouvoir.<br />
    Je prendrai ensuite le contrôle du Monde. Une fois
    que ce sera fait, j'irai explorer les confins de l'Univers à la
    recherche de nouveaux peuples à soumettre à ma terrible puissance.
    MooUUuUuuUAhahHaaAhAAAAah !!!! (rire diabolique).
  </p>
</div>

<!-- Le pied de page -->

<div id="pied_de_page">
  <p>Copyright "Tout pourri Corporation" 2005, tous droits
  réservés</p>
</div>

</body>
</html>

```

Eh bien moi je dis : félicitations !

Vous venez de réaliser votre première **vraie page XHTML** ! 😊

Je me doute bien que vous n'auriez pas deviné tout seuls, mais le principal c'est que vous compreniez et que vous suiviez en même temps que moi. Il n'y a aucune nouveauté dans ce code XHTML, vous avez déjà appris toutes ces balises. En plus, avouez que ce code XHTML n'est pas franchement très long (on faisait bien pire sur le web il y a quelques années).

Cependant, on n'a pas encore fini. En effet, un code XHTML comme ça tout seul sans CSS, ça ne donne pas quelque chose de bien extraordinaire, voyez plutôt :

[Essayer !](#)



QUOOOOIII ??? C'est sur ça que je travaille depuis une heure ?!

Oui, et le plus drôle c'est que ce code XHTML suffit pour réaliser le design. Il ne nous reste qu'à ajouter un peu de CSS, et notre site sera tout beau tout propre ! 😊

Vous ne me croyez pas ?

Faites attention... Ne sous-estimez pas la puissance du CSS, vous pourriez le regretter 😊

---

## Secundo, le CSS

Nous attaquons maintenant la partie CSS.

C'est une partie très importante, car il faut bien que vous compreniez comment chaque ligne de code modifie le design.

Nous allons donc là encore y aller très "pas à pas" et je vous proposerai de tester la page XHTML avec le CSS que nous avons fait au fur et à mesure.



Quant à vous, si vous voulez comprendre, vous avez tout intérêt à essayer de faire quelques modifications sur le CSS. Si une ligne vous perturbe, essayez de l'enlever tout simplement, et regardez le résultat que ça produit. C'est comme ça que j'ai appris, et je pense que c'est la meilleure façon de faire. Je vous invite donc à faire comme moi 😊

### Centrer le design

Pour commencer, nous allons travailler juste sur la balise <body>. Oui, je sais c'est une balise sur laquelle on n'a pas vraiment travaillé jusqu'ici, mais le principe reste le même pour le CSS.

<body>, je vous le rappelle, c'est la balise qui englobe tout le contenu de votre page web. Si on dit que notre <body> fait tant de pixels de large (pour avoir un design de taille fixe) et qu'on veut le centrer (avec margin:auto;), bah on a déjà pratiquement gagné 😊

Pour la taille de la page, je vais mettre 760 pixels. Pourquoi 760 ? Parce que c'est moins que 800px, et ainsi les visiteurs qui sont en résolution 800\*600 n'auront pas à se déplacer vers la droite pour voir le reste de la page.

Pour ce qui est des marges, un margin:auto suffira à centrer le design.

Je rajouterai une marge en haut (top) et en bas (bottom) de 20 pixels pour éviter que notre page soit trop "collée" avec le haut et le bas du navigateur.

Enfin, je mettrai une petite image de fond (background-image) à la page pour faire un peu moins vide.

Voilà ce que ça devrait donner :

#### Code : CSS

```
body
{
    width: 760px;
    margin: auto; /* Pour centrer notre page */
    margin-top: 20px; /* Pour éviter de coller avec le haut de la
fenêtre du navigateur. Essayez d'enlever pour voir ! */
    margin-bottom: 20px; /* Idem pour le bas du navigateur */
    background-image: url("images/fond.png"); /* Une petite image de
fond pour éviter d'avoir un vieux fond blanc :p */
}
```

Essayer !

Permettez-moi d'insister : essayez d'enlever quelques lignes pour observer le résultat sur la page :

- Si vous enlevez le fond (background-image), vous retrouverez un fond blanc de base.
- Si vous enlevez le margin-top ou le margin-bottom, vous verrez que votre page sera un peu trop "collée" avec les bords de la page.
- Si vous enlevez le margin:auto, votre page ne sera plus centrée et se retrouvera à gauche. Notez que le margin:auto ne marche pas sous IE 5 : ces visiteurs-là auront un design placé à gauche, mais ce n'est pas bien grave, vous pouvez me croire il y a pire 😊

## L'en-tête

Comme promis, nous allons afficher la bannière du site à l'aide du CSS (avec *background-image*). On a fait un truc un peu nouveau et apparemment idiot tout à l'heure : on n'a rien mis dans le <div> de l'en-tête.

C'est une technique que vous pourrez utiliser si besoin est pour faire en sorte que tout votre design soit compris dans le fichier CSS. Ca sera très pratique (entre autres) pour réaliser un site qui propose plusieurs designs différents.

A nous maintenant d'afficher notre bannière avec uniquement du CSS :

1. La première chose à faire, si on veut que la bannière soit visible, ce sera d'agrandir la taille du block "en\_tete" afin qu'il puisse contenir la bannière entière.

Comme notre bannière fait 758x100pixels, on va définir une largeur et une hauteur de ces dimensions :

```
width: 758px;
height: 100px;
```

2. Ensuite, on va indiquer quelle image de fond on veut mettre à notre grand block qui est pour l'instant vide :

```
background-image: url("images/banniere.png");
```

3. Normalement notre fond n'a la place d'apparaître qu'une seule fois, mais ça ne nous coûte pas grand chose de s'assurer que cette image ne sera pas répétée en mosaïque avec background-repeat :

```
background-repeat: no-repeat;
```

4. Enfin, c'est un détail mais afin d'éviter que l'en-tête ne soit trop collé avec ce qui se trouvera dessous (le menu et le corps), on va définir une petite marge en-dessous (margin-bottom) de quelques pixels :

```
margin-bottom: 10px;
```

Nous y sommes. Ca donne finalement cela :

### Code : CSS

```
body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
}

/* L'en-tête */

#en_tete
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}
```

[Essayer !](#)

A partir d'un block vide, nous avons pu afficher notre bannière uniquement en CSS. Désormais, si vous voulez changer la bannière de votre site, il faudra modifier le fichier CSS et non le fichier XHTML. Et ça, c'est une bonne chose vous le comprendrez tout à l'heure 😊



Euh, pourquoi il y a un # devant "en\_tete" ? C'est pas un point normalement qu'on met ???

Non, je vous rappelle pour ceux qui l'auraient oublié :

- Pour une class, on met un "." (point) devant le nom dans le CSS.
- Pour un id, on met un "#" (dièse) devant le nom. Or, regardez le code XHTML, "en\_tete" est un id !

## Le menu

Ca c'est un peu plus difficile.

Il va nous falloir placer le menu "à gauche" du corps, alors que pour l'instant le menu se trouve au-dessus.

Pour faire cela, la technique la plus courante et la plus rapide, c'est d'utiliser la propriété CSS *float*. C'est une propriété un peu particulière comme nous l'avons vu, et elle va être très pratique ici pour placer correctement le menu et le corps.

On va aussi donner une taille de 120 pixels au menu, pour qu'il ne soit pas trop grand.

Ensuite, nous allons travailler sur notre class "element\_menu" (qui correspond à une partie de menu). On va lui mettre une couleur de fond un peu plus gris foncé, une image de fond en repeat-x (qui se répètera uniquement horizontalement). Ensuite, on rajoutera une bordure à chaque élément de menu pour qu'on voie bien ses limites. "2px" solid black ça devrait être bon.

Enfin, on mettra un *margin-bottom* à chaque élément de menu pour éviter qu'ils ne soient trop collés les uns par rapport aux autres.

### Code : CSS

```
body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
}

/* L'en-tête */

#en_tete
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}

/* Le menu */

#menu
{
    float: left; /* Le menu flottera à gauche */
    width: 120px; /* Très important : donner une taille au menu */
}

.element_menu
{
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;
```

```
border: 2px solid black;

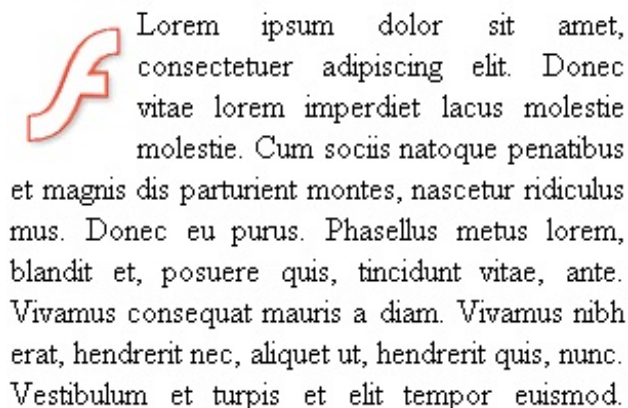
margin-bottom: 20px; /* Pour éviter que les éléments du menu ne
soient trop collés */
}
```

Essayer !

Encore une fois, si vous voulez comprendre tout ce charabia je vous conseille fortement d'essayer d'enlever quelques lignes du CSS pour voir ce que ça change.

Comme vous pouvez le constater si vous testez ce code, c'est pas encore la joie mais ça s'améliore. On distingue déjà bien nos bouts de menus, par contre le corps se retrouve sous le menu une fois qu'il l'a dépassé par le bas.

C'est normal car c'est comme ça que fonctionne un *float*, rappelez-vous ce petit schéma simple sur une image flottante :



Comment faire alors pour que le corps de la page ne passe pas sous le menu flottant ?

Il nous faudra rajouter une marge à gauche sur le corps (*margin-left*). On fera ça tout à l'heure quand on travaillera sur le corps. Si ça peut vous rassurer, c'est une astuce toute simple et pourtant je ne la connaissais pas il y a encore 5 minutes !

... Comment ça ça ne vous rassure pas ? 🤔

Bon, on n'a pas encore fini de travailler sur les menus. On va rajouter quelques effets.

Je ne vais pas m'attarder à expliquer ça trop en détail, c'est du tout simple, et c'est juste pour améliorer la présentation.

Le principal qu'il faut bien comprendre ici, c'est que je me sers des "imbrications de balises".

Vous vous souvenez, quand on écrit par exemple :

```
.element_menu ul
```

... cela signifie "Ce CSS va concerner les balises `<ul>` se trouvant à l'intérieur d'un `element_menu`".

#### Code : CSS

```
body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
}

/* L'en-tête */

#en_tete
```



```
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}

/* Le menu */

#menu
{
    float: left;
    width: 120px;
}

.element_menu
{
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;

    margin-bottom: 20px;
}

/* Quelques effets sur les menus */

.element_menu h3 /* Tous les titres de menus */
{
    color: #B3B3B3;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
    text-align: center;
}

.element_menu ul /* Toutes les listes à puces se trouvant dans un
menu */
{
    list-style-image: url("images/puce.png"); /* On change
l'apparence des puces */
    padding: 0px; /* Tous les côtés ont une marge intérieure de 0
pixels */
    padding-left: 20px; /* ... mais on modifie ensuite la marge de
gauche, donc celle-là fera finalement 20 pixels */
    margin: 0px; /* Idem pour margin, ça nous évite d'avoir à en
écrire 4 (margin-left, margin-right...) */
    margin-bottom: 5px; /* Même chose que tout à l'heure, on modifie
ensuite juste margin-bottom, mais tous les autres sont à 0px */
}

.element_menu a /* Tous les liens se trouvant dans un menu */
{
    color: #B3B3B3;
}

.element_menu a:hover /* Quand on pointe sur un lien du menu */
{
    background-color: #B3B3B3;
    color: black;
}
```

Essayer !

C'est déjà plus joli non ?

En plus, l'effet du `:hover` sur les liens du menu est très facile à réaliser, donc c'est tout bénéf pour nous 😊

Petite précision quand même, c'est un détail mais je préfère que tout soit clair. Vous avez vu que j'ai enchaîné un `"padding"` (qui modifie toutes les marges intérieures) et un `"padding-left"` (qui modifie uniquement la marge intérieure gauche).

En fait c'est une technique pour gagner de la place. Il faut lire les lignes dans l'ordre :

1. Premièrement, on modifie toutes les marges et on leur met à toutes la valeur `"0px"` avec *padding*
2. Deuxièmement, on modifie la marge de gauche. Elle valait `0px`, désormais on change sa valeur et on lui met `20px` avec *padding-left* (on dit qu'on "écrase" l'ancienne valeur)

Si j'avais voulu, j'aurais pu faire à la place :

```
padding-top:0px;
padding-bottom:0px;
padding-right:0px;
padding-left:20px;
```

Mais ça aurait été plus long, et comme tout bon webmaster je suis une feignasse 🙄

## Le corps

Pour le corps, il va être important de définir de bonnes marges.

Le problème qu'on veut régler en priorité, c'est le texte du corps qui passe "sous" le menu quand il est trop long. Comme je vous l'ai expliqué, c'est un comportement normal car on a fait un *float:left* sur le menu.



Comment éviter que le texte passe sous le menu alors ?

Il suffit de modifier la marge "à gauche" du corps. C'est une marge extérieure, donc un *margin-left*.

On lui met une valeur suffisamment grande pour "pousser" le corps sur la droite, afin qu'il ne passe plus sous le menu (donc une valeur supérieure à la largeur du menu). Ici par exemple, on va mettre une valeur de `140px`. Comme par magie, vous allez voir, le corps sera correctement placé !

Ne nous arrêtons pas en si bon chemin 😊

On va rajouter un `margin-bottom` afin que le pied de page en-dessous ne soit pas trop "collé" au corps :

```
margin-bottom:20px;
```

On va aussi mettre un petit `padding` (marge intérieure) sur tous les côtés afin que le texte ne colle pas trop avec les bords du corps :

```
padding:5px;
```

On va aussi mettre une couleur de fond, une petite image de fond qui se répète horizontalement, une bordure... Bref, tout ça c'est de la déco 😊

### Code : CSS

```
body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
```

```
}

/* L'en-tête */

#en_tete
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}

/* Le menu */

#menu
{
    float: left;
    width: 120px;
}

.element_menu
{
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;

    margin-bottom: 20px;
}

/* Quelques effets sur les menus */

.element_menu h3
{
    color: #B3B3B3;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
    text-align: center;
}

.element_menu ul
{
    list-style-image: url("images/puce.png");
    padding: 0px;
    padding-left: 20px;
    margin: 0px;
    margin-bottom: 5px;
}

.element_menu a
{
    color: #B3B3B3;
}

.element_menu a:hover
{
    background-color: #B3B3B3;
    color: black;
}

/* Le corps de la page */

#corps
```

```

{
    margin-left: 140px; /* Une marge à gauche pour pousser le corps,
    afin qu'il ne passe plus sous le menu */
    margin-bottom: 20px; /* Ca c'est pour éviter que le corps colle
    trop au pied de page en-dessous */
    padding: 5px; /* Pour éviter que le texte à l'intérieur du corps
    ne colle trop à la bordure */

    color: #B3B3B3;
    background-color: #626262; /* Une couleur de fond pour le corps
    */
    background-image: url("images/motif.png");
    background-repeat: repeat-x; /* Une petite image de fond qui se
    répètera horizontalement en haut */

    border: 2px solid black; /* Une bordure pour bien marquer les
    limites du corps et pour faire joli */
}

```

Essayer !

Il nous reste quand même quelques petits trucs à décorer dans notre corps. Par exemple, les titres : on va changer leur police pour mettre une "Arial" plus jolie je trouve sur les titres.

On va aussi rajouter une image de fond sur les titres h2, avec *background-repeat:no-repeat*, ce qui signifie que le fond ne se répètera pas (il restera donc à gauche).

Pour éviter que le texte du titre ne s'écrive sur l'image de fond, on met un *padding-left* de quelques pixels. Comme ça, on aura une image d'engrenage devant chaque titre <h2> automatiquement !

#### Code : CSS

```

body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
}

/* L'en-tête */

#en_tete
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}

/* Le menu */

#menu
{
    float: left;
    width: 120px;
}

.element_menu
{
    background-color: #626262;
    background-image: url("images/motif.png");
}

```

```
background-repeat: repeat-x;

border: 2px solid black;

margin-bottom: 20px;
}

/* Quelques effets sur les menus */

.element_menu h3
{
    color: #B3B3B3;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
    text-align: center;
}

.element_menu ul
{
    list-style-image: url("images/puce.png");
    padding: 0px;
    padding-left: 20px;
    margin: 0px;
    margin-bottom: 5px;
}

.element_menu a
{
    color: #B3B3B3;
}

.element_menu a:hover
{
    background-color: #B3B3B3;
    color: black;
}

/* Le corps de la page */

#corps
{
    margin-left: 140px;
    margin-bottom: 20px;
    padding: 5px;

    color: #B3B3B3;
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;
}

#corps h1 /* Tous les titres h1 du corps */
{
    color: #B3B3B3;
    text-align: center;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
}

#corps h2 /* Tous les titres h2 du corps */
{
    height: 30px;
```

```
background-image: url("images/titre.png"); /* Une petite image de
fond sur les titres h2 */
background-repeat: no-repeat; /* L'image ne se répètera pas, elle
sera à gauche du titre */

padding-left: 30px;
color: #B3B3B3;
text-align: left;
}
```

[Essayer !](#)

Sympathique n'est-ce pas ? 😊

## Le pied de page (et on a fini !)

Allez, le dernier pour la route et c'est fini, promis ! 😊

Pour le pied de page, rien de compliqué ni d'extraordinaire, ce sont des modifications similaires au reste. On met une couleur de fond, on met une bordure, on vérifie si les marges nous plaisent... Notez que je fais un "clear: both" pour supprimer l'effet flottant du menu. Cela permet de s'assurer que le pied de page est bien sous le menu.

### Code : CSS

```
/*
Design d'exemple du Site du Zéro
Réalisé par zaz, venom et mateo21
<lien
url="http://www.siteduzero.com">http://www.siteduzero.com</lien>
*/

body
{
    width: 760px;
    margin: auto;
    margin-top: 20px;
    margin-bottom: 20px;
    background-image: url("images/fond.png");
}

/* L'en-tête */

#en_tete
{
    width: 760px;
    height: 100px;
    background-image: url("images/banniere.png");
    background-repeat: no-repeat;
    margin-bottom: 10px;
}

/* Le menu */

#menu
{
    float: left;
    width: 120px;
}
```

```
.element_menu
{
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;

    margin-bottom: 20px;
}

/* Quelques effets sur les menus */

.element_menu h3
{
    color: #B3B3B3;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
    text-align: center;
}

.element_menu ul
{
    list-style-image: url("images/puce.png");
    padding: 0px;
    padding-left: 20px;
    margin: 0px;
    margin-bottom: 5px;
}

.element_menu a
{
    color: #B3B3B3;
}

.element_menu a:hover
{
    background-color: #B3B3B3;
    color: black;
}

/* Le corps de la page */

#corps
{
    margin-left: 140px;
    margin-bottom: 20px;
    padding: 5px;

    color: #B3B3B3;
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;
}

#corps h1
{
    color: #B3B3B3;
    text-align: center;
    font-family: Arial, "Arial Black", "Times New Roman", Times,
serif;
}
```

```
#corps h2
{
    height: 30px;

    background-image: url("images/titre.png");
    background-repeat: no-repeat;

    padding-left: 30px;
    color: #B3B3B3;
    text-align: left;
}

/* Le pied de page (qui se trouve tout en bas, en général pour les
copyrights) */

#pied_de_page
{
    padding: 5px;
    clear: both;

    text-align: center;

    color: #B3B3B3;
    background-color: #626262;
    background-image: url("images/motif.png");
    background-repeat: repeat-x;

    border: 2px solid black;
}
```

[Essayer !](#)

C'était notre dernier CSS, le design est terminé ! (ouf 🤔)

Et voilà notre site fin prêt. Bon ce design "Mechanical Spirit" inventé pour l'occasion ne vous conviendra sûrement pas. Remarquez, c'est un peu fait exprès 🤖

Maintenant que le design est fini, vous savez comment procéder, quelles sont les grandes lignes à suivre. C'est à vous de créer le design entier de votre site, d'y mettre vos couleurs, vos images de fond, vos effets, vos polices... Bref, tout ça vous savez le faire maintenant, depuis le temps que je vous rabâche et re-rabâche du CSS 😄

Vous êtes autorisés (et invités) à vous inspirer de mon CSS. Je sais que ça ne coule pas de source quand on débute et qu'on fait son premier design, donc je comprends tout à fait que toutes ces lignes de code vous donnent la nausée.

La bonne nouvelle, c'est qu'avec de la pratique non seulement vous n'aurez plus la nausée, mais surtout vous saurez créer un design entier sans l'aide de personne ! 😊



## Attendez, c'est pas fini !

Vous ne croyiez quand même pas que j'allais vous lâcher aussi rapidement 😊

J'ai quelques informations "en vrac" à vous livrer. Ce ne sont pas des petits détails pour la plupart, et vous allez peut-être découvrir de nouvelles choses, donc soyez attentifs y'en a plus pour longtemps 😊

## Rappel : insérer un CSS dans le XHTML

Ça peut paraître idiot mais vous l'avez peut-être déjà oublié : pour que notre fichier XHTML s'affiche avec la présentation indiquée dans le CSS, il faut rajouter la ligne suivante entre <head> et </head> :

```
<link rel="stylesheet" media="screen" type="text/css" title="Exemple"
href="design_mecanique.css" />
```

Cette ligne indique que l'on utilise le fichier "design\_mecanique.css" pour la présentation.

## Changez le CSS, changez le design !

En modifiant le CSS, on peut changer tout le design sans toucher au code XHTML !

En fait, regardez à nouveau attentivement le code XHTML final qu'on a écrit : il ne contient aucune information sur le design. C'est tout juste s'il dit qu'il y a un en-tête, un menu qui contient tels liens, un corps qui contient tel texte etc. **Et c'est tout** !

Ca, c'est ce que je me tue à répéter depuis le premier chapitre de ce cours (et j'ose espérer que depuis le temps ça commence à rentrer 😊) : le XHTML c'est le contenu de votre page, le CSS c'est la présentation de votre page.

En gros, changer de CSS c'est comme changer de vêtement : on modifie son look (son apparence), mais la personne à l'intérieur est toujours la même 😊

Pour illustrer ce que je viens de vous dire, j'ai demandé à zaz et venom de me faire un second design sans toucher au code XHTML. Seul le CSS a été modifié, et regardez tout ce qu'on peut changer sur un site rien qu'à coup de CSS :

[Essayer !](#)

Petite précision : nous avons porté un peu moins d'attention sur ce second design, et si les couleurs rose-barbie vous donnent la nausée, dites-vous bien que nous aussi :). C'était juste pour illustrer que l'on peut faire un design très différent en modifiant le CSS.

Regardez la différence. Un seul fichier XHTML, 2 fichiers CSS, des résultats très différents :

*Avec design\_mecanique.css :*



Avec design\_poney\_rose.css



Regardez le code source XHTML, c'est le même que tout à l'heure. Avec le CSS, on a :

- Changé la couleur de fond de la page
- Changé l'image de fond de l'en-tête, donc on a changé la bannière
- Changé les bordures
- Changé le menu de position (il suffit de remplacer le *float:left* par un *float:right* !)
- etc...

## Laissez le visiteur choisir le design

Une dernière chose que je souhaite vous montrer : *les feuilles de styles alternatives*.

En fait, si vous avez fait plusieurs designs pour votre site (comme c'est le cas pour moi là), vous pouvez proposer aux visiteurs de changer d'un seul coup le design.

Pour cela, vous devez rajouter des balises `<link />` entre `<head>` et `</head>`.

En plus de la balise que je vous ai rappelé il y a 2 minutes (qui indiquera le design "par défaut" de votre site), vous pourriez rajouter cette ligne à la suite :

```
<link rel="alternate stylesheet" media="screen" type="text/css" title="Design rose pour les filles" href="design_poney_rose.css" />
```

Cette ligne indique qu'il existe une feuille de style alternative ("alternate stylesheet"). Le visiteur pourra choisir le design de votre site qui lui plaît.

Pour info, si vous utilisez Firefox il suffit d'aller dans le menu "Affichage > Style de la page" et de choisir le design que vous voulez.



Les sites qui proposent des feuilles de style alternatives sont plutôt rares. Beaucoup de webmasters ne savent même pas que ça existe, d'autres ont tout simplement un peu la flemme de réaliser plusieurs designs 😊

Vous pouvez mettre autant de feuilles de style alternatives que vous le désirez. Il vous faudra écrire une balise `<link rel="alternate stylesheet" />` par design différent que vous proposez.

Un exemple, et on s'arrêtera là :

### Code : HTML

```
<head>
  <title>Bienvenue sur mon site !</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

  <!-- Ci-dessous le design "par défaut" du site -->
  <link rel="stylesheet" media="screen" type="text/css"
title="Mécanique" href="design_mecanique.css" />
  <!-- Ci-dessous les designs alternatifs que vous proposez -->
  <link rel="alternate stylesheet" media="screen" type="text/css"
title="Rose pour les filles" href="design_poney_rose.css" />
  <link rel="alternate stylesheet" media="screen" type="text/css"
title="Sombre et terrifiant" href="sombre.css" />
  <link rel="alternate stylesheet" media="screen" type="text/css"
title="Aquatique" href="aquatique.css" />
</head>
```

Exceptionnellement, vous n'avez pas de Q.C.M. pour ce chapitre. En effet, le meilleur moyen de vérifier que vous avez compris était de suivre en même temps que moi et de faire vos propres petites manipulations.

Oui je sais, ça a l'air lourd tout ce bazar à première vue. J'espère vous avoir livré le maximum d'informations pour que vous compreniez bien la méthode.

Néanmoins, ne vous affolez pas en vous disant "Mince ! J'aurais jamais pu deviner tout ça !". Moi aussi c'est ce que je me suis dit quand j'ai appris à faire ça la première fois, mais avec un peu de pratique ça ira de plus en plus vite. Je reconnais qu'en XHTML / CSS la création d'un design est assez laborieuse, surtout au début. La bonne nouvelle, c'est qu'on ne fait pas de nouveau design tous les jours quand même 😊

---

## Les tableaux

Non, nous n'en avons toujours pas terminé avec le XHTML (et avec le CSS non plus d'ailleurs !)

Certes, je tiens à vous rassurer : nous avons déjà fait le plus gros et nous avons presque terminé 😊

Cependant, il nous reste encore quelques éléments à voir. Ce ne sont peut-être pas les plus importants, peut-être pas les plus utilisés, mais je mets ma main à couper que vous en aurez besoin tôt ou tard.

Parmi ces éléments que je veux vous faire découvrir avant que l'on se quitte (déjà ?! o\_O), il y a les tableaux. Je ne vous fais pas l'affront de vous en dessiner un, je suppose que tout le monde sait ce qu'est un tableau.

... Oh et puis si, je vais vous faire cet affront

Mesdames, mesdemoiselles, messieurs, voici un... tableau !

Nom	Age	Pays
Anne	27 ans	France
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
Ogasaku Nyagatosoka	18 ans	Japon

Bon, c'est un tableau assez primaire. Bien entendu, nous apprendrons à faire des tableaux plus complexes afin que vous puissiez réaliser tout ce dont vous avez besoin.

Encore une fois, Monsieur CSS sera de passage et vous permettra de rendre vos tableaux super classe en un clin d'oeil. Bonne nouvelle de ce côté : il n'y a pas vraiment de nouvelles propriétés CSS à apprendre, vous connaissez déjà pratiquement tout !



Un tableau se construit à l'aide de balises XHTML. Commençons par voir la structure de base...

---

## Un tableau simple

Première balise à connaître : `<table>` `</table>`. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau. Cette balise est de type block, donc il faut la mettre en dehors d'un paragraphe.

Exemple :

Code : HTML

```
<p>Ceci est un paragraphe avant le tableau.</p>

<table>
  <!-- Ici, on mettra le contenu du tableau -->
</table>

<p>Ceci est un paragraphe après le tableau.</p>
```



Bon, et que met-on à l'intérieur du tableau ?

Alors là, préparez-vous à recevoir une avalanche de nouvelles balises 🤖

Pour commencer en douceur, voici 2 nouvelles balises très importantes :

- `<tr>` `</tr>` : indique le début et la fin d'une ligne du tableau.
- `<td>` `</td>` : indique le début et la fin du contenu d'une cellule.

En XHTML, votre tableau se construit ligne par ligne. Dans chaque ligne (`<tr>`), on indique le contenu des différentes cellules (`<td>`).

Schématiquement, notre tableau de tout à l'heure se construit comme ça :

Nom	Age	Pays
Anne	27 ans	France
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
Ogasaku Nyagatosoka	18 ans	Japon

On a une balise de ligne (`<tr>`) qui "entoure" chacune des cellules (`<td>`)

Par exemple, si je veux faire un tableau à deux lignes, avec 3 cellules par lignes (donc 3 colonnes), je devrai taper ceci :

Code : HTML

```
<table>
  <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>
</table>
```

Essayer !



C'est un tableau ça ? 🤔

Le texte s'est écrit à la suite, et y'a même pas de bordures !

Oui, un tableau sans CSS c'est tout vide. Alors justement, rajouter des bordures c'est très simple, vous connaissez déjà le code CSS qu'il faut écrire !

#### Code : CSS

```
td /* Toutes les cellules des tableaux... */
{
  border: 1px solid black; /* ... auront une bordure de 1px */
}
```

Essayer !

Ce n'est pas encore aussi parfait que ce qu'on voudrait. En effet, on aimerait qu'il n'y ait qu'une seule bordure entre 2 cellules, or ce n'est pas le cas ici.

Heureusement, il existe une propriété CSS spécifique aux tableaux : *border-collapse*, qui signifie "coller les bordures entre elles".

Cette propriété peut prendre 2 valeurs :

- collapse : les bordures seront collées entre elles, c'est l'effet qu'on recherche.
- separate : les bordures seront dissociées (valeur par défaut)

#### Code : CSS

```
table
{
  border-collapse: collapse; /* Les bordures du tableau seront
collées (plus joli) */
}
td
{
  border: 1px solid black;
}
```

Essayer !

Nous verrons plus loin dans ce chapitre d'autres options CSS. Nous essaierons aussi d'égayer un peu les tableaux qui sont pour le moment un peu moches, je dois le reconnaître 🤔

## La ligne d'en-tête

Maintenant que l'on a ce qu'on voulait, on va rajouter la ligne d'en-tête du tableau. Dans notre exemple, les en-têtes sont "Nom", "Age" et "Pays".

La ligne d'en-tête se crée avec un `<tr>` comme on a fait jusqu'ici, mais les cellules à l'intérieur sont cette fois des `<th>` et non pas des `<td>` !

#### Code : HTML

```
<table>
  <tr>
    <th>Nom</th>
    <th>Age</th>
    <th>Pays</th>
  </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>
</table>
```

La ligne d'en-tête est très facile à reconnaître pour 2 raisons :

- Les cellules sont des `<th>` au lieu des `<td>` habituels.
- C'est la première ligne du tableau (c'est idiot, mais encore faut-il le préciser :p)

Comme le nom des cellules est un peu différent pour l'en-tête, il faut penser à mettre à jour le CSS pour lui dire d'appliquer une bordure sur les cellules normales ET sur l'en-tête :

#### Code : CSS

```
table
{
  border-collapse: collapse;
}
td, th /* Mettre une bordure sur les td ET les th */
{
  border: 1px solid black;
}
```

Essayer !

Et voilà le travail ! 😊

Comme vous pouvez le constater, votre navigateur a mis le texte des cellules d'en-tête en gras. C'est ce que font en général les navigateurs, mais si vous le désirez vous pouvez changer ça à coup de CSS : modifier la couleur de fond des cellules d'en-tête, leur police, leur bordure etc...

Nous verrons des exemples de CSS de tableaux un peu plus complets tout à l'heure.

## Titre du tableau

Normalement, tout tableau doit avoir un titre. Le titre permet de renseigner rapidement le visiteur sur le contenu du tableau. Dans notre exemple, on a une liste de personnes... oui mais alors ? Qu'est-ce que ça représente ? Sans titre de tableau, vous le voyez, on est un peu perdus 😞

Heureusement, il y a <caption> ! (ahlala, que ferait-on sans ! ^^)

Cette balise se met tout au début du tableau, juste avant l'en-tête. C'est elle qui indique le titre du tableau :

### Code : HTML

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

  <tr>
    <th>Nom</th>
    <th>Age</th>
    <th>Pays</th>
  </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>
</table>
```

[Essayer !](#)

C'est quand même plus clair ! 😊



## Des tableaux plus élaborés

Nous avons appris à faire des petits tableaux simples. Ces petits tableaux suffisent dans la plupart des cas, mais il arrivera que vous ayez besoin de faire des tableaux plus... complexes.

Nous allons voir 2 techniques particulières :

- Pour les gros tableaux, il est possible de les **diviser** en 3 parties :
  - En-tête
  - Corps du tableau
  - Pied de tableau
- Pour certains tableaux, il se peut que vous ayez besoin de **fusionner** des cellules entre elles.

Commençons par le premier point : vous avez un gros tableau, et vous devez le diviser en plusieurs parties.

### Diviser un gros tableau

Si votre tableau est assez gros, vous aurez tout intérêt à la découper en plusieurs parties. Pour cela, il existe des balises XHTML qui permettent de définir les 3 "zones" du tableau :

- L'en-tête (en haut) : il se définit avec les balises `<thead></thead>`
- Le corps (au centre) : il se définit avec les balises `<tbody></tbody>`
- Le pied du tableau (en bas) : il se définit avec les balises `<tfoot></tfoot>`

Que mettre dans le pied de tableau ? Généralement, si c'est un long tableau, vous recopiez les cellules d'en-tête. Ça permet de voir même en bas du tableau à quoi se rapporte chacune des colonnes.

Schématiquement, un tableau en 3 parties se découpe donc comme cela :

Passagers du vol 377				
En-tête du tableau	Nom	Age	Pays	<thead>
Corps du tableau	Carmen	33 ans	Espagne	<tbody>
	Michelle	26 ans	Etats-Unis	
	François	43 ans	France	
	Martine	34 ans	France	
	Jonathan	13 ans	Australie	
	Xu	19 ans	Chine	
Pied du tableau	Nom	Age	Pays	<tfoot>

La seule chose un peu déroutante, c'est qu'il faut mettre les balises dans l'ordre suivant :

1. `<thead>`
2. `<tfoot>`
3. `<tbody>`

On met donc dans le code d'abord la partie du haut, ensuite la partie du bas, et enfin la partie principale (tbody). Le navigateur se chargera d'afficher les éléments au bon endroit, ne vous inquiétez pas 😊

Voici donc le code à écrire pour faire le tableau en 3 parties :

**Code : HTML**

```

<table>
  <caption>Passagers du vol 377</caption>

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

  <tfoot> <!-- Pied de tableau -->
    <tr>
      <th>Nom</th>
      <th>Age</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>Etats-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>

```

Essayer !



Il n'est pas obligatoire d'utiliser ces 3 balises (thead, tbody, tfoot) sur son tableau. En fait, vous vous en servirez surtout si votre tableau est assez gros et que vous avez besoin de l'organiser plus clairement 😊

Pour les "petits" tableaux vous pouvez garder l'organisation qu'on a vu au début (un peu plus simple), il n'y a pas de problème.

## 3, 2, 1... Fusioooooon !

Sur certains tableaux complexes, vous aurez besoin de "fusionner" des cellules entre elles.  
Un exemple de fusion ? Regardez, ce tableau liste des films et indique à qui ils s'adressent :

Titre du film	Pour enfants ?	Pour adolescents ?
Massacre à la tronçonneuse	Non, trop violent	Oui
Les bisounours font du ski	Oui, adapté	Pas assez violent...
Lucky Luke, seul contre tous	Pour toute la famille !	

Pour le dernier film, vous voyez que les cellules ont été fusionnées : elles ne font plus qu'un. C'est exactement l'effet qu'on cherche à obtenir.

Pour effectuer une fusion, il faut rajouter un attribut à la balise `<td>`. Il faut savoir qu'il existe 2 types de fusion :

- **La fusion de colonnes** : c'est ce que je viens de faire sur cet exemple. La fusion s'effectue horizontalement. On utilisera l'attribut *colspan*.
- **La fusion de lignes** : là, deux lignes seront groupées entre elles. La fusion s'effectuera verticalement. On utilisera l'attribut *rowspan*.

Comme vous le savez, vous devez donner une valeur à l'attribut (que ce soit *colspan* ou *rowspan*).

Cette valeur, c'est le nombre de cellules à fusionner entre elles. Sur notre exemple, on a fusionné deux cellules : celle de la colonne "Pour enfants ?", et celle de "Pour adolescents ?". On devra donc écrire :

```
<td colspan="2">
```

Qui signifie : *"Cette cellule est la fusion de 2 cellules"*

Il est possible de fusionner plus de cellules à la fois (3, 4, 5... tant que vous voulez).

Voilà le code XHTML qui me permet de réaliser la fusion de tout à l'heure :

### Code : HTML

```
<table>
  <tr>
    <th>Titre du film</th>
    <th>Pour enfants ?</th>
    <th>Pour adolescents ?</th>
  </tr>
  <tr>
    <td>Massacre à la tronçonneuse</td>
    <td>Non, trop violent</td>
    <td>Oui</td>
  </tr>
  <tr>
    <td>Les bisounours font du ski</td>
    <td>Oui, adapté</td>
    <td>Pas assez violent...</td>
  </tr>
  <tr>
    <td>Lucky Luke, seul contre tous</td>
    <td colspan="2">Pour toute la famille !</td>
  </tr>
</table>
```

Essayer !



Je n'ai pas coloré le texte des cellules pour ne pas compliquer le code XHTML. Ceci étant, vous êtes assez grands pour le faire tous seuls maintenant : il suffit d'utiliser des *class*.

**Remarque importante :** vous voyez que la dernière ligne ne contient que 2 cellules au lieu de 3 (il n'y a que 2 balises <td>). C'est tout à fait normal, car j'ai fusionné les deux dernières cellules entre elles. Le <td colspan="2"> indique que cette cellule prend la place de 2 cellules à la fois. C'est logique, non ?



Et pour le rowspan, on fait comment ?

Ca complique un petit peu. Pour notre exemple, on va "inverser" l'ordre de notre tableau : au lieu de mettre les titres de films à gauche, on va les mettre en haut.

C'est une autre façon de voir le tableau : au lieu de le faire en hauteur, on peut le faire en longueur.

Du coup, le colspan n'est plus adapté, c'est un rowspan qu'il faut faire pour dire que Lucky Luke est pour toute la famille. Prenez le temps de lire et de comprendre cet exemple, ça vaut le coup d'oeil 😊

#### Code : HTML

```
<table>
  <tr>
    <th>Titre du film</th>
    <td>Massacre à la tronçonneuse</td>
    <td>Les bisounours font du ski</td>
    <td>Lucky Luke, seul contre tous</td>
  </tr>
  <tr>
    <th>Pour enfants ?</th>
    <td>Non, trop violent</td>
    <td>Oui, adapté</td>
    <td rowspan="2">Pour toute la famille !</td>
  </tr>
  <tr>
    <th>Pour adolescents ?</th>
    <td>Oui</td>
    <td>Pas assez violent...</td>
  </tr>
</table>
```

Essayer !

Ne mélangez pas tout quand même ! Si cet exemple vous embrouille, vous pouvez l'oublier pour le moment il n'est pas forcément "vital".

Vous pourrez toujours y revenir plus tard, quand vous vous serez déjà habitués avec l'autre façon de faire les tableaux. Il n'y a pas de honte à remettre un exemple pour plus tard 😊

Bon, il n'empêche que nos tableaux ne sont pas très "design" pour le moment. Comme promis, nous allons voir dans la dernière partie de ce chapitre comment embellir un tableau à grands coups de CSS 😊

## Et avec un peu de CSS...

Alors bonne nouvelle : vous connaissez déjà la plupart des propriétés CSS dont nous aurons besoin pour embellir le tableau. Quelques exemples :

- Pour modifier la bordure des cellules, il suffit d'utiliser *border*.
- Pour modifier la couleur de fond d'une cellule, on utilisera *background-color*.
- Pour modifier l'image de fond d'une cellule, on utilisera *background-image*.
- Mais on peut aussi modifier la taille (*font-size*) et la police (*font-family*) du texte des cellules d'en-tête, en appliquant les propriétés adaptées aux balises `<th>`.
- On peut aussi agrandir le tableau tout entier (*width*), le centrer (*margin:auto* car le tableau est un block).
- On peut centrer le texte à l'intérieur des cellules (*text-align:center*), modifier les marges intérieures des cellules (*padding*) pour aérer le tableau.

Bref, vous connaissez déjà toutes ces propriétés et je n'ai pas à vous les réapprendre, c'est une bonne chose 😊

Le tout est de penser à les utiliser, et à les utiliser sur les bonnes balises. Par exemple, si vous mettez une couleur de fond noire à la balise `<table>`, tout le tableau sera noir. Tandis que si vous mettez la couleur de fond sur les balises `<th>`, seules les cellules d'en-tête changeront !

## Rien qu'avec des propriétés CSS connues

Vous vous souvenez du tableau en 3 parties qu'on est parvenu à faire tout à l'heure ? Revoyons le code encore une fois :

### Code : HTML

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

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

  <tfoot> <!-- Pied de tableau -->
    <tr>
      <th>Nom</th>
      <th>Age</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>Etats-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>

```

Eh bien, rien qu'en modifiant le CSS avec des propriétés qu'on connaît, il va avoir d'un coup beaucoup plus d'allure !

#### Code : CSS

```

caption /* Titre du tableau */
{
    margin: auto; /* Centre le titre du tableau */
    font-family: Arial, Times, "Times New Roman", serif;
    font-weight: bold;
    font-size: 1.2em;
    color: #009900;
    margin-bottom: 20px; /* Pour éviter que le titre ne soit trop
collé au tableau en-dessous */
}

table /* Le tableau en lui-même */
{
    margin: auto; /* Centre le tableau */
    border: 4px outset green; /* Bordure du tableau avec effet 3D
(outset) */
    border-collapse: collapse; /* Colle les bordures entre elles */
}

th /* Les cellules d'en-tête */
{
    background-color: #006600;
    color: white;
    font-size: 1.1em;
    font-family: Arial, "Arial Black", Times, "Times New Roman",
serif;
}

td /* Les cellules normales */
{
    border: 1px solid black;
    font-family: "Comic Sans MS", "Trebuchet MS", Times, "Times New
Roman", serif;
    text-align: center; /* Tous les textes des cellules seront
centrés*/
    padding: 5px; /* Petite marge intérieure aux cellules pour éviter
que le texte touche les bordures */
}

```

[Essayer !](#)

C'était pas bien dur, avouez 😊

Sur les tableaux, plus que partout ailleurs, vous aurez intérêt à travailler votre CSS. En effet, avec et sans CSS, c'est le jour et la nuit.

Il suffit de comparer. Lequel de ces 2 tableaux préférez-vous ?

Passagers du vol 377

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
François	43 ans	France
Martine	34 ans	France
Jonathan	13 ans	Australie
Xu	19 ans	Chine

Nom Age Pays

Sans CSS

### Passagers du vol 377

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
François	43 ans	France
Martine	34 ans	France
Jonathan	13 ans	Australie
Xu	19 ans	Chine
Nom	Age	Pays

Avec CSS

Pour ma part, mon choix est fait 😊

Et si le design de mon tableau ne vous plaît pas, n'hésitez pas à créer votre propre CSS !

## Quelques nouvelles propriétés CSS

Il existe quelques nouvelles propriétés CSS qui sont propres aux tableaux.

Voici celles que je vous conseille de retenir :

- *caption-side* : indique où doit se trouver le titre du tableau. Cette propriété peut prendre les valeurs suivantes :
  - [top](#) : le titre sera placé en haut du tableau (par défaut).
  - [bottom](#) : le titre sera placé en bas du tableau.
  - [left](#) : le titre sera placé à gauche du tableau.
  - [right](#) : le titre sera placé à droite du tableau.
- *border-collapse* : on l'a déjà vue tout à l'heure, cette propriété indique que les bordures des cellules seront collées entre elles. On utilise souvent cette propriété car l'effet qu'elle procure est intéressant. Les valeurs possibles sont :
  - [separate](#) : les bordures seront séparées les unes par rapport aux autres (par défaut).
  - [collapse](#) : les bordures seront collées entre elles.
- *vertical-align* : alignement vertical du contenu d'une cellule. Si une cellule a une hauteur importante, il est possible de placer son contenu en haut, en bas ou au milieu de la cellule :
  - [top](#) : le contenu sera aligné en haut de la cellule
  - [middle](#) : le contenu sera aligné au milieu de la cellule
  - [bottom](#) : le contenu sera aligné en bas de la cellule

Comme on connaît déjà *border-collapse*, je vais me contenter de vous montrer un CSS de tableau qui utilise le *vertical-align* et le *caption-side*.

J'utilise les mêmes fichiers XHTML et CSS que tout à l'heure, mais je rajoute dans le CSS les deux propriétés qu'on vient d'apprendre :

#### Code : CSS

```
caption
{
    caption-side: bottom; /* Le titre sera placé en bas du tableau */
    margin: auto;
    font-family: Arial, Times, "Times New Roman", serif;
    font-weight: bold;
    font-size: 1.2em;
    color: #009900;
    margin-top: 20px; /* La marge doit se faire au-dessus et non en-
dessous maintenant */
}

table
{
    margin: auto;
    border: 4px outset green;
    border-collapse: collapse;
}

th
{
    background-color: #006600;
    color: white;
    font-size: 1.1em;
    font-family: Arial, "Arial Black", Times, "Times New Roman",
serif;
}

td
{
    height: 80px; /* J'agrandis la hauteur des cellules pour que l'on
puisse voir l'alignement vertical */
    vertical-align: bottom; /* Alignement vertical, le contenu des
cellules sera placé en bas */
    border: 1px solid black;
    font-family: "Comic Sans MS", "Trebuchet MS", Times, "Times New
Roman", serif;
    text-align: center;
    padding: 5px;
}
```

Essayer !



Les lignes qui ont changé sont commentées

Le titre est placé en bas du tableau grâce à *caption-side*.

Petit détail, j'ai modifié le *margin-bottom* de tout à l'heure en *margin-top* : eh oui, maintenant que le titre se trouve en bas du tableau, la marge s'effectue au-dessus du titre !

Pour pouvoir tester l'alignement vertical, j'ai dû modifier la hauteur des cellules. En effet, pour que l'alignement vertical se



voie, il faut que les cellules soient suffisamment grandes.

Ensuite, le *vertical-align* m'a permis de dire "Je veux que le texte soit placé en bas de chaque cellule" !

---

Ainsi s'achève notre tour d'horizon des tableaux. La façon de les créer n'est peut-être pas naturelle je reconnais, mais on s'y fait vite.

Du temps que vous ne partez pas en courant parce que vous voyez des noms de balises imprononçables (tr, td, th... :p), vous avez toutes les chances de vous y habituer en peu de temps.

Je vous conseille de surtout bien vérifier que vos balises s'ouvrent et se ferment dans le bon ordre, c'est très important. Ne mettez par exemple JAMAIS de balise <td> si elle n'est pas entourée d'une balise de ligne <tr>.

Enfin, je vous l'ai déjà dit et je vous le répète : un tableau sans CSS c'est... pas très esthétique 😊. Profitez-en donc pour mettre à l'épreuve vos connaissances en CSS, c'est l'occasion idéale !

---

## Les formulaires

Le temps passe, les amis. Plus on avance dans le cours, plus vous allez voir qu'on commence à atteindre les limites.



Comment ça ? On m'aurait menti ?!

Le XHTML et le CSS ont des limites ? On ne peut pas tout faire avec ?!

Ben quoi, j'ai rien dit moi 😊

Plus sérieusement, oui le XHTML et le CSS ont des limites. Vous pouvez déjà faire un bon site avec tout ce que je vous ai appris, mais imaginez qu'un jour vous vouliez faire un *SUPER MEGA site hyper-génial* ?

On y arrive. Ce que je vais vous apprendre, c'est toujours du XHTML, mais vous allez vous rendre compte par vous-mêmes que vous vous heurtez aux "limites" du langage.

En fin de compte, c'est bon signe. Ça veut peut-être dire que vous êtes devenus bons et que le XHTML ne vous suffit plus 😊

De quoi va-t-on parler dans ce chapitre ? A force de divaguer, j'allais oublier quel était le thème du jour !

Nous allons parler des **formulaires**. L'idée est simple : vous avez créé un site, et vous aimeriez par exemple que vos visiteurs puissent vous donner leur avis dessus, en cochant des cases, en entrant leurs commentaires, leurs suggestions....

Bienvenue dans le monde merveilleux des formulaires, un monde où les boutons, les cases à cocher et les listes déroulantes vivent en harmonie 😊 (enfin presque 😊)

---

## Créer un formulaire

Lorsqu'il vous prend subitement l'envie d'insérer un formulaire dans votre page XHTML, vous devez obligatoirement créer une balise `<form></form>`. C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

### Code : HTML

```
<p>Texte avant le formulaire</p>

<form>
  <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```



Notez qu'il faut obligatoirement mettre des balises de type block (comme `<p></p>`) à l'intérieur de votre formulaire si vous avez besoin d'écrire du texte dedans.

Donc ça, c'était la structure de base.

Maintenant restez attentifs, parce que ce que j'ai à vous dire n'est pas évident vu qu'on est à la limite du XHTML.

Il faut savoir qu'un formulaire est fait pour être envoyé. On va prendre un exemple pour que les choses soient claires.

Supposons que votre visiteur vienne de taper un commentaire dans votre formulaire, comme par exemple "Ouah ton site il déchiiiiire". Ce message doit être *envoyé* pour que vous puissiez le recevoir (logique non ?), et afin que vous sachiez ce que le visiteur pense de votre site.

Eh bien c'est là le problème, ou plutôt les problèmes que l'on va se poser :

- **Problème n°1** : comment envoyer le texte rentré par le visiteur ? Par quel moyen ?
- **Problème n°2** : une fois que les données ont été envoyées, comment les traiter ? Souhaitez-vous recevoir le message automatiquement par mail, ou préférez-vous qu'un programme se charge de l'enregistrer quelque part, puis de l'afficher sur une page visible par tout le monde ?

Cela revient d'ailleurs à faire un livre d'or pour votre site si vous avez bien suivi 😊

Vous devez indiquer 2 attributs à la balise `<form>` afin de donner les réponses à ces 2 problèmes :

- **method** : cet attribut indique par quel moyen les données vont être envoyées (**problème n°1**). Il existe 2 moyens pour envoyer des données sur le web :
  - **method="get"** : c'est une méthode en général assez peu adaptée, car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page (`http://...`), mais ce détail ne nous intéresse pas vraiment pour le moment. La plupart du temps, je vous recommande d'utiliser l'autre méthode : "post".
  - **method="post"** : c'est la méthode la plus utilisée pour les formulaires car on peut rentrer un grand nombre d'informations grâce à elle.

- **action** : c'est l'adresse de la page ou du programme qui va *traiter* les informations (**problème n°2**). Cette page se chargera de vous envoyer un mail avec le message si c'est ce que vous voulez, ou bien d'enregistrer le message avec tous les autres dans une base de données.

Cela ne peut pas se faire en XHTML / CSS, on utilisera en général un autre langage dont vous avez peut-être entendu parler : le **PHP**. On aura l'occasion d'y revenir par la suite, ne vous en faites pas 😊

On va donc maintenant compléter la balise `<form>` avec les 2 attributs qu'on vient de voir.

Pour *method*, vous l'aurez deviné je vais mettre la valeur "post".

Pour *action*, je vais taper le nom d'une page spéciale en PHP ("traitement.php"). C'est cette page qui sera appelée lorsque le

visiteur cliquera sur le bouton "Envoyer le formulaire".

**Code : HTML**

```
<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>
```

Pour le moment, on ne sait pas ce qu'il se passe à l'intérieur de la page "traitement.php" : je vous demande de me faire confiance et d'imaginer que cette page existe et fonctionne. Nous verrons par la suite comment cette page PHP fait pour analyser les données du formulaire, mais ce n'est pas notre priorité.

Notre priorité là, c'est de voir comment en XHTML / CSS on fait pour insérer des zones de texte, des boutons et des cases à cocher dans votre page web.

C'est ce qu'on va apprendre maintenant 😊

## Les zones de saisie

Bon, retour au concret 🤔

Nous allons voir quelles sont les balises XHTML qui nous permettent de rentrer du texte dans un formulaire.  
Pour commencer, il faut savoir qu'il y a 2 zones de texte différentes :

- **La zone de texte à une ligne** : comme son nom l'indique, on ne peut écrire qu'une seule ligne à l'intérieur :p. Elle sert pour rentrer des textes courts, comme par exemple : "Entrez votre pseudo"
- **La zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes, comme par exemple : "Faites une dissertation sur l'utilité du XHTML dans le développement des pays d'Asie du Sud-Est"

### Zone de texte à une ligne

Vous ne savez pas ce qu'est une zone de texte ?  
Ca tombe bien, j'en ai pris une en photo :

Votre pseudo :

Pour insérer une zone de texte à une ligne, on va utiliser la balise `<input />`.  
On la retrouvera plusieurs fois par la suite dans ce chapitre. A chaque fois, c'est son attribut `type` qui va changer.

Pour une zone de texte à une ligne, on doit taper :  
`<input type="text" />`

Mais ce n'est pas tout ! Il manque un attribut qui sera très important : c'est le nom de votre zone de texte. En effet, cela vous permettra plus tard (dans le langage PHP) de reconnaître que tel texte est le pseudo du visiteur, tel texte est son mot de passe etc...

Il faut donc donner un nom à cette zone de texte, grâce à l'attribut `name`. Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :  
`<input type="text" name="pseudo" />`

Allez, on va tester ça déjà :

#### Code : HTML

```
<form method="post" action="traitement.php">  
  <p><input type="text" name="pseudo" /></p>  
</form>
```

[Essayer !](#)



Pensez à entourer votre balise `<input />` par une balise block comme `<p></p>` car sinon votre page web ne sera pas valide (cela vous est expliqué dans l'annexe "Le W3C et les standards du web").

### Les labels

Cette zone de texte est bien jolie, mais si votre visiteur tombe dessus il ne sait pas trop ce qu'il doit mettre dedans. On va justement lui expliquer qu'il doit rentrer son pseudo.

Pour indiquer ce que signifie une zone de texte au visiteur, on utilise la balise <label> qui entoure le libellé, comme ceci :

**Code : HTML**

```
<form method="post" action="traitement.php">
  <p>
    <label>Votre pseudo</label> : <input type="text"
name="pseudo" />
  </p>
</form>
```

Mais ça ne suffit pas. Il faut lier le label avec la zone de texte.

Pour ce faire, il faut donner un nom à la zone de texte, non pas avec l'attribut name mais avec l'attribut id (que l'on peut utiliser sur toutes les balises).



Un name et un id sur le champ ? Ca ne va pas faire double emploi ça ?

Si, un peu. Personnellement, je donne la même valeur au name et à l'id, ça ne pose pas de problème.

Pour lier le label au champ, il faut lui donner un attribut for qui a la même valeur que l'id du champ... 🤔

En un mot comme en cent, ça donne ça :

**Code : HTML**

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo</label> : <input type="text"
name="pseudo" id="pseudo" />
  </p>
</form>
```

Essayer !

Essayez de cliquer sur le texte "Votre pseudo" : vous allez voir que le curseur se place automatiquement dans la zone de texte correspondante.

On a "lié" le label avec sa zone de texte pour qu'on sache à quoi il correspond.



Mais... Ca sert juste à ce que la zone de texte soit sélectionnée si on clique sur "Votre pseudo" ?

Non, justement 🤔

Cela permet aussi aux personnes atteintes d'un handicap de pouvoir se repérer plus facilement dans votre formulaire. On n'y pense pas assez souvent, mais parfois les formulaires sont tellement gros et mal construits qu'il est difficile de savoir à quoi se rapporte chaque zone de texte.

Ici, les non-voyants par exemple sauront que la zone de texte correspond au label "Votre pseudo" à coup sûr, et ils vous seront reconnaissants d'avoir pris le temps de rendre votre formulaire plus clair grâce aux labels.

## Quelques attributs supplémentaires

Il existe quelques autres attributs sur la balise <input /> qui vous intéresseront sûrement.

Il est ainsi possible, si vous en avez besoin, de donner une valeur par défaut à votre zone de texte.

Pour faire cela, il vous suffit d'ajouter l'attribut *value* à `<input />` en indiquant quelle valeur vous voulez mettre au départ.

Exemple :

```
<input type="text" name="pseudo" value="M@teo21" />
```

Autre chose : vous pouvez modifier la largeur de votre zone de texte ainsi que le nombre maximal de caractères que l'on peut mettre dedans.

La largeur se définit avec *size*.

Le nombre maximal de caractères se définit avec *maxlength*.

Dans l'exemple suivant, la zone de texte contient le pseudo "M@teo21" par défaut, elle fait 30 caractères de long mais on ne peut mettre que 10 caractères maximum à l'intérieur :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo :</label>
    <input type="text" name="pseudo" id="pseudo" value="M@teo21"
size="30" maxlength="10" />
  </p>
</form>
```

[Essayer !](#)

## Zone de mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une zone "mot de passe", c'est-à-dire une zone où on ne voit pas à l'écran ce qui est tapé dedans.

La seule chose que vous avez besoin de changer, c'est l'attribut *type* de `<input />`. Mettez *type="password"*, et le tour est joué !

Je complète mon formulaire. Il demande maintenant au visiteur son pseudo ET son mot de passe :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo :</label>
    <input type="text" name="pseudo" id="pseudo" value="M@teo21"
  />

  <br />
  <label for="pass">Votre mot de passe :</label>
  <input type="password" name="pass" id="pass" />

  </p>
</form>
```

[Essayer !](#)

## Zone de texte multiligne

On arrive (enfin) aux zones de texte multilignes. Rassurez-vous, ça va aller plus vite maintenant que vous connaissez les labels



Voici une zone de texte multiligne :

Comment pensez-vous que je pourrais améliorer mon site ?  
Difficile d'améliorer la perfection non ?  
Enfin, moi ce que j'en dis...  
  
A toi de voir !

La zone de texte multiligne est une balise existant par paire (contrairement à `<input />`). C'est la balise `<textarea></textarea>`. A elle aussi, comme à tout autre élément du formulaire, il faut lui donner un nom avec `name`, et utiliser un label qui explique de quoi il s'agit.

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="ameliorer">Comment pensez-vous que je pourrais
    améliorer mon site ?</label><br />
    <textarea name="ameliorer" id="ameliorer"></textarea>
  </p>
</form>
```

[Essayer !](#)

On peut modifier la taille du `textarea` de 2 façons différentes :

- **En CSS** : il suffit d'appliquer les propriétés CSS *width* et *height* au `textarea`.
- **Avec des attributs** : on peut ajouter les attributs `rows` et `cols` à la balise `<textarea>`. Le premier indique le nombre de lignes du `textarea`, et le second le nombre de colonnes.

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="ameliorer">Comment pensez-vous que je pourrais
    améliorer mon site ?</label><br />
    <textarea name="ameliorer" id="ameliorer" rows="10"
    cols="50"></textarea>
  </p>
</form>
```

[Essayer !](#)



Mais euh, au fait : pourquoi on ouvre `<textarea>` pour le fermer juste après ? Une balise simple comme `<input />`



aurait suffit non ?

En fait, cela sert à pré-remplir le textarea. C'est un peu ce qu'on faisait tout à l'heure avec *value*, sauf que là on peut le faire sur plusieurs lignes 😊

Vous pouvez donc mettre la valeur par défaut du textarea comme ceci :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="ameliorer">Comment pensez-vous que je puisse
    améliorer mon site ?</label><br />
    <textarea name="ameliorer" id="ameliorer" rows="10"
    cols="50">
      Améliorer ton site ?!
      Mais enfin ! Il est tellement génialissime qu'il n'y a pas
      besoin de l'améliorer !
    </textarea>
  </p>
</form>
```

[Essayer !](#)



## Eléments d'options

En plus des zones de saisies, le XHTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. On va voir dans cette partie :

- Les cases à cocher, que vous connaissez sûrement...
- Les zones d'options, que vous connaissez aussi...
- Les listes déroulantes, que vous avez déjà dû voir... Bon en fait vous connaissez déjà tout 😊

Ou plutôt : vous avez déjà vu tous ces éléments, mais je parie que vous ne savez pas comment on les crée en XHTML ! ... J'ai gagné ? 😊

Je suis décidément trop fort 😊

### Les cases à cocher

Ceci, mes amis, ce sont des cases à cocher :

☐ Frites  
☐ Steak haché  
☐ Epinards  
☐ Huitres

Bonne nouvelle : ça va aller vite 😊

En effet, la balise à utiliser vous la connaissez déjà : c'est `<input />`

On va seulement changer la valeur de son attribut *type* pour mettre "checkbox" :

```
<input type="checkbox" name="choix" />
```

Rajoutez un `<label>` bien placé, et le tour est (déjà ?) joué !

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    Cochez les aliments que vous aimez manger :<br />
    <input type="checkbox" name="frites" id="frites" /> <label
for="frites">Frites</label><br />
    <input type="checkbox" name="steak" id="steak" /> <label
for="steak">Steak haché</label><br />
    <input type="checkbox" name="epinards" id="epinards" />
<label for="epinards">Epinards</label><br />
    <input type="checkbox" name="huitres" id="huitres" /> <label
for="huitres">Huitres</label>
  </p>
</form>
```

[Essayer !](#)

Qu'est-ce que je peux rajouter ?

Grâce aux label, vous n'êtes pas obligés de cliquer directement sur la case, vous pouvez aussi cliquer sur le texte juste à côté (enfin, ça ne marche pas sur IE comme je vous l'ai dit...).

N'oubliez pas aussi de donner un nom à chaque case à cocher, cela vous permettra d'identifier plus tard quelles cases le visiteur a coché.

Ah si, j'allais oublier. Vous pouvez faire en sorte qu'une case soit déjà cochée par défaut. Pour faire cela, il faut rajouter l'attribut `checked="checked"` (oui oui, même attribut et valeur).

Cela nous permet d'influencer les choix dans notre exemple 😊

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    Cochez les aliments que vous aimez manger :<br />
    <input type="checkbox" name="frites" id="frites" /> <label
for="frites">Frites</label><br />
    <input type="checkbox" name="steak" id="steak" /> <label
for="steak">Steak haché</label><br />
    <input type="checkbox" name="epinards" id="epinards"
checked="checked" /> <label for="epinards">Epinards</label><br />
    <input type="checkbox" name="huitres" id="huitres" /> <label
for="huitres">Huitres</label>
  </p>
</form>
```

[Essayer !](#)

## Les zones d'options

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités :

- ☐ Moins de 15 ans
- ☐ 15-25 ans
- ☐ 25-40 ans

Ca ressemble aux cases à cocher, avec juste une petite difficulté supplémentaire, vous allez voir.

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

La grosse différence avec les cases à cocher, c'est que les zones d'options fonctionnent par "groupe". Tout un groupe d'options a le même nom, mais un attribut `value` différent à chaque fois.

Les choses seront plus claires sur l'exemple ci-dessous :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    Veuillez indiquer la tranche d'âge dans laquelle vous vous
situez :<br />
    <input type="radio" name="age" value="moins15" id="moins15"
/> <label for="moins15">Moins de 15 ans</label><br />
    <input type="radio" name="age" value="medium15-25"
id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />
    <input type="radio" name="age" value="medium25-40"
id="medium25-40" /> <label for="medium25-40">25-40 ans</label><br />
    <input type="radio" name="age" value="plus40" id="plus40" />
<label for="plus40">Encore plus vieux que ça ?!</label>
  </p>
</form>
```

[Essayer !](#)

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache dans quel "groupe" le bouton fait partie.

Essayez d'enlever les attributs *name*, vous verrez que chaque élément d'option deviendra sélectionnable. Or, ce n'est pas ce que l'on veut, c'est pour ça qu'on les "lie" entre eux en leur donnant un nom identique.



**Vous noterez que cette fois on a choisi un id différent de name. En effet, les name étant identiques, on n'aurait pas pu les différencier (et vous savez bien qu'un id doit être unique !). Voilà donc pourquoi on a choisi de mettre à l'id la même valeur que value.**

Si vous avez 2 zones d'options différentes, il faut donner un nom unique à chaque groupe comme ceci :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    Veuillez indiquer la tranche d'âge dans laquelle vous vous
    situez :<br />
    <input type="radio" name="age" value="moins15" id="moins15"
  /> <label for="moins15">Moins de 15 ans</label><br />
    <input type="radio" name="age" value="medium15-25"
  id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />
    <input type="radio" name="age" value="medium25-40"
  id="medium25-40" /> <label for="medium25-40">25-40 ans</label><br />
    <input type="radio" name="age" value="plus40" id="plus40" />
  <label for="plus40">Encore plus vieux que ça ?!</label>
  </p>
  <p>
    Sur quel continent habitez-vous ?<br />
    <input type="radio" name="continent" value="europe"
  id="europe" /> <label for="europe">Europe</label><br />
    <input type="radio" name="continent" value="afrique"
  id="afrique" /> <label for="afrique">Afrique</label><br />
    <input type="radio" name="continent" value="asie" id="asie"
  /> <label for="asie">Asie</label><br />
    <input type="radio" name="continent" value="amerique"
  id="amerique" /> <label for="amerique">Amérique</label><br />
    <input type="radio" name="continent" value="australie"
  id="australie" /> <label for="australie">Australie</label>
  </p>
</form>
```

[Essayer !](#)

Si, au lieu de mettre *name="continent"* on avait continué à mettre des *name="age"*, le visiteur n'aurait pas pu sélectionner son âge ET son continent.

Essayez de changer les noms, vous verrez bien ce qu'il se passe 😊

Dernière petite précision : si vous voulez qu'une des options soit cochée par défaut, vous rajoutez un *checked="checked"* comme pour les cases à cocher, et le tour est joué !

## Les listes déroulantes

Les listes déroulantes sont un autre moyen élégant de faire un choix entre plusieurs possibilités :

Dans quel pays habitez-vous ?



Cette fois, ça fonctionne un peu différemment. On va utiliser la balise `<select></select>` qui indique le début et la fin de la liste déroulante.

On ajoute l'attribut `name` à la balise pour donner un nom à la liste. Par exemple "pays" :

```
<select name="pays">
```

Et maintenant, à l'intérieur du `<select></select>`, on va mettre plusieurs balises `<option></option>` (une par choix possible). On rajoute un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

Embrouillé ?

Meuh non, c'est d'une simplicité enfantine !

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays">
      <option value="france">France</option>
      <option value="espagne">Espagne</option>
      <option value="italie">Italie</option>
      <option value="royaume-uni">Royaume-Uni</option>
      <option value="canada">Canada</option>
      <option value="etats-unis">Etats-Unis</option>
      <option value="chine">Chine</option>
      <option value="japon">Japon</option>
    </select>
  </p>
</form>
```

[Essayer !](#)

Le principe est un peu différent de ce qu'on a vu jusqu'ici, mais ça se comprend néanmoins très bien.

Autre nouveauté, on ne peut plus utiliser le `checked="checked"` ici, on doit utiliser à la place le... `selected="selected"`. Il nous permet comme le `checked` de sélectionner une valeur par défaut :

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays">
      <option value="france">France</option>
      <option value="espagne">Espagne</option>
      <option value="italie">Italie</option>
```

```
        <option value="royaume-uni">Royaume-Uni</option>
        <option value="canada"
selected="selected">Canada</option>
        <option value="etats-unis">Etats-Unis</option>
        <option value="chine">Chine</option>
        <option value="japon">Japon</option>
    </select>
</p>
</form>
```

[Essayer !](#)

Mais les listes d'options savent faire encore mieux !

On peut créer des **groupes d'options** à l'intérieur de la liste, grâce à la balise <optgroup></optgroup>. Vous devez lui ajouter l'attribut *label* qui permet de donner un nom au groupe (à ne pas confondre avec la balise <label> !).

Dans notre exemple, pourquoi ne pas séparer les pays en fonction de leur continent ?

#### Code : HTML

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

[Essayer !](#)

C'est assez pratique, surtout quand on a une graaaaaande liste déroulante 😊

## Un formulaire accessible et design ?

Maintenant, on va essayer d'aller encore plus loin.

Notre objectif sera double : faire en sorte que notre formulaire soit *accessible* (= compréhensible) et *design* (= pas trop moche :-°)

On va faire ça en 4 étapes :

1. Définir un ordre de tabulation (*accessibilité*)
2. Définir des touches de raccourci (*accessibilité*)
3. Organiser le formulaire en plusieurs zones (*accessibilité et design*)
4. Rajouter du CSS (*design*)

### Définir un ordre de tabulation

C'est le premier des points que nous allons voir censé nous faciliter la vie.

Comme vous le savez peut-être, on peut se déplacer dans un formulaire uniquement grâce à la touche "Tab" (tabulation) située à gauche de votre clavier. A chaque fois qu'on appuie sur Tab, on va au champ suivant. A chaque fois qu'on fait Maj + Tab, on retourne au champ précédent.

Le but est de dire en XHTML dans quel ordre on doit se déplacer dans le formulaire. Par exemple, après le champ "nom" si je tape Tab je dois tomber sur le champ "prénom", puis sur "e-mail" etc...

On va utiliser l'attribut *tabindex* qui peut se rajouter sur toutes les balises du formulaire qu'on a apprises.

On doit lui mettre un nombre pour valeur. Chaque champ du formulaire doit avoir un nombre différent.

Les nombres indiquent dans quel ordre on se déplace dans le formulaire : d'abord le n°1, puis le n°2, le n°3 etc...



Vous n'êtes pas obligés de mettre des nombres qui se suivent. Il est même conseillé de laisser des "espaces" entre les nombres au cas où vous auriez besoin de rajouter plus tard des champs.

Ainsi, il est tout à fait possible de compter 10 par 10 : n°10, n°20, n°30 etc... Ca ne coûte pas plus cher de compter de 10 en 10, et si plus tard on a besoin de créer un champ n°25, on n'aura aucun problème 😊

Sur ce formulaire, j'ai rajouté les *tabindex* à chaque champ. Comme on l'a vu, le premier champ est celui qui a le numéro le plus petit, et le dernier celui qui a le plus grand.

#### Code : HTML

```
<form method="post" action="traitement.php">
  <p>
    <label for="nom">Quel est votre nom ?</label><br />
    <input type="text" name="nom" id="nom" tabindex="10" /><br />

    <label for="prenom">Quel est votre prénom ?</label><br />
    <input type="text" name="prenom" id="prenom" tabindex="20"
  /><br />

    <label for="email">Quel est votre e-mail ?</label><br />
    <input type="text" name="email" id="email" tabindex="30"
  /><br />

    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays" tabindex="40">
      <optgroup label="Europe">
        <option value="france">France</option>
        <option value="espagne">Espagne</option>
        <option value="italie">Italie</option>
        <option value="royaume-uni">Royaume-Uni</option>
      </optgroup>
    </select>
  </form>
```

```

<optgroup label="Amérique">
  <option value="canada">Canada</option>
  <option value="etats-unis">Etats-Unis</option>
</optgroup>
<optgroup label="Asie">
  <option value="chine">Chine</option>
  <option value="japon">Japon</option>
</optgroup>
</select>
</p>
</form>

```

Essayer !

Essayez de taper plusieurs fois d'affilée sur "Tab", vous allez voir que vous vous déplacerez dans l'ordre que vous avez défini avec *tabindex*.

Cela est particulièrement utile pour les personnes qui ne peuvent pas se servir d'une souris (eh oui, ça existe !).



Par défaut, si aucun *tabindex* n'est mis, le navigateur dira que le premier champ est celui tout en haut, et que le dernier celui tout en bas de la page.

Cependant, je vous conseille de toujours mettre vous-même les *tabindex*, car si votre formulaire se complexifie par la suite cela sera très utile.

## Définir des touches de raccourci

Une touche de raccourci ("access key" en anglais) est une touche qui permet d'accéder directement à un champ de votre formulaire sans avoir à cliquer dessus avec la souris et sans avoir à appuyer plusieurs fois sur "Tab" comme un forcené avant de tomber sur le champ qui vous intéresse.

Ce qui est très pratique, c'est que le XHTML vous permet de choisir quelles touches du clavier serviront de raccourcis. Ce qui l'est moins, c'est que les raccourcis s'utilisent de manière différente en fonction du navigateur :

- [Firefox et Internet Explorer](#) (Windows) : il faut faire la combinaison de touches Alt + Raccourci. Si ça ne marche pas, essayez Alt + Maj + Raccourci.
- [Safari et IE-Mac \(Macintosh\)](#) : il faut taper Ctrl + Raccourci.

Pour définir une touche de raccourci, vous utiliserez l'attribut *accesskey* qui, comme *tabindex*, peut se mettre sur tous les types de champs de formulaire qu'on a vus.

Vous devez lui mettre comme valeur la touche du clavier qui doit servir de raccourci pour le champ.

Sur cet exemple, le champ de recherche est accessible directement avec la touche R :

Code : HTML

```

<form method="post" action="traitement.php">
  <p>
    <label for="nom">Quel est votre nom ?</label><br />
    <input type="text" name="nom" id="nom" tabindex="10" /><br />

    <label for="prenom">Quel est votre prénom ?</label><br />
    <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

    <label for="email">Quel est votre e-mail ?</label><br />
    <input type="text" name="email" id="email" tabindex="30" /><br />
  </p>
</form>

```

```
<label for="recherche">Que recherchez-vous sur ce site ?
<em>(raccourci : R)</em></label><br />
<input type="text" name="recherche" id="recherche"
tabindex="40" accesskey="R" />
</p>
</form>
```

Essayer !

Sous Windows, il faut donc faire Alt + R pour arriver directement sur le champ de recherche.  
Sous Mac, il faut faire Ctrl + R.



Le gros problème des touches de raccourci est que certains caractères sont déjà utilisés par le navigateur. Si vous utilisez les mêmes, il y aura un conflit et vos visiteurs ne pourront plus utiliser les raccourcis auxquels ils sont habitués.

L'idéal est d'utiliser des chiffres en raccourci, ils sont en général très peu utilisés par les navigateurs.

N'oubliez pas d'indiquer quelque part sur la page quels sont les raccourcis utilisables, parce que vos visiteurs ne pourront pas les deviner.

Enfin bon, si, ils peuvent regarder le code source de votre page pour repérer les attributs *accesskey*, mais j'ai déjà connu plus pratique personnellement 🤔

## Organiser le formulaire en plusieurs zones

La technique que nous allons voir a 2 avantages :

- Elle permet de rendre le formulaire plus clair, donc plus accessible.
- Elle permet d'améliorer le design de votre formulaire.

Concrètement, quelle est l'idée ?

Si vous avez un formulaire assez gros (et en général ça sera le cas), il est facile que le visiteur se perde dans la masse d'informations qu'il a à entrer. Il est possible en XHTML de grouper plusieurs champs ayant un thème entre eux.

On utilisera la balise `<fieldset></fieldset>` pour délimiter un groupe de champs.

A l'intérieur de cette balise, vous mettrez vos champs (vos `<input />` entre autres...) ainsi qu'une autre balise : `<legend></legend>`. Celle-ci permet de donner le nom du groupe.



Ca ressemble étrangement aux `<optgroup>` qu'on vient de voir ça... Mais pourquoi tout à l'heure on a utilisé un attribut (label) pour donner un titre au groupe, et là on utilise une balise spéciale `<legend>` ? 🤔

Alors ça, ça fait partie des grands mystères du XHTML. Je n'en sais strictement rien, et pour ma part j'aurais préféré qu'on se mette d'accord : soit on utilise des attributs pour donner un titre, soit on utilise des balises.

Là, ça sera à vous de vous en souvenir. Comme d'hab, ça vient avec la pratique... enfin, essayez de le retenir rapidement quand même, parce qu'on m'annonce que ça pourrait être une question de QCM 🤔

Allez, trêve de bavardages, voici un exemple concret d'utilisation des `<fieldset>` :

Code : HTML

```
<form method="post" action="traitement.php">
  <fieldset>
```



```

<legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

<label for="nom">Quel est votre nom ?</label><br />
<input type="text" name="nom" id="nom" tabindex="10" /><br />

<label for="prenom">Quel est votre prénom ?</label><br />
<input type="text" name="prenom" id="prenom" tabindex="20"
/><br />

<label for="email">Quel est votre e-mail ?</label><br />
<input type="text" name="email" id="email" tabindex="30"
/><br />
</fieldset>

<fieldset>
  <legend>Votre souhait</legend> <!-- Titre du fieldset -->

  <p>
    Faites un souhait que vous voudriez voir exaucé :<br />
    <input type="radio" name="souhait" value="riche"
id="riche" tabindex="40" /> <label for="riche">Etre riche</label><br
/>

    <input type="radio" name="souhait" value="celebre"
id="celebre" tabindex="50" /> <label for="celebre">Etre
célèbre</label><br />
    <input type="radio" name="souhait" value="intelligent"
id="intelligent" tabindex="60" /> <label for="intelligent">Etre
<strong>encore</strong> plus intelligent</label><br />
    <input type="radio" name="souhait" value="autre"
id="autre" tabindex="70" /> <label for="autre">Autre...</label><br
/>

  </p>

  <p>
    <label for="precisions">Si "Autre", veuillez préciser
: </label><br />
    <textarea name="precisions" id="precisions" cols="40"
rows="4" tabindex="80"></textarea>
  </p>
</fieldset>
</form>

```

Essayer !



A l'intérieur des <fieldset></fieldset>, l'utilisation de balises de paragraphe <p></p> n'est plus obligatoire comme c'était le cas tout à l'heure.

Le résultat est de toute évidence plus clair : on voit de suite quelles sont les grandes parties du formulaire. On y gagne forcément en clarté, et le visiteur vous en sera reconnaissant.

Bien sûr, ce formulaire ne serait pas la perfection ultime (hum hum) si on n'y rajoutait pas une petite couche de peinture en CSS 😊

## Gouache time !

La bonne nouvelle maintenant, c'est qu'après avoir appris un nombre incalculable de nouvelles balises XHTML, vous n'allez pas apprendre de nouvelles propriétés CSS (pfiou :D)

Comme vous connaissez déjà tout, je vais vous faire l'affront de vous donner un code CSS sans explications préalables. Na !

## Code : CSS

```
input, textarea
{
    font-family: "Times New Roman", Times, serif; /* On modifie la
    police du texte tapé l'intérieur des champs */
}

input:focus, textarea:focus /* Quand le curseur est sur un champ */
{
    background-color: #FFFF99;
}

label
{
    color: blue; /* Colorer en bleu tous les labels (bah oui,
    pourquoi pas en bleu ?) */
}

legend /* On met un peu plus en valeur les titres des fieldset */
{
    font-family: Arial, "Arial Black", Georgia, "Times New Roman",
    Times, serif;
    color: #FF9933;
    font-weight: bold;
}

fieldset
{
    margin-bottom: 15px; /* Une marge pour séparer les fieldset */
    background-color: #FFFCC;
}
```

[Essayer !](#)

Bon, c'est moche, mais c'est parce que c'est moi qui l'ai fait tout seul 😅

L'idée, c'est surtout de vous montrer que vous en savez désormais assez en CSS pour créer n'importe quel habillage de formulaire. Vous avez tous les outils en main, vous n'avez plus qu'à improviser ! (enfin pas trop quand même 😊)

On a réutilisé pour l'occasion le pseudo-format ":focus" qui, pour ceux qui s'en souviennent encore, permet d'appliquer des styles CSS lorsqu'un objet est sélectionné. En l'occurrence, on s'en sert pour mettre un fond jaune aux champs lorsqu'on clique dessus, ce qui donne un effet assez intéressant 😊

## Y'a plus qu'à appuyer sur le bouton !

Nous y sommes presque.

Nous avons vu la plupart des éléments que l'on peut intégrer dans un formulaire, mais il manque le plus important : le bouton de validation !

Heureusement, c'est très facile à créer, d'autant plus que vous connaissez déjà la balise...



Quoi ? C'est pas encore `<input />` dis-moi ?

Si

Décidemment, c'est la balise à tout faire !

Bon, bien sûr un seul type de bouton ne suffisait pas aux webmasters (faut pas rêver non plus), il a fallu en créer 3 :

- **Le bouton d'envoi** : il déclenche l'envoi du formulaire. Le visiteur se retrouve automatiquement télétransporté à la page indiquée dans l'attribut `action` du formulaire (on l'a vu au début de ce chapitre).

Un bouton d'envoi se crée avec l'attribut `type="submit"`. Vous pouvez lui ajouter un attribut `value` pour changer le texte à l'intérieur du bouton, mais vous pouvez laisser la valeur par défaut, c'est aussi clair :

```
<input type="submit" />
```

- **Le bouton de remise à zéro** : il remet à zéro automatiquement toutes les valeurs du formulaire. On doit utiliser cette fois l'attribut `type="reset"`

```
<input type="reset" />
```

- **Le bouton qui sert à rien** : c'est un bouton "générique" qui n'effectue aucune action particulière. Le formulaire n'est pas envoyé, il n'est pas remis à zéro, non rien ne se passe.

Quel intérêt ? Ca vous servira principalement à lancer des scripts en Javascript (un autre langage qu'on peut utiliser sur une page web, oui je sais ça fait beaucoup de langages ). Nous, on ne s'en servira pas, mais je vous en parle pour que vous sachiez que ça existe au cas où vous en auriez besoin un jour.

Cette fois, je vous recommande de mettre l'attribut `value` pour que l'on sache à quoi sert le bouton :

```
<input type="button" value="Je sers à rien" />
```

On va tester les 2 premiers boutons (envoi et remise à zéro) dans un petit formulaire fictif :

**Code : HTML**

```
<form method="post" action="cible_formulaire.php">

  <fieldset>
    <legend>Vos coordonnées</legend>

    <label for="nom">Quel est votre nom ?</label><br />
    <input type="text" name="nom" id="nom" tabindex="10" /><br />

    <label for="prenom">Quel est votre prénom ?</label><br />
    <input type="text" name="prenom" id="prenom" tabindex="20"
  /><br />

    <label for="email">Quel est votre e-mail ?</label><br />
    <input type="text" name="email" id="email" tabindex="30"
  /><br />
  </fieldset>

  <fieldset>
    <legend>Votre souhait</legend>

    <p>
      Faites un souhait que vous voudriez voir exaucé :<br />
      <input type="radio" name="souhait" value="riche"
      id="riche" tabindex="40" /> <label for="riche">Etre riche</label><br />
      <input type="radio" name="souhait" value="celebre"
      id="celebre" tabindex="50" /> <label for="celebre">Etre
      célèbre</label><br />
    </p>
  </fieldset>
</form>
```

```

        <input type="radio" name="souhait" value="intelligent"
id="intelligent" tabindex="60" /> <label for="intelligent">Etre
<strong>encore</strong> plus intelligent</label><br />
        <input type="radio" name="souhait" value="autre"
id="autre" tabindex="70" /> <label for="autre">Autre...</label><br
/>
    </p>

    <p>
        <label for="precisions">Si "Autre", veuillez préciser
:</label><br />
        <textarea name="precisions" id="precisions" cols="40"
rows="4" tabindex="80"></textarea>
    </p>
</fieldset>

<p>
    <input type="submit" /> <input type="reset" />
</p>

</form>

```

Essayer !



Dans le cas présent, le formulaire ne fait strictement rien. Rassurez-vous donc, aucune information n'a été enregistrée

Lorsque vous cliquez sur "Envoyer", le formulaire vous amène donc à une page "cible\_formulaire.php", qui est une page fictive en PHP que vous ne savez pas faire.

Comme je vous l'ai dit, c'est un peu là la limite : on sait construire un formulaire en XHTML / CSS, mais pour traiter les données (les enregistrer ou les envoyer par mail) on est obligés de passer par le langage PHP... dont nous allons parler, pas d'inquiétude

Ladies and gentlemen, j'ai l'honneur de vous annoncer que vous venez de lire un chapitre entier sur les formulaires et que vous ne savez toujours pas vous en servir.

Quoi, pourquoi vous me regardez comme ça ?... 🤔  
 ... NooOOooOOoonnnn, pas les tomaaaates !!! 🤔

Avant de me lyncher sur la place publique, écoutez ce que j'ai à vous dire : vous avez ter-mi-né d'apprendre le XHTML et le CSS ! 😊

Oui oui, vous avez bien entendu !

Mais, comme j'ai (malheureusement) une conscience, je m'en voudrais de vous lâcher comme ça dans la nature. Je vais donc me farcir l'écriture d'un chapitre supplémentaire, qui sera un chapitre de **conclusion** 😊

Comme dans toute dissertation qui se respecte :

- Nous ferons le bilan de ce que nous avons appris (et de ce que nous n'avons pas appris)
- Et l'*ouverture* sur ce mystérieux langage "PHP" qui est, paraît-il, censé pouvoir traiter les informations du formulaire (et bien d'autres choses encore !)

## Et maintenant, on fait quoi ? (Conclusion)

A chaque fois c'est la même chose. Vous arrivez la première fois devant le sommaire du tutoriel, vous regardez la liste des

chapitres et vous vous dites : *"Quoi ?! Il faut apprendre TOUT ça pour faire un site web ? "*

Et mine de rien, vous finissez par lire tous les chapitres un à un et... vous arrivez forcément à un moment ou à un autre sur une page appelée "Conclusion".

Et là, c'est marrant parce que systématiquement le commentaire que j'entendrai c'est *"Ah bon, ça y est c'est fini ?"*

Je suppose que la question que vous vous posez en réalité est : *"A-t-on tout appris ?"*, mais aussi : *"Qu'est-ce que je vais faire maintenant, je vais m'ennuyer !"*

Mais non ! Tonton Mateo a pensé à tout.

Ben oui quoi, je m'en serais voulu de vous lâcher dans la nature sans vous avoir mis sur le dos un **autre** tutoriel bien mastoc avec tout plein de chapitres et de mots barbares à retenir 😊😊😊

Je dois vraiment être un gros sadique... 😏

---

## L'heure du bilan a sonné

Je pense que la première chose qu'il faut faire, c'est le bilan. C'est-à-dire : résumer ce qu'on a appris en faisant quelques commentaires dessus maintenant qu'on commence à avoir un peu de recul.

Et puis, ce bilan sera l'occasion de constater aussi qu'il y a des choses que l'on n'a pas vues.

Pourquoi ?

De quoi s'agit-il ?

Comment en savoir plus ?

Tant de questions existentielles auxquelles je vais essayer de répondre 😊

## Résumons ce qu'on a vu

Vous devez vous souvenir que le cours a été séparé en 3 parties :

1. **Partie I : les bases du XHTML.** Après une brève introduction sur les logiciels à utiliser (Bloc-Notes, Notepad++) et sur les navigateurs (Internet Explorer, Firefox etc...), nous avons commencé à réaliser nos premières pages web entièrement en XHTML.  
Nous avons appris quelle était la structure de base d'une page XHTML, comment écrire du texte dedans, comment faire des liens et enfin comment insérer des images.

Site du Zér0



Site réservé aux débutants

*Eh Simone, viens voir ! J'ai réussi à créer un lien !*

Dans cette partie, j'ai volontairement évité de vous parler du langage CSS. Certes, les pages web que nous faisons étaient sacrément moches, mais cela vous a permis d'apprendre un seul langage à la fois pour commencer.

2. **Partie II : c'est plus joli avec du CSS !** Au bout d'un moment, il a fallu commencer à parler de CSS, c'était inévitable. Lorsque j'ai estimé que vous aviez assez de connaissances en XHTML, nous avons attaqué le CSS. Les chapitres étaient assez gros, je le reconnais, et le challenge était de taille. En effet, une fois les principes de base du CSS acquis, il vous fallait tout simplement... apprendre des tas et des tas de propriétés CSS. D'où l'impression de lourdeur que vous avez pu ressentir lors de la partie II.

## De toutes les couleurs

Salut et bienvenue dans cette page haute en couleurs ! J'utilise des **noms de couleurs standards** dans mon CSS pour égayer un peu la page. Ainsi, "red" signifie "rouge", "blue" signifie "bleu" etc.

*Avec le CSS, fini les sites en Noir & Blanc !*

J'ai fait de mon mieux pour éviter que ces chapitres ne soient trop répétitifs, mais c'est le langage qui est construit comme ça, je n'ai rien inventé. Toutefois, cela aura eu un aspect positif : à force de vous bourrer le crâne, il y a bien des choses qui auront réussi à rentrer 😊

Et ça c'était déjà pas mal 😊

3. **Partie III : XHTML & CSS, toujours plus forts !** Il a fallu ensuite passer à la vitesse supérieure. Dans cette partie, XHTML & CSS étaient tous deux omniprésents, et le moins que l'on puisse dire est qu'il a fallu s'accrocher. Bon d'accord, le premier chapitre sur les listes à puces n'était qu'une petite entrée en matière pour que vous voyiez de quelle manière XHTML & CSS se complètent sur une page web. Mais ensuite sont venus les chapitres sur les blocks, le positionnement CSS... et là c'était le vrai "point chaud" du tutoriel. C'était délicat et pourtant indispensable si vous vouliez ensuite créer le design de votre page web.



*Voilà mon site fin prêt à conquérir le Monde !*

Après avoir vu le fonctionnement des tableaux, nous nous sommes finalement arrêtés sur les formulaires. Un chapitre qui, vous vous en serez doutés, n'a pas été placé là par hasard. C'était en effet un chapitre frustrant car vous ne pouviez que créer le formulaire, mais vous n'étiez pas (et n'êtes toujours pas) capables de récupérer les informations entrées par le visiteur.

Je vous ai promis de vous en dire plus sur le langage PHP, grâce auquel vous pourrez récupérer ces informations du formulaire. Nous allons en parler un peu plus loin, justement.



Euh au fait, y a-t-il des balises XHTML et des propriétés CSS que nous n'avons pas vues ?

La réponse est bien entendu... oui.  
Voyons voir de quoi il s'agit.

## Ce qu'on n'a pas vu

Tout d'abord, je tiens à clarifier une chose : s'il y a des éléments dont je n'ai pas parlé, c'est principalement parce qu'ils ne rentraient dans aucun des thèmes des chapitres. Et comme j'aurais eu horreur de faire un chapitre "Autres balises" qui soit bordélique, j'ai préféré éviter le sujet.

Par souci d'honnêteté, je tiens quand même à parler de ce qu'on n'a pas vu 😊

Il y a donc des balises que vous ne connaissez pas. Cela vous pénalise-t-il ?

Pas franchement. Tenez par exemple, on n'a pas parlé de la balise `<hr />`. C'est une balise qui sert à créer une ligne de séparation sur la page. L'explication tient en une phrase, il n'y a rien d'autre à ajouter, difficile de créer un chapitre là-dessus



Mais alors, comment je peux avoir la liste des balises XHTML qui existent pour compléter mes connaissances ?

Tout simplement en lisant l'annexe "Liste des balises XHTML" de ce tutoriel. Vous trouverez le lien parmi les autres annexes

en bas du sommaire du cours.

Il y a franchement très peu de balises que nous n'avons pas vues. Ceci étant, je vous conseille de vous assurer de bien comprendre déjà tout le reste du cours avant d'aller ajouter de nouvelles balises à votre petite tête déjà bien remplie.

Pour le CSS, les choses sont différentes. Il y a beaucoup de propriétés, et j'ai fait l'effort, comme je vous l'ai dit, de vous en faire voir un gros paquet.

Pourtant, il en reste quelques-unes dont nous n'avons pas parlé. Pour la plupart, c'est parce que leur utilisation est spécifique : il en existe par exemple qui servent à indiquer la prononciation, d'autres à donner des instructions à l'imprimante etc...

Enfin, pour les autres, j'en ai parfois sauté dans un souci de simplification (la simplification, toujours la simplification ;))

Mais là, encore une fois si vous avez envie de tout connaître, je vous invite à consulter l'annexe "*Liste des propriétés CSS*"

Pour ce qui est du CSS, il faut savoir qu'il existe de très nombreuses astuces qui permettent de réaliser des effets précis. Ces astuces ne peuvent pas rentrer dans le cours car elles sont là aussi très spécifiques. Certaines vous permettent de créer de jolis menus, d'autres d'éviter d'avoir des bugs sous Internet Explorer etc...

Pour obtenir ces informations, il faudra partir à la pêche. Il existe de nombreux sites de qualité qui parlent du CSS plus en détail qu'ici (le seul souci, c'est qu'ils sont rarement faits pour les débutants :p). Je vous recommande chaudement ces deux-là, en français :

- [Alsacrérations](#)
- [OpenWeb](#)

Si vous voulez plus d'adresses, n'hésitez pas à en demander sur les forums 😊

---



## On va quand même pas s'arrêter là ?

Je vous avais dit qu'on allait reparler du PHP, ce moment est arrivé.

Plus généralement, nous allons voir s'il n'y a pas d'autres langages que vous pourriez apprendre qui soient en rapport avec XHTML & CSS.

Ces langages peuvent être divisés en 2 catégories :

- **Les langages "client"** : ce sont des langages exécutés par les ordinateurs de vos visiteurs. En gros, ce seront les ordinateurs de vos visiteurs qui feront les calculs pour réaliser certains effets. Parmi les langages client, on connaît surtout le **Javascript**, dont vous avez très probablement entendu parler car il est très populaire. Je vous ferai un petit speech dessus un peu plus bas.
- **Les langages "serveur"** : là, c'est un peu plus sérieux. Ce sont des langages qui s'exécutent sur une machine appelée le "serveur". C'est une sorte de gros ordinateur que vous n'avez jamais vu de votre vie et dont le seul travail consiste... à distribuer votre site web aux visiteurs.

Les langages serveur sont plus complexes, mais aussi beaucoup beaucoup plus puissants que les langages client.

Il existe plusieurs langages serveur :

- **Le PHP** : très populaire, vous en avez peut-être entendu parler... Ah oui je suis bête, c'est moi qui vous en parle depuis tout à l'heure sans vous dire ce que c'est 😊  
Ce qu'il faut retenir pour le moment, c'est que le PHP est très populaire chez les webmasters qui ont un peu d'expérience car c'est un langage gratuit et puissant.
- **L'ASP** : ce langage est moins courant et se fait voler la vedette par PHP. L'ASP a été développé par une entreprise que vous connaissez bien : Microsoft. Et comme Microsoft fait rarement les choses gratuitement... 🤨  
L'ASP a donc moins de succès que le PHP parce qu'il coûte cher, et je suis sûr que vous n'avez pas envie de payer pour créer votre site web (c'est un sentiment tout à fait humain, il ne faut pas en avoir honte 😊)  
Au final, ce sont surtout les entreprises qui utilisent l'ASP. Et encore... beaucoup d'entre elles se mettent au PHP, qui est globalement aussi puissant que l'ASP en plus d'être gratuit.

Il existe d'autres langages côté serveur, comme le **Perl**, mais comme je ne les connais pas vraiment je vais éviter de vous dire des bêtises dessus 😊

Ces langages sont en général plus anciens que le PHP, peut-être un peu moins maniables aussi, en tout cas c'est l'impression que j'en ai. Maintenant que le PHP est là, on n'a pratiquement plus recours à ces "vieux" langages.



Je comprends que vous ayez encore du mal à distinguer ce qu'est un client et ce qu'est un serveur. Si vous décidez par la suite de vous intéresser au PHP, sachez que le premier chapitre explique plus en détail de quoi il s'agit.

Nous allons parler un peu plus en détail maintenant d'un langage client (le Javascript) et d'un langage serveur (le PHP).

## Un langage client : le Javascript

Le langage Javascript a une longue histoire derrière lui. Il s'agit, comme je vous le disais plus tôt, d'un langage qui est exécuté sur la machine du client, c'est-à-dire sur la machine des visiteurs. Il permet d'effectuer des opérations sur le code XHTML de la page web sans avoir à recharger la page.

Contrairement au XHTML, le Javascript est un langage de programmation. Il ressemble un peu au langage C.

Pour vous donner une idée de ce à quoi ça ressemble, voici un bout de code Javascript :

### Code : JavaScript

```
function switch_spoiler(div2)
{
    if (div2.getElementsByTagName('div').length > 0)
        var divs = div2.getElementsByTagName('div');
    else
        var divs = div2.parentNode.nextSibling.getElementsByTagName('div');
    var div3 = divs[0];

    if (div3.style.visibility == 'visible')
```

```
div3.style.visibility = 'hidden';  
else  
div3.style.visibility = 'visible';  
return false;  
}
```

Ce code modifie le style d'une balise lorsqu'on clique dessus : il fait passer son statut "visibility" à "hidden" si la balise était visible, et inversement il la fait passer à "visible" si elle était cachée. Cela permet d'afficher ou de cacher un texte sur la page web lorsque le visiteur clique dessus.

On peut écrire du code Javascript directement dans le fichier .html, ou mieux : dans un fichier .js (tout comme on fait en CSS avec les fichiers .css).

Le Javascript est difficile à résumer en quelques lignes car il permet de réaliser des effets très variés :

- 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 (c'est pas possible à faire avec un :hover car ça concerne 2 balises différentes, c'est bien là une limite du CSS).
- On peut l'utiliser aussi pour modifier la source XHTML sans avoir à recharger 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.

Le Javascript est utilisé dans une technique très en vogue en ce moment dont vous avez peut-être entendu parler : **AJAX** (*Asynchronous JavaScript and XML*). Il s'agit d'une utilisation combinée de Javascript et de XML pour communiquer avec un serveur et recharger dynamiquement certaines parties de la page. Des services comme [Gmail](#) et [Netvibes](#) l'utilisent intensivement, et ils ne sont pas les seuls.

Personnellement, je ne peux pas vous apprendre pour le moment le Javascript car je n'ai pas pris le temps de rédiger un cours là-dessus. En revanche, vous trouverez de nombreux cours de qualité sur le Site du Zéro rédigés par les membres. Je vous invite notamment à consulter :

- [Ce tutoriel sur Javascript](#), un bon point de départ pour commencer.
- [Ce tutoriel sur AJAX](#), à lire si vous avez déjà manipulé un peu de Javascript avant et que vous voulez aller plus loin.
- [Le sommaire de la catégorie Javascript](#), qui vous présente tous les cours Javascript disponibles sur le Site du Zéro.

Bonne lecture ! 😊

## Un langage serveur : le PHP

Les langages serveur et les langages client ne peuvent pas être comparés car ils font des choses différentes. N'essayez donc pas de comparer le Javascript et le PHP, ces langages-là ne sont pas en concurrence.

Je vous ai parlé du Javascript et je vous ai expliqué que l'intérêt de ce langage était minime. Il permet de réaliser certains effets, mais globalement on peut s'en passer.

En revanche, le PHP est pratiquement incontournable. Oui, on peut faire un site web sans PHP (la preuve, vous savez déjà le faire), mais au bout d'un moment vous finirez par avoir besoin du PHP.

Tenez, par exemple c'est le langage PHP qui va vous permettre de "récupérer" les informations entrées par le visiteur dans un formulaire, et qui se chargera de vous transmettre les informations par mail par exemple.

Voici quelques exemples parmi tant d'autres de choses que permet de faire le PHP :

- Livre d'or
- Forums

- Newsletter
- Chat
- Compteur de visiteurs
- Système de news automatisé
- Système de membres
- Jeux PHP sur son site web (jeux de stratégie, élevage d'animaux virtuels...)
- ~~Le café~~ (ah non ça il sait pas faire c'est vrai)
- ... Je continue ou je peux arrêter là ? 😊

Le PHP, c'est aussi et surtout une grande communauté de programmeurs qui peuvent vous aider.

Le PHP enfin, c'est une mascotte : l'éléPHPant :



*L'éléPHPant*

Le PHP est assez différent du XHTML et du CSS car il ressemble déjà plus à de la vraie programmation. Il a d'ailleurs quelques ressemblances avec le langage C++ (à la différence près que le C++ est fait pour créer des programmes, et le PHP pour créer des sites web).

Il y a une seule chose que le PHP requiert : c'est que vous connaissiez déjà le XHTML et le CSS. Ca tombe bien, c'est votre cas.

Le PHP va, pour dire les choses clairement, vous simplifier la vie (après vous l'avoir compliquée un peu, bien entendu 😊).

Il permet de rendre votre site en quelque sorte "automatique". Un site en PHP, c'est un site qui peut se mettre à jour tout seul sans que vous ayez besoin d'être devant votre ordinateur.

Donc, pour rester simple, on peut distinguer 2 sortes de sites :

- **Les sites en XHTML + CSS (vous savez faire maintenant)** : ces sites fonctionnent très bien, mais pour les mettre à jour il vous faut systématiquement modifier le code XHTML & CSS. Ces sites sont en général simples, et le plus souvent il s'agit de petits sites.  
Un site en XHTML & CSS est dit **statique**, car il ne se modifie pas sans votre intervention.
- **Les sites en XHTML + CSS + PHP** : avec l'ajout du PHP, le site peut être en grande partie automatisé. Une fois le code XHTML, CSS et PHP mis en place, vous n'avez (presque) plus besoin de toucher le code source de votre site. Par exemple, pour ajouter une news vous avez juste besoin de compléter un formulaire et de cliquer sur "Valider" !  
Un site en XHTML, CSS et PHP est dit **dynamique** car c'est un site qui n'a pas vraiment besoin de votre intervention pour évoluer. Un site dynamique est beaucoup plus attrayant et facile à mettre à jour, ce qui explique pourquoi la plupart des gros sites utilisent le PHP.

Vous voudriez peut-être un aperçu de code PHP pour voir à quoi ça ressemble ?

#### Code : PHP

```
<?php
// Connexion à la base de données
try
{
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');
}
catch(Exception $e)
{
    die('Erreur : '.$e->getMessage());
}

// On récupère les 5 derniers billets
$req = $bdd->query('SELECT id, titre, contenu,
DATE_FORMAT(date_creation, \'%d/%m/%Y à %Hh%imin%ss\') AS
date_creation_fr FROM billets ORDER BY date_creation DESC LIMIT 0,
5');
```

```

while ($donnees = $req->fetch())
{
    ?>
    <div class="news">
        <h3>
            <?php echo htmlspecialchars($donnees['titre']); ?>
            <em>le <?php echo $donnees['date_creation_fr']; ?></em>
        </h3>

        <p>
            <?php
            // On affiche le contenu du billet
            echo nl2br(htmlspecialchars($donnees['contenu']));
            ?>
            <br />
            <em><a href="commentaires.php?billet=<?php echo $donnees['id'];
            ?>">Commentaires</a></em>
        </p>
    </div>
    <?php
    } // Fin de la boucle des billets
    $req->closeCursor();
    ?>

```

C'est un bout du code PHP qui permet d'afficher les 5 derniers billets d'un blog. 😊

Si j'étais vous, je m'entraînerais encore un peu sur le XHTML / CSS pour être sûr de bien les maîtriser, puis j'irais lire le Chapitre 1 du tutoriel PHP (disponible sur le Site du Zéro, ça tombe bien là aussi :p).

Ca ne vous coûte rien de lire le 1er chapitre, et ça peut vous donner une idée un peu plus précise de ce que c'est. Ca serait quand même rudement dommage que vous passiez à côté d'un langage épatant, génialissime, extraordinaire, sublissime, totalement fantasmagorique (...ça va, j'en fais pas trop là ? 😊)

---

Cette fois les amis, on arrive vraiment à la fin 😊

Que pourrait-on rajouter avant de se quitter ?

Eh bien, déjà je tiens à vous dire que j'ai été très heureux d'avoir pu partager mes modestes connaissances avec vous (et si c'est réciproque, alors tant mieux 😊)

Si vous n'y avez pas encore jeté un oeil, pensez à lire [les annexes](#) de ce cours. Vous y dénicherez très certainement des infos intéressantes supplémentaires, comme "[Comment envoyer son site sur le Web](#)", ou encore "[Comment gagner de l'argent grâce à la publicité](#)" etc etc.

Comme vous le voyez, le petit monde de l'informatique recèle de surprises qui, si on prend la peine de s'y intéresser, finissent vraiment par nous étonner. C'est cette curiosité qu'il vous faut conserver, en informatique mais aussi partout ailleurs, car c'est elle qui vous fait progresser et qui vous empêche de vous contenter de ce que vous savez déjà.

Etes-vous curieux d'en savoir plus sur ce langage "PHP" ?

Si c'est le cas... alors croyez-moi, vous n'avez pas fini d'être étonnés 😊

## Partie 3 : Annexes

Les annexes contiennent d'autres informations qui vous seront utiles lors de la création de votre site web, comme des Mémos (résumé), ou encore des explications sur la façon dont on envoie un site sur le web.

Vous n'êtes pas obligés de lire ces informations à la fin, vous pouvez vous en servir n'importe quand lors de votre lecture du cours.

---

### Envoyez votre site sur le web

Votre site est tout beau, tout propre, tout prêt... Mais comme il est sur votre disque dur, personne d'autre ne va pouvoir en profiter !

Vous aimeriez donc l'envoyer sur le web, mais... bien sûr vous ne savez pas comment faire 🤔

Nous allons découvrir dans cette annexe tout ce qu'il faut savoir pour envoyer son site sur le web. Dans l'ordre :

1. Nous découvrirons comment réserver **un nom de domaine**
  2. Puis nous verrons ce qu'est **un hébergeur** et comment cela fonctionne
  3. Enfin, une fois notre hébergeur choisi, nous verrons comment utiliser **un client FTP** pour enfin pouvoir transférer les fichiers sur le net 😊
-

## Le nom de domaine

Savez-vous ce qu'est un **nom de domaine** ?

Il s'agit en fait d'une adresse sur le Web : siteduzero.com est par exemple un nom de domaine.

Un nom de domaine est constitué de 2 parties :

**siteduzero.com**

- **En rouge, le nom de domaine** proprement dit. Il s'agit d'un nom que l'on peut généralement choisir librement, du tant que personne ne l'a réservé avant nous. Il peut contenir des lettres et des chiffres, mais pas de symboles particuliers (comme le ç français, le é, le è, les espaces, etc).
- **En bleu, l'extension (aussi appelée tld)**. Il existe grosso modo une extension par pays (.fr pour la France, .be pour la Belgique, .ca pour le Canada). Toutefois, il y a aussi des extensions utilisées au niveau international comme .com, .net, .org. Elles étaient au départ réservées aux sites commerciaux, aux organisations, etc... mais cela fait longtemps que tout le monde peut les réserver. D'ailleurs, .com est très probablement l'extension la plus utilisée sur le Web.



En général, un site web voit son adresse précédée par "www", comme par exemple "www.siteduzero.com". Cela ne fait pas partie du nom de domaine : en fait, "www" est ce qu'on appelle un sous-domaine, et on peut en théorie en créer autant qu'on veut une fois qu'on est propriétaire du nom de domaine 😊

Le "www" a été adopté par tous les webmasters, c'est une sorte de convention, mais elle n'est absolument pas obligatoire.

## Réserver un nom de domaine



Moi aussi je veux un nom de domaine pour mon site ! Comment dois-je faire ?

Alors j'ai une bonne et une mauvaise nouvelle 😊

Comme d'hab, on va commencer par la mauvaise :

- **La mauvaise** : ce n'est pas gratuit...
- **La bonne** : ... ce n'est vraiment pas cher du tout 😊

En effet, un nom de domaine coûte entre 7 et 12 € pour un an.

Le prix peut varier en fonction de l'extension. Ainsi, l'extension .info est généralement proposée à plus bas prix et peut s'avérer être une alternative intéressante. Mais si vous voulez une adresse plus "courante", il faudra plutôt viser une extension de type ".com", ou encore ".fr".

Pour réserver un nom de domaine, 2 solutions :

- Passer par un **registrar** spécialisé. C'est un organisme qui sert d'intermédiaire entre l'ICANN (l'organisation qui gère les noms de domaine au niveau international, tel les .com) et vous. 1&1, OVH et Gandi sont de célèbres registrars français.
- Encore mieux : vous pouvez commander le nom de domaine en même temps que l'hébergement (c'est ce que je vous conseille). Comme ça vous faites d'une pierre deux coups, vu que vous aurez de toute façon besoin de l'hébergement et du nom de domaine.



Pour plus d'info sur l'ICANN, je vous invite à lire [cette page](#) en français de leur site qui détaille un peu plus leur rôle sur le Web. C'est que c'est un sacré boulot de gérer la plupart des noms de domaine du Web !

Dans ce chapitre, nous allons voir comment commander un nom de domaine en même temps que l'hébergement, c'est de loin la solution la plus simple et la moins coûteuse pour vous.

---

---

## L'hébergeur

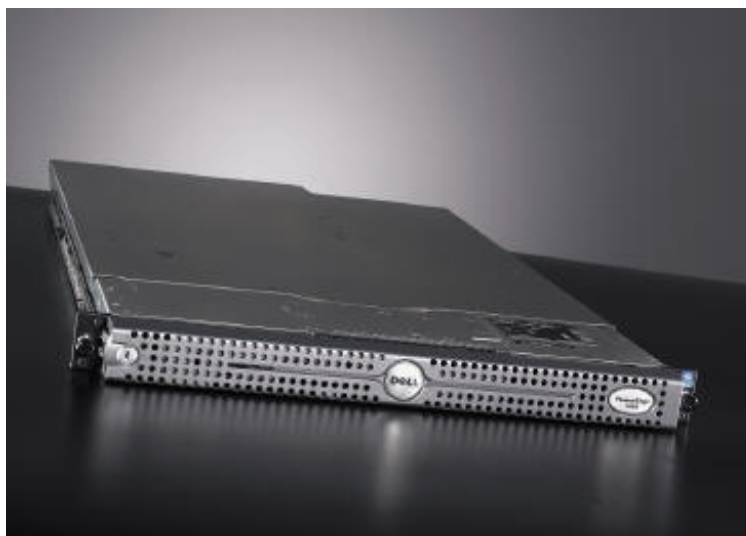
Intéressons-nous maintenant à l'hébergeur.



Qu'est-ce qu'un hébergeur et pourquoi aurais-je besoin de lui ?

Sur Internet, tous les sites web sont stockés sur des ordinateurs particuliers appelés "**Serveurs**". Ce sont des ordinateurs généralement très puissants qui restent tout le temps allumés. Ils contiennent les pages des sites web et les délivrent aux internautes qui les demandent, à toute heure du jour et de la nuit.

Voici à quoi ressemble par exemple un serveur :



*Un serveur*

Un serveur ne possède pas d'écran car, la plupart du temps, il tourne tout seul sans avoir besoin qu'on fasse quoi que ce soit dessus. Comme vous le voyez, les serveurs sont très plats : c'est un format spécial de serveur (appelé "1U"). Cela permet de les empiler dans des **baies** (une sorte d'armoire climatisée pour serveurs 🤖).

Voici à quoi ressemble une baie :



*Une baie de serveurs*



Comme vous le voyez, il y a un écran pour toute la baie. C'est suffisant car on ne branche l'écran sur un serveur que si celui-ci rencontre un problème. La plupart du temps, heureusement, le serveur travaille sans broncher 🤖.

## Le rôle de l'hébergeur

L'hébergeur est une entreprise qui se charge de gérer des baies de serveurs. Elle s'assure du bon fonctionnement des serveurs 24h/24 7j/7. En effet, si l'un d'eux tombe en panne, tous les sites présents sur la machine deviennent inaccessibles (et ça fait des clients mécontents 😞).

Ces baies se situent dans des lieux particuliers appelés **datacenters**. Les datacenters sont donc des "entrepôts à serveurs" en quelque sorte, et leur accès est très protégé.



*Un datacenter. On voit ici plusieurs baies de serveurs.*



Il est aussi possible en théorie d'héberger un site sur son propre ordinateur. Toutefois, c'est complexe, il vaut mieux avoir des connaissances en Linux, l'ordinateur doit être assez puissant, tourner jour et nuit et... surtout... la connexion doit être à très très haut débit (surtout en *upload*, la vitesse d'envoi des fichiers compte énormément). Les particuliers n'ont en règle générale pas une connexion suffisamment puissante pour héberger des sites, tandis que les datacenters oui : ils sont câblés en fibre optique (ça peut aller à une vitesse de plusieurs Gbps ! 🤖)

Bref, gérer un serveur soi-même est complexe, et la plupart du temps les particuliers et les entreprises font appel à un hébergeur dont c'est le métier.

## Trouver un hébergeur

Les hébergeurs, contrairement aux registrars, sont très très nombreux. Il y en a de tous types, à tous les prix. Il y a un vocabulaire à connaître pour vous repérer dans leurs offres :

- **Hébergement mutualisé** : si vous optez pour une offre d'hébergement mutualisée, votre site sera placé sur un serveur gérant plusieurs sites à la fois (peut-être une centaine, peut-être plus). C'est l'offre la moins chère et c'est celle que je vous recommande de viser si vous démarrez votre site web.
- **Hébergement dédié virtuel** : cette fois, le serveur ne gère que très peu de sites (généralement moins d'une dizaine). Cette offre est généralement adaptée aux sites qui ne peuvent plus tenir sur un hébergement mutualisé car ils ont trop de trafic (trop de visiteurs), mais qui ne peuvent pas se payer un hébergement dédié (voir ci-dessous).
- **Hébergement dédié** (on parle aussi de "serveur dédié") : c'est le nec plus ultra. Le serveur gère uniquement votre site et aucun autre. Attention, cela coûte assez cher et il vaut mieux avoir des connaissances en Linux pour administrer le serveur à distance.  
Par exemple, le Site du Zéro est lui-même sur un hébergement dédié, car son trafic est très important.



Mais où puis-je trouver un hébergeur ?

Oh ça c'est très simple 😊

Une recherche dans Google de "hébergeur web" vous donnera plusieurs millions de résultats. Vous n'aurez que l'embarras du choix.



Si je puis me permettre un conseil, je vous recommande de jeter un œil à l'hébergeur **1&1** qui propose des services d'hébergement de qualité.

Si les offres de 1&1 ne vous conviennent pas, vous pouvez regarder chez [l'hébergeur PlanetHoster](#) (un concurrent 😊) ou encore [MavenHosting](#), qui proposent d'autres offres pour les particuliers et entreprises.



PlanetHoster fait des réductions spéciales pour les visiteurs du Site du Zéro sur tous ses hébergements mutualisés via un code promotionnel :



- 5% de remise pour le plan essentiel grâce au code "**SiteDuZero-Perso**"
- 15% de remise pour les plans performance et illimité grâce au code "**SiteDuZero**"

Ces remises sont valables pour une commande annuelle de l'une de ces 3 offres.

La suite de ce chapitre vous détaille la procédure pour héberger votre site chez 1&1, mais sachez que cela fonctionne quasiment de la même manière avec PlanetHoster et MavenHosting.

Revenons à **1&1**. L'hébergeur propose plusieurs offres d'hébergement mutualisé :

	1&1 DUAL ESSENTIEL	1&1 DUAL CLASSIQUE	1&1 DUAL AVANCÉ	1&1 DUAL ILLIMITÉ
	<b>6 mois à 0€*</b> après 1,99 € HT/mois (2,38 € TTC/mois) <b>Continuer</b>	<b>3 mois à 0€*</b> après 4,99 € HT/mois (5,97 € TTC/mois) <b>Continuer</b>	<b>3 mois à 0€*</b> après 9,99 € HT/mois (11,95 € TTC/mois) <b>Continuer</b>	<b>3 mois à 0€*</b> après 19,99 € HT/mois (23,91 € TTC/mois) <b>Continuer</b>
Espace disque	2,5 Go	100 Go	250 Go	illimité
Trafic/mois	illimité	illimité	illimité	illimité
Noms de domaine inclus	1	2	3	4
Extensions proposées	.fr, .com, .info, .net, .org	.fr, .com, .info, .net, .org	.fr, .com, .info, .net, .org	.fr, .com, .info, .net, .org
Comptes email (2 Go par compte)	10	100	500	illimité
1&1 TopSite Express	1 site de 10 pages	3 sites de 15 pages chacun	5 sites de 25 pages chacun	10 sites de 30 pages chacun
Programmation	PHP, PHPDev, Zend Framework	PHP, PHPDev, Perl, Python, Ruby, SSI, Zend Framework	PHP, PHPDev, Perl, Python, Ruby, SSI, Zend Framework	PHP, PHPDev, Perl, Python, Ruby, SSI, Zend Framework
Bases de données MySQL 5 (1 Go) <b>Nouveau !</b>	1	10	100	illimité
1&1 Applications Click & Build <b>Nouveau !</b> 65 applications gratuites à installer en quelques clics	2	illimité	illimité	illimité
Redondance géographique <b>Nouveau !</b>	Disponibilité maximale de votre site : vos données hébergées simultanément dans 2 centres de données ultrasécurisés.	Disponibilité maximale de votre site : vos données hébergées simultanément dans 2 centres de données ultrasécurisés.	Disponibilité maximale de votre site : vos données hébergées simultanément dans 2 centres de données ultrasécurisés.	Disponibilité maximale de votre site : vos données hébergées simultanément dans 2 centres de données ultrasécurisés.

- **1&1 Dual Essentiel** : 2,5 Go d'espace disque, trafic illimité, 1 nom de domaine et 10 comptes email.
- **1&1 Dual Classique** : 100 Go d'espace disque, trafic illimité, 2 noms de domaine et 100 comptes email.
- **1&1 Dual Avancé** : 250 Go d'espace disque, trafic illimité, 3 noms de domaine et 500 comptes email.
- **1&1 Dual Illimité** : espace disque, trafic et comptes email illimités, 4 noms de domaine.



Tous ces packs donnent la possibilité d'avoir des noms de domaine en .fr !

Ces offres ont chacune leurs caractéristiques. Plus l'offre que vous choisissez est élevée, plus vous aurez d'espace disque, de noms de domaine, de comptes email, de bases de données etc. Pour visualiser les différentes caractéristiques, cliquez sur le bouton "Afficher toutes les fonctions", qui se trouve juste sous les offres. N'hésitez pas à regarder les détails : le 1&1 Dual Essentiel ne propose pas les langages Perl, Python ou Ruby par exemple, alors que les autres offres les proposent.



Mais qu'est-ce qu'ils appellent le "trafic" ? 🤔

Le trafic, c'est la quantité de données envoyées par mois aux visiteurs de votre site. Par exemple, si vous avez une image de 1 Mo sur votre site et qu'elle est chargée 500 fois dans le mois par vos visiteurs, alors vous créez un trafic de 500 Mo. En pratique, il faut savoir que les navigateurs des visiteurs mettent en cache les images, ce qui leur évite d'avoir à recharger plusieurs fois une même image. Cela diminue d'autant plus le trafic nécessaire.

Ici le trafic étant illimité sur toutes les offres, cela ne pose pas de problème.



Et comment je sais quelle offre choisir ?



C'est en fonction de vos besoins. Si vous ne voulez héberger que quelques pages HTML par exemple, 2,5 Go de disque seront largement suffisants, vous pouvez alors vous contenter de l'offre 1&1 Dual Essentiel. Si au contraire vous voulez héberger des photos, des vidéos ou des musiques, il vous faudra plus d'espace disque, et donc prendre une offre supérieure. De la même manière, si vous voulez héberger uniquement un site en PHP (un blog par exemple), la première offre suffit ; en revanche, si vous voulez héberger des applications en Ruby, alors il vous faudra prendre une offre supérieure. Au final, c'est à vous de voir ce dont vous aurez besoin.

Peut-être vous êtes vous demandé ce que signifiait ce "dual", dans le nom des offres. Hé bien sachez que 1&1 héberge en réalité vos sites sur deux serveurs en même temps, à deux endroits différents. Ainsi si l'un des serveurs tombe en panne pour une raison quelconque, vos visiteurs pourront toujours accéder à votre site, via le deuxième serveur. D'où l'appellation de "dual".

C'est une sécurité non négligeable, à prendre en compte lorsque vous hébergez un site.



1&1 met à votre disposition des applications web à installer, via "1&1 Applications Click & Build". Ce sont des sites déjà codés, comme Wordpress, phpBB, Joomla! ou encore Magento. Au total, ce ne sont pas moins de 65 applications prêtes à l'emploi, à installer facilement directement via 1&1. De quoi se faciliter grandement la vie ! **En effet, vous n'avez même pas besoin de savoir coder pour avoir votre blog, forum ou site en ligne.**

Vous trouverez la liste complète de ces applications [sur la page dédiée](#).

### Applications Click & Build



- ✓ 65 applications GRATUITES à installer en 1 clic
- ✓ Disponibles dans tous les packs Linux
- ✓ Des applications pour tous vos besoins : e-commerce, blog, forums, etc.
- ✓ Mises à jour automatiques pour que vous disposiez toujours des dernières versions

## Commander un hébergement pour votre site web

Après avoir cliqué sur n'importe quel bouton "Continuer", nous arrivons sur la page suivante :

## RÉSERVEZ VOTRE NOM DE DOMAINE DÈS MAINTENANT !

Votre Pack : \*

**noms de domaine inclus dans votre pack !**

Vérifiez la disponibilité de votre nom de domaine :

www.

☒ .fr
 ☐ .com
 ☐ .biz
 ☐ .eu
 ☐ .info

☐ .be
 ☐ .mobi
 ☐ .net
 ☐ .org
 ☐ .name

☐ .tv
 ☐ .cc
 ☐ .ws
 ☐ .co

[Informations et prix](#)

**JE RÉSERVE MON NOM DE DOMAINE PLUS TARD**

Si vous ne souhaitez pas réserver votre nom de domaine maintenant, vous pouvez le faire plus tard dans votre Espace Client.

> Rechercher

> Continuer sans domaine

Ce cadre vous invite à vérifier la disponibilité de votre nom de domaine. 1&1 va alors se charger de vérifier instantanément si le domaine est disponible. Si c'est bon, vous pouvez passer à la suite. 😊

Sinon, il faudra choisir un autre nom de domaine car quand le domaine est déjà pris vous ne pouvez pas faire grand chose. 😞  
Néanmoins, 1&1 vous propose des noms alternatifs disponibles. Il est également possible de choisir votre nom de domaine

après avoir validé votre commande.

**NOMS DE DOMAINE DISPONIBLES :**

	Statut	Frais annuels	Votre sélection
www.adressedemonsite.fr	disponible	inclus*	<input checked="" type="checkbox"/>

> Continuer



Obtenir un "vrai" nom de domaine (.fr, .com, .net, .org...) est habituellement payant chez les hébergeurs. Néanmoins, si vous achetez un hébergement d'un an chez 1&1, le nom de domaine est offert (il est compris dans l'hébergement).

1&1 vous propose ensuite différentes options, libre à vous de les accepter ou de les refuser.

Il ne vous reste plus qu'à renseigner vos coordonnées et finaliser l'achat par carte bancaire ou par chèque.

Une fois les formalités et le paiement effectués, vous êtes redirigé vers 1&1, qui vous confirme la bonne prise en compte de votre commande.

Vous devriez recevoir un peu plus tard un e-mail vous indiquant toutes les informations nécessaires pour mettre en place votre site. Conservez-les précieusement, vous en aurez besoin.

Lorsque vous aurez reçu par email vos identifiants pour vous connecter au serveur de votre hébergeur, vous pouvez passer à l'étape suivante : **envoyer votre site web sur le serveur de votre hébergeur !**

## Utiliser un client FTP

### Installer un client FTP

FTP signifie *File Transfer Protocol* et, pour faire court et simple, c'est le moyen que l'on utilise pour envoyer nos fichiers. Il existe des logiciels permettant d'utiliser le FTP pour transférer vos fichiers sur Internet.

Bien entendu, des logiciels FTP il en existe des centaines, gratuits, payants, français, anglais etc... Pour que nous soyons sur la même longueur d'ondes, je vais vous proposer celui que j'utilise qui est gratuit et en français : **FileZilla**.



Ce logiciel n'a rien à avoir avec Mozilla, si ce n'est qu'il se termine lui aussi par "zilla". N'allez donc pas croire que je vous force à utiliser des logiciels d'un même éditeur, c'est tout à fait faux. D'ailleurs, vous pouvez utiliser n'importe quel autre logiciel FTP si ça vous chante, ça ne me dérange absolument pas.

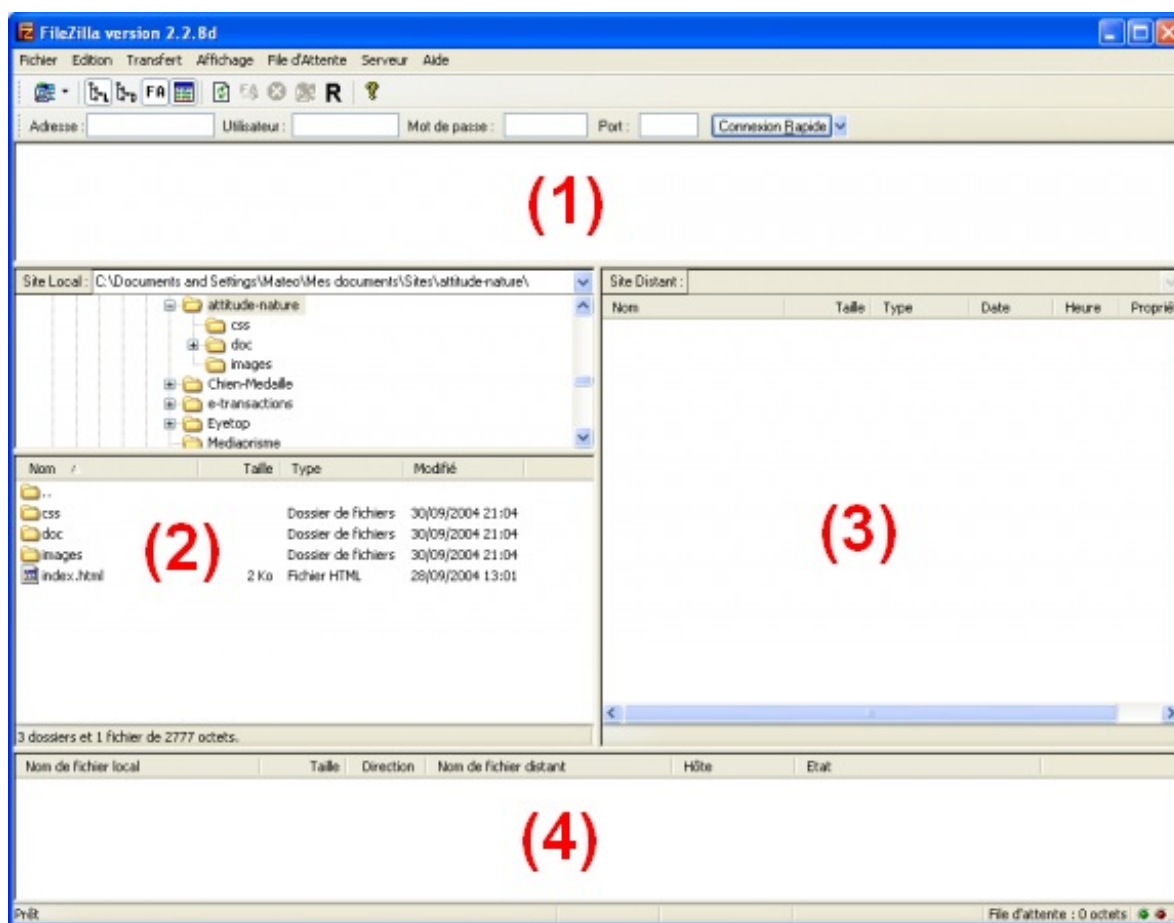
Quoiqu'il en soit, je vais vous montrer quelle est la marche à suivre avec FileZilla.

Première étape : ... le télécharger bien entendu 😊

### Télécharger FileZilla (3,9 Mo)

Je vous fais confiance pour l'installation, elle est toute simple et vous ne devriez pas avoir de problème 😊

Lancez le logiciel, et vous devriez voir ceci :



A première vue, ça a l'air un peu compliqué (à première vue seulement). En fait, le principe est très simple. Il y a 4 grandes zones dans la fenêtre à connaître :

1. En haut, vous verrez apparaître les messages qu'envoie et reçoit le logiciel. Si vous avez un peu de chance, vous verrez même la machine vous dire Bonjour ! (si si je vous jure ;) ) En général, cette zone ne nous intéresse pas vraiment, sauf s'il y a des messages d'erreur... et comme ils sont écrits en rouge, vous devriez pas les louper 😊

2. A gauche, c'est votre disque dur. Dans la partie du haut vous avez les dossiers, et dans la partie du bas la liste des fichiers du dossier actuel.
3. A droite, c'est la liste des fichiers envoyés sur Internet. Pour le moment il n'y a rien, car on ne s'est pas "connecté", mais ça va venir ne vous en faites pas.
4. Enfin, en bas, vous verrez apparaître les fichiers en cours d'envoi (et le pourcentage d'envoi).

La première étape va être de se "connecter" au serveur de votre hébergeur.

## Configurer le client FTP


Quel que soit l'hébergeur que vous avez choisi, cela fonctionne toujours de la même manière. On va vous fournir **3 informations** qui sont indispensables pour que FileZilla puisse se connecter au serveur :

- **L'IP** : c'est "l'adresse" du serveur. Le plus souvent, on vous donnera quelque chose du genre *ftp.mon-site.com*, mais il peut aussi s'agir d'une suite de nombres comme *122.65.203.27*
- **Le login** : c'est votre identifiant, on vous l'a probablement demandé. Vous avez peut-être mis votre pseudo, ou le nom de votre site. Mon login pourrait par exemple être *mateo21*.
- **Le mot de passe** : soit on vous a demandé un mot de passe, soit (c'est plus probable) on vous en a attribué un d'office (un truc imprononçable du genre *crf45u7h*)

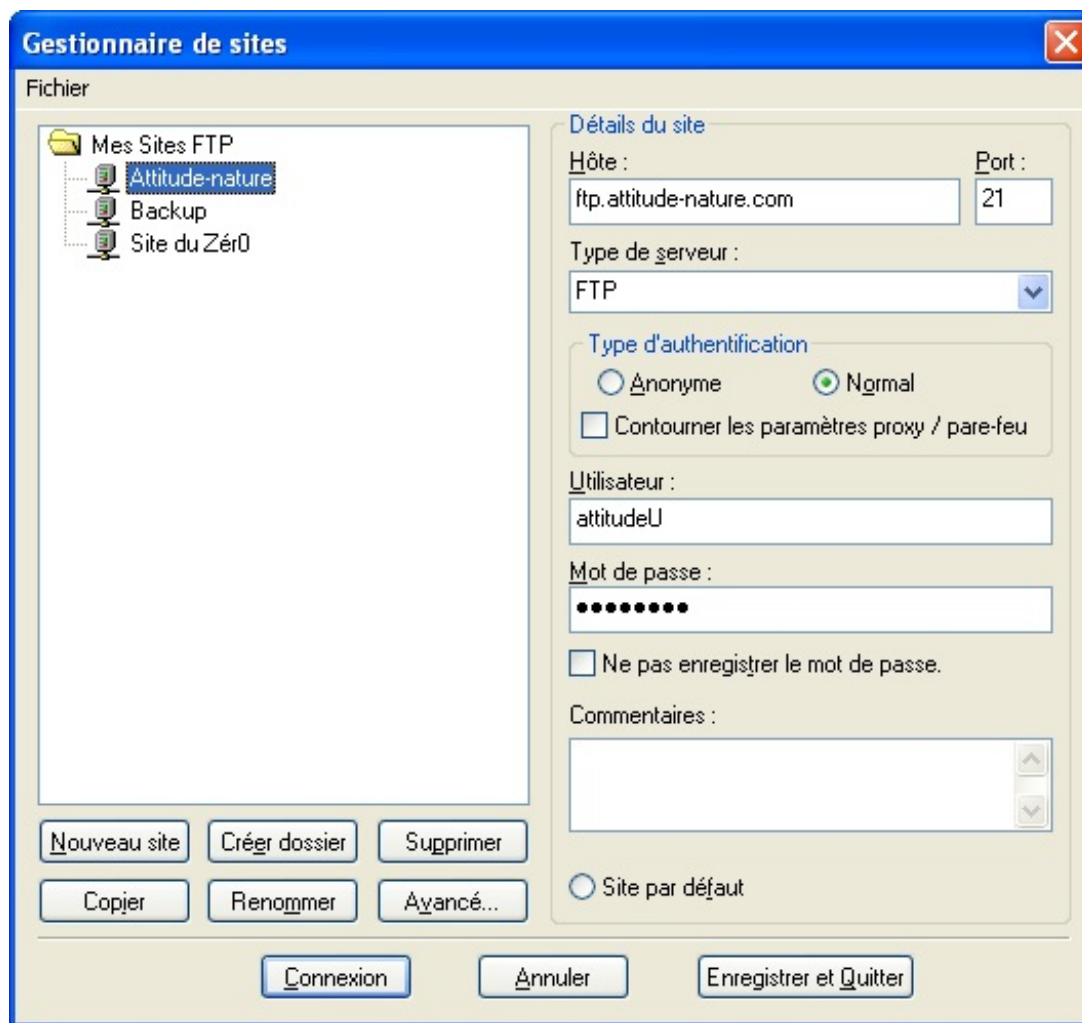
Si vous avez ces 3 informations, vous êtes le roi du monde 😁 (du moins, vous allez pouvoir continuer ce tutoriel 😊)

Si vous ne les avez pas, il faut que vous les cherchiez c'est indispensable. On vous les a probablement envoyées par mail. Sinon, n'hésitez pas à les demander à votre hébergeur (IP / Login / Mot de passe).

Maintenant que nous sommes en possession de ces informations, nous allons les donner à FileZilla qui en a besoin pour se connecter au serveur.

Cliquez sur la petite icône en haut à gauche (pas sur la petite flèche à droite, mais sur l'image) : 

Une fenêtre s'ouvre. Cliquez sur "Nouveau site" et donnez-lui le nom que vous voulez (par exemple *Site du Zéro*). A droite, vous allez devoir indiquer les 3 informations dont je viens de vous parler, comme ceci :



Vous pouvez distinguer en haut l'hôte (c'est là qu'il faut indiquer l'adresse ip, du genre *ftp.attitude-nature.com*).

Pour pouvoir entrer le login / mot de passe, cochez **Type d'authentification : Normal**

Ici le login est *attitudeU* et le mot de passe est... caché (vous croyiez quand même pas que j'allais vous le donner ?! :p)

Cliquez sur "Connexion" et le tour est (presque) joué.

## Transférer les fichiers

A ce stade, 2 possibilités :

- Soit la connexion a réussi, et vous voyez apparaître en haut des messages en vert comme "Connecté". Dans ce cas, la zone de droite de la fenêtre de FileZilla devrait s'activer et vous verrez les fichiers qui se trouvent déjà sur le serveur (il se peut qu'il y en ait quelques-uns déjà présents).
- Soit ça a planté, vous avez plein de messages écrits en rouge et là bah... Il n'y a pas 36 solutions : vous vous êtes plantés en tapant l'IP ou le login ou le mot de passe. Un de ces éléments est incorrect, veuillez à les redemander à votre hébergeur car s'ils sont bons ça doit marcher.

Si la connexion a réussi, alors ce que vous avez à faire est très simple : dans la partie de gauche, cherchez où se trouvent sur votre disque dur vos fichiers .php, .html et .css (mais aussi vos images .jpg, .png, .gif etc...).

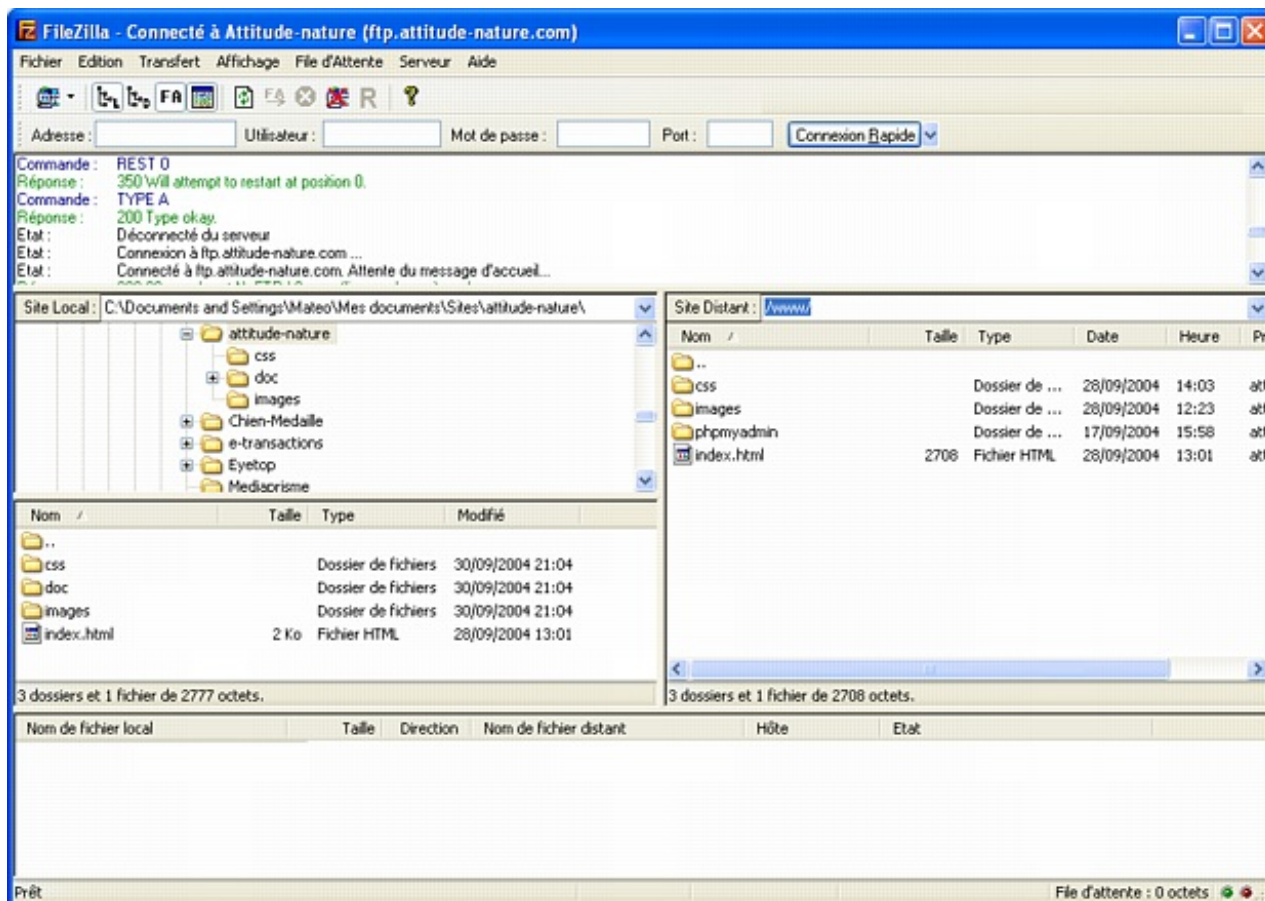
Double-cliquez sur le fichier que vous voulez transférer à gauche. Au bout de quelques secondes, il apparaîtra à droite, ce qui voudra dire qu'il a été correctement envoyé sur le serveur, donc qu'il est accessible sur Internet !!! 😊



Vous pouvez envoyer n'importe quel type de fichier. Bien entendu, généralement on envoie des fichiers .php, .html, .css et des images, mais vous pouvez aussi très bien envoyer des .pdf, des programmes, des .zip, etc.



Voici par exemple ce que ça donne quand on a transféré un fichier "index.html" :



Il apparaît à droite, ce qui veut dire qu'il est maintenant disponible sur le serveur 😊



Veuillez noter qu'il faut que votre page d'accueil s'appelle "index.html". C'est la page qui sera chargée lorsqu'un nouveau visiteur arrivera sur votre site.

Vous pouvez aussi transférer des dossiers entiers d'un seul coup : il suffit de faire glisser-déplacer le dossier depuis la partie de gauche jusqu'à la partie de droite de la fenêtre.

Allez, avouez qu'une fois configuré, c'est pas bien compliqué 😊

C'est à peu près tout ce que vous avez besoin de savoir.

Bien entendu, vous ne pouviez pas deviner tout seul tout ceci. C'est d'ailleurs pour cela que j'ai rédigé cette annexe, car bon nombre de débutants sont perdus et ne comprennent pas l'intérêt des registrars, hébergeurs, serveurs et compagnie 😊

Allez, au boulot, vous avez des fichiers à transférer je crois 😊

## Gagner de l'argent grâce à la publicité

Vaste sujet que celui de la publicité sur Internet...

Pour beaucoup, il s'agit d'une vraie plaie. En effet, qui n'a jamais eu envie de commettre un meurtre à cause des popups ?

Pourtant, pour les webmasters la publicité est le seul moyen *simple* de gagner de l'argent.

Sans la publicité, le Site du Zéro n'existerait d'ailleurs tout simplement pas aujourd'hui. En effet, lorsque votre site commencera à grandir, vous aurez besoin de prendre un hébergement payant. Il vous faudra donc de l'argent.

D'autre part, peut-être que certains d'entre vous ont simplement envie de gagner un peu d'argent grâce au site web qu'ils ont mis tant de mal à faire. Quel mal y a-t-il à cela ? 😊

Dans cette annexe, nous allons voir comment mettre en place une publicité sur son site et quelles sont les règles à suivre.

Je tenterai aussi de répondre aux questions les plus courantes que vous vous posez 😊



## Qu'est-ce qu'une bonne publicité ?

Nous allons voir dans un premier temps ce que j'entends par **une bonne publicité**.

En effet, on diabolise aujourd'hui souvent la publicité sur le net à cause de certains mauvais souvenirs : popups, popunders, animations multicolores etc...

Pourtant, je veux vous montrer qu'une publicité peut être intelligente et s'adapter au visiteur.

De mon point de vue, la publicité doit :

- **Eviter à tout prix de gêner la navigation du visiteur** : interdisez-vous notamment d'utiliser des popups, ce moyen est dépassé et la plupart des navigateurs les bloquent aujourd'hui.
- **Etre ciblée**, c'est-à-dire être dans les centres d'intérêt du visiteur. Une publicité pour les casinos n'a rien à faire sur un site de passionnés de dauphins par exemple.

Si votre publicité ne gêne pas la navigation, vos visiteurs ne vous en voudront pas.

Si votre publicité intéresse vos visiteurs, alors en plus vous aurez des chances de les voir cliquer sur la pub, donc vous pourrez gagner de l'argent !

Je vais maintenant détailler un peu plus en profondeur ces 2 principes qui me semblent fondamentaux.

### Principe n°1 : ne pas gêner la navigation

Il n'y a rien de pire qu'une publicité qui perturbe la navigation de vos visiteurs.

Voici quelques exemples de choses qu'il ne faut pas faire :

- Pas d'ouverture automatique de fenêtres (**popups, popunders**). Cette méthode a de toute façon de moins en moins d'avenir car les navigateurs bloquent ces fenêtres intempestives aujourd'hui. A une époque, les popups pour des sites de Casino s'ouvraient toutes les 2 minutes, un vrai cauchemar. Aujourd'hui toutefois, on assiste à l'apparition d'un genre nouveau de popups lancées par des animations Flash ou simulées par du Javascript. La lutte n'est donc pas encore terminée...
- **Evitez les énormes publicités colorées qui clignotent**. Quand on essaie de se concentrer sur le texte de la page, on n'apprécie pas que son oeil soit dérangé par un autre élément. Par exemple, la publicité ci-dessous est trop grosse, trop colorée et détourne trop l'attention du visiteur :



- Si votre site est dans un ton bleu, évitez de mettre une publicité rouge. En clair, **évitez les forts contrastes** 😬 Il est préférable que votre publicité se "fonde" dans le décor de votre page web autant que possible. Attention toutefois à ne

pas tomber dans l'excès inverse : une publicité qu'on ne voit plus est une publicité qui ne rapporte rien 🤔

- **Ne pas placer la publicité au centre de la page.** Ok, vous voulez que votre pub soit vue, mais vous risquez de "couper" la lecture du visiteur en mettant la pub au beau milieu du texte. Mettez-la de préférence dans un menu ou tout en haut de la page.

Si on prend en compte tous ces éléments, une bonne publicité doit donc :

- Etre compatible dans le design de votre site
- Etre placée dans un menu ou en haut de la page
- Elle doit être une image ou du texte, mais pas une fenêtre popup ou popunder.

## Principe n°2 : une publicité ciblée

Un des éléments les plus importants à prendre en compte est *le ciblage de votre publicité*.

Faites une enquête sur votre site, essayez de déterminer le profil de vos visiteurs. Pour cela, vous allez devoir récupérer ces informations à l'aide de sondages par exemple :

- L'âge moyen de vos visiteurs
- Le sexe de vos visiteurs
- Leur(s) passion(s) principale(s)
- L'heure à laquelle ils se connectent sur votre site en moyenne

Grâce à ces informations, vous saurez à qui vous avez affaire. Vous serez donc plus facilement capables de trouver une publicité adaptée, et c'est dans votre intérêt ! En effet, si la publicité est très bien ciblée vous augmentez vos chances de les voir visiter les annonces publicitaires. C'est mathématique 🤔

---

## Installation d'une bannière publicitaire

Vous voulez vous lancer ? Parfait !

Mais... où aller maintenant ? 🤔

Il existe de nombreuses **régies publicitaires** : ce sont des entreprises qui font le lien entre les gens qui veulent diffuser leur publicité (*les annonceurs*) et ceux qui affichent ces publicités (*les webmasters*, c'est-à-dire vous 😊)

En ce qui me concerne, j'ai déjà pas mal baroudé. J'ai eu l'occasion de passer par de nombreuses régies publicitaires pendant l'évolution du Site du Zéro. J'ai vraiment vu de tout, entre les publicités qui rapportaient 0,30 euros au bout d'un mois (*waouh !*), celles qui ouvraient des popups alors que j'avais pourtant demandé à ne pas en avoir, ou encore les régies qui mettent 6 mois à vous payer (et encore c'est parce que je les avais appelé 3 fois dans la même semaine pour les relancer !).

Bref, il y a du bon et du moins bon. Pas mal de moins bon pour être franc 🙄

Parmi les régies montantes du moment, je vous invite à regarder du côté d'**AdFever**. Il s'agit d'une régie française.



Il faut aussi connaître **Google AdSense**, qui est un éditeur de publicités célèbre. Néanmoins, ses rémunérations ont fortement baissé ces derniers temps et il n'est plus aussi intéressant qu'il pouvait l'être autrefois.

Bref, vous pouvez regarder du côté d'AdSense aussi si vous voulez, mais je vais ici plutôt vous expliquer comment fonctionne **AdFever**. Pour information, le Site du Zéro utilise AdFever.

## Les avantages d'AdFever

**AdFever** possède les particularités suivantes :

- La publicité est **entièrement paramétrable**, que ce soit au niveau de la taille ou des couleurs. Vous pouvez donc l'adapter au design de votre site. C'est une des rares régies à proposer cela !
- Les publicités sont le plus souvent **sous forme de texte** : ce ne sont donc pas de grosses images clignotantes qui ont de quoi rendre épileptique les plus solides d'entre vous.
- **Les publicités sont ciblées**, et ça c'est à mon avis le gros point fort du système. AdFever analyse le contenu de la page actuellement visitée. En fonction des mots-clés qui s'y trouvent, des publicités adaptées sont diffusées. Si votre visiteur passe sur une page sur le chocolat, alors il aura des publicités en rapport avec le chocolat ! 😊
- **Les publicités sont innovantes** : on peut y trouver de nouveaux types de publicités originales, comme les Tag Clouds ou les HotSpots.

## S'inscrire à AdFever

Pour vous inscrire à AdFever, rendez-vous sur la page suivante :

### Inscription à AdFever

Dans la page qui s'ouvre à vous, vous cliquerez sur le bouton "S'inscrire".  
Vous aurez un court formulaire à remplir pour vous inscrire.

Une fois que cela sera fait, l'équipe d'AdFever passera analyser votre site et vérifier s'il convient. Notez que ce n'est pas un robot mais un humain qui ira visiter votre site.



Mais... Comment je sais si mon site convient à AdFever ?  
Il y a un minimum de visiteurs par mois à avoir ?

Non, AdFever n'impose pas de minimum de visiteurs (contrairement à d'autres régies). Néanmoins, ils ne semblent accepter à l'heure actuelle que des sites ayant leur propre nom de domaine (monsite.fr, monsite.com...).

Je vous conseille donc de tenter le coup, au pire des cas si votre site est refusé vous pourrez toujours essayer AdSense.

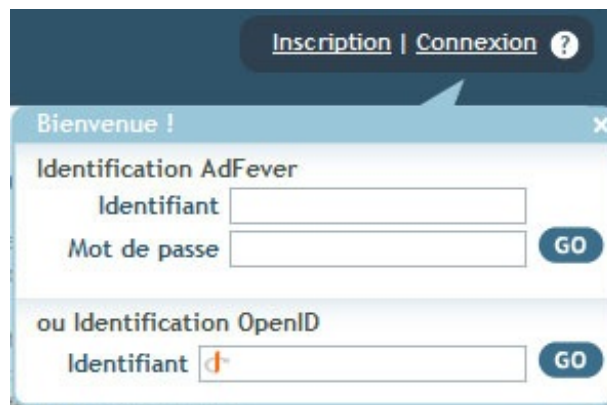
Notez dans tous les cas qu'aucune régie publicitaire n'accepte les sites en construction. Il faut que votre site soit terminé pour qu'il passe la validation.

Normalement vous recevrez au bout de quelques jours un e-mail qui vous informera si votre site est validé (ou non 🤪).

## L'interface d'AdFever

L'interface d'administration d'AdFever qui vous est offerte est un de ses plus gros points forts. De là, vous pouvez gérer vos publicités, en créer de nouvelles, et visualiser vos gains.

Il faudra tout d'abord vous logger (login / mot de passe) pour pouvoir y accéder. Utilisez ce petit cadre en haut à droite de la [page d'accueil d'AdFever](#) pour rentrer vos identifiants :



Votre page d'accueil de l'administration ressemble à ceci :



L'interface est assez claire et vous ne devriez pas avoir de mal à naviguer dedans 😊

Dans un premier temps, vous voudrez certainement créer une bannière publicitaire pour l'insérer sur votre site. Rendez-vous sur l'onglet "Objets publicitaires" puis cliquez sur "Créer un objet".

Là, un assistant vous aidera à paramétrer votre bannière. C'est justement un des points forts d'AdFever : les bannières sont très paramétrables.





Voici ce que signifient les différentes étapes dans l'ordre :

1. **Site** : indiquez pour quel site vous voulez créer votre bannière. Il est très probable que vous n'ayez qu'un seul site, mais si vous en avez plusieurs, vous avez la possibilité de distinguer les bannières en fonction de vos différents sites.
2. **Solution de monétisation** : choisissez "Liens sponsorisés". L'autre choix, "affiliation interne", ne sert que si vous souhaitez faire de l'affiliation, ce qui n'est probablement pas votre objectif dans un premier temps.
3. **Objet publicitaire** : vous avez le choix entre plusieurs types de publicités. De nouveaux types devraient d'ailleurs apparaître prochainement. Pour le moment, on compte :
  - *Content Match* : de la publicité sous forme de lien texte, ciblée en fonction du contenu de votre page. C'est le plus classique et c'est probablement ce que vous choisirez dans la plupart des cas.
  - *HotSpots* : de la publicité qui propose différentes catégories de liens par thèmes. C'est assez original et je vous invite à l'essayer aussi.
  - *Flux XML* : pour une intégration plus avancée à votre site, les flux XML sont puissants mais réservés aux programmeurs qui savent déjà se servir de XML.
4. **Format** : indiquez la taille de la publicité.
5. **Habillage** : choisissez le "look" de la publicité.
6. **Couleurs** : personnalisez le look avec une couleur qui correspond bien à votre site.

A la fin, cliquez sur "Terminer". On vous proposera de récupérer le code de l'objet publicitaire.

Copiez-collez ce code où vous le souhaitez sur votre site web (il s'agit de balises javascript de type <script>). Bravo, vous avez réussi, la publicité devrait s'afficher ! 😊



## Le vocabulaire de la pub

La publicité sur Internet, c'est tout un vocabulaire.

Lorsqu'on voit ces mots barbares pour la première fois, on se demande comment on va faire pour comprendre tout ça 😊

Rassurez-vous, j'ai prévu pour vous un petit lexique qui vous aidera à bien démarrer 😊

- **Clics (ou Adclicks)** : nombre de clics effectués sur la publicité.
- **CPC (Coût par Clic)** : somme qui vous est versée pour chaque clic sur la publicité.
- **PAP (Pages avec Publicité)** : nombre de fois que la publicité a été affichée. Cela correspond généralement au nombre de pages qui ont été vues sur votre site.  
On parle aussi d'*impressions de pages*. C'est un synonyme.
- **CPM (Coût pour Mille)** : certains annonceurs, malheureusement très rares, vous rémunèrent au nombre de fois que vous affichez la publicité (et non au nombre de clics sur la publicité). Le CPM est donc la somme que l'on vous verse pour 1000 affichages de la publicité.
- **eCPM (CPM effectif)** : rapport entre l'argent gagné et le nombre de fois que la pub a été affichée (à ne pas confondre avec le CPM). C'est donc le calcul  $Cl\dot{ic}s / PAP$ .  
Un eCPM élevé indique que votre pub est efficace et donc qu'une proportion importante de vos visiteurs visite les annonces publicitaires.
- **CTR** : nombre de clics par rapport au nombre d'affichages de la pub. Si vous avez un clic pour 100 affichages de la pub, alors vous avez un CTR de 1%. En pratique, le CTR tourne entre 0,1% et 0,4%.
- **Inscription** : dans le cas d'un parrainage, il s'agit du nombre de visiteurs qui se sont inscrits à un programme de parrainage.

Avec ces informations vous devriez comprendre un peu mieux le panel d'administration d'AdFever 😊

---

## Vos questions courantes

Vous vous posez généralement de nombreuses questions au sujet d'AdFever.

Dans cette partie, je vais essayer de résumer les plus courantes d'entre elles et d'y répondre 😊



Quand est-ce que je recevrai mon argent ?

A la fin de chaque mois, AdFever vérifie si vous avez gagné au moins 100€.

Si tel est le cas, AdFever vous rémunérera vers la fin du mois suivant.

Si vous avez gagné moins de 100€ pas de panique. La somme sera rajoutée le mois suivant. Si le mois suivant vous cumulez au moins 100€ vous serez rémunéré, sinon il faudra attendre encore le mois d'après 😊



Combien un clic sur la publicité rapporte-t-il ?

C'est totalement impossible à dire. Cela dépend des publicités que vous affichez : certaines rémunèrent plus que d'autres. Par ailleurs, AdFever prend une commission à chaque clic.

En bref, si vous avez du bol et que vous affichez les bonnes pubs, vous gagnerez plus d'argent 😊



Si je clique 10 fois d'affilée sur la pub, ça comptera pour 10 clics ?

Vous rêvez ! 😊

Tout d'abord, il est interdit au webmaster de cliquer sur ses propres pubs.

Ensuite, AdFever analyse qui clique sur les pubs. Ils utilisent des techniques spéciales (qu'ils ne dévoilent pas) pour repérer les tricheurs. Si vous êtes repéré en train de tricher, vous serez banni du système AdFever et vous ne toucherez pas l'argent que vous avez accumulé jusque-là.

Bref, laissez les choses se faire et ne forcez pas le destin ou vous le regretterez amèrement !



Doit-on être majeur pour recevoir l'argent des publicités ?

Oui. Mais ce n'est pas un obstacle pour autant 😊

En effet, vous pouvez passer par vos parents : dans ce cas il faudra vous inscrire sous le nom d'un de vos parents. Prévenez-les avant quand même.



J'ai plusieurs sites. Comment faire si je veux mettre mes pubs sur tous mes sites ?

Il faut demander dans le panel d'administration à rajouter un site à la liste des sites que vous gérez.

Vous pourrez ensuite créer des publicités propres à ce nouveau site.

---

J'espère que cette annexe vous aura aidé à démarrer votre première activité sur Internet 😊

Comme vous le voyez, aujourd'hui ce n'est pas bien difficile et il y a moins de contraintes. Il y a quelques années encore, si vous n'aviez pas un gros site il était très très difficile de trouver une régie publicitaire potable 😊

Bref, vous avez de la chance, profitez-en. Surtout que ça ne vous engage à rien 😊

**Retenez bien** : la publicité sur un site web n'est PAS un problème si vous respectez les quelques règles que je vous ai énoncées au début du chapitre. La publicité ne doit pas être subie, elle doit au contraire aider le visiteur et lui proposer des liens vers des sites qui l'intéressent.

---

## Le W3C et les standards du web

Voilà une annexe au nom bien pompeux, et pourtant terriblement importante.

Je préfère vous prévenir de suite : vous n'allez pas apprendre ici à créer des supers effets dynamiques qui clignotent dans tous les sens, mais vous allez apprendre à connaître un peu de l'histoire du web.

Pourquoi aujourd'hui on fait du XHTML et non plus du HTML ? Pourquoi on a décidé de séparer le contenu (XHTML) de la mise en forme (CSS) ?

Pourquoi on dit que certains sites sont valides alors que d'autres ne le sont pas ?

Et le votre de site... est-il valide ?

---

## L'histoire du web

Nous allons commencer par la base à connaître : l'histoire du web. Tout webmaster se doit de savoir depuis quand le web existe et quelle est sa petite histoire.

Ce n'est pas seulement de la "culture générale", c'est aussi ça qui va vous permettre de comprendre la suite de cette annexe. Soyez attentifs, et vous paraîtrez un peu moins bête quand vous discuterez du web avec vos amis webmasters 🤪

- Tout a commencé au début des années 70, au cours d'une réunion à l'imprimerie du gouvernement canadien. Un certain William Tunnicliffe propose de séparer le contenu de l'information (le texte) de la manière de l'afficher. Déjà à l'époque, alors qu'Internet n'était réservé qu'à quelques très rares privilégiés, on parlait de séparer le contenu (ce qui est aujourd'hui devenu le langage XHTML) de la mise en forme (qui est devenu le CSS).
- Les choses s'accélérent en 1978, des personnes se réunissent pour créer un "langage informatique standard" riche et évolutif.
- En 1986, le **SGML** finit par naître. C'est l'abréviation de *Standard Generalized Markup Language*. Effectivement, c'est un langage puissant, mais trop complexe. Pour tout vous dire, sa documentation comportait 1290 pages ! Pas de quoi motiver le grand public à son utilisation... Le SGML ne connaîtra donc jamais de grande gloire 😞
- En 1991, un certain Tim Berners-Lee invente le World Wide Web (abrégez WWW). Pour créer les premières pages web, Tim Berners-Lee crée le **langage HTML** qui s'inspire un peu du SGML (comme quoi il aura servi à quelque chose celui-là 😊)

Le HTML (abréviation de *HyperText Markup Language*) est diffusé gratuitement (merci Tim !).



Tim Berners-Lee

- A peine 2 ans plus tard, un des premiers navigateurs gratuits apparaît : il s'appelle Mozaïc et fonctionne aussi bien sur Mac que sur PC.  
Entre 1993 et 1994, le nombre de sites web passe de 500 à 10 000, la croissance est énorme. Déjà, le Web semble être promis à un grand avenir.
- C'est en Europe que les choses commencent à se précipiter. Le Web connaît un essor fulgurant, et il n'y a pas vraiment d'organisme chargé de surveiller son évolution. Il faut vérifier que ça ne parte pas dans tous les sens, il faut que les gens travaillent ensemble pour assurer l'évolution du Web (et du HTML !)
- Tim Berners-Lee, celui qui a inventé le Web, crée en 1994 le **W3C** (abréviation de *World Wide Web Consortium*). Il en devient le directeur.  
Le W3C est un organisme dont la mission va être de "surveiller" l'évolution du web : il faut éviter que certaines grosses entreprises tentent de stopper son évolution pour des raisons commerciales. Le W3C aura aussi la tâche de proposer au fur et à mesure de nouvelles versions du HTML, justement afin que le Web évolue.



Le logo du W3C

- Il est à noter que, dès le début (1997), le nombre de français travaillant au W3C est important. Mais, petit à petit des universités un peu partout dans le monde vont contribuer au W3C, notamment les japonais.
- En 2003, ce sont plusieurs organismes qui travaillent pour le W3C. Il y a environ 450 membres.  
L'ERCIM (*European Research Consortium for Informatics and Mathematics*) remplace l'INRIA (un institut français) dans la direction du W3C. L'ERCIM est situé en France lui aussi, à Sophia Antipolis (près de Nice) et va permettre de donner plus de poids au W3C, encore aujourd'hui parfois royalement ignoré par les Webmasters (dont de nombreux sont des "professionnels").

Comme vous pouvez le voir, tout au long de cette petite histoire des gens ont fait attention à ce que le web se développe

correctement et qu'il ne soit pas le joujou de grosses entreprises multinationales.

Aujourd'hui encore (et plus que jamais), le W3C a la mission d'assurer un avenir au Web. Nous allons justement voir quel a été son travail sur le langage HTML, et pourquoi *on parle aujourd'hui de standards avec le XHTML et le CSS*.

---

## Du HTML au XHTML

Comme je vous l'ai dit dans ma petite histoire, dès les débuts du Web Tim Berners-Lee crée la première version du langage HTML.

Car en effet, comme pour un programme informatique, le langage HTML a connu plusieurs versions :

- **HTML 1.0** : c'est la toute première version créée par Tim Berners-Lee en 1991.
- **HTML 2.0** : la deuxième version du HTML apparaît en 1994 et se finira en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des prochaines versions du HTML. Les règles et le fonctionnement de cette version sont données par le W3C (tandis que la première version a été créée par un seul homme).
- **HTML 3.0** : apparue en 1996, cette nouvelle version du HTML rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images etc...
- **HTML 4.0** : c'est une des dernières versions du HTML (la toute dernière étant une version légèrement modifiée, le 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 l'utilisation de feuilles de style, notre fameux CSS !
- **XHTML 1.0** : début 2000, le W3C décide de mettre un terme au joyeux bordel qu'est devenu le langage HTML. En effet, au fur et à mesure de son évolution, des balises HTML ont été "inventées" par les navigateurs Netscape et Internet Explorer (de Microsoft). Certaines balises marchaient sur un navigateur, mais pas sur l'autre ><  
On décide d'arrêter le développement du langage HTML et d'en créer un nouveau. Le **XHTML** (*Extensible HyperText Markup Language*) devient alors le **standard** : ce langage doit fonctionner de la même manière sur tous les navigateurs, et pas le droit de créer de nouvelles balises tant que le W3C ne l'a pas autorisé !  
Concrètement, par rapport au HTML il y a assez peu de différences, mais le langage est plus "Strict" : vous n'avez pas autant le droit à l'erreur qu'en HTML. Ça ne veut pas dire pour autant qu'il est plus difficile à utiliser, c'est juste une habitude à prendre.



Dans le cours de création de site web du Site du Zéro que vous êtes en train de lire, je ne parle que du XHTML car c'est le langage qu'il faut aujourd'hui utiliser. Je vous ai donc enseigné les bonnes habitudes dès le début, ce qui va vous faciliter grandement la création de vos sites web. Ceux qui ont commencé par apprendre le HTML ont en effet parfois quelques mauvaises vieilles habitudes.

C'est désormais l'avenir du Web, et il va permettre de créer des sites web consultables sur toutes sortes de PDA, téléphones multimédia et autres produits transportables qui vont se multiplier dans l'avenir.

Vous imaginez s'il y avait 35 formats de CD Audio différents ? Votre lecteur de CD pourrait en lire certains, mais pas tous... Je vous dis pas le bordel que ça serait. Eh bien c'est pareil pour le XHTML : des gens se sont réunis et se sont mis d'accord pour dire comment doit fonctionner une page web. Au final, tout le monde y gagne : vous (le webmaster) et eux (vos visiteurs). Créer un standard, ça permet donc de s'assurer que tout le monde parle et comprend le même langage. C'est toujours en inventant des standards (et pas seulement en informatique) qu'on arrive à faire évoluer les choses ensemble 😊

## Et le CSS ?

L'histoire du CSS débute, elle, en 1996. On revient à l'idée lancée par le monsieur dont je vous ai parlé au tout début de cette annexe (William Tunnicliffe) : il faut séparer le contenu de la mise en page. Cela apporte de nombreux avantages : l'apparence du site web pourra être plus facilement mise à jour, les pages seront plus rapides à charger, on pourra proposer plusieurs designs aisément etc...

- **CSS 1** : dès 1996, la première version du CSS est utilisable. 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, cette nouvelle version de CSS rajoute de nombreuses options très intéressantes. On peut désormais utiliser le positionnement absolu, les pseudo-formats :before et :after etc etc...

## Votre site est-il valide ?

Pour la dernière partie de ce chapitre, on arrête un peu le blabla et on retourne à la technique.

Ce que je vous ai dit jusqu'ici avait pour but de vous faire connaître un peu mieux l'histoire du web, mais aussi de vous sensibiliser à ce qui va suivre...



Qu'est-ce que c'est que cette histoire de validité ?

Il y a des sites qui sont valides et d'autres qui ne le sont pas ?

Souvenez-vous : le W3C a établi des normes. Il est nécessaire de les respecter, pour qu'on soit sûrs que tous les sites web parlent la même "langue".

Le W3C propose sur son site web ([www.w3.org](http://www.w3.org)) un outil qui s'appelle le "Valideur" ("Validator" en anglais :p).

Le valideur est une sorte de programme qui va analyser votre code source et vous dire s'il est correctement fait, ou s'il comporte des erreurs que vous devez corriger.

Il existe 2 valideurs différents :

- Un valideur (x)HTML.
- Un valideur CSS.

Nous allons apprendre à nous servir des 2 en commençant par le valideur XHTML.

Pour information, le valideur XHTML risque de vous montrer plusieurs erreurs sur votre page web. Ne vous inquiétez pas, c'est normal la première fois 😊

En revanche, le valideur CSS ne nous posera que peu de problèmes, il est plus rare de faire des erreurs en CSS qu'en XHTML.

### Le valideur XHTML

Commençons par le gros morceau voulez-vous 😊

Tout d'abord, commencez par mettre cette page en favoris, c'est l'adresse du Valideur XHTML : <http://validator.w3.org/>.  
Ceux qui ont téléchargé le plugin "Web Developer" pour Firefox peuvent d'ailleurs effectuer une validation automatiquement via un menu spécial.

La page que vous avez sous les yeux est très simple. Elle vous propose 2 façons de valider :

- Soit vous validez une page XHTML qui se trouve déjà sur le Web. Dans ce cas, il suffit d'indiquer l'URL de la page (par exemple : "http://www.monsite.com/page.html").
- Soit vous n'avez pas encore envoyé vos pages sur le Web (elles se trouvent donc toujours sur votre disque dur). Vous avez la possibilité d'envoyer directement la page au Valideur depuis votre disque pour qu'il l'analyse.

Dans les 2 cas, le résultat sera le même, il n'y a aucune différence.

Si tout se passe bien (vous avez beaucoup de chance 😊), vous verrez le message suivant :

**This Page Is Valid XHTML 1.0 Strict!**



Le site vous proposera alors de mettre l'icône ci-contre sur votre site web pour montrer que vos pages sont conformes aux normes, et donc que vous respectez les règles établies.

Ce n'est pas du tout une obligation, si vous trouvez ce logo trop moche rien ne vous impose de le mettre 😊

Bien entendu, si votre site web comporte plus d'une page (ce qui est très très fréquent ^^), il faudra tester chacune d'elles.

Au SEEECOUUUUUUURS !!! 🤖



Ma page web n'est pas valide, je vais pas m'en sortir je suis cerné par les erreurs, faites quelque chose aidez mmmoi!!!!!! 🤔

*C'était la réaction classique du débutant qui découvre que sa page web qu'il croyait parfaite ne l'est finalement pas tellement*

Tout d'abord, une chose à bien vous mettre dans la tête : *ce n'est pas parce que votre page web s'affiche correctement qu'elle ne comporte pas d'erreurs.*

Votre page web peut être toute belle et comporter pas mal d'erreurs.



Quel intérêt de les corriger alors ?

Je n'ai pas arrêté de vous le rappeler : si vous dites que votre page web parle le "XHTML 1.0", il faut qu'elle le parle correctement pour que tout le monde comprenne. Il faut savoir que les navigateurs "essaient" de ne pas afficher les erreurs lorsqu'ils en rencontrent pour ne pas perturber l'internaute, mais rien ne vous dit que d'autres navigateurs ne vont pas se comporter bizarrement !

Avoir une page web valide, c'est donc avoir la possibilité de dormir tranquille en sachant que l'on a bien fait les choses comme il faut.

De plus, et c'est vérifié, une page web correctement construite aura plus de chances d'être mieux positionnée dans les résultats de recherche de Google, ce qui vous amènera... plus de visiteurs !

Je ne vais pas vous lister toutes les erreurs possibles, ce serait trop long. En revanche, je vais vous faire une liste des erreurs les plus courantes. Vous vous apercevrez certainement que vous en avez commises quelques-unes, et vous saurez comment les corriger.

- Tout d'abord, ça peut paraître évident parce que je l'ai dit au début, mais il faut le rappeler : les noms de vos balises et attributs doivent être en **minuscules**. La valeur d'un attribut, en revanche, peut comporter des majuscules :  
`<balise attribut="Valeur de l'attribut">`
- Les balises seules, comme le retour à la ligne "br" et la balise d'image "img" doivent obligatoirement **comporter un slash (/)** à la fin : `<br />`  
 Notez que l'espace entre le "br" et le slash n'est pas obligatoire, mais le W3C recommande de le mettre, alors en gentil garçon sage je le mets 😊
- Tous vos textes doivent être dans des balises de paragraphes. Il est interdit de mettre du texte directement entre les balises `<body></body>` sans l'avoir entouré des fameux `<p></p>`  
 Ceci est aussi valable pour les retours à la ligne `<br />` et les images `<img />` : **ces balises doivent obligatoirement se trouver entre deux balises `<p></p>` !**  
 C'est une erreur ultra-courante chez les débutants. Voici un exemple de ce qu'il ne faut pas faire :

#### Code : HTML

```
<p>Ceci est un texte correctement placé dans un paragraphe.<br />
Les balises "br" doivent se trouver à l'intérieur d'un
paragraphe, ne l'oubliez pas</p>

Ceci est un texte en-dehors d'un paragraphe. C'est interdit.<br />
De même, les sauts de lignes "br" sont interdits en dehors d'un
paragraphe, ainsi que les images.


<p>Ce texte est à nouveau dans une balise de paragraphe, là
tout va bien.<br />
N'oubliez pas que les images aussi doivent se trouver dans un
paragraphe.</p>

<p>
(Cette image est bien placée dans un paragraphe)</p>
```



- **Toutes vos images doivent comporter un attribut "alt"** qui indique ce que contient l'image :  

```

```

 Si, par hasard, votre image est purement décorative (vous ne pouvez pas en trouver de description), vous êtes autorisés à ne rien mettre comme valeur pour l'attribut alt, comme ceci :  

```

```
- Vos balises doivent être **fermées dans l'ordre**. Je m'explique, voici ce qu'il ne faut pas faire :  

```
<p>Texte <em>important</p></em>
```

 Maintenant, voici la bonne façon de faire :  

```
<p>Texte <em>important</em></p>
```

 Gardez bien ce schéma en tête, beaucoup de débutants font cette erreur.
- Si vos liens comportent des & (c'est le cas des sites utilisant le langage PHP), vous devez taper le code `&amp;` à la place pour éviter une confusion au navigateur. Voici le mauvais exemple :  

```
<a href="http://www.site.com/?jour=15&mois=10&an=2000">
```

 Voici le bon exemple, il m'a suffit de transformer les & en `&amp;` :  

```
<a href="http://www.site.com/?jour=15&amp;mois=10&amp;an=2000">
```
- Vérifiez enfin que vous n'avez utilisé que des balises XHTML **et non pas de vieilles balises HTML**. Si vous n'utilisez que les balises présentées dans ce cours, vous n'aurez aucun problème. Si vous avez utilisé des balises étrangères (comme `<marquee>`) ou obsolètes (comme le vieux `<frame>`), votre page contiendra des erreurs. Le validateur vous dira "Element *bidule* undefined" ou encore "There is no attribute *truc*"

## Le validateur CSS

Là je vous rassure, ça va être plus rapide 😊

Tout d'abord, voici l'URL du validateur CSS à mettre en favoris :

<http://jigsaw.w3.org/css-validator/>

Vous devez indiquer l'adresse où se trouve le fichier CSS que vous voulez valider. Vous pouvez aussi, comme pour les pages XHTML, envoyer directement depuis votre disque dur vos CSS.

Notez que, comme pour les fichiers XHTML, si vous avez plusieurs fichiers CSS vous devez tous les valider.



**Vous n'êtes pas obligé d'indiquer le fichier CSS directement, vous pouvez donner l'url de votre page XHTML (.html) et le validateur ira directement chercher tous les fichiers CSS que vous utilisez (grâce à la balise `<link />` !)**

Il faut savoir qu'il est nécessaire de passer d'abord par le validateur XHTML, de corriger vos erreurs, puis de passer au validateur CSS.

Le validateur CSS refusera probablement de se lancer si votre page XHTML comporte encore des erreurs.

Bon, en général la validation des CSS est une vraie partie de plaisir (ça change un peu du validateur XHTML :p).

Pour avoir des erreurs, il faut que vous vous soyez complètement plantés quelque part ou que vous ayez utilisé des propriétés CSS non officielles. Voici les quelques rares erreurs que vous pouvez avoir fait :

- Vous avez oublié de **terminer une ligne d'une propriété CSS par un point-virgule (;)**. Si vos propriétés CSS ne se terminent pas par un point-virgule, votre fichier CSS est inexploitable.
- Vous avez oublié de **donner un nom de police standard à la fin de la propriété *font-family***. Pas bien :  

```
font-family:Arial, "Trebuchet MS", Impact;
```

 Bien : 

```
font-family:Arial, "Trebuchet MS", Impact, sans-serif;
```
- Vous avez utilisé **une propriété CSS non officielle**. C'est le cas de Microsoft qui a inventé seul dans son coin des propriétés CSS qui ne marchent que sur Internet Explorer. Parmi ces propriétés inventées, signalons `scrollbar-face-color` et compagnie (couleur de la barre de défilement) et les filtres comme `filter:glow()`. Ces propriétés n'étant pas officielles, vous ne devez pas les utiliser sur votre site web.



Vous avez réussi à créer des fichiers CSS valides ? Bravo ! (C'était pas bien dur :p)  
 Vous pouvez mettre le logo "Valide CSS" ci-contre pour montrer que vous utilisez des feuilles de styles qui respectent les normes. Là encore, rien ne vous empêche de créer le vôtre si vous trouvez que celui-ci est moche, ou même de ne rien mettre du tout si ça ne fait pas joli dans le design de votre site web.

---

J'espère que cette annexe vous a plu et que vous y avez appris de nouvelles choses intéressantes.

... Bon ok ça fait un peu bateau comme phrase de conclusion 😊

Plus sérieusement, c'est une annexe à ne pas prendre à la légère : j'ai essayé d'y résumer tous les enjeux du Web, son développement, son histoire. Une histoire dont VOUS allez faire partie désormais en tant que Webmaster.

Il est important que vos sites soient valides XHTML et CSS, que vous respectiez ces normes afin de pouvoir aider le Web à évoluer.

Et au-delà du respect des standards, c'est aussi le respect de l'internaute que vous devez avoir en tête. Mettre une petite publicité pour aider à financer votre site, ok. Mais noyer le visiteur de popups et javascripts en tous genres, s'il vous plaît non. N'êtes vous pas parfois agacés sur certains sites un peu trop envahissants ? Mettez-vous à la place des internautes, créez le site que vous auriez envie de visiter.

Si vous respectez toutes ces consignes, vous n'avez pas à vous en faire : le succès de votre site suivra 😊

Ce n'est qu'une question de temps et de persévérance 😊

---

## Liste des balises XHTML

Cette page est une liste non exhaustive des balises XHTML qui existent.

Vous trouverez ici un grand nombre de balises XHTML, certaines qu'on a vues dans le cours, d'autres qu'on n'a pas eu l'occasion d'étudier. Généralement, les balises qu'on n'a pas étudiées sont des balises plus rarement utilisées. Peut-être trouverez-vous votre bonheur dans ce lot de nouvelles balises 😊

Vous pouvez vous servir de cette page comme page d'aide lorsque vous développez votre site web 😊



Attention j'insiste : cette page n'est pas complète et c'est volontaire. Je préfère mettre *moins* de balises et garder seulement celles qui me semblent les plus utiles dans la pratique.

---

## Balises de premier niveau

Les balises de premier niveau sont les principales balises qui structurent une page XHTML. Elles sont indispensables pour réaliser le "code minimal" d'une page web.

Balises	Description
<html>	<p>Balise principale de toute page web. On lui donne généralement 2 attributs :</p> <ul style="list-style-type: none"><li>xmlns : la liste des balises xhtml existantes (appelée <i>espace de noms</i>).</li><li>xml:lang : la langue utilisée sur votre page web. Utilisez "fr" pour un document en français.</li></ul> <p>En temps normal, votre balise &lt;html&gt; doit ressembler à ceci :</p> <p><b>Code : HTML</b></p> <pre>&lt;html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"&gt; &lt;/html&gt;</pre>
<head>	En-tête de la page
<body>	Corps de la page

## Le code minimal d'une page XHTML

Vous trouverez ci-dessous le code minimal de toute page web XHTML.

### Code : HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
    <title>Titre du site</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    </body>
</html>
```

## Balises d'en-tête

Ces balises sont toutes situées dans l'en-tête de la page web, c'est-à-dire entre <head> et </head>

Balise	Description
<link />	<p>Cette balise permet d'indiquer certaines informations sur la page web. On l'utilise le plus souvent pour inclure une feuille de style CSS, comme ceci :</p> <p><b>Code : HTML</b></p> <pre>&lt;link rel="stylesheet" media="screen" type="text/css" title="Mon design" href="design.css" /&gt;</pre> <p>On peut aussi s'en servir pour 2-3 autres choses :</p> <p><b>Code : HTML</b></p> <pre>&lt;!-- Page d'accueil du site --&gt; &lt;link rel="start" title="Accueil" href="index.html" /&gt; &lt;!-- Page d'aide du site --&gt; &lt;link rel="help" title="Politique d'accessibilité" href="accessibilite.html" /&gt; &lt;!-- Fil RSS du site --&gt; &lt;link rel="alternate" type="application/rss+xml" title="News de mon site" href="news.xml" /&gt; &lt;!-- Icône du site (favicon) --&gt; &lt;link rel="shortcut icon" type="image/x-icon" href="favicon.ico" /&gt;</pre> <p>La favicon est une icône qui s'affiche généralement à gauche de l'adresse de votre site sur le navigateur de vos visiteurs. C'est un moyen de personnaliser un peu plus son site. Quant au fil RSS, il s'agit d'une technique permettant à vos visiteurs de suivre l'actualité de votre site depuis un logiciel spécial (un navigateur tel que Firefox le fait d'ailleurs). En général on génère des fils RSS en PHP (si vous ne faites que du XHTML / CSS ça ne vous intéresse donc pas pour le moment).</p>
<meta />	<p>Cette balise permet de définir les propriétés de la page web. On s'en sert pour une foule de choses. Voici quelques exemples pratiques :</p> <p><b>Code : HTML</b></p> <pre>&lt;!-- Auteur de la page --&gt; &lt;meta name="author" content="Jean Dupont" /&gt; &lt;!-- Description de la page --&gt; &lt;meta name="description" content="La page personnelle de Jean Dupont" /&gt; &lt;!-- Mots-clés de la page --&gt; &lt;meta name="keywords" content="expériences, recherche, laboratoire, chimie" /&gt; &lt;!-- Adresse de contact --&gt; &lt;meta name="reply-to" content="monadresse@email.com" /&gt; &lt;!-- Empêcher la mise en cache de la page par le navigateur --&gt; &lt;meta http-equiv="pragma" content="no-cache" /&gt; &lt;!-- Table de caractères --&gt; &lt;meta http-equiv="content-type" content="text/html;</pre>

	<pre> charset=iso-8859-1" /&gt; &lt;!-- Rafraîchissement automatique au bout de 10 secondes --&gt; &lt;meta http-equiv="refresh" content="10; URL=http://www.monsite.com" /&gt; </pre> <p>En général, on utilise surtout le meta pour la table de caractères.</p> <p>La description et les mots-clés de la page n'influencent pratiquement plus les moteurs de recherche. Inutile donc de passer du temps à mettre tout plein de mots là-dedans 🤪</p>
<script>	<p>Permet de placer un script. On l'utilise souvent pour mettre du code Javascript :</p> <p><b>Code : HTML</b></p> <pre> &lt;script type="text/javascript"&gt; /* Votre script ici */ &lt;/script&gt; </pre>
<style>	<p>Permet de définir du code CSS pour la page. On lui met l'attribut <code>type="text/css"</code>.</p> <p>Exemple :</p> <p><b>Code : HTML</b></p> <pre> &lt;style type="text/css"&gt; /* Votre code CSS ici */ &lt;/style&gt; </pre>
<title>	<p>Titre de la page web.</p> <p>C'est probablement la balise la plus importante d'une page web. Choisissez bien votre titre car il a beaucoup d'importance pour les moteurs de recherche (ils donnent de l'importance aux mots qui se trouvent dans le titre).</p> <p><b>Code : HTML</b></p> <pre> &lt;title&gt;Les petites expériences chimiques de M. Dupont&lt;/title&gt; </pre>

## Balises de structuration du texte

Balise	Type	Description
<code>&lt;acronym&gt;</code>	<i>Inline</i>	<p>Sert à définir des acronymes, comme C.I.A. On utilise généralement l'attribut <i>title</i> pour donner la définition de l'acronyme quand on pointe dessus :</p> <p><b>Code : HTML</b></p> <pre>&lt;acronym title="Central Intelligence Agency"&gt;C.I.A.&lt;/acronym&gt;</pre>
<code>&lt;blockquote&gt;</code>	<i>Block</i>	<p>Citation (longue) Vous devez obligatoirement mettre une balise de paragraphe à l'intérieur du blockquote. Par exemple :</p> <p><b>Code : HTML</b></p> <pre>&lt;blockquote&gt;   &lt;p&gt;     Texte de la citation   &lt;/p&gt; &lt;/blockquote&gt;</pre>
<code>&lt;cite&gt;</code>	<i>Inline</i>	Citation (moyenne)
<code>&lt;q&gt;</code>	<i>Inline</i>	Citation (courte)
<code>&lt;sup&gt;</code>	<i>Inline</i>	Mise en exposant
<code>&lt;sub&gt;</code>	<i>Inline</i>	Mise en indice
<code>&lt;strong&gt;</code>	<i>Inline</i>	<p>Mise en valeur (forte) Le texte est généralement mis en gras.</p>
<code>&lt;em&gt;</code>	<i>Inline</i>	<p>Mise en valeur (faible) Le texte est généralement mis en italique.</p>
<code>&lt;h6&gt;</code>	<i>Block</i>	Titre de niveau 6
<code>&lt;h5&gt;</code>	<i>Block</i>	Titre de niveau 5
<code>&lt;h4&gt;</code>	<i>Block</i>	Titre de niveau 4
<code>&lt;h3&gt;</code>	<i>Block</i>	Titre de niveau 3
<code>&lt;h2&gt;</code>	<i>Block</i>	Titre de niveau 2
<code>&lt;h1&gt;</code>	<i>Block</i>	Titre de niveau 1
<code>&lt;img /&gt;</code>	<i>Inline</i>	<p>Insère une image. Utilisez les attributs <i>src</i> (pour indiquer l'adresse de l'image) et <i>alt</i> (pour indiquer un texte de remplacement). Ces 2 attributs sont obligatoires. Exemple :</p> <p><b>Code : HTML</b></p> <pre>&lt;img src="Templates/images/smiley.png" alt=":) " /&gt;</pre>

<a>	<i>Inline</i>	<p>Lien hypertexte. Indiquez l'url de destination grâce à l'attribut <i>href</i> :</p> <p><b>Code : HTML</b></p> <pre>&lt;a href="autrepage.html"&gt;Rendez-vous sur l'autre page&lt;/a&gt;</pre>
 	<i>Inline</i>	Retour à la ligne
<p>	<i>Block</i>	Paragraphe
<hr />	<i>Block</i>	Crée une ligne de séparation horizontale
<address>	<i>Block</i>	Permet d'indiquer une adresse, ou éventuellement l'auteur d'un document. Le texte est généralement mis en italique.
<del>	<i>Inline</i>	Permet d'indiquer un texte qui a été supprimé. Le texte est généralement barré.
<ins>	<i>Inline</i>	Permet d'indiquer un texte qui a été inséré. Le texte est généralement souligné.
<dfn>	<i>Inline</i>	Permet d'indiquer une définition.
<kbd>	<i>Inline</i>	Permet d'indiquer un code que doit taper le visiteur.
<pre>	<i>Block</i>	Le texte à l'intérieur de la balise <pre> sera affiché tel qu'il a été tapé dans le code (espaces et entrées compris). Une police de taille fixe est utilisée.

## Balises de liste

Cette partie énumère toutes les balises XHTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions...)

Balise	Type	Description
<code>&lt;ul&gt;</code>	<i>Block</i>	<p>Liste à puces non numérotée. Vous devez mettre un <code>&lt;li&gt;&lt;/li&gt;</code> par élément de la liste. Exemple :</p> <p><b>Code : HTML</b></p> <pre>&lt;ul&gt; &lt;li&gt;Un élément&lt;/li&gt; &lt;li&gt;Un autre élément&lt;/li&gt; &lt;/ul&gt;</pre>
<code>&lt;ol&gt;</code>	<i>Block</i>	<p>Liste à puces numérotée. Vous devez mettre un <code>&lt;li&gt;&lt;/li&gt;</code> par élément de la liste. Exemple :</p> <p><b>Code : HTML</b></p> <pre>&lt;ol&gt; &lt;li&gt;Elément n°1&lt;/li&gt; &lt;li&gt;Elément n°2&lt;/li&gt; &lt;/ol&gt;</pre>
<code>&lt;li&gt;</code>	<i>list-item</i>	<p>Permet de créer un élément de liste. Le type de la balise est particulier car elle n'est ni block ni inline. On dit qu'elle est de type <i>list-item</i>.</p>
<code>&lt;dl&gt;</code>	<i>Block</i>	<p>Liste de définitions. Vous devez alterner chaque terme <code>&lt;dt&gt;</code> par sa définition <code>&lt;dd&gt;</code>. Exemple :</p> <p><b>Code : HTML</b></p> <pre>&lt;dl&gt; &lt;dt&gt;Porte&lt;/dt&gt; &lt;dd&gt;Ouverture dans un mur permettant d'entrer et de sortir&lt;/dd&gt; &lt;dt&gt;Théâtre&lt;/dt&gt; &lt;dd&gt;Lieu où l'on représente des ouvrages dramatiques&lt;/dd&gt; &lt;/dl&gt;</pre>
<code>&lt;dt&gt;</code>	<i>Block</i>	Terme à définir
<code>&lt;dd&gt;</code>	<i>Block</i>	Définition du terme



## Balises de tableau

Balise	Type	Description
<code>&lt;table&gt;</code>	<i>Block</i>	<p>Délimite un tableau. Voici un exemple de tableau simple :</p> <p><b>Code : HTML</b></p> <pre> &lt;table&gt;   &lt;caption&gt;Passagers du vol 377&lt;/caption&gt;    &lt;tr&gt;     &lt;th&gt;Nom&lt;/th&gt;     &lt;th&gt;Age&lt;/th&gt;     &lt;th&gt;Pays&lt;/th&gt;   &lt;/tr&gt;    &lt;tr&gt;     &lt;td&gt;Carmen&lt;/td&gt;     &lt;td&gt;33 ans&lt;/td&gt;     &lt;td&gt;Espagne&lt;/td&gt;   &lt;/tr&gt;   &lt;tr&gt;     &lt;td&gt;Michelle&lt;/td&gt;     &lt;td&gt;26 ans&lt;/td&gt;     &lt;td&gt;Etats-Unis&lt;/td&gt;   &lt;/tr&gt;   &lt;tr&gt;     &lt;td&gt;François&lt;/td&gt;     &lt;td&gt;43 ans&lt;/td&gt;     &lt;td&gt;France&lt;/td&gt;   &lt;/tr&gt; &lt;/table&gt; </pre>
<code>&lt;caption&gt;</code>	-	Permet de donner un titre au tableau
<code>&lt;tr&gt;</code>	-	Ligne de tableau
<code>&lt;th&gt;</code>	-	Cellule d'en-tête du tableau (généralement mise en gras)
<code>&lt;td&gt;</code>	-	Cellule du tableau
<code>&lt;thead&gt;</code>	-	<p>Balise non obligatoire permettant d'insérer l'en-tête du tableau. Si vous choisissez d'utiliser <code>&lt;thead&gt;</code>, <code>&lt;tfoot&gt;</code> et <code>&lt;tbody&gt;</code>, vous devez les mettre dans l'ordre suivant dans votre code source :</p> <ol style="list-style-type: none"> <li>1. <code>&lt;thead&gt;</code></li> <li>2. <code>&lt;tfoot&gt;</code></li> <li>3. <code>&lt;tbody&gt;</code></li> </ol>
<code>&lt;tbody&gt;</code>	-	Balise non obligatoire permettant d'insérer le corps du tableau
<code>&lt;tfoot&gt;</code>	-	Balise non obligatoire permettant d'insérer le pied du tableau

## Balises de formulaire

Balise	Type	Description
<form>	Block	<p>Délimite un formulaire. Vous devrez généralement donner 2 attributs à la balise &lt;form&gt;</p> <ul style="list-style-type: none"> <li><b>method</b> : indique la méthode d'envoi du formulaire (get ou post). Si vous ne savez pas quoi utiliser, mettez post.</li> <li><b>action</b> : la page vers laquelle le visiteur doit être redirigé lorsqu'il a validé votre formulaire.</li> </ul>
<fieldset>	Block	<p>Permet de regrouper plusieurs éléments d'un formulaire. On l'utilise généralement dans de grands formulaires.</p> <p>Pour donner un titre à votre groupe, utilisez la balise &lt;legend&gt;</p>
<legend>	Inline	<p>Titre d'un groupe dans un formulaire. A utiliser à l'intérieur d'un &lt;fieldset&gt;</p>
<label>	Inline	<p>Titre d'un élément de formulaire. Généralement, vous devrez mettre l'attribut <i>for</i> sur cette balise pour indiquer l'ID de l'élément auquel correspond le label.</p>
<input />	Inline	<p>Champ de formulaire. Il existe de nombreux types de champs différents. Vous choisissez le type de champ que vous désirez grâce à l'attribut <i>type</i> :</p> <p style="text-align: center;"><b>Code : HTML</b></p> <pre> &lt;!-- Zone de texte d'une ligne --&gt; &lt;input type="text" /&gt; &lt;!-- Mot de passe (le texte est caché) --&gt; &lt;input type="password" /&gt; &lt;!-- Envoi de fichier --&gt; &lt;input type="file" /&gt; &lt;!-- Case à cocher --&gt; &lt;input type="checkbox" /&gt; &lt;!-- Bouton d'option --&gt; &lt;input type="radio" /&gt; &lt;!-- Bouton --&gt; &lt;input type="button" /&gt; &lt;!-- Bouton d'envoi --&gt; &lt;input type="submit" /&gt; &lt;!-- Bouton de remise à zéro --&gt; &lt;input type="reset" /&gt; &lt;!-- Champ caché --&gt; &lt;input type="hidden" /&gt; </pre> <p>Pensez à donner un nom à vos champs grâce à l'attribut <i>name</i></p>
<textarea>	Inline	<p>Zone de saisie multiligne. Vous pouvez définir sa taille grâce aux attributs <i>rows</i> et <i>cols</i> (nombre de lignes et colonnes) ou bien le faire en CSS grâce aux propriétés <i>width</i> et <i>height</i>.</p>
<select>	Inline	<p>Liste déroulante. Utilisez la balise &lt;option&gt; pour créer chaque élément de la liste :</p> <p style="text-align: center;"><b>Code : HTML</b></p> <pre> &lt;select name="pays"&gt; </pre>

		<pre>&lt;option value="france"&gt;France&lt;/option&gt; &lt;option value="espagne"&gt;Espagne&lt;/option&gt; &lt;option value="italie"&gt;Italie&lt;/option&gt; &lt;/select&gt;</pre>
<option>	<i>Block</i>	Element d'une liste déroulante
<optgroup>	<i>Block</i>	Groupe d'éléments d'une liste déroulante. A utiliser dans le cas d'une grande liste déroulante. Vous devez utiliser l'attribut <i>label</i> pour donner un nom au groupe.

## Balises génériques

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises XHTML ont un **sens** : <p> signifie "Paragraphe", <h2> signifie "Sous-titre" etc.

Parfois, on a besoin d'utiliser des balises génériques (aussi appelées *balises universelles*) car aucune des autres balises ne convient. On utilise le plus souvent des balises génériques pour construire son design.

Il y a 2 balises génériques : l'une est inline, l'autre est block.

Balise	Type	Description
<span>	<i>Inline</i>	Balise générique de type inline
<div>	<i>Block</i>	Balise générique de type block

Ces balises ont un intérêt uniquement si vous leur donnez un attribut *class*, *id* ou *style* :

- **class** : indique le nom de la classe CSS à utiliser.
- **id** : donne un nom à la balise. Ce nom doit être unique sur toute la page car il permet d'identifier la balise. Vous pouvez vous servir de l'ID pour de nombreuses choses, comme par exemple pour un lien vers une ancre, pour un style CSS de type ID, pour des manipulations en Javascript etc.
- **style** : cet attribut vous permet d'indiquer directement le code CSS à appliquer. Vous n'êtes donc pas obligés d'avoir une feuille de style à part, vous pouvez juste mettre directement les attributs CSS. Notez qu'il est préférable de ne pas utiliser cet attribut et de passer à la place par une feuille de style externe car cela rend votre site plus facile à mettre à jour par la suite.

Ces 3 attributs ne sont pas réservés aux balises génériques : vous pouvez aussi les mettre sur la plupart des autres balises sans aucun problème 😊

---

Comme je vous l'ai dit au début de ce chapitre, il y a plusieurs balises que j'ai volontairement omises.

Vous l'aurez constaté, en XHTML tout est affaire de *sens* (on parle de *sémantique*). Ce qui compte, c'est d'utiliser la balise qui convient le mieux à chaque moment.

En théorie, on pourrait faire presque tout un site rien qu'avec les balises génériques <div> et <span> (en utilisant du CSS), mais votre site n'aurait aucun sens logique ! Or, respecter la logique de son code source est une chose que les webmasters considèrent comme fondamentale. Une page sémantique a plus de chances d'être mieux indexée dans Google qu'une page utilisant des balises inadaptées.

Souvenez-vous en ! 😊

---

## Liste des propriétés CSS

Cette page est une liste non exhaustive des propriétés CSS qui existent.

Le but est de **réunir sur une même page un maximum de propriétés CSS**. Pour la plupart, ce sont des propriétés que nous avons vues dans le cours, mais vous trouverez aussi sûrement quelques nouvelles propriétés que nous n'avons pas eu le temps d'aborder. Ce sont généralement des propriétés plus rares mais faciles à utiliser.

La liste est non exhaustive car mon but n'est pas de faire la liste de toutes les propriétés CSS qui peuvent exister. Il y en a trop, et certaines sont très rarement utilisées.

Je préfère donc me concentrer sur les principales propriétés, ce qui en fait déjà un petit paquet quand même 😊

---

## Propriétés de formatage de texte

Je résume ici la plupart des propriétés de **formatage de texte**.

Qu'est-ce que le formatage de texte ? C'est tout ce qui consiste à mettre en forme le texte : mettre en gras, italique, souligné, changer la police, l'alignement etc...

### Police, taille et décorations

Type	Nom	Valeurs possibles
Nom de police	font-family	<p>Indiquer les noms de polices possibles par ordre de préférence :</p> <p><b>Code : CSS</b></p> <pre>font-family: police1, police2, police3;</pre> <p>Si le visiteur a la police 1, il l'utilisera. Sinon, il regarde s'il a la police 2, puis la police 3 etc. Utilisez des guillemets si le nom de la police comporte des espaces. Essayez de toujours mettre comme dernière police possible "serif" ou "sans-serif".</p> <p><b>Code : CSS</b></p> <pre>font-family: "Arial Black", Arial, Verdana, serif;</pre>
Taille du texte	font-size	<p>Indiquez la taille du texte. Plusieurs unités sont possibles :</p> <ul style="list-style-type: none"> <li>• px (pixels)</li> <li>• % (pourcentage, 100% = normal)</li> <li>• em (taille relative, 1.0 = normal)</li> <li>• ex (taille relative à la hauteur de la lettre "x". 1.0 = normal)</li> <li>• nom de taille : <ul style="list-style-type: none"> <li>◦ xx-small : très très petit</li> <li>◦ x-small : très petit</li> <li>◦ small : petit</li> <li>◦ medium : moyen</li> <li>◦ large : grand</li> <li>◦ x-large : très grand</li> <li>◦ xx-large : très très grand</li> </ul> </li> </ul>
Gras	font-weight	<p><b>bold</b> : gras  <b>bolder</b> : plus gras  <b>lighter</b> : plus fin  <b>normal</b> : pas gras (par défaut)</p>
Italique	font-style	<p><b>italic</b> : italique  <b>oblique</b> : autre façon de mettre en italique  <b>normal</b> : normal (par défaut)</p>
		<p><b>underline</b> : souligné</p>

Décoration	text-decoration	<b>overline</b> : ligne au-dessus <b>line-through</b> : barré <b>blink</b> : clignotant <b>none</b> : normal (par défaut)
Petites capitales	font-variant	<b>small-caps</b> : petites capitales <b>normal</b> : normal (par défaut)
Capitales	text-transform	<b>uppercase</b> : tout mettre en majuscules <b>lowercase</b> : tout mettre en minuscules <b>capitalize</b> : début des mots en majuscules <b>none</b> : normal (par défaut)
Méga-propriété de police	font	<p>Indiquez dans n'importe quel ordre des valeurs possibles pour <i>font-weight</i>, <i>font-style</i>, <i>font-size</i>, <i>font-variant</i>, <i>font-family</i>.</p> <p><b>Attention exception</b> : le nom de la police (<i>font-family</i>) doit être placé en dernier dans la liste dans tous les cas.</p> <p>Vous n'êtes pas obligés de mettre une valeur de chacune de ces propriétés.</p> <p>Exemple :</p> <p><b>Code : CSS</b></p> <pre>font: bold 16px Arial;</pre> <p>Cela mettra votre texte en gras, 16 pixels, Arial.</p>

## Alignement

Type	Propriété	Valeurs possibles
Alignement horizontal	text-align	<b>left</b> : à gauche (par défaut) <b>center</b> : centré <b>right</b> : à droite <b>justify</b> : texte justifié (prend toute la largeur de la page)
Alignement vertical	vertical-align	<p>A utiliser dans des cellules de tableau, ou dans des éléments inline eux-mêmes contenus dans un élément inline.</p> <b>top</b> : en haut <b>middle</b> : au milieu <b>bottom</b> : en bas
Hauteur de ligne	line-height	Indiquer une valeur en pixels (px) ou en pourcentage (%)
Alinéa	text-indent	Indiquez une valeur en pixels (px) pour définir l'alinéa de vos paragraphes. Vos paragraphes commenceront avec le retrait que vous avez indiqué.
Césure	white-space	<b>normal</b> : le passage à la ligne est automatique (par défaut) <b>nowrap</b> : pas de passage à la ligne automatique, à moins qu'une balise xHTML comme <code>&lt;br /&gt;</code> ne soit présente. <b>pre</b> : le passage à la ligne se fait tel que le texte a été saisi dans le code source (comme la balise xHTML <code>&lt;pre&gt;</code> )

## Propriétés de couleur et de fond

### Couleur

Type	Propriété	Valeurs possibles
Couleur de texte	color	Indiquer une couleur avec l'une des méthodes suivantes : <ul style="list-style-type: none"> <li>En tapant le nom de la couleur en anglais (black, blue, green, white, red...).</li> <li>En indiquant la couleur en hexadécimal (#CC48A1)</li> <li>En indiquant la couleur en RGB : rgb (128, 255, 0)</li> </ul>
Couleur de fond	background-color	Même fonctionnement que <i>color</i> . Cela définit cette fois la couleur de fond du texte

### Image de fond

Type	Propriété	Valeurs possibles
Image de fond	background-image	Indiquer l'url de l'image (notation absolue ou relative) <b>Code : CSS</b> <pre>background-image:url("images/fond.png"); /* Notation relative */ background-image:url("http://www.monsite.com/images/fond.png"); /* Notation absolue */</pre>
Fond fixé	background-attachment	<b>fixed</b> : le fond reste fixe quand on descend plus bas sur la page <b>scroll</b> : le fond défile avec le texte (par défaut)
Répétition du fond	background-repeat	<b>repeat</b> : le fond se répète (par défaut) <b>repeat-x</b> : le fond ne se répète que sur une ligne, horizontalement <b>repeat-y</b> : le fond ne se répète que sur une colonne, verticalement <b>no-repeat</b> : le fond ne se répète pas, il n'est affiché qu'une fois
Position du fond	background-position	2 façons de faire : <ul style="list-style-type: none"> <li>En notant une distance en px ou %, par rapport au coin en haut à gauche. <b>Code : CSS</b> <pre>background-position:50px 200px; /* 50 px à droite, 200px en bas */</pre> </li> <li>En utilisant des valeurs prédéfinies, une pour la verticale et une pour l'horizontale : <ul style="list-style-type: none"> <li><b>top</b> : en haut, verticalement</li> <li><b>center</b> : au milieu, verticalement</li> <li><b>bottom</b> : en bas, verticalement</li> <li><b>left</b> : à gauche, horizontalement</li> <li><b>center</b> : au centre, horizontalement</li> <li><b>right</b> : à droite, horizontalement</li> </ul> </li> </ul>

		<b>Code : CSS</b>  <pre>background-position : bottom right; /* en bas à droite */</pre>
Méga-propriété de fond	background	<p>Indiquer une ou plusieurs valeurs issues des propriétés <i>background-image</i>, <i>background-repeat</i>, <i>background-attachment</i>, <i>background-position</i>. L'ordre des valeurs n'a pas d'importance et vous n'êtes pas obligés de mettre toutes les valeurs de ces propriétés (au moins une suffit)</p> <b>Code : CSS</b>  <pre>/* Le fond fond.png reste affiché en haut à droite de l'écran et n'est pas répété. */ background:url("images/fond.png") no-repeat fixed top right;</pre>



## Propriétés des boîtes

### Dimensions

Type	Propriété	Valeurs possibles
Largeur	width	Valeur en px, %, ou encore "auto" (valeur par défaut, la largeur dépendra du texte à l'intérieur)
Hauteur	height	Idem
Largeur minimale	min-width	Indiquer une valeur, en pixels par exemple.
Largeur maximale	max-width	Idem
Hauteur minimale	min-height	Idem
Hauteur maximale	max-height	Idem

### Marges extérieures

Type	Propriété	Valeurs possibles
Marge en haut	margin-top	Indiquer une valeur comme 20px, 1.5em...
Marge à gauche	margin-left	Idem
Marge à droite	margin-right	Idem
Marge en bas	margin-bottom	Idem
Méga-propriété de marge	margin	<p>Indiquez de 1 à 4 valeurs à la suite. Selon le nombre de valeurs que vous mettez, la signification change :</p> <ul style="list-style-type: none"> <li>1 valeur : ce sera la marge pour le haut, le bas, la gauche et la droite</li> <li>2 valeurs : la première correspond à la marge pour le haut et le bas, la seconde pour la gauche et la droite</li> <li>3 valeurs : la première correspond à la marge du haut, la seconde aux marges à gauche et à droite, la troisième à la marge du bas</li> <li>4 valeurs : respectivement la marge du haut, de la droite, du bas, de la gauche.</li> </ul> <p>Par exemple, si je mets 2 valeurs :</p> <p><b>Code : CSS</b></p> <pre>margin:20px 5px; /* 20px de marge en haut et en bas, 5px à gauche et à droite */</pre>

## Marges intérieures

Type	Propriété	Valeurs possibles
Marge intérieure en haut	padding-top	Indiquer une valeur comme 20px, 1.5em...
Marge intérieure à gauche	padding-left	Idem
Marge intérieure à droite	padding-right	Idem
Marge intérieure en bas	padding-bottom	Idem
Méga-propriété de marge intérieure	padding	<p>Indiquez de 1 à 4 valeurs à la suite. Selon le nombre de valeurs que vous mettez, la signification change :</p> <ul style="list-style-type: none"> <li>1 valeur : ce sera la marge pour le haut, le bas, la gauche et la droite</li> <li>2 valeurs : la première correspond à la marge pour le haut et le bas, la seconde pour la gauche et la droite</li> <li>3 valeurs : la première correspond à la marge du haut, la seconde aux marges à gauche et à droite, la troisième à la marge du bas</li> <li>4 valeurs : respectivement la marge du haut, de la droite, du bas, de la gauche.</li> </ul>


## Bordures

Type	Propriété	Valeurs possibles
Épaisseur de la bordure	border-width	Indiquer une valeur en px.
Couleur de la bordure	border-color	Indiquer une valeur de couleur.
Type de bordure	border-style	<p><b>none</b> : pas de bordure (par défaut)</p> <p><b>hidden</b> : bordure cachée</p> <p><b>solid</b> : ligne pleine</p> <p><b>double</b> : ligne double (nécessite une taille de bordure de 3px minimum)</p> <p><b>dashed</b> : en tirets</p> <p><b>dotted</b> : en pointillés</p> <p><b>inset</b> : effet 3D "enfoncé"</p> <p><b>outset</b> : effet 3D "surélevé"</p> <p><b>ridge</b> : autre effet 3D</p>
Bordure à gauche	border-left	<p>Indiquer la couleur, l'épaisseur et le type de bordure pour la bordure gauche. L'ordre n'a pas d'importance. Exemple :</p> <p><b>Code : CSS</b></p> <pre>border-left: 2px inset blue; /* Bordure bleue de 2px avec effet 3D "enfoncé" */</pre>

Bordure en haut	border-top	Idem
Bordure à droite	border-right	Idem
Bordure en bas	border-bottom	Idem
Méga-propriété de bordure	border	Indiquera l'apparence des bordures en haut, à droite, en bas et à gauche.

## Propriétés de positionnement et d'affichage

### Affichage

Type	Propriété	Valeurs possibles
Type d'élément	display	<b>none</b> : l'élément ne sera pas affiché <b>block</b> : l'élément devient de type "block" (bloc, comme <p>) <b>inline</b> : l'élément devient de type "inline" (en ligne, comme <strong>) <b>list-item</b> : l'élément devient de type "élément de liste à puce" (comme <li>)
Affichage	visibility	<b>hidden</b> : masqué <b>visible</b> : visible (par défaut)  <b>display:none</b> ; fait complètement disparaître l'élément, tandis que <b>visibility:hidden</b> ; masque l'élément, qui continue quand même à prendre de la place sur l'écran.
Afficher seulement une partie	clip	Indiquer 4 valeurs comme ceci :  <b>Code : CSS</b> <pre>clip: rect (valeur1, valeur2, valeur3, valeur4) ;</pre> Cela permet de n'afficher qu'une partie d'un élément. rect() permet d'indiquer les coordonnées du rectangle qui sera affiché. Les valeurs 1 à 4 correspondent respectivement aux coins haut, droite, bas et gauche du rectangle.
Limiter les dimensions	overflow	<b>visible</b> : tout l'élément sera affiché (par défaut). <b>hidden</b> : l'élément sera coupé s'il dépasse les limites définies par height et width. On ne pourra pas voir la partie du texte coupée. <b>scroll</b> : tout comme hidden, l'élément sera coupé s'il dépasse les limites. Toutefois, cette fois le navigateur ajoutera des barres de défilement pour qu'on puisse voir la suite du texte. <b>auto</b> : c'est le navigateur qui décide d'ajouter des barres de défilement ou pas en fonction des cas. Bien souvent, utiliser cette valeur revient à utiliser la valeur "scroll".

### Positionnement

Type	Propriété	Valeurs possibles
Flottant	float	<b>left</b> : flottant à gauche <b>right</b> : flottant à droite <b>none</b> : pas de flottant (par défaut)
Stopper un flottant	clear	<b>left</b> : supprime l'effet d'un flottant à gauche précédent <b>right</b> : supprime l'effet d'un flottant à droite précédent <b>both</b> : supprime l'effet d'un flottant précédent, qu'il soit à gauche ou à droite <b>none</b> : pas de suppression de l'effet du flottant (par défaut)
Type de positionnement		<b>absolute</b> : position absolue par rapport au coin en haut à gauche <b>fixed</b> : position fixe (fonctionne comme la position absolue). L'élément reste à sa position

type de positionnement	position	même quand on descend plus bas dans la page. <b>relative</b> : position relative, par rapport à la position "normale" de l'élément <b>static</b> : positionnement normal (par défaut)
Position par rapport au haut	top	Valeur en px, %, em... A utiliser pour un positionnement absolu, fixe ou relatif.
Position par rapport au bas	bottom	Idem
Position par rapport à gauche	left	Idem
Position par rapport à droite	right	Idem
Ordre d'affichage	z-index	En cas de positionnement absolu par exemple, si 2 éléments se chevauchent, z-index permet d'indiquer quel élément doit être affiché au-dessus de l'autre. Indiquez un nombre. Plus ce nombre est élevé, plus l'élément sera affiché en avant.  Par exemple, si vous avez 2 éléments positionnés en absolus avec un z-index de 10 pour l'un et de 20 pour l'autre, c'est celui qui a un z-index de 20 qui sera affiché par-dessus.

## Propriétés des listes

Type	Propriété	Valeurs possibles
Type de liste	list-style-type	<ul style="list-style-type: none"> <li>Pour les listes non ordonnées (&lt;ul&gt;) :               <ul style="list-style-type: none"> <li><a href="#">disc</a> : un disque noir (par défaut).</li> <li><a href="#">circle</a> : un cercle.</li> <li><a href="#">square</a> : un carré.</li> <li><a href="#">none</a> : aucune puce ne sera utilisée.</li> </ul> </li> <li>Pour les listes ordonnées (&lt;ol&gt;) :               <ul style="list-style-type: none"> <li><a href="#">decimal</a> : des nombres 1, 2, 3, 4, 5... (par défaut)</li> <li><a href="#">decimal-leading-zero</a> : des nombres commençant par zéro (01, 02, 03, 04, 05...).</li> <li><a href="#">upper-roman</a> : numérotation romaine, en majuscules (I, II, III, IV, V...)</li> <li><a href="#">lower-roman</a> : numérotation romaine, en minuscules (i, ii, iii, iv, v...)</li> <li><a href="#">upper-alpha</a> : numérotation alphabétique, en majuscules (A, B, C, D, E...)</li> <li><a href="#">lower-alpha</a> : numérotation alphabétique, en minuscules (a, b, c, d, e...)</li> <li><a href="#">lower-greek</a> : numérotation grecque.</li> </ul> </li> </ul>
Position en retrait	list-style-position	<b>inside</b> : sans retrait <b>outside</b> : avec retrait (par défaut)
Puce personnalisée	list-style-image	Indiquer l'url de l'image qui servira de puce. Exemple :  <b>Code : CSS</b> <pre>list-style-image: url("images/puce.png");</pre>
Méga-propriété de liste	list-style	Vous pouvez réunir les valeurs de list-style-type, list-style-position et list-style-image. Vous n'êtes pas obligés de mettre toutes les valeurs, et l'ordre n'a pas d'importance. Exemple :  <b>Code : CSS</b> <pre>list-style: inside square;</pre>

## Propriétés des tableaux

Type	Propriété	Valeurs possibles
Type de bordure	border-collapse	<b>collapse</b> : les bordures du tableau et des cellules sont mélangées. <b>separate</b> : les bordures du tableau et des cellules sont séparées (par défaut).
Cellules vides	empty-cells	<b>show</b> : les bordures des cellules vides sont affichées. <b>collapse</b> : les cellules vides sont masquées (par défaut).
Position du titre	caption-side	Indique la position du titre du tableau, défini via la balise <caption>  <b>top</b> : en haut du tableau <b>bottom</b> : en bas du tableau <b>left</b> : à gauche du tableau <b>right</b> : à droite du tableau

## Autres propriétés

Type	Propriété	Valeurs possibles
Curseur de souris	cursor	<p> <b>auto</b> : curseur automatique (par défaut)  <b>default</b> : curseur standard  <b>pointer</b> : curseur en forme de main, comme quand on pointe sur un lien  <b>text</b> : curseur utilisé quand on pointe sur du texte  <b>wait</b> : curseur utilisé pour indiquer une attente (sablier)  <b>progress</b> : curseur utilisé pour indiquer une tâche de fond (curseur avec sablier)  <b>help</b> : curseur en forme de point d'interrogation, indiquant une aide  <b>move</b> : curseur en forme de croix, indiquant un déplacement possible </p> <p> <b>n-resize</b> : flèche vers le nord  <b>ne-resize</b> : flèche vers le nord-est  <b>e-resize</b> : flèche vers l'est  <b>se-resize</b> : flèche vers le sud-est  <b>s-resize</b> : flèche vers le sud  <b>sw-resize</b> : flèche vers le sud-ouest  <b>w-resize</b> : flèche vers l'ouest  <b>nw-resize</b> : flèche vers le nord-ouest </p> <p> <b>url</b> : curseur personnalisé, de type .cur ou .ani. Exemple : </p> <p><b>Code : CSS</b></p> <pre>cursor: url("images/curseur.cur"), auto;</pre> <p>           Vous devez utiliser un logiciel dédié à la création de curseurs pour créer des .cur et des .ani. Notez aussi la présence du mot <b>auto</b> à la fin. Cela signifie que le navigateur tentera de charger le curseur personnalisé. S'il n'a pu être chargé pour une quelconque raison, le curseur correspondant à la valeur <b>auto</b> sera utilisé. </p>

Pfiou !

Recenser les propriétés CSS n'est ni reposant, ni amusant je vous l'assure !

Les plus experts d'entre vous auront remarqué que je n'ai pas mis toutes les propriétés CSS, comme je l'ai dit au début. En effet, mon but n'était pas de faire la liste complète mais plutôt de vous fournir un support lorsque vous codez en CSS.

J'ai donc conservé les propriétés CSS qui me paraissaient les plus souvent utilisées.

Si je trouve le temps, ou plutôt le courage, de rajouter de nouvelles propriétés CSS dans le futur, j'éditerai ce chapitre.

Toutefois, je pense que vous aurez déjà de quoi faire avec ce qui est déjà là 😊

Le cours s'arrête là !

Si cela vous intéresse, sachez que j'ai écrit un [livre](#) sur le même sujet paru aux éditions Eyrolles. Cela peut vous servir de support papier si la lecture à l'écran vous fatigue, et c'est aussi une façon de me soutenir si le cours vous a plu 😊